RANDOM SIMULATIONS, AZÉMA MARTINGALES AND STRUCTURE EQUATIONS

Anthony Phan

April 21, 2004

TABLE OF CONTENTS

Introduction 1

1. Purpose of those random simulations 1

2. The old C random number generator 2

3. The BSD random generator 2

4. The METAFONT/POST random generator 3

Histograms of simulations of Azéma martingales starting from 0 at time 1 4

Introduction 4

1. C program 4

2. Histograms (Sas) 5

Simulations of trajectories of solutions of Markovian structure equations 8

Introduction 8

1. METAPOST program 8

2. $\phi(x) = \mathrm{sgn}(x)\,|x|^{0.25}$ 8

2.1. *Trajectories on a long time interval* 8

2.2. *Trajectories on a short time interval* 8

3. $\phi(x) = \mathrm{sgn}(x)\,|x|^{0.5}$ 9

3.1. *Trajectories on a long time interval* 9

3.2. *Trajectories on a short time interval* 9

4. $\phi(x) = \mathrm{sgn}(x)\,|x|^{0.75}$ 10

4.1. *Trajectories on a long time interval* 10

4.2. *Trajectories on a short time interval* 10

5. $\phi(x) = \mathrm{sgn}(x)\,|x|^{1}$ 11

5.1. *Trajectories on a long time interval* 11

5.2. *Trajectories on a short time interval* 11

6. $\phi(x) = \mathrm{sgn}(x)\,|x|^{1.25}$ 12

6.1. *Trajectories on a long time interval* 12

6.2. *Trajectories on a short time interval* 12

7. $\phi(x) = \mathrm{sgn}(x)\,|x|^{1.5}$ 13

7.1. *Trajectories on a long time interval* 13

7.2. *Trajectories on a short time interval* 13

Appendices 14

A. C PROGRAM 14

B. METAPOST PROGRAM 15

Bibliography 18

INTRODUCTION

This document is quite unstable: every part of it certainly deserves some quite long explanations, but also many other parts have to be written.

1. **Purpose of those random simulations**

Some processes are well-known: nobody really needs random simulations to visualize Brownian trajectories or the law profile of a Brownian motion starting from 0 and taken at time 1. In this case simulations just help drawing graphs.

As one can derive from a celebrated Lévy theorem, linear Brownian motion with initial law μ on $\mathbb{R}$ is the unique (in law) real martingale $X = (X_t)_{t\geqslant 0}$ with initial law μ and quadratic variation equal to the time:

$$[X, X]_t = t, \qquad t > 0.$$

In [1], Émery introduces a class of martingale problems that extends the case of Brownian motion called structure equations. Let Φ a predictable process over a filtered probability space $(\Omega, \mathcal{A}, \mathbb{P}, \mathcal{F})$. A solution of the structure equation associated to Φ is a local martingale X defined on the same filtered probability space such that

$$[X, X]_t = t + \int_0^t \Phi_s \, \mathrm{d}X_s, \qquad t > 0,$$

or, written into a differential form,

$$\mathrm{d}[X, X]_t = \mathrm{d}t + \Phi_t \, \mathrm{d}X_t, \qquad t > 0.$$

Our main interest concerns Markovian structure equations: the predictable process Φ is equal to $\phi(X_-)$ for a Borelian map $\phi : \mathbb{R} \to \mathbb{R}$ and X is the coordinate process of the Skorohod space Ω which is endowed with its usual right-continous filtered structure (and which may also be universally completed). Thus, a solution of the Markovian structure equation associated to ϕ

$$\mathrm{d}[X, X]_t = \mathrm{d}t + \phi(X_{t-}) \, \mathrm{d}X_t, \qquad t > 0,$$

is any probability law over the Skorohod space such that the coordinate process is a local martingale satisfying the previous equation. The questions of existence and uniqueness arise and only partial answers are known (*see* references).

Those problems are of theoritical nature and random simulations can give no proof of whatsoever. A good reason for that is that simulations are based on discrete schemes, and, at least in the context of structure equations, existence and uniqueness always holds in the discrete case.

But random simulations are helpful to visualize phenomena that are, even in a uniqueness case, hard to understand or analyze. A first situation is the one when it is quite tedious or even impossible to compute explicit laws densities, or, worse, to prove the existence of densities. Sampling a process at some fixed time allows to graphically test such a regularity condition. It also allows to evaluate the weights

of tail distributions and have an idea of the related moment problems. Another situation is the following: suppose that you are quite sure that uniqueness holds but you don't know how to prove it since it should rely on some infinitesimal behavior of a solution at a singular point. Simulations of the behavior of solutions close to a singular point would help to find how to prove that uniqueness effectively holds.

2. The old C random number generator

The C random number generator was based on the linear congruential method

$$x_n = (a \times x_{n-1} + c) \bmod m, \qquad n \geqslant 1,$$

where, by default, the multiplier a is equal to 25 214 903 917, the increment c equal to 11 and the modulus $m = 2^{48}$. Thus, those generators deals with 48 bits long integers. The properties of such sequences are well known (*see* [6], chap. 3), among them: their period is 2^{48} and they are uniformly distributed in $\{0, 1, \ldots, 2^{48} - 1\}$.

A method for using such pseudo-random sequence is considering only one x_n at time n. This is what is done with `srand48` and `rand48`. Seeds given as arguments to `srand48` or integers output by `rand48` are only 32 bits long. They are the 32 higher bits of the numbers appearing in the method.

`srand48(`$\langle int \rangle$`)` (formerly `srand(`$\langle int \rangle$`)`). This function initializes the actual random seed to the value $\langle int \rangle \times 2^{16} + 1370$, this last number being arbitrary.

`lrand48()` (formerly `rand()`). This function computes the next pseudo-random number and returns its 32 higher bits, *i.e.*, an integer in $\{0, 1, \ldots, 2^{32} - 1\}$.

Other such external functions are defined in `stdlib.h`. These functions seem to be now depreceated. By the way, one should recall that those kind of random generators are certainly the best ones among those whose behavior is completely known. Other generators such as the next ones have, it seems, good empirical properties but since few of their mathematical properties are known it may be illegitimate to use them. Also, many professional software still use linear congrential sequences (for instance, SAS, Maple, ...).

3. The BSD random generator

The BSD, and then the gcc, random number generator is based on a much more complex method than the previous one. Alas, it is quite hard to find a precise and complete description of it. The following explanation is based on what I've guessed from `random.c` of the standard C library and what I've read in [6].

In fact, this generator is quite similar to the METAFONT/POST one: it is based on *lagged Fibonacci sequences*

$$x_n = (x_{n-k} + x_{n-\ell}) \bmod m, \qquad n \geqslant \ell$$

where k, ℓ, m are integers such that, $k \leqslant l$. The main difference is that are provided different rules for such sequences. These rules are selected with `initstate` or `setstate`.

(i) `srandom(`$\langle int \rangle$`)` (same as `srand(`$\langle int \rangle$`)`). This function does the same job as `srand` except that it also computes the $\ell - 1$ next terms of the sequence.

(ii) `random()` (same as `rand()`). This function compute the next pseudo-random number and returns its 32 higher bits which is an integer lying in $\{0, 1, \ldots, 2^{32} - 1\}$.

(iii) `initstate(`⟨*int*⟩`,`⟨*char*⟩`,`⟨*size_t*⟩`)`. The argument ⟨*size_t*⟩ is the size of the stack used for these sequences, i.e., ℓ. Its possible values are 8, 32, 64, 128, 256. The argument ⟨*char*⟩ should be the name of the stack, the argument ⟨*int*⟩ the initial value. The whole stack is then filled with some complicated method.

(iv) `setstate(`⟨*char*⟩`)`. The size of the stack being already defined through `initstate`, one can reset the stack with `setstate`.

Frankly, it is hard to understand what really lies under those control sequences. Also their implementation may depend on every Unix and such distributions.

4. The METAFONT/POST random generator

The random number generator implemented in METAFONT/POST is the one described in Subsection 3.2.2, p. 27–29, of [6]. It is based on a *lagged Fibonacci sequence*

$$x_n = (x_{n-24} + x_{n-55}) \bmod m, \qquad n \geqslant 55,$$

with $x_0, \ldots, x_1$ given. The modulus m is equal to 2^{28} in METAFONT/POST. Then, the period of such a sequence is $2^{27}(2^{55} - 1)$ which is very much larger than periods one can obtain usually with linear congruential methods. But its other properties as a pseudo-random sequence are not quite well known. The primitives related to random number generation are:

`randomseed :=` ⟨*numeric*⟩. This primitive is rather particular since its syntax makes it look like an assignment, even if no numerical variable named "randomseed" is accessible. Its effect is the initialization of the row $x_0, \ldots, x_{54}$ based on the numerical value given as argument. If no such assignment is given, initialization will be based on time and date. A first row is computed with respect to the seed and $x_{164}, \ldots, x_{219}$, the terms of the corresponding lagged Fibonacci sequence, are shifted back to $x_0, \ldots, x_{54}$. One has no direct access to the row $x_0 \ldots, x_{54}$ or to any of its entries.

`uniformdeviate` ⟨*numeric*⟩. This function outputs a pseudo-random number approximatively uniformly distributed in the real interval with bounds 0 and the numerical argument. If every entries of the row $x_0, \ldots, x_{54}$ have been used, the next 55 terms of the lagged Fibonacci sequence are computed. The last unused x_i is used to compute the result in a straightforward way.

`normaldeviate`. This primitive function with no argument outputs some pseudo-random number choosen with respect to the standard normal distribution. It makes some kind of repeated Bernoulli trials and then calls and refresh the row as many time as necessary to obtain some acceptable value (*see* [6], p. 130–132).

HISTOGRAMS OF SIMULATIONS OF AZÉMA MARTINGALES STARTING FROM 0 AT TIME 1

Introduction

For $\beta \in]-2, 0]$, K. R. Parthasarathy has proved that the law at every time of the Azéma martingale of parameter β starting from 0 admits a density and he has given a recurrent procedure for computing it (*see* [8]). The proofs of Parthasarathy rely on a key argument: for such β these laws have a compact support and then their moments determinate the laws completely. He has discovered that successive moments satisfy integral relations; from the moments property those relations extend to the laws themselves and show the absolute continuity.

It is well known that for $\beta = -2$ at a given time the law of such a martingale (or more generally of an Azéma martingale of parameter $\beta = -2$ and with a deterministic initial condition) is carried by two points (which belong to a parabola determined by β and the initial condition). It is also well known that for any value of the parameter and most—all when $\beta < 0$, none when $\beta = 0$—initial value, the law is not absolutely continuous with respect to the Lebesgue measure.

We do presume that the absolute continuity is satisfied for every value of $\beta \in \mathbb{R}\backslash\{-2\}$. In order to confirm ourself in this idea but also to visualize (at least!) the shape of these densities, we have proceed to random simulations based on a discrete approximation of these martingales as it is described in the paper of Meyer [7]. Start from the Markovian structure equation

$$\mathrm{d}[X, X]_t = \mathrm{d}t + \Phi_t \, \mathrm{d}X_t, \qquad t > 0;$$

discretize it to obtain

$$(\Delta X_t)^2 = \Delta t + \Phi_{t-\Delta t} \, \Delta X_t, \qquad t \in \mathbb{Z}^* \Delta t = \{\Delta t, 2\Delta t, 3\Delta t, \ldots\};$$

solve this equation in order to obtain

$$X_t^{\pm} = X_{t-\Delta t} + \frac{-\Phi_{t-\Delta t} \pm \sqrt{\Phi_{t-\Delta t}^2 + 4\Delta t}}{2}.$$

and choose one the two next values in order to obtain a martingale, i.e., $\mathbb{E}[\Delta X_t] = 0$, independently of what could have happen before, that is only upon the value of $\Phi_{t-\Delta t}$. What we obtain a discrete martingale that approximates a solution of the continuous case.

1. C program

The following program is written in C. It gets the parameters β, x_0 and n and create a file `azema.dat` whose content is n lines, each one of these consisting of an outcome of X_1.

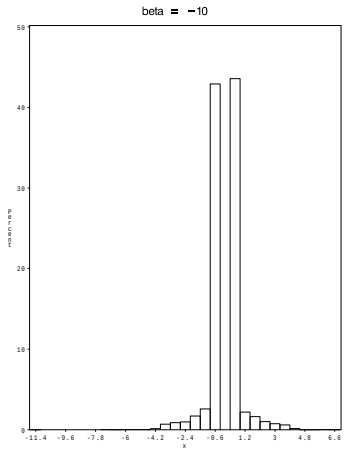
It is worth to notice that for every outcome, 10 000 calls to the random generator are made ($\Delta t = 0.0001$), so, for a whole data set, 10^8 such calls are made. The

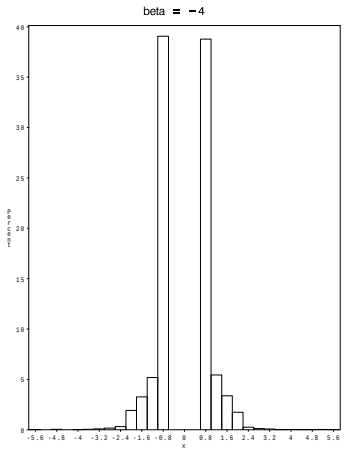
random generator used with the C Compiler is said to have a period approximatively equal to $16 \times ((2^{31}) - 1)$ which is (only!) about 300 times larger than 10^8.

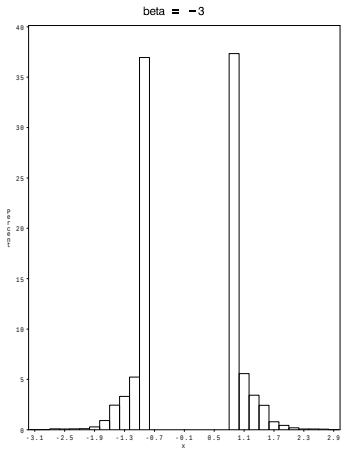
2. **Histograms** (Sas)

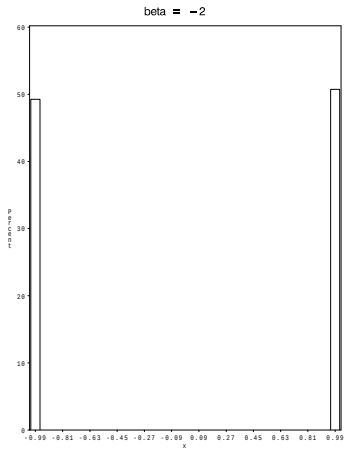
For every histogram, $n = 10\,000$ values have been sampled. Since the size of the data files is quite large, we decided to use a specific software for drawing the histograms: Sas.

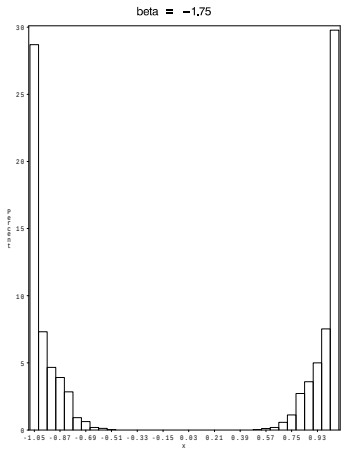
One shall remark that for the three cases for which the laws are explicitely known—$\beta \in \{-2, -1, 0\}$—the adequation of the histograms with these ones. (Let us recall that the density for $\beta = -1$ is $x \mapsto 1/(\pi\sqrt{2 - x^2})$.)

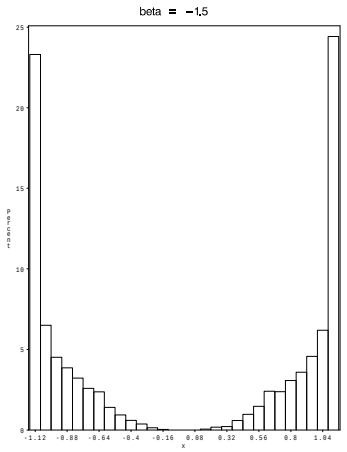


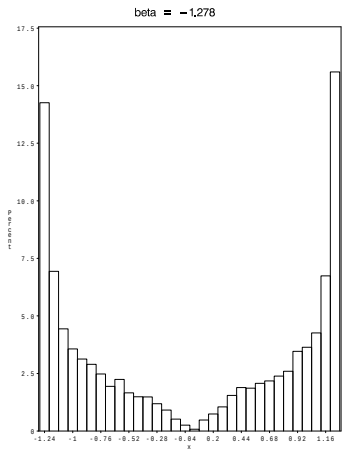


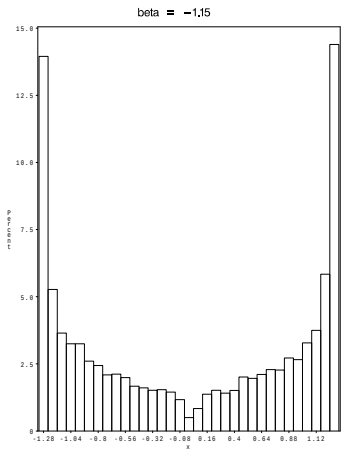


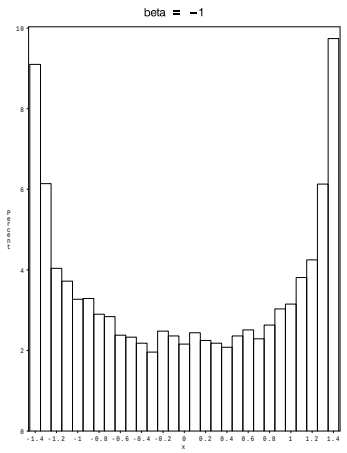


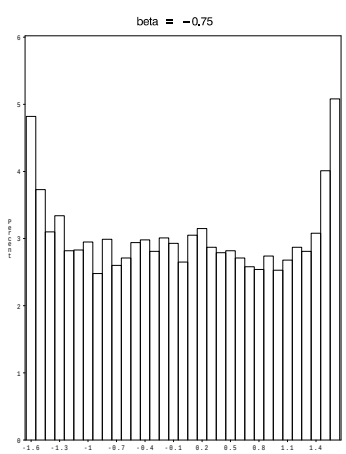


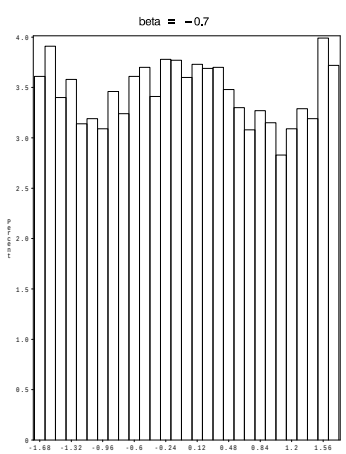


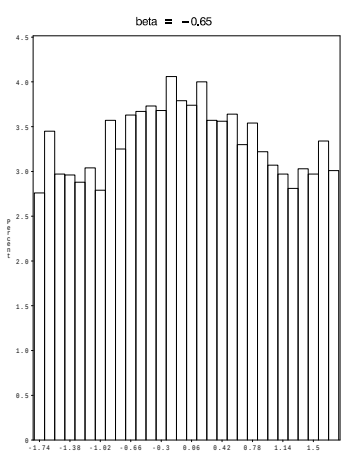

beta = −1,278
PERCENT
x

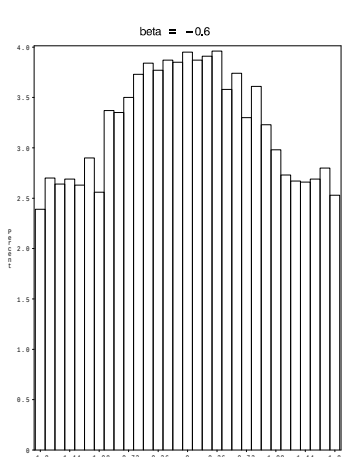

beta = −1,15
PERCENT
x

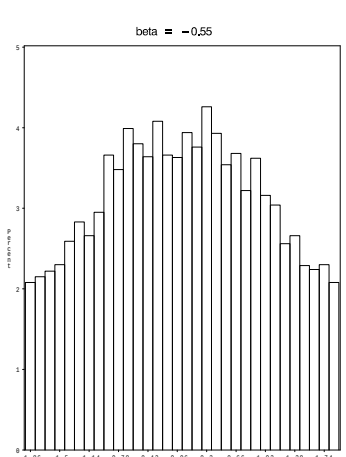

beta = −1
PERCENT
x

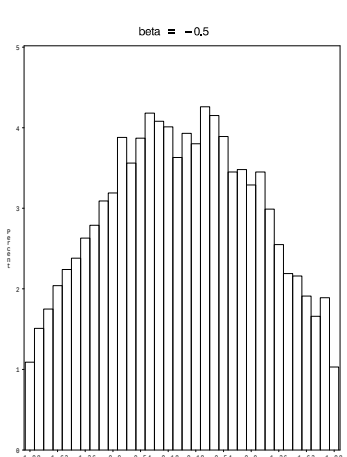

beta = −0,75
PERCENT
x

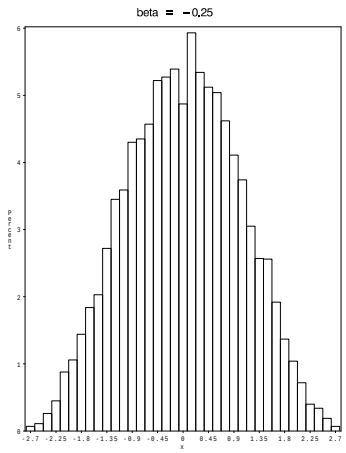

beta = −0,7
PERCENT
x

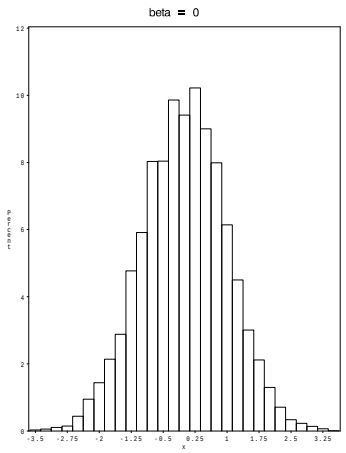

beta = −0,65
PERCENT
x

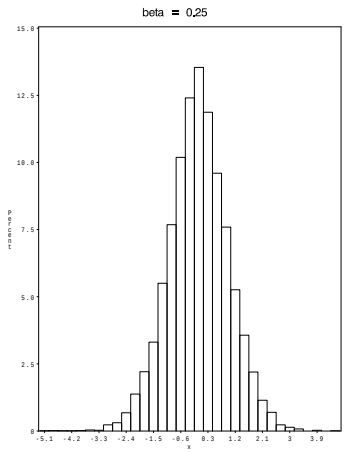

beta = −0,6
PERCENT
x

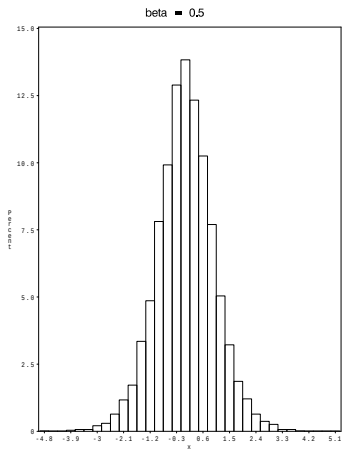

beta = −0,55
PERCENT
x

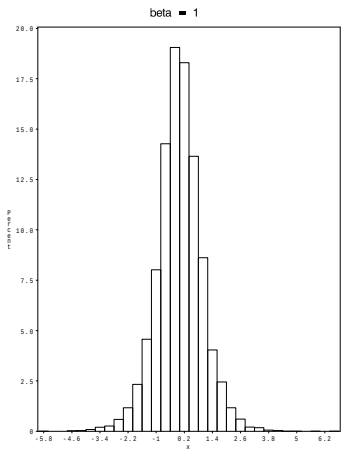

beta = −0,5
PERCENT
x

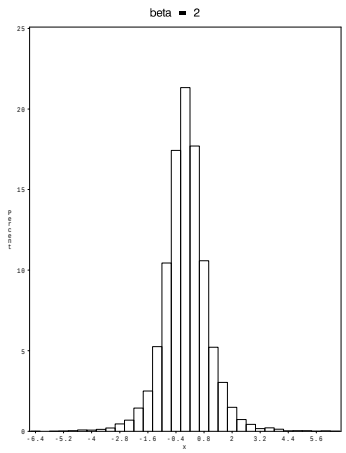

beta = −0.25
PERCENT
x

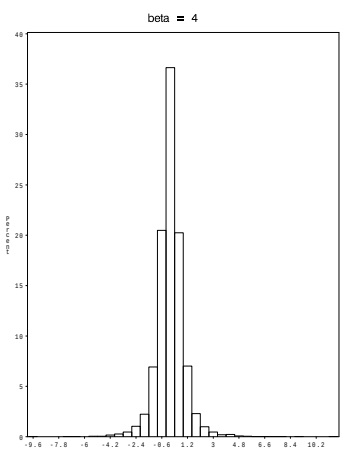

beta = 0
PERCENT
x

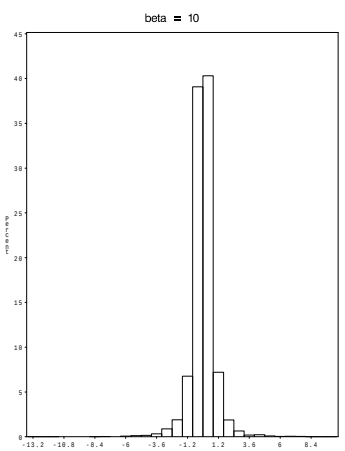

beta = 0.25
PERCENT
x


beta = 0.5
PERCENT
x


beta = 1
PERCENT
x


beta = 2
PERCENT
x


beta = 4
PERCENT
x


beta = 10
PERCENT
x

SIMULATIONS OF TRAJECTORIES OF SOLUTIONS OF MARKOVIAN STRUCTURE EQUATIONS

Introduction

The aim of these simulations is to examinate how a solution of a Markovian structure equation

$$\mathrm{d}[X,X]_t = \mathrm{d}t + \phi(X_{t-})\,\mathrm{d}X_t$$

behaves in a neighbourhood of a vanishing point of its parameter-function ϕ. Here, the parameter-function is such that the integral flow associated, *i.e.*, the flow of

$$0 = \mathrm{d}t + \phi(x)\,\mathrm{d}x,$$

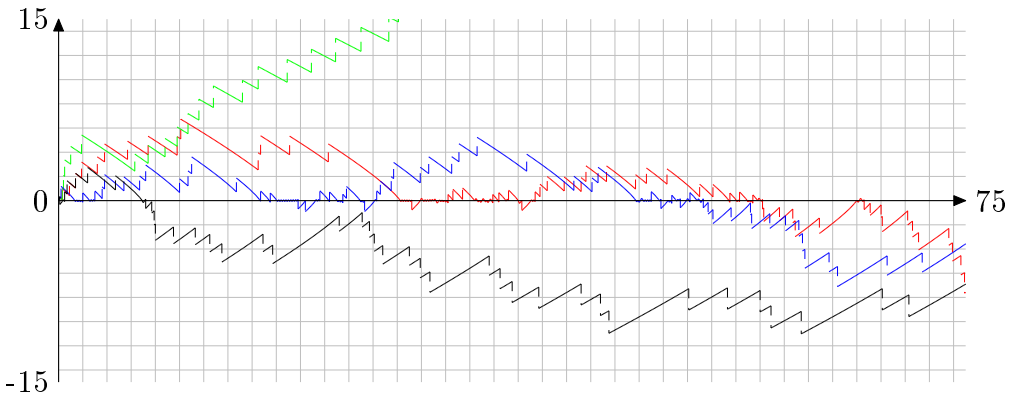
collapse on the singularity. The curious case is the one for which $1/\phi$ is integrable on this singularity since then a solution in a neighbourhood of this singularity has a strictly positive probability to reach it without jumping and, conversely, has hard time to escape it.

1. METAPOST program

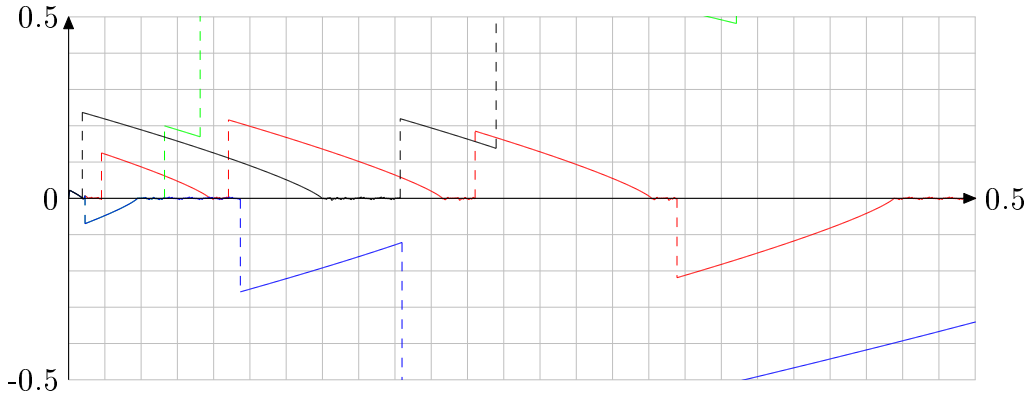
The following program is written in METAPOST (J. D. Hobby, D. E. Knuth, *see* [3, 5]). Its syntax relies on `plain.mp` and on `statsmac.mp` (A. Phan), this last program is used only fo defining scale and drawing the background grid.

2. $\phi(x) = \mathrm{sgn}(x)\,|x|^{0.25}$

2.1. Trajectories on a long time interval

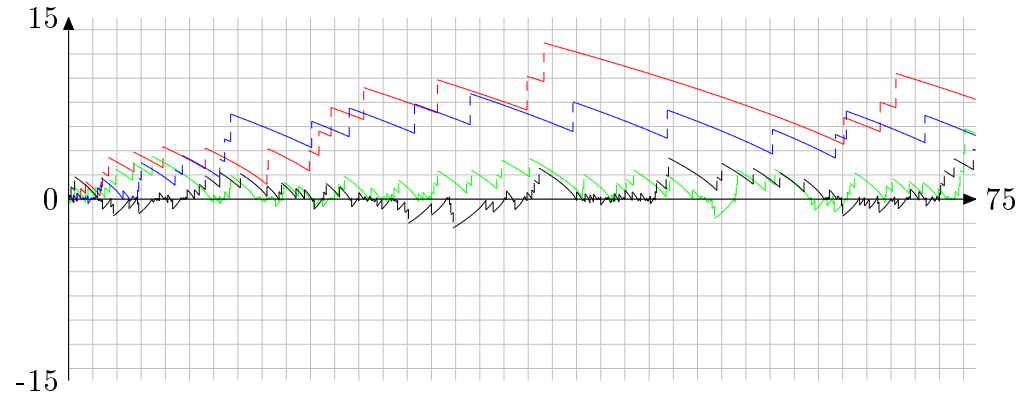


2.2. Trajectories on a short time interval

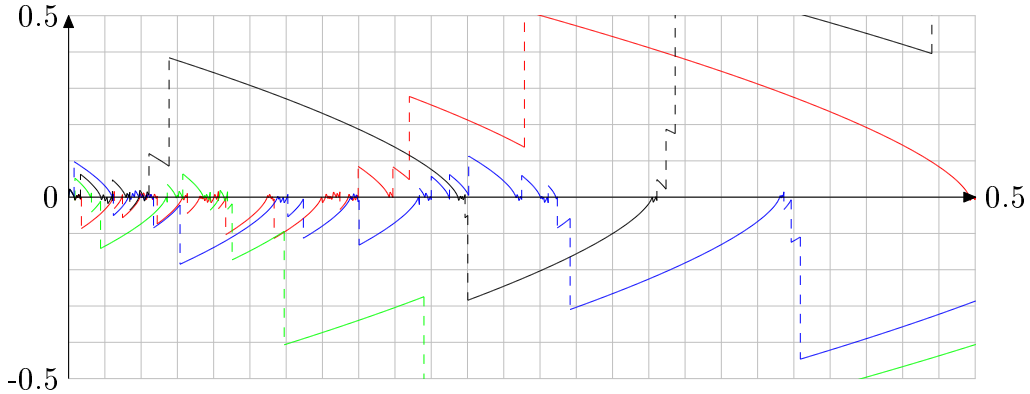


3. $\phi(x) = \operatorname{sgn}(x)\,|x|^{0.5}$

3.1. Trajectories on a long time interval

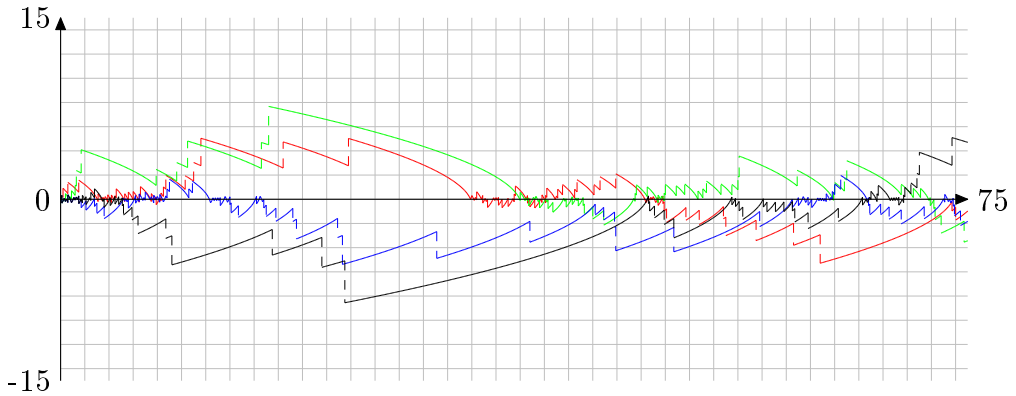


3.2. Trajectories on a short time interval

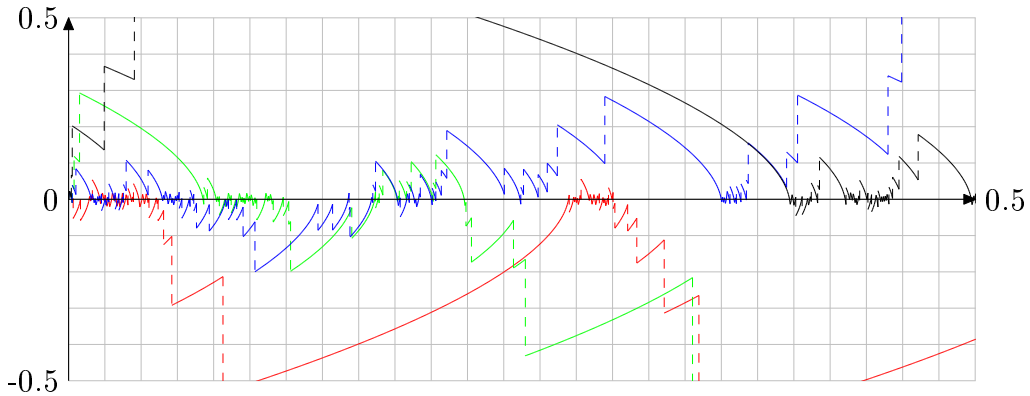


4. $\phi(x) = \operatorname{sgn}(x)\,|x|^{0.75}$

4.1. Trajectories on a long time interval

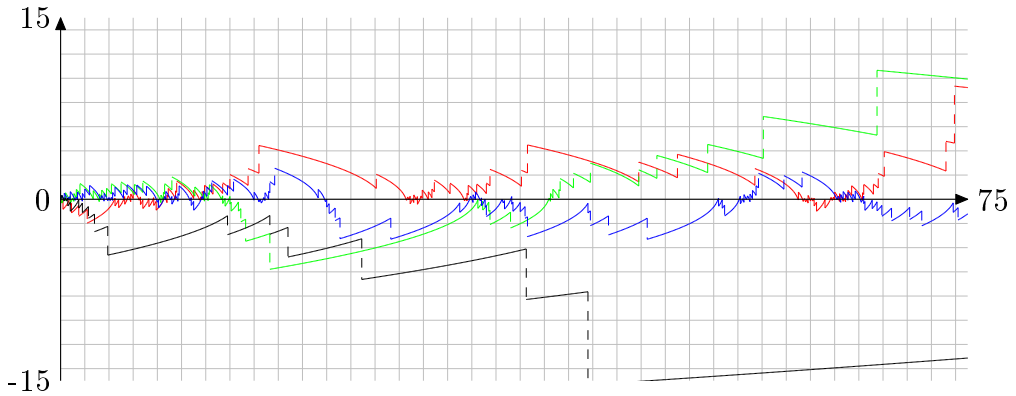


4.2. Trajectories on a short time interval

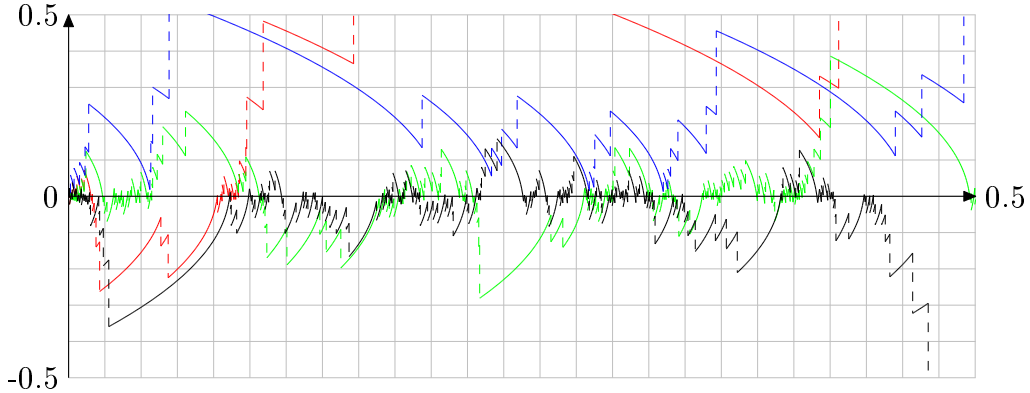


5. $\phi(x) = \operatorname{sgn}(x)\,|x|^1$

5.1. Trajectories on a long time interval

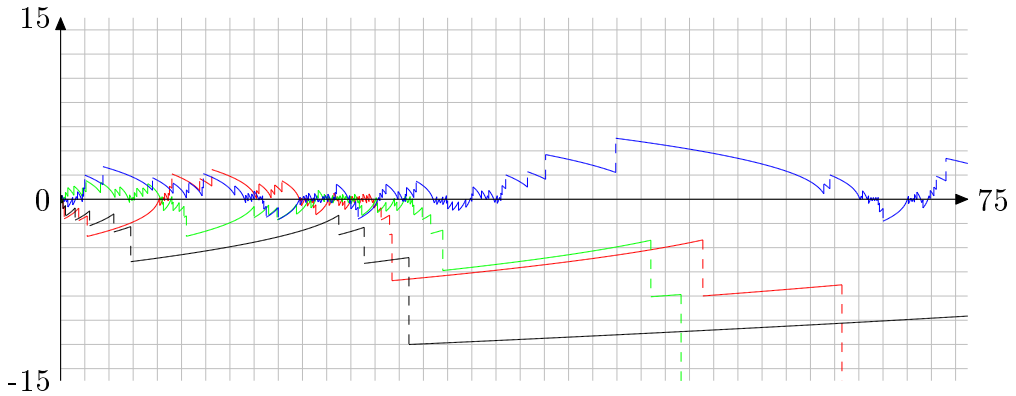


5.2. Trajectories on a short time interval

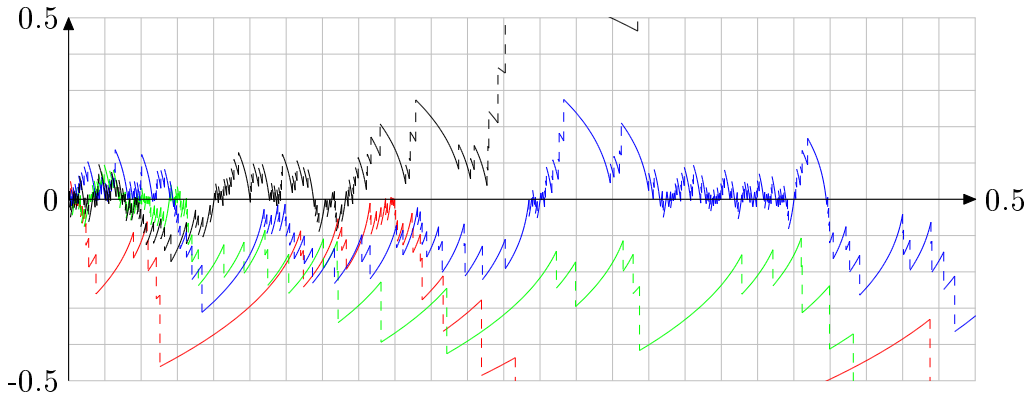


6. $\phi(x) = \operatorname{sgn}(x)\,|x|^{1.25}$

6.1. Trajectories on a long time interval

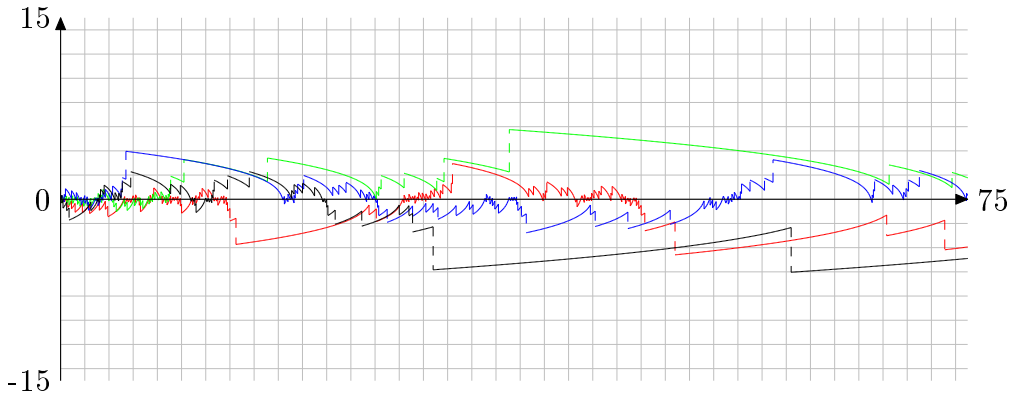


6.2. Trajectories on a short time interval

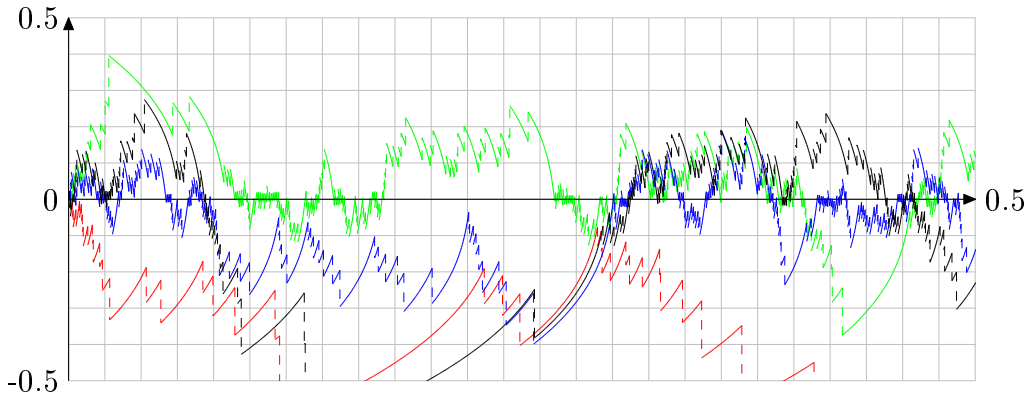


7. $\phi(x) = \operatorname{sgn}(x)\,|x|^{1.5}$

7.1. Trajectories on a long time interval



7.2. Trajectories on a short time interval



BIBLIOGRAPHY

[1] Émery (M.), "On the Azéma Martingales", *Séminaire de Probabilités XXIII*, Lecture Notes in Mathematics 1372, Springer (1989), p. 66–87.

[2] _____, "Sur les martingales d'Azéma (suite)", *Séminaire de Probabilités XXIV*, Lecture Notes in Mathematics 1426, Springer (1990), p. 442–447.

[3] Hobby (J. D.), *A User's Manual for* METAPOST, CSTR 162, AT&T Bell Labs.

[4] Hobby (J. D.), *Drawing Graphs with* METAPOST, CSTR 164, AT&T Bell Labs, TUGboat (the journal of the TeX Users Group).

[5] Knuth (D. E.), *The* METAFONT*book*, Addison Wesley (1993).

[6] Knuth (D. E.), *The Art of Computer Programming, volume 2, Seminumerical Algorithms*, third edition, Addison–Wesley (1998).

[7] Meyer (P.-A.), "Construction de solutions d''équations de structure'", *Séminaire de Probabilités XXIII*, Lecture Notes in Mathematics 1372, Springer (1989), p. 142–145.

[8] Parthasarathy (K. R.), "Azéma martingales and quantum stochastic calculus", *Proceedings of the R. C. Bose Symposium on Probability, Statistic and Design of Experiments*, Wiley Eastern (1990), p. 551–569.

[9] _____, "Azéma martingales with drift", *Probability and Mathematical Statistics*, Vol. 15 (1995), p. 461–468.

[10] Phan (A.), "Martingales d'Azéma asymétriques. Description élémentaire et unicité", *Séminaire de Probabilités XXXV*, Lecture Notes in Mathematics 1755, Springer (2001), p. 48–86.

[11] _____, *Contribution à l'étude des équations de structure markoviennes*, thèse de doctorat, Université Louis Pasteur (2000), `http://www-irma.u-strasbg.fr/irma/publications/2000/00046.shtml`.