

Solving Stochastic Difference Equations Originated from Continuous-time Threshold AR(1) Models Through Perfect Simulation

M. S. Carvalho and J. N. Corcoran
University of Colorado *

October 16, 2003

Abstract

In this paper we look at a discretization of a continuous-time threshold AR(1) process (CTAR(1)) and show how to apply perfect simulation to find the stationary distribution of the resulting stochastic difference equation. Such method leads to perfect solutions of stochastic difference equations of this kind. Depending on how sensitive the original CTAR(1) process is to the discretization, the method may give an approximate solution to the original continuous-time process. Also, we introduce the phenomenon of decoupling in a perfect simulation algorithm.

1 Introduction

A fundamental assumption of time series modeling is that the value of the series at any given time depends only on its previous values in a deterministic way and on an additional stochastic disturbance. If this dependence is assumed to be linear, we get a traditional and well-studied ARMA (autoregressive moving average) model

$$X_n - \sum_{k=1}^p \phi_k X_{t-k} = Z_n + \sum_{k=1}^q \theta_k Z_{t-k}. \quad (1)$$

Here, the coefficients are real constants and $\{Z_n\}$ is a mean zero white noise process.

Discrete time series, however, are often obtained by observing an underlying continuous-time process at a discrete sequence of observation times. It therefore can be quite natural to model the underlying process driving the observed time series as a continuous time series.

*Postal Address: Department of Applied Mathematics, University of Colorado, Box 526 Boulder CO 80309-0526, USA; email: corcoran@colorado.edu, marcio.carvalho@colorado.edu; phone: 303-492-0685

⁰Keywords: stochastic difference equations, perfect simulation, coupling from the past, exact sampling, perfect sampling, CTARMA, decoupling

AMS Subject classification: 60J10, 65C05, 62F25, 60K30

This point of view is particularly useful for modeling data observed at irregularly spaced time points.

A CARMA process, $\{X(t)\}$, is defined to be the continuous-time analogue of (1) and is given by a stationary solution of the stochastic differential equation

$$D^{(p)}X(t) + \sum_{k=1}^p \phi_k D^{(p-k)}X(t) = \sigma \left[DB(t) + \sum_{k=1}^q \theta_k D^{(k+1)}B(t) \right] + c. \quad (2)$$

Here, $\phi_1, \phi_2, \dots, \phi_p, \theta_1, \theta_2, \dots, \theta_q, \sigma$, and c are constants, $D^{(i)}$ denotes i -fold differentiation with respect to t , and $\{B(t)\}$ is standard Brownian motion.

A continuous-time threshold ARMA process allows coefficients in (2) to depend on X_t . For example, we might write

$$\phi_1(X(t)) = \begin{cases} \phi_1^+ & , \text{ if } X(t) > \tau \\ \phi_1^- & , \text{ if } X(t) < \tau \end{cases}$$

where τ is some *threshold*.

There are only a small number of cases where the stationary distributions for solutions of such models can be written down explicitly. (See, for example, [3],[2],[1].) In the rest of the cases, simulation is a valuable tool for the study of these distributions as well as other properties of CTARMA processes. For purposes of simulation, time discretized models must be considered, effectively turning the defining stochastic differential equations into stochastic difference equations. At this point, one would then introduce a second level of simulational error. In this paper, it is this second level error that we seek to eliminate through the use of a perfect simulation algorithm. In Section 2 we describe CTARMA models and discretization in more detail. In Section 3 we give a brief overview of the principles of perfect simulation and detail the specific perfect simulation algorithm we use which include a ‘‘slice and folding coupler’’ of Corcoran and Schneider [5]. Finally in Sections 4, 5, and 6, we apply a perfect sampling algorithm to a simple CTARMA model and discuss the results of a simulation.

2 CTARMA Models

The CTARMA(p,q) $0 \leq q < p$ process, $\{X(t)\}$, with threshold at τ is defined to be a stationary solution of the p th order linear differential equation

$$D^{(p)}X(t) + \sum_{k=1}^p \phi_k(X(t))D^{(p-k)}X(t) = \sigma(X(t)) \left[DB(t) + \sum_{k=1}^q \theta_k(X(t))D^{(k+1)}B(t) \right] + c(X(t)) \quad (3)$$

where $D^{(i)}$ denotes i -fold differentiation with respect to t , and $\{B(t)\}$ is standard Brownian motion.

The coefficients in (3) vary depending on whether $X(t)$ is above or below a *threshold* τ . That is, for $k = 1, 2, \dots, p$, we have

$$\phi_k(X(t)) = \begin{cases} \phi_k^+ & , \text{ if } X(t) > \tau \\ \phi_k^- & , \text{ if } X(t) < \tau \end{cases} ,$$

for $k = 1, 2, \dots, q$, we have

$$\theta_k(X(t)) = \begin{cases} \theta_k^+ & , \text{ if } X(t) > \tau \\ \theta_k^- & , \text{ if } X(t) < \tau \end{cases} ,$$

and

$$\sigma(X(t)) = \begin{cases} \sigma^+ & , \text{ if } X(t) > \tau \\ \sigma^- & , \text{ if } X(t) < \tau \end{cases} , \quad c(X(t)) = \begin{cases} c^+ & , \text{ if } X(t) > \tau \\ c^- & , \text{ if } X(t) < \tau \end{cases} .$$

Discretization:

For the purpose of simulation, we must approximate (3) with a discrete-time analogue. For example, for the CTAR(1) model

$$dX(t) + \phi X(t)dt = \sigma dB(t),$$

we instead consider the discrete-time process given by

$$X(t + \Delta t) - X(t) + \phi X(t)\Delta t + \sigma(B(t + \Delta t) - B(t)). \quad (4)$$

for $t = t_0, t_1, t_2, \dots$, where $t_0 = 0$, $t_n = t_{n-1} + \Delta t$, for $n = 1, 2, \dots$. So,

$$X(t + \Delta t) = (1 - \phi\Delta t)X(t) + \sigma W(t)$$

where $\{W(t_n)\}_{n=0}^{\infty}$ is a sequence of independent and identically distributed normal random variables with mean zero and variance $\sigma^2\Delta t$. Here, ϕ and σ^2 are threshold dependent.

Of course, since we have strayed from the continuous-time model, we will be simulating only an approximation of the stationary distribution of the original model. This approximation introduces a “first level error”. Standard Monte Carlo simulations introduce a sampling error or, “second level error”. In this paper we use so-called *perfect simulation* techniques in order to completely eliminate the second level error. We will simulate an approximate version of the continuous-time model which will be a perfect simulation of the difference equation model given, for example, by (4).

3 Perfect Simulation

There has been considerable recent work on the development and application of “perfect sampling” algorithms that will enable the simulation of the invariant (or stationary) measure

π of a Markov chain, either exactly (that is, by drawing a random sample known to be from π) or approximately, but with computable order of accuracy. These were sparked by the seminal paper of [14], and several variations and extensions of this idea have appeared since – see [7, 8, 9, 10, 11, 12], and [13]. These ideas have proven effective in areas such as statistical physics, spatial point processes and operations research, where they provide simple and powerful alternatives to methods based on iterating transition laws, for example.

The essential idea of most of these approaches is to find a random epoch $-T$ in the past such that, if we construct sample paths (according to a transition law $P(x, y)$ that is invariant for π) from every point in the state space starting at $-T$, then all paths will have coupled successfully by time zero. The common value of the paths at time zero is a draw from π . Intuitively, it is clear why this result holds with such a random time T . For consider a chain starting at $-\infty$ with the stationary distribution π . At every iteration it maintains the distribution π . But at time $-T$ it must pick *some* value x , and from then on it follows the trajectory from that value. But of course it arrives at the same place at time zero no matter what value x is picked at time $-T$: so the value returned by the algorithm at time zero must itself be a draw from π .

Perfect sampling algorithms can be particularly efficient if the chain is *stochastically monotone* in the sense that paths from lower starting points stay below paths from higher starting points. In this case, one need only couple sample paths from the “top” and “bottom” of the space, as all other paths will be sandwiched in between. It is possible to generalize one step further to monotone chains on an unbounded state space by considering *stochastically dominating* processes to bound the journeys of sample paths.

For this to be practicable we need to ensure that T is indeed finite. [14] show that this occurs for irreducible aperiodic finite space chains, and for a number of stochastically monotone chains possessing maximal and minimal elements. Indeed, [6] show that if the distribution at $-\infty$ is any fixed (or even random) value x_0 , then under fairly standard conditions the value at time zero is still distributed according to π , and this observation is crucial in what follows.

There are several easy-to-read perfect sampling tutorials available, and we refer interested readers to [4]. In this paper we only wish to emphasize that the key idea in the search successively further and further back in time for the so-called *backward coupling time* T requires that one reuse random number streams. That is, if sample paths run forward to time 0 from time -1 using a random number (or random vector) U_{-1} have not coalesced by time 0, then one must go back further, say to time -2 and run paths forward for two steps using a random number U_{-2} and then the **previously used** U_{-1} .

We now describe a particular coupling approach that will be used in our CTARMA sampling algorithms.

3.1 Methods of Coupling

We now briefly describe two methods of coupling that will be used in this paper. Further details may be found in [5].

3.1.1 Slice Coupling

Slice coupling, which should be recognized as distinct from the closely related idea of slice sampling, was developed by Wilson [15] and later expanded upon by Corcoran and Schneider [5]. Both slice coupling (also referred to by Wilson as layered multishift coupling) and slice sampling are based on the idea that one can sample from a distribution by sampling uniformly from the region under the plot of its density function. While both procedures are based on considering horizontal and vertical “slices” of a density of interest, the goal of slice sampling is to simulate or draw a value from the distribution with the given density whereas slice *coupling* assumes we already have a technique for drawing such values. The goal of the latter is to draw a value from each of two different distributions with given densities and potentially have them be the same values for the purpose of coupling sample paths.

We now briefly describe slice coupling and refer the interested reader to [15] and [5] for details.

As mentioned, one way to sample from a distribution is to sample uniformly from the region under the plot of its density function. Suppose we can draw a value x for a random variable X with density function $\pi(x) = c \cdot h(x)$ where the constant of proportionality c is possibly unknown. We then draw a value Y given $X = x$ uniformly over the interval $(0, h(x))$ and finally draw a value X' uniformly from $H(y)$ where $H(y)$ is defined to be the “horizontal slice”

$$H(y) = \{x : h(x) > y\}.$$

It is not difficult to show that X' has density $\pi(x)$. This algorithm is simply a one-iteration stationary version of Gibbs sampling from the density

$$\pi(x, y) \propto I_{\{(x,y):0 \leq y \leq h(x)\}}$$

where $I_{\{\cdot\}}$ is the indicator function. Slice sampling assumes one cannot start with a “draw” from π ; the goal is to get this draw through many Gibbs iterations. If we instead assume that we already have a way to draw values from the distribution with density π , slice coupling is a means for drawing a second value from π , and when used in tandem for two densities, say π_1 and π_2 , it is a way to obtain potentially **common** draws from the corresponding distributions.

A **naïve** slice coupling strategy for drawing potentially common values from two different densities

$$\pi_1(x) = c_1 h_1(x) \quad \text{and} \quad \pi_2(x) = c_2 h_2(x)$$

is to

1. Draw values x_1 and x_2 from π_1 and π_2 , respectively.
2. Draw values y_1 and y_2 uniformly from the intervals $(0, h_1(x_1))$ and $(0, h_2(x_2))$, respectively.
3. Define the corresponding horizontal slices

$$H_1(y_1) = \{x : h_1(x) > y_1\} \quad \text{and} \quad H_2(y_2) = \{x : h_2(x) > y_2\}.$$

4. If $H_1(y_1) \subseteq H_2(y_2)$ ($H_2(y_2) \subseteq H_1(y_1)$), draw a value uniformly on $H_2(y_2)$ ($H_1(y_1)$).
 - If this value is also contained in the other horizontal slice, it is uniform over that slice as well, and is therefore a common value drawn from π_1 and π_2 .
 - If this value is not contained in the other horizontal slice, we have a resulting value from one distribution. Draw the resulting value from the other distribution uniformly from the other horizontal slice.

In step 4 the assumption that one horizontal slice is contained in the other is taken merely for illustration purposes and can be removed when applying different techniques on drawing the uniform random number in that step. As written, the algorithm may not have a high success rate for achieving common draws. Furthermore, for purposes of coupling sample paths of Markov chains, one must take care to run each path in the same way. This includes using the same random numbers to draw both x_1 and x_2 in step 1, using a single $\text{uniform}(0,1)$ random number to produce y_1 and y_2 in step 2, and drawing the final value in part 2 of step 4 in a “meaningful” way (for example, with a series of accept/reject steps).

We refer the reader to [15] and [5] for further information on this technique including shifting and scaling methods for increasing the likelihood of a successful coupling between distributions with different locations and shapes.

3.1.2 The Folding Coupler

The “folding coupler” is a method for drawing a potentially common value from two uniform distributions where the support of one is contained within the support of the other using a **single** random number. This may be extended to other distributions using standard slice sampling ideas. The use of a single random number is a simple alternative to a more standard accept/reject algorithm that requires an a priori unknown number of random numbers. This is especially useful in a perfect simulation setting where one stores and reuses random numbers.

We now define and illustrate the folding coupler for a uniform distribution only. For details on its extension to other distributions, we refer the reader to [5].

Suppose that we want to generate a random variable X that is uniformly distributed on the interval $[a, b]$ and another random variable Y that is uniformly distributed on the interval $[c, d]$ where $a \leq c < d \leq b$. The containment of $[c, d]$ in $[a, b]$ is not a requirement to couple uniform draws in general, but this simple case is sufficient for the algorithms provided in this paper.

The folding coupler, illustrated in Figure 1 proceeds as follows:

1. Draw a value $X \sim \text{Uniform}[a, b]$.
2. If $X \in [c, d]$, accept this also as a uniform draw from $[c, d]$ and set $Y = X$.

3. If $X \notin [c, d]$, map X onto the interval $[c, d]$ by “folding it in” according to the function

$$f(x) = f(x; a, b, c, d) = \begin{cases} \frac{x-a}{(c-a)+(b-d)}(d-c) + c, & \text{if } x \in [a, c) \\ d - \frac{b-x}{(c-a)+(b-d)}(d-c), & \text{if } x \in (d, b] \end{cases} \quad (5)$$

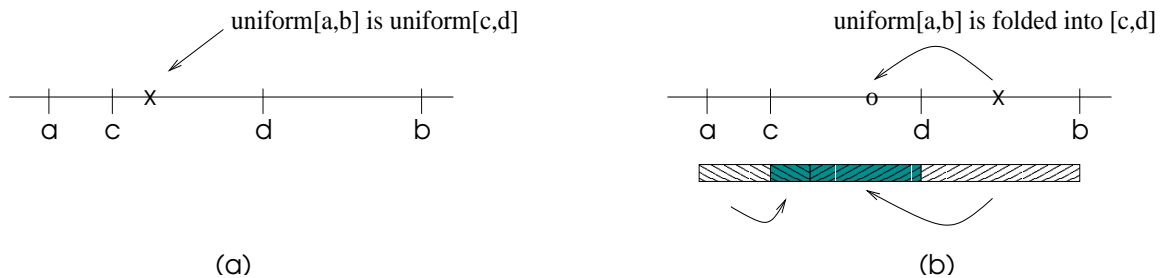
and set $Y = f(X)$.

It is routine to show that the resulting Y value will be uniformly distributed on the interval $[c, d]$ and easy to see that $Y = X$ whenever $X \in [c, d]$.

We have two particular cases holding that will be of interest when we use this technique to generate random transitions for Markov chains:

1. If $c = a$ and $d \leq b$, (The interval for Y is contained in and left aligned with the interval for X .) then this algorithm will result in draws x, y , where $y \leq x$.
2. If $m := (c + d)/2 = (a + b)/2$, (The interval for Y is contained in and centered in the interval for X .) then this algorithm will result in a draws x and y , where $|y - m| \leq |x - m|$.

Figure 1: (a) Step 2 of the Folding Coupler, (b) Step 3 of the Folding Coupler



3.2 An Unbounded State Space

Perfect simulation requires that one be able to achieve a coupling between every potential sample path of the process of interest. For a Markov chain on a finite state it is possible (though perhaps not desirable) to construct sample paths from every point in the state space in order to ensure that we are able to track the progress of any particular sample path of interest. For an infinite, continuous, or even finite but large state space, one can often take advantage of monotonicity properties of the Markov chain (either natural or imposed through a creative reordering of the state space) in order to track a particular sample path by sandwiching it between “extreme” sample paths started at the top and bottom of the space. At first sight, the assumption of finiteness, or at least compact support, of the state

space may appear to be critical to the success of perfect simulation algorithms. However, we may use an idea due to Kendall [11] of an upper bounding random process in order to remove this restriction. David Wilson (personal communication) has also developed a system for doing this.

For the purpose of illustration, we suppose that the state space of our process of interest is $[0, \infty)$ although it may just as easily be $(-\infty, \infty)$. Let our Markov chain of interest be denoted by $\{X_n\}$, and let π denote the stationary distribution for $\{X_n\}$. We wish to identify an “upper bounding process” $\{U_n\}$, with stationary distribution π_U with the following properties.

1. A sample path of $\{U_n\}$ started above a sample path of $\{X_n\}$ and simulated forward with the same random deviates will always stay above $\{X_n\}$.
2. One can simulate (draw) values directly from π_U .
3. The process $\{U_n\}$ is *reversible* in the sense that

$$\pi_U(x)P_U(x, y) = \pi_U(y)P_U(y, x), \quad \text{for all } x, y$$

where P_U denotes the transition law of $\{U_n\}$.

4. One can reverse the process $\{U_n\}$ in practice. This means that if one uses a random deviate R_n (possibly a vector) to generate a value U_{n+1} given U_n , one can use R_n to reproduce U_n starting only with the knowledge of U_{n+1} .

If these conditions are satisfied, assuming monotonicity of the process $\{X_n\}$ we proceed as follows.

Let $n = 0$. Generate a value $U_0 \sim \pi_U$. For $n = 1, 2, 3, \dots$,

1. Generate U_{-n} from U_{-n+1} and store the random deviate R_{-n} that will reproduce this transition forward using P_U from $-n$ to $-n + 1$.
2. At time $-n$ start two sample paths of $\{X_n\}$; one from $X_{-n} \stackrel{\text{set}}{=} 0$ (the “bottom” of the space) and another from $X_{-n} \stackrel{\text{set}}{=} U_{-n}$ (the artificial “top” of the space).
3. Run both of these paths forward to time zero using the stored random deviates $R_{-n}, R_{-n+1}, \dots, R_{-1}$.
4. (a) If both paths have the same value at time zero, stop. We have achieved a coupling and the common value at time zero is a draw from π .
 (b) If the two paths have a different value at time zero, let $n = n + 1$ and go back to step 2.

4 Perfect Simulation of the CTAR(1) Model

4.1 CTAR(1) Model

The CTAR(1) model to be considered here is of the form

$$dX(t) + \phi X(t)dt = \sigma dB(t) \quad (6)$$

with threshold value τ . ϕ and σ assume the positive constant values ϕ^+ and σ^+ when $X(t) \geq \tau$, and the positive constant values ϕ^- and σ^- if $X(t) < \tau$.

For the purpose of illustration, we consider only the case with $\tau = 0$ and $\sigma^+ > \sigma^-$. With minor adjustments, everything that follows also applies for $\tau \neq 0$ and/or $\sigma^- > \sigma^+$.

4.2 Discretization of the Model

Time discretizing (6) with n steps between time $t - 1$ and t , we get

$$X_t = \left(1 - \frac{\phi}{n}\right)X_{t-\frac{1}{n}} + \sigma(B_t - B_{t-\frac{1}{n}})$$

Letting Y_k be $X_{\frac{k}{n}}$, we arrive at the following *stochastic difference equation*,

$$Y_k = \begin{cases} \left(1 - \frac{\phi^+}{n}\right)Y_{k-1} + W_{k-1}^+ & , \text{ if } Y_{k-1} \geq 0 \\ \left(1 - \frac{\phi^-}{n}\right)Y_{k-1} + W_{k-1}^- & , \text{ if } Y_{k-1} < 0 \end{cases}$$

where W_k^+ is a normal random variable with mean zero and variance $\frac{(\sigma^+)^2}{n}$ and W_k^- is a normal random variable with mean zero and variance $\frac{(\sigma^-)^2}{n}$.

4.3 Moving Sample Paths of $\{Y_k\}$ Forward

Given the value of Y_k , one can easily compute the value of Y_{k+1} . But since our goal is to use slice coupling to couple paths, we will need to generate our normal random variables (namely W_k^+ and W_k^-) through a slice sampling procedure.

At each time step k , we generate a vector (U_k, V_k, Z_k) , where

- U_k is a uniform (0,1) random variable
- V_k is a uniform (0,1) random variable
- Z_k is a normal random variable with mean zero and variance $\frac{(\sigma^+)^2}{n}$

Let f_{σ^+} and f_{σ^-} denote, respectively, the density functions for mean zero normal distributions with variances $(\sigma^+)^2/n$ and $(\sigma^-)^2/n$.

$$\boxed{\mathbf{Y}_{k+1} | \mathbf{Y}_k, \mathbf{Y}_k \geq \mathbf{0}}$$

To generate Y_{k+1} given $Y_k \geq 0$, we generate W_k^+ as follows.

1. Use Z_k as the location of a vertical slice under the normal (f_{σ^+}) density. That is, define a vertical slice from the point $(Z_k, 0)$ to the point $(Z_k, f_{\sigma^+}(Z_k))$.
2. Define a horizontal slice across the normal (f_{σ^+}) density at height $U_k \cdot f_{\sigma^+}(Z_k)$. That is, find left and right endpoints of $H(U_k \cdot f_{\sigma^+}(Z_k))$ where

$$H(y) := \{x : f_{\sigma^+}(x) \geq y\}.$$

Let a and b denote the left and right endpoints, respectively.

3. Use V_k to draw a value uniformly from the interval $[a, b]$. Call this value W_k^+ .

Since we have completed one iteration of a slice sampling procedure that was started in stationary mode (started with a value drawn from the $N(0, (\sigma^+)^2/n)$ distribution), the result is a value

$$W_k^+ \sim N(0, (\sigma^+)^2/n).$$

We then let

$$Y_{k+1} = \left(1 - \frac{\phi^+}{n}\right)Y_k + W_k^+.$$

$$\boxed{\mathbf{Y}_{k+1} | \mathbf{Y}_k, \mathbf{Y}_k < \mathbf{0}}$$

Since our ultimate goal is to achieve a coupling of sample paths, in order to generate Y_{k+1} given $Y_k < 0$, we do not simply rehash the steps above in the $Y_k \geq 0$ case. Instead we proceed as follows.

1. Compute $Y^* = \left(1 - \frac{\phi^-}{n}\right)Y_k$.
2. Rescale and shift the value Z_k to give a value

$$Z'_k = Y^* + \frac{\sigma^-}{\sigma^+}Z_k \sim N(Y^*, (\sigma^-)^2/n).$$

3. Using f_{Y^*, σ^-} , to denote the normal density with mean Y^* and variance $(\sigma^-)^2/n$ we now proceed as above, using the same uniform deviates:
 - Use Z'_k as the location of a vertical slice under the normal (f_{Y^*, σ^-}) density. That is, define a vertical slice from the point $(Z'_k, 0)$ to the point $(Z'_k, f_{Y^*, \sigma^-}(Z'_k))$.

- Define a horizontal slice across the normal (f_{Y^*, σ^-}) density at height $U_k \cdot f_{Y^*, \sigma^-}(Z'_k)$. That is, find left and right endpoints of $H(U_k \cdot f_{Y^*, \sigma^-}(Z'_k))$ where

$$H(y) := \{x : f_{Y^*, \sigma^-}(x) \geq y\}.$$

Let c and d denote the left and right endpoints, respectively.

4. To get the final value of $Y_{k+1}|Y_k \sim N\left(1 - \frac{\phi^-}{n} Y_k, \frac{(\sigma^-)^2}{n}\right)$, we consider two cases.

Case One: If the interval $[c, d]$ is completely contained in the interval $[a, b]$, we scale V_k to be uniformly distributed on $[a, b]$ and using the folding coupler as described in Section 3.1.2 to produce a value uniformly distributed on $[c, d]$. At the last step in a stationary slice sampling procedure, this value is normally distributed with mean Y^* and variance $(\sigma^-)^2/n$, and is potentially the same as the result above in the $Y_k > 0$ case. This value is Y_{k+1} .

Case Two: If the interval $[c, d]$ is not completely contained in the interval $[a, b]$, we scale V_k to be uniformly distributed on $[c, d]$. At the last step in a stationary slice sampling procedure, this value is normally distributed with mean Y^* and variance $(\sigma^-)^2/n$. This value is Y_{k+1} .

5 Solving the Stochastic Difference Equation with Perfect Simulation

Now that we know how to move the unbounded process $\{Y_k\}$ forward, we need to be able to move it backwards from $k = 0$ in the same fashion in order to generate an upper process as described in Section 3.2. Since the process $\{Y_k\}$ is unbounded, we now describe how to generate bounding processes $\{Q_k^+\}$, $\{Q_k^-\}$ that will bound the sample paths of $\{Y_k\}$ from above and below, respectively. We require these bounding processes to be reversible and to be such that we are able draw values from their stationary distributions for their starting positions at $k = 0$, namely, Q_0^+ , Q_0^- .

5.1 Choosing Upper and Lower Bounding Processes

If we let $\lambda > \max\{(1 - \frac{\phi^-}{n}), (1 - \frac{\phi^+}{n})\}$ with $0 < \lambda < 1$ (to ensure stationarity of the CTAR(1) process), we can choose our upper and lower bounding processes to be

$$\begin{aligned} Q_k^+ &= \max\{\lambda Q_{k-1}^+ + W_{k-1}^+, 0\} \\ Q_k^- &= \min\{\lambda Q_{k-1}^- + W_{k-1}^-, 0\}. \end{aligned}$$

It is routine to verify that, $Q_k^- \leq Y_k \leq Q_k^+ \Rightarrow Q_{k+1}^- \leq Y_{k+1} \leq Q_{k+1}^+$.

Consider now the process

$$Q_k = \lambda Q_{k-1} + W_{k-1},$$

where W_k is a normal random variable with mean zero and variance σ^2/n . Then, the process $\{Q_k\}$ will be a collection of normal random variables. For it to be stationary, we require that

$$\mathbb{E}[Q_k] = 0 \quad \text{and} \quad \text{Var}[Q_k] = \frac{\sigma^2}{n(1-\lambda^2)}.$$

Therefore, we can compute an upper (lower) bound draw for Q_0^+ (Q_0^-) by taking the absolute value (negative of the absolute value) of a draw from some normal random variable with mean zero and variance $(\sigma^+)^2/[n(1-\lambda^2)]$ (or $(\sigma^-)^2/[n(1-\lambda^2)]$) and adding (subtracting) any positive value we want.

5.2 Moving the Bounding Processes Backwards

Moving the upper bounding process $\{Q_k^+\}$ backwards is quite simple.

Given Q_k^+ , we look at the slice draw W_{k-1}^+ using $U_{k-1}, V_{k-1}, Z_{k-1}$ and let

$$Q_{k-1}^+ = \max \left\{ \frac{Q_k^+ - W_{k-1}^+}{\lambda}, 0 \right\}.$$

Moving the lower bounding process $\{Q_k^-\}$ backwards poses more of a challenge. If we move forward from Q_k^- to Q_{k+1}^- , we wish to reverse the process so that starting with Q_{k+1}^- we may reproduce the value of Q_k^- . The problem is that we don't know whether the forward move would have come from the Case One or Case Two situation described in Section 4.3. So, to reverse this lower bounding process, we consider a "worst case scenario".

The procedure is to look at the interval $[a, b]$ generated when slicing to get W_{k-1}^+ and at the interval $[c, d]$ generated when slicing to get W_{k-1}^- .

If $Q_k^- < a$, no coupling could have occurred, and we let

$$Q_{k-1}^- = \min \left\{ \frac{Q_k^- - W_{k-1}^-}{\lambda}, 0 \right\}.$$

If $Q_k^- \geq a$, coupling could have occurred, and we are forced to assign to Q_{k-1}^- the lowest value possible that it might have achieved. Thus, we let

$$Q_{k-1}^- = \frac{Q_k^- - \frac{d-c}{2}}{\lambda}.$$

5.3 Drawing Values from the Process $\{Y_k\}$

Starting at Q_0^+ and Q_0^- , we move the bounding processes backwards to Q_{-1}^+ , Q_{-1}^- . So, at $k = -1$, all possible values for Y_{-1} lie in between Q_{-1}^+ and Q_{-1}^- . We then generate a forward move for the $\{Y_k\}$ process starting at Q_{-1}^+ and another starting at Q_{-1}^- . Let's call the process originating at the upper bound position $\{Y_k^+\}$, and the one from the lower

bound position $\{Y_k^-\}$. Thus, we start moving the processes forward with $Y_{-1}^+ = Q_{-1}^+$ and $Y_{-1}^- = Q_{-1}^-$. As we move forward, if $Y_0^+ = Y_0^-$, coupling was achieved and $Y_0 = Y_0^+ = Y_0^-$ is a draw from the stationary distribution for $\{Y_k\}$. If $Y_0^+ \neq Y_0^-$, coupling was not achieved, and we must go further back in time.

We then move Q_{-1}^+ back to Q_{-2}^+ and Q_{-1}^- back to Q_{-2}^- . We set $Y_{-2}^+ = Q_{-2}^+$ and $Y_{-2}^- = Q_{-2}^-$ and start moving the processes $\{Y_k^+\}$, $\{Y_k^-\}$ forward from $k = -2$ until $k = 0$. If $Y_{-1}^+ = Y_{-1}^-$, coupling was achieved, and we move both processes to Y_0^+ , Y_0^- . If $Y_{-1}^+ \neq Y_{-1}^-$, we still move the processes forward to $k = 0$, for coupling might occur then. In the case they both end up at $k = 0$ with the same value, such value is a draw from the stationary distribution for $\{Y_k\}$. If this is not the case, we must go farther back in time.

Eventually, we may find ourselves i steps back in time, with Q_{-i}^+ and Q_{-i}^- . We let $Y_{-i}^+ = Q_{-i}^+$, $Y_{-i}^- = Q_{-i}^-$ and start moving the processes $\{Y_k^+\}$, $\{Y_k^-\}$ forward from $k = -i$ until $k = 0$. If, once we get to $k = 0$, $Y_0^+ = Y_0^-$, no matter where the process $\{Y_k\}$ started at time minus infinity, at $k = 0$ it must have the value $Y_0 = Y_0^+ = Y_0^-$, and that is a draw from its stationary distribution. If $Y_0^+ \neq Y_0^-$, we have not achieved a successful coupling, and we must go farther back in time in order to attempt it again.

It might be clear to the reader at this point that if coupling doesn't occur as we move both processes forward, at $k = 0$ their values will not be the same, and we will need to start the process farther back in time. What may not be clear yet is that if coupling occurred as we moved forward, the values of the two process might actually differ at time zero. This is a phenomenon that we call *decoupling* which arises from reversing paths with a worst case scenario, and it contradicts one of the basic tenets of perfect simulation that coupled paths remain forever coupled. In fact, it is possible to give perfect simulation algorithm for the CTAR(1) model where decoupling will not occur, but it is excessively complex and not amenable in practice.

One thing to notice is that we must move the processes $\{Y_k^+\}$, $\{Y_k^-\}$ forward according to the rules explained in Section 4.3, but with one remark. Since we know the upper and lower bound values for the process (given by $\{Q_{-i}^+, \dots, Q_0^+\}$ and $\{Q_{-i}^-, \dots, Q_0^-\}$), none of the following situations can arise:

$$Y_k^+ > Q_k^+, \quad Y_k^- > Q_k^+, \quad Y_k^+ < Q_k^-, \quad Y_k^- < Q_k^-. \quad (7)$$

These situations would never occur if we took into account the values of the upper and lower bound Q -processes in the rules of Section 4.3 to come up with conditional one step transition probabilities for each Y -process move. Such approach would further complicate what already is a somewhat complex algorithm, but without taking this into consideration, the Y -process paths could jump outside of the bounding Q -process paths. To avoid the more complex updates required for Y -process sample paths, in the case that one of the four scenarios in (7) happens, we simply set the value of the violating process at k to be that of its bounding Q -process at k , as we know that if we moved the Y -processes forward correctly, they would have to be sandwiched between the Q -processes. This is precisely where a decoupling may occur in the event that the processes have already coupled. As an example, consider moving from Y_{-7}^+ to Y_{-6}^+ . If we compute the value of Y_{-6}^+ to be greater

than that of Q_{-6}^+ , we must reset the value of Y_{-6}^+ to being equal to that of Q_{-6}^+ , for we don't know where in between Q_{-6}^- and Q_{-6}^+ the value of Y_{-6}^+ should fall, but we know that it must be greater than any value that the process $\{Y_k\}$ could have at $k = -6$. Analogously, if $Y_k^- < Q_k^-$ or $Y_k^- > Q_k^+$, we must set $Y_k^- = Q_k^-$.

Such approach to fix the lack of conditioning when moving forward is what leads to the possibility of decoupling. Suppose our paths had coupled at $k = -9$, and as we moved forward they remained coupled, until we reached $k = -6$ and $Y_{-6}^- = Y_{-6}^+ > Q_{-6}^+$. Since in this scenario both processes violated the boundaries, they both must be reset to the lower and upper bounding processes values, decoupling what originally was two coupled paths. Notice that when implementing this algorithm, once decoupling occurs, there's no need to keep moving forward to $k = 0$, for there's not a chance for coupling the paths again (otherwise we would already have gotten a draw and would not be in this situation). But if only one of the processes must be reset at a particular k , one should keep going forward until $k = 0$, for there's still a chance of coupling.

This section explained exactly how to get perfect draws from the stationary distribution of the process $\{Y_k\}$. Once you have a reasonable amount of draws, they can be used to construct an approximation of the distribution function for the stationary distribution of the process $\{Y_k\}$.

6 Solving a Sample Case

Consider the following CTAR(1) process

$$dX(t) + 1.0X(t)dt = 1.0dB(t), \text{ if } X(t) \geq 0$$

$$dX(t) + 0.5X(t)dt = 0.5dB(t), \text{ if } X(t) < 0$$

It has threshold $\tau = 0$ and it can be discretized into the following stochastic difference equation (with $n = 10$):

$$Y_k = 0.90Y_{k-1} + W_{k-1}^+, \text{ if } Y_{k-1} \geq 0$$

$$Y_k = 0.95Y_{k-1} + W_{k-1}^-, \text{ if } Y_{k-1} < 0$$

where W_k^+ is a normal random variable with mean zero and variance 0.1 and W_k^- is a normal random variable with mean zero and variance 0.025.

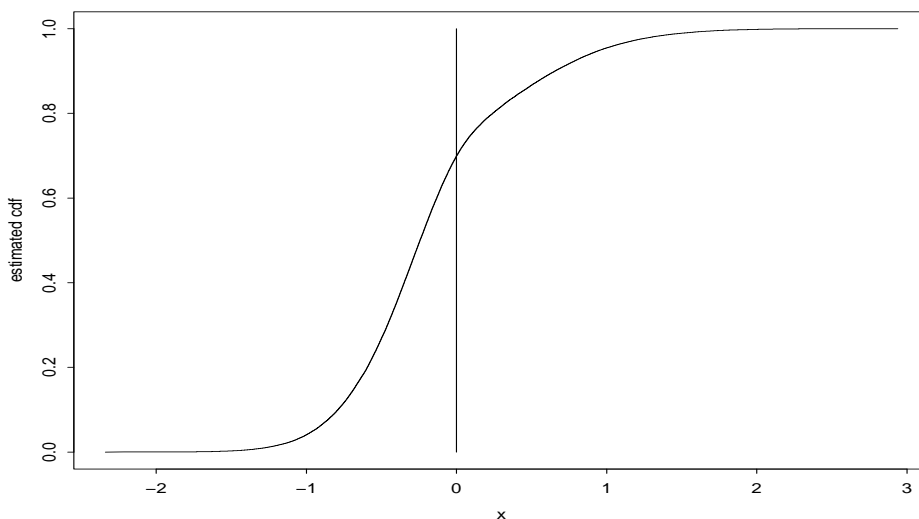
So, in the notation of the previous section, $\tau = 0$, $\sigma^+ = 1.0 > 0.5 = \sigma^-$, so all we have to do now is choose an appropriate value for λ in the upper and lower bounding processes. Since the model requires that $0.95 < \lambda < 1$, we arbitrarily choose $\lambda = 0.99$.

The algorithm described in this paper uses two “worst case scenario” steps. One is involved when we simulate the lower bounding process $\{Q_k^-\}$ and the other is introduced in order to more efficiently run the $\{Y_k\}$ process while keeping it between the bounding Q -processes. This may significantly increase our backward coupling times. While, in theory, we can perfectly execute the algorithm, in this example we have truncated our search for

a backward coupling time. This introduces bias into the algorithm (for example, see [7]), but we minimize this bias by choosing a sufficiently large backward coupling time cutoff limit (determined through simulation) so that approximately 99% of the time we will have a successful coupling in the time allotted. In this example, we choose to search back for 200 time steps.

Then, solving the above using perfect simulation as explained in the previous section to generate 100,000 draws, gives the curve in Figure 2 as the approximate solution for the process cumulative distribution function (cdf) of the stationary distribution. This curve is indistinguishable from stationary distribution cdf obtained by simulating 100,000 draws through a standard forward simulation for 100,000 time steps.

Figure 2: The CDF of the Stationary Distribution of a CTAR(1) Process



6.1 About Coupling, Decoupling and Backwards Coupling Steps

Due to decoupling, out of the 100,000 sample paths, only 98,691 were able to provide us with draws. The 1.3% of the paths that were not used introduced some bias in the final result, but this error is negligible when compared with the error introduced by the initial discretization.

The histogram depicted in Figure 3 shows the distribution of the number of backward steps needed in order to achieve a perfect draw.

In 100,000 trials, we found that the backward coupling time had a median of 65 backward steps, a mean of 69 and standard deviation 20 backward time steps.

Figure 3: The Distribution of Backward Coupling Times

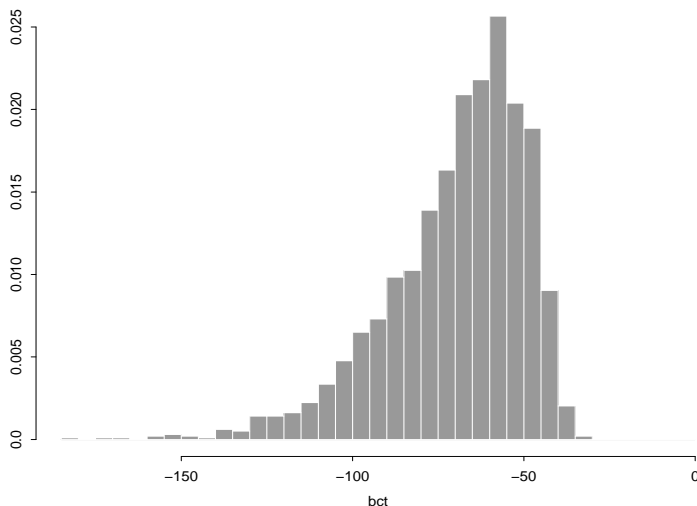


Figure 4 shows (that for one particular draw) the upper and lower bounding processes $\{Q_k^+\}$, $\{Q_k^-\}$ had to go 25 steps back in time in order for the processes $\{Y_k^+\}$, $\{Y_k^-\}$ to couple at $k = -10$ and remain coupled until $k = 0$, generating a draw.

Figure 5 shows an example of decoupling, with the processes $\{Y_k^+\}$, $\{Y_k^-\}$ coupling at $k = -11$ and decoupling at $k = -1$.

6.2 About the Original CTAR(1) Process

The CTAR(1) process considered in this section can be solved analytically, and the stationary density of $\{X(t)\}$ is given by Brockwell [3]:

$$\pi(x) = \begin{cases} ce^{-x^2} & , x \geq 0 \\ 2ce^{-2x^2} & , x < 0 \end{cases} .$$

with $c = (2\sqrt{2} - 2)/\sqrt{\pi}$.

Figure 6 shows the plot of the cdf for the CTAR(1) overlaid with that for the solution of its correspondent discretized version.

Clearly the solution of the discretized model doesn't even come close to the solution of the original CTAR process. But this is not a flaw of the technique developed to use perfect simulation to solve this type of Stochastic Difference Equation. This is simply showing that the original CTAR process is very sensitive to discretization, and thus it is not well represented by its discretized version. Attempts were made with different values of n , all

Figure 4: A Depiction of a Successful Coupling (The thin Lines are Y^+ and Y^- .)

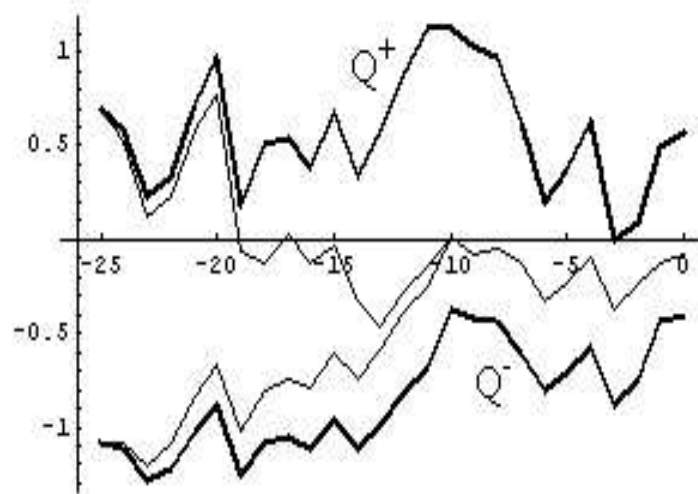


Figure 5: A Depiction of Coupling and then Decoupling (The thin Lines are Y^+ and Y^- .)

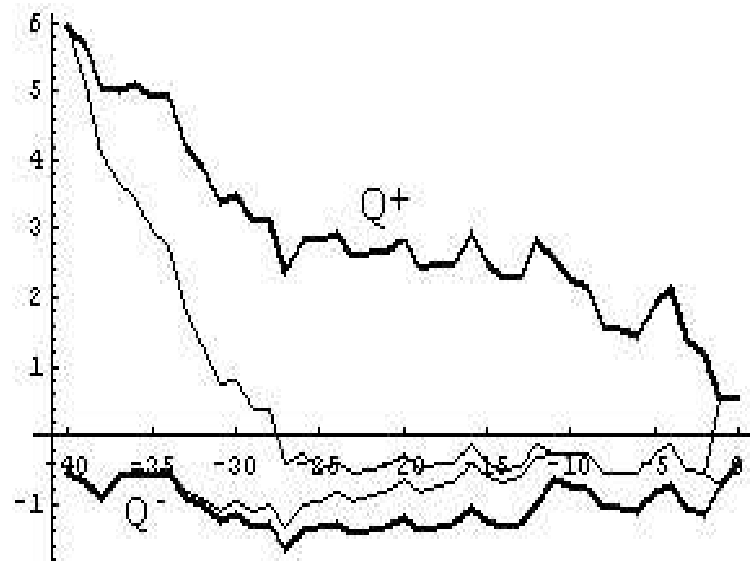
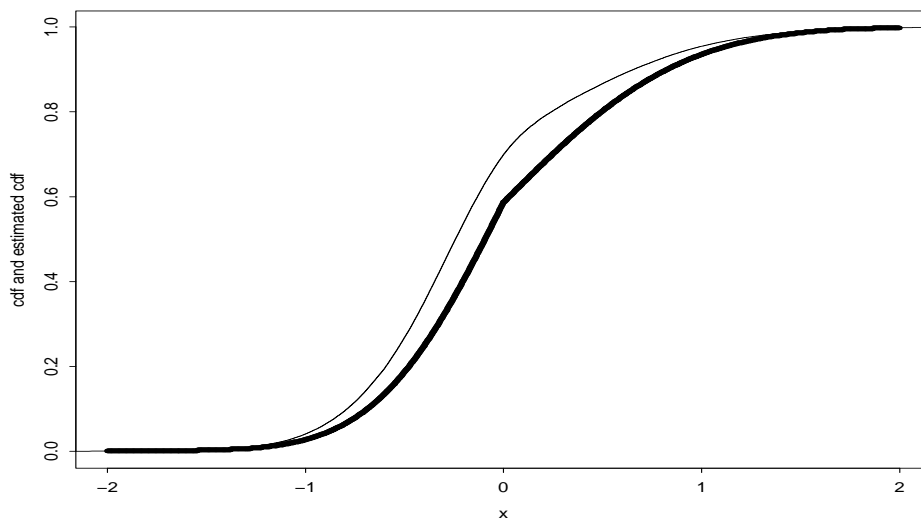


Figure 6: The CDF for a CTAR(1) model (in bold) Overlaid With the CDF for a Discretized CTAR(1)



leading to the same correct solution of the respective difference equation, but not gaining any accuracy on the solution for the original continuous-time process.

References

- [1] P.J. Brockwell. On continuous time threshold arma processes. *J. Statistical Planning and Inference*, 39:291–304, 1994.
- [2] P.J. Brockwell and R.J. Hyndman. On continuous time threshold autoregression. *International J. Forecasting*, 8:157–173, 1992.
- [3] P.J. Brockwell, R.J. Hyndman, and G.K. Grunwald. Continuous time threshold autoregressive models. *Statistica Sinica*, 1:401–410, 1991.
- [4] G. Casella, M. Lavine, and C. Robert. Explaining the perfect sampler. Working Paper 00-16, State University of New York at Stony Brook, Duke University, Durham., 2000.
- [5] J.N. Corcoran and U. Schneider. Shift and scale coupling methods for perfect simulation. *Probability in the Engineering and Informational Sciences*, 17:277–303, 2003.
- [6] J.N. Corcoran and R.L. Tweedie. Perfect sampling from Independent Metropolis-Hastings Chains. *Journal of Statistical Planning and Inference*, 104:297–314, 2002.

- [7] J.A. Fill. An interruptible algorithm for perfect sampling via Markov chains. *ANNAP*, 8:131–162, 1998.
- [8] S.G. Foss and R.L. Tweedie. Perfect simulation and backward coupling. *Stochastic Models*, 14:187–203, 1998.
- [9] S.G. Foss, R.L. Tweedie, and J.N. Corcoran. Simulating the invariant measures of Markov chains using horizontal backward coupling at regeneration times. *Prob. Eng. Inf. Sci.*, 12:303–320, 1998.
- [10] O. Häggström, M.N.M. van Liesholt, and J. Møller. Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5:641–659, 1999.
- [11] W.S. Kendall. Perfect simulation for the area-interaction point process. In L. Accardi and C.C. Heyde, editors, *Probability Towards the Year 2000*, pages 218–234. Springer, New York, 1998.
- [12] J. Møller. Perfect simulation of conditionally specified models. *Journal of the Royal Statistical Society, Ser. B*, 61(1):251–264, 1999.
- [13] D.J. Murdoch and P.J. Green. Exact sampling from a continuous state space. *Scandinavian Journal of Statistics*, 25:483–502, 1998.
- [14] J.G. Propp and D.B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252, 1996.
- [15] D.B. Wilson. Layered multishift coupling for use in perfect sampling algorithms (with a primer on cftp). *Fields Institute Communications*, 26:141–176, 2000.