SIMULATION 74:5, 264-274
©2000, Simulation Councils Inc.
ISSN 0037-5497/00
Printed in the United States of America

Poisson Simulation—A Method for Generating Stochastic Variations in Continuous System Simulation

Leif Gustafsson
Department of Agricultural Engineering
Swedish University of Agricultural Sciences
Box 7033, S-750 07 Uppsala, Sweden
e-mail: leif.gustafsson@lt.slu.se

In continuous systems simulation a model is built as a system of differential equations. An implicit assumption is that the number of items is so large that the changes can then be regarded as continuous. However, many systems can be modeled by differential equations when the numbers involved are moderately large. Such systems show stochastic variations that can be described in terms of events per time unit. If the events happen one at a time, and if the number of events during the interval is independent of both the number of past events and the times these events occurred, we have a Poisson process. *The system's variations can be modeled by replacing the flow to or from a state during the integration step with a Poisson probability. This not only adds variations to the model, but can also reveal system properties not covered by continuous system simulation. Furthermore, the "intensity parameter" controlling the flow may vary over time without further problems.*

Keywords: CSS, differential equations, Poisson distribution, Poisson process, random, stochastic process

1. Introduction

Most kinds of computer simulation can be classified in static/dynamic and deterministic/stochastic terms (see Figure 1).

In dynamic simulation, Continuous System Simulation (CSS) usually represents a macro view where the number of entities involved is so large that you may think in terms of amounts and continuous flows. Therefore, you may represent your model with a system of differential equations, and statistical variations are usually not important because of big numbers. The results emerge as curves of the states, flows or auxiliary quantities as functions of time.

Discrete Event Simulation (DES), on the other hand, represents a micro view where each individual entity (actor) is modeled in terms of its actions and interactions at certain events of time. The modeling focuses on actors with properties and behavior and on resources of different kinds. The micro perspective also implies that the results emerge as statistics of events, waiting times, utilization of resources, etc.

However, a model of differential equations often gives good representation of the dynamics of the system, when the number of entities is not so large, but

Stochastic	• Statistical models *Monte Carlo Simulation*	• Actors with logical behavior • Stochastic differential and difference equation models *Discrete Event Simulation* *Simulation of Markovian models*
Deterministic	• Algebraic models *Experiments on spreadsheet models*	• Systems of differential equations *Continuous System Simulation*
	Static	***Dynamic***

Figure 1. Classification of models and simulation techniques in the terms static/dynamic and deterministic/stochastic.

then the stochastic variations usually *can't* be neglected. This may, for example, be the case when describing an ecological system of animals or plants, an epidemiological system of prevalences and incidences, effects of a medical screening program, the plants in an agricultural system, or migration of people between areas. Often randomness is then handled by adding noise of some distribution to input, state or output variables, which is often not a proper way to handle stochastic aspects.

In many stochastic processes in nature a random number of events occurs during a time interval. If the events happen one at a time, and if the number of events during the interval is independent both of the number of events in the past and of the times these events occurred, then we have by definition a *Poisson process*. Examples of such stochastic processes may be the number of decays per second from a radioactive source, the number of phone calls per hour to an office, the number of new cases of cancer per year in a population, the number of customers per hour arriving at a shop, the number of offspring per year in a population, etc. The expected number of events per time unit (intensity) may be constant or vary over time. If it is constant, we have a *stationary* Poisson process; otherwise it is *nonstationary*. In both cases, the number of events during an interval, dt, is described by a Poisson distribution, Po(dt*λ), where the intensity λ (events per time unit) times the interval length dt is the expected number of events during the interval. In modeling, Poisson-distributed samples are easily obtained from a random number generator.

When the stochastics in a system originate from a Poisson process, it can easily be modeled by including a call to a Poisson random number generator in a flow rate equation. We call this technique Poisson simulation.

The purpose of this compendium is to present a method for handling stochastic variations occurring from finite numbers in the frame of a system of differential equations. Hereby, we also offer a straightforward and simple alternative to use discrete event simulation in order to handle stochastic variations.

2. Method and Realization

2.1 Outline: Modification of a Simple CSS Model

A CSS model is represented by a system of ordinary differential equations. Each equation can be written as: dx/dt = f(x,t), together with an initial value x(0) = xo. In CSS languages this dynamic equation is sometimes written as: x = INTEG(f, xo), or in the Euler approximation: $x(t + \Delta t) \approx x(t) + \Delta t{*}f(t)$, where f is the net flow rate to/from the state x.

For computational purposes, the dynamic model may be separated into dynamic state equations, and flow rate equations (and possibly also auxiliary equations), the last two types being purely algebraic. The order of computation is state equations, then auxiliary equations and, finally, the flow rate equations.

For the sake of simplicity, we demonstrate this with a first-order system in the Euler form. This system has one state x and one outflow rate f which is proportional to the state value. A computer program then gets a model structure such as:

```
  Initialization
AGAIN:
  x:=x + dt*( –f )
  f := p*x
  time:=time + dt
  if time ≤ Tend then Goto AGAIN
```

For each time step, dt, a certain fraction of the state value, p*x, leaves the state as an outflow. When the state represents, e.g., the amount of water or gas, the number of molecules is so large that we have virtually no stochastic variations. But when the state variable represents rabbits, prevalence of sick persons, emigrants, etc., the number, x, may be large—but not so large that we can neglect the stochastic variations. The *average* outflow is still f = p*x per time unit or dt*f = dt*p*x during the time interval dt. If the properties of single events and independency, discussed above, are fulfilled, the number of events during dt should be Poisson-distributed with the intensity λ = f = p*x. Thus, the outflow during the time interval, dt, has a Poisson distributed variation denoted Po(dt*λ). The flow rate then becomes Po(dt*λ)/dt. We therefore reformulate the model as:

```
    Initialization
  AGAIN:
    x: = x + dt*( –f )
    f : = Po(dt*p*x)/dt
    time: = time + dt
    if time ≤ Tend then Goto AGAIN
```

This model correctly reveals the stochastic properties with regard to the number of items in state x.

In general, the states in a CSS model are updated by inflows and outflows. Some flows may be deterministic and some Poisson-distributed, and for other flows, other distributions may be used.

2.2 *Some Properties of the Poisson Distribution*

A stochastic process of events is said to be a *Poisson process* if:

1. Events happens one at a time (not in batches).
2. The number of events in the interval (t, t+dt] is independent of the number of events in the past and independent of the times these events occurred.

If the expected number of events per time unit (intensity = λ) does not depend on time, we have a *stationary* Poisson process, while if the intensity varies with time ($\lambda = \lambda(t)$) we have a *nonstationary* Poisson process.

Note that the flow rate in a Poisson simulation is an approximation of a Poisson process, since events are not defined as events in time in DES meaning, but as a number of events during a small time interval, dt.

For each time interval, dt, of the Poisson process, the number of events is a Poisson-distributed stochastic variable with the expected value $\theta = dt^*\lambda$, denoted as Po(dt*λ).

The *Poisson distribution*, Po(θ), is a discrete distribution that describes independent events in time (or space). The density function is: $p(k) = e^{-\theta} \cdot \theta^k/k!$ where $k = 0, 1, 2, \ldots$

To understand the Poisson distribution and its use in this paper, a list of fundamental properties of this distribution is listed below. For a more detailed description of the Poisson distribution and Poisson processes, see a textbook on elementary statistics or simulation [1, 2].

- If the stochastic variable $X \in Po(\theta)$, then $E(X) = \theta$, and $Var(X) = \theta$.
- The Poisson distribution has an important property of addition: if $X \in Po(\theta_1)$ and $Y \in Po(\theta_2)$, where X and Y are *independent*, then $X + Y \in Po(\theta_1 + \theta_2)$.
- For large θ, $X \in N(\theta, \sqrt{\theta})$ is a good approximation. (θ should be larger than, say, 15.) Note that the Normal distribution is not a discrete distribution, and thus some form of roundoff must be made in order to keep the integer property, if required.

2.3 *The Automatic Control Approach*

Stochastic difference or differential equation models are studied in the field of automatic control [3], but in a different form. In the linear theory, a given model with a given step size Δt is represented by:

$$x(t + \Delta t) = Fx(t) + Gu(t) + e_1(t)$$

$$y(t) = Hx(t) + e_2(t)$$

where x is a state vector, u is an input vector, y is an output vector and F, G and H are matrices. The stochastic term $e_1(t)$ is the state noise and $e_2(t)$ is the output noise. The discrete time noise is sampled from some distribution (e.g., Normal) and with some variances.

For example, the first-order model presented above can (if $\theta = dt^*p^*x$ is large) be modeled with an additive, state-dependent, time varying noise as: $x(t + \Delta t) = x(t) - \Delta t\ {}^*p^*x(t) - \Delta t^*e(t)$, where e(t) is a Gaussian noise with the (variable) variance equal to p*x(t).

2.4 *Example 1: The Lotka-Volterra Equations*

The Lotka-Volterra equations describe a prey-predator system by differential equations [4, 5].

Assume that you have two species, X (rabbits) and Y (foxes), in a restricted area. The rabbits breed at a rate proportional to their number X. They die because of encounters with the foxes, which is proportional to X*Y. Also, there is competition among the rabbits, where each rabbit competes with all the others. Competition, therefore, is proportional to X*X. The encounters with rabbits give the foxes energy to breed, so they increase in proportion to X*Y. Finally, the fox death rate is proportional to the number of foxes, Y. The Lotka-Volterra model therefore has the form:

$$dX/dt = aX - bXY - kX^2$$

$$dY/dt = cXY - dY$$

where a, b, c, d and k are proportionality constants for fertility, mortality and competition.

By setting the derivatives dX/dt and dY/dt to zero and solving for X and Y, we obtain three possible stationary solutions:

1. $X = 0$ and $Y = 0$
2. $X = a/k$ and $Y = 0$
3. $X = d/c$ and $Y = (a - kd/c)/b$

Setting a = 0.2, b = 0.005, c = 0.005, d = 0.3, and k = 0.001 gives in the second case (foxes become extinct) X = 200 and Y = 0, and in the third case (both species survive) X = 60 and Y = 28.

Some CSS languages, such as Powersim [6], have a built-in Poisson random number generator. If not, it can usually be built and included as a macro or external function. (See Appendix A.)

A Powersim model of the Lotka-Volterra system is shown in Figure 2. (A free demo version of Powersim

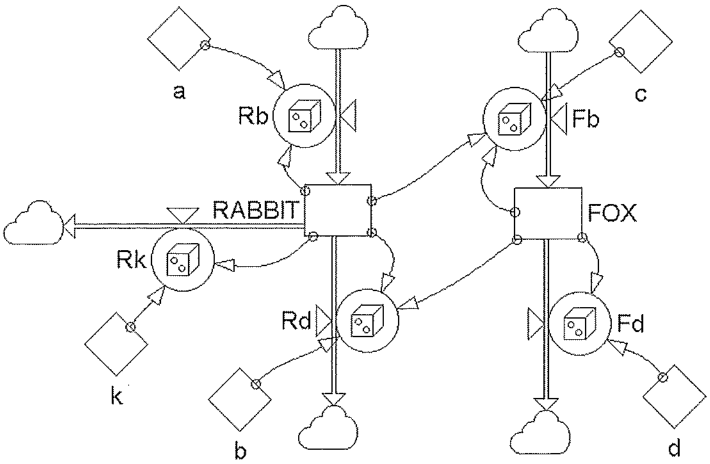

Figure 2. The Lotka-Volterra model in Powersim. The flow rates are implemented as: POISSON(TIMESTEP*q*Z)/TIMESTEP, where q*Z is a*X for rabbit births, b*X*Y for rabbit deaths, k*X*X for rabbit competition, c*X*Y for fox births and d*Y for rabbit deaths.

with which you can build this model is available from the Internet at *www.powersim.com* [6].) The model is also given in Pascal in Appendix B.

Initializing this simulation in steady state with X(0) = 60 and Y(0) = 28 may give the results shown in the top portion of Figure 3 (one experiment).

In Poisson simulation the inherent dynamic of the system is excited by the stochastic fluctuations. We also see that this dynamic not only causes the numbers of rabbits and foxes to vary, but it also displays the periodical pattern and its typical period length. Also note that in some simulations all foxes will starve to death, making the system vary around the second stationary solution. In the simulation shown in Figure 3, this happens at about time 750, making the rabbits increase and fluctuate around the steady state X = a/k = 200.

Note that a deterministic simulation would here only give two horizontal lines for X and Y. Even if we have disturbed the system to generate variations, these would die out as shown at the bottom of Figure 3.

If we also had added output noise to the deterministic model, it would only have given a superimposed noise without exciting the dynamics. Adding state noise of some distribution (say Normal$(0,\sigma_i)$) to each of the five flows would cause several problems. First, it would give five constants σ_i to estimate; second, we have to normalize the variation for changing dt (otherwise the noise influence would decrease with decreased step size). Solving these problems gives a model that excites the dynamic so that it oscillates with the periodicity given by the dynamic. But the foxes may recover from zero or even negative values. There is also a substantial risk that a negative state value makes the model unstable and causes numerical overflow. But the results would, for example, never have revealed a switch to the mode X = a/k and Y = 0, which inevitably happens sooner or later in the real system. This is because the variance should vary in accordance with the state values and because we have no integer mechanism.

2.5 *Example 2: Estimation of Screening Effects*

In medicine and epidemiology a common design is to study the effects of an exposure (from, e.g., drug, vaccination, radiation, stress or screening) by comparing the outcomes (e.g., morbidity or mortality) of an exposed *study group* and an unexposed *control group*. A main problem is that the results have to be interpreted in statistical terms because of finite numbers of participants in the groups. In many cases, it is necessary also to regard the dynamics of the system.

One application is medical screening with a subsequent treatment of findings (a negative or protective

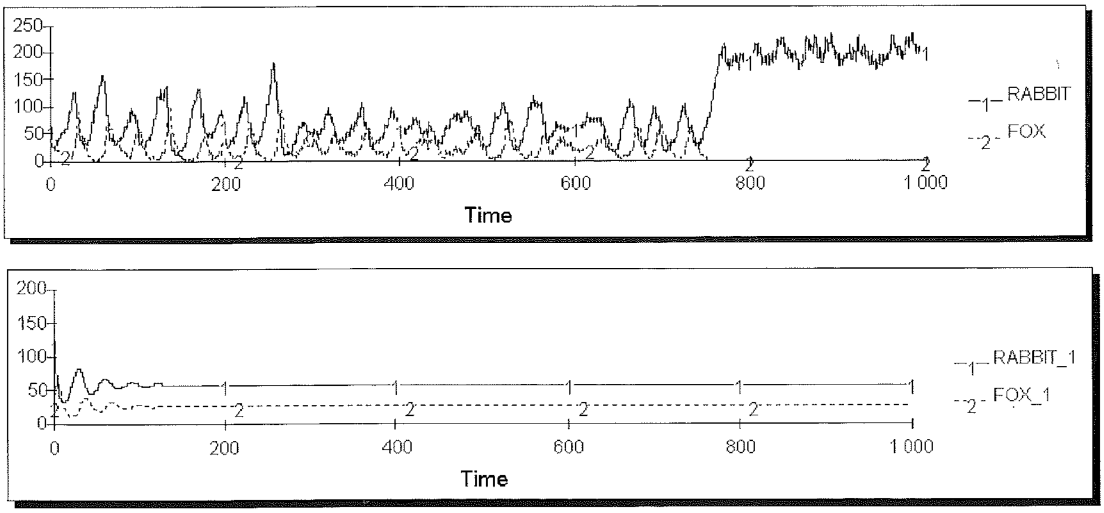

Figure 3. One simulation of the Lotka-Volterra system shown in Figure 2. At the top, the results for a Poisson simulation. Eliminating the Poisson mechanisms gives two horizontal lines. A corresponding deterministic simulation is given below, where X(0) has been set at 120 to disturb the system.

exposure) of a population to reduce future morbidity or mortality. When testing the effects of medical screening on a population, one is interested in the *sensitivity* of the test (which is the proportion of sick people who are *correctly* classified as sick by the test).

In practice, you usually have to compare the effects of screening by monitoring the number of clinical cases surfacing during a follow-up period *after* screening of the study group and comparing that number with the number of surfacing cases in an unscreened control group during the same time period (see Figure 4). (Adjustments then have to be made because all pre-clinical cases will not surface during a limited follow-up period, and some rapid cases, not in the pre-clinical stage at screening time, will surface during follow-up [7]. Here we omit these complications.)

Since the numbers of surfacing cases in the study group and in the control group during follow-up are stochastic variables, it is important to regard both dynamics and stochastic properties. This is the case when, e.g., optimizing the length of a follow-up time after screening (a short follow-up time gives few cases and a long one gives inclusion of fast cases not in pre-clinical stage at screening time), or when estimating the sensitivity of the screening test.

In this example, the purpose is to study the stochastic fluctuations in the accumulation of surfacing clinical cases under the following conditions:

A study group and a control group, each of 10,000 persons, are randomized from a population. The incidence rate of new pre-clinical cases is 150 cases per year and 10^5 persons (0.0015), yielding an expected annual number of 15 persons from each group to become pre-clinically sick (Poisson-distributed). The disease has a pre-clinical stage with an average duration, D, of 12 years. Therefore, we expect about 15*12 = 180 persons to be in the pre-clinical stage in each group when the study begins. This number should be random and different for the two groups (an easy way to do that is to start simulating the model a number of years before screening). But for the sake of simplicity we use the expected value as initial values in the model.

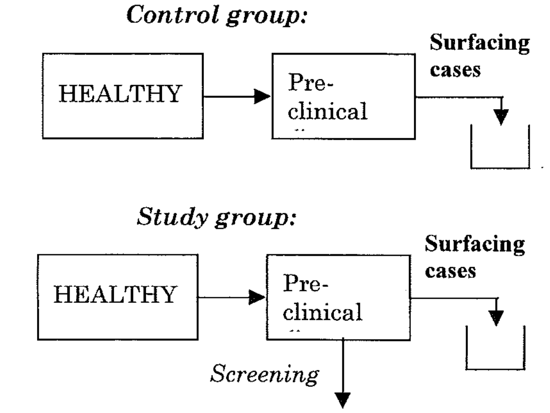

Figure 4. A screened study population and an unscreened control population. Screening affects people in both healthy and pre-clinical stage but here we are only interested in the latter because positive findings among these cases are removed by treatment.

When the study starts, screening with a test sensitivity of S = 0.75 takes place, momentarily reducing the prevalence in the study group by 75 percent.

If we want to model the case that the test, applied to each of the n persons in the pre-clinical stage, has a binary outcome of positive (p = S = 0.75) or negative (1 – p), then the number of positive tests will be

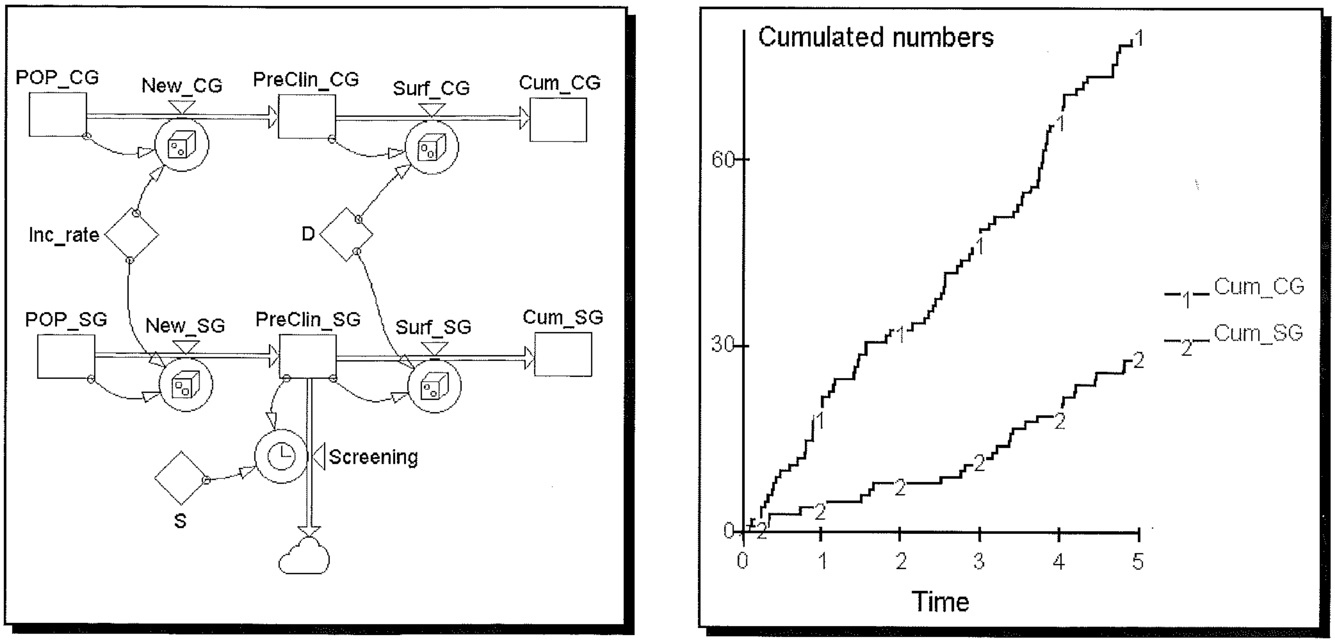

Figure 5. A Poisson model of the number of surfacing cases after screening of the Study Group (suffixed "_SG") and for the unscreened Control Group (suffixed "_CG"). In this figure screening is deterministic, but it may be modeled on a binomial distribution. To the right, the cumulated numbers of surfacing cases are shown during a follow-up time of five years.

binomially distributed as Bi(n, p). For large numbers (say n > 10), this can be approximated by the Normal distribution N(np, Sqrt(np(1 – p))). A Normal distribution is implemented, which is even simpler than what has been shown for a Poisson distribution:

PreClinicalStage: x: = x + dt*(–f) (just the screening reduction outflow is shown)

ScreeningReduction: f: = N(np, Sqrt (np(1–p)))*Pulse(1, ScrTime, 9999)

The Pulse function opens the outflow (with size = 1) only once at time = ScrTime when a random sample is taken from the Normal distribution. (The last parameter in the Pulse function is the interval between pulses, which is just given a large value). As opposed to the Poisson case, no time step, dt, is involved in the Normal distribution formula.

We build a Poisson simulation model in Powersim corresponding to that in Figure 4. For the sake of simplicity we use a first-order delay representing the pre-clinical stage. See Figure 5.

Repeated simulations give the variances or confidence intervals for the outcomes of this study. The results can also be used to obtain an optimal follow-up period. (A short period gives large stochastic variations and a long follow-up period harms the results by including many cases not in the pre-clinical stage at screening time.)

2.6 *Example 3: Genetic Modeling*

One field for Poisson simulation is when genetic effects are studied in a dynamic context. It may, for example, be necessary to model males and females separately. Such a model can be designed in different ways. Let us start from a simple dynamic model like that in Figure 6. The annual number of male and female offspring is here assumed to be proportional to the number of females in reproductive ages and to the expected proportions of births among the sexes. In this example, *inflows* of female births (Bx) and male births (By) become stochastic by using the Poisson mechanism: Bx = Po(dt*kx*X)/dt and By = Po(dt*ky*X)/dt, where X is the number of reproductive females and kx and ky are proportionality constants for the fertility in terms of female and male offspring. (If expected numbers of females and males are the same, only one constant, k, is needed).

In addition to sex, we may include genes (alleles) of types "a" and "A" with some genetically important property for survival or reproduction. The females then have the combinations Xaa, XaA (including XAa) and XAA, and the males Yaa, YaA and YAA. All combinations of random or nonrandom mating are possible. Figure 7 shows part of a dynamic and genetic model where only the reproduction from mating between XaA and YaA is included.

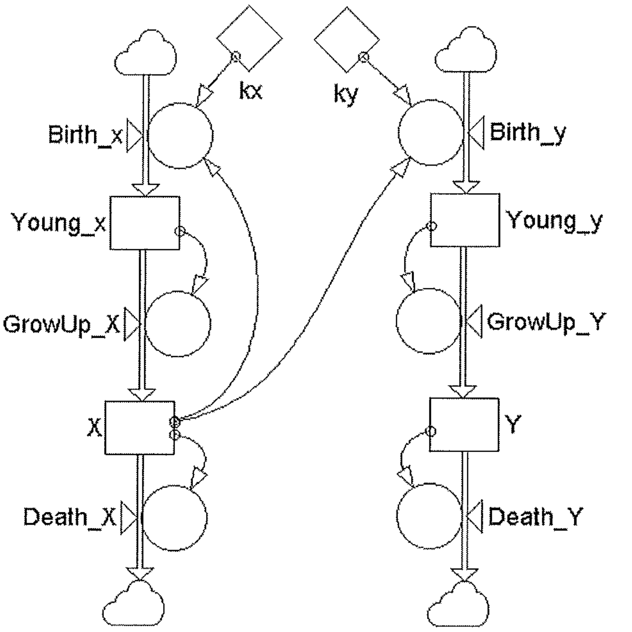

Figure 6. Female and male offspring in a dynamic context: Young_x and Young_y represent young females and males, and X and Y represent females and males in reproductive ages.

Genetic models like these can be used to study the development of a population in a dynamic context where sex or genes have to be considered. For example, sex and genetic factors may be included to study the risk of extinction of the population or the risk of depletion of an important gene.

2.7 *Technicalities*

(1) *Poisson Distribution or Not?*

When modeling a certain stochastic process, the question, if it is a Poisson process, is important. This requires that the events happen one at a time and independently. In the best situation, we know this from our knowledge of the process (e.g., radioactive decay), or we may know the system from statistical data. In this second case, we may have records of individual events or we only know the cumulated number of events during a time period (e.g., annual incidence or mortality rate for a disease).

A first view can be obtained from the figures presented in a histogram and compared with a Poisson distribution. But a more powerful method is to use statistical tests. There are a number of tests, e.g., the Chi-square test and the Kolmogorov-Smirnov test, that can be used to compare statistical data with a certain distribution. When applying this to the Poisson distribution, you usually use the fact that "the events should be randomly spaced" and compare it with a uniform U(0,T) test where T is the entire period studied, or you sort the inter-event times and compare them with a negative exponential distribution. See also [1, 2, 8]. Note, however, that theoretically it is not enough to test if the inter-event times are exponentially

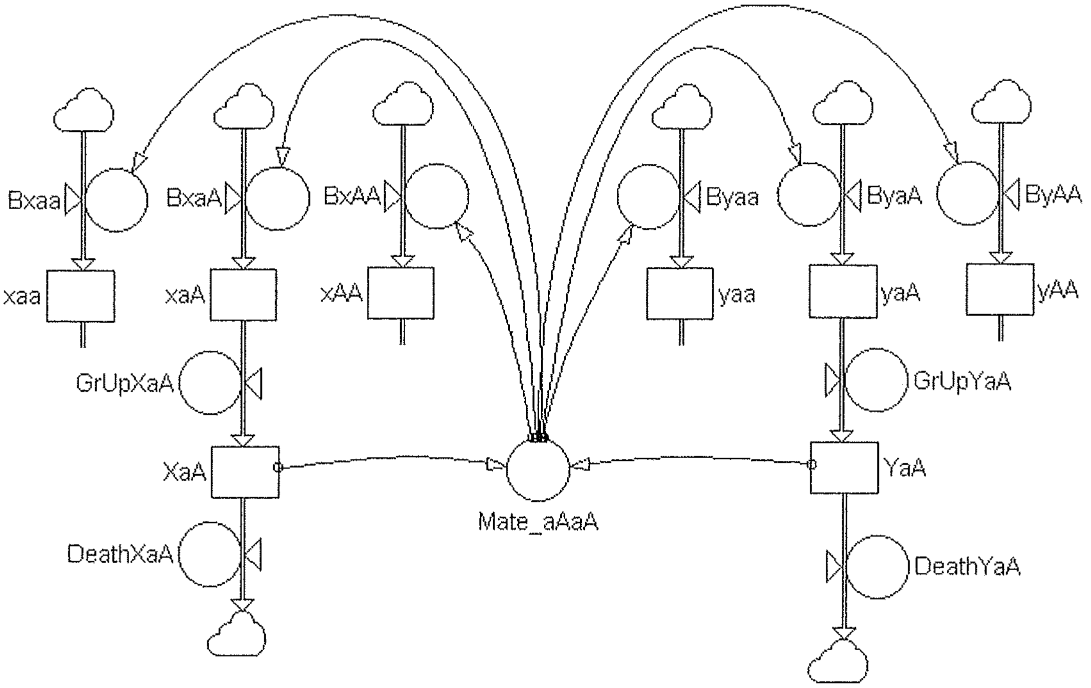

Figure 7. Part of a genetic model in a dynamic context. Females are denoted by x and males by y. The genes aa, aA and AA may be randomized in proportions 1:2:1.

distributed. You also have to check whether the events are independent. In the case of an infectious disease, for example, dependency is probably stronger than for a non-infectious one like cancer.

You should also remember that the more data you have, the more likely it is that your test is able to reject your hypothesis that the studied distribution is of a certain kind—and that is not exactly what we want.

An open but important question is the degree of strictness to apply when judging if data are likely to come from a certain distribution. In modeling, a "true" model cannot be built, but the aim is to construct a model that, for a given purpose, behaves as closely as possible to the real system. The question is then whether you do better with a deterministic flow or a flow generated from a Poisson or from any other statistical distribution.

(2) Interpretation of Flow Rates

In Poisson simulation, you can study the stochastic variations of the states. The flow rate (Po(dt*λ)/dt) has a correct and well-defined average equal to λ, but the events happen momentarily. Therefore, when plotting a flow rate (for every dt), the smaller the time step, the larger the variations around the average. Using a very small time step, say dt = 0.001, means that the probability of an event during dt becomes very small. But when it occurs (one event during a dt), it gives a flow rate of 1/dt = 1000. This is correct because the state will be updated by dt*1000 = 1 for that dt. (Mathematically, the flow rate is infinite when an event happens, and otherwise zero.) Thus, one should be careful when interpreting flow rates.

(3) Random Number Algorithms for Poisson Distribution

Poisson distributed random numbers can be generated from uniformly distributed numbers by using the inverse transform method [8] or the rejection method [1, 9]. (A random number generator for uniformly distributed numbers is a part of every programming or simulation language.) Code for a Poisson generator using the inversion method is given in Appendix A.

Both the inverse transform as well as the rejection algorithms for Poisson-distributed random numbers become slower as the Poisson parameter (here θ = dt*λ) increases. For large numbers (say larger than 15), the Poisson distribution approximates the Normal distribution that can be generated by the inversion method. Roundoff could then be used so that the numbers continue to be integers. A compound rejection algorithm that uses the Gamma distribution for large parameter values is given in [9].

(4) Integration Step Size

What happens when the step size is changed, for example, from dt to DT = k*dt? Then the flow during DT becomes Po(DT*λ) = Po(k*dt*λ). If the flow rate is rather constant over DT, we see that the expected number is increased by a factor k because the interval is k times as long. But the flow rate becomes Po(DT*λ)/DT = Po(k*dt*λ)/(k*dt). The expected flow rate therefore becomes the same. Also the variance is properly taken care of by the Poisson distribution. We are thus free to change the step size to what is reasonable from the dynamic point of view.

In Example 1 (Section 2.4), 50 simulations with the step size dt = 1, 0.1 and 0.01, respectively, were tested. For dt = 1, the dynamic was too coarse, making the extinction of foxes (and even rabbits) occur too soon. (For dt = 0.1 and 0.01, no particular difference was seen. A step size, dt = 0.1, gives a θ in the order dt*a*X = 0.1*0.2*60 = 1.2 for the largest flow of rabbits, and dt*d*28 = 0.1*0.3*28 ≈ 0.8 for the largest flow of foxes.) But note that the requirement of a small step size comes from the integration of the dynamics and not from the stochastics. (However, see also under Item 8, below, that a smaller step size can handle a specific stochastic problem.)

(5) Integration Method

When stochastic variations are included, the gain of using more sophisticated integration algorithms is more limited because there is no well defined exact solution for the next time step.

A multi-step or a Runge-Kutta integration algorithm evaluates the derivatives (flow rates) at a number of points of time for each time step to decide the next value of the state. If used for a Poisson simulation model, each evaluation of a flow rate will be stochastic. Probably the integration should instead perform deterministically to estimate the expected value of the derivative and thereafter use the Poisson generator once. This is theoretically more sound and also speeds up the process since unnecessary calls to the Poisson generator are eliminated.

Testing the behavior of different unmodified Runge-Kutta algorithms in Example 1, above, in Powersim did not reveal clear advantages or disadvantages compared with using Euler's method regarding accuracy of estimates, stability (when k = 0) or variance. But this subject has to be further investigated.

(6) Several Outflows

We have confirmed above that: if $X \in Po(\theta_1)$ and $Y \in Po(\theta_2)$, where X and Y are *independent*, then $X + Y \in Po(\theta_1 + \theta_2)$. This means that a number of Poisson distributed outflows with expected values $\theta_1, \theta_2, ... \theta_n$ may be represented by one total flow described by $Po(\theta_1 + \theta_2 + ... + \theta_n)$—just as in the deterministic case.

Technically, any statistical distribution can be used to control the flow rates. For example, if a compound outflow rate f = Po(dt*k*X)/dt is to be divided into three separate outflow rates, fa, fb and fc in, say, proportions 1:2:3, then there is a multinomial distribution of the Poisson-distributed compound outflow. But it is easier to let fa = Po(dt*k/6*X)/dt, fb = Po(dt*2k/6*X)/dt and fc = Po(dt*3k/6*X)/dt.

Warning: Never replace an in-and an outflow to a state with a net flow when using the Poisson mechanism. Even when $Po(\theta_1)$ represents an inflow and $Po(\theta_2)$ an outflow, you may not use $Po(\theta_1-\theta_2)$ to represent a net flow. First, the variance would be too small because both the in-and the outflow add to the variance. Second, if θ_1 becomes less than θ_2, indicating a reversed net flow direction, $Po(\theta_1-\theta_2)$ will be zero since a probability is always non-negative.

(7) External Inflows

Each internal flow is generated as the output of a state. But there may also be external inflows to a model. In Example 2, the flows of new pre-clinical cases are generated from the study and control groups. But we could instead have taken them from real statistics (if observable), or we could have used the average incidence. Then there is no need to model the healthy states giving new pre-clinical cases as outflows. An external inflow is then controlled by statistical data and can be realized by an ordinary table look-up function. If we use the average incidence, the inflow is acquired by taking Poisson samples of the given average.

The genetic model in Example 3 shows a case where Poisson distributed external inflows are controlled from within the model.

(8) Negative State Values

In deterministic CSS, the state variables are defined on the whole real axis. However, often a state represents a non-negative quantity, such as the number of animals or the amount of water. In most cases, the structure of the problem guarantees that these states will remain non-negative without any precautions. But there are certainly exceptions to this. For example, if there is a hole in the bottom of a bucket of water, there will be an outflow that is proportional to the square root of the height of water in the bucket ($dx/dt = -k*\sqrt{x}$). This means that, independently of the step size and of k, one will come to a point (small value) where the outflow during the next time step ($k\sqrt{x}$) is larger than the remaining amount of water (x). This makes the content, x, negative (and in the next time step the program will halt because it takes the square root of a negative number). In such situations, one has to take some precaution, using, e.g., "if" or "max" statements.

In Poisson simulation we have an additional problem. When an outflow f is a random and unlimited quantity there is always a theoretical possibility that it becomes larger than x, giving a negative state value. Since the Poisson distribution function has a faculty term in the denominator, this risk is often negligible except for small values of x (compared with f). Therefore, f may be restricted to less than or equal to x, just as is sometimes necessary in ordinary CSS simulation. (Such a device can also be built into the Poisson random number generator, but it then has to consider all outflows during the time step.)

Another way is to reduce the step size, which implies that virtually all Poisson calls will return zero or one as outcomes. Since we work with integer units, this will then end up at exactly zero—a phenomenon that normally will not occur in CSS. This is not a problem, but an advantage. It means, e.g., that the model correctly describes the extinction of a species rather than letting it recover from, say, 10^{-10} as in CSS.

(9) Modeling a Nonstationary Poisson Process

In DES, a Poisson process is modeled by using a negative exponential distribution to control the time between events. Although this is a mathematically and statistically correct method, it causes problems when the intensity, λ, varies over time. Assume that the intensity, λ, is very small, giving (randomly) a very large time interval to the next event. But long before that, the intensity might have become large. In DES, special methods, e.g., thinning, have to be used [1, 2].

In Poisson simulation there is no problem related to a variable intensity, $\lambda = \lambda(t)$. In fact, λ can have a new value for every time step. For example, the fertility rate constants, a and c in Example 1, may vary with the time of the year. This could be easily implemented by, e.g., letting a and c be table look-up functions. In Example 2, the flow of new pre-clinical cases and the average time in the pre-clinical stage are functions of age (and therefore also time), which has to be taken into account if the follow-up time is long.

(10) Correlated Variables

So far we have assumed that the flows have been uncorrelated. In Example 1, however, there is a possible number of encounters between rabbits and foxes (X*Y). Some of these are realized and result in the death of rabbits and the breeding of foxes. For given values of X, Y, b and c, it is reasonable to assume a positive correlation between rabbit deaths and fox births. Since both b and c have the same value (0.005), in this example we could use one common sample for rabbit deaths and fox births (Po(dt*0.05*X*Y)/dt), implying complete correlation (which is not probable).

Warning: Sampling Po(dt*X*Y)/dt and then multiplying by b or c gives correct average values but too small variations, since b and c are less than unity. In this context it is important to realize that $Po(\theta_1 + \theta_2) = Po(\theta_1) + Po(\theta_2)$ if and only if $X \in Po(\theta_1)$ and $Y \in Po(\theta_2)$ are *independent*. Remember that $Var(X + Y) = Var(X) + Var(Y) + 2Cov(X,Y)$, and it is independence that makes the covariance disappear. But, e.g., $Var(X + X) = Var(2X) = 2^2 Var(X)$ because X is fully correlated to itself.

Using the property of addition for Poisson-distributed stochastic variables (random numbers) described in Section 2.2, we can write: $X_1 = Y_1 + Y_{12}$ and $X_2 = Y_2 + Y_{12}$ where Y_1, Y_2 and Y_{12} are *mutually independent* Poisson random variables with averages θ_1, θ_2 and θ_{12}, respectively. Since Y_{12} is common for X_1 and X_2, and Y_1 and Y_2 are independent, one can control the correlation between X_1 and X_2 by separating the Poisson parameters into a common (correlated) and an independent (uncorrelated) part. Antithetic sampling from the uniform distribution may be used to produce negatively correlated Poisson random numbers. Letting $X_1 = Y_1 + Y_{12}$ and $X_2 = Y_2 + A(Y_{12})$, where $A(Y_{12})$ stands for an antithetic sample, gives a negative correlation. A more detailed discussion of multivariate Poisson distributions can be found in Chapter 37 of [10].

(11) Repeated Simulations to Estimate Statistical Properties

Since randomness implies random output, several simulations have to be performed to study the stochastic variations—just as in DES or Monte Carlo simulation. (Therefore, the model should be embedded in, or repeatedly called from, a superior procedure which includes statistical calculations and perhaps variance reducing capabilities.)

3. Discussion

In this paper, we have used the term Poisson simulation for the method presented to extend Continuous System Simulation. The fact that the Poisson distribution is discrete should not be a contradiction because "Continuous" in CSS stands for continuous time as opposed to "Discrete Event" in DES.

In biology, medicine, epidemiology, ecology, genetics, agricultural sciences, population dynamics, sociological studies, etc., there are many models based on the solid concepts of differential equations but where stochastic variations, because of finite numbers, should be modeled in a theoretically correct way rather than added as noise of some distribution. This makes the randomness interact with the dynamics in a correct way. Poisson simulation may be a proper solution for many such problems. The results can then easily be compared with deterministic simulations in CSS. It also has the advantage that hidden modes, like extinction of the foxes in Example 1, may be revealed.

Compared with the alternative of using DES, the approach presented is often simpler and computationally more efficient. But especially, its foundations in differential equations make it closer to a theoretical analysis and easier to comprehend. Yet another advantage, compared with DES simulation, is that the intensity, λ, may vary in time without additional precautions.

Randomness is used when we lack exact information. It may be of many kinds. It should also be stressed that the Poisson mechanism should not be used for all kinds of randomness—but only for variations because of numbers. For example, a model of an agricultural system might also be affected by the weather, which can be described by statistical data or generated according to some random distribution. The important thing is that the randomness enters at the right place in the model to excite the dynamic modes in a correct way. (In some cases the stochastic variations may be smoothed by the model structure and thus become of less importance.)

It should also be noted that deterministic and stochastic descriptions in the flow rate equations can be mixed freely.

To use Poisson simulation efficiently, a special package should be developed, including a fast Poisson generator, multiple run facility, a device for stochastic estimates and confidence intervals from several runs, and facilities for variance reduction with common random numbers and antithetic sampling. To make variance reduction efficient there is also a need for several separate generators (as in DES but in contrast to most CSS languages).

4. Acknowledgment

The author wishes to acknowledge Kjell Pernestål and Mikael Sternad for valuable discussions and proposals. This study was partly supported through a grant from the Swedish Cancer Society.

5. References

[1] Law, A.M. and Kelton, W.D. *Simulation Modeling and Analysis.* McGraw-Hill, New York, 1991.

[2] Bratley, P., Fox, B.L. and Schrage, L.E. *A Guide to Simulation.* Springer-Verlag, New York, 1983.

[3] Åström, K.J. and Wittenmark, B. *Computer Controlled Systems,* Third Edition. Prentice-Hall, Englewood Cliffs, NJ, 1997.

[4] Luenberger, D.G. *Introduction to Dynamic Systems. Theory, Models and Applications.* John Wiley & Sons, New York, 1979.

[5] Volterra, V. "Variazioni e fluttuazioni del numero d'individui in specie animali conviventi." *Memoire della R. Accademia Nazionale dei Lincei, anno CCCCXXIII,* II , pp 1-110, 1926.

[6] *Powersim© Reference Manual,* Powersim Press 1996. Powersim Corporation, 1175 Herndon Parkway, Suite 600, Herndon, VA, 22170. (A free demo is found at: *http://www.powersim.com.)*

[7] Day, N.E. "Estimating the Sensitivity of a Screening Test." *Journal of Epidemiological Community Health,* Vol. 39, No. 4, pp 364-366, 1985.

[8] Fishman, G.S. *Principles of Discrete Event Simulation.* John Wiley & Sons, New York, 1978.

[9] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. *Numerical Recipes in Pascal—The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, 1989. (Also available in FORTRAN or C.)

[10] Johnson, N.L., Kotz, S. and Balakrishnan, N. *Discrete Multivariate Distributions.* Wiley, New York, 1997.

Leif Gustafsson earned his PhD in Automatic Control at Uppsala University, Sweden, in 1977. He is now a Professor in Systems Analysis at the Swedish University of Agricultural Sciences. He is also affiliated with the Department of Medical Epidemiology at the Karolinska Institute in Stockholm, where he is developing dynamical methods and models in the field of cancer epidemiology. His main interest areas are in the fields of systems analysis, modeling, simulation and optimization.

Appendix A. A Poisson Random Number Algorithm

An algorithm for Poisson-distributed random numbers based on the inverse transform method [8] may, in Pascal, look like the code below. (Use double precision.)

```
Function POISSON(theta: real): integer;
var X: integer;
    A,B,U,W: real;
Label L;
begin
    W:=Exp(-theta);
    X:=0;
    A:=W;
    B:=A;
    U:=Random;
L:  If U>A then
    begin
        X:=X+1;
        B:=B*theta/X;
        A:=A+B;
        Goto L;
    end;
    Poisson:=X;
end; (* Poisson *)
```

Random is a call to the built-in uniform, U(0,1), generator in Pascal.

Appendix B. A Lotka-Volterra Model for Poisson Simulation in Pascal

A simple Lotka-Volterra model with Euler's integration and Poisson stochastics is given in Pascal code below. (Just include the Poisson generator in Appendix A.)

```
Program Volterra_Poisson;
type Real=double;                        (* Gives double precision. *)

var dt,time,Tstart,Tend: real;
    RABBIT,FOX,Rbirth,Rdeath,Rcomp,Fbirth,Fdeath: real;
    a,b,c,d,comp: real;

Function POISSON(theta: real): integer;(* As given in Appendix A above. *)

Procedure SIMULATE;
Label START, AGAIN;
begin
    Randomize;(* Randomizes the seed so each simulation is a new experiment*)
    dt:=0.5; Time:=0; Tstart:=0; Tend:=1000;
    a:=0.2; b:=0.005; c:=0.005; d:=0.3; comp:=0.001;
    RABBIT:=d/c;                        (* Starting in dynamic equilibrium. *)
    FOX:=(a-comp*d/c)/b;                (*  -"-                             *)
    Writeln; Writeln('  Time   RABBIT    FOX');
    Goto START;

AGAIN:
    RABBIT :=RABBIT+dt*(Rbirth-Rdeath-Rcomp);
    FOX    :=FOX+dt*(Fbirth-Fdeath);

START:
    Rbirth :=Poisson(dt*a*RABBIT)/dt;
    Rdeath :=Poisson(dt*b*RABBIT*FOX)/dt;
    Rcomp  :=Poisson(dt*comp*RABBIT*RABBIT)/dt;
    Fbirth :=Poisson(dt*c*RABBIT*FOX)/dt;
    Fdeath :=Poisson(dt*d*FOX)/dt;

    Writeln(time:7:2,RABBIT:8:2,FOX:8:2);

    Time :=Time+dt;
    if Time <= Tend then Goto AGAIN;
end; (* SIMULATE *)

begin
  SIMULATE;    (* The main program *)
end.
```