STUDY OF SOME FUNCTIONALS OF STANDARD AND FRACTIONAL BROWNIAN MOTIONS WITH APPLICATIONS IN QUANTITATIVE FINANCE AND STATISTICS

Timothy Gregory Ling

Supervisor: Professor Alex Novikov

Department of Mathematical Sciences,
University of Technology, Sydney.

February 2014

Submitted in fulfillment of the requirements of the degree of Doctor of Philosophy in Mathematics
Acknowledgements

First I want to thank Professor Novikov for supervising, teaching and advising me over the last 5 years. From my undergraduate studies, through to my honours year and the completion of my doctorate, Prof. Novikov has been a constant mentor to me.

Next, I thank Dr. Alex Radchik for introducing me to the first topic of my thesis and many useful discussions.

Finally I want to thank my family for love and support. In particular, thanks to my dearest wife Iris for everything!
Abstract

This thesis contains results on two important problems arising in quantitative finance and statistics.

The first problem is about option pricing with a volume weighted average price (VWAP) as an underlying process. The VWAP is a very important quantity in finance; it appears in Australian taxation law to specify the price of share-buybacks for publicly-listed companies and it is a standard benchmark price for market participants. Pricing options on VWAP is a challenging problem from a mathematical point of view because it involves two sources of randomness: the price of the asset and its traded volume. To solve the problem we have applied the moment-matching approach to a range of "stock and volume" models and as a result obtained an accurate approximation for prices of "call" options. All results have been verified by intensive Monte Carlo simulation.

The second problem is concerned with study of analytical properties and simulation algorithms for a fractional Brownian motion (fBM) which is considered a good alternative to modelling stochastic processes with long range dependence in modern Mathematical finance. In particular, we have reviewed the known simulation algorithms and have implemented the fastest of them ("circulant embedding") on a modern multicore computer. We applied the algorithm to two longstanding open problems in statistics, namely, to study distributions of exponential functionals of fBm and the maximum of fBm. The results of our simulations exhibit new and striking properties of these distributions.
# Contents

Chapter 1  Introduction ........................................... 2

Chapter 2  Mathematical background ............................... 3
  2.1  Basic tools from probability theory ....................... 3
    2.1.1  Conditional expectation ............................ 3
    2.1.2  Multivariate normal distribution ................. 8
  2.2  Stochastic processes ...................................... 13
    2.2.1  Brownian motion .................................. 14
  2.3  Stochastic Integration .................................... 14
  2.4  Stochastic differential equations ......................... 29
    2.4.1  Geometric Brownian motion ....................... 30
    2.4.2  The Ornstein-Uhlenbeck process ............... 32

I  Options with volume-weighted average price (VWAP) as underlying 35

Chapter 3  Introduction ........................................... 36
  3.1  Overview ................................................. 36
  3.2  Existing literature ....................................... 36
    3.2.1  Volume weighted average price (VWAP) ........... 36
    3.2.2  Asian options .................................... 37
    3.2.3  Options on VWAP ................................. 39
  3.3  Summary of the contributions of part I .................. 39
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Market impact example</td>
<td>40</td>
</tr>
<tr>
<td>6.1</td>
<td>Numerical comparisons for first VWAP moment</td>
<td>65</td>
</tr>
<tr>
<td>6.2</td>
<td>Numerical comparisons for second VWAP moment</td>
<td>65</td>
</tr>
<tr>
<td>11.1</td>
<td>Standard errors for $T^{1-H} \mathbb{P}(\max_{0 \leq t \leq T} W_t^{(H)} &lt; 1)$</td>
<td>97</td>
</tr>
</tbody>
</table>
List of Figures

6.1 Comparison of VWAP call option prices .......................... 66
6.2 Relative error of VWAP call option prices for $K = 100$ ............ 66
6.3 Relative error of VWAP call option prices for $K = 110$ ............ 67
9.1 Memory Hierarchy ............................................. 85
9.2 An example of a simple cache. .................................. 85
9.3 Shared memory microprocessor with two processors ............... 87
10.1 Parallel implementation of crude Monte Carlo simulation ........ 93
11.1 $T^{1-H\eta}_{1-H\eta} \left( \max_{0 \leq t \leq T} W_t^{(H)} < 1 \right)$ for $H = 0.40$. ................. 98
11.2 $T^{1-H\eta}_{1-H\eta} \left( \max_{0 \leq t \leq T} W_t^{(H)} < 1 \right)$ for $H = 0.75$. ................. 98
11.3 $T^{1-H\eta}_{1-H\eta} \left( \max_{0 \leq t \leq T} W_t^{(H)} < 1 \right)$ for $H = 0.95$. ................. 99
11.4 $\text{var}(\zeta_H)$ for a range of $H$ values. Region of integration was truncated to $[-10000,10000]$. .................................................. 99
CERTIFICATE OF ORIGINAL AUTHORSHIP

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student:

Date:

Timothy Gregory Ling
This thesis is divided into two parts; the pricing of options with volume-weighted average price (VWAP) as underlying and computing special functionals of fractional Brownian motion. The work described in this thesis may also be found in the following papers

- *On moments of the limit distribution of Pitman estimators*, A. Novikov, N. Kordzakhia, T.G. Ling. (Accepted, Theory of probability and its applications)
Chapter 2
Mathematical background

In this chapter we describe the mathematics required to give a self-contained presentation of our work.

2.1 Basic tools from probability theory

Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. For $p \in [1, \infty)$ and any random variable $X$ we define

$$\|X\|_p \triangleq \mathbb{E}[|X|^p]^{1/p}.$$ 

Next we define $L^p(\Omega, \mathcal{A}, \mathbb{P}) \triangleq \{X : \Omega \to \mathbb{R} \text{ measurable and } \|X\|_p < \infty\}.$

2.1.1 Conditional expectation

Let $X \in L^1(\Omega, \mathcal{A}, \mathbb{P})$ and $\mathcal{F} \subset \mathcal{A}$ be a sub-$\sigma$-algebra of $\mathcal{A}$.

**Definition 2.1.1.** The conditional expectation of $X$ given $\mathcal{F}$ is a random variable $Y$ such that

(i) $Y$ is $\mathcal{F}$-measurable

(ii) For any $A \in \mathcal{F}$, $\mathbb{E}[X1_A] = \mathbb{E}[Y1_A]$ 

We denote $\mathbb{E}[X|\mathcal{F}] \triangleq Y$. If $Z$ is another random variable then we define $\mathbb{E}[X|Z] \triangleq \mathbb{E}[X|\sigma(Z)].$

Existence of conditional expectation follows from the Radon-Nikodym theorem and it is easy to show almost-sure uniqueness.

We require the following basic properties of conditional expectation

**Theorem 2.1.2.**

(i) $\mathbb{E}[aX + bY|\mathcal{F}] = a\mathbb{E}[X|\mathcal{F}] + b\mathbb{E}[Y|\mathcal{F}]$ for all $a, b \in \mathbb{R}$ (Linearity)
(ii) If \( X \leq Y \) a.s. then \( \mathbb{E}[X|\mathcal{F}] \leq \mathbb{E}[Y|\mathcal{F}] \) (Monotonicity)

(iii) If \( \mathbb{E}[|XY|] < \infty \) and \( Y \) is \( \mathcal{F} \)-measurable then

\[
\mathbb{E}[XY|\mathcal{F}] = Y\mathbb{E}[X|\mathcal{F}] \quad \text{and} \quad \mathbb{E}[Y|\mathcal{F}] = \mathbb{E}[Y|\mathcal{F}] = Y
\]

(iv) \( \mathbb{E}[\mathbb{E}[X|\mathcal{F}]|\mathcal{G}] = \mathbb{E}[\mathbb{E}[X|\mathcal{G}]|\mathcal{F}] = \mathbb{E}[X|\mathcal{G}] \) (Tower property)

(v) If \( \sigma(X) \) and \( \mathcal{F} \) are independent then \( \mathbb{E}[X|\mathcal{F}] = \mathbb{E}[X] \)

**Proof.** See any graduate textbook on probability (for e.g. Shiryaev [54]).

We will need to explicitly calculate conditional expectations of the form \( \mathbb{E}[X|Y] \) but the formal definition above gives no indication how to do so. It is easy to calculate for certain cases and we now describe how.

**Theorem 2.1.3.** Let \( X \) be an integrable random variable on \( (\Omega, \mathcal{A}) \) and \( Y : (\Omega, \mathcal{A}) \rightarrow (\Omega', \mathcal{A}') \) be a measurable map. Then there exists a \( \mathbb{P}_Y \)-integrable and \( \mathbb{P}_Y \)-almost surely unique function \( g : (\Omega', \mathcal{A}') \rightarrow (\mathbb{R}, \mathcal{B}(\mathbb{R})) \) such that \( g \circ Y \) is a version of \( \mathbb{E}[X|Y] \). We define \( \mathbb{E}[X|Y = y] \triangleq g(y) \).

**Proof.** For all \( A \in \mathcal{A}' \) define \( \mu(A) = \mathbb{E}[1_{\{Y \in A\}}X] \). It is not hard to show that \( \mu \) is a signed measure on \( (\Omega', \mathcal{A}') \). Furthermore,

Now let \( A \in \mathcal{A}' \) such that \( \mathbb{P}_Y(A) = 0 \) where \( \mathbb{P}_Y \) denotes the distribution of \( Y \). Then

\[
\mathbb{E}[1_{\{Y \in A\}}] = 0
\]

\[\Rightarrow 1_{\{Y \in A\}} = 0 \ \mathbb{P}\text{-a.s.} \]

\[\Rightarrow 1_{\{Y \in A\}}X = 0 \ \mathbb{P}\text{-a.s.} \]

\[\Rightarrow \mu(A) = 0 \]
So $\mu \ll \mathbb{P}_Y$. By the Radon-Nikodym theorem there exists a $\mathbb{P}_Y$-integrable function $g : (\Omega', \mathcal{A}') \to (\mathbb{R}, \mathcal{B}(\mathbb{R}))$ such that
\begin{align*}
\mathbb{E}[1_{\{Y \in A\}}X] &= \int_A g d\mathbb{P}_Y \\
&= \int_{\{Y \in A\}} g(Y) d\mathbb{P} \\
&= \mathbb{E}[1_{\{Y \in A\}}g(Y)]
\end{align*}
where the second equality follows from the integral projection theorem B.2.1. So for all $B \in \sigma(Y)$, $\mathbb{E}[X1_B] = \mathbb{E}[g(Y)1_B]$ thus $g \circ Y$ is indeed a version of $\mathbb{E}[X|Y]$. To show $\mathbb{P}_Y$-a.s. uniqueness, let $h : (\Omega', \mathcal{A}') \to (\mathbb{R}, \mathcal{B}(\mathbb{R}))$ such that $h \circ Y = \mathbb{E}[X|Y]$. Since $\mathbb{E}[X|Y]$ is integrable, we have from the integral projection theorem and (2.1) that
\begin{align*}
\int_A hd\mathbb{P}_Y &= \mathbb{E}[1_{\{Y \in A\}}h \circ Y] = \mathbb{E}[1_{\{Y \in A\}}X] = \int_A g d\mathbb{P}_Y
\end{align*}
and result follows. \hfill $\square$

$\mathbb{E}[X|Y = y]$ can be explicitly calculated when the distribution of $(X, Y)$ has a density.

Recall that a probability measure $\mathbb{P}$ on $(\mathbb{R}^n, \mathcal{B}^n)$ has a density $f$ if $f$ is a nonnegative measurable function on $\mathbb{R}^n$ satisfying for all $A \in \mathcal{B}^n$, $\mathbb{P}(A) = \int_A f d\lambda^n$.

**Theorem 2.1.4.** Let $X$ and $Y$ be random variables with a joint density function $f$ on $\mathbb{R}^2$. Then $X$ and $Y$ each have a density function on $\mathbb{R}$ given by:
\begin{align*}
  f_X(x) &= \int_{-\infty}^\infty f(x, y) dy; \quad f_Y(y) = \int_{-\infty}^\infty f(x, y) dx
\end{align*}

The densities $f_X$ and $f_Y$ are called the marginal densities of $f$. 

5
Proof. Let $A \in \mathcal{B}$ and consider

$$
P [X \in A] = P [(X, Y) \in A \times \mathbb{R}]
= \int_{A \times \mathbb{R}} f d\lambda^2
= \int_A \int_{\mathbb{R}} f(x, y) dy dx \quad \text{by Fubini's theorem}
= \int_A f_X(x) dx
$$

So $f_X$ is indeed a density for the random variable $X$. Proof for $Y$ is the same. \qed

Theorem 2.1.5. Further to theorem 2.1.3, suppose that $Y$ is also a random variable and that the joint distribution of $(X, Y)$ has a density function $f$ with strictly positive marginal density $f_Y$. Then given an integrable function $h : \mathbb{R} \to \mathbb{R}$, $E[h(X)|Y = y]$ satisfies

$$
E[h(X)|Y = y] = \frac{1}{f_Y(y)} \int h(x) f(x, y) dx, \quad \text{for } P_Y\text{-almost all } y \in \mathbb{R}
$$

and

$$
E[h(X)|Y] = \frac{1}{f_Y(Y)} \int h(x) f(x, Y) dx, \quad P\text{-almost surely}
$$

Proof. By Fubini’s theorem, the mapping $y \mapsto \int h(x) f(x, y) dx$ is measurable. We now show that it is $\lambda$-integrable. Note by theorem 2.1.4 that $f_X(x) = \int f(x, y) dy$ is a density for $X$. So

$$
E [|h(X)|] = \int_\mathbb{R} |h(x)| f_X(x) dx = \int_\mathbb{R} \int_\mathbb{R} |h(x)| f(x, y) dx dy
$$
where the last equality is due to Fubini’s theorem. Next consider
\[
\int_{\mathbb{R}} \left| \int_{\mathbb{R}} h(x)f(x,y)\,dx \right| \,dy \leq \int_{\mathbb{R}} \int_{\mathbb{R}} |h(x)f(x,y)| \,dx \,dy = \mathbb{E} \left[ |h(X)| \right] < +\infty
\]
thus proving \( \lambda \)-integrability. Since \( \int_{\mathbb{R} \times \mathbb{R}} f \,d\lambda^2 = 1 \), it is easy to see that \( f_Y(y) \) is \( \lambda \)-integrable. Now recall that integrability of a nonnegative function implies that it is almost everywhere finite. So there exists some \( N \in \mathcal{B}(\mathbb{R}) \), \( \lambda(N) = 0 \) such that for all \( y \in \mathbb{R} \setminus N \)
\[
\int_{\mathbb{R}} |h(x)f(x,y)| \,dx < \infty, \quad f_Y(y) < \infty.
\]

Next note that \( \mathbb{P}_Y(N) = \int_{N^c} f_Y(y) \,dy = 0 \). Now we define a measurable function \( g : \mathbb{R} \to \mathbb{R} \) by
\[
g(y) = \frac{1_{N^c}(y)}{f_Y(y)} \int_{\mathbb{R}} h(x)f(x,y) \,dx.
\]

Note that \( g \) is \( \mathbb{P}_Y \)-integrable:
\[
\int_{\mathbb{R}} |g| \,d\mathbb{P}_Y = \int_{\mathbb{R}} |g| \,f_Y \,d\lambda = \int_{N^c} \left| \int_{\mathbb{R}} h(x)f(x,y) \,dx \right| \,dy \\
\leq \int_{\mathbb{R}} \int_{\mathbb{R}} |h(x)| \,f(x,y) \,dxdy \\
= \mathbb{E} \left[ |h(X)| \right] < \infty.
\]
Using the integral projection theorem B.2.1, $g \circ Y$ is $\mathbb{P}$-integrable and for any $A \in \mathcal{B}(\mathbb{R})$

$$
\mathbb{E} \left[ 1_{\{Y \in A\}} g \circ Y \right] = \int_A g d\mathbb{P}_Y
= \int_\mathbb{R} 1_A(y) g(y) f_Y(y) dy
= \int_{\mathbb{R}^\mathbb{N}_c} 1_A(y) \int_\mathbb{R} h(x) f(x, y) dx dy
= \int_A \left( \int_\mathbb{R} h(x) f(x, y) dx \right) dy
= \int_{\mathbb{R} \times A} h(x) f(x, y) (dx, dy)
= \mathbb{E} \left[ 1_{\{Y \in A\}} h(X) \right]
$$

Thus $g \circ Y$ is a version of $\mathbb{E} [h(X)|Y]$.

\[\square\]

**Corollary 2.1.6.** Let $X$ and $Y$ be random variables with a joint density function $f$ on $\mathbb{R}^2$ with a strictly positive marginal density $f_Y$ and suppose that $X$ is integrable. Then the conditional distribution of $X$ given $Y$ has a density

$$
f_{X|Y}(x, y) \triangleq \frac{f(x, y)}{f_Y(y)}
$$

**Proof.** For each $B \in \mathcal{B}(\mathbb{R})$ set $h(x) = 1_B$. Then applying theorem 2.1.5,

$$
\mathbb{P} [X \in B|Y = y] \triangleq \mathbb{E} \left[ 1_{\{X \in B\}}|Y = y \right] = \int_B \frac{f(x, y)}{f_Y(y)} dx
$$

\[\square\]

**2.1.2 Multivariate normal distribution**

Here we collect results for the multivariate normal distribution that are required in future chapters.
Definition 2.1.7. A $\mathbb{R}^n$-valued random variable $X = (X_1, \cdots, X_n)$ has a Multivariate Normal distribution if it has characteristic function

$$
\varphi_X(u) = \exp \left\{ i \langle u, \mu \rangle - \frac{1}{2} \langle u, Qu \rangle \right\}
$$

where $\mu = \mathbb{E}[X]$ and $Q$ is the covariance matrix of $X$ i.e. $Q_{ij} = \text{Cov}(X_i, X_j)$ for all $i, j \in \{1, \cdots, n\}$. We use the notation $X \sim N_n(\mu, Q)$. For confirmation that $\varphi_X(u)$ is indeed a characteristic function, see for e.g. Shiryaev [54, p. 299]

The following theorem gives an equivalent definition for the Multivariate distribution.

Theorem 2.1.8. $X$ is Multivariate Normal with dimension $n$ if and only if every linear combination $\sum_{i=1}^n a_i X_i$ of components of $X$ has a univariate Normal distribution.

Proof. Suppose that $X \sim N_n(\mu, Q)$. Let $Y = \langle a, X \rangle = \sum_{j=1}^n a_j X_j$ be any linear combination of components of $X$. Now consider for any $v \in \mathbb{R}$,

$$
\varphi_Y(v) = \mathbb{E} \left[ e^{iv \sum_{j=1}^n a_j X_j} \right] \\
= \mathbb{E} \left[ e^{i\langle av, X \rangle} \right] \\
= \varphi_X(\langle av \rangle) \\
= \exp \left\{ iv \langle a, \mu \rangle - \frac{1}{2} v^2 \langle a, Qa \rangle \right\}
$$

So $Y$ has a univariate Normal distribution with mean $\langle a, \mu \rangle$ and variance $\langle a, Qa \rangle$.

Conversely suppose that every linear combination of components of $X$ is univariate normal. Take $Y$ defined as above. Let $Q$ denote the covariance matrix of $X$ and $\mu = \mathbb{E}[X]$. 

By linearity of expectation we have $E[Y] = \langle a, \mu \rangle$ and

\[
\text{Var}(Y) = \text{Cov}(Y,Y) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \text{Cov}(X_i, X_j) = \sum_{i=1}^{n} a_i Qa = \langle a, Qa \rangle.
\]

Since $Y$ is univariate normal, by definition 2.1.7 the characteristic function of $Y$ is

\[
\varphi_Y(v) = \exp \left\{ iv\langle a, \mu \rangle - \frac{1}{2} v^2 \langle a, Qa \rangle \right\}
\]

Now set $\varphi_Y(1) = \varphi_{(a,X)}(1) = E[\exp \{ i\langle a, X \rangle \}] \triangleq \varphi_X(a)$. So by definition $X$ is multivariate normal.

**Theorem 2.1.9.** (Existence of density) Let $X \sim N_n(\mu, Q)$. Then $X$ has a density if and only if its covariance matrix $Q$ is non-degenerate i.e. $\det(Q) \neq 0$. If the density exists, then it is given by

\[
f_X(x) = \frac{1}{2\pi^{n/2}\sqrt{\det Q}} \exp \left\{ -\frac{1}{2} \langle x - \mu, Q^{-1}(x - \mu) \rangle \right\}
\]

**Theorem 2.1.10.** (Conditional distributions) Let $X$ be $n$-dimensional Multivariate Normal with $n \geq 2$, mean vector $\mu$ and covariance matrix $Q$ with $\det(Q) \neq 0$. Let $q \in \{1, \cdots, n-1\}$ and suppose that $X, \mu$ and $Q$ are partitioned as follows

\[
X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \quad Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}
\]

where
• $X_1$ and $\mu_1$ are column vectors of length $q$.
• $X_2$ and $\mu_2$ are column vectors of length $(n - q)$.
• $Q_{11}$ is a $q \times q$ matrix.
• $Q_{12}$ is a $q \times (n - q)$ matrix.
• $Q_{21}$ is a $(n - q) \times q$ matrix.
• $Q_{22}$ is a $(n - q) \times (n - q)$ matrix.

Then the distribution of $X_1$ conditional on $X_2 = a$, which we denote by $(X_1 | X_2 = a)$, is $q$-dimensional Multivariate Normal with mean and covariance given by

$$
\tilde{\mu} = \mu_1 + Q_{12}Q_{22}^{-1}(a - \mu_2) \quad \text{and} \\
\tilde{Q} = Q_{11} - Q_{12}Q_{22}^{-1}Q_{21}
$$

Proof. Define the $n \times n$ matrix

$$
A = \begin{bmatrix}
I_q & -Q_{12}Q_{22}^{-1} \\
0 & I_{(n-q)}
\end{bmatrix}
$$

where $I_q$ and $I_{(n-q)}$ denote $q \times q$ and $(n - q) \times (n - q)$ identity matrices, respectively. By standard properties of determinants, it follows that $\det(A) = 1$ and so $A$ is invertible. Using another standard result from linear algebra we have

$$
\det(Q) = \det(Q_{22})\det(Q_{11} - Q_{12}Q_{22}^{-1}Q_{21}) \quad (2.2)
$$

It follows in particular that $Q_{11} - Q_{12}Q_{22}^{-1}Q_{21}$ is invertible. Next consider,

$$
AQA^\top = \begin{bmatrix}
I_q & -Q_{12}Q_{22}^{-1} \\
0 & I_{(n-q)}
\end{bmatrix}
\begin{bmatrix}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{bmatrix}
\begin{bmatrix}
I_q & 0 \\
(-Q_{12}Q_{22}^{-1})^\top & I_{(n-q)}
\end{bmatrix}
= \begin{bmatrix}
Q_{11} - Q_{12}Q_{22}^{-1}Q_{21} & 0 \\
0 & Q_{22}
\end{bmatrix} \quad (2.3)
$$
Pre-multiplying and post-multiplying both sides of (2.3) by $A^{-1}$ and $(A^{-1})^\top$ respectively gives

$$Q = A^{-1} \begin{bmatrix} Q_{11} - Q_{12}Q_{22}^{-1}Q_{21} & 0 \\ 0 & Q_{22} \end{bmatrix} (A^{-1})^\top$$

$$\Rightarrow Q^{-1} = A^\top \begin{bmatrix} Q_{11} - Q_{12}Q_{22}^{-1}Q_{21} & 0 \\ 0 & Q_{22} \end{bmatrix}^{-1} A$$

$$= A^\top \begin{bmatrix} (Q_{11} - Q_{12}Q_{22}^{-1}Q_{21})^{-1} & 0 \\ 0 & Q_{22}^{-1} \end{bmatrix} A \quad (2.4)$$

A simple calculation gives

$$A(x - \mu) = \begin{bmatrix} x_1 - \mu_1 - Q_{12}Q_{22}^{-1}(x_2 - \mu_2) \\ x_2 - \mu_2 \end{bmatrix} \quad (2.5)$$

Using (2.4) and (2.5)

$$(x - \mu)^\top Q^{-1}(x - \mu)$$

$$= (x - \mu)^\top A^\top \begin{bmatrix} (Q_{11} - Q_{12}Q_{22}^{-1}Q_{21})^{-1} & 0 \\ 0 & Q_{22}^{-1} \end{bmatrix} A(x - \mu)$$

$$= \begin{bmatrix} x_1 - \mu_1 - Q_{12}Q_{22}^{-1}(x_2 - \mu_2) \\ x_2 - \mu_2 \end{bmatrix}^\top \begin{bmatrix} (Q_{11} - Q_{12}Q_{22}^{-1}Q_{21})^{-1} & 0 \\ 0 & Q_{22}^{-1} \end{bmatrix} \begin{bmatrix} x_1 - \mu_1 - Q_{12}Q_{22}^{-1}(x_2 - \mu_2) \\ x_2 - \mu_2 \end{bmatrix}$$

$$= [x_1 - \mu_1 - Q_{12}Q_{22}^{-1}(x_2 - \mu_2)] (Q_{11} - Q_{12}Q_{22}^{-1}Q_{21})^{-1} [x_1 - \mu_1 - Q_{12}Q_{22}^{-1}(x_2 - \mu_2)]$$

$$+ (x_2 - \mu_2)Q_{22}^{-1}(x_2 - \mu_2) \quad (2.6)$$
Combining theorem 2.1.9 with (2.2) and (2.6) we can see that the density of \( X \), \( f(x_1, x_2) \) can be expressed as the product of the densities of the random variables

\[
Y \sim N_q(\mu_1 + Q_{12}Q_{22}^{-1}(x_2 - \mu_2), Q_{11} - Q_{12}Q_{22}^{-1}Q_{21}) \quad \text{and} \\
X_2 \sim N_{n-q}(\mu_2, Q_{22})
\]

Setting \( x_2 = a \) and dividing \( f(x_1, a) \) by the marginal density \( f_{X_2}(a) \) we conclude from theorem 2.1.6 that \( Y \) has the required conditional distribution. \( \square \)

2.2 Stochastic processes

A fundamental building block that is needed for our work is a stochastic process which we can regard as a collection of random variables \( X = \{X_t : t \in T\} \) where \( T \) is an index set (usually used to denote time in applications). When \( T \) is a countable subset of the real line (e.g. \( T = \{0, 1, 2, \ldots\} \)) the process is said to be a discrete-time process. If \( T \) is an interval of the real line, then \( X_t \) is called a continuous-time process.

We use stochastic processes to model the asset price and volumes traded.

It is also useful to be able to keep track of past and present behaviour of a stochastic process, which leads to a construct called a natural filtration. The natural filtration which we denote by \( \mathcal{F}_t \) is defined to be the \( \sigma \)-algebra generated by the pre-images of all elements of \( \mathcal{F} \) under \( X_s \) for all \( s \) up to and including time \( t \). Stated formally, \( \mathcal{F}_t = \sigma \{X_s^{-1}(A) : s \in T, s \leq t, A \in \mathcal{F}\} \). \( \mathcal{F}_t \) contains all the information about the process \( X \) up to and including time \( t \).

Filtrations are needed in particular to define a class of stochastic processes called martingales. A \( \mathcal{F}_t \)-adapted\(^1\) process \( Z_t \) is said to be a martingale if \( E(Z_t|\mathcal{F}_s) = Z_s \), for all \( 0 \leq s \leq t \).

\(^1\)A process \( Z_t \) is said to be adapted to \( \mathcal{F}_t \) if \( Z_t \) is \( \mathcal{F}_t \)-measurable for all \( t \geq 0 \).
2.2.1 Brownian motion

A stochastic process \( W_t, t \in [0, \infty) \) is said to be a \textit{Brownian motion} (or a \textit{Wiener process}), if \( W_t \) is a Gaussian process\(^2\) with

\[
E(W_t) = mt, \quad \text{Cov}(W_t, W_s) = \sigma^2 \min(t, s)
\]

for all \( t, s \in [0, \infty) \). In particular, this implies \( W_t \sim N(mt, \sigma^2 t) \). We say that \( W_t \) is \textit{standard Brownian motion} if \( m = 0 \) and \( \sigma = 1 \).

Brownian motion is the fundamental process in the theory of stochastic processes. This process “is to stochastic process theory as the normal distribution is to the theory for real random variables” [49].

2.3 Stochastic Integration

We use stochastic differential equations (SDE) to model the asset price and volume traded over time. This requires the theory of stochastic integration. In this section, we provide a detailed construction of the stochastic integral of a so-called \textit{progressively measurable} process with respect to Brownian motion.

\textbf{Definition 2.3.1.} Let \( X = (X_t)_{t \geq 0} \) be a real-valued stochastic process on \( (\Omega, \mathcal{F}, \mathbb{P}) \). We say that \( X \) is

- \textbf{product measurable} if \((\omega, t) \mapsto X_t(\omega) \) is \( \mathcal{F} \otimes \mathcal{B}([0, \infty)) \)-\( \mathcal{B}(\mathbb{R}) \) measurable.
- \textbf{progressively measurable} if for every \( t \geq 0 \) the map \( \Omega \times [0, t] \to \mathbb{R}, (\omega, s) \mapsto X_s(\omega) \) is \( \mathcal{F}_t \otimes \mathcal{B}([0, t]) \)-\( \mathcal{B}(\mathbb{R}) \) measurable.
- \textbf{predictable} if \((\omega, t) \mapsto X_t(\omega) \) is \( \mathcal{P} \)-\( \mathcal{B}(\mathbb{R}) \) measurable, where \( \mathcal{P} \) denotes the \textit{predictable} \( \sigma \)-algebra on \( \Omega \times [0, \infty) \):

\[
\mathcal{P} \triangleq \sigma(Y : Y \text{ is left continuous and adapted})
\]

\(^2\)A stochastic process \( X(t), t \in D \) (where \( D \) is an index set) is called a \textit{Gaussian process} if any random vector \( X = (X(t_1), X(t_2), \ldots, X(t_n))^T \) is Gaussian.
Let $\mathcal{E}$ denote the set of all processes $X$ which are product measurable, adapted and satisfy $\|X\|_2^2 \triangleq \mathbb{E}\left[\int_0^\infty X_s^2\,ds\right] < \infty$. Note that $\|\cdot\|_E$ defines a seminorm on $\mathcal{E}$.

We first define the stochastic integral for simple integrands.

**Definition 2.3.2.** Let $\mathcal{E}_0$ denote the vector space of maps $G : \Omega \times [0, \infty) \to \mathbb{R}$ of the form

$$G_t(\omega) = \sum_{i=1}^n g_{i-1}(\omega) 1_{(t_{i-1}, t_i]}(t)$$

where $t_0 = 0$, $t_0 < t_1 < \cdots < t_n$ and $g_{i-1}$ is bounded and $\mathcal{F}_{t_{i-1}}$-measurable for all $i = 1$ to $n$. Clearly $G$ is an adapted, left-continuous process and therefore predictable. We call $\mathcal{E}_0$ the vector space of *simple, predictable processes*. It is routine to check that $\mathcal{E}_0$ is indeed a vector space.

**Definition 2.3.3.** For $G \in \mathcal{E}_0$ and $t \geq 0$ define the following mappings from $\mathcal{E}_0$ to $L^2(\Omega, \mathcal{F}, \mathbb{P})$

$$I_t(G) = \sum_{i=1}^n g_{i-1}(W_{t_i \wedge t} - W_{t_{i-1} \wedge t})$$
$$I_\infty(G) = \sum_{i=1}^n g_{i-1}(W_{t_i} - W_{t_{i-1}})$$

**Theorem 2.3.4.**

(i) The process $(I_t(G))_{t \geq 0}$ is a $L^2$-bounded continuous $\mathbb{F}$-martingale.

(ii) $\mathbb{E}[I_\infty(G)|\mathcal{F}_s] = I_s(G)$ for all $s \in \mathbb{R}^+$.

**Proof.** (i): We first show that $(I_t(G))_{t \geq 0}$ is adapted and continuous. We can then use the optional sampling theorem to show that it is a martingale.
Let \( t \in [0, \infty) \). If \( t_1 < t \leq t_n \), there exists a unique integer \( m \in \{1, \ldots, n-1\} \) such that \( t_m < t \leq t_{m+1} \). Now \( I_t(G) \) may be expressed as

\[
I_t(G) = \begin{cases} 
  g_0 W_t & \text{if } t \leq t_1 \\
  \sum_{i=1}^{m} g_{i-1} (W_{t_i} - W_{t_{i-1}}) + g_m (W_t - W_{t_m}) & \text{if } t_1 < t \leq t_n \\
  \sum_{i=1}^{n} g_{i-1} (W_{t_i} - W_{t_{i-1}}) & \text{if } t > t_n
\end{cases}
\] (2.7)

We know that Brownian motion is continuous and a martingale, so it follows from the previous expression that \( I_t(G) \) is continuous and adapted. Next let \( \tau \) be a bounded stopping time and consider

\[
\mathbb{E} [I_\tau(G)] = \sum_{i=1}^{n} \mathbb{E} \left[ g_{i-1} \left( W_{t_i}^\tau - W_{t_{i-1}}^\tau \right) \right]
\]

where \( W_t^\tau \equiv W_{t^\wedge \tau} \) (stopped process)

\[
= \sum_{i=1}^{n} \mathbb{E} \left[ g_{i-1} \mathbb{E} \left[ W_{t_i}^\tau - W_{t_{i-1}}^\tau | \mathcal{F}_{t_{i-1}} \right] \right]
\]

by theorem 2.1.2 (iii) and (iv)

\[
= 0
\]

\[
= \mathbb{E} [I_0(G)]
\]

where the second-last equality holds since \( W^\tau \) is a \( \mathbb{F} \)-martingale by the optional stopping theorem. \( I_t(G) \) is then a \( \mathbb{F} \)-martingale by the optional sampling theorem. It is also clearly \( L^2 \) bounded.

(ii): From (2.7) it can be seen that for any \( t > t_n \), \( I_\infty(G) = I_t(G) \). Then for any \( t > s \vee t_n \)

\[
\mathbb{E} [I_\infty(G) | \mathcal{F}_s] = \mathbb{E} [I_t(G) | \mathcal{F}_s]
\]

\[
= I_s(G)
\]

since \( I_t \) is a martingale by (i)

\[\square\]

**Theorem 2.3.5.** \( I_\infty(G) \) and \( I_t(G) \), \( t \in \mathbb{R}^+ \) are linear isometries with respect to the seminorms \( \|\cdot\|_E \) and \( \|\cdot\|_2 \).
Proof. We first show that for all \( i, j \in \{0, \cdots, n-1\}, i \neq j \)

\[
\mathbb{E} \left[ (I_{t_{i+1}}(G) - I_t(G))(I_{t_{j+1}}(G) - I_t(G)) \right] = 0 \tag{2.8}
\]

Assume \( j < i \). Then

\[
\mathbb{E}[(I_{t_{i+1}}(G) - I_t(G))(I_{t_{j+1}}(G) - I_t(G))]
= \mathbb{E}[g_i(W_{t_{i+1}} - W_{t_i})g_j(W_{t_{j+1}} - W_{t_j})]
= \mathbb{E}[g_ig_j(W_{t_{i+1}} - W_{t_i})\mathbb{E}[W_{t_{j+1}} - W_{t_j}|\mathcal{F}_{j}]] \quad \text{by theorem 2.1.2 (iii), (iv)}
= 0
\]

Next we show that for any \( m \in \{1, \cdots, n\} \)

\[
\mathbb{E} \left[ \left( \sum_{i=1}^{m} (W_{t_i} - W_{t_{i-1}}) \right)^2 \right] = \sum_{i=1}^{m} \mathbb{E} \left[ (W_{t_i} - W_{t_{i-1}})^2 \right] \tag{2.9}
\]

Indeed,

\[
\mathbb{E} \left[ \left( \sum_{i=1}^{n} (W_{t_i} - W_{t_{i-1}}) \right)^2 \right] = \mathbb{E} \left[ \left( \sum_{i=1}^{n} (I_t(G) - I_{t_{i-1}}(G)) \right)^2 \right]
= \mathbb{E} \left[ \sum_{i=1}^{n} (I_t(G) - I_{t_{i-1}}(G))^2 \right] \quad \text{by (2.8)}
= \sum_{i=1}^{n} \mathbb{E} \left[ g_{i-1}^2 (W_{t_i} - W_{t_{i-1}})^2 \right]
= \sum_{i=1}^{n} \mathbb{E} \left[ \mathbb{E} \left[ g_{i-1}^2 (W_{t_i} - W_{t_{i-1}})^2 |\mathcal{F}_{t_{i-1}} \right] \right]
= \sum_{i=1}^{n} g_{i-1}^2 \mathbb{E} \left[ (W_{t_i} - W_{t_{i-1}})^2 \right]
= \sum_{i=1}^{n} g_{i-1}^2 \mathbb{E} \left[ (t_i - t_{i-1}) \right]
\]

Setting \( m = n \) in (9.2) gives us \( \|I_\infty(G)\|_2^2 = \mathbb{E} \left[ \int_0^\infty G_s^2 ds \right] \) which proves that \( I_\infty \) is an isometry. For a proof that \( I_\infty \) is linear see theorem B.3.4. Similarly we can show that \( I_t \)
is a linear isometry by considering the representation in (2.7).

To define the stochastic integral for progressively measurable integrands, we will show that there exists an almost-surely unique continuous extension of $I_\infty$ from $\mathcal{E}_0$ to $\mathcal{E}$.

Using the seminorm $\|\cdot\|_\mathcal{E}$ we may define a pseudometric $d_\mathcal{E}$ on $\mathcal{E}$ by $d_\mathcal{E}(G_1,G_2) = \|G_1 - G_2\|_\mathcal{E}$. This gives rise to the so called seminorm topology on $\mathcal{E}$

$$\tau_{d_\mathcal{E}} = \{U \subseteq E : \forall G \in U, \exists \delta > 0, B(G,\delta) \subseteq U\}$$

where $B(G,\delta) = \{H \in \mathcal{E} : d_\mathcal{E}(G,H) < \delta\}$. Now let $A \in \tau_{d_\mathcal{E}}$. Then the closure of $A$ in $\mathcal{E}$ is defined as $\bar{A} = \{G \in \mathcal{E} : \forall \delta > 0, B(G,\delta) \cap A \neq \emptyset\}$. We now show that all processes in $\mathcal{E}$ can be approximated by the simple processes i.e. $\mathcal{E}_0$ is dense in $\mathcal{E}$. We start with the following lemma.

**Lemma 2.3.6.** Let $G \in \mathcal{E}$ and suppose that for all $T > 0$, there exists a sequence $(G^{(n,T)})_{n \in \mathbb{N}}$ of elements of $\mathcal{E}_0$ such that $\lim_{n \to \infty} \mathbb{E}\left[\int_0^T (G_s - G_s^{(n,T)})^2 \, ds\right] = 0$ then $G \in \bar{\mathcal{E}}_0$.

**Proof.** We have by hypothesis that for any positive integer $m$, there exists a sequence $(G^{(n,m)})_{n \in \mathbb{N}}$ in $\mathcal{E}_0$ such that $\lim_{n \to \infty} \mathbb{E}\left[\int_0^m (G_s - G_s^{(n,m)})^2 \, ds\right] = 0$. So there exists an integer $p_m$ such that $\mathbb{E}\left[\int_0^m (G_s - G_s^{(p_m,m)})^2 \, ds\right] < 1/m$. So the sequence $(G^{(p_m,m)})_{m \in \mathbb{N}}$ satisfies

$$\sup_{T > 0} \lim_{m \to \infty} \mathbb{E}\left[\int_0^T (G_s - \tilde{G}_s^{(p_m,m)})^2 \, ds\right] = 0$$

i.e. $\sup_{T > 0} \lim_{n \to \infty} \mathbb{E}\left[\int_0^T (G_s - \tilde{G}_s^{(n)})^2 \, ds\right] = 0$ where $\tilde{G}^{(n)} = G^{(p_m,n)}$

$$\Rightarrow \lim_{n \to \infty} \mathbb{E}\left[\int_0^\infty (G_s - \tilde{G}_s^{(n)})^2 \, ds\right] = 0$$

So for any $\delta > 0$, $B(G,\delta) \cap \mathcal{E}_0 \neq \emptyset$ and this implies that $G \in \bar{\mathcal{E}}_0$. \qed

**Lemma 2.3.7.** If $G$ is an adapted process that is left or right continuous, then $G$ is also progressively measurable.
Proof. Suppose that $G$ is right-continuous. Fix any real $T > 0$. For every $n \in \mathbb{N}$ define a process $G^n : \Omega \times [0, T] \to \mathbb{R}$ by

$$G^n_t(\omega) = \begin{cases} G_{Ti/2^n}(\omega), & \text{for } t \in \left(\frac{T(i-1)}{2^n}, \frac{Ti}{2^n}\right) \text{ and } i \in \{1, \ldots, 2^n\} \\ G_0(\omega), & \text{for } t = 0 \end{cases}$$

We next show that for each $n \in \mathbb{N}$, $G^n$ is a $\mathcal{F}_T \otimes \mathcal{B}([0, T]) - \mathcal{B}(\mathbb{R})$ measurable process. Let $B \in \mathcal{B}(\mathbb{R})$. We now show that

$$\{(\omega, t) \in \Omega \times [0, T] \mid G^n_t(\omega) \in B\} = \left(\bigcup_{i=1}^{2^n} \{G_{i/2^n} \in B\} \times \left(\frac{T(i-1)}{2^n}, \frac{Ti}{2^n}\right)\right)$$

$$\cup \{G_0 \in B\} \times \{0\}.$$

Let $(\omega, t) \in \{(\omega, t) \in \Omega \times [0, T] \mid G^n_t(\omega) \in B\}$. If $t = 0$ then clearly $(\omega, t) \in \{G_0 \in B\} \times \{0\}$. Otherwise there exists an $i \in \{1, \ldots, 2^n\}$ such that $t \in \left(\frac{T(i-1)}{2^n}, \frac{Ti}{2^n}\right]$. Then $G_{i/2^n}(\omega) = G^n_t(\omega) \in B$. This implies that

$$\{(\omega, t) \in \Omega \times [0, T] \mid G^n_t(\omega) \in B\} \subseteq \left(\bigcup_{i=1}^{2^n} \{G_{i/2^n} \in B\} \times \left(\frac{T(i-1)}{2^n}, \frac{Ti}{2^n}\right)\right)$$

$$\cup \{G_0 \in B\} \times \{0\}.$$

The proof of the reverse containment is similar. Since $G$ is a stochastic process the set

$$\{G_t \in B\} \in \mathcal{F}_T$$

for all $t \in [0, T]$. It then follows that $\{(\omega, t) \in \Omega \times [0, T] \mid G^n_t(\omega) \in B\} \in \mathcal{F}_T \otimes \mathcal{B}([0, T])$ and thus $G^n$ is $\mathcal{F}_T \otimes \mathcal{B}([0, T]) - \mathcal{B}(\mathbb{R})$ measurable.

Next choose a $t \in [0, T)$, $\omega \in \Omega$ and any $\varepsilon > 0$. By right-continuity of $G$ there exists $\delta > 0$ such that for all $s \in (t, t + \delta) \subseteq [0, T]$ we have $|G_t(\omega) - G_s(\omega)| < \varepsilon$. We then select any positive integer $n$ such that $\frac{1}{2^n} < \delta$. By a property of the real numbers there exists an integer $i \in \{1, \ldots, 2^n\}$ such that $t \in \left(\frac{T(i-1)}{2^n}, \frac{Ti}{2^n}\right]$. So it follows that $(\frac{T(i-1)}{2^n}, \frac{Ti}{2^n}] \subseteq (t, t + \delta)$ and thus

$$|G_t(\omega) - G^n_t(\omega)| = |G_t(\omega) - G_{Ti/2^n}(\omega)| < \varepsilon.$$
This proves that $G^n \to G$ on $\Omega \times [0, T]$, therefore the restriction of $G$ to $\Omega \times [0, T]$ is also $\mathcal{F}_T \otimes \mathcal{B}([0, T]) - \mathcal{B}(\mathbb{R})$ measurable. Therefore $G$ is progressively measurable. 

\textbf{Theorem 2.3.8.} $\bar{\mathcal{E}}_0 = \mathcal{E}$.

\textbf{Proof.} By lemma 2.3.7, $\bar{\mathcal{E}}_0 \subseteq \mathcal{E}$ (note that predicatable processes are left-continuous).

Now let $G \in \mathcal{E}$. By lemma 2.3.6 it is enough to show for any $T > 0$, there exists a sequence $(G^n)_{n \in \mathbb{N}}$ in $\mathcal{E}_0$ such that

$$\lim_{n \to \infty} \mathbb{E} \left[ \int_0^T (G_s - G^n_s)^2 ds \right] = 0 \quad (2.10)$$

\textbf{Step 1:} Suppose that $G$ is continuous and bounded. For each positive integer $n$ define

$$G^n_t = \sum_{i=1}^{2^n-1} G_{i2^{-n}T} \cdot 1_{(i2^{-n}T,(i+1)2^{-n}T)}(t)$$

Clearly $G^n \in \mathcal{E}_0$ and we now show that $\forall t \in (0, T]$ and $\omega \in \Omega$, $\lim_{n \to \infty} G^n_t(\omega) = G_t(\omega)$. Let $\delta > 0$. By continuity of $G$, there exists $\eta > 0$ such that

$$\forall s \in (t - \eta, t + \eta) \subset (0, T], \quad |G_t(\omega) - G_s(\omega)| < \delta$$

Now choose any positive integer $n$ such that $2^{-n}T < \eta$. There exists $i \in \{0, 1, \ldots, 2^n - 1\}$ such that $t \in (i2^{-n}T, (i+1)2^{-n}T]$. Then $2^{-n}T < \eta \Rightarrow (i2^{-n}T, (i+1)2^{-n}T] \subset (t - \eta, t + \eta)$. In particular $i2^{-n}T \in (t - \eta, t + \eta)$ and so

$$|G_t(\omega) - G_{i2^{-n}T}(\omega)| < \delta$$

i.e. $|G_t(\omega) - G^n_t(\omega)| < \delta$
This shows that \( \lim_{n \to \infty} G^n_t(\omega) = G_t(\omega) \). This implies \( \lim_{n \to \infty} 1_{(0,T]} (G_t(\omega) - G^n_t(\omega))^2 = 0 \). By assumption, \( G \) is bounded so using the DCT we get

\[
0 = \lim_{n \to \infty} \int_{\mathbb{R}^+} 1_{(0,T]} (G_t(\omega) - G^n_t(\omega))^2 \, dt
= \lim_{n \to \infty} \int_{\mathbb{R}^+} 1_{[0,T]} (G_t(\omega) - G^n_t(\omega))^2 \, dt
= \lim_{n \to \infty} \int_{0}^{T} (G_t(\omega) - G^n_t(\omega))^2 \, dt
\]

Note that the second equality above comes from the fact that the integrands can only differ on \( \{0\} \), which is a set with Lebesgue measure zero. This means the integrands are a.e. equal and thus integrals are equal. Another application of the DCT shows that (2.10) holds.

**Step 2:** Suppose that \( G \) is progressively measurable and bounded. In light of step 1, it is enough to show existence of continuous, adapted and bounded processes \( (G^n)_{n \in \mathbb{N}} \) satisfying (2.10). Define for all \( \omega \in \Omega \)

\[
G^n_t(\omega) = n \int_{(t-(1/n) \vee 0, t \wedge T)} G_s(\omega) \, ds = n \int_{\mathbb{R}^+} 1_{[(t-(1/n) \vee 0, t \wedge T)]}(s) G_s(\omega) \, ds
\]

Let \( (t_m)_{m \in \mathbb{N}} \) be a sequence in \( \mathbb{R}^+ \) such that \( \lim_{m \to \infty} t_m = t \). Clearly \( 1_{[(t_m-(1/n) \vee 0, t_m \wedge T)]} \to 1_{[(t-(1/n) \vee 0, t \wedge T)]} \). Because \( G \) is bounded, it follows from the DCT that \( \lim_{m \to \infty} G^n_{t_m} \to G^n_t \) and so \( G^n \) is a continuous process. Clearly, the process is also adapted. To show boundedness, note that

\[
G^n_t \leq n \int_{(t-(1/n) \vee 0)}^{t \wedge T} \|G(\omega)\|_\infty \, ds
= n \lambda \left( [(t_m - 1/n) \vee 0, t_m \wedge T] \right) \|G(\omega)\|_\infty
\leq n \frac{1}{n} \|G(\omega)\|_\infty
< \infty
\]

21
Next note that \( G^n_t(\omega) = F_\omega(t^nT) - F_\omega((t-(1/n)\vee 0)) \) where \( F_\omega(x) = \int_{[0,x]} G(\omega)d\lambda \). This implies that

\[
\lim_{n\to\infty} G^n_t(\omega) = \frac{d}{dt} F_\omega(t) = G_t(\omega)
\]

for \( \lambda \)-almost all \( t \in [0,T] \) by the Fundamental theorem of calculus for Lebesgue integral. Again applying DCT, we get

\[
\lim_{n\to\infty} \int_0^T (G_s(\omega) - G^n_s(\omega))^2 ds = 0.
\]

Next consider

\[
\mathbb{E} \left[ \int_0^T (G_s - G^n_s)^2 ds \right] = \int_\Omega \left( \int_0^T (G_s(\omega) - G^n_s(\omega))^2 ds \right) \mathbb{P}(d\omega)
\]

Taking the limit \( n \to \infty \) and applying DCT again, we see that (2.10) holds.

**Step 3:** Suppose that \( G \in \mathcal{E} \). In light of steps 1 and 2 it suffices to show the existence of a sequence of bounded, progressively measurable processes \( (G^n)_{n \in \mathbb{N}} \) satisfying (2.10). The processes \( G^n_t = G_t 1_{\{|G_t|<n\}} \) fit the bill. Finally we may conclude that \( G \in \bar{\mathcal{E}}_0 \) and thus \( \mathcal{E} \subset \bar{\mathcal{E}}_0 \). \( \square \)

We can now extend the map \( I_{\infty} : \mathcal{E}_0 \to L^2(\mathbb{P}) \) to \( \bar{\mathcal{E}}_0 = \mathcal{E} \).

**Theorem 2.3.9.** There exists an almost-surely unique, continuous extension \( \tilde{I}_{\infty} \) of \( I_{\infty} \) to \( \bar{\mathcal{E}}_0 = \mathcal{E} \). For any \( G \in \mathcal{E} \), it is defined by

\[
\tilde{I}_{\infty}(G) \triangleq \lim_{n \to \infty} I_{\infty}(G^n)
\]

where \( (G^n)_{n \in \mathbb{N}} \) is any sequence in \( \mathcal{E}_0 \) such that \( \lim_{n \to \infty} \|G - G^n\|_\mathcal{E} = 0 \) and \( \lim_{n \to \infty} \|G - G^n\|_2 \) denotes \( L^2 \)-convergence.

**Proof.** We proved in theorem 2.3.5 that \( I_{\infty} \) is an isometry between \( \mathcal{E}_0 \) and \( L^2(\Omega, \mathcal{F}, \mathbb{P}) \), and isometries are clearly uniformly continuous mappings. Note that a seminorm can always be changed into norm by introducing the equivalence relation of almost-everywhere equality. So let us now regard \( \|\cdot\|_\mathcal{E} \) and \( \|\cdot\|_2 \) as norms on the respective equivalence classes.
Recall that \( L^2(P) \) is complete (Cauchy sequences in \( L^2 \) converge to some element of \( L^2 \)). Theorem B.1.1 then gives us the desired extension of \( I_\infty \).

Given any stopping time \( \tau \) and process \( G \), we define another process \( G^{(\tau)} \) by \( G_s^{(\tau)} \triangleq G_s1_{\{s \leq \tau\}}. \)

**Lemma 2.3.10.** If \( G \) is a progressively measurable process, \( G^{(\tau)} \) is also progressively measurable for any stopping time \( \tau \).

**Proof.** By definition of stopping time \( \{t \leq \tau\} \in \mathcal{F}_t \). So the process \( 1_{\{t \leq \tau\}} \) is adapted. This process is also left continuous and thus progressively measurable by lemma 2.3.7. It follows that the map from \( \Omega \times [0, t] \) to \( \mathbb{R} \) given by \( (\omega, s) \mapsto G_s(\omega)1_{\{s \leq \tau\}}(\omega) \) is \( \mathcal{F}_t \times \mathcal{B}([0, t]) \)-\( \mathcal{B}(\mathbb{R}) \) measurable.

**Theorem 2.3.11.** For every \( G \in \mathcal{E} \) the process \( \hat{I}(G) \) defined by \( \hat{I}_t(G) = \hat{I}_\infty(G^{(t)}) \) is a \( L^2 \)-bounded \( \mathcal{F} \)-martingale which has a continuous modification denoted by \( I^{(c)}(G) \).

**Proof.** Let \( H \in \mathcal{E}_0 \). For any \( t \in \mathbb{R}^+ \) it is easy to see that

\[
I_\infty((H)^{(t)}) = I_t(H) \tag{2.11}
\]

Let \( (G^n)_{n \in \mathbb{N}} \) be a sequence in \( \mathcal{E}_0 \) such that \( \lim_{n \to \infty} \|G^n - G\|_\mathcal{E} = 0 \). Consider for any \( t \in \mathbb{R}^+ \)

\[
\hat{I}_t(G^n) = \hat{I}_\infty((G^n)^{(t)}) = I_t(G^n) \tag{2.11}
\]

By (2.11)

\[
= \mathbb{E}[I_\infty(G^n)|\mathcal{F}_t] \tag{2.12}
\]

By theorem 2.3.4 (ii)
Next consider

\[\| (G^n)_{(t)} - G_{(t)} \|_{\mathcal{E}} = \mathbb{E} \left[ \int_0^\infty 1_{\{s \leq t\}} (G^n_s - G_s)^2 \, ds \right] \leq \mathbb{E} \left[ \int_0^\infty (G^n_s - G_s)^2 \, ds \right] = \|G^n - G\|_{\mathcal{E}} \xrightarrow{n \to \infty} 0 \quad (2.13)\]

Furthermore,

\[\hat{I}_t(G) \triangleq \tilde{I}_\infty (G_{(t)}) \]
\[\triangleq \lim_{n \to \infty} I_\infty \left( (G^n)_{(t)} \right) \quad \text{by (2.13)} \]
\[= \lim_{n \to \infty} \mathbb{E} [I_\infty(G^n) | \mathcal{F}_t] \quad \text{by (2.12)} \]
\[= \mathbb{E} \left[ \lim_{n \to \infty} I_\infty(G^n) | \mathcal{F}_t \right] \quad \text{by theorem B.3.1} \]
\[= \mathbb{E} [\hat{I}_\infty(G) | \mathcal{F}_t] \]
\[= \mathbb{E} [\tilde{I}_\infty(G) | \mathcal{F}_t] \quad \text{since } G^{(\infty)} = G \quad (2.14) \]

Now choose any \( u \in [t, \infty) \) and consider

\[\mathbb{E} [\hat{I}_u(G) | \mathcal{F}_t] = \mathbb{E} [\mathbb{E} [\hat{I}_\infty(G) | \mathcal{F}_u] | \mathcal{F}_t] \quad \text{by (2.14)} \]
\[= \mathbb{E} [\hat{I}_\infty(G) | \mathcal{F}_t] \quad \text{by tower property} \]
\[= \hat{I}_t(G) \quad \text{again by (2.14)} \]

This proves that \( \hat{I}(G) \) is an \( \mathbb{F} \)-martingale. Clearly, it is also \( L^2 \)-bounded. Next we have

\[\lim_{n \to \infty} I_t(G^n) = \lim_{n \to \infty} (G^n)_{(t)} \quad \text{by (2.11)} \]
\[\triangleq \tilde{I}_\infty (G_{(t)}) \quad \text{from (2.13)} \]
\[\triangleq \hat{I}_t(G) \quad (2.15) \]
For every \( n \in \mathbb{N} \), \( I_n(G^n) \) is continuous by thm 2.3.4 (i). Applying B.3.2 in conjunction with (2.15) it follows that \( \hat{I}(G) \) has a continuous modification \( I^{(c)}(G) \).

**Definition 2.3.12. (Itô integral).** For \( G \in \mathcal{E} \) the **Itô integral on** \( \mathbb{R}^+ \) is defined as

\[
\int_0^\infty G_s dW_s \triangleq \hat{I}_{\infty}(G)
\]

For \( s, t \in \mathbb{R}^+ \), \( s \leq t \) we define the **Itô integral on** \([s, t]\) by

\[
\int_s^t G_r dW_r \triangleq I^{(c)}_t(G) - I^{(c)}_s(G)
\]

The final step in our construction of the Itô integral is to relax the integrability condition \( \mathbb{E} \left[ \int_0^\infty G_s^2 ds \right] < \infty \). This can be done using a localisation technique by stopping times. Given a stopping time \( \tau \) and any \( G \in \mathcal{E} \) define

\[
\int_0^\tau G_s dW_s \triangleq \int_0^\infty G^{(\tau)}_s dW_s
\]

**Lemma 2.3.13.** Let \( G \in \mathcal{E} \) and \( \sigma, \tau \) be a stopping times such that \( \sigma \leq \tau \) a.s. Then

\[
1_{\{t \leq \sigma\}} \int_0^{\tau \wedge t} G_s dW_s = 1_{\{t \leq \sigma\}} \int_0^t G_s dW_s
\]

(2.17)

where \( \int_0^t G_s dW_s \triangleq I^{(c)}_t(G) \) from definition 2.3.12.

**Proof.** Note that \( t \) and \( \tau \wedge t \) are stopping times. This means that the definition of (9.1) coincides with definition 2.3.12 for \( \int_0^t G_s dW_s \). If \( G \in \mathcal{E}_0 \) then (9.3) clearly holds by
definition of $I_\infty$. Now suppose that $G \in \mathcal{E}$ and let $(G^p)_{p \in \mathbb{N}}$ be a sequence in $\mathcal{E}_0$ such that $G^p \xrightarrow{p \to \infty} G$ in $L^2$. Then $(G^p)^{(t \wedge \tau)} \xrightarrow{p \to \infty} G^{(t \wedge \tau)}$ in $L^2$ and

$$\tilde{I}_\infty (G^{(t \wedge \tau)}) = \lim_{p \to \infty} I_\infty ((G^p)^{(t \wedge \tau)})$$

$$\Rightarrow 1_{\{t \leq \sigma\}} \tilde{I}_\infty (G^{(t \wedge \tau)}) = 1_{\{t \leq \sigma\}} \lim_{p \to \infty} I_\infty ((G^p)^{(t \wedge \tau)})$$

$$= \lim_{p \to \infty} 1_{\{t \leq \sigma\}} I_\infty ((G^p)^{(t \wedge \tau)})$$

$$= \lim_{p \to \infty} 1_{\{t \leq \sigma\}} I_\infty ((G^p)^{(t)})$$

$$= 1_{\{t \leq \sigma\}} \tilde{I}_\infty (G^{(t)})$$

This proves (9.3) for $G \in \mathcal{E}$. \square 

**Definition 2.3.14.** Let $\mathcal{E}_{loc}$ denote the space of progressively measurable processes $G$ satisfying

$$\int_0^T G_s^2 ds < \infty \text{ a.s. for all } T > 0$$

**Theorem 2.3.15.** For every $G \in \mathcal{E}_{loc}$, there exists a sequence $(\tau_n)_{n \in \mathbb{N}}$ of stopping times such that $\tau_n \uparrow \infty$ and $G^{(\tau_n)} \in \mathcal{E}$ for all $n \in \mathbb{N}$.

**Proof.** For each $n \in \mathbb{N}$ define

$$\tau_n \triangleq \inf \left\{ t \geq 0 : \int_0^t G_s^2 ds \geq n \right\}$$
By definition of $\mathcal{E}_{loc}$, $\tau_n \uparrow \infty$. Now consider

$$
\mathbb{E} \left[ \int_0^\infty (G_{s}^{(\tau_n)})^2 \, ds \right] = \mathbb{E} \left[ \int_0^\infty (G_s 1_{s \leq \tau_n})^2 \, ds \right] \\
= \mathbb{E} \left[ \int_0^{\tau_n} G_s^2 \, ds \right] \\
\leq n \quad \text{by definition of } \tau_n \\
< \infty
$$

\[\square\]

**Theorem 2.3.16.** Following from the previous theorem, let $m,n \in \mathbb{N}$, $m \leq n$. Then

$$1_{\{t \leq \tau_m\}} \int_0^t G_s^{(\tau_n)} \, dW_s = 1_{\{t \leq \tau_m\}} \int_0^{\tau_n} G_s \, dW_s \quad (2.18)$$

**Proof.** First note that the integrands on both sides of (2.18) belong to $\mathcal{E}$ by lemma 2.3.10. Now consider

$$1_{\{t \leq \tau_m\}} \int_0^t G_s^{(\tau_n)} \, dW_s \triangleq 1_{\{t \leq \tau_m\}} I_t^{(c)} (G^{(\tau_n)})$$

$$\triangleq 1_{\{t \leq \tau_m\}} \tilde{I}_t (G^{(\tau_n)})^{(t)}$$

$$= 1_{\{t \leq \tau_m\}} \int_0^{\tau_n} G_s 1_{\{s \leq t \wedge \tau_m\}} \, dW_s$$

$$= 1_{\{t \leq \tau_m\}} \int_0^{\tau_n} G_s 1_{\{s \leq t\}} \, dW_s$$

where the second last equality follows from lemma 2.3.13. Similarly we can show that

$$1_{\{t \leq \tau_m\}} \int_0^t G_s \, dW_s = 1_{\{t \leq \tau_m\}} \int_0^{\tau_n} G_s 1_{\{s \leq t \wedge \tau_m\}} \, dW_s$$

$$= 1_{\{t \leq \tau_m\}} \int_0^{\tau_n} G_s 1_{\{s \leq t\}} \, dW_s$$

and the result follows. \[\square\]
**Definition 2.3.17.** Let $G \in \mathcal{E}_{loc}$ and let $(\tau_n)_{n \in \mathbb{N}}$ be the sequence of stopping times defined in theorem 2.3.15. The Itô integral of $G$ over $[0, t]$ is defined as the process $\int_0^t G_s dW_s$ satisfying

$$1_{\{t \leq \tau_m\}} \int_0^t G_s dW_s = 1_{\{t \leq \tau_m\}} \int_0^t G_s^{(\tau_m)} dW_s$$

for all $m \in \mathbb{N}$.

Note from theorem 2.3.16 that the definition is well-defined and does not depend on the choice of stopping time $\tau_m$.

Note that the definition above isn’t very useful when trying to actually evaluate a stochastic integral. The famous Itô formula (also known as Itô’s lemma) is the basic tool to evaluate these integrals. We state a version of the result which is sufficient for our needs.

**Theorem 2.3.18.** (Itô formula for 1 dimensional Brownian motion) Let $W_t$ be a standard Brownian motion and $F : [0, \infty) \times \mathbb{R} \to \mathbb{R}$, $F(t, x)$ be a twice continuously differentiable function. Then

$$F(T, W_T) - F(0, W_0) = \int_0^T \frac{\partial}{\partial x} F(s, W_s) dW_s + \int_0^T \left[ \frac{\partial}{\partial t} F(s, W_s) + \frac{1}{2} \frac{\partial^2}{\partial x^2} F(s, W_s) \right] ds$$

**Proof.** For a proof of a more general formulation see for instance [25, pg. 149].

A simple application of the Itô formula gives us the following result which is needed later.

**Corollary 2.3.19.** Let $W_t$ be a standard Brownian motion. Then $\int_0^T W_t dW_t = \frac{1}{2} (W_T^2 - T)$.

**Proof.** Apply the Ito formula with $F(t, x) = \frac{1}{2} x^2$. 

28
2.4 Stochastic differential equations

Having constructed the Itô integral in the previous section we now consider stochastic differential equations (SDE) of the form

$$
X_0 = \xi, \quad dX_t = \sigma(t, X_t) dW_t + b(t, X_t) dt
$$

(2.19)

Where $W$ is a 1-dimensional Brownian motion, $\xi$ is a random variable independent of $W$ and the maps $(t, x) \mapsto \sigma(t, x)$ and $(t, x) \mapsto b(t, x)$ are measurable. The notation in (2.19) is a standard shorthand for

$$
X_t = \xi + \int_0^t \sigma(s, X_s) dW_s + \int_0^t b(s, X_s) ds, \ P - \text{a.s. for all } t \geq 0
$$

(2.20)

and we say that a continuous and adapted stochastic process $X$ is a solution of the SDE (2.19) if it satisfies (2.20). But to which filtration is $X$ adapted? When the filtration is generated by $W$ and $\xi$, we are led to what is called a strong solution of the SDE.

**Definition 2.4.1.** A strong solution of the SDE (2.19) is a stochastic process $X$ such that

(i) $X$ is adapted to the filtration $\mathbb{F} = (\mathcal{F}_t)_{t \geq 0}$, where

- $\mathcal{G}_t = \sigma(\xi, W_s; s \in [0, t])$,
- $\mathcal{G}_\infty \triangleq \sigma(\bigcup_{t \geq 0} \mathcal{G}_t)$,
- $\mathcal{N} = \{ N \subseteq \Omega : \exists G \in \mathcal{G}_\infty, N \subseteq G \text{ and } P(G) = 0 \}$ and
- $\mathcal{F}_t = \sigma(\mathcal{G}_t \cup \mathcal{N})$.

(ii) $P[X_0 = \xi] = 1$,

(iii) $P \left[ \int_0^t [\sigma^2(s, X_s) + |b(s, X_s)|] ds < \infty \right] = 1$, for all $0 \leq t < \infty$ and

(iv) $X$ satisfies (2.20).
Theorem 2.4.2. Following from the previous definition, suppose that there exists a $K > 0$ such that for all $x, x' \in \mathbb{R}$ and $t \geq 0$,

$$|\sigma(t, x) - \sigma(t, x')| + |b(t, x) - b(t, x')| \leq K |x - x'|$$

and that

$$(\sigma(t, x))^2 + (b(t, x))^2 \leq K^2 (1 + x^2), \text{ for all } x \in \mathbb{R}, t \geq 0$$

Then for every $X_0 = x \in \mathbb{R}$, there exists a unique strong solution $X$ of the SDE (2.19). Furthermore, this solution is a Markov process.

Proof. For existence and uniqueness of a strong solution refer for example [25, pg. 289]. For a proof of the Markov property refer to [47, pg. 111].

2.4.1 Geometric Brownian motion

A stochastic process $S_t$ is called Geometric Brownian motion if it is a solution of the following SDE

$$dS_t = \mu S_t dt + \sigma S_t dW_t,$$

$$S_0 = s$$

where $s, \mu, \sigma \geq 0$ are constants and $W_t$ is a standard Brownian motion.

Theorem 2.4.3. The SDE (2.21) has a unique, strong solution given by

$$S_t = s \exp \left\{ \sigma W_t + \left( \mu - \frac{\sigma^2}{2} \right) t \right\}.$$
Proof. Note that the SDE (2.21) can be expressed as $dS_t = b(t, S_t)dt + v(t, S_t)dW_t$ where $b(t, x) = \mu x$ and $v(t, x) = \sigma x$. Let $x, x' \in \mathbb{R}$, $t \geq 0$ and consider

$$|v(t, x) - v(t, x')| + |b(t, x) - b(t, x')| = \sigma |x - x'| + \mu |x - x'| \leq K |x - x'|$$

for any $K > \min \{ (\sigma + \mu), 0 \}$. Also consider

$$[v(t, x)]^2 + [b(t, x)]^2 = (\sigma^2 + x^2)x^2 \leq K(1 + x^2)$$

for any $K > \sigma^2 + \mu^2$. So by theorem 2.4.2 the SDE (2.21) has a unique strong solution which is a Markov process. We now show that $S_t$ given by equation (2.22) is the strong solution. First define $F : \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ by $F(t, x) = S_0 \exp \{ \sigma x + (\mu - \sigma^2/2) t \}$. Then using Itô’s formula (theorem 2.3.18),

$$S_t - S_0 = F(t, W_t) - F(0, W_0)$$

$$= \int_0^t \frac{\partial}{\partial x} F(s, W_s) dW_s + \int_0^t \left[ \frac{\partial}{\partial t} F(s, W_s) + \frac{1}{2} \frac{\partial^2}{\partial x^2} F(s, W_s) \right] ds$$

$$= \sigma \int_0^t F(s, W_s) dW_s + \int_0^t \left[ \left( \mu - \frac{\sigma^2}{2} \right) F(s, W_s) + \frac{\sigma^2}{2} F(s, W_s) \right] ds$$

$$= \int_0^t \mu S_s ds + \int_0^t \sigma S_s dW_s$$

Corollary 2.4.4. The mean and variance of GBM $S_t$ is given by

$$\mathbb{E}[S_t] = S_0 e^{\mu t},$$

$$\text{Var} (S_t) = S_0^2 e^{2\mu t} \left( e^{\sigma^2 t} - 1 \right).$$
Proof. Note that $W_t \sim N(0, t)$ and so its moment generating function is $\phi(u) = \exp \left\{ \frac{tu^2}{2} \right\}$. Then by the previous theorem,

$$
E[S_t] = S_0 \exp \left\{ \left( \mu - \frac{\sigma^2}{2} \right) t \right\} E[e^{\sigma W_t}]
$$

$$
= S_0 \exp \left\{ \left( \mu - \frac{\sigma^2}{2} \right) t \right\} \phi(\sigma)
$$

$$
= S_0 e^{\mu t}
$$

Next consider

$$
E[S_t^2] = S_0^2 \exp \left\{ 2 \left( \mu - \frac{\sigma^2}{2} \right) t \right\} E[e^{2\sigma W_t}]
$$

$$
= S_0^2 \exp \left\{ 2\mu t - \sigma^2 t \right\} \phi(2\sigma)
$$

$$
= S_0^2 \exp \left\{ 2\mu t + \sigma^2 t \right\}
$$

Finally

$$
\text{Var}(S_t) = E[S_t^2] - (E[S_t])^2
$$

$$
= S_0^2 e^{2\mu t} \left( e^{\sigma^2 t} - 1 \right).
$$

2.4.2 The Ornstein-Uhlenbeck process

The Ornstein-Uhlenbeck process $X_t$ is a solution of the following SDE

$$
dX_t = \theta (\mu - X_t) \, dt + \sigma dW_t,
$$

$$
X_0 = x
$$

(2.23)

where $x, \mu \in \mathbb{R}$, $\theta > 0$, $\sigma > 0$ are parameters and $W_t$ denotes standard Brownian motion.

The following theorem gives the most important properties of the Ornstein-Uhlenbeck process.
Theorem 2.4.5. The SDE (2.23) has a unique strong solution

\[ X_t = xe^{-\theta t} + \mu \left( 1 - e^{-\theta t} \right) + \int_0^t \sigma e^{\theta (s-t)} dW_s. \]  

(2.24)

This solution is a Gaussian-Markov process with mean and covariance functions

\[ \mathbb{E}[X_t] = xe^{-\theta t} + \mu \left( 1 - e^{-\theta t} \right) \]
\[ \text{cov}(X_s, X_t) = \frac{\sigma^2}{2\theta} e^{-\theta(s+t)} \left( e^{2\theta \min(s,t)} - 1 \right) \]

Proof. It is routine to check that the SDE (2.23) satisfies the conditions of theorem 2.4.2 and thus it has a unique strong solution which is also a Markov process. Next consider

\[ x + \int_0^t \sigma dW_s + \int_0^t \theta (\mu - X_s) ds \]
\[ = x + \sigma W_t + \theta \mu t - \int_0^t \theta \left[ xe^{-\theta s} + \mu \left( 1 - e^{-\theta s} \right) + \int_0^s \sigma e^{\theta (u-s)} dW_u \right] ds \]
\[ = xe^{-\theta t} + \mu \left( 1 - e^{-\theta t} \right) + \sigma W_t - \int_0^t \int_0^s \sigma \theta e^{\theta (u-s)} dW_u ds \]
\[ = xe^{-\theta t} + \mu \left( 1 - e^{-\theta t} \right) + \sigma W_t - \int_0^t \sigma \left[ \int_u^t \theta e^{\theta (u-s)} ds \right] dW_u \]
\[ = xe^{-\theta t} + \mu \left( 1 - e^{-\theta t} \right) + \sigma W_t - \int_0^t \sigma (1 - e^{\theta (u-t)}) dW_u \]
\[ = xe^{-\theta t} + \mu \left( 1 - e^{-\theta t} \right) + \int_0^t \sigma e^{\theta (s-t)} dW_s \]
\[ = X_t \]

where the inexchange of order of integration underlined above follows from the Fubini theorem for stochastic integrals (for a proof of this result see [50, pg. 210]).

The following theorem is fundamental to our work in chapter 4.

Theorem 2.4.6. Let \( B_t \) be standard Brownian motion and \( \xi_t \) be a standard Ornstein-Uhlenbeck process with parameter \( \lambda \) i.e. \( \xi_t \) satisfies the SDE \( d\xi_t = -\lambda \xi_t dt + dB_t, \xi_0 = 0. \)
Let \( \mu_{B,T} \) and \( \mu_{\xi,T} \) denote the measures generated by the processes \( B_t \) and \( \xi_t \) up to time \( T \), respectively. Then \( \mu_{B,T} \) and \( \mu_{\xi,T} \) are equivalent measures and the Radon-Nikodym derivative \( \frac{\mu_{\xi,T}}{\mu_{B,T}} \) is given by

\[
\frac{\mu_{\xi,T}}{\mu_{B,T}} = \exp \left\{ -\lambda \int_0^T B_t dB_t - \frac{\lambda^2}{2} \int_0^T B_t^2 dt \right\}
\]

Proof. See Liptser & Shiryaev [35, pg. 5] for the proof of a more general result. \( \square \)
Part I

Options with volume-weighted average price (VWAP) as underlying
Chapter 3

Introduction

3.1 Overview

The calculation of a volume weighted average price (VWAP) occurs frequently in finance. It is used as a benchmark price by market participants and can be regarded as an estimate for the price that a passive trader will pay to purchase securities in a market. The VWAP is commonly used in brokerage houses to assess the performance of a trader and has applications in algorithmic trading (see [9, volume 4]). The VWAP even appears in Australian taxation law as part of determining the price of share buy-backs in publicly listed companies [61].

Suppose that in a given time interval (day, week, etc.) there are $N$ transactions involving shares of a particular company. Let $S_i$ and $U_i$ denote the price and trading volume respectively of transaction $i \in \{1 \ldots N\}$. We define the VWAP as

$$\text{VWAP} = \frac{\sum_{i=1}^{N} S_i U_i}{\sum_{i=1}^{N} U_i}$$

(3.1)

3.2 Existing literature

3.2.1 Volume weighted average price (VWAP)

The buyers and sellers within a financial market incur costs on their trading, and these costs can be broadly separated into two components. The first is commission costs. This
is the fee that a buyer or seller pays to their broker to execute a trade on their behalf. This cost is simple to analyse and account for.

The second cost is called market impact, which is defined as the deviation of the transaction price from some market reference price. This reference price is thought of as the price that is attainable by a “passive” trader. The VWAP is a very popular reference price with which to measure market impact and was first introduced by Berkowitz et. al. [5].

Almost all the existing literature on VWAP focuses on strategies and algorithms to execute orders as close as possible to the VWAP price (see for e.g. [6], [24], [28] and [38]).

3.2.2 Asian options

Our work on VWAP options is a generalisation of the Asian option about which we now briefly survey. An Asian option is an option with a payoff that is a function of an average of the underlying price over a pre-set interval of time. The most common Asian option uses the arithmetic mean of the underlying price in the payoff function. The pricing of Asian options is of great interest to both practitioners and academic researchers and the problem is difficult; under the Black-Scholes framework no closed-form solution has yet been discovered. This is due to the surprising fact that there is no recognisable distribution for the sum of two lognormally-distributed random variables.

The literature contains a range of different techniques to address this problem and we give a brief survey here.

Analytical approximations

Since the distribution of the arithmetic average price is unknown under the Black-Scholes model, attempts have been made to approximate the process. The moment matching approach is a method of approximating a complicated stochastic process with a simpler process, by imposing a condition where some number of moments of both processes are equal. In Turnbull & Wakeman [60], the arithmetic average was approximated by the lognormal distribution via moment matching. A Black-Scholes-type pricing formula can then be derived. See also Levy [33]. Building on the work of Turnbull-Wakeman, Milevsky & Posner [42] match the first four moments to a reciprocal gamma distribution.
Pricing bounds

Another technique is to find lower and upper bounds on the price of Asian options. Curran [10] utilised conditional expectation and Jensen’s inequality to obtain an accurate lower bound on the price of the continuously monitored Asian option under the Black-Scholes framework. This method required conditioning on a specific form of Gaussian random variable. Rogers & Shi [52] formulated a lower bound by similar techniques but has the benefit of conditioning by an arbitrary Gaussian random variable. By also considering the error of their lower bound, Rogers & Shi provide a conservative upper bound as well. In the unpublished paper of Thompson [59] a sharper upper bound was derived. A further sharpening of the Thompson upper bound and overall summary of known results in this area up to 2006 appear in Lord [37]. More recently the papers of Fusai and Meucci [17] and Lemmens et al [32] give bounds on discretely monitored Asian options under the Levy processes framework.

Numerical methods

Monte Carlo simulation is a flexible method to price options and is relatively easy to implement. It is well known that the variance of the crude Monte Carlo estimate is of order $n^{-1/2}$ and variance reduction techniques can be applied to speed up convergence. The paper of Kemna & Vorst [26] use a geometric average Asian option as a control variate to estimate the arithmetic average Asian option, resulting in a large reduction in variance.

Another method that can be used to value Asian options involve the use of binomial and trinomial tree lattices. The standard implementation of this method is very inefficient due to exponential growth in the number of possible values for the arithmetic average. An interpolation technique described in Hull & White [20] curbs the growth to give a more tractable method. Improvements to the method are described in Klassen [27] and Hsu & Lyuu [19].

There are a number of other numerical methods that can treat Asian options; Boyle & Potapchik [7] give an in-depth survey and also discuss numerical implementations.
3.2.3 Options on VWAP

Currently there exists only one published paper on the pricing of VWAP options by Stace [57]. The author used a moment matching approach to find a lognormal approximation for the call option via approximation of VWAP first and second moments. The following approximations for computing the moments was used:

\[
\mathbb{E}\left(\frac{Y}{Z}\right) \approx \frac{\mathbb{E}(Y)}{\mathbb{E}(Z)} - \frac{\text{Cov}(Y, Z)}{(\mathbb{E}(Z))^2} + \frac{\mathbb{E}(Y)}{(\mathbb{E}(Z))^3} \text{Var}(Z),
\]

\[
\text{Var}\left(\frac{Y}{Z}\right) \approx \left(\frac{\mathbb{E}(Y)}{\mathbb{E}(Z)}\right)^2 \left(\frac{\text{Var}(Y)}{(\mathbb{E}(Y))^2} + \frac{\text{Var}(Z)}{(\mathbb{E}(Z))^2} - 2 \frac{\text{Cov}(Y, Z)}{\mathbb{E}(Y)\mathbb{E}(Z)}\right).
\]

This approximation is based on a truncated Taylor series expansion, see [44]. The author then provided a continuous time setting for VWAP with a geometric Brownian motion for \( S_t \) and CIR model for \( U_t \). It was shown that approximations for the first and second moments can be found by solving a very large system (nineteen!) of ordinary differential equations. Calculating the VWAP moments is not a simple task because it involves computing the moments of a ratio of two random variables \( \frac{Y}{Z} \).

3.3 Summary of the contributions of part I

Our contribution consists in the derivation of exact analytical formulas for the first and second moments of a continuous-time VWAP process under the assumption that the volume process is modelled by the square of an Ornstein-Uhlenbeck process (which is very close by nature to a CIR process) and the asset price is a geometric Levy process. In contrast to [57] our setup allows for dependent price and volume processes. We provide a working numerical implementation under the assumption that the asset price follows geometric Brownian motion.
The VWAP model and calculation of its moments

The VWAP arose from the study of costs of market trading. The cost to make a market trade can be separated into two components. The first is commission costs, which is the fee that is paid to the broker to execute a trade. This is a simple and pre-determined cost requiring no further study. The second cost – which is of great interest to both researchers and academics – is called market impact. We first give an illustration. Suppose that company ABC’s shares are traded at a stock exchange and that at a certain point in time the shares are for sale at the prices and quantities specified in table 4.1:

<table>
<thead>
<tr>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.00</td>
<td>5000</td>
</tr>
<tr>
<td>10.01</td>
<td>6500</td>
</tr>
<tr>
<td>10.02</td>
<td>3000</td>
</tr>
<tr>
<td>10.10</td>
<td>4000</td>
</tr>
<tr>
<td>10.15</td>
<td>10000</td>
</tr>
<tr>
<td>10.20</td>
<td>22000</td>
</tr>
</tbody>
</table>

Table 4.1: Price and quantity for ABC shares

If given these quotes a trader buys 10000 shares he will purchase 5000 at $10.00 and 5000 at $10.01 for an average price of

\[
\frac{10 \times 5000 + 10.01 \times 5000}{10000} = \$10.005
\]

On the other hand, if this trader were to buy 30,000 shares then they will be acquired at an average price of $10.0775. So buying a larger quantity of shares leads to a
higher average price; we say that this trade has a larger *market impact* than the previous purchase of 10,000 shares.

So market impact is a sort of measure of the extent to which price moves against a buyer or seller when making a trade. A more formal way to describe market impact is to define it as the deviation of the transaction price from some reference price which estimates the price that is attainable by any randomly selected trader in any relevant trading interval. This is the definition used by Berkowitz *et al.* [5] and moreover they proposed the VWAP to be this reference price.

Suppose that in a given time interval (day, week, etc.) there are \( N \) transactions involving shares of a particular company. Let \( S_i \) and \( U_i \) denote the price and trading volume respectively of transaction \( i \in \{1 \ldots N\} \). There are a number of ways to define the VWAP (see [38]) but the standard definition is given by:

\[
\text{VWAP} = \frac{\sum_{i=1}^{N} S_i U_i}{\sum_{i=1}^{N} U_i}
\]

So now we can define market impact as

\[
\text{market impact} = \begin{cases} 
\text{VWAP} - \text{buy price}, & \text{if transaction is a buy order} \\
\text{sell price} - \text{VWAP}, & \text{if transaction is a sell order}
\end{cases}
\]

The VWAP is now a popular benchmark price in financial markets. Institutional investors are increasingly looking to buy and sell at the VWAP and broker performance is frequently measured against the VWAP benchmark [9]. The VWAP also appears in Australian taxation law as part of determining the price of share buy-backs in publicly listed companies [61].

4.1 The volume process

We assume the usual framework of a probability space \( (\Omega, \mathcal{F}, \mathbb{P}) \) equipped with a filtration \( (\mathcal{F}_t)_{t \geq 0} \) satisfying the usual conditions. Let \( S_t \) denote the asset price at time \( t \) with known
mean and covariance functions. Let $U_t$ denote the volume (quantity) of assets that are traded at time $t$. We define the continuous-time VWAP process by

$$A_t = \frac{\int_0^t S_s U_s ds}{\int_0^t U_s ds}$$  \hspace{1cm} (4.1)$$

In this thesis, we model dependence of $S_t$ and $U_t$ by the relation

$$\log \left( \frac{S_t}{S_0} \right) = \rho B_t + \tilde{L}_t$$  \hspace{1cm} (4.2)$$

where $B_t$ and $\tilde{L}_t$ are independent Levy processes. Furthermore, $B_t$ will be the driver for the volume process $U_t$.

In this thesis we model the volume process $U_t$ is by

$$U_t = X_t^2 + \delta$$  \hspace{1cm} (4.3)$$

where $dX_t = \lambda(a - X_t)dt + vdB_t$, $X_0 = a$, $\delta \geq 0$ and $B_t$ denotes standard Brownian motion. $X_t$ is the Ornstein-Uhlenbeck process with parameters $a$ and $\lambda$. $X_t$ has a well known representation,

$$X_t = a + v\xi_t$$  \hspace{1cm} (4.4)$$

where $\xi_t$ is a standard Ornstein-Uhlenbeck process satisfying the SDE

$$d\xi_t = -\lambda\xi_t dt + dB_t, \ \xi_0 = 0.$$  \hspace{1cm} (4.5)$$

There are two reasons for our particular choice of $U_t$. Firstly, we use the Ornstein-Uhlenbeck process for mean-reverting behaviour which is suitable for modelling volume. During special periods such as corporate news releases, the new information revealed to the market leads to a flurry of buy and sell orders as market participants react to the event. After a period of time, trading activity settles back to some mean level. Secondly,
the Ornstein-Uhlenbeck process can take on negative values, hence we square the process to ensure non-negativity of the volume of transactions.

4.2 Calculation of VWAP moments

In this section we show how the first two VWAP moments \( \mathbb{E}[A_T] \) and \( \mathbb{E}[A_T^2] \) can be computed. The method depends on the determination of certain Laplace transforms. We begin with the following important lemma.

**Lemma 4.2.1.** Let \( X \) be a positive random variable. Then for any \( k > 0 \)

\[
\frac{1}{X^k} = \frac{1}{\Gamma(k)} \int_0^\infty q^{k-1}e^{-qX} dq
\]

**Proof.** Consider

\[
X^k \int_0^\infty q^{k-1}e^{-qX} dq = X \int_0^\infty (Xq)^{k-1}e^{-qX} dq
\]

\[
= \int_0^\infty r^{k-1}e^{-r} dr \quad \text{where } r = qX
\]

\[
= \Gamma(k)
\]

The result then follows by division. \( \square \)

Let \( V_T = \int_0^T U_t dt. \) The next two theorems give formulas for the first and second VWAP moments. The proof of existence of these moments is deferred to section 4.4.

**Theorem 4.2.2.** (VWAP first moment)

\[
\mathbb{E}[A_T] = S_0 \int_0^T \left( \mathbb{E}\left[e^{L_t} \int_0^\infty \frac{\partial}{\partial z} \Phi_1(\rho, z, q)\right] dq\right) dt
\]

where \( \Phi_1(\rho, z, q) = \mathbb{E}[\exp\{\rho B_t + zU_t - qV_T\}] \).
Proof.

\[
E [A_T] = E \left[ \frac{\int_0^T S_t U_t dt}{V_T} \right]
\]

\[
= S_0 \int_0^T E \left[ \exp \left\{ \rho B_t + \tilde{L}_t \right\} \frac{U_t}{V_T} \right] dt \quad \text{(by Fubini theorem)}
\]

\[
= S_0 \int_0^T E \left[ e^{L_t} \right] E \left[ \frac{e^{\rho B_t} U_t}{V_T} \right] dt \quad \text{($B_t$ and $\tilde{L}_t$ independent)}
\]

Next we apply lemma 4.2.1 to compute

\[
E \left[ \frac{e^{\rho B_t} U_t}{V_T} \right] = E \left[ e^{\rho B_t} U_t \int_0^\infty e^{-qV_T} dq \right]
\]

\[
= E \left[ \int_0^\infty U_t e^{\rho B_t - qV_T} dq \right]
\]

\[
= \int_0^\infty E \left[ U_t e^{\rho B_t - qV_T} \right] dq \quad \text{(Fubini theorem)}
\]

\[
= \int_0^\infty E \left[ \frac{\partial}{\partial z} (e^{\rho B_t + zU_t - qV_T}) \right]_{z=0} dq
\]

\[
= \int_0^\infty \frac{\partial}{\partial z} \Phi_1(\rho, z, q)_{z=0} dq
\]

\[\Box\]

**Theorem 4.2.3.** (VWAP second moment)

\[E(|A_T^2|) < \infty \text{ and} \]

\[E \left( |A_T|^2 \right) = S_0^2 \int_0^T \int_0^T E \left( e^{L_t + L_s} \right) \left( \int_0^\infty q \frac{\partial^2}{\partial z_1 \partial z_2} \Phi_2(\rho, z, r, q)_{z_1=0, z_2=0} dq \right) dt ds\]

where \( \Phi_2(\rho, z, r, q) = E(\exp\{\rho(B_t + B_s) + zU_t + rU_s - qV_T\}), \ s \leq t.\)
Proof. Consider

\[ E(A_T^2) = E \left( \frac{\int_0^T S_t U_t dt \int_0^T S_s U_s ds}{V_T^2} \right) \]

\[ = E \left( \frac{\int_0^T \int_0^T S_t S_s U_t U_s dt ds}{V_T^2} \right) \]

\[ = E \left( \int_0^T \int_0^T \exp \left\{ \tilde{L}_t + \tilde{L}_s + \rho(B_t + B_s) \right\} \frac{U_t U_s}{V_T^2} dt ds \right) \]

\[ = \int_0^T \int_0^T E \left( \exp \left\{ \tilde{L}_t + \tilde{L}_s \right\} \right) E \left( \frac{\exp \left\{ \rho(B_t + B_s) \right\} U_t U_s}{V_T^2} \right) dt ds \]

Next note from lemma 1 that

\[ \frac{1}{V_T^2} = \int_0^\infty q e^{-qV_T} dq \]

And so

\[ E \left( \frac{\exp \left\{ \rho(B_t + B_s) \right\} U_t U_s}{V_T^2} \right) = E \left( U_t U_s \exp \left\{ \rho(B_t + B_s) \right\} \int_0^\infty q e^{-qV_T} dq \right) \]

\[ = \int_0^\infty q E \left( U_t U_s \exp \left\{ \rho(B_t + B_s) - qV_T \right\} \right) dq \]

\[ = \int_0^\infty q \frac{\partial^2}{\partial z \partial r} \Phi_2(\rho, z, r, q)|_{z=0, r=0} dq \]

The two previous theorems would of course be useless if we don’t know \( \Phi_1(\rho, z, q) \) and \( \Phi_2(\rho, z, r, q) \). Based on a repeated application of theorem 2.4.6 and some help from a symbolic algebra package, we are able to find these Laplace transforms.

4.3 Derivation of the Laplace transforms

In this section we derive an exact formula for the Laplace transform \( \Phi_1(\rho, z, q) \) specified in theorem 4.2.2. This is required for the calculation of the first VWAP moment. The derivation of \( \Phi_2(\rho, z, r, q) \) is very similar to that of \( \Phi_1 \) and is treated in theorem
Lemma 4.3.1. The Laplace transform $\Phi_1(\rho, z, q)$ may be expressed as

$$\Phi_1(\rho, z, q) = c^{(1)} E \left( e^{\rho e^C} \right)$$

where

$$c^{(1)} = \exp \left\{ (\delta + a^2)z - (\delta + a^2)Tq + \frac{(\lambda - \kappa)T}{2} \right\},$$

$$\kappa = \sqrt{\lambda^2 + 2qv^2},$$

$$\eta_t = (2zav + \rho)Y_t + zv^2Y^2_t + (\lambda\rho - 2qv) \int_0^t Y_s ds,$$

$$\zeta = \frac{(\kappa - \lambda)}{2} Y^2_t - 2qv \int_t^T Y_s ds$$

and $Y_t$ is an Ornstein-Uhlenbeck process satisfying $dY_t = -\kappa Y_t dt + dB_t$, $Y_0 = 0$.

Proof. Since $X_t$ satisfies $dX_t = \lambda(a - X_t)dt + vdB_t$ we have

$$X_t - X_0 = \int_0^t \lambda(a - X_t)dt + \int_0^t vdB_t$$

$$\Rightarrow B_t = \frac{1}{v}(X_t - X_0 - \lambda at + \lambda \int_0^t X_s ds)$$

Next consider

$$\rho B_t + zU_t - qV_T =$$

$$= \exp \left\{ \frac{\rho}{v} \left( X_t - X_0 - \lambda a t + \lambda \int_0^t X_s ds \right) + z(X_t^2 + \delta) - q \int_0^T (X_s^2 + \delta) ds \right\}$$

$$= c_1 \exp \left\{ \frac{\rho}{v} \left( X_t + \lambda \int_0^t X_s ds \right) + zX_t^2 - q \int_0^T X_s^2 ds \right\}$$

$$= c_1 \exp \left\{ \frac{\rho}{v} \left( a + v\xi_t + \lambda \int_0^t (a + v\xi_s) ds \right) + z(a + v\xi_t)^2 - q \int_0^T (a + v\xi_s)^2 ds \right\}$$

where $c_1 = \exp \left\{ -\frac{\rho}{v}(a + \lambda at) + z \delta - q\delta T \right\}$. Now in light of theorem 2.4.6 define

$$\mathcal{E}_T(\lambda) \triangleq \frac{\mu_{\xi,T}}{\mu_{B,T}} = \exp \left\{ -\lambda \int_0^T B_t dB_t + \frac{\lambda^2}{2} \int_0^T B^2_t dt \right\}$$

46
Then consider

\[ \Phi_1(\rho, z, q) \]

\[ = \mathbb{E} \left[ \exp \left\{ \rho B_t + zU_t - qV_T \right\} \right] \]

\[ = c_1 \mathbb{E} \left[ \exp \left\{ \frac{\rho}{v} \left( a + v \xi_t + \lambda \int_0^t (a + v \xi_s) ds \right) + z(a + v \xi_t)^2 - q \int_0^T (a + v \xi_s)^2 ds \right\} \right] \]

\[ = c_1 \mathbb{E} \left[ \mathcal{E}_T(\lambda) \exp \left\{ \frac{\rho}{v} \left( a + v B_t + \lambda \int_0^t (a + v B_s) ds \right) + z(a + v B_t)^2 - q \int_0^T (a + v B_s)^2 ds \right\} \right] , \text{ by theorem 2.4.6} \]

\[ = c_1 c_2 \mathbb{E} \left[ \exp \left\{ (2zav + \rho)B_t + zv^2B_t^2 + \lambda \rho \int_0^t B_s ds - 2qav \int_0^T B_s ds \right. \right. \]

\[ - \frac{1}{2} (\lambda^2 + 2qv^2) \int_0^T B_s^2 ds - \sqrt{\lambda^2 + 2qv^2} \int_0^T B_s dB_s \]

\[ + \left( \sqrt{\lambda^2 + 2qv^2} - \lambda \right) \int_0^T B_s dB_s \left\} \right] \]

\[ = c_1 c_2 \mathbb{E} \left[ \mathcal{E}_T(\kappa) \exp \left\{ (2zav + \rho)B_t + zv^2B_t^2 + \lambda \rho \int_0^t B_s ds - 2qav \int_0^T B_s ds \right. \right. \]

\[ + (\kappa - \lambda) \int_0^T B_s dB_s \left\} \right] \]

where

\[ \kappa = \sqrt{\lambda^2 + 2qv^2} \]

\[ c_2 = \exp \left\{ \frac{\rho}{v} (a + \lambda at) + za^2 - qa^2T \right\} \]

From corollary 2.3.19 we have \( \int_0^T B_s dB_s = (1/2)(B_T^2 - T) \) and applying theorem 2.4.6 once more with \( \mathcal{E}_T(\kappa) \) we have

\[ \Phi_1(\rho, z, q) \]

\[ = c_1 c_2 \mathbb{E} \left[ \exp \left\{ (2zav + \rho)Y_t + zv^2Y_t^2 + \lambda \rho \int_0^t Y_s ds - 2qav \int_0^T Y_s ds \right. \right. \]

\[ + \left( \frac{\kappa - \lambda}{2} \right) (Y_T^2 - T) \left\} \right] \]

\[ = c_1 c_2 c_3 \mathbb{E} \left( e^{\eta} e^\xi \right) \]

47
where \( c_3 = \exp\{ (\lambda - \kappa)T/2 \} \) and \( Y_t \) is an Ornstein-Uhlenbeck process satisfying \( dY_t = -\kappa Y_t dt + dB_t, \ Y_0 = 0 \). Finally we set \( c^{(1)} = c_1c_2c_3 \).

Note that the random variable \( \eta_1 \) is \( F_t \)-measurable. To evaluate \( \mathbb{E} (e^{\eta_1} e^{\xi}) \) we will require \( \mathbb{E}(e^{\xi}|Y_t) \).

**Lemma 4.3.2.** Continuing from lemma 4.3.1 there exist known constants \( H, J \) and \( L \) such that

\[
\mathbb{E}(e^{\xi}|Y_t) = \exp \{ HY_t^2 + JY_t + L \}
\]

The constants are obtainable by a symbolic algebra package such as Mathematica.

**Proof.** Set \( X_1 = \int_0^T Y_s ds, \ X_2 = Y_T \) and \( X_3 = Y_t \). Since Ornstein-Uhlenbeck processes are Gaussian it follows that \( (X_1, X_2, X_3)^\top \) has a multivariate Normal distribution with mean \((0,0,0)^\top\) and covariances \( \sigma_{ij} = \text{Cov}(X_i, X_j) \). Details regarding calculation of \( \sigma_{ij} \) are given in section A.1.

The distribution of \( (X_1, X_2)^\top \) given \( X_3 = z \) is also multivariate Normal with mean vector and covariance matrix given by

\[
\mu = \begin{bmatrix} \sigma_{13} \\ \frac{\sigma_{33}}{2} \\ \frac{\sigma_{23}}{2} \\ \frac{\sigma_{23}}{2} \\ \frac{\sigma_{33}}{2} \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sigma_{11} & \frac{\sigma_{13}^2}{\sigma_{33}} & \sigma_{12} - \frac{\sigma_{13}\sigma_{23}}{\sigma_{33}} \\ \frac{\sigma_{33}}{2} & \sigma_{22} - \frac{\sigma_{23}^2}{\sigma_{33}} & \sigma_{23} \end{bmatrix} \tag{4.6}
\]

To compute \( \mathbb{E}(e^{\xi}|Y_t) \) we need

\[
\mathbb{E} (e^{\xi}|Y_t = z) = \mathbb{E} \left( \exp \left\{ -2qavX_1 + \frac{(\kappa - \lambda)X_2^2}{2} \right\} \bigg| X_3 = z \right) \\
= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left\{ -2qavx + \frac{(\kappa - \lambda)y^2}{2} \right\} f_{X_1,X_2|X_3}(x,y \mid z) \, dx \, dy \tag{4.7}
\]

where \( f_{X_1,X_2|X_3}(x,y \mid z) \) is the density function of \( (X_1, X_2)^\top \) given \( X_3 = z \).
Computation of the double integral of (4.7) essentially requires us to solve

\[ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left\{ -Ax^2 - By^2 + Cx + Dy + Fxy + G \right\}
\] 
\[ dx \, dy \]  \hspace{1cm} (4.8)

where \( A, B, C, D, F \) and \( G \) are constants. If \( F^2 < 4AB \) then by theorem A.1.1 the solution of (4.8) is

\[ 2\pi \exp \left\{ \frac{BC^2 + D(AD + CF)}{4AB - F^2} + G \right\} (4AB - F^2)^{-1/2}. \]  \hspace{1cm} (4.9)

Expressing the integrand of (4.7) as \( \exp \{g(x, y)\} \), we can obtain expressions for \( A, B, C, D, F \) and \( G \) by using the Mathematica commands:

\[ A = -\text{Coefficient}[g(x, 0), x^2] \]
\[ B = -\text{Coefficient}[g(0, y), y^2] \]
\[ C = \text{Coefficient}[g(x, 0), x] \]
\[ D = \text{Coefficient}[g(0, y), y] \]
\[ F = \text{Coefficient}[g(x, y), xy] \]
\[ G = g(0, 0) \]

Note from (4.6) that the \( z \) term appears only in the mean vector; in light of the form of a density function of a two-dimensional Normal distribution it follows that \( z \) only appears in the constants \( C, D, F \) and \( G \). Thus \( \mathbb{E} \left( e^\zeta | Y_t = z \right) = \exp \{ Hz^2 + Jz + L \} \) and

\[ \mathbb{E} \left( e^\zeta | Y_t \right) = \exp \{ HY_t^2 + JY_t + L \} \]

**Theorem 4.3.3.** The Laplace transform \( \Phi_1(\rho, z, q) \) is obtainable in explicit form.
*Proof.* By lemmas 4.3.1 and 4.3.2, we have

\[
\Phi_1(\rho, z, q) = c^{(1)} \mathbb{E} \left( e^{\eta} e^{\zeta} \right) \\
= c^{(1)} \mathbb{E} \left( e^{\eta} \mathbb{E} [ e^{\zeta} | \mathcal{F}_t ] \right) \\
= c^{(1)} \mathbb{E} \left( e^{\eta} \mathbb{E} [ e^{\zeta} Y_t ] \right) \\
= c^{(1)} \mathbb{E} \left( e^{\eta} \exp \{ HY_t^2 + J Y_t + L \} \right)
\]

Finally, the last expectation above is of the form of equation (4.8), which may be determined explicitly to obtain \( \Phi_1 \).

The derivation of the Laplace transform \( \Phi_2(\rho, z, r, q) \) is essentially the same as \( \Phi_1(\rho, z, q) \) but with some extra terms.

**Lemma 4.3.4.** The Laplace transform \( \Phi_2(\rho, z, r, q) \) can be expressed as

\[
\Phi_2(\rho, z, r, q) = c^{(2)} \mathbb{E} \left[ \exp \{ \eta_2 \} \exp \{ \zeta \} \right]
\]

where

\[
c^{(2)} = \exp \left\{ (z + r)(\delta + a^2) - (\delta + a^2)qT + (\lambda - \kappa)T/2 \right\},
\]

\[
\kappa = \sqrt{\lambda^2 + 2qv^2},
\]

\[
\eta_2 = (\rho + za\sigma)Y_t + zv^2Y_t^2 + (\lambda \rho - 2qv) \int_0^t Y_u du + ravY_s + rv^2Y_s^2,
\]

\[
\zeta = \frac{(\kappa - \lambda)}{2} Y_T^2 - 2qv \int_t^T Y_u du
\]

and \( Y_t \) is an Ornstein-Uhlenbeck process satisfying \( dY_t = -\kappa Y_t dt + dB_t, \ Y_0 = 0 \).

*Proof.* The proof is exactly the same as lemma 4.3.1, except for the presence of some extra terms. We remind the reader that within \( \Phi_2 \), there is the assumption that \( s \leq t \).

**Theorem 4.3.5.** The Laplace transform \( \Phi_2(\rho, z, r, q) \) is obtainable in explicit form.
Proof. Applying lemmas 4.3.4 and 4.3.2, we have

\[ \Phi_2(\rho, z, r, q) = e^{(2)E} \left[ \exp \{ \eta_2 \} \exp \{ HY_t^2 + JY_t + L \} \right] \]

Note that the expectation is a triple integral involving the joint probability density function of the multivariate normal distribution of \((Y_s, Y_t, \int_0^t Y_u du)^T\), and this can be evaluated symbolically in Mathematica.

\[ \square \]

4.4 Integrability of VWAP moments

Here we prove the integrability result to ensure that our VWAP moments are finite. We first establish some lemmas.

**Lemma 4.4.1.** Define \( g : \mathbb{R}^+ \rightarrow \mathbb{R}^+ \) by

\[ g(q) = \Phi_1(0, 0, q) \mid_{a=0, \delta=0} \]

where the Laplace transform \( \Phi_1 \) was given in lemma 4.2.2. Then

\[ g(q) = \left( \frac{2\kappa e^{\lambda T}}{(\kappa - \lambda) e^{-\kappa T} + (\kappa + \lambda) e^{\kappa T}} \right)^{1/2} \]

**Proof.** Applying lemma 4.3.1 with \( a = 0 \) and \( \delta = 0 \) we have

\[ g(q) = \exp \left\{ \frac{(\lambda - \kappa)T}{2} \right\} \mathbb{E} \left[ \exp \left\{ \frac{(\lambda - \kappa)T}{2} Y_T^2 \right\} \right] \]

where \( Y_t \) is an Ornstein-Uhlenbeck process satisfying \( dY_t = \kappa Y_t dt + dB_t, Y_0 = 0 \). From theorem 2.4.5, \( Y_T \) is a Gaussian random variable with mean and variance given by 0 and \((1 - e^{-2\kappa T})/2\kappa \) respectively. Now consider

\[ \mathbb{E} \left[ \exp \left\{ \frac{(\lambda - \kappa)T}{2} Y_T^2 \right\} \right] = \int_{-\infty}^{\infty} \left[ \frac{\pi(1 - e^{-2\kappa T})}{\kappa} \right]^{-\frac{1}{2}} \exp \left\{ \frac{(\kappa - \lambda)}{2} y^2 - \kappa y^2 / (1 - e^{-2\kappa T}) \right\} dy \]

\[ = \sqrt{\frac{\kappa}{\pi(1 - e^{-2\kappa T})}} \int_{-\infty}^{\infty} \exp \left\{ -y^2 \left[ -\frac{(\kappa - \lambda)}{2} + \frac{\kappa}{1 - e^{-2\kappa T}} \right] \right\} dy \]

51
Next note that

\[- \frac{(\kappa - \lambda)}{2} + \frac{\kappa}{1 - e^{-2\kappa T}} = \frac{(\lambda - \kappa)(1 - e^{-2\kappa T}) + 2\kappa}{2(1 - e^{-2\kappa T})} = \frac{1}{2\frac{1 - e^{-2\kappa T}}{(\lambda - \kappa)(1 - e^{-2\kappa T}) + 2\kappa}}.\]

Noting the form of a density function of a Gaussian random variable we obtain

\[
\mathbb{E} \left[ \exp \left\{ \frac{(\lambda - \kappa)T}{2} Y^2 \right\} \right] = \left( \frac{\kappa}{\pi(1 - e^{-2\kappa T})} \times \frac{2\pi(1 - e^{-2\kappa T})}{(\lambda - \kappa)(1 - e^{-2\kappa T}) + 2\kappa} \right)^{1/2} = \left( \frac{2\kappa}{(\kappa + \lambda) + (\kappa - \lambda)e^{-2\kappa T}} \right)^{1/2},
\]

The result follows.

\[\square\]

**Lemma 4.4.2.** Continuing from lemma 4.4.1, \(g(q) = O(e^{-\kappa T})\) as \(q \to \infty\).

**Proof.** Consider

\[
g(q) = \left[ \frac{2\kappa e^{\lambda T} + \kappa T}{(\kappa - \lambda) + (\kappa + \lambda)e^{2\kappa T}} \right]^{1/2} = \sqrt{2e^{\lambda T}} \left[ \frac{\kappa e^{\kappa T}}{(\kappa - \lambda) + (\kappa + \lambda)e^{2\kappa T}} \right]^{1/2} \leq \sqrt{2e^{\lambda T}} \left[ \frac{\kappa e^{\kappa T}}{\kappa e^{2\kappa T}} \right]^{1/2}, \text{ since } \kappa - \lambda \geq 0
\]

\[
= \sqrt{2e^{\lambda T}} \frac{1}{\sqrt{e^{-\kappa T}}} \leq \sqrt{2e^{\lambda T}} \left( \frac{1}{\kappa e^{-\kappa T}} \right), \text{ for } \kappa \geq 1/T
\]

This proves the result.
**Theorem 4.4.3.** Let $p > 0$. Then

$$
\mathbb{E} \left( \frac{U_t}{V_T} \right)^p < \infty
$$

**Proof.** The Laplace transform of $V_T$ given by $\Phi(0, 0, q)$ was originally derived in [46]. In particular, the following expression was obtained in [46] (see also Section 17.3 in [35]) for the case $a = 0$ and $q \geq 0$:

$$
g(q) = \mathbb{E} \exp \left( -qv^2 \int_0^T \xi_s^2 ds \right) = \left[ \frac{2\kappa e^{\lambda T}}{(\kappa - \lambda)e^{-\kappa T} + (\kappa + \lambda)e^{\kappa T}} \right]^{1/2},
$$

where the process $\xi_s$ is defined in (4.5) and $\kappa = \sqrt{\lambda^2 + 2qv^2}$. In view of Andersen’s Lemma ([3], see also Section 2.10 in [21]) and taking into account equation (4.4) we have for any $X_0 = a$ and $x > 0$

$$
\mathbb{P} \left\{ \int_0^T X_s^2 ds < x \right\} \leq \mathbb{P} \left\{ v^2 \int_0^T \xi_s^2 ds < x \right\}.
$$

This implies the following estimate for any $p > 0$

$$
\mathbb{E}(V_T^{-p}) = \frac{1}{\Gamma(p)} \int_0^\infty q^{p-1} \Phi(0, 0, q) dq \leq \frac{1}{\Gamma(p)} \int_0^\infty q^{p-1} g(q) dq.
$$

Since $g(q) = O(e^{-\kappa T})$ as $q \to \infty$ this estimate implies

$$
\mathbb{E}(V_T^{-p}) < \infty.
$$

When $\delta > 0$ this result clearly holds. Since $U_t$ is a shifted squared Gaussian process we have also $\mathbb{E}(U_t^p) < \infty$ for any $p > 0$. Using the Hölder inequality we obtain that for any $p > 0$

$$
\mathbb{E}(U_t^p/V_T^p) < \infty
$$

$\square$
The next chapter will use the VWAP moments derived here to give approximated prices to options with VWAP as underlying.
Chapter 5

Pricing options with VWAP as underlying

The previous chapter provided semi-analytical formulas for the first and second VWAP moments. We can use these moments to give approximated prices to options with VWAP as the underlying.

5.1 Pricing by the moment-matching method

The moment matching approach is a method of approximating a complicated stochastic process with a simpler process, by imposing a condition where some number of moments of both processes are equal at some specified times. We match the minimum number of moments such that the parameters of the simpler process may be obtained. For example, Geometric Brownian motion $S_t$ is a process with 2 parameters; the drift $\mu$ and volatility $\sigma$. By corollary 2.4.4 the mean and variance of $S_t$ is given by

$$
E[S_t] = S_0 e^{\mu t},
$$

$$
\text{Var}(S_t) = S_0^2 e^{2\mu t} \left( e^{\sigma^2 t} - 1 \right). \tag{5.1}
$$

Given a process $A_t$ where the first and second moments are known we can then match moments $E[S_t] = E[A_t]$ and $E[S_t^2] = E[A_t^2]$. Solving (5.1) for $\mu$ and $\sigma$, we obtain

$$
\mu = \frac{1}{t} \log \left( \frac{E[A_t]}{S_0} \right),
$$

$$
\sigma^2 = \frac{1}{t} \log \left( \frac{E[A_t^2]}{(E[A_t])^2} \right). \tag{5.2}
$$
Using the formulas for the first and second VWAP moments derived in chapter 4 we are able to give approximated prices for options with a VWAP as underlying. In particular the payoff function of a VWAP-underlying European call option with strike price $K$ and maturity $T$ is $\max(A_T - K, 0)$.

**Theorem 5.1.1. (Pricing by lognormal approximation)** Let $A_t$ denote the VWAP process defined in section 4.1 and $S_t$ denote a geometric Brownian motion with drift and volatility parameters $\mu$ and $\sigma$, respectively (see section 2.4.1). Approximating $A_t$ by $S_t$ with the following matching of the moments

$$
\mathbb{E}[S_T] = \mathbb{E}[A_T], \\
\mathbb{E}[S_T^2] = \mathbb{E}[A_T^2]
$$

the price of a VWAP-underlying European call option under real-world measure is approximated by

$$
\mathbb{E}[\max(A_T - K, 0)] \approx \int_{\frac{K}{S_0}}^{\infty} (xS_0 - K)f(x)dx
$$

where $f(x)$ denotes the density of the lognormal distribution with parameters $\tilde{\mu} = \left(\mu - \frac{\sigma^2}{2}\right)t$ and $\tilde{\sigma}^2 = \sigma^2 t$.

**Proof.** Given values for the VWAP moments we obtain parameter values for $\mu$ and $\sigma$ from (5.2). Since $S_t$ is a GBM, we have

$$
S_t = S_0 \exp \left\{ \sigma W_t + \left( \mu - \frac{\sigma^2}{2} \right) t \right\}
$$

$$
\Rightarrow \ln \left( \frac{S_t}{S_0} \right) = \left( \mu - \frac{\sigma^2}{2} \right) t + \sigma W_t
$$

$$
= \left( \mu - \frac{\sigma^2}{2} \right) t + \sigma \sqrt{t} Z
$$
where $Z \sim N(0, 1)$. So $S_t/S_0$ follows a lognormal distribution with parameters $\bar{\mu}$ and $\bar{\sigma}^2$. Now consider

$$\mathbb{E}[\max(A_T - K, 0)] \approx \mathbb{E}[\max(S_T - K)]$$

$$= \mathbb{E}\left[\frac{S_T}{S_0} - K\right]$$

$$= \int_{-\infty}^{\infty} \max(xs_0 - K, 0)f(x)dx$$

$$= \int_{K/s_0}^{\infty} (xs_0 - K)f(x)dx$$

Suppose that we are able to compute the third VWAP moment $\mathbb{E}[A_T^3]$. Then we may perform a moment-matching approximation with a better fitting distribution. In this thesis we fit to the generalised inverse Gaussian (GIG) distribution [23]. This distribution has a probability density function given by

$$f(x) = \frac{(a/b)^{p/2}}{2K_p(\sqrt{ab})}x^{(p-1)}e^{-ax^p/b}e^{-b/x^2}, x > 0$$

(5.3)

where $K_p$ is a modified Bessel function of the second kind, $a > 0$, $b > 0$ and $p \in \mathbb{R}$. We use $GIG(a, b, p)$ to denote this distribution. The $i^{th}$ moment of a GIG random variable $G$ is given by

$$\mathbb{E}[G^i] = \left(\frac{b}{a}\right)^{i/2} \frac{K_{p+i}(\sqrt{ab})}{K_p(\sqrt{ab})}.$$ 

**Theorem 5.1.2.** (Pricing by GIG approximation) Let $G \sim GIG(a, b, p)$. By matching the first three moments of $A_T$ and $G$,

$$\mathbb{E}[G^i] = \mathbb{E}[A_T^i], \ i = 1, 2, 3$$
the price of a VWAP-underlying European call option under real-world measure is then approximated by

$$\mathbb{E}[\max(A_T - K, 0)] \approx \int_K^{\infty} (x - K)f(x)dx$$

where $f(x)$ is given by (5.3).

Proof. Note that when matching moments $\mathbb{E}[G_i] = \mathbb{E}[A_T^i]$ for $i = 1, 2, 3$ we have a system of three non-linear equations with three unknowns $a$, $b$ and $p$. Solving for the unknown parameters numerically, the result easily follows. \hfill \Box
This chapter discusses the implementation of our approximated pricing scheme by matching of moments, verification of results by Monte Carlo simulation and comparing the accuracy of our schemes.

6.1 Implementation of the moment-matching pricing scheme

To price VWAP-underlying European options we proceed in two stages

1. Compute the VWAP moments as described in chapter 4.

2. If pricing by lognormal approximation, numerically integrate the expectation in theorem 5.1.1. Otherwise if approximating by the GIG distribution, numerically integrate the expectation in theorem 5.1.2.

6.1.1 Symbolic algebra with Mathematica

Here we remark on some intricacies in obtaining the VWAP moments. The key to obtaining $\mathbb{E}[A_T]$ and $\mathbb{E}[A_T^2]$ was to be able to evaluate integrals of the form

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left\{ -Ax^2 - By^2 + Cx + Dy + Fxy + G \right\} \, dx \, dy.$$

This integral had a solution given by (4.9) in section 4.2. To get a working implementation in Mathematica, we could not instruct the software to compute the integrals directly but had to re-arrange the integrand to the form just given above. Then we could match and obtain expressions for $A$, $B$, $C$, $D$, $F$ and $G$. This is done by the Mathematica command
**Coefficient.** For example given an expression \( f(x) \) we may obtain the coefficient of \( x^2 \) with the command

\[
\text{Coefficient} \left[ f[x], x^2 \right]
\]

In addition, partial derivatives of the Laplace transforms \( \Phi_1 \) and \( \Phi_2 \) were required. These can be computed exactly in Mathematica. For the final integral to compute the moments, we used standard numerical integration, which can be performed in Mathematica with the command \( \text{NIntegrate} \).

6.2 Verification by Monte Carlo simulation

We described an elaborate method to compute the VWAP moments in chapter 4 and applied the moment matching method to price options with VWAP underlying in chapter 5. We can verify all our computations by applying Monte Carlo simulation to estimate these quantities.

6.2.1 Basics of Monte Carlo simulation

Suppose that we are interested in computing the following integral:

\[
\alpha = \int_0^T \frac{f(x)}{T} \, dx, \text{ where } 0 < T < \infty \tag{6.1}
\]

The probability density function of the uniform distribution over \([0, T]\) is \(1/T\). Thus the above integral can be expressed as the expectation \( \mathbb{E}[f(X)] \) where \( X \sim \text{Uniform}(0, T) \). When the explicit solution of (6.1) is not known, we must settle for an approximation. The Monte Carlo method provides an approximation, and it is justified by the following well known result of probability theory:

**Theorem 6.2.1. (Strong Law of Large Numbers)** Let \( X_1, X_2, \ldots \) be a sequence of iid random variables with \( \mathbb{E}|X_i| < \infty \) and \( \mathbb{E}[X_i] = \mu \). Define \( \overline{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i \). Then \( \overline{X}_n \) converges to \( \mu \) almost surely, that is

\[
\mathbb{P} \left[ \lim_{n \to \infty} \overline{X}_n = \mu \right] = 1
\]
Proof. See Shiryaev [54].

So if we draw \( n \) independent samples \( X_i \sim \text{Uniform}(0, T) \) and define

\[
\hat{\alpha}_n = \frac{1}{n} \sum_{i=1}^{n} f(X_i)
\]

we get the following approximation,

\[
\hat{\alpha}_n \approx \int_{0}^{T} \frac{f(x)}{T} \, dx
\]

The quantity \( \hat{\alpha}_n \) is called the crude Monte Carlo estimate. To assess the accuracy of this approximation we apply the following famous theorem.

**Theorem 6.2.2. (Central Limit Theorem)** Let \( (X_j)_{j \geq 1} \) be a sequence of independent and identically distributed random variables with \( \mathbb{E}[X_j] = \mu \) and \( \text{var}(X_j) = \sigma^2 < \infty \), for all \( j \in \mathbb{N}^+ \). Define

\[
Y_n = \frac{\sum_{j=1}^{n} X_j - n\mu}{\sigma \sqrt{n}}
\]

Then \( Y_n \) converges in distribution to \( Y \sim N(0, 1) \).

Proof. See Jacod & Protter [22].

We assume that the integrand \( f \) is also square integrable and define

\[
\sigma^2_f = \int_{0}^{T} \frac{(f(x) - \alpha)^2}{T} \, dx
\]

Note that \( \sigma^2_f \) is the variance of \( f(X_j) \) for any \( j \). As a consequence of the Central Limit Theorem, for large \( n \) our estimator \( \hat{\alpha}_n \) will be approximately Normally distributed.
with mean $\alpha$ and standard deviation $\sigma_f/\sqrt{n}$. In practice, $\sigma_f$ is usually unknown but can be estimated by the sample standard deviation

$$s_f = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (f(X_i) - \hat{\alpha}_n)^2}$$

The quantity $SE\hat{\alpha}_n = s_f/\sqrt{n}$ is called the standard error and it provides a way to measure the error of approximation as described by the following theorem.

**Theorem 6.2.3.** Let $n \in \mathbb{N}^+$ and $X_1, X_2, \ldots, X_n$ be a sequence of i.i.d random variables with $\mathbb{E}[X_i] = \mu < \infty$ and $\text{Var}(X_i) = \sigma^2 < \infty$. Define

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$$
$$S_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X}_n)^2$$
$$L_n = L(X_1, \ldots, X_n) = \bar{X}_n - z_{\delta/2} \frac{S_n}{\sqrt{n}}$$
$$U_n = U(X_1, \ldots, X_n) = \bar{X}_n + z_{\delta/2} \frac{S_n}{\sqrt{n}}$$

where $z_{\alpha/2}$ denotes the $1 - \delta/2$ quantile of the standard normal distribution. Then $[L_n, U_n]$ is a confidence interval for $\mu$ with asymptotic confidence coefficient $1 - \delta$. i.e. $\lim_{n \to \infty} \mathbb{P}[L_n \leq \mu \leq U_n] = 1 - \delta$.

**Proof.** We first show that $\lim_{n \to \infty} S_n^2 = \sigma^2$ a.s. Define $T_n^2 = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X}_n)^2$. Then

$$T_n^2 = \frac{1}{n} \left( \sum_{i=1}^{n} X_i^2 - 2\bar{X}_n \sum_{i=1}^{n} X_i + n\bar{X}_n^2 \right)$$
$$= \frac{1}{n} \left( \sum_{i=1}^{n} X_i^2 - n\bar{X}_n^2 \right)$$
$$= \frac{1}{n} \sum_{i=1}^{n} X_i^2 - \bar{X}_n^2$$

Next note that $\mathbb{E}[X_i^2] = \sigma^2 + \mu^2 < \infty$, for all $i = 1, 2, \ldots, n$. It then follows from the Strong Law of Large Numbers that $\frac{1}{n} \sum_{i=1}^{n} X_i^2 \to \sigma^2 + \mu^2$ a.s. Similarly $\bar{X}_n \to \mu$
a.s. and since \( x \mapsto x^2 \) is continuous it follows that \( \bar{X}_n^2 \to \mu^2 \) a.s. as well. Thus \( T_n^2 \to \sigma^2 + \mu^2 - \mu^2 = \sigma^2 \) a.s.

Next note that \( S_n = \frac{n}{n-1} T_n \) and since \( \frac{n}{n-1} \to 1 \) we can now assert that \( \lim_{n \to \infty} S_n^2 = \sigma^2 \) a.s. Recalling that almost sure convergence implies convergence in probability, it follows that \( S_n / \sigma \xrightarrow{p} 1 \).

Applying the Central Limit Theorem, we have \( \sqrt{n}(\bar{X}_n - \mu) / \sigma \xrightarrow{d} N(0, 1) \). By Slutsky’s theorem,

\[
\frac{\sqrt{n}(\bar{X}_n - \mu)}{S_n} = \frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \xrightarrow{d} \frac{N(0, 1)}{1} = N(0, 1)
\]

Finally,

\[
P[L_n \leq \mu \leq U_n] = P[\bar{X}_n - z_{\delta/2} S_n / \sqrt{n} \leq \mu \leq \bar{X}_n + z_{\delta/2} S_n / \sqrt{n}] \\
= P[-z_{\delta/2} \leq \frac{\sqrt{n}(\bar{X}_n - \mu)}{S_n} \leq z_{\delta/2}] \\
\xrightarrow{n \to \infty} P[-z_{\delta/2} \leq Z \leq z_{\delta/2}] \\
= 1 - \delta
\]

where \( Z \sim N(0, 1) \). This proves the result.

So applying the preceding theorem, our interval estimate for \( \alpha \) is then

\[
[\hat{\alpha}_n - z_{\delta/2} SE_{\hat{\alpha}_n}, \hat{\alpha}_n + z_{\delta/2} SE_{\hat{\alpha}_n}]
\]

Note that the error of approximation decreases proportional to \( n^{-1/2} \). Suppose that we estimate \( \mu \) by using \( m \) samples. For fixed \( \delta \), an increase in accuracy of one decimal place requires a sample size \( n \) satisfying

\[
\frac{S_n}{\sqrt{n}} = 0.1 \\
\frac{S_m}{\sqrt{m}}
\]
and assuming that \( S_n \approx S_m \) it follows that \( n = 100m \). An increase in accuracy of one decimal place requires a 100 fold increase in samples generated.

### 6.2.2 Simulating stochastic processes

Note in theorem 6.2.3 that no assumptions are made on the distribution of the random variables \( X_i \). For certain tractable stochastic processes (such as Brownian motion) we may generate an entire trajectory of the process. This allows us to estimate expectations of path-dependent functions of these stochastic process by Monte Carlo simulation.

### 6.3 Results

We applied the moment-matching method to pricing call options with VWAP as the underlying. We matched to two processes; Geometric Brownian motion and the Generalised Inverse Gaussian distribution. For the asset price we used Geometric Brownian motion.

We chose parameters that were close to that of Stace’s published result (though of course there will be differences since Stace used different processes to model volume). In addition we assumed that the asset price and volume traded were independent \( (\rho = 0) \). This lead to the following models

\[
dS_t = (0.1)S_t^i dt + \sigma S_t^i dW_t, S_0 = 110
\]

for the asset price and

\[
U_t = X_t^2, \\
dX_t = 2(22 - X_t) dt + 5dW_t, X_0 = 22
\]

for the volume dynamics. We set maturity time \( T = 1 \). All our Monte Carlo simulations were performed using \( n = 1,000,000 \) trajectories and 500 discretisation points over \([0, T]\). Note that the GIG approximation requires the third VWAP moment, and here we used Monte Carlo simulation to obtain those values.

With these choices of parameters we can see from tables 6.1 and 6.2 that the relative error for the first and second VWAP moments between our method and Monte Carlo
Table 6.1: Our method to compute the first moment $E_{AT}$ is compared with Monte Carlo simulation $\hat{E}_{AT}$ for different stock price volatility $\sigma$.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>$E_{AT}^a$</th>
<th>$\hat{E_{AT}}$</th>
<th>MC std. error</th>
<th>Rel. error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>115.681</td>
<td>115.67</td>
<td>0.006</td>
<td>0.009</td>
</tr>
<tr>
<td>0.2</td>
<td>115.681</td>
<td>115.693</td>
<td>0.013</td>
<td>0.010</td>
</tr>
<tr>
<td>0.3</td>
<td>115.681</td>
<td>115.674</td>
<td>0.020</td>
<td>0.006</td>
</tr>
<tr>
<td>0.4</td>
<td>115.681</td>
<td>115.703</td>
<td>0.0276</td>
<td>0.0190</td>
</tr>
<tr>
<td>0.5</td>
<td>115.681</td>
<td>115.623</td>
<td>0.0348</td>
<td>0.050</td>
</tr>
</tbody>
</table>

Note that $\sigma$ does not enter into the computation of $E_{AT}$, which leads to unchanging values for this column.

Table 6.2: Computation of $E_{AT}^2$ is compared with Monte Carlo simulation $\hat{E}_{AT}^2$ for different stock price volatility $\sigma$.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>$E_{AT}^2$</th>
<th>$\hat{E_{AT}^2}$</th>
<th>MC std. error</th>
<th>Rel. error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>13427.92</td>
<td>13426.4</td>
<td>0.0068</td>
<td>0.0113</td>
</tr>
<tr>
<td>0.2</td>
<td>13566.90</td>
<td>13568.4</td>
<td>0.0136</td>
<td>0.0110</td>
</tr>
<tr>
<td>0.3</td>
<td>13803.32</td>
<td>13797.9</td>
<td>0.0205</td>
<td>0.0393</td>
</tr>
<tr>
<td>0.4</td>
<td>14144.68</td>
<td>14141</td>
<td>0.0276</td>
<td>0.02600</td>
</tr>
<tr>
<td>0.5</td>
<td>14602.11</td>
<td>14603</td>
<td>0.0349</td>
<td>0.0061</td>
</tr>
</tbody>
</table>

For $K = 100$ it can be seen that the relative error of the lognormal approximation stays within 1% in the volatility range $[0.2, 0.37]$ with a tendency to increase. The relative error for the GIG approximation stays beneath 0.8% over the entire range of volatilities. The lognormal relative error behaves similarly for $K = 110$, however we note that the GIG approximation struggles for small $\sigma$. In fact, for $\sigma < 0.13$ the GIG approximation is not suitable due to the instability of the numerical calculations. These findings for VWAP options conform with a classical market observation for Asian options, namely, for a small volatility of underlying process and near at-the-money options the log-normal approximation outperforms others; see [36].
Figure 6.1: Comparison of call option prices for different stock price volatility $\sigma$. Here $K = 100$.

Figure 6.2: Relative error of options prices in figure 6.1 as a function of $\sigma$. The solid line represents the lognormal approximation error and the dashed line is the corresponding error for GIG.
Figure 6.3: Relative error of options prices for the case $K = 110$ as a function of $\sigma$. The solid line represents the lognormal approximation error and the dashed line is the corresponding error for GIG.
Part II

The distribution of maximum of fractional Brownian motion and asymptotic variance of optimal Bayesian estimators
Chapter 7

Introduction

7.1 Overview

In part II of this thesis we study specific functionals of fractional Brownian motion (fBm) which arise from two longstanding problems in statistics. Let $W_t^{(H)}$ denote fBm with Hurst parameter $H \in [0, 1]$ and $t \in \mathbb{R}$ (see chapter 8 for the formal definition). Then the two problems of interest are

1. Finding the asymptotic of $\mathbb{P} \left[ \max_{0 \leq t \leq T} W_t^{(H)} \leq 1 \right]$ as $T \to \infty$.

2. Determining the variance of the random variable,

$$
\zeta_H = \frac{\int_{-\infty}^{\infty} u e^{Y_u} du}{\int_{-\infty}^{\infty} e^{Y_u} du}
$$

where $Y_u := W_t^{(H)} - \frac{1}{2} |u|^{2H}$.

The limiting distribution of the maximum of fBm in problem 1 has many important applications in areas as diverse as queuing theory and mechanics.

The random variable of problem 2 is a limiting distribution of the so-called Pitman estimator from a non-regular parameter estimation problem based on observations of a diffusion process.

Little is known about both problems for general values of $H$ and to date there has been no serious attempt to obtain empirical data through simulation studies. One reason is the lack of fast and efficient algorithms to simulate fBm trajectories. We now give a background to both the problems and a review of existing literature.
7.2 The problem settings and existing literature

7.2.1 Distribution of the maximum of fBm

The problem of finding an asymptotic of the maximum of fBm

\[ \mathbb{P} \left[ \max_{0 \leq t \leq T} W_t^{(H)} \leq 1 \right], T \to \infty \quad (7.1) \]

arises in many diverse applications such as fluid mechanics, pursuit problems of stochastic processes and random polynomials (see [34] for a discussion of these applications). Note that we may determine \( \mathbb{P} \left[ \max_{0 \leq t \leq T} W_t^{(H)} \leq x \right], T \to \infty \) for arbitrary \( x \) due to the self-similarity property of fBm.

In [56], an inequality for the density of crossing time leads to an upper bound of the distribution of the maximum of fBm. In [43] Molchan proved the following bound,

\[ T^{-(1-H)} e^{-k \sqrt{\log T}} \leq \mathbb{P} \left[ \max_{0 \leq t \leq T} W_t^{(H)} \leq 1 \right] \leq T^{-(1-H)} e^{+k \sqrt{\log T}} \quad (7.2) \]

for some positive constant \( k \) and \( T \) large enough.

More recently, Aurzada [4] improved on (7.2) with

\[ T^{-(1-H)} (\log T)^{-c} \leq \mathbb{P} \left[ \max_{0 \leq t \leq T} W_t^{(H)} \leq 1 \right] \leq T^{-(1-H)} (\log T)^{+c} \quad (7.3) \]

where \( c = c(H) > 0 \) is a constant and \( T \) is large enough. This is currently the sharpest known bound for the distribution.

7.2.2 Variance of the limiting distribution of certain Pitman estimators

Let \( X = (X_1, X_2, \ldots, X_n) \) be a \( \mathbb{R}^n \)-valued random variable with a probability density function \( f \) depending on an unknown parameter \( \theta \); we denote this density by \( f(x|\theta) \). In addition we suppose that the parameter \( \theta \in \Theta = (\alpha, \beta) \), where \( \Theta \) has a prior distribution with density denoted by \( \pi(\theta) \). Then the Bayes estimator for \( \theta \) under squared error loss (see for e.g. [31]) is given by
\[ \hat{\theta} = \frac{\int_{\Theta} v f(x|v) \pi(v) dv}{\int_{\Theta} f(x|v') \pi(v') dv'} = \frac{\int_{\Theta} v L(x|v) \pi(v) dv}{\int_{\Theta} L(x|v') \pi(v') dv'} \]

where the likelihood function is defined by \( L(v|x) = f(x|v) \).

Considering the estimation problem in the asymptotics of large samples, we move to continuous-time observations i.e. the observation of a stochastic process over \([0, T]\) for some \( T \in \mathbb{R}^+ \). In particular suppose that we can observe the following diffusion process \( X^T \) defined by

\[ dX_t = S(\theta, X_t) dt + \sigma(X_t) dW_t, \quad X_0, \ 0 \leq t \leq T \]  

(7.4)

where \( W_t \) is a standard Brownian motion, \( \sigma(\cdot) \) is a known function and \( S(\cdot, \cdot) \) is also a known function depending on the unknown parameter \( \theta \). Under this setting the Bayes estimator for \( \theta \) under squared error loss then becomes

\[ \hat{\theta} = \frac{\int_{\Theta} v L(v, X^T) \pi(v) dv}{\int_{\Theta} L(v', X^T) \pi(v') dv'} \]

where the likelihood ratio function \( L(\theta, X^T) \) is given by

\[ L(\theta, X^T) = \exp \left\{ -\frac{1}{2} \int_0^T \frac{S(\theta, X_t)}{\sigma(X_t)^2} dX_t - \frac{1}{2} \frac{S(\theta, X_t)^2}{\sigma(X_t)^2} dt \right\} \]

The study of the limiting distributions of maximum likelihood (MLE) and Bayesian estimations as \( T \to \infty \) has been of interest by researchers for some time. The monograph of Imbragimov and Khasminskii [21] considers the limiting behaviour of such estimators under a variety of different forms of \( S(\cdot, \cdot) \).

A particular example discussed in the monograph concerns the observation of

\[ dX_t = \frac{1}{\varepsilon} S(t - \theta) dt + dW_t \quad t \in [0, 1] \]

(7.5)

where \( S \) contains at least one discontinuity of the first kind and \( \varepsilon \) is a small parameter.
It was shown in [21] that when $\varepsilon \rightarrow 0$, the normalised Bayesian estimate for $\theta$ converges in distribution to $\zeta_2/c$ where $c$ is a known constant and

$$
\zeta_2 = \frac{\int_{-\infty}^{\infty} tZ_2^\varepsilon(t)dt}{\int_{-\infty}^{\infty} Z_2^\varepsilon(t)dt},
$$
(7.6)

$$
Z_2^\varepsilon(t) = \exp \left\{ W_t - \frac{1}{2} |t| \right\}
$$
(7.7)

Here $W_t$ denotes a two-sided standard Brownian motion. It was proved in [53] that

$$
\mathbb{E} \left[ \zeta_2^2 \right] = 16\zeta(3)
$$

where $\zeta(x) = \sum_{n=1}^{\infty} \frac{1}{n^x}$ is Riemann’s zeta function.

More recently, [11] studied properties of MLE and Bayesian estimators for $\theta$ under the setting of a cusp in the trend coefficient i.e. when the observed process is of the form,

$$
dX_t = (\kappa |X_t - \theta|^{\kappa - X_t}) dt + dW_t, \; X_0, \; 0 \leq t \leq T
$$

where $\kappa \in (0, \frac{1}{2})$. For $H \in (0, 1)$ we introduce the limit likelihood process

$$
Z_H(u) = \exp \left\{ W_t^{(H)} - \frac{|u|^{2H}}{2} \right\}
$$
(7.8)

where $W_t^{(H)}$ is a two-sided fractional Brownian motion with Hurst parameter $H$. It was then shown in [11] that for $H = \kappa + \frac{1}{2}$,

$$
\lim_{T \rightarrow \infty} T^{1/2H} (\tilde{\theta} - \theta) = \frac{\zeta_H}{\gamma_0}
$$
in distribution, where

$$
\zeta_H = \frac{\int_{-\infty}^{\infty} uZ_H(u)du}{\int_{-\infty}^{\infty} Z_H(u)du}
$$
(7.9)
and $\gamma_0$ is a known constant. Note that for $H = \frac{1}{2}$, the random variables of (7.7) and (7.8) coincide and similarly for (7.6) and (7.9).

Knowledge of $E[\zeta_H^2]$ is necessary for determining the asymptotic mean square error of $\tilde{\theta}$. To date nothing is known about this quantity for any $H \in (0,1) \setminus \{\frac{1}{2}\}$.

### 7.3 Summary of the contributions of part II

We take the current best exact-simulation algorithm for fBm and construct a parallel implementation for use on modern multicore desktop PCs. We then apply our implementation to obtain never before seen empirical results for problems 1 and 2. From these results we formulate conjectures which will assist in resolving these longstanding problems.
Chapter 8

Fractional Brownian motion

8.1 Basic properties

Fractional Brownian motion (fBm) is a continuous-time Gaussian process \( W_t^H \) with zero mean and covariance function given by

\[
\mathbb{E} [W_t^H W_s^H] = \frac{1}{2} (|t|^{2H} + |s|^{2H} - |t-s|^{2H})
\]

for \( t \in \mathbb{R} \). This formulation is sometimes called *two-sided fBm*. In many applications, only one-sided fBm is used, and we can obtain this by restricting \( t \in [0, \infty) \).

The following theorem gathers together some basic properties of fBm.

**Theorem 8.1.1.** \( W_t^H \) has the following properties,

(i) \( W_t^{(H)} \) is self similar: \( W_{at}^{(H)} \overset{d}{=} |a|^H W_t^{(H)} \)
(ii) \( W_t^{(H)} \) has stationary increments.
(iii) \( W_t^{(1/2)} = W_t \) is standard Brownian motion.
(iv) \( W_t^{(1)} = t \theta \) where \( \theta \sim N(0,1) \)

**Proof.** Since fBm is a zero mean process, we have

\[
\text{Var} \left( W_{at}^{(H)} \right) = \mathbb{E} \left[ W_{at}^{(H)} W_{at}^{(H)} \right] \\
= \frac{1}{2} \left( |at|^{2H} + |at|^{2H} - |at - at|^{2H} \right) \\
= |a|^{2H} |t|^{2H} \\
= \text{Var}(|a|^H W_t^{(H)})
\]
and this proves (i). To prove (ii) note that a stochastic process \( X \) has stationary increments if for all \( s, t \in \mathbb{R} \) with \( s \leq t \) we have \( X_t - X_s \overset{d}{=} X_{t-s} \). Now consider

\[
\begin{align*}
\text{Var} \left( W_t^{(H)} - W_s^{(H)} \right) &= \text{Var} \left( W_t^{(H)} \right) + \text{Var} \left( W_s^{(H)} \right) - 2 \text{Cov} \left( W_t^{(H)}, W_s^{(H)} \right) \\
&= |t-s|^{2H} \\
&= \text{Var} \left( W_{t-s}^{(H)} \right)
\end{align*}
\]

so (ii) follows. (iii) and (iv) are trivial to prove.

\[\square\]

8.2 Simulation of fBm

Let \( t_1 < t_2 < \cdots < t_n \) be points on the real line. By exactly simulating a process \( B \) we mean generating a realisation of a sequence of random variables \( \tilde{B}_{t_1}, \tilde{B}_{t_2}, \cdots, \tilde{B}_{t_n} \) such that the distributions of \( \tilde{B}_{t_i} \) and \( B_{t_i} \) coincide for all \( i = 1, \cdots, n \).

On the other hand we say that a simulation method is approximate if the distributions from which the generated realisations are drawn differ from the distributions of the original process.

The literature contains a number of exact and approximate methods to simulate fBm.

8.2.1 Exact simulation methods

Currently, all known methods in this class generate realisations of Gaussian processes; no special properties of fBm are used.

Cholesky decomposition method

Let \( \Gamma \) denote the covariance matrix of the random vector \( \mathbf{W}^{H} = (W_{t_1}^{H}, W_{t_2}^{H}, \cdots, W_{t_n}^{H})^{\top} \). Assuming \( \Gamma \) is positive definite, the Cholesky decomposition is \( \Gamma = LL^{\top} \), where \( L \) is a lower triangular matrix. Given a vector with independent and identically distributed Normal random variable components \( \xi = (\xi_1, \xi_2, \cdots, \xi_n)^{\top} \) it is easy to check that \( \mathbf{W}^{H} = L\xi \) in distribution.

Though clearly easy to implement, this method is not practical for intensive simulation. It is expensive in time, taking order \( n^3 \) operations to construct the matrix. It is
also memory intensive. Assuming double-precision floating point numbers, the memory-footprint of \( L \) for \( n = 10,000 \) is 382mb, and for \( n = 50,000 \) the amount required jumps to around 9gb!

**Hosking’s method**

The Hosking method [18] improves on the Cholesky method by exploiting properties of the conditional distribution of Gaussian random variables and the property that the covariance matrix of a stationary process is Toeplitz. Suppose that \( \{ X_t : t \in \mathbb{Z} \} \) is a stationary Gaussian process with \( \mathbb{E}[X_t] = \mu \) and autocovariance function \( \gamma(n) \). It is a well known fact that the conditional distribution of \( X_{t+n+1} \) given \( \xi_n = (X_{t_1}, X_{t_2}, \cdots, X_{t_n})^\top \) is Normally distributed with mean and variance given by

\[
\begin{align*}
\mathbb{E}[X_{t+n+1} | \xi_n] &= \mathbb{E}[X_{t+n+1}] + \text{cov}(X_{t+n+1}, \xi_n) [\text{cov}(\xi_n, \xi_n)]^{-1} (\xi_n - \mathbb{E}[\xi_n]) \\
\text{Var}[X_{t+n+1} | \xi_n] &= \text{Var}[X_{t+n+1}] - \text{cov}(X_{t+n+1}, \xi_n) [\text{cov}(\xi_n, \xi_n)]^{-1} [\text{cov}(X_{t+n+1}, \xi_n)]^\top 
\end{align*}
\]

Note that the conditional expectation in (8.1) is expressed as a linear combination of \( X_{t_1}, X_{t_2}, \cdots, X_{t_n} \), so the best predictor of \( X_{t+n+1} \) given \( \xi_n \) (which is \( \mathbb{E}[X_{t+n+1} | \xi_n] \)) is the best linear predictor \( a_0 + a_1 X_{t_n} + a_2 X_{t_{n-2}} + \cdots + a_n X_{t_1} \) for \( X_{t+n+1} \). By best, we mean the predictor \( Y \) that minimises mean squared error \( \mathbb{E}[(X_{t+n+1} - Y)^2] \).

Let \( \Gamma_n \) denote the autocovariance matrix of \( \xi_n \) and \( \gamma_n = (\gamma(1), \cdots, \gamma(n))^\top \). It can be shown (see [51]) that the conditional mean and variance in (8.1) and (8.2) have the form

\[
\begin{align*}
\mathbb{E}[X_{t+n+1} | \xi_n] &= \mu + \sum_{j=1}^{n} a_j^{(n)} (X_{t_{n+1-j}} - \mu) \\
\text{Var}[X_{t+n+1} | \xi_n] &= \gamma(0) \prod_{j=1}^{n} \left(1 - [a_j^{(n)}]^2\right) 
\end{align*}
\]

where the vector \( a_n = (a_1^{(n)}, \cdots, a_n^{(n)})^\top \) satisfies the equation

\[
\Gamma_n a_n = \gamma_n 
\]
Applying the Durbin-Levinson algorithm [8] we can recursively compute \( a_{n+1} \) given \( a_n \) in \( O(n) \) operations. The simulation method is then:

1. Simulate \( X_{t_1} \) from \( N(\mu, \gamma(0)) \).
2. Set \( n = 1 \) in (8.5) to obtain \( a^{(1)}_1 = \frac{\gamma^{(1)}}{\gamma^{(0)}} \). We can then simulate \( X_{t_2} \) from the normal distribution with mean and variance given by (8.3) and (8.4).
3. Given \( a_n \) and generated values \( X_{t_1}, X_{t_2}, \ldots, X_{t_n} \), use the Durbin-Levinson algorithm to compute \( a_{n+1} \) and then generate \( X_{t_{n+1}} \).

This method requires order \( n^2 \) operations and uses far less memory than the Cholesky method. It also allows for on-demand generation; given an existing trajectory and the latest vector \( a \), we can readily generate a new point in the trajectory.

**Circulant embedding method**

The *circulant embedding* method [12] generates stationary Gaussian processes. It is based on the discrete Fourier transform (DFT) and thus generates a trajectory of \( n \) points in \( O(n \log n) \) time. This is the algorithm that we use for our simulation studies, so we now discuss the workings of the method in some detail.

If we take \( X \) to be fractional Brownian noise, we can take cumulative sums and use the self-similarity property of fBm to generate a trajectory of fBm over any interval \([0, L]\). Let \( \Gamma_n \) denote the covariance matrix of \((X_0, X_1, \ldots, X_{n-1})^\top \).

A circulant matrix \( C \) is a square matrix of the form

\[
C = \begin{pmatrix}
c_0 & c_{n-1} & \cdots & c_2 & c_1 \\
c_1 & c_0 & c_{n-1} & \cdots & c_2 \\
\vdots & c_1 & c_0 & \ddots & \vdots \\
c_{n-2} & \vdots & \ddots & \ddots & c_{n-1} \\
c_{n-1} & c_{n-2} & \cdots & c_1 & c_0
\end{pmatrix}
\]  

(8.6)

Note that the first column of \( C \) uniquely determines the entire matrix, as the \( n^{th} \) column is a cyclic downshift of the \( (n-1)^{th} \) column, for \( n > 1 \). The cyclic downshift is defined by \((c_0, c_1, \ldots, c_{n-1})^\top \mapsto (c_{n-1}, c_0, c_1, \ldots, c_{n-2})^\top \).
Let $\omega = \exp\{-2\pi i/n\}$ where $i = \sqrt{-1}$. It is easy to check that the eigenvalues of $C$ are $\lambda_s = \sum_{j=0}^{n-1} c_j \omega^{-js}$ with associated eigenvectors $v_s = (1, \omega^s, \omega^{2s}, \cdots, \omega^{(n-1)s})^\top$ for $s = 0, \cdots, n-1$.

Next note that the covariance matrix of a stationary process is Toeplitz (a matrix with constant diagonals). Defining a $2n \times 2n$ circulant matrix $C$ with first column

$$(\gamma(0), \gamma(1), \cdots, \gamma(n-1), \gamma(n-1), \gamma(n-2), \cdots, \gamma(1))^\top$$

it can be seen that the covariance matrix $\Gamma_n$ is embedded into the top left of $C$. The matrix $C$ is also symmetric and thus admits an eigendecomposition $Q\Lambda Q^*$, where $Q$ is a unitary matrix with conjugate transpose $Q^*$ and $\Lambda = \text{diag}\{\lambda_0, \lambda_1, \cdots, \lambda_{n-1}\}$. The $jk$th element of $Q$ is

$$Q_{j,k} = \frac{1}{\sqrt{n}}\omega^{jk}$$

**Theorem 8.2.1.** $Q^*Z = S + iT$ where $S = (S_0, \cdots, S_{n-1})^\top$ and $T = (T_0, \cdots, T_{n-1})^\top$ are real-valued Gaussian random vectors with zero mean and

1. $S$ and $T$ are independent.
2. If $0 \leq j < n$ and $j = 0$ or $j = n/2$ then for each $k$, $0 \leq k < n$

$$\text{cov}(S_j, S_k) = \mathbb{1}_{\{j=k\}}$$

$$T_j = 0$$

3. For $j \in \{0, \cdots, n-1\}$ define $k = k(j)$ $\{0, \cdots, n-1\}$ by

$$k(j) = \begin{cases} j, & \text{if } j = 0 \text{ or } j = n/2 \\ n - j, & \text{otherwise} \end{cases}$$
Then for \( j \in \{1, \cdots, n-1\} \setminus \{n/2\} \) we have for each \( h \in \{0, \cdots, n-1\} \)

\[
\text{cov}(S_j, S_h) = \begin{cases} 
  j, & \text{if } h = j \text{ or } h = k \\
  0, & \text{otherwise}
\end{cases}
\]

\[
\text{cov}(T_j, T_h) = \begin{cases} 
  \frac{1}{2}, & \text{if } h = j \\
  -\frac{1}{2}, & \text{if } h = k \\
  0, & \text{otherwise}
\end{cases}
\]

**Proof.** See proposition 3 of [62]. \(\square\)

**Corollary 8.2.2.** Let \( X \) and \( Y \) denote independent vectors of \( m/2 \) i.i.d. standard normal random variables. Define the random vector \( W \) of length \( m \) by

\[
\begin{align*}
W_0 &= X_0 \\
W_{m/2} &= Y_0 \\
W_j &= \frac{1}{\sqrt{2}} (X_j + iY_j) \\
W_{m-j} &= \frac{1}{\sqrt{2}} (X_j - iY_j)
\end{align*}
\]

where \( j \in \{1, \cdots, m-1\} \). Then \( W \) has same distribution as \( Q^*Z \).

**Proof.** It is routine to check that \( W \) has the same covariance structure as \( Q^*Z \) given by theorem 8.2.1. \(\square\)

Next note that

\[
Q^*\Lambda^{1/2} = \begin{pmatrix}
\sqrt{\lambda_0} Q_{0,0} & \sqrt{\lambda_1} Q_{0,1} & \cdots & \sqrt{\lambda_{n-1}} Q_{0,n-1} \\
\sqrt{\lambda_0} Q_{1,0} & \sqrt{\lambda_1} Q_{1,1} & \cdots & \sqrt{\lambda_{n-1}} Q_{1,n-1} \\
\vdots & \vdots & \ddots & \vdots \\
\sqrt{\lambda_0} Q_{n-1,0} & \sqrt{\lambda_1} Q_{n-1,1} & \cdots & \sqrt{\lambda_{n-1}} Q_{n-1,n-1}
\end{pmatrix}
\]
This leads to the following formula for $Y$

$$Y = Q \Lambda^{1/2} Q^* Z$$

$$= Q \Lambda^{1/2} W$$

$$= \left( \sum_{k=0}^{n-1} \sqrt{\lambda_k} Q_{0,k} W_k, \sum_{k=0}^{n-1} \sqrt{\lambda_k} Q_{1,k} W_k, \ldots, \sum_{k=0}^{n-1} \sqrt{\lambda_k} Q_{n-1,k} W_k \right)^T$$

$$= \left( \sum_{k=0}^{n-1} \sqrt{\lambda_k} \omega^{0 \times k} W_k, \sum_{k=0}^{n-1} \sqrt{\lambda_k} \omega^{1 \times k} W_k, \ldots, \sum_{k=0}^{n-1} \sqrt{\lambda_k} \omega^{(n-1)k} W_k \right)^T \quad (8.7)$$

We then scalar-multiply $Y$ by $(\frac{1}{2\pi})^H$ and take the first $n/2$ points of $Y$, which is the generated trajectory of the increments of fBm. Computing cumulative sums then gives us the required fBm trajectory.

8.2.2 Approximate simulation methods

There are a range of approximate simulation methods which we briefly survey here. Since we are looking for the highest accuracy in our simulation studies, we don’t apply any methods in this subsection.

It was seen that the Hosking method generated each point of a trajectory by conditioning on all previously generated points. By restricting the number of points to condition on, we have an approximate simulation method. For a method utilising this idea, see [45].

Finally, it was shown in [14] that fBm with $H \in (0, 1)$ has the following series representation

$$W^H_t = \sum_{n=1}^{\infty} \frac{\sin(x_n t)}{x_n} X_n + \sum_{n=1}^{\infty} \frac{1 - \cos(y_n t)}{y_n} Y_n \quad (8.8)$$

where $x_1 < x_2 < \cdots$ are the positive zeros of the Bessel function $J_{-H}$ of the first kind of order $-H$ and $y_1 < y_2 < \cdots$ are the positive zeros of $J_{1-H}$. Also, $X_1, X_2, \cdots$ and $Y_1, Y_2, \cdots$ are independent Gaussian random variables with zero mean with variance given by $\text{Var}(X_n) = 2c_H^2 x_n^{-2H} J_{1-H}^2(x_n)$, $\text{Var}(Y_n) = 2c_H^2 y_n^{-2H} J_{-H}^2(y_n)$ and $c_H^2 = \pi^{-2} \Gamma(1+2H) \sin(\pi H)$. The series (8.8) converges absolutely and uniformly for $t \in [0, 1]$ with probability 1.
To simulate fBm with this representation, it is necessary to truncate the series. In [29], error bounds for the truncated series are proved in a more general setting where the $X_n$ and $Y_n$ are so-called generalised sub-Gaussian random variables. These results were then used in [30] to simulate fBm.
One of the major objectives of part two is to construct a parallel implementation of the fastest known exact simulation method for fBm. This is essential for us to obtain empirical results from Monte Carlo simulation in a reasonable amount of time.

There are major challenges with parallel programming. Knowledge of the underlying hardware architecture is required to write parallel programs. We begin this section with a quick summary of the classical uniprocessor architecture.

### 9.1 Basic Computer Architecture

All terminology defined in this section comes from the reference textbook of Patterson and Hennessy [48].

A computer is a machine that carries out instructions to manipulate data. Modern computers rest upon the fundamental principle that instructions and data may be stored internally as numbers. This is called the *stored-program* concept. Viewed through the hardware perspective, a computer is just a machine composed of millions of on/off switches called transistors. Using the stored-program concept, we can represent all data and instructions as strings of binary (0/1) digits which have a one-to-one correspondence with the on/off states of the transistor. For example, consider the integer 97 which may be represented as

\[
97 = 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0
\]
Therefore 97 can be represented as the binary number 1100001, a form that is suitable for use by a computer.

Every modern computer has an *instruction set architecture*, which we can regard as the language of the computer. Collections of instructions from the instruction set are called *programs*, which are executed by the computer to perform a desired task. Basic instructions all computers must be able to perform are

1. *Arithmetic*. All data and instructions are represented as numbers, so it is crucial to be able to apply arithmetic operations on them.
2. *Data transfer*. Programs need to be able to read and store data in memory.
3. *Conditional branch*. Branching instructions allow the computer to make decisions. It is the functionality that allows a programmer to use *if statements*, for example.
4. *Jump*. These instructions are used to shift control of a program to another location.

It is required to implement subroutines or functions in high-level languages.

Today, the most common way to program a computer is to use a *high-level programming language* such as C. A special program called a *compiler* translates the language into *assembly language*. From this point the instructions are transformed and processed further and the end product is a functional program.

### 9.1.1 The “Power Wall”

Over the last two decades, CPU clock rates have been increasing rapidly. Roughly speaking, the clock rate is a measure of how fast a computer can execute instructions; the faster the clock rate, the faster a program will run. In 1982, the clock rate of an Intel 80286 CPU was 12.5 MHz. 22 years later, the Intel Pentium 4 had clock rates in excess of 3000 MHz [48]. However as the clock rate of the CPU increases there is an associated increase in the power consumption. This leads to more heat-dissipation, and there is only so much cooling that a manufacturer can apply to prevent overheating of the CPU. This upper bound on the clock rate has been termed the *Power Wall* [39] [48].

In response to the Power Wall, a shift to parallel processors was made in order to achieve performance increases. A *multicore microprocessor* is one which contains two or more independent processors (*cores*) on a single chip.
Most mainstream programming tasks involving high level languages require little knowledge of the underlying hardware. Modern compilers are able to perform sophisticated optimisations (without any programmer input) to generate high performance code. Before the arrival of the Power Wall in 2004, microprocessor manufacturers were releasing CPUs with increasingly faster clock rates. This provided a recurring free lunch for software developers. As described in [58],

Most classes of applications have enjoyed free and regular performance gains for several decades, even without releasing new versions or doing anything special, because the CPU manufacturers (primarily) and memory and disk manufacturers (secondarily) have reliably enabled ever-newer and ever-faster mainstream systems.

In order to get increased performance today and in the future, programmers must utilise parallel programming techniques.

9.1.2 The Memory System

A computer’s memory is the storage area where executing programs are kept. It also stores the data needed by running programs. Programs access memory in accordance with the principle of locality,

- **Temporal locality.** If a data location has recently been referenced it is likely to be referenced again soon. For example, program code that uses loops will repeatedly make reference to an index variable.

- **Spatial locality.** If a data location has been referenced then the nearby locations are also likely to be referenced soon. For example, program code to sum a list of numbers requires sequential access of each element, exhibiting a high degree of spatial locality.

Given the common access patterns of typical programs, the memory system of a computer is arranged in a memory hierarchy (see figure 9.1). The fastest (but smallest) memories at the top of the hierarchy are located closest to the CPU. Moving down the hierarchy, the slower (but higher capacity) memories are situated further away from the
CPU, corresponding to slower access times. The fastest programs and algorithms seek to avoid accessing memory in the lower hierarchies.

![Memory Hierarchy Diagram](image)

**Figure 9.1: Memory Hierarchy**

In order to decrease memory access times, hardware designers make use of *caches*, a fast memory which stores duplicate data to reduce access to the slower memories in the lower hierarchy.

![Cache Example](image)

**Figure 9.2: An example of a simple cache. The arrows designate the memory addresses currently duplicated by the cache. Accessing memory address 0100 results in a cache hit (coloured green). Accessing address 0101 results in a cache miss (coloured red).**
Figure 9.2 above gives a simplified example of a cache. Each row of the cache is called a block. Suppose that a program wishes to read the data in address 0100 of main memory containing data element \( a \). The address is decomposed as

\[
\begin{array}{c|c}
\text{tag} & \text{cache index} \\
01 & 00 \\
\end{array}
\]

The two lower order bits of the address are 00, corresponding to the first row of the cache (index 00). The two higher order bits are 01, which is called the tag. For this example, a request of the data at address 0100 produces a cache hit, since a copy of the data is found within the cache. The contents of the data is then copied from the cache to the processor. This is the ideal situation, as accessing from the cache is very fast.

Suppose now that a program wishes to read the data in address 0101 of main memory. The hardware first checks that the data is stored in the cache. The two lower order bits and the tag are both 01. But the tag at index 01 in the cache is 10, and so the requested address is not found in the cache. This situation is called a cache miss.

To complete the request, the hardware will have to access the address directly in main memory, update the cache block at index 01 and deliver the data back to the processor. The time that it takes to complete these steps is called the miss penalty.

It is a top priority of hardware manufacturers to optimise their memory hierarchies in order to minimise the number of cache misses and the duration of miss penalties. The details are beyond the scope of this thesis (refer to chapter 5 of [48] for more information).

9.2 Parallel architectures

9.2.1 Flynn’s taxonomy

A useful way to classify different parallel architectures is to look at the number of instruction streams and data streams. This is called Flynn’s taxonomy [13]

- SISD (single instruction, single data stream) This architecture does not take advantage of instruction or data parallelism. The most well known example of SISD is the single core CPU found in older mainstream PCs.
- MISD (multiple instruction, single data stream) This architecture takes advantage of instruction parallelism only. It is hardly used in practice.
- SIMD (single instruction, multiple data stream) This architecture has multiple processors executing the same instructions on different data.
- MIMD (multiple instruction, multiple data stream) Here the architecture consists of multiple independent processors executing separate instructions on different data.

9.2.2 Shared and distributed memory MIMD architectures

MIMD architectures can be further broken down into two main categories which differ based on how the processors access memory:

**Shared memory architecture**

For this category, the processors share the same physical memory. Data in the shared memory is accessible to all processors. As a result, synchronisation between processors must be enforced. Otherwise, a data race or deadlock might occur. Modern CPUs from the mainstream manufacturers Intel and AMD are of this type. Figure 9.3 depicts an example of a shared memory architecture with two processors.

![Shared memory microprocessor with two processors](image)

**Message passing architecture**

Instead of having processors share access to a common memory, message passing multiprocessors equip each processor with its own private memory space. Coordination between processors is performed via *message passing*. The most common example of a message
passing system is the *cluster*. A cluster is a collection of regular computers connected via standard networking hardware.

9.3 Concepts of parallel programming

Parallel computing is a form of computation where instructions are executed simultaneously. To aid our discussion, we first present a couple of terms taken from [41]. A *task* is a sequence of instructions that is performed as a group. For example, in our Monte Carlo simulations discussed in chapter 10, we will define a task to be the simulation of a single realisation of a fBm trajectory and the calculation of the required functional. We then map the task to a *unit of execution (UE)*, an entity that will execute the instructions within a task. In a parallel setting, a collection of UEs will execute tasks concurrently.

UEs are executed by *processing elements (PE)*, which is the hardware element that physically executes the instructions.

Parallel programming is difficult for a number of reasons. We discuss a couple of issues below.

9.3.1 Load balancing

It is the job of a programmer to map tasks to the units of execution, and then map the units of execution to the processing elements of the hardware architecture. Spreading the mappings evenly across the PEs is called *load balancing*. The goal is to avoid having a small portion of the PEs doing all the work while the others remain idle, as we want to utilise as much of the available computing power as possible.

9.3.2 Synchronisation

When executing a parallel program, the order in which a collection of tasks are executed are often nondeterministic. For many algorithms, the ordering of the tasks is crucial and so *synchronisation* must be enforced. For example, suppose that an algorithm is made up of three tasks, A, B and C. Suppose that tasks B and C can be executed independently of each other, but that both tasks depend on the result of task A. A synchronisation mechanism must be used to ensure that task A completes execution before tasks B and C start.
A type of error that sometime arises from incorrect synchronisation is the *deadlock*. This occurs when two tasks both depend on the output of the other to proceed. For example, suppose we have two tasks $A$ and $B$. Task $A$ requires task $B$ to complete execution and similarly task $B$ requires task $A$ to complete. Because both tasks are waiting for each other, we have a block that cannot be resolved. Fortunately it is relatively easy to locate deadlocks because the program will halt execution at the point of the deadlock.

Another type of error which is much harder to deal with is the *race condition*. It occurs when the output of a program changes when the ordering of the tasks change. Because the collection of tasks are executed in a nondeterministic fashion, it is very hard to reproduce the error.

### 9.4 Common approaches to parallel programming

Here we describe the two main approaches to enable parallel execution which correspond to the two categories of MIMD architectures described in section 9.2.2.

A running computer program is called a *process*. Each process has its own allocation of memory (system state) with which it can operate. The simplest process is one which executes statements one after another in a sequential fashion. A major responsibility of a computer operating system is scheduling the work of a large number of processes.

A process is able to create *subprocesses* which all have their own system state and execute independently of each other. However these processes can communicate with each other by sending messages. This parallel programming approach is called *message passing* or *multiprocessing*. This approach is highly scalable as a process can reside on another CPU in another machine. Communication between processes then passes through networking hardware.

The other common approach is called *multithreading*. As the name suggests it relies on the creation and execution of *threads*, which are similar to processes with one crucial difference: the threads created from a process *share* system state. This is the approach we take for our Monte Carlo simulations. Details of our implementation are given in chapter 10.
9.5 Amdahl’s law

The amount of speedup that can be achieved using parallel techniques depends on the proportion of the work that is parallelisable. Let $T_1$ denote the execution time of a sequential algorithm, and $T_N$ denote the execution time of an associated parallel algorithm with $N$ processors. Then the speedup achieved by the parallel algorithm is given by

$$S_N = \frac{T_1}{T_N} \quad (9.1)$$

The expected speedup of a parallel algorithm over an associated sequential algorithm can be quantified by using Amdahl’s law [48], which states that

$$S_N = \frac{1}{(1 - P) + \frac{P}{N}} \quad (9.2)$$

where $P$ denotes the proportion of the program that is parallelisable, and $N$ is the number of processors used by the parallel algorithm.

The law can be justified as follows: Given a proportion $P$ that is parallelisable, the running time for 1 processor can be decomposed as

$$T_1 = (1 - P)T_1 + PT_1$$

Now we make the assumption that if we are given $N$ processors, then the work on the parallel portion may be divided up equally amongst each processor. That is,

$$T_N = (1 - P)T_1 + \frac{PT_1}{N} \quad (9.3)$$

Substituting (9.3) into (9.1) and cancelling terms gives (9.2).

Note that if $P < 1$, taking the limit $N \to \infty$ in (9.2) gives us the maximum speedup possible:

$$S_\infty = \frac{1}{1 - P}$$
So for example, if 95% of a program were parallelisable, then we could achieve a speedup of at most 20, regardless of the number of processors available.

Monte Carlo simulation is an example of an *embarrassingly parallel* problem. This is because each simulation run is totally independent of the other runs. The proportion of the algorithm that is parallelisable is very close to 100%. This makes Monte Carlo simulation a perfect candidate for acceleration via parallel computation.
This chapter gives an overview of our parallel implementation for Monte Carlo schemes. Suppose we want to apply crude Monte Carlo to estimate \( \alpha = \mathbb{E}[f(Y)] \) for some given random variable \( Y \) and function \( f \). By generating an iid sequence of \( N \) random variables \( Y_0, \ldots, Y_{N-1} \), the crude Monte Carlo estimator of \( \alpha \) is given by \( \hat{\alpha}_N = \sum_{i=0}^{N-1} f(Y_i) \).

Suppose also that simulating \( Y_i \) is very slow (for example \( Y_i \) is a trajectory of fractional Brownian motion) such that even an efficient single-threaded implementation of the simulation algorithm is impractical. Then given a PC with a \( n \)-core processor, we can obtain a \( n \)-fold speedup of the simulation by dividing the work as follows:

- For each \( i \in \{0, 1, \ldots, n-2\} \) core \( i \) will generate realisations of
  
  \[
  f \left( Y_{i\left\lfloor \frac{N}{n} \right\rfloor} \right), f \left( Y_{i\left\lfloor \frac{N}{n} \right\rfloor+1} \right), \ldots, f \left( Y_{i+1\left\lfloor \frac{N}{n} \right\rfloor-1} \right)
  \]

- Core \( n-1 \) will generate realisations
  
  \[
  f \left( Y_{(n-1)\left\lfloor \frac{N}{n} \right\rfloor} \right), \ldots, f \left( Y_{N-1} \right)
  \]

Once all realisations have been generated, we can compute our sample statistics. See figure 10.1.

10.1 Simulation of realisations for problems 1 and 2

We used the circulant embedding method to simulate trajectories of fBm; see section 8.2.1.
To generate estimates for problem 1 we need to generate trajectories of fBm over the time interval \([0, T]\) for varying values of \(T\), and then compute the indicator function of the event \(\{\max_{0 \leq t \leq T} W_t^{(H)} < x\}\).

For problem 2 we use Riemann-sum approximations for the integrals so that

\[
var(\zeta_H) = \mathbb{E} \left[ \frac{\int_{-\infty}^{\infty} u e^{Y_u} du}{\int_{-\infty}^{\infty} e^{Y_u} du} \right]^2 \approx \mathbb{E} \left[ \frac{\sum(u_i e^{Y_{u_i} \Delta u})}{\sum(e^{Y_{u_i} \Delta u})} \right]^2
\]

where \(u_0 = -M, u_1 = -M + \Delta u, \ldots, u_n = M\) is a partition of \([-M, M]\), for large \(M \in \mathbb{R}^+\). Note that we need to generate two-sided fBm, but this is easy to overcome since fBm has stationary increments (theorem 8.1.1) so that

\[
W_{-M+i\Delta u}^{(H)} \overset{d}{=} W_{i\Delta u}^{(H)} - W_{M}^{(H)}.
\]

So we first generate a trajectory over \([0, 2M]\), then apply relation (10.1) to get the desired trajectory over \([-M, M]\).
10.2 Implementation details

We implemented our simulation codes in C++.

10.2.1 Third-party libraries used

A number of external third-party libraries were used:

- The **FFTW** library [16] was used to compute Fast Fourier transforms required to simulate fBm trajectories.
- We used the **Boost** library for threads and computation of sample statistics [1].
- A concurrent vector was used from the **Intel Threading Building Blocks** library [2] to allow each thread to concurrently store their results in a centralised data structure.
- Generation of pseudo-random numbers was handled by the **uniform random number generator** of Agner Fog [15]. In particular we used their multithreaded implementation of the well-known **Mersenne twister** generator [40].

10.2.2 Thread safety

When writing multithreaded programs the number one concern is to clearly distinguish the parts of the code are safe to execute concurrently in multiple threads and the parts that are not safe. This concept is called **thread safety**. A portion of code is **thread-safe** if there is a guarantee that the implementation is free of race conditions when it is accessed by multiple threads concurrently. Code is deemed **not thread-safe** if it cannot be accessed simultaneously by different threads.

In practice multiple threads still need to execute non-thread-safe code, and to ensure problems don’t arise the program must enforce mutually exclusive access to this code. A basic synchronisation primitive called a **mutex** can enforce this access. As an example, suppose that three threads A, B and C need to execute the following code which is potentially not thread-safe,

```cpp
{  
    lock_guard<mutex> mutex_lock(mutex);
    //Execute code...  
    //...
}
```
Within each of the three running threads the mutex_lock variable will attempt to acquire a lock. If thread B first acquires the lock then the code starting from line 3 can be executed. Whilst thread B holds the lock, then the mutex_lock variable in threads A and C must wait until thread B releases its lock when execution reaches line 6.

10.2.3 Overview of the main classes

The classes that estimate \( T^{1-H} P \left[ \max_{0 \leq t \leq T} B_t^{(H)} < 1 \right] \) and \( \text{Var}(\zeta) \) are ProbMaxFBMEstimator (section C.3.3) and FBMZetaIntegralEstimator (section C.3.2), respectively. Both classes implement the thread-safe method

\[
\text{void computeBlock(int const threadNum, int const left, int const right)}
\]

whose responsibility is to sequentially generate realisations for trials \( m, m+1, \ldots, n \), where \( m = \text{left} \) and \( n = \text{right} \).

Parallel execution is achieved by the WorkDistributor class (section C.3.4). After being told that \( N \) trials are to be generated with \( n \) threads of execution, the WorkDistributor class is given an instance of the estimator class and a pointer to its computeBlock method where the work is then divided between the \( n \) threads as shown in figure 10.1.

The following code excerpt (lines 27-33 of ProbMaxFBMEstimator.cpp) from the ProbMaxFBMEstimator class demonstrates this:

```cpp
27 double ProbMaxFBMEstimator::computeResult() {
28     int N = simulationData_.numTrajectories();
29     int n = boost::thread::hardware_concurrency(); // We set n = (# of cores)
30     WorkDistributor workDistrib(N, n);
31     workDistrib.allocateWorkToThreads(*this,
32         &ProbMaxFBMEstimator::computeBlock);
33     // Store results
34 }
```

The class that is responsible for generating fBm trajectories is FBMPathGenerator. It generates trajectories using the circulant embedding method of section 8.2.1. Each thread contains its own FBMPathGenerator object and all class methods are thread-safe. The code listing for the class is in section C.3.1. We note that the algorithm relies on the FFTW library and this library does not have thread-safe initialisation and shutdown functions. So we must synchronise access to these functions, and this is done by using a mutex. Given the following code excerpt from FBMPathGenerator.cpp (lines 52-58),

95
the mutex on line 53 ensures that the initialisation function `fftw_plan_dft_1d` executes within at most a single thread at any given time.

Lastly we note that concurrent storage of the trial results is handled by our class `FBMSimulationData` (section C.3.5). This is achieved by a concurrent data structure (`tbb::concurrent_vector`) provided by the Intel Threading Building Blocks library.
Chapter 11

Empirical results for certain functionals of fractional Brownian motion

Using a parallel version of the circulant embedding algorithm described in previous chapters, we present our simulated findings which shed light on problems 1 and 2. Each estimate was computed from 500,000 fBm trajectories, each of which was generated over $2^{18} \approx 262,000$ evenly-spaced time points. A parallel implementation was essential:

- On a PC with an Intel Core2Duo (2-core processor), each estimate takes approximately 30 hours to compute.
- On an AMD Phenom II 6-core processor, the same estimate takes about 7 hours.
- The findings presented in this chapter was the result of 40 days of continual computing, with all CPU cores utilised.

11.1 Results for problem 1

Here we simulated $f_H(T) := T^{1-H} P\left[\max_{0 \leq t \leq T} W_t^{(H)} < 1\right]$ for $T = 1000, 2000, \ldots, 30000$ and for $H = 0.4, 0.75$ and 0.95. The results are shown in figures 11.1, 11.2 and 11.3. Some standard error values are reported in table 11.1. The standard error is proportional to $T$.

<table>
<thead>
<tr>
<th>$T$</th>
<th>$H = 0.4$</th>
<th>$H = 0.75$</th>
<th>$H = 0.95$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.007</td>
<td>0.0017</td>
<td>0.00065</td>
</tr>
<tr>
<td>2000</td>
<td>0.009</td>
<td>0.0018</td>
<td>0.00066</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>30000</td>
<td>0.024</td>
<td>0.0027</td>
<td>0.00072</td>
</tr>
</tbody>
</table>

Table 11.1: Standard errors for $T^{1-H} P\left(\max_{0 \leq t \leq T} W_t^{(H)} < 1\right)$
Figure 11.1: \( T^{1-H} \mathbb{P} \left( \max_{0 \leq t \leq T} W_t^{(H)} < 1 \right) \) for \( H = 0.40 \).

Figure 11.2: \( T^{1-H} \mathbb{P} \left( \max_{0 \leq t \leq T} W_t^{(H)} < 1 \right) \) for \( H = 0.75 \).

11.2 Results for problem 2

Here we computed estimates for

\[
\text{var}(\zeta_H) = \mathbb{E} \left[ \frac{\int_{-\infty}^{\infty} u e^{Y_u} du}{\int_{-\infty}^{\infty} e^{Y_u} du} \right]^2
\]

The results are shown in figure 11.4.
Figure 11.3: \(T^{1-H} \mathbb{P}\left( \max_{0 \leq t \leq T} W_t^{(H)} < 1 \right) \) for \( H = 0.95 \).

Figure 11.4: \( \text{var}(\zeta_H) \) for a range of \( H \) values. Region of integration was truncated to \([-10000,10000]\).

11.3 Conjectures

From the results just presented we conjecture
Conjecture 11.3.1. For $H \geq \frac{1}{2}$,

$$\lim_{T \to \infty} T^{1-H} \mathbb{P}\left[\max_{0 \leq t \leq T} W_t^{(H)} < 1\right] = \text{const}$$

Conjecture 11.3.2. Regarding problem 2,

$$\lim_{H \to 0} \text{var}(\zeta_H) = \infty$$
Part III

Appendices


APPENDIX A

VWAP calculations

A.1 Calculations for model 1

**Theorem A.1.1.** Let $A, B, C, D, F$ and $G$ be real constants such that $A > 0$, $B > 0$ and $F^2 < 4AB$. Then

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left\{ -Ax^2 - By^2 + Cx + Dy + Fxy + G \right\} \, dx \, dy
= 2\pi \exp \left\{ \frac{BC^2 + D(AD + CF)}{4AB - F^2 + G} \right\} (4AB - F^2)^{1/2}$$

*Proof.* First note that

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left\{ -Ax^2 - By^2 + Cx + Dy + Fxy + G \right\} \, dx \, dy
= e^G \int_{-\infty}^{\infty} \exp \left\{ -By^2 + Dy \right\} \left( \int_{-\infty}^{\infty} \exp \left\{ -Ax^2(C + Fy) \right\} \, dx \right) \, dy \tag{A.1}$$

Treating $y$ as a constant in the inner integral and completing the square, we get

$$\int_{-\infty}^{\infty} \exp \left\{ -Ax^2(C + Fy) \right\} \, dx
= \exp \left\{ \frac{(C + Fy)^2}{4A} \right\} \sqrt{\frac{2\pi}{2A}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{\frac{2\pi}{2A}}} \exp \left\{ -\frac{(x - \frac{C + Fy}{2A})^2}{2\frac{1}{2A}} \right\} \, dx
= \exp \left\{ \frac{(C + Fy)^2}{4A} \right\} \sqrt{\frac{\pi}{A}} \tag{A.2}$$
Substituting back, collecting terms and completing the square again gives us

\[
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left\{ -Ax^2 - By^2 + Cx + Dy + Fxy + G \right\} \, dx \, dy
\]

\[
= \sqrt{\pi} \exp \left\{ \frac{C^2}{4A} + G - \frac{4AB - F^2}{4A} \times \left( \frac{CF + 2AD}{4AB - F^2} \right)^2 \right\}
\]

\[
\int_{-\infty}^{\infty} \exp \left\{ -\frac{(4AB - F^2)}{4A} \left( y - \frac{CF + 2AD}{4AB - F^2} \right)^2 \right\} \, dy
\]

\[
= \sqrt{\pi} \exp \left\{ \frac{C^2}{4A} + G - \frac{4AB - F^2}{4A} \times \left( \frac{CF + 2AD}{4AB - F^2} \right)^2 \right\}
\]

\[
= 2\pi \exp \left\{ \frac{BC^2 + D(AD + CF)}{4AB - F^2} + G \right\} (4AB - F^2)^{-1/2}
\]

The following result is needed in the derivation of the Laplace transform in theorem 4.2.3

**Lemma A.1.2.** Let \( Y_t \) denote the standard Ornstein-Uhlenbeck process with parameter \( \kappa \) i.e. \( Y_t \) satisfies \( dY_t = -\kappa Y_t \, dt + dW_t, \ Y_0 = 0 \). Set \( Z_1 = a\sigma \int_t^T Y_s \, ds, Z_2 = Y_T, Z_3 = Y_t \) and \( \sigma_{ij} = \text{Cov}(Z_i, Z_j) \) for \( i, j \in \{1, 2, 3\} \). Then

\[
\sigma_{11} = a^2 \int_t^T \left( \int_t^s \frac{1}{2\kappa} e^{-\kappa(u-s)} (e^{2\kappa u} - 1) \, du + \int_s^T \frac{1}{2\kappa} e^{-\kappa(u-s)} (e^{2\kappa s} - 1) \, du \right) \, ds
\]

\[
\sigma_{22} = \frac{1}{2\kappa} (1 - e^{-2\kappa T})
\]

\[
\sigma_{33} = \frac{1}{2\kappa} (1 - e^{-2\kappa t})
\]

\[
\sigma_{12} = \int_t^T \frac{a}{2\kappa} e^{-\kappa(s+t)} (e^{2\kappa s} - 1) \, ds
\]

\[
\sigma_{13} = \int_t^T \frac{a}{2\kappa} e^{-\kappa(s+t)} (e^{2\kappa t} - 1) \, ds
\]

\[
\sigma_{23} = \frac{1}{2\kappa} e^{-\kappa(t+T)} (e^{2\kappa t} - 1)
\]

Note that \( \sigma_{11}, \sigma_{12} \) and \( \sigma_{13} \) can be explicitly computed with a symbolic algebra package.
Proof. Note by theorem 2.4.5 that $\mathbb{E}[Y_t] = 0$. Using Fubini’s theorem this implies that $\mathbb{E}\left[a \int_t^T Y_s ds\right] = 0$. Now consider

$$\sigma_{11} = \text{Cov}\left(a \int_t^T Y_s ds, a \int_t^T Y_s ds\right)$$

$$= \mathbb{E}\left(a^2 \int_t^T \int_t^T Y_s Y_u duds\right)$$

$$= a^2 \int_t^T \left(\int_t^s \mathbb{E}(Y_s Y_u) du + \int_s^T \mathbb{E}(Y_s Y_u) du\right) ds,$$ by Fubini theorem

$$= a^2 \int_t^T \left(\int_t^s \frac{1}{2\kappa} e^{-\kappa(u+s)}(e^{2\kappa u} - 1) du + \int_s^T \frac{1}{2\kappa} e^{-\kappa(u+s)}(e^{2\kappa s} - 1) du\right) ds$$

The remaining covariances are proved in a similar way. \qed
APPENDIX B

Extra mathematical results

B.1 Some results from mathematical analysis

Theorem B.1.1. Let $X$ be a subspace of a metric space $(Z, d_1)$, and let $(Y, d_2)$ denote a complete metric space. If the function $f : X \to Y$ is uniformly continuous, then there exists a unique, uniformly continuous extension $\tilde{f} : \bar{X} \to Y$.

Proof. See [55] for a proof.

B.2 Some more results from measure theory

Theorem B.2.1. (Integral projection) Let $(\Omega, \mathcal{F})$ and $(\Omega', \mathcal{F}')$ be measure spaces, $\mu$ a measure on $(\Omega, \mathcal{F})$ and $X : \Omega \to \Omega'$ a measurable map. Then for all measurable $f : \Omega' \to \mathbb{R}$, $f \circ X$ is $\mu$-integrable if and only if $f$ is $X(\mu)$-integrable, in which case

$$\int_{\Omega} f \circ X d\mu = \int_{\Omega'} f dX(\mu)$$

Proof. See for e.g. [54].

B.3 Some more results from probability theory

For this section we let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space.
Theorem B.3.1. \((L^p \text{ convergence of conditional expectation})\) Let \(p \in [1, \infty)\) and \(\mathcal{G} \subset \mathcal{F}\) be a sub-\(\sigma\)-algebra. Given a sequence of random variables \((X_n)_{n \in \mathbb{N}}\) in \(L^p(\Omega, \mathcal{F}, \mathbb{P})\) such that \(\|X_n - X\|_p \xrightarrow{n \to \infty} 0\) for some \(X \in L^p(\Omega, \mathcal{F}, \mathbb{P})\) we have

\[
\| \mathbb{E}[X_n | \mathcal{G}] - \mathbb{E}[X | \mathcal{G}] \|_p \xrightarrow{n \to \infty} 0
\]

**Proof.** Note that \(f(x) = |x|^p\) is a convex function. Consider

\[
\| \mathbb{E}[X | \mathcal{G}] \|_p^p = \mathbb{E}[(\mathbb{E}[X | \mathcal{G}])^p]
\]

\[
= \mathbb{E}[f(\mathbb{E}[X | \mathcal{G}])]
\]

\[
\leq \mathbb{E}[f(X) | \mathcal{G}] \quad \text{by Jensen’s inequality}
\]

\[
= \mathbb{E}[|X|^p]
\]

\[
= \|X\|^p_p
\]

So \(X \mapsto \mathbb{E}[X | \mathcal{G}]\) is a contraction mapping, and it is also linear. We now show that it is continuous with respect to the norm topologies on \(L^p(\Omega, \mathcal{F}, \mathbb{P})\). Let \(\epsilon > 0\). Then define \(\delta = \epsilon\) and let \(\tilde{Y} \in L^p(\Omega, \mathcal{F}, \mathbb{P})\) such that \(\|X - \tilde{Y}\|_p < \delta\). Consider

\[
\| \mathbb{E}[X | \mathcal{G}] - \mathbb{E}[Y | \mathcal{G}] \|_p = \| \mathbb{E}[X - Y | \mathcal{G}] \|_p \quad \text{by linearity}
\]

\[
\leq \|X - Y\|_p
\]

\[
< \delta = \epsilon
\]

This proves continuity and the result follows. \(\square\)

Theorem B.3.2. Let \(p \in (1, \infty)\), \(\hat{X}\) be an \(L^p\)-integrable process and \((X^n)_{n \in \mathbb{N}}\) be a sequence of \(L^p\)-integrable, continuous martingales such that for any \(t \in [0, \infty)\), \(X^n_t \xrightarrow{n \to \infty} \hat{X}_t\) in \(L^p\). Then \(\hat{X}\) is a martingale and there exists a continuous modification \(X\) of \(\hat{X}\) such that \(X\) is a martingale with \(X^n_t \xrightarrow{n \to \infty} X_t\) in \(L^p\).
Proof. Let \( t, s \in [0, \infty), s \leq t \). By theorem B.3.1 \( L^p \) convergence of \( X^n_t \) implies \( \mathbb{E}[X^n_t|\mathcal{F}_s] \xrightarrow{n \to \infty} \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] \) in \( L^p \). That is, \( X^n_s \xrightarrow{n \to \infty} \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] \) in \( L^p \) since \( X^n \) is a martingale. Now consider

\[
\| \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] - \tilde{X}_s \|_p = \| \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] - X^n_s + X^n_s - \tilde{X}_s \|_p \\
\leq \| \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] - X^n_s \|_p + \| X^n_s - \tilde{X}_s \|_p
\]

Taking limit \( n \to \infty \) on both sides it follows that \( \| \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] - \tilde{X}_s \|_p = 0 \) and so \( \mathbb{E}[\tilde{X}_t|\mathcal{F}_s] = \tilde{X}_s \) \( \mathbb{P} \)-almost surely. So \( \tilde{X} \) is a martingale.

Now choose any \( \epsilon > 0 \) and consider

\[
\mathbb{P}\left[ \sup_{t \in \mathbb{R}^+} |X^n_t - \tilde{X}_t| > \epsilon \right] \leq \frac{\mathbb{E}\left[ \sup_{t \in \mathbb{R}^+} |X^n_t - \tilde{X}_t| \right]^p}{\epsilon^p} \text{ by Markov inequality} \\
\leq \left( \frac{p}{p-1} \right)^p \sup_{t \in \mathbb{R}^+} \mathbb{E}\left[ |X^n_t - \tilde{X}_t| \right]^p \text{ by Doob's inequality}
\]

since \( \lim_{n \to \infty} X^n_t = \tilde{X}_t \) in \( L^p \). So \( \lim_{n \to \infty} \mathbb{P}\left[ \sup_{t \in \mathbb{R}^+} |X^n_t - \tilde{X}_t| = 0 \right] = 1 \). Define for each \( n \in \mathbb{N} \)
\[
A_n = \left\{ \sup_{t \in \mathbb{R}^+} |X^n_t - \tilde{X}_t| = 0 \right\}.
\]

Then we can choose a subsequence \( (X_{nm})_{m \in \mathbb{N}} \) such that \( A_{nm} \uparrow \bigcup_{m=1}^{\infty} A_{nm} \). By monotone limits of measures, it follows that \( \mathbb{P}\left[ \bigcup_{m=1}^{\infty} A_{nm} \right] = 1 \). That is, the sequence of continuous martingales \( (X_{nm})_{m \in \mathbb{N}} \) converges uniformly to \( \tilde{X} \) almost surely. Recall that the uniform limit of continuous functions is also continuous. The existence of our required modification follows. \( \square \)

We now want to show that the mapping \( I_\infty \) defined in definition 2.3.3 is linear. We will use the following lemma,

**Lemma B.3.3.** Let \( H_t(\omega) = \sum_{i=1}^{m} h_{i-1}(\omega) 1_{(s_{i-1},s_i]}(t) \) be an element of \( \mathcal{E}_0 \). Let \( t_1 \) and \( t_2 \) be strictly positive real numbers such that \( t_1 < t_2 \), and \( g_1 : \Omega \to \mathbb{R} \) be bounded and
$\mathcal{F}_{t_1}$-measurable. Then $H + g_1 1_{(t_1, t_2]} \in \mathcal{E}_0$ and

$$I_\infty (H + g_1 1_{(t_1, t_2]}) = I_\infty (H) + I_\infty (g_1 1_{(t_1, t_2]}) = I_\infty (H) + g_1 (W_{t_2} - W_{t_1})$$

Proof. First set $G_t (\omega) = g_1 1_{(t_1, t_2]}(t)$ and note that $G \in \mathcal{E}_0$ since $G_t (\omega) = 0 \times 1_{(0, t_1]} + g_1 1_{(t_1, t_2]}$ and any constant function is of course measurable. There are now four cases to consider.

Case 1: If $s_m < t_1$ then

$$H_t (\omega) + G_t (\omega) = \sum_{i=1}^{m} h_{i-1} (\omega) 1_{(s_{i-1}, s_i]}(t) + 0 \times 1_{(s_m, t_1]} + g_1 (\omega) 1_{(t_1, t_2]}(t) \in \mathcal{E}_0$$

and clearly

$$I_\infty (H + G) = I_\infty (H) + I_\infty (G) = I_\infty (H) + g_1 (W_{t_2} - W_{t_1})$$

Case 2: If $s_{p-1} < t_1 < t_2 \leq s_p$ for some $p \in \{1, \cdots, m\}$ then if in particular $p < m$

$$H_t (\omega) + G_t (\omega) = \sum_{i=1}^{p-1} (h_{i-1} (\omega) + g_0 (\omega)) 1_{(s_{i-1}, s_i]}(t) + h_{p-1} (\omega) 1_{(s_{p-1}, t_1]}(t)$$

$$+ (h_{p-1} (\omega) + g_1 (\omega)) 1_{(t_1, t_2]}(t) + h_{p-1} (\omega) 1_{(t_1, s_p]}(t) + \sum_{i=p+1}^{m} h_{i-1} (\omega) 1_{(s_{i-1}, s_i]}(t)$$

$$\in \mathcal{E}_0$$

If $p = m$ then we omit the last summation above. After some calculations we get

$$I_\infty (H + G) = \sum_{i=1}^{m} h_{i-1} (W_{t_i} - W_{t_{i-1}}) + g_1 (W_{t_2} - W_{t_1})$$

$$= I_\infty (H) + I_\infty (G)$$
Case 3: If \( t_1 \leq s_p < t_2 \) for some \( p \in \{1, \ldots, m\} \) then

\[
H_t(\omega) + G_t(\omega) = \sum_{i=1}^{p-1} (h_{i-1}(\omega) + g_0(\omega)) \mathbb{1}_{(s_{i-1}, s_i]}(t) + h_{p-1}(\omega) \mathbb{1}_{(s_{p-1}, t_1]}(t)
\]

\[+ (h_{p-1}(\omega) + g_0(\omega)) \mathbb{1}_{(t_1, s_p]}(t) + (h_p(\omega) + g_0(\omega)) \mathbb{1}_{(s_p, t_2]}(t)\]

\[+ h_p(\omega) \mathbb{1}_{(t_2, s_{p+1}]}(t) + \sum_{i=p+2}^{m} h_{i-1}(\omega) \mathbb{1}_{(s_{i-1}, s_i]}(t)\]

If \( p = m - 1 \), we omit the final summation and if \( p = m \) then omit the last line above. Result then follows.

Case 4: If \( t_1 \leq s_p < \cdots < s_{p+r} \leq t_2 \) for some \( p \in \{1, \ldots, m-1\} \) and \( r \in \{1, \ldots, m-p\} \), then the result follows using the same analysis as above. \( \square \)

**Theorem B.3.4.** The mapping \( I_\infty : \mathcal{E}_0 \to L^2(\Omega, \mathcal{F}, \mathbb{P}) \) defined in definition 2.3.3 is linear.

**Proof.** Let \( G_t(\omega) = \sum_{i=1}^{n} g_{i-1}(\omega) \mathbb{1}_{(t_{i-1}, t_i]}(t) \) and \( H_t(\omega) = \sum_{i=1}^{m} h_{i-1}(\omega) \mathbb{1}_{(s_{i-1}, s_i]}(t) \) be elements of \( \mathcal{E}_0 \). We will prove the result by induction. Suppose that \( n = 1 \). Then \( G_t(\omega) = g_0(\omega) \mathbb{1}_{(0, t_1]}(t) \) and the result follows from lemma B.3.3.

Now suppose that the result holds for \( n = k \) where \( k \geq 1 \). Let

\[
G_t(\omega) = \sum_{i=1}^{k+1} g_{i-1}(\omega) \mathbb{1}_{(t_{i-1}, t_i]}(t)
\]

be an element of \( \mathcal{E}_0 \). Then again applying lemma B.3.3 we have

\[
I_\infty (G + H) = I_\infty \left( \sum_{i=1}^{k} g_{i-1}(\omega) \mathbb{1}_{(t_{i-1}, t_i]}(t) + H_t(\omega) + g_k(\omega) \mathbb{1}_{(t_k, t_{k+1}]}(t) \right)
\]

\[= I_\infty \left( \sum_{i=1}^{k} g_{i-1}(\omega) \mathbb{1}_{(t_{i-1}, t_i]}(t) + H_t(\omega) \right) + g_k(\omega) (W_{t_{k+1}} - W_{t_k}) \]

\[= \sum_{i=1}^{k} g_{i-1}(\omega) (W_{t_{i-1}} - W_{t_i}) + I_\infty (H) + g_k(\omega) (W_{t_{k+1}} - W_{t_k}) \]

\[= I_\infty (G) + I_\infty (H)\]

109
where the third equality follows from the inductive hypothesis.
Appendix C

Code listings

C.1 VWAP First Moment

Needs ["MultivariateStatistics"]

\[ a_1 = 2\pi \sqrt{-\sigma_{12}^2 + \sigma_{11} \sigma_{22} - \frac{\sigma_{13}^2 \sigma_{22}}{\sigma_{33}} + \frac{2\sigma_{12} \sigma_{13} \sigma_{23}}{\sigma_{33}} - \frac{\sigma_{11} \sigma_{23}^2}{\sigma_{33}}}; \]

MeanVector = \left\{ \begin{array}{l}
\mu_{x1} + \frac{\sigma_{13}}{\sigma_{33}} (z - \mu_{x3}) ; \\
\mu_{x2} + \frac{\sigma_{23}}{\sigma_{33}} (z - \mu_{x3}) 
\end{array} \right\};

CovMatrix = \left\{ \begin{array}{l}
\left\{ \begin{array}{l}
\sigma_{11} - \frac{\sigma_{13}^2}{\sigma_{33}} , \sigma_{12} - \frac{\sigma_{13} \sigma_{23}}{\sigma_{33}} \\
\sigma_{12} - \frac{\sigma_{13} \sigma_{23}}{\sigma_{33}} , \sigma_{11} - \frac{\sigma_{23}^2}{\sigma_{33}} 
\end{array} \right\} , \\
\left\{ \begin{array}{l}
\sigma_{12} - \frac{\sigma_{13} \sigma_{23}}{\sigma_{33}} , \sigma_{11} - \frac{\sigma_{23}^2}{\sigma_{33}} \\
\sigma_{11} - \frac{\sigma_{13} \sigma_{23}}{\sigma_{33}} , \sigma_{22}\sigma_{33} - \sigma_{13} \sigma_{23} \sigma_{33} 
\end{array} \right\} 
\right\};

\text{expon}[x_-, y_] = a_1 \text{PDF}[\text{MultinormalDistribution}[\text{MeanVector}, \text{CovMatrix}, \{x, y\}]] - 2q\sigma x - \frac{(\kappa + \lambda) y^2}{2};

G = \text{FullSimplify}[\text{expon}[0, 0]]; 
F = \text{FullSimplify}[\text{Coefficient}[\text{expon}[x, y], xy]];
A = -\text{FullSimplify}[\text{Coefficient}[\text{expon}[x, 0], x^2]];
B = \text{FullSimplify}[\text{Coefficient}[\text{expon}[0, y], y^2]];
C_x = \text{FullSimplify}[\text{Coefficient}[\text{expon}[x, 0], x]];
D_y = \text{FullSimplify}[\text{Coefficient}[\text{expon}[0, y], y]];

\text{JYt} = \text{FullSimplify}[\text{Coefficient}\left[\frac{B(C_x)^2 - D_y(AD_y + C_x F)}{4AB + F^2} + G, z\right]];
\text{HYt2} = \text{FullSimplify}[\text{Coefficient}\left[\frac{B(C_x)^2 - D_y(AD_y + C_x F)}{4AB + F^2} + G, z^2\right]];

\text{constZ} = \frac{B(C_x)^2 - D_y(AD_y + C_x F)}{4AB + F^2} + G;
\[ L = \text{FullSimplify}[\text{constZ}[0]]; \]
\[ OC1 = \text{FullSimplify}[\frac{1}{\sqrt{A} \sqrt{-\frac{4AB + F^2}{A}}}]; \]
\[ a_2 = 2\pi \sqrt{-\sigma_{ab}^2 + \sigma_{aa} \sigma_{bb}} \]
\[ \text{expon1}[x, y] = a_2 \text{PDF}[\text{MultinormalDistribution}[\{\mu_a, \mu_b\}, \{(\sigma_{aa}, \sigma_{ab}), (\sigma_{ab}, \sigma_{bb})\}, \{(x, y)\}]] + c_1 x + c_2 x^2 + c_3 y; \]
\[ G1 = \text{FullSimplify}[\text{expon1}[0, 0]]; \]
\[ F1 = \text{FullSimplify}[\text{Coefficient}[\text{expon1}[x, y], xy]]; \]
\[ A1 = -\text{FullSimplify}[\text{Coefficient}[\text{expon1}[x, 0], x^2]]; \]
\[ B1 = \text{FullSimplify}[\text{Coefficient}[\text{expon1}[0, y], y^2]]; \]
\[ C_{x1} = \text{FullSimplify}[\text{Coefficient}[\text{expon1}[x, 0], x]]; \]
\[ D_{y1} = \text{FullSimplify}[\text{Coefficient}[\text{expon1}[0, y], y]]; \]
\[ OC2 = \text{FullSimplify}[\frac{1}{\sqrt{A} \sqrt{-\frac{4AB + F^2}{A}}}]; \]
\[ \kappa = \sqrt{\lambda^2 + 2q\sigma^2}; \]
\[ J = JYt; \]
\[ H = HYt2; \]
\[ c_1 = -2z\sigma + J; \]
\[ c_2 = -z\sigma^2 + H; \]
\[ c_3 = -2q\sigma; \]
\[ \gamma[z, q] = OC1 \ast OC2 \ast \text{Exp}[L] \ast \text{Exp}[\frac{B1(C_{x1})^2 - D_{y1}(A1D_{y1} + C_{x1}F1)}{4A1B1 + F1^2} + G1]; \]
\[ m[t] = a; \]
\[\sigma_{22} = \frac{1}{2\kappa} (1 - \text{Exp}[-2\kappa T]) ; \]
\[\sigma_{33} = \frac{1}{2\kappa} (1 - \text{Exp}[-2\kappa t]) ; \]
\[\sigma_{12} = \int_t^T m[s] \left( \frac{1}{2\kappa} \text{Exp}[-\kappa(s + t)] (\text{Exp}[2\kappa s] - 1) \right) ds ; \]
\[\sigma_{13} = \int_t^T m[s] \left( \frac{1}{2\kappa} \text{Exp}[-\kappa(s + t)] (\text{Exp}[2\kappa t] - 1) \right) ds ; \]
\[\sigma_{23} = \frac{1}{2\kappa} (\text{Exp}[\kappa(t - T)] - \text{Exp}[-\kappa(t + T)]) ; \]
\[\sigma_{aa} = \frac{1}{2\kappa} (1 - \text{Exp}[-2\kappa t]) ; \]
\[\sigma_{ab} = \int_t^T m[s] \left( \frac{1}{2\kappa} \text{Exp}[-\kappa(s + t)] (\text{Exp}[2\kappa s] - 1) \right) ds \]

\[S0 = 20; \sigma = 0.6; T = 1; \lambda = 0.5; b = 100.1; a = 30; x = 195; \mu = -0.94; \]
\[\sigma_{GBM} = 15.9; \mu_{x1} = 0; \mu_{x2} = 0; \mu_{x3} = 0; \mu_a = 0; \mu_b = 0; \]
\[\Phi[z_, q_] = \text{Exp} \left[ -za^2 - qa^2T + \frac{(\lambda + \kappa)T}{2} - zb - qbT \right] \ast \text{gamma}[z, q]; \]
\[dz\Phi[q_, t_] = \text{Derive}[1, 0][\Phi[z, q]] / . z \rightarrow 0; \]
\[\text{NIntegrate}[-S0\text{Exp}[\mu t]dz\Phi[q, t], \{q, 0, \infty\} , \{t, 0, T\}] \]

**C.2 VWAP Monte Carlo C++ code**

```cpp
#include <iostream>
#include <fstream>
#include <ctime>
#include <cmath>
#include <vector>
#include <algorithm>
#include <stdio.h>
#include <boost/timer.hpp>
#include <boost/accumulators/accumulators.hpp>
#include <boost/accumulators/statistics/stats.hpp>
```

\[\Phi[z_, q_] = \text{Exp} \left[ -za^2 - qa^2T + \frac{(\lambda + \kappa)T}{2} - zb - qbT \right] \ast \text{gamma}[z, q]; \]
\[dz\Phi[q_, t_] = \text{Derive}[1, 0][\Phi[z, q]] / . z \rightarrow 0; \]
\[\text{NIntegrate}[-S0\text{Exp}[\mu t]dz\Phi[q, t], \{q, 0, \infty\} , \{t, 0, T\}] \]

\[S0 = 20; \sigma = 0.6; T = 1; \lambda = 0.5; b = 100.1; a = 30; x = 195; \mu = -0.94; \]
\[\sigma_{GBM} = 15.9; \mu_{x1} = 0; \mu_{x2} = 0; \mu_{x3} = 0; \mu_a = 0; \mu_b = 0; \]
# include <boost/accumulators/statistics/mean.hpp>
# include <boost/accumulators/statistics/moment.hpp>
# include <boost/accumulators/statistics/variance.hpp>
# include <gsl/gsl_rng.h>
# include <gsl/gsl_randist.h>
# include <gsl/gsl_math.h>

using std::vector;
using std::cout;
using std::ofstream;
using namespace boost::accumulators;

class VWAPOption {
public:
    VWAPOption(gsl_rng * rng,
               int NPaths,
               int numIncrements,
               double T,
               double lambda,
               double a,
               double sigmaVol,
               double mu,
               double sigmaPrice,
               double X0,
               double S0,
               double K,
               double interestRate)
        : NPaths_(NPaths),
          numIncrements_(numIncrements),
          lambda_(lambda),
          a_1(a),
          sigmaVol_(sigmaVol),
          mu_(mu),
          sigmaPrice_(sigmaPrice),
          X0_(X0),
          S0_(S0),
          K_(K),
          interestRate_(interestRate) {

        delta_ = T/numIncrements;
        rng_ = rng;
    }

    void computeFirst2Moments(ofstream& file) {
        boost::timer t;
        accumulator_set<double.
stats<tag::moment<1>,
tag::variance(immediate) >> moment1;
accumulator_set<double,
stats<tag::moment<1>,
tag::variance(immediate) >> moment2;

// compute estimate for 1st moment
for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    moment1(VWAP_T);
}

// compute estimate for 2nd moment
for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    moment2(VWAP_T * VWAP_T);
}

cout << std::endl;
cout << "computation time = " << t.elapsed() << std::endl;
cout << "sigmaPrice = " << sigmaPrice_ << std::endl;
cout << "E(VWAP) = " << moment<1>(moment1) << std::endl;
cout << "E(VWAP^2) = " << sqrt(variance(moment1) / NPaths_) << std::endl;

file << sigmaPrice_" 
    << " , " << moment<1>(moment1) //E(VWAP) 
    << " , " << sqrt(variance(moment1) / NPaths_) //StdErr(VWAP) 
    << " , " << moment<1>(moment2) //E(VWAP^2) 
    << " , " << sqrt(variance(moment2) / NPaths_) //StdErr(VWAP^2) 
    << std::endl;
}

void computeMoment3(ofstream& file) {
boost::timer t;
accumulator_set<double, stats<tag::moment<3> >> moment3;

// compute estimate for 3rd moment
for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    moment3(VWAP_T);
}
cout << std::endl;
cout << "computation time = " << t.elapsed() << std::endl;
cout << "sigmaPrice = " << sigmaPrice_ << std::endl;
cout << "E(VWAP^3) = " << moment3(moment3) << std::endl;

file << sigmaPrice_ << " , " << moment3(moment3) / 3 << std::endl;

void computePrice(ofstream& file) {
  boost::timer t;
  accumulator_set<double, stats<tag::mean>,
          tag::variance(immediate) >> price;

  //compute estimate for VWAP price
  for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    if((VWAP_T - K_) > 0)
      price(exp(-interestRate_)*(VWAP_T - K_));
    else
      price(0.0);
  }
  cout << std::endl;
cout << "computation time = " << t.elapsed() << std::endl;
cout << "sigmaPrice = " << sigmaPrice_ << std::endl;
cout << "price = " << mean(price) << std::endl;
cout << "stdDevPrice = " << sqrt(variance(price)/NPaths_) << std::endl;

  file << sigmaPrice_ << " , " << mean(price) / option price
       << " , " << sqrt(variance(price)/NPaths_) << std::endl;
}

void computeMoment3AndPrice(ofstream& file) {
  boost::timer t;
  accumulator_set<double, stats<tag::moment<3>> > moment3;
  accumulator_set<double, stats<tag::mean>,
          tag::variance(immediate) >> price;
// compute estimate for 3rd moment
for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    moment3(VWAP_T);
}

// compute estimate for VWAP price
for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    if((VWAP_T - K_) > 0)
        price(VWAP_T - K_);
    else
        price(0.0);
}

} 

// print results

file << sigmaPrice_ 
    << " , ";
    << "moment <3>(moment3) //E(VWAP^3) 
    << " , ";
    << mean(price) //option price 
    << " , ";
    << sqrt(variance(price)) << std::endl;
}

void computeParameters(std::ofstream& file) {
boost::timer t;
for (int i = 0; i < NPaths_; ++i) {
    double VWAP_T = simulateSinglePath();
    VWAPMoments_(VWAP_T);
    if((VWAP_T - K_) > 0)
        callOptionPayoffs_(VWAP_T - K_);
    else
        callOptionPayoffs_(0.0);
}

}
cout << std::endl;
cout << "computation time = " << t.elapsed() << std::endl;
cout << "sigmaPrice = " << sigmaPrice << std::endl;
cout << "E(VWAP) = " << moment<1>(VWAPMoments) << std::endl;
cout << "E(VWAP^2) = " << moment<2>(VWAPMoments) << std::endl;
cout << "E(VWAP^3) = " << moment<3>(VWAPMoments) << std::endl;
cout << "price = " << mean(callOptionPayoffs) << std::endl;
cout << "stdDev = " << sqrt(variance(callOptionPayoffs)) << std::endl;
}

private:

// return a realisation of VWAP_T
double simulateSinglePath() {
    double sumStUt = S0_*X0_*X0_;
    double sumUt = X0_*X0_;
    double Sti = S0_;  
    double Xti = X0_;  
    double Uti = 0.0;
    const double muX = a_*(1-exp(-lambda_*delta_));
    const double sigmaX2 = pow(sigmaVol_,2)/(2*lambda_)*(1-exp(-2*lambda_*delta_));

    // simulate path over [0,T]
    for (int i = 1; i <= numIncrements_; i++) {
        Sti = Sti*exp((mu_-0.5*sigmaPrice_*sigmaPrice_)*delta_
                        + sigmaPrice_*sqrt(delta_)
                        *gsl_ran_gaussian_ziggurat (rng_,1.0));
        Xti = exp(-lambda_*delta_)*Xti + muX
                + sqrt(sigmaX2)*gsl_ran_gaussian_ziggurat (rng_,1.0);
        Uti = pow(Xti,2);
        sumStUt += Sti*Uti;
        sumUt += Uti;
    }
}
245 }  
246 return (sumStUt/sumUt);  
247 
248 gsl_rng * rng_;  
249 int NPPaths_;  
250 int numIncrements_;  
251 double lambda_;  
252     a_.  
253     sigmaVol_.  
254     mu_.  
255     sigmaPrice_.  
256     XO_.  
257     SO_.  
258     delta_.  
259     K_.  
260     interestRate_; /\textit{strike price}  
261 
262 accumulator_set<double,  
263     stats<tag::moment<1>,  
264     tag::moment<2>,  
265     tag::moment<3>> VWAPMoments_;  
266 accumulator_set<double,  
267     stats<tag::mean,  
268     tag::variance(immediate)>> callOptionPayoffs_;  
269 
270 #endif // VWAPOPTION_HPP_INCLUDED

C.3 Fractional Brownian Motion Monte Carlo C++ code

C.3.1 Code to generate fractional Brownian motion trajectories

FBMPathGenerator.hpp

1 #ifndef FBMPATHGENERATOR_H
2 #define FBMPATHGENERATOR_H
3 
4 #include <boost/shared_array.hpp>
5 #include <boost/shared_ptr.hpp>
6 #include "stocc.h"
7 #include "fftw3.h"
8 
9 //**************************************************************************/
10 // This class generates trajectories of fractional Brownian motion. 
11 // It is based on the Wood/Chan algorithm.
12 #endif // FBMPATHGENERATOR_H
// Length of the trajectory is of the form 2^m, for any given positive integer. The class and methods are all thread-safe
//****************************************************************************
class FBMPathGenerator {
public:
    FBMPathGenerator(void);

    FBMPathGenerator(const int n,
                     const double H,
                     const double L,
                     const int cum,
                     const int baseSeed,
                     const int threadNum);

    "FBMPathGenerator(void);

    // Compute and store a realisation of a fBm trajectory into the preallocated shared_array output (which is assumed to be of size 2^m).
    //
    //*****************************************************************************
    void computeTrajectory(boost::shared_array<double> output);

    // Returns 'n', where the number of generated points is 2^m
    long const lengthExponent();

    // Returns the Hurst exponent of the path
    double const H();

private:
    FBMPathGenerator(const FBMPathGenerator&);
    FBMPathGenerator& operator=(const FBMPathGenerator&);

    void init_(const int baseSeed);
    void computeEigenvector_(void);
    void computeSandT_();
    double covariance_(long i) {
        if (i == 0) return 1;
        else return (pow(i-1.2*H_)-2*pow(i.2*H_)+pow(i+1.2*H_))/2;
    }
    
    const int n_;
    double H_;
    const double L_;
    const int cum_;
    long m_;
const int threadNum_; 
boost::shared_ptr<fftw_complex> ev_;  // vector of eigenvalues 
boost::shared_ptr<fftw_complex> SPlusIT_; 
fftw_plan STPlan_; 
StochasticLib1 sto_; 
}; 
#endif 

FBMPathGenerator.cpp 
#include <boost/timer.hpp> 
#include <boost/thread.hpp> 
#include <cmath> 
#include <ctime> 
#include <cstdlib> 
#include <iostream> 
#include <fstream> 
#include <sstream> 
#include "FBMPathGenerator.h" 

using namespace std; 

boost::mutex mutex; 

FBMPathGenerator::FBMPathGenerator (void) 
: n_(0), H_(0), L_(0), cum_(0), sto_(0), threadNum_(0) { 
} 

FBMPathGenerator::FBMPathGenerator (const int n, 
const double H, 
const double L, 
const int cum, 
const int baseSeed, 
const int threadNum) 
: n_(n), H_(H), L_(L), cum_(cum), sto_(0), threadNum_(threadNum) { 
init_(baseSeed); 
} 

long const FBMPathGenerator::lengthExponent () { 
return n_; 
} 

double const FBMPathGenerator::H () { 
return H_; 
}
```cpp
void BMPathGenerator::init_(const int baseSeed) {
    m_ = (long)pow(2.0, n_ + 1);
    boost::shared_ptr<fftw_complex>
        ev((fftw_complex*)fftw_malloc(sizeof(fftw_complex) * m_),
            fftw_free),
        SPlusIT((fftw_complex*)fftw_malloc(sizeof(fftw_complex) * m_),
            fftw_free);
    ev_ = ev;
    SPlusIT_ = SPlusIT;
    computeEigenvector_();
    int seedArray[] = {baseSeed, threadNum_};
    sto_.RandomInitByArray(seedArray, 2);

    // LOCK !!!!!
    {
        boost::lock_guard<boost::mutex> lock(mutex);
        STPlan_ = fftw_plan_dft_1d(m_,
            SPlusIT_.get(),
            SPlusIT_.get(),
            FFTW_BACKWARD, FFTW_ESTIMATE);
    }
}

void BMPathGenerator::computeEigenvector_(void) {
    // LOCK !!!!!!!!
    fftw_plan eigenvaluesPlan;
    {
        boost::lock_guard<boost::mutex> lock(mutex);
        eigenvaluesPlan
            = fftw_plan_dft_1d(m_,
                ev_.get(),
                ev_.get(),
                FFTW_BACKWARD, FFTW_ESTIMATE);
    }
    for (long i=0; i < m_; i++) {
        if (i <= pow(2.0, n_))
            ev_.get()[i][0] = covariance_(i);
        else
            ev_.get()[i][0] = ev_.get()[m_ - i][0];
    }
    fftw_execute(eigenvaluesPlan);
```
for (long i=0; i<m_; i++) {
    if (ev_.get()[i][0] <= 0) {
        cout << "fbm::Negative_eigenvalues_for_H=\" << H_ << endl;
        ev_.get()[i][0] = 0.0;
    }
}

//LOCK!!!!!!!!
{
    boost::lock_guard<boost::mutex> lock(mutex);
    fftw_destroy_plan(eigenvaluesPlan);
}

void FBMPathGenerator::computeTrajectory( boost::shared_array<double> output ) {
    computeSandT_();
    fftw_execute(STPlan_);
    double scaling = pow(L_/pow(2.0,m_),H_);
    for (long i=0; i<pow(2.0,m_); i++) {
        output[i] = scaling*(SPlusIT_.get()[i][0]);
        if (cum_ && i>0) {
            output[i] += output[i-1];
        }
    }
}

void FBMPathGenerator::computeSandT_() {
    SPlusIT_.get()[0][0] = sqrt(ev_.get()[0][0])*sto_.Normal(0,1)/sqrt((double)m_);
    SPlusIT_.get()[0][1] = 0;
    SPlusIT_.get()[m_/2][0] = sqrt(ev_.get()[m_/2][0])*sto_.Normal(0,1)/sqrt((double)m_);
    SPlusIT_.get()[m_/2][1] = 0;

    for (long i=1; i<m_/2; i++) {
        SPlusIT_.get()[i][0] = sqrt(ev_.get()[i][0])*sto_.Normal(0,1)/sqrt((double)2*m_);
        SPlusIT_.get()[i][1] = sqrt(ev_.get()[i][0])*sto_.Normal(0,1)/sqrt((double)2*m_);
        SPlusIT_.get()[m_-i][0] = SPlusIT_.get()[i][0];
        SPlusIT_.get()[m_-i][1] = -SPlusIT_.get()[i][1];
    }
}

FBMPathGenerator::FBMPathGenerator() {
    //LOCK!!!!!!!!


C.3.2 Code to estimate var(zeta)

**FBMZetaIntegralEstimator.h**

```cpp
#ifndef FBMZETAINTEGRALESTIMATOR_H
#define FBMZETAINTEGRALESTIMATOR_H

#include "FBMSimulationData.h"

class FBMZetaIntegralEstimator {
    public:
        FBMZetaIntegralEstimator();
        FBMZetaIntegralEstimator(int N, long n, double H, double T);
        ~FBMZetaIntegralEstimator();
        double computeResult();
        double computeResult_Golubev();
        double computeDiscrepancy();
    private:
        void computeBlock(int const threadNum,
                          int const left,
                          int const right);
        double T_; // domain of integration to be estimated falls in [-T,T]
        int baseSeed_; 
        FBMSimulationData simulationData_; 
};

#endif
```

**FBMZetaIntegralEstimator.cpp**

```cpp
#include <ctime>
#include <cmath>
#include <iostream>
#include <fstream>
#include <sstream>
#include <boost/shared_array.hpp>
#include <boost/thread.hpp>
#include <boost/accumulators/accumulators.hpp>
#include <boost/accumulators/statistics/stats.hpp>
#include <boost/accumulators/statistics/mean.hpp>
#include <boost/accumulators/statistics/variance.hpp>
```
# include <boost/accumulators/statistics/covariance.hpp>
# include <boost/accumulators/statistics/variates/covariate.hpp>

# include "FBMZetaIntegralEstimator.h"
# include "FBMPathGenerator.h"
# include "WorkDistributor.h"

using namespace boost::accumulators;
using namespace std;

FBMZetaIntegralEstimator::FBMZetaIntegralEstimator ( void ) {}

FBMZetaIntegralEstimator::FBMZetaIntegralEstimator ( int N, long n, double H, double T)
: T_(T), simulationData_(N,n,H),
  usualEstimateData_(N,n,H),
  controlVarData_(N,n,H){
  baseSeed_=(int)time(0);
}

double FBMZetaIntegralEstimator::computeResult () {
  int numThreads = boost::thread::hardware_concurrency();
  WorkDistributor workDistrib(simulationData_.numTrajectories(), numThreads);
  workDistrib.allocateWorkToThreads(*this, &FBMZetaIntegralEstimator::computeBlock);

  std::cout << "estimate of zeta^2 = " << simulationData_.estimate () << std::endl;
  std::cout << "std. Error = " << simulationData_.standardError() << std::endl;

  // file output
  std::string fileName;
  std::stringstream HVal;
  HVal << (this->simulationData_.H() * 100);
  std::stringstream nVal;
  nVal << simulationData_.lengthExponent();
  fileName = "zeta_n" + nVal.str() + "H" + HVal.str() + ".txt";
  std::ofstream file(fileName.c_str());
  file << "N, T, H, est, stdErr" << std::endl;
  file << simulationData_.numTrajectories() << ", "
       << T_ << ", "
       << simulationData_.H () << ", "
       << simulationData_.estimate () << ", "
       << simulationData_.standardError() << std::endl;
  file << std::endl << std::endl << "Trial results" << std::endl;
  for (int i = 0; i < simulationData_.numTrajectories(); ++i) {
    file << simulationData_.trialResult ( i ) << std::endl;
  }
}
return 0.0;
}

void FBMZetaIntegralEstimator::computeBlock(int const threadNum,  
    int const left,  
    int const right){
    // fBM trajectory stored here
    int n = simulationData_.lengthExponent();  
    double H = simulationData_.H();

    long numDiscretisationPoints = (long)pow(2.0,n);  
    boost::shared_array<double> output(new double[numDiscretisationPoints]);
    FBMPathGenerator fbm(n,H,2.0*T_,1,baseSeed_,threadNum);

    double deltaT = 2*T_/numDiscretisationPoints;

    for(int i = left ; i <= right ; ++i) {  
        fbm.computeTrajectory(output);
        // compute estimate for \int_{-\infty}^{\infty} \exp{B_t^{(H)} -1/2 u^{2H}} du
        double s1 = 0.0;
        // compute estimate for \int_{-\infty}^{\infty} u \exp{B_t^{(H)} -1/2 u^{2H}} du
        double s2 = 0.0;
        double t_i = -T_;

        // index of fBm_T in generated path over [0,2T]
        long midPointIndex = (long)pow(2.0,n-1);

        for(long j = 1; j < numDiscretisationPoints; ++j) {
            double expTerm = exp(output[j]-output[midPointIndex]-0.5*pow(abs(t_i),2*H));
            s1 += expTerm*deltaT;
            s2 += t_i*expTerm*deltaT;
            t_i += deltaT;
        }
        double estimate = s2/s1;
        estimate *= estimate;
        simulationData_.submitTrialResult(estimate,i);
    }

FBMZetaIntegralEstimator::FBMZetaIntegralEstimator(void) {}
# define PROBMAXFBMESTIMATOR_H
#include "FBMPathGenerator.h"
#include "FBMSimulationData.h"

// This class computes an estimate for Prob( max B^H_t < x/(T^{(1-H)}) ) for t in [0,1]
// with 2^n discretization points in [0,1], and N trajectories
/////////////////////////////////////////////////////////////////////
class ProbMaxFBMEstimator {
  public:
    ProbMaxFBMEstimator(int N, long n, double H, double x, double T);
    ~ProbMaxFBMEstimator(void);
    double computeResult();
  private:
    void computeBlock(int const threadNum,
                      int const left,
                      int const right);
    int baseSeed_; // following 2 values specify the boundary
    double x_;     // double T_;     //
    FBMSimulationData simulationData_; //
};
#endif

ProbMaxFBMEstimator.cpp
#include "ProbMaxFBMEstimator.h"
#include "FBMPathGenerator.h"
#include "WorkDistributor.h"
#include <cmath>
#include <iostream>
#include <boost/shared_array.hpp>
#include <boost/thread.hpp>
#include <cstdlib>
#include <fstream>
#include <sstream>
#include <boost/accumulators/accumulators.hpp>
#include <boost/accumulators/statistics/stats.hpp>
#include <boost/accumulators/statistics/mean.hpp>
#include <boost/accumulators/statistics/variance.hpp>
using namespace boost::accumulators;

ProbMaxFBMEstimator::ProbMaxFBMEstimator(int N, long n, double H, double x, double T)
  : x_(x), T_(T), simulationData_(N, n, H) {
21    baseSeed_=(int)time(0);
22 }
23
24 ProbMaxFBMEstimator::ProbMaxFBMEstimator(void) {
25 }
26
double ProbMaxFBMEstimator::computeResult() {
27
    int N = simulationData_.numTrajectories();
    int n = boost::thread::hardware_concurrency(); // We set n = (# of cores)
    WorkDistributor workDistrib(N, n);
    workDistrib.allocateWorkToThreads(*this,
        &ProbMaxFBMEstimator::computeBlock);

    std::cout << "estimate of \( T^{1-H}\) * P(\( \text{maxFBM} \leq x/T^H \)) = "
        << simulationData_.estimate() << std::endl;
    std::cout << "std. Error = " << simulationData_.standardError() << std::endl;

    // file output
    std::string fileName;
    std::stringstream TVal;
    TVal << (T_);
    std::stringstream HVal;
    HVal << (this->simulationData_.H() * 100);
    std::stringstream nVal;
    nVal << simulationData_.lengthExponent();
    fileName = "ProbMAXFBM_n" + nVal.str() + "H" + HVal.str()
        + "T" + TVal.str() + ".txt";
    std::ofstream file(fileName.c_str());
    file << "N, n, H, x, T, est, stdErr" << std::endl;
    file << simulationData_.numTrajectories() << ","
        << simulationData_.lengthExponent() << "," 
        << simulationData_.H() << "," 
        << x_ << "," 
        << T_ << "," 
        << simulationData_.estimate() << "," 
        << simulationData_.standardError() << std::endl;
    return 0.0;
}

void ProbMaxFBMEstimator::computeBlock(int const threadNum,
    int const left,
    int const right) {

    // FBM trajectory stored here
    long n = simulationData_.lengthExponent();
    double H = simulationData_.H();
long numDiscretisationPoints = (long)pow(2.0, n);
boost::shared_array<double> output(new double[numDiscretisationPoints]);
FBMPathGenerator fbm(n, H, 1.0, 1.0, baseSeed_, threadNum);

double boundary = x_ / (pow(T_, H));

for(int i = left; i <= right; ++i) {
    fbm.computeTrajectory(output);
    bool isBoundaryPierced = false;
    for(long j = 1; j < numDiscretisationPoints; ++j) {
        if (output[j] > boundary) {
            simulationData_.submitTrialResult(0.0, i);
            isBoundaryPierced = true;
            break;
        }
    }
    if (isBoundaryPierced == false)
        // boundary was never pierced
        simulationData_.submitTrialResult(pow(T_, (1 - H)), i);
}

C.3.4 WorkDistributor code

WorkDistributor.h

```cpp
#include <boost/thread.hpp>
#include <boost/bind.hpp>

// Given 'N' units of 'identical' work to be performed and 'n' available threads, divide
// up the 'N' tasks into 'n' evenly spaced 'chunks'. Each thread is then assigned
// a chunk to work on via a provided thread-safe method.
// e.g. Given 21 tasks and 2 threads, thread '0' works on tasks 0...9 and thread
// '1' works on tasks 10..20

class WorkDistributor {
public:
    WorkDistributor(void){}
    WorkDistributor(int const N, int const numThreads)
        : N_(N), numThreads_(numThreads){}
    ~WorkDistributor(void){}

    // Initialize work distribution
    void initializeWorkDistribution(int const N, int const numThreads) {
        // Implement work distribution logic here
    }

    // Submit trial result
    void submitTrialResult(double result, int index) {
        // Implement submit trial result logic here
    }

    // Get the number of threads
    int getNumThreads() const {
        return numThreads_;}

    // Get the number of tasks
    int getN() const {
        return N_;}

private:
    int N_; // Total number of tasks
    int numThreads_; // Number of threads
};
```
//Given an object reference and an associated pointer-to-member-method
//Each thread executes obj.pmf(threadNum, left, right)
//where each thread is assigned its own 'chunk' [left, right) to work on.
//The method pointed-to by pmf must be thread-safe.

template<typename T>
void allocateWorkToThreads(T& obj, 
  void (T::*pmf)(int const threadNum, 
          int const left, 
          int const right)){
  boost::thread_group tg;
  for(int i = 0; i < numThreads_; ++i) {
    int left = i*(N_/numThreads_);
    int right;
    if (i == numThreads_-1)
      right = (i+1)*(N_/numThreads_)-1+(N_%numThreads_);
    else
      right = (i+1)*(N_/numThreads_)-1;
    std::cout << "Thread #" << i << " working on (" << left << "," << right << ");" <<std::endl;
    tg.create_thread(boost::bind(pmf, &obj,i,left,right));
  }
  tg.join_all();
}
protected:
private:
  int N_, numThreads_;
};

C.3.5 FBMSimulationData code

FBMSimulationData.h

#undef FBMSIMULATIONDATA_H
#endif /* FBMSIMULATIONDATA_H */
#include <tbb/concurrent_vector.h>
#include <boost/accumulators/accumulators.hpp>
#include <boost/accumulators/statistics/stats.hpp>
#include <boost/accumulators/statistics/mean.hpp>
#include <boost/accumulators/statistics/variance.hpp>
#include "FBMPathGenerator.h"

/*
This class stores results of a MC simulation run. Once simulation is
class FBMSimulationData {
public:
    FBMSimulationData(void);
    FBMSimulationData(int N, long n, double H);
    ~FBMSimulationData(void);

    // Following 3 methods can only be called after all simulations are complete
    double estimate();
    double var(); // variance
    double standardError();

    double H();
    int numTrajectories();
    long lengthExponent();

    // The only thread-safe member function
    void submitTrialResult(double result, int index);

private:
    int N_; // number of trajectories to draw
    long n_; // each fBM trajectory will have 2^n discretization points in [0,1]
    double H_; // Hurst parameter
    int resultsProcessed_;

    tbb::concurrent_vector<double> results_;
    boost::accumulators::accumulator_set<double,
        boost::accumulators::stats<boost::accumulators::tag::moment<1>,
            boost::accumulators::tag::moment<2>,
            boost::accumulators::tag::variance(
                boost::accumulators::immediate) >> moments_;

    void processResults_();
};

FBMSimulationData.cpp
#include <cmath>
#include "FBMSimulationData.h"
using namespace boost::accumulators;
FBMSimulationData::FBMSimulationData(void) {}

FBMSimulationData::FBMSimulationData(int N, long n, double H)
    : N_(N), n_(n), H_(H), resultsProcessed_(0), results_(N) {
}

double FBMSimulationData::estimate()
{
    if(!resultsProcessed_) {
        processResults_();
        resultsProcessed_ = 1;
    }
    return moment<1>(moments_);
}

double FBMSimulationData::var()
{
    if(!resultsProcessed_) {
        processResults_();
        resultsProcessed_ = 1;
    }
    return variance(moments_);
}

double FBMSimulationData::standardError()
{
    if(!resultsProcessed_) {
        processResults_();
        resultsProcessed_ = 1;
    }
    return (sqrt(variance(moments_)/N_));
}

void FBMSimulationData::submitTrialResult(double result, int index) {
    results_[index] = result;
}

void FBMSimulationData::processResults_()
{
    for (tbb::concurrent_vector<double>::size_type i = 0; i < results_.size(); ++i) {
        moments_(results_[i]);
    }
}

double FBMSimulationData::H() { return H_; }
int FBMSimulationData::numTrajectories() { return N_; }
long FBMSimulationData::lengthExponent() { return n_; }
double FBMSimulationData::trialResult(int index) { return results_[index]; }
FBMSimulationData:

FBMSimulationData(void){}
References


