Technical Report 1306

The Kineticist's Workbench: Combining Symbolic and Numerical Methods in the Simulation of Chemical Reaction Mechanisms

Michael A. Eisenberg

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

The Kineticist's Workbench:
Combining Symbolic and Numerical Methods in the
Simulation of Chemical Reaction Mechanisms

by

Michael Allen Eisenberg

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 1991 in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Abstract

The Kineticist's Workbench is a program that expands the expressiveness of computer simulation: it combines symbolic and numerical techniques in simulating a particular class of complex systems—chemical reaction mechanisms. The Workbench assists chemists by predicting, generating, and interpreting numerical data. Prior to simulation, it analyzes a given mechanism to predict that mechanism's behavior; it then simulates the mechanism numerically; and afterward, it interprets and summarizes the data that it has generated. In performing these tasks, the Workbench brings to bear a wide variety of techniques: graph-theoretic algorithms (for the analysis of mechanisms), traditional numerical simulation methods, and algorithms that examine the simulation results and reinterpret them in qualitative terms. Moreover, the Workbench can use symbolic procedures to help guide or simplify the task of numerical simulation; and it can sometimes use its summary of numerical results to suggest additional numerical analysis. Thus, it serves as a prototype for a new class of scientific computational tools—tools that provide symbiotic collaborations between qualitative and quantitative methods.

Keywords:
Chemical kinetics, symbolic/numerical simulation

Thesis Supervisor: Harold Abelson
Title: Associate Professor of Computer Science and Engineering

This empty page was substituted for a blank page in the original document.

Acknowledgments

It has been the greatest good fortune of my life to have known so many thoroughly amazing people at MIT; and the poorest fortune to have so little space in which to thank them. Foremost to acknowledge is Hal Abelson, who more than anyone else has shaped my intellectual life over the past decade. I really don't think I can convey in words the value of the education that I have received from Hal. As a scholar, mentor, and person, he has taught me how to think, and he has my gratitude.

Gerald Sussman has likewise been a teacher and role model. Whether speaking of programming or astrophysics or politics, he is a pleasure; and he is living proof that intellectual integrity and a sense of fun can be combined in one person.

Thanks also to Andy diSessa (now at Berkeley), for teaching me that programming can be a means of expression.

Mitchel Resnick and Franklyn Turbak have been a joy to know, to talk with, to collaborate with. I want to thank them for sharing their ideas, their companionship, and (not least) their pioneering efforts in recursive humor. Likewise, thanks to the other members of our lab group: Andy Berlin, Mark Friedman, Arthur Gleckler, Chris Hanson, Brian LaMacchia, Jonathan Rees, Bill Rozas, Pete Skordos, Henry Wu, and Feng Zhao. The phrase "gentleman and scholar" is used these days in a half-humorous way; but these men really *are* scholars and gentlemen all.

Various officemates over the years have exhibited superhuman tolerance in dealing with me: Uri Leron, Henry Lieberman, Linda (and Jennifer) Morecroft, Michelle Lee, Joe Marshall. A special acknowledgment to the two most recent colleagues in this history: Liz Bradley and Orca Starbuck. Liz Bradley is so accomplished that when I describe her to people outside MIT, they think I'm inventing her. Besides that, she is the only person I know who has a boat named after her. Orca Starbuck, self-effacing and eerily multitalented, has lately been a wondrous collaborator in making "Schemepaint" pictures.

Profs. Ramesh Patil (of USC) and Irving Epstein (of Brandeis) both generously agreed to be readers for this thesis, and have shown a much-appreciated

patience in waiting for it to be completed. I want to thank them for their time and advice. I also want to acknowledge Prof. Epstein's assistance in pointing me toward important source material for implementing the graphical-analysis portion of the Workbench.

There are many others whose conversation and influence have been invaluable; so varied and distinctive are these people that it would be hopeless to acknowledge them appropriately in a book, much less a single page. Among these individuals, my special thanks to Edith Ackermann, Barry Dworkin, Wallace Feurzeig, Sherry Fine, Paul Horwitz, Jim Miller, Seymour Papert, Roy Pea, William Siebert, Fran Streeter, and Ken Yip.

My financial survival at MIT has been aided immeasurably by Bell Labs, by Paul Kelly at the Scientific Press, and by Al Moyé at Hewlett-Packard; they have my thanks.

Finally, my love to my parents and sister. And gratitude to five friends who made me laugh and preserved my perspective over the past decade. This thesis is dedicated to those friends: Peter Basch, Peter Finder, Adam Greenberg, Toby Muller, and (the late) Steve Werner.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-89-J-3202.

Contents

1. The Kineticist's Workbench: an Introduction 1
1.1 A Versatile Simulator for Chemical Kinetics 2
1.2 Design Principles of the Kineticist's Workbench 3
1.3 Outline of this Paper 5

2. Kinetics: The Nature of the Problem 6
2.1 Chemical Reactions and Chemical Mechanisms 6
2.1.1 Mechanisms 6
2.1.2 Mechanisms as Systems of Differential Equations 9
2.1.3 Special Assumptions in the Treatment of Mechanisms 11
2.1.4 Sources of Complexity in Mechanisms 15
2.1.5 Additional Complexities 16
2.2 Strategies for Simplifying Mechansims 17
2.2.1 Rapid Equilibrium 18
2.2.2 Steady-State Assumption 19
2.3 Numerical Simulation of Mechanisms 22
2.3.1 Stiff Systems 23
2.3.2 Stochastic Simulation 24
2.3.3 Limitations of Numerical Simulation 25
2.4 Why Kinetics Matters 28

3. The Kineticist's Workbench: A Design Overview 32
3.1 Numerical Simulation 32
3.2 Pre-Simulation Analysis of Mechanisms 34
3.3 Post-Simulation Interpretation of Mechanisms 35
3.4 The Workbench as a Whole 37

4. Numerical Simulation in the Workbench 39
4.1 The Mechanism Data Structure 39
4.2 Simulating a Mechanism with the Runge-Kutta Integrator 42
4.3 Simulating a Mechanism with the Gear Integrator 44

4.4 Graphing Numeric Results 47
4.5 Parametrized Mechanisms 53

5. Analyzing Mechanisms Before Simulation 57
5.1 Graphical Analysis 57
5.1.1 Structural Information from Mechanisms 57
5.1.2 Necessary Nonzero Species 60
5.1.3 Obvious Declining Sets of Species 61
5.1.4 Consistent Zero Sets of Species 62
5.1.5 The Zero Deficiency Theorem 63
5.1.6 Submechanisms 70
5.1.7 Examples with the Workbench 71
5.1.8 Special Structural Features of Mechanisms 81
5.2 Numerical/Graphical Analysis 85
5.2.1 Fast Equilibria and Fast Submechanisms 86
5.2.2 Steady State Candidates 93
5.2.3 A Note on Qualitative Arithmetic 96
5.3 Rapid Location of Equilibria 99
5.4 Notating Mechanism Objects with Predictions 106
5.5 Mechanisms not Handled by Graphical Analysis 107

6. Analyzing the Numbers 110
6.1 Constructing Episode Histories of Simulations 111
6.1.1 The Episode Data Structure 111
6.1.2 Episode Histories 114
6.1.3 Initial Processing of the Episode History 116
6.1.4 Attaching Features to Episodes 118
6.1.5 A Brief Example 120
6.1.6 Looking for Patterns in Episode Histories 124
6.2 Zero-Crossings of Derivatives 129
6.2.1 The Zero-Crossing Data Structure 129
6.2.2 Looking for Oscillation Patterns 130
6.2.3 Coarse- and Fine-Grained Oscillations 132

6.3 Constructing Feature Summaries of Simulations 132
6.4 Saving Qualitative Analysis Results; Parameter Space Checks 134
6.5 Checking Predictions 136
6.6 Additional Analysis 137

7. The Workbench in Operation 140
7.1 The "Tiny Mechanism" 140
7.2 N205 Decomposition 145
7.3 Enzyme Kinetics 150
7.4 Brusselator 154
7.5 Oregonator 162
7.6 Rössler Band System 167
7.7 Chlorite-Iodide Oscillator 169

8. Related and Future Work 176
8.1 Related Work 176
8.1.1 Chemistry Simulators 176
8.1.2 Qualitative Physics 178
8.1.3 Combining Symbolic and Numerical Computation 181
8.2 Philosophical Reflections I 183
8.2.1 Using Qualitative Interpretation to Guide Numerical Work 186
8.2.2 Using Qualitative Interpretation to Guide Data Management 188
8.2.3 Mixing Qualitative and Quantitative Representations 189
8.3 Problems with the Workbench; Future Work 191
8.3.1 Speed and Range of the Workbench 191
8.3.2 Integration of Various Modules 192
8.3.3 Improvements in Qualitative Analyses 194
8.3.4 More Complex Systems 195
8.3.5 Interface 196
8.4 Philosophical Reflections II 197

References 201

Appendix A: The Zero-Deficiency Theorem 212

Appendix B: Algorithms for Graphical Analysis 217
B.1 Necessary Nonzero Species 217
B.2 Obvious Declining Sets of Species 217
B.3 Consistent Zero Sets of Species 219
B.4 Catalytic Pathways 220
B.5 Autocatalytic Pathways 221
B.6 Qualitative Arithmetic 222
B.7 Fast Submechanisms 223
B.8 Steady State Candidates 224
B.9 Rapid Location of Equilibria 225

Appendix C: Algorithms for Qualitative Analysis 229
C.1 Attaching Feature Descriptors to Episodes 229
C.2 Repeating Patterns in Episodes 231
C.3 Grouping Zero Crossings into Oscillations 232
C.4 Classifying Oscillation Types 234
C.5 Feature Summaries 236
C.6 Coarse-Grained Zero Crossings 238

List of Figures

2.1 A Laboratory System with Sources and Sinks 13
2.2 A Parameter Space Graph 27

3.1 A Graph of Three Concentrations Versus Time 33
3.2 A Graph of a Mechanism 34
3.3 A Parameter-Space Graph 36
3.4 The Workbench Program Structure 37

4.1 Graph of an Oregonator Run 49
4.2 Graph Region for Species X 50
4.3 Graph of Tiny Mechanism Run 53

5.1 A Graph of Mechanism [12] 73
5.2 Dropped Species in Mechanism [12] 75
5.3 Graph of N205 Decomposition 76
5.4 One Zero-Set in N205 Decomposition 77
5.5 Second Zero-Set in N205 Decomposition 77
5.6 Brusselator Graph 79

6.1 Simulating Mechanism [4.1]-[4.3] 122
6.2 Brusselator Simulation 125

7.1 Graph of "Tiny Mechanism" 141
7.2 Simulation of "Tiny Mechanism" 144
7.3 Simulation of N205 Decomposition 147
7.4 Enzyme Simulation ([S]o = 1M) 152
7.5 Enzyme Simulation ([S]o = 10M) 153
7.6 Final [P] vs. Initial [S] 154
7.7 Brusselator ($k3 = 1.5$ $k4 = 1.0$) 158
7.8 Brusselator ($k3 = 3.5$ $k4 = 1.5$) 158
7.9 Brusselator ($k3 = 1.5$ $k4 = 0.5$) 159

7.10 Brusselator Parameter Space Graph 160
7.11 Oregonator ($k5 = 60$, $f = 0.4$) 163
7.12 Oregonator ($k5 = 60$, $f = 0.75$) 163
7.13 Oregonator ($k5 = 60$, $f = 1.1$) 164
7.14 Oregonator Parameter Space Graph 166
7.15 Oregonator Steady-State Stability 166
7.16 The Rössler Band 167
7.17 Simulation of Rössler System 168
7.18 Chlorite-Iodide Oscillator Simulation 172
7.19 Two Power Spectra of I- Results 174

Chapter 1

The Kineticist's Workbench: an Introduction

Computers are marvelous numerical simulators. In virtually every field of science, their influence in that role is profound. Colliding galaxies, evolving ecosystems, turbulent fluids—all have been the subject of extensive computational simulation.

But when we say that a computer is "simulating" a complex system, we are generally speaking only of numbers: a mathematical model is provided to the machine (usually in the form of differential equations), and the machine returns a table (or graph) of numbers as its output. This is useful, but a great deal of conceptual work is still left to the scientist. After all, the numbers must be interpreted: the scientist examines the numerical record for interesting phenomena, perhaps matching those phenomena to observations from the real world. Moreover, the numbers must be viewed in relation to the model that generated them: why did *this* model produce *these* results? And before the computer is employed altogether, the scientist may spend time analyzing the model, reasoning about its potential behavior, trying to simplify it. From the human standpoint, then, "simulation" is much more than number-crunching; but from the computational standpoint, this is usually all that the term ever means.

Indeed, the very real successes of computational number-crunching have resulted in the need for more "extra-numerical" work. The scientist is left, in effect, with a devil's bargain: it is easier than ever before to generate data, but much harder to obtain computational assistance in predicting and interpreting that data.

This paper describes a program, *The Kineticist's Workbench*, that is a "simulator" in the richer sense of the term. Specifically, the Workbench predicts, generates, and interprets numerical data in the simulation of a particular class of complex systems—chemical reaction mechanisms. It as-

sists the scientist prior to numerical simulation by performing analysis of mechanisms to predict their potential behavior; it simulates the mechanism; and afterward, it interprets and summarizes the data that it has generated. In performing these tasks, the program brings to bear a wide range of symbolic and numerical techniques, including graph-theoretic algorithms (for the analysis of mechanisms), traditional numerical simulation methods, and algorithms that examine the simulation results and reinterpret them in qualitative terms. Moreover, beyond its conception as a scientific tool, the Workbench represents a salutary exercise in formalizing a number of scientific reasoning techniques that are often left unspecified or implicit in textbooks.

1.1 A Versatile Simulator for Chemical Kinetics

Chemical reactions are, as it happens, surprisingly complicated events. A "simple" reaction might appear in a textbook as a direct transition from reactant molecules to product molecules; but in actuality, that simple notation is often just a summary of the combined action of many simpler, "elementary" chemical steps. These elementary steps—typically, unimolecular decompositons and bimolecular collisions—together constitute a *mechanism* for the overall reaction. By constructing and testing hypothetical mechanisms, a chemist can develop an understanding of some net, overall reaction of interest. Put this way, the job may sound easy enough; but because even small mechanisms can give rise to complicated behavior, the task of formulating and understanding such chemical models is in fact extremely difficult.

In the current state of the art, chemists make extensive use of computers to study the kinetics of the hypothetical mechanisms they construct. Typically—in accordance with the scenario described above—the chemist will translate a mechanism into a system of differential equations, and will then use the computer as a high-speed numerical integrator. The machine produces huge amounts of numerical output (perhaps in graphical form), representing the concentrations of various species over time; and it is then the chemist's job to interpret these numbers. The chemist may look for interesting patterns (such as the presence of steady states or oscillations), and may vary parameters in the mechanism to see how these patterns are affected by the changes in parameter values.

Although this kind of computational work has become an indispensable part of the kineticist's repertoire, it provides only a narrow sort of assistance. More important, it fails to take advantage of the full range of the computer's capabilities. Much of the chemist's interpretive work—spotting rapid jumps in concentrations, for instance—is relatively straightforward and can be performed by the computer. Some of the non-numerical work that precedes simulation—e.g., deciding whether a mechanism is capable of "exotic" behavior like oscillations—can likewise be performed or assisted by the computer. And beyond this, the ways in which these modes of activity *interact*—the ways, for instance, in which a structural understanding of a mechanism can affect the interpretation of results—are capable of computational expression. In short, a true "kinetics simulator" can and should do much more than print out numbers.

The Kineticist's Workbench is designed in pursuit of this idea—a program that fruitfully extends the range of computational work in chemical simulation. It includes procedures that allow the chemist to examine a given mechanism for special (potentially simplifying) properties, to simulate the mechanism numerically, and to summarize the results of a simulation qualitatively. In many instances, the Workbench can use symbolic techniques to help guide or simplify the task of numerical simulation; and it can sometimes use its qualitative summary of numerical results to suggest additional numerical analysis of those results. Thus, the Workbench is designed not merely as a collection of distinct subprograms, but with an eye toward symbiotic connections between symbolic and numerical techniques.

The Workbench is a prototype only. Even so, it illustrates how a new class of computational tool can assist scientists in simulating, analyzing, and understanding complex systems; and it provides a study in the extension of standard integration algorithms with a spectrum of additional symbolic and numerical techniques.

1.2 Design Principles of the Kineticist's Workbench

The notion of multiple cooperating techniques is the basic design principle of the Workbench system. In part, this principle is thrust upon the

program by the very nature of its domain. As noted, chemical mechanisms are complex systems, and lend themselves to a multitude of different styles of analysis. Sometimes the chemist tries to simplify a mechanism before simulation, by looking for particular features in the mechanism itself; sometimes a simplification (or numerical approximation) is deemed possible based on numerical results; sometimes the mechanism must be altered because some feature of its simulated behavior is inconsistent with laboratory results. The root of this complexity is that chemical mechanisms are nonlinear systems of ordinary differential equations, and thus can exhibit all the myriad behaviors of which nonlinear systems are capable.

The Workbench correspondingly avoids promoting a single all-purpose computational formalism for analyzing mechanisms, as such a formalism would inevitably misrepresent the opportunistic spirit in which kinetics is actually performed. Rather, the program includes multiple representations of mechanisms (both as differential equations and as certain types of graphs); it has a variety of "special case" procedures appropriate to mechanisms with special simplifying properties (e.g., mechanisms described by linear differential equations); it is written in Scheme (a Lisp dialect), thus facilitating the interactive development of the program via the addition of new procedures. Overall, the Workbench can be used to make powerful deductions about the behavior of moderately simple mechanisms, but can also be used to study recently-developed complex models.

An important theme in the design of the Workbench is the augmentation, not replacement, of numerical simulation. The Workbench does not try to obviate the need for numerical simulation by developing a purely qualitative formalism for the behavior of chemical mechanisms; its aim is rather to use symbolic techniques in ways that help to predict and interpret numerical results. In this sense the Workbench is to be distinguished from more explicitly non-numerical efforts in the field known as *qualitative physics.*[1]

The fact that numbers are the primary data for the Workbench is related to another design principle of the system—namely, that the techniques and terminology be derived from the domain itself. The Workbench uses concepts like "rapid equilibrium," "autocatalysis," and so forth in examining

[1]Further discussion of this comparison appears in Chapter 8.

mechanisms for special features; likewise, it produces qualitative summaries of simulations employing terms like "steady state," "stable oscillations," and so forth. There is a concerted attempt throughout to stay close to the language and concepts of chemical kinetics, so that the system can ultimately be of use to working chemists.

Finally, the Workbench is designed so that its various modules produce data in a form that is usable by other computer programs. That is, the Workbench is not specifically concerned with "data visualization" in the usual (human interface) sense of that term. The Workbench does produce tables, graphs, and textual printout for the user, but it also ensures that its results may be passed to other procedures. This is an important corollary of the notion of multiple cooperating techniques mentioned earlier: the idea is that the various portions of the program should be designed to take advantage of the possibility of sending useful information to other programs.

1.3 Outline of This Paper

The following chapter lays the groundwork for an examination of the Kineticist's Workbench by presenting a (somewhat telegraphic) discussion of chemical kinetics. This chapter will introduce the basic terminology of the field, and will also motivate some of the specific features of the Workbench system. The third chapter is devoted to an overview of the Workbench's implementation; and the three following chapters (4-6) expand on this overview by examining the three major modules of the Workbench. Chapter 7 is devoted to a number of examples showing the Workbench in operation. The final chapter is a discussion of related work, as well as problems in the current Workbench system and prospects for future development.

Chapter 2

Kinetics: The Nature of the Problem

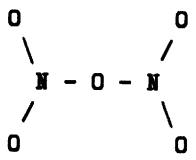
2.1 Chemical Reactions and Chemical Mechanisms

2.1.1 Mechanisms

The textbook picture of a chemical reaction is usually that of a direct transition from reactants to products, as in the following example:

```
[1]     2 N2O5    -->  4 NO2  +  O2
```

It would appear from this notation as though the reaction [1] proceeds by the collision of two molecules of nitrogen pentoxide (N2O5), causing a rearrangement into five product molecules. In point of fact, however, this scenario is extremely unlikely on geometric grounds alone. A rough (planar) picture of the N2O5 molecule looks as follows {58}:



Looking at this molecule, it is hard to imagine how such an intricate rearrangement of atoms could possibly take place as the result of a straightforward collision. And there is another, even more disturbing fact: the rate of collision of two N2O5 molecules in gas phase is proportional to the square of N2O5 concentration (at least for dilute concentrations).{58} Thus, we might expect that if we double the concentration of N2O5, the initial rate of appearance of O2 molecules will increase by a factor of 4:[1]

[1]The concentration of a given species such as O2 is written [O2]. Throughout this

$$[2] \qquad \left(\frac{d[O_2]}{dt}\right)_{t=0} \propto [N_2O_5]^2$$

The problem with [2] is that it is contradicted by experimental evidence. In point of fact, the rate of initial production of O_2 molecules is proportional to the concentration of N_2O_5:{55}

$$[3] \qquad \left(\frac{d[O_2]}{dt}\right)_{t=0} \propto [N_2O_5]$$

It would thus appear that the simple scenario suggested by the notation of the overall rearrangement [1] does not adequately describe the events of the actual chemical reaction. How, then, does reaction [1] take place? To answer this question, the chemist must propose a model, or *mechanism*, for the reaction: a sequence of two-molecule collisions (or occasionally single-molecule decompositions). Each individual step in this mechanism should be chemically plausible, and taken together the steps should account for both the overall reaction and experimental data such as [3] above.

A plausible mechanism for nitrogen pentoxide decomposition is the following:{55}

$$[4.1] \qquad N_2O_5 \longrightarrow NO_2 + NO_3$$

$$[4.2] \qquad NO_2 + NO_3 \longrightarrow N_2O_5$$

$$[4.3] \qquad NO_2 + NO_3 \longrightarrow NO_2 + NO + O_2$$

$$[4.4] \qquad NO + N_2O_5 \longrightarrow 3NO_2$$

---

paper we will assume that the units of concentration are moles/liter (this is a common choice; another choice of units often encountered in the literature is molecules/cc).{74}

In contrast to the original reaction [1], each step in the set [4.1]–[4.4] is intended to be read as a direct transition from reactants to products. To introduce some standard terminology, each of these four reactions is referred to as an *elementary step* (indicating that it proceeds directly "as written," unlike [1]). Reaction [4.1] is a *unimolecular* step, while the other three are *bimolecular*.[2]

It should be stressed that [4.1]–[4.4], like any mechanism, is only a hypothesis—often, the elementary steps that a chemist proposes cannot be observed directly in the laboratory. Mechanisms may be disproven in numerous ways: by showing that they contradict experimental data, or that a proposed step does not occur, or that some unanticipated step does indeed occur. Mechanisms cannot, however, ever be said to be proven.{55}

Without going into detail, we can accept that [4.1]–[4.4] is chemically reasonable—i.e., that each elementary step could conceivably occur via collisions and plausible rearrangements of atoms. The kind of chemical knowledge that goes into proposing such a mechanism, and into verifying that each step is plausible (if not observable), is often profound; but that purely chemical knowledge is not at issue in this paper. Rather, our interest is in the next step—verifying that the mechanism [4.1]–[4.4] is capable of accounting for experimental results such as [3] above. In other words, we want to know how our proposed mechanism *behaves*—how it accounts for the changing concentrations over time of each of its constituent species. If our mechanism is incapable of accounting for [3], then it cannot be correct, regardless of how sensible each step appears; perhaps we have ignored a step that actually occurs, or included one that doesn't.

What we need, then, is a method for translating a mechanism such as [4.1]–[4.4] into a mathematical model that allows us to predict the concentration profiles (over time) of all the observable species. We can then compare the results of our model to experimental data such as [3], and corroborate (or disprove) our mechanism. The next subsection examines this process of translating mechanisms into predictive models.

[2]On rare occasions, a three-molecule collision or *termolecular* step will be proposed.

2.1.2 Mechanisms as Systems of Differential Equations

We can begin translating [4.1]–[4.4] into a mathematical model by using the fact—alluded to earlier—that the rate of collision of two molecules of type A and B is proportional to the product of their concentrations [A] [B]. Thus, if we restrict our attention to reaction [4.2] alone, we can write:

$$[5] \qquad -\frac{d[NO2]}{dt} = -\frac{d[NO3]}{dt} = \frac{d[N2O5]}{dt} = k2\,[NO2]\,[NO3]$$

The first two relations follow from the stoichiometry of reaction [4.2] (one mole of NO2 is consumed in this reaction for every mole of NO3 consumed and N2O5 produced). The last relation expresses the fact that the rate of production of N2O5 is proportional to the collision rate of NO2 and NO3 molecules. Here, $k2$ is a proportionality constant, or *rate constant*, to use the standard terminology. Specifically, $k2$ is a *second-order* rate constant (corresponding to a bimolecular step), and its units are concentration^{-1} * time^{-1}.

As for the unimolecular step [4.1], we assume that at any given moment, the rate of reaction is proportional to the concentration of the reactant.[3] Thus, for this step we can write:

[3]There are a number of theoretical models that account for this treatment of unimolecular elementary steps. One of the more popular such theories, the Rice-Ramsperger-Kassel (or RRK) theory, actually treats unimolecular steps as "shorthand" for a collision step followed by reaction of an activated species.{55} The RRK theory partially accounts for the experimental limitations of the simple treatment embodied in [6]. For our purposes, we will not concern ourselves with the "internals" of unimolecular steps, and will assume that [6] is correct. As a heuristic explanation for this "standard" kinetic treatment, it may be simplest to imagine that for a decomposition or unimolecular rearrangement of some species A, the reaction occurs in a probabilistic fashion—e.g., for all molecules of A above a certain level of internal vibrational energy, or for all molecules of A occupying a special conformation. It is reasonable to assume that at any given time a constant percentage of species A will be able to undergo reaction: hence expression [6] above. As a final aside, it may also be worth mentioning that this treatment of unimolecular reactions is identical to that of radioactive decay reactions.{58}

$$[6] \qquad \frac{-d[N_2O_5]}{dt} = \frac{d[NO_2]}{dt} = \frac{d[NO_3]}{dt} = k1\,[N_2O_5]$$

Again, the first two relations are derived from the stoichiometry of [4.1], and the last relation indicates that this step obeys first-order kinetics. (The units of the first-order rate constant $k1$ are time^{-1}.)

We can now combine the expressions derived from [4.1] and [4.2] with analogous expressions derived from the other two steps, and arrive at the following system of ordinary differential equations:

$$[7.1] \quad d[N_2O_5]/dt = -k1\,[N_2O_5] + k2\,[NO_2][NO_3] - k4\,[NO][N_2O_5]$$

$$[7.2] \quad d[NO_2]/dt = k1\,[N_2O_5] - k2\,[NO_2][NO_3] + 3\,k4\,[NO][N_2O_5]$$

$$[7.3] \quad d[NO_3]/dt = k1\,[N_2O_5] - k2\,[NO_2][NO_3] - k3\,[NO_2][NO_3]$$

$$[7.4] \quad d[NO]/dt = k3\,[NO_2][NO_3] - k4\,[NO][N_2O_5]$$

$$[7.5] \quad d[O_2]/dt = k3\,[NO_2][NO_3]$$

These five equations [7.1]–[7.5] constitute a translation of the mechanism [4.1]–[4.4] into a mathematical model. For each species in the mechanism, we have a differential equation that dictates how the concentration of that species changes with time; and each equation consists of terms derived from steps in the mechanism in which that species is produced or consumed. (For

example, the three terms in equation [7.1] are derived from steps [4.1], [4.2], and [4.4] respectively.)

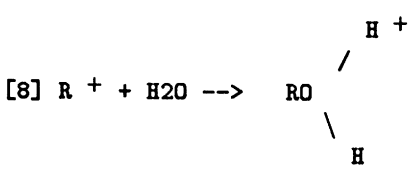
Our mathematical model is still not quite complete, because we have yet to choose numerical values for the rate constants k1-k4. Sometimes these constants may be determined by direct laboratory observation of a particular elementary step; often they are treated as unknown parameters within the model. In any event, once rate constants (and initial concentrations) are chosen, equations [7.1]–[7.5] will determine the subsequent concentration profiles of all species. These predictions may then be compared with experimental data (such as the earlier expression [3]) to corroborate the original mechanism.

2.1.3 Special Assumptions in the Treatment of Mechanisms

The previous discussion illustrated the essential process of translating a chemical mechanism into differential equations. There are, however, several additional points that deserve mention.

First, it is often the case that a particular species within a mechanism is assumed to have constant concentration. This is superficially something of a paradox: how can a given species enter into a reaction (or group of reactions) and yet have unchanging concentration? The answer is that in some cases, the assumption of constant concentration is an excellent approximation that simplifies the treatment and understanding of the mechanism.

A typical example of this sort of simplification occurs for reactions in solution in which the solvent enters into the reaction. For instance, consider the following the elementary step (taken from a mechanism for hydrolysis of alkyl halides {9}):



Here, the reaction is taking place in water solution, so the concentration of H2O may be assumed constant for the entire course of the reaction.

More generally, the assumption of constant concentration is often used when a particular species is present in such excess that its concentration can be affected very little by any reactions into which it may enter. (The case of "constant solvent concentration" may be viewed as a special case of this principle.) Occasionally, it may be that a laboratory system is prepared in which a particular species is kept at (approximately) constant concentration by external control.

A second common assumption is that a mechanism has external "sources" or "sinks" for particular species. We might, for example, stipulate that some species A is being fed into a reactive system at a given rate; in this case there is an external source for A (independent of its concentration within the reactive system). Our differential equation for A would then have a "source" term:

$$[9] \qquad \frac{d[A]}{dt} = k_{src} + \textit{other terms}$$

Here, the first term k_{src} is a constant (one might think of it as a "zero-th order rate constant") that indicates that species A is being added to the system at a constant rate.

A "sink" for a given species B may be viewed conceptually as an elementary reaction of the form:

$$[10] \qquad B \longrightarrow \textit{External World}$$

Such a reaction is expressed as a first-order term in the differential equation for B:

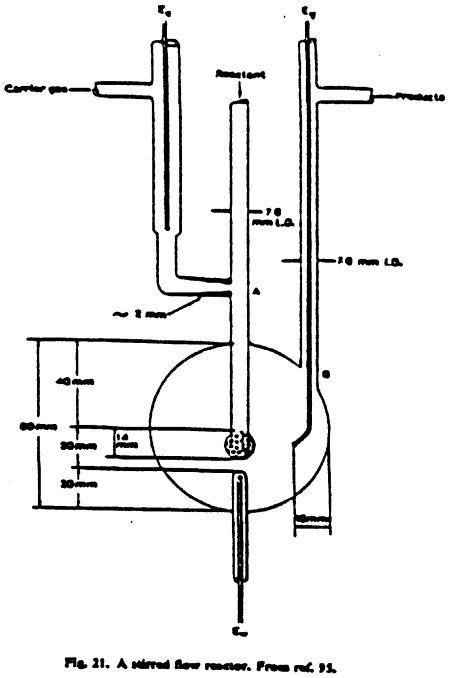


Figure 2.1: A Laboratory System With Sources and Sinks (from {7})

$$[11] \qquad \frac{d[B]}{dt} = -k_{snk}[B] + \textit{other terms}$$

Physically, sources and sinks correspond to situations often encountered in open laboratory systems. We might, for example, have a constant-volume reactor of volume V with input and output channels, as depicted in Figure 2.1.{7} As the reaction proceeds, a certain amount v of "feedstream" carrier gas is input to the tank within each second, and during that same time the identical volume v is withdrawn from the tank. The input stream contains a fixed concentration of species A, while the concentrations in the output stream are just those within the reacting system. In this case, our model of the reaction would include a source term for species A, and sink terms for all species in the reactive system.[4]

A third type of assumption often made allows for certain species to be "driven off" from a reactive system, or treated as having a concentration of zero within the reactive system. (Physically, this might correspond to a

[4]It is sometimes the case that certain species will be removed via such a sink term, and others will not. One might imagine, for instance, a tank reactor for liquid solution with a semi-permeable membrane blocking the output channel, allowing only certain species to leave the tank.

gaseous product of a reaction taking place in open solution.) In a sense, this is merely an extreme example of the "constant concentration" assumption discussed earlier; formally, the same approximation might apply to a species that is not in fact removed from the system, but that is simply an inert (nonreactive) product whose real concentration is of no concern.

Finally, it is worth mentioning the possibly disturbing convention of including "one-way" reactions within mechanisms. Since every reaction, viewed microscopically, is reversible, it would seem necessary to include a "back reaction" for every instance of a "forward reaction."

There are various reasons why the convention of using one-way reactions may not be dangerous. First, we may be interested in a given reaction

```
A ---> B
```

only for a certain brief period of time, and with an initial concentration of zero for species B. In this case, we are implicitly assuming that the concentration of B will be too small to affect our results by any appreciable amount during the period of interest. Another instance is given by reaction [4.4], shown earlier:

```
[4.4]    NO + N2O5 --> 3NO2
```

Here, the backward reaction would occur via a collision of three NO2 molecules, which is sufficiently unlikely (at moderate concentrations) so as to be treated as negligible. Yet another possibility might be that thermodynamic considerations dictate an overwhelming preponderance of products over reactants—that is, that under normal or expected conditions the "forward" reaction will run virtually to completion. And finally, there might be an implicit assumption that the backward reaction is rendered negligible for mechanical reasons; we might be removing products from the system very fast relative to the rate of the backward reaction. In every case, the inclusion of a one-way reaction is just an approximation that is built in to the formalism of the mechanism itself.

2.1.4 Sources of Complexity in Mechanisms

The preceding sections have outlined the path by which a chemical mechanism, expressed as a collection of plausible elementary chemical steps, is rendered into a mathematical model. Having created this model, the chemist now must study its behavior—an extremely difficult task. In general, even a relatively unthreatening mechanism will take the form of [7.1]–[7.5]: that is, a set of strongly coupled nonlinear ordinary differential equations. As is well known, the range of behavior available to such systems of equations is large, and few of these systems are solvable analytically.{74} Over the past several decades, mechanisms have been devised to reconstruct or explain a wide variety of "exotic" chemical phenomena—bistability, oscillations, birhythmicity, "chemical chaos," and so forth.{67}

The link between nonlinear differential equations and mechanism behavior is thus not usually obvious from the syntactic form of the equations themselves; this is true even for mechanisms consisting of four or five steps. Beyond this, sheer size may be a complicating factor in understanding the behavior of a mechanism. Some mechanisms in the literature include hundreds of steps (cf. {49}), while others are created by coupling together two independently difficult mechanisms.{17}

Even after the behavioral record of a particular mechanism has been generated—whether by direct simulation or other means—the task of describing and classifying that behavior can be problematic. Consider, for instance, the difficulty of merely describing whether a particular jump in concentration is a "rapid jump" or a "slow rise." Language of this sort implies a known time scale of interesting behavior; if the concentration of species A rises by ten percent in ten minutes, this may appear "fast" in the laboratory time frame, but it may be slow relative to other interesting events observed in the record. Because chemical reactions can take place over a remarkably wide range of time scales,[5] the observed "events" of a mechanism's behavior—jumps in concentration, stable periods, oscillations—have to be described

[5]Some fluorescence reactions take place in a matter of microseconds{42}; others, such as the (uncatalyzed) reaction between hydrogen and ethylene, proceed at a rate too slow to measure in the laboratory.{58} This wide range of time scales is true both for elementary chemical steps and overall reactions.

relative to an overall record of behavior, rather than as isolated occurrences.

2.1.5 Additional Complexities

Beyond the sources of complexity described above, there are additional factors that can make the understanding (and indeed the simulation) of mechanisms even thornier. Because these extra factors are not handled by the current Kineticist's Workbench, they will not be discussed at length; but they are worth mentioning, if only to provide a sense of how the picture presented above is itself a simplification of some "real-world" problems.

First, our presentation of systems of equations such as [7.1]–[7.5] implicitly assumes that our rate constants (such as $k1$) are indeed constant over the course of a reaction. In point of fact, rate constants are temperature dependent. The standard form of this rate dependence, derived from the Arrhenius equation, is:{55}

$$[12] \qquad \frac{d\,(\ln k)}{d\,T} = E_{act}/RT^{2}$$

where R is the gas constant, T is in degrees Kelvin, and E_{act} is a constant (thought of as an "energy barrier height" for the elementary reaction). Other forms for this relation have been proposed, but the essential point is that reaction rates can change noticeably with temperature. In many situations, this fact is immaterial since the reactive system is assumed to be kept at constant temperature; but there are some situations in which (for instance) a strongly exothermic reaction raises the temperature of the entire system, and this higher temperature in turn affects the rates of the various elementary reactions within the system. Thus, the changing temperature of the system plays an essential role in the progress of events.(cf. {55})

Another assumption implicit in the discussion thus far is that our reactive system is homogeneous, with no spatial variation in concentrations. In the laboratory, this condition often corresponds to the use of rapid mixing; but there are interesting cases in which concentrations of species vary in space as

well as time.{36, 81, 79} Inhomogeneous systems must be modelled by partial differential equations, and as such are a good deal more computationally demanding than the systems we will study in this paper.

Finally, we have been using the important assumption that the collision rate of two species A and B is proportional to the product of their concentrations. This is often a good approximation, but in practice there are numerous qualifying circumstances: in gases at very high pressures, for instance, the simple kinetic theory model on which this assumption is based begins to break down.{58} Charged species (e.g., ions in solution) represent another complication: often mechanisms involving such species can be recast in the standard form by using *activities* instead of concentrations as the variables of interest. The difficulty here arises from the fact that the activity coefficient (the "correction factor") of a species depends in a complicated way on the ionic strength of the solution (as well as other less important factors).{9} In practice, the form of equations [7.1]–[7.5], also known as *mass action kinetics* for elementary reactions, is widely employed for reactions in both gas and liquid phase.

To sum up this discussion, then, the systems handled by the Kineticist's Workbench are isothermal, homogeneous systems obeying mass action kinetics. Although these stipulations rule out many reactions for study, there is no lack of interest in the reactions left available to us. We will return to the additional complications mentioned in this section in the final chapter of this paper.

2.2 Strategies for Simplifying Mechanisms

Once a mechanism is constructed, there are various "special properties" that we can occasionally spot that allow us some conceptual purchase on its behavior. An extreme example of this would be a mechanism that happens to generate a set of linear ODEs (this could happen if, for instance, all elementary steps within the mechanism were unimolecular). The behavior of linear systems is well understood, so a mechanism of this type is especially simple to handle.{74}

More typically, a mechanism (along with expectations about the conditions under which it operates) will have some ad hoc numerical properties that permit simplifying assumptions to be made. We have already encountered a few of these in the previous sections: for instance, the use of "constant concentration" species often reflects the approximation that a particular species will be present in excess throughout the course of the reaction. In the following discussion, we will explore two other strategies for simplifying mechanisms that are commonly encountered in textbooks: the assumption of rapid equilibrium, and the steady-state assumption.

2.2.1 Rapid Equilibrium

Consider the following (very simple) mechanism:

```
[13.1]  A --> B      k1 = 100

[13.2]  B --> A      k2 = 200

[13.3]  B --> C      k3 = 0.1
```

Informally, what we have is a rapidly equilibrated mixture of A and B, and a slow transition from B to C. Thus, if we examine the behavior of this mechanism, with initial conditions of (say) concentrations of 1, 0, and 0 for A, B, and C respectively, what we will see shortly thereafter is a slow growth of C (up to almost 1 M concentration) during which the ratio of [A] to [B] is approximately 2:1.

The key to making this rapid equilibrium approximation is that the reactions perturbing the equilibrium between A and B dictate slow changes relative to the equilibrating process. Thus, at any given moment, the equilibrium ratios represent a good approximation to the truth. Mechanism [13.1]–[13.3] might be used to explain an overall first-order expression for the reaction A --> C, since we have

$$[14] \qquad \frac{d[C]}{dt} = k3\,[B] = k3\,(k1/k2)\,[A]$$

The rapid equilibrium approximation is used in a number of standard examples of kinetic theory; for instance, the treatment of unimolecular reactions by Lindemann and Christiansen, and later by Hinshelwood, is derived from this assumption.{55}

2.2.2 Steady-State Assumption

Earlier, the assumption of a constant concentration species was alluded to as a common approximation used to simplify mechanisms. In a similar spirit, one can often assume that some highly reactive intermediate species will be at a (low) constant concentration throughout the running time of a reaction. To see how this assumption arises, consider the following mechanism, represented graphically:

$$\text{[15]} \qquad A \xrightarrow{k1} B^* \xrightarrow{k2} C$$

We begin with the assumption that B* is a much more unstable (thus reactive) molecule than A; this implies that the transition from B* to C is much faster than that from A to B*, or in other words that $k2 \gg k1$. In this case, if we initialize our system with some of substance A (and nothing else), we will find that for most of the reaction time, there will be a very small (and nearly constant) concentration of B. Thus, we can write:

$$\text{[16.1]} \qquad \frac{d[B^*]}{dt} = k1\,[A] - k2\,[B^*] \simeq 0$$

$$\text{[16.2]} \qquad \frac{d[C]}{dt} = k2\,[B^*] \simeq k1[A]$$

where [16.1] has been used in [16.2].[6]

[6]As with the mechanism [13.1]–[13.3], the overall kinetics of the mechanism [15] is

Inasmuch as "reactive intermediates"—such as unstable species or free radicals—are not uncommon in chemical mechanisms, the steady-state assumption has wide utility. As a more complex (and realistic) example of its use, we can return to the decomposition of N2O5 shown in mechanism [4.1]–[4.4], and illustrate how the first-order expression [3] can be accounted for.[7]

The mechanism and accompanying differential equations are reproduced below:

```
[4.1]    N2O5  -->  NO2 + NO3

[4.2]    NO2 + NO3 --> N2O5

[4.3]    NO2 + NO3 --> NO2 + NO + O2

[4.4]    NO + N2O5 --> 3NO2

[7.1]   d [N2O5]/dt   =

            -k1 [N2O5] + k2 [NO2][NO3] - k4 [NO][N2O5]

[7.2]   d [NO2]/dt    =

             k1 [N2O5] - k2 [NO2][NO3] + 3 k4 [NO][N2O5]


[7.3]   d [NO3]/dt    =

             k1 [N2O5] - k2 [NO2][NO3] - k3 [NO2][NO3]


[7.4]   d [NO]/dt     =

            k3 [NO2][NO3] - k4 [NO][N2O5]
```

first-order in A; but the relationship of the overall rate expression to the elementary rate constants is different.

[7]The treatment here closely follows that in Laidler {55}.

[7.5] d [O2]/dt = k3 [NO2][NO3]

We will use a steady-state approximation for two intermediates: NO and NO3. Starting with NO3 and equation [7.3], we derive:

[17.1] k1 [N2O5] = (k2 + k3) [NO2] [NO3]

From the steady-state approximation for NO, we get:

[17.2] k3 [NO2] [NO3] = k4 [NO][N2O5]

As it happens, [17.1] is enough to derive the first-order expression for the rate of production of O2. We substitute [17.1] into [7.5] and derive the following:

[17.3] d[O2]/dt = k3 (k1/(k2 + k3)) [N2O5]

We can also derive a first-order expression for the rate of disappearance of N2O5 by using both [17.1] and [17.2] in expression [7.1]:

[17.4] d [N2O5]/dt = -k1[N2O5] + (k2 - k3) [NO2] [NO3]

= (-k1 + (k2 - k3) (k1 / (k2 + k3))) [N2O5]

= (-2 k1 k3 / (k2 + k3)) [N2O5]

Expression [17.3] accounts for the experimental results summarized in [3] earlier, and [17.4] predicts a similar result for the rate of disappearance of N2O5. It may also be worth mentioning that these expressions allow the overall measured rates of change for O2 and N2O5 to be related to the rate constants for the individual elementary steps. Thus, if we are somehow able

to measure (say) $k1$ and $k2$ independently, we can derive $k3$ from expression [17.4] and our experimentally determined rate of decomposition for N2O5.

2.3 Numerical Simulation of Mechanisms

The previous section delineated two common strategies for simplifying the numerical treatment of mechanisms. These techniques are often encountered in textbook examples, where they facilitate algebraic treatment of mechanisms (note, for instance, that we required nothing more than algebra and the steady-state assumption to recover [3] from mechanism [7.1]–[7.4] above).

Unfortunately, strategies such as those described above are not universally applicable. For example, in mechanism [13.1]–[13.3], which we used to illustrate the "fast equilibrium" approximation, we may be interested in a situation in which the rates of all three steps are comparable; in that case, our approximation no longer holds. Or it may be the case that a steady-state approximation holds for only some portion of the running time of a reaction. (Even in the case of reaction [15], we might imagine a hypothetical initial situation in which only A is present; in this case, it takes some brief amount of time before the steady-state approximation becomes a reasonable one.) More generally, as mechanisms of interest grow larger and more complex, these strategies become ever more limited in their applicability; and numerical simulation becomes a necessity for understanding behavior.

Over the past several decades, with the advent of digital computers, numerical simulation has become the prevalent method for studying the behavior of chemical mechanisms. By now a number of powerful simulation packages exist {5, 13, 23, 46}. Typically, the chemist will use such a package by entering a set of differential equations (much like [7.1]–[7.5] above), along with parameters such as rate constants and initial concentrations; having the computer integrate the equations with the given parameter values; and finally, saving or displaying the resulting concentration profiles. Unlike the techniques described in the previous section, numerical simulation is of course completely general, and its accuracy is limited only by that of the computer and the choice of integration algorithm.

2.3.1 Stiff Systems

As mentioned earlier in this chapter, chemical systems often involve processes taking place at widely divergent time scales. A (relatively mild) example of this phenomenon was encountered in mechanism [13.1]–[13.3]: here, the two elementary steps [13.1] and [13.2] dictate an equilibrium between A and B that is reached (to a good approximation) within seconds, while the third reaction converts B to C at a much slower rate. Many other mechanisms contain much more dramatic distances between the time scales of elementary reactions.

Systems of this type, exhibiting both "fast" and "slow" processes proceeding simultaneously, are known as *stiff systems.*[8] Chemical mechanisms are a notorious source of stiff systems, and as such constitute a challenge to numerical integration routines. Besides the usual criterion of numerical stability, integration routines for chemical systems must allow for adaptive time-steps (so that both fast "transient" behavior and slow "near-equilibrium" behavior may be captured in a reasonable number of integration steps).

The most common integration routine used for handling stiff systems is the Gear algorithm. The kth-order Gear algorithm is an implicit (predictor-corrector) integration algorithm whose value at each time-step is obtained by solving for the next state $x[n+1]$ in terms of that state itself and the k previous states $x[n]$,.... $x[n-k+1]$. As an example, the implicit equation for the fifth-order Gear's algorithm is shown below: {15}

```
[18] x[n+1] = (1/137) ( 300 x[n] - 300 x[n-1] + 200 x[n-2]
                                 -  75 x[n-3] +  12 x[n-4]
                                 +  60 h f(x[n+1], t[n+1]))
```

Here, we solve for $x[n+1]$ in terms of itself and the five most recent state values. Much, of course, is still unspecified by this equation alone: how an

[8] More formally, a linear system of ODEs is stiff if it has wide distances between eigenvalues. A nonlinear system is stiff if its Jacobian has wide distances between eigenvalues in some phase space region of interest.{15}

initial prediction for each time step is to be chosen, the correction algorithm to use, the strategy for changing the time step value h, and so forth. Chua and Lin {15} provide detailed discussion of these points, as well as a derivation for the particular constants in equation [18]; and they also discuss matters such as strategies for implementing adaptive-order Gear routines.

The Kineticist's Workbench supplies the user with a choice of integration routines. For stiff systems, the Workbench can employ the Gear algorithm supplied by IMSL, Inc.{50}; for non-stiff systems, the Workbench uses a fixed-time-step fourth-order Runge-Kutta algorithm (a relatively simple explicit integration routine {44}). More detail on the Workbench's numerical integration module will be provided in Chapter 4.

2.3.2 Stochastic Simulation

Earlier in this chapter, we saw that certain types of approximations (such as near-constant concentrations) may be incorporated into mechanisms, thus simplifying the differential equations produced. In this sense, mechanisms are often unabashedly approximate models of chemical systems. But there is yet another approximation that is so pervasive in our discussion thus far that it may go unnoticed: namely, that chemical systems can be described by differential equations at all. In truth, concentrations are not continuous quantities; since any system has a discrete number of molecules of any given species, it is an idealization to speak of the "derivative" of a concentration over time.

Of course in most situations this "idealization" is not problematic. A typical laboratory system will have on the order of 10^{20} molecules of any given species; and even a very dilute solution might contain 10^{10} solute molecules. Thus, speaking of derivatives of concentrations is as reasonable as, say, speaking of the derivative of electric current in a DC circuit (another "idealization"). Nevertheless, there are situations in which the differential equation formalism proves inadequate: for instance, we may actually be dealing with exceedingly small numbers of reacting molecules. In these situations, we would like some other method for simulating chemical systems—some method that reflects the discrete nature of the system being modelled.

The mathematical approach used in such situations is a probabilistic one: the chemist sets up a "master equation" in which the probability of finding the system in a given state at time t is obtained by summing the probabilities that the system entered this state from some previous state at time $t - dt$. Formally, such an equation looks as follows:

$$[19] \qquad P(s,t) = \sum_{j} P(j,t-dt) * W(j,s)$$

In prose, equation [19] says that if we know the probability of finding our system in any state j at time $t-dt$, and the probability that a system in state j changes in time dt to state s (this the so-called "transition probability," denoted by $W(j,s)$), then we can compute the probability of finding the system in state s at time t.[9]

Simulating a system described in this way generally means starting from a given initial state $s0$, and using the (known) transition probabilities to generate subsequent states nondeterministically. Computer programs for this purpose use Monte Carlo methods and are typically run numerous times to generate statistical averages of system behavior.{74}

Despite the utility and interest of stochastic simulation, we will not pursue the subject beyond this brief outline. The Kineticist's Workbench employs differential equation models exclusively to generate numerical output, and these "classical" models will be the sole focus of our attention henceforth. Nevertheless it is worth including this telegraphic description of stochastic techniques, if only to indicate some of the hidden limitations of differential equation models, and the range of simulation methods available to the modern chemist.

2.3.3 Limitations of Numerical Simulation

Despite its usefulness and increasing affordability, numerical simulation

[9]It is not hard to show that for simple systems in which we assume *dt* very small and a large number of particles, we can recover the expected differential equations from a stochastic treatment such as this.{74}

has unavoidable limitations as an aid to understanding mechanisms. Typically, the results of running a numerical simulation are large tables (or graphs) of numbers representing the concentrations of various species over time. Although this data is indispensable, the chemist is still required to do a good deal of "qualitative" work, both before the simulation is run and after the results are retrieved.

Before actually running the simulation, the chemist will look over the mechanism to see if there are deductions about its behavior that may be reached even in the absence of simulation. For instance, he may try to spot special mechanistic features like "rapid equilibria" of the kind mentioned earlier; or he may look to see whether the mechanism has certain properties that guarantee that it will reach a stable equilibrium; or he may note that this particular mechanism gives rise to linear differential equations, and that he can thus express its behavior directly, by a mathematical formula. All of this may be done before any simulation results are obtained—though of course those results may later be helpful in checking the intuitions and predictions developed during this stage of the work.

After running the simulation, the chemist will look over the results, trying to spot features of interest—e.g., rapid changes in concentration, or long periods during which some concentration is nearly constant, or oscillations. He will try to relate those observed features to aspects of the mechanism itself; for example, he might ask which particular steps in the mechanism were most directly responsible for causing some large jump in the concentration of a given species. In essence, the chemist attempts to fit a narrative structure to the overall results, as evidenced in the following passage (in this passage, α and β refer to small collections of elementary steps in the mechanism, and symbols such as F3 refer to individual steps):

> "The rate of (α) is proportional to Y, and when this process is dominant X attains a steady state approximated by X_{min}... At such a time, Y will be depleted by the dominance of (α). If (F3) is rate-determining for (F3-4), then when (β) is dominant the rate is proportional to X and X attains a steady state concentration approximated by X_{max}... Transition between dominance by (α) and (β) is strongly dependent on Y and takes place whenever that concentration passes through $Y_{critical}$." {62}

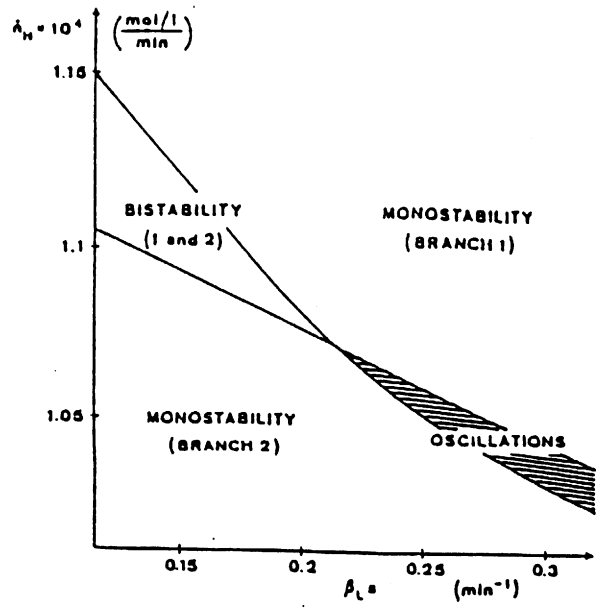


Fig. 5 Calculated domains of monostability, bistability and oscillations on the plane of the hydrazine concentration feeding rate ($\dot{n}_H$) and the volumetric mass transfer coefficient of oxygen stripping ($\beta_L a$). The domains correspond to the kinetic branches 1 or 2 as indicated in parentheses.

Figure 2.2: A Parameter Space Graph (from Markus {59})

This kind of narrative interpretation is a significant part of what it means to "understand a mechanism." The chemist reads the simulation not as numbers, but as a sequence of interesting events; and he is able to relate those events back to the participation of individual steps within the original mechanism.

In fact, the chemist's interpretive powers must generally be extended beyond the results of a single simulation. Typically the chemist will perform a series of simulations, varying one or more parameters (such as a rate constant or initial concentration of some species). He will then attempt to relate the presence or absence of particular features (such as oscillations) to the values of the parameters chosen. Often he will generate a parameter-space graph, showing how certain types of mechanism behavior vary with the choice of parameters. Figure 2.1 shows such a graph from Markus {59}. This graph summarizes the behavior of a particular model of an oscillating reaction; the axes correspond to parametrized source and sink rates within the model. The various annotated areas of the graph correspond to parameter regions in which the model exhibits bistability (two possible steady states), monostability, or oscillations.

It is thus fair to say that while the numerical results of simulation represent the primary data for the chemist—there is no purely qualitative for-

malism that can accommodate the richness of detail provided by the actual numbers—nevertheless, the mere generation of numbers is only a fraction of the chemist's job in understanding and analyzing mechanism behavior. The greater and more interesting part of that job is spent in making predictions of the numbers beforehand, and making narrative summaries of the results afterward.

The Kineticist's Workbench is designed in accordance with this view of the scientist's work. The Workbench augments the standard techniques of numerical simulation with additional symbolic procedures that assist the chemist in predicting and analyzing mechanism behavior. The Workbench uses graphical procedures to examine the structure of a proposed mechanism and determine whether any meaningful predictions of mechanism behavior may be drawn prior to simulation; for certain mechanisms it can determine whether the system must necessarily reach a stable equilibrium. The program can also spot a variety of interesting special features within mechanisms, including "rapid equilibria" (as described earlier) and autocatalytic loops (a feature associated with exotic behavior such as oscillations).

After simulation is complete, the Workbench is able to analyze the events of the simulation by several means. The concentration profiles are divided into "episodes," roughly corresponding to changes in the mechanism steps determining local concentration changes[10]; these may be used to generate narrative descriptions of the simulation. The program also separately keeps track of local concentration maxima and minima which it can use to spot a variety of standard oscillation patterns; and it is able to generate parameter-space graphs showing how the presence or absence of simulation events changes with one or two parameter variations.

2.4 Why Kinetics Matters

The remainder of this paper will be devoted almost exclusively to the Kineticist's Workbench program—its features, implementation, use, and limitations. But before proceeding with this examination of the program, it is worthwhile stepping back to ask why chemical kinetics should hold any

[10]The full description of this part of the program is given in chapter 6 below.

interest (or utility) as a domain of study. Why learn, or explore, kinetics at all?

First, kinetics is an important aspect of the study of how chemical reactions occur. A typical contrast drawn in chemistry courses is that between thermodynamics and kinetics: the former can be used to determine whether a given reaction is energetically favored or not (i.e., whether energy considerations dictate an equilibrium constant that favors the "forward" or "backward" reaction). The latter is used to ascertain how a reaction occurs over time. A good illustration of this distinction is given by the following reaction:{55}

```
2H2 + O2  <----> 2 H2O
```

Thermodynamically, the right side of this reaction is strongly favored at moderate temperature and pressure; the equilibrium constant for this reaction dictates a huge excess of H2O. But the kinetics of the reaction dictates that, if we mix hydrogen and oxygen gas under normal laboratory conditions, appreciable conversion to H2O will take many millions of years.[11]

Formulating a mechanism for a particular reaction can provide the chemist with insight into other, related reactions. For instance, if we are studying a reaction system in which methyl radicals (CH3) are present, and we find that the elementary step

```
2CH3 ---> C2H6
```

appears in our mechanism, then we would expect the same elementary step (with the same rate constant) to appear in any mechanism in which methyl radicals appear.[12] The elementary steps that we postulate for one reaction thus carry over, and lend us information about, other reactions.

[11]Lighting a flame can speed things up considerably!{55}

[12]In point of fact, this step appears in mechanisms for acetaldehyde decomposition {55} and methane decomposition {74}. In some systems with CH3 radicals present, we might not include this step because some other step(s) with CH3 as reactant may proceed much faster and thus be more important.

It is also possible to find structural similarities between mechanisms of different reactions. If we find, for example, that both nitric oxide and propylene are able to inhibit certain organic decomposition reactions {55}, and that the rate laws governing inhibition have the same general form, then we might suspect that the mechanism of inhibition is similar for the two substances. Or, having elucidated a "chain reaction" schema for the formation of hydrogen bromide, we might look for similar steps in the reactions of bromine with a variety of molecules of the form X-H (such as methane, CH4, in which "X" can be thought of as the methyl CH3- group).

In a similar vein, one can construct "abstract mechanisms" that in effect illustrate the basic pattern of a mechanism, but that do not refer to any one particular chemical reaction. (Mechanism [13.1]–[13.3], the earlier instance of a "rapid equilibrium" reaction, is an example.) This can be a fruitful technique for understanding or generating new classes of reactions: if we create a very abstract mechanism that exhibits oscillations, but that does not actually refer to any particular chemical system, we might then try to prepare laboratory reactions that obey a similar mechanism in the hopes of finding oscillations.[13] Thus, the study of chemical mechanisms has a "synthetic" as well as "analytic" purpose: starting from known reactions, we can proceed (analytically) to a hypothetical mechanism, while starting from a sample or abstract mechanism we can proceed (synthetically) to look for reactions that appear to illustrate it.

The study of chemical kinetics has real-world consequences. Historically, the study of various chain reaction schemas led to an understanding of explosive systems and polymerization reactions.{55, 51} More recently, much of the debate on the role of fluorocarbons in the depletion of stratospheric ozone has centered on the comparison of the behavior of rival mechanisms.{16, 74} In the past several decades, chemical systems have been regarded as an especially good source of examples in the study of nonlinear dynamics; oscillating reactions (such as the Belousov-Zhabotinskii and Briggs-Rauscher reactions) are used in the laboratory to study (and demonstrate) limit cycles, period-doubling, strange attractors and other fundamental concepts of dynamical systems theory.

[13]The Brusselator mechanism, to be discussed in detail later, is an example of such an abstract "illustrative" mechanism.

As a final point, it should also be mentioned that chemical mechanisms are often syntactically equivalent (or similar) to models derived from other disciplines. The Lotka-Volterra model—an early simple model of oscillations—was a chemical mechanism adapted for use in population biology.{56}[14] A variety of bacterial growth models may be recast in the formalism of chemical mechanisms {10, 78}; and as noted earlier, radioactive decay reactions may be modelled as unimolecular elementary steps. Thus, a system presented "formally" as a chemical mechanism may in fact be capable of modelling interesting non-chemical phenomena.[15]

[14]The Lotka-Volterra model results in conservative, as opposed to limit-cycle, oscillations.

[15]Samardzija *et al* {70} in fact, demonstrate nonlinear transformations through which several famous examples of ODE systems—including a forced RLC circuit and the Lorenz system (derived from meterology)—are recast as chemical mechanisms with qualitatively similar phase portraits.

Chapter 3

The Kineticist's Workbench: A Design Overview

The Kineticist's Workbench program is best thought of as a set of three modules. The first module is primarily concerned with numerical simulation of mechanisms; the second with pre-simulation analysis of mechanisms; and the third with post-simulation interpretation of results.[1] Although this division represents a useful conceptual classification—each module handles a different stage of mechanism analysis—the various portions of the program are close-knit and have extensive communication with one another. For instance, the numerical simulation routines generate on the fly much of the symbolic information that will be used (along with the numerical results) by the post-simulation interpretation routines; likewise, the results of (pre-simulation) graphical analysis can occasionally be incorporated into numerical simulation (in ways that will be described at length later on in this paper). Nevertheless, the division into numerical, pre-simulation, and post-simulation portions of the program provides a close match to the way in which the user would employ the Workbench, and as such will be the basis of discussion here.[2]

3.1 Numerical Simulation

The Workbench contains a collection of procedures that together comprise a flexible package for simulating mechanisms. The user enters a mechanism (in the format of a Scheme list); the Workbench then translates this mechanism into differential equation procedures, using much the same strategy

[1] There are also a few additional procedures that supply a rudimentary user interface.

[2] An implementation note may be in order here. The Workbench is written in MIT Scheme, and is run on a Hewlett Packard 9000/350 workstation. The compiled program currently requires a little over a megabyte of memory storage; in addition, as mentioned in the previous chapter, the program makes use of the IMSL mathematical library {50}, and uses calls to the X Window system as the basis of its graphical output.

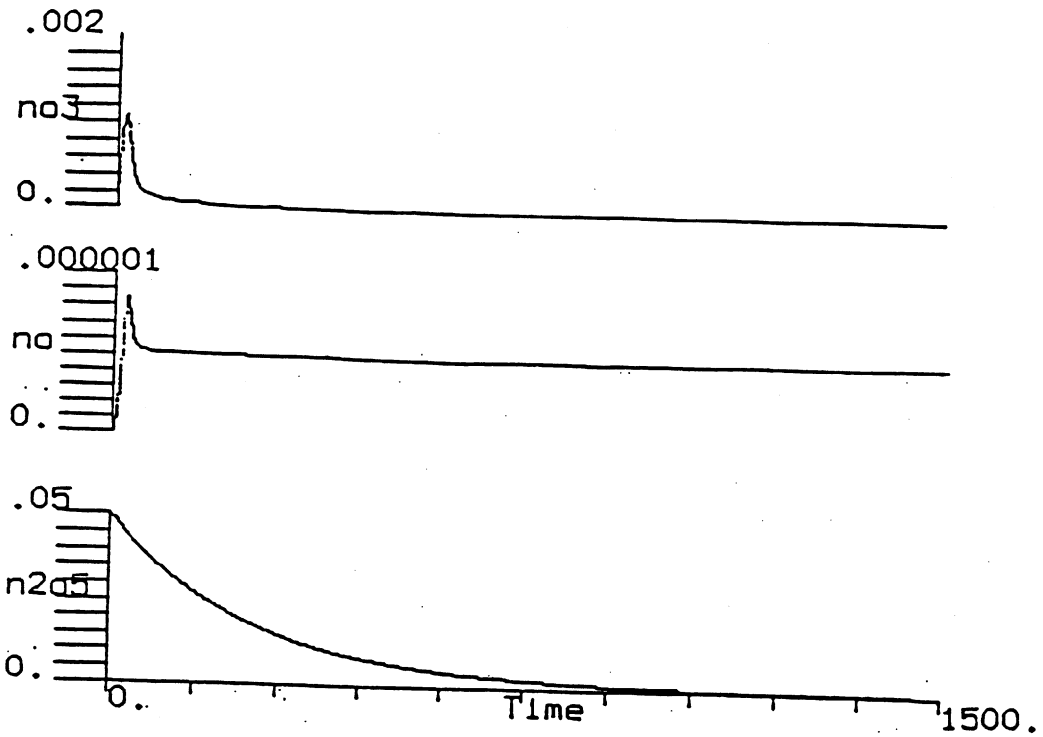


Figure 3.1: A Graph of Three Concentrations Versus Time

outlined in Chapter 2 earlier; and the resulting system of ODE procedures is passed off to a fourth-order Runge-Kutta integration routine, which can print (or graph) its numerical results on the computer screen or save them to a file.

Should the user wish to employ the IMSL Gear integration routine, again the mechanism is entered symbolically; but in this case the Workbench actually uses the symbolic mechanism to generate a Fortran program which is then compiled and run (and which itself calls the IMSL library routines), saving the numerical results to a specified filename. These results may then be read back, at which time they may be printed or graphed.

The graphing procedures provided by the program allow the user to display several concentration-versus-time graphs simultaneously as the simulation progresses (see Figure 3.1 for an example). These pictures may also be stored as files.

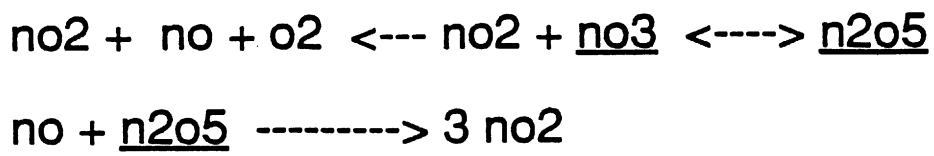


Figure 3.2: A Graph of a Mechanism

3.2 Pre-Simulation Analysis of Mechanisms

The Workbench includes a variety of useful techniques for analyzing mechanisms prior to simulation. First, the program generates a "graph representation" of the mechanism that facilitates structural analysis; by performing certain graph-theoretic algorithms on this representation (to be described at length in Chapter 5), the program is often able to make powerful predictions about the mechanism's eventual behavior. In some cases, the Workbench is able to conclude that the mechanism as written must eventually approach an equilibrium state (with all species at non-zero concentration). In other cases, the Workbench may conclude that the system cannot reach equilibrium with all positive concentrations; and the program can then offer possible asymptotic states for the system's behavior. Even if the mechanism is of a more complex type that does not lend itself to predictions of this kind, the program may still look for interesting graph-theoretic features within the mechanism often associated with exotic behavior (such as the presence of autocatalytic loops).

The results of this graphical analysis are stored in a data structure associated with the original mechanism; they may also be displayed on the computer screen, and in some cases are depicted "pictorially," as suggested by Figure 3.2. (A complete description of this representation will be provided in Chapter 5.)

Going beyond the purely graph-theoretic algorithms alluded to above, the Workbench is also able to look for certain standard simplifying "reaction patterns"—such as the presence of a rapid equilibrium reaction pair—and

to note these in its analysis of the mechanism. Notions such as "fast equilibrium" and the "steady-state" approximation require some approximate numerical decisions (eg, whether one rate constant is "much bigger" than another), and so cannot be based on graph-theoretic information alone. This type of analysis might be categorized as "quasi-numeric," as opposed to the purely structural conclusions of the previous two paragraphs.

Finally, in those occasional cases in which the Workbench is able to conclude that the mechanism must reach a unique equilibrium state independent of initial concentrations, it can use simple minimization methods to locate that equilibrium state directly, without performing the complete simulation.

3.3 Post-Simulation Interpretation of Mechanisms

Having performed a numerical simulation, the Workbench uses a variety of means to intepret the numerical results and to note the presence of (possibly) interesting behavior. The program uses a special structure known as the *episode* (to be described at length in Chapter 6) to link behavioral patterns of the mechanism to the activity of elementary reactions. The program generates an "episode history" which can often be used directly as a kind of narrative depiction of the simulation; this history can also be used to spot portions of the simulation during which simplifying assumptions might be made (eg, periods during which the concentration of some particular species remains virtually constant). The episode history also can be used (on occasion) to find individual elementary steps that prove to be unimportant in determining the mechanism's behavior and which may be dropped from the original mechanism specification.

For each species of interest, the Workbench keeps track of when that species reaches a local maximum or minimum of concentration during the simulation. By examining patterns of maxima and minima, the program can spot a number of common oscillation patterns (including stable limit cycles and damped oscillations). Having done so, the program also generates informative parameters of the identified oscillation (such as the period and amplitude of a stable limit cycle). In certain cases the program can also identify even more exotic behavior such as the presence of birhythmicity or

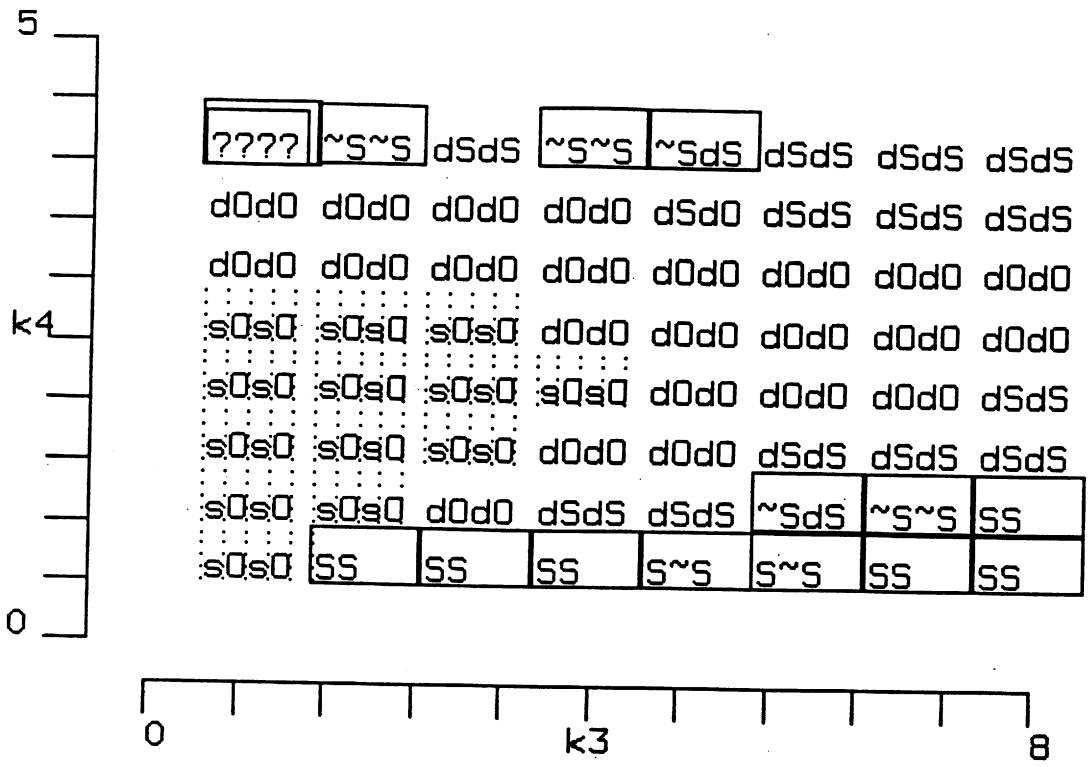


Figure 3.3: A Parameter-Space Graph

chaotic (non-periodic) oscillations.

Going beyond the results of a single numerical simulation: the Workbench can be used to run a sequence of simulations in which up to two mechanism parameters (rate constants or initial concentrations) are systematically varied. In this case, the program stores its qualitative interpretation of each run; and the stored information can then be retrieved and used to generate parameter-space graphs that indicate how qualitative behavior changes with the given parameters. An example of such a graph is shown in Figure 3.3, and the origins of this example will be described in Chapter 7 of this paper.[3]

It should also be mentioned that the qualitative information produced by the Workbench is in the form of Scheme structures (generally, lists of symbols and numbers). This implies that this information may be used as the input to other Scheme procedures. A specific example of this approach is indicated in one of the examples to be shown in Chapter 7: in that example, a few special procedures are written to identify that portion of the numerical results in

[3]The graph shown in Figure 3.3 is reproduced from original Workbench output. The Workbench uses color-coding, rather than "stippling" and "box-outlines," to distinguish different symbols.

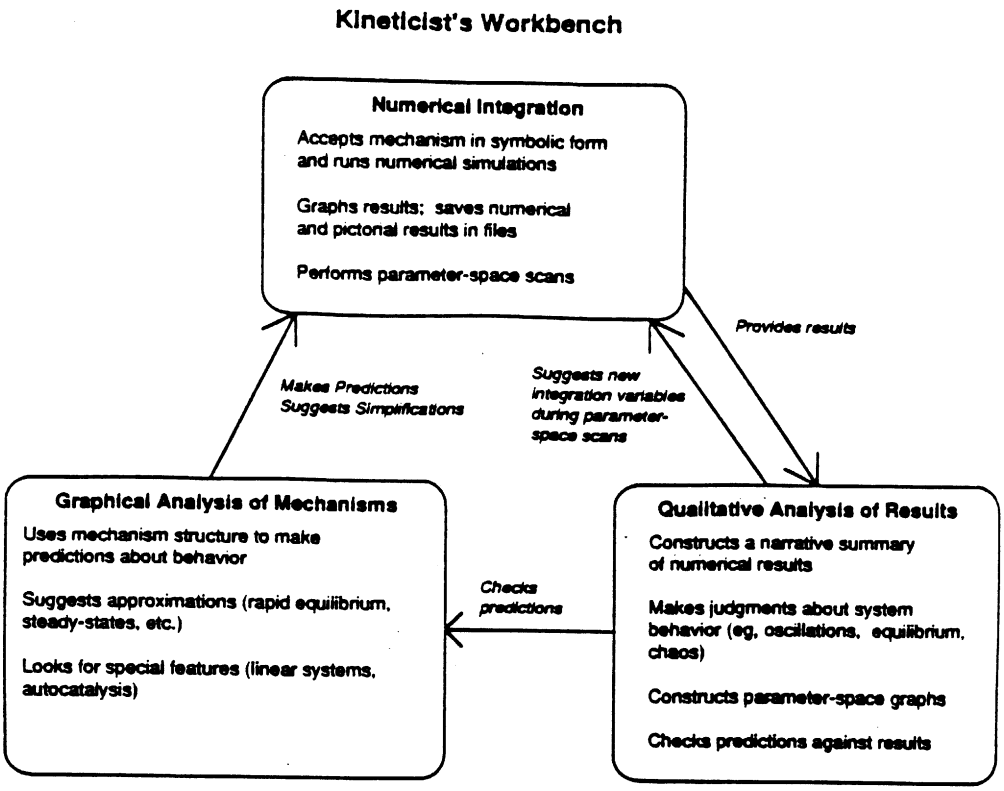


Figure 3.4: The Workbench Program Structure

which a stable oscillation has been spotted, and to use only that (oscillating) portion of the results as input to an FFT routine. In other words, we can use the qualitative interpretation of our numerical results to identify portions of the numerical results that are appropriate to study via still other numerical methods.

3.4 The Workbench as a Whole

As mentioned above, there are several paths of communication between modules that have been implemented. The pre-simulation routines are able to send information (such as known equilibrium concentrations) to the numerical simulation module; the numerical module provides the results (and additional information, such as "episodes") that are analyzed by the post-simulation interpretation routines; and in some cases, the post-simulation routines can call procedures from the other two modules. (For instance, if the interpretation routines are unable to identify any of several standard simulation results—equilibrium, stable oscillations, damped oscillations, and so forth—they can sometimes redo the original numerical run, but with a longer

simulation time, in the hopes of identifying some standard behavior pattern that is not apparent in the earlier part of the run.)

Although much has been done, there is still plenty of room for expansion of the Workbench program, both within modules and involving communication between modules. Some of the possible directions for expansion will be discussed in Chapter 8; but in any event, the complexity of chemical mechanisms (and the variety of types of reasoning involved in understanding them) imply that there will be ample room for elaboration in the Workbench for a long time to come. The current structure of the Workbench program as summarized by this chapter—namely, the three modules and the types of communication possible between them—is indicated by Figure 3.4.

The following three chapters will describe, in turn, the numerical, pre-simulation, and post-simulation modules of the Workbench. After this, Chapter 7 will illustrate the overall capabilities of the program with several complete examples.

Chapter 4

Numerical Simulation in the Workbench

4.1 The Mechanism Data Structure

The first stage in simulating a mechanism is to enter a representation of that mechanism into the program. The Workbench's mechanism data structure is a Scheme list of the following general form:

```
mechanism =

          ( <step-list>
            <constant-species-list>
            <sources>
            <sinks>
            <driven-off>
            <functions-of-time> )
```

Thus, a mechanism is a list of six elements, each of which is itself a list. The first *step-list* element represents the sequence of elementary reactions, along with their rate constants. The second element indicates which species within the *step-list* are to be regarded as constant, and what their concentrations are. The third and fourth elements indicate which species are replenished by sources or lost to external "sinks" as described in Chapter 2. The fifth element indicates which species are regarded as "driven-off," i.e. having constant zero concentration.[1] Finally, the last element is a list of species whose concentrations will be given as functions of the time parameter in the simulation (generally, simulations will run from $t = 0$ to some final time value t_f).

[1] Pragmatically, this is only of use for the occasional species that occurs as the product of a reaction. We would obviously not wish to include an elementary step one of whose reactants is constant at zero concentration.

Beyond this description, it is probably easiest to introduce the format of the mechanism data structure through examples. Suppose, then, we wish to represent the *Brusselator* mechanism; this is an early abstract mechanism for oscillating reactions invented by Prigogine and Lefever {61}, and will be used extensively as a later example:

```
[1.1]     A --> X              k1 = 1
[1.2]     B + X --> Y + D      k2 = 1
[1.3]     2X + Y --> 3X        k3 = 1
[1.4]     X --> E              k4 = 1
```

In this version of the mechanism, A and B are interpreted as constant species (with concentrations chosen as 1 and 3, respectively), and D and E are interpreted as "driven-off" products. The Workbench representation of this mechanism would be:

```
(define brusselator
  '(
    (  ( ((a 1))        ((x 1))        1)
       ( ((b 1) (x 1))  ((y 1) (d 1))  1)
       ( ((x 2) (y 1))  ((x 3))        1)
       ( ((x 1))        ((e 1))        1) )  ; step-list

    ( (a 1) (b 3) )                          ; constants
    ()                                       ; sources
    ()                                       ; sinks
    ( d e )                                  ; driven off
    ()))                                     ; functions of time
```

The correspondence between the first (step-list) element in this data structure and the four steps [1.1]-[1.4] should be straightforward: each element of the step-list represents a single step, consisting of a list of reactants, a list of products, and a numeric rate-constant value. Among the subsequent elements, constant species are indicated as lists of species-and-concentration pairs; driven-off species are simply indicated by the "driven-off" element.

Just for completeness, we might imagine including an external source for B (which is thus no longer constant); an external "sink" for D (which is no

longer a driven-off species); and finally, we might also imagine varying **A** as a sinusoidal function of time, rather than treating it as constant:

```
(define brusselator-with-sinusoidal-a
  '(
    (  ( ((a 1))        ((x 1))        1)
       ( ((b 1) (x 1))  ((y 1) (d 1))  1)
       ( ((x 2) (y 1))  ((x 3))        1)
       ( ((x 1))        ((e 1))        1) )    ; step-list

    ()                                         ; constants
    ((b 0.01 5))                                    ; sources
    ((d 0.01))                                      ; sinks
    (e)                                             ; driven off
    ((a ,(lambda (time)
           (+ 2 (* 2 (sin (* 0.1 time)))))))   ; functions
    ))
```

Here, the external sources are indicated by a list giving the source species, the flow rate, and the source concentration; and sinks are indicated by giving the sink flow rate. Just to see what this means vis-a-vis the differential equations governing the system, we would now have the following system of ODEs:

$$[2.1] \quad \frac{d[X]}{dt} = k1[A] - k2\,[B][X] + k3[X]^2[Y] - k4\,[X]$$

$$[2.2] \quad \frac{d[Y]}{dt} = k2\,[B][X] - k3[X]^2[Y]$$

$$[2.3] \quad \frac{d[B]}{dt} = 0.05 - k2\,[B][X]$$

$$[2.4] \quad \frac{d[D]}{dt} = k2\,[B][X] - 0.01\,[D]$$

The functions-of-time element in our mechanism data structure is a list of species-function pairs: the function part of the pair is a procedure that, when called on the current time value, will return the concentration of the given species.

The mechanism structure introduced here is the core information provided by the user to the Workbench. This structure is employed by the program as the basis of a more elaborate "mechanism object" which contains not only the original mechanism data structure, but a great deal of other information deduced by the Workbench. We will return to the subject of mechanism objects in Chapter 5.

4.2 Simulating a Mechanism with the Runge-Kutta Integrator

Having entered a mechanism into the program, we might now wish to perform a numerical simulation of that mechanism using a fourth-order Runge-Kutta integration routine. In order to do this, a few additional parameters are needed: in particular, we need an initial state (set of initial concentrations), a time-step value to use for the integrator, and starting and ending times for the simulation. These can be provided in a "run-parameters" list; pursuing the Brusselator example (given by [1.1]-[1.4] above), we could construct a sample run-parameters list as follows:

```
(define brusselator-run-parameters
  '((starting-dt 0.025)
    (start-time 0.)
    (end-time 40.)
    (actual-starting-concs
      ((a 1.) (b 3.) (d 0.) (e 0.) (x 0.) (y 0.)))
    (integration-method runge-kutta)
    (focus-species (x y))
    (steps-per-display 40)
    (save-numeric-results? ,false)))
```

The run-parameters list is a simple association list of parameter-value pairs. Briefly summarizing this example: the first three pairs indicate that we

wish to simulate the system from time $t = 0$ to $t_f = 40$ seconds, with a Runge-Kutta dt value of 0.025 seconds. The fourth pair provides a list of starting concentrations; and the fifth indicates that we wish to use the Runge-Kutta integrator. The `focus-species` pair indicates that for our purposes, the concentrations of species `x` and `y` are of interest, and these concentrations will be printed out every 40 time-steps (as specified by the `steps-per-display` pair).[2] Finally, the `save-numeric-results?` pair indicates that we do not wish to save out the numeric results into an external file (if the value of this parameter were `true`, then we would need an additional parameter in the list to specify a file name to use).

Having constructed both a mechanism structure and a run-parameter list, we can simulate the mechanism as follows:

```
(do-simple-mechanism-run brusselator
                         brusselator-run-parameters)
```

The `do-simple-mechanism-run` takes as arguments a mechanism structure and a list of run parameters, and simulates the mechanism numerically, printing out on the screen the appropriate concentration values. Because we have selected the Runge-Kutta integrator, the Workbench first constructs a set of differential-equation procedures for each species (via the translation algorithm indicated in Chapter 2), and uses these as input to its integration routine. The results of the integration are printed out on the screen (here, we are printing the concentrations of `X` and `Y` every 40 time-steps, corresponding to one second of simulated time):

[2]Note that for a large mechanism, it might well be the case that we only wish to follow the concentrations of one or two "interesting" species; this is the point of having a user-defined set of "focus species," rather than having the program assume that information about all species needs to be retained.

```
Species: y     Concentration: 0.
Species: x     Concentration: 0.

Species: y     Concentration: .55844
Species: x     Concentration: .25122
Current time: 1.

Species: y     Concentration: 1.2790
Species: x     Concentration: .26938
Current time: 2.
```

The program continues printing at one-second intervals until the end of the run.

4.3 Simulating a Mechanism with the Gear Integrator

As mentioned in Chapter 2, certain chemical mechanisms prove too stiff for adequate simulation by an explicit integration algorithm such as the Runge-Kutta routine. An example is the Oregonator {61, 77}, proposed by Field, Koros, and Noyes as a mechanism for the Belousov-Zhabotinskii reaction:

```
[3.1]   A + Y    -->  X          k1 = 1.34
[3.2]   X + Y    -->  P          k2 = 1.6e9
[3.3]   B + X    -->  2X + Z     k3 = 8.0e3
[3.4]     2 X    -->  Q          k4 = 4.0e7
[3.5]        Z   -->  fY         k5 = 0.8
```

Typically, f and $k5$ are parameters to the mechanism (for our present purposes, we will let $f = 1$); species A and B have constant concentrations (both at 0.06), and species P and Q are driven off species.

We represent this mechanism as follows:

```
(define oregonator
  '(
    ;list of steps
    (
     ( ((a 1) (y 1))    ((x 1))       1.34)
     ( ((x 1) (y 1))    ((p 1))       1.6e9)
     ( ((b 1) (x 1))    ((x 2) (z 1)) 8.0e3)
     ( ((x 2))          ((q 1))       4.0e7)
     ( ((z 1))          ((y 1))       0.8)
     )
    ((a 0.06) (b 0.06)) ;constant species
    () ;sources
    () ;sinks
    (p q) ; driven off
    () ;functions of time
    ))
```

And we specify run parameters appropriate to the Gear integrator:

```
(define oregonator-run-parameters
  '((starting-dt 1.0e-9)
    (start-time 0.)
    (end-time 200.)
    (actual-starting-concs
     ((a1 0.06) (b 0.06) (p 0.) (q 0.)
      (x 0.0000000001) (y 0.00002) (z 0.0000000001)))
    (integration-method gear)
    (gear-tolerance 0.0000000001)
    (focus-species (x y z))
    (steps-per-display 200)
    (time-per-gear-read-out 0.025)
    (save-numeric-results? ,true)
    (numeric-result-filename
      "/mydata/oregrun01.out")
    ))
```

The run parameters here are similar to those for the Runge-Kutta example earlier, but there are a few differences worth noting. First, we have chosen a much smaller *dt* value; in the case of the Gear algorithm, which uses an adaptive time-step, this will only be an initial value for the algorithm (typically,

later time steps will be a good deal larger). Having specified that we are using the Gear algorithm, we also need to specify an error tolerance. This will be a parameter passed to the eventual IMSL routine, and it corresponds to a number used by the "corrector" portion of the algorithm indicating when a sufficiently small change has been noted between corrections.[3] At present, the Workbench has no alternative except to save Gear results to an external file: hence we have a `save-numeric-results?` parameter value of `true`, and we have provided a file name for the `numeric-result-filename` parameter. The `time-per-gear-read-out` specifies that the IMSL routine will print out to this file numeric values for all concentrations at every 0.025 of simulated time. Finally, the `focus-species` and `steps-per-display` parameters indicate that after the numeric results are saved, they will be read back from the file; and after every two hundred reads (that is, every 5 seconds of simulated time), the concentration values of species `X`, `Y`, and `Z` will be printed out on the screen.

At this point, we can evaluate another call to `do-simple-mechanism-run` as follows:

```
(do-simple-mechanism-run oregonator
                         oregonator-run-parameters)
```

In this case, the Workbench constructs, compiles, and runs a FORTRAN program that calls the IMSL Gear routine with an appropriate representation of the Oregonator mechanism. The numeric output of this call is sent to the filename provided by the user; read back; and the following printout appears on the screen:

```
Species: z     Concentration: .0000000001
Species: y     Concentration: .00002
Species: x     Concentration: .0000000001

Species: z     Concentration: .000000030458
Species: y     Concentration: .0000089341
Species: x     Concentration: 5.1996e-11
```

[3] If this value is chosen too high—say, 10^{-7} in this case—then the simulation results include negative concentration values. Thus, a very "tight" tolerance is usually indicated for stiff chemical systems.

```
Current time: 5.


Species: z    Concentration: .000000032145
Species: y    Concentration: .0000039972
Species: x    Concentration: 5.4266e-11
Current time: 10.
```

Generally, the choice of whether to use the Gear or Runge-Kutta integrator depends on the complexity of the mechanism being simulated: the former method is required for most "exotic" mechanisms from the literature. Additionally, the current version of the Workbench is limited in that it cannot use Gear integration on mechanisms with a non-null functions-of-time element.

As a final note, it is worth mentioning that Runge-Kutta numeric results, like those produced by the Gear integrator, may be saved out to an external file by using the appropriate settings of the `save-numeric-results?` and `numeric-result-filename` parameters.

4.4 Graphing Numeric Results

Besides printing out numeric values of concentrations on the display screen, the Workbench can produce graphs of concentrations. Pursuing the Oregonator example from the previous section, we might wish to obtain a graph of the concentrations of `X`, `Y`, and `Z` over time. In this case, we need to add one additional parameter to the run-parameters list:

```
(define oregonator-run-parameters
  '((starting-dt 1.0e-9)
     etc.
    (graph-window ,*display-graph-window*)))
```

The final `graph-window` parameter indicates to the program that we wish to display the graphed results in the window to which the name `*display-graph-window*` is bound.[4]

[4]This graphics window is created and initialized when the Workbench program is

Having specified that we want a graph of our numeric results, we now need to create a third parameter list that dictates the configuration of the graph:

```
(define oregonator-graph-parameters
  '((graph-type numeric)
    (time-low 0.)
    (time-high 200.)
    (species-to-graph (x y z))
    (species-concentration-boundaries
     ((0. 2.0e-5) (0. 1.0e-3) (0. 1.0e-3)))
    (numeric-limits-for-graphs
     ((-30. -0.4e-5 210. 2.2e-5) (-30. -0.1e-3 210. 1.1e-3)
      (-30. -0.1e-3 210. 1.2e-3)))
    (window-regions-to-use
     ((0 0 400 120) (0 121 400 240) (0 241 400 310)))
    (axis-colors-to-use
     (8 8 8))
    (default-colors (2 3 5))
    (color-procedure
     ,graph-object-use-default-color)
    ))
```

The first parameter indicates that our graph will have numeric abscissa and ordinate values.[5] Before proceeding to a term-by-term explanation of the other parameter values, it is worth seeing what the eventual desired graph looks like in Figure 4.1.

Figure 4.1 shows that we have three separate concentration-versus-time graphs for the species `X`, `Y`, and `Z`.[6] Each of these graphs is displayed in its own portion of the graphics window, and each uses its own ordinate line for reference (though all three graphs use the same abscissa line). Having seen this end result, we can now return to the graph parameters provided to the Workbench. The `time-low`, `time-high`, and `species-to-graph` parameters indi-

started up.

[5] As opposed to symbolic values chosen from a finite set; these non-numeric graphs have not yet been implemented in the program.

[6] The three graphs for `X`, `Y`, and `Z` are in fact shown in three different colors, a fact not indicated by Figure 4.1.

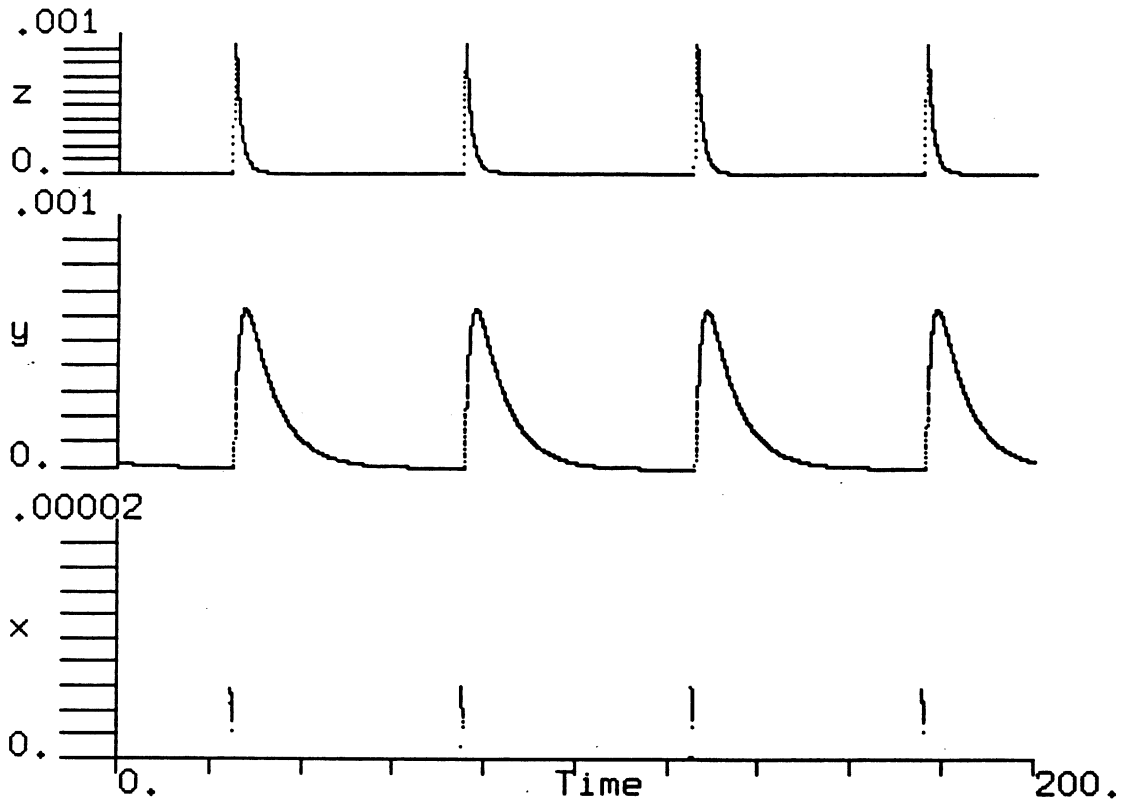


Figure 4.1: Graph of an Oregonator Run

cate that we wish to graph three species over time, and that the abscissa value will run from 0 to 200. The fourth `species-concentration-boundaries` parameter indicates that the ordinate values of our three graphs should run from 0 to 0.00002, 0.001, and 0.001 for the three species `X`, `Y`, and `Z`, respectively. The `numeric-limits-for-graphs` and `window-regions-to-use` parameters indicate that for species X, the rectangular region in the graphics window between (0, 0) and (400, 120) will correspond to the concentration-versus-time region between (-30, -0.000004) and (210, 0.000022), as shown in Figure 4.2. The regions for species `Y` and `Z` are determined similarly; note that the entire graphics window has a width of 420 units and a height of 320 units.

Finally, the last three graphics parameters specify the colors to be used for the concentration graphs. The `axis-colors-to-use` choice indicates that we wish the same color (here, color number 8) to be used for the axes in each of the three graphs. The `default-colors` specifies that the three concentrations will be plotted by default in colors 2, 3, and 5 respectively. Finally, the `color-procedure` choice indicates that we wish to use only default colors in plotting concentration values. (A second "typical" procedure choice will be mentioned in Chapter 7. For now, suffice it to say that the

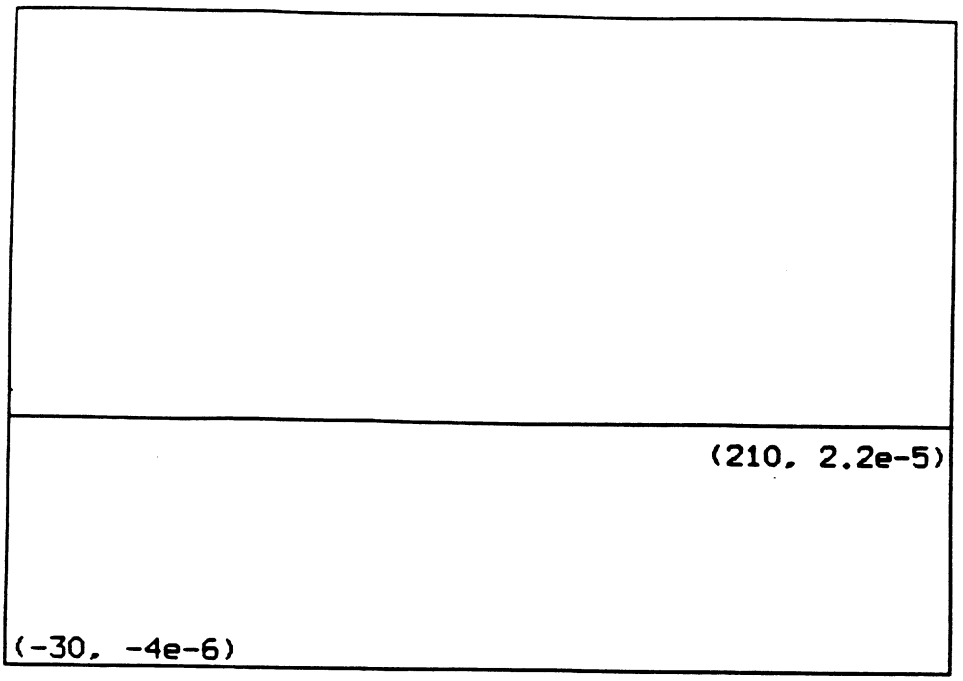


Figure 4.2: Graph Region for Species X

color-procedure choice gives us some flexibility in how concentration-graph colors are chosen: briefly, we can choose a color based on any computable function of the current state of the reaction system.)

Having created the run-parameters and graph-parameters list, we are now able to create the graph shown in Figure 4.1 earlier by evaluating the following expression:

```
(do-simple-graphed-mechanism-run
  oregonator
  oregonator-run-parameters
  oregonator-graph-paraemters)
```

Superficially, the collection of available graphics parameters gives the user a fair amount of flexibility in visualizing concentration data (later, in Chapter 7, we will see an example in which the same species is graphed at two different "magnifications"); but pragmatically, there are only a few typical "cliched" patterns that would likely prove useful. One additional (and very useful) feature, however, is that the user is able to graph functions of state other than simple concentrations.

As an example of this capability, consider the following "tiny mechanism":

```
[4.1]        A --> B   k1 = 20
[4.2]        B --> C   k2 = 2
[4.3]        C --> B   k3 = 1
```

Suppose that we wish to graph, not concentrations, but the ratio of the concentration of C to that of B; examination of the mechanism shows that this ratio should approach a value of 2 over time. We create the appropriate data structure for the mechanism, as well as a run-parameters list with a **graph-window** parameter:

```
(define tiny-mechanism
  '(
      (  ( ((a 1))  ((b 1)) 20)
         ( ((b 1))  ((c 1)) 2)
         ( ((c 1))  ((b 1)) 1) )
      () () () () ()))

(define tiny-run-parameters
  '((starting-dt 0.025)
    (start-time 0.)
    (end-time 10.)
    (actual-starting-concs
      ((a 3.0) (b 0.0) (c 0.0)))
    (integration-method runge-kutta)
    (focus-species (a b c))
    (steps-per-display 40)
    (save-numeric-results? ,false)
    (graph-window ,*display-graph-window*)))
```

And we create a graph-parameters list as follows:

```
(define tiny-graph-parameters
  '((graph-type numeric)
    (time-low 0.)
    (time-high 10.)
    (symbols-to-graph (c/b))
    (species-concentration-boundaries ((0. 6.)))
    (numeric-limits-for-graphs
      ((-2.5 -0.8 11. 4.5)))
    (window-regions-to-use
      ((0 0 400 300)))
    (functions-of-state-to-graph
      ,(list
         (lambda () (let ((conc-c (current-conc-of 'c))
                          (conc-b (current-conc-of 'b)))
                      (list (if (= conc-b 0.) 100. (/ conc-c conc-b))
                            3)))))
    (axis-colors-to-use (8))))
```

Now, by evaluating the following expression:

```
(do-simple-graphed-mechanism-run
  tiny-mechanism
  tiny-run-parameters
  tiny-graph-parameters)
```

we obtain the graph shown in Figure 4.3.

Each function in the list of `functions-of-state-to-graph` is a procedure of no arguments that returns a list of a numeric value and a color value (an integer in the range 0–15). Thus, in this example, we have written a procedure that, at each step, returns a list of the ratio `[C]/[B]`, and the integer 3 (corresponding to green). It is not hard to see that a wide range of function-types is possible with this facility — indeed, any function of state expressible in Scheme may be graphed. It is often desirable, for instance to graph the log of a concentration rather than the concentration itself (particularly when the concentration value has a wide range); by using a function of the form

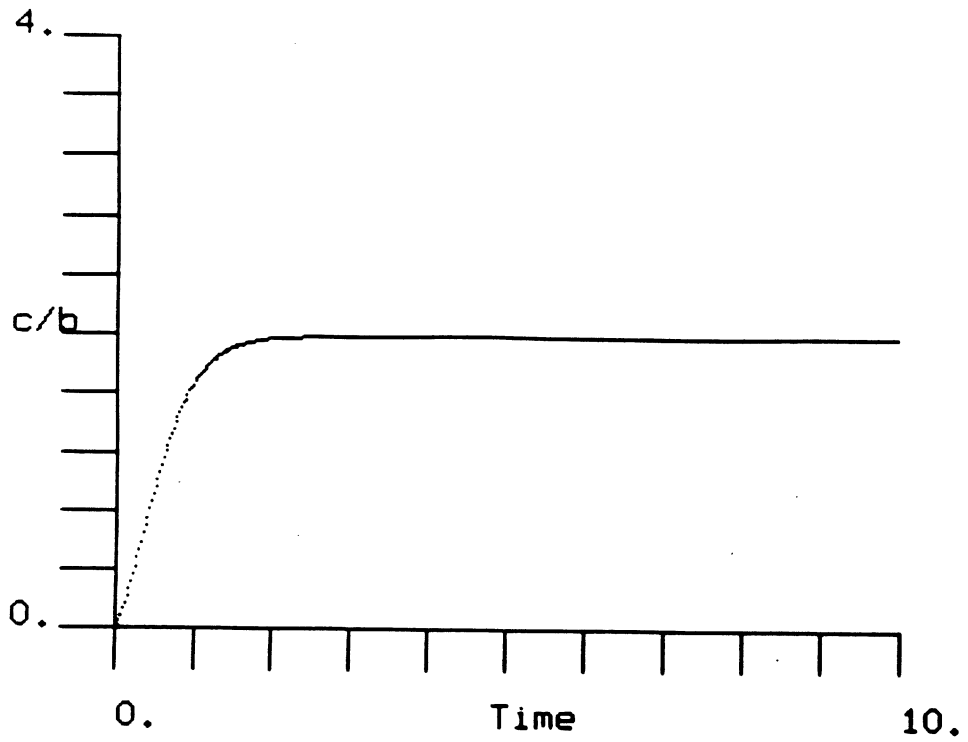


Figure 4.3: Graph of Tiny Mechanism Run

```
(lambda () (list (log (current-conc-of 'x))
                 3))    ; green
```

we can graph log[X] versus time.

Like numeric results, pictorial results may be saved at the end of a run (in a special "PIC" format file that may be retrieved by commands in the X-Windows system). If the user includes a `save-picture?` entry in the run-parameters list (with a value of `true`), and a `pictorial-filename` entry with the appropriate (string) pathname, then at the end of the simulation the Workbench will save the graph of the just-completed run. In the following section, dealing with parametrized mechanisms, we will see an example of this picture-saving capability.

4.5 Parametrized Mechanisms

Often, when exploring a mechanism, the chemist wishes to systematically vary some numerical value and simulate the mechanism repeatedly as this

value varies; in other words, the chemist wishes to employ some value as a *parameter* for the simulation. Typically, this parameter will be either a rate constant or initial concentration: for instance, the chemist might want to examine the behavior of the Brusselator mechanism over a range of possible values of (say) rate constants $k3$ and $k4$. The parametrized mechanism might be expressed as shown below:

```
[5.1]     A --> X              k1 = 1
[5.2]     B + X --> Y + D      k2 = 1
[5.3]     2X + Y --> 3X        k3 = (0.5 - 7.5)
[5.4]     X --> E              k4 = (0.5 - 4.0)
```

The idea here is that $k3$ might fall somewhere in the range of 0.5 to 7.5, and $k4$ in the range of 0.5 to 4.0.

The Workbench allows the user to simulate a mechanism repeatedly over a given range of one or two specified parameter values (which may be rate constants, species coefficients in reactions, or initial concentrations). To do this, the user includes symbolic (as opposed to numeric) entries in the mechanism structure or run-parameters list, and then creates another *parameter range list* that will be used to provide the ranges and parameter increment-size for the Workbench to use. Continuing with the current example, we might imagine that we wish to simulate the Brusselator at the specified ranges of $k3$ and $k4$, using an increment of 1.0 for $k3$ and 0.5 for $k4$; that is, we wish to try $k3$ values of 0.5, 1.5,... 7.5, and $k4$ values of 0.5, 1.0,... 4.0. To do this, we first write a "parametrized" version of the Brusselator:

```
(define double-parametrized-brusselator
  '(
    (
     ( ((a))  ((x)) 1)
     ( ((b) (x)) ((y) (d)) 1)
     ( ((x 2) (y)) ((x 3)) k3)
     ( ((x))  ((e)) k4)
     )

    ((a 1) (b 3)) ;constant species
    () ;sources
    () ;sinks
    (d e) ; driven off
    () ;functions of time
    ))
```

We now create a parameter range list as follows:[7]

```
(define double-brusselator-param-range-list
  '((parameter-name1 k3)
    (parameter-low-value1 0.5)
    (parameter-high-value1 7.5)
    (parameter-increment1 1.0)
    (parameter-name2 k4)
    (parameter-low-value2 0.5)
    (parameter-high-value2 4.0)
    (parameter-increment2 0.5)
    (parameter-list ())))
```

In the current implementation of the Workbench, parametrized runs must be accompanied by filenames in which to save numeric, pictorial, and qualitative-analysis results; thus, we specify "template" filenames in yet another list:[8]

[7]The final entry of this list represents a "symbolic parameter" feature that has not yet been implemented in the Workbench.

[8]Because we have not yet encountered the qualitative analysis module of the Workbench, this chapter has omitted any description of how these results are saved. The details are mentioned in Chapter 6, but in any event the manner in which this analysis-storing is implemented is completely analogous to the numerical and pictorial file-saving

```
(define double-parametrized-brusselator-file-templates
  '("/mydata/brussruns/numbers-"
    "/mydata/brussruns/qualanal-"
    "/mydata/brussruns/pict-"))
```

And now, after constructing appropriate run- and graph-parameters lists (much as in earlier examples in this chapter), we evaluate the following expressions:[9]

```
(define double-parametrized-brusselator-mech-object
        (make-mechanism-object double-parametrized-brusselator))

(do-double-parameter-space-scan-with-graphs
 double-parametrized-brusselator-mech-object
 double-parametrized-brusselator-file-templates
 double-parametrized-brusselator-run-parameters
 double-parametrized-brusselator-graph-parameters
 double-brusselator-param-range-list)
```

The result of this final call is to run, in succession, all (in this case 64) appropriate simulations, storing the numeric, qualitative, and pictorial results in three separate groups of files. (The numeric results, for example, are stored in files named `numbers-0`, `numbers-1`, and so forth.) We will encounter a fuller portrait of this process—as well as a description of how the stored files may now be used for additional analysis—in Chapters 6 and 7 of this paper.

features mentioned earlier in the chapter.

[9]The first expression creates a "mechanism object" of the kind mentioned earlier. We will encounter this data structure again in the next chapter; for now, we merely note that the procedures that perform parameter space scans employ mechanism objects as arguments, rather than the simpler mechanism data structure.

Chapter 5

Analyzing Mechanisms Before Simulation

In the previous chapter, we saw how the Workbench performs numerical simulation; in a sense, that discussion centered around the "standard" use of computers in chemical kinetics. Before the Workbench performs such a simulation, however, it is often capable of making interesting predictions regarding a given mechanism's behavior; and on occasion, the program is capable of suggesting simplifications that might enable the chemist to replace the original mechanism with a smaller, more easily understood variant.

In the first section of this chapter, we describe a variety of "graphical techniques" that the Workbench can bring to bear in analyzing mechanisms. These techniques are distinguished by the fact that they make no use of numerical information (i.e., rate constants), but rather base their analysis solely on the structure of the elementary reactions themselves. The succeeding two sections may be thought of as "introducing numbers into the discussion"; these two sections explain how the Workbench can use both a mechanism's structure *and* rate constants to suggest numerical approximations and locate equilibrium states. Finally, we discuss how the Workbench effectively notates its predictions of mechanism behavior, and we mention some of the current limitations of the program's pre-simulation analysis module.

5.1 Graphical Analysis

5.1.1 Structural Information from Mechanisms

Often there is a great deal that we can tell about the behavior of a mechanism before even having to simulate it. As an extremely simple instance, consider the mechanism below:

```
[1.1]       A --> B       (k1 = 2)

[1.2]       B --> A       (k2 = 1)
```

By inspection, we know that if we start off our simulation with non-zero concentrations of either A or B, the system will eventually reach a stable equilibrium. The rate constants $k1$ and $k2$ dictate the equilibrium ratio [B]eq/[A]eq, and in fact—since the differential equations governing this system are linear—we can derive analytic expressions for the concentrations of both [A] and [B] as functions of time and initial concentrations. In any event, without belaboring the point, we were able to derive key information—namely, that the system does indeed reach equilibrium—without having to use the precise values of the rate constants. Simply by virtue of the structure of the mechanism itself, and without using numerical information, an important aspect of the mechanism's behavior is known beforehand.

A very slight elaboration on this example might be provided by adding a third step:

```
[1.3]     C --> A       (k3 = 0.2)
```

In this case, we know that if the system starts out with a nonzero concentration of A, B, or C, it will over time approach a state in which the concentration of C = 0, and the concentrations of A and B are at equilibrium values (which again may be determined as functions of the rate constants and initial concentrations). We also know—by easy inspection—that if C has an initial nonzero concentration, the concentration of C will decline toward zero throughout the simulation.

It is not always such a straightforward matter to interpret the structural information that the mechanism offers. Consider once again the nitrogen pentoxide decomposition mechanism introduced in Chapter 2:

[2.1] $N_2O_5 \rightarrow NO_2 + NO_3$

[2.2] $NO_2 + NO_3 \rightarrow N_2O_5$

[2.3] $NO_2 + NO_3 \rightarrow NO_2 + NO + O_2$

[2.4] $NO + N_2O_5 \rightarrow 3NO_2$

One might ask, can such a mechanism achieve an equilibrium state in which all species have nonzero concentration? If not, which species will approach a concentration of zero? And again if not, will there be an equilibrium state between the remaining nonzero-concentration species (as there was between A and B in mechanism [1.1]-[1.3])?

We will return to this example later, so an analysis of this mechanism will be postponed; mechanism [2.1]-[2.4] is in any case only slightly trickier than the previous ones. A still more subtle example is shown below:{32}

[3.1] $D \rightarrow A + B$

[3.2] $D \rightarrow 2C$

[3.3] $2C \rightarrow D$

[3.4] $B + C \rightarrow E$

[3.5] $E \rightarrow B + C$

[3.6] $E \rightarrow A + D$

[3.7] $A + D \rightarrow E$

Staring at mechanism [3.1]-[3.7] for a while, we discover that every species is both a reactant and product in some elementary step. Thus, unlike [1.1]-[1.3], in which C obviously declined toward zero concentration, it is far from obvious whether any species here will approach zero concentration. Perhaps this mechanism will achieve equilibrium with all species at nonzero concentration; indeed we may wonder whether the mechanism is not perhaps capable of still more exotic behavior such as oscillations. As with [2.1]-[2.4], we will return to this mechanism later in this chapter, but for now it is worth stressing that

the process of gleaning structural information from mechanisms is not merely a matter of restating the intuitively apparent.

The next several subsections present a variety of techniques for analyzing mechanism structure; once we have acquired this collection of techniques, we will return to the two previous examples and see how the Workbench program performs on them.

5.1.2 Necessary Nonzero Species

In Chapter 2, it was stated that mechanisms are often endowed with certain special features that reflect built-in assumptions (or approximations) about the reacting system: for instance, it might be stipulated that a given species is supplied at a constant rate from outside the system, or that a given species is held at a constant concentration. (The Brusselator mechanism, with two "constant species," illustrates the second of these special features.) In such cases, it is obvious that the constant or externally-supplied species must have nonzero concentration in any asymptotically-approached state of the system; we call such species "necessary nonzero species." Moreover, any species produced in a reaction whose reactants are all necessary nonzero species is itself a necessary nonzero species. Thus, if we have the set of reactions:

```
[4.1]        A --> B

[4.2]   B + C --> D

[4.3]   D + E --> F + B
```

and if, in addition, we are informed that species A is held at a constant (nonzero) concentration, then we can conclude that both A and B are necessary nonzero species. If we are told additionally that there is an external source for species C, then we know that A, B, C, and D are all necessary nonzero species (but not E and F).

Note that the concept of necessary nonzero species only tells us that certain species will be present *if* the system should approach some equilibrium state. If it does not, then there is little that we can say about these species.

For instance, in the case of the Brusselator mechanism, both X and Y are necessary nonzero species:

```
[5.1]     A --> X              k1 = 1

[5.2]     B + X --> Y + D      k2 = 1

[5.3]     2X + Y --> 3X        k3 = 1

[5.4]     X --> E              k4 = 1
```

However, it happens that with this choice of rate constants, the system approaches a stable limit cycle; so our classification of X and Y is not especially useful. (There are, however, other choices of rate constants for this system in which an equilibrium state is reached, and in these cases, both X and Y naturally have nonzero concentration.)

5.1.3 Obvious Declining Sets of Species

Consider the following sample mechanism:

```
[6.1]          A + B --> C

[6.2]              C --> D

[6.3]              D --> C
```

As with the earlier mechanism [3.1]-[3.3], we can tell "by inspection" that either A or B (or both) must asymptotically approach a zero concentration in a system in which all species initially have nonzero concentrations. What makes this conclusion obvious is simply that A and B appear as reactants but not products in this system. We might say, then, that the set of species [A B] is an "obviously declining set"—that is, the product of the concentrations of the species in the set must approach zero, given enough time.

As in the case of nonzero species, we might likewise conclude that any set of species whose presence in the system is exclusively determined by the presence of a declining set is itself a declining set. This reasoning, when

applied to the following mechanism:[1]

```
[7]        A --> B --> C --> D <--> E
```

would enable us to conclude that over time, the concentrations of A, B, and C must all approach zero. Again, this is straightforward, but subtler examples can be found. For instance, consider the following mechanism:

```
[8]        A + B --> C <--> D

           A <--> E

           B <--> F
```

At least one (but quite possibly not both) of E and F must approach a concentration of zero in this system. Thus, the set [E F] is a declining set; and if we augment this mechanism with the additional steps

```
[9]        E + F <--> G
```

then we can conclude that the singleton set [G] is likewise a declining set, since if there is some asymptotically-approached state in which G has a nonzero concentration it must likewise be the case that the product of the concentrations of E and F is also nonzero.

5.1.4 Consistent Zero Sets of Species

It is often of interest, for some particular mechanism, to answer a question of the form, "If species X is at zero concentration in some asymptotically-approached state of the system, which other species must likewise have zero concentration in that state?" Or, as a similar question, we might ask, "Is it possible for the set of species X, Y, and Z—and no others—to have zero concentration in some asymptotically-approached state?" In both cases we are

[1]Mechanism [7] is presented in a more compact "graphical" format, without rate constants; we will often employ this format for convenience, particularly when the specific values of the rate constants are immaterial to the discussion.

asking about which sets of species may collectively have zero concentration at some asymptotically-approached state.

Consider, for example, the mechanism

```
[10]          A <--> B <--> C
```

In this case, if any one species has zero concentration at a stable state, then all must; thus, the only consistent set of zero-concentration species is the set `[A B C]`. The reasoning used here is simply that if a product of some reaction has zero concentration at a stable state, then at least one of the reactants of that reaction must also have zero concentration.

A slightly more elaborate example is the following:

```
[11]          A + B <--> C + D <--> E
```

Here, if `A` has zero concentration at some stable state, then so must `E` and at least one of `C` and `D`; but it is possible that `B` (and at least one of `C` and `D`) has nonzero concentration. Thus, consistent sets of zero-concentration species include `[A C E]` and `[A D E]`, along with sets of which these two are subsets.

Taking this last example a step further, we might now ask for all the possible (internally consistent) sets of zero-concentration species. We have already found that `[A C E]` and `[A D E]` are the only such (minimal) sets that include species `A`; starting with species `B`, we likewise find that `[B C E]` and `[B D E]` are good candidates. As it happens, these four sets (along with supersets of any of them) comprise a complete list of internally consistent zero-concentration sets. Thus, if we are told only that *some* species in mechanism [11] is at zero concentration, and that the system is in a stable state, then we know that in fact at least three species are at zero concentration, and that these three must be specified by one of the four possible sets that we have discovered.

5.1.5 The Zero Deficiency Theorem

More sophisticated, but also more powerful, than any of the techniques discussed so far is the "Zero Deficiency Theorem," derived by M. Feinberg,

F. Horn, R. Jackson and colleagues at the University of Rochester.{32} The statement of the theorem requires some auxiliary definitions, but because of the theorem's power and interest it is worth digressing here to present it in outline.[2]

We begin by representing a mechanism in graphical notation, in the same form as mechanisms [7]-[11] above. As a specific example, consider the following mechanism:

```
[12]        A + B <--> C <--> D
```

Now, we let n denote the number of distinct vertices (i.e., left or right sides of reaction arrows) in the graph above; these will be called *complexes*. In our example, there are three complexes—represented as `C`, `D`, and `A + B`. We next let l denote the number of connected components in the graph. In this case, the graph consists of one component, so the value of l is 1. Let m denote the number of species found on either side of a reaction in the graph; in this example, m is 4 (corresponding to the species `A`, `B`, `C`, and `D`).

Finally, for each reaction (directed edge) in the graph we will create a representative vector in R^m. We imagine that each coordinate of R^m corresponds to one of the species in our graph: in our current example, we will let the first coordinate correspond to species `A`, the second to `B`, and so on. The vector that we will create for a given reaction will have negative coordinate values corresponding to the coefficients of each reactant, and positive coordinate values corresponding to each product. To take one instance, the reaction `A + B --> C` is represented by the vector (-1 -1 1 0). The other three reactions in our system would be represented by the vectors (1 1 -1 0), (0 0 -1 1), and (0 0 1 -1). We let s denote the dimension of the linear space spanned by these four vectors; in our current example, s has a value of 2. (This quantity is also called the *dimension of the stoichiometric subspace* for the mechanism.) Yet another way of phrasing the definition of s is to say that it is the rank of the matrix of reaction vectors:

```
-1   -1    1   0
 1    1   -1   0
 0    0   -1   1
 0    0    1  -1
```

[2]A brief sketch of the theorem's proof is provided in Appendix A.

Having defined all these quantities, we now check the value of the expression $n - l - s$. (Feinberg {32} calls this quantity the *deficiency* and denotes it by the symbol *d*.) For many reaction mechanisms in the literature, and for our current example, this expression has a value of 0. If the mechanism has zero deficiency, we then go on to check the graph for one more property called "weak reversibility": specifically, the graph is weakly reversible if every pair of vertices $V1$ and $V2$ connected by a reaction pathway from $V1$ to $V2$ is *also* connected by a reaction pathway from $V2$ to $V1$. In our current example, the graph is weakly reversible: for instance, the two vertices represented by `A + B` and `D` are connected by the pathways

```
A + B --> C --> D
```

and by

```
D --> C --> A + B
```

so that there is a "forward" and "backward" reaction path between the two vertices.[3]

The Zero Deficiency Theorem now states that if a mechanism has a deficiency of zero, then the presence of a unique equilibrium state in which all concentrations are positive is determined by whether or not the graph is weakly reversible. If the graph is weakly reversible, then the mechanism, when started from some initial state, gives rise to exactly one stable equilibrium state in which all concentrations are positive; if not, then the mechanism cannot give rise a stable equilibrium state in which all concentrations are positive.[4]

[3]The graph in our example has, in fact, a slightly stronger property than weak reversibility: each reaction is accompanied by a "reverse reaction." (Feinberg {32} calls this "strong reversibility.") Thus, any path between vertices $V1$ and $V2$ will be matched by a "reverse" path from $V2$ to $V1$. Strong reversibility implies weak reversibility, but the converse is not true.

[4]There is actually a bit more to the statement of the theorem than is presented here. For instance, even if a zero-deficiency mechanism is weakly reversible, that does not mean that *every* initial state will attain equilibrium with all positive concentrations; it merely means that there exists one such equilibrium state and that it is locally stable. Another

To return to our example, we know that we have a weakly reversible, zero-deficiency mechanism. Hence, the Zero Deficiency Theorem tells us that for any given initial state, there is exactly one stable equilibrium state in which all concentrations are nonzero. If we perturb our original mechanism by imagining that the step `A + B --> C` is irreversible, we obtain the graph

```
[13]          A + B --> C <--> D
```

Our new mechanism is similar to the previous one, but we have deleted one reaction. The deficiency of the new mechanism is still zero, but the graph is no longer weakly reversible. Hence, there is no stable equilibrium state (given any set of initial concentrations) in which the concentrations of all species are nonzero. Indeed, this is the same mechanism [6.1]-[6.3] that we looked at earlier, and for which we concluded that either A or B (or both) would asymptotically approach a concentration of zero.

It may seem that for the examples [12] and [13], the Zero Deficiency Theorem demands a great deal of terminology to confirm the obvious: after all, it is not hard to conclude by inspection that mechanism [12] will reach equilibrium with nonzero concentrations, and that mechanism [13] will not. But there are many mechanisms for which the Zero Deficiency Theorem dictates non-obvious conclusions. Consider once again the mechanism [3.1]-[3.7] (taken from Feinberg {32}), represented in graphical form below:

```
[14]          A + B <-- D <--> 2C

              B + C <--> E <--> A + D
```

This is a zero deficiency mechanism whose graph is not weakly reversible; hence the Zero Deficiency Theorem dictates that the mechanism cannot reach equilibrium with all species at nonzero concentrations. As we discovered earlier, this conclusion is by no means apparent by casual inspection of the graph. A similar conclusion may be reached for the dinitrogen pentoxide decomposition reaction [2.1]-[2.4]; again we have a zero deficiency mechanism that is not weakly reversible, and again the theorem tells us that an equi-

point worth mentioning is that we are implicitly making use of the fact that all mechanisms are assumed to obey mass action kinetics.

librium state (with all nonzero concentrations) is impossible. We will return yet again to both of these examples later in this chapter.

There are still several more points to mention about the application of the Zero Deficiency Theorem. First, the treatment of constant-concentration, source, and sink species must be handled specially in the construction of our graphical representation. Constant species cannot be included in any complex; rather they must be incorporated into an "effective rate constant" for any step in which they act as reactant. Thus, considering mechanism [12] once more:

```
[12]      A + B <--> C <--> D
```

if we now stipulate that species A is held at constant concentration, the graphical representation for the mechanism becomes

```
[12']     B <--> C <--> D
```

Here, we have subtracted the constant species A from the complex A + B; and we have effectively rewritten the original bimolecular step:

```
A + B --> C k1
```

as a unimolecular step

```
B --> C     k1' = k1 * [A]const
```

It should be apparent that this formal transformation of the elementary step has no effect on the actual differential equations governing the mechanism [12], as long as the concentration of A is indeed constant. A slightly more complicated transformation is required if we stipulate that species D (rather than A) will be held constant in mechanism [12]. In this case, the complex consisting of the single species D is replaced formally by a special "zero complex," and the two elementary steps

```
C --> D       k2

D --> C       k3
```

are replaced by

```
C --> 0        k2

0 --> C        k3 * [D]
```

This notation is consistent with the definition of the stoichiometric subspace dimension s shown earlier, and again the differential equations governing the system are unaffected. The new graph for the (updated) mechanism would now be:

```
[12'']    A + B <--> C <--> 0
```

It is not hard to see, parenthetically, that both mechanisms [12'] and [12"] are zero-deficiency, weakly (indeed, strongly) reversible mechanisms and thus meet the conditions for having a unique locally stable equilibrium state with nonzero concentrations for any given set of initial conditions.

As for external sources and sinks, these are treated by the use of the same "zero complex" introduced in the previous paragraphs. For instance, an external source for a species A would be represented by the elementary step:

```
0 --> A        ksource
```

where k_{source} is the product of the source concentration of A and the rate (in time^{-1}) at which the source is added to the reacting system. Similarly, a "sink" step for A is represented as:

```
A --> 0        ksink
```

where k_{sink} is the rate (in time^{-1}) at which the reactive system is removed to the external sink. Again, a comparison of this notation for "special steps" with the more explicit version introduced in Chapter 2 (and represented directly in the mechanism data structure via the presence of "constant" "source" and "sink" fields) shows that the mathematical treatment of mechanisms remains unaffected by the new formalism.

As a final example it is worth noting how we would treat the Brusselator mechanism [5.1]-[5.4], and reproduced below:

```
[5.1]     A --> X              k1 = 1

[5.2]     B + X --> Y + D      k2 = 1

[5.3]     2X + Y --> 3X        k3 = 1

[5.4]     X --> E              k4 = 1
```

Since both A and B are treated as constant species, we first rewrite the mechanism as follows:

```
[15.1]     0 --> X             k1' = 1M * k1

[15.2]     X --> Y + D         k2' = 3M * k2

[15.3]    2X + Y --> 3X        k3  = 1

[15.4]     X --> E             k4  = 1
```

We now note that since both D and E are treated as "driven off," i.e., having constant zero concentrations, we can further rewrite the mechanism:

```
[16.1]     0 --> X             k1' = 1M * k1

[16.2]     X --> Y             k2' = 3M * k2

[16.3]    2X + Y --> 3X        k3  = 1

[16.4]     X --> 0             k4  = 1
```

And now, by noting that identical complexes comprise a single vertex in our "mechanism graph," we arrive at a final graphical form for the Brusselator mechanism:

```
[17]      0 <--> X --> Y

          2X + Y --> 3X
```

The number of complexes, n, for this graph is 5; the number of linkage classes l is 2; and the dimension of the stoichiometric subspace is the rank of the matrix

```
-1    0
 1    0
-1    1
 1   -1
```

which is 2. Thus the deficiency of the overall mechanism is 1, and the Zero Deficiency Theorem does not apply; we can make no predictions about the necessary presence or absence of an equilibrium state for this system. Indeed, we know that the Brusselator is capable of "non-standard" behavior such as oscillation and that there are parameter values at which an unstable equilibrium state exists (and the system has a stable limit cycle).

5.1.6 Submechanisms

In analyzing the mechanism [12]

```
A + B --> C <--> D
```

we noted that the asymptotically-approached state of the system must include nonzero concentrations of C and D. The way in which such a conclusion is derived is by noting that at least one of A and B must approach zero concentration, and then considering the "remaining" portion of the mechanism once the concentration of A or B approaches a near-zero value:

```
[18]          C <--> D
```

This remaining "submechanism" can be analyzed on its own, using the very same concepts introduced earlier. For instance, by the Zero Deficiency Theorem we can see that this simple mechanism possesses a unique equilibrium state with positive concentrations (given some starting state); and since the only consistent set of zero-concentration species for this submechanism

is [C D], we can further conclude that the only possible asymptotically-approached states for this system are either the state in which C and D are at zero concentration, or the state in which both are at their equilibrium concentrations.

This example illustrates the notion of a "submechanism," a collection of reactions that remain operative after a certain set of species attains zero concentration. It is not hard to devise mechanisms with several independent submechanisms, or alternatively to devise submechanisms which themselves evolve to (second-level) submechanisms, and so forth. We will also encounter this notion in discussing mechanisms with "fast" components later in this chapter.

5.1.7 Examples with the Workbench

The current version of the Workbench uses the mechanism data structure described in the previous chapter to create a more "textured" *mechanism object* that contains (besides the original mechanism) a large number of special fields that serve as annotation to the mechanism. Many of these fields relate to the techniques of graphical analysis discussed in this section. In particular, the program derives, for the given mechanism:[5]

- Necessary nonzero species for the mechanism.
- Some (though not necessarily all) sets of declining species for the mechanism.
- All consistent sets of species with zero concentration for the mechanism.
- The deficiency of the mechanism, along with other useful graph-theoretic properties (e.g., strong and weak reversibility); and various conclusions drawn from these.
- Submechanisms derived from the original mechanism, corresponding to "remaining" portions of the mechanism after reac-

[5]The algorithms used by the Workbench for these deductions are summarized in Appendix B at the end of this paper.

tions involving declining sets have been deleted.

There are still other graphical techniques employed by the Workbench; these relate to "special properties" of certain mechanisms, and will be discussed later in this chapter. Before proceeding to a discussion of these additional techniques, however, it is worth pausing to examine the Workbench's performance on several illustrative examples; this will give the reader a feeling for the kinds of deductions that the program is able to make when presented with a mechanism.

As an initial example, we consider the simple mechanism [12] with the stipulation that species A is held at constant concentration:

```
[12]          A + B --> C <--> D
```

We first create a mechanism data structure for [12] by evaluating the following expression:

```
(define chap5example1
  '(
    ( ( ((a 1) (b 1)) ((c 1)) 1)
      ( ((c 1)) ((d 1)) 1)
      ( ((d 1)) ((c 1)) 1) )
    ((a 2.0)) () () () () ))
```

Now we can ask the Workbench for an analysis of this mechanism by evaluating a call to the `analyze-mechanism-graphically` procedure:

```
(analyze-mechanism-graphically chap5example1)
```

The Workbench responds first by asking the user if he wishes to display the mechanism graphically on the screen; if the user answers "yes," the representation shown in Figure 5.1 is drawn in a screen graphics window.[6] It is worth noting that Figure 5.1 indicates a bit of initial analysis on the part of

[6]The figures in this chapter—and on several other occasions in this paper—have been

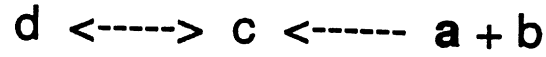


Figure 5.1: A Graph of Mechanism [12]

the Workbench: necessary nonzero species (here, A) are shown in red, while the rest of the mechanism is shown in green.

The Workbench now prints out a thorough (and somewhat verbose) analysis, of which excerpts follow:

```
Number of complexes: 3
Number of linkage classes: 1
Dimension of stoichiometric subspace: 2
Deficiency: 0
Reversibility: none

Rule number 2.3 -- Deficiency Theorem part 2 is firing

If the deficiency of the mechanism is 0, and the mechanism
is not weakly reversible, then the mechanism cannot reach
equilibrium at positive concentrations of all species.


Rule number 3.1 -- Searching for Species to Drop from System is firing

If the zero deficiency theorem stipulates that the mechanism
```

translated into reproducible (black-and-white) form from original Workbench output, which often uses color-coding instead to convey information. In Figure 5.1, for instance, the boldface a represents a species shown in red; normal font corresponds to the color green in the actual Workbench output. All such "de-colorized" figures will be noted as they occur.

```
does not reach equilibrium, then we first examine the
mechanism for species that are reactants only, and hence
cannot enter into any reactions that occur at steady state.
Sequence of Species to Drop: ((b))


Rule number 3.2 -- Creating and Examining Submechanism is firing

We have found that there are species that we can drop from
the original reaction mechanism. We now examine the smaller
mechanism defined by ignoring all reactions with droppable
species among the reactants.

Number of complexes: 2
Number of linkage classes: 1
Dimension of stoichiometric subspace: 1
Deficiency: 0
Reversibility: strong

Rule number 2.1 -- Deficiency Theorem part 1 is firing

If the deficiency of the mechanism is 0 and the
mechanism is weakly reversible, then the mechanism must
reach equilibrium at positive (non-zero) concentrations
of all species.
```

In summary: the Workbench has deduced that over time, the mechanism can be well approximated by the submechanism

```
C <--> D
```

and that this mechanism has a stable equilibrium state in which both species have nonzero concentrations. Having printed out this analysis, the program also shows the submechanism on the screen by "graying out" part of the original mechanism, as shown in Figure 5.2. The "dropped species" B, and the reaction arrow for which B is a reactant, is removed from the system and only the remaining two reactions are shown in green.

As a second example, we return to the nitrogen pentoxide decomposition reaction [2.1]-[2.4]. We create the mechanism in the usual way:

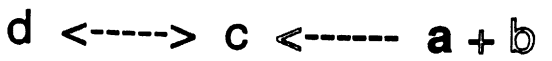


Figure 5.2: Dropped Species in Mechanism [12]
In the original Workbench output, a is shown in red, b and the rightmost arrow (and plus sign) in gray, and all others in green.

```
(define n2o5-decomposition
  '(
    (
     ( ((n2o5 1)) ((no2 1) (no3 1)) 0.002)
     ( ((no2 1)(no3 1)) ((n2o5 1)) 0.001)
     ( ((no2 1)(no3 1)) ((no2 1)(no 1) (o2 1)) 30.)
     ( ((no 1) (n2o5 1)) ((no2 3)) 4000.)
     )
    () () () () ()))
```

Now we request a graphical analysis of this mechanism from the Workbench:

```
(analyze-mechanism-graphically n2o5-decomposition)
```

The mechanism is displayed graphically as in Figure 5.3, and the following analysis is displayed (again, the output has been slightly abridged for readability):

```
Number of complexes: 5
Number of linkage classes: 2
Dimension of stoichiometric subspace: 3
Deficiency: 0
Reversibility: none

Rule number 2.3 -- Deficiency Theorem part 2 is firing
```

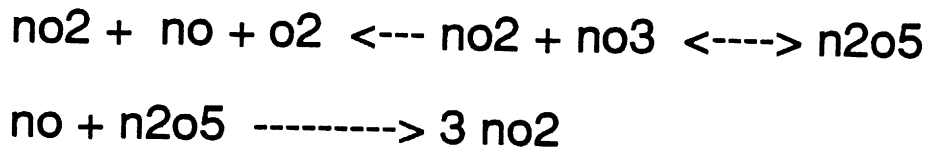


Figure 5.3: Graph of N2O5 Decomposition
All species are shown in green in the original Workbench output.

```
If the deficiency of the mechanism is 0, and the mechanism
is not weakly reversible, then the mechanism cannot reach
equilibrium at positive concentrations of all species.


Rule number 4.3 -- Looking for asymptotic zero concentrations is firing

The mechanism cannot reach equilibrium with all nonzero concentrations.
It also does not contain any *obvious* declining species or sets.
We now look for those sets of species that might asymptotically
have zero concentration.
Possible sets of zero-concentration species: ((no3 n2o5) (no2 n2o5))
```

The program now alters the display of the mechanism to show each of the possible sets of zero-concentration species (highlighting these species in magenta). Thus, the mechanism is first displayed as in Figure 5.4 then as in Figure 5.5.

To summarize the example thus far in prose: the Workbench has announced that the mechanism cannot reach equilibrium with all species at nonzero concentrations. It was also unable to find any species (like B in the previous example) that clearly approach zero concentration over time—in this mechanism, every reactant species is also a product. Nevertheless, we know that *some* species must approach a zero concentration; therefore, the program looks for those sets of species that are possible candidates for

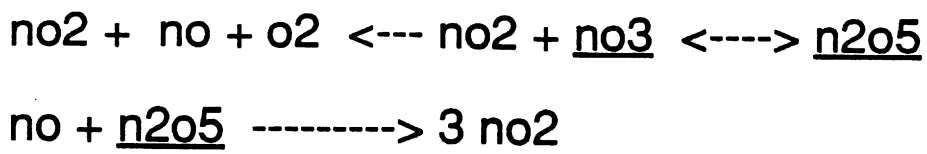


Figure 5.4: One Zero-Set in N2O5 Decomposition
In the original Workbench output, underlined species
are shown in magenta, all others in green.

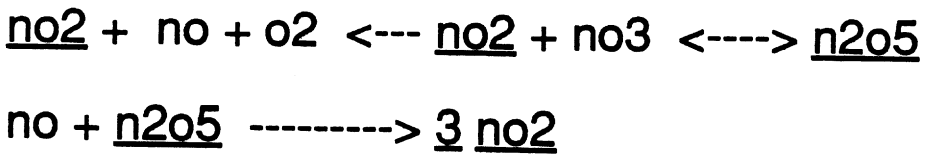


Figure 5.5: Second Zero-Set N2O5 Decomposition
In the original Workbench output, underlined species
are shown in magenta, all others in green.

achieving a concentration of zero over time. It finds two such sets, consisting of the species N2O5 and either NO2 or NO3.[7]

The Workbench continues its analysis by attempting to subtract each of the two possible sets of zero-concentration species, to see whether the remaining steps form a submechanism that can itself be examined:

```
Subtracting the following zero set: (no3 n2o5)
This mechanism contains no elementary steps.

Subtracting the following zero set: (no2 n2o5)
This mechanism contains no elementary steps.
```

The program has thus found that under either assumption—whether N2O5 and NO2 or N2O5 and NO3 are both at zero concentration—there are no elementary steps in the remaining system (i.e., no steps with all reactants at nonzero concentration).

There is still a bit more that the program is capable of doing at this point: depending on rate constants, the Workbench can suggest certain approximations, and it can also annotate a previously-created "mechanism object" with the results of its graphical analysis. We will return to these topics shortly.

Yet another example of a zero-deficiency mechanism is [3.1]-[3.7], cited earlier as an instance of a mechanism whose behavior is difficult to predict by casual inspection. The Workbench's analysis of this mechanism is similar to that of the N2O5 decomposition; briefly, the program finds that this mechanism cannot reach equilibrium with all nonzero concentrations, and then looks for possible sets of species that might have zero concentration:

```
The mechanism cannot reach equilibrium with all nonzero concentrations.
It also does not contain any *obvious* declining species or sets.
We now look for those sets of species that might asymptotically
have zero concentration.
Possible sets of zero-concentration species: ((d e c))

Subtracting the following zero set: (d e c)
```

[7] In point of fact, only the second of these two possibilities is realistic, but the Workbench's analysis is not sufficiently sophisticated to rule out the former possibility.

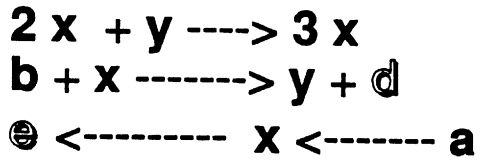


Figure 5.6: Brusselator Graph
Species in boldface—a, b, x, and y—are shown in red in the original Workbench output. "Outline-font" species d and e are shown in blue.

```
This mechanism contains no elementary steps.
```

Here, there is only one possible set of zero-concentration species—namely, [D E C]. As in the preceding example, the Workbench looks to see whether, once these three species all achieve zero concentration, there are any remaining elementary steps that might constitute a submechanism; and it finds that there are none. Again, it is worth looking back at the original mechanism to confirm that the Workbench has revealed something interesting and non-obvious: namely, that this mechanism cannot reach equilibrium, and that it will eventually "run down" with zero concentrations of the species C, D, and E. It is also worth noting that the Zero Deficiency Theorem *alone* is not able to tell us anything more about the mechanism than that it has no equilibrium state in which all species have nonzero concentration; by combining this knowledge with additional analysis, the Workbench is able to predict precisely *which* species are bound to achieve zero concentration.

There are, of course, cases in which the program cannot make any firm predictions about behavior. One such case is the Brusselator mechanism: the graph produced by the program is shown in Figure 5.6 (the color coding indicates that A, B, X, and Y are all "necessary nonzero species" and that D and E are "driven-off" species). The printed output produced by the program reveals that the mechanism has a deficiency of one, and thus that the Zero Deficiency Theorem does not apply:

```
Number of complexes: 5
Number of linkage classes: 2
Dimension of stoichiometric subspace: 2
Deficiency: 1
Reversibility: none
```

Even in this case, the Workbench is able to say at least a bit more, relating to the presence of autocatalysis in the mechanism; we will return to this shortly.

It should be noted that the most typical way of using the Workbench is not to use the `analyze-mechanism-graphically` procedure as in these examples, but rather to first create a *mechanism object* (as mentioned earlier in this chapter) from the original mechanism, and to use the procedure `analyze-graphically` on this:

```
(define brusselator-mech-object
  (make-mechanism-object brusselator))

(analyze-graphically brusselator-mech-object)
```

A mechanism object is a data structure containing not only the original mechanism, but a collection of additional methods and data fields; some of these fields are used by the Workbench to store the results of its graphical analysis. (For instance, there is a `deficiency` field whose value is set by this analysis.) Thus, the point of creating a mechanism object first, as we have just done, is that we allow the Workbench to annotate the mechanism object with its graphical results; and, as will be illustrated, the program can often make use of this extra "annotating" information in its future work with the mechanism.

5.1.8 Special Structural Features of Mechanisms

Occasionally, a mechanism will have some structural feature that is worth noting: such a feature might permit even stronger predictions than the kind we have seen so far, or it might give us a little more information about a complex mechanism. An example of the former can be seen in the following mechanism:

```
[19]    0 <---> A <---> 2B
```

Here, we have a mechanism in which there is an external source and sink for species A, and the elementary reactions

```
A --> 2B

2B --> A
```

We have expressed [19] in the formalism appropriate to the application of the Zero Deficiency Theorem, because this is a zero-deficiency mechanism with an additional special property—namely, the stoichiometric subspace of the system has the same dimension (2) as the overall state space. This implies that the unique equilibrium stipulated for the mechanism [19] is independent of starting concentrations of A and B—no matter what values of [A] and [B] we start out with, there is only one stable equilibrium for the system. Thus, we might be interested in knowing, for mechanism [19], what its global equilibrium state is (without necessarily bothering to find this value by numerical simulation of the mechanism). We will pursue this idea later in the chapter, but for now it is worth mentioning that this property of some zero-deficiency mechanisms is of potential interest, and is noted by the Workbench.

An additional "simplifying" feature is that some mechanisms happen to give rise to linear differential equations. This occurs precisely when each elementary step in the mechanism has at most one (non-constant) species:

```
[20]          A + B  <---> C  <---> D

              [A] constant
```

The differential equations created from this system are:

$$\frac{d[B]}{dt} = -k1[A][B] + k2[C]$$

$$\frac{d[C]}{dt} = k1[A][B] - (k2 + k3)[C] + k4[D]$$

$$\frac{d[D]}{dt} = k3[C] - k4[D]$$

and (recalling that [A] is constant) each of these is a linear equation. Again, since linear systems are mathematically tractable, by noting that a mechanism has this property we automatically have the benefit of linear system theory in predicting its behavior.[8]

Mechanism [20] will give rise to a stable equilibrium in which the concentrations are dependent both on rate constants and initial concentrations; in this sense it is different from mechanism [19]. An especially tractable case is one in which the system is linear *and* has a unique equilibrium independent of initial conditions:

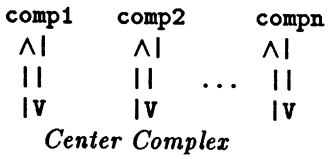
```
[21]          0 <---> A <---> B
```

[8] As an aside, it is not true that all linear systems have a deficiency of zero; counterexamples are easily constructed. Thus, the tractability of linear systems is not conceptually reducible to the tractability of zero-deficiency systems.

In this case we have a linear system in which the equilibrium concentrations may be determined from the rate constants alone. Again, this is a special case noted by the Workbench; we will return to this topic later in the chapter.

A rare type of mechanism structure mentioned in Feinberg {32} is a *reversible star-like* structure in which a single complex is either the reactant or product complex for every elementary step:

[22]

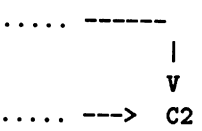


It happens that mechanisms of this structure, regardless of their deficiency, behave like reversible zero-deficiency mechanisms in that they have a unique positive stable equilibrium value. This too is a special feature noted by the Workbench.

Thus far, the examples shown in this section—linear systems, star mechanisms, and so forth—have allowed for special (and powerful) predictions to be made. There are other structural features of mechanisms that are more heuristic in nature: they are worth noting because they are often important elements in the chemist's description of the mechanism's behavior. In other words, we don't necessarily know whether these features will play a major role in the behavior of the mechanism, but it is nonetheless worth flagging them at the outset.

An example of a feature of this kind is the presence of "competing reaction pathways"—that is, pathways of the form illustrated by [23] below:

[23]



Here, there are two sequences of reactions leading from complex $C1$ to complex $C2$, with no shared complexes in between. This feature is always present

in reversible cyclic reactions[9], or may occasionally indicate that we are comparing "catalyzed" and "uncatalyzed" reaction pathways (with constant concentration of the catalyst).[10] As with the other features described in this section, the Workbench spots competing pathways in a given mechanism and annotates the mechanism object with this observation.

A more general type of pattern involves catalytic pathways in which the catalyst is not assumed to be at constant concentration. An illustrative example of this pattern is seen in the standard textbook representation of enzyme catalysis:

```
[24]          E + S <---> X  ---> E + P
```

In mechanism [24], E represents an enzyme, S a substrate, X an "enzyme-substrate complex," and P a product. The important structural aspect of [24] is that the enzyme E is left unaffected by the overall transition from substrate to product, and thus may react with additional substrate (the common assumption in such reactions is that [E] $\ll$ [S]).[11] Patterns of this sort, in which a reaction sequence of three steps or less leaves one of the original reactants unchanged, are noted by the Workbench. (Details of the algorithm are given in Appendix B.)

Autocatalysis is another "heuristic" structural feature; it is common in oscillating reaction mechanisms. The basic notion behind autocatalysis is that a certain amount of species A can be used to create more of species A:

```
A ---> .... ---> 2A
```

A specific example can be seen in the Brusselator, in which the third step is autocatalytic for species X:

```
2X + Y ---> 3X
```

[9]An example can be seen in {74}, problem 2.7, p. 104

[10]Compare, for instance, steps (22) and (18) in {74}, p. 147.

[11]See for instance {74}, pp. 190-193.

In other mechanisms, the autocatalysis may be the result of two or more consecutive steps. Again, the Workbench can spot such sequences when they consist of three or fewer steps. (Again, details are provided in Appendix B.)

5.2 Numerical/Graphical Analysis

All of the examples of graphical analysis shown thus far share an important feature: they rely only on the reaction structure of the mechanism, and not on the particular values of rate constants. We know, for instance, whether a mechanism satisfies the conditions of the Zero Deficiency Theorem simply by examining which reactants go to which products in the various elementary steps; and, if the mechanism is reversible and has zero deficiency, we know that a stable equilibrium with positive concentrations does exist, but we do not have any information about how long the system might take to reach that equilibrium from a given starting point. Similarly, such features as "consistent zero concentration sets" and "catalytic pathways" are derived from reaction structure without regard to particular rate constants. This "minimalist approach" to mechanism analysis is both a strength and a weakness: on the one hand, we are able to deduce important facts about the mechanism without recourse to blatantly numerical reasoning.[12] On the other hand, the types of deductions that we reach from structure alone generally relate to limiting or asymptotic behavior; there isn't much that can be said about the mechanism's behavior "in the interim" (e.g., on the way toward equilibrium).

In this context, the "simplifying strategies" described in Chapter 2—looking for rapid equilibria and steady-state candidates—are especially useful. These are techniques that take into account both mechanism structure and approximate (though not exact) numerical reasoning. For instance, consider the mechanism [13.1]-[13.3] from Chapter 2, reproduced below:

[12]In many cases, we may not know the rate constants of all elementary steps, which renders "purely structural" deduction especially useful.

```
[25.1]  A --> B      k1 = 100

[25.2]  B --> A      k2 = 200

[25.3]  B --> C      k3 = 0.1
```

The only numerical information that we really need in order to invoke the notion of "rapid equilibrium" is that both $k1$ and $k2$ are much greater than $k3$. The specific values of these constants are of course necessary if we want quantitative information, such as the near-constant ratio of B to A throughout the course of reaction; but the fact that this ratio will indeed be approximately constant does not depend on the particular numbers but only on their relation.[13] Thus, capturing the idea of "rapid equilibrium" (and "steady-state candidates" also) depends on formalizing the notion that one quantity is much bigger than another.

5.2.1 Fast Equilibria and Fast Submechanisms

The Workbench includes two heuristic techniques for spotting fast equilibria—a simple technique that finds "obvious" examples (like [25.1]-[25.3] above), and a more sophisticated strategy that looks for "fast submechanisms" larger than simple reaction pairs. The first technique looks for patterns that meet all the following conditions:

a. A reaction/reverse reaction pair

b. The species involved in this pair appear nowhere else in the mechanism (that is, the two complexes in this reaction pair contain all instances of their constituent species)

c. These two complexes only appear as reactants in reactions outside the reaction pair

d. The rate constants of the reaction pair are much greater than the rate constants of the other reactions involving these

[13] Generally, when examples like [25.1]-[25.3] appear in kinetics textbooks, the author will not even bother to include specific rate constants but will merely represent the relations between rate constants algebraically.

complexes.

Using mechanism [25] above can help to clarify these conditions. We have a reaction pair `A <--> B`, meeting condition *a*; the only complexes that contain species `A` and `B` are in this pair, meeting condition *b*; these complexes appear only as reactants in all other reactions, meeting condition *c*; and the rate constants of the reaction pair are much greater than the rate constants (in this case, the lone rate constant) of all other reactions involving these complexes, meeting condition *d*. (We will specify what is meant by "much greater than" later in this section, but for now the concept should be reasonably clear—we are merely comparing the relative magnitudes of numerical values.)

Conditions *b* and *c* assure that we are dealing with a particularly simple situation: we know from *b* that there are no reactions "elsewhere" in the mechanism that might be affecting the relative concentrations of the equilibrium species, and from *c* we know that there are no reactions *producing* the equilbrium species that might invalidate our approximation of fast equilibrium. As examples of "invalid" mechanisms, consider [26] and [27] below:

```
                 k1     k3              k4
[26]       A <---> B ---> C      D + B --> 2C
                 k2

                 k1
[27]       A <---> B <--- C
                 k2     k3
```

In mechanism [26], a very large concentration of `D` might invalidate the fast equilibrium approximation, even if $k4$ is much less than $k1$ and $k2$; this mechanism violates condition *b* above. Similarly, a large concentration of `C` can cause problems in mechanism [27] (which violates condition *c*). Thus, all four of the indicated conditions must be met to invoke an "obvious" judgment of a fast equilibrium situation.

There are other, less glaring cases in which the fast equilibrium approximation can be used. Consider the following mechanism:

[28] A + B <---> C <---> D + E

B ---> E

Suppose that the four reactions represented in the upper component of [28] have much larger rate constants than the reaction B --> E. This suggests that these four reactions might constitute a "rapid submechanism" in which the ratios [D][E]/[C] and [C]/[A][B] are nearly constant over time. The situation is not as clearcut as [25], since the concentrations of A and D may be extremely small, so that the actual rate of the reaction A + B --> C might in fact be comparable to that of B --> E. However, under "ordinary" assumptions of the concentrations of A and D—that is, assuming that their concentrations are comparable to those of the other species—the rapid submechanism approximation is a reasonable one.[14] In this case, we can examine the four fast reactions as though they constituted a complete mechanism:

[28'] A + B <---> C <---> D + E

This is a (strongly) reversible, zero deficiency mechanism; thus, we know that the system has a unique locally stable equilibrium state (for any particular set of nonzero initial concentrations). As mechanism [28] runs, then, it will generally look like a "perturbed" version of [28'] in which the available stoichiometric subspace varies slowly over time.

Note that we have now extended the "fast equilibrium" approximation to a broader and more general "fast tractable submechanism" approximation. In doing so, we are making use not only of numerical comparisons, but also of the previous section's graphical heuristics (in particular, the Zero Deficiency Theorem). An especially powerful illustration of this type of reasoning is provided by mechanism [29] below:

[29] 0 <---> A <---> 2B

A + C <---> 2D

[14]Note that by the Zero Deficiency Theorem, some species in this mechanism will eventually approach zero concentration. Thus, in the *limiting* situation, the assumption of rapid equilibrium fails; but it may be a good approximation for a long time period as the mechanism approaches this limiting state.

Again, suppose that the reactions in the upper graph component are very fast relative to those in the lower component (note that we have used the "zero complex" notation to indicate an external source and sink for species A). The "fast submechanism" here is one in which the dimension of the stoichiometric subspace is the same as that of the species space (namely, 2); thus, to a good approximation, both A and B have constant concentrations (which can be determined from the rate constants of the upper component). This in turn implies that the overall reaction may be simplified to:

```
[29']       C <--> 2D
```

where the effective rate constant of the step C --> 2D now has the (near-constant) concentration of A incorporated into it. The conceptual path from [29] to [29'] thus reflects a combination of numerical approximation, purely numerical work (to determine the constant concentrations of A and B) and graphical analysis; in tandem, these techniques are able to achieve a major simplification of the overall mechanism.

The Workbench is able to find certain plausible instances of "fast submechanisms" of the kind mentioned here, and notes when those submechanisms allow still further simplification (such as treating some species as constant).[15] As an example, we can look at the Workbench's treatment of mechanism [29]; here, we set the rate constants so that the four "fast reactions" are indeed much faster than the two involving species C and D:

```
(define *mechanism-29*
  '(
    (
     (  ((a 1)) ((b 2)) 2.)
     (  ((b 2)) ((a 1)) 1.)
     (  ((a 1) (c 1)) ((d 2)) 0.00001)
     (  ((d 2)) ((a 1) (c 1)) 0.000005)
     )
    ()
    ((a 1. 1.)) ((a 2.)) () ()
    ))
```

[15]The algorithm used by the Workbench is given in Appendix B.

Now, we can create a mechanism object from this data structure, and we request a graphical analysis of that object:

```
(define *mechanism-29-object*
  (make-mechanism-object *mechanism-29*))

(analyze-graphically *mechanism-29-object*)
```

The Workbench notes that this is a (strongly) reversible zero deficiency mechanism, and thus has a unique equilibrium state; it then goes on to flag the "fast submechanism" within this mechanism. (In the printout shown below, the program depicts the steps of the fast submechanism, incorporating the "zero complex" notation described earlier.)

```
Rule number 5.3 -- Fast Submechanisms is firing

We look for fast zero-deficiency submechanisms.
Fast Submechanisms Found:
(((((b 2)) ((a 1)) 1.)
 (((*zero*)) ((a 1)) 1.)
 (((a 1)) ((b 2)) 2.)
 (((a 1)) ((*zero*)) 2.)))

Rule number 5.4 -- Approximate Constants  is firing

We now look for approximate constants to add from the fast submechanisms.
The following species have approximately constant values: (a b)


Rule number 5.4 Part 2-- Approximate Constants is firing

Would you like the value of the constants?(y/n) y
We now find the concentrations from the fast mechanisms.

Here are the concentrations of the near-constant species:
  (((a .5000536403990566) (b 1.000105590506487)))
```

In summary, the Workbench has first noted that the four reactions involving A and B only constitute a fast submechanism. The program then notes that this tractable submechanism is reversible with zero deficiency and in addition that the equilibrium concentrations of A and B do not depend

on their initial concentrations; hence they may be treated as "approximately constant" within the mechanism. Finally, the program uses fast numerical methods (as described in Section 3 of this chapter) to estimate those near-constant concentrations of A and B. In the course of this analysis, the Workbench annotates the original mechanism object, setting slots such as approximate-constants-found; these results are used in making predictions about the behavior of the mechanism, as described in Section 4 of this chapter.

In this particular case, the Workbench's annotation is also used to do more than predict behavior: the program can exploit the identification of "approximately constant" species to suggest a simplification to the mechanism when it is eventually simulated numerically. If the user were now to simulate mechanism [29] along the lines shown in Chapter 4, the Workbench would notify the user that an "approximate version" of the mechanism is available. Just to see how the program actually achieves this, we can pursue our current example and evaluate the following expression (after creating run- and graph-parameters lists as shown in the previous chapter):

```
(do-simple-graphed-run
  *mechanism-29-object*
  *mechanism-29-run-parameters*
  *mechanism-29-graph-parameters*)
```

The Workbench now replies with:

```
This mechanism has approximate versions. Do you want to see them? (y/n):
```

And if we respond in the affirmative, the program prints out the following (excerpts are shown):

```
*** Approximate Mechanism: ***

a + c ---> 2d          Rate constant: .00001
2d ---> a + c          Rate constant: .000005

Constant species (and concentrations):
Species: a     Concentration: .5000536403990566
Species: b     Concentration: 1.000105590506487
```

```
These are retrieved by sending this mechanism object an
APPROXIMATE-CONSTANT-MECHANISMS message.
You may wish to change some run parameters:
(starting-dt .05)
(steps-per-display 100)
(focus-species (a b c d))
```

The program has printed out for us, not only the approximate version of the mechanism that it wishes to suggest, but run parameters that might profitably be alterable if we choose to simulate the simplified mechanism. We could now pursue this example further by retrieving the simpler mechanism and numerically simulating that:

```
(define *mechanism-29-prime*
  (car (*mechanism-29-object* 'approximate-constant-mechanisms)))

(do-simple-graphed-run
  (make-mechanism-object *mechanism-29-prime*)
  *mechanism-29-prime-run-parameters*
  *mechanism-29-prime-graph-parameters*)
```

In the first of these expressions we retrieve the simpler mechanism from the original mechanism object. (The message passed to our mechanism object actually returns a list of possible simpler mechanisms; in this case there is only one, so we retrieve the first object in the list.) We then create a new mechanism object from this—which we could analyze graphically if we so desired—and simulate our new simpler mechanism with new run and graph parameters.

It is worth stepping back from this example a moment, just to see the progression of operations employed by the Workbench. Originally, we presented the program with mechanism [29]; using a variety of methods, some of them symbolic, some numerical, the program discovered that the [29] had a fast "submechanism'; that this submechanism would inevitably reach a global equilibrium state, independent of initial concentrations; and the values of those concentrations. It was then able to use its earlier deductions to suggest a simpler mechanism to use for the numerical simulator—an instance of the communication between graphical analysis and numerical simulation mentioned in Chapter 3.

5.2.2 Steady State Candidates

In Chapter 2, the notion of "steady-state approximation" was introduced. The basic idea in this approximation is that a particular species—often a reactive intermediate, like a free radical—will be produced by slow reactions and consumed by faster ones; thus it is likely to be at a very low (and very slowly changing) concentration throughout much of the running time of the mechanism. Often, this assumption can render an algebraic analysis of the mechanism feasible (as with mechanism [4.1]-[4.4] in Chapter 2); we might also use the assumption of constant concentration to simplify the overall mechanism, much as we did with mechanism [29] above.

The Workbench contains two techniques for finding steady-state candidates: one that looks for "obvious" candidates (and makes no assumptions about the concentrations of other species), and one that looks for "possible" candidates, based on plausible assumptions about the concentrations of other species in the mechanism.

The first, "obvious" technique looks for patterns roughly of the form:

```
                k1      k2
[30]        0 ---> X ---> [products]
```

in which the value of $k2$ is much greater than that of $k1$. (Again, we are using the "zero complex" notation to indicate the presence of an external source or constant-concentration reactant.) In such a case, the species `X` is deemed a good candidate for the steady-state approximation. Specifically, such a pattern has to meet the following conditions:

a. The species `X` must be produced only by external sources or constant-concentration species.

b. In any reaction with `X` as reactant, `X` must be the only species appearing.

c. At least one of the rate constants for a reaction with `X` as reactant must be much greater than all the rate constants for those steps which have `X` as a product.

In addition, if X only appears as a reactant in unimolecular steps the Workbench will return its estimate for the near-constant concentration of X.

The conditions listed above are rather restrictive, and designed to ensure that the steady-state approximation is as defensible as possible. Condition *a* ensures that the rate of production of the candidate species is a known constant, and does not depend on the concentrations of other species. Condition *b* likewise ensures that we needn't consider wide fluctuations in the concentrations of other species to determine the rate of disappearance of X. Finally, condition *c* ensures that there is at least one "very fast" reaction in which X is consumed. As with the "obvious" technique for spotting fast equilibria, the only qualitative comparison here is between numerical values (namely, rate constants).

A more heuristic method for finding probable steady-state candidates uses an *a priori* assumption that other species are at "normal" concentrations, and searches for a species X such that at least one step with X as reactant is much faster than all the steps with X as product. There are some subtleties involved in making such a judgment; consider, for instance, the following mechanism:

```
                    k1                k2
[31]        A + B ---> X     X + C ---> D
```

Now, it may be in this case that $k2$ is much greater than $k1$; but in order for X to be a good steady state candidate, we need to know that the product `k2 * [C]` is much greater (under "typical" conditions) than the product `k1 * [A] * [B]`. If A and B have large concentrations (much larger than C), then this assumption is false and X will not obey the steady-state approximation. The Workbench's heuristic uses conservative assumptions in a case like this, and insists that the relation between $k2$ and $k1$ be even more extreme (in a way that will be explained later) than in an "obvious" case like [30] above. That is, for every additional reactant molecule (like A or B above) that produces X in an elementary step, the conditions under which we decide that one reaction is much faster than another become more restrictive. This is of course not a foolproof technique—it will admit cases in which X really is not a good candidate, and it will exclude some cases in

which the steady-state approximation could have been made—but it does reflect plausible judgments about unknown aspects of the mechanism.

As an example of the "steady-state approximation" in action, we can once more consider the Workbench's performance on the N2O5 decomposition mechanism [2.1]-[2.4].[16] The original data structure is reproduced below:

```
(define n2o5-decomposition
  '(
    (
     ( ((n2o5 1)) ((no2 1) (no3 1)) 0.002)
     ( ((no2 1)(no3 1)) ((n2o5 1)) 0.001)
     ( ((no2 1)(no3 1)) ((no2 1)(no 1) (o2 1)) 30.)
     ( ((no 1) (n2o5 1)) ((no2 3)) 4000.)
     )
    () () () () ()))
```

The initial portion of the Workbench's analysis, in which the program discovered that certain species must approach a concentration of zero, was shown earlier. Continuing with the analysis, the Workbench prints out the following:

```
Rule number 5.5 -- Steady State Candidates is firing

We now look for obvious steady state candidates in the mechanism.
We have found some steady-state candidates.
Possible sets of steady-state species: ((n2o5) (no3) (no no3))
```

The Workbench here indicates that, depending on initial conditions, we might be able to use a steady-state approximation for N2O5 alone, for NO3 alone, or for both NO and NO3. The reader may recall that in the analysis in Chapter 2, we used the approximation for both NO and NO3 to analyze the behavior of this mechanism; this is appropriate, since the "usual" assumption is that our initial conditions will involve a significant amount of N2O5 and very little of either NO or NO3 (we are, after all, interested in the process

[16]Recall that the steady-state approximation was originally introduced in Chapter 2, in which this mechanism was used as an example.

of "N2O5 decomposition"). Under these conditions, the steady-state approximation for both NO and NO3 proves useful. Nevertheless, it is interesting to note that the Workbench has also pointed out the possibility of employing a steady-state approximation to N2O5 alone; conceivably, if our initial conditions were to involve a large amount of NO2 and NO3 (and very little N2O5), this would be an appropriate approximation to make. The Workbench has no notion of "typical" starting conditions in suggesting this possibility; it bases its judgment on the rate constants and mechanism structure alone. As an additional point, the Workbench has indicated that the steady-state approximation for NO depends on making the prior approximation for NO3 (that is, that the approximation would not be evident unless we assume that the concentration for NO3 is small). This follows from studying the two reactions in which NO is involved:

```
NO2 + NO3  ---> NO2 + NO + O2     k3 = 30

NO + N2O5  ---> 3NO2              k4 = 4000
```

The rate at which the concentration of NO is increasing due to the first reaction is 30[NO2][NO3], and the rate at which it is declining due to the second reaction is 4000[NO][N2O5]. If we were to assume that NO2, N2O5, and NO3 were all at about the same concentration, then the rate of production of NO might easily be rather high compared to its rate of consumption, invalidating the steady-state approximation. Thus, in order to use this approximation for NO, we must first justify its use for NO3; this type of "dependence relationship" among approximations is rarely made explicit in chemistry textbooks.

5.2.3 A Note on Qualitative Arithmetic

The previous two sections discussed the ability of the Kineticist's Workbench to employ concepts such as "fast tractable submechanisms" and "steady-state (low-concentration) species" to analyze mechanisms. In order to express these concepts computationally, the Workbench must be able to represent the notion that one number is "much bigger" than another. Consider, for instance, the simple steady-state example shown in mechanism [30] above:

```
                k1      k2
[30]        0 ---> X ---> [products]
```

We can say that X is a good candidate for a steady-state approximation if $k2 \gg k1$; but what precisely does such a statement mean? The Workbench uses, as the heart of its numerical comparisons, a special global variable *qualitative-epsilon*, which has a default value of 0.01. This value is used to generate a sequence of other globals:

```
*greater-than-factor* =   1/*qualitative-epsilon*

*much-greater-than-factor* = *greater-than-factor*^2
```

and so forth.

These values are in turn used to make the comparisons necessary for the heuristics described in this section. For instance, in mechanism [30], the value of $k2$ must exceed that of $k1$ by a factor of 10000 (the default value of *much-greater-than-factor*) in order to justify a steady-state approximation for X.

In its current state, the Workbench does not use (or need) a complete axiomatization of qualitative arithmetic such as that described by Raiman {65} or Mavrovouniotis and Stephanopoulos {60}. The systems described in these papers focus upon rules that derive relations between expressions based solely on qualtitative information; for instance, one axiom in Raiman {65} reads:[17]

$$\text{If } x \ll y \text{ and } y \simeq z \text{ then } x \ll z$$

The Workbench does not have the capacity for qualitative deductions of this kind. However, it does sometimes need to compare expressions with both numeric and symbolic elements. Consider, for instance, the process of making a steady-state approximation for the N2O5 decomposition mechanism. In the course of its analysis, the program looks to see if NO3 may be a good candidate for this approximation; and to do this it must compare the relative rates of the following two steps:

[17]This is a "prose translation" from Raiman's formalism.

```
N2O5 ---> NO2 + NO3              k1 = 0.002

NO2 + NO3  ---> NO2 + NO + O2    k3 = 30.
```

This is tantamount to comparing the algebraic expressions

```
0.002 [N2O5]
```

and

```
30 [NO2]
```

Obviously, the program cannot make a firm comparison of these expressions, since the symbolic elements are unspecified; but in the absence of any further information, we make the conservative assumption that N2O5 exceeds NO2 in concentration by 100 (the value of *greater-than-factor*). Our effective comparison, then, is between

```
0.2 [NO2]
```

and

```
30 [NO2]
```

We now test whether the value of the second expression exceeds that of the first by 100 (again derived from *greater-than-factor*). In this case, the test is met, so we can justify a steady-state approximation based (in part) upon this comparison.[18]

There are a number of other, similar tests made by the Workbench to compare expressions,[19] but the basic idea is illustrated by these examples. The single global value of *qualitative-epsilon* "cascades" through all tests made by the Workbench; thus, the user can specify a more lenient qualitative comparison by resetting this value using the procedure

[18]The approximation cannot be made based *solely* upon the comparison shown: we must also demonstrate that there are no other steps generating NO3 that might be "fast" relative to the fastest step in which this species is consumed. In the current example, NO3 is a product of only one step, so the conclusion is immediate.

[19]Summaries of the algorithms are given in Appendix B.

`reset-qualitative-epsilon!` For instance, evaluating the following expression:

```
(reset-qualitative-epsilon! 0.1)
```

would imply that in the previous example, N2O5 would be assumed to have 10 times the concentration of NO2 (rather than 100), and under this assumption, the expressions would only have to differ by a factor of 10 (rather than 100). Under these assumptions, a much higher value of $k1$ (say, a value of 0.2) would still be compatible with a steady-state approximation for NO3.

5.3 Rapid Location of Equilibria

Earlier in this chapter, in the discussion of "special structural features of mechanisms," we noted that some mechanisms have the interesting property that they must reach the same equilibrium state regardless of initial concentrations. (More specifically, these are reversible zero-deficiency mechanisms whose state space has the same dimension as the available stoichiometric subspace.) We made use of this property in one example of the "fast equilibrium" approximation: in analyzing mechanism [29], the Workbench noted that a "fast submechanism" could be spotted, and in addition that the near-constant equilibrium concentrations of this mechanism could be determined without simulating the mechanism. Again, the ability to perform this simplification depended on the fact that the fast submechanism of [29] had a unique equilibrium state, independent of initial concentrations.

When it encounters a situation of this kind, the Workbench has two methods for rapid determination of equilibrium concentrations. If the given mechanism has a unique (initial-state-independent) equilibrium, and if that mechanism gives rise to linear differential equations, then this is an especially tractable case: we are then able to find the equilibrium concentrations by solving an easily-constructed linear system. As an example, we can return to mechanism [21], reproduced below:

```
                 k1      k3
[32]        0 <---> A <---> B
                 k2      k4
```

Here, we have augmented the original graph of the mechanism with symbolic rate constants ($k1$ and $k3$ represent the "forward" reactions, $k2$ and $k4$ the "backward" reactions). The differential equations governing this system are:

$$[33.1] \quad \frac{d[A]}{dt} = k1 - (k2 + k3)\,[A] + k4\,[B]$$

$$[33.2] \quad \frac{d[B]}{dt} = k3\,[A] - k4\,[B]$$

Thus, to find the equilibrium concentrations for this system we need to solve a linear system of the form

$$[34] \quad M \begin{bmatrix} [A] \\ [B] \end{bmatrix} = \begin{bmatrix} -k1 \\ 0 \end{bmatrix}$$

where

$$M = \begin{bmatrix} k2 + k3 & k4 \\ k3 & -k4 \end{bmatrix}$$

Since we have already determined that this mechanism has a unique positive equilibrium state, we likewise know that these equations have a unique solution. Given numeric values for the rate constants, it is thus easy to find the solution to [34] and to determine the equilibrium concentrations of [32] without bothering to simulate it.

As an example of the Workbench's performance on such a mechanism, we can first create a mechanism based on [32]:

```
(define *chap5-mech32*
  '(
    (
     ( ((c 1)) ((a 1)) 2)
     ( ((a 1)) ((c 1)) 1)
     ( ((a 1)) ((b 1)) 2)
     ( ((b 1)) ((a 1)) 3))
    ((c 2)) () () () ()))
```

We now create a mechanism object from this and use the `analyze-graphically` procedure:

```
(analyze-graphically *chap5-mech32-obj*)
```

In the course of its analysis, the Workbench prints the following:

```
Rule number 2.2 -- Deficiency Theorem with Unique Equilibrium is firing

If rule 2.1 applies, and the mechanism must reach
positive equilibrium, *and* if the stoichiometric
subspace has the same dimensionality as the species
state space, then the equilibrium state is independent
of initial concentrations.

Rule number 2.2 -- Finding Unique Equilibrium Concentrations is firing

Would you like to compute equilibrium concentrations?
```

If we answer in the affirmative by typing `y`, the Workbench prints out:

```
We now look for equilibrium concentrations.
This is a linear mechanism.
The equilibrium concentrations: ((a 4.) (b 2.6666666666666665))
```

Here, the Workbench has noted that the mechanism in fact gives rise to linear differential equations, and it uses a simple Gaussian elimination technique to solve for the equilibrium concentrations.

A more difficult case arises if the system is not linear but involves quadratic (or on rare occasions cubic) terms. In such a case, the Workbench does not use algebraic methods but rather employs a kind of "fast equilibrium search"

algorithm to locate the equilibrium concentrations numerically. This algorithm, described in more detail in Appendix B, uses two separate "phases" of search. Initially, it uses a "star-walk" procedure that seeks to minimize the magnitude of the derivative vector obtained from the differential equations. That is, we seek to minimize the magnitude of the vector

$$(d[S1]/dt, d[S2]/dt....d[Sn]/dt)$$

where $S1$, $S2$... Sn are the n species that determine the state of the system. Clearly, this value is always non-negative, and it reaches its unique minimum value of 0 precisely at that point at which the system is in equilibrium.

The star-walk procedure looks at each separate reaction in the mechanism, and finds which of these reactions will most diminish the magnitude of the derivative vector. It then "preferentially" runs this particular reaction until another is found to be more effective. To take a concrete example, let us use the "fast submechanism" that we saw before in mechanism [29]:

```
[35]          0  <---> A <---> 2B
```

The star-walk procedure looks to see which of the four reactions in this mechanism is most effective (at the current state of the system) in reducing the magnitude of the system derivative; for the sake of argument, suppose this preferred reaction is

```
A --> 2B
```

The algorithm would now call for a certain amount of A to be "transferred over" to twice that amount of B; in other words, we run this one reaction alone, temporarily stopping all others. The advantage of this method is that we can transfer a large amount of A all at once, moving to a new point in state space at which the system derivative is a good deal smaller.[20] We continue this process, adapting the "transfer amount" to take advantage of the possibility of taking larger and larger steps in state space.

Typically, the star-walk procedure will move fairly rapidly at first, transferring large amounts along different reaction vectors; but then it slows down,

[20]Naturally, some care must be taken to preserve positive concentrations.

eventually calling for only tiny transfer amounts even when the system is not yet at the equilibrium point. At this juncture, the algorithm switches to its second phase, in which it uses a standard fixed-step Runge-Kutta integration routine to integrate the system until it appears that the system is approaching some final equilibrium point in a linear fashion; that is, we look for a series of steps in state space of the form

$$dV, xdV, x^2dV, x^3dV...$$

where dV is the small step in state space dictated by a particular iteration of the Runge Kutta integrator, and x is a positive scalar such that $|x| < 1$. When the program spots a sequence of steps of this kind, it immediately jumps to a new point:

$$P_0 + (1/(1-x))dV$$

where P_0 is the spot in state space where the sequence of linear jumps was first noted. Thus, what the program is doing is looking for a chance to estimate an equilibrium state by noting when the system seems to be approaching such a state along a slow eigenvector.

Having taken this large "linear jump," the system now iterates the entire procedure, alternating star-walk and Runge-Kutta phases, until it deems that it is sufficiently close to an equilibrium point. In practice—admittedly, with fairly tractable examples such as mechanism [35]—the procedure has not required more than two complete star-walk/Runge-Kutta cycles to find the desired equilibrium state to a good approximation (better than one percent accuracy).[21] An example of the Workbench's performance in this regard is shown in its analysis of mechanism [29] earlier in this chapter; finding the fast-equilibrium concentrations required approximately 120 star-walk steps and 30 Runge-Kutta steps in total, starting from initial (default) concentrations of 1 for both `A` and `B`. (Note that since the initial concentrations do not affect the ultimate equilibrium state, we can choose any starting state as long as all concentrations are nonzero.)

The star-walk/Runge-Kutta algorithm outlined here is useful even in

[21]It should be noted that this algorithm is inadequate for stiff systems, because of the poor performance of Runge Kutta integration on these systems. In principle, the same basic algorithm could be adapted to use a Gear integrator, but the Workbench does not have this capability at present.

those cases where the eventual equilibrium state does depend on initial concentrations. In this case, we can start the algorithm at the desired initial state; the equilibrium point reported by the algorithm will be consistent with that initial state. The reason that the algorithm performs well in this case is that each step of both the star-walk and Runge-Kutta phase keeps the overall system state within the available stoichiometric subspace; in other words, the algorithm will not "move" the system into a state that is inaccessible from its initial state. Thus, we may elect to use the Workbench to find an equilibrium point for a given reversible zero-deficiency mechanism starting from some initial state.

As an example of the Workbench's performance in this capacity, we can consider mechanism [12], shown earlier in this chapter and reproduced below:

```
[36]        A + B <--> C <--> D
```

We create a mechanism data structure for [36], choosing a set of rate constants:

```
(define *chap5-mech36*
  '(
    (
     ( ((a 1)(b 1)) ((c 1)) 2)
     ( ((c 1)) ((a 1)(b 1)) 1)
     ( ((c 1)) ((d 1)) 2)
     ( ((d 1)) ((c 1)) 3))
    () () () () ()))
```

We now create a mechanism object from this data structure and use the Workbench's `find-rapid-equilibrium` procedure, which takes as arguments a mechanism object and a set of initial concentrations:

```
(find-rapid-equilibrium
  *c5m36obj* '((a 1.) (b 1.) (c 0.) (d 0.)))
```

After 34 star-walk steps and 26 Runge-Kutta steps,[22] the Workbench prints out the result of its equilibrium search:

```
Final state:
```

[22]The calculation took less than two minutes in interpreted Scheme on an HP 350 workstation.

```
((a .41786353744276783) (b .41786353744276783)
 (c .34927453982662937) (d .23286192273060285))
```

Checking this solution against an algebraic solution for the original differential equations shows that it is well within one percent of the correct value. Note also that this particular mechanism has four species that constitute its state variables, but that for any given starting state it must obey two additional constraints that limit the stoichiometric subspace to two dimensions:

$$[A] - [A] = [B] - [B]$$
$$0 \qquad\qquad 0$$

$$[B] + [C] + [D] = [B] + [C] + [D]$$
$$0 \qquad 0 \qquad 0$$

Examination of the Workbench's approximation of the equilibrium state reveals that both these constraints have likewise been met by the program's solution.

The previous example shows that the Workbench is often capable of locating an equilibrium state relatively quickly—as long as it is provided with a reversible zero-deficiency mechanism and known starting conditions. The program does not, however, perform this calculation during the "pre-simulation" phase, since it is not supplied with starting conditions (the mechanism data structure, and the mechanism object created from it, make no default assumptions about initial concentrations). Thus, although the user may employ the Workbench to find a particular equilibrium state for a mechanism such as [36], the program will only try to do this calculation *automatically* for especially tractable mechanisms like [35], in which the equilibrium state is independent of initial concentrations.

The Workbench's ability to locate a global equilibrium point "on its own" for mechanisms like [32] can be employed to guide its numerical methods: if desired, a simulation can be automatically terminated when the (already-known) equilibrium state is approached within a pre-defined tolerance. To take advantage of this option, the user can include an `end-conditions` parameter in the run-parameters list, and give this entry a value of `(end-time use-known-equilibrium)`. With this specification, the numerical simulation will run until either the given end-time is reached, or until the system

nears its equilibrium point, whichever comes first. This "linking" of graphical analysis with numerical simulation is an instance of the cooperation between symbolic and numerical methods mentioned earlier in this paper; we saw another instance of this symbiosis in the fast-submechanism example of Section 5.2.1.

5.4 Notating Mechanism Objects with Predictions

Often the reason for performing graphical analyses of mechanisms is to predict their likely or necessary behavior. The Kineticist's Workbench, as it analyzes a mechanism, annotates the appropriate mechanism object with a symbolic representation of whatever behavioral predictions it can make. These symbolic predictions may later be checked against actual simulation results (as will be shown in the next chapter).

Most of the Workbench's predictions concern the asymptotic or long-term behavior of mechanisms. For example, if the Workbench is able to deduce that the mechanism must reach a unique equilibrium state independent of initial concentrations (as was the case with example [35]), it annotates the `prediction-list` slot of the mechanism object with a list containing the symbol `global-equilibrium` and the predicted concentrations. Naturally, such a prediction is not foolproof—we may later choose to simulate the mechanism for only a brief time, and the system may not achieve the predicted concentrations. Nevertheless, the prediction is useful in that it tells us what the program "expects" to see in any given simulation; and, occasionally, the fact that a prediction is *not* met may also be of interest to the chemist.

A summary of the types of predictions made by the Workbench is shown below. All of these prediction-types are represented by symbolic entries in the `prediction-list` slot of a mechanism object. In the following chapters, we will see how the program can later use the numerical results of simulation to check these predictions for accuracy.

- Global Equilibrium (all nonzero concentrations)
  This prediction indicates that the system should reach a unique equilibrium state, independent of initial concentrations (as in example [35]). The predicted concentrations are also included

in this entry.

- Necessary Equilibrium (all nonzero concentrations)
This indicates that the mechanism is a reversible zero-deficiency mechanism, and thus can be expected to reach an equilibrium state in which all species have nonzero concentration. (Mechanism [36] is an example of this type of system.)

- Possible Equilibrium (some species at zero concentrations)
This indicates that the mechanism may approach a steady state in which some species are at nonzero concentration, and some reactions are proceeding at a nonzero rate; but other species are expected to approach a concentration of zero. (Mechanism [7] is an example of this type.)

- Possible "Non-Running" Mechanism
This indicates that the mechanism may approach a state in which none of the elementary reactions has a nonzero rate—i.e., the mechanism as written can "run down." (The nitrogen pentoxide decomposition reaction [2.1]-[2.4] is an example of this type.)

- Possible Steady-State Species
This indicates that the Workbench has spotted some species that are good candidates for a steady-state approximation.

- Possible Fast Equilibrium Pairs of Reactions
This indicates that the Workbench has spotted some forward/backward reaction pairs (such as [25.1] and [25.2] in mechanism [25]) that are good candidates for a "fast equilibrium" approximation.

5.5 Mechanisms not Handled by Graphical Analysis

There are two major classes of mechanisms that may be represented in the Workbench, but that the program is incapable of analyzing. Currently, the Workbench cannot analyze *parameterized* mechanisms—i.e., mechanisms in which rate constants are provided symbolically. (These mechanisms are used as the bases for parameter-space searches, as described in the next chapter.)

The superficial reason for this limitation is that the use of a symbolic rate constant renders some of the program's techniques inapplicable; for example, we can no longer perform a fast numerical calculation to find an equilibrium state. However, many of the graphical techniques shown in this chapter (especially in Section 1) could in principle still be applied to parameterized mechanisms. Thus, this limitation is at least partially remediable; and future versions of the Workbench should be able to perform at least limited analysis of parameterized mechanisms.

Mechanisms with explicit "function-of-time" species are also not handled by the Workbench. (These are mechanisms in which a particular species is specified via a numerical procedure that takes a "time" argument and returns a concentration value; an example named `brusselator-with-sinusoidal-a` was shown in the previous chapter.) Here, the limitation of the program is unavoidable; in general, by specifying that a concentration will be given by an arbitrary function of time, we render the mechanism intractable to analysis. Among other complications, we could stipulate that the given "function-of-time species" has a concentration that is discontinuous, or varies over tremendously wide ranges, or depends on user input. Thus, there is little that the program could possibly say about the behavior of these mechanisms without a great deal of specific (and necessarily ad hoc) information.

In practice, neither of these limitations is particularly severe. A chemist who wishes to analyze a parameterized mechanism might, for instance, create a version of the mechanism with "representative" parameter values, and use the `analyze-graphically` procedure on this specific example; most of the results printed out by the program would be applicable to the original parameterized version of the mechanism as well.[23] As for mechanisms

[23]The one remaining limitation in this example is that the program cannot use the predictions generated by graphical analysis to do any tests within parameter-space checks—that is, the symbolic predictions have been added to the chemist's "representative" mechanism, not to the parameterized mechanism that will later be simulated. For example, the chemist cannot simulate the parameterized mechanism at a series of parameter values and ask the program to graph at which parameter values a given prediction, such as the approach to a unique equilibrium state, is met within a certain amount of time. Thus, the example in the text demonstrates how the results of graphical analysis may be used by the *chemist*; but in using the program this way, the symbolic predictions are not accessible to the *program*.

with function-of-time species, these are way too under-specified for the user to expect much automatic analysis. In pragmatic terms, these mechanisms are included in the program to provide the chemist with a richer collection of mechanisms to study—and indeed, most of the techniques that generate "qualitative summaries," as described in the next chapter, remain applicable to these systems.

Chapter 6

Analyzing the Numbers

The previous two chapters have described how the Workbench performs numerical simulation of mechanisms, and how it can precede that simulation by analyzing mechanism structure with an eye toward prediction and simplification. In this chapter, the final phase of the Workbench's operation—analyzing and summarizing numerical results—will be discussed. Before proceeding with a detailed discussion of the program's operation, however, it is worth stepping back to provide some context: what exactly is the purpose of this part of the program? Why would we want a computer program to describe the numerical results produced by its own simulation?

The answer is that this is a significant part of the chemist's usual activity. After a chemist simulates a newly-created mechanism, he must then look at the numerical results, devoting his attention all the while to potentially interesting features of those results: rapid jumps in concentration, or concentrations that remain at a nearly constant value for much of the simulation, or stable oscillations, or a myriad of other features. (Recall, for instance, the quote from Noyes {62} in Chapter 2, in which a "narrative structure" is imposed on the numerical results obtained from simulating a particular oscillating mechanism.) Much of this interpetive work is not especially creative: spotting whether oscillations have occurred, for example, is a fairly routine job that might profitably be taken over by the computer that performed the simulation.

It is this "cliched" aspect of numerical interpretation and summary that the Workbench attempts to capture. The program is able to construct a symbolic narrative history of the numerical results, employing qualitative terms like "steady state" and "rapid increase in concentration"; it can look for typical patterns within a numerical record, such as stable oscillations or final equilibrium states; and it can generate reports that summarize its findings for the user. Additionally, the Workbench can use the results of

its qualitative analysis to guide further numerical work; and in those cases where it was able to predict mechanism behavior, it can use its interpretation of the numerical results to check the accuracy of those predictions.

The first two sections of this chapter describe two important data structures used by the Workbench in constructing its qualitative analyses. The first of these structures, the *episode*, can be viewed as a symbolic "chunk" representing a portion of the numerical record; the Workbench constructs a narrative history of the results by enumerating a sequence of distinct, consecutive episodes. The second data structure is a record of local maxima and minima of species concentrations; this proves helpful in identifying patterns such as oscillations in complex mechanisms.

The third section of this chapter discusses how the Workbench uses both these data structures to construct summary reports of numerical results; these reports may then be used to generate parameter-space graphs, such as Figure 3 in Chapter 3. The process of creating parameter-space graphs is outlined in Section 4.

Section 5 describes how the Workbench checks the predictions of mechanism behavior that were generated in the pre-simulation phase; and in Section 6 we discuss some additional miscellaneous capabilities of the Workbench that allow for storing and additional analysis of qualitative results.

6.1 Constructing Episode Histories of Simulations

In this section, we describe one of the two main data structures used by the Workbench in constructing its qualitative summaries of numerical results. We begin by defining this structure, the *episode*, and show how a sequence of episodes are used as a narrative history of the simulation.

6.1.1 The Episode Data Structure

Early in Chapter 2, the relationship between a mechanism and its associated mathematical model was described. At that time it was shown how a mechanism such as the Brusselator:

```
[1.1]     A --> X              k1 = 1
[1.2]     B + X --> Y + D      k2 = 1
[1.3]     2X + Y --> 3X        k3 = 1
[1.4]     X --> E              k4 = 1

  ([A] = 1, [B] = 3, [D] = [E] = 0)
```

dictated the construction of an associated set of ordinary differential equations:

$$[2.1] \quad \frac{d[X]}{dt} = k1\,[A] - k2\,[B]\,[X] + k3\,[X]^2\,[Y] - k4\,[X]$$

$$[2.2] \quad \frac{d[Y]}{dt} = k2\,[B]\,[X] - k3\,[X]^2\,[Y]$$

Now, when comparing differential equations such as [2.1]-[2.2] to the mechanism [1.1]-[1.4] from which they were generated, it is not hard to see how each individual terms in the equations is derived. The term $-k2$[B][X] in equation [2.1], for instance, represents the consumption of X from step [1.2]; likewise, the term $k3$ $[X]^2$ [Y] represents the net production of X from step [1.3]. Thus, at any given moment of time, the derivative of [X] is a sum of terms, each of which represents a particular elementary step in which X participates as reactant or product.

Suppose, then, we study mechanism [1.1]-[1.4] by performing the usual numerical integration, but that as we perform the integration we keep track of the magnitudes of individual terms in the differential equations. These magnitudes, as we have just seen, will indicate the relative participation of each elementary reaction in dictating the local derivatives of [X] and [Y]. Thus, by noting (for instance) which term in [2.1] has the largest magnitude at any given moment, we can determine which elementary reaction is "dominant" for species X.

This notion is the basis of the *episode* data structure used by the Workbench. As it performs numerical integration, the Workbench keeps track of the ordering of each individual term in the differential equations for the

system. Each time this ordering changes, the Workbench notes a change of episode. Thus, an episode may be defined as a period of time in a simulation during which the ordering (by absolute magnitude) of the terms in the governing differential equations remains constant.

This definition allows for some refinement in practice. It implies, for instance, that the *complete* ordering of terms is used to distinguish episode boundaries: to take a concrete example, the largest two terms in [2.1] may remain unchanged, but if the third- and fourth-largest terms exchange places, an episode change will be said to have occurred. This represents a "fine-grained" definition of episode, in which *any* change in the ordering of terms is deemed sufficient to warrant a change of episode. More often, a more "coarse-grained" definition is used in which only the most important term is used to distinguish episode boundaries; thus, in our example, the only time a new episode is begun is when the most important term in either [2.1] or [2.2] changes.

Further refinements are possible. For instance, in large mechanisms, a particular species may participate in many elementary reactions, and the chemist may only want to focus on the relative importance of several of these. The Workbench thus allows the chemist to select among the reactions that will be counted in determining episode boundaries. To use the Brusselator mechanism [1.1]-[1.4] as an (unrealistically simple) example, the chemist might decide that reaction [1.4] need not be used in determining episode boundaries; thus, only changes in the ordering of terms derived from [1.1]-[1.3] would be involved in determining a change of episode.

There are also hidden complications in the episode definition, particularly in the matter of determining "ties" between the contributions of terms. Often, it is reasonable to say that two terms are contributing "equally" to the local derivative of a particular species; consider, for instance, the simple mechanism shown below:

```
                k1
[3]          A <--> B
                k2
```

If we start off this system with a nonzero concentration of species A, and zero concentration of species B, we eventually arrive at a state in which the two

opposing reactions are in near equilibrium; thus the contribution of the two terms $k1$ [A] and $-k2$ [B] in determining the local derivative for [B] will be about equal (though the first will always be very slightly larger). In this case, we would like to say that the two reactions are approximately "tied" for episode-determining purposes, rather than to say that the forward reaction is "dominant" in any meaningful way.

The Workbench's solution to this problem is to provide a special parameter used in determining ties between terms. If two terms differ by less than this parameter (which, by default, is set at 0.2 percent), then the two terms are deemed to have approximately equal contributions for the purposes of determining episode boundaries.

The notion of "ties" between terms in certain episodes leads to other complications in constructing "episode histories" of simulations; we will return to this point in the following subsection.

6.1.2 Episode Histories

When a numerical simulation is performed, the user can request the Workbench to construct an "episode history"—a list of the successive episodes encountered—for this run. This is done by adding entries to the "run parameters" list introduced in Chapter 4. As a specific example, we could use the following as a run-parameters list for the Brusselator mechanism:

```
(define brusselator-run-parameters
  '((starting-dt 0.05)
    (start-time 0.)
    (end-time 40.)
    (actual-starting-concs
     ((a 1.) (b 3.) (d 0.) (e 0.) (x 0.) (y 0.)))
    (integration-method runge-kutta)
    (focus-species (x y))
    (steps-per-display 20)
    (end-conditions (end-time))
    (maintain-episode-history? ,true)
    (episode-change-depth 0)
    (graph-window ,*display-graph-window*)
    ))
```

The `maintain-episode-history?` parameter here tells the Workbench that we wish the numerical simulation to be accompanied by the construction of an episode history. The `episode-change-depth` parameter indicates that only the most important reaction (the "Oth-place" reaction) should be used in determining changes of episode: that is, an episode boundary will be flagged only if the most important term in the equations for `X` and `Y` changes.[1]

If the mechanism is now simulated numerically (as shown in Chapter 4) a list of episodes—the "episode history"—is automatically maintained and updated as the simulation runs. This history is then used as one of the major sources of information for later qualitative analysis. Each episode in the history is accompanied by the following information:

- The starting time for the episode
- The duration of the episode
- The list of "term-orderings" for each focus species that defines this episode
- Concentrations of focus species at the beginning of the episode
- Maximum and minimum concentrations of focus species during the episode
- Maximum and minimum species deltas during the episode

In addition to this information, the final episode in the history (i.e., the episode that concludes the simulation) also contains a number of special slots for the final concentrations, derivatives, and second derivatives of focus species. Collectively, this information can be used to do closer analysis of the qualitative history of the overall run, as shown in the following subsection.

[1]It should also be noted that the `focus-species` parameter is used to indicate those species whose "changes of dominant reaction" will determine an episode boundary. Thus, if we used only `x` as our focus species, an episode boundary would occur only if the dominant reaction for species `X` should change.

6.1.3 Initial Processing of the Episode History:
Removing Transition Episodes and Creating an Intrinsic Time-Scale

When the episode history is initially constructed, it often contains occasional "transition episodes" that represent a kind of midway state between longer episodes. For example, we might have a situation in which Brusselator step 3 is gradually becoming more important than step 2 for species Y. Initially, the term ordering for Y is simply (2 3); after the change, the ordering is (3 2). However, it may be that for one or two time-steps during the simulation, the two terms are nearly "tied" in value (this occurs when Y is at a local maximum of concentration).

Transition episodes of this kind typically make the episode history harder to analyze. We might, for example, be looking within the episode history for repeating episode patterns; and indeed, in the Brusselator example, we might find repeating patterns in which first step 2, and then step 3, becomes the more important step for species Y. But the occurence of intermittent transition episodes in the episode record could well make the job of finding such repeating patterns difficult. Instead of seeing the following pattern of episodes in the history:

Episode	Ordering of Y terms	Starting time
n	(2, 3)	12.0
n+1	(3, 2)	17.0
n+2	(2, 3)	19.0
n+3	(3, 2)	21.0

(etc.)

we see a more difficult-to-interpret pattern:

Episode	Ordering of Y terms	Starting time
n	(2, 3)	12.0
n+1	(2 and 3 tied)	16.95
n+2	(3, 2)	17.0
n+3	(2 and 3 tied)	19.0
n+4	(2, 3)	19.05
n+5	(3, 2)	21.0

Looking down the central column of the first table, the repetitions are obvious; but the presence of transition episodes in the second table illustrates the difficulties that they can introduce in automating the interpretation process.

It is for this reason that the first processing step that the Workbench performs in analyzing an episode history is to filter out of it those episodes that appear to be transition episodes—namely, very brief episodes that seem to arise from one term "passing" another one. The remaining episodes are used as the basis for future analysis.

The (filtered) episode history is also used as the basis for decisions about what constitutes a "long" or "short" time-span within the simulation. The Workbench creates a "time ruler" by using the shortest and longest episode lengths as extreme values; it subsequently defines notions of "long time," "short time," "very long time," and so forth by reference to this ruler. For instance, a "short episode" is defined as one whose duration D meets the following criterion:

$$\log D < (\log S + (0.25(\log L - \log S)))$$

where L and S are the longest and shortest measured episode-durations, respectively.[2]

The rationale behind this episode-history-based definition of terms like "long" and "short" is that it allows these terms to be rooted in the events of the simulation itself. Obviously there is no objective meaning to these terms: a ten-second period may be a "short time" within a simulation of a year-long

[2]The full set of qualitative time definitions is given in Appendix C.

process, or a "long time" in a twelve-second simulation. The Workbench's heuristic, then, is to let the simulation itself, and the pace of its episode changes, dictate what will be called "long" and "short" for the purposes of qualitative analysis. These terms likewise come into play in the definition of "rapid" versus "slow" changes, as will be shown shortly.

6.1.4 Attaching Features to Episodes

Once the episode history has been filtered of transition episodes, and a qualtitative time-ruler has been established, the Workbench produces a "feature history" associating qualitative features with the remaining episodes. The purpose of the feature history is to note, for each episode during the simulation, whether that episode included some interesting feature such as a rapid rise or decline in the concentration of a given species.

The full set of feature-descriptors that the Workbench uses is shown in Table 6.1 on the following page. These are accompanied by a brief description of their meaning within the program.

As Table 6.1 makes apparent, most of the episode feature descriptors (with the exception of SHORT, LONG, and FINAL) apply to the concentration path of some particular focus species. Thus, a possible feature descriptor list for some episode might read:

```
((SHORT) (LARGE-INCREASE X) (RAPID-INCREASE X)
 (STEADY-DECREASE Y))
```

This list—which would comprise a single entry in the feature history, corresponding to a particular episode in the episode history—indicates that during the given short episode, species X experienced a large, rapid increase, while species Y experienced a steady (moderate) decrease. We also know, parenthetically, that X must have decreased during at least one integration time-step in this episode (otherwise the Workbench would have included a "steady-increase" feature descriptor for `X` in its list).

The feature list is used as a descriptive annotation to the episode history, attaching qualitative terms to the time-chunks designated by individual episodes. For relatively short feature histories, it is perfectly reasonable for

Descriptor	Definition
SHORT/LONG	This episode is short (long) by the measure of the "intrinsic time-ruler"
STEADY-STATE *spec*	The species concentration varied little during this episode
LARGE-INCREASE/DECREASE *spec*	The species concentration increased (decreased) by a "large amount" during this episode
WIDE-SWING *spec*	The species concentration varied by a great deal during the episode (relative to its net overall change for the episode period)
RAPID-INCREASE/DECREASE *spec*	The pace of increasing (decreasing) concentration change for the overall episode is rapid for this species
SLOW-INCREASE/DECREASE *spec*	The pace of increasing (decreasing) concentration change for the overall episode is slow for this species
STEADY-INCREASE/DECREASE *spec*	The species never experienced a decrease (increase) for any time-step during this episode
FINAL	This is the final episode

Table 6.1: Feature Descriptors for Episodes

the user to examine the feature history alone and thereby get a sense of the major events of the simulation. Moreover, this list is used as a data structure in its own right by other portions of the program to be described later; for instance, the algorithm for checking whether a simulation met a prediction of obeying a "steady-state approximation" for some species includes a check of the feature history to see if that species was associated with a "steady-state" feature descriptor for a long period of time.

6.1.5 A Brief Example

It is worth pausing at this juncture to consider a brief example, just to illustrate the creation and use of an episode history (and its associated feature history). We consider, then, the "tiny mechanism" introduced in Chapter 4:

```
[4.1]    A --> B      k1 = 20
[4.2]    B --> C      k2 = 2
[4.3]    C --> B      k3 = 1
```

We create a mechanism object for this mechanism, as well as appropriate run- and graph-parameters lists. Although there is nothing new and surprising to say about these, it is worthwhile to show them in full just to provide a sense of how the entire example can be run in the Workbench:

```
(define *chap6-mech4*
  '(
    (
     ( ((a 1)) ((b 1)) 20)        ; step 4.1
     ( ((b 1)) ((c 1)) 2)         ; step 4.2
     ( ((c 1)) ((b 1)) 1))        ; step 4.3
    () () () () ()))              ; no constants, sources, etc.

(define *chap6-mech4-run-parameters*
  '((starting-dt 0.025)
    (start-time 0.)
    (end-time 5.)
    (actual-starting-concs
     ((a 5.) (b 0.) (c 0.)))
    (integration-method runge-kutta)
    (focus-species (a b c))      ; all species involved
                                 ; in episode construction
    (steps-per-display 40)
```

```
    (maintain-episode-history? ,true) ; construct the history
    (episode-change-depth 0)          ; based on most important term
    (graph-window ,*display-graph-window*) ; graph concentrations
    ))

(define *chap6-mech4-graph-parameters*
  '((graph-type numeric)
    (time-low 0.)
    (time-high 5.)
    (species-to-graph (c b a))
    (species-concentration-boundaries
     ((0. 6.) (0. 6.) (0. 6.)))  ; ordinate range for c, b, a
    (numeric-limits-for-graphs
     ((-1. -1.5 6. 7.) (-1. -1. 6. 7.) (-1. -1. 6. 7.)))
           ; the window regions below correspond to the
           ; virtual regions above
    (window-regions-to-use
     ((0 0 400 100) (0 101 400 200) (0 201 400 300)))
    (axis-colors-to-use
     (8 8 8))
    (default-colors (1 1 1))
    (color-procedure ,graph-object-use-default-color)))


(define *chap6-mech4-object*
  (make-mechanism-object *chap6-mech*))
```

Here, we have created: a mechanism data structure representing [4.1]-[4.3]; a list of run parameters, analogous to the Brusselator run-parameter list shown earlier; and a graph-parameter list that specifies how concentrations will be graphed. Finally, we create a mechanism object from the original mechanism data structure. We now evaluate the following expression:

```
(do-simple-graphed-run
 *chap6-mech4-object*
 *chap6-mech4-run-parameters*
 *chap6-mech4-graph-parameters*)
```

and the numerical simulation is run, constructing the graph shown in Figure 6.1.

So far, there is nothing particularly new about our example: we have simulated our mechanism much as we did in Chapter 4. However, once the

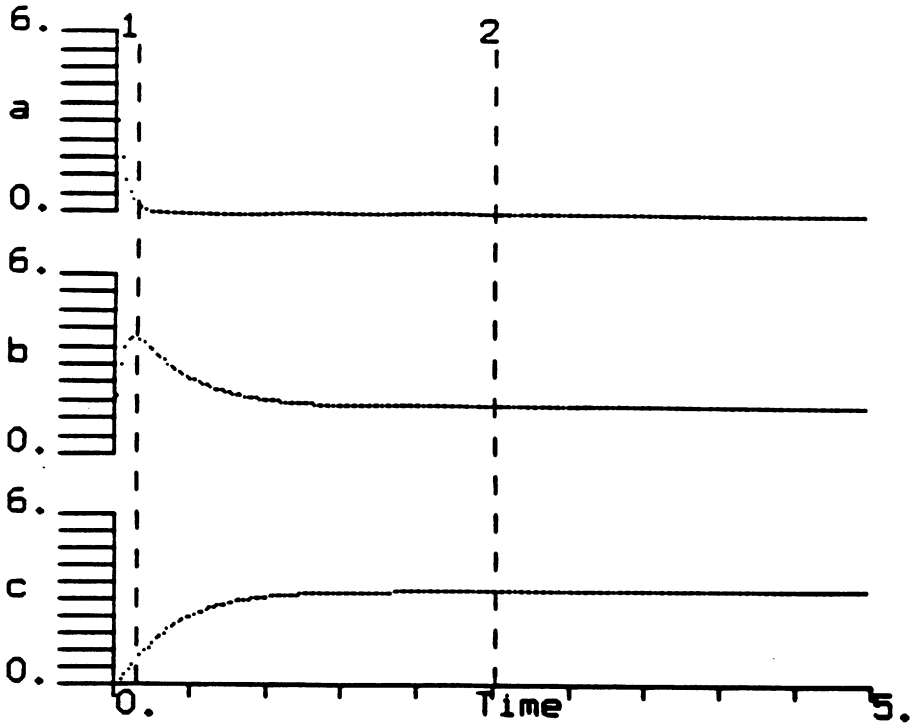


Figure 6.1: Simulating Mechanism [4.1]-[4.3]

simulation is complete, the Workbench prints out the following post-mortem of the run:

```
We now begin to analyze the run results.
The grain for the analyzed episode history will be: 0

The feature history of the run:
((short)
 (large-decrease a) (rapid-decrease a) (steady-decrease a)
 (large-increase b) (rapid-increase b)
 (large-increase c) (rapid-increase c) (steady-increase c))

((long)
 (large-decrease a) (rapid-decrease a) (steady-decrease a)
 (large-decrease b) (steady-decrease b)
 (large-increase c) (steady-increase c))

((long) (final)
 (large-decrease a) (rapid-decrease a) (steady-decrease a)
 (steady-state b) (slow-decrease b) (steady-decrease b)
 (steady-state c) (slow-increase c) (steady-increase c))


Analysis variables are now initialized.
```

```
The apparent final state of the system:
    ((a (probable-steady-state zero a) () () () ())
     (b (steady-state nonzero b) () () () ())
     (c (steady-state nonzero c) () () () ()))

done
```

Here, the Workbench has chunked the time history of the run into three episodes, and has printed out the qualitative features associated with those episodes. That is, the Workbench has decided that the history of the run can be summarized by a short period during which B and C rise rapidly and A declines rapidly; a long period during A and B experience large decreases in concentration, while C continues to rise; and finally a long period during which A continues decreasing, while B and C are at near-steady-state concentrations. The last portion of the printout, in which the Workbench assesses the final state of the system, will be discussed later in this chapter.

To understand how the Workbench reached its various decisions about the salient features of the run, we can examine the episode history itself. Here, for instance, is the first episode of the simulation as printed by the Scheme interpreter (comments have been added):

```
#(.025                                ; starting time of episode
  .125                                ; duration of episode
  #(                                  ; a vector of information for focus species
   #(a 3.033854166666667              ; starting concentration for A
       (((0 100.)))                   ; most important step (and percent of last
                                      ; delta that it represents) for A
       (3.033854166666667 .025)       ; largest conc for A and time of that conc
       (.24952736718289784 .15)       ; smallest conc for A and time of that conc
       (-.16171086886101962 .15)      ; maximum delta for A
                                      ;  and final time of that delta
       (-1.192999945746528 .05)       ; minimum delta and final time
       () ())                         ; extra symbolic information slots
                                      ; similar sturctures for B and C
                                      ; have been omitted
     )
       5)                             ; number of time-steps of this episode
```

This structure indicates that the first recorded episode lasted 0.1 seconds, from 0.025 to 0.125.[3] Since this is the shortest episode in the history, it is deemed a "short time" (hence the inclusion of this feature in the printout above). Moreover, we see that species A declines from a concentration of 3.03 to a final concentration of 0.25 during this episode[4]. This decline is large enough to be deemed "large," and it happens in a brief enough time to be deemed "rapid." Moreover, since the maximum delta for A during any time-step is -0.16, we know that all deltas for A must be negative in this episode and that A thus experienced a steady decline. Similar data structures for B and C (omitted above) are used to generate feature descriptions for these species as well.

By examining the episode history further, we discover that the overall episode history of the run consists of three episodes beginning at times 0.025, 0.15, and 0.2525. The boundaries between these episodes are shown in Figure 6.1, and indicate specifically the times at which the most important step for B changed from [4.1] to [4.2], and then from [4.2] to an effective tie between [4.2] and [4.3]. It is worth looking at Figure 6.1 for a moment and comparing it to the feature history printed out earlier; functionally, what the Workbench is trying to do is to report on the simulation results in the way that an observant (but perhaps not very imaginative) assistant might.

6.1.6 Looking for Patterns in Episode Histories; Limitations of the Episode Data Structure

Once an episode history has been constructed (as in the example of the previous subsection), the Workbench tries to find repeating patterns—specifically, those indicative of the presence of oscillations—within the episode

[3]The initial episode does not start at time 0 because certain episode initialization routines take place when an integration step is actually performed. Thus, the Workbench first recognizes an existing episode after the first time-step for integration has been completed. This "time-offset" in starting the initial episode does on very rare occasions cause slightly unexpected results for qualitative analysis, due to the fact that the initial episode begins with different concentrations than the overall simulation itself. It is thus probably worth amending in future versions of the program.

[4]See the previous footnote regarding the value of A's starting concentration; the final concentration is obtained from the starting-concentration slot for A in the second episode.

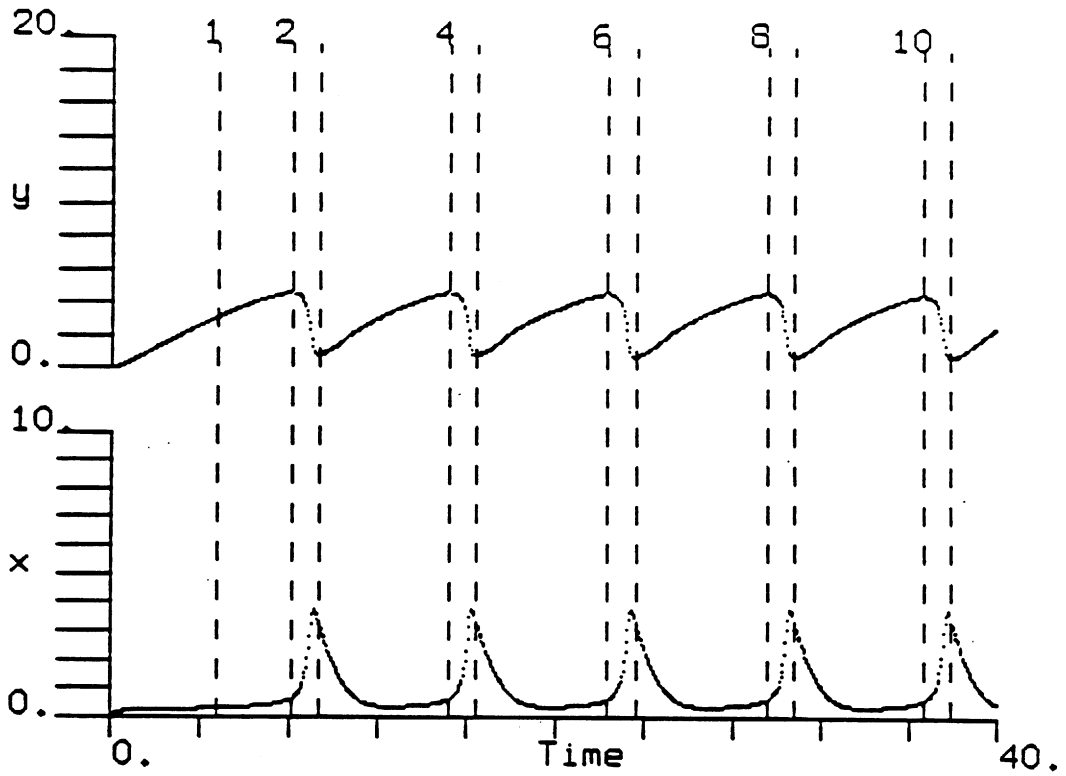


Figure 6.2: Brusselator Simulation

record. The algorithm used for this purpose is rather involved, as it is intended to look for possibly complex repeating patterns (this notion will be explained more fully later). Basically, however, the idea is that by examining the episode history for patterns in which "similar" episodes recur at regular intervals, the Workbench can spot the occurrence of patterns such as stable oscillations.

The example in the previous subsection produced an episode history that obviously did not contain repeating patterns; thus, in order to illustrate this capability we need to use a more complex mechanism such as the Brusselator. If we simulate the Brusselator using run- and graph-parameters analogous to those in the previous subsection, we obtain a concentration-versus-time graph as shown in Figure 6.2. (Here, the episode boundaries have also been included on the graph, as in Figure 6.1.)

The qualitative analysis printed out by the Workbench reads in part as follows:

```
We will now examine the coarse-grained episode history
for possible oscillations.
```

```
((long)
 (large-increase x) (steady-increase x)
 (large-increase y) (steady-increase y))
((large-increase x) (steady-increase x))

     ((short)
      (large-increase x) (rapid-increase x)
      (large-decrease y) (rapid-decrease y) (steady-decrease y))
     ((long)
      (large-decrease x) (rapid-decrease x)
      (large-increase y))


     ((short)
      (large-increase x) (rapid-increase x)
      (large-decrease y) (rapid-decrease y))
     ((long)
      (large-decrease x) (rapid-decrease x)
      (large-increase y))
```

Two more two-episode pairs exactly the same as the two above.

```
     ((short)
      (large-increase x) (rapid-increase x)
      (large-decrease y) (rapid-decrease y))
     ((final)
      (large-decrease x) (rapid-decrease x) (steady-decrease x)
      (large-increase y) (rapid-increase y) (steady-increase y))
```

Here, the Workbench has indicated (by the indentation) that it has found repeating patterns of two-episode pairs: a short episode in which `X` increases rapidly, followed by a longer episode in which `X` decreases rapidly and `Y` increases. These repeating pairs are apparent in Figure 6.2, beginning with the third episode.

The way in which the Workbench looks for repeating patterns involves, as an initial step, trying to classify "equivalent" episodes within the history. The basis for this classification is just the defining characteristic for any episode—namely, the ordering of terms in the differential equations for the

focus species. In other words: if two episodes are identical in their respective term-orderings,[5] we can tentatively assume that these are "equivalent" episodes. Beyond this, the Workbench will not claim that two episodes are equivalent if they have "inconsistent" feature descriptions: for instance, if two episodes are described as "long" and "short," respectively, the Workbench will not allow them to be classified as equivalent regardless of their term-orderings.

The overall algorithm used by the Workbench to find repeating patterns is described in detail in Appendix C. Basically, the program looks first for large-scale patterns, and will then attempt to find smaller-scale repetitions within those patterns. As an illustration of the idea, consider the letter-sequence below, and suppose that each letter represents an "episode-type":

```
A B C D C D A B C D C D A B C D C D A B ...
```

This sequence contains an apparent six-letter repeating pattern:

```
[A B C D C D] [A B C D C D] [A B C D C D] ...
```

Moreover, within each sequence we can see a smaller pattern of repetition:

```
[A B [C D] [C D]] [A B [C D] [C D]] ...
```

The Workbench's algorithm is able to find repeating patterns of this type, as long as it can unambiguously identify equivalence between instances of distinct "episode-types" and the record is long enough to identify at least several complete oscillations.

The problem of finding repeating patterns in episode histories is not a trivial one, despite the apparent straightforwardness of Figure 6.2. It was this task that originally motivated the identification and filtering of "transition episodes," as discussed earlier in this section: we saw that the occasional transition episode inserted in the record can make it difficult to find "clean" repetitive patterns among episodes.

More generally, the episode data structure has proved to be unsatisfactory in practice as the sole basis for analyzing the behavior of complex reaction

[5] In practice, the Workbench usually employs only the most important ("zeroth") term in determining both episode boundaries and episode equivalence.

systems. This is true for two main reasons. First, it is not easy to look for a variety of common patterns, such as damped oscillations, by examining the episode record alone. To pursue this particular thought, we can summarize what happens when the Brusselator is run with an altered set of rate constant values resulting in damped oscillations: we see one "repetition pattern" of episodes (a two-episode pattern) give way to another (four-episode) pattern. It is not easy to analyze this succession of repetition patterns in such a way as to make apparent that these correspond to damped oscillations in the original results.{24}

More troubling is the fact that the basis for the episode definition itself—the ordering of differential equation terms—becomes problematic for especially stiff systems. The problem arises from potentially large variation in the magnitudes of individual terms even within extremely tiny regions of state space: this means that term-orderings may change even within portions of the numerical record that appear relatively flat. On reflection, this is not too surprising: in a sense, the fact that a system is stiff indicates that the differential equation terms are highly variable and do not give a particularly "clean" indication of the local behavior of the system. (If they did, then the next state of the system after one more time-step would be indicated by the differential equation terms at the previous time-step, suggesting that an explicit integration method would be workable; but in point of fact, the initial explicit prediction made by a numerical integrator for a stiff system can be very far from the ultimate next-state value arrived at from iterative corrections. To put the matter another way, if differential equation terms were a highly accurate reflection of local system behavior, we would expect a forward Euler integrator to be more-or-less adequate in finding the next state of the system; and the fact that such an integrator is wildly inaccurate should make us suspicious of the value of any individual term-ordering as reflecting much information about the system's behavior.)

When stiff systems are run on the Workbench, the resulting episode record is usually very large and difficult to parse—even when some of the "refining" strategies mentioned earlier are used.[6] What this has meant in practice is

[6]For instance, we might try using only a few elementary steps as the basis of episode boundaries. This strategy does in fact deserve more examination; but preliminary experiments with the chlorite-iodide mechanism to be discussed later suggest that the difficulties

that the elaborate pattern-recognition techniques implemented in the Workbench, which are most appropriate for use with complex systems, have not proven useful: those mechanisms that exhibit the complex behavior do not produce "clean" enough episode records for analysis. As a result, a new data structure has been introduced to provide more adequate analysis of complex systems; and this data structure is discussed in the following section.

6.2 Zero-Crossings of Derivatives

To help in identifying patterns common to oscillating mechanisms, the Workbench maintains a history list of *derivative zero-crossing* data structures. This history keeps track of local maxima and minima in the numerical record of concentrations versus time, and, in concert with the episode history, allows the Workbench to make more reliable judgments about the behavior of complex mechanisms.

6.2.1 The Zero-Crossing Data Structure

As the Workbench performs a numerical simulation, it notes those time-steps in which the (numerical) derivative of any focus species' concentration changes sign—corresponding to a local maximum or minimum in the numerical record. When such an event occurs, the program constructs a new derivative zero-crossing data structure consisting of the following information:

- The species for which a maximum or minimum has been encountered;
- The delta of the previous time step;
- The delta of this time step;
- Current concentrations of all species.

The newly-created zero-crossing structure is then added to a running history

in analyzing the episode record do not disappear when this strategy is employed.

of these structures for later analysis.

6.2.2 Looking for Oscillation Patterns

Once a numerical simulation has been completed and the process of qualitative analysis has begun, the Workbench examines the list of zero crossings to find successive sets of n crossings for a given species that have the following properties:

- The corresponding initial zero crossings of successive sets occur at approximately constant time intervals. (For instance, if the initial zero crossing of the first set occurs at time t, and the initial crossing of the second set occurs at time $t + 12$, then the initial crossing of the third set ought to occur at time $t + 24$.)

- The concentrations of all species are at comparable values for each pair of corresponding zero crossings within successive sets, or if not, the percent change in all species is comparable.

The purpose of the first check is to find a series of sets of n crossings that occur at regular intervals; the second check is to ensure that concentrations (or changes in concentrations) are not extremely disparate at comparable points in the hypothetical oscillation.[7] The Workbench scans the zero-crossing history with increasing values for n up to 8 (that is, a complete identifiable oscillation may consist of up to eight zero-crossings, corresponding to four separate "humps").

Having found sequences of apparently-repeating zero crossings, the Workbench now examines those crossings more closely to see if they seem to comprise a straightforward, recognizable pattern such as stable or damped oscillations. (The check for stable oscillations, for instance, looks to see whether the oscillations appear to have near-constant amplitudes; or, if the amplitudes are increasing, the program checks to see whether the ratio of successive pairs of amplitudes seems to be approaching 1.) In addition, the program attempts to check the oscillations to ensure that they are not "noise"—that is, arti-

[7]The algorithm used by the Workbench is provided in more detail in Appendix C.

facts of integration.[8] Even the possibility of some more exotic pattern such as chaotic oscillations is acknowledged by the program, but this is defined in a negative way: if no "obvious" pattern is detected, and the oscillations appear to be long-lived, the Workbench suggests that they may be chaotic.[9] A brief description of the Workbench's oscillation-recognition heuristics may be found in Appendix C.

It should be noted that there are certain types of complex oscillation patterns that the Workbench is unable to deal with at present. Oscillations with a large number of "bumps"—for instance, a single large peak followed by four small ones—are too complex to be recognized by the Workbench.[10] Likewise, oscillation patterns with highly regular "sub-patterns" (such as one large peak followed by two nearly-identical small ones) may not be recognized.[11] And finally, very erratic oscillations (possibly chaotic) may not be recognized if they exhibit wide variations in period and amplitude—though even in this case, the Workbench will at least note during its qualitative analysis that *some* kind of oscillation occurred.

Although the Workbench attempts to make some judgment about the type of *every* repeating pattern of zero crossings that it finds, the program focuses most of its interest on the very last pattern, as we will see later in this chapter. In particular, the final repeating pattern is assumed to give some clue as to the asymptotic state of the simulation—e.g., whether the system seemed to be approaching a steady state, or perhaps whether it appeared to

[8]Occasionally, one can find that a species near an equilibrium concentration will "oscillate" around that concentration on alternate time steps. This oscillation is due only to the discrete time steps taken by the integrator and can be detected by its (typically) small amplitude and (two time-step) period.

[9]The Workbench currently makes no automatic attempt to distinguish between, e.g., quasiperiodic oscillations and chaotic oscillations. In this and the following chapter, however, we describe an additional feature of the Workbench that the user may employ to help distinguish between these cases.

[10]Examples of such complex oscillations in a laboratory system (the Belousov-Zhabotinskii reaction) may be seen in Figure 8.3 of {28}.

[11]The algorithm that looks for repeating sequences of zero crossings first flags the small-period oscillations, and then does not seek longer oscillations that might "interfere" with oscillations already found. Thus, in the current example, each pair of two small peaks would be noted as a brief oscillation (of unidentifiable type)—and many such brief oscillations would be noted—but the larger three-peak sequence would be missed.

be approaching a limit cycle.

6.2.3 Coarse- and Fine-Grained Oscillations

In addition to the analysis described in the previous section, the Workbench also uses a more "coarse-grained" approach that filters the zero crossing history to retrieve only those zero crossings in which the concentration of the given species changes by at least a small percentage from its neighboring zero crossings.[12] This filtering step was motivated by the "noisiness" of certain complex systems integrated with the Gear algorithm, in which small (and somewhat irregular) variations in concentration could be misinterpreted as derivative zero-crossings (even in clearly flat regions of the numerical record). To compensate for these fine-grained irregularities, it proved worthwhile to look for regularities among the more "exaggerated" zero crossings.

The Workbench uses a similar (though a bit more "lenient") algorithm to look for repetitive sequences in the coarse-grained zero crossing record as it does in the fine-grained case described earlier. Having located these sequences, the Workbench seeks to classify them (as "stable," "damped," and so forth) in exactly the same way as described earlier.

6.3 Constructing Feature Summaries of Simulations

Thus far, we have seen how the Workbench constructs (and analyzes) an episode history of a given simulation; and we have likewise seen how it constructs (and analyzes) a history of local minima and maxima—i.e., derivative zero crossings—encountered during a simulation. Having performed these tasks, the program now undertakes an additional stage of analysis that makes use of both these data structures. The purpose of this next stage is to make some judgment about the overall behavior of the simulated system, with particular attention to its apparent asymptotic behavior. These judgments are summarized in a "feature summary" list that is constructed for each focus species in the simulation.

[12]The current default "percentage-difference" value is 2.5.

The feature summary for each species consists of descriptor elements that provide information about:

- Whether the species seemed to be at a near-constant concentration at the end of the simulation; and if so, whether that concentration was nonzero or zero.

- Whether some oscillation appeared to be going on at the end of the simulation; and if so, a descriptor of that oscillation's type (stable, unstable, damped, possibly chaotic, probably noise, or unidentified). This information is provided based on both the "fine-grained" and "coarse-grained" zero crossing information.

- Whether multiple fine-grained oscillation types (i.e., oscillations with different numbers of "bumps" or with markedly different periods) were encountered during the simulation.

- Descriptors for the types of all fine- and coarse-grained oscillation patterns encountered.

The first of these descriptors—providing a judgment about whether the concentration of a given species seemed to be "flattening out" at the end of a run—involves checking the episode history for several types of information. The final episode is checked for the values of both first and second derivatives of the concentration-versus-time graph; these are used to estimate the flatness of the graph at the end of the run. Additionally, the presence of a final long episode in which the species is deemed to be at a steady state is taken as corroborative evidence that the species is indeed at a near-constant value. In exploring the question of whether the species concentration seems to be approaching zero or not, the Workbench uses the final first and second derivative values to estimate the given species' concentration after a long time past the end of the simulation (where "long" is defined according to the episode history, as described earlier). If this estimated future value is sufficiently small (compared to the maximum concentration of the species during the run), and if additionally certain other conditions are met (such as a negative first derivative and positive second derivative at the end of the run), then the program hypothesizes that the concentration is approaching

zero at the close of the run.

The descriptors dealing with oscillations are constructed according to the same criteria as described in the previous section; the only point worth mentioning here is that the descriptor for "final oscillating state" must make some judgment as to whether the oscillation pattern was still identifiable at the end of the run. In the case of damped oscillations, it is quite conceivable for the Workbench to conclude both that a species was approaching a (nonzero) steady state and that it was experiencing damped oscillations at the end of the run.

One more capability of the Workbench concerning oscillations also deserves mention: when the program observes an oscillation pattern for a given species, it prints out a brief report of that pattern on the display screen, including the time of onset of oscillations, the average period, the average amplitude (for stable oscillations), and the average ratio of successive oscillations (for damped or unstable oscillations). In reporting a stable oscillation, the program also prints out the "typical episode sequence" for this oscillation: that is, the most commonly found sequence in the episode history that overlaps with a single period. Thus, the Workbench is able to provide the user with a conceptual link between the observed oscillations and the reactions within the mechanism that give rise to those oscillations.

6.4 Saving Qualitative Analysis Results; Parameter Space Checks

If desired, the results of the Workbench's qualitative analysis may be saved in an external file. To do this, a new parameter, `save-qualitative-results?`, is added to the run parameters list and given a value of `true`; a (string) pathname must also be specified in a `qualitative-result-filename` parameter. With these additions, the Workbench will save out a variety of its qualitative-analysis data structures at the end of its simulation.

Currently, the contents of a qualitative analysis file include the following:

- The episode history for the run, and the "filtered" episode history (in which transitional episodes have been removed).

- The feature history corresponding to the filtered episode history.

- The derivative zero crossing information maintained for the run.

- A data structure indicating the repetitive patterns detected in the zero crossing information.

- The overall feature summary for the run.

- Other information about the saved run, including the original reaction mechanism, focus species, initial concentrations, integration method, time-step (dt) for integration, start and end times for the simulation, and several more.

Saving the results of qualitative analysis is particularly important for performing parameter space checks. As mentioned earlier, the Workbench contains procedures that allow the user to run a simulation for a given parametrized mechanism repeatedly, varying the parameter (or pair of parameters) over a pre-specified range.[13] An example was described in Chapter 4; there we created and simulated a "parametrized Brusselator," with two rate constants as its parameters. It was also mentioned at that time that parameter space scans must be accompanied by a "template filename" for saving numerical, pictorial, and qualitative-analysis results.

The procedure that performs parameter space scans actually makes partial use of qualitative analysis results as it performs repetitive simulations. In particular, the program checks the feature summaries after each simulation; and if it finds that a given simulation resulted in no conclusion—neither a steady state nor an oscillation—the program redoes the simulation for a longer period, and with a smaller timestep. (Note, by the way, that the "oscillation" in this case need not be stable; the program will accept even an oscillation of unidentified type for these purposes.) This additional try is only performed once: if the second attempt at simulation still results in no identifiable feature summary, the program gives up and goes on to the next parameter value. Nevertheless, this is a worthwhile feature that partially

[13]At present, the Workbench is only able to vary at most two parameters in such a scan.

automatizes the alteration of simulation parameters based on qualitative interpretation of the results.

Once a parameter space scan has been performed, the results may be retrieved and summarized in a parameter space graph. Currently, the set of Workbench procedures available for this purpose are a bit restrictive: only a few types of graphs may be constructed. However, it is possible to create a graph in which two parameter ranges are used as axes, and (e.g.) the stored feature summaries are used to generate symbolic entries in the graph. Complete examples of parameter space scans will be shown in Chapter 7.

6.5 Checking Predictions

In the previous chapter, we saw that the Workbench can often make interesting predictions about the behavior of a mechanism under simulation, and can add symbolic notations to a mechanism object indicating the predictions that it has made. When a simulation is later performed, the Workbench uses the numerical record (and associated qualitative analysis) to check those predictions that it made earlier.

As an example of this capability, we can once more consider mechanism [36] from the previous chapter, reproduced below:

```
[5]       A + B <--> C <--> D
```

In analyzing this mechanism graphically, the Workbench notes that it is a reversible zero-deficiency mechanism and thus can be expected to reach a unique stable equilibrium with positive concentrations of all species:[14]

```
Rule number 2.1 -- Deficiency Theorem part 1 is firing

If the deficiency of the mechanism is 0 and the
mechanism is weakly reversible, then the mechanism must
reach equilibrium at positive (non-zero) concentrations
of all species.
```

[14]The rate constants used for the actual mechanism data structure are the same as in the Chapter 5 example.

This conclusion becomes the basis of a prediction—specifically, a "necessary equilibrium" prediction[15]—and a symbolic descriptor of the newly-made prediction is added to the mechanism object being analyzed. When the mechanism is later simulated (with all four species as focus species), the Workbench conducts its qualitative analysis, in the course of which it checks the necessary equilbrium prediction that it made earlier:

```
The apparent final state of the system:
    ((a (steady-state nonzero a) () () () () ())
     (b (steady-state nonzero b) () () () () ())
     (c (steady-state nonzero c) () () () () ())
     (d (steady-state nonzero d) () () () () ()))

CHECKING PREDICTION:
Mechanism must reach equilibrium with all nonzero concentrations.

PREDICTION MET:
(a b c d) all appear to be at nonzero steady state values.
```

Currently, the program does no additional processing of behavioral predictions: it does not specifically flag unmet predictions, nor does it store the results of its checks in an external file. Thus, the only capabilities implemented in the Workbench are those that display the checking of predictions, and (when they occur) the meeting of predictions.

Additional examples along these lines will be encountered in the following chapter.

6.6 Additional Analysis

We have seen in a variety of cases how the Workbench is able to make use of the qualitative analyses that it produces: it can create graphs that summarize the analyses, it can use the analyses to check predictions about mechanisms, and it can even use the *absence* of an informative analysis to motivate re-doing a simulation. The point of mentioning these examples once more is to stress that the purpose of qualitative analysis is not merely to summarize simulation results for the user, but also to provide information

[15]As described in the previous chapter.

of various kinds to other programs. Thus, we can envision writing additional Workbench files to make use of the qualitative results stored by previous Workbench simulations.

As an example, consider the problem of analyzing apparently irregular oscillations in a simulated chemical system. Suppose that a chemist observes such a pattern, and wants to know whether what she is seeing is quasiperiodicity or chaos. This distinction, as it happens, is difficult if not impossible to make by simply viewing the time series. Moreover, the features of the Workbench encountered thus far offer no help in this regard, since both a quasiperiodic and chaotic system would simply be flagged as showing possibly chaotic (or perhaps "unidentified") oscillation types.

In this situation, the most common numerical strategy for the dynamicist is to take a power spectrum of the system using an FFT algorithm. A quasiperiodic system with two fundamental frequencies will show two distinguished sharp peaks at those frequencies, and smaller (but still visible) peaks at other known frequencies derived from "small" sums and differences of the fundamental frequencies.{76} A chaotic system, on the other hand, may be distinguished by a broad frequency spectrum.{29} Using an FFT algorithm on the numerical results can thus distinguish between the quasiperiodic and chaotic systems, where simply observing the time series by eye cannot.

However, there is still a problem. The oscillations to be analyzed may not begin at the onset of simulation, but rather sometime during the simulation; that is, there may be an "induction period" before the irregular oscillations begin. Moreove, the oscillations may actually end at some time during the simulation as well; it is not hard to imagine a situation in which a chemical simulation exhibits irregular oscillations for a certain period of time, but then switches to a different mode of behavior. Thus, if the chemist wishes to run an FFT algorithm on the numerical results, she had better have some estimate of where the puzzling oscillations begin and end; otherwise, she risks sending misleading numerical data to the algorithm and obtaining a wrong conclusion.

Seeing where oscillations begin and end may not seem like much of a chore for a particular simulation, but when it has to be done reliably over many simulations, it is precisely one of those tasks that might be profitably

left to the computer. The Workbench, we recall, does not automatically distinguish between quasiperiodic and chaotic oscillations, but it *does* retain information that we can use to find where oscillation patterns begin and end within the numerical record. Thus, if we have the Workbench store both numerical and qualitative results, we can create a new file to retrieve only that portion of the record in which "problematic oscillations" occur, and to send those results to an FFT algorithm.

In point of fact, an additional file *has* been written with precisely this capability. This file is not intended to be seen as a "basic" part of the Workbench system, but as an illustration of a conceivable set of library files that could be added as needed for additional processing of Workbench results. An example showing this post-analysis file will be included in the next chapter; but again, the main point of this example is to indicate how a large number of new files, combining numerical and qualitative results, could well be added to the basic Workbench structure. For example, we might imagine a file to obtain the fractal dimension of an attractor in phase space (using a box-counting algorithm{6}); this file could use the results of qualitative analysis to find the "transient" portion of the numerical record that may safely be ignored. Or again, we might have a file that looks over a series of runs in a parameter scan and graphs the equilibrium concentration of some species *only* for those runs that achieved an identifiable equilibrium state. These hypothetical projects are reasonable precisely because the Workbench stores its results—numerical and qualitative—in a form that may be read by still other computer programs.

Chapter 7

The Workbench in Operation

This chapter is devoted to showing examples of the Workbench in use. We begin with an exploration of relatively simple mechanisms, and then go on to discuss how the Workbench can be used to study more complex models.

7.1 The "Tiny Mechanism"

As an introduction—just to provide a sense of how the various portions of the Workbench come into play in a single example—we use the same "tiny mechanism" shown in Chapters 4 and 6:

```
[1.1]        A --> B   k1 = 20
[1.2]        B --> C   k2 = 2
[1.3]        C --> B   k3 = 1
```

As in Chapter 4, we create a mechanism data structure for [1.1]-[1.3]:

```
(define tiny-mechanism
  '(
     (  ( ((a 1))  ((b 1)) 20)
        ( ((b 1))  ((c 1)) 2)
        ( ((c 1))  ((b 1)) 1) )    ; step-list

     ()                            ; constants
     ()                            ; sources
     ()                            ; sinks
     ()                            ; driven-off
     ()))                          ; functions of time
```

We could now go ahead and simulate this mechanism numerically, as in the original discussion; but before doing this, we will first create a mechanism object and perform graphical analysis on it:

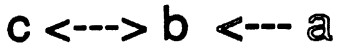


Figure 7.1: Graph of "Tiny Mechanism"
In the original Workbench output, a and the rightmost
arrow are shown in gray, all others in green.

```
(define tiny-mechanism-object
  (make-mechanism-object tiny-mechanism))

(analyze-graphically tiny-mechanism-object)
```

The Workbench now prints out the results of its analysis. First, the program asks if we want to see a graph of the original mechanism; we answer in the affirmative, and see the graph shown in Figure 7.1.

Continuing with its analysis, the Workbench prints out the following on the display screen:

```
Network-graph-mechanism:
a ---> b          Rate constant: 20
b ---> c          Rate constant: 2
c ---> b          Rate constant: 1


Number of complexes: 3
Number of linkage classes: 1
Dimension of stoichiometric subspace: 2
Deficiency: 0
Reversibility: none

Rule number 2.3 -- Deficiency Theorem part 2 is firing

If the deficiency of the mechanism is 0, and the mechanism
```

```
is not weakly reversible, then the mechanism cannot reach
equilibrium at positive concentrations of all species.


Rule number 3.1 -- Searching for Species to Drop from System is firing

If the zero deficiency theorem stipulates that the mechanism
does not reach equilibrium, then we first examine the
mechanism for species that are reactants only, and hence
cannot enter into any reactions that occur at steady state.
Sequence of Species to Drop: ((a))


Rule number 3.2 -- Creating and Examining Submechanism is firing

We have found that there are species that we can drop from
the original reaction mechanism. We now examine the smaller
mechanism defined by ignoring all reactions with droppable
species among the reactants.

Network-graph-mechanism:
b ---> c          Rate constant: 2
c ---> b          Rate constant: 1

Number of complexes: 2
Number of linkage classes: 1
Dimension of stoichiometric subspace: 1
Deficiency: 0
Reversibility: strong

Rule number 2.1 -- Deficiency Theorem part 1 is firing

If the deficiency of the mechanism is 0 and the
mechanism is weakly reversible, then the mechanism must
reach equilibrium at positive (non-zero) concentrations
of all species.

Adding prediction...(nonzero-equilibrium-2 ())
```

In prose, the Workbench has found that species A should be expected to decline toward a concentration of zero, and that the remaining mechanism (steps [1.2] and [1.3]) should reach a stable equilibrium state with positive

concentrations of both B and C. The program annotates the mechanism object with these predictions.

Having completed our graphical analysis of the mechanism, we now wish to do a numerical simulation. In preparation for this, we create lists of run parameters and graph parameters:

```
(define tiny-run-parameters
  '((starting-dt 0.025)
    (start-time 0.)
    (end-time 5.)
    (actual-starting-concs
     ((a 3.) (b 0.) (c 0.)))
    (integration-method runge-kutta)
    (focus-species (a b c))
    (steps-per-display 40)
    (end-conditions (end-time))
    (maintain-episode-history? ,true)
    (episode-change-depth 0)
    (graph-window ,*display-graph-window*)
    ))

(define tiny-graph-parameters
  '((graph-type numeric)
    (time-low 0.)
    (time-high 5.)
    (species-to-graph (c b a))
    (species-concentration-boundaries
     ((0. 3.) (0. 3.) (0. 3.)))
    (numeric-limits-for-graphs
     ((-1. -1. 6. 4.) (-1. -0.5 6. 4.) (-1. -0.5 6. 4.)))
    (window-regions-to-use
     ((0 0 400 100) (0 101 400 200) (0 201 400 300)))
    (axis-colors-to-use
     (8 8 8))
    (default-colors (1 1 1))
    (color-procedure ,graph-object-use-default-color)))
```

Now, simulating the mechanism for 5 seconds yields the graphs of concentration versus time shown in Figure 7.2. At the end of the run, the Workbench prints out its analysis of the results as in Chapter 6:

```
We now begin to analyze the run results.
```

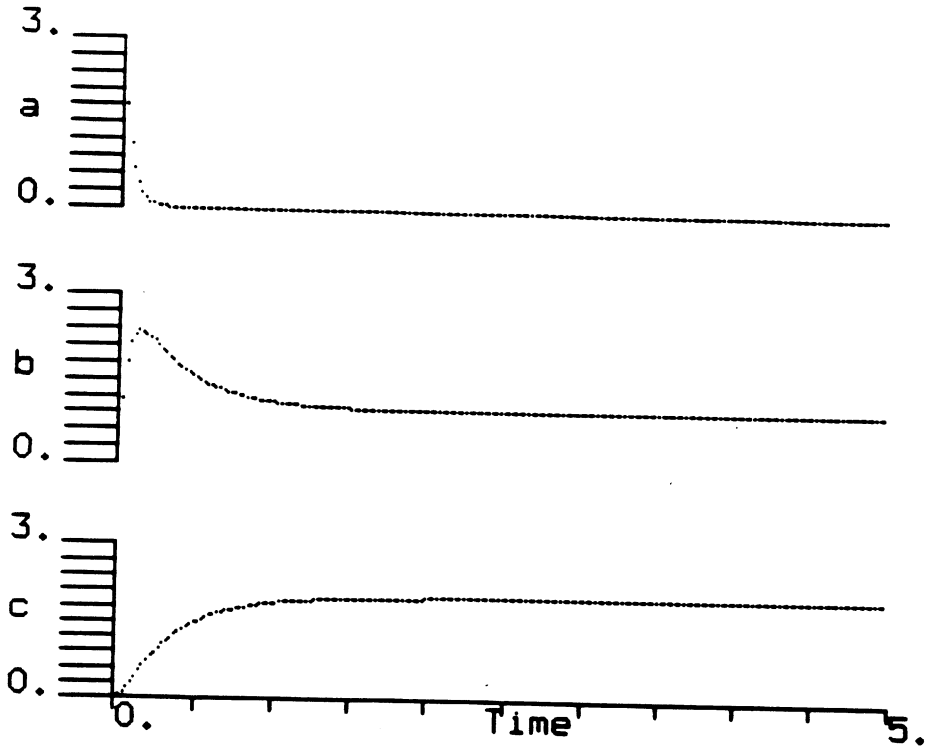


Figure 7.2: Simulation of "Tiny Mechanism"

```
The grain for the analyzed episode history will be: 0
The feature history of the run:
((short)
 (large-decrease a) (rapid-decrease a) (steady-decrease a)
 (large-increase b) (rapid-increase b)
 (large-increase c) (rapid-increase c) (steady-increase c))
((long)
 (large-decrease a) (rapid-decrease a) (steady-decrease a)
 (large-decrease b) (steady-decrease b)
 (large-increase c) (steady-increase c))
((long) (final)
 (large-decrease a) (rapid-decrease a) (steady-decrease a)
 (steady-state b) (slow-decrease b) (steady-decrease b)
 (steady-state c) (slow-increase c) (steady-increase c))
Analysis variables are now initialized.

The apparent final state of the system:
    ((a (probable-steady-state zero a) () () () () ())
     (b (steady-state nonzero b) () () () () ())
     (c (steady-state nonzero c) () () () () ()))

CHECKING PREDICTION:
Mechanism contains obvious declining sets.
```

```
PREDICTION MET:
((a)) all seem to have at least one species at zero concentration.

CHECKING PREDICTION:
Mechanism must reach equilibrium with some zero concentrations.

PREDICTION MET:
() all seem to be close to zero concentration and

PREDICTION MET:
(b c) all seem to be at nonzero steady-state concentrations.
```

Again, the Workbench has announced that it has verified the prediction made during the graphical analysis phase, and indicates (in the feature summaries) that species A seems to have approached a concentration of zero, while species B and C seem to be approaching nonzero steady state concentrations.[1]

While there is nothing terribly startling about this example, it does illustrate a natural pattern for using the Workbench. Specifically, we perform the following sequence of tasks:

- Analyze a mechanism graphically to find whatever may be known or surmised about it beforehand.
- Simulate the mechanism numerically.
- Summarize and classify the numerical results produced, verifying if possible whatever predictions were made earlier.

We now try the same pattern of use on a slightly more complicated mechanism.

7.2 N2O5 Decomposition

The mechanism of N2O5 decomposition has been used from time to time throughout this paper as a running example; here we bring together elements

[1]The penultimate "prediction met" line indicates that there were no zero concentration sets to check in conjunction with the check for steady states for both B and C.

from those previous discussions (as well as new material) to present a second illustration of the Workbench in use. Once more, the mechanism in question (as introduced in Chapter 2) is:

```
[2.1]    N2O5  -->  NO2 + NO3          k1 = 0.002
[2.2]    NO2 + NO3 --> N2O5            k2 = 0.001
[2.3]    NO2 + NO3 --> NO2 + NO + O2   k3 = 30.
[2.4]    NO + N2O5 --> 3NO2            k4 = 4000.
```

Here, the rate constants are the same as those used in the example shown in Chapter 5.[2] We recall from that discussion that the Workbench made several conclusions based on graphical analysis: first, that this mechanism must "run down" with zero concentrations of N2O5 and either NO2 or NO3; second, that there were several good candidates for "steady state" candidates (namely, N2O5, NO3, or both NO3 and NO). These conclusions are summarized below in excerpts from the Workbench's graphical analysis of the mechanism:

```
Number of complexes: 5
Number of linkage classes: 2
Dimension of stoichiometric subspace: 3
Deficiency: 0
Reversibility: none

Rule number 2.3 -- Deficiency Theorem part 2 is firing

If the deficiency of the mechanism is 0, and the mechanism
is not weakly reversible, then the mechanism cannot reach
equilibrium at positive concentrations of all species.


Rule number 4.3 -- Looking for asymptotic zero concentrations is firing

The mechanism cannot reach equilibrium with all nonzero concentrations.
It also does not contain any *obvious* declining species or sets.
We now look for those sets of species that might asymptotically
have zero concentration.
Possible sets of zero-concentration species: ((no3 n2o5) (no2 n2o5))


Subtracting the following zero set: (no3 n2o5)
```

[2]These rate constants are chosen to be consistent with the Chapter 5 example, and are not taken from the chemical literature.

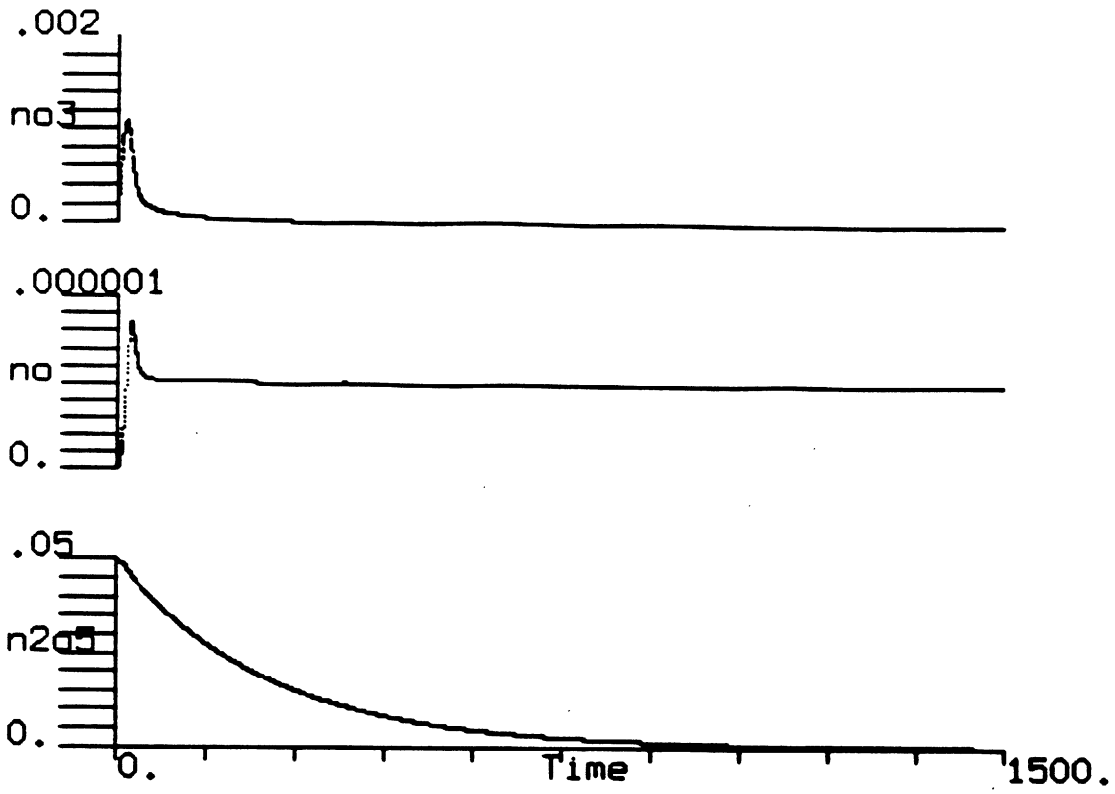


Figure 7.3: Simulation of N2O5 Decomposition

```
Adding prediction...(zero-final-state-1 (no3 n2o5))
This mechanism contains no elementary steps.

Subtracting the following zero set: (no2 n2o5)
Adding prediction...(zero-final-state-1 (no2 n2o5))
This mechanism contains no elementary steps.


Rule number 5.5 -- Steady State Candidates is firing

We now look for obvious steady state candidates in the mechanism.
We have found some steady-state candidates.
Possible sets of steady-state species: ((n2o5) (no3) (no no3))
Adding prediction...(steady-state-candidates)
```

Having analyzed the mechanism beforehand, we can now simulate it numerically as described in Chapter 4. When a simulation is performed with the Gear integrator for 1500 seconds, we obtain the graph shown in Figure 7.3. After the simulation is complete, the Workbench analyzes the numerical results that it has obtained. Excerpts of its printout are shown below:

```
We now begin to analyze the run results.
```

```
The grain for the analyzed episode history will be: 0
The feature history of the run:
((short)
 (steady-state n2o5) (steady-decrease n2o5)
 (large-increase no) (rapid-increase no) (steady-increase no)
 (large-increase no2) (rapid-increase no2) (steady-increase no2)
 (rapid-increase no3) (steady-increase no3))
((steady-state n2o5) (steady-decrease n2o5)
 (large-increase no) (rapid-increase no) (steady-increase no)
 (large-increase no2) (rapid-increase no2) (steady-increase no2)
 (large-increase no3) (rapid-increase no3) (steady-increase no3))
```

[20 more episodes]

```
((long) (final)
 (large-decrease n2o5) (steady-decrease n2o5)
 (steady-state no) (slow-decrease no)
 (slow-increase no2) (steady-increase no2)
 (large-decrease no3) (steady-decrease no3))


The apparent final state of the system:
    ((n2o5 (probable-steady-state zero n2o5) () () () () ())
     (no (steady-state nonzero no) ()
         (multiple-oscillations no)
         ((probably-noise) (probably-noise) (probably-noise) [29 more])
         () ())
     (no2 (probable-steady-state nonzero no2) () () () () ())
     (no3 (probable-steady-state zero no3) () () () () ()))

CHECKING PREDICTION:
Mechanism must cease running with some species at zero concentration.

PREDICTION MET:
(zero-final-state-1 (no3 n2o5)) are all close to zero final concentration.

PREDICTION MET:
(zero-final-state-1 (no3 n2o5)) seem to have reached a zero steady state.

CHECKING PREDICTION:
There are steady-state possibilities to check.

PREDICTION MET:
(no3) all appear as if they may have long steady states.
```

```
PREDICTION MET:
(no no3) all appear as if they may have long steady states.
```

Here, the Workbench has produced a long (and not terribly informative) episode history whose major distinguishing feature is the long final episode in which NO and NO2 are increasing (slowly) to their asymptotically-approached steady-state values, and NO3 and N2O5 are decreasing toward zero concentration. The large number of intermediate episodes is mainly due to intermittent "ties" between steps in determining the changes in concentration of different species.[3] Additionally, small variations in the concentration of NO are read as "noisy" oscillations, which the Workbench ignores for the purposes of analysis. Even so, the program is able to identify asymptotic steady states for all species (two of which—NO and NO2—are at nonzero concentration). These findings are seen in the feature summaries, listed under the "apparent final state of the system." The Workbench goes on to check its findings against the "predictions" slot in the original mechanism object, and finds that its predictions have been verified.

As it happens, then, the predictions that the Workbench made before the simulation began have been met during this run: specifically, NO3 and N2O5 are both approaching zero concentrations, and both NO3 and NO prove to be appropriate steady-state candidates. Naturally, these conditions might not hold for every conceivable run of this mechanism. If the mechanism is run for a very brief time, for example, the program will fail to verify any predictions: both NO and NO3 experience brief "jumps" at the beginning of the run (before the steady state prediction can be verified), and likewise both NO3 and N2O5 do not approach zero concentration until some time has passed. Slightly more subtly, it is not hard to alter parameters for the Gear integrator to render analysis more difficult. For instance, if we use a single-precision Gear algorithm with a longer time-step (that is, a longer time between numerical printouts of the IMSL Gear algorithm), then the program is no longer able to recognize a clear steady state for N2O5 at the end of the run: the episode record becomes "muddied" with numerous short episodes resulting from small numerical errors within the integrator. In this case, then, the episode history

[3]For instance, the final seventeen episodes are distinguished only by alternating ties between steps 2.1 and 2.4 in determining the change in concentration of N2O5.

does not exhibit a long final episode (which the program uses as a signal that the system has reached a steady state).

In a sense, we should not be surprised by this dependence of the Workbench's performance on integrator parameters. After all, in order to make a sensible analysis of numerical results, those results have to be as free of numerical "artifacts" as we can ensure; if the integrator introduces small errors due (for instance) to roundoff, then the Workbench will have that much more difficulty identifying patterns such as steady states within the numerical record. Even so, the fact that the Workbench can be thrown off by small errors is a source of concern; the program is generally far less tolerant of "noisy" results than the user. Moreover, this seems like a natural path for program development: one can imagine strategies that the program could use to pick appropriate integrator parameters automatically. We will return to this topic in the final chapter.

7.3 Enzyme Kinetics

In this example we use the Workbench in its capacity as a parameter-space scanner to illustrate a "classic" example from kinetics textbooks. The reaction mechanism is that of enzyme catalysis:{43}

```
[3]        E + S  <--> X  <--> E + P
```

The mechanism illustrated in [3] involves four species: an enzyme (E), a substrate (S), a product (P), and an intermediate species (X, sometimes denoted ES). The idea here is that E and S react to form an intermediate enzyme-plus-substrate combination, which eventually releases a (transformed) product molecule, leaving the enzyme free to combine with additional substrate. The assumption is also made—a realistic assumption—that the initial enzyme concentration is low compared to the initial substrate concentration.

We can use the Workbench to investigate the question of how the initial rate of production of P varies with initial substrate concentration. Our mechanism is written as follows:

```
(define enzyme-mech
  '(
    (
     ( ((e 1) (s 1)) ((x 1))  20.)
     ( ((x 1)) ((e 1) (s 1))   4.)
     ( ((x 1)) ((e 1) (p 1))   8.)
     ( ((e 1) (p 1)) ((x 1))   2.))
    () () () () ()))
```

We could now, as in previous examples, use the program to perform graphical analysis on this mechanism; but the results are somewhat obvious. This is a strongly reversible, zero deficiency mechanism and will clearly reach an equilibrium state. Our interest, however, is not in the equilibrium state of this system but rather in its initial "transient" behavior. We therefore create a run parameters list that includes a parametrized value for the initial concentration of substrate:

```
(define enzyme-run-parameters
  '((starting-dt 0.005)
    (start-time 0.)
    (end-time 1.)
    (actual-starting-concs
     ((e  0.01) (p 0.) (s sconc)(x 0.))) ; note symbolic conc value
    (integration-method runge-kutta)
    (focus-species (x))
    (steps-per-display 20)
    (end-conditions (end-time))
    (save-numeric-results? ,true)
    (save-qualitative-analysis? ,true)
    (save-picture? ,true)
    (maintain-episode-history? ,true)
    (episode-change-depth 1)
    (graph-window ,*display-graph-window*)
    ))
```

In addition, we create lists of graph parameters and file template names (which will be omitted here for brevity), and a parameter range list:[4]

[4]The parameter name is provided with a special syntax that indicates that this is a concentration value, rather than a value within the mechanism data structure.

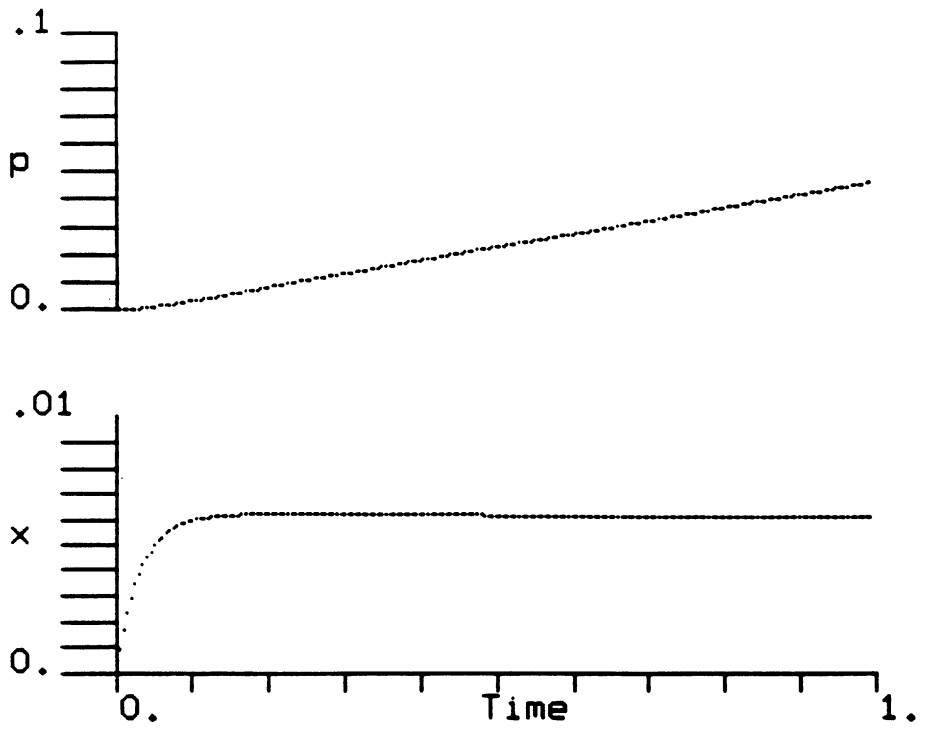


Figure 7.4: Enzyme Simulation ([S]o = 1M)

```
(define parametrized-enzyme-param-range-list
  '((parameter-name (conc sconc))
    (parameter-low-value 0.5)
    (parameter-high-value 10.0)
    (parameter-increment 0.5)
    (parameter-list ())))
```

Our purpose is to run repeated simulations of the mechanism [3], using initial substrate concentrations that vary between 0.5 and 10. We can tell the Workbench to perform the simulations by evaluating the following expression:

```
(do-single-parameter-space-scan-with-graphs
 *enzyme-mech-object*
 parametrized-enzyme-file-templates
 enzyme-run-parameters
 enzyme-graph-parameters
 parametrized-enzyme-param-range-list)
```

The results of two of the resulting sample runs are shown in Figures 7.4 and 7.5.

We can now read back the results of previous runs and graph the final concentration of P against the initial concentration of S. Because our simulations are very brief, the final concentration of P (after one second) is a

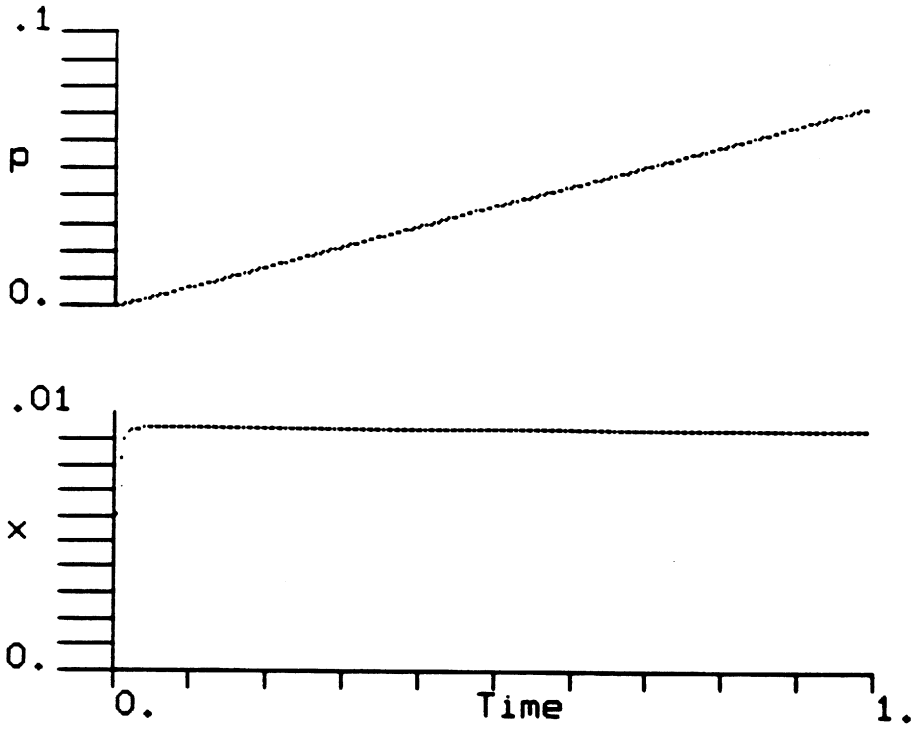


Figure 7.5: Enzyme Simulation ([S]o = 10M)

good approximation to the initial rate of production of P. We evaluate the following expression:

```
(graph-parameter-scan-files
 "/enzymedata/qualanal-"
 0 19 (make-retrieve-init-conc-final-conc 's 'p)
 (make-two-numeric-values-graphing-proc
  'So 'Pf 0 10 0 0.1))
```

and the Workbench produces the graph shown in Figure 7.6. The curve indicated by the points in Figure 7.6 is an experimental verification of the well-known Michaelis-Menten rate expression for enzyme kinetics:

$$\left(\frac{d\,[P]}{dt}\right)_{t=0} = \frac{(k3[E]o)}{1 + (k2 + k3)/(k1[S]o)}$$

$$= \frac{0.08}{1 + (0.6/[S]o)}$$

The purpose of this example is not to illustrate any of the more exotic capabilities of the Workbench, but merely to show that the program can be used

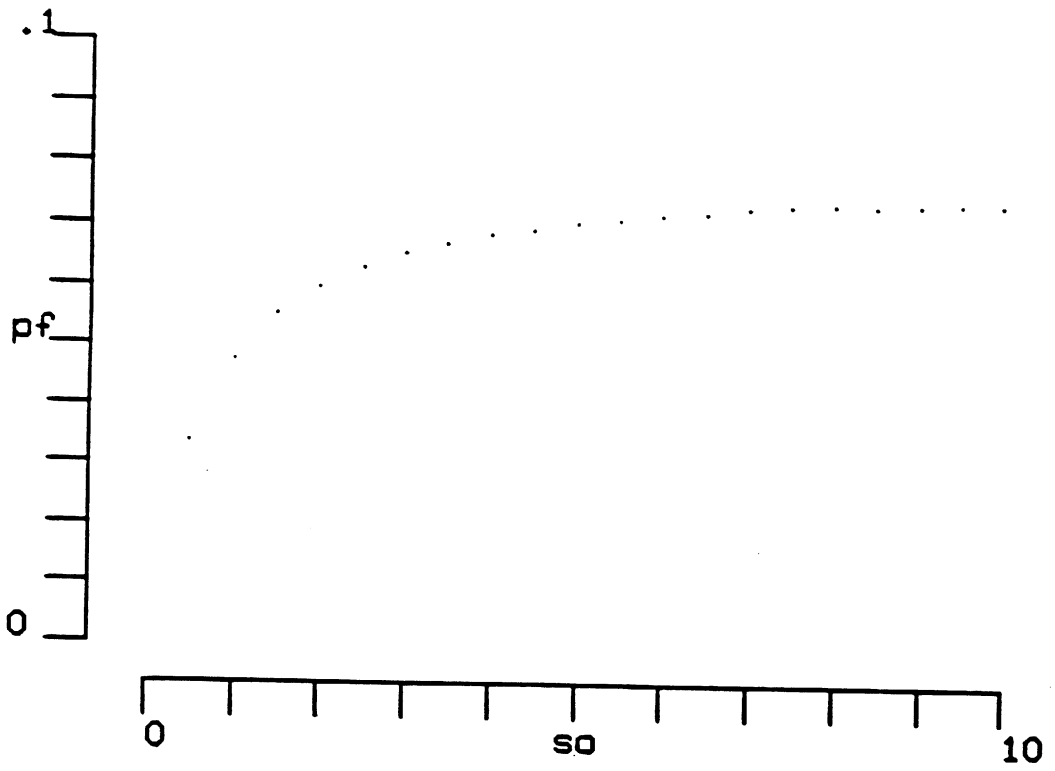


Figure 7.6: Final [P] vs. Initial [S]

as an effective tool for studying "mainstream kinetics." In fact, the Workbench's numerical capabilities alone provide a flexible (though admittedly slow) medium for simulation experiments.

7.4 Brusselator

The Brusselator mechanism, introduced in Chapter 4 and used in several examples thereafter, is a simple "abstract" mechanism capable of generating oscillatory behavior.{61} Once again, the elementary steps are:

```
[4.1]    A --> X
[4.2]    B + X --> Y + D
[4.3]    2X + Y --> 3X
[4.4]    X --> E
```

Here, we assume that [A] = 1, [B] = 3, and $k1 = k2 = 1$. In addition, the concentrations of D and E may be ignored (or treated as identically 0). For our purposes, the values of $k3$ and $k4$ will be treated as parameters to the mechanism.

Toward the end of Chapter 4, we showed how the Workbench could be used to run the Brusselator mechanism repeatedly, over a range of values for the parameters $k3$ and $k4$. We can resume that earlier example now, and show how the qualitative results generated for the series of runs may be used to create a parameter-space graph. Once more, our data structure for the parametrized Brusselator is defined as follows:

```
(define double-parametrized-brusselator
  '(
    (
     ( ((a))  ((x)) 1)
     ( ((b) (x)) ((y) (d)) 1)
     ( ((x 2) (y)) ((x 3)) k3)
     ( ((x))  ((e)) k4)
     )

    ((a 1) (b 3)) ;constant species
    () ;sources
    () ;sinks
    (d e) ; driven off
    () ;functions of time
    ))
```

And, as seen in Chapter 4, we create a parameter range list and a list of "templates" for filenames:

```
(define double-brusselator-param-range-list
  '((parameter-name1 k3)
    (parameter-low-value1 0.5)
    (parameter-high-value1 7.5)
    (parameter-increment1 1.0)
    (parameter-name2 k4)
    (parameter-low-value2 0.5)
    (parameter-high-value2 4.0)
    (parameter-increment2 0.5)
    (parameter-list ())))

(define double-parametrized-brusselator-file-templates
  '("/mydata/brussruns/numbers-"
    "/mydata/brussruns/qualanal-"
    "/mydata/brussruns/pict-"))
```

The lists of run and graph parameters were not included in the Chapter 4 illustration, but for completeness we present them here:

```
(define double-parametrized-brusselator-run-parameters
  '((starting-dt 0.05)
    (start-time 0.)
    (end-time 80.)
    (actual-starting-concs
     ((a 1.) (b 3.) (d 0.) (e 0.) (x 0.) (y 0.)))
    (integration-method runge-kutta)
    (focus-species (x y))
    (steps-per-display 160)
    (maintain-episode-history? ,true)
    (save-numeric-results? ,true)
    (save-qualitative-analysis? ,true)
    (save-picture? ,true)
    (episode-change-depth 0)
    (graph-window ,*display-graph-window*)
    ))

(define double-parametrized-brusselator-graph-parameters
  '((graph-type numeric)
    (time-low 0.)
    (time-high 80.)
    (species-to-graph (x y))
    (species-concentration-boundaries
     ((0. 10.) (0. 20.)))
    (numeric-limits-for-graphs
     ((-10. -3. 82 11.) (-10. -2. 82 22.)))
    (window-regions-to-use
     ((0 0 400 150) (0 151 400 300)))
    (axis-colors-to-use
     (8 8))
    (default-colors (2 3))
    (color-procedure
     ,(make-graph-object-use-reaction-table '(4 5 6 7)))
    ))
```

One point worth mentioning is the `color-procedure` entry in the graph-parameters list. This entry indicates that the color for a given species will correspond at any moment to the elementary reaction currently dictating the largest (absolute) change in concentration. Thus, if the largest local change in `[X]` is dictated by reaction [4.1], then `[X]` will be graphed in color

number 4; if reaction [4.2] is most important, then we will use color 5, and so on. (Examination of the mechanism shows that [Y] can only be graphed in either color 5 or 6.) By oberving color changes in the graphs, we can get a rough overall view of how terms within the differential equations for [X] and [Y] are changing, corresponding to the "coarse-grained" episode boundaries mentioned in the previous chapter.[5]

Finally, we create the appropriate mechanism object and evaluate an expression that now performs the series of simulations:

```
(define double-parametrized-brusselator-mech-object
        (make-mechanism-object double-parametrized-brusselator))

(do-double-parameter-space-scan-with-graphs
 double-parametrized-brusselator-mech-object
 double-parametrized-brusselator-file-templates
 double-parametrized-brusselator-run-parameters
 double-parametrized-brusselator-graph-parameters
 double-brusselator-param-range-list)
```

The Workbench now performs 64 individual simulations of the Brusselator in sequence; for each simulation, the numerical results are saved along with the graph of concentrations versus time (in pictorial form) and a summary of the results of qualitative analysis. Several representative graphs taken from this series of simulations are shown in Figures 7.7-7.9.

We now have a large number of files that can be retrieved and examined by other procedures. Currently, the Workbench has a limited repertoire for processing files of this kind: essentially, sequences of named files may be retrieved, and numeric or symbolic values from those files may be graphed in a few "cliched" ways. In this case, we use a procedure that retrieves a series of qualitative-analysis files with a given template (here, "qualanal-"), looks for the values of two rate constants (here, the third and fourth rate constant values), and graphs a symbolic representation of the feature summary within each file:

[5]Figures 7.7-7.9 are in black and white for the sake of reproducibility, so our color-coding is not depicted here.

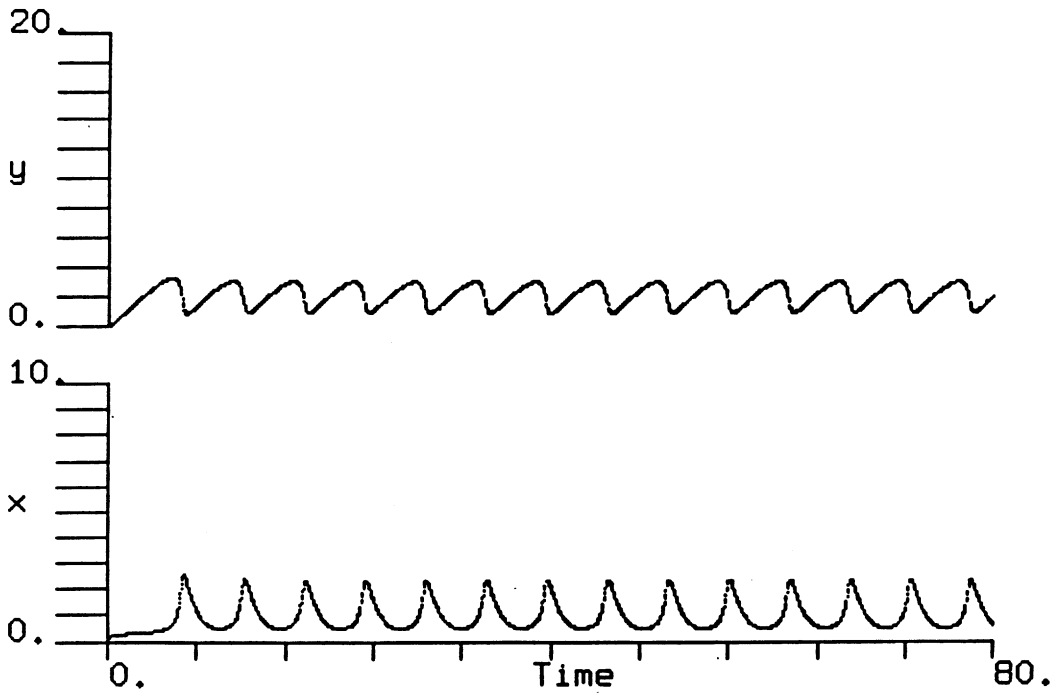


Figure 7.7: Brusselator ($k3 = 1.5$ $k4 = 1.0$)

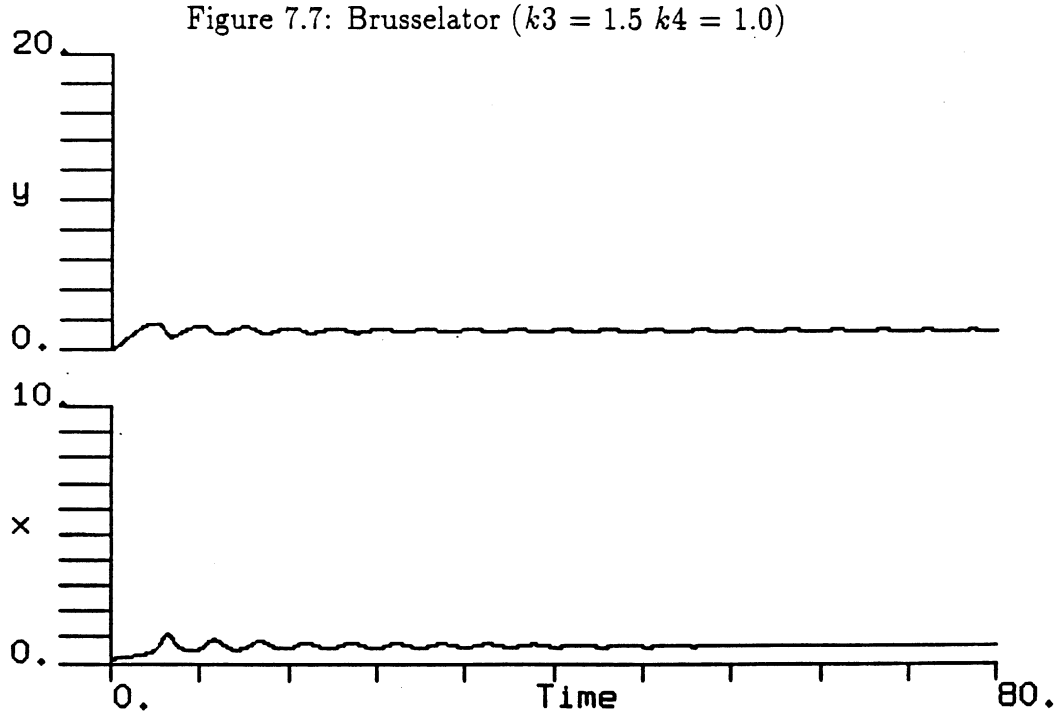


Figure 7.8: Brusselator ($k3 = 3.5$, $k4 = 1.5$)

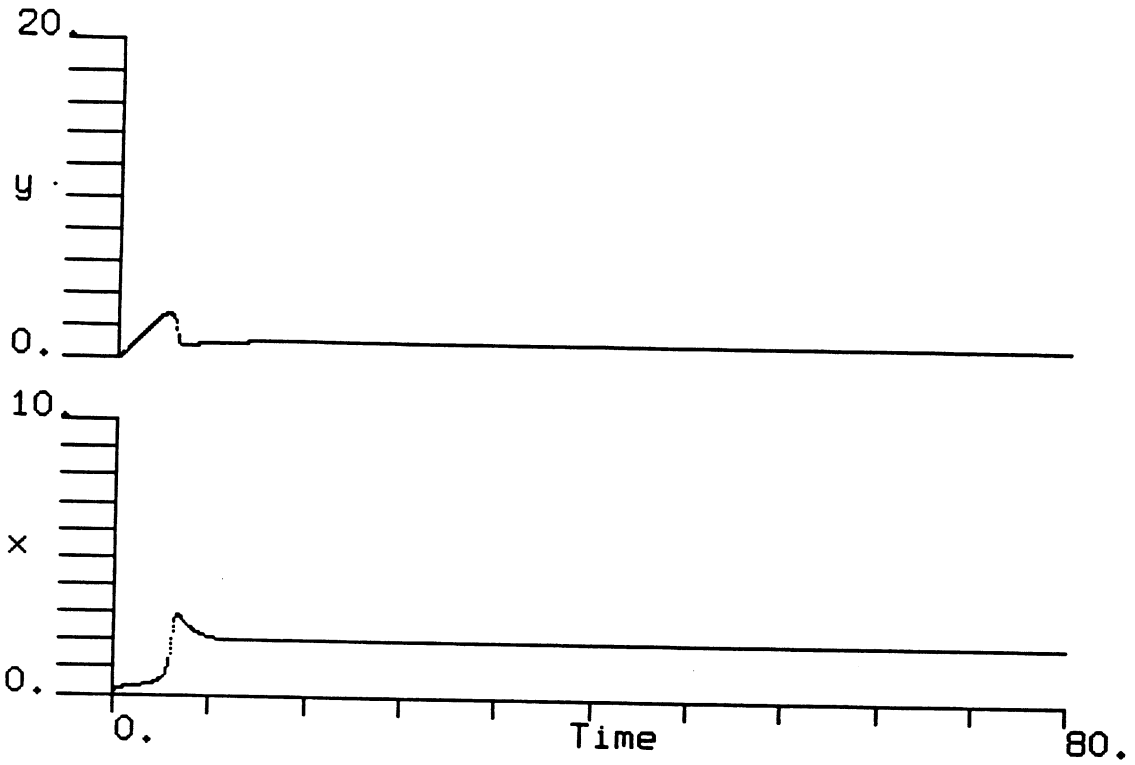


Figure 7.9: Brusselator ($k3 = 1.5$, $k4 = 0.5$)

```
(graph-parameter-scan-files
 "/mydata/brussruns/qualanal-"
 0 63 (make-retrieve-two-rate-constants-and-feature-summary 2 3)
 (make-selective-two-rate-constants-feature-summary-graphing-proc
  'k3 'k4 0 8 0 5 (lambda (k3 k4) true)))
```

When this expression is evaluated, the Workbench reads each of the 64 saved qualitative-analysis files in sequence, and uses the values of $k3$ and $k4$ to obtain a graph location at which it prints out a representation of the feature summary for this file. The parameter space graph created is shown in Figure 7.10: the feature summary for both X and Y is shown at each appropriate location ($k3$, $k4$). The string "sO" indicates that a stable oscillation annotation has been found in the feature summary; "SS" indicates a steady-state; "dO" indicates a damped oscillation that is still at a measurable level at the simulation's conclusion; "dS" is a damped oscillation that is too small to measure by the end of the simulation; and " S" indicates a run that has some sort of oscillating pattern early on, but that appears to end in a steady state.

The boundary between stable and damped oscillations is clearly visible

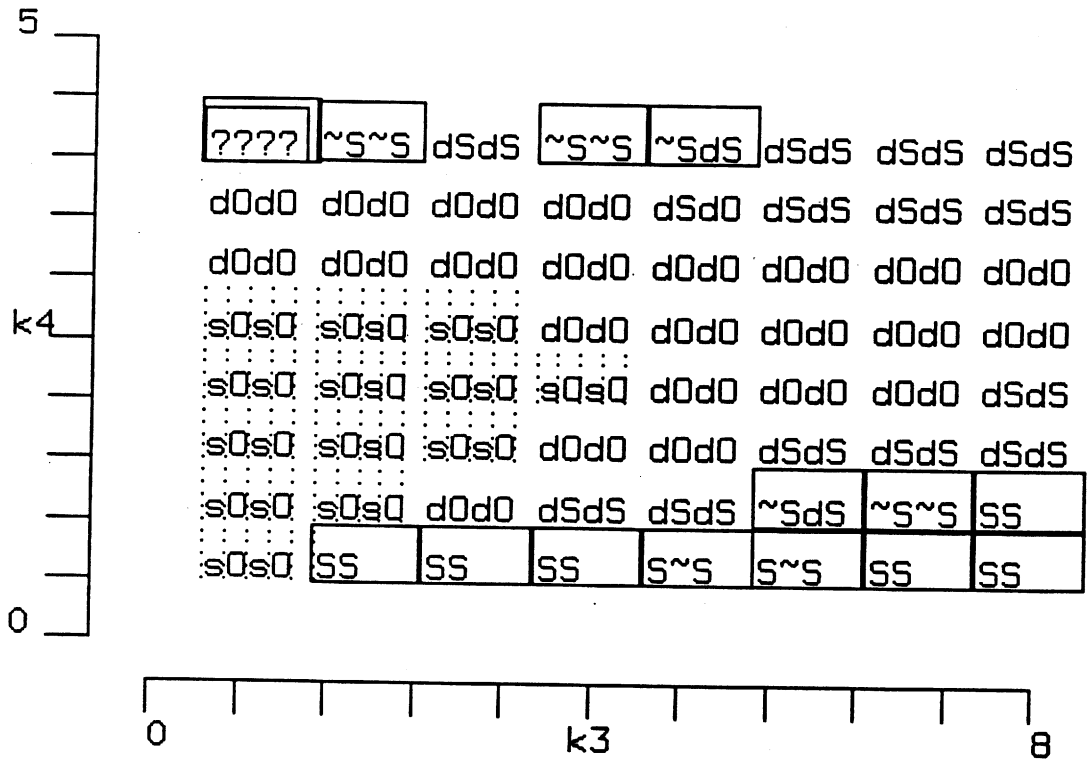


Figure 7.10: Brusselator Parameter Space Graph
In the original Workbench output, the string sOsO is shown in red; strings SS,S~S, ~S~S, and ~SdS are in green; dOdO, dSdO, and dSdS are in dark blue; and ???? is in white. These colors are represented here by stippling, box-outline, normal-font, and double-outlined entries, respectively.

in Figure 7.10, and appears with a somewhat parabolic shape toward the left of the graph. Steady states are identified toward the bottom of the graph and at the upper left; the entry in the upper left corner at ($k3 = 0.5$, $k4 = 4.0$) indicates that this run could not be interpreted by the Workbench. It is illustrative for the reader to compare Figures 7.7-7.9, the results of several simulations, with their summary string in Figure 7.10.

The parameter space graph 7.10 can also be compared with theoretical results obtained by solving for the Brusselator steady-state point and linearizing the differential equations around this point. It is not hard to show that for our system, the condition for steady-state stability is just:

[5] $\quad k4 + (k3/k4^2) > 3$

The boundary between stable oscillations and steady-states or damped oscillations is in agreement with this: that is, stable oscillations are observed precisely when condition [5] is not met. The Workbench is a bit less successful in distinguishing between true (theoretical) damped oscillations and a steady state. The condition for damped oscillations in this system is:

[6] $\quad ((k4 + (k3/k4^2) - 3)^2 - (4\ k3/k4)) > 0$

We should expect to see damped oscillations when conditions [6] and [5] are both met; if only [5] is met, a steady state (without transient oscillations) is predicted. As it happens, whenever the Workbench has noted a damped oscillation for either `X` or `Y`, these statements hold; so the program registers no "false positives." On the other hand, there are "unsure" identifications of steady states at three locations—corresponding to the points (1.5, 4.0), (3.5, 4.0), and (6.5, 1.0)—that in fact are theoretical damped-oscillation points.[6]

[6]Of course, in fairness to the program, even these points are noted as "unsure" in identification; and it should not be surprising that there is a "gray area" between identifying (heavily) damped oscillations and a steady state.

7.5 Oregonator

The Oregonator is a mechanism originally devised by Field and Noyes as a rough model for the Belousov-Zhabotinskii reaction. A number of variants of the Oregonator exist; the one used here is described in Nicolis and Prigogine {61} and Troy {77}. We begin by presenting the mechanism:

```
[5.1]  A + Y  ---> X          k1 = 1.34

[5.2]  X + Y  ---> P          k2 = 1.6e9

[5.3]  B + X  ---> 2X + Z     k3 = 8e3

[5.4]     2X  ---> Q          k4 = 4e7

[5.5]      Z  ---> fY         k5
```

Here, the stoichiometric coefficient f and the rate constant $k5$ are treated as parameters to the mechanism. A typical experimental range for f is from about 0.5 to 2.0, while the range of $k5$ is from 0 to to about 500.[7] In addition, the concentrations of A and B are treated as constant at 0.06, and P and Q are "driven off" species.

As in the case of the Brusselator, we create a doubly-parametrized mechanism, and simulate it repeatedly using the Gear algorithm. (The various data structures are analogous to those used for the Brusselator example, and so will be omitted here for conciseness; the only added feature worth noting is that we use Y alone as a focus species, and use steps 5.1 and 5.5 alone as determiners of episode boundaries.) A selection of sample graphs taken from the sequence of runs is shown in Figures 7.11-7.13.

The Workbench stores qualitative-analysis files, much as in the previous example; excerpts from several of these, corresponding to the runs shown in

[7] Although the notion of a "parametrized stoichiometric coefficient" may seem strange at first, one can view it as shorthand for a class of mechanisms written in standard form. For instance, the two steps Z --> Y and Z --> 2Y, both with rate constant $k5/2$, may be summarized by the single step Z --> 1.5 Y with rate constant $k5$. The equivalence may be seen immediately by comparing the differential equations generated by both alternative notations.

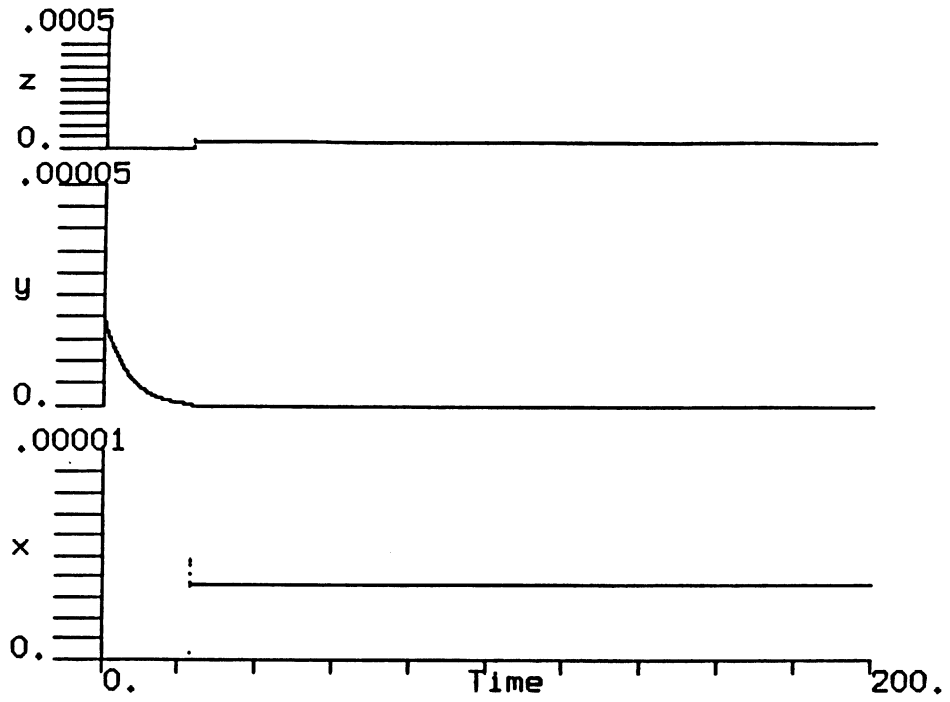


Figure 7.11: Oregonator ($k5 = 60$, $f = 0.4$)

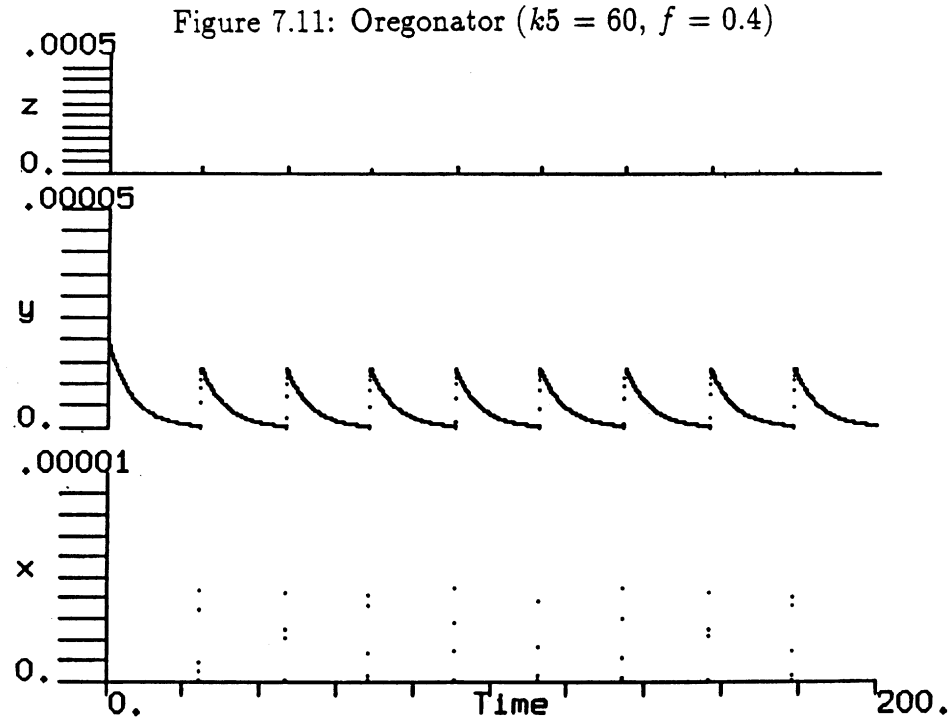


Figure 7.12: Oregonator ($k5 = 60$, $f = 0.75$)

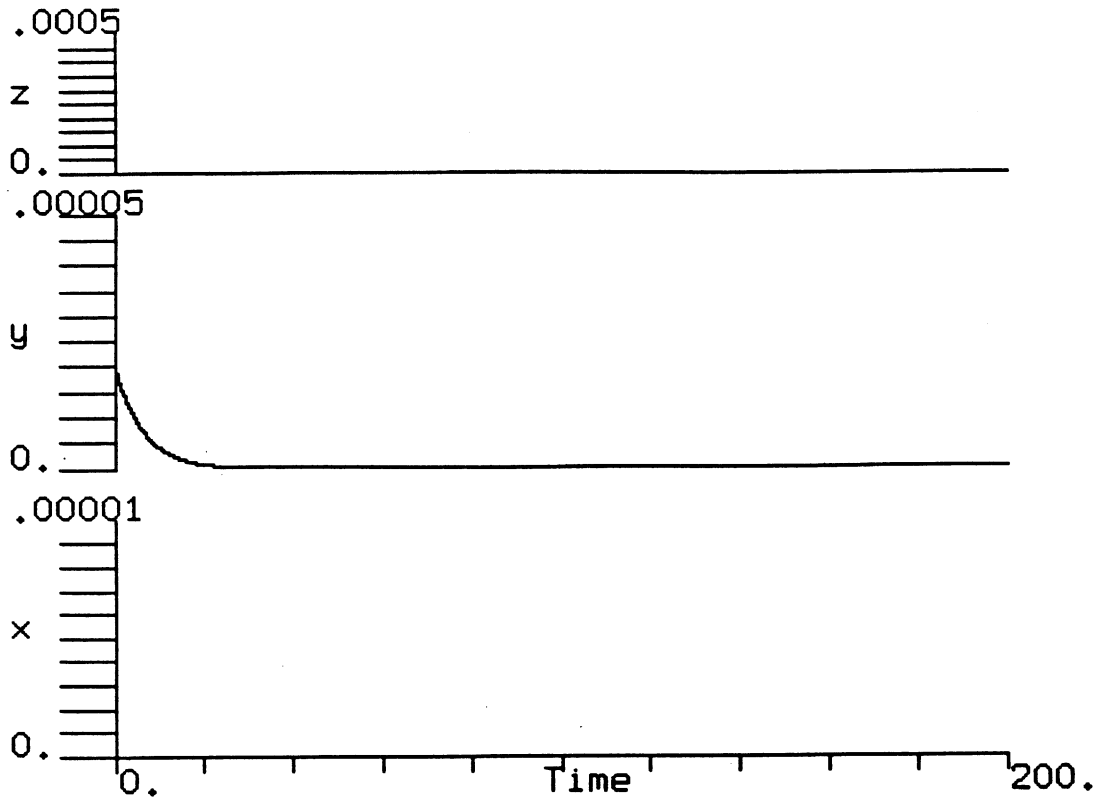


Figure 7.13: Oregonator ($k5 = 60$, $f = 1.1$)

Figures 7.11-7.13, are shown here:

Excerpts for $f = 0.4$, $k5 = 60$.

```
;Feature analysis
((short) (large-decrease y) (rapid-decrease y) (steady-decrease y))
((long) (final) (large-decrease y) (rapid-decrease y))

;Feature Summaries
(*feature-summaries* 1)
(y (probable-steady-state nonzero y) ()
   (multiple-oscillations y) ((uninterpretable) (probably-noise)) () ())
```

Excerpts for $f = 0.75$, $k5 = 60$.

```
;Feature analysis
((long) (large-decrease y) (rapid-decrease y) (steady-decrease y))
((short) (large-increase y) (rapid-increase y))
((long) (large-decrease y) (rapid-decrease y) (steady-decrease y))
((short) (large-increase y) (rapid-increase y))
((long) (large-decrease y) (rapid-decrease y) (steady-decrease y))
((short) (large-increase y) (rapid-increase y))
```

5 more 2-episode pairs

```
((long) (large-decrease y) (rapid-decrease y) (steady-decrease y))
((short) (final) (large-decrease y) (rapid-decrease y) (steady-decrease y))
```

```
;Feature Summaries
(*feature-summaries* 1)
(y () (stable-oscillation y) () ((stable)) ((stable)) ())
```

Excerpts for $f = 1.1$, $k = 60$.

```
;Feature analysis
((short) (large-decrease y) (rapid-decrease y) (steady-decrease y))
((long) (final) (large-decrease y) (rapid-decrease y))

;Feature Summaries
(y (probable-steady-state nonzero y) ()
   (multiple-oscillations y)
   ((probably-noise) (probably-noise) (probably-noise)) () ())
```

And, as before, we can retrieve the stored qualitative summary files and use these to generate a parameter space graph summarizing the Oregonator's behavior. In Figure 7.14, we see the two parameters f and $k5$ used as axes for the graph; only stable oscillations or (probable) steady states have been identified for Y in this graph.

It is illustrative to compare the Workbench's results with theoretical considerations summarized by Figure 7.15, taken from Nicolis and Prigogine {61}. The shaded region in Figure 7.15 indicates a parameter space region in which the lone positive steady-state solution for the Oregonator is unstable; outside this region, the steady-state is locally stable.[8] In fact, the parameter values at which the Workbench has found stable oscillations correspond to points within the shaded region of Figure 7.15.

[8]In the unshaded region, the steady-state solution is not always globally stable, however; at some parameter values in this region there is also a stable limit cycle.

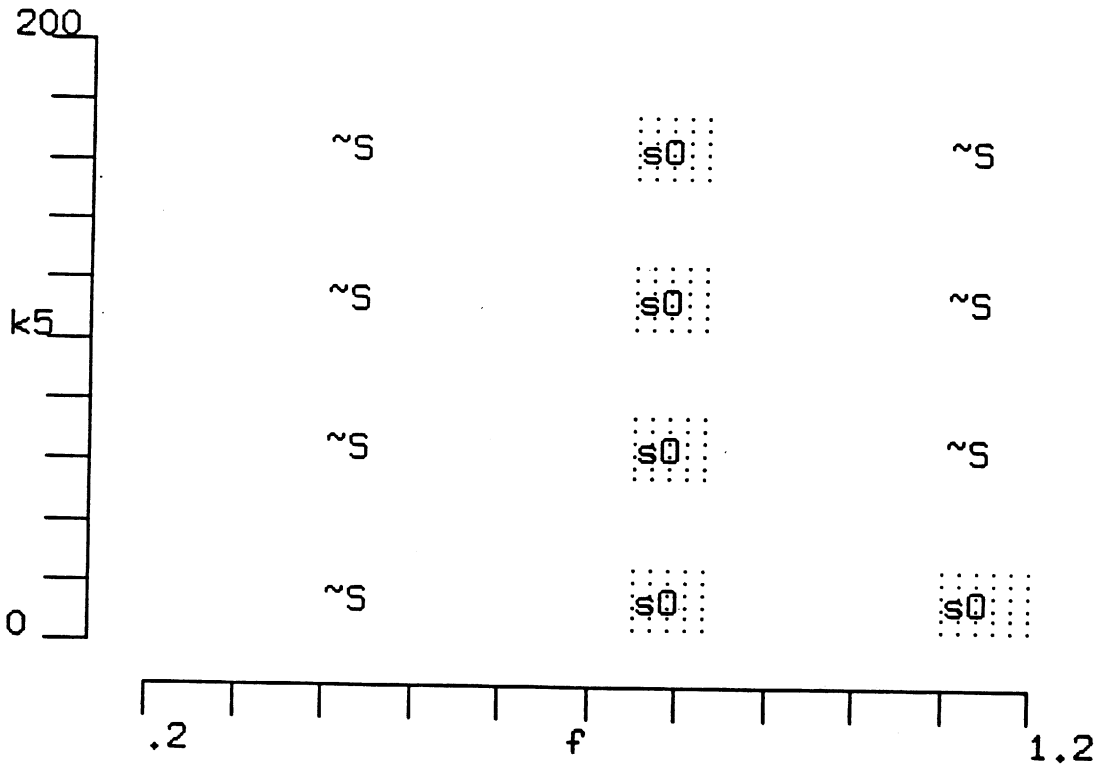


Figure 7.14: Oregonator Parameter Space Graph
Stable oscillations s0 are in red in the original Workbench output, and probable steady states ~S are in green. Here, these are represented by "stippled" and unmarked regions, respectively.

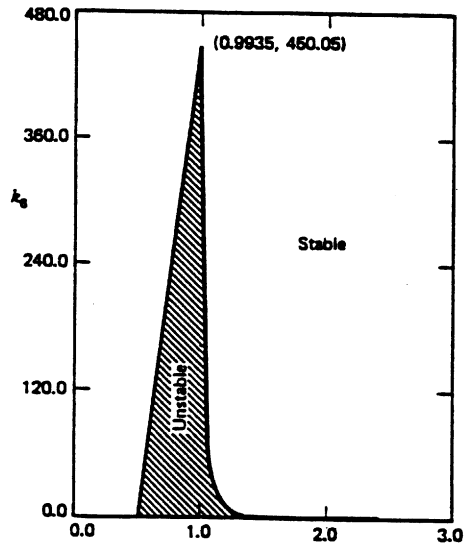


Figure 13.4. Linear stability diagram for Oregonator for $q = 8.375 \times 10^{-6}$, $s = 77.27$, $w = 0.1610\,k_0$.

Figure 7.15: Oregonator Steady-State Stability (from {61})

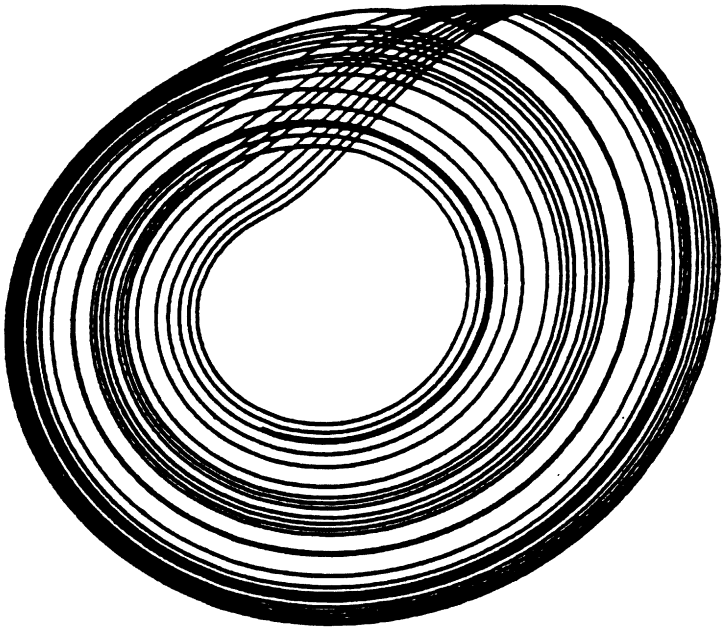


Figure 7.16: The Rössler Band (from {4})

7.6 Rössler Band System

In the previous examples, the behavior noted by the Workbench included oscillations (both stable and damped), and steady states. Just to see that the Workbench can in fact recognize instances of behavior besides these, we present a "chemical translation" of a classical chaotic system: the "Rössler Band."[9] The Rössler Band is described in several texts on chaos {4, 76}; the "band" is a fractal attractor embedded in three dimensions, and its behavior derives from a saddle point with one stable eigenvector and two eigenvectors with complex eigenvalues forming an unstable focus. A portrait of the band is shown in Figure 7.16; and the chemical mechanism that we will use to generate this attractor is shown below.[10]

[9]In developing our mechanism, we follow the treatment of Samardzija {70}.

[10]Our mechanism employs a step with four reactant molecules, and is thus even more "unchemical" than the Brusselator; but the Workbench is capable of handling any mechanism that obeys the formalism outlined in Chapter 2.

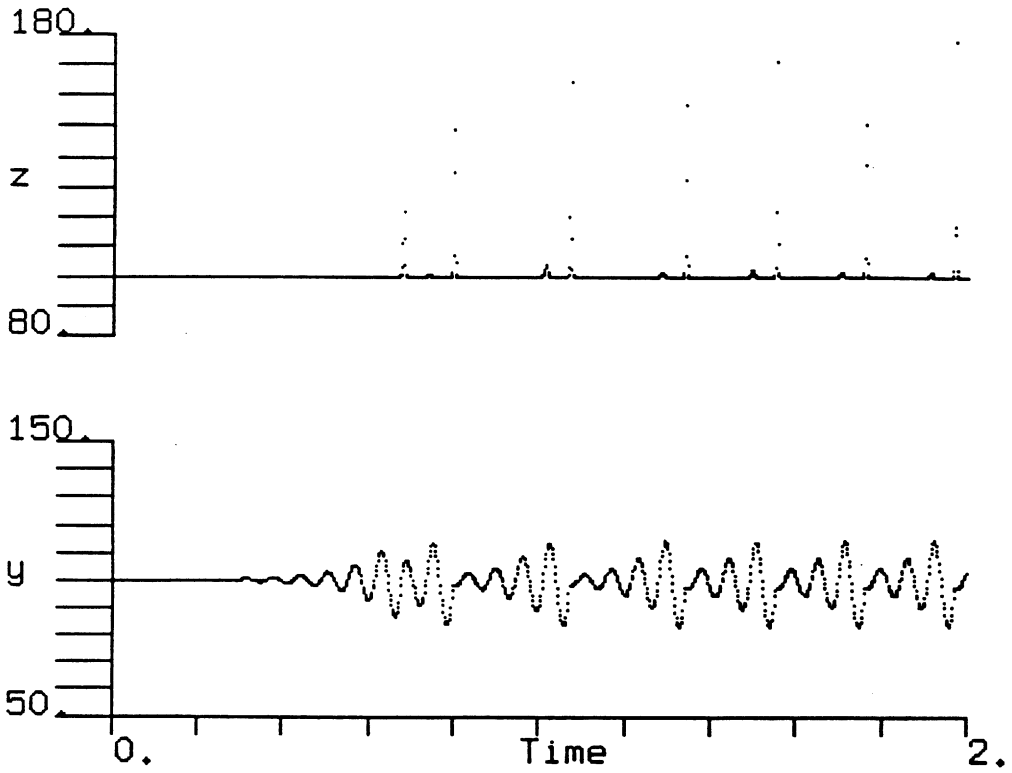


Figure 7.17: Simulation of Rössler System

```
[6.1]  A1 + X + 2Z  -->  X + 3Z   k1 = 1.
[6.2]       X +  Y  -->  2Y       k2 = 1.
[6.3]  A2 + X +  Z  -->  Z + P1   k3 = 1.
[6.4]  A3 + X +  Z  -->  X + P2   k4 = 100.
[6.5]      A4 + 2Y  -->  3Y       k5 = 0.2
[6.6]           2Z  -->  P3       k6 = 55.
[6.7]      A5 +  X  -->  2X       k7 = 200.
[6.8]            Y  -->  P4       k8 = 120.
[6.9]      A6 +  Z  -->  2Z       k9 = 11000.2
```

The concentrations of species A1, A2, A3, A4, A5, and A6 are all constant at 1, and species P1, P2, P3, and P4 are driven off.

Simulating this mechanism for two seconds (using the Gear integrator) produces the graph shown in Figure 7.17. The graph for species Y shows a pattern of repeating "unstable" bursts, punctuated by larger bursts for Z. The Workbench generates an episode history with little in the way of repeating patterns (not surprisingly), and produces the following feature summary:

```
((y () (possible-chaotic-oscillation y)
    (multiple-oscillations y)
    ((possibly-chaotic) (uninterpretable) (uninterpretable)
     (probably-noise) (uninterpretable) (probably-noise)
     (uninterpretable) (probably-noise) (uninterpretable))
      similar interpretations of "grainy" patterns)
 (z () ()
    (multiple-oscillations z)
    ((probably-noise) (possibly-chaotic) (probably-noise)
     (uninterpretable) (probably-noise) (probably-noise)
     (probably-noise) (uninterpretable) (probably-noise)
     (probably-noise) (uninterpretable))
    ((uninterpretable)) ()))
```

The Workbench has classified the pattern for Y as possibly chaotic. It does so more or less in the absence of any other coherent intepretation; having found no other explanation for the data, the program is able to link the various individual oscillations (whose separate interpretations are shown in the feature summary for Y) into one larger, non-periodic structure.[11] The interpretation for Z is more difficult, and the Workbench has no overall classification; it spots multiple oscillations (one of which is deemed possibly chaotic), but these occur too far apart for the program to classify them as part of a continuing behavior pattern. Conceivably, the program could have a richer vocabulary for patterns such as that evidenced by Z (such as "intermittent bursts"); we will return to this topic in the final chapter.

7.7 Chlorite-Iodide Oscillator

In this final example, we will illustrate how the Workbench's numeric files and qualitative-analysis data structures may be used by additional Scheme programs. The mechanism that we will simulate is a chlorite-iodide oscillator described in Citri and Epstein {17}:

[11]The rationale for attempting this "linkage" among oscillations is that they appear close together and have similar characteristics (for instance, they have similar apparent "periods").

```
[7.1]  H+ + HClO2 + I-  --> HOCl + HOI        k1 = 500

[7.2]    HOI + I- + H+  --> I2                k2 = 4.17e11

[7.3]              I2   --> HOI + I- + H+     k3 = 7.413

[7.4]      HClO2 + HOI  --> HIO2 + HOCl       k4 = 6.0e7

[7.5]        HOCl + I-  --> HOI + Cl-         k5 = 1.4e8

[7.6]   HIO2 + I- + H+  --> 2HOI              k6 = 1.0e6

[7.7]            2HOI   --> H+ + I- + HIO2    k7 = 25.

[7.8]          2HIO2    --> HOI + IO3- + H+   k8 = 3000.

[7.9]       HIO2 + HOI  --> IO3- + I- + 2H+   k9 = 230.

[7.10]      HIO2 + HOCl --> IO3- + Cl- + 2H+  k10 = 1000.
```

The concentration of H+ is constant at 0.01; there are sources for both I- and HClO2, and sinks for all species.[12]

We are going to simulate this mechanism for 40 seconds; but just out of interest, we can precede our simulation with graphical analysis of the mechanism. Not unexpectedly, the mechanism has a deficiency greater than zero:

```
Number of complexes: 23
Number of linkage classes: 7
Dimension of stoichiometric subspace: 8
Deficiency: 8
Reversibility: none
```

so none of the "obvious" theorems apply. The Workbench proceeds to look for catalytic and autocatalytic pathways, and its findings for the latter are shown here:

[12]Specifically, the source and sink flow rate terms are all 0.098, and the source concentrations of I- and HClO2 are 0.004 and 0.0025, respectively.

```
Autocatalytic pathways:
(((((hclo2) (hoi)) ((hio2) (hocl)) 60000000.)
  (((hio2) (i-)) ((hoi 2)) 10000.)))
(((((hio2) (i-)) ((hoi 2)) 10000.)
  (((hclo2) (hoi)) ((hio2) (hocl)) 60000000. unknown)))
```

The Workbench has found two two-step autocatalytic pathways in the mechanism. The first is a combination of steps [7.4] and [7.6], which involves autocatalytic production of HOI (assuming that HClO2 and I- are temporarily in excess):

```
[7.4]      HClO2 + HOI  --> HIO2 + HOCl       k4 = 6.0e7
[7.6]   HIO2 + I- + H+  --> 2HOI              k6 = 1.0e6
```

The second is, in fact, the same pair of reactions—but written in opposite order. In this order, the autocatalysis for HIO2 is visible (again, HClO2 and I- must be at sufficiently sizable concentration):

```
[7.6]   HIO2 + I- + H+  --> 2HOI              k6 = 1.0e6
[7.4]      HClO2 + HOI  --> HIO2 + HOCl       k4 = 6.0e7
```

The only other noteworthy aspect of the Workbench's analysis is that it finds a possible steady-state candidate in I-, based on the wide variation in rate constants for reactions producing and consuming this species. As it happens, I- exhibits oscillations during the run, and it also exhibits a period of time during which a steady-state approximation *is* appropriate. We can see this by viewing Figure 7.18, the graph produced by simulating [7.1]-[7.10] for 40 seconds. Here, we see I- graphed at two different resolutions; the top graph is a hundred-fold expansion of the other's lowest one percent. Viewed at "coarse grain," in the bottom graph, I- appears to drop off toward zero concentration. A closer look, in the top graph, reveals that I- is in fact oscillating after about 21 seconds; but indeed, the period between 12 and 21 seconds fits the steady-state approximation for I-.

Once the simulation is complete, the Workbench performs qualitative analysis on the numerical results.[13] Excerpts from the Workbench's analysis are shown below:

[13]We have used I- as the only focus species for this run.

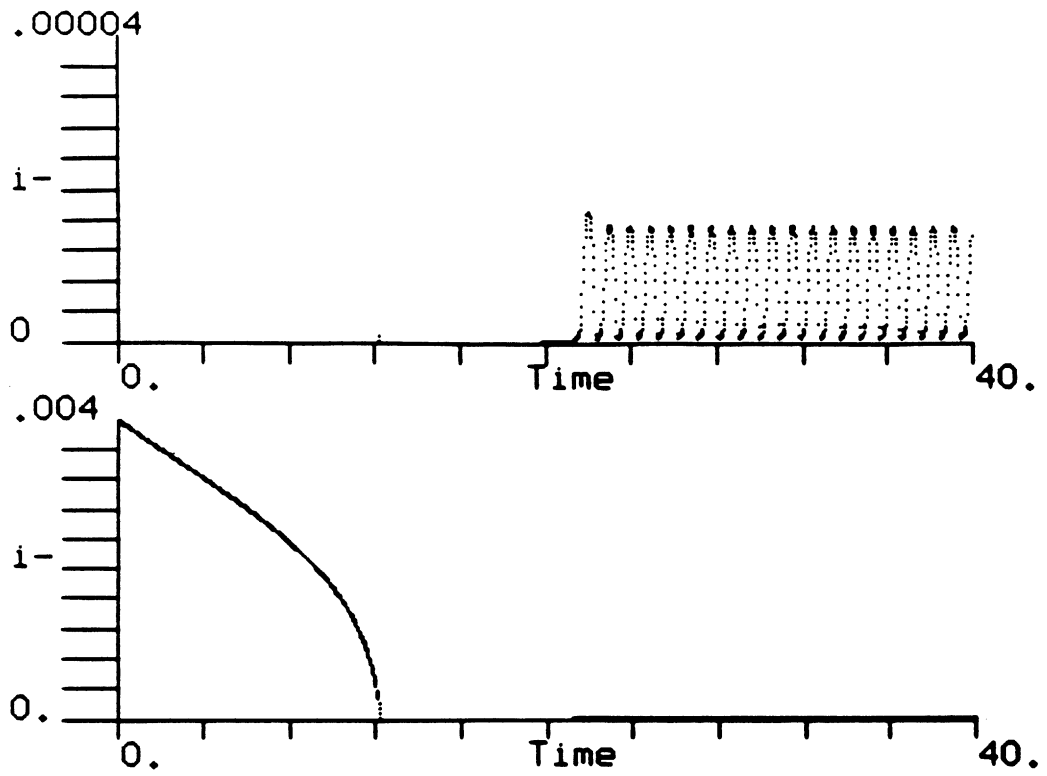


Figure 7.18: Chlorite-Iodide Oscillator Simulation

```
There are apparent oscillations for species: i-
Only one oscillation structure was found.

The oscillations for i- appear to be stable.
The oscillations appear to begin at time: 22.025
with average amplitude 1.4533946076111112e-5
and average period .9513890000000002.
The typical features of an oscillation for i- are shown below:
Oscillation beginning at time: 37.25
Features:
     ((short) (large-decrease i-) (rapid-decrease i-) (steady-decrease i-))
      Episode begins at time: 37.525002
     Important step(s):
          (((i2)) ((hoi) (i-) (h+)) 7.413)

     ((wide-swing i-))
      Episode begins at time: 37.549999
     Important step(s):
          (((hoi) (i-) (h+)) ((i2)) 417000000000.)


At the end of the run, oscillations were still apparent.
```

```
The apparent final state of the system:
    ((i- () (stable-oscillation i-) () ((stable)) () ()))

CHECKING PREDICTION:
There are steady-state possibilities to check.

PREDICTION MET:
(i-) all appear as if they may have long steady states.
```

The Workbench has identified stable oscillations for I-, and provides a "typical episode sequence" (here, consisting of two episodes) for these oscillations. The program then goes on to check its steady-state prediction, and finds that the prediction is met. Although it would be nice to report that the Workbench has identified the period from 12 to 21 seconds as the "steady-state period" for species I-, in point of fact closer examination shows that the Workbench has identified the entire period from 12.2 onward as a reasonable match for this approximation. (In effect, then, the Workbench has used the "coarse-grained view" to sustain this prediction.)

So far, our example is a challenging one, but we have seen no new uses for the Workbench. However, it is worth pausing to consider Figure 7.18 once more, and to imagine what kinds of numerical analysis we might want to perform on this record. A reasonable idea is to study the oscillations of I- by looking at a power spectrum, as mentioned in the previous chapter; but we have to be cautious in doing so. Using an FFT algorithm on the entire numerical record of Figure 7.18 isn't sensible—the initial twelve seconds muddy the spectrum of the concluding oscillations. Instead, we want to perform an FFT only on some integral number of complete oscillations chosen from the end of the run.

Identifying a reasonable "window" of results to pass to an FFT algorithm is—like identifying the presence of oscillations—a task typically left to the chemist. But because the Workbench has produced qualitative data structures that can be used to find both the onset and known conclusion of oscillations, we can now write a program that retrieves only a reasonable chunk of the numerical results, and then produces a power spectrum for this chunk only. We evaluate the following expression:

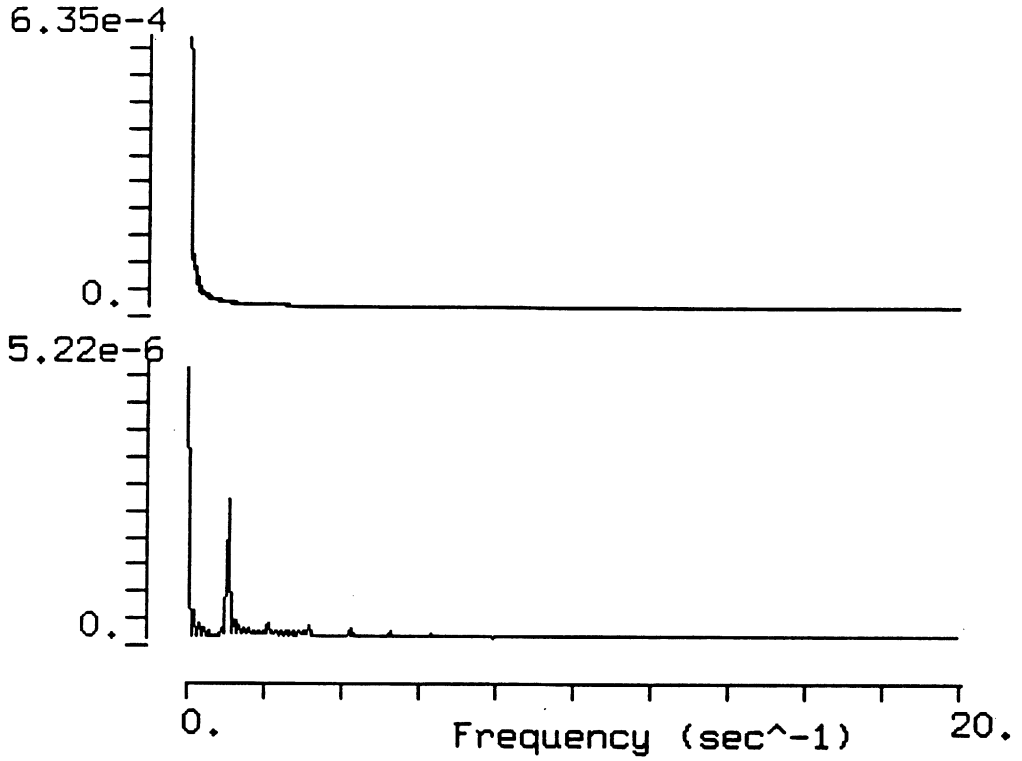


Figure 7.19: Two Power Spectra of I- Results

```
(define *iodide-spectrum*
  (get-power-spectrum-of-last-oscillating-region 'i-))
```

and thus create a power spectrum for only a particular time-period[14] of our results. We have graphed this spectrum at the bottom of Figure 7.19; the horizontal axis corresponds to frequencies between 0 and 20 sec^{-1}. The large peak (besides the "DC component" at far left) corresponds to a frequency of 1.06 (close to the Workbench's identified value of 1.05). Interestingly, additional harmonic peaks are also visible.

The upper graph in Figure 7.19 is the result of calling an FFT algorithm on the entire numerical record of Figure 7.18. (The y-range for the upper graph is much larger than that for the finer-grained graph below it.) As the figure shows, the structure visible in the power spectrum of the lower graph has been lost in the Fourier transform of our entire record. Once again, the central point is not that such a windowing of data is useful (obviously it is), nor that it is especially difficult (we could perform this task by eye), but rather that the Workbench has produced data structures that allow a *program* to perform this task with relative ease. We could now imagine adding

[14]Specifically, the period from 22.025 until 39.15.

such a post-analysis procedure to our parameter-space searches, such that any simulation producing an interpretation of "stable oscillations" could be followed by computation of a power spectrum and again, all of this "dirty" additional processing can be done automatically, without the intervention of the chemist.

Chapter 8

Related and Future Work

The previous chapters have been devoted to a close-up description of the Kineticist's Workbench—how it has been designed and implemented, and how it can be used. In this final chapter, we take a step back and examine the Workbench in relation to other work and to its own goals. We begin by looking at related work in the fields of chemical simulation, qualitative physics, and mixed symbolic/numerical computation. This summary of related work provides background for the discussion that follows: a portrait of the role that qualitative reasoning may eventually play in the creation of truly useful scientific tools. We then explore the myriad flaws in the Workbench implementation, and how these may be addressed by subsequent work; and finally, we conclude with a post mortem critique of the Workbench design.

8.1 Related Work

8.1.1 Chemistry Simulators

As noted in Chapter 2, there are numerous well-developed systems for performing numerical simulation of chemical mechanisms. One program, Acuchem {12}, runs on PCs and contains most of the numerical features of the Workbench: it allows the user to specify reactions and rate constants in a symbolic input file, and then integrates the appropriate system of ODEs using either a Runge-Kutta or Gear integrator, graphing the results on the display screen. Unlike the Workbench, Acuchem does not allow graphing of arbitrary functions of state; it does not allow termolecular steps (though these can be effectively created via concatenating simpler steps); it does not include parametrized mechanisms; it does not include functions-of-time species (which the Workbench can use with its Runge-Kutta integrator); and its graphics routines are less flexible and do not employ color graphics. Even

so, the core of the Workbench's numerical module is captured by Acuchem (no doubt with greater speed), and the two programs share some limitations in that they both model homogeneous, isothermal systems exclusively.

Finzel and Moore {37} describe a kinetics simulation program for a DEC-10 computer; their program is similar to Acuchem, and has greater flexibility for graphing (though less than the Workbench). However, it is not clear whether the Finzel-Moore program allows zeroth-order reactions (as does Acuchem), and it apparently does not contain the additional special features mentioned in the previous paragraph. Shacham and Cutlip {73} discuss a program which, like Finzel and Moore's, is designed mainly for pedagogical purposes. The Shacham-Cutlip program is tailored for the PLATO educational system; it requires differential equations (rather than mechanisms) as input, and uses only a Runge-Kutta integrator (either explicit or implicit). On the positive side, unlike the other programs described so far, the Shacham-Cutlip program can solve systems of linear and nonlinear equations (the Workbench can solve some systems of equations, but only in the context of its graphical analysis of mechanisms). The system introduced by Edelson {23} is also similar to Finzel and Moore's; the added feature here is an ability to check reactions for atomic balance, and to check simulation results for certain types of conservation laws.[1]

The program described by Byrne {13} is a numerical simulator more powerful than the Workbench or the other programs described above. His EPISODE program allows a much richer form for differential equations than that assumed by the Workbench, and is aimed at simulating somewhat more complex chemical systems (such as atmospheric systems with daily variations in temperature). Another sophisticated program, described by Bader *et al* {5}, is a large-scale professional simulator for ODEs derived from chemical systems; while Hindmarsh {46} reviews a collection of numerical routines for solving and integrating ODE systems.

Many more examples could be cited, but these should provide a representative sampling of chemical simulators. In every case, these systems perform exclusively numerical computations (or perhaps near-exclusively: a few in-

[1]It is unclear from Edelson's paper whether his system actively disallows "abstract" mechanisms such as the Brusselator, in which reactions do not balance atomically.

clude algebraic techniques for solving systems of equations, and Edelson's program does atomic balancing). The consequences of this limitation were first discussed in Chapter 2, and we will return to this topic in the next section.

Graphical heuristics for chemical systems have been incorporated into a powerful computer program written in APL by Clarke {18}. Clarke's program includes some techniques not contained in the Workbench (though the reverse is also true). Clarke's program also uses notions of "fast" and "slow" species (rather than reactions; "fast" species may be thought of as a generalization of "steady-state" species), and allows the user to specify sets of fast species interactively. There is no indication in Clarke's paper that his program is linked to a numerical simulation module; nor does the program use rate constant information to assist the user, e.g., by locating global equilibrium states numerically. On the other hand, Clarke's program is capable of analyzing certain complex mechanisms (such as the Oregonator) that are beyond the current capabilities of the Workbench.

8.1.2 Qualitative Physics

Many of the techniques incorporated in the Workbench have analogues—often more completely developed analogues—in the field of "qualitative physics." This is a broad term that encompasses a wide range of research; but generally (to venture a one-sentence summary), qualitative physicists seek to develop intuitive non-mathematical formalisms for explaining and simulating physical phenomena. In pursuing this goal, they face some of the same issues as the Workbench: how to summarize physical events in meaningful ways, how to represent phenomena that take place on vastly different time scales, how to make reasonable predictions of a system's behavior from its structure.

The Workbench uses the episode data structure (described in Chapter 6) as its primary method for "descriptive chunking" of simulation results. These "episodes" are reminiscent of the notion described by the same term in de Kleer and Brown's {22} seminal paper in qualitative physics. De Kleer and Brown's emphasis is on the modelling of multicomponent physical devices, and an "episode" for them is a period of time during which all the device's individual components (such as valves, springs, etc.) have unchanging states:

for instance, an episode might consist of the time during which a valve is open and a spring exerting a positive upward force. In this sense, the de Kleer and Brown "episode" is like the Workbench's (both are based upon the notion of "temporarily unchanging state descriptions"); and in each case, the episode can be the basis of a historical summary of some process.

There are important differences between the two notions, however. De Kleer and Brown construct their episodes not by analyzing numerical results, but rather by "qualitative simulation"; first they construct a formal qualitative description of a device (including rules that link the behavior of the components), and then they simulate the device via those rules ("confluences," in their terminology). There is a great deal more detail that could be supplied, but the main point here is that for de Kleer and Brown, episodes represent possible time periods of a qualitatively-simulated device, while for the Workbench they represent periods of time derived from interpreting a numerical record. Again, though, it is worth noting that both episode notions are motivated by the need to find an intuitive "parsing" for physical phenomena.

The same thread of thought can be detected in a number of important qualitative physics papers: Forbus {38} models processes via a series of "process episodes" in which state quantities of various objects of interest remain constant; Williams {80} models circuits via transitions between "operating regions"; de Kleer {21} models simple mechanics scenarios by chunking the events into motions occurring under specific conditions (eg, the motion of a sliding block on a segment of track with a constant slope). Indeed, for anyone concerned with providing meaningful descriptions of physical phenomena—however those phenomena may be simulated—the issue of finding boundary points within the historical record is central. The Workbench's solution to this problem is more prosaic than the other strategies mentioned here, since the Workbench is able to work from both a numerical record and a known, uncontroversial formalism that generates that record. Even for the Workbench, however, the parsing problem is not easy; nor has the problem been completely solved, as can be seen by examining the less-than-useful episode history of nitrogen pentoxide decomposition in Chapter 7.

A second theme that the Workbench shares with qualitative physics is

the representation of phenomena whose "sub-processes" operate on widely varying time scales. The Workbench looks for standard patterns of interest to the kineticist, such as rapid equilibria; these patterns enable the chemist to understand a given complex system as the (constrained) interaction between two simpler subsystems. In the realm of qualitative physics, Kuipers {52} seeks a similar decomposition between systems; as an example, he provides a qualitative simulation of physiological water and sodium balance, in which the two processes interact, but the former process can be viewed as much faster than the latter. In Kuipers' system, unlike the Workbench, the different time scales are not deduced from structural information, but rather must be explicitly provided to the program, which can then use these time-scale specifications to treat slow processes as constant relative to faster ones. Similar issues can be seen in Davis' {20}paper on order-of-magnitude reasoning in qualitative systems; his concerns are more general than just time-scale differences (they also include interactions between, eg, particles of widely varying mass), but include phenomena like a "quickly settling control parameter."

Representing widely varying time-scales leads to considerations of problems involving "qualitative arithmetic," since there is a need to express notions like "quantity x is negligible compared to quantity y." Raiman {65} and Mavrovouniotis and Stephanopoulos {60} describe two formalisms for approximate arithmetic statements of this kind. The former includes basic relational concepts like "is negligible compared to," and develops deductive rules such as "If A is negligible compared to B and B to C, then A is negligible compared to C." Mavrovouniotis and Stephanopoulos have a more extensive and powerful formalism that (unlike Raiman's) combines approximate and exact (numerical) computation. The Workbench in fact must compare expressions that mix symbolic and numerical quantities[2], but does not require anything like the power of Mavrovouniotis and Stephanopoulos' system, since the form of expressions being compared is highly restricted.

Predicting the behavior of a mechanism from its structure is an important aspect of the Workbench's operation; likewise, in qualitative physics, the commonly-encountered notion of "envisionment" may be seen as an attempt to understand the possible alternative behavior patterns of a physical

[2]As described in Chapter 5.

system.{39} The term has been given slightly varying meanings, but typically a qualitative physics program engages in envisionment when it uses known qualitative relationships between state variables to generate all possible internally consistent qualitative behaviors for a given system. A good example of the idea can be seen in de Kleer and Brown's paper, in which the possible behaviors of a pressure regulator are summarized in a state transition diagram; in this case, the envisionment process would be responsible for generating some or all allowable paths through the diagram.

The Workbench's techniques for predicting mechanism behavior are in some cases much weaker than those of qualitative physicists: about complex mechanisms, the program is not able to say very much. On the other hand, techniques such as the Zero Deficiency Theorem are much more powerful than most tools at hand for qualitative reasoning. The Zero Deficiency Theorem, after all, is able to place fairly tight restrictions on the behavior of those mechanisms to which it applies whereas the most commonly encountered problem for qualitative envisioning is an overgeneration of possible behaviors.{54} In some sense, however, the Zero Deficiency Theorem is not really "qualitative"; it cannot be rephrased (in any way that I know of) in purely intuitive terms. The Zero Deficiency Theorem (like other similar techniques) is more in the nature of a "symbolic enhancement" to numerical computation; and we now turn our attention to other work along these lines.

8.1.3 Combining Symbolic and Numerical Computation

It is in fact hard to make a clean distinction between "symbolic" and "qualitative" methods. Programs that employ qualitative reasoning almost inevitably involve symbolic processing of some sort; but the converse is not true. The term "symbolic methods" often applies to techniques involving computer algebra, for instance; automatic deductions using predicate calculus might also fall under this heading. Some techniques, like those involving approximate arithmetic, are harder to classify—and indeed, it may not ultimately be terribly important whether a technique is labelled "qualitative" or "symbolic." For our purposes, a symbolic (as opposed to qualitative) technique might be defined as a primarily mathematical technique that employs symbolic data; the term "qualitative" implies an attempt to represent

intuitive or "naive" concepts computationally.

Regardless of these fine semantic distinctions, some of the techniques employed in the Workbench—the Zero Deficiency Theorem, for instance—are arguably less qualitative than others. In a similar vein, the program by Clarke mentioned earlier seems to fall more on the symbolic end of the spectrum. There are other examples in which computer algebra is used to study the behavior of systems of ODEs: Rand and Keith {66} describe a MACSYMA program that derives an expression for the center manifold of a system with one or more zero eigenvalues (and all others negative). The main example in their paper is not taken from chemistry, but the same types of calculations can be used to study the behavior of nonlinear systems at bifurcation points. Hanusse {45} discusses a symbolic "reaction scheme translator" that looks for classes of bifurcations in ODEs derived from chemical mechanisms; Hanusse's system is also part of a larger simulation package that performs numerical integration.

There are other examples of symbolic/numerical computation that are closer to the Workbench in their focus upon the behavior of parametrized dynamical systems. Gladd and Krall {41}, for instance, describe a program that uses both Lisp and FORTRAN code to perform parameter space searches. In their program, the user notates the results of runs symbolically (by labelling them as "oscillating" or "chaotic" or whatever); the program can then use these labelled parameter scans to suggest parameter step-sizes and initial state values for future runs. Unlike the Workbench, Gladd and Krall's program does not appear to do any interpretation of numerical results itself. Abelson's Bifurcation Interpreter {1} explores the behavior of parametrized nonlinear maps and identifies the types of bifurcations encountered as the (single) parameter changes. The Bifurcation Interpreter, unlike the Workbench, varies parameter step-sizes to investigate regions of bifurcations more accurately. Sacks' Piecewise Linear Reasoner program {68} constructs qualitative descriptions of solutions for differential equations; it approximates nonlinear systems with piecewise linear systems, and checks the approximation with Sacks' own Qualitative Mathematical Reasoning program. The Piecewise Linear Reasoner represents a marvelous welding of symbolic and numerical methods; on the problematic side, it is not clear how Sacks' transition-state formalism for system behavior extends to com-

plex systems with a phase space of three or more dimensions. Finally, Yip's KAM program {82, 83} shares with the Workbench the goal of classifying the results of its numerical simulations—in the case of KAM, the systems being studied are Hamiltonian maps. KAM uses a Euclidean minimal spanning tree algorithm to classify the types of orbits generated by these maps, and uses knowledge of dynamics to locate "expected" orbits that the simulation has not yet produced.

All these programs represent efforts to integrate symbolic and numerical processing for particular classes of problems—Hamiltonian maps (Yip), maps with attractors (Abelson), two-dimensional systems (Sacks)—in much the same spirit as the Workbench focuses on chemical mechanisms. A larger-scale effort, Abelson and Sussman's Dynamicist's Workbench {3}, may be seen as an attempt to create a special-purpose language (embedded in Scheme) in which symbolic and numerical processing can be easily combined. This system is more in line with the concerns of the following section, and we will return to the Dynamicist's Workbench a bit later on.

8.2 Philosophical Reflections I:

Beyond the Purely Numerical in Scientific Computation

The Kineticist's Workbench is an exercise in combining different kinds of programming paradigms—numerical computation, graph algorithms, "approximative" arithmetic, symbolic interpretation of numerical results. None of the individual techniques in the program is particularly arcane; most have been implemented at one time or another in earlier existing programs. What makes the Workbench potentially interesting as a computer science project is the attempt to combine all of these different programming "styles" synergistically, so that they can communicate with and support one another. The Workbench is thus not devoted to studying a particular numerical method or symbolic representation, but is rather a prototype for a complete, coherent tool for scientific computation. Ideally, such a tool should provide the scientist with something approximating an "automated assistant" specializing in simulation.

I do not in fact believe that the Workbench is more than a partial success in this regard. The two sections that follow this one will be devoted to a dissection of various problems and flaws in the program, as well as directions for future work. But before embarking on such a critique, it is worth devoting some reflection to the ideal toward which the Workbench project is directed.

Scientific computation should support the reasoning processes and creative expression of the scientist. This statement alone implies a kind of heterogeneity of methods that need to be incorporated into scientific tools. After all, a scientist confronted with a complex system—a chemical reaction, for example—must think about that system from a variety of viewpoints. He must observe the events of the reaction itself closely, looking for "interesting events." (Are there rapid jumps in concentrations? Are there unexpected patterns in the numerical results? Do the numerical results resemble those of other known reactions?) He must imagine different experimental techniques that could be brought to bear on this system. (Again, there might be analogies with other systems: if an earlier reaction was successfully studied via a relaxation experiment, perhaps this new one could be also.) He must use his knowledge of real-world chemistry, and of other well-studied reactions, to propose a mechanism, and might spend some time tinkering with this mechanism until it looks capable of generating behavior similar to that observed in the laboratory. Then he must simulate the mechanism numerically, scanning parameters, looking for patterns within the numerical results, and explaining those results with reference to the specific reactions in his hypothetical model.

Throughout this process, the scientist reasons "visually," by noticing phenomena in the world and by looking at graphs; "numerically," by performing mathematical tasks—algebraic simplification, comparisons and approximations, numerical integration; "symbolically," by describing certain phenomena with terms like *rapid*, *large*, *oscillating*, *stable*, *autocatalytic*, *intermittent*; "pictorially," by imagining the results of certain experiments or processes; "encyclopedically," by a knowledge of other similar reactions and phenomena. And throughout this process, an appropriately-devised computational tool could provide assistance: by looking for interesting patterns in experimental results from the laboratory and simulations; by performing numerical integration and other mathematical tasks; by labelling results

with symbolic descriptors; by presenting quick pictorial approximations of simulations; by accessing databases of known reaction mechanisms and experimental results.

As mentioned earlier, in Chapter 2, computers are employed only for a few select tasks in this litany. Generally they act as number-crunching devices, and occasionally are called upon to do symbolic algebra or imaginative data presentation. But even among these limited tasks, there is little cooperation: no commonly-used program that I know of can use numerical results to suggest algebraic simplifications (e.g., by noticing that a particular state variable is effectively constant during a simulation). And of course, even if these few typically-automated techniques were exploited much more effectively, there would still be huge areas of potential computer usage left untapped.

Computers are indeed superb number-crunchers, and numerical computation is a relatively advanced art. But because the numerical aspect of computation has advanced so far, and in such isolation, the scientist is often faced with a daunting problem of recent vintage: how to predict, interpret, and manage the reams of data so easily produced by machines. Numerical programs generate data, but offer little or no assistance in understanding that data.

Qualitative physics represents a type of response to this problem. In fact, the previous complaints about purely numerical computation are heard frequently from researchers in qualitative physics. Consider for instance the following passage from de Kleer and Brown {22}:

> The behavior of a physical system can be described by the exact values of its variables (forces, velocities, positions, pressures, etc.) at each time instant. Such a description, although complete, fails to provide much insight into how the system functions. The insightful concepts and distinctions are usually qualitative, but they are embedded within the much more complex framework established by continuous real-valued variables and differential equations.

The complaints are valid; what, then, should be the response of the computer scientist? Much of the effort in qualitative physics has gone toward replacing numerical formalisms with new, qualitative formalisms on the ground

that these are simpler, more intuitive, and more representative of human thought.[3] Indeed, the sentence following the quote above reads,

> Our long-term goal is to develop an alternate physics in which these same concepts are derived from a far simpler, but nevertheless formal, qualitative basis.

The goal of creating a new, simple formalism for physical phenomena is ambitious, and much of the work in this direction has been fascinating and useful. Qualitative formalisms may well be of particular interest in describing systems for which a mathematical treatment is grossly incomplete or unavailable (cf. {40}). My own view, however, is that in developing computational tools for scientists, qualitative reasoning should be used not to *supplant* numerical computation but rather to *enrich* it. Often scientists are indeed interested in obtaining a numerical answer: a steady-state concentration, a power-law relationship between variables, a parameter value at which a bifurcation occurs, a critical concentration at which oscillations begin. Numerical computation is indispensable to this task; but why not support that computation with programs that understand concepts like "steady state," "power-law," "bifurcation," and so on?

This, then, should be one of the important goals of research in qualitative reasoning: to find ways of cooperating with numerical techniques and thereby increasing the expressiveness of scientific computation. Indeed the synergy being proposed here should extend beyond qualitative reasoning and include other types of symbolic (and less openly "intuitive") formalisms, such as algebra, as suggested in the previous section. The benefits of such heterogeneous cooperation would be enormous, and could take a huge variety of forms; but in the interest of providing a few concrete examples, the following paragraphs suggest some particular tasks for which the combination of numerical and qualitative computation would be especially well-suited.

8.2.1 Using Qualitative Interpretation to Guide Numerical Work

In the previous chapter we saw several examples showing how the Work-

[3] Researchers in qualitative physics display different opinions on the question of whether their programs should in fact represent naive or expert cognition. (cf. {40, 69})

bench can be used to perform a parameter-space scan—that is, to vary one or two parameters and classify the behavior of the mechanism as those parameters change. Typically, computers are used to perform such scans iteratively, parameter value by parameter value, without interpretation; the Workbench does this as well but can produce a graph (like Figure 10 in the previous chapter) indicating its interpretation of the mechanism's behavior during each run. This is all well and good, but another issue becomes immediately apparent when looking at these parameter space graphs. We notice that there are some relatively uninteresting regions, in which mechanism behavior does not change much over large variations in parameters, and there are also more intriguing regions in which mechanism behavior changes from one type of behavior (eg, damped oscillations) to another type of behavior (eg, stable oscillations). These latter regions are the ones we would most like to focus our attention on—we would like the parameter step-size to be much finer so that we can get an accurate picture of the boundaries between different behavior patterns. On the other hand, we don't need nearly as much detail in the more uninteresting regions of the parameter space graph.

What we would like then, is a tool that uses its judgment of system behavior to dictate, on the fly, the fineness of the parameter step size. In fact, both KAM and the Bifurcation Interpreter do have this capability, though the Workbench at present does not; the main point, however, is that such capabilities could conceivably be a common attribute of simulation programs. Qualitative interpretation, in other words, is not just a frill or afterthought tacked on to a numerical routine, but rather can be an integral part of the simulation process.

More generally, qualitative interpretation can be used to suggest appropriate numerical routines and parameters.{2} The Workbench exhibits a hint of this talent in its parameter-scanning process: when a given simulation fails to produce recognizable behavior, the Workbench tries one more run of twice the length and half the time-step in the hope that this will produce an unambiguously interpretable behavior pattern. More generally, though, a simulation program could be used to suggest whether, eg, the Gear or Runge-Kutta integrator should be used based on initial "dry runs" (in the case of the Workbench, this decision is left to the sometimes unwilling user); or the program could vary the time between Gear integrator read-outs depending

on how interesting the system behavior is (for instance, long flat concentration values could be summarized by a few data points, while "spikes" of concentration like those seen in the Oregonator runs could be shown at a high level of detail).

It may well be the case that new numerical methods are suggested by the incursion of qualitative interpretation. Consider, for instance the following simple mechanism:

```
[1]      A <---> B ---> C
```

Now, suppose we are told that the equilibrium between A and B is extremely fast relative to the conversion reaction between B and C. If we wish to simulate the mechanism [1] numerically, we have to choose an integration algorithm—and because the rates of the reactions vary so widely in this instance, we are likely to choose a high-powered algorithm like the Gear integrator. But we know that in some sense all this numerical power is unnecessary: the system is essentially a first-order reaction

```
[2] X --> C
```

in which X may be viewed as a kind of "compound species" consisting of a near-constant ratio of A to B concentrations. In other words, we could use a simple integrator to study this mechanism, as long as we can express the notion that the system is constrained to maintain a constant ratio [A]/[B]. Chemists make these approximations on their own, but there is no way of *directly* representing the approximation to the computer, and no way of expressing a "constrained integration" algorithm. These are the sorts of numerical issues that could conceivably come to the fore once qualitative concepts such as "fast equilibrium" find routine computational expression.

8.2.2 Using Qualitative Interpretation to Guide Data Management

In studying chemical mechanisms, it is often of interest to find classes of mechanisms with a particular structural feature or type of behavior. Noyes {63}, for instance, lists a set of structural features (including the presence of an intermediate species shared by two "competing" sets of reactions) common to oscillating mechanisms. We might therefore plausibly expect some chemist

to ask a question like "Which mechanisms studied over the past decade have this kind of intermediate?" To answer this question, the chemist must do a search of the literature; but one can imagine a "mechanism database" in which concepts such as "intermediate species" are directly represented and retrievable. Potentially, such a database could enable the chemist to find (for instance) all chlorite-iodide mechanisms that exhibit relaxation oscillations; or all mechanisms that exhibit a parameter-space graph similar to that of the Oregonator.

More powerfully, such a database could be accessible not only to the chemist but to the computer as well. We might imagine a program that, as it generates a parameter space graph, can retrieve already-existing graphs with similar characteristics; thus a program could report (say) that it has observed behavior similar to that of a certain enzyme reaction. Or the program might automatically compare the behavior of a given mechanism to the recorded behavior of other, structurally similar mechanisms.

8.2.3 Mixing Qualitative and Quantitative Representations

A bit earlier in this discussion, the notion of "constrained numerical integration" was introduced in a speculative way. The idea is that we want to represent some system partly through "standard" methods of differential equations, and partly through a formalism for expressing known constraints. Taking this idea a step further, we might begin to imagine a broader class of simulation strategies that enable us to freely mix differential equation models with elements of a more qualitative nature.

As an example of what this type of simulation might look like, we can consider an example drawn from outside the realm of chemistry: a water-pump. This is the type of engineered, multicomponent device that is often the subject of qualitative explanations; but suppose we wanted to develop a simulation language suitable for predicting the behavior of such a device.[4] Certain portions of the pump are best described by differential equations: a water tank with open inlet and outlet valves, for instance. But other portions

[4]The specific type of pump being referred to here is explained—qualitatively—in Macaulay.{57}

of the pump—valves, for example—might best be modelled as computational elements that obey logical constraints such as "if the pressure on one side of the valve is greater than the pressure on the other, then the state is *open*, otherwise it is *closed*." In other words, those portions of the device that lend themselves to the type of formalism suggested by de Kleer and Brown could be represented appropriately; these are precisely those elements for which accurate numerical representations are unavailable, numerically expensive, or hideously complicated. On the other hand, those portions of the device that lend themselves to straightforward numerical treatment can be modelled that way (rather than, eg, treating a water tank as something that goes through logical states like "empty" "part-full" and "full").

Such speculation is much harder to put into practice, of course, than it is to generate. What is imagined here is a natural computational formalism for mixing differential equations and symbolic elements into a single simulation language.

The first major step toward such a system is discernible in the Dynamicist's Workbench {3}, whose name suggested that of the project described in this paper. The Dynamicist's Workbench is a collection of procedures designed for simulation of dynamical systems; it uses algebraic expressions that can be treated either symbolically (for instance, for the purpose of simplification) or numerically (by evaluating an expression with certain bindings for variables). The program also allows the creation of elements (like resistors) that are expressed in terms of constraints between variables. Probably the most problematic aspect of the Dynamicist's Workbench is its forbidding (Scheme) syntax and lack of user-interface features, a topic to which we will return later.

It should also be mentioned that there is an increasing trend within qualitative physics research toward integration of qualitative reasoning and numerical methods. Kuipers and Berleant {53}, for instance, describe a system in which quantitative values, when known, can be propagated through a qualitative simulator to obtain (or constrain) the values of important "landmark" quantities (such as the maximum level of water in a given tank). Forbus and Falkenhainer {40} have a program that creates differential equation models for distinct operating regions corresponding to distinguishable qualitative

states of a given system. There thus seems to be a growing convergence of interests between researchers in the field of scientific computation (recall, e.g., the Gladd and Krall article mentioned earlier) and those in the field of qualitative reasoning.

8.3 Problems with the Workbench; Future Work

There are numerous problems in the current version of the Workbench. One major issue, the design philosophy behind the Workbench program, will be the focus of the section following this one. Here, we concentrate on those flaws in the Workbench that permit a solution consistent with the current basic design of the program.

8.3.1 Speed and Range of the Program

The first, immediately apparent flaw in the Workbench is in its speed. The program is simply too slow for professionals to use on anything but an experimental basis. Each of the twelve Oregonator runs shown in Chapter 7 took approximately two hours of computer time; thus the overall parameter space scan required over a day. This situation is capable of improvement in several ways; there have, in fact, been a few recent rewrites to increase the program's speed since the Oregonator experiment was performed.[5]

A bit less prosaically, there are a number of "quantitative improvements" that could be made to the Workbench—that is, bigger-and-better versions of techniques already included in the program. Certainly the number and variety of graphical heuristics in the program could well be improved; the earlier-mentioned program by Clarke, for instance, employs some techniques (including the so-called "knot-tree theorem" for determining the possibility of bistability) not included in the Workbench. Schlosser and Feinberg {71} describe some additional graph representations appropriate for exploring questions of bistability, while Feinberg {32, 33} mentions graph theorems

[5]One major improvement will occur when the Workbench is run on newer hardware. Currently the program is run on an HP 350 workstation, but on the HP 720, the program's speed is likely to increase by a factor of at least five.

for some deficiency-one mechanisms. All of these additional techniques could be added to the Workbench in a relatively straightforward manner, as could more elaborate techniques for finding catalytic and autocatalytic pathways in mechanisms.

The range of approximations available to the Workbench could also be increased. Currently, the program has no way of representing approximations that can be derived from information regarding initial conditions. (This, in fact, is why the program was unable to make a steady-state approximation for the enzyme mechanism in Chapter 7; the approximation in this case is based on the assumption that $[S]_0 \gg [E]_0$.) Likewise, there are no approximations available to the program based on assumptions of, eg, linear concentration growth (as opposed to near-constant concentrations); consider, for example, the graph of [P] shown in Figure 5 in Chapter 7. Extensions along these lines might well suggest corollary elaborations to the Workbench's qualitative arithmetic abilities: representing an *a priori* assumption about the concentrations of two species, for instance, implies the need for systematic deduction rules of the kind implemented by Stephanopoulos and Mavrovouniotis. {60}

A larger range of graphing and data-analysis techniques would be helpful to Workbench users. The program does not permit phase-space graphs—e.g., graphing [X] versus [Y] in the Brusselator mechanism rather than concentrations versus time. Other facilities for, say, plotting a planar projection of a three- (or n-)dimensional path in phase space, or for plotting Poincaré sections, would likewise belong on the "wish list." So would various direct-measurement tools. (For example, there is no ability to measure the values of points on the graphs themselves; if the user wishes to know the value of a particular location on a graph, she has no alternative but to estimate that value by eye.) In the realm of data-analysis, additional programs—analogous to the FFT program of the previous chapter—would be beneficial; one possibility might allow the user to perform sensitivity analyses (showing how certain quantities change with slight changes in parameters such as rate constants).

8.3.2 Integration of Various Modules

Much of the interest of the Workbench is derived from the way that its

various modules communicate with each other: for instance, the qualitative analysis module uses the results of numerical simulation to check the predictions made by the graphical analysis module. However there are avenues of communication between techniques that should be included in future versions of the program. One example was alluded to earlier: it would be desirable to have the results of qualitative analysis dictate the step-size of a parameter-space scan on the fly. A bit less ambitious would be the prospect of having the numerical simulation module suggest integration step-sizes and the limits of graph axes.

The results of graphical analysis currently have little bearing on the running of the numerical simulation module, except in terminating simulations at a known "global equilibrium point" (a relatively unusual occurrence) or suggesting simplifications that the chemist could then use in reimplementing the original mechanism. In an earlier version of the Workbench, the results of graphical analysis could be used to drop certain species from a running mechanism when their concentration approached zero; this technique was more of conceptual than practical interest, but it could nonetheless be profitably reimplemented in the current program.[6]{25}

Adding channels of communication between program modules is more difficult than the sorts of "quantitative" improvements mentioned earlier. Both the "sending" and "receiving" module have to be altered, and occasionally re-thought at a deeper level, in order to implement the new communication strategy. Currently, for instance, the Workbench's technique for Gear integration involves performing the entire numerical run; saving the numerical results to an output file; and then reading those values back. In order to have the program alter integration parameters in mid-run, however, it would be preferable to have a FORTRAN Gear routine callable one step at a time from a Scheme program; and this in turn suggests a vast enhancement project for the MIT Scheme programming environment.

[6]The reason that this technique is not in the current Workbench is that the original implementation, when eliminating a droppable species, actually substituted a new mechanism object to continue the numerical simulation. In the current version, this strategy is more problematic to code because the original mechanism object, annotated with predictions from graphical analysis, must be retained and checked after the simulation is complete.

8.3.3 Improvements in Qualitative Analyses

When the Workbench was originally conceived, the episode data structure was intended as the basis for all qualitative analysis: the hope was that even somewhat complex patterns, such as compound oscillations, could be adequately represented and interpreted via episode patterns. As experience with the program accumulated, however, it became clear that episodes, while informative, were not themselves sufficient for qualitative analysis.

One problem is that an episode record is often difficult to interpret for even rather common patterns like damped oscillations. In an earlier paper about the Workbench {24}, a damped oscillation is exhibited that produces an episode history composed of two-episode repeating sequences, followed by four-episode sequences (as the oscillations die down). It is a messy process to determine from such a record that the same basic oscillation pattern is simply dying down in amplitude. The subsequent addition of derivative zero-crossing structures was motivated by this difficulty in dealing with relatively standard oscillation patterns in the numerical results; though it should be mentioned that the episode record is far from useless in this case, and is employed to link the events of a "typical" oscillation period with the contributions of reactions in the original mechanism.

There are deeper and more interesting reasons why the episode structure has proved problematic for complex mechanisms. The problem arises from the inherent uncertainty in the numerical values of derivative terms—the terms used to define episode boundaries. A major reason why an implicit algorithm like the Gear integrator is needed for stiff systems is that the local values of first-order derivative terms can vary widely over tiny distances in state space; a high Lipschitz constant is in fact a defining feature of stiff systems.{72} This implies that very small numerical errors in the output of the integrator can lead to spurious episode boundaries, and the resulting episode history can prove hard to interpret.

The larger lesson to be taken from this situation is that there is a deep interaction between the choice of numerical methods and the choice of qualitative techniques to interpret numerical results. The very properties that make certain systems difficult to treat numerically can likewise make those

systems difficult to describe qualitatively. This observation suggests, too, that decisions made with the intent of easing numerical analysis—for instance, rescaling of variables—can reap benefits on the qualitative side as well.

Returning to the issue of episode-history interpretation, there are still other enhancements that could be made to the Workbench. One interesting avenue for study involves comparisons of episode histories. For instance, we might want to compare two mechanisms that differ only by the addition of a single step; to do so, we would want a systematic procedure for comparing episode histories of similar mechanisms in a meaningful way. In a similar vein, it would be desirable to look for larger patterns among qualitative results in parameter-space graphs. A kineticist, for example, will often spot a particular pattern of behavioral changes in such a graph (such as the "cross-shaped diagram" pattern for oscillating reactions); but currently the Workbench cannot identify (and therefore make use of) such patterns.

There are still other strategies for qualitative analysis that could be added to the Workbench. One method used by Cheung and Stephanopoulos {14} for intepreting a numerical record employs a series of "triangular episodes" that may be used to parse the record into distinguishable chunks; these data structures are less susceptible than the Workbench's episodes to problems introduced by stiffness. (On the other hand, they also are derived purely from a numerical record and provide no easy conceptual link with the mechanism generating that record.) Another promising possibility would be to employ computer vision techniques like those used by Yip to determine, for instance, the geometry of an attractor in phase space. Such additional techniques might also enhance the "qualitative vocabulary" of the Workbench and enable it to recognize currently problematic numerical records, like the "intermittent bursting" pattern seen for species Z in the Rössler Band example of the previous chapter.

8.3.4 More Complex Systems

An ambitious improvement to the Workbench—really, a reworking of the program—would look toward treating chemical systems described by partial (rather than ordinary) differential equations. In chemical terms, this means

studying systems with concentration gradients in space, as well as reactions that cause concentration changes over time. Especially interesting and difficult examples along these lines occur in the study of oscillating reactions: some models generate "scroll-shaped waves" similar to those seen in the laboratory for the Belousov-Zhabotinskii reaction.{81}

An expansion of the program to handle partial differential equations could also be used to study non-isothermal systems, thus accommodating the dependence of rate constant values upon temperature. Certain explosion reactions are modelled primarily as a combination of reactions and thermal effects.{74}

The changes that would be required to implement such a reworking are enormous and challenging. Prediction of system behavior becomes much less powerful (there is no body of graphical analysis for these systems analogous to the work for homogeneous systems); numerical simulation becomes far more computationally expensive; qualitative analysis must now be employ terms like "wavefronts," "propagating velocity," "destructive interference," and so forth in addition to the (purely temporal) terms already used. Developing computational tools for these systems may be especially worthwhile, however, for precisely the same reasons that make the task difficult: the systems are so complex, and the simulations are so difficult to manage, that researchers may feel themselves that much more in need of non-numerical computational assistance.

8.3.5 Interface

The last topic to be discussed here—the user interface to the Kineticist's Workbench—has been left until the end of this section, not because it is the least important, but because it leads to the considerations of the final section below. The current Workbench interface is underdeveloped, and consists mostly of interaction with the MIT Scheme interpreter. Although there is a foundation for good graphical output in the Workbench, both for viewing mechanism structures and simulation results, the user has little flexibility in working with the program. She cannot, for instance, expand a region of a concentration-versus-time graph; she cannot alter the positions of vertices in a mechanism-structure graph; she cannot use menus to perform common

tasks such as setting parameter ranges; and on and on.

Some of the gaps in the Workbench derive from incomplete aspects of the underlying Scheme environment. At the moment, for instance, there is no direct method in Scheme for using a mouse to "click-and-drag" objects in an X-graphics window—a handy technique for, eg, rearranging the vertices of a graph. But more of the problem lies in the fact that the central goals of the Workbench project have focused more on issues of combining programming paradigms than on user interface.

Despite the need for providing the Workbench with interface features like menus, dialog boxes, mouse-sensitive graphs, and so on, the fact that user input to the program is done through Scheme expressions is not, in and of itself, a problem. In truth, having Workbench commands embedded within Scheme adds a great deal to the flexibility of the program and makes it easy to add new functionality. The "linguistic" aspect of the program interface causes difficulty only through its overuse—because other interface features are unavailable—and through its evolutionary, *ad hoc* design. The Workbench did not have, at its original conception, a draft for a user language; and this is the topic of the next section.

8.4 Philosophical Reflections II: System Design

More than any other type of object, computers seem to be understood through metaphors—they are "giant calculators" or "information processors" or "artificial minds." My own pet metaphor is that computers are a means of expression: people can use these machines to say and understand things that were incomprehensible a half-century ago. The burgeoning field of nonlinear dynamics is an instance of this: portraits of the Mandelbrot set, the Lorenz attractor, the iterates of the quadratic map, become new "things," new objects in the world, for those who see them on the computer screen.

Computational tools should be conceived with the goal of expanding the expressive range of those who use them. The present state of scientific computation has addressed numerical problems, with impressive results; but scientific tools, more than providing convenient output, should engage the sci-

entist in re-expressing the input. Ideally, a good tool should suggest new experiments (both in simulations and in the laboratory) that never would have been conceived without the presence of the computer.

In order to be truly expressive, a computational tool—scientific or otherwise—needs two central elements. The first is a user interface that permits easy, natural expression of operations that are either "cliched" or best performed directly, by hand; this interface should be designed for learnability and to facilitate rapid experimentation. Most likely, such an interface would look like (or be derived from) the marvelous direct manipulation interfaces that have become a growing standard in the personal computer community.

The second element required for a good tool is a language—a programming language in which to express those ideas that inevitably arise and that are beyond the capabilities of any direct manipulation interface. This language will almost certainly require more effort to use than the interface; but once learned, it permits a vastly higher level of creativity on the part of the user.

Currently, the Kineticist's Workbench has a "language" of sorts: a collection of special-purpose objects and procedures that have been embedded in Scheme. The problem with the Workbench's language is that it was designed piecemeal, as issues came up in the original program development: for instance, the notion of a "mechanism object" as an elaboration of the original mechanism data structure arose from the need to annotate mechanisms with the results of graphical analysis. The upshot of adding this new object type is that there are some Workbench procedures that operate upon mechanisms, others that operate on mechanism objects. This not a severe matter for the program developer, but it illustrates a more general point: the current collection of Workbench procedures arose from an accretion of desired operations and functionality, and not from a conscious language-design effort. Perhaps, as a first step, such a development pattern was unavoidable; but the foremost priority in a redesign of the Workbench should now be to construct an appropriate language for combining numerical and qualitative techniques. This, in addition to the interface development mentioned earlier, could be the basis for a tool that is not only more usable than the current program, but that could allow scientists to rethink the very nature of what

it means to perform chemical simulations.

We can indulge in some speculation about how a redesigned Workbench program might appear to the user. First, the chemist could select a menu choice, *Enter New Mechanism*, to create a new mechanism for study. The steps of this mechanism—and other features, like constant species—could be entered via dialog boxes.[7] The chemist could then specify integration methods and parameters in similar ways; and could then perform a numerical simulation with the results graphed on the screen. Alternatively, by selecting a menu choice to *Graphically Analyze* the mechanism, the chemist could obtain the same sorts of information that the Workbench provides. Perhaps additional choices could be provided allowing different sorts of graph representations, or allowing the user to try simplifying assumptions (like Clarke's "fast" and "slow" species) interactively.

Once a simulation is complete, the program might present its qualitative analysis in a text window, and the chemist might have the option of editing this analysis interactively, to add her own observations. She could save the annotated results into a database that allows retrieval of pictorial, numerical, or qualitative results.

Thus far, our imaginary system has been described with the emphasis on interface features; but inevitably, the chemist would want to express ideas that the interface—*any* interface, no matter how well designed—would fail to capture. She might want to "smooth" the numerical data before analyzing it; or find a functional relationship between the amplitude of oscillations and a given parameter; or graph a Poincaré section of the attractor produced by a particular oscillator; or graph the difference between the numerical records obtained with two different methods of integration; or add corrections of various kinds to differential equations to reflect non-mass-action kinetics. These ideas should be economically expressible within a language provided with the system. By gradually gaining experience with this language, the chemist can eventually build more and more complexity into her simulations, going well beyond the functionality apparent in the original interface.

[7] In fact, I have written a Scheme program with precisely this kind of interface for the Apple Macintosh computer. Unlike the Workbench, this program does numerical simulations only, using a Runge-Kutta integrator; but its interface features are more developed than those of the Workbench.

This type of system design should ultimately be the basis for future versions of the Kineticist's Workbench—and "Workbenches" for other professionals. Tools for scientists, artists, mathematicians—tools that combine learnable interfaces with powerful languages—can be realized, even if imperfectly at first. When we finally have these programs in hand, the computer will no longer be a metaphor itself, but a means of expression, a medium for expressing new metaphors.

References

[1] Abelson, Harold
"The Bifurcation Interpreter: A Step Toward the Automatic
Analysis of Dynamical Systems"
MIT AI Memo 1174, Sept. 1989

[2] Abelson, Harold *et al*
"Intelligence in Scientific Computing"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[3] Abelson, Harold and Sussman, Gerald Jay
"The Dynamicist's Workbench: I
Automatic Preparation of Numerical Experiments"
MIT AI Memo 955, May 1987

[4] Abraham, R. and Shaw, C.
Dynamics—The Geometry of Behavior (v. 2)
Aerial Press, Santa Cruz, CA 1984

[5] Bader, G; Nowak, U.; and Deuflhard, P.
"An Advanced Simulation Package for Large Chemical Reaction Systems"
In *Stiff Computation*
Aiken, R.C., ed.
Oxford University Press, 1985

[6] Barnsley, Michael
Fractals Everywhere
Academic Press, Inc. Boston, MA 1988

[7] Batt, L.
"Experimental methods for the study of slow reactions."
In *Comprehensive Chemical Kinetics*
Bamford, C. H. and Tipper, C. F. H., eds., v. I
Elsevier, Amsterdam 1969

[8] Berlekamp, Elwyn; Conway, John; and Guy, Richard
Winning Ways (vol. 2)
Academic Press, London 1982

[9] Benson, Sidney W.
The Foundations of Chemical Kinetics
Robert E. Krieger, Malabar, FL 1982

[10] Biebricher, C.K. *et. al.*
"Modeling Studies of RNA Replication and Viral Infection"
In *Complex Chemical Reaction Systems*
J. Warnatz and W. Jäger, eds.
Springer-Verlag, Berlin 1987

[11] Bodet, J.M.; Vidal, C.; Pacault, A.; Argoul, F.
"Experimental Study of Target Patterns Exhibited by the B.Z. Reaction"
In *Non-Equilibrium Dynamics in Chemical Systems*
Vidal, C. and Pacault, A. eds.
Springer-Verlag, Berlin 1984

[12] Braun, Walter; Herron, John T; Kahaner, David K.
"Acuchem: A Computer Program for Modeling Complex Chemical Reaction Systems"
Intnl Journal of Chem Kinetics, Vol. 20, pp. 51-62 1988

[13] Byrne, George D.
"Software for Differential Systems and Applications Involving Macroscopic Kinetics"
Computers and Chemistry, 5:4, pp.151-158 1981

[14] Cheung, J., and Stephanopoulos, G.
"Representation of Process Trends, Part I.
A Formal Representation Framework"
Laboratory for Intelligent Systems in Process Engineering, May 1989 (LISPE-89-052)

[15] Chua, L.O. and Lin, P.
Computer-Aided Analysis of Electronic Circuits
Prentice-Hall, Englewood Cliffs, NJ, 1975

[16] Cicerone, Ralph J. (1987)
"Changes in Stratospheric Ozone"
Science, v. 237, July 3, 1987

[17] Citri, Ofra and Epstein, Irving R.
"Mechanistic Study of a Coupled Chemical Oscillator:
The Bromate-Chlorite-Iodide Reaction"
Journal of Physical Chemistry, 1988 v.92 p. 1865

[18] Clarke, Bruce L.
"Qualitative Dynamics and Stability of Chemical Reaction Networks"
In *Chemical Applications of Topology and Graph Theory*
R.B. King, Ed.
Studies in Physical and Theoretical Chemistry
Elsevier, Amsterdam 1983

[19] Clarke, Bruce L.
"Recent Developments in the Theory of Stoichiometric
Networks and Application to the Belousov-Zhabotinsky System"
In *Nonlinear Phenomena in Chemical Dynamics*
Vidal, C. and Pacault, A., eds.
Springer-Verlag, Berlin 1979

[20] Davis, Ernest
"Order of Magnitude Reasoning in Qualitative Differential Equations"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[21] de Kleer, J.
"Qualitative and Quantitative Knowledge in Classical Mechanics"
Technical Report No. 352, MIT AI Lab, Cambridge, MA 1975

[22] de Kleer, J. and Brown, J. S.
"A Qualitative Physics Based on Confluences"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[23] Edelson, D.
"A Chemical-Reaction Interpreter for Simulation of Complex Kinetics"
In *Artificial Intelligence Applications in Chemistry*
Pierce, T. and Hohne, B., eds.
American Chemical Society, Washington, DC. 1986

[24] Eisenberg, Michael
"Descriptive Simulation: Combining Symbolic and Numerical Methods in the Analysis of Chemical Reaction Mechanisms"
Artificial Intelligence in Engineering, v. 5 1990

[25] Eisenberg, Michael
"Combining Qualitative and Quantitative Techniques in the Simulation of Chemical Reaction Mechanisms"
In *AI and Simulation: Theory and Applications*,
Webster, W. and Uttamsingh, R., eds.
Society for Computer Simulation, San Diego, CA 1990

[26] Epstein, Irving R.
"Oscillatory Chemical Reactions"
In *Complex Chemical Reaction Systems*
J. Warnatz and W. Jäger, eds.
Springer-Verlag, Berlin 1987

[27] Epstein, Irving R. and Kustin, Kenneth
"A Mechanism for Dynamical Behavior in the Oscillatory Chlorite-Iodide Reaction"
Journal of Physical Chemistry, v. 89, p. 2275 1985

[28] Epstein, Irving R. and Orban, Miklos
"Halogen-Based Oscillations in a Flow Reactor"
In *Oscillations and Traveling Waves in Chemical Systems*
Field, Richard J. and Burger, Maria, eds.
John Wiley and Sons, New York 1985

[29] Farmer, J. Doyne (1981)
"Spectral Broadening of Period-Doubling Bifurcation Sequences"
In *Universality in Chaos*
Cvitanovic, P., ed.
Adam Hilger Ltd., Bristol 1984

[30] Feinberg, M.
"On Chemical Kinetics of a Certain Class"
Arch. Rational Mech. Anal., v. 46 1972

[31] Feinberg, M.
"Complex Balancing in General Kinetic Systems"
Arch. Rational Mech. Anal., v. 49 1972

[32] Feinberg, M.
"Chemical Oscillations, Multiple Equilibria, and Reaction Network Structure"
In *Dynamics and Modelling of Reactive Systems*
Stewart, Warren E., Ray, W. Harmon, and Conley, Charles C., eds.
Academic Press, New York, 1980.

[33] Feinberg, M.
Reaction Network Structure, Multiple Steady States, and Sustained
Composition Oscillations: A Review of Some Results
In *Modelling of Chemical Reaction Systems*
Ebert, K. H., Deuflhard, P., and Jäger, W., eds.
Springer-Verlag, Berlin 1981

[34] Feinberg, M.
"Mathematical Aspects of Mass Action Kinetics"
In *Chemical Reactor Theory: A Review*
Lapidus, L. and Amundson, N., eds.
Prentice-Hall, Englewood Cliffs, NJ 1977

[35] Feinberg, M. and Horn, F.
"Chemical Mechanism Structure and the Coincidence of the Stoichiometric and Kinetic Subspaces"
Arch. Rational Mech. Anal., v. 66 (1977)

[36] Fife, Paul C.
"Propagator-Controller Systems and Chemical Patterns"
In *Non-Equilibrium Dynamics in Chemical Systems*
Vidal, C. and Pacault, A. eds.
Springer-Verlag 1984

[37] Finzel, Barry C. and Moore, John W.
"A Versatile Kinetics Simulation System"
In *Iterations: Computing in the Journal of Chemical Education*
Moore, John W., ed.
Journal of Chemical Education, 1981

[38] Forbus, Kenneth D.
"Qualitative Process Theory"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[39] Forbus, Kenneth D.
Qualitative Physics: Past Present and Future
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[40] Forbus, Kenneth D. and Falkenhaimer, B.
"Self-Explanatory Simulations: An Integration of Qualitative
and Quantitative Knowledge"
Eighth National Conference on Artificial Intelligence, 1990

[41] Gladd, N.T. and Krall, N.A.
"Artificial Intelligence Methods for Facilitating Large Scale
Numerical Computations"
In *Coupling Symbolic and Numerical Computing in Expert Systems*
Kowalik, J.S., ed.
North-Holland, Amsterdam 1986

[42] Hague, D.N.
"Experimental Methods for the Study of Fast Reactions"
In *Comprehensive Chemical Kinetics*
Bamford, C. H. and Tipper, C. F. H., eds., v. I
Elsevier, Amsterdam 1969

[43] Hammes, Gordon G.
Principles of Chemical Kinetics
Academic Press, Inc., Orlando, FL 1978

[44] Hamming, R. W.
Numerical Methods for Scientists and Engineers (second edition)
Dover, NY 1973

[45] Hanusse, P.
Modelling of Chemical Dynamics
In *Non-Equilibrium Dynamics in Chemical Systems*
Vidal, C. and Pacault, A., eds.
Springer-Verlag, Berlin 1984

[46] Hindmarsh, A.
The ODEPACK solvers
In *Stiff Computation*,
Aiken, R., ed.
Oxford University Press 1985

[47] Horn, F.
"Necessary and Sufficient Conditions for Complex Balancing in Chemical Kinetics"
Arch. Rational Mech. Anal., v. 49 1972

[48] Horn, F. and Jackson, R.
"General Mass Action Kinetics"
Arch. Rational Mech. Anal., v. 47 1972

[49] Isbarn, G.; Ederer, H.; and Ebert, K.
"The Thermal Decomposition of *n*-Hexane: Kinetics, Mechanism,
and Simulation"
In *Modelling of Chemical Reaction Systems*
Ebert, K. H., Deuflhard, P., and Jäger, W., eds.
Springer-Verlag, Berlin 1981

[50] IMSL, Inc.
FORTRAN Subroutines for Mathematical Applications
Houston, Texas, 1987

[51] Kondratiev, V. N.
"Chain reactions."
In *Comprehensive Chemical Kinetics*
Bamford, C. H. and Tipper, C. F. H., eds., v. I
Elsevier, Amsterdam 1969

[52] Kuipers, Benjamin J.
Abstraction by Time-Scale in Qualitative Simulation
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[53] Kuipers, Benjamin J. and Berleant, Daniel
"Using Incomplete Knowledge in Qualitative Reasoning"
Proceedings, Seventh National Conference on
Artificial Intelligence, 1988

[54] Kuipers, Benjamin J. and Chiu, Charles
"Taming Intractable Branching in Qualitative Simulation"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[55] Laidler, Keith J.
Chemical Kinetics (third edition)
Harper and Row, New York 1987

[56] Lotka, Alfred J. (1924)
Elements of Mathematical Biology
Dover Books (reprint) 1956

[57] Macaulay, David
The Way Things Work
Houghton Mifflin, Boston, MA 1988

[58] Mahan, Bruce H.
University Chemistry (second edition)
Addison-Wesley, Reading, MA, 1969

[59] Markus, M., Liefke E., and Onken U.
"Bistability and Oscillations in the Oxidation of Hydrazine"
In *Complex Chemical Reaction Systems*
J. Warnatz and W. Jäger, eds.
Springer-Verlag, Berlin 1987

[60] Mavrovouniotis, Michael and Stephanopoulos, George
"Formal Order-of-Magnitude Reasoning in Process Engineering"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[61] Nicolis, G. and Prigogine, I.
Self-Organization in Nonequilibrium Systems
John Wiley and Sons, New York, 1977

[62] Noyes, R.M.
"Mechanisms of Chemical Oscillators"
In *Synergetics: Far from Equilibrium*
Pacault, A. and Vidal, C., eds.
Springer-Verlag, Berlin 1979

[63] Noyes, R. M.
"Some Factors Generating Temporally Ordered Behavior"
In *Temporal Order*
Rensing, L. and Jaeger, N.I., eds.
Springer-Verlag, Berlin 1985

[64] Parker, T.S. and Chua, L.O.
Practical Numerical Algorithms for Chaotic Systems
Springer-Verlag, New York, 1989

[65] Raiman, Olivier
"Order of Magnitude Reasoning"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[66] Rand, R.H. and Keith, W. L.
"Normal Form and Center Manifold Calculations on MACSYMA"
In *Applications of Computer Algebra*
Pavelle, R., ed.
Kluwer, Boston 1985

[67] Rössler, O.E.
"Chaos and Strange Attractors in Chemical Kinetics"
In *Synergetics: Far from Equilibrium*
Pacault, A. and Vidal, C., eds.
Springer-Verlag, Berlin 1979

[68] Sacks, Elisha
"Automatic Qualitative Analysis of Ordinary Differential
Equations Using Piecewise Linear Approximations"
Tech. Report 1031, MIT Artificial Intelligence Laboratory 1988

[69] Sacks, Elisha and Doyle, Jon
"Prolegomena to Any Future Qualitative Physics"
Unpublished manuscript, 1991

[70] Samardzija, N., Greller, L. and Wasserman, E.
"Nonlinear Chemical Kinetic Schemes Derived
From Mechanical and Electrical Dynamical Systems"
Unpublished manuscript, 1988

[71] Schlosser, P. M. and Feinberg, M.
"A Graphical Determination of the Possibility of Multiple Steady States
in Complex Isothermal CFSTRs"
In *Complex Chemical Reaction Systems*
J. Warnatz and W. Jäger, eds.
Springer-Verlag, Berlin 1987

[72] Shampine, L.F.
"What is Stiffness?"
In *Stiff Computation*,
Aiken, R.C., ed.
Oxford University Press 1985

[73] Shacham, Mordechai and Cutlip, Michael B.
"A Simulation Package for the Plato Educational Computer System"
Computers and Chem. Eng., 6:3, pp 209-218 1982

[74] Steinfeld, Jeffrey I., Francisco, Joseph S., and Hase, William L.
Chemical Kinetics and Dynamics
Prentice Hall, Englewood Cliffs, NJ 1989

[75] Swinney, H. L. and Roux, J. C.
Chemical Chaos
In *Non-Equilibrium Dynamics in Chemical Systems*
Vidal, C. and Pacault, A., eds.
Springer-Verlag, Berlin 1984

[76] Thompson, J.M.T. and Stewart, H.B.
Nonlinear Dynamics and Chaos
John Wiley and Sons, Chichester 1986

[77] Troy, William C.
"Mathematical Analysis of the Oregonator Model of the Belousov-Zhabotinskii Reaction"
In *Oscillations and Traveling Waves in Chemical Systems*
Field, Richard J. and Burger, Maria, eds.
John Wiley and Sons, New York 1985

[78] Tyson, J.J.
"The Coordination of Cell Growth and Division: A Comparison of Models"
In *Temporal Order*
Rensing, L. and Jaeger, N. I., eds.
Springer-Verlag, Berlin 1985

[79] Vidal, C. and Pacault, A.
"Spatial Chemical Structures, Chemical Waves. A Review"
In *Evolution of Order and Chaos*
H. Haken, ed.
Springer-Verlag, Berlin 1982

[80] Williams, Brian C.
"Temporal Qualitative Analysis: Explaining How Physical Systems Work"
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[81] Winfree, Arthur T.
"Organizing Centers for Chemical Waves in Two and Three Dimensions"
In *Oscillations and Traveling Waves in Chemical Systems*
Field, Richard J. and Burger, Maria, eds.
John Wiley and Sons, New York 1985

[82] Yip, Kenneth
Generating Global Behaviors Using Deep Knowledge of Local Dynamics
Reprinted in *Readings in Qualitative Reasoning About Physical Systems*
Weld, Daniel S. and de Kleer, Johan, eds.
Morgan Kaufmann, San Mateo, CA 1990

[83] Yip, Kenneth
KAM: Automatic Planning and Interpretation of
Numerical Experiments Using Geometrical Methods
Tech. Report 1163, MIT Artificial Intellgence Laboratory 1989

Appendix A

The Zero Deficiency Theorem

This appendix contains a brief sketch of the proof of the Zero Deficiency Theorem (ZDT). The sources for this discussion are {30, 31, 34, 35, 47, 48}.

The key "intermediate concept" in the proof of the ZDT is *complex balancing*: a mechanism is said to be complex balanced if it is in a state in which, for every complex, the rate of increase of all species due to reactions contributing to that complex is equal to the rate of decrease due to reactions in which this complex is a reactant. For example, consider the following mechanism:

```
[1]      A + B <--> C <--> D

         D + E <--> 2B
```

To check for balancing for the complex `A + B`, we now look to see whether the rates of the following two reactions are equal:

```
A + B --> C

C --> A + B
```

If these rates are equal—if the rate of decrease in `[A]` due to the first is equal to the rate of increase in `[A]` due to the second—then the mechanism is balanced for the complex `A + B`. If similar statements hold true for *every* complex, then the mechanism is complex balanced.

Obviously, a complexed balanced mechanism is in equilibrium—no concentrations are changing. However, it is important to note that the converse need not be true. For instance, just focusing our attention on species `B`, it is possible that the concentration of `B` is declining due to the the two reactions involving the complex `A + B`; but the net concentration of `B` may be stable due to the contribution of the two reactions involving the complex `2B`. Thus,

the mechanism may be in equilibrium for B without being complex balanced. The following is a more striking example, from {48}:

```
[2] A --> B

    2B --> 2A
```

Obviously, one can choose rate constants to provide an equilibrium state for [2]; but the mechanism is never complex balanced.

We can now state the first important theorem regarding complex balancing:

Theorem 1. {48} A mass action mechanism is either complex balanced at all its equilibria or at none of them. ("Equilibria" here do not include states with zero concentration values.) Moreover, if it is complex balanced, the equilibrium is stable within the stoichiometric subspace.

The proof of this in {48} is done in several steps:

1. First the theorem is proven for simple "cyclic" systems, i.e., mechanisms consisting of a cycle of complexes. Two examples are shown below:

```
[3]      A <--> B

[4.1]    C --> D + E
[4.2]    D + E --> F
[4.3]    F --> C
```

2. Then the theorem is proven for arbitrary mechanisms by decomposing any given mechanism into a set of cyclic mechanisms. Since the theorem holds for each component submechanism, it likewise holds for the overall composite mechanism.

3. Moreover, it is shown earlier in {48} that for complex balanced mechanisms, each equilibrium state is locally stable, and unique for a given stoichiometric subspace. (In the terminology of {48}, the mechanism is *quasithermodyamic.*)

Since we are going to assume mass action kinetics throughout this discussion we can shorten the statement of Theorem 1 to the following:

Theorem 1. A mechanism is complex balanced either at all equilibria or none. If it is complex balanced at all equilibria, then the system is quasithermodynamic.

From Theorem 1, we can conclude that "complex-balancing" is a property of the mechanism itself—not just a property of the mechanism at some particular set of concentrations. In other words, we can divide those mechanisms that are capable of equilibrium states into "mechanisms with all complex-balanced equlibria" and "mechanisms with non-complex-balanced equilibria." The next two theorems relate structural properties of a given mechanism to the question of complex balancing.

Theorem 2. {31} If a mechanism has deficiency 0, then it will exhibit complex balancing at every equilibrium (if any exist).

The deficiency 0 property is expressed as a statement of linear algebra. (Briefly: that if we express the reactions of the mechanism as vectors in "complex space," then the dimension of the stoichiometric subspace is equal to the dimension of the space spanned by the "complex space vectors." In other words, if we express the available state space either through vectors in species space, or through vectors in complex space, we arrive at spaces of the same dimension.) Likewise, a particular sufficient condition for complex balancing at every equilibrium is expressed in linear algebra. (Briefly, an equilibrium point is expressed as a point in complex space, corresponding to the rate of formation of each individual complex; and it is shown that all equilibrium points must lie within a particular linear subspace of complex space. If that subspace has dimension zero—i.e., if it consists only of the zero vector in complex space—then any equilibrium point must correspond to that zero vector in complex space. Thus, if this special subspace has dimension zero, any equilibrium point must correspond to a point in which the rate of formation of all complexes is zero.) It is then shown that the two statements in linear algebra are in fact equivalent: that the deficiency 0 property corresponds to a sufficient condition to guarantee complex balancing at any equilibrium, should one exist.

Theorem 3. {47} If a mechanism is capable of a complex-balanced equilibrium, it is weakly reversible.

The proof in {47} uses a notion of "adopting" and "decaying" subsets of the complex set—roughly, a set is "adopting" relative to some complex C if it "gets reaction flow" from C. It is then shown that it is impossible for a non-reversible graph to exhibit complex balancing (since this would imply that an "adopting" set of complexes relative to some particular starting complex C is a "non-decaying" set relative to C, which condition is incompatible with complex balancing).[1]

From Theorems 2 and 3, we can conclude:

Theorem 4. If a mechanism has deficiency 0, but is not weakly reversible, then it has no nonzero equilibrium state in any given stoichiometric subspace.

The next theorem, from {47}, is the last crucial step in the proof of the ZDT:

Theorem 5. If a mechanism has zero deficiency and is weakly reversible, then it has a complex balanced equilibrium.

First it is shown that the weak reversibility condition (for any mechanism) implies the presence of a nonzero kernel for the "rate matrix" A which takes points in complex space to their associated complex-formation vectors. In other words, weak reversibility implies an "equilibrium subspace" in complex space. Zero deficiency—again expressed in terms of linear algebra—implies that we can find a set of concentrations within the available stoichiometric subspace that will correspond to a point in this equilibrium subspace. (In other words, we can find some set of concentrations $\mathbf{c}$ such that the matrix A, when applied to the complex vector derived from $\mathbf{c}$, returns the zero complex-formation vector.) Thus, the two conditions—zero deficiency and weak reversibility—together guarantee that at some set of concentrations, the net rate of formation of each particular complex is zero.

Now, using Theorems 5 and 1, we conclude:

[1] I believe that a simpler, inductive proof based in graph theory should be possible for this theorem.

Theorem 6. If a mechanism has zero deficiency and is weakly reversible, then it has a unique stable nonzero equilbrium in each stoichiometric subspace.

The overall Zero Deficiency Theorem is now the conjunction of Theorems 4 and 6.[2]

[2]There is actually still a bit more to the theorem, involving the inability of zero deficiency mechanisms to exhibit cyclic trajectories in state space. For reversible mechanisms, this is guaranteed by the "quasithermodynamic" condition mentioned earlier. For nonreversible mechanisms, the proof is outlined in {34}.

Appendix B

Algorithms for Graphical Analysis

This appendix contains outlines of the important algorithms used to perform graphical analysis of mechanisms.

B.1 Necessary Nonzero Species

1. Let CS denote all species that are constant with nonzero values, or that have external sources. These are clearly necessary nonzero species. Let NZ denote already-found nonzero species (initially CS).

2. Now, find all species with the following properties:

   This species s is not in NZ.

   This species is a product in a reaction whose reactants are all in NZ.

3. If no species were found in step 2, return the current value of NZ. Otherwise, append all species found in step 2 to NZ and go back to step 2.

B.2 Obvious Declining Sets of Species

1. Let LHS_i denote the left hand (reactant) side of the reaction i, and RHS_i denote the right hand (product) side. Find all LHS_i such that for all j, $LHS_i \cap RHS_j = \{\}$. For each such LHS_i, let $LHS_i - CS$ denote the difference of LHS_i and nonzero constants and source species. Include $LHS_i - CS$ as an element of DS, the set of declining sets of species.

2. Delete from the mechanism all reactions found in step 1, and perform step 1 again until no new sets are found.

3. Find all LHS_i that contain a species that only occurs as a reactant. Add the (non-constant and non-source) portion of each such LHS_i to DS, delete these reactions and go back to step 1.

4. Find all LHS_i with the following properties:

> a. The corresponding RHS_i consists solely of sink species, driven off species, or species that only occur as products.
>
> b. At least one species in LHS_i does not have a nonzero constant or source among its "ancestors" (i.e., there is no sequence of reactions R_j, R_{j+1},...R_{j+m} such that a nonzero constant or source is among the reactants of R_j; the target species is among the products of R_{j+m}; and for each pair of adjacent reactions, there is a non-null intersection of the products of the first and the reactants of the second). In prose, what we are looking for is a reaction none of whose reactants has any conceivable dependence on nonzero constants or sources, and whose reactants produce "disappearing" (sink or driven-off) species.

Also,

5. Find all reactions all of whose reactants satisfy condition *b* of the previous step, and at least one of whose reactants is a sink or driven-off species. Delete all reactions found in steps 4 and 5 (appending their reactants to DS) and go back to step 1. If no such reactions are found, go on to step 6.

6. Look for reactions for which the set of "ancestor" species (in the sense of condition *b* of step 4) and the set of "daughter" species (in the same sense) are distinct, and for which the reactants satisfy condition *b* of step 4. In such a case, the reaction is "feeding" a product set that cannot possibly serve to augment any of the species in the reactant set. Thus, the reactants must constitute a declining set. Append this set to DS and go back to step 1.

7. Delete all the reactions whose reactants or products are a superset of some element in DS. Append the reactant sets of such steps to DS, and go back to step 1. If none are found go on to step 8.

8. For each element *declset* in DS, look at each species s in *declset* and find those species that must have zero concentration if this species does (this is done via the algorithm described below for consistent zero sets). The union of such "related species" found for *declset* must itself constitute a declining set (the product of all these species' concentrations must approach zero). If the result of this process is a new set (one that is not a superset of any existing declining set), include this in DS.

We do not go back to step 1 at the conclusion of step 8, though perhaps we should; in any event, the algorithm concludes with this "second-level" search. All the returned sets are declining sets, though there is no guarantee that every declining set will be found (eg, there may be "third-level sets" that go unnoticed). In practice, this algorithm has found all declining sets for the mechanisms tested thus far.

It should also be mentioned that this algorithm may be fooled by mechanisms in which constant species and sources are not expressed explicitly. For instance, in the mechanism:

```
A --> 2A -->  External World
```

this algorithm would find that A is a declining set (it is "lost" to a sink), even though a nonzero steady state concentration of A is possible. The problem with this mechanism is that, as expressed, it does not obey mass conservation. A more plausible chemical mechanism might appear as follows:

```
A + B --> 2A --> External World
```

with B treated as constant. In this case, the algorithm would (correctly) find no declining sets.

B.3 Consistent Zero Sets of Species

1. Find those species that are not necessary nonzeros, "driven-offs," or species that only occur as reactants.

2. For each such species s, assume that its concentration is zero and that the system is at a steady state. The initial set of zero sets Z is the singleton

consisting of (again a singleton) $\{s\}$.

3. Find those reactions R_i which have s among their products. Take the power set of the reactants in R_i, and for each element in this power set, take the set union with $\{s\}$. This is the new set of zero sets Z.

4. We continue the process begun in step 3. For each set in Z, and for each newly-added species in that set, we find those zero subsets that might be appended to the set. We continue this process until each set in Z has the property that all its species have been examined as "roots," and thus there are no new species that can be added to the set by tracing any of the species "one reaction arrow back."

5. At the end of step 4, we have a (possibly large) collection of possible internally-consistent zero sets in Z. We now remove all those sets in Z that are a superset of some other set in Z, since we only want minimal internally-consistent zero sets.

B.4 Catalytic Pathways

1. We are interested in the possibility of a "catalytic pathway" for some species s (i.e., a reaction path in which s acts as a catalyst). We find all sequences of one or more reactions R_i, R_{i+1},... R_{i+m} that have the following properties:

   a. s is among the reactants in R_i and the products in R_{i+m}, and does not appear in any other reactant or product set in the sequence.

   b. For each adjacent pair of reactions in the sequence, there is a non-null intersection between the products of the first reaction in the pair and the reactants of the second. (That is, we can think of each reaction as "contributing one or more reactants to" the next one.)

   c. For each reaction after the first in the sequence, none of the reactants must enter into a reaction with s elsewhere in the mechanism.

Currently, the program looks for such sequences only up to a maximum length of three.

2. Having found such a sequence, we now find how many molecules of s in the final reaction are produced from one molecule of s in the initial reaction. In doing so, we need to "multiply" intermediate reaction coefficients; if there is more than possible factor to multiply by (corresponding to a pair of reactions (R_i, R_{i+1}) in which more there is more than one species in the intersection of the products of R_i and the reactants of R_{i+1}), then we multiply by the minimal factor (that is, we choose the "connection species" between reactions that corresponds to the least production of s in the final reaction). Also we check that our path is not simply a (two-step) reverse path—that is, a "forward" and "backward" reaction pair.

3. If there is one molecule of s produced in this sequence for every molecule consumed by the first reaction, then this path is labelled a "catalytic path."

B.5 Autocatalytic Pathways

This algorithm is similar to the one described above, except in the final step we check that more than one molecule of s is produced for each molecule of s consumed in the first reaction.

It should be noted that there are "autocatalytic sequences" (and similar catalytic sequences) not caught by these algorithms. For instance, we might have the following set of reactions:

```
A --> B + C

B --> A

C --> A
```

The algorithm described in B.4 will find two two-step catalytic sequences, but we will not find the (net) three-step autocatalytic production of **A** since there is no three-step simple path of reactions meeting the conditions given above. The problem here arises from the fact that the autocatalytic generation of **A** is due to the combined effects of two separate products of **A** in the first reaction. It would certainly be possible (and desirable) to enhance the algorithms already in the Workbench with additional techniques to look for patterns like this one.

B.6 Qualitative Arithmetic

As mentioned in Chapter 5, the current qualitative arithmetic capabilities of the Workbench are rather primitive. There is a global variable named `*qualitative-epsilon*`, whose value determines the "scale" for a number of qualitative comparisons between numbers. By default, the value of `*qualitative-epsilon*` is 0.01, and (writing this value as ϵ), a number of other quantities are defined in related fashion:

`*qualitative-epsilon*` $= \epsilon$

`*approx-equal-factor*` $= 1 + \epsilon$

`*equal-factor*` $= 1 + \epsilon^2$

`*greater-than-factor*` $= \epsilon^{-1}$

`*much-greater-than-factor*` $= \epsilon^{-2}$

All of these quantities are effectively reset by performing a call to the procedure `reset-qualitative-epsilon!` as illustrated in Chapter 5:

```
(reset-qualitative-epsilon! 0.1)
```

The most important procedure in the qualitative arithmetic package is the `get-approximate-factor` procedure, which takes two numeric quantities and, using the global variables shown above, returns a symbol indicating the relation between those two quantities. This procedure is sufficiently brief to be shown in full:

```
(define (get-approximate-factor n1 n2)
  (cond ((= n1 0) (if (= n2 0) '= 'undefined))
        (else (let ((fct (abs (/ n2 n1))))
                (cond ((> fct *much-greater-than-factor*) '>>>)
                      ((> fct *greater-than-factor*) '>>)
                      ((< fct (/ *much-greater-than-factor*)) '<<<)
                      ((< fct (/ *greater-than-factor*)) '<<)
                      ((and (> fct (/ *equal-factor*))
                            (< fct *equal-factor*)) '=)
                      ((and (> fct (/ *approx-equal-factor*))
                            (< fct *approx-equal-factor*)) '~)
```

```
((< (abs n2) (abs n1)) '<)
(else '>))))))
```

Clearly there is very little sophistication in this arrangement; though again, it should be pointed out in fairness that the Workbench does not currently require an elaborate qualitative arithmetic system to perform interesting identifications of fast equilibria and steady-state candidates. The algorithms for these classification procedures are described below.

B.7 Fast Submechanisms

The Workbench algorithm for identifying fast zero-deficiency submechanisms is as follows:

1. First, all reactions are ordered by the magnitude of their rate constants (to achieve a rough ordering from "fast" to "slow" reactions). In the remainder of the algorithm, we will look, one by one, at the fastest still-unexamined reaction; and the set of already-picked reactions will constitute the "current reaction set" in which we are seeking fast submechanisms.

2. Pick the next reaction and add it to the current set. If possible, find a subset of reactions S that constitutes a weakly-reversible zero-deficiency mechanism. The set S must have the following properties:

   a. No other complex in the current set (i.e., no product or reactant set) is part of a reaction that would, if added to S, create a non-weakly-reversible mechanism.

   b. No other complex in the current set contains a species that occurs in S.

3. We add S to a list of subsets tried, and look to see if the slowest reaction in S is faster than the fastest reaction (now assuming concentrations of `*greater-than-factor*`) involving these species in the overall mechanism. If so, we note that S is a fast submechanism; remove all reactions in S from the overall mechanism; and continue picking reactions as described in step 2, until there are no further reactions to examine.

B.8 Steady State Candidates

For every individual species in the mechanism, we check to see whether this species is a plausible candidate for a steady-state approximation. Our heuristic for this purpose will be to examine whether at least one of the reactions in which this species participates as reactant is much faster than all of the reactions producing this species, under "reasonable" assumptions for concentrations of all species. To examine a given species for steady-state candidacy, we do the following:

1. Find all reactions *Rct* in which this species *s* participates as reactant with a coefficient of 1, and all reactions *Prd* producing this species. Let *RctS* be the set of species in *Rct* that do not occur as reactants in *Prd*, and let *StSt* be the set of species that we already assume to be steady-state candidates. (If *s* occurs as reactant in any reaction and has a coefficient greater than 1, as in (say) `2 s --> P`, then we do not consider it as a possible steady-state candidate. Likewise, we do not consider species that participate in steps both as reactant and product.)

2. For all reactions in *Prd* that produce this species, assume concentrations of `*greater-than-factor*`$^{-1}$ for all species in *StSt* (that is, we assume a "small" concentration for our current set of steady-state species); assume a concentration of `*greater-than-factor*` for all species that occur as reactants in *Prd*; and assume a concentration of 1 for all other species (i.e., those in *RctS*).

3. We now have a "reaction-out-factor," corresponding to the fastest step in which *s* participates as reactant, to compare to a "reaction-in-factor," corresponding to the fastest (assumed) step producing this species. The two expressions being compared are *reaction-in-factor* and [*s*] * *reaction-out-factor*. If *reaction-out-factor* is much greater than *reaction-in-factor*, in the sense described earlier in the section on qualitative arithmetic, then we consider *s* to be a plausible steady-state candidate. Our claim is based on the assumption that a small concentration of *s* will render the two terms of fastest production and consumption approximately equal.

As noted in the text, this algorithm is approximate in several respects. First,

it does not return every species that might in fact constitute a good steady-state candidate: for instance, it does not consider species X to be a good candidate in the following mechanism:

```
A --> X    k1 = 1

2X --> B   k2 = 1e6
```

The reason our algorithm does not consider X here is because it participates as reactant with a coefficient of 2 in the mechanism. (Such situations could be handled at the cost of a little extra complication in the algorithm.)

A second problem is that the algorithm returns as possible candidates species that may in fact not constitute steady-state species for one of a variety of reasons: the assumptions regarding species concentrations may, for instance, be inaccurate (it may be that certain species are present with much greater or smaller concentration than we assume). Another flaw occurs when the steady-state assumption leads, in a sense, to its own undoing: by assuming a steady state for s, we build up the concentration of some other species that eventually contradicts the assumptions that led us to a steady-state assumption in the first place. Finding patterns of this kind would be a somewhat more sophisticated project (and an interesting one); but the Workbench's current algorithm does appear to handle the easier "textbook" cases, such as the `N2O5` decomposition example shown in the text.

B.9 Rapid Location of Equilibria

The Workbench has two algorithms for rapid location of global equilibrium points. The first applies to linear mechanisms only, and locates the global equilibrium by a simple Gaussian elimination technique. (The algorithm as currently implemented does only the easiest possible thing, triangulating the appropriate system matrix and solving for the given unknown concentrations; searches for numerically optimal "pivot" variables, for instance, are not employed.)

The second, more elaborate algorithm consists of two phases, as outlined in Chapter 5: a "star-walk" phase, and a "Runge-Kutta integration" phase.

These two portions of the algorithm run in interleaved fashion: the star-walk is tried for a certain amount of time (until the step-size becomes sufficiently small), and then the RK integration is done for a certain number of time-steps (or until the equilibrium point is approached to a sufficient tolerance). A more detailed explanation follows.[1]

B.9.1 Star-Walk

Since we are seeking an equilibrium state, we are (equivalently) seeking a point in phase space, accessible from the starting state, in which the magnitude of the system derivative is zero. Moreover, we are guaranteed from our graphical analysis that our system has only one such (nonzero) state.

1. Beginning at the initial state, we look at each reaction in the mechanism and find that reaction which, when run to a certain percentage p of completion, would result in the largest decrease in the magnitude of the system derivative.[2] We run p percent of this best reaction, if one is found. If no such reaction is found, we decrease p by a certain factor and try again. If, on the other hand, a best reaction is found *and* if this is also the best reaction for a larger value of p (increased by a given factor), then we also increase p by an increase-factor in preparation for the next step.

2. If we find that p has decreased for a certain number of consecutive steps, and that the system derivative's magnitude is small (relative to its original value); or if we find that the second condition alone has been true for some

[1]The current algorithms for finding equilibria work well for mathematically tractable mechanisms, and could conceivably be extended to more complex mechanisms, as mentioned in Chapter 5. An alternative and potentially powerful strategy would be based directly in computer algebra—to use a nonlinear algebraic equation solver (the Workbench only employs this direct solution for linear systems). Adding such a capability to the Workbench is feasible though nontrivial, and could well be listed with some of the other desirable extensions mentioned in Chapter 8.

[2]The notion of "running a reaction to completion" simply means running the reaction until one or more of the reactants reaches zero concentration. Thus, to run the reaction `A + B --> C` to completion would mean allowing `A` or `B`, whichever had lesser initial concentration, to reach zero concentration. Assuming for argument's sake that `[A] < [B]`, running the reaction to p percent of completion means that p`[A]` moles per liter are subtracted from both `[A]` and `[B]`, and added to `[C]`.

number n of consecutive steps, then we halt the star-walk procedure and go on to the RK phase. Otherwise, we continue by iterating step 1.

What we have provided here is an overall picture of the algorithm; all of the symbolic parameters mentioned here have default values in the current version of the program. For instance, the initial value of p is 0.005 (that is, one half of one percent). The factor by which to increase p (if appropriate) is 1.2, and the factor by which to decrease p is $(1.2)^{-1}$. (There are a few additional complications: for example, there is a maximum value beyond which p will not go, which for the current algorithm is 0.04.) Likewise, the initial value of the "successive small derivative counter" is 10, the initial "successive p-decrease counter" is 25 (or, if the system derivative is sufficiently small, 9), and so forth. These values have been arrived at by experimentation with a variety of reasonably simple systems whose equilibrium values are at moderate concentrations; more robust, adaptive methods for setting these parameters would be needed to handle more complex systems.

B.9.2 Runge-Kutta Integration with "Predictive Jumps"

In the RK phase of the algorithm, we simply perform fixed-time-step Runge-Kutta integration (the default time-step is 0.1 seconds) for up to some maximum number n (default 100) steps. During the course of these steps, the program looks for a series of m consecutive steps in approximately the same direction and having the form

$$dV, xdV, x^2dV, x^3dV...x^{m-1}dV$$

where $x > 0$ and $|x| < 1$ then the algorithm returns the value

$$P_0 + (1/(1-x))dV$$

where P_0 is the spot in state space where the sequence was first detected. (The default value of m is 5, and two vectors are deemed to be "in approximately the same direction" if their individual components each differ by a factor f such that $0.96x < f < 1.04x$.)

If the RK phase was entered due to the star-walk phase arriving at a local minimum, then the returned value of this phase is the final value of the entire equilibrium-search algorithm. Otherwise, we return to the star-walk phase,

which tests the returned value for a sufficiently small system derivative; if the value is sufficiently small, it is returned, and if not, the star-walk procedure is resumed. A "sufficiently small" system derivative is one whose magnitude is less than a given factor (default 0.0015) of the initial derivative magnitude, or is less than another factor (default 0.015) of the magnitude of the actual state vector itself. Again, in practice, these values are chosen based on experience with straightforward examples whose equilibrium concentration values have moderate values; to extend the algorithm to more complex systems, a more careful choice (possibly based on the second test above, but checking individual components of the derivative vector against the components of the state vector) would need to be implemented.

Appendix C

Algorithms for Qualitative Analysis

C.1 Attaching Feature Descriptors to Episodes

Short/Long Episodes

As described in Chapter 6, the episode history itself is used as the basis of the definition of "short" and "long" times. After transition episodes are removed (again, see Chapter 6), the shortest episode length t_s and the longest t_l are used to define a "time ruler" defined as the difference of the logs of the two values:

$$ruler = \log t_l - \log t_s$$

A "short" time is then defined as a time t such that

$$\log t < \log t_s + 0.25 * ruler$$

and a "long" time is a time t such that

$$\log t > \log t_s + 0.85 * ruler$$

The time-periods represented by the right sides of the last two conditions are known as the "longest short time" and the "shortest long time," respectively; these time-periods appear in some of the tests that follow.

Steady-State

This is the most complicated of all the Workbench's feature tests. Basically, we look to see if both the net difference in concentrations (from beginning to end of the episode) is small, and if the net variation in derivative is also small. In doing so, we test for one of several possible conditions, of which the most important (and common) is as follows:

```
Net-difference / Starting-concentration < 0.01  and

Net-difference-of-deltas / Starting-concentration < 0.005
```

Another possible condition is that this is the final episode, and that it is a long episode. In this case the two tests above are made a bit more lenient (the "test values" are 0.025 and 0.01 instead of 0.01 and 0.005, respectively), but we also check to see that the final delta is small:

```
(shortest-long-time / dt) * (final-delta / final-conc) < 0.005
```

Yet another possibility is that the original two tests are not met because the episode is extremely long (and hence the net concentration difference is just large enough to make the first test inapplicable). In this case, we keep the second of the two tests above, but change the first test to:

```
(shortest-long-time / episode-duration)

    *  (net-difference / starting-concentration)  < 0.01
```

Large-Increase/Decrease

A "large increase" is one in which either the concentration of a given species increases from an initial value of 0, or increases by more than half of the final concentration. A "large decrease" is one in which either the species decreases to a final concentration of 0, or decreases by a total greater than one third of the final concentration.

Wide-Swing

There are several conditions tested here, but the only non-anomalous condition tests that the difference between the maximum and minimum value

of the species over the course of this episode is greater than 5 times the net difference in concentration from the beginning to the end of the episode. (The idea here is that we wish to note the possibility of a large jump followed by a large decrease in concentration over the course of the episode—or a decrease followed by a jump—such that the beginning and ending concentrations are close.)

Rapid-Increase/Decrease

A "rapid" increase in concentration is one which corresponds to a growth of more than 0.5 of the final concentration within the "longest short time." A rapid decrease corresponds to a decrease of more than 0.1998 of the final concentration within the "longest short time."

Slow-Increase/Decrease

A "slow" increase corresponds to a growth of less than 0.05 of the final concentration within the "shortest long time." A slow decrease corresponds to a decrease of less than 0.0475 of the final concentration within the "shortest long time."

C.2 Repeating Patterns in Episodes

The Workbench looks within episode histories to find repeating episode patterns that might signal the occurrence of oscillations. Although this capability is not as important now as it was at the program's inception (given the introduction of derivative zero-crossings), the episode-chunking is still printed out after runs when the episode history is sufficiently short.

Once an episode history is obtained, it is "coarse-grained" as described in Chapter 6 to eliminate apparent "transition episodes" (and typically to reduce the history solely to episodes that differ in the most important step). The program then looks for repetitions of groups of episodes that might allow as many as 1.6 complete oscillations; for instance, an 8-episode history will be tested for repeating patterns of as many as 5 episodes (allowing for a

5-episode sequence followed by a partial repeat of that sequence in the final three episodes).

For the purposes of chunking episodes into groups, two episodes are deemed equal if they have equal step-orderings up to a given position (usually the "zeroth" or initial position); beyond this, successive corresponding episodes in an attempted chunking are tested for "feature-compatibility." The idea is that if we wish to find a three-episode chunking of some history, and want to compare (eg) the fourth and seventh episodes in the history as part of this attempted chunking, then we would expect the features of these two episodes to be, if not identical, at least not wildly incompatible. For example, if some species `X` has a steady increase in the fourth episode and a decrease in the seventh episode, we would not want to declare these episodes as occupying similar locations in successive oscillations, regardless of the step-orderings in the two episodes. The Workbench's feature-compatibility tests basically look to see whether the feature-sets of two episodes are logically inconsistent; it allows, for instance a "rapid increase" episode to be matched with a "steady increase" episode, even if these are the only features noted for the two distinct episodes.

Having found the largest possible chunking within the episode history, the Workbench continues to look for others within the "unchunked" portion of the record. For example, the program will locate a series of two-episode oscillations followed later in the history by (say) three-episode oscillations (the latter will in fact be identified first by the algorithm). The Workbench also recursively looks for "sub-oscillations" within the oscillating patterns themselves; for instance, if a six-episode chunk contains two two-episode chunks at its conclusion (as in the pattern "A B C D C D") then the Workbench will note that each oscillation is itself composed of a smaller oscillating pattern.

C.3 Grouping Zero Crossings into Oscillations

The algorithm for grouping derivative zero-crossings into "chunks" is in fact not as sophisticated as the episode-history grouping algorithm described in the previous section: unlike the previous algorithm, it starts by looking for the smallest possible (two-crossing) repeating pattern, and works its way

upward from that. (The episode-grouping algorithm looks for the *largest* possible repeating chunk and then looks for sub-patterns within each chunk.) This means that complex oscillation structures (eg, consisting of one large "hump" followed by three near-identical smaller ones) will not be classified as such; only the smallest repeating portions of these structures will be found.[1] In practice this has not yet proven to be a liability, but future versions of the Workbench will probably have to incorporate a more sophisticated algorithm analogous to the one described above.

One of the delicate problems associated with grouping derivative zero-crossings is how much information to look at: for instance, are we anticipating that *all* species in the system are oscillating in concentration (a presumably common occurrence if the system has reached a limit cycle), or do we only wish to find repeating patterns for the zero-crossings of individual species? The Workbench employs something of a compromise approach, looking for repeating patterns of a particular species using fairly "stringent" criteria for that species alone or more "lenient" criteria for all the species in the system. In this way, we can find oscillations that occur for a particular species even if certain other portions of the mechanism are not exhibiting oscillations; but if this should occur, our criteria for grouping zero-crossings together are more demanding.

The Workbench's criteria for matching two zero-crossings for some species `X` as part of an oscillation pattern involves checking several possible conditions, any of which can justify a match. In one test, we examine the concentrations of all species: if all species differ in concentration by less than some given percentage (default 12), then these two zero-crossings are deemed matchable. Alternatively, the concentrations for `X` alone could differ by some smaller percentage (default 6), in which case the other concentrations can be ignored; or a greater percentage difference for `[X]` is allowed (default 24) if the two concentrations differ by an absolute value less than five percent of the net range of `[X]` over the course of the run. (This permits, for instance, very low values of `[X]` to differ by a larger percentage as long as those low values are small compared to the overall range of `[X]`.) Yet another possibility is

[1] To continue the parenthesized example: our algorithm would find a series of apparently separated "three-hump" oscillations, rather than a continuous series of complex "four-hump" oscillations.

that all concentrations change over successive chunks by a similar amount (default 5 percent maximum difference between the ratios of successive concentrations). Another is as follows: if these compared crossings appear in the "interior" of a chunk, we will allow them to be matched if the time difference between these crossings and the initial (already-matched) crossings of their respective chunks is within a certain small percentage difference (default 4 percent). (This means that if we have matched two initial crossings of a possible two-crossing chunk, and the second crossings appear exactly one second after the initial crossings in both cases, we will match those second crossings as well.)

The program looks for repeating chunks in this fashion up to a maximum chunk size of 8 (corresponding to a "four-part" oscillation). Again, in practice this has been sufficient for all examples thus far; but a more extensive check may be needed for mechanisms that exhibit more complex oscillation patterns than those already tried.

C.4 Classifying Oscillation Types

In general, in seeking to classify a particular group of linked zero-crossings, we look at the final 80 percent or so of the crossings. This permits us to ignore the possible complications that occur for the first several oscillations in a series; often it is the case that the first oscillation or two has a somewhat different shape than the "typical" portion of the series that follows.

Stable Oscillations

The test for stable oscillations looks for one of several possible conditions. If the oscillations are neither all declining nor all increasing in amplitude, and if the ratio of the net range of amplitudes to the mean of the amplitudes does not exceed 15 percent, the program deems these to be stable oscillations. Alternatively, if the amplitudes are all declining and their successive ratios are increasing, and if the final ratio between successive amplitudes is between 0.95 and 1.05, these are stable oscillations; a similar test is made for all-increasing amplitudes with decreasing ratios. Another possibility is that all

ratios are between 0.999 and 1.001, in which case the oscillations are deemed stable (or, if the ratios are all between 0.92 and 1.08, the oscillations are deemed "probably stable").

Damped/Unstable Oscillations

A series of oscillations is deemed to be an instance of "damped oscillations" if the amplitudes form a decreasing series and if the ratios of successive oscillations meet several additional conditions: a small range (the ratio does not change by more than 8 percent over the observed oscillations), the minimal ratio is sufficiently small (eg, less than 0.99 for a short series of oscillations), and—if the series of oscillations is shorter than 5–the series of ratios is a decreasing series.

Similar tests are used for "unstable oscillations"; here, we test for increasing amplitudes with a series of ratios that meet analogous tests to those described in the previous paragraph (a small range, a sufficiently large maximum).

Possible Chaotic Oscillations

Oscillations are classified as "possibly chaotic" when they do not fit any of the other categories: stable, damped, unstable, or noise (see below). In addition, at least four peaks must have been recorded, and the state space for the system must have a dimension greater than two (as specified by the Poincaré-Bendixson Theorem).

Probable Noise

Occasionally, apparent oscillations are caused by a consistent pattern of tiny round-off errors within the integrator: we might see, for instance, a repeating pattern of "maxima" and "minima" around a steady-state concentration, where the amplitude of the oscillation is some tiny fraction of the average concentration value.

The Workbench classifies an oscillation as "probably noise" if it is either

very flat (the amplitude is less than one two-hundredth of a percent of the mean concentration), or if it is rather flat (the amplitude is less than half a percent of the mean concentration) and brief (fewer than four amplitudes recorded, or fewer than six recorded with no clear pattern of increasing or decreasing amplitudes).

C.5 Feature Summaries

Feature summaries are interpretations of the overall behavior of the simulation based on the information gathered in the episode history and derivative zero-crossing analysis. Typically, the classification decision is based on features found in the final episode, analyses of the last oscillation structure or structures (if any), and numerical values at the end of the simulation.

Steady States

The Workbench uses a number of independent tests, whose results are combined in different combinations, to decide whether the simualtion concluded with a steady state for a given species (and, if so, whether the steady-state appears to have a zero or nonzero concentration value). One test is for a long final episode in the episode history; another is for a final episode with a "steady-state" feature for the given species; yet another possibility—a less compelling piece of evidence—is a final episode with either a "slow decreasse" or "slow increase" feature, which could at least be consistent with an approach toward a final steady state value.

The program also checks the final numerical values of concentration, derivative, and second derivative to estimate the "flatness" of the species' concentration curve at the conclusion of the simulation. If the derivative values are sufficiently small to cause little change over a long time (where "long" is defined through the episode history, as discussed above), then this is taken as corroborating evidence for a final steady state; if the final concentration is low compared to its maximum value, and the final derivative values indicate that the concentration will continue to drop at a slow rate, this is taken as evidence of a "zero-concentration steady state."

Certain oscillation tests are also used in the context of steady-state classification. If the species seems to have undergone a damped oscillation, this is taken as corroborating evidence of a final steady state (in conjunction with several of the other tests); on the other hand, the presence of stable oscillations at the close of the simulation is taken as potential disqualifying evidence of a steady-state classification. (For instance, the "numerical flatness" test alone is sufficient for a designation of "possible steady state" as long as stable oscillations were not detected at the end of the simulation.)

Possible Chaos

The typical feature-summary analysis for oscillations of various types is to look at the classification of the very last oscillation structure recorded. This strategy is problematic for chaotic behavior, because the program may not identify chaotic oscillations as such, but rather as a series of unidentified oscillation-types (see, for instance, the Rössler Band example in the text). Thus, the Workbench employs a slightly more sophisticated test to look for the possibility of chaotic behavior: in the event that multiple oscillations structures were found, and that they occurred in relatively rapid succession, and that all of them were of type "chaotic," "unstable," "uninterpretable," or "noise," and that at least one of them was deemed unstable or chaotic, then we have reason to believe that these linked oscillation structures may form a larger chaotic oscillation.

A few specific parameters and additional facts are worth noting here: again, the program will not attribute chaotic behavior to a system with fewer than three state variables, and again the number of complete periods accounted for by the "chaotic" designation must be large (greater than 10 in the current version). Also, the algorithm for finding possible linked oscillation structures tests for "reasonably similar" periods of the linked elements (within 33 percent of each other), and checks as well that the "shape" (peaks-per-oscillation) are the same and that, when two structures are linked, the later oscillation occurs within a little more than 3 periods after the earlier identified structure (that is, we will not link together two identified oscillation patterns that occur far apart in time, beyond 3.1 periods of the earlier structure).

Oscillations of Various Types

The feature summaries for "standard" oscillation types are generally based on checking two properties: whether the simulation apparently ended while an identified oscillation structure was still in effect, and the type of oscillation identified. Thus, if a stable oscillation structure was identified at the end of the run, the feature summary will contain a "stable oscillation" element. Similar tests exist for damped and unstable oscillations (and for chaotic oscillations as well, in addition the tests listed in the previous subsection).

C.6 Coarse-Grained Zero Crossings

In addition to the analyses of zero-crossing structures described earlier, the Workbench also attempts to look at the zero-crossings in a more "coarse-grained" fashion, looking only at zero-crossings that differ from their neighbors by a large amount. The point of this technique is to deal with numerical records for certain stiff systems in which "noise oscillations" due to numerical roundoff error tend to appear between more noticeable peaks.[2]

We can think of the coarse-graining process as being based on the notion that we can only distinguish points within a concentration-versus-time curve that differ by at least 2.5 percent of the net range of the curve. (For instance, if the concentration of species `X` varies from 0 to 10 over the course of some simulation, then we can say that two concentration values are "indistinguishably close together" if they are within 0.25 of each other.) The zero-crossings are filtered so that the remaining set consists of zero-crossings that occur "distinguishably far apart" from those that occur before and after. Similar algorithms may then be run on this filtered set as were run on the original zero-crossings history: again, we group zero-crossings into stable,

[2]Currently, the coarse-grained analysis is only receiving mild attention—we include the results of coarse-grained analysis in the feature summaries for focus species, but do not use it to generate parameter space graphs. Using a finer time-step and double-precision arithmetic for Gear integration generally renders the numerical records for stiff systems sufficiently tractable so that the "fine-grained" zero-crossing analysis is reliable.

damped, unstable, and chaotic oscillations.[3]

[3]The major difference in this case are that the algorithms for spotting chaotic oscillations are a bit less developed than they are for the "[illegible]" structures.

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 6/8/95

Report # AI-TR-1306

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 249 (250-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☒ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form (2) ☐ Funding Agent Form ☒ Cover Page
☒ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): PAGE FOLLOWING TITLE PAGE

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP	(1-10)
UNNUMBERED TITLE, BLANK, ACK(2), CONT.(4), LIST OF FIG.S(2)	(11-249)
PAGES #'ED 1-239	(250-254)
SCANCONTROL, COVER, SPINE, DOD(2)	(255-255)
TARGETS(0)	

Scanning Agent Signoff:
Date Received: 6/8/95 Date Scanned: 6/14/95 Date Returned: 6/15/95

Scanning Agent Signature: Michael W. Cook

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public report... estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1991	technical report

4. TITLE AND SUBTITLE
The Kineticist's Workbench: Combining Symbolic and Numerical Methods in the Simulation of Chemical Reaction Mechanisms

5. FUNDING NUMBERS
N0014-89-J-3202

6. AUTHOR(S)
Michael A. Eisenberg

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

8. PERFORMING ORGANIZATION REPORT NUMBER
AI-TR 1306

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Information Systems
Arlington, Virginia 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AD-A259439

11. SUPPLEMENTARY NOTES
None

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution of this document is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Kineticist's Workbench is a program that expands the expressiveness of computer simulation: it combines symbolic and numerical techniques in simulating a particular class of complex systems—chemical reaction mechanisms. The Workbench assists chemists by predicting, generating, and interpreting numerical data. Prior to simulation, it analyzes a given mechanism to predict that mechanism's behavior; it then simulates the mechanism numerically; and afterward, it interprets and summarizes the data that it has generated. In performing these tasks, the Workbench brings to bear a wide variety of techniques: graph-theoretic algorithms (for the analysis of mechanisms), traditional numerical simulation methods, and algorithms that examine the

(continued on back)

14. SUBJECT TERMS (key words)
chemical kinetics
symbolic/numerical simulation

15. NUMBER OF PAGES
248

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Block 13 continued:

simulation results and reinterpret them in qualitative terms. Moreover, the Workbench can use symbolic procedures to help guide or simplify the task of numerical simulation; and it can sometimes use its summary of numerical results to suggest additional numerical analysis. Thus, it serves as a prototype for a new class of scientific computational tools—tools that provide symbiotic collaborations between qualitative and quantitative methods.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 6/14/1995

M.I.T. Libraries
Document Services