Systems Simulation:
The Shortest Route to Applications

This site features information about discrete event system modeling and simulation. It includes discussions on descriptive simulation modeling, programming commands, techniques for sensitivity estimation, optimization and goal-seeking by simulation, and what-if analysis.

Advancements in computing power, availability of PC-based modeling and simulation, and efficient computational methodology are allowing leading-edge of prescriptive simulation modeling such as optimization to pursue investigations in systems analysis, design, and control processes that were previously beyond reach of the modelers and decision makers.

Professor Hossein Arsham

To search the site, try Edit | Find in page [Ctrl + f]. Enter a word or phrase in the dialogue box, e.g. "optimization" or "sensitivity" If the first appearance of the word/phrase is not what you are looking for, try Find Next.

MENU

Companion Sites:

Statistics and Probability for Simulation

Topics in Descriptive Simulation Modeling

Techniques for Sensitivity Estimation

Simulation-based Optimization Techniques

Metamodeling and the Goal seeking Problems

"What-if" Analysis Techniques

Introduction & Summary

Computer system users, administrators, and designers usually have a goal of highest performance at lowest cost. Modeling and simulation of system design trade off is good preparation for design and engineering decisions in real world jobs.

In this Web site we study computer systems modeling and simulation. We need a proper knowledge of both the techniques of simulation modeling and the simulated systems themselves.

The scenario described above is but one situation where computer simulation can be effectively used. In addition to its use as a tool to better understand and optimize performance and/or reliability of systems, simulation is also extensively used to verify the correctness of designs. Most if not all digital integrated circuits manufactured today are first extensively simulated before they are manufactured to identify and correct design errors. Simulation early in the design cycle is important because the cost to repair mistakes increases dramatically the later in the product life cycle that the error is detected. Another important application of simulation is in developing "virtual environments" , e.g., for training. Analogous to the holodeck in the popular science-fiction television program Star Trek, simulations generate dynamic environments with which users can interact "as if they were really there." Such simulations are used extensively today to train military personnel for battlefield situations, at a fraction of the cost of running exercises involving real tanks, aircraft, etc.

Dynamic modeling in organizations is the collective ability to understand the implications of change over time. This skill lies at the heart of successful strategic decision process. The availability of effective visual modeling and simulation enables the analyst and the decision-maker to boost their dynamic decision by rehearsing strategy to avoid hidden pitfalls.

System Simulation is the mimicking of the operation of a real system, such as the day-to-day operation of a bank, or the value of a stock portfolio over a time period, or the running of an assembly line in a factory, or the staff assignment of a hospital or a security company, in a computer. Instead of building extensive mathematical models by experts, the readily available simulation software has made it possible to model and analyze the operation of a real system by non-experts, who are managers but not programmers.

A simulation is the execution of a model, represented by a computer program that gives information about the system being investigated. The simulation approach of analyzing a model is opposed to the analytical approach, where the method of analyzing the system is purely theoretical. As this approach is more reliable, the simulation approach gives more flexibility and convenience. The activities of the model consist of events, which are activated at certain points in time and in this way affect the overall state of the system. The points in time that an event is activated are randomized, so no input from outside the system is required. Events exist autonomously and they are discrete so between the execution of two events nothing happens. The SIMSCRIPT provides a process-based approach of writing a simulation program. With this approach, the components of the program consist of entities, which combine several related events into one process.

In the field of simulation, the concept of "principle of computational equivalence" has beneficial implications for the decision-maker. Simulated experimentation accelerates and replaces effectively the "wait and see" anxieties in discovering new insight and explanations of future behavior of the real system.

Consider the following scenario. You are the designer of a new switch for asynchronous transfer mode (ATM) networks, a new switching technology that has appeared on the marketplace in recent years. In order to help ensure the success of your product in this is a highly competitive field, it is important that you design the switch to yield the highest possible performance while maintaining a reasonable manufacturing cost. How much memory should be built into the switch? Should the memory be associated with incoming communication links to buffer messages as they arrive, or should it be associated with outgoing links to hold messages competing to use the same link? Moreover, what is the best organization of hardware components within the switch? These are but a few of the questions that you must answer in coming up with a design.

With the integration of artificial intelligence, agents and other modeling techniques, simulation has become an effective and appropriate decision support for the managers. By combining the emerging science of complexity with newly popularized simulation technology, the PricewaterhouseCoopers, Emergent Solutions Group builds a software that allows senior management to safely play out "what

if" scenarios in artificial worlds. For example, in a consumer retail environment it can be used to find out how the roles of consumers and employees can be simulated to achieve peak performance.

Statistics for Correlated Data

We concern ourselves with n realizations that are related to time, that is having n correlated observations; the **estimate of the mean** is given by

$$\text{mean} = \Sigma \; X_i / n,$$

where the sum is over i = 1 to n.

Let

$$A = \Sigma \; [1 - j/(m + 1)] \; \rho_{j,x}$$

where the sum is over j = 1 to m, then the **estimated variance** is:

$$[1 + 2A \,] \, S^2 / n$$

Where

S^2 = the usual variance estimate
$\rho_{j,x}$ = the jth coefficient of autocorrelation
m = the maximum time lag for which autocorrelations are computed, such that j = 1, 2, 3, ..., m

As a good rule of thumb, the maximum lag for which autocorrelations are computed should be approximately 2% of the number of n realizations, although each $\rho_{j,x}$ could be tested to determine if it is significantly different from zero.

Sample Size Determination: We can calculate the minimum sample size required by

$$n = [1 + 2A \,] \, S^2 \, t^2 / (\delta^2 \, \text{mean}^2)$$

Application: A pilot run was made of a model, observations numbered 150, the mean was 205.74 minutes and the variance S^2 = 101, 921.54, estimate of the lag coefficients were computed as: $\rho_{1,x} = 0.3301$ $\rho_{2,x} = 0.2993$, and $\rho_{3,x} = 0.1987$. Calculate the minimum sample size to assure the estimate lies within $\pm\delta$ = 10% of the true mean with α = 0.05.

$$n = \left[(1.96)^2 \, (101{,}921.54) \, \{1 + 2 \, [(1\text{-}1/4) \, 0.3301 + (1 - 2/4) \, 0.2993 + (1\text{-} \, 3/4) \, 0.1987]\}\right] / (0.1)^2 \, (205.74)^2$$

$\cup$ 1757

You may like using Statistics for Time Series, and Testing Correlation JavaScript.

What Is Central Limit Theorem?

For practical purposes, the main idea of the central limit theorem (CLT) is that the average of a sample of observations drawn from some population with any shape-distribution is approximately distributed as a normal distribution if certain conditions are met. In theoretical statistics there are several versions of the central limit theorem depending on how these conditions are specified. These are concerned with the types of assumptions made about the distribution of the parent population (population from which the sample is drawn) and the actual sampling procedure.

One of the simplest versions of the theorem says that if is a random sample of size n (say, n larger than 30) from an infinite population, finite standard deviation , then the standardized sample mean converges to a standard normal distribution or, equivalently, the sample mean approaches a normal distribution with mean equal to the population mean and standard deviation equal to standard deviation of the population divided by the square root of sample size n. In applications of the central limit theorem to practical problems in statistical inference, however, statisticians are more interested in how closely the approximate distribution of the sample mean follows a normal distribution for finite sample sizes, than the limiting distribution itself. Sufficiently close agreement with a normal distribution allows statisticians to use normal theory for making inferences about population parameters (such as the mean) using the sample mean, irrespective of the actual form of the parent population.

It is well known that whatever the parent population is, the standardized variable will have a distribution with a mean 0 and standard deviation 1 under random sampling. Moreover, if the parent population is normal, then it is distributed exactly as a standard normal variable for any positive integer n. The central limit theorem states the remarkable result that, even when the parent population is non-normal, the standardized variable is approximately normal if the sample size is large enough (say > 30). It is generally not possible to state conditions under which the approximation given by the central limit theorem works and what sample sizes are needed before the approximation becomes good enough. As a general guideline, statisticians have used the prescription that if the parent distribution is symmetric and relatively short-tailed, then the sample mean reaches approximate normality for smaller samples than if the parent population is skewed or long-tailed.

In this lesson, we will study the behavior of the mean of samples of different sizes drawn from a variety of parent populations. Examining sampling distributions of sample means computed from samples of different sizes drawn from a variety of distributions, allow us to gain some insight into the behavior of the sample mean under those specific conditions as well as examine the validity of the guidelines mentioned above for using the central limit theorem in practice.

Under certain conditions, in large samples, the sampling distribution of the sample mean can be approximated by a normal distribution. The sample size needed for the approximation to be adequate depends strongly on the shape of the parent distribution. Symmetry (or lack thereof) is particularly important. For a symmetric parent distribution, even if very different from the shape of a normal distribution, an adequate approximation can be obtained with small samples (e.g., 10 or 12 for the uniform distribution). For symmetric short-tailed parent distributions, the sample mean reaches approximate normality for smaller samples than if the parent population is skewed and long-tailed. In some extreme cases (e.g. binomial) samples sizes far exceeding the typical guidelines (e.g., 30) are needed for an adequate approximation. For some distributions without first and second moments (e.g., Cauchy), the central limit theorem does not hold.

```
 '' Central Limit Theorem Justification ''

 Preamble
        Define X, XBAR as a real 1-dimensional arrays
        Define I, J, M, N as integer variables
 End

Main
     Open 3 for output, Name = "CLT.OUT"
     Use 3 for output
     LET N=50
     LET M = 1000
     Reserve X(*) as N
     Reserves XBAR(*) as M
     For J = 1 to M
     DO
     For I = 1 to N
     DO
     X(I) = Uniform.f(1., 0., 1)
     Compute
     XBAR (j) as the average of X(i)
     Loop
```

```
Compute
Ave as the average of XBAR(j)
Compute
ST as the standard deviation of XBAR(j)
Loop
LL = Ave -1.96*ST/SQRT.F(real.f(M))
UL = Ave + 1.96*ST/SQRT.F(real.f(M))
For J =1 to M
Do
IF XBAR(j) <  LL  AND  XBAR(j) >  XU
Let Count = Count + 1
Always
Loop
Print 1 line with Count/M thus
The P-value is = *****
END
```

What Is a Least Squares Model?

Many problems in analyzing data involve describing how variables are related. The simplest of all models describing the relationship between two variables is a linear, or straight-line, model. The simplest method of fitting a linear model is to "eye-ball" a line through the data on a plot. A more elegant, and conventional method is that of "least squares", which finds the line minimizing the sum of distances between observed points and the fitted line.

Realize that fitting the "best" line by eye is difficult, especially when there is a lot of residual variability in the data.

Know that there is a simple connection between the numerical coefficients in the regression equation and the slope and intercept of regression line.

Know that a single summary statistic like a correlation coefficient does not tell the whole story. A scatter plot is an essential complement to examining the relationship between the two variables.

ANOVA: Analysis of Variance

The tests we have learned up to this point allow us to test hypotheses that examine the difference between only two means. Analysis of Variance or ANOVA will allow us to test the difference between 2 or more means. ANOVA does this by examining the ratio of variability between two conditions and variability within each condition. For example, say we give a drug that we believe will improve memory to a group of people and give a placebo to another group of people. We might measure memory performance by the number of words recalled from a list we ask everyone to memorize. A t-test would compare the likelihood of observing the difference in the mean number of words recalled for each group. An ANOVA test, on the other hand, would compare the variability that we observe between the two conditions to the variability observed within each condition. Recall that we measure variability as the sum of the difference of each score from the mean. When we actually calculate an ANOVA we will use a short-cut formula

Thus, when the variability that we predict (between the two groups) is much greater than the variability we don't predict (within each group) then we will conclude that our treatments produce different results.

Exponential Density Function

An important class of decision problems under uncertainty concerns the chance between events. For example, the chance of the length of time to next breakdown of a machine not exceeding a certain time, such as the copying machine in your office not to break during this week.

Exponential distribution gives distribution of time between independent events occurring at a constant rate. Its density function is:

$$f(t) = \lambda \exp(-\lambda t),$$

where λ is the average number of events per unit of time, which is a positive number.

The mean and the variance of the random variable t (time between events) are $1/\lambda$, and $1/\lambda^2$, respectively.

Applications include probabilistic assessment of the time between arrival of patients to the emergency room of a hospital, and arrival of ships to a particular port.

Comments: Special case of both Weibull and gamma distributions.

You may like using Exponential Applet to perform your computations.

You may like using the following Lilliefors Test for Exponentially to perform the goodness-of-fit test.

Poisson Process

An important class of decision problems under uncertainty is characterized by the small chance of the occurrence of a particular event, such as an accident. Gives probability of exactly x independent occurrences during a given period of time if events take place independently and at a constant rate. May also represent number of occurrences over constant areas or volumes. The following statements describe the *Poisson Process*:

1. The occurrences of the events are independent. The occurrence of events from a set of assumptions in an interval of space or time has no effect on the probability of a second occurrence of the event in the same, or any other, interval.
2. Theoretically, an infinite number of occurrences of the event must be possible in the interval.
3. The probability of the single occurrence of the event in a given interval is proportional to the length of the interval.
4. In any infinitesimally small portion of the interval, the probability of more than one occurrence of the event is negligible.

Poisson process are often used, for example in quality control, reliability, insurance claim, incoming number of telephone calls, and queuing theory.

An Application: One of the most useful applications of the Poisson Process is in the field of queuing theory. In many situations where queues occur it has been shown that the number of people joining the queue in a given time period follows the Poisson model. For example, if the rate of arrivals to an emergency room is λ per unit of time period (say 1 hr), then:

$$P(\text{n arrivals}) = \lambda^n e^{-\lambda} / n!$$

The mean and variance of random variable n are both λ . However if the mean and variance of a random variable having equal numerical values, then it is not necessary that its distribution is a Poisson.

Applications:

$$P(\text{0 arrival}) = e^{-\lambda}$$
$$P(\text{1 arrival}) = \lambda e^{-\lambda} / 1!$$
$$P(\text{2 arrival}) = \lambda^2 e^{-\lambda} / 2!$$

and so on. In general:

$$P(\text{n+1 arrivals}) = \lambda \quad Pr(\text{n arrivals}) / n.$$

You may like using Poisson Applet to perform your computations.

Goodness-of-Fit for Poisson

Replace the numerical example data with your up-to-14 pairs of **Observed** values & their **frequencies**, and then click the **Calculate** button. Blank boxes are not included in the calculations.

In entering your data to move from cell to cell in the data-matrix use the **Tab key** not arrow or enter keys.

Observed Xi	0	1	2	3	4	5								
Frequencies Fi	35	33	20	6	1	0								
							CALCULATE	CLEAR						
								Estimated Rate						
								Chi-square						
								P-value						
								Conclusion						

For Technical Details, Back to:
Statistical Thinking for Decision Making

Uniform Density Function

Application: Gives probability that observation will occur within a particular interval when probability of occurrence within that interval is directly proportional to interval length.

Example: Used to generate random numbers in sampling and Monte Carlo simulation.

Comments: Special case of beta distribution.

The mass function of geometric mean of n independent uniforms [0,1] is:

$$P(X = x) = n\, x^{(n-1)} (\text{Log}[1/x^n])^{(n-1)} / (n-1)!.$$

$z_L = [U^L-(1-U)^L] / L$ is said to have Tukey's symmetrical λ -distribution.

You may like using Uniform Applet to perform your computations.

Some Useful SPSS Commands

Test for Binomial:

```
NPAR TEST BINOMIAL(p)=GENDER(0, 1)
```

Gooness-of-fit for discrete r.v.:

```
   NPAR TEST CHISQUARE=X (1,3)/EXPECTED=20 30 50
```

Needed information to perform the t-test:

```
   DISCRIPTIVES X
      /STATISTICS= 1 2
```

Two population t-test

```
   T-TEST GROUPS=GENDER(1,2)/VARIABLES=X
```

Plot x vs y:

```
PLOT FORMAT=REGRESSION/SYMBOLS='*'
  /TITLE='PLOT OF Y ON X'
  /VERTICAL='Y'
  /HORIZONTAL='X'
  /PLOT=Y WITH X
```

RANDOM VARIATES GENERATORS:

```
  LOOP #I = 1 to 100.
(normal with mean = 0 and std = 1)
  COMPUTE XNORM = RV.NORMAL(0,1)
 (chi-square with 2 d.f.)
  COMPUTE XCHISQ=RV.CHISQ(2)
(exponential with mean 2)
  COMPUTE XEXPON = RV.EXP(1/2)
(binomial n = 10 and p = .50
  COMPUTE XBINOM = RV.BINOM(10, 0.5).
               END CASE
         END LOOP
        END FILE
END INOUT PROGRAM
EXAMINE VARS=ALL
        /STATISTICS
        /HISTOGRAM(NORMSAL)= XNORM
        /HISTOGRAM(NORMSAL)= XCHISQ
        /HISTOGRAM(NORMSAL)= XEXPON
        /HISTOGRAM(NORMSAL)= XBINOM
```

NORMAL RANOM VARIATE GENERATOR, K-S, AND RUNS TESTS:

```
SPSS/OUTPUT=HW3.OUT
TITLE     'GENERATING FROM NORMAL 0,1'
INPUT PROGRAM
LOOP I=1 TO 50
COMPUTE X2=NORMAL(1)
END CASE
END LOOP
END FILE
END INPUT PROGRAM
VAR LABLE
                X2 'NORMAL VARIATE'
LIST CASE       CASE=50/VARIABLE=ALL//
CONDESCRIPTIVE  X2(ZX2)
STATISTICS      ALL
FREQUENCIES     VARIABLE=ZX2/FORMAT=NOTABLE/
                HISTOGRAM MIN(-3.0) MAX(+3.0) INCREMENT(0.2)/
NPAR TESTS      RUNS(MEAN)=ZX2/
NPAR TESTS      K-S(NORMAL,0.0,1.0)=ZX2/
SAMPLE 10 FROM 50
LIST CASE       CASE=10/VARIABLES=X2,ZX2/
```

```
FINISH
```

K-S LILLIEFORS TEST FOR NORMALITY:

```
$SPSS/OUTPUT=L.OUT
TITLE    'K-S LILLIEFORS TEST FOR NORMALITY'
DATA LIST     FREE FILE='L.DAT'/X
VAR LABELS
            X 'SAMPLE VALUES'
LIST CASE   CASE=20/VARIABLES=ALL
CONDESCRIPTIVE X(ZX)
LIST CASE CASE=20/VARIABLES=X ZX/
SORT CASES BY ZX(A)
RANK VARIABLES=ZX/RFRACTION INTO CRANK/TIES=HIGH
COMPUTE Y=CDFNORM(ZX)
COMPUTE SPROB=CRANK
COMPUTE DA=Y-SPROB
COMPUTE DB=Y-LAG(SPROB,1)
COMPUTE DAABS=ABS(DA)
COMPUTE DBABS=ABS(DB)
COMPUTE LILLSTAT=MAX(DAABS,DBABS)
LIST VARIABLES=X,ZX,Y,SPROB,DA,DB
LIST VARIABLES=LILLSTAT
SORT CASES BY LILLSTAT(D)
LIST CASES CASE=1/VARIABLES=LILLSTAT
FINISH
```

K-S TEST FOR EXPONENTIAL DATA WITH MEAN = 1

```
DATA LIST  FREE FILE='Ex.DAT'/ X
DESCRIPTIVES VARIABLES=X
STATISTICS MEAN
COMPUTE Y=1.-EXP(-X)
NPAR TESTS K-S(UNIFORM, 0.0, 1.0) Y  (For large sample size)
FINISH
```

For more SPSS programs useful to simulation input/output analysis, visit Data Analysis Routines.

Random Number Generators

Classical uniform random number generators have some major defects, such as, short period length and lack of higher dimension uniformity. However, nowadays there are a class of rather complex generators which is as efficient as the classical generators while enjoy the property of a much longer period and of a higher dimension uniformity.

Computer programs that generate "random" numbers use an algorithm. That means if you know the algorithm and the seedvalues you can predict what numbers will result. Because you can predict the numbers they are not truly random - they are pseudorandom. For statistical purposes "good" pseudorandom numbers generators are good enough.

```
      real function random()
c
c     Algorithm AS 183 Appl. Statist. (1982) vol.31, no.2
c
c     Returns a pseudo-random numbers with rectangular distribution.
c     between 0 and 1.   The cycle length is 6.95E+12 (See page 123
c     of Applied Statistics (1984) vol.33), not as claimed in the
c     original article.
c
c     IX, IY and IZ should be set to integer values between 1 and
c     30000 before the first entry.
c
c     Integer arithmetic up to 30323 is required.
c
      integer ix, iy, iz
      common /randc/ ix, iy, iz
c
```

```
      ix = 171 * mod(ix, 177) - 2 * (ix / 177)
      iy = 172 * mod(iy, 176) - 35 * (iy / 176)
      iz = 170 * mod(iz, 178) - 63 * (iz / 178)
c
      if (ix .lt. 0) ix = ix + 30269
      if (iy .lt. 0) iy = iy + 30307      if (iz .lt. 0) iz = iz + 30323
c
c     If integer arithmetic up to 5212632 is available, the preceding
c     6 statements may be replaced by:
c
c     ix = mod(171 * ix, 30269)
c     iy = mod(172 * iy, 30307)
c     iz = mod(170 * iz, 30323)
c
      random = mod(float(ix) / 30269. + float(iy) / 30307. +
     +                        float(iz) / 30323., 1.0)
      return
      end
c
c
c
c
        real function uniform()
c
c       Generate uniformly distributed random numbers using the 32-bit
c       generator from figure 3 of: L'Ecuyer, P., 1988.
c       The cycle length is claimed to be 2.30584E+18
c       Seeds can be set by calling the routine set_uniform
c       It is assumed that the Fortran compiler supports long variable
c       names, and integer*4.
c
        integer*4 z, k, s1, s2
        common /unif_seeds/ s1, s2
        save /unif_seeds/
c
        k = s1 / 53668
        s1 = 40014 * (s1 - k * 53668) - k * 12211
        if (s1 .lt. 0) s1 = s1 + 2147483563
c
        k = s2 / 52774
        s2 = 40692 * (s2 - k * 52774) - k * 3791
        if (s2 .lt. 0) s2 = s2 + 2147483399
c
        z = s1 - s2
        if (z .lt. 1) z = z + 2147483562
c
        uniform = z / 2147483563.
        return
        end


        subroutine set_uniform(seed1, seed2)
c
c       Set seeds for the uniform random number generator.
c
        integer*4 s1, s2, seed1, seed2
        common /unif_seeds/ s1, s2
        save /unif_seeds/

        s1 = seed1
        s2 = seed2
        return
        end
```

The Random Number Generator RANECU

A FORTRAN code for a generator of uniform random numbers on [0,1]. RANECU is multiplicative linear congruential generator suitable for a 16-bit platform. It combines three simple generators, and has a period exceeding 81012.

It is constructed for more efficient use by providing for a sequence of such numbers, LEN in total, to be returned in a single call. A set of three non-zero integer seeds can be supplied, failing which a default set is employed. If supplied, these three seeds, in order, should lie in the ranges [1,32362], [1,31726] and [1,31656] respectively.

```
        SUBROUTINE RANECU (RVEC,LEN)
C Portable random number generator for 16 bit computer.
C Generates a sequence of LEN pseudo-random numbers, returned in
C RVEC.
          DIMENSION RVEC(*)
          SAVE ISEED1,ISEED2, ISEED3
          DATA ISEED1,ISEED2,ISEED3/1234, 5678, 9876/
C Default values, used if none suppliedvia an ENTRY
C call at RECUIN
          DO 100 I = 1,LEN
         K=ISEED1/206
         ISEED1 = 157 * (ISEED1 - K * 206) - K * 21
         IF(ISEED1.LT.0) ISEED1=ISEED1+32363
         K=ISEED2/217
         ISEED2 = 146 * (ISEED2 - K*217) - K* 45
         IF(ISEED2.LT.O) ISEED2=ISEED2+31727
         K=ISEED3/222
         ISEED3 = 142 * (ISEED3 - K *222) - K * 133
         IF(ISEED3.LT.0) ISEED3=ISEED3+31657
         IZ=ISEED1-ISEED2
         IF(IZ.GT.706)IZ = Z - 32362
         IZ = 1Z+ISEED3
         IF(IZ.LT.1)IZ = 1Z + 32362
         RVEC(I)=REAL(IZ) * 3.0899E - 5
100      CONTINUE
         RETURN
         ENTRY RECUIN(IS1, IS2, IS3)
         ISEED1=IS1
         ISEED2=IS2
         ISEED3=IS3
         RETURN
         ENTRY RECUUT(IS1,IS2,IS3)
         IS1=ISEED1
         IS2=ISEED2
         IS3=ISEED3
         RETURN
         END
```

The Shuffling Routine in Visual Basic

```
Dim Ran0Y As Double
Dim Ran0V(97) As Double

Function RandShuffle(idum As Integer)
    Dim dum As Double
    Dim j As Integer
    If idum <  0 Then
        Randomize (-idum)
        For j = 1 To 97
            dum = Rnd()
        Next
        For j = 1 To 97
            Ran0V(j) = Rnd()
        Next
        Ran0Y = Rnd()
    End If
    Ran0Y = Rnd()
```

```
    j = 1 + Int(97 * Ran0Y)
    If (j >  97) Or (j <  1) Then
        MsgBox "Error"
    End If
    Ran0Y = Ran0V(j)
    RandShuffle = Ran0Y
    Ran0V(j) = Rnd()
End Function
```

The Square Histogram Method
We are given a histogram, with vertical bars having heights proportional to the probability with which we want to produce a value indicated by the label at the base.

A simple such histogram, layed flat, might be:

```
a:********************************32
b:************************27
c:************************26
d:************12
e:***3
```

The idea is to cut the bars into pieces then reassemble them into a square histogram, all heights equal, with each final bar having a lower part, as well as an upper part indicating where it came from. A single uniform random variable U can then be used to choose one of the final bars and to indicate whether to use the lower or upper part. There are many ways to do this cutting and reassembling; the simplest seems to be the Robin Hood Algorithm: Take from richest to bring the poorest up to average.

STEP 1: The original (horizontal) histogram, average "height" 20:

```
a:******************************** 32
b:************************       27
c:************************         26
d:************                     12
e:***                             3
```

Take 17 from strip 'a' to bring strip 'e' up to average. Record donor and use old 'poor' level to mark lower part of donee:

```
a:***************            15
b:************************* 27
c:************************   26
d:************               12
e:**|****************|       20 (a)
```

Then bring 'd' up to average with donor 'b'. Record donor and use old 'poor' level to mark lower part of donee:

```
a:***************           15
b:******************        19
c:************************  26
d:***********|*******|      20 (b)
e:*****|*************|      20 (a)
```

Then bring 'a' up to average with donor 'c'. Record donor and use old 'poor' level to mark lower part of donee:

```
a:***************|***|      20(c)
b:******************       19
c:*********************     21
d:***********|*******|      20(b)
e:*****|*************|      20(a)
```

Finally, bring 'b' up to average with donor 'c'. Record donor and use old 'poor' level to mark lower part of donee:

```
a:**************|****|      20(c)
b:******************||      20(c)
c:********************|      20
d:************|********|      20(b)
e:******|*************|      20(a)
```

We now have a "squared histogram", i.e., a rectangle with 4 strips of equal area, each strip with two regions. A single uniform variate U can be used to generate a,b,c,d,e with the required probabilities, .32, .27, .26, .12 .06.

Setup: Make tables,

```
V[1]=a    K[1]=c    T[1]=0+16/20
V[2]=b    K[2]=c    T[2]=1+19/20
V[3]=c    K[3]=c    T[3]=2+20/20
V[4]=d    K[4]=b    T[4]=3+12/20
V[5]=e    K[5]=a    T[5]=4+ 6/20
```

Generation Process:

Let j be the integer part of 1+5*U, with U uniform in (0,1). If U < T[j] return V[j], else return V[K[j]]. In many applications no V table is necessary: V[i]=i and the generating procedure becomes If U < T[j] return j, else return K[j].

For more, visit the Web site: Modeling & Simulation Resources.

References & Further Readings:

Aiello W., S. Rajagopalan, and R. Venkatesan, Design of practical and provably good random number generators, *Journal of Algorithms*, 29, 358-389, 1998.
Dagpunar J., *Principles of Random Variate Generation*, Clarendon, 1988.
Fishman G., *Monte Carlo*, Springer, 1996.
James, Fortran version of L'Ecuyer generator, *Comput. Phys. Comm.*, 60, 329-344, 1990.
Knuth D., *The Art of Computer Programming, Vol. 2*, Addison-Wesley, 1998.
L'Ecuyer P., Efficient and portable combined random number generators, Comm. ACM, 31, 742-749, 774, 1988.
L'Ecuyer P., Uniform random number generation, *Ann. Op. Res.*, 53, 77-120, 1994.
L'Ecuyer P., Random number generation. In *Handbook on Simulation*, J. Banks (ed.), Wiley, 1998.
Maurer U., A universal statistical test for random bit generators, *J. Cryptology*, 5, 89-105, 1992.
Sobol' I., and Y. Levitan, A pseudo-random number generator for personal computers, *Computers & Mathematics with Applications*, 37(4), 33-40, 1999.
Tsang W-W., A decision tree algorithm for squaring the histogram in random number generation, *Ars Combinatoria*, 23A, 291-301, 1987

Test for Randomness

We need to test for both randomness as well as uniformity. The tests can be classified in 2 categories: Empirical or statistical tests, and theoretical tests.
Theoretical tests deal with the properties of the generator used to create the realization with desired distribution, and do not look at the number generated at all. For example, we would not use a generator with poor qualities to generate random numbers.
Statistical tests are based solely on the random observations produced.

Test for Randomness:

A. Test for independence:
Plot the x_i realization vs x_{i+1}. If there is independence, the graph will not show any distinctive patterns at all, but will be perfectly scattered.

B. Runs tests.(run-ups, run-downs):
This is a direct test of the independence assumption. There are two test statistics to consider: one based on a normal approximation and another using numerical approximations.

Test based on Normal approximation:
Suppose you have N random realizations. Let a be the total number of runs in a sequence. If the number of positive and negative runs are greater than say 20, the distribution of a is reasonably

approximated by a Normal distribution with mean (2N - 1) /3 and (16N - 29) / 90. Reject the hypothesis of independence or existence of runs if | Zo| > Z(1-alpha/2) where Zo is the Z score.

C. Correlation tests:
Do the random numbers exhibit discernible correlation? Compute the sample Autcorrelation Function.

Frequency or Uniform Distribution Test:
Use Kolmogorov-Smirimov test to determine if the realizations follow a U(0,1)

References & Further Readings:

Headrick T., Fast fifth-order polynomial transforms for generating univariate and multivariate nonnormal distributions, *Computational Statistics and Data Analysis*, 40 (4), 685-711, 2002.
Karian Z., and E. Dudewicz, *Modern Statistical Systems and GPSS Simulation*, CRC Press, 1998.
Kleijnen J., and W. van Groenendaal, *Simulation: A Statistical Perspective*, Wiley, Chichester, 1992
Korn G., Real statistical experiments can use simulation-package software, *Simulation Modelling Practice and Theory*, 13(1), 39-54, 2005.
Lewis P., and E. Orav, *Simulation Methodology for Statisticians, Operations Analysts, and Engineers*, Wadsworth Inc., 1989
Madu Ch., and Ch-H. Kuei, *Experimental Statistical Designs and Analysis in Simulation Modeling*, Greenwood Publishing Group, 1993.
Pang K., Z. Yang, S. Hou, and P. Leung, Non-uniform random variate generation by the vertical strip method, *European Journal of Operational Research*, 142(3), 595-609, 2002.
Robert C., and G. Casella, *Monte Carlo Statistical Methods*, Springer, 1999.

Modeling & Simulation

Simulation in general is to pretend that one deals with a real thing while really working with an imitation. In operations research the imitation is a computer model of the simulated reality. A flight simulator on a PC is also a computer model of some aspects of the flight: it shows on the screen the controls and what the "pilot" (the youngster who operates it) is supposed to see from the "cockpit" (his armchair).

Why to use models? To fly a simulator is safer and cheaper than the real airplane. For precisely this reason, models are used in industry commerce and military: it is very costly, dangerous and often impossible to make experiments with real systems. Provided that models are adequate descriptions of reality (they are valid), experimenting with them can save money, suffering and even time.

When to use simulations? Systems that change with time, such as a gas station where cars come and go (called dynamic systems) and involve randomness. Nobody can guess at exactly which time the next car should arrive at the station, are good candidates for simulation. Modeling complex dynamic systems theoretically need too many simplifications and the emerging models may not be therefore valid. Simulation does not require that many simplifying assumptions, making it the only tool even in absence of randomness.

How to simulate? Suppose we are interested in a gas station. We may describe the behavior of this system graphically by plotting the number of cars in the station; the state of the system. Every time a car arrives the graph increases by one unit while a departing car causes the graph to drop one unit. This graph (called sample path), could be obtained from observation of a real station, but could also be artificially constructed. Such artificial construction and the analysis of the resulting sample path (or more sample paths in more complex cases) consists of the simulation.

Types of simulations: Discrete event. The above sample path consisted of only horizontal and vertical lines, as car arrivals and departures occurred at distinct points of time, what we refer to as events. Between two consecutive events, nothing happens - the graph is horizontal. When the number of events are finite, we call the simulation "discrete event."

In some systems the state changes all the time, not just at the time of some discrete events. For example, the water level in a reservoir with given in and outflows may change all the time. In such cases "continuous simulation" is more appropriate, although discrete event simulation can serve as an approximation.

Further consideration of discrete event simulations.

How is simulation performed? Simulations may be performed manually. Most often, however, the system model is written either as a computer program (for an example click here) or as some kind of input into simulator software.

System terminology:

State: A variable characterizing an attribute in the system such as level of stock in inventory or number of jobs waiting for processing.

Event: An occurrence at a point in time which may change the state of the system, such as arrival of a customer or start of work on a job.

Entity: An object that passes through the system, such as cars in an intersection or orders in a factory. Often an event (e.g., arrival) is associated with an entity (e.g., customer).

Queue: A queue is not only a physical queue of people, it can also be a task list, a buffer of finished goods waiting for transportation or any place where entities are waiting for something to happen for any reason.

Creating: Creating is causing an arrival of a new entity to the system at some point in time.

Scheduling: Scheduling is the act of assigning a new future event to an existing entity.

Random variable: A random variable is a quantity that is uncertain, such as interarrival time between two incoming flights or number of defective parts in a shipment.

Random variate: A random variate is an artificially generated random variable.

Distribution: A distribution is the mathematical law which governs the probabilistic features of a random variable.

A Simple Example: Building a simulation gas station with a single pump served by a single service man. Assume that arrival of cars as well their service times are random. At first identify the:

states: number of cars waiting for service and number of cars served at any moment

events: arrival of cars, start of service, end of service

entities: these are the cars

queue: the queue of cars in front of the pump, waiting for service

random realizations: interarrival times, service times

distributions: we shall assume exponential distributions for both the interarrival time and service time.

Next, specify what to do at each event. The above example would look like this: At event of entity arrival: Create next arrival. If the server is free, send entity for start of service. Otherwise it joins the queue. At event of service start: Server becomes occupied. Schedule end of service for this entity. At event of service end: Server becomes free. If any entities waiting in queue: remove first entity from the queue; send it for start of service.

Some initiation is still required, for example, the creation of the first arrival. Lastly, the above is translated into code. This is easy with an appropriate library which has subroutines for creation, scheduling, proper timing of events, queue manipulations, random variate generation and statistics collection.

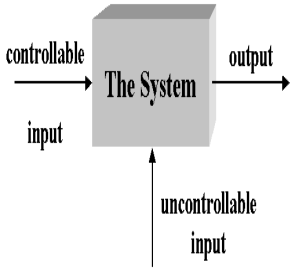
How to simulate? Besides the above, the program records the number of cars in the system before and after every change, together with the length of each event.

Development of Systems Simulation

Discrete event systems (DES) are dynamic systems which evolve in time by the occurrence of events at possibly irregular time intervals. DES abound in real-world applications. Examples include traffic systems, flexible manufacturing systems, computer-communications systems, production lines, coherent lifetime systems, and flow networks. Most of these systems can be modeled in terms of discrete events whose occurrence causes the system to change from one state to another. In designing, analyzing and operating such complex systems, one is interested not only in performance evaluation but also in sensitivity analysis and optimization.

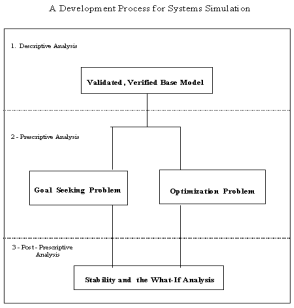
A typical stochastic system has a large number of control parameters that can have a significant impact on the performance of the system. To establish a basic knowledge of the behavior of a system under variation of input parameter values and to estimate the relative importance of the input parameters, sensitivity analysis applies small changes to the nominal values of input parameters. For systems simulation, variations of the input parameter values cannot be made infinitely small. The sensitivity of the performance measure with respect to an input parameter is therefore defined as (partial) derivative.

Sensitivity analysis is concerned with evaluating sensitivities (gradients, Hessian, etc.) of performance measures with respect to parameters of interest. It provides guidance for design and operational decisions and plays a pivotal role in identifying the most significant system parameters, as well as bottleneck subsystems. I have carried out research in the fields of sensitivity analysis and stochastic optimization of discrete event systems with an emphasis on computer simulation models. This part of lecture is dedicated to the estimation of an entire response surface of complex discrete event systems (DES) from a single sample path (simulation), such as the expected waiting time of a customer in a queuing network, with respect to the controllable parameters of the system, such as service rates, buffer sizes and routing probabilities. With the response surfaces at hand, we are able to perform sensitivity analysis and optimization of a DES from a single simulation, that is, to find the optimal parameters of the system and their sensitivities (derivatives), with respect to uncontrollable system parameters, such as arrival rates in a queuing network. We identified three distinct processes. Descriptive Analysis includes: Problem Identification & Formulation, Data Collection and Analysis, Computer Simulation Model Development, Validation, Verification and Calibration, and finally Performance Evaluation. Prescriptive Analysis: Optimization or Goal Seeking. These are necessary components for Post-prescriptive Analysis: Sensitivity, and What-If Analysis. The prescriptive simulation attempts to use simulation to prescribe decisions required to obtain specified results. It is subdivided into two topics- Goal Seeking and Optimization. Recent developments on "single-run" algorithms for the needed sensitivities (i.e. gradient, Hessian, etc.) make the prescriptive simulation feasible.



Click on the image **to enlarge** it and THEN print it.

Problem Formulation: Identify controllable and uncontrollable inputs. Identify constraints on the decision variables. Define measure of system performance and an objective function. Develop a preliminary model structure to interrelate the inputs and the measure of performance.



Click on the image **to enlarge** it and THEN print it.

Data Collection and Analysis: Regardless of the method used to collect the data, the decision of how much to collect is a trade-off between cost and accuracy.

Simulation Model Development: Acquiring sufficient understanding of the system to develop an appropriate conceptual, logical and then simulation model is one of the most difficult tasks in simulation analysis.

Model Validation, Verification and Calibration: In general, verification focuses on the internal consistency of a model, while validation is concerned with the correspondence between the model and the reality. The term validation is applied to those processes which seek to determine whether or not a simulation is correct with respect to the "real" system. More prosaically, validation is concerned with the question "Are we building the right system?". Verification, on the other hand, seeks to answer the question "Are we building the system right?" Verification checks that the implementation of the simulation model (program) corresponds to the model. Validation checks that the model corresponds to reality. Calibration checks that the data generated by the simulation matches real (observed) data.

Validation: The process of comparing the model's output with the behavior of the phenomenon. In other words: comparing model execution to reality (physical or otherwise)
Verification: The process of comparing the computer code with the model to ensure that the code is a correct implementation of the model.
Calibration: The process of parameter estimation for a model. Calibration is a tweaking/tuning of existing parameters and usually does not involve the introduction of new ones, changing the model structure. In the context of optimization, calibration is an optimization procedure involved in system identification or during experimental design.

Input and Output Analysis: Discrete-event simulation models typically have stochastic components that mimic the probabilistic nature of the system under consideration. Successful input modeling requires a close match between the input model and the true underlying probabilistic mechanism associated with the system. The input data analysis is to model an element (e.g., arrival process, service times) in a discrete-event simulation given a data set collected on the element of interest. This stage performs intensive error checking on the input data, including external, policy, random and deterministic variables. System simulation experiment is to learn about its behavior. Careful planning, or designing, of simulation experiments is generally a great help, saving time and effort by providing efficient ways to estimate the effects of changes in the model's inputs on its outputs. Statistical experimental-design methods are mostly used in the context of simulation experiments.

Performance Evaluation and What-If Analysis: The `what-if' analysis is at the very heart of simulation models.

Sensitivity Estimation: Users must be provided with affordable techniques for sensitivity analysis if they are to understand which relationships are meaningful in complicated models.

Optimization: Traditional optimization techniques require gradient estimation. As with sensitivity analysis, the current approach for optimization requires intensive simulation to construct an approximate surface response function. Incorporating gradient estimation techniques into convergent algorithms such as Robbins-Monroe type algorithms for optimization purposes, will be considered.

Gradient Estimation Applications: There are a number of applications which measure sensitivity information, (i.e., the gradient, Hessian, etc.), Local information, Structural properties, Response surface generation, Goal-seeking problem, Optimization, What-if Problem, and Meta-modelling

Report Generating: Report generation is a critical link in the communication process between the model and the end user.

A Classification of Stochastic Processes

A stochastic process is a probabilistic model of a system that evolves randomly in time and space. Formally, a stochastic process is a collection of random variables {X(t), t ⌊ T} all defined on a common sample (probability) space. The X(t) is the state while (time) t is the index that is a member of set T.

Examples are the delay {D(i), i = 1, 2, ...} of the ith customer and number of customers {Q(t), T ³ 0} in the queue at time t in an M/M/1 queue. In the first example, we have a discrete- time, continuous state, while in the second example the state is discrete and time in continuous.

The following table is a classification of various stochastic processes. The man made systems have mostly discrete state. Monte Carlo simulation deals with discrete time while in discrete even system simulation the time dimension is continuous, which is at the heart of this site.

		Change in the States of the System	
		Continuous	**Discrete**
Time	**Continuous**	Level of water behind a dam	Number of customers in a bank
	Discrete	Weekdays' range of temperature	Sales at the end of the day

A Classification of Stochastic Processes

Simulation Output Data and Stochastic Processes

To perform statistical analysis of the simulation output we need to establish some conditions, e.g. output data must be a covariance stationary process (e.g. the data collected over n simulation runs).

Stationary Process (strictly stationary): A stationary stochastic process is a stochastic process {X(t), t ⌊ T} with the property that the joint distribution all vectors of h dimension remain the same for any fixed h.

First Order Stationary: A stochastic process is a first order stationary if expected of X(t) remains the same for all t.

For example in economic time series, a process is first order stationary when we remove any kinds of trend by some mechanisms such as differencing.

Second Order Stationary: A stochastic process is a second order stationary if it is first order stationary and covariance between X(t) and X(s) is function of t-s only.

Again, in economic time series, a process is second order stationary when we stabilize also its variance by some kind of transformations such as taking square root.

Clearly, a stationary process is a second order stationary, however the reverse may not hold.

In simulation output statistical analysis we are satisfied if the output is *covariance stationary*.

Covariance Stationary: A covariance stationary process is a stochastic process $\{X(t), t \geq T\}$ having finite second moments, i.e. expected of $[X(t)]^2$ be finite.

Clearly, any stationary process with finite second moment is covariance stationary. A stationary process may have no finite moment whatsoever.

Since a Gaussian process needs a mean and covariance matrix only, it is stationary (strictly) if it is covariance stationary.

Two Contrasting Stationary Process:

Consider the following two extreme stochastic processes:

- A sequence $Y_0, Y_1,.....$, of independent identically distributed, random-value sequence is a stationary process, if its common distribution has a finite variance then the process is covariance stationary.

- Let Z be a single random variable with known distribution function, and set $Z_0 = Z_1 = ...Z$. Note that in a realization of this process, the first element, Z_0, may be random but after that there is no randomness. The process $\{Z_i, i = 0, 1, 2, ..\}$ is stationary if Z has a finite variance.

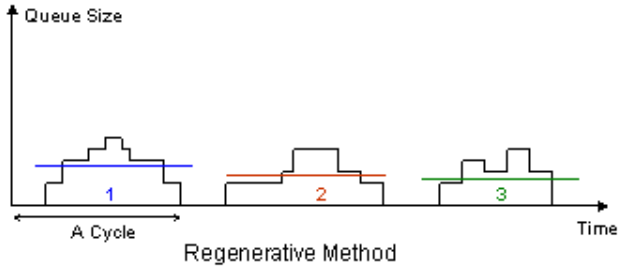
Output data in simulation fall between these two type of process. Simulation outputs are identical, and mildly correlated (how mild? It depends on e.g. in a queueing system how large is the traffic intensity ρ). An example could be the delay process of the customers in a queueing system.

Techniques for the Steady State Simulation

Unlike in queuing theory where steady state results for some models are easily obtainable, the steady state simulation is not an easy task. The opposite is true for obtaining results for the transient period (i.e., the warm-up period).

Gather steady state simulation output requires statistical assurance that the simulation model reached the steady state. The main difficulty is to obtain independent simulation runs with exclusion of the transient period . The two technique commonly used for steady state simulation are the Method of Batch means, and the Independent Replication.

None of these two methods is superior to the other in all cases. Their performance depend on the magnitude of the traffic intensity. The other available technique is the Regenerative Method, which is mostly used for its theoretical nice properties, however it is rarely applied in actual simulation for obtaining the steady state output numerical results.



Regenerative Method

Suppose you have a regenerative simulation consisting of m cycles of size $n_1, n_2,\ldots n_m$, respectively. The cycle sums is:

$$y_i = \Sigma \; x_{ij} / n_i, \quad \text{the sum is over } j=1, 2, ..,n_i$$

The overall estimate is:

$$\text{Estimate} = \Sigma\ y_i / \Sigma\ n_i, \quad \text{the sums are over } i=1, 2, ..,m$$

The 100(1-α /2)% confidence interval using the Z-table (or T-table, for m less than, say 30), is:

$$\text{Estimate} \pm\ Z.\ S/ (n.\ m^{½})$$

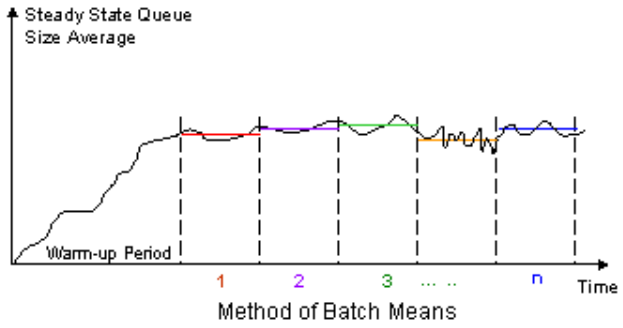
where,

$$n = \Sigma\ n_i /m, \quad \text{the sum is over } i=1, 2, ..,m$$

and the variance is:

$$S^2 = \Sigma\ (y_i - n_i\ .\ \text{Estimate})^2/(m-1), \quad \text{the sum is over } i=1, 2, ..,m$$

Method of Batch Means: This method involves only one very long simulation run which is suitably subdivided into an initial transient period and n batches. Each of the batch is then treated as an independent run of the simulation experiment while no observation are made during the transient period which is treated as warm-up interval. Choosing a large batch interval size would effectively lead to independent batches and hence, independent runs of the simulation, however since number of batches are few on cannot invoke the central limit theorem to construct the needed confidence interval. On the other hand, choosing a small batch interval size would effectively lead to significant correlation between successive batches therefore cannot apply the results in constructing an accurate confidence interval.



Method of Batch Means

Suppose you have n equal batches of m observations each. The means of each batch is:

$$\text{mean}_i = \Sigma\ x_{ij} / m, \quad \text{the sum is over } j=1, 2, ..,m$$

The overall estimate is:

$$\text{Estimate} = \Sigma\ \text{mean}_i / n, \quad \text{the sum is over } i=1, 2, ..,n$$

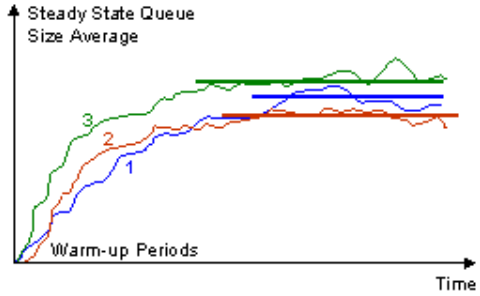
The 100(1-α /2)% confidence interval using the Z-table (or T-table, for n less than, say 30), is:

$$\text{Estimate} \pm\ Z.\ S$$

where the variance is:

$$S^2 = \Sigma\ (\text{mean}_i - \text{Estimate})^2/(n-1), \quad \text{the sum is over } i=1, 2, ..,n$$

Method of Independent Replications: This method is the most popularly used for systems with short transient period. This method requires independent runs of the simulation experiment different initial random seeds for the simulators' random number generator. For each independent replications of the simulation run it transient period is removed. For the observed intervals after the transient period data is collected and processed for the point estimates of the performance measure and for its subsequent confidence interval.



Method of Independent Replications

Suppose you have n replications with of m observations each. The means of each replication is:

$$\text{mean}_i = \Sigma\ x_{ij} / m, \quad \text{the sum is over } j=1, 2, ..,m$$

The overall estimate is:

$$\text{Estimate} = \Sigma\ \text{mean}_i / n, \quad \text{the sum is over } i=1, 2, ..,n$$

The 100(1-α /2)% confidence interval using the Z-table (or T-table, for n less than, say 30), is:

$$\text{Estimate} \pm\ Z.\ S$$

where the variance is:

$$S^2 = \Sigma\ (\text{mean}_i - \text{Estimate})^2/(n-1), \quad \text{the sum is over } i=1, 2, ..,n$$

Further Reading:

Sherman M., and D. Goldsman, Large-sample normality of the batch-means variance estimator, *Operations Research Letters*, 30, 319-326, 2002.
Whitt W., The efficiency of one long run versus independent replications in steady-state simulation, *Management Science*, 37(6), 645-666, 1991.

Determination of the Warm-up Period

To estimate the long-term performance measure of the system, there are several methods such as Batch Means, Independent Replications and Regenerative Method.

Batch Means is a method of estimating the steady-state characteristic from a single-run simulation. The single run is partitioned into equal size batches large enough for estimates obtained from different batches to be approximately independent. In the method of Batch Means, it is important to ensure that the bias due to initial conditions is removed to achieve at least a covariance stationary waiting time process. An obvious remedy is to run the simulation for a period large enough to remove the effect of the initial bias. During this warm-up period, no attempt is made to record the output of the simulation. The results are thrown away. At the end of this warm-up period, the waiting time of customers are collected for analysis. The practical question is "How long should the warm-up period be?". Abate and Whitt provided a relatively simple and nice expression for the time required (t_p) for an M/M/1/ queue system (with traffic intensity ρ) starting at the origin (empty) to reach and remain within 100p% of the steady- state limit as follows:

$$t_p(\rho\ ) = 2C(\rho\ )\ \text{Ln}\ \{1/[(1-p)(1+2C(\rho\ ))]\}/(1-\rho\ )^2$$

where

$$C(\rho\ )=[2+\rho\ +(\rho\ ^2 + 4\rho\ )^{½}] / 4.$$

Some notions of $t_p(\rho\ )$ as a function of r and p, are given in following table:

Traffic Intensity	100p			
ρ	**95.0**	**99.0**	**99.9**	**99.99**
0.10	3.61	6.33	10.23	14.12
0.20	5.01	8.93	14.53	20.14
0.30	7.00	12.64	20.71	28.79
0.40	10.06	18.39	30.31	42.23
0.50	15.18	28.05	46.47	64.89
0.60	24.70	46.13	76.79	107.45
0.70	45.51	85.87	143.61	201.36
0.80	105.78	201.53	338.52	475.51
0.90	435.74	838.10	1413.70	1989.40

Time (t_p) required for an M/M/1 queue to reach and remain with 100p% limits of the steady-state value.

Although this result is developed for M/M/1 queues, it has already been established that it can serve as an approximation for more general; i.e., GI/G/1 queues.

Further Reading:

Abate J., and W. Whitt, Transient behavior of regular Brownian motion, *Advance Applied Probability*, 19, 560-631, 1987.
Chen E., and W. Kelton, Determining simulation run length with the runs test, *Simulation Modelling Practice and Theory*, 11, 237-250, 2003.

Determination of the Desirable Number of Simulation Runs

The two widely used methods for experimentation on simulation models are method of bath means, and independent replications. Intuitively one may say the method of independent replication is superior in producing statistically a "good" estimate for the system's performance measure. In fact, not one method is superior in all cases and it all depends on the traffic intensity ρ .

After deciding what method is more suitable to apply, the main question is determination of number of runs. That is, at the planning stage of a simulation investigation of the question of number of simulation runs (n) is critical.

The confidence level of simulation output drawn from a set of simulation runs depends on the size of data set. The larger the number of runs, the higher is the associated confidence. However, more simulation runs also require more effort and resources for large systems. Thus, the main goal must be in finding the smallest number of simulation runs that will provide the desirable confidence.

Pilot Studies: When the needed statistics for number of simulation runs calculation is not available from existing database, a pilot simulation is needed.

For large pilot simulation runs (n), say over 30, the simplest number of runs determinate is:

$$[(Z_{\alpha / 2})^2 S^2] / \delta^2$$

where δ is the desirable margin of error (i.e., the absolute error), which is the half-length of the confidence interval with 100(1- α)% confidence interval. S^2 is the variance obtained from the pilot run.

One may use the following sample size determinate for a desirable relative error Δ in %, which requires an estimate of the coefficient of variation (C.V. in %) from a pilot run with n over 30:

$$[(Z_{\alpha / 2})^2 (C.V.)^2] / \Delta^2$$

These sample size determinates could also be used for simulation output estimation of unimodal output populations, with discrete or continuous random variables provided the pilot run size (n) is larger than (say) 30.

The aim of applying any one of the above number of runs determinates is at improving your pilot estimates at feasible costs.

You may like using the following Applet for determination of number of runs.

Further Reading:

Díaz-Emparanza I, Is a small Monte Carlo analysis a good analysis? Checking the size power and consistency of a simulation-based test, *Statistical Papers*, 43(4), 567-577, 2002.
Whitt W., The efficiency of one long run versus independent replications in steady-state simulation, *Management Science*, 37(6), 645-666, 1991.

Determination of Simulation Runs' Size

At the planning stage of a simulation modeling the question of number of simulation runs (n) is critical. The following Java applets compute the needed Runs Size based on current avialable information ontained from a pilot simulation run, to achieve an acceptable accuracy and/or risk.

Enter the needed information, and then click the **Calculate** button.

The aim of applying any one of the following number of simulation runs determinates is at improving your pilot estimates at a feasible cost.

Notes: The normality condition might be relaxed for number of simulation runs over, say 30. Moreover, determination of number of simulation runs for mean could also be used for other unimodal simulation output distributions including those with discrete random variables, such as proportion, provided the pilot run is sufficiently large (say, over 30).

Runs' Size with Acceptable Absolute Precision

Pilot Runs' Size (n):	35
Current Estimate:	5
Current Variance Estimate:	2
Acceptable Significant Level (α):	.05
Acceptable Absolute Error:	.5
Calculate Runs' Size	
The Required Runs' Size Is:	

Runs' Size with Acceptable Relative Precision

Pilot Runs' Size (n):	50
Current Estimate:	1.65

Current Variance Estimate:	.51
Acceptable Significant Level (α):	.05
Acceptable Relative Error:	.20
Calculate Runs' Size	
The Required Runs' Size Is:	

Runs' Size Based on the Null and an Alternative

Pilot Runs' Size (n):	35
Current Estimate:	5
Current Variance Estimate:	2.93
The Value in H_0:	14
The Value in H_a:	16
Calculate Runs' Size	
The Required Runs' Size Is:	

Simulation Software Selection

The vast amount of simulation software available can be overwhelming for the new users. The following are only a random sample of software in the market today:

ACSL, APROS, ARTIFEX, Arena, AutoMod, C++SIM, CSIM, Call$im, FluidFlow, GPSS, Gepasi, JavSim, MJX, MedModel, Mesquite, Multiverse, NETWORK, OPNET Modeler, POSES++, Simulat8, Powersim, QUEST, REAL, SHIFT, SIMPLE++, SIMSCRIPT, SLAM, SMPL, SimBank, SimPlusPlus, TIERRA, Witness, SIMNON, VISSIM, and javasim.

There are several things that make an ideal simulation package. Some are properties of the package, such as support, reactivity to bug notification, interface, etc. Some are properties of the user, such as their needs, their level of expertise, etc. For these reasons asking which package is best is a sudden failure of judgment. The first question to ask is for what purpose you need the software? Is it for education, teaching, student-projects or research?

The main question is: What are the important aspects to look for in a package? The answer depends on specific applications. However some general criteria are: Input facilities, Processing that allows some programming, Optimization capability, Output facilities, Environment including training and support services, Input-output statistical data analysis capability, and certainly the Cost factor.

You must know which features are appropriate for your situation, although, this is not based on a "Yes" or "No" judgment.

For description of available simulation software, visit Simulation Software Survey.

Reference & Further Reading:

Nikoukaran J., Software selection for simulation in manufacturing: A review, *Simulation Practice and Theory*, 7(1), 1-14, 1999.

Animation in Systems Simulation

Animation in systems simulation is a useful tool. Most graphically based software packages have default animation. This is quite useful for model debugging, validation, and verification. This type of animation comes with little or no additional effort and gives the modeler additional insight into how the model. This type of animation comes with little or no additional effort and gives the modeler additional insight into how the model works. However, it augments the modeling tools available. The more realistic animation presents qualities which intend to be useful to the decision-maker in implementing the developed simulation model. There are also, good model management tools. Some tools have been developed which combined a database with simulation to store models, data, results, and animations. However, there is not one product that provides all of those capabilities.

SIMSCRIPT II.5

Without computer one cannot perform any realistic dynamic systems simulation.

SIMSCRIPT II.5 is a powerful, free-format, English-like simulation language designed to greatly simplify writing programs for simulation modelling. Programs written in SIMSCRIPT II.5 are easily read and maintained. They are accurate, efficient, and generate results which are acceptable to users. Unlike other simulation programming languages, SIMSCRIPT II.5 requires no coding in other languages. SIMSCRIPT II.5 has been fully supported for over 33 years. Contributing to the wide acceptance and success of SIMSCRIPT II.5 modelling are:

DESIGN:

A powerful worldview, consisting of Entities and Processes, provides a natural conceptual framework with which to relate real objects to the model.

PROGRAMMING:

SIMSCRIPT II.5 is a modern, free-form language with structured programming constructs and all the built-in facilities needed for model development. Model components can be programmed so they clearly reflect the organization and logic of the modeled system. The amount of program needed to model a system is typically 75% less than its FORTRAN or C counterpart.

DEBUGGER:

A well designed package of program debug facilities is provided. The required tools are available to detect errors in a complex computer program without resorting an error. Simulation status information is provided, and control is optionally transferred to a user program for additional analysis and output.

EVOLUTION:

This structure allows the model to evolve easily and naturally from simple to detailed formulation as data becomes available. Many modifications, such as the choice of set disciplines and statistics are simply specified in the Preamble.

DOCUMENTATION:

You get a powerful, English-like language supporting a modular implementation. Because each model component is readable and self-contained, the model documentation is the model listing; it is never obsolete or inaccurate.

For more information contact SIMSCRIPT

Guidelines for Running SIMSCRIPT on the VAX System

Network Access/Utilities
Connect UBE
Username:
Password:

Get $ sign

Step 1. Create and edit your source program. Type in

$EDT PROG.SIM

Step 2. Attach SIMSCRIPT. Type in

$SIMSCRIPT

Step 3. Compile the source program file. Type in

$SIMSCOMP PROG.SIM

Check for compilation errors. To locate the errors, type in

$EDT PROG.LIST

Step 4. Link the object file. Type in

$SIMLINK PROG

now you have a file containing your program in an executable format.

Step 5. To execute the program type in

$RUN PROG

Step 6. To get your hard copy print, type

$PRINT/NAME=your own name PROG.OUT, PROG.LIS

Alternatively, use submit command to place the command procedure in the *batch* job que.
Create a command file say PROG.COM containing:

```
$SIMSCRIPT
$SIMSCOMP PROG.SIM
$SIMLINK PROG
$RUN PROG
```

Then, submit

$Submit PROG.COM

An Example:

```
'' Solving an analytic equation arising from optimization
'' of a coherent reliability system with 3 homogeneous components

Preamble
        Define V as a real 1-dimensional arrays
        Define I, N as integer variables
End

Main
     Open 3 for output, Name = "PROG.OUT"
     Use 3 for ouTPUT
```

```
          LET N=50
          Reserve V(*) as N
          LET V(1)=1.
          For I = 1 to N-1
          DO
          LET U=V(I)
           LET PPRAM=-9./((1+U)**2) + 9./((2.+U)**2) + 1./U**2
              LET V(I+1)=V(I)+PPRAM/I
          LOOP
       PRINT 1 LINE WITH V(N) AND PPRAM THUS
       OPTIMAL RATE IS  ****.*****,  DERIVATIVE  IS ***.*****
    END
```

The output

```
        OPTIMAL RATE IS     0.76350,  DERIVATIVE  IS  -0.00000
```

System Dynamics and Discrete Event Simulation

The modeling techniques used by system dynamics and discrete event simulations are often different at two levels: The modeler way of representing systems might be different, the underlying simulators' algorithms are also different. Each technique is well tuned to the purpose it is intended. However, one may use a discrete event approach to do system dynamics and vice versa.

Traditionally, the most important distinction is the purpose of the modeling. The discrete event approach is to find, e.g., how many resources the decision maker needs such as how many trucks, and how to arrange the resources to avoid bottlenecks, i.e., excessive of waiting lines, waiting times, or inventories. While the system dynamics approach is to prescribe for the decision making to, e.g., timely respond to any changes, and how to change the physical structure, e.g., physical shipping delay time, so that inventories, sales, production, etc.

System dynamics is the rigorous study of problems in system behavior using the principles of feedback, dynamics and simulation. In more words system dynamics is characterized by:

- Searching for useful solutions to real problems, especially in social systems (businesses, schools, governments,...) and the environment.
- Using computer simulation models to understand and improve such systems.
- Basing the simulation models on mental models, qualitative knowledge and numerical information.
- Using methods and insights from feedback control engineering and other scientific disciplines to assess and improve the quality of models.
- Seeking improved ways to translate scientific results into achieved implemented improvement.
- Systems dynamics approach looks at systems at a very high level so is more suited to strategic analysis. Discrete event approach may look at subsystems for a detailed analysis and is more suited, e.g., to process re-engineering problems.
- Systems dynamics is indicative, i.e., helps us understand the direction and magnitude of effects (i.e., where in the system do we need to make the changes), whereas discrete event approach is predictive (i.e., how many resources do we need to achieve a certain goal of throughout).
- Systems dynamics analysis is continuous in time and it uses mostly deterministic analysis, whereas discrete event process deals with analysis in a specific time horizon and uses stochastic analysis.

Some interesting and useful areas of system dynamics modeling approach are:

- Short-term and long term forecasting of agricultural produce with special reference to field crops and perennial fruits such as grapes, which have significant processing sectors of different proportions of total output where both demand and supply side perspectives are being considered.

- Long term relationship between the financial statements of balance sheet, income statement and cash flow statement balanced against scenarios of the stock market's need to seek a stable/growing share price combined with a satisfactory dividend and related return on shareholder funds policy.

- Managerial applications include the development and evaluation of short-term and long-term strategic plans, budget analysis and assessment, business audits and benchmarking.

A modeler must consider both as complementary tools to each other. Systems dynamic to look at the high level problem and identify areas which need more detailed analysis. Then, use discrete event modeling tools to analyze (and predict) the specific areas of interest.

What Is Social Simulation?

Social scientists have always constructed models of social phenomena. Simulation is an important method for modeling social and economic processes. In particular, it provides a "middle way" between the richness of discursive theorizing and rigorous but restrictive mathematical models. There are different types of computer simulation and their application to social scientific problems.

Faster hardware and improved software have made building complex simulations easier. Computer simulation methods can be effective for the development of theories as well as for prediction. For example, macro-economic models have been used to simulate future changes in the economy; and simulations have been used in psychology to study cognitive mechanisms.

The field of 'social simulation' seems to be following an interesting line of inquiry. As a general approach in the field, a 'world' is specified with much computational detail. Then the 'world' is simulated (using computers) to reveal some of the 'non-trivial' implications (or 'emergent properties') of the 'world'. When these 'non trivial' implications are made known (fed back) in world, apparently it constitutes some 'added values'.

Artificial Life is an interdisciplinary study enterprise aimed at understanding life-as-it-is and life-as-it-could-be, and at synthesizing life-like phenomena in chemical, electronic, software, and other artificial media. Artificial Life redefines the concepts of artificial and natural, blurring the borders between traditional disciplines and providing new media and new insights into the origin and principles of life.

Simulation allows the social scientist to experiment with ‘artificial societies' and explore the implications of theories in ways not otherwise possible.

Reference and Further Readings:

Gilbert N., and K. Troitzsch, *Simulation for the Social Scientist*, Open University Press, Buckingham, UK, 1999.
Sichman J., R. Conte, and N. Gilbert, (eds,), *Multi-Agent Systems and Agent-Based Simulation*, Berlin, Springer-Verlag, 1998.

What Is Web-based Simulation?

Web-based simulation is quickly emerging as an area of significant interest for both simulation researchers and simulation practitioners. This interest in web-based simulation is a natural outgrowth of the proliferation of the World-Wide Web and its attendant technologies, e.g. HTML, HTTP, CGI, etc. Also the surging popularity of, and reliance upon, computer simulation as a problem solving and decision support systems tools.

The appearance of the network-friendly programming language, Java, and of distributed object technologies like the Common Object Request Broker Architecture (CORBA) and the Object Linking and Embedding / Component Object Model (OLE/COM) have had particularly acute effects on the state of simulation practice.

Currently, the researchers in the field of web-based simulation are interested in dealing with topics such as methodologies for web-based model development, collaborative model development over the Internet, Java-based modeling and simulation, distributed modeling and simulation using web technologies, and new applications.

Parallel and Distributed Simulation

The increasing size of the systems and designs requires more efficient simulation strategies to accelerate the simulation process. Parallel and distributed simulation approaches seem to be a promising approach in this direction. Current topics under extensive research are:

Synchronization, scheduling, memory management, randomized and reactive/adaptive algorithms, partitioning and load balancing.

Synchronization in multi-user distributed simulation, virtual reality environments, HLA, and interoperability.

System modeling for parallel simulation, specification, re-use of models/code, and parallelizing existing simulations.

Language and implementation issues, models of parallel simulation, execution environments, and libraries.

Theoretical and empirical studies, prediction and analysis, cost models, benchmarks, and comparative studies.

Computer architectures, VLSI, telecommunication networks, manufacturing, dynamic systems, and biological/social systems.

Web based distributed simulation such as multimedia and real time applications, fault tolerance, implementation issues, use of Java, and CORBA.

References & Further Readings:

Bossel H., *Modeling & Simulation*, A. K. Peters Pub., 1994.
Delaney W., and E. Vaccari, *Dynamic Models and Discrete Event Simulation*, Dekker, 1989.
Fishman G., *Discrete-Event Simulation: Modeling, Programming and Analysis*, Springer-Verlag, Berlin, 2001.
Fishwick P., *Simulation Model Design and Execution: Building Digital Worlds*, Prentice-Hall, Englewood Cliffs, 1995.
Ghosh S., and T. Lee, *Modeling & Asynchronous Distributed Simulation: Analyzing Complex Systems*, IEEE Publications, 2000.
Gimblett R., *Integrating Geographic Information Systems and Agent-Based Modeling: Techniques for Simulating Social and Ecological Processes*, Oxford University Press, 2002.
Harrington J., and K. Tumay, *Simulation Modeling Methods: An Interactive Guide to Results-Based Decision*, McGraw-Hill, 1998.
Haas P., *Stochastic Petri Net Models Modeling and Simulation*, Springer Verlag, 2002.
Hill D., *Object-Oriented Analysis and Simulation Modeling*, Addison-Wesley, 1996.
Kouikoglou V., and Y. Phillis, *Hybrid Simulation Models of Production Networks*, Kluwer Pub., 2001.
Law A., and W. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, 2000.
Nelson B., *Stochastic Modeling: Analysis & Simulation*, McGraw-Hill, 1995.
Oakshott L., *Business Modelling and Simulation*, Pitman Publishing, London, 1997.
Pidd M., *Computer Simulation in Management Science*, Wiley, 1998.
Rubinstein R., and B. Melamed, *Modern Simulation and Modeling*, Wiley, 1998.
Severance F., *System Modeling and Simulation: An Introduction*, Wiley, 2001.
Van den Bosch, P. and A. Van der Klauw, Modeling, *Identification & Simulation of Dynamical Systems*, CRC Press, 1994.
Woods R., and K. Lawrence, *Modeling and Simulation of Dynamic Systems*, Prentice Hall, 1997.

Techniques for Sensitivity Estimation

Simulation continues to be the primary method by which engineers and managers obtain information about complex stochastic systems, such as telecommunication networks, health service, corporate planning, financial modeling, production assembly lines, and flexible manufacturing systems. These

systems are driven by the occurrence of discrete events; and complex interactions within these discrete events occur over time. For most discrete event systems (DES) no analytical methods are available, so DES must be studied via simulation. DES are studied to understand their performance, and to determine the best ways to improve their performance. In particular, one is often interested in how system performance depends on the system's parameter v, which could be a vector.

DES's system performance is often measured as an expected value. Consider a system with continuous parameter v ⌊ V ⌋ R^n, where V is an open set. Let

$$J(v) = E_{Y \mid v}[Z(Y)]$$

be the steady state expected performance measure, where Y is a random vector with known probability density function (pdf), f(y; v) depends on v, and Z is the performance measure.

In discrete event systems, Monte Carlo simulation is usually needed to estimate J(v) for a given value $v = v_0$. By the law of large numbers

$$J(v_0) = 1/n \Sigma \quad Z(y_i)$$

converges to the true value, where y_i, i = 1, 2,..., n are independent, identically distributed, random vector realizations of Y from f (y; v_0), and n is the number of independent replications.

We are interested in sensitivities estimation of J(v) with respect to v.

Applications of sensitivity information

There are a number of areas where sensitivity information (the gradient, Hessian, etc.) of a performance measure J(v) or some estimate of it, is used for the purpose of analysis and control. In what follows, we single out a few such areas and briefly discuss them.

Local information: An estimate for dJ/dv is a good local measure of the effect of on performance. For example, simply knowing the sign of the derivative dJ/dv at some point v immediately gives us the direction in which v should be changed. The magnitude of dJ/d? also provides useful information in an initial design process: If dJ/dv is small, we conclude that J is not very sensitive to changes in , and hence focusing concentration on other parameters may improve performance.

Structural properties: Often sensitivity analysis provides not only a numerical value for the sample derivative, but also an expression which captures the nature of the dependence of a performance measure on the parameter v . The simplest case arises when dJ/dv can be seen to be always positive (or always negative) for any sample path; we may not be able to tell if the value of J(v) is monotonically increasing (or decreasing) in v . This information in itself is very useful in design and analysis. More generally, the form of dJ/dv can reveal interesting structural properties of the DES (e.g., monotonicity, convexity). Such properties must be exploited in order to determine optimal operating policies for some systems.

Response surface generation: Often our ultimate goal is to obtain the function J(v), i.e., a curve describing how the system responds to different values of v . Since J(v) is unknown, one alternative is to obtain estimates of J(v) for as many values of v as possible. This is clearly a prohibitively difficult task. Derivative information, however may include not only first-order but also higher derivatives which can be used to approximate J(v). If such derivative information can be easily and accurately obtained, the task of response surface generation may be accomplished as well.

Goal-seeking and What-if problems: Stochastic models typically depend upon various uncertain parameters that must be estimated from existing data sets. Statistical questions of how input parameter uncertainty propagates through the model into output parameter uncertainty is the so-called "what-if" analysis. A good answer to this question often requires sensitivity estimates. The ordinary simulation output results are the solution of a direct problem: Given the underlying pdf with a

particular parameter value v , we may estimate the output function J(v). Now we pose the goal-seeking problem: given a target output value J_0 of the system and a parameterized pdf family, find an input value for the parameter, which generates such an output. There are strong motivations for both problems. When v is any controllable or uncontrollable parameter [the decision maker is, for example, interested in estimating J(v) for a small change in v], the so called "what-if" problem, which is a "direct problem" and can be solved by incorporating sensitivity information in the Taylor's expansion of J(v) in the neighborhood of v . However, when v is a controllable input, the decision maker may be interested in the goal-seeking problem: what change in the input parameter will achieve a desired change in output value J(v). Another application of goal-seeking arises when we want to adapt a model to satisfy a new equality constraint (condition) for some stochastic function. The solution to the goal-seeking problem is to estimate the derivative of the output function with respect to the input parameter for the nominal system; use this estimate in a Taylor's expansion of the output function in the neighborhood of the parameter; and finally, use Robbins-Monro (R-M) type of stochastic approximation algorithm to estimate the necessary controllable input parameter value within the desired accuracy.

Optimization: Discrete-event simulation is the primary analysis tool for designing complex systems. However, simulation must be linked with a mathematical optimization technique to be effectively used for systems design. The sensitivity dJ/dv can be used in conjunction with various optimization algorithms whose function is to gradually adjust v until a point is reached where J(v) is maximized (or minimized). If no other constraints on v are imposed, we expect dJ/dv = 0 at this point .

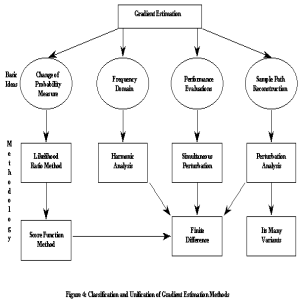


Figure 4: Classification and Unification of Gradient Estimation Methods

Click on the image **to enlarge** it and THEN print it.

Finite difference approximation

Kiefer and Wolfowitz proposed a finite difference approximation to the derivative. One version of the Kiefer-Wolfwitz technique uses two-sided finite differences. The first fact to notice about the K-W estimate is that it requires 2N simulation runs, where N is the dimension of vector parameter θ . If the decision maker is interested in gradient estimation with respect to each of the components of θ , then 2N simulations must be run for each component of v . This is inefficient. The second fact is that it may have a very poor variance, and it may result in numerical calculation difficulties.

Simultaneous perturbation methods

The simultaneous perturbation (SP) algorithm introduced by Dr. J. Spall has attracted considerable attention. There has recently been much interest in recursive optimization algorithms that rely on measurements of only the objective function to be optimized, not requiring direct measurements of the gradient of the objective function. Such algorithms have the advantage of not requiring detailed modeling information describing the relationship between the parameters to be optimized and the objective function. For example, many systems involving complex simulations or human beings are difficult to model, and could potentially benefit from such an optimization approach. The simultaneous perturbation stochastic approximation (SPSA) algorithm operates in the same framework as the above K-W methods, but has the strong advantage of requiring a much lower number of simulation runs to obtain the same quality of result. The essential feature of SPSA, which accounts for its power and relative ease of use in difficult multivariate optimization problems--is the underlying gradient approximation that requires only TWO objective function measurements regardless of the dimension of the optimization problem (one variation of basic SPSA uses only ONE

objective function measurement per iteration). The underlying theory for SPSA shows that the N-fold savings in simulation runs per iteration (per gradient approximation) translates directly into an N-fold savings in the number of simulations to achieve a given quality of solution to the optimization problem. In other words, the K-W method and SPSA method take the same number of iterations to converge to the answer despite the N-fold savings in objective function measurements (e.g., simulation runs) per iteration in SPSA.

Perturbation analysis

Perturbation analysis (PA) computes (roughly) what simulations would have produced, had v been changed by a "small" amount without actually making this change. The intuitive idea behind PA is that a sample path constructed using v is frequently structurally very similar to the sample path using the perturbed v . There is a large amount of information that is the same for both of them. It is wasteful to throw this information away and to start the simulation from scratch with the perturbed v . In PA, moreover, we can let the change approach zero to get a derivative estimator without numerical problems. We are interested in the affect of a parameter change on the performance measure. However, we would like to realize this change by keeping the order of events exactly the same. The perturbations will be so small that only the duration, not the order, of the states will be affected. This effect should be observed in three successive stages:

Step 1: How does a change in the value of a parameter vary the sample duration related to that parameter?

Step 2: How does the change in an individual sample duration reflect itself as a change in a subsequent particular sample realization?

Step 3: Finally, what is the relationship between the variation of the sample realization and its expected value?

Score function methods

Using the score function method, the gradient can be estimated simultaneously, at any number of different parameter values, in a single-run simulation. The basic idea is that, the gradient of the performance measure function, J'(v), is expressed as an expectation with respect to the same distribution as the performance measure function itself. Therefore, the sensitivity information can be obtained with little computational (not simulation) cost, while estimating the performance measure. It is well-known that the crude form of the SF estimator suffers from the problem of linear growth in its variance as the simulation run increases. However, in the steady-state simulation the variance can be controlled by run length. Furthermore, information about the variance may be incorporated into the simulation algorithm. A recent flurry of activity has attempted to improve the accuracy of the SF estimates. Under regenerative conditions, the estimator can easily be modified to alleviate this problem, yet the magnitude of the variance may be large for queueing systems with heavy traffic intensity. The heuristic idea is to treat each component of the system (e.g. each queue) separately, which synchronously assumes that individual components have "local" regenerative cycles. This approach is promising since the estimator remains unbiased and efficient while the global regenerative cycle is very long.

Now we look at the general (non-regenerative) case. In this case any simulation will give a biased estimator of the gradient, as simulations are necessarily finite. If n (the length of the simulation) is large enough, this bias is negligible. However, as noted earlier, the variance of the SF sensitivity estimator increases with increase in n so, a crude SF estimator is not even approximately consistent. There are a number of ways to attack this problem. Most of the variations in an estimator comes from the score function. The variation is especially high, when all past inputs contribute to the performance and the scores from all are included. When one uses batch means, the variation is reduced by keeping the length of the batch small.

A second way is to reduce the variance of the score to such an extent that we can use simulations long enough to effectively eliminate the bias. This is the most promising approach. The variance may be

reduced further by using the standard variance reduction techniques (VRT), such as importance sampling. Finally, we can simply use a large number of iid replications of the simulation.

Harmonic analysis

Another strategy for estimating the gradient simulation is based on the frequency domain method, which differs from the time domain experiments in that the input parameters are deterministically varied in sinusoidal patterns during the simulation run, as opposed to being kept fixed as in the time domain runs. The range of possible values for each input factor should be identified. Then the values of each input factor within its defined range should be changed during a run. In time series analysis, t is the time index. In simulation, however, t is not necessarily the simulation clock time. Rather, t is a variable of the model, which keeps track of certain statistics during each run. For example, to generate the inter-arrival times in a queueing simulation, t might be the variable that counts customer arrivals.

Frequency domain simulation experiments identify the significant terms of the polynomial that approximates the relationship between the simulation output and the inputs. Clearly, the number of simulation runs required to identify the important terms by this approach is much smaller than those of the competing alternatives, and the difference becomes even more conspicuous as the number of parameters increases.

Conclusions & Further Readings

PA and SF (or LR) can be unified. Further comparison of the PA and SF approaches reveals several interesting differences. Both approaches require an interchange of expectation and differentiation. However, the conditions for this interchange in PA depend heavily on the nature of the problem, and must be verified for each application, which is not the case in SF. Therefore, in general, it is easier to satisfy SF unbiased conditions. PA assumes that the order of events in the perturbed path is the same as the order in the nominal path, for a small enough change in v , allowing the computation of the sensitivity of the sample performance for a particular simulation. For example, if the performance measure is the mean number of customer in a busy period, the PA estimate of the gradient with respect to any parameter is zero! The number of customers per busy period will not change if the order of events does not change.

In terms of ease of implementation, PA estimators may require considerable analytical work on the part of algorithm developer, with some "customization" for each application, whereas SF has the advantage of remaining a general definable algorithm whenever it can be applied.

Perhaps the most important criterion for comparison lies in the question of accuracy of an estimator, typically measured through its variance. If an estimator is strongly consistent, its variance is gradually reduced over time and ultimately approaches to zero. The speed with which this happens may be extremely important. Since in practice, decisions normally have to be made in a limited time, an estimator whose variance decreases fast is highly desirable. In general, when PA does provide unbiased estimators, the variance of these estimators is small. PA fully exploits the structure of DES and their state dynamics by extracting the needed information from the observed sample path, whereas SF requires no knowledge of the system other than the inputs and the outputs. Therefore when using SF methods, variance reduction is necessary. The question is whether or not the variance can be reduced enough to make the SF estimator useful in all situations to which it can be applied. The answer is certainly yes. Using the standard variance reduction techniques can help, but the most dramatic variance reduction occurs using new methods of VR such as conditioning, which is shown numerically to have a mean squared error that is essentially the same as that of PA.

For more, visit the Web site: Modeling & Simulation Resources.

References & Further Readings:

Arsham H., Algorithms for Sensitivity Information in Discrete-Event Systems Simulation, *Simulation Practice and Theory*, 6(1), 1-22, 1998.
Fu M., and J-Q. Hu, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, Kluwer Academic Publishers, 1997.
Rubinstein R., and A. Shapiro, *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, John Wiley & Sons, 1993.
Whitt W., Minimizing delays in the GI/G/1 queue, *Operations Research*, 32(1), 41-51, 1984.

Simulation-based Optimization Techniques

Discrete event simulation is the primary analysis tool for designing complex systems. Simulation, however, must be linked with a optimization techniques to be effectively used for systems design. We present several optimization techniques involving both continuous and discrete controllable input parameters subject to a variety of constraints. The aim is to determine the techniques most promising for a given simulation model.

Many man-made systems can be modeled as Discrete Event Systems (DES); examples are computer systems, communication networks, flexible manufacturing systems, production assembly lines, and traffic transportation systems. DES evolve with the occurrence of discrete events, such as the arrival of a job or the completion of a task, in contrast with continuously variable dynamic processes such as aerospace vehicles, which are primarily governed by differential equations. Owing to the complex dynamics resulting from stochastic interactions of such discrete events over time, the performance analysis and optimization of DES can be difficult tasks. At the same time, since such systems are becoming more widespread as a result of modern technological advances, it is important to have tools for analyzing and optimizing the parameters of these systems.

Analyzing complex DES often requires computer simulation. In these systems, the objective function may not be expressible as an explicit function of the input parameters; rather, it involves some performance measures of the system whose values can be found only by running the simulation model or by observing the actual system. On the other hand, due to the increasingly large size and inherent complexity of most man-made systems, purely analytical means are often insufficient for optimization. In these cases, one must resort to simulation, with its chief advantage being its generality, and its primary disadvantage being its cost in terms of time and money. Even though, in principle, some systems are analytically tractable, the analytical effort required to evaluate the solution may be so formidable that computer simulation becomes attractive. While the price for computing resources continue to dramatically decrease, one nevertheless can still obtain only a statistical estimate as opposed to an exact solution. For practical purposes, this is quite sufficient.

These man-made DES are costly, and therefore it is important to operate them as efficiently as possible. The high cost makes it necessary to find more efficient means of conducting simulation and optimizing its output. We consider optimizing an objective function with respect to a set of continuous and/or discrete controllable parameters subject to some constraints.

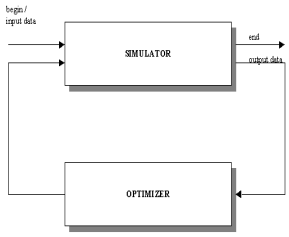


Fig. 1: An Optimizer Support System

Click on the image **to enlarge** it and THEN print it.

The above figure illustrates the feedback loop application. Although the feedback concept is not a simulation but a systemic concept, however, whatever paradigm we use one can always incorporate feedback. For example, consider a discrete event system (DES) model that employs resources to achieve certain tasks/processes, by only incorporating decision rules regarding how to manage the stocks and thence how the resource will be deployed depending on the stock level, clearly, in the system structure there are feedback loops.

Usually when modelers choose a DES approach they often model the system as open loop or nearly open loop system, making the system behave as if there where no superior agent controlling the whole production/service/ process. Closing the loops should be an elemental task that simulation modeler should take care of, even if the scope does not involve doing it, there must be awareness of

system behavior, particularly if there is known to be that the system if under human decision making processes/activities.

In almost all simulation models, an expected value can express the system's performance. Consider a system with continuous parameter v ⌊ V, where V is the feasible region. Let

$$J(v) = E_{Y \mid v}[Z(Y)]$$

be the steady state expected performance measure, where Y is a random vector with known probability density function (pdf), f(y; v) depends on v, and Z is the performance measure.

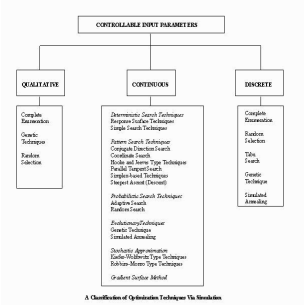
In discrete event systems, Monte Carlo simulation is usually needed to estimate J(v) for a given value $v = v_0$. By the law of large numbers

$$J(v_0) = 1/n \Sigma \quad Z(y_i)$$

converges to the true value, where y_i, i = 1, 2,..., n are independent, identically distributed, random vector realizations of Y from f (y; v_0), and n is the number of independent replications.

The aim is to optimize J(v) with respect to v .

We shall group the optimization techniques for simulation into seven broad categories; namely, Deterministic Search, Pattern Search, Probabilistic Search, Evolutionary Techniques, Stochastic Approximation, Gradient Surface, and some Mixtures of the these techniques;



A Classification of Optimization Techniques Via Simulation

Click on the image **to enlarge** it and THEN print it.

Deterministic search techniques

A common characteristic of deterministic search techniques is that they are basically borrowed from deterministic optimization techniques. The deterministic objective function value required in the technique is now replaced with an estimate obtained from simulation. By having a reasonably accurate estimate, one hopes that the technique will perform well.

Deterministic search techniques include heuristic search, complete enumeration, and random search techniques.

Heuristic search technique

The heuristic search technique is probably most commonly used in optimizing response surfaces. It is also the least sophisticated scheme mathematically, and it can be thought of as an intuitive and experimental approach. The analyst determines the starting point and stopping rule based on previous experience with the system. After setting the input parameters (factors) to levels that appear reasonable, the analyst makes a simulation run with the factors set at those levels and computes the value of the response function. If it appears to be a maximum (minimum) to the analyst, the experiment is stopped. Otherwise the analyst changes parameter settings and makes another run. This process continues until the analyst believes that the output has been optimized. Suffice it to say that, if the analyst is not intimately familiar with the process being simulated, this procedure can turn into

a blind search and can expend an inordinate amount of time and computer resources without producing results commensurate with input. The heuristic search can be ineffective and inefficient in the hand of a novice.

Complete enumeration and random techniques

The complete enumeration technique is not applicable to continuous cases, but in discrete space v it does yield the optimal value of the response variable. All factors (v) must assume a finite number of values for this technique to be applicable. Then, a complete factorial experiment is run. The analyst can attribute some degree of confidence to the determined optimal point when using this procedure. Although the complete enumeration technique yields the optimal point, it has a serious drawback. If the number of factors or levels per factor is large, the number of simulation runs required to find the optimal point can be exceedingly large. For example, suppose that an experiment is conducted with three factors having three, four, and five levels, respectively. Also suppose that five replications are desired to provide the proper degree of confidence. Then 300 runs of the simulator are required to find the optimal point. Hence, this technique should be used only when the number of unique treatment combinations is relatively small or a run takes little time.

The random search technique resembles the complete enumeration technique except that one selects a set of inputs at random. The simulated results based on the set that yields the maximum (minimum) value of the response function is taken to be the optimal point. This procedure reduces the number of simulation runs required to yield an 'optimal' result; however, there is no guarantee that the point found is actually the optimal point. Of course, the more points selected, the more likely the analyst is to achieve the true optimum. Note that the requirement that each factor assumes only a finite number of values is not a requirement in this scheme. Replications can be made on the treatment combinations selected, to increase the confidence in the optimal point. Which strategy is better, replicating a few points or looking at a single observation on more points, depends on the problem.

Response surface search

Response surface search attempts to fit a polynomial to J(v). If the design space v is suitably small, the performance function J(v) may be approximated by a response surface, typically a first order, or perhaps quadratic order in v , possibly after transformation, e.g., log (v). The response surface method (RSM) requires running the simulation in a first order experimental design to determine the path of steepest descent. Simulation runs made along this path continue, until one notes no improvement in J(v). The analyst then runs a new first order experimental design around the new 'optimal' point reached, and finds a new path of steepest descent. The process continues, until there is a lack of fit in the fitted first order surface. Then, one runs a second order design, and takes the optimum of the fittest second order surface as the estimated optimum.

Although it is desirable for search procedures to be efficient over a wide range of response surfaces, no current procedure can effectively overcome non-unimodality (surfaces having more than one local maximum or minimum). An obvious way to find the global optimal would be to evaluate all the local optima. One technique that is used when non-unimodality is known to exist, is called the "Las Vegas" technique. This search procedure estimates the distribution of the local optima by plotting the estimated J(v) for each local search against its corresponding search number. Those local searches that produce a response greater than any previous response are then identified and a curve is fitted to the data. This curve is then used to project the "estimated incremental" response that will be achieved by one more search. The search continues until the value of the estimated improvement in the search is less than the cost of completing one additional search.

It should be noted that a well-designed experiment requires a sufficient number of replications so that the average response can be treated as a deterministic number for search comparisons. Otherwise, since replications are expensive, it becomes necessary to effectively utilize the number of simulation runs. Although each simulation is at a different setting of the controllable variables, one can use smoothing techniques such as exponential smoothing to reduce the required number of replications.

Pattern search techniques

Pattern search techniques assume that any successful set of moves used in searching for an approximated optimum is worth repeating. These techniques start with small steps; then, if these are successful, the step size increases. Alternatively, when a sequence of steps fails to improve the objective function, this indicates that shorter steps are appropriate so we may not overlook any promising direction. These techniques start by initially selecting a set of incremental values for each factor. Starting at an initial base point, they check if any incremental changes in the first variable yield an improvement. The resulting improved setting becomes the new intermediate base point. One repeats the process for each of the inputs until one obtains a new setting where the intermediate base points act as the initial base point for the first variable. The technique then moves to the new setting. This procedure is repeated, until further changes cannot be made with the given incremental values. Then, the incremental values are decreased, and the procedure is repeated from the beginning. When the incremental values reach a pre-specified tolerance, the procedure terminates; the most recent factor settings are reported as the solution.

Conjugate direction search

The conjugate direction search requires no derivative estimation, yet it finds the optimum of an N-dimensional quadratic surface after, at most, N-iterations, where the number of iterations is equal to the dimension of the quadratic surface. The procedure redefines the n dimensions so that a single variable search can be used successively. Single variable procedures can be used whenever dimensions can be treated independently. The optimization along each dimension leads to the optimization of the entire surface.

Two directions are defined to be conjugate whenever the cross-product terms are all zero. The conjugate direction technique tries to find a set of n dimensions that describes the surface such that each direction is conjugate to all others.

Using the above result, the technique attempts to find two search optima and replace the n^{th} dimension of the quadratic surface by the direction specified by the two optimal points. Successively replacing the original dimension yields a new set of n dimensions in which, if the original surface is quadratic, all directions are conjugate to each other and appropriate for n single variable searches. While this search procedure appears to be very simple, we should point out that the selection of appropriate step sizes is most critical. The step size selection is more critical for this search technique because - during axis rotation - the step size does not remain invariant in all dimensions. As the rotation takes place, the best step size changes, and becomes difficult to estimate.

Steepest ascent (descent)

The steepest ascent (descent) technique uses a fundamental result from calculus (that the gradient points in the direction of the maximum increase of a function), to determine how the initial settings of the parameters should be changed to yield an optimal value of the response variable. The direction of movement is made proportional to the estimated sensitivity of the performance of each variable.

Although quadratic functions are sometimes used, one assumes that performance is linearly related to the change in the controllable variables for small changes. Assume that a good approximation is a linear form. The basis of the linear steepest ascent is that each controllable variable is changed in proportion to the magnitude of its slope. When each controllable variable is changed by a small amount, it is analogous to determining the gradient at a point. For a surface containing N controllable variables, this requires N points around the point of interest. When the problem is not an n-dimensional elliptical surface, the parallel-tangent points are extracted from bitangents and inflection points of occluding contours. Parallel tangent points are points on the occluding contour where the tangent is parallel to a given bitangent or the tangent at an inflection point.

Tabu search technique

An effective technique to overcome local optimality for discrete optimization is the Tabu Search technique. It explores the search space by moving from a solution to its best neighbor, even if this results in a deterioration of the performance measure value. This approach increases the likelihood of moving out of local optima. To avoid cycling, solutions that were recently examined are declared tabu (Taboo) for a certain number of iterations. Applying intensification procedures can accentuate the search in a promising region of the solution space. In contrast, diversification can be used to broaden the search to a less explored region. Much remains to be discovered about the range of problems for which the tabu search is best suited.

Hooke and Jeeves type techniques

The Hooke and Jeeves pattern search uses two kinds of moves; namely, an exploratory and a pattern move. The exploratory move is accomplished by doing a coordinate search in one pass through all the variables. This gives a new "base point" from which a pattern move is made. A pattern move is a jump in the pattern direction determined by subtracting the current base point from the previous base point. After the pattern move, another exploratory move is carried out at the point reached. If the estimate of J(v) is improved at the final point after the second exploratory move, it becomes the new base point. If it fails to show improvement, an exploratory move is carried out at the last base point with a smaller step in the coordinate search. The process stops when the step gets "small" enough.

Simplex-based techniques

The simplex-based technique performs simulation runs first at the vertices of the initial simplex; i.e., a polyhedron in the v -space having N+1 vertices. A subsequent simplex (moving towards the optimum) are formed by three operations performed on the current simplex: reflection, contraction, and expansion. At each stage of the search process, the point with the highest J(v) is replaced with a new point foundvia reflection through the centroid of the simplex. Depending on the value of J(v) at this new point, the simplex is either expanded, contracted, or unchanged. The simplex technique starts with a set of N+1 factor settings. These N+1 points are all the same distance from the current point. Moreover, the distance between any two points of these N+1 points is the same. Then, by comparing their response values, the technique eliminates the factor setting with the worst functional value and replaces it with a new factor setting, determined by the centroid of the N remaining factor settings and the eliminated factor setting. The resulting simplex either grows or shrinks, depending on the response value at the new factor settings. One repeats the procedure until no more improvement can be made by eliminating a point, and the resulting final simplex is small. While this technique will generally performance well for unconstrained problems, it may collapse to a point on a boundary of a feasible region, thereby causing the search to come to a premature halt. This technique is effective if the response surface is generally bowl- shaped even with some local optimal points.

Probabilistic search techniques

All probabilistic search techniques select trial points governed by a scan distribution, which is the main source of randomness. These search techniques include random search, pure adaptive techniques, simulated annealing, and genetic methods.

Random search

A simple, but very popular approach is the random search, which centers a symmetric probability density function (pdf) [e.g., the normal distribution], about the current best location. The standard normal N(0, 1) is a popular choice, although the uniform distribution U[-1, 1] is also common.

A variation of the random search technique determines the maximum of the objective function by analyzing the distribution of J(v) in the bounded sub-region. In this variation, the random data are fitted to an asymptotic extreme-value distribution, and J* is estimated with a confidence statement. Unfortunately, these techniques cannot determine the location of J* , which can be as important as the J value itself. Some techniques calculate the mean value and the standard deviation of J(v) from the random data as they are collected. Assuming that J is distributed normally in the feasible region., the

first trial, that yields a J-value two standard deviations within the mean value, is taken as a near-optimum solution.

Pure adaptive search

Various pure adaptive search techniques have been suggested for optimization in simulation. Essentially, these techniques move from the current solution to the next solution that is sampled uniformly from the set of all better feasible solutions.

Evolutionary Techniques

Nature is a robust optimizer. By analyzing nature's optimization mechanism we may find acceptable solution techniques to intractable problems. Two concepts that have most promise are simulated annealing and the genetic techniques.

Simulated annealing

Simulated annealing (SA) borrows its basic ideas from statistical mechanics. A metal cools, and the electrons align themselves in an optimal pattern for the transfer of energy. In general, a slowly cooling system, left to itself, eventually finds the arrangement of atoms, which has the lowest energy. The is the behavior, which motivates the method of optimization by SA. In SA we construct a model of a system and slowly decrease the "temperature" of this theoretical system, until the system assumes a minimal energy structure. The problem is how to map our particular problem to such an optimizing scheme.

SA as an optimization technique was first introduced to solve problems in discrete optimization, mainly combinatorial optimization. Subsequently, this technique has been successfully applied to solve optimization problems over the space of continuous decision variables. SA is a simulation optimization technique that allows random ascent moves in order to escape the local minima, but a price is paid in terms of a large increase in the computational time required. It can be proven that the technique will find an approximated optimum. The annealing schedule might require a long time to reach a true optimum.

Genetic techniques

Genetic techniques (GT) are optimizers that use the ideas of evolution to optimize a system that is too difficult for traditional optimization techniques. Organisms are known to optimize themselves to adapt to their environment.

GT differ from traditional optimization procedures in that GT work with a coding of the decision parameter set, not the parameters themselves; GT search a population of points, not a single point; GT use objective function information, not derivatives or other auxiliary knowledge; and finally, GT use probabilistic transition rules, not deterministic rules. GT are probabilistic search optimizing techniques that do not require mathematical knowledge of the response surface of the system, which they are optimizing. They borrow the paradigms of genetic evolution, specifically selection, crossover, and mutation.

Selection: The current points in the space are ranked in terms of their fitness by their respective response values. A probability is assigned to each point that is proportional to its fitness, and parents (a mating pair) are randomly selected.

Crossover: The new point, or offspring, is chosen, based on some combination of the genetics of the two parents.

Mutation: The location of offspring is also susceptible to mutation, a process, which occurs with probability p, by which a offspring is replaced randomly by a new offspring location.

A generalized GT generates p new offspring at once and kills off all of the parents. This modification is important in the simulation environment. GT are well suited for qualitative or policy decision optimization such as selecting the best queuing disciplines or network topologies. They can be used to help determine the design of the system and its operation. For applications of GT to inventory systems, job-shop, and computer time-sharing problems. GT do not have certain shortcomings of other optimization techniques, and they will usually result in better calculated optima than those found with the traditionally techniques. They can search a response surface with many local optima and find (with a high probability) the approximate global optimum. One may use GT to find an area of potential interest, and then resort to other techniques to find the optimum. Recently, several classical GT principles have been challenged.

Differential Evolution: Differential Evolution (DE) is a genetic type of algorithm for solving continuous stochastic function optimization. The basic idea is to use vector differences for perturbing the vector population. DE adds the weighted difference between two population vectors to a third vector. This way, no separate probability distribution has to be used, which makes the scheme completely self-organizing.

A short comparison

When performing search techniques in general, and simulated annealing or genetic techniques specifically, the question of how to generate the initial solution arises. Should it be based on a heuristic rule or on a randomly generated one? Theoretically, it should not matter, but in practice this may depend on the problem. In some cases, a pure random solution systematically produces better final results. On the other hand, a good initial solution may lead to lower overall run times. This can be important, for example, in cases where each iteration takes a relatively long time; therefore, one has to use some clever termination rule. Simulation time is a crucial bottleneck in an optimization process. In many cases, a simulation is run several times with different initial solutions. Such a technique is most robust, but it requires the maximum number of replications compared with all other techniques. The pattern search technique applied to small problems with no constraints or qualitative input parameters requires fewer replications than the GT. GT, however, can easily handle constraints, and have lower computational complexity. Finally, simulated annealing can be embedded within the Tabu search to construct a probabilistic technique for global optimization.

References & Further Readings:

Choi D.-H., Cooperative mutation based evolutionary programming for continuous function optimization, *Operations Research Letters*, 30, 195-201, 2002.
Reeves C., and J. Rowe, *Genetic Algorithms: Principles and Perspectives*, Kluwer, 2002.
Saviotti P., (Ed.), *Applied Evolutionary Economics: New Empirical Methods and Simulation Techniques*, Edward Elgar Pub., 2002.
Wilson W., *Simulating Ecological and Evolutionary Systems in C*, Cambridge University Press, 2000.

Stochastic approximation techniques

Two related stochastic approximation techniques have been proposed, one by Robbins and Monro and one by Kiefer and Wolfowitz. The first technique was not useful for optimization until an unbiased estimator for the gradient was found. Kiefer and Wolfowitz developed a procedure for optimization using finite differences. Both techniques are useful in the optimization of noisy functions, but they did not receive much attention in the simulation field until recently. Generalization and refinement of stochastic approximation procedures give rise to a weighted average, and stochastic quasi-gradient methods. These deal with constraints, non-differentiable functions, and some classes of non-convex functions, among other things.

Kiefer-Wolfowitz type techniques

Kiefer and Wolfowitz proposed a finite difference approximation to the derivative. One version of the Kiefer-Wolfwitz technique uses two-sided finite differences. The first fact to notice about the K-W estimate is that it requires 2N simulation runs, where N is the dimension of vector parameter v . If the decision maker is interested in gradient estimation with respect to each of the components of v , then 2N simulations must be run for each component of v . This is inefficient. The second fact is that it may have a very poor variance, and it may result in numerical calculation difficulties.

Robbins-Monro type techniques

The original Robbins-Monro (R-M) technique is not an optimization scheme, but rather a root finding procedure for functions whose exact values are not known but are observed with noise. Its application to optimization is immediate: use the procedure to find the root of the gradient of the objective function.

Interest was renewed in the R-M technique as a means of optimization, with the development of the perturbation analysis, score function (known also as likelihood ratio method), and frequency domain estimates of derivatives. Optimization for simulated systems based on the R-M technique is known as a "single-run" technique. These procedures optimize a simulation model in a single run simulation with a run length comparable to that required for a single iteration step in the other methods. This is achieved essentially be observing the sample values of the objective function and, based on these observations, updating the values of the controllable parameters while the simulation is running, that is, without restarting the simulation. This observing-updating sequence is done repeatedly, leading to an estimate of the optimum at the end of a single-run simulation. Besides having the potential of large computational savings, this technique can be a powerful tool in real-time optimization and control, where observations are taken as the system is evolving in time.

Gradient surface method

One may combine the gradient-based techniques with the response surface methods (RSM) for optimization purposes. One constructs a response surface with the aid of n response points and the components of their gradients.

The gradient surface method (GSM) combines the virtue of RSM with that of the single- run, gradient estimation techniques such as Perturbation Analysis, and Score Function techniques. A single simulation experiment with little extra work yields N + 1 pieces of information; i.e., one response point and N components of the gradient. This is in contrast to crude simulation, where only one piece of information, the response value, is obtained per experiment. Thus by taking advantage of the computational efficiency of single-run gradient estimators. In general, N-fold fewer experiments will be needed to fit a global surface compared to the RSM. At each step, instead of using Robbins-Monro techniques to locate the next point locally, we determine a candidate for the next point globally, based on the current global fit to the performance surface.

The GSM approach has the following advantages; The technique can quickly get to the vicinity of the optimal solution because its orientation is global [23, 39]. Thus, it produces satisfying solutions quickly; Like RSM, it uses all accumulated information; And, in addition, it uses gradient surface fitting, rather than direct performance response-surface fitting via single-run gradient estimators. This significantly reduces the computational efforts compared with RSM. Similar to RSM, GSM is less sensitive to estimation error and local optimality; And, finally, it is an on-line technique, the technique may be implemented while the system is running.

A typical optimization scheme involves two phases: a Search Phase and an Iteration Phase. Most results in analytic computational complexity assume that good initial approximations are available, and deal with the iteration phase only. If enough time is spent in the initial search phase, we can reduce the time needed in the iteration phase. The literature contains papers giving conditions for the convergence of a process; a process has to be more than convergent in order to be computationally interesting. It is essential that we be able to limit the cost of computation. In this sense, GSM can be thought of as helping the search phase and as an aid to limit the cost of computation. One can adopt standard or simple devices for issues such as stopping rules.

For on-line optimization, one may use a new design in GSM called 'single direction' design. Since for on-line optimization it may not be advisable or feasible to disturb the system, random design usually is not suitable.

Post-solution analysis

Stochastic models typically depend upon various uncertain and uncontrollable input parameters that must be estimated from existing data sets. We focus on the statistical question of how input-parameter uncertainty propagates through the model into output- parameter uncertainty. The sequential stages are descriptive, prescriptive and post-prescriptive analysis.

Rare Event Simulation

Large deviations can be used to estimate the probability of rare events, such as buffer overflow, in queueing networks. It is simple enough to be applied to very general traffic models, and sophisticated enough to give insight into complex behavior.

Simulation has numerous advantages over other approaches to performance and dependability evaluation; most notably, its modelling power and flexibility. For some models, however, a potential problem is the excessive simulation effort (time) required to achieve the desired accuracy. In particular, simulation of models involving rare events, such as those used for the evaluation of communications and highly-dependable systems, is often not feasible using standard techniques. In recent years, there have been significant theoretical and practical advances towards the development of efficient simulation techniques for the evaluation of these systems.

Methodologies include: Techniques based on importance sampling, The "restart" method, and Hybrid analytic/simulation techniques among newly devised approaches.

Conclusions & Further Readings

With the growing incidence of computer modeling and simulation, the scope of simulation domain must be extended to include much more than traditional optimization techniques. Optimization techniques for simulation must also account specifically for the randomness inherent in estimating the performance measure and satisfying the constraints of stochastic systems. We described the most widely used optimization techniques that can be effectively integrated with a simulation model. We also described techniques for post-solution analysis with the aim of theoretical unification of the existing techniques. All techniques were presented in step-by-step format to facilitate implementation in a variety of operating systems and computers, thus improving portability.

General comparisons among different techniques in terms of bias, variance, and computational complexity are not possible. However, a few studies rely on real computer simulations to compare different techniques in terms of accuracy and number of iterations. Total computational effort for reduction in both the bias andvariance of the estimate depends on the computational budget allocated for a simulation optimization. No single technique works effectively and/or efficiently in all cases.

The simplest technique is the random selection of some points in the search region for estimating the performance measure. In this technique, one usually fixes the number of simulation runs and takes the smallest (or largest) estimated performance measure as the optimum. This technique is useful in combination with other techniques to create a multi-start technique for global optimization. The most effective technique to overcome local optimality for discrete optimization is the Tabu Search technique. In general, the probabilistic search techniques, as a class, offer several advantages over other optimization techniques based on gradients. In the random search technique, the objective function can be non-smooth or even have discontinuities. The search program is simple to implement on a computer, and it often shows good convergence characteristics in noisy environments. More importantly, it can offer the global solution in a multi-modal problem, if the technique is employed in the global sense. Convergence proofs under various conditions are given in.

The Hooke-Jeeves search technique works well for unconstrained problems with less than 20 variables; pattern search techniques are more effective for constrained problems. Genetic techniques are most robust and can produce near-best solutions for larger problems. The pattern search technique is most suitable for small size problems with no constraint, and it requires fewer iterations than the genetic techniques. The most promising techniques are the stochastic approximation, simultaneous perturbation, and the gradient surface methods. Stochastic approximation techniques using perturbation analysis, score function, or simultaneous perturbation gradient estimators, optimize a

simulation model in a single simulation run. They do so by observing the sample values of the objective function, and based on these observations, the stochastic approximation techniques update the values of the controllable parameters while the simulation is running and without restarting the simulation. This observing-updating sequence, done repeatedly, leads to an estimate of the optimum at the end of a single-run simulation. Besides having the potential of large savings in computational effort in the simulation environment, this technique can be a powerful tool in real-time optimization and control, where observations are taken as the system is evolving over time.

Response surface methods have a slow convergence rate, which makes them expensive. The gradient surface method combines the advantages of the response surface methods (RSM) and efficiency of the gradient estimation techniques, such as infinitesimal perturbation analysis, score function, simultaneous perturbation analysis, and frequency domain technique. In the gradient surface method (GSM) the gradient is estimated, and the performance gradient surface is estimated from observations at various points, similar to the RSM. Zero points of the successively approximating gradient surface are then taken as the estimates of the optimal solution. GSM is characterized by several attractive features: it is a single run technique and more efficient than RSM; at each iteration step, it uses the information from all of the data points rather than just the local gradient; it tries to capture the global features of the gradient surface and thereby quickly arrive in the vicinity of the optimal solution, but close to the optimum, they take many iterations to converge to stationary points. Search techniques are therefore more suitable as a second phase. The main interest is to figure out how to allocate the total available computational budget across the successive iterations.

For when the decision variable is qualitative, such as finding the best system configuration, a random or permutation test is proposed. This technique starts with the selection of an appropriate test statistic, such as the absolute difference between the mean responses under two scenarios. The test value is computed for the original data set. The data are shuffled (using a different seed); the test statistic is computed for the shuffled data; and the value is compared to the value of the test statistic for the original, un-shuffled data. If the statistics for the shuffled data are greater than or equal to the actual statistic for the original data, then a counter c, is incremented by 1. The process is repeated for any desired m number of times. The final step is to compute (c+1)/(m+1), which is the significant level of the test. The null hypothesis is rejected if this significance level is less than or equal to the specified rejection level for the test. There are several important aspects to this nonparametric test. First, it enables the user to select the statistic. Second, assumptions such as normality or equality of variances made for the t-test, ranking-and-selection, and multiple-comparison procedures, are no longer needed. A generalization is the well-known bootstrap technique.

What Must Be Done

1. computational studies of techniques for systems with a large number of controllable parameters and constraints.

2. effective combinations of several efficient techniques to achieve the best results under constraints on computational resources.

3. development of parallel and distributed schemes

4. development of an expert system that incorporates all available techniques.

For more, visit the Web site: Modeling & Simulation Resources.

References & Further Readings:

Arsham H., Techniques for Monte Carlo Optimizing, *Monte Carlo Methods and Applications*, 4(3), 181-230, 1998.
Arsham H., Stochastic Optimization of Discrete Event Systems Simulation, *Microelectronics and Reliability*, 36(10), 1357-1368, 1996.
Fu M., and J-Q. Hu, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, Kluwer Academic Publishers, 1997.
Rollans S. and D. McLeish, Estimating the optimum of a stochastic system using simulation, *Journal of Statistical Computation and Simulation*, 72, 357 - 377, 2002.
Rubinstein R., and A. Shapiro, *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*, John Wiley & Sons, 1993.

Metamodeling and the Goal seeking Problems

The simulation models although simpler than the real-world system, are still a very complex way of relating input (v) to output J(v). Sometimes a simpler analytic model may be used as an auxiliary to the simulation model. This auxiliary model is often referred to as a metamodel.

In many simulation applications such as systems analysis and design applications, the decision maker may not be interested in optimization but wishes to achieve a certain value for J(v), say J_0. This is the goal-seeking problem: given a target output value J_0 of the performance and a parameterized pdf family, one must find an input value for the parameter, which generates such an output.

Metamodeling

The simulation models although simpler than the real-world system, are still a very complex way of relating input (v) to output J(v). Sometimes a simpler analytic model may be used as an auxiliary to the simulation model. This auxiliary model is often referred to as a metamodel. There are several techniques available for metamodeling including: design of experiments, response surface methodology, Taguchi methods, neural networks, inductive learning, and kriging. Metamodeling may have different purposes: model simplification and interpretation, optimization, what-if analysis, and generalization to models of the same type. The following polynomial model can be used as an auxiliary model.

$$J(v) = J(v_0) + \delta\ v.J'(v_0) + (\delta\ v)^2\ J''(v_0) / 2 + ...,$$

where $\delta\ v = v\text{-}v_0$ and the primes denote derivatives. This metamodel approximates J(v) for small $\delta\ v$. To estimate J(v) in the neighborhood of v_0 by a linear function, we need to estimate the nominal J(v) and its first derivative. Traditionally, this derivative is estimated by crude Monte Carlo; i.e., finite difference which requires rerunning the simulation model. Methods which yield enhanced efficiency and accuracy in estimating, at little additional computational (Not simulation) cost, are presented in this site. The Score Function method of estimating the first derivative is:

$$J'(v) = E_{Y|v}\ [\ Z(y(v))\ .S]$$

where S=f'(y; v) / f(y; v)=d Lnf(y; v) / dv is the Score function and differentiations is with respect to v, provided that, f'(y; v) exist, and f(y; v) is positive for all v in V.

The Score function approach can be extended in estimating the second and higher order of derivatives. For example, an estimate for the second derivative based on the Score Function method is:

$$J''(v_0) = E\ Z(y_i).H(y_i\ ;\ v_0),$$

where,

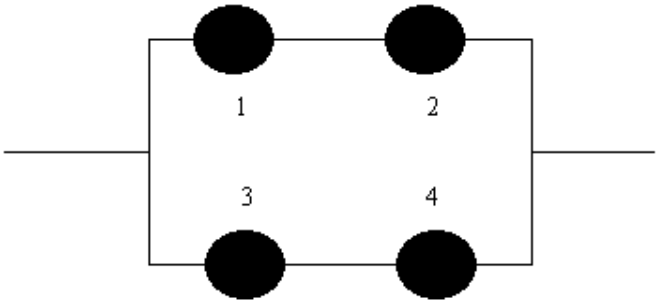
$$H(y_i;\ v_0) = f''(y_i;\ v_0) / f\ (y_i;\ v_0).$$

Where S and $H = S' + S^2$ are the score and information functions, respectively, widely used in statistics literature, such as in the construction of Cramer-Rao bounds. By having gradient and Hessian in our disposal, we are able to construct a second order local metamodel using the Taylor's series.

An Illustrative Numerical Example: For most complex reliability systems, the performance measures such as mean time to failure (MTTF) are not available in analytical form. We resort to Monte Carlo Simulation (MCS) to estimate MTTF function from a family of single-parameter density functions of the components life with specific value for the parameter. The purpose of this section is to solve the inverse problem, which deals with the calculation of the components' life parameters (such as MTTF)

of a homogeneous subsystem, given a desired target MTTF for the system. A stochastic approximation algorithm is used to estimate the necessary controllable input parameter within a desired range of accuracy. The potential effectiveness is demonstrated by simulating a reliability system with a known analytical solution.

Consider the coherent reliability sub-system with four components component 1, and 2 are in series, and component 3 and 4 also in series, however these two series of components are in parallel, as illustrated in the following Figure. All components are working independently and are homogeneous; i.e., manufactured by an identical process, components having independent random lifetimes Y1, Y2, Y3, and Y4, which are distributed exponentially with rates $v = v_0 = 0.5$.



A simple reliability subsystem

The system lifetime is

$$Z\,(Y1,Y2,Y3,Y4;\, v_0) = \max\,[\min\,(Y3,Y4),\, \min\,(Y1,Y2)].$$

It is readily can be shown that the theoretical expected lifetime of this sub-system is

$$J(v_0) = 3/(4\, v_0).$$

The underlying pdf for this system is:

$$f(y;\, v) = v^4 \exp(-v\, \Sigma\ y_i),$$
the sum is over i = 1, 2, 3, 4.

Applying the Score function method, we have:

$$S(y) = f\,'\,(y;\, v) \,/\, f(y;\, v) = 4/v - \Sigma\ \ y_i,$$
the sum is over i = 1, 2, 3, 4.

and

$$H(y) = f\,'\,'\,(y;\, v) \,/\, f(y;\, v) = [v^2\,(\Sigma\ \ y_i)^2 - 8v\,(\Sigma\ \ y_i) + 12] \,/\, v^2,$$
the sums are over i = 1, 2, 3, 4.

The estimated average lifetime and its derivative for the nominal system with $v = v_0 = 0.5$, are:

$$J(v_0\,) = \Sigma\ \ \max\,[\min\,(Y_{3,\,j}, Y_{4,\,j}),\, \min\,(Y_{1,\,j},\, Y_{2,\,j})] \,/\, n,$$

$$J'(v_0\,) = \Sigma\ \ \max\,[\min\,(Y_{3,\,j}, Y_{4,\,j}),\, \min\,(Y_{1,\,j},\, Y_{2,\,j})]\ .\ S(Y_{i,j}) \,/\, n,$$

and

$$J''(v_0\,) = \Sigma\ \ \max[\min\,(Y_{3,\,j}, Y_{4,\,j}),\, \min\,(Y_{1,\,j},\, Y_{2,\,j})]\ .\ H(Y_{i,\,j})/n,$$

respectively, where $Y_{i, j}$ is the j^{th} observation for the i^{th} component (i = 1, 2, 3, 4). We have performed a Monte Carlo experiment for this system by generating n = 10000 independent replications using SIMSCRIPT II.5 random number streams 1 through 4 to generate exponential random variables Y1, Y2, Y3, Y4 , respectively, on a VAX system. The estimated performance is J(0.5) = 1.5024, with a standard error of 0.0348. The first and second derivatives estimates are -3.0933 and 12.1177 with standard errors of 0.1126 and 1.3321, respectively.

The response surface approximation in the neighborhood of v = 0.5 is:

$$J(v) = 1.5024 + (v - 0.5)(-3.0933) + (v - 0.5)^2 (12.1177)/2 =$$
$$6.0589v^2 - 9.1522v + 4.5638$$

A numerical comparison based on exact and the approximation by this metamodel reveals that the largest absolute error is only 0.33% for any v in the range of [0.40, 0.60]. This error could be reduced by either more accurate estimates of the derivatives and/or using a higher order Taylor expansion. A comparison of the errors indicates that the errors are smaller and more stable in the direction of increasing v. This behavior is partly due to the fact that lifetimes are exponentially distributed with variance 1/v. Therefore, increasing v causes less variance than the nominal system (with v = 0.50).

Goal seeking problem

In many systems modeling and simulation applications, the decision maker may not be interested in optimization but wishes to achieve a certain value for J(v), say J_0. This is the goal-seeking problem: given a target output value J_0 of the performance and a parameterized pdf family, one must find an input value for the parameter, which generates such an output. When is a controllable input, the decision maker may be interested in the goal-seeking problem: namely, what change of the input parameter will achieve a desired change in the output value. Another application of the goal-seeking problem arises when we want to adapt a model to satisfy a new equality constraint with some stochastic functions. We may apply the search techniques, but the goal-seeking problem can be considered as an interpolation based on a meta-model. In this approach, one generates a response surface function for J(v). Finally, one uses the fitted function to interpolate for the unknown parameter. This approach is tedious, time-consuming, and costly; moreover, in a random environment, the fitted model might have unstable coefficients.

For a given J(v) the estimated δ v, using the first order approximation is:

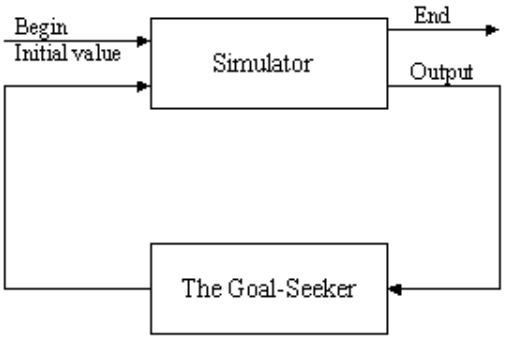
$$\delta v = [J(v)-J(v_0)] / J'(v_0),$$

provided that the denominator does not vanish for all v_0 in set V.

The Goal-seeker Module: The goal-seeking problem can be solved as a simulation problem. By this approach, we are able to apply variance reduction techniques (VRT) used in the simulation literature. Specifically, the solution to the goal-seeking problem is the unique solution of the stochastic equation $J(v) - J_0 = 0$. The problem is to solve this stochastic equation by a suitable experimental design, to ensure convergence. The following is a Robbins- Monro (R-M) type technique.

$$v_{j+1} = v_j + d_j [J_0 - J(v_j)] / J'(v_j),$$

where d_j is any divergent sequence of positive numbers. Under this conditions, $\delta v = J_0 - J(v_j)$ converges to approach zero while dampening the effect of the simulation random errors. These conditions are satisfied, for example, by the harmonic sequence $d_j = 1/j$. With this choice, the rate of reduction of di is very high initially but may reduce to very small steps as we approach the root. Therefore, a better choice is, for example $d_j = 9 / (9 + j)$. This technique involves placing experiment i+1 according to the outcome of experiment i immediately preceding it, as is depicted in the following Figure:



Under these not unreasonable conditions, this algorithm will converge in mean square; moreover, it is an almost sure convergence. Finally, as in Newton's root-finding method, it is impossible to assert that the method converges for just any initial $v = v_0$, even though J'(v) may satisfy the Lipschits condition over set V. Indeed, if the initial value v_0 is sufficiently close to the solution, which is usually the case, then this algorithm requires only a few iterations to obtain a solution with very high accuracy.

An application of the goal-seeker module arises when we want to adapt a model to satisfy a new equality constraint (condition) for some stochastic function. The proposed technique can also be used to solve integral equations by embedding the Importance Sampling techniques within a Monte Carlo sampling.

One may extend the proposed methodology to the inverse problems with two or more unknown parameters design by considering two or more relevant outputs to ensure uniqueness. By this generalization we could construct a linear (or even nonlinear) system of stochastic equations to be solved simultaneously by a multidimensional version of the proposed algorithm. The simulation design is more involved for problems with more than a few parameters.

For more, visit the Web site: Modeling & Simulation Resources.

References and Further Readings:

Arsham H., The Use of Simulation in Discrete Event Dynamic Systems Design, *Journal of Systems Science*, 31(5), 563-573, 2000.
Arsham H., Input Parameters to Achieve Target Performance in Stochastic Systems: A Simulation-based Approach, *Inverse Problems in Engineering*, 7(4), 363-384, 1999.
Arsham H., Goal Seeking Problem in Discrete Event Systems Simulation, *Microelectronics and Reliability*, 37(3), 391-395, 1997.
Batmaz I., and S. Tunali, Small response surface designs for metamodel estimation, *European Journal of Operational Research*, 145(3), 455-470, 2003.
Ibidapo-Obe O., O. Asaolu, and A. Badiru, A New Method for the Numerical Solution of Simultaneous Nonlinear Equations, *Applied Mathematics and Computation*, 125(1), 133-140, 2002.
Lamb J., and R. Cheng, Optimal allocation of runs in a simulation metamodel with several independent variables, *Operations Research Letters*, 30(3), 189-194, 2002.
Simpson T., J. Poplinski, P. Koch, and J. Allen, Metamodels for Computer-based Engineering Design: Survey and Recommendations, *Engineering with Computers*, 17(2), 129-150, 2001.
Tsai C-Sh., Evaluation and optimisation of integrated manufacturing system operations using Taguch's experiment design in computer simulation, *Computers And Industrial Engineering*, 43(3), 591-604, 2002.

"What-if" Analysis Techniques

Introduction

The simulation models are often subject to errors caused by the estimated parameter(s) of underlying input distribution function. "What-if" analysis is needed to establish confidence with respect to small changes in the parameters of the input distributions. However the direct approach to "what-if" analysis requires a separate simulation run for each input value. Since this is often inhibited by cost, as an alternative, what people are basically doing in practice is to plot results and use a simple linear interpolation/extrapolation. This section presents some simulation-based techniques that utilize the current information for estimating performance function for several scenarios without any additional simulation runs.

Simulation continues to be the primary method by which system analysts obtain information about analysis of complex stochastic systems. In almost all simulation models, an expectedvalue can

express the system's performance. Consider a system with continuous parameter v ⎣ V, where V is the feasible region. Let

$$J(v) = E_{Y \mid v}[Z(Y)]$$

be the steady state expected performance measure, where Y is a random vector with known probability density function (pdf), f(y; v) depends on v, and Z is the performance measure.

In discrete event systems, Monte Carlo simulation is usually needed to estimate J(v) for a given value v. By the law of large numbers

$$J(v) = 1/n \Sigma \quad Z(y_i)$$

where y_i, i = 1, 2,..., n are independent, identically distributed, random vector realizations of Y from f (y; v), and n is the number of independent replications. This is an unbiased estimator for J(v) and converges to J(v) by law of large numbers.

There are strong motivations for estimating the expected performance measure J(v) for a small change in v to v + δ v, that is to solve the so-called "what if" problem.

The simulationist must meet managerial demands to consider model validation and cope with uncertainty in the estimation of v. Adaptation of a model to new environments also requires an adjustment in v.

An obvious solution to the "what if" problem is the Crude Monte Carlo (CMC) method, which estimates J(v+ δ v) for each v separately by rerunning the system for each v+ δ v. Therefore costs in CPU time can be prohibitive The use of simulation as a tool to design complex computer stochastic systems is often inhibited by cost. Extensive simulation is needed to estimate performance measures for changes in the input parameters. As as an alternative, what people are basically doing in practice is to plot results of a few simulation runs and use a simple linear interpolation/extrapolation.

In this section we consider the "What-if" analysis problem by extending the information obtained from a single run at the nominal value of parameter v to the closed neighborhood. We also present the use of results from runs at two or more points over the intervening interval. We refer to the former as extrapolation and the latter as interpolation by simulation. The results are obtained by some *computational cost* as opposed to *simulation cost*. Therefore, the proposed techniques are for estimating a performance measure at multiple settings from a simulation at a nominal value.

Likelihood Ratio (LR) Method

A model based on Radon-Nikodym theorem to estimate J(v+ δ v) for stochastic systems in a single run is as follows:

$$J(v+ \delta v) = E_{Y \mid v + \delta v}[Z(Y)] = E_{Y \mid v}[Z(Y).W]$$

where the likelihood ratio W is:

$$W = f(y; v+ \delta v) / f(y; v)$$

adjusts the sample path, provided f(y; v) does not vanish. Notice that by this change of probability space, we are using the common realization as J(v).

The generated random vector y is roughly representative of Y, with f(v). Each of these random observations, could also hypothetically came from f(v+ δ v). W weights the observations according to this phenomenon.

Therefore, the "What-if" estimate is:

$$J(v) = 1/n \Sigma \quad Z(y_i).W_i$$

which is based on only one sample path of the system with parameter v and the simulation for the system with v+ δ v is not required.

Unfortunately LR produces a larger variance compared with CMC. However, since E(W)=1, the following variance reduction techniques (VRT) may improve the estimate.

$$J(v) = \Sigma \quad Z(y_i).W_i / \Sigma \quad W_i$$

Exponential Tangential in Expectation Method

In the statistical literature the efficient score function is defined to be the gradient

$$S(y) = d \, Ln \, f(y; v) / dv$$

We consider the exponential (approximation) model for J(v+ δ v) in a first derivative neighborhood of v by:

$$J(v+ \delta v) = E [Z(y). \exp[\delta vS(y)] / E[\exp(\delta S(y))]$$

Now we are able to estimate J(v+ δ v) based on n independent replications as follows:

$$J(v+ \delta v) = \Sigma \quad A_i / \Sigma \quad B_i$$

where,

$$A_i = Z(y_i).\exp[\delta v.S(y_i)]$$

and

$$B_i = \exp[\delta v. S(y_i)].$$

Taylor Expansion of Response Function

The following linear Taylor model can be used as an auxiliary model.

$$J(v + \delta v) = J(v) + \delta v.J'(v) + ...,$$

where the prime denotes derivative. This metamodel approximates J(v + δ v)) for small δ v. For this estimate, we need to estimate the nominal J(v) and its first derivative. Traditionally, this derivative is estimated by crude Monte Carlo; i.e., finite difference, which requires rerunning the simulation model. Methods which yield enhanced efficiency and accuracy in estimating, at little additional cost, are of great value.

There are few ways to obtain efficiently the derivatives of the output with respect to an input parameter as presented earlier on this site. The most straightforward method is the Score Function (SF). The SF approach is the major method for estimating the performance measure and its derivative, while observing only a single sample path from the underlying system. The basic idea of SF is that the derivative of the performance function, J'(v), is expressed as expectation with respect to the same distribution as the performance measure itself.

Therefore, for example, using the estimated values of J(v) and its derivative J'(v), the estimated J(v + δ v) is:

$$J(v + \delta v) = J(v) + \delta v \, J'(v_0)$$

with variance

$$\text{Var}[J(v+\delta\ v)\ ] = \text{Var}[J(v)] + (\delta\ v)^2\ \text{Var}[J'(v)] + 2\delta\ v\ \text{Cov}[J(v), J'(v)].$$

This variation is needed for constructing a confidence interval for the perturbed estimate.

Interpolation Techniques

Given two points, v1 and v2 (scalars only) "sufficiently" close, one may simulate at these two points then interpolates for any desired points in between. Assuming the given v1 and v2 are sufficiently close and looks for the "best" linear interpolation in the sense of minimum error on the interval. Clearly,

$$J(v) = E_{Y\ |\ v}[Z\ (Y)] = \phi\ \ .\ E_{Y\ |\ v}[Z\ (Y)] + (1 - \phi\ )\ .\ E_{Y\ |\ v}[Z\ (Y)]$$

Similar to the Likelihood Ratio approach, this can be written as:

$$J(v) = E_{Y\ |\ v}[Z\ (Y)] = \phi\ \ .\ E_{Y\ |\ v1}[Z\ (Y).W1] + (1 - \phi\ )\ .\ E_{Y\ |\ v2}[Z\ (Y).W2]$$

where the likelihood ratios W1 and W2 are W1 = f(y; v) / f(y; v1) and W2 = f(y; v) / f(y; v2), respectively.

One obvious choice is $\phi\ $ = f(y; v1) / [f(y; v1)+f(y; v2)]. This method can easily extended to k-point interpolation.

For 2-point interpolation, if we let $\phi\ $ to be constant within the interval [0, 1], then the linear interpolated "what-if" estimated value is:

$$J(v) = \phi\ \ .\ J_1 + (1-\phi\ )\ .\ J_2$$

where the two estimates on the RHS of are two independent Likelihood Ratio extrapolations using the two end-points.

We define $\phi\ ^*$ as the $\phi\ $ in this convex combination with the minimum error in the estimate. That is, it minimizes

$$\text{Var}[J(v)] = \phi\ ^2\text{Var}[J_1] + (1-\phi\ )^2\text{Var}[J_2)]$$

By the first order necessary and sufficient conditions, the optimal $\phi\ $ is:

$$\phi\ ^* = \text{Var}[J_2] / \{\text{Var}[J_1] + \text{Var}[J_2]\}$$

Thus, the "best linear" interpolation for any point in interval [v1, v2] is:

$$J(v) = \phi\ ^*\ .\ J_1 + (1- \phi\ ^*)\ .\ J_2$$

which is the optimal interpolation in the sense of having minimum variance.

Conclusions & Further Readings

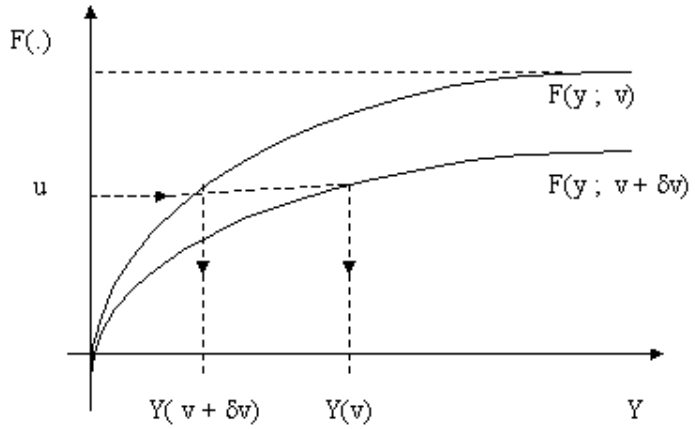
Estimating system performance for several scenarios via simulation generally requires a separate simulation run for each scenario. In some very special cases, such as the exponential density f(y; v)=ve^{-vy}, one could have obtained the perturbed estimate using Perturbation Analysis directly as follow. Clearly, one can generate random variate Y by using the following inverse transformation:

$$Y_i=(1/v)\ \text{Ln}\ (1/U_i)$$

where Ln is the natural logarithm and U_i is a random number distributed Uniformly [0,1]. In the case of perturbed v, the counterpart realization using the same U_i is

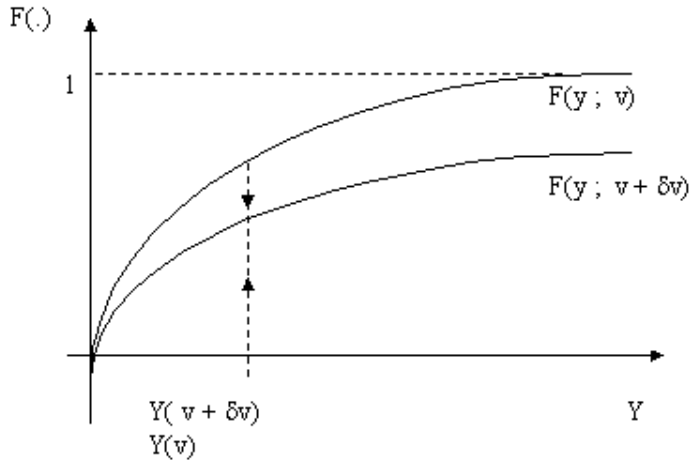
$$Y_i=[1/(v+\delta\ v)]\ \text{Ln}\ (1/U_i).$$

Clearly, this single run approach is limited, since the inverse transformation is not always available in closed form. The following Figure illustrates the Perturbation Analysis Method:



Parameters(s) Perturbation via Perturbation Analysis Method

Since the Perturbation Analysis Approach has this serious limitation, for this reason, we presented some techniques for estimating performance for several scenarios using a single-sample path, such as the Likelihood Ratio method, which is illustrated in the following Figure.



Parameters(s) Perturbation via Likelihood Ratio Method

Research Topics: Items for further research include:

i) to introduce efficient variance reduction and bias reduction techniques with a view to improving the accuracy of the existing and the proposed methods;

ii) to incorporate the result of this study in a random search optimization technique. In this approach one can generate a number of points in the feasible region uniformly distributed on the surface of a hyper-sphere each stage the value of the performance measure is with a specified radius centered at a starting point. At estimated at the center (as a nominal value). Perturbation analysis is used to estimate the performance measure at the sequence of points on the hyper-sphere. The best point (depending whether the problem is max or min) is used as the center of a smaller hyper- sphere. Iterating in this fashion one can capture the optimal solution within a hyper-sphere with a specified

small enough radius. Clearly, this approach could be considered as a sequential self-adaptive optimization technique.

iii) to estimate the sensitivities i.e. the gradient, Hessian, etc. of J(v) can be approximated using finite difference. For example the first derivative can be obtained in a single run using the Likelihood Ratio method as follows:

$$J'(v) = [\Sigma\ (Z_i\ W_i - Z_i)]/(n\ \delta\ v),$$

$$J'(v) = [\Sigma\ (Z_i\ W_i - Z_i)]/(\delta\ v.\ \Sigma\ W_i),$$

or

$$J(v) = \Sigma\ Z_i[W_i^{+} - W_i^{-}] / \{\ 2\ \Sigma\ [(1+\ \delta\ v)W_i^{+} - (1-\delta\ v)W_i^{-}]\}$$

the sums are over all i, i =1, 2, 3,.., n, where

$$W_i^{+} = f(y:v+\ v)/f(y;\ v),\ \text{and}\ W_i^{-} = f(y;\ v-\ v)/f(y;\ v).$$

The last two estimators may induce some variance reductions.

iv) Other interpolation techniques are also possible. The most promising one is based on Kriging. This technique gives more weight to `neighboring' realizations, and is widely used in geo-statistics.

Other items for further research include some experimentation on large and complex systems such as a large Jacksonian network with routing that includes feedback loops in order to study the efficiency of the presented technique.

For more, visit the Web site: Modeling & Simulation Resources.

References & Further Readings:

Arsham H., Performance Extrapolation in Discrete-event Systems Simulation, *Journal of Systems Science*, 27(9), 863-869, 1996.
Arsham H., A Simulation Technique for Estimation in Perturbed Stochastic Activity Networks, *Simulation*, 58(8), 258-267, 1992.
Arsham H., Perturbation Analysis in Discrete-Event Simulation, *Modelling and Simulation*, 11(1), 21-28, 1991.
Arsham H., What-if Analysis in Computer Simulation Models: A Comparative Survey with Some Extensions, *Mathematical and Computer Modelling*, 13(1), 101-106, 1990.
Arsham H., Feuerverger, A., McLeish, D., Kreimer J. and Rubinstein R., Sensitivity analysis and the what-if problem in simulation analysis, *Mathematical and Computer Modelling*, 12(1), 193-219, 1989.
PDF Version

Kindly e-mail me your comments, suggestions, and concerns. Thank you.

Professor Hossein Arsham

This site was launched on 2/11/1995, and its intellectual materials have been thoroughly revised on a yearly basis. The current version is the 9th Edition. All external links are checked once a month.

This project was supported, in part by the
National Science Foundation
Opinions expressed are those of the authors and not necessarily those of the Foundation

Back to:

Dr Arsham's Home Page