10

Identification and Generation of Realistic Input Sequences for Stochastic Simulation with Markov Processes

Carl Sandrock
University of Pretoria
South Africa

1. Introduction

A simulation is a reproduction under controlled circumstances of a real-life situation. The term has recently become strongly associated with numeric evaluation of a computer model due to the increase in speed and availability of computing resources. This increase in speed has led to much interest in stochastic simulation, where processes with elements that are random are simulated. An attractive branch of stochastic simulation termed Monte Carlo simulation uses deterministic models driven by stochastic input sequences to approximate the distributions of output variables over time. To do this, a good deterministic model of the process is needed in addition to a good method of generating realistic input sequences.
Correct input sequences are a prerequisite for reliable results from stochastic simulation. To generate them, the modeller must either generate input sequences by hand, develop a model based on intuition or understanding of the process, or use existing data. Generating input sequences by hand is a tedious and error-prone process and intuition is not a particularly verifiable source of information. This means that data-driven model development has been gaining favour steadily as data becomes more accessible.
This chapter covers three aspects of input signal generation: First, the basic theory of Markov processes and hidden Markov models is reviewed with a view on using them as generating processes for input models. Second, signal segmentation is introduced. This is the first step in identifying state transition probabilities for discrete Markov processes. In this part, novel work done on the identification of state transitions using multi-objective optimisation is introduced and ideas for future research are posed. Third, the problem of estimating state transition probabilities from the segmented signals is discussed, touching on the issues that modellers should be aware of.
Markov processes have featured strongly in stochastic sequence identification and generation for many years, but some of the related problems are still active research fields.

2. Markov Processes

2.1 Discrete-time Markov Processes

A stochastic process with state space S has the Markov property if the current state completely determines the probability of the following state. A sequence $X_1, X_2, \ldots, X_t$ having this property is known as a Markov chain.

Stated mathematically, a Markov chain obeys the property

$$\Pr(X_{t+1} = j | X_t = i) = \Pr(X_{m+1} = j | X_m = i) = p_{ij} \tag{1}$$

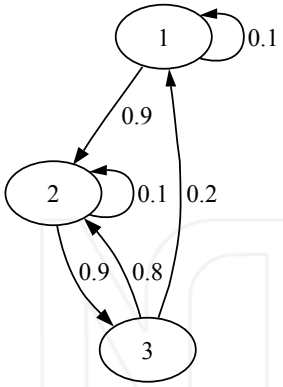
in words, the probability that the next state will be equal to j given that the current state is i is only dependant on the current state.
When S is a countable set, the state transition probabilities can be written as a state transition matrix P as shown for a 3 state process in equation 2

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{2}$$

The probability of remaining within the state space must be unity, hence we may write

$$\sum_{j \in S} p_{ij} = 1 \; \forall \; i \in S. \tag{3}$$

Matrices with this property as well as the common-sense property that $0 \leq p_{ij} \leq 1$ (as they are probabilities) are called stochastic matrices.
The orientation of P is not unique. The arrangement with the current state in the rows and next state in the columns is known as a right transition matrix. The transpose arrangement has also been used (see for instance Bhar & Hamori (2004)) and is then described as a left transition matrix. Modern engineering usage leans toward the description used in this work.
A common way of visualising a Markov process with countable state space is by showing a directed graph with the states in the nodes and the transition probabilities on the edges as shown in Figure 1. In these representations, it is customary to neglect edges with zero probabilities.



$$P = \begin{bmatrix} 0.1 & 0.9 & 0 \\ 0 & 0.1 & 0.9 \\ 0.2 & 0.8 & 0 \end{bmatrix}$$

Fig. 1. Markov process represented by a transition matrix and a graph

The state transition probabilities sufficiently describe the time dependence of the process, but the initial state can not be determined using the transition probabilities alone. The probability of the process starting out in a given state i is denoted $\pi_i, i \in S$, and the vector of initial state probabilities is called $\boldsymbol{\pi}$. It can be seen that a discrete time Markov process is completely described by its state space S, its state transition matrix P and its initial state probability vector $\boldsymbol{\pi}$. If S is countable and has N elements, $N^2 + N$ probabilities have to be known to fully characterise the process. For convenience, the model is written $\lambda = (P, \boldsymbol{\pi})$.

2.2 Hidden Markov Models

It is not always possible to observe the state (in the state space S) of a Markov process directly. It may, however, be possible to make observations from an observation space O related to the state of the process. If the probability of making a particular observation is only related to the current state of the process, the process may be described by a hidden Markov model (HMM). What is "hidden" in this case are the true values of the Markov process states.
Figure 2 shows the situation graphically. If the Markov process is in state 1, there are even odds that observation 2 or 3 will be made. In state 2, only observation 2 is made and state 3 is associated with observation 1 80% of the time and observation 2 20% of the time.

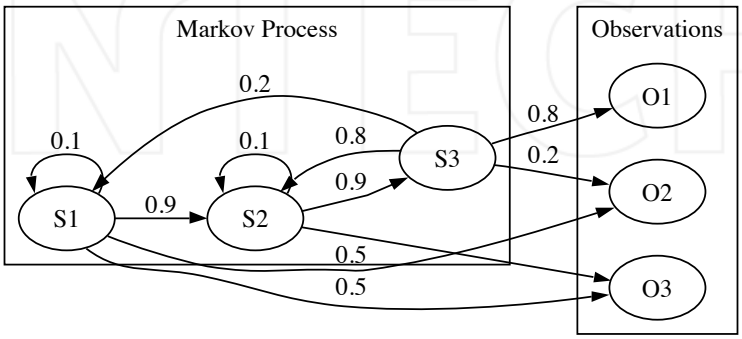


Fig. 2. Graphical representation of a finite hidden Markov model

The probabilities associated with a observation k being made when the process is in state i may be written b_{ik} and can be arranged into an observation probability matrix B in a similar fashion as was previously done for P. One difference is that, while P was an $N \times N$ square matrix, B will have N rows associated with the N states of the Markov process and M columns associated with the observation space. An HMM as described here can therefore be characterised by the same $N^2 + N$ probabilities describing the Markov process in addition to MN observation probabilities. The model description can be abbreviated to $\lambda = (P, B, \boldsymbol{\pi})$.
There are three main problems associated with HMMs (Gamerman & Lopes, 2006):

- What is the probability of generating a specific output sequence from a particular model?
- What is the most likely sequence of states that would lead to a particular output sequence?
- How do we identify the model that corresponds to a given output sequence?

2.3 Continuous-time Markov Processes

The description of discrete-time Markov processes assumed that the transition time was known or unimportant and one could imagine simulating the process by picking a state i and moving to the next state with probability p_{ij}. One shortcoming of such a description is that there is no information about the amount of time a process remains in a particular state before moving to the next (or possibly same) state.

Continuous-time Markov processes encode the transition probabilities as transition rates q_{ij} (forming a Q matrix as p_{ij} formed a P matrix), such that

$$\Pr(X(t+\Delta t)=j|X(t)=i)=\begin{cases}1-q_{ii}\Delta t+o(\Delta t) & \text{for } i=j\\ q_{ij}\Delta t+o(\Delta t) & \text{otherwise}\end{cases} \tag{4}$$

The idea is that, having changed to state i, the probability of changing to state j increases at a linear rate.

3. Segmentation

Segmentation aims to approximate an input signal of length N by $n < N$ events. Segmentation can be used as a compression measure, as a method of smoothing the data or to investigate underlying structure in the signal. Keogh et al. (1993) gives a good review of several segmentation algorithms applied to EKG time series. Segmentation is also used in a different sense in fields like speech recognition to mean identification of transitions in the data without explicit fitting of a curve or reduction of data. This usage will not be discussed here.
The most popular event type is straight line or piecewise-linear segmentation. However, more interesting functions like general polynomials (Arora & Khot, 2003) have been proposed.

3.1 Objectives

A good segmentation algorithm:

1. minimises the error of the segmented description (or at least satisfies some upper bound on the error),
2. uses the simplest description possible for the data (which may be in terms of the number or complexity of the identified segments) and
3. is efficient in computer time and space requirements.

If the algorithm is to be used on-line to segment signals as they are read, it is also beneficial if the algorithm can incorporate new data efficiently.
Some of these objectives are contradictory – a more complex description will almost always allow a lower segmentation error than a simpler one, for instance. Also, it is always possible to segment with zero error by simply dividing the segmented data at every single sample point, so direct minimisation of the fitting error is clearly insufficient on its own.
The next sections summarise commonly employed algorithms. The details are taken largely from Keogh et al. (1993), with some reinterpretation to fit within the structure of this document. Note that the algorithms have been significantly reworked.

3.2 Top-down methods

These methods can also be described as subdivision methods and feature a recursive subdivision of the signal that stops when an error measure has been reduced below a threshold. The algorithm described in Algorithm 1 lends itself to optimisation by dynamic programming, as the optimal subdivisions of smaller sequences can be stored as partial solutions to the larger problem. The Douglas-Peuker algorithm (Douglas & Peucker, 1973) is also an example of a top-down algorithm, although it does not search for optimal breaks recursively – it simply uses the node with the maximum perpendicular distance from the line as the break point.

Algorithm 1 Top-down algorithm

```
function TOPDOWN(T, ε)
    if approximationerror(T) < ε then
        return approximate(T)
    else
        N ← length(T)
        b ← min_i splitcost(T, i)                    ▷ Find best split point
        return topdown(T[1 ... b]) + topdown(T[b + 1 ... N])
    end if
end function
```

3.3 Bottom-up methods

Bottom-up or composition methods start with segments between all data points and merge similar segments until there are no segment pairs to merge without violating an error measure.

Algorithm 2 Bottom-up algorithm

```
function BOTTOMUP(T, ε)
    N ← length(N)
    for i ∈ (1 ... N − 1) do                  ▷ Start with lines between all points
        segments.append(segment(T[i ... i + 1]))
    end for
    for i ∈ 1 ... length(segments)-1 do       ▷ Find cost of merging each pair
        c(i) ←error(merge(segments[i],segments[i + 1]))
    end for
    while min(c) < ε do
        i ← minindex(c)                       ▷ Find "cheapest" pair to merge
        segments[i] ← merge(segments[i],segments[i + 1])   ▷ Merge them
        delete(segments[i + 1])                            ▷ Update records
        c(i) ←error(merge(segments[i],segments[i + 1]);
        c(i − 1) ← error(merge(segments[i − 1], segments[i]));
    end while
end function
```

Algorithm 2 shows a sample algorithm for a bottom-up method.
If properly implemented, both bottom-up and top-down methods should give similar results.

3.4 Methods employing sliding windows

Sliding window or incremental methods process the signal to be segmented sequentially, in one pass. This means that they can be employed on-line, in contrast to the recursive methods discussed before, which require the entire data set to be loaded into memory before being started.
Algorithm 3 shows a possible sliding window algorithm.

3.5 Optimisation-based methods

Any one of the objectives mentioned in Section 3.1 can be rewritten as an objective function for an optimisation algorithm. This objective function could then be minimised by choosing

Algorithm 3 Sliding window algorithm

```
function SLIDINGWINDOW(T, ε)
    a ← 1
    segments ← ∅
    while b < N do
        b ← a + 1
        while error(T[a...b]) < ε do
            b ← b + 1
        end while
        segments.append(T[a...b − 1])
        a ← b
    end while
    return segments
end function
```

the number of parameters, the number of segments and the parameter values for each of the identified segments. Application of this reasoning can be seen in the direct fitting of line segments to data described in Cantoni (1971), which leads to a direct analytical solution via the pseudo-inverse, or in where numerical optimisation is employed to fit more complicated segmentation functions.

A common thread in the optimisation-based methods is that the number of line segments must be known in advance. This is required when using derivative-based optimisation, as the number of design variables fixes the dimensions of the derivative and the current position in the design space. This is a disadvantage when compared to the previous methods, which would automatically fit varying numbers of segments given different data set. Recall, however, that these methods would terminate when a certain error bound had been met, and that this bound had to be set in advance. When the error is reduced using optimisation, this bound is not required.

Another, more significant benefit of using optimisation rather than the direct methods, is that it enables a more general description of a segment to be used with very little additional effort beyond deciding on the parameters of the description. While it is clear how to approach subdivision for line segments, it is not as simple to adjust the algorithms for other functions (Waibel & Lee, 1990)

3.6 Multi-objective optimisation

The trade-off between accuracy and generality of a fit would traditionally be decided by the designer of an algorithm. Perhaps some noise reduction would be done before identifying events, or constraints on the fitting functions would be enforced to avoid over fitting (Arora & Khot, 2003; Punskaya et al., 2002). One could specify an acceptable error bound before segmentation or one could specify a number of segments.

Multi-objective optimisation provides a different approach. All the objective function values are evaluated and a solution is retained if it is better in any way than all of the solutions already encountered. Such solutions are called Pareto optimal or nondominated solutions (Steuer, 1986). The result of such an optimisation algorithm is a *list* of Pareto optimal solutions, or more properly an approximation of the Pareto front. This list is most commonly called the archive.

Evolutionary algorithms are a natural fit for multi-objective optimisation, as they are already population based. Genetic algorithms in particular have enjoyed popularity (Deb & Kalyanmoy, 2001). Recent work in Particle Swarm Optimisation has rekindled interest in using it for multi-objective optimisation.

3.6.1 Application

As an example of the results obtainable using multi-objective optimisation, the MOPSO-CD (Multi-Objective Particle Swarm Optimisation with Crowding Distance) algorithm proposed by Raquel & Naval (2005)was used to fit a first-order response prototype to input signals. The algorithm is a modification of Particle Swarm Optimisation that adds an archive of nondominated solutions and uses a crowding distance measure to prevent many similar Pareto optimal solutions from being retained in the archive.
A problem description based on a prototypical first order response was used in this study. Figure 3 shows the prototype function.

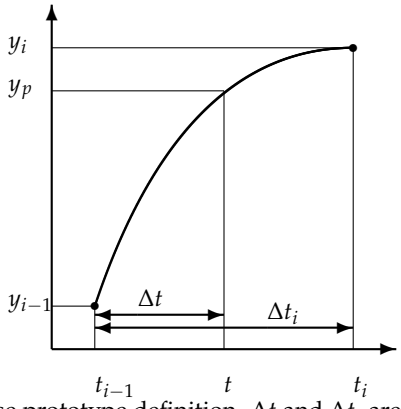


Fig. 3. First order response prototype definition. Δt and Δt_i are the times of the interpolation time and end point time relative to the prototype start.

Our goal is to find a sequence of prototypes that fits the sequence of events. We wish to fit the entire data set, so the first and last times are to coincide with the first and last times in the data set. Therefore, given that we are fitting N prototypes, we seek to find $N-1$ transition times and N parameter value sets.
A few key decisions were made to ease optimisation. Firstly, a linear term was added to the exponential response to ensure that the prototype interpolates through the initial (x_{i-1}, y_{i-1}) and final (x_i, y_i) points. This did not add any parameters to the description. The predicted value for the prototype at a given time t is shown in equation 5:

$$y_p = y_{i-1} + y_i \left(1 - \underbrace{e^{\Delta t/\tau_i}}_{\text{exponential}} + \underbrace{\frac{\Delta t}{\Delta t_i} e^{\Delta t_i/\tau_i}}_{\text{linear}} \right) \tag{5}$$

Secondly, the optimisation parameters were chosen to reduce coupling in the problem parameters by using absolute times for each starting point and constraining these times to be sequential rather than time differences constrained to be positive. This reduced the effect of any one starting point on the error produced by the remaining fit functions.

3.6.2 Objective functions

Two objectives were defined: the RMS error of the fit over all the prototypes and the "complexity" of the fit, which was calculated as

$$c = \sum_{i}^{N} \frac{1}{\tau_i}. \tag{6}$$

This complexity measure works due to the addition of the linear correction term, which dominates for large τ, meaning that as τ increases, one sees more of the linear behaviour and less of the exponential. Therefore, larger c corresponds to greater curvature of the fitting prototypes.

3.6.3 Prototype to event mapping

Each sequence of prototypes identified was mapped back to a sequence of event types by using the following heuristics:

- If the difference between the start and end values is less than a cut-off value ϵ_c, the prototype is taken to represent a constant event.
- If the time constant is larger than a cut-off time constant τ_c, it is taken as a ramp.
- If neither of these holds, the prototype is a first order response.

The values of ϵ_c and τ_c are problem-dependant and should be chosen to represent an insignificant change in y and a large time constant (in the chosen time units) respectively.

3.6.4 Results

To illustrate the type of result that is obtained using the technique, we show the results on a a signal consisting of 6 events, attempting to fit 4 events. Figure 4 shows the evolution of the Pareto front in terms of fit complexity and RMS error for different numbers of iterations.
It should be noted that, although the front seems to be converging, population based multi-objective optimisation algorithms can not guarantee convergence with a finite archive. This is due to the pruning that must inevitably be done when the archive is full. Figure 4 does, however, show that the front has not receded.

3.7 Optimisation with variable numbers of events

The optimisation methods discussed so far have a significant disadvantage: it is not possible for them to choose the "optimal" number of segments as one of the design variables, as their design space needs to have constant dimension. It is, however, possible to use genetic algorithms (GAs) for this purpose, by using a crossover operator allowing varying chromosome lengths. One such operator is the simple "cut and splice" operator, which chooses a crossover point on the chromosome of each parent independently before exchanging material. The application of multi-objective GAs with varying chromosome lengths may yield the first fully-automated optimisation for fitting events, as it allows the number of events to be included in the objective set.
Finding the Pareto-optimal set of fits in this way will enable much richer analysis of time series.

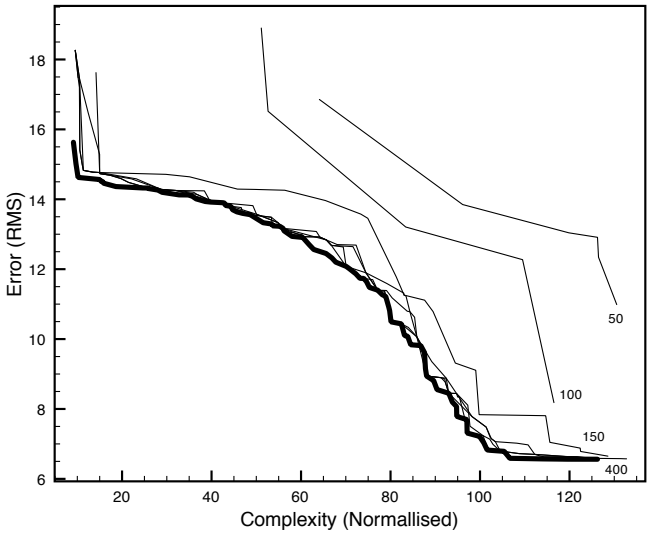


Fig. 4. Evolution of Pareto Front for 6 events being fit by 4 events.

4. Estimating state transition probabilities

The most direct method of estimating the state transition probabilities of a Markov process is to count the number of transitions in an input signal. This strategy has some problems:

1. Certain transitions may not occur in the input signal, so that these transitions will never be simulated by the identified model
2. Segmentation of the input signal may bias the event types or transitions – if a certain event is more often fit by the segmentation algorithm, that event will be over-represented in the transition matrix.

If transitions between some events are very rare, it may be advisable to introduce a small artificial probability into the matrix to ensure that the event has a chance of getting generated during the simulation. This is especially true if the repercussions of a certain event combination are significant.
Segmentation bias can be combated by generating a large unbiased test set and testing the segmentation algorithm on it. If a segmentation bias is detected, the transition probabilities can be modified to take these into account.

5. Conclusions

Markov processes provide a simple yet powerful method for generating realistic input sequences. The theory in this chapter should be enough for the interested reader to get started in this fascinating field and should enable simulation of a system with little additional reading required. The techniques for segmenting input signals and identifying model parameters are applicable to a broad range of fields and includes novel work on the employment of multi-objective optimisation to signal segmentation and estimation.

The most interesting future work suggested by this research is the use of variable-length multi-objective GAs to segment signals.

6. References

Arora, S. & Khot, S. (2003). Fitting algebraic curves to noisy data, *Special Issue on STOC 2002* **67**(2): 325âĂŞ340. http://portal.acm.org/citation.cfm?id=963875.963882.

Bhar, R. & Hamori, S. (2004). *Hidden Markov Models: Applications to Financial Economics*, Advanced studies in theoretical and applied econometrics, Kluwer Academic Publishers, Boston, Mass.

Cantoni, A. (1971). Optimal curve fitting with piecewise linear functions, *Computers, IEEE Transactions on* **C-20**(1): 59–67.

Deb, K. & Kalyanmoy, D. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley. http://www.amazon.ca/Multi-Objective-Optimization-using-Evolutionary-Algorithms/dp/047187339X/.

Douglas, D. H. & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, *Cartographica: The International Journal for Geographic Information and Geovisualization* **10**(2): 112–122. http://dx.doi.org/10.3138/FM57-6770-U75U-7727.

Gamerman, D. & Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, 2nd ed edn, Chapman & Hall/CRC, Boca Raton.

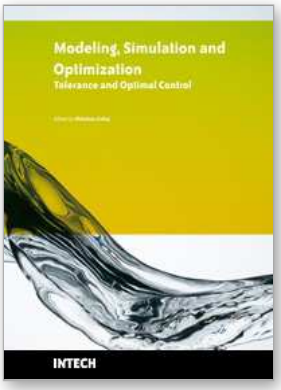
Keogh, E., Chu, S., Hart, D. & Pazzani, M. (1993). Segmenting time series: A survey and novel approach, *IN AN EDITED VOLUME, DATA MINING IN TIME SERIES DATABASES. PUBLISHED BY WORLD SCIENTIFIC* pp. 1—22. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.9924.

Punskaya, E., Andrieu, C., Doucet, A. & Fitzgerald, W. (2002). *Bayesian curve fitting using MCMC with applications to signal segmentation*, IEEE Press. http://citeseer.ist.psu.edu/548064.html.

Raquel, C. R. & Naval, P. C. (2005). An effective use of crowding distance in multiobjective particle swarm optimization, *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, p. 257âĂŞ264. http://portal.acm.org/citation.cfm?id=1068009.1068047.

Steuer, R. E. (1986). *Multiple Criteria Optimization (Probability & Mathematical Statistics)*, John Wiley & Sons Inc. http://www.amazon.ca/Multiple-Criteria-Optimization-Computation-Application/dp/047188846X/.

Waibel, A. & Lee, K. (1990). *Readings in speech recognition*.



Modeling Simulation and Optimization - Tolerance and Optimal Control

Edited by Shkelzen Cakaj

ISBN 978-953-307-056-8
Hard cover, 304 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010

Parametric representation of shapes, mechanical components modeling with 3D visualization techniques using object oriented programming, the well known golden ratio application on vertical and horizontal displacement investigations of the ground surface, spatial modeling and simulating of dynamic continuous fluid flow process, simulation model for waste-water treatment, an interaction of tilt and illumination conditions at flight simulation and errors in taxiing performance, plant layout optimal plot plan, atmospheric modeling for weather prediction, a stochastic search method that explores the solutions for hill climbing process, cellular automata simulations, thyristor switching characteristics simulation, and simulation framework toward bandwidth quantization and measurement, are all topics with appropriate results from different research backgrounds focused on tolerance analysis and optimal control provided in this book.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821