Computational Aspects of the Numerical Solution of SDEs

by

Nicholas Yannios, M.Sc, MAustMS

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

July, 2001
DEAKIN UNIVERSITY

CANDIDATE DECLARATION

I certify that the thesis entitled:

**Computational Aspects of the Numerical Solution of SDEs**

submitted for the degree of:

**Doctor of Philosophy**

is the result of my own research, except where otherwise acknowledged, and that this thesis in whole or in part has not been submitted for an award, including a higher degree, to any other university or institution.

Full Name: Nicholas Yannios

Signed: Date: 30\textsuperscript{th} July, 2001
Abstract

In the last 30 to 40 years, many researchers have combined to build the knowledge base of theory and solution techniques that can be applied to the case of differential equations which include the effects of noise. This class of “noisy” differential equations is now known as stochastic differential equations (SDEs).

Markov diffusion processes are included within the field of SDEs through the drift and diffusion components of the Itô form of an SDE. When these drift and diffusion components are moderately smooth functions, then the processes’ transition probability densities satisfy the Fokker-Planck-Kolmogorov (FPK) equation – an ordinary partial differential equation (PDE). Thus there is a mathematical inter-relationship that allows solutions of SDEs to be determined from the solution of a noise free differential equation which has been extensively studied since the 1920s.

The main numerical solution technique employed to solve the FPK equation is the classical Finite Element Method (FEM). The FEM is of particular importance to engineers when used to solve FPK systems that describe noisy oscillators. The FEM is a powerful tool but is limited in that it is cumbersome when applied to multidimensional systems and can lead to large and complex matrix systems with their inherent solution and storage problems.

I show in this thesis that the stochastic Taylor series (TS) based time discretisation approach to the solution of SDEs is an efficient and accurate technique that provides transition and steady state solutions to the associated FPK equation.

The TS approach to the solution of SDEs has certain advantages over the classical techniques. These advantages include their ability to effectively tackle stiff systems, their simplicity of derivation and their ease of implementation and re-use. Unlike the FEM approach, which is difficult to apply in even only two dimensions, the simplicity of the TS approach is independent of the dimension of the system under investigation. Their main disadvantage, that of requiring a large number of simulations and the associated CPU requirements, is countered by their underlying structure which makes them perfectly suited for use on the now prevalent parallel or distributed processing systems.

In summary, I will compare the TS solution of SDEs to the solution of the associated
FPK equations using the classical FEM technique. One, two and three dimensional FPK systems that describe noisy oscillators have been chosen for the analysis. As higher dimensional FPK systems are rarely mentioned in the literature, the TS approach will be extended to essentially infinite dimensional systems through the solution of stochastic PDEs.

In making these comparisons, the advantages of modern computing tools such as computer algebra systems and simulation software, when used as an adjunct to the solution of SDEs or their associated FPK equations, are demonstrated.
Dedication

This work is dedicated to my son Emmanuel Yannios, stillborn twin brother of Dimitri. He fought and suffered for as long as it took to allow his brother the chance to live.

This thesis is the only thing I can give him now.
Acknowledgment

My sincerest thanks to my technical supervisor, Professor Peter Kloeden of the Johann Wolfgang Goethe Universität (and formerly Deakin University), for his invaluable guidance, advice, help and friendship throughout the development of this thesis. This was of particular importance during the second half of my candidature when the physical distance between us was immense.

Many thanks to my administrative supervisor, Professor Lynn Batten of Deakin University, for her important support and assistance in many matters relating to the requirements of my candidature.

Many thanks also to the Faculty of Science and Technology of Deakin University for its financial assistance in covering all HECS costs and liabilities during my candidature and especially to the Faculty Secretary, Ms. Irene Henderson, for all her support. The Faculty should also be acknowledged for their financial support in sending me to the Johann Wolfgang Goethe Universität in Frankfurt, Germany to spend two weeks with Professor Kloeden to complete this thesis.

Further, I would like to thank the staff of the Rusden and Burwood campus libraries and especially my direct contact, Ms. Libby Brackenridge, for their support and helpful attitude at all times.

I would like to thank my current employer, ANZ Bank Ltd., and my manager, Mr. James Marinopoulos for providing a generous amount of study leave to help me complete my thesis. Further, my other employers during my candidature; Deakin University and Tenix Defence Systems should be thanked for their support.

Finally, completing this thesis would not have been possible without the support of my family; my wife Denise and my children Dimitri, Chrysoula and the newly arrived John, who had to put up with my frequent absences to “...finish the bloody Ph.D!” I thank them all.
Preface ... A personal journey

I began this Ph.D in May 1993 and I am at the end of my maximum allowed duration of 8 years. A lot has changed since I began this journey.

My personal motivations for undertaking this Ph.D were twofold:

Firstly, I had always dreamt of completing a Ph.D. In 1986, against the wishes of my then supervisor, Dr. J. O. Murphy of Monash University, I submitted my thesis entitled “Boundary Effects on Convection in a Fluid Layer Subjected to Underside Heating” which gained me my M.Sc degree. Dr. Murphy wanted me to spend a further six to twelve months to further develop the thesis and to submit it for a Ph.D. However, financial considerations at the time conspired against this option.

My second motivation followed the first round of drastic changes to the Australian higher education system in 1992. At that time I was employed by Victoria College (Rusden Campus) as a Lecturer in Mathematics. Following the upheaval, Victoria College was merged into Deakin University and changed its ethos from having an emphasis on teaching to an emphasis on research. As a consequence, I decided to instill some direction to the new research component of my workload and took the opportunity to begin research to obtain a Ph.D. Broadly, the topic of my research efforts was to investigate the numerical solution of stochastic differential equations, and my supervisor was Professor Peter Kloeden.

My research progressed well with a substantial proportion of my numerical experiments completed by the end of 1996. Unfortunately, this coincided with the second major upheaval of higher education in Australia, major funding cuts. This led to my tenured position being made redundant.

At that point I had the choice of whether to continue or end the research towards the Ph.D. Based on my first motivating factor and my belief that I was at the 75% completion level I decided to continue. Unfortunately, at the same time, and again as a consequence of the funding cuts, Professor Kloeden left Australia for Germany. This has meant that the last half of my candidature was supervised via email.

Unfortunately, as I have learnt over the past four years, completing a Ph.D whilst working full time and with family commitments is very difficult. It has taken me four years to complete the final 25%, the thesis write up!
# Contents

1 Introduction ................................................. 1
   1.1 Introduction and Motivation ........................... 2
      1.1.1 Overview ........................................ 2
      1.1.2 Brief History of SDEs ............................. 4
      1.1.3 Motivation ........................................ 5
   1.2 Issues to be Examined .................................... 7
      1.2.1 FEM approach for FPK equations .................. 7
      1.2.2 Stochastic time discretisation approach for SDEs 8
   1.3 Use of modern tools ...................................... 11
      1.3.1 Application of MAPLE ............................. 11
      1.3.2 Application of SIMULINK .......................... 12
   1.4 Structure of this Thesis ................................. 17
   1.5 Chapter Summary ........................................ 19

2 Solution Approaches ........................................ 20
   2.1 Finite Element Method Approach ....................... 21
      2.1.1 Introduction ...................................... 21
      2.1.2 FEM for One Dimensional Systems ................. 22
      2.1.3 FEM for Higher Dimensional Systems .............. 29
   2.2 Taylor Series Time Discretisation Approach .......... 30
      2.2.1 Underlying Concepts .............................. 30
      2.2.2 Strong Taylor Schemes ............................. 41
      2.2.3 Weak Taylor Schemes .............................. 42
      2.2.4 Variations of Order 2.0 Weak Schemes .............. 45
      2.2.5 The use of MAPLE ................................. 48
### 2.3 Monte Carlo Approach .......................................................... 51
### 2.4 Chapter Summary ..................................................................... 53

### 3 Time Discretisation Approach Tools ....................................... 54

#### 3.1 Random Number Generation .............................................. 55
  - 3.1.1 Introduction ................................................................. 55
  - 3.1.2 The History of Random Number Generation .................. 55
  - 3.1.3 Types of Pseudo-Random Number Generators .............. 57
  - 3.1.4 Conversion to Normal Distribution ............................... 62
  - 3.1.5 Testing Random Number Generators ............................ 65
  - 3.1.6 Experimental Results ................................................ 69
  - 3.1.7 Concluding Remarks .................................................. 81

#### 3.2 A Distributed Processing System ........................................ 82
  - 3.2.1 Introduction ................................................................. 82
  - 3.2.2 The “Master” and the “Slaves” .................................... 84
  - 3.2.3 System Development .................................................. 85
  - 3.2.4 Numerical Experiment ............................................... 88
  - 3.2.5 Concluding Remarks .................................................. 90

#### 3.3 Chapter Summary ............................................................... 91

### 4 FPK Equation in One Dimension .......................................... 92

#### 4.1 Introduction ................................................................. 93

#### 4.2 SIMULINK Solution .......................................................... 96
  - 4.2.1 Overview ................................................................. 96
  - 4.2.2 SIMULINK 1D SDE model .......................................... 96
  - 4.2.3 Results ................................................................. 97
  - 4.2.4 Concluding Remarks ................................................ 100

#### 4.3 Finite Element Method (FEM) Solution ............................... 101
  - 4.3.1 Overview ................................................................. 101
  - 4.3.2 FEM Solution of the FPK Equation ............................ 101
  - 4.3.3 Concluding Remarks ................................................ 107

#### 4.4 Taylor Series Time Discretisation Approach ......................... 108
  - 4.4.1 The solution process ................................................ 108
4.4.2 Use of MAPLE in the solution process ......................... 109
4.4.3 Weak 2.0 Order Numerical Schemes Utilised .................. 109
4.4.4 Results ................................................. 114
4.5 Comparison of Methods ........................................... 127
4.6 Chapter Summary .................................................. 127

5 FPK Equation in Two Dimensions .................................. 128

5.1 Introduction ....................................................... 129
5.2 Control System Solution .......................................... 131
  5.2.1 Overview .................................................... 131
  5.2.2 SIMULINK 2D SDE model .................................. 131
  5.2.3 Results ...................................................... 136
5.3 Finite Element Method (FEM) Solution ............................ 139
  5.3.1 Overview .................................................... 139
  5.3.2 FEM Approach .............................................. 140
  5.3.3 Use of MAPLE in the solution process ...................... 141
  5.3.4 Solution of the FPK Equation ............................... 147
5.4 Taylor Series Time Discretisation Solution ......................... 148
  5.4.1 The Solution Process ....................................... 148
  5.4.2 Weak 2.0 Order Schemes Used .............................. 148
  5.4.3 Use of MAPLE in the solution process ...................... 152
  5.4.4 Results ...................................................... 153
5.5 Comparison of Methods ............................................ 161
5.6 Chapter Summary .................................................... 162

6 FPK Equation in Three Dimensions ................................. 163

6.1 Introduction ....................................................... 164
  6.1.1 Benchmark FPK system ..................................... 164
  6.1.2 Duffing Oscillator ......................................... 167
  6.1.3 Solution Approaches ....................................... 167
6.2 Control System Solution ........................................... 168
  6.2.1 SIMULINK 3D SDE Model .................................. 168
  6.2.2 Results ...................................................... 169
6.3 FEM Results .............................................................. 173
6.4 Taylor Series Time Discretisation Solution ................................. 174
  6.4.1 Introduction ......................................................... 174
  6.4.2 Schemes used – Benchmark system .................................. 174
  6.4.3 Schemes used – Duffing oscillator system .......................... 175
  6.4.4 Benchmark system solution ......................................... 177
  6.4.5 Duffing oscillator system solution .................................. 177
  6.4.6 Microsoft Excel/VBA Random Number Generator .................... 191
6.5 Discussion ........................................................................ 192
6.6 Chapter Summary .................................................................. 193

7 Multidimensional SDEs through SPDEs ........................................ 194
  7.1 Introduction ...................................................................... 195
  7.2 Spatial Discretisation Methods .......................................... 197
    7.2.1 Method of Lines Approach .......................................... 197
    7.2.2 Galerkin Approach .................................................... 198
  7.3 Linear Example ................................................................... 200
    7.3.1 Solution of Example 1 without Noise .............................. 200
    7.3.2 Method of Lines approach to Example 1 .......................... 202
    7.3.3 Galerkin approach to Example 1 .................................... 204
  7.4 Nonlinear Example ................................................................ 210
    7.4.1 Solution of Example 2 without Noise .............................. 210
    7.4.2 Method of Lines approach to Example 2 .......................... 212
    7.4.3 Galerkin approach to Example 2 .................................... 214
  7.5 The Stochastic KPP Equation ............................................. 222
    7.5.1 Solution of Stochastic KPP equation without Noise .............. 222
    7.5.2 Method of Lines approach to the SKPP Equation .................. 225
    7.5.3 Galerkin approach to the Stochastic KPP equation ............... 228
  7.6 The SKPP equation with initial point mass ............................... 240
  7.7 Comparison of Methods .................................................... 243
  7.8 Chapter Summary ................................................................ 243

Thesis Summary ......................................................................... 244
List of Figures

1.1 Useful blocks for creating Simulink models to solve SDEs. .......................... 13
1.2 Example of a Simulink model. ................................................................. 13
1.3 Inputs for the uniform random number generator block. ......................... 14
1.4 Marsaglia diagram for the MATLAB uniform random number generator. 15
1.5 Statistical summary of output of MATLAB Gaussian random number gen-
erator for samples sizes $10^3$, $10^4$, $10^5$, and $10^6$. ......................... 15

2.1 Linear trial functions $H_i(x)$ ................................................................. 23
2.2 Exact pdf (line) against 5 node FEM approximation (circles) .................. 26
2.3 Exact pdf (line) against 9 node FEM approximation (circles) .................. 26
2.4 Exact pdf (line) against 17 node FEM approximation (circles) ............... 27
2.5 Exact pdf (line) against 33 node FEM approximation (circles) ............... 27
2.6 Exact pdf (line) against 65 node FEM approximation (circles) ............... 28
2.7 A typical sample path $W_t$ of the Wiener process ................................. 31
2.8 Euler-Maruyama approximation against analytic solution ....................... 39

3.1 Accept and Reject regions for the Ratio of Uniforms approach. ............. 64
3.2 Marsaglia diagram for poor LCG ............................................................ 72
3.3 Marsaglia diagram for intrinsic PASCAL/C generator ............................ 73
3.4 Marsaglia diagram for intrinsic FORTRAN generator ........................... 73
3.5 Marsaglia diagram for intrinsic FORTRAN (VAX) generator ................. 74
3.6 Marsaglia diagram for Wichmann–Hill generator ................................ 74
3.7 Marsaglia diagram for Park–Miller generator ....................................... 75
3.8 Marsaglia diagram for L’Ecuyer generator ........................................... 75
3.9 Marsaglia diagram for Sherif–Dear generator ...................................... 76
3.10 Analysis of Box–Muller conversion ...................................................... 78
3.11 Analysis of Polar–Marsaglia conversion. ............................. 79
3.12 Analysis of Ratio of Uniforms conversion. ............................. 80
3.13 Typical single processor simulation flowchart. ............................. 83
3.14 Typical multiple processor simulation flowchart. ............................. 84
3.15 Comparison of distributed processing systems ............................. 89
3.16 Performance Ratio for Version 1 ............................. 90

4.1 Probability density for bimodal distribution with $\eta_1 = 40$ and $\eta_2 = 10, 20, 40, 50$
and 80. ........................................... 94
4.2 Variation of the normalising constant $C_0$ against $\eta_2$ for $\eta_1 = 40$. ....... 95
4.3 A SIMULINK model to solve 1 dimensional SDEs. .................. 96
4.4 Stationary pdf solution of equation 4.1 for $\eta_2 = 10$. .................. 98
4.5 Stationary pdf solution of equation 4.1 for $\eta_2 = 40$. .................. 98
4.6 Stationary pdf solution of equation 4.1 for $\eta_2 = 80$. .................. 99
4.7 FEM solution for $\eta_2 = 10$ using 11, 31, 51 and 101 nodes ............ 102
4.8 FEM solution for $\eta_2 = 20$ using 11, 31, 51 and 101 nodes ............ 103
4.9 FEM solution for $\eta_2 = 40$ using 11, 31, 51 and 101 nodes ............ 104
4.10 FEM solution for $\eta_2 = 50$ using 11, 31, 51 and 101 nodes ............ 105
4.11 FEM solution for $\eta_2 = 80$ using 11, 31, 51 and 101 nodes ............ 106
4.12 CPU requirement as a function of number of nodes for the Gaussian Density
   Solution example. ........................................... 107
4.13 Steps in the numerical solution of SDEs. ............................. 108
4.14 Graphical comparison of the effect of number of realisations on the solution.116
4.15 Effect of number of simulations on CPU time required. ............. 117
4.16 Comparison of observed (histograms) to actual probability density (line) for
   the standard case of $\eta_1 = 40$ and $\eta_2 = 10$ for the five schemes investigated:
   (a) Taylor Series; (b) Implicit; (c) Predictor–Corrector; (d) Derivative Free;
   (e) Linearly Implicit. ........................................... 120
4.17 Comparison of observed (histograms) to actual probability density (line)
   for the weak 2.0 order Taylor series scheme with $\eta_1 = 40$ and varying $\eta_2$:
   (a) $\eta_2 = 20$; (b) $\eta_2 = 40$; (c) $\eta_2 = 50$; (d) $\eta_2 = 80$. ............. 121
4.18 Comparison of observed (histograms) to actual (line) results for single (a) and averaged (b) simulations for the weak 2.0 order Taylor series scheme for the \( \eta_1 = 40 \) and \( \eta_2 = 80 \) case. 

4.19 Comparison of simulated (histograms) and actual (line) results for the weak 2.0 order Taylor series scheme using a varying starting value for the \( \eta_1 = 40 \) and \( \eta_2 = 80 \) case. 

5.1 A SIMULINK model to solve 2 dimensional SDEs. 

5.2 Modular SIMULINK model to solve multidimensional SDEs (upper) with respective details of modules (lower). 

5.3 SIMULINK generated pdf solution. 

5.4 Time evolution of the SIMULINK generated solution of the Duffing oscillator with additive noise. 

5.5 Scanned results from Spencer and Bergman [83]. 

5.6 General two dimensional finite element with four nodes. 

5.7 The shape function \( N_1 \). 

5.8 General coordinates of nodes for each element. 

5.9 Evolution of Experiment 1 pdf in unit time steps. 

5.10 Surface representation of Experiment 1 stationary pdf solution. 

5.11 Effect of timestep on Experiment 1 stationary pdf. Timesteps are \( \Delta = \{0.2, 0.1, 0.02, 0.01, 0.002\} \) respectively. 

5.12 Effect of number of realisations on Experiment 1 stationary pdf. Number of realisations are \( 10^4 \) and \( 10^5 \) respectively. 

5.13 Stationary pdf solution for Experiment 2. 


5.15 Stationary pdf solution for Experiment 3. 

5.16 Bergman et al.[5] results for Experiment 3. 

6.1 Solution of the benchmark FPK equation for \( X_3 = 0 \). 

6.2 A SIMULINK model to solve 3 dimensional autonomous SDE systems. 

6.3 Time variation of the errors of the second moments of the benchmark system via the SIMULINK approach. 

6.4 SIMULINK generated solution for the benchmark system at \( X_3 = 0 \).
6.5 Time variation of the errors of the second moments of the benchmark system via the stochastic Euler approach.
6.6 Stochastic Euler scheme solution of benchmark system with $10^4$, $10^5$ and $10^6$ realisations respectively.
6.7 Stochastic Euler, weak 2.0 and weak 3.0 order Taylor series solution of benchmark system with $10^6$ realisations.
6.8 Time evolution of the benchmark system in the $X_3 = 0$ plane via the stochastic Euler scheme.
6.9 Time variation of the second moments of the unimodal Duffing Oscillator system via the stochastic Euler scheme.
6.10 Time variation of the second moments of the bimodal Duffing Oscillator system via the stochastic Euler scheme.
6.11 Stochastic Euler solution of unimodal Duffing oscillator at $X_3 = -0.96$.
6.12 Stochastic Euler solution of unimodal Duffing oscillator at $X_3 = 0$.
6.13 Stochastic Euler solution of unimodal Duffing oscillator at $X_3 = 0.96$.
6.14 Time evolution of the unimodal Duffing oscillator system in the $X_3 = 0$ plane via the stochastic Euler scheme.
6.15 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = -1.76$.
6.16 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = -0.96$.
6.17 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = -0.48$.
6.18 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 0$.
6.19 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 0.48$.
6.20 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 0.96$.
6.21 Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 1.76$.
6.22 Time evolution of the bimodal Duffing oscillator system in the $X_3 = 0$ plane via the stochastic Euler scheme.
6.23 Plot-Pairs diagram for the Microsoft Excel random number generator.

7.1 Finite Difference solution to Example 1 with no noise ($\sigma = 0$).
7.2 Exact solution to Example 1 with no noise ($\sigma = 0$).
7.3 Exact solution to Example 1 with $\sigma = 1$.
7.4 Galerkin approach solution to Example 1 with $\sigma = 0$.
7.5 Galerkin approach solution to Example 1 with $\sigma = 0.1$.
7.6 Galerkin approach solution to Example 1 with \( \sigma = 0.5 \) .......... 208
7.7 Galerkin approach solution to Example 1 with \( \sigma = 1 \) .......... 208
7.8 Galerkin approach solution to Example 1 with \( \sigma = 2 \) .......... 209
7.9 Galerkin approach solution to Example 1 with \( \sigma = 5 \) .......... 209
7.10 Finite Difference solution to Example 2 with no noise (\( \sigma = 0 \)). .......... 211
7.11 Galerkin approach solution to Example 2 with \( \sigma = 0 \). .......... 218
7.12 Galerkin approach solution to Example 2 with \( \sigma = 0.1 \) .......... 218
7.13 Galerkin approach solution to Example 2 with \( \sigma = 0.2 \) .......... 219
7.14 Galerkin approach solution to Example 2 with \( \sigma = 0.5 \) .......... 219
7.15 Galerkin approach solution to Example 2 with \( \sigma = 1 \) .......... 220
7.16 Galerkin approach solution to Example 2 with \( \sigma = 2 \) .......... 220
7.17 Galerkin approach solution to Example 2 with \( \sigma = 5 \) .......... 221
7.18 Galerkin approach solution to Example 2 with \( \sigma = 10 \) .......... 221
7.19 Finite Differences solution to KPP equation with \( \mu = 0.2 \) and 0.5 .......... 223
7.20 Finite Differences solution to KPP equation with \( \mu = 0.8 \) and 1.0 .......... 224
7.21 Finite Differences solution to KPP equation with \( \mu = 1.4 \) .......... 224
7.22 Galerkin approach solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0 \) .......... 229
7.23 Galerkin approach solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0.1 \) .......... 230
7.24 Galerkin approach solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0.2 \) .......... 230
7.25 Galerkin approach solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0.5 \) .......... 231
7.26 Galerkin approach solution to SKPP equation with \( \mu = 0.5 \) and \( k = 1 \) .......... 231
7.27 Gaines’ method solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0.1 \) .......... 232
7.28 Gaines’ method solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0.2 \) .......... 232
7.29 Gaines’ method solution to SKPP equation with \( \mu = 0.5 \) and \( k = 0.5 \) .......... 233
7.30 Gaines’ method solution to SKPP equation with \( \mu = 0.5 \) and \( k = 1.0 \) .......... 233
7.31 Galerkin approach solution to SKPP equation with \( \mu = 0.2 \) and \( k = 0 \) .......... 234
7.32 Galerkin approach solution to SKPP equation with \( \mu = 0.2 \) and \( k = 0.1 \) .......... 235
7.33 Galerkin approach solution to SKPP equation with \( \mu = 0.2 \) and \( k = 0.2 \) .......... 235
7.34 Galerkin approach solution to SKPP equation with \( \mu = 0.2 \) and \( k = 0.5 \) .......... 236
7.35 Galerkin approach solution to SKPP equation with \( \mu = 0.2 \) and \( k = 1 \) (20
    terms) .......... 236
7.36 Gaines’ method solution to SKPP equation with \( \mu = 0.2 \) and \( k = 0.1 \) .......... 237
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.37</td>
<td>Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 0.2$</td>
<td>237</td>
</tr>
<tr>
<td>7.38</td>
<td>Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 0.5$</td>
<td>238</td>
</tr>
<tr>
<td>7.39</td>
<td>Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 1.0$</td>
<td>238</td>
</tr>
<tr>
<td>7.40</td>
<td>Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 1$ (50 terms)</td>
<td>239</td>
</tr>
<tr>
<td>7.41</td>
<td>Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 1$ (100 terms)</td>
<td>239</td>
</tr>
<tr>
<td>7.42</td>
<td>Galerkin scheme approach for SKPP with point mass initial conditions</td>
<td>241</td>
</tr>
<tr>
<td>7.43</td>
<td>Finite Differences solution for SKPP with point mass initial conditions</td>
<td>242</td>
</tr>
</tbody>
</table>
## List of Tables

3.1 Critical values of $\chi^2_{0.05,\nu}$ ......................................................... 66
3.2 Comparison of $U(0,1)$ generators .......................................................... 71
3.3 Comparison of $U(0,1)$ to $N(0,1)$ converters ........................................... 77

4.1 Normalising constant and summary statistics for the probability density solution for $\eta_1 = 40$ and various values of $\eta_2$. ................................. 95
4.2 Effect of the number of simulations on the $\chi^2$ statistic, summary statistics and CPU time required. ................................................................. 115
4.3 Effect of initial value on the $\chi^2$ statistic and summary statistics for the weak 2.0 order Taylor series scheme with $\eta_1 = 40$ and $\eta_2 = 10$. ............. 118
4.4 Effect of scheme used on the $\chi^2$ statistic and summary statistics for the $\eta_1 = 40$ and $\eta_2 = 10$ case. ................................................................. 119
4.5 Effect of scheme used on the CPU time required for the $\eta_1 = 40$ and $\eta_2 = 10$ case. ................................................................. 119
4.6 Effect of the value of $\eta_2$ on the $\chi^2$ statistic and summary statistics for the weak 2.0 order Taylor series scheme with $X_0 = 0.5$. ........................ 122
4.7 Effect of initial value on the $\chi^2$ statistic and summary statistics for the weak 2.0 order Taylor series scheme for the $\eta_1 = 40$ and $\eta_2 = 80$ case. 124
4.8 Effect of scheme used and value of $\eta_2$ on the $\chi^2$ statistic. Recall that the critical $\chi^2$ value is 56.0685 ......................................................... 125
4.9 Effect of scheme used and value of $\eta_2$ on the mean. ............................... 125
4.10 Effect of scheme used and value of $\eta_2$ on the variance. ........................... 126

5.1 Comparison of relative CPU times ......................................................... 161

6.1 Comparison of the simulated second moments of the benchmark system at $T = 20$ against the exact and FEM second moments. ...................... 171
6.2 Comparison of the stochastic Euler scheme generated second moments of the benchmark system at \( T = 20 \) against the exact and FEM second moments. 

6.3 Comparison of the weak 2.0 order scheme generated second moments of the benchmark system at \( T = 20 \) against the exact and FEM second moments. 

6.4 Comparison of the weak 3.0 order scheme generated second moments of the benchmark system at \( T = 20 \) against the exact and FEM second moments.
Chapter 1

Introduction
1.1 Introduction and Motivation

1.1.1 Overview

In many fields of science, engineering and commerce, differential equations are used to model or describe observed phenomena. The wide ranging applicability of differential equations has led to the development of a wealth of analytical and numerical techniques for their solution.

However, many differential equations used to model real world phenomena are often crude approximations, and a particular component often neglected in the modelling is the “noise” affecting the system. In fact, solutions of complex behaviour which neglect the intrinsic noise are only descriptions of what is occurring “on the average”. This universally prevalent noise can be considered as the sum or product of the multiple sources of random influences affecting the system and can include the measurement errors inherent in any experimental based approach. For engineers and researchers it is much easier to neglect these influences and only analyse a system that can make use of the vast volumes of current knowledge in that field.

In the last 30 to 40 years, many researchers have collaborated to build a knowledge base of theory and solution techniques that can be applied to differential equations that include the effects of noise, the field of stochastic differential equations (SDEs).

Mathematicians and theoretical scientists are mainly inclined to investigate the individual sample paths of the SDE. These are termed “strong” approximations and result from the direct solution of SDEs.

Other researchers such as engineers and applied scientists solve noisy systems to obtain the underlying probability distribution or density function (pdf) solution. Such a solution is termed a “weak” approximation and is obtained by the solution of a deterministic PDE, the Fokker-Planck-Kolmogorov (FPK) equation. The FPK equation corresponds to the Itô form of an SDE. Solutions of the FPK equation are usually obtained by the Finite Element Method (FEM) or Monte Carlo method approaches. Finite difference (FD) methods, often used in the solution of deterministic PDEs, have been found to be unwieldy when applied to the FPK equation of dimension greater than two. See Reuter [68], Saulév and Cherinkiv [75], Chang and Cooper [11], To [88] and Toland and Yang [89] for examples of the use of finite difference schemes in the solution of FPK
equations. Note though that FEM techniques typically have special advantages over FD methods when solving FPK equations and as such FD methods will not be considered further in this thesis.

Once the mathematical framework for SDEs was in place, researchers began to concentrate on developing analytic and numerical solution techniques for classes of SDEs. However they found that using the techniques of classical ODEs as a blind guide led to many difficulties with inconsistencies or slow convergence. Hence there was the need for new types of schemes for the accurate and efficient numerical solution of SDEs. What has arisen is the relatively new numerical solution approach for SDEs, the stochastic Taylor series based time discretisation approach. This approach uses repeated applications of the stochastic Itô formula to the integrands that arise from the Taylor series expansion to generate iterative schemes of almost any order of convergence.

As mentioned and as will be further described in Section 2.2.1 there is an interrelationship between a FPK equation and an Itô SDE. This allows the weak versions of the time discretisation schemes to be used to obtain the probability density solution of the FPK equation by solving the corresponding Itô SDEs. This is the approach that will be used in this thesis when using weak versions of the stochastic time discretisation schemes to solve FPK equations.

Applications of FPK equations and associated SDEs vary widely in both the theoretical and applied engineering fields. Langley [51] applied and solved FPK equations in the dynamic analysis of multibody offshore structures. In Langley’s models, offshore structures subjected to wind, current and random wave forces had dynamic responses that were stochastic processes. Roberts [72, 73] studied the effects of random sea motions on both moored and rolling ships. Harris [30] derived and solved a five dimensional system of SDEs to describe wastewater treatment. Heemink [31] used SDEs to study the dispersion of pollutants in shallow water. On a larger scale, Mignolet and Fan [61] looked at the first order response of structures subject to random motions induced by seismic activity.

To further emphasise the diversity of fields that involve SDEs, consider the following: astrophysics (Chandrasekhar [10]), chemical reactions (Benson [4]), blood clotting (Fogelson [25]) and communications (Jazwinski [36]). Finally, the ubiquitous Black and Scholes model [7] used in options pricing in the finance field is just a simple SDE.
1.1.2 Brief History of SDEs

Stochastic Differential Equations are related historically with the theory of diffusion processes and followed from the discovery of the concept of Brownian Motion.

Brownian Motion was described following experimental work by R. Brown in 1827. He observed that a small particle on the surface of a fluid was in constant irregular motion. Based on his observations Brown was able to disprove the then common belief that this random motion was due to the motion of living organisms. It is still considered one of the most interesting examples of a random physical process. As will be described below, it was the starting point for the description of SDEs.

In 1900, F. M. Exner proposed the correct physical explanation for Brownian Motion; that the motion was caused by the random collisions of the fluid molecules with the particle. Important theoretical breakthroughs were due to Einstein [19] in 1905 and then by Smoluchowski [80] in 1906 who generalised and extended Einstein’s work. They described the physical process of Brownian motion in terms of a probability density function of a particle’s displacement. They showed that the underlying probability density function was the solution of a deterministic differential equation, later to be called the Fokker-Planck(-Kolmogorov) equation.

In 1908 Langevin produced a mathematical description that encompassed Einstein’s and Smoluchowski’s work. Langevin’s mathematical description defined the concept of a stochastic differential equation. In his approach, Langevin applied Stokes’ Law to a spherical particle of mass \( m \) and described Brownian Motion by the equation

\[
m \frac{dv}{dt} = f(t)
\]

where at time \( t \), \( v \) is the velocity of the particle and \( f(t) \) is the force applied to the particle by the surrounding medium. As this force comprised a dissipative and a fluctuating component Langevin expressed \( f(t) \) in the form

\[
f(t) = -\beta v + \eta(t)
\]

where \(-\beta v\) is the regular dissipative influence of the surrounding medium and \( \eta(t) \) (the \textit{Langevin force}) is the random force exerted on the particle by the collisions. Using \( a = \beta/m \) and \( \xi(t) = \eta(t)/m \) the Langevin equation is often written as

\[
\frac{dv}{dt} = -av + \xi(t)
\]
The importance of Langevin’s work in the field of SDEs is reflected by the use of the alternative term *Langevin Equations* when referring to SDEs.

Continuing the development, Wiener developed the mathematical construct for Brownian Motion and Fokker, Planck and Kolmogorov derived the partial differential equations for the transition probabilities which bears their names.

Finally, in 1945, Itô [34] developed a rigorous mathematical treatment of SDEs culminating in the Stochastic Chain Rule. This was expanded upon in Itô’s 1951 paper [35]. Note that Stratonovich also developed a rigorous mathematical treatment of SDEs where the Chain Rule of classical calculus is retained. The Itô and Stratonovich treatments are inter-related in that the scalar Itô SDE

\[ dx = a(x, t) \, dt + b(x, t) \, dW, \]

is equivalent to the Stratonovich SDE

\[ dx = \left[ a(x, t) - \frac{1}{2} \frac{\partial b}{\partial x} \right] dt + b(x, t) \, dW. \]

Note that Itô’s form is directly related to the diffusion process and the corresponding one dimensional FPK equation

\[ \frac{dp}{dt} + \frac{d}{dx} \left\{ a(x) p \right\} - \frac{1}{2} \frac{d^2}{dx^2} \left\{ b^2(x) p \right\} = 0. \]  \hspace{1cm} (1.1)

The Itô form was used exclusively in this thesis.

### 1.1.3 Motivation

My academic motivation for tackling the numerical solution of SDEs as my field of research stemmed from my interest in numerical and computational methods. My M.Sc thesis involved convection in astrophysics with an emphasis on the numerical techniques rather than the physics. The second motivation was the bold statement:

“These results have not been seen previously and, to the authors’ knowledge, likely cannot be obtained to this resolution *in any other manner*” (my italics).

by Spencer and Bergman [83] who used the Finite Element Method (FEM) to solve for the transition density solutions of a two dimensional FPK equation that described the Duffing oscillator.
This statement is probably fair in the simplistic manner in that the FEM is a proven method that has been available for about 60 years whereas the techniques expounded by Kloeden and Platen [45] have only recently been formally developed. However FEM solutions are cumbersome to set up and to implement, and often specialised techniques are required to manage and solve large matrix systems. Often, and particularly in higher dimension, they require supercomputers to provide reasonable completion time. Moreover, a simple change to a parameter, or the requirement for more detailed analysis, can often lead to major modifications to the FEM implementation. The stochastic Taylor series based time discretisation schemes, due to their ease of implementation and suitability for running on modern quick parallel or distributed processor systems, seemed a viable alternative.

Thus in this thesis, by investigating the use of the newly developed stochastic Taylor series based time discretisation approach to the solution of FPK equations through their associated SDEs, I would be formally “testing” and comparing these techniques when applied to realistic systems. At the same time a major aspect of the research is to try to determine techniques that can be applied to simplify the solution of FPK equations/SDEs whether it be via stochastic time discretisation schemes or via the FEM approach.
1.2 Issues to be Examined

In this thesis, investigations and comparisons of the numerical solution of “real world” applicable FPK equations (or their corresponding SDEs) by the traditional FEM and the newer Itô-Taylor time discretisation approach are explored. The emphasis will be on determining the advantages, disadvantages and the class of SDE systems best tackled by each approach. I will also provide a description of my original use of modern mathematical and engineering tools such as MAPLE and SIMULINK to minimise the disadvantages of each approach. This will be achieved by describing techniques that support and greatly reduce the complexity of the schemes when used to solve SDE systems.

The following descriptions of each technique provide a general overview to the numerical solution of FPK equations or their associated SDEs:

1.2.1 FEM approach for FPK equations

The FEM approach will be detailed in Section 2.1. It is based upon first discretising the state space into nodes with connecting elements. The system is then analysed by taking into account any bounding elements. Finally the overall solution is obtained by combining the effects of all the elements. This approach typically generates large matrix systems that are cumbersome to manage and solve.

The FEM based approach has the following advantages:

- the schemes are very accurate
- for low dimensional systems the schemes tend to be computationally quick
- the schemes are applicable to systems with complex boundaries
- complex solution structures can be further investigated using system specific FEM grid structures
- the schemes do not have problems with excessive variances

However, they have the following disadvantages:

- typically the schemes are very cumbersome to set up
- a parameter change in a model can require a major re-configuration
there tend to be difficulties when trying to solve stiff systems

good knowledge of the boundary is required before starting any in-depth analysis

the schemes are very difficult to apply to systems of dimension greater than two

as the dimension of the system increases so too there is greater computer memory requirements for matrix storage

the solution accuracy is dependent upon $N$, the number of nodes or elements used, however CPU requirements increase in $N^3$

moments of the solution can be difficult or laborious to determine

there is little advantage gained in running them on parallel or distributed processing systems

1.2.2 Stochastic time discretisation approach for SDEs

The derivation of these schemes will be detailed in Section 2.2. They are essentially generated by the repeated application of the stochastic Itô formula on the integrands that arise in the Taylor series expansion. This process generates increasingly complex stochastic integrals that need to be evaluated or at least approximated. In some cases there is scope to use simpler random variables in these approximations which can reduce the computational requirements when these schemes are used.

The schemes based on the stochastic Itô-Taylor formula have the following advantages:

both strong and weak solutions are very easy to generate using an iterative approach

the accuracy of the solution can be specified by the order of the scheme chosen

the schemes take into account the structure of the SDE to help reduce the complexity of the solution approach

specialised variations of the schemes such as implicit, predictor-corrector and linearly implicit schemes are available when solving complex or stiff systems

the schemes can be easily generalised to solve systems in any number of dimensions
• CPU requirements only increase quadratically with respect to the solution accuracy
• simulations can be run without first defining the spatial boundary
• the schemes are perfectly suited for use on parallel or distributed processing systems thereby reducing their completion time
• minimal computer memory is required
• moments of any functional are very easy to generate without resorting to messy integrations

However, they have the following disadvantages:

• for the weak solutions many thousands or hundreds of thousands of runs of each sample path need to be undertaken. This can require substantial computation time
• there are occasionally problems with large variances so variation reduction techniques may need to be applied
• the schemes can be very messy to derive, especially when using higher order convergence schemes
• there are often difficulties in trying to describe or approximate multiple stochastic integrals
• the solution is dependent on the generation of suitable “random enough and for long enough” pseudo-random numbers

Monte Carlo approach

Finally, the Monte Carlo (MC) approach will only be briefly mentioned in this thesis. It was not formally investigated due to its low order of accuracy. The MC approach is based upon direct simulation of either the governing equations or simple approximations of the underlying random processes. It does not take into account any inherent structure or characteristics of the system that may reduce the complexity. Further details are presented in Section 2.3.
The Monte Carlo schemes have advantages:

- the schemes can be generalised to solve systems in any number of dimensions
- CPU requirements only increase quadratically with respect to the solution accuracy
- simulations can be run without first defining the system boundary
- the schemes are perfectly suited for use on parallel or distributed processing systems thereby reducing their turn around time
- minimal computer memory is required
- moments of any functional are very easy to generate without resorting to messy integrations

However, they have the following disadvantages:

- accuracy is of low order. The order is no better than just 1.0 which is equivalent to the weak stochastic Euler scheme
- many hundreds of thousands of runs of each sample path need to be undertaken which can require substantial computation time
- there are often problems with large variances so variation reduction techniques are usually required
- the solution is dependent on the generation of suitable “random enough and for long enough” pseudo-random numbers
1.3 Use of modern tools

One of the main aspects of my thesis was to investigate the use of modern tools to facilitate the generation of numerical solutions of FPK equations or their associated SDEs. In the past fifteen years or so, many mathematicians and analysts have made use of symbolic manipulators or computer algebra packages. For serious mathematical work most researchers use MATHEMATICA or MAPLE whereas engineers and applied mathematicians tend to use MATHCAD or MATLAB.

The computer algebra system MAPLE was made available to me through Deakin University in 1994. On leaving Deakin University and joining a defence engineering company in 1997 I was introduced to MATLAB and more specifically to its associated simulation modelling tool SIMULINK. The two tools MAPLE and SIMULINK were used extensively in the analyses presented in this thesis. The techniques employed using each package are described in the next two subsections.

1.3.1 Application of MAPLE

Research into the use of symbolic computation in stochastic analysis is only very recent; it was initially investigated by Talay [87] in 1989 and then by Kendall [38] and Valkeila [90] in the early 1990s.

Deakin University, being a major MAPLE site, had been involved in the application of MAPLE in the analysis of SDEs. One of the major contributors was Sascha Cyganowski, who developed a MAPLE package called stochastic as part of his minor thesis during his Honours year under the supervision of Prof. P. E. Kloeden. This work was based on a paper by Kloeden and Scott [46] that first provided a description and routines for the stochastic time discretisation schemes for SDEs. The stochastic package has subsequently been upgraded to include structures to help generate time discretisation schemes for SDEs. See Cyganowski [13] and Cyganowski et al [14] for a description of the package.

The stochastic package has been included in and is available from the MAPLE share library. The stochastic package is known to work for MAPLE V, Release 5.1 for Windows, Unix and Macintosh and for MAPLE 6 for Unix.

The package contains a variety of functional routines for constructing strong (up to
order 2.0) and weak (up to order 3.0) schemes. It also contains routines for finding explicit solutions to some SDEs. Finally it also contains “service” routines such as routines to generate the partial differential operators $\mathcal{L}^0$ and $\mathcal{L}^j, (j = 1, 2, \ldots)$, that arise in stochastic calculus. Some additional routines not in the stochastic package were also written to support my analyses.

Since the creation of the stochastic package Cyganowski and Kloeden [12] have used Maple to qualitatively look at the asymptotic behaviour of stochastic dynamical systems and their sensitivity to small changes. Only recently, Higham [32] created a set of MATLAB programs that provide a comprehensive introduction to the numerical solution of SDEs. Since MATLAB uses a matrix approach in its calculations it is naturally suited to the numerical solution of SDEs.

In the relevant sections within the thesis, I will describe my application of Maple and the stochastic package in supporting my numerical solutions of SDEs. I will also detail my use of Maple to help with the implementation issues encountered when trying to apply the FEM in solving FPK equations. I will describe how the use of Maple can have a major impact on minimising much of the inherent complexity when solving SDEs or the associated FPK equation. This is especially true when the FEM approach is applied.

1.3.2 Application of Simulink

Simulink is a modern software tool where objects can be graphically manipulated to build simulation models. It uses the MATLAB computational engine. It is commonly used by engineers in the field of control systems and processes. Simulink allows users to create a simulation via a drag and drop process involving components or “blocks”. These blocks are linked simply by direction control line segments. Figure 1.1 presents typical Input, Component and Output blocks that can be used to create Simulink models.

Figure 1.2 presents an example of a Simulink model, the analog control system for the three degrees of freedom model of the reaction jets on the lunar module. The scope output device shows the position for the lunar module’s roll, pitch, and yaw. This model is courtesy of The MathWorks, Inc., creators of MATLAB and Simulink.
Figure 1.1: Useful blocks for creating SIMULINK models to solve SDEs.

Figure 1.2: Example of a SIMULINK model.
Parameters for most blocks can be set or modified by double clicking the block and entering values. As an example, Figure 1.3 presents the parameter inputs screen for the uniform random number generator block. This block allows the random number generator limits and an initial seed to be set. Note that the variable `rand` refers to the MATLAB uniform random number generator and is used to create a unique starting seed for each run of the SIMULINK model. Given its features, SIMULINK can be considered as a simple object oriented programming tool where each component is an object that can be adjusted and used as required.

![Uniform Random Number Block Parameters](image)

**Figure 1.3:** Inputs for the uniform random number generator block.

As a test of the MATLAB uniform random number generator a Marsaglia (or Plot-Pairs) diagram was generated and is presented in Figure 1.4. The Marsaglia diagram presents the generated points \((U_{2n-1}, U_{2n}), n = 1, 2, \ldots\) of the random number generator on \([0, 1] \times [0, 1]\). These 2-ples can be generalised to \(m\)-ples but become difficult to graphically display. Any banding effects are usually indicative of a poor uniform random number generator. For the MATLAB uniform random number generator there are no obvious banding effects.

MATLAB also has a White Noise (Gaussian) random number generator and this feature can be accessed directly from SIMULINK. Using this white noise random number generator I created histograms of the output and compared them to the Standard Normal distribution. The results are presented in Figure 1.5 which presents histograms based on samples of \(10^3, 10^4, 10^5, \text{ and } 10^6\) generated MATLAB “White Noise” random numbers placed in 101 histogram bins.
Figure 1.4: Marsaglia diagram for the MATLAB uniform random number generator.

Figure 1.5: Statistical summary of output of MATLAB Gaussian random number generator for samples sizes $10^3$, $10^4$, $10^5$, and $10^6$. 
An engineering simulation package such as Simulink is a natural extension to the use of symbolic manipulators for the numerical solution of SDEs. The advantage of this approach is the quick creation of simulation models that can be used to generate “rough” first approximations to the solution of SDEs. Yannios [98] has demonstrated that this approach leads to an almost code free solution approach to SDEs. These rough first approximations can then be used to determine domains of interest and highlight specific solution areas of interest for further more detailed study.
1.4 Structure of this Thesis

This thesis focuses on the computational aspects of the numerical solution of FPK equation through their associated SDEs. Particular reference is made to the comparison of the classical FEM approach to the stochastic Taylor Series based time discretisation schemes as developed and described by Kloeden and Platen [45].

As an overview, FPK equations used in the applied mathematics and engineering field of vibration and oscillators will be solved by first expressing them in their Itô-SDE form. Once in this form, the classical FEM approach and the stochastic Taylor series based time discretisation approach are used to obtain the solution. As part of the solution process, specific details in relation to the use of the two main tools utilised, MAPLE and SIMULINK, will be described.

FKP equations of dimension one, two and three were studied in detail. Extending the comparison to equations in higher dimensions was difficult as there are very few described in the literature. In addition FPK equations in dimension greater than three are geometrically difficult to visualise. To extend the comparison of the techniques to higher dimensional SDEs, a set of stochastic PDEs (SPDEs) were investigated. When discretised in space an SPDE reduces to a multidimensional set of SDEs. Thus in this thesis examples and comparisons of techniques to all dimensional systems are provided.

Chapter 2 begins with a very brief introduction to the concepts that are embodied in the study of SDEs. The concepts outlined are stochastic processes, stochastic (Itô) calculus, the stochastic Taylor series, numerical methods and convergence. Much greater detail can be obtained in Kloeden and Platen [45]. The chapter will then introduce various numerical schemes that are used in later chapters in the analysis of one, two and three dimensional FPK equations/SDEs and a multidimensional set of SDEs that arise from a reaction-diffusion SPDE.

Chapter 3 provides two topics of crucial importance that support the numerical solution of SDEs when using the stochastic time discretisation approach. These topics are also relevant to the Monte Carlo approach which will not be developed directly in this thesis. The first topic involves a detailed history and description of the random number generators required by the time discretisation solutions. The second is a description of a simple distributed processing system that can be used to efficiently and quickly solve SDEs and SPDEs using time discretisation schemes.
Chapter 4 investigates the solution of a one dimensional FPK equation with known analytic solution. By varying one of its parameters the corresponding one dimensional SDE examined could be made “stiffer”. This is an effective technique to test the robustness of the numerical schemes. A Simulink model will be first used to solve the SDE to gain insight into the structure of its solution. Once the domain is determined, a traditional FEM based scheme as well as a selection of weak order 2.0 stochastic time discretisation schemes will be applied to obtain solutions. Finally the solutions will be compared to the known analytic solution.

Chapter 5 then considers a two dimensional FPK equation – one that describes a noisy Duffing oscillator subject to both additive and parametric noise. A two dimensional Simulink SDE model will be used to first provide a rough guide to the solution. Based on the Simulink results a FEM approach and a set of stochastic time discretisation schemes will be applied to solve the system. Furthermore, the solutions will be compared to the analytic solution which is known in the additive noise case only. The stochastic time discretisation scheme approach will be shown to generate the same transition solution as described by Spencer, Bergman et al [83] who believed such a solution was only possible through their FEM approach.

Chapter 6 extends the solution of FPK equations to three dimensions, in this case investigating simple Duffing oscillators through their associated system of SDEs. Again, Simulink will be initially used to gain insight into the solution. This insight will be used by the stochastic time discretisation scheme to obtain more accurate solutions to the system. The direct FEM solution of the three dimensional FPK equation will not be described since solutions to such systems are at the boundary of applicability when using that approach. Instead a comparison will be made with research undertaken by Wojtkiewicz, Bergman and Spencer [94] that make use of the FEM in three dimensions. Monte Carlo solutions by Johnson, Wojtkiewicz and Bergman [37] for three dimensional systems will also be presented.

Finally, Chapter 7 extends the analysis to higher dimensional systems via the solution of a set of SDEs that arise from the spatial discretisation of SPDEs. The SPDE analysed is a stochastic Kolmogorov-Petrovsky-Piscounoff (KPP) semi-linear reaction-diffusion equation. Two spatial discretisation approaches will be examined; a “method of lines” (Rothe’s Method) discretisation and a Galerkin based discretisation. Generated
results will be compared to the equivalent finite difference based schemes in the noise free case. Also compared are the results of Gaines [26] in her method of lines solution of the stochastic KPP equation.

1.5 Chapter Summary

This chapter began with a brief history of the development of the field of numerical solution of SDEs. It has presented a description of my motivation for undertaking the topic of the numerical solution of SDEs as my Ph.D research topic. It has presented the main issues to be examined in this thesis and introduced the concept of the use of modern tools such as symbolic manipulators and simulation packages as an adjunct to the solution of FPK equations and SDEs.
Chapter 2

Solution Approaches
2.1 Finite Element Method Approach

2.1.1 Introduction

The Finite Element Method (FEM) is a powerful analysis tool which is used to obtain approximate solutions to a variety of physical systems. It is well suited to solving partial differential equations, for example see Lapidus and Pinder [54] and Mitchell [62]. It was developed in the early 1940s for use in structural engineering analysis and was initially used for the study of stresses in aircraft structures. Since then it has been applied to many areas of mathematical physics, for example in the fields of elasticity, general field lubrication, fluid mechanics, etc., see Huebner [33]. Its growing importance as a tool has followed the development of high speed digital computing. For more details of the history of the FEM see Davies [16] and Huebner [33].

The typical FEM approach to obtaining solutions to differential equations including FPK equations consists of the following steps:

1. Model the physical system by discretising the state space into small parts called “elements”. In one dimensional systems these can be straight rods whereas in two dimensional systems they can be triangles, rectangles or other two dimensional shapes. In higher dimensional systems, more complex elements need to be defined.

2. An analysis, including a linear equation solve, is made on each element whilst taking into account the effects of bounding elements.

3. Finally, all the elements are reformed into the original state space and their solutions are combined to give the overall solution.

The FEM has been employed by a variety of researchers for the numerical solution of Fokker-Planck-Kolmogorov (FPK) equations, especially in one and two dimensions. The FPK equation is of particular importance in physics, vibration analysis, chemical physics and communications theory. For more information see Risken [71].

Two key papers on the use of the FEM in the solution of FPK equations are by Langley [52] and Langtangen [53]. Much work has been undertaken since, for example Spencer and Bergman [83] have used and expanded upon the techniques of Langley and Langtangen in studying oscillators.
2.1.2 FEM for One Dimensional Systems

FEM Approach

The Bubnov-Galerkin Finite Element Method (BGFEM) is one of a class of Finite Element Methods (FEM) that, in the one dimensional case, provide an approximate solution $p(x)$ of the FPK equation

$$\frac{\partial p}{\partial t} + \frac{\partial}{\partial x}\{a(x)p\} - \frac{1}{2} \frac{\partial^2}{\partial x^2}\{b^2(x)p\} = 0. \quad (2.1)$$

The asymptotic (stationary) solution is in the form

$$p(x) = \sum_{j=1}^{n} p_j H_j(x)$$

where the $H_j(x)$ are suitable FEM trial functions and the $p_j$ are unknown coefficients to be determined.

The BGFEM solution involves determining the unknown coefficients $p_j$ which are elements of the vector $p$ such that

$$Kp = 0 \quad (2.2)$$

and

$$C^T p = 1.$$ 

Here the elements $K_{ij}$ of matrix $K$ are given by

$$K_{ij} = \int_\Omega \frac{dH_i}{dx} \left\{a(x)H_j - \frac{1}{2} \frac{d}{dx}[c(x)H_j] \right\} dx$$

and the elements $C_i$ of matrix $C$ are given by

$$C_i = \int_\Omega H_i(x) \, dx.$$ 

Note here that the integrals are over the relevant implied maximal domain $\Omega$ for the system.

A suitable FEM trial function for one dimensional systems is the linear trial function

$$H_j(x) = \begin{cases} 
1 - \frac{|x - x_j|}{\Delta x}, & |x - x_j| \leq \Delta x \\
0, & \text{elsewhere}
\end{cases} \quad (2.3)$$

which forms a triangular pattern as illustrated in Figure 2.1.
Figure 2.1: Linear trial functions $H_i(x)$

Note that the trial function expression, equation (2.3), reduces to

$$H_j(x) = \begin{cases} 
1 - (x - x_j)/\Delta x, & 0 \leq x - x_j \leq \Delta x \\
1 + (x - x_j)/\Delta x, & 0 \leq x_j - x \leq \Delta x \\
0, & \text{elsewhere}
\end{cases}$$

and has derivative function

$$\frac{dH_j(x)}{dx} = \begin{cases} 
-1/\Delta x, & 0 < x - x_j < \Delta x \\
1/\Delta x, & 0 < x_j - x < \Delta x \\
0, & \text{elsewhere}
\end{cases}$$

Using equations (2.4) and (2.5) the elements of $C$ simplify to

$$C_i = \int_{\Omega} H_i(x) \, dx = \begin{cases} 
\Delta x/2, & i = 1 \text{ or } i = n \\
\Delta x, & i = 2, 3, \ldots, n - 1 \\
0, & \text{elsewhere}
\end{cases}$$

Example: Gaussian Density Solution

To describe the implementation of the FEM approach in the one dimensional case, consider the following simple example. The stationary solution of the FPK equation corresponding to the autonomous, scalar, single noise SDE

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

when

$$a(x) = -\left(\frac{x - \mu}{\sigma}\right) \quad \text{and} \quad c(x) = 2$$
for $-\infty < x < \infty$ is a Gaussian density solution. Specifically, when $\mu = 0$ and $\sigma = 1$, then $a(x) = -x$ and the stationary solution is

$$p_{\text{exact}}(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$  

Note that the solution for small $t$ is given by Mehler’s formula and a derivation is presented in Theorem 1.5.10 in Glimm and Jaffe [27].

In this example the elements $K_{ij}$ simplify to

$$K_{ij} = \int_{-\infty}^{\infty} \frac{dH_i}{dx} \left\{ -xH_j - \frac{d}{dx}H_j \right\} dx$$

where $H_j$ are the trial functions as given in equation (2.3). Note that the expressions for $K_{ij}$ are relatively straightforward in this case because of the simple form of $a(x)$ and $c(x)$. When expressions for $a(x)$ and $c(x)$ are more complex then Maple can be used to simplify the integrands before integrating to determine the coefficients $K_{ij}$. Note that even the small change of $\sigma$ from 1 to 2 requires a complete re-derivation of the $K_{ij}$.

As is necessary for the FEM approach, the infinite domain of interest must be truncated to a finite domain. Since $p_{\text{exact}}$ is known and $p_{\text{exact}}(x) \to 0$ as $x \to \pm \infty$, then following simple analysis, the domain was truncated to $[-8, 8]$. When the domain is not known, then more theoretical analysis needs to be undertaken or a number of exploratory runs of the FEM scheme may be necessary to resolve the domain size issue. In the examples that follow in later chapters, the simulation package SIMULINK will be used to first obtain details of the domain of interest.

Using five nodes for the truncated domain $\Omega = \{x : x \in [-8, 8]\}$, then $X = \{x_1, x_2, x_3, x_4, x_5\} \equiv \{-8, -4, 0, 4, 8\}$ and $\Delta x = 4$.

Hence equation (2.2) becomes

$$\frac{1}{12} \begin{bmatrix}
-43 & -29 & 0 & 0 & 0 \\
43 & 10 & -5 & 0 & 0 \\
0 & 19 & 10 & 19 & 0 \\
0 & 0 & -5 & 10 & 43 \\
0 & 0 & 0 & -29 & -43
\end{bmatrix} \begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
p_4 \\
p_5
\end{bmatrix} = \mathbf{0}.$$  

Note that the matrix $\mathbf{K}$ is singular. However the above system can be solved by setting $p_1 = \xi$, an unknown constant, and then solving the reduced system
\[
\begin{bmatrix}
10 & -5 & 0 & 0 \\
19 & 10 & 19 & 0 \\
0 & -5 & 10 & 43 \\
0 & 0 & -29 & -43
\end{bmatrix}
\begin{bmatrix}
p_2 \\
p_3 \\
p_4 \\
p_5
\end{bmatrix}
= -\frac{\xi}{12}
\begin{bmatrix}
43 \\
0 \\
0
\end{bmatrix}.
\]

This leads to the solution

\[
p = \begin{bmatrix}
\xi \\
-1.482758621\xi \\
5.634482759\xi \\
-1.482758621\xi \\
\xi
\end{bmatrix}
\]

Since the solution must satisfy \(C^T p = 1\), then \(\xi\) is readily determined (\(\xi = 0.068139097\)) so

\[
p = \begin{bmatrix}
0.0681390970 \\
-0.1010338346 \\
0.3839285715 \\
-0.1010338346 \\
0.0681390970
\end{bmatrix}.
\]

Graphically, the resulting approximate solution and the exact solution are displayed for comparison in Figure 2.2.

Note that this is only a rough approximation to the known solution as it is only based on five nodes. Increasing this to 9, 17, 33 and then 65 nodes on \(x \in [-8, 8]\) leads to better approximations as displayed in Figures 2.3 to 2.6 respectively.
Figure 2.2: Exact pdf (line) against 5 node FEM approximation (circles)

Figure 2.3: Exact pdf (line) against 9 node FEM approximation (circles)
Figure 2.4: Exact pdf (line) against 17 node FEM approximation (circles)

Figure 2.5: Exact pdf (line) against 33 node FEM approximation (circles)
Figure 2.6: Exact pdf (line) against 65 node FEM approximation (circles)
2.1.3 FEM for Higher Dimensional Systems

The FEM approach when analysing two and three dimensional systems follows that of the one dimensional case. However, as the dimension increases, so too does the complexity, especially with regard to visualisation and element design. Details of the approach in two dimensional systems will be presented in Chapter 5.

Overall, for higher than two dimensional systems, the FEM is very cumbersome but is still often used, especially in industrial applications.
2.2 Taylor Series Time Discretisation Approach

2.2.1 Underlying Concepts

Preliminaries

In its most trivial form, a stochastic process is a sequence of random variables $X_i$ that describe the evolution of a probabilistic system over the discrete, monotonically increasing time intervals $t_i$ ($i = 0, 1, 2, \ldots$). A more general form is one where each random variable depends to some degree on its predecessor(s) and purely random influences. If a stochastic process has an unbounded time domain then it is referred to as a continuous stochastic process.

Let $T$ denote the time domain of interest and let $(\Omega, \mathcal{A}, \mathcal{P})$ represent the underlying probability space. Here $\Omega$ denotes the sample space, $\mathcal{A}$ is a subset of $\Omega$ comprising the events of interest, (the event space) and finally $\mathcal{P}$ is the probability measure.

Mathematically, a stochastic process $X = \{X(t), t \in T\}$ is defined by the mapping $X : T \times \Omega \rightarrow \mathbb{R}$ where $X_t \equiv X(t) = X(t, \cdot)$ is a random variable for each $t \in T$.

For each $\omega \in \Omega$, then $X(\cdot, \omega) : T \rightarrow \mathbb{R}$ is called a sample path, or trajectory, of the stochastic process. A simple example is the random walk process.

An important class of stochastic processes is the Standard Wiener Process, $W = \{W(t), t \geq 0\}$ which is a Gaussian (Normally) distributed process, $W(t) \sim N(0, t)$, with independent increments such that

1. $W(0) = 0$
2. $E(W(t)) = 0$
3. $\text{Var}(W(t) - W(s)) = t - s$ for all $t, s \in [0, T]$.

An example of a sample path of the Wiener process, as generated by MATLAB, is presented in Figure 2.7. Note that the Wiener process is continuous but nowhere differentiable.

The Standard Wiener Process was originally proposed as a mathematical model to describe Brownian motion. Note that it can be approximated by a scaled random walk.
Figure 2.7: A typical sample path $W_t$ of the Wiener process

The Standard Wiener process has transition density

$$p(s, x; t, y) = \frac{1}{\sqrt{2\pi(t-s)}} \exp \left( -\frac{(y-x)^2}{2(t-s)} \right)$$

which satisfies the partial differential equations

$$\frac{\partial p}{\partial t} - \frac{1}{2} \frac{\partial^2 p}{\partial y^2} = 0 \quad \text{for } (s, x) \text{ fixed}$$
$$\frac{\partial p}{\partial s} + \frac{1}{2} \frac{\partial^2 p}{\partial x^2} = 0 \quad \text{for } (t, y) \text{ fixed.}$$

The Standard Wiener process is an example of a homogeneous continuous time Markov process as its transition densities depend only on the time difference $t - s$ rather than on the individual values of $t$ and $s.$

In general, if a Markov process with given transition densities $p(s, x; t, y)$ for any $\epsilon > 0,$ $s \geq 0$ and $x \in \mathbb{R}$ satisfies

$$\lim_{t \to s} \frac{1}{t-s} \int_{|y-x|>\epsilon} p(s, x; t, y) \, dy = 0$$
$$\lim_{t \to s} \frac{1}{t-s} \int_{|y-x|>\epsilon} (y-x)p(s, x; t, y) \, dy = a(s, x)$$
$$\lim_{t \to s} \frac{1}{t-s} \int_{|y-x|>\epsilon} (y-x)^2p(s, x; t, y) \, dy = b^2(s, x)$$

then the Markov process is known as a diffusion process. A physical example of such a process is the dispersion of a drop of ink in a container of water. The ink spreads out and becomes more and more evenly dispersed over time.
For the Markov diffusion process, the functions $a(s, x)$ and $b(s, x)$, where $s$ is time and $x$ is position, are called the drift and diffusion coefficient (or just diffusion) respectively.

Analogous to the Standard Wiener Process, if $a$ and $b$ are moderately smooth functions then the transition densities $p(s, x; t, y)$ satisfy the partial differential equations

$$
\frac{\partial p}{\partial t} + \frac{\partial}{\partial y} \left\{ a(t, y)p \right\} - \frac{1}{2} \frac{\partial^2}{\partial y^2} \left\{ b^2(t, y)p \right\} = 0 \quad \text{for } (s, x) \text{ fixed}
$$

$$
\frac{\partial p}{\partial s} + a(s, x) \frac{\partial p}{\partial x} + \frac{1}{2} b^2(s, x) \frac{\partial^2 p}{\partial x^2} = 0 \quad \text{for } (t, y) \text{ fixed}.
$$

These are the Kolmogorov forward and backward equations respectively. Note that the forward equation is often called the Fokker–Planck or Fokker–Planck–Kolmogorov (FPK) equation in mathematical physics. Note also that the Markov diffusion process satisfies the Itô-SDE, equation (2.8), which will be developed in the next section and as such there is a correspondence between the FPK equation and the Itô-SDE.

Finally, White Noise or Gaussian White Noise, is a “wide sense stationary process” which means it is a process in probabilistic equilibrium with respect to its first and second moments. It has mean zero and constant non zero spectral density. It takes the descriptor White as it has the same characteristics as white light.

**Stochastic Calculus**

Just as the general ODE

$$
\frac{dx}{dt} = a(t, x) \quad \text{with } x = x_0 \text{ at } t = t_0
$$

can be expressed as an integral equation of the form

$$
x(t) = x_0 + \int_{t_0}^{t} a(s, x(s)) \, ds,
$$

then the general stochastic differential equation (SDE) used to describe Brownian motion

$$
\frac{dX_t}{dt} = a(t, X_t) + b(t, X_t) \xi_t
$$

(2.6)

(where $\xi_t$ is the noise process) can be written in integral form as

$$
X_t(\omega) = X_{t_0}(\omega) + \int_{t_0}^{t} a(s, X_s(\omega)) \, ds + \int_{t_0}^{t} b(s, X_s(\omega)) \xi_s(\omega) \, ds
$$

(2.7)

for each sample path $\omega$. 
Note that SDEs differ to Random Differential Equations (RDEs) in that the RDEs are simply ODEs with random coefficients and regular random forcing terms. For example the one dimensional RDE

\[
\frac{dx}{dt} = a(t, \omega)x + b(t, \omega)
\]

has solutions \( x = x(t, \omega) \) which are differentiable functions in \( t \). Standard ODE solution techniques can be used with RDEs along individual sample paths.

For \( a \equiv 0 \) and \( b \equiv 1 \) then equation (2.6) reduces to

\[
\frac{dX_t}{dt} = \xi_t
\]

which implies that \( \xi_t \) can be considered as the derivative of Brownian motion, or the derivative of the Wiener process, ie.,

\[
dW_t = \xi_t \, dt.
\]

Thus in the integral form of the equation (2.7), \( \xi_s(\omega) \) \( ds \) can be replaced by \( dW_s(\omega) \), ie.,

\[
X_t(\omega) = X_{t_0}(\omega) + \int_{t_0}^{t} a(s, X_s(\omega)) \, ds + \int_{t_0}^{t} b(s, X_s(\omega)) \, dW_s(\omega).
\]  \hspace{1cm} (2.8)

However, the Wiener process is nowhere differentiable, and in fact it is not even of bounded variation on any bounded interval. Hence the second integral above cannot be evaluated as a standard Riemann integral.

Itô studied this integral by firstly considering the case where \( b(t, x) \equiv b, \) a constant. In this case

\[
\int_{t_0}^{t} b(s, X_s(\omega)) \, dW_s(\omega) \equiv b \int_{t_0}^{t} dW_s(\omega) = b\{W_t(\omega) - W_{t_0}(\omega)\}.
\]

Simply put, Itô determined that if

\[
dx_t(\omega) = a(t, \omega) \, dt + b(t, \omega) \, dW_t(\omega),
\]

then the Itô formula (Stochastic Chain Rule) provides

\[
dY_t = \left\{ \frac{\partial U}{\partial t} + a \frac{\partial U}{\partial x} + \frac{1}{2} b^2 \frac{\partial^2 U}{\partial x^2} \right\} dt + b \frac{\partial U}{\partial x} dW_t
\]  \hspace{1cm} (2.9)

where \( Y_t(\omega) = U(t, X_t(\omega)) \). For a thorough description see Kloeden & Platen [45].

As an example, if \( a \) and \( b \) are constant, then the solution of the Langevin Equation

\[
dx_t = -a X_t + b dW_t
\]
\[ X_t = e^{-at}X_0 + e^{-at} \int_0^t b e^{as} dW_s \]

which is an Ornstein-Uhlenbeck process. Instead, if \( b(t, X) = b \cdot X \) then the solution of the autonomous homogeneous linear SDE

\[ dX_t = aX_t + b \cdot X_t dW_t \]

is

\[ X_t = X_0 \exp \left[ \left( a - \frac{1}{2} b^2 \right) t + b W_t \right]. \]

The first example, where \( b(t, X) = b(t) \), is a case of additive noise whereas the second is an example of multiplicative noise.

Another approach is to use the inter-related Stratonovich calculus which uses the classical deterministic Chain Rule. This will not be discussed as the descriptions in this thesis are all based on the Itô calculus approach since Itô calculus is directly linked to diffusion processes and martingale theory of stochastic analysis.

To date we have considered simple SDEs. Other examples include

\[ dX_t = a(X_t) dt + \sum_{j=1}^m b^j(X_t) dW^j_t \]

which has an \( m > 1 \) dimensional Wiener process \( W_t = (W^1_t, W^2_t, \ldots, W^m_t) \) and the non autonomous version

\[ dX_t = a(t, X_t) dt + \sum_{j=1}^m b^j(t, X_t) dW^j_t. \]

Finally, the general case of \( d \) component vector SDEs which can be considered as the system of \( d \) SDEs with multidimensional noise (\( m \) sources) is given by

\[ dX_t^i = a^i(t, X) dt + \sum_{j=1}^m b^{ij}(t, X) dW^j_t \]

for \( i = 1, 2, \ldots, d \) and \( X_t = (X^1_t, X^2_t, \ldots, X^d_t) \).

**Taylor Series Expansions**

Consider the initial value ODE system

\[ \frac{dX_t}{dt} = a(X_t), \quad X_t(t_0) = X_{t_0} \]
over the time domain $[t_0, T]$ with $0 \leq t_0 \leq T$. This can be written as an integral equation as

$$X_t = X_{t_0} + \int_{t_0}^{t} a(X_s) \, ds.$$  

Applying the Chain Rule to a continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ gives

$$\frac{d}{dt} f(X_t) = \frac{d}{dX_t} f(X_t) \frac{dX_t}{dt},$$

i.e.

$$\frac{d}{dt} f(X_t) = \frac{d}{dX_t} f(X_t) \cdot a(X_t).$$  \hspace{1cm} (2.10)

This can be written using operator notation by

$$\mathcal{L} f = af^\prime$$

where $\mathcal{L} = d/dt$ and $f^\prime = df/dx$. Using the operator notation, the integral equation form of equation (2.10) is then

$$f(X_t) = f(X_{t_0}) + \int_{t_0}^{t} \mathcal{L} f(X_s) \, ds.$$  \hspace{1cm} (2.11)

Consider the case when $f(x) \equiv x$, (which is continuously differentiable). This implies that $\mathcal{L} f = a$ so the integral equation (2.11) reduces to

$$X_t = X_{t_0} + \int_{t_0}^{t} a(X_s) \, ds.$$  \hspace{1cm} (2.12)

Now applying equation (2.11) to the function $a(X_s)$ leads to

$$a(X_t) = a(X_{t_0}) + \int_{t_0}^{s} \mathcal{L} a(X_u) \, du.$$  \hspace{1cm} (2.13)

Hence substituting equation (2.12) into equation (2.13) gives

$$X_t = X_{t_0} + \int_{t_0}^{t} \left[ a(X_{t_0}) + \int_{t_0}^{s} \mathcal{L} a(X_u) \, du \right] \, ds$$

$$= X_{t_0} + a(X_{t_0}) \int_{t_0}^{t} \, ds + \int_{t_0}^{t} \int_{t_0}^{s} \mathcal{L} a(X_u) \, du \, ds$$

$$= X_{t_0} + a(X_{t_0})(t - t_0) + R$$

with the remainder $R$ given by

$$R = \int_{t_0}^{t} \int_{t_0}^{s} \mathcal{L} a(X_u) \, du \, ds.$$  

Neglecting the remainder leads to the deterministic Euler Method for ODEs.
Including the remainder and applying equation (2.11) to $\mathcal{L}a(X_t)$ in the double integral in $R$ leads to

$$X_t = X_{t_0} + a(X_{t_0})(t - t_0) + \mathcal{L}a(X_{t_0}) \int_{t_0}^{t} \int_{t_0}^{s} du \, ds + R^*$$

$$= X_{t_0} + a(X_{t_0})(t - t_0) + \mathcal{L}a(X_{t_0}) \frac{(t - t_0)^2}{2} + R^*$$

where $R^*$, the remainder, is given by

$$R^* = \int_{t_0}^{t} \int_{t_0}^{s} dv \, du \, ds.$$ 

Repeating these step $r$ times leads to the general integral form of the Taylor series for deterministic calculus

$$f(X_t) = f(X_{t_0}) + \sum_{l=1}^{r} \frac{(t - t_0)^l}{l!} \mathcal{L}^l f(X_{t_0}) + \int_{t_0}^{t} \cdots \int_{t_0}^{s_{r+1}} \mathcal{L}^{r+1} f(X_{s_1}) \, ds_1 \, ds_2 \cdots ds_{r+1}$$

where $f$ is assumed to be at least $r + 1$ times continuously differentiable. The Taylor series is very important in deterministic methods of numerical analysis since it is used to help determine the order of numerical schemes that are based on heuristic methods.

The process outlined above can be applied to expand smooth functions of an Itô process about a given value. Care needs to be taken to ensure consistency with Itô calculus.

Consider the integral form of a general autonomous SDE

$$X_t = X_{t_0} + \int_{t_0}^{t} a(X_s) \, ds + \int_{t_0}^{t} b(X_s) \, dW_s.$$  \hspace{1cm} (2.14)

Then, for any twice differentiable function $f : \mathbb{R} \to \mathbb{R}$, the Itô formula is

$$f(X_t) = f(X_{t_0}) + \int_{t_0}^{t} \left\{ a(X_s)f'(X_s) + \frac{1}{2} b^2(X_s)f''(X_s) \right\} \, ds + \int_{t_0}^{t} b(X_s)f'(X_s) \, dW_s$$

$$= f(X_{t_0}) + \int_{t_0}^{t} \mathcal{L}^0 f(X_s) \, ds + \int_{t_0}^{t} \mathcal{L}^1 f(X_s) \, dW_s$$

where

$$\mathcal{L}^0 = a \frac{\partial}{\partial x} + \frac{1}{2} b^2 \frac{\partial^2}{\partial x^2} \quad \text{and} \quad \mathcal{L}^1 = b \frac{\partial}{\partial x}.$$ 

For $f(x) \equiv x$ then $\mathcal{L}^0 f = a$ and $\mathcal{L}^1 f = b$ so the original form of equation (2.14) is returned. Now, applying the Itô formula to $f = a$ and $f = b$ respectively leads to

$$X_t = X_{t_0} + \int_{t_0}^{t} \left\{ a(X_{t_0}) + \int_{t_0}^{s} \mathcal{L}^0 a(X_u) \, du + \int_{t_0}^{s} \mathcal{L}^1 a(X_u) \, dW_u \right\} \, ds$$

$$+ \int_{t_0}^{t} \left\{ b(X_{t_0}) + \int_{t_0}^{s} \mathcal{L}^0 b(X_u) \, du + \int_{t_0}^{s} \mathcal{L}^1 b(X_u) \, dW_u \right\} \, dW_s$$

$$= X_{t_0} + a(X_{t_0}) \int_{t_0}^{t} ds + b(X_{t_0}) \int_{t_0}^{t} dW_s + R.$$  \hspace{1cm} (2.15)
\[ X_t = X_{t_0} + a(X_{t_0}) \int_{t_0}^{t} \, ds + b(X_{t_0}) \int_{t_0}^{t} \, dW_s + R \]

where the remainder \( R \) is given by

\[
R = \int_{t_0}^{t} \int_{t_0}^{s} \mathcal{L}^0 a(X_u) \, du \, ds + \int_{t_0}^{t} \int_{t_0}^{s} \mathcal{L}^1 a(X_u) \, dW_u \, ds \\
+ \int_{t_0}^{t} \int_{t_0}^{s} \mathcal{L}^0 b(X_u) \, du \, dW_s + \int_{t_0}^{t} \int_{t_0}^{s} \mathcal{L}^1 b(X_u) \, dW_u \, dW_s.
\]

Truncating and neglecting the remainder \( R \) leads to the corresponding stochastic version of the Euler Method. Note that the stochastic Euler scheme resembles the deterministic Euler scheme.

The Itô formula can then be applied to any of the integrands in \( R \) to provide higher order expressions with any number of terms. Note that higher order expansions can contain higher order multiple stochastic integrals. The higher order multiple integrals are functionals of the underlying Wiener process and provide the information required to achieve the higher orders of convergence. For example, applying the Itô formula to the successive integrand functions of the stochastic Taylor Series expansion of \( f(t, X_t) \) about \((t_0, X_{t_0})\), ie.,

\[ \mathcal{L}^0 f, \ \mathcal{L}^j f, \ \mathcal{L}^0 \mathcal{L}^0 f, \ \mathcal{L}^0 \mathcal{L}^j f, \ \mathcal{L}^j \mathcal{L}^0 f, \ \text{and} \ \mathcal{L}^j \mathcal{L}^j f \]

and substituting them into the ongoing expansions leads to the stochastic integrals

\[
\int_{t_0}^{t} ds, \ \int_{t_0}^{t} dW_s, \ \int_{t_0}^{t} \int_{t_0}^{s} \, d\tau \, ds, \ \int_{t_0}^{t} \int_{t_0}^{s} \, dW_s^j \, d\tau \, ds, \ \text{and} \ \int_{t_0}^{t} \int_{t_0}^{s} \, d\tau \, dW_s^j
\]

respectively. Often such integrals cannot be expressed in terms of simple noise increments and must be approximated. The approximation of such stochastic integrals is a major field of study in its own right and readers are directed to Kloeden, Platen and Wright [48] and to Kloeden and Platen [45] and the reference therein.

**Stochastic Taylor Series Time Discretisations**

Consider the general one dimensional SDE

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

where \( X_t(t_0) = X_{t_0} = X_0 \) for \( t \in [t_0, T] \). This SDE has Itô process solution \( X = \{ X_t, t_0 \leq t \leq T \} \).
Recall the simplest form of the stochastic Taylor series,

\[ X_t = X_{t_0} + a(t_0, X_{t_0}) \int_{t_0}^t ds + b(t_0, X_{t_0}) \int_{t_0}^t dW_s + R. \]

By neglecting the remainder \( R \) we obtain the approximation

\[ X_t \approx X_{t_0} + a(t_0, X_{t_0})(t - t_0) + b(t_0, X_{t_0}) \int_{t_0}^t dW_s. \]  \( \text{(2.16)} \)

Consider a time discretisation of the interval \([t_0, T]\), say

\[ t_0 = \tau_0 < \tau_1 < \cdots < \tau_n < \cdots < \tau_N = T. \]

It then follows that the discrete time stochastic approximation corresponding to equation (2.16) is \( Y = \{Y(t), t_0 \leq t \leq T\} \) is given by

\[ Y_{n+1} = Y_n + a(\tau_n, Y_n)(\tau_{n+1} - \tau_n) + b(\tau_n, Y_n)(W_{\tau_{n+1}} - W_{\tau_n}) \]

for \( n = 0, 1, 2, \ldots, N - 1 \) and \( Y_0 = X_0 \) is the initial value.

For simplicity, assume a constant time discretisation difference \( \Delta = \tau_{n+1} - \tau_n \). Thus \( \tau_n = t_0 + n\Delta \). Letting \( \Delta W_n = W_{\tau_{n+1}} - W_{\tau_n} \) where \( E(\Delta W_n) = 0 \) and \( E((\Delta W_n)^2) = \Delta \) allows \( Y_{n+1} \) to be expressed in the form

\[ Y_{n+1} = Y_n + a(\tau_n, Y_n)\Delta + b(\tau_n, Y_n)\Delta W_n \]

or alternatively

\[ Y_{n+1} = Y_n + a\Delta + b\Delta W_n \quad \text{where} \quad n = 0, 1, 2, \ldots, N - 1. \]  \( \text{(2.17)} \)

Equation (2.17) is called the discrete time Euler or Euler-Maruyama approximation.

As a simple example of the use of the Euler–Maruyama scheme, consider the general autonomous SDE

\[ dX_t = aX_t + bX_t \ dW_t \]

where \( a \) and \( b \) are constants. The corresponding Euler–Maruyama scheme is

\[ Y_{n+1} = Y_n + aY_n\Delta + bY_n \Delta W_n. \]

This scheme was used to generate a single path for the case \( a = 1.0, b = 1.0, Y_0 = 1 \) and \( \Delta = \{0.002, 0.004\} \). This approximation was compared to the analytic solution

\[ Y_{n+1} = Y_0 \exp[0.5t + W_t] \]
Figure 2.8: Euler-Maruyama approximation against analytic solution

and to the analytic solution in the absence of noise $Y_{n+1} = Y_0 \exp(t)$. The results are presented graphically in Figure 2.8. It is clear that there is good correspondence between the Euler–Maruyama approximation and the analytic solution.

This example was simple enough that the analysis was undertaken using Microsoft Excel on a personal computer. However, the Euler–Maruyama scheme is usually of too low an order to apply to more complex systems and it can have difficulties with stiff systems. For more complex systems the usual approach is to write computer programs using a scientific programming language. Note that time discretisation solution schemes have some problems when the domain is finite and a particular realisation has, by some chance, moved outside the domain. To avoid this problem, structures need to be implemented in the programming code to neglect such realisations. For all analyses undertaken in this thesis the domains were unbounded and as such there was no need for any extra code for this task.

Convergence

The solution of

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

is dependent on the form of $a$ and $b$ and the Wiener process $W_t$. If all three are specified then the corresponding solution is referred to as the strong solution. If the Wiener process
is not specified then we denote the solution as the *weak* solution.

In the simplest form we use the schemes that have strong convergence (strong approximations) when we want a good approximation of an individual sample path. We use the schemes that have weak convergence (weak approximations) when we want a good approximation to the probability distribution of a set of sample paths, for example, the moments of the sample paths.

**Definition of Strong Convergence**

A general time discrete approximation $Y^{\delta}$ with maximum step size $\delta$ converges *strongly* to $X$ at time $T$ if

$$\lim_{\delta \to 0} E \left( \left| X_T - Y^{\delta}(T) \right| \right) = 0.$$ 

In particular, $Y^{\delta}$ is said to converge strongly *with order* $\gamma$, $(\gamma > 0)$, if there exists a positive constant $C_s$, which is independent of $\delta$, and there exists a $\delta_0 > 0$ such that for every $\delta \in (0, \delta_0)$

$$E \left( \left| X_T - Y^{\delta}(T) \right| \right) \leq C_s \delta^\gamma.$$ 

**Definition of Weak Convergence**

A general time discrete approximation $Y^{\delta}$ with maximum step size $\delta$ converges *weakly* to $X$ at time $T$ with respect to a class of test functions $g : \mathbb{R}^d \to \mathbb{R}$ if

$$\lim_{\delta \to 0} \left| E(g(X_T)) - E(g(Y^{\delta}(T))) \right| = 0.$$ 

In particular, $Y^{\delta}$ is said to converge weakly *with order* $\gamma$, $(\gamma > 0)$, if for each $g$ there exists a positive constant $C_w$, which is independent of $\delta$, and there exists a $\delta_0 > 0$ such that for every $\delta \in (0, \delta_0)$

$$\left| E(g(X_T)) - E(g(Y^{\delta}(T))) \right| \leq C_w \delta^\gamma.$$ 

Note that the class of functions $g$ includes the polynomials and polynomially bounded growth functions. Hence such weak convergence includes the convergence of all moments that exist.

When applied to the following numerical schemes, $Y^{\delta}(T)$ represents the approximation $Y^{\delta}_{N_T}$ where $N_T$ is the iteration index corresponding to time $T$ and $\delta$ is the maximum stepsize. In all simulations undertaken in this thesis the stepsize $\Delta$ was kept constant i.e., $\delta = \Delta$. 
2.2.2 Strong Taylor Schemes

Consider the general $d$-dimensional SDE with $m \geq 1$ noise sources and $X_t = (X^1_t, X^2_t, \ldots, X^d_t)$

$$dX^k_t = a^k(t, X_t)dt + \sum_{j=1}^m b^{kj}(t, X_t)dW^j_t$$

for $k = 1, 2, \ldots, d$.

Note that in this and subsequent sections the time step $\Delta$ will be assumed constant, and

$$\mathcal{L}^j = \sum_{k=1}^d b^{kj} \frac{\partial}{\partial x^k}$$

and

$$\mathcal{L}^0 = \frac{\partial}{\partial t} + \sum_{k=1}^d a^k \frac{\partial}{\partial x^k} + \frac{1}{2} \sum_{k,l=1}^d \sum_{j=1}^m b^{kj}b^{lj} \frac{\partial^2}{\partial x^k \partial x^l}.$$

The following are examples of strongly converging numerical schemes for the solution of SDEs. Note that the strong Euler–Maruyama and Milstein schemes described will be used in conjunction with a Galerkin approach to solve stochastic PDEs in Chapter 7.

**Order 0.5 Strong Scheme (The Euler–Maruyama Scheme)**

For $k = 1, 2, \ldots, d$ and initial values $(Y^1_0, Y^2_0, \ldots, Y^d_0)$, then the Euler–Maruyama scheme is given by

$$Y^{k+1}_n = Y^k_n + a^k(t_n, Y_n)\Delta + \sum_{j=1}^m b^{kj}(t_n, Y_n)\Delta W^j_n.$$

**Order 1.0 Strong Scheme (The Milstein Scheme)**

For $k = 1, 2, \ldots, d$ and initial values $(Y^1_0, Y^2_0, \ldots, Y^d_0)$, then the Milstein scheme is given by

$$Y^{k+1}_n = Y^k_n + a^k(t_n, Y_n)\Delta + \sum_{j=1}^m b^{kj}(t_n, Y_n)\Delta W^j_n + \sum_{j_1,j_2=1}^m \mathcal{L}^{j_1} b^{j_1,j_2}(t_n, Y_n) I_{(j_1,j_2)}$$

where $I_{(j_1,j_2)}$ represent the double stochastic integrals

$$I_{(j_1,j_2)} = \int_{t_n}^{t_{n+1}} \int_{s_1}^{s_2} dW^{j_1}_{s_2} dW^{j_2}_{s_1}.$$

When $j_1 = j_2 = j$ then $I_{(j_1,j_2)}$ simplifies to

$$I_{(j,j)} = \frac{1}{2} (\langle \Delta W^j \rangle^2 - \Delta).$$

Hence when $d = m = 1$ the Milstein scheme reduces to

$$Y_{n+1} = Y_n + a(t_n, Y_n)\Delta + b\Delta W + \frac{1}{2} b(t_n, Y_n)b'(t_n, Y_n) [\langle \Delta W \rangle^2 - \Delta].$$
Higher Order Strong Schemes

As an extension into higher order convergent schemes, consider the order 1.5 Strong scheme for the autonomous, scalar \((d = 1)\), single noise source \((m = 1)\) case. Adopting the notation \(a = a(Y_n)\) and \(b = b(Y_n)\) the scheme can be written as

\[
Y_{n+1} = Y_n + a\Delta + b\Delta W_n + \frac{1}{2}bb'[\left(\Delta W_n\right)^2 - \Delta] + a b' \Delta Z_n + \frac{1}{2} \left( aa' + \frac{1}{2} bb'' \right) \Delta^2
+ \left( ab' + \frac{1}{2} bb'' \right) (\Delta W_n \cdot \Delta - \Delta Z_n) + \frac{1}{2} b(b b'' + (b')^2) \left[ \frac{1}{3} (\Delta W_n)^2 - \Delta \right] \Delta W_n
\]

where

\[
\Delta Z_n = I_{(1, 0)} = \int_{\tau_n}^{\tau_{n+1}} \int \Delta dW_s.
\]

Note that \(\Delta Z_n\) is Gaussian with \(E(\Delta Z_n) = 0\) and \(E(\Delta Z_n^2) = \Delta_n^2/3\), therefore

\[
\Delta Z_n \sim N(0, \Delta_n^2/3).
\]

Moreover \(\Delta Z_n\) and \(\Delta W_n\) are correlated with \(E(\Delta W_n \Delta Z_n) = \frac{1}{6} \Delta^2\). Hence \(\Delta W_n\) and \(\Delta Z_n\) can be generated from two standard normal (Gaussian) distributed variables \(G_{1n}\) and \(G_{2n}\) by

\[
\Delta W_n = \Delta^{1/2} G_{1n} \quad \text{and} \quad \Delta Z_n = \frac{1}{2} \Delta^{3/2} \left( G_{1n} + \frac{1}{\sqrt{3}} G_{2n} \right).
\]  (2.18)

2.2.3 Weak Taylor Schemes

Consider the general \(d\)-dimensional SDE with \(m \geq 1\) noise sources and \(X_t = \left(X_1^t, X_2^t, \ldots, X_d^t\right)\), where

\[
dX_t^k = a^k(t, X_t) dt + \sum_{j=1}^{m} b^{k,j}(t, X_t) dW_t^j
\]

and \(k = 1, 2, \ldots, d\). In the following expressions I will use the simplified notation

\[
a^k \equiv a^k(t, Y_n), \quad b^{k,j} \equiv b^{k,j}(t, Y_n) \quad \text{and} \quad b^k \equiv b^k(t, Y_n).
\]

The following are examples of weakly converging numerical schemes for the solution of SDEs. The majority of analyses undertaken in this thesis involve the use of order 2.0 weak schemes, however others are included for completeness.

Order 1.0 Weak Scheme (The Euler Scheme)

For \(k = 1, 2, \ldots, d\) and initial values \((Y_{0}^{1}, Y_{0}^{2}, \ldots, Y_{0}^{d})\), then the Euler scheme is given by

\[
Y_{n+1}^k = Y_n^k + a^k \Delta + \sum_{j=1}^{m} b^{k,j} \Delta W_n^j.
\]
Note that this expression is identical to the strong 0.5 order Euler-Maruyama scheme. However in the weak case, simpler random variables can be used instead of Gaussian variables for $\Delta \tilde{W}_n$. Specifically the two-point random variables
\[
\Pr \left( \Delta \tilde{W}_n = \pm \sqrt{\Delta} \right) = \frac{1}{2}
\]
can be used. This does not alter the order 1.0 of weak convergence.

**Order 2.0 Weak Scheme**

The general order 2.0 weak scheme can be given in terms of its $k$-th component by
\[
Y_{n+1} = Y_n + a^k \Delta + \frac{1}{2} \mathcal{L} a^k \Delta^2 \\
+ \sum_{j=1}^{m} \left\{ b^{k,j} \Delta \tilde{W}_n^j + \mathcal{L} b^{k,j} I_{(0,j)} + \mathcal{L} a^k I_{(j,0)} \right\} \\
+ \sum_{j_1,j_2=1}^{m} \mathcal{L}^{j_1} b^{k,j_2,j_2} I_{(j_1,j_2)}.
\] (2.19)

Here the $I_{(s_1,s_2)}$ represent multiple Itô integrals involving different components of the Wiener process. As mentioned in Section 2.2.1 these multiple Itô integrals are typically difficult to generate. However in the weak case, simpler random variables approximating these Itô integrals can be used. This leads to more practical expressions such as
\[
Y_{n+1} = Y_n + a^k \Delta + \frac{1}{2} \mathcal{L} a^k \Delta^2 \\
+ \sum_{j=1}^{m} \left\{ b^{k,j} + \frac{1}{2} \Delta \left( \mathcal{L} b^{k,j} + \mathcal{L} a^k \right) \right\} \Delta \tilde{W}_n^j \\
+ \frac{1}{2} \sum_{j_1,j_2=1}^{m} \mathcal{L}^{j_1} b^{k,j_2,j_2} \left( \Delta \tilde{W}_n^{j_1} \Delta \tilde{W}_n^{j_2} + V_{j_1,j_2} \right).
\] (2.20)

Here simpler random variables $\Delta \tilde{W}_n$ are three-point distributed, ie.
\[
\Pr \left( \Delta \tilde{W}_n = \pm \sqrt{3\Delta} \right) = \frac{1}{6} \quad \text{and} \quad \Pr \left( \Delta \tilde{W}_n = 0 \right) = \frac{2}{3}.
\]

Furthermore, for $j_2 = j_1 + 1, \ldots, m$ and $j_1 = 1, 2, \ldots, m$, then
\[
\Pr (V_{j_1,j_2} = \pm \Delta) = \frac{1}{2}
\]
with $V_{j_1,j_1} = V_{j_2,j_2} = -\Delta$ and $V_{j_1,j_2} = -V_{j_2,j_1}$. Again the use of simpler random variables does not alter the order of convergence.

Finally note that other simplifications are possible in special cases such as in the one dimensional autonomous case with only one noise source. In this case the scheme can be written as
\[ Y_{n+1} = Y_n + a \Delta + b \Delta W_n + \frac{1}{2} b b' \left[ (\Delta W_n)^2 - \Delta \right] + a' b \Delta Z_n + \frac{1}{2} \left( a a' + \frac{1}{2} a'' b^2 \right) \Delta^2 + \left( a b' + \frac{1}{2} b' b^2 \right) (\Delta W_n \Delta - \Delta Z_n). \]

A further simplification is also possible by using simpler expressions for the \( \Delta W_n \) and \( \Delta Z_n \). These simpler expressions can be used as they are mathematically suitable under the weak convergence criterion. Specifically

\[ \Delta W_n = \Delta \tilde{W}_n \quad \text{and} \quad \Delta Z_n = \frac{1}{2} \Delta \tilde{W}_n \cdot \Delta \]

(2.21)
can be used. The scheme then reduces to

\[ Y_{n+1} = Y_n + a \Delta + b \Delta \tilde{W}_n + \frac{1}{2} b b' \left[ (\Delta \tilde{W}_n)^2 - \Delta \right] + \frac{1}{2} \left( a a' + \frac{1}{2} a'' b^2 \right) \Delta \tilde{W}_n \Delta + \frac{1}{2} \left( a a' + \frac{1}{2} a'' b^2 \right) \Delta^2. \]

**Higher Order Weak Schemes**

The general order 3.0 weak scheme for \( m, d = 1, 2, \ldots \) is given in its \( k \)-th component form by

\[ \dot{Y}_{n+1}^k = \dot{Y}_n^k + a^k \Delta + \sum_{j=1}^m b^k_j \Delta \tilde{W}_n^j + \sum_{j=1}^m \mathcal{L}^j a^k I_{j,0}^k + \sum_{j_1=0}^m \sum_{j_2=1}^m \mathcal{L}^{j_1} b^{j_2} I_{j_1,j_2}^{j_2} + \sum_{j_1=0}^m \sum_{j_2=1}^m \mathcal{L}^{j_1} a^k I_{j_1,j_2}^{j_2,0} \]

This expression contains stochastic triple integrals \( I_{(s_1,s_2,s_3)} \) which are difficult to generate.

Some simplification is possible in special cases. For example, for the scalar single noise case, ie. \( d = m = 1 \), Kloeden and Platen [45] suggest the scheme

\[ Y_{n+1} = Y_n + a \Delta + b \Delta \tilde{W}_n + \frac{1}{2} b b' \left[ (\Delta \tilde{W}_n)^2 - \Delta \right] + \mathcal{L}^1 a \Delta \tilde{Z}_n + \frac{1}{2} \mathcal{L}^0 a \Delta^2 \]

\[ + \mathcal{L}^0 (\Delta \tilde{W}_n \cdot \Delta - \Delta \tilde{Z}_n) + \frac{1}{6} (\mathcal{L}^0 \mathcal{L}^0 b + \mathcal{L}^0 \mathcal{L}^1 a + \mathcal{L}^1 \mathcal{L}^0 a) \Delta \tilde{W}_n \Delta^2 \]

\[ + \frac{1}{6} (\mathcal{L}^1 \mathcal{L}^1 a + \mathcal{L}^1 \mathcal{L}^0 b + \mathcal{L}^0 \mathcal{L}^1 b) \left[ (\Delta \tilde{W}_n)^2 - \Delta \right] \Delta \]

\[ + \frac{1}{6} \mathcal{L}^0 \mathcal{L}^0 a \Delta^3 + \frac{1}{6} \mathcal{L}^1 \mathcal{L}^1 b \left[ (\Delta \tilde{W}_n)^2 - 3 \Delta \right] \Delta \tilde{W}_n \]

where the correlated random variables \( \Delta \tilde{W}_n \) and \( \Delta \tilde{Z} \) can be distributed as in equations 2.18 or 2.21.
2.2.4 Variations of Order 2.0 Weak Schemes

As the order 2.0 weak schemes will be used in the numerical investigations of one, two and three dimensional SDEs I decided to investigate the extension of classical deterministic numerical scheme variants such as explicit, implicit, etc. These are described in the following sections.

Order 2.0 Explicit Weak Scheme (Platen)

The general multidimensional case with \( m \) noise sources \( (d, m = 1, 2, \ldots) \) has \( k \)-th component

\[
Y_{n+1}^k = Y_n^k + \frac{1}{2} \left( a^k(\bar{Y}) + a^k \right) \Delta \\
+ \frac{1}{4} \sum_{j=1}^{m} \left\{ b^{k,j}(\bar{R}_n^+ + b^{k,j}(\bar{R}_n^-) + 2b^{k,j} \right\} \Delta \bar{W}_n^j \\
+ \sum_{r=1, r \neq j}^{m} \left\{ b^{k,r}(\bar{U}_n^+ + b^{k,r} + 2b^{k,r} \right\} \Delta \bar{W}_n^r \\
+ \frac{1}{4} \sum_{j=1}^{m} \sum_{r=1, r \neq j}^{m} \left\{ b^{k,j}(\bar{R}_n^+ + b^{k,j} + 2b^{k,j} \right\} \Delta \bar{W}_n^j \Delta \bar{W}_n^r + V_{r,j} \right\} \Delta^{-1/2}
\]

Here

\[
\bar{Y} = Y_n^k + a^k \Delta + \sum_{j=1}^{m} b^{k,j} \Delta \bar{W}_n^j
\]

and

\[
\bar{R}_n^k = Y_n^k + a^k \Delta \pm b^{k,j} \sqrt{\Delta} \quad \text{and} \quad \bar{U}_n^k = Y_n^k \pm b^{k,j} \sqrt{\Delta}.
\]

In the additive noise case, \( b(t, x) \equiv b(t) \), this simplifies to

\[
Y_{n+1}^k = Y_n^k + \frac{1}{2} \left[ a^k(\bar{Y}) + a^k \right] \Delta + \sum_{j=1}^{m} b^{k,j} \Delta \bar{W}_n^j.
\]

For the above case with \( d = 1, 2, \ldots \) and one noise source \( (m = 1) \) this simplifies to

\[
Y_{n+1}^k = Y_n^k + \frac{1}{2} \left( a^k(\bar{Y}) + a^k \right) \Delta \\
+ \frac{1}{4} \left( b^k(\bar{Y}^+) + b^k(\bar{Y}^-) + 2b^k \right) \Delta \bar{W} \\
+ \frac{1}{4} \left( b^k(\bar{W}^+) - b^k(\bar{W}^-) \right) \left[ (\Delta \bar{W})^2 - \Delta \right] \Delta^{-1/2}
\]

where

\[
\bar{Y} = Y_n^k + a^k \Delta + b^k \Delta \bar{W}_n
\]
and

\[ Y^\pm = Y^k_n + a^k \Delta \pm b^k \sqrt{\Delta}. \]

Here \( \Delta \bar{W}_n \) can be either three point or Gaussian distributed random variables.

**Order 2.0 Implicit Weak Scheme**

In the multidimensional autonomous case, Milstein suggested the scheme

\[
Y^k_{n+1} = Y^k_n + [\alpha k(Y_n) + (1 - \alpha) a^k] \Delta + \frac{1}{2} \sum_{j_1, j_2=1}^m \mathcal{L}^{j_1} b^{j_1, j_2} \left( \Delta \bar{W}^{j_1}_n \Delta \bar{W}^{j_2}_n + V_{j_1, j_2} \right) \\
+ \sum_{j=1}^m \left[ b^{k, j} + \frac{1}{2} \left( \mathcal{L}^0 b^{k, j} + (1 - 2\alpha) \mathcal{L}^i a^k \right) \Delta \right] \Delta \bar{W}^j_n \\
+ \frac{1}{2} (1 - 2\alpha) \left[ \beta \mathcal{L}^0 a^k + (1 - \beta) \mathcal{L}^0 a^k(Y_{n+1}) \right] \Delta^2
\]  

(2.22)

where \( \alpha, \beta \in [0, 1] \) are level of implicitness factors.

In the case where \( d = m = 1 \) the scheme reduces to

\[
Y_{n+1} = Y_n + a(Y_{n+1}) \Delta + b \Delta \bar{W} - \frac{1}{2} \left[ a(Y_{n+1}) a'(Y_{n+1}) + \frac{1}{2} b^2(Y_{n+1}) a''(Y_{n+1}) \right] \Delta^2 \\
+ \frac{1}{2} bb' \left[ (\Delta \bar{W})^2 - \Delta \right] + \frac{1}{2} \left[ a b' - a' b + \frac{1}{2} b'' b \right] \Delta \bar{W} \Delta
\]

where again \( \Delta \bar{W} \) is three point or Gaussian distributed random variables.

**(Platen’s) Order 2.0 Derivative Free Implicit Weak Scheme**

The autonomous multidimensional noise version has \( k \)-th component

\[
Y^k_{n+1} = Y^k_n + \frac{1}{2} \left[ a^k + a^k(Y_{n+1}) \right] \Delta \\
+ \frac{1}{4} \sum_{j=1}^m \left[ b^{k, j}(R^k_{n+1}) + b^{k, j}(-R^k_{n+1}) + 2b^{k, j} \right] \\
+ \sum_{r=1, r \neq j}^m \left[ \left( b^{k, j}(U^k_{n+1}) + b^r(U^k_{n+1}) - 2b^{k, j} \right) \Delta^{-1/2} \right] \Delta \bar{W}^r_n \\
+ \frac{1}{4} \sum_{j=1}^m \left[ \left\{ b^{k, j}(R^k_{n+1}) - b^r(R^k_{n+1}) \right\} \left( \Delta \bar{W}_n \right)^2 - \Delta \right] \Delta^{-1/2} \\
+ \frac{1}{4} \sum_{j=1}^m \left[ \sum_{r=1, r \neq j}^m \left\{ b^{k, j}(U^k_{n+1}) - b^{k, j}(U^k_{n+1}) \right\} \left( \Delta \bar{W}^r_n \Delta \bar{W}^r_n + V_{r, j} \right) \right] \Delta^{-1/2} \tag{2.23}
\]

where

\[
R^k_{\pm} = Y^k_n + a^k \Delta \pm b^{k, j} \sqrt{\Delta} \quad \text{and} \quad U^k_{\pm} = Y^k_n \pm b^{k, j} \sqrt{\Delta}.
\]
In the simpler case with only one noise source the scheme reduces to

\[
Y_{n+1} = \frac{1}{2} [a + a(Y_{n+1})] \Delta \\
+ \frac{1}{4} \left[ b^k(\bar{Y}^+) + b^k(\bar{Y}^-) + 2b^k \right] \Delta \bar{W}_n \\
+ \frac{1}{4} \left[ b^k(\bar{Y}^+) - b^k(\bar{Y}^-) \right] \left\{ (\Delta \bar{W}_n)^2 - \Delta \right\} \Delta^{-1/2}
\]

where

\[
\bar{Y}^\pm = Y^k_n + a^k \Delta \pm b^k \sqrt{\Delta}.
\]

The derivative free schemes are often mistakenly called Stochastic Runge-Kutta schemes in that they mimic the deterministic version by replacing derivatives with finite differences expressed in terms of supporting values. Note that Kloeden and Platen [45] have shown that such heuristic adaptations of deterministic numerical methods to SDEs have serious shortcomings and as such they will not be referred to as Runge-Kutta schemes in this thesis.

**Order 2.0 Weak Predictor–Corrector Scheme**

In the general case, the k-th component is given by the Predictor–Corrector pair

**Predictor:**

\[
\bar{Y}^k_{n+1} = Y^k_n + a^k \Delta + \Psi^k_n + \frac{1}{2} \mathcal{L}^0 a^k \Delta^2 + \frac{1}{2} \sum_{j=1}^m \mathcal{L}^j a^j \Delta \bar{W}^j_n \Delta
\]

**Corrector:**

\[
Y^k_{n+1} = Y^k_n + \frac{1}{2} \left[ a^k (\bar{Y}_{n+1}) + a^k \right] \Delta + \Psi^k_n
\]

where

\[
\Psi^k_n = \sum_{j=1}^m \left\{ b^k \Delta \mathcal{L}^0 b^k \Delta \right\} \Delta \bar{W}^j_n + \frac{1}{2} \sum_{j_1, j_2=1}^m \mathcal{L}^{j_1} b^{j_1, j_2} \left( \Delta \bar{W}^{j_1} \Delta \bar{W}^{j_2} + V_{j_1, j_2} \right).
\]

In the simplest case of a one dimensional SDE with one noise source the system reduces to

**Predictor:**

\[
\bar{Y}_{n+1} = Y_n + a \Delta + \Psi_n + \frac{1}{2} a' b \Delta \bar{W} \Delta + \frac{1}{2} \left( aa' + \frac{1}{2} a'' b^2 \right) \Delta^2
\]

**Corrector:**

\[
Y_{n+1} = Y_n + \frac{1}{2} \left[ a (\bar{Y}_{n+1}) + a \right] \Delta + \Psi_n
\]

where

\[
\Psi_n = b \Delta \bar{W}_n + \frac{1}{2} b b' \left( (\Delta \bar{W}_n)^2 - \Delta \right) + \frac{1}{2} \left[ a b' + \frac{1}{2} b'' b' \right] \Delta \bar{W}_n \Delta.
\]
Order 2.0 Linearly Implicit Scheme

Deuflhard [17] described the linearly implicit schemes for ODEs where only the linear part of the drift is expressed implicitly. This allows the implicit term to be solved algebraically. These schemes therefore provide a compromise between the increased stability but the extra computational effort required by the implicit schemes. See Petersen [66] for details of its application to SDEs and its order of convergence. For the SDE with drift written

\[ a(X) = A \cdot X + N(X) \]

where \( A \) is a constant and \( N(x) \) is nonlinear in \( x \) then the weak order 2.0 scheme for \( m = d = 1 \) reduces to

\[
Y_{n+1} = Y_n + (AY_{n+1} + N) \Delta + b \Delta W + \frac{1}{2}bb'\{(\Delta W)^2 - \Delta \} \\
+ (A + N')b \Delta Z + \frac{1}{2} \left( (AY_{n+1} + N)(A + N') + \frac{1}{2}N''b^2 \right) \Delta^2 \\
+ \left( (AY_{n+1} + N)b' + \frac{1}{2}b''b^2 \right) \{\Delta W \cdot \Delta - \Delta Z \}.
\]

In the case of additive noise the scheme further reduces to

\[
Y_{n+1} = Y_n + (AY_{n+1} + N) \Delta + b \Delta W + (A + N')b \Delta Z \\
+ \frac{1}{2} \left( (AY_{n+1} + N)(A + N') + \frac{1}{2}N''b^2 \right) \Delta^2.
\]

Given expressions for \( A, a, b, N \) and the derivatives of \( N \) the scheme simplifies to

\[
Y_{n+1} = \frac{f(Y_n)}{g(Y_n)} = \frac{Y_n + N \Delta + b \Delta W + (A + N')b \Delta Z + [N(A + N') + 1/2N''b^2] \Delta^2/2}{1 - A \Delta - A(A + N') \Delta^2/2}.
\]

Thus \( Y_{n+1} \) can be calculated algebraically and as such the linearly implicit scheme does not require the extra computational effort of the solution of a nonlinear equation at every time step.

2.2.5 The use of Maple

The Maple package stochastic provides a range of routines to expedite the process of analysing and generating numerical schemes for SDEs. As this package was only available to me after I had manually derived most of my schemes it was mainly used to validate my manual approach. These validations will be described in the subsequent relevant sections. Some extra routines in Maple, for example routines that derive derivative free schemes, were also written.
As an overview the stochastic package contains a variety of routines that can be used in the analysis of SDEs. For example, it contains routines that return the explicit solution of certain scalar SDEs, routines that can convert between the Itô and Stratonovich forms of an SDE, routines that apply the Itô formula through the application of the operators $\mathcal{L}^0$ and $\mathcal{L}^j$ and routines that derive the FPK equation associated with an SDE. Of particular use to my analysis in this thesis, the package also contains routines that returned numerical schemes of up to order 2.0 in the strong sense and up to order 3.0 in the weak sense.

The following example illustrates one of the routines in the stochastic package. The wktay2 procedure is used to determine the weak 2.0 order scheme. Consider the general SDE

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

where the drift and diffusion are given by

$$a = \begin{bmatrix} a^1 \\ a^2 \\ \vdots \\ a^N \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b^{1,1} & b^{1,2} & \ldots & b^{1,M} \\ b^{2,1} & b^{2,2} & \ldots & b^{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ b^{N,1} & b^{N,2} & \ldots & b^{N,M} \end{bmatrix}$$

respectively. The weak 2.0 order scheme can be generated by the Maple command

```maple
wktay2([[a1, a2, \ldots, aN], [[b11, b12, \ldots, b1M], \ldots, [[bN1, bN2, \ldots, bNM]]]);
```

Note that for all routines in the stochastic package the inputs of the diffusion $b$ are row-wise rather than column-wise. Care needs to be taken as this is not intuitive!

Consider the example of the scalar SDE with drift $a(X) = a \cdot X$ and diffusion $b(X) = b \cdot X$. Then the Maple command:

```maple
> wktay2([[a(x[1])], [[b(x[1])]]]);
```

results in the weak 2.0 order scheme
\[ \begin{align*}
1 &= (Y_{I_{n+1}} = Y_{I_n} + a(Y_{I_n}) \Delta_n \\
+ \frac{1}{2} (a(Y_{I_n})(\frac{\partial}{\partial Y_{I_n}} a(Y_{I_n})) + \frac{1}{2} b(Y_{I_n})^2 (\frac{\partial^2}{\partial Y_{I_n}^2} a(Y_{I_n}))) \Delta_n^2 + (b(Y_{I_n}) \\
+ \frac{1}{2}\Delta_n (a(Y_{I_n})(\frac{\partial}{\partial Y_{I_n}} b(Y_{I_n})) + \frac{1}{2} b(Y_{I_n})^2 (\frac{\partial^2}{\partial Y_{I_n}^2} b(Y_{I_n})) + b(Y_{I_n})(\frac{\partial}{\partial Y_{I_n}} a(Y_{I_n}))) \Delta \\
W_{si} + \frac{1}{2} b(Y_{I_n})(\frac{\partial}{\partial Y_{I_n}} b(Y_{I_n}))(\Delta^2 W_{si}^2 + V_{i,1})) \\
\end{align*} \]

Note that this corresponds to the scheme in section 2.2.3.

Readers are referred to the associated help files in the stochastic package for more details.
2.3 Monte Carlo Approach

Even though Monte Carlo methods will not be directly examined in this thesis, this section can be considered as background information to provide a general description of the method.

Monte Carlo methods comprise the branch of experimental mathematics which is concerned with experiments that use random numbers. The Monte Carlo method is particularly useful for diffusion type systems and as such it has direct application to SDEs. Monte Carlo methods can also be applied to deterministic systems such as solving multidimensional integrals, matrix inversions, solution of the Dirichlet problem etc. A classic textbook example is its use to determine the value of $\pi$ in the Buffon needle problem.

Historically, the name dates back to about 1944, during work on the atomic bomb. This work involved direct simulation of probabilistic systems concerned with random neutron diffusion in fusable material. See Hammersley and Handscomb [29] for a detailed history of the Monte Carlo method.

Typically, a Monte Carlo “scheme” has to be developed for each system and must take into account any special requirements of the particular system. In many applications of Monte Carlo methods the physical process is simulated directly, without the need to write down any governing differential equations. The number of realisations or “runs” of a Monte Carlo simulation needs to be large enough to provide the desired accuracy. Typically the total number of computations required in a Monte Carlo scheme is proportional to the dimension of the system and the number of runs undertaken. Even though a large number of runs is necessary, the structure of the Monte Carlo approach is ideally suited for use on parallel or distributed processing systems. However unfortunately, there is little computational efficiency to be gained since variance reduction techniques are necessary and need to be incorporated. This adds extra computational complexity both to the coding and the running of the Monte Carlo schemes.

Essentially, the Monte Carlo approach uses an averaging process to obtain the expectation of some functional $f$ of the solution $X$, ie.

$$E(f(X)) \approx \frac{1}{N} \sum_{j=1}^{N} f(X(\omega_j))$$

where $\omega_j$ is a random realisation of $X$. For example, $X$ may be Gaussian, ie. $X \sim \mathcal{N}(\mu, \sigma)$. 
If the probability distribution of $X$ is not known then something like a weak Euler scheme needs to be implemented. Typically, Monte Carlo methods require the following: an underlying pdf, a random number generator, a sampling rule, some type of tallying process and the inclusion of variance reduction techniques. Depending on the hardware available there is the option of parallelisation for use on modern supercomputers. Finally, once the results are computed, an error estimation is required to determine the accuracy of the generated results.

As an applied example, for a previous employer, I created a Monte Carlo based anti-ship missile defence simulation model. This model required a probabilistic determination of the radar system’s likely detection of an incoming missile. Once identified, the model then determined a number of probabilistic delays for the warship’s counter reaction to launching defensive missiles against the threat. It then required me to determine the likelihood, based on probability of kill tables, that the defensive missile would intercept the threat. Large numbers of realisations for various scenarios were run and the ship’s probability of survival, with associated confidence intervals, was calculated.

The SIMULINK approach in Section 1.3.2 can be considered, in the broadest sense of the Monte Carlo approach, as an Euler method based Monte Carlo scheme. As it is only of order 1.0 it is of little use as an accurate method but can be used to provide a first approximation. The best feature of a tool like SIMULINK is its ease of use and natural visual approach. It can be used to quickly generate approximate solutions to SDEs before deciding whether or not to undertake more complex and time consuming studies. For example SIMULINK can be used to determine the domain of the SDE’s solution. This information can then be used to optimise the FEM approach for more detailed study. For example, a refined FEM mesh can be placed over the more interesting parts of the solution domain.

Finally, since the stochastic time discretisation schemes rely on random numbers, some researchers consider them as Monte Carlo methods. This is definitely the case with the Order 0.5 Strong and Order 1.0 Weak schemes. However, numerical schemes and arithmetic averages based on higher order stochastic Taylor series utilise the underlying structure of the SDE. With the inclusion of simpler random variables the stochastic Taylor series schemes provide more accurate results more efficiently than do the Monte Carlo schemes.
2.4 Chapter Summary

This chapter has detailed the classical FEM approach for the one dimensional FPK equation. The solution of a simple one dimensional system was described in detail to fully step through the process. Further, the Stochastic Taylor series based time discretisation schemes that will be used in later analyses was developed. For the stochastic time discretisation approach I have defined the concepts of strong and weak convergence and specified the classes of systems that the resulting schemes should be applied to. The use of the symbolic manipulator MAPLE, as an adjunct to the scheme derivation process, was also introduced.
Chapter 3

Time Discretisation Approach Tools
3.1 Random Number Generation

3.1.1 Introduction

One of the most crucial requirements for the successful and accurate numerical solution of SDEs using the Taylor series based time discretisation schemes is the accurate modelling of the noise. The process of generating suitably “random enough and for long enough” random number sequences is important, especially as simulations of noisy systems on computers almost exclusively use pseudo-random number generators.

This section will provide a description of the history and background of random number generators. It will detail the classical types of random number generators and explain the tests on sequences of numbers to check whether they are “random enough”. Finally it will describe tests undertaken on seven different random number generators with reference to the generation of $N(0,1)$ distributed random numbers.

Note that the historical review that follows was completed in middle 1994. As a result more recent work has been omitted. For more recent work readers are directed to:

1. Ferrenberg and Landau [23] for a description of errors that can be introduced even with good random number generators.

2. Vattulainen et al [91] who present a set of physical tests of random number generators.

3. Kirkpatrick and Stoll [40] who present their version of a shift register type random number generator.

4. Entacher [22] who presents sequences that lead to poorly performing linear congruential type random number generators.

5. Matsumoto and Nishimura [60] who have developed the Mersenne Twister random number generator which has a “super astronomical” period of $2^{19937} - 1$.

3.1.2 The History of Random Number Generation

Random numbers were originally obtained by physical means: coin tosses, random draws, roulette wheels or counts of emitted particles. Tables of random digits were also compiled
from figures chosen at random from census reports or produced from electronic noise [43, 63, 69].

There were numerous disadvantages of such methods. The random numbers could not be analysed theoretically for randomness but had to be tested as produced. Tables of random numbers were too short for some applications and sequences of mechanically produced random numbers could not be repeated for testing purposes and had to be stored if results were to be validated [43, 63, 69].

With the development of digital computers, physical means of production became even less popular as it was awkward and slow to incorporate the generated random numbers into the computer. Another problem initially encountered was the high computer storage requirements [69, 49].

Therefore the development of the computer instigated the development of deterministic methods of producing random numbers. Random numbers produced in this manner have been dubbed “pseudo-random” numbers. They appear random, but cannot philosophically be called random [69, 49].

As discussed by Bratley et al [8], Von Neumann introduced the “middle square” method of generating random numbers. This technique involved choosing a starting number, squaring it, and then selecting the middle four digits as the next number to be input to the process. This proved to be less than useful as it degenerated to a predictable cycle or to zero [69, 49].

Another early deterministic method which is still in use is Lehmer’s linear congruential generator (refer to the next subsection for details). This has many versions including those which combine the output from two or more generators. In the 1960’s the form of the generator tended to be based on a divisor of , where was the word length of the machine. The mod operation could then easily be actioned by just retaining the last log bits of the dividend. An emphasis on the ease of coding and quick processing speed led to many bad generators being used. For example, “RANDU”, with generator \( X_{n+1} = 65539 \times X_n \mod 2^{31} \), has generated subsequences \( \{U_i, U_{i+1}, U_{i+2}\} \) which lie on one of 15 planes in the unit cube [65, 69].

Many algorithms for earlier microcomputers used this type of generator with less than perfect parameters producing obviously non-random output or unacceptable cyclic behaviour [70]. “Blatantly bad generators”, see [56], are reportedly still in use in some
applications and have been recommended in computer science textbooks over the years [65]. Much work has been done on examining the theoretically acceptable versions and analysing them statistically. Their acceptance seems to be based on their ease of implementation, reproducibility and ability to be understood as much as on evidence of their superior structure [55, 2, 70].

3.1.3 Types of Pseudo–Random Number Generators

Pseudo-random numbers are usually generated as Uniformly distributed numbers over the domain $[0, 1]$, i.e. $U(0, 1)$. They are then “converted” to take on other distributions as required. This section discusses some common means of obtaining the original $U(0, 1)$ distributed pseudo-random numbers and the next section describes the methods of converting these to Normally distributed pseudo-random numbers.

Of crucial importance of any RNG is the period. A period $2^{24}$ generator exhausting the whole sequence in about 30 seconds on a 750MHz Pentium III in 32-bit IEEE floating point format [67]. Thus for large simulations the choice of RNG is crucial.

Linear Congruential Generators (LCG’s)

As mentioned earlier, the Linear Congruential Generators (LCGs) are the most commonly used pseudo-random number generators. The general form of their sequence for $n = 0, 1, 2, \ldots$ is given by

$$X_{n+1} = a \cdot X_n + c$$  \hspace{1cm} (3.1)

where $0 \leq X_0 < m$ is the initial seed and $m > 0$, $0 < a < m$ and $0 \leq c < m$. The coefficient $a$ is called the multiplier, $c$ is called the increment and $m$ is called the modulus. It is usually normalised to a sequence between 0 and 1 by dividing by $m$. When the parameters are wisely chosen, the sequence is uniformly distributed [43, 2].

When $c$ is greater than zero, the LCG is called a “mixed” congruential generator. It is usually implemented with $m = 2^n$ and has period of $m$ (i.e., it does not repeat in less than $m$ iterations) when $c$ is odd and $a - 1$ is a multiple of 4. In this form it is easy to implement in assembly language but can alternate between odd and even numbers if the low order bits are used alone [2, 8].

When $c$ is zero and $m = 2^n$ then the maximum period is quoted as $2^{n-2}$ [2] or $m/4$ [55] (but $a$ must be of the form $8k + 3$ or $8k + 5$ and the initial seed should be odd).
However, when \( c \) is zero and \( m \) is prime, randomness in the low order bits is not a problem. In this case the period is maximal at \( m - 1 \) provided \( a \) is a “primitive element modulo \( m \)” (ie. \( a^n \mod m \) is not equal to 1). The most popular implementation of this form is \( m = 2^{31} - 1 \) and \( a = 16807 \). This implementation, although recommended by Park and Miller [65] as a “minimal standard” to be used when judging other generators, is not without flaws. Park and Miller [65] admit that, whilst it passes all statistical tests, it does not necessarily score well and it demonstrates a lattice structure by the Marsaglia (plotting of pairs) graphical test [55].

Alternative values of \( a \) recommended by L’Ecuyer [55] and Park and Miller [65] are 40692, 48271 and 69621 although Park and Miller recommend that correct implementation is an important consideration. Furthermore they provide examples of PASCAL code which allows a correct implementation by preventing integer overflow from the \( a \cdot X_n \) operation.

**Shift Register Generators**

These generators have sequences defined by

\[
\begin{align*}
temp & = X_n \oplus (X_n \text{ SHR}i) \\
X_{n+1} & = temp \oplus (temp \text{ SHL}j)
\end{align*}
\]

where \( \oplus \) is the exclusive OR operator, SHR\( i \) means shift right \( i \) bits and SHL\( j \) means shift left \( j \) bits.

These generators have the disadvantage that they have sometimes demonstrated poor statistical results [2]. Provided the parameters are well chosen, they have been shown to have a better lattice structure, a longer period and are faster than some LCG’s [63, 55]. However, they tend to require a large amount of computer memory [55].

**Lagged–Fibonacci Generators**

These are of the form

\[
X_n = X_{n-r} \text{ op } X_{n-s}
\]

where \( n \geq r \) and \( 0 < s < r \). The operator \( \text{ op } \) is either addition mod \( m \), subtraction mod \( m \), multiplication mod \( m \) or the exclusive OR (if \( m \) is a power of 2).
Fibonacci generators where \( r = 1 \) and \( s = 2 \) have demonstrated serial correlation and the permutations
\[
X_{i-1} < X_{i+1} < X_i \quad \text{and} \quad X_i < X_{i+1} < X_{i-1}
\]
ever occur despite there being a probability of 1/6 that three numbers could be so ordered, see [8].

Very long periods are possible with these generators [43, 2] but the initial sequence used to initiate the generator is extremely critical to its performance [2].

**Improvements**

Although Kloeden et al [43] and Ripley [69] recommend persevering with simpler, well understood, algorithms with some theoretical background, there is evidence that improvements can be made with the following processes [55, 93, 2, 77]:

1. Shuffling

Shuffling can involve the use of one or two pseudo-random number generators. When only one is used a table of \( k \) \((k > 20)\) random numbers is set up. Another random number \( X \) is generated to determine the index position \((\text{int}(k \cdot X)+1)\) of one number which is then output. The next random number is generated to replace this chosen number in the table. This is repeated until the required number of pseudo-random numbers has been produced. For example consider two pseudo-random number generators, say \( A \) and \( B \). Assuming that the \( k \) numbers are generated using \( A \) then the number to be output is chosen using \( B \) and then replaced by another number generated using \( A \) [43, 2].

There is little theoretical support for this method [70], but it has been shown to improve poor generators [43, 76] by increasing the period and removing “local non-randomness” [76]. However, it can also result in a poorer performing generator than the original ones [2].

2. Combined or Composite Generators

These are defined by the general sequence:
\[
Z_n = (X_n \pm Y_n) \mod m
\]  
(3.5)

where \( X_n \) and \( Y_n \) are produced by separate pseudo-random number generators.
They have also been used with more than two generators [55, 93] and in a form where the next number is chosen randomly from those produced by a number of generators [2].

Combined generators have been shown to be very good if the underlying generators are good but the only proven theoretical advantage is a longer period. The combined $k$-sequence generator

$$Z_n = \left(X_n^1 \pm X_n^2 \pm X_n^3 \pm \ldots \pm X_n^k\right) \mod m$$

can have period

$$\prod_{j=1}^{k} (m_j - 1)$$

where $m_j$ is the period of the $X_n^j$th sequence [55, 2]. They can also be somewhat slower than the shuffling modified generators [76]. Knuth [49] considers these combined generators as “dangerous”.

L’Ecuyer [55] proposed two combined generators which have passed statistical tests. These are recommended as “correct” implementations of the underlying LCG’s by first respecifying each LCG as follows:

$$k = x \text{ div } q$$

$$s = a(x - k \cdot q) - k \cdot r$$

if $s < 0$ then $s = s + m$.

Here $q = m \text{ div } a$, $r = m \text{ mod } a$, $x$ is the number previously generated by this LCG, and $a$ and $m$ are the parameters of the LCG. The output from each generator $(s_1, s_2)$ is then combined.

- **Combined generator recommended for 32 bit computers:**

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$m_i$</th>
<th>$q$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCG 1</td>
<td>40014</td>
<td>2147483563</td>
<td>53668</td>
<td>12211</td>
</tr>
<tr>
<td>LCG 2</td>
<td>40692</td>
<td>2147483399</td>
<td>52774</td>
<td>3791</td>
</tr>
</tbody>
</table>

Combine as $z = s_1 - s_2$.

If $z < 1$, let $z = z + 2147483562$. 
Finally \( U = z/2147483562 \).

This generator has a period of \((m_1 - 1)(m_2 - 1)/2\), ie. approximately \(2.30584 \times 10^{18}\).

- **Combined generator recommended for 16 bit computers:**

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>( m_i )</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCG 1</td>
<td>157</td>
<td>32363</td>
<td>206</td>
<td>21</td>
</tr>
<tr>
<td>LCG 2</td>
<td>146</td>
<td>31727</td>
<td>217</td>
<td>45</td>
</tr>
<tr>
<td>LCG 3</td>
<td>142</td>
<td>31657</td>
<td>222</td>
<td>133</td>
</tr>
</tbody>
</table>

Combine as \( z = s_1 - s_2 \).

If \( z > 706 \), let \( z = z - 32362 \).

Then \( z = z + s_3 \)

If \( z < 1 \), let \( z = z + 32362 \).

Finally \( U = z/m_1 \).

This generator has a period of \((m_1 - 1)(m_2 - 1)(m_3 - 1)/4\), ie. approximately \(8.12544 \times 10^{12}\).

Another combined algorithm was proposed by Wichman and Hill [93]. It uses three LCG’s. Their algorithm was:

\[
s = a(x \mod q) - l(x \div q)
\]

if \( s < 0 \) then \( s = s + m \)

where \( q = m \div a \) and \( l = m - (a \ast q) = m \mod a \). This allows for “correct” implementation on a 16 bit computer.

This generator combines the three LCG’s output \((s_1, s_2, s_3)\) as follows

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>( m_i )</th>
<th>q</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCG 1</td>
<td>171</td>
<td>30269</td>
<td>177</td>
<td>2</td>
</tr>
<tr>
<td>LCG 2</td>
<td>172</td>
<td>30307</td>
<td>176</td>
<td>35</td>
</tr>
<tr>
<td>LCG 3</td>
<td>170</td>
<td>30323</td>
<td>178</td>
<td>63</td>
</tr>
</tbody>
</table>

Use \( z = s_1/m_1 + s_2/m_2 + s_3/m_3 \) where \( U = z - \text{trunc}(z) \).
This generator has been shown to be equivalent to an LCG with very large parameters of \( a = 16555425264690 \) and \( m = 27817185604309 \) [99]. In this case the period is \((m_1 - 1)(m_2 - 1)(m_3 - 1)/4\), i.e. approximately \(6.9536 \times 10^{12}\).

A variation by Sherif and Dear [77] combines two random number generators with the purpose of improving the period and the statistical properties of the first generator. It combines the first two digits \((D_1 \ & \ D_2)\) of the number produced by the first pseudo-random number generator (the example used was an LCG) with two digits \((C_1 \ & \ C_4)\) from the cube of a 4 digit seed. The authors show that the first and fourth digits from the right of the cube of a 4-digit number are independent and identically distributed. The digits are combined in the order: \(C_1D_1C_4D_2\).

A problem that arises is that the cube of a four digit number is too large for integer arithmetic on some computer systems and its calculation may be inaccurate with real arithmetic. This can be overcome, however, by the following (PASCAL) code:

\[
cubeno := (\text{num}**2 \mod 10000) * \text{num}.
\]

### 3.1.4 Conversion to Normal Distribution

Theoretically, producing a normally distributed random variable can be accomplished by inverting the probability distribution function. However, the integrals for the normal pdf must be evaluated numerically, so an exact computation is not possible using this method [43, 69, 1]. Other approaches have been developed to avoid the time consuming process of numerical integration, for example the following methods can be implemented:

**The Box–Muller Method**

The Box–Muller method produces a pair of normally distributed random variables \((G_1, G_2)\) from a pair of uniformly distributed random variables \((U_1, U_2)\) using:

\[
G_1 = \sqrt{-2 \ln(U_1)} \cos(2\pi U_2) \\
G_2 = \sqrt{-2 \ln(U_1)} \sin(2\pi U_2).
\] (3.6)

It has been suggested that this method has the disadvantage of poor speed due to using the computationally intensive sine and cosine functions. However, depending on the programming language and compiler used, the system intrinsic functions \(\sin, \cos, \log\)
and \texttt{sqrt} may be highly optimised and so computational speed may not be compromised. Further, it has also been reported that this method is not particularly good when used with an LCG [43, 69, 1].

**The Polar–Marsaglia Method**

The Polar-Marsaglia method is a variation of the Box–Muller Method which is often recommended. It is useful for applications requiring many numbers, even though, as part of its algorithm, the proportion \(1 - \pi/4\) (or approximately 21.4\%) of generated numbers are discarded [43, 1].

This method is easy to program and has “essentially perfect accuracy”[49]. The method involves converting \(U_1\) and \(U_2\) to be \(U(-1,1)\) by the transformation

\[ V_i = 2U_i - 1 \quad \text{for} \quad i = \{1, 2\}, \]

then calculating

\[ W = V_1^2 + V_2^2 \]

and rejecting any points outside the unit circle (reject if \(W > 1\)). Then replacing the sine and cosine functions by

\[
\cos \theta = \frac{V_1}{\sqrt{W}}, \quad \sin \theta = \frac{V_2}{\sqrt{W}}
\]

leads to the normally distributed random variables [43, 1]:

\[
G_1 = V_1 \sqrt{-2 \ln(W)/W} \quad G_2 = V_2 \sqrt{-2 \ln(W)/W}. \tag{3.7}
\]

Note that when \(W\) is small, the reciprocal \(1/W\) can lead to problems; thus many researchers avoid the Polar-Marsaglia method.

**Ratio of Uniforms**

Another method, called the “ratio of uniforms” accepts any point \(P(u,v)\), which is contained in a certain region of the plane. Here \(u\) and \(v\) are \(U(0,1)\) distributed. The region is determined so that the the ratio of the coordinates has the desired distribution in that region. For the normal distribution the \(U(0,1)\) numbers \(u\) and \(v\) are converted to be \(U(-\sqrt{2/e}, \sqrt{2/e})\) distributed and the “accept” region is \(\mathcal{A} = \{(u, v) \mid v^2 < -4u^2 \ln u\}\). Points outside the region are rejected [57, 69, 39], see Figure 3.1.
Figure 3.1: Accept and Reject regions for the Ratio of Uniforms approach.

The method can be quite slow because of the logarithm calculations required [57, 69]. For this reason simple inner and outer bounding curves have been proposed which allow acceptance or rejection before checking for the exact region.

For the bounding curves Leva [57] uses the quadratic form

\[ Q = x^2 + y(ay - bx) \]

where \( x = u - s, \ y = |v| - t, \ a = 0.19600 \) and \( b = 0.25472 \). The point \((s, t)\), which is the centre of the quadratic form, is \((0.449871, -0.386595)\). The quadratic sits around the boundary of \( \mathcal{A} \) and Leva shows this method to be both fast and easily coded. It accepts points where \( Q(u, v) < 0.27597 \) as these are definitely inside the region \( \mathcal{A} \). It rejects points where \( Q(u, v) > 0.27846 \) as these are definitely outside the region \( \mathcal{A} \). Then, it checks the rest to see if they are exactly in \( \mathcal{A} \), accepting those that are. The ratio \( v/u \) of accepted numbers is returned as the pseudo-random number.
Composition Algorithms

These are designed for speed and only approximate the normal distribution. One in particular, the Marsaglia–Bray which is recommended by Ripley [69], approximates the normal pdf on (−3,3) using the sums of 2 or 3 uniform pseudo-random numbers. The process is more involved but supposedly faster than those mentioned previously. Ripley suggests that if satisfactory speed is obtained using the Polar–Marsaglia or Ratio of Uniforms method, then the effort involved in the Marsaglia–Bray could be wasted.

3.1.5 Testing Random Number Generators

The objective here is to test the pseudo-random number generators to ensure that the numbers produced could be a sample from a population of $U(0, 1)$ numbers (with a certain degree of confidence). The samples can be tested for randomness in a variety of ways, although Wichmann and Hill [93] and Bratley et al [8] make an important observation – they suggest that it could be a concern if these methods always pass the tests, since if truly random, they would be expected to fail occasionally. Obvious deficiencies in generators can be picked up by subjective tests. One such test compares frequency histograms of the pseudo-random sequences with a histogram of the required pdf. Another popular method of quickly evaluating these deficiencies is to produce a scatterplot of pairs from the generated sequence. This is referred to as the Marsaglia diagram. Usually the pseudo-random sequences are subjected to a variety of tests since “bad” generators have been known to pass some tests (e.g. Von Neumann’s middle square generator passed the frequency, gap and poker tests but failed the serial test [49]).

The following methods are used to test for the uniformity and independence of the numbers generated in order to determine whether or not the pseudo-random sequences behave enough like “truly” random numbers [74].

Tests for Uniformity

1. Chi-squared Goodness of Fit Test

The Chi-squared ($\chi^2$) test is used to determine how well the distribution of the pseudo-random number sequence matches the uniform distribution. The set of random numbers is sorted and the data is then divided into $k$ mutually exclusive
groups. The number of values from the pseudo-random sequence which fall into each of these \( k \) categories is counted. To compare these counts to the expected number in each category, the statistic

\[
Y = \sum_{j=1}^{k} \frac{(N_j - Np_j)^2}{Np_j}
\]

is calculated. Here \( N_j \) is the number of values from the pseudo-random sequence counted in category \( j \) and \( Np_j \) is the expected number in category \( j \).

For the uniform distribution \( Np_j = N/k \) (where \( N \) is the total number of values being tested) so the statistic \( Y \) simplifies to

\[
Y = \sum_{j=1}^{k} \frac{(N_j - N/k)^2}{N/k} = \sum_{j=1}^{k} \frac{k}{N} \left( N_j - \frac{N}{k} \right)^2.
\]

For large samples the statistic \( Y \) is known to be approximately \( \chi^2 \) with \( k - 1 \) degrees of freedom. Note that \( \Pr(Y > \chi^2_{1-\alpha, k-1}) = \alpha \) is used to measure the strength of the relationship where \( 100\alpha \) is the referred to as the significance level of the test. Bratley et al [8] suggest that the number of categories \( k \) can be determined by an approximation to the Mann-Wald \( N^* \) rule. According to Knuth and Rubinstein [49, 74], \( N \) should be greater than \( 5k \) where \( k \) is a power of 2 and greater than 1000.

Tables of critical \( \chi^2 \) value are readily available and critical values at the \( \alpha = 5\% \) level, \( z_{\alpha} = 1.645 \), are given in Table 3.1.

<table>
<thead>
<tr>
<th>( \nu )</th>
<th>( \chi^2_{0.05, \nu} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>123.225</td>
</tr>
<tr>
<td>999</td>
<td>1073.649</td>
</tr>
<tr>
<td>9999</td>
<td>10232.758</td>
</tr>
<tr>
<td>99999</td>
<td>100735.798</td>
</tr>
<tr>
<td>999999</td>
<td>1002326.518</td>
</tr>
</tbody>
</table>

Table 3.1: Critical values of \( \chi^2_{0.05, \nu} \)

An asymptotic form for the \( \chi^2 \) value for large number of degrees of freedom, \( \nu \), is

\[
\chi^2_{\alpha, \nu} = \nu \left( 1 - \frac{2}{9\nu} + z_{\alpha} \sqrt{\frac{2}{9\nu}} \right)^3
\]
where $z_\alpha$ is the Standard Normal distribution at significance level $\alpha$ [18].

The $\chi^2$ test is recommended as useful for comparing generators but is not recommended as a means of accepting the hypothesis that a pseudo-random sequence is uniform as it rejects only a small proportion of the cases where the numbers are not uniformly distributed [8].

2. Kolmogorov–Smirnov Goodness of Fit Test

The $\chi^2$ test determines how well the pseudo-random sequence fits the desired distribution. It does not, however, rely on $N$ being large. Whilst the $\chi^2$ test can be applied to continuous and discrete distributions, the Kolmogorov–Smirnov (KS) test works only for continuous distributions [49]. This test is said to make more use of all the available data since it does not rely on “arbitrarily” chosen groups [74].

The KS test involves sorting the numbers in increasing order so as to calculate the two statistics:

$$K_N^+ = \sqrt{N} \max_j \left\{ \frac{j}{N} - F(X_j) \right\}$$
$$K_N^- = \sqrt{N} \max_j \left\{ F(X_j) - \frac{j-1}{N} \right\}$$

for $j = 1, 2, 3, \ldots, N$. Here $F(X_j)$ is the value of the required cumulative density function at $X_j$.

These are the maximum and minimum positive and negative distances between the generated numbers and their supposed distribution. They should be distributed according to tabled values [43, 49, 8, 74]. The hypothesis that $X \sim U(0,1)$ is rejected if the statistics are too large or too small compared to the tabled values.

The cumulative distribution function for the Kolmogorov–Smirnov test is

$$H(x) = 1 - 2 \sum_{j=1}^{\infty} (-1)^{j-1} e^{-2j^2x^2}$$

and critical values at the 95% and 99% level are 1.358 and 1.628 respectively [74].

**Tests for independence**

1. The Serial Test

The Serial test is used to determine to what extent the pseudo-random sequence generated has “pairs of successive numbers uniformly distributed in an independent
manner” [49]. In two dimensions the test involves dividing the plane into \( r^2 \) sections, each section with area \( 1/r^2 \). The number of pairs, \( V \), falling into each section is then counted. A \( \chi^2 \) test on the counts is undertaken using the test statistic

\[
Y = \frac{r^2}{N} \sum \left( V - \frac{N}{r^2} \right)^2.
\]

The test statistic is compared to a \( \chi^2 \) distribution with \( r^2 - 1 \) degrees of freedom [74]. This test has large computer storage requirements and is very CPU time intensive in dimensions greater than two, hence the test is rarely used [49, 74]. Recent literature, for example [55] quotes Serial test results conducted in three and four dimensions.

2. The Runs Up and Down Test

This test records the direction of change between each pair of numbers. The number of runs up or the number of runs down in a sequence of \( N \) numbers should be asymptotically normally distributed with mean and variance \((2N - 1)/3\) and \((16N - 29)/90\) respectively [8]. The observed and expected number of runs can be statistically compared using a \( \chi^2 \) test.

The length of the runs can also be tested. The analysis determines whether too large a number of “long” runs exists. The expected number of occurrences of runs of length \( k \) is given by

\[
\frac{2(k^2 + 3k + 1)N - (k^3 + 3k^2 - k - 4)}{(k + 3)!}
\]

when \( k < N - 1 \) [74].

3. The Poker Test

The Poker test counts the occurrence of patterns in groups of \( k \) (usually 5) integers. A \( \chi^2 \) test is then performed using the probability

\[
p = \frac{d(d - 1) \cdots (d - r + 1)}{d^k} \binom{k}{r}
\]

where \( d \) is the number of different integers, \( r \) is the number of distinct elements in the set, and

\[
\binom{k}{r}
\]
are Stirling numbers (the number of ways of partitioning a set of \( k \) elements into exactly \( r \) parts [49].)

4. Coupon Collectors Test

This test considers the lengths of sequences needed to “collect” all the integers \( 0, \ldots, K - 1 \). The probability of a length \( r \) being needed is found and compared to the observed lengths using a \( \chi^2 \) test [69, 49].

5. Gap Test

The Gap test considers the lengths of “gaps” between numbers belonging to the same range \( 0 \leq \alpha < \beta \leq 1 \), where the gaps are a count of the numbers in between but not belonging to the range [49].

### 3.1.6 Experimental Results

I carried out a large number of numerical experiments in order to determine which generator is best suited for use in the SDE simulations. Furthermore, a comparison of the Uniform to Standard Normal converting algorithms was also undertaken. Emphasis on the computational efficiency of the algorithms was explored. All experiments were performed on Sun Sparc 10 Workstations running the Solaris OS.

Overall seven standard Uniform distribution generators were compared. They were:

1. the intrinsic PASCAL/C\(^1\)

2. the intrinsic FORTRAN

3. the intrinsic FORTRAN (VAX FORTRAN emulation)

4. the Wichmann–Hill [93]

5. the “minimal standard” of Park and Miller [65]

6. the L’Ecuyer [55]

7. the Sherif–Dear [77]

\(^1\)Note that the PASCAL and C intrinsic generators produce identical sequences and hence are considered as one generator.
Each generator was run five times using different starting seeds. A total of 100,000 numbers were generated per run. Results from three tests were used to compare these methods – these tests were:

1. a $\chi^2$ test over the interval $[0, 1]$ which was split into 100 equally spaced subintervals
2. a Kolmogorov–Smirnov test
3. a Serial test where the $[0, 1] \times [0, 1]$ area was divided into $100 \times 100$ cells.

Each generator’s CPU requirements (in seconds) were also recorded for comparison purposes. Details are presented in Table 3.2.
<table>
<thead>
<tr>
<th>Uniform Generator</th>
<th>Seeds</th>
<th>Time (sec)</th>
<th>$A^2$</th>
<th>K-S</th>
<th>Serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrinsic PASCAL/C</td>
<td>170796</td>
<td>3.1</td>
<td>108.203</td>
<td>0.00331</td>
<td>9461.0</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>3.1</td>
<td>125.833</td>
<td>0.00347</td>
<td>10039.8</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>3.2</td>
<td>88.345</td>
<td>0.00151</td>
<td>10207.4</td>
</tr>
<tr>
<td></td>
<td>121212</td>
<td>3.1</td>
<td>96.240</td>
<td>0.00297</td>
<td>10122.8</td>
</tr>
<tr>
<td></td>
<td>2134</td>
<td>3.0</td>
<td>98.428</td>
<td>0.00284</td>
<td>9851.6</td>
</tr>
<tr>
<td>Intrinsic FORTRAN(UNIX)</td>
<td>170796</td>
<td>61.6</td>
<td>482.034</td>
<td>0.01341</td>
<td>10262.8</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>61.6</td>
<td>442.234</td>
<td>0.00438</td>
<td>10205.2</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>61.5</td>
<td>409.252</td>
<td>0.00619</td>
<td>10349.2</td>
</tr>
<tr>
<td></td>
<td>121212</td>
<td>61.6</td>
<td>509.008</td>
<td>0.00819</td>
<td>10736.8</td>
</tr>
<tr>
<td></td>
<td>2134</td>
<td>61.5</td>
<td>496.108</td>
<td>0.00929</td>
<td>10028.0</td>
</tr>
<tr>
<td>Intrinsic FORTRAN(VAX)</td>
<td>170796</td>
<td>3.3</td>
<td>112.724</td>
<td>0.00479</td>
<td>9921.6</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>3.3</td>
<td>99.702</td>
<td>0.00191</td>
<td>9872.4</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>3.3</td>
<td>107.696</td>
<td>0.00298</td>
<td>10000.0</td>
</tr>
<tr>
<td></td>
<td>121212</td>
<td>3.3</td>
<td>98.124</td>
<td>0.00272</td>
<td>9988.8</td>
</tr>
<tr>
<td></td>
<td>2134</td>
<td>3.3</td>
<td>112.908</td>
<td>0.00275</td>
<td>10081.6</td>
</tr>
<tr>
<td>Wichman-Hill</td>
<td>234</td>
<td>3.7</td>
<td>94.388</td>
<td>0.00222</td>
<td>9938.8</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>3.7</td>
<td>94.388</td>
<td>0.00222</td>
<td>9938.8</td>
</tr>
<tr>
<td></td>
<td>611</td>
<td>3.7</td>
<td>94.388</td>
<td>0.00222</td>
<td>9938.8</td>
</tr>
<tr>
<td></td>
<td>17786520</td>
<td>3.5</td>
<td>116.124</td>
<td>0.74325</td>
<td>9828.8</td>
</tr>
<tr>
<td></td>
<td>909999673</td>
<td>3.5</td>
<td>116.124</td>
<td>0.74325</td>
<td>9828.8</td>
</tr>
<tr>
<td></td>
<td>217652122</td>
<td>3.5</td>
<td>116.124</td>
<td>0.74325</td>
<td>9828.8</td>
</tr>
<tr>
<td></td>
<td>74</td>
<td>3.5</td>
<td>86.160</td>
<td>0.00173</td>
<td>10117.6</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>3.5</td>
<td>86.160</td>
<td>0.00173</td>
<td>10117.6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.5</td>
<td>86.160</td>
<td>0.00173</td>
<td>10117.6</td>
</tr>
<tr>
<td></td>
<td>534</td>
<td>3.5</td>
<td>114.726</td>
<td>0.99634</td>
<td>10038.8</td>
</tr>
<tr>
<td></td>
<td>987501322</td>
<td>3.5</td>
<td>114.726</td>
<td>0.99634</td>
<td>10038.8</td>
</tr>
<tr>
<td></td>
<td>2349970</td>
<td>3.5</td>
<td>114.726</td>
<td>0.99634</td>
<td>10038.8</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>3.5</td>
<td>96.698</td>
<td>0.00311</td>
<td>10303.2</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3.5</td>
<td>96.698</td>
<td>0.00311</td>
<td>10303.2</td>
</tr>
<tr>
<td></td>
<td>3000</td>
<td>3.5</td>
<td>96.698</td>
<td>0.00311</td>
<td>10303.2</td>
</tr>
<tr>
<td>Minimal Standard</td>
<td>170796</td>
<td>3.1</td>
<td>88.794</td>
<td>0.00293</td>
<td>10156.8</td>
</tr>
<tr>
<td></td>
<td>53</td>
<td>3.3</td>
<td>115.016</td>
<td>0.00410</td>
<td>10120.4</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>3.1</td>
<td>87.094</td>
<td>0.00218</td>
<td>10050.8</td>
</tr>
<tr>
<td></td>
<td>121212</td>
<td>3.2</td>
<td>86.518</td>
<td>0.00267</td>
<td>10012.8</td>
</tr>
<tr>
<td></td>
<td>2134</td>
<td>3.1</td>
<td>96.548</td>
<td>0.00258</td>
<td>10015.6</td>
</tr>
<tr>
<td>L'Ecuyer</td>
<td>234</td>
<td>3.5</td>
<td>72.924</td>
<td>0.00222</td>
<td>10148.8</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>3.5</td>
<td>72.924</td>
<td>0.00222</td>
<td>10148.8</td>
</tr>
<tr>
<td></td>
<td>17786520</td>
<td>3.5</td>
<td>113.542</td>
<td>0.00214</td>
<td>10014.0</td>
</tr>
<tr>
<td></td>
<td>909999673</td>
<td>3.5</td>
<td>113.542</td>
<td>0.00214</td>
<td>10014.0</td>
</tr>
<tr>
<td></td>
<td>74</td>
<td>3.3</td>
<td>117.938</td>
<td>0.00323</td>
<td>10031.2</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>3.3</td>
<td>117.938</td>
<td>0.00323</td>
<td>10031.2</td>
</tr>
<tr>
<td></td>
<td>534</td>
<td>3.8</td>
<td>103.576</td>
<td>0.00288</td>
<td>9850</td>
</tr>
<tr>
<td></td>
<td>987501322</td>
<td>3.8</td>
<td>103.576</td>
<td>0.00288</td>
<td>9850</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>3.3</td>
<td>122.957</td>
<td>0.00351</td>
<td>9935.8</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3.3</td>
<td>122.957</td>
<td>0.00351</td>
<td>9935.8</td>
</tr>
<tr>
<td>Sherif-Dear</td>
<td>1212</td>
<td>4.2</td>
<td>94.532</td>
<td>0.00291</td>
<td>9828.4</td>
</tr>
<tr>
<td></td>
<td>170796</td>
<td>4.2</td>
<td>94.532</td>
<td>0.00291</td>
<td>9828.4</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>4.6</td>
<td>90.636</td>
<td>0.00201</td>
<td>9866.0</td>
</tr>
<tr>
<td></td>
<td>1707</td>
<td>5.3</td>
<td>117.190</td>
<td>0.00350</td>
<td>10023.6</td>
</tr>
<tr>
<td></td>
<td>999999</td>
<td>5.3</td>
<td>117.190</td>
<td>0.00350</td>
<td>10023.6</td>
</tr>
<tr>
<td></td>
<td>5730</td>
<td>4.4</td>
<td>120.858</td>
<td>0.00183</td>
<td>10035.2</td>
</tr>
<tr>
<td></td>
<td>121212</td>
<td>4.4</td>
<td>120.858</td>
<td>0.00183</td>
<td>10035.2</td>
</tr>
<tr>
<td></td>
<td>4235</td>
<td>4.3</td>
<td>102.164</td>
<td>0.00348</td>
<td>9772.8</td>
</tr>
</tbody>
</table>

Table 3.2: Comparison of $U(0, 1)$ generators
With regard to CPU requirements, the Intrinsic FORTRAN generator performed woe-
fully at about 20 times slower than the other generators. As expected, the combined
generators (Wichmann-Hill, L'Ecuyer and Sherif-Dear) were slower than the single gen-
erators. The difference in speed was only about 10%, however in a large SDE simulation
this difference could prove prohibitive. The Sherif-Dear generator was the slowest of the
combined generators.

All generators passed the three statistical tests with the exception of one run of the
Intrinsic PASCAL/C generator, which just failed the $\chi^2$ test. The only other interesting
aspect was that the second and fourth runs of the Wichmann-Hill generator led to sub-
stantially larger than usual values of the Kolmogorov-Smirnov statistic. This could be
explained by the fact that the Wichmann-Hill generator requires seed values less than
30,000.

As a further check a Marsaglia comparison was produced for each generator. In this
test, the generated pseudo random numbers $(U_{2n-1}, U_{2n})$, $n = 1, 2, \ldots$, were plotted on
$[0, 1] \times [0, 1]$. Any patterns or band effects evident are an indication of possible non
uniformity. Figure 3.2 is an example of a poor generator with banding. Figures 3.3 to 3.9
are the Marsaglia plot test results for the seven generators.

![Marsaglia diagram for poor LCG](image)

**Figure 3.2:** Marsaglia diagram for poor LCG
Figure 3.3: Marsaglia diagram for intrinsic PASCAL/C generator

Figure 3.4: Marsaglia diagram for intrinsic FORTRAN generator
Figure 3.5: Marsaglia diagram for intrinsic FORTRAN (VAX) generator

Figure 3.6: Marsaglia diagram for Wichmann–Hill generator
Figure 3.7: Marsaglia diagram for Park–Miller generator

Figure 3.8: Marsaglia diagram for L’Ecuyer generator
Figure 3.9: Marsaglia diagram for Sherif-Dear generator

Only the Sherif-Dear generator showed any evidence of banding and the banding was restricted to the lowest 10%, of the unit square, i.e. on \([0,1] \times [0,0.1]\). All generators produced some “floral” patterns but none were regular enough to suggest non-uniformity. This suggests that they are all as good (or bad) as each other in this respect.

Three uniform to normal distribution conversion algorithms were tested; the Box-Muller, the Polar-Marsaglia and Leva’s Ratio of Uniforms methods. The CPU requirements for these were also determined and are provided in Table 3.3.

The average counts per second for the conversion algorithms were 14,974 for Box-Muller, 14,066 for Polar-Marsaglia and 8,357 for the Ratio of Uniforms method. This does not support Leva’s results of 477, 333 and 166 microseconds per normal deviate even taking into account the fact that Leva’s figures include the generation of the uniformly distributed numbers. Leva’s results show that the Box-Muller method produces 2,096 normal deviates per second, the Polar-Marsaglia 3,003 per second and the ratio of Uniforms 6,024 per second. This is the reverse of our results which indicate that the Box-Muller is fastest and the Ratio of uniforms is slowest.
<table>
<thead>
<tr>
<th>Box-Muller</th>
<th>Polar-Marsaglia</th>
<th>Ratio of Uniforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Time</td>
<td>Count/Sec</td>
</tr>
<tr>
<td>100000</td>
<td>6.6</td>
<td>15152</td>
</tr>
<tr>
<td>100000</td>
<td>6.8</td>
<td>14706</td>
</tr>
<tr>
<td>100000</td>
<td>6.8</td>
<td>14706</td>
</tr>
<tr>
<td>100000</td>
<td>6.7</td>
<td>14925</td>
</tr>
<tr>
<td>100000</td>
<td>6.6</td>
<td>15152</td>
</tr>
<tr>
<td>100000</td>
<td>6.9</td>
<td>14493</td>
</tr>
<tr>
<td>100000</td>
<td>6.6</td>
<td>15152</td>
</tr>
<tr>
<td>100000</td>
<td>6.6</td>
<td>15152</td>
</tr>
<tr>
<td>100000</td>
<td>6.6</td>
<td>15152</td>
</tr>
<tr>
<td>100000</td>
<td>6.6</td>
<td>15152</td>
</tr>
</tbody>
</table>

| Average of 14974 | Average of 14067 | Average of 8357 |

Table 3.3: Comparison of $U(0, 1)$ to $N(0, 1)$ converters

The normally distributed sequences were produced for each of the conversion methods with each of the uniform pseudo-random sequences, i.e. $3 \times 7 \times 5 = 105$ sequences of normally distributed pseudo-random numbers were produced. They were analysed by calculating frequency distribution tables and producing graphs presented in the top half of Figures 3.10 to 3.12. The class intervals chosen were from $< -3.0$ to $\geq 3.0$ using class widths of 0.2.

The counts in each interval were averaged over the five sets of frequencies produced for each uniform–normal combination. These average frequency counts were used to produce histograms which were each plotted with the normal curve to highlight any non-normal behaviour. Furthermore, the percentage discrepancy between the averaged actual counts and expected counts in each interval were calculated for each uniform–normal combination and the results plotted as line graphs (see bottom half of Figures 3.10 to 3.12).

The results show that all sequences were normally distributed with some slight skewness in the FORTRAN(UNIX)–Polar Marsaglia combination. The error graphs highlight some high (percentage) discrepancies in the tail areas. The most variation was produced from the ratio of uniforms method.
Figure 3.10: Analysis of Box–Muller conversion.
Figure 3.11: Analysis of Polar-Marsaglia conversion.
Figure 3.12: Analysis of Ratio of Uniforms conversion.
3.1.7 Concluding Remarks

The uniform pseudo-random number generators tested were indistinguishable as far as
the statistical tests are concerned. Thus, the main factor influencing choice of generator
is the computational time in which case the simpler generators are preferred. The longer
periods of the combined generators should also be taken into account.

The normal conversion algorithms all performed well, but once again the time taken
to perform the conversion influences which method is selected for SDE simulations. The
fastest converter appeared to be the Box-Muller.

My tests indicated that the intrinsic PASCAL, C or the FORTRAN(VAX) generators
together with the Box Muller normal converter were the fastest methods of producing
sequences of normally distributed random numbers. These should be the preferred options
for SDE simulations or any simulation requiring many random numbers. The fact that
the combined uniform generators produced sequences with longer periods may influence
the decision as to which generator to use, provided a 10% increase in computation time
is not a concern.

As a final comment, in all analyses care must be exercised when dealing with pseudo-
random number generators. It is tempting with the constantly increasing computational
performance of computers to try to analyse larger and more complex tasks. However the
danger exists in assuming that the pseudo-random numbers being generated are indeed
"random enough and long enough". The speed of modern day computers can mean that
repetition of the stream of pseudo-random numbers can occur within a short space of
computing time. Bearing this in mind, researchers who feel that their generated results
may have been biased by a non "random enough or random long enough" pseudo-random
number generator are probably best advised to repeat their simulations on different com-
puter systems or at least try different pseudo-random number generators.
3.2 A Distributed Processing System

3.2.1 Introduction

The numerical solution of stochastic differential equations (SDEs) by stochastic time discretisation schemes requires computer programs to perform massive number of computations. These schemes require the SDE or system of SDEs to be solved a large number of times. For example the numerical solution of a two dimensional SDE using a time discretisation scheme typically requires $10^6-10^8$ simulations with the final result presented as a statistical summary of the result of each simulation. Details are presented in Chapter 5. The time discretisation schemes use an iterative process where the solution at each time step is calculated directly from the preceding iteration. Fully implicit schemes, which are required when dealing with stiff systems, require the solution of a system of nonlinear equations at each time step. Simulations using these implicit schemes require substantially more computations than the explicit schemes when using the same stepsize. Note though that often a larger time step can be used due to the better stability properties of implicit schemes. A typical simulation flowchart is presented in Figure 3.13.

On single processor machines, such as the relatively slow VAX 5000 that was originally available to me, many one dimensional system simulations required up to 30 hours of CPU time whilst initial investigative two dimensional system simulations required in the order of 100 hours CPU time. With required CPU time expected to increase exponentially as the system dimension increases, the turn-around time when investigating higher dimensional “real world” or industrial application systems would be unacceptably prohibitive.

The structure of the time discretisation schemes used in the numerical solution of SDEs leads to their applicability for running on multiprocessor parallel or distributed processor supercomputers. In this case simulations can run simultaneously on each processor as only the final answer from each simulation is required. This is best described in the flowchart in Figure 3.14. Standish [84] has used a similar approach in molecular dynamics simulations.

Upon superseding the VAX 5000s in early 1996, Deakin University installed a large number of Sun (UNIX OS) minicomputers and workstations which were accessible through the university network. Depending on the machine, they were 3–10 times quicker than the superseded VAXes where most of my original analyses were undertaken. Yet they were not powerful enough as stand alone machines to ensure reasonable simulation completion
time. The workstations were only used by students during daytime and typically only during the academic year. They were almost always idle overnight, weekends and during the university semester breaks.

This situation led me to investigate the possibility of using the large number of available minicomputers as processors of a distributed processing system. In theory, and depending on any other load on the processors, given \( N + 1 \) available processors (one controller, the “master”, and \( N \) simulation processors, the “slaves”) there should be almost an \( N \) fold decrease in simulation completion time.

Based on having 101 available processors there should be an observable 300–1,000 times decrease in completion time for any simulation as compared to the time required when using the single processor VAX.

One aspect that did not prove to be the obstacle that I expected was the choice of programming language. FORTRAN, (specifically FORTRAN77), because of its age and some shortcomings, is often derided by computer scientists. However FORTRAN, by its design, is still one of the most efficient languages for arithmetic computations and func-
tion evaluations. Furthermore, modern versions of FORTRAN such as FORTRAN90 have available enough constructs to allow structured and modular programming. Finally, the version of FORTRAN available on the Deakin University SUN computers included system subprograms and functions to implement a distributed FORTRAN77 based processing system.

3.2.2 The "Master" and the "Slaves"

Overview

The system was made up of two equally important processes; a "Master" and a "Slave". For a distributed processing system multiple Slave processes are designed to run simultaneously.

Typically, the Master process initiated the individual Slave processes on the available processors. Upon their completion the Master collated their results into its collective set of results.
Each Slave process undertook its share of the required calculations and made its results available to the Master program for collection and collation. Two main versions of the Master/Slaves distributed processing system were examined.

3.2.3 System Development

In all versions that were created the key to the distributed processing system was the intrinsic system function SYSTEM which has the FORTRAN format

\[ I=\text{SYSTEM('COMMAND-STRING')} \]

Here \( I \) is an integer variable which returns the completion status and \( \text{COMMAND-STRING} \) is a character variable that contains the UNIX OS commands to be executed.

UNIX allows a command to be initiated on any network connected computer via the rsh command. It has the format

\[ \text{rsh machine-name command} \]

Hence the FORTRAN statement

\[ I=\text{SYSTEM('rsh machine-name command')} \]

initiates “command” on the computer “machine-name” and returns the value \( I=0 \) if successful.

Version 1

In Version 1, the Master process initially created a datafile called “PROC\(\text{ESS.DAT}\)” and wrote the value “1” to it. It then initiated one Slave process on each available processor.

Once initiated, each Slave process accessed PROC\(\text{ESS.DAT}\) and read the number stored within it. The stored number was incremented by 1 and the file closed for access by the next Slave process. The number read from PROC\(\text{ESS.DAT}\) was the unique processor number allocated to each Slave process.

The Slave process then completed its task and wrote its data into a datafile whose name was dependent on the Slave’s process number.

Meanwhile, the Master process remained in a loop until a Slave results file was created and its existence was detected. The Master then extracted the data from the Slave’s
results file and collated this data with any existing data collected from the other Slave processes.

It was not until the Master process had collated all the results files that the Master process output the overall results and shut itself down.

Apart from its obvious lack of finesse, this scheme had some problems. The first problem was caused by the open nature of the UNIX operating system which allowed files to be shared by multiple processes. The problem occurred when two Slave processes accessed PROCESS.DAT at on or about the same time. In this case the two Slave processes had identical process number and created identically named result files (the former was then automatically overwritten). Apart from the loss of data, the other problem here was that the Master process, having initiated $n$ Slave processes, had received less than $n$ results files to collate and would sit waiting indefinitely in a loop. The Master process would then have to be shut down manually.

**Version 2**

As with Version 1, the Master process in Version 2 created a datafile, this time named “PIDFILE.DAT”. Into this was written two numbers. The first was the process identification number ($pid$) allocated to the Master process by the UNIX OS. This $pid$ remained allocated and unique to the Master process until it was shut down. The second number initially written was “0”.

In a similar manner to that described in Version 1, all Slave processes were initiated on available processors.

Each initiated Slave process accessed PIDFILE.DAT to retrieve the Master’s $pid$ and the second number. This second number was incremented by one and set as the Slave’s processor number before being written into PIDFILE.DAT for the next Slave process to use.

In this version, instead of the Master process being in a wait loop until result datafiles were created, each Slave process sent a signal to the Master process that it had completed its task. Signals were sent using the UNIX kill command which is usually used to shut down running processes. It has format

```
kill -signum pid
```
where `signum` is an integer descriptor of the type of shutdown and `pid` is the processes `pid`.

However, with the Solaris version of FORTRAN, there was provision to harness such signals and use them as message numbers. Note that not all signals could be harnessed, for example, `kill -9 pid` will always shut down the process. For the Solaris OS, numbers 9, 14 and 23 are not available from the set 1 to 33, thus a total of 30 Slaves is possible. The FORTRAN function

```fortran
signal( signum , subroutine-name , -1 )
```

was used by the Master process to harness a received kill signal. Specifically, if the kill number `signum` was received, then the process’ control was passed to the subroutine `subroutine-name`.

Continuing in this manner, each Slave process completed its allocated tasks and wrote out its results to a datafile whose name corresponded to its processor number. It then used the system call to send

```bash
rsh machine-name kill -signum pid
```

where `signum` was the kill number corresponding to the Slave’s processor number and `machine-name` was the name of the computer that was running the Master process.

Meanwhile, the Master process waited until all the expected signals were received. As each one was received the harnessing subroutine flagged that the signal had been received and then opened and read the Slave written data into the corresponding results file.

When all signals had been received and output result files collated, the overall results were output by the Master process.

This version still had the major problem inherent in Version 1 whereby multiple Slave processes could access `PIDFILE.DAT` to obtain the same processor number.

**Other Options**

The main problems with Version 1 and 2 were readily overcome by the Master process inserting a suitable delay between the start of each Slave process. Another option was to include suitable code to bypass the open nature of the operating system, i.e., to allow locking of files. Therefore, once a Slave process had opened `PROCESS.DAT` or `PIDFILE.DAT` then no other Slave could access it until the current Slave process released its control
(lock). Due to the inter-relationship between UNIX and the C programming language it is possible to control the access to data files by using specialised C code. For example a C code subroutine that can be linked with the FORTRAN code for locking and unlocking files is available in [85]. This was tested and was shown to work correctly, though it was not implemented in the final version as I wanted to complete the system entirely in FORTRAN.

Another way I overcame the problem of multiple Slave processes simultaneously accessing the datafile was to create a version that made use of aspects of both Versions 1 and 2. In this version the Master started the first Slave and then waited for a message (sent via the kill system described for Version 2) from the Slave on the end of its access to the pid datafile. The Master then started the next Slave. Once all the Slaves were running the Master inelegantly waited and checked for the creation of Slave result datafiles before collating.

Finally, another option that could be undertaken involves a double messaging system with an initial message from the Slave upon the end of the access of the datafile. Then a second message would be sent upon completion of its simulation task. This would require a substantially more complex Master process to harness two messages from each Slave. Furthermore, there may be a requirement for a message buffer for the storage of messages before action by the Master. This theoretical option was not investigated further.

### 3.2.4 Numerical Experiment

For comparison purposes an experiment involving the generation of normally distributed random variables was undertaken. These normally distributed random variables are required in the numerical solution of SDEs as outlined earlier. The generation was based on the Polar-Marsaglia method, see Section 3.1.4 or Marsaglia and Bray [59].

The experiments involved determining the CPU time required to generate $10^5$--$10^9$ \( N(0,1) \) distributed random variables using direct simulation and Versions 1 and 2 of the Master and Slaves systems. The analysis was based on the use of eleven SPARC workstations, one as Master and ten as Slaves.

Figure 3.15 provides a graphical comparison of the distributed processing system's performance relative to direct simulation. As expected, the direct simulation method has completion time directly proportional to the number of simulations. Both distributed
processing versions provided better performance for large number of simulations than via direct simulation. In particular, the inelegant Version 1 provided better performance than Version 2 with its completion messaging system. Finally, as expected and demonstrated from the results for Version 1, for the small number of simulations case the computational overheads of the distributed processing system led to poorer performance when compared to the direct realisations method.

![Graph](image)

**Figure 3.15: Comparison of distributed processing systems**

Figure 3.16 provides the Performance Ratio of Version 1. The performance ratio is a measure of how much quicker the distributed processing system is over the direct simulation system. The Performance Ratio is bounded above with a maximum value of ten (the number of available Slave processors in these experiments). The Performance ratio rapidly increases towards 7 as the number of realisations increases. It is expected that as the number of realisations increases then this Performance Ratio will increase asymptotically to some value less than the theoretical maximum of ten. There will always be some computational overhead so the theoretical maximum value will never be reached.
3.2.5 Concluding Remarks

The current distributed processing system versions were fairly rudimentary. They had minor problems such as not having any fallback for processors or processes that crashed; in this case the Master would just sit and wait until shut down by the operator. Such problems could be overcome with the use of more complex programming but that was beyond the scope of the task at hand. The objective was to create a simple distributed processing system that could substantially reduce the turn around time on computationally intensive tasks by making use of available processors.

This section has highlighted that it is possible to make use of multiple processor networks as a distributed processing system for repetitive CPU intensive tasks. Furthermore, as described in this section, this is possible using FORTRAN which is often considered an archaic programming language.²

²Note that FORTRAN is still extensively used by older mathematicians, engineers and scientists who require high precision calculations. Furthermore, FORTRAN is direct in its data access constructions so it is easier for the backend to optimally schedule the instructions of the compiled code.
3.3 Chapter Summary

This chapter has investigated two important topics in the study of the numerical solution of SDEs.

The first was an investigation of the accurate and efficient generation of random numbers. The use of suitable “random enough” pseudo-random numbers is absolutely crucial for the accurate numerical solution of SDEs.

Secondly, since many sample paths must be generated to obtain weak numerical solutions of SDEs an efficient processing system is required. However, most researchers have little or no access to supercomputers or distributed processing systems. In this section I showed how it was possible, even with FORTRAN, to create a distributed processing system using the readily available network computers at Deakin University.
Chapter 4

FPK Equation in One Dimension
4.1 Introduction

With the exception of a small number of cases, the statistical analysis of a nonlinear systems subjected to random excitation can cause difficulties for researchers. If the excitation can be expressed as white noise then the theory of Markov processes can be employed leading to an equation which can be used to describe the transition probability density function (pdf) of the response.

This equation is the time independent (or stationary) forward Kolmogorov equation (called the Fokker–Planck–Kolmogorov (FPK) equation in mathematical physics). Its one dimensional form is

$$\frac{d}{dx}\{a(x)p\} + \frac{1}{2} \frac{d^2}{dx^2}\{c(x)p\} = 0$$  \hspace{1cm} (4.1)

where \( p \equiv p(x) \) is the stationary pdf solution. It has analytic solution given by Langtangen [53] as

$$p(x) = \frac{C_0}{c(x)} \exp \left\{ 2 \int \frac{a(x)}{c(x)} \, dx \right\}$$

where \( C_0 \) is a normalising constant such that

$$\int_{-\infty}^{\infty} p(x) \, dx = 1.$$  

This distribution is called the Landau-Ginsberg distribution, see Klauder et al [41] for more details.

The corresponding one dimensional autonomous SDE to this FPK equation is

$$dX_t = a(X_t)dt + \sqrt{c(X_t)} \, dW_t$$  \hspace{1cm} (4.2)

where \( c(X_t) > 0 \). The choice of \( a(x) \) and \( c(x) > 0 \) leads to a variety of known probability density distributions. In this study the drift term was set to

$$a(x) = -\eta_1 x^3 + \eta_2 x$$

and the diffusion was set to

$$b(x) = \sqrt{c(x)} = \sqrt{2}$$

The choice of \( c(x) \) constant implied the case of additive noise which represented white noise with spectral density \( 1/\pi \). The drift and diffusion chosen led to a bimodal solution which was symmetric about \( x = 0 \) with local maxima at

$$\left( \pm \sqrt{\frac{\eta_2}{\eta_1}}, \frac{C_0}{\eta_1} \exp \left[ \frac{\eta_2^2}{4\eta_1} \right] \right)$$
and a local minimum at \((0, C_0/2)\). The analytic pdf solution was thus given by

\[
p(x) = \frac{C_0}{2} \exp \left\{ \frac{\eta_2 x^2}{2} - \frac{\eta_2 x^4}{4} \right\}.
\]

For \(\eta_1 = 40\) and varying the value of \(\eta_2\) through the range of values \(\{10, 20, 40, 50, 80\}\), as presented in Figure 4.1, created bimodal distributions with sharper and sharper peaks that steadily diverged. Thus the parameter \(\eta_2\) was considered as a stiffness parameter for the SDE.

![Figure 4.1](image)

Figure 4.1: Probability density for bimodal distribution with \(\eta_1 = 40\) and \(\eta_2 = 10, 20, 40, 50\) and 80.

In the specific range of cases investigated the normalising constant and summary statistics were determined and they are provided in Table 4.1. The value of the normalising constant was calculated (numerically) using MAPLE and varies as displayed in Figure 4.2.
<table>
<thead>
<tr>
<th>$\eta_2$</th>
<th>$C_0$</th>
<th>$E(X)$</th>
<th>$\text{Var}(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.90976</td>
<td>0.0</td>
<td>0.21615</td>
</tr>
<tr>
<td>20</td>
<td>0.18645</td>
<td>0.0</td>
<td>0.43569</td>
</tr>
<tr>
<td>40</td>
<td>$1.58593 \times 10^{-4}$</td>
<td>0.0</td>
<td>0.97252</td>
</tr>
<tr>
<td>50</td>
<td>$6.44848 \times 10^{-7}$</td>
<td>0.0</td>
<td>1.22888</td>
</tr>
<tr>
<td>80</td>
<td>$2.13354 \times 10^{-17}$</td>
<td>0.0</td>
<td>1.98725</td>
</tr>
</tbody>
</table>

Table 4.1: Normalising constant and summary statistics for the probability density solution for $\eta_1 = 40$ and various values of $\eta_2$.

Figure 4.2: Variation of the normalising constant $C_0$ against $\eta_2$ for $\eta_1 = 40$. 
4.2 Simulink Solution

4.2.1 Overview

In this section I will describe an approach to obtain a first rough solution to SDE equation (4.2). This will provide insight to support later more detailed analysis. This Simulink approach has the advantage of being very easy to implement as it requires very little computer coding.

4.2.2 Simulink 1D SDE model

Figure 4.3 presents the Simulink model for solving a one dimensional autonomous SDE. It generates one trajectory (solution) $X$ of an SDE with drift and diffusion functions as entered into the drift and diffusion blocks. These blocks are combined with sum and product blocks and with a White Noise block to build the SDE system. It has a manual switch to set the initial value for the simulation to either a user input constant or a random number. It has another manual switch, the Scope Switch block, to provide graphical presentation of the trajectory if required. The Zero-Order Hold block is designed to provide the final value of the trajectory on the display block.

![Simulink model diagram]

Figure 4.3: A Simulink model to solve 1 dimensional SDEs.
The Simulink approach can initially take a little time to set up to ensure that it works correctly. Furthermore, the Simulink approach is fairly slow and doesn’t make direct use of the underlying MATLAB capability with vector and matrix processing. However, it is suitable for quick ad-hoc analyses and once it is set up, it is a simple matter of double clicking blocks and modifying the SDE parameters to solve other SDEs.

4.2.3 Results

The solution of equation (4.2), and its associated FPK equation (4.1), is in the form of a probability distribution of the steady state of many trajectories. A simple MATLAB script was written to run the Simulink model $10^5$ times and store the final value of the trajectory in the MATLAB workspace for postprocessing and creating graphical summaries. The script is presented below:

```matlab
%% This m-file sets up the data needed to run the Simulink SDE model.

for i=1:100000
    sim('multiplicative')
    yan(i)=xFinal;
end
HIST(yan,50)
```

The Simulink results for the cases $\eta_2 = \{10, 40, 80\}$ are presented in Figures 4.4, 4.5 and 4.6. These results favourably match the superimposed analytic results even for the stiff case when $\eta_2 = 80$. 
Figure 4.4: Stationary pdf solution of equation 4.1 for $\eta_2 = 10$.

Figure 4.5: Stationary pdf solution of equation 4.1 for $\eta_2 = 40$. 
Figure 4.6: Stationary pdf solution of equation 4.1 for \( \eta_2 = 80 \).
4.2.4 Concluding Remarks

The Simulink approach has clearly provided insight into the solution of the one dimensional SDE. In this example the analytic solution was known, however in more complex SDE systems without analytic solution the insight gained could be applied to both the FEM and time discretisation approach. For example the Simulink solution has indicated that the domain of interest is $[-2, 2]$. The knowledge of the domain is crucial for the FEM approach and is of importance for the histogram binning process of the time discretisation approach. Furthermore, the Simulink solution indicated that the system becomes stiffer as $\eta_2$ increases, so the FEM approach would need to be modified with more detailed finite element gridding about regions of high probability mass. If using the stochastic time discretisation approach an implicit time discretisation scheme may be necessary to overcome the stiffness. Finally, the Simulink solution indicates a symmetric solution which could mean an FEM approach would only need to be applied to the half interval $[0, 2]$, thereby minimising CPU requirements.
4.3 Finite Element Method (FEM) Solution

4.3.1 Overview

Recall the one dimensional autonomous SDE
\[ dX_t = a(X_t) \, dt + \sqrt{c(X_t)} \, dW_t \tag{4.3} \]
where \( a(x) \) is the drift term, \( b(x) = \sqrt{c(x)} \) is the diffusion term and \( dW_t \) is the derivative of the Wiener process.

A solution was obtained in the last section using the Simulink modelling approach. The solution provided key insights into the domain of interest which will be used in the FEM approach. For the range of parameters of interest in this chapter the Simulink approach indicated that the infinite solution domain could be truncated to \([-2, 2]\).

4.3.2 FEM Solution of the FPK Equation

The BGFEM method as detailed in Section 2.1.2 was applied to equation (4.3) with drift and diffusion given by
\[ a(x) = -\eta_1 x^3 + \eta_2 x \quad \text{and} \quad b(x) = \sqrt{2} \]
respectively. Recall from Section 4.1 that this SDE has analytic solution
\[ p(x) = \frac{C_0}{2} \exp \left\{ \frac{\eta_2}{2} x^2 - \frac{\eta_1}{4} x^4 \right\} \]
where \( C_0 \) is a normalising constant.

The cases \( \eta_1 = 40 \) with \( \eta_2 = \{10, 20, 40, 50, 80\} \) were analysed using the BGFEM. For each case 11, 31, 51 and 101 nodes over the truncated computational domain \([-2, 2]\) were considered. The results are compared graphically in Figures 4.7 to 4.11. It is clear that as the value of \( \eta_2 \) increased, the SDE became stiffer and the FEM solution broke down to oscillatory and single peak solutions. This observation was true irrespective of the number of nodes considered.
Figure 4.7: FEM solution for $\eta_2 = 10$ using 11, 31, 51 and 101 nodes
Figure 4.8: FEM solution for $\eta_2 = 20$ using 11, 31, 51 and 101 nodes
Figure 4.9: FEM solution for $\eta_2 = 40$ using 11, 31, 51 and 101 nodes
Figure 4.10: FEM solution for $\eta_2 = 50$ using 11, 31, 51 and 101 nodes
Figure 4.11: FEM solution for $\eta_2 = 80$ using 11, 31, 51 and 101 nodes
4.3.3 Concluding Remarks

The Finite Element Method with simple linear trial functions has been used to solve the one dimensional FPK equation. Its drawback is the large number of nodes required to provide a suitably accurate solution. As indicated in Figure 4.12, the increase in computation time increases quadratically with the increase in the number of nodes. Furthermore it tends to provide solutions with oscillations, though of only small scale, at the extremes of the domain of interest. The major problem with the FEM schemes was that they collapsed when solving stiff SDEs.

Figure 4.12: CPU requirement as a function of number of nodes for the Gaussian Density Solution example.


4.4 Taylor Series Time Discretisation Approach

4.4.1 The solution process

In this section, five weak 2.0 order schemes were used to solve the equation (4.2) and their solutions were compared. The schemes investigated were the standard weak scheme, the implicit variant, the predictor-corrector, the derivative free implicit and a linearly implicit version of the standard weak scheme.

In all the computer experiments undertaken the following steps were performed (See Figure 4.13):

1. An initial (non random) value $X_0$ was chosen. In all the examples except those stated otherwise the initial value chosen was $X_0 = 0.5$.

2. The SDE was solved using each of the five chosen schemes from $t = 0$ to the designated end time $t = T = 5$ using 5000 time steps, ie. constant time step $\Delta = 0.001$. The resulting sample path value $X_T$ at $t = T$ was noted and a frequency distribution of the values of $X_T$ was updated. Based on the initial analysis using Simulink the frequency distribution used 35 equal subdivisions between $X = -1.75$ and $X = 1.75$. The end time of $T = 5$ was chosen based on initial trials to ensure

Figure 4.13: Steps in the numerical solution of SDEs.

\[ \text{Figure 4.13: Steps in the numerical solution of SDEs.} \]
that any transients had fully settled down. In theory the time scale to convergence was only $T \sim |2\eta_1|^{-1}$, ie. only $T \sim 1/20$.

3. The step above was repeated a set number of times, typically $10^4$ times except when the effect of the number of simulations was being investigated.

4. The resulting frequency distribution was qualitatively compared to the actual probability density as given by the analytic solution. A $\chi^2$ statistic and summary statistics mean and variance were determined for quantitative comparison.

4.4.2 Use of Maple in the solution process

Following the example in section 2.2.5, Maple was used to generate the weak 2.0 order scheme for SDE (4.2). The input into Maple was

```latex
> wktay2([-eta_1*x[1]^-3+eta_2*x[1]],[[sqrt(2)]]);
```

with resulting output

```latex
table[

1 = ( Yl_{n+1} = Yl_n + ( -eta_1 Yl_n^3 + eta_2 Yl_n ) \Delta_n \\
+ \frac{1}{2} ( -eta_1 Yl_n^3 + eta_2 Yl_n ) ( -3 eta_1 Yl_n^2 + eta_2 ) - 6 eta_1 Yl_n ) \Delta_n^2 \\
+ \sqrt{2} + \frac{1}{2} \Delta_n \sqrt{2} ( -3 eta_1 Yl_n^2 + eta_2 ) ) \Delta Ws_l_n
]
```

I also used an additional Maple routine to generate the weak 2.0 order derivative free implicit scheme. This routine is not currently available in the stochastic package. The input was

```latex
> impwk2([-eta_1*x[1]^-3+eta_2*x[1]],[[sqrt(2)]]);
```

and the output was

$Yl_{n+1} = Yl_n + \frac{1}{2} ( -eta_1 Yl_{n+1}^3 + eta_2 Yl_{n+1} - eta_1 Yl_n^3 + eta_2 Yl_n ) \Delta_n + \sqrt{2} \Delta W_1_n$

4.4.3 Weak 2.0 Order Numerical Schemes Utilised

In the following schemes $\Delta$ represents the constant time step at the $n^{th}$ iterate, $\Delta W \equiv \Delta W_n$ is a normally distributed random variable with mean 0 and variance $\Delta$ and $\Delta Z \equiv$
\( \Delta Z_n \) is a normally distributed random variable with mean 0, variance \( \frac{1}{3} \Delta^3 \) and the covariance between \( \Delta W \) and \( \Delta Z \) is \( \frac{1}{2} \Delta^2 \).

Recall from Section 2.2.2 that two normally distributed random variables can be determined from two Gaussian distributed random variables \( G_1 \) and \( G_2 \) via

\[
\Delta W = G_1 \sqrt{\Delta} \quad \Delta Z = \frac{1}{2} \Delta^{3/2} \left( G_1 + \frac{1}{\sqrt{3}} G_2 \right).
\]

Furthermore, recall that it is a property of weak schemes that \( \Delta W \) can be replaced by a simpler process which have moments equal to the order of the system. See Kloeden and Platen [45] for details. Thus, in the cases examined here, \( \Delta W \) could be replaced by \( \Delta \tilde{W} \) where \( \Delta \tilde{W} \) is three point distributed.

Finally, \( Y_0 = X_0 = X_{t_0} \) is the initial starting value, \( Y_1, Y_2, Y_3 \ldots \) denote the numerical approximations to \( X_1, X_2, X_3 \ldots \), \( a = a(x) \) and \( b = b(x) \). Also, all displayed derivatives using the prime notation are derivatives with respect to \( x \).

**The 2.0 order Weak Taylor Scheme**

In this autonomous case with a single noise source, \( d = m = 1 \), the order 2.0 weak scheme reduced to

\[
Y_{n+1} = Y_n + a\Delta + b\Delta W + \frac{1}{2} bb' \{ (\Delta W)^2 - \Delta \} \\
+ a'b \Delta Z + \frac{1}{2} \left( aa' + \frac{1}{2} a''b \right) \Delta^2 \\
+ \left( ab' + \frac{1}{2} b''b \right) \{ \Delta W \cdot \Delta - \Delta Z \}
\]

for \( n = 0, 1, 2, \ldots \). Here \( a = a(Y_n) \) and \( b = b(Y_n) \). Since the system under consideration had additive noise, ie. \( b(Y_n) \) =constant, then the spatial derivative \( b' = b'' = 0 \) so the scheme further simplified to

\[
Y_{n+1} = Y_n + a\Delta + b\Delta W \\
+ a'b \Delta Z + \frac{1}{2} \left( aa' + \frac{1}{2} a''b \right) \Delta^2.
\]

Upon entering expressions for \( a \) and \( b \) and the derivatives of \( a \), the scheme for \( n = 0, 1, 2, \ldots \) can be expressed in the form

\[
Y_{n+1} = Y_n + A_n \Delta + \sqrt{2}(\Delta W + B_n \Delta Z) + \frac{1}{2} \left( A_n B_n - 6q_1 Y_n \right) \Delta^2
\]
where
\[ A_n = -\eta_1 Y_n^3 + \eta_2 Y_n \quad \text{and} \quad B_n = -3\eta_1 Y_n^2 + \eta_2. \]

Note that \( \Delta Z \) can be replaced by \( \frac{1}{2} \Delta W \cdot \Delta \). With this substitution the above expression is then identical to the one generated by MAPLE as described in Section 4.4.2.

**Implicit 2.0 order Weak Taylor Series**

In this case the order 2.0 weak implicit scheme for \( n = 0, 1, 2, \ldots \) reduced to
\[
Y_{n+1} = Y_n + a(Y_{n+1}) \Delta + b \Delta W \\
- \frac{1}{2} \left\{ a(Y_{n+1})a'(Y_{n+1}) + \frac{1}{2} b^2 (Y_{n+1})a''(Y_{n+1}) \right\} \\
+ \frac{1}{2} \left\{ -a'b + ab' + \frac{1}{2} b'' b^2 \right\} \Delta W \cdot \Delta.
\]

Since \( b \) was constant this scheme simplified to
\[
Y_{n+1} = Y_n + a(Y_{n+1}) \Delta + b \Delta W \\
- \frac{1}{2} \left\{ a(Y_{n+1})a'(Y_{n+1}) + \frac{1}{2} b^2 a''(Y_{n+1}) \right\} \\
- \frac{1}{2} a'(Y_{n+1})b \Delta W \cdot \Delta.
\]

Upon substituting expressions for \( a, b \) and the derivatives of \( a \), the above equation can be written in the form
\[
AY_{n+1}^5 + BY_{n+1}^3 + CY_{n+1} = D_n
\]

where
\[
A = \frac{3}{2} \eta_1^2 \Delta^2 \\
B = \eta_1 \Delta (1 - 2 \eta_2 \Delta) \\
C = 1 - \eta_2 \Delta + (\eta_2^2 - 6 \eta_1) \frac{\Delta^2}{2} \\
D_n = Y_n + \sqrt{2} \Delta W \left( 1 + \frac{\Delta}{2} (3 \eta_1 Y_n^2 - \eta_2) \right).
\]

In this implementation a nonlinear equation needed to be solved for \( Y_{n+1} \) at each time step. A simple Newton–Raphson algorithm with prescribed error stopping strategy was incorporated to solve the resulting nonlinear equation. This led to a substantial increase in the computational requirements for this scheme. Note that in theory the implicit schemes can run with a larger time step thus minimising their CPU usage and negating the effect of the nonlinear equation solving.
Weak 2.0 order Predictor Corrector

For \( d = m = 1 \) the Predictor Corrector pair have the form

**Predictor:**

\[
\bar{Y}_{n+1} = Y_n + a \Delta + \frac{1}{2} a'b \Delta W \cdot \Delta + \frac{1}{2} \left( a'^2 + \frac{1}{2} a''b^2 \right) \Delta^2 + \Psi_n
\]

and

**Corrector:**

\[
Y_{n+1} = Y_n + \frac{1}{2} \left( a(\bar{Y}_{n+1}) + a \right) \Delta + \Psi_n
\]

where

\[
\Psi_n = b \Delta W + \frac{1}{2} b' \{ (\Delta W)^2 - \Delta \} + \frac{1}{2} \left( ab' + \frac{1}{2} b'' \Delta W \cdot \Delta \right)
\]

Since \( b \) was constant, the \( \Psi_n \) term reduced to

\[
\Psi_n = b \Delta W.
\]

Upon entering expressions for \( a \) and \( b \) and the derivatives of \( a \), the scheme became

**Predictor:**

\[
\bar{Y}_{n+1} = Y_n + A_n \Delta + \sqrt{2} \Delta \bar{W} + \frac{\sqrt{2}}{2} B_n \Delta \bar{W} \cdot \Delta + \frac{\Delta^2}{2} (A_n B_n - 6 \eta_1 Y_n)
\]

and

**Corrector:**

\[
Y_{n+1} = Y_n + \frac{\Delta}{2} \left( A_n + \eta_1 Y_{n+1} - \eta_1 \bar{Y}_{n+1} \right) + \sqrt{2} \Delta \bar{W}
\]

where \( A_n = a(Y_n) \) and \( B_n = b(Y_n) \). Recall that \( \Delta \bar{W} \) which is three point distributed, see Section 2.2.3, can be used in place of the more complex \( \Delta W \).

Weak 2.0 order Derivative Free Implicit

This scheme avoids the use of derivatives of the drift term in the implicit weak 2.0 order Taylor series scheme. For \( d = m = 1 \) this derivative free implicit scheme was

\[
Y_{n+1} = Y_n + \frac{1}{2} (a(Y_{n+1}) + a(Y_n)) \Delta \\
+ \frac{1}{4} \left( b(Y^+) + b(Y^-) + 2b(Y_n) \right) \Delta W \\
+ \frac{1}{4} \left( b(Y^+) + b(Y^-) \right) \{ (\Delta W)^2 - \Delta \} \Delta^{-1/2}
\]
\[ Y^\pm = Y_n + a(Y_n)\Delta \pm b(Y_n)\sqrt{\Delta}. \]

Upon entering expressions for \( a \) and \( b \) this led to

\[ AY_{n+1}^3 + BY_{n+1} - C_n = 0 \]

where

\[ A = \frac{\Delta}{2}\eta_1 \]
\[ B = 1 - \frac{\Delta}{2}\eta_2 \]
\[ C_n = Y_n + \frac{\Delta}{2}(\eta_2 Y_n - \eta_1 Y_n^3) + \sqrt{2}\Delta W. \]

As with the other implicit schemes the above cubic had to be solved numerically every time step for \( Y_{n+1} \). The cubic formula could have been employed for this task however the Newton-Raphson algorithm was employed to provide for the general case.

As expected, the generated scheme is identical to the MAPLE output described in Section 4.4.2.

**Weak 2.0 order Linearly Implicit**

For \( d = m = 1 \) the weak 2.0 order linearly implicit scheme was

\[ Y_{n+1} = Y_n + (AY_{n+1} + N)\Delta + b\Delta W + \frac{1}{2}bb'\{(\Delta W)^2 - \Delta\} \]
\[ + (A + N')b\Delta Z + \frac{1}{2}\left( (AY_{n+1} + N)(A + N') + \frac{1}{2}N''b^2 \right)\Delta^2 \]
\[ + \left( (AY_{n+1} + N)b' + \frac{1}{2}b''b^2 \right)\{\Delta W \cdot \Delta - \Delta Z\}. \]

In the example investigated, since \( A = \eta_2 \), \( N(x) = -\eta_1 x^3 \) and \( b \) is constant, the scheme reduced to

\[ Y_{n+1} = Y_n + (AY_{n+1} + N)\Delta + b\Delta W + (A + N')b\Delta Z \]
\[ + \frac{1}{2}\left( (AY_{n+1} + N)(A + N') + \frac{1}{2}N''b^2 \right)\Delta^2. \]

By substituting the expressions for \( A \), \( a \), \( b \) and the derivatives of \( N \) the scheme simplified to

\[ Y_{n+1} = \frac{f(Y_n)}{g(Y_n)}. \]
where
\[
 f(Y_n) = Y_n - \eta_1 Y_n^3 \Delta + \sqrt{2}(\Delta W + A_n \Delta Z) + \frac{\Delta^2}{2}(-A_n \eta_1 Y_n^3 - 6 \eta_1 Y_n),
\]
\[
 g(Y_n) = 1 - \eta_2 \Delta - \frac{\Delta^2 \eta_2}{2} A_n
\]
and with
\[
 A_n = \eta_2 - 3 \eta_1 Y_n^2.
\]
Note that since \(Y_{n+1}\) was calculated algebraically at each time step the linearly implicit scheme did not require the additional computational effort needed to solve a nonlinear equation at every time step.

### 4.4.4 Results

**Analysis of the 2.0 order Weak Taylor Series Scheme**

In this section an investigation into the number of realisations required to obtain a good fit between the simulated to actual results was undertaken. Firstly, a visual comparison of the final probability distribution against the expected distribution is presented in Figure 4.14 for increasing number of realisations. The results are for the case of \(\eta_1 = 40\) and \(\eta_2 = 10\) when using the weak 2.0 order time discretisation scheme.

For a quantitative comparison the \(\chi^2\) statistic, mean and variance were computed for the standard case of \(\eta_1 = 40\) and \(\eta_2 = 10\) using the weak 2.0 order Taylor Series scheme for a variety of experiments with different numbers of realisations. The \(\chi^2\) statistic provides a measure of the deviation between what is observed and what is expected. If \(\chi^2 < \chi^2_{\text{critical}}\), then accept the null hypothesis that the observed follows the expected distribution. The critical value of the \(\chi^2\) statistic at the 1% level with 34 degrees of freedom is \(\chi^2_{\text{critical}} = 56.0685\). The statistics together with the measure of CPU time required are presented in Table 4.2.
<table>
<thead>
<tr>
<th>Number of realisations</th>
<th>$\chi^2$ value</th>
<th>Average CPU (seconds)</th>
<th>Average Mean</th>
<th>Average Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>11.82</td>
<td>22.12</td>
<td>-0.02800</td>
<td>0.21537</td>
</tr>
<tr>
<td>$5 \times 10^2$</td>
<td>17.6</td>
<td>108.79</td>
<td>-0.00900</td>
<td>0.21313</td>
</tr>
<tr>
<td>$10^3$</td>
<td>8.3</td>
<td>205.75</td>
<td>-0.00730</td>
<td>0.22262</td>
</tr>
<tr>
<td>$5 \times 10^3$</td>
<td>17.1</td>
<td>1071.74</td>
<td>-0.00832</td>
<td>0.21356</td>
</tr>
<tr>
<td>$10^4$</td>
<td>29.2</td>
<td>2080.48</td>
<td>-0.00254</td>
<td>0.21344</td>
</tr>
<tr>
<td>$5 \times 10^4$</td>
<td>34.1</td>
<td>10377.21</td>
<td>-0.00248</td>
<td>0.21703</td>
</tr>
<tr>
<td>$10^5$</td>
<td>18.4</td>
<td>20852.81</td>
<td>-0.00179</td>
<td>0.21629</td>
</tr>
<tr>
<td>$5 \times 10^5$</td>
<td>34.8</td>
<td>104554.47</td>
<td>-0.00082</td>
<td>0.21659</td>
</tr>
<tr>
<td>$10^6$</td>
<td>20.9</td>
<td>208384.43</td>
<td>0.00036</td>
<td>0.21702</td>
</tr>
<tr>
<td>$5 \times 10^6$</td>
<td>21.7</td>
<td>1057818.20</td>
<td>-0.00004</td>
<td>0.21715</td>
</tr>
</tbody>
</table>

Table 4.2: Effect of the number of simulations on the $\chi^2$ statistic, summary statistics and CPU time required.
Figure 4.14: Graphical comparison of the effect of number of realisations on the solution.
As expected, and shown in Figure 4.15, the CPU time used increased linearly with the increase in the number of realisations. From this analysis, it was decided that $10^4$ realisations per experiment was sufficient for an acceptable value of the fit statistic as well as leading to a reasonable computer turn around time.

![Effect of number of simulations on CPU time required.](image)

**Figure 4.15: Effect of number of simulations on CPU time required.**

**Effect of starting value**

Initial runs with the weak 2.0 order Taylor Series scheme used a starting value of $Y_0 = 0.5$ for each realisation. An analysis was made to determine if the starting value had any effect on the overall results. Starting values of $\{-4, -3, \ldots, 3, 4\}$ and $\{-0.9, -0.8, \ldots, 0.8, 0.9\}$ were used in a set of experiments. The effect of the starting value on the value of $\chi^2$ statistic, mean and variance are presented in Table 4.3. This table demonstrates that there is no major observable difference between the statistics. Since it seemed that the starting value did not affect the overall results, subsequent experiments used a constant starting value of 0.5.
<table>
<thead>
<tr>
<th>Initial value</th>
<th>$\chi^2$ value</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4.0</td>
<td>12.0</td>
<td>0.00407</td>
<td>0.21765</td>
</tr>
<tr>
<td>-3.0</td>
<td>12.9</td>
<td>0.00211</td>
<td>0.21631</td>
</tr>
<tr>
<td>-2.0</td>
<td>16.1</td>
<td>0.00431</td>
<td>0.21295</td>
</tr>
<tr>
<td>-1.0</td>
<td>19.4</td>
<td>-0.00371</td>
<td>0.21660</td>
</tr>
<tr>
<td>-0.9</td>
<td>30.8</td>
<td>-0.00368</td>
<td>0.21978</td>
</tr>
<tr>
<td>-0.8</td>
<td>30.6</td>
<td>-0.00361</td>
<td>0.21497</td>
</tr>
<tr>
<td>-0.7</td>
<td>25.8</td>
<td>-0.00874</td>
<td>0.21509</td>
</tr>
<tr>
<td>-0.6</td>
<td>20.4</td>
<td>0.00094</td>
<td>0.21695</td>
</tr>
<tr>
<td>-0.5</td>
<td>10.6</td>
<td>-0.00277</td>
<td>0.21666</td>
</tr>
<tr>
<td>-0.4</td>
<td>10.7</td>
<td>0.00156</td>
<td>0.21447</td>
</tr>
<tr>
<td>-0.3</td>
<td>13.6</td>
<td>-0.00502</td>
<td>0.21540</td>
</tr>
<tr>
<td>-0.2</td>
<td>25.9</td>
<td>0.00517</td>
<td>0.26141</td>
</tr>
<tr>
<td>-0.1</td>
<td>15.8</td>
<td>0.00821</td>
<td>0.21781</td>
</tr>
<tr>
<td>0.0</td>
<td>15.6</td>
<td>0.00372</td>
<td>0.21741</td>
</tr>
<tr>
<td>0.1</td>
<td>8.2</td>
<td>0.00172</td>
<td>0.21765</td>
</tr>
<tr>
<td>0.2</td>
<td>18.8</td>
<td>0.00222</td>
<td>0.21470</td>
</tr>
<tr>
<td>0.3</td>
<td>10.0</td>
<td>-0.00240</td>
<td>0.21434</td>
</tr>
<tr>
<td>0.4</td>
<td>17.0</td>
<td>-0.00189</td>
<td>0.21668</td>
</tr>
<tr>
<td>0.5</td>
<td>7.0</td>
<td>-0.00112</td>
<td>0.21600</td>
</tr>
<tr>
<td>0.6</td>
<td>20.7</td>
<td>-0.00621</td>
<td>0.21812</td>
</tr>
<tr>
<td>0.7</td>
<td>22.7</td>
<td>0.00129</td>
<td>0.21668</td>
</tr>
<tr>
<td>0.8</td>
<td>13.4</td>
<td>-0.00076</td>
<td>0.21585</td>
</tr>
<tr>
<td>0.9</td>
<td>24.4</td>
<td>0.00534</td>
<td>0.21895</td>
</tr>
<tr>
<td>1.0</td>
<td>9.5</td>
<td>0.00123</td>
<td>0.21827</td>
</tr>
<tr>
<td>2.0</td>
<td>7.7</td>
<td>-0.00463</td>
<td>0.21664</td>
</tr>
<tr>
<td>3.0</td>
<td>28.6</td>
<td>-0.00432</td>
<td>0.21924</td>
</tr>
<tr>
<td>4.0</td>
<td>13.1</td>
<td>0.00093</td>
<td>0.21625</td>
</tr>
</tbody>
</table>

Table 4.3: Effect of initial value on the $\chi^2$ statistic and summary statistics for the weak 2.0 order Taylor series scheme with $\eta_1 = 40$ and $\eta_2 = 10$. 
Comparison of schemes

For comparison of the schemes the standard case of $\eta_1 = 40$ and $\eta_2 = 10$ was investigated using each scheme. Figure 4.16 presents a graphical comparison of the observed results (histogram bars) to the actual distribution (line) for the five schemes. Furthermore, Table 4.4 compares the value of $\chi^2$ statistic, mean and variance for the schemes and Table 4.5 compares the CPU requirements for each scheme. As Table 4.4 describes, all schemes modelled the expected distribution very well. From Table 4.5 the computationally quickest scheme was the linearly implicit scheme. The other implicit schemes suffered in comparison because of the added requirement to solve a nonlinear equation at each time step. In all likelihood, the advantages of the linearly implicit scheme will increase as the order and complexity of the system studied increase.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$\chi^2$ value</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taylor Series</td>
<td>29.2</td>
<td>-0.00832</td>
<td>0.21344</td>
</tr>
<tr>
<td>Implicit</td>
<td>24.3</td>
<td>0.00540</td>
<td>0.21451</td>
</tr>
<tr>
<td>Predictor-Corrector</td>
<td>19.7</td>
<td>-0.00891</td>
<td>0.21586</td>
</tr>
<tr>
<td>Implicit Derivative Free</td>
<td>11.7</td>
<td>0.00862</td>
<td>0.21732</td>
</tr>
<tr>
<td>Linearly Implicit</td>
<td>9.8</td>
<td>0.00098</td>
<td>0.21587</td>
</tr>
</tbody>
</table>

Table 4.4: Effect of scheme used on the $\chi^2$ statistic and summary statistics for the $\eta_1 = 40$ and $\eta_2 = 10$ case.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taylor Series</td>
<td>2080.48</td>
</tr>
<tr>
<td>Implicit</td>
<td>5891.42</td>
</tr>
<tr>
<td>Predictor-Corrector</td>
<td>2132.85</td>
</tr>
<tr>
<td>Implicit Derivative Free</td>
<td>3301.73</td>
</tr>
<tr>
<td>Linearly Implicit</td>
<td>1789.67</td>
</tr>
</tbody>
</table>

Table 4.5: Effect of scheme used on the CPU time required for the $\eta_1 = 40$ and $\eta_2 = 10$ case.
Figure 4.16: Comparison of observed (histograms) to actual probability density (line) for the standard case of $\eta_1 = 40$ and $\eta_2 = 10$ for the five schemes investigated: (a) Taylor Series; (b) Implicit; (c) Predictor-Corrector; (d) Derivative Free; (e) Linearly Implicit.
Effect of varying $\eta_2$

The order 2.0 weak Taylor series scheme was used to compare simulation results to expected results for the cases where $\eta_2$ was varied. Figure 4.17 provides a graphical comparison of each case. Disturbingly, as the value of $\eta_2$ increased, the simulated results did not model the actual distribution at all well. Table 4.6 provides a summary of the calculated statistics. It seems that as $\eta_2$ increased a greater proportion of the simulated values shifted to the right peak leading to a loss of symmetry. This also occurred when the other four weak schemes were tested as well as occurring for a number of strong schemes also tested.

Figure 4.17: Comparison of observed (histograms) to actual probability density (line) for the weak 2.0 order Taylor series scheme with $\eta_1 = 40$ and varying $\eta_2$: (a) $\eta_2 = 20$; (b) $\eta_2 = 40$; (c) $\eta_2 = 50$; (d) $\eta_2 = 80$. 
<table>
<thead>
<tr>
<th>$\eta_2$</th>
<th>$\chi^2$ value</th>
<th>$E(x)$</th>
<th>Var($x$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>29.2</td>
<td>-0.00832</td>
<td>0.21344</td>
</tr>
<tr>
<td>20</td>
<td>32.2</td>
<td>0.017800</td>
<td>0.43807</td>
</tr>
<tr>
<td>40</td>
<td>9880.8</td>
<td>0.97172</td>
<td>0.02765</td>
</tr>
<tr>
<td>50</td>
<td>9984.3</td>
<td>1.10210</td>
<td>0.01478</td>
</tr>
<tr>
<td>80</td>
<td>10023.5</td>
<td>1.40717</td>
<td>0.00711</td>
</tr>
</tbody>
</table>

Table 4.6: Effect of the value of $\eta_2$ on the $\chi^2$ statistic and summary statistics for the weak 2.0 order Taylor series scheme with $X_0 = 0.5$.

In spite of what was determined earlier about the effect of the starting value, I investigated whether the starting value affected the results at higher values of $\eta_2$. Upon using a starting value of $Y_0 = -0.5$ the same shifting effect was observed, however this time the shift was to the left peak. However, using an initial value of $Y_0 = 0.0$ led to a symmetric distribution with no shifting and hence a very good fit. To further investigate, simulations were systematically run using starting values of $\{-0.9, -0.8, \ldots, 0.8, 0.9\}$ and the statistics of these investigations are presented in Table 4.7. From this table it can be seen that the further in magnitude the starting value was from zero, the worse the fit. Interestingly though, averaging the observed results for the listed cases in Table 4.7 led to a very good fit. See Figure 4.18 where the case with $\eta_2 = 80$ is presented.

![Figure 4.18: Comparison of observed (histograms) to actual (line) results for single (a) and averaged (b) simulations for the weak 2.0 order Taylor series scheme for the $\eta_1 = 40$ and $\eta_2 = 80$ case.](image-url)
The choice of $Y_0 = 0.0$ as starting value is not always suitable, especially where the drift or diffusion terms imply a discontinuity at the origin. For that reason a number of runs were made where the initial value was not set constant for every simulation but was a randomly specified value on $[-1, 1]\{0\}$. Figure 4.19 presents a graphical comparison for the case when $\eta_2 = 80$ and using a variable starting value. The use of variable starting value with the other schemes also provided similar results.

![Graphical comparison](image_url)

Figure 4.19: Comparison of simulated (histograms) and actual (line) results for the weak 2.0 order Taylor series scheme using a varying starting value for the $\eta_1 = 40$ and $\eta_2 = 80$ case.
<table>
<thead>
<tr>
<th>Initial value</th>
<th>$\chi^2$ value</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.9</td>
<td>10011.2</td>
<td>-1.40783</td>
<td>0.00728</td>
</tr>
<tr>
<td>-0.8</td>
<td>10004.9</td>
<td>-1.40817</td>
<td>0.00742</td>
</tr>
<tr>
<td>-0.7</td>
<td>10025.1</td>
<td>-1.40869</td>
<td>0.00719</td>
</tr>
<tr>
<td>-0.6</td>
<td>10019.4</td>
<td>-1.40602</td>
<td>0.00740</td>
</tr>
<tr>
<td>-0.5</td>
<td>10010.6</td>
<td>-1.40669</td>
<td>0.00757</td>
</tr>
<tr>
<td>-0.4</td>
<td>10005.1</td>
<td>-1.40563</td>
<td>0.01044</td>
</tr>
<tr>
<td>-0.3</td>
<td>9836.2</td>
<td>-1.39464</td>
<td>0.041946</td>
</tr>
<tr>
<td>-0.2</td>
<td>8503.6</td>
<td>-1.29548</td>
<td>0.30759</td>
</tr>
<tr>
<td>-0.1</td>
<td>3947.7</td>
<td>-0.88389</td>
<td>1.21066</td>
</tr>
<tr>
<td>0.0</td>
<td>9.0</td>
<td>-0.01633</td>
<td>1.98907</td>
</tr>
<tr>
<td>0.1</td>
<td>3925.6</td>
<td>0.87984</td>
<td>1.20975</td>
</tr>
<tr>
<td>0.2</td>
<td>8572.7</td>
<td>1.30038</td>
<td>0.29114</td>
</tr>
<tr>
<td>0.3</td>
<td>9851.6</td>
<td>1.39603</td>
<td>0.04032</td>
</tr>
<tr>
<td>0.4</td>
<td>10002.9</td>
<td>1.40637</td>
<td>0.00887</td>
</tr>
<tr>
<td>0.5</td>
<td>10023.5</td>
<td>1.40717</td>
<td>0.00711</td>
</tr>
<tr>
<td>0.6</td>
<td>10009.0</td>
<td>1.40693</td>
<td>0.00806</td>
</tr>
<tr>
<td>0.7</td>
<td>9997.9</td>
<td>1.40654</td>
<td>0.00817</td>
</tr>
<tr>
<td>0.8</td>
<td>9999.7</td>
<td>1.40603</td>
<td>0.00812</td>
</tr>
<tr>
<td>0.9</td>
<td>10004.9</td>
<td>1.40624</td>
<td>0.00785</td>
</tr>
</tbody>
</table>

Table 4.7: Effect of initial value on the $\chi^2$ statistic and summary statistics for the weak 2.0 order Taylor series scheme for the $\eta_1 = 40$ and $\eta_2 = 80$ case.
Overall comparison of schemes

For the purposes of an overall comparison the simulations were run for all the schemes using the variable starting value approach. The results are summarised in Tables 4.8, 4.9 and 4.10 which compare the $\chi^2$ statistic, mean and variance for all schemes investigated and all cases studied. In all experiments the simulated results fit the actual distribution very well.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Value of $\eta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Taylor Series</td>
<td>18.7</td>
</tr>
<tr>
<td>Implicit</td>
<td>18.0</td>
</tr>
<tr>
<td>Predictor-Corrector</td>
<td>18.8</td>
</tr>
<tr>
<td>Implicit Derivative Free</td>
<td>18.9</td>
</tr>
<tr>
<td>Linearly Implicit</td>
<td>17.9</td>
</tr>
</tbody>
</table>

Table 4.8: Effect of scheme used and value of $\eta_2$ on the $\chi^2$ statistic. Recall that the critical $\chi^2$ value is 56.0685

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Value of $\eta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>Taylor Series</td>
<td>0.00157</td>
</tr>
<tr>
<td>Implicit</td>
<td>-0.00192</td>
</tr>
<tr>
<td>Predictor-Corrector</td>
<td>0.00171</td>
</tr>
<tr>
<td>Implicit Derivative Free</td>
<td>-0.00025</td>
</tr>
<tr>
<td>Linearly Implicit</td>
<td>-0.00103</td>
</tr>
<tr>
<td>Exact</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 4.9: Effect of scheme used and value of $\eta_2$ on the mean.
<table>
<thead>
<tr>
<th>Scheme</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>50</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taylor Series</td>
<td>0.21706</td>
<td>0.43653</td>
<td>0.97323</td>
<td>1.23034</td>
<td>1.98620</td>
</tr>
<tr>
<td>Implicit</td>
<td>0.21757</td>
<td>0.43656</td>
<td>0.97341</td>
<td>1.23035</td>
<td>1.98824</td>
</tr>
<tr>
<td>Predictor–Corrector</td>
<td>0.21723</td>
<td>0.43704</td>
<td>0.97392</td>
<td>1.22972</td>
<td>1.98665</td>
</tr>
<tr>
<td>Implicit Derivative Free</td>
<td>0.21714</td>
<td>0.43573</td>
<td>0.97321</td>
<td>1.22928</td>
<td>1.98621</td>
</tr>
<tr>
<td>Linearly Implicit</td>
<td>0.21729</td>
<td>0.43796</td>
<td>0.97309</td>
<td>1.22967</td>
<td>1.98654</td>
</tr>
<tr>
<td>Exact</td>
<td>0.21615</td>
<td>0.43569</td>
<td>0.97252</td>
<td>1.22888</td>
<td>1.99725</td>
</tr>
</tbody>
</table>

Table 4.10: Effect of scheme used and value of $\eta_2$ on the variance.
4.5 Comparison of Methods

Overall, the stochastic Taylor series based time discretisation schemes provided more accurate solutions than the FEM schemes when applied to stiff SDEs. The only disadvantage of the time discretisation scheme approach was that a random initial value was required to avoid the effects of stiffness.

In general, the FEM schemes can suffer in cases where the boundary conditions are crucial to the analysis. This was not the case in the example considered as the Simulink approach was shown to provide a good first order approximation. This was the case even with the stiff system investigated, thus the domain of interest could be determined. Note however, the stochastic Taylor series based time discretisation schemes can be run independent of knowledge of the domain. If the domain is not known, suitably relevant histogram bins can be organised at the postprocessing stage after analysis of the generated results. In my analyses, as I had first hand knowledge of the domain from the analytic solution (and from the Simulink approach), relevant histogram bins were initially set up thus avoiding any postprocessing.

4.6 Chapter Summary

This chapter has presented two approaches to the solution of one dimensional FPK equations or their associated SDEs; the FEM approach and the the stochastic time discretisation scheme approach.

The Simulink graphical modelling approach was first used to obtain initial insights into the solution domain of the SDE and the MAPLE package was used to create and verify the time discretisation schemes.

The stochastic time discretisation scheme approach generated accurate results irrespective of the stiffness of the system. A range of variants of the weak 2.0 order schemes such as implicit, derivative free, predictor corrector and the relatively new linearly implicit scheme were also tested and produced satisfactory results. The linearly implicit scheme was impressive, especially with its ability to solve stiff SDE systems without the computational burden usually associated with implicit schemes. The FEM approach had considerable problems with stiffness which could not be alleviated by increasing the number of elements.
Chapter 5

FPK Equation in Two Dimensions
5.1 Introduction

The two dimensional FPK equation has many applications in mathematical physics and engineering. From Spencer and Bergman [83] the FPK equation for a Duffing oscillator with negative linear restoring force is

\[
\frac{\partial p}{\partial t} = -\frac{\partial}{\partial x_1}(x_2 p) + \frac{\partial}{\partial x_2} \left[ \left\{ 2\zeta \omega_0 x_2 + \omega_0^2 x_1 (\gamma + \varepsilon x_1^2) \right\} p \right] + \frac{1}{2} \frac{\partial^2}{\partial x_2^2}(2Dp). \tag{5.1}
\]

Here \( D = D_{12} \omega_0^2 x_1^4 + D_{22} \) and \( \zeta, \omega_0, \varepsilon, \gamma, D_{11} \) and \( D_{22} \) are parameters of the oscillator. Its solution \( p(x_1, x_2) \) at any time \( t \) is a probability density function (pdf) which describes the likelihood that at time \( t \) the oscillator is at \( x_1 \) with velocity \( x_2 \).

The FPK equation for the Duffing oscillator, subject only to additive Gaussian white noise \( (D_{11} = 0) \), has the explicitly known stationary pdf

\[
p(x_1, x_2) = C_0 \exp \left\{ -\frac{\zeta \omega_0}{D_{22}} \left( x_2^2 + \omega_0^2 x_1^2 \left( \gamma + \frac{1}{2} \varepsilon x_1^2 \right) \right) \right\}. \tag{5.2}
\]

where \( C_0 \) is a normalisation constant. For more details see Liu [58] and Caughey [9]. This pdf is bimodal with two stable and one unstable equilibria and thus its inherent nonlinearity makes it a good test of the robustness of the solution approaches. The evolution in time of a pdf is known for linear systems of arbitrary dimension, however, such solutions are rare for nonlinear systems of dimension greater than one. My analysis will also investigate the time evolution of the pdf solution, equation (5.2) of FPK equation (5.1).

The FPK equation (5.1) reduces to the general vector form of an SDE

\[
dX_t = a(X_t)dt + b(X_t) dW_t
\]

or specifically to the system of SDEs

\[
dX^1_t = X^2_t dt \\
dX^2_t = \left( -2\zeta \omega_0 X^2_t - \gamma \omega_0^2 X^1_t - \varepsilon \omega_0^2 (X^1_t)^3 \right) dt + \sqrt{2D_{22}} dW^1_t - \sqrt{2D_{11}} \omega_0^2 X^1_t dW^2_t. \tag{5.3}
\]

Here \( W^1_t \) and \( W^2_t \) represent two independent Wiener processes where \( W^j_t \sim N(0, t) \), \( j = 1, 2 \). The noise intensity is built into the stochastic differential equation through the parameters \( \sqrt{2D_{22}} \) and \( \sqrt{2D_{11}} \) rather than into the Wiener process itself.
Following Bergman et al [5] the parameters used were $\varepsilon = 0.1$, $\zeta = 0.2$, $\omega_0 = 1$ and $\gamma = -1.0$. Furthermore, the parameter $D_{11} = 0$ led to the case of additive noise only and this formed the basis of the first numerical experiment. In the other experiments, $D_{11}$ took values 0.08 and 0.24 which represent 20% and 60% respectively of the additive noise parameter $D_{22} = 0.4$. 
5.2 Control System Solution

5.2.1 Overview

In this section, details of the use of the Simulink graphical modelling approach to obtain the initial solution of the FPK equation (5.1) will be presented. This initial solution will be used to determine the domain of interest of the solution and it is of particular use in the FEM solution approach.

5.2.2 Simulink 2D SDE model

Figure 5.1 presents the Simulink model for solving a two dimensional autonomous SDE system. It generated one trajectory, \((X_1, X_2)\), of an SDE with drift and diffusion functions as entered into the corresponding drift \((a_1(X_1,X_2)\) and \(a_2(X_1,X_2)\)) and diffusion \((b_1(X_1,X_2,dW_1,dW_2)\) and \(b_2(X_1,X_2,dW_1,dW_2)\)) blocks. These blocks were combined with sum and product blocks, and with White Noise 1 and White Noise 2 blocks to build the SDE structures. It has manual switches to set the initial values for the simulation to either user input constants or random numbers. It also has other manual switches to save results to datafiles or to provide graphical presentation of the trajectory.

Note that Simulink allows the user to create their own blocks so, for example, a block could be created to solve a single SDE. This block could then be re-used multiple times to solve a multidimensional system of SDEs. This would be the best approach to tackle higher dimensional systems without having to visually debug complex process control system diagrams. In this way Simulink is very much an object oriented approach to simulation modelling. For example, Figure 5.2 presents an example of a modular Simulink approach that could be used to solve a multidimensional system of SDEs.
Figure 5.1: A SIMULINK model to solve 2 dimensional SDEs.
Figure 5.2: Modular SIMULINK model to solve multidimensional SDEs (upper) with respective details of modules (lower).
The solution of the coupled set of equations (5.3) was in the form of a two dimensional probability distribution of the steady state, or final \((X_1, X_2)\) value, of many trajectories. A MATLAB script was written to run the Simulink model 10\(^4\) times and store interim and final values of the trajectory in the MATLAB workspace for post-processing and graphical summaries. The script used was

```
%% This m-file sets up the data needed to run Simulink "SDE22"
%%
%% _____ initialise constants
z=0.2;
w=1.0;
g=-1.0;
e=0.1;
D11=0.4;
D22=0.0;

%% _____ clear storage arrays
clear y1_x1
clear y1_x2
clear X
clear Y
clear Z

%% _____ describe domain
X=[-15:1:15];
Y=[-20:1:20];
Z=zeros(31,41,50);

%% _____ specify number of runs and the completion time
nruns=10000;
ntime=16;

%% _____ computation loop
for i=1:nruns
    sim('multi_22')
    for j=1:ntime
        y1_x1(j,i)=X1_simout(j);
y1_x2(j,i)=X2_simout(j);
    end
end
```
\[ Z(16+\text{round}(X1_{\text{simout}}(j)), 21+\text{round}(X2_{\text{simout}}(j))), j) = \\
Z(16+\text{round}(X1_{\text{simout}}(j)), 21+\text{round}(X2_{\text{simout}}(j))), j) + 1/n\text{runs}; \]

end

end

% output graphics

for \( j = 1 : n\text{time} \)
    figure(j)
    \% surf(Y,X,Z(:, :, j))
    \% contour(Y,X,Z(:, :, j))
    \% stem3(Y,X,Z(:, :, j))
    \% grid on
    pcolor(Y,X,Z(:, :, j))
    shading interp
    colorbar
    colormap(flipud(gray))
    \% colormap(hsv)
end

% store data

\text{dummy} = \text{yan\_x1'};
\text{save data1.out dummy -ascii}
\text{dummy} = \text{yan\_x2'};
\text{save data2.out dummy -ascii}

Note that after the computation loop there is also code to provide a variety of graphic outputs such as surface plots, contour plots and three dimensional histograms.
5.2.3 Results

Following Spencer and Bergman [83], the initial \((x_1, x_2)\) distribution used was \((N(0, 0.5), N(10, 0.5))\). This initial distribution was used by Spencer and Bergman to mimic a point distribution as the FEM model could not cope with an initial point distribution.

The results of the simulations undertaken are presented in Figures 5.3 and 5.4. Firstly, Figure 5.3 presents the stationary pdf solution once any initial or external influences had diminished. The results follow the exact solution as given in equation (5.2).

![Figure 5.3: Simulink generated pdf solution.](image)

Figure 5.4 describes the time evolution of the solution in unit time intervals until steady state. This result corresponds to those of Spencer and Bergman [83], as presented in Figure 5.5. This result also corresponds to my results using time discretisation schemes, see Section 5.4.1 later in this thesis for details.
Figure 5.4: Time evolution of the SIMULINK generated solution of the Duffing oscillator with additive noise.
Figure 5.5: Scanned results from Spencer and Bergman [83].
5.3 Finite Element Method (FEM) Solution

5.3.1 Overview

Spencer and Bergman [83] utilised a Bubnov-Galerkin FEM with 4 node quadrilateral element discretisation over a truncated rectangular domain. For the FEM approach the domain had to be made sufficiently large to eliminate the diffusion of the probability mass through the boundary. Thus, for a solution that is widely dispersed, for instance if \( p \) is in reality \( N(\mu, \sigma^2) \) distributed and \( \sigma >> 1 \), then the FEM mesh would need to be large enough to detect the solutions. With an infinite domain, the FEM approach has three options. The first option is to divide an interior subdomain into a FEM grid and surround it by “infinite elements”. The second option is to follow the first option and apply an FEM grid to a subdomain but then model the rest of the domain with boundary elements or a series solution. The third option is to neglect the outer infinite region and consider only the subdomain. This option is particularly relevant to the solution of FPK equations in that their solutions are pdfs and are therefore bounded to zero at the infinite domain. Langley [52] intimated that the first two approaches led to a substantial increase in the computational overheads as large complex numerical integrations were involved.

To determine what constitutes a suitably large enough domain requires further analysis which is typically an analysis of the linearised version of the SDE to determine the mean squared response. Langley [52] showed that for the Duffing oscillator FPK equation, a factor of three had to be applied to the mean square response values to determine the bounds of the truncated domain. In this analysis the domain of interest was determined via Simulink in Section 5.2.

To use the standard Galerkin FEM to determine the time evolution solution of the general SDE

\[
dX = a(t, X)dt + b(t, X)dW_t
\]

where \( a \) is the drift and \( b \) the diffusion, the discretisation must be assembled into the form

\[
M\dot{P} - LP = 0.
\]

Here \( P \) is the resulting pdf solution of the general SDE and \( M \) and \( L \) are the coefficient matrices. This needs to be further discretised in time using the Crank-Nicholson method
such that
\[
\left(M - \frac{\Delta L}{2}\right) P^{n+1} = \left(M + \frac{\Delta L}{2}\right) P^n
\]
where \( n \) denotes the \( n^{th} \) time step.

In the following sections I will describe my use of the FEM to solve for the stationary solution of the FPK equation 5.1 and my use of MAPLE to expedite the process.

### 5.3.2 FEM Approach

Recall from Chapter 2 that the FEM involves firstly truncating the infinite solution domain to a suitably large finite region and dividing this into a number of connected two dimensional rectangular finite elements. In this analysis, rectangular elements with sides parallel and perpendicular to the coordinate axes were used. The four corner points (nodes) of the rectangular elements are numbered as per Figure 5.6.

![Diagram of a two dimensional finite element with four nodes](image)

Figure 5.6: General two dimensional finite element with four nodes

The value of \( p \) within each element, \( p^e \), is determined by

\[
p^e(x_1, x_2) = \sum_{i=1}^{4} p_i N_i(x_1, x_2)
\]

where \( p_i \) is the value of \( p(x_1, x_2) \) at each node \( i = 1, \ldots, 4 \) and \( N_i \) is the shape function for node \( i \). The shape functions, \( N_i \), are chosen to give the value one at node \( i \) and zero at all other nodes. For example, Figure 5.7 provides a representation of \( N_1 \).

Using analogous mathematical derivations to those used in the one dimensional case in Chapter 2, (see also \[52, 53\]), the FEM solution reduced to first evaluating the nodal coefficients

\[
k_{rs}^e = \sum_{i=1}^{2} \sum_{k=1}^{4} g_i^k f_{ikrs} - \frac{1}{2} \sum_{i=1}^{2} \sum_{j=1}^{2} B_{ij} h_{ijs},
\]
for \( r = 1, \ldots, 4 \) and \( s = 1, \ldots, 4 \) for each element. The next stage required combining them in such a way that for each node

\[
\sum_{e^*} \left\{ \sum_{k=1}^{4} k^k_{r,s} p_s \right\} = 0,
\]

where \( \sum_{e^*} \) represents the sum of all elements that contain that node.

The constants \( g^k_i \) are the values of \( g_i \) at node \( k \) and \( g_i(x_1, x_2) \) is approximated by

\[
g_i(x_1, x_2) = \sum_{k=1}^{4} g^k_i N_k(x_1, x_2).
\]

Furthermore, the coefficients \( B_{ij} \) are the elements of a matrix \( B = 2\pi b b^T \) where \( b \) is the diffusion matrix of the SDE corresponding to the FPK equation (5.1). Finally, we define

\[
\begin{align*}
    f_{ikrs} &= \int_{e^*} \int N_k N_s \frac{\partial}{\partial x_i} \{ N_r \} \, dx_1 \, dx_2 \\
    h_{ijrs} &= \int_{e^*} \int \frac{\partial}{\partial x_i} \{ N_r \} \frac{\partial}{\partial x_j} \{ N_s \} \, dx_1 \, dx_2
\end{align*}
\]  

(5.4)

and note that these are defined solely in terms of the shape functions and their piecewise derivatives.

### 5.3.3 Use of Maple in the solution process

Using standard numerical schemes the evaluation of the above double integrals (5.4) can require substantial computational effort. In an effort to reduce the computational effort Maple can be used to first express these integrals in terms of a set of parameters. Maple can then convert these expressions into Fortran or C code for inclusion in a solution program.
Specifically, changing notation to use \( x = x_1 \) and \( y = x_2 \), always setting the coordinates of any element’s Node 1 as \((a,b)\) and setting the width and height of each element to constant \( \Delta x \) and \( \Delta y \) respectively as in Figure 5.8, then the shape functions used were

\[
N_1(x, y) = \left(1 - \frac{x - a}{\Delta x}\right) \left(1 - \frac{y - b}{\Delta y}\right)
\]
\[
N_2(x, y) = \left(1 - \frac{x - a}{\Delta x}\right) \left(\frac{y - b}{\Delta y}\right)
\]
\[
N_3(x, y) = \left(\frac{x - a}{\Delta x}\right) \left(1 - \frac{y - b}{\Delta y}\right)
\]
\[
N_4(x, y) = \left(\frac{x - a}{\Delta x}\right) \left(\frac{y - b}{\Delta y}\right)
\]

Figure 5.8: General coordinates of nodes for each element

Since the shape functions were defined in terms of the four parameters \( a, b, \Delta x \) and \( \Delta y \), then all constants \( f_{ikrs} \) and \( h_{ijrs} \) could also be defined in terms of these parameters.

I first defined the shape functions \( N_i \) in a general manner via

\[
N_i(x, y) = \left(2 - f(i) + (-1)^i \frac{(x - a)}{\Delta x}\right) \left(2 - g(i) + (-1)^i \frac{(y - b)}{\Delta y}\right)
\]

where

\[
f(i) = \lfloor i/2 + 0.5 \rfloor
\]
\[
g(i) = i - (f(i) - 1)f(i)
\]

and \( \lfloor \rfloor \) represents the greatest integer function.
I then used MAPLE (Version 5) to enter the following expressions

\[ f := (i) \rightarrow \text{trunc}(1/2*i+1/2); \]

\[ g := (i) \rightarrow i-(f(i)-1)*f(i); \]

\[ N := (i) \rightarrow (2-f(i)*(-1)^f(i)*(x-a)/dx)*(2-g(i)*(-1)^g(i)*(y-b)/dy); \]

Here the MAPLE function \( N(i,x,y) \) represented the shape functions \( N_i(x,y) \). Note that the MAPLE variables \( dx \) and \( dy \) represent \( \Delta x \) and \( \Delta y \) respectively.

The other main requirements were the derivatives of the shape functions, so I entered

\[ DN := (i, x, y, j) \rightarrow \text{if } j=1 \text{ then diff}(N(i,x,y),x) \]

\[ \text{else diff}(N(i,x,y),y) \text{ fi}; \]

Hence the MAPLE function \( DN(i,x,y,j) \) represented \( \frac{\partial}{\partial x_i} (N_j(x,y)) \).

The components of each element matrix are made up of \( f_{ibr_s} \) and \( h_{ijrs} \) terms. To determine these components I then entered the MAPLE expressions

\[ F := (i, k, r, s) \rightarrow \]

\[ \int(\int(N(k,x,y)*N(s,x,y)*DN(r,x,y,i),x = a .. a+dx),y = b .. b+dy); \]

and

\[ H := (i, j, r, s) \rightarrow \]

\[ \int(\int(DN(r,x,y,i)*DN(s,x,y,j),x = a .. a+dx),y = b .. b+dy); \]

Hence the MAPLE functions \( F(i,k,r,s) \) and \( H(i,j,r,s) \) represented \( f_{ibr_s} \) and \( h_{ijrs} \) respectively.

The next requirements were the functions \( g_1(x,y) \) and \( g_2(x,y) \) and the values of \( g_1 \) and \( g_2 \) at the four nodes of each element. These are all system dependent. The system specific functions \( g_i \) were entered by defining the MAPLE functions \( G1(x,y) \) and \( G2(x,y) \).

For the system under investigation with \( \zeta = 0.2, \omega_0 = 1.0, \gamma = -1.0 \) and \( \varepsilon = 0.1 \) I used the MAPLE commands

\[ G1 := (x, y) \rightarrow y; \]

\[ G2 := (x, y) \rightarrow -y-x-.1*x^3; \]
Continuing, the MAPLE function \( G(i,k) \) below represents \( g_i^k \) for \( i = \{1,2\} \) and \( k = \{1,2,3,4\} \).

\[
G := (i, k) \rightarrow \text{if } i = 1 \text{ then} \\
G_1(a + (\text{floor}((k-1)/2) \mod 2) \cdot dx, b \cdot (k-1 \mod 2) \cdot dy) \\
\text{else} \\
G_2(a + (\text{floor}((k-1)/2) \mod 2) \cdot dx, b \cdot (k-1 \mod 2) \cdot dy) \\
\text{fi;}
\]

Furthermore, in this example for the matrix \( B \) I used the MAPLE command

\[
B := (i,j) \rightarrow \text{if } i=2 \text{ and } j=2 \text{ then } D22+2*D11*x^2 \text{ else } 0.0 \text{ fi;}
\]

and specifically for the additive noise case

\[
B:=(i,j) \rightarrow \text{if } i=2 \text{ and } j=2 \text{ then } 0.4 \text{ else } 0.0 \text{ fi;}
\]

Finally, the element array components \( k_{rs} \) were determined. Using MAPLE I entered

\[
K := (r,s) \rightarrow \text{sum('sum('G(i,j)*F(i,j,r,s)', 'j'=1..4)', 'i'=1..2) - 1/2*sum('sum('B(i,j)*H(i,j,r,s)', 'j'=1..2)', 'i'=1..2);}
\]

Hence entering the MAPLE command \( K(r,s); \) with values of \( r \) and \( s \) from 1 to 4, generated the general form for each array element. For example, entering the MAPLE command \( K(1,1); \) generated the following output

\[
-1/8*b*dy+(b+dy)*(1/36*(b+dy)*(-dy^3+3*b^3+3*b*dy^2+5*b^2*dy)/dy^3-1/36*b^2*(3*b^2+6*dy^2+8*b*dy)/dy^3)+(b+dy)*(1/72*(b+dy)*(-dy^3+3*b^3+3*b*dy^2+5*b^2*dy)/dy^3-1/72*b^2*(3*b^2+6*dy^2+8*b*dy)/dy^3)-1/12*(-4*b+a_*1*a^3)*dx+(-4*b^4+4*dy+a_*1*a^3)*(1/24*dx*(b+dy)*(2*b^2+2*b*dy-dy^2)/dy^3-1/24*dx*b^2*(2*b+3*dy)/dy^3)-1/36*(-4*b^2*dy+a*dx-1*(a+dx)*3)*dx+(-4*b^2*4*dy+a*dx-1*(a+dx)^3)*(1/72*dx*(b+dy)*(2*b^2+2*b*dy-dy^2)/dy^3-1/72*dx*b^2*(2*b+3*dy)/dy^3)-3666666666e-1*dx/dy^2*(b+dy)+.66666666666e-1*dx/dy^2*b
\]

To save the computational effort of determining the double integrals in a numerical solution of SDEs the expressions obtained for \( F(i,k,r,s), H(i,j,r,s) \) or \( K(r,s) \) were converted by MAPLE to optimised FORTRAN code. For example the MAPLE code
> fortran(K(1,1),optimized);

generated the output

\[
\begin{align*}
t2 & = dy^{**2} \\
t6 & = dx^{**2} \\
t7 & = t6*dy \\
t8 & = t6^{**2} \\
t11 & = dx*dy \\
t12 & = a^{**2} \\
t22 & = x^{**2} \\
t24 & = -30.E0*t2*dy-120.E0*b*t2+16.E0*dx*t2-30.E0*t7+3.E0*t8*dy-120 \\
# & .E0*dx*D22+12.E0*t11*t12*a+48.E0*t11*b-120.E0*t11*a+9.E0*t7*t12+9. \\
# & E0*t6*dx*dy*a-240.E0*dx*D11*t22 \\
t26 & = 0.1388889E-2/dy*t24
\end{align*}
\]

Numerical values of each element's nodal value of $k^e_{rs}$ from the FORTRAN code were verified by substituting values of a, b, dx and dy into MAPLE's expression for $K(r,s)$. For example the MAPLE code and its output is

> subs(a=-2,b=-2,dx=0.2,dy=0.2,K(1,1));

.01269544013

Note in general that my use of MAPLE to generate expressions for the $f_{ikrs}$ and $h_{ijrs}$ terms was a better option than determining expressions for $k^e_{rs}$. The reasoning for this is that the $f_{ikrs}$ and $h_{ijrs}$ terms are dependent only on the shape functions and their derivatives and not on the system specific governing functions $g_1$ and $g_2$ or the matrix components $B_{ij}$. Thus, once complete, my FEM numerical program could be used to solve many two dimensional FPK systems with only minimal changes to the code.
Examples of generating optimised code for $F(i,k,r,s)$ and $H(i,j,r,s)$ are

```latex
> F(1,2,3,4);

\[
\begin{align*}
3 & \quad 2 & \quad 2 & \quad 3 & \quad 3 \\
(b + dy) & (dy + b) & (dy - b) & dy + 3b & b (3b + 4dy) \\
\end{align*}
\]

\[
1/72 \-------------------------------------------------- \ 1/72 \--------------------------------------------------
3 \\
dy \\
\]

> fortran(F(1,2,3,4),optimized);

\[
\begin{align*}
t2 &= dy**2 \\
t3 &= t2 \times dy \\
t4 &= b**2 \\
t7 &= t4 \times b \\
t10 &= 1/t3 \\
t15 &= (b+dy) \times (t3+t4 \times dy-b \times t2+3 \times t7) \times t10/72-t7 \times (3b+4dy) \times t10/72
\end{align*}
\]

> H(1,2,3,4);

\[
\begin{align*}
2 \\
(b + dy) & b (2 dy + b) \\
\end{align*}
\]

\[
1/4 \-------------------------------------------------- \ 1/4 \--------------------------------------------------
2 \\
dy \\
\]

> fortran(H(1,2,3,4),optimized);

\[
\begin{align*}
t2 &= (b+dy)**2 \\
t3 &= dy**2 \\
t4 &= 1/t3 \\
t9 &= t2 \times t4/4 - b \times (2 \times dy + b) \times t4/4
\end{align*}
\]
5.3.4 Solution of the FPK Equation

The FEM approach with the support of MAPLE was employed to set up the FEM coefficients and FORTRAN code for the steady state solution of the the FPK equation (5.1). The generated results were almost identical to the results generated using the time discretisation schemes and to the FEM solutions given by Spencer and Bergman [83]. In contrast to Spencer and Bergman [83], my approach using MAPLE substantially reduced the inherent FEM procedural overheads and allowed me to create an FEM based solution approach that could readily be applied to many FPK equations.
5.4 Taylor Series Time Discretisation Solution

5.4.1 The Solution Process

In this analysis, six weak 2.0 order schemes were used to solve the two dimensional SDE system (5.3) corresponding to the FPK system (5.1) and were compared. In all the computer experiments undertaken the following steps were performed:

1. An initial distribution in \((x_1, x_2)\) was implemented.

2. The SDE was solved using each of the six schemes. Simulations were run from \(t = 0\) to the designated end time \(t = T = 30\) using constant time step of 0.001. The resulting sample path values at \(t = T\), \((X_T^1, X_T^2)\), were noted and a frequency distribution of the values of \(X_T^1\) and \(X_T^2\) over the domain of interest was updated.

3. The above step was repeated a set number of times, typically \(10^5\) times.

4. Where possible, the resulting frequency distribution was qualitatively compared to the actual probability density as given by the analytic solution.

5.4.2 Weak 2.0 Order Schemes Used

As described in Chapter 4, stochastic Taylor expansions are used on intervals \([t_n, t_{n+1}]\) and truncated to form one–step numerical schemes, the stochastic Taylor schemes.

Many variations of the stochastic Taylor schemes are possible, such as derivative free schemes which are simplified schemes in which the stochastic integrals are approximated by more easily generated random variables. Other variations include predictor–corrector schemes and the relatively new linearly implicit scheme.

The schemes investigated and implemented for the solution of the Duffing oscillator system are detailed below. As in the case of the solution of one dimensional SDEs, in all of the second order weak schemes considered here the Gaussian random variables in the schemes could be replaced, without loss of accuracy, by the simpler random variables \(\Delta \tilde{W}\) and \(\Delta \tilde{Z} = \Delta \tilde{W} \Delta\). Here, as described in Chapter 2, \(\Delta \tilde{W}\) is a three–point random variable.
Weak 2.0 order Taylor Series scheme:

The general weak 2.0 order Taylor series based time discretisation scheme was presented in section 2.2.3. When applied to the Duffing oscillator (5.3) the scheme for \( n = 0, 1, 2, \ldots \) was

\[
Y_{n+1}^1 = Y_n^1 \left( 1 - \frac{\nu}{2} \omega_0^2 \Delta^2 \right) - \frac{\varepsilon}{2} \omega_0^2 \Delta^2 (Y_n^1)^3 + (1 - \zeta \omega_0 \Delta) \Delta Y_n^2 \\
+ \frac{\Delta}{2} \left[ \sqrt{2D_{22}} \Delta W^1 - \sqrt{2D_{11}} \omega_0^2 Y_n^1 \Delta W^2 \right],
\]

\[
Y_{n+1}^2 = Y_n^1 [\gamma \omega_0^2 \Delta (\zeta \omega_0 \Delta - 1)], \quad \quad \quad \\
+ Y_n^2 \left[ 1 - 2 \zeta \omega_0 \Delta (1 - \zeta \omega_0 \Delta) - \frac{\nu}{2} \omega_0^2 \Delta^2 \right] \\
+ \varepsilon \omega_0^2 \Delta (\zeta \omega_0 \Delta - 1)(Y_n^1)^3 - \frac{3}{2} \varepsilon \omega_0^2 \Delta^2 (Y_n^1)^2 Y_n^2 \\
+ \sqrt{2D_{22}} (1 - \zeta \omega_0 \Delta) \Delta W^1 \\
+ \sqrt{2D_{11}} \omega_0^2 \left( Y_n^1 (\zeta \omega_0 \Delta - 1) - \frac{\Delta}{2} Y_n^2 \right) \Delta W^2.
\]

Weak 2.0 order Implicit scheme:

The general weak 2.0 order implicit scheme is presented in section 2.2.4. When applied to the Duffing oscillator the scheme for \( n = 0, 1, 2, \ldots \) was

\[
Y_{n+1}^1 - \frac{\Delta}{2} Y_{n+1}^2 = Y_n^1 + \frac{\Delta}{2} Y_n^2, \\
\left( \frac{\gamma}{2} \omega_0^2 \Delta \right) Y_{n+1}^1 + (1 + \zeta \omega_0 \Delta) Y_{n+1}^2 + \frac{\varepsilon}{2} \omega_0^2 \Delta (Y_{n+1}^1)^3 = Y_n^1 (1 - \zeta \omega_0 \Delta) \\
- \left[ \frac{\gamma}{2} \omega_0^2 Y_n^1 + \frac{\varepsilon}{2} \omega_0^2 (Y_n^1)^3 \right] \Delta \\
+ \sqrt{2D_{22}} \Delta W^1 \\
- \sqrt{2D_{11}} \omega_0^2 \left( Y_n^1 + \frac{\Delta}{2} Y_n^2 \right) \Delta W^2.
\]

Note that as is the case with most implicit schemes, the resulting expressions form a set of nonlinear equations that need to be solved at each timestep for \( Y_{n+1} \). In this case, the set of equations is quasi-linear as one equation is linear and the other is a cubic. Thus an algebraic approach could have been utilised to reduce the system into a single cubic equation which could be solved by the cubic formula. However, for the sake of generality, I implemented a Newton-Raphson method to numerically determine \( Y_{n+1} \).
Weak 2.0 order Linearly Implicit Scheme:

The linearly implicit scheme, based on an idea of Deuflhard [17] for deterministic differential equations and extended by Petersen [66] to a general implicit splitting approach for SDEs, generated a scheme where only the linear parts of the coefficients depended on $Y_{n+1}$ with the nonlinear parts depending on $Y_n$.

The weak 2.0 order linearly implicit scheme when applied to the Duffing oscillator had the following implementation for $n = 0, 1, 2, \ldots$

$$\left[ 1 + \frac{\gamma}{2} \omega_0^2 \Delta^2 \right] Y_{n+1}^1 + [\Delta (\zeta \omega_0 \Delta - 1)] Y_{n+1}^2 = Y_n^1 - \frac{\varepsilon}{2} \omega_0^2 \Delta^2 (Y_n^1)^3$$
$$+ \frac{\Delta}{2} \left[ \sqrt{2D_{22}} \Delta W_n^1 - \omega_0^2 Y_n^1 \sqrt{2D_{11}} \Delta W_n^2 \right],$$

$$\left[ \gamma \omega_0^2 \Delta (1 - \zeta \omega_0 \Delta) \right] Y_{n+1}^1$$
$$+ \left[ 1 + 2\zeta \omega_0 \Delta - \frac{\omega_0^2}{2} \Delta^2 (4\zeta^2 \omega - \gamma) \right] Y_{n+1}^2 = Y_n^2$$
$$+ \frac{\varepsilon \omega_0^2 (Y_n^1)^3}{2} \left[ (2\zeta \omega_0^2 - 3Y_n^2) \frac{\Delta^2}{2} - \Delta \right]$$
$$+ \sqrt{2D_{22}} (1 - \zeta \omega_0 \Delta) \Delta W^1$$
$$+ \sqrt{2D_{11}} \omega_0^2 \left[ (\zeta \omega_0 \Delta - 1) Y_n^1 - \frac{\Delta}{2} Y_n^2 \right] \Delta W^2.$$

Note that in this scheme the resulting set of equations to be solved at each time step is linear and so this avoids the need for a numerical nonlinear equation solver.

Weak 2.0 order Derivative Free Taylor Series scheme:

The general weak 2.0 order derivative free Taylor series based time discretisation scheme is presented by equation 2.23. When applied to the Duffing oscillator, the scheme for implementation for $n = 0, 1, 2, \ldots$ was
\[ Y_{n+1}^1 = Y_n^1 \left(1 - \frac{\gamma}{2} \omega_0^2 \Delta^2 \right) - \frac{\varepsilon}{2} \omega_0^2 \Delta^2 (Y_n^1)^3 + Y_n^2 \{ \Delta(1 - \zeta \omega_0 \Delta) \} \]
\[ + \frac{\Delta}{2} \left\{ \sqrt{2D_{11}} \Delta W^1 - \sqrt{2D_{22}} \omega_0^2 Y_n^1 \Delta W^2 \right\}, \]

\[ Y_{n+1}^2 = Y_n^1 [\gamma \omega_0^2 \Delta(\zeta \omega_0 \Delta - 1)] + Y_n^2 \left[1 - 2\zeta \omega_0 \Delta(1 - \zeta \omega_0) - \frac{\gamma}{2} \omega_0^2 \Delta^2 \right] \]
\[ + \frac{\varepsilon}{2} \omega_0^2 \Delta (2\zeta \omega_0 \Delta - 1)(Y_n^1)^3 - \frac{\varepsilon}{2} \omega_0^2 \Delta (Y_n^1 + Y_n^2 \Delta)^3 \]
\[ + \sqrt{2D_{22}}(1 - \zeta \omega_0 \Delta) \Delta W^1 \]
\[ + \sqrt{2D_{11}} \omega_0^2 \left(Y_n^1 (\zeta \omega_0 \Delta - 1) - \frac{\Delta}{2} Y_n^2 \right) \Delta W^2. \]

**Weak 2.0 order Derivative Free Implicit scheme:**

An implicit variant of the weak 2.0 order derivative free implementation scheme for \( n = 0, 1, 2, \ldots \), which is described in section 5.4.2, is presented below. Again, as this was an implicit scheme the resulting set of equations was nonlinear and the Newton-Raphson equation solving algorithm was required at each time step. The scheme is

\[ Y_{n+1}^1 - \frac{\Delta}{2} (Y_{n+1}^2)^3 + (1 + \zeta \omega_0 \Delta) Y_{n+1}^2 = Y_n^1 + \frac{\Delta}{2} Y_n^2, \]

\[ \frac{\gamma}{2} \omega_0^2 \Delta Y_{n+1}^1 + (1 + \zeta \omega_0 \Delta) Y_{n+1}^2 + \frac{\varepsilon}{2} \omega_0^2 \Delta (Y_{n+1}^1)^3 = Y_n^2 (1 - \zeta \omega_0 \Delta) \]
\[ - \left[ \frac{\gamma}{2} \omega_0^2 Y_n^1 + \frac{\varepsilon}{2} \omega_0^2 (Y_n^1)^3 \right] \Delta \]
\[ + \sqrt{2D_{22}} \Delta W^1 \]
\[ - \sqrt{2D_{11}} \omega_0^2 \left(Y_n^1 + \frac{\Delta}{2} Y_n^2 \right) \Delta W^2. \]
Weak 2.0 order Derivative Free Linearly Implicit Scheme:

Finally, a linearly implicit variant of the weak 2.0 order derivative free scheme for \( n = 0, 1, 2, \ldots \) is

\[
\begin{align*}
\left[ 1 + \frac{\gamma}{2} \omega_0^2 \Delta^2 \right] Y^1_{n+1} + [\Delta(\zeta_0 \Delta - 1)] Y^2_{n+1} &= Y^1_n - \frac{\varepsilon}{2} \omega_0^2 \Delta^2 (Y^1_n)^3 \\
&\quad + \frac{\Delta}{2} \left( \sqrt{2D_{22}} \Delta W^1_n - \omega_0^2 Y^1_n \sqrt{2D_{11}} \Delta W^2_n \right),
\end{align*}
\]

\[
\begin{align*}
\left[ \gamma \omega_0^2 (1 - \zeta_0 \Delta) \right] Y^1_{n+1} + \\
\left[ 1 - 2(\zeta_0 \Delta (\zeta_0 \Delta - 1)) + \frac{\gamma}{2} \omega_0^2 \Delta^2 \right] Y^2_{n+1} &= Y^2_n + \varepsilon \omega_0^2 \Delta (Y^1_n + Y^2_n \Delta)^3 \\
&\quad + \sqrt{2D_{22}} (1 - \zeta_0 \Delta) \Delta W^1_n \\
&\quad + \sqrt{2D_{11}} \omega_0^2 \left[ (\zeta_0 \Delta - 1) Y^1_n - \Delta \frac{Y^2_n}{2} \right] \Delta W^2.
\end{align*}
\]

5.4.3 Use of Maple in the solution process

Maple and the stochastic package were used to verify the weak 2.0 order Taylor series time discretisation scheme via the expression

\[
\text{wktay2( } [ x[2], -2*z*w*x[2]-g*w^2*x[1]-e*w^2*x[1]^3 ] , \\
[ [0,0] , [sqrt(2*D22),-sqrt(2*D11)] ] );
\]

Note that the Maple variables e, g, w and z represent \( \varepsilon \), \( \gamma \), \( \omega_0 \) and \( \zeta \) respectively. The above expression generated the output

\[
\text{table(}[ \\
\quad 1 = \left( Y^1_{n+1} = \\
\quad \quad Y^1_n + Y^2_n \Delta_n + \frac{1}{2} \%1 \Delta_n^2 + \frac{1}{2} \Delta_n \sqrt{2} \sqrt{D_{22}} \Delta Ws_{1n} - \frac{1}{2} \Delta_n \sqrt{2} \sqrt{D_{11}} \Delta Ws_{2n} \right) \\
\quad 2 = \left( Y^2_{n+1} = Y^2_n + \%1 \Delta_n + \frac{1}{2} \left( Y^2_n \left( -g w^2 - 3 e w^2 Y^1_n^2 \right) - 2 \%1 z w \right) \Delta_n^2 \\
\quad \quad + \left( \sqrt{2} \sqrt{D_{22}} - \Delta_n \sqrt{2} \sqrt{D_{22}} z w \right) \Delta Ws_{1n} \\
\quad \quad + \left( -\sqrt{2} \sqrt{D_{11}} \Delta_n \sqrt{2} \sqrt{D_{11}} z w \right) \Delta Ws_{2n} \right) \right])
\]
\[
\%1 := -2 z \, w \, Y_2^2 \, n - g \, w^2 \, Y_1 \, n - \varepsilon \, w^2 \, Y_1^3 \, n
\]

This expression corresponds exactly with scheme given by equation (5.5).

### 5.4.4 Results

The parameter values \( \zeta = 0.2, \omega_0 = 1.0, \gamma = -1.0, \varepsilon = 0.1 \) and \( D_{22} = 0.4 \) (following [5, 83]) were used in the three numerical experiments.

The parameter \( D_{11} \) was set to 0.0, 0.08 and 0.24 in Experiments 1, 2 and 3 respectively. For Experiment 1, the initial \((x_1, x_2)\) distribution was \((N(0, 0.5), N(10, 0.5))\) with computational domain \([-15 \leq x_1 \leq 15, -20 \leq x_2 \leq 20]\). In Experiments 2 and 3, the initial distribution was \((N(0, 0.5), N(0, 0.5))\) with computational domain \([-10 \leq x_1 \leq 10, -10 \leq x_2 \leq 10]\). Spencer and Bergman chose these independent component Gaussian initial conditions instead of the required point initial condition due to the computational instability inherent in their FEM procedure. The specification of the computational domain followed the initial solution generated via the SIMULINK approach in Section 5.2.

In each experiment, \(10^5\) realisations over the time domain \([0, 30]\) with constant step size \(\Delta = 0.001\) were simulated. The FORTRAN intrinsic random number generator was used to produce uniformly distributed \([0, 1]\) random numbers and these were transformed into standard Gaussian \(N(0, 1)\) values via the Polar Marsaglia method.

The six weak 2.0 order numerical schemes described in Section 5.4.2 were computed on a VAX 5000 single processor computer and on a variety of SUN workstations.

As with Bergman et al and Spencer et al [5, 83], contour plots and three dimensional surface representations of transient and steady state probability densities were generated.

**Experiment 1**

Contour plots and three dimensional surface representations of the transition densities were prepared on a \(60 \times 80\) rectangular grid over the computational domain. These contour plots for the output of the weak 2.0 order stochastic Taylor scheme, which are representative of those for the other schemes, are displayed at unit time intervals in Figure 5.9. A three dimensional surface representation of the stationary probability density function is presented in Figure 5.10. Comparison of these figures with the expected ana-
lytic solution and the Bubnov-Galerkin based FEM generated results presented in Figures 5 and 7 respectively of Spencer and Bergman [83] shows a good correspondence of results.
Figure 5.9: Evolution of Experiment 1 pdf in unit time steps.
Figure 5.10: Surface representation of Experiment 1 stationary pdf solution.
To investigate the effect of the timestep on the pdf solution, $10^5$ realisations of the weak 2.0 order Taylor series scheme were run with timesteps $\Delta = \{0.2, 0.1, 0.02, 0.01, 0.002\}$. The stationary pdf solutions at $T = 15$ are presented in Figure 5.11. No major discernible differences can be detected.

Figure 5.11: Effect of timestep on Experiment 1 stationary pdf. Timesteps are $\Delta = \{0.2, 0.1, 0.02, 0.01, 0.002\}$ respectively.

Finally, the effect of the number of realisations is presented in Figure 5.12 where the stationary pdf solution of Experiment 1 is compared for $10^4$ and $10^5$ realisations. As expected, the simulation with the larger number of realisations generated stationary pdfs with smoother contours.
Figure 5.12: Effect of number of realisations on Experiment 1 stationary pdf. Number of realisations are $10^4$ and $10^5$ respectively.
Experiments 2 & 3

There is no known analytic solution for the FPK equation for the Duffing oscillator subject to both additive and multiplicative noise, so a direct comparison with theory was impossible.

In both numerical experiments, contour plots and three dimensional surface representations were prepared from a $40 \times 40$ rectangular grid over the computational domain ($-10 \leq x_1 \leq 10, -10 \leq x_2 \leq 10$). The stationary probability density functions for the two experiments are presented in Figures 5.13 and 5.15. The generated results compare well with the Bubnov-Galerkin based FEM scheme results presented in Figures 7 and 11 in Bergman et al [5] as presented in Figures 5.14 and 5.16.

Figure 5.13: Stationary pdf solution for Experiment 2.

Figure 5.14: Bergman et al [5] results for Experiment 2.
Figure 5.15: Stationary pdf solution for Experiment 3.

Figure 5.16: Bergman et al [5] results for Experiment 3.
5.5 Comparison of Methods

Table 5.1 presents the relative CPU requirements for simulations undertaken as part of Experiment 1. These simulations were based on $10^5$ realisations of compiler optimised schemes run on the VAX 5000. The reference unit corresponds to 89619 CPU seconds (25 CPU hours). In the year 2001 context, 25 hours seems a long time, however these simulations were undertaken in 1994/1995 on the VAX which was already out of date and in the process of being superseded by quicker SUN workstations.

As anticipated, the fully implicit schemes ran substantially slower (more than twice) than the original explicit schemes. Of great interest was the speed of the linearly implicit versions of these schemes. These linearly implicit schemes have much potential since they have the advantage of being (partially) implicit and hence are more stable at a larger time step. They have this advantage without incurring the extra computational overhead of the fully implicit schemes. Finally, and again as anticipated, in all cases the derivative-free schemes ran quicker, though only marginally, than their counterparts. However, in all the cases examined, they required more complex programming code.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Relative CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taylor</td>
<td>1.0259</td>
</tr>
<tr>
<td>Derivative-free Taylor</td>
<td>1.0000</td>
</tr>
<tr>
<td>Implicit Taylor</td>
<td>2.2125</td>
</tr>
<tr>
<td>Implicit derivative-free Taylor</td>
<td>2.0526</td>
</tr>
<tr>
<td>Linearly implicit Taylor</td>
<td>1.0472</td>
</tr>
<tr>
<td>Linearly implicit derivative-free Taylor</td>
<td>1.0034</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of relative CPU times.

The most obvious advantages of the time discretisation numerical schemes considered here are their conceptual simplicity and the relative ease with which they can be implemented and parameter values changed. They can also be readily and easily applied to SDEs with combinations of additive and multiplicative noise. Implicit and linearly implicit versions of such schemes now allow hitherto intractable stiff equations to be solved successfully, while extrapolation methods based on weak schemes, see Kloeden Platen and Hofmann [42], can double the order of accuracy. A fairly complete error analysis is also
possible, with the theoretical discretisation error given by the order of the scheme and the sampling error (due to using only a finite number of sample paths) being the dominant factors. While not included in the figures presented in this chapter, all moments, confidence intervals and bands can be readily calculated and plotted if required.

Counterbalancing these advantages is the need to simulate a large number of sample paths to assure desired levels of accuracy. This is, however, less of a problem now with the widening accessibility of multi-processor computers or distributed networks of single processor machines. The turnaround times can be substantially reduced by running the simulations on a parallel or distributed processing multi-processor machine. Due to the structure of the schemes an almost $N$-fold reduction in computation time can be expected from a machine with $N$ processors. Further difficulties may also arise in generating the higher multiplicity stochastic integrals needed in the schemes, especially when several independent driving Wiener processes are present.

5.6 Chapter Summary

This chapter has described the use of three solution approaches to solve the two dimensional SDE that describes the Duffing oscillator for the cases of additive and combined additive with multiplicative noise. The time discretisation approach and the Simulink approach accurately predicted the time evolution of the the pdf solution. This is of particular importance in that this time evolution has only ever been demonstrated before by using the FEM approach. Even with the use of powerful symbolic manipulators such as Maple, the creation of FEM models in dimension two and greater is particularly cumbersome.
Chapter 6

FPK Equation in Three Dimensions
6.1 Introduction

In this chapter I will investigate three dimensional FPK systems. Specifically, I will investigate how the stochastic Taylor series time discretisation schemes cope with more complex systems than the one and two dimensional systems presented in Chapters 4 and 5 respectively. The FEM approach to three dimensional FPK systems is fraught with problems in regard to the computational approach to large matrix storage and sparse matrix techniques. As such I have not undertaken my own FEM investigations, rather I will quote from the limited literature.

The FPK systems investigated are presented in Wojtkiewicz, Spencer and Bergman [94]. Wojtkiewicz et al presented two systems. The first was a three state linear stochastic dynamical system with known analytic solution which was used as a benchmark system. The second was a nonlinear Duffing oscillator system with a parameter specifying whether the system had positive or negative linear restoring force.

6.1.1 Benchmark FPK system

The benchmark three dimensional system investigated has stationary FPK equation given by

\[
- \frac{\partial}{\partial x_1} (x_2p) - \frac{\partial}{\partial x_2} \left( \left(-2\zeta_0 x_2 - \omega_0^2 x_1 + x_3\right)p \right) - \frac{\partial}{\partial x_3} (-\alpha x_3 p) - D \frac{\partial^2 p}{\partial x_3^2} = 0. \tag{6.1}
\]

This corresponds to the three dimensional SDE system

\[
d \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} X_2 \\ -2\zeta_0 X_2 - \omega_0^2 X_1 + X_3 \\ -\alpha X_3 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 0 \\ \sqrt{2D} \end{bmatrix} dW_t \tag{6.2}
\]

where \( W \equiv W_t \) is a Gaussian white noise process with \( E(W(t)) = 0 \) and \( E(W(t_1)W(t_2)) = 2D \delta(t_2 - t_1) \) where \( D \) is the spectral density of the excitation. This benchmark system has analytic solution given by

\[
p(x) = \left(8\pi^3 |\Gamma|\right) \exp \left( -\frac{1}{2} x^T \Gamma^{-1} x \right)
\]
where $\mathbf{G}$ is the stationary covariance matrix of the system. Soong and Grigoriu [82] showed that $\mathbf{G}$ was given by

$$
\mathbf{G} = \begin{bmatrix}
E(X_1^2) & E(X_1X_2) & E(X_1X_3) \\
E(X_2X_1) & E(X_2^2) & E(X_2X_3) \\
E(X_3X_1) & E(X_2X_3) & E(X_3^2)
\end{bmatrix} = \begin{bmatrix}
\frac{D(2\zeta \omega_0 + \alpha)}{2\zeta \omega_0^2 \alpha^N} & 0 & \frac{D}{\alpha \mathbf{N}} \\
0 & \frac{D}{2\zeta \omega_0^2 N} & \frac{D}{\mathbf{N}} \\
\frac{D}{\alpha \mathbf{N}} & \frac{D}{\mathbf{N}} & \frac{D}{\alpha}
\end{bmatrix}
$$

where $\mathbf{N} = \omega_0^2 + 2\zeta \omega_0 \alpha + \alpha^2$. For the benchmark example, the parameters used were $\zeta = 0.2$, $\omega_0 = 1$, $\alpha = 1$ and $D = 0.4$. In this benchmark system the covariance matrix is

$$
\mathbf{G} = \begin{bmatrix}
0.583 & 0 & 0.16 \\
0 & 0.416 & 0.16 \\
0.16 & 0.16 & 0.4
\end{bmatrix}.
$$

In the plane where $X_3 = 0$ the exact solution has graphical representation given in Figure 6.1.

To test the accuracy of their FEM solution approach, Wojtkiewicz et al [94] used two techniques. The first was to determine the error between their FEM solution and the exact solution using the error norm

$$
||e|| = \left( \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{p}_i - p_i)^2 \right)^{\frac{1}{2}}.
$$

Here the FEM approximation $\mathbf{p}_i$ and the exact solution $p_i$ were determined at every node $i = 1, 2, \ldots, N$ of the FEM mesh. The second approach was to calculate the second moments of the solution and compare them to the elements of the covariance matrix $\mathbf{G}$. 
Figure 6.1: Solution of the benchmark FPK equation for $X_3 = 0$. 
6.1.2 Duffing Oscillator

The three dimensional Duffing oscillator system subjected to first order white noise has stationary FPK equation given by

\[- \frac{\partial}{\partial x_1}(x_2 p) - \frac{\partial}{\partial x_2}\left\{(-2\zeta \omega_0 x_2 - \gamma \omega_0^2 x_1 - \epsilon x_1^3 + x_3)p\right\} - \frac{\partial}{\partial x_3}(-\alpha x_3 p) - D \frac{\partial^2 p}{\partial x_3^2} = 0. \] (6.3)

Here \( \zeta \) is a damping parameter, \( \omega_0 \) is the natural frequency in the linear case if there was no damping, \( \epsilon \) is the magnitude parameter of the nonlinearity and \( \gamma = \{-1, 1\} \) is a parameter that specifies the sign of the linear restoring term. Specifically, if \( \gamma = 1 \) a unimodal response is evident whereas if \( \gamma = -1 \) a bimodal response eventuates.

This FPK system corresponds to the three dimensional SDE system

\[
d\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} X_2 \\ -2\zeta \omega_0 X_2 - \gamma \omega_0^2 X_1 - \epsilon X_1^3 - \omega_0 X_1 + X_3 \\ -\alpha X_3 \end{bmatrix} dt + \begin{bmatrix} 0 \\ 0 \\ \sqrt{2D} \end{bmatrix} dW_t
\]

where the Wiener process \( W_t \) is as specified for the benchmark system. The parameters investigated were \( \epsilon = 1, \omega_0 = 1, \zeta = 0.2, \alpha = 1, D = 0.4 \) and \( \gamma = \pm 1 \).

6.1.3 Solution Approaches

In the next section a Simulink model to solve the three dimensional benchmark system of SDEs given in equation (6.2) will be presented and used to determine the domain of interest. Following that the stochastic Taylor series based time discretisation schemes will be employed and compared to the known analytic solution in the benchmark case and to the FEM results of Wojtkiewicz et al [94] in the Duffing oscillator case. Finally the time discretisation scheme approach will be compared to the FEM approach and the Monte Carlo approach as described by Wojtkiewicz et al [94].
6.2 Control System Solution

This section will present the details of the use of the graphical modelling tool Simulink in the initial solution of the FPK equation (6.1). This initial solution was used to determine the domain of interest of the solutions and would be of particular interest in the FEM approach where knowledge of the domain is crucial.

6.2.1 Simulink 3D SDE Model

Figure 6.2 presents the Simulink model for solving a three dimensional autonomous SDE system. It generated one trajectory \((X_1, X_2, X_3)\) of an SDE system with drift \(a(X_1, X_2, X_3), a_2(X_1, X_2, X_3)\) and \(a_3(X_1, X_2, X_3)\), and diffusion \(b(X_1, X_2, X_3, dW_1, dW_2, dW_3), b_2(X_1, X_2, X_3, dW_1, dW_2, dW_3)\) and \(b_3(X_1, X_2, X_3, dW_1, dW_2, dW_3)\) blocks.

![Simulink model diagram](image)

Figure 6.2: A Simulink model to solve 3 dimensional autonomous SDE systems.
6.2.2 Results

Initially $5 \times 10^4$ realisations of the Simulink model were run over $0 \leq T \leq 20$. At unit
time intervals the results were stored for later use to generate the transition solution. The
following MATLAB script was used to control the Simulink model as well as determine
the resulting second moments.

```matlab
% This m-file sets up the data needed to run Simulink "MULTI_3D"

% Initialise system parameters
alpha=1.0;
omega0=1.0;
zeta=0.2;
D=0.4;
% Clear storage arrays
clear yan_x1
clear yan_x2
clear yan_x3
% Initialise simulation parameters
nruns=100;
% Run simulation and store results
for i=1:nruns
    sim('multi_3d')
    yan_x1(i)=X1_simout(21);          
    yan_x2(i)=X2_simout(21);          
    yan_x3(i)=X3_simout(21);          
    if mod(i,nruns/100)==0
        percent_complete=i/nruns*100
    end
end
% Save stored results to external data files
dummy=yan_x1';
save data_x1.out dummy -ascii
```
dummy=yan_x2';
save data_x2.out dummy -ascii
dummy=yan_x3';
save data_x3.out dummy -ascii
% Determine all second moments
C(1,1)=mean(yan_x1.*yan_x1);
C(1,2)=mean(yan_x1.*yan_x2);
C(2,1)=C(1,2);
C(1,3)=mean(yan_x1.*yan_x3);
C(3,1)=C(1,3);
C(2,2)=mean(yan_x2.*yan_x2);
C(2,3)=mean(yan_x2.*yan_x3);
C(3,2)=C(2,3);
C(3,3)=mean(yan_x3.*yan_x3)

Figure 6.3 presents the time evolution error in the Simulink generated second moments of X1, X2 and X3. From this I determined that running future Simulink simulations until T = 20 was sufficient to allow any transients to diminish to obtain the stationary solution.

Using Wojtkiewicz et al's error norm and comparisons I obtained the results presented in Table 6.1. Included in the table are the FEM results of Wojtkiewicz et al and the elapsed simulation time for the Simulink approach solution. The simulation was undertaken on a 700 MHz Pentium III PC.
Figure 6.3: Time variation of the errors of the second moments of the benchmark system via the Simulink approach.

<table>
<thead>
<tr>
<th>Moment</th>
<th>Exact</th>
<th>FEM</th>
<th>Simulink</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E(X_1^2)$</td>
<td>0.583</td>
<td>-0.58353</td>
<td>0.58312</td>
</tr>
<tr>
<td>$E(X_1 X_2)$</td>
<td>0</td>
<td>-0.00014</td>
<td>-0.00096</td>
</tr>
<tr>
<td>$E(X_1 X_3)$</td>
<td>0.16</td>
<td>0.16668</td>
<td>0.16754</td>
</tr>
<tr>
<td>$E(X_2^2)$</td>
<td>0.416</td>
<td>0.41712</td>
<td>0.42085</td>
</tr>
<tr>
<td>$E(X_2 X_3)$</td>
<td>0.16</td>
<td>0.16663</td>
<td>0.17069</td>
</tr>
<tr>
<td>$E(X_3^2)$</td>
<td>0.4</td>
<td>0.39994</td>
<td>0.40349</td>
</tr>
</tbody>
</table>

\[ ||e|| \quad 1.133 \times 10^{-4} \quad 5.394 \times 10^{-5} \]

<table>
<thead>
<tr>
<th>time</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18 hours</td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of the Simulated second moments of the benchmark system at $T = 20$ against the exact and FEM second moments.
The Simulink steady state solution at cross sectional plane $X_3 = 0$ is presented in Figure 6.4. These show similar form to the analytic solution but, as only $5 \times 10^4$ runs were undertaken, smooth contours are not present.

![Simulink generated solution](image)

Figure 6.4: Simulink generated solution for the benchmark system at $X_3 = 0$.

Overall the Simulink approach generated more accurate (by a factor of two) second moments for the system than the FEM approach. The domain of interest was determined from the steady state solution as $\{-2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2\}$. The final steady state solution was very inaccurate due to the small number of realisations completed. Running a much larger number of realisations was not undertaken due to the long turnaround time for the Simulink approach.
6.3 FEM Results

For completeness the results of Wojtkiewicz et al [94] are presented in this section.

Wojtkiewicz et al used a standard Bubnov-Galerkin FEM approach and in three dimensions it was an extension of the two dimensional version described in Section 5.3. Specifically Wojtkiewicz et al used a total of 125,000 eight node isoparametric “brick” finite elements over the computational cube \(-4 \leq x_i \leq 4, \ i = 1, 2, 3\) to determine the stationary solution of the FPK systems. Complex sparse matrix storage techniques were employed to minimise the computational memory requirements. Furthermore, the FEM approach generated matrix system for \(p(x)\) that required Krylov subspace based methods and other specialised techniques to solve the system of symmetric inhomogeneous equations.

Wojtkiewicz et al do not specify the computing platform employed nor the time taken to complete their FEM analyses. However, as their research was supported by two grants through the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, it is a fair assumption that supercomputers were necessary.

Scanned copies of the graphical results generated by Wojtkiewicz et al are presented for comparison purposes where relevant in the results section (Section 6.5) of this chapter.
6.4  Taylor Series Time Discretisation Solution

6.4.1  Introduction

In this analysis the weak Euler scheme (order 1.0), weak 2.0 and 3.0 order Taylor series schemes were employed against the benchmark and Duffing oscillator SDE systems corresponding to the FPK systems (6.1) and (6.3).

6.4.2  Schemes used – Benchmark system

For the benchmark system, equation (6.1), MAPLE generated the following schemes

Weak Euler scheme

\[
\begin{align*}
\text{table}([1] & \equiv Y_{1,n+1} = Y_{1,n} + Y_{2,n} \Delta_n, \\
2 & \equiv Y_{2,n+1} = Y_{2,n} + (-\omega Y_{1,n} - 2\zeta \omega Y_{2,n} + Y_{3,n}) \Delta_n, \\
3 & \equiv Y_{3,n+1} = Y_{3,n} - \alpha Y_{3,n} \Delta_n + \sqrt{2} \sqrt{\Delta} \Delta Ws_l_n)
\end{align*}
\]

Weak 2.0 order scheme

\[
\begin{align*}
\text{table}([1] & \equiv Y_{1,n+1} = Y_{1,n} + Y_{2,n} \Delta_n + \frac{(-\omega Y_{1,n} - 2\zeta \omega Y_{2,n} + Y_{3,n}) \Delta_n^2}{2}, \\
2 & \equiv Y_{2,n+1} = Y_{2,n} + (-\omega Y_{1,n} - 2\zeta \omega Y_{2,n} + Y_{3,n}) \Delta_n \\
& \quad + \frac{(-Y_{2,n} \omega - 2(-\omega Y_{1,n} - 2\zeta \omega Y_{2,n} + Y_{3,n}) \zeta \omega - \alpha Y_{3,n}) \Delta_n^2}{2} \\
& \quad + \frac{\Delta_n \sqrt{2} \sqrt{\Delta} \Delta \ Ws_l_n}{2}, \\
3 & \equiv Y_{3,n+1} = Y_{3,n} - \alpha Y_{3,n} \Delta_n + \frac{\alpha^2 Y_{3,n} \Delta_n^2}{2} + \left(\sqrt{2} \sqrt{\Delta} - \frac{\Delta_n \sqrt{2} \sqrt{\Delta} \alpha}{2} \right) \Delta Ws_l_n)
\end{align*}
\]
Weak 3.0 order scheme

\[ \begin{align*}
\text{table}[1] &= YI_{n+1} = YI_n + Y2_n \Delta_n + \left(-\omega YI_n - 2 \zeta \omega Y2_n + Y3_n\right) I_{0,0} \\
&\quad + \left(- Y2_n \omega - 2 \left(-\omega YI_n - 2 \zeta \omega Y2_n + Y3_n\right) \zeta \omega - \alpha Y3_n\right) I_{0,0,0} + \sqrt{2} \sqrt{D} I_{1,0,0}, \\
\text{table}[2] &= Y2_{n+1} = Y2_n + \left(-\omega YI_n - 2 \zeta \omega Y2_n + Y3_n\right) \Delta_n \\
&\quad + \left(- Y2_n \omega - 2 \left(-\omega YI_n - 2 \zeta \omega Y2_n + Y3_n\right) \zeta \omega - \alpha Y3_n\right) I_{0,0} + \sqrt{2} \sqrt{D} I_{1,0} \\
&\quad + \left(2 Y2_n \omega^2 \zeta + \left(-\omega YI_n - 2 \zeta \omega Y2_n + Y3_n\right) \left(-\omega + 4 \zeta^2 \omega^2\right) - \alpha Y3_n \left(-2 \zeta \omega - \alpha\right)\right) I_{0,0,0} \\
&\quad + \sqrt{2} \sqrt{D} \left(-2 \zeta \omega - \alpha\right) I_{1,0,0}, \\
\text{table}[3] &= Y3_{n+1} = Y3_n - \alpha Y3_n \Delta_n + \sqrt{2} \sqrt{D} \Delta W I_n + \alpha^2 Y3_n I_{0,0} - \sqrt{2} \sqrt{D} \alpha I_{1,0} \\
&\quad - \alpha^3 Y3_n I_{0,0,0} + \sqrt{2} \sqrt{D} \alpha^2 I_{1,0,0})
\end{align*} \]

Here the weak 2.0 order scheme makes use of multiple stochastic integrals which can be approximated by

\[ I_{0,0} = \frac{1}{2} \Delta^2, \quad I_{0,0,0} = \frac{1}{3!} \Delta^3, \quad I_{1,0} = \Delta \tilde{Z}, \quad I_{1,0,0} = \frac{1}{6} \Delta^2 \Delta \tilde{W} \]

where \( \Delta \tilde{W} \) and \( \Delta \tilde{Z} \) are as defined in Section 4.4.3.

### 6.4.3 Schemes used – Duffing oscillator system

For the Duffing oscillator system, equation (6.3), MAPLE generated the following schemes

**Euler scheme**

\[ \begin{align*}
\text{wkeuler}([x[2],-gamma*omega^2*x[1]-2*zeta] \\
\text{ta*omega*x[2]-epsilon*x[1]*3+x[3],-alpha*x[3]],[[0],[0],[sqrt(2*D)]]})
\end{align*} \]

with output
table([1 = Y_{n+1}^1 = Y_n + Y_{n}^2 \Delta_n,
2 = Y_{n+1}^2 = Y_n + (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \Delta_n,
3 = Y_{n+1}^3 = Y_n - \alpha Y_n \Delta_n + \sqrt{2} \sqrt{\Delta} W s I_n])

Weak 2.0 order scheme

> wktay2([x[2], -gamma*omega^2*x[1], -2*zet]
> a*omega*x[2] - epsilon*x[1]^3 + x[3], -alpha*x[3], [[0], [0], [sqrt(2)*D]]);

with output

table([1 = Y_{n+1}^1 = Y_n + Y_{n}^2 \Delta_n + \frac{(-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \Delta_n}{2},
2 = Y_{n+1}^2 = Y_n + (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \Delta_n
+ (Y_n^2 (-\gamma \omega^2 - 3 \epsilon Y_n^3) - 2 (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \zeta \omega - \alpha Y_n^3) \Delta_n
+ \frac{\Delta_n \sqrt{2} \sqrt{\Delta} W s I_n}{2},
3 = Y_{n+1}^3 = Y_n - \alpha Y_n \Delta_n + \frac{\alpha^2 Y_n^3 \Delta_n}{2} + \left(\frac{\sqrt{2} \sqrt{D}}{2} - \frac{\Delta_n \sqrt{2} \sqrt{D} \alpha}{2}\right) \Delta W s I_n])

Weak 3.0 order scheme

> wktay3([x[2], -gamma*omega^2*x[1], -2*zet]
> a*omega*x[2] - epsilon*x[1]^3 + x[3], -alpha*x[3], [[0], [0], [sqrt(2)*D]]);

with output

table([1 = Y_{n+1}^1 = Y_n + Y_{n}^2 \Delta_n + (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) I_{0,0}
+ (Y_n^2 (-\gamma \omega^2 - 3 \epsilon Y_n^3) - 2 (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \zeta \omega - \alpha Y_n^3) I_{0,0,0}
+ \sqrt{2} \sqrt{D} I_{1,0,0},
2 = Y_{n+1}^2 = Y_n + (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \Delta_n
+ (Y_n^2 (-\gamma \omega^2 - 3 \epsilon Y_n^3) - 2 (-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) \zeta \omega - \alpha Y_n^3) I_{0,0}
+ \sqrt{2} \sqrt{D} I_{1,0,0} + (Y_n^2 (-6 Y_n^3 e Y_n^3 - 2 (-\gamma \omega^2 - 3 \epsilon Y_n^3) \zeta \omega)) I_{0,0,0}
+ ((-\gamma \omega^2 Y_n^2 - 2 \zeta \omega Y_n^2 - \epsilon Y_n^3 + Y_n^3) (-\gamma \omega^2 - 3 \epsilon Y_n^3 + 4 \zeta \omega^2)) I_{0,0,0}
- \alpha Y_n^3 (-2 \zeta \omega - \alpha) I_{0,0,0} + \sqrt{2} \sqrt{D} (-2 \zeta \omega - \alpha) I_{1,0,0},
3 = Y_{n+1}^3 = Y_n - \alpha Y_n \Delta_n + \sqrt{2} \sqrt{D} W I_n + \alpha^2 Y_n I_{0,0}
- \sqrt{2} \sqrt{D} \alpha I_{1,0,0} - \alpha^3 Y_n I_{0,0,0} + \sqrt{2} \sqrt{D} \alpha^2 I_{1,0,0})}
Here the multiple stochastic integrals $I_{(0,0)}$, $I_{(0,0,0)}$, $I_{(1,0)}$ and $I_{(1,0,0)}$ were approximated as per the benchmark solution.

### 6.4.4 Benchmark system solution

Initially $10^4$ realisations using the stochastic Euler scheme were run over $0 \leq T \leq 50$. At unit time intervals the results were stored to generate the transition solution. Figure 6.5 presents the time evolution of the errors in the calculated second moments of $X_1$, $X_2$ and $X_3$. From this I determined that running future simulations until $T = 20$ was sufficient to allow any transients to diminish to obtain the stationary solution.

![Figure 6.5: Time variation of the errors of the second moments of the benchmark system via the stochastic Euler approach.](image)
For comparison purposes a total of $10^3$, $10^4$ and $10^5$ realisations were run for the Euler and weak 2.0 and 3.0 order Taylor series schemes using time steps of $\Delta = 0.01$ and 0.1. Each simulation ran until end time $T = 20$ to obtain the stationary solution. Parameters investigated were $\omega_0 = 1$, $\zeta = 0.2$, $\alpha = 1$ and $D = 0.4$. Using Wojtkiewicz et al’s error norm and covariance comparisons I obtained the results presented in Table 6.2 to 6.4. Included in the table are the FEM results of Wojtkiewicz et al and the elapsed simulation time in seconds for the Euler and weak 2.0 and 3.0 order Taylor series schemes.

<table>
<thead>
<tr>
<th>Moment</th>
<th>Exact</th>
<th>FEM</th>
<th>10^3</th>
<th>10^4</th>
<th>10^5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E(X_1^2)$</td>
<td>0.583</td>
<td>0.58353</td>
<td>0.7740</td>
<td>0.5907</td>
<td>0.7725</td>
</tr>
<tr>
<td>$E(X_1X_2)$</td>
<td>0</td>
<td>-0.00014</td>
<td>-0.0329</td>
<td>0.0104</td>
<td>-0.0435</td>
</tr>
<tr>
<td>$E(X_2X_3)$</td>
<td>0.16</td>
<td>0.16668</td>
<td>0.1423</td>
<td>0.1636</td>
<td>0.1557</td>
</tr>
<tr>
<td>$E(X_3^2)$</td>
<td>0.416</td>
<td>0.41712</td>
<td>0.6163</td>
<td>0.4463</td>
<td>0.6252</td>
</tr>
<tr>
<td>$E(X_2X_3)$</td>
<td>0.16</td>
<td>0.16663</td>
<td>0.1763</td>
<td>0.1585</td>
<td>0.1762</td>
</tr>
<tr>
<td>$E(X_3^2)$</td>
<td>0.4</td>
<td>0.39994</td>
<td>0.4310</td>
<td>0.3848</td>
<td>0.4226</td>
</tr>
</tbody>
</table>

Table 6.2: Comparison of the stochastic Euler scheme generated second moments of the benchmark system at $T = 20$ against the exact and FEM second moments.

<table>
<thead>
<tr>
<th>Moment</th>
<th>Exact</th>
<th>FEM</th>
<th>10^3</th>
<th>10^4</th>
<th>10^5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E(X_1^2)$</td>
<td>0.583</td>
<td>0.58353</td>
<td>0.6026</td>
<td>0.5527</td>
<td>0.5647</td>
</tr>
<tr>
<td>$E(X_1X_2)$</td>
<td>0</td>
<td>-0.00014</td>
<td>-0.0090</td>
<td>0.0292</td>
<td>-0.0017</td>
</tr>
<tr>
<td>$E(X_2X_3)$</td>
<td>0.16</td>
<td>0.16668</td>
<td>0.1878</td>
<td>0.1729</td>
<td>0.1607</td>
</tr>
<tr>
<td>$E(X_3^2)$</td>
<td>0.416</td>
<td>0.41712</td>
<td>0.4219</td>
<td>0.4162</td>
<td>0.4197</td>
</tr>
<tr>
<td>$E(X_2X_3)$</td>
<td>0.16</td>
<td>0.16663</td>
<td>0.1582</td>
<td>0.1692</td>
<td>0.1681</td>
</tr>
<tr>
<td>$E(X_3^2)$</td>
<td>0.4</td>
<td>0.39994</td>
<td>0.4305</td>
<td>0.4061</td>
<td>0.4263</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison of the weak 2.0 order scheme generated second moments of the benchmark system at $T = 20$ against the exact and FEM second moments.

The weak 2.0 and 3.0 order Taylor series scheme with $10^5$ realisations of timestep 0.01 displayed superior error norm than the FEM approach, whereas the same schemes with $10^4$ realisations had comparable error norm. The more accurate weak 3.0 order schemes
Table 6.4: Comparison of the weak 3.0 order scheme generated second moments of the benchmark system at $T = 20$ against the exact and FEM second moments.

took 3.3 and 3.8 hours respectively to complete. The most impressive aspect was that the time discretisation scheme simulations were undertaken on a 700 MHz Pentium III PC using code written in Visual Basic for Applications (VBA) within Microsoft Excel (97). Essentially, the Maple output of Section 6.4.2 was converted to VBA code though, for more complex systems, optimised Fortran from the Maple output could be first generated before converting to VBA.

Graphically, the generated results at $X_3 = 0$ from the stochastic Euler scheme for increasing number of realisations are presented in Figure 6.6. The simulation time step was $\Delta = 0.01$.

![Figure 6.6](image)

Figure 6.6: Stochastic Euler scheme solution of benchmark system with $10^4$, $10^5$ and $10^6$ realisations respectively.
Furthermore a comparison of the results generated from the Euler and weak 2.0 and 3.0 order Taylor series schemes is presented in Figure 6.7 for $10^6$ realisations with timestep $\Delta = 0.01$.

Figure 6.7: Stochastic Euler, weak 2.0 and weak 3.0 order Taylor series solution of benchmark system with $10^6$ realisations.

Overall, the stochastic Taylor series based schemes performed well in solving the benchmark three dimensional system. Unlike the FEM approach, which almost certainly required supercomputing facilities, these schemes were undertaken in around three and a half hours on a standard desktop PC in the inefficient Microsoft Excel computing environment. Better performance would have been observed if a stand alone PC was available rather than the networked PC I used.

Another advantage of the stochastic Taylor series based schemes is that the time evolution of the system can be obtained as a consequence of the solution approach without any extra computational burden. With the FEM approach the system needs to be further discretised in time to generate the transition distributions. Not surprisingly, due to the complexity involved, Wojtkiewicz et al [94] did not present any time evolution distributions.

For the benchmark system, Figure 6.8 presents the time evolution of the distribution in unit time intervals based on a simulation involving $10^6$ realisations.
Figure 6.8: Time evolution of the benchmark system in the $X_3 = 0$ plane via the stochastic Euler scheme.
6.4.5 Duffing oscillator system solution

Initially $10^4$ realisations using the stochastic Euler scheme were run over $0 \leq T \leq 50$ for both the unimodal ($\gamma = 1$) and bimodal ($\gamma = -1$) cases. At unit time intervals the results were stored to generate the transition solution. Figures 6.9 and 6.10 present the time evolution of the second moments of $X_1$, $X_2$ and $X_3$ for the unimodal and bimodal cases respectively. From this I determined that running future simulations until $T = 20$ was sufficient to allow any transients to diminish to obtain the stationary solution.

![Graph showing time variation of second moments](image)

Figure 6.9: Time variation of the second moments of the unimodal Duffing Oscillator system via the stochastic Euler scheme.
Figure 6.10: Time variation of the second moments of the bimodal Duffing Oscillator system via the stochastic Euler scheme.

The stochastic Euler scheme with $\Delta = 10^4$ was used to solve the Duffing oscillator system with parameters $\epsilon = 1$, $\omega_0 = 1$, $\zeta = 0.2$, $\alpha = 1$, $D = 0.4$ and $\gamma = \pm 1$.

For the unimodal case, $3.5 \times 10^6$ realisations were run and the generated results at the cross sectional planes $X_3 = \{-0.96, 0, 0.96\}$ are presented in Figures 6.11 to 6.13. They can be compared to the scanned FEM results from Wojtkiewicz, Spencer and Bergman [94] that are included. Excellent correspondence of results is evident.

For the Duffing oscillator system with $\gamma = 1$, Figure 6.14 presents the time evolution of the unimodal distribution in unit time intervals. This was based on $10^5$ realisations of the stochastic Euler scheme with time step $\Delta = 0.01$. 
Figure 6.11: Stochastic Euler solution of unimodal Duffing oscillator at $X_3 = -0.96$.

Figure 6.12: Stochastic Euler solution of unimodal Duffing oscillator at $X_3 = 0$.

Figure 6.13: Stochastic Euler solution of unimodal Duffing oscillator at $X_3 = 0.96$. 

Figure 6.14: Time evolution of the unimodal Duffing oscillator system in the $X_3 = 0$ plane via the stochastic Euler scheme.
Similar excellent correspondence is observed in the bimodal case at cross sections $X_3 = \{ -1.76, -0.96, -0.48, 0, 0.48, 0.96, 1.76 \}$ as presented in Figures 6.15 to 6.21 respectively. These figures are based on $5.5 \times 10^6$ realisations of the stochastic Euler scheme with $\Delta = 0.01$.

Figure 6.15: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = -1.76$.

Figure 6.16: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = -0.96$. 
Figure 6.17: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = -0.48$.

Figure 6.18: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 0$.

Figure 6.19: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 0.48$. 
Figure 6.20: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 0.96$.

Figure 6.21: Stochastic Euler solution of bimodal Duffing oscillator at $X_3 = 1.76$. 
For the Duffing oscillator system with $\gamma = -1$, Figure 6.22 presents the time evolution of the bimodal distribution in unit time intervals in the plane $X_3 = 0$. This was based on $10^5$ realisations of the stochastic Euler scheme with $\Delta = 0.01$. This time evolution of the bimodal system shows a greater richness of features than the unimodal case.
Figure 6.22: Time evolution of the bimodal Duffing oscillator system in the $X_3 = 0$ plane via the stochastic Euler scheme.
6.4.6 Microsoft Excel/VBA Random Number Generator

Since the simulations using the stochastic Taylor series based time discretisation schemes were written in VBA within Microsoft Excel, I first undertook a check of the intrinsic random number generator. The details of the random number generator algorithm used by Microsoft are:

\[
\text{random\_number}(n) = \text{fractional part of } [9821 \times \text{random\_number}(n-1) + 0.211327]
\]

where the value of the first random number, \(\text{random\_number}(1)\), is based on the system clock.

Microsoft details that their random number generator has a period of “over one million” which is quite poor. I undertook an independent test of the period and showed that it was 16,777,216 or \(2^{24}\) which is the mantissa size for IEEE standard floating point arithmetic. This period was greater than the total number of random numbers I needed in my analysis, but as this period is not particularly large, care would need to be applied if Microsoft Excel was used to solve much larger systems.

As a further test I undertook a Plot-Pairs test. The results are presented in Figure 6.23. There is no evidence of banding or other “non-random” features.

![Figure 6.23: Plot-Pairs diagram for the Microsoft Excel random number generator.](image-url)
6.5 Discussion

Johnson, Wojtkiewicz and Bergman [37] compared their numerical solution of the Duffing oscillator using the FEM approach and the Monte Carlo approach. They found that the Monte Carlo approach, when using as few as 50,000 realisations, was equivalent to the FEM approach when investigating the evolution of the second moments. However, the Monte Carlo approach required in the order of 200,000 realisations to provide the same order of performance when investigating the response pdf. Johnson et al have concluded that the FEM suffered in comparison to the Monte Carlo approach if greater fidelity (a finer grid of nodes for FEM) was required.

Systems of order higher than two pose significant difficulty when using the standard FEM approach due to memory requirements and computational expense. As pointed out by Johnson, Wojtkiewicz and Bergman [37], a standard \( d \) dimensional FEM solution with an \( n \) point grid in each spatial dimension and uniform time step requires a single reduction to upper triangular form of \( n^d \) equations followed by backward and forward substitution at each time step. Hence the number of required computations and memory allocation grows exponentially with the dimension of the system.

Spencer and Bergman admitted that their FEM technique was limited in applicability in that it could only be used efficiently to solve systems of dimension two and, with difficulty, dimension three. Wojtkiewicz, Spencer and Bergman [94] have only recently, and with difficulty, applied the FEM approach to a three dimensional system.

In this chapter, I have demonstrated that the time discretisation schemes of Kloeden and Platen can be successfully used to generate accurate solutions of three dimensional FPK equations through the numerical solution of the corresponding SDEs. In the solution of the comparative benchmark system, the weak 2.0 order Taylor series schemes generated results comparable to the FEM approach. Furthermore, the time discretisation schemes could also produce the time transition distributions. Most impressively, the time discretisation schemes could be run under Microsoft Excel on a standard desktop PC, whereas the FEM and the inaccurate Monte Carlo schemes required the use of complex programming and almost certainly a supercomputer.
6.6 Chapter Summary

This chapter has described the Taylor series time discretisation approach to three dimensional SDEs corresponding to FPK systems. Maple was used to generate the schemes employed and an initial solution was obtained by using Simulink.

Thus to date, the stochastic Taylor series based time discretisation approach has been successfully used to solve one, two and three dimensional FPK systems. The next chapter will extend the approach to higher dimensional systems by solving stochastic PDEs using spatial discretisation to obtain an $n$ dimensional SDE system.
Chapter 7

Multidimensional SDEs through SPDEs
7.1 Introduction

As discussed in Chapter 6, using the FEM approach to solve three dimensional FPK equations is a complex task. Extending the FEM approach to higher dimensional FPK systems is even more difficult. An alternative technique, that of solving the system of SDEs that correspond to a FPK equation via stochastic time discretisation schemes has been shown in this thesis to be acceptable in one, two and three dimensions. This analysis approach could be extended to higher dimensional FPK systems, however, few such systems are described in the literature. Instead, the stochastic time discretisation schemes will be applied to stochastic PDEs which reduce to multidimensional systems of SDEs.

Partial differential equations (PDEs) are used to model (noise free) complex behaviour. Typically such systems cannot be modelled by ODEs. Similarly, stochastic partial differential equations (SPDEs) are used to model complex “noisy” behaviour that cannot be modelled with SDEs.

The solution of SDEs has only been investigated in the last 40 to 50 years and only recently (in the last 15 years) have techniques been explored for SPDEs. This is due to the ready availability of computers and the phenomenal increase in computing power. These factors have been the catalysts for the creation of powerful numerical schemes for systems once thought impossible to solve. However, the field of study of SPDEs is in its infancy both in terms of analytical and numerical solution techniques. At this stage numerical techniques used in the solution of ODEs, PDEs and SDEs provide insight that can allow analysts the opportunity to develop improved theoretical and numerical approaches. This process can continue to mature the field of solution approaches to SPDEs. This was Gaines’ [26] motivation in tackling two parabolic SPDEs; the stochastic generalised Kolmogoroff-Petrovsky-Pisconouff equation (SKPP) with time varying noise and a noisy heat equation compounded by having noise that varied both with time and space.

Before tackling the SKPP equation, two simpler SPDEs were analysed for the purposes of illumination. These SPDEs were
Example 1
\[ \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} dt + \sigma u \ dW_t \]

Example 2
\[ \frac{\partial u}{\partial t} = \left( \frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial x} \right) dt + \sigma u \ dW_t \]

In both Example 1 and 2, \( x \in [0, 1], \ t \geq 0, \) the boundary conditions were Dirichlet and \( \sigma \) was a multiplicative noise magnitude parameter. In both cases and also in the case of the SKPP equation, the noise free special cases were investigated first.
7.2 Spatial Discretisation Methods

The approach to numerically solving an SPDE is to discretise the SPDE in the space dimension which generates a set of $N$ SDEs. These can then be solved as a coupled $N$ dimensional system of SDEs. Gaines [26] used finite differences to first spatially discretise the SPDEs before attempting to solve the resulting set of SDEs.

In this chapter I will present two spatial discretisation approaches to solve the $N$ dimensional system of SDEs. The first approach is the Method of Lines (Rothe’s Method) which uses finite differences for the spatial discretisation. The second method uses a Galerkin approach for the spatial discretisation. Following the spatial discretisation, the system of SDEs will be solved with strong versions of the stochastic Taylor series based time discretisation schemes.

Three systems will be investigated, increasing in complexity and culminating in the stochastic KPP equation. Where possible, the resulting equations will be compared to the equivalent PDE in the absence of noise.

7.2.1 Method of Lines Approach

This approach to solving SPDEs involved spatially discretising the SPDE using finite differences to obtain a system of SDEs that were then solved. Gaines [26] used the following discretisations

\[
\begin{align*}
    u & = u_j \\
    \frac{\partial u}{\partial x} & \approx \frac{u_{j+1} - u_j}{\Delta x} \\
    \frac{\partial^2 u}{\partial x^2} & \approx \frac{u_{j-1} - 2u_j + u_{j+1}}{\Delta x^2}.
\end{align*}
\]

Here $\Delta x$ is the constant spatial discretisation interval.

Kloeden [44] has shown that for timestep $\Delta t$ the Method of Lines approach with the Euler-Maruyama method has global space-time discretisation error

\[
E[|U_T(x_i) - Y^N_{N_T}|] \leq \kappa_{T, u_0} \left( \Delta x + \frac{\Delta t^{1/2}}{(\Delta x)^2} \right).
\]

When the linearly implicit Euler-Maruyama method is used the global space-time discretisation error is

\[
E[|U_T(x_i) - Y^N_{N_T}|] \leq \kappa_{T, u_0} \left( \Delta x + \Delta t^{1/2} \right).
\]
Davie and Gaines [15] have found a sharper error bound in the special case of additive noise.

### 7.2.2 Galerkin Approach

An alternative approach was developed and proposed by Grecksch and Kloeden [28]. This approach used a Galerkin approximation instead of finite differences to first discretise the space variable. Specifically, Grecksch and Kloeden [28] showed that the parabolic SPDE

\[ dU_t = \{ AU_t + f(U_t) \} dt + g(U_t) \, dW_t \]

can be considered as the \( N \)-dimensional system of Itô SDEs

\[ dU_t = \{ AU_t + f(U_t) \} dt + g(U_t) \, dW_t. \]

Here \( A \) is a linear operator and \( f \) and \( g \) are functions which are formed from real valued functions of a real variable with uniformly bounded derivatives of appropriate order.

The corresponding strong order \( \gamma \) Taylor series scheme with constant time step \( \Delta \) has the form

\[ Y_{k+1}^N = Y_k^N + \sum_{\alpha \in A \setminus \{ \nu \}} F_{\alpha}^N \left( Y_k^N \right) I_{\alpha, k, \Delta} \]

where \( F_{\alpha}^N \) are coefficient functions, \( I_{\alpha, k, \Delta} \) are multiple stochastic integrals and \( \gamma = \{0.5, 1, 1.5, 2, \ldots\} \). This system has global space-time discretisation error

\[ E|U_{k\Delta} - Y_k^N| \leq \kappa \left( \lambda_{N+1}^{-1/2} + \lambda_N^{[\gamma+1/2]+1} \Delta^{\gamma} \right). \]

Here \([ \cdot ]\) represents the integer part of the constant \( \kappa \) depends on the initial value, bounds on the other coefficients in the SPDE and the time interval duration.

Note that \( \lambda_j \) are the eigenvalues and \( \phi_j \) are the corresponding eigenfunctions of the operator \(-A\), i.e.,

\[ -A \phi_j = \lambda_j \phi_j \quad \text{for} \quad j = 1, 2, \ldots. \]

These eigenvalues and eigenfunctions form the basis of the Galerkin approach.

In the general case the Euler-Maruyama scheme for SPDEs was

\[ Y_{k+1}^N = Y_k^N + \left\{ A_n Y_k^N + f^N \left( Y_k^N \right) \right\} \Delta + g^N(Y_k^N) \Delta W_k. \]

Typically though, stiffness is inherent in the Itô-Galerkin SDE since \( \lambda_n \to \infty \) as \( n \to \infty \) and this can lead to numerical problems unless a small time step \( \Delta \) is used.
An obvious solution to this problem was to use an implicit scheme, however this led to the extra computational burden of solving a set of nonlinear equations at each time step. However, the stiffness was due to the linear part of the drift term, then the linearly implicit scheme approach could be applied to inhibit the growth in the eigenvalue multiplier. This led to the strong linearly implicit Euler-Maruyama scheme for SPDEs

\[ Y_{k+1}^N = (I_N - \Delta A_N)^{-1} \left( Y_k^N + f^N(Y_k^N) \Delta + g^n(Y_k^N) \Delta W_k \right) \]

where \( I_N \) denotes the \( N \times N \) identity matrix. This is a simple algebraic expression for \( Y_{k+1}^N \) and so negated the need for an equation solver. Thus, linearly implicit schemes are ideal for use with the Galerkin space discretisation approach to obtain numerical solutions to SPDEs.
7.3 Linear Example

In this section the solution to Example 1 is investigated. Recall Example 1

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} dt + \sigma u \, dW_t. \tag{7.1}
\]

7.3.1 Solution of Example 1 without Noise

Consider the noise free (\(\sigma = 0\)) analog to Example 1, equation (7.1), which reduces to

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}.
\]

This equation is defined on \(0 \leq x \leq 1\) for \(t \geq 0\) with boundary conditions

\[ u(0, t) = u(1, t) = 0 \]

and initial condition

\[ u(x, 0) = x(1 - x). \]

Using simple forward time differencing for time and space led to the computational finite difference scheme

\[ u_{i,j+1} = ru_{i-1,j} + (1 - 2r)u_{i,j} + ru_{i+1,j} \]

with

\[ r = \frac{\Delta t}{\Delta x^2}. \]

Here \(\Delta t\) was the time step and \(\Delta x\) was the spatial step, \(i = 0, 1, 2, \ldots, n\) was the spatial index, \(j = 0, 1, 2, \ldots\) was the time index, \(u_{0,j} = u_{n,j} = 0\) and \(u_{i,0} = x_i(1 - x_i)\).

The finite difference solution with \(\Delta t = 0.005\) and \(\Delta x = 0.1\) is presented in Figure 7.1 and it compares well to the analytic solution given by

\[ u(x, t) = \sum_{n=1}^{\infty} B_n \sin(n \pi x) \exp[-n^2 \pi^2 t] \]

where

\[ B_n = 2 \int_{0}^{1} x(1 - x) \sin(n \pi x) \, dx = \begin{cases} 8/(n^3 \pi^3), & n \text{ odd} \\ 0, & n \text{ even} \end{cases} \]

This analytic solution is presented in Figure 7.2. It is clear from Figure 7.2 and evident from the form of the analytic solution that \(u(x, t) \to 0\) exponentially as \(t\) increases.
Figure 7.1: Finite Difference solution to Example 1 with no noise ($\sigma = 0$).

Figure 7.2: Exact solution to Example 1 with no noise ($\sigma = 0$).
7.3.2 Method of Lines approach to Example 1

Applying the Method of Lines approach to equation (7.1) allows the noisy \((\sigma \neq 0)\) case to be examined. In this case, discretising the space interval \([x_1, x_N]\) and using reflective boundary conditions led to the system of SDEs

\[
\begin{align*}
    dX^1 &= \frac{1}{(\Delta x)^2}(X^2 - X^1) \, dt + \sigma X^1 \, dW_t, \\
    dX^j &= \frac{1}{(\Delta x)^2}(X^{j-1} - 2X^j + X^{j+1}) \, dt + \sigma X^j \, dW_t \\
    dX^N &= \frac{1}{(\Delta x)^2}(X^{N-1} - X^N) \, dt + \sigma X^N \, dW_t
\end{align*}
\]

for \(j = 2, 3, \ldots, N - 1,\)

where the \(X^i\) are the approximate solutions to \(u_t(x)\).

Given \(X = (X^1, X^2, \ldots, X^N)\) then system of SDEs of the form

\[
dX_t = \alpha(X_t) \, dt + \beta(X_t) \, dW_t
\]

have Euler-Maruyama (0.5 order strong) scheme

\[
Y^k_{n+1} = Y^k_n + \alpha^k \Delta_t + \beta^k \Delta W
\]

and Milstein (1.0 order strong) scheme

\[
Y^k_{n+1} = Y^k_n + \alpha^k \Delta_t + \beta^k \Delta W + \frac{1}{2} \sum_{l=1}^{N} b^l \frac{\partial b^k}{\partial x^l} \{\Delta W^2 - \Delta\}.
\]

As with the weak stochastic Taylor series based time discretisation schemes, \(\Delta \equiv \Delta t\).

Letting

\[
A = \frac{1}{(\Delta x)^2}
\]

then for \(n = 0, 1, 2, \ldots\) the Euler-Maruyama scheme reduced to

\[
\begin{align*}
    Y^1_{n+1} &= Y^1_n + A(Y^2_n - Y^1_n) \Delta + \sigma Y^1_n \Delta W, \\
    Y^j_{n+1} &= Y^n_j + A(Y^{j-1}_n - 2Y^j_n + Y^{j+1}_n) \Delta + \sigma Y^n_j \Delta W \\
    Y^N_{n+1} &= Y^N_n + A(Y^{N-1}_n - Y^N_n) \Delta + \sigma Y^N_n \Delta W.
\end{align*}
\]

for \(j = 2, 3, \ldots, N - 1,\)
The corresponding Milshtein scheme reduced to
\[ Y_{n+1}^l = Y_n^l + A(Y_n^2 - Y_n^1)\Delta + \sigma Y_n^1 \Delta W + \frac{1}{2}\sigma^2 Y_n^1 \{\Delta W^2 - \Delta\}, \]
\[ Y_{n+1}^j = Y_n^j + A(Y_n^{j-1} - 2Y_n^j + Y_n^{j+1})\Delta + \sigma Y_n^j \Delta W + \frac{1}{2}\sigma^2 Y_n^j \{\Delta W^2 - \Delta\} \]
for \( j = 2, 3, \ldots, N - 1, \)
\[ Y_{n+1}^N = Y_n^N + A(Y_n^{N-1} - Y_n^N)\Delta + \sigma Y_n^N \Delta W + \frac{1}{2}\sigma^2 Y_n^N \{\Delta W^2 - \Delta\}. \]

The implicit Euler-Maruyama scheme for this system was given by
\[
(1 + A\Delta) Y_{n+1}^l - A\Delta Y_{n+1}^2 = Y_n^l + \sigma Y_n^1 \Delta W, \\
-A\Delta Y_{n+1}^{j-1} + [1 + 2A\Delta] Y_{n+1}^j - A\Delta Y_{n+1}^{j+1} = Y_n^j + \sigma Y_n^j \Delta W \]
for \( j = 2, 3, \ldots, N - 1, \)
\[-A\Delta Y_{n+1}^{N-1} + [1 + A\Delta] Y_{n+1}^N = Y_n^N + \sigma Y_n^N \Delta W. \]

Finally the implicit Milshtein scheme was given by
\[
(1 + A\Delta) Y_{n+1}^l - A\Delta Y_{n+1}^2 = Y_n^l + \sigma Y_n^1 \Delta W + \frac{1}{2}\sigma^2 Y_n^N \{\Delta W^2 - \Delta\}, \\
-A\Delta Y_{n+1}^{j-1} + [1 + 2A\Delta] Y_{n+1}^j - A\Delta Y_{n+1}^{j+1} = Y_n^j + \sigma Y_n^j \Delta W + \frac{1}{2}\sigma^2 Y_n^N \{\Delta W^2 - \Delta\} \]
for \( j = 2, 3, \ldots, N - 1, \)
\[-A\Delta Y_{n+1}^{N-1} + [1 + A\Delta] Y_{n+1}^N = Y_n^N + \sigma Y_n^N \Delta W + \frac{1}{2}\sigma^2 Y_n^N \{\Delta W^2 - \Delta\}. \]

In matrix notation the implicit Euler-Maruyama scheme was written as
\[ A \cdot \mathbf{Y} = \mathbf{B} \]
where
\[
A = \begin{bmatrix}
1 + A\Delta & -A\Delta & 0 & 0 & \cdots & 0 & 0 \\
-A\Delta & 1 + 2A\Delta & -A\Delta & 0 & \cdots & 0 & 0 \\
0 & -A\Delta & 1 + 2A\Delta & -A\Delta & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \cdots & \cdots & 0 & -A\Delta & 1 + 2A\Delta & -A\Delta \\
\vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\
\vdots & \cdots & \cdots & \cdots & \cdots & \cdots & -A\Delta & 1 + A\Delta \\
\end{bmatrix}, \\
\mathbf{Y} = \begin{bmatrix} Y_{n+1}^l \ Y_{n+1}^2 \ \cdots \ Y_{n+1}^N \end{bmatrix}^T.
\]
and

\[
\mathbf{B} = \begin{bmatrix}
Y_n^1 + \sigma Y_n^1 \Delta W \\
Y_n^2 + \sigma Y_n^2 \Delta W \\
\vdots \\
Y_n^N + \sigma Y_n^N \Delta W
\end{bmatrix}
\]

The implicit Milshtein scheme differs in that it uses a different \( \mathbf{B} \) vector, i.e.

\[
\mathbf{B} = \begin{bmatrix}
Y_n^1 + \sigma Y_n^1 \Delta W + \sigma^2 Y_n^1 \{\Delta W^2 - \Delta\} / 2 \\
Y_n^2 + \sigma Y_n^2 \Delta W + \sigma^2 Y_n^2 \{\Delta W^2 - \Delta\} / 2 \\
\vdots \\
Y_n^N + \sigma Y_n^N \Delta W + \sigma^2 Y_n^N \{\Delta W^2 - \Delta\} / 2
\end{bmatrix}
\]

Using these implicit schemes usually requires the computationally intensive process of nonlinear root solving at each time step. However, as the original system was linear, the resulting set of equations formed a linear set of equations to solved. When expressed in matrix form, a tridiagonal banded matrix is formed. This band matrix system is readily solved using the Thomas Algorithm. See Smith [79] for a detailed description of the algorithm. Note that in the analysis of multidimensional systems with small spatial discretisation, the matrix \( \mathbf{A} \) can become very large. In this instance the Thomas algorithm is computationally very efficient and can solve a tridiagonal system of \( N \) equations using only \( 5N - 4 \) computations.

**7.3.3 Galerkin approach to Example 1**

Consider again the SPDE (7.1)

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} dt + \sigma u \, dW_t
\]

which is defined on \( 0 \leq x \leq 1 \) and has Dirichlet boundary conditions \( u(0, t) = u(1, t) = 0 \) for all \( t \). Using the Galerkin approach its solution \( u(x) \) could be written as

\[
u(x) = \sum_{j=1}^{\infty} u_j \phi_j(x),
\]

or truncated to

\[
u^N(x) = \sum_{j=1}^{N} u_j \phi_j(x).
\]

Here the \( u_j \) were constants,

\[
\phi_j(x) = \sqrt{2} \sin(j \pi x)
\]
were the eigenfunctions and
\[ \lambda_j = j^2 \pi^2 \]
were the eigenvalues. Obviously, as \( N \) increased, \( \lambda_j \to \infty \) and so this system of SDEs stiffened.

In this first example, the truncated Itô-Galerkin approximation decoupled componentwise and could be expressed in the form
\[ du_i^{N,t} = -\lambda_i u_i^{N,t} \, dt + \sigma u_i^{N,t} \, dW_t \quad (7.2) \]
for \( l = 1, 2, \ldots, N \). Each SDE in this system was in the form of a homogeneous linear SDE, ie.
\[ dX_t = aX_t \, dt + bX_t \, dW_t \]
where \( a \) and \( b \) are constants. Such SDEs have explicit solution
\[ X_t = X_0 + \exp \left[ \left( a - \frac{1}{2} b^2 \right) t + bW_t \right]. \]
Thus the analytic solution of equation (7.2) using the Galerkin approach is
\[ u_i^{N,t} = u_0^{N,t} \exp \left[ - \left( \lambda_i + \frac{1}{2} \sigma^2 \right) t + \sigma W_t \right]. \]
For the case when \( \sigma = 1 \) the exact solution is presented in Figure 7.3. The effect of the noise was that it led to a non-smooth and quicker decay when compared to the noise free case.

Now, the Euler-Maruyama (0.5 order strong) scheme applied to the truncated Itô-Galerkin SDE (7.1) was
\[ Y_{n+1}^l = Y_n^l - \lambda_i Y_n^l \Delta + \sigma Y_n^l \Delta W. \]
The corresponding Milshtein (1.0 order strong) scheme was
\[ Y_{n+1}^l = Y_n^l - \lambda_i Y_n^l \Delta + \sigma Y_n^l \Delta W + \frac{1}{2} \sigma^2 Y_n^l \{ \Delta W^2 - \Delta \}. \]

As the drift was linear in this example, the implicit and linearly implicit versions were identical, so for both cases the Euler-Maruyama scheme reduced to
\[ Y_{n+1}^l = \frac{Y_n^l + \sigma Y_n^l \Delta W}{1 + \lambda_i \Delta}. \]
Furthermore, the Milshtein scheme reduced to
\[ Y_{n+1}^l = \frac{Y_n^l + \sigma Y_n^l \Delta W + \sigma^2 Y_n^l \{ \Delta W^2 - \Delta \} / 2}{1 + \lambda_i \Delta}. \]
Figure 7.3: Exact solution to Example 1 with $\sigma = 1$.

For comparison purposes, the linearly implicit Maruyama scheme results for a variety of $\sigma$ values are presented. Figure 7.4 presents the Galerkin scheme results for the no noise case, $\sigma = 0$. Figure 7.5 to Figure 7.9 present the Galerkin scheme results for increasing noise intensity, specifically $\sigma = 0.1, 0.5, 1, 2$ and $\sigma = 5$ respectively. The results for $\sigma = 0$ can be directly compared to the analytic solution for the noise free case, Figure 7.2. Figure 7.7 (for $\sigma = 1$) can be compared directly to the corresponding “exact” solution in Figure 7.3. In comparing the results it is clear that the numerical Galerkin approach based approximations show very good correspondence to the exact solutions.
Figure 7.4: Galerkin approach solution to Example 1 with $\sigma = 0$.

Figure 7.5: Galerkin approach solution to Example 1 with $\sigma = 0.1$
Figure 7.6: Galerkin approach solution to Example 1 with $\sigma = 0.5$

Figure 7.7: Galerkin approach solution to Example 1 with $\sigma = 1$
Figure 7.8: Galerkin approach solution to Example 1 with $\sigma = 2$

Figure 7.9: Galerkin approach solution to Example 1 with $\sigma = 5$
7.4 Nonlinear Example

Consider now an example of a nonlinear SPDE. Recall Example 2

\[
\frac{\partial u}{\partial t} = \left( \frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial x} \right) dt + \sigma u \, dW_t. \tag{7.3}
\]

7.4.1 Solution of Example 2 without Noise

The noise free \((\sigma = 0)\) analog to Example 2, equation (7.3) simplifies to

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial x}.
\]

It is defined on \(0 \leq x \leq 1\) for \(t \geq 0\) with boundary conditions

\[
u(0, t) = u(1, t) = 0
\]

and initial condition

\[
u(x, 0) = x(1 - x).
\]

Using simple forward time differencing for time and space led to the computational finite difference scheme

\[
u_{i,j+1} = r
\]

with

\[
\Delta t = \frac{\partial u}{\partial x^2}.
\]

Here \(\Delta t\) was the time step and \(\Delta x\) was the spatial step, \(i = 0, 1, 2, \ldots, n\) was the spatial index, \(j = 0, 1, 2, \ldots\) was the time index, \(u_{0,j} = u_{n,j} = 0\) and \(u_{i,0} = x_i(1 - x_i)\).

The finite difference solution with \(\Delta t = 0.005\) and \(\Delta x = 0.1\) is presented in Figure 7.10. The solution exhibited similar features to those in the solution of Example 1 in the noise free case, i.e. a steady exponential decay to zero. This similarity of solutions is to be expected by comparing the finite difference schemes. The scheme for this example differed to that in Example 1 by the term

\[
r(u_{i,j} u_{i+1,j} - u_{i,j}^2) \Delta x.
\]

As \(u_{i+1,j} \approx u_{i,j}\) for small \(\Delta x\) then the magnitude of this term was close to zero.
Figure 7.10: Finite Difference solution to Example 2 with no noise \((\sigma = 0)\).
7.4.2 Method of Lines approach to Example 2

Applying the Method of Lines approach to equation (7.3) allows the noisy ($\sigma \neq 0$) case to be examined. In this case discretising the space interval $[x_1, x_N]$ and using reflective boundary conditions led to the system of SDEs

$$
\begin{align*}
    dX^1 &= \left[ \frac{1}{(\Delta x)^2}(X^2 - X^1) + \frac{1}{\Delta x} \left( X^1 X^2 - (X^1)^2 \right) \right] dt + \sigma X^1 \, dW_t, \\
    dX^j &= \left[ \frac{1}{(\Delta x)^2}(X^{j-1} - 2X^j + X^{j+1}) + \frac{1}{\Delta x} \left( X^1 X^2 - (X^1)^2 \right) \right] dt + \sigma X^j \, dW_t \\
    & \quad \text{for} \quad j = 2, 3, \ldots, N - 1, \\
    dX^N &= \left[ \frac{1}{(\Delta x)^2}(X^{N-1} - X^N) + \frac{1}{\Delta x} \left( X^1 X^2 - (X^1)^2 \right) \right] dt + \sigma X^N \, dW_t
\end{align*}
$$

where the $X^i$ are the approximate solutions to $u_t(x)$.

Letting 

$$
A = \frac{1}{(\Delta x)^2} \quad \text{and} \quad B = \frac{1}{\Delta x}
$$

then for $n = 0, 1, 2, \ldots$ the Euler-Maruyama scheme reduced to

$$
\begin{align*}
    Y^1_{n+1} &= Y^1_n + A(Y^2_n - Y^1_n)\Delta + B \left( Y^1_n Y^2_n - (Y^1_n)^2 \right) \Delta + \sigma Y^1_n \Delta W, \\
    Y^j_{n+1} &= Y^j_n + A(Y^{j-1}_n - 2Y^j_n + Y^{j+1}_n)\Delta + B \left( Y^j_n Y^{j+1}_n - (Y^j_n)^2 \right) \Delta + \sigma Y^j_n \Delta W \\
    & \quad \text{for} \quad j = 2, 3, \ldots, N - 1, \\
    Y^N_{n+1} &= Y^N_n + A(Y^{N-1}_n - Y^N_n)\Delta + B \left( Y^N_n Y^{N-1}_n - (Y^N_n)^2 \right) \Delta + \sigma Y^N_n \Delta W.
\end{align*}
$$

The corresponding Milstein scheme reduced to

$$
\begin{align*}
    Y^1_{n+1} &= Y^1_n + A(Y^2_n - Y^1_n)\Delta + B \left( Y^1_n Y^2_n - (Y^1_n)^2 \right) \Delta + \sigma Y^1_n \Delta W + \frac{1}{2} \sigma^2 Y^1_n \{\Delta W^2 - \Delta\}, \\
    Y^j_{n+1} &= Y^j_n + A(Y^{j-1}_n - 2Y^j_n + Y^{j+1}_n)\Delta + B \left( Y^j_n Y^{j+1}_n - (Y^j_n)^2 \right) \Delta + \sigma Y^j_n \Delta W + \frac{1}{2} \sigma^2 Y^j_n \{\Delta W^2 - \Delta\} \\
    & \quad \text{for} \quad j = 2, 3, \ldots, N - 1, \\
    Y^N_{n+1} &= Y^N_n + A(Y^{N-1}_n - Y^N_n)\Delta + B \left( Y^N_n Y^{N-1}_n - (Y^N_n)^2 \right) \Delta + \sigma Y^N_n \Delta W + \frac{1}{2} \sigma^2 Y^N_n \{\Delta W^2 - \Delta\}.
\end{align*}
$$

The implicit Euler-Maruyama scheme for this system was given by

$$
\begin{align*}
    (1 + A\Delta)Y^1_{n+1} - A \Delta Y^2_{n+1} + B \Delta Y^1_{n+1}(Y^1_{n+1} - Y^2_{n+1}) &= Y^1_n + \sigma Y^1_n \Delta W, \\
    -A \Delta Y^j_{n+1} + [1 + 2A\Delta]Y^j_{n+1} - A \Delta Y^{j+1}_{n+1} + B \Delta Y^j_{n+1}(Y^j_{n+1} - Y^{j+1}_{n+1}) &= Y^j_n + \sigma Y^j_n \Delta W \\
    & \quad \text{for} \quad j = 2, 3, \ldots, N - 1, \\
    -A \Delta Y^{N-1}_{n+1} + [1 + A\Delta]Y^{N-1}_{n+1} + B \Delta Y^{N-1}_{n+1}(Y^N_{n+1} - Y^{N-1}_{n+1}) &= Y^N_n + \sigma Y^N_n \Delta W.
\end{align*}
$$
Finally, the implicit Milshtein scheme was

\[
(1 + A\Delta)Y_{n+1}^1 - A\Delta Y_{n+1}^{2} + B\Delta Y_{n+1}^1(Y_{n+1}^1 - Y_{n+1}^{2}) = Y_{n}^1 + \sigma Y_{n}^1 \Delta W \\
+ \frac{1}{2} \sigma^2 Y_{n}^N \{ \Delta W^2 - \Delta \},
\]

\[
-A\Delta Y_{n+1}^{j-1} + [1 + 2A\Delta]Y_{n+1}^{j} - A\Delta Y_{n+1}^{j+1} + B\Delta Y_{n+1}^j(Y_{n+1}^j - Y_{n+1}^{j+1}) = Y_{n}^j + \sigma Y_{n}^j \Delta W \\
+ \frac{1}{2} \sigma^2 Y_{n}^N \{ \Delta W^2 - \Delta \}
\]

for \( j = 2, 3, \ldots, N - 1, \)

\[
-A\Delta Y_{n+1}^{N-1} + [1 + A\Delta]Y_{n+1}^{N} + B\Delta Y_{n+1}^N(Y_{n+1}^N - Y_{n+1}^{N-1}) = Y_{n}^N + \sigma Y_{n}^N \Delta W \\
+ \frac{1}{2} \sigma^2 Y_{n}^N \{ \Delta W^2 - \Delta \}.
\]

However, the use of these implicit schemes required the computationally intensive process of nonlinear root solving at each time step.

Another option was to use a semi-implicit method, as described by Talay [86], or the strong version of the linearly implicit scheme, see Kloeden and Shott [47]. Recall that the linearly implicit schemes are obtained from the explicit scheme by making only the linear part of the drift term implicit. Thus in this case, the linearly implicit Euler-Maruyama scheme was

\[
[1 + A\Delta]Y_{n+1}^1 + [-A\Delta]Y_{n+1}^{2} = Y_{n}^1 + BY_{n}^1(Y_{n}^2 - Y_{n}^{1})\Delta + \sigma Y_{n}^1 \Delta W,
\]

\[
\{-A\Delta\}Y_{n+1}^{j-1} + [1 + 2A\Delta]Y_{n+1}^{j} + [-A\Delta]Y_{n+1}^{j+1} = Y_{n}^j + BY_{n}^j(Y_{n}^{j+1} - Y_{n}^{j})\Delta + \sigma Y_{n}^j \Delta W
\]

for \( j = 2, 3, \ldots, N - 1, \)

\[
\{-A\Delta\}Y_{n+1}^{N-1} + [1 + A\Delta]Y_{n+1}^{N} = Y_{n}^N + BY_{n}^N(Y_{n}^{N-1} - Y_{n}^{N})\Delta + \sigma Y_{n}^N \Delta W.
\]

The linearly implicit Milshtein scheme was

\[
[1 + A\Delta]Y_{n+1}^1 + [-A\Delta]Y_{n+1}^{2} = Y_{n}^1 + BY_{n}^1(Y_{n}^2 - Y_{n}^{1})\Delta + \sigma Y_{n}^1 \Delta W \\
+ \frac{1}{2} \sigma^2 Y_{n}^N \{ \Delta W^2 - \Delta \},
\]

\[
\{-A\Delta\}Y_{n+1}^{j-1} + [1 + 2A\Delta]Y_{n+1}^{j} + [-A\Delta]Y_{n+1}^{j+1} = Y_{n}^j + BY_{n}^j(Y_{n}^{j+1} - Y_{n}^{j})\Delta + \sigma Y_{n}^j \Delta W \\
+ \frac{1}{2} \sigma^2 Y_{n}^N \{ \Delta W^2 - \Delta \}
\]

for \( j = 2, 3, \ldots, N - 1, \)

\[
\{-A\Delta\}Y_{n+1}^{N-1} + [1 + A\Delta]Y_{n+1}^{N} = Y_{n}^N + BY_{n}^N(Y_{n}^{N-1} - Y_{n}^{N})\Delta + \sigma Y_{n}^N \Delta W \\
+ \frac{1}{2} \sigma^2 Y_{n}^N \{ \Delta W^2 - \Delta \}.
\]
In matrix form, the linearly implicit Euler-Maruyama scheme was written as

\[ \mathbf{A} \cdot \mathbf{Y} = \mathbf{B} \]

where

\[
\mathbf{A} = \begin{bmatrix}
1 + A\Delta & -A\Delta & 0 & 0 & \cdots & 0 & 0 \\
-A\Delta & 1 + 2A\Delta & -A\Delta & 0 & \cdots & 0 \\
0 & -A\Delta & 1 + 2A\Delta & -A\Delta & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & -A\Delta & 1 + A\Delta \\
\end{bmatrix},
\]

\[
\mathbf{Y} = \begin{bmatrix} Y_{n+1}^1, Y_{n+1}^2, \ldots, Y_{n+1}^N \end{bmatrix}^T
\]

and

\[
\mathbf{B} = \begin{bmatrix}
Y_n^1 + BY_n^1(Y_n^2 - Y_n^1)\Delta + \sigma Y_n^1 \Delta W \\
Y_n^2 + BY_n^2(Y_n^3 - Y_n^2)\Delta + \sigma Y_n^2 \Delta W \\
\vdots \\
Y_n^N + BY_n^N(Y_n^{N-1} - Y_n^N)\Delta + \sigma Y_n^N \Delta W
\end{bmatrix}.
\]

The implicit Milstein scheme differs in that it uses a different \( \mathbf{B} \) vector, ie.

\[
\mathbf{B} = \begin{bmatrix}
Y_n^1 + BY_n^1(Y_n^2 - Y_n^1)\Delta + \sigma Y_n^1 \Delta W + \sigma^2 Y_n^1 \{\Delta W^2 - \Delta\} / 2 \\
Y_n^2 + BY_n^2(Y_n^3 - Y_n^2)\Delta + \sigma Y_n^2 \Delta W + \sigma^2 Y_n^2 \{\Delta W^2 - \Delta\} / 2 \\
\vdots \\
Y_n^N + BY_n^N(Y_n^{N-1} - Y_n^N)\Delta + \sigma Y_n^N \Delta W + \sigma^2 Y_n^N \{\Delta W^2 - \Delta\} / 2
\end{bmatrix}.
\]

The advantage of the linearly implicit scheme was that the matrix \( \mathbf{A} \) was tridiagonal and so the system could be readily solved. As in Example 1, my numerical solutions used the computationally efficient Thomas Algorithm. This algorithm was coded directly into my Fortran code.

### 7.4.3 Galerkin approach to Example 2

Consider again the SPDE (7.3)

\[
\frac{\partial u}{\partial t} = \left( \frac{\partial^2 u}{\partial x^2} + u \frac{\partial u}{\partial x} \right) dt + \sigma u \, dW_t
\]
which is defined on $0 \leq x \leq 1$ and has Dirichlet boundary conditions $(u(0, t) = u(1, t) = 0$ for all $t$. Using the Galerkin approach its solution $u(x)$ could be written as

$$u(x) = \sum_{j=1}^{\infty} u_j \phi_j(x),$$

Here the $u_j$ were constants,

$$\phi_j(x) = \sqrt{2} \sin(j \pi x)$$

were the eigenfunctions and

$$\lambda_j = j^2 \pi^2$$

were the corresponding eigenvalues.

This is a more complex case than that in Example 1 since the $u_{\frac{\partial u}{\partial x}}$ term introduces quadratic nonlinearities.

Now

$$\frac{\partial u}{\partial x} = \sum_{j=1}^{\infty} u_j \frac{\partial \phi_j}{\partial x}$$

and

$$u \frac{\partial u}{\partial x} = \sum_{i,j=1}^{\infty} u_i u_j \phi_i(x) \phi_j(x).$$

As

$$\frac{\partial \phi_j}{\partial x} = \sqrt{2} j \pi \cos(j \pi x)$$

then

$$u \frac{\partial u}{\partial x} = \sum_{i,j=1}^{\infty} 2 j \pi u_i u_j \sin(i \pi x) \cos(j \pi x).$$

Using the notation

$$(f, g) = \int_0^1 f(x) g(x) \, dx$$

then

$$\left( u \frac{\partial u}{\partial x}, \phi_l(x) \right) = \int_0^1 \left( \sum_{i,j=1}^{\infty} 2 j \pi u_i u_j \sin(i \pi x) \cos(j \pi x) \right) \sqrt{2} \sin(l \pi x) \, dx$$

$$= \sum_{i,j=1}^{\infty} 2 \sqrt{2} j \pi u_i u_j \int_0^1 \sin(i \pi x) \cos(j \pi x) \sin(l \pi x) \, dx$$

$$\equiv \sum_{i,j=1}^{\infty} \alpha_{ij} u_i u_j \, dx$$

where

$$\alpha_{ij} = 2 \sqrt{2} j \pi \int_0^1 \sin(i \pi x) \cos(j \pi x) \sin(l \pi x) \, dx.$$
Now, using standard trigonometric identities, the following simplifications

\[
\{ \sin[(i + j)\pi x] + \sin[(i - j)\pi x] \} \sin(l\pi x) = \{ \sin(i\pi x) \cos(j\pi x) + \cos(i\pi x) \sin(j\pi x) + \\
\sin(i\pi x) \cos(j\pi x) - \cos(i\pi x) \sin(j\pi x) \} \sin(l\pi x)
\]

\[
= 2 \sin(i\pi x) \cos(j\pi x) \sin(l\pi x)
\]

and

\[
\{ \sin[(i + j)\pi x] + \sin[(i - j)\pi x] \} \sin(l\pi x) = \sin \left[ (i + j)\pi x \right] \sin \left( l\pi x \right) + \sin \left[ (i - j)\pi x \right] \sin \left( l\pi x \right)
\]

\[
= \frac{1}{2} \{ \cos[(i + j - l)\pi x] - \cos[(i + j + l)\pi x] + \\
\cos[(i - j - l)\pi x] - \cos[(i - j + l)\pi x] \}
\]

lead to

\[
\alpha_{ijl} = \frac{\sqrt{2}}{2} \pi \int_0^1 \{ \cos[(i + j - l)\pi x] - \cos[(i + j + l)\pi x] + \cos[(i - j - l)\pi x] - \cos[(i - j + l)\pi x] \} \, dx.
\]

Since

\[
\int_0^1 \cos(k\pi x) \, dx = \left[ \frac{1}{k\pi} \sin k\pi x \right]_0^1 = \frac{1}{k\pi} \sin(k\pi) = \begin{cases} 1, & k = 0 \\ 0, & k \neq 0 \end{cases} = \delta_{k,0}
\]

where \( \delta \) represents the Kronecker Delta, then \( \alpha_{ijl} \) simplifies to

\[
\alpha_{ijl} = \frac{\sqrt{2}}{2} \pi \{ \delta_{i+j-l,0} - \delta_{i+j+l,0} + \delta_{i-j-l,0} - \delta_{i-j+l,0} \}
\]

or

\[
\alpha_{ijl} = \frac{\sqrt{2}}{2} \pi \{ \delta_{i+j,l} + \delta_{i-j,l} - \delta_{i+j,-l} - \delta_{i-j,-l} \}.
\]

However, \( \delta_{i+j,-l} \equiv 0 \) so

\[
\alpha_{ijl} = \frac{\sqrt{2}}{2} \pi \{ \delta_{i+j,l} + \delta_{i-j,l} - \delta_{i-j,-l} \}.
\]

Importantly, this expression is in terms of the Kronecker delta which emphasises which terms are retained.

Thus the Galerkin Scheme for the SPDE \((7.3)\) for \( l = 1, 2, \ldots N \) was

\[
dU_t^{N,l} = \left\{ -\lambda_l U_t^{N,l} + \sum_{i,j=1}^{N} \alpha_{ijl} U_t^{N,i} \phi_j(t) \right\} \, dt + \sigma U_t^{N,l} \, dW_t
\]

where

\[
U^N(t, x) = \sum_{j=1}^{N} U^{N,j}(t) \phi_j(x).
\]
For $l = 1, 2, \ldots, N$ the corresponding Euler-Maruyama (0.5 order strong) scheme was

\[ Y_{n+1}^l = Y_n^l + \left\{ -\lambda_i Y_n^l + \sum_{i,j=1}^{N} \alpha_{ij} Y_n^{i,j} \right\} \Delta + \sigma Y_n^l \Delta W. \]

Furthermore, the corresponding linearly implicit Euler-Maruyama scheme was

\[ Y_{n+1}^l = \frac{Y_n^l + \sum_{i,j=1}^{N} \alpha_{ij} Y_n^{i,j} \Delta + \sigma Y_n^l \Delta W}{1 + \lambda_i \Delta}. \]

Of greater accuracy is the Milstein (1.0 order strong) scheme where

\[ Y_{n+1}^l = Y_n^l + \left\{ -\lambda_i Y_n^l + \sum_{i,j=1}^{N} \alpha_{ij} Y_n^{i,j} \right\} \Delta + \sigma Y_n^l \Delta W + \frac{1}{2} \sigma^2 Y_n^l (\Delta W^2 - \Delta) \]

with corresponding linearly implicit variant

\[ Y_{n+1}^l = \frac{Y_n^l + \sum_{i,j=1}^{N} \alpha_{ij} Y_n^{i,j} \Delta + \sigma Y_n^l \Delta W + \frac{1}{2} \sigma^2 Y_n^l (\Delta W^2 - \Delta)}{1 + \lambda_i \Delta}. \]

Again, for comparison purposes, the linearly implicit Maruyama scheme results for a variety of $\sigma$ values are presented. Figure 7.11 presents the Galerkin scheme results for the noise free case, $\sigma = 0$. Figure 7.12 to Figure 7.18 present the Galerkin scheme results for increasing noise intensity, specifically $\sigma = 0.1, 0.2, 0.5, 1, 2, 5$ and $\sigma = 10$ respectively. Again, as in Example 1, the effect of the increase in noise intensity leads to a quicker (but more varying) decay to zero. The results for $\sigma = 0$ can be directly compared to the finite difference solution for the noise free case, Figure 7.10.
Figure 7.11: Galerkin approach solution to Example 2 with $\sigma = 0$.

Figure 7.12: Galerkin approach solution to Example 2 with $\sigma = 0.1$
Figure 7.13: Galerkin approach solution to Example 2 with $\sigma = 0.2$

Figure 7.14: Galerkin approach solution to Example 2 with $\sigma = 0.5$
Figure 7.15: Galerkin approach solution to Example 2 with $\sigma = 1$.

Figure 7.16: Galerkin approach solution to Example 2 with $\sigma = 2$. 
Figure 7.17: Galerkin approach solution to Example 2 with $\sigma = 5$. 

Figure 7.18: Galerkin approach solution to Example 2 with $\sigma = 10$. 
7.5 The Stochastic KPP Equation

The example presented by Gaines, and expanded upon by Elworthy et al [21, 20], is the stochastic variant of the Kolmogoroff-Petrovsky-Piscounoff (KPP) semilinear reaction-diffusion equation

$$\frac{\partial u}{\partial t} = \frac{1}{2} \frac{\partial^2 u}{\partial x^2} + cu(1 - u).$$

In its original form the KPP equation

$$u_t = u_{xx} + u - u^2, \quad u(0, x) = u_0(x)$$

arose as a model for the spread of a gene through a geographically spread population. It was one of the first equations that was found to have a travelling wave solution, i.e., there exists a function $f$ and constant $c$ such that $u(t, x) = f(x - ct)$ is a solution. This equation, and its travelling wave solution, were initially described in 1937 by Kolmogoroff, Petrovsky and Piscounoff [50] and independently by Fisher [24].

The stochastic KPP equation (SKPP) presented in Gaines [26] is

$$d u_t = \left[ \frac{1}{2} \mu^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{\mu^2} (1 - u_t) u_t \right] dt + F_{\mu} u_t \, dW_t$$

(7.4)

where $u_t = u_t(x)$, $u_0 = \phi(x)$ is the initial condition and the choice of $\mu$, which is typically small, leads to the case of

$$F_{\mu} = \begin{cases} 
  k, & \text{"weak noise"} \\
  k/\mu, & \text{"mild noise"} \\
  k/\mu^2, & \text{"strong noise"}
\end{cases}$$

7.5.1 Solution of Stochastic KPP equation without Noise

Consider the noise free ($k = 0$) variant of equation (7.4)

$$\frac{\partial u}{\partial t} = \frac{1}{2} \mu^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{\mu^2} (1 - u) u.$$

This PDE is defined on $0 \leq x \leq 1$ for $t \geq 0$ with boundary conditions

$$u(0, t) = u(1, t) = 0$$

and initial condition

$$u(x, 0) = x(1 - x).$$
Using simple forward time differencing for time and space led to the computational
finite difference scheme
\[ u_{i,j+1} = \frac{1}{2} \mu^2 r (u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \frac{1}{\mu^2} u_{i,j} (1 - u_{i,j}) \Delta t + u_{i,j} \]
with
\[ r = \frac{\Delta t}{\Delta x^2}. \]
Here \( \Delta t \) was the time step and \( \Delta x \) was the spatial step, \( i = 0, 1, 2, \ldots, n \) was the spatial
index, \( j = 0, 1, 2, \ldots \) was the time index, \( u_{0,j} = u_{n,j} = 0 \) and \( u_{i,0} = x_i (1 - x_i) \).

This finite difference solution for \( \mu = 0.2, 0.5 \) and 1.0 is presented in Figures 7.19,
7.20 and 7.21 respectively. In the first two cases the solution increased in magnitude from
the initial distribution values to 1.0 as \( t \) increased. In the third case it decayed to zero.
Overall, as \( \mu \) increased the transition at the boundaries smoothened. Note though, for
\( \Delta x = 0.1 \) and \( \Delta t = 0.05 \) the finite difference solution became unstable for \( \mu > 1.4 \).

Figure 7.19: Finite Differences solution to KPP equation with \( \mu = 0.2 \) and 0.5
Figure 7.20: Finite Differences solution to KPP equation with $\mu = 0.8$ and $1.0$

Figure 7.21: Finite Differences solution to KPP equation with $\mu = 1.4$
7.5.2 Method of Lines approach to the SKPP Equation

Applying the Method of Lines approach to equation (7.4) allows the noisy \((k \neq 0)\) case to be examined. In this mild noise case, discretising the space interval \([x_1, x_N]\) and using reflective boundary conditions led to the system of SDEs

\[
dX^1 = \left[ \frac{\mu^2}{2(\Delta x)^2} (X^1 - X^0) + \frac{1}{\mu^2} (1 - X^1) X^1 \right] dt + \frac{k}{\mu} X^1 dW_t,
\]

\[
dX^j = \left[ \frac{\mu^2}{2(\Delta x)^2} (X^{j-1} - 2X^j + X^{j+1}) + \frac{1}{\mu^2} (1 - X^j) X^j \right] dt + \frac{k}{\mu} X^j dW_t
\]

for \(j = 2, 3, \ldots, N - 1,\)

\[
dX^N = \left[ \frac{\mu^2}{2(\Delta x)^2} (X^{N-1} - X^N) + \frac{1}{\mu^2} (1 - X^N) X^N \right] dt + \frac{k}{\mu} X^N dW_t
\]

where the \(X^i\) are the approximate solutions to \(u_t(x)\).

Letting

\[
A = \frac{\mu^2}{2(\Delta x)^2}, \quad B = \frac{1}{\mu^2} \quad \text{and} \quad C = \frac{k}{\mu},
\]

then for \(n = 0, 1, 2, \ldots\) the Euler-Maruyama scheme reduced to

\[
Y^1_{n+1} = Y^1_n + \left[ A(Y^2_n - Y^1_n) + B(1 - Y^1_n) Y^1_n \right] \Delta + CY^1_n \Delta W,
\]

\[
Y^j_{n+1} = Y^j_n + \left[ A(Y^{j-1}_n - 2Y^j_n + Y^{j+1}_n) + B(1 - Y^j_n) Y^j_n \right] \Delta + CY^j_n \Delta W
\]

for \(j = 2, 3, \ldots, N - 1,\)

\[
Y^N_{n+1} = Y^N_n + \left[ A(Y^{N-1}_n - Y^N_n) + B(1 - Y^N_n) Y^N_n \right] \Delta + CY^N_n \Delta W.
\]

The corresponding Milstein scheme reduced to

\[
Y^1_{n+1} = Y^1_n + \left[ A(Y^2_n - Y^1_n) + B(1 - Y^1_n) Y^1_n \right] \Delta + CY^1_n \Delta W + \frac{1}{2} C^2 Y^1_n \{ \Delta W^2 - \Delta \},
\]

\[
Y^j_{n+1} = Y^j_n + \left[ A(Y^{j-1}_n - 2Y^j_n + Y^{j+1}_n) + B(1 - Y^j_n) Y^j_n \right] \Delta + CY^j_n \Delta W + \frac{1}{2} C^2 Y^j_n \{ \Delta W^2 - \Delta \}
\]

for \(j = 2, 3, \ldots, N - 1,\)

\[
Y^N_{n+1} = Y^N_n + \left[ A(Y^{N-1}_n - Y^N_n) + B(1 - Y^N_n) Y^N_n \right] \Delta + CY^N_n \Delta W + \frac{1}{2} C^2 Y^N_n \{ \Delta W^2 - \Delta \}.
\]

As with numerical solution of parabolic PDEs, there were stability considerations that needed to be followed. In this case, Gaines indicated that the use of an explicit time discretisation scheme required the choice of computational time step \(\Delta\) dependent on the
space step $\Delta x = (x_N - x_1)/N$ such that

$$\Delta < \frac{1}{2\mu^2(\Delta x)^2}.$$

The implicit Euler-Maruyama scheme for this system was given by

$$B\Delta (Y_{n+1}^1)^2 + [1 + (A - B)\Delta]Y_{n+1}^1 - A\Delta Y_{n+1}^2 = Y_n^1 + CY_n^1\Delta W,$$

$$-A\Delta Y_{n+1}^j + B\Delta (Y_{n+1}^j)^2 + [1 + (2A - B)\Delta]Y_{n+1}^j - A\Delta Y_{n+1}^{j+1} = Y_n^j + CY_n^j\Delta W$$

for $j = 2, 3, \ldots, N - 1,$

$$-A\Delta Y_{n+1}^{N-1} + B\Delta (Y_{n+1}^{N-1})^2 + [1 + (A - B)\Delta]Y_{n+1}^{N-1} = Y_n^{N-1} + CY_n^{N-1}\Delta W$$

and the implicit Milstein scheme was given by

$$B\Delta (Y_{n+1}^1)^2 + [1 + (A - B)\Delta]Y_{n+1}^1 - A\Delta Y_{n+1}^2 = Y_n^1 + CY_n^1\Delta W$$

$$+\frac{1}{2}C^2Y_n^1 \{\Delta W^2 - \Delta\},$$

$$-A\Delta Y_{n+1}^j + B\Delta (Y_{n+1}^j)^2 + [1 + (2A - B)\Delta]Y_{n+1}^j - A\Delta Y_{n+1}^{j+1} = Y_n^j + CY_n^j\Delta W$$

$$+\frac{1}{2}C^2Y_n^j \{\Delta W^2 - \Delta\}$$

for $j = 2, 3, \ldots, N - 1,$

$$-A\Delta Y_{n+1}^{N-1} + B\Delta (Y_{n+1}^{N-1})^2 + [1 + (A - B)\Delta]Y_{n+1}^{N-1} = Y_n^{N-1} + CY_n^{N-1}\Delta W$$

$$+\frac{1}{2}C^2Y_n^{N-1} \{\Delta W^2 - \Delta\}. $$

The use of these implicit schemes required the computationally intensive process of non-linear root solving at each time step. Again, the linearly implicit scheme could be used, thus in this case the linearly implicit Euler-Maruyama scheme was

$$[1 + (A - B)\Delta]Y_{n+1}^1 + [-A\Delta]Y_{n+1}^2 = Y_n^1 - B(Y_n^1)^2\Delta + CY_n^1\Delta W,$$

$$\{ -A\Delta \}Y_{n+1}^j + [1 + (2A - B)\Delta]Y_{n+1}^j + [-A\Delta]Y_{n+1}^{j+1} = Y_n^j - B(Y_n^j)^2\Delta + CY_n^j\Delta W$$

for $j = 2, 3, \ldots, N - 1,$

$$\{ -A\Delta \}Y_{n+1}^{N-1} + [1 + (A - B)\Delta]Y_{n+1}^{N-1} = Y_n^{N-1} - B(Y_n^{N-1})^2\Delta + CY_n^{N-1}\Delta W$$

and the linearly implicit Milstein scheme

$$[1 + (A - B)\Delta]Y_{n+1}^1 + [-A\Delta]Y_{n+1}^2 = Y_n^1 - B(Y_n^1)^2\Delta + CY_n^1\Delta W$$

$$+\frac{1}{2}C^2Y_n^1 \{\Delta W^2 - \Delta\},$$

$$\{ -A\Delta \}Y_{n+1}^j + [1 + (2A - B)\Delta]Y_{n+1}^j + [-A\Delta]Y_{n+1}^{j+1} = Y_n^j - B(Y_n^j)^2\Delta + CY_n^j\Delta W$$

$$+\frac{1}{2}C^2Y_n^j \{\Delta W^2 - \Delta\}.$$
\[ (-A \Delta) Y_{n+1}^{N-1} + [1 + (A - B) \Delta] Y_n^N = Y_n^N - B(Y_n^N)^2 \Delta + C Y_n^N \Delta W + \frac{1}{2} C^2 Y_n^N \{ \Delta W^2 - \Delta \}. \]

In matrix form, the linearly implicit Euler-Maruyama scheme was written as

\[ \mathbf{A} \cdot \mathbf{Y} = \mathbf{B} \]

where \( \mathbf{A} = \)

\[
\begin{bmatrix}
1 + (A - B) \Delta & -A \Delta & 0 & 0 & \cdots & 0 & 0 \\
-A \Delta & 1 + (2A - B) \Delta & -A \Delta & 0 & \cdots & 0 & 0 \\
0 & -A \Delta & 1 + (2A - B) \Delta & -A \Delta & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 & -A \Delta & 1 + (2A - B) \Delta & -A \Delta \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & -A \Delta & 1 + (A - B) \Delta
\end{bmatrix},
\]

\[ \mathbf{Y} = \left[ Y_{n+1}^1 Y_{n+1}^2 \ldots Y_{n+1}^N \right]^T \]

and

\[ \mathbf{B} = \left[ \begin{array}{c}
Y_n^1 - B(Y_n^1)^2 \Delta + CY_n^1 \Delta W \\
Y_n^2 - B(Y_n^2)^2 \Delta + CY_n^2 \Delta W \\
\vdots \\
Y_n^N - B(Y_n^N)^2 \Delta + CY_n^N \Delta W
\end{array} \right]. \]

The linearly implicit Milstein scheme differs in that it uses a different \( \mathbf{B} \) vector, ie.

\[ \mathbf{B} = \left[ \begin{array}{c}
\frac{Y_n^1 - B(Y_n^1)^2 \Delta + CY_n^1 \Delta W + C^2 Y_n^1 \{ \Delta W^2 - \Delta \}}{2} \\
\frac{Y_n^2 - B(Y_n^2)^2 \Delta + CY_n^2 \Delta W + C^2 Y_n^2 \{ \Delta W^2 - \Delta \}}{2} \\
\vdots \\
\frac{Y_n^N - B(Y_n^N)^2 \Delta + CY_n^N \Delta W + C^2 Y_n^N \{ \Delta W^2 - \Delta \}}{2}
\end{array} \right]. \]

The advantage of the linearly implicit scheme was that the matrix \( \mathbf{A} \) was tridiagonal and so the system could be solved by inverting the matrix \( \mathbf{A} \). In my numerical solutions I once again applied the Thomas Algorithm, coded directly into my Fortran code.
7.5.3 Galerkin approach to the Stochastic KPP equation

Consider again the SPDE (7.4), the stochastic KPP equation

\[ du = \left[ \frac{1}{2} \mu^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{\mu^2} (1 - u)u \right] dt + F_\mu u \, dW_t, \]

Applying the Galerkin approach with eigenvalues and eigenfunctions in Example 1 and 2 led to the system of SDEs

\[ dU_{l}^{N,t} = \left[ -\frac{1}{2} \mu^2 \lambda_i U_{l}^{N,t} + \frac{1}{\mu^2} U_{l}^{N,t} - \frac{1}{\mu^2} \sum_{i,j=1}^{N} \alpha_{ij} U_{l}^{N,i} U_{l}^{N,j} \right] dt + F_\mu U_{l}^{N,t} \, dW_t \]

for \( l = 1, 2, \ldots, N \) where

\[ U_i^N = \sum_{l=1}^{N} U_{l}^{N,i} \phi_l(x). \]

Then, with

\[ \alpha_{ij,l} = \frac{\sqrt{2}}{\pi} \left\{ \frac{\delta_{l+i-j, \text{odd}}}{l+i-j} + \frac{\delta_{l-i+j, \text{odd}}}{l-i+j} - \frac{\delta_{l+i+j, \text{odd}}}{l+i+j} - \frac{\delta_{l-i-j, \text{odd}}}{l-i-j} \right\} \]

where

\[ \delta_{k, \text{odd}} = \begin{cases} 1, & \text{k is odd} \\ 0, & \text{k is even} \end{cases} \]

the Euler-Maruyama (0.5 order strong) approximation for \( l = 1, 2, \ldots, N \) was

\[ Y_{n+1}^l = Y_n^l + \left[ -\frac{1}{2} \mu^2 \lambda_i Y_n^l + \frac{1}{\mu^2} Y_n^l - \frac{1}{\mu^2} \sum_{i,j=1}^{N} \alpha_{ij} Y_n^i Y_n^j \right] \Delta + F_\mu Y_n^l \Delta W. \]

The corresponding linearly implicit scheme was

\[ Y_{n+1}^l = \frac{Y_n^l - \frac{1}{\mu^2} \sum_{i,j=1}^{N} \alpha_{ij} Y_n^i Y_n^j \Delta + F_\mu Y_n^l \Delta W}{1 + \left( \frac{1}{2} \mu^2 \lambda_i - \frac{1}{\mu^2} \right) \Delta}. \]

Furthermore, the explicit Milstein (1.0 order strong) scheme was

\[ Y_{n+1}^l = Y_n^l + \left[ -\frac{1}{2} \mu^2 \lambda_i Y_n^l + \frac{1}{\mu^2} Y_n^l - \frac{1}{\mu^2} \sum_{i,j=1}^{N} \alpha_{ij} Y_n^i Y_n^j \right] \Delta + F_\mu Y_n^l \Delta W + \frac{1}{2} F_\mu^2 Y_n^l (\Delta W^2 - \Delta). \]

Finally, the corresponding linearly implicit version was

\[ Y_{n+1}^l = \frac{Y_n^l - \frac{1}{\mu^2} \sum_{i,j=1}^{N} \alpha_{ij} Y_n^i Y_n^j \Delta + F_\mu Y_n^l \Delta W + \frac{1}{2} F_\mu^2 Y_n^l (\Delta W^2 - \Delta)}{1 + \left( \frac{1}{2} \mu^2 \lambda_i - \frac{1}{\mu^2} \right) \Delta}. \]
Again, for comparison purposes, the linearly implicit Euler-Maruyama scheme results for a variety of parameter cases are presented. Each figure also provides a comparison to the solution obtained using Gaines’ Method of Lines approach. Figure 7.22 presents the Galerkin scheme results for the case when $\mu = 0.5$ and $k = 0$ (no noise). Note that this solution barely differed from the finite differences solution for the noise free case presented in Figure 7.19.

Figure 7.22: Galerkin approach solution to SKPP equation with $\mu = 0.5$ and $k = 0$.

Figure 7.23 to Figure 7.26 present the Galerkin scheme results for $\mu = 0.5$ and increasing noise intensity, specifically $k = 0.1, 0.2, 0.5$ and $k = 1$ respectively. For comparison purposes, Gaines’ method was implemented and applied to the same parameter set. The corresponding results are presented in Figure 7.27 to Figure 7.30.
Figure 7.23: Galerkin approach solution to SKPP equation with $\mu = 0.5$ and $k = 0.1$

Figure 7.24: Galerkin approach solution to SKPP equation with $\mu = 0.5$ and $k = 0.2$
Figure 7.25: Galerkin approach solution to SKPP equation with $\mu = 0.5$ and $k = 0.5$.

Figure 7.26: Galerkin approach solution to SKPP equation with $\mu = 0.5$ and $k = 1$. 

$x \ u(x,t) \ ^* \ (8 \ 0 + \ 0.5$
Figure 7.27: Gaines’ method solution to SKPP equation with $\mu = 0.5$ and $k = 0.1$

Figure 7.28: Gaines’ method solution to SKPP equation with $\mu = 0.5$ and $k = 0.2$
Figure 7.29: Gaines' method solution to SKPP equation with $\mu = 0.5$ and $k = 0.5$

Figure 7.30: Gaines' method solution to SKPP equation with $\mu = 0.5$ and $k = 1.0$
Figure 7.31 presents the Galerkin scheme results for the case when $\mu = 0.2$ and $k = 0$ (no noise). Note that this solution barely differs from the finite differences solution for the noise free case presented in Figure 7.19.

![Figure 7.31: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 0$.](image)

Figure 7.32 to Figure 7.35 present the Galerkin scheme results for $\mu = 0.2$ and increasing noise intensity, specifically $k = 0.1, 0.2, 0.5$ and $k = 1$ respectively. Again, for comparison purposes, Gaines’ method was implemented and applied to the same parameter set. The corresponding results are presented in Figure 7.36 to Figure 7.39.
Figure 7.32: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 0.1$

Figure 7.33: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 0.2$
Figure 7.34: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 0.5$

Figure 7.35: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 1$ (20 terms).
Figure 7.36: Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 0.1$

Figure 7.37: Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 0.2$
Figure 7.38: Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 0.5$

Figure 7.39: Gaines’ method solution to SKPP equation with $\mu = 0.2$ and $k = 1.0$
At noise intensity level of $k = 1.0$, the Galerkin approach became numerically unstable at $t \geq 0.8$. Attempts were made at overcoming the instability by increasing the number of Galerkin terms from 20 to 50 (see Figure 7.40) and to 100 (see Figure 7.41). With 50 Galerkin terms the instability manifested itself at $t \geq 0.99$, whereas for 100 Galerkin terms the model ran without the numerical instability.

Figure 7.40: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 1$ (50 terms).

Figure 7.41: Galerkin approach solution to SKPP equation with $\mu = 0.2$ and $k = 1$ (100 terms).
7.6 The SKPP equation with initial point mass

In her solution of the SKPP equation

\[ du_t = \left[ \frac{1}{2} \mu^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{\mu^2} (1 - u_t) u_t \right] dt + F_{\mu} u_t \, dW_t, \]

Gaines [26] used the point mass conditions

\[ \phi(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{otherwise} \end{cases} \]

over the domain \([-2, 2]\). In the examples presented in this section the domain of interest was \([0, 1]\). Thus the equivalent comparative initial condition was

\[ \phi^*(x) = \begin{cases} 1 & x = 1/2 \\ 0 & \text{otherwise} \end{cases}. \]

This was mimicked by the continuous function

\[ f(x) = \exp \left[ - \left( x - \frac{1}{2} \right)^2 / A \right] \]

which has the required property

\[ \lim_{A \to 0} f(x) = \phi^*(x). \]

In my analysis A was set to \(10^{-6}\), so

\[ u(0, x) \equiv f(x) = \exp \left[ -10^6 \left( x - \frac{1}{2} \right)^2 \right]. \]

Applying the Galerkin approach

\[ u(0, x) \equiv \sum_{j=1}^{\infty} u^j(0) \phi_j(x) = \sqrt{2} \sum_{j=1}^{\infty} b^j \sin(j \pi x), \]

thus

\[ \sum_{j=1}^{\infty} b^j \sin(j \pi x) = \frac{1}{\sqrt{2}} \exp \left[ -10^6 \left( x - \frac{1}{2} \right)^2 / A \right]. \]

The left hand side of this expression is the Fourier series representation of the right hand side, hence the Fourier coefficients \(b^j\) could be written as

\[ b^j = \sqrt{2} \int_0^1 \exp \left[ -10^6 \left( x - \frac{1}{2} \right)^2 / A \right] \sin(j \pi x) \, dx. \]
Numerical experiments indicated that it required at least 2000 Galerkin terms to obtain a point mass magnitude of 0.9999.... To minimise the CPU requirements only 1750 Galerkin terms were used. This provided a point mass magnitude of 0.9998....

The linearly implicit Maruyama scheme results, based on the use of 1750 Galerkin terms, is presented in Figure 7.42. Note that this analysis took in excess of 330 CPU hours processing time.

Figure 7.42: Galerkin scheme approach for SKPP with point mass initial conditions.

For comparison purposes, the finite difference solution in the no noise case is presented in Figure 7.43.
Figure 7.43: Finite Differences solution for SKPP with point mass initial conditions.
7.7 Comparison of Methods

The Galerkin approach coupled with the linearly implicit Euler-Maruyama scheme has been shown to be a viable alternative to Gaines' finite difference discretisation approach. It suffers from the fact that it has high CPU requirements; with 20 Galerkin terms it ran approximately 7.5 times slower than the corresponding finite differences approach. This speed ratio can be improved by decreasing the number of Galerkin terms, but to the detriment of accuracy.

However, the finite difference scheme results suffer the disadvantages of most finite difference schemes and in particular when applied to stiff systems. These disadvantages are that the space and time step are either restricted or need to be carefully selected to avoid instabilities.

Finally, the application of the finite difference schemes in multi-dimensional systems can lead to substantially more complex implementations whereas the Galerkin–time discretisation approach is easily adapted to more complex systems.

7.8 Chapter Summary

This chapter has investigated the use of stochastic Taylor series based strong schemes as part of a multidimensional spatial discretisation in the solution of stochastic SPDEs. This discretisation has been applied via two avenues of discretisation; a finite differences approach and a Galerkin approach.

Following the spatial discretisation, the multidimensional system of SDEs was readily solved using the stochastic Taylor series based schemes. Thus the analysis undertaken in this chapter has validated that the stochastic Taylor series based time discretisation schemes can successfully solve multidimensional systems of SDEs. Thus the stochastic Taylor series based time discretisation schemes can be applied to multidimensional FPK systems.
Thesis Summary
Thesis Summary

This thesis has described the research that I have undertaken over the last eight years. My research has directly compared the numerical solution of SDEs using the stochastic Taylor series based time discretisation approach to the classical FEM approach. I have undertaken this comparative study by investigating the numerical solution of FPK equations in one, two and three dimensions. The stochastic Taylor series based time discretisation schemes’ solution was via the solution of the corresponding SDEs associated with FPK equations. The FEM approach solved the FPK equations directly, however, the three dimensional case is currently at the limit of practical application.

In the one, two and three dimensional systems analysed, the stochastic Taylor series based time discretisation approach generated suitably accurate versions of the underlying pdf solution. In some cases, especially in the case of the stiff one dimensional system considered, the time discretisation scheme solution succeeded, whereas the FEM approach solution degenerated due to numerical instabilities.

However, the main advantage of the time discretisation approach over the classical FEM approach was the ease of generation of the numerical schemes and the ease of implementation of the resulting schemes to generate accurate solutions. The time discretisation approach’s ease of generation, especially in higher dimensional problems, was partially due to the use of modern computer algebra systems – with MAPLE and the stochastic package used in my analyses. Furthermore, MAPLE was also used to generate optimised computer programming code representing the scheme, thus minimising the implementation phase.

To emphasise the ease of implementation of the time discretisation approach, consider that the three dimensional SDE system was solved using MICROSOFT EXCEL on a standard desktop PC. In comparison, the FEM approach in solving the same three dimensional system almost certainly required the use of a supercomputer and definitely required advanced matrix storage and manipulation algorithms. Note though, the complex FEM approach only generated the stationary pdf solution, whereas, as it is a natural consequence of the stochastic time discretisation approach, it readily provided the time evolution of the pdf. Furthermore, the FEM approach was critically dependent on the prior knowledge of the solution domain which can be difficult to determine. One approach to minimise the effort in determining the solution domain was to use a modern simula-
tion tool such as SIMULINK. As shown in this thesis, the SIMULINK approach readily and easily provided first order approximations to the solution of SDEs and these first order solutions could be used to determine the solution domain.

Finally, to further emphasise the advantages of the time discretisation approach, it was applied to a higher dimensional system of SDEs, via the solution of a spatially discretised SPDE. A 100 dimensional system was readily and efficiently solved. The approach used can be readily extended to any dimensional system.

Overall, this thesis has shown that the time discretisation approach provided solutions to SDEs, and specifically SDEs corresponding to the associated FPK equation, much more easily and readily than via the classical FEM approach. Their main disadvantage, in that their solutions are not as accurate as the FEM approach, can be minimised by running many more simulations. As the time discretisation approach led to schemes that are naturally suited to parallel or distributed processor systems, the running of many more simulations doesn't necessarily lead to a correspondingly greater turn around time. Even if supercomputers or parallel processor machines are not available, I have also shown in this thesis that it is easily possible to create a distributed processing system using the resources of a typical computer science laboratory.
Bibliography

[1] Ahrens, J. H. and Dieter, U.,

Computer Methods for Sampling from the Exponential and Normal Distributions,


Random Number Generators on Vector Supercomputers and other Advanced Architectures,


[3] Arnold, L.,

*Stochastic Differential Equations: Theory and Applications*,


[4] Benson, S. W.,

*The Foundations of Chemical Kinetics*,


Robust numerical solution of the Fokker–Planck equation for second order dynamical systems under parametric and external white noise excitation,

*Stochastic Modelling and Nonlinear Dynamics: Applications to Mechanical Systems*,

(Editor: W. Kliemann), CRC Press, 1994, (to appear)

[6] Bhandari, R. G. and Sherrer, R. E.,

Random vibrations in discrete nonlinear dynamic systems,

[7] Black, F., and Scholes, M.,
The pricing of options and corporate liabilities,

[8] Bratley, P., Fox, B. L. and Schrage, L. E.,

[9] Caughey, T. K.,
Nonlinear theory of random vibrations,

[10] Chandrasekhar, S.,
Stochastic problems in physics and astronomy: Noise and Stochastic Processes (ed: N. Wax),
Dover, New York, 1954.

A Practical Difference Scheme for Fokker-Planck Equations,

[12] Cyganowski, S. and Kloeden, P. E.,
Stochastic Stability Examined through Maple,
Proc. 15th IMACS World Congress, Volume 1: Computational Mathematics (Editor A. Sydow),

[13] Cyganowski, S.,
A MAPLE Package for Stochastic Differential Equations,
[14] Cyganowski, S., Grüne, L. and Kloeden, P. E.,
  MAPLE for Stochastic Differential Equations,
  Proceedings of the IX-th Summer School in Numerical Analysis, University of

  Convergence of numerical schemes for the solution of parabolic stochastic partial
  differential equations,

[16] Davies, A. J.,
  The Finite Element Method: A First Approach,

[17] Deuflhard P.,
  Recent Progress in Extrapolation Methods for Ordinary Differential Equations,

[18] Devore, J. L.,
  Probability and Statistics for Engineering and the Sciences,

[19] Einstein A.,
  Über die von der molekular-Kinetischen Theorie der Wärme geforderte Bewegung
  von in ruhenden Flüssigkeiten,

[20] Elworthy, K. D., Truman, A., Zhao, H. Z. and Gaines, J. G.,
  Approximate travelling waves for generalized KPP equations and classical mechanics,
[21] Elworthy, K. D., Zhao, H. Z. and Gaines, J. G.,

The Propagation of Travelling Waves for Stochastic Generalized KPP Equations,

[22] Entacher, K.,

Bad Subsequences of Well-Known Linear Congruential Pseudorandom Number Generators,

[23] Ferrenberg, A. M. and Landau, D. P.,

Monte Carlo Simulations: Hidden Errors from ”Good” Random Number Generators,


The wave of advance of advantageous genes,

[25] Fogelson, A. L.,


[26] Gaines, J. G.,

Numerical Experiments with S(P)DE’s,

[27] Glimm, J. and Jaffe, A.,

Quantum Physics. A Functional Integral Point of View,
Springer-Verlag, 1981.

[28] Grecksch, W., and Kloeden, P. E.,

Time-discretized Galerkin approximations of parabolic stochastic PDEs,
[29] Hammersley, J. M. and Handscomb, D. C.,

*Monte Carlo Methods*,


[30] Harris, C. J.,

Minimum variance control, estimation and simulation of nonlinear models of wastewater treatment systems,

*Proc. IEE, 125*, 1978, 441-446.

[31] Heemink, A. W.,

Stochastic modelling of dispersion in shallow water,


[32] Higham, D. J.,

An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations,


[33] Huebner, K. H.,

*The Finite Element Method for Engineers*,


[34] Itô, K.,

On Stochastic Processes,


[35] Itô, K.,

On stochastic differential equations,


[36] Jazwinski, A. H.,

*Stochastic Processes and Filtering Theory*,


Some experiments with massively parallel computation for Monte Carlo simulation of stochastic dynamical systems,

[38] Kendall, W. S.,

Computer algebra and stochastic calculus,


Computer Generation of Random Variables Using the Ratio of Uniform Deviates,

[40] Kirkpatrick, S. And Stoll, E.,

A Very Fast Shift-Register Sequence Random Number Generator,

[41] Klauder, J. R. and Petersen, W. P.,

Spectrum of Certain Nonself-Adjoint Operators and Solutions of Langevin Equations with Complex Drift,

[42] Kloeden, P. E., Platen, E. and Hofmann, N.,

Extrapolation Methods for the Weak Approximation of Itô Diffusions,

[43] Kloeden, P. E., Platen, E. and Schurz, H.,

The Numerical Solution of Stochastic Differential Equations through Computer Experiments,
[44] Kloeden, P. E.,

Method of Lines - Rothe's Method,

Presented at the Stochastic Numerics 2001 Conference, Zurich, February 19-21, 2001,

convener: Dr. Wes Petersen.

[45] Kloeden, P. E. and Platen, E.,

Numerical Solution of Stochastic Differential Equations,


[46] Kloeden, P. E. and Scott, W. D.,

Construction of Stochastic Numerical Schemes through MAPLE,


[47] Kloeden, P. E. and Shott, S.,

Linear–Implicit strong schemes for Ito–Galerkin approximations of stochastic PDEs,


[48] Kloeden, P. E., Platen, E., and Wright, I. W.,

The Approximation of Multiple Stochastic Integrals,


[49] Knuth, D. E.,

The Art of Computer Programming Vol 2,

Addison Wesley, Massachusetts, 1969.

[50] Kolmogoroff, A., Petrovsky, I. and Piscounoff, N.,

Etude de l'équation de la Diffusion Avec Croissance de la Matiere et son Application a un Probleme Biologique,


[51] Langley, R. S.,

Random Dynamic Analysis of Multibody Offshore Structures,

[52] Langley, R. S.,

A Variational Formulation of the FPK Equations with Application to the First Passage Problem in Random Vibration,


[53] Langtangen H. P.,

A general solution method for Fokker–Planck equations with applications to structural reliability,


[54] Lapidus, L. and Pinder, G. F.,

Numerical Solution of Partial Differential Equations in Science and Engineering,


[55] L’Ecuyer, P.,

Efficient and Portable Combined Random Number Generators,


[56] L’Ecuyer, P.,

Random Numbers for Simulation,

Communications of the ACM, Volume 33, Number 10, October, 1990.

[57] Leva, J. L.,

A Fast Normal Random Number Generator,


[58] Liu, S. C.,

Solutions of Fokker-Planck Equations with Applications in Nonlinear Random Vibration,

[59] Marsaglia, G. and Bray, T. A.,
A Convenient Method for Generating Normal Variables,

[60] Matsumoto, M. and Nishimura, T.,
Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator,

[61] Mignolet, M. P., and Fan, G. W.,
Non-Stationary Response of Some First-Order Non-Linear Systems Associated with the Seismic Study of Rigid Structures,

The Finite Element Method in Partial Differential Equations,

[63] Niederreiter, H.,
*Random Number Generation and Quasi-Monte Carlo Methods*,

[64] Oksendal, B.,
*Stochastic Differential Equations*,

[65] Park, S. C. and Miller, K. W.,
Random Number Generators: Good Ones Are Hard to Find,

[66] Petersen W. P.,
A general implicit splitting for stabilizing numerical simulations of Itô stochastic differential equations,
[67] Petersen W. P.,

SAM, Dept. of Mathematics HG G52.1
ETHZ
CH-8092, Zurich

[68] Reuter G. W.,

Numerical Solution to a Stochastic Interception Problem,

[69] Ripley, B. D.,

Stochastic Simulation,

[70] Ripley, B. D.,

Computer Generation of Random Variables: A Tutorial,

[71] Risken, H.,

The Fokker–Planck Equation,

[72] Roberts, J. B.,

Nonlinear Analysis of Slow Drift Oscillations of Moored Vessels in Random Seas,

[73] Roberts, J. B.,

A stochastic theory for Nonlinear Ship Rolling in Irregular Seas,
[74] Rubinstein, R. Y.,  
*Simulation and the Monte Carlo Method*,  

[75] Sauleév, V. K. and Cherinkiv, A. A.,  
Solution of Fokker-Planck-Kolmogorov Equation by the Method of Finite Differences,  

[76] Sharp, W. E. and Bays, C.,  
A Portable Random Number Generator for Single Precision Floating-Point Arithmetic,  

[77] Sherif, Y. S. and Dear, R. G.,  
The Development and Testing of a New Composite Random Number Generator,  

[78] Shinozuka M.,  
Monte Carlo Solution of Structural Dynamics,  

[79] Smith, G. D.,  

[80] Smoluchowski von, M.,  
Drei Vortage über Diffusion Brownsche Bewegung und Koagulation von Kolloidteilchen,  

[81] Sobczyk, K.,  
*Stochastic Differential Equations,*

[82] Soong, T. T. and Grigoriu, M.,

*Random Vibration of Mechanical and Structural Systems*,


[83] Spencer, Jnr., B. F. and Bergman, L. A.,

On the Numerical Solution of the Fokker–Planck Equation for Nonlinear Stochastic Systems,


[84] Standish, R. K.,

The Burnett Problem: Harnessing ANU’s hidden supercomputer,


[85] Stevens, W. R.,

*UNIX Network Programming*,


[86] Talay, D.,

*Analyse numérique des équations différentielles stochastiques*,


[87] Talay, D.,

Presto, mode d’emploi,


[88] To, C. W. S.,

Random Response of a Duffing Oscillator by the Stochastic Central Difference Method,

[89] Toland, R. H. and Yang, C. Y.,
Random Walk Model for First Passage Probability,

[90] Valkeila, E.,
Computer algebra and stochastic analysis, some possibilities,

[91] Vattulainen, I., Ala-Nissala, T. and Kankaala, K.,
Physical models as tests of randomness,

[92] Wagner W.,
Monte Carlo Evaluation of Functionals of Solutions of Stochastic Differential Equations, Variance Reduction and Numerical Examples,

[93] Wichmann, B. and Hill, D.,
Building a Random Number Generator,

[94] Wojtkiewicz, S. F., Bergman, L. A. and Spencer, B. F.,

[95] Yannios N. and Kloeden, P. E.,
Time Discretisation Solution of Stochastic Differential Equations,
[96] Yannis N.,

Finite Element Method Solution of One Dimensional FPK Equations,


[97] Yannis N.,

Using MAPLE in the Finite Element Solution of the Fokker-Plank-Kolmogorov Equation,


[98] Yannis, N.,

A nearly code free numerical solution of Stochastic Differential Equations,

Presented at the Stochastic Numerics 2001 Conference, Zurich, February 19-21, 2001,

convener: Dr. Wes Petersen.

[99] Zeisel, H.,

A Remark on Algorithm AS183, An Efficient And Portable Pseudo-random Number Generator,