

DISCRETE-EVENT SIMULATION:  
A FIRST COURSE

*Lawrence Leemis*  
*Professor of Mathematics*  
*The College of William & Mary*  
*Williamsburg, VA 23187-8795*  
*757-221-2034*  
`leemis@math.wm.edu`

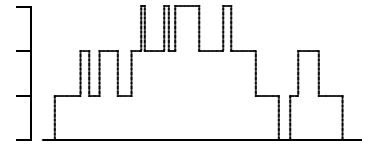
*Steve Park*  
*Professor of Computer Science*  
*The College of William & Mary*

© December 1994  
Revisions 9-1996, 9-1997, 9-1998,  
1-1999, 9-1999, 1-2000, 8-2003, 6-2004  
Current Revision, December 2004

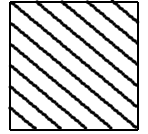
*Blank Page*

# Brief Contents

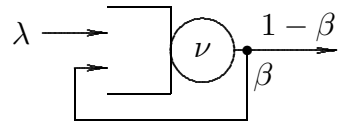
1. Models 1



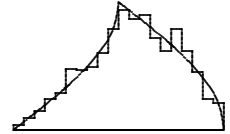
2. Random Number Generation 37



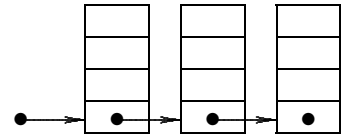
3. Discrete-Event Simulation 100



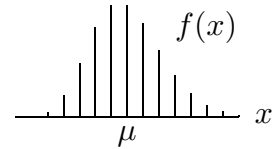
4. Statistics 131



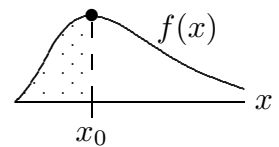
5. Next-Event Simulation 185



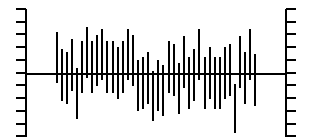
6. Discrete Random Variables 223



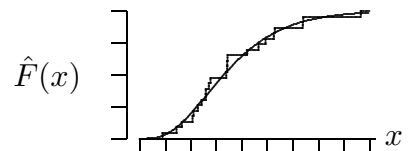
7. Continuous Random Variables 279



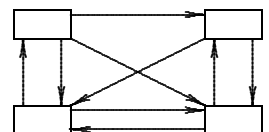
8. Output Analysis 346



9. Input Modeling 396



10. Projects 438



# Contents

<b>Chapter 1. Models</b>	1
Section 1.1. Introduction	2
Section 1.2. A Single-Server Queue (program <code>ssq1</code> )	12
Section 1.3. A Simple Inventory System (program <code>sis1</code> )	26
<b>Chapter 2. Random Number Generation</b>	37
Section 2.1. Lehmer Random Number Generation: Introduction	38
Section 2.2. Lehmer Random Number Generation: Implementation (library <code>rng</code> )	48
Section 2.3. Monte Carlo Simulation (programs <code>galileo</code> and <code>buffon</code> )	61
Section 2.4. Monte Carlo Simulation Examples (programs <code>det</code> , <code>craps</code> , <code>hat</code> , and <code>san</code> )	74
Section 2.5. Finite-State Sequences	88
<b>Chapter 3. Discrete-Event Simulation</b>	100
Section 3.1. Discrete-Event Simulation (programs <code>ssq2</code> and <code>sis2</code> )	101
Section 3.2. Multi-Stream Lehmer Random Number Generation (library <code>rngs</code> )	111
Section 3.3. Discrete-Event Simulation Models (program <code>ssms</code> )	120
<b>Chapter 4. Statistics</b>	131
Section 4.1. Sample Statistics (program <code>uvs</code> )	132
Section 4.2. Discrete-Data Histograms (program <code>ddh</code> )	148
Section 4.3. Continuous-Data Histograms (program <code>cdh</code> )	159
Section 4.4. Correlation (programs <code>bvs</code> and <code>acs</code> )	172
<b>Chapter 5. Next-Event Simulation</b>	185
Section 5.1. Next-Event Simulation (program <code>ssq3</code> )	186
Section 5.2. Next-Event Simulation Examples (programs <code>sis3</code> and <code>msq</code> )	198
Section 5.3. Event List Management (program <code>ttr</code> )	206
<b>Chapter 6. Discrete Random Variables</b>	223
Section 6.1. Discrete Random Variables	224
Section 6.2. Generating Discrete Random Variables	236
Section 6.3. Discrete Random Variable Applications (program <code>sis4</code> )	248
Section 6.4. Discrete Random Variable Models	258
Section 6.5. Random Sampling and Shuffling	268

<b>Chapter 7. Continuous Random Variables</b>	279
Section 7.1. Continuous Random Variables	280
Section 7.2. Generating Continuous Random Variables	291
Section 7.3. Continuous Random Variable Applications (program <code>ssq4</code> )	302
Section 7.4. Continuous Random Variable Models	313
Section 7.5. Nonstationary Poisson Processes	325
Section 7.6. Acceptance-Rejection	335
<b>Chapter 8. Output Analysis</b>	346
Section 8.1. Interval Estimation (program <code>estimate</code> )	347
Section 8.2. Monte Carlo Estimation	360
Section 8.3. Finite-Horizon and Infinite-Horizon Statistics	368
Section 8.4. Batch Means	375
Section 8.5. Steady-State Single-Server Service Node Statistics	383
<b>Chapter 9. Input Modeling</b>	396
Section 9.1. Trace-Driven Modeling of Stationary Processes	397
Section 9.2. Parametric Modeling of Stationary Processes	408
Section 9.3. Modeling Nonstationary Processes	423
<b>Chapter 10. Projects</b>	438
Section 10.1. Empirical Tests of Randomness	439
Section 10.2. Birth-Death Processes	460
Section 10.3. Finite-State Markov Chains	487
Section 10.4. A Network of Single-Server Service Nodes	507
<b>Appendices</b>	520
A. Simulation Languages	521
B. Integer Arithmetic (program <code>sieve</code> )	528
C. Parameter Estimation Summary	535
D. Random Variate Models (library <code>rvms</code> )	537
E. Random Variate Generators (library <code>rvgs</code> )	545
F. Correlation and Independence	546
G. Error in Discrete-Event Simulation	557
<b>References</b>	569

# Preface

This book presents an introduction to computational and mathematical techniques for *modeling, simulating, and analyzing* the performance of various systems using simulation. For the most part the system models studied are: *stochastic* (at least some of the system state variables are random); *dynamic* (the time evolution of the system state variables is important); and *discrete-event* (significant changes in system state variables are associated with events that occur at discrete time instances only). Therefore, the book represents an introduction to what is commonly known as *discrete-event simulation*. There is also a significant, but secondary, emphasis on *Monte Carlo simulation* and its relation to *static* stochastic systems. Deterministic systems, static or dynamic, and stochastic dynamic systems that evolve continuously in time are not considered in any significant way.

Discrete-event simulation is a multi-disciplinary activity studied and applied by students of applied mathematics, computer science, industrial engineering, management science, operations research, statistics, and various hybrid versions of these disciplines found in schools of engineering, business, management, and economics. As it is presented in this book, discrete-event simulation is a computational science — a mix of theory and experimentation with a computer as the primary piece of laboratory equipment. In other words, discrete-event simulation is a form of computer-aided model building and problem solving. The goal is insight, a better understanding of how systems operate and respond to change.

## Prerequisites

In terms of formal academic background, we presume the reader has taken the undergraduate equivalent of the first several courses in a conventional computer science program, two calculus courses and a course in probability or statistics. In more detail, and in decreasing order of importance, these prerequisites are as follows.

*Computer Science* — readers should be able to program in a contemporary high-level programming language, for example C, C++, Java, Pascal, or Ada, and have a working knowledge of algorithm complexity. Because the development of most discrete-event simulation programs necessarily involves an application of queues and event lists, some familiarity with dynamic data structures is prerequisite, as is the ability to program in a language that naturally supports such things. By design, the computer science prerequisite is strong. We firmly believe that the best way to learn about discrete-event simulation is by hands-on model building. We consistently advocate a structured approach wherein a model is constructed at three levels — conceptual, specification, and computational. At the computational level the model is built as a computer program; we believe that this construction is best done with a standard and widely available general-purpose high-level programming language, using already-familiar (editor, compiler, debugger, etc.) tools.\*

---

\* The alternative to using a general-purpose high-level programming language is to use a (proprietary, generally unfamiliar and potentially expensive) special-purpose simulation language. In some applications this may be a superior alternative, particularly if the simulation language is already familiar (and paid for); see Chapter 1 and Appendix A for more discussion of this trade-off.

*Calculus* — readers should be able to do single-variable differential and integral calculus. Although still relatively strong, the calculus prerequisite is as weak as possible. That is, for example, we have generally avoided the use of multi-variate calculus; however, single-variable integration and differentiation is used as appropriate in the discussion of continuous random variables, for example. In addition, we freely use the analogous, but more computationally intuitive, discrete mathematics of summation and differencing in the discussion of discrete random variable. By design, we maintain a balance between continuous and discrete stochastic models, generally using the more easily understood, but less common, discrete (non-calculus) techniques to provide motivation for the corresponding continuous (calculus) techniques.

*Probability* — readers should have a working knowledge of probability including random variables, expected values, and conditioning. Some knowledge of statistics is also desirable, but not necessary. Those statistical tools most useful in discrete-event simulation are developed as needed. Because of the organization of the material, our classroom experience has been that students with strength in the computer science and calculus prerequisites only can use this book to develop a valid intuition about things stochastic. In this way the reader can learn about discrete-event simulation and, if necessary, also establish the basis for a later formal study of probability and statistics. That study is important for serious students of discrete-event simulation because without the appropriate background a student is unlikely to ever be proficient at modeling and analyzing the performance of stochastic systems.

## Organization and Style

The book has ten chapters, organized into 41 sections. The shortest path through the text could *exclude* the 15 optional Sections 2.4, 2.5, 4.4, 5.3, 6.4, 6.5, 7.4, 7.5, 7.6, 8.5, 9.3, and 10.1–10.4. All the 26 remaining core sections are consistent with a 75-minute classroom lecture and together they define a traditional one-semester, three credit-hour course.\* Generally, the optional sections in the first nine chapters are also consistent with a 75-minute presentation and so can be used in a classroom setting as supplemental lectures. Each section in the tenth chapter provides relatively detailed specifications for a variety of discrete-event simulation projects designed to integrate much of the core material. In addition, there are seven appendices that provide background or reference material.

In a traditional one-semester, three credit-hour course there may not be time to cover more than the 26 core sections. In a four credit-hour course there will be time to cover the core material, some of the optional sections (or appendices) and, if appropriate, structure the course around the projects in the tenth chapter as a culminating activity. Similarly, some optional sections can be covered in a three credit-hour course, provided student background is sufficient to warrant not devoting classroom time to some of the core sections.

---

\* Because of its multi-disciplinary nature, there is not universal agreement on what constitutes the academic core of discrete-event simulation. It is clear, however, that the core is large, sufficiently so that we have not attempted to achieve comprehensive coverage. Instead, the core sections in the first nine chapters provide a self-contained, although limited, first course in discrete-event simulation.

The book is organized consistent with a dual philosophy: (*i*) begin to model, simulate, and analyze simple-but-representative systems as soon as possible; (*ii*) whenever possible, encourage the experimental exploration and self-discovery of theoretical results before their formal presentation. As an example of (*i*), detailed trace-driven computational models of a single-server queue and a simple inventory system are developed in Chapter 1, then used to motivate the need for the random number generator developed in Chapter 2. The random number generator is used to convert the two trace-driven models into stochastic models that can be used to study both transient and steady-state system performance in Chapter 3. Similarly, as an example of (*ii*), an experimental investigation of sampling uncertainty and interval estimation is motivated in Chapters 2 and 3. A formal treatment of this topic is presented in Chapter 8.

We have tried to achieve a writing style that emphasizes *concepts* and *insight* without sacrificing rigor. Generally, formalism and proofs are not emphasized. When appropriate, however, definitions and theorems (most with proofs) are provided, particularly if their omission could create a sense of ambiguity that might impede a reader's ability to understand concepts and develop insights.

## Software

Software is an integral part of the book. We provide this software as source code for several reasons. Because a computer program is the logical product of the three-level approach to model building we advocate, an introductory discrete-event simulation book based on this philosophy would be deficient if a representative sampling of such programs were not presented. Moreover, many important exercises in the book are based on the idea of extending a system model at the computational level; these exercises are conditioned on access to the source code. The software consists of many complete discrete-event programs and a variety of libraries for random number generation, random variate generation, statistical data analysis, priority queue access, and event list processing.

The software has been translated from its original development in Turbo Pascal to ANSI C with *units* converted to C libraries. Although experienced C programmers will no doubt recognize the Pascal heritage, the result is readable, structured, portable, and reasonably efficient ANSI C source code.\*

## Exercises

There are exercises associated with each chapter and some appendices — about 400 in all. They are an important part of the book, designed to reinforce and extend previous material and encourage computational experimentation. Some exercises are routine, others are more advanced; the advanced exercises are denoted with an ‘**a**’ superscript. Some of the advanced exercises are sufficiently challenging and comprehensive to merit consideration as (out-of-class) exam questions or projects. Serious readers are encouraged to work a representative sample of the routine exercises and, time permitting, a large portion of the advanced exercises.

---

\* All the programs and libraries compile successfully, without warnings, using the GNU C compiler `gcc` with the `-ansi -Wall` switches set. Alternatively, the C++ compiler `g++` can be used instead.

Consistent with the computational philosophy of the book, a significant number of exercises require some computer programming. If required, the amount of programming is usually small for the routine exercises, less so for the advanced exercises. For some of the advanced exercises the amount of programming may be significant. In most cases when programming is required, the reader is aided by access to source code for the programs and related software tools the book provides.

Our purpose is to give an introductory, intuitive development of algorithms and methods used in Monte Carlo and discrete-event simulation modeling. More comprehensive treatments are given in the textbooks referenced throughout the text.

### **Acknowledgments**

We have worked diligently to make this book as readable and error-free as possible. We have been helped in this by student feedback and the comments of several associates. Our thanks to all for your time and effort. We would like to acknowledge the contribution of Don Knuth whose  $\text{\TeX}$  makes the typesetting of technical material so rewarding and Michael Wichura whose  $\text{\Pictex}$  macros give  $\text{\TeX}$  the ability to do graphics. Particular thanks to former students Mousumi Mitra and Rajeeb Hazra who  $\text{\TeX}$ -set preliminary versions of some material, Tim Seltzer who  $\text{\Pictex}$ -ed preliminary versions of some figures, Dave Geyer who converted a significant amount of Pascal software into C, and Rachel Siegfried who proofread much of the manuscript. Special thanks goes to our colleagues and their students who have class-tested this text and provided us lists of typos and suggestions: Dave Nicol at William & Mary, Tracy Camp at the University of Alabama, Dan Chrisman at Radford University, Rahul Simha at William & Mary, Evgenia Smirni at William & Mary, Barry Lawson at the University of Richmond, Andy Miner at Iowa State University, Ben Coleman at Moravian College, and Mike Overstreet at Old Dominion University. Thanks also to Barry Lawson, Andy Miner, and Ben Coleman, who have prepared PowerPoint slides to accompany the text. We appreciate the help, comments, and advice on the text and programs from Sigrún Andradóttir, Kerry Connell, Matt Duggan, Jason Estes, Diane Evans, Andy Glen, James Henrikson, Elise Hewett, Whit Irwin, Charles Johnson, Rex Kincaid, Pierre L'Ecuyer, Chris Leonetti, David Lutzer, Jeff Mallozzi, Nathan & Rachel Moore, Bob Noonan, Steve Roberts, Eric & Kristen Rozier, Jes Sloan, Michael Trosset, Ed Walsh, Ed Williams, and Marianna Williamson concerning the programming or parts of this text. Bruce Schmeiser, Barry Nelson, and Michael Taaffe provided valuable guidance on the framework introduced in Appendix G. Thanks to Barry Lawson, and Nathan & Rachel Moore for contributing exercise solutions to the manual that is being edited by Matt Duggan. A special word of thanks goes to Barry Lawson for his generous help with  $\text{\TeX}$ ,  $\text{\Pictex}$ , setting up make files, and proofreading the text. The authors also thank the College of William & Mary for some teaching relief needed to complete this text.

*Steve Park & Larry Leemis  
January, 2000*

*Blank Page*