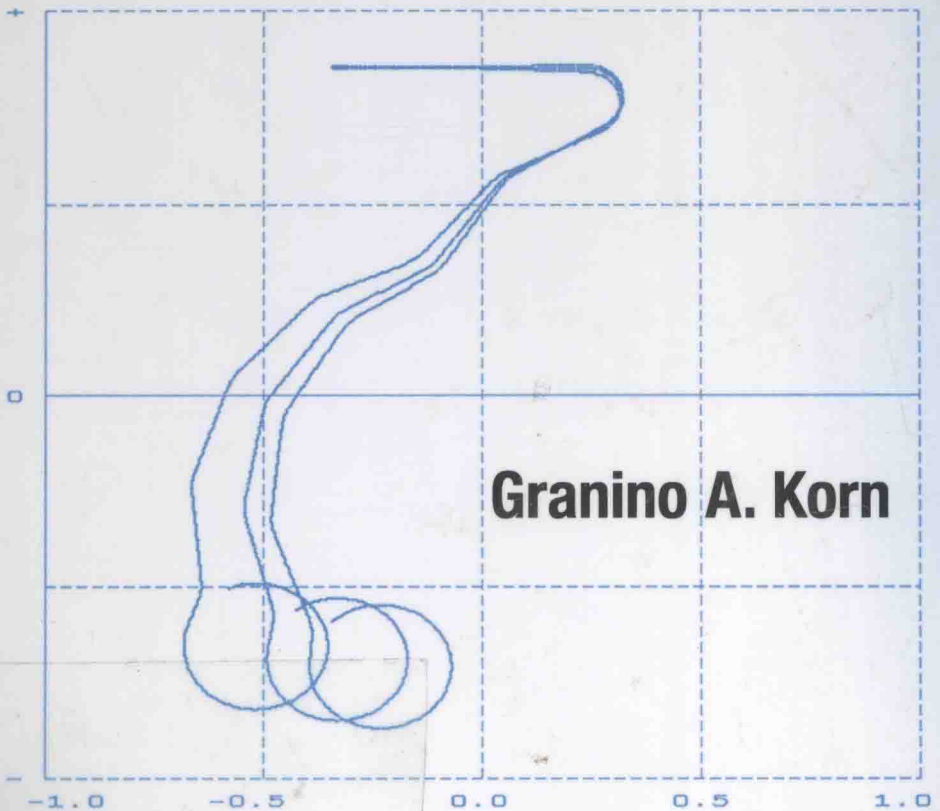


SECOND EDITION

# Advanced Dynamic-System Simulation

*Model Replication and Monte Carlo Studies*



**Granino A. Korn**



INCLUDES CD-ROM

**WILEY**

SECOND EDITION

---

***ADVANCED  
DYNAMIC-SYSTEM  
SIMULATION***

Model Replication and  
Monte Carlo Studies

**GRANINO A. KORN**

*University of Arizona*



 **WILEY**

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2013 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at [www.copyright.com](http://www.copyright.com). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

**Limit of Liability/Disclaimer of Warranty:** While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at [www.wiley.com](http://www.wiley.com).

***Library of Congress Cataloging-in-Publication Data:***

Korn, Granino A. (Granino Arthur), 1922–

Advanced dynamic-system simulation : model replication and Monte Carlo studies / by Granino A.

Korn. – Second edition.

pages cm

Includes bibliographical references.

ISBN 978-1-118-39735-0 (hardback)

1. System analysis—Simulation methods. 2. Open source software. 3. Computer software—Development. I. Title.

QA402.K665 2013

003'.85—dc23

2012034771

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

---

# PREFACE

Simulation is experimentation with models. In this book we describe efficient interactive computer programs that model dynamic systems such as control systems, aerospace vehicles, and biological systems. Simulation studies can involve many hundreds of model changes, so programs must be fast and user-friendly.

For hands-on experiments with each program example, the book CD now includes industrial-strength open-source simulation software for both Windows™ and Linux, not just toy demonstration programs. The Desire modeling/simulation program implements very fast and respectably large simulations on personal computers. Runtime-compiled programs display results immediately to permit true interactive modeling.

A readable mathematical notation, for example,

$$\mathbf{x} = 23.4 \quad | \quad \alpha = 0$$

$$d/dt \mathbf{x} = -\mathbf{x} * \cos(\mathbf{w} * t) + 2.22 * \mathbf{a} * \mathbf{x}$$

$$\text{Vector } \mathbf{y} = \mathbf{A} * \mathbf{x} + \mathbf{B} * \mathbf{u}$$

lets readers try different parameter values without learning details of the programming language. Note that one can read ebook pages and run live simulations on the same computer display.

In Chapter 1 we introduce our subject with a few familiar differential-equation models and a small guided-missile simulation. The remainder of the book presents more advanced topics; most of our example programs were rewritten to clarify the modeling technique and to increase computing speed.

Chapter 2 begins with a newly revised systematic procedure for programming *difference equations* and applies this to model plants with digital controllers. We then discuss *limiters and switches* and model useful devices such as track/hold circuits, trigger circuits, and signal generators with simple difference equations. Last but not least, we propose a simplified technique for numerical integration of switched variables.

Advanced simulation programs must handle differential and difference equations with vector and matrix assignments. In Chapter 3 we introduce runtime *vector compilation*. This speeds up conventional vector and matrix operations, but more significantly, personal computers can now implement model *replication (vectorization)*, a technique originally developed for supercomputers. A single vector-model run replaces hundreds or thousands of conventional simulation runs. Chapter 3 also demonstrates the convenience of user-defined *submodels*.

In the remaining chapters we describe applications of vectorization. In Chapter 4 we discuss parameter-influence studies and introduce *vectorized statistics*

*computation*, including rapid estimation of probability densities. We then introduce *Monte Carlo simulation of random processes*. In Chapter 5 we apply Monte Carlo simulation to several real engineering systems. Vectorization lets us study *time histories of random-process statistics*. An inexpensive 3-GHz personal computer running under 64-bit Linux can exercise over 1000 random-input control-system models in 1 second.

In Chapters 6 and 7 we demonstrate *vector models of neural networks*; our simple vector notation has been particularly useful for short courses on neural networks. In Chapter 6 we apply backpropagation, functional-link, and radial-basis-function networks to classical regression and pattern-classification problems and describe several competitive-learning schemes. In the newly added Chapter 7 we turn to *dynamic neural networks* for prediction, pattern classification, and model matching. The chapter includes a new method for online prediction and simplified programs for recurrent networks.

Chapter 8 deals with vectorized programs for *fuzzy-set controllers*, *partial differential equations*, and *agroecological models replicated at over 1000 points of a landscape map*. The Appendix gets a small selection of reference material out of the way of the main text.

The writer would like to express his sincere thanks to Professor M. Main of the University of Colorado for his assistance with Windows graphics, to Dr. R. Wieland of the Leibniz Center for Agricultural Landscape Research (ZALF) for much good advice, and above all to Theresa M. Korn for her consistent help with this and many other projects.

GRANINO A. KORN

*Wenatchee, Washington*

---

# CONTENTS

PREFACE

xiii

---

## CHAPTER 1 DYNAMIC-SYSTEM MODELS AND SIMULATION

1

### SIMULATION IS EXPERIMENTATION WITH MODELS 1

- 1-1 Simulation and Computer Programs 1
- 1-2 Dynamic-System Models 2
  - (a) Difference-Equation Models 2
  - (b) Differential-Equation Models 2
  - (c) Discussion 3
- 1-3 Experiment Protocols Define Simulation Studies 3
- 1-4 Simulation Software 4
- 1-5 Fast Simulation Program for Interactive Modeling 5

### ANATOMY OF A SIMULATION RUN 8

- 1-6 Dynamic-System Time Histories Are Sampled Periodically 8
- 1-7 Numerical Integration 10
  - (a) Euler Integration 10
  - (b) Improved Integration Rules 10
- 1-8 Sampling Times and Integration Steps 11
- 1-9 Sorting Defined-Variable Assignments 12

### SIMPLE APPLICATION PROGRAMS 12

- 1-10 Oscillators and Computer Displays 12
  - (a) Linear Oscillator 12
  - (b) Nonlinear Oscillator: Duffing's Differential Equation 14
- 1-11 Space-Vehicle Orbit Simulation with Variable-Step Integration 15
- 1-12 Population-Dynamics Model 17
- 1-13 Splicing Multiple Simulation Runs: Billiard-Ball Simulation 17

### INTRODUCTION TO CONTROL-SYSTEM SIMULATION 21

- 1-14 Electrical Servomechanism with Motor-Field Delay and Saturation 21
- 1-15 Control-System Frequency Response 23
- 1-16 Simulation of a Simple Guided Missile 24
  - (a) Guided Torpedo 24
  - (b) Complete Torpedo-Simulation Program 26

### STOP AND LOOK 28

- 1-17 Simulation in the Real World: A Word of Caution 28

References 29

**CHAPTER 2** *MODELS WITH DIFFERENCE EQUATIONS, LIMITERS, AND SWITCHES*

31

---

SAMPLED-DATA SYSTEMS AND DIFFERENCE EQUATIONS	31
2-1 Sampled-Data Difference-Equation Systems	31
(a) Introduction	31
(b) Difference Equations	31
(c) A Minefield of Possible Errors	32
2-2 Solving Systems of First-Order Difference Equations	32
(a) General Difference-Equation Model	32
(b) Simple Recurrence Relations	33
2-3 Models Combining Differential Equations and Sampled-Data Operations	35
2-4 Simple Example	35
2-5 Initializing and Resetting Sampled-Data Variables	35
TWO MIXED CONTINUOUS/SAMPLED-DATA SYSTEMS	37
2-6 Guided Torpedo with Digital Control	37
2-7 Simulation of a Plant with a Digital PID Controller	37
DYNAMIC-SYSTEM MODELS WITH LIMITERS AND SWITCHES	40
2-8 Limiters, Switches, and Comparators	40
(a) Limiter Functions	40
(b) Switching Functions and Comparators	42
2-9 Integration of Switch and Limiter Outputs, Event Prediction, and Display Problems	43
2-10 Using Sampled-Data Assignments	44
2-11 Using the <b>step</b> Operator and Heuristic Integration-Step Control	44
2-12 Example: Simulation of a Bang-Bang Servomechanism	45
2-13 Limiters, Absolute Values, and Maximum/Minimum Selection	46
2-14 Output-Limited Integration	47
2-15 Modeling Signal Quantization	48
EFFICIENT DEVICE MODELS USING RECURSIVE ASSIGNMENTS	48
2-16 Recursive Switching and Limiter Operations	48
2-17 Track/Hold Simulation	49
2-18 Maximum-Value and Minimum-Value Holding	50
2-19 Simple Backlash and Hysteresis Models	51
2-20 Comparator with Hysteresis (Schmitt Trigger)	52
2-21 Signal Generators and Signal Modulation	53
References	55

**CHAPTER 3** *FAST VECTOR-MATRIX OPERATIONS AND SUBMODELS*

57

---

ARRAYS, VECTORS, AND MATRICES	57
3-1 Arrays and Subscripted Variables	57
(a) Improved Modeling	57
(b) Array Declarations, Vectors, and Matrices	57
(c) State-Variable Declarations	58
3-2 Vector and Matrices in Experiment Protocols	58
3-3 Time-History Arrays	58
VECTORS AND MODEL REPLICATION	59
3-4 Vector Operations in DYNAMIC Program Segments: The Vectorizing Compiler	59

(a)	Vector Assignments and Vector Expressions	59
(b)	Vector Differential Equations	60
(c)	Vector Sampled-Data Assignments and Difference Equations	60
3-5	Matrix-Vector Products in Vector Expressions	61
(a)	Definition	61
(b)	Simple Example: Resonating Oscillators	61
3-6	Index-Shift Operation	63
(a)	Definition	63
(b)	Preview of Significant Applications	63
3-7	Sorting Vector and Subscripted-Variable Assignments	64
3-8	Replication of Dynamic-System Models	64
	<b>MORE VECTOR OPERATIONS</b>	<b>65</b>
3-9	Sums, <b>DOT</b> Products, and Vector Norms	65
(a)	Sums and <b>DOT</b> Products	65
(b)	Euclidean, Taxicab, and Hamming Norms	65
3-10	Maximum/Minimum Selection and Masking	66
(a)	Maximum/Minimum Selection	66
(b)	Masking Vector Expressions	66
	<b>VECTOR EQUIVALENCE DECLARATIONS SIMPLIFY MODELS</b>	<b>67</b>
3-11	Subvectors	67
3-12	Matrix-Vector Equivalence	67
	<b>MATRIX OPERATIONS IN DYNAMIC-SYSTEM MODELS</b>	<b>67</b>
3-13	Simple Matrix Assignments	67
3-14	Two-Dimensional Model Replication	68
(a)	Matrix Expressions and <b>DOT</b> Products	68
(b)	Matrix Differential Equations	68
(c)	Matrix Difference Equations	69
	<b>VECTORS IN PHYSICS AND CONTROL-SYSTEM PROBLEMS</b>	<b>69</b>
3-15	Vectors in Physics Problems	69
3-16	Vector Model of a Nuclear Reactor	69
3-17	Linear Transformations and Rotation Matrices	70
3-18	State-Equation Models of Linear Control Systems	72
	<b>USER-DEFINED FUNCTIONS AND SUBMODELS</b>	<b>72</b>
3-19	Introduction	72
3-20	User-Defined Functions	72
3-21	Submodel Declaration and Invocation	73
3-22	Dealing with Sampled-Data Assignments, Limiters, and Switches	75
	References	75

## **CHAPTER 4** *EFFICIENT PARAMETER-INFLUENCE STUDIES AND STATISTICS COMPUTATION*

77

---

	<b>MODEL REPLICATION SIMPLIFIES PARAMETER-INFLUENCE STUDIES</b>	<b>77</b>
4-1	Exploring the Effects of Parameter Changes	77
4-2	Repeated Simulation Runs Versus Model Replication	78
(a)	Simple Repeated-Run Study	78
(b)	Model Replication (Vectorization)	78
4-3	Programming Parameter-Influence Studies	80
(a)	Measures of System Performance	80
(b)	Program Design	81



**viii** CONTENTS

- (c) Two-Dimensional Model Replication 81
- (d) Cross-Plotting Results 82
- (e) Maximum/Minimum Selection 83
- (f) Iterative Parameter Optimization 83
- STATISTICS 84
- 4-4 Random Data and Statistics 84
- 4-5 Sample Averages and Statistical Relative Frequencies 85
- COMPUTING STATISTICS BY VECTOR AVERAGING 85
- 4-6 Fast Computation of Sample Averages 85
- 4-7 Fast Probability Estimation 86
- 4-8 Fast Probability-Density Estimation 86
  - (a) Simple Probability-Density Estimate 86
  - (b) Triangle and Parzen Windows 87
  - (c) Computation and Display of Parzen-Window Estimates 88
- 4-9 Sample-Range Estimation 90
- REPLICATED AVERAGES GENERATE SAMPLING DISTRIBUTIONS 91
- 4-10 Computing Statistics by Time Averaging 91
- 4-11 Sample Replication and Sampling-Distribution Statistics 91
  - (a) Introduction 91
  - (b) Demonstrations of Empirical Laws of Large Numbers 93
  - (c) Counterexample: Fat-Tailed Distribution 95
- RANDOM-PROCESS SIMULATION 95
- 4-12 Random Processes and Monte Carlo Simulation 95
- 4-13 Modeling Random Parameters and Random Initial Values 97
- 4-14 Sampled-Data Random Processes 97
- 4-15 "Continuous" Random Processes 98
  - (a) Modeling Continuous Noise 98
  - (b) Continuous Time Averaging 99
  - (c) Correlation Functions and Spectral Densities 100
- 4-16 Problems with Simulated Noise 100
- SIMPLE MONTE CARLO EXPERIMENTS 100
- 4-17 Introduction 100
- 4-18 Gambling Returns 100
- 4-19 Vectorized Monte Carlo Study of a Continuous Random Walk 102
- References 106

**CHAPTER 5** MONTE CARLO SIMULATION OF REAL DYNAMIC SYSTEMS

**109**

---

- INTRODUCTION 109
- 5-1 Survey 109
- REPEATED-RUN MONTE CARLO SIMULATION 109
- 5-2 End-of-Run Statistics for Repeated Simulation Runs 109
- 5-3 Example: Effects of Gun-Elevation Errors on a 1776 Cannonball Trajectory 110
- 5-4 Sequential Monte Carlo Simulation 113
- VECTORIZED MONTE CARLO SIMULATION 113
- 5-5 Vectorized Monte Carlo Simulation of the 1776 Cannon Shot 113
- 5-6 Combined Vectorized and Repeated-Run Monte Carlo Simulation 115
- 5-7 Interactive Monte Carlo Simulation: Computing Runtime Histories of Statistics with DYNAMIC-Segment **DOT** Operations 115

5-8	Example: Torpedo Trajectory Dispersion	117
SIMULATION OF NOISY CONTROL SYSTEMS 119		
5-9	Monte Carlo Simulation of a Nonlinear Servomechanism: A Noise-Input Test	119
5-10	Monte Carlo Study of Control-System Errors Caused by Noise	121
ADDITIONAL TOPICS 123		
5-11	Monte Carlo Optimization	123
5-12	Convenient Heuristic Method for Testing Pseudorandom Noise	123
5-13	Alternative to Monte Carlo Simulation	123
	(a) Introduction	123
	(b) Dynamic Systems with Random Perturbations	123
	(c) Mean-Square Errors in Linearized Systems	124
References 125		

## CHAPTER 6 VECTOR MODELS OF NEURAL NETWORKS

127

---

ARTIFICIAL NEURAL NETWORKS 127		
6-1	Introduction	127
6-2	Artificial Neural Networks	127
6-3	Static Neural Networks: Training, Validation, and Applications	128
6-4	Dynamic Neural Networks	129
SIMPLE VECTOR ASSIGNMENTS MODEL NEURON LAYERS 130		
6-5	Neuron-Layer Declarations and Neuron Operations	130
6-6	Neuron-Layer Concatenation Simplifies Bias Inputs	130
6-7	Normalizing and Contrast-Enhancing Layers	131
	(a) Pattern Normalization	131
	(b) Contrast Enhancement: Softmax and Thresholding	131
6-8	Multilayer Networks	132
6-9	Exercising a Neural-Network Model	132
	(a) Computing Successive Neuron-Layer Outputs	132
	(b) Input from Pattern-Row Matrices	133
	(c) Input from Text Files and Spreadsheets	133
SUPERVISED TRAINING FOR REGRESSION 134		
6-10	Mean-Square Regression	134
	(a) Problem Statement	134
	(b) Linear Mean-Square Regression and the Delta Rule	135
	(c) Nonlinear Neuron Layers and Activation-Function Derivatives	136
	(d) Error-Measure Display	136
6-11	Backpropagation Networks	137
	(a) The Generalized Delta Rule	137
	(b) Momentum Learning	139
	(c) Simple Example	139
	(d) The Classical XOR Problem and Other Examples	140
MORE NEURAL-NETWORK MODELS 140		
6-12	Functional-Link Networks	140
6-13	Radial-Basis-Function Networks	142
	(a) Basis-Function Expansion and Linear Optimization	142
	(b) Radial Basis Functions	143
6-14	Neural-Network Submodels	145
PATTERN CLASSIFICATION 146		
6-15	Introduction	146

## X CONTENTS

6-16	Classifier Input from Files	147
6-17	Classifier Networks	147
	(a) Simple Linear Classifiers	147
	(b) Softmax Classifiers	148
	(c) Backpropagation Classifiers	148
	(d) Functional-Link Classifiers	149
	(e) Other Classifiers	149
6-18	Examples	149
	(a) Classification Using an Empirical Database: Fisher's Iris Problem	149
	(b) Image-Pattern Recognition and Associative Memory	151
PATTERN SIMPLIFICATION 155		
6-19	Pattern Centering	155
6-20	Feature Reduction	156
	(a) Bottleneck Layers and Encoders	156
	(b) Principal Components	156
NETWORK-TRAINING PROBLEMS 157		
6-21	Learning-Rate Adjustment	157
6-22	Overfitting and Generalization	157
	(a) Introduction	157
	(b) Adding Noise	158
	(c) Early Stopping	158
	(d) Regularization	159
6-23	Beyond Simple Gradient Descent	159
UNSUPERVISED COMPETITIVE-LAYER CLASSIFIERS 159		
6-24	Template-Pattern Matching and the <b>CLEARN</b> Operation	159
	(a) Template Patterns and Template Matrix	159
	(b) Matching Known Template Patterns	160
	(c) Template-Pattern Training	160
	(d) Correlation Training	162
6-25	Learning with Conscience	163
6-26	Competitive-Learning Experiments	164
	(a) Pattern Classification	164
	(b) Vector Quantization	164
6-27	Simplified Adaptive-Resonance Emulation	165
SUPERVISED COMPETITIVE LEARNING 167		
6-28	The LVQ Algorithm for Two-Way Classification	167
6-29	Counterpropagation Networks	167
EXAMPLES OF <b>CLEARN</b> CLASSIFIERS 168		
6-30	Recognition of Known Patterns	168
	(a) Image Recognition	168
	(b) Fast Solution of the Spiral Benchmark Problem	169
6-31	Learning Unknown Patterns	173
References 174		

## CHAPTER 7 DYNAMIC NEURAL NETWORKS

177

### INTRODUCTION 177

7-1	Dynamic Versus Static Neural Networks	177
7-2	Applications of Dynamic Neural Networks	177
7-3	Simulations Combining Neural Networks and Differential-Equation Models	178

NEURAL NETWORKS WITH DELAY-LINE INPUT	178
7-4 Introduction	178
7-5 The Delay-Line Model	180
7-6 Delay-Line-Input Networks	180
(a) Linear Combiners	180
(b) One-Layer Nonlinear Network	181
(c) Functional-Link Network	181
(d) Backpropagation Network with Delay-Line Input	182
7-7 Using Gamma Delay Lines	182
STATIC NEURAL NETWORKS USED AS DYNAMIC NETWORKS	183
7-8 Introduction	183
7-9 Simple Backpropagation Networks	184
RECURRENT NEURAL NETWORKS	185
7-10 Layer-Feedback Networks	185
7-11 Simplified Recurrent-Network Models Combine Context and Input Layers	185
(a) Conventional Model of a Jordan Network	185
(b) Simplified Jordan-Network Model	186
(c) Simplified Models for Other Feedback Networks	187
7-12 Neural Networks with Feedback Delay Lines	187
(a) Delay-Line Feedback	187
(b) Neural Networks with Both Input and Feedback Delay Lines	188
7-13 Teacher Forcing	189
PREDICTOR NETWORKS	189
7-14 Off-Line Predictor Training	189
(a) Off-Line Prediction Using Stored Time Series	189
(b) Off-Line Training System for Online Predictors	189
(c) Example: Simple Linear Predictor	190
7-15 Online Training for True Online Prediction	192
7-16 Chaotic Time Series for Prediction Experiments	192
7-17 Gallery of Predictor Networks	193
OTHER APPLICATIONS OF DYNAMIC NETWORKS	199
7-18 Temporal-Pattern Recognition: Regression and Classification	199
7-19 Model Matching	201
(a) Introduction	201
(b) Example: Program for Matching Narendra's Plant Model	201
MISCELLANEOUS TOPICS	204
7-20 Biological-Network Software	204
References	204

---

**CHAPTER 8 MORE APPLICATIONS OF VECTOR MODELS**
**207**

VECTORIZED SIMULATION WITH LOGARITHMIC PLOTS	207
8-1 The EUROSIM No. 1 Benchmark Problem	207
8-2 Vectorized Simulation with Logarithmic Plots	207
MODELING FUZZY-LOGIC FUNCTION GENERATORS	209
8-3 Rule Tables Specify Heuristic Functions	209
8-4 Fuzzy-Set Logic	210
(a) Fuzzy Sets and Membership Functions	210
(b) Fuzzy Intersections and Unions	210
(c) Joint Membership Functions	213
(d) Normalized Fuzzy-Set Partitions	213

**xii** CONTENTS

8-5 Fuzzy-Set Rule Tables and Function Generators 214  
8-6 Simplified Function Generation with Fuzzy Basis Functions 214  
8-7 Vector Models of Fuzzy-Set Partitions 215  
    (a) Gaussian Bumps: Effects of Normalization 215  
    (b) Triangle Functions 215  
    (c) Smooth Fuzzy-Basis Functions 216  
8-8 Vector Models for Multidimensional Fuzzy-Set Partitions 216  
8-9 Example: Fuzzy-Logic Control of a Servomechanism 217  
    (a) Problem Statement 217  
    (b) Experiment Protocol and Rule Table 217  
    (c) DYNAMIC Program Segment and Results 220  
PARTIAL DIFFERENTIAL EQUATIONS 221  
8-10 Method of Lines 221  
8-11 Vectorized Method of Lines 221  
    (a) Introduction 221  
    (b) Using Differentiation Operators 221  
    (c) Numerical Problems 224  
8-12 Heat-Conduction Equation in Cylindrical Coordinates 225  
8-13 Generalizations 225  
8-14 Simple Heat-Exchanger Model 227  
FOURIER ANALYSIS AND LINEAR-SYSTEM DYNAMICS 229  
8-15 Introduction 229  
8-16 Function-Table Lookup and Interpolation 230  
8-17 Fast-Fourier-Transform Operations 230  
8-18 Impulse and Frequency Response of a Linear Servomechanism 231  
8-19 Compact Vector Models of Linear Dynamic Systems 232  
    (a) Using the Index-Shift Operation with Analog Integration 232  
    (b) Linear Sampled-Data Systems 235  
    (c) Example: Digital Comb Filter 236  
REPLICATION OF AGROECOLOGICAL MODELS ON MAP GRIDS 237  
8-20 Geographical Information System 237  
8-21 Modeling the Evolution of Landscape Features 239  
8-22 Matrix Operations on a Map Grid 239  
References 242

*APPENDIX: ADDITIONAL REFERENCE MATERIAL* **245**

---

A-1 Example of a Radial-Basis-Function Network 245  
A-2 Fuzzy-Basis-Function Network 245  
References 248

*USING THE BOOK CD* **251**

*INDEX* **253**

---

# DYNAMIC-SYSTEM MODELS AND SIMULATION

## SIMULATION IS EXPERIMENTATION WITH MODELS

### 1-1. Simulation and Computer Programs

Simulation is experimentation with models. For system design, research, and education, simulations must not only construct and modify many different models but also store and access a large volume of results. That is practical only with models programmed on computers [1, 2].

In this book we model changes of system variables with time; we represent physical time by the simulation time variable  $t$ . Our models then attempt to predict different time histories  $y_1 = y_1(t)$ ,  $y_2 = y_2(t)$ , ... of system variables such as velocity, voltage, and biomass. *Static models* simply relate values of system variables  $x(t)$ ,  $y(t)$ , ... at the same time  $t$ ; a gas pressure  $P(t)$ , for instance, might be a function  $P = aT$  of the slowly changing temperature  $T(t)$ .

*Dynamic-system models* predict values of model-system *state variables*  $x_1(t)$ ,  $x_2(t)$ , ... by relating them to past states  $[x_1(t), x_2(t), \dots]$  (Sec. 1-2). Computer simulation of such systems was applied first in the aerospace industry. Simulation is now indispensable not only in all engineering disciplines, but also in biology, medicine, and agroecology. At the same time, discrete-event simulation gained importance for business and military planning.

Simulation is most effective when it is combined with mathematical analyses. But simulation results often provide insight and suggest useful decisions where exact analysis is difficult or impossible. This was true for many early control-system optimizations. As another example, Monte Carlo simulations simply measure statistics over repeated experiments to solve problems too complicated for explicit probability-theory analysis. All simulation results must eventually be validated by real experiments, just like analytical results.

Computer simulations can be speeded up or slowed down for the experimenter's convenience. One can simulate a flight to Mars or to Alpha Centauri in one second. Periodic clock interrupts synchronizing suitably scaled simulations with real time permit "hardware in the loop": One can "fly" a real autopilot—or a human pilot—on a tilt table controlled by computer flight simulation. In this book we are interested

in very fast simulation because we need to study many different model changes very quickly. Specifically, we would like to

- *enter and edit programs* in convenient editor windows.
- use *typed or graphical-interface commands* to start, stop, and pause simulations, to select displays, and to make parameter changes. Displays of simulation results ought to *appear immediately* to provide an intuitive “feel” for the effects of model changes (*interactive modeling*).
- *program* systematic parameter-optimization studies and produce cross-plots and statistics.

## 1-2. Dynamic-System Models

### (a) *Difference-Equation Models*<sup>1</sup>

The simplest way to relate present values  $\mathbf{x}(t)$  and past values  $\mathbf{x}(t - \Delta t)$  of a *state variable*  $\mathbf{x} = \mathbf{x}(t)$  is a difference equation such as the simple recurrence

$$\mathbf{x}(t) = \mathbf{F}[\mathbf{x}(t), \mathbf{x}(t - \Delta t)]$$

More general difference-equation models relate several state variables and their past values. In Chapter 2 we discuss such models in detail.

### (b) *Differential-Equation Models*

Much of classical physics and engineering is based on *differential-equation models* that relate delayed interactions of continuous *differential-equation state variables*  $\mathbf{x}_1(t), \mathbf{x}_2(t), \dots$  with first-order ordinary differential equations (*state equations*)<sup>2</sup>

$$(d/dt) \mathbf{x}_i = \mathbf{f}_i(t; \mathbf{x}_1, \mathbf{x}_2, \dots; \mathbf{y}_1, \mathbf{y}_2, \dots; \mathbf{a}_1, \mathbf{a}_2, \dots) \quad (i = 1, 2, \dots) \quad (1-1a)$$

Here  $t$  again represents the time, and the quantities

$$\mathbf{y}_j = \mathbf{g}_j(t; \mathbf{x}_1, \mathbf{x}_2, \dots; \mathbf{y}_1, \mathbf{y}_2, \dots; \mathbf{b}_1, \mathbf{b}_2, \dots) \quad (j = 1, 2, \dots) \quad (1-1b)$$

are *defined variables*.  $\mathbf{a}_1, \mathbf{a}_2, \dots$  and  $\mathbf{b}_1, \mathbf{b}_2, \dots$  are constant *model parameters*.

A computer-implemented *simulation run* exercises such a model by solving the state-equation system (1-1) to produce time histories of the system variables  $\mathbf{x}_i = \mathbf{x}_i(t)$  and  $\mathbf{y}_j = \mathbf{y}_j(t)$  for  $t = t_0$  to  $t = t_0 + T_{MAX}$ . An *integration routine* increments the model time  $t$  and integrates the derivatives (1-1a) to produce successive values of  $\mathbf{x}_i(t)$  (Sec. 1-7), starting with given initial values  $\mathbf{x}_i = \mathbf{x}_i(t_0)$ .

<sup>1</sup>We refer to recursive relations in general as difference equations, whereas some authors reserve this term for relations formulated in terms of explicit finite differences [11].

<sup>2</sup>We reduce higher-order differential equations to first-order systems by introducing derivatives as extra state variables. Thus,  $d^2\mathbf{x}/dt^2 = -k\mathbf{x}$  becomes

$$d\mathbf{x}/dt = \mathbf{x}\dot{\quad} \quad d\mathbf{x}\dot{\quad}/dt = -k\mathbf{x}$$

(see also Sec. 1-10).

Each state variable  $\mathbf{x}_i$  is a model output. There are three types of defined variables  $\mathbf{y}_j$ :

1. model inputs (specified functions of the time  $\mathbf{t}$ ),
2. model outputs, and
3. intermediate results needed to compute the derivatives  $\mathbf{f}_i$ .

The defined-variable assignments (1-1b) must be sorted into a procedure that derives updated values for all  $\mathbf{y}_j$  from current values of the state variables  $\mathbf{x}_i$ , already computed  $\mathbf{y}_j$  values, and/or  $\mathbf{t}$  without “algebraic loops” (Sec. 1-9).

Some dynamic systems (e.g., systems involving linkages in automotive engineering and robotics) are modeled with differential equations that cannot be solved explicitly for state-variable derivatives as in Eq. (1-1a). Simulation then requires solution of algebraic equations at each integration step. Such *differential-algebraic-equation systems* are not treated in this book. References 6 to 11 describe suitable mathematical methods and special software.

### (c) Discussion

Much of classical physics (Newtonian dynamics, electrical-circuit theory, chemical reactions) uses differential equations. As a result, most legacy simulation programs are basically differential-equation solvers and relegate difference equations to accessory “procedural” program segments. Modern engineering systems, though, often involve digital controllers and thus sampled-data operations that implement difference equations. In this book we introduce a program package specifically designed to handle such problems. We start with differential-equation problems in Chapter 1 and go on to difference equations and mixed continuous/sampled-data models in Chapter 2.

## 1-3. Experiment Protocols Define Simulation Studies

Effective computer simulation is not simply a matter of programming model equations. It must also be truly convenient to modify models and to try many different experiments (see also Sec. 1-5). In addition to program segments that list model equations such as those in Sec. 1-2, every simulation needs an *experiment-protocol program* that sets and changes initial conditions and parameters, calls differential-equation-solving simulation runs, and displays or lists solutions.

A simple experiment protocol implements a sequence of successive commands:  
say

```

a = 20.0 | b = -3.35 (set parameter values)
x = 12.0 (set the initial value of x)
drun (make a differential-equation-solving simulation run)
reset (reset initial values)
a = 20.1 (change model parameters)
b = b - 2.2
drun (try another run)
.....

```



Each **drun** command calls a new simulation run. The command **reset** resets initial conditions for new runs.

A *command interpreter* executes typed commands immediately. Users can inspect the solution output after each simulation run and then enter new commands for another run. Command-mode operation permits interactive programming and program debugging [2].

*Graphical-user-interface (GUI) simulation programs* replace typed commands with windows for entering model parameters and menus and/or buttons for executing such commands as **run** and **reset** using mouse clicks. This is convenient for special-purpose simulation programs with simple experiment protocols. Typed and programmed commands entered in a console window (command window) permit a much wider choice of operations.

A programmed *simulation study* combines experiment-protocol commands into a stored program called an *experiment-protocol script*. Such a program can branch and loop to call repeated simulation runs (e.g., for parameter optimization or statistical studies). Proper experiment-protocol scripts require a full-fledged computer language with functions, procedures, program loops, conditional execution, and file operations.

Simulation studies can involve many model and parameter changes, so program execution must be prompt and fast. We can *interpret* experiment-protocol scripts. But “dynamic” program segments that implement simulation runs update system variables hundreds or thousands of times. Such time-critical operations must be *compiled*.<sup>3</sup>

## 1-4. Simulation Software

*Equation-oriented simulation programs* such as ACSL<sup>TM</sup> accept model equations in a more or less human-readable notation, sort defined-variable assignments as needed, and feed the sorted equations to a Fortran or C compiler [1]. Berkeley Madonna and Desire (see below) have runtime equation-language compilers and execute immediately. *Block-diagram interpreters* (e.g., Simulink<sup>TM</sup> and the free open-source Scicoslab program) let users compose block-diagram models on the display screen. Such programs execute interpreted simulation runs immediately but relatively slowly. To improve computing speed, most block-diagram interpreters admit precompiled equation-language blocks for complicated expressions, and production runs are sometimes translated into C for faster execution. Alternatively, ACSL, Easy5<sup>TM</sup>, and Berkeley Madonna have *block-diagram preprocessors* for compiled simulation programs. Differential-algebraic (DAE) models need substantially more complicated software, preferably using the Modelica Language [3–6]. Dynasim<sup>TM</sup> and Maplesim<sup>TM</sup> are examples.

<sup>3</sup>*Interpreter* programs translate individual commands one-by-one into the computer’s machine language. *Compilers* speed program execution by translating complete program segments.