# Chapter 1
# Acceptance-rejection methods for generating random variates from matrix exponential distributions and rational arrival processes

Gábor Horváth, Miklós Telek

**Abstract** Stochastic models based on matrix exponential structures, like matrix exponential distributions and rational arrival processes, have gained popularity in analytical models recently. However the application of these models in simulation based evaluations is not as widespread yet. One of the possible reasons is the lack of efficient random variates generation methods. In this paper we propose methods for efficient random variates generation for matrix exponential stochastic models based on appropriate representations of the models.

## 1.1 Introduction

Despite of the wide-spread usage of Markovian traffic models, phase-type (PH) distributions [Neuts(1981)] and Markov arrival processes (MAPs) [Latouche and Ramaswami(1999)], in simulations, there are surprisingly few results available on the efficient generation of random variates of these models. Furthermore, there are practically no results available on the efficient generation of random variates of matrix exponential (ME) distributions [Lipsky(1992)] and rational arrival processes (RAPs) [Asmussen and Bladt(1999)] apart from the trivial and computationally heavy method based on the numerical inversion of the cumulative distribution function [Brown et al(1998)Brown, Place, and de Liefvoort]. The aim of this paper is to propose efficient numerical methods for random variate

Gábor Horváth
Department of Telecommunications Technical University of Budapest, H-1521 Budapest, Hungary, e-mail: ghorvath@hit.bme.hu

Miklós Telek
Department of Telecommunications Technical University of Budapest, H-1521 Budapest, Hungary, e-mail: telek@hit.bme.hu

generation based on ME distributions and various versions of RAPs. The few works dealing with efficient generation of PH distributed random variates are based on the stochastic interpretation of PH distributions. These methods simulate the Markov chain which defines the PH distribution until it reaches the absorbing state and generates the required random variates in an efficient way [Neuts and Pagano(1981)]. This procedure of simulating the underlying Markov chain is referred to as *play* method in the sequel. Markovian traffic models are defined by a set of matrices (including vectors as special matrices) referred to as representation. The representation is not unique. Different sets of matrices can represent the same model. More recently, it has been recognized that the computational complexity of the play method depends on the particular representation of the PH distribution [Reinecke et al(2009)Reinecke, Wolter, Bodrog, and Telek, Reinecke et al(2010)Reinecke, Telek, and Wolter].

ME distributions and RAPs do not have a straightforward stochastic interpretation. Consequently the methods available for generating random variates of Markovian traffic models cannot be used for their simulation. To overcome this difficulty we propose a version of the acceptance-rejection method. The acceptance-rejection method is a widely used method in simulation [Robert and Casella(2004)]. It consists of two main steps, drawing random samples from an easy to compute distribution, and accept the sample with a sample dependent probability such that the overall probability density of the accepted samples is identical with the required one. The computational complexity of this method depends on the sample efficiency, which is the ratio of the number of accepted and the number of generated samples. Using a general distribution (e.g. exponential) whose shape is different from the required one results in a low sample efficiency. We propose specific methods with higher sample efficiency.

It turns out that, similar to the case of Markovian traffic models, the representation of ME distributions and RAPs affects the sample efficiency and the computational complexity of generating random variates of these models. We evaluate the behaviour of two particular representations with nice structural properties.

As it is demonstrated among the numerical experiments, there are cases when the proposed method which is developed for simulating ME distributions and RAPs is more efficient for the simulation of Markovian models (PH distributions and MAPs) than the existing methods based on their stochastic interpretations.

A procedure to generate pseudo random numbers uniformly distributed on $(0, 1)$ is part of all common programming languages and simulation packages. In this work we investigate the computational effort to generate random variates of ME distribution and RAP using these uniformly distributed pseudo random numbers. The complexity of various computational steps might differ in various programming environments. We define the computational complex-

ity of the proposed methods as a function of the more complex computational steps (number of pseudo random samples, log operations, exp operations).

The main part of the paper is devoted to ME distributed random variate generation because it is a main building block of RAP simulation. Section 1.2 introduces ME distributions and RAPs and Section 1.3 summarizes the steps and the complexity of generating random variates of Markovian traffic models. Having these preliminaries Section 1.4 introduces the proposed acceptance-rejection method. Section 1.5 specializes the acceptance-rejection method to particular representations which are efficient for random variate generation. The use of ME distributed random number generation for simulating various RAPs is explained in Section 1.6. To demonstrate the efficiency of the proposed methods examples and related numerical experiments are presented in Section 1.7.

## 1.2 Matrix exponential distributions and rational arrival processes

We start the summary of the preliminaries with the definition of ME and PH distributions and later we introduce RAPs and MAPs and their variants.

**Definition 1.** The real valued row vector square matrix pair of size $N$, $(\tau, \mathbf{T})$, defines a matrix exponential distribution iff

$$F(x) = Pr(X < x) = 1 - \tau e^{\mathbf{T}x}\mathbb{1}, \qquad x \geq 0 \tag{1.1}$$

is a valid cumulative distribution function (cdf), i.e., $F(0) \geq 0$, $\lim_{x\to\infty} F(x) = 1$ and $F(x)$ is monotone increasing.

In (1.1), the row vector, $\tau$, is referred to as the initial vector, the square matrix, $\mathbf{T}$, as the generator and $\mathbb{1}$ as the closing vector. Without loss of generality [Lipsky(1992)], throughout this paper we assume that the closing vector is a column vector of ones, i.e., $\mathbb{1} = [1, 1, \ldots, 1]^T$. Further more we restrict our attention to the case when there is no probability mass at 0, i.e., $F(0) = 0$, or equivalently $\tau\mathbb{1} = 1$.

The probability density function (pdf) of the matrix exponential distribution defined by $(\tau, \mathbf{T})$ is

$$f(x) = \tau e^{\mathbf{T}x}(-\mathbf{T})\mathbb{1}. \tag{1.2}$$

To ensure that $\lim_{x\to\infty} F(x) = 1$, $\mathbf{T}$ has to fulfill the necessary condition that the real parts of its eigenvalues are negative (consequently $\mathbf{T}$ is non-singular).

The remaining constraint is the monotonicity of $F(x)$, or, equivalently, the non-negativity of $f(x)$. This constraint is the most difficult to check. The

simulation methods proposed below implement control checks to indicate if this condition is violated during the simulation run.

**Definition 2.** If $\tau$ is non-negative and $\mathbf{T}$ has negative diagonal and non-negative off diagonal elements then $(\tau, \mathbf{T})$ is said to be Markovian and defines a PH distribution.

PH distributions can be interpreted as a time duration in which a Markov chain having $N$ transient and an absorbing state arrives to the absorbing state. In case of a non-Markovian representation, however, there is no such simple stochastic interpretation available.

In case of $N = 2$ the class of ME distributions is identical with the class of PH distributions, but if $N > 2$ the class of PH distributions is a proper subset of the class of ME distributions [van de Liefvoort(1990)].

A rational arrival process (RAP) is a point process in which the inter-arrival times are ME distributed [Asmussen and Bladt(1999), Mitchell(2001)].

**Definition 3.** The square matrix pair of size $N$, $(\mathbf{H_0}, \mathbf{H_1})$, satisfying $(\mathbf{H_0} + \mathbf{H_1})\ \mathbb{1} = \mathbf{0}$ defines a stationary RAP iff the joint density function of the interarrival times

$$f(x_1, \ldots, x_k) = \tau e^{\mathbf{H_0} x_1} \mathbf{H_1} e^{\mathbf{H_0} x_2} \mathbf{H_1} \ldots e^{\mathbf{H_0} x_k} \mathbf{H_1} \mathbb{1} \tag{1.3}$$

is non-negative for all $k \geq 1$ and $x_1, x_2, \ldots, x_k \geq 0$ and $\tau$ is the unique solution of $\tau(-\mathbf{H_0})^{-1}\mathbf{H_1} = \tau$, $\tau\mathbb{1} = 1$.

If the solution $\tau(-\mathbf{H_0})^{-1}\mathbf{H_1} = \tau$, $\tau\mathbb{1} = 1$ is not unique then $(\mathbf{H_0}, \mathbf{H_1})$ does not define the stationary behaviour of the process.

RAPs inherit several properties from ME distributions. The real parts of the eigenvalues of matrix $\mathbf{H_0}$ are negative; consequently the matrix is non-singular. There is a real eigenvalue with maximal real part. Similar to the case of ME distributions the non-negativity of the joint density function is hard to check and the proposed simulation methods contain run time checks to indicate if the non-negativity of the joint density is violated. The first interarrival time of the RAP is ME distributed with initial vector $\tau$ and square matrix $\mathbf{H_0}$. Vector $\tau$ and the off-diagonal blocks of matrix $\mathbf{H_0}$ may contain negative elements. If $\mathbf{H_1} = -\mathbf{H_0}\mathbb{1}\tau$ then the consecutive interarrivals are independent and identically distributed, that is the RAP is a renewal process with ME distributed interarrivals.

**Definition 4.** If $\mathbf{H_1} \geq 0$ and all non-diagonal elements of $\mathbf{H_0}$ are non-negative, then the matrix pair $(\mathbf{H_0}, \mathbf{H_1})$ is said to be Markovian and define a Markov Arrival Process (MAP).

The joint density function (1.3) of a MAP is always positive and $\tau \geq 0$. In case of MAPs one can interpret the non-diagonal elements of matrix $\mathbf{H_0}$ and the elements of $\mathbf{H_1}$ as transition rates corresponding to hidden and visible

events, respectively. Vector $\tau$ can be interpreted as the state of the MAP at time zero.

The extension of plain (single arrival, single event type) MAPs to MAPs with batch arrivals (BMAPs) [Latouche and Ramaswami(1999)] and with different types of arrivals (MMAPs) [He and Neuts(1998)] can be applied to RAPs as well. This extension results in batch rational arrival process (BRAP) and marked rational arrival process (MRAP) [Bean and Nielsen(2010)], respectively. The stochastic behaviour of MRAPs and BRAPs is practically the same. Below, we discuss MRAPs only.

**Definition 5.** A set of square matrices of size $N$, $(\mathbf{H_0}, \mathbf{H_1}, \ldots, \mathbf{H_K})$, satisfying $\sum_{k=0}^{K} \mathbf{H_k}\ \mathbb{1} = \mathbf{0}$, defines a stationary MRAP with $K$ event types iff the joint density function of the arrival sequence (consecutive interarrival times and event types)

$$f(x_1, k_1, \ldots, x_j, k_j) = \tau e^{\mathbf{H_0} x_1} \mathbf{H_{k_1}} e^{\mathbf{H_0} x_2} \mathbf{H_{k_2}} \ldots e^{\mathbf{H_0} x_j} \mathbf{H_{k_j}} \mathbb{1} \tag{1.4}$$

is non-negative for all $j \geq 1$ and $x_1, x_2, \ldots, x_j \geq 0$, $1 \leq k_1, k_2, \ldots, k_j \leq K$ and $\tau$ is the unique solution of $\tau(-\mathbf{H_0})^{-1} \sum_{k=1}^{K} \mathbf{H_k} = \tau$, $\tau \mathbb{1} = 1$.

If the solution $\tau(-\mathbf{H_0})^{-1} \sum_{k=1}^{K} \mathbf{H_k} = \tau$, $\tau \mathbb{1} = 1$ is not unique then $(\mathbf{H_0}, \mathbf{H_1}, \ldots, \mathbf{H_K})$ does not define the stationary behaviour of the process.

The class of MRAPs contains MMAPs since an MRAP is an MMAP if $\tau \geq 0$, $\mathbf{H_k} \geq 0$ for $k = 1, \ldots, K$ and all non-diagonal elements of $\mathbf{H_0}$ are non-negative.

For later use we also define the initial vector after the first event. If a RAP with representation $(\mathbf{H_0}, \mathbf{H_1})$ starts with initial vector $\alpha$ and the first arrival happens at time $x$ then the initial vector characterizing the second arrival is $\alpha e^{\mathbf{H_0} x} \mathbf{H_1}$ / $\alpha e^{\mathbf{H_0} x} \mathbf{H_1} \mathbb{1}$. If an MRAP with representation $(\mathbf{H_0}, \mathbf{H_1}, \ldots, \mathbf{H_K})$ starts with initial vector $\alpha$, and the first event happens at time $x$ then the probability that the event is of type $k$ is $\alpha e^{\mathbf{H_0} x} \mathbf{H_k} \mathbb{1}$ / $\sum_{j=1}^{K} \alpha e^{\mathbf{H_0} x} \mathbf{H_j} \mathbb{1}$. Further more, if an MRAP with representation $(\mathbf{H_0}, \mathbf{H_1}, \ldots, \mathbf{H_K})$ starts with initial vector $\alpha$, the first arrival happens at time $x$ and it is of type $k$ then the initial vector characterizing the second arrival is $\alpha e^{\mathbf{H_0} x} \mathbf{H_k}$ / $\alpha e^{\mathbf{H_0} x} \mathbf{H_k} \mathbb{1}$.

The above matrix representations of the introduced processes are not unique. Various similarity transformations allow generating different matrix representation of a given process. Similarity transformations exists for matrix representations of identical size [Buchholz and Telek(2011)] and different sizes [Telek and Horváth(2007)]. We recall one of the possible similarity transformations for MRAPs from [Telek and Horváth(2007)] without proof. Similar transformations for RAPs and ME distributions can be obtained as a special case [Buchholz and Telek(2010)].

**Theorem 1.** *If there is a matrix* $\mathbf{W} \in \mathbb{R}^{n,m}$, $m \geq n$ *such that* $\mathbb{1}_n = \mathbf{W} \mathbb{1}_m$ *(where* $\mathbb{1}_n$ *is the column vector of size* $n$*),* $\mathbf{W H_k} = \mathbf{G_k W}$ *for* $k = 0, \ldots, K$ *then* $(\mathbf{H_0}, \ldots, \mathbf{H_K})$ *and* $(\mathbf{G_0}, \ldots, \mathbf{G_K})$ *define the same MRAP.*

## 1.3 Generating random variates of Markovian traffic models

A trivial way to generate PH and ME distributed random numbers is based on the numerical inversion of the cdf. This computationally heavy method can be replaced by more efficient ones if the distribution allows a simple stochastic interpretation, e.g., in case of PH distributions. Due to the simple stochastic interpretation of PH distributions through Markov chains the generation of PH distributed random variate can be made without the inversion of the numerical matrix exponential function in (1.1). Simulation approaches based on the underlying Markov-chain interpretation are presented in [Neuts and Pagano(1981), Reinecke et al(2009)Reinecke, Wolter, Bodrog, and Telek,
Reinecke et al(2010)Reinecke, Telek, and Wolter]. Below we list some of the related results of these papers and introduce some concepts which are used also in the current work for efficient random number generation.

- General PH distributions: General PH distributions can be interpreted as time to absorption of a Markov chain with $N$ transient states and an absorbing state. The behavior of the Markov chain can be simulated by drawing random samples for the initial state, and by drawing random samples for the state sojourn times and successor states, repeatedly, until the absorbing state is reached. This method is referred to as play method. Drawing samples of the state sojourn times requires drawing exponentially distributed random numbers ($R_{Exp(\lambda)}$) that are generated by transforming a random number $U$ uniformly distributed on $(0, 1)$ as

$$R_{Exp(\lambda)} = -\frac{\log U}{\lambda} \tag{1.5}$$

  Choosing the initial or a successor state requires drawing an additional random number $U$ uniformly distributed on $(0, 1)$ and comparing with the partial sums of elements of the probability vector. The play method is efficient if the mean number of state transitions before absorption is low. More efficient ways of generating random samples form PH distributions are proposed and analyzed in [Neuts and Pagano(1981), Reinecke et al(2009)Reinecke, Wolter, Bodrog, and Telek].
  [Neuts and Pagano(1981)] recommends to sample the behavior of the discrete time Markov chain embedded at state transitions instances, count the number of visits to each states (each sets of states with identical rate parameters) till absorption and compute the PH distributed random sample as the sum of Erlang distributed random variables according to the number of visits and the rate of the associated state (set of states). Drawing Erlang distributed random variates require only a single evaluation of the logarithm function that is a considerable advantage:

$$R_{Erl(\lambda,n)} = \sum_{i=1}^{n} -\frac{\log U_i}{\lambda} = -\frac{1}{\lambda} \log \prod_{i=1}^{n} U_i. \tag{1.6}$$

[Reinecke et al(2009)Reinecke, Wolter, Bodrog, and Telek] recommends to apply a similarity transformation of the original PH representation such that the transformed representation is cheaper to simulate. The complexity of these methods can further be improved by efficient discrete random variable sampling using the alias method [Kronmal and Peterson(1979)].

- APH distributions: if the PH distribution has an acyclic representation, even more efficient algorithms exist to generate random variates. Each APH can be transformed to one of the three canonical forms [Cumani(1982)]. Assuming that an APH distribution is given in CF-1 form a random variate is generated in two steps: first the initial state is drawn, then the time until absorption is sampled as the sum of exponentially distributed sojourn times of states between the initial and the absorbing state. Due to the structure of the CF-1 form, there is always exactly one successor state so there is no need to draw sample for choosing next states. Another important feature of the CF-1 is the lack of cycles; thus the procedure terminates in at most as many steps as the phases of the APH.
- Hyper-Erlang (HEr) distribution: HEr distribution is a convex combination of Erlang distributions. In case of a HEr representation first the Erlang branch has to be chosen and than the Erlang distributed random number has to be drawn.
- Hyper-Exponential (HE) distribution: HE distribution is a convex combination of exponential distributions. HE distribution is the most efficient representation of PH distributions with respect to random number generation. Only two operations are required: to choose the branch and to draw sample for the selected exponential distribution.
- Feedback-Erlang block (FEB): A Feedback-Erlang block is a series of independent, identical exponentially distributed phases with a single feedback from the last phase to the first one, as it is depicted in Figure 1.1. It is the main building block of the monocyclic representation introduced in [Mocanu and Commault(1999)]. FEB has 3 parameters the number of states $n$, the parameter of the exponential distribution $\sigma$ and the feedback probability $z$.

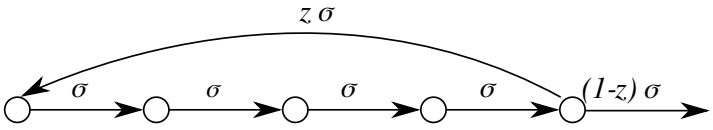


**Fig. 1.1** A single Feedback-Erlang block

FEB has the following advantages:

– it can represent complex eigenvalues in a Markovian way,
– it represents a real eigenvalue as a single exponential phase ($n = 1, z = 0$),
– its eigenvalues are easy to obtain which makes the construction of FEBs easy,
– it is efficient to draw random numbers from FEB.

The generation of a sample from a FEB is similarly efficient to the generation of an Erlang distributed sample. First a geometrically distributed discrete random variate is sampled with parameter $z$, $\Delta$, and after that

$$R_{FEB(\sigma,n,z)} = -\frac{1}{\sigma}\log\prod_{i=1}^{n\Delta} U_i. \tag{1.7}$$

## 1.4 Generating random variates from matrix-exponential distributions having a Markovian generator

In this section we present the main concept of the proposed acceptance-rejection method to generate random variates from a ME distribution. To apply this method we assume that the representation of the ME distribution has a Markovian generator and a general initial vector (which might contain negative elements). The next section proposes such representations with Markovian generators. This section focuses only on the main idea of the proposed method. This acceptance-rejection approach is the basis of the later introduced simulation of ME distributions and various RAPs.

Let $(\alpha, \mathbf{A})$ of size $N$ be the representation of the ME distribution such that $\mathbf{A}$ is a Markovian generator matrix (non-diagonal elements are non-negative, and the row sums are non-positive). The probability density function can be expressed as a non-convex combination of PH distributions as follows:

$$f(x) = \alpha e^{\mathbf{A}x}(-\mathbf{A})\mathbb{1} = \sum_{i=1}^{N} \alpha_i \cdot \underbrace{\mathbb{e}_i\, e^{\mathbf{A}x}(-\mathbf{A})\mathbb{1}}_{g_i(x)}, \tag{1.8}$$

with $\mathbb{e}_i$ denoting a row vector of size $N$ whose $i$th element is one and all other elements are zeros. Observe that $(\mathbb{e}_i, \mathbf{A})$ is a Markovian representation of the PH distribution with pdf $g_i(x)$; consequently $\int_0^\infty g_i(x)dx = 1$.

To cope with the negative coefficients, we apply an acceptance-rejection method to generate a random variate as follows. The set of coefficients of the density function is divided into $\mathcal{A}_+$ and $\mathcal{A}_-$, such that $i \in \mathcal{A}_+$ if $\alpha_i \geq 0$ and $i \in \mathcal{A}_-$ otherwise. In this way $f(x)$ is separated into a positive part $(f_+(x))$ and a negative part$(f_-(x))$

$$f(x) = \underbrace{\sum_{i\in\mathcal{A}_+} \alpha_i \cdot g_i(x)}_{f_+(x)} + \underbrace{\sum_{i\in\mathcal{A}_-} \alpha_i \cdot g_i(x)}_{f_-(x)}. \tag{1.9}$$

Note that $f_+(x) \geq 0, \quad \forall x \geq 0$ and $f_-(x) \leq 0, \quad \forall x \geq 0$ holds.

Multiplying with $p^* = 1/\sum_{j\in\mathcal{A}_+} \alpha_j$, the positive part gets normalized and we get

$$\hat{f}_+(x) = \sum_{i\in\mathcal{A}_+} \alpha_i\, p^* \cdot g_i(x), \tag{1.10}$$

that is a valid phase-type distribution with Markovian representation $(p^* \sum_{i\in\mathcal{A}_+} \alpha_i\, \mathbb{e}_i, \mathbf{A})$, where the initial vector is non-negative and normalized. With these notations and definitions, the acceptance-rejection based method to generate random numbers from $(\alpha, \mathbf{A})$ is formalized by Algorithm 1.

**Algorithm 1** Algorithm for generating ME distributed random variates having a Markovian generator

```
Start: Draw a f̂+(x) distributed random sample:
        I =  discrete random sample with distribution
             p* Σ_{i∈A+} α_i e_i
        R =  random sample with pdf g_I(x)
              by any PH sampling method
if A− = ∅ then
   return R
else
   Calculate acceptance probability:
   p_accept(R) = (f+(R)+f−(R)) / f+(R)
   if p_accept(R) < 0 then
      error "INVALID DENSITY !!!"
   end if
   Draw a uniform sample U
   if U < p_accept(R) then
      return R
   else
      goto Start
   end if
end if
```

**Theorem 2.** *Algorithm 1 provides an $f(x)$ distributed random number and the mean number of required samples is geometrically distributed with parameter $p^*$, i.e., the probability that $n$ samples are required is $(1-p^*)^{n-1}p^*$.*

*Proof.* Let $f^*(x)$ be the probability density of the sample generated by Algorithm 1. In accordance with the standard proof of the acceptance rejection method we are going to show that $f^*(x) = f(x)$. The probability density that

the first step of the algorithm results in sample $R$ is $\hat{f}_+(R)$. The probability density that sample $R$ is the accepted can be computed as

$$f^*(R) = \frac{\hat{f}_+(R)p_{accept}(R)}{\int_x \hat{f}_+(x)p_{accept}(x)dx} = \\ \frac{p^*\, f_+(R)\, \dfrac{f_+(R)+f_-(R)}{f_+(R)}}{\displaystyle\int_x p^*\, f_+(x)\, \frac{f_+(x)+f_-(x)}{f_+(x)}dx} = \frac{p^*\, f(R)}{p^* \int_x f(x)dx} = f(R) \tag{1.11}$$

The steps of the iterative procedure are independent. The probability of accepting a sample is $\int_x \hat{f}_+(x)p_{accept}(x)dx = p^*$.

## 1.5 Generating matrix-exponentially distributed random variates using Feedback-Erlang blocks

As it is shown in Section 1.4, there are several representations from which it is very efficient to draw random numbers. In this section we present two general representations with special structures which are composed by Feedback-Erlang blocks.

### *1.5.1 Hyper-Feedback-Erlang Representation*

**Definition 6.** A Hyper-Feedback-Erlang (Hyper-FE) distribution is defined by an initial probability vector $\alpha$ and a transient generator having the following special structure (see Figure 1.2):

$$\mathbf{A} = \begin{bmatrix} \mathbf{M_1} & & & \\ & \mathbf{M_2} & & \\ & & \ddots & \\ & & & \mathbf{M_J} \end{bmatrix}, \tag{1.12}$$

where matrices $\mathbf{M_j}$ of size $n_j m_j \times n_j m_j$ are the sub-generators of several concatenated feedback Erlang blocks:

$$\mathbf{M_j}(\sigma_j, n_j, z_j, m_j) = \left[\begin{array}{c|c|c} \begin{matrix} -\sigma_j & \sigma_j & & \\ & \ddots & \ddots & \\ & & -\sigma_j & \sigma_j \\ z_j\sigma_j & & & -\sigma_j \end{matrix} & & \\ \hline & \begin{matrix} & \\ \ddots & \\ & \end{matrix} & \begin{matrix} & \\ \ddots & \\ & \end{matrix} \\ \hline & & \begin{matrix} -\sigma_j & \sigma_j & & \\ & \ddots & \ddots & \\ & & -\sigma_j & \sigma_j \\ z_j\sigma_j & & & -\sigma_j \end{matrix} \end{array}\right] \begin{matrix} \\ \\ (1-z_j)\sigma_j \\ \\ \\ \\ \\ \\ \\ \end{matrix} . \tag{1.13}$$

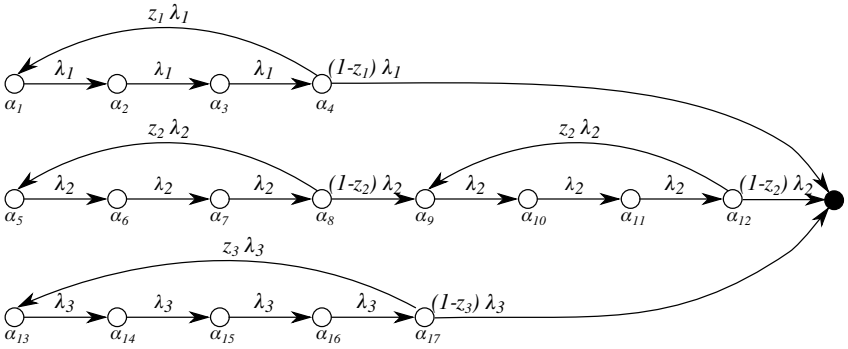


**Fig. 1.2** Structure of the Hyper-Feedback-Erlang Distribution

Having a general non-Markovian representation of a ME distribution, $(\tau, \mathbf{T})$, we look for an equivalent representation $(\alpha, \mathbf{A})$ where $\mathbf{A}$ has Hyper-FE structure. We denote the $j$th eigenvalue of $\mathbf{T}$ by $\lambda_j$ (or, if it is a complex eigenvalue, the complex conjugate eigenvalue pair by $\lambda_j = a_j + b_j\mathbb{i}$ and $\overline{\lambda_j} = a_j - b_j\mathbb{i}$) and its multiplicity by $\rho_j$. The number of distinct real eigenvalues and complex conjugate eigenvalue pairs is $J$.

In the generator of the resulting Hyper-FE representation each matrix $\mathbf{M_j}$ in the block diagonal of $\mathbf{A}$ implements one real eigenvalue or a conjugate complex eigenvalue pair of $\mathbf{T}$. The construction of matrix $\mathbf{M_j}$ is performed as follows [Mocanu and Commault(1999)]:

- if $\lambda_j$ is real, the corresponding matrix degrades to an Erlang block; thus the parameters of $\mathbf{M_j}$ are:

$$\sigma_j = \lambda_j, \quad n_j = 1, \quad z_j = 0, \quad m_j = \rho_j, \tag{1.14}$$

- if $\lambda_j$ is complex, the parameters of $\mathbf{M_j}$ are:

$$n_j = \text{the smallest integer for which } a_j/b_j > \tan(\pi/n_j), \tag{1.15}$$

$$\sigma_j = \frac{1}{2}\left(2a_j - b_j \tan\frac{\pi}{n_j} + b_j \cot\frac{\pi}{n_j}\right), \tag{1.16}$$

$$z_j = \left(1 - \left(a_j - b_j \tan\frac{\pi}{n_j}\right)/\sigma_j\right), \tag{1.17}$$

$$m_j = \rho_j. \tag{1.18}$$

This construction ensures that $\mathbf{A}$ is a valid Markovian transient generator that has all the eigenvalues of $\mathbf{T}$ with the proper multiplicities. However, the FEBs, implementing the complex eigenvalues, introduce "extra" eigenvalues as well, but they do not cause problems because the initial vector $\alpha$ is set such that the "extra" eigenvalues have zero coefficients.

Initial vector $\alpha$ is obtained as follows [Buchholz and Telek(2011)]. Let $n$ and $m$ ($n \leq m$) be the size of $\mathbf{T}$ and $\mathbf{A}$ respectively. Compute matrix $\mathbf{W}$ of size $n \times m$ as the unique solution of

$$\mathbf{TW} = \mathbf{WA}, \qquad \mathbf{W}\mathbb{1} = \mathbb{1}, \tag{1.19}$$

and based on $\mathbf{W}$ the initial vector is

$$\alpha = \tau \cdot \mathbf{W}. \tag{1.20}$$

Vector $\alpha$ is decomposed into sub-vectors according to the block structure of $\mathbf{A}$ and the vector element associated with state $i$ of block $j$ is denoted by $\alpha_{j,i}$. Similar to (1.8), the probability density function can be then expressed as:

$$f(x) = \alpha e^{\mathbf{A}x}(-\mathbf{A})\mathbb{1} = \sum_{j=1}^{J}\sum_{i=1}^{n_j m_j} \alpha_{j,i} \cdot \underbrace{\mathbb{e}_i\, e^{\mathbf{M_j}x}(-\mathbf{M_j})\mathbb{1}}_{g_{j,i}(x)}, \tag{1.21}$$

Observe that $(\mathbb{e}_i, \mathbf{M_j})$ is a Markovian representation for $g_{k,i}(x)$, from which it is very efficient to draw random numbers since it is composed by FEBs.

The method to obtain a random variate with density $g_{k,i}(x)$ denoted by $R_{g_{k,i}}$ is the following:

$$L_{j,i} = n_j\, m_j - i + 1 + \sum_{\ell=\lceil i/n_j \rceil}^{m_j} n_j \cdot \left\lfloor \frac{\log U_\ell}{\log z_j} \right\rfloor, \tag{1.22}$$

$$R_{g_{j,i}} = -\frac{1}{\sigma_j} \log \prod_{\ell=1}^{L_{j,i}} U_\ell.$$

In this expression, $L_{j,i}$ corresponds to the number of steps (exponential distributions) taken before absorption. The first term, $n_j\, m_j - i + 1$ is the number of steps that is taken without feedback, while the sum represents the steps due

to feedbacks: $\lfloor \log U_\ell / \log z_j \rfloor$ is the geometrically distributed random variate for the number of feedback loops and $n_j$ is the number of extra steps for a feedback loop.

In the case when $\alpha_{j,i} \geq 0, \forall i,j$, generating a random variate from $f(x)$ is simple: draw a discrete random sample with distribution $\alpha$ for the starting point of the Hyper-FE structure, and draw a $g_{j,i}(x)$ distributed random number according to (1.22).

However, in case the initial vector has negative elements, we apply the acceptance-rejection method to generate a random variate as described in Section 1.3. Utilizing the efficient Hyper-FE structure of $\mathbf{A}$ the random variate in the third line of Algorithm 1 is generated efficiently.

In each iteration of the algorithm before accepting a sample there is exactly one logarithm function computed to obtain a sample from an Erlang distribution of order $L_{j,i}$ and $(m_j - \lceil i/n_j \rceil + 1)$ logarithm functions are computed to draw the number of times a feedback loop is traversed in the FEBs. Note that it is not necessary to evaluate $\log z_j$ every time, since it can be pre-calculated before starting the algorithm. The total number of logarithms evaluated is

$$\#ilog = \sum_{j=1}^{J} \sum_{i=1}^{n_j m_j} \alpha_{j,i} \cdot (2 + m_j - \lceil i/n_j \rceil). \tag{1.23}$$

For the average number of uniformly distributed random samples required in one iteration before accepting the sample we get

$$\#iuni = \sum_{j=1}^{J} \sum_{i=1}^{n_j m_j} \alpha_{j,i} \cdot \left[ \underbrace{\left(1 + m_j - \lceil i/n_j \rceil\right)}_{\text{to evaluate } L_{j,i}} + \underbrace{\left(n_j\, m_j - i + 1 + (1 + m_j - \lceil i/n_j \rceil) n_j/(1 - z_j)\right)}_{E(L_{j,i}) \text{ uniforms required by } R_{g_{j,i}}} \right] \tag{1.24}$$

Taking into consideration that the mean number of rejections until a sample is accepted is $p^*$, we have the following mean total number of basic operations

$$\#log = \frac{\#ilog}{p^*}, \quad \#uni = \frac{\#iuni}{p^*}. \tag{1.25}$$

### *1.5.2 Hypo-Feedback-Erlang Representation*

**Definition 7.** A Hypo-Feedback-Erlang (Hypo-FE) distribution is defined by an initial probability vector $\alpha$ and a transient generator having the following special structure (see Figure 1.3):

$$\mathbf{A} = \begin{bmatrix} \mathbf{M_1} & \mathbf{M'_1} & & \\ & \mathbf{M_2} & \mathbf{M'_2} & \\ & & \ddots & \\ & & & \mathbf{M_J} \end{bmatrix}, \tag{1.26}$$

where matrices $\mathbf{M_j}$ are defined in (1.13) and

$$\mathbf{M'_j} = (-\mathbf{M_j})\mathbb{1} \cdot \mathbb{e}_1 \tag{1.27}$$

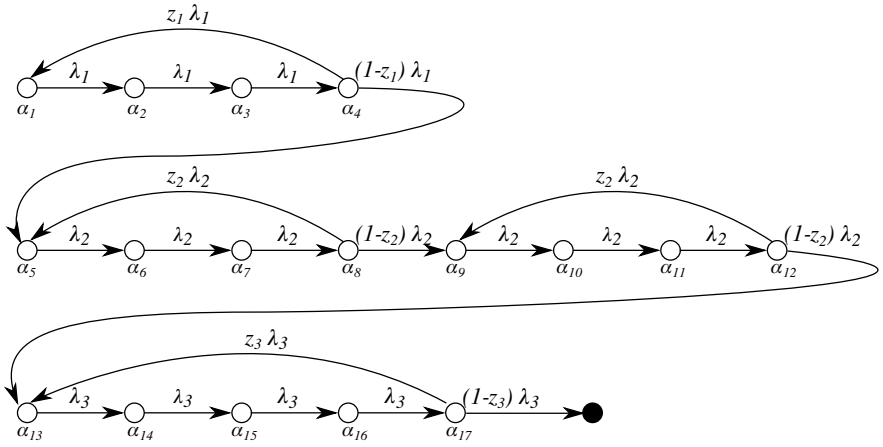


**Fig. 1.3** Structure of the Hypo-Feedback-Erlang Distribution

Matrices $\mathbf{M_j}$ are constructed the same way as in Section 1.5.1, and the initial vector is obtained by the same procedure.

Similar to the Hyper-FE structure, from the Hypo-FE structure it is also very efficient to draw random numbers.

$$L_{j,i} = n_j\, m_j - i + 1 + \sum_{\ell=\lceil i/n_j \rceil}^{m_j} n_j \cdot \left\lfloor \frac{\log U_\ell}{\log z_j} \right\rfloor, \tag{1.28}$$

$$R_{g_{j,i}} = -\frac{1}{\sigma_j} \log \prod_{\ell=1}^{L_{j,i}} U_\ell + \sum_{r=j+1}^{J} (-1) \frac{1}{\sigma_r} \log \prod_{\ell=1}^{L_{r,1}} U_\ell. \tag{1.29}$$

The only difference compared to Hyper-FE structure is that after traversing the initially selected block ($j$), all consecutive blocks are traversed until the absorption.

The cost of generating a random sample from the Hypo-FE structure is calculated similar to the Hyper-FE case. The final expressions including the cost of sample rejections are

$$\#log = \frac{1}{p^*} \sum_{j=1}^{J} \sum_{i=1}^{n_j m_j} \alpha_{j,i} \cdot \left[ (2 + m_j - \lceil i/n_j \rceil) + \sum_{r=j+1}^{J} (1 + m_r) \right] \tag{1.30}$$

$$\begin{aligned} \#uni = \frac{1}{p^*} \sum_{j=1}^{J} \sum_{i=1}^{n_j m_j} \alpha_{j,i} \cdot \Bigg[ & \underbrace{(1 + m_j - \lceil i/n_j \rceil)}_{\text{to evaluate } L_{j,i}} \\ & + \underbrace{\left( n_j\, m_j - i + 1 + (1 + m_j - \lceil i/n_j \rceil) n_j/(1 - z_j) \right)}_{E(L_{j,i}) \text{ uniforms required by first term of } R_{g_{j,i}}} \\ & + \underbrace{\sum_{r=j+1}^{J} \left( m_r + n_r\, m_r + n_r\, m_r/(1 - z_r) \right)}_{\text{uniforms required by the sum in } R_{g_{j,i}}} \Bigg] \end{aligned} \tag{1.31}$$

It might appear that generating Hypo-FE distributed sample is more expensive compared to Hyper-FE distributed one due to the additional sum appearing in $R_{g_{j,i}}$ in (1.28). However, the initial vectors ($\alpha$) of the two representations are different; consequently the mean number of rejections $p^*$ are different as well. There are examples in which the Hyper-FE, and other examples in which the Hypo-FE representation gives the better performance and the difference can be significant in either direction.

## 1.6 Generating random variates from various rational arrival processes

The introduced random number generation method can be used to generate samples of various versions of rational arrival processes. The simple case is when a RAP generates single arrivals of a single type. More complex cases, BRAPs or MRAPs, arise when batch arrivals or arrivals of different types are allowed.

Generating RAP samples

When generating random variates from RAPs the state vector of the RAP has to be stored between consecutive arrivals. Thus, the procedure consists of two steps: in the first step the inter-arrival time is drawn (that is ME distributed with parameters being the current state vector and $\mathbf{H_0}$), then the new state vector is calculated just after the arrival. Considering the RAP with representation $(H_0, H_1)$ the following procedure generates a stationary series of random variates.

1: $\alpha = \tau$
2: **while** samples required **do**
3: $R =$ a random sample from ME$(\alpha, \mathbf{H_0})$
4: $\alpha = \dfrac{\alpha e^{\mathbf{H_0} R} \mathbf{H_1}}{\alpha e^{\mathbf{H_0} R} \mathbf{H_1} \mathbb{1}}$
5: **end while**

The output of the algorithm is composed by the consecutive $R$ values.

If additionally, an initial vector of the RAP is known at time 0 then, instead of the stationary initial vector, this initial vector needs to be stored in $\alpha$ in the first step of the algorithm.

Generating MRAP samples

Considering an MRAP (or equivalently a BMRAP) with representation $(H_0, H_1, \ldots, H_K)$ the following procedure generates stationary random samples of the process.

1: $\alpha = \tau$
2: **while** samples required **do**
3: $R =$ random sample from ME$(\alpha, \mathbf{H_0})$
4: **for** $k = 1$ to $K$ **do**
5: $p_k = \alpha e^{\mathbf{H_0} R} \mathbf{H_k} \mathbb{1} \;/\; \sum_{j=1}^{K} \alpha e^{\mathbf{H_0} R} \mathbf{H_j} \mathbb{1}$
6: **if** $p_k < 0$ **then**
7: **error "INVALID PROCESS !!!"**
8: **end if**
9: **end for**
10: $B =$ random sample with distribution $\{p_1, \ldots p_K\}$
11: **store** $R, B$
12: $\alpha = \dfrac{\alpha e^{\mathbf{H_0} R} \mathbf{H_B}}{\alpha e^{\mathbf{H_0} R} \mathbf{H_B} \mathbb{1}}$
13: **end while**

In this algorithm, each random sample is a pair representing the inter-arrival time $R$ and the type of the arrival (the batch size) $B$.

Similar to the previous RAP sample generation case the first step of the algorithm needs to be modified if the process starts from an initial vector different from the stationary one.

If any of these algorithms is called with a set of vectors and matrices which do not represent a valid distribution or a valid process the procedure might throw two kinds of errors: either in line 10 of Algorithm 1 or in line 7 of the MRAP algorithm (in case of MRAP simulation). The first one due to a negative density in case of ME simulation or a sample path which results in a negative density in case of RAP and MRAP simulation. The second one due to a sample path which results in that the probability a type $k$ sample is negative. Indeed simulation is one of the few available methods to check if a set of matrices define a valid ME distribution or arrival process.

## 1.7 Numerical experiments

The two methods presented in the paper have been implemented in C++ using the Eigen3 linear algebra library. During the implementation it turned out that the most time consuming step of the algorithm is the evaluation of $f_+(x)$ and $f_-(x)$ for every sample. This step is required only when the target distribution has a Hyper-FE or a Hypo-FE representation with some negative elements in the initial vector ($p^* < 1$). The computation of $f_+(x)$ and $f_-(x)$ requires the evaluation of a matrix exponential function. Our implementation uses a Jordan decomposition based solution for the matrix exponential. The decomposition step has to be performed only once during the initialization of the computation. The repeated sampling of a ME distribution requires only the calculation of as many (scalar-) exponentials as the size of the representation of the distribution. The number of the computed scalar exponentials is $\#iexp$. All the results in this section are obtained on an average PC with an Intel Core2 processor running at 3 GHz.

### *1.7.1 Generating PH distributed samples*

In this section we examine how the efficiencies of the proposed procedures compare to the play method for PH distributions. For this reason we generated a large number of random PH distributions of order 8 and executed all the procedures. All the elements of the generator and the initial vector of the PH were uniformly distributed random numbers in $(0, 1)$, except the transition rates to the absorbing state that is considered to be a free parameter (denoted by $\lambda$). With this parameter we can control the number of steps before absorption in the play method.

The average number of basic operations is summarized by Table 1.1. In case of the Hypo-FE and Hyper-FE based methods, the cost of computing the exponential function to calculate $f_+(x)$ and $f_-(x)$, if required, appears as well. The $p^*$ parameter that indicates the mean number of rejected samples

is also given in the Table. The basic operations $\#ilog$, $\#iuni$ and $\#iexp$ are meant for one iteration only. To obtain the total number of basic operations they have to be multiply by the mean number of iterations that is $1/p^*$. Interestingly, the 3000 random PH distributions generated during the experiment had a valid Hypo-FE representation in all of the cases. This way the Hypo-FE based method did not calculate the acceptance probability, $f_+(x)$ and $f_-(x)$; thus no exponential functions were computed. The Table shows that the per-iteration cost of the Hyper-FEs is the best among the compared procedures. However, most PH distributions do not have a Hyper-FE representation with $p^* = 1$. As $\lambda$ increased, some PH distributions have got a Hyper-FE representation with $p^* = 1$.

| | Play method | | Hyper-FE | | | | Hypo-FE | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\lambda$ | #uni | #log | #iuni | #ilog | #iexp | $p^*$ | #iuni | #ilog | #iexp | $p^*$ |
| 0.1 | 144.19 | 71.594 | 1.0074 | 1.0039 | 8 | 0.99724 | 16.017 | 7.6263 | 0 | 1 |
| 0.5 | 32.393 | 15.696 | 1.0377 | 1.0192 | 8 | 0.98685 | 13.791 | 6.4696 | 0 | 1 |
| 1 | 17.686 | 8.3432 | 1.0747 | 1.0378 | 8 | 0.97469 | 11.541 | 5.4732 | 0 | 1 |
| 2 | 10.703 | 4.8514 | 1.1331 | 1.0649 | 8 | 0.95899 | 8.8631 | 4.3851 | 0 | 1 |
| 4 | 7.0355 | 3.0178 | 1.1992 | 1.099 | 7.984 | 0.93797 | 6.1525 | 3.4279 | 0 | 1 |
| 8 | 5.1654 | 2.0827 | 1.1892 | 1.0945 | 7.808 | 0.94059 | 4.0318 | 2.7877 | 0 | 1 |

**Table 1.1** Number of basic operations required in case of random PH distributions

The results of the actual implementations are depicted in Figure 1.4.

The figure indicates that the play method is very sensitive to the number of steps taken before absorption, while the Hypo-FE and Hyper-FE based methods provide an almost constant performance. Interestingly, in spite of the larger cost per iteration, the Hypo-FE based method provides better performance than Hyper-FE based one in several cases because that representation gives better acceptance probability thus less rejections are required. We can conclude this numerical experiment of generating PH distributed random samples that the Hypo-FE and Hyper-FE based methods provide a better performance than the play method if the PH takes several steps until absorption.

### *1.7.2 Generating ME distributed samples*

Consider the ME distribution with representation $(\tau, \mathbf{T})$, where

$$\tau = \{7.69231, -6.69231, 0\}, \quad \mathbf{T} = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -3 & 1 \\ 0 & -1 & -3 \end{pmatrix}.$$

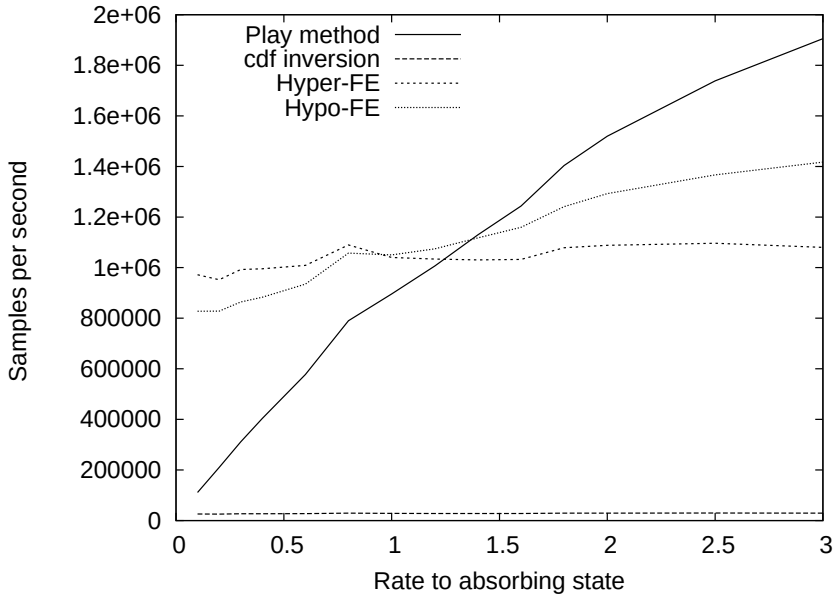


**Fig. 1.4** Random samples per second in case of random PH distributions

Its probability density function is depicted in Figure 1.5. The eigenvalues of

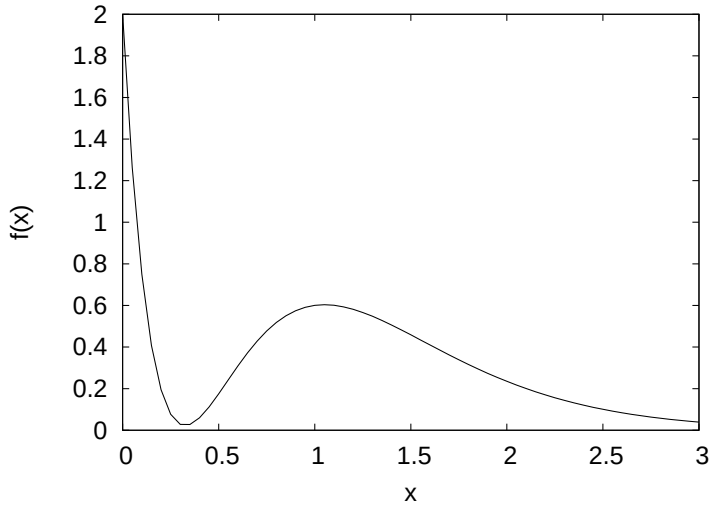


**Fig. 1.5** Probability density function of ME distribution with representation $(\tau, \mathbf{T})$

$\mathbf{T}$ are $\{-2, -3+1\mathbb{i}, -3-1\mathbb{i}\}$, and the corresponding feedback Erlang blocks (in both the Hyper-FE and the Hypo-FE representations) are

$$\mathbf{M_1} = -2, \qquad \mathbf{M_2} = \begin{bmatrix} -\sigma & \sigma & 0 \\ 0 & -\sigma & \sigma \\ z\sigma & 0 & \sigma \end{bmatrix}, \tag{1.32}$$

with $\sigma = 2.42265, z = 0.108277$. The transformation matrix to the Hyper-FE and Hypo-FE representations are obtained based on (1.19).

$$W^{(hyper)} = \begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & -0.46943 & 0.543647 & 0.925783 \\ 0. & -0.0281766 & -0.82339 & 1.85157 \end{bmatrix},$$
$$W^{(hypo)} = \begin{bmatrix} -0.11547 & 0.0281766 & 0.16151 & 0.925783 \\ 0 & -0.46943 & 0.543647 & 0.925783 \\ 0 & -0.0281766 & -0.82339 & 1.85157 \end{bmatrix}.$$

Based on these transformation matrices the initial vectors of the Hyper-FE and Hypo-FE representations are

$$\alpha^{(hyper)} = \begin{bmatrix} 7.69231 & 3.14157 & -3.63825 & -6.19563 \end{bmatrix},$$
$$\alpha^{(hypo)} = \begin{bmatrix} -0.888231 & 3.35832 & -2.39587 & 0.925783 \end{bmatrix}.$$

Based on the initial vectors the mean number of required iterations can be obtained as

$$1/p^{*(hyper)} = \sum_{i \in \mathcal{A}_+} \alpha^{(hyper)} = 10.83388,$$
$$1/p^{*(hypo)} = 4.284103,$$

thus, more than twice as many rejections occur when using the Hyper-FE structure.

To illustrate the behaviour of Algorithm 1 Figure 1.6 and 1.7 depict the density to draw samples from, $f_+(x)$, and the acceptance probability function, $p_{accept}(x)$, respectively. It can be observed that the $f_+(x)$ density corresponding to the Hypo-FE representation captures the behavior of the original pdf better; thus the acceptance probabilities are higher. It can also be observed that the original pdf approaches 0 at around $x = 0.32$, this behaviour can not be captured by the PH distribution of low order that is why the acceptance-rejection method is required. The acceptance probability function takes very low value to ensure the low density of the samples around $x = 0.32$.

The number of basic operations per random sample and the overall performance of the methods are summarized in Table 1.2. The Hypo-FE based method is 5 times faster than the cdf inversion based method for this example.

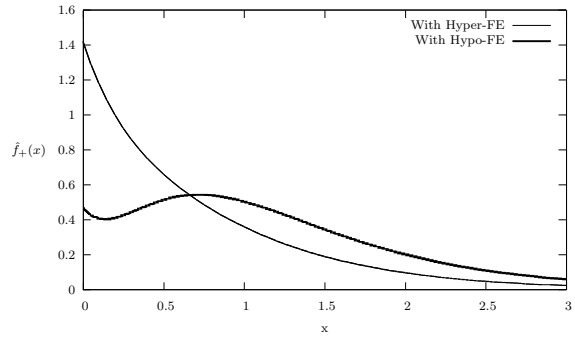


**Fig. 1.6** Probability density function $\hat{f}_+(x)$

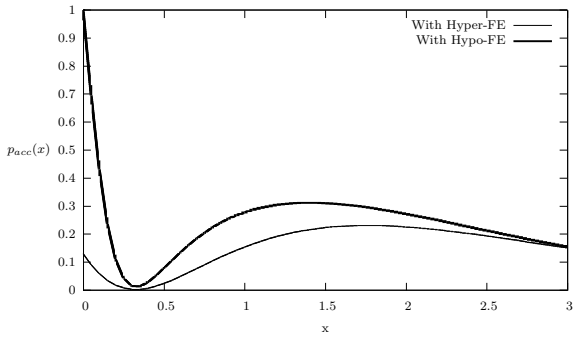


**Fig. 1.7** Probability of accepting sample $x$

| Method | #uni | #log | #exp | $p^*$ | Samples/sec |
|---|---|---|---|---|---|
| Cdf inversion | 1 | 0 | 324.83 | n/a | **54869** |
| Hyper-FE | 28.998 | 13.428 | 31.681 | 0.094693 | **179560** |
| Hypo-FE | 27.51 | 8.022 | 12.033 | 0.24932 | **277581** |

**Table 1.2** Comparison of three methods for generating ME distributed samples

### *1.7.3 Generating RAP samples*

From Section 1.6 it is obvious that random samples from a RAP can be generated efficiently once we have an efficient method to draw ME distributed random numbers. Through the previous two examples the behavior of the presented acceptance-rejection methods have been studied and compared in details. Here we provide a simpler example to demonstrate the efficiency of our methods for generating samples from a RAP.

The matrices of the RAP used in this example are as follows:

$$\mathbf{H_0} = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -3 & 1 \\ 0 & -1 & -2 \end{bmatrix}, \qquad \mathbf{H_1} = \begin{bmatrix} 1.8 & 0.2 & 0 \\ 0.2 & 1.8 & 0 \\ 0.2 & 1.8 & 1 \end{bmatrix}. \tag{1.33}$$

A significant performance hit over the ME distributed random number generators is that a matrix exponential function has to be evaluated after drawing a sample to calculate the initial state vector for the next arrival. However, this time consuming step is required in all methods for generating random variates. Consequently, we expect lower performance than in case of ME distributed random sample generation, but according to Table 1.3 the Hyper-FE and the Hypo-FE based methods are still 6 times faster than the cdf inversion based one in this particular example. The number of basic operations is omitted since they vary with the initial vector in each step.

| Method for ME | Samples/sec |
|---|---|
| Cdf inversion | 55872 |
| Hyper-FE | 336247 |
| Hypo-FE | 329224 |

**Table 1.3** Comparison of three methods for generating RAP samples

## 1.8 Conclusions

The paper proposes acceptance rejection based numerical methods for generating ME, RAP and MRAP samples. The key of the numerical efficiency of the acceptance rejection based methods is the high acceptance probability and the low computational cost of elementary random number generation. Numerical investigations show that both of these elements depend on the representation of the models. We investigated the efficiency of two FEB based representations, which were relatively efficient in a wide range of cases, but optimal representations of these models, which make the simulation most efficient are still open research problems.

## References

[Asmussen and Bladt(1999)] Asmussen S, Bladt M (1999) Point processes with finite-dimensional conditional probabilities. Stochastic Processes and their Application 82:127–142

[Bean and Nielsen(2010)] Bean NG, Nielsen BF (2010) Quasi-birth-and-death processes with rational arrival process components. Stochastic Models 26(3):309–334

[Brown et al(1998)Brown, Place, and de Liefvoort] Brown E, Place J, de Liefvoort AV (1998) Generating Matrix Exponential Random Variates. Simulation 70:224–230

[Buchholz and Telek(2010)] Buchholz P, Telek M (2010) Stochastic Petri nets with matrix exponentially distributed firing times. Performance Evaluation 67(12):1373–1385

[Buchholz and Telek(2011)] Buchholz P, Telek M (2011) On minimal representations of rational arrival processes. Annals of Operations Research To appear, DOI 10.1007/s10479-011-1001-5

[Cumani(1982)] Cumani A (1982) On the canonical representation of homogeneous Markov processes modelling failure-time distributions. Microelectronics and Reliability 22:583–602

[He and Neuts(1998)] He QM, Neuts M (1998) Markov arrival processes with marked transitions. Stochastic Processes and their Applications 74:37–52

[Kronmal and Peterson(1979)] Kronmal R, Peterson A (1979) On the alias method for generating random variables from a discrete distribution. The American Statistician 33(4):214–218

[Latouche and Ramaswami(1999)] Latouche G, Ramaswami V (1999) Introduction to Matrix-Analytic Methods in Stochastic Modeling. Series on statistics and applied probability, ASA-SIAM

[van de Liefvoort(1990)] van de Liefvoort A (1990) The moment problem for continuous distributions. Tech. rep., University of Missouri, WP-CM-1990-02, Kansas City

[Lipsky(1992)] Lipsky L (1992) Queueing Theory: A linear algebraic approach. MacMillan, New York

[Mitchell(2001)] Mitchell K (2001) Constructing a correlated sequence of matrix exponentials with invariant first order properties. Operations Research Letters 28:27–34

[Mocanu and Commault(1999)] Mocanu S, Commault C (1999) Sparse representations of phase-type distributions. Commun Stat, Stochastic Models 15(4):759 – 778

[Neuts(1981)] Neuts MF (1981) Matrix-Geometric Solutions in Stochastic Models. An Algorithmic Approach. Dover Publications, Inc., New York

[Neuts and Pagano(1981)] Neuts MF, Pagano ME (1981) Generating random variates from a distribution of phase type. In: WSC '81: Proceedings of the 13th conference on Winter simulation, IEEE Press, Piscataway, NJ, USA, pp 381–387

[Reinecke et al(2009)Reinecke, Wolter, Bodrog, and Telek] Reinecke P, Wolter K, Bodrog L, Telek M (2009) On the cost of generating PH-distributed random numbers. In: Int. Workshop on Performability Modeling of Computer and Communication Systems (PMCCS), Eger, Hungary, pp 1 – 5

[Reinecke et al(2010)Reinecke, Telek, and Wolter] Reinecke P, Telek M, Wolter K (2010) Reducing the cost of generating APH-distributed random numbers. In: 15th Int. Conf. on Measurement, Modelling and Evaluation of Computing Systems (MMB), Springer, Essen, Germany, LNCS, vol 5987, pp 274–286

[Robert and Casella(2004)] Robert C, Casella G (2004) Monte Carlo Statistical Methods. Springer-Verlag, New-York

[Telek and Horváth(2007)] Telek M, Horváth G (2007) A minimal representation of Markov arrival processes and a moments matching method. Performance Evaluation 64(9-12):1153–1168