Numerical Solution of Stochastic Differential Equations by use of Path Integration

A study of a stochastic Lotka-Volterra model

Gaute Halvorsen

Master of Science in Physics and Mathematics
Submission date: October 2011
Supervisor: Arvid Næss, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Abstract

Some theory of real and stochastic analysis in order to introduce the Path Integration method in terms of stochastic operators. A theorem presenting sufficient conditions for convergence of the Path Integration method is then presented. The solution of a stochastic Lotka-Volterra model of a prey-predator relationship is then discussed, with and without the predator being harvested. And finally, an adaptive algorithm designed to solve the stochastic Lotka-Volterra model well, is presented.

Notation

- $\mathbb{N}$: The set of all non-negative integers
- $\mathbb{Z}$: The set of all integers
- $\mathbb{Q}$: The set of all rational numbers
- $\mathbb{R}$: The set of all real numbers
- $\mathbb{C}$: The set of all complex numbers
- $\mathbb{R}^+$: The set of all non-negative real numbers
- $\mathbb{R}^n$: the Cartesian product $\underbrace{\mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R}}_{n \text{ times}}$

Contents

Chapter 1

Preliminaries

1.1 Basic real analysis and probability theory

In this section, we shall review some of the basis concepts of real analysis, that will be used later in defining and deriving other concepts.

We start by first defining σ-algebra.

Definition 1.1. Let X be a set, and denote by $\mathcal{P}(X)$ the power set of X. Then $\mathcal{A} \subseteq X$ is called σ-algebra, if it satisfies the following conditions.

- $\mathcal{A}$ is non-empty
- if $A \in \mathcal{A}$, then $A^c \in \mathcal{A}$, where A^c denotes the complement of A in X.
- if the sets $A_1, A_2, \dots$ are in $\mathcal{A}$, then $\bigcup_i A_i \in \mathcal{A}$

A *probability space* is a triplet $(\Omega, \mathcal{A}, P)$, sometimes denoted simply by Ω, where Ω is a set, $\mathcal{A}$ is a σ-algebra of subsets of Ω, and P is a probability measure on $\mathcal{A}$.

Definition 1.2. A real-valued function $P : \mathcal{A} \to [0, 1]$ is a probability measure on $\mathcal{A}$ of the triplet $(\Omega, \mathcal{A}, P)$ if the following conditions are satisfied

- $P(\emptyset) = 0$
- $P(\Omega) = 1$
- If $\{E_i\}$ is a countable collection of pairwise disjoint sets in $\mathcal{A}$, then

$$P\left(\bigcup_i E_i\right) = \sum_i P(E_i)$$

For all practical purposes for this thesis, the set Ω will be $\mathbb{R}^n$ for some $n \in \mathbb{N}$, and the σ-algebra we use will be $\mathcal{M}(\mathbb{R}^n)$, the collection of all subsets of $\mathbb{R}^n$ that satisfies the Carathéodory criterion.

Definition 1.3. A set $E \subset \mathbb{R}^n$ is said to satisfy the *Carathéodory criterion* if

$$\lambda^*(W) = \lambda^*(W \cap E) + \lambda^*(W \cap E^c) \tag{1.1}$$

for all subets W of $\mathbb{R}^n$, where λ^* is the Lebesgue outer measure.

Let $c_i < d_i$ for $i = 1, 2, \ldots, n$ be a collection of pairs of real numbers, and define an open box in $\mathbb{R}^n$ as the set $I_n = f(c_1, d_1, \ldots, c_n, d_n) = \{x \in \mathbb{R}^n : c_i < x_i < d_i, \text{ for } i = 1, \ldots, n\}$, and define the volume of the open box as $V(I_n) = \prod_{i=1}^n (d_i - c_i)$, then the Lebesgue outer measure on $\mathbb{R}^n$ is defined as:

Definition 1.4. For each subset $A \subset \mathbb{R}^n$ the Lebesgue outer measure of A, denoted by $\lambda^*(A)$, is defined by

$$\lambda^*(A) = \inf \left\{ \sum_i V(I_n^i) : \{V_n^i\}_i \text{ open boxes, } \bigcup_i V_n^i \supset A \right\}$$

Said in another way, the Lebesgue outer measure of a subset A of $\mathbb{R}^n$ is the volume of the smallest covering of open boxes of A.

Theorem 1.5. *$\mathcal{M}$ is a σ-algebra and $\mathcal{M}(\mathbb{R}^n) \supset \mathcal{B}(\mathbb{R}^n)$, where $\mathcal{B}(\mathbb{R}^n)$ is the Borel set, the smallest σ-algebra containing all open sets in $\mathbb{R}^n$*

Denote by $(\mathbb{R}^n, \mathcal{M}(\mathbb{R}^n), \lambda)$ or simply $\mathbb{R}^n$ the basic measure space. Here, λ is the Lebesgue measure, which is simply the Lebesgue outer measure restricted to $\mathcal{M}(\mathbb{R}^n)$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be Lebesgue-measurable if $f^{-1}(S) \in \mathcal{M}(\mathbb{R}^n)$ for all $S \in \mathcal{B}(\mathbb{R})$.

Definition 1.6. A Lebesgue-measurable function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be of class $L^p(\mathbb{R}^n)$ if

$$\|f\|_p = \left[\int_{\mathbb{R}^n} |f(x)|^p \, \mathrm{d}x \right]^{(1/p)} < \infty \tag{1.2}$$

for $p \in [1, \infty)$ and of class $L^\infty(\mathbb{R}^n)$ if it is bounded almost everywhere, that is:

$$\|f\|_\infty = \inf\{C \geq 0 : \mu(\{|f| > C\}) = 0\}\} < \infty \tag{1.3}$$

where μ is the Lebesgue-measure. The families of functions $L^P(\mathbb{R}^n)$, $p \in [1, \infty]$ are normed spaces with norms $\| \cdot \|_p$.

In the case of $L^2(\mathbb{R}^n)$, the space has an inner product that induces the norm on $L^p(\mathbb{R}^n)$ defined by$\|f\|_p = \left[\int_{\mathbb{R}^n} |f|^p \, \mathrm{d}x \right]^{(1/p)}$

Definition 1.7. $L^2(\mathbb{R}^n)$ is an inner product space with inner product

$$\langle f, g \rangle = \int_{\mathbb{R}^n} f g \, \mathrm{d}x \tag{1.4}$$

We also mention the very useful Cauchy-Schwartz inequality for this inner product space

Theorem 1.8. *For any pair of functions $f, g \in L^2(\mathbb{R}^n)$, the following inequality holds*

$$|\langle f, g \rangle| \leq \|f\|_2 \|g\|_2$$

This inequality is valid for any inner product space, and thus in particular for $L^2(\mathbb{R}^n)$.

We also introduce the C^α space

Definition 1.9. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be of class $C^\alpha(\mathbb{R}^n)$ for some $\alpha \in \mathbb{N}$ if its partial derivates

$$\frac{\partial^l f}{(\partial x_1)^{l_1}(\partial x_2)^{l_2}\dots(\partial x_n)^{l_n}}, \text{ where } l = \sum_{i=1}^{n} l_i$$

exists, is continuous and is bounded for all $0 \leq l_i \leq n$ such that $l \leq \alpha$.

We now turn our attention more to stochastic analysis and define the stochastic variable and the Markov chain.

We define the filtered probability space $(\Omega, \mathcal{F}, \mathcal{F}_t, P)$, where Ω is a set $\mathcal{F}$ is a σ-algebra of Ω, while $\mathcal{F}_t$ is an increasing family of sub-σ-algebras with the properties $\mathcal{F}_t \subseteq \mathcal{F}$ for all $t \in [0, \infty)$ and in addition satisfies for all $t \in [0, \infty)$: $\mathcal{F}_s \subseteq \mathcal{F}_t$ for all $s \in [0, t]$. P is as before a probability measure satisfying the basic axioms mentioned above.

A stochastic variable X is formally a mapping $X : (\Omega, \mathcal{F}) \to \mathbb{R}^n$ that is measurable with respect to $(\Omega, \mathcal{F})$, that is, $X^{-1}(S) \in \mathcal{F}$ for all $S \in \mathcal{B}(\mathbb{R}^n)$. A family of stochastic variables $\{X_t\}_t$ is called a stochastic process if it is indexed by either $t \in [0, \infty)$ or $t \in \mathbb{N}$. The stochastic process is *adapted to the filtration* $\mathcal{F}_t$ if X_t is measurable with respect to $(\Omega, \mathcal{F}_t)$ for all $t \in [0, \infty)$. Next, we define what it means for a stochastic variable to have a probability density.

Definition 1.10. A stochastic variable X is said to have a probability density if there exists a function $d_X : \mathbb{R}^n \to \mathbb{R}$ such that for all $S \in \mathcal{B}(\mathbb{R}^n)$ we have

$$P(\{\omega \in \Omega : X(\omega) \in S\}) = \int_S d_X(x)\,\mathrm{d}x. \tag{1.5}$$

One very important density, which will also be important for the derivation of Path Integration is the Gaussian density

Definition 1.11. A random variable $X : (\Omega, \mathcal{F}) \to \mathbb{R}^n$ is said to be Gaussian or Normal if there exists a vector $\mu \in \mathbb{R}^n$ and a symmetric positive-definite $n \times n$ matrix Σ such that

$$d_X(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right), \tag{1.6}$$

where $|\Sigma|$ is the determinant of Σ.

Definition 1.12. The expectation operator E applied on a stochastic variable X is defined as

$$E[X] = \int_\Omega X\,\mathrm{d}P \tag{1.7}$$

When X has a probability density d_X, the expected value takes on the more familiar form

$$E[X] = \int_{\mathbb{R}^n} x d_X(x)\,\mathrm{d}x.$$

A mapped stochastic variable $f(X)$ has an expected value defined as

$$E[f(X)] = \int_{\mathbb{R}^n} f(x) d_X(x)\,\mathrm{d}x. \tag{1.8}$$

The concept of Markov processes will be very important for this thesis, as it will turn out that the numerical approach we will use to derive Path Integration results in a Markov process.

Definition 1.13. A stochastic process is called a Markov process if for times $0 \leq t_1 < t_2 < \cdots < t_m < t_{m+1} < \infty$ with corresponding spatial points $x_1, x_2, \ldots, x_m \in \mathbb{R}^n$ we have the the equality

$$P(X_{t_{m+1}} \in S | X_{t_1} = x_1 \cap \cdots \cap X_{t_m} = x_m) = P(X_{t_{m+1}} \in S | X_{t_m} = x_m), \text{ for all } S \in \mathcal{B}(\mathbb{R}^n). \tag{1.9}$$

In a similar manner, we call the stochastic process X_t a Markov chain if the same property holds, but for non-negative integer times.

Definition 1.14. Let X_t be a Markov Process. Define the measure $K(S, x', t', t)$ with $t' > t$ by

$$K(S, x, t', t) = P(X_{t'} \in S | X_t = x). \tag{1.10}$$

We call $K(S, x, t', t)$ the transition measure of X_t. In addition, if there exists a function $k(y, x, t', t)$ such that

$$K(S, x, t', t) = \int_S k(y, x, t', t)\, \mathrm{d}y, \text{ for all } S \in \mathcal{B}(\mathbb{R}^n), \tag{1.11}$$

then we call $k(y, x, t', t)$ the transition kernel of X_t.

Sometimes the transition kernel will not depend explicitly on the times t and t', but rather, on $\Delta t = t' - t$. In this case we call the transition kernel *time-invariant*, which is defined more precisely below

Definition 1.15. If the transition kernel is *time-invariant*, that is

$$k(y, x, t', t) = k(y, x, t' + h, t + h), \text{ for all } h \in [0, \infty). \tag{1.12}$$

We say that the Markov process is time-homogeneous and write $k(y, x, t' - t) = k(y, x, t', t)$.

One important consequence of a Markov process being time-homogeneous, is that Chapman-Kolmogorov equation is valid for the transition kernel of these processes. We mention this equation briefly, as it will be used later

Theorem 1.16. *(Chapman-Kolmogorov Equation): The transition kernel of a time-homogeneous Markov process satisfies*

$$k(y, x, t + t') = \int_{\mathbb{R}^n} k(y, z, t) k(z, x, t')\, dz, \textit{ for all } t, t' > 0. \tag{1.13}$$

1.2 The stochastic differential equation

The class of SDE that we will consider here is of the type

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t, \tag{1.14}$$

where Y_t is an n-dimensional state space vector, and W_t is an m-dimensional Lévy-process, $a(X_t, t) : \mathbb{R}^n \times [0, T] \to \mathbb{R}^n$ is a vector function and $b(X_t, t) : \mathbb{R}^n \times [0, T] \to \mathrm{MAT}(n, m)$ is a matrix function. A Lévy-process is characterized by the following properties

- $W_0 = 0$, almost surely
- W_t has independent increments
- W_t has stationary increments
- $t \to W_t$ is almost surely Cádlág.

The independent increments property means that if $0 \leq t_1 < t_2 < t_3 < \infty$, then $W_{t_2} - W_{t_1}$ and $W_{t_3} - W_{t_2}$ are independent. The stationary increments property means that for $t > s$, $W_t - W_s$ distributed as W_{t-s}. The Cádlág property means that for any $0 < t < \infty$, the limit $\lim_{s \to t^-} W_s$ exists, and $\lim_{s \to t^+} W_s = W_t$ almost surely.

A lévy process is quite general, and has been studied quite extensively for use with Numerical Path Integration in (Kleppe). In this thesis however, a specific Lévy process will be used, namely the Wiener process, which is perhaps better know under the name Brownian motion.

Definition 1.17. A Wiener process B_t is characterized by the following properties

- $B_0 = 0$
- B_t is, almost surely, continuous
- B_t has independent increments with $B_t - B_s \sim \mathcal{N}(0, t - s) \,:\, 0 \leq s < t$

For the purposes of this thesis, B_t always denotes a Wiener process.

We shall throughout this report, interpret equation (1.14) as a symbolic way of representing the integral equation

$$X_t = X_{t'} + \int_{t'}^{t} a(X_t, t)\,\mathrm{d}t + b(X_t, t)\mathrm{d}W_t \tag{1.15}$$

where the last integral is an Itô Stochastic Integral, which is explained in the next section. Furthermore, we shall sometimes, for typographical reasons write $a(y)$ and $b(y)$ instead of $a(y, t)$ and $b(y, t)$, but all the below results are still valid for functions a and b depending explicitly on time.

The term $a(X_t, t)$ is often referred to the deterministic term, and $b(X_t, t)$ is often referred to the diffusion term. One way to explain this, is by observing that in the absence of the term $b(X_t, t)$, the solution X_t would have a purely deterministic behavior, and in the absence of the term $a(X_t, t)$, $X_t - X_{t'}$ would be a pure white-noise process.

1.3 Itô Calculus

On a probability space $(\Omega, \mathcal{A}, P)$, let $\{\mathcal{A}_t, t \geq 0\}$ be an increasing family of sub-σ-algebras of $\mathcal{A}$. Let W_t be a Brownian motion process such that W_t is $\mathcal{A}_t$-measurable with

$$\mathrm{E}(W_t|\mathcal{A}_0) = 0, \text{ and } \mathrm{E}(W_t - W_s|\mathcal{A}_s) = 0.$$

Denote by $\mathcal{L}_T^2$ the set of functions $f : \Omega \times [0, T] \to \mathbb{R}$ that satisfies

- F is jointly $\mathcal{L} \times \mathcal{A}$-measurable
- $\int_0^T \mathrm{E}_x(f(x,t)^2)\,\mathrm{d}t < \infty$
- $\mathrm{E}_x(f(x,t)^2) < \infty$ for all $0 \leq t \leq T$
- $g_t(x) = f(x,t)$ is $\mathcal{A}_t$-measurable for all $0 \leq t \leq T$.

Let $\mathcal{S}_T^2$ denote the subset of all step functions in $\mathcal{L}_T^2$, then it can be proved that $\mathcal{S}_T^2$ is dense in $\mathcal{L}_T^2$ under the norm

$$\|f\|_{2,T} = \sqrt{\int_0^T \mathrm{E}_x(f(x,t)^2\,\mathrm{d}t},$$

that is, any function in $\mathcal{L}_T^2$ can be approximated arbitrarily well by a sequence of step functions in $\mathcal{S}_T^2$. Let $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T$ be a partition of $[0, T]$, and let the corresponding step function that converges to f be represented by the random variable $f_j(x)$ at time t_j for $0 \leq j \leq n$, then the Itô stochastic integral of f is defined as

$$\int_0^T f(x,s)\,\mathrm{d}W_s = \lim_{n\to\infty} \sum_{j=1}^n f_j(x)[W_{t_{j+1}}(x) - W_{t_j}(x)].$$

The Itô stochastic integral has a couple of useful properties, which will be listed here, but not proved. Supposing that the above conditions are satisfied and $f, g \in \mathcal{L}_T^2$, and $\alpha, \beta \in \mathbb{R}$ then

$$\int_0^T f(x,s)\,\mathrm{d}W_s \text{ is } \mathcal{A}_T\text{-measurable,}$$

$$\mathrm{E}\left(\int_0^T f(x,s)\,\mathrm{d}W_s\right) = 0$$

$$\mathrm{E}\left[\left(\int_0^T f(x,s)\,\mathrm{d}W_s\right)^2\right] = \int_0^T \mathrm{E}(f(x,s)^2)\,\mathrm{d}t$$

$$\int_0^T \alpha f(x,s) + \beta g(x,s)\,\mathrm{d}W_s = \alpha \int_0^T f(x,s)\,\mathrm{d}W_s + \beta \int_0^T g(x,s)\,\mathrm{d}W_s$$

$$\mathrm{E}\left[\left(\int_{t_0}^t f(x,s)\,\mathrm{d}W_s\right)|\mathcal{A}_{t_0}\right] = 0, \text{ for } 0 \leq t_0 \leq t \leq T$$

$$Z_t(x) = \int_{t_0}^t f(x,s)\,\mathrm{d}W_s \text{ has an, almost surely, continuous stochastic path}$$

We shall, throughout this report, assume that these conditions are true, so that the above properties can be used.

Another interesting application of the Itô stochastic integral answers the question; if X_t is governed by the equation

$$X_t = X_{t_0} + \int_{t_0}^{t} a \,\mathrm{d}s + \int_{t_0}^{t} b \,\mathrm{d}W_s$$

and if $Y_t = U(X_t, t)$ is a transformation of X_t can we say something about the behavior of Y_t? It turns out that, if U has continuous partial derivatives $\frac{\partial U}{\partial t}$, $\frac{\partial U}{\partial x}$, and $\frac{\partial^2 U}{\partial x^2}$, then

$$Y_t = Y_{t_0} + \int_{t_0}^{t} \frac{\partial U(X_s,s)}{\partial t} + a\frac{\partial U(X_s,s)}{\partial X_s} + \frac{1}{2}b^2\frac{\partial^2 U(X_s,s)}{\partial X_s^2}\,\mathrm{d}s + \int_{t_0}^{t} b\frac{\partial U(X_s,s)}{\partial X_s}\,\mathrm{d}W_s.$$

In [1], this is called *The Itô Formula.*

1.4 Existence and uniqueness of solution

The following theorem, which will be stated without proof, gives sufficient conditions for existence and uniqueness of a solution of a SDE.

Theorem 1.18. *Suppose that the functions $a, b : \mathbb{R}^n \times [t_0, T] \to \mathbb{R}^n$ satisfies the following conditions*

- *$a(x,t)$ and $b(x,t)$ are jointly $\mathcal{L}^2$-measurable in $(x,t) \in \mathbb{R}^n \times [t_0, T]$*
- *There exists a constant $K > 0$ such that*

$$\|a(x,t) - a(y,t)\| \le K\|x - y\|$$
$$\|b(x,t) - b(y,t)\| \le K\|x - y\|$$

  for all $t \in [t_0, T]$ and $x, y \in \mathbb{R}^n$

- *There exists a constant $D > 0$ such that*

$$\|a(x,t)\|^2 \le D^2(1 + \|x\|^2)$$
$$\|b(x,t)\|^2 \le D^2(1 + \|x\|^2)$$

and in addition, the random variable X_{t_0} is $\mathcal{A}_{t_0}$-measurable with $E(\|X_{t_0}\|) < \infty$, then the stochastic differential equation

$$dX_t = a(X_t, t)\, dt + b(X_t, t)\, dW_t$$

has a pathwise unique solution X_t on $[t_0, T]$ with

$$\sup_{t_0 \le t \le T} E(\|X_t\|^2) < \infty$$

The second and third requirement puts severe restrictions on the functions a and b, but in many cases the solution will with a high probability be contained in some bounded closed subset of $\mathbb{R}^n$. If that is the case, then if a and b are bounded continuous functions with bounded derivatives on this compact subset, then a and b satisfies all the requirements. We shall throughout this thesis assume that a and b are sufficiently regular, such that an unique solution to the SDE exists.

1.5 The Kahan Summation Algorithm

In exact arithmetic, calculating a sum

$$S = \sum_{i=1}^{n} a_i \tag{1.16}$$

is a straightforward process. When calculating the sum on a computer however, we can run into problems due to the finite representation of digits on a computer, and round-off error when adding small numbers to big numbers. This can especially become a problem when n is a very large integer, where we can end up with very big relative round-off errors. This section assumes some knowledge of the finite precision and round-off error in computer arithmetics.

The naive way to add up a large sum on a computer, can be given by the algorithm

```
**Input**: $\{a_1, a_2, \ldots, a_n\}$
**Output**: $S$, the sum of the inputs.
$S := 0$
**foreach** $i \in \{1, 2, \ldots n\}$ **do**
    $S \leftarrow S + a_i$
**end**
```

Algorithm 1: Naive summation algorithm

If S is the sum of the terms $\{a_0, a_1, \ldots, a_n\}$ calculated on a computer, it can be shown [2] that the relative error of summation is bounded by

$$\frac{|S - \sum_{i=1}^{n} a_i|}{|\sum_{i=1}^{n} a_i|} \leq \epsilon \frac{[(n-1)|a_1| + \sum_{i=2}^{n}(n-i+1)|a_i|]}{|\sum_{i=1}^{n} a_i|} \tag{1.17}$$

where ϵ is the machine epsilon, that is, the upper bound for the relative round-off error for the data type the terms $\{a_1, \ldots, a_n\}$ are stored in. For a double precision number, the machine epsilon is approximately 10^{-16}. Thus the relative error is in the class of $\mathcal{O}(\epsilon n)$, and in the worst case scenario, the relative round-off error grows linearly with n.

Round off-error is not something that we can prevent on a computer, we can, however, employ a balancing strategy which severely reduces the round-off error. If we define $t = S + a_i$, then $c = (t - S) - a_i$ is always zero, in exact arithmetics. In computer arithmetics, however, $t - S$ only represents the digits of a_i that was not cut off by round-off, in short, it recovers the high-order parts of a_i. When subtracting a_i from $(t - S)$, we obtain the low-order part of a_i, that is, the rounding error we obtained by adding a_i to S, up to rounding error. Balancing this round-off error, by subtracting c from the next term a_{i+1} to be added to the sum, should thus provide better results.

The Kahan summation algorithm is given as follow

```
**Input**: $\{a_1, a_2, \ldots, a_n\}$
**Output**: $S$, the sum of the inputs.
$S := 0$
$c := 0$
**foreach** $i \in \{1, 2, \ldots n\}$ **do**
    $y \leftarrow a_i - c$
    $t \leftarrow S + y$
    $c \leftarrow (t - S) - y$
    $S \leftarrow t$
**end**
```

Algorithm 2: The Kahan summation algorithm

It can be shown that for the result of using the Kahan summation algorithm, the following is valid

$$\frac{|S - \sum_{i=1}^{n} a_i|}{|\sum_{i=1}^{n} a_i|} \leq \left[2\epsilon + \mathcal{O}(n\epsilon^2)\right] \frac{\sum_{i=1}^{n} |a_i|}{|\sum_{i=1}^{n} a_i|} \tag{1.18}$$

where $(\sum_{i=1}^{n} |a_i|)/|\sum_{i=1}^{n} a_i|$ is called the condition number of the sum. We see that here, the relative error is in the class of $\mathcal{O}(2\epsilon + n\epsilon^2)$, which is a considerable improvement over $\mathcal{O}(n\epsilon)$. The round-off error can be quite bad, also for the Kahan summation algorithm, if the sum is badly conditioned. That means that the condition number $(\sum_{i=1}^{n} |a_i|)/|\sum_{i=1}^{n} a_i|$ is very large in magnitude. But note that if all the terms $\{a_1, \ldots, a_n\}$ share the same sign, then the condition number is 1, which is the lowest possible condition number we can obtain. The Kahan summation algorithm will work very well in these cases.

This algorithm will be used in some calculations to improve accuracy, when adding relatively large numbers of positive numbers.

Chapter 2

Path Integration

2.1 Pathwise numerical approximation of Stochastic Differential Equations

In this section, as an introduction to Path Integration, we shall look at the most intuitive and simple method of approximating the path of a SDE. Consider the SDE

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}B_t \tag{2.1}$$

Where B is a Brownian motion process as usual, X_t is a state vector in $\mathbb{R}^n$ and $a, b : \mathbb{R}^n \times [0, T] \to \mathbb{R}^n$. Or written in its equivalent form

$$X_{t'} = X_t + \int_t^{t'} a(X_\tau, \tau)\,\mathrm{d}\tau + \int_t^{t'} b(X_\tau, \tau)\,\mathrm{d}B_\tau. \tag{2.2}$$

We make the simple assumption that $a(X_\tau, \tau) \approx a(X_t, t)$ and $b(X_\tau, \tau) \approx b(X_t, t)$ in $\tau \in [t, t']$, thus we get

$$X_{t'} \approx X_t + a(X_t, t)\Delta t + b(X_t, t)\Delta B_t \tag{2.3}$$

where $\Delta t = t' - t$ and $\Delta B_t = B_{t'} - B_t$. This approximation is known as the Euler Scheme, as it uses the same idea Euler used to approximate ordinary differential equations in his time. The Euler Approximation belongs to a more general class of approximations called Runge-Kutta methods. Where in general, the explicit Runge-Kutta method takes the form

$$\tilde{I} = \Delta t \sum_{i=1}^{s} b_i k_i \tag{2.4}$$

where

$$\begin{aligned}
k_1 &= a(X_t, t)\\
k_2 &= a(X_t + a_{2,1}\Delta t k_1, t + c_2 \Delta t)\\
&\ \vdots\\
k_s &= a(X_t + a_{s,1}\Delta t k_1 + a_{s,2}\Delta t k_2 + \cdots + a_{s,s-1}\Delta t k_{s-1}, t + c_s \Delta t).
\end{aligned}$$

where $\{b_i\}_i$ and $\{a_{i,j}\}_{i,j}$ are appropriately chosen sequences of real numbers such that the methods are consistent and converge. We make the notion of convergence precise in the next definition.

Definition 2.1. Given the Ordinary Differential Equation

$$\mathrm{d}x = a(x,t)\,\mathrm{d}t \tag{2.5}$$

$$x(0) = x_0 \tag{2.6}$$

and some time interval $[0,T]$, with a corresponding set of times $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T$. Denote by $x(T)$ the exact solution of the above ODE at time $t = T$, and by x_T the approximate solution achieved from the RK-method by iterating over $X_{t_{n+1}} = X_{t_n} + (t_{n+1} - t_n)\sum_{i=1}^{s} b_i k_i$, then we say that the RK-method converges at time T if

$$\lim_{h\to 0^+} \|x(T) - x_T\| = 0. \tag{2.7}$$

where $h = \max_n \{t_{n+1} - t_n\}$.

Definition 2.2. A convergent RK-method is of order k if we have the relationship between the exact solution $X(T)$ at time T, the approximate solution x_T at time T and $h = \max_n \{t_{n+1} - t_n\}$

$$\|x(T) - x_T\| < Ch^k \tag{2.8}$$

Denote the term $\sum_{i=1}^{s} b_i k_i$ by $r(a, X_t, t)$ for short, then by replacing the vector $a(X_t, t)$ by $r(a, \Delta t)$ we obtain a higher order approximation for the deterministic term,

$$X_{t'} = X_t + r(a, X_t, t)\,\Delta t + b(X_t, t)\,\Delta B_t. \tag{2.9}$$

It is also possible to obtain a higher order approximation for the diffusion term $b(X_t, t)$, and we shall also look at a higher order scheme, known as the Milstein scheme later in this thesis.

It is interesting to know whether an approximation like in equation (2.9) converges, and in what sense. We would at last want the approximate solution to approximate the expected value and the variance of the exact solution arbitrarily well as $\Delta t \to 0$, as well as any pth moment. A notion of convergence that captures all of these properties, is the notion of *weak convergence*, which is given below

Definition 2.3. A pathwise scheme $\tilde{X}_t$ approximating the solution of X_t of a given stochastic differential equation at discrete equidistant times $0 \leq t_1 < \cdots < t_n = T$, is said to have a weak convergence of order δ if there exists a number $C_h < \infty$ independent of $\delta t = t_{i+1} - t_i$ such that

$$|E[h(\tilde{X}_T)] - E[h(X_T)]| < C_h(\Delta t)^{\delta} \tag{2.10}$$

for all $h \in C_P^{2(\delta+1)}$. The subscript P denotes that h has at most polynomial growth.

The author has not been able to find any general hard results on the weak convergence of the approximation type of equation (2.9), however theory exists and is well known [1][3] that when $r(a, X_t, t) = a(X_t, t)$, that is, the approximation is an Euler Scheme, then the following result exists

Theorem 2.4. *Suppose that $r(a, X_t, t) = a(X_t, t)$, and that there exists constants $K, D > 0$ such that for all $t \in [0,T]$, the following properties holds*

- $\|a(x,t) - a(y,t)\| \leq K\|x-y\|$
- $\|b(x,t) - b(y,t)\| \leq K\|x-y\|$
- $\|a(x,t)\|^2 \leq D^2(1+\|x\|^2)$
- $\|b(x,t)\|^2 \leq D^2(1+\|x\|^2)$
- $a(X_t,t), b(X_t,t) \in C^4$
- $h(x) \in C_p^4$
- *X_0 has finite 4th moment.*

Then there exists a constant C independent of Δt such that the approximation $\tilde{X}_t$ of X_t at $t=T$, given by equation (2.9) *satisfies*

$$|E[h(\tilde{X}_T)] - E[h(X_T)]| < C\Delta t \tag{2.11}$$

In particular, this means that the method converges weakly of order 1. It seems reasonable to assume that similar results will also hold, when $r(a, X_t, t)$ is a higher order method

Consider now the stochastic process $\{X_{i\Delta t}\}_{i=0}^{\infty}$ where (2.9) is the governing equation, we observe that by conditioning $X_{n\Delta t}$ on its past, and using the independent increment property of the Wiener-process, we obtain

$$\begin{aligned}
&(X_{(n+1)\Delta t}|X_{n\Delta t} = x_n \cap \cdots \cap X_0 = x_0) \\
&\qquad = x_n + r(a, x_n, t) + b(x_n, t)(\Delta B_{n\Delta t}|X_{n\Delta t} = x_n \cap \cdots \cap X_0 = x_0) \\
&\qquad = x_n + r(a, x_n, t) + b(x,t)\Delta B_{n\Delta t} = (X_{(n+1)\Delta t}|X_{n\Delta t} = x_n)
\end{aligned} \tag{2.12}$$

Since $\Delta B_{n\Delta t} \sim \mathcal{N}(0, \Delta t I)$, then $X_{(n+1)\Delta t}$ should similarly also be Gaussian distributed under certain conditions.

If zero-rows in the $n \times m$ matrix $b(X_t,t)$ exists, say $r < n$ rows, then rearrange the state space vector X_t and $b(X_t,t)$ such that

$$b_{ij}(X_t,t) = 0 \text{ for } i = 1, \ldots, r \text{ and } j = 1, \ldots, m$$

and such that the remaining $n-r$ rows are not zero rows. Then by defining the *diffusion matrix* $g(x,t)$ as the matrix product $g(x,t) = b(x,t)b(x,t)^T$, we see that this matrix takes the form

$$g(x,t) = \begin{bmatrix} 0 & 0 \\ 0 & \tilde{g}(x,t) \end{bmatrix} \tag{2.13}$$

where 0 represents appropriate 0-matrices and $\tilde{g}(x,t)$ is the $(n-r)\times(n-r)$ submatrix corresponding to the product $\tilde{b}(x,t)\tilde{b}(x,t)^T$, where $\tilde{b}(x,t)$ is the $(n-r)\times m$ submatrix of $b(x,t)$ of non-zero rows. For reasons we shall see later, we would like the submatrix $\tilde{g}(x,t)$ to symmetric positive definite, the next lemma provides a nice criterion for just that.

Lemma 2.5. *Suppose that the $n \times m$ matrix $b(t)$ has $r < n$ zero-rows, then the $(n-r)\times(n-r)$ sub-matrix $\tilde{g}(t)$ of $g(t)$ as defined in equation* (2.13) *is symmetric positive definite if and only if $b(t)$ has rank $n-r$*

Proof. Let $\tilde{b}(t)$ be defined, as before, as the $(n-r)\times m$ sub-matrix of $b(t)$ consisting of the non-zero rows, then the rows $\tilde{b}(t)$ must be linearly independent, as the rows remaining r rows of $b(t)$ are zero-rows and the rank of $b(t)$ is $n-r$, thus the sub-matrix $\tilde{b}(t)$ also has rank $n-r$.

$\tilde{g}(t) = \tilde{b}(t)\tilde{b}(t)^T$ is positive definite if and only if $x^T\tilde{g}(t)x > 0$ for all $(x \neq 0) \in \mathbb{R}^{n-r}$, this product can be rewritten

$$x^T\tilde{g}(t)x = x^T(\tilde{b}(t)\tilde{b}(t)^T)x = (\tilde{b}(t)^Tx)^T(\tilde{b}(t)^Tx) = \|\tilde{b}(t)^Tx\|_2^2. \tag{2.14}$$

From the last equality, we can see that $x^T\tilde{g}(t)x \geq 0$ for all $x \in \mathbb{R}^{n-r}$ and that $x^T\tilde{g}(t)x = 0$ if and only if $\tilde{b}(t)^Tx = 0$. Now we have already noted that $\tilde{b}(t)$ has rank $n-r$ so $\tilde{b}(t)^Tx = 0$ if and only if $x = 0$, thus $x^T\tilde{g}(t)x > 0$ for all $(x \neq 0) \in \mathbb{R}^{n-r}$.

$\tilde{g}(t)$ is symmetric as $\tilde{g}(t)^T = (\tilde{b}(t)\tilde{b}(t)^T)^T = \tilde{b}(t)\tilde{b}(t)^T = \tilde{g}(t)$.

Suppose that $\tilde{g}(t)$ is positive definite, then $\tilde{b}(t)^Tx = 0$ if and only if $x = 0$, which implies that $\tilde{b}(t)$ has rank $n-r$, which in turn implies that $b(t)$ has rank $n-r$ □

Note that it is necessary to have $m \geq n-r$ for $\tilde{g}(t)$ to be positive definite.

Suppose now that $B(t)$ with r zero-rows has rank $n-r$, then by looking at equation (2.12), it is clear that $X_{(n+1)\Delta t}$ has a degenerate multivariate Gaussian distribution with mean value $\mu(x,t,t') = x + r(a,x,t)\Delta t$ and covariance matrix $\Sigma(x,t,t') = b(x,t)(\Delta t I)b(x,t)^T = \Delta t g(x,t)$, as long as $\tilde{g}(x,t)$ is symmetric positive definite, given by:

$$d_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x',t',x,t) = \prod_{i=1}^{r} \delta(x_i' - x_i - r_i(a,x_i,t)\Delta t) \cdot \tilde{d}_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x',t',x,t) \tag{2.15}$$

where δ denotes the Dirac delta function and

$$\tilde{d}_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x',t',x,t) = \frac{1}{(2\pi\Delta t)^{(n-r)/2}|\tilde{g}(x,t)|^{1/2}}$$
$$\exp\left[-\frac{1}{2\Delta t}\sum_{i=r+1}^{n}\sum_{j=r+1}^{n}(x_i' - x_i - r_i(a,x,t)\Delta t)[\tilde{g}(x,t)^{-1}]_{i-r,j-r}(x_j' - x_j - r_j(a,x,t)\Delta t)\right] \tag{2.16}$$

where $[\tilde{g}(x,t)^{-1}]_{i,j}$ denotes the element in the ith row and the jth column of the inverse matrix of $\tilde{g}(x,t)$. What has been discussed in this section can be nicely summarized in the following theorem.

Theorem 2.6. *The stochastic process $\{X_{i\Delta t}\}_{i=0}^{\infty}$ governed by*

$$X_{t'} = X_t + r(a,X_t,t)\Delta t + b(X_t,t)\Delta B_t \tag{2.17}$$

is a Markov chain with transition kernel $k(x',x,t',t) = d_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x',t',x,t)$ as in (2.15) *and transition measure*

$$K(S,x,t',t) = \int_S k(y,x,t',t)\,dy \tag{2.18}$$

if the $n\times m$ matrix $b(x,t)$ with r zero-rows has rank $n-r$ for all values of x and $t \geq 0$

This result will be very central in deriving Path Integration.

2.1.1 The Milstein Scheme

We shall here briefly discuss the Milstein scheme, which in a certain sense is a higher order approximation for the diffusion term than the Euler scheme.

The idea of the scheme, starting with the stochastic differential equation

$$X_{t'} = X_t + \int_t^{t'} a(X_\tau, \tau)\,\mathrm{d}\tau + \int_t^{t'} b(X_\tau, \tau)\,\mathrm{d}B_\tau, \tag{2.19}$$

is to apply Itô's formula on the mapping $X_t- > B(X_t, t)$, which gives

$$\begin{aligned} X_{t'} = X_t &+ \int_t^{t'} a(X_\tau, \tau)\,\mathrm{d}\tau + \int_t^{t'} [b(X_t, t) \\ &+ \int_t^{\tau} \left(\frac{\partial b(X_s, s)}{\partial s} + b(X_s, s)\frac{\partial B(X_s, s)}{\partial X_s} + \frac{1}{2}(b(X_s, s))^2 \frac{\partial^2 B(X_s, s)}{\partial x^2} \right) ds \\ &+ \int_t^{\tau} b(X_s, s)\frac{\partial b(X_s, s)}{\partial X_s}\,\mathrm{d}B_s]\,\mathrm{d}B_\tau \end{aligned} \tag{2.20}$$

It can be shown, that in a certain sense, in terms of order of convergence, $\mathrm{d}B_t \approx \sqrt{\mathrm{d}t}$[4], so applying this logic we have $\mathrm{d}t\,\mathrm{d}B_t \approx (\mathrm{d}t)^{(3/2)}$. As the double integral with $\mathrm{d}t\mathrm{d}B_t$, heuristically speaking, is the highest order term, we neglect this double integral, which leaves us with.

$$X_{t'} \approx X_t + \int_t^{t'} a(X_\tau, \tau)\,\mathrm{d}\tau + \int_t^{t'} [b(X_t, t) + \int_t^{\tau} b(X_s, s)\frac{\partial b(X_s, s)}{\partial X_s}\,\mathrm{d}B_s]\,\mathrm{d}B_\tau \tag{2.21}$$

By assuming, as before, that the integrands are approximately constant over the integration interval, then

$$X_{t'} \approx X_t + a(X_t, t)\Delta t + b(X_t, t)\Delta B_t + \frac{1}{2}b(X_t, t)\frac{\partial b(X_t, t)}{\partial X_t}((\Delta B_t)^2 - \Delta t), \tag{2.22}$$

this is known as the *Miller scheme*. By replacing $a(X_t, t)$ with a Runge-Kutta approximation $r(a, X_t, t)$, we get what we shall refer to as the *Runge-Kutta-Milstein*(RKMI) approximation, given by

$$X_{t'} \approx X_t + r(a, X_t, t)\Delta t + b(X_t, t)\Delta B_t + \frac{1}{2}b(X_t, t)\frac{\partial b(X_t, t)}{\partial X_t}((\Delta B_t)^2 - \Delta t). \tag{2.23}$$

Theorem 2.6 is still valid for the stochastic process that arises from this approximation, however, the transition kernel changes. By conditioning on X_t, it can be shown[5], by observing that the right hand side of (2.23) is simply a quadratic polynomial of the $N(0, \Delta t)$-distributed variable ΔB_t, that the transition kernel is given by, assuming that the noise only enter through the last dimension

$$k(x', t', x, t) = \prod_{i=1}^{n-1} \delta(x'_i - x_i - r(a, x_i, t)\Delta t)\frac{[k_2(x'_n - c)]^{-1/2}}{2\sqrt{2\pi\Delta t}} \sum_{j=1}^{2} e^{\left[-\frac{(\sqrt{(x'_n - c)/k_2} + (-1)^j k_1)^2}{2\Delta t}\right]} \tag{2.24}$$

$$\begin{aligned} c &= x_n + r_n(a, x_n, t)\Delta t - k_2\Delta t - k_1^2 k_2 \\ k_1 &= \left(\frac{\partial b_{nn}(x_n, t)}{\partial x_n}\right)^{-1} \\ k_2 &= \frac{1}{2}b_{nn}(x_n, t)\frac{\partial b_{nn}(x_n, t)}{\partial x_n}. \end{aligned}$$

This expression is valid if $c < x'n$, if it is not, then $x(x', t', x, t) = 0$.

It can be shown [1][4] that the approximation given in (2.22) converges weakly of order 1, which is exactly the same as the order of convergence for the Euler scheme, but with a more complicated expression. So how exactly is the Milstein scheme better than the Euler scheme? The answer lies in the concept of strong convergence

Definition 2.7. A pathwise scheme $\tilde{X}_t$ approximating the solution of X_t of a given stochastic differential equation at discrete equidistant times $0 \leq t_1 < \cdots < t_n = T$, is said to have a strong convergence of order γ if there exists a number K independent of $\Delta t = t_{i+1} - t_i$ such that

$$E\left(\sup_{t\in[0,T]} |X_t - \tilde{X}_t|\right) < K(\Delta t)^{\gamma} \tag{2.25}$$

Strongly, the Euler scheme only converges of order (1/2)[1][4], the Milstein approximation can be shown to converge of order 1, however, and is better in this regard.

We shall also explore later in an example, if the solution can be improved with the RKMI-approximation when $b(X_t, t)$ takes on values close to zero, on the relevant grid.

2.2 Generalized Cell Mapping

Before deriving Path Integration, we will introduce the technique of Generalized Cell Mapping (GCM). It builds on the discussion of the previous section, and the result of theorem 2.6, which states that the approximation given in equation (2.9).

The basic idea is very simple, we divide the state space into a countable number of cells C_i, which are generally boxes, and assign some probability $p_i(n)$ for the system to stay in cell C_i at time step n. The governing equation that gives the probability for the cells at the next time step, is simply the law of total probability

$$p_i(n) = \sum_j P_{ij}^{(n,n-1} p_j(n-1) \tag{2.26}$$

The transition probabilities $P_{ij}^{(n,n-1)}$ are generally calculated from the transition kernel, but the computation can be quite tricky. It is possible, and usual, to approximate by assigning the probability of cell j to its center c_j and then use the transition kernel defined in theorem 2.6, to approximate the transition probability to cell i by

$$P_{ij}^{(n,n-1)} \approx \int_{C_i} p(x, n\Delta t, x_j, (n-1)\Delta t)\, \mathrm{d}x \tag{2.27}$$

Not surprisingly, the more cells that we divide a bounded subset of $\mathbb{R}^n$ into, the more we expect the approximate distribution of the solution to follow the distribution of the exact solution of the SDE.

2.3 Derivation of Path Integration

Now that we have had a brief review of path-wise numerical approximation of the SDE and the GCM method, we are ready to derive Path Integration.

The basic idea, and the governing equation is similar to the one used in deriving Generalized Cell Mapping. But instead of first dividing the state space into subsets, we look at the we look at the approximation in equation (2.9), which is valid for any value of $x \in \mathbb{R}^n$, and instead use the integral version of the law of total probability

$$p(x', t') = \int_{\mathbb{R}^n} p(x', t'|x, t)p(x, t)\, \mathrm{d}x. \tag{2.28}$$

We shall look at Path Integration in terms of Stochastic Operators. We first define what we mean by the *Path Integration Operator* (PIO).

Definition 2.8. The Path Integration Operator is a mapping $PI_t : L^1(\mathbb{R}^n) \to L^1(\mathbb{R}^n)$, defined for each probability density u_0 by

$$u_t = PI_{t',t}u_0 = \int_{\mathbb{R}^n} k(y, x, t', t)u_0(x)\, \mathrm{d}x \tag{2.29}$$

where k is the transition kernel, given in theorem 2.6.

If transition kernel is time-invariant, we define the Time-Homogeneous Path Integration Operator as

$$u_t = PI_{\Delta t}u_0 \int_{\mathbb{R}^n} k(y, x, \Delta t)u_0(x)\, \mathrm{d}x \tag{2.30}$$

Typically, we find an approximation for the distribution of the solution at time T, by dividing the set $[0, T]$ into subsets between the points $0 = t_0 < t_1 < \cdots < t_n = T$, and iterate

$$u_{t_i} = PI_{t_i,t_{i-1}}u_{t_{i-1}} \tag{2.31}$$

Now it should be noted that if the points are equidistant, that is $t_{i+1} - t_i = \Delta t$ for all $i = 0, 2, \ldots, n-1$, and if the functions a and b in equation (2.9) do not depend on time explicitly, then the PI-operator becomes time-homogeneous, and in particular, it means that we only need to calculate the transition kernel once, and can use it at every iteration. For a SDE not depending explicitly on time, this can be very useful, and will be explored in later chapters

As it stands, we have said very little about the PI-operator other than stating what it is. But there are properties the PI-operator is desired to have. For example, if f is some distribution, we would like the PI-operator to preserve this property. In particular, the function must be everywhere non-negative, and its norm must be one.

Definition 2.9. Let $(X, \mathcal{F}, \mu)$ be a measure space, then a linear operator $P : L^1(X) \to L^1(X)$ is said to be a Markov Operator if the following properties are satisfied

- $Pf \geq 0$ for all $f \in L^1(\mathbb{R}^n)$ satisfying $f \geq 0$
- $\|Pf\|_1 = \|f\|_1$ for all $f \in L^1(\mathbb{R}^n)$ satisfying $f \geq 0$

Theorem 2.10. *The PI-operator is a Markov Operator*

Proof. By linearity of an operator, we mean that for any functions $f, g \in L^1(\mathbb{R}^n)$,

$$P(f + \alpha g) = Pf + \alpha Pg, \text{ for all } \alpha \in \mathbb{C} \tag{2.32}$$

- The linearity property follows directly from the fact that function multiplication and integration are linear mathematical operators.
- If $f(x) \geq 0$, then $k(y, x, t', t)f(x) \geq 0$, and a basic property of the Lebesgue integral is then that $PI_{t',t} = \int_{\mathbb{R}^n} k(y, x, t', t)f(x)\,\mathrm{d}x \geq 0$.
- Note first that $k(y, x, t', t) \in L^1_x\mathbb{R}^n$ for all $y \in \mathbb{R}^n$ and $t \neq t'$, so from Fubini's theorem[6], we can exchange the order of integration:

$$\|PI_{t',t}f\| = \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} k(y, x, t', t)f(x)\,\mathrm{d}x\,\mathrm{d}y \tag{2.33}$$

$$= \int_{\mathbb{R}^n} f(x)\left(\int_{\mathbb{R}^n} k(y, x, t', t)\,\mathrm{d}y\right)\,\mathrm{d}x, \tag{2.34}$$

  note that the mapping $y \to k(y, x, t', t)$, where k is as defined in theorem 2.6, is a distribution for all values of $x \in \mathbb{R}^n$ and $t' \neq t$, so $\int_{\mathbb{R}^n} k(y, x, t', t)\,\mathrm{d}y = 1$, thus

$$\|PI_{t',t}f\|_1 = \int_{\mathbb{R}^n} f(x)\,\mathrm{d}x = \|f\|_1. \tag{2.35}$$

□

So the PI-operator does preserve distributions, and in addition we also saw that it is linear. This is a very nice property, that allows us to divide the state space into subsets and use the PI-operator separately on each subset. To see this, let $\{C_i\}$ be a countable family of Borel-measurable subsets of $\mathbb{R}^n$, then we can represent the function f by $f(x) = \sum_i \mathbf{1}_{C_i}(x)f(x)$, where $\mathbf{1}$ is the indicator function

$$\mathbf{1}_{C_i}(x) = \begin{cases} 1 & : x \in C_i \\ 0 & : x \notin C_i \end{cases} \tag{2.36}$$

Denote by $f_n(x)$ the sequence of partial sums $f_n(x) = \sum_{i=1}^n \mathbf{1}_{C_i}(x)f(x)$, then clearly $f_n(x) \to f(x)$ as $n \to \infty$,$f_n(x) \leq f(x) \in L^1(\mathbb{R}^n)$ with $f \geq 0$ for all $n \in \mathbb{Z}$, and $f_n(x) \in L^1(\mathbb{R}^n)$ for all $n \in \mathbb{Z}$ as $\mathbf{1}_{C_i} \in L^1(\mathbb{R}^n)$ for all Borel-measurable sets C_i. Thus, by the Dominated Convergence Theorem we have

$$PI_{t',t}f = PI_{t',t}\left(\lim_{n\to\infty} f_n\right) \tag{2.37}$$

$$= \lim_{n\to\infty} PI_{t',t}f_n \tag{2.38}$$

$$= \lim_{n\to\infty} PI_{t',t}\left(\sum_{i=1}^n \mathbf{1}_{C_i}(x)f(x)\right) \tag{2.39}$$

$$= \sum_i PI_{t',t}(1_{C_i}(x)f(x)) = \sum_i \int_{C_i} k(y, x, t', t)f(x)\,\mathrm{d}x. \tag{2.40}$$

In particular, for each time-step we can calculate each term in the sum independently of each other, which is great for parallel-computing.

The concept of semi-groups is slightly more general, and can be defined as

Definition 2.11. Let $(X, \mathcal{F}, \mu)$ be a measure space. The family of stochastic operators $P_t : L^1(X) \to L^1(X)$ is said to form a Stochastic Semi-Group, if the operator P_t is a Markov Operator for each $t \geq 0$ and in addition

$$P_{t+t'}f = P_t(P_{t'}f), \text{ for all } f \in L^1(X). \tag{2.41}$$

If in addition

$$\lim_{t \to t'} \|P_t f - P_{t'} f\|_1 = 0 \tag{2.42}$$

the family of operators is called a Continuous Stochastic Semi-Group.

The time-homogeneous PI-operator forms a stochastic semi-group. We have already shown that it is a Markov operator, the property that $PI_{t+t'}f = PI_t(PI_{t'}f)$ follows directly from the Chapman-Kolmogorov equation, and by using Fubini's theorem once again to exchange order of integration

$$PI_{t+t'}f = \int_{\mathbb{R}^n} k(y, x, t', t) f(x) \, \mathrm{d}x = \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} k(y, z, t') k(z, x, t) \, \mathrm{d}z \right) f(x) \, \mathrm{d}x \tag{2.43}$$

$$= \int_{\mathbb{R}^n} k(y, z, t') \left(\int_{\mathbb{R}^n} k(z, x, t) f(x) \, \mathrm{d}x \right) \mathrm{d}z = PI_{t'}(PI_t f). \tag{2.44}$$

No that we have derived the PI-operator, the governing equation and seen that the operator behaves as we want it, we take a look at the practical challenges of implementing Path Integration numerically on a computer.

2.4 Implementation of Path Integration

Now that the theoretical groundwork the Path Integration has been laid, we will discuss more practical consideration when implementing this method on a computer which have finite memory and finite computation speed.

2.4.1 Calculating integrals of unbounded sets on a computer

We start by looking at the PI-operator

$$PI_{t',t}f = \int_{\mathbb{R}^n} k(y, x, t', t) f(x) \, \mathrm{d}x. \tag{2.45}$$

To be able to compute this in finite time on a computer, we need to restrict the above integral to some subset of $\mathbb{R}^n$, ideally we want to have a manageable subset that covers most of the area where $k(y, x, t', t) f(x) > \epsilon$ for some small value of ϵ.

If we consider the case where there is noise in only one dimension, that is, the dimension of the matrix $\tilde{g}(x, t)$ in equation (2.13) is 1×1, then the transition kernel for the corresponding Markov chain becomes

$$k(x', x, t', t) = \prod_{i=1}^{n-1} \delta(x'_i - x_i - r_i(a, x_i, t)\Delta t) \cdot \tilde{d}_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x', t', x, t) \tag{2.46}$$

where

$$\tilde{d}_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x',t',x,t) = \frac{1}{\sqrt{2\pi\Delta t\tilde{g}(x,t)}} \exp\left[-\frac{(x'_n - x_n - r_n(a,x,t)\Delta t)^2}{2\Delta t\tilde{g}(x,t)}\right] \quad (2.47)$$

We see that $\tilde{d}_{X_{(n+1)\Delta t}|X_{n\Delta t}}(x',t',x,t)$ is Gaussian with variance $\sigma^2 = \tilde{g}(x,t)\Delta t$. If $g(x,t)$ is independent of x and t, say $g(x,t) = C < \infty$, then we can say by looking at a standard table giving probabilities for normal distributions that

$$P(|X-\mu| < 3\sigma) \approx 0.998 \quad (2.48)$$

In other words, the error when calculating $\tilde{I} = \int_{\mu-3\sigma}^{\mu+3\sigma} d_X(x)\,\mathrm{d}x$ as opposed to the exact integral $I = \int_{\mathbb{R}} d_X(x)\,\mathrm{d}x$ is

$$|\tilde{I} - I| \approx 0.002, \quad (2.49)$$

which should be small enough for most practical considerations.

We assume that $k(y,x,t',t)f(x)$ does not decay much slower to zero than $k(y,x,t',t)$, which will most of the time be true for Path Integration, as the solution at the previous time step often is bounded and has exponential decay at the tails. To be even more sure, we can make the length of the integration interval 10 times the variance of k(y,x,t',t), rather than 6.

Thus, as a rule of thumb, the integration interval $[\mu - 5\sigma, \mu + 5\sigma]$ will be used when calculating the integral given in the PI-operator, when there is noise in one dimension, and the variance of the normal distribution part is constant.

Suppose that $\tilde{g}(x,t)$ is not independent of x and t, but that $\sup_x |g(x,t)|$ exists for each t, and is finite. Then we can set $\sigma^2 = \sup_x |g(x,t_n)|$ for each time-step t_n and apply the same integration interval as above.

If $\tilde{g}(x,t)$ is not a 1×1 matrix, but is diagonal, and for each diagonal element $g_{ii}(x,t)$, $\sup_x |g_{ii}(x,t)$ exists and is finite for each t. Then we can set the $\sigma_i^2 = \sup_x |g_{ii}(x,t)|$, and let the integration area be the smallest box that contains the n-ellipsoid with radii $r_i = 5\sigma_i$.

The other cases gets more complicated, and will not be discussed in this thesis.

2.4.2 Numerical Integration

Just as we can not consider the whole space $\mathbb{R}^n$ when calculating a computer, we can not consider non-finite subsets of points of $\mathbb{R}^n$ either. When we have chosen an appropriate bounded subset to integrate over, we must choose some appropriate finite set of points $\{x^i\}$ to represent the bounded subset. Typically, we divide the bounded subset, which will often be a box, into a grid, and choose the set of points $\{x_i\}$ where the lines intersects appropriately.

Consider now the transition kernel

$$k(y,x,t',t) = \prod_{i=1}^{r} \delta(x'_i - x_i - r_i(a,x,t)\Delta t) \cdot \tilde{d}_{(n+1)\Delta t}(x',t',x,t) \quad (2.50)$$

where $\tilde{d}_{(n+1)\Delta t}(x',t',x,t)$ is as in equation (2.16) as before. The distribution of the solution at the next timestep t' given the distribution of the solution $p(x,t)$ at the previous time step then becomes

$$p(x',t') = PI_{t',t}p(x,t) = \int_D \tilde{1}d_{(n+1)\Delta t}(x',t',x,t)p(x,t)\,\mathrm{d}x, \tag{2.51}$$

where $D = \{(x_{r+1},\dots,x_n) \in \mathbb{R}^{n-r}$: there exists $(x_1,\dots,x_r) \in \mathbb{R}^r$ such that $x_i' = x_i + r_i(a,x,t)\Delta t)$ for each $i = 1,\dots,r\}$. But to find the the set D is very impractical, difficult, and in some cases impossible. A better approach, is to do a substitution $\tilde{x} = x + r(a,x,t)\Delta t$, which removes the dependency of the integrals, and gives

$$p(x',t') = \int_{\mathbb{R}^{n-r}} \tilde{d}_{(n+1)\Delta t}(x',t',\tilde{x},t)p(g^{-1}(x_1',x_2'\dots,x_r',\tilde{x}_{r+1},\dots,\tilde{x}_n),t)|J_{g^{-1}}|\,\mathrm{d}x, \tag{2.52}$$

where $g^{-1}(x_1',x_2',\dots,x_r',\tilde{x}_{r+1},\tilde{x}_n)$ is the unique vector $x = (x_1,x_2,\dots,x_n)$ such that $(x_1',x_2',\dots,x_r',\tilde{x}_{r+1},\tilde{x}_n) = x + r(a,x,t)\Delta t$, $|J_{g^{-1}}|$ is the determinant of the Jacobian of $g^{-1}(x_1',x_2'\dots,x_r',\tilde{x}_{r+1},\dots,\tilde{x}_n),t)$, where the Jacobian is defined as

$$J_{g^{-1}} = \begin{bmatrix} \frac{\partial g_1^{-1}}{\partial x_1} & \frac{\partial g_1^{-1}}{\partial x_2} & \cdots & \cdots & \frac{\partial g_1^{-1}}{\partial x_n} \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ & & & & \\ \vdots & & & \ddots & \vdots \\ \frac{\partial g_n^{-1}}{\partial x_1} & \cdots & \cdots & \cdots & \frac{\partial g_n^{-1}}{\partial x_n} \end{bmatrix} \tag{2.53}$$

and

$$\tilde{d}_{(n+1)\Delta t}(x',t',\tilde{x},t) = \frac{1}{(2\pi\Delta t)^{(n-r)/2}|\tilde{g}(g^{-1}(x_1',\dots,x_r',\dots,\tilde{x}_n,),t)|^{1/2}}$$
$$\exp\left[\frac{1}{2\Delta t}\sum_{i=r+1}^{n}\sum_{j=r+1}^{n}(x_i'-\tilde{x}_i)[\tilde{g}(g^{-1}(x_1',\dots,x_r',\dots,\tilde{x}_n),t)^{-1}]_{i-r,j-r}(x_j'-\tilde{x}_j)\right]. \tag{2.54}$$

The main advantage of this substitution is to remove the dependency of the integration area that the delta functions enforce, by making the $\{\tilde{x}_i\}$ independent of each other in the delta function, which the $\{x_i\}$ were not, such that we do not need to integrate over the arbitrary, problem dependent set D. The second advantage is a transition which has a simpler form, which becomes easier to use if one wants to choose quadrature points, according to the transition kernel.

This substitution makes clear another challenge. We need to calculate $p(g^{-1}(x_1',x_2'\dots,x_r',\tilde{x}_{r+1},\dots,\tilde{x}_n),t)$ where $g^{-1}(x_1',x_2'\dots,x_r',\tilde{x}_{r+1},\dots,\tilde{x}_n)$ generally is not in the grid, for which we store function values for p. A method to solve this, would be to approximate $g^{-1}(x_1',x_2'\dots,x_r',\tilde{x}_{r+1},\dots,\tilde{x}_n)$ by choosing the point in the grid which is closest to $g^{-1}(x_1',x_2'\dots,x_r',\tilde{x}_{r+1},\dots,\tilde{x}_n)$ in the $\|\cdot\|_2$-norm sense.

Let $\{\tilde{x}^i\}$ denote the set of points that solve $x'^{,i} = x + r(a,\tilde{x}^i,t)\Delta t$ for each $x'^{,i}$. In order to integrate, we need $p(\tilde{x}^i,t)$ for each i. These points will generally not lie in the chosen grid however, and unless we choose a really dense grid, then approximations by $p(\tilde{x}^i,t) \approx p(x^j,t)$ where x^j is the closest point in grid, will generally be poor.

A better approach, is to use the values we already have for p at time t at the points $\{x^i\}$ to interpolate p at $\{\tilde{x}^i\}$. The interpolation method we will use, is *basis splines* (B-splines), which will be discussed in section 2.4.4.

2.4.3 Methods of integration

As noted earlier, we cannot in general calculate the integral

$$p(x',t') = \int_{\mathbb{R}^{n-r}} \tilde{d}_{(n+1)\Delta t}(x',t',\tilde{x},t)p(g^{-1}(x'_1,x'_2\dots,x'_r,\tilde{x}_{r+1},\dots,\tilde{x}_n),t)|J_{g^{-1}}|\,\mathrm{d}x, \tag{2.55}$$

analytically. We need to restrict the integral to a bounded subset of $\mathbb{R}^{n-r}$ and even then we cannot calculate the integral analytically, because we only know the function value at a finite number of points within the bounded subset.

We therefore generally use quadrature rules, which given points $a \leq x_1 \leq \cdots \leq x_n \leq b$ approximate the integral of $f(x)$ from $x = a$ to $x = b$, by the sum

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=1}^n w_i f(x_i) \tag{2.56}$$

where w_i are finite weights, which generally have the property $\sum_{i=1}^n w_i = b - a$.

There are many different ways of both choosing the weight and the quadrature points in order to obtain different degrees of precision, but for ease we will stick with equidistant quadrature points and a couple of basic quadrature rules. In particular, the quadrature we will be looking at, are all based on polynomial interpolation.

The first, and simplest rule of interpolation, is to simply assume that $f(x)$ is approximately equal to $f(a)$ over the interval $[a,b]$, which leads to the quadrature rule

$$\int_a^b f(x)\,\mathrm{d}x \approx \int_a^b f(a)\,\mathrm{d}x = f(a)(b-a) \tag{2.57}$$

if the function $f(x)$ actually is constant of the interval $[a,b]$ that is, it is a 0-degree polynomial, then naturally, the above formula calculates the integral of the function exactly, but if f is not constant over the interval, the above formula does not necessarily give the exact value of the integral. We therefore say that this quadrature rule is of precision 0.

The second rule, which is commonly known as the trapezoidal rule, and it is derived from approximating the function f by linear interpolation between $f(a)$ and $f(b)$, that is

$$f(x) \approx \frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b) \tag{2.58}$$

which leads to the quadrature rule

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \int_a^b \frac{x-b}{a-b}f(a) + \frac{x-a}{b-a}f(b)\,\mathrm{d}x \\ &= \frac{1}{2}\left[\frac{(x-b)^2}{a-b}f(a) + \frac{(x-a)^2}{b-a}f(b)\right]_a^b \\ &= \frac{b-a}{2}(f(a)+f(b)).\end{aligned} \tag{2.59}$$

As a linear function is uniquely defined by two function values, the polynomial interpolation of any linear function, that is, first order polynomial is exact. Thus, the above formula also exactly integrates any first order polynomial over $[a,b]$. But it does not necessarily integrate exactly a polynomial of any higher degree, therefore we say that this quadrature rule is of precision 1.

The third and final rule which will be presented here, is commonly known as Simpson's rule. This rule is based on interpolation of the the function f by a second order polynomial, using $f(a)$, $f((a+b)/2)$ and $f(b)$ by

$$f(x) \approx 2\frac{(x-\frac{a+b}{2})(x-b)}{(a-b)^2}f(a) + 4\frac{(x-a)(x-b)}{(b-a)(a-b)}f\left(\frac{a+b}{2}\right) + 2\frac{(x-a)(x-\frac{a+b}{2}}{(b-a)^2}f(b) \tag{2.60}$$

With an easy but longer calculation, it can be shown that this leads to the quadrature rule

$$\begin{aligned}&\int_a^b f(x)\,\mathrm{d}x \\ &\approx \int_a^b 2\frac{(x-\frac{a+b}{2})(x-b)}{(a-b)^2}f(a) + 4\frac{(x-a)(x-b)}{(b-a)(a-b)}f\left(\frac{a+b}{2}\right) + 2\frac{(x-a)(x-\frac{a+b}{2}}{(b-a)^2}f(b)\,\mathrm{d}x \\ &= \frac{b-a}{6}\left[f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right]\end{aligned} \tag{2.61}$$

It can be shown that, not only does Simpson's rule integrate second degree polynomials, as one might expected, but it also integrates third degree polynomials exactly. But it does not necessarily integrate exactly polynomials of degree 4 or higher, so Simpson's rule is said to be of precision 3.

These quadrature rules are good, but become imprecise when $b-a>1$, we therefore often divide the interval $[a,b]$ into smaller sub-intervals restricted by the points $a = x_1 < x_2 < \cdots < x_n = b$, where they are equidistant: $x_{i+1} - x_i = h$ for $i = 1, \ldots, n-1$ and we preferably have that $h << 1$. We can thus write

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{i=1}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\,\mathrm{d}x \tag{2.62}$$

By applying the quadrature rules on each of these sub integrals, we get *composite quadrature rules.* For the first mentioned rule, it becomes

$$\int_a^b f(x)\,\mathrm{d}x \approx h\sum_{i=1}^{n-1} f(x_i). \tag{2.63}$$

For the trapezoidal rule, we get

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{2}(f(x_1) + 2f(x_2) + 2f(x_3) + \cdots + 2f(x_{n-1}) + f(x_n)). \tag{2.64}$$

And finally, for Simpson's rule, we get, provided that $[a,b]$ is divided into an even number of sub-intervals

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3}(f(x_1) + 4f(x_2) + 2f(x_3) + 4f(x_4) + \cdots + 4f(x_{n-1}) + f(x_n)) \tag{2.65}$$

Note that h in (2.65) corresponds to $2h$ in (2.61), we therefore get 3 in the denominator instead of 6 in the composite rule of Simpson's rule.

Because of no added cost of calculation, and the high precision, we will stick with Simpson's rule for calculating the integral in equation (2.55). In the following example we will present a rule of thumb for the amount of quadrature point for a certain type of SDE

Example 2.12. Consider the SDE

$$\mathrm{d}X_t = a_1(X_t, Y_t, t)\,\mathrm{d}t \tag{2.66a}$$

$$\mathrm{d}Y_t = a_2(X_t, Y_t, t)\,\mathrm{d}t + \sqrt{D_\xi}\mathrm{d}B_t \tag{2.66b}$$

where D_ξ is a positive constant. The solution at time t' then becomes, according to equation (2.55)

$$p(x', y', t') = \int_{\mathbb{R}} \frac{\exp\left(-\frac{(y'-\tilde{y})^2}{2\Delta t D_\xi}\right)}{\sqrt{2\pi\Delta t D_\xi}} p(g^{-1}(x', \tilde{y}, t))|J_{g^{-1}}|\,\mathrm{d}\tilde{y} \tag{2.67}$$

where g^{-1} is, as usual, the inverse of the transformation $(\tilde{x}, \tilde{y}) = (x + r_1(x, y, t)\Delta t, y + r_2(x, y, t))$, where r is the RK-step vector for the SDE, and $|J_{g^{-1}}|$ is the determinant of the Jacobian of the inverse of the transformation.

Assuming that the assumption that the transition kernel dominates the behavior of the integral, we earlier presented the rule of thumb for the integration interval, where we chose the length of the integration interval to be 10 times the size of the variance. In other words, we integrate over $[y' - 5\sqrt{\Delta t D_\xi}, y' + 5\sqrt{\Delta t D_\xi}]$ where $\Delta t D_\xi$ in this case is the variance. In a similar manner, we want to find a rule of thumb for the number of equidistant points when numerically integrating over this interval.

By introducing the linear transformation $y = (y' - \tilde{y})/\sqrt{\Delta t D_\xi}$, we obtain from the integral

$$\int_{y'-5\sqrt{\Delta t D_\xi}}^{y'+5\sqrt{\Delta t D_\xi}} \frac{1}{\sqrt{2\pi\Delta t D_\xi}} \exp\left(-\frac{(y'-y)^2}{2\Delta t D_\xi}\right)\mathrm{d}x = \int_{-5}^{5} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right)\mathrm{d}x \tag{2.68}$$

which is the standard normal distribution with mean value 0 and variance equal to 1. Since the transformation this transformation only translates and magnifies the solution, we do not change any of the basic properties of the function, and finding a good number of quadrature point for the standard normal distribution, should therefore be equally good for any other normal distribution unless $\sqrt{\Delta t D_\xi}$ is a very small number.

Since the value of the integral in equation (2.68) ideally should be close to 1, we define the integration error in calculating the numerical approximation using the composite trapezoidal rule with a number n of equidistant quadrature points, as the absolute value of the difference of the calculated approximation and 1.

Table 2.1 shows the progression of the error when using an increasing number of quadrature nodes on the standard normal distribution, while integrating on the interval, and as can be seen, for $n > 25$, not much is gained in terms of accuracy by having more quadrature points. So as to not use an excessive amount of quadrature points, choosing $n = 25$ quadrature points seems to be a good rule of thumb, which at the same time provides high accuracy.

n	Error
9	0.028
13	$5.48 \cdot 10^{-4}$
17	$2.88 \cdot 10^{-6}$
25	$6.12 \cdot 10^{-7}$
29	$5.96 \cdot 10^{-7}$
33	$5.87 \cdot 10^{-7}$

Table 2.1: The table shows integration error in example 2.12 as a function of the number of quadrature points, used for the numerical integration. The error is represented by the absolute value difference between the calculated values of the integral from the composite Simpson's rule and 1.

2.4.4 B-splines

In one dimension, a basis spline is formally constructed from a set of points $x_0 < x_1 < \ldots x_m \in \mathbb{R}$ called knots. Associated to these knots, we have numbers $\{P_0, P_1, \ldots, P_{m-n-1}$ called control points. A B-spline is then a function

$$S(x) = \sum_{i=0}^{m-n-1} P_i b_{i,n}(x) \tag{2.69}$$

where $b_{i,n}$ is the basis polynomial-spline associated with P_i of degree n, given by the recursive formula

$$b_{i,0} = \begin{cases} 1 & : x_i \le x \le x_{i+1} \\ 0 & : \text{ otherwise} \end{cases} \tag{2.70}$$

$$b_{i,n} = \frac{t - t_i}{t_{i+n} - t_i} b_{i,n-1}(x) + \frac{x_{i+n+1} - x}{x_{i+n+1} - x_{i+1}} b_{i+1,n-1}(x) \tag{2.71}$$

given a uniform sequence of knots $-m\Delta x < -(m+1)\Delta xz < \cdots < (n-1)\Delta x < n\Delta x$, it can be seen that the cubic basis-splines with $n = 3$, becomes

$$b_{i-2,3} = \begin{cases} \frac{1}{6}\left(2 + \left(\frac{x-i\Delta x}{\Delta x}\right)\right)^3 & : (i-2)\Delta x \le x \le (i-1)\Delta x \\ \frac{1}{6}\left(4 - 6\left(\frac{x-i\Delta x}{\Delta x}\right)^2 - 3\left(\frac{x-i\Delta x}{\Delta x}\right)^3\right) & : (i-1)\Delta x \le x \le i\Delta x \\ \frac{1}{6}\left(4 - 6\left(\frac{x-i\Delta x}{\Delta x}\right)^2 + 3\left(\frac{x-i\Delta x}{\Delta x}\right)^3\right) & : i\Delta x \le x \le (i+1)\Delta x \\ \frac{1}{6}\left(2 - \left(\frac{x-i\Delta x}{\Delta x}\right)\right)^3 & : (i+1)\Delta x \le x \le (i+2)\Delta x \end{cases} \tag{2.72}$$

For ease of notation, and since we shall only use cubic splines, we shall henceforth denote $b_i(x) = b_{i-2,3}(x)$.

Suppose that for a function $f(x)$ we have sampled the points $\{f(-m\Delta x), f((-m+1)\Delta x), \ldots, f((n-1)\Delta x), f(n\Delta x)\}$ and want $S(x)$ to interpolate $f(x)$ at the sampled points, then it becomes clear from the definition of $S(x)$ and $b_i(x)$ that we need to solve a linear system of equations

$$\begin{aligned}
&P_{-m}b_{-m}(-m\Delta x) + P_{-m+1}b_{-m+1}(-m\Delta x) = f(-m\Delta x)\\
&P_{-m+1}b_{-m+1}((-m+1)\Delta x) + P_{-m+1}b_{-m+1}((-m+1)\Delta x)\\
&+ P_{-m+2}b_{-m+2}((-m+1)\Delta x) = f((-m+1)\Delta x)\\
&\vdots\\
&P_{n-2}b_{n-2}((n-1)\Delta x) + P_{n-1}b_{n-1}((n-1)\Delta x) + P_n b_n((n-1)\Delta x) = f((n-1)\Delta x)\\
&P_{n-1}b_{n-1}(n\Delta x) + P_n b_n(n\Delta x) = f(n\Delta x)
\end{aligned}$$

Evaluating the function $b_i(x)$ at knots, and representing the system of equations as a matrix, we get:

$$Mp = \frac{1}{6}\begin{bmatrix} 4 & 1 & 0 & \cdots & \cdots & 0\\ 1 & 4 & 1 & 0 & \cdots & \vdots\\ 0 & 1 & 4 & 1 & \ddots & \vdots\\ \vdots & 0 & \ddots & \ddots & \ddots & 0\\ \vdots & \cdots & \ddots & 1 & 4 & 1\\ 0 & \cdots & \cdots & 0 & 1 & 4 \end{bmatrix}\begin{bmatrix} P_{-m}\\ P_{-m+1}\\ \vdots\\ \vdots\\ P_{n-1}\\ P_n\end{bmatrix} = \begin{bmatrix} f(-m)\\ f(-m+1)\\ \vdots\\ \vdots\\ f(n-1)\\ f(n)\end{bmatrix}$$

Since A is strictly diagonal-dominant, it can easily be established from Gershgorin's theorem that M is positive definite, and thus invertible. Thus we can find an unique set control points for the cubic B-spline $S(x)$, such that $S(x)$ interpolates $f(x)$.

2.4.5 Implementation of Basis-splines in $\mathbb{R}^2$

It is not immediately obvious, how to extend the interpolation of cubic B-splines from $\mathbb{R}$ to $\mathbb{R}^2$ in a sensible way: We should find a way to compute the spline coefficients from function data, without too much computations. Chapter 17 in [7] provides a quite long exposition of this, by generalizing and making abstract the notion of interpolation. We will, however, only need one result from this chapter, which is explained in the following paragraph.

Given some knots defined in the grid $y_1 \times y_2$, where y_1 is a vector, defined element-wise by $y_1(i) = i\Delta x$ for $i = -m_1, -m_1+1, \ldots, n_1-1, n_1$ and y_2 is a vector defined element-wise by $y_2(i) = i\Delta x$ for $i = -m_2, -m_2+1, \ldots, n_2-1, n_2$, with a corresponding $(m_1+n_1+1) \times (m_2+n_2+1)$-matrix Γ of spline coefficients. Let the cubic B-spline on these knots be defined by

$$S(y_1, y_2) = \sum_{i,j}\Gamma(i,j)b_i \otimes b_j = \sum_{i,j}\Gamma(i,j)b_i(y_1)b_j(y_2) \tag{2.73}$$

where $b_i(x)$ are the cubic basis-splines defined in the previous section. Furthermore, let A and B be matrices defined by $A(i,j) = b_j(y_1(i))$ for $i,j = 1,\ldots,m_1+n_1+1$ and $B(i,j) = b_j(y_2(i))$ for $i,j = 1,\ldots,m_2+n_2+1$, and let L_S be the matrix of values of equation (2.73) at the knots. Then we have the following relationship between the matrix Γ and the matrix L_S

$$L_S = A\Gamma B^T$$

This can be shown directly by writing up

$$\begin{aligned} L_S(r,s) = S(y_1(r), y_2(s)) &= \sum_{i,j} \Gamma(i,j) b_i(y_1(r)) b_j(y_2(s)) \\ &= \sum_{i,j} \Gamma(i,j) A(r,i) B(s,j) \\ &= (A\Gamma B^T)(r,s), \text{ for all } r, s. \end{aligned}$$

The last equation needs some thought to see, but can easily be verified by constructing the matrix-product. In particular, we can retrieve the spline coefficients from the values we wish the B-spline to have at the knots, simply by calculating: $\Gamma = A^{-1} L_S B^{-T}$, if such inverse matrices exists.

For simplicity, let $m = m_1 + n_1 + 1$ and $n = m_2 + n_2 + 1$, we see that A and B, such that they are defined, for the basis-splines defined in the previous section, are simply $m \times m$ and $n \times n$ versions, respectively, of the matrix M also defined in the previous section. That is, the tri-diagonal matrix with the value $\frac{4}{6}$ on the diagonal, and the value $\frac{1}{6}$ on both the sub-diagonal, and 0 elsewhere. As we noted earlier, this matrix is always positive definite, so A^{-1} and B^{-1} naturally exists.

The matrix Γ can be calculated from $L_S = A\Gamma B^T$ in the following way: Set $X = \Gamma B^T$, and let $L_S^{(j)}$ and $X^{(j)}$ be the jth column of L_S and X, then we can obtain the matrix X by solving

$$AX^{(j)} = L_S^{(j)}, \text{ for } j = 1, \dots, n$$

After we obtain X, we have $\Gamma B^T = X$, or equivalently: $B\Gamma^T = X^T$, so by letting $X_{(i)}$ and $\Gamma_{(i)}$ be the ith row of X and Γ, then we finally obtain Γ by solving

$$B\Gamma^T_{(i)} = X^T_{(i)}, \text{ for } i = 1, \dots, m$$

There is much to gain by calculating the spline matrix in such a manner, as opposed to calculating them the "brute-force" method by solving the linear system of equations

$$\sum_{i,j} \Gamma(i,j) b_i(y_1(r)) b_j(y_2(s)) = L_S(r,s), \text{ for } r = 1, \dots, m \ , \ s = 1, \dots, n.$$

This is a system of nm equations with nm unknowns, which results in a $nm \times nm$-matrix, which, in order to solve using Gauss-elimination, requires $\mathcal{O}((nm)^3)$ operations. In contrast, the above calculations, requires us to solve a $m \times m$ system of equations n times, and then a $n \times n$ system of equations m times, which leads to a total cost of $\mathcal{O}(m^3 n + n^3 m)$, which is substantially better. This is not even the best we could do, we could produce an LU-factorization of the matrices A and B, which costs $\mathcal{O}(m^3 + n^3)$, however for the subsequent calculations, we only need $\mathcal{O}(m^2 n + n^2 m)$ calculations, which brings us up to a total cost of $\mathcal{O}(m^3 + m^2 n + n^2 m + n^3)$.

2.5 Implementation of Runge-Kutta

In calculating the PI-operator we need to calculate $g^{-1}(x'_1, x'_2, \dots, \tilde{x}_{r+1}, \dots, \tilde{x}_n)$, which is the unique vector $x = (x_1, x_2, \dots, x_n)$ such that $x + r(a, x, t)\Delta t = (x'_1, x'_2, \dots, x'_r, \tilde{x}_{r+1}, \dots, \tilde{x}_n)$

for the relevant SDE. x can be approximated by using a root finding algorithm, such as Newton's method for example. It is however important to note, that even if we find x exactly, then this is still just an approximation to the stochastic path governed by the SDE that begins in some point $x'' \in \mathbb{R}^n$ at time t, and end in $(x'_1, x'_2, \ldots, \tilde{x}_{r+1}, \ldots, \tilde{x}_n)$ at time t'. If the order of the Runge-Kutta method used is p, then the error is in the order of $\mathcal{O}((\Delta t)^{p+1})$.

We can however do better than this, by exploiting the simple fact, that for the general explicit Runge-Kutta method of order p:

$$x' = x_{t+\Delta t} = x_t + \Delta t \sum_{i=1}^{s} b_i k_i(x_t, \Delta t, k_1, \ldots, k_{i-1}) + \mathcal{O}((\Delta t)^{p+1})$$

the above equation is valid for any Δt. In particular, it is valid for negative time-steps, so by replacing y_t with y and Δt with $-\Delta t$, we obtain

$$x = x_{t-\Delta t} = x' - \Delta t \sum_{i=1}^{s} b_i k_i(x', -\Delta t, k_1, \ldots, k_{i-1}) + \mathcal{O}((\Delta t)^{p+1}).$$

This gives a method that is implicit for steps forward in time, but explicit for y', thus we retain the same order of convergence, without the need to solve any non-linear system of equations! Two different RK-schemes have been implemented, which are given below, in where we in both cases assume a standard SDE with deterministic term $a(x,t)$.

2.5.1 Euler's method

Euler's method, which is the simplest RK-method, becomes when we step backwards in time

$$x_n = x_{n+1} - a(x_{n+1}, t_{n+1})\Delta t + \mathcal{O}(\Delta t^2)$$

2.5.2 RK4

RK4 is perhaps the most used RK-method, and is very accurate, as it is a fourth order method. The idea is to first calculate the slope at the known solution, successively get to estimates for the slope in the middle of the interval time interval between the known solution and the unknown solution from this, and then finally get an estimate for the slope at the unknown solution. From this slopes, an weighted average is produced, where extra weight is given to the two mid-point estimates. For the backwards-stepping method, we calculate

$$\begin{aligned}
k_1 &= a(x_{n+1}, t_{n+1}) \\
k_2 &= a(x_{n+1} - \frac{1}{2}k_1\Delta t, t_{n+1} - \frac{1}{2}\Delta t) \\
k_3 &= a(x_{n+1} - \frac{1}{2}k_2\Delta t, t_{n+1} - \frac{1}{2}\Delta t) \\
k_4 &= a(x_{n+1} - k_3\Delta t, t_n) \\
y_n &= x_{n+1} - \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$

2.6 The Fokker-Planck Equation

The *Fokker-Planck equation* describes physically the probability density of the velocity of a particle, and is due to Adriaan Fokker and Max Planck. The equation is also known under the name *Kolmogorov forward equation.*

In one spatial dimension x, when the particle is driven by an Itô drift $D_1(x,t)$ and diffusion $D_2(x,t)$, the Fokker-Planck equation can mathematically be given by

$$\frac{\partial}{\partial t}f(x,t) = -\frac{\partial}{\partial x}[D_1(x,t)f(x,t)] + \frac{\partial^2}{\partial x^2}[D_2(x,t)f(x,t)]. \tag{2.74}$$

This can be extended to more dimensions, when $x = (x_1, \dots, x_n) \in \mathbb{R}^n$ and the equation is given by

$$\frac{\partial}{\partial t}f(x,t) = -\sum_{i=1}^{n}\frac{\partial}{\partial x_i}[D_i^1(x)f(x)] + \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2}{\partial x_i \partial x_j}[D_{ij}^2(x)f(x)], \tag{2.75}$$

where $D^1 : \mathbb{R}^n \to \mathbb{R}^n$ is the drift vector and $D^2 : \mathbb{R}^n \to \mathrm{MAT}(n,m)$ is the diffusion matrix.

One great advantage of the Fokker-Planck equation is that it can be used to compute the probability density for a stochastic process described by a stochastic differential equation (interpreted in the Itô-sense)

$$\mathrm{d}X_t = a(X_t,t)\,\mathrm{d}t + b(X_t,t)\,\mathrm{d}B_T \tag{2.76}$$

where the solution X_t, if the problem is well posed, satisfies the Fokker-Planck equation with drift and diffusion terms

$$D_i^1(x,t) = a_i(x,t) \tag{2.77a}$$

$$D_{ij}^2(x,t) = \frac{1}{2}\sum_k b_{ik}(x,t)b_{jk}(x,t), \tag{2.77b}$$

so the time-depended distribution of the solution can sometimes be directly calculated from the Fokker-Planck equation. It is mainly used to calculate time-independent (stationary) solutions, however, which is readily given by the Fokker-Planck equation by setting $\frac{\partial}{\partial t}f(x,t) = 0$.

In the examples that follows, we shall look at some of the stationary solutions as well as explicit time dependent solutions that exists for various stochastic differential equations

Example 2.13. A well known 1-dimensional SDE that models the velocity X_t of a particle submerged in fluid, is calles the Langevin equation, and is given by

$$\mathrm{d}X_t = -aX_t\,\mathrm{d}t + b\,\mathrm{d}B_t, \quad X_t \in \mathbb{R}, \quad a,b \in \mathbb{R}^+ \tag{2.78}$$

This equation does in fact have a well known exact analytical solution[1]. We can find this heuristically by assuming that $X_t = \exp(-at)Y_t$ and that the common product rule for differentiation applies to this term. We then obtain, by plugging into the equation:

$$-a(\exp(-at)Y_t) + \exp(-at)\dot{Y}_t = -a(\exp(-at)Y_t) + b\dot{B}_t + \tag{2.79}$$

$$\implies \mathrm{d}Y_t = g\exp(at)\,\mathrm{d}B_t \tag{2.80}$$

integrating the last integral, yields $Y_t = Y_0 + b\int_0^t \exp(as)\,\mathrm{d}B_s$, which, with the initial condition $Y_0 = x \in \mathbb{R}$ gives the exact solution

$$X_t = \exp(-at)\left(x + b\int_0^t \exp(as)\,\mathrm{d}B_s\right) \tag{2.81}$$

Since the integrand in the above expression is non-stochastic, then the resulting random variable is Gaussian, with expected value and variance that can be found, by using properties of the Itô stochastic integral:

$$\mathrm{E}[X_t] = \exp(-at)x + \exp(-at)b\mathrm{E}\left[\int_0^t \exp(as)\,\mathrm{d}B_s\right] = \exp(-at)x \tag{2.82}$$

$$\mathrm{Var}(X_t) = e^{-2at}b^2\mathrm{Var}\left[\int_0^t e^{as}\,\mathrm{d}B_s\right] = e^{-2at}b^2\int_0^t E([e^{as}]^2)\,\mathrm{d}s = \frac{b^2}{2a}(1-e^{-2at}) \tag{2.83}$$

Thus as the solution of a stochastic differential equation forms as stochastic process, an explicit expression for the exact analytical is at hand which is also the SDEs analytic transition kernel, and is given by

$$k(x', x, \Delta t) = \frac{1}{\sqrt{2\pi b^2(1-\exp(-2a\Delta t))/(2a)}}\exp\left(-\frac{(x' - x\exp(-a\Delta t))^2}{b^2(1-\exp(-2a\Delta t))/a}\right). \tag{2.84}$$

We can confirm by using the Fokker-Planck equation that this is indeed the distribution of a time-dependent solution for the SDE by calculating

$$\frac{\partial k(x', x, t)}{\partial t} = ae^{-2at}\left(-\frac{1}{(1-e^{-2at})} - \frac{2axe^{at}(x' - xe^{-at})}{b^2(1-e^{-2at})} + \frac{2a(x'-xe^{-at})^2}{b^2(1-e^{-2at})^2}\right)k(x', x, t) \tag{2.85}$$

$$\frac{\partial}{\partial x'}(x' \cdot k(x', x, t)) = \left(1 - \frac{2ax'(x'-xe^{-at})}{b^2(1-e^{-2at})}\right)k(x', x, t) \tag{2.86}$$

$$\frac{\partial^2 k(x', x, t)}{\partial x'^2} = \left(-\frac{2a}{b^2(1-e^{-2at})} + \frac{4a^2(x'-xe^{-at})^2}{(b^2(1-e^{-2at}))^2}\right)k(x', x, t) \tag{2.87}$$

By combining fractions, it can be seen (we omit the calculations here, as the expressions become very large) that

$$\frac{\partial}{\partial t}k(x', x, t) - (a\frac{\partial}{\partial x}(x' \cdot k(x', x, t)) + \frac{b^2}{2}\frac{\partial^2}{\partial x'^2}k(x', x, t)) = 0. \tag{2.88}$$

We see on letting $t \to \infty$ that

$$f(y) = \lim_{t\to\infty} k(y, x, t) = \frac{1}{\sqrt{2\pi b^2/(2a)}}\exp\left(-\frac{y^2}{b^2/2}\right) \tag{2.89}$$

By calculating $(yf(y))'$ and $f''(y)$, it can readily be shown that

$$a\frac{\partial}{\partial y}(yf(y)) + \frac{b^2}{2}\frac{\partial^2}{\partial y^2}f(y) = 0 \tag{2.90}$$

showing that $f(y)$ is a stationary solution for the Langevin equation. In particular, this shows us that starting at any distribution of the type $p_{X_0}(x') = \delta(x'-x)$, the solution always reaches this stationary solution as $t \to \infty$. Which shows that this stationary solution is stable in some sense.

As noted in [8], a stationary solution for a SDE can often be found if a analytical expression for the transition kernel $k(x', x, \Delta t)$ is at hand, by letting $\Delta t \to \infty$.

Example 2.14. A Stochastic Oscillator with damping is simply a modification of the 1-dimensional classic oscillator with driving force F(t)

$$\ddot{x} + \alpha\dot{x} + \omega^2 x = F(t) \tag{2.91}$$

where α relates to the damping of the system, and ω is the natural oscillation frequency of the system. By setting the driving force $F(t) = \xi(t)$, where $\xi(t)$ is a Gaussian white noise process with $\langle \xi(t) \rangle = 0$ and $\langle \xi(t)\xi(t+\tau)\rangle = D_\xi^2 \delta(\tau)$, where D_ξ is a positive constant, the equation becomes a coupled stochastic differential equation given by

$$\begin{aligned} \mathrm{d}X_t &= Y_t \,\mathrm{d}t && (2.92\text{a}) \\ \mathrm{d}Y_t &= -(\alpha Y_t + \omega^2 X_t)\,\mathrm{d}t + D_\xi \,\mathrm{d}B_t. && (2.92\text{b}) \end{aligned}$$

The PDE for the joint stationary distribution $f(x,y)$ of X_t and Y_t becomes by the Fokker-Planck equation

$$-y\frac{\partial}{\partial x} f(x,y) + \alpha \frac{\partial}{\partial y}(y f(x,y)) + \omega^2 x \frac{\partial}{\partial y} f(x,y) + \frac{D_\xi^2}{2}\frac{\partial^2}{\partial y^2} f(x,y) = 0. \tag{2.93}$$

One can readily see that if $\frac{\partial}{\partial y} f(x,y) = -\frac{2\alpha}{D_\xi^2} y f(x,y)$, then the fourth term in the above equation cancels the second one, and the PDE is reduced to the system

$$\begin{aligned} \frac{\partial}{\partial x} f(x,y) &= -\frac{2\alpha\omega^2}{D_\xi^2} x f(x,y) && (2.94\text{a}) \\ \frac{\partial}{\partial y} f(x,y) &= -\frac{2\alpha}{D_\xi^2} y f(x,y). && (2.94\text{b}) \end{aligned}$$

Integrating the first equation with respect to x gives $f(x,t) = h(y)\exp(-\frac{\alpha\omega^2}{D_\xi^2}x^2)$. Substituting this into the second equation gives $h'(y) = -\frac{2\alpha}{D_\xi^2} y h(y)$, with the solution $h(y) = C\exp(-\frac{\alpha}{D_\xi^2}y^2)$. Thus a stationary solution for the SDE becomes

$$f(x,y) = C\exp\left(-\frac{\alpha}{D_\xi^2}(\omega^2 x^2 + y^2)\right), \tag{2.95}$$

where C is some appropriate normalization constant. Note that for the distribution to make sense, it must be in $L^1(\mathbb{R}^2)$, which in particular implies that we must have $\alpha > 0$, for the stationary solution to exist.

Example 2.15. We shall now look at a SDE that can be written on the form

$$\dot{X}_t = \frac{\partial H}{\partial Y_t} \tag{2.96a}$$

$$\dot{Y}_t = -\frac{\partial H}{\partial X_t} - k\frac{\partial H}{\partial Y_t} + D_\xi \dot{B}_t, \tag{2.96b}$$

where $H(X, Y)$ is a differentiable continuous function. The PDE for the joint distribution of the solution becomes, according to the Fokker-Planck equation

$$-\frac{\partial}{\partial x}\left[\frac{\partial H}{\partial y}f(x,y)\right] + \frac{\partial}{\partial y}\left[\frac{\partial H}{\partial x}f(x,y)\right] + k\frac{\partial}{\partial y}\left[\frac{\partial H}{\partial y}f(x,y)\right] + \frac{D_\xi^2}{2}\frac{\partial^2}{\partial y^2}f(x,y) = 0 \tag{2.97}$$

By letting $\frac{\partial}{\partial y}f(x,y) = -\frac{2k}{D_\xi^2}\frac{\partial H}{\partial y}f(x,y)$ the fourth term cancels the third term, and by applying the product rule of differentiation on the two remaining terms, we get

$$\begin{aligned} &-\frac{\partial^2 H}{\partial x \partial y}f(x,y) - \frac{\partial H}{\partial y}\frac{\partial}{\partial x}f(x,y) + \frac{\partial^2 H}{\partial x \partial y}f(x,y) - \frac{2k}{D_\xi^2}\frac{\partial H}{\partial x}\frac{\partial H}{\partial y}f(x,y) = 0 \\ \Longrightarrow &\frac{\partial}{\partial x}f(x,y) = -\frac{2k}{D_\xi^2}\frac{\partial H}{\partial x}f(x,y). \end{aligned} \tag{2.98}$$

Thus, we are left with the following set of partial differential equations

$$\frac{\partial}{\partial x}f(x,y) = -\frac{2k}{D_\xi^2}\frac{\partial H}{\partial x}f(x,y) \tag{2.99a}$$

$$\frac{\partial}{\partial y}f(x,y) = -\frac{2k}{D_\xi^2}\frac{\partial H}{\partial y}f(x,y). \tag{2.99b}$$

Integrating the first equation with respect to x, gives

$$f(x,y) = C\exp\left(-\frac{2k}{D_\xi^2}H(x,y)\right) \tag{2.100}$$

where C is some appropriate normalization constant. We see that $f(x, y)$ also satisfies (2.99b), so $f(x, y)$ is a stationary solution for the system. As before, $f(x, y)$ must be in $L^1(\mathbb{R}^2)$, therefore, it is necessary for $H(x, y)$ to satisfy

$$\lim_{(|x|,|y|)\to(\infty,\infty)} H(x,y) = \infty \tag{2.101}$$

Example 2.16. We shall in this example return to the SDE introduced in example 2.14, but with specific parameters $\alpha = 5$, $\omega^2 = 4$ and $D_\xi = 1$. We shall, by introducing a transformation to obtain two seperate Langevin equations and derive an explicit time solution.

Originally, the time-dependent solution derived for this equation was intended to benchmark the order of convergence of the PI-method in a separate chapter. But this later turned out to be outside the scope of this thesis, and the author wanted to concentrate more on the Lotka-Volterra equation. It is nonetheless a nice illustration of how we can use the explicit time solution for the Langevin equation, obtained from the Fokker-Planck equation, to derive explicit time solutions for more complicated problems.

We start by noting, that with the parameters $\alpha = 5$, $\omega^2 = 4$ and $D_\xi = 1$, the system can be rewritten in matrix form as

$$\begin{bmatrix} \mathrm{d}X_t \\ \mathrm{d}Y_t \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix} \begin{bmatrix} X_t \\ Y_t \end{bmatrix} \mathrm{d}t + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{d}B_t^1 \\ \mathrm{d}B_t^2 \end{bmatrix} = A \begin{bmatrix} X_t \\ Y_t \end{bmatrix} + B \begin{bmatrix} \mathrm{d}B_t^1 \\ \mathrm{d}B_t^2 \end{bmatrix} \tag{2.102}$$

We want to introduce a transformation $(X_t, Y_t)^T = U(V_t, W_t)^t$ where U is a invertible 2×2 matrix, such that when we multiply the equation from the left with U^{-1}, $U^{-1}AU$ becomes a diagonal matrix. With an spectral decomposition, we can do precisely this. A has eigenvalue $\lambda_1 = -1$ and $\lambda_2 = -4$ with corresponding normalized eigenvectors $u_1 = (1, -1)^T/\sqrt{2}$ and $u_2 = (-1, 4)^T/\sqrt{5}$. So by letting $U = \begin{bmatrix} u_1 & u_2 \end{bmatrix}$, we obtain, by inserting $(X_t, Y_t)^T = U(V_t, W_t)^t$

$$\begin{bmatrix} \mathrm{d}V_t \\ \mathrm{d}W_t \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -4 \end{bmatrix} \begin{bmatrix} V_t \\ W_t \end{bmatrix} \mathrm{d}t + U^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{d}B_t^1 \\ \mathrm{d}B_t^2 \end{bmatrix} \tag{2.103}$$

The inverse of the matrix U is given by

$$U^{-1} = \begin{bmatrix} \frac{4}{3}\sqrt{2} & \frac{1}{3}\sqrt{2} \\ \frac{1}{3}\sqrt{5} & \frac{1}{3}\sqrt{5} \end{bmatrix} \tag{2.104}$$

We obtain the two seperate Langevin equations

$$\mathrm{d}V_t = -V_t \, \mathrm{d}t + \frac{\sqrt{2}}{3} \, \mathrm{d}B_t^2 \tag{2.105}$$

$$\mathrm{d}W_t = -4W_t \, \mathrm{d}t + \frac{\sqrt{5}}{3} \, \mathrm{d}B_t^2 \tag{2.106}$$

With the solutions assuming the initial conditions $(V_0, W_0)^T = (v_0, w_0)^T = U^{-1}(x_0, y_0)$

$$V_t = \exp(-t) \left(v_0 + \frac{\sqrt{2}}{3} \int_0^t \exp(s) \, \mathrm{d}B_s \right) \tag{2.107}$$

$$W_t = \exp(-4t) \left(w_0 + \frac{\sqrt{5}}{3} \int_0^t \exp(4s) \, \mathrm{d}B_s \right) \tag{2.108}$$

transforming back to the original variables with $(X_t, Y_t)^T = U(V_t, W_t)^t$, we get

$$X_t = \left(e^{-t} \frac{v_0}{\sqrt{2}} - e^{-4t} \frac{w_0}{\sqrt{5}} \right) + \frac{1}{3} \int_0^t e^{s-t} - e^{4(s-t)} \, \mathrm{d}B_s \tag{2.109}$$

$$Y_t = \left(e^{-4t} \frac{4w_0}{\sqrt{5}} - e^{-t} \frac{v_0}{\sqrt{2}} \right) + \frac{1}{3} \int_0^t 4e^{4(s-t)} - e^{s-t} \, \mathrm{d}B_s \tag{2.110}$$

and finally, we obtain the solution, as we transform the initial conditions back with

$(v_0, w_0)^T = U^{-1}(x_0, y_0)$

$$X_t = \tilde{x}_0(t) + \frac{1}{3}\int_0^t e^{s-t} - e^{4(s-t)}\,\mathrm{d}B_s \tag{2.111}$$

$$Y_t = \tilde{y}_0(t) + \frac{1}{3}\int_0^t 4e^{4(s-t)} - e^{s-t}\,\mathrm{d}B_s \tag{2.112}$$

$$x_0(t) = \frac{1}{3}\left[(4e^{-t} - e^{-4t})x_0 + (e^{-t} - e^{-4t})y_0\right] \tag{2.113}$$

$$y_0(t) = \frac{1}{3}\left[4(e^{-4t} - e^{-t})x_0 + (4e^{-4t} - e^{-t})y_0\right] \tag{2.114}$$

As with the Langevin equation, the integrand in the stochastic integral is non-stochastic, so the distribution of is Gaussian. As a Gaussian distribution is uniquely defined by its expected value and covariance matrix, we calculate these values, and obtain

$$\mathrm{E}(X_t) = \tilde{x}_0(t) \tag{2.115}$$

$$\mathrm{E}(Y_t) = \tilde{y}_0(t) \tag{2.116}$$

$$\begin{aligned}\mathrm{E}[(X_t - x_0(t))^2] &= \frac{1}{9}\int_0^t e^{2(s-t)} - 2e^{5(s-t)} + e^{8(s-t)}\,\mathrm{d}s \\ &= \frac{1}{40}(1 - \frac{1}{9}(20e^{-2t} - 16e^{-5t} + 5e^{-8t}))\end{aligned} \tag{2.117}$$

$$\begin{aligned}\mathrm{E}[((X_t - x_0(t))(Y_t - y_0(t))] &= \frac{1}{9}\int_0^t 4e^{5(s-t)} - e^{2(s-t)} - 4e^{8(s-t)} + e^{5(s-t)}\,\mathrm{d}s \\ &= \frac{1}{18}(2e^{-5t} - e^{-2t} - e^{-8t})\end{aligned} \tag{2.118}$$

$$\begin{aligned}\mathrm{E}[(X_t - x_0(t))^2] &= \frac{1}{9}\int_0^T 16e^{8(s-t)} - 8e^{5(s-t)} + e^{2(s-t)}\,\mathrm{d}s \\ &= \frac{1}{10}(1 - \frac{1}{9}(5e^{-2t} - 16e^{-5t} + 20e^{-8t}))\end{aligned} \tag{2.119}$$

Let Σ be the corresponding covariance matrix, then the joint distribution of the solution of the solution, which also is the analytical transition kernel for the problem, is

$$p(x, y, t) = \frac{1}{2\pi\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}\begin{bmatrix}x - x_0(t) & y - y_0(t)\end{bmatrix}\Sigma^{-1}\begin{bmatrix}x - x_0(t) \\ y - y_0(t)\end{bmatrix}\right) \tag{2.120}$$

We can see, by looking at the expressions for the expected value and the covariance matrix, that in the limit $t \to \infty$

$$\lim_{t\to\infty} p(x, y, t) = \frac{10}{\pi}\exp\left(-5(4x^2 + y^2)\right) \tag{2.121}$$

We obtain precisely what is the stationary solution for this SDE with the parameters $\alpha = 5$, $\omega^2 = 4$ and $D_\xi = 1$.

It should be noted, as a final remark on this example, that the method presented to get an analytical solution is not limited to this example, but can be used for any linear system of SDE if A has n distinct real eigenvalues. If there are n distinct complex eigenvalues, then we generally get a complex solution as well. It should however in most cases be possible to find a workaround, to obtain a real solution, typically with sinusoidal terms.

In theory, we should be able to approximate the time evolution of the distribution of the solution of a SDE, by applying the finite difference method or the finite element method to the Fokker-Planck equation, in order to compare results to PI. In reality however, there is a problem with this approach: We must choose a bounded subspace of the space $\mathbb{R}^n$ in which the state vector X_t lies in, and must apply boundary values to this subspace for the distribution of the solution. These boundary values are generally unknown and generally non-zero and non-constant however. One solution is to choose a very large subspace, and set the boundary value to some low constant value, but this becomes an unreliable solution, unless we have a very large number of grid nodes, which quickly increases computation time. On the contrary, PI does not explicitly require boundary conditions, and is therefore more advantageous in this regard.

2.7 Convergence

In this section we will look at the convergence of the PI-method. A thorough exposition of the convergence of the PI-method is given in [8], we shall however only present the most central theorems, and mostly without proof here.

It should first be noted that Path Integration does in general not converge as $(h, \Delta t) \to (0, 0)$, in particular if we let h be positive and fixed and let $\Delta t \to 0$ while evaluating the solution on a non-degenerate time-interval then the error will diverge to infinity. This has to do with interpolation error and the non-exact numerical evaluation of transition kernel.

To see this, suppose that we can calculate the exact transition kernel for each step, that is, we ignore any error that arises from the time discretization of the SDE, the spline interpolations and so forth, then in evaluating the PI operator (2.29) in order to calculate $p(x, y, t)$, we will need to use a numerical method, which will generally not be exact. Depending on the order of the method, we will typically have an error each time we evaluate this integral of order $\mathcal{O}(h^p)$ where p is an integer that represents the order of the method. This error accumulates with each time step, and if N_t is the number of time-steps over the time interval $[0, t]$, then the global error will be of order

$$\|\tilde{p}(x, y, t) - p(x, y, t)\|| = \mathcal{O}(N_t h^p) = \mathcal{O}\left(t\frac{h^p}{\Delta t}\right)$$

where $\tilde{p}(x, y, t)$ is the approximate solution and $p(x, y, t)$ is the exact solution. From this expression, it is clear that even if we get an exact representation of $k(x', y', t'x, y, t)$ for each step, the approximate solution will still diverge if h is fixed and positive while we let $\Delta t \to 0$, this error will of course accumulate even more if we take interpolation error and the error from evaluating A_1' and A_2' numerically into consideration.

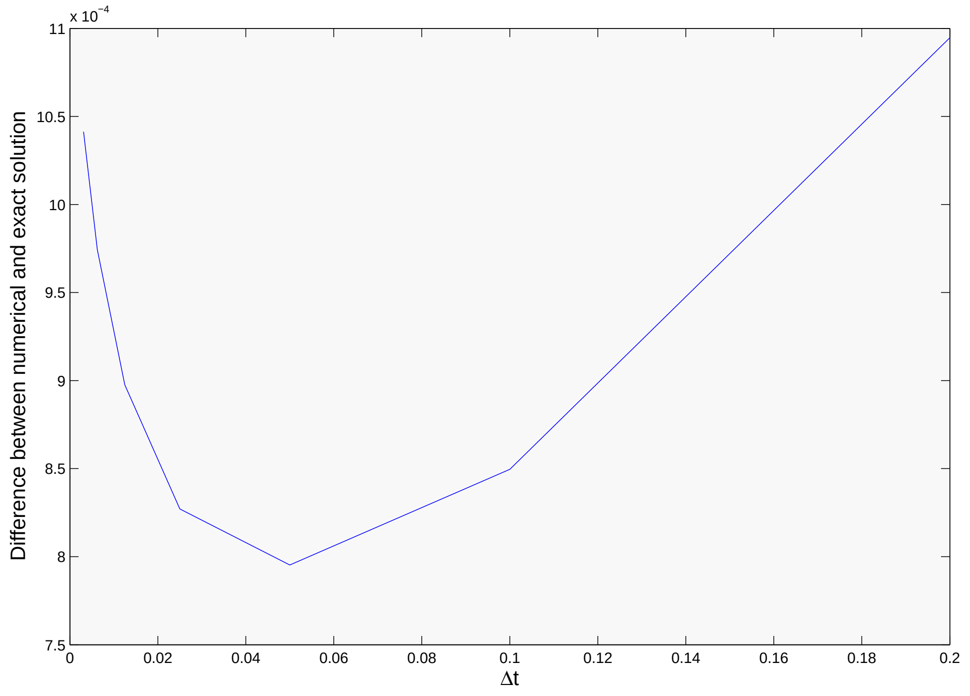


Figure 2.1: Plot of error for the stationary solution from example 2.14 with $\alpha = 0.5$, $\gamma = 1$ and $\omega^2 = 0.1$, with different values for Δt with the fixed number of 20 grid points in x- and y-direction on the mesh $[-10, 10] \times [-10, 10]$. The initial distribution was the stationary solution. The error was measured at $t = 1$

Figure 2.1 illustrates this divergence for the SDE used in example 2.14, and as we see the error decreases until $\Delta t \approx 0.05$, but then takes a sharp turn when Δt becomes smaller, and increases quickly, exactly as expected.

2.7.1 Convergence of the forward time discrete transition kernel

We will first look at the behavior of the mapping $x' \to k(x', x, t)$ which we shall call the forward transition kernel, for an autonomous SDE of the type

$$X_{t'} = X_t + \int_t^{t'} a(X_t)\,\mathrm{d}x + b\Delta B_t \tag{2.122}$$

with the Euler approximation

$$X_{t'} = X_t + a(X_t)\Delta t + b\Delta B_t \tag{2.123}$$

we get a forward transition kernel, that we shall in this section refer to as the forward time discrete transition kernel, that is given by

$$f(x) = \tilde{k}(x', x, \Delta t) = \prod_{i=1}^{r} \delta(x_i' - x_i - a_i(x,t)\Delta t) \cdot e^{\left[\frac{1}{2}(\tilde{x}'-\tilde{x}-\tilde{a}(x,t))^T \tilde{g}^{-1}(\tilde{x}'-\tilde{x}-\tilde{a}(x)\Delta t)\right]} \tag{2.124}$$

where $\tilde{x} = (x_{r+1}, \dots, x_n)$, $\tilde{a}(x,t) = (a_{r+1}(x,t), \dots, a_n(x,t))$ and $\tilde{g} = \tilde{b}^T\tilde{b}$, where $\tilde{b}$ is the submatrix of b, containing all the non-zero rows of b. Suppose that we an exact forward transition kernel is obtainable, then the following result provides conditions for the forward time discrete transition kernel to converge

Theorem 2.17. *Let $1 > \epsilon > 0$, and suppose that*

- $\|a(x,t) - a(y,t)\| \leq K\|x - y\|$
- $\|b(x,t) - b(y,t)\| \leq K\|x - y\|$
- $\|a(x,t)\|^2 \leq D^2(1 + \|x\|^2)$
- $\|b(x,t)\|^2 \leq D^2(1 + \|x\|^2)$
- $a(X_t,t), b(X_t,t) \in C^4$
- *The exact and time discrete transition kernel $x \to k(x', x, \Delta t), x \to \tilde{k}(x', x, \Delta t) \in C_P^4$*

Then there exists a constant K_ϵ independent of Δt such that

$$\|k(x', x, \Delta t) - \tilde{k}(x', x, \Delta t\|_{x',2} < K_\epsilon\sqrt{\Delta t} \text{ for all } \Delta t > \epsilon \tag{2.125}$$

Proof. Given the five first conditions, we know by theorem 2.4 that the exact solution X_t and the Pathwise Euler approximation of the solution $\tilde{X}_t$ satisfies

$$|E[h(\tilde{X}_T)] - E[h(X_T)]| < C_h\Delta t \tag{2.126}$$

We know that $p_{X_t}(x,t) = \int_{\mathbb{R}^n} k(x,y,t)p_0(x,\Delta t)\,\mathrm{d}y$ and $p_{\tilde{X}_t}(x,t) = \int_{\mathbb{R}^n} \tilde{k}(x,y,\Delta t)\,\mathrm{d}y$, for some initial distribution $p_0(x,t)$. Fix an $\epsilon > 0$ and a points $x_0 \in \mathbb{R}^n$, then the initial distribution becomes $p_0(x,t) = \delta(x - x_0) = \prod_{i=1}^n \delta(x_i - x_{i,0})$, and we can calculate

$$|E[g(X_{\Delta t})] - E[g(\tilde{X}_{\Delta t})]| = \left| \int_{\mathbb{R}^n} h(y) \int_{\mathbb{R}^n} [k(y,x,\Delta t) - \tilde{k}(y,x,\Delta t)]\delta(x - x_0)\,\mathrm{d}x\,\mathrm{d}y \right| \tag{2.127}$$

$$= \left| \int_{\mathbb{R}^n} g(y)[k(y,x_0,\Delta t) - \tilde{k}(y,x_0,\Delta t)]\,\mathrm{d}y \right|. \tag{2.128}$$

Since $x \to k(x', x, \Delta t), x \to \tilde{k}(x', x, \Delta t) \in C_P^4$, then so is $x \to [k(x', x, \Delta t) - \tilde{k}(x', x, \Delta t)]$, and we can set $h_{\Delta t}(y) = [k(y,x,\Delta t) - \tilde{k}(y,x,\Delta t)]$. Moreover by setting

$$K_\epsilon^2 = \sup_{\Delta t \in (\epsilon,1)} \left(K_{h_{\Delta t}(y)}\right) < \infty \tag{2.129}$$

then we finally obtain

$$\|k(x', x, \Delta t) - \tilde{k}(x', x, \Delta t\|_{x',2}^2 < K_\epsilon^2\Delta t \tag{2.130}$$

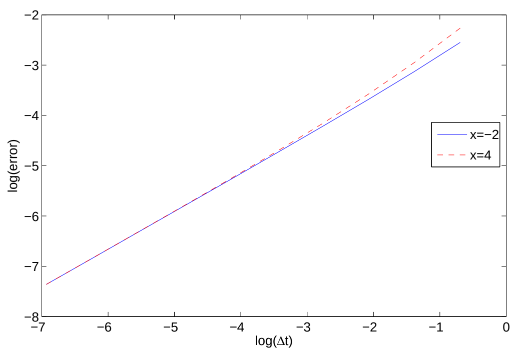
□

It should be noted that the above theorem does not guarantee that the forward time discrete transition kernel converges as $\epsilon \to 0$: The time discrete forward transition kernel given in (2.124) is, if $a(x,t)$ and $b(x,t)$ is sufficiently well behaved, in C^∞ and for all $\Delta t \in [\epsilon, 1)$, the expression is bounded, and thus of polynomial growth. however, with $\Delta t = \epsilon$ the resulting expression when $\epsilon \to 0^+$ becomes unbounded, and not of polynomial growth, making the above theorem invalid in this limit.

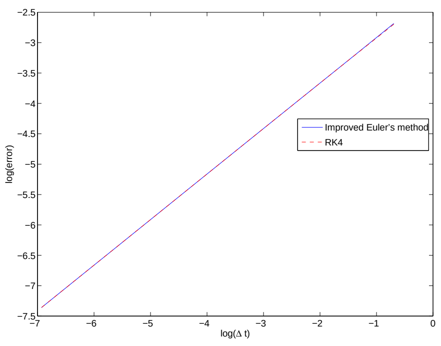
However, as long as $K_\epsilon = \mathcal{O}(\epsilon^{-r})$ with $r < 1$, then the above result shows us that we can at least make the error very small, by choosing Δt to be small. We shall illustrate this with the Langevin equation from example 2.13, where we found the exact analytical transition kernel given by

$$k(x', x, \Delta t) = \frac{1}{\sqrt{2\pi b^2(1-\exp(-2at))/(2a)}} \exp\left(-\frac{(x' - x\exp(-at))^2}{b^2(1-\exp(-2at))/a}\right). \tag{2.131}$$

The corresponding time discrete transition kernel, if we use the Euler scheme on



(a) log(error)= $\ln(\|k(x',x,\Delta t) - \tilde{k}_{\mathrm{E}}(x',x,\Delta t)\|_{2,x'})$



(b) The blue solid line gives $\ln(\|k(x',x,\Delta t) - \tilde{k}_{\mathrm{IE}}(x',x,\Delta t)\|_{2,x'})$, while the dashed red line gives $\ln(\|k(x',x,\Delta t) - \tilde{k}_{\mathrm{RK4}}(x',x,\Delta t)\|_{2,x'})$, both with $x = 4$

Figure 2.2: The figures displays log-plots of the error of the time discrete transition kernel in terms of the $\|\cdot\|_2$ norm, for different approximation methods for the deterministic term $-\int aX_t\,\mathrm{d}t$ of the Langevin equation. The parameters used were $a = 0.5$ and $b = 1$

the SDE, is

$$\tilde{k}_{\mathrm{E}}(x', x, \Delta t) = \frac{1}{\sqrt{2\pi b^2 \Delta t}} \exp\left(-\frac{(x' - (1 - a\Delta t)x)^2}{2b^2\Delta t}\right) \tag{2.132}$$

This allows us to test the bounds found in the above theorem, by computing $\|k(x',x,t) - \tilde{k}(x',x,t)\|$ with an appropriate numerical method and integration interval.

This has been done with MATLAB and the results are displayed in figure 2.2(a) which gives the error in terms of the $\|\cdot\|_2$ norm of the forward transition kernel against the time discrete transition given in equation (2.132). The plot displays, more or less, straight lines and fitting the given curves with a linear polynomial in the least square sense, gave $0.74\ln(\Delta t) - 2.16$ which seems to indicate a convergence of order $\mathcal{O}((\Delta t)^{3/4)})$, indicating that the bound found in theorem 2.17 might be slightly conservative.

It is also interesting to find out if the order of convergence changes if we improve the approximation of the deterministic term $\int_t^{t'} -aX_t \, \mathrm{d}t$. We will try this with the improved Euler method and RK4. The corresponding transition kernels become, respectively

$$\tilde{k}_{\mathrm{IE}}(x', x, \Delta t) = \frac{1}{\sqrt{2\pi b^2 \Delta t}} \exp\left(-\frac{(x' - (1 - a\Delta t + \frac{1}{2}(a\Delta t)^2)x)^2}{2b^2\Delta t}\right) \tag{2.133}$$

$$\tilde{k}_{\mathrm{RK4}}(x', x, \Delta t) = \frac{1}{\sqrt{2\pi b^2 \Delta t}} \exp\left(-\frac{(x' - (1 - a\Delta t + \frac{1}{2}(a\Delta t)^2 - \frac{1}{6}(a\Delta t)^3 + \frac{1}{24}(a\Delta t)^4)x)^2}{2b^2\Delta t}\right) \tag{2.134}$$

Figure 2.2(b) shows the calculated error for these transition kernels as a function of Δt, and goes to show that the improvement, if any, is very slight by using higher order approximation for the deterministic term. Just as was the case with Euler's method, the error seems to decay in the order of $\mathcal{O}((\Delta t)^{3/4})$. This might seem surprising, but it should be noted that while we improve the approximation of the expected value of the exact Gaussian transition kernel, we retain the same approximation for the variance, which is not better than an Euler approximation.

2.7.2 Convergence of the backward time discrete transition kernel

We shall look at the convergence of the the backward time discrete transition kernel, that is the mapping $x \to \tilde{k}(x', x, \Delta t)$. The author has not been able to find any concrete results on this, as was the case with the forward transition kernel, however we shall continue to look at the example of the Langevin equation.

Figure 2.3 gives the error in a log-plot in terms of the numerical calculated $\|\cdot\|_2$-norm of the difference between the exact backward transition kernel and the time discrete transition kernel. The results are very similar as the results obtained with the forward transition kernel, and the slope of the line fitted to the curves in the least square sense, for sufficiently small Δt, was approximately $\frac{3}{4}$ which agrees with the results found in the previous section, and indicates a convergence of order $\mathcal{O}((\Delta t)^{3/4})$. This also indicates that there might exist similar bounds on the error, as presented in theorem 2.17.

Under sufficient regularity conditions on the deterministic term $a(X_t, t)$ and the diffusion term $b(X_t, t)$ of the SDE (that is, such that a unique solution of the SDE exists) , it seems reasonable to assume, following this example, that we should have $\|k(x', x, \Delta t) - \tilde{k}(x', x, \Delta t)\|_{2,x} \to 0$ for all $x' \in \mathbb{R}^n$ as $\Delta t \to 0$.

Interestingly, just as with the forward time discrete transition kernel, any difference in the error observed in the Langevin equation example when using different approximations for the deterministic term is only visible for large values of Δt. In the heuristic discussion that follows, we shall attempt to explain this.

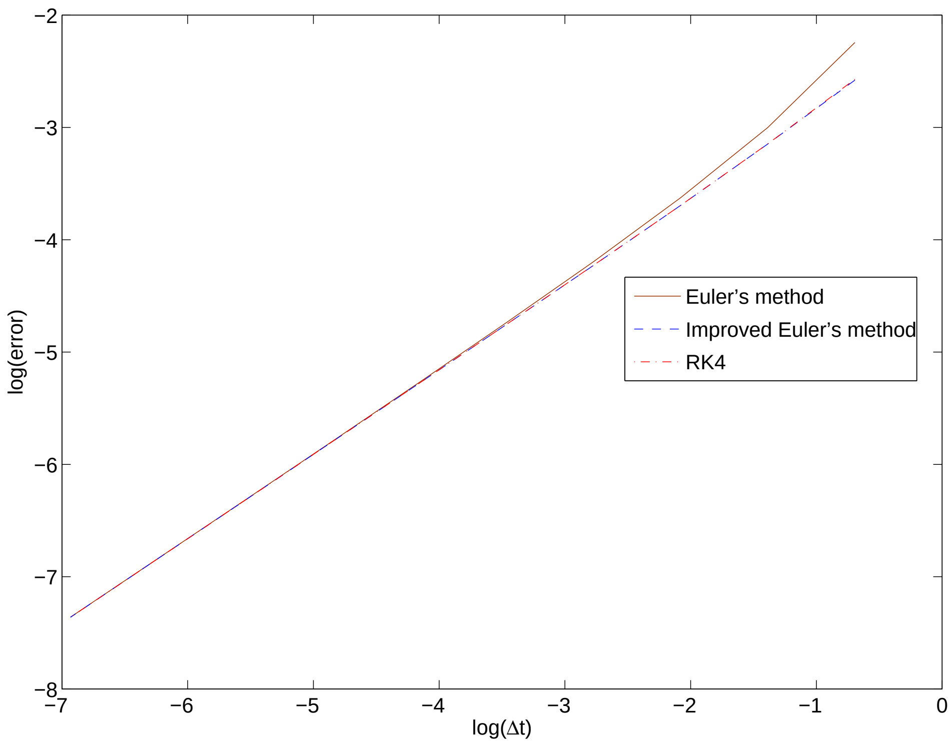


Figure 2.3: log(error)= $\|k(x', x, \Delta t) - \tilde{k}(x', x, \Delta t)\|_{2,x}$, where $\tilde{k}(x', x, \Delta t) = \tilde{k}_{\mathrm{E}}(x', x, \Delta t)$ in (2.132) for the brown solid curve, $\tilde{k}(x', x, \Delta t) = \tilde{k}_{\mathrm{IE}}(x', x, \Delta t)$ in (2.133) for the dashed blue curve, and $\tilde{k}(x', x, \Delta t) = \tilde{k}_{\mathrm{RK4}}(x', x, \Delta t)$for the red dashed and dotted curve. The parameters used were $\alpha = 0.5$, $\beta = 1$ and $x' = 4$

In order to do this, for ease of notation, we first introduce

$$f(\Delta t) = \exp(-a\Delta t) \tag{2.135}$$

$$g(\Delta t) = \frac{b^2}{2a}(1 - \exp(-2a\Delta t)) \tag{2.136}$$

Let us at first ignore all error from the exponential function in the time discrete transition kernel, thus $\|k(x', x, \Delta t) - \tilde{k}(x', x, \Delta t)\|_{2,x'}$ becomes

$$\left\| \left[\frac{1}{\sqrt{2\pi b^2 \Delta t}} - \frac{1}{\sqrt{2\pi g(t)}} \right] e^{\left(-\frac{(x' - x f(\Delta t))^2}{g(\Delta t)} \right)} \right\|_{2,x'} = \left[\frac{1}{\sqrt{2\pi b^2 \Delta t}} - \frac{1}{\sqrt{2\pi g(t)}} \right] \mathcal{O}((\Delta t)^{1/4}) \tag{2.137}$$

The above calculation can be seen by noting that

$$g(\Delta t) = \mathcal{O}(\Delta t) = b^2 \Delta t + \mathcal{O}((\Delta t)^2) \tag{2.138}$$

and by computing the integral

$$\left\| e^{\left(-\frac{(x' - x f(\Delta t))^2}{2g(\Delta t)} \right)} \right\|_{2,x'} = \sqrt{\int_{\mathbb{R}} e^{-\frac{(x' - x f(\Delta t))^2}{g(\Delta t)}} \, \mathrm{d}x'} = \sqrt{\sqrt{\frac{g(t)}{2}}} = \mathcal{O}(\Delta t)^{1/4}). \tag{2.139}$$

Next, by expanding $(2\pi b^2\Delta t)^{-(1/2)} = (2\pi g(t) + \mathcal{O}((\Delta t)^2))^{-(1/2)}$ in a Taylor series around $x = 2\pi g(t)$, we get

$$\frac{1}{\sqrt{2\pi b^2\Delta t}} = \frac{1}{\sqrt{2\pi g(t)}} - \frac{\mathcal{O}((\Delta t)^2)}{(2\pi g(t))^{3/2}} + \frac{\mathcal{O}((\Delta t)^4)}{(2\pi g(t))^{5/2}} - \cdots = \frac{1}{\sqrt{2\pi g(t)}} + \mathcal{O}((\Delta t)^{1/2}). \tag{2.140}$$

Thus, by inserting this Taylor expansion into equation (2.137), we get that the error between the exact and time discrete transition kernel, when we ignore error from the exponential function, is in the order of $\mathcal{O}((\Delta t)^{3/4})$. This means that an error in the order of $\mathcal{O}((\Delta t)^{3/4})$ is the best we can achieve with the current transition kernel for this example, no matter how good the RK-approximation of the drift-term $-aX_t$ of the original SDE. This also teaches us the important heuristic lesson, that a numerical approximation is only as good as its weakest link, so improving stronger links, does not help very much.

t	$\|k(x',x,t) - \tilde{k}(x',x,t)\|_{2,x'}$ (Euler's method)	$\|k(x',x,t) - \tilde{k}(x',x,t)\|_{2,x'}$ (IE method)	$\|k(x',x,t) - \tilde{k}(x',x,t)\|_{2,x'}$ (RK4)
$\frac{1}{2}$	$435 \cdot 10^{-4}$	$740 \cdot 10^{-5}$	$23 \cdot 10^{-6}$
$\frac{1}{4}$	$174 \cdot 10^{-4}$	$146 \cdot 10^{-5}$	$1.1 \cdot 10^{-6}$
$\frac{1}{8}$	$71 \cdot 10^{-4}$	$29 \cdot 10^{-5}$	$5.88 \cdot 10^{-8}$
$\frac{1}{16}$	$29 \cdot 10^{-4}$	$6.2 \cdot 10^{-5}$	$3 \cdot 10^{-9}$
$\frac{1}{32}$	$12 \cdot 10^{-4}$	$1.2 \cdot 10^{-5}$	$1 \cdot 10^{-10}$
$\frac{1}{64}$	$5.2 \cdot 10^{-4}$	$0.2 \cdot 10^{-5}$	$8.2 \cdot 10^{-12}$
$\frac{1}{128}$	$2.1 \cdot 10^{-4}$	$0.05 \cdot 10^{-5}$	$4.3 \cdot 10^{-13}$

Table 2.2: Convergence rate of the time discrete transition kernel given in equation (2.141) with parameters $\alpha = 0.5$ and $b = 1$.

The same result, that the error is bound from below for the backward transition kernel, can be shown by writing $\exp[(x' - f(t)x)^2/(2g(t))] = \exp[f(t)^2(x - x'/f(t))/(2g(t))]$, as $f(t) = \mathcal{O}(1)$, we get the same bounds on the error when taking the $\|\cdot\|_2$-norm with respect to x.

It should be interesting to see what kind of error we could expect if the variance of the gaussian transition kernel was exact, while we had a RK-approximation for the determinstic term, that is, a transition kernel of the form

$$\tilde{k}(x', x, \Delta t) = \frac{1}{\sqrt{2\pi g(t)}} \exp\left[-\frac{(x' - x - r(a,x)\Delta t)^2}{2g(t)}\right] \tag{2.141}$$

where $x + r(a,x)\Delta t$ is a RK-approximation of $xf(t)$, that is $x + r(a,x)\Delta t = xf(t) + \mathcal{O}((\Delta t)^{p+1})$ where p is the order of the RK-method. Focusing first at the argument of the exponential function we get

$$\begin{aligned}\frac{(x' - x - r(a,x)\Delta t)^2}{2g(t)} &= \frac{(x' - xf(t))^2 - (x' - xf(t))\mathcal{O}((\Delta t)^{p+1}) + \mathcal{O}((\Delta t)^{2p+2})}{2g(t)} \\ &= \frac{(x' - x - r(a,x)\Delta t)^2}{2g(t)} + \mathcal{O}((\Delta t)^p)(x' - xf(t)) \end{aligned} \tag{2.142}$$

expanding $\exp\left[-\frac{(x'-x-r(a,x)\Delta t)^2}{2g(t)}\right]$ around its taylor series (centered at x=0) yields

$$\exp\left[-\frac{(x'-x-r(a,x)\Delta t)^2}{2g(t)}\right] = \exp\left[-\frac{(x'-xf(t))^2}{2g(t)}\right]\left(1+\frac{\mathcal{O}((\Delta t)^p)(x'-xf(t))}{2}+\dots\right) \tag{2.143}$$

thus, for the error of the forward transition kernel, we get the following expression for $\|k(x',x,\Delta t)-\tilde{k}(x',x,\Delta t)\|_{2,x'}$

$$\sqrt{\int_{\mathbb{R}}(x'-xf(t))^2\frac{\mathcal{O}((\Delta t)^{2p})}{2\pi g(t)}\exp\left[-\frac{(x'-xf(t))^2}{g(t)}\right]} \tag{2.144}$$

$$=\sqrt{\int_{\mathbb{R}}(x'-xf(t))^2\frac{\mathcal{O}((\Delta t)^{2p-1/2})}{\sqrt{\pi g(t)}}\exp\left[-\frac{(x'-xf(t))^2}{g(t)}\right]} \tag{2.145}$$

$$=\mathcal{O}((\Delta t)^{p-1/4})\sqrt{\int_{\mathbb{R}}(x'-xf(t))^2\frac{1}{\sqrt{\pi g(t)}}\exp\left[-\frac{(x'-xf(t))^2}{g(t)}\right]} \tag{2.146}$$

we recognize the integral under the square root of the second central moment of a Gaussian distribution, which is equal to its variance $g(t)/2=\mathcal{O}(\Delta t)$, we thus get

$$\|k(x',x,\Delta t)-\tilde{k}(x',x,\Delta t)\|_{2,x'}=\mathcal{O}((\Delta t)^{p-1/4})\sqrt{\mathcal{O}(\Delta t)})=\mathcal{O}((\Delta t)^{p+1/4}) \tag{2.147}$$

The data given in table 2.2 confirms these calculations, as plotting the data in log plots gives more or less straight lines, with the slope of the line fitted to the logarithm of the data in the least square sense were 1.27, 2.26 and 4.29 for Euler's method, improved Euler's method and RK4, respectively.

2.7.3 Convergence of Path Integration

We shall in this section present a theorem, that was given a lengthy exposition and proof for in [8]. As proving the convergence is not one of the main goals of this thesis, we shall only give the theorem, without proving it.

We shall first introduce some concepts. In particular, in order to prove the convergence, we need to make sure that the PI-operator preserves smoothness of the density, that it does not suddenly introduce singularities or discontinuities in the solution. We make this concept precise by introducing the space $\mathbb{D}$, the space of *smooth densities*

Definition 2.18. A function $f:\mathbb{R}\to\mathbb{R}\in L^1(\mathbb{R})$ is said to be of class $\mathbb{D}$ if the following condition are satisfied

- $f\geq 0$.
- $\int_{\mathbb{R}}f\,\mathrm{d}x=1$.
- $f\in C^2(\mathbb{R})$

Furthermore, in implementing the PI operator $PI_{\Delta t}u_0 = \int_{\mathbb{R}} k(x', x, \Delta t)u_0(x, t)\,\mathrm{d}x$ we must limit the integration interval to one with finite length. We can do this formally by replacing the transition kernel by a *truncated transition kernel* $k_{[L,R]}(x', x, \Delta t)$ defined by

$$k_{[L,R]}(x', x, \Delta t) = k(x', x, \Delta t)\mathbf{1}_{[L,R]} \tag{2.148}$$

Furthermore, we evaluate the the path integration solution only at discrete points within a bounded interval $[L, R] \subset \mathbb{R}$ on a computer. We denote this bounded interval and the bounded interval in the truncated transition kernel as truncations limits (L, R).

We look at a 1-dimenisional autonomous well-posed (a unique solution exists) SDE of the type

$$\mathrm{d}X_t = a(X_t)\,\mathrm{d}t + b(X_t)\,\mathrm{d}B_s \quad p_{X_0}(x) = u_0 \tag{2.149}$$

Let $u^*_T = p_{X_t}(x, T)$ for the exact solution X_t. Let $M \in \mathbb{N}$ and define $\Delta t = T/M$. Let u^s_i denote the ith timestep corresponding to $t_i = i\Delta t$ of the numerical PI solution density represented as a spline, resulting from the time discrete transition kernel $\tilde{k}(x', x, \Delta t)$.

Theorem 2.19. *The numerical path integration error, of the problem presented in equation* (2.16)*, in terms of the 2-norm $\|u^*_T - u^s_M\|_2$ can be made arbitrarily small, provided that the following conditions are satisfied:*

- *The time discrete transition kernel has the backward convergence property*

$$\lim_{\Delta t \to 0} \|k(x', x, \Delta t) - k(x', x, \Delta t)\|_{x,2} = 0, \textit{ for all } x' \in \mathbb{R}. \tag{2.150}$$

- *$u^*_t \in \mathbb{D}$ for all $t \in [0, T]$.*
- *All truncation limits $(L, R) \to (-\infty, \infty)$.*
- *The spatial grid step $\Delta x \to 0$.*
- *The quadrature grid step $\Delta I \to 0$.*

Chapter 3

Lotka-Volterra system with harvesting

3.1 The basic equation

The basic equation that we shall consider is given by the pair of differential equations

$$\dot{U} = -mU + k\beta UV \tag{3.1a}$$

$$\dot{V} = \alpha V[1 + h(t)] - \beta UV - \gamma V^2 \tag{3.1b}$$

which describes the dynamics of the population of two interacting species. We group these species into two groups: predators/parasites and prey/hosts, and denote by $U(t)$ the population of predators/parasites and $V(t)$ the population of prey/hosts. The function $h(t)$ essentially modifies parts of the reproduction rate α according to temporal variations of the prey's environment. m and k denotes the death rate and growth rate of the predator, respectively. β is a parameter that relates to the response of the predator to the prey, while $\frac{1}{\gamma}$ governs the carrying capacity of the population of prey in the presence of the self-limitation term V^2.

As $h(t)$ simulates temporal variations in the growth of prey, we might expect periodic annual variations due to the seasons. However, there might also be more complex factors involved, that are better modeled by random components. We shall consider two different random processes for $h(t)$

$$h(t) = \xi(t) \tag{3.2}$$

where $\xi(t)$ is a Gaussian white noise process in the Stratonovich sense, with $\langle\xi(t)\rangle = 0$ and $\langle\xi(t)\xi(t+\tau)\rangle = D_\xi\delta(t)$, where D_ξ is a constant related to the strength of the diffusion process, and $\delta(t)$ is the Dirac delta function. This model only accounts for purely random variations, and in case we want to take seasonal varations into account, it is appropriate to study a model of $h(t)$ with some underlying periodicity. Adding a sinusoidal to (3.2) gives such a model, and gives

$$h(t) = \lambda\sin(\nu t) + \xi(t) \tag{3.3}$$

where $\xi(t)$ is as given in (3.2). These are the two models which will be discussed.

3.2 Harvesting

Harvesting is a way to to control population dynamics and avoid unwanted population dynamics. One such technique is called Treshold Policy (TP), which works in the following simple way: When the population of a species is above a certain level of threshold, harvesting starts, and when the population falls below that level, harvesting stops. If u denotes the population of the species, then a simple function that models TP would be

$$\phi(u) = \begin{cases} 0 & : u < T \\ \epsilon & : u \geq T \end{cases} \tag{3.4}$$

where T is the population threshold and ϵ is the harvesting rate. This is not a model that is very usable for real-life applications however, as the function is discontinuous at $u = T$, time delays and capital constraints preventing a sudden start of harvesting makes it difficult to follow this in practice. An alternative continuous threshold policy give a gradual increase in harvesting rate as the population increases above the threshold:

$$\psi(u) = \begin{cases} 0 & : u < T \\ \frac{\epsilon(u-T)}{\alpha-t} & : T \leq u < \alpha \\ \epsilon & : u \geq \alpha \end{cases} \tag{3.5}$$

where ϵ represents the maximal harvesting rate, and α represents the threshold where the maximal harvesting rate starts if the population goes above it.

A slightly more practical function, and which captures the idea of a gradual continuous increase is one given in is

$$H(u) = \begin{cases} 0 & : u < T \\ \frac{h(u-T)}{h+(u-T)} & : u \geq T \end{cases} \tag{3.6}$$

where h is the upper limit of harvesting rate, this is the representation that shall be used in this thesis. Figure 3.1 illustrates how the harvesting rate as a function of u increases.

We will study the effects of controlling the population of predators, so the modification of the basic model becomes

$$\dot{U} = -mU + k\beta UV - H(U) \tag{3.7a}$$
$$\dot{V} = \alpha V[1 + h(t)] - \beta UV - \gamma V^2 \tag{3.7b}$$

where the function H is as given in (3.6).

3.3 Equilibrium points and stationary solutions

We will first make a brief analysis of the equilibrium points of equations (3.1) and (3.7). The deterministic system corresponding to equation (3.7) is

$$\dot{u} = -u[m - k\beta v] - H(u) \tag{3.8a}$$
$$\dot{v} = v[\alpha - \beta u - \gamma v] \tag{3.8b}$$

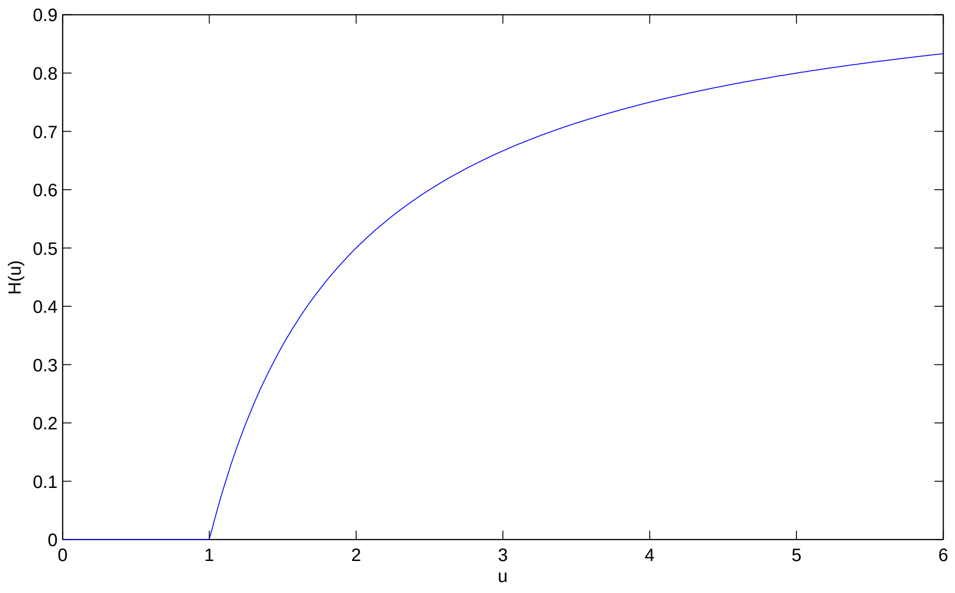


Figure 3.1: The harvesting function $H(u)$ with $T = h = 1$

if we constrain $u < T$, then the above equations represents the corresponding deterministic system of equation (3.1). In this case, we see immediately that $(u, v) = (0, 0)$ and $(u, v) = (0, \frac{\alpha}{\gamma})$ are two equilibrium points where one or both of the species are extinct. For the purposes of this discussion, however, we are more interested in equilibrium points where both species remain alive. In addition, we also want to make sure that the population sizes u and v are physically acceptable i.e. $u, v \geq 0$. Since the mapping $x \to \exp(x)$ is a bijection from $\mathbb{R}$ to $(0, \infty)$, the logarithmic transformation

$$x = \ln u, \quad y = \ln v \tag{3.9}$$

will do nicely for these purposes. Inserting this into (3.8) gives

$$\dot{x} = -m + k\beta \exp(y) - \tilde{H}(x) \tag{3.10a}$$

$$\dot{y} = \alpha - \beta \exp(x) - \gamma \exp(y) \tag{3.10b}$$

where

$$\tilde{H}(x) = \begin{cases} 0 & : x < \ln(T) \\ \frac{h(1-T\exp(-x))}{h+(\exp(x)-T)} & : x \geq \ln(T) \end{cases} \tag{3.11}$$

Still constraining $x < \ln(T)$; the system given by equation (3.10) has a single equilibrium point $(\ln(\frac{\alpha}{\beta} - \frac{\gamma m}{k\beta^2}), \ln(\frac{m}{k\beta}))$. By linearizing the system at this equilibrium point, we get

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & m \\ -\left(\alpha - \frac{\gamma m}{k\beta}\right) & -\frac{\gamma m}{k\beta} \end{bmatrix} \begin{bmatrix} x - \ln(\frac{\alpha}{\beta} - \frac{\gamma m}{k\beta^2})) \\ y - \ln(\frac{m}{k\beta}) \end{bmatrix} \tag{3.12}$$

The characteristic equation for the above matrix is $\lambda^2 + \frac{\gamma m}{k\beta}\lambda + m(\alpha - \frac{\gamma m}{k\beta}) = 0$, with roots

$$\lambda_{1,2} = \frac{-\frac{\gamma m}{k\beta} \pm \sqrt{\left(\frac{\gamma m}{k\beta}\right)^2 - 4m(\alpha - \frac{\gamma m}{k\beta})}}{2} \tag{3.13}$$

Cleary, if the equilibrium point is physically acceptable, that is $\frac{m}{k\beta} > 0$ and $(\frac{\alpha}{\beta} - \frac{\gamma m}{k\beta^2}) > 0$ and and all parameters are positve then the two eigenvalues of the above matrix satisfies $\mathrm{Re}(\lambda_{1,2}) < 0$, which makes the equilibrium point stable. We should therefore expect to see, in any stationary solution of the corresponding stochastic system when the term $H(U)$ is not present, a global maximum of the probability distribution near this point.

Indeed, by stating the SDE that arises in the absence of the term $\tilde{H}(x)$, and $h(t) = \xi(t)$, where $\xi(t)$ is a Gaussian white noise process with $\langle\xi(t)\rangle = 0$ and $\langle\xi(t)\xi(t+\tau)\rangle = D_\xi\delta(\tau)$, we get by applying Itô's formula to the transformations $X_T = \ln(U_t)$ and $Y_t = \ln(V_t)$

$$\mathrm{d}X_t = (-m + k\beta\exp(Y_t))\,\mathrm{d}t \tag{3.14a}$$

$$\mathrm{d}Y_t = (\tilde{\alpha} - \beta\exp(X_t) - \gamma\exp(Y_t))\,\mathrm{d}t + \alpha\sqrt{D_\xi}\,\mathrm{d}B_t, \tag{3.14b}$$

where $\tilde{\alpha} = \alpha - \frac{1}{2}\alpha^2 D_\xi$. This set of equations can, when

$$G(x,y) = k\beta\exp(y) - my + \beta\exp(x) - (\tilde{\alpha} - \gamma m/k\beta)x, \tag{3.15}$$

be rewritten as

$$\mathrm{d}X_t = \frac{\partial G(X_t, Y_t)}{\partial Y_t}\,\mathrm{d}t \tag{3.16a}$$

$$\mathrm{d}Y_t = -\left(\frac{\partial G(X_t, Y_t)}{\partial X_t} - \frac{\gamma}{k\beta}\frac{\partial G(X_t, Y_t)}{\partial Y_t}\right)\mathrm{d}t + \alpha\sqrt{d_\xi}\,\mathrm{d}B_t. \tag{3.16b}$$

As we have previously shown by using the Fokker-Planck equation, such a SDE has a stationary solution given by the function

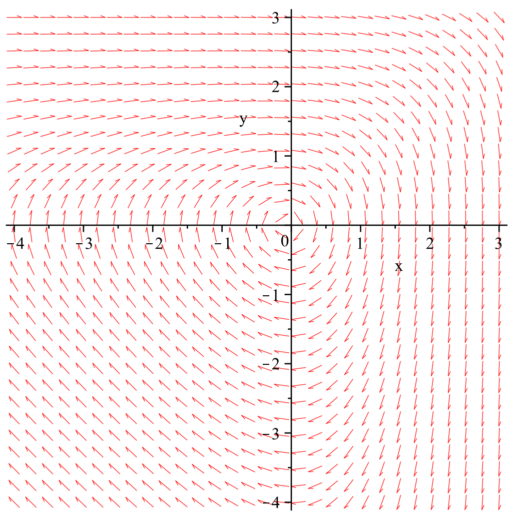
$$p_{XY}(x,y) = C\exp[-(2\gamma/k\beta D_\xi\alpha^2)G(x,y)]. \tag{3.17}$$

As long as $k, m, \beta, (\alpha - \gamma m/k\beta) > 0$, then the above equation is valid as a distribution. As $P_{XY}(x,y) \to 0$ as $(|x|,|y|) \to (\infty,\infty)$, the global maximum of must be at a critical point of $G(X,Y)$. Setting $\frac{\partial G}{\partial x} = 0$ and $\frac{\partial G}{\partial y} = 0$ gives

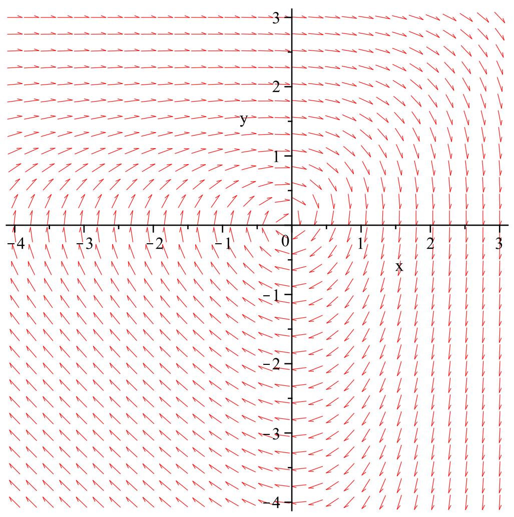
$$x = \ln\left(\frac{1}{\beta}(\alpha - \frac{1}{2}\alpha^2 D_\xi - \frac{\gamma m}{k\beta})\right) \tag{3.18a}$$

$$y = \ln\left(\frac{m}{k\beta}\right) \tag{3.18b}$$

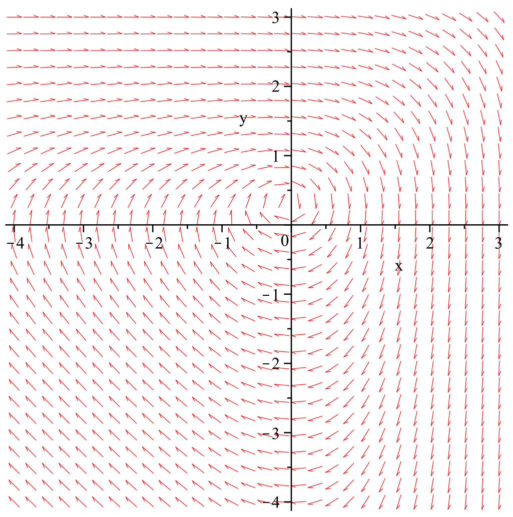
this point is close to the equilibrium point that was found in equation (3.10) when the constraint $x < \ln(T)$ was put on x, as predicted, in the sense that the y-component is equal to the equilibrium point, while the argument of the log-function in x is the same as for the argument of the x-component of the equilibrium point, but shifted with the factor $-\frac{1}{2}\alpha^2 D_\xi$. That the x-component of the maximum of the solution shifts in such a way, compared to the equilibrium point found in equation (3.10) is a strange peculiarity of Itô stochastic calculus, but nonetheless, if D_ξ is not big, they should still be close.



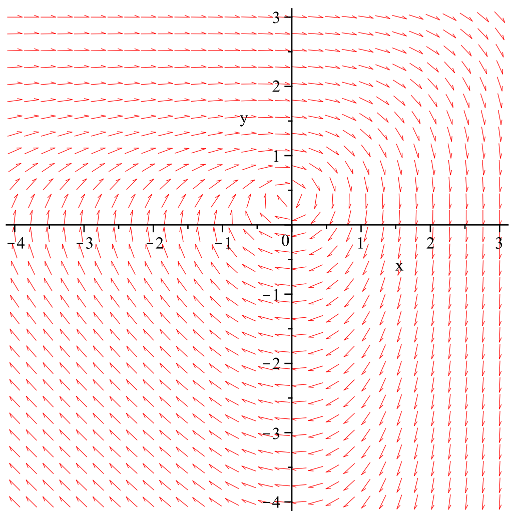
(a) No harvesting term present



(b) Harvesting term present, with $\ln(T) = 0$ and $h = 1$



(c) Harvesting term present, with $\ln(T) = -1$ and $h = 1$



(d) Harvesting term present, with $\ln(T) = -1$ and $h = 2$

Figure 3.2: Phase diagram for (3.10) with $m = k = \beta = \alpha = 1$ and $\gamma = 0.05$.

By denoting by the shifted arguments of the equilibrium points: $u_0 = (\alpha - \frac{1}{2}\alpha^2 D_\xi - \frac{\gamma m}{k\beta})/\beta$ and $v_0 = \frac{m}{k\beta}$, the stationary solution in the absence of the term $\tilde{H}(x)$, becomes, in terms of the equilibrium point:

$$p_{XY}(x, y) = C \exp[-(2\gamma/D_\xi\alpha^2)((\exp(x) - u_0 x)/k + \exp(y) - v_0 y)]. \qquad (3.19)$$

We now return to equation (3.10) and drop the restriction $x < \ln(T)$. By taking into account the the term $\tilde{H}(x)$ becomes non-zero when $x > \ln(T)$, the analysis of equilibrium becomes much more complicated, and we shall only discuss this qualitatively by looking at an example.

Numerical calculations of the phase portrait of (3.10) can be quite useful for making hypotheses of how the dynamics of the system changes when introducing

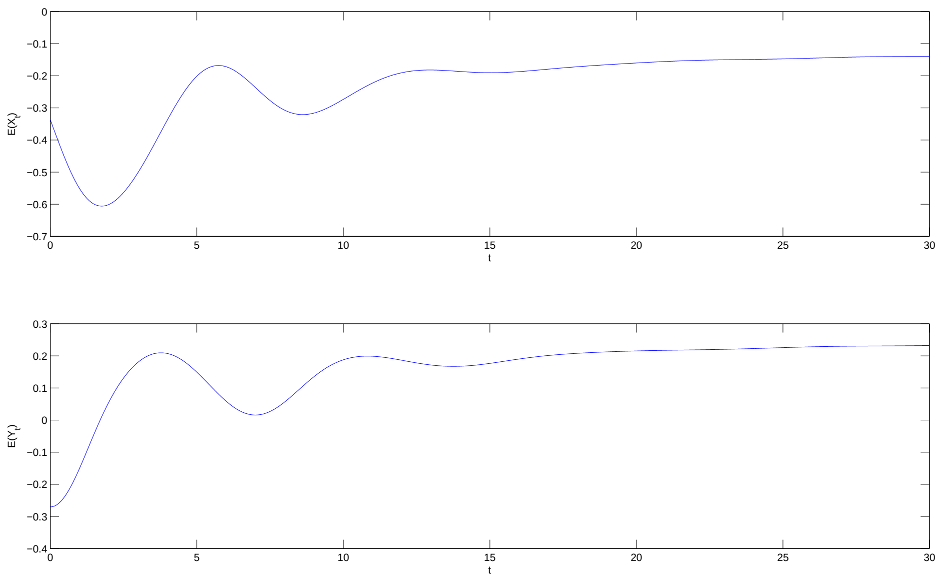


Figure 3.3: Expected value for X_t and Y_t in (3.14) calculated on $t = [0, 30]$ using PI with $\Delta t = 0.001$ on the mesh $[-6, 3] \times [-6, 3]$ with 80 grid points in each direction. The parameters used were $m = K = \beta = \alpha = 1$, $\gamma = 0.05$, $\ln(T) = -1$, $h = 2$ and $D_\xi = 0.05$, with the corresponding stationary solution given in (3.19) as initial distribution.

the harvesting term $\tilde{H}(x)$. Four phase portraits of equation (3.10) are shown in figure 3.2, with $m = k = \beta = \alpha = 1$ and $\gamma = 0.05$. It is quite interesting to see that the introduction of the harvesting term, does not seem to change the nature of the dynamics: There is seemingly still just one real-valued equilibrium point, and that equilibrium point seems to remain a stable spiral point.

As seen in figure 3.2 (b) the position of the equilibrium point does not change much by introducing a harvesting threshold at the equilibrium point $\ln(u_0) = 0$, this is not that surprising as with $x > \ln(u_0)$ and $y > \ln(v_0)$ the solution declines very rapidly towards the equilibrium point as long as $x, y, v_0, u_0 < \ln(\alpha/\gamma)$ (if $y > \ln(\alpha/\gamma)$, then $x \to -\infty$ as $t \to \infty$, and the predator will eventually becomes extinct [9].

However, when harvesting starts at $\ln(T) = -1$ as in figure 3.2(c) and 3.2(d), then there is a noticeable shift of the equilibrium point, where the value for x decreases and y increases. As $\exp(x)$ denotes the population of the predator, and the predator is the species that is being harvested, this is not surprising: When the predator is being harvested, the state of the system where the predator and the prey is in will naturally shift in a way that favors the prey. By increasing the maximal harvesting rate h, the equilibrium point is further shifted in the same direction, as seen in figure 3.2(d).

Following this discussion, it might be reasonable to conjecture that a stationary solution exists when the harvesting function is present in this example in the neighborhood of the stationary solution with no harvesting function present. Figure 3.3 shows the expected value for a simulation of this example starting at the

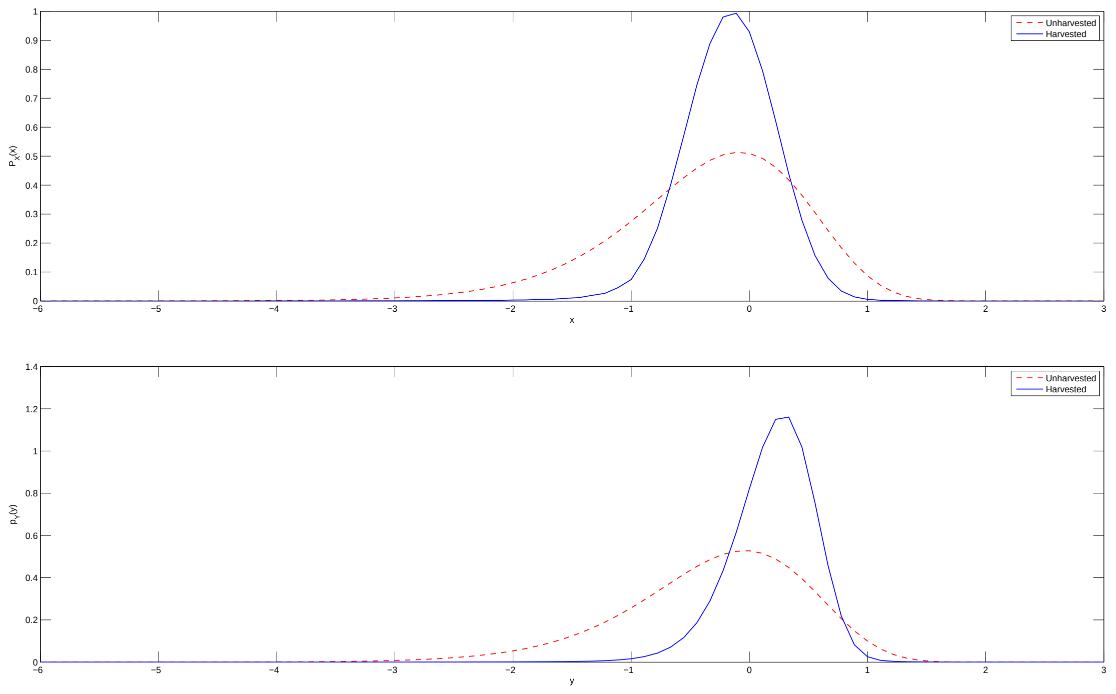


Figure 3.4: Marginal distribution for X_t and Y_t in (3.14) calculated at $t = 30$ using PI with $\Delta t = 0.001$ on the mesh $[-6, 3] \times [-6, 3]$ with 80 grid points in each direction. The parameters used were $m = K = \beta = \alpha = 1$, $\gamma = 0.05$, $\ln(T) = -1$, $h = 2$ and $D_\xi = 0.05$, with the corresponding stationary solution given in (3.19) as initial distribution. The solid blue line represents the solution when the harvesting term was present, the dashed red line represents the solution when it was not.

corresponding stationary solution when no harvesting term is present, this figure provides additional evidence that a stationary solution exists when a harvesting term is present, as there is a clear trend that the expected value seem to stabilize at a time-independent value.

Figure 3.4 shows the marginal distributions of the same simulation at $T = 30$ against the marginal distributions of the stationary solution, that the simulation was started at. Assuming that this is indeed a good approximation of the stationary solution of this example, if it exists, then there are a couple of interesting facts that can be read from this graph: First of all the global maximum of the the distributions corresponds to the equilibrium point found in the phase diagram of figure 3.2(d) with the same parameters. This is exactly as expected, and further reinforces the conjecture that a stationary solution exists.

What is slightly more surprising however, is that introducing a harvesting term seems to decrease the variance, and thus the uncertainty of the population size of both species, which is clearly seen by the sharper peak of the distribution in both X_t and Y_t. In retrospect however, since we are essentially directly manipulating and controlling the size of the population of one of the species, this might be expected.

Going back to the stationary solution given in equation (3.19), we find the corresponding distribution for $U_t = \ln(X_t)$ and $V_t = \ln(V_t)$, by calculating

$$p_{UV}(u,v) = \frac{\partial^2}{\partial u \partial v} P_{XY}(\ln(u), \ln(v)) = \frac{1}{uv} p_{XY}(\ln(u), \ln(v)), \tag{3.20}$$

where $P_{XY}(x,y) = \int_{-\infty}^{x}\int_{-\infty}^{y} p_{XY}(x,y)\,\mathrm{d}x\,\mathrm{d}y$ is the joint cumulative distribution of X_t and Y_t.

On applying the normalization condition on the first quadrant of the (u,v)-plane, we thus obtain the stationary distribution of the solution $p_{UV}(u,v) = p_U(u)P_V(v)$ where it turns out that U_t and V_t are independently Γ-distributed [9], with distributions given by

$$p_U(u) = [(\delta/k)^{\delta u_0/k} u^{\delta u_0/k - 1} \exp(-\delta u/k)]/\Gamma(\delta u_0/k) \tag{3.21}$$
$$p_V(v) = [(\delta)^{\delta v_0} v^{\delta v_0 - 1} \exp(-\delta v)]/\Gamma(\delta v_0) \tag{3.22}$$

where $\Gamma(\cdot)$ is the Euler Gamma function, $u_0 = (\alpha - \frac{1}{2}\alpha^2 D_\xi - \gamma m/k\beta)/\beta$, $v_0 = m/k\beta$ and $\delta = 2\gamma/\alpha^2 D_\xi$.

It can readily be verified by insertion into the Fokker-Planck equation, that this is indeed a stationary solution for (3.7), when $h(t)$ is Gaussian white noise. In fact, we can show that this is the only integrable stationary distribution of the form $p_{UV}(u,v) = u^{k_1-1}v^{k_2-1}\exp(-u\theta_1 - v\theta_2)$, by calculating

$$-\frac{\partial}{\partial u}[(-m + k\beta v)u p_{UV}(u,v)] = mk_1 - k\beta k_1 y - m\theta_1 x + k\beta\theta_1 yx \tag{3.23}$$
$$\begin{aligned}-\frac{\partial}{\partial v}[(\alpha - \beta u - \gamma v)v p_{UV}(u,v)] =& -\alpha k_2 + ((1+k_2)\gamma + \alpha\theta_2)y + \beta k_2 x \\ & - \beta\theta_2 xy - \gamma\theta_2 y^2\end{aligned} \tag{3.24}$$
$$\frac{\partial^2}{\partial v^2}[\frac{1}{2}\alpha^2 D_\xi v^2 p_{UV}(u,v)] = \frac{1}{2}\alpha^2 D_\xi (k_2 + k_2^2 - 2(1+k_2)\theta_2 y + \theta_2^2 y^2). \tag{3.25}$$

By setting $-\frac{\partial}{\partial u}[(-m+k\beta v)u p_{UV}(u,v)] - \frac{\partial}{\partial v}[(\alpha - \beta u - \gamma v)v p_{UV}(u,v)] + \frac{\partial^2}{\partial v^2}[\frac{1}{2}\alpha^2 D_\xi v^2 p_{UV}(u,v)] = 0$, and collecting terms of equal power, we get the following set of equations

$$mk_1 - \alpha k_2 + \frac{1}{2}\alpha^2 D_\xi (k_2 + k_2^2) = 0 \quad \text{(constant term)} \tag{3.26}$$
$$\beta k_2 - m\theta_1 = 0 \quad (x) \tag{3.27}$$
$$k\beta\theta_1 - \beta\theta_2 = 0 \quad (xy) \tag{3.28}$$
$$-k\beta k_1 + \gamma(1+k_2) - \alpha^2 D_\xi (1+k_2)\theta_2 + \alpha\theta_2 = 0 \quad (y) \tag{3.29}$$
$$\frac{1}{2}\alpha^2 D\theta_2^2 - \gamma\theta_2 = 0 \quad (y^2). \tag{3.30}$$

The last equation gives that either: $\theta_2 = 0$ or $\theta_2 = 2\gamma/\alpha^2 D_\xi = \delta$, the first option gives from the second and third equation that $k_2 = \theta_1 = 0$, inserting this into the first and fourth equation however, gives two equations for k_1 that does not agree unless $\gamma = 0$, but in this case, we obtain the solution $p_{UV}(u,v) = (uv)^{-1}$ which is not integrable, thus, we must have $\theta_2 = \delta$.

Proceeding with $\theta_2 = \delta$, we obtain uniquely from the second and the third equation that $\theta_1 = \delta/k$, $k_2 = \delta(m/k\beta) = \delta v_0$, inserting this into the first and fourth equation, gives two equal expressions for k_1, which gives uniquely that $k_1 = (\delta/k)(\alpha - \frac{1}{2}\alpha^2 D_\xi \gamma m/k\beta)/\beta = \delta u_0/k$.

We shall conclude this section on stationary solutions with an example, showing how PI performs with equation (3.7).

Example 3.1. Consider the SDE

$$\mathrm{d}U_t = (-mU_t + k\beta U_t V_t)\,\mathrm{d}t \tag{3.31a}$$

$$\mathrm{d}V_t = (\alpha V_t - \beta U_t V_t - \gamma V_t^2)\,\mathrm{d}t + \alpha\sqrt{D_\xi} V_t\,\mathrm{d}B_t \tag{3.31b}$$

where $U_t, V_t \geq 0$. Implementation of this SDE requires some extra care, as the diffusion term $\alpha\sqrt{D_\xi}v \to 0$ as $v \to 0$. By writing up the transition kernel that results from an Euler approximation, we get

$$k(u', v', u, v, \Delta t) = \delta(u' - u - r_1(u,v)\Delta t)\frac{\exp\left(\frac{(v'-v-r_2(u,v)\Delta t)^2}{2\alpha^2 D_\xi v^2 \Delta t}\right)}{\sqrt{2\pi\alpha^2 D_\xi v^2 \Delta t}}. \tag{3.32}$$

By doing the usual transformation $(\tilde{u}, \tilde{v}) = (u + r_1(u,v)\Delta t, v + r_2(u,v))\Delta t = g(u,v)$ we obtain

$$k(u', v', u, v, \Delta t) = \delta(u' - \tilde{u})\frac{\exp\left(\frac{(v'-\tilde{v})^2}{2\alpha^2 D_\xi (g_v^{-1}(\tilde{u},\tilde{v}))^2 \Delta t}\right)}{\sqrt{2\pi\alpha^2 D_\xi (g_v^{-1}(\tilde{u},\tilde{v}))^2 \Delta t}}. \tag{3.33}$$

where $v = g_v^{-1}(\tilde{u}, \tilde{v})$ is the unique real number such that together with $u = g_u^{-1}(\tilde{u}, \tilde{v})$ we have $(\tilde{u}, \tilde{v}) = g(u,v)$, we approximate v by $(u,v) = g^{-1}(\tilde{u},\tilde{v}) \approx (\tilde{u} - r_1(\tilde{u},\tilde{v})\Delta t, \tilde{v} - r_2(\tilde{u},\tilde{v})\Delta t)$. The probability density of the solution at the next time step then becomes

$$p(u', v', t') = \int_{\mathbb{R}} \frac{\exp\left(\frac{(v'-\tilde{v})^2}{2\alpha^2 D_\xi (\tilde{v} - r_2(u',\tilde{v})\Delta t)^2 \Delta t}\right)}{\sqrt{2\pi\alpha^2 D_\xi (\tilde{v} - r_2(u',\tilde{v})\Delta t)^2 \Delta t}} p(g^{-1}(u',\tilde{v}))|J_{g^{-1}}|\,\mathrm{d}\tilde{v} \tag{3.34}$$

where $|J_{g^{-1}}|$ is the determinant of the Jacobian of the inverse transformation. One quickly realizes that the transition probability, which can for small Δt be approximated by $\exp((v'-\tilde{v})^2/2\alpha^2 D_\xi \tilde{v}^2 \Delta t)/\sqrt{2\pi\alpha^2 D_\xi \tilde{v}^2 \Delta t}$ is highly localized and peaked around $\tilde{v} = v'$ as shown for some sample functions in figure 3.5(a), but is not integrable over $\mathbb{R}^+$ as the function is asymptotically bounded from below by $C/\tilde{v}$ as $\tilde{v} \to \infty$ for some constant $C \in \mathbb{R}^+$.

That the transition probability is non-integrable is not a problem in terms of evaluating equation (3.34) as the integrand is dominated by $p(u,v)$, which is integrable. It does however present a challenge in terms of choosing quadrature points for the numerical integration. Due to non-integrability of the transition probability, we must integrate over the entire interval $v \in [V_{\min}, V_{\max}]$ in which we represent the distribution of the numerical solution in order to keep reasonable bounds on the error due to truncating the function: We can not simply choose quadrature points around the peak of the transition probability, as we can in the case of the diffusion term being constant.

Due to the peaked nature of the transition probability however, we should choose quadrature points densely around the peak. Having uniformly distributed quadrature points chosen dense over the interval $[V_{\min}, V_{\max}]$ is not prudent however, as it quickly leads to excessive computation time: As we do not need densely distributed quadrature points at the tails of the transition probability, a better idea would be to choose a non-uniform grid with a dense distribution around the peaks: More precisely, we would like to implement an algorithm that chooses the quadrature points adaptively according to the nature of the transition probability function.

The algorithm which will be used, and which will shortly be described is based on a step-size control which is commonly implemented for Runge-Kutta methods: It is an idea due to Fehlberg [10], and for a more thorough explanation, the reader is referred to [10].

The basic idea in this case is to choose two different quadrature rules, one of order p,and one of order $p+1$ to approximate $\int_x^{x+\Delta x} f(x)\,\mathrm{d}x$ for some function $f(x)$. If we let $\tilde{f}_1$ and $\tilde{f}_2$ denote the value of the approximate value for the first and the second quadrature rule, such that

$$\tilde{f}_1 = \sum_{i=1}^{N} a_i f(x_i) = \int_x^{x+\Delta x} f(x)\,\mathrm{d}x + \mathcal{O}(h^p) \tag{3.35}$$

$$\tilde{f}_2 = \sum_{i=1}^{N} b_i f(x_i) = \int_x^{x+\Delta x} f(x)\,\mathrm{d}x + \mathcal{O}(h^{p+1}) \tag{3.36}$$

then $|f_1 - f_2|$ should be a pth order approximation of the error, that is

$$|f_1 - f_2| = e + \mathcal{O}(\Delta x^p), \tag{3.37}$$

where e is the true difference in absolute value between the exact and the approximate integration for f_1 and f_2. If $e_1 = |f_1 - f_2|$ and e_2 is the bound which we do not want the error to exceed when integrating over $[x, x+\Delta x]$. Then we should expect to have approximately the following relationship between the current step size Δx_1 and the desired step size Δx_2 where the error should be approximately e_2

$$\frac{e_2}{e_1} = \left(\frac{\Delta x_2}{\Delta x_1}\right)^p \tag{3.38}$$

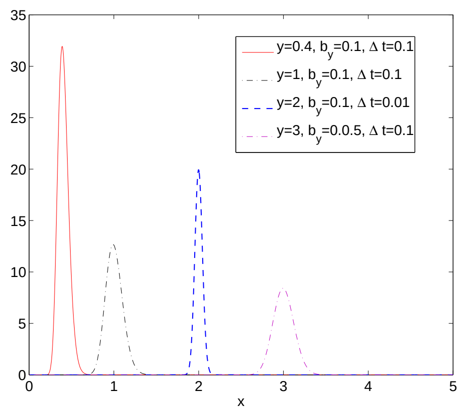
Suppose now that we for some step size Δx_1 calculate the error $e_1 \leq e_2$, then naturally we want to keep the current step size, but we also want to increase the step size for the next step, so that we do not take an unnecessary number of steps. Conversely, if $e_1 > e_2$, we want to redo the step with a decreased step size. We choose the new step size with a slight modification of equation (3.38):

$$\Delta x_2 = P\left(\frac{e_2}{e_1}\right)^{1/p} \Delta x_1 \tag{3.39}$$

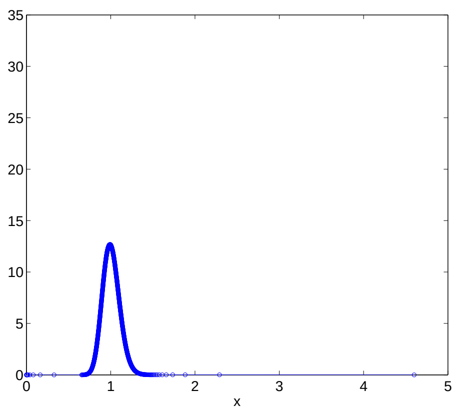
where $P \in (0,1)$ is known as the pessimistic constant: We want to choose a Δx_2 slightly lower than the predicted one, in order to not waste too many steps, it is typically chosen between 0.7 and 0.9.

The function f we integrate, and which we choose the quadrature points from is a slightly modified version of the transition probability, and is given by

$$f(v) = \frac{\exp\left(\frac{(v'-v)^2}{2\alpha^2 D_\xi v^2 \Delta t}\right)}{\sqrt{2\pi\alpha^2 D_\xi v^2 \Delta t}}. \tag{3.40}$$



(a)



(b)

Figure 3.5: Illustration of the shape of transition probabilities when using the Euler approximation in (a), and illustration of the distribution of quadrature points, when using the adaptive algorithm in (b) using $e_2 = 10^{-3}$. The parameters used in all cases were $\alpha = m = k = \beta = 1$, $\gamma = 0.05$ and $D_\xi = 0.01$, on the grid $[0, 5] \times [0, 5]$.

This function is the same as the transformed transition kernel, but omits the expensive to calculate RK-step in the denominator of the exponential argument and the denominator of the transition kernel. For small Δt, this should be a reasonably good approximation. The implementation of the algorithm using the first order Euler's method

$$f_1 = \Delta x f(x_1) \tag{3.41}$$

and the second order trapezoidal rule

$$f_2 = \frac{\Delta x}{2}(f(x_1) + f(x_2)). \tag{3.42}$$

Figure 3.5(b) illustrates the distribution of quadrature points using the described algorithm with the Euler-trapezoidal rule pair. We see that we achieve the distribution of points we desired, with quadrature points densely distributed around the peak, and a much sparser distribution elsewhere.

In figure 3.6 the results from a simulation using the algorithm just described, and as can be seen, the computed PI-solution does not preserve the stationary solution very well, as a significant deviation can be seen for the computed function at $t = 60$ from the stationary solution. The computed solution is peculiar, in that it does not seem to fundamentally change the shape or the expected value of the distribution, but rather the variance: It seems to stabilize at a stationary solution with a lower value for α.

The author has tried to figure out the reason for this discrepancy, and has worked with a couple of theories. The first theory was that this perhaps is due to the singularity of the denominator of the expontential argument in the transition kernel as $v \to 0$. Since the denominator goes to infinity as $\mathcal{O}(v^{-2})$ there might be a significant round-off error in the solution. To test out this theory, an implementation

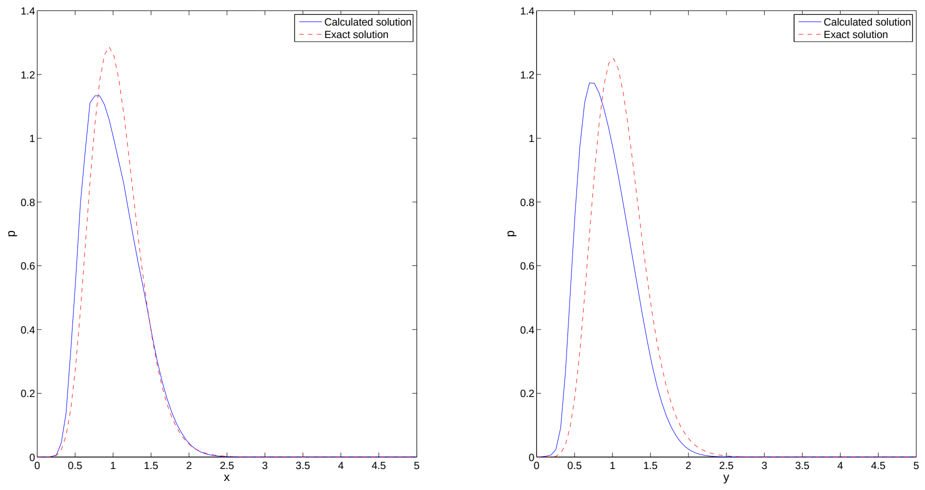


Figure 3.6: Marginal distribution for U_t on the left hand-side and V_t on the right-hand side in (3.31) calculated at $t = 60$ using PI using an Euler-discretization and the algorithm for choosing quadrature points described in example 3.1 with $e_2 = 10^{-3}$, $\Delta t = 0.01$ on the mesh $[0,5] \times [0,5]$ with 80 grid points in each direction. The parameters used were $m = K = \beta = \alpha = 1$, $\gamma = 0.05$, $D_\xi = 0.01$, with the corresponding stationary solution as initial distribution. The solid blue line represents the PI-solution at $t = 60$, the dashed red line represents the stationary solution.

of this algorithm, using a Milstein scheme for time-discretization rather than the Euler scheme. The transition kernel in this case becomes

$$k(u', v', t', u, v, t) = \delta(u' - v - r(u,v)\Delta t \tilde{k}(u', v', u, v, t) \tag{3.43}$$

where

$$\begin{aligned}
\tilde{k}(u', v', t', u, v, t) &= \frac{[k_2(v'-c)]^{-1/2}}{2\sqrt{2\pi\Delta t}} \sum_{j=1}^{2} e^{\left[-\frac{(\sqrt{(v'-c)/k_2}+(-1)^j k_1)^2}{2\Delta t}\right]} \qquad (3.44)\\
c &= v + r_2(u,v)\Delta t - k_2 \Delta t - k_1^2 k_2\\
k_1 &= D_\xi^{-1/2}\\
k_2 &= \frac{1}{2} D_\xi v.
\end{aligned}$$

Here, there is also a singularity in the denominator term of the exponential argument at $v = 0$, but here it only goes to infinity as $\mathcal{O}(v^{-1})$, if the theory of round off error is correct, we should see a significantly more accurate solution, when using a transition kernel based on the Milstein scheme.

In using the Milstein scheme, we use an algorithm for choosing the quadrature points completely analogous to the one we used when using an Euler discretization, but we use another function f, which we choose the quadrature points from, which

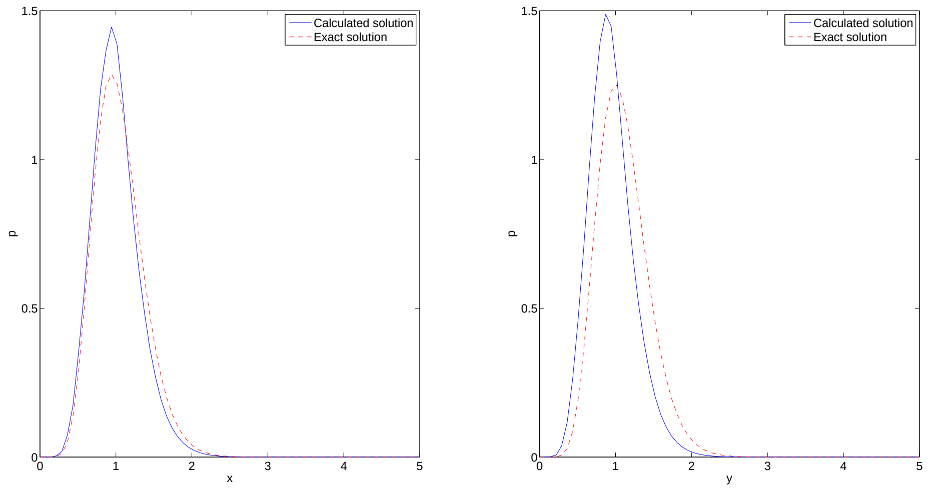


Figure 3.7: Marginal distribution for U_t on the left-hand side and V_t on the right-hand side in (3.31) calculated at $t = 60$ using PI using the Milstein scheme and the algorithm for choosing quadrature points described in example 3.1 with $e_2 = 10^{-3}$ $\Delta t = 0.01$ on the mesh $[0, 5] \times [0, 5]$ with 80 grid points in each direction. The parameters used were $m = K = \beta = \alpha = 1$, $\gamma = 0.05$, $D_\xi = 0.01$, with the corresponding stationary solution as initial distribution. The solid blue line represents the PI-solution at $t = 60$, the dashed red line represents the stationary solution.

is given by the expression

$$f(v) = \frac{[k_2(v' - c)]^{-1/2}}{2\sqrt{2\pi\Delta t}} \sum_{j=1}^{2} e^{\left[-\frac{(\sqrt{(v'-c)/k_2}+(-1)^j k_1)^2}{2\Delta t}\right]} \tag{3.45}$$

where

$$\begin{aligned} c &= v - k_2\Delta t - k_1^2 k_2 \\ k_1 &= D_\xi^{-1/2} \\ k_2 &= \frac{1}{2} D_\xi v. \end{aligned}$$

which results once again from doing the usual transformation $(\tilde{u}, \tilde{v}) = (u + r_1(u, v)\Delta t, v + r_2(u, v)\Delta t)$, and then omitting the expensive to calculate terms $r(u, v)$.

Figure 3.7 the results from using the Milstein scheme are shown, and also here, there is a significant difference between the stationary solution that the simulation began with, and the obtained solution at $t = 60$, though it is visibly closer than with the Euler discretization.

This leaves us with three possibilities, as far as the author can see. The first possibility is that the error is indeed due to a large round-off error which could still be possible: Even though the singularity grows slower in the exponential argument of the Milstein scheme transition than the singularity in the exponential argument of the Euler scheme, there is still a singularity in the denominator of the transition kernel, which grows us $\mathcal{O}(v^{-1})$ both when using the Milstein scheme, and when using the Euler scheme.

A second possibility is that this is a an example of an SDE for which the PI-solution does not converge, an assumption which is made somewhat stronger by the fact that the PI-solution, while using both the Euler scheme and the Milstein scheme seemed to diverge when choosing Δt smaller than 0.01. The SDE does certainly not satisfy the criteria for convergence of the PI-solution, as an existence and uniqueness of the solution is not guaranteed, since the deterministic term of the SDE does not satisfy the Lipschitz-criterion.

A third possibility, which the author finds more likely than the second possibility, is that there is simply a bug in the program which was written to simulate the solution, or an implementation mistake: Even though there is a possibility of a large round-off error, the grid was intentionally chosen coarse, such that the grid points with the smallest values for v wouldn't be too small. In addition, the case of $v = 0$ was handled analytically by adding the constraint of the PI-solution that $p(u, 0) = 0$, which is true for the stationary solution in this example. The author has looked very carefully throughout the whole code for an implementation error or a bug, but was unsuccessful in this regard. The entire source code, which can be run as a stand-alone C++ program has therefor been included in apppendix A.

3.4 Non-stationary solutions

In the previous section we considered $h(t)$ when it was a pure Gaussian white noise process. We shall now turn our attention to temporal variations with hidden time-depended periodicity. That is, $h(t)$ of the form

$$h(t) = \lambda \sin(\upsilon t) + \xi(t) \tag{3.46}$$

where $\xi(t)$ still is a Gaussian white noise process with $\langle \xi(t) \rangle = 0$ and $\langle \xi(t)\xi(t+\tau) \rangle = D_\xi \delta(\tau)$. The transformed SDE with no harvesting term thus becomes

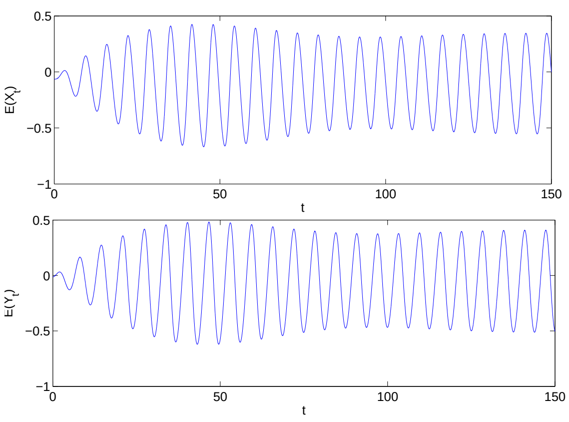
$$\mathrm{d}X_t = (-m + k\beta \exp(Y_t))\,\mathrm{d}t \tag{3.47a}$$

$$\mathrm{d}Y_t = (\alpha(1 + \lambda \sin(\upsilon t)\frac{1}{2}D_\xi \alpha^2) - \beta \exp(X_t) - \gamma \exp(Y_t))\,\mathrm{d}t + \alpha\sqrt{D_\xi}\,\mathrm{d}B_t \tag{3.47b}$$

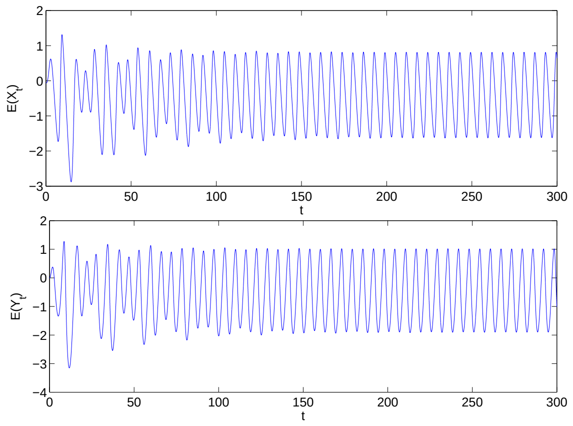
Due to the non-autonomous nature of this SDE, it can be shown quite easily that the corresponding deterministic ODE (i.e. we set $D_\xi = 0$) has no equilibrium points. We should therefore not expect any stationary solution for the SDE to exist, we do however get periodic non-stationary solutions [9].

A realization of the solution, using PI confirms the existence of the periodic non-stationary solution. Figure 3.8 shows the time-evolution of the expected value with respect to X_t and Y_t, and as can be seen, the expected value seems to be a sinusoidal oscillation with an oscillating amplitude which asymptotically becomes stable at a fixed value.

The author counted approximately 24.5 periods for $\lambda = 0.05$ over $t \in [0, 150]$ and approximately 48.5 periods for $\lambda = 0.5$ over $t \in [0, 300]$, which implies a period of 6.12 and 6.18 respectively, which is just short of 2π. This suggests that the oscillations of the expected values are independent of α, and is driven by the sinusoidal term.



(a) $\lambda = 0.05$ on the mesh $[-1.5, 1.5] \times [-1.5, 1.5]$ with 100 grid points in each direction



(b) $\lambda = 0.5$ on the mesh $[-4, 2] \times [-4, 2]$ with 80 grid points in each direction

Figure 3.8: Calculated expected value of the PI-realization of equation (3.47) with $\Delta t = 0.01$, using the parameters $\alpha = m = k = \beta = 1$, $\gamma = 0.05$, $\upsilon = 1$ and $D_\xi = 0.05$. The initial distribution used is given in equation (3.19) with the relevant parameters

It is also interesting to note that the function seems to oscillate about the function values which are the maximal value of the Lotka-Volterra equation, with only pure white noise as the variation of the growth rate of the prey for the same parameters. That is in figure 3.8, the expected value for X_t and Y_t oscillates around $\mathrm{E}(X_t) = \ln(0.925) \approx -0.078$ and $\mathrm{E}(Y_t) = \ln(1) = 0$, which are, respectively, the x- and y-coordinates of the global maximum of the corresponding stationary solution, when the variation in the growth rate of the prey is driven only by white noise.

The oscillations in the amplitude may be due to natural oscillations in the system which occurs when it is not in a state of equilibrium, and when the system eventually reaches a periodic equilibrium state the large oscillations die out. It also seems that the oscillations in the amplitude are present longer when $\lambda = 0.5$ than with $\lambda = 0.05$, as can be seen in figure 3.8(b) where the big oscillations are quite pronounced, even when $t = 150$, the physical reason for this might be a delayed reaction from the predator in response to the more rapid change of growth of the prey. As a result, the large oscillations also takes a longer time to die out.

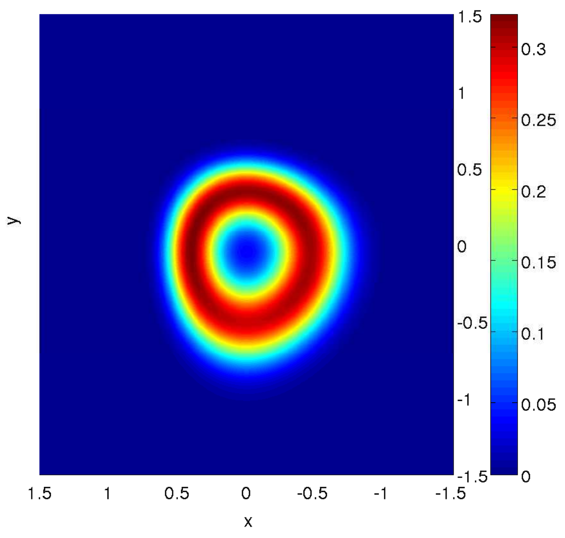
Figure 3.9 shows the time averaged solution on $t \in [100, 150]$ for figure 3.9(a) and on $t \in [250, 300]$ for figure 3.9(b), for the same calculations used in figure 3.8, and as the expected value suggested, we see that the solutions typically go in elliptical-like orbits with bigger radii as α is chosen bigger.

We now return to the transformed Lotka-Volterra equation with harvesting and random temporal variations in the growth rate of prey

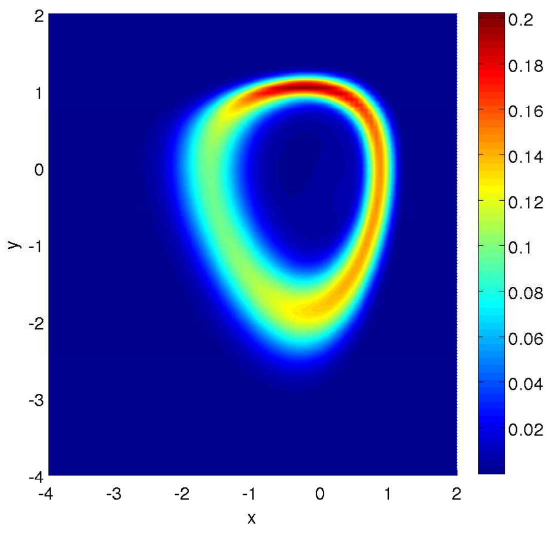
$$\mathrm{d}X_t = (-m + k\beta \exp(Y_t) - \tilde{H}(X_t))\,\mathrm{d}t \tag{3.48a}$$

$$\mathrm{d}Y_t = (\tilde{\alpha} - \beta \exp(X_t) - \gamma \exp(Y_t))\,\mathrm{d}t + \alpha\sqrt{D_\xi}\,\mathrm{d}B_t, \tag{3.48b}$$

where $\tilde{H}(x)$ is given in equation (3.11). We are interested in seeing qualitatively how the the harvesting term affects the solution of the Lotka-Volterra equation with



(a) $\lambda = 0.05$ on the mesh $[-1.5, 1.5] \times [-1.5, 1.5]$ with 100 grid points in each direction



(b) $\lambda = 0.5$ on the mesh $[-4, 2] \times [-4, 2]$ with 80 grid points in each direction

Figure 3.9: Time averaged solution of the PI-realization of equation (3.47) with $\Delta t = 0.01$, using the parameters $\alpha = m = k = \beta = 1$, $\gamma = 0.05$, $\upsilon = 1$ and $D_\xi = 0.05$. The initial distribution used is given in equation (3.19) with the relevant parameters

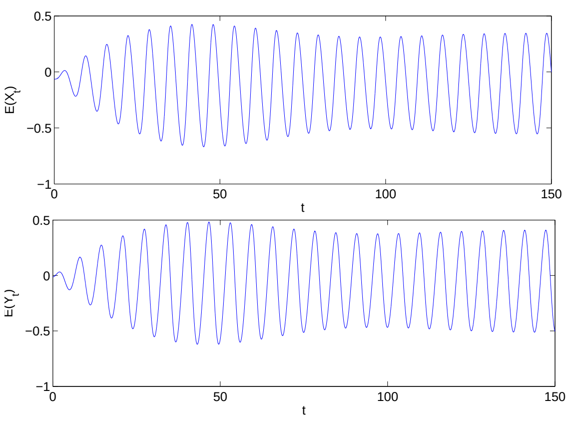
random temporal variations in the growth rate of the prey.

We can make some a priori guesses of how the solution will behave. We have good reasons to believe that we will asymptotically obtain a periodic solution, as that was the case with the Lotka-Volterra equation with random temporal variations in the growth rate of the prey. The sinusoidal seemed to be the driving force behind the small oscillations, and we should expect to at least see them in the solution when the harvesting term is present.

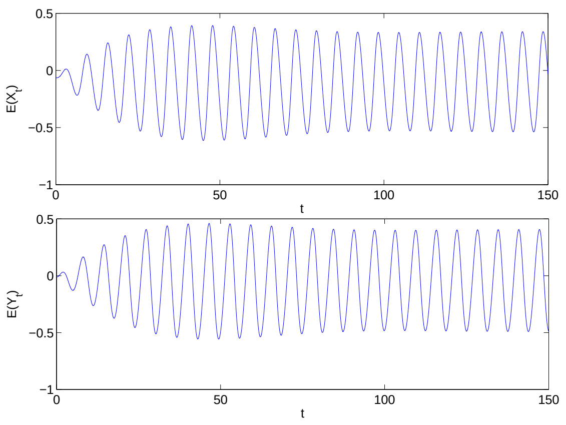
We should expect to see a noticeable change of the radii of the ellipses and the focus towards bigger values of y and smaller values for x, because x, which represents the population of the predator is being harvested.

In figure 3.10(b) the results from simulations of equation (3.48) for the given parameters can be seen, which is compared side-to-side with the simulation of the same equation with the same parameters but without the harvesting term in figure 3.10(a). With the parameter $\lambda = 0.05$ for the temporal variation in the growth rate of the prey, there is not a large difference between the two graphs: The amplitude of the expected value, can be seen to be a little more damped, and stabilizes a little quicker in figure 3.10(b) where the harvesting term is present, but except for that there are little differences.

In the analog figure 3.11 with the same parameters but where $\lambda = 0.5$ however, one can see that there is a significant damping of the oscillation of the amplitude of the expected values. It is not very surprising that the harvesting term has a damping effect on the amplitude of the expected values, because the damping term can essentially be thought of as a force that acts against movement of X_t and thus also Y_t towards extreme values, preventing the initial big oscillations in the



(a) $\lambda = 0.05$ on the mesh $[-1.5, 1.5] \times [-1.5, 1.5]$ with 100 grid points in each direction, no harvesting term



(b) $\lambda = 0.05$ on the mesh $[-1.5, 1.5] \times [-1.5, 1.5]$ with 100 grid points in each direction, no harvesting term

Figure 3.10: Calculated expected value of the PI-realization of equation (3.48) with $\Delta t = 0.01$, using the parameters $\alpha = m = k = \beta = 1$, $\gamma = 0.05$, $\upsilon = 1$, $D_\xi = 0.05$, $\ln(T) = -1$ and $h = 2$. The initial distribution used is given in equation (3.19) with the relevant parameters.

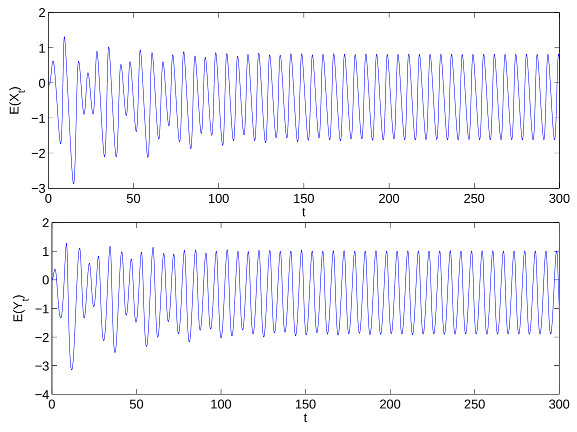
amplitude.

We also see a slight tendency in figure 3.10(b), but more so in figure 3.11(b) that when the amplitude of the expected values stabilize at a constant value, that the minimum values for $\mathrm{E}(X_t)$ become slightly lower, and the and the maximal values of $\mathrm{E}(Y_t)$ becomes higher.

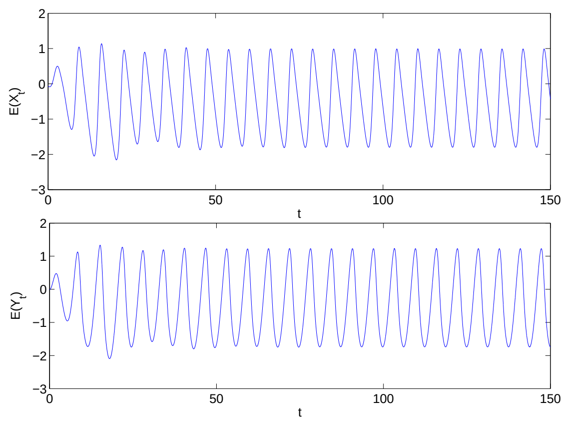
This is in line with what we discovered for the stationary solution and the equilibrium point analysis for the Lotka-Volterra equation, when the variation in the growth of the predator is a pure white noise process: When harvesting occurs, there is a slight shift negative shift in X_t, the population of the predator, and a larger gain in Y_t, the population of the prey. The reason for the discrepancy in shift is likely the following: The prey population was held in check primarily by the population of the predator, but since the population of the predator is controlled by the harvesting, there is a gain in the population of the prey. This gain in population in the prey, results in increased growth of the predator, which balances out much of the negative effect of the growth of the predator which is introduced by the prey, therefore there is only a slight net loss in the maximal population of the predator.

Figure 4.6 shows plots of the time averaged solution for $t \in [100, 150]$ in figure 4.6(a) and 4.6(b), and the time averaged solution for $t \in [250, 300]$ in figure 4.6(b) and 4.6(d), with a corresponding side-by-side comparison of the solution without the harvesting term in figure 4.6(a) and 4.6(c). We see that introducing the harvesting term, we get a reduction of the variance of the solution and the path of the orbital becomes more sharply defined. This is a phenomenon that we also saw in the stationary solution the Lotka-Volterra equation, when the variation in the growth of the prey is controlled by pure white noise, when introducing a harvesting term, and the reasons are most likely the same.

The shape however, remains largely unchanged. We see interestingly that figure



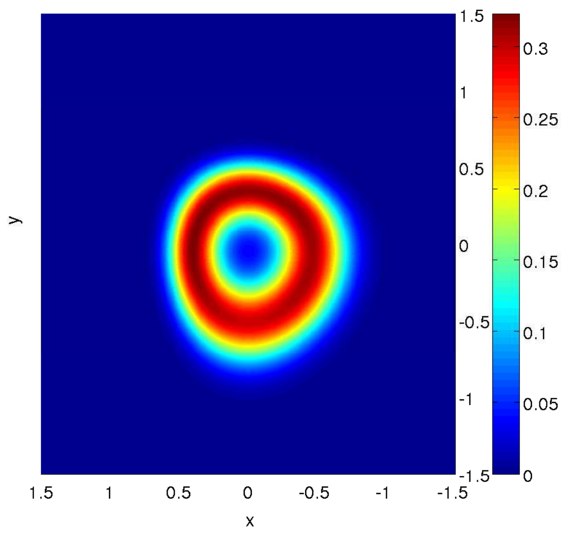
(a) $\lambda = 0.5$ on the mesh $[-4, 2] \times [-4, 2]$ with 80 grid points in each direction, no harvesting term

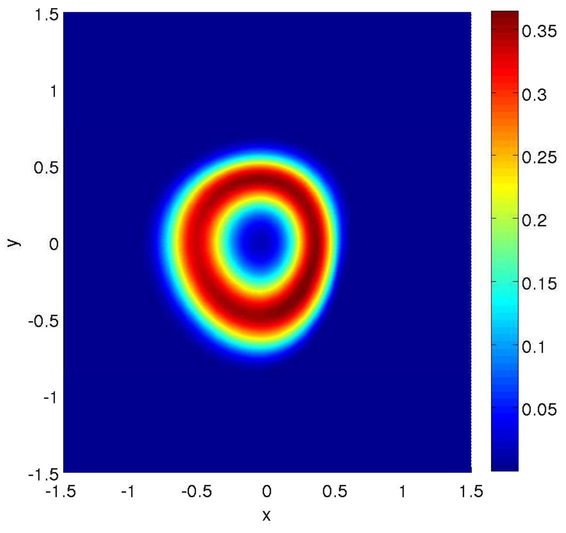


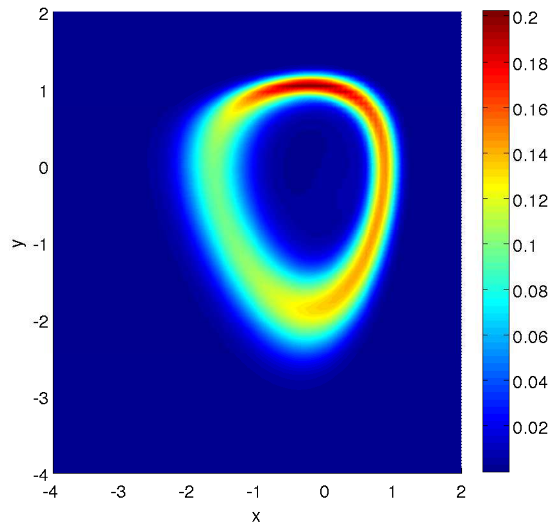
(b) $\lambda = 0.5$ on the mesh $[-4, 2] \times [-4, 2]$ with 80 grid points in each direction, with harvesting term

Figure 3.11: Calculated expected value of the PI-realization of equation (3.48) with $\Delta t = 0.01$, using the parameters $\alpha = m = k = \beta = 1$, $\gamma = 0.05$, $\upsilon = 1$, $D_\xi = 0.05$, $\ln(T) = -1$ and $h = 2$. The initial distribution used is given in equation (3.19) with the relevant parameters.

4.6(b) is almost a mirror reflection of figure 4.6(a), and in both figure 4.6(b) and 4.6(d) the maximal values of the solutions shift from high values of y to low values of y. Except for that, there are not many differences. This shows that the a priori assumption that the radii of the ellipses would change, increasing in the direction of y and decreasing in the direction of x was somewhat false.


(a) $\lambda = 0.05$ on the mesh $[-1.5, 1.5] \times [-1.5, 1.5]$ with 100 grid points in each direction, no harvesting term


(b) $\lambda = 0.05$ on the mesh $[-1.5, 1.5] \times [-1.5, 1.5]$ with 100 grid points in each direction, with harvesting term


(c) $\lambda = 0.5$ on the mesh $[-4, 2] \times [-4, 2]$ with 80 grid points in each direction, no harvesting term

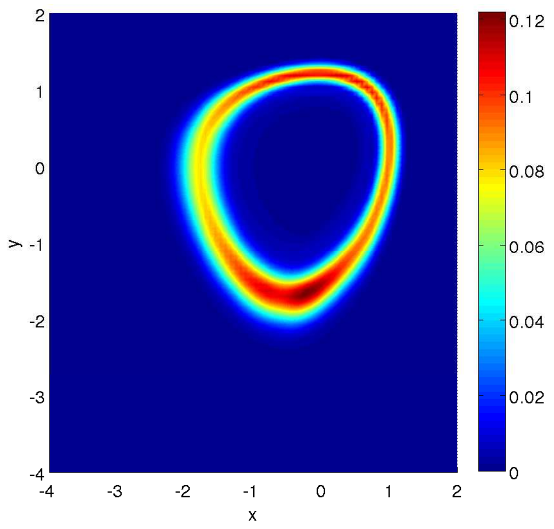

(d) $\lambda = 0.5$ on the mesh $[-4, 2] \times [-4, 2]$ with 80 grid points in each direction, with harvesting term

Figure 3.12: Time averaged solution of the PI-realization of equation (3.48) with $\Delta t = 0.01$, using the parameters $\alpha = m = k = \beta = 1$, $\gamma = 0.05$, $\upsilon = 1$, $D_\xi = 0.05$, $\ln(T) = -1$ and $h = 2$. The initial distribution used is given in equation (3.19) with the relevant parameters.

Chapter 4

An adaptive algorithm

In this section, we will try to develop an algorithm that is fast but also accurate by exploiting, and building on specific properties of SDE that we will be looking at. The type of SDE that we will consider, are two dimensional autonomous SDE with noise in one dimension of the type

$$\mathrm{d}X_t = a_1(X_t, Y_t)\,\mathrm{d}t \tag{4.1a}$$

$$\mathrm{d}Y_t = a_2(X_t, Y_t)\,\mathrm{d}t + \sqrt{D_\xi}\,\mathrm{d}B_t \tag{4.1b}$$

where we will look at the logarithm-transformed Lotka-Volterra equation

$$\mathrm{d}X_t = (-m + k\beta\exp(Y_t))\,\mathrm{d}t \tag{4.2a}$$

$$\mathrm{d}Y_t = (\alpha - \frac{1}{2}\alpha^2 D_\xi - \beta\exp(X_t) - \gamma\exp(Y_t))\,\mathrm{d}t + \alpha\sqrt{D_\xi}\,\mathrm{d}B_t \tag{4.2b}$$

which requires extra care in the space domain, for reasons that will be made clearer later.

The method developed here, will make strong use of the time-independent nature of the Lotka-Volterra equation, to show how we can pre-calculate most of the heavy calculations and greatly reduce the number of calculations for each subsequent time step. We will also implement adaptive mesh refinements, which is refined according to the current information of the stationary solution. We will also use the stability of the stationary solution, that is that we obtain a stationary solution, regardless of initial smooth condition, to hopefully gain gradually better estimates of the true stationary solution, by gradually decreasing the time step size.

4.1 PI for an autonomous SDE

Recall that when we apply the PI-operator to the solution $p(x, y, t)$, in order to get the solution at the next time step $p(x', y', t')$ for the equation of the type given in (4.1), we have to calculate

$$p(x', y', t') = \int_{\mathbb{R}} \frac{\exp\left(\frac{(y'-\tilde{y})^2}{2D_\xi\Delta t}\right)}{\sqrt{2\pi D_\xi \Delta t}} p(g^{-1}(x', \tilde{y}))|J_{g^{-1}}|\,\mathrm{d}\tilde{y} \tag{4.3}$$

where $g^{-1}(x,y) \approx (x - r_1(x,y,-\Delta t)\Delta t, y - r_2(x,y,-\Delta t)\Delta t)$, where $(r_1(x,y), r_2(x,y)$ represents a step with the RK-method chosen, where the time step is done backwards, that is, the time step is negative.

Since the position of $g^{-1}(x,y)$ can be arbitrary, we must either choose to approximate $p(g^{-1}(x',\tilde{y}))$ by choosing the point $(\bar{x},\bar{y})$ in the grid, which is closest to $g^{-1}(x',\tilde{y})$, or represent $p(x,y)$ as a spline. We choose the latter, as it adds relatively little costs, but increases accuracy quite dramatically. We thus represent

$$p(x,y,t) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \Gamma_t(i,j) b_{i,j,n}(x,y) \tag{4.4}$$

where $\Gamma_t(i,j)$ is the control point associated with (x_i,y_j) in the grid, and $b_{i,j,n}(x,y)$ is the basis function of degree n, associated with (x_i,y_j). By this representation we thus get

$$p(x',y',t') = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \Gamma_t(i,j) \int_{\mathbb{R}} \frac{\exp\left(\frac{(y'-\tilde{y})^2}{2D_\xi \Delta t}\right)}{\sqrt{2\pi D_\xi \Delta t}} b_{i,j,n}(g^{-1}(x',\tilde{y}))|J_{g^{-1}}|\, \mathrm{d}\tilde{y} \tag{4.5}$$

As was observed in [11], when the SDE is autonomous (time independent) and Δt is constant, the integral in each term of the sum is also time independent, and thus, only needs to be calculated once. By denoting

$$B_{i,j}^{k,l} = \int_{\mathbb{R}} \frac{\exp\left(\frac{(y_l-\tilde{y})^2}{2D_\xi \Delta t}\right)}{\sqrt{2\pi D_\xi \Delta t}} b_{i,j,n}(g^{-1}(x_k,\tilde{y}))|J_{g^{-1}}|\, \mathrm{d}\tilde{y} \tag{4.6}$$

We thus get for each point (x_i,y_j) in the grid $\{x_1,\ldots,x_{n_x}\} \times \{y_1,\ldots,y_{n_y}\}$ that

$$p(x_k,y_l,t') = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} \Gamma_t(i,j) B_{i,j}^{k,l}, \tag{4.7}$$

all we need to do at each time step is to calculate the control points $\Gamma_t(i,j)$ and evaluate the above sum.

In this particular section, we will use linear basis functions, which has the basis properties that it is linear and that $b_{i,j}(x_k,y_l) = \delta_{i,k}\delta_{j,l}$. These basis functions are also called tent-functions, for obvious reasons. Linear basis functions are preferred here, because of ease of implementation when we are dealing with non-uniform grids, and the fact that we don't have to expensively calculate the control points $\Gamma_t(i,j)$ from linear systems of equations, we obtain the controls points directly by $\Gamma_t i, j = p(x_i,y_j,t)$, as only the basis function $b_{i,j,1}(x,y)$ is non-zero at $(x,y) = (x_i,y_j)$. The calculation is then reduced, for each time step to simply adding up the terms in the sum

$$p(x_k,y_l,t') = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y} p(x_i,y_j,t) B_{i,j}^{k,l} \tag{4.8}$$

It should be realized that the tensor $B_{i,j}^{k,l}$ has a strongly banded character, with elements decreasing rapidly towards zero off the main diagonals $i = k$ and $j = l$. In fact, since each basis function $b_{i,j,1}(x,y)$ is compactly supported in the convex hull of the nearest points of (x_i,y_j), most of the terms in the above sum are zero. This has important implications for the implementation, which will be discussed later.

4.2 Adaptive mesh refinement

Consider the stationary solution for the logarithm-transformed equation Lotka-Volterra equation, given by

$$P_{XY}(x,y) = C\exp[-(2\gamma/k\beta D_\xi\alpha^2)G(x,y)], \tag{4.9}$$

where

$$G(x,y) = k\beta\exp(y) - my + \beta\exp(x) - \left(\alpha - \frac{1}{2}\alpha^2 D_\xi - \gamma m/k\beta\right)x. \tag{4.10}$$

since $P_{XY}(x,y)$ decreases as an iterated exponential, that is, as $\exp(-\exp(x))$ when $x \to \infty$, and similarly for y, there is a very steep decrease of the function towards zero for $y > \ln(m/k\beta)$ and $x > \ln(\alpha - 1/2\alpha^2 D_\xi - \gamma m/k\beta)/\beta)$, the global maximum of the stationary solution.

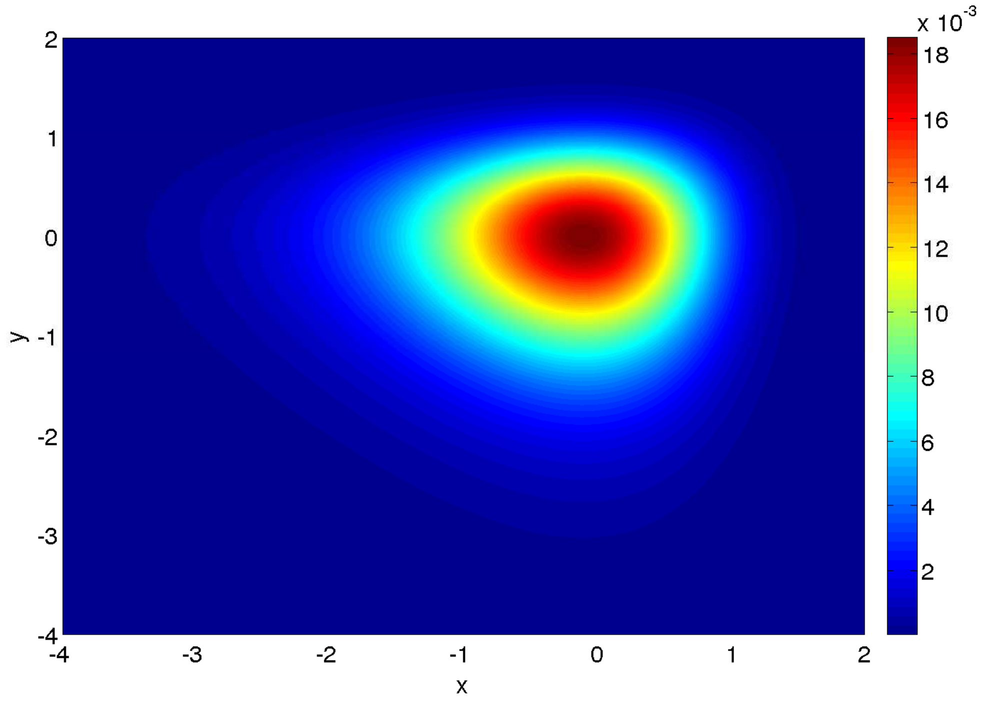


Figure 4.1: Plot of an unscaled version the stationary solution of the Lotka-Volterra equation on the grid $[-4,2] \times [-4,2]$, with parameters $k = \beta = m = \alpha = 1$ and $\gamma = \beta = 0.05$. The scaling constant C, was set to 1.

This is illustrated in figure 4.1, where you can see that the decrease in the function value is much more rapid for $x, y > 0$ than for $x, y < 0$, where the function is decaying exponentially. For this reason, a finer grid would be desired around $x \in [0,1]$ and $y \in [0,1]$ to get a more accurate spline representation of the function here. I fine enough grid to accurately represent the function around the areas of $x \in [0,1]$ and $y \in [0,1]$ would be wasted on other domains of the functions, in particular for large or small values of x and y, where the function is almost constantly equal to zero.

Therefore, we should adopt a non-equidistant grid. The question is how we can do this in a systematic manner without having to make a plot of the analytical solution, and consider it qualitatively each time we want to do it.

We will base the terminology and the method on a technique of adaptive mesh refinement often used when forming a finite element method solution of a static PDE. What we will do, is to start PI from an initial distribution, using a usual Cartesian equidistant grid, which is quite coarse, and run PI on this grid until we approximately have a stationary solution for the discrete PI-operator on this level of refinement. We then go to each square cell on the grid, and test some criterion that measures whether the true stationary solution is accurately represented on that cell, if it is not, then we divide this cell into smaller sub-cells.

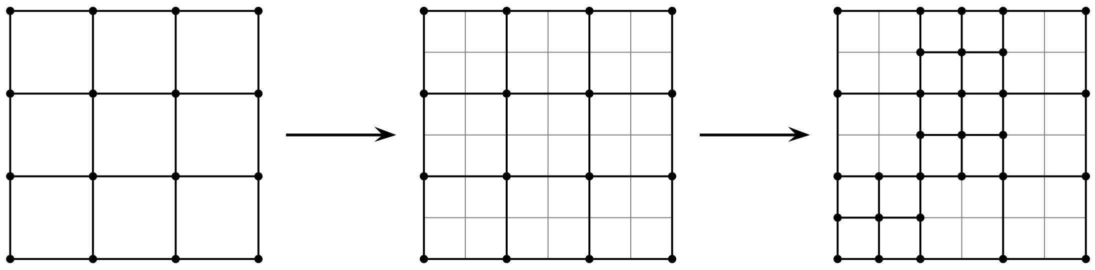


Figure 4.2: Illustration of the process of cell refinement.

Figure 4.2 illustrates how the process of cell-refinement is implemented. The 4 by 4 grid is what we start with, where each black node represents a grid-point for which we calculate a function value for the PI calculation. We then run PI on these grids on those grid-points, until we reach an approximate solution, and then go to a finer 7 by 7 grid as illustrated in the middle picture, where we store the value for the black nodes in every two grid node, such that there is one free grid point between each grid point, horizontally, vertically and diagonally. We then go over each cell, which are formed by the thick border, and test whether the cell should be refined. The final picture shows that the bottom-left, the middle and the top-middle cell has been tagged for refined, where one black node is added for each free grid point inside each cell tagged.

We then run PI anew on the new set of black nodes, until we obtain an approximate stationary solution on this level of refinement, and then do the same process over again. The initial values of the new black nodes which were previously not in the grid, we simply obtain from linear interpolation of the nearest black nodes which we have a PI-solution for.

4.2.1 Criteria for cell-refinement

An easy criterion, and one somewhat common from FEM, is to set some maximum value for the mass of each cell, that is, the volume under the surface of each cell. Over cell i with (x_j, y_k) as the bottom-left corner that covers $l+1$ grid-nodes, we have that the mass over that cell is approximately by the trapezoidal rule

$$\begin{aligned} M_i &= \int_{x_j}^{x_{j+l}} \int_{y_k}^{y_{k+l}} p(x,y,t)\,\mathrm{d}y\,\mathrm{d}x \\ &\approx \int_{x_j}^{x_{j+l}} \frac{lh}{2}\left[p(x,y_k)+p(x,y_{k+l})\right]\mathrm{d}x \\ &\approx \frac{(lh)^2}{4}\left[p(x_j,y_k)+p(x_j,y_{k+l})+p(x_{j+l},y_k)+p(x_{j+l},y_{k+1})\right] = \hat{M}_i \end{aligned} \tag{4.11}$$

We then apply the restriction that

$$\hat{M}_i < \tau, \text{ for all cells } i \tag{4.12}$$

where τ represents the maximum mass for each cell. If one of the cells mass exceeds τ, then we divide it into sub-cells as shown in figure 4.2. The idea is then to refine the grid one level at the time, and then run PI each time until we obtain a stationary solution that satisfies the above criterion. The hope is that to ensure, ideally, equal mass over each cell, we get a good representation of the stationary solution on the non-equidistant grid.

Another criterion that can be applied, is to try to to control the integration error when we calculate the mass of each cell. We will consider two ways of doing this. The first one is to calculate the mass of each cell, using methods of different order. A second order method, which is based on the trapezoidal rule, is already given as

$$M_i \approx \hat{M}_i = \frac{(lh)^2}{4}\left[p(x_j,y_k)+p(x_j,y_{k+l})+p(x_{j+l},y_k)+p(x_{j+l},y_{k+1})\right]. \tag{4.13}$$

A first order method of integration, would be to assume that the function is constantly equal to some value over the cell, say, $p(x_j, y_k)$, then

$$M_i \approx \tilde{M}_i = (lh)^2 p(x_j,y_k). \tag{4.14}$$

With two different schemes for approximating the mass of the cell of different order, we can approximate the integration error on that cell, by calculating

$$|\hat{M}_i - \tilde{M}_i| = \frac{(lh)^2}{4}|p(x_j,y_{k+l})+p(x_{j+l},y_k)+p(x_{j+l},y_{k+1})-3p(x_j,y_k)|. \tag{4.15}$$

We then put a similar constraint on this quantity, that is

$$|\hat{M}_i - \tilde{M}_i| < \tau \tag{4.16}$$

where τ is the maximal integration error. We proceed to refine the grid in the same manner with this criterion, as the maximum mass criterion.

Another approach to approximating the integration error, is to use a step-halving approach. This needs a new approach to how we define the cells, which is illustrated in figure 4.3. The left picture how we initially divide cells, which covers 9 black nodes instead of 4. We then follow the same procedure as before to move to a finer grid, store the value of each black node in every other grid node, and then divide cells into 4 sub-cells where appropriate. In the case of figure 4.3, only the bottom-left cell was refined.

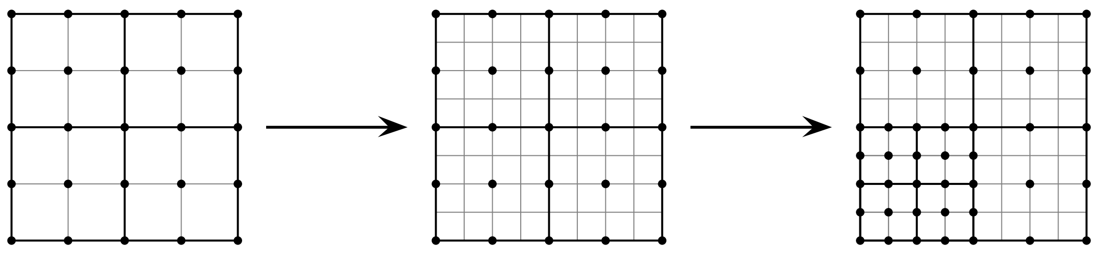


Figure 4.3: Illustration of the process of cell refinement, when one cell covers 9 nodes instead of 4

If, again, $2l$ denotes the size of a cell, then the approximate mass of that cell, using the trapezoidal rule is

$$\hat{M}_i = (lh)^2 \left[p(x_j, y_k) + p(x_j, y_{k+2l}) + p(x_{j+2l}, y_k) + p(x_{j+2l}, y_{k+21})\right]. \tag{4.17}$$

We can, however, also form the approximation of the mass of the cell by seeing that we can integrate over four cells of size l and, then add these together, this gives us

$$\begin{aligned}\tilde{M}_i = {} & \frac{(lh)^2}{4}\left[p(x_j, y_k) + p(x_j, y_{k+l}) + p(x_{j+l}, y_k) + p(x_{j+l}, y_{k+1})\right] \\ & + \frac{(lh)^2}{4}\left[p(x_{j+l}, y_k) + p(x_{j+l}, y_{k+l}) + p(x_{j+2l}, y_k) + p(x_{j+2l}, y_{k+1})\right] \\ & + \frac{(lh)^2}{4}\left[p(x_j, y_{k+l}) + p(x_j, y_{k+2l}) + p(x_{j+l}, y_{k+l}) + p(x_{j+l}, y_{k+21})\right] \\ & + \frac{(lh)^2}{4}\left[p(x_{j+l}, y_{k+l}) + p(x_{j+l}, y_{k+2l}) + p(x_{j+2l}, y_{k+l}) + p(x_{j+2l}, y_{k+21})\right] \\ = {} & \frac{(lh)^2}{4}[p(x_j, y_k) + 2p(x_j, y_{k+l}) + p(x_j, y_{k+2l}) + 2p(x_{j+l}, y_k) + 4p(x_{j+l}, y_{k+1}) \\ & + 2p(x_{j+l}, y_{k+21}) + p(x_{j+2l}, y_k) + 2p(x_{j+2l}, y_{k+l}) + p(x_{j+2l}, y_{k+2l})]\end{aligned} \tag{4.18}$$

approximating the integration error, as before, by $|\hat{M}_i - \tilde{M}_i|$, we get

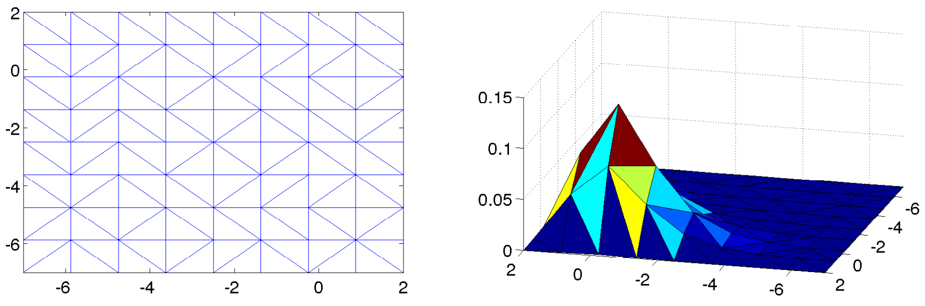
$$\begin{aligned}|\hat{M}_i - \tilde{M}_i| = {} & \frac{(lh)^2}{4}| - 3p(x_j, y_k) + 2p(x_j, y_{k+l}) + -3p(x_j, y_{k+2l}) + 2p(x_{j+l}, y_k) \\ & + 4p(x_{j+l}, y_{k+1}) + 2p(x_{j+l}, y_{k+21}) - 3p(x_{j+2l}, y_k) + 2p(x_{j+2l}, y_{k+l}) - 3p(x_{j+2l}, y_{k+2l})|\end{aligned} \tag{4.19}$$

Using the criterion again in (4.16), and the same procedure for refining the cells, we then obtain a non-uniform grid for the stationary solution based on this step-halving technique.

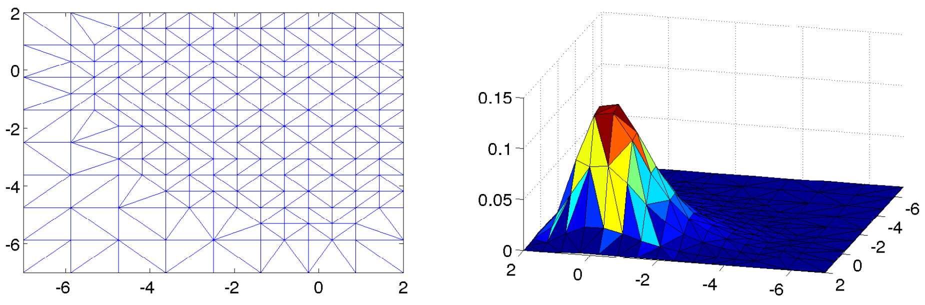
Example 4.1. We will take a look at the distribution

$$p_{XY}(x, y) = \exp[-(\exp(x) - x + \exp(y) - y)] \tag{4.20}$$

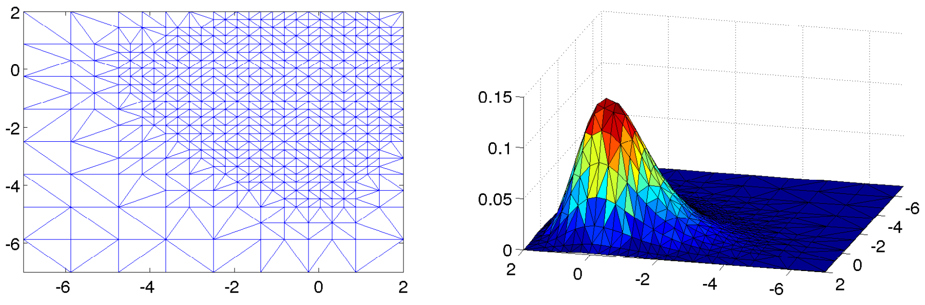
which is based on the stationary solution for the logarithm-transformed Lotka-Volterra equation, to see how the criteria for cell-refinement performs. Integrating



(a) Number of grid nodes: 81, number of black nodes: 81



(b) Number of grid nodes: 289, number of black nodes: 212



(c) Number of grid nodes: 1089, number of black nodes: 560

Figure 4.4: Plot of the black nodes, along with a plot of the stationary solution of the Lotka-Volterra equation using the black nodes, starting with 9×9 nodes. The refinement-criterion used here, was the maximal mass criterion given in (4.14), using $\tau = 10^{-3}$.

this distribution over some subset of $\mathbb{R}^2$ is quite easy, as when we introduce the transformation $(u, v) = (\exp(x), \exp(y))$, we get

$$\begin{aligned}\int_{y_1}^{y_2}\int_{x_1}^{x_2} e^{[-(e^x - x + e^y - y)]}\,\mathrm{d}x\,\mathrm{d}y &= \int_{v_1}^{v_2}\int_{u_1}^{u_2} e^{[-(u-x+v-y)]}\,\frac{\mathrm{d}u}{e^x}\,\frac{\mathrm{d}v}{e^y}\\ &= \int_{v_1}^{v_2}\int_{u_1}^{u_2} e^{-(u+v)}\,\mathrm{d}u\,\mathrm{d}v\\ &= (e^{-u_1} - e^{-u_2})(e^{-v_1} - e^{-v_2}) \qquad (4.21)\end{aligned}$$

where $u_1 = \exp(x_1)$, $u_2 = \exp(x_2)$, $v_1 = \exp(y_1)$ and $v_2 = \exp(y_2)$. In particular, we see that when $(x_1, x_2) \to (-\infty, \infty)$ and $(y_1, y_2) \to (-\infty, \infty)$, then the integral is equal to 1, showing that it is indeed a distribution.

We will look at this distribution over grids constrained by $[-7, 2] \times [-7, 2]$, which should be a reasonable representation of the whole solution, as the mass over this area is approximately 0.99908, and will test each of the criteria to see which one performs best, in terms of which method gives a distribution of black nodes, which gives the best approximation of the mass of the distribution over $[-7, 2] \times [-7, 2]$, when we integrate numerically, using linear elements. This should also give us a reasonably good measure of how good linear interpolation of the black nodes approximates the actual function.

In figure 4.4, one can see the process of cell refinement for the maximal mass criterion, using $\tau = 10^{-3}$, and how the graph of the stationary solution becomes more detailed as we refine the grid. Notice how the graph of the function in figure 4.4 (c) is represent with black nodes that are only half in number of the grid nodes, while still getting more or less the same representation, by concentrating the black nodes around the area of the graph where it is significantly greater than zero.

Number of grid nodes	Number of black nodes	% of black nodes	Error
289	289	100%	0.0018
1089	1089	100%	$2.98 \cdot 10^{-4}$
4225	4222	99.9%	$8.77 \cdot 10^{-5}$
16641	16404	98.5%	$2.49 \cdot 10^{-5}$
66049	62449	94.5%	$6.18 \cdot 10^{-6}$
263169	229449	87.1%	$1.33 \cdot 10^{-6}$
1050625	788386	75%	$3.79 \cdot 10^{-8}$
4198401	2396020	57%	$5.84 \cdot 10^{-7}$
16785409	6506308	38.7%	$9.16 \cdot 10^{-7}$

Table 4.1: Table for the integration error over the function in (4.20) when using linear interpolation over the black nodes, while using the maximal mass criterion, where the numerically calculated mass is given in (4.11) for cell refinement, with $\tau = 10^{-7}$. The calculations were performed on the grid $[-7, 2] \times [-7, 2]$.

Tables 4.1, 4.2 and 4.3 shows how the three different cell refinement techniques, given in equations (4.11), (4.15) and (4.19) performs, in terms of calculating the mass under the curve, using linear interpolation, when compared to the exact mass under the curve, given in equation (4.21) for $u_1 = v_1 = \exp(-7)$ and $u_2 = v_2 = \exp(2)$.

Note that in all cases, the error actually increases as the number of black nodes becomes big in all three cases. In table 4.1 and 4.2, this happens after the number of grid nodes exceeds 1050625, while in table 4.3 this happens when the number of grid nodes exceeds 4198401. One might think that this is due to the increasing number of black nodes, and that the round-off error is starting to dominate the error, due to the large sum that needs to be calculated. This is not very likely the case however, since the Kahan summation algorithm, introduced in chapter 1, is used on the very good-conditioned sum, of only positive numbers.

The cause of the increasing error, is much more likely due to the following: The distribution given in equation (4.20) is both convex and concave. In particular, this

Number of grid nodes	Number of black nodes	% of black nodes	Error
289	289	100%	0.0018
1089	1089	100%	$2.98 \cdot 10^{-4}$
4225	4225	100%	$1.16 \cdot 10^{-4}$
16641	16596	99.7%	$2.59 \cdot 10^{-5}$
66049	63330	95.8%	$6.19 \cdot 10^{-6}$
263169	220711	83.8%	$1.73 \cdot 10^{-6}$
1050625	620869	59%	$3.35 \cdot 10^{-7}$
4198401	1363903	23.8%	$3.78 \cdot 10^{-7}$
16785409	1487944	8.8%	$8.86 \cdot 10^{-7}$

Table 4.2: Table for the integration error over the function in (4.20) when using linear interpolation over the black nodes, while using the maximal integration error criterion for cell refinement, where the approximate error is given in (4.15), with $\tau = 5 \cdot 10^{-9}$. The calculations were performed on the grid $[-7, 2] \times [-7, 2]$

Number of grid nodes	Number of black nodes	% of black nodes	Error
289	289	100%	0.0018
1089	1089	100%	$2.98 \cdot 10^{-4}$
4225	4225	100%	$1.16 \cdot 10^{-4}$
16641	16641	100%	$2.70 \cdot 10^{-5}$
66049	66037	99.9%	$6.31 \cdot 10^{-6}$
263169	250781	95.2%	$1.55 \cdot 10^{-6}$
1050625	773785	73.6%	$3.76 \cdot 10^{-7}$
4198401	1680377	40%	$2.2 \cdot 10^{-7}$
16785409	2005433	11.9%	$7.47 \cdot 10^{-7}$

Table 4.3: Table for the integration error over the function in (4.20) when using linear interpolation over the black nodes, while using the maximal integration error criterion for cell refinement, where the approximate error is given in (4.19), with $\tau = 2 \cdot 10^{-10}$. The calculations were performed on the grid $[-7, 2] \times [-7, 2]$

means that when we use linear interpolation to integrate elements, we will have estimates of the integral over the relevant element that overshoots the true value of the integral, and estimates of the integral over the relevant elements which undershoots the integral. When adding up the integrals, over the individual elements, these overestimates and underestimates cancel out in a rather arbitrary fashion, sometimes bringing the final integral closer to the true value of the mass, than each individual part of integration would suggest.

But when the grid becomes finer, these overestimates and underestimates become less relevant as we get a tighter fit around the distribution, and the actual error from the sum of the integrals becomes something more in line with the accuracy of integrating each element. The measure of the accuracy of representing the stationary solution by linear interpolation by measuring the accuracy of mass-calculation by linear interpolation is therefore slightly deceptive, and the results for lower number of grid nodes, generally seem better than what they are.

Nonetheless the author has used the mesh refinement criterion based on the the step-halving measure, as it taking into account the curvature of the function over a cell. Since the function is best approximated by linear basis functions where the curvature of the function is small.

4.3 Computing the PI-coefficients

In order to compute the PI-coefficients

$$B_{i,j}^{k,l} = \int_{\mathbb{R}} \frac{\exp\left(\frac{(y_l-\tilde{y})^2}{2D_\xi \Delta t}\right)}{\sqrt{2\pi D_\xi \Delta t}} b_{i,j,n}(g^{-1}(x_k,\tilde{y}))|J_{g^{-1}}|\,\mathrm{d}\tilde{y}, \tag{4.22}$$

we need to find a good way to represent the basis functions on the non-equidistant grid. The basis functions are, as previously mentioned, linear "tent"-functions, with the basic property that $b_{i,j,1}(x_k,y_l) = \delta_{i,k}\delta_{j,l}$. A good way to represent these functions, is to split them up in parts by making a triangulation of the grid. For each triangle element Ω_k, we assign three linear functions defined on the triangle which is equal to 1 in one corner of the triangle, and zero in the two others.

The question is then, how we can define these functions mathematically on an arbitrary triangle. The answer, which we borrow from the finite element method, is that we do not: Instead we only operator on a master triangle with corners in $(0,0)$, $(1,0)$ and $(0,1)$. And more precisely, we do this by introducing the following transformation for the arbitrary triangle with corners in (x_1,y_1), (x_2,y_2) and (x_3,y_3)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \psi_1(\xi,\eta)x_1 + \psi_2(\xi,\eta)x_2 + \psi_3(\xi,\eta)x_3 \\ \psi_1(\xi,\eta)y_1 + \psi_2(\xi,\eta)y_2 + \psi_3(\xi,\eta)y_3 \end{bmatrix} \tag{4.23}$$

where

$$\psi_1(\xi,\eta) = 1-\xi-\eta \tag{4.24}$$
$$\psi_2(\xi,\eta) = \xi \tag{4.25}$$
$$\psi_3(\xi,\eta) = \eta \tag{4.26}$$

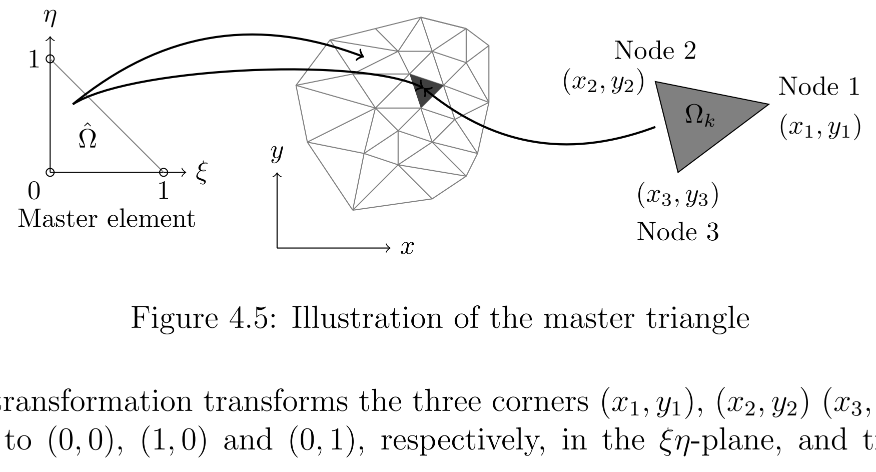


Figure 4.5: Illustration of the master triangle

This transformation transforms the three corners (x_1,y_1), (x_2,y_2) (x_3,y_3) in the xy-plane to $(0,0)$, $(1,0)$ and $(0,1)$, respectively, in the $\xi\eta$-plane, and transforms each point within the arbitrary triangle to the corresponding position in the master triangle. Figure 4.5 illustrates the master triangle and the transformation visually.

Suppose now that we know some point (x, y) within a triangle, say, the triangle Ω_k, then in order to find the corresponding (ξ, η)-coordinates, we need to write up the transformation in (4.23) in terms of the variables ξ and η. It can be written in matrix form in the following way

$$\begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.27}$$

If the points (x_1, y_1), (x_2, y_2) and (x_3, y_3) are not all on a straight line, then the left matrix is non-singular, and has an inverse given by

$$\frac{1}{(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)} \begin{bmatrix} y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{bmatrix} \tag{4.28}$$

Isolating ξ and η on the left-hand side and multiplying with the inverse matrix from the left, then gives

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \frac{1}{(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)} \begin{bmatrix} y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix} \tag{4.29}$$

Once we have calculated ξ and η we can calculate the value for the three parts of the basis functions that belong to that triangle. These are simply $\psi_1(\xi, \eta) = 1 - \xi - \eta$ which corresponds to the part of the basis function that belongs to (x_1, y_1), $\psi_2(\xi, \eta) = \xi$ which corresponds to a part of the basis function belonging to (x_2, y_2) and $\psi_3(\xi, \eta) = \eta$ which corresponds to a part of the basis function belonging to (x_3, y_3).

In order to calculate the PI-coefficient $B_{i,j}^{k,l}$, we integrate the integral in equation (4.22) by restricting the integration interval to $[y_l - 5\sqrt{D_\xi \Delta t}, y_l + 5\sqrt{D_\xi \Delta t}]$, and then subdividing this integration interval into an odd number n of equidistant quadrature points $y_l - 5\sqrt{D_\xi \Delta t} = \tilde{y}_1 < \tilde{y}_2 < \cdots < \tilde{y}_n = y_l + 5\sqrt{D_\xi \Delta t}$. Once we have calculated the integrand at all of the quadrature points we add these up, using the composite Simpson's rule. We can break the process of calculating the $B_{i,j}^{k,l}$ for each fixed (k, l) into performing the following tasks for each $i \in \{1, 2, \ldots, n\}$

- Calculate $(x, y) = g^{-1}(x_k, \tilde{y}_i)$, where g^{-1} as usual is the inverse of the transformation $g(x, y) = (\tilde{x}, \tilde{y}) = (x + r_1(x, y, t)\Delta t, y + r_2(x, y, t)\Delta t)$

- Find which triangle element Ω_k belongs to. The mathematics for doing this, is not presented here, as the author has simply used a built-in function in MATLAB called *pointLocation*, which takes as input a Delaunay-triangulation of the grid, as well as the point (x, y) and returns the index k of the triangle in the triangulation in which (x, y) lies.

- Calculate the corresponding value for (ξ, η) by using equation (4.29)

- Suppose for example that $b_{i,j,1}(x, y)$, $b_{i+2,j,1}(x, y)$ and $b_{i,j+3,1}(x, y)$ are the basis functions that belongs to the corners (x_1, y_1), (x_2, y_2) and (x_3, y_3) respectively of the triangle element Ω_k, then the final step consists of adding to the PI-coefficients $B_{i,j}^{k,l}$, $B_{i+2,j}^{k,l}$ and $B_{i,j+3}^{k,l}$ the integrand of equation (4.22) according to the composite Simpson's rule, using the basis functions $\psi_1(\xi, \eta) = 1 - \xi - \eta$, $\psi_2(\xi, \eta) = \xi$ and $\psi_3(\xi, \eta) = \eta$, respectively.

4.4 Integration over a triangle element

Suppose that the function f is represented by linear basis functions on a grid, then it can be represented on the triangle Ω_k by, say

$$f(x,y) = f(x_1,y_1)b_{i,j,1}(x,y) + f(x_2,y_2)b_{i+2,j,1}(x,y) + f(x_3,y_3)b_{i,j+3,1} \tag{4.30}$$

where $b_{i,j,1})(x,y)$, $b_{i+2,j,1}(x,y)$ and $b_{i,j+3,1}(x,y)$ are the linear basis functions corresponding to the points (x_1,y_1), (x_2,y_2) and (x_3,y_3). These are not actual grid points, but are given this way for simplicity.

The integral of the function over this element is then given by

$$\begin{aligned}&\int_{\Omega_k} f(x)\,\mathrm{d}x \\ &= \int_{\Omega_k} f(x_1,y_1)b_{i,j,1}(x,y) + f(x_2,y_2)b_{i+2,j,1}(x,y) + f(x_3,y_3)b_{i,j+3,1}(x,y)\,\mathrm{d}x,\mathrm{d}y.\end{aligned} \tag{4.31}$$

By using the transformation in (4.23), the integration area is then transformed to that of the master triangle, and the basis functions are transformed to the basis functions for the master triangle, that is, we get

$$\begin{aligned}&\int_{\Omega_k} f(x)\,\mathrm{d}x \\ &= \int_0^1\int_0^{1-\eta} (f(x_1,y_1)\psi_1(\xi,\eta) + f(x_2,y_2)\psi_2(\xi,\eta) + f(x_3,y_3)\psi_3(\xi,\eta))|\det(J)|\,\mathrm{d}\xi\,\mathrm{d}\eta,\end{aligned} \tag{4.32}$$

where $\det(J)$ is the determinant of the Jacobian of the transformation, which is precisely the matrix

$$J = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}. \tag{4.33}$$

An observant reader will see that the absolute value of the determinant of the above Jacobian, is simply equal to the area of the parallelogram formed by the vectors $[x_2 - x_1, y_2 - y_1]$ and $[x_3 - x_1, y_3 - y_1]$. Therefore $A_k = |\det(J)|/2$ is equal to the area of the triangle. By calculating the above integral, we then get

$$\int_{\Omega_k} f(x)\,\mathrm{d}x = \frac{A_k}{3}\left[f(x_1,y_1) + f(x_2,y_2) + f(x_3,y_3)\right] \tag{4.34}$$

where $A_k = |(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)|$. We clearly see from this formula, that the integral of the function f represented by linear basis functions over the triangle element Ω_k is simply given by the area of the triangle multiplied the average value of the three function values in the corners of the triangle.

4.5 Results

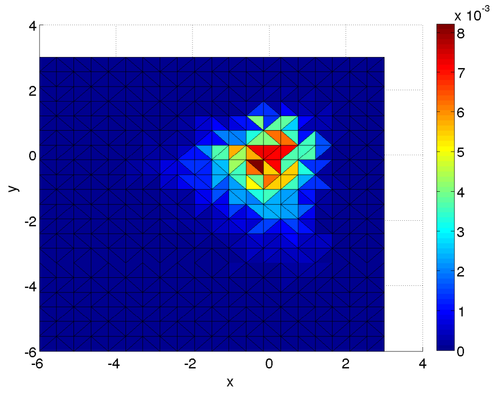
In order to gauge the results we will use the unharvested transformed Lotka-Volterra equation

$$\mathrm{d}X_t = (-m + k\beta \exp(Y_t))\,\mathrm{d}t \tag{4.35a}$$

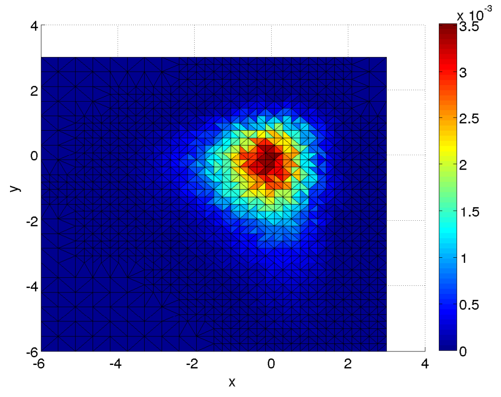
$$\mathrm{d}Y_t = (\alpha - \frac{1}{2}\alpha^2 D_\xi - \beta \exp(X_t) - \gamma \exp(Y_t))\,\mathrm{d}t + \alpha\sqrt{D_\xi}\,\mathrm{d}B_t \tag{4.35b}$$

A plot of the solution, where we have gradually refined the mesh and made the time step smaller in order to try to obtain a better estimate of the stationary solution of the SDE. There is unfortunately a bug in the program, which causes the scale of the solution to be wrong. But as can be seen, the refinement of the cells works as intended, giving a fine mesh around the critical areas of the function, and a much coarser grid where the function is close to zero.

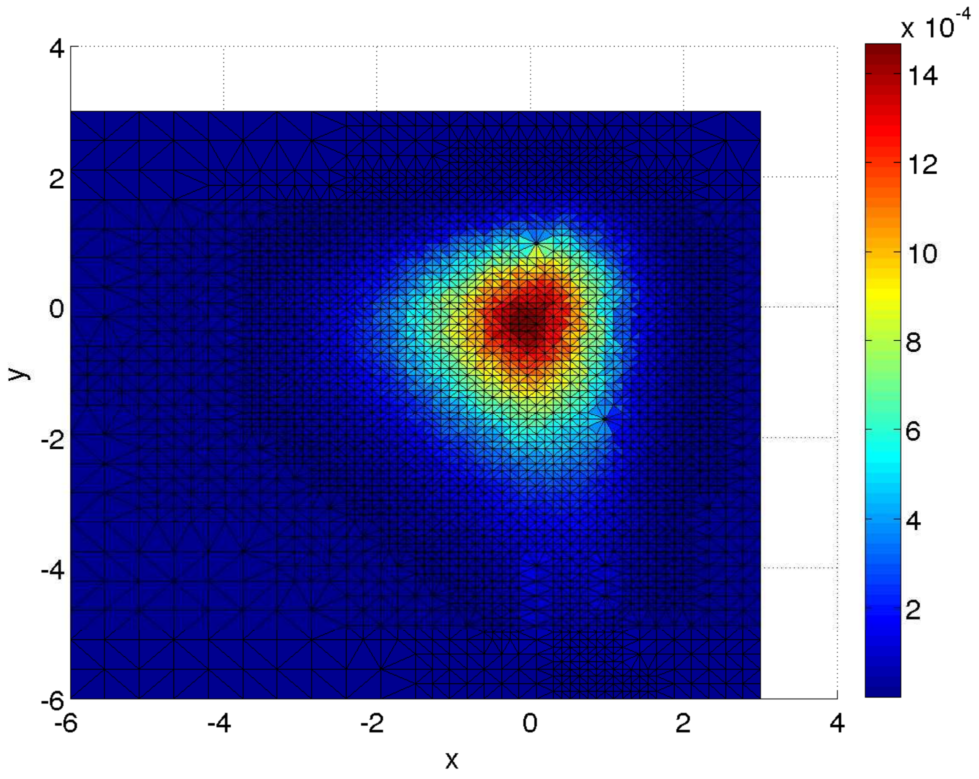
Unfortunately, the author did not have the time to explore more of the aspects of the solution using this algorithm.



(a) Initial level of refinement, black nodes: 441



(b) First level of refinement, Black nodes: 1033



(c) Black nodes:2045

Figure 4.6: An approximation of the stationary solution of equation (4.35) using parameters $\alpha = m = k = \beta = 1$, $\gamma = D_\xi = 0.05$. The initial number of black nodes was set to 441, and initial time step was $\Delta t = 0.1$, which is reduced by a factor of 10 for each refinement. The algorithm went to the next level of refinement when the maximal difference in absolute values between the solution at two subsequent time steps was less than 10^{-4}

Chapter 5

Conclusion

Through the first part of this thesis the mathematical ground work for the Path Integration method, as well as a discussion of the practical considerations of its implementation on a computer has been discussed. We have discussed convergence, and referred to sufficient conditions for convergence of the Path Integration method. The second part, which uses results from the first part, concerns itself mainly with the solution of a stochastic model of the Lotka-Volterra equation, where we have discussed it qualitatively for some chosen parameters, with and without harvesting on the population of the predator.

We ultimately presented an adaptive algorithm which was designed to be a fast and accurate method to seek out the stationary solution for an autonomous 2D-SDE with constant noise entering through the last dimension. It was designed to be especially well suited to find the stationary solution of the logarithm-transformed Lotka-Volterra equation, since it requires a precise space-representation around the critical points of the solution.

Appendix A

Source code

```
/*This is the source code used for calculations on PI with fixed time steps.
The code is highly based on code written by Eirik Mo in his Ph.D-thesis, and is converted
from FORTRAN to C++ line for line in most lines.

The main function in this program is at the bottom of the source code, where the program is further
explained, above it, are function declerations.

This code has been altered several times during the project, as I never wrote a complete interface to
cover every possible situation. Thus there is not included code for all the calculations done in this
project. The current version of the code, does only support constant values for the noise process for
example.

In the case of non-linear system of equations needing to be solved, a method from the GNU scientific
library has been implemented. It used method of steepest descent to approximate the solution first
and when the solution is close enough, uses Newton's method to accelerate the solution*/

#include <iostream>
#include <vector>
#include <fstream>
#include <cmath>
#include <stdlib.h>
#include <stdio.h>
#include <cstdio>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_multiroots.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
//#include <gsl/gsl_odeiv2.h>


using namespace std;

typedef vector<double> Vector;
typedef vector<vector<double> > Matrise;
double maxnorm(Matrise X, Matrise Y, int nx, int ny) {
  //Returns the maximal elementwise difference between the matrix Y and X in absolute value
  double max=0;
  double test;
  for (int i = 0; i < nx; i += 1)
  {
    for (int j = 0; j < ny; j += 1)
    {
      test=fabs(X[i][j]-Y[i][j]);
      if (test>max)
      {
        max=test;
      }
    }
  }
  return max;
}
Vector vector_add(Vector X, Vector Y, double c) {
//Computes X+c*Y, and returns the result
X[0]=X[0]+c*Y[0];
X[1]=X[1]+c*Y[1];
return X;
}
Vector vector_prod(Vector X, Vector Y) {
//produces the elementwise product of the two vectors X and Y and returns the result as a vector
X[0]=X[0]*Y[0];
X[1]=X[1]*Y[1];
return X;
}
Vector linspace(double min, double max, int nx) {
  //Creates a vector of length nx with an uniform grid starting at min and ending at max
  Vector x(nx);
  double h=(max-min)/(nx-1);
  for (int i = 0; i < nx; i += 1) {
    x[i]=min+i*h;
  }
  return x;
}
double milstein(double y, double x, double dt, double by){
double PI=3.1415926535;
double k1=1/by;
double k2=0.5*by*by*x;
double c=x*(1-0.5*k1);
double norm_factor=1/(2*sqrt(2*PI*dt*k2*(y-c)));
return norm_factor*(exp(-pow(sqrt(y-c)/sqrt(k2)+k1,2)/(2*dt))+exp(-pow(sqrt(y-c)/sqrt(k2)-k1,2)/(2*dt)));
}
Vector partition(double y, double dt, double by,double dy_init, double ymin,
double ymax, double TOL, double n_min) {
Vector p;
double x;
if (ymin>0)
{
  x=ymin;
}
else
  x=0;
double PI=3.1415926535;
bool try_cont;
double err;
double fx_old=0;
double fx_new;
double fx_mid;
double dy=dy_init;
while (x<ymax)
{
  fx_new=fx_old;
  try_cont=true;
  while (try_cont)
  {
    fx_new=exp(-pow(y-(x+dy),2)/(2*dt*pow(by*(x+dy),2)))/sqrt(2*PI*dt*pow(by*(x+dy),2));
    fx_mid=exp(-pow(y-(x+0.5*dy),2)/(2*dt*pow(by*(x+0.5*dy),2)))/sqrt(2*PI*dt*pow(by*(x+0.5*dy),2));
    err=dy*fabs(fx_new-2*fx_mid+fx_old)/2;
    if (err>TOL)
    {
      dy=0.9*pow(TOL/err,1.0/3)*dy;
    }
    else
    {
      x=x+dy;
      if (x<ymax)
      {
        p.push_back(x);
      }
      //cout << x << " " << fx_new;
      //cin.get();
      if (err>TOL*pow(10,-4))
      {
        dy=0.9*pow(TOL/err,1.0/3)*dy;
      }
      else
        dy=2*dy;
      try_cont=false;
    }
  }
}
p.push_back(ymax);
if (p.size()<n_min)
{
  p=partition(y,dt,by,dy_init/10,ymin,ymax,TOL/3,n_min);
}
if (p.size()>4000)
{
  p=partition(y,dt,by,dy_init,ymin,ymax,TOL*2,n_min);
}
return p;
}
double heval(double x) {
  double T=1;
  double h=10;
  if (exp(x)<T)
  {
    return 0;
  }
  return h*(1-T*exp(-x))/(h+(exp(x)-T));
}
Vector aeval(Vector x, double t) {
  /*This is the deterministic term in the SDE dX_t = a(X_t,t) dt + b(X_t,t) dW_t, in this case
  we have used the SDE from example 3 in the report*/
  Vector a(2);
  a[0]=(-1.0+x[1])*x[0];
  a[1]=(1.0-x[0]-0.05*x[1])*x[1];
  return a;
}
Vector adiffeval(Vector x,double t) {
  Vector a(2);
  a[0]=-1.0+x[1];
  a[1]=1.0-x[0]-0.1*x[1];
  return a;
}
Vector RK4step(Vector x, double t, double h) {
  /*This represents the result of one step of RK4, using the function a(X_t,t) as
  the derivative*/
  Vector k1=aeval(x,t);
  Vector k2=aeval(vector_add(x,k1,-h/2),t-h/2);
  Vector k3=aeval(vector_add(x,k2,-h/2),t-h/2);
  Vector k4=aeval(vector_add(x,k3,-h),t-h);
  Vector k(2);
  k[0]=x[0]-(k1[0]+2*k2[0]+2*k3[0]+k4[0])*(h/6);
  k[1]=x[1]-(k1[1]+2*k2[1]+2*k3[1]+k4[1])*(h/6);
  return k;
}
Vector RK4stepd(Vector x, double t, double h) {
  Vector one(2);
  one[0]=1;
  one[1]=1;
  Vector k1=aeval(x,t);
  Vector k1d=adiffeval(x,t);
  Vector k2=aeval(vector_add(x,k1,-h/2),t-h/2);
  Vector k2d=vector_prod(vector_add(one,k1d,-h/2),adiffeval(vector_add(x,k1,-h/2),t-h/2));
  Vector k3=aeval(vector_add(x,k2,-h/2),t-h/2);
  Vector k3d=vector_prod(vector_add(one,k2d,-h/2),adiffeval(vector_add(x,k2,-h/2),t-h/2));
  Vector k4d=vector_prod(vector_add(one,k3d,-h),adiffeval(vector_add(x,k3,-h),t-h));
  Vector k(2);
  k[0]=1-(k1d[0]+2*k2d[0]+2*k3d[0]+k4d[0])*(h/6);
  k[1]=1-(k1d[1]+2*k2d[1]+2*k3d[1]+k4d[1])*(h/6);
  return k;
}
void Matrise2matrix(gsl_matrix* matrix, Matrise matrise,int nx, int ny) {
  for (int i = 1; i < nx-1; i += 1)
  {
    for (int j = 1; j < ny-1; j += 1)
    {
      gsl_matrix_set(matrix,i-1,j-1,matrise[i][j]);
    }
  }
}
Matrise matrix2Matrise(gsl_matrix *matrix,int nx,int ny) {
  Matrise matrise(nx, vector<double>(ny, 0) );
  for (int i = 0; i < nx-2; i += 1)
  {
    for (int j = 0; j < ny-2; j += 1)
    {
      matrise[i+1][j+1]=gsl_matrix_get(matrix,i,j);
    }
  }
  return matrise;
}
Matrise Pathint(int nx, int ny, double t, double dt, int nt, double by, double xmin,
double xmax, double ymin, double ymax,int method, char* filename){
  /*This is the main routine in this program, calculating the PI solution after
  a chosen number of timesteps from an inital distribtion.

  Input:
  nx = Number of grid points in x-direction
  ny = Number of grid points in y-direction
  t = starting time
  dt = time step
  nt = number of time steps
  by = The constant for the diffusion process: dX_t = a(X_t,t) dt + by dW_t
  xmin, xmax, ymin, ymax = defines the calculation domain [xmin,xmax]X[ymin,ymax]
  method = number for the method used, explained in the main function
  filename = name of the file to write the result to

  Output:
  Matrix p_new with the numerical solution.
  */
  const double PI=3.1415926535;
  Matrise p_new(nx+2, vector<double>(ny+2, 0) );
  Matrise avg=p_new;
  Matrise ex(nx+2, vector<double>(ny+2, 0) );
  vector < vector <double > > partitions; /*the number of integration points in order to
  calculate equation (2) in the report, in this case it is always chosen
  to be twice the number of grid points in y-direction*/

  /*Calculates the Diagonal of the U matrix for the LU factorization of the
  uniform cubic B-spline. */
  Vector qx=mo_splinit(nx);
  Vector qy=mo_splinit(ny);

  /*Precalculate some constants that will be repeatedly used later*/
  double norm_factor;
  double norm_exponent;

  // Counters
  int i, j, k;

  //Initializes grid
  double mass=0;
  Vector x=linspace(xmin,xmax,nx);
  double dx=x[1]-x[0];
  Vector y = linspace(ymin,ymax,ny);
  double dy = y[1]-y[0];
  //Construct the inital distribution
  for (i = 1; i < nx+1; i += 1) {
    for (j = 1; j < ny+1; j += 1) {
      //p_new[i][j]=exp(-(2.0/0.5)*(pow(x[i-1]-2,2)+pow(y[j-1]-2,2)));
      //p_new[i][j]=pow(x[i-1],0.85)*pow(y[j-1],1)*exp(-2*(y[j-1]+x[i-1]));
      p_new[i][j]=pow(x[i-1],8.25)*pow(y[j-1],9)*exp(-10*(x[i-1]+y[j-1]));
      mass=mass+p_new[i][j];
    }
  }
  mass=mass*dx*dy;
  for ( j = 1; j < ny+1; j += 1)
  {
    for ( k = 1; k < nx+1; k += 1)
    {
      p_new[k][j]=p_new[k][j]/mass;
      ex[k][j]=p_new[k][j];
    }
  }
  double masstmp=mass;
  mass=0;
  //Initialize variables that will be used in the following calculations
  vector < vector < vector <double> > > x_final;
  vector < vector < vector <double> > > y_final;
  vector < vector < vector < vector < double > > > > jac;
  //allocates memory for the 3D-array x_final and y_final
  x_final.resize(ny);
  y_final.resize(ny);
  jac.resize(ny);
  for ( j = 0; j < ny; j += 1)
  {
    x_final[j].resize(nx);
    y_final[j].resize(nx);
    jac[j].resize(nx);
  }
  vector <double> pp_tab;
  vector <double> success;
  Vector tmp(2);
  double xv, yv;
  double py1, py2, py3, py4, px1, px2, px3, px4;
  int kk2, ll2;
  double pp;
  Vector Ex(nt+1,0);
  Vector Ey(nt+1,0);
  for (i = 1; i < nx+1; i += 1) {
    for (j = 1; j < ny+1; j += 1) {
      Ex[0]=Ex[0]+x[i-1]*p_new[i][j]*dx*dy;
      Ey[0]=Ey[0]+y[j-1]*p_new[i][j]*dx*dy;
    }
  }
  Matrise po;
  gsl_matrix *po_c=gsl_matrix_alloc(nx,nx);
  gsl_vector* e=gsl_vector_alloc(nx);
  gsl_vector* f=gsl_vector_alloc(nx-1);
  gsl_vector_set_all(e,4.0/6);
  gsl_vector_set_all(f,1.0/6);
  double ptmp;
  double error;
  double error_max=0;
  double yy;
  //Main loop
  fstream output( filename,fstream::out); //open file to write the result to
  fstream
  graf("graf", fstream::out);
  //Generates grid
  for (j = 0; j < ny-1; j += 1)
  {
    partitions.push_back(partition(y[j+1],dt,by,0.001*sqrt(dt),y[j]-(by*by*dt)*5,ymax,0.01*sqrt(dt),ny));
  }
  cout << "done";
  cin.get();
  for (i = 0; i < nt; i += 1) {
    //Stores the old density
    po=p_new;
    Matrise2matrix(po_c,po,nx+2,ny+2);
    //Calculate B-spline coefficients.
        for (int ll=0; ll< ny; ll++) {
          gsl_vector_view b = gsl_matrix_column(po_c,ll);
          gsl_linalg_solve_symm_tridiag(e,f,&b.vector,&b.vector);
         }
         for (int kk=0; kk< nx; kk++) {
            gsl_vector_view b=gsl_matrix_row(po_c,kk);
          gsl_linalg_solve_symm_tridiag(e,f,&b.vector,&b.vector);
         }
        //FILE* koeff=fopen("koeff","w");
        //gsl_matrix_fprintf (koeff, po_c, "%g ");
        //fclose(koeff);
        po=matrix2Matrise(po_c,nx+2,ny+2);
    for (j = 1; j < ny; j += 1) {
      pp_tab.resize(partitions[j-1].size());
      success.resize(partitions[j-1].size());
      for (k = 0; k < nx; k += 1) {
        if (i==0)
        {
        for (int i_partition = 0; i_partition < partitions[j-1].size(); i_partition += 1) {
          //cout <<i_partition << " " << tmp[0] << " " << tmp[1] << endl;
          tmp[0]=x[k];
          tmp[1]=partitions[j-1][i_partition];
          if (method==1)
          {
            //If Euler's method was chosen as method
            tmp=Estep(tmp,t,dt);
  }
  else if (method==2)
  {
    //If Improved Euler's method was chosen as method
    tmp=rstep(tmp,t,dt);
  }
  else if (method==3)
  {
    //If RK4 was chosen as method
    //RK4step(tmp,t,dt);
    jac[k][j].push_back(RK4stepd(tmp,t,dt));
    tmp=RK4step(tmp,t,dt);
  }
  else if (method==4)
  {
    //If implicit RK4 was chosen as method
    tmp=nmet(tmp,t,dt,tmp,pow(dt,5),1);
  }
  else if (method==5)
  {
    //If Crank-Nicholson was chosen as method
    tmp=nmet(tmp,t,dt,tmp,pow(dt,5),2);

  }
  else if (method==6)
  {
    //If implicit Euler's method was chosen as method
    tmp=nmet(tmp,t,dt,tmp,pow(dt,5),3);
  }
  x_final[k][j].push_back(tmp[0]);
  y_final[k][j].push_back(tmp[1]);
  }
}
p_new[k+1][j+1]=0;
int failures=0;
for (int i_partition = 0; i_partition < partitions[j-1].size(); i_partition += 1)
{
  yy=partitions[j-1][i_partition];
  xv=x_final[k][j][i_partition];
  yv=y_final[k][j][i_partition];
  error=exp(-2*(exp(yv)-yv+exp(xv)-0.95*xv))/masstmp;
  /*Find the position of x and y in the reference domain*/
  if (yv>y[0] && yv<y[ny-1] && xv>=x[0] && xv<x[nx-1])
  {
    ll2=floor((yv-y[0])/dy);
    yv=(yv-y[ll2])/dy;
    kk2=floor((xv-x[0])/dx);
    xv=(xv-x[kk2])/dx;

    /*Computing the parts of the uniform cubic B-spline*/
    py4=pow(yv,3)/6;
    py3=-pow(yv,3)/2+pow(yv,2)/2+yv/2+1.0/6;
    py2=pow(yv,3)/2-pow(yv,2)+2.0/3;
    py1=pow(yv,2)/2-yv/2+1.0/6-py4;

    px4=pow(xv,3)/6;
    px3=-pow(xv,3)/2+pow(xv,2)/2+xv/2+1.0/6;
    px2=pow(xv,3)/2-pow(xv,2)+2.0/3;
    px1=pow(xv,2)/2-xv/2+1.0/6-px4;
    pp=(po[kk2][ll2]*px1+po[kk2+1][ll2]*px2+po[kk2+2][ll2]*px3+po[kk2+3][ll2]*px4)*py1
    +(po[kk2][ll2+1]*px1+po[kk2+1][ll2+1]*px2+po[kk2+2][ll2+1]*px3+po[kk2+3][ll2+1]*px4)*py2
    +(po[kk2][ll2+2]*px1+po[kk2+1][ll2+2]*px2+po[kk2+2][ll2+2]*px3+po[kk2+3][ll2+2]*px4)*py3
    +(po[kk2][ll2+3]*px1+po[kk2+1][ll2+3]*px2+po[kk2+2][ll2+3]*px3+po[kk2+3][ll2+3]*px4)*py4;
    error=abs(pp-error);
    if (error>error_max)
    {
      error_max=error;
    }
    /*Corrects, if the value is under zero */
    if (pp<0)
    {
      pp=0;
    }
    if (y_final[k][j][i_partition]==0)
    {
      pp_tab[i_partition-failures]=0;
    }
    else
    {
      double norm_factor=1/(sqrt(2*PI*dt)*by*y_final[k][j][i_partition]);
        double norm_exponent=1/(sqrt(2*dt)*by*y_final[k][j][i_partition]);
      pp_tab[i_partition-failures]=pp*norm_factor*exp(-pow((y[j]-yy)*norm_exponent,2))
      *jac[k][j][i_partition][0]*jac[k][j][i_partition][1];
    }
    success[i_partition-failures]=partitions[j-1][i_partition];
  }
  else
  {
    failures++;
  }
}
for (int i_partition = 0; i_partition < (int)partitions[j-1].size()-1-failures; i_partition += 1)
{
  p_new[k+1][j+1]=p_new[k+1][j+1]+fabs(success[i_partition+1]-success[i_partition])
  *(pp_tab[i_partition]+pp_tab[i_partition+1])/2;
}
```

```
        mass=mass+p_new[k+1][j+1];
      }
    }
    //cout << error_max;
    mass=mass*dx*dy;
    for ( j = 1; j < ny+1; j += 1)
    {
      for ( k = 1; k < nx+1; k += 1)
      {
        //Scales the distribution such that the total mass is 1.
        p_new[k][j]=p_new[k][j]/mass;
        if (t>10)
        {
          avg[j][k]=avg[j][k]+p_new[j][k];
        }
      }
    }
    t=t+dt; //updates the time
    mass=0;
    for (j = 1; j < nx+1; j += 1) {
      for (k = 1; k < ny+1; k += 1) {
        Ex[i+1]=Ex[i+1]+x[j-1]*p_new[j][k]*dx*dy;
        Ey[i+1]=Ey[i+1]+y[k-1]*p_new[j][k]*dx*dy;
      }
    }
  }
  for ( j = 0; j < nx; j += 1)
  {
    for ( k = 0; k < ny-1; k += 1)
    {
      /*Writes the result to a chosen file in a format MATLAB can read
      by using the command dlmread('filenam'). This function returns a
      matrix in MATLAB equal to p_new*/
      graf << p_new[j+1][k+1] << ",";
    }
    graf << p_new[j+1][ny] << endl;
  }
  for (unsigned int j = 0; j < nt+1; j += 1)
  {
    output << j*dt << "," << Ex[j] << "," << Ey[j] << endl;
  }
  output.close(); //closes the output file
  graf.close();
  cout << maxnorm(ex,p_new,nx+2,ny+2);
  return p_new;
}
int main(int argc, char *argv[]) {
  /*
  argv[1] = number of grid points in x-direction
  argv[2] = number of grid points in y-direction
  argv[3] = timestep
  argv[4] = number of timesteps
  argv[5] = number for the method used:
        1: Backward Euler's method
        2: Backward Improved Euler's method
        3: Backward RK4
        4: RK4 forward, implicitly solved
        5: Crank Nicholson, implicitly solved
        6: Forward Euler, implicitly solved
  argv[6] = name of the file to write the results to
  */
  Pathint(atoi(argv[1]),atoi(argv[2]),0,atof(argv[3]),atoi(argv[4]),0.1,0,2,0,2,atoi(argv[5]), argv[6]);
}
```

Bibliography

[1] Eckhard Platen Peter E. Kloeden. *Numerical Solution of Stochastic Differentation Equations*. Springer, third edition, 1999.

[2] Andreas Klein. A generalized kahan-babuška-summation-algorithm, April 2005.

[3] Arnulf Jentzen and Martin Hutzenthaler. Convergence of the stochastic euler scheme for locally lipscitz coefficients, November 2009.

[4] Jan Palczewski. Milstein scheme and convergence, 2009.

[5] A. Naess and V.Moe. Efficient path integration methods for nonlinear dynamic systems. *Probabilistic engineering mechanics*, (15):221–231, 2000.

[6] John N. McDonald and Neil A. Weiss. *A Course in Real Analysis.* Academic Press, 2001.

[7] Carl de Boor. *A Practical Guide to Splines, Revised Edition.* Springer, 2001.

[8] Tore Selland Kleppe. Numerical path integration for lèvy driven stochastic differential equations. Master's thesis, NTNU, May 2006.

[9] Arvid Naess, Michael F. Dimentberg, and Oleg Gaidai. Lotka-volterra systems in environments with randomly disordered temporal periodicity. *Physical review*, (78), 2008.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 2nd edition, 1992.

[11] A. Naess. Lecture notes on the numerical solution of stochastic differential equations. *Preprint Statistics*, (11), 2001.