Poisson Simulation outperforms Markov Simulation

Leif Gustafsson [a,b,*]

[a] Signals and Systems, Dept. of Engineering and Sciences, Uppsala University, P.O. Box 534, S-751 21 Uppsala, Sweden.
[b] Associated with Dept. of Medical Epidemiology and Biostatistics, Karolinska Institutet, S-171 77 Stockholm, Sweden.
[*] Corresponding author, Tel.: +46 18 500061.
E-mail address: Leif.Gustafsson@bt.slu.se

Abstract

Markov Simulation and the more recent Poisson Simulation are two fully consistent ways of modelling, applicable to the same types of problems.

A Markov model is based on a detailed description of every situation, called state, that a system can be in and every possible transition between these states. This approach allows powerful analysis and makes Markov models a fundamental part of the theory of stochastic processes. However, Markov models are also frequently used for simulation. Here the detailed description gives huge, inflexible models, slow execution and problems with model fitting and validation because the natural parameters are spread out as parts of the transition probabilities. These problems are effectively eliminated in Poisson Simulation, where the model is based on a small number of state variables (compartments) and where each parameter is explicitly represented only once.

This paper first presents Markov Simulation and Poisson Simulation and demonstrates how these fully consistent methods are related. Then a systematic scrutiny of the merits and demerits of using Markov Simulation and Poisson Simulation in a modelling project reveals that Poisson Simulation is superior to Markov Simulation in every phase of the project and can handle considerably larger and more complex models. The power of Poisson Simulation also extends far outside the field of Markov models.

A number of illustrative examples are included to demonstrate the differences and advantages of using Poisson Simulation instead of Markov Simulation.

Keywords: Markov Simulation, Poisson Simulation, state-based modelling, stochastic compartment modelling, stochastic process, tau-leap simulation

1. Introduction

Markov models are a fundamental part of the theory of stochastic processes [1-4] and are also frequently used for simulation [4-8 and references therein]. This paper focuses only on the latter aspect. Poisson Simulation [9], sometimes denoted tau-leap simulation [10], is a more recent method for model building and simulation that facilitates construction, model fitting, execution, validation, modification and analysis of a stochastic and dynamic model.

Both Markov Simulation and Poisson Simulation models are based on the Poisson process and are applicable to the same types of problems. Both model types generate discrete-valued stochastic processes in continuous or 'almost continuous' time. The two approaches are fully consistent and produce consistent results.

However, Markov Simulation is based on a detailed mathematical description where *every possible situation*, called *state*, the system can take and *every possible transition* between the

states are explicitly represented. This makes a Markov model huge, except for very simple cases. Poisson Simulation, on the other hand, is developed in the opposite direction. Here the state space is *aggregated* into a small number of *state variables* (*compartments*) that can hold any number of entities. In addition, the transitions are aggregated over time into *flows* that can transfer many entities per time-step.

Furthermore, Poisson Simulation is designed so that natural parameters (such as risk, fertility, interest rate, sojourn time, etc.) of a system under study are explicitly represented once for each parameter instead of being implicitly dispersed into the transition probability elements. The aggregations over state space and time give a transparent, comprehensive, flexible and rapidly executable model, and the explicit representation of the model parameters simplifies model fitting, validation, sensitivity analysis and optimisation.

The main purpose of this paper is to identify and compare the merits and demerits of Markov Simulation and Poisson Simulation as they appear during each phase of a modelling and simulation project (from Problem recognition and Problem definition via Model building, Validation and Analysis to Result evaluation and Result presentation) [11-14]. In particular, issues about transparency, model size, model fitting, risk of over-parameterisation, flexibility to modify the model and execution time are examined.

Before performing this comparison, it is necessary to first present the Markov and Poisson Simulation models and to demonstrate the structural relationship between a Markov model and a Poisson Simulation model, so it becomes clear why these types of models produce consistent results.

This study is restricted to simulation of stochastic processes with discrete states in continuous-time. To avoid making the presentation larger and more detailed than necessary, no distinction is made between the *continuous-time* of the system under study and the '*almost continuous-time*' using sufficiently small time-steps in the numerical models. Therefore, both the denotations $x(t)$ and x_t are used when appropriate. Here discrete-time Markov models are often treated because they produce the same results as continuous-time Markov models when the time-step is sufficiently small. Jump Markov processes, where time jumps from one event to the next event, are also discussed. Furthermore, a Poisson Simulation model is sometimes presented in a compact differential form but implemented numerically by difference equations.[1]

The paper is organised as follows. Section 2 provides a brief introduction to how models can be represented in different ways. Section 3 describes the Markov model and its state concept, and presents possible ways to design a Markov model according to different time handling methods. At the end of Section 3, a way of separating the Markov transition matrix into two parts – one for increments and one for decrements – is presented. The purpose of this construction is to demonstrate a very simplified updating scheme that is used in Poisson Simulation. In Section 4, Poisson Simulation is introduced. Sections 5 and 6 are the central parts of this paper. Section 5 compares the merits and demerits of Markov Simulation and Poisson Simulation from the perspective of performing every phase of a full modelling and simulation project, while Section 6 shows that Poisson Simulation can handle stochastic processes far outside the field of Markov models. Finally, a discussion and summary of conclusions are presented in Section 7.

[1] As a consequence of using differential equations the exponential distribution is dealt with, but for a difference equation the corresponding discrete distribution is geometric, although it approaches the exponential distribution for sufficiently small time-steps. So, for the sake of simplicity, this paper refers to exponential distributions also in the case of models with almost continuous-time.

2. Representation of a stochastic model based on compartments or states

When objectives and boundaries of what to include and exclude are defined for a *system under study* we have a *conceptual model*. The next step is to select a proper *representation* for the realisation of an executable simulation model [15].

The conceptual model can be implemented as a simulation model in three generic ways based on what the basic component of the model *represents*. These basic components can be *compartments*[2], *states* or *entities*. This leads to fundamentally different structures of *Compartment-based models*, *State-based models or Entity-based models*. The choice of representation also has a profound impact on the concepts, mechanisms, size, flexibility, etc. of the model and the study (project) where the model is a central part.

To grasp the idea of different representations and their consequences, Figure 1 illustrates a *conceptual model* where boundaries of what should be included or excluded in the model are defined so that the objectives of the study can be fulfilled. This figure is based on a simple example in the form of an epidemic SIR model for a population of *1000* individuals, where each individual is in one of the *stages* Susceptible, Infectious or Recovered (meaning immune). However, a conceptual model cannot be executed.

The next step is to decide the *representation* of an executable simulation model [15]. It may be based on: a) compartments for the stages 'Susceptible', 'Infectious' and 'Recovered' holding the numbers of indistinguishable entities, or b) states describing all situations the *entire system* can be in. The third option of basing the model on *n=1000* entities (individuals); each with the attribute *stage* set to '*S*', '*I*' or '*R*', is not included in the figure since it is outside the scope of this paper, but it is briefly commented on below.

The choice of representation decides the main structure of the simulation model, which in principle is a device to update the representation over time.

[2] The term 'compartment' is here preferred to 'state variable' to avoid the word 'state' in two different meanings. 'State' in 'state variable' refers to the content of a single compartment while state=situation refers to the whole system, i.e. a state is a tuple of compartment values. The term 'state variable' referring to the content of a compartment is, however, a standard concept and is also used in the text.

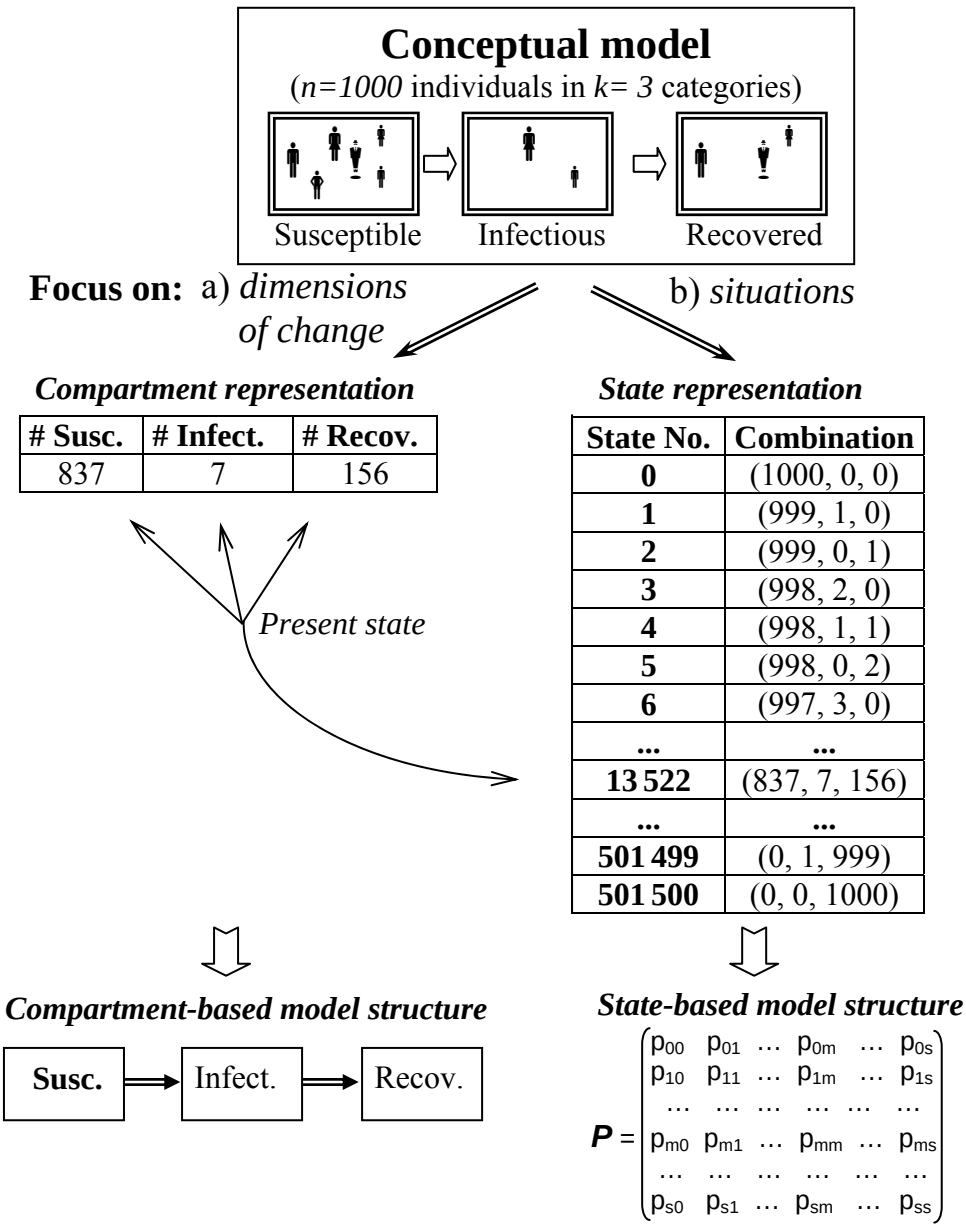


Figure 1. The conceptual SIR model with *1000* individuals can be represented by: **a)** Compartments or **b)** States as the fundamental components. The choice of representation has a profound impact on the structure of the simulation model.

C*ompartment-based models* include stochastic compartment models, stochastic differential or difference equation systems, etc. For compartment-based simulation models, the changes take place by flows into, between or out of the compartments that change their contents, illustrated by arrows in Figure 1. In the following the focus is on *Poisson Simulation,* which is presented in Section 4.

State-based models include Markov chain models, continuous-time Markov models, semi-Markov models, generalised semi-Markov process models, chain-binomial models, etc. For Markov models the state of the system is updated over time by conditional transition probabilities, usually ordered in the form of a square *transition matrix, illustrated by* ***P*** in Figure 1. Simulation by Markov models is described in Section 3.

Comment: *Entity-based models* include micro-Simulation, Discrete Event Simulation, Individual-based Modelling, Agent-based Modelling, etc. In this type of model the entities (with imbedded attributes) interact with each other and with the environment in accordance with logical and probabilistic rules, resulting in the attributes of the entities changing over time. Entity-based models can be implemented as: records with attribute fields (e.g. in an $n \times k$ matrix where the n rows represent the n entities and the columns their attributes), as objects or even as agents.

However, entity-based models are beyond the scope of this paper and are mentioned here only to clarify the difference between *entity-based* and *state-based* models. The misunderstanding sometimes arises that any model that can be designed in the form of a matrix can be called a state-based (or Markov) model.

For example, the conceptual epidemic model in Figure 1 can also be represented by $n=1000$ entities where each entity can take the attribute values: '*Susceptible*', '*Infectious*' or '*Recovered*'. This can e.g. be modelled and simulated using a 1000×3 matrix where the rows represent the individuals and one of the columns holds the current value ($=1$) of the attribute for each individual. Such a model is based on individual entities and is *not* a Markov model.

A key concept to stochastic processes is the *state space*. In Figure 1 the *501501* states comprise the state space representing every situation the system can be in. The conceptual model of Figure 1, represented by only three compartments, can also be in *501501* different situations represented by the combination (c_1, c_2, c_3), where c_i is the value of compartment i.

To understand the relationship and difference between a *compartment model* (e.g. a Poisson Simulation model) and a *state-based model* (e.g. a Markov model) is to realise that the same discrete-valued *state space* can be represented in *k dimensions* $(c_1, c_2, ..., c_k)$ or mapped onto a finite (or countable infinite) *set of states S*. Thus, the model can be based on either *k state variables* (compartments), one for each dimension, or on *states* (*k*-tuples of coordinate values $(c_1, c_2, ..., c_k)$). Since k compartments together can represent any state of the system, a compartment model is aggregated dimension-wise into a very compact form, while the state-based model must consider every possible transition between pairs of states and therefore often becomes huge.

When the model in Figure 1 is closed in the meaning that no entities can enter or leave the model ($c_1 + c_2 + c_3 = n$), the *degrees of freedom* becomes $d = k-1 = 2$ so the state space lies in an oblique plane in the space spanned by the dimensions c_1, c_2 and c_3. If entities could leave the model, the state space would be restricted by the oblique plane and the planes formed by the coordinate axes, so we would have $d = 3$ degrees of freedom. Furthermore, if entities could also enter the model, then the 3-dimensional state space would be unlimited.[3]

In a compartment model $k-1$ (or k if the total number of entities can vary) compartments are sufficient. The evolution then takes place because of flows into, out of or between compartments. When the system is realised as a state-based model, the evolution takes place by transitions between states. The evolution mechanism can then be assembled in an *$(s+1)\times(s+1)$ transition matrix*, $\mathbf{P}$, of transition probabilities p_{ij} (where the first index ($i=0,1,2,...s$) represents the *present* state and the second index ($j=0,1,2,...s$) represents the *next* state of the system).

[3] This last case causes no problems for a compartment model since a state variable can take any value. In a state-based model the set of states needed would, in principle, be unlimited. However, a given initial condition, upper growth rate limit and a limited time period for the process will set an upper limit for the required number of states.

For both the compartment and the state-based models, a specification of *the initial state* (c_{10}, c_{20}, c_{30}) is needed. In the compartment model of Figure 1, c_{10} is placed in '*Susceptible*', c_{20} in '*Infectious*' and c_{30} in '*Recovered*'. The compartments are then updated by the flows between the compartments. In the state-based model (Figure 1), a *state vector* $\boldsymbol{p}$ of size $s+1$ is initialised to an initial state by setting initial state (c_{10}, c_{20}, c_{30}) to one and the other states to zero. This state vector, $\boldsymbol{p}$, is then updated over time by the transition matrix $\boldsymbol{P}$. Since the model is stochastic, the elements of the state vector represent the absolute probabilities of being in the $s+1$ states.

By simulating corresponding stochastic compartment and state-based models a large number of times starting in the same initial state, the stochastic processes generated by the two kinds of models will be consistent [15].

3. Markov Simulation

3.1. Theoretical aspects of a Markov process

A *stochastic process* is a sequence of stochastic variables, separated by an index t and defined on a sample space Ω. The process is denoted by $\{X(t,\omega),\ t\in T,\ \omega\in\Omega\}$. If t is fixed, a stochastic variable is obtained, while if ω is fixed a replication/trajectory is obtained.

The classification of *stochastic processes* depends on three things: 1) the state space (discrete/continuous), 2) the index (time) parameter (discrete/continuous) and 3) the statistical dependence among the stochastic variables $X(t)$ for different values of the index parameter t. The complete *joint distribution function* among the stochastic variables must in some way be (explicitly or implicitly) specified.

A *Markov process* is a stochastic process that complies with the *Markov property*.

Definition: A discrete-valued stochastic process $\{X(t):\ t\in[0,\infty)\}$ is a Markov process if, for any $t_0 < t_1 < \ldots\ t_i < t_{i+1}$ and any discrete states $x_0, x_1,\ldots,x_i, ,x_{i+1}$,

$$Pr[X(t_{i+1})=x_{i+1} \mid X(t_i)=x_i,\ X(t_{i-1})=x_{i-1},\ldots,\ X(t_0)=x_0] = Pr[X(t_{i+1})=x_{i+1} \mid X(t_i)=x_i]. \quad (1)$$

Thus, if the process at present time (t_i) is in state x_i then the conditional probability of being in state x_{i+1} at t_{i+1} is independent of the past states $x_0, x_1, \ldots, x_{i-1}$.

An important part of the definition is that the process is defined in terms of *states* (rather than in terms of e.g. *state variables* as used in compartment modelling or in terms of *entities* as used in e.g. Discrete Event Simulation), and that *the state space* can be mapped onto a finite (or countable infinite) set of positive integers.

The Markov property (1) is often referred to as memorylessness. However, this is *not* a serious restriction because when a process is dependent on both the present and the past m states (the system has a memory of the past), i.e. when $Pr[X(t_{i+1}) = x_{i+1} \mid X(t_i)=x_i,\ X(t_{i-1}) = x_{i-1},\ldots,\ X(t_0)=x_0] = Pr[X(t_{i+1}) = x_{i+1} \mid X(t_i)=x_i,\ X(t_{i-1})=x_{i-1},\ldots,\ X(t_{i-m})=x_{i-m}]$, it is possible to reconstruct the process as a Markov process $\{Y_n\}$ from $\{X_n\}$ where $Y_n = (X_i, X_{i-1},\ldots, X_{i-m})$ is an

ordered $m+1$-tuple of X values. So this can be accomplished by extending the state space from $\boldsymbol{S}$ to $\boldsymbol{S}^{m+1}$.

Furthermore, the Markov property is *not* exclusive to Markov models. It is found when integrating from present time t_1 to future time t_2 in e.g. $x(t_2) = x(t_1) + \int f(x(s))ds$ where the past (before time t_1) is not included, or in a compartment model $dx(t)/dt=f(x,t)$ numerically updated by e.g. the Euler scheme: $x(t+\Delta t) = x(t) + \Delta t \cdot f(x(t))$.

A Markov model is *stationary* with respect to time if the transition probabilities p_{ij} are independent of time. We will let this be the case until Section 6.

Discrete and continuous-time modelling

When modelling a system, one can choose between a *continuous-time model* where an *event* may occur at any point in time t_i, as in equation (1), or a *discrete-time model* where the actual state may only change at equally spaced points $t=0, 1, 2, ...$ in time. In this latter case there is an implicit step-size involved, i.e. the time unit used.

In the discrete-time case time is advanced in time-steps of equal length. The step-size may then be so short (h) that zero or at most one event will happen during the time-step with almost certainty (*Bernoulli time handling*), or it may be relatively longer (Δt), allowing several events to happen, but not so long that the dynamics of the model are considerably changed during a time-step (*Poisson time handling*).

The *transition matrix* $\boldsymbol{P}$ of conditional transition probabilities p_{ij} gives the probability of going to state j at time t_j on condition that it is in state i at time t_i. The state vector $\boldsymbol{p}$ of absolute probabilities will then be updated by the transition matrix according to:

$$\boldsymbol{p}^{(1)} = \boldsymbol{p}^{(0)}\boldsymbol{P},$$

and by iteration over n time-steps by $\boldsymbol{p}^{(n)} = \boldsymbol{p}^{(0)}\boldsymbol{P}^{n}$. In the discrete-time case the time in a state is *geometrically distributed.*

Although the process is implemented in discrete-time, the time-step is chosen small enough to resemble the corresponding continuous-time process ('almost continuous-time'). When $h \to 0$ (or $\Delta t \to 0$) the discrete-time case approaches the continuous-time case and the geometrical distribution approaches the exponential distribution.

In the continuous-time case the model must handle any point in time, especially the times for the events $t=t_1, t_2, ...$. The transition matrix $\boldsymbol{P}$ then has to be adjusted to update the state vector to any point in time. This can be performed by the Kolmogorov (forward) differential equation $d\mathbf{P}/dt = \mathbf{PQ}$, where the intensity matrix $\mathbf{Q}$ is composed of *intensities* q_{ij} (measured as number of events per unit time for transitions from state i to state j). The Kolmogorov equation gives: $\mathbf{P}(t) = e^{\mathbf{Q}t}$. The $\mathbf{P}(t)$ matrix is then used to update the state vector $\boldsymbol{p}$ to time t:

$$\boldsymbol{p}(t) = \boldsymbol{p}(0)\boldsymbol{P}(t),$$

The use of the matrix exponential function is a consequence of the time in a state being exponentially distributed in the continuous-time case.

The theory of discrete-time and continuous-time Markov processes is fundamental for e.g. stochastic processes and queuing theory and the theoretical results are important in many sciences. The strength of the theory originates from the decomposition into states that are

expressed in statistical terms and analysed by matrix algebra. However, the mathematical aspects are beyond the scope of this paper – except when they are necessary for understanding simulation of Markov models.

When it comes to *simulation* of a Markov model, decomposition into states is no longer a necessary prerequisite, nor is it an asset. A stochastic compartment model, e.g. a Poisson Simulation model, is an option.

3.2. Single and multiple entity models

In textbooks, Markov theory is usually taught by trivial examples such as: '*a single entity* makes a choice conditional to a current situation'. For example the entity chooses the brand of a new car conditional on the brand of the current one. Hereby, the focus remains on the basic concepts of Markov theory. However, such single-entity models have a state-space consisting of two states per dimension (*0* and *1*) where the state of the entity is defined by the *k*-tuple (c_1, c_2, ...c_k) consisting of one $c_i=1$ and the other equal to zero. The number of states then becomes the same as the number of compartments in a compartment model. The compartment or state is occupied or not.

For this *single-entity case* there is little need for simulation, so we will focus on models with many (*n*) entities. We will sometimes refer to models with many entities as *population models*.[4]

However, there is a special case where the *n* entities are *non-interacting*. For example, when modelling the development of a non-contagious disease such as cancer, the decay of radioactive atoms or the risk of neutrons from a radioactive source penetrating a lead shield, the entities are non-interacting. In such cases a *single-entity* Markov model can be used to *repeatedly* simulate the cancer patients or neutrons one-by-one, resulting in *n* Markov simulations to perform one experiment. The results of the *n* simulations are then super-positioned to a common time axis. This case is briefly discussed in Section 5.3.1.

3.3. The ideas behind simulation of a Markov model

Note that the analysis sketched above *updates the complete state vector* constituting the probability distribution function (p.d.f.) of being in a certain state at time *t*. There are, however, a number of numerical considerations that make analysis so intractable and computer-intensive that simulation of the model is preferred – even though many replications are then needed to approximate the p.d.f. of the state vector.

In this paper we use the term *replication* for one simulation of a stochastic simulation model (fixed ω *for the replication of a stochastic process* $\{X(t,\omega),\ t\in T,\ \omega\in\Omega\}$) to generate a sequence of *states* over time.

---

[4] The terms '*entity*' or '*population*' are not part of the Markov property definition, which only refers to states. However, '*entity*' or '*population*' are underlying concepts of *the system at study* that after a modelling process are replaced by the more abstract term state in a Markov model.

When the Markov model describes *events* or *decisions*, in a real system at study these are related to some kind of entity. Here we use the term '*population*' which may consist of several or a single entity. Simulation is mainly about the non-trivial examples where there are many entities involved. The term '*population*' is then very useful to make the discussion more concrete.

If the simulation model is based on a transition matrix where every state and transition are explicitly represented, we talk about Markov simulation. If the model is based instead on individual entities or on the number of entities in compartments it will *not* be defined as a Markov simulation model, even though the model may be expressed as a matrix.

The Markov model can be designed and simulated in different ways depending on the *time handling* approach chosen. The choice of time handling has profound impacts on the form of the transition matrix, how simple it is to construct, the way to calculate or simulate the development over time and the speed of the execution. The three ways of handling time in simulation of a Markov model originate from the definition of the Poisson process and its properties.

The Poisson process

To understand the design of Markov models, it is necessary to discuss how time can be handled. The natural starting point for this discussion is to consider state changing events in a stationary process.

Definition: A *stationary Poisson process* is defined in the following way:
Let $X(t), t \geq 0$ be the number of times an event occurs in the time interval $(0, t)$. If the stochastic process $\{X(t), t \geq 0\}$ has the properties:

- The process has independent increments,
- $Pr[\text{an event occurs exactly once in the interval } (t, t+h)] = \lambda \cdot h + o(h)$,
- $Pr[\text{an event occurs more than once in the interval } (t, t+h)] = o(h)$,

then the process is a stationary Poisson process with intensity λ.

A *stationary* Poisson process (constant intensity λ) has a number of useful properties [16]:
Property 1: For a Poisson process with intensity λ the time between consecutive events is exponentially distributed with expected value $1/\lambda$.
Property 2: If $X \in Po(m_1)$ and $Y \in Po(m_2)$, where X and Y are independent processes, then the next event originates from the X process with probability $m_1/(m_1+m_2)$ and from the Y process with probability $m_2/(m_1+m_2)$.
Property 3: The number of events during a time interval (t_1, t_2) is *Poisson-distributed,* i.e. $X(t_1, t_2) \in Po(\lambda \cdot (t_2 - t_1))$. In particular, for a time-step of length Δt we have $X(t, t+\Delta t) \in Po(\lambda \cdot \Delta t)$.

In Section 3.4, *Bernoulli time handling* is presented. This approach is based directly on the definition of the Poisson process and gives a simple model that, however, is inefficient for simulation. In Section 3.5, *exponential time handling* is introduced. This approach is based on Properties 1 and 2 of the Poisson process above. Here the actual state is fixed between events, which allows a description where time *jumps* with irregular step-size from one event to the next. Since the time between events is exponentially distributed, we refer to this as exponential time handling. The transition matrix is then used to update the state at the time for each event occurring. In Section 3.6, *Poisson time handling* is introduced. The basis for this approach is Property 3 of the Poisson process above. Section 3.7 demonstrates how the transition elements in a Markov model are related to the terms of a Poisson distribution. This approach is extremely complicated but illustrative. It is included here only because it forms a bridge to Poisson Simulation.

3.4. Bernoulli time handling

According to the definition of the stationary Poisson process, the probability of an event occurring during a very short time interval is proportional to the length of the interval. When the time increment is a very small time-step (h), time is updated as: *time:=time+h*. A sequence of events can then be located by a sequence of points in time where single events that may happen are handled at the time increments.

In a numerical context *the Bernoulli distribution* with probability $\lambda \cdot h$ is used to decide whether an event will happen during $(t, t+h)$. For example, by introducing a stochastic variable X that accumulates the number of events, the second condition in the Poisson process definition can be reformulated in Markov terms as $Pr[X(t+h)=i+1|X(t)=i] = \lambda \cdot h$.

A good property of the Bernoulli approach is that the transition matrix is sparse compared with when many events may happen during the time-step. However, the Bernoulli method is a very inefficient way of handling time, because if several events occur during $(t, t+h)$ the model only captures one of them. Therefore, h has to be so short that it guarantees that more than one event almost never occurs during an interval $(t, t+h)$. This requires that $\lambda \cdot h << 1$. The very short time-step implies that the number of time-steps of a replication will be large and that the vast majority of them will be empty.

Example 1: *A birth-death model* [17]
Let us consider a 'birth-death' Markov model where the actual state can only be updated by *–1, 0* or *+1* entity per time-step. It is therefore sufficient to only consider transitions between 'neighbouring' states. In this example we let the *state space* of all possible states be limited to *0*, *1*, *2*… *s-1*, *s*. Directly applying the Poisson process definition results in the following Markov model with the 'from' states denoted in a column to the left and the 'to' states denoted above the transition matrix and where the matrix elements are transition probabilities:

	0	**1**	**2**	**…**	**s-1**	**s**
0	$1-p_{01}$	p_{01}	0	…	0	0
1	p_{10}	$1-p_{10}-p_{12}$	p_{12}	…	0	0
2	0	p_{21}	$1-p_{21}-p_{23}$	…	0	0
…	…	…	…	…	…	…
s-1	0	0	0	…	$1-p_{s-1s-2}-p_{s-1s}$	p_{s-1s}
s	0	0	0	…	p_{ss-1}	$1-p_{ss-1}$

The diagonal element of the state transition matrix will then be $1- \Sigma_{j \neq i} p_{ij}$ so that the rows add up to unity. ■

3.5. Exponential time handling

A continuous-time Markov process may change its state because of an event at any point in time $t=t_0, t_1, t_2, ...$ Between these event points the process remains in a state. The process can therefore be generated by answering the two questions: A) *When* will the next event occur? and B) *Where* (to which new state) will the process go? These two questions are answered, in probability terms, by Properties 1 and 2 of the Poisson process in Section 3.3 above.

Because this approach makes use of that the time intervals between events for a stationary Markov process are exponentially distributed, it is referred to as an exponential time handling approach. (Because the process jumps from event to event it is usually called a jump Markov process [2,4].)

Although the Markov process is in principle continuous-time, it can also be regarded as a sequence of events numbered: *0,1,2, ...* (detached from time) and treated by a discrete-time transition matrix $\boldsymbol{P}$. The timing of the events at $t=t_0, t_1, t_2, ...$ is then handled separately.

The size of the transition matrix is of course still $(s+1)\times(s+1)$ and the matrix is sparse since the transition probabilities p_{ij} are non-zero only for states one event apart.

The exponential time handling approach has a number of advantages.

- First, the matrix has a simpler structure than for the Bernoulli time handling approach.
- Second, since an event happens at the end of the interval, a '*new*' state is *always* reached, which significantly speeds up the simulation compared with using a short fixed time-step *h*.
- Third, a new state also means that the diagonal elements p_{ii} of the transition matrix, representing the probability of the system remaining in the state, are therefore equal to zero.[5]
- Fourth, when the diagonal elements are all zero (or one), all the elements of a row are proportional to the probabilities (or intensities) of the respective transitions. The transition probabilities of rows then only have to be scaled so that they add up to unity (without involving a time-step). This also makes the transition matrix easier to construct.
- Fifth, the problem of finding an appropriate step-size is eliminated.
- Sixth, the time of an event is exact rather than confined within a short interval.

A disadvantage is that the transition matrix only gives the *sequence* of transitions so the timing of the events has to be calculated separately [2].

Technically, the questions *when* and *where* does the next event occur are implemented in the following way.

When the system is in state *i* there may be several types of independent events taking the system to state *j1, j2, ...,jm* with intensities $\lambda_{ij1}, \lambda_{ij2}, ... , \lambda_{ijm}$. The total intensity of any event occurring is then $\lambda_i = \lambda_{ij1}+\lambda_{ij2}+...+ \lambda_{ijm}$. The next event will occur (according to Property 1 of the Poisson process in Section 3.3) at a time drawn from an exponential probability distribution function given by $\lambda_i exp(-\lambda_i t)$ where $t \geq 0$. Thus, by drawing a number from an exponential random number generator $Expo[1/\lambda_i]$, the next event is scheduled at *time := time* + $Expo[1/\lambda_i]$.

It now remains to be decided to which new state *j1, j2, ...,jm* the process will go. (This is answered by Property 2 of the Poisson process in Section 3.3.) Since the probabilities of the candidates are proportional to their intensities $\lambda_{ij1}, \lambda_{ij2}, ... , \lambda_{ijm}$ the respective transition probabilities will be $p_{ij} = p_{ij}/\Sigma p_{ij}$, where the sum is taken over *j1, j2, ...,jm*. We also see that the row of transition probabilities adds up to one. So by dividing a unit length interval into sections proportional to the transition probabilities p_{ij} of the *i*th row, a uniform generator for random numbers between zero and one can be used to decide where the process will go.

[5] Except for an absorbing state. Then, the corresponding diagonal element is equal to one, preventing any further events from happening.

Thus, each new event requires two calls to random number generators: One exponential to decide when the event will occur and one uniform (plus sorting logics) to decide the new state. The exponential Markov approach is best demonstrated by a simple example.

Example 2: *Ehrenfest's diffusion model* [18]
In two boxes, A and B, there are a total number of, say, *4* molecules (to make it very simple). The state of the system is defined as the number of molecules in box A. The expected sojourn time for each molecule in box A is $T_A=1/\alpha$ time units and the expected sojourn time in box B is $T_B=1/\beta$ time units (both exponentially distributed, with α and β the *intensities of a single molecule* leaving box A and box B, respectively). An event occurs when one of the molecules switches to the other box. Now we want to study the distribution of molecules in the boxes over time. Neglecting time, the following transition matrix is constructed.

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \alpha/(\alpha+3\beta) & 0 & 3\beta/(\alpha+3\beta) & 0 & 0 \\ 0 & 2\alpha/(2\alpha+2\beta) & 0 & 2\beta/(2\alpha+2\beta) & 0 \\ 0 & 0 & 3\alpha/(3\alpha+\beta) & 0 & \beta/(3\alpha+\beta) \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Here, the 'from-states', describing the number of molecules in box A, are shown to the left of the transition matrix, and the 'to-states' are shown above it. After scaling by dividing each intensity of a row by its row-sum, each row of transition probabilities adds up to unity. Note that the diagonal elements are all zero and that no time-step is involved in the matrix elements.

This model will correctly handle the sequence of events, but not the timing. If $\alpha < \beta$ and all particles are in box A, the expected time to the next event is longer than when they all are in box B. This means that the expected time intervals between events are not equally distributed among the states.

However, time can be calculated separately by drawing random numbers from the table below for the sojourn time of the state being visited. The arguments in this table are just the expected times to the next event. For example when the system is in state 1 with one molecule in box A and three in box B, the intensity of the molecule in box A is $\lambda_A=1/T_A=\alpha$ per time unit and the intensity of each of the three molecules in box B is $\lambda_B=1/T_B=\beta$ per time unit. The sum of these four intensities is therefore $\lambda=\lambda_A+3\lambda_B=\alpha+3\beta$ causing the expected time to the next event to be $Expo[1/\lambda] = Expo[1/(\alpha+3\beta)]$.

$$\begin{array}{l} \textbf{Time to next} \\ \textbf{event when} \\ \textbf{in state:} \end{array} \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} = \begin{pmatrix} Expo[1/4\beta] \\ Expo[1/(\alpha+3\beta)] \\ Expo[1/(2\alpha+2\beta)] \\ Expo[1/(3\alpha+\beta)] \\ Expo[1/4\alpha] \end{pmatrix}$$

To obtain the time the system spends in each state, the occurrence of a state (from the transition matrix) is weighted by the actual time it spends in the state (from the Time-to-next-event table). ■

This event-driven approach gives a simpler construction of the transition matrix, without the need to find an appropriate time-step, and is considerably more efficient for a numerical solution than the Bernoulli time handling approach. It is usually the most efficient approach for Markov simulation.

3.6. Poisson time handling

In the Poisson time handling approach time is advanced in time-steps of equal size, as it is in the Bernoulli approach. However, here the restriction of the time-step being so short that at most one event may happen is relaxed. This means that the present state i may be transferred to a next state j several events away, so the transition matrix becomes dense. To stress that the step-size may now be considerable, we denote it Δt instead of h.

For pedagogic reasons the presentation below begins with a simple, strictly decreasing system. Thereafter a strictly increasing system is presented, and finally a system that can both increase and decrease is composed. To make it easier to grasp, the presentation is based on concrete examples.

Example 3: *Radioactive decay*
Consider s radioactive atoms that decay with a time constant of T time units. Now, the transition matrix should handle any number of decays during a time-step. This is accomplished by assigning the transition elements of lower left triangle non-zero probabilities. According to Property 3 of the Poisson process (see Section 3.3), the number of events during an interval $(t, t+\Delta t)$, for a process with intensity λ, is *Poisson distributed* with the expected value $\Delta t \cdot \lambda$ as the argument, so the number of events during Δt is $Po(\Delta t \cdot \lambda)$ distributed. Thus the probabilities for $k=0, 1, 2, ...$ events during Δt are: $p(k)=e^{-\Delta t \cdot \lambda}(\Delta t \cdot \lambda)^k/k!$. In the present example the decaying intensity for i remaining radioactive atoms is $\lambda=i/T$. The matrix element b_{ij} (where $j \le i$), representing $d=i-j$ decays during Δt, is then:

$$b_{ij}=p(i-j)=e^{-i \cdot \Delta t/T} \cdot (\Delta t \cdot i/T)^{i-j}/(i-j)! \qquad (2)$$

The i first matrix elements of the ith row are then the i first terms in the Poisson distribution $Po(\Delta t \cdot i/T)$ (in reverse order), and the diagonal element is one minus the sum of these row elements. (In this particular example it suits perfectly that the Poisson series is truncated after $i+1$ terms, since there are no more than i atoms that can decay.) Implementing this gives:

	0	**1**	**2**	**....**	**s-1**	**s**
0	1	0	0	…	0	0
1	b_{10}	$1-b_{10}$	0	…	0	0
2	b_{20}	b_{21}	$1-b_{20}-b_{21}$	…	0	0
...	…	…	…	…	…	…
s-1	b_{s-10}	b_{s-11}	b_{s-12}	…	$1-\Sigma b_{s-1j}$	0
s	b_{s0}	b_{s1}	b_{s2}	…	b_{ss-1}	$1-\Sigma b_{sj}$

(**B** matrix)

Note that for this approach it is sufficient that $\Delta t << T$ (independently of s), while for the Bernoulli approach we would need $h \cdot \lambda << 1$; where $\lambda = s/T$ implying $h << T/s$. Thus the time-step can be increased by a factor s compared with what would be required for the Bernoulli approach. ■

Comment: A major problem with Markov models is *parameterisation*. In the example of radioactive decay above, the single natural parameter of the problem – the time constant T – is not explicitly defined once in the Markov model but is included in all the non-zero transition elements. Note also that the time-step Δt is included in all non-zero elements of the model. This makes it problematic to find an appropriate step-size since it involves changing the values in all the non-zero elements of the transition matrix.

Next consider an example of exponential growth.

Example 4: *Exponential growth*
In this example a Markov model is considered for an increasing number of entities, e.g. growth of a population, arrivals of entities, etc. The arguments are parallel to that of Example 3, but here the probability elements a_{ij} (where $i \leq j$) occupy the upper right triangle of the transition matrix, so that several events (births, arrivals, etc.) can occur during a time-step. The following transition matrix is obtained:

	0	**1**	**2**	**....**	**s-1**	**s**
0	$1-\Sigma a_{0j}$	a_{01}	a_{02}	….	a_{0s-1}	a_{0s}
1	0	$1-\Sigma a_{1j}$	a_{12}	….	a_{1s-1}	a_{1s}
2	0	0	$1-\Sigma a_{2j}$	….	a_{2s-1}	a_{2s}
...	…	…	…	…	…	…
s-1	0	0	0	….	$1-\Sigma a_{s-1s-j}$	a_{s-1s}
s	0	0	0	….	0	1

(**A** matrix)

Again, the elements a_{ij} are terms in a *Poisson($\Delta t \cdot \lambda$)* distribution where $\lambda = i/T_a$.

However, this growth example is more complex than the previous one. The problem is the unknown maximal number of entities and thus the size of the state vector and the transition matrix. The last example started with *s* entities that could decrease so the possible states were *{0, 1, ... , s-1, s}*. In this example concerning growth, it cannot be known *a priori* how many entities there will be after a full simulation. Furthermore, one has to dimension for the worst case that might appear when studying the model by performing say *10 000* replications. ■

Example 5: *Both increase and decrease*
Many systems contain both growth and decline mechanisms, e.g. a population changing because of births and deaths, number of people in a certain stage of a disease, number of intermediate isotopes in a sequence of radioactive decay, number of entities in a queuing system, etc. In this case the whole matrix becomes filled with non-zero probabilities.

	0	**1**	**2**	**...**	**s-1**	**s**
0	$1-\Sigma c_{0j}$	c_{01}	c_{02}	…	c_{0s-1}	c_{0s}
1	c_{10}	$1-\Sigma c_{1j}$	c_{12}	…	c_{1s-1}	c_{1s}
2	c_{20}	c_{21}	$1-\Sigma c_{2j}$	…	c_{2s-1}	c_{2s}
...	…	…	…	…	…	…
s-1	c_{s-10}	c_{s-11}	c_{s-12}	…	$1-\Sigma c_{s-1j}$	c_{s-1s}
s	c_{s0}	c_{s1}	c_{s2}	…	c_{ss-1}	$1-\Sigma c_{sj}$

(**C** matrix)

However, the transition matrix **C** is composed of the ***A*** and ***B*** matrices in a very complex way. It is *not* the elements of ***A*** and ***B*** that are to be plugged into the positions of the **C** matrix. Instead, it is the transition from State *i* to State *j* (an index updating by *j-i* entities) in the **C** matrix that should be the same as the combined effects of State *i* to State *j1* in the ***A*** matrix (updating by adding *j1-i* entities) and from State *i* to State *j2* in the ***B*** matrix (updating by subtracting *i-j2* entities). The updating by *j-i* entities in the **C** matrix should thus correspond to the total updating from the ***A*** and ***B*** matrices with *(j1-i) – (i-j2)* entities. Each element c_{ij} in the ***C*** matrix is therefore a function of Poisson terms from the *i*th row of the ***A*** and ***B*** matrices. For example the element c_{ij} is composed of all combinations changing the number of elements from *i* to *j* entities; i.e. *{j-i in & 0 out, j-i+1 in & 1 out, j-i+2 in & 2 out, etc.}*. Furthermore, the Poisson terms in a_{ij} and b_{ij} may already be complicated functions because of non-linear or time-dependent conditions according to $\lambda_a=\lambda_a(f(i(t),t,\Delta t)$ and $\lambda_b=\lambda_b(g(i(t),t,\Delta t)$. So in practice it is very problematic to theoretically derive the elements even in simple cases. ■

3.7. Relationship between a Markov model and a corresponding Poisson Simulation model

In order to see the parallels between a Markov and a Poisson Simulation model at a later stage, the increase and decrease matrices are here kept separate. This means that matrix **A** is used to update increments, e.g. the number of entities from arrivals during *(t, t+Δt)*. If the entities increase from *i* to *j1* only because of arrivals, then this is obtained by entering the *i*th row of the ***A*** matrix and finding the a_{ij1} element so the column index becomes *j1*.

In the same way decrements are handled by the ***B*** matrix. If the entities decrease from *i* to *j2* only because of departures, then matrix ***B*** is used to update the departures from the index row *i* to the column with index *j2*. The *total update* then is from *i* entities to *(j1-i) – (i-j2)* entities. The new number of entities therefore becomes: *i(t+Δt) = i(t) + (j1(t)-i(t)) – (i(t)-j2(t))*. This is the familiar Euler updating scheme!

Figure 2 illustrates the updating sequence during execution of the Markov model.

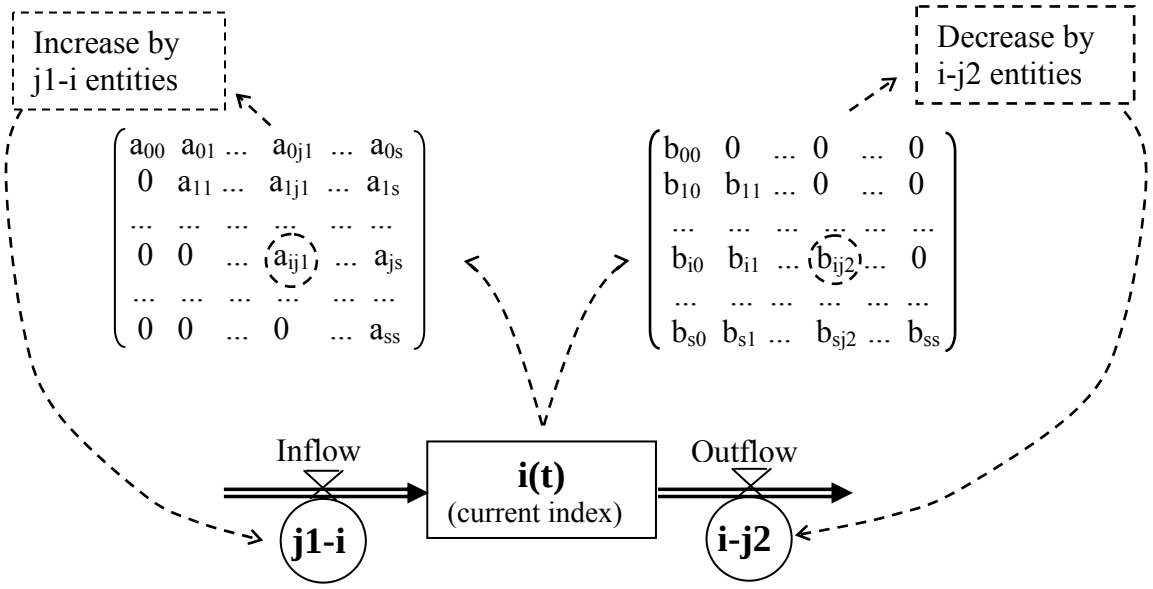


Figure 2. The number of entities *i* is increased by Inflow of entities and decreased by Outflow of entities according to: *i(t+Δt) = i(t) + (j1(t)-i(t)) – (i(t)-j2(t))*.

In detail, the updating of $i(t)$ to $i(t+\Delta t)$ entities is performed in the following way:

1. Enter line i of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices.
2. Decide the outcomes of the increase and the decrease processes. For matrix **A** the elements a_{ii} to a_{is} are the probabilities for $0, 1, \ldots s\text{-}i$ arriving entities. To realise only one of these probabilities a *Uniform[0,1]* distributed random number generator (RNG) is used. If element a_{ij1} is chosen it means an increase by $(j1\text{-}i)$ entities. The same is valid for the realisation of the number of departures from the $\boldsymbol{B}$ matrix, where a decrease of $(i\text{-}j2)$ entities is assumed.
3. Then $i(t)$ is updated by: $i(t+\Delta t) = i(t) + (j1(t)\text{-}i(t)) - (i(t)\text{-}j2(t))$.
4. Next the time index $t+\Delta t$ is renamed to t and the process starts over with the new index value of i.
5. Finally, after many iterations time reaches the end of the replication.

It can be seen that a Markov Simulation model, representing all possible states and all possible transitions between these states, is a large and complex construction just to update an index.

However, the probabilities a_{ii}, a_{ii+1} through a_{is} in point 2, above, are the first $s\text{-}i+1$ terms of the Poisson distribution. Therefore, a Poisson distributed RNG, $Po[\lambda_a \cdot \Delta t]$ (instead of a *Uniform[0,1]* RNG and the transition matrix $\boldsymbol{A}$) can be used to realise the $j1\text{-}i$ arrival events. The same is valid for the realisation of the $(i\text{-}j2)$ departures from the $\boldsymbol{B}$ matrix.

Thus, instead of the Markov model with its transition matrix, state vector and a *Uniform[0,1]* RNG, an index could be represented as just one (or a few) state variable(s), and stochastic functions based directly on the Poisson distribution could be used to update the model. Such a model would be much smaller, more comprehensive, more robust, easier to fit and faster to execute. This is what Poisson Simulation is about.

4. Poisson Simulation

4.1. A simple Poisson Simulation model

For simple models like the population model in Example 3 above, Figure 2 and the related text in Section 3.7 show how arrivals and departures of a Markov Model can be separated into **A** and **B** matrices and that their row terms are the terms in the Poisson distributions: $Po(\Delta t \cdot \lambda_a(i))$ and $Po(\Delta t \cdot \lambda_b(i))$. They also show that this enables an updating scheme where the matrices are no longer needed. The $s+1$ states are represented by one index which holds the current value of the state $i(t)$, and the transition probabilities from state $i(t)$ to state $j(t+\Delta t)$, located at the ith row of the matrices, are represented by Poisson distributions for arrivals and departures. In Markov Simulation, the actual outcomes from the ith row of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices are achieved by drawing random numbers according to the probabilities a_{ij} and b_{ij}; $j=0,1,\ldots$. In Poisson Simulation, the outcome is generated by drawing from a Poisson distributed RNG. Using this updating scheme produces:

$$\begin{cases} I(t+\Delta t) = I(t) + Po[\Delta t \cdot \lambda_a] - Po[\Delta t \cdot \lambda_b] & \text{The stochastic model in Euler's form.} \\ I(0) = i0 & \text{Initial value of entities.} \end{cases}$$

Po[~] means that for each time-step a random number is sampled from a Poisson distribution with the argument specified by the expression within the brackets.

For practical reasons, it is preferable to denote the index variable *I* by *x* and call it a *state variable*. The equation is also rewritten as a system of equations where the arrivals and the departures are separate equations. This separation becomes important when the departures from one state variable constitute the arrivals to another. This gives:

$$\begin{cases} x(t+\Delta t) = x(t) + \Delta t \cdot F1(t) - \Delta t \cdot F2(t) & x(t) \text{ holds the actual number of entities to be updated by } \Delta t \cdot F1 \text{ and } \Delta t \cdot F2 \text{ for the next } \Delta t. \\ \Delta t \cdot F1(t) = Po[\Delta t \cdot \lambda_a] & \text{Arrivals in a separate equation.} \\ \Delta t \cdot F2(t) = Po[\Delta t \cdot \lambda_b] & \text{Departures in a separate equation.} \\ x(0) = x0 & \text{Initial value of } x. \end{cases}$$

The intention with this subsection and Section 3.7 above is to demonstrate how a simple Markov model can be redesigned into a stochastic difference equation. For the general case, however, the state space is naturally expressed in *k* dimensions (see Figure 1). It is then better to define Poisson Simulation from the underlying stationary Poisson process.

4.2. General form of a Poisson Simulation model

A formal derivation of Poisson Simulation, without any references to Markov models, can be based directly on the stationary Poisson process, see [9,19]. In short, the reasoning is as follows. By extending the time-step Δt of a stationary Poisson process (Section 3.3), the condition of only zero or one event no longer holds. Property 3 in Section 3.3 then states that the number of entities during the time-step is Poisson distributed $Po[\Delta t \cdot \lambda]$.

The idea is now to use the stationary Poisson process as an approximation during appropriately short time intervals, even when the intensity λ changes because dynamics $\lambda(\mathbf{x}(t))$ or time $\lambda(t)$ are involved. The powerful properties of the stationary Poisson process can thereby be maintained as good approximations within each updating time-step. These methods can also be used in a dynamic non-stationary case by regarding the process as step-wise constant. Thus, by holding the argument $\Delta t \cdot \lambda(x(t),t)$ of the Poisson process piece-wise constant (Euler's scheme), this principle can be used step by step over the whole replication. (Although Δt is still limited to a step-size where λ can be regarded as almost constant.)

Thus to update the stochastic process

$$X(t+\Delta t)=X(t)+Po[\Delta t \cdot \lambda(X(t))]$$

on *a sequence of intervals* $[0, \Delta t)$, $[\Delta t, 2\Delta t)$, $[2\Delta t, 3\Delta t)$, ..., $[(N-1)\Delta t, N \cdot \Delta t)$; where $N \cdot \Delta t$ is the length of the simulation (where $[t_1, t_2)$ means $t_1 \leq time < t_2$). Thus, updating of the stochastic variable *X* is performed by:

$$\sum_{i=1}^{N} NumberOfEvents(i) = \sum_{i=1}^{N} \mathrm{Po}[\Delta t \cdot \lambda_i]; \text{ where the interval } i \text{ is } [(i-1)\cdot\Delta t, i\cdot\Delta t)$$

using only the *stationary* Poisson process on short intervals $[t, t+\Delta t)$.

Since $Po[\Delta t \cdot \lambda_i] \geq 0$, one term is needed for the *inflow* and one for the *outflow*, giving the fundamental model structure:

$$X(t+\Delta t) = X(t) + Po[\Delta t \cdot \lambda_{in}] - Po[\Delta t \cdot \lambda_{out}]$$

This principle can be extended to a model of any number of state variables in the same way as in Continuous System Simulation. For a model where the number of entities is sub-divided into k categories (k dimensions, see Figure 1), the Poisson Simulation model has k state variables, each represented by a stochastic difference equation. Since a state variable can hold any number of individuals, the structure of a Poisson Simulation model is independent of the number of entities.

All Poisson Simulation models turn out to have the same general form given by:

$$dx_i = Po[dt \cdot \lambda_{in\_i}(\mathbf{x},t)] - Po[dt \cdot \lambda_{out\_i}(\mathbf{x},t)] \text{ with initial values } x_i(0) = x0_i; \quad i=1,2...k, \qquad (3)$$

where x_i denotes state variable i and $\mathbf{x}$ is a k-dimensional vector of state variables. (Note that λ can be a function of any of the k state variables in $\mathbf{x}$.) *Reformatting* the differential equation into numerical form gives:

$$\begin{cases} x_i(t+\Delta t) = x_i(t) + \Delta t \cdot F_{in\_i} - \Delta t \cdot F_{out\_i} \\ \Delta t \cdot F_{in\_i} = Po[\Delta t \cdot \lambda_{in\_i}(\mathbf{x},t)] \\ \Delta t \cdot F_{out\_i} = Po[\Delta t \cdot \lambda_{out\_i}(\mathbf{x},t)] \end{cases} \qquad (4)$$

where $i=1 \ldots k$ and $x_i(0) = x0_i$.

Thus, as seen from (4), Poisson Simulation can be defined as a number of coupled (interacting) simultaneous processes that are modelled as sequences of step-wise independent, stationary Poisson processes during short time-steps Δt.

Thus, within $[t, t+\Delta t)$ all flow rates F_{in_i} and F_{out_j} are independent. However, over time the interactions are considered in $\Delta t \cdot F_{in\_i} = Po[\Delta t \cdot \lambda_{in\_i}(\mathbf{x},t)]$ and $\Delta t \cdot F_{out\_i} = Po[\Delta t \cdot \lambda_{out\_i}(\mathbf{x},t)]$.

The detailed structures of the models vary. State variables may be connected by physical flows in series, parallel or in a feedback way. A state variable may also influence the flows to or from the own or other state variables.

The system of equations can be directly implemented in any programming language (see Appendix) or better still in any Continuous System Simulation language [20-22] provided a Poisson random number generator is included, which it usually is. The execution comprises alternately updating the state variable equation and the flow equations (using a Poisson RNG to draw the number of arriving and departing entities) for each time-step.

Comment: When using a Continuous System Simulation language, one is often bound to a structure such as: $x(t+\Delta t) = x(t) + \Delta t \cdot (F1(t) - F2(t))$ and $F1(t) = Po[\Delta t \cdot \lambda_{in}]/\Delta t$ and $F2(t) = Po[\Delta t \cdot \lambda_{out}]/\Delta t$. The extra divisions by Δt do little harm.

4.3. Graphical representation of a Poisson Simulation model

In Continuous System Simulation, a frequently used technique to represent dynamic models is the Forrester diagram [20-25]. It was first introduced by Professor Jay W Forrester at MIT

for economic models [23]. It is now also frequently used in biology, ecology, epidemiology, medicine and many other areas.

In the Forrester diagram each state variable (compartment) is represented by a box containing the current number of entities. This number can change only because of physical inflows and outflows. These flows are represented by double-lined directed arrows. On each flow is a valve controlling the flow rate. Single line arrows are used to show where the valves acquire their influence (information) (see Figure 3).

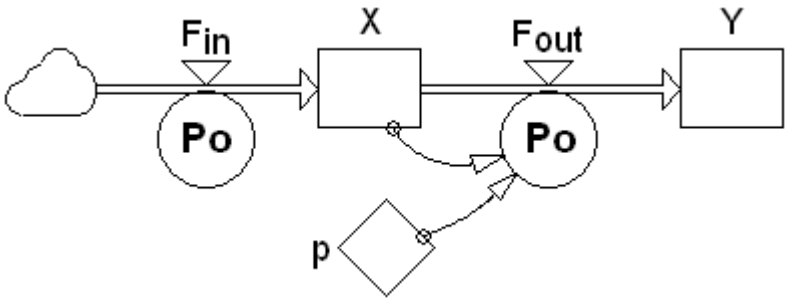


Figure 3. Forrester diagram of a model with arrivals and departures. The compartments *X* and *Y* contain the current integer numbers of entities. The compartments can only change by flows of arriving entities F_{in} or by departing entities F_{out}. Parameters are represented by rhombus symbols (◇). The single-lined arrows from *X* and *p* to F_{out} show that F_{out} is dependent on the current number of entities in *X* and on the parameter *p*.

Back in the 1960s, this type of diagram was used as the foundation for building and coding the model in the Continuous System Simulation language DYNAMO [25] – one equation of a specified form for each type of symbol. In Continuous System Simulation languages such as Powersim [20], Stella [21] and Vensim [22] to name but a few, the model is constructed directly on the screen as a Forrester diagram using icons in a click and play manner. Thereafter, each symbol is 'opened' by double clicking and values, functions or tables are specified. The 'graphical model' is then ready for execution.

The Forrester diagram works equally well for representing discrete entities in a Poisson Simulation model. To highlight the fact that the number of entities entering or leaving the state variable box through the inflows and outflows are Poisson distributed, 'Po' is included in the valves in Figure 3.

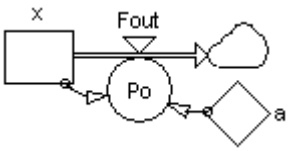
The physical flows F_{in} and F_{out} may be connected to other state variable boxes or come from/lead to a source/sink that is unspecified, i.e. outside the boundaries of the model. In the latter case the arrow starts from, or ends, in a cloud symbol for aesthetic reasons.

4.4. Some examples of Poisson Simulation models

Figures 4a-f show some Poisson Simulation models in equation form and as Forrester diagrams to display the model structures.

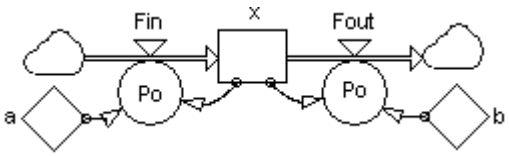
a) Radioactive decay model [26,27].

$dx = -Po[dt \cdot a \cdot x]$



b) Logistic model [26,27].

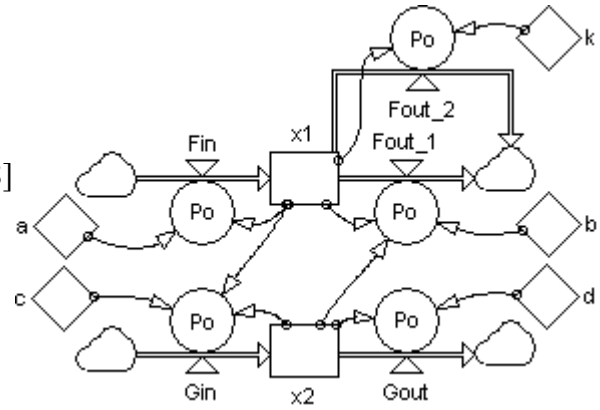
$dx = Po[dt \cdot a \cdot x] - Po[dt \cdot b \cdot x^2]$



c) Lotka-Volterra model – a prey-predator system [26-28]

$dx_1 = Po[dt \cdot a \cdot x_1] - Po[dt \cdot b \cdot x_1 \cdot x_2] - Po[dt \cdot k \cdot x_1 \cdot x_1]$

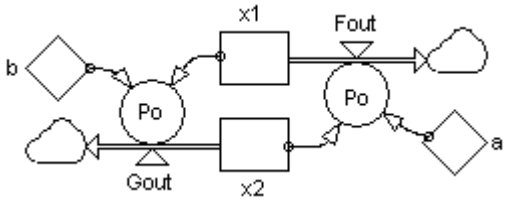
$dx_2 = Po[dt \cdot c \cdot x_1 \cdot x_2] - Po[dt \cdot d \cdot x_2]$



d) Lanchester's model of warfare [26,27,29].

$dx_1 = -Po[dt \cdot a \cdot x_2]$

$dx_2 = -Po[dt \cdot b \cdot x_1]$

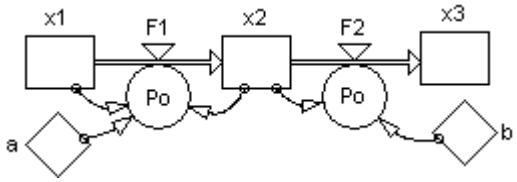


e) Epidemic SIR model [19,30-32].

$dx_1 = -Po[dt \cdot a \cdot x_1 \cdot x_2]$

$dx_2 = Po[dt \cdot a \cdot x_1 \cdot x_2] - Po[dt \cdot b \cdot x_2]$

$dx_3 = Po[dt \cdot b \cdot x_2]$



f) M/M/1 queue mode [17,33].

$dx = Po[dt \cdot a] - MIN(Po[dt \cdot b], x)$

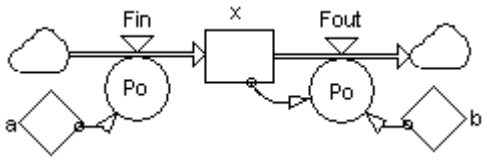


Figure 4 a-f. Some Poisson Simulation models. See references [9,19,27,33] for a presentation of these and other models treated by Poisson Simulation.

Comment on Figure 4e: For models where what leaves one state variable x_i enters another x_j, it is important that the numerical realisation uses a *single* flow equation. For the epidemic SIR

model, the flow of individuals ($\Delta t \cdot F1$) leaving stage x_1 is identical to the flow entering stage x_2. Instead of writing: $x_1(t+\Delta t)=x_1(t)-Po[\Delta t \cdot a \cdot x_1 \cdot x_2]$ and $x_2(t+\Delta t)=x_2(t)+Po[\Delta t \cdot a \cdot x_1 \cdot x_2]-Po[\Delta t \cdot b \cdot x_2]$, where two statistically independent *F1* flow rates are obtained from two separate drawings from the RNG, the model should be implemented as: $x_1(t+\Delta t)=x_1(t)-\Delta t \cdot F1(t)$ and $x_2(t+\Delta t)=x_2(t)+\Delta t \cdot F1(t)-\Delta t \cdot F2(t)$, $\Delta t \cdot F1(t)=Po[\Delta t \cdot a \cdot x_1 \cdot x_2]$. The same is valid for the flow rate *F2* between x_2 and x_3.

Comment on Figure 4f: In the *M/M/1* queue model *x* is the actual number of queuing and served individuals, *a* is the arrival intensity and *1/b* is the mean service time. For this model it is important to prevent service to an empty queue. The *MIN(arg1, arg2)* function takes the smallest of its two arguments to prevent more departures than the current number of entities in the queue.

One feature of the state variables in a Poisson Simulation model is that if the state variables are initiated to integer values they will *stay integer*, as opposed to a Continuous System Simulation model, where the state variables can take any real values.

An important advantage with Poisson Simulation is that each parameter is explicitly defined *once* in the model. Another advantage is that the time-step Δt can be changed to handle the dynamics properly without distorting the stochastic behaviour of the model.

This is perhaps an appropriate place for a warning about a possible mistake when constructing Poisson Simulation models. Even though two or more inflows (or outflows) to the same state variable may be merged in accordance with $Po[\Delta t \cdot \lambda_1] + Po[\Delta t \cdot \lambda_2] = Po[\Delta t \cdot (\lambda_1+\lambda_2)]$, it seriously distorts the model to use this on differences because: $Po[\Delta t \cdot \lambda_1] - Po[\Delta t \cdot \lambda_2] \neq Po[\Delta t \cdot (\lambda_1-\lambda_2)]$.

4.5. Implementation and statistical tools for Poisson Simulation

4.5.1. Implementation

Poisson Simulation is a theoretically sound, easy handled and computer-efficient way of realising a stochastic compartment model. The calculation effort is also almost independent of the number of entities. It is easily implemented in any Continuous System Simulation language where a random number generator for the Poisson distribution is included. It is also easy to write the model directly in a general purpose programming language such as Pascal, C, FORTRAN, Java, etc., see Appendix. Poisson random number generators can be found in e.g. [34-36].

4.5.2. Tools for statistical analysis

The very essence of a stochastic model is that the detailed outcome of a replication is unpredictable so the results differ between replications. Therefore, a number of replications must be performed and the results from them collected, statistically analysed and presented.

To estimate means, standard deviations, confidence intervals, correlation, min and max of quantities *X* and *Y*, it is sufficient to cumulate X, Y, X^2, Y^2 and $X \cdot Y$ and to check if *X* and *Y* are the smallest/largest results so far for each of the *N* replications. After the loop these statistical measures are calculated in a few lines of code.

Furthermore, it is convenient to have a supervisory programme that can order a number of replications, collect the outcomes and present them in probability distribution functions (p.d.f.). From the p.d.f., statistical information such as mean, standard deviation, confidence intervals, percentiles, results of hypothesis tests, etc. can be derived. Correlations between different quantities can also be directly studied (or derived from multi-dimensional p.d.f.s).

Such programmes [37,38] are available for the Continuous System Simulation language Powersim [20] and for MATLAB [39].

Although the time to execute a Poisson Simulation model is only a few times longer than for executing a corresponding deterministic model, the need for *100* to *10 000* replications to build the p.d.f. makes a substantial difference.

5. Comparisons

Markov Simulation and Poisson Simulation models are both based on the Poisson process. If correctly handled they should therefore produce the same results – they are *consistent*. To be exact, any large number of replications with the two methods should produce probability distribution functions (p.d.f.s) of the outcomes that are not significantly different.

A fair way of comparing Markov Simulation and Poisson Simulation is to investigate what happens when they are used in a real modelling and simulation project. Such a study may concern any kind of discrete entities to be modelled such as decaying atoms, bacterial growth, epidemic, combat, competition between species, etc. Here the system under study and the purpose of the study are intentionally not specified beforehand in order to provide the flexibility to discuss different aspects and problems that may emerge.

The phases of the project must then be gone through systematically in order to identify problems, performance, advantages and disadvantages of using Markov Simulation and Poisson Simulation.

Apart from the terminology, which varies somewhat, a model study of a system should contain the following project phases [11-14]:

1. **Problem recognition:** (Identifying a problem. A first sketch e.g. in the form of a graphical representation.)
2. **Problem definition:** (Formulating objectives and setting system boundaries. Creating a conceptual model.)
3. **Model building:** (Choice of model type. Finding an appropriate model structure. Model fitting to calibrate the parameters. Choice of step-size. etc.)
4. **Validation of the model:** (Is the model trustworthy? Is the model good enough for the specified objectives? Verification of technical correctness. Validation using data from the system independent of those used for model fitting.)
5. **Analysis:** (Using the model in accordance with the objectives; e.g. estimation of consequences, optimisation, sensitivity analysis, control, prediction.)
6. **Result evaluation:** (A final evaluation of the results. Is the model structure realistic? What would other assumptions give? Sensitivity analysis. Interpretation of the results.)
7. **Result presentation:** (Graphical, verbal and numerical presentation of system, model and results in a comprehensive form in accordance with the objectives.)

- **Data acquisition:** (Collection of measurements, observations and other information needed for phases 1 to 7.)

Many issues, depending on the choice of a Markov or a Poisson Simulation approach, occur throughout the different phases of the project. These issues are investigated, phase by phase, in the following. Since many aspects occur in different phases of the project, each is discussed more thoroughly the first time it appears and is simply mentioned when it occurs in later phases.

In the comparison, the type of Markov simulation approach (Bernoulli, exponential or Poisson) that is compared with Poisson Simulation is also important. The Bernoulli, exponential and Poisson approaches are all based on matrices of the same size, although of different structures. However, the Poisson approach is much too complex to construct. It was originally included to demonstrate the structural relationship to Poisson Simulation, but is of limited interest in the following comparison.

5.1. Problem recognition

As soon as a problem is recognised, powerful concepts are needed to comprehend the nature of the system under study. There is also a need for a way to sketch any thoughts and ideas that arise in a more formalised way.

5.1.1. Problem-orientated modelling

The comprehensive concepts of *state variables* (*compartments*), *flows* and *distinct parameters* make the Poisson Simulation approach considerably more *problem-orientated* than the Markovian approach.

State variables, such as number of predators, infected individuals, queuing entities, etc., are concepts with a direct counterpart in the system under study. This contrasts with the huge number of *states* representing all possible *k*-tuples, where *k* is the number of dimensions (see Section 2). Such tuples are abstract, theoretical constructs, to enable a detailed mathematical analysis of a model without a physical counterpart in the system under study.

In a similar way, *flows* as an aggregation of events that transfer the entities from one state variable to another also have a direct physical correspondence in the studied system where they can often be directly measured. The transition probabilities, on the other hand, are an often huge assembly of conditional probabilities of the different transitions between the (more abstract) states.

Furthermore, each *parameter* of a studied system, such as fertility, sojourn time, risk, etc., is a comprehensible concept that can be directly modelled at a single place in a Poisson Simulation model. In a Markov model it becomes distributed over the transition matrix.

5.1.2. Graphic representation of the model

Already for the *conceptualisation* of the system or process under study some form of graphical description is needed to sketch a model. Later on, the graphical description is needed for structuring knowledge, for overview and for communication with other people within or outside the project. It also helps in documenting the model in a comprehensive way and in describing the model in the final presentation of the project.

For Markov models a *state transition diagram*, where every state and every possible transition is depicted, is often used in textbooks to explain how Markov models work and to illustrate theoretical concepts such as *periodic*, *absorbing*, *recurrent* or *isolated* states, with simple examples. However state transition diagrams are far too complicated and detailed to be used for all but the very simplest models, as the number of states and transitions soon becomes huge. Furthermore, a state transition diagram is not suited to reveal the dynamic structure of the studied system.

The *Forrester diagram* (Figures 3 and 4) is perfectly suited for the stochastic and dynamic Poisson Simulation models. It is a powerful tool for conceptualisation, structuring the knowledge, overview, communication, documentation, presentation and even for direct implementation as a computer model. In all these respects it is widely superior to a state transition diagram.

5.2. Problem definition

Problem definition means formulating the objectives of the project and defining the boundaries of what to include in or exclude from the study. This has a profound impact on the whole project – e.g. type of model, model structure, what data are needed, what to validate, what to analyse, control, optimise, etc., what to evaluate and present as results. However no decision has yet been made on what model approach (e.g. Markov model or Poisson Simulation model) to use.

The problem definition can be supported by the problem-orientated, comprehensive concepts of state variables, flows and distinct parameters that are the basic elements of Poisson Simulation and by the unifying Forrester diagram.

5.3. Model building

The model building phase has many aspects to consider when comparing a Markov and Poisson Simulation approach. Here, these are divided into: 5.3.1 Model size, 5.3.2 The sojourn time distribution of a stage, 5.3.3 Finding an appropriate step-size, 5.3.4 Specifying parameters or transition probabilities, 5.3.5 Execution time, and 5.3.6 Flexibility to change the model.

5.3.1. Model size

In Poisson Simulation the studied system is represented by a small number (k) of *state variables*, where one state variable can represent any number (n) of entities.

The state space contains the same states in a Markov model and a Poisson Simulation model, but the k dimensions spanning the state space act as coordinates of k-tuple for definition of the states in a Markov Model (see Figure 1). Thus, the Markov state vector is a massive *disaggregation* from the state space dimensions into all possible *states* in which the process can exist. For a population model ($n>1$) this means all possible combinations of subpopulation sizes. For a model of $k=3$ dimensions and $n=3$ entities, these entities can be combined into any possible states described by the 3-tuple (c_1, c_2, c_3) where $c_1+c_2+c_3=n$; i.e. into the set of ten states $\{(3,0,0), (2,1,0), (2,0,1), (1,2,0), (1,1,1), (1,0,2), (0,3,0), (0,2,1), (0,1,2), (0,0,3)\}$.

The number of states in a Markov model follows Bose-Einstein statistics, which describe the number of ways n identical particles can be distributed into k cells without restriction on the number of particles per cell.

For a model structure where every particle (entity) can populate every cell (dimension of state space) the formula for the number of states in the state space is: $\binom{n+k-1}{k-1} \equiv \binom{n+k-1}{n}$, where k is the dimension of the state space and n is the constant number of entities in the population [4].

For $k=3$ dimensions and $n=10$ entities, a Markov model has a state space of 66 states. However even for $n=100$ entities, the state space consists of *5151* states, while for $n=1000$ entities the state space comprises about half a million states (instead of 3 *state variables* in a Poisson Simulation model).

The number of states in the state space (i.e. the number of rows in the transition matrix in a Markov model) with k=5 dimensions and n=1000 individuals is then around $4 \cdot 10^{10}$.

Thus the number of states, even for quite a small Markov model, can become astronomically high, while in Poisson Simulation there is no problem with models of e.g.

hundreds or even thousands of state variables since the model size grows only linearly with the number of state variables (*k*) and not at all with the population size (*n*).

As Figure 5 shows, the model size in terms of state space dimensions (*k*) and population (*n*) that can be handled with Markov Simulation is indeed very limited, while Poisson Simulation can handle really large models.

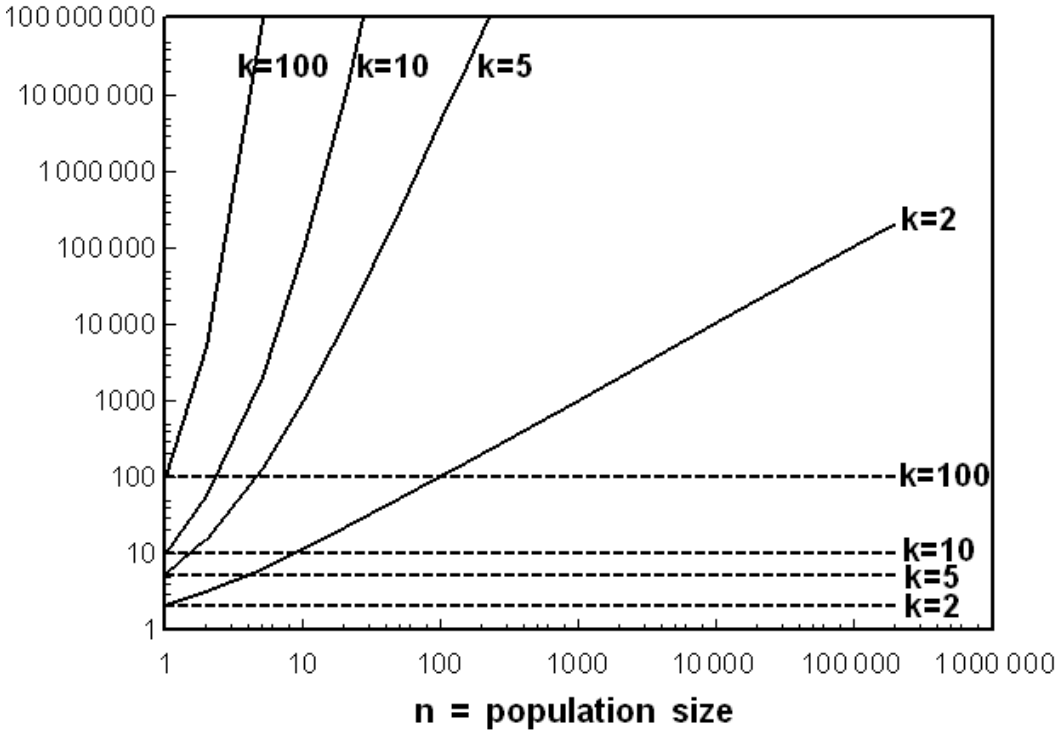


Figure 5. The number of *states* for a Markov model (solid lines) and the number of *state variables* for a Poisson Simulation model (dashed lines) as a function of dimension (*k*) and population size (*n*).

The model size discussed above assumes a constant number of entities. For models where the number of entities is growing, the number of states is unknown *a priori* in a Markov model. If the model is to be run e.g. *10 000* times to produce an accurate p.d.f. or confidence interval, allocation must be made for the largest number of entities that may arise – or some mechanism for dynamic expansion of the model has to be included. In Poisson Simulation this is never a problem since the state variables can hold any number of entities.

However, there is a more 'trivial' *special case* where the *n* entities are *non-interacting* – see Section 3.2 for examples. For such a system the size of a Markov model can be drastically reduced by studying the entities one-by-one and afterwards superimposing the *n* results. This reduction in size – see Figure 5 with *n=1* instead of say *n=1000* – gives the Markov model a reasonable size. However, this comes at a cost of *n=1000* replications of the Markov model for one experiment, compared with a single replication when using a Poisson Simulation model.

5.3.2. Sojourn time distribution of a stage

In both Poisson Simulation and Markov Simulation it is equally important to appropriately model the statistical distribution of the time in a stage [40]. Only in the rare case when the

sojourn time has an exponential distribution is it appropriate to model the stage by a single state variable or the current stage by a single state. Otherwise the stage has to be modelled by a structure of state variables or states in series, parallel and/or feedback.

An illustrative example of how the sojourn time distribution of a stage may heavily influence the results is the following epidemic SIR model (see also Figure 4e). Models of infectious diseases are generally based on a sequence of *stages* from Susceptible via Infectious to Removed. Such a model is therefore denoted a SIR model.

Example 6: An epidemic *SIR model*

The first SIR model was published by Kermack and McKendrick in 1927 [30] and since then books and studies on the epidemiology of infectious diseases are usually based on the SIR and related models [4,31,32].

In its deterministic form an epidemic of a certain magnitude occurs if, and only if, $R_0=S\cdot T\cdot p>1$, where S is the number of Susceptibles, T is the expected sojourn time in the Infectious stage, and p is the risk-of-infection parameter. The model to be studied is specified as $S(0)=1000$, $I(0)=1$, $p=0.0003$ and the average sojourn time in the Infectious stage is $T=4$ time units (so $R_0=1.2$ and an epidemic is created in the deterministic case).

In a stochastic setting (see Figure 4e where $a=p$ and $b=1/T$), the outcomes from many replications vary and have to be collected into a probability distribution function over outbreaks of different sizes.

Since the infection process within the individual always requires a finite time, the sojourn time distribution is unlikely to be exponential. Depending on the disease studied, the sojourn time will have some specific statistical distribution. Consider what happens if the sojourn time distribution of the Infectious stage is allowed to have a distribution other than exponential (implied by modelling the Infectious stage by a single state variable).

An often useful family of distributions for biological and medical systems is the gamma family. So let us test the *gamma*(α,β) distribution for $\alpha=1,2,3,5$ and *10* and $\beta=T/\alpha=4/\alpha$ time units. This is accomplished in Poisson Simulation by a structure of α state variables in a series, each having the sojourn time of T/α time units. (The model for $\alpha=3$ is given in the Appendix.)

The effects of different statistical distributions of the sojourn time are easily tested using Poisson Simulation. The results from *10 000* replications of the stochastic model give a p.d.f. that is condensed to the average size of the epidemic and the *95%* confidence interval of that estimate in Table 1, which also shows the results of the corresponding deterministic model.

Table 1. Number of individuals becoming ill for different assumptions on the sojourn time distribution. Results obtained from *10 000* replications of the Poisson Simulation model (average number of Susceptibles that become infected: *S(0)-S(End)*, and the *95%* C.I.), and results from the corresponding *deterministic* model

Model type	SIR Γ(1,T)	SIIR Γ(2,T/2)	SIIIR Γ(3,T/3)	SI_5R Γ(5,T/5)	$SI_{10}R$ Γ(10,T/10)
Stochastic: Av. (95% C.I.)	55.1 (52.7-57.5)	68.1 (65.5-70.7)	75.1 (72.4-77.8)	83.0 (80.2-85.8)	84.1 (81.3-86.8)
Deterministic	318.5	318.5	318.5	318.5	318.5

As shown in Table 1, the average results from a stochastic SIR model differ greatly from those of a deterministic model, as is well known. Furthermore, the model structure, SIR, SIIR etc. do not affect the size of the epidemic for the deterministic models.

In the stochastic case, however, using a SIIIR instead of a SIR structure changes the average size of the epidemic by about *35%*. ■

However, as a consequence of size, the costs of modelling the Infectious stage are much higher in a Markov model than in a Poisson Simulation model. For example, if the sojourn time T in the Infectious stage of the disease has a *gamma(3,T/3)* distribution, it can be modelled by three state variables in a series in the Poisson Simulation model. For the epidemic SIR model this would increase the number of state variables from *3* to *5*, independently of the size of the studied population. However for a Markov model with only *10* individuals the increase would be from *66* to *1001* states. Therefore, the Markov modeller pays a very high price for making the model more realistic. On the other hand, oversimplification creates poor results and erroneous conclusions.

Although the shape of the sojourn time distribution may have a strong impact on the results of a stochastic model, this fact is often neglected in Markov modelling. One reason is probably that Markov models rapidly become too huge when the number of states is increased. Another reason may be that the deterministic model does not reveal that the sojourn time distribution of the Infectious stage is of any importance.

5.3.3. Finding an appropriate step-size

Real time is continuous. Modelled time can be divided into finite slices only because of the way digital computers work. Fortunately, the errors in doing this can be reduced to any small number provided that the size of the time-step is sufficiently small. An alternative way of handling time is to jump from the instant of an event to the instant of the next one.

The value of a sufficiently small step-size depends on the time constants for the dynamic processes. A rule of thumb is that it should be considerably smaller than the smallest time constant of the system (T_{min}). How much smaller depends on the required precision. For moderate precision $T_{min}/10$ is often an acceptable step-size.

In Continuous System Simulation the standard method is to run the model with some large Δt. This Δt is then repeatedly halved to find an appropriate value where the effects on precision are acceptable. In Poisson Simulation the way to find an appropriate step-size starts with using the deterministic model *embedded* in the Poisson Simulation model (the model where the stochastic Po-parts – but not the arguments – are eliminated). Then different step-sizes of Δt can be tested. Thereafter, the stochastic Poisson Simulation model is resumed with the appropriate Δt from the deterministic model. A slight modification of Δt may then be necessary so the Poisson Simulation model should also be tested with Δt and, say, $\Delta t/2$ to see whether the resulting p.d.f. from a number of replications are similar.

Since Poisson Simulation is an extension of Continuous System Simulation, an important property of Poisson Simulation is that the model behaviour is preserved when a sufficiently small step-size is decreased. This property is valuable for finding an appropriate step-size – small enough to handle the dynamics of the model but not smaller than necessary because too small a Δt only increases the execution time.

For Markov Simulation the time handling approach chosen has to be considered. For the exponential approach there is no technical time-step to be chosen, which eliminates the problem of finding a proper step-size.

For the Bernoulli and Poisson approaches to Markov modelling it is more problematic to find an appropriate step-size, since a change in the step-size affects all the transition probabilities of the model. An erroneous and dangerous practice of avoiding the problem by setting the step-size to the same as that for the interval between observations is often used.

This confusion of sampling interval and step-size must be strongly condemned since, unless the sampling time is sufficiently short, it severely distorts the results and conclusions.

An important point is that when the distribution of a stage is changed from e.g. *exponential(T)* to *gamma(3,T/3)*, the new structure of the modelled stage has three state variables in series, each with a sojourn time of *T/3*. If *T* is the shortest time constant in the former model and Δt is adequate to handle it, the step-size now needs to be reduced to $\Delta t/3$ to keep the precision of the calculations. This is trivial to accomplish in Poisson Simulation but requires complete rebuilding of the Markov model based on the Bernoulli or Poisson time handling approaches.

5.3.4 Specifying parameters or transition probabilities

In principle, there are two different ways of building a model.

1. Relying on known relations; e.g. the number of decays *of* n radioactive atoms is *n/T* per time unit where the quantities *n(t=0)* and *T* have known values.

2. First constructing a model structure. In this case the parameters are unknown and have to be estimated, which is achieved by tuning the set of unknown parameters so that the model behaves as similarly as possible to the behaviour of the studied system. This is called *parameter estimation*.

In both cases each parameter of the studied system (such as *time constant*, *sojourn time*, *fertility*, *interest rate*, *risk*, etc.) can be directly defined and assigned a value at a single place in a Poisson Simulation model (as for *p* and *T* in the epidemic SIR models in Figure 4e, Example 4 and Appendix). This contrasts with the Markov model, where the parameters are not defined explicitly anywhere, but are spread out over all the transition elements where they are mixed with other quantities. Therefore, assignment of values to all the transition probabilities of a transition matrix is a much more problematic procedure. When all the transition elements are unknown there are a huge number of parameters to estimate. This often leads to a severely over-parameterised model and to disastrous results.

Parameter estimation is closely connected with the number of parameters and the risk of over-parameterisation. To get an idea of this problem, consider a static and deterministic model where there are *M* equations and *Np* unknown parameters. If $M=Np$ there is exactly one solution. If $M>Np$ the model can be fitted by regression. The problem comes when $M<Np$, in which case there are infinitely many solutions. Taking any one of these infinitely many solutions will handle the fitting problem – but the model cannot be expected to pass the validation against new data. For a dynamic model (stochastic or not) the problem is similar. Using an over-parameterised model produces nonsense results, since one cannot fit more parameters than there are reliable observations to match. If the number of parameters is too large the model is over-parameterised and the model has to be simplified or more data are needed.

The problem of over-parameterisation is particularly problematic for Markov models since all the non-zero elements, p_{ij}, of the transition matrix have to be estimated.

Even when the transition probabilities, p_{ij}, can be deduced from known relations so we have the same number of unknown parameter values as in a Poisson Simulation model, the transition matrix has to be rebuilt for each new test of a parameter set in the search for the best estimate of the parameters.

5.3.5. Execution time

The execution time for a Poisson Simulation model was explored here by testing the SIIIR model presented in Example 6 in Section 5.3.2. This model was coded as the complete Pascal program including a simple Poisson RNG shown in Appendix. This model (where the screen output was suppressed) was executed for *10 000* replications, which took *19.5* seconds on a *3* GHz PC.

Virtually all of the execution time was spent in the ten state variable, flow and auxiliary statements of the '*AGAIN loop*' close to the bottom of the code. In particular, the four calls to '*Function POISSON( )*', which in turn frequently calls the built-in uniform *U[0,1]* RNG named 'Random', required most of the execution time.

The simple Poisson distributed RNG used here (*Function POISSON* in Appendix) is appropriate for models of this size. This RNG requires $r+1$ calls to the *U[0,1]* generator for an outcome of r events. However, there are often more efficient ways to acquire Poisson distributed random numbers. For example, for larger arguments (say $arg > 15$) the Normal approximation can be used. The cumulative distribution functions of the Poisson distribution can even be tabulated so that an inverse transform algorithm can be used [35].

For Markov Simulation the situation is more problematic. Even this small SIIIR model with $k=5$ state variables and $n=1001$ individuals gives rise to a huge Markov model with some $4 \cdot 10^{10}$ states.

Now, assume that it has been possible to correctly build such a Markov model and also to deduce or estimate all the transition elements. How long would the execution time be compared with that of the corresponding Poisson Simulation model? The answer is that a Poisson Simulation model executes many times faster than a corresponding Markov Simulation model. The reason is as follows: For a Markov model using the Bernoulli time handling, the time-step h must be so small that it almost guarantees that just zero or one transition occurs in $(t, t+h)$. Then an event only *might happen* during the time-step. If, for example, h is short enough to only, erroneously, give two or more events in 1 case out of a hundred, then it will create one event in about 13 cases out of the hundred – and zero events in the other *86* cases! (Calculated from the Poisson distribution.) This means that the number of time-steps in this case is about seven times the number of events. Thus, in the Bernoulli approach to Markov Simulation the calls to a random number generator are to check *whether* an event might occur in the next time-step.

The aggregation of many events (for each of several flows) during a large time-step, instead of a very short time-step (where zero events will most probably occur), makes the execution of a Poisson Simulation model orders of magnitude faster than execution of a corresponding Bernoullian Markov Simulation model (if it even is possible to build the Markov model).

Finally, a Markov model based on the exponential approach would execute considerably faster than one based on the Bernoulli time handling approach. Here, one event is handled for each time-step. So for each time-step one RNG call is needed to obtain the instant of the next event and another RNG call to determine which is the next state to be visited. The execution time for this approach is of the same order as the execution time when the same algorithm is used in a stochastic compartment model using the so-called Stochastic Simulation Algorithm (SSA) [41]. Experiments show that stochastic simulation of compartment models with few events per time-step have about the same execution speed as the same model using Poisson Simulation. However, for many events in several flows during a time-step the SSA method is considerably slower. In any case, a Markov model based on the exponential time handling approach has a much faster execution than one based on the Bernoullian approach, and might in very favourable cases (few events per time-step) execute with a speed of the same order as Poisson Simulation.

5.3.6. Flexibility to change the model

In the model building phase as well as during other phases of the project, more information is obtained about the system under study, the model and its defects. For example: programming errors are detected, hypotheses or assumptions are changed, components are added or removed and the model structure may be modified. It is then important that the model is flexible enough to incorporate these changes with a reasonable effort. As seen above, this is considerably easier in Poisson Simulation than in Markov Simulation because of transparency, size, explicit parameterisation, ease of adjusting the step-size, etc. However, the exponential Markov approach is somewhat more flexible than the other Markov approaches since no time-step is involved in the transition matrix.

In conclusion on model building, *size*, *over-parameterisation* and *over-simplified models* in particular are frequent and often insurmountable problems in Markov modelling, while little extra effort is needed to explore more detailed and realistic structures without making the model large and over-parameterised when using Poisson Simulation.

5.4. Validation

The purpose of validation is to decide whether the model is trustworthy, and good enough to solve the problem stated as objective(s) in the problem definition. Is it useful for its purpose?

A systematic validation may uncover deviations because of wrong hypotheses, poor data or programming errors. However, the major issue is whether the model will behave like the system described by new data – independent of those already used at the model fitting (Section 5.3.4). Often, deviations are found and the model has to be more or less modified, a process that is simplified if the model is flexible (Section 5.3.6).

Looking for technical errors such as programming errors is often called verification. Again it is easier to find errors in a Poisson Simulation model since it is much smaller and more transparent than a Markov model.

In validation, sensitivity analysis is a frequently used technique. For stochastic models sensitivities are affected by random variations. In Poisson Simulation one has access to the random number generators and can reuse the seeds of the generators to reduce the stochastic influence on the sensitivity estimates. This should if possible also be done using a Markov model for sensitivity analysis.

5.5. Analysis

The analysis phase of the project is where the model is used to solve the problem specified as the objective. This may concern understanding the behaviour of the studied system, estimates of the results of certain measures, predicting the future, controlling the system, optimising its behaviour, etc.

It would be too extensive to discuss all these possibilities, but the conclusion is that the more flexible the model is, the simpler and better the analysis is performed. In general, Poisson Simulation is considerably better suited than Markov Simulation in these respects.

5.6. Result evaluation and Result presentation

Finally, it is time to check whether the results of the model study comply with the issue formulated in the objective. It is also time to interpret the results and put them in context, and to reformulate the simulation results into digestible project results in the form of text, result tables and diagrams. Issues already discussed, such as graphical presentation, transparency, model size, quality of validation, etc. are then once again of interest.

6. The full power of Poisson Simulation

In order to present the similarities between Poisson Simulation and Markov Simulation and to compare their merits and demerits, this study has so far intentionally restricted its focus to *stationary* processes where the Markov property is valid.

The *Markov property* states that the model is memoryless and implies an exponential (or geometrical) sojourn time distribution in a state. When this is relaxed the model belongs to the wider class of semi-Markov models. Then the process is only required to behave as a Markov process at instants of state transitions.

The stationarity means that the *intensities* λ_{ij} (or *transition probabilities* p_{ij}) are constant in time. In Markov terms this can also be expressed as: $Pr[X(t_{n+1})=j \mid X(t_n)=i]$ is independent of t_n. In the non-stationary case λ_{ij} or p_{ij} in the Markov model and *parameters* in a Poisson Simulation model become functions of time.

As we will see, general sojourn times and non-stationarity bring additional problems to Markov simulation, while Poisson Simulation already has the capability to handle these qualities.

6.1. Sojourn time distributions other than exponential

Poisson Simulation can swiftly cope with non-exponential sojourn time distributions. Assume that the aim is to model a *Uniform[0,T]* or a *Triangular[0,T/3,T]* or even an *empirical* sojourn time distribution of a stage. Such non-dynamic distributions can be approximated by a structure of state variables in Poisson Simulation in series, parallel or feedback, but would require a number of state variables. The same could in principle be done in a Markov model, but this would increase the number of states considerably.

Alternatively in Poisson Simulation, one can use *a multi-entrance array* of $T/\Delta t$ elements that for each Δt shuffles the contents by one place. When an entity arrives at the stage, modelled by the array, a random sample from the proper probability distribution is drawn to decide the time it should stay there. If this time is τ the entity enters an array element located $\tau/\Delta t$ places before the array exit. For a more detailed description see [33]. Technically, the array can be implemented as a static circular buffer [42,43], so the array size is small, and the execution is fast since only a pointer to the actual time is updated at each time-step instead of shuffling the contents around. (Entrance and exit are synchronised relative to the pointer.)

But what would happen if such a multi-entrance shift mechanism were to be implemented into a Markov model? A matrix that randomises the entrance row in accordance with any probability distribution can be created, and a compulsory shift mechanism to the next 'array state' is trivial. However, the state space would be horrendously large. For only a small array of $T/h=100$ cells and some tenths of entities, this construction alone would be googol-sized – the number of states far exceeding the number of atoms in the known universe.

6.2. Non-stationary models

For non-stationary stochastic processes, the Markov property stating that the probability for the next state only depends on the current one fails, since it also depends on time. This implies that the elements p_{ij} of the transition matrix have to be reconstructed for each time-step instead of remaining static.

For Poisson Simulation, non-stationarity is directly handled without any complications. For example, constant parameters are replaced by time functions and time-dependent flows are modelled to include the time dependency.

6.3. General stochastic processes

Let us now further broaden the field from semi-Markov to stochastic processes in general. There is still a state space and a time index, but the Markov condition is replaced by a much more general relationship between the random variables X_i where the joint cumulative distribution function among the random variables is to be specified. (See the classification of stochastic processes in Section 3.1)

Example 7: *Correlation*

Just as an example, consider the stochastic Lotka-Volterra model in Figure 4c in Section 4.4. Qualitatively, this model produces oscillations (as does a corresponding deterministic model). It also produces stochastic variations with the risk of extinction of one or both species (which a deterministic model fails to do). However, the model could be refined in a number of ways. Here we only focus on the concept of *correlation*. Assume that the Lotka-Volterra model is implemented as:

$$\begin{cases} x_1(t+\Delta t) = x_1(t) + \Delta t \cdot F_1 - \Delta t \cdot F_2 - \Delta t \cdot F_3 \\ x_2(t+\Delta t) = x_2(t) + \Delta t \cdot F_4 - \Delta t \cdot F_5 \\ \Delta t \cdot F_1 = Po[\Delta t \cdot a \cdot x_1] \\ \Delta t \cdot F_2 = Po[\Delta t \cdot b \cdot x_1 \cdot x_2] \\ \Delta t \cdot F_3 = Po[\Delta t \cdot k \cdot x_1 \cdot x_1] \\ \Delta t \cdot F_4 = Po[\Delta t \cdot c \cdot x_1 \cdot x_2] \\ \Delta t \cdot F_5 = Po[\Delta t \cdot d \cdot x_2] \end{cases}$$

where x_1 is the number of prey and x_2 is the number of predators. Then the number of prey-deaths because of predators during Δt ($\Delta t \cdot F2$) is uncorrelated to the number of predator-births during Δt ($\Delta t \cdot F4$).

To instead make prey-deaths and predator-births correlated, a single Po-call is made for both $\Delta t \cdot F2$ and $\Delta t \cdot F4$, for example using the code: $Encounters = Po[\Delta t \cdot x_1 \cdot x_2]$; $\Delta t \cdot F_2 = b \cdot Encounters$ and $\Delta t \cdot F_4 = c \cdot Encounters$.

Prey-deaths and predator-births can be made partly correlated by dividing the flow rates F_2 and F_4 into $F_{2,corr} + F_{2,uncorr}$ and $F_{4,corr} + F_{4,uncorr}$ where the uncorrelated parts have independent calls to the Poisson distributed RNG and the correlated parts have a common call to it. ■

6.4. Combined simulation

Deterministic and stochastic sub-models can be mixed in a Poisson Simulation model. This is because a deterministic equation can be linked to the integer outcome of a stochastic equation, and a stochastic equation can take any value (not only integers) as argument in a Poisson function.

For example, assume that the prey x_1 in Example 7 above are so many that they may be regarded as continuous matter (e.g. micro-organisms or grass). We can then drop the *Po[]* part of equations $\Delta t \cdot F_1$, $\Delta t \cdot F_2$ and $\Delta t \cdot F_3$ but keep it for the predators (equations $\Delta t \cdot F_4$ and $\Delta t \cdot F_5$).

In a wider scope this opens the way for new possibilities in *combined simulation* where, traditionally, Discrete Event Simulation (DES) was used to handle discrete entities and Continuous System Simulation (CSS) was used to handle continuous quantities. Hereby, several advantages are obtained. First, a theoretically sound foundation is achieved in which aggregation can be performed safely. Second, combined type problems can often be performed exclusively in PoS, which is often considerably simpler than including DES. The computational complexity is reduced when a large number of discrete entities can be modelled as flows into and out of a relatively small number of state variables. Third, even when it is practical to include DES, the combined DES/Poisson Simulation approach is more powerful and flexible than a combination of DES and CSS

6.5. Other facilities in Poisson Simulation

Poisson Simulation can utilise the rich library of functions and procedures of its host language – be it a Continuous System Simulation language or a general purpose programming language. Such a library contains *mathematical functions* such as trigonometric functions, logarithms, square root, etc., *time triggered functions* such as *PULSE*, *STEP*, *RAMP* or *SAMPLE functions*, *random number generators* of different distributions, *logical functions* such as *IF*, *MIN* and *MAX* functions, *dynamic time delays* of specified order, *static delays* that can handle pure time delays of the form *X(t-T)*. Furthermore, there are *table look-up functions* for one or several dimensions, so any empirically found relationship can simply be plugged into the model without any mathematical treatment or complication. Even vectorisation of state variables, flows and parameters is often very handy if e.g. a human population is to be subdivided into five-year age groups.

It is also a great advantage that the model can be directly written and executed in a Continuous System Simulation language with its problem-orientated concepts, the possibility of giving the concepts meaningful names, syntax checking, checking for algebraic loops, output facilities, etc. Tools for statistical analysis of many replications of a Poisson Simulation model are also a great asset [37,38], see Section 4.5.2.

7. Discussion and conclusions

Markov theory plays a fundamental role in the theoretical analysis of stochastic processes and it enables a mathematical/statistical analysis of simpler models from various fields. Markov models are also frequently used for simulation of stochastic and dynamic models, which is the main focus of this paper.

The theoretical foundation of both Markov Simulation and Poisson Simulation is the Poisson process. The two types of models are also consistent in that in principle they are applicable to the same types of problems and produce consistent results. Technically, the Markov approach is based on transition probabilities between all possible states, while the Poisson Simulation approach focuses on flows between state variables.

7.1. Comparisons

The main purpose of this paper is to compare how well Markov Simulation and Poisson Simulation perform in a project. To identify problems, merits and demerits and to compare performance, the paper examined how Markov Simulation and Poisson Simulation affected the different phases of a project from Problem recognition and Problem definition via Model building, Validation, Analysis to Project evaluation and Presentation.

Before the main comparisons we established that of the three time handling approaches to Markov simulation, the exponential approach is usually superior in several regards. It creates a simpler structure of the transition matrix, it eliminates the need to find a time-step for updating the model over time and removes the approximation generated by a fixed time-step; and the model executes considerably faster than a Markov model based on Bernoulli time handling.

However, Markov simulation was inferior in all aspects studied compared with Poisson Simulation. Two problems of Markov Simulation stood out as monumental for all but the very simplest models – size and the problem of assigning values to the transition elements.

Size: The Markov model is a hugely *disaggregated* construction because *every possible state of the system* and *every possible transition between these states* are represented. For a Markov model the number of states in the state space grows combinatorially with the population size and with the number of categories of the population (Figure 5). Poisson Simulation, on the other hand, *aggregates the state space* into a small number of *state variables*, where a state variable can hold any number of entities. This makes a Poisson Simulation model independent of the population size and orders of magnitudes smaller than the corresponding Markov model. Therefore, much larger and more complicated models can be built and executed with Poisson Simulation than is possible with the Markov approach. The practical limitations for Poisson Simulation are about the same as for Continuous System Simulation – it is possible to have thousands of state variables without any limitation on the number of entities. For Markov Simulation models, the practical limit is a few or say ten dimensions of the state space, provided the studied number of entities is really small. (The only exception is when the entities do not interact so the 'special case' described in Sections 3.2 and 5.3.1 can be used – but at the cost of a large number of replications for each experiment.) The limitation in size for Markov models also tends to result in the modeller over-simplifying the model structure. The Poisson Simulation model is not only much easier to build, but also gives a more transparent and comprehensive model.

Assigning values to the parameters: Parameterisation is perhaps the most problematic part of Markov modelling. What are natural parameters of the real system – e.g. time constants, fractions, fertility, interest rate, etc. – are also distinct parameters in a Poisson Simulation model. In a Markov model, however, these parameters become part of all the non-zero transition probabilities. The need to assign values to all the transition elements often makes the Markov model over-parameterised.

The aspects execution time, finding a proper step-size, flexibility to change the model and graphical representation of the model are also worthy of comment.

Execution time: A Poisson Simulation model is *also aggregated over time* so that many entities (for each of several flows) can be transferred during a comparatively long time-step. This contrasts with the Bernoullian Markov Simulation, where the time-step usually has to be much smaller, so usually zero or at most one event may occur during this short time-step. The exponential Markov approach is considerably faster than the Bernoullian Markov approach. However, Poisson Simulation executes faster than Markov Simulation.

In a test in this paper, *10 000* replications of an epidemic SIIIR model with a population of 1001 individuals was executed within *20* seconds on an ordinary PC. Such a system is not even possible to build as a Markov model.

Finding a proper step-size: An important property in Poisson Simulation is that the model is preserved when a sufficiently small time-step is changed. This property is valuable for finding an appropriate step-size – small enough to properly handle the dynamics of the model but not smaller than necessary. For Markov Simulation based on Bernoullian and Poisson time handling this is more problematic since the values of the transition probabilities are all

functions of the step-size, so any change in it affects the whole model. There are many examples in the literature of Markov models where a sufficiently small step-size was not used. An additional problem to lack of precision is then that the results become functions of the step-size. However, these problems are eliminated when the exponential time handling approach to Markov modelling is used.

Flexibility to modify the model: It is important that the model is flexible enough to incorporate changes with a reasonable effort. Changing the model is considerably easier in Poisson Simulation than in Markov Simulation because of transparency, size, parameterisation, etc.

Graphical representation: For even the *conceptualisation* of the system under study, some form of graphical representation is needed to sketch a model from the underlying concepts, to structure our knowledge, for overview, and for communication with other people within or outside the project. For documentation and presentation of the study too, it is crucial to describe the model in a comprehensive way.

For Markov models the state transition diagram is sometimes used for pedagogic reasons. In this diagram every state and possible transition is depicted, so it is far too complicated and detailed to be used for all but the simplest models. Furthermore, such a diagram is not suited to reveal the dynamic structure of the studied system. For Poisson Simulation the Forrester diagram is a powerful tool for conceptualisation, structuring the knowledge, overview, communication, documentation, presentation and, in some simulation packages, even for direct implementation of the diagram into an executable computer programme. In all these respects the Forrester diagram is widely superior to the state transition diagram.

Poisson Simulation also outperforms Markov Simulation in other aspects, such as not having to worry about the maximum size of a growing population, etc.

For every issue compared in this study, Poisson Simulation had the advantage. The overall conclusion is that Poisson Simulation widely outperforms Markov Simulation. This study did not find any non-trivial case where Markov Simulation is smaller, easier, faster or in any other respect superior to Poisson Simulation.

7.2. The full power of Poisson Simulation

The first five sections of this paper intentionally restricted the application of Poisson Simulation and Markov Simulation to stationary processes. However, Poisson Simulation covers a much wider field than Markov processes. A presentation of the full power of Poisson Simulation is outside the scope of this paper, but to give an idea of its potential a few examples of more complex and realistic modelling can be mentioned:

Poisson Simulation can easily handle non-stationary models, which for Markov Simulation would require reconstruction of the transition matrix for each time-step.

Poisson Simulation can also cope with semi-Markov models, where one can have an arbitrary probability distribution of the time a process remains in a stage. This can e.g. be efficiently implemented with structures of compartments in series/parallel/feedback or with multi-entrance circular buffers.

Furthermore, more general stochastic processes can be handled with Poisson Simulation. For example correlation between flows is easily constructed. Poisson Simulation can also utilise mathematical, time-triggered, logical and table look-up functions. In addition, it can use dynamic time delays of specified order and static delays that e.g. can handle pure time delays. Random number generators of different kinds can also be used in a Poisson Simulation model.

7.3. Some further remarks

Even when Markov theory is used for mathematical analysis, various kinds of assumptions and simplifications are often used. It is then important to test the consequences of such assumptions or simplifications. This can be swiftly done in Poisson Simulation – especially when Markov Simulation is not a possible option.

Furthermore, when Markov theory is not sufficient, the embedded differential equation model is sometimes used. This embedded model – which is the same for a Poisson Simulation and a Markov model – is most easily acquired by stripping the *Po[]* parts of the Poisson Simulation model. This means that all statistical information is lost, but even worse, incorrect results may be obtained (see Example 6 above). In fact, of the six models in Figure 4, using the embedded deterministic model gives more or less biased results for 4b) Logistic model, 4c) Lotka-Volterra model, 4d) Lanchester's model of warfare (even though this model is perfectly linear with constant coefficients), 4e) Epidemic SIR model, and 4f) Queue model. With Poisson Simulation this can easily be tested – or better still the embedded model is not needed when one can rely on Poisson Simulation and also preserve the stochastics.

Although using Markov models for simulation is a problematic choice, Markov theory and Poisson Simulation constitute a strong combination to theoretical insight and realistic experiments.

Acknowledgments

The author wishes to thank Professor Mikael Sternad of Uppsala University for many valuable discussions and proposals.

Appendix. Programming a Poisson Simulation model

A Poisson Simulation model of an epidemic SIIIR model (see Example 6 in Section 5.3.2) is given in Pascal code [44] below. Pascal is chosen only because of its readability. The model could equally well be coded in e.g. Fortran, C, BASIC or Java. The included function for generation of Poisson distributed random numbers is given in [35]. This generator can handle Poisson-arguments up to *theta=88*. (Otherwise there is overflow in the statement: M:=exp(theta);.) For more advanced generators see e.g. [34,35].

```
Program SIIIR;
type double=real;                          (* Gives double precision. *)
var dt, time: real;
   S, I1, I2, I3, R, si1, i1i2 ,i2i3, i3r, I: real;   (* si1 is the flow between S and I1 etc. and I=I1+I2+I3. *)
   p, T: real;                             (* The model parameters. *)
Label START, AGAIN;

Function POISSON(theta: real): integer;    (* Poisson random number generator. *)
var X: integer;
   m: real;
begin
  X:=-1;
  M:=Exp(theta);
  Repeat
    X:=X+1;
    M:=M*Random;                           (* Random is a call to the built-in *)
  until M<1;                               (* uniform U(0,1)-generator in Pascal. *)
  POISSON:=X;
end; (* Poisson *)

begin
 Randomize;                                (* Randomises seed so each simulation is a new experiment. *)
 dt:=0.05; Time:=0;
 p:=0.0003; T:=4;
 S:=1000; I1:=1; I2:=0; I3:=0; R:=0;       (* Initial values of the states. *)
 Writeln('  Time     S     I     R');       (* Write headline. *)
 Goto START;

AGAIN:
 S:= S - si1;                              (* FIRST CALCULATE THE STATE VARIABLES, *)
 I1:=I1 + si1 - i1i2;
 I2:=I2 + i1i2 - i2i3;
 I3:=I3 + i2i3 - i3r;
 R:=R  + i3r;
START:
 I:=I1+I2+I3;                              (* THEN THE AUXILIARY QUANTITY, *)
 si1:= POISSON(dt*p*S*I);                  (* AND LAST THE FLOWS. *)
 i1i2:= POISSON(dt*I1/(T/3));
 i2i3:= POISSON(dt*I2/(T/3));
 i3r:= POISSON(dt*I3/(T/3));

 Writeln(Time:6:2, S:6:0, I:6:0, R:6:0);
 Time:=Time+dt;
 If  I > 0 then Goto AGAIN;                (* A better breaking criterion than:          *)
end.                                       (* If  Time < large value then Goto AGAIN;. *)
```

Comment: In the execution time test (see Section 5.3.5) a loop over the model part was executed *10 000* times, and the screen output lines "Writeln…;" were eliminated.

References

[1] A.A. Markov, Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain, The Notes of the Imperial Academy of Sciences of St. Petersburg VIII Series, Physio-Mathematical College, Vol. XXII, No 9, December 5, 1907.

[2] P. Brémaud, Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues, Springer-Verlag, NY, 2001.

[3] J.R. Clymer, Systems Analysis Using Simulation and Markov Models, Prentice Hall International Series in Industrial & Systems Engineering. NJ, 1991.

[4] C.J. Mode, C.K. Sleeman, Stochastic Processes in Epidemiology: HIV/AIDS, Other Infectious Diseases and Computers, World Scientific Publishing Co., Singapore, 2000.

[5] R. Bäuerie, Markov cohort simulation study reveals evidence for sex-based risk difference in intensive care unit patients, The Am. J. of Surgery, vol. 179, issue 3 (2009) 207-211.

[6] H.R. Mannan, M. Knuiman, M. Hobbs, Using a Markov simulation model to assess the impact of changing trends in coronary heart disease on requirements for coronary artery revascularization procedures in Western Australia, MBC Cardiovascular Disorders !0:2 (2010), http://www.biomedcentral.com/1471-2261/10/2.

[7] K. Canfell, R. Barnabas, J. Patnick, V. Beral, The predicted effect of changes in cervical screening practice in the UK: results from a modelling study, Br J Cancer. 2004 August 2; 91(3): (2004) 530–536.

[8] J.R. Beck and S.G. Pauker, The Markov Process in Medical Prognosis, Med. Decis. Making, 3, No.4, (1983) 419-458.

[9] L. Gustafsson, Poisson Simulation – A method for generating stochastic variations in Continuous System Simulation, Simulation 74:5 (2000) 264-274.

[10] D.L. Gillespie, Approximate accelerated stochastic simulation of chemically reacting systems, J. Chem. Phys. 115 (2001) 1716-1733.

[11] J.R. Emshoff and R.L. Sisson, Design and use of computer simulation models, Macmillan Publ. Co, NY, 1970.

[12] A.M. Law, W.D. Kelton, Simulation Modeling and Analysis, McGraw-Hill Inc, NY, 1991.

[13] W. Kreutzer, System Simulation: Programming Styles and Languages, Addison-Wesley Publishing Company Inc, Sydney, 1986.

[14] J.W. Haefner, Modelling Biological Systems: Principles and Applications, Chapman & Hall, International Thomson Publishing, NY, 1996.

[15] L. Gustafsson and M. Sternad, Consistent Micro, Macro and State-based population modelling, Math. Biosci. 225 (2010) 94-107.

[16 J.L. Devore, Probability and Statistics for Engineering and Sciences, Sixth edition, Thomson Learning, Inc., Toronto, Canada, 2004.

[17] L. Kleinrock, Queueing Systems. Vol. 1: Theory, John Wiley & Sons Inc, NY, 1975.

[18] Ehrenfest P and Ehrenfest T, Über zwei bekannte Einwände gegen das Boltzmannsche H-Theorem, Physikalische Zeitschrift 8 (1907) 311-413.

[19] L. Gustafsson and M. Sternad, Bringing consistency to simulation of population models – Poisson Simulation as a bridge between micro and macro simulation. Math. Biosci. 209 (2007) 361-385.

[20] Powersim Corporation, Powersim 2.5 User's Guide, Powersim Press, Powersim Corporation, 1175 Herndon Parkway, suite 600, Herndon, VA 22170, USA, 1996.

[21] MM High Performance Systems Inc., Getting Started with STELLA v 6.0.

[22] Vensim: Ventana Systems, Inc. User's Guide, Version 5, 2002.

[23] J.W. Forrester, Industrial Dynamics, Cambridge, MIT Press, MA, 1961.

[24] B. Hannon, M. Ruth, Modeling Dynamic Biological Systems, Springer Verlag, NY, 1997.

[25] Pugh A. III, DYNAMO User's Manual, Cambridge, MA., 1963.

[26] M. Braun, Differential Equations and Their Applications, Springer-Verlag, NY, 1993.

[27] L. Gustafsson, Studying dynamic and stochastic systems using Poisson Simulation, in: H. Liljenström and U. Svedin, (Eds.), Micro – Meso – Macro: Addressing Complex Systems Couplings, World Scientific Publishing Company, Singapore, 2004, pp. 131-170.

[28] V. Volterra, 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.

[29] F.W. Lanchester, Aircraft in Warfare, the Dawn of the Fourth Arm, Tiptree, Constable and Co., Ltd., London, 1916.

[30] W.O. Kermack, A.G. McKendrick, Contributions to the mathematical theory of epidemics, Proc. Royal Soc. A 115 (1927) 700-721.

[31] N.T.J. Bailey, The Mathematical Theory of Infectious Diseases and its Applications, Griffin, London, 1975.

[32] J. Giesecke, Modern Infectious Disease Epidemiology, Oxford University Press Inc, NY, 1994.

[33] L. Gustafsson, Poisson Simulation as an extension of Continuous System Simulation for the modeling of queuing systems, Simulation 79:9 (2003) 528-541.

[34] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in Pascal – The Art of Scientific Computing, Cambridge University Press, Cambridge, UK, 1989. (The book is also available in FORTRAN and C.)

[35] P. Bratley, B.L. Fox, L.E. Schrage, A Guide to Simulation, Springer-Verlag, NY, 1983.

[36] G.S. Fishman, Principles of Discrete Event Simulation, John Wiley & Sons, NY, 1978.

[37] L. Gustafsson, Tools for Statistical Handling of Poisson Simulation: Documentation of StocRes and ParmEst. Dept. of Biometry and Engineering, Swedish University of Agricultural Sciences, Report – biometry 2004:01, (2004), 1-29.

[38] T. Hedqvist, Wanda for MATLAB. Methods and Tools for Statistical Handling of Poisson Simulation, Uppsala University, ISSN: 1401-5749, UPTEC IT0422, Uppsala, Sweden, (2004), 1-60.

[39] Using MATLAB, The MathWorks, Inc, Natick, MA., 2002. www.mathworks.com.

[40] M.S. Hamilton, Estimating length and orders of delays in system dynamics models, in: J. Randers, (Ed.), Elements of the System Dynamics Method, The MIT Press, MA, 1980, pp. 162-183.

[41] D.T. Gillespie, A general method for numerically simulating the stochastic time evolution of coupled chemical reactions, *J. Comp. Phys*. 22 (1976) 403-434.

[42] D.E. Knuth, The Art of Computer Programming, vol.1: Fundamental Algorithms, Addison-Wesley, 1968.

[43] E.S. Page, L.B. Wilson, Information Representation and Manipulation in a Computer. Cambridge University Press, Cambridge, UK, 1973.

[44] Turbo Pascal User's Guide. Version 6.0. Borland International, CA, 1990.