SIMULATING GAUSSIAN RANDOM FIELDS AND SOLVING STOCHASTIC DIFFERENTIAL EQUATIONS USING BOUNDED WIENER INCREMENTS

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

2013

Phillip Taylor
School of Mathematics
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>9</td>
</tr>
<tr>
<td>Declaration</td>
<td>10</td>
</tr>
<tr>
<td>Copyright Statement</td>
<td>11</td>
</tr>
<tr>
<td>Publications</td>
<td>12</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>13</td>
</tr>
<tr>
<td>List of abbreviations</td>
<td>14</td>
</tr>
<tr>
<td>Notation</td>
<td>15</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>20</td>
</tr>
<tr>
<td>1.1 Part I: Simulation of random fields</td>
<td>20</td>
</tr>
<tr>
<td>1.2 Implementation of the circulant embedding method in the NAG Fortran Library</td>
<td>22</td>
</tr>
<tr>
<td>1.3 Part II: Solution of SDEs</td>
<td>24</td>
</tr>
<tr>
<td><strong>1 Simulating Gaussian Random Fields</strong></td>
<td>26</td>
</tr>
<tr>
<td><strong>2 Stochastic Processes and Random Fields</strong></td>
<td>27</td>
</tr>
<tr>
<td>2.1 Random Variables</td>
<td>27</td>
</tr>
<tr>
<td>2.1.1 The Gaussian distribution</td>
<td>30</td>
</tr>
<tr>
<td>2.2 Stochastic Processes</td>
<td>33</td>
</tr>
<tr>
<td>2.2.1 Martingales</td>
<td>34</td>
</tr>
<tr>
<td>2.2.2 The Wiener process</td>
<td>35</td>
</tr>
</tbody>
</table>
3 Linear Algebra

3.1 Discrete Fourier Transforms ........................................ 45
3.2 Toeplitz and Circulant Matrices .................................... 48
3.3 Embeddings of Toeplitz matrices into circulant matrices .......... 49
3.4 Block Toeplitz and Block Circulant Matrices ....................... 54
    3.4.1 Additional notation for BSTSTB and symmetric BTTB matrices 56
    3.4.2 Notation for padding ........................................ 57
3.5 Embeddings of BSTSTB and symmetric BTTB matrices into BSCSCB and symmetric BCCB matrices ......................... 60
3.6 Matrix Factorisations ............................................. 61

4 The Circulant Embedding Method .................................... 64

4.1 Introduction ..................................................... 64
4.2 The circulant embedding method in one dimension ................ 67
4.3 The circulant embedding method in two dimensions ............... 69
4.4 Example: Simulating Fractional Brownian Motion using the circulant embedding method .............................. 73
4.5 Approximation in the case that a non-negative definite embedding matrix cannot be found ........................... 75
4.6 Cases for which the embedding matrix is always non-negative definite ................. 76
4.7 Implementation in the NAG Fortran Library ....................... 77
    4.7.1 The NAG Library routines ................................ 78
    4.7.2 List of variograms for G05ZNF and G05ZRF ............. 82
4.8 Numerical examples ............................................. 84
    4.8.1 Notes on the numerical examples ......................... 85
    4.8.2 Stringent tests ........................................ 93
6.3 Two examples of variable stepsize methods

6.3.1 Method proposed by Gaines and Lyons

6.3.2 Method proposed by Lamba

6.3.3 A comparison of the two methods

7 Simulating a Wiener Process

7.1 Deriving the PDE to solve for bounded diffusion

7.1.1 Background

7.2 Simulating the exit time for a one-dimensional Wiener process

7.2.1 Separation of variables

7.2.2 Using a Green’s function

7.3 Simulating the exit point of a Wiener process if \( \tau = T_{\text{inc}} \)

7.3.1 Separation of variables

7.3.2 Using a Green’s function

7.4 Simulating the exit point and exit time of a one-dimensional Wiener process with unbounded stepsize and bounded stepsize

7.5 Simulating the exit time of a multi-dimensional Wiener process

7.5.1 Simulating the exit point of a multi-dimensional Wiener process if \( \tau = T_{\text{inc}} \)

7.5.2 Simulating the exit point of the multi-dimensional Wiener process if \( \tau < T_{\text{inc}} \)

7.5.3 The final algorithm for simulating the bounded Wiener process

7.6 Using bounded increments to generate a path of the Wiener process

8 Solving SDEs using Bounded Wiener Increments

8.1 Background

8.1.1 Pathwise global error of order \( \gamma - \frac{1}{2} \) for numerical schemes in \( \mathcal{S}_1 \)

8.1.2 Pathwise global error of order \( \gamma \) for numerical schemes in \( \mathcal{S}_2 \)

8.2 The Bounded Increments with Error Control (BIEC) Method

8.3 Numerical results

8.3.1 The numerical methods used

8.3.2 The numerical results

8.3.3 Conclusions from the numerical experiments
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.01$ and $\nu = 0.5$.</td>
<td>86</td>
</tr>
<tr>
<td>4.2</td>
<td>Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.01$ and $\nu = 2.5$.</td>
<td>86</td>
</tr>
<tr>
<td>4.3</td>
<td>Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.002$ and $\nu = 0.5$.</td>
<td>87</td>
</tr>
<tr>
<td>4.4</td>
<td>Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.002$ and $\nu = 2.5$.</td>
<td>87</td>
</tr>
<tr>
<td>4.5</td>
<td>Plots of the variogram (4.11) for the different values of $\ell$ and $\nu$.</td>
<td>88</td>
</tr>
<tr>
<td>4.6</td>
<td>Four realisations of fractional Brownian motion with Hurst parameter $H = 0.2$.</td>
<td>88</td>
</tr>
<tr>
<td>4.7</td>
<td>Four realisations of fractional Brownian motion with Hurst parameter $H = 0.8$.</td>
<td>89</td>
</tr>
<tr>
<td>4.8</td>
<td>Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.25$, $\ell_2 = 0.5$ and $\nu = 1.1$.</td>
<td>89</td>
</tr>
<tr>
<td>4.9</td>
<td>Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.5$, $\ell_2 = 0.25$ and $\nu = 1.8$.</td>
<td>90</td>
</tr>
<tr>
<td>4.10</td>
<td>Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.05$, $\ell_2 = 0.1$ and $\nu = 1.1$.</td>
<td>90</td>
</tr>
<tr>
<td>4.11</td>
<td>Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.1$, $\ell_2 = 0.05$ and $\nu = 1.8$.</td>
<td>91</td>
</tr>
<tr>
<td>4.12</td>
<td>Plots of the variogram (4.10) in one dimension for different values of $\ell$ and $\nu$.</td>
<td>91</td>
</tr>
<tr>
<td>6.1</td>
<td>An illustration of the Brownian tree, for $K = 4$.</td>
<td>137</td>
</tr>
</tbody>
</table>
6.2 An illustration to show how each element of the Wiener increments
\[ \Delta W_{2k-1,\ell+1} \text{ and } \Delta W_{2k,\ell+1} \] are generated from \[ \Delta W_{k,\ell} \text{, for } s = (k - 1) \Delta t_{\text{max}} \text{, } t = k \Delta t_{\text{max}}. \] 138

7.1 The possible cases for the exit points of \[ X(t) \] from \[ G \], with \[ m = 1, \]
\[ X(0) = 0 \text{ and } G = (-r, r). \] 150

7.2 The possible cases for the behaviour of \[ X(t) \], with \[ m = 1, X(0) = 0 \text{ and } G = (-r, r). \] 150

8.1 Numerical results over 1000 samples for geometric Brownian motion
with \[ d = m = 1, a = 0.1 \text{ and } b = 0.1. \] 187

8.2 Numerical results over 1000 samples for geometric Brownian motion
with \[ d = m = 1, a = 1.5 \text{ and } b = 0.1. \] 188

8.3 Numerical results over 1000 samples for geometric Brownian motion
with \[ d = m = 1, a = 0.1 \text{ and } b = 1.2. \] 189

8.4 Numerical results over 1000 samples for geometric Brownian motion
with \[ d = m = 1, a = 1.5 \text{ and } b = 1.2. \] 190

8.5 Numerical results over 1000 samples for geometric Brownian motion
with \[ d = m = 2. \] 191

8.6 Numerical results over 1000 samples for (8.29) with \[ a = 1 \text{ and } b = 0.1 \]. 192

8.7 Numerical results over 1000 samples for (8.29) with \[ a = 1 \text{ and } b = 1 \]. 193

8.8 Numerical results over 1000 samples for (8.31) with \[ a = 0.2. \] 194

8.9 Numerical results over 1000 samples for (8.31) with \[ a = 2. \] 195
This thesis is in two parts. Part I concerns simulation of random fields using the
circulant embedding method, and Part II studies the numerical solution of stochastic
differential equations (SDEs).

A Gaussian random field $Z(x)$ is a set of random variables parameterised by a
variable $x \in D \subseteq \mathbb{R}^d$ for some $d \geq 1$ which, when sampled at a finite set of sampling
points, produces a random vector with a multivariate Gaussian distribution. Such a
random vector has a covariance matrix $A$ which must be factorised as either $A = R^T R$
for a real-valued matrix $R$, or $A = R^* R$ for a complex-valued matrix $R$, in order to
take realisations of $Z(x)$. Using the Cholesky factorisation is expensive; however if $A$
is Toeplitz, it is possible to embed $A$ into a circulant matrix $B$ which can be factorised
cheaply using the fast Fourier transform. Similarly if $A$ is block Toeplitz with Toeplitz
blocks (BTTB), it can be embedded into a matrix $B$ which is block circulant with
circulant blocks (BCCB). This forms the basis for the circulant embedding method.

An SDE is an equation of the form

$$X(t) = x_0 + \int_0^t f(X(s)) \, ds + \sum_{j=1}^{m} \int_0^t g_j(X(s)) \, dW_j(s),$$

where $W(t)$ is an $m$-dimensional Wiener process. Many numerical schemes have been
derived to solve SDEs, including the Euler–Maruyama and Milstein schemes, often
using a fixed stepsize. However it is often advantageous to use variable stepsizes in
order to improve efficiency and to reduce the strong global error. This thesis proposes
a new variable stepsize method for the solution of SDEs. Using theory from \[57\], it is
possible to bound both the stepsize $t_{n+1} - t_n$ and the Wiener increment components
$W_j(t_{n+1}) - W_j(t_n)$ for each step, and change the bounds on the stepsize and Wiener
increment bounds for each step based on error control criteria. Once the stepsize
and Wiener increment have been simulated the solution is advanced using the Euler–
Maruyama scheme. This so-called Bounded Increments with Error Control method is
then tested against a fixed stepsize method using the Euler–Maruyama scheme.

I spent a total of 12 weeks working with the Numerical Algorithms Group (NAG).
Most of this time was spent implementing the circulant embedding method in the
NAG Fortran Library, and the rest was devoted to an implementation of fixed stepsize
methods for solving SDEs, also intended for inclusion in the NAG Fortran Library.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s Policy on Presentation of Theses.
Publications

The material in Chapter 8 will form part of the following paper:
I have many people to thanks for the help, support and encouragement they have given me throughout my PhD.

Firstly I would like to thank my supervisors, Dr. Tony Shardlow and Dr. Catherine Powell, for all of their help, support and guidance they have given to me over the last four years. I would also like to thank my examiners, Prof. David Silvester and Prof. Michael Tretyakov, for their comments and suggestions made during my viva.

I would like to thank the EPSRC and Numerical Algorithms Group (NAG) for the funding I received during my PhD programme, and I would also like to thank all of the staff at NAG for providing me with the invaluable opportunity to work with them. In particular I thank Dr. Lawrence Mulholland, my primary contact at NAG, for all of his technical expertise.

I have a lot of friends who have managed to keep me going through all of the tough times. In particular my housemates - Jasmina, Richard, Alex and Adele - and my friends within the University of Manchester mathematics department who understand how tough a PhD is! I am also thankful to the Manchester University Music Society and Bridge Society for giving me an interesting and varied life outside mathematics.

My parents, Miriam and Hugh, have provided unwavering support throughout my life and I will always be forever in their debt, and I would also like to thank the rest of my family for their support.

Finally a special mention to a truly special person, Katherine, for her incredible levels of love and encouragement.
# List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B(S)C(S)CB</td>
<td>block (symmetric) circulant with (symmetric) circulant blocks</td>
</tr>
<tr>
<td>B(S)T(S)TB</td>
<td>block (symmetric) Toeplitz with (symmetric) Toeplitz blocks</td>
</tr>
<tr>
<td>BVP</td>
<td>boundary value problem</td>
</tr>
<tr>
<td>BI</td>
<td>Bounded Increments</td>
</tr>
<tr>
<td>BIEC</td>
<td>Bounded Increments with Error Control</td>
</tr>
<tr>
<td>cdf</td>
<td>cumulative distribution function</td>
</tr>
<tr>
<td>DFT</td>
<td>Discrete Fourier Transform</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>IDFT</td>
<td>Inverse Discrete Fourier Transform</td>
</tr>
<tr>
<td>iid</td>
<td>independent and identically distributed</td>
</tr>
<tr>
<td>LTE</td>
<td>local truncation error</td>
</tr>
<tr>
<td>M(O)CE</td>
<td>minimal (odd) circulant embedding</td>
</tr>
<tr>
<td>NAG</td>
<td>Numerical Algorithms Group</td>
</tr>
<tr>
<td>ODE</td>
<td>ordinary differential equation</td>
</tr>
<tr>
<td>PDE</td>
<td>partial differential equation</td>
</tr>
<tr>
<td>PGE</td>
<td>pathwise global error</td>
</tr>
<tr>
<td>pdf</td>
<td>probability density function</td>
</tr>
<tr>
<td>SDE</td>
<td>stochastic differential equation</td>
</tr>
<tr>
<td>SGE</td>
<td>strong global error</td>
</tr>
<tr>
<td>WGE</td>
<td>weak global error</td>
</tr>
</tbody>
</table>
Notation

General

\( \mathbb{R} \) set of real numbers
\( \mathbb{R}_+ \) set of positive real numbers
\( \mathbb{R}^d \) set of \( d \)-dimensional vectors with real entries
\( \mathbb{C} \) set of complex numbers
\( \mathbb{C}^d \) set of \( d \)-dimensional vectors with complex entries
\( \mathbb{N} \) set of natural numbers
\( \mathbb{Z} \) set of integers
\( \mathbb{Z}_+ \) set of integers greater than or equal to zero

Functions

\( \|u\|_H \) norm of \( u \) with respect to the Banach space \( H \)
\( \langle u, v \rangle_H \) inner product of \( u \) and \( v \) with respect to the Hilbert space \( H \)
\( \hat{u} \) Fourier transform of \( u \)
\( \check{u} \) inverse Fourier transform of \( u \)
\( C^k(X,Y) \) set of continuous functions from \( X \) to \( Y \) with \( k \) continuous derivatives
\( C^k_0(X,Y) \) set of functions in \( C^k(X,Y) \) with bounded support
\( C^k(X), C^k_0(X) \) \( C^k(X, \mathbb{R}), C^k_0(X, \mathbb{R}) \)
\( C^{j,k}(X \times Y, Z) \) set of continuous functions from \( X \times Y \) to \( Z \) with \( j \) continuous derivatives in \( X \) and \( k \) continuous derivatives in \( Y \)
\( O(x) \) of order \( x \)
Probability theory

\( \Omega \) sample space
\( F \) \( \sigma \)-algebra
\( \{F_t\}_{t \geq 0}, F_t \) filtration of the \( \sigma \)-algebra \( F \)
\( \mathbb{P} \) probability measure
\( (\Omega, F, \mathbb{P}) \) probability space
\( (\Omega, F, \{F_t\}_{t \geq 0}, \mathbb{P}) \) filtered probability space
\( \mathbb{E} \) expectation
\( \mathcal{N}(\mu, \sigma^2) \) normal distribution
\( \mathbb{C}\mathcal{N}(\mu, \sigma^2) \) complex normal distribution
\( L^p(\Omega, H) \) the Banach space of \( H \)-valued random variables
\( L^p(\Omega) \) the Banach space of real-valued random variables
\( 1_E \) the indicator function for the event \( E \)

Matrices

\( \mathbf{0}_m \) vector of length \( m \) with all entries zero
\( \mathbf{0}_{m,n} \) matrix of size \( m \times n \) with all entries zero
\( I_m \) \( m \times m \) identity matrix
\( \text{vec}(A) \) vectorisation of the matrix \( A \)
\( \text{diag}(A) \) diagonal of \( A \)
\( A \otimes B \) Kronecker product of \( A \) and \( B \)
\( a_i \) subvector of \( a \) with indexing vector \( i \)
\( A_{i,j} \) submatrix of \( A \) with indexing vectors \( i,j \)
\( i' \) complement of indexing vector \( i \)
\( W_n \) Fourier matrix
\( W_{n_1, \ldots, n_d} \) \( d \)-dimensional Fourier matrix
\( \hat{u} \) DFT of \( u \)
\( \hat{\mathbf{u}} \) IDFT of \( u \)
\( \text{Toe}(\mathbf{a}, \mathbf{b}) \) Toeplitz matrix with first row \( \mathbf{a} \) and first column \( \mathbf{b} \)
\( \text{symToe}(\mathbf{a}) \) symmetric Toeplitz matrix with first row/column \( \mathbf{a} \)
circ(\(a\)) circulant matrix with first row \(a\)
Toe(\(A,B\)) BTTB matrix with first block row \(A\) and first column \(B\)
symToe(\(A\)) BSTSTB matrix with first block row/column \(A\)
circ(\(A\)) circulant matrix with first block row \(A\)
symBTTB(\(a\)) symmetric BTTB matrix with first row/column \(a\)
BSTSTB(\(a\)) BSTSTB matrix with first row/column \(a\)
MCEvec(\(a\)) first row of the MCE of symToe(\(a\))
MCE(\(a\)) MCE of symToe(\(a\))
MCEvec(\(a, b\)) first row of the MCE of Toe(\(a, b\))
MCE(\(a, b\)) MCE of Toe(\(a, b\))
MOCEvec(\(a\)) first row of the MOCE of symToe(\(a\))

etc.
BEvec_{mn}^{\alpha\beta}(\(a\)) first row of the block embedding of the symmetric BTTB or BSTSTB matrix with first row \(a\)
BE_{mn}^{\alpha\beta}(\(a\)) block embedding of the symmetric BTTB or BSTSTB matrix with first row \(a\)

Random fields
\(C(x,y)\) covariance function
\(\gamma(x)\) variogram

Stochastic processes and Brownian motion
\(W(t)\) one-dimensional Wiener process
\(W(t)\) multi-dimensional Wiener process
\(B(t)\) Brownian bridge process
\(B^H(t)\) Fractional Brownian motion with Hurst parameter \(H\)
Abbreviations for stepsizes and Wiener increments

- $\Delta t_{\text{max}}$: maximum stepsize
- $\Delta t_n$: $t_{n+1} - t_n$
- $\Delta W_{\text{max}}, \Delta W_{\text{max}}$: maximum Wiener increment
- $\Delta W_{\text{max},j}$: $j$-th component of the maximum Wiener increment
- $\Delta W(s,t), \Delta W(s,t)$: $W(t) - W(s)$, $W(t) - W(s)$
- $\Delta W_n, \Delta W_n$: $W(t_{n+1}) - W(t_n)$, $W(t_{n+1}) - W(t_n)$
- $\Delta W^n_j$: $W_j(t_{n+1}) - W_j(t_n)$ (for multidimensional Wiener process $W(t)$)

Multi-indices

- $\alpha$: multi-index of form $(j_1, \ldots, j_\ell)$
- $\ell(\alpha)$: length of $\alpha$
- $n(\alpha)$: number of zero elements in $\alpha$
- $\alpha$: $\alpha$ with last element removed
- $-\alpha$: $\alpha$ with first element removed
- $\emptyset$: multi-index of length zero
- $\mathcal{M}$: set of all multi-indices
- $\mathcal{A}$: hierarchical set
- $\mathcal{B}(\mathcal{A}), \mathcal{B}$: remainder set of $\mathcal{A}$
- $\mathcal{A}_S, \mathcal{B}_S$: hierarchical and remainder sets associated with numerical scheme $S$

Partitions

- $D([t-, t_+]), D$: partition of interval $[t-, t_+]$
- $D\Delta([t-, t_+]), D\Delta_{t_{\text{max}}}$: partition with equal spacing of $\Delta t$
- $\mathcal{D}([t-, t_+])$: set of partitions of $([t-, t_+])$
- $\mathcal{D}\Delta_{t_{\text{max}}}([t-, t_+])$: set of partitions of $([t-, t_+])$ with maximum stepsize $\Delta t_{\text{max}}$
- $\mathcal{D}\text{stop}([t-, t_+])$: set of partitions of $([t-, t_+])$ composed of stopping times
- $\mathcal{D}\text{stop}\Delta_{t_{\text{max}}}([t-, t_+])$: set of partitions of $([t-, t_+])$ composed of stopping times with maximum stepsize $\Delta t_{\text{max}}$
\( \mathcal{P}_D \) partition \( \sigma \)-algebra of partition \( D \)

\( \mathcal{P}_N \) partition \( \sigma \)-algebra of partition \( D \) composed of \( N \) equal steps

**Stochastic integrals**

- \( I_{[0,T]}(g), I_{[0,T]}(g) \): Itô integral of \( g \)
- \( J_{[0,T]}(g), J_{[0,T]}(g) \): Stratonovich integral of \( g \)
- \( I_\alpha[Y(\cdot)]_{[s,t]} \): Itô integral of \( Y(t) \) over \([s,t]\) with multi-index \( \alpha \)
- \( I_\alpha[1]_{[s,t]} \)
- \( I^{nk}_\alpha[Y(\cdot)], I^{nk}_\alpha \): Itô integral over time interval \([t_n, t_k]\)
- \( J_\alpha[Y(\cdot)]_{[s,t]}, \) etc.: Stratonovich integrals

**Itô and Stratonovich coefficient functions**

- \( L^0, L^j \): operators associated with the Itô formula; see (5.10) and (5.11)
- \( \tilde{L}^0, \tilde{L}^j \): operators associated with the Stratonovich version of the Itô formula; see (5.13) and (5.14)
- \( F^0_\alpha(t, x) \): Itô coefficient function for multi-index \( \alpha \) and function \( u(t, x) \)
- \( F^{kl}_\alpha(x) \): Stratonovich coefficient function for multi-index \( \alpha \) and function \( u(t, x) = x \)
- \( G^0_\alpha(t, x) \): Itô coefficient function for multi-index \( \alpha \) and function \( u(t, x) \)
- \( G^{kl}_\alpha(x) \): Stratonovich coefficient function for multi-index \( \alpha \) and function \( u(t, x) = x \)

**Numerical schemes**

- EM, EM\( _\theta \): Euler–Maruyama scheme, Euler-type \( \theta \)-scheme
- EH, EH\( _\theta \): Euler–Heun scheme, Heun-type \( \theta \)-scheme
- Itô-Mil, Itô-Mil\( ^\theta \): Itô Milstein scheme and \( \theta \)-scheme
- Str-Mil, Str-Mil\( ^\theta \): Stratonovich Milstein scheme and \( \theta \)-scheme
- Mil, Mil\( ^\theta \): Milstein scheme and \( \theta \)-scheme (if obvious whether Itô or Stratonovich scheme is being used) truncation error of scheme
- \( T^S_{[s,t]} \): \( S \) over interval \([s,t]\)
- \( T^S_{n} \): truncation error of scheme \( S \) over interval \([t_n, t_{n+1}]\)
Chapter 1

Introduction

When performing mathematical modelling it is often important to model uncertainty and random effects in some way. For example, when modelling groundwater flow, we would like to know the permeability of the rock at every point in a domain; however, although we may know some of the statistics of the rock permeability, it is impossible to measure it everywhere. Similarly we may want to predict the price of certain stock at some future time in order to price an option, and the price of this stock may change in an unpredictable manner. Hence we often use random fields to model properties such as rock permeability over a certain domain, and stochastic differential equations (SDEs) to model the evolution of a quantity such as a stock price over some time interval.

This chapter gives a brief overview of the thesis, which is split into two parts: simulation of random fields using the circulant embedding method, and the solution of SDEs using adaptive stepsizes and, in particular, bounded Wiener increments.

1.1 Part I: Simulation of random fields

We study the simulation of Gaussian stochastic processes and random fields in part I of this thesis. A random field \{Z(x) : x ∈ D\}, or Z(x), is a set of random variables which is parameterised by a variable x ∈ D ⊂ R^d for some d ≥ 1, and the random field Z(x) is Gaussian if the random vector X with entries X_j = Z(x_j) follows a multivariate Gaussian distribution for any choice N ≥ 1 and any choice of sampling points x_j ∈ D for j = 1, . . . , N. In the case d = 1, the random field can also be called a random
CHAPTER 1. INTRODUCTION

function. All random fields have a mean function \( \mu(x) \) and a non-negative definite covariance function \( C(x, y) \). We provide background material for random fields in Chapter 2, including material regarding stochastic processes. The Wiener process, used extensively for solving SDEs in part II, is also defined in this chapter.

When simulating a Gaussian random field it is necessary to sample at a finite number \( N \) of sample points \( x_j \in D \), and the resulting random vector \( X \) will follow a multivariate Gaussian distribution with a known mean vector \( \mu \) and a symmetric non-negative definite covariance matrix \( A \). Realisations of the random vector \( X \) can easily be found if we can factorise the covariance matrix \( A \) as either \( A = RTR \) for a real-valued matrix \( R \), or \( A = R^\ast R \) for a complex-valued matrix \( R \). For example, if \( A \) is positive definite the Cholesky factorisation can be used to do this; however the Cholesky factorisation has a high computational cost of \( O(N^3) \) flops. The cost can be reduced to \( O(N^2) \) flops if \( A \) is Toeplitz; however if \( A \) is circulant, we can use the fast Fourier transform (FFT) to produce a factorisation \( A = R^\ast R \) with a cost of \( O(N \log N) \) flops.

The circulant embedding method exploits the fact that a circulant matrix can be factorised with such a low cost, particularly if \( N \) is a power of 2 or 3. Although it is extremely unlikely that the covariance matrix \( A \) will be circulant, it is relatively straightforward to embed a Toeplitz matrix \( A \) into a circulant matrix \( B \) and factorise \( B \) instead. The matrix \( A \) is automatically Toeplitz if the random field is one-dimensional (i.e. \( d = 1 \)), stationary and sampled over a regularly spaced grid. Now \( A \) is embedded into a circulant matrix \( B \) of size \( M = 2^G \geq 2(N-1) \) for some \( G \) which, if non-negative definite, is the covariance matrix for some Gaussian random vector \( Y \) of length \( M \). The matrix \( B \) is known as the embedding matrix. We can now sample the vector \( Y \) and then take the first \( N \) elements of \( Y \) as our realisation of \( X \). We can guarantee non-negative definiteness of \( B \) in some cases; if so the random field will have the correct finite-dimensional distributions. If a non-negative definite embedding matrix \( B \) cannot be found it may be necessary to use an approximation procedure to ensure that the covariance matrix of \( Y \) is non-negative definite, in which case the finite-dimensional distributions of the resulting random field will not be exact.

The procedure is similar for a higher dimensional random field. In the two-dimensional case, \( A \) will be block Toeplitz with Toeplitz blocks (BTTB) if the random
field is stationary and sampled on a regularly spaced grid, and we can embed $A$ into an embedding matrix $B$ which is block circulant with circulant blocks (BCCB). Such a matrix can still be factorised using the FFT, and will again be the covariance matrix for a Gaussian vector $Y$, from which a realisation of the random field can be obtained.

Circulant and BCCB matrices can be expressed using their first row or column only, so it is only necessary to store one row or column of $B$, and the first row or column of $B$ can be constructed without needing to form or store the original Toeplitz or BTTB matrix $A$. This means that the memory requirements of the circulant embedding method are low.

The linear algebra involved in embedding Toeplitz matrices into circulant matrices, and BTTB matrices into BCCB matrices, is described in Chapter 3 and full details of the circulant embedding method are given in Chapter 4.

1.2 Implementation of the circulant embedding method in the NAG Fortran Library

As part of my PhD programme I had three placements, each of four weeks duration, working at the Numerical Algorithms Group (NAG) in Oxford. During the first two of these, I developed code for the circulant embedding method for the NAG Fortran Library, while during the third placement I started development of code for fixed stepsize solutions of SDEs, based on a MATLAB implementation developed by Gilsing and Shardlow [28].

The NAG Fortran Library is a collection of over 1700 routines, designed to perform a variety of mathematical and statistical tasks. These include optimisation, the solution of ODEs and PDEs, evaluating special functions such as the gamma function and Bessel functions, linear equations and random number generation. The majority of the routines have error checks, to detect both invalid input parameters and problem within the routines themselves. Each routine is tested thoroughly using test programs designed to test the error checking procedures and ensure that the routine is producing the correct results. Full documentation for all of the routines can be found at [64]. Implementations of the NAG Fortran Library are also available for other programming languages such as C (the NAG C Library) and MATLAB (the NAG Toolbox for
MATLAB). Mark 24 of the NAG Fortran Library is currently available and it includes the routines developed through my work at NAG for the circulant embedding method, which are the routines G05ZMF–G05ZTF \[64\] routines G05ZMF-G05ZTF].

Grace Chan, co-author of a paper written with Andrew Wood on the circulant embedding method \[12\], kindly gave us access to FORTRAN 77 code which had been developed by herself and Wood to implement the circulant embedding method in one and two dimensions, including code for the simulation of fractional Brownian motion. It was then my job to work with this code so that it could be added to the NAG Fortran Library. The routines for the Library are written in Fortran 90 which includes features not included in FORTRAN 77, including dynamic memory allocation and the ability to perform operations on arrays \[21\], as well as slightly different syntax.

It was therefore necessary to change the code to use the Fortran 90 syntax, and desirable to use the additional features of Fortran 90 in order to improve efficiency of the code. The code provided by Chan used obsolete NAG Fortran Library routines to perform random number and generation and fast Fourier transforms, so these also had to be updated to use the newest versions. A major reorganisation of the existing routines was required in order to create NAG Library routines for performing the different tasks; separate routines were created to find the square roots of the eigenvalues of the embedding matrix $B$ and then to perform the realisations of the random field. It was also necessary to program error-checking procedures; the vast majority of NAG Library routines have a procedure for detecting and displaying errors.

Although the mathematics in the original FORTRAN 77 code was correct, it was necessary to understand what the code was doing in order to improve its efficiency, and to perform the necessary restructuring of the code. This was particularly true for the two-dimensional circulant embedding method, where the original code for the setting up of the embedding matrix $B$ was rather complex.

Only the one- and two-dimensional cases were implemented due to time constraints, although generalisation to higher dimensions is possible \[12\].

I am extremely grateful to NAG for the opportunity to work with them on developing code for both the circulant embedding method and fixed stepsize solutions of SDEs, and for the additional funding they gave to me.
1.3 Part II: Solution of SDEs

Part II of this thesis is dedicated to the solution of stochastic differential equations (SDEs) and, in particular, using a variable stepsize method to do so. We find the solutions of SDEs of the form

\[ X(t) = x_0 + \int_0^t f(X(s)) \, ds + \sum_{j=1}^m \int_0^t g_j(X(s)) \, dW_j(s), \tag{1.1} \]

where \( f, g_j : \mathbb{R}^d \to \mathbb{R}^d \) are time-dependent, \( x_0 \) is an initial condition and \( W(t) \) is an \( m \)-dimensional Wiener process, with independent and identically distributed (iid) components \( W_j(t) \). SDEs with time-dependent also exist; however they can be converted into SDEs of the form (1.1) and so are not studied in this thesis. Background material for the study of SDEs and their solutions is provided in Chapter 5, along with background material required in Chapters 7 and 8.

Many numerical schemes are available to solve SDEs - for example the explicit and implicit Euler–Maruyama scheme, the explicit and implicit Milstein scheme, and various Runge–Kutta-type schemes, and strong convergence of these schemes is well-known when using fixed stepsizes. However, using fixed stepsizes is not optimal if the size of the local truncation error (LTE) changes significantly along the time interval. If the LTE and the stepsize are both small, it is inefficient to take many steps, since fewer steps are required to obtain a small strong error. On the other hand, if the LTE and stepsize are both large, then large strong errors result.

As a result of this, there have been several attempts to use variable stepsizes to solve SDEs, and some of these are discussed in Chapter 6. Using variable stepsizes for SDEs is more complex than for ODEs because of issues relating to the simulation of the Wiener process at a time \( t \), when the value of the Wiener process has already been calculated at a time \( t_+ > t \). This issue also applies to multiple stochastic integrals of the form

\[ \int_s^t \int_s^{s_1} \ldots \int_s^{s_2} dW_{j_1}(s_1) \, dW_{j_2}(s_2) \ldots \, dW_{j_\ell}(s_\ell) \]

which, even for fixed stepsizes, cannot be calculated exactly from only the values of \( W(t) \) and are expensive to simulate.

This thesis proposes a new variable stepsize method for solving SDEs, using bounded Wiener increments. It is possible to bound each Wiener increment \( \Delta W_n = W_{n+1} - W_n \)
such that each component $\Delta W^n_j$ of $\Delta W_n$ is bounded in an interval $[-\Delta W^n_{\text{max},j}, \Delta W^n_{\text{max},j}]$ for some choices of $\Delta W^n_{\text{max},j}$, while also bounding the stepsize $\Delta t_n$ by a quantity $\Delta W^n_{\text{max}}$. Milstein and Tretyakov [57] outline how to simulate such bounded Wiener increments for the case where the $\Delta W^n_{\text{max},j}$ are equal for each $j = 1, \ldots, m$ - the procedure is described in detail in Chapter 7.

The proposed method for solving SDEs of the form (1.1) is outlined in detail in Chapter 8 and also forms part of a forthcoming paper [73]. We use bounded Wiener increments with the Euler–Maruyama scheme, where the maximum stepsizes $\Delta t_n$ and the maximum Wiener increment components $\Delta W^n_{\text{max},j}$ are chosen using an estimate of the LTE at each step - the name for this method is the Bounded Increments with Error Control method, or BIEC method. We claim that the pathwise global error for both the BIEC method, and the Euler–Maruyama scheme using a fixed stepsize, has order $\frac{1}{2} - \epsilon$ for any $0 < \epsilon < \frac{1}{2}$. Finally the BIEC method, along with modifications to the BIEC method, are compared to the fixed step method and another method called the Bounded Increments method, or BI method, in numerical experiments. The BI method is the method obtained by using bounded increments along with the Euler–Maruyama scheme, where the maximum stepsize $\Delta t_n$ and maximum Wiener increment components $\Delta W^n_{\text{max},j}$ are kept fixed throughout.
Part I

Simulating Gaussian Random Fields
Chapter 2

Stochastic Processes and Random Fields

In order to study stochastic processes and random fields, we shall first need to discuss some basic probability theory, and random variables in particular.

2.1 Random Variables

Let \((\Omega, \mathcal{F}, \mathbb{P})\) be a probability space, where \(\Omega\) is a sample space, \(\mathcal{F}\) is a \(\sigma\)-algebra and \(\mathbb{P}\) is a probability measure. Then, for some measurable normed space \((\Psi, \mathcal{G})\), a \(\Psi\)-valued random variable is a measurable function \(X: \Omega \rightarrow \Psi\). We shall usually take \(\Psi\) to be either \(\mathbb{R}^d\) or \(\mathbb{C}^d\) for some \(d \geq 1\), and \(\mathcal{G}\) to be the associated Borel \(\sigma\)-algebra \(\mathcal{B}(\mathbb{R}^d)\) or \(\mathcal{B}(\mathbb{C}^d)\) respectively.

Hilbert space valued random variables often belong to a Banach space.

**Definition 2.1** For a Hilbert space \(H\), the probability space \((\Omega, \mathcal{F}, \mathbb{P})\), and some \(p \geq 1\), we define the Banach space \(L^p(\Omega, H)\) to be the space of measurable \(H\)-valued random variables \(X\) such that

\[
\|X\|_{L^p(\Omega, H)} = \mathbb{E}[\|X\|_H^p]^{\frac{1}{p}} < \infty,
\]

where \(\|\cdot\|_H\) denotes the norm associated with \(H\). The space \(L^p(\Omega, \mathbb{R})\) is often written as \(L^p(\Omega)\). In particular \(L^2(\Omega, H)\) is itself a Hilbert space with inner product

\[
\langle X, Y \rangle_{L^2(\Omega, H)} = \mathbb{E}[\langle X, Y \rangle_H].
\]
CHAPTER 2. STOCHASTIC PROCESSES AND RANDOM FIELDS

Since \( \mathbb{R}^d \) and \( \mathbb{C}^d \) are Hilbert spaces, \( L^2(\Omega, \mathbb{R}^d) \) and \( L^2(\Omega, \mathbb{C}^d) \) are Hilbert spaces for all \( d \geq 1 \).

Since \( L^2(\Omega, H) \) is a Hilbert space, the Cauchy-Schwarz inequality applies and

\[
\langle X, Y \rangle_{L^2(\Omega, H)} \leq \|X\|_{L^2(\Omega, H)} \|Y\|_{L^2(\Omega, H)}.
\] (2.1)

**Definition 2.2** An \( \mathbb{R}^d \)-valued random variable \( X \) is of second order if \( X \in L^2(\Omega, \mathbb{R}^d) \).

We introduce some important results from probability theory which we use for our analysis.

**Lemma 2.3 (Jensen inequality)** Let \((X, \Sigma, \mu)\) be a measure space where \( \mu \) is a positive measure such that \( \mu(X) = 1 \). If \( f : X \to (a, b) \) is a real function for which \( \int_X f \, d\mu < \infty \) and if \( \phi : (a, b) \to \mathbb{R} \) is a convex function, then

\[
\phi(\int_X f \, d\mu) \leq \int_X (\phi \circ f) \, d\mu.
\] (2.2)

**Proof** See [69, p. 62]. \(\square\)

**Corollary 2.4 (Consequence of the deterministic Jensen inequality)** For \( p \geq 1 \),

\[
\left| \int_s^t f(r) \, dr \right|^p \leq |t - s|^{p-1} \int_s^t |f(r)|^p \, dr.
\] (2.3)

**Lemma 2.5 (The probabilistic Jensen inequality)** For a convex function \( \phi : \mathbb{R} \to \mathbb{R} \) and a real-valued random variable \( X \) satisfying \( \mathbb{E}[|X|] < \infty \), the inequality

\[
\phi(\mathbb{E}[X]) \leq \mathbb{E}[\phi(X)]
\] (2.4)

holds. In particular the result holds if we set \( \phi(x) = |x|^p \) for some \( p > 1 \).

**Proof** See [76, pp. 60–61]. \(\square\)

**Corollary 2.6 (Consequence of the probabilistic Jensen inequality)** Suppose that \( X \) is a non-negative random variable for which there exists \( p^* \geq 1 \) such that \( X \in L^{p^*}(\Omega) \) for all \( p \geq p^* \). Then \( X \in L^q(\Omega) \) for all \( q \geq 1 \).
**Proof** Suppose that $q \geq 1$ and $p \geq p^*$. We now bound $\|X\|_{L^q(\Omega)}$ as follows. Firstly, by applying the Jensen inequality with $\phi(x) = |x|^p$,

$$E[X^q]^p = |E[X^q]|^p \leq E[|X|^p]^q.$$ 

Hence, taking the $pq$-th root on each side,

$$\|X\|_{L^q(\Omega)} \leq \|X^{pq}\|_{\Omega}^{\frac{1}{pq}} = \|X\|_{L^{pq}(\Omega)}.$$ 

Since $pq \geq p \geq p^*$, the result follows. \hfill \Box

**Lemma 2.7 (The Chebyshev inequality)** Let $p \geq 1$ and let $X \in L^p(\Omega,H)$ be a $H$-valued random variable for some Hilbert space $H$. Then

$$P(\|X\|_H \geq R) \leq \frac{E[\|X\|_H^p]}{R^p} = \frac{\|X\|_{L^p(\Omega,H)}^p}{R^p}$$

for all $R > 0$.

**Proof** Let $Y = \|X\|_H$. The inequality

$$P(Y \geq R) \leq \frac{E[Y^p]}{R^p}$$

is proved in [1, pp. 126-127]. \hfill \Box

**Lemma 2.8 (The Borel–Cantelli Lemma)** For a probability space $(\Omega, \mathcal{F}, P)$, let $F_j \in \mathcal{F}$ for $j \geq 1$ be a collection of events. If $\sum_{j=1}^\infty P(F_j) < \infty$, then

$$P(\{\omega \in \Omega \mid F_j \text{ occurs for infinitely many } j\}) = 0,$$

or equivalently

$$P\left(\bigcap_{n=1}^\infty \bigcup_{j=n}^\infty F_j\right) = 0.$$ 

**Proof** See [1, p. 205]. \hfill \Box
2.1.1 The Gaussian distribution

We shall make extensive use of the $d$-dimensional Gaussian distribution $N(\mu, C)$, for some mean vector $\mu \in \mathbb{R}^d$ and symmetric non-negative definite covariance matrix $C \in \mathbb{R}^{d \times d}$, in order to simulate Gaussian random fields. We also need the one-dimensional Gaussian distribution in order to simulate various Gaussian stochastic processes such as the Wiener process and Brownian bridge process.

We state a result about the moments of the one-dimensional Gaussian random variable $N(0, \sigma^2)$, and a result about what happens when we want to know the distribution of a subvector of a Gaussian random vector.

**Proposition 2.9** For a Gaussian random variable $X \sim N(0, \sigma^2)$ and any $p \geq 0$,

$$E[X^{2p}] \leq \frac{(2p)! \sigma^{2p}}{2^p p!},$$

$$E[X^{2p+1}] = 0.$$

**Proof** See [77].

**Definition 2.10** Let $a \in \mathbb{R}^d$ be a vector and $A \in \mathbb{R}^{d \times d}$ be a matrix. An indexing vector $i$ is a vector of size $n$, where $n \leq d$, containing some combination of the natural numbers $1, \ldots, d$ in ascending order in which no number can appear more than once. The complement indexing vector $i'$ of $i$ is the vector of size $d - n$ containing all of the numbers $1, \ldots, d$ in ascending order which do not appear in $i$.

We then define the subvector $a_i$ as the vector formed by picking out the elements of $a$ corresponding to the elements in $i$, and the submatrix $A_{i,j}$ as the matrix formed by picking out the rows of $A$ corresponding to the elements in $i$, and the columns of $A$ corresponding to the elements in $j$. For example, if $d \geq 6$, $i = [2, 3, 5]$, and $j = [1, 2, 6]$, then

$$a_i = \begin{bmatrix} a_2 \\ a_3 \\ a_5 \end{bmatrix}, \quad A_{i,j} = \begin{bmatrix} a_{21} & a_{22} & a_{26} \\ a_{31} & a_{32} & a_{36} \\ a_{51} & a_{52} & a_{56} \end{bmatrix}.$$
Proposition 2.11 Let $X \sim N(\mu, C)$, for some mean vector $\mu \in \mathbb{R}^d$ and some covariance matrix $C \in \mathbb{R}^{d \times d}$. Then, for any indexing vector $i \in \mathbb{R}^n$ where $n \leq d$, $X_i \sim N(\mu_i, C_{ii})$.

Proof We can write

$$X = \begin{bmatrix} X_i \\ X' \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_i \\ \mu' \end{bmatrix}, \quad C = \begin{bmatrix} C_{ii} & C_{ii'} \\ C_{i'i} & C_{i'i'} \end{bmatrix},$$

where $i'$ is the complement of $i$. It is now obvious that $\mu_i$ is the mean vector of $X_i$. We can also write

$$C = \mathbb{E}[XX^T] = \mathbb{E} \begin{bmatrix} X_i \\ X' \end{bmatrix} \begin{bmatrix} X_i^T & X'_i \end{bmatrix} = \mathbb{E} \begin{bmatrix} X_i X_i^T & X_i X'_i \\ X'_i X_i^T & X'_i X'_i \end{bmatrix} = \begin{bmatrix} C_{ii} & C_{ii'} \\ C_{i'i} & C_{i'i'} \end{bmatrix},$$

and from this it is clear that the covariance matrix of $X_i$ is $C_{ii}$.

The complex Gaussian distribution will prove to be of use when simulating random fields using the circulant embedding method in Chapter 4. As we shall see, when simulating from $N(\mu, C)$ we need to factorise $C$ as either $C = R^T R$ for some real-valued matrix $R$, or $C = R^* R$ for some complex-valued matrix $R$. In the latter case we will need the complex Gaussian distribution.

Definition 2.12 A $\mathbb{C}^d$-valued random variable $Z$ follows the complex Gaussian distribution with mean vector $\mu$ and symmetric positive definite covariance matrix $C$, written $Z \sim \mathbb{C}N(\mu, C)$ if the $\mathbb{R}^d$-valued random variables $\Re(Z - \mu)$ and $\Im(Z - \mu)$ are independent and both follow the $N(0_d, \frac{C}{2})$ distribution.

We now look at how to sample a Gaussian random vector.

Simulating a Gaussian random vector

Suppose that we would like to sample an $\mathbb{R}^d$-valued Gaussian random vector $X \sim N(\mu, C)$, for some mean vector $\mu \in \mathbb{R}^d$ and a (symmetric) covariance matrix $C \in \mathbb{R}^{d \times d}$. In order to do so, we need to factorise $C$ as either $C = R^T R$, for some $R \in \mathbb{R}^{d \times d}$, or $C = R^* R$, for some $R \in \mathbb{C}^{d \times d}$.
If $C$ is positive definite we can use the Cholesky factorisation to factorise $C = R^T R$, where $R$ is an upper triangular matrix. If $C$ is only non-negative definite, it may still be possible to factorise $C$ as required. If $C$ is circulant, for example, $C = W_d \Lambda W_d^*$, where $W_d$ is the $d \times d$ Fourier matrix, $\Lambda$ is diagonal and all elements $\lambda_j$ of $\Lambda$ are greater than or equal to zero. Hence $C = R^* R$, where $R = \Lambda^{1/2} W_d$ and $\Lambda^{1/2}$ is diagonal and has entries $\sqrt{\lambda_j}$.

$C = R^T R$ for a real-valued matrix $R$

If a factorisation $C = R^T R$ exists for a real matrix $R$, we can sample $X$ as follows. We generate a random vector $Z = [Z_1, \ldots, Z_d]^T \sim N(0, I_d)$; we can do this easily since $Z_j \sim N(0, 1)$ and are independent, for $j = 1, \ldots, d$, and samples from the $N(0, 1)$ distribution can be found, for example, by using the MATLAB command `randn` or the NAG Toolbox for MATLAB routine `g05sk`. Now we let $X = \mu + R^T Z$; $X$ is a linear combination of Gaussian random variables and so is itself Gaussian. We see that

$$E[X] = E[\mu + R^T Z] = \mu + R^T E[Z] = \mu,$$

and the covariance matrix of $X$ is


since $E[ZZ^T] = I_d$ is the covariance matrix of $Z$. Hence $X \sim N(\mu, C)$ as required.

$C = R^* R$ for a complex-valued matrix $R$

If we factorise $C = R^* R$ where $R$ is complex valued, we have to modify the procedure slightly. If we generate a random vector $Z$ as above and let $X = R^* Z$, then $X$ will be complex-valued with mean vector zero and covariance matrix $C$; thus $\hat{X} \sim CN(0, C)$, and $\Re \hat{X}$ and $\Im \hat{X}$ will be independent with distributions $N(0, C/2)$. To get the correct covariance matrix, we instead set $Z = Z_1 + iZ_2$, where $Z_1$ and $Z_2$ both follow the $N(0, I_d)$ distribution (so $Z \sim CN(0, 2I_d)$), and let $\hat{X} = R^* Z$. Then

$$E[\hat{X}] = E[R^* (Z_1 + iZ_2)] = R^* E[Z_1 + iZ_2] = 0,$$

and

$$E[\hat{XX}^*] = E[R^* ZZ^* R] = R^* E[ZZ^*] R = R^* (2I) R = 2C,$$
and so \( \hat{X} \sim \mathbb{C}N(0, 2C) \). Hence \( \Re \hat{X} \) and \( \Im \hat{X} \) have distribution \( N(0, C) \). We have two samples of \( X \) by taking \( X_1 = \mu + \Re \hat{X} \) and \( X_2 = \mu + \Im \hat{X} \); \( X_1 \) and \( X_2 \) both have the desired distribution \( N(\mu, C) \).

### 2.2 Stochastic Processes

We are now in a position to define stochastic processes. In particular we introduce the Wiener process, which is the driving process of an SDE. We shall also study the Brownian bridge process, which can be used to simulate the Wiener process, and fractional Brownian motion, which we shall simulate using the circulant embedding method in Chapter 4.

**Definition 2.13** An \( \mathbb{R}^d \)-valued random function \( \{X(t) : t \in \mathcal{T}\} \), where \( \mathcal{T} \subset \mathbb{R} \), is a set of random variables on a probability space \((\Omega, \mathcal{F}, \mathbb{P})\) parameterised by the variable \( t \in \mathcal{T} \); \( X(t) \) is a random variable for all \( t \in \mathcal{T} \). We shall often abbreviate \( \{X(t) : t \in \mathcal{T}\} \) to just \( X(t) \). The random function \( X(t) \) is a stochastic process if the parameter \( t \) is interpreted as time.

The random function \( \{X(t) : t \in \mathcal{T}\} \) is of second order if \( X(t) \in L^2(\Omega, \mathbb{R}^d) \) for all \( t \in \mathcal{T} \).

In general, stochastic processes can be \( H \)-valued for any Hilbert space \( H \), but we shall restrict ourselves to real- and \( \mathbb{R}^d \)-valued stochastic processes. The variable \( t \) will usually be interpreted as time, and the set \( \mathcal{T} \) will usually be \( [0, T] \) for some \( T > 0 \). The deterministic function \( X(t, \omega) \), for a fixed \( \omega \in \Omega \), is a sample path of the stochastic process \( X(t) \). If a stochastic process \( X(t) \) is second-order, then it has a mean function \( \mu(t) = \mathbb{E}[X(t)] \) and a symmetric non-negative definite covariance function \( C(s, t) = \mathbb{E}[(X(t) - \mu(t))(X(s) - \mu(s))] \). Non-negative definiteness of a function is defined below.

**Definition 2.14** A function \( \phi : \mathcal{T} \times \mathcal{T} \to \mathbb{R} \), for some set \( \mathcal{T} \subset \mathbb{R} \), is non-negative definite if, for any \( N \in \mathbb{N} \), any \( a_1, \ldots, a_N \in \mathbb{C} \) and any \( t_1, \ldots, t_N \in \mathcal{T} \),

\[
\sum_{j=1}^N \sum_{k=1}^N a_j \bar{a}_k \phi(t_j, t_k) \geq 0.
\]

The function \( \phi \) is positive definite if the inequality is strict.
An important class of stochastic processes is the class of Gaussian stochastic processes.

**Definition 2.15** A real-valued stochastic process \( \{X(t) : t \in \mathcal{T}\} \) is a Gaussian stochastic process if, when sampled at \( N \) points \( t_j \in \mathcal{T} \), for any \( N \in \mathbb{N} \), the random vector \( \mathbf{X} \) with entries \( X_j = X(t_j) \) follows a multivariate Gaussian distribution.

The finite dimensional distributions of any Gaussian stochastic process are uniquely determined by its mean and covariance functions, and any mean function and valid (i.e. symmetric non-negative definite) covariance function uniquely determines a Gaussian stochastic process; see [46].

### 2.2.1 Martingales

Martingales form an important sub-class of stochastic processes. In particular, the Wiener process, defined in §2.2.2, and the Itô integral, defined in §5.1.1, are martingales.

Firstly we need some more basic definitions. We firstly define filtrations of a \( \sigma \)-algebra \( \mathcal{F} \), and then we define adapted and predictable random variables.

**Definition 2.16** For a probability space \( (\Omega, \mathcal{F}, \mathbb{P}) \), the filtration \( \{\mathcal{F}_t\}_{t \geq 0} \) is a family of sub \( \sigma \)-algebras of \( \mathcal{F} \) which satisfy the following:

1. \( \{\mathcal{F}_t\}_{t \geq 0} \) is increasing; for all \( s \geq t \), \( \mathcal{F}_s \) is a sub \( \sigma \)-algebra of \( \mathcal{F}_t \);

2. \( \{\mathcal{F}_t\}_{t \geq 0} \) is right-continuous; for all \( t \geq 0 \), \( \mathcal{F}_t = \bigcup_{\epsilon > 0} \mathcal{F}_{t-\epsilon} \);

3. Every null-set which is contained in \( \mathcal{F} \) is also contained in \( \mathcal{F}_t \) for all \( t \geq 0 \).

The filtration \( \{\mathcal{F}_t\}_{t \geq 0} \) will often be written simply as \( \mathcal{F}_t \).

We can extend the definition of the probability space \( (\Omega, \mathcal{F}, \mathbb{P}) \) to include the filtration \( \{\mathcal{F}_t\}_{t \geq 0} \), to form the filtered probability space \( (\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t \geq 0}, \mathbb{P}) \).

**Definition 2.17** An \( \mathbb{R}^d \)-valued stochastic process \( \mathbf{X}(t) \) is adapted to the filtration \( \mathcal{F}_t \) if, for all \( t \geq 0 \), \( \mathbf{X}(t) \) is an \( \mathcal{F}_t \)-measurable random variable. We can also say that \( \mathbf{X}(t) \) is \( \mathcal{F}_t \)-adapted.
Definition 2.18 An $\mathbb{R}^d$-valued stochastic process is predictable if, for all $t > 0$, $X(t)$ is $\mathcal{F}_{t^-}$-measurable, where

$$\mathcal{F}_{t^-} = \bigcap_{\epsilon > 0} \mathcal{F}_{t^-\epsilon}. $$

We note that if $X(t)$ is continuous and adapted, then it is predictable \[46, p. 57\].

Now we are in a position to define martingales.

Definition 2.19 Let $X(t)$ be an $\mathbb{R}^d$-valued stochastic process. $X(t)$ is a martingale if the conditional probability $E[X(t) | \mathcal{F}_s] = X(s)$ (or equivalently, $E[X(t) - X(s) | \mathcal{F}_s] = 0$) almost surely, for all $0 \leq s \leq t$.

We now define three important second-order Gaussian stochastic processes; the Wiener process, the Brownian bridge process, and fractional Brownian motion.

2.2.2 The Wiener process

The Wiener process is particularly important since it drives the solution of SDEs. It is often called ‘Brownian motion’ since it is a mathematical representation of Brownian motion, which was first observed by the botanist Robert Brown in 1827.

Definition 2.20 The Wiener process $\{W(t) : t \geq 0\}$ is a second-order Gaussian stochastic process satisfying all of the following:

1. Sample paths of $W(t)$ are continuous;
2. $W(0) = 0$ almost surely;
3. $W(t)$ has independent increments; for $0 < r < s \leq t < u$, the increments $W(s) - W(r)$ and $W(u) - W(t)$ are independent;
4. $W(t) - W(s) \sim N(0, |t - s|)$ for all $s, t > 0$.

If $W(t)$ is adapted to the filtration $\mathcal{F}_t$, then $W(t)$ is an $\mathcal{F}_t$-adapted Wiener process.

The Wiener process $W(t)$ has mean function $\mu(t) = 0$ and covariance function $C(s, t) = \min(s, t)$; hence $\text{Var}[W(t)] = C(t, t) = t$ for all $t \geq 0$. There are several ways to simulate the Wiener process; we shall explore two of them in \[2.2.5\] and a third method in Chapter \[7\]. The Wiener process $W(t)$ also satisfies two other properties:
• It is \emph{self-similar}; the process $\bar{W}(t) = \alpha \frac{1}{\alpha} W\left(\frac{t}{\alpha}\right)$ is also a Wiener process for all $\alpha > 0$. This can be proven by checking that $\bar{W}(t)$ has mean function $\mu(t) = 0$ and covariance function $C(s, t) = \min(s, t)$;

• The increments of $W(t)$ are \emph{stationary}; both $W(t) - W(s)$ and $W(t+h) - W(s+h)$ follow the $N(0, |t - s|)$ distribution for any $h, s, t > 0$. This property follows immediately from Definition 2.20.

\textbf{The $m$-dimensional Wiener process}

For many applications we shall need a multi-dimensional Wiener process $W(t)$.

\textbf{Definition 2.21} The $m$-dimensional Wiener process $\{W(t) : t \geq 0\}$ is the Gaussian stochastic process with pairwise independent components $W_j(t)$, where $j = 1, \ldots, m$ and each $W_j(t)$ is a one-dimensional Wiener process.

Since all the components of a multi-dimensional Wiener process $W_j(t)$ are independent, we can simulate each component $W_j(t)$ of $W(t)$ independently.

\textbf{2.2.3 The Brownian bridge process}

If we take the Wiener process in $[0, T]$ and fix the value of $W(T)$, the resulting stochastic process is a Brownian bridge process. For the standard Brownian bridge process, $T = 1$ and $W(1) = 0$. We can define the standard Brownian bridge process formally as follows:

\textbf{Definition 2.22} The standard Brownian bridge process $\{B(t) : t \in [0, 1]\}$ is the (unique) second-order Gaussian stochastic process with mean function $\mu(t) = 0$ and covariance function $C(s, t) = \min(s, t) - st$ (and hence $\text{Var}[B(t)] = t(1 - t)$).

We can modify the standard Brownian bridge in order to satisfy different boundary conditions; in general we would like to create a stochastic process $\{\bar{B}(t) : t \in [t_{\text{min}}, t_{\text{max}}]\}$ which is a linear transformation of the standard Brownian bridge process $B(t)$ such that $\bar{B}(t_{\text{min}}) = \alpha$ and $\bar{B}(t_{\text{max}}) = \beta$. If we let

$$
\bar{B}(t) = \alpha + (\beta - \alpha) \frac{t - t_{\text{min}}}{t_{\text{max}} - t_{\text{min}}} + \sqrt{t_{\text{max}} - t_{\text{min}}} B\left(\frac{t - t_{\text{min}}}{t_{\text{max}} - t_{\text{min}}}\right),
$$

...
then $\bar{B}(t)$ satisfies the required boundary conditions. The new process $\bar{B}(t)$ has mean function $\mu(t) = \alpha + (\beta - \alpha) \frac{t - t_{\min}}{t_{\max} - t_{\min}}$ and covariance function $C(s, t) = \min(s, t) - t_{\min} - \frac{(s - t_{\min})(t - t_{\min})}{t_{\max} - t_{\min}}$. In particular, $\text{Var}[\bar{B}(t)] = \frac{(t - t_{\min})(t_{\max} - t)}{t_{\max} - t_{\min}}$.

The Brownian bridge is particularly useful when we have simulated the Wiener process $W(t)$ at times $t_{\min}$ and $t_{\max}$, and need to simulate $W(t)$ at a time $s \in (t_{\min}, t_{\max})$. We may do this if we are solving an SDE numerically, since:

- We may have used one stepsize (and/or error control, if using adaptive stepsize) to solve an SDE, and now want to refine the solution using a smaller stepsize (and/or tighter error control), using the same sample path of $W(t)$;
- If using adaptive stepsize, we may reject a stepsize and have to repeat the stepsize using a smaller stepsize.

### 2.2.4 Fractional Brownian Motion

Fractional Brownian motion is a Gaussian stochastic process with similar properties to Brownian motion. We denote it by $B_H(t)$, where $H \in (0, 1)$ is a parameter known as the Hurst parameter and define it formally as follows.

**Definition 2.23** Fractional Brownian motion, with Hurst parameter $H \in (0, 1)$, is the (unique) second-order Gaussian stochastic process $\{B_H(t) : t \geq 0\}$ with mean function $\mu(t) = 0$ and covariance function

$$C(s, t) = \frac{1}{2} \left( |t|^{2H} + |s|^{2H} - |t - s|^{2H} \right). \quad (2.6)$$

Note that for $H = \frac{1}{2}$, $C(s, t) = \frac{1}{2} \left( |t| + |s| - |t - s| \right) = \min(s, t)$ and hence $B_{\frac{1}{2}}(t) = W(t)$, the Wiener process.

It is easy to prove [46] that $B_H(t)$ satisfies the following properties, also satisfied by the Wiener process:

- $B_H(t)$ is self-similar; for any $\alpha > 0$, the stochastic process

  $$\bar{B}^H(t) = \alpha^H B_H \left( \frac{t}{\alpha} \right)$$

  is also a fractional Brownian motion;
• $B^H(t)$ has stationary increments: $B^H(t) - B^H(s)$ and $B^H(t + h) - B^H(s + h)$ follow the same probability distribution for any $s, t, h > 0$.

The major feature differentiating fractional Brownian motion from the Wiener process is that disjoint increments are not independent in general. To see this, we calculate the covariance of the increments $B^H(s) - B^H(r)$ and $B^H(u) - B^H(t)$, for $0 \leq r < s \leq t < u$ and $H \neq \frac{1}{2}$. We have

$$
\begin{align*}
\mathbb{E}[(B^H(s) - B^H(r))(B^H(u) - B^H(t))] &= \mathbb{E}[B^H(s)B^H(u)] - \mathbb{E}[B^H(r)B^H(u)] - \mathbb{E}[B^H(s)B^H(t)] + \mathbb{E}[B^H(r)B^H(t)] \\
&= \frac{1}{2} (s^{2H} + u^{2H} - |s - u|^{2H} - r^{2H} - |r|^{2H} + |r - u|^{2H}) \\
&\quad - |s|^{2H} - |t|^{2H} + |s - t|^{2H} + |r|^{2H} + |t|^{2H} - |s - t|^{2H}) \\
&= \frac{1}{2} (|u - r|^{2H} + |t - s|^{2H} - |u - s|^{2H} - |t - r|^{2H}) \neq 0.
\end{align*}
$$

Hence the increments are correlated and therefore not independent. In fact, the covariance for the increments $B^H(s) - B^H(r)$ and $B^H(u) - B^H(t)$ is negative for $H < \frac{1}{2}$, and positive for $H > \frac{1}{2}$.

We do not need fractional Brownian motion to solve SDEs; however it is a driving process for other types of differential equations. We shall look at a method to simulate fractional Brownian motion in chapter 4.

### 2.2.5 Simulating the Wiener process

We shall discuss two simple methods to simulate the multidimensional Wiener process $W(t)$ at a point $t_{\text{new}} > 0$. The first is based on simulating each increment as a Gaussian random variable, and the second is based on using the Brownian bridge. It is often necessary to combine the two methods; for example if using the Brownian bridge, we need to know the value of $W(T)$ for a time $T > t_{\text{new}}$.

Throughout this section, $t_-$ will denote the largest time smaller than $t_{\text{new}}$ for which $W(t)$ has been simulated, and $t_+$ will denote the smallest time larger than $t_{\text{new}}$ for which $W(t)$ has been simulated, if such a time exists. We shall usually store the values of $t$ in a row vector $t$ with length $N$ and entries $t_j$ for $j = 1, \ldots, N$, where $N$ is the number of points of $W(t)$ which have been generated, and the values of $W(t)$ in an $m \times N$ matrix $W$ with columns $W(t_j)$ for $j = 1, \ldots, N$. The values in $t$ shall be kept
in ascending order, with the columns of $\mathbf{W}$ also being sorted if necessary to keep the association between entries of $t$ and columns of $\mathbf{W}$.

**Simulating increments of the Wiener process**

The most direct way to simulate $\mathbf{W}(\mathbf{t}_{\text{new}})$ is to simulate the increment $\Delta \mathbf{W}(t_- \to t_{\text{new}}) = \mathbf{W}(t_{\text{new}}) - \mathbf{W}(t_-)$. This is possible only if there is no time $t_+ > t_{\text{new}}$ for which $\mathbf{W}(t_+)$ is known. We know that $\Delta W_j(t_- \to t_{\text{new}})$, the $j$-th component of $\Delta \mathbf{W}(t_- \to t_{\text{new}})$, has distribution $\mathcal{N}(0, t_{\text{new}} - t_-)$ by Definition 2.20 and so we have the following algorithm for simulating $\mathbf{W}(t_{\text{new}})$:

- Find the largest time $t_- < t_{\text{new}}$ for which $\mathbf{W}(t_-)$ is known;
- Simulate a Gaussian random vector $\mathbf{Z}$ with components $Z_j \sim \mathcal{N}(0, 1)$ - this can be done using the MATLAB routine `randn` or the NAG Toolbox for MATLAB routine `g05sk`;
- Calculate $\Delta \mathbf{W}(t_- \to t_{\text{new}}) = \sqrt{t_{\text{new}} - t_-} \mathbf{Z}$;
- Finally set $\mathbf{W}(t_{\text{new}}) = \mathbf{W}(t_-) + \Delta \mathbf{W}(t_- \to t_{\text{new}})$, and add $t_{\text{new}}$ as an entry in $\mathbf{t}$ and $\mathbf{W}(t_{\text{new}})$ as a column in $\mathbf{W}$.

**Using the Brownian bridge**

If $\mathbf{W}(t_+)$ is known at a time $t_+ > t_{\text{new}}$ then we cannot simulate the increment $\Delta \mathbf{W}(t_- \to t_{\text{new}})$ in the same way as above. Instead we must use the Brownian bridge to generate $\mathbf{W}(t_{\text{new}})$. The procedure is:

- Find the largest time $t_- < t_{\text{new}}$ for which $\mathbf{W}(t_-)$ is known, and the smallest time $t_+ > t_{\text{new}}$ for which $\mathbf{W}(t_+)$ is known. It may be necessary to use a searching routine to do this; for example the `find` function in MATLAB, or the NAG Toolbox for MATLAB routine `m01na`.
- Simulate a Gaussian random vector $\mathbf{Z}$ with components $Z_j \sim \mathcal{N}(0, 1)$;
- Calculate $\mathbf{Y} = \sqrt{\frac{(t_{\text{new}} - t_-)(t_+ - t_{\text{new}})}{t_+ - t_-}} \mathbf{Z}$;
Finally set $W(t_{\text{new}}) = W(t_{\text{old}}) + (W(t_{\text{old}}) - W(t_{\text{new}})) \frac{t_{\text{new}} - t_{\text{old}}}{t_{\text{new}} - t_{\text{old}}} + Y$, and add $t_{\text{new}}$ as an entry in $t$ and $W(t_{\text{new}})$ as a column in $W$, making sure that the entries in $t$ are in ascending order and the entries in $W$ are in the corresponding order. One way to do this is to add the new entries at the end of $t$ and $W$, then rank the entries of $t$ using $\text{m01da}$ and finally sort both $t$ and $W$ in the correct order using $\text{m01ea}$.

### 2.3 Random Fields

Random fields are a generalisation of stochastic processes. Instead of being parameterised by a scalar variable which is usually a time variable, they are parameterised by a vector variable which usually denotes a point in a spatial domain. Random fields are often used to model geological properties of rocks such as porosity or permeability [3,36], in order to solve partial differential equations with random data. For example, [2] solves equations of the form

$$L(a(x,\omega))u(x,\omega) = f(x,\omega) \quad \text{in } D$$

for some domain $D \subset \mathbb{R}^d$, where $L$ is an elliptic operator, $a(x,\omega)$ is a random field, $f(x,\omega)$ is known and $u(x,\omega)$ is the unknown quantity of interest.

**Definition 2.24** A $d$-dimensional random field $\{Z(x) : x \in D\}$, usually abbreviated to $Z(x)$, is a set of real-valued random variables on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ parameterised by the variable $x \in D \subset \mathbb{R}^d$.

The random field $\{Z(x) : x \in D\}$ is of second order if $Z(x) \in L^2(\Omega)$ for all $x \in D$.

For a given $\omega \in \Omega$, the deterministic function $Z(x,\omega)$ is a realisation of the random field $Z(x)$. Like stochastic processes, second-order random fields have a mean function $\mu(x) = \mathbb{E}[Z(x)]$ and a symmetric non-negative definite covariance function $C(x,y) = \mathbb{E}[(Z(x) - \mu(x))(Z(y) - \mu(y))]$. We can also classify Gaussian random fields.

**Definition 2.25** A random field $\{Z(x) : x \in D\}$ is Gaussian if, when sampled at any $N$ points $x_j \in D$ for any $N \in \mathbb{N}$, the random vector $X$ with entries $X_j = Z(x_j)$ follows a multivariate Gaussian distribution.

Random fields can be stationary in the strong sense and/or stationary in the wide sense.
Definition 2.26 A random field $Z(x)$ is stationary in the strong sense if, for any $n \in \mathbb{N}$,

$$
P(Z(x_1 + x) \in B_1, \ldots, Z(x_n + x) \in B_n) = P(Z(x_1) \in B_1, \ldots, Z(x_n) \in B_n),$$

for all $x, x_j \in \mathbb{R}^d$ and all $B_j \in \mathcal{B}$, where $\mathcal{B}$ is the Borel $\sigma$-algebra of $\mathbb{R}^d$.

A second-order random field $Z(x)$ is stationary in the wide sense if its mean function $\mu(x)$ is constant for all $x \in \mathbb{R}^d$ and the covariance function $C(x, y)$ satisfies

$$
C(x + t, y + t) = C(x, y),
$$

(2.7)

for all $x, y, t \in \mathbb{R}^d$.

Stationarity in the wide sense allows us to write the covariance function as a function of one variable, as follows:

Definition 2.27 Any covariance function $C(x, y)$ satisfying (2.7) for all $x, y, t \in \mathbb{R}^d$ satisfies $C(x, y) = C(x - y, 0)$ for all $x, y \in \mathbb{R}^d$. Hence we can write $C(x, y)$ as a function of one variable $\gamma(x - y) = C(x, y)$; the function $\gamma : \mathbb{R}^d \to \mathbb{R}$ is called a variogram. Since the covariance function $C(x, y)$ is symmetric, the variogram $\gamma$ must also be symmetric about the origin; i.e. $\gamma(t) = \gamma(-t)$ for all $t \in \mathbb{R}^d$.

It is possible to use any positive definite variogram as a covariance function for a stationary Gaussian random field. A list of criteria determining whether a candidate variogram is positive definite, and a list of possible variograms, are given in [72, §2.1, §3].

For a Gaussian random field, stationarity in the strong and wide senses are equivalent; however this is not the case in general [72]. If a Gaussian random field is stationary in the strong and wide senses, we shall simply refer to the random field as being stationary. There are many methods for simulating Gaussian random fields, including the turning bands method [50] and the circulant embedding method [12,20], which we shall study in Chapter 4.

One consequence of a Gaussian random field $Z(x)$ being stationary is that if we were to sample it on a domain $D \in \mathbb{R}^d$, any realisation of $Z(x)$ would also be a valid realisation of the random field $Z(x + y)$ over the domain $D' = D + y$. Thus when
simulating \( Z(\mathbf{x}) \), we can choose a convenient domain \( D + \mathbf{y} \) and simulate \( Z(\mathbf{x} + \mathbf{y}) \) instead.

The variogram \( \gamma \) must be symmetric, but it is possible for it to be uneven in one or more coordinates.

**Definition 2.28** A variogram \( \gamma \) is even in the \( k \)-th coordinate if

\[
\gamma \left( \begin{bmatrix} t_1 \\ \vdots \\ -t_k \\ t_d \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} t_1 \\ \vdots \\ t_k \\ t_d \end{bmatrix} \right),
\]

for all \( \mathbf{t} = [t_1, \ldots, t_d]^T \in \mathbb{R}^d \), and uneven in the \( k \)-th coordinate if there exists some \( \mathbf{t} \in \mathbb{R}^d \) for which (2.8) does not hold.

Clearly if \( d = 1 \), a variogram must be even in its only coordinate. Note that it is impossible for a variogram to be even in all but one coordinate, which we prove by contradiction. Without loss of generality, suppose that the variogram \( \gamma(\mathbf{t}) \) is even in its first \( d - 1 \) coordinates but uneven in its \( d \)-th coordinate. Then, since \( \gamma(-\mathbf{t}) = \gamma(\mathbf{t}) \), there exists some \( \mathbf{t} \in \mathbb{R}^d \) such that

\[
\gamma \left( \begin{bmatrix} t_1 \\ \vdots \\ -t_{d-1} \\ t_d \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} -t_1 \\ \vdots \\ -t_{d-1} \\ t_d \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} t_1 \\ \vdots \\ t_{d-1} \\ -t_d \end{bmatrix} \right) \neq \gamma \left( \begin{bmatrix} t_1 \\ \vdots \\ t_{d-1} \\ t_d \end{bmatrix} \right),
\]

which is a contradiction. Similarly it is impossible for a variogram to be even in only one coordinate - without loss of generality, suppose \( \gamma(\mathbf{t}) \) is even in its first coordinate and uneven in the other \( d - 1 \) coordinates. Again using \( \gamma(-\mathbf{t}) = \gamma(\mathbf{t}) \) there exists some \( \mathbf{t} \in \mathbb{R}^d \) such that

\[
\gamma \left( \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_d \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} -t_1 \\ t_2 \\ \vdots \\ t_d \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} t_1 \\ -t_2 \\ \vdots \\ -t_d \end{bmatrix} \right) \neq \gamma \left( \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_d \end{bmatrix} \right),
\]
again a contradiction. If $d = 2$, this means that either the variogram is even in both coordinates, or uneven in both coordinates. So in this case we can simply refer to the variogram as *even* or *uneven*. 
Chapter 3

Linear Algebra for the Circulant Embedding Method

We introduce some concepts from linear algebra which will be needed in Chapter 4. In particular we discuss Discrete Fourier Transforms, Toeplitz and circulant matrices, block Toeplitz and block circulant matrices, and matrix factorisations. We shall see that we can embed Toeplitz and block Toeplitz matrices into circulant and block circulant matrices, and that circulant and block circulant matrices have a convenient factorisation in terms of Fourier matrices. This forms the basis for the circulant embedding method for simulating random fields, which is discussed in Chapter 4.

We start by introducing some basic linear algebra notation which shall be needed later.

General notation

For the whole of this chapter we shall start indices for vectors and matrices from 0 rather than 1 unless stated otherwise. This simplifies notation, especially when discussing Discrete Fourier Transforms and circulant matrices.

We shall define $\mathbf{0}_m$ to be the column vector of length $m$ with all entries zero, and similarly the matrix $0_{mn}$ to be the $m \times n$ matrix with all entries zero. The $m \times m$ identity matrix shall be denoted $I_m$. 

44
Vectorisation function

For a matrix $A \in \mathbb{R}^{m \times n}$ with columns $a_j$, for $j = 0, \ldots, n-1$, we define the vectorisation function

$$\text{vec}(A) = \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix}.$$  

(3.1)

Kronecker product

The Kronecker product of two matrices $A \in \mathbb{R}^{m_1 \times n_1}$ and $B \in \mathbb{R}^{m_2 \times n_2}$ is given by

$$A \otimes B = \begin{bmatrix} a_{00}B & a_{01}B & \ldots & a_{0,n_1-1}B \\ a_{10}B & a_{11}B & \ldots & a_{1,n_1-1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1-1,0}B & a_{m_1-1,1}B & \ldots & a_{m_1-1,n_1-1}B \end{bmatrix},$$

(3.2)

and has dimension $m_1m_2 \times n_1n_2$.

Now we introduce Discrete Fourier Transforms.

### 3.1 Discrete Fourier Transforms

#### One-dimensional Fourier matrix and Discrete Fourier Transform

We shall use the Discrete Fourier Transform (DFT) extensively when simulating Gaussian random fields using the circulant embedding method in Chapter 4. We can define a matrix called the Fourier matrix to give us an easy way to express the DFT.

**Definition 3.1** We define the Fourier matrix of size $n \times n$ as the matrix $W_n$ with entries

$$w_{j,k} = \frac{\omega_n^{-j,k}}{\sqrt{n}}, \quad \omega_n = e^{\frac{2\pi i}{n}},$$

(3.3)

for $j, k = 0, \ldots, n - 1$. 

The Fourier matrix $W_n$ is unitary: if we let $w_j$ and $w_k$ be the $j$-th and $k$-th columns of $W_n$ respectively, then the $(j,k)$ entry of $W_n^*W_n$ is

$$w_j^*w_k = \sum_{\ell=0}^{n-1} \bar{w}_{\ell,j} w_{\ell,k} = \frac{1}{n} \sum_{\ell=0}^{n-1} \omega_n^{fj} \omega_n^{-\ell k} = \frac{1}{n} \sum_{\ell=0}^{n-1} \omega_n^{(j-k)} = \delta_{j,k},$$

(3.4)

and so $W_n^*W_n = I$. We can show that $W_nW_n^* = I$ by a similar argument.

We can now define the Discrete Fourier Transform, along with its inverse.

**Definition 3.2** We define the Discrete Fourier Transform (DFT) of a vector $\mathbf{u} \in \mathbb{R}^n$ as the vector $\hat{\mathbf{u}} = W_n \mathbf{u}$. The elements $\hat{u}_j$ of $\hat{\mathbf{u}}$, $j = 0, \ldots, n - 1$, are given by

$$\hat{u}_j = \sum_{k=0}^{n-1} w_{j,k} u_k = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} u_k \omega_n^{-jk}.$$  

(3.5)

The Inverse Discrete Fourier Transform (IDFT) of $\mathbf{u}$ is the vector $\check{\mathbf{u}} = W_n^* \mathbf{u}$; the elements $u_j$ of $\check{\mathbf{u}}$, $j = 0, \ldots, n - 1$ are given by

$$\check{u}_j = \sum_{k=0}^{n-1} \bar{w}_{j,k} u_k = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} u_k \omega_n^{jk}.$$  

(3.6)

**Note**

The DFT can also be defined with the scaling factor of $\frac{1}{\sqrt{n}}$ removed, and the corresponding inverse transform will have a scaling factor of $\frac{1}{n}$. This scaling is implemented by the MATLAB routines `fft` and `ifft`, which use an algorithm called the Fast Fourier Transform (FFT). The FFT algorithm is described in, for example, [5].

It is clear that $\hat{\mathbf{u}}$ and $\check{\mathbf{u}}$, the DFT and IDFT of $\mathbf{u}$ respectively, are complex conjugates of each other. We now define Hermitian vectors and matrices, so that we can prove a fact about the DFT and IDFT.

**Definition 3.3** We define a vector $\mathbf{u} \in \mathbb{R}^n$ to be a Hermitian vector if

1. $u_0$ is real, and

2. $u_j = \overline{u}_{n-j}$ for $j = 1, \ldots, n - 1$ (if $n$ is even then this implies that $u_{\frac{n}{2}}$ is real).
This definition is based on the NAG Fortran Library documentation [64, C06 Chapter Introduction], which uses the name Hermitian sequence rather than Hermitian vector. A matrix $U$ is a Hermitian matrix if $U = U^*$, where $U^*$ is the conjugate transpose of $U$.

We now have the following result.

**Proposition 3.4** The DFT and IDFT of a real vector is Hermitian, and the DFT and IDFT of a Hermitian vector is real.

**Multi-dimensional Fourier matrices and Discrete Fourier Transforms**

We can also define a multi-dimensional DFT and IDFT - we shall use the two-dimensional DFT in Chapter 4. Firstly we define the multi-dimensional Fourier matrix, which helps us express the multi-dimensional DFT.

**Definition 3.5** We define the $d$-dimensional Fourier matrix of sizes $n_1, \ldots, n_d$ as the matrix $W_{n_1, \ldots, n_d}$ where

$$W_{n_1, \ldots, n_d} = W_{n_d} \otimes W_{n_{d-1}} \otimes \cdots \otimes W_{n_1}.$$  \hspace{1cm} (3.7)

Note that

$$W_{n_1, \ldots, n_d}^* = W_{n_1}^* \otimes W_{n_{d-1}}^* \otimes \cdots \otimes W_{n_1}^*.$$  

In particular, we shall use the two-dimensional Fourier matrix $W_{m,n}$ of sizes $m$ and $n$ often. It is given by

$$W_{m,n} = W_n \otimes W_m$$  \hspace{1cm} (3.8)

and has entries

$$(W_{m,n})_{j+m,k+p+m} = \frac{\omega_m^{-j} \omega_n^{-k}}{\sqrt{mn}}, \quad \omega_m = \exp \left( \frac{2\pi i}{m} \right), \quad \omega_n = \exp \left( \frac{2\pi i}{n} \right).$$  \hspace{1cm} (3.9)

**Definition 3.6** The $d$-dimensional DFT of an array $u \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ is the array $\hat{u} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$, where

$$\hat{u}_{j_1, \ldots, j_d} = \frac{1}{\sqrt{\prod_{p=1}^{d} n_p}} \sum_{k_1=0}^{n_1-1} \cdots \sum_{k_d=0}^{n_d-1} u_{k_1, \ldots, k_d} \prod_{p=1}^{d} \omega_{n_p}^{-jp_k},$$

and the $d$-dimensional IDFT $\check{u}$ of $u$ is

$$\check{u}_{j_1, \ldots, j_d} = \frac{1}{\sqrt{\prod_{p=1}^{d} n_p}} \sum_{k_1=0}^{n_1-1} \cdots \sum_{k_d=0}^{n_d-1} u_{k_1, \ldots, k_d} \prod_{p=1}^{d} \omega_{n_p}^{jp_k}.$$
For a matrix $u \in \mathbb{R}^{m \times n}$ we can also define the two-dimensional DFT of the vector $v = \text{vec}(u)$ as $\hat{v} = W_{m,n}v$, and the two-dimensional IDFT of $v$ as $\check{v} = W_{m,n}^*v$. Note that $u_{j_1,j_2} = v_{j_1+m,j_2}$, and similarly for $\hat{u}$ and $\check{u}$, so

$$\hat{v}_{j_1+m,j_2} = \frac{1}{\sqrt{mn}} \sum_{k_1=0}^{m-1} \sum_{k_2=0}^{n-1} v_{k_1+k_2} \omega_{m}^{-jk_1} \omega_{n}^{-jk_2},$$

and

$$\check{v}_{j_1+m,j_2} = \frac{1}{\sqrt{mn}} \sum_{k_1=0}^{m-1} \sum_{k_2=0}^{n-1} v_{k_1+k_2} \omega_{m}^{jk_1} \omega_{n}^{jk_2}.$$ 

### 3.2 Toeplitz and Circulant Matrices

Toeplitz matrices appear when simulating a one-dimensional stochastic process, with a given covariance function, on a regularly spaced grid. A Toeplitz matrix can in turn be embedded into a circulant matrix; this forms the basis of the circulant embedding method for sampling a random field, described in Chapter 4. Here we introduce Toeplitz and circulant matrices, and how to embed a Toeplitz matrix into a circulant matrix.

**Definition 3.7** An $m \times m$ matrix $A$ is Toeplitz if its entries are constant on every diagonal; that is, if $A$ has the form

$$A = \begin{bmatrix}
a_0 & a_1 & a_2 & \ldots & a_{m-1} \\
a_{-1} & a_0 & a_1 & \ldots & a_{m-2} \\
a_{-2} & a_{-1} & a_0 & \ldots & a_{m-3} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{-m+1} & a_{-m+2} & a_{-m+3} & \ldots & a_0
\end{bmatrix} \quad \text{(3.10)}$$

for some $a_{-m+1}, \ldots, a_{m-1}$. Any Toeplitz matrix can be constructed from its first row and its first column. If $a_{-j} = a_j$ for $j = 1, \ldots, m - 1$, then $A$ is symmetric; $A$ is then given by

$$A = \begin{bmatrix}
a_0 & a_1 & a_2 & \ldots & a_{m-1} \\
a_1 & a_0 & a_1 & \ldots & a_{m-2} \\
a_2 & a_1 & a_0 & \ldots & a_{m-3} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
a_{-m+1} & a_{-m+2} & a_{-m+3} & \ldots & a_0
\end{bmatrix}. \quad \text{(3.11)}$$
Any symmetric Toeplitz matrix can be constructed from either its first row or first column, which are both the same.

Let $a^r = [a_0, \ldots, a_{m-1}] \in \mathbb{R}^m$ be the first row of the matrices $A$ in (3.10) and (3.11), and let $a^c = [a_0, a_{-1}, \ldots, a_{-m+1}] \in \mathbb{R}^m$ be the first column of the matrix $A$ in (3.10). We shall use the notation $\text{Toe}(a^r, a^c)$ to denote the Toeplitz matrix in (3.10), and $\text{symToe}(a^r) = \text{Toe}(a^r, a^r)$ to denote the symmetric Toeplitz matrix in (3.11).

**Definition 3.8** An $m' \times m'$ matrix $B$ is circulant if each of its rows is the same as the previous row, but periodically shifted one place to the right. Thus $B$ has the form

$$B = \begin{bmatrix}
    b_0 & b_1 & b_2 & \ldots & b_{m' - 1} \\
    b_{m' - 1} & b_0 & b_1 & \ldots & b_{m' - 2} \\
    b_{m' - 2} & b_{m' - 1} & b_0 & \ldots & b_{m' - 3} \\
    \vdots & \ddots & \ddots & \ddots & \vdots \\
    b_1 & b_2 & b_3 & \ddots & b_0
\end{bmatrix}.) \quad (3.12)$$

$B$ is also Toeplitz according to Definition 3.7 with $b_{-j} = b_{m' - j}$, for $j = 1, \ldots, m' - 1$, and can be formed from its first row, which we denote $b$.

For a row vector $b$, we shall also use the notation $\text{circ}(b)$ to denote the circulant matrix with $b$ as its first row.

### 3.3 Embeddings of Toeplitz matrices into circulant matrices

Any Toeplitz matrix can be embedded into any circulant matrix in a variety of different ways. Symmetric Toeplitz matrices are easier to embed than non-symmetric ones, so we deal with the symmetric case first.

**Minimal embedding of a symmetric Toeplitz matrix into a symmetric circulant matrix**

Suppose that we wish to embed an $m \times m$ symmetric Toeplitz matrix $A = \text{symToe}(a^r)$, where $a^r \in \mathbb{R}^m$, into a symmetric circulant matrix $B$; by necessity $B$ will have the
form

\[
B = \begin{bmatrix} A \end{bmatrix} = \begin{bmatrix}
 a_0 & a_1 & a_2 & \ldots & a_{m-1} \\
 a_1 & a_0 & a_1 & \ldots & a_{m-2} \\
 a_2 & a_1 & a_0 & \ldots & a_{m-3} \\
 \vdots & \ddots & \ddots & \ddots & \vdots \\
 a_{m-1} & a_{m-2} & a_{m-3} & \ldots & a_0 \\
 \end{bmatrix}.
\]

Let \( b \) denote the first row of the new circulant matrix \( B \). In order to make \( B \) circulant, the last entry of \( b \) must be the first entry of the second row of \( B \) (i.e. \( a_1 \)). Next, the penultimate entry of \( b \) must be the first entry of its third row (i.e. \( a_2 \)), and so on. We keep filling in \( b \) backwards until we reach the point where we must set an entry to the first entry of the \((m - 1)\)-th row of \( B \) (i.e. \( a_{m-2} \)), where we can stop. This entry will be the \((m + 1)\)-th entry of \( b \), which now reads

\[ b = [a_0, a_1, \ldots, a_{m-2}, a_{m-1}, a_{m-2}, \ldots, a_2, a_1], \]

and the matrix \( B \) will be

\[
B = \text{circ}(b) = \begin{bmatrix}
 a_0 & a_1 & a_2 & \ldots & a_{m-1} | a_{m-2} & a_{m-3} & \ldots & a_1 \\
 a_1 & a_0 & a_1 & \ldots & a_{m-2} | a_{m-1} & a_{m-2} & \ldots & a_2 \\
 a_2 & a_1 & a_0 & \ldots & a_{m-3} | a_{m-2} & a_{m-3} & \ldots & a_3 \\
 \vdots & \ddots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots \\
 a_{m-1} & a_{m-2} & a_{m-3} & \ldots & a_0 | a_1 & a_2 & \ldots & a_{m-2} \\
 a_{m-2} & a_{m-1} & a_{m-2} & \ldots & a_1 | a_0 & a_1 & \ldots & a_{m-3} \\
 a_{m-3} & a_{m-2} & a_{m-1} & \ldots & a_2 | a_1 & a_0 & \ldots & a_{m-4} \\
 \vdots & \ddots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots \\
 a_1 & a_2 & a_3 & \ldots & a_{m-2} | a_{m-3} & a_{m-4} & \ldots & a_0
\end{bmatrix}.
\]

The matrix \( B \) is now the minimal circulant embedding (MCE) of \( A \). Note that the size of \( B \) is \( 2(m - 1) \times 2(m - 1) \). This is the minimum size that a circulant matrix \( B \) must be in order to embed \( A \) correctly. Also note that every \( m \times m \) submatrix \( B_{ii} \) of \( B \), where \( i \) is an indexing vector containing \( m \) consecutive elements of the set \( \{1, \ldots, 2(m - 1)\} \), is equal to \( A \) (see Definition 2.10 for the definition of an indexing vector).
Other embeddings of symmetric Toeplitz matrices into symmetric circulant matrices

An alternative way of embedding $A$ in a circulant matrix is to let $B' = \text{circ}(b')$, where

$$b' = [a_0, a_1, \ldots, a_{m-2}, a_{m-1}, a_{m-2}, \ldots, a_2, a_1].$$

Here we are adding an extra copy of $a_{m-1}$ into the first row. $B'$ would also be an embedding matrix of $A$, but would be larger ($(2m - 1) \times (2m - 1)$ instead of $2(m - 1) \times 2(m - 1)$) - note that its size is now odd. We call this the minimal odd circulant embedding (MOCE) of $A$.

There is also the possibility of using padding to create a larger circulant matrix which embeds $A$. This is achieved by first embedding $A$ into a larger symmetric Toeplitz matrix $\bar{A}$ of size $(m + p) \times (m + p)$, and then performing either the MCE or MOCE of $\bar{A}$.

To clarify, let $u^r = [u_1, \ldots, u_p] \in \mathbb{R}^p$ be a padding vector, let $\bar{a} = [a^r, u^r]$ and let $\bar{A} = \text{symToe}(\bar{a})$. Then $\bar{A}$ contains $A$ as its principal $m \times m$ submatrix. Now let

$$\bar{b} = [a_0, \ldots, a_{m-1}, u_1, \ldots, u_p, u_{p-1}, \ldots, u_1, a_{m-1}, \ldots, a_1]$$

and

$$\bar{b}' = [a_0, \ldots, a_{m-1}, u_1, \ldots, u_p, u_p, u_{p-1}, \ldots, u_1, a_{m-1}, \ldots, a_1].$$

Then $\bar{B} = \text{circ}(\bar{b})$ and $\bar{B}' = \text{circ}(\bar{b}')$ are the MCE and MOCE of $\bar{A}$ respectively and have sizes $2(m + p - 1) \times 2(m + p - 1)$ and $(2(m + p) - 1) \times (2(m + p) - 1)$ respectively.

Note that whatever our choices of $u^r$, the resulting matrices $\bar{B}$ and $\bar{B}'$ will be circulant embeddings of the original matrix $A$. 
Minimal embedding of a non-symmetric Toeplitz matrix into a circulant matrix

Suppose that we wish to embed a non-symmetric Toeplitz matrix $A = \text{Toe}(\mathbf{a}^r, \mathbf{a}^c)$, for $\mathbf{a}^r, \mathbf{a}^c \in \mathbb{R}^m$, into a circulant matrix $B$, which will have the form

$$
B = \begin{bmatrix}
A & ? \\
? & ?
\end{bmatrix} = \\
\begin{bmatrix}
a_0 & a_1 & a_2 & \ldots & a_{m-1} \\
a_{-1} & a_0 & a_1 & \ldots & a_{m-2} \\
a_{-2} & a_{-1} & a_0 & \ldots & a_{m-3} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
a_{-m+1} & a_{-m+2} & a_{-m+3} & \ddots & a_0 \\
? & ? & ? & \ddots & ?
\end{bmatrix}.
$$

We can use a similar approach to the one we used for symmetric Toeplitz matrices; if we let $\mathbf{b}$ denote the first row of the circulant matrix $B$, we can let the last entry of $\mathbf{b}$ be the first entry of the second row of $B$ (i.e. $a_{-1}$), the penultimate entry of $\mathbf{b}$ be the first entry of the third row of $B$ (i.e. $a_{-2}$), and so on. However, if $a_{m-1} \neq a_{-m+1}$ we cannot stop when we set the $(m+1)$-th entry of $\mathbf{b}$ to the first entry of the $(m-1)$-th row of $B$ (i.e. $a_{-m+2}$); indeed, if we set

$$
\mathbf{b} = [a_0, a_1, \ldots, a_{m-2}, a_{m-1}, a_{-m+2}, \ldots, a_{-2}, a_{-1}],
$$

then

$$
B = \text{circ}(\mathbf{b}) = \\
\begin{bmatrix}
a_0 & a_1 & a_2 & \ldots & a_{m-1} & a_{m+2} & a_{m+3} & \ldots & a_{-1} \\
a_{-1} & a_0 & a_1 & \ldots & a_{m-2} & a_{m-1} & a_{m+2} & \ldots & a_{-2} \\
a_{-2} & a_{-1} & a_0 & \ldots & a_{m-3} & a_{m-2} & a_{m-1} & \ldots & a_{-3} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
a_{-m+1} & a_{-m+2} & a_{-m+3} & \ddots & a_0 & a_1 & a_2 & \ldots & a_{m-2} \\
\end{bmatrix}.
$$
The underlined elements are wrong - they should be \(a_{-m+1}\) in order to embed \(A\) correctly, but setting these elements to \(a_{-m+1}\) by setting 

\[
\mathbf{b} = [a_0, a_1, \ldots, a_{m-2}, a_{-m+1}, a_{m-1}, a_{m-2}, \ldots, a_2, a_1]
\]

will not work either, since the resulting matrix will embed \(A\) correctly but it will not be circulant. The solution is to add an extra element to \(\mathbf{b}\); let 

\[
\mathbf{b}' = [a_0, a_1, \ldots, a_{m-2}, a_{m-1}, a_{m-2}, \ldots, a_2, a_1, a_{-m+1}, a_{-m+2}, \ldots, a_{-2}, a_{-1}].
\]

Now \(A\) is embedded correctly inside \(B\). Hence to perform a circulant embedding of a non-symmetric Toeplitz matrix, where \(a_{m-1} \neq a_{-m+1}\), we must use the MOCE - the MCE does not work. Note that the MCE does work if \(a_{m-1} = a_{-m+1}\), since we do not have the conflict described above.

**Other embeddings of non-symmetric Toeplitz matrices into non-symmetric circulant matrices**

We can also use padding to embed a non-symmetric Toeplitz matrix into a non-symmetric circulant matrix. Let \(\mathbf{u}^r = [u_1, \ldots, u_p] \in \mathbb{R}^p\) and \(\mathbf{u}^c = [u_{-1}, \ldots, u_{-p}] \in \mathbb{R}^p\) be padding vectors, let \(\mathbf{\bar{a}}^r = [\mathbf{a}^r, \mathbf{u}^r]\) and \(\mathbf{\bar{a}}^c = [\mathbf{a}^c, \mathbf{u}^c]\) and let \(\bar{A} = \text{Toe}(\mathbf{\bar{a}}^r, \mathbf{\bar{a}}^c)\). Then we can use the MOCE to embed \(\bar{A}\) into a circulant matrix \(\bar{B}\) of size \(2(m + p - 1) \times (2(m + p) - 1)\). If \(u_{-p} = u_p\) we can also use the MCE to embed \(\bar{A}\) into a circulant matrix \(\bar{B}\) of size \(2(m + p - 1) \times 2(m + p - 1)\).

**Notation for the MCE and the MOCE**

We introduce more notation to help facilitate our discussion about circulant embeddings. Let the vectors \(\mathbf{a}^r\) and \(\mathbf{a}^c\) be given by

\[
\mathbf{a}^r = [a_0, a_1, \ldots, a_{m-1}] \in \mathbb{R}^m,
\]

\[
\mathbf{a}^c = [a_0, a_{-1}, \ldots, a_{-m+1}] \in \mathbb{R}^m.
\]

We shall assume that padding has already been applied (so \(\mathbf{a}^r\) and \(\mathbf{a}^c\) correspond to \(\mathbf{\bar{a}}^r\) and \(\mathbf{\bar{a}}^c\) from before, and \(m\) here corresponds to \(m + p\) from before), and introduce notations for the various circulant embeddings we have introduced as follows:
• The first row of the MCE of $A = \text{symToe}(a^r)$ is

$$\text{MCEvec}(a^r) = [a_0, \ldots, a_{m-2}, a_{m-1}, a_{m-2}, \ldots, a_1].$$

The resulting circulant matrix is denoted by

$$B = \text{MCE}(a^r) = \text{circ}(\text{MCEvec}(a^r));$$

• The first row of the MOCE of $A = \text{symToe}(a^r)$ is

$$\text{MOCEvec}(a^r) = [a_0, \ldots, a_{m-2}, a_{m-1}, a_{m-2}, \ldots, a_1].$$

The resulting circulant matrix is denoted by

$$B = \text{MOCE}(a^r) = \text{circ}(\text{MOCEvec}(a^r));$$

• The first row of the MCE of $A = \text{Toe}(a^r, a^c)$ is

$$\text{MCEvec}(a^r, a^c) = [a_0, \ldots, a_{m-2}, a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1].$$

Note that we only use this if $a_{m-1} = a_{-m+1}$. The resulting circulant matrix is denoted by

$$B = \text{MCE}(a^r, a^c) = \text{circ}(\text{MCEvec}(a^r, a^c));$$

• The first row of the MOCE of $A = \text{Toe}(a^r, a^c)$ is

$$\text{MOCEvec}(a^r, a^c) = [a_0, \ldots, a_{m-1}, a_{m-1}, a_{m+1}, a_{m+2}, \ldots, a_{-1}].$$

The resulting circulant matrix is denoted by

$$B = \text{MOCE}(a^r, a^c) = \text{circ}(\text{MOCEvec}(a^r, a^c)).$$

### 3.4 Block Toeplitz and Block Circulant Matrices

Similarly to Toeplitz and circulant matrices, block Toeplitz and block circulant matrices will appear when simulating two-dimensional stochastic processes on a regularly spaced grid.
Definition 3.9 An $n \times n$ block matrix $A$ is block Toeplitz matrix if its blocks are constant on every diagonal; that is, if $A$ has the form

$$A = \begin{bmatrix}
A_0 & A_1 & A_2 & \ldots & A_{n-1} \\
A_{-1} & A_0 & A_1 & \ldots & A_{n-2} \\
A_{-2} & A_{-1} & A_0 & \ldots & A_{n-3} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
A_{-n+1} & A_{-n+2} & A_{-n+3} & \ddots & A_0
\end{bmatrix}$$

for some $A_{-n+1}, \ldots, A_{n-1}$ all of size $m \times m$. Any block Toeplitz matrix can be constructed from its first block row and its first block column. If $A_{-j} = A_j$ for $j = 1, \ldots, n-1$, then $A$ is block symmetric; $A$ is then given by

$$A = \begin{bmatrix}
A_0 & A_1 & A_2 & \ldots & A_{n-1} \\
A_1 & A_0 & A_1 & \ldots & A_{n-2} \\
A_2 & A_1 & A_0 & \ldots & A_{n-3} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
A_{n-1} & A_{n-2} & A_{n-3} & \ddots & A_0
\end{bmatrix}$$

Any block symmetric Toeplitz matrix can be constructed from either its first block row or first block column, which are both the same.

We can also have general block matrices with Toeplitz blocks, and block Toeplitz matrices with Toeplitz blocks. For a block of matrices $A^r = [A_0, \ldots, A_{n-1}]$ and a block of matrices $A^c = [A_0, A_{-1}, \ldots, A_{-n+1}]$, where each $A_j$ is of size $m \times m$, we shall use the notation $\text{Toe}(A^r, A^c)$ to denote the block Toeplitz matrix in (3.13), and $\text{symToe}(A^r)$ to denote the symmetric block Toeplitz matrix in (3.14).

Definition 3.10 An $n' \times n'$ block matrix $B$ is block circulant if each of its block rows is the same as the previous block row, but periodically shifted one place to the right. Thus $A$ has the form

$$B = \begin{bmatrix}
B_0 & B_1 & B_2 & \ldots & B_{n'-1} \\
B_{n'-1} & B_0 & B_1 & \ldots & B_{n'-2} \\
B_{n'-2} & B_{n'-1} & B_0 & \ldots & B_{n'-3} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
B_1 & B_2 & B_3 & \ddots & B_0
\end{bmatrix}$$
where each $B_j$ is of size $m' \times m'$ for some $m'$. $A$ is also block Toeplitz according to Definition 3.7 with $B_{-j} = B_{n' - j}$, for $j = 1, \ldots, n' - 1$, and can be formed from its first block row.

Similarly, we can also have general block matrices with circulant blocks, and block circulant matrices with circulant blocks. For a block of matrices $B = [B_0, \ldots, B_{n' - 1}]$, where each $B_j$ is of size $m' \times m'$, we shall use the notation $\text{circ}(B)$ to denote the block circulant matrix in (3.15).

**Abbreviations**

It is convenient to abbreviate the names of some of these matrices as follows:

- A block (symmetric) Toeplitz matrix with (symmetric) Toeplitz blocks will be referred to as a $B(S)T(S)TB$ matrix;
- A block (symmetric) circulant matrix with (symmetric) circulant blocks will be referred to as a $B(S)C(S)CB$ matrix.
- In addition, we refer to any block matrix as an $(m, n)$ block matrix if it is an $n \times n$ block matrix, where each block is of size $m \times m$.

It is possible to embed any block Toeplitz matrix into a block circulant matrix, by using the procedure in §3.3 blockwise, and any matrix with Toeplitz blocks into a matrix with circulant blocks, by using the procedure in §3.3 to each individual block. Naturally any BTTB matrix can be embedded into a BCCB matrix by performing both procedures. We shall concentrate on studying the embeddings of BSTSTB matrices into BSCSCB matrices, and symmetric BTTB matrices into symmetric BCCB matrices, since these are the cases required in Chapter 4.

### 3.4.1 Additional notation for BSTSTB and symmetric BTTB matrices

It is possible to express any $(m, n)$ symmetric BTTB matrix using two vectors of length $m \times n$. Suppose that

$$A = \text{Toe}(A^r, A^c),$$

(3.16)
where
\[ A^r = [A_0, A_1, \ldots, A_{n-1}], \quad (3.17) \]
\[ A^c = [A_0, A_1^T, \ldots, A_{n-1}^T] \quad (3.18) \]
are the first block row and block column of \( A \) respectively, by symmetry of \( A \). We can now express \( A \) using the vectors
\[ a^{rr} = [a^r_0, a^r_1, \ldots, a^r_{n-1}], \quad (3.19) \]
\[ a^{rc} = [a^c_0, a^c_1, \ldots, a^c_{n-1}] \quad (3.20) \]
where \( a^r_j \) and \( a^c_j \) are the first row and column of \( A_j \) respectively for \( j = 0, \ldots, n-1 \), and \( a^r_0 = a^c_0 \) by symmetry of \( A \) and hence of \( A_0 \). The blocks \( A_j \) can be written as
\[ A_j = \text{Toe}(a^r_j, a^c_j), \quad A_j^T = \text{Toe}(a^c_j, a^r_j) \quad (3.21) \]
for \( j = 0, \ldots, m \). Now we denote the \( i \)-th entries of \( a^r_j \) and \( a^c_j \) by \( a_{ij} \) and \( a_{-i,j} \) respectively, and express \( A \) as
\[ A = \text{symBTTB}_{mn}(a^{rr}, a^{rc}). \quad (3.22) \]
If \( A \) is BSTSB, then \( a^r_j = a^c_j \) for all \( j = 0, \ldots, m \) and hence \( a^{rr} = a^{rc} \) and \( A^r = A^c \). Now
\[ A = \text{symToe}(A^r), \quad (3.23) \]
\[ A_j = \text{symToe}(a^r_j) \quad (3.24) \]
for all \( j = 0, \ldots, m \) and we can express \( A \) as
\[ A = \text{BSTSB}_{mn}(a^{rr}). \quad (3.25) \]

3.4.2 Notation for padding

We can pad each block of the BSTSB or symmetric BTTB matrix \( A \), which we shall call \textit{padding the blocks}, or we can pad the overall block matrix \( A \) with extra blocks, which we shall call \textit{block padding}. We can also pad the blocks and use block padding at the same time.
CHAPTER 3. LINEAR ALGEBRA

Padding the blocks

We can pad each block of $A$ using $p$ extra elements in each row and column, so that each block of the resulting padded matrix $\tilde{A}$ has blocks of size $m+p$. We can define two vectors $u^{rr}$ and $u^{rc}$, analogous to the vectors $a^{rr}$ and $a^{rc}$, in order to express padding of the blocks of $A$. Each vector is of length $p \times n$ and they are given by

$$u^{rr} = [u^r_0, u^r_1, \ldots, u^r_{n-1}],$$

(3.26)

$$u^{rc} = [u^r_0, u^c_1, \ldots, u^c_{n-1}],$$

(3.27)

where $u^r_j$ and $u^c_j$ are the padding for the first row and column of $A_j$ respectively (and, because of the symmetry of $A$, are the padding for the first column and row of $A_{n-j}$ respectively) for $j = 0, \ldots, n-1$. The $i$-th entries of $u^r_j$ and $u^c_j$ shall be denoted $u_{i,j}$ and $u_{-i,j}$ respectively for $i = 1, \ldots, p$. Now we denote $\tilde{a}^r_j = [a^r_j, u^r_j]$ and $\tilde{a}^c_j = [a^c_j, u^c_j]$. The padded blocks are denoted by

$$\tilde{A}_j = \text{Toe}(\tilde{a}^r_j, \tilde{a}^c_j), \quad \tilde{A}_j^T = \text{Toe}(\tilde{a}^c_j, \tilde{a}^r_j),$$

(3.28)

for $j = 0, \ldots, n-1$. If we now let $\tilde{a}^{rr} = [\tilde{a}^r_0, \ldots, \tilde{a}^r_{n-1}]$ and $\tilde{a}^{rc} = [\tilde{a}^c_0, \ldots, \tilde{a}^c_{n-1}]$, then we can write

$$\tilde{A} = \text{BSTSTB}^m+p,n(\tilde{a}^{rr})$$

(3.29)

in the BSTSTB case, or

$$\tilde{A} = \text{symBTTB}^m+p,n(\tilde{a}^{rr}, \tilde{a}^{rc})$$

(3.30)

in the symmetric BTTB case. We shall use the notation

$$\tilde{A}^r = [\tilde{A}_0, \tilde{A}_1, \ldots, \tilde{A}_{n-1}],$$

(3.31)

$$\tilde{A}^c = [\tilde{A}_0, \tilde{A}_1^T, \ldots, \tilde{A}_{n-1}]$$

(3.32)

when discussing block padding.

Block padding

Once we have decided whether or not to pad the blocks of $A$ to produce a new matrix $\tilde{A}$, we can also choose whether or not to pad the block matrix $\tilde{A}$ using $q$ extra blocks...
in each block row and block column so that the padded matrix $\bar{A}$ is an $n + q$ block matrix. Let

$$V^r = [V_1, \ldots, V_q], \quad V^c = [V_1^T, \ldots, V_q^T]$$

be the extra blocks in the first block row and block column of $A$ respectively, where each $V_j$ is of size $(m + p) \times (m + p)$ ($p = 0$ if we chose not to pad the blocks), and is Toeplitz if $A$ is BTTB, or symmetric Toeplitz if $A$ is BSTSTB. Now let $\bar{A}^r = [\bar{A}^r, V^r]$ and $\bar{A}^c = [\bar{A}^c, V^c]$. We can now write

$$\bar{A} = \text{Toe}(\bar{A}^r, \bar{A}^c).$$

We would still like to be able to express $\bar{A}$ in terms of two vectors (or one if $\bar{A}$ is BSTSTB). Hence we define two vectors $v^{rr}$ and $v^{rc}$ of length $(m + p) \times q$ to express the elements in the matrices $V_j$ as follows:

$$v^{rr} = [v_{r1}^r, v_{r2}^r, \ldots, v_q^r], \quad v^{rc} = [v_{c1}^r, v_{c2}^r, \ldots, v_q^r],$$

where $v_j^r$ and $v_j^c$ are the first row and column of $V_j$ respectively, for $j = 1, \ldots, q$. Hence

$$V_j = \text{Toe}(v_j^r, v_j^c)$$

for $j = 1, \ldots, q$, and we denote the $i$-th entry of $v_j^r$ and $v_j^c$ by $v_{ij}$ and $v_{-i,j}$ respectively for $i = 0, \ldots, m + p - 1$. Finally, let $\bar{a}^{rr} = [\bar{a}^{rr}, v^{rr}]$ and $\bar{a}^{rc} = [\bar{a}^{rc}, v^{rc}]$. We can now express the padded block matrix as

$$\bar{A} = \text{BSTSTB}^{m+p,q+n}(\bar{a}^{rr})$$

in the BSTSTB case, or

$$\bar{A} = \text{symBTTB}^{m+p,q+n}(\bar{a}^{rr}, \bar{a}^{rc})$$

in the symmetric BTTB case.
3.5 Embeddings of BSTSTB and symmetric BTTB matrices into BSCSCB and symmetric BCCB matrices

We can use the notation from §3.3 to express the embeddings of \((m, n)\) BSTSTB matrices into BSCSCB matrices, and \((m, n)\) symmetric BTTB matrices into symmetric BCCB matrices. We shall assume that any padding has already been applied, and shall use (3.16)-(3.25) from §3.4.1 extensively.

Embedding BSTSTB matrices into BSCSCB matrices

The first step when embedding BSTSTB matrices into BSCSCB matrices is to embed the block Toeplitz matrix \(A = \text{symToe}(A^r)\) into a block symmetric circulant matrix \(\tilde{B}\). This is achieved by taking either \(\tilde{B} = \text{MCE}(A^r)\) or \(\tilde{B} = \text{MOCE}(A^r)\). \(\tilde{B}\) is a \((m, n')\) block circulant matrix, where \(n' = 2(n - 1)\) if the MCE is used or \(n' = 2n - 1\) if the MOCE is used. Once \(\tilde{B}\) is formed we can embed each of its symmetric Toeplitz blocks \(A_j = \text{symToe}(a^r_j)\), for \(j = 0, \ldots, n - 1\), into symmetric circulant matrices \(B_j\), by taking either \(B_j = \text{MCE}(a^r_j)\) or \(B_j = \text{MOCE}(a^r_j)\). Note that each of the blocks \(B_j\) must be the same size and so we must use either the MCE or the MOCE to embed all the blocks \(A_j\). The size of the blocks is \(m' = 2(m - 1)\) if the MCE is used or \(m' = 2m - 1\) if the MOCE is used.

By embedding the block Toeplitz matrix, and then the Toeplitz blocks, we produce a \((m', n')\) BSCSCB matrix \(B\). It is possible to express \(B\) in terms of its first row \(b\), which we denote

\[
b = \text{BEvec}_{\alpha\beta}^{mn}(a^{rr}),
\]

where

\[
\alpha = \begin{cases} 
0, & \text{if the MCE is used to embed the blocks}, \\
1, & \text{if the MOCE is used to embed the blocks}, 
\end{cases}
\]

\[
\beta = \begin{cases} 
0, & \text{if the MCE is used to embed the block matrix}, \\
1, & \text{if the MOCE is used to embed the block matrix}, 
\end{cases}
\]
and $a^{rt}$ is given by (3.19). $B$ itself is denoted

$$B = BE^m_{a\beta}(a^{rt}).$$

(3.44)

Embedding symmetric BTTB matrices into symmetric BCCB matrices

We can embed symmetric BTTB matrices into symmetric BCCB matrices using exactly the same procedure used to embed BSTSTB matrices into BSCSCB matrices. The block Toeplitz matrix $A = \text{Toe}(A^r, A^c)$ is embedded into a block circulant matrix $\tilde{B}$ by taking either $\tilde{B} = \text{MCE}(A^r, A^c)$ or $\tilde{B} = \text{MOCE}(A^r, A^c)$ - however the MCE can only be used if $A_{n-1}$ is symmetric. Once $\tilde{B}$ is formed we again embed each of its Toeplitz blocks $A_j = \text{Toe}(a_{j}^r, a_{j}^c)$ into circulant matrices $B_j$ by taking either $B_j = \text{MCE}(a_{j}^r, a_{j}^c)$ or $B_j = \text{MOCE}(a_{j}^r, a_{j}^c)$. Using the MCE is only possible if $a_{m-1,j}$ and $a_{-(m-1),j}$, the final elements of $a_{j}^r$ and $a_{j}^c$ respectively, are equal for all $j = 0, \ldots, n - 1$; otherwise the MOCE must be used.

Again we can express the final symmetric BCCB matrix $B$ in terms of its first row $b$, which we denote

$$b = BE\text{vec}^m_{a\beta}(a^{rt}, a^{rc}),$$

(3.45)

where $\alpha$ and $\beta$ are given by (3.42) and (3.43), and $a^{rt}$ and $a^{rc}$ are given by (3.19) and (3.20) respectively. We denote $B$ itself by

$$B = BE^m_{a\beta}(a^{rt}, a^{rc}).$$

(3.46)

### 3.6 Matrix Factorisations

Matrices can often be factorised in a variety of different ways. For example, an $LU$ factorisation may be used to solve a linear equation $Ax = b$ efficiently, where $A$ is a square matrix [18, pp. 38–44], and any matrix can be factorised using the singular value decomposition (SVD) which is used, for example, for solving linear least square problems [18, pp. 109–117].

We are particularly interested in matrix factorisations of symmetric non-negative definite matrices, since covariance matrices of random vectors are always symmetric non-negative definite. We know, for example, that any symmetric positive definite matrix can be factorised using the Cholesky factorisation [18, pp. 77–78]. Circulant
and BCCB matrices have a particularly simple factorisation which makes use of the
Discrete Fourier Transform, a fact we use to our advantage in Chapter 4.

Decompositions of Circulant Matrices

It turns out that circulant matrices are *unitarily diagonalisable*, as proved in the fol-
lowing result. A matrix $A$ is unitarily diagonalisable if it has a decomposition of the
form $A = UDU^*$ for a unitary matrix $U$ and a diagonal matrix $D$, and this is equiv-
alent to the condition $A^*A = AA^*$ [68, pp. 569–570]. This makes circulant matrices
particularly easy to factorise; hence why they are used in the circulant embedding
method for simulating Gaussian random fields in Chapter 4.

**Proposition 3.11** Let $B = \text{circ}(\mathbf{b})$ be an $m \times m$ circulant matrix with first row $\mathbf{b} \in \mathbb{R}^m$. Then $B$ is unitarily diagonalisable and has the decomposition $B = W_m \Lambda W_m^*$, where:

- $W_m$ is the Fourier matrix with entries as defined in (3.3), and the columns of
  $W_m$ are eigenvectors of $B$;
- $\Lambda = \text{diag}(\lambda)$, where $\lambda = \sqrt{m} \hat{\mathbf{b}} = \sqrt{m} W_m \mathbf{b}$, the DFT of $\mathbf{b}$ multiplied by $\sqrt{m}$, and
  contains the eigenvalues of $B$.

**Proof** See [16, pp. 72–74].

**Corollary 3.12** If $B$ is symmetric then all the eigenvalues $\lambda_k$ are real-valued for all
$k = 0, \ldots, m - 1$.

**Proof** This follows since the eigenvalues of a symmetric (or indeed Hermitian) matrix
are always real [66, p. 34].

Decompositions of Block Circulant, Circulant Block and BCCB matrices

Block circulant, circulant block and BCCB matrices also have special decompositions,
involving the two-dimensional Fourier matrix.
**Proposition 3.13** If \( B = \text{circ}(B_0, \ldots, B_{n-1}) \) is an \((m,n)\) block circulant matrix, it has the decomposition \( B = W_{m,n}^* \Lambda W_{m,n} \), where \( \Lambda \) is an \((m,n)\) block diagonal matrix of the form \( \Lambda = \text{diag}(\Lambda_0, \Lambda_1, \ldots, \Lambda_{n-1}) \) for some \( m \times m \) matrices \( \Lambda_0, \ldots, \Lambda_{n-1} \), and \( W_{m,n} \) is given by (3.8).

**Proof** See [16, pp. 176–181].

**Proposition 3.14** If \( B \) is an \((m,n)\) block matrix with circulant blocks, it has the decomposition \( B = W_{m,n}^* \Lambda W_{m,n} \), where \( \Lambda \) is an \((m,n)\) block matrix with diagonal blocks \( \Lambda_k = \text{diag}(\lambda_{0,k}, \lambda_{1,k}, \ldots, \lambda_{m-1,k}) \) for \( k = 0, \ldots, n-1 \).

**Proof** See [16, pp. 181–184].

We would expect the two above results to be combined for a BCCB matrix; this is indeed the case.

**Proposition 3.15** An \((m,n)\) BCCB matrix \( B = \text{BCCB}^{mn}(b) \), where \( b = (b_0, \ldots, b_{n-1}) \) and each \( b_k \) has entries \( b_{jk} \) for \( j = 0, \ldots, m-1 \) and \( k = 0, \ldots, n-1 \), has the decomposition \( B = W_{m,n} \Lambda W_{m,n}^* \), where:

- \( W_{m,n} \) is the two-dimensional Fourier matrix with entries as defined in (3.9);
- \( \Lambda = \text{diag}(\lambda) \), where \( \lambda = (\lambda_0, \ldots, \lambda_{n-1}) \) and each \( \lambda_k \) has entries \( \lambda_{jk} \), for \( j = 0, \ldots, m-1 \) and \( k = 0, \ldots, n-1 \). The vector \( \lambda = \sqrt{mn} \hat{b} = \sqrt{mn} W_{m,n} b \), the two-dimensional DFT of \( b \) multiplied by \( \sqrt{mn} \), and contains the eigenvalues of \( B \).

**Proof** See [16, pp. 184–186].

**Corollary 3.16** If \( B \) is symmetric BCCB or BSCSCB, its eigenvalues are all real.

**Proof** Again, from [66, p. 34], the eigenvalues of any symmetric matrix are real.
Chapter 4

The Circulant Embedding Method

Suppose that we wish to create realisations of a random field $Z(x)$, with a given covariance function, over a domain $D \in \mathbb{R}^d$. In general we can sample $Z(x)$ at an arbitrary set of points in $D$, and the problem is now reduced to sampling from a multivariate normal distribution with some covariance matrix $A$, which will be symmetric non-negative definite and must be factorised. To do this in general we must use the Cholesky factorisation, which is expensive; however we can reduce this cost by using the circulant embedding method. This method can be used for $d$-dimensional fields for any $d \geq 1$ (see for example [12]); however we shall focus on simulating one- and two-dimensional fields.

4.1 Introduction

We list some conventions which shall be used throughout this chapter:

- $Z(x)$ shall always denote the random field we are interested in, with mean function zero and covariance function $C(x, y)$. In the case of a one-dimensional random field, the random field shall be denoted $Z(x)$ and the covariance function shall be denoted $C(x, y)$.

- The mean function is assumed to be zero since for a random field $\hat{Z}(x)$ with mean function $\mu(x)$ and covariance function $C(x, y)$, we can simulate realisations of $Z(x)$ and then add $\mu(x)$ to each of the realisations to obtain realisations of $\hat{Z}(x)$. 

64
• Indexing for vectors and matrices shall start from 0 rather than 1 in order to simplify notation.

Direct simulation of Gaussian random fields

One method of simulating Gaussian random fields $Z(\mathbf{x})$ is to sample $Z(\mathbf{x})$ at an arbitrary set of $N$ points $\mathbf{x}_i \in D$, where $i = 0, \ldots, N - 1$. This produces a random vector $\mathbf{X} \in \mathbb{R}^N$, with entries $X_i = Z(\mathbf{x}_i)$ for $i = 0, \ldots, N - 1$. $\mathbf{X}$ will then follow the $N(0_N, A)$ distribution, where the covariance matrix $A$ is symmetric and has entries $a_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$.

As we have seen in §2.1.1, if we can factorise $A$ as either $A = R^T R$ for a matrix $R \in \mathbb{R}^{N \times N}$, or $A = R^* R$ for a matrix $R \in \mathbb{C}^{N \times N}$, we can then generate realisations of the random vector $\mathbf{X}$ easily by sampling from the $N(0, 1)$ distribution and performing a matrix-vector multiplication. We hence have values for $Z(\mathbf{x})$ at the points $\mathbf{x}_i$ from each realisation of $\mathbf{X}$.

Of the tasks required for this method of simulating Gaussian random fields, the factorisation of the matrix $A$ is the most computationally expensive. For example, a Cholesky factorization of $A$ has a computational cost of $O(N^3)$ flops.

The circulant embedding method in a nutshell

The main purpose behind the circulant embedding method is to reduce the cost of the matrix factorisation. We saw in Proposition 3.11 that a circulant matrix $B = \text{circ}(\mathbf{b}) \in \mathbb{R}^{N \times N}$, where $\mathbf{b} \in \mathbb{R}^N$, can be factorised as $B = W_N \Lambda W_N^*$ where $W_N$ is the Fourier matrix of size $N \times N$ and $\Lambda = \sqrt{N} \hat{\mathbf{b}}$, the DFT of $\mathbf{b}$ multiplied by $\sqrt{N}$. Hence $B = R^* R$, where $R = \Lambda^{1/2} W_N^*$ and all that is needed to decompose $B$ is a DFT of its first row. We can use the Fast Fourier Transform (FFT) algorithm to perform the DFT - Brigham [5, pp. 134–135] shows that if $N$ is a power of 2, the FFT has a computational cost of $O(N \log_2 N)$ flops, although algorithms have been developed which have a cost of $O(N \log N)$ flops for any $N$ [24].

The problem is that, when sampling a random field $Z(\mathbf{x})$ at a set of $N$ points $\mathbf{x}_i$ for $i = 0, \ldots, N-1$, it is very unlikely that the covariance matrix $A$ of the resulting random
vector $\mathbf{X}$ with entries $X_i = Z(x_i)$ will be a circulant matrix. However, if the one-dimensional random field $Z(x)$ is stationary and the $N$ sampling points form a regularly spaced grid, the covariance matrix $A$ of $\mathbf{X}$ will be symmetric and Toeplitz (symmetric since it is a covariance matrix). We can then embed $A$ in a larger, symmetric circulant matrix $B$ of size $M \geq 2(N - 1)$. $B$ will be the covariance matrix for a Gaussian random vector $\mathbf{Y} \sim N(0_M, B)$, which can be simulated, and the first $N$ elements of $\mathbf{Y}$ (i.e. the vector $\mathbf{Y}_{[0,\ldots,N-1]}$) will produce a realisation of $\mathbf{X} \sim N(0_N, A)$, since $A = B_{[0,\ldots,N-1],[0,\ldots,N-1]}$ is the covariance matrix of $\mathbf{X}$ by Proposition 2.11.

The two-dimensional case is similar, except that $A$ will be an $(N_1, N_2)$ BTTTB or symmetric BTTTB matrix, where $N_1$ and $N_2$ are the number of gridpoints in the $x$ and $y$ coordinates respectively, and $B$ will be an $(M_1, M_2)$ BSCSCB or symmetric BCCB matrix for some $M_1 \geq 2(N_1 - 1)$ and $M_2 \geq 2(N_2 - 1)$. Extracting the desired realisation of $\mathbf{X} \sim N(0_{N_1 \times N_2}, A)$ from the realisation of $\mathbf{Y} \sim N(0_{M_1 \times M_2}, B)$ is also more involved; we must now take $\mathbf{X} = \mathbf{Y}_i$, where

$$i = [0, \ldots, N_1 - 1, M_1, \ldots, M_1 + N_1 - 1, \ldots, (N_2 - 1)M_1, \ldots, (N_2 - 1)M_1 + N_1 - 1].$$

(4.1)

The covariance matrix of $\mathbf{X}$ is thus $B = A_{i,i}$ by Proposition 2.11, as required.

Note that in both the one- and two-dimensional cases, the fact that the Gaussian random field is stationary means that any realisation of the random field $Z(x)$ over a domain $D$ is equivalent to a realisation over a different domain $D'$ which is a translation of the original domain $D$. For example, any realisation of a one-dimensional random field $Z(x)$ over the interval $[x_{\min}, x_{\max}]$ is equivalent to a realisation of $Z(x)$ over the interval $[0, x_{\max} - x_{\min}]$.

In order for the circulant embedding method to work, we require $B$ to be non-negative definite. If this is not the case, we can try increasing its size, which may produce a non-negative definite matrix; or we can use an approximation procedure or simply simulate the random field directly.
4.2 The circulant embedding method in one dimension

Let $Z(x)$ be a stationary Gaussian one-dimensional random field with mean function zero and covariance function $C(x, y)$. Since $Z(x)$ is stationary, we can rewrite the covariance function as a variogram $\gamma(x - y) = C(x, y) = \gamma(y - x)$. To simulate $Z(x)$ over a domain $D = [x_{\text{min}}, x_{\text{max}}]$ using the circulant embedding method, we choose the $N$ sampling points $x_i = x_{\text{min}} + \Delta x + i \Delta$ for $i = 0, \ldots, N - 1$, where $\Delta = \frac{x_{\text{max}} - x_{\text{min}}}{N}$ is the spacing. Now let $X$ be the random vector with entries $X_i = Z(x_i)$. Then $X \sim N(0, A)$, where $A \in \mathbb{R}^{N \times N}$ has entries $(A)_{ij} = C(x_i, x_j) = \gamma(x_i - x_j) = \gamma((i - j)\Delta) = \gamma((j - i)\Delta) = \gamma((j - i)\Delta) = \gamma((x_j - x_i)) = C(x_j, x_i) = (A)_{ji}$.

Hence $A$ is symmetric Toeplitz and can be written as $A = \text{symToe}(a)$, where $a$ is the first row of $A$ and has entries $a_i = \gamma(i \Delta)$.

We now embed $A$ into a larger circulant matrix $B$ of size $M$, as described in §3.3. Since $A$ is symmetric, we can use either the MCE or MOCE to do this, and we can also choose to use padding. Both Dietrich and Newsam [20] and Wood and Chan [12] suggest using the MCE; however, Dietrich and Newsam suggest using no padding, so $M = 2(N - 1)$, whereas Wood and Chan suggest using padding such that $M = 2^G$ for the smallest integer $G$ such that $M \geq 2(N - 1)$; that is, $G = \lceil \log_2 2(N - 1) \rceil = 1 + \lceil \log_2 (N - 1) \rceil$. This is because the DFT, which is used later, is most efficient when $M$ is a power of 2.

We choose to use the second approach, and denote the padding vector by $u = [u_1, \ldots, u_P]$, where $P$ is the size of the padding vector. Since we are using the MCE, $M = 2^G = 2(N + P - 1)$ and hence $P = \frac{M}{2} - N + 1 = 2^{G-1} - N + 1$. Any choice of $u$ is guaranteed to produce a circulant embedding matrix, but we choose to use one of the following options for the elements of $u$:

1. $u_i = 0$ for all $i = 1, \ldots, P$, which we call padding with zeros; or

2. $u_i = \gamma((i + N - 1)\Delta)$ for all $i = 1, \ldots, P$, which we call extending the grid.

Now we set $\bar{a} = [a, u]$ (or $\bar{a} = a$ if $P = 0$) and denote the new padded matrix by $\bar{A} = \text{symToe}(\bar{a})$. The second choice of padding is called extending the grid because $\bar{A}$
is the covariance matrix we would have obtained if we had sampled the random field \( Z(x) \) over an interval of size \((N + P)\Delta \) with spacing \( \Delta \), instead of the interval of size \( x_{\text{max}} - x_{\text{min}} = N\Delta \) which we are actually sampling over. This choice of padding is more likely to produce a non-negative definite embedding matrix in several cases - see §4.6.

We now embed \( A \) into a circulant matrix \( B \) of size \( M \times M \), with first row \( \mathbf{b}; \mathbf{b} \) and \( \mathbf{B} \) are given by \( \mathbf{b} = \text{MCEvec}(\bar{\mathbf{a}}) \), \( \mathbf{B} = \text{MCE}(\bar{\mathbf{a}}) \). Note that we do not need to store \( \mathbf{B} \) - only \( \mathbf{b} \) needs to be stored, and neither \( A \) nor \( \bar{A} \) need to be stored since \( \mathbf{b} \) can be formed directly. \( \mathbf{B} \) is a symmetric circulant embedding matrix of \( A \), as required.

As proved in Proposition 3.11, \( \mathbf{B} \) is diagonalisable and has decomposition \( \mathbf{B} = \mathbf{W}\mathbf{M}\Lambda\mathbf{W}^* \), where \( \mathbf{W} \) is the Fourier matrix of size \( M \) and \( \Lambda = \text{diag}(\lambda) \). \( \mathbf{b} \) contains the eigenvalues \( \lambda_k \) of \( \mathbf{B} \), where \( \hat{\mathbf{b}} \) is the DFT of \( \mathbf{b} \). Since \( \mathbf{B} \) is symmetric, all of its eigenvalues are real-valued by Corollary 3.12.

Now, if \( \mathbf{B} \) is non-negative definite then it is a valid covariance matrix for a random vector \( \mathbf{Y} \sim \mathcal{N}(0, \mathbf{B}) \). We can check whether \( \mathbf{B} \) is non-negative definite easily, since we know its eigenvalues; if all the eigenvalues \( \lambda_k \geq 0 \) then \( \mathbf{B} \) is non-negative definite. If this is the case, we can now form the matrix \( \Lambda_* = \text{diag}(\lambda_*^\frac{1}{2}) \), where \( \lambda_*^\frac{1}{2} \) has elements \( \lambda_k^\frac{1}{2} \), the square roots of the eigenvalues, and now \( \mathbf{B} = \mathbf{R}^*\mathbf{R} \), where \( \mathbf{R} = \Lambda_*^\frac{1}{2}\mathbf{W}_M^* \) is a complex matrix. We can now sample \( \mathbf{Y} \) using the procedure described in §2.1.1 as follows:

- Simulate a random vector \( \mathbf{Z} \sim \mathcal{CN}(0_M, 2I_M) \) - we can do this by setting \( \mathbf{Z} = \mathbf{Z}_1 + i\mathbf{Z}_2 \), where \( \mathbf{Z}_1, \mathbf{Z}_2 \sim \mathcal{N}(0_M, I_M) \);
- Let \( \hat{\mathbf{Y}} = \mathbf{R}^*\mathbf{Z} = \mathbf{W}_M\Lambda_*^\frac{1}{2}\mathbf{Z} \). Note that \( \hat{\mathbf{Y}} = \hat{\Lambda}_*^\frac{1}{2}\mathbf{Z} \), the DFT of the vector \( \Lambda_*^\frac{1}{2}\mathbf{Z} \) and so this calculation can be executed efficiently by forming \( \Lambda_*^\frac{1}{2}\mathbf{Z} \) and then performing a DFT;
- Since \( \hat{\mathbf{Y}} \sim \mathcal{CN}(0_M, 2\mathbf{B}) \), its real part \( \mathbf{Y}_1 = \Re\hat{\mathbf{Y}} \) and imaginary part \( \mathbf{Y}_2 = \Im\hat{\mathbf{Y}} \) both follow the \( \mathcal{N}(0_M, \mathbf{B}) \) distribution, and are both realisations of \( \mathbf{Y} \).

From here we simply let \( \mathbf{X}_1 \) and \( \mathbf{X}_2 \) be the vectors formed by taking the first \( N \) elements of \( \mathbf{Y}_1 \) and \( \mathbf{Y}_2 \), and by Proposition 2.11 \( \mathbf{X}_1, \mathbf{X}_2 \sim \mathcal{N}(0_N, \mathbf{A}) \) as required (we
can, in fact, take any \( N \) consecutive elements of \( Y_1 \) and \( Y_2 \) but for simplicity we shall take the first \( N \).

### 4.3 The circulant embedding method in two dimensions

Now suppose \( Z(x) \) is a stationary two-dimensional random field to be sampled over a domain \( D = [x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}] \), with covariance function \( C(x, y) \) which can be written as a variogram \( \gamma(x - y) = C(x, y) = \gamma(y - x) \). We now simulate \( Z(x) \) using an equally spaced grid of \( N = N_1 N_2 \) sampling points, where \( N_1 \) and \( N_2 \) are the number of points in the \( x \) and \( y \) directions respectively (note that \( N_1 \) and \( N_2 \) are allowed to be different). The coordinates \((x_i, y_j)\) of the \( N \) grid points are 

\[
\begin{align*}
  x_i &= x_{\text{min}} + \frac{\Delta_1}{2} + i \Delta_1 \\
  y_j &= y_{\text{min}} + \frac{\Delta_2}{2} + j \Delta_2 
\end{align*}
\]

for \( i = 0, \ldots, N_1 - 1 \) and \( j = 0, \ldots, N_2 - 1 \), where \( \Delta_1 = \frac{x_{\text{max}} - x_{\text{min}}}{N_1} \) and \( \Delta_2 = \frac{y_{\text{max}} - y_{\text{min}}}{N_2} \) are the grid spacing in the \( x \) and \( y \) directions respectively. Our random vector \( \mathbf{X} \) now has entries 

\[
X_{i+jN_1} = Z \left( \begin{bmatrix} x_i \\ y_j \end{bmatrix} \right)
\]

Now \( \mathbf{X} \sim N(0_N, A) \), where \( A \) is a symmetric matrix with entries 

\[
\begin{align*}
A_{i+jN_1, k+\ell N_1} &= C \left( \begin{bmatrix} x_i \\ y_j \end{bmatrix}, \begin{bmatrix} x_k \\ y_\ell \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} x_k - x_i \\ y_\ell - y_j \end{bmatrix} \right) \\
&= \gamma \left( \begin{bmatrix} (k-i) \Delta_1 \\ (\ell-j) \Delta_2 \end{bmatrix} \right) = \gamma \left( \begin{bmatrix} (i-k) \Delta_1 \\ (j-\ell) \Delta_2 \end{bmatrix} \right) \\
&= \gamma \left( \begin{bmatrix} x_i - x_k \\ y_j - y_\ell \end{bmatrix} \right) = C \left( \begin{bmatrix} x_k \\ y_\ell \end{bmatrix}, \begin{bmatrix} x_i \\ y_j \end{bmatrix} \right) = A_{k+\ell N_1, i+j N_1},
\end{align*}
\]

or for \( i, k = 0, \ldots, N_1 - 1, \ j, \ell = 0, \ldots, N_2 - 1 \). Note that \( A \) is an \((N_1, N_2)\) symmetric BTTB matrix; whether \( A \) is BSTSTB or not depends on whether \( \gamma \) is even or uneven. If \( \gamma \) is even then 

\[
A_{i+j N_1, k+\ell N_1} = A_{k+j N_1, i+\ell N_1} = A_{i+\ell N_1, k+j N_1} = A_{k+\ell N_1, i+j N_1}, \quad (4.2)
\]

and hence \( A \) is BSTSTB. If \( \gamma \) is uneven then \( (4.2) \) does not hold in general and so \( A \) is only symmetric BTTB.
We now use the notation in §3.4.1 to express the covariance matrix $A$. If $\gamma$ is even then

$$A = \text{BSTSTB}^{N_1,N_2}(a^{rr}),$$

where $a^{rr} = [a^r_0, \ldots, a^r_{N_2-1}]$ (as in (3.19) with $n = N_2$). The $i$-th element of $a^r_j$ (so the $(i + jN_1)$-th element of $a^{rr}$) is given by

$$a_{ij} = \gamma \left( \begin{bmatrix} i\Delta_1 \\ j\Delta_2 \end{bmatrix} \right),$$

for $i = 0, \ldots, N_1 - 1$ and $j = 0, \ldots, N_2 - 1$. If $\gamma$ is uneven then

$$A = \text{symBTTB}^{N_1,N_2}(a^{rr},a^{rc}),$$

where $a^{rc} = [a^c_0, \ldots, a^c_{N_2-1}]$ (as in (3.20) with $n$ replaced by $N_2$ again). The $i$-th element of $a^c_j$ is given by

$$a_{-i,j} = \gamma \left( \begin{bmatrix} -i\Delta_1 \\ j\Delta_2 \end{bmatrix} \right),$$

for $i = 0, \ldots, N_1 - 1$ and $j = 0, 1, \ldots, (N_2 - 1)$.

We now embed $A$ into a BCCB matrix, which will be a BSCSCB matrix if $\gamma$ is even ($A$ is BSTSTB), or a symmetric BCCB matrix if $\gamma$ is uneven ($A$ is only symmetric BTTB). Again we have a choice whether to use padding to improve the efficiency of the DFT, or no padding to keep $B$ smaller, and for the padding itself we can either pad with zeros or extend the grid.

**Padding $A$**

Suppose that we choose to use padding to extend $A$ into a larger $(N_1 + P, N_2 + Q)$ BSTSTB or symmetric BTTB matrix $\bar{A}$. If $A$ is symmetric BTTB then we can express the padding using two vectors $u^{rr}$ and $u^{rc}$, each of length $PN_2$, to express the padding for each block, and two vectors $v^{rr}$ and $v^{rc}$ each of length $(N_1 + P)Q$, to express the padding for the extra blocks. If $A$ is BSTSTB then only the vectors $u^{rr}$ and $v^{rr}$ are needed.

We define $u^{rr}$, $u^{rc}$, $v^{rr}$ and $v^{rc}$ as in (3.26), (3.27), (3.36) and (3.37) respectively (with $p = P$ and $q = Q$). The $i$-th elements of $u^r_j$ and $u^r_{-i,j}$ are denoted by $u_{ij}$ and $u_{-i,j}$ respectively and are given by

$$u_{ij} = u_{-i,j} = 0.$$
CHAPTER 4. THE CIRCULANT EMBEDDING METHOD

if padding with zeros, or

\[ u_{ij} = \gamma \left( \begin{bmatrix} (i + N_1 - 1) \Delta_1 \\ j \Delta_2 \end{bmatrix} \right), \quad u_{-i,j} = \gamma \left( \begin{bmatrix} -(i + N_1 - 1) \Delta_1 \\ j \Delta_2 \end{bmatrix} \right) \]

if extending the grid, for \( i = 1, \ldots, P \) and \( j = 0, 1, \ldots, (N_2 - 1) \). Similarly the i-th elements of \( v_j^r \) and \( v_j^c \) are denoted by \( v_{ij} \) and \( v_{-i,j} \) respectively and given by

\[ v_{ij} = v_{-i,j} = 0 \]

if padding with zeros, or

\[ v_{ij} = \gamma \left( \begin{bmatrix} i \Delta_1 \\ (j + N_2 - 1) \Delta_2 \end{bmatrix} \right), \quad v_{-i,j} = \gamma \left( \begin{bmatrix} -i \Delta_1 \\ (j + N_2 - 1) \Delta_2 \end{bmatrix} \right) \]

if extending the grid, for \( i = 0, \ldots, N_1 + P - 1 \) and \( j = 1, \ldots, Q \).

Now, as in §3.4.2 let \( \tilde{a}_j^r = [a_j^r, u_j^r] \) and \( \tilde{a}_j^c = [a_j^c, u_j^c] \) for \( j = 0, \ldots, N_2 - 1 \), and let \( \tilde{a}^{rr} = [\tilde{a}_0^r, \ldots, \tilde{a}_{n-1}^r] \) and \( \tilde{a}^{rc} = [\tilde{a}_0^c, \ldots, \tilde{a}_{n-1}^c] \), to include the padded blocks. Finally let \( \tilde{a}^{rr} = [\tilde{a}^{rr}, v^{rr}] \) and \( \tilde{a}^{rc} = [\tilde{a}^{rc}, v^{rc}] \), to include the block padding. We now denote the padded matrix by

\[ \tilde{A} = BSTSB^{N_1+P,N_2+Q}(\tilde{a}^{rr}) \]

in the BSTSB case, or

\[ \tilde{A} = symBTTB^{N_1+P,N_2+Q}(\tilde{a}^{rr}, \tilde{a}^{rc}) \]

in the symmetric BTTB case.

The embedding procedure if \( \gamma \) is even and \( A \) is BSTSB

If \( A \) is BSTSB then we can use the MCE to embed both the block matrix \( A \) and its blocks, such that \( A \) is embedded into an \((M_1, M_2)\) BSCSCB matrix \( B \) with first row \( b \), whether we use padding or not. If we choose not to use padding then \( M_1 = 2(N_1 - 1) \) and \( M_2 = 2(N_2 - 1) \). If padding is used, then \( M_1 = 2(N_1 + P - 1) \) and \( M_2 = 2(N_2 + Q - 1) \). Now \( b \) and \( B \) are given by

\[ b = BEvec_{00}^{N_1+P,N_2+Q}(\tilde{a}^{rr}), \]
\[ B = BE_{00}^{N_1+P,N_2+Q}(\tilde{a}^{rr}) = BCCB^{M_1,M_2}(b), \]
where \( \bar{a}^{rr} = a^{rr} \) if padding is not used. Note that again only the first row \( b \) needs to be stored and, since \( b \) can be formed directly, neither \( A \) nor \( \bar{A} \) needs to be stored.

Wood and Chan \[12\] suggest using padding such that \( M_1 \) and \( M_2 \) are both powers of 2, as in the one-dimensional case. In this case \( M_1 = 2^{G_1} \) and \( M_2 = 2^{G_2} \), where \( G_1 = 1 + \lceil \log_2(N_1 - 1) \rceil \), \( G_2 = 1 + \lceil \log_2(N_2 - 1) \rceil \), \( P = 2^{G_1 - 1} - N_1 + 1 \) and \( Q = 2^{G_2 - 1} - N_2 + 1 \).

**The embedding procedure if \( \gamma \) is uneven and \( A \) is symmetric BTTB**

If \( A \) is only symmetric BTTB, then in general we have to use the MOCE to embed \( A \) and its blocks into an \((M_1, M_2)\) symmetric BCCB matrix \( B \). It is possible to use the MCE to embed either the blocks and/or the block matrix if, for example, we pad the blocks and/or the block matrix respectively with zeros; however we shall assume that we use the MOCE for *both* embedding the blocks and the block embedding, for simplicity. Hence \( B \) and its first row \( b \) are

\[
\begin{align*}
b &= \text{BEvec}_{11}^{N_1 + P, N_2 + Q}(\bar{a}^{rr}, \bar{a}^{rc}), \\
B &= \text{BE}_{11}^{N_1 + P, N_2 + Q}(\bar{a}^{rr}, \bar{a}^{rc}) = \text{BCCB}_{M_1, M_2}(b),
\end{align*}
\]

where \( \bar{a}^{rr} = a^{rr} \) and \( \bar{a}^{rc} = a^{rc} \) if padding is not used. Again, only the first row \( b \) needs to be stored.

As before, Wood and Chan \[12\] advocate using padding in order to increase efficiency of the DFT. Since we are using the MOCE, \( M_1 \) and \( M_2 \) have to be odd and so we cannot have them being powers of 2, so we now use powers of 3 instead. So \( M_1 = 3^{G_1} \) and \( M_2 = 3^{G_2} \), where \( G_1 = \lceil \log_3(2N_1 - 1) \rceil \) and \( G_2 = \lceil \log_3(2N_2 - 1) \rceil \), and hence \( P = \frac{3^{G_1 + 1}}{2} - N_1 \) and \( Q = \frac{3^{G_2 + 1}}{2} - N_2 \).

**Obtaining realisations of the random field**

Whether \( \gamma \) is even or uneven, and whether padding is used or not, we will end up with an \((M_1, M_2)\) symmetric BCCB or BSCSCB matrix \( B \) which will successfully embed the \((N_1, N_2)\) symmetric BTTB or BSTSB matrix \( A \). We must now obtain samples of the random vector \( X \sim N(0_N, A) \), where \( N = N_1N_2 \). By Proposition \[3.15\] \( B \) is diagonalisable with decomposition \( B = W_{M_1, M_2} \Lambda W_{M_1, M_2}^* \), where \( W_{M_1, M_2} \) is the two-dimensional Fourier matrix and, for \( M = M_1M_2 \), \( \Lambda = \text{diag}(\lambda) \) where \( \lambda = \)
\[ \sqrt{MW_{M_1, M_2}}b = \sqrt{M} \hat{b}, \] the two-dimensional DFT of \( b \) multiplied by \( \sqrt{M} \). The \( j + M_1k \)-th entry of \( \lambda \) is denoted \( \lambda_{j,k} \).

As in the one-dimensional case, if \( B \) is non-negative definite (if and only if all the eigenvalues \( \lambda_{j,k} \geq 0 \)) it is a valid covariance matrix for a random vector \( Y \sim N(0_M, B) \), so we can form the matrix \( \Lambda^{\frac{1}{2}} = \text{diag}(\lambda^{\frac{1}{2}}) \), where \( \lambda^{\frac{1}{2}} \) has entries \( \lambda_{j,k}^{\frac{1}{2}} \). As before, \( B = R^*R \) where this time \( R = \Lambda^{\frac{1}{2}}W_{M_1, M_2} \). We sample \( Y \) in exactly the same way as for the one-dimensional case:

- Simulate \( Z \sim CN(0_M, 2I_M) \) by setting \( Z = Z_1 + iZ_2 \), where \( Z_1, Z_2 \sim N(0, I_M) \);
- Let \( \hat{Y} = R^*Z = W_{M_1, M_2}\Lambda^{\frac{1}{2}}Z = \Lambda^{\frac{1}{2}}\hat{Z} \);
- \( \hat{Y} \sim CN(0_M, 2B) \) so its real part \( Y_1 = \Re \hat{Y} \) and imaginary part \( Y_2 = \Im \hat{Y} \) both follow the \( N(0_M, B) \) distribution and are therefore samples of \( Y \) as required.

As before, we must now extract the samples of \( X \) from the samples of \( Y \). This time we must take \( X_1 = (Y_1)_i \) and \( X_2 = (Y_2)_i \), where \( i \) is given in (4.1). The indexing vector \( i \) divides \( Y_1 \) and \( Y_2 \) into \( M_2 \) blocks of size \( M_1 \), and extracts the first \( N_1 \) elements from each of the first \( N_2 \) blocks of \( Y_1 \) and \( Y_2 \). We notice that \( B_{i,i} = A \) from the embedding procedure and hence, by Proposition 2.11, \( X_1 \) and \( X_2 \) both follow the \( N(0_N, A) \) distribution as required.

### 4.4 Example: Simulating Fractional Brownian Motion using the circulant embedding method

We can use the circulant embedding method to simulate Fractional Brownian Motion \( B^H(t) \) over the interval \([0, T]\). Fractional Brownian Motion is a Gaussian stochastic process with mean function \( \mu(t) = 0 \) and covariance function

\[ C(s, t) = \mathbb{E}[B^H(s)B^H(t)] = \frac{1}{2}[t^{2H} + s^{2H} - |t - s|^{2H}]. \] (4.5)

Note that \( B^H(t) \) is not a stationary process, but if we choose to sample \( B^H(t) \) on the equally spaced grid \( t_i = \frac{iT}{N}, \ i = 0, \ldots, N \), we can sample the increments as if they form a stationary process. Let \( X(i) = B^H(t_{i+1}) - B^H(t_i) \), so that \( B^H(0) = 0 \) and \( B^H(t_j) = \sum_{i=0}^{j-1} X(i) \). We show that the increment process \( X(i) \) is stationary.
Proposition 4.1 The increment process $X(i)$ has mean zero and covariance function

$$C(i, j) = \frac{1}{2N^{2H}} [\lvert j - i + 1 \rvert^{2H} + \lvert j - i - 1 \rvert^{2H} - 2 \lvert j - i \rvert^{2H}],$$

and so $X(i)$ is a stationary Gaussian process. We can write $C(i, j) = \gamma(j - i) = \gamma(i - j)$.

**Proof** The fact that $X(i)$ has mean zero is obvious from the definition since $B^H(t)$ has mean zero. For the covariance, note that

$$C(i, j) = \mathbb{E} \left[ \left( B^H \left( \frac{j + 1}{N} \right) - B^H \left( \frac{j}{N} \right) \right) \left( B^H \left( \frac{i + 1}{N} \right) - B^H \left( \frac{i}{N} \right) \right) \right].$$

Since expectation is a linear operator, we can split this expression into four parts $A_k$, $k = 1, 2, 3, 4$, writing $C(i, j) = A_1 + A_2 + A_3 + A_4$, and using (4.5), we obtain

$$A_1 = \mathbb{E} \left[ B^H \left( \frac{j + 1}{N} \right) B^H \left( \frac{i + 1}{N} \right) \right] = \frac{1}{2} \left[ \left( \frac{j + 1}{N} \right)^{2H} + \left( \frac{i + 1}{N} \right)^{2H} - \left( \frac{j - i}{N} \right)^{2H} \right],$$

$$A_2 = -\mathbb{E} \left[ B^H \left( \frac{j + 1}{N} \right) B^H \left( \frac{i}{N} \right) \right] = -\frac{1}{2} \left[ \left( \frac{j + 1}{N} \right)^{2H} + \left( \frac{i}{N} \right)^{2H} - \left( \frac{j - i + 1}{N} \right)^{2H} \right],$$

$$A_3 = -\mathbb{E} \left[ B^H \left( \frac{j}{N} \right) B^H \left( \frac{i + 1}{N} \right) \right] = -\frac{1}{2} \left[ \left( \frac{j}{N} \right)^{2H} + \left( \frac{i + 1}{N} \right)^{2H} - \left( \frac{j - i - 1}{N} \right)^{2H} \right],$$

$$A_4 = \mathbb{E} \left[ B^H \left( \frac{j}{N} \right) B^H \left( \frac{i}{N} \right) \right] = \frac{1}{2} \left[ \left( \frac{j}{N} \right)^{2H} + \left( \frac{i}{N} \right)^{2H} - \left( \frac{j - i}{N} \right)^{2H} \right].$$

Hence

$$C(i, j) = \frac{1}{2N^{2H}} [\lvert j - i + 1 \rvert^{2H} + \lvert j - i - 1 \rvert^{2H} - 2 \lvert j - i \rvert^{2H}] = \gamma(j - i),$$

a function of one variable as claimed. \qed

For the purposes of coding, it is convenient to write the covariance function $\gamma(j - i)$ as a function of $t = t_j - t_i$, the distance between two grid points, as this is done for all other possible covariance functions, and use $\Delta$ as a parameter. Since $t_j - t_i = \Delta(j - i)$, this results in

$$\gamma(t) = \frac{\Delta^{2H}}{2} \left[ \frac{t}{\Delta} + 1 \right]^{2H} + \left| \frac{t}{\Delta} - 1 \right|^{2H} - 2 \left| \frac{t}{\Delta} \right|^{2H}. $$

The code also defines the increments as $\hat{X}(i)$, where $\hat{X}(i) = \Delta^{-H} X(i)$. Following the same procedure, we see that the new covariance function $\hat{\gamma}(t) = \Delta^{-2H} \gamma(t)$, so that

$$\hat{\gamma}(t) = \frac{1}{2} \left[ \frac{t}{\Delta} + 1 \right]^{2H} + \left| \frac{t}{\Delta} - 1 \right|^{2H} - 2 \left| \frac{t}{\Delta} \right|^{2H}. $$

(4.6)
This definition has the advantage that \( \hat{\gamma}(0) = 1 \). We can thus sample \( \hat{X}(i) \) using the circulant embedding method described in §4.2, and set \( B^H(0) = 0 \) and \( B^H(t_j) = \sum_{i=0}^{j-1} N^{-H} \hat{X}(i) \). Notice that we can simulate the Wiener process this way by setting \( H = \frac{1}{2} \), but since the increments are uncorrelated (and therefore independent), it is far more efficient to simulate the increments directly in this case.

### 4.5 Approximation in the case that a non-negative definite embedding matrix cannot be found

For both one- and two-dimensional random field simulation using the circulant embedding method, we require that the embedding matrix \( B \) is non-negative definite, or equivalently that all the elements of the diagonal matrix \( \Lambda \), containing the eigenvalues of \( B \), are all greater than or equal to zero. If this is not the case, we can use an approximation procedure. We can write \( \Lambda = \Lambda_+ + \Lambda_- \), where \( \Lambda_+ \) contains all of the positive and zero eigenvalues of \( B \), with any negative values replaced with zeros, and \( \Lambda_- \) contains all of the negative eigenvalues of \( B \), with any positive eigenvalues replaced with zeros. We can then write \( B = B_+ + B_- \), where \( B_+ = W^*\Lambda_+W \) and \( B_- = W^*\Lambda_-W \). Instead of using the covariance matrix \( B \), we now use the matrix \( \rho B_+ \) for some choice of nonzero \( \rho \). According to [12], the choice

\[
\rho = \frac{\text{Tr} \Lambda}{\text{Tr} \Lambda_+}
\]  

(4.7)

minimizes the random error

\[
\epsilon = \sqrt{\frac{(1-\rho)^2 \text{Tr}(\Lambda) + \rho^2 \text{Tr}(\Lambda_-)}{M}}
\]

due to having to approximate \( B \), and the choice

\[
\rho = \sqrt{\frac{\text{Tr} \Lambda}{\text{Tr} \Lambda_+}}
\]  

(4.8)

gives the correct one-dimensional marginal distributions of the random field. We could alternatively choose to set \( \rho = 1 \).
4.6 Cases for which the embedding matrix is always non-negative definite

One of the primary drawbacks of the circulant embedding method is that we cannot guarantee that the embedding matrix $B$ will be non-negative definite. Dembo, Mallows and Shepp [17] show that any positive definite Toeplitz matrix $A$ can be embedded in a circulant matrix; however their upper bound on the size $M$ is $O(\kappa(A)N^2)$, where $\kappa(A)$ is the condition number of $A$. Although Newsam and Dietrich [60] show that this bound can be improved to $O(\kappa(A)^{1/2}N^{5/4})$, the size $M$ is still likely to be infeasible. Despite this, Dietrich and Newsam [20, §3] show that we can guarantee a non-negative definite circulant embedding in certain cases, one of which is when simulating fractional Brownian motion. Firstly we have the following:

**Theorem 4.2** Let $\gamma(x)$ be a symmetric non-negative definite function. If its Fourier transform $\hat{\gamma}(\xi)$ is integrable with bounded variation, and the Fourier transform of $\hat{\gamma}(\xi)$ recovers the original function $\gamma(x)$, then the function

$$\eta_\Delta(\xi) = \gamma(0) + 2 \sum_{i=1}^{\infty} \gamma(i\Delta) \cos(2\pi i\xi)$$ (4.9)

takes non-negative values and is uniformly convergent and uniformly continuous for any $\Delta > 0$.

**Proof** See [20, §3, Theorem 1].

If $\gamma(x)$ satisfies the conditions in Theorem 4.2, then the $N' \times N'$ Toeplitz matrix $A = \text{symToe}(a)$, where $a$ has entries $a_i = \gamma(i\Delta)$ for $i = 0, \ldots, N' = 1$, has a minimal embedding matrix $B = \text{MCE}(a)$ of size $M = 2(N' - 1)$ in the following cases:

1. $\gamma(x)$ has finite support and is zero for $|x| > N'\Delta$;

2. $\gamma(x)$ does not have finite support, but the function $\eta_\Delta(\xi)$ defined in (4.9) takes strictly positive values - in this case there exists $N_{\text{min}}$ depending on $\Delta$ such if $N' \geq N_{\text{min}}$ then $B$ is non-negative definite;

3. The sequence of values $a_i = \gamma(i\Delta)$, for $i = 0, \ldots, N' - 1$, is convex and decreasing and $a_i \geq 0$ for all $i = 0, \ldots, N' - 1$. 
See \cite{20, §3, Corollary 1, Corollary 2, Theorem 2} for proofs. In particular, the increment process for fractional Brownian motion satisfies 3 above for \( H \in \left[ \frac{1}{2}, 1 \right] \) \cite{20}, and \cite{14} proves that a minimal non-negative definite embedding also exists for the case \( H \in (0, \frac{1}{2}) \).

These results impact which choice of padding is best to use as follows. Suppose that we are using the circulant embedding method with padding, with \( N \) being the number of sampling points for the one-dimensional random field and \( P \) the size of the padding vector. Then:

- If \( \gamma(x) = 0 \) for all \( |x| > N\Delta \), then padding with zeros and extending the grid are the same, since \( u_i = 0 \) for all \( i = 1, \ldots, P \);

- If \( \gamma(x) \) has finite support but is nonzero for some \( |x| > N\Delta \), or if \( \gamma(x) \) has infinite support but satisfies case 2 above, then padding by extending the grid is guaranteed to produce a non-negative embedding matrix if, respectively, \( \gamma(x) = 0 \) for \( |x| > (N + P)\Delta \) or if \( P \geq N_{\text{min}} - N \). However padding with zeros is not guaranteed to produce a non-negative definite embedding matrix since the sequence of values

\[
\alpha_i = \begin{cases} 
\gamma(i\Delta), & i = 0, \ldots, N - 1, \\
0, & i = N, \ldots, N + P - 1,
\end{cases}
\]

may not be convex as required by case 3 above.

### 4.7 Implementation in the NAG Fortran Library

One of the main contributions of this thesis is the implementation of both the one- and two-dimensional circulant embedding methods in Mark 24 of the NAG Fortran Library. As mentioned in Chapter 1, I spent a total of 12 weeks working with the Numerical Algorithms Group (NAG), most of which was spent developing code to implement the circulant embedding method. Grace Chan provided NAG with FORTRAN 77 code developed by herself and Andrew Wood \cite{12}, and it was my job to modify the code so that it could be included in the NAG Library, the routines for which are written in Fortran 90. Several tasks had to be done, including:
• Updating the syntax used so that it conformed to Fortran 90 syntax (some of the syntax had already been changed before I started working with the code), and updating the code to take advantage of features provided in Fortran 90 which were not included in FORTRAN 77, such as dynamic memory allocation, the use of an \texttt{IMPLICIT NONE} statement to disallow implicit declaration of variables, and easier manipulation of arrays. For details and other differences between FORTRAN 77 and Fortran 90, see for example [21];

• Updating calls to obsolete NAG Library routines, so that they used the latest versions of the NAG Library;

• Adding error checks so that illegal parameter values cannot be used, and problems within the calculation itself can be detected;

• Re-arranging the code into routines that performed different parts of the calculation - the final routines are described below.

In addition, I also wrote documentation so that users would know how to use the routines, and wrote small example programs to provide users with an example of a main program which used the routines.

Other staff at NAG wrote testing programs in order to confirm that the error checks were working correctly, and that the correct results were being output. For the parts of the code which produced realisations of the random field, statistical checks were performed on the results to check that the random fields had mean zero and the claimed covariance function.

4.7.1 The NAG Library routines

There are a total of seven library routines, each performing different tasks:

• \texttt{G05ZMF} and \texttt{G05ZNF} perform setup for producing realisations of a one-dimensional random field $Z(x)$. \texttt{G05ZMF} allows the user to input a custom variogram, whereas \texttt{G05ZNF} asks the user to select from a list of provided variograms, including a special case for fractional Brownian motion - this list is provided in §4.7.2. Both routines output the square roots of the eigenvalues of the embedding matrix $B$.
(not the eigenvalues themselves), performing the approximation procedure outlined in §4.5 to produce a non-negative definite embedding matrix if necessary;

- **G05ZPF** takes the square roots of the eigenvalues of $B$ produced by **G05ZMF** or **G05ZNF** and produces the realisations of the one-dimensional random field $Z(x)$;

- Similarly **G05ZQF** and **G05ZRF** perform setup for producing realisations of a two-dimensional random field $Z(x)$, by outputting the square roots of the eigenvalues of $B$ for custom variograms and pre-determined variograms respectively;

- **G05ZSF** produces the realisations of the two-dimensional random field $Z(x)$;

- **G05ZTF** is a special routine that produces realisations of fractional Brownian motion. The routine **G05ZMF** must be called in order to find the square roots of the eigenvalues of the embedding matrix.

We shall list all of the input parameters for the setup routines **G05ZMF**, **G05ZNF**, **G05ZQF** and **G05ZRF**. They are as follows:

- **NS**, the number of gridpoints to use (in **G05ZMF** and **G05ZNF** this is an integer, whereas in **G05ZQF** and **G05ZRF** this is a vector with two integer elements, indicating the number of gridpoints in the $x$- and $y$-directions respectively);

- **XMIN** and **XMAX**, the limits of the $x$-coordinates of the domain over which realisations are produced;

- **YMIN** and **YMAX**, the limits of the $x$-coordinates of the domain over which realisations are produced (**G05ZQF** and **G05ZRF** only);

- **MAXM**, the maximum size of embedding matrix to use (again this is an integer for **G05ZMF** and **G05ZNF** and a vector with two integer elements for **G05ZQF** and **G05ZRF**);

- **VAR**, a value by which the variogram is multiplied - so instead of using the variogram $\gamma(x)$ or $\gamma(x)$, the routine will use the variogram $\sigma^2\gamma(x)$ or $\sigma^2\gamma(x)$, where $\sigma^2$ corresponds to the value **VAR**;

- **COV1/COV2**, a subroutine for **G05ZMF/G05ZQF** respectively which calculates the variogram;
• ICOV1/ICOV2, an integer for G05ZNF/G05ZRF respectively which determines which of the preset variograms to use. See \[64\] \text{routine G05ZNF} \text{ or } \[64\] \text{routine G05ZRF} for the options available;

• EVEN (G05ZQF only), an integer which indicates whether the (user-input) variogram is even (EVEN = 1) or uneven (EVEN = 0);

• PAD, an integer which indicates whether to pad with zeros (PAD = 0) or to pad by extending the grid (PAD = 1);

• ICORR, an integer which indicates which value of $\rho$ to use for the approximation (if necessary). The options are 0 for $\rho$ as in (4.7), 1 for $\rho$ as in (4.8) or 2 for $\rho = 1$;

• IUSER and RUSER, for G05ZMF and G05ZQF, which are passed to the subroutine COV1 or COV2 in G05ZMF or G05ZQF respectively and can be used to change parameters in these subroutines;

• NP and PARAMS, for G05ZNF and G05ZRF, which are used to input parameters into the preset variogram. NP indicates how many are parameters are set and PARAMS is a vector containing the parameters themselves, which do different things depending on the variogram. For simulating fractional Brownian motion, the first element of PARAMS must be the Hurst parameter $H$ and the second element must be the stepsize $\frac{\text{XMAX}}{\text{NS}}$. See §4.7.2, \[64\] \text{routine G05ZNF} \text{ and } \[64\] \text{routine G05ZRF} for details.

The setup routines G05ZMF, G05ZNF, G05ZQF and G05ZRF produce the following output:

• $M$, the size of the embedding matrix. In G05ZQF and G05ZRF it is a vector with two elements, the first of which denotes the size of each block and the second of which denotes the number of blocks;

• LAM, the square roots of the eigenvalues of the embedding matrix;

• $XX$, the $x$-coordinates of the grid on which samples are to be output. For simulating fractional Brownian motion the points in $XX$ are the midpoints of the final gridpoints;
• YY, the y-coordinates of the grid on which samples are to be output (G05ZQF and G05ZRF only);

• APPROX, an integer which indicates whether approximation was used (APPROX = 1) or not (APPROX = 0).

• RHO, the value of $\rho$ used (if no approximation then RHO = 1);

• ICOUNT, the number of negative eigenvalues in the original matrix of eigenvalues $\Lambda$ which have been set to zero to produce $\Lambda_+$ (if no approximation then ICOUNT = 0);

• EIG, a vector containing information about the negative eigenvalues. $EIG(1)$ is the smallest eigenvalue (i.e. the negative eigenvalue with largest absolute value), $EIG(2)$ is the sum of the squares of the negative eigenvalues, and $EIG(3)$ is the sum of the absolute values of the negative eigenvalues (i.e. $-\text{Tr} \Lambda_-$). If no approximation was used then all the elements of EIG are zero.

The routines G05ZPF and G05ZSF produce the realisations of the one- or two-dimensional random field, using the outputs from the setup routine G05ZMF/G05ZNF or G05ZQF/G05ZRF respectively. G05ZTF does the same for fractional Brownian motion. Each of these three routines have the following inputs:

• NS, the number of gridpoints as before - it must be the same as input to the setup routine;

• S, the number of realisations of the random field to produce;

• M, the size of the embedding matrix as output from the setup routine;

• LAM, the square roots of the eigenvalues of the embedding matrix as output from the setup routine;

• RHO, the value of $\rho$ as output from the setup routine;

• STATE, a vector of integers which indicates which random number generator is being used and gives its state. This must be set up before using G05ZPF, G05ZSF or G05ZTF, and is updated and output at the end.
In addition \texttt{G05ZTF} has extra inputs:

- \( H \), the Hurst parameter of the fractional Brownian motion, which must equal the first element of \texttt{PARAMS} as input to \texttt{G05ZNF};

- \( \text{XMAX} \), the upper limit of the interval over which the fractional Brownian motion is simulated.

Each of the routines \texttt{G05ZPF}, \texttt{G05ZSF} and \texttt{G05ZTF} has an output \( Z \). For \texttt{G05ZPF} it is a \( NS \times S \) matrix, for \texttt{G05ZSF} it is a \( NS(1)NS(2) \times S \) matrix and for \texttt{G05ZTF} it is a \( (NS + 1) \times S \) matrix. In each case, each column of \( Z \) contains a realisation of the random field \( Z(x) \) or \( Z(x) \). For \texttt{G05ZPF}, \( Z(i, j) \) contains the value of \( Z(XX(i)) \) for the \( j \)-th realisation (starting indices from 1 rather than 0). For \texttt{G05ZSF}, \( Z(i + (j - 1)NS(1), k) \) contains the value of \( Z([XX(i), YY(j)])^T \) for the \( k \)-th realisation.

\texttt{G05ZTF} has an additional output \( XX \), which indicate the gridpoints on which values of the fractional Brownian motion are output (the output \( XX \) from \texttt{G05ZNF} contains the midpoints of the final gridpoints), so \( Z(i, j) \) contains the value of \( B^H(XX(i)) \) for the \( j \)-th realisation.

All of the routines have an input/output parameter \texttt{IFAIL}. On input, \texttt{IFAIL} can be set to 0, 1 or \(-1\) depending on whether a hard exit, soft exit without error messages or soft exit with error messages is desired respectively (a hard exit means that execution of the calling program is terminated, and a soft exit means that execution of the calling program is not terminated). On exit, \texttt{IFAIL} will be set to 0 if no error has occurred, or some other integer if an error has occurred. The values of \texttt{IFAIL} returned depend on the individual routine called; see [64] for details.

### 4.7.2 List of variograms for \texttt{G05ZNF} and \texttt{G05ZRF}

We now list the variograms which have been provided for use with \texttt{G05ZNF} and \texttt{G05ZRF}. Throughout this section we shall denote \( x' \) to be \( x' = \frac{|x|}{\ell} \) in the one-dimensional case and \( x' = \left\| \frac{x_1}{\ell_1}, \frac{x_2}{\ell_2} \right\| \) in the two-dimensional case, where \( \ell, \ell_1 \) and \( \ell_2 \) are positive correlation lengths, \( \sigma^2 = \gamma(0) \) is the variance specified by the variable \texttt{VAR} and the norm used is the one specified by the variable \texttt{NORM}. 
• The symmetric stable variogram

\[ \gamma_1(x) = \sigma^2 \exp\left(- (x')^\nu\right), \quad 0 < \nu \leq 2; \quad (4.10) \]

• The Cauchy variogram

\[ \gamma_2(x) = \sigma^2 \left(1 + (x')^2\right)^{-\nu}, \quad \nu > 0; \quad (4.11) \]

• The differential variogram with compact support

\[ \gamma_3(x) = \begin{cases} 
\sigma^2 \left(1 + 8x' + 25(x')^2 + 32(x')^3 \right) (1 - x')^8, & x' < 1, \\
0, & x' \geq 1; 
\end{cases} \quad (4.12) \]

• The exponential variogram \( \gamma_4(x) \), which is (4.10) with \( \nu = 1 \);

• The Gaussian variogram \( \gamma_5(x) \), which is (4.10) with \( \nu = 2 \);

• The nugget variogram \( \gamma_6(x) = \sigma^2 \mathbf{1}_0(x) \);

• The spherical variogram

\[ \gamma_7(x) = \begin{cases} 
\sigma^2 \left(1 - \frac{1.5x'}{x'} + 0.5(x')^3\right), & x' < 1, \\
0, & x' \geq 1; 
\end{cases} \quad (4.13) \]

• The Bessel variogram

\[ \gamma_8(x) = \sigma^2 \frac{2^\nu \Gamma(\nu + 1) J_\nu(x')}{(x')^\nu}, \quad \nu \geq 0, \quad (4.14) \]

where \( J_\nu(\cdot) \) is the Bessel function of the first kind;

• The hole effect variogram

\[ \gamma_9(x) = \sigma^2 \frac{\sin(x')}{x'}; \quad (4.15) \]

• The Whittle–Matérn variogram

\[ \gamma_{10}(x) = \sigma^2 \frac{2^{1-\nu}(x')^\nu K_\nu(x')}{\Gamma(\nu)}, \quad \nu > 0, \quad (4.16) \]

where \( K_\nu(\cdot) \) is the modified Bessel function of the first kind;
• The continuously parameterised variogram with compact support

\[
\gamma_{11}(x) = \frac{1}{\sigma^2} \gamma_{10}(x) \gamma_3(x_s) \\
= \begin{cases} 
\sigma^2 \frac{2^1 - (x'_s)^2 K_\nu(x'_s)}{\Gamma(\nu)} (1 + 8x'_s + 25(x'_s)^2 + 32(x'_s)^3) (1 - x'_s)^8, & x'_s < 1, \\
0, & x'_s \geq 1,
\end{cases}
\]

(4.17)

where \( \nu > 0 \) and

\[
x_s = \left[ \begin{array}{c} x \\ y \\ \end{array} \right] , \quad x'_s = \left\| \begin{array}{c} x \\ y \\ \ell_1 s_1 \\ \ell_2 s_2 \\ \end{array} \right\| ,
\]

for some correlation lengths \( s_1, s_2 > 0 \);

• The generalised hyperbolic distribution variogram

\[
\gamma_{12}(x) = \sigma^2 \left( \frac{\delta^2 + (x'_s)^2}{\delta^2 K_\lambda(\kappa \delta)} \right) K_\lambda\left(\kappa \left( \delta^2 + (x'_s)^2 \right)^{\frac{1}{2}} \right), \quad \delta, \kappa > 0,
\]

(4.18)

where \( K_\lambda(\cdot) \) is the modified Bessel function of the second kind;

• The cosine variogram

\[
\gamma_{13}(x) = \sigma^2 \cos(x'_s),
\]

(4.19)

which is only a valid covariance function for one-dimensional random fields;

• The variogram for the so-called ‘increment process’ of fractional Brownian motion. The scaled increments \( X(i) = \Delta^{-H}(B^H(t_i) - B^H(t_{i-1})) \) has variogram

\[
\gamma_{14}(x) = \frac{1}{2} \left[ \frac{x}{\Delta} + 1 \right]^{2H} + \left[ \frac{x}{\Delta} - 1 \right]^{2H} - 2 \left[ \frac{x}{\Delta} \right]^{2H}.
\]

(4.20)

4.8 Numerical examples

We present several realisations of random fields which have been generated using the NAG Fortran Library. The first example is the one-dimensional random field with variogram (4.11) and \( \sigma^2 = 1 \). Figures 4.1-4.4 each show four realisations of this random field, with different values for \( \nu \) and the correlation length \( \ell \), and Figure 4.5 plots the variogram (4.11) for the different parameter values. Each realisation was generated over the interval \([-1, 1]\) using 1000 gridpoints and a maximum size MAXM
of 2048 for the embedding matrix - this is the minimum size of embedding matrix possible so it is also the size of the final embedding matrix. **PAD** is set to 1 and **ICORR** is set to 2. The same random numbers and the same seed were used to generate each set of realisations.

For the parameter values $\ell = 0.01$ and $\nu = 0.5$, approximation was required. The value $\rho = 1$ was used for the approximation procedure (since **ICORR** was set to 2) and **ICOUNT** = 101. However, **EIG**(1) and **EIG**(2) were less than $10^{-5}$, and **EIG**(3) = $10^{-4}$ to 5 decimal places.

Figures 4.6 and 4.7 each show four realisations of fractional Brownian motion, with Hurst parameters of 0.2 and 0.8 respectively. The realisations were generated on the time interval [0, 5] with 500 grid points. The maximum size of the embedding matrix was 2048, but this time the final size of the embedding matrix was 1024. As before, **PAD** = 1 and **ICORR** = 2. No approximation was required for either case, and the same random numbers were again used for both examples.

Finally, figures 4.8-4.11 each show four realisations of the two-dimensional random field with variogram (4.10), where $\sigma^2 = 1$, and Figure 4.12 plots the variogram (4.10) in one dimension for the different values of $\nu$ and the longer of the correlation lengths used for each case. The one-dimensional variogram is plotted instead of the two-dimensional variogram for extra clarity. The random field is generated over the domain $[-1, 1]^2$ and plotted using a $500 \times 500$ grid. **MAXM** was set to [1024, 1024], which again is the minimum possible and so the embedding matrix was of size $1048576 \times 1048576$. As in the previous experiments, **PAD** was set to 1 and **ICORR** was set to 2, and the same random numbers were again used for all four experiments.

Approximation was required for the case $\ell_1 = 0.05$, $\ell_2 = 0.1$ and $\nu = 1.1$. This time $\rho = 1$ (since **ICORR** = 2), **ICOUNT** = 2065 and **EIG** = $[-0.06215, 0.61529, 25.91112]$.

### 4.8.1 Notes on the numerical examples

Each of Figures 4.1 - 4.4 were generated using the same random seed (so, for example, the top-left graph in each of the figures is generated using the random seed, the top-right graph in each of the figures is generated using the same random seed etc.). This is also the case for Figures 4.6 and 4.7 and Figures 4.8 - 4.11. The effect of using the same random seeds can be seen in Figures 4.1 - 4.4 (particularly Figures 4.3 and 4.4).
Figure 4.1: Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.01$ and $\nu = 0.5$.

Figure 4.2: Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.01$ and $\nu = 2.5$. 
Figure 4.3: Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.002$ and $\nu = 0.5$.

Figure 4.4: Four realisations of the one-dimensional random field with variogram (4.11), with $\sigma^2 = 1$, $\ell = 0.002$ and $\nu = 2.5$. 
CHAPTER 4. THE CIRCULANT EMBEDDING METHOD

Figure 4.5: Plots of the variogram (4.11) for the different values of $\ell$ and $\nu$.

Figure 4.6: Four realisations of fractional Brownian motion with Hurst parameter $H = 0.2$. 
Figure 4.7: Four realisations of fractional Brownian motion with Hurst parameter $H = 0.8$.

Figure 4.8: Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.25$, $\ell_2 = 0.5$ and $\nu = 1.1$. 
Figure 4.9: Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.5$, $\ell_2 = 0.25$ and $\nu = 1.8$.

Figure 4.10: Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.05$, $\ell_2 = 0.1$ and $\nu = 1.1$. 
Figure 4.11: Four realisations of the two-dimensional random field with variogram (4.10), with $\sigma^2 = 1$, $\ell_1 = 0.1$, $\ell_2 = 0.05$ and $\nu = 1.8$.

Figure 4.12: Plots of the variogram (4.10) in one dimension for different values of $\ell$ and $\nu$. 
and in Figures 4.6 and 4.7, where the random field realisations clearly follow a similar trajectory. This effect can also be seen between Figures 4.8 and 4.10 and between Figures 4.9 and 4.11. It is not so obvious between the other pairs of figures because the correlation lengths have been switched.

We would expect that the faster the variogram $\gamma(x)$ decreases to zero as $\|x\|$ increases, the smaller the correlation between two neighbouring gridpoints will be and hence the rougher the random field realisation will be. This is certainly the case for the one-dimensional examples. The variogram $\gamma(x)$ given by (4.11) decreases to zero faster for larger values of $\nu$ and smaller values of $\ell$, as illustrated by Figure 4.5, and it is clear that the realisations of the random fields in Figures 4.3 and 4.4 are rougher than those in Figures 4.1 and 4.2 respectively. For the two-dimensional random fields simulated in Figures 4.8 - 4.11, we can clearly see that the realisations are smoother for larger correlation lengths, and it appears that they are also smoother for $\nu = 1.8$ than for $\nu = 1.1$. The variogram $\gamma(x)$ given by (4.10) stays close to 1 for larger values of $\|x\|$ the larger $\mu$ is, as illustrated by Figure 4.12, although it does not decay to zero as quickly as $\|x\|$ increases.

As noted in §2.2.4, the increments of fractional Brownian motion are positively correlated for $H > \frac{1}{2}$ and negatively correlated for $H < \frac{1}{2}$. We would therefore expect there to be a large difference between successive increments for $H = 0.2$, resulting in rough paths for the realisations of fractional Brownian motion. For $H = 0.8$ we would expect a smaller difference between successive increments and hence a smoother path of the realisations - since the increments are likely to take similar values, the path of fractional Brownian motion is also likely to keep moving in the same direction as $t$ increases. This is indeed the case as illustrated by Figures 4.6 and 4.7.

For the two-dimensional random field, we have switched the correlation lengths between Figures 4.8 and 4.9, and again between Figures 4.10 and 4.11. When $\ell_1 > \ell_2$ we expect the realisations to be smoother in the $x$-direction than in the $y$-direction (and vice versa if $\ell_1 < \ell_2$); however this is not obvious from the figures.
4.8.2 Stringent tests

Stringent tests were written by other staff at NAG in order to test the routines. Firstly each stringent test checks is that any input parameters which are set to illegal values produce the correct value of \texttt{IFAIL} on output. Secondly each test runs examples where the correct outputs are known in advance. For example, the stringent test for \texttt{G05ZNF} takes different combinations of input parameters, and checks that both the square roots of the embedding matrix output in \texttt{LAM} and the approximation information vector \texttt{EIG} return the values expected. Since the results of \texttt{G05ZPF}, \texttt{G05ZSF} and \texttt{G05ZTF} are random, each of these routines are run several times with the same input parameters, and statistical checks are performed on the results to check that the mean and covariance of the random fields are as expected. Specifically, for 500 sets of 100 realisations of the random field, the sample means and cross-covariances of the columns of \texttt{Z} produced by \texttt{G05ZPF}, \texttt{G05ZSF} or \texttt{G05ZTF} are calculated using the NAG routine \texttt{G02BTF}, and these are checked against the predicted mean (which is zero), and the predicted cross-covariances (which depend on the variogram being used).
Part II

Solving Stochastic Differential Equations
Chapter 5

Background Material for SDEs

5.1 Stochastic Differential Equations

Stochastic differential equations (SDEs) are an extension of ordinary differential equations (ODEs) where the solution is partially driven by a Wiener process - the Wiener process was defined in §2.2.2. To study SDEs, it is necessary to define integrals of stochastic processes with respect to the Wiener process - the two most widely studied interpretations of these integrals are Itô and Stratonovich integrals, which are discussed in §5.1.1 which produce Itô and Stratonovich SDEs respectively. SDEs have many applications, including economics [35, §5.8] [52], population dynamics [27] and modelling the orbits of satellites [38, §7.5].

We define SDEs formally and introduce some notation and important results which will be used in later chapters.

5.1.1 Stochastic integration

SDEs can be defined using two types of stochastic integrals - Itô integrals and Stratonovich integrals. We shall use both extensively; a formal definition of stochastic integrals can be found in, for example, [34]. We shall denote the Itô integral of a predictable stochastic process \( g : [0, T] \times \Omega \rightarrow \mathbb{R} \) by

\[
I_{[0,T]}(g) = \int_0^T g(t, \omega) \, dW(t),
\]

where \( W(t) \) is a Wiener process, and the corresponding Stratonovich integral by

\[
J_{[0,T]}(g) = \int_0^T g(t, \omega) \circ dW(t).
\]
If \( g : [0, T] \to \mathbb{R}^d \) is vector-valued then we integrate componentwise; that is,

\[
I_{[0,T]}(g) = \int_0^T g(t, \omega) \, dW(t)
\]

has components

\[
(I_{[0,T]}(g))_i = \int_0^T g_i(t, \omega) \, dW(t),
\]

for \( i = 1, \ldots, d \), and similarly for \( J_{[0,T]}(g) \).

### 5.1.2 Definition of SDEs

We now define both Itô and Stratonovich SDEs, and we say that an SDE is **multi-dimensional** if it is driven by an \( m \)-dimensional Wiener process for \( m \geq 1 \), and **one-dimensional** if the Wiener process is one-dimensional - the dimension of the solution itself is not taken into account for the purposes of this definition. Numerical schemes to solve multi-dimensional SDEs can be significantly more complicated than those to solve one-dimensional SDEs; we shall examine this further in \( \S 5.1.7 \).

**Definition 5.1** For functions \( f, g_j : \mathbb{R}^d \to \mathbb{R}^d \) where \( j = 1, \ldots, m \), and an \( m \)-dimensional \( \mathcal{F}_t \)-Wiener process \( W(t) \), a multi-dimensional SDE is an equation of the form

\[
X(t) = x_0 + \int_0^t f(X(s)) \, ds + \sum_{j=1}^m \int_0^t g_j(X(s)) \, dW_j(s),
\]

where again \( t \in [0, T] \) and \( x_0 = X(0) \). This can also be written in differential form; \( 5.1 \) becomes

\[
dX(t) = f(X(t)) \, dt + \sum_{j=1}^m g_j(X(t)) \, dW_j(t), \quad X(0) = x_0.
\]

We can also write the diffusion term in matrix-vector form. Let \( g : \mathbb{R}^d \to \mathbb{R}^{d \times m} \) be the matrix-valued function given by \( g(x) = [g_1(x), \ldots, g_m(x)] \). Then \( 5.1 \) becomes

\[
X(t) = x_0 + \int_0^t f(X(s)) \, ds + \int_0^t g(X(s))dW(s),
\]

and \( 5.2 \) becomes

\[
dX(t) = f(X(t)) \, dt + g(X(t)) \, dW(t), \quad X(0) = x_0.
\]
Notes

• We shall study SDEs with time-independent coefficients, since any SDE with
time-dependent coefficients of the form
\[ dXXX(t) = \mathbf{f}(t, Y(t)) dt + \sum_{j=1}^{m} g_j(t, Y(t)) dW_j(t), \quad X(0) = x_0 \tag{5.5} \]
can be converted into an SDE of the form (5.2) by setting
\[
Y(t) = \begin{bmatrix} t \\ X(t) \end{bmatrix}, \quad \tilde{f}(Y(t)) = \begin{bmatrix} 1 \\ \mathbf{f}(t, X(t)) \end{bmatrix}
\]
and
\[
\tilde{g}_j(Y(t)) = \begin{bmatrix} 0 \\ g_j(t, X(t)) \end{bmatrix}, \quad j = 1, \ldots, m.
\]
However, many papers in the literature (for example [31–33]) study SDEs with
time-dependent coefficients of the form (5.5) and treat SDEs with additive noise,
where the diffusion functions \( \tilde{g}_j(t) \) depend only on \( t \), as a special case.

• The \( i \)-th component of \( g_j(x) \), equivalently the \( (i, j) \) component of \( g(x) \), shall be
denoted \( g_{ij}(x) \).

• It will often be convenient to write \( t = W_0(t) \) and \( \mathbf{f}(x) = g_0(x) \), particularly
when discussing multiple stochastic integrals. If we do this we can re-write (5.2) as
\[ dX(t) = \sum_{j=0}^{m} g_j(X(t)) dW_j(t), \quad X(0) = x_0. \tag{5.6} \]

• Using the Itô formula (5.9), the Itô SDE (5.2) can be converted into the Stratonovich
SDE
\[ dX(t) = \bar{\mathbf{f}}(X(t)) dt + \sum_{j=1}^{m} g_j(X(t)) \circ dW_j(t), \quad X(0) = x_0, \tag{5.7} \]
where \( \bar{\mathbf{f}} \) has components
\[
\bar{f}_i(x) = f_i(x) - \frac{1}{2} \sum_{j=1}^{m} \sum_{k=1}^{d} g_{kj}(x) \frac{\partial g_{ij}(x)}{\partial x_k}, \tag{5.8}
\]
for all \( i = 1, \ldots, d \). It may be easier to study (5.7) for error analysis purposes
than to study (5.2).
• For simplicity we shall always assume that the initial condition $X(0) = x_0$ is deterministic, and that the initial time is zero.

• The solutions $X(t)$ of (5.2) will sometimes be referred to as Itô diffusions, particularly in §5.2.4.

5.1.3 The Itô formula

We now state the Itô formula, which is a useful tool to evaluate Itô integrals and, as noted earlier, can be used to convert an Itô SDE into a Stratonovich SDE and vice versa.

Theorem 5.2 (The Itô formula) Let $X(t)$ be the solution of the SDE (5.1) with initial condition $X(0) = x_0$ and let $u(t,x) \in C^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R}^n)$ be a vector-valued function. Then

$$u(t,X(t)) = u(s,X(s)) + \int_s^t L^0 u(r,X(r)) \, dr + \sum_{j=1}^m \int_s^t L^j u(r,X(r)) \, dW_j(r), \quad (5.9)$$

where

$$L^0 = \frac{\partial}{\partial t} + \sum_{i=1}^d f_i \frac{\partial}{\partial x_i} + \frac{1}{2} \sum_{k=1}^m \sum_{i=1}^d \sum_{\ell=1}^d g_{ik} g_{\ell k} \frac{\partial^2}{\partial x_i \partial x_\ell}, \quad (5.10)$$

$$L^j = \sum_{i=1}^d g_{ij} \frac{\partial}{\partial x_i}. \quad (5.11)$$

Note that we can apply the Itô formula componentwise to each element of $u(t,x)$.

Proof See [40, pp. 92–93]. \qed

Similar expressions are available for Stratonovich SDEs. If $X(t)$ is the solution of (5.7) and $u(t,x) \in C^{1,2}([0,T] \times \mathbb{R}^d, \mathbb{R}^n)$, then

$$u(t,X(t)) = u(s,X(s)) + \int_s^t L^0 u(r,X(r)) \, dr + \sum_{j=1}^m \int_s^t \tilde{L}^j u(r,X(r)) \circ dW_j(r), \quad (5.12)$$

where

$$\tilde{L}^0 = \frac{\partial}{\partial t} + \sum_{i=1}^d f_i \frac{\partial}{\partial x_i}, \quad (5.13)$$

$$\tilde{L}^j = \sum_{i=1}^d g_{ij} \frac{\partial}{\partial x_i}. \quad (5.14)$$
5.1.4 Existence and uniqueness of solutions for SDEs

We state conditions necessary and sufficient for SDEs of the form (5.3), for a drift function $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ and a diffusion function $g : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times m}$ and a particular Wiener path $W(t)$, to have unique solutions.

**Definition 5.3** The drift function $f(x)$ and diffusion function $g(x)$ satisfy a linear growth condition if, for all $x \in \mathbb{R}^d$, there exists a constant $K$ such that

$$
\|f(x)\|_2^2 \leq K(1 + \|x\|_2^2),
$$

$$
\|g(x)\|_F^2 \leq K(1 + \|x\|_2^2).
$$

**Definition 5.4** The functions $f(x)$ and $g(x)$ satisfy a Lipschitz condition if, for all $x, y$ in $\mathbb{R}^d$, there exists some constant $L$ such that

$$
\|f(x) - f(y)\|_2 \leq L \|x - y\|_2,
$$

$$
\|g(x) - g(y)\|_F \leq L \|x - y\|_2.
$$

**Theorem 5.5** Suppose that $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ and $g : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times m}$ satisfy the linear growth condition and Lipschitz condition described in Definitions 5.3 and 5.4 respectively. Then there exists a unique solution to the SDE (5.4).

**Proof** See [34, pp. 167–190], and [48, pp. 51–55].

5.1.5 Notation for stochastic integrals

It is convenient to have a concise notation for stochastic integrals, especially multiple stochastic integrals, in order to express numerical schemes and state the Itô–Taylor Theorem, and hence local truncation errors for numerical schemes, in a concise form. We shall use the notation from [38] to abbreviate stochastic integrals where appropriate. The same notation is also used in [39], and similar notation is also used by [67].

Firstly we shall introduce the concept of a multi-index, which helps identify stochastic integrals.
Multi-indices

Definition 5.6 A row vector \( \alpha = (j_1, \ldots, j_\ell) \), where \( j_k \in \{0, 1, \ldots, m\} \) for all \( k = 1, \ldots, \ell \) and \( m \) is the dimension of the Wiener process being considered, is called a multi-index. The length of a multi-index \( \alpha \) is denoted \( \ell(\alpha) \), and the number of elements \( j_k = 0 \) contained in \( \alpha \) is denoted \( n(\alpha) \).

We shall need some more notation for multi-indices:

- The multi-index of length zero is denoted by \( \emptyset \).
- The set of all possible multi-indices is denoted \( \mathcal{M} \).
- For a multi-index \( \alpha = (j_1, \ldots, j_\ell) \) we use the notation \( \alpha - \) to denote the multi-index \( (j_1, \ldots, j_{\ell-1}) \) (\( \alpha \) with the last element deleted), and \( -\alpha \) to denote the multi-index \( (j_2, \ldots, j_\ell) \) (\( \alpha \) with the first element deleted).
- For multi-indices \( \alpha = (j_1, \ldots, j_{\ell_1}) \) and \( \beta = (k_1, \ldots, k_{\ell_2}) \) we use the notation \( \alpha \ast \beta \) to denote the multi-index \( (j_1, \ldots, j_{\ell_1}, k_1, \ldots, k_{\ell_2}) \). \( \ast \) is called the concatenation operator.

Next we define partitions of an interval \([t_-, t_+]\).

Partitions

Definition 5.7 A partition \( D([t_-, t_+]) \) of an interval \([t_-, t_+]\) is any set of the form

\[
D = \{t_- = t_0, t_1, \ldots, t_N = t_+\},
\]

where \( t_n < t_{n+1} \) for all \( n = 0, \ldots, N-1 \). \( D([t_-, t_+]) \) will often be abbreviated to \( D \) if the interval \([t_-, t_+]\) is obvious from the context. The \( n \)-th stepsize in \( D \) is \( \Delta t_n = t_{n+1} - t_n \), and the maximum stepsize of \( D \) is defined as \( \Delta t_{\text{max}} = \max_{n=0,\ldots,N-1}(\Delta t_n) \). The size of the partition \( D \) is \( N \).

We denote by \( D_{\Delta t}([t_-, t_+]) \) (or \( D_{\Delta t} \)) the partition for which the stepsize \( \Delta t = \frac{t_+ - t_-}{N} \) is fixed, i.e. \( \Delta t_n = \Delta t \) for all \( n = 0, \ldots, N - 1 \).

We define several sets of partitions as follows:

- The set of all partitions of \([t_-, t_+]\) is denoted by \( \mathcal{D}([t_-, t_+]) \);
The set of all partitions with maximum stepsize $\Delta t_{\text{max}}$ is denoted by $D_{\Delta t_{\text{max}}}(\llbracket t_-, t_+ \rrbracket)$;

We shall also define partitions and sets of partitions based on different numerical methods. One numerical method is to use a fixed stepsize for a certain scheme, in which case the partition $D_{\Delta t}$ is used.

Now we are in a position to abbreviate multiple stochastic integrals.

Multiple Itô stochastic integrals

For a real-valued, $\mathcal{F}_t$-adapted, right-continuous stochastic (or deterministic) process $\{Y(t) : t \in [0,T]\}$, an $m$-dimensional $\mathcal{F}_t$-Wiener process $W(t)$ and a multi-index $\alpha = (j_1, \ldots, j_\ell)$ we now define the multiple Itô stochastic integral

$$I_\alpha[Y(\cdot)]_{[s,t]} := \int_s^t \int_s^{s_1} \cdots \int_s^{s_\ell} Y(s_1) dW_{j_1}(s_1) \cdots dW_{j_\ell-1}(s_{\ell-1}) dW_{j_\ell}(s_\ell), \quad (5.16)$$

using the conventions $W_0(t) = t$ and $I_\emptyset[Y(\cdot)]_{[s,t]} = Y(t)$. In special cases we shall abbreviate further:

- $I_\alpha[1]_{[s,t]}$ will be abbreviated to $I_{\alpha,[s,t]}$;
- For a partition $D \in D([0,T])$ (see (5.15)), then:
  - For $k \geq n$ we shall abbreviate $I_\alpha[Y(\cdot)]_{[t_n,t_k]}$ to $I_{\alpha,[t_n,t_k]}$ and $I_{\alpha,[t_n,t_k]}$ to $I_{\alpha}^{nk}$,
  - We may also choose to abbreviate $I_{\alpha}^{n,n+1}[Y(\cdot)]$ and $I_{\alpha}^{n,n+1}$ to $I_{\alpha}^{n}[Y(\cdot)]$ and $I_{\alpha}^{n}$ respectively.

The integral $I_\alpha[Y(\cdot)]_{[s,t]}$ is well-defined if $Y(t) \in \mathcal{H}_\alpha$, where the sets $\mathcal{H}_\alpha$ are classes of stochastic processes. A stochastic process $Y(t)$ belongs to $\mathcal{H}_\alpha$ if it is $\mathcal{F}_t$-adapted, right-continuous and, for any $t \geq 0$:

- for $\alpha = \emptyset$, $|Y(t,\omega)| < \infty$;
- for $\alpha = (0)$, $\int_0^t |Y(s,\omega)| \, ds < \infty$;
- for $\alpha = (j)$ where $j = 1, \ldots, m$, $\int_0^t |Y(s,\omega)|^2 \, ds < \infty$;
- for any $\alpha$ where $\ell(\alpha) \geq 2$, the stochastic process $I_{\alpha-}[Y(\cdot)]_{[s,t]}$ belongs to $\mathcal{H}_{(j\ell)}$ for any $0 \leq s \leq t$. 

The stochastic integrals $I_\alpha[Y(\cdot)]_{[s,t]}$ can be defined recursively as follows:

\[
I_\alpha[Y(\cdot)]_{[s,t]} = \begin{cases} 
Y(t), & \ell(\alpha) = 0 \quad (\alpha = \emptyset), \\
\int_s^t I_\alpha[Y(\cdot)]_{[s,s_1]} \, ds_1, & \ell(\alpha) \geq 1, \ j_1 = 0, \\
\int_s^t I_\alpha-Y(\cdot)]_{[s,s_1]} dW_{j_\ell}(s_1), & \ell(\alpha) \geq 1, \ j_\ell \geq 1.
\end{cases}
\]

Clearly the following hold:

- For two stochastic processes $Y(t)$ and $Z(t)$,
  \[
  I_\alpha[Y(\cdot) + Z(\cdot)]_{[s,t]} = I_\alpha[Y(\cdot)]_{[s,t]} + I_\alpha[Z(\cdot)]_{[s,t]}; \quad (5.17)
  \]

- For any constant $\lambda$,
  \[
  I_\alpha[\lambda]_{[s,t]} = I_{\alpha,\lambda} \lambda; \quad (5.18)
  \]

- For any $s \leq t \leq u$,
  \[
  I_\alpha[Y(\cdot)]_{[s,t]} + I_\alpha[Y(\cdot)]_{[t,u]} = I_\alpha[Y(\cdot)]_{[s,u]}. \quad (5.19)
  \]

By applying the integration componentwise, it is also possible to extend the definition of stochastic integrals to vector-valued stochastic processes. These integrals will be denoted by $I_\alpha[Y(\cdot)]_{[s,t]}$.

### Multiple Stratonovich stochastic integrals

We shall define multiple Stratonovich integrals similarly; that is, for the multi-index $\alpha = (j_1, \ldots, j_\ell)$,

\[
J_\alpha[Y(\cdot)]_{[s,t]} := \int_s^t \int_s^{s_1} \cdots \int_s^{s_{\ell-1}} Y(s_1) \circ dW_{j_1}(s_1) \circ \cdots \circ dW_{j_{\ell-1}}(s_{\ell-1}) \circ dW_{j_\ell}(s_\ell), \quad (5.20)
\]

where $\circ$ denotes Stratonovich integration. The same abbreviations which apply to multiple Itô integrals will also apply to multiple Stratonovich integrals, and multiple Stratonovich integrals also satisfy $\textbf{(5.17)-\textbf{(5.19)}}$. We can extend this definition to vector-valued stochastic processes; these integrals will be denoted by $J_\alpha[Y(\cdot)]_{[s,t]}$. 
Some simple stochastic integrals

We evaluate several simple stochastic integrals as follows:

- \( I_{(0)}[s,t] = J_{(0)}[s,t] = t - s; \)
- \( I_{(0,0,...,0)}[s,t] = J_{(0,0,...,0)}[s,t] = \frac{(t-s)t}{\ell!}, \) where \( \ell \) is the length of the multi-index (see (5.68));
- For \( j \geq 1, I_{(j)}[s,t] = J_{(j)}[s,t] = W_j(t) - W_j(s), \) which shall be abbreviated to \( \Delta W_j[s,t]; \)
- For \( j \geq 1, I_{(j,j)}[s,t] = \frac{1}{2}(\Delta W_j^2[s,t] - (t-s)) \) and \( J_{(j,j)}[s,t] = \frac{1}{2}\Delta W_j(t,s)^2; \)
- For \( j,k \geq 1 \) where \( j \neq k, I_{(j,k)}[s,t] = J_{(j,k)}[s,t]. \) These integrals appear in the Milstein scheme for solving SDEs, but cannot be calculated solely from the Wiener path \( W(t). \) Approximations to \( I_{(j,k)}[s,t] \) are available, for example by using a Karhunen–Loève expansion of the Brownian bridge process as detailed in [28,38,39].

### 5.1.6 The Itô–Taylor and Stratonovich–Taylor Theorems

The Itô–Taylor and Stratonovich–Taylor Theorems generalise the deterministic Taylor Theorem to the solutions of SDEs, and can be used to express the local truncation error of a numerical scheme. We can write the Itô–Taylor and Stratonovich–Taylor Theorems in a concise fashion thanks to the notation for multiple stochastic integrals introduced above and some further notation which we shall now introduce. Firstly we define Itô and Stratonovich coefficient functions, and then we define hierarchical and remainder sets.

**Definition 5.8** For the Itô SDE (5.2), a function \( u : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^n \) for some \( n \in \mathbb{N} \) and a multi-index \( \alpha = (j_1, \ldots, j_\ell) \) we now define the Itô coefficient function \( F^u_\alpha(t,x) \) by

\[
F^u_\alpha(t,x) = L^{j_1} \cdots L^{j_\ell} u(t,x),
\]

(5.21)

using the conventions \( g_0(x) = f(x) \) and \( F^u_0(t,x) = u(t,x), \) where the operators \( L^j \) and \( L^j \) are given by (5.10) and (5.11) respectively. We now define the simple Itô
coefficient functions $F^\text{id}_\alpha(x)$ to be the Itô coefficient functions for $u(t,x) = x$. These can be written as

$$F^\text{id}_\alpha(x) = L^1 \ldots L^{j-1}g_j(x),$$

since $L^j(x) = g_j(x)$ for all $j \geq 0$.

Similarly, for the Stratonovich SDE (5.7), the Stratonovich coefficient function $G^\text{S}_\alpha(t,x)$ is

$$G^\text{S}_\alpha(t,x) = \bar{L}^1 \ldots \bar{L}^\ell g_\ell(x), \quad (5.22)$$

where the operators $\bar{L}^0$ and $\bar{L}^j$ are given by (5.13) and (5.14) respectively, and the simple Stratonovich coefficient functions are

$$G^\text{id}_\alpha(x) = \bar{L}^1 \ldots \bar{L}^\ell g_\ell(x). \quad (5.23)$$

Definition 5.9 A subset $A \subset M$ of multi-indices is a hierarchical set if:

- $A$ is non-empty;
- All multi-indices $\alpha \in A$ have finite length;
- For all $\alpha \in A \setminus \{\emptyset\}$, $-\alpha \in A$ also.

Definition 5.10 A remainder set for a hierarchical set $A$ is the set

$$B(A) = \{\alpha \in M \setminus A | -\alpha \in A\},$$

that is, for all $\alpha$ in the hierarchical set $A$, $B(A)$ contains all possible multi-indices $(j) \ast \alpha$ which are not already in $A$.

We are finally in a position to state the Itô–Taylor Theorem.

Theorem 5.11 (Itô–Taylor Theorem) Let $X(t)$ be the solution to the Itô SDE (5.2) for some functions $f, g_j : \mathbb{R}^d \to \mathbb{R}^d$ where $j = 1, \ldots, m$. Then for any two stopping times $s, t \in [0,T]$, any hierarchical set $A \subset M$ and any function $u : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^n$,

$$u(t,X(t)) = \sum_{\alpha \in A} F^u_\alpha(s,X(s))I_{\alpha,[s,t]} + \sum_{\alpha \in B(A)} I_{\alpha}[F^u_\alpha(\cdot,X(\cdot))]_{[s,t]}, \quad (5.24)$$

where $F^u_\alpha(t,X)$ are the Itô coefficient functions associated with the function $u$, assuming all required derivatives of $f, g_j$ for $j = 1, \ldots, m$ and $u$ exist.
Proof See [38, §5.5]. □

Next we state the following important corollary, which deals with the case where $u(t, X) = X$.

**Corollary 5.12** In the case $u(t, x) = x$, (5.24) becomes

$$X(t) = \sum_{\alpha \in A} F^\text{id}_\alpha(X(s))I_{\alpha,[s,t]} + \sum_{\alpha \in B(A)} I_\alpha[F^\text{id}_\alpha(X(\cdot))]_{[s,t]}.$$  (5.25)

The Stratonovich–Taylor Theorem is similar, and has a similar corollary.

**Theorem 5.13 (Stratonovich–Taylor Theorem)** Let $X(t)$ be the solution to the Stratonovich SDE (5.7) for some functions $f, g_j : \mathbb{R}^d \to \mathbb{R}^d$ where $j = 1, \ldots, m$. Then for any two stopping times $s, t \in [0, T]$, any hierarchical set $A \subset M$ and any function $u : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^n$,

$$u(t, X(t)) = \sum_{\alpha \in A} G^u_\alpha(s, X(s))J_{\alpha,[s,t]} + \sum_{\alpha \in B(A)} J_\alpha[G^u_\alpha(\cdot, X(\cdot))]_{[s,t]},$$  (5.26)

where $G^u_\alpha(t, X)$ are the Stratonovich coefficient functions associated with the function $u$, assuming all required derivatives of $f, g_j$ for $j = 1, \ldots, m$ and $u$ exist.

Proof See [38, §5.6]. □

**Corollary 5.14** In the case $u(t, x) = x$, (5.26) becomes

$$X(t) = \sum_{\alpha \in A} G^\text{id}_\alpha(X(s))J_{\alpha,[s,t]} + \sum_{\alpha \in B(A)} J_\alpha[G^\text{id}_\alpha(X(\cdot))]_{[s,t]}.$$  (5.27)

### 5.1.7 Numerical schemes for solving SDEs

When solving an SDE over an interval $[0, T]$, we divide $[0, T]$ into smaller intervals $[t_n, t_{n+1}]$ using a partition (to be defined below) and approximate the true solution $X(t_n)$, at each of the times $t_n$, by an approximation $X_n$. A numerical scheme is used to obtain the value of $X_{n+1}$ from $X_n$ (and possibly earlier values, depending on the scheme).

We distinguish between numerical schemes and numerical methods, which we define as being a particular algorithm for solving an SDE, using a numerical scheme (or
possibly two or more schemes) to advance the solution. A numerical method may involve using a pre-determined partition (such as a fixed-step method) or an adaptive partition (known as using adaptive stepsizes). We shall discuss various numerical methods in Chapter 6.

**Definition 5.15** Let $D \in \mathcal{D}([0,T])$ be a partition. A one-step numerical scheme $S$ for solving (5.2) or (5.7) is a recurrence relation of the form

$$X_{n+1} = S_n(X_n) = S(X_n, t_n, t_{n+1}, W(\theta) - W(t_n), t_n \leq \theta \leq t_{n+1})$$ (5.28)

where $X_0 = x_0$ and $X_n$ is a numerical approximation of $X(t_n)$, for $n = 0, \ldots, N$. For a $k$-step numerical scheme, for $k > 1$, $X_{n+1}$ also depends on $X_{n-\ell+1}$ for $\ell = 2, \ldots, k$.

We will often use the abbreviations

$$\Delta t_n = t_{n+1} - t_n,$$ (5.29)
$$\Delta W_n = W(t_{n+1}) - W(t_n),$$ (5.30)
$$\Delta W^n_j = W_j(t_{n+1}) - W_j(t_n).$$ (5.31)

when expressing numerical schemes. There are many numerical schemes available, and we define some of them here - others will be discussed briefly in §6.1.

**Euler-type schemes for solving the Itô SDE (5.2)**

Euler-type schemes can be used to solve SDEs in Itô form. We have:

- The (explicit) Euler–Maruyama scheme EM

$$\text{EM}^\theta_n(X_n) = X_{n+1} = X_n + f(X_n)\Delta t_n + \sum_{j=1}^m g_j(X_n)\Delta W^n_j;$$ (5.32)

- The semi-implicit Euler-type $\theta$-scheme $\text{EM}^\theta$ [28, 37, 54, 58]

$$\text{EM}^\theta_n(X_n) = X_{n+1} = X_n + ((1 - \theta)f(X_n) + \theta f(X_{n+1})) \Delta t_n$$
$$+ \sum_{j=1}^m g_j(X_n)\Delta W^n_j,$$ (5.33)

where $\theta \in [0, 1]$. Setting $\theta = 0$ recovers the Euler–Maruyama scheme, and the scheme obtained by setting $\theta = 1$ is a backward Euler-type scheme.
Maruyama [49, Theorem 1] proves that, for any $\Delta t_{\text{max}}$ and any partition $D \in \mathcal{D}_{\Delta t_{\text{max}}} ([0, T])$, that the approximate solution of the SDE (5.1) for $d = m = 1$ obtained using the partition $D$ and the Euler–Maruyama scheme (5.32) converges to the real solution in the $L^2 (\Omega)$ (mean square) sense, as long as $f(x)$ and $g(x)$ satisfy a global Lipschitz condition. Yamada [78] proved that the Euler–Maruyama scheme converges in the strong sense as $\Delta t_{\text{max}} \to 0$, for any partition $D \in \mathcal{D}_{\Delta t_{\text{max}}} ([0, T])$, if $f(x)$ and $g(x)$ satisfy a linear growth condition, $f(x)$ is globally Lipschitz and $|g(x) - g(y)| \leq \rho(|x - y|)$ for some function $\rho$ such that $\rho(0) = 0$, $\rho$ is increasing on $\mathbb{R}^+$ and 

$$
\int_0^\infty \frac{1}{\rho^2 (u)} \, du = +\infty.
$$

Note that fixed stepsizes are not required for either of these proofs. It is well known that the Euler–Maruyama scheme, when used with a fixed stepsize, converges with a strong order of $1/2$ - see, for example, [38, Theorem 10.2.2], for a proof.

**Heun-type schemes for solving the Stratonovich SDE (5.7)**

We can use Heun-type schemes to solve SDEs in Stratonovich form, which are:

- The (explicit) *Euler–Heun scheme* $EH$

  $$
  \dot{X}_n = X_n + \sum_{j=1}^m g_j (X_n) \Delta W^n_j, \tag{5.34}
  $$

  $EH_n (X_n) = X_{n+1} = X_n + \tilde{f} (X_n) \Delta t_n + \frac{1}{2} \sum_{j=1}^m \left( g_j (\dot{X}_n) + g_j (X_n) \right) \Delta W^n_j, \tag{5.35}
  $$

- The *semi-implicit Heun-type $\theta$-scheme* $EH^\theta$

  $$
  EH^\theta_n (X_n) = X_{n+1} = X_n + \left( (1 - \theta) \tilde{f} (X_n) + \theta \tilde{f} (X_{n+1}) \right) \Delta t_n 
  $$

  $$
  + \frac{1}{2} \sum_{j=1}^m \left( g_j (\dot{X}_n) + g_j (X_n) \right) \Delta W^n_j, \tag{5.36}
  $$

  where $\dot{X}_n$ is given by (5.34) and $\theta \in [0, 1]$.

**The Milstein schemes**

The explicit and implicit *Milstein* schemes are suitable for solving both the Itô SDE (5.2) and the Stratonovich SDE (5.7) (with slight modifications). The explicit Milstein
scheme Itô-Mil and implicit Milstein scheme Itô-Mil\(^\theta\) for solving (5.2) are given by

\[
\text{Itô-Mil}_n(X_n) = X_{n+1} = S_n^{EM}(X_n) + \sum_{j=1}^{m} \sum_{r=1}^{m} h_{jr}(X_n) I_{(j,r)}^n \tag{5.37}
\]

and

\[
\text{Itô-Mil}_n^{\theta}(X_n) = X_{n+1} = S_n^{EM,\theta}(X_n) + \sum_{j=1}^{m} \sum_{r=1}^{m} h_{jr}(X_n) I_{(j,r)}^n \tag{5.38}
\]

respectively, where

\[
h_{jr}(x) = Dg_r(x)g_j(x) = L^j g_r(x), \quad j, r = 1, \ldots, m \tag{5.39}
\]

and again \(\theta \in [0, 1]\). The explicit and implicit Milstein schemes Str-Mil and Str-Mil\(^\theta\) for solving (5.7) are given by

\[
\text{Str-Mil}_n(X_n) = X_{n+1} = X_n + \bar{f}(X_n) \Delta t_n + \sum_{j=1}^{m} g_j(X_n) \Delta W_j^n + \sum_{j=1}^{m} \sum_{r=1}^{m} h_{jr}(X_n) J_{(j,r)}^n \tag{5.40}
\]

and

\[
\text{Str-Mil}_n^{\theta}(X_n) = X_{n+1} = X_n + ((1 - \theta) \bar{f}(X_n) + \theta \bar{f}(X_{n+1})) \Delta t_n + \sum_{j=1}^{m} g_j(X_n) \Delta W_j^n + \sum_{j=1}^{m} \sum_{r=1}^{m} h_{jr}(X_n) J_{(j,r)}^n \tag{5.41}
\]

respectively, with \(h_{jr}(x)\) given by (5.39). If it is clear whether we are solving an Itô SDE or a Stratonovich SDE (or if it does not matter which we are solving), we will use the notation Mil and Mil\(^\theta\) for the explicit and implicit Milstein schemes respectively.

Note that for \textit{constant} diffusion functions \(g(x)\), the final term vanishes for each of the Milstein schemes, since \(\frac{\partial g_j(x)}{\partial x_i} = 0\) for all \(i = 1, \ldots, d\) and all \(j = 1, \ldots, m\). This reduces the Itô Milstein scheme to the Euler–Maruyama scheme (but the Stratonovich scheme does not reduce to the Euler–Heun scheme).

Milstein [53] created the explicit Milstein scheme by adding an extra term to the Euler–Maruyama scheme for solving SDEs of the form (5.1) for the case \(d = m = 1\), and he proved that the approximate solution \(X_n\) of the true solution \(X(t_n)\) obtained using the partition \(D_{\Delta t}\), for some fixed stepsize \(\Delta t = \frac{T}{N}\), and the explicit Milstein scheme (5.37), satisfies

\[
\mathbb{E}[(X(T) - X_N)^2] = \mathcal{O}(\Delta t^2).
\]
Simplication of the Milstein scheme

Since the ‘off-diagonal’ integrals $I_n^{(j,r)}$ (or $J_n^{(j,r)}$) cannot be calculated exactly from the Wiener path $W(t)$ alone, for $j \neq r$, we would like to avoid calculating these integrals whenever possible. By re-arranging the last term of each of the Milstein schemes, it is possible to find diffusion functions $g(x)$ where calculation of the off-diagonal integrals is unnecessary, saving valuable computation time. We shall look at the Itô case only, for $m > 1$; the Stratonovich case is similar since $I_n^{(j,r)} \neq J_n^{(j,r)}$ only if $j = r$, and the case $m = 1$ is trivial.

By using the identity $I_n^{(j,r)} + I_n^{(r,j)} = I_n^{(j,j)} + I_n^{(r,r)} = \Delta W_j^n \Delta W_r^n$, it is possible to write

$$
\sum_{j=1}^m \sum_{r=1}^m h_{jr}(X_n) I_n^{(j,r)} = \sum_{j=1}^m h_{jj}(X_n) I_n^{(j,j)} + \sum_{r=1}^m \sum_{j<r} \left[ (h_{jr}(X_n) + h_{rj}(X_n)) (I_n^{(j,r)} + I_n^{(r,j)}) \right] + (h_{jr}(X_n) - h_{rj}(X_n)) (I_n^{(j,j)} - I_n^{(r,r)})
$$

$$
= \sum_{j=1}^m h_{jj}(X_n) I_n^{(j,j)} + \frac{1}{2} \sum_{r=1}^m \sum_{j<r} (h_{jr}(X_n) + h_{rj}(X_n)) (\Delta W_j^n \Delta W_r^n)
$$

$$
+ \sum_{r=1}^m \sum_{j<r} (h_{jr}(X_n) - h_{rj}(X_n)) A_{jr}, \tag{5.42}
$$

where $A_{jr} = \frac{1}{2} \left( I_n^{(j,r)} - I_n^{(r,j)} \right)$ is known as the Lévy area and $h_{jr}(x)$ is given by (5.39). Hence, if $h_{jr}(x) = h_{rj}(x)$ for all $j, r = 1, \ldots, m$, all we need are the integrals $I_n^{(j,j)}$ and the Wiener increments $\Delta W_j^n$ for $j = 1, \ldots, m$. We say that the diffusion function $g(x)$ is commutative if this is the case, and diagonal if $h_{jr}(x) = 0$ for all $j \neq r$ (diagonality implies commutativity).

If the diffusion function $g(x)$ is not commutative then, by (5.42), we can still perform the Milstein scheme by approximating the Lévy areas $A_{jr}$. There are various methods for doing this; one method is described in [26] and will be discussed in Chapter 6.
5.1.8 Local truncation errors for numerical schemes

Any scheme \( S \) used to solve the Itô SDE \( \text{(5.2)} \) has a local truncation error, which is the error accrued by taking one step of the scheme \( S \). To help define the local truncation error formally, let \( Y_{s,z}(t) \) denote the solution of the SDE

\[
dY_{s,z}(t) = f(Y_{s,z}(t)) \, dt + \sum_{j=1}^{m} g_j(Y_{s,z}(t)) \, dW_j(t), \quad Y_{s,z}(s) = z,
\]

where \( t \in [s, T] \).

**Definition 5.16** Let \( S \) be a numerical scheme used to solve \( \text{(5.2)} \) on the interval \([0, T]\) and let \( D \in \mathcal{D}([0, T]) \) be a partition. Then for \( n = 0, \ldots, N - 1 \), the local truncation error (LTE) of the scheme \( S \) is

\[
T^S_{[t_n, t_{n+1}]}(z) = T^S_n(z) = Y_{t_n, z}(t_{n+1}) - S_n(z),
\]

where \( Y_{t_n, z}(t) \) is the solution of \( \text{(5.43)} \) with \( s = t_n \) there, for \( t \in [t_n, T] \). In particular, we shall define

\[
T^S_n = T^S_n(X_n) = Y_{t_n, X_n}(t_{n+1}) - X_{n+1},
\]

by setting \( z = X_n \).

Some explicit numerical schemes can be associated with hierarchical sets; the multi-indices contained in the hierarchical set indicate which stochastic integrals are used by the scheme, and the integrals used are always from a hierarchical set. We can write such an explicit numerical scheme \( S \), with hierarchical set \( \mathcal{A}_S \), as

\[
X_{n+1} = S_n(X_n) = \sum_{\alpha \in \mathcal{A}_S} F^{id}_\alpha(X_n) I^n_\alpha.
\]

We shall now use \( \text{(5.43)} \) and Corollary 5.12 to express the LTE of \( S \) using the remainder set \( \mathcal{B}_S \). Corollary 5.12 applied to \( \text{(5.43)} \), with \( s = t_n, t = t_{n+1} \) and \( z = X_n \), yields

\[
Y_{t_n, X_n}(t_{n+1}) = \sum_{\alpha \in \mathcal{A}_S} F^{id}_\alpha(Y_{t_n, X_n}(t_n)) I^n_\alpha + \sum_{\alpha \in \mathcal{B}_S} I^n_\alpha[F^{id}_\alpha(Y_{t_n, X_n}(\cdot))] = S_n(X_n) + \sum_{\alpha \in \mathcal{B}_S} I^n_\alpha[F^{id}_\alpha(Y_{t_n, X_n}(\cdot))] \quad \text{from (5.46)}.
\]
Hence, from (5.45),
\[ T_n^S = \sum_{\alpha \in B_S} I_{\alpha}^{n}[F_{\alpha}^{id}(Y_{tn}, x_n(\cdot))]. \] (5.47)

We are particularly interested in the Euler–Maruyama and Milstein schemes, so we list the hierarchical and remainder sets for these schemes.

- The hierarchical set \( A_{EM} \) associated with the Euler–Maruyama scheme is
\[ A_{EM} = \{ \emptyset, (0), (j) \mid j = 1, \ldots, m \} = \{ \alpha \mid \ell(\alpha) \leq 1 \}, \] (5.48)
and the remainder set \( B_{EM} \) is
\[ B_{EM} = \{ (0, 0), (j, 0), (0, r), (j, r) \mid j, r = 1, \ldots, m \}. \] (5.49)

- The hierarchical set \( A_{Mil} \) associated with the Milstein scheme is
\[ A_{Mil} = \{ \emptyset, (0), (j), (j, r) \mid j, r = 1, \ldots, m \} = \{ \alpha \mid \ell(\alpha) + n(\alpha) \leq 2 \} \] (5.50)
and the remainder set \( B_{Mil} \) is
\[ B_{Mil} = \{ (0, 0), (j, 0), (0, r), (0, r, p), (j, r, p) \mid j, r, p = 1, \ldots, m \}. \] (5.51)

We find bounds for the LTEs of the Euler–Maruyama and Milstein schemes in Lemma 5.26.

A similar expression for the truncation error can be found for the numerical solution of the Stratonovich SDE (5.7). Let \( Y_{tn}, x_n(t) \) be the solution of
\[ dY_{s,z}(t) = \tilde{f}(Y_{s,z}(t)) dt + \sum_{j=1}^{m} g_{j}(Y_{s,z}(t)) \circ dW_{j}(t), \quad Y_{s,z}(s) = z, \] (5.52)
where \( t \in [s, T] \). The truncation error associated with the scheme \( S \) is now
\[ T_n^S = \sum_{\alpha \in B_S} J_{\alpha}^{n}[G_{\alpha}^{id}(Y_{tn}, x_n(\cdot))]]. \] (5.53)

5.1.9 Types of global error and order of a numerical scheme

There are several types of global error associated with the numerical solution of SDEs of the form (5.2): strong global error, weak global error and pathwise global error. We now define these types of global errors and the associated types of order of a numerical
scheme. For some $T > 0$ and a fixed stepsize $\Delta t = \frac{T}{N}$, let $D_{\Delta t}$ be the partition of $[0, T]$ with fixed stepsize $\Delta t$, let $S$ be a numerical scheme and let $X_n$ be the numerical approximation to the true solution $X(t_n)$ of (5.2) produced using the scheme $S$ and the partition $D$.

**Definition 5.17** The strong global error, or $SGE$, of the numerical approximation at a time $t_n \in D$ is

$$
\mathbb{E} [\|X_n - X(t_n)\|_2],
$$

(5.54)

and the scheme $S$ has strong order $\gamma$ if

$$
\max_{0 \leq n \leq N} \mathbb{E} [\|X_n - X(t_n)\|_2] \leq K_{SGE} \Delta t^\gamma,
$$

where $K_{SGE}$ is a constant which depends only on the initial condition $x_0$, $f$ and $g_j$ for $j = 1, \ldots, m$.

An alternative to the strong error is the strong error in $p$-th mean for some $p \geq 1$, which is given by

$$
\mathbb{E} [\|X_n - X(t_n)\|_2^p]^{\frac{1}{p}}.
$$

(5.55)

The strong error in second mean is known as the strong error in mean square. Müller–Gronbach \[59\] develops a variable stepsize method to reduce the strong error in $p$-th mean, where the discretisation of the interval $[0, T]$ depends on $p$. Others, for example \[31,33,44,45\], aim to reduce the strong error in mean square.

**Definition 5.18** Let $C$ be the class of functions $u : \mathbb{R}^d \to \mathbb{R}^d$ which satisfy the following: for all $x \in \mathbb{R}^d$, there exist constants $K, \kappa > 0$ such that

$$
\|u(x)\| \leq K (1 + \|x\|^\kappa).
$$

Let $\gamma > 0$, and now suppose that $\phi : \mathbb{R}^d \to \mathbb{R}^d$ and all of its derivatives of order up to $2\gamma + 2$ belong to the class $C$. Then the weak global error, or $WGE$, of the numerical approximation at a time $t_n \in D$ is

$$
\|\mathbb{E}[\phi(X_n)] - \mathbb{E}[\phi(X(t_n))]|,
$$

(5.56)

and the scheme $S$ has weak order $\gamma$ if

$$
\|\mathbb{E}[\phi(X_n)] - \mathbb{E}[\phi(X(t_n))]| \leq K_{WGE} \Delta t^\gamma.
$$
for all $\phi \in C$, where $K_{WGE}$ is a constant which depends only on the function $\phi$, $f$ and $g_j$ for $j = 1, \ldots, m$.

**Definition 5.19** The pathwise global error, or PGE, of the scheme $S$ at a time $t_n \in D$ is

$$\|X_n - X(t_n)\|_2,$$  \hspace{1cm} (5.57)

and the pathwise global error is of order $\gamma$ if

$$\max_{0 \leq n \leq N} \|X_n - X(t_n)\|_2 \leq K_{PGE}(\omega) \Delta t^{\gamma - \epsilon}$$

almost surely for all $0 < \epsilon < \frac{1}{2}$, where $K_{PGE}(\omega)$ is a random variable depending on $\epsilon$, the functions $f(x)$ and $g_j(x)$ for $j = 1, \ldots, m$ and the path of $W(t)$.

### 5.1.10 Stopping times

Stopping times are a particular type of random time, which we now define. We shall see later that the Dynkin formula is only valid at stopping times, and we will use the exit time of a Wiener process from an open domain, which is an example of a stopping time, as the basis for an algorithm for solving SDEs in Chapter 8.

**Definition 5.20** For $T > 0$ and a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t \geq 0}, \mathbb{P})$, a function $\tau : \Omega \to [0, T]$ is a stopping time with respect to the filtration $\{\mathcal{F}_t\}_{t \geq 0}$ if the event $\{\omega : \tau < t\}$ is in $\mathcal{F}_t$ for all $t \geq 0$.

Any deterministic time is automatically a stopping time. Another example of a stopping time is the first exit time of a stochastic process from an open set. Let $\{X(t) : t \in [0, T]\}$ be a $d$-dimensional stochastic process and let $A \subset \mathbb{R}^d$ be an open set which contains $X(0)$. Then the first exit time of $X(t)$ from $A$, given by

$$\tau = \inf\{t \in (0, T] : X(t) \notin A\},$$

is a stopping time.

### 5.2 Important results for pathwise error analysis of SDEs

We introduce some of the results which will be needed in our analysis of pathwise convergence of numerical methods for solving SDEs.
Firstly we state the Burkholder–Davis–Gundy inequalities and the Kolmogorov–Čentsov Theorem. Both of these results can then be used in order to find bounds for multiple stochastic integrals, and using these bounds we can then bound the LTE for any explicit numerical scheme. Bounds for LTEs will be used extensively in Chapter 8.

Next we state and prove the Dynkin formula, which allows us to derive partial differential equations needed to simulate bounded Wiener increments in Chapter 7, and finally we state and prove results concerning the modulus of continuity of a stochastic process, which is used to bound pathwise global errors in Chapter 8.

5.2.1 The Burkholder–Davis–Gundy inequalities and Kolmogorov–Čentsov Theorem

The Burkholder–Davis–Gundy inequalities and the Kolmogorov–Čentsov Theorem will later allow us to find bounds for multiple stochastic integrals and LTEs.

Lemma 5.21 (Burkholder–Davis–Gundy inequalities) Suppose that an \( \mathbb{R}^d \)-valued predictable stochastic process \( \{X(t) : t \geq 0\} \) satisfies

\[
\mathbb{P} \left( \int_0^T \|X(s)\|_2^2 \, ds < \infty \right) = 1.
\]

Then, for any stopping time \( t > 0 \) and any \( p > 1 \), both of the following hold:

\[
c_p \left( \int_0^t \mathbb{E} \left[ \|X(s_1)\|_2^2 \right]^\frac{p}{2} \, ds_1 \right)^\frac{2}{p} \leq \mathbb{E} \left[ \|I_{(j)}[X(\cdot)]|_{0,t}\|_2^p \right] \leq C_p \left( \int_0^t \mathbb{E} \left[ \|X(s_1)\|_2^2 \right]^\frac{p}{2} \, ds_1 \right)^\frac{2}{p}, \tag{5.58}
\]

\[
k_p \mathbb{E} \left[ \left( \int_0^t \|X(s_1)\|_2^2 \, ds_1 \right)^\frac{p}{2} \right] \leq \mathbb{E} \left[ \sup_{0 \leq r \leq t} \|I_{(j)}[X(\cdot)]|_{0,r}\|_2^p \right] \leq K_p \mathbb{E} \left[ \left( \int_0^t \|X(s_1)\|_2^2 \, ds_1 \right)^\frac{p}{2} \right], \tag{5.59}
\]

where \( c_p, C_p, k_p \) and \( K_p \) are constants which depend only on \( p \). In particular when \( d = 1 \) and \( X(t) = 1 \), (5.58) becomes

\[
c_p t^\frac{p}{2} \leq \mathbb{E} \left[ |I_{(j)}|_{0,t}^p \right] \leq C_p t^\frac{p}{2}. \tag{5.60}
\]
CHAPTER 5. BACKGROUND MATERIAL FOR SDES

Proof See [35, Proposition 3.26]. □

Note that (5.59) applies for any $p > 0$; for a proof of this fact, see [35, Theorem 3.28].

Theorem 5.22 (Kolmogorov–Centsov Theorem) Suppose that an $\mathbb{R}^d$-valued stochastic process $\{X(t) : t \in [0, T]\}$ satisfies

$$E[\|X(t) - X(s)\|_2^p] \leq C|t - s|^{1+b}$$

for any $s, t \in [0, T]$, and any $a, b, C > 0$. Then there exists a continuous modification $\tilde{X}(t)$ of $X(t)$, which is locally Hölder continuous with exponent $\gamma$ for all $\gamma \in (0, \frac{b}{a})$; that is, there exists a constant $h(\omega)$ depending on the sample path of $X(t)$, and a constant $K_\gamma(\omega)$ depending on both the sample path of $X(t)$ and $\gamma$, such that

$$\mathbb{P}\left\{\omega \mid \sup_{0 \leq t - s < h(\omega)} \frac{\|\tilde{X}(t; \omega) - \tilde{X}(s; \omega)\|_2}{|t - s|^{\gamma}} \leq K_\gamma(\omega) \right\} = 1,$$

or alternatively

$$\|\tilde{X}(t) - \tilde{X}(s)\|_2 \leq K_\gamma(\omega)|t - s|^{\gamma}$$

almost surely for all $0 < t - s < h(\omega)$.

See [35, pp. 53–4].

5.2.2 Bounds for stochastic integrals

We can bound multiple stochastic integrals where the integrand is a predictable stochastic process, as demonstrated by the following result.

Lemma 5.23 Let $\{Y(t) : t \in [0, T]\}$ be a predictable $\mathbb{R}^d$-valued stochastic process with continuous sample paths. Then for all multi-indices $\alpha$ with $\ell(\alpha) \geq 1$ and all $\epsilon > 0$, there exists a random variable $C_{Y, \epsilon, \alpha}(\omega)$, depending on the $L^p(\Omega)$-norm of $Y(t)$ for $p = \lceil \frac{1}{\epsilon} \rceil$, $\epsilon, \alpha$ and the sample path of $W(t)$, such that

$$\|I_\alpha[Y(\cdot)]_{[s,t]}\|_2 \leq C_{Y, \epsilon, \alpha}(\omega)|t - s|^{\ell(\alpha) + \frac{\ell(\alpha) - 1}{2} \epsilon}. \quad (5.61)$$

almost surely for all $0 \leq s \leq t \leq T$.  

CHAPTER 5. BACKGROUND MATERIAL FOR SDES

116

Proof We prove the result for \(d = 1\) (the case \(d \geq 1\) follows). Firstly we shall prove by induction on \(\ell(\alpha)\) that, for \(p = \left\lceil \frac{1}{\epsilon} \right\rceil\),

\[
E[|I_{\alpha}[Y(\cdot)]|_{s,t}] \leq C_{Y,\epsilon,\alpha}^p |t - s|^p \left(\frac{(\ell(\alpha) + n(\alpha))}{2}\right),
\]

(5.62)

for some constant \(C_{Y,\epsilon,\alpha}\) depending on the \(L^p(\Omega)\)-norm of \(Y(t)\), the multi-index \(\alpha\) and \(\epsilon\).

For the base case suppose \(\ell(\alpha) = 1\). There are two cases to consider: \(\alpha = (0)\) and \(\alpha = (j)\), for \(j \geq 1\). Firstly we prove (5.62) for \(\alpha = (0)\). By (2.3),

\[
E \left[ |I_{(0)}[Y(\cdot)]|_{s,t} \right] \leq E \left[ \left| \int_s^t Y(s_1) \, ds_1 \right|^p \right] \\
\leq E \left[ |t - s|^{p-1} \int_s^t |Y(s_1)|^p \, ds_1 \right] \\
\leq |t - s|^{p-1} \int_s^t E \left[ |Y(s_1)|^p \right] \, ds_1 \\
\leq |t - s|^{p-1} |t - s| \sup_{s \leq s_1 \leq t} E \left[ |Y(s_1)|^p \right] \\
\leq C_{Y,\epsilon,(0)}^p |t - s|^p,
\]

where \(C_{Y,\epsilon,(0)} = \sup_{0 \leq s \leq T} \| Y(s) \|_{L^p(\Omega)}\), a constant depending only on the \(L^p(\Omega)\)-norm of \(Y\), \(\alpha = (0)\) and \(p\), or equivalently the \(L^p(\Omega)\)-norm of \(Y(t)\), \(\alpha = (0)\) and \(\epsilon\). Hence, since \(\ell(\alpha) = 1\) and \(n(\alpha) = 1\), (5.62) holds for \(\alpha = (0)\).

Now we prove (5.62) for \(\alpha = (j)\) for any \(j \geq 1\). By the first Burkholder–Davis–Gundy inequality (5.58) there exists a constant \(C_p\) depending only on \(p\) such that,

\[
E \left[ |I_{(j)}[Y(\cdot)]|_{s,t} \right] \leq C_p \left( \int_s^t E \left[ |Y(s_1)|^p \right] \frac{2}{p} \, ds_1 \right)^{\frac{p}{2}} \\
\leq C_p \left( \sup_{s \leq s_1 \leq t} E \left[ |Y(s_1)|^p \right]^{\frac{2}{p}} \left| t - s \right| \right)^{\frac{p}{2}} \\
\leq C_p \sup_{s \leq s_1 \leq t} E \left[ |Y(s_1)|^p \right] \left| t - s \right|^{\frac{p}{2}} \\
\leq C_{Y,\epsilon,(1)}^p |t - s|^{\frac{p}{2}},
\]

where \(C_{Y,\epsilon,(1)} = C_p^\frac{1}{2} \sup_{0 \leq s \leq T} \| Y(s) \|_{L^p(\Omega)}\) is a constant which again depends only on the \(L^p(\Omega)\)-norm of \(Y(t)\), \(\alpha = (j)\) and \(p\), or equivalently the \(L^p(\Omega)\)-norm of \(Y(t)\), \(\alpha = (j)\) and \(\epsilon\). Since \(\ell(\alpha) = 1\) and \(n(\alpha) = 0\), (5.62) also holds for \(\alpha = (j)\) for all \(j \geq 1\).

Now let us suppose that (5.62) holds for some \(\alpha\). We must now prove, for the multi-index \(\bar{\alpha} = \alpha * (j)\), that

\[
E[|I_{\bar{\alpha}}[Y(\cdot)]|_{s,t}] \leq C_{Y,\epsilon,\bar{\alpha}} |t - s|^p \left(\frac{(\ell(\bar{\alpha}) + n(\bar{\alpha}))}{2}\right),
\]

(5.63)
where $C_{Y,t,\alpha}$ depends only on the $L^p(\Omega)$-norm of $Y(t)$, $\bar{\alpha}$ and $\epsilon$, and

$$
\ell(\bar{\alpha}) = \ell(\alpha) + 1, \quad n(\bar{\alpha}) = \begin{cases} 
 n(\alpha) + 1, & j = 0, \\
 n(\alpha), & j \geq 1.
\end{cases}
$$

Firstly we look at the case $j = 0$. Then

$$
\mathbb{E}[|I_\alpha[Y(\cdot)]_{[s,t]}|^p] = \mathbb{E} \left[ \left| \int_s^t I_\alpha[Y(\cdot)]_{[s,s_1]} \, ds \right|^p \right]
$$

$$
\leq \mathbb{E} \left[ |t - s|^{p-1} \int_s^t |I_\alpha[Y(\cdot)]_{[s,s_1]}|^p \, ds \right] \quad \text{by (2.3)}
$$

$$
= |t - s|^{p-1} \mathbb{E} \left[ \int_s^t |I_\alpha[Y(\cdot)]_{[s,s_1]}|^p \, ds \right]
$$

$$
= |t - s|^{p-1} \int_s^t \mathbb{E}[|I_\alpha[Y(\cdot)]_{[s,s_1]}|^p] \, ds
$$

$$
\leq |t - s|^{p-1} \int_s^t C_{Y,t,\alpha}^{p}|s_1 - s|^p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right) \, ds_1 \quad \text{from (5.62)}
$$

$$
= C_{Y,t,\alpha}^{p}|t - s|^{p-1}\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right) = C_{Y,t,\alpha}^{p}|t - s|^{p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right)},
$$

where $C_{Y,t,\alpha} = C_{Y,t,\alpha} \left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right)^{-\frac{1}{p}}$. Hence (5.63) holds for $\bar{\alpha} = \alpha * (0)$. If $j \geq 1$, then we can use the first Burkholder–Davis–Gundy inequality (5.58) to prove the result. So

$$
\mathbb{E}[|I_\alpha[Y(\cdot)]_{[s,t]}|^p] = \mathbb{E} \left[ \left| \int_s^t I_\alpha[Y(\cdot)]_{[s,s_1]} \, dW_j(s_1) \right|^p \right]
$$

$$
\leq C_p \left( \int_s^t \left( \mathbb{E}[|I_\alpha[Y(\cdot)]_{[s,s_1]}|^p] \right)^{\frac{p}{2}} \, ds_1 \right)^{\frac{p}{2}} \quad \text{from (5.58)}
$$

$$
\leq C_p \left( \int_s^t \left( C_{Y,t,\alpha}^{p}|s_1 - s|^p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right) \right)^{\frac{p}{2}} \, ds_1 \right)^{\frac{p}{2}} \quad \text{from (5.62)}
$$

$$
\leq C_p C_{Y,t,\alpha}^{p} \left( \int_s^t \left| s_1 - s \right|^p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right) \, ds_1 \right)^{\frac{p}{2}}
$$

$$
= C_{Y,t,\alpha}^{p}|t - s|^{p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right)} = C_{Y,t,\alpha}^{\frac{1}{2}}|t - s|^{p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right)},
$$

where $C_{Y,t,\alpha} = C_{Y,t,\alpha} C_p \left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right)^{-\frac{1}{p}}$. Hence (5.63) holds for all $\bar{\alpha} = \alpha * (j)$ where $j \geq 0$.

Finally we use the Kolmogorov–Čentsov Theorem (Theorem 5.22) with $a = p$, $b = p \left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right) - 1$ and $p \geq \frac{1}{\epsilon}$, to obtain

$$
|I_\alpha[Y(\cdot)]_{[s,t]}| \leq C_{Y,t,\alpha}(\omega)|t - s|^{p\left(\frac{\ell(\bar{\alpha}) + n(\bar{\alpha})}{2}\right)} - \epsilon
$$

(5.64)
almost surely, where \( C_{Y,\epsilon,\alpha}(\omega) \) is a random variable depending on \( Y(t) \), \( \epsilon \) and the path of the Wiener process.

\[ I_{\alpha,[s,t]} \leq C_{\epsilon}(\omega)|t - s|^\frac{\ell(\alpha)+n(\alpha)}{2} - \epsilon. \]  

(5.65)

In particular

\[ |\Delta W_{j,[s,t]}| \leq C_{\epsilon}(\omega)|t - s|^{1-\epsilon} \]  

(5.66)

and

\[ |I_{(j,r),[s,t]}| \leq C_{\epsilon}(\omega)|t - s|^{1-\epsilon}. \]  

(5.67)

**Proof** Set \( d = 1 \) and \( Y(t) = 1 \) in Lemma 5.23.

**Corollary 5.25** If \( \alpha \) is the multi-index of length \( \ell \) with all entries zero, then

\[ I_{\alpha,[s,t]} = \frac{1}{\ell!}(t - s)^\ell. \]  

(5.68)

**Proof** Note that

\[ I_{\alpha,[s,t]} = \int_s^t \int_s^{s_2} \ldots \int_s^{s_3} \int_s^{s_2} \, ds_1 \, ds_2 \ldots \, ds_{\ell-1} \, ds_{\ell}. \]

Performing the integration yields the result.

### 5.2.3 Local truncation error bounds for explicit numerical schemes

We saw earlier that the LTE for such a scheme is given by (5.47), where \( Y_{t_n},X_n(t) \) is the solution of (5.43) with \( s = t_n \) and \( z = X_n \). Now that we have bounds for stochastic integrals, we can bound the LTE of any explicit scheme \( S \) associated with a hierarchical set \( A_S \) and a remainder set \( B_S \).
Lemma 5.26 (LTE of an explicit numerical scheme) Let $X(t)$ be the solution of (5.2) with initial condition $X(0) = x_0$, let $D \in \mathcal{D}([0,T])$ be a partition and let $S$ be a numerical scheme with associated hierarchical set $\mathcal{A}_S$ and remainder set $\mathcal{B}_S$. Suppose also that a continuous, predictable solution of (5.2) exists and is unique. Then, for any $0 < \epsilon < \frac{1}{2}$, we can bound the LTE of the scheme $S$ by

$$\|T_S^n\|_2 \leq K_{X,\epsilon}(\omega)|\Delta t_n|^\frac{\beta}{2} - \epsilon,$$  (5.69)

where $\beta = \min_{\alpha \in \mathcal{B}_S}(\ell(\alpha) + n(\alpha))$ and $K_{X,\epsilon}(\omega)$ is a random variable which depends on the $L^p(\Omega)$-norm of $X(t)$ for $p = \lceil \frac{1}{\epsilon} \rceil$, $\epsilon$ and the path of $W(t)$, providing that all of the stochastic integrals and derivatives appearing in (5.47) exist.

Corollary 5.27 (LTE of the Euler–Maruyama scheme) The LTE of the Euler–Maruyama scheme $S^{EM}$ is bounded by

$$\|T^{EM}_n\|_2 \leq K_{X,\epsilon}(\omega)|\Delta t_n|^{1-\epsilon}.$$  (5.70)

Corollary 5.28 (LTE of the Milstein scheme) The LTE of the Milstein scheme $Mil$ is bounded by

$$\|T^{Mil}_n\|_2 \leq K_{X,\epsilon}(\omega)|\Delta t_n|^\frac{3}{2} - \epsilon.$$  (5.71)

Proof of Lemma 5.26

We can bound $\|T_S^n\|_2$ using the triangle inequality and (5.61), obtaining

$$\|T_S^n\|_2 \leq \sum_{\alpha \in \mathcal{B}_S} \|I_n^\alpha[F^{id}_\alpha(Y_{t_n},X_n(\cdot))]\|_2 \leq \sum_{\alpha \in \mathcal{B}_S} K_{X,\alpha,\epsilon}(\omega)|\Delta t_n|^\frac{\ell(\alpha) + n(\alpha)}{2} - \epsilon,$$

where the $K_{X,\alpha,\epsilon}(\omega)$ are random variables depending on the $L^p(\Omega)$-norm of $X(t)$, $\epsilon$ and the sample path of $W(t)$. The result now follows.

Proof of Corollaries 5.27 and 5.28

These follow from Lemma 5.26 because $\beta = 1$ for the remainder set $\mathcal{B}_EM$ and $\beta = \frac{3}{2}$ for the remainder set $\mathcal{B}_Mil$.

5.2.4 The Dynkin formula

We shall use the Dynkin formula to find the exit time of an $m$-dimensional $\mathcal{F}_t$-Wiener process from a set $G \in \mathbb{R}^m$; this allows us to simulate a Wiener process using bounded increments, which we study in Chapter 7.
Let $X(t)$ be the solution of (5.4) with initial condition $X(0) = x_0$ (so $X(t)$ is an Itô diffusion). We now define the generator of $X(t)$.

**Definition 5.29** The (infinitesimal) generator $A$ of the Itô diffusion $X(t)$ is defined by

$$Au(x) = \lim_{t \downarrow 0} \frac{E[u(X(t))] - u(x)}{t}, \quad x \in \mathbb{R}^d.$$  \hspace{1cm} (5.72)

We denote by $G_A(x)$ the set of functions $u : \mathbb{R}^d \to \mathbb{R}$ for which the above limit exists at $x$, and by $G_A$ the set of functions $u$ for which the limit exists for all $x \in \mathbb{R}^d$.

Now we state the Dynkin formula.

**Theorem 5.30 (The Dynkin formula)** For an Itô diffusion $X(t)$ with generator $A$, a function $u \in C^2_0(\mathbb{R}^d)$ and a stopping time $\tau$,

$$E[u(X(\tau))] = u(x_0) + E \left[ \int_0^\tau Au(X(s))ds \right].$$  \hspace{1cm} (5.73)

If $\tau$ is the first exit time of $X(t)$ from any bounded set $G \subset \mathbb{R}^d$, (5.73) applies for any $u \in C^2(\mathbb{R}^d)$.

**Proof** See [65, pp. 121–125]. \hfill \Box

### 5.2.5 Modulus of continuity of stochastic processes

We introduce a concept called the modulus of continuity, which helps our analysis of pathwise global errors in Chapter 8.

**Definition 5.31** Let $\{Y(t) : t \in [0,T]\}$ be an $\mathbb{R}^d$-valued stochastic process. The modulus of continuity of $Y(t)$ on $[0, T]$ is the function $w_Y(h; T)$ defined by

$$w_Y(h; T) = \sup_{s,t \in [0,T], \ |t-s| \leq h} \|Y(t) - Y(s)\|_2.$$  \hspace{1cm} (5.74)

**Properties of the modulus of continuity**

It follows from Definition 5.31 that:

- $w_Y(0; T) = 0$;
• For \( h_1 \leq h_2 \) and \( T > 0 \), \( w_YY(h_1;T) \leq w_YY(h_2;T) \);

• For \( T_1 \leq T_2 \) and \( h > 0 \), \( w_YY(h;T_1) \leq w_YY(h;T_2) \).

We now state a result from [22], which gives us an upper bound on the moments of the modulus of continuity of Itô processes.

**Lemma 5.32** Let \( YY(t) \) be the \( \mathbb{R}^d \)-valued Itô process given by

\[
YY(t, \omega) = y_0 + \int_0^t u(s, \omega) \, ds + \sum_{j=1}^m \int_0^t v_j(s, \omega) \, dW_j(s), \quad t \in [0,T],
\]

where \( W(t) \) is a \( \mathcal{F}_t \)-Wiener process and \( u, v_j : [0, T] \times \Omega \rightarrow \mathbb{R}^d \) are \( \mathcal{F}_t \)-adapted stochastic processes for \( j = 1, \ldots, m \). Now let \( \zeta, \xi > 0 \) be random variables such that, for all \( s, t \in [0, T] \),

\[
\int_s^t \|u(r, \omega)\|_2^2 \, dr \leq \zeta(\omega) \sqrt{|t-s| \ln \left( \frac{2T}{|t-s|} \right)},
\]

\[
\int_s^t \|v_j(r, \omega)\|_2^2 \, ds \leq \xi(\omega)|t-s|,
\]

and, for some \( p \geq 1 \), there exists \( \mu > 0 \) and \( K > 0 \) such that \( \mathbb{E}[\zeta^p] \leq K \) and \( \mathbb{E}[\xi^{p+\mu}] \leq K \). Then there exists a constant \( C_p > 0 \), depending only on \( p, K \) and \( \mu \) such that

\[
\|w_YY(h;T)\|_{L^p(\Omega)} = \mathbb{E}[w_YY(h;T)^p]^{\frac{1}{p}} \leq C_p \sqrt{h \ln \left( \frac{2T}{h} \right)},
\]

for all \( 0 < h \leq T \).

**Proof** See [22, Section 3, Theorem 1]. \( \square \)

**Corollary 5.33** Suppose that Lemma [5.32] holds for some \( p > 2 \). Then, for any \( 0 < \alpha < \frac{1}{2} - \frac{1}{p} \), any \( \gamma \in \left( 0, \frac{1}{2} - \frac{1}{p} - \alpha \right) \) and any \( |t-s| \leq h \), there exists a random variable \( K_{p,\alpha} \in L^p(\Omega) \), depending only on \( C_p \) and \( \alpha \), such that

\[
\|YY(t) - YY(s)\|_2 \leq K_{p,\alpha}|t-s|^{\gamma}
\]

almost surely.
Proof From [4, p. 301], \( \ln(x) \leq \frac{x^\beta}{\beta} \) for any \( \beta > 0 \). Hence, setting \( \beta = 2\alpha \),
\[
\ln \left( \frac{2T}{h} \right) \leq \frac{(2T)^{2\alpha}}{2\alpha} h^{-2\alpha},
\]
and, for \( p \geq 1 \) and \( |t - s| \leq h \),
\[
\|Y(t) - Y(s)\|_{L^p(\Omega, \mathbb{R}^d)} \leq \|w_Y(h; T)\|_{L^p(\Omega)}
\]
\[
\leq C_p \sqrt{h \ln \left( \frac{2T}{h} \right)} \quad \text{from (5.78)}
\]
\[
\leq C_{p, \alpha} h^{\frac{1}{2} - \alpha}, \quad C_{p, \alpha} = C_p \frac{(2T)^{\alpha}}{\sqrt{2\alpha}}.
\]
By taking both sides to the power of \( p \),
\[
E[\|Y(t) - Y(s)\|_{L^p}^p] \leq C_{p, \alpha}^p h^{p \left( \frac{1}{2} - \alpha \right)}.
\]
Since \( p > \frac{1}{2 - \alpha} \), we can now use the Kolmogorov–Čentsov Theorem (Theorem 5.22) with \( a = p \) and \( b = p \left( \frac{1}{2} - \alpha \right) - 1 \) to obtain
\[
\|Y(t) - Y(s)\|_2 \leq K_{p, \alpha} h^\gamma
\]
almost surely, for all \( 0 < \gamma < \frac{1}{2} - \frac{1}{p} - \alpha \). Hence the result follows. \( \square \)
Chapter 6

Existing Variable Stepsize
Numerical Methods for Solving SDEs

We examine some of the existing literature regarding numerical methods for solving SDEs; in particular, those which use variable stepsizes and aim to reduce the strong global error or the pathwise global error. In particular we review numerical methods developed by Gaines and Lyons in [26], and Lamba in [42, 43], both of which use the Milstein scheme but with different error controls to determine the stepsize, and different methods for constructing the Wiener path.

6.1 Numerical Schemes

Any fixed stepsize or variable stepsize method for solving SDEs uses a numerical scheme which, in general, produces a numerical approximation $X_{n+1}$ to the true solution $X(t_{n+1})$ from previous approximations $X_{n-k}$ to the true solution $X(t_{n-k})$ for some $k \geq 0$. A numerical scheme is a one-step scheme if $k = 0$, and a multi-step scheme if $k > 0$. Numerical schemes can also be divided into explicit schemes, where the formula which determines $X_{n+1}$ is explicit, and implicit schemes, where the formula is implicit.
6.1.1 Explicit numerical schemes

Explicit schemes are simpler to implement than implicit schemes because, since $X_{n+1}$ does not depend on itself, we do not need to solve a nonlinear equation at each step. Most explicit numerical schemes are either based on Itô–Taylor or Stratonovich–Taylor expansions of $X(t_{n+1})$ about the point $X(t_n)$ - we call these strong Itô–Taylor or strong Stratonovich–Taylor schemes - or belong to the class of stochastic Runge–Kutta schemes which avoid the evaluation of derivatives of the drift function $f(x)$ and the diffusion functions $g_j(x)$.

Explicit Itô–Taylor and Stratonovich–Taylor schemes

Several numerical schemes for solving SDEs belong to the class of strong Itô–Taylor or Stratonovich–Taylor schemes. The first of these are the Euler–Maruyama and explicit Milstein schemes, which were discussed in §5.1.7. Platen and Wagner [67] derived the Itô–Taylor expansion and used it to produce the class of strong Itô–Taylor schemes. The Itô–Taylor scheme of strong order 1.5 for solving the SDE (5.2) for $m = 1$ over an interval $[0,T]$, for a Wiener process $W(t)$, is given by

$$X_{n+1} = \text{Itô-Mil}_n(X_n) + \frac{1}{2}(L^1 f)(X_n) \Delta t_n^2 + (L^0 g)(X_n) (\Delta t_n \Delta W_n - \Delta Z_n) + \frac{1}{6}(L^1 L^0 g)(X_n) (\Delta W_n^3 - \Delta t_n \Delta W_n),$$

(6.1)

where $\Delta Z_n$ is a random variable representing the stochastic integral $I^1_{(1,0)}$ and can be simulated by simulating a random variable $U_n \sim N(0,1)$ and setting

$$\Delta Z_n = \frac{1}{2} \Delta t_n \left( \Delta W_n + \sqrt{\frac{\Delta t}{3}} U_n \right).$$

It is possible to generalise this scheme to multi-dimensional SDEs, and schemes of order 2 and higher can be derived; see for example [38, Chapter 10].

Newton [61] altered the scheme (6.1) by replacing the terms $(L^1 f)(X_n) \Delta Z_n$ and $(L^0 g)(X_n) (\Delta t_n \Delta W_n - \Delta Z_n)$, which are not $\mathcal{P}_N$-measurable, with their $\mathcal{P}_N$-conditional means $\frac{1}{2}(L^1 f)(X_n) \Delta t_n \Delta W_n$ and $\frac{1}{2}(L^0 g)(X_n) \Delta t_n \Delta W_n$ respectively. $\mathcal{P}_N$ is the partition $\sigma$-algebra, which is defined as the smallest $\sigma$-algebra generated by the values of $W(n \Delta t)$ for $n = 0, \ldots, N$. He then proved that, using a fixed stepsize $\Delta t = \frac{T}{N}$, the resulting scheme is first-order asymptotically efficient; that is, for all nonzero vectors
\[ c \in \mathbb{R}^d, \]
\[
\lim_{N \to \infty} \inf \mathbb{E} \left[ \Delta t^{-2} \left( c^T (X(T) - X_N) \right)^2 \mid P_N \right] + 1 = 1.
\]

almost surely. The resulting scheme can also be extended to commutative SDEs driven by an \( m \)-dimensional Wiener process.

**Explicit Runge–Kutta schemes and other explicit schemes**

Itô–Taylor and Stratonovich–Taylor schemes suffer from the disadvantage that derivatives of the drift function \( f(x) \) and the diffusion functions \( g(x) \) have to be known. It is possible to avoid calculating derivatives by using a stochastic Runge–Kutta scheme. Stochastic Runge–Kutta schemes bear some relation to the deterministic Runge–Kutta schemes first developed in [41, 71], but are necessarily more complex. The Euler–Heun scheme is an example of a one-stage stochastic Runge–Kutta scheme and it can be used to solve Stratonovich SDEs of the form (5.7). Rümelin [70] states the form of the most general \( n \)-stage stochastic Runge–Kutta scheme for any \( n \geq 1 \) and proves that the approximate solution of the SDE (5.7) produced using this scheme with a fixed stepsize converges in mean square to the true solution, assuming that \( f \) and all of its first derivatives, and \( g_j \) and all of its first and second derivatives are bounded for all \( j = 1, \ldots, m \). He also states that if a stochastic Runge–Kutta scheme, using a fixed stepsize \( \Delta t \) and only values of the Wiener process \( W(t) \) at discrete points, is used to solve (5.7) for \( m > 1 \), the best possible local truncation error is \( O(\Delta t^3) \) (which, according to [7], equates to strong order 1.5), and this local truncation error is only possible for SDEs with commutative noise.

Specific examples of stochastic Runge–Kutta schemes have been proposed by Newton, and Burrage and Burrage. Newton [63] proposes two Runge–Kutta schemes of strong order 1 for solving Stratonovich SDEs, one of which is first-order asymptotically efficient, and both of which can be modified to solve Itô SDEs. The schemes there only use Wiener increments and do not involve any of the double stochastic integrals. In contrast, Burrage and Burrage [7] introduce stochastic Runge–Kutta schemes which have strong order 2. They use the classical fourth-order deterministic Runge–Kutta scheme (which is designed for solving ODEs and described in, for example, [10]) for the deterministic component, and the stochastic component uses the stochastic integrals...
Many **multistep schemes** are available to solve ODEs, for example the Adams–Bashforth and Adams–Moulton schemes [30]. Some of these schemes can also be modified and used to solve SDEs. Milstein proposed a two-step difference scheme in [53]. Two two-step schemes are suggested by [38, §11.4], one with strong order 1.0 and another with strong order 1.5, to solve SDEs of the form
\[
\begin{bmatrix}
X(t) \\
Y(t)
\end{bmatrix} = \begin{bmatrix}
Y(t) \\
-a(t)Y(t) + b(t, X(t))
\end{bmatrix} dt + \sum_{j=1}^{m} \begin{bmatrix}
0 \\
c_j(t, X(t))
\end{bmatrix} dW_j(t).
\]

Buckwar and Winkler [6] analyse a general class of multistep schemes, proving mean-square convergence, and apply two-step schemes to SDEs with small noise. Stochastic Adams-type numerical schemes for SDEs with additive noise are considered in, for example, [19].

### 6.1.2 Implicit numerical schemes

Implicit schemes tend to have greater numerical stability than explicit schemes [38, §9.8 and Chapter 12]. In general the equation for \( X_{n+1} \) is nonlinear and it is not possible to solve it algebraically; a numerical method such as the Newton–Raphson method is required. See for example [74, Chapter 1,Chapter 4] for a description of the Newton–Raphson method and other methods for solving nonlinear equations.

Some implicit schemes are essentially modified forms of Itô–Taylor and Stratonovich–Taylor explicit schemes - for example, the implicit Euler-type scheme (5.33), the implicit Euler–Heun scheme (5.36) and implicit Milstein schemes (5.38) and (5.41). These all have the same strong orders as their explicit counterparts. Kloeden and Platen propose implicit Itô–Taylor and Stratonovich–Taylor schemes of strong orders 1.5 and 2.0 in [38, §12.2], [37, §4], and implicit schemes for solving SDEs with additive noise are discussed in [54].

Implicit Runge–Kutta schemes and multistep schemes are also available. Kloeden and Platen propose implicit Runge–Kutta schemes of strong orders 1.5 and 2.0 in [38, §12.4], [37, §5]. Milstein and Tretyakov [56, 58, Chapter 3] were the first to consider specialist methods for solving SDEs with small noise. Such an SDE has the form
\[
dX(t) = f(X(t)) + \epsilon \sum_{j=1}^{m} g_j(X(t)) dW_j(t), \quad X(0) = x_0, \quad 0 \leq \epsilon \leq \epsilon_0,
\]
where \( \epsilon_0 \) is small and positive. The backward differentiation formula

\[
X_{n+1} = \frac{4}{3}X_n - \frac{1}{3}X_{n-1} + \frac{2}{3}f(X_{n+1})\Delta t_n \\
+ \sum_{j=1}^{m} \left[ g_j(X_n)\Delta W^j_n - \frac{1}{3}g_j(X_{n-1})\Delta W^j_{n-1} \right],
\]

(6.3)
is also applicable for solving SDEs with small noise - see for example [6,28].

It is difficult to introduce implicitness in the stochastic part of a numerical scheme, because unless sufficient care is taken the moments of the resulting approximation can become infinite - see [38, pp. 336–337], [37] for an example using a fully implicit Euler-type scheme. However it is possible to construct fully implicit numerical schemes - see for example [58, §1.3].

### 6.1.3 Balanced schemes

Milstein, Platen and Schurz [55] developed a scheme which has strong order 0.5, like the Euler–Maruyama scheme, and has better numerical stability. The so-called balanced scheme has the form

\[
X_{n+1} = \text{EM}_n(X_n) + C_n(X_n - X_{n+1}),
\]

(6.5)

where

\[
C_n = \epsilon_0(X_n)\Delta t_n + \sum_{j=1}^{m} c_j(X_n)\Delta W^j_n
\]

(6.6)

for functions \( c_j : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d} \) satisfying certain conditions (e.g. \( c_j \) being positive definite and having uniformly bounded components for all \( j = 0, \ldots, m \) is sufficient). We shall refer to this scheme as the balanced Euler–Maruyama scheme. Burrage, Herdiana and Burrage create a balanced Milstein scheme in [9], which is also fully implicit and has strong order 1.

### 6.2 Variable stepsize methods

There have been several attempts to combine a numerical scheme with the use of variable stepsizes to produce a numerical method to solve an SDE. We shall summarise
CHAPTER 6. EXISTING VARIABLE STEPSIZE METHODS

128

some variable stepsize methods which have been proposed, focussing in more detail on numerical methods developed by Gaines and Lyons in \[26\] and Lamba in \[43\] in more detail.

Firstly we shall examine an example from \[26\] which shows that we must be careful when forming a variable stepsize method to solve an SDE since convergence to the correct solution is not always guaranteed.

6.2.1 Example to show non-convergence of the Euler–Maruyama scheme with a variable stepsize method

Suppose that we solve
\[
dX(t) = W(t) \, dW(t), \quad X(0) = 0,
\]
(6.7)
over the interval \([0, 1]\) using the Euler–Maruyama scheme \((5.32)\) with two possible step sizes \(\Delta t\) and \(\frac{\Delta t}{2}\), where \(\Delta t = \frac{1}{N}\) and \(N = 2^M\) for some \(M \in \mathbb{N}\). Now, for \(n \geq 0\), let \(X_{2n}\) denote the numerical approximation of \(X(n\Delta t)\) for \(0 \leq n \leq 2^M\) (approximations \(X_{2n+1}\) to \(X((2n+1)\frac{\Delta t}{2})\) only being produced as necessary for this algorithm). To obtain \(X_{2n+2}\), we firstly simulate the Brownian increment \(W_{2n+2} - W_{2n}\) (\(W_n\) being used to denote \(W\left(n\frac{\Delta t}{2}\right)\) for any \(n \geq 0\)) and, for some predetermined choice \(\lambda \geq 0\), we check whether
\[
|W_{2n+2} - W_{2n}| \leq \lambda \sqrt{\Delta t}.
\]
(6.8)
If so, we use one step of the Euler–Maruyama scheme with stepsize \(\Delta t\) to obtain \(X_{2n+2}\), which for our problem reduces to
\[
X_{2n+2} = X_{2n} + W_{2n}(W_{2n+2} - W_{2n}),
\]
and we do not produce an approximation \(X_{2n+1}\). If \((6.8)\) is not satisfied, we instead use two steps of the Euler–Maruyama scheme with stepsize \(\frac{\Delta t}{2}\) to obtain \(X_{2n+2}\). Firstly we simulate \(W_{2n+1} = W(t_{2n+1})\) using the Browian bridge, and then
\[
X_{2n+1} = X_{2n} + W_{2n}(W_{2n+1} - W_{2n}),
\]
\[
X_{2n+2} = X_{2n+1} + W_{2n+1}(W_{2n+2} - W_{2n+1}).
\]

Now let \(D \in \mathcal{D}([0, T])\) be the resulting partition used, where the size of \(D\) is \(K\) for some \(N \leq K \leq 2N\). By applying the Itô formula \((5.9)\) with \(d = m = 1\), \(u(t, x) = x^2\),
\[ f(x) = 0 \text{ and } g(x) = 1, \]
we obtain
\[ W(t)^2 = t + \int_0^t 2W(s) \, dW(s), \]
and hence the exact solution of (6.7) is
\[ X(t) = \frac{1}{2} (W(t)^2 - t). \tag{6.9} \]

Now let the index \( k \) refer to our new partition \( D \), letting \( X_k \) be the approximation to \( X(t_k) \), \( W_k = W(t_k) \) and \( \Delta W_k = W_{k+1} - W_k \). The error at the final time \( t = 1 \) is
\[ X(1) - X_K = \frac{1}{2} (W(1)^2 - 1) - \sum_{k=0}^{K-1} W_k \Delta W_{k+1} \]
\[ = -\frac{1}{2} + \sum_{k=0}^{K-1} \left[ \frac{W_{k+1}^2}{2} - \frac{W_k^2}{2} - W_k(W_{k+1} - W_k) \right] \]
\[ = \frac{1}{2} \left[ -1 + \sum_{k=0}^{K-1} \Delta W_k^2 \right], \tag{6.10} \]

since \( W(1)^2 = \sum_{k=0}^{K} W_{k+1}^2 - W_k^2 \). We now split the sum into two parts: the part in the original partition \( D_{\Delta t} \) for which \( |W_{2n+2} - W_{2n}| \leq \lambda \sqrt{\Delta t} \) and the part for which \( |W_{2n+2} - W_{2n}| > \lambda \sqrt{\Delta t} \). For the second part, we note that
\[
(W_{2n+1} - W_{2n})^2 + (W_{2n+2} - W_{2n+1})^2 \\
= (W_{2n+1} - W_{2n} + W_{2n+2} - W_{2n+1})^2 - 2(W_{2n+1} - W_{2n})(W_{2n+2} - W_{2n+1}) \\
= (W_{2n+2} - W_{2n})^2 - 2(W_{2n+1} - W_{2n})(W_{2n+2} - W_{2n+1}).
\]

Hence we can rewrite (6.10) as
\[
X(1) - X_K = \frac{1}{2} \left[ -1 + \sum_{|W_{2n+2} - W_{2n}| < \lambda \sqrt{\Delta t}} (W_{2n+2} - W_{2n})^2 \\
+ \sum_{|W_{2n+2} - W_{2n}| \geq \lambda \sqrt{\Delta t}} \left[ (W_{2n+2} - W_{2n})^2 - 2(W_{2n+1} - W_{2n})(W_{2n+2} - W_{2n+1}) \right] \right] \\
= \frac{1}{2} \left[ -1 + \sum_{n=0}^{N-1} (W_{2n+2} - W_{2n})^2 \\
- \sum_{|\Delta W_n| \geq \lambda \sqrt{\Delta t}} 2(W_{2n+1} - W_{2n})(W_{2n+2} - W_{2n+1}) \right].
\]
Now we take the expectation on both sides. We know that \( \mathbb{E}[(W_{2n+2} - W_{2n})^2] = \Delta t \) and so \( \mathbb{E} \left[ \sum_{n=0}^{N-1} (W_{2n+2} - W_{2n})^2 \right] = N \Delta t = 1 \), hence
\[
\mathbb{E}[X(1) - X_K] = -\mathbb{E} \left[ \sum_{n=0}^{N-1} (W_{2n+1} - W_{2n})(W_{2n+2} - W_{2n+1}) \right]. \tag{6.11}
\]

For any \( n \) such that \( |W_{2n+2} - W_{2n}| \geq \lambda \sqrt{\Delta t} \), let \( Z = W_{2n+2} - W_{2n}, U = W_{2n+1} - W_{2n} \)
and \( V = W_{2n+2} - W_{2n+1} \). Then \( Z = U + V \) and both \( U \) and \( V \) are obtained using a Brownian bridge process, and are conditional on \( Z \). In particular the distribution of \( W_{2n+1} \) conditioned on \( Z \) is
\[
W_{2n+1} \sim N \left( W_n + \frac{Z}{2}, \frac{\Delta t}{4} \right),
\]
and hence the distribution of \( U \), conditioned on \( Z \), is
\[
U \sim N \left( \frac{Z}{2}, \frac{\Delta t}{4} \right). \tag{6.12}
\]

Note that the choice of \( n \) does not affect the distributions of \( U \) and \( V \) (as long as \( n \) is such that \( |Z| \geq \lambda \sqrt{\Delta t} \)).

Now (6.11), with \( a = \lambda \sqrt{\Delta t} \), becomes
\[
\mathbb{E}[X(1) - X_K] = -\mathbb{E} \left[ \sum_{|W_{2n+2} - W_{2n}| \geq a} UV \right]
= -N \mathbb{E} \left[ 1_{|Z| \geq a} U(Z - U) \right]
= N \mathbb{E} \left[ 1_{|Z| \geq a} U^2 - ZU \right]
= N \mathbb{E} \left[ \mathbb{E} \left[ 1_{|Z| \geq a} U^2 - ZU \mid Z \right] \right]
= N \mathbb{E} \left[ 1_{|Z| \geq a} \mathbb{E} \left[ U^2 - ZU \mid Z \right] \right] \quad \text{since} \quad 1_{|Z| \geq a} \text{ is } Z\text{-measurable}
= N \mathbb{E} \left[ 1_{|Z| \geq a} \left( \mathbb{E} \left[ \left( U - \frac{Z}{2} \right)^2 \mid Z \right] - \mathbb{E} \left[ \frac{Z^2}{4} \mid Z \right] \right) \right]
= N \mathbb{E} \left[ 1_{|Z| \geq a} \left( \text{Var} \left[ U \mid Z \right] - \frac{Z^2}{4} \right) \right]
= N \mathbb{E} \left[ 1_{|Z| \geq a} (\Delta t - Z^2) \right].
\]

Since \( \mathbb{E}[f(Z)] = \int_{-\infty}^{\infty} f(z)p_Z(z) \, dz \), where
\[
p_Z(z) = \frac{e^{-\frac{z^2}{2\Delta t}}}{\sqrt{2\pi\Delta t}}
\]
is the pdf of the random variable $Z$, we have
\[
\mathbb{E}[1_{|Z|\geq a}g(Z)] = \int_{-\infty}^{-a} g(z)p_Z(z) \, dz + \int_{a}^{\infty} g(z)p_Z(z) \, dz,
\]
and hence
\[
\mathbb{E}[X(1) - X_K] = \frac{N}{4} \left( \int_{-\infty}^{-a} (\Delta t - z^2) \frac{e^{-\frac{z^2}{2\Delta t}}}{\sqrt{2\pi \Delta t}} \, dz + \int_{a}^{\infty} (\Delta t - z^2) \frac{e^{-\frac{z^2}{2\Delta t}}}{\sqrt{2\pi \Delta t}} \, dz \right)
\]
\[
= \frac{N}{4\sqrt{2\pi \Delta t}} \left( \left[ \Delta tz e^{-\frac{z^2}{2\Delta t}} \right]_{-\infty}^{-a} + \left[ \Delta tz e^{-\frac{z^2}{2\Delta t}} \right]_{a}^{\infty} \right)
\]
\[
= -\frac{N\sqrt{\Delta t}}{2\sqrt{2\pi}} e^{-\frac{a^2}{2\Delta t}} = -\frac{\lambda\Delta t}{2\sqrt{2\pi}} e^{-\frac{\lambda^2}{2}},
\]
since $\frac{d}{dz} \left( ze^{-\frac{z^2}{2\Delta t}} \right) = \left( 1 - \frac{z^2}{\Delta t} \right) e^{-\frac{z^2}{2\Delta t}}$ and $N\Delta t = 1$. Hence, for any $\lambda \in (0, \infty)$, we expect the solution obtained using the variable stepsize method to be incorrect, whereas using $\lambda = 0$ or $\lambda = \infty$ (corresponding to using a fixed stepsize of $\Delta t_2$ or $\Delta t$ respectively), $\mathbb{E}[X(1) - X_K] = 0$ and so we expect convergence to the solution.

### 6.2.2 Variable stepsize methods based on the Euler–Maruyama and Milstein schemes

Many variable stepsize methods are based on the Euler–Maruyama scheme or explicit Milstein scheme, since they are the simplest schemes to implement. We now discuss some of these.

**Method proposed by Cambams and Hu**

Cambams and Hu [11] divide the interval $[0, T]$ into $N$ intervals by setting the times $t_n$ such that
\[
\int_{t_n}^{t_{n+1}} p(s) \, ds = \frac{1}{N}
\]
for $n = 0, \ldots, N - 1$, where $p(t)$ is a continuous, positive, probability density function subject to
\[
\int_{0}^{T} p(t) \, dt = 1.
\]
The subject of their paper is to solve Itô SDEs of the form (5.2) for $m = 1$ and to find the limit of the quantity
\[
A_{\text{SDE},p} = \lim_{N \to \infty} N \mathbb{E} \left[ \left\| X(t) - \bar{X}(t) \right\|_2^2 \right] \quad (6.13)
\]
and the related quantity
\[
\tilde{A}_{\text{SDE},p} = \lim_{N \to \infty} N \int_0^T \mathbb{E} \left[ \| \dot{X}(t) - \ddot{X}(t) \|_2^2 \right] dt, \tag{6.14}
\]
where \( \ddot{X}(t_n) = X_n \) for \( n = 0, \ldots, N \) and, for \( t_n \leq t < t_{n+1} \),
\[
\dddot{X}(t) = \dddot{X}_n + f(X_n)(t - t_n) + g(X_n)(W(t) - W(t_n)).
\]

The quantity \( A_{\text{SDE},p} \) depends on both the SDE and the function \( p(t) \). They show it is possible to find an explicit form for the optimal function \( p^*(t) \), which minimises the limit \( A_{\text{SDE},p} \), if the SDE is linear, and from studying this case they deduce that \( p(t) \) should be large at times \( t \) near zero (to produce small stepsizes) and get smaller as \( t \) increases.

**Methods proposed by Hofmann, Müller–Gronbach and Ritter**

Hofmann, Müller–Gronbach and Ritter have produced several papers on adaptive stepsize methods for SDEs with time-dependent coefficients of the form (5.5), all aiming to reduce the global error in mean square or \( p \)-th mean. In [31], they study the one-dimensional case \( (m = 1) \), using the Euler–Maruyama scheme and choosing stepsizes of the form
\[
\Delta t_n = \max \left( T - t_n, \Delta t^2, \frac{\Delta t}{\| g(t_n, X_n) \|} \right), \tag{6.15}
\]
where some ‘standard’ (but not maximum) stepsize \( \Delta t \) is chosen. They prove that, in the additive noise case where \( g \) is a function of \( t \) only, their choice of stepsizes reduces the strong error in mean square by a factor of \( \| g \|_1 \| g \|_2 \) and the method is asymptotically optimal amongst all those that use only values of the Wiener process \( W(t) \) at discrete points. Restricting the maximum stepsize to \( \Delta t^2 \) ensures that the error is also controlled in the small noise case.

For multi-dimensional SDEs with additive noise, Hofmann, Müller–Gronbach and Ritter [32] proposed the stepsize choice
\[
\Delta t_n = \max \left( T - t_n, \Delta t^2, \frac{\Delta t}{g^2(t_n)} \right), \tag{6.16}
\]
where
\[
g^2(t_n) = \max_{i=1, \ldots, d} \left( \sum_{j=1}^m g_{ij}^2(t) \right)^{\frac{1}{2}},
\]
in order to reduce the error

$$
E \left[ \left\| X \right\|_\infty^p \right]^{\frac{1}{p}}, \quad \left\| X \right\|_\infty = \sup_{0 \leq t \leq T} \max_{i=1,...,d} |X_i(t) - \bar{X}_i(t)|.
$$

They also modified this choice of stepsize to reduce the strong error in $p$-th mean, as defined by (5.55). This replaced $\Delta_t g(t_n)^2$ with $\Delta_t \|g(t_n)\|_F$ in (6.16).

Another method proposed by Hofmann, Müller–Gronbach and Ritter in [33] was to use the Milstein scheme, with stepsizes determined by (6.15), to solve one-dimensional SDEs with time-dependent coefficients, aiming to reduce the strong error in mean square. In order to help analyse this numerical method, they also created numerical methods based on using the Milstein scheme for a fixed stepsize, and using smaller stepsizes and the Euler–Maruyama scheme to generate approximations at intermediate points, the spacing of which is determined by the value of $g(t)$ at the original fixed points. Müller-Gronbach [59] refines this numerical method further in order to reduce the strong error in $p$-th mean for general $p \geq 1$.

**Methods proposed by Lamba, Mattingly and Stuart**

Lamba, Mattingly and Stuart [44,45] have proposed a numerical method based on the Euler–Maruyama scheme to solve multi-dimensional SDEs with time-independent coefficients. Their method uses an error estimate based on the drift function $f(x)$ only and, for some maximum stepsize $\Delta t_{\text{max}}$, only allows stepsizes of size $2^{-k} \Delta t_{\text{max}}$ for $k \geq 0$. In [44], the error estimate is based on the difference between the deterministic Euler and Heun schemes to solve the ODE

$$
\frac{dX(t)}{dt} = f(X(t))
$$

as follows. Let $X_n^*$ and $\hat{X}_n$ be the results of taking an Euler and a Heun step of size $\Delta t_n$ respectively:

$$
X_n^* = X_n + f(X_n) \Delta t_n,
$$

$$
\hat{X}_n = X_n + \frac{1}{2} \left[ f(X_n) + f(X_n^*) \right] \Delta t_n.
$$

Now, for a user-defined tolerance $\tau$, the solution is advanced (i.e. $X_{n+1}$ is computed) using the Euler–Maruyama scheme if

$$
E(X_n, \Delta t_n) = \left\| \hat{X}_n - X_n^* \right\|_2 \leq \Delta t_n \tau.
$$

(6.18)
If (6.18) is not satisfied then the stepsize is halved. [44] also suggest alternative error criteria and use these in numerical experiments.

The method in [45] involves a slightly different choice of stepsize. Suppose that the stepsize for the previous step is \( \Delta t_{n-1} = 2^{k_{n-1}} \Delta t_{\text{max}} \) for some \( k_{n-1} \). Then the new stepsize is chosen to be \( 2^{k_n} \Delta t_{\text{max}} \), where

\[
k_n = \mathcal{G}(X_n, k_{n-1})
\]

\[
= \min\{k \geq k_{n-1} \mid \|f(X_n + 2^{-k} \Delta t_{\text{max}} f(X_n)) - f(X_n)\|_2 \leq \tau\}.
\]

Now the approximation \( X_{n+1} \) is produced using the Euler–Maruyama scheme.

Note that in both cases, simulating the Wiener increment \( \Delta W_n \) is not required until the stepsize has been chosen, which saves on computational cost. The stepsize is also not allowed to increase beyond a value of twice that used for the previous step.

**Method proposed by Newton**

Another numerical method for solving one-dimensional SDEs over the interval \([0, T]\), using either the Euler–Maruyama or Milstein scheme, was proposed by Newton [62]. He proposes using the first passage times of the Wiener process \( W(t) \) through the points \( k\Delta W_{\text{max}} \), for some choice of maximum Wiener increment \( \Delta W_{\text{max}} > 0 \) and \( k \in \mathbb{Z} \), to generate the stepsizes. Specifically,

\[
\Delta t_n = \inf\{t \geq 0 \mid |\Delta W_n| = \Delta W_{\text{max}}\},
\]

for all \( n \geq 0 \). It is proved that

\[
\lim_{\Delta W_{\text{max}} \to 0} t_N = T, \quad N = \left\lfloor \frac{T}{\Delta W_{\text{max}}^2} \right\rfloor
\]

almost surely, and that the Euler–Maruyama approximation \( X_n^{\text{EM}} \) to \( X(t_n) \) satisfies

\[
\limsup_{\Delta W_{\text{max}} \to 0} \sup_{0 \leq n \leq N} \mathbb{E}\left[\left\| \frac{1}{\Delta W_{\text{max}}} (X(t_n) - X_n^{\text{EM}}) \right\|^p\right]_2 < \infty,
\]

and the Milstein approximation \( X_n^{\text{Mil}} \) to \( X(t_n) \) satisfies

\[
\limsup_{\Delta W_{\text{max}} \to 0} \sup_{0 \leq n \leq N} \mathbb{E}\left[\left\| \frac{1}{\Delta W_{\text{max}}^2} (X(t_n) - X_n^{\text{Mil}}) \right\|^p\right]_2 < \infty.
\]
6.2.3 Variable stepsize methods based on other schemes

Some variable stepsize methods for solving SDEs are based on the Runge–Kutta scheme, including those of Mauthner \[51\], who solves SDEs of the form (5.7) for \( m = 1 \), and Burrage and Burrage \[8\], who solve Stratonovich SDEs of the form (5.7) for any \( m \geq 1 \). Both methods use a stochastic Runge–Kutta scheme of strong order 1 to advance the solution, along with a stochastic Runge–Kutta scheme of strong order 1.0 to estimate the error at every step as follows. For some user-defined tolerance vector \( \mathbf{\tau} \) with elements \( \tau_i \) representing the tolerance for the \( i \)-th component of the solution \( X(t) \), the estimated error for the \((n + 1)\)-th step is given by

\[ \epsilon = \left( \frac{1}{d} \sum_{i=1}^{d} \left( \frac{X_{n+1}^i - \hat{X}_{n+1}^i}{\tau_i} \right)^2 \right)^{\frac{1}{2}}, \]

where \( X_{n+1} \) and \( \hat{X}_{n+1} \) are the approximations produced by the order 1.5 and order 1.0 stochastic Runge–Kutta schemes respectively (in \[51\], where \( d = m = 1, \epsilon = \frac{X_{n+1} - \hat{X}_{n+1}}{\tau} \)). The solution is advanced only if \( \epsilon \leq 1 \), and a new stepsize is chosen to be as close to

\[ \Delta t_{\text{opt}} = \theta \Delta t \frac{1}{\epsilon^{\frac{1}{p}}} \]

as possible, in order to try and get \( \epsilon \) as close to 1 as possible in the next step. Mauthner sets \( p = 1.5 \), while Burrage and Burrage set \( p = 2 \). The parameter \( \theta \in (0, 1] \) is a safety factor which prevents too many stepsize rejections.

The stochastic Runge–Kutta scheme of strong order 1.5 requires approximations of the integrals \( J_{1,0}^n \). Because of the complexity involved in calculating these integrals for general stepsizes, Mauthner suggests only allowing halving and doubling of the stepsize; however Burrage and Burrage allow any stepsize to be used.

An alternative method is also proposed in \[9\], which uses the balanced Milstein scheme to advance the solution, and the balanced Euler–Maruyama scheme to approximate the error.

6.3 Two examples of variable stepsize methods

We describe two variable stepsize methods in more detail. One is proposed by Gaines and Lyons in \[26\], and the other is proposed by Lamba in \[43\]. We also briefly discuss
an improvement Lamba made to his method in [42], which involves the usage of first exit times of the Wiener process.

6.3.1 Method proposed by Gaines and Lyons

Gaines and Lyons [26] solve Stratonovich SDEs of the form

$$dX(t) = \sum_{j=0}^{m} g_j(X(t)) \circ dW_j(t), \quad X(0) = x_0,$$

where we use the convention $t = W_0(t)$, driven by an $m$-dimensional $\mathcal{F}_t$-Wiener process for $m \geq 1$ where the solution $X(t) \in \mathbb{R}^d$ for $d \geq 1$. The primary aim of their numerical method is to minimize the pathwise global error, using the Milstein scheme. This is achieved by studying the propagation of the error through time for a predetermined final value of the Wiener process, and making each step of the numerical method contribute as equally as possible to the final error. To study the error propagation it is usually necessary to derive another SDE to represent the error propagation and solve it using the Milstein scheme with a fixed stepsize, although in some cases the solution of this SDE is known explicitly.

The Milstein schemes used for Itô and Stratonovich SDEs are given by (5.37) and (5.40) respectively. In particular, for non-commutative SDEs, (5.42) is used to express the last term in (5.37) and (5.40) in terms of the Lévy area. A maximum stepsize of $\Delta t_{\text{max}}$ is chosen and only stepsizes of size $\Delta t_{\text{max}}^{2\ell}$ for integers $\ell \geq 0$ are allowed, for integers $k \geq 0$, so that the Wiener increments are stored in a so-called ‘Brownian tree’. This allows for a simple approximation of the Lévy area.

In what follows, for a partition $D \in \mathcal{D}([0,T])$, $\mathcal{P}_D$ denotes the partition $\sigma$-algebra, the $\sigma$-algebra generated by values of the Wiener process $W(t)$ at points $t \in D$. The error estimate at each step is based on the $\mathcal{P}_D$-conditional variance of the local truncation error. Any scheme for which this is less than the square of the $\mathcal{P}_D$-conditional mean of the local truncation error can be used.

**Generation of the Wiener process**

Once we have chosen a maximum stepsize $\Delta t_{\text{max}}$, which for simplicity we assume to be $\frac{T}{K}$ for some integer $K$, we allow only stepsizes of size $\Delta t_{\text{max}}^{2\ell}$ for integers $\ell \geq 0$ - only
halving and doubling of the stepsize is allowed. This allows us to store the Wiener increments in a tree structure, which we call a Brownian tree. Level 0 of the tree consists of $K$ Wiener increments $\Delta W_{k,0} = W(k\Delta t_{\text{max}}) - W((k - 1)\Delta t_{\text{max}})$, each of which has distribution $N(0, \Delta t_{\text{max}})$, for $k = 1, \ldots, K$. We can simulate these in order, or we can choose to simulate $W(T)$ first and use the Brownian bridge to simulate the increments. For $\ell \geq 1$, level $\ell$ of the tree consists of Wiener increments

$$
\Delta W_{k,\ell} = W \left( k \frac{\Delta t_{\text{max}}}{2^\ell} \right) - W \left( (k - 1) \frac{\Delta t_{\text{max}}}{2^\ell} \right),
$$

over time intervals $[(k - 1)\Delta t_{\text{max}}2^\ell, k\Delta t_{\text{max}}2^\ell]$, for $k = 1, \ldots, K2^\ell$; however we do not need to complete any level of the tree apart from level 0. Figure 6.1 shows an example of the Brownian tree, for which levels 0 and 1 have been fully completed, but only parts of levels 2 to 4 have been generated.

![Brownian Tree Diagram](image)

**Figure 6.1:** An illustration of the Brownian tree, for $K = 4$.

Each Wiener increment $\Delta W_{k,\ell}$ at level $\ell$ of the tree can be used to generate the Wiener increments $\Delta W_{2k - 1,\ell + 1}$ and $\Delta W_{2k,\ell + 1}$ at level $\ell + 1$ of the tree, by setting

$$
\Delta W_{2k - 1,\ell + 1} = \frac{1}{2} \Delta W_{k,\ell} + y_{k,\ell}, \\
\Delta W_{2k,\ell + 1} = \frac{1}{2} \Delta W_{k,\ell} - y_{k,\ell},
$$

(6.20) (6.21)
where the elements of $y_{k,\ell}$ are iid and follow the $N\left(0, \frac{\Delta t_{\text{max}}}{2^{\ell+1}}\right)$ distribution. This is known as Lévy’s construction (see for example [35, pp. 56–59]) and is equivalent to using the Brownian bridge, as described in §2.2.5. Figure 6.2 illustrates how the increments $\Delta W_{2k-1,\ell+1}$ and $\Delta W_{2k,\ell+1}$ are generated.

![Figure 6.2: An illustration to show how each element of the Wiener increments $\Delta W_{2k-1,\ell+1}$ and $\Delta W_{2k,\ell+1}$ are generated from $\Delta W_{k,\ell}$, for $s = (k-1)\Delta t_{\text{max}}, t = k\Delta t_{\text{max}}$.](image)

Once level 0 of the tree is created, further increments can be generated as required by the variable stepsize algorithm. The Brownian tree can then be stored and reused in another simulation - for example with a smaller tolerance for the error - and expanded if necessary.

**Approximating the Lévy areas**

If $m > 1$ and the SDE we are solving is non-commutative, we will need approximations $\tilde{A}_{jr}(s, t)$ to the Lévy areas

$$A_{jr}(s, t) = \frac{1}{2} \left( J_{(j,r),[s,t]} - J_{(r,j),[s,t]} \right)$$

in order to use the Milstein scheme. Suppose we need to calculate the approximate Lévy area $\tilde{A}_{jr}(s, t)$ for $s = (k-1)\Delta t_{\text{max}}, t = k\Delta t_{\text{max}}$. Firstly we must use the Wiener increment $\Delta W_{k,\ell}$ to generate all possible ‘descendant’ increments down to level $\ell + \ell'$ of the Brownian tree, for some $\ell' \geq 1$, i.e. all the increments $\Delta W_{2qj-p,k+q}$, for all $q = 1, \ldots, \ell'$ and all $p = 0, \ldots, 2^q - 1$. Now let $\tilde{A}$ be the so-called ‘Lévy area matrix’
CHAPTER 6. EXISTING VARIABLE STEPSIZE METHODS

with entries $\hat{A}_{jr}$. We now set

$$\hat{A} = \frac{1}{2} \sum_{q=1}^{2^\ell-1} \sum_{p=1}^{2^\ell} \left[ \Delta W_{2^{q}(k-1)+2p-1,\ell+q} \Delta W_{2^{q}(k-1)+2p,\ell+q} \right], \quad (6.22)$$

where $[a, b] = a \otimes b^T - b \otimes a^T$. For example, if $k = 1$, $\ell = 0$ and $\ell' = 3$,

$$\hat{A} = \frac{1}{2} \left( [\Delta W_{1,1}, \Delta W_{2,1}] + [\Delta W_{1,2}, \Delta W_{2,2}] + [\Delta W_{3,2}, \Delta W_{4,2}] \right.$$

$$+ [\Delta W_{1,3}, \Delta W_{2,3}] + [\Delta W_{3,3}, \Delta W_{4,3}] + [\Delta W_{5,3}, \Delta W_{6,3}] + [\Delta W_{7,3}, \Delta W_{8,3}] \big).$$

It is possible to verify that (6.22) is equivalent to

$$\hat{A} = \frac{1}{2} \sum_{1 \leq p < q \leq 2^\ell'} \left[ \Delta W_{2^\ell'(k-1)+p,\ell'+q, \Delta W_{2^\ell'(k-1)+q,\ell'+p} \right], \quad (6.23)$$

but the expression (6.22) has two advantages. Firstly each term

$$\frac{1}{2} \left[ \Delta W_{2^{q}(k-1)+2p-1,\ell+q} \Delta W_{2^{q}(k-1)+2p,\ell+q} \right]$$

there can be stored alongside the ‘parent’ Wiener increment $\Delta W_{2^{q-1}(k-1)+p,\ell+q-1}$, and there are only $2^{\ell'} - 1$ matrices to add together instead of $(2^{\ell'} - 1)(2^\ell - 1)$ matrices.

**Studying error propagation**

In order to control the final global error, Gaines and Lyons approximate how the global error propagates along the time interval $[0, T]$, by assuming that it propagates linearly for small stepsizes, and assuming that the LTE at each step is random and independent of the LTE at previous steps. By replacing $X(t)$ by $X(t) + \epsilon Y(t)$ in (6.19), we obtain the Stratonovich SDE

$$dX(t) + \epsilon dY(t) = \sum_{j=0}^{m} \left[ g_j(X(t)) + \sum_{k=1}^{d} \frac{\partial g_j(X(t))}{\partial x_k} Y_k(t) \right] \circ dW_j(t) + O(\epsilon^2),$$

$$X(s) = x_s, \quad Y(s) = y_s,$$

over the interval $[s, T]$, where $\epsilon y_s$ represents a small error at time $s$. By taking the terms of order $\epsilon$ we obtain the Stratonovich SDE

$$dY(t) = \sum_{j=0}^{m} \left[ \sum_{k=1}^{d} \frac{\partial g_j(X(t))}{\partial x_k} Y_k(t) \right] \circ dW_j(t), \quad (6.24)$$

as $\epsilon \to 0$. 
Choosing the stepsizes

It is claimed that the optimal choice for reducing the global error is to let the local error for each stepsize contribute equally (or at least, as equally as possible) to the $\mathcal{P}_D$-conditional variance of the final global error. To this end, a value $C$ is chosen to be the desired contribution of the $\mathcal{P}_D$-conditional variance of the local error for each step to the global error. A fixed stepsize is now used to solve (6.19) to obtain an approximation $\tilde{X}(t)$ to the true solution $X(t)$. Next, $\tilde{X}(t)$ is substituted into (6.24), which is then solved backwards in time with ‘initial condition’ $y(T) = C$, in order to find out the desired error contribution $y(t, C)$ at times $t \in [0, T]$. To find the stepsize $\Delta t_n$, the conditional variance of the local error is estimated using a candidate stepsize, and the stepsize is accepted if this error estimate is less than $y(t_n, C)$. If the stepsize is not accepted it is halved successively until the error estimate becomes less than $y(t_n, C)$.

6.3.2 Method proposed by Lamba

Lamba [43] focuses on the solutions of Itô SDEs of the form (5.2) driven by a one-dimensional Wiener process, where the solution can be $\mathbb{R}^d$-valued for any $d \geq 1$ (though $d = 1$ in the test problems used). The Milstein scheme is used - since $m = 1$ no Lévy area approximation is required. The aim of Lamba’s numerical method is to estimate the LTE of the Milstein scheme and split it a stochastic part and a deterministic part, estimating both separately after calculating a Wiener increment for the next step - the stochastic error will usually depend on the Wiener increment and it is advantageous to calculate it before taking a step of the Milstein scheme. The stepsize is reduced if either (or both) errors are deemed to be too large, with a slightly different stepsize control applied depending on which is larger, and the Brownian bridge is used to determine the new value of the Wiener process. Similarly the stepsize can be increased if the error is deemed to be small enough. Unlike in Gaines and Lyons [26], there is no restriction on the stepsize.
CHAPTER 6. EXISTING VARIABLE STEPSIZE METHODS

Error estimation

To estimate the LTE $T_n^{\text{Mil}}$, the SDE (6.2) is converted into its Stratonovich form (5.7) for $\tilde{f}$ with components given by (5.8). We saw in (5.1.8) that the LTE is given by (5.53).

Writing this out in full, assuming $d = 1$ for notational simplicity, and abbreviating $Y_{t_n,X_n}(\cdot)$, the solution of (5.52), by $Y(\cdot)$,

$$T_n^{\text{Mil}} = J_{(0,0)}^n[\tilde{f}(Y(\cdot))\tilde{f}(Y(\cdot))] + J_{(1,0)}^n[g'(Y(\cdot))\tilde{f}(Y(\cdot))] + J_{(0,1)}^n[f'(Y(\cdot))g(Y(\cdot))]$$

$$+ J_{(1,1)}^n[(g''(Y(\cdot)) + (g'(Y(\cdot)))^2)f(Y(\cdot))] + O(\Delta t_n)$$

$$+ J_{(1,1)}^n[g''(Y(\cdot))(g(Y(\cdot)))^2 + (g'(Y(\cdot)))^2g(Y(\cdot))].$$

This is now approximated by leaving out the fourth term (since it has order $\Delta t_n^2$), and performing a further Stratonovich–Taylor expansion on the remaining terms about $t_n$, yielding

$$\tilde{T}_n^{\text{Mil}} = J_{(0,0)}^n\tilde{f}(X_n)\tilde{f}(X_n) + J_{(1,0)}^n[g'(X_n)\tilde{f}(X_n) + J_{(0,1)}^n(f'(X_n))g(X_n)$$

$$+ J_{(1,1)}^n[g''(X_n)(g(X_n))^2 + (g'(X_n))^2g(X_n)] + O(\Delta t_n^2)$$

$$= \frac{\Delta t_n^2}{2} \tilde{f}(X_n)\tilde{f}(X_n) + J_{(1,0)}^n[g'(X_n)\tilde{f}(X_n) + J_{(0,1)}^n(f'(X_n))g(X_n)$$

$$+ \frac{\Delta W_n^3}{6}g''(X_n)(g(X_n))^2 + \frac{\Delta W_n^3}{6}(g'(X_n))^2g(X_n) + O(\Delta t_n^2).$$

Since the integrals $J_{(1,0)}^n$ and $J_{(0,1)}^n$ cannot be computed from values of the Wiener process alone, and calculation of $g''(X_n)$ is also undesirable, we estimate the stochastic error $E_s(X_n, \Delta t_n)$ as

$$E_s(X_n, \Delta t_n) = \frac{\Delta W_n^3}{6}[(g'(X_n))^2g(X_n)],$$

and assume that controlling this term will control the stochastic part of the LTE acceptably. We also estimate the deterministic error $E_d(X_n, \Delta t_n)$ as

$$E_d(X_n, \Delta t_n) = \left| \frac{1}{2} (\tilde{f}(X_n + \Delta t \tilde{f}(X_n)) - \tilde{f}(X_n)) \right|,$$

since this is an estimate of $\frac{\Delta t^2}{2} \tilde{f}(X_n)\tilde{f}(X_n)$ obtained by taking the difference between the Euler and Euler-Heun schemes applied to solving

$$\frac{dX(t)}{dt} = \tilde{f}(X),$$

and it avoids calculating $\tilde{f}'(x)$. We can also replace $\tilde{f}$ by $f$ in (6.28) if, for example, $\tilde{f}'(x)\tilde{f}(x)$ is identically zero or values of $\tilde{f}(x)$ are significantly more difficult to compute than values of $f(x)$.
Choosing the stepsize and generating the Wiener process

Suppose that we have performed \( n \) steps of the approximation and produced an approximation \( X_n \) to the true solution \( X(t_n) \) of (5.2). We now need to choose the stepsize \( \Delta t_n = t_{n+1} - t_n \) in order to produce an approximation \( X_{n+1} \) to the true solution \( X(t_{n+1}) \). In order to advance the solution, we require that

\[
\max(E_s(X_n, \Delta t_n), E_d(X_n, \Delta t_n)) \leq \sigma(X_n, \tau),
\]

where:

- \( \tau \) is a user-defined tolerance;
- \( \sigma \) is a function which determines which type of error control we want to use. For example, we can choose \( \sigma(X_n, \tau) = \tau, \sigma(X_n, \tau) = |X_n|\tau \) or \( \sigma(X_n, \tau) = \tau \max(1, |X_n|) \) which correspond to using absolute, relative or mixed error control respectively.

We shall define some quantities which shall appear later. They are:

\[
\kappa_s = \frac{E_s}{\sigma}, \quad \kappa_d = \frac{E_d}{\sigma}, \quad \Delta W_{opt} = 0.9\kappa_d^{-\frac{3}{2}}|\Delta W|, \quad \alpha = \frac{|\Delta W|}{\sqrt{\Delta t}},
\]

where \( E_s, E_d, \sigma, \Delta W \) and \( \Delta t \) are the last values of these variables which have been calculated, either from a Milstein step or an error estimate resulting in a stepsize rejection, and the value of 0.9 in the definition of \( \Delta W_{opt} \) is a ‘safety factor’ to ensure we do not need to perform too many stepsize rejections. Our aim is to get \( \Delta W \) to be as close to, but smaller than, \( \Delta W_{opt} \) as possible, so that \( \kappa_s \) remains as close to 1 as possible without exceeding 1. The quantity \( \alpha \) is the number of standard deviations from the mean of \( \Delta W \); we consider \( \Delta W \) to be an ‘outlier’ if \( \alpha > 2 \).

We have two methods of choosing the stepsize, depending on which of \( E_s \) or \( E_d \) is bigger. If \( E_d \geq E_s \) then we choose a maximum candidate stepsize \( \Delta t' \) based on the ratio \( \kappa_d; h' \) is given by

\[
\Delta t' = \min(\Delta t_{max}, 1.5\Delta t, 0.8\Delta t\kappa_d^{-\frac{3}{2}}),
\]

where the 0.8 is another ‘safety factor’ to ensure there are not too many stepsize rejections. The maximum possible stepsize is 1.5 times the previous one - this ratio is chosen because \( |\Delta W| \), and hence \( E_s \), may be particularly small and we want to
make sure the new value of $E_s$ is not too big. We now choose a candidate stepsise $\Delta t_{\text{new}} = \frac{j \Delta t'}{3}$, for $j = 1, 2$ or $3$, by generating Wiener increments $\Delta W_j$ for the stepsizes $\frac{j \Delta t'}{3}$ and choosing $j$ such that $\Delta W_k \leq \Delta W_{\text{opt}}$ for all $k = 1, \ldots, j$ (or, if none of $\Delta W_j$ are less than $\Delta W_{\text{opt}}$, we choose $j = 1$). This allows us to reject a stepsise based solely on the Wiener increment $\Delta W$ without having to perform a Milstein step.

There are two differences to the procedure for choosing the stepsise when $E_s > E_d$. Firstly $\Delta t' = \Delta t$. Secondly the candidate stepsise $\Delta t_{\text{new}} = \frac{j \Delta t'}{3}$ is chosen as before, but this time $j = 1, \ldots, j_{\text{max}}$ where $j_{\text{max}}$ is as follows:

- If $\kappa_s > 1$ (i.e. the last stepsise was rejected), $j_{\text{max}} = 2$;
- If $\kappa_s \leq 1$ and $\alpha < 2$ (i.e. the last Wiener increment $\Delta W$ was not an outlier), then $j_{\text{max}} = 4$;
- If $\kappa_s \leq 1$ and $\alpha \geq 2$, then $j_{\text{max}} = 6$.

Thus we ensure that the stepsise decreases if the last stepsise was rejected, and we allow the stepsise to increase if the last stepsise was accepted, particularly if the last Wiener increment $\Delta W$ was an outlier.

**Refinement of the method using first exit times of the Wiener process**

Lamba refines his numerical method in [42]. Firstly he introduces a third error estimate

$$E_t(X_n, \Delta t) = |J_{[1,0]}^{n} f'(X_n) g(X_n) \Delta W_n|,$$

since for additive noise, $E_s(X_n, \Delta t) = 0$ and so the stochastic error control has no effect on the stepsises which are chosen. Secondly, attempts are made to control the global error by bounding the sum of the deterministic errors $E_d(X_n, \Delta t)$, and the standard deviations of the sums of the stochastic errors $E_s(X_n, \Delta t)$ and $E_t(X_n, \Delta t)$. These error estimations resulted in a bound for both the stepsise $\Delta t_n$ and the Wiener increment $\Delta W_n$ at each step. A bounded Wiener increment is now simulated, using a similar method to that described in Chapter [7] ensuring that both the stepsise and the Wiener increment satisfy their respective bounds.
6.3.3 A comparison of the two methods

Although both the method proposed by Gaines and Lyons [26], and that proposed by Lamba [43], both use the Milstein scheme, it is clear that the method of Gaines and Lyons is much more complex to implement, due to the analysis required in order to examine the error propagation. This is designed to reduce the global strong error, whereas the method of Lamba is only designed to reduce the LTE at each step and not the final global error directly - and Lamba also reduces computational cost by only considering part of the LTE, and ignoring terms which involve multiple stochastic integrals.

The method of Gaines and Lyons is also more complex because it deals with multi-dimensional, non-commutative SDEs and so approximation of Lévy areas is required, whereas Lamba only deals with one-dimensional SDEs. Gaines and Lyons only allow stepsizes of size $\Delta t_{\text{max}}^{\frac{1}{2}}$ to be used, which provides a relatively simple method of approximating the Lévy areas using the Brownian tree. Since Lamba only studies one-dimensional SDEs and therefore does not need to approximate Lévy areas, his method allows any stepsize to be used.

Some of the other methods discussed earlier - for example, for the method of Cambams and Hu [11], and those of Hofmann, Müller–Gronbach and Ritter [31–33], do not allow a stepsize to be rejected, and neither does the method Lamba introduces in [42]. However, for both the method of Gaines and Lyons and Lamba’s original method, stepsizes can be rejected - although both methods are designed to minimise the number of stepsize rejections.

A note on the BIEC method

In Chapter 8 we shall introduce a method called the Bounded Increments with Error Control method, or BIEC method, which can be used to solve multi-dimensional SDEs. This method uses the Euler–Maruyama scheme with error controls, based on both the drift and diffusion, to bound both the stepsize and the Wiener increment at each step - the Wiener increment can be bounded using the procedure described in Chapter 7. Stepsize rejections are not allowed.

Many of the methods reviewed in this chapter control the error produced by the drift function or the diffusion functions, but not both at once. Additionally, many
of these methods do not have an effective way of controlling the Wiener increment. By using an error control for both the drift and diffusion, we can effectively control the error for both drift- and diffusion-dominated SDEs, and by using bounded Wiener increments, we can avoid errors resulting from unusually large Wiener increments. The fact that stepsize rejections are not allowed is not considered to be a disadvantage.
Chapter 7

Simulating a Wiener Process using Bounded Wiener Increments

There are various ways of simulating an $m$-dimensional Wiener process $W(t)$ over an interval $[0, T]$, and the choice of method is often determined by the choice of method used to solve an SDE driven by $W(t)$. We have already seen different ways of simulating $W(t)$, for example simulating each increment in turn or using the Brownian bridge as described in §2.2.5 or using a ‘Brownian tree’ as described in §6.3.1. It is also possible to simulate $W(t)$ using bounded Wiener increments. As we shall see in this chapter, given an open domain $G \in \mathbb{R}^m$ containing $0_m$ it is possible to form a partition $D \in D_{\Delta t_{\text{max}}}([0, T])$ for some $\Delta t_{\text{max}}$ which bounds the increments $\Delta W_n$ so that either $\Delta t_n$ is the first exit time of $\Delta W_n$ from $G$, or $\Delta t_n = \Delta t_{\text{max}}$ and $\Delta W_n$ does not exit $G$ in the time interval $[0, \Delta t_{\text{max}}]$ (see §5.1.7 for the definition of $D_{\Delta t_{\text{max}}}([0, T])$). The resulting times $t_n \in D$ are all stopping times. We can also choose to change the maximum stepsize and domain $G$ at each step. This forms the basis of the Bounded Increments with Error Control algorithm described in §8.2.

We derive and solve PDE problems which must be solved in order to simulate the bounded Wiener increments, for both one- and multi-dimensional Wiener processes, and finally we state an algorithm to simulate the bounded Wiener increments. The PDE problems and their solutions are all found in [57, 58], and the algorithms for simulating bounded increments in the one-dimensional case are also found there.
7.1 Deriving the PDE to solve for bounded diffusion

7.1.1 Background

Let \( L \) be a semi-elliptic partial differential operator of the form

\[
L = \sum_{i=1}^{m} f_i(x) \frac{\partial}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij}(x) \frac{\partial^2}{\partial x_i \partial x_j},
\]

(7.1)

where \( a_{ij}(x) = a_{ji}(x) \) and \( f_i(x) \) are bounded and Lipschitz continuous functions, for all \( i, j = 1, \ldots, m \). \( L \) is a semi-elliptic, respectively elliptic, operator if all of the eigenvalues of the matrix \( a(x) = [a_{ij}(x)]_{i,j=1}^{m} \) are non-negative, respectively positive.

Now we consider the Cauchy problem

\[
\frac{\partial w}{\partial t} = Lw, \quad t > 0, \quad x \in \mathbb{R}^m, \quad w(0, x) = h(x), \quad x \in \mathbb{R}^m.
\]

(7.2) (7.3)

We claim that if we can find an Itô diffusion \( X(t) \) whose generator \( A \) coincides with \( L \) on \( C^2_0(\mathbb{R}^m) \), then we have a probabilistic representation for the solution \( w \) of the Cauchy problem (7.2)-(7.3). Let \( X(t) \) be the solution of the time-homogeneous SDE

\[
dX(t) = f(X(t)) \, dt + g(X(t)) \, dW(t), \quad X(0) = x,
\]

(7.4)

where \( g(x) \) is a ‘square root’ of \( 2a(x) \); that is, \( \frac{1}{2} g(x) g(x)^T = a(x) \). Then:

**Theorem 7.1** If \( w(t, x) \in C^{1,2}(\mathbb{R}_+ \times \mathbb{R}^m, \mathbb{R}) \) is a solution of the Cauchy problem (7.2)-(7.3), then we can write \( w(t, x) \) as

\[
w(t, x) = \mathbb{E}[h(X(t))].
\]

(7.5)

**Proof** See [23, Chapter 2, Theorem 1.1] (with \( c(x) \equiv 0 \) there). \( \square \)

The equation (7.2) is known as Kolmogorov’s backward equation, and the representation of the solution \( u(t, x) \) in (7.5) is known as the Feynman–Kac formula.

Next we let \( G \subset \mathbb{R}^m \) be a bounded, open domain with boundary \( \partial G \), and we consider the semi-elliptic Dirichlet boundary value problem (BVP)

\[
Lw = 0, \quad \text{in } G,
\]

(7.6)

\[
w(x) = \phi(x), \quad \forall x \in \partial G.
\]

(7.7)
Now for the Itô diffusion $X(t)$ given by (7.4) with $X(0) = x$, let $\tau$ be the first exit time of $X(t)$ from $G$; that is, $\tau = \inf\{t : X(t) \in \partial G\}$. Then:

**Theorem 7.2** If $w(x) \in C^2(G)$ is a solution of \((7.6) \sim (7.7)\), then it can be written as

$$w(x) = \mathbb{E}[\phi(X(\tau))].$$

(7.8)

**Proof** See [23, Chapter 2, Theorem 2.1, Remark 4]. We can guarantee that the exit time $\tau$ is finite with probability one; however there are other cases considered in [23] for which this is not the case. \hfill \square

In particular, if we set

$$\phi(x) = 1_{x \in \partial G_0},$$

(7.9)

where $1_E$ is the indicator function for an event $E$, then

$$w(x) = \mathbb{E}[1_{X(\tau) \in \partial G_0}] = \mathbb{P}(X(\tau) \in \partial G_0),$$

(7.10)

so if we can find solve \((7.6) \sim (7.7)\) for this particular $\phi$, the solution gives us the probability that $X(t)$ will exit $G$ at a point in $\partial G_0$. Figure 7.1 illustrates the possible cases for the exit points of the one-dimensional Itô diffusion $X(t)$, for the case $m = 1$, $X(0) = 0$ and $G = (-r, r)$. The boundary $\partial G$ in this case is $\partial G = \{-r, r\}$. We can use the solution $w(0)$ with boundary condition given by either $\phi(x) = 1_{x=r}$ or $\phi(x) = 1_{x=-r}$ to express the probability of $X(\tau)$ taking a value of either $r$ or $-r$ respectively.

Now we consider the parabolic mixed problem (with both initial and boundary conditions)

$$\frac{\partial w}{\partial t} = Lw \quad t > 0, \ x \in G,$$

(7.11)

$$w(0, x) = h_1(x), \quad x \in G \cup \partial G,$$

(7.12)

$$w(t, x) = h_2(t, x), \quad t > 0, \ x \in \partial G,$$

(7.13)

where $h_2(0, x) = h_1(x)$ for $x \in \partial G$. As before, for the Itô diffusion $X(t)$ given by (7.4) with $X(0) = x$, let $\tau$ be the first exit time of $X(t)$ from $G$. Then:

**Theorem 7.3** If $w(t, x) \in C^{1,2}(\mathbb{R}_+ \times \mathbb{R}^m, \mathbb{R})$ is a solution of \((7.11) \sim (7.13)\), then we can write $w(t, x)$ as

$$w(t, x) = \mathbb{E}[h_1(X(t))1_{\tau \geq t}] + \mathbb{E}[h_2(\tau, X(\tau))1_{\tau < t}],$$

(7.14)

where $1_E$ is the indicator function for an event $E$. 


CHAPTER 7. SIMULATING A WIENER PROCESS

Proof See [23, Chapter 2, Theorem 2.3]. □

Setting the initial and boundary conditions

We look at particular cases for the initial boundary conditions. The first case is

\[ h_1(x) = 0, \quad h_2(t, x) = 1_{x > 0} 1_{x \in \partial G_0}, \]  

(7.15)

where \( \partial G_0 \) is a subset of the boundary \( \partial G \). From (7.14), this would mean that the solution \( w(t, x) \) of (7.11)-(7.13) can be written as

\[ w(t, x) = \mathbb{E}[1_{\tau < t} 1_{x > 0} 1_{X(\tau) \in \partial G_0}] = \mathbb{P}(0 < \tau < t, X(\tau) \in \partial G_0). \]  

(7.16)

Hence \( w(t, x) \) is simply the probability that the Itô diffusion \( X(t) \) exits the domain \( G \) somewhere in \( \partial G_0 \), at a time \( \tau < t \). The second case is

\[ h_1(x) = 1_{x \in G_0}, \quad h_2(t, x) = 0, \]  

(7.17)

where \( G_0 \) is a subset of \( G \). From (7.14), the solution \( w(t, x) \) of (7.11)-(7.13) can be written as

\[ w(t, x) = \mathbb{E}[1_{x \in G_0} 1_{\tau \geq t}] = \mathbb{P}(\tau \geq t, X(t) \in G_0). \]  

(7.18)

This is the probability that \( X(t) \) does not exit the domain \( G \) at a time \( \tau < t \), and its value at time \( t \) lies in the set \( G_0 \).

Figure 7.2 illustrates the possibilities for the behaviour of the one-dimensional Itô diffusion \( X(t) \) for the case \( m = 1, X(0) = 0 \), and \( G = (-r, r) \) (again \( \partial G = \{-r, r\} \)).

For a time \( T > 0 \), \( X(t) \) can either exit \( G \) at a time \( \tau < T \), where either \( X(\tau) = r \) or \( X(\tau) = -r \); the probabilities of this happening can be expressed by the solution \( w(T, 0) \) with initial and boundary conditions given by (7.15), with \( \partial G_0 = \{r\} \) or \( \{-r\} \) respectively. Alternatively \( X(t) \) can fail to exit \( G \) at a time \( \tau < T \), and we can use the solution \( w(T, 0) \) with initial and boundary conditions given by (7.17) to express the probability of \( X(\tau) \) having a final value in the set \( G_0 \).
CHAPTER 7. SIMULATING A WIENER PROCESS

Figure 7.1: The possible cases for the exit points of $X(t)$ from $G$, with $m = 1$, $X(0) = 0$ and $G = (-r, r)$.

Figure 7.2: The possible cases for the behaviour of $X(t)$, with $m = 1$, $X(0) = 0$ and $G = (-r, r)$.
CHAPTER 7. SIMULATING A WIENER PROCESS

The final PDE problems to solve

As a consequence of the above, we have two PDE problems to solve. The first uses the initial and boundary conditions in (7.15), with \( \partial G_0 = \partial G \), and is

\[
\frac{\partial w}{\partial t} = Lw, \quad t > 0, \quad x \in G,
\]

\[
w(0, x) = 0, \quad x \in G \cup \partial G,
\]

\[
w(t, x) = 1, \quad t > 0, \quad x \in \partial G.
\]

The solution has a probabilistic representation as

\[
w(x, t) = \mathbb{P} (\tau < t).
\]

Hence \( w(x, t) \) gives us the cumulative distribution function (cdf) of \( \tau \), given that \( X(0) = x \). The second problem uses the initial and boundary conditions from (7.17), and is

\[
\frac{\partial w}{\partial t} = Lw, \quad t > 0, \quad x \in G,
\]

\[
w(0, x) = 1_{x \in G_0}, \quad x \in G \cup \partial G,
\]

\[
w(t, x) = 0, \quad t > 0, \quad x \in \partial G.
\]

The solution to this problem has the probabilistic representation

\[
w(x, t) = \mathbb{P} (\tau \geq t, X(\tau) \in G_0).
\]

In what follows we shall often refer to the exit point \([\tau, X(\tau)]^T\) of the process \([t, X(t)]^T\) from the region \( Q = [0, T_{\text{inc}}) \times G \), where we fix a time \( T_{\text{inc}} > 0 \). The exit time \( \tau = \min(\tau_G, T_{\text{inc}}) \), where \( \tau_G \) is the first exit time of \( X(t) \) from \( G \) (which may be larger than \( T_{\text{inc}} \)), and hence \( 0 < \tau \leq T_{\text{inc}} \).

7.2 Simulating the exit time for a one-dimensional Wiener process

For a fixed time \( T_{\text{inc}} > 0 \), we would like to simulate the exit point \([\tau, W(\tau)]^T\) of the process \([t, W(t)]^T\), where \( W(t) \) is a one-dimensional Wiener process \( W(t) \) with initial value \( W(0) = x \), from the region \( Q = [0, T_{\text{inc}}) \times G \), where \( G \subset \mathbb{R} \). We can do this if
we know the cdf $\mathbb{P}(\tau < T_{inc})$ of $\tau$, and we can find this from the theory in §7.1. We can get $W(t)$ from the Itô diffusion $X(t)$ in (7.4) by setting $f(x) = 0$ and $g(x) = 1$, and so $W(t)$ has generator

$$A = \frac{1}{2} \frac{\partial^2}{\partial x^2}.$$ (7.27)

We shall set $G = (-r, r)$, and hence $\partial G = \{-r, r\}$. Hence, from (7.19)-(7.21), we must solve the following mixed problem:

$$\frac{\partial u}{\partial t} = \frac{1}{2} \frac{\partial^2 u}{\partial x^2}, \quad t > 0, \quad x \in (-r, r),$$ (7.28)

$$u(0, x_0) = 0, \quad x_0 \in [-r, r],$$ (7.29)

$$u(t, -r) = u(t, r) = 1, \quad t > 0.$$ (7.30)

The solution $u(t, x)$ has the probabilistic representation

$$u(t, x) = \mathbb{P}(\tau < t).$$ (7.31)

We can simplify the problem by setting

$$v(t, x) = u(t, x) - 1 = \mathbb{P}(\tau < t) - 1;$$ (7.32)

the new problem is then

$$\frac{\partial v}{\partial t} = \frac{1}{2} \frac{\partial^2 v}{\partial x^2}, \quad t > 0, \quad x \in (-r, r),$$ (7.33)

$$v(0, x) = -1, \quad x \in [-r, r],$$ (7.34)

$$v(t, -r) = v(t, r) = 0, \quad t > 0.$$ (7.35)

There are two possible methods for solving the problem (7.33)-(7.35); separation of variables, and using a Green’s function.

### 7.2.1 Separation of variables

By setting $v(t, x) = X(x)Y(t)$, we get the eigenproblem

$$\frac{1}{2} \frac{X''(x)}{X(x)} = \frac{Y'(t)}{Y(t)} = \mu,$$ (7.36)

for some constant $\mu$ ($\mu$ is a constant since it must be independent of both $t$ and $x$).

The boundary condition (7.35) can be used to get a boundary condition for $X(x)$,

$$X(-r) = X(r) = 0,$$ (7.37)
where we assume $Y(t) \neq 0$ (we cannot get an initial condition for $Y(t)$ from (7.34)). Firstly let us solve

$$X''(x) - 2\mu X(x) = 0.$$  

(7.38)

The case $\mu \geq 0$ leads to the trivial solution $X(x) = 0$, whereas the case $\mu < 0$ leads to two possible eigenfunction/eigenvalue pairs for $X(x)$:

$$X_{\mu}(x) = \cos \left( \frac{2k + 1}{2r} \pi x \right), \quad \mu = -\frac{(2k + 1)^2\pi^2}{8r^2}, \quad k \geq 0,$$

(7.39)

$$X_{\mu}(x) = \sin \left( \frac{k + 1}{r} \pi x \right), \quad \mu = -\frac{(k + 1)^2\pi^2}{2r^2}, \quad k \geq 0.$$  

(7.40)

Clearly the eigenfunction/eigenvalue pair for $Y(t)$ is

$$Y_{\mu}(t) = \exp(\mu t).$$

(7.41)

We can then write the general solution of (7.33)-(7.35) as

$$v(x, t) = \sum_{k=0}^{\infty} a_k \cos \left( \frac{2k + 1}{2r} \pi x \right) \exp \left( -\frac{(2k + 1)^2\pi^2 t}{8r^2} \right)$$

$$+ b_k \sin \left( \frac{k + 1}{r} \pi x \right) \exp \left( -\frac{(k + 1)^2\pi^2 t}{2r^2} \right).$$

(7.42)

From the initial condition $v(x, 0) = -1$ we see that setting $t = 0$ in (7.42) yields

$$v(x, 0) = -1 = \sum_{k=0}^{\infty} \left[ a_k \cos \left( \frac{2k + 1}{2r} \pi x \right) + b_k \sin \left( \frac{k + 1}{r} \pi x \right) \right],$$

(7.43)

and since $-1$ is an even function, $v(x, 0)$ must be also, and hence $b_k = 0$ for all $k \geq 0$. To find $a_k$, we multiply both sides of (7.43) by $\frac{1}{r} \cos \left( \frac{2m + 1}{2r} \pi x \right)$ and integrate over $G = [-r, r]$. Since $\left\{ \frac{1}{\sqrt{r}} \cos \left( \frac{2k + 1}{2r} \pi x \right) \right\}_{k \geq 0}$ forms an orthonormal basis for $[-r, r]$, we have

$$- \int_{-r}^{r} \frac{1}{r} \cos \left( \frac{2m + 1}{2r} \pi x \right) \cos \left( \frac{2k + 1}{2r} \pi x \right) dx = -\frac{4}{\pi} \frac{(-1)^m}{2m + 1} = a_m,$$

(7.44)

and hence

$$u(t, x) = \mathbb{P}(\tau < t)$$

$$= 1 - \frac{4}{\pi} \sum_{k=0}^{\infty} \frac{(-1)^k}{2k + 1} \cos \left( \frac{2k + 1}{2r} \pi x \right) \exp \left( -\frac{(2k + 1)^2\pi^2 t}{8r^2} \right).$$

(7.45)
7.2.2 Using a Green’s function

We use the method of images, outlined in [79, §7.5] in order to solve the problem (7.33)-(7.35) using a Green’s function. For a fixed $T > 0$, the Green’s function $K(x, t; y, s)$ for the problem (7.33)-(7.35) satisfies

$$\frac{\partial K}{\partial t} + \frac{1}{2} \frac{\partial^2 K}{\partial t^2} = -\delta(x - y)\delta(t - s), \quad s, t < T, x, y \in (-r, r),$$

$$K(x, T; y, s) = 0, \quad x, y \in [-r, r],$$

$$K(-r, t; y, s) = K(r, t; y, s) = 0, \quad s, t > 0,$$

where $\delta(\cdot)$ denotes the Dirac delta function and the point $x = y, t = s$ is a source point. Once such a Green’s function is found, the solution of (7.33)-(7.35) can be expressed as

$$v(t, x) = -\int_{-r}^{r} K(x', 0; x, t) dx'.$$  

(7.49)

We shall first find the free space Green’s function $K_F$, valid for all $x, y \in \mathbb{R}$, which does not have to satisfy the boundary condition (7.48), and then use this along with the method of images to find the Green’s function $G$ which satisfies (7.48).

Finding $K_F$

The free space Green’s function $K_F(x, t; y, s)$ must satisfy

$$\frac{\partial K_F}{\partial t} + \frac{1}{2} \frac{\partial^2 K_F}{\partial t^2} = -\delta(x - y)\delta(t - s), \quad x, y \in \mathbb{R}, s, t < T_{\text{inc}},$$

$$K_F(x, T; y, s) = 0.$$

(7.50)

Firstly we must perform a Fourier transform of (7.50). Recall that for a function $f(x)$, its Fourier transform $\hat{f}(\lambda)$ is defined as

$$\hat{f}(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\lambda x} f(x) dx,$$

(7.52)

and satisfies

$$\hat{f}^{(n)}(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\lambda x} f^{(n)}(x) dx = (-i\lambda)^n \hat{f}(\lambda).$$

(7.53)

Let $\hat{K}_F(\lambda, t; y, s)$ denote the Fourier transform of $K_F(x, t; y, s)$. Then using (7.53) and the fact that the Fourier transform of $f(x) = \delta(x - y)$ is

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\lambda x} \delta(x - y) dx = \frac{1}{\sqrt{2\pi}} e^{i\lambda y},$$

(7.54)
we can now transform (7.50)-(7.51), resulting in
\[
\frac{\partial \hat{K}_F}{\partial t} - \frac{\lambda^2}{2} \hat{K}_F = -\frac{1}{\sqrt{2\pi}} e^{i\lambda y} \delta(t - s), \quad \lambda, y \in \mathbb{R}, \quad s, t \leq T
\]
(7.55)
\[
\hat{K}_F(\lambda, T; y, s) = 0.
\]
(7.56)

We now solve (7.55)-(7.56). Firstly we multiply by the integrating factor \( e^{-\frac{\lambda^2}{2} t} \) and also use the fact that for a function \( g(x) \),
\[
g(x) \delta(x - y) = g(y) \delta(y - x).
\]
This results in
\[
e^{-\frac{\lambda^2}{2} t} \frac{\partial \hat{K}_F}{\partial t} - e^{-\frac{\lambda^2}{2} t} \frac{\lambda^2}{2} \hat{K}_F = -\frac{\partial}{\partial t} \left( e^{-\frac{\lambda^2}{2} t} \hat{K}_F \right) = -\frac{1}{\sqrt{2\pi}} e^{i\lambda y - \frac{\lambda^2}{2} s} \delta(s - t).
\]
(7.57)

Now we simply integrate both sides with respect to \( t \), over the interval \([0, T]\). Note that
\[
\int_0^T \delta(s - t) \, dt = [H(s - t)]_0^T = H(s - T) - H(s) = -1,
\]
(7.58)
and so using this and (7.56), we obtain
\[
\hat{K}_F(\lambda, t; y, s) - e^{-\frac{\lambda^2}{2} T} \hat{K}_F(\lambda, T; y, s) = \hat{K}_F(\lambda, t; y, s) = \frac{1}{\sqrt{2\pi}} e^{i\lambda y + \frac{\lambda^2}{2} (t - s)}.
\]
(7.59)

Since we only need \( K_F(x, 0; y, s) \), we set \( t = 0 \) to obtain
\[
\hat{K}_F(\lambda, 0; y, s) = \frac{1}{\sqrt{2\pi}} e^{i\lambda y - \frac{\lambda^2}{2} s}.
\]
(7.60)

To recover \( K_F(x, 0; y, s) \) we perform an inverse Fourier transform; hence
\[
K_F(x, 0; y, s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-i\lambda x} \hat{K}_F(\lambda, 0; y, s) \, d\lambda = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\lambda(y - x) - \frac{\lambda^2}{2} s} \, d\lambda = \frac{1}{\sqrt{2\pi}} \int_0^\infty e^{-\frac{\lambda^2}{2} s} \cos(\lambda(x - y)) \, d\lambda.
\]

We can calculate this explicitly. Let
\[
I(\alpha) = \frac{1}{\pi} \int_0^\infty e^{-\frac{\lambda^2}{2} s} \cos(\alpha \lambda) \, d\lambda.
\]

Due to the exponential decay of the integrand, this can be differentiated directly and hence
\[
\frac{dI(\alpha)}{d\alpha} = -\frac{1}{\pi} \int_0^\infty \lambda e^{-\frac{\lambda^2}{2} s} \sin(\alpha \lambda) \, d\lambda = \frac{1}{\pi s} \int_0^\infty \sin(\alpha \lambda) \, d(e^{-\frac{\lambda^2}{2} s}) \, d\lambda = \frac{1}{\pi s} \left( \left[ \sin(\alpha e^{-\frac{\lambda^2}{2} s}) \right]_0^\infty - \int_0^\infty \alpha \cos(\alpha \lambda) e^{-\frac{\lambda^2}{2} s} \, d\lambda \right) = -\frac{\alpha}{s} I(\alpha),
\]

where we used the fact that \( [\sin(\alpha e^{-\frac{\lambda^2}{2} s})]_0^\infty = 0 \).
a first order ODE for $I(\alpha)$. Separating the variables and integrating, we have

$$\int_0^\alpha \frac{dI(\beta)}{d\beta} = \int_0^\alpha -\frac{\beta}{s} d\beta,$$

and hence

$$\ln \left( \frac{I(\alpha)}{I(0)} \right) = -\frac{\alpha^2}{2s},$$

Now we use $I(0) = \frac{1}{\pi} \int_0^\infty e^{-\frac{\lambda^2}{2s}} d\lambda = \frac{1}{\sqrt{2\pi s}}$, to obtain

$$I(\alpha) = I(0) \exp \left( -\frac{\alpha^2}{2s} \right) = \frac{1}{\sqrt{2\pi s}} \exp \left( -\frac{\alpha^2}{2s} \right).$$

Finally we can write down $K_F(x, 0; y, s)$ as

$$K_F(x, 0; y, s) = \frac{1}{\sqrt{2\pi s}} \exp \left( -\frac{(x-y)^2}{2s} \right). \quad (7.61)$$

**Finding $K$**

In order for $K$ to satisfy the boundary condition (7.48), we use the method of images. Since we need $K(r, 0; y, s) = 0$, we firstly set $K_0(x, 0; y, s) = K_F(x, 0; y, s)$ and

$$K_1(x, 0; y, s) = K_0(x, 0; y, s) - K_F(x, 0; 2r - y, s)
= \frac{1}{\sqrt{2\pi s}} \left[ \exp \left( -\frac{(x-y)^2}{2s} \right) - \exp \left( -\frac{(x+y-2r)^2}{2s} \right) \right].$$

Here we are adding a source point $x = 2r - y$, which is the original source point $x = y$ reflected in $x = r$. We see that $K_1(r, 0; y, s) = 0$ as required; however we also need $K_1(-r, 0; y, s) = 0$. This is not satisfied, so we need to add another term to $K_1$ in an attempt to fix this, by adding a source point $x = -2r - y$, which is the original source point $x = y$ reflected in $x = -r$. So now we have

$$K_1(x, 0; y, s) = K_0(x, 0; y, s) - K_F(x, 0; 2r - y, s) - K_F(x, 0; -2r - y, s)
= \frac{1}{\sqrt{2\pi s}} \left[ \exp \left( -\frac{(x-y)^2}{2s} \right) - \exp \left( -\frac{(x+y-2r)^2}{2s} \right) \right. \left. - \exp \left( -\frac{(x+y+2r)^2}{2s} \right) \right].$$
However, we find that now neither $K_1(r, 0; y, s)$ nor $K_1(-r, 0; y, s)$ is zero; in fact

$$K_1(1, 0; y, s) = \frac{1}{\sqrt{2\pi s}} \left[ \exp \left( -\frac{(r - y)^2}{2s} \right) - \exp \left( -\frac{(y - r)^2}{2s} \right) \right]$$

$$= - \frac{1}{\sqrt{2\pi s}} \exp \left( -\frac{(-3r - y)^2}{2s} \right) = -K_F(-3r, 0; y, s),$$

$$K_1(-r, 0; y, s) = \frac{1}{\sqrt{2\pi s}} \left[ \exp \left( -\frac{(-r - y)^2}{2s} \right) - \exp \left( -\frac{(y - 3r)^2}{2s} \right) \right]$$

$$= - \frac{1}{\sqrt{2\pi s}} \exp \left( -\frac{(3r - y)^2}{2s} \right) = -K_F(3r, 0; y, s).$$

We shall attempt to fix this by adding more source points; we need to reflect each of the two new source points in the boundary which was not the one used to create them. So we reflect $x = 2r - y$ in $x = -r$ to get a source point $x = -4r + y$, and we reflect $x = -2r - y$ in $x = r$ to get a source point $x = 4r + y$, and set

$$K_2(x, 0; y, s) = K_1(x, 0; y, s) + K_F(x, 0; -4r + y, s) + K_F(x, 0; 4r + y, s).$$

Similarly to above, we find that

$$K_2(r, 0; y, s) = K_F(5r, 0; y, s),$$

$$K_2(-r, 0; y, s) = K_F(-5r, 0; y, s).$$

This necessitates adding more source points; inductively this process continues to infinity, and for any $k \geq 0$,

$$K_{k+1}(x, 0; y, s) = K_k(x, 0; y, s) + (-1)^k K_F(x, 0; 2kr + (-1)^k y, s)$$

$$+ (-1)^k K_F(x, 0; -2kr + (-1)^k y, s).$$

Hence

$$K(x, 0; y, s) = K_0(x, 0; y, s) + \sum_{k=1}^{\infty} \left[ (-1)^k K_F(x, 0; 2kr + (-1)^k y, s)$$

$$+ (-1)^k K_F(x, 0; -2kr + (-1)^k y, s) \right]$$

$$= \frac{1}{\sqrt{2\pi s}} \exp \left( -\frac{(x - y)^2}{2s} \right)$$

$$+ \frac{1}{\sqrt{2\pi s}} \sum_{k=1}^{\infty} \left[ \exp \left( -\frac{(x - 2kr - (-1)^k y)^2}{2s} \right) \right]$$
Finally, from (7.49) and (7.32), it follows that
\[
u(t, x) = \mathbb{P}(\tau < t)
= 1 - \int_{-r}^{r} K(x', 0; x, t) \, dx'
= 1 - \frac{1}{\sqrt{2\pi t}} \int_{-r}^{r} \exp\left(-\frac{(x' - x)^{2}}{2t}\right) \, dx'
- \frac{1}{\sqrt{2\pi t}} \sum_{k=1}^{\infty} \left[ \int_{-r}^{r} \exp\left(-\frac{(x' + 2kr - (-1)^{k}x)^{2}}{2t}\right) \, dx' \right.
+ \left. \int_{-r}^{r} \exp\left(-\frac{(x' - 2kr - (-1)^{k}x)^{2}}{2t}\right) \, dx' \right].
\tag{7.62}
\]

Setting \(x = 0\) and \(t = T_{\text{inc}}\)

When simulating bounded Wiener increments using a Wiener process \(W(t)\), we need the case \(W(0) = 0\) and \(t = T_{\text{inc}}\). Hence, to get probability distributions for the exit time \(\tau\) in this case, we set \(x = 0\) in (7.45) and (7.62). Since \(\nu(T_{\text{inc}}, 0) = \mathbb{P}(\tau < T_{\text{inc}})\) is the cdf of \(\tau\), we set \(\mathbb{P}(T_{\text{inc}}) = \nu(T_{\text{inc}}, 0)\), and setting \(x = 0\) in (7.45) and (7.62) yields
\[
\mathbb{P}(T_{\text{inc}}) = 1 - \frac{4}{\pi} \sum_{k=0}^{\infty} \frac{(-1)^{k}}{2k + 1} \exp\left(-\frac{(2k + 1)^{2}\pi^{2}}{8r^{2}T_{\text{inc}}}\right),
\tag{7.63}
\]
and
\[
\mathbb{P}(T_{\text{inc}}) = 1 - \frac{1}{\sqrt{2\pi T_{\text{inc}}}} \int_{-r}^{r} \exp\left(-\frac{x^{2}}{2T_{\text{inc}}}\right) \, dx
- \sum_{k=1}^{\infty} \left[ \int_{-r}^{r} \exp\left(-\frac{(x + 2kr)^{2}}{2T_{\text{inc}}}\right) \, dx + \int_{-r}^{r} \exp\left(-\frac{(x - 2kr)^{2}}{2T_{\text{inc}}}\right) \, dx \right].
\tag{7.64}
\]

We can manipulate (7.64) to get an explicit expression for \(\mathbb{P}(T_{\text{inc}})\), which is more suitable for calculations. To this end, we look at the integral
\[
I(k) = \frac{1}{\sqrt{2\pi T_{\text{inc}}}} \int_{-r}^{r} \exp\left(-\frac{(x + 2kr)^{2}}{2T_{\text{inc}}}\right) \, dx
\]
and write
\[
\mathbb{P}(T_{\text{inc}}) = 1 - I(0) - \sum_{k=1}^{\infty} (-1)^{k}(I(k) + I(-k)).
\tag{7.65}
\]
By using the substitution \(s = \frac{x + 2kr}{\sqrt{2T_{\text{inc}}}}\), we obtain
\[
I(k) = \frac{1}{\sqrt{2\pi T_{\text{inc}}}} \int_{-r}^{r} \sqrt{2T_{\text{inc}}} \exp(-s^{2}) \, ds = \frac{1}{\sqrt{\pi}} \int_{-r}^{r} \sqrt{2T_{\text{inc}}} \exp(-s^{2}) \, ds.
\]
We need to evaluate $I(k)$ and $I(-k)$, for $k \geq 1$, and $I(0)$. Firstly

$$I(k) = \frac{1}{2} \left[ \text{erfc} \left( \frac{(2k-1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{(2k+1)r}{\sqrt{2T_{\text{inc}}}} \right) \right],$$

which follows directly from

$$\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty \exp(-s^2) \, ds.$$

Next, due to the fact that $\exp(-s^2)$ is an even function of $s$, we obtain

$$I(-k) = \frac{1}{\sqrt{\pi}} \int_{-r(1+2k)/\sqrt{2T_{\text{inc}}}}^{r(1-2k)/\sqrt{2T_{\text{inc}}}} \exp(-s^2) \, ds = \frac{1}{\sqrt{\pi}} \int_{r(2k-1)/\sqrt{2T_{\text{inc}}}}^{r(2k+1)/\sqrt{2T_{\text{inc}}}} \exp(-s^2) \, ds = I(k).$$

Finally, again using the fact that $\exp(-s^2)$ is an even function of $s$,

$$I(0) = \frac{1}{\sqrt{\pi}} \int_{-r/\sqrt{2T_{\text{inc}}}}^{r/\sqrt{2T_{\text{inc}}}} \exp(-s^2) \, ds$$

$$= \frac{2}{\sqrt{\pi}} \int_{0}^{r/\sqrt{2T_{\text{inc}}}} \exp(-s^2) \, ds = 1 - \text{erfc} \left( \frac{r}{\sqrt{2T_{\text{inc}}}} \right),$$

where we use $\text{erfc}(1) = 0$. Substituting into (7.65), we find that

$$\mathcal{P}_r(T_{\text{inc}}) = 1 - \frac{4}{\pi} \sum_{k=0}^\infty (-1)^k \exp \left( -\frac{(2k+1)^2\pi^2}{8r^2} T_{\text{inc}} \right), \quad (7.66)$$

$$\mathcal{P}_r(T_{\text{inc}}) = 2 \sum_{k=0}^\infty (-1)^k \text{erfc} \left( \frac{(2k+1)r}{\sqrt{2T_{\text{inc}}}} \right). \quad (7.67)$$

In summary

We now have two expressions for the cdf of $\tau$, expressed by $\mathcal{P}_r(T_{\text{inc}}) = \mathbb{P}(\tau < T_{\text{inc}})$.

They are:

$$\mathcal{P}_r(T_{\text{inc}}) = 1 - \frac{4}{\pi} \sum_{k=0}^\infty (-1)^k \frac{(2k+1)^2\pi^2}{8r^2} T_{\text{inc}}, \quad (7.66)$$

$$\mathcal{P}_r(T_{\text{inc}}) = 2 \sum_{k=0}^\infty (-1)^k \text{erfc} \left( \frac{(2k+1)r}{\sqrt{2T_{\text{inc}}}} \right). \quad (7.67)$$
CHAPTER 7. SIMULATING A WIENER PROCESS

It is clearly impractical to evaluate an infinite sum for practical purposes, so we look at both of these expressions to find a suitable truncation point. To do this, we follow \[57,58\].

We can differentiate both (7.66) and (7.67) term-by-term to obtain $P'_r(T_{\text{inc}})$, the probability density function (pdf) of $\tau$, which is

$$P'_r(T_{\text{inc}}) = \frac{\pi}{2r^2} \sum_{k=0}^{\infty} (-1)^k (2k + 1) \exp \left( -\frac{(2k + 1)^2 \pi^2}{8r^2} T_{\text{inc}} \right),$$  

(7.68)

$$P'_r(T_{\text{inc}}) = \frac{2r}{\sqrt{2\pi T_{\text{inc}}^3}} \sum_{k=0}^{\infty} (-1)^k (2k + 1) \exp \left( -\frac{(2k + 1)^2 r^2}{2T_{\text{inc}}} \right).$$  

(7.69)

We look at the remainder terms $c_k(T_{\text{inc}})$ and $d_k(T_{\text{inc}})$, which are the absolute values of the $(k+1)$-th terms of the sums in (7.68) and (7.69) respectively:

$$c_k(T_{\text{inc}}) = \frac{\pi}{2r^2} (2k + 3) \exp \left( -\frac{(2k + 3)^2 \pi^2}{8r^2} T_{\text{inc}} \right),$$  

(7.70)

$$d_k(T_{\text{inc}}) = \frac{2r}{\sqrt{2\pi T_{\text{inc}}^3}} (2k + 3) \exp \left( -\frac{(2k + 3)^2 r^2}{2T_{\text{inc}}} \right).$$  

(7.71)

Similarly we denote by $C_k(T_{\text{inc}})$ and $D_k(T_{\text{inc}})$ the absolute values of the $(k+1)$-th terms of the sums in (7.66) and (7.67) respectively:

$$C_k(T_{\text{inc}}) = \frac{4}{\pi} \exp \left( -\frac{(2k + 3)^2 \pi^2}{8r^2} T_{\text{inc}} \right),$$  

(7.72)

$$D_k(T_{\text{inc}}) = 2 \text{erfc} \left( \frac{(2k + 3)r}{\sqrt{2T_{\text{inc}}}} \right).$$  

(7.73)

We see that $c_k \left( \frac{2r^2}{\pi} \right) = d_k \left( \frac{2r^2}{\pi} \right)$ and that

$$c_k(T_{\text{inc}}) < c_k \left( \frac{2r^2}{\pi} \right), \quad T_{\text{inc}} > \frac{2r^2}{\pi},$$

$$d_k(T_{\text{inc}}) < c_k \left( \frac{2r^2}{\pi} \right), \quad T_{\text{inc}} < \frac{2r^2}{\pi}.$$  

We find that for $k = 2$, $c_2 \left( \frac{2r^2}{\pi} \right) = d_2 \left( \frac{2r^2}{\pi} \right) = 2.1262 \times 10^{-16}$, and so if we set

$$P'_r(T_{\text{inc}}) = \begin{cases} 
\frac{\pi}{2r^2} \sum_{k=0}^{2} (-1)^k (2k + 1) \exp \left( -\frac{(2k + 1)^2 \pi^2}{8r^2} T_{\text{inc}} \right), & T_{\text{inc}} > \frac{2r^2}{\pi}, \\
\frac{2r}{\sqrt{2\pi T_{\text{inc}}^3}} \sum_{k=0}^{2} (-1)^k (2k + 1) \exp \left( -\frac{(2k + 1)^2 r^2}{2T_{\text{inc}}} \right), & 0 < T_{\text{inc}} \leq \frac{2r^2}{\pi},
\end{cases}$$  

(7.74)
then the maximum error in approximating $P_r'(T_{inc})$ by $\bar{P}_r'(T_{inc})$ is less than $2.13 \times 10^{-16}$ (the terms for $k > 2$ are negligible; for example $c_3\left(\frac{2}{\pi}\right) = 3.3246 \times 10^{-27}$). We approximate $P_r(T_{inc})$ in the same way, such that

$$
\bar{P}_r(T_{inc}) = \begin{cases} 
1 - \frac{4}{\pi} \sum_{k=0}^{2} \exp \left( -\frac{(2k+1)^2}{8r^2} T_{inc} \right), & T_{inc} > \frac{2r^2}{\pi}, \\
2 \sum_{k=0}^{2} (-1)^k \text{erfc} \left( \frac{(2k+1)r}{\sqrt{2}T_{inc}} \right), & 0 < T_{inc} \leq \frac{2r^2}{\pi}.
\end{cases} \quad (7.75)
$$

To compute the error in approximating $P_r(T_{inc})$ by $\bar{P}_r(T_{inc})$, note that

$$
|P_r(T_{inc}) - \bar{P}_r(T_{inc})| = \left| \int_0^{T_{inc}} (P_r'(s) - \bar{P}_r'(s)) \, ds \right| \leq \int_0^{T_{inc}} |P_r'(s) - \bar{P}_r'(s)| \, ds.
$$

The quantity $|P_r'(s) - \bar{P}_r'(s)|$ can be approximated to high accuracy by the minimum of $c_2(s)$ and $d_2(s)$ and so if $T_{inc} \leq \frac{2r^2}{\pi}$,

$$
|P_r(T_{inc}) - \bar{P}_r(T_{inc})| = \int_0^{T_{inc}} d_2(s) \, ds,
$$

and if $T_{inc} > \frac{2r^2}{\pi}$ then

$$
|P_r(T_{inc}) - \bar{P}_r(T_{inc})| = \int_0^{2r^2/\pi} d_2(s) \, ds + \int_{2r^2/\pi}^{T_{inc}} c_2(s) \, ds.
$$

Note that the antiderivatives of $c_k$ and $d_k$ are $-C_k$ and $D_k$ respectively and hence in the case where $T_{inc} \leq \frac{2r^2}{\pi}$,

$$
|P_r(T_{inc}) - \bar{P}_r(T_{inc})| = D_2(T_{inc}) - D_2(0) \leq D_2\left(\frac{2r^2}{\pi}\right),
$$

since $D_2(0) = 0$, and in the case where $T_{inc} > \frac{2r^2}{\pi}$,

$$
|P_r(T_{inc}) - \bar{P}_r(T_{inc})| = D_2\left(\frac{2r^2}{\pi}\right) - D_2(0) + C_2\left(\frac{2r^2}{\pi}\right) - C_2(T_{inc})
\leq C_2\left(\frac{2r^2}{\pi}\right) + D_2\left(\frac{2r^2}{\pi}\right).
$$

Hence the error for all $T_{inc}$ is bounded above by $C_2\left(\frac{2r^2}{\pi}\right) + D_2\left(\frac{2r^2}{\pi}\right) = 6.9905 \times 10^{-18}$, which is an acceptable error for practical calculations. To simulate the exit time $\tau$, let $\gamma \sim U(0,1)$ be a uniformly distributed random variable. Then the random variable

$$
\tau = P_r^{-1}(\gamma),
$$

where $P_r^{-1}$ is the inverse of $P_r$, has cdf $P_r(T_{inc})$. In practice we cannot solve this directly, since we do not know $P_r^{-1}$; instead we use a Newton–Raphson method, or other root-finding method, to solve

$$
\bar{P}_r(\tau) - \gamma = 0.
$$
7.3 Simulating the exit point of a Wiener process

if $\tau = T_{\text{inc}}$

Let us suppose that $[t, W(t)]^T$ does not exit $Q$ at a time $\tau < T_{\text{inc}}$; in this instance $\tau = T_{\text{inc}}$. To simulate the exit point $\xi = W(\tau)$ of $W(t)$, $W(0) = x$ from $G = (-r, r)$, given that $\tau = T_{\text{inc}}$, we need to know its conditional cdf, which is given by

$$Q_r^x(\beta; T_{\text{inc}}) = \mathbb{P}(\xi < \beta | \tau = T_{\text{inc}}) = \frac{\mathbb{P}(\tau \geq T_{\text{inc}}, \xi < \beta)}{\mathbb{P}(\tau \geq T_{\text{inc}})}.$$  \hfill (7.76)

From the theory in §7.1, if we solve the PDE problem

$$\frac{\partial u}{\partial t} = \frac{1}{2} \frac{\partial^2 u}{\partial x^2}, \quad t > 0, x \in (-r, r),$$  \hfill (7.77)

$$u(0, x) = 1_{x \in [-r, \beta]}, \quad x \in [-r, r],$$  \hfill (7.78)

$$u(t, -r) = u(t, r) = 0, \quad t > 0,$$  \hfill (7.79)

the solution $u(t, x)$ has the probabilistic representation

$$u(t, x) = \mathbb{P}(\tau \geq T_{\text{inc}}, -r \leq \xi < \beta),$$  \hfill (7.80)

and hence

$$Q_r^x(\beta; t) = \frac{u(t, x)}{1 - \mathbb{P}(\tau < t)}.$$  

We are interested in the case $x = 0$, $t = T_{\text{inc}}$, and here

$$Q_r(\beta; T_{\text{inc}}) = \frac{\mathbb{P}(\tau \geq T_{\text{inc}}, \xi < \beta)}{1 - \mathbb{P}_r(T_{\text{inc}})} = \frac{u(0, T_{\text{inc}})}{1 - \mathbb{P}_r(T_{\text{inc}})}.$$  \hfill (7.81)

Again we have two ways to solve (7.77)-(7.79): separation of variables, and using a Green’s function.

7.3.1 Separation of variables

If we set $u(t, x) = X(x)Y(t)$, we get the eigenproblem (7.36) as before, with the same boundary conditions (7.37) for $X(x)$. Hence we get the same eigenfunctions (7.39) and (7.40) for $X(x)$, and the same eigenfunction (7.41) for $Y(t)$ as before, and the general solution is again given by (7.42). However, this time the initial condition is different; instead of $u(x, 0) = -1$ we have $u(x, 0) = 1_{x \in [-r, \beta]}$. Hence, we set $t = 0$ in (7.42) and find

$$u(x, 0) = 1_{x \in [-r, \beta]} = \sum_{k=0}^{\infty} \left[ a_k \cos \left( \left( \frac{2k + 1}{2r} \right) \pi x \right) + b_k \sin \left( \left( \frac{k + 1}{r} \right) \pi x \right) \right].$$  \hfill (7.82)
Since $1_{x \in [-r,\beta]}$ is neither even (apart from the special case $\beta = r$) nor odd, we cannot simply state that $a_k$ or $b_k$ is zero; however it can be shown that

$$
\left\{ \frac{1}{\sqrt{r}} \cos \left( \frac{2k+1}{2r} \pi x \right), \frac{1}{\sqrt{r}} \sin \left( \frac{k+1}{r} \pi x \right) \right\}_{k \geq 0}
$$

is an orthonormal basis for $[-r,r]$. Hence by multiplying both sides of (7.82) by $\frac{1}{r} \cos \left( \frac{2m+1}{2r} \pi x \right)$ and integrating over $[-r,r]$, we find that

$$
a_m = \frac{1}{r} \int_{-r}^{r} 1_{x \in [-r,\beta]} \cos \left( \frac{2m+1}{2r} \pi x \right) \, dx = \left[ \frac{2}{(2m+1)\pi} \sin \left( \frac{2m+1}{2r} \pi x \right) \right]_{-1}^{\beta} = \frac{2}{(2m+1)\pi} \sin \left( \frac{2m+1}{2r} \pi \beta \right) - \sin \left( \frac{2m+1}{2r} \pi \right),
$$

$$
= \frac{2}{(2m+1)\pi} \left[ \sin \left( \frac{2m+1}{2r} \pi \beta \right) - \sin \left( \frac{2m+1}{2r} \pi \right) \right]
$$

$$
= \frac{2}{(2m+1)\pi} \left[ \sin \left( \frac{2m+1}{2r} \pi \beta \right) - \sin \left( \frac{2m+1}{2r} \pi \right) \right] + (-1)^m.
$$

Multiplying both sides of (7.82) by $\frac{1}{r} \sin \left( \frac{m+1}{r} \pi x \right)$ and integrating over $[-r,r]$ yields

$$
b_m = \frac{1}{r} \int_{-r}^{r} 1_{x \in [-r,\beta]} \sin \left( \frac{m+1}{r} \pi x \right) \, dx = - \left[ \frac{1}{(m+1)\pi} \cos \left( \frac{m+1}{r} \pi x \right) \right]_{-1}^{\beta} = \frac{1}{(m+1)\pi} \left[ \cos \left( - \frac{m+1}{r} \pi \right) - \cos \left( \frac{m+1}{r} \pi \beta \right) \right]
$$

$$
= \frac{1}{(m+1)\pi} \left[ \cos \left( \frac{m+1}{r} \pi \right) - \cos \left( \frac{m+1}{r} \pi \beta \right) \right]
$$

$$
= \frac{2}{(m+1)\pi} \sin \left( \frac{m+1}{2} \pi (\beta + 1) \right) \sin \left( \frac{m+1}{2} \pi (\beta - 1) \right),
$$

where we use the trigonometric identity $\cos(A) - \cos(B) = 2 \sin \left( \frac{B+A}{2} \right) \sin \left( \frac{B-A}{2} \right)$. Hence, if we set $x = 0$ as before,

$$
Q_r(\beta; t) = \frac{1}{1 - P_r(t)} \frac{2}{\pi} \sum_{k=0}^{\infty} \frac{1}{2k+1} \left[ \sin \left( \frac{2k+1}{2r} \pi \beta \right) + (-1)^k \right] \times \exp \left( -\frac{(2k+1)^2\pi^2}{8r^2} t \right).
$$

(7.83)
7.3.2 Using a Green’s function

The Green’s function for this problem is the same as the one we calculated previously; the only difference between this problem and the previous problem is the initial condition. Hence

\[
u(t, x) = \int_{-r}^{r} 1_{x \in [-r, \beta]} G(x', 0; x, t) \, dx' = \int_{-r}^{\beta} G(x', 0; x, t) \, dx'.
\]

Setting \(x_0 = 0\) and \(t = T_{\text{inc}}\),

\[
u(T_{\text{inc}}, 0) = \frac{1}{\sqrt{2\pi T_{\text{inc}}}} \int_{-r}^{\beta} \exp \left( -\frac{x^2}{2T_{\text{inc}}} \right) \, dx + \frac{1}{\sqrt{2\pi T_{\text{inc}}}} \sum_{k=1}^{\infty} (-1)^k \left[ \int_{-r}^{\beta} \exp \left( -\frac{(x + 2kr)^2}{2T_{\text{inc}}} \right) \, dx \right. \\
\left. + \int_{-r}^{\beta} \exp \left( -\frac{(x - 2kr)^2}{2T_{\text{inc}}} \right) \, dx \right].
\]

Similarly to before, let

\[
I(k) = \frac{1}{\sqrt{2\pi T_{\text{inc}}}} \int_{-r}^{\beta} \exp \left( -\frac{(x + 2kr)^2}{2T_{\text{inc}}} \right) \, dx,
\]

and write

\[
u(T_{\text{inc}}, 0) = I(0) + \sum_{k=1}^{\infty} (-1)^k (I(k) + I(-k)).
\]  

Then for \(k \geq 1\), we follow the working from before (only the limits of the integrals are different) to obtain

\[
I(k) = \frac{1}{2} \left[ \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right],
\]

\[
I(-k) = \frac{1}{2} \left[ \text{erfc} \left( \frac{2kr - \beta}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{(2k + 1)r}{\sqrt{2T_{\text{inc}}}} \right) \right],
\]

\[
I(0) = \frac{1}{2} \left[ \text{erfc} \left( \frac{-r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{\beta}{\sqrt{2T_{\text{inc}}}} \right) \right].
\]

Substituting into (7.85) and (7.81) yields

\[
Q_r(\beta; T_{\text{inc}}) = \frac{1}{2(1 - P_r(T_{\text{inc}}))} \left\{ \text{erfc} \left( \frac{-r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{\beta}{\sqrt{2T_{\text{inc}}}} \right) \right. \\
\left. + \sum_{k=1}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{2kr - \beta}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{(2k + 1)r}{\sqrt{2T_{\text{inc}}}} \right) \right. \\
\left. \left. \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right] \right\}
\]

\[
= \frac{1}{2(1 - P_r(T_{\text{inc}}))} \left\{ \sum_{k=0}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right] \right\}
\]

\[
= \frac{1}{2(1 - P_r(T_{\text{inc}}))} \left\{ \sum_{k=0}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right] \right\}
\]
CHAPTER 7. SIMULATING A WIENER PROCESS

\[ + \sum_{k=1}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{2k - \beta}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{(2k + 1)r}{\sqrt{2T_{\text{inc}}}} \right) \right] \]

\[ = \frac{1}{2(1 - \mathcal{P}_r(T_{\text{inc}}))} \left\{ \sum_{k=0}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right] \right\} \]

\[ + \sum_{k'=0}^{\infty} (-1)^{k'+1} \left[ \text{erfc} \left( \frac{(2k' + 2)r - \beta}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{(2k' + 3)r}{\sqrt{2T_{\text{inc}}}} \right) \right] \]

\[ = \frac{1}{2(1 - \mathcal{P}_r(T_{\text{inc}}))} \sum_{k=0}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right] \]

\[ \quad - \text{erfc} \left( \frac{(2k + 2)r - \beta}{\sqrt{2T_{\text{inc}}}} \right) + \text{erfc} \left( \frac{(2k + 3)r}{\sqrt{2T_{\text{inc}}}} \right) \] \quad (7.86)

where \( k' = k - 1 \).

**In summary**

We now have two expressions for the conditional cdf of the exit point \( \xi = W(\tau) \) of \( W(t), W(0) = 0 \) from \( G = (-r, r) \), given that \( \tau = T_{\text{inc}} \), characterised by \( Q_r(\beta; T_{\text{inc}}) \) given in (7.81). They are

\[ Q_r(\beta; T_{\text{inc}}) = \frac{1}{1 - \mathcal{P}_r(T_{\text{inc}}) \frac{2}{\pi}} \sum_{k=0}^{\infty} \frac{1}{2k + 1} \left[ \sin \left( \frac{(2k + 1)r}{2r} \pi \beta \right) + (-1)^k \right] \]

\[ \times \exp \left( -\frac{(2k + 1)^2 \pi^2 T_{\text{inc}}}{8r^2} \right), \quad (7.87) \]

\[ Q_r(\beta; T_{\text{inc}}) = \frac{1}{2(1 - \mathcal{P}_r(T_{\text{inc}}))} \sum_{k=0}^{\infty} (-1)^k \left[ \text{erfc} \left( \frac{(2k - 1)r}{\sqrt{2T_{\text{inc}}}} \right) - \text{erfc} \left( \frac{2kr + \beta}{\sqrt{2T_{\text{inc}}}} \right) \right] \]

\[ \quad - \text{erfc} \left( \frac{(2k + 2)r - \beta}{\sqrt{2T_{\text{inc}}}} \right) + \text{erfc} \left( \frac{(2k + 3)r}{\sqrt{2T_{\text{inc}}}} \right) \] \quad (7.88)

As before, for calculations we shall approximate \( Q_r(\beta; T_{\text{inc}}) \) by using (7.87) for \( T_{\text{inc}} > \frac{2r^2}{\pi} \), and (7.88) for \( T_{\text{inc}} \leq \frac{2r^2}{\pi} \), with the sums truncated to 3 terms, and call this approximation \( \tilde{Q}_r(\beta; T_{\text{inc}}) \). Note that \( \mathcal{P}_r(T_{\text{inc}}) \) is very close to 1 for large \( T_{\text{inc}} \), which could cause division by zero if \( T_{\text{inc}} \) is allowed to be too large.

To simulate the exit point \( \xi \), let \( \gamma \sim U(0, 1) \) be a uniformly distributed random variable. Then the random variable

\[ \xi = Q_r^{-1}(\gamma; T_{\text{inc}}) \]

where \( Q_r^{-1} \) is the inverse of \( Q_r \), has cdf \( Q_r(\beta; T_{\text{inc}}) \). Again, since this equation cannot be solved directly, we use a Newton–Raphson method or other root-finding method to
solve
\[ \bar{Q}_r(\beta; T_{\text{inc}}) - \gamma = 0. \]

Scaling to set \( r = 1 \)

Notice that in all of the above,
\[
\begin{align*}
\mathcal{P}_r(T_{\text{inc}}) &= \mathcal{P}_1 \left( \frac{T_{\text{inc}}}{r^2} \right), \\
\bar{\mathcal{P}}_r(T_{\text{inc}}) &= \bar{\mathcal{P}}_1 \left( \frac{T_{\text{inc}}}{r^2} \right), \\
\mathcal{Q}_r(\beta, T_{\text{inc}}) &= \mathcal{Q}_1 \left( \frac{\beta}{r} ; \frac{T_{\text{inc}}}{r^2} \right), \\
\bar{\mathcal{Q}}_r(\beta, T_{\text{inc}}) &= \bar{\mathcal{Q}}_1 \left( \frac{\beta}{r} ; \frac{T_{\text{inc}}}{r^2} \right).
\end{align*}
\]

This comes from the fact that for a Wiener process \( W(t) \), \( W(0) = 0 \), the stochastic process \( \frac{1}{r} W \left( r^2 t \right) \) is also a Wiener process. This aids us computationally since we now only need to store the functions \( \mathcal{P}_1(T_{\text{inc}}) \) and \( \mathcal{Q}_1(\beta; T_{\text{inc}}) \) and scale the arguments to get \( \mathcal{P}_r(T_{\text{inc}}) \) and \( \mathcal{Q}_r(\beta; T_{\text{inc}}) \) for any value of \( r \).

7.4 Simulating the exit point and exit time of a one-dimensional Wiener process with unbounded stepsize and bounded stepsize

We now have a procedure for simulating the exit point of \( W(t) \) from \( G = (-r, r) \), where the exit time is unbounded, and also the exit point of \([t, W(t)]^T \) from \( Q = [0, T_{\text{inc}}) \times G \), where the exit time is bounded above by \( T_{\text{inc}} > 0 \).

Unbounded stepsize

1. Simulate a uniform random variable \( \gamma \in U(0, 1) \).

2. Find the exit time \( \tau \) by solving \( \bar{\mathcal{P}}_r(\tau) - \gamma = \bar{\mathcal{P}}_1 \left( \frac{T_{\text{inc}}}{r^2} \right) - \gamma = 0 \), where \( \bar{\mathcal{P}}_r \) is given by (7.75).

3. Simulate the exit point \( \xi = W(\tau) \), for which \( \mathbb{P}(\xi = r) = \mathbb{P}(\xi = -r) = \frac{1}{2} \). Both an exit point of \( r \) and \( -r \) are equally likely due to the symmetry of the problem.
(Note that we can do step 3 before steps 1 and 2.)

**Stepsize bounded by** $T_{\text{inc}}$

Here the procedure changes slightly, as if $\tau$ is simulated to be larger than $T_{\text{inc}}$ then we must set $\tau = T_{\text{inc}}$ and simulate the exit point $\xi$ using the conditional cdf $Q_r(\beta; T_{\text{inc}})$.

The procedure is:

1. Simulate $\gamma \sim U(0, 1)$.

2. Find the exit time $\tau$ by solving $\bar{P}_r(\tau) - \gamma = 0$ or $\bar{P}_1 \left( \frac{\xi}{\tau} \right) - \gamma = 0$.

3. If $\tau < T_{\text{inc}}$, then simulate $\xi = \pm r$ as before.

4. If $\tau \geq T_{\text{inc}}$, then
   
   (a) Set $\tau = T_{\text{inc}}$.
   
   (b) Simulate another uniform random variable $\gamma_2 \sim U(0, 1)$.
   
   (c) Find the exit point $\xi$ by solving $\bar{Q}_r(\xi; \tau) - \gamma_2 = \bar{Q}_1 \left( \frac{\xi}{\tau} \right) - \gamma_2 = 0$.

We can in fact modify this procedure further. We know that $P(\tau < T_{\text{inc}}) = \bar{P}_r(T_{\text{inc}})$ (simulated using $\bar{P}_r(T_{\text{inc}})$), and so we can simulate a random variable $\iota$ for which $P(\iota = 0) = \bar{P}_r(T_{\text{inc}})$ and $P(\iota = 1) = 1 - \bar{P}_r(T_{\text{inc}})$. We could use another $U(0, 1)$ random variable to do this; or we could use, for example, the NAG Toolbox for MATLAB routine $g05tb$, which simulates a logical variable with a probability $p = 1 - \bar{P}_r(T_{\text{inc}})$ of being true. If $\iota = 0$ or 'false' then we can simulate the exit point as in the unbounded stepsize case; however when simulating the exit time we much scale the random variable $\gamma$ used before by $\bar{P}_r(T_{\text{inc}})$ (since we cannot allow $\tau > T_{\text{inc}}$). The new procedure then reads:

1. Calculate $\bar{P}_r(T_{\text{inc}})$.

2. Simulate random variable $\iota$ with probability $1 - \bar{P}_r(T_{\text{inc}}) = 1 - \bar{P}_1 \left( \frac{T_{\text{inc}}}{\tau} \right)$ of being 1 or 'true', and probability $\bar{P}_r(T_{\text{inc}}) = \bar{P}_1 \left( \frac{T_{\text{inc}}}{\tau} \right)$ of being 0 or 'false'.

3. Simulate $\gamma \sim U(0, 1)$.

4. If $\iota = 0$ or 'false', then simulate $\xi = \pm r$ as before.
5. Find $\tau$ by solving $\mathcal{P}_r(\tau) - \mathcal{P}_r(T_{\text{inc}})\gamma = \mathcal{P}_1(T_{\text{inc}})\gamma = 0$.

6. If $\iota = 1$ or 'true' then
   (a) Set $\tau = T_{\text{inc}}$.
   (b) Find $\xi$ by solving $\mathcal{Q}_r(\xi; \tau) - \gamma = \mathcal{Q}_1(\xi; T_{\text{inc}})$.

7.5 Simulating the exit time of a multi-dimensional Wiener process

We would now like to simulate the exit time of an $m$-dimensional Wiener process $W(t)$, $W(0) = 0$ from a region $G \subset \mathbb{R}^m$. This time we shall set

$$G = \prod_{i=1}^m (-r_i, r_i),$$

for some choices $r_i$, $i = 1, \ldots, m$, and so

$$\partial G = \{x \in G \mid x_i = \pm r_i \text{ for some } i = 1, \ldots, m\}.$$ 

Let $\tau$ denote the first exit time of $W(t)$ from $G$. We now have the following result for the cdf of $\tau$:

**Lemma 7.4** For the vector $r \in \mathbb{R}^m$ with entries $r_i$, $i = 1, \ldots, m$, let $\mathcal{P}_{m,r}(T_{\text{inc}}) = \mathbb{P}(\tau < T_{\text{inc}})$ be the cdf of $\tau$ for an $m$-dimensional Wiener process $W(t)$. Then

$$\mathcal{P}_{m,r}(T_{\text{inc}}) = 1 - \prod_{i=1}^m (1 - \mathcal{P}_{r_i}(T_{\text{inc}})) = 1 - \prod_{i=1}^m \left(1 - \mathcal{P}_1(T_{\text{inc}})\right).$$  \hfill (7.89)

**Proof** The result can be obtained using the independence of the components $W_i(t)$ of $W(t)$ as follows. Let $\tau_i$ be the first exit time of the component $W_i(t)$ of $W(t)$ from $[-r_i, r_i]$. The $\tau_i$ are independent, $\tau = \min_{i=1,\ldots,m} \tau_i$ and

$$\mathbb{P}(\tau > T_{\text{inc}}) = 1 - \mathcal{P}_{m,r}(T_{\text{inc}}) = \mathbb{P}\left(\bigcap_{i=1}^m \tau_i > T_{\text{inc}}\right) = \prod_{i=1}^m \mathbb{P}(\tau_i > T_{\text{inc}}) \quad \text{by independence of } \tau_i$$

$$= \prod_{i=1}^m (1 - \mathcal{P}_{r_i}(T_{\text{inc}})).$$
Hence (7.89) follows.

We can also get the result of Lemma 7.4 by directly solving a BVP, as we did for the one-dimensional case.

**Deriving and solving the BVP for the problem**

We can get $W(t)$ from the Itô diffusion $X(t)$ in (7.4) by setting $f(x) = 0$ and $g(x) = I_m$, so the generator of $W(t)$ is

$$Af(x) = \frac{1}{2} \nabla^2 f(x). \quad (7.90)$$

Hence from (7.19)-(7.21), we must now solve the following BVP:

$$\frac{\partial u}{\partial t} = \frac{1}{2} \nabla^2 u, \quad t > 0, \quad x \in G,$$

$$u(0, x) = 0, \quad x \in G \cup \partial G,$$

$$u(t, x) = 1, \quad t > 0, \quad x \in \partial G.$$  

As before, the solution $u(t, x)$ has the probabilistic representation

$$u(t, x) = \mathbb{P}(\tau < t), \quad (7.91)$$

where $\tau$ is the first exit time of $W(t)$ from $G$. If we now set $v(t, x) = u(t, x) - 1$ we get the BVP

$$\frac{\partial v}{\partial t} = \frac{1}{2} \nabla^2 v, \quad t > 0, \quad x \in G,$$

$$v(0, x) = -1, \quad x \in G \cup \partial G,$$

$$v(t, x) = 0, \quad t > 0, \quad x \in \partial G. \quad (7.94)$$

Conveniently, this problem is separable; we can let $v(t, x) = \prod_{i=1}^m v_i(t, x_i)$, where each $v_i$ is a solution of

$$\frac{\partial v_i}{\partial t} = \frac{1}{2} \frac{\partial^2 v_i}{\partial x_i^2}, \quad x \in (-r_i, r_i), \quad (7.95)$$

$$v_i(0, x_i) = c_i, \quad x_i \in [-r_i, r_i], \quad (7.96)$$

$$v_i(t, -r_i) = v_i(t, r_i) = 0, \quad t > 0, \quad (7.97)$$

with $\prod_{i=1}^m c_i = -1$. We have the solutions of (7.95)-(7.97) already by slightly modifying the results in §7.2; they are
CHAPTER 7. SIMULATING A WIENER PROCESS

\[ v_i(t, x_i) = \frac{4c_i}{\pi} \sum_{k_i=0}^{\infty} \frac{(-1)^{k_i}}{2k_i + 1} \cos \left( \left( \frac{2k_i + 1}{2r_i} \right) \pi x_i \right) \exp \left( -\frac{(2k_i + 1)^2 \pi^2}{8r_i^2} t \right), \quad (7.98) \]

\[ v_i(t, x_i) = c_i \int_{-r_i}^{r_i} K_i(x_i', 0; x_i, t) \, dx_i', \quad (7.99) \]

where

\[ K_i(x_i, 0; y_i, s) = \frac{1}{\sqrt{2\pi s}} \exp \left( -\frac{(x_i - y_i)^2}{2s} \right) + \frac{1}{\sqrt{2\pi s}} \sum_{k_i=1}^{\infty} \exp \left( -\frac{(x_i - 2k_i r_i - (-1)^k y_i)^2}{2s} \right) + \exp \left( -\frac{(x_i + 2k_i r_i - (-1)^k y_i)^2}{2s} \right). \quad (7.100) \]

If we now set \( x = 0 \) and \( t = T_{\text{inc}} \), we again modify the working from section 7.2 and the solutions for \( v_i \) become

\[ v_i(0, T_{\text{inc}}) = \frac{4c_i}{\pi} \sum_{k_i=0}^{\infty} \frac{(-1)^{k_i}}{2k_i + 1} \cos \left( \left( \frac{2k_i + 1}{2r_i} \right) \pi x_i \right) \exp \left( -\frac{(2k_i + 1)^2 \pi^2}{8r_i^2} T_{\text{inc}} \right), \quad (7.101) \]

\[ v_i(0, T_{\text{inc}}) = c_i \left[ 1 - 2 \sum_{k_i=0}^{\infty} (-1)^{k_i} \text{erfc} \left( \frac{(2k_i + 1)r_i}{\sqrt{2T_{\text{inc}}}} \right) \right]. \quad (7.102) \]

Now let \( P_{m,r}(T_{\text{inc}}) = \mathbb{P}(\tau < T_{\text{inc}}) = u(T_{\text{inc}}, 0), \) where \( r \) has elements \( r_i \) for \( i = 1, \ldots, m \).

Since \( u(T_{\text{inc}}, 0) = 1 + v(T_{\text{inc}}, 0) = 1 + \prod_{i=1}^{m} v_i(t, 0) \), and \( \prod_{i=1}^{m} c_i = -1 \), we obtain the following two expressions for \( P_{m,r}(T_{\text{inc}}) \):

\[ P_{m,r}(T_{\text{inc}}) = 1 - \prod_{i=1}^{m} \frac{4}{\pi} \sum_{k_i=0}^{\infty} \frac{(-1)^{k_i}}{2k_i + 1} \exp \left( -\frac{(2k_i + 1)^2 \pi^2}{8r_i^2} T_{\text{inc}} \right), \quad (7.103) \]

\[ P_{m,r}(T_{\text{inc}}) = 1 - \prod_{i=1}^{m} \left[ 1 - 2 \sum_{k_i=0}^{\infty} (-1)^{k_i} \text{erfc} \left( \frac{(2k_i + 1)r_i}{\sqrt{2T_{\text{inc}}}} \right) \right]. \quad (7.104) \]

Note that if we set \( m = 1 \) in the above, we recover (7.66) and (7.67) as expected.

We can prove Lemma 7.4 simply by using (7.66), (7.67), (7.103) and (7.104).

**Simulating the exit time \( \tau \)**

We can simulate the exit time \( \tau \) by simulating the exit times \( \tau_i \) of \( W_i(t) \) from \([-r_i, r_i]\) separately, by solving

\[ \mathcal{P}_{r_i}(\tau_i) - \gamma_i = 0 \]
for independent $U(0, 1)$ random variables $\gamma_i$ using a root-finding method, and then taking $\tau = \min_{i=1,\ldots,m}(\tau_i)$. If we let $j$ be the index for which the minimum is attained, then we know that $W_j(\tau) = \pm r_j$ and it remains to find the values of $W_i(\tau)$ for $i \neq j$.

### 7.5.1 Simulating the exit point of a multi-dimensional Wiener process if $\tau = T_{\text{inc}}$

This time we are interested in what happens if the $m$-dimensional Wiener process $W(t)$ does not exit $G = \prod_{i=1}^{m}(-r_i, r_i)$ at a time $\tau < T_{\text{inc}}$; in this case $\tau = T_{\text{inc}}$. By independence of the components $W_i(t)$ of $W(t)$, the conditional cdf of the exit point $\xi_i = W_i(\tau)$ is

\[ Q_{r_i}(\beta_i; T_{\text{inc}}) = \mathbb{P}(\xi_i < \beta_i \mid \tau = T_{\text{inc}}), \quad (7.105) \]

for $i = 1, \ldots, m$. Hence the joint conditional cdf of the exit point $\xi = W(\tau)$ can be written as

\[ Q_{m,r}(\beta; T_{\text{inc}}) = \mathbb{P}(\cap_{i=1}^{m}(\xi_i < \beta_i \mid \tau = T_{\text{inc}})) \]

\[ = \prod_{i=1}^{m} \mathbb{P}(\xi_i < \beta_i \mid \tau = T_{\text{inc}}) \quad \text{by independence} \]

\[ = \prod_{i=1}^{m} Q_{r_i}(\beta_i; T_{\text{inc}}). \quad (7.106) \]

We can demonstrate this by deriving and solving the BVP associated with this problem. A more rigorous justification for this is also provided in [57, Theorem 5.1] for the case where all of the $r_i$ are equal.

### Deriving and solving the BVP for the problem

Let $Q_{m,r}^x$ be the conditional cdf for the exit point $\xi$ of $W(t)$ with the initial value $W(0) = x_0$; $Q_{m,r}^x$ is given by

\[ Q_{m,r}^x(\beta; T_{\text{inc}}) = \mathbb{P}\left( \xi \in \prod_{i=1}^{m}[-r_i, \beta_i] \mid \tau = T_{\text{inc}} \right) = \frac{\mathbb{P}(\tau \geq T_{\text{inc}}, \cap_{i=1}^{m}(\xi_i < \beta_i))}{\mathbb{P}(\tau \geq T_{\text{inc})}}, \quad (7.107) \]
since the events $\xi \in \prod_{i=1}^m [-r_i, \beta_i]$ and $\cap_{i=1}^m (\xi < \beta_i)$ are equivalent. The theory in §7.1 states that the solution $u(T_{inc}, x_0)$ of the PDE problem

$$\frac{\partial u}{\partial t} = \frac{1}{2} \nabla^2 u, \quad t > 0, \ x \in G,$$

(7.108)

$$u(0, x) = 1_{x \in \prod_{i=1}^m [-r_i, \beta_i]} = \prod_{i=1}^m 1_{x_i \in [-r_i, \beta_i]}, \quad x_0 \in G \cup \partial G,$$

(7.109)

$$u(t, x) = 0, \quad t > 0, \ x \in \partial G,$$

(7.110)

has the probabilistic representation

$$u(t, x) = \mathbb{P} \left( \tau \geq t, \ x \in \prod_{i=1}^m [-r_i, \beta_i] \right),$$

(7.111)

and hence

$$Q_{m, r}(\beta; t) = \frac{u(t, x)}{1 - \mathbb{P}(\tau < T_{inc})}.$$

(7.112)

Again we are interested in the case $x = 0$ and $t = T_{inc}$, where

$$Q_{m, r}(\beta; T_{inc}) = \frac{u(T_{inc}, 0)}{1 - \mathbb{P}_{m, r}(T_{inc})}.$$  

(7.113)

The problem (7.108)-(7.110) is again separable. Let $u(t, x) = \prod_{i=1}^m u_i(t, x_i)$, where $u_i(t, x_i)$ solves

$$\frac{\partial u}{\partial t} = \frac{1}{2} \partial^2 u \quad t > 0, \ x \in (-r_i, r_i),$$

(7.114)

$$u(0, x_i) = 1_{x_0 = i \in [-r_i, \beta_i]}, \quad x_i \in [-r_i, r_i],$$

(7.115)

$$u(t, -r_i) = u(t, r_i) = 0, \quad t > 0.$$  

(7.116)

This is exactly the problem (7.77)-(7.79) and hence has the same solution. Since $1 - \mathbb{P}_{m, r}(T_{inc}) = \prod_{i=1}^m (1 - \mathbb{P}_{r_i}(T_{inc}))$ by (7.89), we obtain

$$Q_{m, r}(\beta; T_{inc}) = \prod_{i=1}^m \frac{u_i(T_{inc}, 0)}{1 - \mathbb{P}_{r_i}(T_{inc})} = \prod_{i=1}^m Q_{r_i}(\beta_i; T_{inc}).$$

(7.117)

### 7.5.2 Simulating the exit point of the multi-dimensional Wiener process if $\tau < T_{inc}$

We are so far able to simulate the exit time of the $m$-dimensional Wiener process from $G$, and find the point $W(\tau)$ if $\tau = T_{inc}$. However, we still need to find the point $W(\tau)$ for the case $\tau < T_{inc}$.
We have already found that $\tau = \min_{i=1,\ldots,m}(\tau_i)$, where $\tau_i$ is the exit time of $W_i(t)$ from $[-r_i,r_i]$, and we have also found the index of the component $j$ of $W(t)$ which exits first (so $\tau = \tau_j$). We can then simulate the exit point $\xi_j = W_j(\tau)$ easily, since $P(\xi_j = \pm r_j) = \frac{1}{2}$, and now we must simulate the other exit points $\xi_i = W_i(\tau)$, for $i \neq j$.

The cdf of $\xi_i$ conditioned on $\tau$ is just $P(\xi_i < \beta_i \mid \tau_i \geq \tau) = Q_{r_i}(\beta_i; \tau_j)$, since all of the components of $W(t)$ are independent; simulating $\xi_i$ is analogous to simulating the exit point of a one-dimensional Wiener process, given that it does not exit the interval $[-r_i,r_i]$ before time $\tau$. We also know that if $W(t)$ does not exit $G$ in a time $\tau < T_{inc}$, the conditional cdf of each $\xi_i$ is $Q_{r_i}(\beta_i; T_{inc})$.

A more detailed justification for this can be found in [57, §4] for the case where all of the $r_i$ are equal. If the $r_i$ are not equal it is possible to simulate a bounded Wiener process where all of the $r_i$ are equal, and use a linear transformation to produce the original bounded Wiener process.

### 7.5.3 The final algorithm for simulating the bounded Wiener process

We now have the following algorithm for simulating the bounded Wiener process:

1. Calculate the vector $p$ of length $m$ with entries $\mathcal{P}_{r_i}(T_{inc})$, for $i = 1, \ldots, m$.
2. Simulate a logical vector $\iota$ of length $m$ with the $i$-th entry $\iota_i$ having probability $1 - p_i$ of being true and probability $p_i$ of being false.
3. Simulate a random vector $\gamma$ of length $m$ with entries $\gamma_i \sim U(0, 1)$.
4. Now simulate the exit times $\tau_i$ of $W_i(t)$ from $[-r_i,r_i]$ as follows:
   
   (a) If $\iota_i$ is true, then $\tau_i = T_{inc}$;
   
   (b) If $\iota_i$ is false, then $\tau_i$ is the solution of $\mathcal{P}_{r_i}(\tau_i) - \mathcal{P}_{r_i}(T_{inc})\gamma_i = 0$, found using a root-finding method.
5. Find $\tau = \min_{i=1,\ldots,m}(\tau_i, T_{inc})$ and, if $\tau < T_{inc}$, find the component $j$ for which the minimum is attained - this is the component of $W(t)$ which exits first (if $\tau = T_{inc}$, none of the components $W_i(t)$ exit $[-r_i,r_i]$).
6. If $\tau < T_{\text{inc}}$, then for the component $j$ which exits first, simulate $\xi_j = W_j(\tau)$ for which $P(\xi_j = r_j) = P(\xi_j = -r_j) = \frac{1}{2}$.

7. Simulate another random vector $\eta$ of length $m$ with entries $\eta_i \sim U(0, 1)$.

8. Find the exit points $\xi_i = W_i(\tau)$, for all $i \neq j$ if $\tau < T_{\text{inc}}$ and $\xi_j$ has already been simulated, or for all $i$ if $\tau = T_{\text{inc}}$. This is done by solving $\bar{Q}_r(\xi; \tau) - \eta_i = \bar{Q}_1\left(\frac{\xi_i}{\tau}; \tau\right) - \eta_i = 0$.

**A more efficient algorithm for simulating the bounded Wiener process**

Our algorithm for simulating the bounded Wiener process requires the simulation of $3m$ random variables. A more efficient algorithm, which requires only $m + 3$ random variables, is outlined in [57, §5], for the case where all of the $r_i$ are equal for $i = 1, \ldots, m$. This can be extended to the case where the $r_i$ are different by simulating a Wiener process for which all of the $r_i$ are equal, and then applying a linear transformation.

7.6 Using bounded increments to generate a path of the Wiener process

We can use bounded Wiener increments to generate a path of a Wiener process $W(t)$, which can in turn be used along with a numerical scheme to solve the SDE (7.4) over an interval $[0, T]$. For practical purposes, we choose to bound each stepsize $\Delta t_n$ so that we do not overshoot the final time $t$. We could choose to leave some (or all) of the Wiener increments $\Delta W_n$ unbounded (this results in using deterministic stepsizes).

The first Wiener increment $\Delta W_0 = W(t_1)$ and first stepsize $\Delta t_0 = t_1$ are simulated as described earlier, for some maximum stepsize $\Delta t_{\text{max}}^0$ and some (optional) maximum increment vector $\Delta W_{\text{max}}^0$. Then, for $n \geq 1$, successive Wiener increments $\Delta W_n$ and stepsizes $\Delta t_n$ are simulated in the same way, by pretending that the $\Delta W_n$ are independent Wiener processes, for some maximum stepsize $\Delta t_{\text{max}}^n$ and some (optional) maximum increment vector $\Delta W_{\text{max}}^n$. The stepsizes $\Delta t_n$ will form a partition $D \in D([0, T])$, whose size $N$ will not be known in advance (this is also the case for many of the methods we studied in Chapter 6).

Note that we cannot reject any stepsizes if we only use bounded Wiener increments.
to generate the Wiener path. We could choose to reject a stepsize and use the Brownian bridge to produce a smaller stepsize; however the new time point generated will no longer be a stopping time.
Chapter 8

Using Bounded Wiener Increments
in an Adaptive Stepsize Algorithm
to Solve SDEs

In Chapter 7 we studied how to simulate bounded Wiener increments, and we discussed how to use bounded Wiener increments to generate a path of an $m$-dimensional Wiener process $W(t)$ for $t \in [0, T]$, along with a partition $D \in D([0, T])$ consisting of stopping times ($D([0, T])$ is defined in §5.1.7). We would now like to use such a Wiener path, along with a numerical scheme $S$, to solve an SDE.

We have several options as to which bounds to use for the Wiener increments and the stepsizes. One such option is to set all the $\Delta t_n^{\text{max}}$ and $\Delta W_n^{\text{max}}$ to have some fixed values $\Delta t^{\text{max}}$ and $\Delta W^{\text{max}}$ for all $n \geq 0$ (although the final stepsize bound $\Delta t_{N-1}^{\text{max}}$ may need to be less than $\Delta t^{\text{max}}$ so that we do not overshoot the end time $T$). Alternatively we can adjust $\Delta t_n^{\text{max}}$ and $\Delta W_n^{\text{max}}$ for each step in order to try and reduce the global error, while ensuring that we maintain efficiency by not using unnecessarily many steps. This chapter is aimed towards finding such an error control strategy.

We shall find bounds for the pathwise global error of numerical schemes, in particular for the case where the partition $D$ consists of stopping times. Later we shall propose a variable stepsize strategy, using bounded Wiener increments with error control, aimed at reducing the global error of the Euler–Maruyama scheme. Finally we shall perform some numerical experiments to see how the method performs in practice.

Some of the results in this chapter are similar to results in [15], where differential
equations driven by general rough paths are discussed (for further discussion of this class of equations, see for example [15, 25, 47]). General rough path theory includes use of a control function $\omega(t)$ to express the regularity of the rough path; however the control function $\omega(t) = t$ is a control function for the Wiener process $W(t)$.

8.1 Background

To aid in our analysis, we introduce a Banach space of continuous functions with uniformly bounded derivatives. We shall then introduce the SDE problem and then some additional notation which shall be used in our analysis.

Definition 8.1 The Banach space $C^k_b(\mathbb{R}^d, \mathbb{R}^d)$, for $k \in \mathbb{N}$, is the set of bounded functions $f : \mathbb{R}^d \to \mathbb{R}^d$ with $k$ continuous and uniformly bounded derivatives. We shall define

$$
\| f \|_{C^k_b} = \sup_{|\alpha| \leq k} \left\| \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \right\|_\infty,
$$

where $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index with entries $\alpha_i \in \mathbb{Z}_+$, for $i = 1, \ldots, d$, and $|\alpha| = \sum_{i=1}^d \alpha_i$.

The SDE problem

For some $d \geq 1$ and an $m$-dimensional $\mathcal{F}_t$-Wiener process $W(t)$ for some $m \geq 1$, we look at solving the $\mathbb{R}^d$-valued Itô SDE

$$
dX(t) = f(X(t)) \, dt + \sum_{j=1}^m g_j(X(t)) \, dW_j(t), \quad X(0) = x_0,
$$

(8.1)

where $W_j(t)$ are the (iid) components of $W(t)$. For convenience, we use the conventions $f(x) = g_0(x)$ and $W_0(t) = t$, and rewrite (8.1) as

$$
dX(t) = \sum_{j=0}^m g_j(X(t)) \, dW_j(t), \quad X(0) = x_0.
$$

(8.2)

We shall use the following set of assumptions to prove results in this chapter.

Assumption 8.2 The functions $g_j : \mathbb{R}^d \to \mathbb{R}^d$, for $j = 0, \ldots, m$, satisfy the following:

- They belong to the Banach space $C^3_b(\mathbb{R}^d, \mathbb{R}^d)$.
• They satisfy the linear growth condition in Definition 5.3;
• They satisfy the global Lipschitz condition in Definition 5.4.

Hence the solution of (8.2) is unique by Theorem 5.5.

Additional notation

For \( s \in [0, T] \) and \( z \in \mathbb{R}^d \), let \( Y_{s,z}(t) \) be the solution of

\[
dY_{s,z}(t) = \sum_{j=0}^{m} g_j(Y_{s,z}(t)) \, dW_j(t), \quad Y_{s,z}(s) = z, \quad (8.3)
\]

for \( t \in [s, T] \). Note that, for any \( s' \in [s, t] \),

\[
Y_{s,z}(t) = Y_{s',Y_{s,z}(s')}(t), \quad (8.4)
\]

due to the Markovian property of the solution of (8.2).

Let \( X_n \) denote the numerical approximation to \( X(t_n) \) \( (= Y_{0,x_0}(t_n)) \) obtained using the scheme \( S \) and the partition \( D \). We shall denote by \( S_\gamma \) the set of numerical schemes \( S \) with LTE satisfying

\[
\|T_n^S(z)\|_2 \leq K_{LTE}(\omega) \Delta t_n^{\gamma + \frac{1}{2} - \epsilon} \quad (8.5)
\]

almost surely, for any \( 0 < \epsilon < \frac{1}{2} \) and \( 0 \leq n \leq N - 1 \), where \( K_{LTE} \) depends on \( \epsilon \). If (8.5) is satisfied we say that the LTE has order \( \gamma + \frac{1}{2} \). For example, the Euler–Maruyama scheme EM is in \( S_{\frac{1}{2}} \) and has LTE of order 1, and the Milstein scheme Mil is in \( S_1 \) and has LTE of order \( \frac{3}{2} \), as proved in Corollaries 5.27 and 5.28 respectively. For \( n \geq k \) we shall also write

\[
\Delta t_{kn} = t_n - t_k, \quad (8.6)
\]
\[
\Delta W_{j,n} = W_j(t_n) - W_j(t_k), \quad (8.7)
\]
\[
\Delta W_{0,n} = \Delta t_{kn}. \quad (8.8)
\]

In order to facilitate the discussion in this chapter, we shall add more sets of partitions to the ones already defined in §5.1.7. Firstly, \( D_{\text{stop}}([0, T]) \) shall denote the set of partitions such that for all partitions \( D \in D_{\text{stop}}([0, T]) \), the times \( t_n \in D \) are all stopping times, and \( D_{\Delta t_{\text{max}}}^{\text{stop}}([0, T]) = D_{\text{stop}}([0, T]) \cap D_{\Delta t_{\text{max}}}([0, T]) \) shall denote the set of partitions \( D \in D_{\text{stop}}([0, T]) \) which have maximum stepsize \( \Delta t_{\text{max}} \).
Since deterministic times are stopping times, any partition \( D \) where the times \( t_n \in D \) are deterministic is in \( \mathcal{D}^{\text{stop}}([0,T]) \). This means that, for example, \( D_{\Delta t} \in \mathcal{D}^{\text{stop}}([0,T]) \) (and \( D_{\Delta t} \in \mathcal{D}_{\Delta t}^{\text{stop}}([0,T]) \)) for any \( \Delta t \). However, in §6.2.1 we came across a partition where not all of the times in \( D \) are stopping times. For \( \Delta t > 0 \) we took the partition \( D_{\Delta t} \) and, for a choice \( \lambda \geq 0 \), we checked whether \( |\Delta W_n| \leq \lambda \sqrt{\Delta t} \), and if not we added a new time point \( t_{\text{new}} = \frac{t_n + t_{n+1}}{2} \). Since \( W(t_{\text{new}}) \) needs to be calculated using the Brownian bridge, \( t_{\text{new}} \) is not a stopping time.

### 8.1.1 Pathwise global error of order \( \gamma - \frac{1}{2} \) for numerical schemes in \( \mathcal{S}_\gamma \)

We now state and prove a result due to Gaines and Lyons [26], which proves that the pathwise global error (PGE) for any scheme \( S \in \mathcal{S}_\gamma \), where \( \gamma \geq 1 \), has order \( \gamma - \frac{1}{2} \). We defined the pathwise global error (PGE) and its order in Definition 5.19. We shall assume throughout this section that we are solving (8.2) on the finite interval \([0,T]\), for some \( T > 0 \), using the scheme \( S \) and a partition \( D \in \mathcal{D}([0,T]) \) of size \( N \), and that Assumption 8.2 applies.

In order to prove the main result of this section, we also need a result on the solution of the SDE.

**Proposition 8.3** Let \( \mathbf{Y}_{s,\mathbf{z}}(t) \) and \( \mathbf{Y}_{s,\mathbf{z}'}(t) \) be solutions of (8.3) for the same path of \( \mathbf{W}(t) \) and different initial conditions \( \mathbf{z}, \mathbf{z}' \in \mathbb{R}^d \) and \( t \in [s,T] \), for some \( T > 0 \). Then

\[
\| \mathbf{Y}_{s,\mathbf{z}}(t) - \mathbf{Y}_{s,\mathbf{z}'}(t) \|_2 \leq K_{\text{SDE}}(\omega) \| \mathbf{z} - \mathbf{z}' \|_2 ,
\]

almost surely, where \( K_{\text{SDE}}(\omega) \) is a random variable depending only on the SDE itself (i.e. the functions \( g_j(x) \) for \( j = 0, \ldots, m \)) and the path of \( \mathbf{W}(t) \).

**Proof** See [26, Lemma 4.1], [25, Theorem 10.26]. \( \square \)

**Theorem 8.4** (Global error of order \( \gamma - \frac{1}{2} \)) Let \( g_j \in C^3_b(\mathbb{R}^d, \mathbb{R}^d) \) for \( j = 0, \ldots, m \), let \( D \in \mathcal{D}_{\Delta t_{\text{max}}}( [0,T] ) \) be a partition for some \( \Delta t_{\text{max}} > 0 \), and let \( S \in \mathcal{S}_\gamma \) be a numerical
scheme for some $\gamma \geq 1$. Now suppose that Proposition 8.3 holds for the SDE (8.2). Then, for any $k \geq 1$ and any $0 < \epsilon < \frac{1}{2}$,

$$
\|X_n - X(t_n)\|_2 \leq K_{\text{SDE}}K_{\text{LTE}}t_n\Delta t_{\text{max}}^{\gamma - \frac{1}{2} - \epsilon}.
$$

(8.10)

In particular, by setting $k = N$ we obtain

$$
\|X_N - X(T)\|_2 \leq K_{\text{SDE}}K_{\text{LTE}}T\Delta t_{\text{max}}^{\gamma - \frac{1}{2} - \epsilon}.
$$

(8.11)

**Proof** By taking norms in (5.45) and using (8.5),

$$
\|T_k^n\|_2 = \|Y_{t_k}x_k(t_{k+1}) - X_{k+1}\|_2 \leq K_{\text{LTE}}t_k^{\frac{1}{2} - \epsilon}
$$

almost surely, for all $0 \leq k < n$ and some random variable $K_{\text{LTE}}$. Now, by Proposition 8.3, there exists some constant $K_{\text{SDE}}$ such that

$$
\|Y_{t_k}x_k(t_{k+1}) - Y_{t_{k+1}}x_{k+1}(t_{k+1})\|_2 \leq K_{\text{SDE}}\|Y_{t_k}x_k(t_{k+1}) - X_{k+1}\|_2 \text{ by (8.4)}
$$

$$
\leq K_{\text{LTE}}\|Y_{t_k}x_k(t_{k+1}) - X_{k+1}\|_2 \text{ by (8.5)}
$$

$$
\leq K_{\text{LTE}}K_{\text{SDE}}\Delta t_k^{\gamma - \frac{1}{2} - \epsilon},
$$

(8.12)

since $Y_{t_{k+1}}x_{k+1}(t_{k+1}) = X_{k+1}$. If we now sum both sides of (8.12) over $k$ from $k = 0$ to $k = n - 1$, the triangle inequality applies and we obtain

$$
\|X(t_n) - X_n\|_2 = \|Y_{0,x_0}(t_n) - Y_{t_n}x_n(t_n)\|_2
$$

$$
\leq \sum_{k=0}^{n-1} \|Y_{t_k}x_k(t_n) - Y_{t_{k+1}}x_{k+1}(t_{k+1})\|_2
$$

$$
\leq K_{\text{LTE}}K_{\text{SDE}}\sum_{k=0}^{n-1} \Delta t_k^{\gamma - \frac{1}{2} - \epsilon},
$$

since $Y_{0,x_0}(t_n) = X(t_n)$ and $Y_{t_k}x_k(t_k) = X_n$. Since $\gamma \geq 1$ and $0 < \epsilon < \frac{1}{2}$, $\Delta t_k^{\gamma - \frac{1}{2} - \epsilon} \leq \Delta t_{\text{max}}^{\gamma - \frac{1}{2} - \epsilon}$. Hence

$$
\|X(t_n) - X_n\|_2 \leq K_{\text{LTE}}K_{\text{SDE}}\sum_{k=0}^{n-1} \Delta t_k \Delta t_k^{\gamma - \frac{1}{2} - \epsilon}
$$

$$
\leq K_{\text{LTE}}K_{\text{SDE}}\sum_{k=0}^{n-1} \Delta t_k \Delta t_{\text{max}}^{\gamma - \frac{1}{2} - \epsilon}
$$

$$
\leq K_{\text{LTE}}K_{\text{SDE}}\Delta t_{\text{max}}^{\gamma - \frac{1}{2} - \epsilon} \sum_{k=0}^{n-1} \Delta t_k \leq K_{\text{LTE}}K_{\text{SDE}}t_n \Delta t_{\text{max}}^{\gamma - \frac{1}{2} - \epsilon}.
$$
Note that Theorem 8.4 cannot be applied to any numerical scheme of pathwise order $\frac{1}{2}$, including the Euler–Maruyama scheme.

### 8.1.2 Pathwise global error of order $\gamma$ for numerical schemes in $S_\gamma$

We obtained a PGE bound of order $\gamma - \frac{1}{2}$ in the previous section, for any scheme $S \in S_\gamma$ and any partition $D \in D_{\Delta t_{\text{max}}}([0, T])$. However, we know that schemes $S \in S_\gamma$ have a strong order of $\gamma$, and we would like to show that the scheme $S$ also has a pathwise global error of order $\gamma$. In [73], we claim that if the partition $D \in D_{\Delta t_{\text{max}}}^{\text{step}}([0, T])$, the pathwise global error of the scheme $S$ is of order $\gamma$. Several of our results in [73] are similar to those found in [15], and we also make use of Corollary 5.33 in order to bound the sums of consecutive local truncation errors. We also propose a new variable stepsize method for solving SDEs, which we now describe.

### 8.2 The Bounded Increments with Error Control (BIEC) Method

We describe an algorithm for solving the SDE (8.1) which uses bounded Wiener increments, as described in Chapter 7, to produce the stepsizes and Wiener increments for each step. An error control is used to decide what the maximum stepsize and maximum Wiener increment will be at each step, so the algorithm shall be called Bounded Increments with Error Control (BIEC).

Let $\Delta t_{\text{max}} = \frac{T}{N}$ for some integer $N \geq T$ (again to ensure that $\Delta t_{\text{max}} \leq 1$), and suppose we have taken $n$ stepsizes and found a numerical approximation $X_n$ at a time $t_n \in [0, T]$. To find the size of the $(n+1)$-th stepsize $\Delta t_n = t_{n+1} - t_n$, we choose $\Delta t_n$ to be the largest stepsize such that $\Delta t_n \leq \min(\Delta t_{\text{max}}, T - t_n)$ (to avoid overshooting the final time $T$) and

$$\max_{r=0, \ldots, m} \|F_{(0,r)}(X_n)\|_2 \Delta t_n \leq \alpha \frac{T}{N} = \alpha \Delta t_{\text{max}},$$

$$\max_{r=0, \ldots, m} \|F_{(j,r)}(X_n)\|_2 |\Delta W^n_j| \leq \alpha^{\frac{1}{2}} \frac{T^{\frac{1}{2}}}{N^{\frac{1}{2}}} = \alpha^{\frac{1}{2}} \Delta t_{\text{max}}^{\frac{1}{2}} \quad j = 1, \ldots, m.$$
for some parameter $\alpha$ which the user can set. This is done by choosing a maximum stepsize $\Delta t_{\text{max}}^n$ and a maximum Wiener increment vector $\Delta W_{\text{max}}^n$ using

$$
\Delta t_{\text{max}}^n = \min \left( \alpha \frac{\alpha}{\max_{r=0,\ldots,m} \| \mathbf{F}^{\text{id}}_{(0,r)}(\mathbf{X}_n) \|_2}, 1 \right) \Delta t_{\text{max}},
$$

(8.15)

$$
\Delta W_{\text{max}}^n, j = \frac{\alpha^{\frac{1}{2}}}{\max_{r=0,\ldots,m} \| \mathbf{F}^{\text{id}}_{(j,r)}(\mathbf{X}_n) \|_2} \Delta t_{\text{max}}^{\frac{1}{2}}, \quad j = 1, \ldots, m.
$$

(8.16)

In the special case where $\Delta t_{\text{max}}^n$ is larger than $T - t_n$, we set $\Delta t_{\text{max}}^n = T - t_n$ to avoid overshooting the final time $T$. The final stepsize $\Delta t_n$ and Wiener increment $\Delta W_n$ form the exit point of $[t, \mathbf{W}'(t)]^T$ from the domain $[0, \Delta t_{\text{max}}^n) \times G$, where

$$
G = \prod_{j=1}^m (-\Delta W_{\text{max}}^n, j, \Delta W_{\text{max}}^n, j),
$$

(8.17)

and the algorithm described in §7.5.3 is used to simulate $\Delta t_n$ and $\Delta W_n$. A partition $D \in \mathcal{D}_{\Delta t_{\text{max}}^n}([0, T])$ will be formed once all of the stepsizes and Wiener increments have been produced. It is possible for $\Delta W_{\text{max}}^n, j$ to be infinite for some $j$ if $\mathbf{F}^{\text{id}}_{(j,r)}(\mathbf{X}_n) = 0$ for all $r = 0, \ldots, m$, but in practice we can bound $\Delta W_{\text{max}}^n, j$ by some maximum. For the numerical experiments in §8.3, we bound $\Delta W_{\text{max}}^n, j$ by 10 (we do not expect any of the $|\Delta W_j^n|$ to attain this value for any of the maximum stepsizes considered!).

We expect the BIEC algorithm to be effective whether the SDE (8.1) has large noise or small noise. If (8.1) has small noise then we expect $\Delta W_{\text{max}}^n, j$ to be large in comparison to $\Delta t_{\text{max}}^n$, for $j = 1, \ldots, m$, so stepsizes are more likely to be chosen based on (8.13) alone and comparatively large Wiener increments are allowed. If (8.1) has large noise then we expect $\Delta W_{\text{max}}^n, j$ to be small for each $1, \ldots, m$, (8.14) is more likely to determine the stepsize and hence we do not allow large Wiener increments.

Pathwise global error of order $\frac{1}{2}$ for the BIEC method In [73], we claim that the BIEC method has a pathwise global error of order $\frac{1}{2}$. In particular we claim that, for any $0 < \epsilon < \frac{1}{2}$ and all $p \geq 1$, there exists a random variable $K_{\text{BIEC}}$ such that

$$
\sup_{0 \leq n \leq N} \| \mathbf{X}_n - \mathbf{X}(t_n) \|_2 \leq K_{\text{BIEC}} \Delta t_{\text{max}}^{\frac{1}{2} - \epsilon}
$$

almost surely, where the partition $D \in \mathcal{D}_{\Delta t_{\text{max}}^n}([0, T])$ produced by the method is of size $N$. We note that this does not automatically guarantee strong convergence; however we expect to see strong convergence with an error of order $\frac{1}{2}$.
8.3 Numerical results

We present several numerical examples to see how the Bounded Increments with Error Control (BIEC) method proposed in §8.2 performs in practice. We shall compare the BIEC method with four other numerical methods for three test problems, and focus on how each method controls the strong error and the efficiency of each method. Here we shall say that one method is more efficient than another if fewer steps are required to get the same strong error, rather than having a smaller numerical cost or taking a smaller amount of time for the same strong error. This is due to the fact that at present, the generation of bounded Wiener increments is a complex procedure and so the BIEC method (and the BIEC2, BIEC3 and BI methods described below) are a lot slower than using the Euler–Maruyama scheme with a fixed stepsize.

8.3.1 The numerical methods used

We shall introduce the numerical methods against which the BIEC method will be compared. The first is the BI method, and the other two are the BIEC2 and BIEC3 methods, which are small modifications of the BIEC method. We shall refer to the BIEC, BIEC2 and BIEC3 methods collectively as the BIEC methods.

The Bounded Increments (BI) method

The Bounded Increments (BI) method uses the Euler–Maruyama scheme to advance the solution, and uses bounded increments as described in Chapter 7 to determine the stepsizes. However, the maximum stepsize \( \Delta t_{\text{max}} \) and maximum Wiener increment vector \( \Delta \mathbf{W}_{\text{max}} \) are fixed and, for \( j = 1, \ldots, m \),

\[
\Delta \mathbf{W}_{\text{max},j} = \left( \frac{\Delta t_{\text{max}}}{\rho} \right)^{\frac{1}{2}},
\]

for some fixed constant \( \rho \). The smaller the value of \( \rho \), the more likely it is that the stepsizes \( \Delta t_n \) will be less than \( \Delta t_{\text{max}} \) - from (7.89),

\[
\mathbb{P}(\Delta t_n < \Delta t_{\text{max}}) = \mathcal{P}_{m,\Delta \mathbf{W}_{\text{max}}}(\Delta t_{\text{max}})
\]

\[
= 1 - \prod_{j=1}^{m} \left( 1 - \mathcal{P}_{\Delta \mathbf{W}_{\text{max},j}}(\Delta t_{\text{max}}) \right)
\]

\[
= 1 - (1 - \mathcal{P}_{1}(\rho))^{m}.
\]
Unlike the BIEC methods, the BI method does not take the SDE into account when choosing stepsizes.

The second Bounded Increments with Error Control (BIEC2) method

The second Bounded Increments with Error Control (BIEC2) method is identical to the BIEC method, except that we omit the terms in the error control which have second derivatives of the drift and diffusion functions. Specifically, \((8.13)\) is replaced with

\[
\max_{r=0,\ldots,m} \left\| F_{(0,r)}^{\text{mod}}(X_n) \right\|_2 \Delta t_n \leq \alpha T \frac{\Delta t_{\text{max}}}{N} = \alpha \Delta t_{\text{max}},
\]

where

\[
F_{(0,r)}^{\text{mod}}(x) = \sum_{i=1}^{d} g_{0i}(x) \frac{\partial g_{r}(x)}{\partial x_i}.
\]

If all of the second derivatives of \(g_j\) are zero for all \(j = 0,\ldots,m\), then the BIEC2 method is the same as the BIEC method.

The third Bounded Increments with Error Control (BIEC3) method

The third Bounded Increments with Error Control (BIEC3) method is again identical to the BIEC method but with a different modification. The terms \(\left\| F_{(j,r)}^{\text{id}}(X_n) \right\|_2\) in \((8.14)\) are replaced with their square roots, so \((8.14)\) becomes

\[
\max_{r=0,\ldots,m} \left\| F_{(j,r)}^{\text{id}}(X_n) \right\|_2 \left| \Delta W_n^j \right| \leq \alpha \frac{T^\frac{1}{2}}{N^\frac{1}{2}} = \alpha \Delta t_{\text{max}}^\frac{1}{2} \quad j = 1,\ldots,m.
\]

8.3.2 The numerical results

We now present numerical results for three test problems. In each case, we examine the strong error (as defined in \((5.17)\)) and the mean number of steps taken over 1000 samples, for maximum stepsizes \(\Delta t_{\text{max}} = 0.1 \times 2^{-k}\) for \(k = 0,\ldots,4\). Our estimate of the strong error is thus

\[
\frac{1}{1000} \sum_{i=1}^{1000} \sup_{0 \leq n \leq N_i} \left\| X_n^i - X^i(t_n) \right\|_2,
\]

where \(i\) is the sample number, \(N_i\) is the number of steps taken for sample \(i\), \(X^i(t)\) is the exact solution for sample \(i\) and \(X_n^i\) is the numerical approximation to \(X^i(t_n)\) for sample \(i\). The methods compared are slightly different in each case.
For each test problem, three different values of $\alpha$ are chosen for the BIEC methods and, for the BI method, the values 2, 1 and 0.5 are used for $\rho$. Theoretically the BIEC methods can be used without a maximum Wiener increment; however we shall bound each component of each Wiener increment by 1 (although we do not expect this bound to be attained).

Note that the first two test problems do not satisfy the conditions of Assumption 8.2; however we shall see convergence of the BIEC method despite this.

**Test problem 1: Geometric Brownian motion**

The first test problem studied is geometric Brownian motion. For $d \geq 1$ and $m \geq 1$, let $A,B_j \in \mathbb{R}^{d\times d}$ be matrices with constant coefficients for $j = 1,\ldots,m$. Then

$$dX(t) = AX(t) \, dt + \sum_{j=1}^{m} B_j X(t) \, dW_j(t), \quad X(0) = x_0.$$  \hfill (8.26)

If the matrices $A,B_j$ commute for all $j = 1,\ldots,m$, then (8.26) has the exact solution

$$X(t) = \exp \left( \left( A - \frac{1}{2} \sum_{j=1}^{m} B_j^2 \right) t + \sum_{j=1}^{m} B_j W_j(t) \right) x_0.$$  \hfill (8.27)

If $d = 1$ we write $a$ and $b_j$ in place of $A$ and $B_j$ respectively - the exact solution will always exist in this case. For this problem we compare the BIEC method and BIEC3 method with the BI method and the explicit Euler–Maruyama scheme with fixed stepsizes. The BIEC2 method will not be used because second derivatives of the drift and diffusion functions are zero and so the BIEC2 and BIEC methods are the same.

Figures 8.1–8.4 give numerical results for geometric Brownian motion (8.26) over the interval $[0,1]$ with $d = m = 1$ and initial condition $X(0) = 1$. Four sets of parameters are used:

- $a = 0.1$, $b = 0.1$;
- $a = 1.5$, $b = 0.1$;
- $a = 0.1$, $b = 1.2$ (for this case, $a - \frac{b^2}{2} < 0$ and so we expect all of the solutions to approach zero as $t$ increases); and
- $a = 1.5$, $b = 1.2$. 
We also test geometric Brownian motion for $d = m = 2$, over the interval $[0, 1]$ with the initial condition $X(0) = [1; 1.5]^T$. We use the matrices

$$A = -2I, \quad B_1 = \begin{bmatrix} 0.3106 & 0.1360 \\ 0.1360 & 0.3106 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0.9027 & -0.0674 \\ -0.0674 & 0.9027 \end{bmatrix},$$

as used in [28]. Figure 8.5 displays the results in this case.

**Test problem 2**

The second test problem is

$$dX(t) = -(a + b^2 X(t))(1 - X(t)^2) dt + b(1 - X(t)^2) dW(t), \quad X(0) = x_0,$$

for constants $a$ and $b$, which has the exact solution

$$X(t) = \frac{(1 + x_0) \exp(-2at + 2bW(t)) + x_0 - 1}{(1 + x_0) \exp(-2at + 2bW(t)) + 1 - x_0}.$$  

For this problem we compare the BIEC methods with the BI method and the backward Euler–Maruyama method ([5.33] with $\theta = 1$), since the explicit Euler–Maruyama scheme is unstable for this problem when used with a fixed stepsize (see for example [13]). Figures 8.6 and 8.7 give numerical results for 8.29 over the interval $[0, 1]$ with initial condition $X(0) = 0$. The parameters used are:

- $a = 1$, $b = 0.1$; and
- $a = 1$, $b = 1$.

**Test problem 3**

The final test problem is

$$dX(t) = -a^2 \sin(X(t)) \cos^3(X(t)) dt + a \cos^2(X(t)) dW(t), \quad X(0) = x_0,$$

for a constant $a$, which has the exact solution

$$X(t) = \tan^{-1}(aW(t) + \tan(x_0)).$$

We compare the BIEC methods with the BI method and the Euler–Maruyama scheme with fixed stepsizes. Figures 8.8 and 8.9 display the numerical results for 8.31 over the interval $[0, 1]$ with initial condition $X(0) = 0$, where the parameters used are $a = 0.2$ and $a = 2$. 
Figure 8.1: Numerical results over 1000 samples for geometric Brownian motion with $d = m = 1$, $a = 0.1$ and $b = 0.1$. 
CHAPTER 8. SOLVING SDES USING BOUNDED WIENER INCREMENTS

Figure 8.2: Numerical results over 1000 samples for geometric Brownian motion with $d = m = 1$, $a = 1.5$ and $b = 0.1$. 
Figure 8.3: Numerical results over 1000 samples for geometric Brownian motion with $d = m = 1$, $a = 0.1$ and $b = 1.2$. 
Figure 8.4: Numerical results over 1000 samples for geometric Brownian motion with $d = m = 1$, $a = 1.5$, and $b = 1.2$. 
CHAPTER 8. SOLVING SDES USING BOUNDED WIENER INCREMENTS

Figure 8.5: Numerical results over 1000 samples for geometric Brownian motion with $d = m = 2$. 
Figure 8.6: Numerical results over 1000 samples for (8.29) with $a = 1$ and $b = 0.1$. 

\[ y(x) = -x + 1 \]
Figure 8.7: Numerical results over 1000 samples for (8.29) with $a = 1$ and $b = 1$. 
Figure 8.8: Numerical results over 1000 samples for (8.31) with $a = 0.2$. 

CHAPTER 8. SOLVING SDES USING BOUNDED WIENER INCREMENTS 194
Figure 8.9: Numerical results over 1000 samples for (8.31) with $a = 2$. 

CHAPTER 8. SOLVING SDES USING BOUNDED WIENER INCREMENTS
8.3.3 Conclusions from the numerical experiments

In each of our experiments, the BIEC methods and BI method generally produce lower strong errors than the fixed stepsize methods and, as $\alpha$ decreases for the BIEC methods and as $\rho$ decreases for the BI method, the strong error decreases while the number of steps increases. That is, the more restrictions that are placed on the maximum stepsize and maximum Wiener increments, the lower the strong error and the larger the number of steps used, as one might expect. However, it is not clear which method is the most efficient in terms of the number of steps required for the same strong error - different methods perform better in different cases, and sometimes the efficiency of a BIEC method either increases or decreases as $\alpha$ (or $\rho$, for the BI method) decreases.

Recall that here, we say that a method is more efficient than another if fewer steps are required to obtain the same strong error - the numerical cost and time taken are not considered, due to the large numerical overhead required to generate the bounded increments.

For each experiment, all of the methods have approximately the same order of strong convergence as each other. The strong error is usually of order approximately 0.5, except for geometric Brownian motion with $a = 1.5$ and $b = 0.1$, and (8.29) with $a = 1$ and $b = 0.1$, where the strong error is of order approximately 1. This is to be expected, since in these two cases the diffusion is small compared to the drift.

Conclusions from experiments using geometric Brownian motion

For geometric Brownian motion with $d = m = 1$, $a = 0.1$ and $b = 0.1$ (Figure 8.1), the BIEC method performs similarly to the fixed step Euler–Maruyama method - the error decreases for smaller $\alpha$, but the efficiency remains the same. The story is more complex for the BIEC3 and BI methods - the efficiency of the BIEC3 method increases as $\alpha$ decreases, and the same effect is even more pronounced for the BI method, efficiency increasing by a larger amount as $\rho$ decreases. For $a = 1.5$ and $b = 0.1$ (Figure 8.2), the BIEC and BIEC3 methods perform almost identically, both being slightly less efficient than the fixed step Euler–Maruyama method. We expect the BIEC and BIEC3 methods to perform similarly for a drift-dominated problem since we would expect the maximum stepsize, and not the maximum Wiener increment, to control the error and so we expect that changing (8.14) to (8.24) has little effect in this case.
CHAPTER 8. SOLVING SDES USING BOUNDED WIENER INCREMENTS

The BI method is less efficient in this case, with efficiency decreasing as \( \rho \) decreases.

In the case \( a = 0.1 \) and \( b = 1.2 \) (Figure 8.3), the BIEC method produces the smallest error, but its efficiency is far worse than any of the other methods, taking a large number of steps. The BI and BIEC3 method perform similarly to the fixed step Euler–Maruyama method in terms of efficiency, with efficiency for the BI method increasing slightly as \( \rho \) decreases. The conclusions from the case \( a = 1.5 \) and \( b = 1.2 \) (Figure 8.4) are similar - again the BIEC method is the worst in terms of efficiency, and the BI and BIEC3 methods are similar in terms of efficiency, with efficiency for the BI method again increasing slightly for smaller \( \rho \). Despite the large diffusion term, we see good convergence of all of the methods, particularly the BIEC methods for small values of \( \alpha \) and the BI method for small values of \( \rho \).

For the two-dimensional geometric Brownian motion (Figure 8.5), the BIEC and BIEC3 methods perform similarly, both being more efficient than the BI method, which in turn is more efficient than the fixed step Euler–Maruyama method. Efficiency of the BI method increases slightly as \( \rho \) decreases.

Conclusions from other experiments

Looking at the results for the second test problem 8.29, all of the BIEC methods have very similar errors and efficiency for the case \( a = 1, b = 0.1 \) (Figure 8.6), and the efficiency is similar to the fixed step implicit Euler–Maruyama method. Again, like geometric Brownian motion with \( a = 1.5 \) and \( b = 0.1 \), we expect BIEC and BIEC3 to perform similarly since we expect the change from (8.14) to (8.24) to have little effect. We also expect the change from (8.13) to (8.22) to have little effect due to the relatively small diffusion term. The BI method has worse efficiency, and the efficiency decreases as \( \rho \) decreases. However, for the case \( a = b = 1 \) (Figure 8.8), the BI method appears to be slightly more efficient than the others, with efficiency improving as \( \rho \) decreases. Among the BIEC methods, the BIEC method has the best efficiency, followed by the BIEC3 and BIEC2 methods. All have better efficiency than the fixed step method.

For the third test problem 8.31, the BI and BIEC3 methods are the most efficient methods for the case \( a = 0.2 \) (Figure 8.8), whereas for the case \( a = 2 \) (Figure 8.9), the BI method is clearly more efficient than any of the other methods. The efficiency of the BI method also improves as \( \rho \) decreases, and for \( a = 0.2 \) the efficiency of the
BIEC3 method improves as $\alpha$ decreases. In each case, the other methods all perform similarly to each other, with the BIEC2 method being the least efficient.

A note on the BIEC methods and the BI method

It may be possible to speed the BIEC methods and BI method up if, for example, a cdf inversion method such as those used to sample from the Gaussian distribution \[75\] were used to sample the exit time and the exit points - this would be complicated by the fact that the function $\bar{Q}_r$, described in §7.3, is a function of two variables.

Currently the BIEC methods and the BI method are slow to implement, because of the nonlinear equations which must be solved in order to find the stepsize and the Wiener increment. There are also other drawbacks to these methods. Firstly, it is not possible to use a pre-determined path of the Wiener process whilst using these methods, as it is for some of the other methods described in Chapter 6. Further, it is not obvious how to choose the parameter $\alpha$ - here we have used larger values of $\alpha$ for problems where we expect the values of $\|F^{ad}_{(j,r)}(X_n)\|_2$ to be larger for $j, r = 0, \ldots, m$, and smaller where we expect these values to be smaller, in order to ensure that we do not have too lax or too strong a restriction on the stepsize (if the restriction is too lax then we may end up effectively using fixed stepsizes, while if the restriction is too strong then we will use stepsizes which are too small). The BI method has an advantage in this respect since the effect of the parameter $\rho$ is much more predictable. For example we know that for $m = 1$ and $\rho = 2$, we will use the stepsize $\Delta t_{\text{max}}$ approximately 89% of the time (since $P_1(2) = 0.89$), and the other 11% of the time we expect the stepsize to be smaller than $\Delta t_{\text{max}}$.

8.3.4 Future directions

We would like to test the BIEC methods and the BI methods for a wider variety of SDEs, especially multi-dimensional SDEs. This may give us a better comparison between the methods, since our range of test problems is limited at present.

For the BIEC methods and BI method to be competitive numerical methods for solving SDEs, faster implementation of the algorithm to simulate bounded Wiener increments is necessary. For example, we could try to use a cdf inversion method such as those used to sample from the Gaussian distribution \[75\] to sample the stepsize and
the exit points. This would be complicated by the fact that the function $\tilde{Q}_r$, described in §7.3, is a function of two variables.

In many of the numerical experiments, the BI method produced results that were as good as, if not better than, the BIEC methods, without needing to calculate the drift and diffusion functions and, in particular, their derivatives. So one possible area of research is to investigate the convergence of the BI method and find out if convergence is guaranteed. Intuitively one may expect the method to converge since it has been proven that the Euler–Maruyama scheme with fixed stepsizes converges. It may also be possible to use aspects of the BI method to improve the BIEC methods further.

One major disadvantage of the BIEC methods is the fact that the parameter $\alpha$ is difficult to choose in advance. Therefore it would be beneficial if we could modify the method so that the effect of the parameter $\alpha$ is more predictable in advance.

Lamba [42] made an attempt to use bounded Wiener increments with error control using the Milstein scheme, in the case $d = 1$. Since the Milstein scheme has a higher order of convergence than the Euler–Maruyama scheme, development of a numerical method which uses bounded Wiener increments along with the Milstein scheme may be desirable.
Bibliography


