Numeriske Løsninger av Stokastiske Differensialligninger

Erik Nesvold

Master i fysikk og matematikk
Oppgaven levert: Juni 2010
Hovedveileder: Arvid Næss, MATH

Norges teknisk-naturvitenskapelige universitet
Institutt for matematiske fag

Oppgavetekst

Numeriske løsninger av stokastiske differensialligninger anvendt i fysikk og finans løses med forskjellige numeriske skjema og sammenlignes, med hovedvekt på Path Integration by FFT.

Oppgaven gitt: 14. desember 2009
Hovedveileder: Arvid Næss, MATH

Preface

This thesis (hopefully) completes the requirements of the course TMA 4905 Statistikk, Masteroppgave. I would like to thank Dr Carlos Lourenco at CERN for letting me come as a technical student to the CMS experiment. Even though doing night shifts in the control room has not always been as fun, it has been a very inspiring stay and I have met a lot of interesting people and listened to Nobel prize winners and Stephen Hawking talk about their research. Life in Geneva has also been pleasant, with a very enjoyable climate and the fantastic Alps just around the corner.

I would also like to thank my supervisor at NTNU, Professor Arvid Naess for providing a very interesting direction to my thesis. Although I spent most of the time in Geneva, Arvid and the author of the PI by FFT paper, Eirik Mo, have always answered my questions online and sent new material.

The following courses at NTNU and UC Berkeley have proved very helpful as a preparation for the thesis:

- TMA 4215 Numerical Mathematics - NTNU - Professor Brynjulf Owren
- TMA 4165 Differential Equations and Dynamical Systems - NTNU - Professor Harald Hanche-Olsen
- TMA 4505 Industrial Mathematics Fordypningsemne - NTNU - Associate Professor Jacob Laading
- STAT 251B Stochastic Analysis with Applications to Mathematical Finance - UC Berkeley - Professor Steven Evans
- MATH 228B Numerical Solutions of Partial Differential Equations - UC Berkeley - Professor James Sethian
- MATH 221 Advanced Matrix Computations - UC Berkeley - Professor Ming Gu

Erik Nesvold
Geneva, June 2010

Abstract

The PI by FFT algorithm is implemented and compared to other numerical solvers for two 2D models in accelerator physics. The solution by finite differences requires a transformation to the Fokker-Planck equation that corresponds to the stochastic differential equation, which is shown in section 3. It is shown how multiplicative noise can be transformed to additive noise by the Itô-Doeblin identity and how this permits an implementation of PI by FFT. It is shown that finite differences is the preferred method when the Fokker-Planck equation is available and that PI by FFT performs better than the other SDE solvers for models with noise in multiple dimensions. Models with noise generated from more general classes of distributions are studied in chapter 4 in the context of option pricing. Finally, limitations of the algorithms and models are discussed throughout the thesis and in the conclusion. The source code in the appendix that is in written in C++ requires the ROOT framework from CERN (root.cern.ch) and the fftw (Fastest Fourier Transform in the West) library from MIT installed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Stochastic Differential Equations

The field of stochastic differential equations (SDEs) is a relatively new one in the world of mathematics. However, the range of applications is already huge and expanding. In the real world there hardly exist any differential equations with deterministic solutions - there is almost always a potential source of noise present in a physical model. Introducing such a source of noise to a familiar differential equation adds a new dimension both theoretically and practically, since elementary calculus is no longer valid and the solution must be considered in terms of a probability density instead of an exact solution. Theoretically, SDEs have close relations with a range of other mathematical fields, i.e. probability theory, measure theory, ordinary differential equations, numerics and even partial differential equations. The applications are numerous, and include interactions in particle physics, predator-prey models, interest rate and stock price modeling, stochastic control problems etc. The most well-known result that has been derived is probably the Black-Scholes-Merton formula [1], which gives the fair price of a European stock option, subject to a number of simplifying assumptions. Myron Scholes and Robert C. Merton received the The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel in 1997 for this work in financial mathematics.

Exactly who pioneered the founding theory of SDEs is contested. However, the name of Brownian motion stems from the Scottish botanist Robert Brown, who in 1828 supposedly studied the motion of pollen particles in water [1]. The random motion of the pollen grain was later explained by an infinitesimal number of collisions with the surrounding molecules in the liquid. Famously, Albert Einstein and Marian Smoluchowski independently studied the mathematical properties of Brownian motion and brought it to the attention of the physics community early in the 20th century. Also, Louis Bachelier published a doctoral thesis in 1900 named 'La Théorie de la Spéculation', which discussed Brownian motion in order to price stock options. The theoretical framework was created by Kiyoshi Itô and Ruslan Stratonovich, by extending methods in calculus to Brownian motion.

In physics, the SDE is usually written as a system of first-order Langevin equations

$$\dot{\mathbf{X}}(t) = \mathbf{A}(\mathbf{X}(t),t) + \mathbf{B}(\mathbf{X}(t),t)\boldsymbol{\xi}(t) \tag{1.1}$$

$$\mathbf{X}(\mathbf{0}) = \mathbf{x}_0 \tag{1.2}$$

Heuristically, one may write the white noise as the time derivative of the noise process (most often a Wiener process/Brownian motion): $\boldsymbol{\xi}(t) = \frac{\mathrm{d}\mathbf{W}(t)}{\mathrm{d}t}$. However, $W(t)$ is in fact not differentiable for any time t, and the derivative does not really exist. In probability theory and finance, the differential form of the equation is usually used:

$$\mathrm{d}\mathbf{X}(t) = \mathbf{A}(\mathbf{X}(t),t)\,\mathrm{d}t + \mathbf{B}(\mathbf{X}(t),t)\,\mathrm{d}\mathbf{W}(t) \tag{1.3}$$

$$\mathbf{X}(\mathbf{0}) = \mathbf{x}_0 \tag{1.4}$$

In other words, $\mathbf{X}(t)$ solves the SDE if

$$\mathbf{X}(t) = \mathbf{x}_0 + \int_0^t \mathbf{A}(\mathbf{X}(s),s)\,\mathrm{d}s + \int_0^t \mathbf{B}(\mathbf{X}(s),s)\,\mathrm{d}\mathbf{W}(s) \quad \forall t > 0 \tag{1.5}$$

1.2 Theory

1.2.1 Probability Theory

Sigma Algebras, Sets and Probability Measures

Although the ideas may seem simple enough, 'white noise', solutions of 1.19 and the stochastic integral have a very precise and rigorous theoretical foundation in measure theory and probability theory. First of all, a σ-algebra is a concept from general probability theory [1]:

Definition 1. If Ω is a given set, then a σ-algebra $\mathcal{F}$ on Ω is a family $\mathcal{F}$ of subsets on Ω with the following properties:

1. $\emptyset \in \mathcal{F}$
2. $F \in \mathcal{F} \rightarrow F^C \in \mathcal{F}$, where F^C is the complement of F in Ω
3. $A_1, A_2, \ldots \in \mathcal{F} \rightarrow A := \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$

The pair $(\Omega, \mathcal{F})$ is called a measurable space. A probability measure P on a measurable space $(\Omega, \mathcal{F})$ is a function $P : (\Omega, \mathcal{F}) \longrightarrow [0,1]$ such that

1. $P(\emptyset) = 0$, $P(\Omega) = 1$
2. If $A_1, A_2, \ldots \in \mathcal{F}$ and $\{A\}_{i=1}^{\infty}$ is disjoint then

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i).$$

The triple $(\Omega, \mathcal{F}, P)$ is called a probability space. It is called a complete probability space if $\mathcal{F}$ contains all subsets G of Ω with P-outer measure zero, i.e. with

$$P^*(G) := \inf\{P(F); F \in \mathcal{F}, G \subset F\} \quad \square$$

The subsets F (or in a probability context; events) of Ω which belong to $\mathcal{F}$ are called $\mathcal{F}$-measurable sets. $P(F)$ is then the probability of the event F, and if it is true except an event of probability zero, it is said to hold almost surely (a.s.). An event F can consist of several sample points $\omega \in \Omega$. Given a family $\mathcal{U}$ of subsets of Ω there is a smallest σ-algebra $\mathcal{H}_{\mathcal{U}}$ that contains $\mathcal{U}$:

$$\mathcal{H}_{\mathcal{U}} = \bigcap \{\mathcal{H}; \mathcal{H}\, \sigma - \mathrm{algebra\,of}\,\Omega,\, \mathcal{U} \subset \mathcal{H}\}$$

$\mathcal{H}_{\mathcal{U}}$ is called the σ-algebra generated by $\mathcal{U}$. If $\mathcal{U}$ is the collection of all open subsets of a topological space Ω, e.g. $\mathbb{R}^n$, then $\mathcal{B} = \mathcal{H}_{\mathcal{U}}$ is called the Borel σ-algebra on Ω and the elements $B \in \mathcal{B}$ are Borel sets. Now, if f is a nonnegative, integrable function, such that $\int_{\mathbb{R}^n} f \, \mathrm{d}x = 1$, one can define

$$P(B) := \int_B f(x) \, \mathrm{d}x$$

for each $B \in \mathcal{B}$. Then $(\mathbb{R}^n, \mathcal{B}, P)$ is a probability space, and f is the density of the probability measure P. Also, given a fixed point $z \in \mathbb{R}^n$: if we define for sets $B \in \mathcal{B}$

$$P(B) := 1 | z \in B \quad P(B) := 0 | z \notin B$$

Then, $(\mathbb{R}^n, \mathcal{B}, P)$ is again a probability space, and P is the Dirac mass concentrated at the point $z \longrightarrow P = \delta_z$. In other words, an appropriate $(\Omega,\, \mathcal{F},\, P)$ must always be identified when attempting to solve a problem. It is an essential mathematical construct, but is however not directly observable.

Sometimes it is necessary to change the probability measure P to another measure $\tilde{P}$. This is often the case in financial mathematics, where one has to use the so-called risk-neutral measure in models of financial markets. This is due to the assumption that the market is complete, in other words that all participants have access to the same information and that no single asset has a tendency to appreciate in value faster than others. This is obviously an idealization, but may anyway give a useful model. Write $Z(\omega)$ as the ratio of the two probability measures:

$$Z(\omega) = \frac{\tilde{P}(\omega)}{P(\omega)}.$$

This is not really sensible as long as Ω is infinite and $P(\omega)$ may be zero, and the equation $Z(\omega)P(\omega) = \tilde{P}(\omega)$ is not much better, since for $\tilde{P}(\omega) = P(\omega) = 0$, $Z(\omega)$ can be anything. The following theorem gives the correct definition [2]:

Theorem 1. *Let $(\Omega,\ \mathcal{F},\ P)$ be a probability space and let Z be an almost surely nonnegative random variable with $\mathbb{E}Z = 1$. For $A \in \mathcal{F}$, define*

$$\tilde{P}(A) = \int_A Z(\omega)\,\mathrm{d}P(\omega) \tag{1.6}$$

Then $\tilde{P}$ is a probability measure. Furthermore, if X is a nonnegative random variable, then $\tilde{\mathbb{E}} = \mathbb{E}[XZ]$. If Z is almost surely strictly positive, we also have $\mathbb{E}Y = \tilde{\mathbb{E}}[\frac{Y}{Z}]$ for every nonnegative random variable Y. $\square$

Under the assumptions in theorem 1, P and $\tilde{P}$ agree on what is possible and what is not. However, they may not give the same probabilities for different outcomes:

Definition 2. Let Ω be a nonempty set and $\mathcal{F}$ a σ-algebra of subsets of Ω. Two probability measures P and $\tilde{P}$ on $(\Omega,\ \mathcal{F})$ are said to be equivalent if they agree which sets in $\mathcal{F}$ have probability zero. $\square$

Stochastic Processes

When dealing with SDEs, one has to relate to random variables depending upon time. A stochastic process is defined as [1]:

Definition 3. A stochastic process is a parameterized collection of random variables $\{X_t\}_{t\in T}$ defined on a probability space $(\Omega,\ \mathcal{F},\ P)$ and assuming variables in $\mathbb{R}^n$ $\square$

Definition 4. Let $W(\,\cdot\,)$ be a 1-dimensional stochastic process defined on some probability space $(\Omega, \mathcal{F}, P)$.

1. The σ-algebra $\mathcal{W}(t) := \mathcal{U}(W(s)|0 \le s \le t)$ is called the history of the stochastic process up to and including time t.
2. The σ-algebra $\mathcal{W}(t)^+ := \mathcal{U}(W(s) - W(t)|s \ge t)$ is the future of the stochastic process beyond time t. $\square$

This leads to the theoretically important concept of a filtration:

Definition 5. A family $\mathcal{F}(\,\cdot\,)$ of σ-algebras $\subseteq \mathcal{U}$ is called nonanticipating with respect to $\mathcal{W}$ if

1. $\mathcal{F}(t) \supseteq \mathcal{F}(s)$ for all $t \ge s \ge 0$
2. $\mathcal{F}(t) \supseteq \mathcal{W}(t)$ for all $t \ge 0$
3. $\mathcal{F}$ is independent of $\mathcal{W}^+(t)$ for all $t \ge 0$

$\mathcal{F}$ is also called a filtration. $\square$

As will be seen later, making assumptions about the future of a stochastic process $X(t)$ leads to a diverging numerical path from the nonanticipating assumption. The essence is that the process at time t depends only on the information that is available in the σ-algebra $\mathcal{F}(t)$.

Definition 6. A real-valued stochastic process $X(\,\cdot\,)$ is called nonanticipating with respect to $\mathcal{F}$ if for each time $t \geq 0$, $G(t)$ is $\mathcal{F}(t)$-measurable. $\square$

The finite-dimensional distributions of the process $\{X_t\}_{t\in T}$ are the measures defined by

$$\mu_{t_1,\dots,t_k}(F_1 \times F_2 \times \dots \times F_k) = P(X_{t_1} \in F_1, \dots, X_{t_k} \in F_k)$$

where $F_1, \dots, F_k$ denote Borel sets in $\mathbb{R}^n$. Given a family $\{\nu_{t_1,\dots,t_k}; k \in \mathbb{N}, t_i \in T\}$ of probability measures, it is essential to be able to deduce a stochastic process $Y = \{Y_t\}_{t\in T}$ having this family as its finite-dimensional distributions. The following theorem by Kolmogorov [1] states that this can be done subject to two natural conditions on $\{\nu_{t_1,\dots,t_k}\}$:

Theorem 2 (Kolmogorov's extension theorem)**.** *For all $t_1, \dots, t_k \in T$, $k \in \mathbb{N}$ let $\nu_{t_1,\dots,t_k}$ be probability measures on $\mathbb{R}^{nk}$ s.t.*

$$\nu_{t_{\sigma(1)},\dots,t_{\sigma(k)}}(F_1 \times \dots \times F_k) = \nu_{t_1,\dots,t_k}(F_{\sigma^{-1}(1)} \times \dots \times F_{\sigma^{-1}(k)}) \tag{1.7}$$

for all permutations σ on $\{1, 2, ..., k\}$ and

$$\nu_{t_1,\dots,t_k}(F_1 \times \dots \times F_k) = \nu_{t_1,\dots,t_k,t_{k+1},\dots,t_{k+m}}(F_1 \times \dots \times F_k \times \mathrm{R}^n \times \dots \mathrm{R}^n) \tag{1.8}$$

for all $m \in \mathbb{N}$, where the set on the right-hand side has a total of $k+m$ factors. Then there exists a probability space (Ω, $\mathcal{F}$, P) and a stochastic process $\{X_t\}$ on Ω, $X_t : \Omega \to \mathbb{R}^n$ s.t.

$$\nu_{t_1,\dots,t_k}(F_1 \times \dots \times F_k) = P(X_{t_1} \in F_1, \dots, X_{t_k} \in F_k)$$

for all $t_i \in T$, $k \in \mathbb{N}$ and all Borel sets F_i. $\square$

Brownian motion is one important example of a stochastic process. Obviously, in an SDE, the true noise process is not always multinormally distributed, but very often it will be assumed to be, i.e. it will be a Brownian motion. If $\{B_t(\omega)\}_{t\geq 0}$ is the position of the pollen grain at time t, it is sufficient to specify a family of probability measures that satisfy 1.7 and 1.8. In n dimensions, let

$$p(t, x, y) = (2\pi t)^{-n/2} \cdot \exp^{\frac{-|x-y|^2}{2t}} \qquad x, y \in \mathbb{R}^n, \quad t > 0$$

With $t_0 \leq t_1 \leq \dots \leq t_k$, define the following measure on $\mathbb{R}^{nk}$

$$\begin{aligned} \nu_{t_1,\dots,t_k}(F_1 \times \dots \times F_k) \quad &= \quad \int_{F_1\times\dots\times F_k} p(t_1 - t_0, x, x_1) \cdot p(t_2 - t_1, x, x_2) \cdot & (1.9) \\ &\cdots \quad p(t_k - t_{k-1}, x_{k-1}, x)\, \mathrm{d}x_1\, \mathrm{d}x_2 \cdots \mathrm{d}x_k & (1.10) \end{aligned}$$

Here, $\mathrm{d}y = \mathrm{d}y_1 \dots \mathrm{d}y_k$ is the Lebesgue measure and $\lim_{t\to 0} \quad p(t, x, y)\, \mathrm{d}y = \delta_x(y)$. Now 1.8 clearly holds, since $\int_{\mathbb{R}^n} p(t, x, y)\, \mathrm{d}y = 1$ for all $t \geq 0$. The definition in 1.9 can be extended to any finite sequence of t_i's as in 1.7. Therefore, the Kolmogorov extension theorem 2 guarantees the existence of a stochastic process with the finite-dimensional probability distribution 1.9, called Brownian motion starting at x. Brownian motion has some important properties:

1. As mentioned above, B_t is a Gaussian process. In other words, the random variable $Z = (B_{t_1}, \ldots, B_{t_k}) \in \mathbb{R}^{nk}$ is multinormally distributed. Hence there exist a vector $M \in \mathbb{R}^{nk}$ and a symmetric, positive-definite matrix $C \in \mathbb{R}^{nk \times nk}$ such that $\mathbb{E}_x[Z] = M$ and $c_{ij} = \mathbb{E}_x[(Z_i - M_i)(Z_j - M_j)]$ when $\mathbb{E}_x$ denotes the expectation with respect to P_x. Moreover,

$$M = (x, x, \ldots, x) \in \mathbb{R}^{nk} \tag{1.11}$$

and, with I_n the identity matrix:

$$C = \begin{pmatrix} t_1 I_n & t_1 I_n & \ldots & t_1 I_n \\ t_2 I_n & t_2 I_n & \ldots & t_2 I_n \\ \vdots & \vdots & & \vdots \\ t_1 I_n & t_2 I_n & \ldots & t_k I_n \end{pmatrix} \in \mathbb{R}^{nk \times nk}$$

2. B_t has independent increments, i.e. B_{t_1}, $B_{t_2} - B_{t_1}$, ..., $B_{t_k} - B_{t_k}$ are independent for all t_i. This can be seen directly from C.

3. By Kolmogorov's continuity theorem 3 [1], Brownian motion can be shown to have a Hölder continuous version in time t.

Theorem 3 (Kolmogorov's continuity theorem)**.** *Suppose that the process $X = \{X_t\}_{t \geq 0}$ satisfies the following condition: For all $T \geq 0$ there exist positive constants α, β, D such that*

$$\mathbb{E}[|X_t - X_s|^\alpha] \leq D \cdot |t - s|^{1+\beta} \quad 0 \leq s, t \leq T$$

Then for each $0 < \gamma < \frac{\beta}{\alpha}$, $T > 0$ and almost every ω, there exists a constant K such that

$$|X(t, \omega) - X(s, \omega)| \leq K|t - s|^\gamma \quad \square$$

The latter is the definition of Hölder continuity [2], and in the special case of $\gamma = 1$, it is Lipschitz continuous. Oksendal does not show explicitly that this applies to Brownian motion, but the proof is not so difficult:

$$\begin{aligned} \mathbb{E}[|X_t - X_s|^{2m}] &= (2\pi(t-s))^{-n/2} \int_{\mathbb{R}^n} |x|^{2m} e^{-\frac{-|x|^2}{2(t-s)}} \, \mathrm{d}x \\ &= (2\pi)^{-n/2} (t-s)^m \int_{\mathbb{R}^n} |y|^{2m} e^{-\frac{|y|^2}{2}} \, \mathrm{d}y \quad (y = \frac{x}{\sqrt{t-s}}) \\ &= C|t-s|^m \end{aligned}$$

Thus this holds for $2m = \alpha$ and $1+\beta = m$, and the Brownian motion is Hölder continuous almost surely for exponents $0 < \gamma < \frac{1}{2} - \frac{1}{2m}$. For example, the one-dimensional Brownian motion has

$$\mathbb{E}[|X_t - X_s|^{2m}] = \frac{\sqrt{2\pi}}{\sqrt{2\pi}} \prod_{i=1}^{m} (2i-1)|t-s|^m = \prod_{i=1}^{m} (2i-1)|t-s|^m$$

As mentioned earlier, even though the Brownian motion can be shown to be continuous for certain exponents, it is not differentiable. For a proof of the theorem, see [2].

Theorem 4. *(i) For each $1/2 < \gamma < 1$ and almost every ω, $t \to \mathbf{W}(t,\omega)$ is nowhere Hölder continuous with exponent γ. (ii) In particular, for almost every ω, the sample path $t \to \mathbf{W}(t,\omega)$ is nowhere differentiable and is of infinite variation on each subinterval.* $\square$

In other words, the white noise process $\xi(t)$ does not really exist. However, one still writes heuristically

$$\mathbb{E}[\xi(t)\xi(s)] = \delta_0(s-t) \tag{1.12}$$

Suppose $h > 0$ and fix $t > 0$, then let

$$\begin{aligned}
\phi_h(s) &= \mathbb{E}[\frac{W(t+h)-W(t)}{h}\frac{W(s+h)-W(s)}{h}] \\
&= \frac{1}{h^2}[\mathbb{E}[W(t+h)W(s+h)] - \mathbb{E}[W(t+h)W(s)] - \mathbb{E}[W(t)W(s+h)] + \mathbb{E}[W(s)W(t)]] \\
&= \frac{1}{h^2}[((t+h)\wedge(s+h)) - ((t+h)\wedge s) - (t\wedge(s+h)) + (t\wedge s)]
\end{aligned}$$

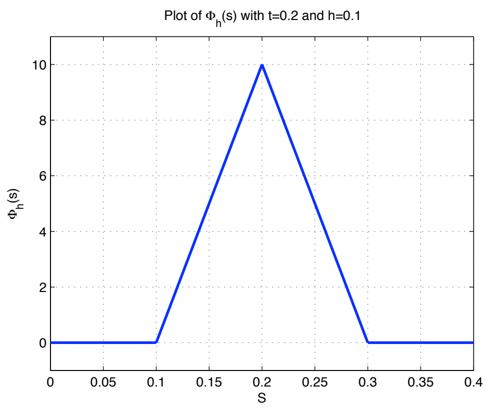


Figure 1.1: $\phi_h(s)$ with h=0.1 and t =0.2.

where the first equality comes from the independence and covariance of Brownian motion. Thus, $\phi(t-h) = 0$, $\phi(t) = \frac{h}{h^2} = \frac{1}{h}$ and $\phi(t+h) = 0$ and $\int \phi_h(s)\,\mathrm{d}s = 1$, so presumably $\lim_{h\to 0}\phi_h(s) = \delta_0(s-t)$. Now, in time series analysis, for a stochastic process $X(\cdot)$ $r(s,t) = \mathbb{E}[X(t)X(s)]$ is called the autocorrelation function of $X(\cdot)$. If $r(t,s) = c(t-s)$ for some function $c : \mathbb{R} \to \mathbb{R}$ and if $\mathbb{E}[X(s)] = \mathbb{E}[X(t)]$ for all $s,t \geq 0$, then $X(\cdot)$ is called stationary in the wide sense. From the reasoning above, $\xi(\cdot)$ is

Gaussian and stationary in the wide sense, with $c(\,\cdot\,) = \delta_0(\,\cdot\,)$. For a process $X(\,\cdot\,)$, the spectral density is defined as

$$f(\lambda) := \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\lambda t} c(t)\,\mathrm{d}t \qquad (\lambda \in \mathbb{R}).$$

For white noise,

$$f(\lambda) := \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\lambda t}\delta_0\,\mathrm{d}t = \frac{1}{2\pi} \quad \forall \qquad \lambda.$$

The spectral density of $\xi(\,\cdot\,)$ is therefore flat - in some sense, all frequencies contribute equally, just as all color frequencies contribute equally to white light.

1.2.2 Stochastic Calculus - Itô's Integral

As mentioned above, the problem that made stochastic calculus a necessity was the second integral in integral equation 1.5 - the Itô integral. $\mathbf{W}(t)$ is of infinite variation on every subinterval and an integral with respect to it cannot be understood as a regular integral. The discussion on Brownian motion in the previous section has some interesting consequences when $\Delta t \to 0$:

$$\mathbb{E}[(W(t_{i+1}) - W(t_i))^2] = \mathrm{Var}[W(t_{i+1}) - W(t_i)] = t_{i+1} - t_i \tag{1.13}$$

Now, if we define the time step as $\Delta(t_i) = t_{i+1} - t_i = \frac{\mathrm{T}}{n}$ and sum the squares of the increments ΔW_i we obtain the following:

$$\sum_{i=0}^{n-1}(W(t_{i+1}) - W(t_i))^2 = \sum_{i=0}^{n-1}\Delta(t_i)\frac{(W(t_{i+1}) - W(t_i))^2}{t_{i+1} - t_i}$$

and if we define the square of the standard normal variable Y_i as

$$\mathrm{Y}_i^2 = \frac{(W(t_{i+1}) - W(t_i))^2}{t_{i+1} - t_i}$$

we know from eq. 1.13 that as $n \to \infty$, by the Law of Large Numbers

$$\mathbb{E}[\mathrm{Y}_i^2] = 1 \qquad \to \qquad \mathbb{E}\sum_{i=0}^{n-1}(W(t_{i+1}) - W(t_i))^2 = T$$

This is why one says heuristically that Brownian motion accumulates quadratic variation at rate one per unit time [2].

Now, going back to definition 5 - a slightly stronger statement about the process $X(t)$ is that it is progressively measurable. In addition to being nonanticipating, it is then jointly measurable in the variables t and ω together. The following definitions are needed later:

Definition 7. Denote by $\mathbb{L}^2(0,T)$ the space of all real-valued, progressively measurable stochastic processes $G(\,\cdot\,)$ such that

$$\mathbb{E}(\int_0^T G^2\,\mathrm{d}t) < \infty \quad \square$$

Definition 8. Similarly, denote by $\mathbb{L}^1(0,T)$ the space of all real-valued, progressively measurable processes $H(\,\cdot\,)$ such that

$$\mathbb{E}(\int_0^T |H|\,\mathrm{d}t) < \infty \quad \square$$

If the interval $[0,T]$ is partitioned as before, in $0 = t_0 \leq \ldots \leq t_n = T$, one can assume that the process $G(X_t, t) \in \mathbb{L}^2(0,T)$ is constant in t on each subinterval $[t_j, t_{j+1})$, as shown in figure 1.2. Each $G(X_{t_j}, t_j)$ is then nonanticipating by definition 5 and therefore a $\mathcal{F}(t_j)$-measurable random variable. Such a process is called simple [2] or elementary [1].

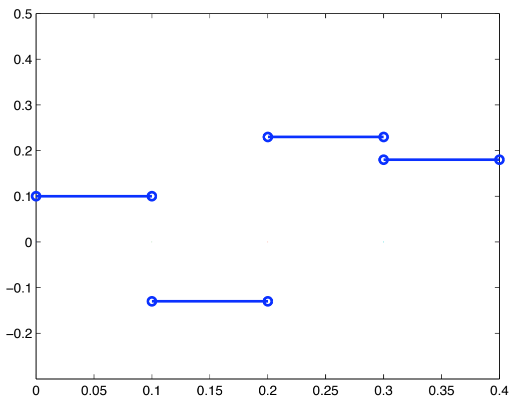


Figure 1.2: An example path of a simple process.

$$I(t) = \int_0^T G(X_u, u)\,\mathrm{d}W(u) := \sum_{j=0}^{n} G(X_{t_j}, t_j)(W_{t_{j+1}} - W_{t_j}) \tag{1.14}$$

$I(t)$ is the Itô integral of the simple process $G()$. It can be shown that 1.14 is a martingale [2]. A martingale is a construct in probability theory that has no tendency to rise or fall, or more precisely:

Definition 9 (Martingale)**.** A discrete-time martingale is a discrete-time stochastic process that satisfies for any s, t, i.e. with respect to the filtration $\mathcal{F}_s$

$$\begin{aligned} \mathbb{E}(|I(t)|) &\leq \infty \\ \mathbb{E}(I(t)|\mathcal{F}(s)) &= \mathbb{E}(I(t)|I(s), I(s-\Delta t), \ldots) = I(s) \quad \forall t \geq s \quad \square \end{aligned}$$

Furthermore,

Theorem 5 (Itô isometry). *The Itô integral defined in 1.14 satisfies*

$$\mathbb{E}(I^2(t)) = \mathbb{E}\int_0^T G^2(X_u, u)\,\mathrm{d}u \quad \square$$

Sketch of Proof:

$$I^2(t) = \sum_{i=0}^{n}\sum_{j=0}^{n} G(X_{t_i}, t_i)G(X_{t_j}, t_j)(W_{t_{i+1}} - W_{t_i})(W_{t_{j+1}} - W_{t_j})$$

The Brownian increment $\Delta W_j = W_{j+1} - W_j$ is independent of the filtration $\mathcal{F}_j$ and $\mathbb{E}(\Delta W_j) = 0$. For the cross-terms $i < j$

$$\mathbb{E}(G(X_{t_i}, t_i)G(X_{t_j}, t_j)\Delta W_i \Delta W_j) = \mathbb{E}(G(X_{t_i}, t_i)G(X_{t_j}, t_j)\Delta W_i)\mathbb{E}(\Delta W_j) = 0$$

Therefore, only the terms $i = j$ are left, and one obtains

$$\begin{aligned}\mathbb{E}I^2(t) &= \sum_{i=0}^{n-1}\mathbb{E}(G^2(X_{t_i}, t_i)\Delta W_i^2) = \sum_{i=0}^{n-1}\mathbb{E}(G^2(X_{t_i}, t_i))\mathbb{E}(\Delta W_i^2)\\ &= \sum_{i=0}^{n-1}\mathbb{E}(G^2(X_{t_i}, t_i))\Delta t_i\end{aligned}$$

Since G_i is constant on the subinterval $[t_i, t_{i+1})$, $\int_{t_i}^{t_{i+1}} G^2(X_u, u)\,\mathrm{d}u = G^2(X_{t_i}, t_i)\Delta t_i$, and

$$\begin{aligned}\mathbb{E}I^2(t) &= \sum_{i=0}^{n-1}\mathbb{E}\int_{t_i}^{t_{i+1}} G^2(X_u, u)\,\mathrm{d}u\\ &= \mathbb{E}\sum_{i=0}^{n-1}\int_{t_i}^{t_{i+1}} G^2(X_u, u)\,\mathrm{d}u = \mathbb{E}\int_0^T G^2(X_u, u)\,\mathrm{d}u \quad \square\end{aligned}$$

In ordinary calculus, for a differentiable function $g(t)$ with $g(0) = 0$, one has

$$\int_0^T g(t)\,\mathrm{d}g(t) = \int_0^T g(t)g'(t)\,\mathrm{d}t = \frac{1}{2}g^2(T)$$

In contrast, denote the Itô integral with $W(t)$ as the integrand by

$$I(t) = \int_0^T \mathbf{W}(t)\,\mathrm{d}\mathbf{W}(t) \quad \mathrm{d}I(t) = \mathbf{W}(t)\,\mathrm{d}\mathbf{W}(t) \tag{1.15}$$

in integral and differential form respectively. They mean almost the same thing, but the second may seem more intuitive. Strictly speaking, however, d and dt mean nothing on their own - the differential form is simply an abbreviation of the integral form of the equation. Another difference is that integrating the differential version, one has to specify an initial condition $I(0)$. It is important to emphasize that the Itô integral is itself a stochastic process with respect to the time parameter, and one can calculate its moments. Now, for the integral 1.15, a reasonable procedure would be to construct the Riemann sum approximation and then find the limit of this. For a fixed $0 \leq \lambda \leq 1$, set $\tau_k := (1-\lambda)t_i + \lambda t_{i+1}$. With a partition $0 = t_0 < ... < t_n = T$, define the Riemann sum approximation to be

$$R = \sum_{i=0}^{n-1} W(\tau_k)(W(t_{i+1}) - W_{t_i}) \tag{1.16}$$

Surprisingly, different λ yield different results: choosing $\lambda = 0$, in other words evaluating the sum at the left-hand side of each interval, gives the rearrangement, with $W_0 = 0$

$$\begin{aligned} R_{\lambda=0} &= \int_0^T W(t)\,\mathrm{d}W(t) = \lim_{n\to\infty} \sum_{i=0}^{n-1} W_i(W_{i+1} - W_i) \\ \sum_{i=0}^{n-1} W_i(W_{i+1} - W_i) &= -W_0^2 + W_0W_1 - W_1^2 + \ldots - W_{n-1}^2 + W_{n-1}W_n \\ = -\frac{1}{2}(2W_0^2 &- 2W_0W_1 + 2W_1^2 - \ldots - 2W_{n-1}W_n + W_n^2) + \frac{1}{2}W_n^2 \\ = \frac{1}{2}W_n^2 &- \frac{1}{2}\sum_{i=0}^{n-1}(W_{i+1} - W_i)^2 \end{aligned}$$

Hence one obtains, for $\lambda = 0$

$$\int_0^T W(t)\,\mathrm{d}W(t) = \frac{1}{2}W(T)^2 - \frac{1}{2}T \tag{1.17}$$

However, choosing $\lambda = 1/2$, the midpoint of the interval is chosen, such that

$$\begin{aligned} R_{\lambda=1/2} &= \lim_{n\to\infty} \sum_{i=0}^{n-1} W\left(\frac{t_i + t_{i+1}}{2}\right)(W(t_{i+1}) - W(t_i)) \\ &= \lim_{n\to\infty} \sum_{i=0}^{n-1} \frac{W_i + W_{i+1}}{2}(W_{i+1} - W_i) \\ &= \frac{1}{2}W(T)^2 \end{aligned}$$

These two diverging paths turn out to be the foundation of two versions of stochastic calculus: Itô stochastic calculus and Stratonovich stochastic calculus. The Itô and Stratonovich integrals are denoted respectively by

$$\int_0^T W(t)\,\mathrm{d}W(t) = \frac{1}{2}W(T)^2 - \frac{1}{2}T, \quad \int_0^T W(t) \circ \mathrm{d}W(t) = \frac{1}{2}W(T)^2$$

In differential form, this means that

$$W_t \, \mathrm{d}W_t = \frac{1}{2}\, \mathrm{d}W_t^2 - \frac{1}{2}\, \mathrm{d}t \quad W_t \, \mathrm{d}W_t = \frac{1}{2}\, \mathrm{d}W_t^2 \tag{1.18}$$

According to definition 5 and 6, the Itô integral is nonanticipating, while the Stratonovich integral is not. In some sense, the Stratonovich definition makes assumptions on the stochastic process half a time step into the future, while the Itô definition depends only on the history and the present, in other words it is $\mathcal{F}$-measurable in definition 5. The Itô definition is therefore most common in for example financial mathematics, since no assumptions can be made about the future. However, a slightly stronger requirement is needed: the stochastic process needs to be *progressively measurable*. This will not be defined here, but some neat identities can be deduced from these two assumptions on the stochastic integral. The following identities, in addition to the ones found earlier, can be shown [2] by rather easy algebraic manipulation

Theorem 6 (Properties of the Itô integral)**.** *For all constants $a, b \in \mathbb{R}$ and for all $G, H \in \mathbb{L}^2(0,T)$, with W a Brownian motion, we have*

1. $\int_0^T aG + bH \, \mathrm{d}W = a\int_0^T G \, \mathrm{d}W + b\int_0^T H \, \mathrm{d}W$
2. $\mathbb{E}(\int_0^T G \, \mathrm{d}W) = 0$
3. $\mathbb{E}((\int_0^T G \, \mathrm{d}W)(\int_0^T H \, \mathrm{d}W)) = \mathbb{E}(\int_0^T GH \, \mathrm{d}W)$ $\square$

Using ordinary calculus, the Taylor expansion of a function $f(X_t, t)$ would be

$$\begin{aligned} f(X_t + \Delta X_t, t + \Delta t) - f(X_t, t) &= \frac{\partial f(x,t)}{\partial x}\Delta x + \frac{\partial f(x,t)}{\partial t}\Delta t \\ &+ \frac{1}{2}\frac{\partial^2 f(x,t)}{\partial t^2}(\Delta t)^2 + \frac{\partial^2 f(x,t)}{\partial x \partial t}\Delta x \Delta t \\ &+ \frac{1}{2}\frac{\partial^2 f(x,t)}{\partial x^2}(\Delta x)^2 + \ldots \end{aligned}$$

With the differential $\mathrm{d}X = A\, \mathrm{d}t + B\, \mathrm{d}W$, Itô calculus in one dimension gives;

$$\mathrm{d}f(X_t, t) = \frac{\partial f(x,t)}{\partial t}\, \mathrm{d}t + \frac{\partial f(x,t)}{\partial x}\, \mathrm{d}X_t + \frac{1}{2}\frac{\partial^2 f(x,t)}{\partial x^2}\, \mathrm{d}X_t^2 + R$$

where

$$|R| \le C(\Delta t^2 + \Delta t|\Delta W|)$$

Using the results from 1.18 and the definition of the Itô integral 1.14,

$$\begin{aligned} f(X_T, T) &= f(X_0, 0) + \int_0^T \frac{\partial f}{\partial t}\, \mathrm{d}t + \int_0^T \frac{\partial f}{\partial x} A\, \mathrm{d}t + \int_0^T \frac{\partial f}{\partial x} B\, \mathrm{d}W_u \\ &+ \frac{1}{2}\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial^2 f_{t_i}}{\partial x^2} B_{t_i}^2 \Delta W_{t_i}^2 + \sum_{i=0}^{n-1} R_i \end{aligned}$$

Since

$$\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial^2 f_{t_i}}{\partial x^2}B_{t_i}^2\Delta W_{t_i}^2=\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial^2 f_{t_i}}{\partial x^2}B_{t_i}^2(\Delta W_i^2-\Delta t_i+\Delta t_i)=\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial^2 f_{t_i}}{\partial x^2}B_{t_i}^2\Delta t_i$$

, taking the limit $\Delta t_i \to 0$ and then the limit $T \to 0$

$$\mathrm{d}f(X_t,t)=\frac{\partial f}{\partial t}\,\mathrm{d}t+\frac{\partial f}{\partial x}A\,\mathrm{d}t+\frac{\partial f}{\partial x}B\,\mathrm{d}W+\frac{1}{2}\frac{\partial^2 f}{\partial x^2}B^2\,\mathrm{d}t$$

Using stochastic calculus, the integral equation 1.5 posited in the introduction is now possible to solve. If $\mathbf{X}\in\mathbb{R}^n$ is a stochastic process satisfying this equation for some $\mathbf{A}\in\mathbb{L}^1(0,T)$ and $\mathbf{B}\in\mathbb{L}^2(0,T)$, the Itô-Doeblin [1] formula [2] generalizes this result to n dimensions:

Theorem 7 (Itô-Doeblin formula in n dimensions)**.** *Let* $\mathbf{X}$ *be defined as above and* $f:\mathbb{R}^n\times[0,T]\to\mathbb{R}^n$ *be continuous and with continuous partial derivatives* $\frac{\partial f}{\partial t}$, $\frac{\partial f}{\partial x_i}$ *and* $\frac{\partial^2 f}{\partial x_i\partial x_j}$. *Also,* $\mathbf{A}:\mathbb{R}^n\times[0,T]\to\mathbb{R}^n$ *and* $\mathbf{B}:\mathbb{R}^{n\times m}\times[0,T]\to\mathbb{R}^{n\times m}$ *Then, for every* $T\geq 0$,

$$\begin{aligned}\mathrm{d}f(\mathbf{X}(t),t) &= \frac{\partial f}{\partial t}\,\mathrm{d}t+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}\,\mathrm{d}X_i+\frac{1}{2}\sum_{i=1}^{n}\sum_{i=j}^{n}\frac{\partial^2 f}{\partial x_i\partial x_j}(\sum_{l=1}^{m}B_{il}B_{jl}\,\mathrm{d}t)\\ &= \frac{\partial f}{\partial t}\,\mathrm{d}t+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}A_i\,\mathrm{d}t+\sum_{i=1}^{n}(\frac{\partial f}{\partial x_i}\sum_{l=1}^{m}B_{il}\,\mathrm{d}W_l))\\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{i=j}^{n}\frac{\partial^2 f}{\partial x_i\partial x_j}(\sum_{l=1}^{m}B_{il}B_{jl}\,\mathrm{d}t)\quad\square\end{aligned}$$

By contrast, the Stratonovic interpretation gives

$$\mathrm{d}f(\mathbf{X}(t),t)=\frac{\partial f}{\partial t}\,\mathrm{d}t+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}A_i\,\mathrm{d}t+\sum_{i=1}^{n}\frac{\partial f}{\partial x_i}\sum_{l=1}^{m}B_{il}\circ\,\mathrm{d}W_l$$

so the normal chain rule is seen to hold for the Stratonovich stochastic differential equation. As a consequence, $\mathbf{X}(t)$ solves the Itô SDE 1.19 if and only if it solves the Stratonovich differential equation

$$\begin{aligned}\mathrm{d}\mathbf{X}(t) &= (\mathbf{A}(\mathbf{X}(t),t)-\mathbf{C}(\mathbf{X},t))\,\mathrm{d}t+\mathbf{B}(\mathbf{X}(t),t)\circ\,\mathrm{d}\mathbf{W}(t)\\ \mathbf{X}(\mathbf{0}) &= \mathbf{x}_0\end{aligned}$$

[1] Doeblin was a French soldier at the German front during World War II, whose discoveries were sent in a letter to France. This was not opened until in 2000 - however it was written independently of Itô

where $C_i(\mathbf{X},t) = \frac{1}{2}\sum_{l=1}^{m}\sum_{j=1}^{n}\frac{\partial B^{jl}}{\partial x_j}B^{jk}$. For example; if $m = n = 1$, $A(X,t) = a(x)$ and $B(X,t) = b(x)$,

$$\mathrm{d}X = a(X)\,\mathrm{d}t + b(X)\,\mathrm{d}W$$

if and only if

$$\mathrm{d}X = (a(X) - \frac{1}{2}b'(X)b(X))\,\mathrm{d}t + b(X)\circ\mathrm{d}W$$

Girsanov's Theorem

Theorem 1 showed that it is possible to change the probability measure of a single random variable defined on Ω. However, it is sometimes useful to change the probability measure of a whole process. Instead of defining Z as the ratio of the probability measures, the Radon-Nikodym derivative of $\tilde{P}$ with respect to P is

$$Z = \frac{\mathrm{d}\tilde{P}}{\mathrm{d}P}.$$

and the Radon-Nikodym derivative process is defined as

$$Z(t) = \mathbb{E}[Z|\mathcal{F}(t)], \quad 0 \leq t \leq T.$$

It is a martingale (no tendency to rise or fall), since by iterating expectations

$$\mathbb{E}[Z|\mathcal{F}(s)] = \mathbb{E}[\mathbb{E}[Z|\mathcal{F}(t)]|\mathcal{F}(s)] = Z(s)$$

It can be shown that $Z(t)$ as defined in Girsanov's theorem is a Radon-Nikodym derivative process, and that with $Y(t)$ a $\mathcal{F}(t)$-measurable random variable, $\tilde{\mathbb{E}}Y = \mathbb{E}[YZ(t)]$ and $\tilde{\mathbb{E}}[Y|\mathcal{F}(s)] = \frac{1}{Z(s)}\mathbb{E}[YZ(t)|\mathcal{F}(s)]$ $(0 \leq s \leq t \leq T)$.

Theorem 8 (Girsanov's theorem in one dimension)**.** *Let $W(t)$, $0 \leq t \leq T$, be a Brownian motion on a probability space (Ω, $\mathcal{F}$, P), and let $\mathcal{F}(t)$ be a filtration for this Brownian motion. Let $A(t)$ be an adapted process. Define*

$$\begin{aligned} Z(t) &= \exp[-\int_0^t A(u)\,\mathrm{d}W(u) - \frac{1}{2}\int_0^t A^2(u)\,\mathrm{d}u] \\ \tilde{W}(t) &= W(t) + \int_0^t A(u)\,\mathrm{d}u \end{aligned}$$

and assume that

$$\mathbb{E}\int_0^t A^2(u)Z^2(u)\,\mathrm{d}u < \infty$$

Set $Z = Z(T)$. Then $\mathbb{E}Z = 1$ and under the probability measure $\tilde{P}$ given by equation 1.6, the process $\tilde{W}(t)$ is a Brownian motion. $\square$

The proof of this theorem relies on Levy's theorem [2], which says that a martingale starting at zero at time zero with continuous paths and with quadratic variation equal to t at each time t is a Brownian motion. By using Itô's product rule, it is straightforward to show that $\tilde{W}$ is a martingale under $\tilde{P}$ and that $\tilde{W}Z$ is a martingale under $\tilde{P}$. The probability measures are equivalent, so the set of outcomes that is possible under P is the same as under $\tilde{P}$. Girsanov's theorem often applies when pricing derivatives of financial assets - the drift of the process must be the risk-neutral one if the market is assumed to be perfect. This will be elaborated in chapter 4.

1.2.3 Jump processes

Solutions of SDEs with Gaussian white noise can be shown to be continuous. However, continuity may not be a desired property for the noise process. In many real-life applications, the time series is necessarily discrete and the measured variable may be subject to discontinuities in the form of jumps. This is the case in earthquake modeling, financial modeling, meteorology and other earth and social sciences. The examples will be further developed in the section on Lévy noise, but this requires a theoretical framework. The fundamental jump process, where the jump is always of size one, is the standard Poisson process. This is a standard, well understood probability distribution, where the interarrival times of the jumps are exponentially distributed with intensity λ:

$$\begin{aligned} f(t) &= \lambda e^{-\lambda t} \quad t \geq 0 \\ f(t) &= 0 \quad t < 0 \end{aligned}$$

The Poisson process itself is a counting process $N(t)$ defined by the intensity of arrivals λ [2]:

$$\mathbb{P}(N(t) = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad k = 0, 1, \dots \tag{1.19}$$

As a consequence, $N(t+s) - N(s)$ is independent of the filtration $\mathcal{F}(s)$. A compound Poisson process $C(t)$ is similar to the Poisson process, with the difference that it may have jumps of random size. The first jump is of size Y_1, the second of size Y_2, etc:

$$\begin{aligned} C(t) &= \sum_{i=1}^{N(t)} Y_i \quad t \geq 0 \\ C(t) &= 0 \quad t < 0 \end{aligned}$$

The problem is now to define the stochastic integral

$$\int_o^t \Phi(s) \, \mathrm{d}X(s)$$

where the integrator X can have jumps. Furthermore, the problem will typically be to integrate the sum of a Riemann integral, an Itô integral and a stochastic jump integral, given an initial condition at time 0:

$$X(t) = X(0) + R(t) + I(t) + J(t)$$

The continuous part is defined to be

$$X^c(t) = X(0) + \int_0^t A(s)\,\mathrm{d}s + \int_0^t B(s)\,\mathrm{d}W(s)$$

which has the same quadratic variation as the Itô process in the previous sections:

$$\begin{aligned}[X^c, X^c](t) &= \int_0^t B^2(s)\,\mathrm{d}s \\ \mathrm{d}X^c(t)\,\mathrm{d}X^c(t) &= B^2\,\mathrm{d}t\end{aligned}$$

J(t) is assumed to be càdlàg ('continue à droite, limitée à gauche' - continuous to the right, limited to the left) and a pure jump process, i.e. that $J(t) = \lim_{s\downarrow t} J(s)$ or that a jump at time t implies that $J(t)$ is the value immediately after the jump and $J(t-)$ the value immediately before the jump, thus $\Delta X(t) = J(t) - J(t-)$ Since the jump process is pure, the value does not change between jumps. The number of jumps is also assumed to be finite for a finite time interval $(0, T]$. The stochastic jump integral of $\Phi(t)$ with respect to X is defined to be, in integral and differential notation respectively [2]:

$$\begin{aligned}\int_0^t \Phi(s)\,\mathrm{d}X(s) &= \int_0^t \Phi(s)A(s)\,\mathrm{d}s + \int_0^t \Phi(s)B(s)\,\mathrm{d}W(s) + \sum_{0<s\leq t} \Phi(s)\Delta J(s) \\ \Phi(t)\,\mathrm{d}X(t) &= \Phi(t)A(t)\,\mathrm{d}t + \Phi(t)B(t)\,\mathrm{d}W(t) + \Phi(t)\,\mathrm{d}J(t)\end{aligned}$$

In order to find the Itô-Doeblin formula for the stochastic jump process, one needs to express the quadratic variation of the terms. Let Δt_i be the same for all i, and

$$Q_{\Delta t}(X) = \sum_{j=0}^{n-1} (X(t_{j+1}) - X(t_j)))^2$$

The quadratic variation of X on T is

$$[X, X](T) = \lim_{\Delta t \to 0} Q_{\Delta t}(X)$$

By regular algebraic expansion one gets the following theorem:

Theorem 9 (Quadratic variation of stochastic jump processes)**.** *Let $X(t) = X(0) + R(t) + I(t) + J(t)$ be a jump process such that $X^c(t) = X(0) + \int_0^t A(s)\,\mathrm{d}s + \int_0^t B(s)\,\mathrm{d}W(s)$ and $J(t)$ is a right-continuous pure jump process. Then the quadratic variation of $X(T)$ is:*

$$\begin{aligned}[X, X](T) &= \int_0^T B^2(s)\,\mathrm{d}s + \sum_{0<s\leq T} (\Delta J(s))^2 \\ \mathrm{d}X(t)\,\mathrm{d}X(t) &= \mathrm{d}X^c(t)\,\mathrm{d}X^c(t) + \mathrm{d}J(t)\,\mathrm{d}J(t) \quad \square\end{aligned}$$

For a full proof, see Shreve [2]. It is similar to the one for the Itô process, the key point being that the cross-term between the continuous part and the discontinuous part converges to 0 since

$$\begin{aligned}
&|\sum_{j=0}^{n-1}(X^c(t_{j+1}) - X^c(t_j))(J(t_{j+1}) - J(t_j))| \\
&\leq \max_j |X^c(t_{j+1}) - X^c(t_j)| \sum_{0<s\leq T} |\Delta J(s)|,
\end{aligned}$$

where $\Delta J(s)$ is a finite number independent of Δt, while the first factor converges to 0 with decreasing Δt. The conclusion is that the cross variation between a Brownian motion and a Poisson process is zero. For it to be nonzero, the two processes must both contain a dW term or have simultaneous jumps. More precisely, with respect to the filtration $\mathcal{F}(t)$, a Brownian motion and a Poisson process must be independent.

Theorem 10 (Itô-Doeblin formula for one jump process)**.** *Let $X(t)$ be a jump process and $f(x)$ a function for which $f'(x)$ and $f''(x)$ are defined and continuous. Then*

$$\begin{aligned}
f(X(t)) &= f(X(0)) + \int_0^t f'(X(s))A(s)\,\mathrm{d}s + \int_0^t f'(X(s))B(s)\,\mathrm{d}W(s) \qquad (1.20)\\
&+ \frac{1}{2}\int_0^t f''(X(s))B^2(s)\,\mathrm{d}s + \sum_{0<s\leq t}(f(X(s)) - f(X(s-))) \quad \square \qquad (1.21)
\end{aligned}$$

In general, there is no explicit form for the sum of the jumps.

1.3 Numerical Solutions

Stochastic calculus arises in the limit $\Delta t \to 0$, but in general a deterministic solution is not available for stochastic differential equations. In order to study the equation one is forced to implement numerical solvers; and there exists a vast collection of different algorithms. The most commonly used method is probably simulation, because of the relatively easy and transparent implementation. Reference [3] is entitled "Numerical Solution of Stochastic Differential Equations", and is almost entirely devoted to simulation schemes. Other important numerical strategies include lattices, finite difference schemes, finite elements, path integration and spectral methods.

Numerical solution implies discretization, and this is where the deterministic analysis diverges from the numerical analysis. For the classical SDE treated in the introduction, the simplest scheme is the first-order and intuitively appealing Euler-Maruyama scheme:

$$X_{n+1} = X_n + A\Delta t + B\Delta W \qquad (1.22)$$

This scheme is very close to the differential equation itself, but it is only of the order of strong convergence 0.5 [3]. Including higher-order terms from the stochastic Taylor expansion will give higher order of convergence. The definition of strong and weak convergence will be taken from Kloeden and Platen's book on the subject [3] mentioned above:

Definition 10 (Strong convergence)**.** A time discrete approximation X^δ converges strongly to Y with order γ at time T if there exists a positive constant C, which does not depend on δ, and a δ_0 such that for each $\delta \in (0, \delta_0)$

$$\epsilon(\delta) = \mathbb{E}(|Y_T - X^\gamma(T)|) \leq C\delta^\gamma \quad \square$$

Stated in words, strong convergence measures the rate of decay of the mean of the error as $\Delta t \to 0$. Conversely, the weak convergence criterion sets a limit on the rate of decay of the error of the mean:

Definition 11 (Weak convergence)**.** A time discrete approximation X^δ converges weakly with order $\beta > 0$ to Y at time T if there exists a positive constant C, which does not depend on δ, a finite $\delta_0 > 0$ and a $2(\beta + 1)$ times continuously differentiable function $g : \mathbb{R}^n \to \mathbb{R}$ such that for each $\delta \in (0, \delta_0)$

$$|\mathbb{E}(g(Y_T)) - \mathbb{E}(g(X^\beta(T))| \leq C\delta^\beta \quad \square$$

Moreover, it can be shown that the Euler-Maruyama scheme is of weak convergence order 1.0 and of strong convergence order 0.5. The strong convergence is shown in figure 1.3, where for each Δt, 1000 realizations of the path have been made. The SDE is the simple

$$\mathrm{d}X = \mu X \,\mathrm{d}t + \sigma X \,\mathrm{d}W \tag{1.23}$$

and the exact solution $Y_T = e^{(\mu - \frac{1}{2}\sigma^2)T + \sigma W_T}$. The mean of the absolute difference between the exact solution and the realized paths $\frac{1}{1000}\sum_{i=1}^{1000}|X_i(T) - Y_T|$ is plotted against a reference line with slope 1/2. The inequality involves an expected value, but this has implications also for single simulation paths: the Markov inequality [4] states that for any random variable X and $a > 0$,

$$\mathbb{P}(|X| \geq a) \leq \frac{\mathbb{E}(|X|)}{a} \tag{1.24}$$

Now, if $a = \Delta t^{1/4}$ and the strong convergence of order $\gamma = 0.5$ holds,

$$\begin{aligned} \mathbb{P}(|X_T^\gamma - Y_T| \geq \Delta t^{1/4}) &\leq \frac{\mathbb{E}(|X_T^\gamma - Y_T|)}{\Delta t^{1/4}} = \frac{C\Delta t^{1/2}}{\Delta t^{1/4}} = C\Delta t^{1/4} \\ \mathbb{P}(|X_T^\gamma - Y_T| < \Delta t^{1/4}) &\geq 1 - C\Delta t^{1/4} \end{aligned}$$

In other words, the error on each simulation can be made arbitrarily small by decreasing Δt (but of course at the expense of CPU time!).

However, it is important to keep in mind that the order of the numerical scheme is not the only factor to take into consideration. When measuring the error here, one assumes that the following errors are negligible:

- The sampling error: the error from the approximation of an expected value by a sample mean.
- Digital floating point rounding errors
- Errors arising from the random number generator

Among these, the sampling error is clearly the most important, since this error decreases proportionally to $\frac{1}{\sqrt{N}}$, where N is the number of realizations.

By including another term from the Itô-Taylor expansion, one obtains the Milstein scheme, which is of strong order 1.0:

$$X_{n+1} = X_n + A\Delta t + B\Delta W + \frac{1}{2}B\frac{\partial B}{\partial x}(\Delta W^2 - \Delta t)$$

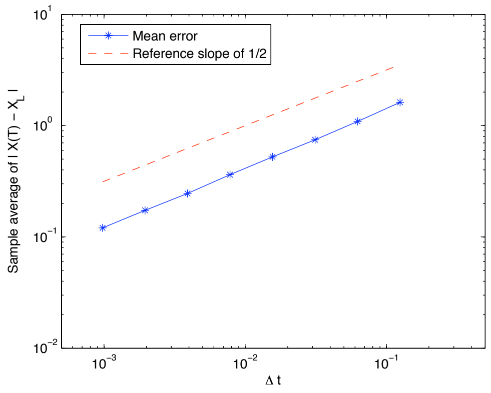


Figure 1.3: Strong convergence of the Euler-Maruyama scheme in Δt for equation 1.23 with $\mu = 2$, $\sigma = 1$.

A disadvantage of the strong schemes is that derivatives of the drift and diffusion factors must be evaluated at each time step. The so-called explicit strong approximations avoid this by using finite difference approximations. For example, from the Milstein scheme several explicit strong schemes can be deduced.

Another problem that arises in analysis of systems of ordinary differential equations is that the system may be stiff, i.e. that $\lambda_1 << \lambda_n$, where $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ are the Liapunov exponents for the system. If the deterministic system is stiff, the corresponding stochastic system is also said to be stiff [3]. This means that consistency and convergence are not enough for the scheme to be used effectively, since information will be lost from error propagation and roundoff errors. There are simulation schemes that handle stochastic stiffness better, namely implicit schemes. They handle numerical instability better, at the expense of the computational cost of solving an additional equation. Many researchers and courses on stochastic differential equations also emphasize variance reduction methods, which are used to reduce the variance of functionals of weak

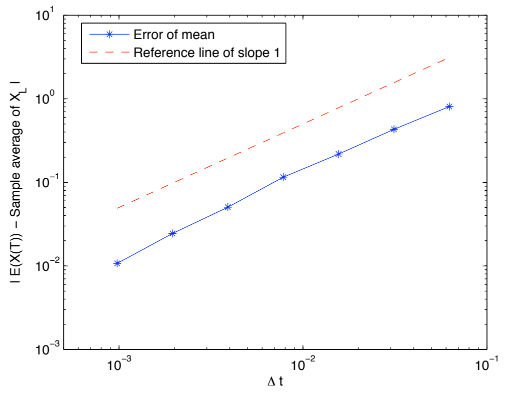


Figure 1.4: Weak convergence of the Euler-Maruyama scheme in Δt for equation 1.23 with $\mu = 2$, $\sigma = 0.1$.

approximations of Itô SDEs. However, simulation schemes are not the main topic of this thesis. Additionally, the convergence rate of simulation methods remains proportional to $\frac{1}{\sqrt{N}}$, where N is the number of repetitions. weak scheme: The order 2.0 weak Taylor scheme

$$L^0 = \frac{\partial}{\partial t} + \sum_{i=1}^{n} A_i \frac{\partial}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{m} B_{i,l} B_{j,l} \frac{\partial^2}{\partial x_i \partial x_j} \tag{1.25}$$

$$L^i = \sum_{j=1}^{n} B_{j,i} \frac{\partial}{\partial x_j} \tag{1.26}$$

$$\tag{1.27}$$

and the general order 2.0 weak Taylor scheme with $m = 1$ is [3]

$$\begin{aligned} X_i^{n+1} &= X_i^n + A_i \Delta t + B_i \Delta W + \frac{1}{2} L^1 B_i [(\Delta W)^2 - \Delta t] \\ &+ \frac{1}{2} L^0 A_i \Delta t^2 + \frac{1}{2} L^0 B_i \Delta W \Delta t + \frac{1}{2} L^1 A_i (\Delta t \Delta W) \end{aligned}$$

The corresponding scheme for $m > 1$ involves multiple Itô integrals in different components of the Wiener process, and will not be stated here.

1.4 The Fast Fourier Transform

The Fast Fourier Transform (FFT) is the name of a collection of algorithms that find the discrete fourier transform (DFT) of an input vector $\{x_i\}_0^{N-1}$ in $N \log(N)$ time. The

DFT is defined as [5]:

$$X_k = \sum_{n=0}^{N-1} x_n \exp(-\frac{i2\pi kn}{N}) \qquad k = 1, \ldots, N$$

The inverse discrete Fourier transform is defined as:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \exp(\frac{i2\pi kn}{N}) \qquad n = 1, \ldots, N$$

The factor N may or may not be included, depending on the software. In this thesis both Matlab and C++ will be used, where in the former it is standard, whereas in the latter, where the fft library used is fftw (”Fastest Fourier Transform in the West” [6]), it is not. Both Matlab and fftw use a combination of Cooley-Tukey and Rader’s algorithms, which decomposes the input vectors into smaller factors and computes the transform for these. This is possible because of the inherent periodicity in the DFT. For example, if the input size has a factor of 2, the first operation would be to split this into two DFTs of equal size:

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n \exp(-\frac{i2\pi kn}{N}) \\ &= \sum_{n=0}^{N/2-1} x_{2n} \exp(-\frac{i2\pi kn}{N/2}) + \exp(-\frac{2\pi ik}{N}) \sum_{n=0}^{N/2-1} x_{2n+1} \exp(-\frac{i2\pi kn}{N/2}) \\ &= E_k + \exp(-\frac{2\pi ik}{N}) O_k \end{aligned}$$

Clearly, since k in this expression is only found in the exponent of the exponentials and $e^{2\pi ik} = e^{2\pi i(k+1)}$, $E_{k+N/2} = E_k$ and $O_k = O_{k+N/2}$. Therefore, $X_{k+N/2}$ can be expressed in terms of O_k and E_k, only manipulating the so-called twiddle factor $\exp(-\frac{2\pi ik}{N})$. This is the core of the Cooley-Tukey algorithm, and can be applied recursively as a divide and conquer strategy. For the factor 2, this is called the radix-2 decimation-in-time form of Cooley-Tukey. Rader’s algorithm handles large prime-number-sized input vectors, and is typically slower by a factor of 3-10, the worst case being $O(N^2)$. This is shown in figure 1.5, where the prime number-sized input vectors are seen to have computational complexity proportional to $O(N \log(N))$. With prime-sized input it is also possible to pad the vector with zeros up to the nearest length that is a power of two.

As shown here, Matlab and C++ are seen to perform at the same speed for vectors with gaussian, random input. The DFT can easily be generalized to m dimensions, with the inverse DFT being the corresponding m-dimensional extension of the inverse DFT in 1 dimension:

$$X_{k_1,\ldots,k_m} = \sum_{n_1=0}^{N_1} \cdots \sum_{n_m=0}^{N_m} x_{n_1,\ldots,n_m} \exp(-2\pi i(\frac{k_1 n_1}{N_1} + \ldots + \frac{k_m n_m}{N_m}))$$

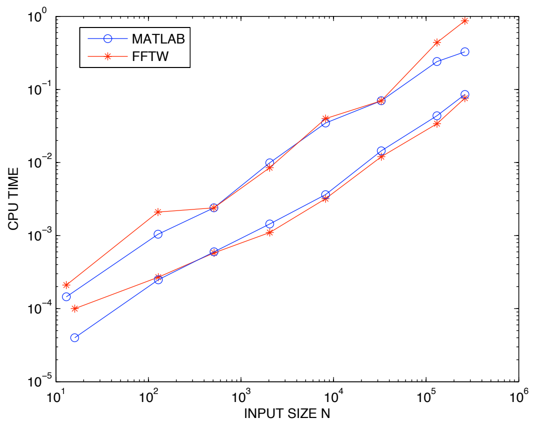


Figure 1.5: CPU time [s] versus input size for one FFT and one inverse FFT in Matlab and C++ (FFTW). The lower line has input vector sizes that are powers of 2, while the upper line has vector sizes that are prime numbers.

Chapter 2

Path Integration and Path Integration by Fast Fourier Transform

The name path integration stems from the famous path integral formulation of quantum mechanics by Richard Feynman in 1948, in which a particle no longer follows a deterministic path, but becomes a random variable and each position is a functional integral over all possible paths. The concept was not new, since e.g. Paul Dirac had been working on a similar formulation decades earlier. However, Feynman was the first to describe how to calculate the sums and relate it to other physics relations such as the Schrödinger equation. The following three postulates were used to describe the path integral [7]:

1. The probability for an event is given by the squared length of a complex number called the probability amplitude
2. The probability amplitude is given by adding together all the contributions of all the histories in the configuration space.
3. The contribution of a history to the amplitude is proportional to $e^{iS/\hbar}$, where $\hbar$ is the reduced Planck's constant, while S is the action of the history, given by the time integral of the Lagrangian along the path.

In order to compute the resulting probability density, a discretization in time and space is usually necessary. In one dimension the path integral can be approximated by

$$\int_{\mathbb{R}} \dots \int_{\mathbb{R}} \exp(\frac{i}{\hbar}\int_0^T \mathcal{L}(x, \dot{x}, t)\,\mathrm{d}t)\,\mathrm{d}x_1 \dots \mathrm{d}x_n \tag{2.1}$$

For the simplest path integral, without the Planck constant (which is a constant of proportionality between the energy and the quantum wavelength of a particle) and the imaginary component (a so-called Wick rotation $t \to it$), the action is the Brownian walk in statistical mechanics, or:

$$S = \int (\frac{\dot{x}^2}{2})\, \mathrm{d}t$$

With the i in the exponential, large deviations are suppressed by cancelling oscillations, whereas without the i it decays exponentially and they are suppressed by small numbers. By considering time steps $\epsilon \to 0$, the probability of going from $x(0) = x$ to $x(T) = y$ is

$$K(x-y, T) = \int_{x(0)=x}^{x(T)=y} \exp(-\int_0^T (\frac{\dot{x}^2}{2})\, \mathrm{d}t) Dx = \int_{x(0)=x}^{x(T)=y} \prod_t \exp(-\frac{1}{2}(\frac{x(t+\epsilon)-x(t)}{\epsilon})^2 \epsilon) Dx$$

Here, Dx represents a finite number (T/ϵ) of integrations over $\mathbb{R}$, and the resulting factors in the integral are Gaussian distributions with variance ϵ^2 and mean $x(t)$. As a result, the product is a repeated series of convolutions in the variables x_{t_i}, and one can take the Fourier transform on both sides:

$$\mathcal{F}(K) = [\mathcal{F}(G)]^{T/\epsilon}$$

where $G(x) = \exp(-\frac{x^2}{2\epsilon})$, so $\mathcal{F}(G) = \sqrt{2\pi\epsilon}\exp(-2\epsilon\pi^2\xi^2)$ and $\mathcal{F}(G)^{T/\epsilon} = \sqrt{2\pi\epsilon}^{T/\epsilon}\exp(-2T\pi^2\xi^2)$

$$K(x, T) \propto \exp(-\frac{x^2}{2T}) \to K(x-y, T) \propto \exp(-\frac{(x-y)^2}{2T})$$

Choosing the normalization constant such that

$$\int_{\mathbb{R}} K(x-y, T)\, \mathrm{d}y = 1$$

one obtains a Gaussian probability distribution for y at time T given x. Obviously, in real applications, the probability distribution of a random variable will not always be a normal distribution, but it is a good point of departure for further analysis. Sometimes a random variable can also be transformed so that the transformed random variable follows a normal distribution, even if the original one does not. For example, this is the case for the standard Black-Scholes formula in financial mathematics, where the stock price is assumed to follow a lognormal distribution.

The path integration algorithm uses precisely what has been developed above: define $t' = t - \Delta t$ and let $\mathbf{x}'$ be the points at t'. For each time step, one uses the transition probability density (TPD) $p(\mathbf{x}, t|\mathbf{x}', t')$ for each point x to integrate over the entire space in t'.

$$p(\mathbf{x}, t) = \int_{\mathbb{R}^n} p(\mathbf{x}, t|\mathbf{x}', t') \cdot p(\mathbf{x}', t')\, \mathrm{d}\mathbf{x}' \tag{2.2}$$

This is what is referred to as the propagator in physics; the probability of a particle going from x to y in time T. Going back to the point of departure, the Langevin SDE 1.1:

$$\begin{aligned} \dot{\mathbf{X}}(t) &= \mathbf{A}(\mathbf{X}(t), t) + \mathbf{B}(\mathbf{X}(t), t)\boldsymbol{\xi}(t) \\ \mathbf{X}(\mathbf{0}) &= \mathbf{x}_0 \end{aligned}$$

where

$$\begin{aligned}
\mathbf{X} &\in \mathbb{R}^n \\
\mathbf{A} &: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n \\
\mathbf{B} &: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^{n \times m} \\
\xi &: \mathbb{R} \to \mathbb{R}^m
\end{aligned}$$

The first k rows of **B** can be assumed to consist of zeros, so that noise only enters through the last $n-k$ dimensions. The path integration algorithm proceeds as follows:

1. Choose a grid for the space and time dimensions. It is convenient to choose a homogeneous time discretization, i.e. such that $0 = t_0 < t_1 < \ldots < t_n = T$ and $\Delta t_i = t_{i+1} - t_i$ is the same for all i. It is important to make the grid so large that the probability distribution integrates to as close to 1 as possible. As with all numerical algorithms, there is a trade-off between accuracy and efficiency, since the algorithm can be made arbitrarily accurate with a fine enough grid. However, the computing power available effectively puts a constraint on the accuracy. This means that x_{max} and x_{min} must be such that $\mathbb{P}(\mathbf{x}_{max}, t) << 1$ and $\mathbb{P}(\mathbf{x}_{min}, t) << 1$ for all t. The path of the probability distribution is typically unknown, so it is possible to make a limited number of simulations first in order to establish sensible upper and lower bounds on the space grid.

2. Find the transition probability density $p(\mathbf{x}, t|\mathbf{x}', t')$. Sometimes it is possible to establish an analytical expression, which obviously makes the subsequent solution easy. More interestingly, when this is not possible a numerical approximation is necessary. This can be a simple Euler-Maruyama step or preferably a Runge-Kutta step. Denote by $\tilde{\mathbf{x}} = r(\mathbf{x}')$ the numerical step for the deterministic part of equation 1.1. The TPD is the product of Dirac delta functions and a probability distribution over the number of dimensions where noise enters, and can be approximated by [8]:

$$p(\mathbf{x}, t|\mathbf{x}', t') = t(\mathbf{x}_\dagger - r(\mathbf{x}'_\dagger)) \cdot \prod_{j=1}^{k} \delta(x_j - r(x'_j))$$

   Here, $t()$ is some probability distribution in $n-k$ variables and $\mathbf{x}_\dagger$ is the truncated vector of length $n-k$. Obviously, if there is no noise, $n-k=0$ and the problem becomes a purely deterministic one. At each time step, one uses the probability density at the previous step, such that the variable transformation requires a multiplication by the Jacobian with respect to the backwards numerical step.

$$p(\mathbf{x}', t')\,\mathrm{d}\mathbf{x}' = p(r^{-1}(\mathbf{x}), t')\,\mathrm{d}r^{-1}(\mathbf{x}) \to p(\mathbf{x}', t') = p(r^{-1}(\mathbf{x}), t')|J_{r^{-1}}|$$

3. If the TPD must be estimated numerically, one can use the order 2.0 weak Taylor scheme 1.28 from Kloeden and Platen [3]. Hence, one must decide how many terms

to include. Naess [9] and Mo [8] include the diagonal of the matrix factor in front of the $\mathrm{d}W\,\mathrm{d}t$ term. Including higher-order terms (HOT) may give rise to noise terms in the dimensions that were originally noise-free. In other words, this is another choice that must be made during the implementation:

- With very small time steps the extra terms are very small, but the TPD also becomes very localized and may be difficult to integrate satisfyingly even with extra interpolation points.
- Including the HOT increases the number of dimensions of the TPD, because noise arises as a cause of cross-terms in the Itô - Taylor expansion. The 'curse of dimensionality' is of course a well-known problem in the field of numerics, since the number of grid points increases by the exponent n.
- If $\mathbf{A}$ and $\mathbf{B}$ are independent of $\mathbf{x}$ and t, the HOT are all zero. This permits large time steps without loss of accuracy.

As an example to illustrate this, take the governing equation with noise for the current in an electric circuit consisting of a source, an inductor, a capacitor and a resistor:

$$LI'' + RI' + \frac{1}{C}I = E_0\omega\cos(\omega t) + \gamma\xi(t)$$

which gives the 2D system with $n = 2$ and $m = 1$:

$$\dot{\begin{bmatrix} I_1 \\ I_2 \end{bmatrix}} = \begin{bmatrix} 0 & 1 \\ -\frac{R}{L} & -\frac{1}{LC} \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{E_0\omega}{L} \end{bmatrix} \cos(\omega t) + \begin{bmatrix} 0 \\ \frac{\gamma}{L} \end{bmatrix} \xi(t)$$

The standard Euler-Maruyama scheme would give:

$$\Delta \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}_i = \begin{bmatrix} 0 & 1 \\ -\frac{R}{L} & -\frac{1}{LC} \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \Delta t_i + \begin{bmatrix} 0 \\ \frac{E_0\omega}{L} \end{bmatrix} \cos(\omega t_i)\Delta t_i + \begin{bmatrix} 0 \\ \frac{\gamma}{L} \end{bmatrix} \Delta W_i$$

while including the $\mathrm{d}W\,\mathrm{d}t$ term from the 2.0 weak scheme would give:

$$\begin{aligned} \Delta I_1 &= A_1\Delta t + B_1\Delta W_1 + \frac{1}{2}(B_{1,1}\frac{\partial A_1}{\partial I_1} + B_{2,1}\frac{\partial A_1}{\partial I_2})\Delta W\Delta t \\ &= I_2\Delta t - \frac{1}{2LC}\Delta t\Delta W \end{aligned}$$

and

$$\begin{aligned} \Delta I_2 &= A_2\Delta t + B_2\Delta W + \frac{1}{2}(B_{1,1}\frac{\partial A_2}{\partial I_1} + B_{2,1}\frac{\partial A_2}{\partial I_2})\Delta t\Delta W \\ &= (-\frac{R}{L}I_1 - \frac{1}{LC}I_2 + \frac{E_0\omega}{L}\cos(\omega t))\Delta t + \frac{\gamma}{L}(1 - \frac{1}{2LC}\Delta t)\Delta W \end{aligned}$$

Clearly, there is now noise in both dimensions, and the TPD must be calculated over $\mathbb{R}^2$ instead of $\mathbb{R}$.

4. Whether an exact TPD is available or not, it is an advantage to use temporary points between the already existing grid points: If the TPD is exact, one may still interpolate the probability surface at these extra points to obtain a finer temporary grid. Typically, with small time steps the TPD is very localized and with a coarse grid only a few points may be non-zero at machine accuracy. In this case, interpolating the probability distribution in the temporary extra grid points may improve the accuracy significantly. If no exact TPD is available, one takes the numerical step backwards from the grid points at time t and generally this step does not coincide with the grid points at time t': here interpolation is strictly necessary. This is illustrated in the figure below. The resulting interpolated value is then multiplied with the distribution of the noise term; the deterministic term and the noise term are treated separately. Three choices have to be made regarding the interpolation:

   - Which interpolation method to use?
   - How many extra points should be used between the grid points?
   - If the TPD is very localized, it may be a waste of time to interpolate over the whole grid. For example, one can interpolate only within the part of the grid that is within seven standard deviations of the mean. If the noise term follows a Gaussian distribution and the standard deviation is 0.1, the contribution of all grid points outside this domain is then multiplied by a factor inferior to $1 \cdot e^{-10}$.

   The easiest choice for the interpolation method is obviously linear interpolation in some form (linear in one dimension, bilinear in 2D, etc.), which uses only the closest grid points, and hence is the least CPU intensive. Both polynomial and spline interpolation are possible, but they have the disadvantage of not being strictly non-negative. Also, the probability distribution usually has the shape of an exponential, and so is not very well described by a polynomial. Polynomial interpolation is also subject to Runge's phenomenon [7]; that the error may increase with the degree of the polynomial. On the other hand, the edges of the grid are more exposed to rounding errors, and splines can become very oscillating because of the continuity requirements (twice continuously differentiable for natural cubic splines). An alternative is to use splines with linear interpolation where the distribution is close to zero. Mo [8] proposes another strategy; to use Bézier splines, which are not interpolating, but can be adjusted to be so (Bézier-Mo splines). Cubic splines are computationally expensive, because with N_i grid points in each dimension i, there are N_i equations in N_i unknowns to be solved for each dimension, and Bézier-Mo are even more expensive. In higher dimensions, the interpolation routine quickly becomes the dominant factor in the total CPU time of the algorithm, so either efficiency or accuracy must be prioritized. Rectangular splines can also be considered instead of cubic splines.

5. In either case, after assigning a numerical value to the probability distribution

in $n-k$ dimensions, a numerical integration method is needed. This is another standard numerical operation, for which there exist multiple good algorithms. In one dimension, we will use Simpson's composite method, which is of order 4 [10]:

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \frac{\Delta x}{3}[f_0 + 2\sum_{j=1}^{n/2-1} f_{2j} + 4\sum_{j=1}^{n/2} f_{2j-1} + f_n] \\ e &\leq \frac{\Delta x^4}{180}(b-a)\max_{\xi\in[a,b]}|f^{(4)}(\xi)|\end{aligned}$$

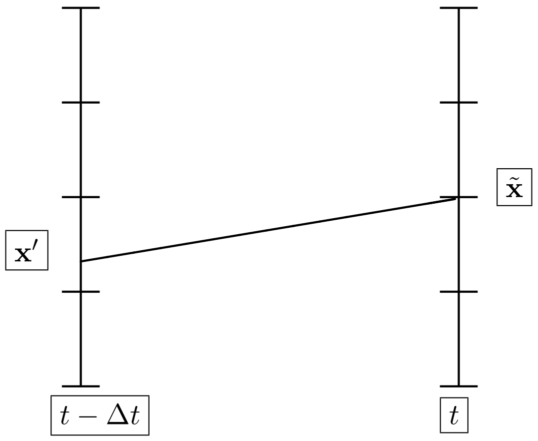
This method can easily be extended to higher dimensions. In two dimensions the coefficients take the following pattern:

$$\begin{bmatrix} 1 & 4 & 2 & \dots & 4 & 1 \\ 4 & 16 & 8 & \dots & 16 & 4 \\ 2 & 8 & 4 & \dots & 8 & 2 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ 1 & 4 & 2 & \dots & 4 & 1 \end{bmatrix}$$

The normalizing constant is then $\frac{\Delta x \Delta y}{9}$ and the error term is of the form $O(\Delta x^4) + O(\Delta y^4)$ [11].

6. The Jacobian is usually relatively easy to calculate: if $\mathbf{A}$ is independent of t, it can be found in the initialization of the algorithm and stored, and if it is time-dependent it is still inexpensive to estimate it analytically or numerically (depending on $\mathbf{A}$) at each iteration in time.

7. After having iterated over the whole grid, increase the time by Δt and return to the first step until T is reached.



2.1 Path Integration by FFT

Mo and Naess propose another strategy for path integration in reference [8]. Here, the equation is treated in two steps: first, solving the deterministic part, then convoluting the solution according to the noise source in 2.3. This requires the noise term to be independent of the physical space, since it is treated after the deterministic step has been completed. An obvious advantage of this algorithm is that it makes use of the FFT algorithm, which is $O(N\log(N))$ in computational complexity, and implies pointwise multiplication of two arrays instead of iterating over the full probability density. The general equation is posited as:

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x},t) + \mathbf{B}\xi(t) \tag{2.3}$$

Here, $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is any deterministic operator, $\mathbf{B} \in \mathbb{R}^{n\times m}$ and $\xi \in \mathbb{R}^m$ is usually Gaussian white noise. In this case, $\mathbf{B}$ takes the form:

$$\mathbf{B} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \\ 0 & & \mathbf{\Sigma} \end{pmatrix} \tag{2.4}$$

where $\mathbf{\Sigma}$ is the $(n-k)\times(n-k)$ covariance matrix for the noise processes. Thus it is symmetric and hence positive definite. This means that $t()$ will be:

$$t(\mathbf{x}_\dagger - \tilde{\mathbf{x}}_\dagger) = \frac{1}{(2\pi\Delta t^2)^{(n-k)/2}|\mathbf{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}_\dagger-\tilde{\mathbf{x}}_\dagger)^T\mathbf{\Sigma}^{-1}(\mathbf{x}_\dagger-\tilde{\mathbf{x}}_\dagger)}$$

Assuming that all noise processes are uncorrelated, $\mathbf{\Sigma}$ is zero except at the diagonal, and the PDF at time t' and the TPD gives a convolution in $\tilde{\mathbf{x}}$:

$$\begin{aligned} p(\mathbf{x},t|\tilde{\mathbf{x}},t') &= \prod_{j=1}^{k} \delta(x_j - \tilde{x}_j) \cdot \prod_{j=k+1}^{n} \frac{1}{\sqrt{2\pi\sigma_j^2\Delta t}} \exp\frac{-(x_j-\tilde{x}_j)^2}{2\sigma^2\Delta t} \\ q(\mathbf{x},t') &= p(r^{-1}(\mathbf{x}),t')|J_{r^{-1}}| \\ p(\mathbf{x},t) &= p(\mathbf{x},t|\tilde{\mathbf{x}},t') * q(\mathbf{x},t') \end{aligned}$$

where the convolution theorem comes in handy:

$$h = f * g \to \mathcal{F}(h) = \mathcal{F}(f)\cdot\mathcal{F}(h)$$

As a result, the difference from regular PI is just that instead of numerical integration, one multiplies two $n-k$-dimensional probability distributions in Fourier space and makes the inverse transformation. Using an interpolation routine that is $O(N\log(N))$, this should significantly improve the efficiency of the algorithm.

Furthermore, since the PI by FFT algorithm is devised for SDEs with constant or time-dependent noise terms, it is useful to be able to transform more general SDEs to this form. From equation 7, it is known that one can rewrite any SDE with a function

$f()$ to obtain another SDE. One can assume that the deterministic part of the equation can be arbitrary, while the third term in equation 7 must be independent of $\mathbf{X}$. This implies that

$$\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\sum_{l=1}^{m} B_{il}\,\mathrm{d}W_l\right) \tag{2.5}$$

must be such that

$$\frac{\partial f}{\partial x_i} B_{il} = C_{il}(t).$$

In other words, one must solve this differential equation for each dimension. If there are multiple noise terms, the system is overdetermined, and it may not be possible to transform the SDE to the desired form. This is the case if one of the noise terms is multiplied by a factor x_i and another is not. This means that PI by FFT usually only works with multiple additive noise terms or a single multiplicative noise term.

Chapter 3

Fokker-Planck Equations in Physics

The Fokker-Planck (also known as the Kolmogorov forward equation) equation is named after Adriaan Fokker and Max Planck, respectively Dutch and German physicists. It can describe a multitude of physical, biological, financial and other phenomena, but one of the most common usages is the description of the position of a particle in one dimension or the position and velocity of a particle in two dimensions. In a particle accelerator such as the Large Hadron Collider (LHC) at the Centre Europeen de Recherche Nucléaire (CERN) on the border between Switzerland and France, particles travel in bunches, typically around 10^{11} per beam bunch ($1.15 \cdot 10^{11}$ at beam injection for the design energy level (14 TeV) is the number given by CERN) [12]. The single-particle perspective is essential in analyzing the scattering and decays of elementary particles resulting from the collisions, but may be inconvenient for describing the behavior of the beam. The charged particles in accelerators are subjected to strong external electromagnetic fields, and in the limit of an infinitesimal number of particles, the Fokker-Planck equation can be used to describe the beam motion to a high level of accuracy. However, what equation to use is highly dependent on the specific particle accelerator technology. Accelerators are either linear (linacs) or cyclic (synchrotrons), and they accelerate charged particles (electrons, protons or heavy-ions) to relativistic velocity levels. In synchrotrons, particles are accelerated once or more per revolution by radio frequency cavities. This frequency and the B field must be synchronized with the particle velocity, hence the name synchrotron. A beam typically makes of the order 10^5 revolutions before colliding or being safely stopped in a large block of concrete, known as a collimator.

The largest electron linac in the world, built in 1962, is at Stanford, California, and measures 3 km [13]. Somewhat surprisingly it has been in use since its construction, and has produced data for three Nobel Prizes in Physics. However, a common misconception is the difference between an accelerator and a detector: the linear accelerator at Stanford has hosted several experiments with different detectors. For example, from 1999 to 2008 it was host to the BaBar experiment, where the goal was to study B mesons, an elementary particle of the standard model of particle physics[13], from electron-positron collisions. At CERN, there are four different detectors: CMS (Compact Muon Solenoid), ATLAS, ALICE (A Large Ion Collider Experiment) and LHCb (Large Hadron Collider beauty). The former two are the largest ones, and are constructed to detect both proton and heavy-ion collisions. The b in LHCb refers to the bottom quark in the standard model, and the detector is designed for so-called b-physics; studying bigger particles (hadrons) that include a b quark. ALICE is dedicated to heavy-ion collisions and the fourth fundamental force; the strong force. This theoretical field is called quantum chromodynamics, and describes the interactions of particles that carry color charge, namely quarks and gluons. This is only possible at extreme energy densities, where a new phase of matter; quark-gluon plasma, is expected.

The smooth distribution picture described by the Fokker-Planck equation is useful for the study of collective beam instabilities. Noise usually enters through the momentum dimensions of the equation, not through the position dimensions. For proton beams, there are many potential sources of noise ... However, in order to make a system described by classical statistics valid, one must ensure that the number of quantum states must far exceed the number of particles in the beam. If not, one enters the field of Fermi-Dirac statistics, that applies to identical particles with half-integer spin (fermions) in a system in thermal equilibrium. No two particles can then occupy the same state, a famous phenomenon called the Pauli Exclusion Principle [13]. This effect has a big influence on the state of the multi-particle system, and since both quarks (the building blocks of protons and neutrons) and leptons (e.g. electrons) have half-integer spin, it must be carefully considered before choosing a model. The number of quantum states available for protons is [12]:

$$N = \frac{2V}{\hbar^3} \tag{3.1}$$

where V is the volume of the 6-dimensional phase space. For 2D damped harmonic motion, the system becomes:

$$\begin{aligned} \dot{q} &= \omega p \\ \dot{p} &= -\omega q - 2\alpha p \end{aligned}$$

Adding a diffusion term gives the Fokker-Planck equation:

$$\frac{\partial \rho}{\partial t} + \omega p \frac{\partial \rho}{\partial q} + (-\omega q - 2\alpha p)\frac{\partial \rho}{\partial p} = 2\alpha\rho + D\frac{\partial^2 \rho}{\partial p^2}$$

This form of F-P has an equilibrium solution which is Gaussian (shown in the next section), and is not unrealistic for beam distributions in electron storage rings. Assuming that the motion in the three dimensions (x and y are the horizontal and vertical axes respectively in the transversal plane, while z is tangent to the longitudinal synchrotron motion) are independent, each 2D system can be modeled separately. For such a model, Chao [12] gives the following table for α, D and the rms in each dimension:

	Horizontal betatron (x)	Vertical betatron (y)	Longitudinal synchrotron (z)
q	$-\frac{c}{\omega_x}x'$	y	z
p	x	$\frac{c}{\omega_y}y'$	$\frac{-\alpha_c c}{\omega_s}\delta$
α	$\frac{U_0}{2E_0T}$	$\frac{U_0}{2E_0T}$	$\frac{U_0}{E_0T}$
D	$\frac{R^2}{E_0^2\nu_x^4}D_u$	$\frac{(c/\omega_y E_0)^2}{2\gamma^2}D_u$	$(\frac{\alpha_c c}{\omega_s E_0})^2 D_u$
rms	$\sigma_x = \frac{c}{\omega_x}\sigma_{x'} = \frac{1}{\nu_x^2}\sqrt{C_q\gamma^2 R}$	$\sigma_y = \frac{c}{\omega_y}\sigma_{y'} = \frac{1}{\nu_y}\sqrt{\frac{C_q R}{2}}$	$\sigma_\delta = \frac{\omega_s}{\alpha_c c}\sigma_z = \sqrt{\frac{C_q\gamma^2}{2R}}$

Table 3.1: Radiation damping and quantum diffusion factors for three dimensions of motion in a synchrotron.

The energy loss per revolution for a relativistic electron in a synchrotron is approximately [13]:

$$\begin{aligned} U_0 &= \frac{4\pi}{3}(\frac{r_0}{R})\mathrm{mc}^2\gamma^4 \\ \gamma &= \frac{1}{\sqrt{1-v^2/c^2}} \end{aligned}$$

As can be seen from the formula, this energy loss increases very rapidly with the velocity v of the particles. The loss is mainly in the form of emission of quantized photons. This is the reason why the electron linear accelerator at Stanford was built - the synchrotron radiation becomes the limiting factor on the final energy obtained in electron synchrotrons such as LEP (Large Electron Positron Collider, an electron-positron collider that made necessary the excavation of the 27 km long tunnel now being used by the LHC) at CERN.

$$D_u = \frac{55}{48\sqrt{3}} r_0 \hbar m c^4 \frac{\gamma^7}{R^3}$$

is the photon emission diffusion rate,

$$C_q = 3.84 \cdot 10^{-13} \quad \mathrm{m}$$

is a fundamental constant defined for convenience [12]. Furthermore, r_0 is the classical radius of the particle ($2.82 \cdot 10^{-15}$ m for an electron), m is the rest mass ($9.11 \cdot 10^{-31}$ kg for an electron), $E_0 = mc^2\gamma$ is the energy of the particles and α_c is the momentum compaction factor (particles with different longitudinal momentum have different equilibrium radii), T is the revolution time and R is the uniform radius of the storage ring. Finally, particles oscillate around the intended orbit in the transverse x and y directions. The number of oscillations per orbit is called the betatron tune, and $\nu_{x,y} = \frac{\omega_{x,y}R}{c}$ are horizontal and vertical betatron tunes respectively. For the x-motion, the synchrotron radiation noise occurs in x rather than x', so p and q are inverted for this dimension.

Now, returning to the quantum states in equation 3.1, V could be of the order of [12]:

$$V = \sigma_x \sigma_{x'} \sigma_y \sigma_{y'} \sigma_z \sigma_\delta P_0^3$$

The particles under consideration are usually electrons, but take the LHC beam as an example, which is accelerated to approximately 99.999999 % of the speed of light, which in turn gives $\gamma = 7.1 \cdot 10^3$. $R = 4290$ m and other typical parameters would be $\alpha_c = 0.0002$, $\nu_x = \nu_y = 80$, $\omega_s = 40\ s^{-1}$. After reaching the stationary distribution, inserting these values into table 3.1 gives an N_q of about 10^{20}, which exceeds the number of protons per bunch by a factor of 10^9. There is no problem in ignoring the Fermi statistics in this case.

The Vlasov equation is another dynamical system, without noise, that can also be used to find a wealth of beam dynamical properties:

$$\frac{\partial \rho}{\partial t} + \rho\omega \frac{\partial \rho}{\partial v} + [-\omega x + N\omega \int_v^\infty W(q'-q)\,\mathrm{d}q' \int_{-\infty}^\infty \rho(x',v',t)\,\mathrm{d}x'] \frac{\partial \rho}{\partial v} = 0 \tag{3.2}$$

It is a partial-differential-integral equation, nonlinear in ρ, so exact solutions are not readily available. The Vlasov equation is most relevant for proton accelerators, where the synchrotron radiation is negligible. This effect occurs when high-energy particles travel in a curved path through magnetic fields. However, close to the speed of light it is 10^{13} times larger for electrons than for protons.

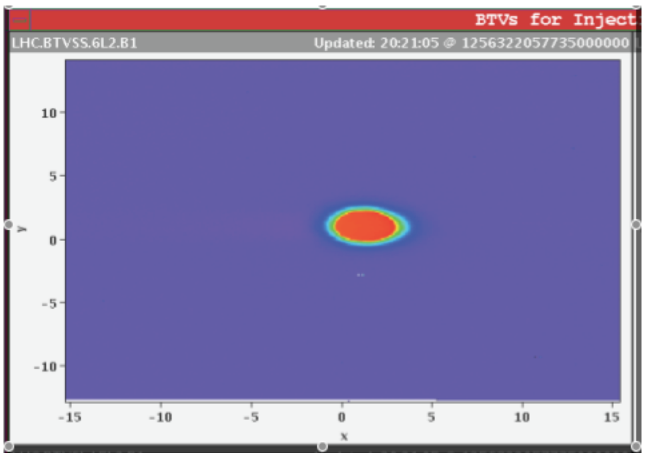
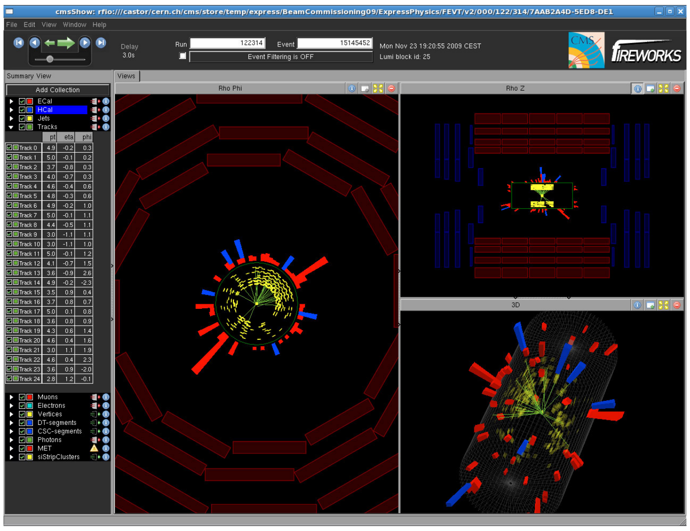


Figure 3.1: Transversal distribution of the first beam entering point 2 of the LHC on 23 October 2009.

The questions that F-P can help answer are

- The longtime behavior of the beam
- The probability for the particle to hit the vacuum chamber


cmsShow: rfio:///castor/cern.ch/cms/store/temp/express/BeamCommissioning09/ExpressPhysics/FEVT/v2/000/122/314/7AAB2A4D-5ED8-DE1

Delay 3.0s

Run 122314 Event 15145452

Event Filtering is OFF

Mon Nov 23 19:20:55 2009 CEST

Lumi block id: 25

	pt	eta	phi
Track 0	4.9	-0.2	0.3
Track 1	5.0	-0.1	0.2
Track 2	3.7	-0.8	0.3
Track 3	4.0	-0.7	0.3
Track 4	4.6	-0.4	0.6
Track 5	4.8	-0.3	0.6
Track 6	4.9	-0.2	1.0
Track 7	5.0	-0.1	1.1
Track 8	4.4	-0.5	1.1
Track 9	3.0	-1.1	1.1
Track 10	3.0	-1.1	1.0
Track 11	5.0	-0.1	1.2
Track 12	4.1	-0.7	1.5
Track 13	3.6	-0.9	2.6
Track 14	4.9	-0.2	-2.3
Track 15	3.5	0.9	0.4
Track 16	3.7	0.8	0.7
Track 17	5.0	0.1	0.8
Track 18	3.6	0.8	0.9
Track 19	4.3	0.6	1.4
Track 20	4.6	0.4	1.6
Track 21	3.0	1.1	1.9
Track 22	4.6	0.4	2.3
Track 23	3.6	0.9	-2.0
Track 24	2.8	1.2	-0.1

Figure 3.2: Event display from the first collision events at CMS on 23 November 2009.

- Average fluctuations of the particles around the periodic design orbit

- Time evolution of the probability density of the beam

3.1 General Form - Derivation

The Fokker-Planck equation is closely related to the Langevin stochastic differential equation (1.1), introduced by Langevin in 1908 [14]. The probability distribution of the solution is usually the statistical property of main interest. When the noise process $\eta(t)$ in the Langevin equation is Gaussian white noise, i.e. it can be represented as the time derivative of a stationary process with independent and normally distributed increments on non-overlapping intervals [15], it corresponds to the ordinary Fokker-Planck equation.

There are at least two formulations of the Itô SDE, since it is always connected with at least one partial differential equation. Going back to the formulation in equation 1.19:

$$\mathrm{d}\mathbf{X}(t) = \mathbf{A}(\mathbf{X}(t), t)\,\mathrm{d}t + \mathbf{B}(\mathbf{X}(t), t)\,\mathrm{d}\mathbf{W}(t)$$

Let t and x be the backward variables and T and y the forward variables (in time). By introducing a Borel-measurable, strictly nonnegative function $h(y) \in \mathbb{R}$, we have in one dimension:

$$
\begin{aligned}
\mathrm{d}h(X) &= \frac{\partial h}{\partial x}\,\mathrm{d}X + \frac{1}{2}\frac{\partial^2 h}{\partial x^2}\,\mathrm{d}X\,\mathrm{d}X \\
&= (A\frac{\partial h}{\partial x} + B^2\frac{1}{2}\frac{\partial^2 h}{\partial x^2})\,\mathrm{d}t + B\frac{\partial h}{\partial x}\,\mathrm{d}W \\
\int_{x(t)}^{y(T)} \mathrm{d}h(z) &= \int_t^T A(u,y)\frac{\partial h}{\partial y}(y)\,\mathrm{d}u + \frac{1}{2}\int_t^T B^2(u,y)\frac{\partial^2 h}{\partial y^2}(y)\,\mathrm{d}u \\
&+ \int_{W(t)}^{W(T)} B(u,y)\frac{\partial h}{\partial y}(y)\,\mathrm{d}W
\end{aligned}
$$

By integrating the expression on the left-hand side, taking expectations and using the fact that $X(u)$ has probability density $p(t,u,x,y)$, the last term on the right-hand side becomes zero and one obtains

$$
\begin{aligned}
\int_{-\infty}^{+\infty} h(y)p(t,T,x,y)\,\mathrm{d}y &= h(x) + \int_t^T\int_{-\infty}^{+\infty}\frac{\partial h}{\partial y}(y)A(u,y)p(t,u,x,y)\,\mathrm{d}y\,\mathrm{d}u \\
&+ \frac{1}{2}\int_t^T\int_{-\infty}^{+\infty}\frac{\partial^2 h}{\partial y^2}(y)B^2(u,y)p(t,u,x,y)\,\mathrm{d}y\,\mathrm{d}u
\end{aligned}
$$

Now, by assuming that $\lim_{y\to\pm\infty} h(y) = 0$ and $\lim_{y\to\pm\infty} h'(y) = 0$ and integrating the second integral by parts:

$$
\begin{aligned}
\int_{-\infty}^{\infty} Ap\frac{\partial h}{\partial y}\,\mathrm{d}y &= [Aph]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} h\frac{\partial}{\partial y}(Ap)\,\mathrm{d}y = -\int_{-\infty}^{\infty} h\frac{\partial}{\partial y}(Ap)\,\mathrm{d}y \\
\int_{-\infty}^{\infty} B^2p\frac{\partial^2 h}{\partial y^2}\,\mathrm{d}y &= [B^2ph']_{-\infty}^{\infty} - \int_{-\infty}^{\infty}\frac{\partial h}{\partial y}\frac{\partial}{\partial y}[B^2p]\,\mathrm{d}y \\
&= -[h\frac{\partial}{\partial y}(B^2p)]_{-\infty}^{\infty} + \int_{-\infty}^{\infty} h\frac{\partial^2}{\partial y^2}[B^2p]\,\mathrm{d}y \\
&= \int_{-\infty}^{\infty} h\frac{\partial^2}{\partial y^2}[B^2p]\,\mathrm{d}y
\end{aligned}
$$

Substituting this into the equation and differentiating with respect to T,

$$
\begin{aligned}
\int_{-\infty}^{\infty} h(y)\frac{\partial p(t,T,x,y)}{\partial T}\,\mathrm{d}y &= \int_{-\infty}^{\infty}(-h(y)\frac{\partial}{\partial y}(A(T,y)p(t,T,x,y)) \\
&+ \frac{1}{2}h(y)\frac{\partial^2}{\partial y^2}(B^2(T,y)p(t,T,x,y)))\,\mathrm{d}y
\end{aligned}
$$

Since $h(y) \geq 0$, we are left with what is known as the forward Kolmogorov equation or the one-dimensional Fokker-Planck equation.

$$
\to \frac{\partial p(t,T,x,y)}{\partial T} = -\frac{\partial}{\partial y}(A(T,y)p(t,T,x,y)) + \frac{1}{2}h(y)\frac{\partial^2}{\partial y^2}(B^2(T,y)p(t,T,x,y))
$$

In n dimensions, the Itô SDE 1.19 can be re-written as the following Fokker-Planck equation:

$$\frac{\partial \rho}{\partial t} = -\sum_{i=1}^{n} \frac{\partial(\rho A_i)}{\partial x_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j} \sum_{k=1}^{m} (B_{il} B_{jl} \rho) \tag{3.3}$$

3.2 Fokker-Planck in Accelerator Physics

The typical SDE for accelerator physics looks like:

$$\begin{aligned} \dot{x} &= v && (3.4) \\ \dot{v} &= -a_1(x) - a_2(x,v) + \sqrt{2\sigma}\eta(t) && (3.5) \end{aligned}$$

The first, simple example of the SDE for particle oscillations around the periodic design trajectory of a storage ring is linear in x and v. It is an harmonic oscillator with damping and additive noise, $K > 0, \gamma > 0$:

$$\begin{aligned} \dot{x} &= v && (3.6) \\ \dot{v} &= -Kx - \gamma v + \sqrt{2\sigma}\eta(t) && (3.7) \end{aligned}$$

According to 3.3, the Fokker-Planck partial differential equation of the harmonic oscillator becomes

$$\frac{\partial \rho}{\partial t} = -\frac{\partial(\rho v)}{\partial x} + \frac{\partial(Kx\rho + \gamma v \rho)}{\partial v} + \sigma \frac{\partial^2}{\partial v^2}\rho \tag{3.8}$$

This equation has a closed form exact solution, shown in the next section, which is useful for comparison purposes for the numerical schemes.

Figure 4.1 shows the phase plot of the 2D system 3.6 and 3.7 without noise. For the system to be Liapunov stable, the eigenvalues need to be strictly negative, or negative with no repeated zero eigenvalue [16]. Also, solutions of the linear system $\dot{x} = A(t)x + f(t)$ have the same stability properties as the regular linear system. This, however, is for a deterministic function $f()$. But any system that is unstable without noise would be of no interest, since stability is essential for a storage ring. For the harmonic oscillator, the eigenvalues are:

$$\begin{aligned} \lambda_1 &= -\frac{1}{2}\gamma + \sqrt{\frac{1}{4}\gamma^2 - K} \\ \lambda_2 &= -\frac{1}{2}\gamma - \sqrt{\frac{1}{4}\gamma^2 - K} \end{aligned}$$

These are both strictly negative for $\gamma^2/4 > K$.

Another model that incorporates two noise terms, whereof one is dependent on x, is the stochastic Duffing oscillator:

$$\begin{aligned} \dot{x} &= v && (3.9) \\ \dot{v} &= -\omega^2[(\alpha + \sqrt{2D_1}\eta_1(t))x + \epsilon x^3)] - 2\tau\omega v + \sqrt{2D_2}\eta_2(t) && (3.10) \end{aligned}$$

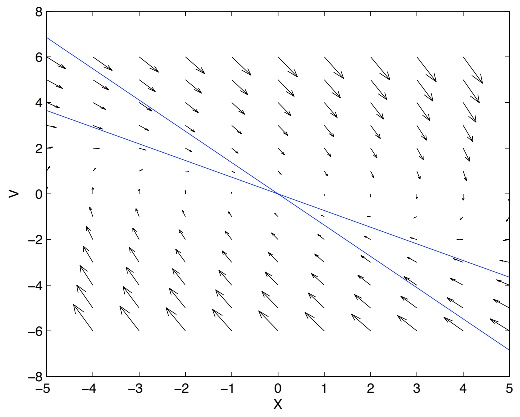


Figure 3.3: Phase plot for the dynamical system 3.6 and 3.7 without noise. $K = 1$ and $\gamma = 2.1$

This SDE becomes the following FP:

$$\frac{\partial \rho}{\partial t} = -\frac{\partial(\rho v)}{\partial x} + \frac{\partial}{\partial v}[\omega^2 x(x + \epsilon x^2) + 2\tau\omega v] + \frac{1}{2}\frac{\partial^2}{\partial v^2}(D_1\omega^4 x^2 + D_2) \tag{3.11}$$

3.3 Exact Solutions

Subrahmanyan Chandrasekhar, an Indian-American astrophysicist and Nobel prize laureate, in his celebrated 1943 paper 'Stochastic Problems in Physics and Astronomy' [14], provides a long discussion on Brownian motion and an exact solution for a class of Fokker-Planck equations. Here, he shows the relation between the Fokker-Planck equation and the Langevin equation. The Langevin equation, when it takes the form of an harmonically bound particle with noise as in 3.6 and 3.7 can also be written as

$$\ddot{x} + \gamma\dot{x} + Kx = \sqrt{2\sigma}\xi(t)$$

The homogeneous differential equation can then be solved by variation of parameters, and

$$\begin{aligned}\mu_1 &= \frac{1}{2}\gamma + \sqrt{\frac{1}{4}\gamma^2 - K}\\ \mu_2 &= \frac{1}{2}\gamma - \sqrt{\frac{1}{4}\gamma^2 - K}\end{aligned}$$

such that $x = a_1 e^{\mu_1 t} + a_2 e^{\mu_2 t}$. Two independent first integrals of the 2D system are then

$$\begin{aligned}\xi &= (x\mu_1 - v)e^{-\mu_2 t}\\ \eta &= (x\mu_2 - v)e^{-\mu_1 t}\end{aligned}$$

It is not explicitly shown in the paper, but in these two variables, the equation 3.8 simplifies and only the terms $\frac{\partial\rho}{\partial t}$, $\gamma\rho$ and $\frac{\partial^2\rho}{\partial v^2}$ are left, and with the variable transformation the latter term becomes (with $\frac{\partial\xi}{\partial v} = -e^{-\mu_2 t}$, $\frac{\partial\eta}{\partial v} = -e^{-\mu_1 t}$, $\frac{\partial^2\xi}{\partial v^2} = 0$ and $\frac{\partial^2\eta}{\partial v^2} = 0$)

$$\begin{aligned}
\frac{\partial^2\rho}{\partial v^2} &= \frac{\partial}{\partial v}(\frac{\partial\rho}{\partial v}) = \frac{\partial}{\partial v}(\frac{\partial\rho}{\partial\xi}\frac{\partial\xi}{\partial v} + \frac{\partial\rho}{\partial\eta}\frac{\partial\eta}{\partial v}) \\
&= \frac{\partial}{\partial v}(\frac{\partial\rho}{\partial\xi})\frac{\partial\xi}{\partial v} + \frac{\partial\rho}{\partial\xi}\frac{\partial^2\xi}{\partial v^2} + \frac{\partial}{\partial v}(\frac{\partial\rho}{\partial\eta})\frac{\partial\eta}{\partial v} + \frac{\partial\rho}{\partial\eta}\frac{\partial^2\eta}{\partial v^2} \\
&= (\frac{\partial^2\rho}{\partial\xi^2}\frac{\partial\xi}{\partial v} + \frac{\partial^2\rho}{\partial\xi\partial\eta}\frac{\partial\eta}{\partial v})\frac{\partial\xi}{\partial v} + (\frac{\partial^2\rho}{\partial\eta^2}\frac{\partial\eta}{\partial v} + \frac{\partial^2\rho}{\partial\xi\partial\eta}\frac{\partial\xi}{\partial v})\frac{\partial\eta}{\partial v} \\
&= (\frac{\partial\xi}{\partial v})^2\frac{\partial^2\rho}{\partial\xi^2} + 2\frac{\partial\xi}{\partial v}\frac{\partial\eta}{\partial v}\frac{\partial^2\rho}{\partial\xi\partial\eta} + (\frac{\partial\eta}{\partial v})^2\frac{\partial^2\rho}{\partial\eta^2} \\
&= e^{-2\mu_2 t}\frac{\partial^2\rho}{\partial\xi^2} + 2e^{(-\mu_1-\mu_2)t}\frac{\partial^2\rho}{\partial\xi\partial\eta} + e^{-2\mu_1 t}\frac{\partial^2\rho}{\partial\eta^2}
\end{aligned}$$

and equation 3.8 becomes

$$\frac{\partial\rho}{\partial t} = \gamma\rho + \sigma(e^{-2\mu_2 t}\frac{\partial^2\rho}{\partial\xi^2} + 2e^{(-\mu_1-\mu_2)t}\frac{\partial^2\rho}{\partial\xi\partial\eta} + e^{-2\mu_1 t}\frac{\partial^2\rho}{\partial\eta^2})$$

and with the further transformation $\rho = pe^{\gamma t}$,

$$\frac{\partial p}{\partial t} = \sigma(e^{-2\mu_2 t}\frac{\partial^2 p}{\partial\xi^2} + 2e^{(-\mu_1-\mu_2)t}\frac{\partial p}{\partial\xi\partial\eta} + e^{-2\mu_1 t}\frac{\partial^2 p}{\partial\eta^2})$$

By Lemma II in Chandrasekhar's paper [14], an equation of the form

$$\frac{\partial f}{\partial t} = \phi^2(t)\frac{\partial^2 f}{\partial x^2} + \phi(t)\psi(t)\frac{\partial^2 f}{\partial x\partial y} + \psi^2(t)\frac{\partial^2 f}{\partial y^2} \tag{3.12}$$

which tends to $\delta(x - x_0, y - y_0)$ as $t \to 0$ has the exact solution

$$f(x,y) = \frac{1}{2\pi\Delta^{1/2}}e^{-(ax^2+2hxy+by^2)/2\Delta} \tag{3.13}$$

where

$$\begin{aligned}
a = 2\int_0^t \psi^2(t)\,\mathrm{d}t \quad h &= -2\int_0^t \phi(t)\psi(t)\,\mathrm{d}t \quad b = 2\int_0^t \phi^2(t)\,\mathrm{d}t \\
\Delta &= ab - h^2
\end{aligned}$$

From this, Zorzano et al. [17] conclude that the exact solution of 3.8 at time t is

$$\rho = \frac{e^{\gamma t}e^{-[a(\xi - xi_0)^2 + 2h(\xi-\xi_0)(\eta-\eta_0) + b(\eta-\eta_0)^2]/2\Delta}}{2\pi\Delta^{1/2}}$$

with

$$\begin{aligned}
a &= 2\sigma\int_0^t e^{-2\mu_1 t}\,\mathrm{d}t = \frac{\sigma}{\mu_1}[1-e^{-2\mu_1 t}] \\
b &= 2\sigma\int_0^t e^{-2\mu_2 t}\,\mathrm{d}t = \frac{\sigma}{\mu_2}[1-e^{-2\mu_2 t}] \\
h &= -2\sigma\int_0^t e^{-(\mu_2+\mu_1)t}\,\mathrm{d}t = \frac{-2\sigma}{\mu_1+\mu_2}[1-e^{-(\mu_1+\mu_2)t}]
\end{aligned}$$

However, this probability distribution does not integrate to 1. This can easily be verified as follows: the variable transformation can be written as

$$\begin{bmatrix}\xi\\ \eta\end{bmatrix} = \begin{bmatrix}e^{-\mu_2 t} & 0\\ 0 & e^{-\mu_1 t}\end{bmatrix}\cdot\begin{bmatrix}\mu_1 & -1\\ \mu_2 & -1\end{bmatrix}\cdot\begin{bmatrix}x\\ y\end{bmatrix}$$

Now, $\tilde{\rho} = e^{\xi^T A\xi/2\Delta}$, with

$$A = \begin{bmatrix}a & h\\ h & b\end{bmatrix}$$

which has $\det(A) = \Delta = ab - h^2$ and $\det(A/2\Delta) = \frac{1}{4\Delta}$. Since A is symmetric and positive definite, one can diagonalize it by an orthogonal matrix Q: $Q^T AQ = \Lambda$. Thus, with $Qx_i = y_i$,

$$\int\int_{\mathbb{R}^2} e^{\mathbf{x}^T A\mathbf{x}}\,\mathrm{d}x_1\,\mathrm{d}x_2 = \int\int_{\mathbb{R}^2} e^{\mathbf{y}^T\Lambda\mathbf{y}}\,\mathrm{d}y_1\,\mathrm{d}y_2$$

$$= \int_{\mathbb{R}} e^{-\lambda_1 y_1^2}\,\mathrm{d}y_1\int_{\mathbb{R}} e^{-\lambda_2 y_2^2}\,\mathrm{d}y_2 = \sqrt{\frac{\pi}{\lambda_1}}\sqrt{\frac{\pi}{\lambda_2}} = \frac{\pi}{\sqrt{\det(A)}}$$

so

$$\int\int_{\mathbb{R}^2}\tilde{\rho}\,\mathrm{d}\xi\,\mathrm{d}\eta = 2\pi\sqrt{\Delta}$$

Now, in terms of (x, v) the Jacobi determinant of the variable transformation is

$$J = e^{-(\mu_1+\mu_2)t}(\mu_2-\mu_1) = e^{\gamma t}\sqrt{\gamma^2-4K}$$

and inserted in the integral above, this gives

$$\int\int_{\mathbb{R}^2}\tilde{\rho}|J|^{-1}\,\mathrm{d}x\,\mathrm{d}v = 2\pi\sqrt{\Delta}|J|^{-1} = \frac{2\pi\sqrt{\Delta}}{e^{\gamma t}\sqrt{\gamma^2-4K}}$$

This is constant for a fixed time t, so the exact probability density for (x, y) is

$$\rho(x,y) = \frac{e^{\gamma t}\sqrt{\gamma^2-4K}}{2\pi\sqrt{\Delta}}\tilde{\rho}$$

Thus the difference from what is stated in Zorzano's paper is the factor $\sqrt{\gamma^2-4K}$. The result was also verified numerically. An example plot of the exact distribution is shown in figure 3.4.

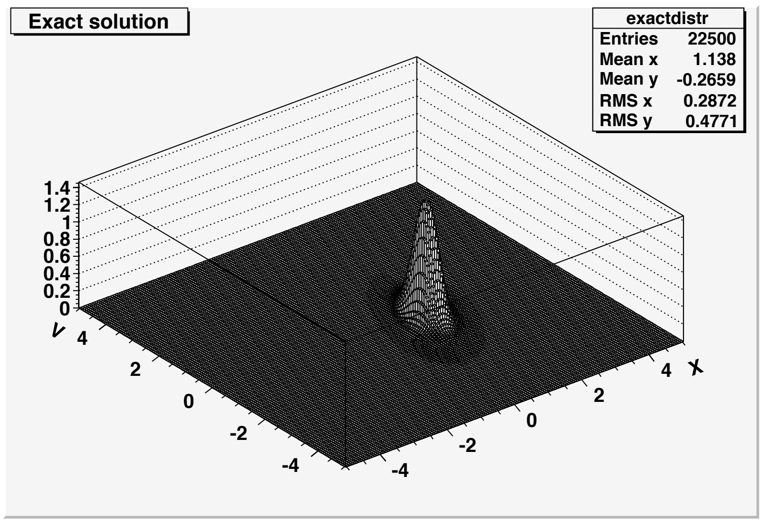


Figure 3.4: Exact solution of the harmonic oscillator at time $T = 1$ with $K = 0.8, \gamma = 2.4, \sigma = 0.6$ and $x_0 = 1, v_0 = 1$.

3.4 Numerical Solution by Simulation

Kloeden and Platen's book [3] "Numerical Solutions of Stochastic Differential Equations" is entirely dedicated to simulation schemes. Usually, 'numerical solutions' of equations with noise in the literature implies solution by simulation. However, even though there are vast amounts of different simulation schemes and variance reduction methods, even the simple Euler-Maruyama scheme usually gives good results with small Δt. Since the convergence is proportional to $1/\sqrt{N}$ realizations, this is usually the limiting factor. The pseudo random number generator used to sample the Brownian motion path will be 'ran2' from Numerical Recipes in C++ [11]. This routine returns uniform random deviates between 0 and 1, has a period of more than 10^{18} and supposedly provides 'perfect' (to machine precision) random numbers. The standard library generator in C++ is called 'rand', and is a so-called linear congruential generator, which generates a sequence of integers based on the recurrence relation

$$I_{j+1} = aI_j + c \quad \mathrm{mod}(m). \tag{3.14}$$

This type of generators is usually not very reliable, since m can be as low as 32767, which would introduce a high level of periodicity in the output. On the machine used for all simulations in this thesis, $m > 2 \cdot 10^9$. The mean and standard deviation of a uniform distribution on $(0, 1)$ are $\frac{1}{2}$ and $\frac{1}{2\sqrt{3}}$ respectively. The error in the sample mean as a function of the number of generated random variables is shown in figure 3.5. The author of this thesis is not quite sure what to conclude from these results - but based on the continuously decreasing error for the 'ran2' routine and recommendations in the literature [11] [18], this number generator will be used. The pseudo uniform random variable is then used in an analytical transformation to get a normal random variable $n(0, 1)$ - shown in figure 3.6.

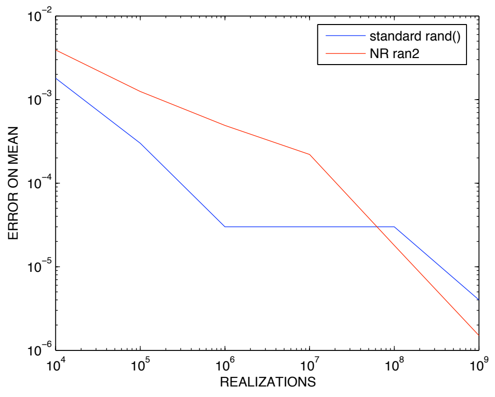


Figure 3.5: Error in the sample mean ($|\bar{x} - 1/2|$) for the pseudo random number generators rand and ran2 in C++.

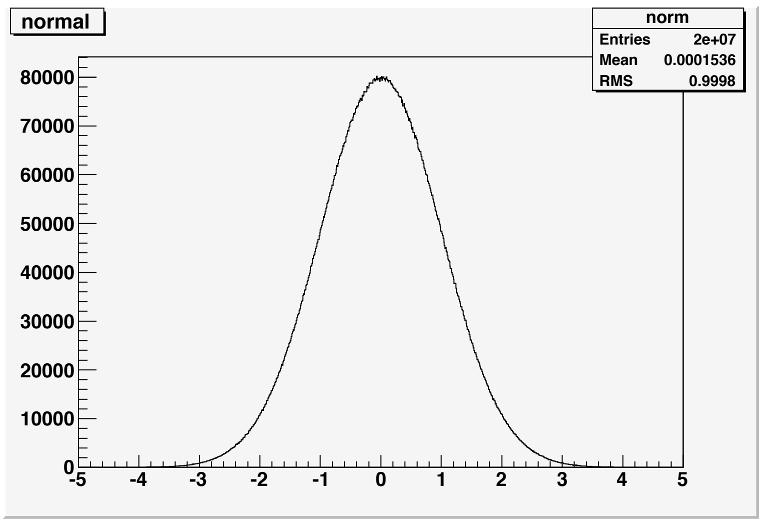


Figure 3.6: 20 million simulated $n(0, 1)$ random variables using the pseudo random number generator ran2.

The numerical schemes used for all simulations in this thesis are the order 2.0 weak Itô-Taylor scheme and the Euler-Maruyama scheme.

3.4.1 Harmonic Oscillator

The error will be measured as

$$||e||_\infty = ||\rho_{sim} - \rho_{ex}||_\infty = \max_{i,j}\{\rho_{sim}(i,j) - \rho_{ex}(i,j)\}. \tag{3.15}$$

Although Zorzano et al. [17] use the RMS error averaged over the whole grid, this will not be used here, since the distribution is usually relatively concentrated and the average RMS does not reflect the true error in the probability distribution. By increasing the size of the grid one would reduce the RMS error, while the max error would remain the same. Other possible benchmarks of the accuracy could be the first and second order moments, which are often of interest. First, convergence will be tested in N and Δt for an example with $K = 0.9, \gamma = 2.4, \sigma = 0.5$ and $T = 2.3$. The errors measured against the exact solution with the Euler-Maruyama scheme are shown in table 3.2 and with the Taylor 2.0 scheme in table 3.3. The order 2.0 weak Taylor scheme becomes

$$\begin{aligned} x_{n+1} &= x_n + v_n\Delta t + \frac{1}{2}(-Kx_n - \gamma v_n)\Delta t^2 + \sqrt{\frac{\sigma}{2}}\Delta W \Delta t \\ v_{n+1} &= v_n + (-Kx_n - \gamma v_n)\Delta t + \frac{1}{2}(-Kv_n + \gamma(Kx_n + \gamma v_n))\Delta t^2 + (\sqrt{2\sigma} - \sqrt{\frac{\sigma}{2}}\gamma\Delta t)\Delta W \end{aligned}$$

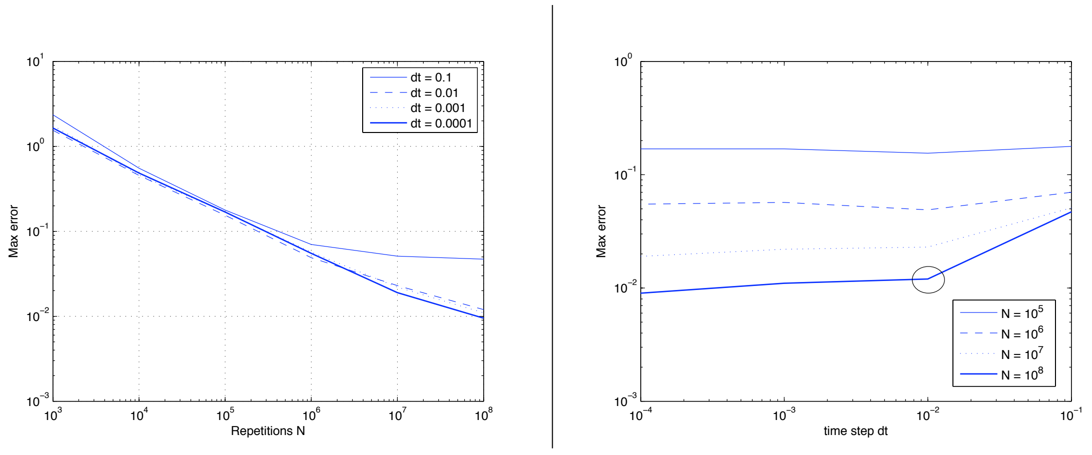


Figure 3.7: Convergence in N and Δt for the simulated Taylor 2.0 scheme.

The results from table 3.3 are the best and show that

- In order to get good results, the number of realizations N must be much larger than the grid size of the histogram that is used. For example, with only 1000 realized paths and a grid of $201^2 = 40401$ bins, each bin will either not be filled or only incremented a few times.

Number of realizations	10^3	10^4	10^5	10^6	10^7
$\|\|e\|\|_\infty$, $\Delta t = 10^{-1}$	1.70	0.403	0.218	0.129	0.109
$\|\|e\|\|_\infty$, $\Delta t = 10^{-2}$	1.84	0.463	0.156	0.052	0.032
$\|\|e\|\|_\infty$, $\Delta t = 10^{-3}$	1.75	0.499	0.161	0.059	0.023
$\|\|e\|\|_\infty$, $\Delta t = 10^{-4}$	1.65	0.485	0.169	0.055	0.028

Table 3.2: Convergence in N and Δt for Euler-Maruyama scheme. $K = 0.9, \gamma = 2.4, \sigma = 0.5$, $T = 2.3$ and a grid of 201×201 points.

- If the grid size is in proportion with N and N is large enough, numerical solution by simulation does not converge significantly in Δt for $\Delta t < 0.01$. The resulting stagnation of the curve is encircled in black in figure 3.7. Theoretically, it is the expected value of the distribution that converges $O(\Delta t^2)$ (weakly). This is shown in figure 3.8, with a reference line of slope 1. However, the Euler scheme is seen to converge faster than Taylor 2.0, which, surprisingly, does not exhibit order 2.0 convergence. It may be that N must be even larger before the convergence in Δt is seen.
- If the grid size is in proportion with N and Δt is small enough, numerical solution by simulation converges by order 0.5 in increasing N.

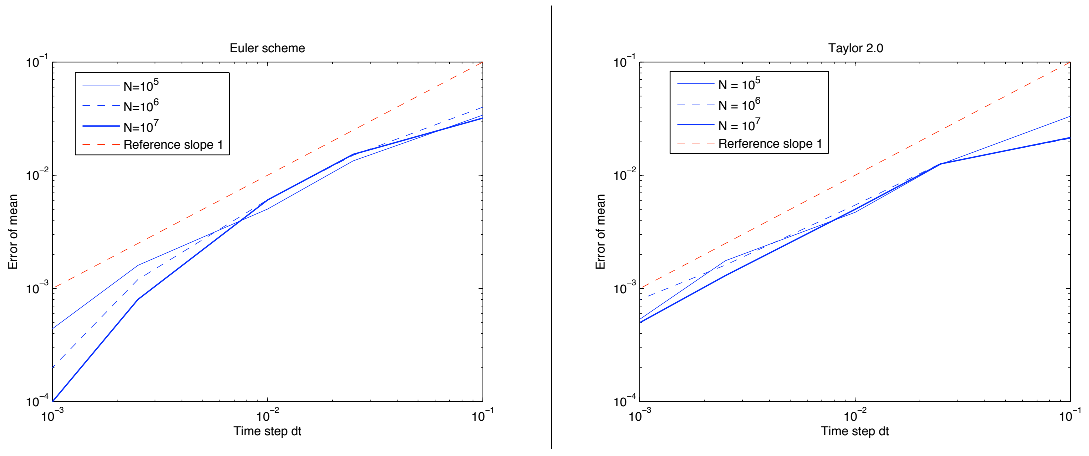


Figure 3.8: Weak convergence in Δt for the Euler scheme (left) and the Taylor 2.0 scheme (right).

3.4.2 Stochastic Duffing Equation

Since there is no closed form solution for the damped stochastic Duffing equation, it is unfortunately not possible to find the errors for different step sizes. Of course, it

Number of realizations	10^3	10^4	10^5	10^6	10^7	10^8
Error, $\Delta t = 10^{-1}$	2.36	0.554	0.178	0.070	0.051	0.047
Error, $\Delta t = 10^{-2}$	1.55	0.455	0.155	0.049	0.023	0.012
Error, $\Delta t = 10^{-3}$	1.76	0.459	0.169	0.057	0.022	0.011
Error, $\Delta t = 10^{-4}$	1.65	0.485	0.169	0.055	0.019	0.009

Table 3.3: Convergence of the max error in N and Δt for the order 2.0 weak Taylor scheme. $K = 0.9, \gamma = 2.4, \sigma = 0.5$, $T = 2.3$ and a grid of 201×201 points.

would be possible to compare with a simulated result using a very large N and a small Δt, but the convergence was seen not to be very impressive for the harmonic oscillator: the combination of $N = 10^8$ and $\Delta t = 0.0001$ took close to 70 hours of CPU time and still gave a max error of 0.009 - far from machine precision. As the convergence of the solution is $O(\sqrt{N})$, it would take too long to obtain a highly accurate distribution by simulation. However, one can at least get an impression of the effect of the multiplicative noise: figure 3.9 shows how the distribution becomes more smeared out further away from the origin. Physically, this means that there is increased noise on the particles in the periphery of the beam axis - and there is a higher probability of loosing them. Effects such as these must be carefully considered and modeled prior to the construction of particle accelerators. For example, the x^3 term in the equation can be due to the interaction between the circulating beams in the storage ring or an octupole, which are both proportional to the third power of the distance from the origin.

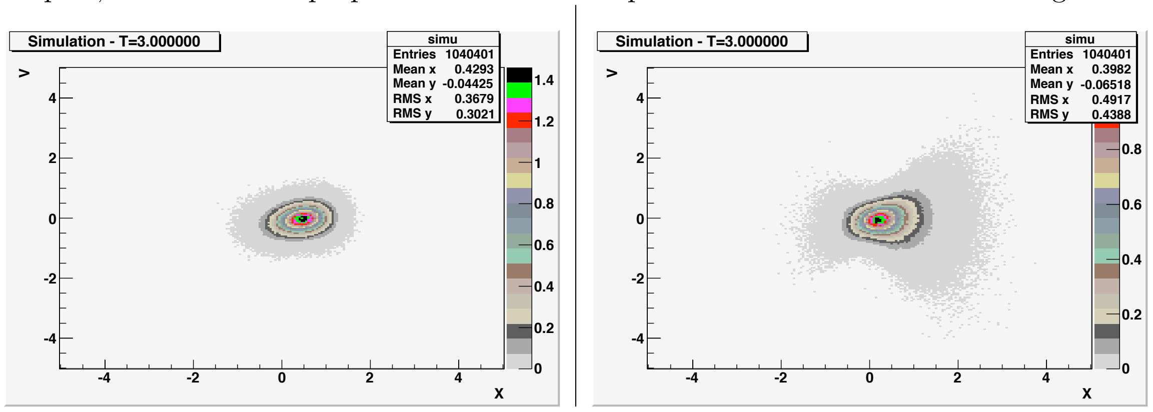


Figure 3.9: Simulation of the stochastic Duffing equation. To the left, $D_1 = 0.0, D_2 = 0.23$ (only additive noise), to the right, $D_1 = 1.5, D_2 = 0.23$ (additive and multiplicative noise).

3.5 Numerical Solution by Path Integration

The PI algorithm is implemented as follows:

1. The grid size is specified in the results that are given. However, Δx and Δt must be in reasonable proportion: Figure 3.10 demonstrates how the convoluting normal distribution can become too localized for the grid. That is, Δt must not be too small, since information is lost, but it must certainly not be too large either, since the theoretical framework is based on the limit $\Delta t \to 0$.

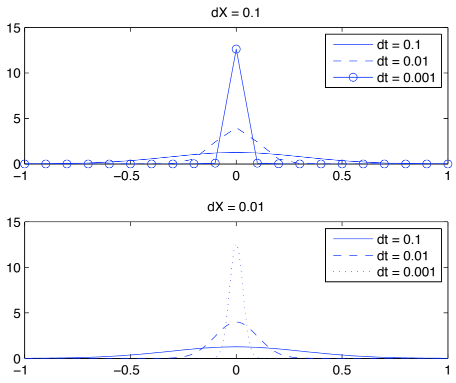


Figure 3.10: Normal distribution $n(0, \sqrt{dt})$ for different Δx and Δt.

2. The harmonic oscillator has an exact solution for the deterministic part of the equation: with λ_1, λ_2 as the eigenvalues and v_1, v_2 as the eigenvectors of the system, the exact solution is

$$\begin{bmatrix} x \\ v \end{bmatrix}(t) = C_1 v_1 e^{\lambda_1 t} + C_2 v_2 e^{\lambda_2 t}$$

The exact solution is used to compare with the Runge-Kutta solver. For the Duffing equation, no exact solution is easily available, so only the latter will be used.

3. Both the Euler-Maruyama scheme and the weak Taylor 2.0 scheme are used.

4. Both the relevant systems are two-dimensional. Thus, the linear interpolation method includes a cross-term such that the interpolating function passes through all four points on a square in the grid:

$$f(x, y) = ax + by + cxy + d$$

This is bilinear interpolation, and is very fast since the interpolating function on each square is independent of the neighboring functions. The coefficients a, b, c, d

can be easily determined given the point coordinates that are to be interpolated and the values at the grid points. On the other hand, this is not very accurate for large grid squares, and the gradient is not continuous at the boundaries of each grid square. In the same spirit, one can also use birectangular or bicubic splines. These methods are first- and second-order continuous respectively at the boundaries. Constructing bicubic splines in one dimension implies iterating four times over the given values at the grid points to ensure second-order continuity. To interpolate one value in two dimensions, N one-dimensional splines are created across the rows of the table, before one spline is made down the column at the intersection at the relevant point. Since one needs to make $N \cdot N$ evaluations, it is advantageous to pre-compute and store derivative information in one direction so that only one spline construction, which is $O(N^2)$, and one evaluation, which is $O(1)$, are needed per point. However, this means that at each time step, bicubic splines are of computational cost $O(N^2(N^2+1)) = O(N^4)$ instead of $O(N^2)$ for bilinear interpolation. The spline scheme is implemented as in reference [11]. Birectangular splines may be an alternative if CPU time is an important constraint.

5. Simpson's method is used for the numerical integration. The integration will only be performed within the limits of the normal distribution that are superior to 10^{-10}.

6. The jacobi determinant of the backwards step is calculated by finite differences. For both the relevant systems it is independent of time, so it can be set in the variable initialization part of the algorithm.

3.5.1 Harmonic Oscillator

Using the same example parameters as for the simulation schemes, the results in table 3.4 are obtained. They suggest that with linear interpolation, adding the extra inter-gridpoints may 'save' the numerical solution from exploding, but no extra precision is gained by adding more points than necessary. For example, the results in the second column ($M = 2$) (in bold font) show that the results are better with a time step of 0.1 than a time step of 0.01. Also, with $\Delta x = 0.025$ and $\Delta t = 0.002$, the error is seen to increase with the number of extra points M. However, decreasing Δx and Δt simultaneously increases the accuracy significantly. This is a reasonable result, since the extra points only add information from linear interpolation, which has an error that is $O(\Delta x^2)$, while composite Simpson's numerical integration has error $O(\Delta x^4)$. It is therefore no surprise that the error is not reduced by the extra points with bilinear interpolation.

Table 3.5, however, is obtained using cubic splines. Here, one sees that for e.g. $\Delta x = 0.1, \Delta t = 0.01$, the algorithm does not work: the backward probability distribution becomes too localized. A maximum error of 0.32 is clearly unacceptable for a probability distribution. Moreover, in this case the final distribution integrates to 1.4, a clear sign that something is wrong. Adding two extra inter-gridpoints on the V-axis

Extra integration points M	0	2	5	10
$\|\|e\|\|_\infty$, $\Delta x = 0.1$, $\Delta t = 0.1$	0.146	**0.146**	0.146	0.146
$\|\|e\|\|_\infty$, $\Delta x = 0.1$, $\Delta t = 0.01$	0.322	**0.187**	0.204	0.205
$\|\|e\|\|_\infty$, $\Delta x = 0.025$, $\Delta t = 0.01$	0.035	0.042	0.042	0.043
$\|\|e\|\|_\infty$, $\Delta x = 0.025$, $\Delta t = 0.002$	0.031	0.060	0.069	0.075

Table 3.4: Euler-Maruyama scheme, bilinear interpolation: errors for different numbers of extra integration points. The parameters are the same as for the example in table 3.3.

Extra integration points M	0	2	5	10
$\|\|e\|\|_\infty$, $\Delta x = 0.1$, $\Delta t = 0.1$	0.091	0.091	0.091	0.091
$\|\|e\|\|_\infty$, $\Delta x = 0.1$, $\Delta t = 0.01$	**0.316**	9.89e-3	9.90e-3	$9.90e-3$
$\|\|e\|\|_\infty$, $\Delta x = 0.1$, $\Delta t = 0.001$	N/A	4.52	$1.01e-3$	$1.01e-3$
$\|\|e\|\|_\infty$, $\Delta x = 0.1$, $\Delta t = 0.0001$	N/A	N/A	N/A	4.3

Table 3.5: Euler-Maruyama scheme, bicubic splines: errors for different numbers of extra integration points. The parameters are the same as for the example in table 3.3.

solves the problem and the solution becomes highly accurate. In other words, the extra integration points permit a smaller time step than the grid usually allows, which in turn gives higher total accuracy with a high-order interpolation procedure. On the other hand, cubic interpolation quickly becomes very time consuming - integration with $\Delta x = 0.02, \Delta t = 0.002$ and $M = 2$ took 48 hours on the computer used in this thesis, while the corresponding solution with bilinear interpolation took only a few minutes. Some of the combinations of step sizes were not possible to run due to lack of CPU power, even on the centralized batch job server of CERN, which has a maximum time limit of one week.

Now, trying the same procedure with the weak Taylor 2.0 scheme, the path integration becomes slightly more complicated since noise now enters through two dimensions. The deterministic part of the equation is still solved using RK4, so the only difference is the scale of the Brownian motion. It becomes, with ΔW a $n(0,1)$ random variable:

$$\begin{bmatrix} \sqrt{\frac{\sigma}{2}}\Delta t \\ \sqrt{2\sigma} - \sqrt{\frac{\sigma}{2}}\gamma\Delta t \end{bmatrix} \sqrt{\Delta t}\Delta W$$

Now, it is seen that the scale of the noise in dimension 1 will be very small compared to the noise in dimension 2. For instance, with $\sigma = 0.5$, $\gamma = 2.2$ and $\Delta t = 0.1$, one gets

$$\begin{bmatrix} 0.0158 \\ 0.3023 \end{bmatrix} \Delta W$$

This gives rise to two new potential problems - first of all, the numerical integration must be carried out over two dimensions instead of only one, which increases the CPU time

Δt	0.1	0.02	0.01	0.005	0.001
$\Delta x = 0.2$, $\|\|e\|\|_\infty$	0.098	0.315	0.390	0.484	0.658
$\Delta x = 0.1$, $\|\|e\|\|_\infty$	0.032	0.075	0.175	0.262	0.514
$\Delta x = 0.04$, $\|\|e\|\|_\infty$	0.090	0.030	0.022	0.042	0.192
$\Delta x = 0.02$, $\|\|e\|\|_\infty$	0.101	$6.45e-3$	0.016	0.011	0.029
$\Delta x = 0.01$, $\|\|e\|\|_\infty$	0.101	0.014	$3.24e-3$	$7.96e-3$	0.012
$\Delta x = 0.005$, $\|\|e\|\|_\infty$	0.104	0.016	$6.76e-3$	$1.63e-3$	$3.63e-3$
$\Delta x = 0.002$, $\|\|e\|\|_\infty$	0.105	0.017	8.29e-3	N/A	N/A

Table 3.6: Error for the PI scheme: Harmonic oscillator with Taylor 2.0 variance in dimension 2, bilinear interpolation.

Δt	0.1	0.01	0.005	0.001	0.0001
$\Delta x = 0.2$, $\|\|e\|\|_\infty$	0.161	0.081	0.084	0.087	X
$\Delta x = 0.1$, $\|\|e\|\|_\infty$	0.104	$8.57e-3$	$4.23e-3$	$1.01e-3$	$1.81e-3$
$\Delta x = 0.04$, $\|\|e\|\|_\infty$	0.104	$8.57e-3$	$4.23e-3$	$8.11e-4$	N/A
$\Delta x = 0.02$, $\|\|e\|\|_\infty$	0.104	$8.58e-3$ (M=1)	$4.24e-3$	N/A	N/A

Table 3.7: Error for the PI scheme: Harmonic oscillator with Taylor 2.0 variance in dimension 2, bicubic splines.

significantly (by a factor $O(N)$, if N is the number of points in this dimension). Second, there is now the same problem as above: a very localized transition distribution, only in two dimensions. Trying an example with these input parameters confirms that the solution 'explodes', i.e. it does not yield a solution, and that the algorithm is much slower. However, if one disregards the added noise term in x that causes these annoyances, and only adds the extra term in v, the scheme should still be improved. Comparing tables 3.6 and 3.7 with 3.4 and 3.5 show that using the Taylor 2.0 variance scheme improves the solution by a factor of between 1 and 10 for both bilinear interpolation and bicubic splines. These values are plotted in figure 3.13, and there are two noticeable effects:

- For a given Δx, Δt must be of the same order as Δx or larger. If Δt is smaller, the problem shown in figure 3.10 comes into effect and the error increases.

- For a given Δt, Δx can be decreased indefinitely. However, there is a minimum error that is reached for every Δt, so no extra accuracy will be obtained once the numerical solution has stabilized. A small cusp appears as the global minimum on each curve (marked with a circle on the figure) - the reason for this is unknown, but there seems to be an optimal Δx for each Δt.

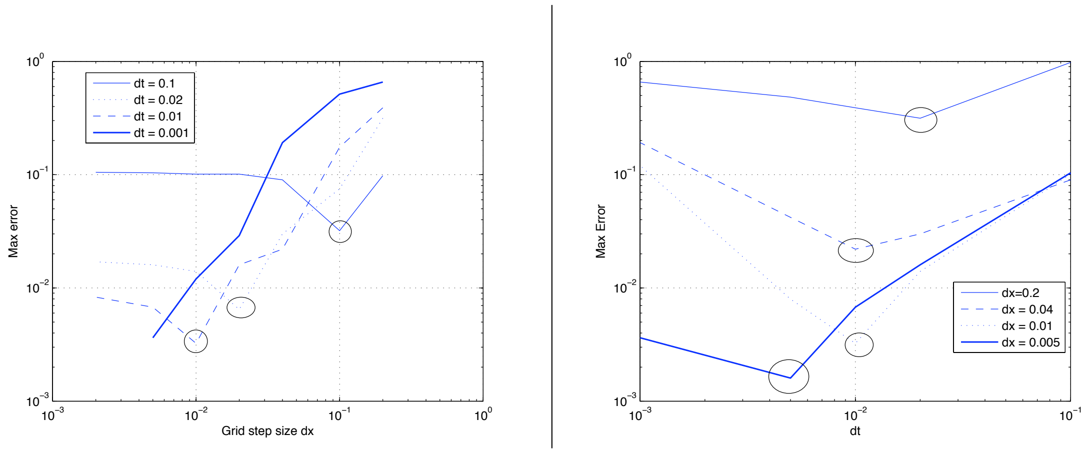


Figure 3.11: Convergence in Δx and Δt for the path integration scheme. The minimum error for each Δx or Δt is encircled in black.

3.5.2 Stochastic Duffing Equation

The stochastic harmonic oscillator is a simple model for beam dynamics. Slightly more advanced is the stochastic Duffing equation. In the 2D SDE system 3.9, 3.10, one incorporates two noise terms, whereof one is multiplicative. The PI technique only permits multiple noise terms when the factors x and y in front of them are the same, so the Duffing equation can only be solved if it is either of the purely additive form with uncorrelated noise:

$$\begin{aligned}\dot{x} &= v\\ \dot{v} &= -\omega^2[\alpha x + \epsilon x^3] - 2\tau\omega v - \omega^2\sqrt{2D_1}\eta_1(t) + \sqrt{2D_2}\eta_2(t)\end{aligned}$$

or for example with one multiplicative noise term in x:

$$\begin{aligned}\dot{x} &= v\\ \dot{v} &= -\omega^2[\alpha x + \epsilon x^3] - 2\tau\omega v - \omega^2 x\sqrt{2D_1}\eta_1(t))\end{aligned}$$

Now, the former version is only an extension of the harmonic oscillator to two noise sources instead of one, and is easy to solve using any of the algorithms introduced earlier, since it is a double convolution in two Gaussians in one dimension. Here, the main objective is to test the performance of the PI algorithm, so by introducing the first noise term in the position dimension instead, one obtains:

$$\dot{x} = v + \sqrt{2D_1}\eta_1(t) \tag{3.16}$$

$$\dot{v} = -\omega^2[\alpha x + \epsilon x^3] - 2\tau\omega v + \sqrt{2D_2}\eta_2(t) \tag{3.17}$$

This would correspond to the position of the particle beam having noise. Obviously, this SDE no longer corresponds to the Fokker-Planck equation that was the point of departure, but when one models the full six-dimensional phase space, noise typically enters in the last three dimensions, while the first three are deterministic. It is therefore highly relevant to test the efficiency of the algorithm with noise in multiple dimensions, although the physical implications of this two-dimensional equation might be unrealistic.

However, this last version must be transformed to one where the noise distribution is not skewed - it must be symmetric. Theorem 7 stated that if one transforms the stochastic system to one in the variables (f_1, f_2), for $\mathrm{d}f_1$ and $\mathrm{d}f_2$

$$\begin{aligned}
\mathrm{d}f_i &= \frac{\partial f_i}{\partial x}A_1\,\mathrm{d}t + \frac{\partial f_i}{\partial v}A_2\,\mathrm{d}t + \frac{\partial f_i}{\partial x}B_1\,\mathrm{d}W + \frac{\partial f_i}{\partial v}B_2\,\mathrm{d}W \\
&+ \frac{1}{2}(\frac{\partial^2 f_i}{\partial x^2}B_1^2 + 2\frac{\partial^2 f_i}{\partial x \partial v}B_1 B_2 + \frac{\partial^2 f_i}{\partial v^2}B_2^2)\,\mathrm{d}t
\end{aligned}$$

Now, f_1 and f_2 can be any analytic functions, but in this case one simply wants constant or time-dependent coefficients in the stochastic term. In the first dimension, the coefficient is already zero, so $f_1(x, v) = x$ for simplicity. Since $B_1 = 0$, one must have $\frac{\partial f_2}{\partial v}\omega^2\sqrt{2D_1}x = C$ in the equation for f_2. Setting $C = \omega^2\sqrt{2D_1}$, $f_2(x, v) = \frac{v}{x}$. The SDE for the function $f(x, v) = \begin{bmatrix} x \\ v/x \end{bmatrix}$ takes the form

$$\begin{aligned}
\mathrm{d}f_1 &= f_1 f_2\,\mathrm{d}t \\
\mathrm{d}f_2 &= (-f_2^2 - \omega^2(\alpha + \epsilon f_1^2) - 2\tau\omega f_2)\,\mathrm{d}t - \omega^2\sqrt{2D_1}\,\mathrm{d}W_1
\end{aligned}$$

The weak Taylor 2.0 scheme for the transformed stochastic Duffing equation with multiplicative noise becomes:

$$\begin{aligned}
\Delta f_1 &= f_1 f_2 \Delta t + \frac{1}{2}(f_1 f_2^2 + f_1^2 f_2)\Delta t^2 - \frac{1}{2}\omega^2\sqrt{2D_1}f_1\Delta t\Delta W \\
\Delta f_2 &= (-f_2^2 - \omega^2\alpha - \omega^2\epsilon f_1^2 - 2\tau\omega f_2)(1 - (f_2 + \tau\omega)\Delta t)\Delta t - \omega^2(\epsilon f_1^2 f_2 + D_1)\Delta t^2 \\
&- \omega^2\sqrt{2D_1}(1 - (f_1 + \tau\omega)\Delta t)\Delta W
\end{aligned}$$

This numerical scheme has the same disadvantage as discussed earlier - the distribution becomes very localized in one dimension. Hence it is only advantageous to include all terms if the space grid has a very high resolution. The Euler-Maruyama scheme is easier to implement and will be used here, since the numerical solution of this problem is for illustration purposes - not to test the accuracy. Because $f_2 = v/x$, the equation will be solved in the half-plane $x \in (0, \infty)$. After solving by PI in the (f_1, f_2) space, the final distribution must be transformed back to the original coordinates (x, v), so a multiplication by the Jacobi determinant is required. In this case, this becomes

$$J = \det(\begin{bmatrix} \frac{\partial x}{\partial f_1} & \frac{\partial x}{\partial f_2} \\ \frac{\partial v}{\partial f_1} & \frac{\partial v}{\partial f_2} \end{bmatrix}) = \det(\begin{bmatrix} 1 & 0 \\ f_2 & f_1 \end{bmatrix}) = f_1$$

so that

$$\int\int p(x,v)\,\mathrm{d}x\,\mathrm{d}v = \int\int p(f_1,f_2)\cdot J\cdot \mathrm{d}f_1\,\mathrm{d}f_2$$

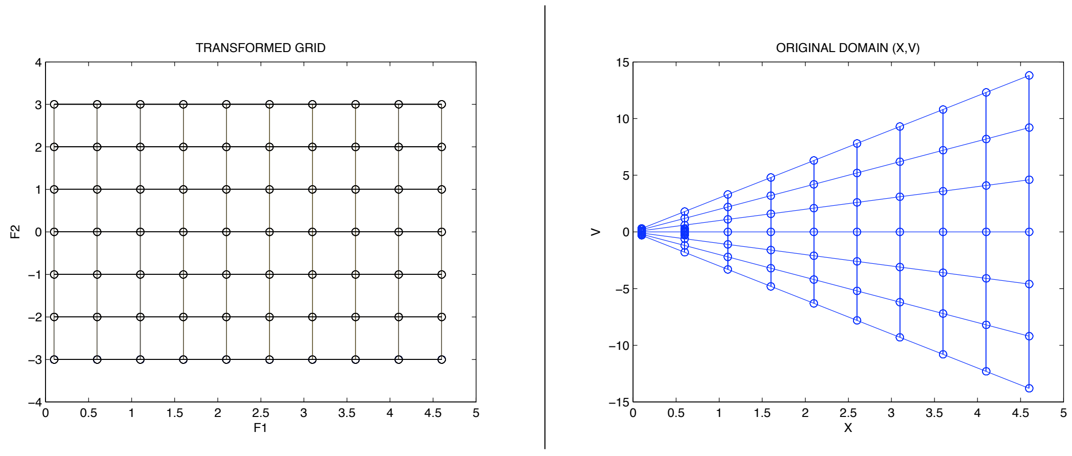


Figure 3.12: A uniform grid in the variables (f_1, f_2) and the corresponding grid in (x, v) after the backward transformation.

As seen in figure 3.12, a disadvantage of transforming the coordinates is that the grid generally will not be uniform and equidistant in both sets of variables. In this case, it is difficult to treat distributions close to the origin.

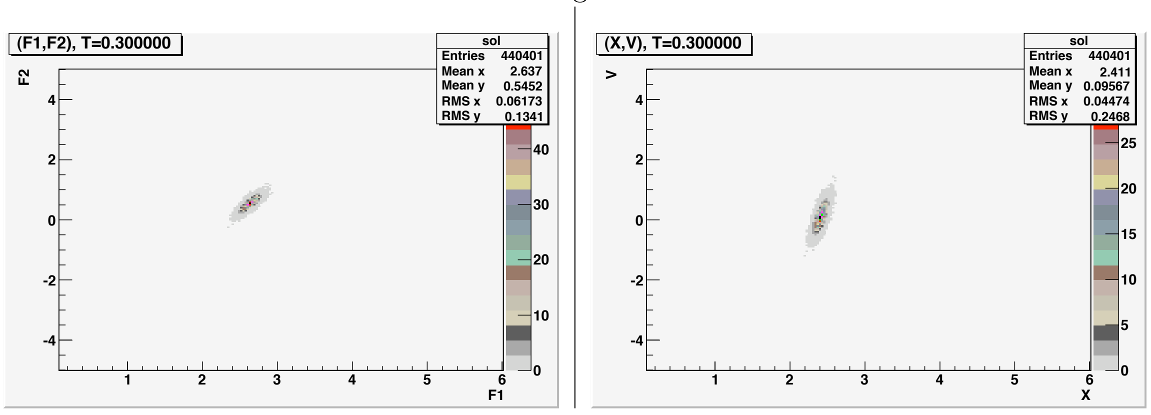


Figure 3.13: Distributions for (f_1, f_2)-coordinates and the transformed version in (x, v)-coordinates.

3.5.3 CPU time

Although the precision of the numerical solutions is of great importance, the CPU time is almost as important. For the PI algorithm, it is obvious that it increases linearly with the number of time steps (or inversely with the size of the time step). It is less clear how it evolves with the step size of the grid. The total computational complexity of PI depends on the choices that are made - for simplicity, assume that there are N points in each of the two dimensions. The additive terms of the CPU time will then be:

1. Initialization of variables is $O(N^2)$.
2. If cubic splines are chosen, the construction of the derivative table in one dimension is $O(4N^2 + 1)$
3. For each point on the grid, one must make one backwards RK4 time step and interpolate. This gives $O(N^2 \cdot (N^2 + 2N + 1))$ with cubic splines or $O(N^2 \cdot 1)$ with linear interpolation.
4. With noise in 1 dimension the integration takes $O(N^3)$ time, with noise in two dimensions it becomes $O(N^4)$.
5. Points 2-4 are repeated $T/\Delta t$ times.

This means that with noise in one dimension and bilinear interpolation PI has $O(N^3)$ CPU time, with bicubic splines it is $O(N^4)$. However, this is a worst-case scenario for the bilinear case, since one only integrates within a subset of the grid. For the bicubic case, $O(N^4)$ is not a worst-case scenario, since the spline construction phase includes iterations over the entire array of length N. Finally, with two-dimensional noise, PI becomes $O(N^4)$ (worst-case) because of the numerical integration, independently of the interpolation routine. However, the asymptotic behavior of the algorithm cannot be read too literally, since the technical specifications of the computer are highly relevant. With limited random-access memory available, reading and writing to the two-dimensional arrays can become very slow once they grow past a certain critical level, up to the fatal level where one gets a segmentation fault.

Figure 3.14 displays the results for the 2D harmonic oscillator with noise in 1 dimension. It is seen that with linear interpolation, the proportions of CPU time for the different components of the PI algorithm remain about the same when the grid size increases. However, the CPU time required by the bicubic splines explodes as the size of the grid grows, as seen in figure 3.15. Running the PI algorithm on the centralized batch job server at CERN with a grid of 1001×1001 points and a time step of $\Delta t = 0.001$ was infeasible, as the submitted job did not finish in one week, which is the maximum time allowed for this type of jobs. Since part of the objective with path integration is computational efficiency, one must resort to other methods, e.g. rectangular splines.

Although the numerical integration only represents a modest proportion of the total CPU time for the harmonic oscillator with noise in one dimension, figure 3.16 shows that it becomes completely dominant with linear interpolation and a model with noise in two

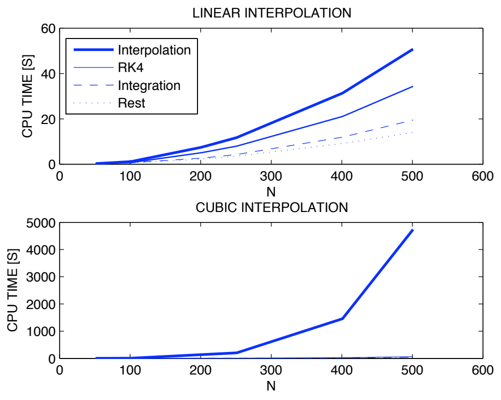


Figure 3.14: CPU time for the different parts of the algorithm - 2D model with noise in one dimension. Each line shows the cumulative time of the computation including the relevant component, i.e. the top line is the total CPU time for each value of N.

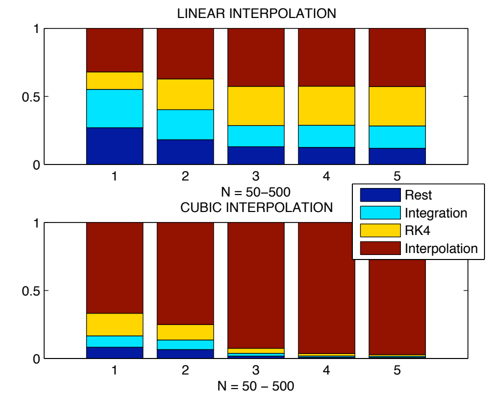


Figure 3.15: Fraction of CPU time spent on the different components - 2D model with noise in one dimension. The values stem from figure 3.14.

Δt	0.1	0.01	0.005	0.001
$\Delta x = 0.2,\ \|\|e\|\|_\infty$	0.103	0.367	0.453	0.647
$\Delta x = 0.1,\ \|\|e\|\|_\infty$	0.032	0.117	0.166	0.464
$\Delta x = 0.04,\ \|\|e\|\|_\infty$	0.090	0.022	0.042	0.070
$\Delta x = 0.02,\ \|\|e\|\|_\infty$	0.101	0.016	0.011	0.028
$\Delta x = 0.01,\ \|\|e\|\|_\infty$	0.101	$3.24e-3$	$7.96e-3$	0.012
$\Delta x = 0.005,\ \|\|e\|\|_\infty$	0.103	$6.76e-3$	$1.63e-3$	$3.63e-3$

Table 3.8: Error for the PI by FFT scheme: Harmonic oscillator with Taylor 2.0 variance in dimension 2, bilinear interpolation.

dimensions. The next section will therefore treat the implementation of path integration by FFT, where the FFT algorithm in lieu of the integration should save considerable amounts of CPU time.

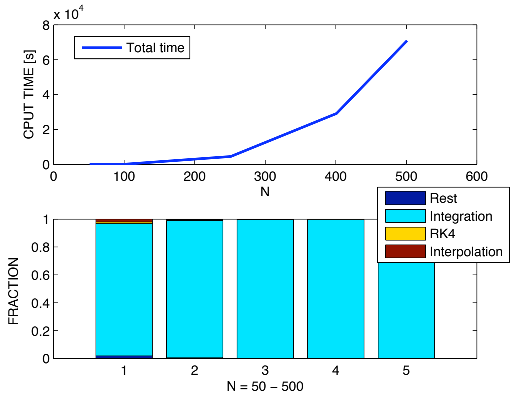


Figure 3.16: PI: CPU time and fraction spent on the different components - 2D model with noise in two dimensions, bilinear interpolation.

3.6 Numerical Solution by Path Integration by FFT

As discussed previously, the implementation of PI by FFT is very similar to regular PI with the exception that a convolution by multiplication in frequency space is performed instead of an integration over the whole subspace.

3.6.1 Harmonic Oscillator

As seen in table 3.8, the numerical solution with PI by FFT is almost identical to the one obtained with regular PI in table 3.6. Obviously, the most important aspect of the algorithm is the computational efficiency, which will be treated in the following section.

3.6.2 CPU time

The FFT algorithm in two dimensions is $O(N^2 \log(N))$, which is better than integrating over a large subset of the grid for every single point, as with regular PI. Convoluting with the gaussian by a pointwise multiplication in Fourier space and choosing an interpolation method that is $O(N^2 \log(N))$, the algorithm becomes much more efficient than PI. Figure 3.17 shows the results with bilinear interpolation - for a grid of 500×500 points, the solution takes about 6.5 minutes, whereas for PI the corresponding time is approximately 22 hours. The FFT routine is seen to dominate the total CPU time as the grid grows. However, since the difference from the other components of the algorithm is only a factor of $\log(N)$, the proportion does not explode as in the previous cases.

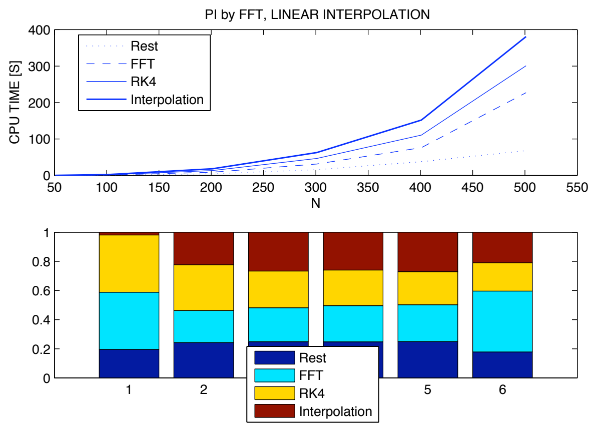


Figure 3.17: PI by FFT: CPU time and fraction spent on the different components - 2D model with noise in two dimensions, bilinear interpolation.

3.7 Numerical Solution by Finite Differences

Writing the SDE in its other form, the two-dimensional Fokker-Planck equation, permits a more common type of numerical solution: A finite difference scheme is presented in reference [17]. This equation can be written in the form of one flux in x and one in v, which permits the use of an operator splitting method [11].

$$\frac{\partial \rho}{\partial t} = \frac{\partial (F_1)}{\partial x} + \frac{\partial (F_2)}{\partial v} \tag{3.18}$$

A tridiagonal scheme is proposed by Zorzano et al. [17]: The first step evaluates the x derivative implicitly and the second step evaluates the v derivative, also implicitly:

$$(1)\quad \frac{\rho_{i,j}^{n+1/2}-\rho_{i,j}^{n}}{\Delta t}=\frac{F_{i,j+1/2}^{n+1/2}-F_{i,j-1/2}^{n+1/2}}{\Delta v} \tag{3.19}$$

$$(2)\quad F_{i,j+1/2}=D\frac{\rho_{i,j+1}-\rho_{i,j}}{\Delta v}+(a_1(x)+a_2(x,v+\Delta v))\frac{\rho_{i,j+1}+\rho_{i,j}}{2} \tag{3.20}$$

$$(3)\quad \frac{\rho_{i,j}^{n+1/2}-\rho_{i,j}^{n+1/2}}{\Delta t}=-v\frac{\rho_{i+1,j}^{n+1}-\rho_{i-1,j}^{n+1}}{2\Delta x} \tag{3.21}$$

Using classical von Neumann stability analysis, one assumes the difference equations are all of the form $\rho_j^n=\xi^n e^{ikj\Delta x}$ [11]. In [17], the amplification factors are given directly, but the steps will be shown below. Substituting 3.20 into 3.19, one obtains for the first step

$$\begin{aligned}
\rho_{i,j}^{n+1/2} &= \rho_{i,j}^{n}+\frac{\Delta t}{\Delta v}(D\frac{\rho_{i,j+1}^{n+1/2}-2\rho_{i,j}^{n+1/2}+\rho_{i,j-1}^{n+1/2}}{\Delta v}+a_{x,v+\Delta v/2}\frac{\rho_{i,j+1}^{n+1/2}-\rho_{i,j-1}^{n+1/2}}{2})\\
1 &= \xi^{1/2}(1-\frac{D\Delta t}{\Delta v^2}(e^{ik\Delta v}-2+e^{-ik\Delta v})-\frac{\Delta t}{2\Delta v}a_{x,v+\Delta v/2}(e^{ik\Delta v}-e^{-ik\Delta v}))\\
1 &= \xi^{1/2}(1+\frac{2D\Delta t}{\Delta v^2}(1-\cos k\Delta v)-i\frac{\Delta t}{\Delta v}a_{x,v+\Delta v/2}\sin k\Delta v)\\
\xi^{1/2} &= \frac{1}{1+\frac{4D\Delta t}{\Delta v^2}\sin^2\frac{k\Delta v}{2}-i\frac{\Delta t}{\Delta v}a_{x,v+\Delta v/2}\sin k\Delta v}
\end{aligned}$$

Thus the amplification factor found here differs slightly from what is found in [17] (by a factor of 4 in the second term of the denominator). However the first step is still unconditionally stable, since $|\xi|\leq 1$ for all k and Δv. For the second step,

$$\begin{aligned}
\rho^{n+1} &= \rho^{n+1/2}-\frac{\Delta t\cdot v}{2\Delta x}(\rho_{i+1,j}^{n+1}-\rho_{i-1,j}^{n+1})\\
\xi^{1/2} &= 1-\frac{\Delta t\cdot v}{2\Delta x}(e^{ik\Delta x}-e^{-ik\Delta x})\\
\xi^{1/2} &= \frac{1}{1+i\frac{v\Delta t}{\Delta x}\sin k\Delta x}
\end{aligned}$$

Hence $|\xi|\leq 1$ for all k and Δx, and the second step is unconditionally stable.

$$g_1=\frac{1}{1+A\sin^2\frac{k\Delta v}{2}-ia_{x,v+\Delta v/2}\frac{\Delta t}{\Delta v}\sin k\Delta v} \tag{3.22}$$

and

$$g_2=\frac{1}{1+i\frac{v\Delta t}{\Delta v}\sin k\Delta x} \tag{3.23}$$

with $A=(4\Delta t/\Delta v)^2 D$ and $a_{x,v-\Delta v/2}\approx a_{x,v+\Delta v/2}$. Therefore, $|g_i|\leq 1$ for $i=1,2$ and the scheme is unconditionally stable.

Δt	10^{-1}	10^{-2}	10^{-3}
$\|\|e\|\|_\infty$, $\Delta x = 0.5$	0.333	0.318	0.316
$\|\|e\|\|_\infty$, $\Delta x = 0.1$	0.077	0.018	0.023
$\|\|e\|\|_\infty$, $\Delta x = 0.05$	0.078	$9.1e-3$	$4.9e-3$
$\|\|e\|\|_\infty$, $\Delta x = 0.01$	0.078	$8.9e-3$	$8.66e-4$

Table 3.9: Errors for the finite difference scheme for the harmonic oscillator for different grid sizes and time step sizes. The grid step size is equal in the two dimensions.

The implementation is done in Matlab, because of the built-in linear algebra packages that let the user handle sparse matrices and vectors with ease. Comparisons of the CPU time with other schemes implemented in C++ should be realistic, since Matlab is highly optimized as long as one uses the built-in functions such as $A\backslash b$, which solves the matrix equation $Ax = b$, and avoids iteration loops as much as possible. In contrast with the PI algorithms and the simulation schemes, the implementation of the finite difference scheme is rather straightforward since the linear algebra library is readily available.

3.7.1 Harmonic Oscillator

The boundary condition for the probability distribution is simply set to zero outside the grid points. Generally, it is impossible to start from a Dirac delta initial distribution, so one must resort to other solutions. Two possibilities are to either use a normal distribution centered on the initial space point or to make a large number of simulations for a reasonable number of the earliest time steps. Since the exact solution is available for the harmonic oscillator, this will be used in order to obtain a benchmark for the accuracy of the algorithm. Zorzano et al take the exact distribution at time $t = 0.95$ for a strongly damped oscillator with high diffusion; $K = 1, \gamma = 2.1, \sigma = 0.8$. They use as error norm

$$||e|| = \sqrt{\frac{1}{N}\sum\sum(\rho_{num}(i,j) - \rho_{ex}(i,j))^2}$$

where N is the total number of grid points. For the example with these parameters, a grid of 80×80 points, $\Delta t = \pi/1000$ and $T = 3$, they report an error norm of $5.3e-4$. However, this error norm is an average over the whole grid, where it is mostly very close to zero, and does not take into consideration the shape of the numerical solution compared to the exact one. Using the same parameters, but a grid size of 101×101 and $\Delta t = 0.005$, the distribution in figure 3.18 is obtained on a 2.4 GHz Intel processor with 2 GB RAM. This gives $||e|| = 2.9e-4$ and $||e||_\infty = 0.0030$. Figure 3.19 shows the difference between the numerical solution and the exact solution using a time step $\Delta t = 0.1$. The finite difference scheme is seen to introduce extra numerical diffusion, which becomes too dominant for time steps of this size. With $\Delta t = 0.1$, $||e||_\infty = 0.036$, which is clearly unacceptable for a probability density. However, $||e|| = 0.0036$, which is not really informative. In the following, only the max error norm will be used.

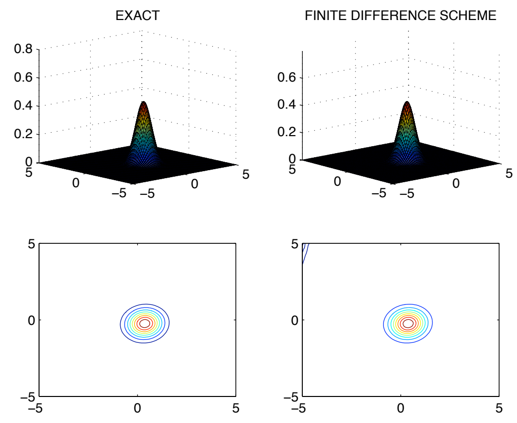


Figure 3.18: Finite difference scheme versus the exact solution for the harmonic oscillator. $K = 1, \gamma = 2.1, \sigma = 0.8, T = 3, \Delta t = .005$

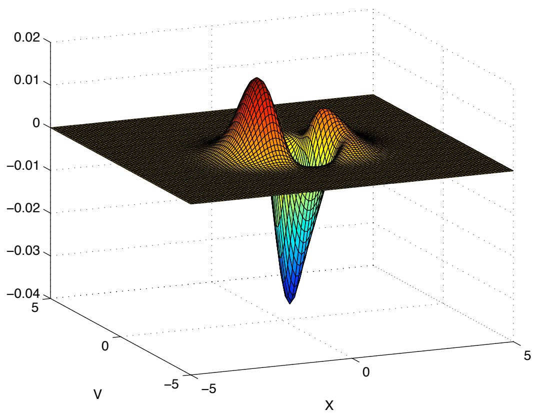


Figure 3.19: Plot of $\rho_{num} - \rho_{ex}$ for the finite difference scheme using the same parameters as in figure 3.18, but a larger time step $\Delta t = 0.1$. The systematic difference that is seen is caused by numerical diffusion from the implicit scheme

Table 3.9 shows how the finite difference scheme converges for grid size and time steps. The CPU time grows linearly with decreasing Δt, but as in the previous sections, the behaviour for smaller Δx depends on the technical specifications of the machine. The numerical results are represented in figure 3.20, which appear to demonstrate:

- Order 1.0 convergence in Δt as long as Δx is sufficiently small. For a given Δx, there is a minimum error level - reducing Δt below this does not give any extra accuracy, but does not cause any harm either, as was the case for PI.
- Order 2.0 convergence in Δx as long as Δt is sufficiently small, until the error reaches the minimum level.

The finite difference scheme is not implemented here for the stochastic Duffing equation, since the physical results have already been illustrated by simulation, and there is no exact solution available. Instead, a comparison of the efficiency of all four algorithms would be of interest.

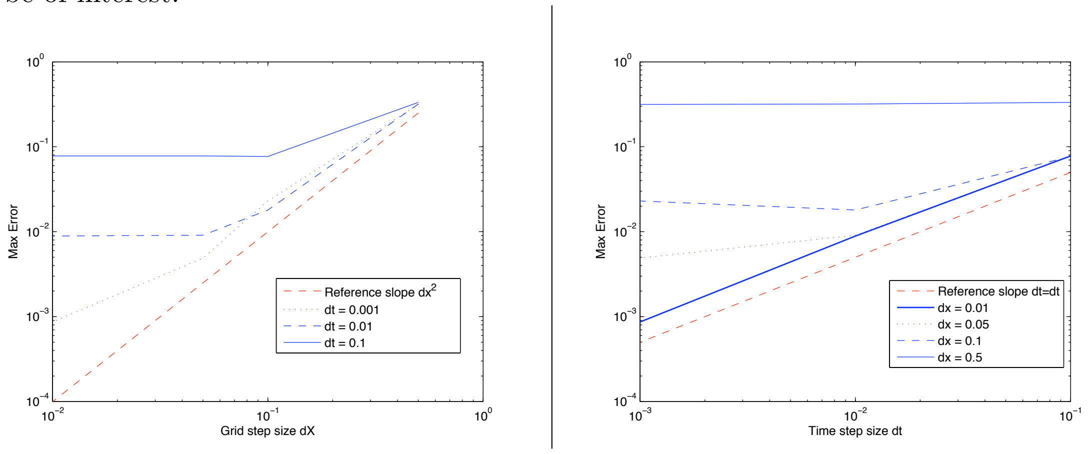


Figure 3.20: Graphical representation of the convergence in Δx (left) and Δt (right) for the finite difference scheme. The values are the ones given in table 3.9

.

3.8 Global Comparison of Accuracy and CPU time

In figure 3.21, an attempt has been made to plot the best accuracy available at a given CPU time T for each of the four algorithms described previously. Since there are many possible choices of step sizes and subcomponents in the algorithms, the results are not perfectly straight lines but rather a picture of the efficiency of the respective numerical solvers. For the model with two-dimensional noise, the accuracy at every combination $(\Delta x, \Delta t)$ has been assumed to be the same as with one-dimensional noise, and only the CPU time has been measured. It is seen that the finite difference scheme provides the best results, but PI with bicubic splines is close for the one-dimensional noise model. PI by FFT with bicubic splines is not plotted since the results are almost exactly the same as with PI - anyway the interpolation routine is by far the most CPU intensive part of the algorithm, as seen in figure 3.15. With noise in two dimensions, the PI by FFT algorithm is clearly a better choice than regular PI, and this effect will become even more pronounced with more dimensions with noise.

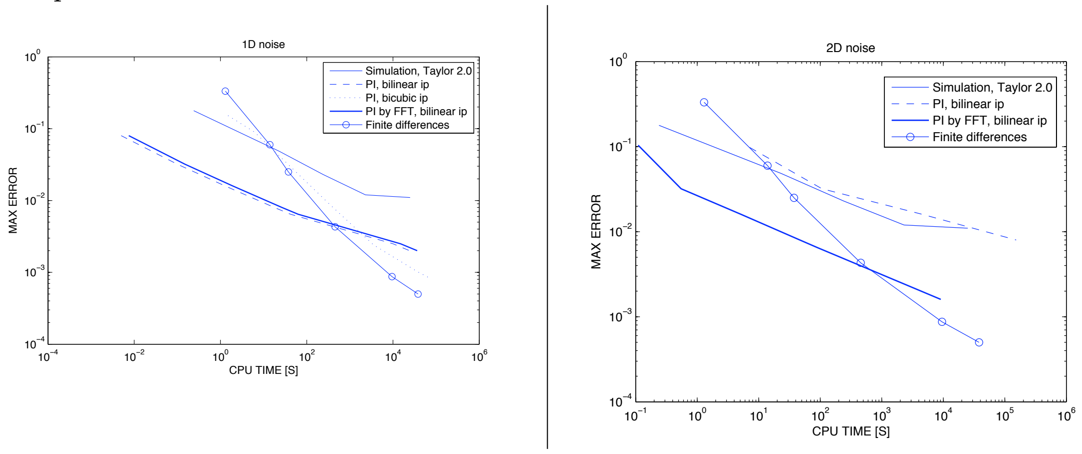


Figure 3.21: Comparison of the convergence of the different algorithms against CPU time: A model with noise in one dimension (left) and noise in two dimensions (right). The accuracy for the two-dimensional noise model is not known and is assumed to be of the same order as with 1-dimensional noise.

3.9 Summary

The Fokker-Planck equation provides an opportunity for benchmarking the path integration algorithms against more traditional numerical solutions, thanks to the exact solution of a class of equations in Chandrasekhars paper [14]. The source of this exact solution had to be consulted since there was shown to be a factor missing in Zorzano's paper. The finite difference scheme with operator splitting described by Zorzano et al. [17] is an example of a classical way to solve the Langevin equation in physics - but this specific scheme only works with SDEs that fit into the format described in their paper. They also tested a finite element based numerical solution, which gave results almost identical (a bit slower) to the finite difference scheme. Path integration proves to be a relatively efficient numerical solution of the SDE, despite some inherent difficulties in the implementation. With splines or interpolation methods that are $O(N \log N)$ in one dimension, PI by FFT is a very efficient algorithm for models that incorporate multiple sources of noise. Simulation has the competitive advantage of being easy to implement, but the $O(\sqrt{N_{rep}})$ convergence in the number of path realizations proves to be a major obstacle in obtaining highly accurate solutions.

Chapter 4

Stochastic Differential Equations with Fat Tail Distributions

In the previous sections, all noise terms in the equations have been assumed to be Gaussian. Unfortunately, not all physical or social phenomena are well described by normal distributions. Although this simplifying assumption gives the relevant equations nice properties, e.g. the sum of normal distributions is also a normal distribution, it may create models with serious flaws. At least part of the cause of the worldwide financial crisis of 2008 was the introduction and reliance on equations from physics into mathematical finance, a field that has been developed very fast during the last decades. Especially correlations between financial assets and the tails of the probability distributions have been seen to be unrealistic in most models during periods of high fluctuations, although they may work well in 'normal' circumstances. Since Fischer Black and Myron Scholes published their famous 1973 paper 'The Pricing of Options and Corporate Liabilities', which was further developed by Robert Merton, stochastic differential equations have received a lot of attention. In this model, the share price evolves as in equation 1.19: it follows a geometric Brownian motion, where the variance is proportional to the share price S:[19]:

$$\mathrm{d}S = \mu S \,\mathrm{d}t + \sigma S \,\mathrm{d}W \tag{4.1}$$

By using Itô's lemma, one gets a partial differential equation for the value of a portfolio $\Pi = V - \Delta S$, where V is the value of the option and Δ is the number of shares that is held. By assuming that trading can be done in continuous time, it can be shown that Δ can be modified at each time step so that the value of the portfolio does not change. Since the value of the option is given at the boundary by $V(T) = \max(0, S - E)$, where E is the strike price of the option, the value can be determined by solution of the partial differential equation. The Black-Scholes option pricing model implies that only the volatility σ and the time determines the correct price of the derivative. This is illustrated respectively in figure 4.1 and 4.2: since the possible loss is limited for the buyer of an option, the broader the probability distribution the higher the price will be. For the same reason, with time the chance of big movements upwards increases.

Therefore, higher volatilities and more time until the strike date increase the option price. Although the framework was originally intended for option prices, it will work for any derivative that depends only on time and the price of an underlying commodity that is subject to the same assumptions as the share price S. For example, options on electricity prices (traded on the Nordpool exchange in Scandinavia) or the oil spot price can be priced using the same methods. A large proportion of the financial mathematics literature since 1973 are variations over this theme: modeling the volatility implicitly as SDEs, pricing exotic derivatives such as Asian options, including transaction costs etc.

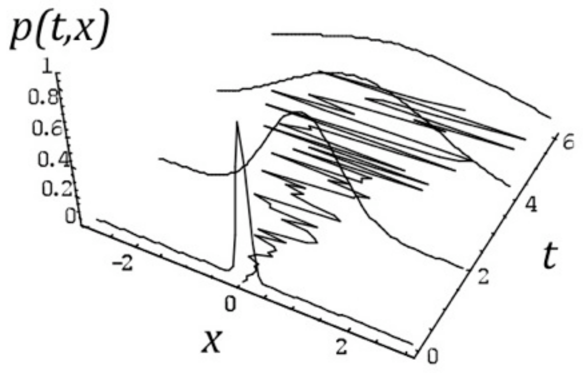


Figure 4.1: Probability distribution of x with volatility as a function of t.

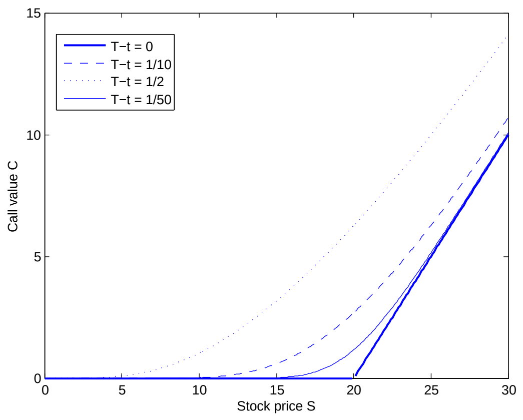


Figure 4.2: Evolution of option value with time and as a function of S, with a strike price $E = 20$. The value is seen to be higher when the time difference is large.

However, there are additional assumptions that must be satisfied: for example, the relevant interest rate must be a known function of time and the random movements must follow a Brownian motion. There is broad agreement that especially the latter is an unreasonable assumption [19]. In his book 'The Black Swan' [20], Nassim Taleb, an influential risk manager and former trader on Wall Street, gives the normal distribution much of the blame for the financial turmoil of 2008-2009. The main argument is that the models are often used where they should not be: for instance, it is well known that stock markets regularly exhibit large jumps, for which the normal distribution gives a much too low probability. Figure 4.3 shows the distribution of the daily fractional increments of the share price of SAS, a Scandinavian aviation company, over the last 5 years. The normal distribution curve seemingly describes the data well, but there are some outliers that should only occur a few times in the lifetime of the universe: the probability of a daily increase of more than 40 %, which has occured once during the last five years, is much smaller than machine precision on a laptop computer. From his empirical evidence, Taleb draws the conclusion that stock market movements are of a fractal nature, similarly to many other social phenomena. For example, sizes of cities and human wealth can be described by a fractal geometry. Benoit Mandelbrot, in his 1983 book The Fractal Geometry of Nature, describes the Levy flight as a better model of financial and economic time series. The same histogram described by a Cauchy distribution is shown in figure 4.4. The latter belongs to another family of probability distributions that has 'fatter tails' than the normal distribution. This motivates a discussion on SDEs with Lévy alpha-stable noise.

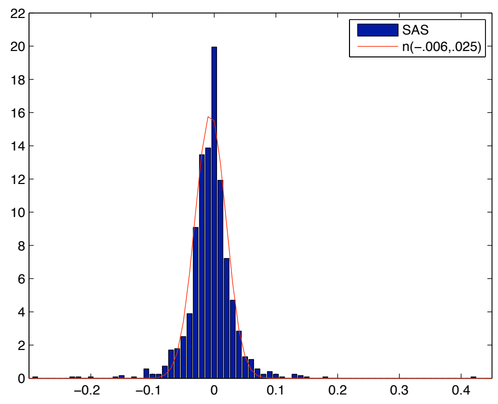


Figure 4.3: Changes in the SAS share price, with a normal distribution

4.1 General Form of Lévy Alpha-Stable Distributions

The characteristic function is an alternative way of describing a random variable. It also completely determines the properties of the probability distribution of the random variable X. Given a density function $f()$, the characteristic function can be found from

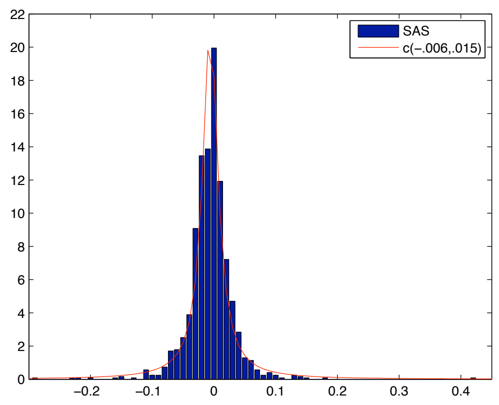


Figure 4.4: Changes in the SAS share price, with a Cauchy distribution

the relation:

$$\phi_X(t) = \mathbb{E}[e^{itX}]$$

and as long as ϕ_X is integrable, the probability density can be found by an inverse Fourier transform of the characteristic function:

$$f_X(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-itx}\phi_X(t)\,\mathrm{d}t$$

It also has some neat properties: If $X_1, \ldots, X_n$ are independent random variables and $a_1, \ldots, a_n$ are some arbitrary constants, then the characteristic function for the linear combination of X_i's is [21]

$$\phi_{a_1X_1+\ldots+a_nX_n}(t) = \phi_{X_1}(a_1t)\cdot\ \ldots\ \cdot\phi_{X_n}(a_nt)$$

A random variable is said to belong to a stable distribution if a linear combination of two independent copies X_1 and X_2 of the random variable X have the same distribution as $cX + d$, where c and d are some scale parameters. Stated in terms of the characteristic function, this is a four-parameter family described by μ (location), σ (scale), α (tail heaviness) and β (skewness/asymmetry) $\in [-1, 1]$:

$$\phi(t) = e^{i\mu t-\sigma^\alpha|t|(1-i\beta\mathrm{sgn}(t)\tan(\pi\alpha/2))} \quad \forall \quad \alpha \neq 1$$
$$\phi(t) = e^{i\mu t-\sigma^\alpha|t|(1+i\frac{2}{\pi}\beta\mathrm{sgn}(t)\log(t))} \quad \forall \quad \alpha = 1$$

For $\beta = 0$, the distribution is symmetric, and it is referred to as a Lévy symmetric α-stable distribution. In general, the probability density function is not analytic, and the normal definition of skewness,

$$\beta = \mathbb{E}(\frac{(X-\mu)^3}{\sigma})$$

does not apply for $\alpha < 2$ because the distribution does not have second order moments. The characteristic function of the generic normal distribution is

$$\phi_X(t) = e^{i\mu t - \frac{1}{2}\sigma^2 t^2}$$

so it is easily seen that with $\alpha = 2$, the Lévy symmetric α-stable distribution has characteristic function

$$\phi_X(t) = e^{i\mu t - \sigma^2 t^2}$$

i.e. a normal distribution with variance $2\sigma^2$. With $\alpha < 2$ and $\mu = 0$, there is only one trivial class of distributions, namely the Cauchy (or usually Lorentz in physics) distribution, with $\alpha = 1$. With $\beta = 1, \alpha = 1$, one obtains a Landau distribution, which is another important distribution in physics. Since a nonzero μ can be incorporated in the drift term of the SDE and a skewed noise distribution can usually be transformed to a symmetric one, a zero-mean, symmetric, α-stable distribution is general enough for the purposes in this thesis. It can be shown that any stable distribution has a continuous and smooth probability density function [21], but for $\alpha \neq 1$, it must be obtained by simulation [22]. These distributions are also scalable, so any distribution characterized by σ, μ can be transformed to a normalized one with $\sigma = 1, \mu = 0$. Naess and Skaug adopt the notation $X \sim S_\alpha(\sigma, \beta, \mu)$ and this will be put to use here as well. Plots of the characteristic function for different values of α are shown in figure 4.5. As $\alpha \to \infty$, the characteristic function approaches the following box function:

$$\begin{aligned} &1 \quad \forall \quad |t| < 1 \\ &0 \quad \forall \quad |t| > 1 \end{aligned}$$

However, the inverse Fourier transform of this function is $\frac{\sin(x)}{x}$, which is not nonnegative, and thus cannot be a probability density. In fact, for all $\alpha > 2$ the inverse Fourier transform assumes negative values, so it is confined to the interval $(0, 2]$ for the class of stable distributions.

The Cauchy distribution with $\mu = 0$ has the following analytic expression:

$$f(x) = \frac{1}{\pi\sigma} \frac{1}{1 + \frac{x^2}{\sigma^2}} \tag{4.2}$$

Although the notations α and β are used for the same parameters almost everywhere in the literature, μ and σ are different almost everywhere. This may be due to the fact that they should not be misinterpreted as the mean and the standard deviation respectively. The Cauchy distribution does not have a finite mean or variance, a fact that can be directly inferred from the probability density function or from the characteristic function.

4.2 Generalized Fokker-Planck equation and Exact Solutions

The Langevin equation driven by Lévy noise yields the so-called fractional Fokker-Planck equation. Denisov et al. show in a 2009 paper [15] how to derive this equation when the

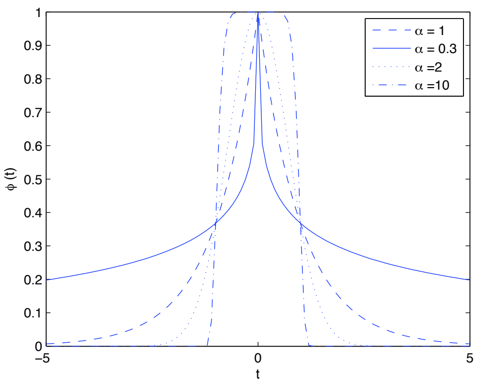


Figure 4.5: Plots of the stable characteristic function $\phi(t) = e^{-|t|^\alpha}$ with $\mu = \beta = 0$ and $\sigma = 1$ for different values of α.

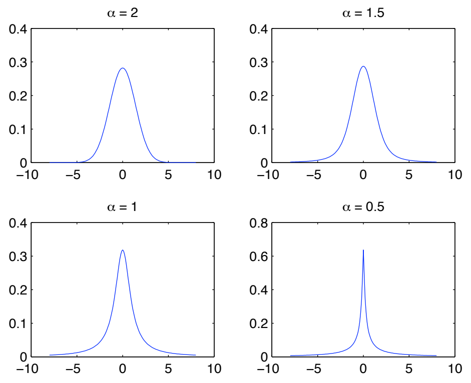


Figure 4.6: Probability distributions with $\mu = \beta = 0$ and $\sigma = 1$ for different values of α. The top left distribution is a Gaussian and in the bottom left corner is the Cauchy distribution.

$\mathbb{P}(X > x)$	1	3	10
$\alpha = 2$	0.159	0.0013	0.000
$\alpha = 1.5$	0.243	0.052	0.007
$\alpha = 1$	0.250	0.102	0.032
$\alpha = 0.5$	0.249	0.161	0.089

Table 4.1: Comparison of tail probabilities for different stable distributions.

Langevin equation is driven by multiplicative alpha-stable Lévy noise, and how the exact solution can be expressed in terms of the inverse Fourier transform of the characteristic function. The SDE is still in familiar form:

$$\mathrm{d}X_t = a(x,t)\,\mathrm{d}t + b(x,t)\,\mathrm{d}\eta(t)$$

However here, $\mathrm{d}\eta(t)$ is no longer necessarily a Brownian motion, but follows the more general alpha-stable Lévy distribution. When $b(x,t) = 1$ and the noise is purely additive, the generalized Fokker-Planck equation becomes:

$$\frac{\partial}{\partial t}P(x,t) = -\frac{\partial}{\partial x}a(x,t)P(x,t) + \mathcal{F}^{-1}[\mathcal{F}(P(x,t)\log(S_k)]$$

where the characteristic function of the noise generating process at dimensionless time $t = 1$ is

$$\begin{aligned} S_k &= \mathbb{E}[e^{-ik\eta(1)}] \\ \eta(1) &= \lim_{\tau\to 0}\sum_{j=0}^{1/\tau-1}\Delta\eta(j\Delta t). \end{aligned}$$

For example for a Gaussian transition probability density, one has $S_k = e^{-Dk^2}$ and hence $\mathcal{F}^{-1}[\mathcal{F}(P(x,t))Dk^2] = -D\frac{\partial^2 P(x,t)}{\partial x^2}$, which clearly gives the same Fokker-Planck equation that was derived in the previous chapter. For Lévy stable noise, the characteristic function was given in equations 4.2 and 4.2. In terms of the Riesz derivative, defined as

$$\frac{\partial^\alpha}{\partial |x|^\alpha}h(x) = -\mathcal{F}^{-1}[|k|^\alpha h_k]$$

the fractional Fokker-Planck equation that corresponds to all forms of Lévy stable noise becomes [15]

$$\frac{\partial}{\partial t}P(x,t) = -\frac{\partial}{\partial x}a(x,t)P(x,t)+\sigma\frac{\partial^\alpha}{\partial|x|^\alpha}b^\alpha(x,t)P(x,t)+\sigma\beta\tan\frac{\pi\alpha}{2}\frac{\partial^{\alpha-1}}{\partial|x|^{\alpha-1}}b^\alpha(x,t)P(x,t) \tag{4.3}$$

With $\alpha = 2$, which corresponds to Gaussian noise, the equation becomes the ordinary Fokker-Planck equation in one dimension. For $\alpha < 2$, there are only some special cases that can be given in closed form. Exact solutions exist for simple examples such as the harmonic oscillator, but they are rather involved and not in explicit form.

4.3 Simulation Scheme and Solution by PI by FFT

Now, write the general form of the stochastic jump process as

$$\mathrm{d}X_t = a(X_t)\,\mathrm{d}t + b(X_t)\,\mathrm{d}L_t^\alpha \tag{4.4}$$

where X_t is the n-dimensional state space process and L_t^α is an α-stable $S_\alpha(1,0,0)$ process with $\alpha \in (0,2]$. In analogy with Gaussian white noise, the time derivative of the Lévy process will be referred to as Lévy white noise. Interestingly, it can be shown that the sum of a compound Poisson process and a Brownian motion with or without drift [23] [24] is a good approximation of a Lévy process. The compound Poisson process C is defined as

$$C(t) = \sum_{k=1}^{N(t)} Y_k \tag{4.5}$$

where $N(t)$ is a Poisson process with intensity λ and Y_i are independent and identically distributed (iid) random variables. A random walk sequence approximating L_α is then

$$L_{\alpha,a}(t) = \gamma(\alpha,a)B(t) + C^{\alpha,a}(t) \tag{4.6}$$

where $\gamma(\alpha,a)B(t)$ is a scaled Brownian motion, with the scale given by [23]

$$\begin{aligned}
\gamma^2(\alpha,a) &= \int_{-a}^{a} y^2 \lambda_L \,\mathrm{d}y = \frac{\alpha}{2-\alpha} c_\alpha a^{2-\alpha} \\
c_\alpha &= \frac{1-\alpha}{\Gamma(2-\alpha)\cos(\pi\alpha/2)}, \quad \forall \quad \alpha \neq 1 \\
c_\alpha &= \frac{2}{\pi}, \quad \text{if} \quad \alpha = 1
\end{aligned}$$

The intensity of the Poisson process is $\lambda^{\alpha,a} = c_\alpha / a^\alpha$. This also explains why models with Levy noise are often referred to as jump-diffusion models - the noise consists of one continuous diffusion component and one pure jump component. Using some identities on probability measures and the Chebyshev inequality, it can be shown that this scheme converges weakly to the exact solution $X(t)$, so that statistics of functionals of X can be approximated by functionals of the numerical solution. For further details, see reference [23]. Using this simulation scheme with $\alpha = 1$ (Cauchy noise), three sample paths are generated for the two SDEs

$$\mathrm{d}X_t = 10\,\mathrm{d}W_t \tag{4.7}$$

$$\mathrm{d}X_t = \mathrm{d}L_t^\alpha \tag{4.8}$$

respectively in figure 4.7. If one compares these example paths with the time series of almost any financial asset, it is quite clear that the Lévy noise more closely resembles its actual behaviour. Figure 4.8 shows the evolution of the share price of SAS AB over a time span of one week. Discontinuous jumps are an intrinsic property of share prices, since they are traded in discrete time and in discrete batches. This effect is not included in a model that incorporates only Brownian motion.

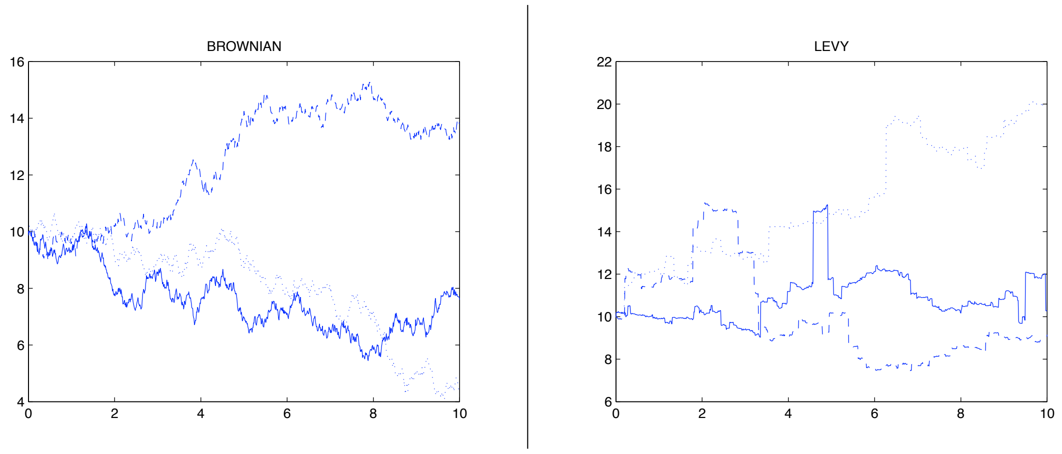


Figure 4.7: Three sample paths for a pure Brownian walk (left) and a pure Levy walk (right).

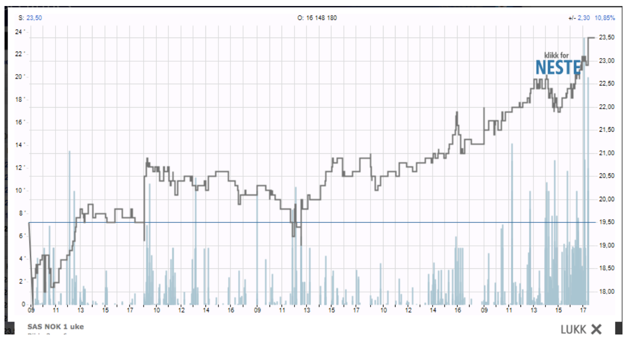


Figure 4.8: Evolution of the SAS share price.

Implementing numerical PI or PI by FFT with Lévy noise is not problematic as long as the noise distribution can be generated analytically or by simulation. Using the Cauchy distribution, the probability distribution is given explicitly, and the algorithm is the same as described in the previous chapter. However, since the Cauchy distribution does not decay as fast as the normal distribution in the tails, the range of the grid must be much larger. Since the FFT is being used, the distribution must be very close to zero in the tails. The convolution is implemented with zero padding of the signal (to get a vector of length 2N) in order to avoid numerical instabilities. Also, if the probability distribution does not integrate to 1 after each time iteration, the numerical solution decays to 0 very fast. This must be avoided by including as much as possible of the probability distribution where the value exceeds machine precision. Integrating the two SDEs 4.7 and 4.8 up to $T = 2$, a grid from -10 to 10 is typically sufficient for the Brownian motion, while for the Levy noise one needs grid points in the range $[-1000, 1000]$. Figure 4.9 displays the numerical simulation and PI by FFT results of solving equation 4.8 up to time 2. Superposing the two curves made it impossible to distinguish them, so they were plotted separately - this seems to confirm that the two methods yield the same results for all practical purposes.

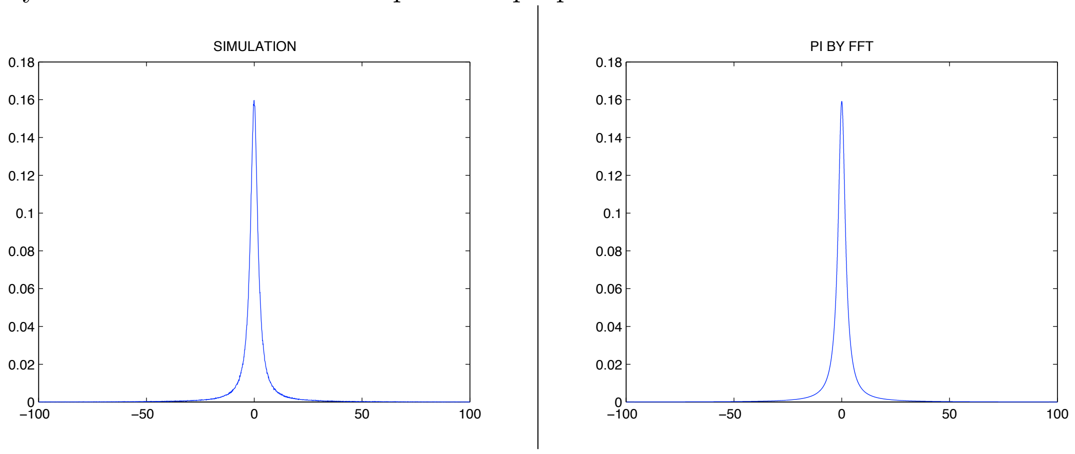


Figure 4.9: Numerical solution of equation 4.8 by simulation and PI by FFT, T=2

4.4 The Normal Inverse Gaussian Process

Another continuous probability distribution that has recently gained popularity in the financial mathematics literature is the Normal Inverse Gaussian (NIG). An inverse Gaussian distribution follows the distribution of the time a Brownian motion with positive drift takes to reach a certain level - it is not in any way the inverse of a Gaussian distribution, so the name can cause some confusion. It has an analytic probability distribution with support on $(0, \infty)$ that takes the form

$$f(x, \nu, \lambda) = \sqrt{\frac{\lambda}{2\pi x^3}} e^{-\lambda(x-\nu)^2/2\mu^2 x}$$

where ν is the mean and λ is the scale parameter. The relation with the Normal Inverse Gaussian is

$$Y = \delta + \gamma V + \tau\sqrt{V}X$$

where X follows a standard normal distribution $n(0, 1)$ and V is an inverse Gaussian random variable, independent from X. The resulting distribution Y is a normal distribution with mean $\delta + \gamma V$ and variance $\tau^2 V$. This is called a variance-mean mixture, where the inverse Gaussian is the mixing distribution [25]. Similarly to the alpha-stable

Lévy distributions, this class of probability distributions is described by four parameters - μ, σ, α and β. As before, μ and σ are the location and scale parameters respectively, while α and β describe the tail heaviness and asymmetry of the distribution. The effects of varying the latter two are shown in figure 4.10. The NIG probability distribution becomes

$$f(x,\mu,\sigma,\alpha,\beta) = \frac{\alpha\sigma}{\pi} e^{\sigma\sqrt{\alpha^2-\beta^2}+\beta(x-\mu)} \cdot \frac{K_1(\alpha\sqrt{\sigma^2-(x-\mu)^2})}{\sqrt{\sigma^2+(x-\mu)^2}}$$

where K_1 is the modified Bessel function of the second kind, and in contrast with the alpha-stable distributions, the NIG distribution has finite mean and variance;

$$\begin{aligned}\mathbb{E}(Y) &= \mu+\sigma\frac{\beta}{\gamma}, \quad \gamma=\sqrt{\alpha^2-\beta^2}\\ \mathrm{Var}(Y) &= \sigma\frac{\alpha^2}{\gamma^3}\end{aligned}$$

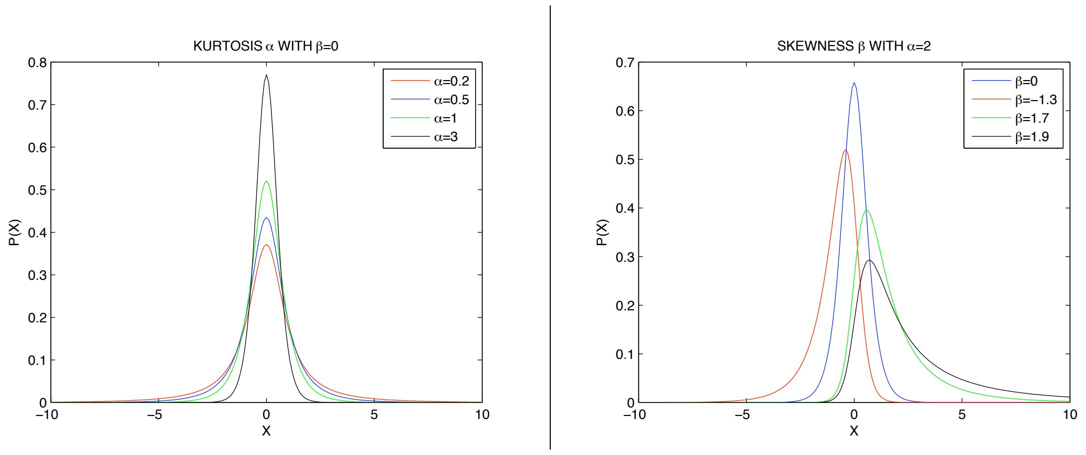


Figure 4.10: NIG distribution: effect of varying the kurtosis α (left) and the skewness β (right) factors. $\mu = 0$ and $\sigma = 1$ in both cases.

It is in fact a subclass of generalized hyperbolic distributions, which include the Student's t-distribution and the hyperbolic distribution. The most common applications cited in the literature are for financial markets. Obviously, the motivation is the greater freedom in fitting the distributions to historical data that the free parameters allow. In reference [26], the four parameters are fitted to the historical data of S&P 500 and OSEBX, the stock market indexes of the New York and Oslo exchanges respectively. Testing the fits with a Pearson χ^2-test yielded a P-value of 0.39 for S&P and 0.64 for OSEBX. The corresponding fits and following tests with a normal distribution both gave P-values inferior to 10^{-10}. Since this test takes into account all given data, the P-value is significantly affected by outliers such as the 40 % increase of the SAS stock price and in this case, the 1987 stock market crash. A normal distribution gives very low probabilities that such events will occur.

Simulation of a NIG process is straightforward, and the PI by FFT algorithm can be implemented as for an equation driven by Lévy noise. In order to further test the practical aspects of these noise distributions, some examples of options pricing will be implemented in the following section.

4.5 Real World Application - Option Pricing

4.5.1 The Black-Scholes Framework

Since PI by FFT is seen to work for more general types of noise than just Brownian motion, it is of interest to study how this fits into the Black-Scholes framework. Estimating the correct price of a stock option is a challenging task, since it involves predictions. However, a good model helps. As stated earlier, the value of the option varies only with time and the price of the underlying asset. At the expiry date, the boundary condition for a European option is given by

$$V(S,T) = \max(0, S-E)$$

where S is the stock price and E is the exercise price. If S is assumed to follow equation 4.1, from the Itô-Doeblin formula 7, the equation for V becomes

$$\mathrm{d}V = \sigma S\frac{\partial V}{\partial S}\,\mathrm{d}W + (\mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t})\,\mathrm{d}t.$$

With the portfolio Π given in the introduction to this chapter, the infinitesimal change in one time step is

$$\mathrm{d}\Pi = \mathrm{d}V - \Delta\,\mathrm{d}S$$

where the number of stocks Δ is held fixed in this time interval. The resulting equation for Π becomes

$$\mathrm{d}\Pi = \sigma S(\frac{\partial V}{\partial S} - \Delta)\,\mathrm{d}W + (\mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + \frac{\partial V}{\partial t} - \mu\Delta S)\,\mathrm{d}t.$$

Since Δ is the number of stocks that are held, the random component can be eliminated by choosing $\Delta = \frac{\partial V}{\partial S}$. In other words, if the drift of the process is different from the risk-neutral one (usually denoted by r), one can buy or sell the call option (depending on whether it is smaller or larger than r) and hedge the portfolio until the expiry date. Furthermore, by assuming a perfect market with a fixed interest rate r, where no 'free money' is available, the evolution of the portfolio must be equal to the return in the bank and one gets a partial differential equation for V:

$$\begin{aligned} r\Delta\Pi\,\mathrm{d}t &= (\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2})\,\mathrm{d}t \quad \rightarrow \\ \frac{\partial V}{\partial t} &+ \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \end{aligned}$$

This is the so-called Black-Scholes partial differential equation, which yields an exact solution, plotted in figure 4.2. An important feature is that the equation is independent of μ, only the volatility parameter σ matters in determining the option price. It is important to realize that the probability density one obtains for the stock price when using the interest rate as the drift is not the one of the real world stock price, but of the risk-neutral one. For a model like 4.1 it is simple enough to model the risk-neutral price - simply replace μ by r. However, for more complicated models it is not as straightforward, and one must resort to theorem 1.2.2. A drawback of the jump-diffusion and NIG models is that under the Girsanov measure transformation all the parameters change. For Brownian motion the volatility is unchanged under this transformation - which makes it much more convenient.

The Black-Scholes model relies on seven assumptions, whereof four of the most important are:

- The stock price follows the Brownian walk model in equation 4.1.
- There are no transaction costs involved with continuously adjusting Δ.
- Trading takes place continuously.
- The risk-free interest rate curve is known and the same for deposits and loans.

All the assumptions are to some extent unrealistic, but the four that are mentioned here are maybe the least plausible. First, if S does not follow the lognormal random walk, but has jumps, it is the Itô-Doeblin formula 1.21 that is relevant. Since there is generally no explicit solution for the sum of the jumps in this equation and the path is no longer continuous, delta hedging is not possible. This means that the NIG and Lévy markets are incomplete - there are many possible pricing formulas. Second, even for an investor with a big portfolio, transaction costs are not negligible, and delta hedging is only possible in a discrete time series. This means that there is some elasticity in the arbitrage argument. Finally, a bank might be able to borrow and lend money at almost the same interest rate, but the curve is certainly not known and is subject to random shocks such as in 2008.

Another strategy for determining the fair derivative price is of course to find the probability distribution of S at the expiry date. This is clearly the easiest way once a model with a numerical solution has been established, such as above.

4.5.2 Pricing a European Call Option

Pricing a plain vanilla option such as a European call is of interest in this context to see

1. how the NIG market model compares to the standard Brownian walk for derivative pricing.
2. how the PI by FFT algorithm performs for both the Brownian walk and for the NIG model.

The Lévy walk has another disadvantage in that the first-order moment and upwards are not finite. This means the price of the derivative would theoretically not be finite. The numerical implementation is on a finite grid so the mean and variance would necessarily be finite here, but it would not make sense to get a numerical value from this. The exact solution of the Black-Scholes partial differential equation with constant drift $\mu = r$ and volatility σ is:

$$V(S,t) = S \cdot N(d_1) - E \cdot e^{-\mu(T-t)} \cdot N(d_2) \tag{4.9}$$

where $N(\,\cdot\,)$ is the standard normal cumulative distribution function, and d_1 and d_2 are:

$$\begin{aligned} d_1 &= \frac{\log(S/E) + (\mu + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \\ d_2 &= \frac{\log(S/E) + (\mu - \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \end{aligned}$$

In evaluating the option price from the probability distributions, we want to find the value of the following integral, discounted using the constant interest rate r

$$V = e^{-r\cdot(T-t)} \int_E^\infty p(S)(S-E)\,\mathrm{d}S$$

Since the noise is multiplicative, for the PI by FFT algorithm one must make the transformation $f(S) = \log(S)$. From the Itô-Doeblin identity one gets:

$$\mathrm{d}f = (\mu - \frac{1}{2}\sigma^2)\,\mathrm{d}t + \sigma\,\mathrm{d}W.$$

In this form, the implementation is straightforward. In order to compare the two distributions, choose a symmetrical NIG distribution ($\beta = 0$) with $\alpha = 1$ and the same variance as a corresponding normal distribution. This implies $\sigma_{NIG} = \sigma^2_{normal}$. Once the distribution p at time $T = 1$ is found, the numerical integration above with varying step sizes is performed as follows:

$$V \approx e^{-rT} \sum_j p_j \cdot (e^{x_j} - e^E)(e^{x_j + \Delta x/2} - e^{x_j - \Delta x/2})$$

The results are shown in table 4.2 and figure 4.11. The NIG walk tends to give a slightly higher price per stock even though the distribution is more acute than the Gaussian version. This is of course due to the heavier tails.

$\sigma_{NIG} = \sigma_N^2$	Exact	PI by FFT with Gaussian	PI by FFT with NIG
0.1^2	0.801	0.801	0.906
0.2^2	1.600	1.600	2.635
0.3^2	2.395	2.395	4.760

Table 4.2: Price of a European call option with $T = 1$, $r = 0.1$, $S_0 = e^3$, $E = e^{3.1}$.

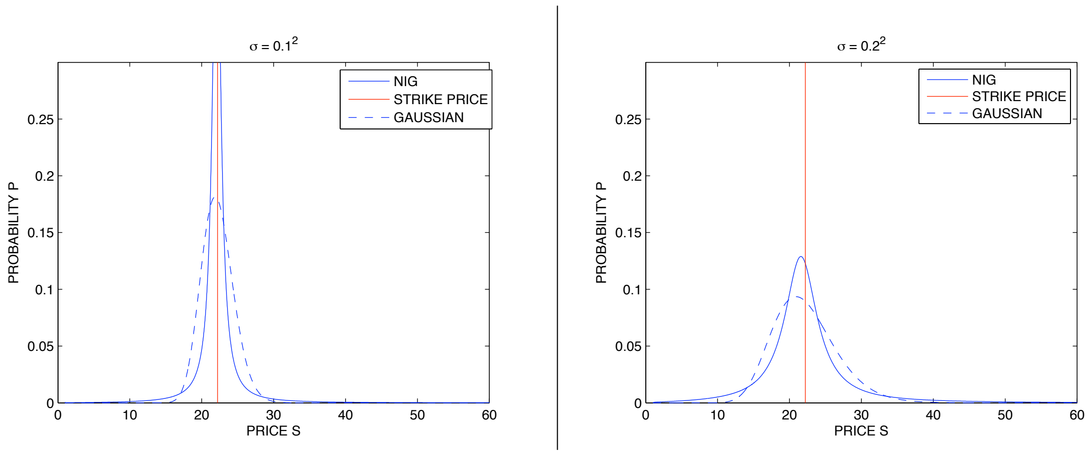


Figure 4.11: Probability distribution resulting from Brownian and NIG random walks for two distinct values of σ.

4.5.3 Pricing a Barrier Option

Although the models with Lévy noise are not always useful, they can be used to price exotic derivatives such as barrier options. Although European and American options are the most common in the financial world, there exist a plethora of variations on option payouts. American options can be exercised by the buyer at any time until the expiry date, and for example Bermuda options (the intermediate of Europe and America!) can be exercised at a predetermined set of discrete times until the expiry date. This means that the fair value of the option depends on the path of the asset price, not only the final price. An up-and-out down-and-out barrier option is a path-dependent derivative of which the value is nulled if the the asset price crosses a predetermined level at any time in the interval $[0, T]$. These so-called barrier events are not necessarily straightforward to agree upon, since a single trade can be enough to cross the level. In theory, it could be traded in any batch size and in private. Here, it will be assumed that the asset is traded on a public exchange where trades are of standardized sizes and that the prices cannot be manipulated to cross the relevant level. Buying a barrier call option is sensible if one believes a certain asset price, say that of SAS, will rise, but not above an upper level U.

Barriers	Simulation	PI by FFT, Gaussian	PI by FFT, NIG	PI by FFT, Lévy
$L = 0, U = \infty$	1.588	1.600	1.619	1.591
$L = e^1, U = e^4$	1.609	1.600	1.131	0.811
$L = e^2, U = e^{3.5}$	1.336	1.347	0.755	0.551

Table 4.3: Price of an up-and-out down-and-out barrier call option with $T = 1$, $r = 0.1$, $S_0 = e^3$, $E = e^{3.1}$, $\sigma^2_{normal} = 0.2^2$. For the NIG distribution, $\alpha = 1, \beta = 0, \sigma_{NIG} = 0.145^2$ and for the Lévy distribution $\alpha = 1$, $\beta = 0$ and $\sigma_L = 0.108^2$

This gives the buyer a discount on the call option compared to a standard European one. On the other hand, if one expects a period of high volatility or several high impact-events for the relevant asset, it makes sense to sell the call option.

An example is shown below with the same parameters as for the European option in the previous section, except the upper and lower barriers. It is seen that the heavy tail distributions give a much bigger discount per stock when the upper and lower limits approach the strike price - even though the distributions are more concentrated around the mean they give higher probabilities for big fluctuations such that the option price is set to zero.

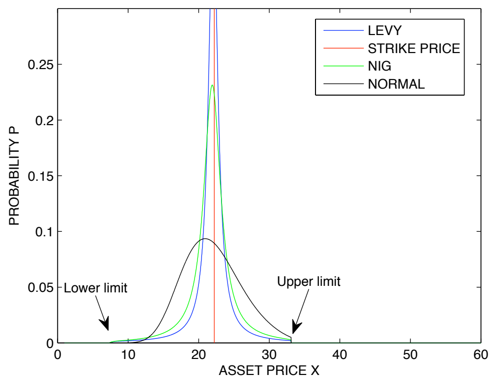


Figure 4.12: Final distributions with $L = e^2 = 7.39$, $U = e^{3.5} = 33.12$

4.6 Summary

It has been shown in this section that PI by FFT can be implemented with more general types of noise than just Brownian motion. Specifically, Lévy and Normal Inverse Gaussian processes stem from families of distributions that have more freedom in shaping the curve than just the standard deviation. These stochastic processes were studied in the context of the Black-Scholes option pricing framework. There are several drawbacks with using other noise sources - generally, there is a nonzero probability that the random variable S takes negative values for a random walk like 4.1. This is not the case with Gaussian noise. Also, with Lévy noise the mean and variance are generally not finite. This is a severe drawback for a standard option pricing model, but it may anyway be useful for stress testing and pricing exotics such as barrier options. In the Black-Scholes framework, the market becomes incomplete, since there are many possible pricing formulas. Usually, the volatility in the random walk is implied by the current option price

However, it seems to be straightforward to implement the models, and the results in reference [26] show that the NIG distribution describes historical data from the OSEBX and Standard and Poor's 500 indexes much better than a normal distribution. The results are much more precise than simulation and since these models are usually one-dimensional they can be run very fast (typically less than one second).

Chapter 5

Discussion, Conclusions and Future Work

5.1 Discussion

It was the great Russian mathematician A. N. Kolmogorov who laid the foundations of probability on measure theory in the 1930s. This provided the tools for studying stochastic processes, the interaction of probability with time. Ω is the set of possible outcomes, $\mathcal{F}$ is a σ-algebra of subsets of Ω and P is a positive measure of total mass 1 on $(\Omega, \mathcal{F})$. In many cases, the relevant state space of the stochastic process $X(t)$ is Euclidean; $\mathbb{R}^d$ with $d = 1$ or $d = 2$. Despite this, there is significant interest in infinite dimensional Hilbert or Banach spaces [24].

Stochastic processes have an extensive range of applications in physics, engineering and economics. However, it is a big family of processes that include the familiar Markov chains and random walks as well as semimartingales and measure-valued diffusions. The 1930s and 1940s were the 'heroic age' of probability, when mathematicians such as Lévy, Itô, Kolmogorov and Khintchine made great contributions to the field. Since the 1970s, there has been a revived interest in stochastic processes, much of it due to the developments in financial mathematics, especially option pricing. In this thesis, the examples are taken from physics and finance, where the examples of uses abound. Analytic solutions exist but usually only for simple cases. It is therefore of great interest to develop efficient and precise numerical strategies. The most common way by far is by simulation, and books that treat numerical solutions of stochastic differential equations' are usually only about simulation. Path integration, lattices, finite differences and spectral methods are other methods that have been shown to work. Path integration by FFT is a relatively recent development that has not received a lot of attention in the literature.

The author of this thesis looked at the possibility of studying lattice QCD, which is a numerical strategy to solve the quantum chromodynamics theory of quarks and gluons in quantum physics. The goal of this theory is to make predictions about the deconfinement of quarks and resulting formation of quark-gluon plasma at high energies. Analytic solutions are usually not available because of the highly nonlinear nature of the strong nuclear force, but by discretizing the domain and reducing the spacing between the lattice points it approaches continuum lattice QCD. However, the theory proved much too comprehensive for the master's thesis of a non-physicist!

The engineering challenges in constructing a particle accelerator provides another stochastic model that is slightly more down-to-earth than lattice QCD, as seen in chapter three. Implementing models with more general types of noise is of interest both in the context of the numerical solution and of the financial mathematics. The goal of the thesis was to study the performance of PI by FFT in the context of applications in physics and finance and to compare the results with other numerical approaches. Some developments have been made and conclusions can be drawn from the results above.

5.2 Conclusions

Most algorithms have drawbacks or disadvantages and are not practical in every case. PI by FFT is only valid for random walks with additive noise, which is a limitation. It has been shown here that by solving an ordinary differential equation one can find the necessary transformation in stochastic calculus to get the model in the desired form. However, as seen in figure 3.12, the resulting transformed grid is not always convenient - it may only be possible to study the evolution in one part of the original grid (in this case, not close to the origin), or the number of grid points required may be exceedingly large after the transformation.

It is also shown that PI by FFT works well with Lévy and NIG processes, which gives many interesting possibilities. A drawback is that the grid must be extended to allow for the heavier tails of these distributions, but in one dimension this is not problematic. For higher dimensional models, it makes the solution more CPU expensive. In the context of option pricing, these processes consist of continuous motion interspersed with jump discontinuities of random size appearing at random times, as seen in figure 4.7. In general, this seems to result in a higher option price because of the higher probability that is allocated to large movements in the path of the underlying asset. One problem of standard financial models is certainly that they do not allow for such jumps.

The two-dimensional model in section 3 provides a way of comparing the performance of PI by FFT against regular PI, simulation and finite differences, thanks to the exact solution of the harmonic oscillator. Surprisingly, it is also shown that a factor is missing in the exact solution in the article by Zorzano et al. [17]. The results shown in figure 3.21 indicate that PI by FFT becomes truly advantageous only in multidimensional models with noise entering in more than one dimension. Also, the finite difference scheme is seen to be the most effective one. It has a great advantage with respect to PI and PI by FFT since the size of the time step can be decreased indefinitely without causing any numerical instabilities. Zorzano et al also obtain approximately the same results with a finite element scheme for the same equation [17]. However, it is not always trivial to transform an SDE to the Fokker-Planck equation, and finite difference schemes are not always available for these, e.g. equation 4.3. In this case, PI by FFT is an excellent candidate for a numerical solution, certainly better than simulation.

A lot of care must be taken in the implementation of both PI and PI by FFT, since the interpolation routine or the numerical integration can claim large amounts of CPU power if the grid is too big. Also, the step sizes must be of corresponding sizes, preferably with more than small enough grid step sizes before one decreases Δt.

5.3 Further Research

The Fokker-Planck equation in section 3 is in two dimensions, whereas the full phase picture is in six. While it is possible to model each of the three physical directions independently in two dimensions each, it may in some cases be too simplistic. An implementation of the six-dimensional model would be of interest both in the numerics and the physics context. However, in order to get a highly accurate solution this would require a bigger machine with a faster processor than what is available for batch jobs at CERN. Access to a supercomputer or a scientific computing grid would allow such computations.

In all the models in this thesis the factors in front of the terms have been independent of time. A further development could be the implementation of time-varying volatility, which is often used in the finance literature. It is argued that the volatility of financial assets changes with time, or that it is even a stochastic process in itself.

Another numerical approach that has received a lot of attention in recent years is a class of spectral methods called the Wiener Chaos Expansion. It can be shown that the Hermite polyonomials

$$h_n(x,t) = \frac{(-t)^n}{n!} e^{x^2/2t} \frac{\mathrm{d}^n}{\mathrm{d}x^n} e^{-x^2/2t} \quad n = 0, 1, 2, \ldots$$

in Itô calculus play the role of $\frac{t^n}{n!}$ in ordinary calculus [1]. Wuan Luo in his Phd thesis [18] shows how the Fourier-Hermite expansion in combination with a tensor product called a Wick polynomial can be applied to solve ordinary SDEs as well as stochastic partial differential equations of the form

$$u_t = \mathcal{L}(u) + \sigma \dot{W}(t),$$

where $\mathcal{L}$ is a linear or nonlinear differential operator in the variables. He applies this strategy to the stochastic Burgers equation and stochastic Navier-Stokes equations. It would be of great interest to see if the ideas behind the PI by FFT algorithm could be applied to stochastic PDEs as well.

Bibliography

[1] Bernt Oksendal. Stochastic Differential Equations. Springer, 2000.

[2] Steven Shreve. Stochastic Calculus for Finance - Continuous-Time Models. Springer Finance, 2004.

[3] Eckhard Platen Peter E. Kloeden. Numerical Solution of Stochastic Differential Equations. Springer, 1992.

[4] Desmond J. Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. SIAM Review, 43(3):525–546, 2001.

[5] Harald Krogstad. The 1d fft. How to use the MATLAB FFT2-routines, 2004.

[6] Steven G. Johnson Matteo Frigo. Fftw for version 3.3.2. FFTW, 2009.

[7] Incnis Mrsi. Path integral formulation. Path Integrals.

[8] Eirik Mo and Arvid Naess. Efficient path integration by FFT. In Applications of Statistics and Probability in Civil Engineering, 2007.

[9] Christian Skaug and Arvid Naess. Fast and accurate pricing of discretely monitored barrier options by numerical path integration. Computational Economics, 30(2):143–151, 2007.

[10] J. Douglas Faires Richard L. Burden. Numerical Analysis. Thomson Brooks/Cole, 8 edition, 2005.

[11] William Vetterling William Press, Saul Teukolsky. Numerical Recipes in C++. Cambridge University Press, 2002.

[12] Alexander W. Chao. Lecture Notes in Physics. Springer Berlin / Heidelberg, 1988.

[13] Donald Perkins. Introduction to High Energy Physics. Cambridge University Press, 2000.

[14] S. Chandrasekhar. Stochastic problems in physics and astronomy. Reviews of Modern Physics, 1943.

[15] P. Hanggi S.I. Denisov, W. Horsthemke. Generalized fokker-planck equation: Derivation and exact solutions. The European Physical Journal B, 2009.

[16] D.W. Jordan and P. Smith. Nonlinear Ordinary Differential Equations. Oxford University Press, 2006.

[17] L. Vazquez M.P. Zorzano, H. Mais. Numerical solution of two-dimensional fokker-planck equations. Applied Mathematics and Computation, 1999.

[18] Wuan Luo. Wiener Chaos Expansion and Numerical Solutions of Stochastic Partial Differential Equations. PhD thesis, California Institute of Technology, 2006.

[19] Jeff Dewynne Paul Wilmott, Sam Howison. The Mathematics of Financial Derivatives. Press Syndicate of the University of Cambridge, 1996.

[20] Nassim Nicholas Taleb. The Black Swan: The Impact of the Highly Improbable. Random House, 2007.

[21] Hans L. Pecseli. Fluctuations in Physical Systems. Cambridge University Press, 2000.

[22] Christian Skaug and Arvid Naess. Path integration methods for calculating response statistics of nonlinear oscillators driven by α-stable lévy noise. IUTAM Symposium on Nonlinearity and Stochastic Structural Dynamics, 2001.

[23] M Grigoriu. Numerical solution of stochastic differential equations with poisson and lévy white noise. Physical Review E, 80, 2009.

[24] David Applebaum. Lévy processes - from probability to finance and quantum groups. Notices of the American Mathematical Society, 51(11):1336–1347, 2009.

[25] M. Sorensen O. Barndorff-Nielsen, J. Kent. Normal variance-mean mixtures and z distributions. International Statistical Review, 50(2):145–159, 1982.

[26] Sjur Westgaard Arvid Naess, Eivind G. Aukrust. Pricing of barrier options under a nig market model using numerical path integration. Working paper, 2010.

Appendix A

Source code

All the source code will not be included, since it is mostly variations on the same framework. Most of it is written in C++ with the exception of the last two, which are in Matlab.

1. `RK4() :`

   Runge Kutta 4 in two dimensions

2. `Exact() :`

   The exact solution of the harmonic oscillator.

3. `Convolute() :`

   Convolutes two input vectors using the FFT.

4. `Fokker FFT() :`

   Numerical solution of the harmonic oscillator by PI by FFT.

5. `Fokker PI() :`

   Numerical solution of the harmonic oscillator by PI.

6. `Bilinear() :`

   Bilinear interpolation.

7. `Splie2(), Splin2() :`

   Bicubic splines.

8. `Fokker Finite() :`

   Numerical solution of the Fokker-Planck equation in Matlab.

9. `FFT Levy() :`

   Option pricing with Levy / NIG noise in Matlab.

A.1 Runge-Kutta 4

```
// ALL THESE LIBRARIES ARE INCLUDED IN THE REST OF THE SOURCE CODE AS WELL
#include "TH1F.h"
#include "TH2F.h"
#include "THStack.h"
#include "TProfile.h"
#include "stdio.h"
#include "TCanvas.h"
#include "TGraphErrors.h"
#include "TVirtualFFT.h"
#include "TMatrixD.h"
#include "TVectorD.h"
#include "TRandom.h"
#include "TBenchmark.h"
#include "TMath.h"
#include "TFile.h"
#include <iostream>
#include <string.h>
#include <math.h>
#include <time.h>
#include <fcntl.h>
#include <vector>
#include <iostream>
#include "Real.h"
#include "Ran2.h"

using namespace std;

double PI = TMath::Pi();
const int N = 201; // NUMBER OF GRID POINTS

const int M = 0; // NUMBER OF INTERMEDIATE GRID POINTS
const Int_t NM = N+(N-1)*M;

// GLOBAL VARIABLES
bool ip_scheme = 1;
bool num_scheme = 1;
const int fft_n2 = 2*NM;
Int_t fft_n = fft_n2;
const double T = 2.3;
double dt = 0.1;
const double v0 = 1.0;
const double x0 = 1.0;
double Xmin = -5;
double Ymin = -5;
double Xmax = 5;
double Ymax = 5;
TVirtualFFT *fft = TVirtualFFT::FFT(1,&fft_n, "R2C");
TVirtualFFT *fft2 = TVirtualFFT::FFT(1,&fft_n, "R2C");
TVirtualFFT *fft_b;

// GLOBAL ARRAYS
double temp1[fft_n2];
double temp2[fft_n2];
double re[fft_n2];
double im[fft_n2];
double re2[fft_n2];
double im2[fft_n2];

// TAKE ONE RK4 STEP OF TIME LENGTH h FROM y[2]
void RK4step(double y[2],  double tm[4], double, double h, double yy[2]){
  int size = 2;
  double k1  [size];
  double k2  [size];
  double k3  [size];
  double k4  [size];
  for (int i=0; i<size; i++){
    k1[i] = 0.0;
    k2[i] = 0.0;
    k3[i] = 0.0;
    k4[i] = 0.0;
  }

  for (int i=0; i<size; i++){
    for(int j=0; j<size; j++){
      k1[i] += tm[i*size+j]*y[j];
    }
  }

  for (int i=0; i<size; i++){
    for (int j=0; j<size; j++){
      k2[i] += tm[i*size+j]*(y[j] + h/2*k1[j] );
    }
  }
  for (int i=0; i<size; i++){
    for (int j=0; j<size; j++){
      k3[i] += tm[i*size+j]*(y[j] + h/2*k2[j] );
    }
  }
  for (int i=0; i<size; i++){
    for (int j=0; j<size; j++){
      k4[i] += tm[i*size+j]*(y[j] + h*k3[j] );
    }
  }

  for (int i=0; i<size; i++){
    yy[i] = y[i] + h/6*(k1[i] + 2*k2[i] + 2*k3[i] + k4[i]);
  }
}
```

A.2 Exact

```
// RETURNS THE EXACT SOLUTION IN A 2D HISTOGRAM
TH2F* exact( double K, double gamma, double sigma, double t){
double x[N];
double y[N];


double dx = (Xmax-Xmin)/(N-1);

for (int i=0; i<N; i++){
  x[i] = Xmin + i*dx;
  y[i] = Ymin + i*dx;
}


//EXACT SOLUTION
double p_ex[N][N];
TH2F *exactdistr = new TH2F("exactdistr","exactdistr",N, Xmin-dx/2,
Xmax+dx/2, N, Ymin-dx/2, Ymax+dx/2);
const double mu1 = -gamma/2 + sqrt(gamma*gamma/4 - K);
const double mu2 = -gamma/2 - sqrt(gamma*gamma/4 - K);
const double a = sigma/mu1*(1 - exp(-2*mu1*t));
const double b = sigma/mu2*(1 - exp(-2*mu2*t));
const double h = -2*sigma/(mu1+mu2)*(1 - exp(-(mu1+mu2)*t));
const double delta = a*b - h*h;
double xi,eta;
const double xi0 = (x0*mu1 - v0);
const double eta0 = (x0*mu2 - v0);

double sum = 0;
for (int i=0; i<N; i++){
  for(int j=0; j<N; j++){
    xi = (x[i]*mu1 - y[j])*exp(-mu2*t);
    eta = (x[i]*mu2 - y[j])*exp(-mu1*t);
    double F1 = TMath::Sqrt(gamma*gamma-4*K)*TMath::Exp(gamma*t)/(2*PI*sqrt(delta));
    double F2 = TMath::Exp((-a*(xi-xi0)*(xi-xi0)-2*h*(xi-xi0)*
     (eta-eta0)-b*(eta-eta0)*(eta-eta0))/(2*delta));

    p_ex[i][j] =F1*F2;


    sum += p_ex[i][j]*dx*dx;
    exactdistr->SetBinContent(i+1,j+1,p_ex[i][j]);
  }
}
exactdistr->GetXaxis()->SetTitle("X");
exactdistr->GetYaxis()->SetTitle("V");
exactdistr->SetTitle("Exact solution");

return exactdistr;
};
```

A.3 Convolute

```
// CONVOLUTES THE VECTORS a AND b OF LENGTH NM
void convolute(double a[NM], double b[NM]){

Int_t NM2 = 2*NM;

TVirtualFFT::SetTransform(0);

for(int i=0; i<NM2; i++){
if(i<NM){
temp1[i] = a[i];
temp2[i] = b[i];
}else{
temp1[i] = 0.0;
temp2[i] = 0.0;
}
}
fft->SetPoints(temp2);
fft->Transform();
fft->GetPointsComplex(re,im);

TVirtualFFT::SetTransform(0);

fft2->SetPoints(temp1);
fft2->Transform();
fft2->GetPointsComplex(re2,im2);

// MULTIPLY THE TWO TRANSFORMS
for (int j=0; j<NM2; j++){
double R = re[j]*re2[j] -im[j]*im2[j];
double I = re[j]*im2[j] +im[j]*re2[j];
re[j]=R;
im[j]=I;
}

TVirtualFFT *fft_b =  TVirtualFFT::FFT(1,&fft_n, "C2R");


fft_b->SetPointsComplex(re,im);
fft_b->Transform();
fft_b->GetPoints(temp1);

// PUT THE RESULT IN THE VECTOR a
double factor = (Ymax-Ymin)/(NM2*(NM-1));
for(int i=(NM-1)/2; i<3*(NM-1)/2+1;i++ ){
a[i-(NM-1)/2] = temp1[i]*factor;

}

delete fft_b;

};
```

A.4 Fokker FFT

```
// RETURNS A 2D HISTOGRAM
TH2F* fokker_FFT(double K, double gamma, double sigma){
gBenchmark->Start("FFT");
// INITIALIZING MATRIX AND ROOT FILE ------------------------
double *tm = new double[4];
```

```
tm[0] = 0;
tm[1] = 1;
tm[2] = -K;
tm[3] = -gamma;

// INITIAL VALUES -------------------------------------

double x[N];
double y[N];
double y_mn[NM];
double dx = (Xmax-Xmin)/(N-1);
double dy_m = (Ymax-Ymin)/(NM -1);
for (int i=0; i<N; i++){
  x[i] = Xmin + i*dx;
  y[i] = Ymin + i*dx;
}
for(int i=0; i<N+(N-1)*M; i++){
  y_mn[i] = Ymin + i*dy_m;
}

//HISTOGRAMS ---------------------------------------------------


TH2F *enddistr = new TH2F("enddistr", "enddistr", N,
Xmin-dx/2 , Xmax+dx/2, N, Ymin-dx/2, Ymax+dx/2);

//INITIAL PROBABILITY DENSITY
double p[N][N];
double p_new[N][N];

//normaldistr2D(p,x,y,x0,v0,.25,.25);
TH2F *initialdistr = exact(K,gamma,sigma,0.5);


//SET THE TH2F
double sum=0.0;
double dxdx = dx*dx;
for(int i=0; i<N; i++){
  for(int j=0; j<N; j++){
p[i][j] = initialdistr->GetBinContent(i+1,j+1);
sum+=p[i][j]*dxdx;
  }
}
```

```
delete initialdistr;


//temporary probability density
double p2[NM];

//the normal distribution vector
double q[NM];

// factor
double fac = (sqrt(2*sigma)-sqrt(sigma/2)*gamma*dt)*sqrt(dt);

normaldistr(y_mn,0,fac,q,NM);

//JACOBIAN ---------------------------------------------
double det1p[2] = {0,0};
double det1m[2] = {0,0};
double det2p[2] = {0,0};
double det2m[2] = {0,0};
double xJ = 1.0;
double yJ = 1.0;
const double eps = 0.000001;
double jac1p[2] ={xJ +eps,yJ};
double jac1m[2] ={xJ-eps,yJ};
double jac2p[2] ={xJ,yJ+eps};
double jac2m[2] ={xJ,yJ-eps};
RK4step(jac1p,tm,0.0,-dt,det1p);
RK4step(jac1m,tm,0.0,-dt,det1m);
RK4step(jac2p,tm,0.0,-dt,det2p);
RK4step(jac2m,tm,0.0,-dt,det2m);

double J_mat[4] = {(det1p[0]-det1m[0])/(2*eps), (det2p[0] - det2m[0])/(2*eps),
 (det1p[1] - det1m[1])/(2*eps), (det2p[1] - det2m[1])/(2*eps)};

double J = J_mat[0]*J_mat[3] - J_mat[1]*J_mat[2];
printf("Jacobian is %f \n \n",J);

// initial

printf("Time step %f, sum %f \n",0.5,sum);

// START THE LOOP -----------------------------------------
```

```
for (double t=.5; t <  T-.00001 ; t+=dt){
  sum= 0.0;

  // 2nd derivative table for cubic splines
  //double y2a[N][N];
  //splie2(x,y,p,y2a);
  // STEP BACK
  for(int i=0; i<N; i++){
    for(int j=0; j<NM; j++){
      double yy[2]={x[i], y_mn[j]};
      double yy_t[2]={0,0};
      RK4step(yy, tm, t, -dt, yy_t);

      // bilinear interpolation
      p2[j] = bilinear(p,x,y,yy_t[0],yy_t[1])*J;


  //p2[j]=splin2(x,y,p,y2a,yy_t[0],yy_t[1])*J;
  //if (p2[j]<0){
//p2[j] = bilinear(p,x,y,yy_t[0],yy_t[1])*J;
  //}


    }
// convolute p2 and the normal distr. convolute() returns a length NM vector

convolute(p2,q);


for(int j=0; j<NM; j++){
if(j%(M+1)==0){
if(p2[j]>0 && p2[j]<100){
p_new[i][j/(M+1)] = p2[j];
sum+=p_new[i][j]*dx*dx;

}else{
p_new[i][j/(M+1)] = 0;
}
}
}

  }
  for(int i=0; i<N; i++){
    for(int j=0; j<N; j++){
p[i][j] = p_new[i][j];
//sum+=p[i][j]*dx*dx;
    }
  }

  printf("Time step %f, sum %f \n",t+dt,sum);

}

for(int i=0; i<N; i++){
  for(int j=0; j<N; j++){
    enddistr->SetBinContent(i+1,j+1,p[i][j]);

  }
}

gBenchmark->Show("FFT");
return enddistr;

};
```

A.5 Fokker PI()

```
// 2D FOKKER-PLANCK : HARMONIC OSCILLATOR WITH DAMPING AND NOISE
TH2F* fokker_PI(double K, double gamma, double sigma){



// INITIALIZING MATRIX  ------------------------


//gBenchmark->Start("PI");

double *tm = new double[4];
tm[0] = 0;
tm[1] = 1;
tm[2] = -K;
tm[3] = -gamma;
```

```
// INITIAL VALUES --------------------------------------


double x[N];
double y[N];

double y_mn[NM];

double dx = (Xmax-Xmin)/(N-1);

double dy_m = (Ymax-Ymin)/(NM -1);
for (int i=0; i<N; i++){
  x[i] = Xmin + i*dx;
  y[i] = Ymin + i*dx;
}
for(int i=0; i<NM; i++){
  y_mn[i] = Ymin + i*dy_m;

}

//HISTOGRAMS ---------------------------------------------------

TH2F *enddistr = new TH2F("enddistr", "enddistr", N,
 Xmin-dx/2 , Xmax+dx/2, N, Ymin-dx/2, Ymax+dx/2);

//INITIAL PROBABILITY DENSITY
double p[N][N];
TH2F *initialdistr = exact(K,gamma,sigma,0.5);


//SET THE TH2F
for(int i=0; i<N; i++){
  for(int j=0; j<N; j++){
p[i][j] = initialdistr->GetBinContent(i+1,j+1);
  }
}

delete initialdistr;


//temporary probability density
double p2[N][NM];
```

```
//the normal distribution vector for Euler and matrix for Taylor 2.0
double q[NM];
//double q2[N][N];

//JACOBIAN ----------------------------------------------
double det1p[2] = {0,0};
double det1m[2] = {0,0};
double det2p[2] = {0,0};
double det2m[2] = {0,0};
double xJ = 1.0;
double yJ = 1.0;
const double eps = 0.0000001;
double jac1p[2] ={xJ +eps,yJ};
double jac1m[2] ={xJ-eps,yJ};
double jac2p[2] ={xJ,yJ+eps};
double jac2m[2] ={xJ,yJ-eps};
RK4step(jac1p,tm,0.0,-dt,det1p);
RK4step(jac1m,tm,0.0,-dt,det1m);
RK4step(jac2p,tm,0.0,-dt,det2p);
RK4step(jac2m,tm,0.0,-dt,det2m);

double J_mat[4] = {(det1p[0]-det1m[0])/(2*eps), (det2p[0] - det2m[0])/(2*eps), (det1p

double J = J_mat[0]*J_mat[3] - J_mat[1]*J_mat[2];
printf("Jacobian is %f \n \n",J);


double fac;
if (num_scheme)  fac = (sqrt(2*sigma)-sqrt(sigma/2)*gamma*dt)*sqrt(dt);
else  fac = sqrt((2*sigma)*dt);

double min = pow(10,-9);

// START THE LOOP ------------------------------------------

for (double t=.5; t<T-.00001; t+=dt){
double sum= 0;

// 2nd derivative table for cubic interpolation


// IF THE CUBIC SPLINE SCHEME IS CHOSEN, CALCULATE
// THE AUXILIARY 2ND DERIVATIVE TABLE HERE
```

```
double y2a[N][N];
if (ip_scheme){
  splie2(x,y,p,y2a);
}


// STEP BACK
for(int i=0; i<N; i++){

for(int j=0; j<NM; j++){

double yy[2]={x[i], y_mn[j]};
double yy_t[2]={0,0};
RK4step(yy, tm, t, -dt, yy_t);

// BILINEAR INTERPOLATION
if (!ip_scheme){
  p2[i][j] = bilinear(p,x,y,yy_t[0],yy_t[1])*J;
}
if (ip_scheme){

  p2[i][j]=splin2(x,y,p,y2a,yy_t[0],yy_t[1])*J;
  if (p2[i][j]<0){
    p2[i][j] = bilinear(p,x,y,yy_t[0],yy_t[1])*J;
  }
}
}
}

// INTEGRATE WITH SIMPSONS COMPOSITE METHOD

for(int j=0; j<N; j++){
normaldistr(y_mn,y[j],fac,q,NM);
for(int i=0; i<N; i++){

p[i][j]=q[0]*p2[i][0]+q[NM-1]*p2[i][NM-1];
for(int k=0; k<NM; k++){
if (q[k]>min){
if(k%2==0)p[i][j]+=2*q[k]*p2[i][k];
if(k%2==1)p[i][j]+=4*q[k]*p2[i][k];
}
}
p[i][j] = p[i][j]*dy_m/3;
if (p[i][j]<0){
p[i][j] = 0;
printf("Zero! Zero! \n" );
}
sum+=p[i][j]*dx*dx;
}
}

printf("Time step %f, sum %f \n",t+dt,sum);
}

for(int i=0; i<N; i++){
for(int j=0; j<N; j++){
enddistr->SetBinContent(i+1,j+1,p[i][j]);
}
}


return enddistr;
};
```

A.6 Bilinear()

```
// BILINEAR INTERPOLATION
double bilinear(double p[N][N], double x[N], double y[N], double xx, double yy){
  int x1 = N;
  int x2 = N;
  for (int i=0; i<N; i++){
    if (x[i]>xx){
      x1 = i-1;
      break;
    }
  }
  if (x1==-1 || x1>=N-1){
    return 0;
  }
  for(int i=0; i<N; i++){
    if(y[i]>yy){
      x2 = i-1;
      break;
    }
  }
  if(x2==-1 || x2>=N-1){
    return 0;
  }
  double t = (xx-x[x1])/(x[1]-x[0]);
  double u = (yy-y[x2])/(y[1]-y[0]);

  return ((1-t)*(1-u)*p[x1][x2] + t*(1-u)*p[x1+1][x2] +
   t*u*p[x1+1][x2+1] + (1-t)*u*p[x1][x2+1]);

};
```

A.7 Splin2(), Splie2()

```
// CONSTRUCT THE AUXILIARY SECOND-DERIVATIVE TABLE y2a
void splie2(double [], double x2a[N], double ya[N][N], double y2a[N][N]){

  double ya_t[N];
  double y2a_t[N];
  for(int j=0; j<N; j++){
    // fill a vector ya_t from ya[j][:]
    for(int k=0; k<N; k++) ya_t[k] = ya[j][k];

    // CALLS THE FUNCTION SPLINE BELOW
    spline(x2a,ya_t,1.0e30, 1.0e30, y2a_t);
    // put them back in the second derivative matrix
    for(int k=0; k<N; k++) y2a[j][k] = y2a_t[k];

  }
};

//EVALUATE THE POINT x1, x2
double splin2(double x1a[N], double x2a[N], double ya[N][N], double y2a[N][N],
const double x1, const double x2){
  double ya_t[N];
  double y2a_t[N];
  double yytmp[N];
  double ytmp[N];
  for(int j=0; j<N; j++){
    for(int k=0; k<N; k++){
      ya_t[k]=ya[j][k];
      y2a_t[k]=y2a[j][k];
    }
    // CALLS THE FUNCTION SPLINT BELOW
    yytmp[j]=splint(x2a,ya_t,y2a_t,x2);
  }
  spline(x1a, yytmp, 1.0e30, 1.0e30,ytmp);
  return splint(x1a, yytmp, ytmp, x1);
};


// COMPUTES SECOND DERIVATIVES IN GRID POINTS IN X-DIRECTION
void spline(double x[N], double y[N], const double yp1,
const double ypn, double y2[N]){
  double p,qn,sig,un;
  double u[N-1];
  if(yp1>0.99e30){
    y2[0] = u[0] = 0;
  }else{
    y2[0] = -0.5;
    u[0] = (3.0/(x[1]-x[0]))*((y[1]-y[0])/(x[1]-x[0])-yp1);
  }

  for(int i=1; i<N-1; i++){
    sig = (x[i]-x[i-1])/(x[i+1]-x[i-1]);
    p=sig*y2[i-1]+2.0;
    y2[i]=(sig-1.0)/p;
    u[i]=(y[i+1]-y[i])/(x[i+1]-x[i]) - (y[i]-y[i-1])/(x[i]-x[i-1]);
    u[i]=(6.0*u[i]/(x[i+1]-x[i-1])-sig*u[i-1])/p;
  }
  if(ypn>0.99e30){
    qn=un=0;
  }else{
    qn=0.5;
    un = (3.0/(x[N-1]-x[N-2]))*(ypn-(y[N-1]-y[N-2])/(x[N-1]-x[N-2]));
  }
  y2[N-1] = (un-qn*u[N-2])/(qn*y2[N-2]+1.0);
  for(int k=N-2; k>=0; k--){
    y2[k] = y2[k]*y2[k+1]+u[k];
  }
}

// SPLINE INTERPOLATION
double splint(double xa[N], double ya[N], double y2a[N], const double x){
int k;
  double h,b,a;

  int klo=0;
  int khi=N-1;
  while(khi-klo>1){
    k = (khi+klo) >> 1;
    if(xa[k]>x) khi =k;
    else klo = k;
  }
  h = xa[khi] - xa[klo];
  if(h==0) exit(1);
  a = (xa[khi]-x)/h;
  b = (x - xa[klo])/h;
  return a*ya[klo] + b*ya[khi] + ((a*a*a -a)*y2a[klo] + (b*b*b-b)*y2a[khi])*(h*h)/6.0 ;
}
```

A.8 Fokker Finite()

```
// RETURNS THE DISTRIBUTION P WITH INPUT PARAMETERS K, GAMMA, SIGMA
function P = fokker_finite (K, gamma,sigma)

Xmin = -5;
Xmax = 5;
Ymin = -5;
Ymax = 5;

hh = .1;
dt = 0.01;
T = 2.3;

x0 = 1;
y0 = 1;

X = Xmin:hh:Xmax;
Y = Ymin:hh:Ymax;

// INITIAL CONDITION
P = fokker_exact(K,gamma,sigma,hh,.5);

INT = sum(sum(P))*hh^2*100
size(P);
%P ((x0-Xmin)/h+1, (y0-Ymin)/h+1) = 1/h^2;
plothandle = surf(X,Y,P);
axis([Xmin Xmax Ymin Ymax 0 2]);
%title(['TIME = ', num2str(0)]);
```

```
m = length(X);
n = length(Y);

D = sigma;

XX = zeros(1,n*m);
YY = kron(ones(1,m),Y);
for i =1:n*m
    XX(i) = Xmin + hh*(ceil(i/(n))-1);
end

e = ones(n,1);
I_n = spdiags(e,0,n,n);
e_m = ones(m,1);
I_m = spdiags(e,0,m, m);

// SPARSE MATRICES INITIALIZATION -----------------

A = spdiags([e -2*e e], -1:1, n,n);
%A(1,1) = 0;
%A(1,2) = 0;
%A(n,n) =0;
%A(n,n-1)=0;

B = spdiags([-e 0*e e], -1:1, n, n);
%B(1,1) = -2;
%B(1,2) = 2;
%B(n,n) = 2;
%B(n,n-1) = -2;
%B2 = spdiags([YY' YY'],[-1 1],n*m,n*m).*kron(I_m,B);
%B = spdiags([XX' XX'],[-1 1],n*m,n*m).*kron(I_m,B);
B = spdiags(XX',0,n*m,n*m)*kron(I_m,B);

C = spdiags([e e], 0:1, n, n);
%C(n,n) =3;
%C(n,n-1) = -1;

%C = kron(I_m,spdiags([YY' YY'],0:1,n,n).*C);
C = kron(I_m,spdiags(YY'+hh/2,0,n,n)*C);

E = spdiags([e e], -1:0, n, n);
%E(1,1) = 3;
```

```
%E(1,2) = -1;
E = kron(I_m,spdiags(YY'-hh/2,0,n,n)*E);


YY = zeros(1,m*n);

for i =1:m*n

    YY(i) = Ymin + hh*(ceil(i/m)-1);

end

F = spdiags([-e_m  e_m], [-1 1], m, m);
F(1,1) = -2; F(1,2) = 2; F(m,m) = 2; F(m,m-1) =-2;
F=spdiags([YY' YY'],[-1 1],m*n,m*n).*kron(I_n,F);

tic

for t=.5:dt:T
    %% STEP 1----------------------------------

    p = P;
    p = reshape(p,n*m,1);

    p = (kron(I_m,I_n) - D*dt/(hh^2)*kron(I_m,A) - dt/(2*hh)*K*B -dt/(2*hh)*gamma*C
     + dt/(2*hh) *gamma*E)\p ;



    p = reshape(p,n,m);

    %% STEP 2 ----------------------------------
    p = reshape(p',m*n,1);

    p = (kron(I_n,I_m) + dt/(2*hh)*F) \  p;


    p = reshape(p,m,n)';

    P = p;

    set(plothandle,'zdata',P);
    title(['TIME = ', num2str(t)]);
    drawnow;


end

INT = sum(sum(P))

t = toc


end
```

A.9 FFT levy()

```
// RANDOM WALK WITH NIG, NORMAL OR LEVY PROCESS
function [x y] = fft_levy()

Xmax = 6;
S0 = 3;
E = 3.1;
U = 3.5;
L = 2;
dx = 0.001;
x = 0:dx:Xmax;
r = 0.1;
T=1;
sig = 0.2;

N = length(x)-1;
dt = 0.1;
y = zeros(1,length(x));

// DELTA INITIAL CONDITION
y(S0/dx+1) = 1/dx;

// CHOOSE ONE OF THESE
%q = nig(x,Xmax/2,sig^2*sqrt(dt),1,0);
%q = cauchydistr(x,Xmax/2,sig^2*sqrt(dt));
q = normaldistr(x,Xmax/2,sig*sqrt(dt));

qfft = fft ([q zeros(1,length(q))]);
```

```
size(qfft)
x2 = x - (r-1/2*sig*sig)*dt;
for t = dt:dt:T
// BUILT-IN INTERPOLATION ROUTINE
    y = interp1(x,y,x2,'splines');
    y = ifft(qfft.*fft([y zeros(1,length(q))]));
    y = y(N/2+1:end-N/2-1)*dx;
    y(1:L/dx+1) = 0;
    y(U/dx+1:end) = 0;

end

x2 = x(E/dx+1:end);
y2 = y(E/dx+1:end);
y3 = exp(x2)-exp(E);
p = y2./exp(x2);
P = y3.*p.*(exp(x2+dx/2)-exp(x2-dx/2));
V = exp(-r*T)*sum(P)


plot(exp(x),y./exp(x),'black')
axis([0 60 0 0.3])


end
```