Simulation
A Modeler's Approach

JAMES R. THOMPSON
Rice University
Houston, Texas

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York / Chichester / Weinheim / Brisbane / Singapore / Toronto

This Page Intentionally Left Blank

Simulation

WILEY SERIES IN PROBABILITY AND STATISTICS
APPLIED PROBABILITY AND STATISTICS SECTION

Established by WALTER A. SHEWHART and SAMUEL S. WILKS

Editors: *Vic Barnett, Noel A. C. Cressie, Nicholas I. Fisher, Iain M. Johnstone, J. B. Kadane, David G. Kendall, David W. Scott, Bernard W. Silverman, Adrian F. M. Smith, Jozef L. Teugels; Ralph A. Bradley, Emeritus, J. Stuart Hunter, Emeritus*

A complete list of the titles in this series appears at the end of this volume.

Simulation
A Modeler's Approach

JAMES R. THOMPSON
Rice University
Houston, Texas

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York / Chichester / Weinheim / Brisbane / Singapore / Toronto

Library of Congress Cataloging-in-Publication Data:

Thompson, James R. (James Robert), 1938–
Simulation : a modeler's approach / James R. Thompson.
p. cm. — (Wiley series in probability and statistics. Applied probability and statistics)
"A Wiley-Interscience publication."
Includes bibliographical references and index.
ISBN 0-471-25184-4 (alk. paper)
1. Experimental design. 2. Mathematical models. 3. Mathematical statistics. I. Title. II. Series.
QA279.T494 1999
003—dc21 99-33022

10 9 8 7 6 5 4 3 2 1

To my mother, Mary Haskins Thompson

This Page Intentionally Left Blank

Contents

Preface

Half a century ago, John von Neumann created the digital computer as a device for carrying out simulations. Although the computer revolution has advanced to the point where 500 MHz boxes are available for under $2000, the simulation revolution is still in its infancy.

The situation where we can produce a probability profile of an investment strategy based on, say, 10,000 simulations with inputs the parameters of that strategy is still more talked about than achieved. The same can be said for a stochastic analysis of a political/military strategy in the Balkans. Stochastic models for a proposed new transportation system in a large city are still not well developed. Cancer profiles for individual patients based on their immune systems and the likely presence of metastases at the time of diagnosis are not readily available. Stochastic profiles of an epidemic vectored into the United States by a hostile power, based on the modality of its introduction and transmission pathways, are undeveloped. Particularly in simulation, software vision has lagged hardware by decades. A major function of this book is to indicate possibilities for synergy between data, models and the digital computer.

What is simulation? Presumably a simple question, but the scientific community is far from a consensus as to the answer. A government administrator might decide to "simulate" the national effect of a voucher system by taking a single school district and implementing a voucher system there. To a geologist, a simulation might be a three-dimensional differential-integral equation dynamic (hence, four-dimensional) model of implementation of tertiary recovery for an oil field. To a numerical analyst, a simulation might be an approximation-theoretic pointwise function evaluator of the geologist's dynamic model. To a combat theorist, a simulation might consist of use of the Lanchester equations to conjecture as to the result of a battle under varying conditions. To a nonparametric bootstrapper, simulation might consist in resampling to obtain the 95% confidence interval of the correlation coefficient between two variables.

While all of the above may be legitimate definitions of *simulation*, we shall concentrate on the notion of a simulation being the generation of pseudodata on the basis of a model, a database, or the use of a model in the light of a database. Some refer to this as *stochastic simulation*, since such pseudodata tends to change from run to run.

A model? What is that? Again, the consensus does not exist. We shall take a model to be a mathematical summary of our best guess as to what is going on in a part of the real world. We should not regard a model as reality, or even as a stochastic perturbation of reality, only as our current best guess as to a portion of reality. We should not be surprised if today's model is considered rather poor ten years hence. Some people quite mistakenly try to build the biggest model they can. It is not unusual for models of really big systems (such as the world) to be completely artificial. (Club of Rome models come to mind.)

In attempting to come up with one big all-encompassing theory of everything, one loses a great deal. Compartmentalization is clearly one way out of the morass. We can try for a theory that works under very specific conditions. A reasonable way to proceed. But like all good ideas, we might carry it to an extreme. Billy takes a test while carrying a good luck charm in his pocket and makes 100. He takes another test without the charm and scores 50. A third test with the charm yields another 100. Inference: the good luck charm produced better scores for Billy than he would otherwise have made, at least on the days when the tests were taken. An extreme case of nominalist logic, but the point is clear enough. *Ad hoc'ery* leads us very quickly to magic rather than to science. Pre-Socratic modalities of thought emerge, and we in trouble.

It is a hallmark of Western thinking that events indexed on time and faithfully recorded give us a database on which inferences might be made. On the other hand, the postmodernist view holds that the recorder of the events creates, more or less arbitrarily, his or her own history, that there is no reality behind the recording. The recording is the history. The recorder has created his or her own reality. If databases are just a reflection of the prejudices of the recorder, the modeler simply concatenates his or her prejudices with those of the creator of the database to give us, well, nothing very useful.

Models are generally oversimplifications, at best. Perhaps it would be better to get the modeler out of the loop. Among those who would like to free us from the bondage of models is University of Southern California Professor Bart Kosko, a recognized leader in neural networks and fuzzy thinking. In *Fuzzy Thinking* , he writes:

> ...linear systems are the toy problems of science and yet most scientists treat real systems as if they were linear. That's because we know so little math and our brains are so small and we guess so poorly at the cold gray unknown nonlinear world out there. ....Fuzzy systems let us guess at the nonlinear world and yet do not make us write down a math model of the world. .... The technical term for it is model-free estimation or approximation. ...
>
> The key is no math model. Model-free estimation. Model

> freedom. If you have a math model, fine. But where do you find one? You find good math models only in textbooks and classrooms. They are toy answers to toy problems. The real world pays no attention to most of them. So how do you know how well your math guess fits Nature's process? You don't and you never can. You have to ask God or No God or Nature and no one knows how to do that. Short of that, we guess and test and guess again. Scientists often respect math more than truth and they do not mention that a math guess is no less a guess than a guess in everyday language. At least a word guess does not claim to be more than a guess and gains nothing from the fact that we have stated it in words. A math guess has more dignity the less math you know. Most math guesses are contrived and brittle and change in big ways if you change only a small value in them. Man has walked Earth for at least a million years and has just started to think in math and is not good at it. Fuzzy systems let us model systems in words.

We may well agree with the basic modeling problem mentioned by Kosko, although not at all with his "solution" of it. It is true that in modeling systems mathematically, people tend to be oriented toward models with whose mathematical complexities they can cope. For example, we can postulate that a tumor will grow, moment by moment, in proportion to its mass. And a further postulate can be made that the probability of a metastasis being generated by a tumor in a short period of time is proportional to the mass of the tumor. Another postulate can be made that the probability that a tumor will be discovered in an instant of time is proportional to the mass of the tumor. These postulates represent quite a simple-sounding model for the progression of cancer in a patient. But when one starts looking at the times of discovery of primary and secondary tumors and using this information to estimate the parameters of the simple-sounding model, it is discovered that getting anything like a likelihood function (a general first step in classical parameter estimation) is a hopelessly complicated business. The reason for the problem is that the axioms are made in a forward temporal direction, whereas the likelihood function is computed looking backward in time to the possible causes of generation of particular tumors. Such complexities have caused most biostatisticians to work with linear aggregate models, such as the survival times of patients taking drug A as opposed to drug B. Such analyses have not worked very well, and it is the failure of such simpleminded linearizations, in part, which have made the War on Cancer a series of losing engagements.

But to use a fuzzy system or a neural net as a way out of the linear oversimplification is to replace fiction with magic. We really need to know how cancer grows and spreads. Some glorified smoothing interpolator is unlikely to get us out of the soup. Later, we shall show how simulation

allows us to perform parameter estimation without false linear simplifications and without an uninformative hodgepodge of neural networks. The SIMEST paradigm will enable us to use postulated models in such a way that backwards-in-time mathematical operations can be eliminated in favor of a large number of forward simulations.

Nearly seven hundred years ago, that most famous of nominalists, William of Occam, gave a pronunciamento which can be viewed as a prototype of that of Kosko:

> Nonetheless, one should know that there are two kinds of universal. One kind is naturally universal, in that it is evidently a sign naturally predictable of many things, in a fashion analogous to the way smoke naturally signifies fire,.... Such a universal is nothing except a notion of the mind, and so no substance outside the mind nor any accident outside the mind is such a universal....The other kind of universal is so by established convention. In this way a word that is produced, which is really one quality, is a universal, because clearly it is a sign established by convention for the signifying of many things. Hence just as a word is said to be common, even so it can be said to be universal; but this does not obtain from the nature of the thing, but only from agreed upon convention.[1]

Nominalism is at odds with the realist (Aristotelian) view of science as an evolutionary search for better and better descriptions of objective reality. One current fashion in the history of science is to look for fundamental change points in the dominant scientific paradigm. These change points are essentially a political phenomenon. This was the view of the late Thomas Kuhn. For example, the Newtonian relationship between force and momentum is given by

$$F = \frac{d}{dt}(mv), \tag{1}$$

which is normally written as

$$F = ma, \tag{2}$$

where F is force, m is mass, v is velocity and a is acceleration. But following the discovery of Einstein that mass changes as the speed of the object increases, we are to view Newton's representation as hopelessly out of date. We reject Newtonianism in favor of Einsteinism and go on about our business anxiously awaiting the advent of the new evangel which will trash Einstein.

In this book, we take the more classical notion that Einstein improved Newton's model rather than making it fit for the dustbin. Consider that

[1] *The Sum of All Logic* translation by Philtheus Boehner, O.F.M.

Einstein has

$$m = \frac{m_0}{\sqrt{1 - v^2/c^2}} \tag{3}$$

where c is the speed of light. We can then take this expression for m and substitute it back in (1) to give

$$F = \frac{d}{dt}\left(\frac{m_0}{\sqrt{1 - v^2/c^2}}v\right) . \tag{4}$$

Beyond that, the Newtonian model gives essentially the same results as that of Einstein for bodies not moving at light speed. Anybody who believes that Newtonian physics has outlived its usefulness has not examined the way mechanics is taught by contemporary departments of physics and mechanical engineering. So we shall take the view of evolution rather than revolution in model progression. It is the view of the author that simulation is simply a computer-facilitated implementation of the model-building process (a.k.a the scientific method).

One might well ask why the necessity for this philosophical digression. Let us just start simulating already. Fair enough. But simulate what? And to what purpose? A great deal of the literature in simulation is oriented to a kind of idealized mathematical formalism. For example, there are many hundreds of papers written on the subject of the generation of random numbers. And very important many of them are. But this can lead us to a kind of formalist copout. If we dwell excessively on an algorithm that yields a string of random numbers which satisfy some kind of arbitrary set of desiderata, we can get lost in the comfortable realm of mathematical theorem statement and proof. All very tempting, but we shall not travel very far down that road. This is a statistics book, and statisticians should be concerned with a reality beyond data-free formalism. That is why statistics is not simply a subset of mathematics.

Our major interest in this book will be using simulation as a computational aid in dealing with and creating models of reality. We will spend a bit of time in going through the philosophy of quasirandom number generators and we will go through some of the old Monte Carlo utilization of simulation in, for example, the approximation of definite integrals. But our main goal will be to use simulation as an integral part of the interaction between data and models which are approximations to the real systems that generated them.

A goodly amount of time will be employed in resampling procedures where we use resampling from a data set (or, in the case of SIMDAT, from the nonparametric density estimator of the density based on the data set) to test some hypothesis and/or obtain some notion of the variability of the data. But much more important will be the use of simulation as an integral part of the modeling process itself. As an example of the latter, let us consider a couple of "toy problems."

We recall the quiz show with master of ceremonies Monty Hall. Three doors were given, say A, B, and C. Behind one of these was a big prize. Behind the others, nothing splendid at all. The contestant would choose one of the doors, say A, and then the MC would tell him one of the other doors, say C, behind which the splendid prize did not exist. Then the contestant was given the option of sticking with his original choice A or switching to B. The quiz show continued for some years in this manner with contestants going both ways, so it is clear that the general consensus was that there was no systematic preference for either strategy. Let us go through a standard Bayesian argument to support this view.

Let us compute the probability of winning of the contestant who "stands pat," i.e., he chose A originally; he learned that the prize was not behind door C, yet he decided to stick with his original choice of door A. Let us compute the probability that he will win given that C is not the prize door.

$$\begin{aligned} P(A|C^c) &= \frac{P(A \cap C^c)}{P(C^c)} \\ &= \frac{P(A)P(C^c|A)}{P(C^c)} \\ &= \frac{(1/3)(1)}{2/3} \\ &= .5. \end{aligned}$$

The reasoning seems to be correct. The prior probability here is $P(A) = 1/3$. If A is the prize door, the chance that C is not the prize door is 1 [i.e., $P(C^c|A) = 1$]. Finally, the prior probability that C will not be the prize door is 2/3 [i.e., $P(C^c) = 2/3$]. Furthermore, once we have been told that C is not the prize door, then $P(A|C^c) + P(B|C^c) = 1$, so $P(B|C^c)$ must equal .50 as well. A formal argument would seem to support the popular wisdom that it makes no difference, over the long haul, whether contestants stand pat or switch to B.

But, so the story goes, somebody went back over the records of the contestants and found that those who switched, on the average, did better than those who stood pat. In fact, the switchers seemed to win about two thirds of the time. Could this be due to the laws of probability, or was something else afoot? Here is a case where the simulations consisted not of computer simulations, but actual implementations of the game.

To help us out, let us write a simple simulation program.

```
Set counter WA equal to zero
Set counter WSwitch equal to zero
Repeat 10,000 times
Generate U, a uniform random number between 0 and 1
If U is greater than .33333, go to *
Let WA = WA+1
```

```
Go to **
* WSwitch = WSwitch + 1
** Continue
WA = WA/10000
WSwitch = WSwitch/10000
Print WA and WSwitch
End
```

The argument here is straightforward. We can associate a random number less than .33333 with a win for A. If the prize is behind door A, the standpat strategy always produces a win. On the other hand, a number greater than .33333 will be associated with a win for B or C. The MC will tell us which of these two doors is not the prize door, so if we switch to the other, we always win. A simulation of 10,000 trials gave a .3328 probability of winning by the standpat strategy, as opposed to a .6672 probability of winning by the switch strategy.

Indeed, once one writes down this simulation, the problem is essentially solved without any simulations. It simply becomes transparent when we flowchart the program. But then, since we do not accept postmodernist and fuzziest notions of the possibility of logical inconsistency, we must needs see where we went wrong (and where the man in the street must have empirically gone wrong). Our writing down of Bayes' theorem is correct. The problem comes in the evaluation of $P(C^c|A)$. We should not interpret this to mean that C is not the prize door when A is the prize door. Rather, it is the probability that the MC will tell us that of the two non-A doors, C is not the prize door. He must pick one of the two B and C with equal probability. Hence $P(C^c|A)$ equals 1/2, rather than unity. Making this correction, we get the probability of winning using the standpat algorithm to be 1/3, as it should be.

Next, let us relate the Monty Hall problem to one of antiquity. Below, we consider one of the many "prisoner's dilemma" problems. A prisoner is one of three condemned to be beheaded on the morrow. But the Sultan, in his mercy, has decided to pardon one of the prisoners. Prisoner A is a mathematician. He wishes to improve his chance of not getting the chop. He knows that the chief jailer knows who is to be spared but has been warned by the Sultan that if he tells any of the prisoners who is to be spared, then he, the jailer, will be disemboweled. The mathematician calls the jailer aside and offers him 100 drachmas to tell him, not the name of the fortunate, but the name of one of those, other than, possibly, himself, who is to be executed. The jailer agrees and tells A that C is one of the condemned. A heaves a sigh of relief, since he now believes that his probability of being spared has increased from one-third to two-thirds. Let us note, however, that A actually is in the position of the standpat player in the Monte Hall example. His probability of survival is actually one-third. On the other hand, if B happens to overhear the exchange between A and his jailer, he does have some reason for relative optimism, since he

stands in the position of the switch player in the Monty Hall example. It is, naturally, an easy matter to write down a simulation program for the prisoner's dilemma situation, but the analogue between the two situations is actually an isomorphism (i.e., the problems are the same precisely.

Let us note, here, the fact that the construction of a simulation is, clearly, a kind of modeling process. It will generally cause us to analogize a temporal process, since computer programs consist of instructions which take place in a sequence. A number of the differential and integral equations of physics were natural summarizing models for the precomputer age. Typically, the closed-form solution for fixed parameter values is not available. We must be satisfied with pointwise approximations to the value of the dependent variable vector y in terms of the independent variable vector x. It turns out that in a large number of cases we can approximately carry out this approximation by a simulation, which frequently is based on the microaxioms that gave rise to the differential-integral equation summary in the first place.

Of greater interest still is the situation where we have the postulates for our model (and hence, in principle, the model itself) and a database and wish to estimate the underlying parameters. According to classical paradigms, in order to estimate these parameters, we must obtain something like a likelihood function. But this is generally a hopelessly complicated task. The SIMEST paradigm, which we examine, allows us to go directly from the postulates and the data to the estimation process itself. This is achieved stepwise by creating a large class of pseudodata predicated on the assumption of a particular (vector) parameter value. By comparing the pseudodata with the actual data, we have a natural means of moving to a good estimate of the characterizing parameter. This is a temporally forward estimation procedure, as opposed to the classical estimation strategies which look backwards in time from the data points.

Another use of simulation will be in the realm of scenario analysis. We shall, for example, examine some of the current models for movement of stock and derivative prices and analyze some pricing strategies in the light of changes made in these models. This is a speculative use of simulation. We are not using data intimately. Rather, we wish to ask "what if?" questions and use simulation to give some clue as to feasible answers.

As we have noted, there are many who, discouraged by the results of the use of bad models, would like to dispense with models altogether. And simulation can frequently be put to good use in dealing with model-free analyses. The basic problem of model-free analysis is that it can work well when one is interpolating within a database, but it generally decays rapidly when we start extrapolating. To make matters even more difficult, for data of high dimensionality, even interpolation within the convex hull of the database is, in fact, a problem of extrapolation, since the data will generally be distributed in clusterlike clumps separated by substantial empty space.

Simulation is also used by those who are happy to assume the correct-

ness of a model and get on with the business, say, of obtaining pointwise approximations of a dependent variable. To these, the formalism is the matter of interest, and they are happy to produce theorems and tables to their purpose.

While conceding that from time to time, both the nominalist and idealist approaches listed above have their uses, we shall in this book, be concerned largely with what would appear to be a middle ground: Namely, we shall usually be working with models, but with a view that the models themselves must be improved whenever it is feasible to do so. To us, simulation will provide a device for working with models, testing models, and building new models. So then, to us, simulation will be a kind of paradigm for realistic evolutionary modeling. At present, simulation is used by many as an adjuvant for dealing with old modeling techniques, say, the numerical approximation to pointwise evaluation of a differential equation. In the future, the author believes that simulation-based modeling will be at least as important as some of the older summarization models, such as differential equations. One will go directly from postulates and data to estimation and approximation without intervening classical summarization models. This would amount to something resembling a paradigm shift in the sense of Kuhn. It is a very big deal indeed to be able to say: "If our assumptions are correct, then here is a program for simulating a host of possible realizations with a variety of frequencies." At present, most simulations still consist of assists in dealing with older modeling summarizations. That is changing. To a large extent, the future of science will belong to those willing to make the shift to simulation-based modeling. This book has been written with such readers in mind.

In acknowledging support for this book, I start with thanks to my thesis advisor, John Tukey, who taught his students continually to question and modify preconceived notions of reality, and the late Elizabeth Tukey, whose graciousness and kindness will always be remembered by her husband's students. Then, I would like to acknowledge the support of the Army Research Office (Durham) under DAAH04-95-1-0665 and DAAD19-99-1-0150 for this work. Among my colleagues in Army science, I would particularly like to thank Robert Launer, Jagdish Chandra, Malcolm Taylor, Barry Bodt, and Eugene Dutoit. At the Polish Academy of Science, I would like to thank Jacek Koronacki. At Rice, I would like to thank Ed Williams, Katherine Ensor, Marek Kimmel, David Scott, Martin Lawera, Diane Brown, Tres Schwalb, Tony Elam, Sidney Burrus, Michael Carroll, Keith Baggerly, Dennis Cox, Patrick King, Peter Olofsson, Roxy Cramer, Mary Calizzi, and John Dobelman. At the University of Texas M.D. Anderson Cancer Center, I would like to thank Barry Brown and Neely Atkinson. At the University of South Carolina, I would like to thank Webster West. At Princeton, I would like to thank Stuart Hunter and the late Geoffrey Watson. At the Rand Corporation, I would like to thank Marc Elliott. I also wish to thank Steven Boswell of Lincoln Laboratories, and James Gen-

tle from Visual Numerics and George Mason University. At John Wiley & Sons, I would like to thank my editor Stephen Quigley.

Finally, and most importantly, I would like to thank my wife, Ewa Majewska Thompson, for her love and encouragement.

James R. Thompson

Houston, Texas
Easter, 1999

Simulation

This Page Intentionally Left Blank

Chapter 1

The Generation of Random Numbers

1.1 Introduction

There are many views as to what constitutes simulation. To the statistician, simulation generally involves randomness as a key component. Engineers, on the other hand, tend to consider simulation as a deterministic process. If, for example, an engineer wishes to simulate tertiary recovery from an oil field, he or she will probably program a finite element approximation to a system of partial differential equations. If a statistician attacked the same problem, he or she might use a random walk algorithm for approximating the pointwise solution to the system of differential equations.

In the broadest sense, we may regard the simulation of a process as the examination of any emulating process simpler than that under consideration. The examination will frequently involve a *mathematical model*, an oversimplified mathematical analogue of the real-world situation of interest. The related simpler process might be very close to the more complex process of interest. For example, we might simulate the success of a proposed chain of 50 grocery stores by actually building a single store and seeing how it progressed. At a far different level of abstraction, we might attempt to describe the functioning of the chain by writing down a series of equations to approximate the functioning of each store, together with other equations to approximate the local economies, and so on. It is this second level of abstraction that will be of more interest to us.

It is to be noted that the major component of simulation is neither stochasticity nor determinism, but rather, analogy. Needless to say, our visions of reality are always other than reality itself. When we see a forest, it is really a biochemical reaction in our minds that produces something to

which we relate the notion of *forest*. When we talk of the *real world*, we really talk of perceptions of that world which are clearly other than that world but are (hopefully) in strong correlation with it. So, in a very real sense, analogy is "hardwired" into the human cognitive system. But to carry analogy beyond that which it is instinctive to do involves a learning process more associated with some cultures than with others. And it is the ability of the human intellect to construct analogies that makes modern science and technology a possibility. Interestingly, like so many other important advances in human thought, the flowering of reasoning by analogy started with Socrates, Plato, and Aristotle. Analogy is so much a part of Western thinking that we tend to take it for granted. In simulation we attempt to enhance our abilities to analogize to a level consistent with the tools at our disposal.

The modern digital computer, at least at the present time, is not particularly apt at analogue formulations. However, the rapid digital computing power of the computer has enormous power as a device complementary to the human ability to reason by analogy. Naturally, during most of the scientific epoch, the digital computer simply did not exist. Accordingly, it is not surprising that most of science is still oriented to methodologies in the formulation of which the computer did not play an intimate part.

The inventor of the digital computer, John von Neumann, created the device to perform something like random quadrature rather than to change fundamentally the precomputer methodology of modeling and analogy. And indeed, the utilization of the computer by von Neumann was oriented toward being a fast calculator with a large memory. This kind of mindset, which is a carryover of modeling techniques in the precomputer age, led to something rather different from what I call simulation, namely the *Monte Carlo method*.

According to this methodology, we essentially start to work on the abstraction of a process (through differential equations and the like) as though we had no computer. Then, when we find difficulties in obtaining a closed form solution, we use the computer as a means of facilitating pointwise function evaluation.

One conceptual difference between simulation and Monte Carlo in this book will be that simulation will be closer to the model of the system underlying the data. However, there is no clear demarcation between the Monte Carlo method on the one hand and simulation on the other. As we shall see later, a fuller utilization of the computer frequently enables us to dispense with abstraction strategies suitable to a precomputer age.

As an example of the fundamental change that the modern digital computer makes in the modeling process, let us consider a situation where we wish to examine particles emanating from a source in the interior of an irregular and heterogeneous medium. The particles interact with the medium by collisions with it and travel in an essentially random fashion.

The classical approach for a regular and symmetrically homogeneous

medium would be to model the aggregate process by looking at differential equations that track the average behavior of the microaxioms governing the progress of the particles. For an irregular and nonsymmetrically homogeneous medium, the Monte Carlo investigator would attempt to use random walk simulations of these differential equations with pointwise change effects for the medium. In other words, the Monte Carlo approach would be to start with a precomputer age methodology and use the computer as a means for random walk implementations of that methodology.

If we wish to use the power of the digital computer more fully, we can go immediately from the microaxioms to random tracking of a large number of the particles. It is true that even with current computer technology, we will still not be in a position to deal with 10^{16} particles. However, a simulation dealing with 10^5 particles is both manageable and probably sufficient to make very reasonable conjectures about the aggregate of the 10^{16} particle system. In distinguishing between simulation and the Monte Carlo method, we will, in the former, be attempting the modeling in the light of the existence of a computer that may take us very close indeed to a precise emulation of the process under consideration. Clearly, then, *simulation* is a moving target. The faster the computer and the larger the storage, the closer we can come to a true simulation.

1.2 The Generation of Random Uniform Variates

Many simulations will involve some aspect of randomness. Theorem 1.1 shows that at least at the one-dimensional level, randomness can be dealt with if only we can find a random number generator from the uniform distribution on the unit interval $\mathcal{U}(0,1)$.

Theorem 1.1. Let X be a continuous random variable with distribution function $F(\cdot)$ [i.e., let $F(x) = P(X \leq x)$]. Consider the random variable $Y = F(x)$. Let the distribution function of Y be given by $G(y) = P(Y \leq y)$. Then Y is distributed as $\mathcal{U}(0,1)$.

Proof

$$G(y) = P(Y \leq y) = P(F(x) \leq y) = P(x \leq F^{-1}(y)) = y \tag{1.1}$$

since $P(x \leq F^{-1}(y))$ is simply the probability that X is less than or equal to that value of X than which X is less y of the time. This is precisely the distribution function of the uniform distribution on the unit interval. This proves the theorem. •

For the simulator, Theorem 1.1 has importance rivaling that of the central limit theorem, for it says that all that is required to obtain a satisfactory

random number generator for any continuous one-dimensional random variable, for which we have a means of inverting the distribution function, is a good $\mathcal{U}(0,1)$ generator. This is conceptually quite easy. For example, we might have an electrical oscillator in which a wavefront travels at essentially the speed of light in a linear medium calibrated from 0 to 1 in increments of, say, 10^{-10}. Then, we simply sense the generator at random times that an observer will pick. Aside from the obvious fact that such a procedure would be prohibitively costly, there seems to be a real problem with paying the observer and then reading the numbers into the computer. Of course, once it were done, we could use table look-up forever, being sure never to repeat a sequence of numbers once used.

Realizing the necessity for a generator that might be employed by the computer itself, without the necessity of human intervention except, perhaps, at the start of the generation process, von Neumann developed such a scheme. He dubbed the generator he developed the *midsquare method*. To carry out such a procedure, we take a number of, say, four digits, square it, and then take the middle four digits, which are used for the generator for the next step. If using base 10 numbering, we simply put a decimal before the first of the four digits. Let us show how this works with the simple example below.

We start with $X_0 = 3333$. Squaring this, we obtain 11108889. Taking the middle four digits, we have $X_1 = 1088$. Squaring 1088 we have 1183744. This gives $X_2 = 8374$, and so on. If we are using base 10, this gives us the string of supposed $\mathcal{U}(0,1)$ random variates.

The midsquare method is highly dependent on the starting value. Depending on the seed X_0, the generator may be terrible or satisfactory. Once we obtain a small value such as 0002, we will be stuck in a rut of small values until we climb out of the well. Moreover, as soon as we obtain 0, we have to obtain a new starter, since 0 is not changed by the midsquare operation.

Examinations of the midsquare method may be rather complicated mathematically if we are to determine, for example, the *cycle length*, the length of the string at which it starts to repeat itself. Some have opined that since this generator was used in rather crucial computations concerning nuclear reactions, civilization is fortunate that no catastrophe came about as a result of its use. As a matter of fact, for reasonable selections of *seeds* (i.e., starting values), the procedure can be quite satisfactory for most applications. It is, however, the specificity of behavior based on starting values that makes the method rather unpopular.

The midsquare method embodies more generally many of the attributes of *random number generators* on the digital computer. First, it is to be noted that such generators are not really random, since when we see part of the string, given the particular algorithm for a generator, we can produce the rest of the string. We might decide to introduce a kind of randomness by using the time on a computer clock as a seed value. However, it is fairly clear that we need to obtain generators realizing that the very nature of

realistic generation of random numbers on the digital computer will produce problems. Attempting to wish these problems away by introducing factors that are random simply because we do not know what they are is probably a bad idea. Knuth [7] gives an example of an extremely complex generator of this sort that would appear to give very random-looking strings of random numbers, but in fact easily gets into the rut of reproducing the seed value forever.

The maxim of dealing with the devil we know dominates the practical creation of random number generators. We need to obtain algorithms which are conceptually simple so that we can readily discern their shortcomings. The most widely used of the current random number generation algorithms is the *congruential random number generator.* The apparent inventor of the congruential random number generation scheme is D.H. Lehmer, who introduced the algorithm in 1951 [10].

First, we note how incredibly simple the scheme is. Starting with a seed X_0, we build up the succession of "pseudorandom" numbers via the rule

$$X_{n+1} = aX_n + b(\text{mod } m). \tag{1.2}$$

One of the considerations given with such a scheme is the length of the string length of pseudorandom numbers before we have the first repeat. Clearly, by the very nature of the algorithm, with its one-step memory, once we have a repeat, the new sequence will repeat itself exactly. If our only concern is the length of the cycle before a repeat, a very easy fix is available. Choosing $a = b = 1$ and $X_0 = 0$, we have for any m,

$$\begin{aligned} X_1 &= 1 \\ X_2 &= 2 \\ X_3 &= 3 \\ &\cdots \quad \cdots \\ X_{m-1} &= m-1. \end{aligned} \tag{1.3}$$

Seemingly, then, we have achieved something really spectacular, for we have a string that does not repeat itself until we get to an arbitrary length of m. Of course, the string bears little resemblance to a random string, since it marches straight up to m and then collapses back to 1. We have to come up with a generator such that any substring of any length appears to be random. Let us consider what happens when we choose $m = 90$, $a = 5$, and $b = 0$. Then, if we start with $X_0 = 7$, we have

$$\begin{aligned} X_1 &= 35 \\ X_2 &= 85 \\ X_3 &= 65 \end{aligned}$$

$$\begin{aligned} X_4 &= 55 \\ X_5 &= 5 \\ X_6 &= 25 \\ X_7 &= 35. \end{aligned} \tag{1.4}$$

It could be argued that this string appears random, but clearly its cycle length of six is far too short for most purposes. Our task will be to find a long cycle length that also appears random. The rather simple proof of the following theorem is due to Morgan [10].

Theorem 1.2. Let $X_{n+1} = aX_n + b (\text{mod } m)$.
Let $m = 2^k$, $a = 4c + 1$, b be odd.
Then the string of pseudorandom numbers so generated has cycle length $m = 2^k$.
Proof
Let $Y_{n+1} = aY_n + b$.
Without loss of generality, we can take $X_0 = Y_0 = 0$.
Then

$$\begin{aligned} Y_1 &= aY_0 + b = b \\ Y_2 &= ab + b = b(1 + a) \\ Y_3 &= aY_2 + b = ab(a + a) + b = b(1 + a + a^2) \\ \dots &\quad \dots \\ Y_n &= b(1 + a + a^2 + a^3 + \dots + a^{n-1}). \end{aligned} \tag{1.5}$$

We observe that $X_i = Y_i - h_1 2^k$.
Now suppose that $X_i = X_j$ for $i > j$.
We wish to show that $i - j \geq 2^k$.
Now, $Y_i - Y_j = b(a^j + a^{j+1} + \dots + a^{i-1})$.
If $X_i = X_j$, then
$ba^j W_{i-j} = ba^j(1 + a + a^2 + \dots + a^{i-j-1}) = h_2 2^k$,
where $W_n = 1 + a + a^2 + \dots + a^{n-1}$ for $n \geq 1$.
To prove the theorem, we must show that W_{i-j} cannot equal an integer multiple of 2^k if $i - j < 2^k$, that is,
$W_{i-j} \neq h_3 2^k$ for $i - j < 2^k$.
We shall suppose first of all that *i - j is odd.*
Then $i - j = 2t + 1$ for $t \geq 0$.
(This is the place we use the fact that $a = 4c + 1$.)
$W_{2t} = (1-a^{2t})/(1-a) = [(1+4c)^{2t}-1]/[4c] = [(1+4c)^t-1][(1+4c)^t+1]/(4c)$
$= [(1 + 4c)^t + 1] \left[\sum_{i=1}^{t} (4c)^{i-1} \binom{t}{i}\right]$.
But $1 + (1 + 4c)^t = 2 + 4c \sum_{i=1}^{t} (4c)^{i-1} \binom{t}{i}$.

So, $W_{2t+1} = W_{2t} + a^{2t}$ is odd, since a is odd.
Hence, if $i - j$ is odd, then $W_{i-j} \neq h_3 2^k$, since W_{i-j} is odd.
Next we wish to consider the case where *i - j is even.*
If $i - j$ is even, then there exists an s such that $i - j = \alpha 2^s$, for some odd integer α.

$$\begin{aligned} W_{i-j} &= W_{\alpha 2^s} \\ &= 1 + a + \ldots + a^{\alpha 2^{s-1}-1} + a^{\alpha 2^{s-1}} + \ldots + a^{\alpha 2^s - 1} \\ &= W_{\alpha 2^{s-1}} + a^{\alpha 2^{s-1}} \left(1 + a + a^2 + \ldots + a^{\alpha 2^{s-1}-1}\right) \\ &= W_{\alpha 2^{s-1}} + a^{\alpha 2^{s-1}} W_{\alpha 2^{s-1}} \\ &= W_{\alpha 2^{s-1}} \left(1 + a^{\alpha 2^{s-1}}\right). \end{aligned} \tag{1.6}$$

Similarly, we have

$$W_{\alpha 2^{s-1}} = W_{\alpha 2^{s-2}} \left(1 + a^{\alpha 2^{s-2}}\right). \tag{1.7}$$

Continuing the decomposition,

$$\begin{aligned} W_{i-j} &= W_{\alpha 2^{s-2}} \left(1 + a^{\alpha 2^{s-2}}\right) \left(1 + a^{\alpha 2^{s-1}}\right) \\ &= W_\alpha \left(1 + a^\alpha\right) \left(1 + a^{\alpha 2}\right) \ldots \left(1 + a^{\alpha 2^{s-1}}\right). \end{aligned} \tag{1.8}$$

Recalling that

$$1 + (4c+1)^j = 2 + 4c \sum_{i=1}^{j} \binom{j}{i} (4c)^i, \tag{1.9}$$

we see that $W_{i-j} = W_\alpha \gamma 2^s$. Note that we have shown in the first part of the proof that for α odd, W_α is also odd. Furthermore, we note that the product of terms such as $1 + even$ is also odd, so γ is odd. Thus if $W_{i-j} = h_2 2^k$, we must have $s = k$. •

The following more general theorem is stated without proof.

Theorem 1.3. Let $X_{n+1} = aX_n + b \bmod(m)$. Then the cycle of the generator is m if and only if
(i) b and m have no common factor other than 1.
(ii) $(a - 1)$ is a multiple of every prime number that divides m.
(iii) $(a - 1)$ is a multiple of 4 if m is a multiple of 4.

So far, we have seen how to construct a sequence of arbitrarily long cycle length. Obviously, if we wish our numbers to lie on the unit interval, we will simply divide the sequence members by m. Thus, the jth random number would be X_j/m. It would appear that there remains the design problem of selecting a and b to give the generator seemingly random results. To do this, we need to examine congruential generators in the light of appropriate perceptions of randomness. We address this issue in the next section.

1.3 Latticing and Other Problems

In a sense, it is bizarre that we ask whether a clearly deterministic sequence, such as one generated by a *congruential random number generator*, is random. Many investigators have expressed amazement at early "primitive" schemes, such as those of von Neumann, devised for random number generation. As a matter of fact, I am personally unaware of any tragedy or near tragedy caused by any of these primitive schemes (although I have experienced catastrophes myself when I inadvertently did something putatively wrong, such as using, repetitively, the same seed to start runs of, supposedly different, sequences of congruential random numbers). The fact is, of course, that the issue is always decided on the basis of what are the minimal requirements for a sequence of random numbers.

Let us note that having generators with long intervals before numbers are repeated is not sufficient. For example, suppose that we decided to use the generator

$$X_{n+1} = X_n + .000000001, \text{ where } X_0 = 0. \tag{1.10}$$

Such a generator will give us 10^9 numbers between 0 and 1 with never a repeat. Clearly, however, it is totally unsatisfactory, since it creeps slowly from 0 to 1 in steady increments. It is true that after one billion numbers are generated, we will have the number of points generated in an interval between 0 and 1 equal to one billion times the interval length, as we should. But, if only, say, 100,000 points are generated, there will be no points at all in the interval [.0001,1]. Probably, no one would use such a generator. A modest criterion would be that even for a small number of points generated, say N, we should observe that the total number of points in an interval of length ϵ should be roughly equal to ϵN. Practically speaking, all the congruential random number generators in use seem to have this property.

Suppose, however, that we are employing a congruential random number generator to give points in the unit hypercube of dimension greater than 1. That could mean, for example, that if we are generating points in two dimensions, we could use a congruential random number generator in such a fashion that the first number in a string would give us the first dimension of a double, the second would give the second dimension of the double. Then the third number in a string would give us the first dimension of a second double, the fourth, the second dimension, and so on.

For many applications, it will be sufficient if we can show that for any small volume, say ϵ, of a hypercube of unit volume, for a large number of generated random numbers, say N, the number of these falling in the volume will be approximately ϵN. But suppose it turned out that there were regions of the hypercube in which we never obtained any points, regardless of the number of points generated. Such behavior is observed for the once popular RANDU generator of IBM. A little work reveals the following

relation [7]:

$$x_{i+1} = (6x_i - 9x_{i-1}) \bmod(2^{31}). \tag{1.11}$$

Such a relationship between successive triples is probably not disastrous for most applications, but it looks very bad when graphed from the proper view. Using D^2 Software's MacSpin, we observe two views of this generator, one, in Figure 1.1, seemingly random, the other in Figure 1.2, very much not.

$$x_{i+1} = (2^{16} + 3)x_i \bmod(2^{31}). \tag{1.12}$$

As a matter of fact, congruential random number generators generally have the problem of latticing. This holds even if we try to be clever by using a different generator for each dimension, or perhaps having one generator which randomly samples from each of M generators to pick a number for each dimension. One reasonable way to lessen this difficulty might be to use a generator that minimizes the maximum distance between two lattices. If this is achieved, then even though we will have vast empty regions (in fact, it is obvious that all the random numbers generated by a congruential random number generator must lie in a set of Lebesgue measure zero in the unit hypercube), it would be very difficult to conceive of a realistic situation where this might cause practical difficulty.

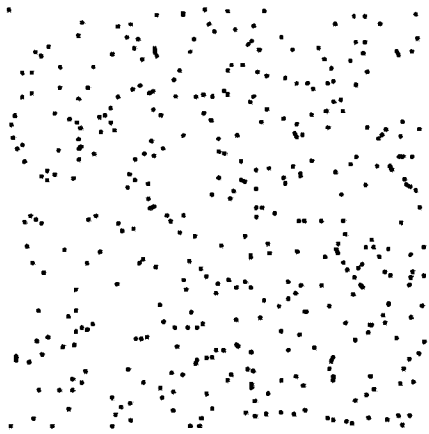

Figure 1.1. RANDU in Two Dimensions.