Collocation techniques for solving neural field models on complex cortical geometries

A thesis by Rebecca Martin
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy.

Department of Physics and Mathematics
Nottingham Trent University
August 2018.

Copyright Statement

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Jonathan Crofts for his invaluable guidance and constant support throughout my studies. I am also very grateful to my second supervisor, Dr. David Chappell and third supervisor, Professor Nadia Chuzhanova, for all the help and advice they have provided. I wish to also thank the Mathematics department for all the support and encouragement, especially Dr. Martin Nelson and Dr. Janis Bajars. I also appreciate the financial support provided by Nottingham Trent University via an RAE funded PhD scholarship.

Finally, I would like to thank my friends and family. In particular my sister Laura, grandad Keith, gran Primrose and partner Thomas, for always being there to support and encourage me throughout the years. I wish to extend my biggest thank you to my mother, Carol, for her support, reassurance and for everything that she has done for me, it is greatly appreciated. Thank you!

Abstract

Neural models are deployed in order to gain an insight into the function and behaviour of the brain at a range of different scales, ranging from the micro-scale modelling of individual neurons, to the meso- and macro-scale modelling of large populations of neurons. Neural field models provide a continuous approach to modelling at this larger scale, and typically take the form of a nonlinear partial integro-differential equation. Such equations are capable of supporting a variety of patterns and have been linked to neurological phenomena, such as, for example, bumps in models of working memory, and thus play an important role in the interpretation and understanding of the complex, dynamic patterns of brain activity observed via modern brain imaging techniques such as EEG, MEG and fMRI.

In this thesis, we present an approach for solving neural field equations on surfaces more akin to the cortical geometries typically obtained from neuroimaging data. Our approach involves solving the integral of the partial integro-differential equation directly using collocation techniques, alongside efficient numerical procedures for determining geodesic distances between neural units. To illustrate our methods we study localised activity patterns in two different neural field models; namely, the Amari equation, for which we consider stationary bump solutions, and an extended version of the Amari equation that admits both stationary and travelling bump solutions. We solve both equations on a variety of domains, including a flat periodic domain, the curved surface of the torus and the folded surface of the rat cortex. Importantly, we find that collocation techniques are able to replicate solutions obtained using more standard Fourier based methods on a flat, periodic domain, independent of the underlying mesh. This result is particularly significant given the highly irregular nature of the type of meshes derived from modern neuroimaging data.

One of the key contributions of this thesis is our ability to solve neural models on curved geometries for which no analytic formula for the geodesic distance exists. Indeed, by deploying efficient numerical schemes to compute geodesics, our approach

is not only capable of modelling macroscopic pattern formation on realistic cortical geometries, such as the rat brain considered herein, but can also be extended to include cortical architectures of more physiological relevance. Importantly, such an approach provides a means by which to investigate the influence of cortical geometry upon the nucleation and propagation of spatially localised neural activity and beyond, and thus promises to provide model-based insights into disorders like epilepsy, or spreading depression, as well as healthy cognitive processes like working memory or attention.

List of Figures

1.1 (a) Simulation of the Hodgkin-Huxley equations in (1.1) for $I = 0$; and (b) a bifurcation diagram showing how oscillatory solutions arise via a Hopf bifurcation as we increase the external current I (plot was produced using the XPPAUT package) in the HH model. 4

1.2 A bifurcation diagram for the Morris-Lecar model as a function of the external current I with all other parameters fixed. 6

1.3 Schematic diagram of the integrate and fire model. 8

1.4 Bifurcation set of a single Wilson-Cowan node for $c_1 = c_2 = c_3 = 10$ and $c_4 = -2$. The dashed curve denotes the saddle-node bifurcation set and the solid curve the Hopf bifurcation set. (Figure created using XPPAUT [1].) 10

2.1 A pair of synaptically coupled neurons 16

2.2 Steady state solution of (2.17) for $N = 15,\ h = 1/3,\ \beta = 20$. 26

2.3 The norm of $\mathbf{v}$ versus the firing rate threshold, h. 27

2.4 Solutions occur at the intersection of $S(u)$ and u. 28

2.5 *Upper*: space-time plot of the solution of (2.17), where the domain has been truncated to $[0, 50]$. Parameters: $\beta = 20$ and $h = 0.47$ *Lower*: the solution at time $t = 15$. 29

2.6 Branch of solutions with initial parameters $c_0 = 0.8$ and $h_0 = 0.3$. . . 30

2.7 Snap shot of the travelling bump solution at $t = 250$. *Left column* u, *right column* a. Parameters: $L = 7.5, \quad A = 2, \quad \beta = 5, \quad h = 0.8, \quad B = 0.4, \quad \tau = 3$ 32

2.8 Speed of the bump c, as a function of the sensitivity A. Parameters: $N = 256, \quad L = 7.5, \quad \beta = 5, \quad h = 0.8, \quad B = 0.4, \quad \tau = 3$ 32

2.9 (a) Multiple bump solution. (b) Travelling wave solution. 34

3.1 The unit simplex with (a) three and (b) six interpolation nodes, respectively. 41

3.2 (a) Illustration of a square domain that uses Cartesian grid points as triangle vertices. (b) Illustration of a triangulated torus using regularly spaced grid points in the toroidal coordinate system as triangle vertices. 46
3.3 (a) Illustration of a DistMesh triangulation of the periodic square. (b) Illustration of a DistMesh triangulation of the torus. 47
3.4 (a) Illustration of a random triangulation of the periodic square. (b) Illustration of a random triangulation of the torus. 49
3.5 Triangulated surface of the left hemisphere of a rat brain. 49
3.6 Illustration of the update upwind procedure for (a) uniform Cartesian meshes and (b) simple triangulated meshes. 52
3.7 Comparing the fast marching algorithm (blue dashed) and exact geodesic algorithm (red) to the analytic distance on the unit sphere; (a) maximal absolute error, (b) maximal relative error, (c) average relative error. 56
3.8 Geometric representation of the pseudo-arclength continuation scheme. 58

4.1 Illustration of a domain that uses Cartesian grid points as triangle vertices. 63
4.2 The error $|I_m - I_{m+1}|$ plotted against grid size N_{m+1} reveals geometric convergence rates for (a) FFTs, (b) trapezoidal rule and (c) linear collocation, when computing the integral in (4.3). 64
4.3 Convergence of quadratic collocation when computing the integral (4.3) with the error $|I_m - I_{m+1}|$ plotted against grid size N_{m+1}. 65
4.4 Illustration of the refinement of each triangle, where A, B and C are the original triangle vertices and a, b and c are the new vertices introduced by the refinement. 67
4.5 An illustration of the refinement procedure for a Distmesh triangulation. 67
4.6 Convergence of linear and quadratic collocation when computing the integral in (4.3) using a DistMesh triangulation (see Figure 4.5). . . . 68
4.7 An illustration of the refinement procedure for a random triangulation. 68
4.8 Convergence of linear and quadratic collocation when computing the integral in (4.3) using a random triangulation as illustrated in Figure 4.7. 69
4.9 Initial condition for u on the periodic square. 69

4.10 (i) Solitary bump solutions obtained when implementing FFTs, trapezoidal and linear collocation on the Cartesian grid based triangulation. (ii) Solitary bump solutions on the random triangulation perturbed by; (a)&(c) 10% and (b)&(d) 20% when implementing linear collocation. (iii) Solitary bump solution on the DistMesh general triangulation. 70
4.11 (i) Three different solutions as highlighted along the solution branch in Figure 4.11(ii): (a) & (d) point P_1 (stable), (b) & (e) point P_2 (stable) and (c) & (f) point P_3 (unstable). (ii) Solution branches as the parameter h is varied when implementing FFTs and linear collocation to solve Equation (4.7). (iii) The eigenvalues of the Jacobian matrix evaluated at the bifurcation point $h^* \approx 1.03$. 72
4.12 (i) The three solutions selected along the solution branch in Figure 4.12(ii): (a)&(d) point P_1 (unstable), (b)&(e) point P_2 (stable) and (c)&(f) point P_3 (unstable). (ii) Solution branches as the parameter A is varied when implementing FFTs and linear collocation to solve Equation (4.7). (iii) Eigenvalues of the Jacobian matrix evaluated at the bifurcation point $A^* \approx 1.2$. 74
4.13 Parameterisation of a torus by coordinates (θ, ϕ). 76
4.14 Convergence of the trapezoidal and linear collocation methods when computing the integral (4.10) on the surface of a torus using a Cartesian grid based triangulation as illustrated in Figure 4.13. 77
4.15 Plot (a) shows an illustration of the first three refinements of a DistMesh triangulation. Plots (b) and (c) display convergence results for linear collocation when computing the integral in (4.10) using DistMesh and random triangulations, respectively. 78
4.16 Initial conditions u_0 when solving Equation (4.1) starting; (a) on the outside of the torus and (b) on the inside of the torus. 79
4.17 (i) Solitary bump solution found on the outside of the Cartesian grid based torus; (a)&(c) trapezoidal method, (b)&(d) linear collocation. (ii) Solitary bump solution found on the inside of the Cartesian grid based torus; (a)&(c) trapezoidal method, (b)&(d) linear collocation. (iii) Solitary bump solutions found on the DistMesh triangulation of the torus. 81

4.18 (i) Solution branches as the parameter h is varied when implementing trapezoidal and linear collocation to solve (4.7) for varying curvatures; *blue* R_1, *green* R_2, *red* R_3 and *black* the periodic square. (ii) The four solutions along the solution branch in Figure 4.18(i); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches. (iii) Plot of the eigenvalues of the Jacobian matrix evaluated at the bifurcation points h^* plotted for each branch. 83

4.19 (i) Solution branches as the parameter h is varied; *blue* R_1, *green* R_2 and *black* the periodic square. (ii) Solution branches as the parameter h is varied for curvature R_3. (iii) Solutions (a)&(b) P_1 and P_2 from Figure 4.19(i) and (c)&(d) P_1 and P_2 from Figure 4.19(ii). (iv) Eigenvalues of the Jacobian matrix evaluated at the bifurcation point for each branch. (v) Plot of the eigenvalues of the Jacobian matrix evaluated at the bifurcation point. 84

4.20 (i) Solution branches as the parameter A is varied for varying curvatures; *blue* R_1, *green* R_2, *red* R_3 and *black* the periodic square. (ii) The four solutions corresponding to points P_1–P_4 highlighted on the solution branches in Figure 4.20(i); the solutions displayed in figures (a)&(b) P_1 and P_2 on the R_1 branch (c)&(d) P_3 and P_4 on the R_3 branch. (iii) The eigenvalues of the Jacobian matrix evaluated at the bifurcation points A^*. 86

4.21 (i) Solution branches as the parameter A is varied for varying curvatures; *blue* R_1, *green* R_2 and *black* the periodic square. (ii) Solution branches as the parameter A is varied on the torus with curvature R_3. (iii) The eigenvalues of the Jacobian evaluated at the bifurcation points on the R_1 and R_2 branches. (iv) The eigenvalues of the Jacobian evaluated at the bifurcation point on the R_3 branch. (v) Solutions (a)&(b) P_1 and P_2 in Figure 4.21(i) (c)&(d) P_3 and P_4 in Figure 4.21(ii). 87

5.1 Initial conditions for; (a) neural activity u and (b) the recovery variable a, on the periodic square. 90

5.2 (i) Travelling bump solutions of (5.1), computed on the Cartesian grid based triangulation of the periodic square implementing; (a) & (d) FFT, (b) & (e) trapezoidal rule and (c) & (f) linear collocation. (ii) Travelling bump solutions of (5.1), solved implementing linear collocation on a random triangulation generated by perturbing the Cartesian grid based triangulation by 20%; (a) & (c) best case and (b) & (d) worst case. (iii) Travelling bump solutions of (5.1), solved implementing linear collocation on a random triangulation generated by perturbing the Cartesian grid based triangulation by 10%. (iv) Travelling bump solutions of (5.1), computed implementing linear collocation on a DistMesh triangulation of the periodic square. 91

5.3 (i) Solutions from Figure (5.3)(ii); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. (ii) Solution branches as the parameter A is varied against c. (iii) The eigenvalues plotted at the saddle-node bifurcation. 96

5.4 (i) Solutions from Figure (5.4)(ii); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. (ii) Solution branches as the parameter h is varied against c. (iii) The eigenvalues plotted at the saddle-node bifurcation. 97

5.5 (a)Travelling bump solution found on the outside of the Cartesian grid based torus when implementing linear collocation. (b) Travelling bump solution found on the inside of the Cartesian grid based torus when implementing linear collocation. (c) The travelling bump solution found on the outside of the DistMesh triangulation of the torus implementing linear collocation. (d) The travelling bump solution found on the inside of the DistMesh triangulation of the torus implementing linear collocation. 100

5.6 (a) The tracked path of a bump solution with non-constant speed plotted on the torus shaded with Gaussian curvature. (b) The curvature and speed of the bump plotted in time. (c) The speed plotted against curvature for the three curvatures of the torus; R, $R/2$ and $3R/4$, alongside a slope of gradient 1. 101

5.7 Travelling bump solutions found when solving (5.1) on the triangulated surface of a rat brain with different initial conditions. 103

5.8 The Gaussian curvature plotted on the surface of the rat brain along with the tracked path of the solution in Figure 5.7 and the geodesic from the start point to the end point. 104
5.9 The speed of the bump solution in Figure 5.7 and the curvature along the trajectory of this solution, which is displayed by red line in Figure 5.8. 105

6.1 Travelling bump solution found when solving (5.1) on the triangulated surface of a human brain. 112

C.1 Solution branches as the parameters; (a) h and (b) A are varied when implementing linear collocation on a general triangulation. 142
C.2 The three solutions selected along the solution branch in Figure C.1(a); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. 142
C.3 The three solutions selected along the solution branch in Figure C.1(b); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. 143
C.4 Solitary bump solutions found on the random triangulation (perturbed by 20%) of the torus. *Left*, from the initial state centred at $\theta = \phi = 0$ and *right* from the initial state centred at $\theta = \phi = \pi$ . . 143
C.5 Solution branches obtained when implementing linear collocation on a general triangulation of a torus when varying h. In all cases; blue - R_1, green - R_2, red -R_3 and black denotes the solution branch obtained on the periodic square. (a) Branches found when considering solutions centred at $\theta = \phi = 0$ (b) branches found when considering solutions centred at $\theta = \phi = \pi$ (c) The branch obtained considering the solution centred at $\theta = \phi = 0$ on the torus with curvature R_3. 144
C.6 The four solutions labelled P_1–P_4 along the branch in Figure C.5(a); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches. 145
C.7 The four solutions labelled P_1–P_4 along the branch in figure C.5(b) and C.5(c); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches. 146

C.8 Solution branches obtained when implementing linear collocation on a general triangulation of a torus when varying A. In all cases; blue - R_1, green - R_2, red -R_3 and black denotes the solution branch obtained on the periodic square. (a) Branches found when considering solutions centred at $\theta = \phi = 0$ (b) branches found when considering solutions centred at $\theta = \phi = \pi$ (c) The branch obtained considering the solution centred at $\theta = \phi = 0$ on the torus with curvature R_3. 147
C.9 The four solutions labelled P_1–P_4 along the branch in Figure C.8(a); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches. 148
C.10 The four solutions labelled P_1–P_4 along the branch in figure C.8(b) and C.8(c); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches. 149
C.11 Solitary bump solutions found, from the initial state centred at $\theta = \phi = 0$, on the torus with curvature R_2, when implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation. 150
C.12 Solitary bump solutions found, from an initial state centred at $\theta = \phi = \pi$, on the inside if the torus with curvature R_2, when implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation. 151
C.13 The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_2, implementing linear collocation on a DistMesh triangulation. 152
C.14 The solitary bump solution found, from an initial state centred at $\theta = \phi = \pi$, on the inside of the torus with curvature R_2, implementing linear collocation on a DistMesh triangulation. 153
C.15 The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_2, implementing linear collocation on a random triangulation. 153
C.16 The solitary bump solution found, from an initial state centred at $\theta = \phi = \pi$, on the inside of the torus with curvature R_2, implementing linear collocation on a random triangulation. 154
C.17 Solitary bump solutions found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_3, implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation. 155

C.18 Solutions found, from an initial state centred at $\theta = \phi = \pi$, on the torus with curvature R_3, implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation. 156
C.19 The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_3, when implementing linear collocation on a DistMesh triangulation. 157
C.20 The solution found, from an initial state centred at $\theta = \phi = \pi$, on the torus with curvature R_3, when implementing linear collocation on a DistMesh triangulation. 157
C.21 The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curature R_3, when implementing linear collocation on a random triangulation. 158
C.22 The solution found, from an initial state centred at $\theta = \phi = \pi$, on the torus with curvature R_3, when implementing linear collocation on a random triangulation. 158
C.23 The travelling bump solution found, from an initial state centred at $\theta = \phi = 0$, on the Cartesian grid based torus when implementing the trapezoidal method. 159
C.24 The travelling bump solution found, from an initial state centred at $\theta = \phi = \pi$, on the Cartesian grid based torus when implementing the trapezoidal method. 160
C.25 Travelling bump solution found, from an initial state centred at $\theta = \phi = 0$, on the random triangulation of the torus when implementing linear collocation. 161
C.26 Travelling bump solution found, from an initial state centred at $\theta = \phi = \pi$ on the random triangulation of the torus when implementing linear collocation. 162
C.27 Solution branches found as the parameters; (a) A and (b) h are varied against c when implementing quadratic collocation. 163
C.28 The three solutions labelled along the branch in Figure C.27(a); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. 164
C.29 The three solutions labelled along the branch in Figure C.27(b) ; (a) & (d) point P_1, (b) & (e) point P_3 and (c) & (f) point P_3. 165

D.1 (a) Initial condition for u on the periodic square. (b) Initial condition for a on the periodic square. 166

D.2 A solitary bump solution obtained when implementing quadratic collocation on the Cartesian grid based triangulation of the periodic square. 167
D.3 A solitary bump solution obtained when implementing quadratic collocation on the DistMesh general triangulation of the periodic square. 168
D.4 Solitary bump solutions obtained when implementing quadratic collocation on the random triangulation of the periodic square with perturbation; (a)&(c) 10% and (b)&(d) 30%. 168
D.5 Travelling bump solution computed on the Cartesian grid based triangulation of the periodic square implementing quadratic collocation. 169
D.6 Travelling bump solutions computed when implementing quadratic collocation on the DistMesh triangulation of the periodic square. . . . 170
D.7 Travelling bump solutions found when implementing quadratic collocation on the random triangulation of the periodic square, perturbed by; (a)&(c) 10% and (b)&(d) 30%. 170

E.1 Triangulation of the down sampled left hemisphere of the human cortical surface. 173
E.2 Travelling bump solution found when solving (5.1) on the triangulated surface of a human brain. 174
E.3 (a) The bump solution shown in figures 6.1 and E.2 at time, $t = 100$. (b) The bump solution shown in figures 6.1 and E.2 at time, $t = 400$. 175

Contents

1 Introduction **1**
- 1.1 A brief historical background . . . 2
  - 1.1.1 Single neuron models . . . 2
  - 1.1.2 Population models . . . 8
  - 1.1.3 Neural field models . . . 11
- 1.2 Thesis overview . . . 12
- 1.3 Publications and presentations . . . 13

2 Neural field models **15**
- 2.1 Firing rate models . . . 16
  - 2.1.1 Deriving the neural field model . . . 19
- 2.2 The well-posedness of the neural field model . . . 19
  - 2.2.1 Assumptions . . . 20
  - 2.2.2 Volterra formulation . . . 20
- 2.3 Techniques for solving neural field models . . . 22
  - 2.3.1 PDE methods . . . 22
  - 2.3.2 Direct numerical methods . . . 24
  - 2.3.3 Numerical solution of a two dimensional NFM . . . 31
- 2.4 Solutions to NFMs and pattern formation . . . 33
- 2.5 Summary . . . 35

3 Technical preliminaries **36**
- 3.1 Nonlinear integral equations . . . 36
  - 3.1.1 Projection methods . . . 37
- 3.2 Solving integral equations on triangulated domains using collocation methods . . . 39
  - 3.2.1 Interpolation over triangles . . . 39
  - 3.2.2 Numerical integration . . . 42
  - 3.2.3 Implementing the collocation method . . . 43
- 3.3 Domains . . . 45
  - 3.3.1 Cartesian grid based triangulation . . . 45
  - 3.3.2 General triangulations . . . 46
  - 3.3.3 Rat brain . . . 48

3.4 Computing geodesics on triangulated surfaces 50
3.4.1 Fast marching algorithm 50
3.4.2 Exact geodesic algorithm 53
3.4.3 Comparing the fast marching and exact geodesic algorithms . 55
3.5 Numerical bifurcation analysis 55
3.5.1 Bifurcation detection of equilibria for NFMs 59
3.6 Summary 60

4 The Amari equation **62**
4.1 Numerical Results 63
4.1.1 Planar domain with periodic boundary conditions 63
4.1.2 Torus 75
4.2 Summary 85

5 An adaptive neural field model **88**
5.1 Numerical Results 89
5.1.1 Planar domain with periodic boundary conditions 89
5.1.2 Torus 98
5.1.3 Cortical surface of the rat brain 99
5.2 Summary 105

6 Conclusions and future work **107**
6.1 Conclusions 107
6.2 Plan of future work 109
6.2.1 Computational optimisation 109
6.2.2 Higher order interpolation methods 110
6.2.3 Extending the method to more convoluted geometries 111
6.2.4 Making the model more physiologically realistic 113
6.2.5 Potential clinical applications 114

Bibliography **115**

A Banach spaces **135**
A.1 Some important examples of Banach spaces 136

B Standard methods **137**
B.1 FFTs in 2D 137
B.2 The trapezoidal method 139

C Additional Results **141**
C.1 Chapter 4 141
C.1.1 R_2 152
C.1.2 R_3 154
C.2 Chapter 5 160

D Quadratic collocation **166**
D.1 Amari equation 166
D.2 Adaptive NFM 169
D.3 Summary 171

E Human Brain **172**

Chapter I

Introduction

The brain is a highly complex system containing billions of connections [2], and so its understanding will require novel computational approaches capable of solving the types of large-scale neural models that arise from the physiologically detailed descriptions necessary to capture fundamental features of neural systems, across multiple scales [3, 4, 5, 6]. Computational neuroscience is a vast, interdisciplinary topic that links many scientific fields such as neuroscience, physics, mathematics and informatics and computer science [7, 8, 9]. The foundations of the subject date back to the beginnning of the 20th century when Lapicque [10, 11] first proposed the now famed *integrate-and-fire* model of a neuron, which despite lacking detailed biophysics, is still one of the most popular models in computational neuroscience to date [12, 13, 14, 15]. More detailed models, that account for the biophysics of excitable membranes, were later introduced, most notably during the 1950's in the early work of Beurle [16], and Hodgkin and Huxley [17]. It was these models that further inspired a multitude of neural models, such as the models of Wilson and Cowan [18, 19] and Amari [3], the overarching aim of which was to use mathematical and computational techniques in order to explain/predict brain function and behaviour.

Recently these goals have gained additional traction, due mainly to recent advances in experimental neuroscience, particularly neuroimaging. This has led to a concerted effort in developing and sharing new techniques for understanding the complex structure and function of the nervous system amongst the scientific community. For example, the US led Human Connectome Project [20, 21], which is partly funded by the European Union, aims to provide access to an unparalleled compilation of neural data, and so is expected to greatly advance our understanding

of the human brain. Another US led program is the Brain Research through Advancing Innovative Neurotechnologies (BRAIN) initiative, which was announced in 2013 by the Obama administration to help researchers studying different brain disorders [22, 23, 24]. A similar European program is that of the Virtual Brain Project [25], which employs cutting edge network and mathematical modelling techniques to simulate patient specific neural behaviour akin to that observed in clinical scanners. Examples of UK initiatives that promote the open access and sharing of neurological data include, for example, the UK Neuroinformatics Society and the Brain Banks Network [26, 27], both of which were developed to aid cross disciplinary research and sharing of neurological data of all types (*e.g.* imaging data, tissue samples, genetic information, *etc.*).

This data deluge enables us to consider increasingly detailed models of neural activity. For example, it is possible to build complicated, multi-scale models of neural activity [28, 29, 30] that increasingly make use of different imaging modalities [31, 32]. This has enabled the accurate reproduction of the types of neural activities observed using imaging technologies, in the same spirit as the Virtual Brain project [25]. Here, however, we restrict to less detailed neural models from a biological point of view, as we focus more on the techniques used to solve the underlying mathematical model, with the overarching aim of solving and studying the solutions of such models directly on complex curved geometries.

1.1 A brief historical background

In this section we outline a brief history of neural modelling starting with models of individual neurons and building up to the modelling of large (possibly infinite) populations of neurons.

1.1.1 Single neuron models

The Hodgkin-Huxley model

In 1952 Sir Alan Lloyd Hodgkin and Sir Andrew Fielding Huxley developed one of the most widely used microscale models in neuroscience, in order to investigate the properties of action potentials in the squid giant axon [17, 33, 34]. The *Hodgkin-Huxley model* (HH) as it is known, consists of four coupled ordinary differential equations that describe the voltage across the cell membrane as well as the three ionic currents responsible for the potential: sodium (Na^+), potassium (K^+) and a

so-called 'leak' current [35, 36]. Note that Hodgkin and Huxley used an ingenious clamping technique in order to isolate the ionic currents and thus determine their functional form [2]. The complete set of HH equations are given by

$$\begin{aligned} C\dot{V} &= I - \bar{g}_{\mathrm{K}} n^4 (V - V_{\mathrm{K}}) - \bar{g}_{\mathrm{Na}} m^3 h (V - V_{\mathrm{Na}}) - \bar{g}_{\mathrm{L}} (V - V_{\mathrm{L}}) \\ \dot{n} &= (n_\infty(V) - n)/\tau_n(V) \\ \dot{m} &= (m_\infty(V) - m)/\tau_m(V) \\ \dot{h} &= (h_\infty(V) - h)/\tau_h(V), \end{aligned} \tag{1.1}$$

where

$$\begin{aligned} n_\infty &= \alpha_n/(\alpha_n + \beta_n), & \tau_n &= 1/(\alpha_n + \beta_n), \\ m_\infty &= \alpha_m/(\alpha_m + \beta_m), & \tau_m &= 1/(\alpha_m + \beta_m), \\ h_\infty &= \alpha_h/(\alpha_h + \beta_h), & \tau_h &= 1/(\alpha_h + \beta_h), \end{aligned}$$

with

$$\begin{aligned} \alpha_n(V) &= \frac{0.01\,(V+55)}{1-\exp(-0.1\,(V+55))}, & \alpha_m(V) &= \frac{0.1\,(V+40)}{1-\exp(-0.1\,(V+40))}, \\ \alpha_h(V) &= 0.07\exp(-0.05\,(V+65)), & \beta_n(V) &= 0.125\exp(-0.0125\,(V+65)), \\ \beta_m(V) &= 4\exp(-0.0556\,(V+65)), & \beta_h(V) &= \frac{1}{1+\exp(-0.1\,(V+35))}. \end{aligned}$$

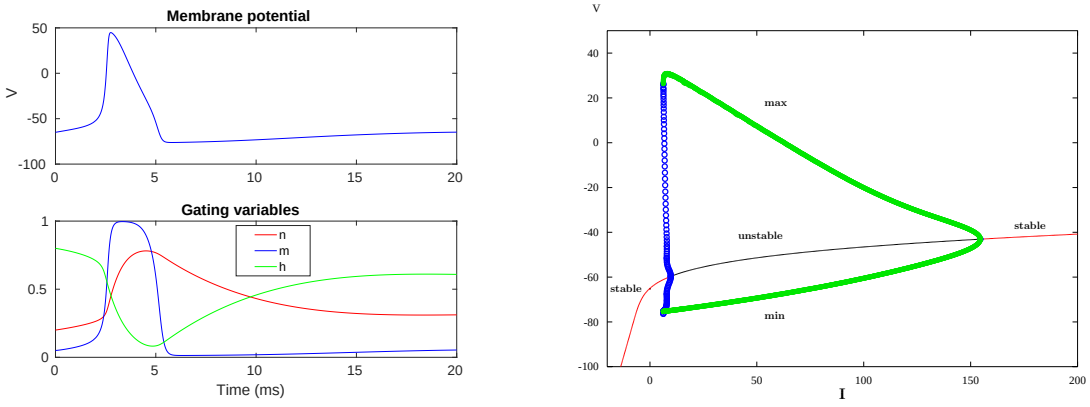
Here, n, m and h are dimensionless quantities lying between 0 and 1 that are associated with K^+ channel activation, Na^+ channel activation, and Na^+ channel inactivation, respectively; $\bar{g}_{\mathrm{K}}, \bar{g}_{\mathrm{Na}}$ and $\bar{g}_{\mathrm{L}}$ are the maximal conductances for the three currents; and the functions $\alpha_p(V)$ and $\beta_p(V)$ for $p \in \{n, m, h\}$ describe the transition rates between open and closed states of the channels.

Figure 1.1(a) shows a simulation of equations (1.1) for the following parameter values

$$\begin{aligned} &I = 0\mu\text{A cm}^2, \quad C = 1\mu\text{F}, \quad V_{\mathrm{K}} = -77\text{mV}, \quad V_{\mathrm{Na}} = 50\text{mV}, \qquad V_{\mathrm{L}} = -54.4\text{mV}, \\ &\bar{g}_{\mathrm{Na}} = 120\text{mmho cm}^{-2}, \quad \bar{g}_{\mathrm{K}} = 36\text{mmho cm}^{-2}, \quad \text{and} \quad \bar{g}_{\mathrm{L}} = 0.3\text{mmho cm}^{-2}. \end{aligned}$$

Here, all potentials are measured in mV, times in ms and currents in μA per cm^2. Initial conditions were chosen as follows

$$V(0) = -65, \quad n(0) = 0.2, \quad m(0) = 0.05 \quad \text{and} \quad h(0) = 0.8.$$



(a) (b)

Figure 1.1: (a) Simulation of the Hodgkin-Huxley equations in (1.1) for $I = 0$; and (b) a bifurcation diagram showing how oscillatory solutions arise via a Hopf bifurcation as we increase the external current I (plot was produced using the XPPAUT package [1]) in the HH model.

Note that for this set-up the system has a solitary stable fixed point and the neuron is said to be *excitable* as any small perturbation from the fixed point results in a strong system response, as can be readily seen from Figure 1.1(a). The above example considers the rather uninteresting case in which the external current is set to zero (*i.e.* $I = 0$); however, when a positive external current is applied the model can admit periodic solutions, which arise via a Hopf bifurcation as shown in Figure 1.1(b).

Despite its complexity one can obtain a greatly simplified model of the HH equations by making the following two observations:

1. $\tau_m(V)$ is small for all V and so the variable $m(t)$ quickly approaches its equilibrium value; and

2. $1-h$ and n behave in a qualitatively similar way and so can be *slaved* together.

The above ideas have led to a number of simplified models that can replicate many of the most important features of the HH model, and indeed, we shall briefly consider a number of such examples below; however, the interested reader should consult, for example, [37] and references therein for further details.

The Fitzhugh Nagumo model

In 1962 Fitzhugh and Nagumo developed one of the most simple models of spike generation [38, 39, 40]. This model is a two-dimensional simplification of the Hodgkin-

Huxley model, and its motivation was to isolate properties of the sodium and potassium ion flow that cause excitation and propagation [41]. The Fitzhugh-Nagumo model is given by,

$$\begin{aligned} \frac{\mathrm{d}V}{\mathrm{d}t} &= V - \frac{V^3}{3} - W + I \\ \frac{\mathrm{d}W}{\mathrm{d}t} &= 0.08(V + 0.7 - 0.8W). \end{aligned} \tag{1.2}$$

Here V is the membrane potential, W a recovery variable and I denotes an external current. Due to the simplicity of the Fitzhugh-Nagumo model its entire phase trajectory can be observed at once, compared to the HH model where you only observe projections of its four-dimensional phase trajectories [42]. Thus allowing a geometrical explanation (*i.e.* a dynamical systems approach) to phenomena linked to spike generating mechanisms and excitability of neurons [43].

The Morris-Lecar model

In 1981 the Morris-Lecar (ML) model was derived by Cathy Morris and Harold Lecar. It was developed to describe voltage patterns of Barnacle muscle fibers [44]; however, it is also useful for modelling fast-spiking neurons [45]. The model is a two-dimensional excitation model [46] which is again a simplification of the HH model. It describes the same three currents as in the HH model but uses only two dynamical variables [47]. This simplification makes the model very popular in computational neuroscience [48]. The ML model is given by

$$\begin{aligned} C_{\mathrm{M}}\frac{\mathrm{d}V}{\mathrm{d}t} &= -\bar{g}_{\mathrm{L}}(V - V_{\mathrm{L}}) - \bar{g}_{\mathrm{Ca}}M_{\infty}(V)(V - V_{\mathrm{Ca}}) - \bar{g}_{\mathrm{K}}N(V - V_{\mathrm{K}}) + I, \\ \frac{\mathrm{d}W}{\mathrm{d}t} &= \frac{W_{\infty}(V) - W}{\tau_W(V)}. \end{aligned} \tag{1.3}$$

As usual, V represents the membrane potential, I is an applied external current, and W a recovery variable that represents the probability that a K^+-ion channel is open [44]. The parameters V_{Ca}, V_{K} and V_{L} denote the equilibrium potentials of Ca^{2+}, K^+ and the leak current respectively and the maximum conductances of the corresponding ionic currents are denoted by $\bar{g}_{\mathrm{Ca}}$, $\bar{g}_{\mathrm{K}}$ and $\bar{g}_{\mathrm{L}}$. In the model $M_{\infty}(V)$

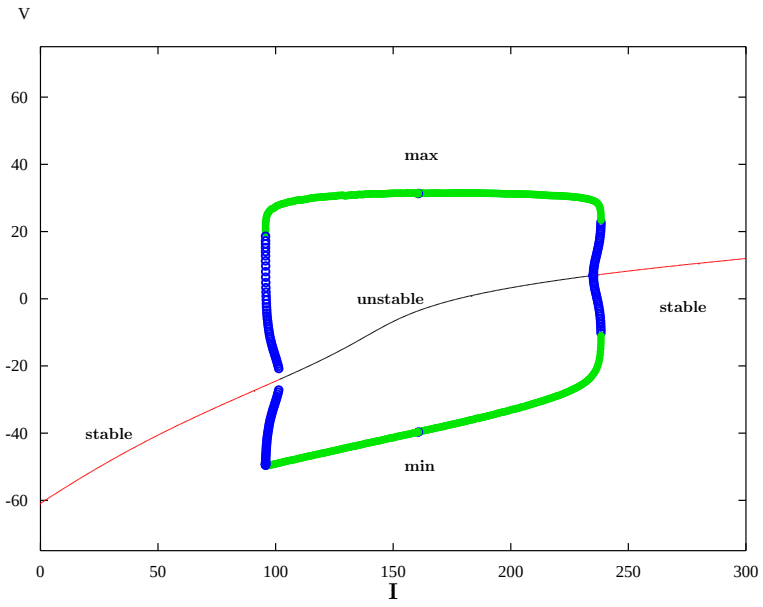


Figure 1.2: A bifurcation diagram for the Morris-Lecar model as a function of the external current I with all other parameters fixed.

and $W_\infty(V)$ are open-state probability functions given by

$$M_\infty = \frac{1 + \tanh\left(\frac{V-V_1}{V_2}\right)}{2}, \tag{1.4}$$

$$W_\infty = \frac{1 + \tanh\left(\frac{V-V_3}{V_4}\right)}{2}. \tag{1.5}$$

Here V_1 and V_3 represent respectively, the midpoint potential at which the calcium and potassium currents are half activated, and V_2 and V_4 represent the respective slopes of activation for the calcium and potassium currents. In the ML model τ_W is a time constant with respect to the K^+-channel given by

$$\tau_W = \tau_0 \operatorname{sech}\left(\frac{V - V_3}{2V_4}\right). \tag{1.6}$$

Here τ_0 is the time scale of the recovery process. As can be seen from the bifurcation diagram displayed in Figure 1.2, the ML model shares many of the features of the more complicated HH Model and so is a popular choice for modelling cortical neurons [49, 50, 51].

Integrate and fire models

Conductance based models such as the ones discussed above incorporate detailed information about the electrical properties of the neuron and so can be used to try and understand the mechanisms underlying action-potential generation [52]; however, any attempt to understand the behaviour of large numbers of coupled neurons requires a much simpler approach. To that end, the *integrate-and-fire model* has had considerable success [37, 53, 54]. It is a reduced model of a neuron, in that it describes the membrane dynamics using typically one or two equations, and so is much more amenable to larger network studies. Here we consider the *leaky integrate and fire model* although many other variations exist [12].

The basic circuit of the leaky integrate and fire model consists of a capacitor C running alongside a resistor R with a driving current I [55]; see Figure 1.3. The current $I(t)$ is split into two components: I_R, the current passing through the resistor and I_C, the current that charges the capacitor, and so we can write the current as $I(t) = I_R + I_C$. Using Ohm's law, $I = u/R$, and the definition of capacity, $C = q/u$, we can write,

$$I_R = \frac{u}{R}$$

and

$$I_C = C\frac{\mathrm{d}u}{\mathrm{d}t}.$$

Here u is the voltage across the resistor R, and q is the charge of the capacitor C. Using these two properties we can write

$$I(t) = I_R + I_C = \frac{u(t)}{R} + C\frac{\mathrm{d}u}{\mathrm{d}t}.$$

Setting $\tau = RC$ we can then rearrange the above to obtain

$$\tau\frac{\mathrm{d}u}{\mathrm{d}t} = -u(t) + RI(t). \tag{1.7}$$

We call τ the *membrane time constant* of the neuron.

This model as well as other threshold models [56, 57, 58, 59, 60] have been used extensively to study networks of spiking neurons [61, 62, 63, 64]. For example, a model composed of integrate and fire neurons labelled by $i = 1, .., n$, can be written as follows

$$\tau\frac{\mathrm{d}u_i}{\mathrm{d}t} = -u_i(t) + R\sum_{j=1}^{n} a_{ij}\sum_{m}\alpha(t - T_j^m), \quad i = 1, \ldots, n, \tag{1.8}$$

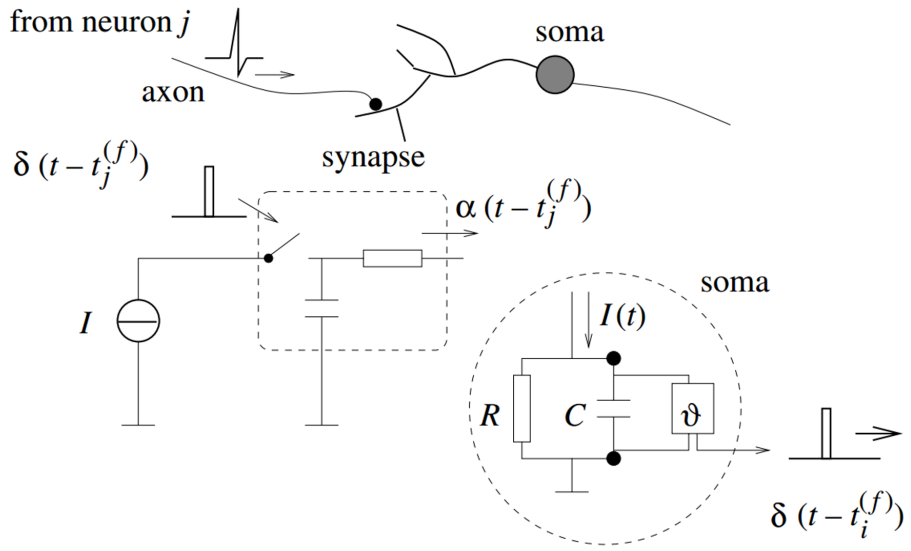


Figure 1.3: Schematic diagram of the integrate and fire model taken from [37].

where T_j^m is the *mth* firing time of the *jth* neuron. The connectivity structure is given by the *network adjacency matrix* $A = (a_{ij})$ and $\alpha(s) = \frac{1}{\tau_s}\exp(-\frac{s}{\tau_s})$ is a simple exponential decay for $s > 0$ and τ_s time constant [56].

1.1.2 Population models

Many areas of the brain are organised into units containing thousands of neurons that are similar from a structural *and* functional point of view [65, 66, 67]. For this reason, as well as the meso-scale resolution of most modern neuroimaging technologies, so-called *population models*, such as that introduced in the seminal work of Hugh Wilson and Jack Cowan in the early 1970's [18, 19], have become increasingly popular.

The Wilson-Cowan model

Here, we give a brief description of the model due to Wilson and Cowan [18, 19]. This model considers a population consisting of two types of neuron: *excitatory* and *inhibitory*. Here, we present an adaptation of the original equations taken from [68], in which case average neural activity for the two populations is described by the

following pair of coupled ordinary differential equations (ODEs):

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= -u + S(c_1 u - c_2 v + P), \\ \frac{\mathrm{d}v}{\mathrm{d}t} &= -v + S(c_3 u - c_4 v + Q). \end{aligned} \tag{1.9}$$

Here u denotes activity levels of excitatory neurons and v activity levels of inhibitory neurons. The parameters P and Q represent external inputs to the excitatory and inhibitory populations respectively and $c_1, ..., c_4$ are the connectivity coefficients which give the average number of excitatory or inhibitory synapses. The response function, S, measures the fraction of cells that exceed the threshold value, and usually takes the form of a sigmoid [69]. Figure 1.4 shows the bifurcation diagram of (1.9) as a function of P and Q for

$$S(x) = \frac{1}{1 + \exp(-x)},$$

and parameters $c_1 = c_2 = c_3 = 10$ and $c_4 = -2$. Note that the dashed curve denotes the saddle-node bifurcation set, whilst the solid curve the Hopf bifurcation set.

Equations such as (1.9) can be deployed to model large scale cortical activity [18, 70, 71] by employing structural connectivity matrices derived via imaging technologies to construct network models, typically referred to as *neural mass models*, in order to simulate neural activity. For example, a network of coupled Wilson-Cowan nodes can be constructed as follows:

$$\begin{aligned} \frac{\mathrm{d}u_i}{\mathrm{d}t} &= -u_i + S(c_1 u - c_2 v + P + \epsilon \sum_j w_{i,j} u_j), \\ \frac{\mathrm{d}v_i}{\mathrm{d}t} &= -v_i + S(c_3 u - c_4 v + Q), \qquad i = 1, \dots n. \end{aligned} \tag{1.10}$$

Here ϵ denotes the strength of the coupling between the masses and $w_{i,j}$ the connectivity between the masses. Despite its relative simplicity, the network model in (1.10) has recently been used to measure the extent to which structural connectivity constrains functional connectivity using both network and multiplex network techniques [72, 73].

The Jansen-Ritt model

Another popular neural mass model that has been successfully used to model large-scale neural activity, in both healthy and diseased brains, is that of Jansen and Ritt [74, 75]. It models each mass using three interconnected neural populations: one for

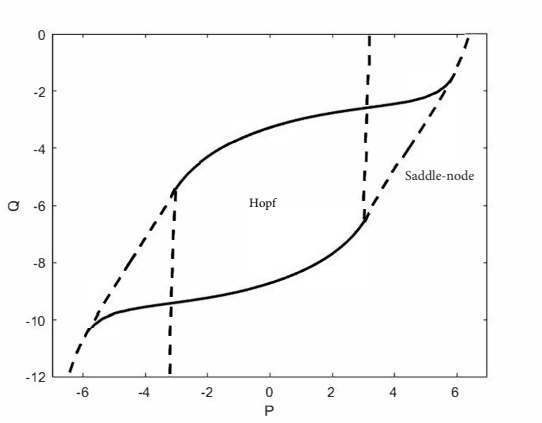


Figure 1.4: Bifurcation set of a single Wilson-Cowan node for $c_1 = c_2 = c_3 = 10$ and $c_4 = -2$. The dashed curve denotes the saddle-node bifurcation set and the solid curve the Hopf bifurcation set. (Figure created using XPPAUT [1].)

pyramidal cells, and one each for excitatory and inhibitory interneurons. Activity in a Jansen-Ritt node is described by the following set of six equations:

$$\begin{aligned}
\frac{\mathrm{d}y_0}{\mathrm{d}t} &= y_3, \\
\frac{\mathrm{d}y_1}{\mathrm{d}t} &= y_4, \\
\frac{\mathrm{d}y_2}{\mathrm{d}t} &= y_5, \\
\frac{\mathrm{d}y_3}{\mathrm{d}t} &= Aaf(y_1 - y_2) - 2ay_3 - a^2 y_0, \\
\frac{\mathrm{d}y_4}{\mathrm{d}t} &= AaP + AaC_2 f(C1y_0) - 2ay_4 - a^2 y_1, \\
\frac{\mathrm{d}y_5}{\mathrm{d}t} &= BbC_4 f(C3y_0) - 2by_5 - b^2 y_2,
\end{aligned} \tag{1.11}$$

where

$$f(x) = \frac{\nu_{\max}}{1 + \exp\left(r(x_0 - x)\right)}.$$

Here, y_0, y_1 and y_2 denote respectively the average activity of the pyramidal neurons, and the excitatory and inhibitory interneurons. Note that as with the Wilson-Cowan model in the previous section one can add a coupling term and consider networks of Jansen-Rit nodes (the interested reader should see [76] and references therein).

1.1.3 Neural field models

In this thesis we are less interested in the discrete way of modelling neural activity, such as that described previously in this chapter, rather our concern is with the continuous approach that results from considering the limiting case (*i.e.* as the number of masses tends to infinity) of the neural mass models discussed above. This limiting case is known as a *neural field model* (NFM) and is the focus of the current study. Neural field models date back to the 1950's with Beurle [16] who first considered the activity of a mass of cells. By treating the activity statistically he formulated the following continuum description for the proportion of cells being active in a cortical region at time t:

$$\int_{-\infty}^{\infty} F(X,t)\xi(x-X)dX. \tag{1.12}$$

This convolution gives the mean rate of impulses arriving at cells from all other cells. Here, F denotes the activity which, in this case, is only considered to change in the x-direction, and ξ denotes the distance between cells.

Following this approach, there were many studies on neural networks that replaced these descrete network architectures by a suitable continuous approximation; see, for example, Griffith [77, 78] in the 1960's and Amari [79], Wilson and Cowan [19] in the 1970's. However, it is the work by Wilson and Cowan along with that of Amari [3] and Nunez [80] that provides the starting point for the formulation of the neural field models that are currently in common use, that is the familiar partial integro-differential equation models of the form

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = -u(\mathbf{x},t) + \int_{\Omega} w(\mathbf{x},\mathbf{x}')S(u(\mathbf{x}'))\mathrm{d}\Omega(\mathbf{x}'). \tag{1.13}$$

Here u denotes neural activity, Ω the domain under consideration, the kernel w is the connectivity between neural units and S is the firing rate function that typically takes the form of a sigmoid. Such models are usually solved on either the real line, $\Omega = \mathbb{R}$, or in the plane, $\Omega = \mathbb{R}^2$, [81, 82] using standard techniques that either transform the NFM into an 'equivalent' PDE [83, 84], or solve directly using Fourier transforms and the convolution theorem [81, 85]. Here, however, we consider the extension of such models to non-flat domains, the motivation of which is to be able to solve these non-local equations directly on curved cortical-like geometries.

1.2 Thesis overview

In this thesis we employ a novel approach to solve neural field models (NFMs) on curved cortical-like geometries. We propose the use of collocation techniques to solve NFMs on general triangulated surfaces such as those obtained from neuroimaging data. In order to test our methods, we compare solutions found when implementing linear collocation to those found by more standard methods, on both the planar periodic square and curved surface of a torus. We find that collocation techniques are capable of replicating solutions found by more standard methods regardless of the underlying mesh. Building on these results we move on to consider NFMs on the cortical geometry of the rat brain and investigate the effect that geometry has on solutions. In particular, we find evidence to suggest that the curvature of the rat cortex has a considerable effect on the propagation speed and path of solutions exhibited by the NFM, thus suggesting that current methods that do not take into account the folded structure of the cortical surface potentially miss vital details.

We start in Chapter 2 by providing an overview of some of the fundamental properties of NFMs, the techniques employed to solve them and the types of solutions they are capable of producing. This includes the background essential to the derivation of NFMs as well as a brief discussion of their existence and uniqueness properties. Both numerical and analytical methods for solving NFMs are discussed, with specific focus on those techniques used to compare against the solutions obtained using linear collocation in this thesis. Following this introduction to NFMs, in Chapter 3 we provide relevant background details of the methods implemented throughout the thesis. We provide an overview of the collocation method and how it is implemented on triangulated domains for solving NFMs. In this thesis we consider a variety of triangulated domains including a flat periodic square, the curved surface of the torus and the folded geometry of the rat cortex, and so in this section we provide some details of the mesh generation/refinement procedures used. We consider two different numerical methods for computing geodesic distances and test their performance on the sphere since geodesics can be computed analytically in that case. To conclude the chapter, we present a brief introduction to numerical bifurcation theory, focussing on the pseudo-arclength method of continuation which is deployed in this thesis.

In Chapter 4 we implement the techniques described in the aforementioned chapters and consider the first of two NFMs, the Amari equation. The Amari equation is perhaps the simplest NFM of the form given by (1.13) and as such is an ideal test case for the methods forwarded in this thesis; in particular, we shall consider

stationary bump solutions of the Amari equation on both a flat periodic domain and the curved surface of the torus. We present numerical results, including an error analysis of the integral term in the NFM, which is the main source of error in the model, as we vary both mesh coarseness and regularity. To further illustrate our approach we solve the Amari equation using the trapezoidal method and linear collocation on both domains of interest as well as FFTs in the case of a periodic square; moreover, we performed a numerical bifurcation analysis in order to determine (a) how system parameters affect the observed solutions; and (b) the extent to which curvature changed these relations, when moving from the planar domain to the torus. In particular, we found that whilst solutions remained qualitatively similar for the different curvatures we considered, the overall strength of neural activity varied considerably.

In Chapter 5 we consider the second NFM studied in this thesis, which is an example of a so-called adaptive neural field model. This model is an extension of the Amari equation that includes an additional recovery variable and is therefore capable of producing travelling solutions as well as stationary solutions. To begin, we consider solutions of the extended model on a periodic, square domain and perform both an error and bifurcation analysis of travelling bump solutions. We then consider travelling bump solutions on two different curved geometries: the torus and the cortical surface of the rat brain. In the non-planar case we can no longer perform a bifurcation analysis since travelling bump solutions propagate at non-constant speeds due to curvature effects. Instead we investigate the relationship between propagation speed and curvature of localised bump solutions in order to highlight the extent to which the folded structure of the cortex influences mechanisms of spreading activity.

We conclude in Chapter 6 with a summary of the work presented in this thesis as well as discussing possible directions for future work.

1.3 Publications and presentations

The material presented in Chapter 4 and Chapter 5 has been written as an article:

1. *A numerical simulation of neural fields on curved geometries.*, R Martin, D J Chappell, N Chuzhanova and J J Crofts, Journal of computational neuroscience, 45(2), 133–145.

Some of the material presented in Chapter 4 and Chapter 5 has been written as two separate conference proceedings

2. *Collocation methods for solving two-dimensional neural field models on complex triangulated domains*, R Martin, D J Chappell, N Chuzhanova and J J Crofts, in Proceedings of the International conference on Integral Methods in Science and Engineering, Volume 2, 2017, 169–178.

3. *Can linear collocation ever beat quadratic?* R Martin, D J Chappell, N Chuzhanova and J J Crofts, in Proceedings of the 11th UK Conference on Boundary Integral Methods (UKBIM 11), 2017, 117–124.

In addition, the material in Chapter 5 was first given as a presentation entitled 'Collocation methods for solving two-dimensional neural field models on complex triangulated domains' at Nottingham Trent University, School of Science and Technology, 10^{th} Annual Research Conference in May 2016. The talk was then extended and presented as both a poster and contributed talk at the European Conference on Mathematical and Theoretical Biology (ECMTB), Nottingham, July 2016 and as a contributed talk at the International Conference on Integral Methods in Science and Engineering (IMSE), Padova, Italy, July 2016.

The material presented in Chapter 4 was given as a presentation entitled 'Collocation methods for solving neural field models on complex triangulated domains' at 11^{th} UK conference on Boundary Integral Methods (UKBIM), Nottingham, UK, July 2017.

Chapter II

Neural field models

The main focus of this thesis is the mathematical modelling of large populations of neurons. Modelling at this larger scale has become increasingly popular as it reduces the high complexity of neuronal interactions to simpler population properties that are easier to analyse [86]. One approach of modelling at the macro-scale, is neural field theory. Neural field theory employs a continuum approach to model the activity of large populations of neurons in the cortex, the foundations of which were laid in the 1970s by Wilson and Cowan [18] and Amari [3]. These techniques are becoming an increasingly popular and effective in neuroscience, and have been applied to research the structure, function and dynamics of the brain [82, 87].

In this chapter we discuss neural field models (NFM) in more detail. We start by providing a heuristic derivation of the standard partial integro-differential form of a NFM, which is given by

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = -u(\mathbf{x},t) + \int_{\Omega} w(\mathbf{x},\mathbf{x}')S(u(\mathbf{x}'),t)\mathrm{d}\Omega(\mathbf{x}'), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^n, t > 0, \tag{2.1}$$

for $n = 1, 2, ...$ and with initial condition

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega.$$

This equation, which is commonly referred to as the *Amari equation*, can be derived by considering the $N \to \infty$ limit of a certain firing rate network model, much like the ones discussed in the previous chapter. In the following we present a derivation of (2.1) (in the 1D case) adapted from the paper by Bressloff [86]. We then briefly consider the necessary mathematical assumptions for a NFM such as (2.1) to be well-posed, before considering the current state-of-the-art methodology for solving NFMs.

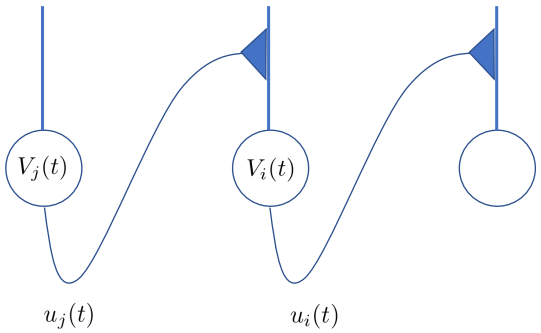


Figure 2.1: A pair of synaptically coupled neurons

2.1 Firing rate models

While analysing large networks of interconnected spiking neurons is possible [88, 89, 90, 91, 92], there is a much simpler approach for analysing large ensembles of neurons that replaces individual spike dynamics by so-called firing rates. Firing rate models study the population dynamics and firing rates of large populations of neurons [93]. Following [86, 94, 95], in this section we show how a spiking network model can be reduced to a firing rate network model.

Suppose that we have a pair of neurons connected by a synapse as shown in Figure 2.1. Then the net synaptic current received by neuron i at time t, due to the spike train emitted by neuron j, is given by

$$\sum_m \Phi_{ij}(t - T_j^m).$$

Here, T_j^m, denote the firing times of neuron j and $\Phi_{ij}(t)$ represents the temporal filtering effects due to dendritic and synaptic processing [86, 96, 97]. Considering a network of such neurons synaptically coupled as described above, and such that the synaptic inputs sum linearly, we can write down the following equation for the total synaptic input, $u_i(t)$, to the ith neuron:

$$u_i(t) = \sum_{j=1}^{N} \sum_m \Phi_{ij}(t - T_j^m) = \sum_{j=1}^{N} \int_{-\infty}^{\infty} \Phi_{ij}(\tau) a_j(t - \tau) \mathrm{d}\tau, \tag{2.2}$$

where

$$a_j(t) = \sum_m \delta(t - T_j^m)$$

is the *neural response function* [2].

To determine a closed set of equations, the firing times, T_j^m, must be determined.

With this in mind, we introduce the following threshold condition

$$T_j^m = \inf\{t, t > T_j^{m-1} | V_j(t) = \kappa, \dot{V}_j(t) > 0\}, \tag{2.3}$$

where κ is the firing threshold and $V_i(t)$ denotes the somatic membrane potential, which evolves according to the conductance-based model

$$C\frac{\mathrm{d}V_i}{\mathrm{d}t} = -I_{\mathrm{c},i}(V_i, \ldots) + u_i, \tag{2.4}$$

where $I_{\mathrm{c},i}$ denotes the membrane current. The potential $V_i(t)$ also satisfies the usual differential equations governing the ionic gating variables, see for example, the HH equations in (1.1). Now, assuming that synapses in the network are sufficiently slow, we can perform a temporal averaging of Equation (2.2) to obtain

$$\begin{aligned}
\langle u_i(t) \rangle &= \frac{1}{r}\int_{t-r}^{t} \left(\sum_{j=1}^{N} \int_{-\infty}^{\infty} \Phi_{ij}(\tau) a_j(t' - \tau) \mathrm{d}\tau \right) \mathrm{d}t' \\
&= \sum_{j=1}^{N} \int_{-\infty}^{\infty} \Phi_{ij}(\tau) \left(\frac{1}{r}\int_{t-r}^{t} a_j(t' - \tau) \mathrm{d}t' \right) \mathrm{d}\tau \\
&= \sum_{j=1}^{N} \int_{-\infty}^{\infty} \Phi_{ij}(\tau) r_j(t - \tau) \mathrm{d}\tau.
\end{aligned}$$

Here, we have introduced the firing rate function $r_j(t) = \frac{1}{r}\int_{t-r}^{t} a_j(t')\mathrm{d}t'$, of the jth neuron. For a slowly varying total synaptic current, $u_i(t)$ (compared to the membrane potential dynamics given by (2.4)), we have that $\langle u_i(t) \rangle \approx u_i(t)$. To obtain a closed system for $u_i(t)$ it is typically assumed that the firing rate function can be written as a function of the total synaptic current, that is $r_j(t) = S(u_j(t))$, giving

$$u_i(t) = \sum_{j=1}^{N} \int_{-\infty}^{\infty} \Phi_{ij}(\tau) S(u_j(t - \tau)) \mathrm{d}\tau. \tag{2.5}$$

In practice, the function S usually takes the form of a sigmoid function [18, 98], *i.e.*

$$S(t) = \frac{1}{1 + e^{-\beta(t-h)}}. \tag{2.6}$$

Here, β is termed the steepness parameter and h the firing rate threshold [81].

To solve Equation (2.5) for the total synaptic current, $u_i(t)$, one typically makes a number of simplifying assumptions concerning the time dependence of $\Phi_{ij}(t)$, which

then enables us to reduce (2.5) to a system of differential equations. For example, if we assume that the postsynaptic potential always has the same shape, then we can write

$$\Phi_{ij}(t) = w_{ij}\phi_i(t). \tag{2.7}$$

Here, w_{ij} is the synaptic strength of the connection between neuron i and neuron j and $\phi_i(t)$ is the time course of the input, which depends on the properties of the postsynaptic neuron i. The shape of this time course usually takes the form of an exponential decay

$$\phi_i(t) = H(t)e^{-t/\tau_i},$$

where H is the Heaviside function with time constant τ. Equivalently ϕ_i satisfies the ODE

$$\tau_i \frac{\mathrm{d}\phi_i(t)}{\mathrm{d}t} + \phi_i(t) = \delta(t), \tag{2.8}$$

with initial condition $\phi_i(0) = 0$.

In order to obtain a system of differential equations for the total current, $u_i(t)$, we assume the above form for the postsynaptic potential, and differentiate (2.5) to obtain

$$\begin{aligned}
\frac{\mathrm{d}u_i}{\mathrm{d}t} &= \sum_{j=1}^{N} \int_{-\infty}^{\infty} w_{ij} \frac{\mathrm{d}\phi_i(t-\tau)}{\mathrm{d}t} S(u_j(\tau))\mathrm{d}\tau, \\
&= \frac{1}{\tau_i} \sum_{j=1}^{N} \int_{-\infty}^{\infty} w_{ij} \left[\delta(t-\tau) - \phi_i(t-\tau)\right] S(u_j(\tau))\mathrm{d}\tau, \\
&= \frac{1}{\tau_i} \left(\sum_{j=1}^{N} \int_{-\infty}^{\infty} w_{ij} S(u_j(\tau))\delta(t-\tau)\mathrm{d}\tau - \sum_{j=1}^{N} \int_{-\infty}^{\infty} w_{ij}\phi_i(t-\tau) S(u_j(\tau))\mathrm{d}\tau \right), \\
&= \frac{1}{\tau_i} \left(\sum_{j=1}^{N} w_{ij} S(u_j(t)) - u_i(t) \right).
\end{aligned}$$

Note that in the above we have used the properties of convolution to move the delay, τ, into the connectivity function, w. Rearranging the above gives

$$\tau_i \frac{\mathrm{d}u_i}{\mathrm{d}t} + u_i = \sum_{j=1}^{N} w_{ij} S(u_j(t)), \tag{2.9}$$

which is a firing rate model that can be solved to find $u_i(t)$, the population activity at each node. Importantly, in the continuum limit the above firing rate model approaches a neural field model as described in the next section.

2.1.1 Deriving the neural field model

Neural field models are built from network models [99], such as the model in Equation (2.9). Instead of considering a discrete number of neural entities, we replace the discrete topology with a continuous one, for example the real line $\mathbb{R}$ or the plane $\mathbb{R}^2$. The sum is replaced by an integral and each point $x \in \mathbb{R}$ is assigned a population consisting of infinitely many neurons [93]. The connections w_{ij} are replaced by the continuous connectivity function $w(x, x')$, which describes how neurons positioned at x' interact with neighbouring neurons at position x [81]. Individual firing rates are replaced by the firing rate function S, which converts population activity to firing frequency. Denoting average population activity by $u(x, t)$ in (2.9) and replacing the sum by the appropriate integral results in the one dimensional neural field model,

$$\tau \frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int_{-\infty}^{\infty} w(x,x')S(u(x',t))\mathrm{d}x'. \tag{2.10}$$

This is a heuristic approach [86] that provides a way to incorporate biophysical mechanisms into the model, such as synaptic and membrane time constants [100, 101, 102], spike frequency adaptation [103, 104, 105, 106] and axonal delays [93, 107, 108, 109]. There has been further progress in the area of deriving neural field models, see for example [110, 111, 112, 113, 114, 115, 116]. Indeed, in [99] Laing was able to obtain an exact derivation of a neural field model from an idealised network of theta neurons, providing that the network is infinite.

2.2 The well-posedness of the neural field model

This section is adapted from the paper by Potthast and Beim Graben [117]. We shall start by stating the assumptions imposed on the initial condition, $u(\mathbf{x}, 0)$, the connectivity function, w, and the firing rate function, S, necessary for the NFM in (2.10) to have a unique, global solution. We then proceed to outline the core elements of the proof. Note that, without loss of generality, we set the domain of interest to be $\Omega = \mathbb{R}^n$ with $n = 1, 2, ...$ for the remainder of this section. All spaces and norms used in this section are defined in Appendix A.

2.2.1 Assumptions

Initial condition

The initial condition $u(\mathbf{x}, 0)$ lies in the Banach space of bounded continuous functions, that is

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}) \in BC(\mathbb{R}^n).$$

The connectivity kernel

The synaptic integral kernel w should satisfy the following criteria:

(i) The connectivity kernel is absolutely integrable:

$$w(\mathbf{x}, \cdot) \in L^1(\mathbb{R}^n) \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

(ii) The connectivity kernel satisfies a Lipschitz condition with Lipschitz constant c_w:

$$||w(\mathbf{x}, \cdot) - w(\tilde{\mathbf{x}}, \cdot)||_{L^1(\mathbb{R}^n)} \leq c_w |\mathbf{x} - \tilde{\mathbf{x}}|, \quad \mathbf{x}, \tilde{\mathbf{x}} \in \mathbb{R}^n.$$

(iii) The connectivity kernel is suitably bounded:

$$|w(\mathbf{x}, \mathbf{x}')| \leq C_\infty \quad \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n \qquad \text{and} \qquad \sup_{\mathbf{x} \in \mathbb{R}^n} ||w(\mathbf{x}, \cdot)||_{L^1(\mathbb{R}^n)} \leq C_w,$$

where C_∞ and C_w are positive constants.

The firing rate function

Although the results below can be extended to non-smooth firing rate functions, we only consider smooth, bounded firing rate functions $S : \mathbb{R} \to \mathbb{R}$. In addition, we require that

$$S(\mathbb{R}) \subset [0, 1].$$

The sigmoid function (2.6) is an example of a function that satisfies the above criteria.

2.2.2 Volterra formulation

In addition to the above criteria, let us define the operators

$$(Fu)(\mathbf{x}, t) := \frac{1}{\tau} \left(-u(\mathbf{x}, t) + \int_{\mathbb{R}^n} w(\mathbf{x}, \mathbf{y}) S(u(\mathbf{y}, t)) \mathrm{d}\mathbf{y} \right), \ \mathbf{x} \in \mathbb{R}^n, t > 0, \qquad (2.11)$$

and

$$(Au)(\mathbf{x},t) := \int_0^t (Fu)(\mathbf{x},s)\mathrm{d}s,\ \ \mathbf{x}\in\mathbb{R}^n, t>0. \tag{2.12}$$

Clearly, we can use the above to reformulate the NFM (2.10) as

$$\frac{\partial u}{\partial t} = Fu. \tag{2.13}$$

After integration, the above can be rewritten as a *Volterra integral equation* [118]:

$$u(\mathbf{x},t) = u(\mathbf{x},0) + \int_0^t (Fu)(\mathbf{x},s)\mathrm{d}s \tag{2.14}$$

$$= u(\mathbf{x},0) + (Au)(\mathbf{x},t). \tag{2.15}$$

The usefulness of the above is immediate given the following result from [117].

LEMMA 2.1. *The Volterra integral equation (2.14) or (2.15), respectively, is solvable on* $\mathbb{R}^n \times (0,\rho)$ *for some* $\rho > 0$ *if and only if the NFM (2.10) or (2.13), respectively, is solvable for* $\mathbf{x}\in\mathbb{R}^n$ *and* $t\in(0,\rho)$.

See ([117], Lemma 2.4) for a proof of the above lemma.

The next step towards a local existence and uniqueness result it to note that the operator A defined in (2.12) is a *contraction mapping*, that is, for $u_1, u_2 \in BC(\mathbb{R}^n \times [0,\rho])$, we have that

$$||Au_1 - Au_2||_\rho < \alpha||u_1 - u_2||_\rho,$$

for $\alpha\in(0,1)$. Then we have the following result.

THEOREM 2.1 (Local existence of solutions for a NFM). *Assume that the synaptic weight kernel* w *and the activation function* S *satisfy the conditions stated above, and let* $\rho > 0$ *be chosen such that*

$$\frac{\rho}{\tau}\left(1 + LC_w\right) < 1,$$

where L *is a Lipschitz constant for* S*. Then we obtain existence and uniqueness of solutions to the NFM (2.10) on* $[0,\rho]$.

The proof of the above theorem employs the Banach fixed point theorem [119] to determine the unique solution, u^*, of the equation

$$u = \tilde{A}u,$$

where $\tilde{A}u := u_0 + Au$ (see [117] for further details), thus proving the unique solvability of (2.15), and hence by the equivalence lemma stated above, also proving the unique solvability of (2.10).

In the last part of this section, we state without proof a global existence result for solutions of (2.10).

THEOREM 2.2 (Global existence of solutions for a NFM). *Under the conditions stated above, we obtain existence of global bounded solutions to (2.10).*

The interested reader should consult [117] for proofs and extensions of the above results. Note that in addition to the well-posedness of the problem, the authors in [117] also consider a thorough stability analysis, as well as considering non-smooth firing rate functions.

2.3 Techniques for solving neural field models

In this section we discuss the state-of-the-art in solving NFMs and discuss the types of solutions such models admit. The methods presented can be implemented for NFMs in 1D and 2D; however, for simplicity we focus on 1D NFMs here, giving only a brief treatment of 2D NFMs towards the end of the section.

2.3.1 PDE methods

PDE methods for solving neural field equations, or rather techniques that enable one to transform to an equivalent partial (or ordinary) differential equation (which typically still requires one to resort to numerical procedures to solve), have been employed on NFMs, but under quite severe restrictions [104, 120, 121, 122]. For example, connectivity kernels are typically restricted to be either homogeneous

$$w(x, x') = w(x - x'),$$

or isotropic

$$w(x, x') = w(|x - x'|).$$

In these cases, one can deploy special techniques that transform the NFM into an equivalent partial or ordinary differential equation. For example, consider the following one-dimensional NFM:

$$\frac{\partial u}{\partial t} = -u(x,t) + \int_{-\infty}^{\infty} w(x - x')S(u(x', t))\mathrm{d}x'.$$

By taking Fourier transforms of both sides of this equation (and using the convolution theorem) one can obtain the following equation

$$F[u_t + u] = F[w] \cdot F[S(u)],$$

in which $F[\cdot]$ denotes the Fourier transform.

Now, following [84], suppose that in the above we choose our connectivity kernel as

$$w(x) = e^{-b|x|}(b\sin(|x|) + \cos(x)).$$

In this case we can evaluate

$$F[w(x)](k) = \frac{4b(b^2+1)}{k^4 + 2k^2(b^2-1) + (b^2+1)^2},$$

where k is the transform variable. It follows that

$$F[u_t + u](k) = \frac{4b(b^2+1)}{k^4 + 2k^2(b^2-1) + (b^2+1)^2} F[S(u)](k).$$

Multiplying this equation by $k^4 + 2k^2(b^2-1) + (b^2+1)^2$ and taking inverse Fourier transforms then results in the following, equivalent PDE:

$$(u+u_t)_{xxxx} - 2(b^2-1)(u+u_t)_{xx} + (b^2+1)^2(u+u_t) = 4b(b^2+1)S(u).$$

The resulting PDE can be solved using standard numerical methods in order to determine solutions of the NFM. Note that the above idea can also be employed to study steady state solutions of the NFM, *i.e.* solutions of the equation

$$u(x) = \int_{-\infty}^{\infty} w(x-x')S(u(x'))\mathrm{d}x'. \tag{2.16}$$

In this case one can deploy Fourier transforms to obtain the ODE

$$\frac{\mathrm{d}^4 u}{\mathrm{d}x^4} - 2(b^2-1)\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + (b^2+1)^2 u = 4b(b^2+1)S(u),$$

which should be solved alongside the boundary conditions

$$\lim_{x\to\pm\infty}(u, u', u'', u''') = (0,0,0,0).$$

The above boundary conditions lead to localised solutions of the NFM – for further

details see [84].

2.3.2 Direct numerical methods

In the case when the neural field model cannot be reduced to an equivalent ODE or PDE, numerical methods can be applied directly to the NFM. In this section we review popular numerical methods for both solving and analysing solutions of NFMs, such as (2.1).

Numerical solutions of a 1D NFM

Here, we follow [81] and study the 1D NFM

$$\frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int_{\Omega} w(x-x')S(u(x',t))\mathrm{d}\Omega(x'), \tag{2.17}$$

on both a periodic domain (*i.e.* $\Omega = [-\pi, \pi]$) and the whole of the real line (*i.e.* $\Omega = \mathbb{R}$). More specifically, we investigate, respectively, steady state and travelling front solutions of these two different scenarios, as well as conducting a bifurcation analysis to study these solutions as important system parameters are varied (*e.g.* the firing threshold h) .

Suppose that Equation (2.17) is defined on a periodic domain (*i.e.* $x \in [-\pi, \pi]$), with S taking the usual sigmoidal form, such as that in Equation (2.6), and let the connectivity kernel be the Mexican-hat function

$$w(x) = 10e^{-4x^2} - 6e^{-x^2}. \tag{2.18}$$

Then we consider stationary patterns of (2.17) by setting the left-hand side (LHS) equal to zero, that is we solve the following equation:

$$-u + \int_{-\pi}^{\pi} w(x-x')S(u)\mathrm{d}x' = 0. \tag{2.19}$$

The solutions of both Equation (2.17) and Equation (2.19) are invariant under translations along the x axis, (*i.e.* if $u(x)$ is a solution of either equation, then so is $u(a+x), a \in \mathbb{R}$) resulting in an infinite family of solutions. A standard way to remove this degree of freedom is to restrict the search to even solutions, that is

solutions where $u(x) = u(-x)$. Thus, representing $u(x)$ as a Fourier series gives

$$u(x) = \frac{u_0}{2} + \sum_{i=1}^{\infty} u_i(t)\cos(ix). \tag{2.20}$$

Recall, that since u is an even function, its Fourier series consists of cosines only; additionally, since $w(x)$ is an even function we may expand it as a Fourier cosine series

$$w(x) = \frac{w_0}{2} + \sum_{i=1}^{\infty} w_i \cos(ix), \tag{2.21}$$

where the coefficients w_i are given by the formula

$$w_i = \frac{1}{\pi}\int_{-\pi}^{\pi} w(x)\cos(ix)\mathrm{d}x. \tag{2.22}$$

Now, substituting (2.20) and (2.21) into Equation (2.17), we obtain

$$\begin{aligned}\sum_{i=0}^{\infty} \frac{\mathrm{d}u_i}{\mathrm{d}t}\cos(ix) = \sum_{i=0}^{\infty} w_i \int_{-\pi}^{\pi} \cos(i(x-x'))S\left(\frac{u_0}{2} + \sum_{j=1}^{\infty} u_j\cos(jx')\right)\mathrm{d}x' \\ - \sum_{i=0}^{\infty} u_i\cos(ix).\end{aligned} \tag{2.23}$$

Using the identity $\cos(A-B) = \cos(A)\cos(B) + \sin(A)\sin(B)$, we can rewrite the first term on the right-hand side (RHS) of Equation (2.23) as

$$\begin{aligned}\sum_{i=1}^{\infty} w_i \int_{-\pi}^{\pi} \cos(i(x-x'))S(\alpha)\mathrm{d}x' &= \sum_{i=1}^{\infty} w_i \int_{-\pi}^{\pi} \cos(ix')S(\alpha)\mathrm{d}x'\cos(ix) \\ &\quad + \sum_{i=1}^{\infty} w_i \int_{-\pi}^{\pi} \sin(ix')S(\alpha)\mathrm{d}x'\sin(ix), \\ &= \sum_{i=1}^{\infty} w_i \int_{-\pi}^{\pi} \cos(ix')S(\alpha)\mathrm{d}x'\cos(ix).\end{aligned}$$

Here $\alpha = \frac{u_0}{2} + \sum_{j=1}^{\infty} u_j\cos(jx')$ and the second line follows since we are integrating an odd function over a symmetric interval (note, as $u(x,t)$ is even as a function of x it follows that $S(u)$ is also even with respect to x).

Putting all this together results in an infinite set of ordinary differential equations

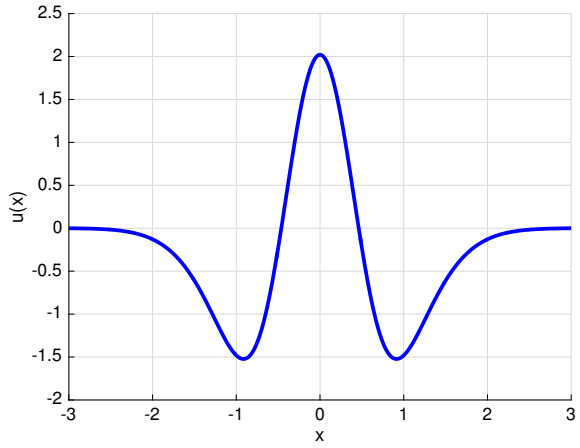


Figure 2.2: Steady state solution of (2.17) for $N = 15,\ h = 1/3,\ \beta = 20$.

describing the neural field:

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -u_i + w_i \int_{-\pi}^{\pi} \cos(ix')S\left(\frac{u_0}{2} + \sum_{j=1}^{\infty} u_j \cos(jx')\right) \mathrm{d}x', \quad \text{for } i = 0, 1, 2, ...$$

In practice, we can only solve finitely many equations and so we truncate to consider only the first N modes, that is we solve

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -u_i + w_i \int_{-\pi}^{\pi} \cos(ix')S\left(\frac{u_0}{2} + \sum_{j=1}^{N-1} u_j \cos(jx')\right) \mathrm{d}x', \tag{2.24}$$

for u_i with $i = 0, 1, 2, ..., N-1$.

It follows that the steady states of (2.17) satisfy the equations

$$-u_i + w_i \int_{-\pi}^{\pi} \cos(ix')S\left(\frac{u_0}{2} + \sum_{j=1}^{N-1} u_j \cos(jx')\right) \mathrm{d}x' = 0, \quad \text{for } i = 0, 1, 2, ...N-1. \tag{2.25}$$

This nonlinear system of N equations can be written as

$$F(\mathbf{v}, h) = 0, \tag{2.26}$$

with components given by

$$F_i = -u_i + w_i \int_{-\pi}^{\pi} \cos(ix')S\left(\frac{u_0}{2} + \sum_{j=1}^{N-1} u_j \cos(jx')\right) \mathrm{d}x', \tag{2.27}$$

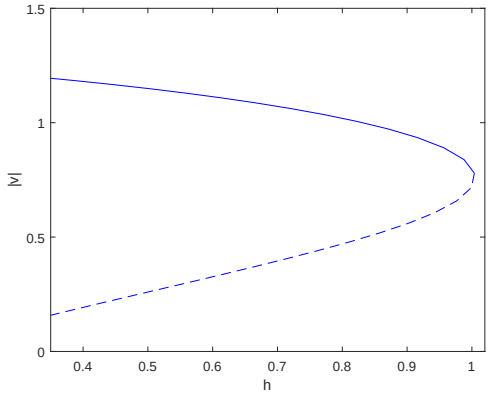


Figure 2.3: The norm of $\mathbf{v}$ versus the firing rate threshold, h.

for $i = 0, 1, 2, ..., N-1$.

To solve Equation (2.26) we use Newton's method [123]. Once we obtain a solution, $\mathbf{v}_0$ say, we can use numerical continuation to trace out a solution branch as we vary a parameter of interest [124]. In our example we vary the firing rate threshold, h, which controls the amount of neuronal activity converted to firing frequency. The level of activity of the neuronal population must be greater than the threshold value in order to 'fire'. Figure 2.2 shows an even solution of Equation (2.19) with firing threshold $h = 1/3$, and Figure 2.3 shows the result of repeating the procedure described above, tracing out the solution branch as the firing threshold h is varied – see Chapter 3 for a detailed discussion of the numerical continuation procedure used. Figure 2.3 shows how the magnitude of $\mathbf{v}$ varies as a function of the firing rate threshold h. The stable solutions are destroyed via a saddle-node bifurcation as h increases. In the above experiments, the parameters were set to $N = 15$ and $\beta = 20$, and all integrals were approximated using the trapezoidal rule [125].

To study moving patterns we consider the following NFM [81]:

$$\frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int_{-\infty}^{\infty} w(x-x')S(u(x',t))\mathrm{d}x', \tag{2.28}$$

which, unlike the previous example, is defined on the whole of the real-line. We choose the connectivity function to take the form

$$w(x) = \frac{1}{2}e^{|x|}.$$

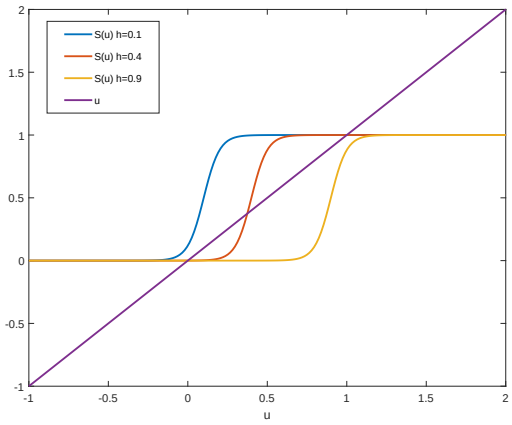


Figure 2.4: Solutions occur at the intersection of $S(u)$ and u.

Note that

$$\begin{aligned}\int_{-\infty}^{\infty} w(x)\mathrm{d}x &= \frac{1}{2}\int_{-\infty}^{\infty} e^{-|x|}\mathrm{d}x \\ &= 1,\end{aligned}$$

and so any spatially homogeneous steady state solution of (2.17) will satisfy

$$u = S(u). \tag{2.29}$$

As can be readily observed from Figure 2.4, varying h results in either one of three solutions of (2.29). Choosing a value of h for which there exist three solutions, $u_1 < u_2 < u_3$, and setting the initial condition in part of the domain to be $u = u_1$ with $u = u_3$ elsewhere, one can observe a travelling front going between the two steady states. Figure 2.5 shows the results of integrating such an initial condition forward in time. Here, the trapezoidal method was used for all integration, while the time-stepping was performed using Euler's method with time step $\Delta t = 0.1$. A truncated spatial domain of $[0, 50]$ was employed with $N = 200$ discretisation points. The parameters were chosen as $\beta = 20$ and $h = 0.47$.

To study this front and its properties as h is varied in the system we move to the travelling coordinate frame, $\xi = x - ct$, which allows us to write (2.28) as

$$\frac{\partial u(\xi,t)}{\partial t} = c\frac{\partial u(\xi,t)}{\partial \xi} - u(\xi,t) + \int_{-\infty}^{\infty} w(\xi - x')S(u(x',t))\mathrm{d}x'.$$

Steady state solutions of this equation map onto travelling front solutions of (2.28),

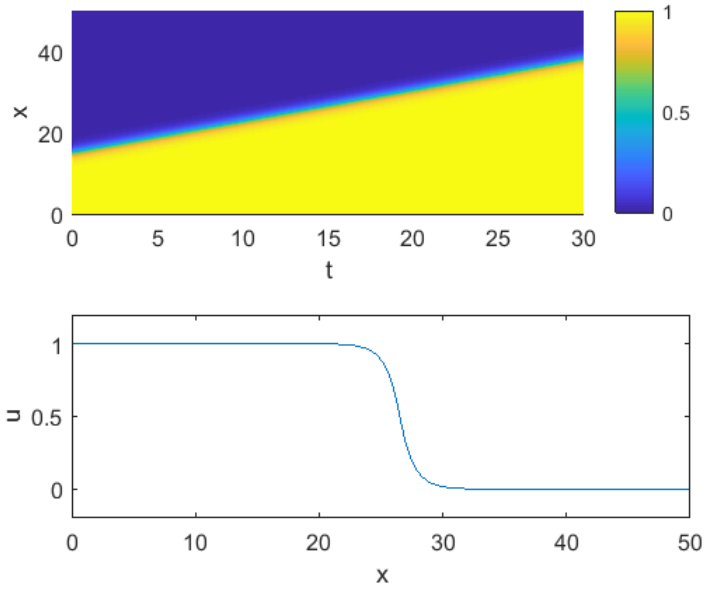


Figure 2.5: *Upper*: space-time plot of the solution of (2.17), where the domain has been truncated to $[0, 50]$. Parameters: $\beta = 20$ and $h = 0.47$ *Lower*: the solution at time $t = 15$.

and so we can solve the following system

$$c\frac{\mathrm{d}u}{\mathrm{d}\xi} - u + \int_{-\infty}^{\infty} w(\xi - x')S(u(x'))\mathrm{d}x' = 0. \tag{2.30}$$

To solve (2.30), we again discretise the domain into N equally spaced grid points and use the Newton-Armijo method to solve the resulting system in the form (2.26) The derivative term was approximated using forward finite difference formulae.

In this case $\mathbf{v}$ is of length $N + 1$, since we have an extra unknown (the speed c) together with the N equations for u. As stated previously, solutions of (2.28) are invariant under translations in x and thus to remove this extra degree of freedom, we introduce an extra equation (known as a pinning condition)

$$\int_{-\infty}^{\infty} (u - \hat{u})\hat{u}_{\xi}\mathrm{d}\xi = 0, \tag{2.31}$$

where $\hat{u}(\xi) = \frac{1}{2}(1 + \tanh(25 - \xi))$. Note that the pinning condition allows us to determine one of the infinite family of translation invariant solutions [81].

Equations (2.30) and (2.31) were then solved simultaneously using the Newton-Armijo method in order to determine an initial solution $\mathbf{v}_0$ from which to start the continuation procedure. Figure 2.6 shows a solution branch displaying the behaviour

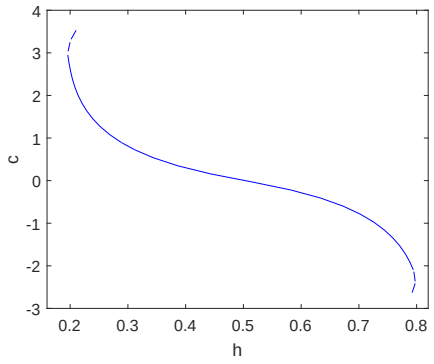


Figure 2.6: Branch of solutions with initial parameters $c_0 = 0.8$ and $h_0 = 0.3$.

of the front speed c as a function of the firing threshold h. Here the solid line shows stable solutions and the dashed line unstable solutions. We can see that for a range of values for h there exists a stable front with speeds varying from positive to negative. As h varies further from $h = 0.5$ the stable front becomes unstable via a saddle node bifurcation.

Fast Fourier transforms

When solving a NFM such as that in Equation (2.28), a popular alternative [81, 85] to the above approach (for integral equations of convolution type), is to deploy the fast Fourier transform (FFT) algorithm. Note that since the Fourier transform of a convolution is the product of Fourier transforms, we can rewrite the convolution integral in (2.28) as

$$\int_{-\infty}^{\infty} w(x - x')S(u(x', t))\mathrm{d}x' = F^{-1}\left[F[w] \cdot F[S]\right].$$

Substituting this expression into the NFM (2.28) gives

$$\frac{\partial u}{\partial t} = -u + F^{-1}\left[F[w] \cdot F[S]\right],$$

which can be solved efficiently using discrete Fourier transforms (via the FFT algorithm) to compute the RHS. The FFT algorithm is extremely efficient since it only requires $\mathcal{O}(N \log N)$ operations [126], where here N is the size of the spatial discretisation. We shall use FFTs for comparative purposes in our work where possible.

2.3.3 Numerical solution of a two dimensional NFM

The NFM described by Equation (2.17) can also be solved in two dimensions (as we shall see later); however, at this point we introduce an extended NFM that includes a so-called recovery variable and thus admits travelling bump solutions, which shall be crucial in the work to come. Following [81] we introduce this two-dimensional NFM on the periodic square $\Omega = [-L, L]^2$:

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} &= A\int_{-L}^{L}\int_{-L}^{L} w(x-x', y-y')S(u(x',y',t))\mathrm{d}x'\mathrm{d}y' \\ &\quad - u(x,y,t) - a(x,y,t), \\ \tau\frac{\partial a(x,y,t)}{\partial t} &= Bu(x,y,t) - a(x,y,t). \end{aligned} \tag{2.32}$$

This NFM is known as an adaptive neural field model since it includes an additional recovery variable a, which acts to repolarize u via negative feedback. Furthermore the parameters B and τ are related to the sensitivities and time-scale of the problem [81]. Later in the thesis we study (2.32) and so for comparative purposes here we investigate travelling bump solutions using standard techniques, and conduct a bifurcation analysis to study steady state solutions as system parameters are varied, in this case the sensitivity, A.

The firing rate function S takes the form of the usual sigmoid function and the connectivity kernel is given by a Mexican-hat function

$$w(x,y) = e^{-(x^2+y^2)} - 0.17e^{-0.2(x^2+y^2)}. \tag{2.33}$$

To find solutions for u and a, Equation (2.32) is integrated for $t \in [0, 250]$ using Euler's method with time step $\Delta t = 0.2$. As outlined above, Fast Fourier transforms are used to evaluate the integral, resulting in the system

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} &= A(F^{-1}\left[F[w]\cdot F[S]\right] - u(x,y,t) - a(x,y,t), \\ \frac{\partial a(x,y,t)}{\partial t} &= \frac{1}{\tau}(Bu(x,y,t) - a(x,y,t)). \end{aligned}$$

The domain is discretised into $N = 256 \times 256$ grid points and the parameters are set as follows: $L = 7.5, \quad A = 2, \quad \beta = 5, \quad h = 0.8, \quad B = 0.4, \quad \tau = 3$. Figure 2.7 shows the travelling bump solutions at $t = 250$.

A bifurcation analysis is conducted in order to study steady state solutions as

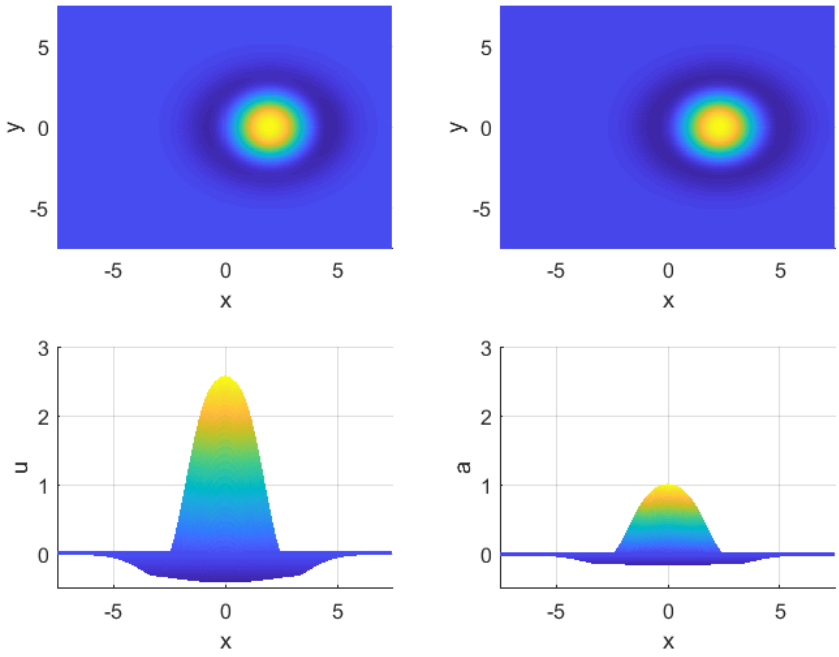


Figure 2.7: Snap shot of the travelling bump solution at $t = 250$. *Left column* u, *right column* a. Parameters: $L = 7.5, \quad A = 2, \quad \beta = 5, \quad h = 0.8, \quad B = 0.4, \quad \tau = 3$

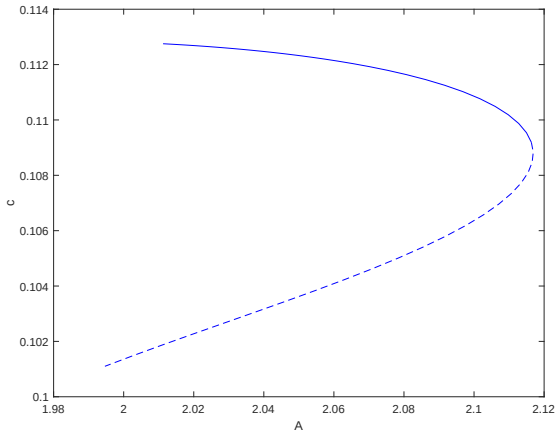


Figure 2.8: Speed of the bump c, as a function of the sensitivity A. Parameters: $N = 256, \quad L = 7.5, \quad \beta = 5, \quad h = 0.8, \quad B = 0.4, \quad \tau = 3$

the parameter A is varied in the system. Moving to the travelling coordinate frame $\xi = x - ct$, means we can write the steady state equations as

$$\begin{aligned} 0 &= A\int_{-L}^{L}\int_{-L}^{L} w(\xi - x', y - y')S(u(x', y'))\mathrm{d}x'\mathrm{d}y' \\ &\quad - u(\xi, y) - a(\xi, y) - c\frac{\partial u(\xi, y)}{\partial \xi}, \\ 0 &= Bu(\xi, y) - a(\xi, y) - c\tau\frac{\partial a(\xi, y)}{\partial \xi}. \end{aligned} \tag{2.34}$$

To solve (2.34) we discretise the domain into N equally spaced grid points, which results in a system of the form

$$F(\mathbf{v}, A) = 0.$$

Here the vector $\mathbf{v}$ is of length $2N + 1$, and contains the N unknowns corresponding to the values of u at the grid points, the corresponding N unknowns for a and the speed c. Any solution of (2.32) is invariant under translations in both the x and y axis, therefore we must enforce two conditions to remove these degrees of freedom. We introduce the pinning condition

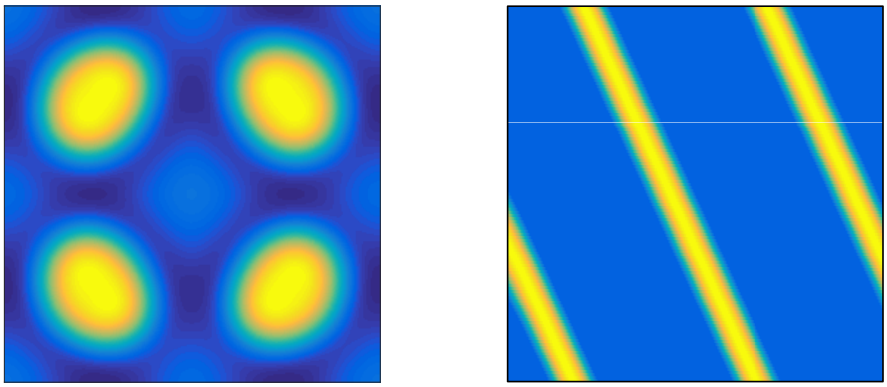
$$u(0, 0) - \frac{1}{2L}\int_{-L}^{L} u(\xi, 0)\mathrm{d}\xi = 0, \tag{2.35}$$

and restrict the solutions to be symmetric about $y = 0$. These conditions are the same as those in [81].

Equations (2.34) and (2.35) were solved simultaneously using a Newton-Krylov solver, which is a Jacobian free Newton solver (see Chapter 3 for more detail). Figure (2.8) shows a solution branch displaying the behaviour of the speed c as a function of the sensitivity A. Again the dashed line denotes stable solutions and the solid line stable. As A is varied the stable bump becomes unstable via a saddle-node bifurcation.

2.4 Solutions to NFMs and pattern formation

Neural field models are of great interest, not only from a mathematical point of view, but also from an experimental neuroscience point of view, since they can replicate many of the dynamic patterns of brain activity that are observed using modern neuroimaging methodologies [24]. They are used for interpreting (and unifying) electroencephalogram (EEG) data, functional magnetic resonance (fMRI) data and



(a) (b)

Figure 2.9: (a) Multiple bump solution. (b) Travelling wave solution.

magnetoencephalography (MEG) data [127, 128].

The principles behind pattern forming systems are well known, see for example, [129, 130]. Neural field models are also capable of supporting a variety of pattern formations in both one and two dimensional space. Examples in one dimensional space include solitary waves (or travelling pulses), stationary pulses and spatially periodic patterns [19, 131]. These patterns are also observed in two dimensions; however, other interesting patterns can arise, such as spiral waves [132, 133, 134] and target waves [135], as well as drifting spots [134] and breathers [136]. Figure 2.9 gives an example of multiple bump and travelling wave patterns that can be observed in two dimensions.

Importantly, different pattern formations can be linked to different neurological phenomena. For example, wave fronts and pulses have been observed in a number of slice preparations [137, 138, 139, 140, 141, 142], solitary pulses (bumps) have been observed in models of working memory [121, 143, 144] and spiral waves are believed to be linked to the generation of visual hallucinations [145, 146, 147, 148]. Our primary goal in this work is to construct more physiologically realistic models of mesoscopic neurodynamic cognitive phenomena, including, for example, curvature information and/or experimentally informed connectivity data, thus allowing us to better understand the types of waves, as well as mechanisms of synchrony and propagation through the brain, that are considered the signature of a range of neurological disorders [149, 150, 151, 152]. Indeed, such levels of detail are likely mandatory if we are to further improve our understanding of both healthy and diseased brains [153, 154].

2.5 Summary

In this chapter we have provided an overview of the current state-of-the-art in neural field modelling. We began by providing a heuristic derivation of the Amari equation, which is the most popular such model, before stating the assumptions under which such equations are well-posed. We then discussed some current techniques available for solving such equations on planar domains and considered a number of different solution types, including localised bump and travelling wave solutions. Our focus was on those methods that can be used to perform comparative analyses against our methods later on in this thesis as these will be important for validating our methods where possible.

Chapter III

Technical preliminaries

In this chapter we provide details of the most important numerical techniques deployed in this thesis. We begin in §3.1 by providing a brief overview of projection method focussing on Galerkin and collocation techniques . In §3.2 we discuss the collocation method in more detail, since we use it to solve the neural field models (NFMs) (see, for example, Equation (2.1)) introduced in the previous chapter. In §3.3 we discuss the construction of triangulated domains on both flat and curved geometries. Here, we also consider the differences that arise when moving from flat to non-flat geometries and conduct an analysis comparing two different numerical methods for computing geodesic distances. Finally, in §3.5 we provide a brief review of numerical bifurcation theory, focussing on the so-called *pseudo-arclength continuation* technique, which is used in later chapters to study the behaviour of both fixed and travelling bump solutions as certain model parameters are varied.

3.1 Nonlinear integral equations

In this section we provide a brief overview of numerical methods for calculating fixed points of a nonlinear integral operator of the form

$$x = \mathcal{K}(x), \qquad x \in X \tag{3.1}$$

where $\mathcal{K}$ is a nonlinear integral operator and X is an appropriately defined function space [155]. In particular, we focus our attention on projection methods; the interested reader should consult the text by Atkinson [118] for details on related methods (*e.g.* Nystöm methods). Such equations arise naturally in many areas of biology as steady state equations for related integro-differential equation models [156, 157]

and the discussed methods will allow us to compute both steady state and travelling wave solutions of the NFMs considered in this work.

3.1.1 Projection methods

Projection methods are a family of techniques that can be applied to determine numerical approximations of integral equations such as that in (3.1). The description given below has been ammended from the survey by Atkinson [155].

Let $\mathcal{X}$ be a Banach space and let $\mathcal{X}_n, n = 1, 2, 3, \ldots$, be a sequence of finite dimensional subspaces used to approximate x^*, the fixed point solution of (3.1) (*i.e.* x^* solves the equation $x^* = \mathcal{K}(x^*)$). Then a projection method amounts to solving the equation

$$x_n = P_n \mathcal{K}(x_n), \tag{3.2}$$

for some bounded projection operator $P_n : \mathcal{X} \to \mathcal{X}_n$. It is typically assumed that

$$P_n x \to x \text{ as } n \to \infty, \quad x \in \mathcal{X}.$$

Now, assuming that P_n can be written as

$$P_n x = \sum_{j=1}^{n} l_j(x)\phi_j, \quad x \in \mathcal{X}$$

with $\{\phi_1, \ldots, \phi_n\}$ a basis of $\mathcal{X}_n$ and $\{l_1, \ldots, l_n\}$ a set of bounded linear functionals that are independent over $\mathcal{X}_n$. Then setting

$$x_n = \sum_{j=1}^{n} \alpha_j \phi_j,$$

we can reduce (3.2) to a finite nonlinear system of equations

$$\sum_{j=1}^{n} \alpha_j \phi_j = l_j \left(\mathcal{K} \left(\sum_{j=1}^{n} \alpha_j \phi_j \right) \right), \quad i = 1, \ldots, n. \tag{3.3}$$

The choice of the basis functions $\{\phi_1, \ldots, \phi_n\}$ and the linear functionals $\{l_1, \ldots, l_n\}$ determine the type of method. Note that under rather general assumptions one can obtain the following error result for the general projection method described above

$$c_1 ||x^* - P_n x^*|| \leq ||x^* - x_n|| \leq c_2 ||x^* - P_n x^*||, \quad n \geq N, \tag{3.4}$$

for suitable constants c_1, c_2. Proof of the above result as well as additional details on convergence estimates for the different projection methods can be found in the text by Atkinson [118].

There are two main types of projection method used in the literature.

Galerkin methods

The Galerkin technique solves a weak formulation of Equation (3.2). Assuming that $\mathcal{X}$ is an inner product space, such as $L^2(\mathbb{R}^n)$, then we can define the projection $P_n x$ to be the orthogonal projection of x onto $\mathcal{X}_n$, based on the associated inner product. Thus

$$(P_n x, y) = (x, y), \quad \text{for all } y \in \mathcal{X}_n \tag{3.5}$$

with $(\cdot, \cdot)$ the inner product. Typically, L^2 and its associated inner product are used in applications.

Letting

$$x_n = \sum_{j=1}^{n} \alpha_j \phi_j$$

with $\{\phi_1, \ldots, \phi_n\}$ a basis of $\mathcal{X}_n$. Then we solve for the $\{\alpha_i\}$ using

$$\sum_{j=1}^{n} \alpha_j \left(\phi_i, \phi_j\right) = \left(\phi_i, \mathcal{K}\left(\sum_{j=1}^{n} \alpha_j \phi_j\right)\right), \quad i = 1, \ldots, n. \tag{3.6}$$

Importantly, solving in this way requires twice as many integrations due to the inner products that need to be computed.

Collocation methods

When implementing the collocation method the projection operator $P_n x$ is defined as the element that interpolates x at nodes $y_1, y_2, \ldots, y_n \in \mathcal{X}_n$. In order to determine x_n we solve the system

$$\sum_{j=1}^{n} \alpha_j \phi_j(t_i) = \mathcal{K}\left(\sum_{j=1}^{n} \alpha_j \phi_j\right)(t_i), \quad i = 1, \ldots, n. \tag{3.7}$$

Note that we implement the collocation technique in our work since, in general, it is more computationally efficient, having approximately half as many integrals to compute, and it is also more practical in the sense that it is easier to implement. Below we give specific details in the case of applying collocation to solve a NFM.

3.2 Solving integral equations on triangulated domains using collocation methods

In this section we follow the text by Atkinson [118] and give an overview of the methods we implement in order to solve neural field models on general triangulated domains.

3.2.1 Interpolation over triangles

The neural field models we consider are examples of multivariable integral equations and so we can implement the techniques outlined in [118] in order to solve them. Generally, when considering multivariable interpolation the domain under consideration is first broken up into smaller, simple regions. Here we consider triangulated domains and a polynomial interpolation is carried out over each triangle.

This interpolation is performed as follows. Let $\triangle_k$ denote a triangle from our domain and let $g(x, y)$ be a continuous function over $\triangle_k$. In order to approximate this continuous function, a polynomial interpolant $p(x, y)$ is introduced for $d \geq 0$, where d is the degree of the interpolant. We denote the interpolant as follows,

$$p(x, y) = \sum_{\substack{i,j \geq 0 \\ i+j \leq d}} c_{i,j} x^i y^j.$$

This interpolant has f_d degrees of freedom, where

$$f_d \equiv \frac{(d+1)(d+2)}{2}. \tag{3.8}$$

In order to determine the coefficients $c_{i,j}$ we must impose f_d interpolation conditions. These conditions require,

$$p(x_k, y_k) = g(x_k, y_k), \;\; k = 1, ..., f_d,$$

for a choice of f_d interpolation nodes on $\triangle_k$. To make the process more tractable we perform the interpolation over the unit simplex which we denote by σ, that is, a the triangle with vertices $(0, 0), (1, 0), (0, 1)$. More specifically, we employ the transformation $T_k : \sigma \to \triangle_k$, given by

$$(x, y) = T_k(r, s) = (1 - r - s)\mathbf{v}_{k,1} + s\mathbf{v}_{k,2} + r\mathbf{v}_{k,3}, \tag{3.9}$$

which maps each $\triangle_k$ on to σ. Here (r, s) are the coordinates defined on the unit simplex, σ, and the $\mathbf{v}_{k,j}$, $j = 1, 2, 3$ are the vertices of kth triangle, $\triangle_k$. This allows us to perform the interpolation over σ, and then using the mapping, $T_k(r, s)$, to define the corresponding interpolation over each $\triangle_k$, thus greatly simplifying the analysis.

Here, we consider the Lagrange formula for the polynomial interpolation on σ, that is

$$P_d(r, s) = \sum_{j=1}^{f_d} G(\mathbf{q}_j) l_j(r, s), \tag{3.10}$$

where $G(\mathbf{q}) = g(T_k(\mathbf{q}))$ is the function to be interpolated, the $\mathbf{q}_j$ are the interpolation points, and l_j the corresponding Lagrange basis functions [118]. The relation between the polynomial interpolant on the unit simplex and that on the kth triangle in our domain is given by $P(r, s) \equiv p(T_k(r, s))$. Considering now the original triangle, $\triangle_k$, and using the map T_k defined above, we can write down the polynomial interpolant of the function g at the nodes $\mathbf{v}_{k,1}, \ldots, \mathbf{v}_{k,f_d}$ as follows:

$$p_d(x, y) = \sum_{j=1}^{f_d} g(T_k(\mathbf{q}_j)) l_j(r, s), \tag{3.11}$$

where $(x, y) = T_k(r, s)$. Since $T_k(\mathbf{q}_j) = \mathbf{v}_{k,j}$, we can simplify the above to:

$$p_d(x, y) = \sum_{j=1}^{f_d} g(\mathbf{v}_{k,j}) l_j(r, s). \tag{3.12}$$

Note that in this thesis, our main focus is on linear interpolation (*i.e.* $d = 1$); however, we do extend to quadratic ($d = 2$) in Chapter 5 and so for completeness define both linear and quadratic schemes below.

Linear interpolation

When considering linear interpolation, we choose the coordinates of the $f_d = 3$ interpolation nodes on σ (Figure 3.1(a)) as follows

$$\mathbf{q}_1 = (0, 0), \quad \mathbf{q}_2 = (0, 1), \quad \mathbf{q}_3 = (1, 0).$$

The linear Lagrange basis functions are given by

$$l_1(r, s) = 1 - r - s, \quad l_2 = s \text{ and } l_3 = r.$$

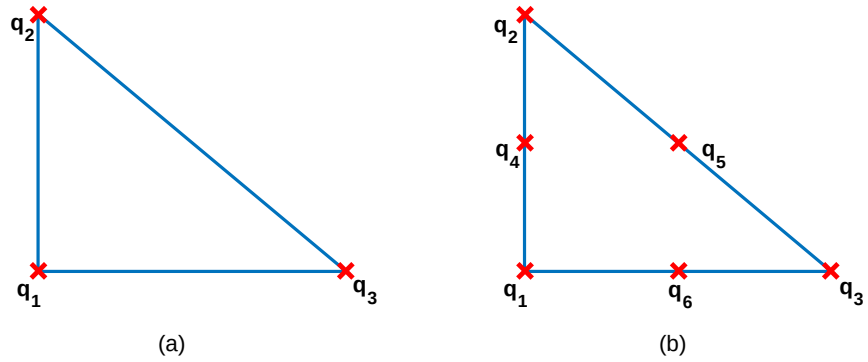


Figure 3.1: The unit simplex with (a) three and (b) six interpolation nodes, respectively.

Thus using (3.10), the unique linear polynomial interpolating $G(r,s)$ at the nodes $\{\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\}$ is

$$P_1(r,s) = \sum_{j=1}^{3} G(\mathbf{q}_j) l_j(r,s).$$

Quadratic interpolation

For the quadratic case $f_d = 6$, and we choose interpolation points on σ (Figure 3.1(b)) as follows

$$\begin{aligned} \mathbf{q}_1 &= (0,0), \quad \mathbf{q}_2 = (0,1), \quad \mathbf{q}_3 = (1,0), \\ \mathbf{q}_4 &= (0,\tfrac{1}{2}), \quad \mathbf{q}_5 = (\tfrac{1}{2},\tfrac{1}{2}), \quad \mathbf{q}_6 = (\tfrac{1}{2},0). \end{aligned}$$

The quadratic Lagrange basis function are

$$\begin{aligned} l_1 &= u(2u-1), \quad l_2 = s(2r-1), \quad l_3 = s(2s-1), \\ l_4 &= 4ru, \qquad\quad l_5 = 4sr, \qquad\quad l_6 = 4su, \end{aligned}$$

where $u = 1-r-s$. Once again, using (3.10) we can now define the unique quadratic polynomial that interpolates $G(r,s)$ at the nodes $\{\mathbf{q}_1, \ldots \mathbf{q}_6\}$ as

$$P_2(r,s) = \sum_{j=1}^{6} G(\mathbf{q}_j) l_j(r,s).$$

3.2.2 Numerical integration

Solving a partial integro-differential equation using collocation techniques requires the accurate computation of integrals over triangulated surfaces, and so we must consider numerical integration over triangles. Here we outline the application of quadrature on triangles, for a more detailed discussion see for example [158, 159]. As for interpolation techniques, integration formulae can first be developed on the unit simplex, σ, before being adapted to general triangles, $\triangle_k$, via the mapping defined in Equation (3.9). Integration over $\triangle_k$ can be defined as

$$\int_{\triangle_k} g(x,y)\mathrm{d}A = 2\mathrm{Area}(\triangle_k)\int_{\sigma} g(T(r,s))\mathrm{d}\sigma, \tag{3.13}$$

where

$$\mathrm{Area}(\triangle_k) = \frac{1}{2}\begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix}$$

and $(x_i, y_i) = \mathbf{v}_{k,i}$ are the vertices of $\triangle_k$. Thus in what follows we consider the approximation of integrals of the form

$$I = \int_{\sigma} G(r,s)\mathrm{d}\sigma, \tag{3.14}$$

where $G(r,s)$ is a continuous function on the unit simplex, σ.

Below, we outline the two most common ways for developing quadrature rules for approximating (3.14).

1. **Interpolation:** In this case the function over σ, $g(r,s)$ is replaced by a polynomial $P(r,s)$. This polynomial interpolates $G(r,s)$ at the points $\{\mu_1, \ldots, \mu_f\}$ in σ. This can be written as

$$\int_{\sigma} G(r,s)\mathrm{d}\sigma \approx \int_{\sigma} P(r,s)\mathrm{d}\sigma = \sum_{j=1}^{f} w_j G(\mu_j),$$

   where w_j, $j = 1, \ldots, f$, are obtained by integrating the Lagrange basis functions.

2. **Undetermined coefficients:** A set of points $\{\mu_1, ..., \mu_f\} \in \sigma$ are chosen such that

$$\int_{\sigma} G(r,s)\mathrm{d}\sigma \approx \sum_{j=1}^{f} w_j G(\mu_j).$$

The weights w_j are chosen so that the degree of precision is maximised, leading to a linear system for w_j, $j = 1, ..., f$.

Quadrature rules

In this thesis we consider integration formulae developed via interpolation. In general these formulae are based on the numerical integration formula

$$\int_\sigma G(r,s)\mathrm{d}\sigma \approx \sum_{j=1}^{f_d} w_j G(\mathbf{q}_j), \quad w_j \equiv \int_\sigma l_j(r,s)\mathrm{d}\sigma. \tag{3.15}$$

This is generated by integrating the interpolation polynomial $P_d(r,s)$ defined in (3.10). When implementing linear collocation we use the three point quadrature rule,

$$\int_\sigma G(r,s)\mathrm{d}\sigma \approx \frac{1}{6}\left[G(0,0) + G(0,1) + G(1,0)\right]. \tag{3.16}$$

When extending to quadratic collocation we need to consider a formula that has a higher degree of precision. In this case we implement the seven point quadrature rule, derived in [159] and given by

$$\begin{aligned}\int_\sigma G(r,s)\mathrm{d}\sigma \approx & \frac{9}{80}G\left(\frac{1}{3},\frac{1}{3}\right) + B\left[G(\alpha,\alpha) + G(\alpha,\gamma) + G(\beta,\alpha)\right] \\ & + C\left[G(\gamma,\gamma) + G(\gamma,\delta) + G(\delta,\gamma)\right],\end{aligned} \tag{3.17}$$

where,

$$\begin{aligned}\alpha &= \tfrac{6-\sqrt{15}}{21}, \quad & \beta &= \tfrac{9+2\sqrt{15}}{21}, \\ \gamma &= \tfrac{6+\sqrt{15}}{21}, \quad & \delta &= \tfrac{9-2\sqrt{15}}{21}, \\ B &= \tfrac{155-\sqrt{15}}{2400}, \quad & C &= \tfrac{155+\sqrt{15}}{2400}.\end{aligned}$$

In contrast to the three point rule, which has degree of precision 1, this formula has degree of precision 5.

3.2.3 Implementing the collocation method

The collocation method is an example of a projection method that approximates an infinite dimensional problem, such as a neural field model, by a finite dimensional one, via a suitably defined projection operator $\mathcal{P}_n$. In this section we provide an outline of the method as applied to the NFMs described in Chapter 2.

Consider the following triangulation $\mathcal{T}_n = \{\triangle_1, ..., \triangle_n\}$ of the domain under consideration. Suppose that on each triangle $\triangle_k$ we employ a piecewise polynomial approximation of the unknown function $u(\mathbf{x}, t)$. In this case the projection operator takes the form

$$\begin{aligned}\mathcal{P}_n u(\mathbf{x}, t) &= u_n(\mathbf{x}, t) \\ &= \sum_{j=1}^{f_d} u(\mathbf{v}_{k,j}, t) l_j(\mathbf{x}), \quad (\mathbf{x}, t) \in \triangle_k, \ k = 1, 2, ..., n. \end{aligned} \tag{3.18}$$

Here, $\mathbf{v}_{k,j}$ denotes the coordinates of the j^{th} interpolation point of the k^{th} triangle, $\triangle_k$, whilst l_j denotes the Lagrange basis functions [118]. This allows us to formulate the following approximation to the Amari Equation (again, see Chapter 2, Equation 2.1):

$$\frac{\partial u_n(\mathbf{x}, t)}{\partial t} = A\mathcal{P}_n \left\{ \int_\Omega w(\mathbf{x}, \mathbf{x}') S(u_n(\mathbf{x}', t)) \mathrm{d}\Omega(\mathbf{x}') \right\} - u_n(\mathbf{x}, t). \tag{3.19}$$

Now, assuming the above expression holds exactly at the node values $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{n_v}$, where n_v refers collectively to a global numbering of the node points $\mathbf{v}_{k,j}$, we obtain a collocation scheme. If we also consider, as discussed previously, solving over the unit simplex, σ, and then transforming back to each $\triangle_k \in \mathcal{T}_n$, via the mapping in (3.9), then we can write (3.19) as follows:

$$\frac{\mathrm{d}u_n(\mathbf{v}_i, t)}{\mathrm{d}t} = 2A \sum_{k=1}^{n} \mathrm{Area}(\triangle_k) \int_\sigma w(\mathbf{v}_i, T_k(r, s)) S\left(\sum_{j=1}^{f_d} u(\mathbf{v}_{k,j}, t) l_j(r, s) \right) \mathrm{d}r\mathrm{d}s, \tag{3.20}$$

for $i = 1, ..., n_v$. The above results in a system of n_v ordinary differential equations that can be solved to determine approximate solutions of the Amari Equation. As we shall see later on, this result is easily extended to the adaptive NFM introduced in the previous Chapter (see Equation (2.32)).

It is straightforward to extend collocation techniques to solve NFMs on triangulated surfaces, embedded in three dimensional space, such as those studied in this thesis. There are two main differences: firstly, the mapping, T_k, now maps to points in $\mathbb{R}^3$, *i.e.*

$$(x, y, z) = T_k(r, s),$$

but is otherwise defined analogously to the two-dimensional case, and secondly, the

formula for the area of the triangles, $\triangle_k$, is now given by

$$\text{Area}(\triangle_k) = \frac{1}{2}\left|\begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \times \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix}\right|.$$

3.3 Domains

In this thesis we consider both flat and non-flat triangulated domains. In particular, we consider the periodic square, $\Omega = [-L, L]^2$, the torus, $\Omega = T^2$, and the surface of the rat brain. In the first two cases, we consider triangulations obtained from both regular Cartesian grids, as well as more general, unstructured triangulations, and so in this section we provide details of the construction process. In the case of the rat brain the triangulation is obtained via neuroimaging data and so we also provide some additional details concerning this data.

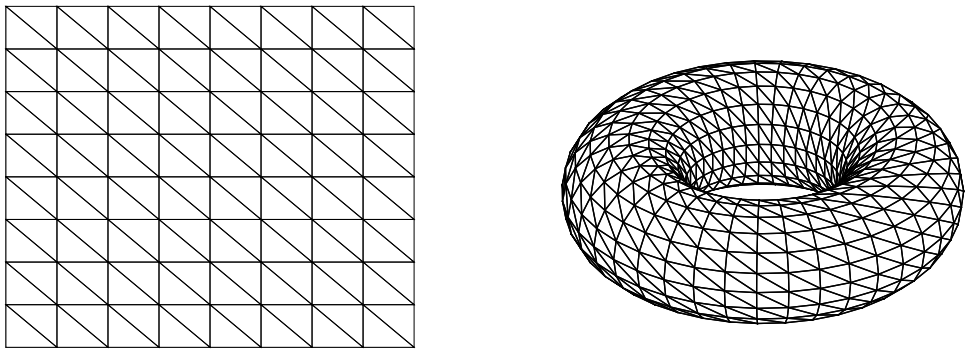
3.3.1 Cartesian grid based triangulation

Firstly we consider a Cartesian grid based triangulation of the periodic square where the triangle vertices are located at regularly spaced Cartesian grid points. Importantly, standard methods, such as the trapezoidal rule and fast Fourier transforms (FFTs), can be deployed to solve NFMs on regular grids, thus allowing direct comparisons between collocation methods and the standard approaches considered in the previous chapter. Such triangulations can be generated using MATLAB's built-in function `delaunay`$(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ and $\mathbf{y}$ are the Cartesian coordinates. Figure 3.2(a) shows an illustrative example of such a triangulated domain.

The surface of the torus in the Euclidean space $\mathbb{R}^3$ can be described using the following parametrisation:

$$(\theta, \phi) \mapsto \begin{pmatrix} (R + r\cos(\theta))\cos(\phi) \\ (R + r\cos(\theta))\sin(\phi) \\ r\sin(\theta) \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{3.21}$$

Here, R denotes the major curvature radius and r the minor curvature radius, while the angles $\phi, \theta \in [0, 2\pi)$. To obtain a regular triangulation of the torus we perform a regular discretisation of $\theta - \phi$ space (such as that shown in Figure 3.2(a)) and then map this via (3.21) on to the torus -see Figure 3.2(b). Importantly, FFTs can not be deployed to solve a NFM on the torus since the above change of coordinates



(a) (b)

Figure 3.2: (a) Illustration of a square domain that uses Cartesian grid points as triangle vertices. (b) Illustration of a triangulated torus using regularly spaced grid points in the toroidal coordinate system as triangle vertices.

result in equations that are no longer of convolution type. We can still deploy the trapezoidal method but only in the case of a regular discretisation of the torus; as we shall see, the collocation method works for both regular and irregular meshes.

3.3.2 General triangulations

Next we consider more general triangulations, that is, triangulations where the triangle vertices do not lie on a Cartesian grid. We consider two different procedures for constructing general triangulations: triangulations obtained using the MATLAB DistMesh package [160] and 'random' triangulations that are obtained by randomly perturbing vertices in the regular triangulations described previously.

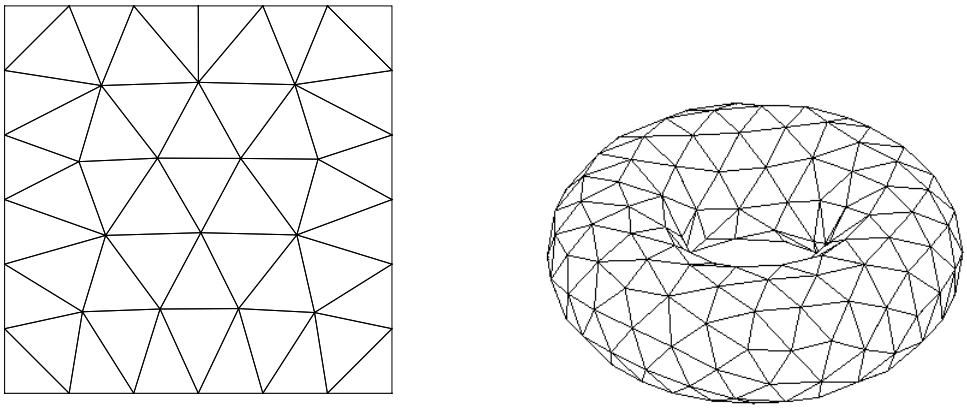
DistMesh triangulation

The DistMesh MATLAB package constructs a triangulation such that each triangle is approximately equilateral and of the same size using optimisation techniques. Below we describe the particular DistMesh functions that we used in order to obtain triangulations of both the square domain, $\Omega = [-L, L]^2$, (Figure 3.3(a)) and the torus, $\Omega = T^2$ (Figure 3.3(b)).

For the periodic square we use the command

```
% DistMesh function
[P,T]=distmesh2d(fd,fh,h0,bbox,pfix,varargin)
```

Here the outputs `P` and `T` are the lists of vertex coordinates and the vertex numbers forming each triangle of the resulting triangulation. The input arguments are



(a) (b)

Figure 3.3: (a) Illustration of a DistMesh triangulation of the periodic square. (b) Illustration of a DistMesh triangulation of the torus.

described as follows.

- `fd` is a distance function describing the geometry, in this case `fd=@dpoly`, defines a polygon with vertices `pv`.
- `fh` is the desired edge length function, in this case we use `fh=@huniform`, which selects a uniform distribution for the edge length.
- The parameter `h0` is the initial value of the distance between points in the node distribution. Throughout the optimisation, the distance between points may stray slightly from this value.
- The parameter `bbox` is the bounding box of the square, which is given as `bbox=[-L/2, -L/2; L/2 L/2]`.
- Any fixed points can be specified using `pfix`, in this case we fix all of the boundary points.
- Any additional parameters for the functions `fd` and `fh` are specified in `varargin`, in this case we input `pv`, the points required to generate a square geometry from `fd=@dpoly`.

In the case of the torus we use the following DistMesh function to triangulate the surface:

```
% DistMesh function
[P,T]=distmeshsurface(fd,fh,h0,bbox,fparams)
```

Here the outputs `P` and `T` are again the resulting list of vertices and triangles. The input arguments are described as follows:

- The inputs `fh, h0` and `bbox` are the same as stated for the periodic square, however you can see that in this case we do fix any vertices in the mesh.
- Again `fd` is a distance function describing the geometry, but in this case

  ```
  fd = @(p) (sum(p^2,2) + R^2-r^2)^2-4*r^2*(p(:,1)^2 +
     p(:,2)^2)
  ```

  Here `R` and `r` are the major and minor radii of the torus respectively.

Random triangulation

Triangulated cortical surfaces obtained from MRI data, may not be particularly regular and hence we consider the performance of our methods on 'random' triangulations. On the periodic square we constructed a random triangulation by perturbing the interior points of the Cartesian grid. This perturbation was performed as follows:

```
P = P + dx*Alpha*(2*rand(length(P),2)-1).
```

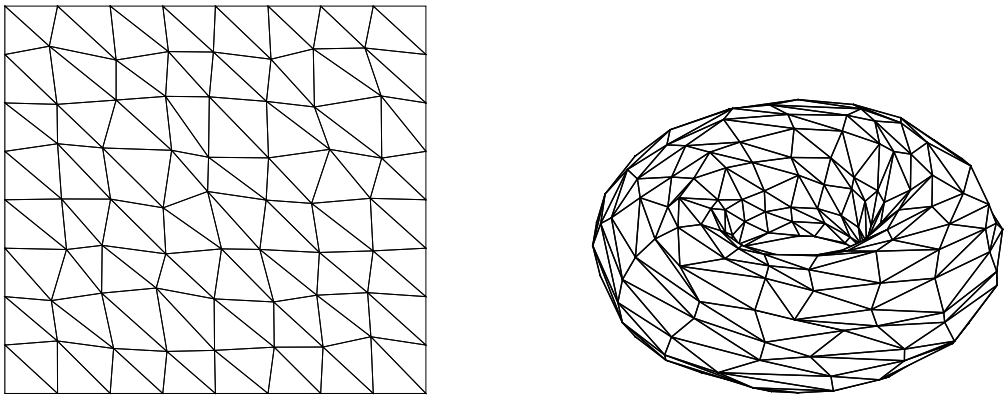
Here `P` are the vertices we wish to perturb, `dx` is the distance between two points in the uniform domain and `Alpha` is the amount by which we wish to perturb the points. Figure 3.4(a) shows an example of a randomly perturbed triangulation of the periodic square with `Alpha`$= 0.1$.

When considering the torus, a 'random' triangulation is obtained by transforming the perturbed points of the square, given in (θ, ϕ) coordinates, via the mapping in (3.21). This results in a list of (x, y, z) points for the triangulation along with the original triangle list obtained on the Cartesian grid, see Figure 3.4(b) for an illustration of a 'random' triangulation of the torus.

3.3.3 Rat brain

The cortical surface of the rat was obtained via the CARET software package, which deploys bespoke algorithms that generate a representation of the cortical surface given a high-quality MR image [161]. More specifically, we downloaded the surface data for the left-hemisphere of the rat, which consists of two files:

- **Coordinate file** – containing the Cartesian coordinates of cortical surface vertices;



(a) (b)

Figure 3.4: (a) Illustration of a random triangulation of the periodic square. (b) Illustration of a random triangulation of the torus.

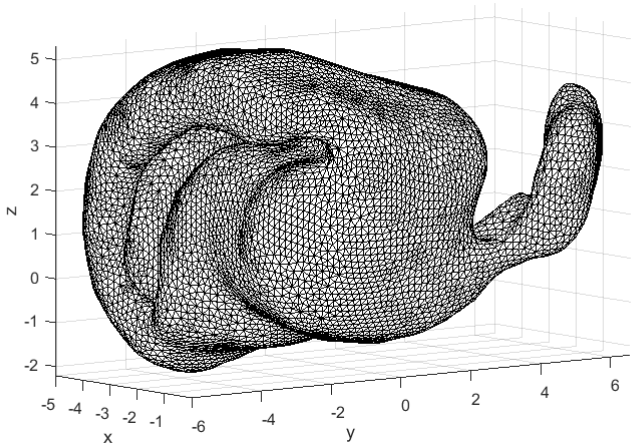


Figure 3.5: Triangulated surface of the left hemisphere of a rat brain.

- **Topology file** – defining the relationship between (connections) surface vertices and the vertices that make up an element in the cortical surface representation.

Using the $n_v = 9623$ available data points (cortical vertices) we deployed the CARET MATLAB toolbox to construct a triangulated mesh representation of the left hemisphere of the rat cortex consisting of 19242 triangles. This results in a genus zero cortical surface representation of the left hemisphere generated via the removal of white matter. Note that whilst the resulting triangulated surface includes both the pial surface as well as subcortical structures it is computationally convenient to consider closed manifolds in our work.

3.4 Computing geodesics on triangulated surfaces

One major difference between solving on non-flat domains as opposed to flat domains is computing the distance, $d(\mathbf{x}, \mathbf{x}')$, between neural units, which is necessary to quantify neural interactions, as defined by the connectivity kernel w. In the case of the flat domain, one simply computes the Euclidean distance between the two points; however, for non-flat domains one needs to compute the geodesic distance between points. Here, we briefly describe two different numerical schemes for computing the geodesic distance between two points on a triangulated surface before performing a comparative analysis on a sphere, in which case the exact geodesic distance can be calculated analytically.

3.4.1 Fast marching algorithm

The fast marching algorithm in two dimensions is a numerical scheme for solving the Eikonal equation

$$|\nabla T| = F(x, y), \quad x, y \in \Omega, \tag{3.22}$$

which is found in problems for wave propagation [162, 163, 164]. Intuitively, one can consider the solution T of the above equation to be the time taken for a wave travelling at speed $1/F$ to propogate from the boundary $\partial\Omega$ to the point (x, y) [165]. It was created by James Sethian [166], with its beginnings found in his PhD thesis from 1982 [167]. Originally, the method solved (3.22) on a uniform Cartesian mesh (see Figure 3.6(a)), but has more recently been extended to work on more general triangulated surfaces, and as a result can be deployed to determine geodesic distances on polygonal surfaces. Importantly, implementations of the algorithm

exist in MATLAB [168] and so the method is straightforward to utilise. Here we give a brief outline of the method. Note that for simplicity we introduce the fast marching method on two-dimensional domains, the extension to the more general case of a surface embedded in three-dimensions will be discussed later.

The fast marching algorithm solves the Eikonal equation (3.22) in $\mathcal{O}(N \log N)$ steps, where N is the number of mesh points. The main idea is to find the solution T by advancing the wave front (gradient term) in an upwind manner over the mesh [169]. The algorithm is accelerated by only considering a narrow band around the wave front and marching it forwards, which fixes the values of existing points and brings in new points to the narrow band. First, all the points in the initial condition are labelled *alive*. All the mesh points that are one grid point away from these points are labelled *close*, whereas all other points are labelled *far*.

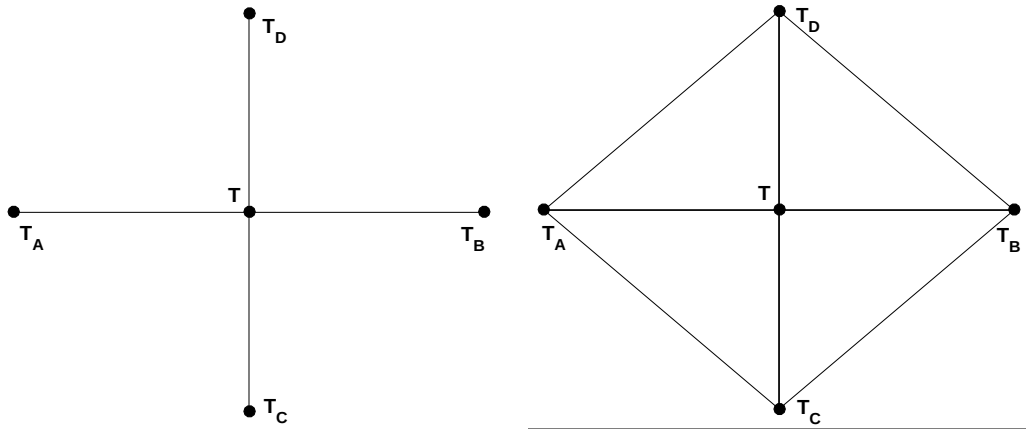
- The point with the smallest T value in *close* is located and labelled *trial*.
- This *trial* point is then added to the set of *alive* points and removed from *close* points.
- Update the set of *close* points such that it now includes all neighbours of the *trial* point that were in *far*, removing them from *far* if they are now *close*.
- The values of T at all the *close* points are found by solving

$$\left[\max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2\right]^{\frac{1}{2}} = F_{ij}, \tag{3.23}$$

  using only the values from points that are *alive*. Here D_{ij} are first order finite difference approximations at (x_i, y_j).
- Repeat.

When recomputing the values of T at the upwind *close* points in the procedure, the value of T cannot be smaller than any value of T already accepted in the *alive* points. Therefore when marching the solution outwards, no value of T in the set of *alive* points needs updating/considering again.

When extending the method to triangulated meshes the update procedure for solving (3.23) needs adapting. For an uniform Cartesian mesh, the update procedure is as follows. Considering a Cartesian square grid such as that shown in Figure 3.6(a), the value of T at the center point (i, j) needs updating. The neighbouring points



(a) Uniform Cartesian mesh update (b) Triangular update

Figure 3.6: Illustration of the update upwind procedure for (a) uniform Cartesian meshes and (b) simple triangulated meshes.

are labelled as $T_A = T_{i-1,j}, T_B = T_{i+1,j}, T_C = T_{i,j-1}, T_D = T_{i,j+1}$, when considering (3.23) we now write

$$\left[\max\left(\frac{T-T_A}{h}, \frac{T-T_B}{h}\right)^2 + \max\left(\frac{T-T_C}{h}, \frac{T-T_D}{h}\right)^2\right] = F_{ij}^2, \tag{3.24}$$

where h is the uniform grid spacing. Considering, for example, the contributions from A and C it is assumed that, without loss of generality, $T_A \leq T_C$ and (3.24) becomes

$$(T-T_A)^2 + (T-T_C)^2 = h^2 F_{i,j}^2.$$

From [166] it is shown that there are two possibilities for the solution:

- There is a real solution T, with $T > T_A$ and $T > T_C$, to the quadratic

$$(T-T_A)^2 + (T-T_C)^2 = h^2 F_{i,j}^2.$$

- There is a real solution T, with $T > T_A$ and $T \leq T_C$, to the one-dimensional update

$$(T-T_A)^2 = h^2 F_{i,j}^2.$$

For each possible pair of vertices on the mesh, all possible solutions of T are obtained and the updated point is the one that produces the smallest value.

To extend the fast marching method to triangulated surfaces the Cartesian grid is adapted as shown in Figure 3.6(b). Now, for this simple case, the solution to the

quadratic will change with the equation of the plane. For example, again considering the contributions of A and C, the equation of the plane is

$$\left(\frac{T-T_A}{h}\right)x+\left(\frac{T-T_C}{h}\right)y+T=z.$$

Therefore (3.23) becomes

$$\left(\frac{T-T_A}{h}\right)^2+\left(\frac{T-T_C}{h}\right)^2=F_{ij}^2, \tag{3.25}$$

where the plane is tilted at the point (i,j) by T so that there is a gradient magnitude equal to F. Again the solution is found in an upwind manner as it is required that T be smaller than any other contributors. The fast marching method can then be implemented on this triangular mesh as it was on the Cartesian grid.

The algorithm can be used to compute distances on triangulated surfaces and therefore to construct minimal geodesics. The Eikonal equation is solved on the triangulated surface with speed $F=1$ in order to compute the distance from a source point. To find the geodesic path $X(s)$, the following differential equation is solved

$$\frac{\mathrm{d}X(s)}{\mathrm{d}s}=-\nabla T, \tag{3.26}$$

from which the distance is found by summing the distances between the nodes found in the path. In order to apply the fast matching algorithm on a triangulated surface we employ the following function

```
% Fast marching toolbox
[D]= perform-fast-marching-mesh(vertex, faces, start-
   point)
```

from the fast marching MATLAB toolbox [168]. Here the output `D` is the distance from a specified input `start-point` to all other points in the mesh. The mesh is described by the inputs `vertex`, a list of vertex coordinates and `faces`, a list of node values which make up the triangle list. For further details see [165, 166].

3.4.2 Exact geodesic algorithm

The exact geodesic toolbox [170] is a MATLAB implementation of the MMP (Mitchell, Mount and Papadimitriou) algorithm [171]. Developed in 1987, the MMP algorithm was employed to solve the *Discrete Geodesic Problem*, *i.e.* find the shortest path

between two given points on the surface of a polyhedron, with the path also lying on the surface of the polyhedron.

The algorithm finds the length of the shortest path in $\mathcal{O}(\log M)$ where M is the number of edges of the surface and can consider any general polyhedra, both of which improve on the work posed in [172, 173, 174]. The MMP algorithm uses a technique the authors call *continuous Dijkstra*, since it is reminscient of Dijkstra's algorithm for finding the shortest paths in a graph [175]. The edges of the polyhedron act as nodes of a graph, where instead of a unique distance there is a function that labels the node. The minimum of the discretised function is monitored by considering *intervals of optimality*, that is, an interval that subdivides the edge under consideration into regions, where the shortest path to other points in the region have the same discrete description of the function.

We now give a brief description of the algorithm described in [171]. A signal is propagated from a source point, s, to all other points on a polyhedral surface. When another point, a, on the surface receives the signal it propagates it further. The point a is labelled with a time $d(a)$, the time it received the signal, which is also the minimum distance from s. This re-propagation is only done for a finite number of points, those listed in the candidate intervals, that is an interval of points that lie on the edges opposite s.

- The algorithm is initialised. The source point, s, is labelled 0 and for every edge opposite s, a candidate interval is created. All points in the candidate intervals are labelled $d(a) = +\infty$.
- For each point in the intervals compute and store the distance from s.
- Find the entry with the smallest value and permanently label the corresponding point with this distance.
- Repeat the process until all distances from s to each point a on the surface are computed.

In order to apply the exact geodesic algorithm on a triangulated surface we employ the following function

```
% Exact geodesic toolbox
[D]= exact-geodesic(V, F, id)
```

from the exact geodesic MATLAB toolbox [170]. Here the output `D` is the distance from a specified start point `id` to all other points in the mesh. The mesh is described

by the inputs `V`, a list of vertex coordinates and `F`, a list of node values forming each triangle. For further details of the algorithm see [171].

3.4.3 Comparing the fast marching and exact geodesic algorithms

A comparative analysis was performed for the two aforementioned techniques for computing geodesics by considering distances on a triangulated sphere, since an analytical formula exists in this instance. Distance matrices were computed for the two numerical schemes as well as an analytical distance matrix for a series of increasingly fine triangulations of the unit sphere, and numerical errors computed for both methods. Figure 3.7(a) shows the maximal absolute error for both the fast marching and exact geodesic methods, whilst Figures 3.7(b) and (c) show the maximal and average relative errors, respectively. In each case, the exact geodesic algorithm (denoted GP in Figure 3.7) displays better convergence properties compared to the fast marching algorithm (denoted FM in Figure 3.7). Note that this result, coupled with numerical simulations of the NFMs studied in this work using both methods (not shown), has led us to deploy the exact geodesic algorithm in the remainder of this work when computing geodesics on general triangulated surfaces.

3.5 Numerical bifurcation analysis

To understand a cell's neurocomputational properties, current research in neuoscience involves studying the cells membrane voltage as well as second-messenger-gated currents [176]. A cell's second-messenger-gated current is an intracellular signal (*i.e.* a signal from inside the cell) that is produced in response to a stimulus [177]. Knowledge of these intracellular signals is thought to provide a complete description of the cell's behaviour. However, accepting this hypothesis contradicts the known fact that cells displaying similar currents often exhibit different dynamics [176]. This difference in dynamics, as shown by Rinzel and Ermentrout [178], is caused by the different bifurcations that occur in excitatory neurons – note that neurons are typically classified into two classes: excitatory and inhibitory. Thus it is of considerable interest to study a cells behaviour, and more generally populations of cells, as certain experimental parameters are varied [179, 180]. As well as bifurcation analysis other popular methods for studying solutions of NFMs include Evans functions [105, 181, 182, 183, 184] and Turing instability analysis [95, 147, 148, 185, 186],

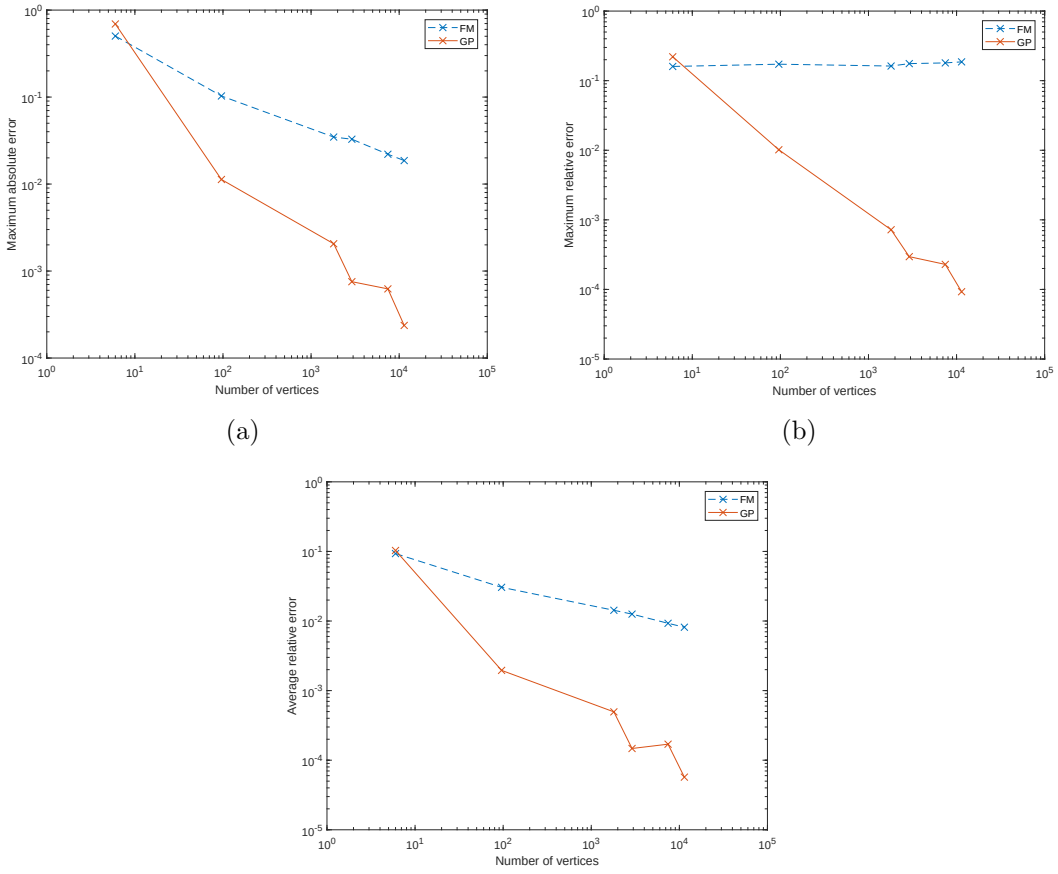


Figure 3.7: Comparing the fast marching algorithm (blue dashed) and exact geodesic algorithm (red) to the analytic distance on the unit sphere; (a) maximal absolute error, (b) maximal relative error, (c) average relative error.

which we do not study in this thesis.

Bifurcations are qualitative changes in the behaviour of a system as certain parameters vary, and determining points at which bifurcations occur can in general only be done numerically. According to Seydel [124], the basic steps in any numerical bifurcation analysis consist of

(a) performing numerical continuation;

(b) determining when bifurcations occur and switching between branches.

Here, we focus on the continuation process, restricting our description to the method of pseudo arc-length, before briefly describing some of the practical aspects of numerical bifurcation. The interested reader should consult the excellent text by Seydel [124] for further details as well as a description of alternative approaches for numerical continuation.

Suppose that $(\mathbf{u_0}, \lambda_0)$ is a steady state solution of our NFM, *i.e.* a solution of the nonlinear algebraic equations

$$0 = f(\mathbf{u}, \lambda), \tag{3.27}$$

where f is the right hand side of the system of ODEs resulting from the discretisation of our NFM. Then using arclength in order to parameterise the solution branch emanating from the point $(\mathbf{u_0}, \lambda_0)$ (see Figure 3.8), we can determine a new point, $(\mathbf{u_1}, \lambda_1)$ say, that also lies on the solution branch, by solving, in addition to Equation (3.27), a so-called *parameterising equation*:

$$p(\mathbf{u}, \lambda, s) = 0,$$

which, in the case of pseudo arclength continuation is given by

$$(\mathbf{u}_1 - \mathbf{u}_0)^T \dot{\mathbf{u}}_0 + (\lambda_1 - \lambda_0)^T \dot{\lambda}_0 - \Delta s = 0. \tag{3.28}$$

Here Δs is the pseudo-arclength step size and $(\dot{\mathbf{u}}, \dot{\lambda})$ is the tangent to the curve at $(\mathbf{u}, \lambda)$. This constraint means that the new point must lie on the perpendicular to the tangent vector, that is, $(\mathbf{u}_1, \lambda_1)$ must lie on the dashed line shown in Figure 3.8.

It follows that in order to determine solution branches of our NFMs we need to

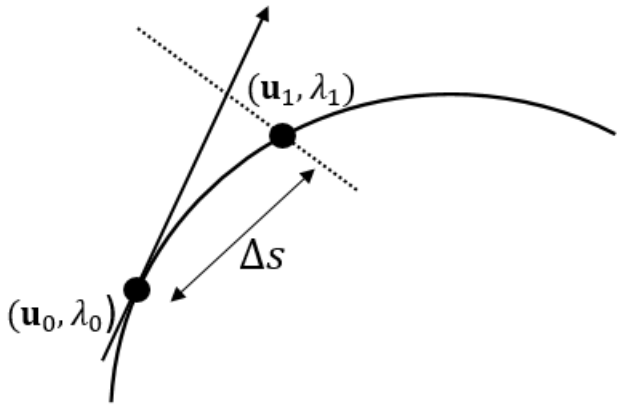


Figure 3.8: Geometric representation of the pseudo-arclength continuation scheme.

solve the extended system of nonlinear algebraic equations given by

$$\begin{aligned} F(\mathbf{u}, \lambda) &:= \begin{pmatrix} f(\mathbf{u}, \lambda) \\ p(\mathbf{u}, \lambda, s) \end{pmatrix} \\ &= \begin{pmatrix} f(\mathbf{u}, \lambda) \\ (\mathbf{u}_1 - \mathbf{u}_0)^T \dot{\mathbf{u}}_0 + (\lambda_1 - \lambda_0)^T \dot{\lambda}_0 - \Delta s \end{pmatrix} = \mathbf{0}. \end{aligned} \tag{3.29}$$

In order to solve the above equation we compute the following Newton iteration:

$$\begin{pmatrix} \mathbf{u}_1^{i+1} \\ \lambda_1^{i+1} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1^{i} \\ \lambda_1^{i} \end{pmatrix} - J_i^{-1} \begin{pmatrix} f(\mathbf{u}_1^i, \lambda_1^i) \\ (\mathbf{u}_1^i - \mathbf{u}_0)^T \dot{\mathbf{u}}_0 + (\lambda_1^i - \lambda_0)^T \dot{\lambda}_0 - \Delta s \end{pmatrix}, \tag{3.30}$$

until sufficient accuracy is attained. Here

$$J_i = \begin{pmatrix} f_{\mathbf{u}} & f_\lambda \\ \dot{\mathbf{u}}_0 & \dot{\lambda}_0 \end{pmatrix}. \tag{3.31}$$

is the Jacobian matrix whose entries are the partial derivatives of F evaluated at the point $\mathbf{u}_1^i, \lambda_1^i$. When solving (3.30) we use a Jacobian free Newton-Krylov solver taken from [123].

Newton-Krylov methods

Equation (3.30) can be rewritten as a linear system of equations as follows

$$J \Delta \mathbf{u}^n = F(\mathbf{u}^n, \lambda^n). \tag{3.32}$$

Here, for simplicity we have dropped the subscripts in Equation (3.30), and so, for example, $\Delta\mathbf{u}^n = \mathbf{u}^{n+1} - \mathbf{u}^n$. To solve (3.32) we can use the *Krylov method for solving linear systems* [123]. Such methods approximate the solution of the linear system in (3.32) by a sum of the form

$$\mathbf{u}^k = \mathbf{u}^0 + \sum_{i=1}^{k} \alpha_i J^{i-1}\mathbf{r}^0,$$

where

$$\mathbf{r}^0 = F(\mathbf{u}^0, \lambda^0) - J\Delta\mathbf{u}^0,$$

is known as the residual, and $\Delta\mathbf{u}^0 = \mathbf{u}^1 - \mathbf{u}^0$, with $\mathbf{u}^0$ an initial iterate.

This is more compactly expressed by saying that $\mathbf{u}^k \in \mathcal{K}_k$, where the kth *Krylov subspace* is

$$\mathcal{K}_k = \text{span}(\mathbf{r}_0, J\mathbf{r}_0, J^2\mathbf{r}_0, ..., J^{k-1}\mathbf{r}_0).$$

There are a number of different Krylov methods but we deploy the Generalised Minimal Residuals (GMRES) method in our work, which attempts to minimise the residual

$$||\mathbf{r}^k|| = ||F(\mathbf{u}^k, \lambda^k) - J\mathbf{u}^k||$$

over the k^{th} Krylov subspace [123].

Such a method is essential for large systems of equations, such as those derived from NFMs, as the Jacobians grow quickly thus potentially resulting in storage problems and significant increases in computation time - note that we shall consider some of these numerical issues in more detail in later chapters.

3.5.1 Bifurcation detection of equilibria for NFMs

After spatial discretisation the NFMs considered in this thesis result in systems of ODEs of the form

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{f}(\mathbf{u}, \lambda),$$

and so we can use the theory of dynamical systems to study these equations as system paramters are varied. For example, we can consider bifurctaion points, *i.e.* values of the parameter of interest for which the solution changes its properties, such as its stability. When considering 1-parameter bifurcations of equilibria, as in this work, there are two generic bifurcations that can occur: (i) Hopf bifurcations and (ii) saddle-node bifurcations [176]. In our studies only saddle-node bifurcations turn out

to be important and so we give precise definitions of such bifurcation points below as well as commenting on how to detect such points when constructing solution branches.

DEFINITION 3.1. $(\mathbf{u}^*, \lambda^*)$ *is a* ***turning point*** *(or* ***fold*** *bifurcation point, or* ***saddle-node*** *bifurcation) of a stationary solution if the following conditions hold:*

(i) $\mathbf{f}(\mathbf{u}^*, \lambda^*) = \mathbf{0}$,

(ii) $rank(\mathbf{f_u}(\mathbf{u}^*, \lambda^*)) = n - 1$,

(iii) $\mathbf{f}_\lambda(\mathbf{u}^*, \lambda^*) \notin range(\mathbf{f_u}(\mathbf{u}^*, \lambda^*))$ *that is,* $rank(\mathbf{f_u}(\mathbf{u}^*, \lambda^*)|\mathbf{f}_\lambda(\mathbf{u}^*, \lambda^*)) = n$,

(iv) there is a parameterisation $\mathbf{u}(\sigma), \lambda(\sigma)$ *with* $\mathbf{u}(\sigma^*) = \mathbf{u}^*, \lambda(\sigma^*) = \lambda^*$ *and*

$$\frac{\mathrm{d}^2\lambda(\sigma^*)}{\mathrm{d}\sigma^2} \neq 0.$$

To detect a bifurcation point during the continuation process we use the following *bifurcation test function*:

$$\tau := \max\{\alpha_1, \ldots, \alpha_n\},$$

that is we monitor the maximum of the real parts of the eigenvalues $\alpha_i + i\beta_i$ of the Jacobian matrix J. Note that many other choices exist and a thorough discussion of the topic is given in [124].

3.6 Summary

In this chapter we have given brief details of how to solve a NFM using the collocation technique on a triangulated surface for both the case where linear and quadratic basis functions are used. We then discussed the different triangulated domains under consideration in this thesis and explained how to construct such domains using MATLAB, except for the rat brain which is derived from neuroimaging data. We considered two different numerical approaches for computing geodesics and performed an error analysis of these techniques on the sphere since an analytic formula for computing geodesics exists in this case. The outcome of these investigations was that the exact geodesic algorithm was the more accurate of the two methods and so we shall use this in all of our experiments in the chapters 4 and 5. Finally, we introduced some of the basics techniques for performing a numerical bifurcation analysis and, in particular, discussed the pseudo arclength method for

numerical continuation. We implement this scheme when analysing steady state and travelling wave solutions of the neural field models studied in this work, as various model parameters are varied.

Chapter IV

The Amari equation

In this chapter we consider the following Amari equation:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = -u(\mathbf{x},t) + A\int_{\Omega} w(\mathbf{x},\mathbf{x}')S(u(\mathbf{x}',t))\mathrm{d}\Omega(\mathbf{x}'). \tag{4.1}$$

Here, u describes the average activity of the neuronal population at position $\mathbf{x} \in \Omega$ at time $t \in [0,T]$, and the parameters A, h are related to the sensitivities of the problem [81]. The nonlinear function S represents the mean firing rate and is given by

$$S(u) = \frac{1}{1+e^{-\beta(u-h)}},$$

whilst the integral kernel, which describes how neurons positioned at $\mathbf{x}$ and $\mathbf{x}'$ interact, is given by

$$w(\mathbf{x},\mathbf{x}') = e^{-d(\mathbf{x},\mathbf{x}')^2} - 0.17e^{-0.2d(\mathbf{x},\mathbf{x}')^2}, \tag{4.2}$$

which is a mexican-hat type function. Note that d is a suitably defined metric and its choice reflects the geometry of the domain Ω.

More specifically, we present results of applying the numerical techniques introduced in the previous chapter to solve the above NMF on both a flat, periodic square domain and the closed surface of a torus. In both cases, we perform a comparative analysis against more standard techniques, deploying either Fourier based techniques and/or the trapezoidal rule to compute the integral in (4.1), and investigate the dependence of our results on the underlying mesh. As well as this we also consider a bifurcation analysis on the periodic square domain and the surface of the torus when varying the parameters h and A in the system. In the case of the torus we investigate the effect of curvature on the observed solutions by repeating

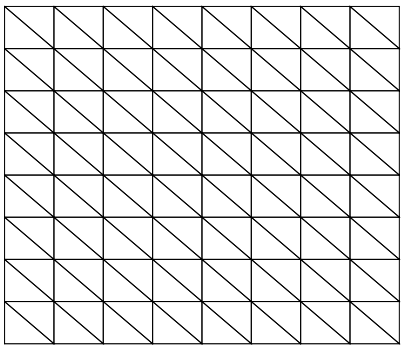


Figure 4.1: Illustration of a domain that uses Cartesian grid points as triangle vertices.

the analysis for several choices of major radius of curvature and fixed minor radius of curvature.

4.1 Numerical Results

In this section we present the results of a number of numerical experiments that were undertaken in order to check the validity of the techniques described in Chapter 3. Of particular importance is our ability to reproduce solutions on generic, irregular triangulations, such as those obtained from neuroimaging studies, and so we begin by investigating the effects of mesh regularity on solutions of (4.1) on a flat, periodic domain, before moving on to look at more general, curved domains.

4.1.1 Planar domain with periodic boundary conditions

When considering the numerical solution of Equation (4.1) the main source of error is the approximation of the integral, which for $\Omega = [-L, L]^2$, becomes

$$I = \int_{-L}^{L} \int_{-L}^{L} w((x, y), (x', y'))S(u(x', y'))\mathrm{d}x'\mathrm{d}y'. \tag{4.3}$$

We note that since

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{(x - x')^2 + (y - y')^2}$$

then the integrals in (4.3) are of convolution type.

We compared the accuracy of computing the integral in (4.3) using linear collocation against fast Fourier transform (FFT) techniques together with the convolution theorem, and the trapezoidal method, both of which require a regular spatial dis-

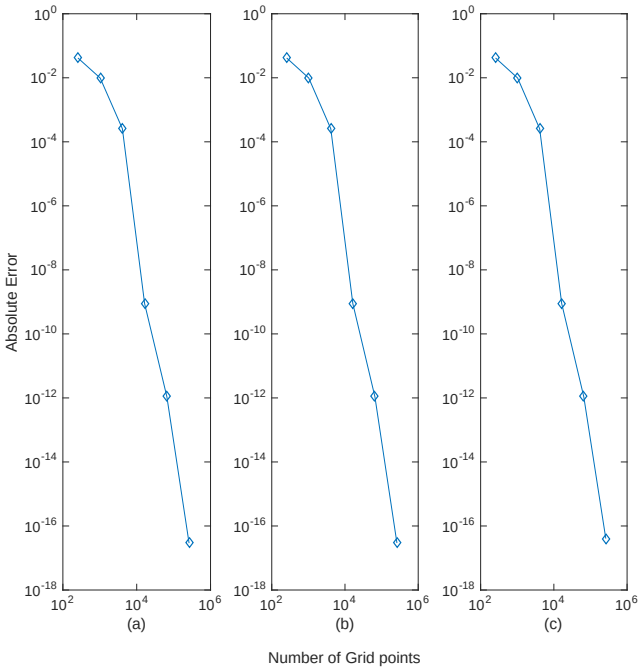


Figure 4.2: The error $|I_m - I_{m+1}|$ plotted against grid size N_{m+1} reveals geometric convergence rates for (a) FFTs, (b) trapezoidal rule and (c) linear collocation, when computing the integral in (4.3).

cretisation on a Cartesian grid. In order to compare these methods directly to piecewise linear collocation, we employ a triangulation whose vertices correspond to the Cartesian grid points for the other two approaches, as shown in Figure 4.1. In our experiments we fixed $L = 7.5$ and set $u(\mathbf{x}) = w(\mathbf{x}, 0)$, *i.e.* the connectivity kernel given in (4.2), since it is qualitatively similar to the bump solutions admitted by (4.1). It is worth noting, however, that similar results are obtained for other sufficiently smooth choices of u (results not shown). To investigate grid convergence, we considered a sequence of refinements of an initial, regular grid consisting of $N_0 = 81$ nodes, such that at the mth stage of refinement, the number of nodes is given by $N_m = (2^m \cdot 8 + 1)^2$ for $m = 1, 2, \ldots, 7$. If we then denote by I_m the numerical approximation of (4.3) on the grid of size N_m, we can approximate the order of convergence of the respective discretisation schemes by considering a log-log plot of the absolute error between consecutive grids, $|I_{m+1} - I_m|$, versus grid size, N_{m+1}. Here we consider point-wise convergence and so all results shown are for a representative grid point. Note that we have repeated the analysis for other grid points and observed almost identical behaviour (experiments not shown).

Our results are displayed in figures 4.2 and 4.3. In particular, we see that both

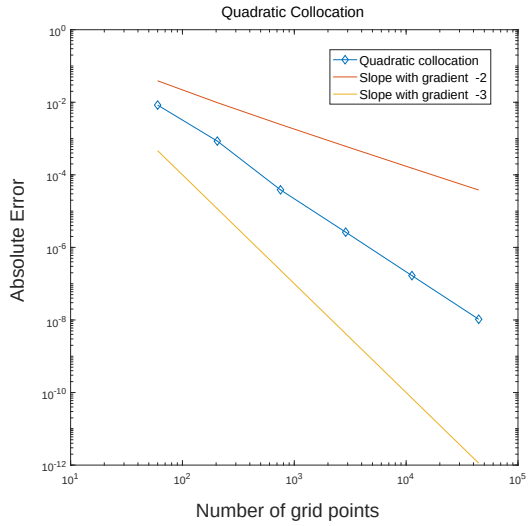


Figure 4.3: Convergence of quadratic collocation when computing the integral (4.3) with the error $|I_m - I_{m+1}|$ plotted against grid size N_{m+1}.

trapezoidal rule and FFTs display geometric convergence, as expected (see the review by Trefethen [125] for a discussion of the convergence properties of the trapezoidal rule on a periodic domain); however, we find, perhaps somewhat surprisingly, that linear collocation also exhibits geometric convergence, whereas quadratic collocation (see Figure 4.3) exhibits only quadratic convergence. To understand the above result, we consider the collocation technique as applied to (4.3) in more detail below.

Firstly, note that employing linear collocation alongside the three point quadrature rule

$$\int_\sigma G(r,s)\mathrm{d}r\mathrm{d}s = \frac{1}{6}[G(0,0)+G(0,1)+G(1,0)],$$

with $G(r,s) = g(T_k(r,s))$, as defined in §3.2, enables us to construct the following numerical approximation to (4.3):

$$\begin{aligned} I \approx \sum_{k=1}^{n} \frac{\mathrm{Area}\,(\triangle_k)}{3} \Bigg[& w(\mathbf{v}, T_k(0,0)) S\left(\sum_{j=1}^{3} u(\mathbf{v}_{k,j}) l_j(0,0)\right) \\ & + w(\mathbf{v}, T_k(0,1)) S\left(\sum_{j=1}^{3} u(\mathbf{v}_{k,j}) l_j(0,1)\right) \\ & + w(\mathbf{v}, T_k(1,0)) S\left(\sum_{j=1}^{3} u(\mathbf{v}_{k,j}) l_j(1,0)\right) \Bigg]. \end{aligned} \tag{4.4}$$

We can further simplify the above by noting that since we are solving on a uniform Cartesian domain, Area $(\triangle_k) = \Delta x^2/2$ for all triangles, where here, Δx $(= \Delta y)$ is the local mesh spacing. Substituting this into (4.4) and evaluating the Lagrange basis functions at the node points gives

$$\frac{\Delta x^2}{6} \sum_{k=1}^{n} \Bigg[w(\mathbf{v}, T_k(0,0)) S(u(\mathbf{v}_{k,1})) + w(\mathbf{v}, T_k(0,1)) S(u(\mathbf{v}_{k,2})) + w(\mathbf{v}, T_k(1,0)) S(u(\mathbf{v}_{k,3})) \Bigg].$$

Recalling that $T_k(0,0)$ denotes the coordinates of the first vertex in $\triangle_k$, $T_k(0,1)$ the second and $T_k(1,0)$ the third, we can rewrite the above equation as follows

$$\sum_{k=1}^{n} \Bigg[w(\mathbf{v}, \mathbf{v}_{k,1}) S(u(\mathbf{v}_{k,1})) + w(\mathbf{v}, \mathbf{v}_{k,2}) S(u(\mathbf{v}_{k,2})) + w(\mathbf{v}, \mathbf{v}_{k,3}) S(u(\mathbf{v}_{k,3})) \Bigg] \frac{\Delta x^2}{6}. \tag{4.5}$$

However, since the triangle vertices are simply the Cartesian grid points, Equation (4.5) is nothing other than the trapezoidal rule for solving (4.3) on a periodic two-dimensional domain. The factor of $1/6$ occurs due to the fact that each node appears six times in the sum in (4.5). Thus, we have shown that for a regular grid with periodic boundary conditions solving Equation (4.3) using linear collocation and a quadrature rule based only on the triangle vertices is equivalent to using the trapezoidal rule. This explains the spectral convergence observed in Figure 4.2.

Next, we considered the effects of mesh regularity on the accuracy of computing the integral in (4.3). To do this we deployed the DistMesh MATLAB package [160] to generate a general mesh, that is, one in which the triangle vertices do not lie on a Cartesian grid, as in our previous investigations. It is important to note that standard techniques such as those deployed above (*i.e.* trapezoidal and FFT methods) cannot be applied in this more general setting. As before, numerical errors were approximated by comparing the numerical solution of (4.3) at the same grid point across a range of increasingly fine meshes. More precisely, we constructed an initial, coarse triangulation of the square $[-L, L]^2$ consisting of $N_0 = 79$ nodes using the DistMesh package, we then proceeded to refine this triangulation by subdividing

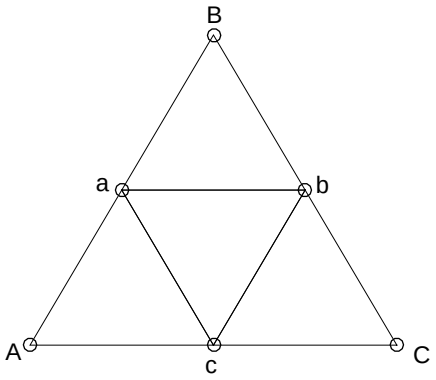


Figure 4.4: Illustration of the refinement of each triangle, where A, B and C are the original triangle vertices and a, b and c are the new vertices introduced by the refinement.

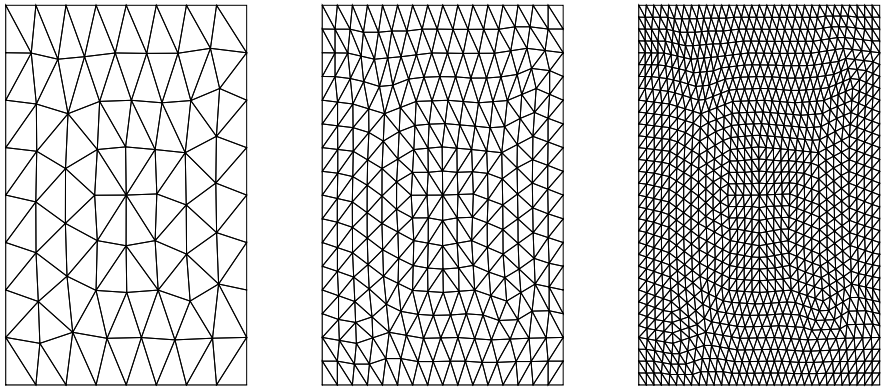
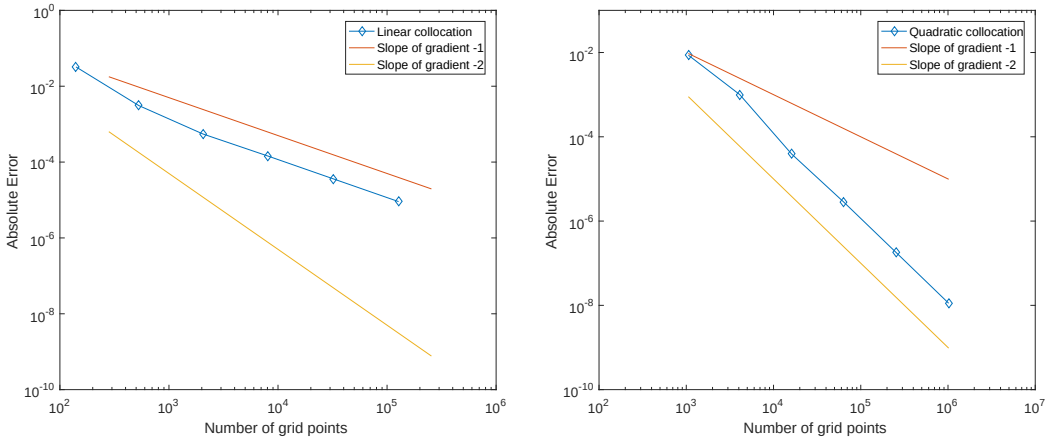


Figure 4.5: An illustration of the refinement procedure for a Distmesh triangulation [160].



(a) Linear collocation (b) Quadratic collocation

Figure 4.6: Convergence of linear and quadratic collocation when computing the integral in (4.3) using a DistMesh triangulation (see Figure 4.5).

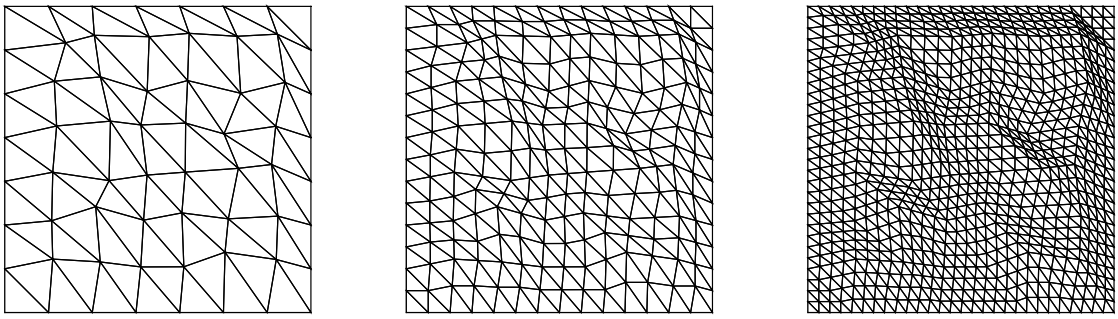
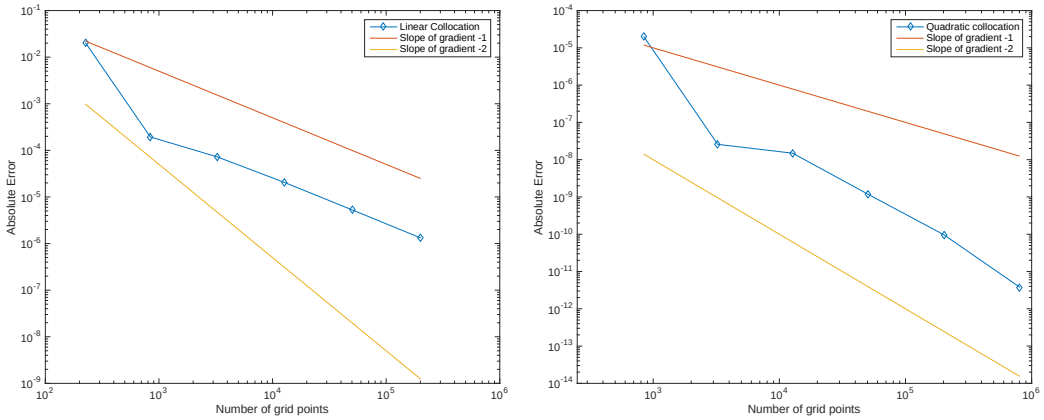


Figure 4.7: An illustration of the refinement procedure for a random triangulation.

each triangle into four smaller triangles, as illustrated in figures 4.4 and 4.5. Note that boundary nodes were fixed in all of our experiments in order to implement the periodicity of the problem more easily. Our results are displayed in Figure 4.6. In particular, we see that in contrast to our earlier results, the geometric convergence breaks down and we recover linear convergence for the linear collocation technique, as expected; whilst in the case of quadratic collocation, we retain the quadratic convergence observed when deploying a regular, Cartesian grid.

The last domain that we considered was a random one, which was constructed by perturbing, at random, the interior points of a Cartesian mesh, such as the ones described above (see Figure 4.1). Note that boundary nodes were fixed constant in order to implement the boundary conditions more easily. More specifically, each interior node was perturbed by 10% relative to the spatial discretisation distance, Δx, in a random direction obtained by the `rand` function in MATLAB, that is



(a) Linear collocation (b) Quadratic collocation

Figure 4.8: Convergence of linear and quadratic collocation when computing the integral in (4.3) using a random triangulation as illustrated in Figure 4.7.

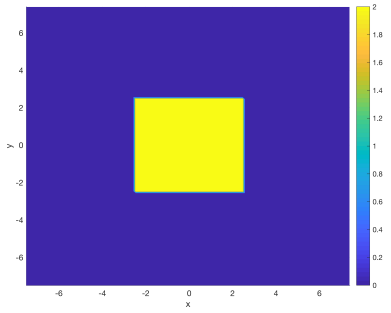


Figure 4.9: Initial condition for u on the periodic square.

`Alpha=0.1` in the following, `P=P+dx*Alpha*(2*rand(length(P),1)-1)`. As before, numerical errors were approximated by comparing the numerical solution of (4.3) at the same grid point across a range of increasingly fine meshes (see Figure 4.7 for an illustration of the refinement procedure), with the initial Cartesian grid consisting of $N_0 = 81$ nodes. Results for linear and quadratic collocation are shown in Figure 4.8, from which it is clear that the two methods produce linear and quadratic convergence, respectively. Note that we have repeated this analysis for a number of grid points and found near identical behaviour.

Next, we solved Equation (4.1) using the collocation techniques described in Chapter 3 on both regular and irregular meshes. In the case of the Cartesian mesh we also solved using trapezoidal and FFTs, for comparative purposes. In all cases, the neural activation u was initially set equal to 2 in a rectangular area centred at the origin – see Figure 4.9. After spatial discretisation, we integrated the resulting

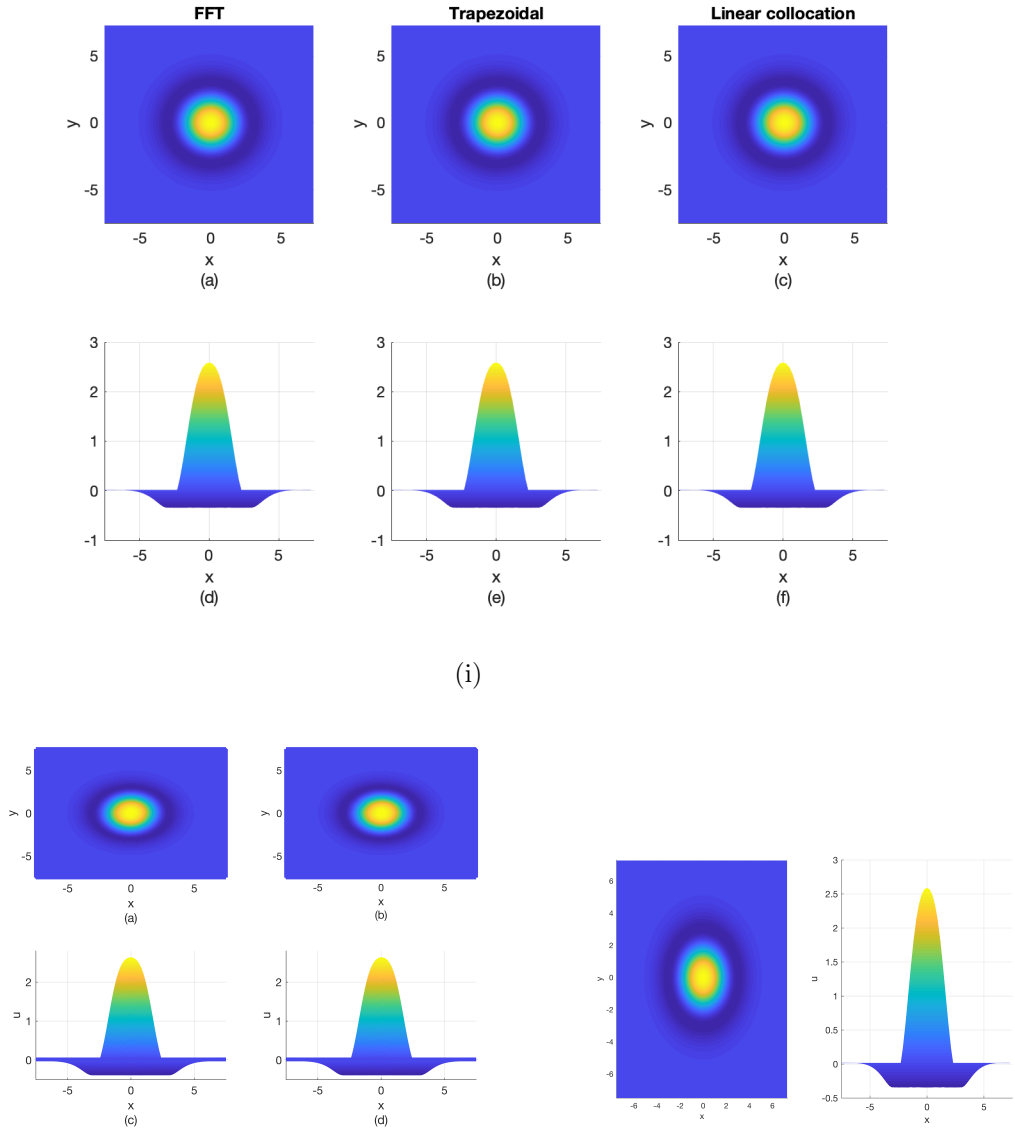


Figure 4.10: (i) Solitary bump solutions obtained when implementing FFTs, trapezoidal and linear collocation on the Cartesian grid based triangulation. (ii) Solitary bump solutions on the random triangulation perturbed by; (a)&(c) 10% and (b)&(d) 20% when implementing linear collocation. (iii) Solitary bump solution on the DistMesh general triangulation.

system of ODEs for $T = 250$ using the built-in MATLAB routine `ode45`, with absolute and relative tolerances set to $1\mathrm{e}-06$. In the case of linear collocation the ODEs are given by

$$\frac{\mathrm{d}u_n(\mathbf{v}_i,t)}{\mathrm{d}t} = 2A\sum_{k=1}^{n}\mathrm{Area}(\triangle_k)\int_{\sigma} w(\mathbf{v}_i, T_k(r,s))S\left(\sum_{j=1}^{3} u(\mathbf{v}_{k,j},t)l_j(r,s)\right)\mathrm{d}r\mathrm{d}s. \tag{4.6}$$

The model parameters were set equal to $A = 1.5, h = 0.8$ and $\beta = 5.0$. The corresponding systems of ODEs when using FFTs and the trapezoidal method are given in Appendix B.

Figure 4.10(i) shows a solitary bump solution centred at $x = y = 0$, for the trapezoidal, FFT and linear collocation methods, using a regular grid on $n_v = 4225$ nodes resulting in a triangulation with $n = 8192$ elements. As expected from our previous analysis, all three methods are in excellent agreement, converging to the same solution up to machine precision. When moving to more general meshes, we find that in order to accurately reproduce the solutions shown in Figure 4.10(i) requires considerably more mesh points. For example, implementing linear collocation (FFTs and trapezoidal can not be deployed on these more general meshes) using a DistMesh grid requires some $n_v = 11094$ nodes and $n = 22188$ triangles, an increase of 162% (in terms of nodes) on that for a regular discretisation, in order to obtain results within $1e-08$, as measured by the infinity norm. Such a solution is shown in Figure 4.10(iii). Note that solutions obtained on the general meshes are interpolated, via the `griddata` function in MATLAB, onto the Cartesian grid in order to perform the error analysis. To further interrogate our ability to replicate such solutions we have solved Equation (4.1) on a random mesh produced by perturbing the nodes of an initial Cartesian mesh, as described earlier. Figure 4.10(ii) shows solutions obtained on two perturbed grids; in the first, nodes have been perturbed by 10%, whereas in the second nodes have been perturbed by 20%. Note that the triangulations that result are highly irregular and as a result require an increase in the number of nodes to $n_v = 16641$ in order to obtain results with accuracy of $1e-08$, as measured by the infinity norm, in both cases.

Next, we deployed the pseudo arclength technique in order to determine the behaviour of the bump solutions as important model parameters are varied. In particular, starting from the initial condition $(\mathbf{u}_0, \lambda_0)$, given by the stationary bump solution derived above, we solved the n_v nonlinear scalar equations

$$F(\mathbf{u}, \lambda) = 0, \tag{4.7}$$

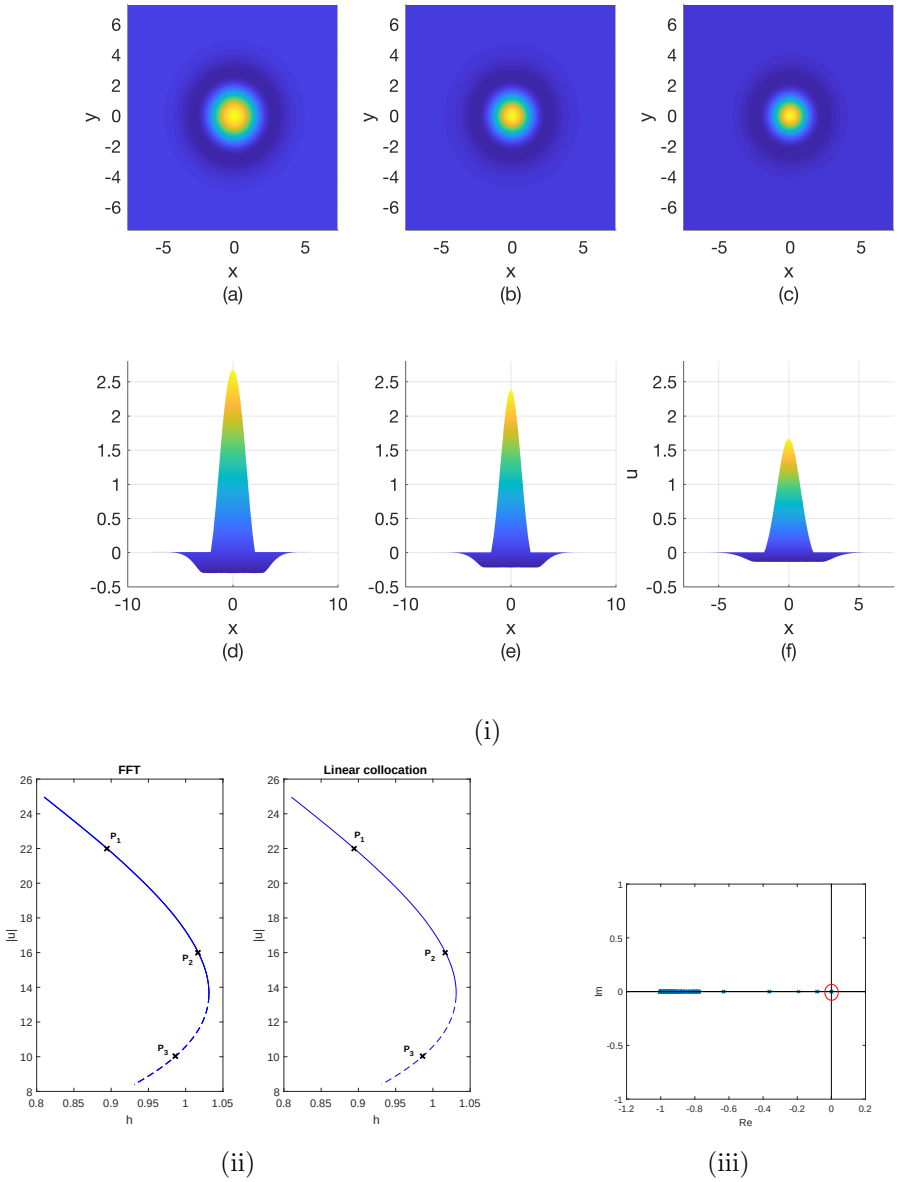


Figure 4.11: (i) Three different solutions as highlighted along the solution branch in Figure 4.11(ii): (a) & (d) point P_1 (stable), (b) & (e) point P_2 (stable) and (c) & (f) point P_3 (unstable). (ii) Solution branches as the parameter h is varied when implementing FFTs and linear collocation to solve Equation (4.7). (iii) The eigenvalues of the Jacobian matrix evaluated at the bifurcation point $h^* \approx 1.03$.

which result from the spatial discretisation of the right hand side of Equation (4.1), alongside the pseudo arclength condition

$$(\mathbf{u}_1 - \mathbf{u}_0)^T \dot{\mathbf{u}}_0 + (\lambda_1 - \lambda_0)^T \dot{\lambda}_0 - \Delta s,$$

in order to determine a nearby solution $(\mathbf{u}_1, \lambda_1)$. We use the `nsoli` algorithm from the book by Kelley [123], which is a Newton-Krylov method. We set the maximum number of Newton steps equal to 40, set the absolute and relative tolerances (*i.e.* the termination criteria) equal to $1e-10$, and implement the GMRES Krylov method. In all of our experiments the continuation step-size $\Delta s = 0.2$. To initialise the search we set

$$\mathbf{u}_1^{(0)} = \mathbf{u}_0 + \dot{\mathbf{u}}_0 \Delta s \quad \text{and} \quad \lambda_1^{(0)} = \lambda_0 + \dot{\lambda}_0 \Delta s.$$

Note that in the first instance, the tangent vector $(\dot{\mathbf{u}}_0, \dot{\lambda}_0)$ is computed by solving the following linear system of equations:

$$\begin{pmatrix} F_{\mathbf{u}} & F_\lambda \end{pmatrix} \begin{pmatrix} \dot{\mathbf{u}}_0 \\ \dot{\lambda}_0 \end{pmatrix} = 0,$$

where here $F_{\mathbf{u}}$ is the $n_v \times n_v$ Jacobian matrix and F_λ is the $n_v \times 1$ column vector of derivatives with respect to λ, both of which were computed using finite difference approximations [81]. However, for large systems it is more efficient to approximate the tangent vector using finite differences for additional points as follows

$$\dot{\mathbf{u}}_i \approx \frac{\mathbf{u}_i - \mathbf{u}_{i-1}}{\Delta s} \quad \text{and} \quad \dot{\lambda}_i \approx \frac{\lambda_i - \lambda_{i-1}}{\Delta s}.$$

For further details on numerical continuation see, for example, the book by Seydel [124].

The result of following solutions of (4.7) as h is varied is shown in Figure 4.11(ii). More specifically, we performed continuation using both FFTs and linear collocation obtaining, as can be readily seen from the figure, identical results. The plot shows that as h is increased a stable bump solution is destroyed in a saddle-node bifurcation; Figure 4.11(iii) plots the eigenvalues of the Jacobian matrix evaluated at the solution with $h^* \approx 1.03$, from which it is clear that a saddle-node bifurcation occurs. Note that, as is conventional, stable branches are denoted by continuous lines and unstable by dashed lines. In our experiments the continuation procedure was typically halted after computation of a small section of the unstable branch. Three different solutions, two stable and one unstable, and labelled P_1, P_2 and P_3

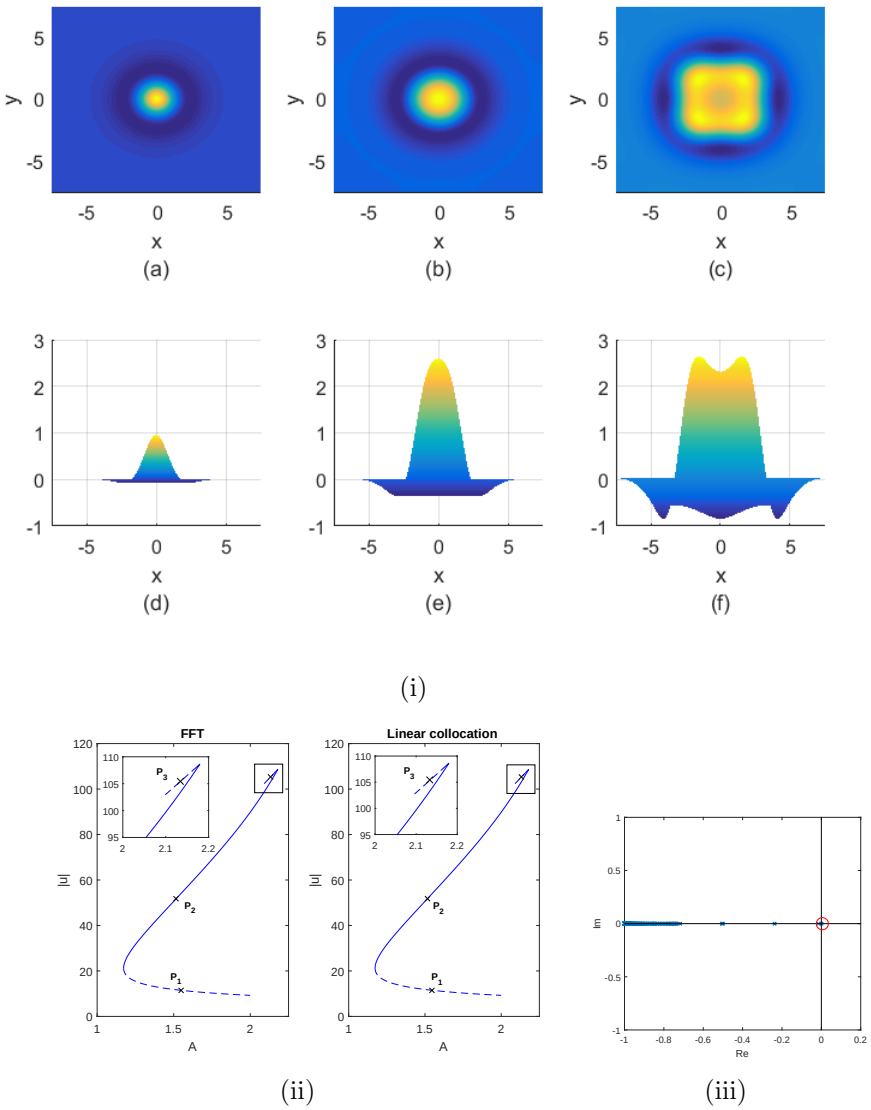


Figure 4.12: (i) The three solutions selected along the solution branch in Figure 4.12(ii): (a)&(d) point P_1 (unstable), (b)&(e) point P_2 (stable) and (c)&(f) point P_3 (unstable). (ii) Solution branches as the parameter A is varied when implementing FFTs and linear collocation to solve Equation (4.7). (iii) Eigenvalues of the Jacobian matrix evaluated at the bifurcation point $A^* \approx 1.2$.

in Figure 4.11(ii), respectively, are shown in Figure 4.11(i); in particular, we see that as we approach the turning point along the stable branch, the bump solutions reduce in size. As stated previously, h represents the firing rate threshold and as it increases the amount of neuronal activity converted to firing frequency is reduced since the contribution of the neuronal population activity must be much greater in order to exceed the threshold and 'fire'. Next, we varied the parameter A, which describes the sensitivity of nonlinear interactions in the model. The results are shown in Figure 4.12(ii). We found a saddle-node bifurcation occurring at $A^* \approx 1.2$ and $A^* \approx 2.2$, Figure 4.12(iii) plots the eigenvalues of the Jacobian matrix evaluated at the bifurcation point $A^* \approx 1.2$. Note that we obtain a near identical figure at the other bifurcation point. For unstable values of A we observe either the bump solutions decreasing in size or splitting to form various different unstable patterns. (see Figure 4.12(i) for an illustrative example).

Note that when solving (4.7) using linear collocation on more general triangulations, such as the DistMesh and random ones discussed earlier, we find that in order to obtain identical results to those in figures 4.11 and 4.12 we have to, in addition to increasing the mesh size, increase the number of Newton steps taken in the continuation algorithm by 150% (these results can be seen in Appendix C). Importantly, the above results suggest that with enough grid points and/or computational power, we can reproduce the same types of solutions as that obtained with FFT or trapezoidal methods, regardless of the underlying mesh. Moreover, early experiments suggest that deploying higher-order polynomial approximations (see the discussion in the concluding chapter) in our collocation scheme enables us to calculate the integral in (4.3) more accurately without such dramatic increases in mesh size, thus potentially circumventing the need for significant increases in computational power.

4.1.2 Torus

In this section we deploy the MMP algorithm in order to solve the NFM in Equation (4.1) on the curved surface of a torus, which unlike the previously studied case of the sphere (see for example [187]) has non-constant curvature. Note that for any given triangulation of the torus, the collocation techniques described in §3.2 can be deployed directly to solve Equation (4.1); however, implementation of the trapezoidal rule requires a regular (in the appropriate polar coordinate system) spatial discretisation of the torus, which is most easily obtained by considering the

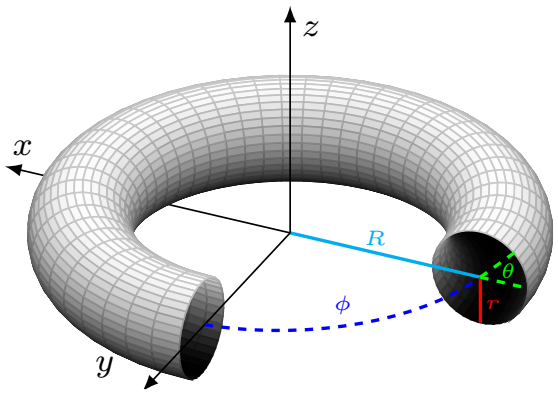


Figure 4.13: Parameterisation of a torus by coordinates (θ, ϕ).

following parameterisation of the toroidal surface:

$$(\theta, \phi) \mapsto \begin{pmatrix} (R + r\cos\theta)\cos\phi \\ (R + r\cos\theta)\sin\phi \\ r\sin\theta \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{4.8}$$

The geometrical meaning of the major curvature radius R, the minor curvature radius r, and the angles θ and ϕ are shown in Figure 4.13.

Importantly, the above parameterisation allows us to rewrite Equation (4.1) as follows:

$$\begin{aligned} \frac{\partial u(\theta,\phi)}{\partial t} = A \int_0^{2\pi} \int_0^{2\pi} w((\theta,\phi),(\theta',\phi'))S(u(\theta',\phi'))r(R + r\cos\theta')\mathrm{d}\theta'\mathrm{d}\phi' \\ - u(\theta,\phi), \end{aligned} \tag{4.9}$$

which is in a form that enables us to apply the trapezoidal rule directly to solve the integral part of the equation, *i.e.*

$$I(\theta,\phi) = \int_0^{2\pi} \int_0^{2\pi} w((\theta,\phi),(\theta',\phi'))S(u(\theta',\phi'))r(R + r\cos\theta')\mathrm{d}\theta'\mathrm{d}\phi'. \tag{4.10}$$

Note that in the above, we have used the fact that the surface area element for the torus is given by

$$\mathrm{d}\Omega(\theta,\phi) = r(R + r\cos\theta)\mathrm{d}\theta\mathrm{d}\phi,$$

which can easily be derived from the first fundamental form. It is also worth pointing out that the above integral is not a convolution integral and so we cannot use FFT techniques to solve (4.1) on a torus, or indeed on more general surfaces.

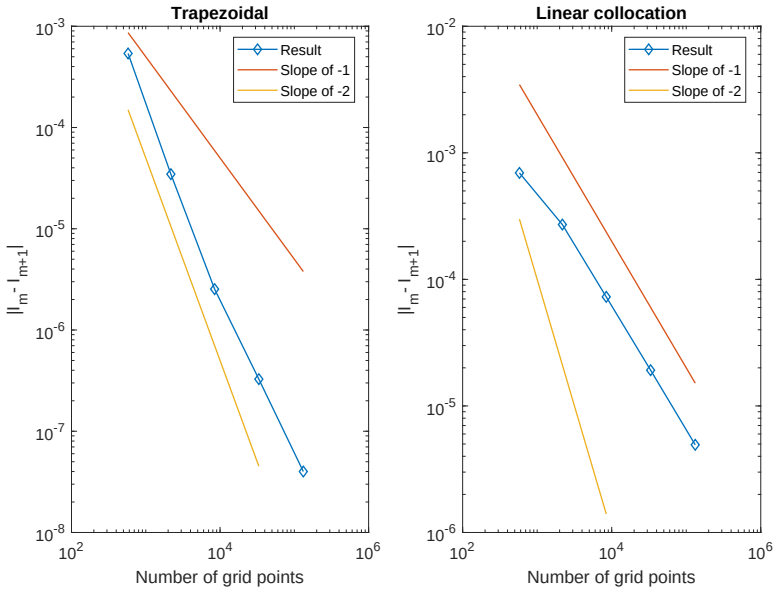


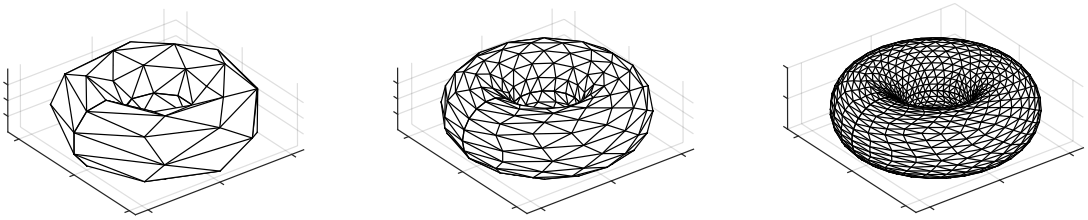
Figure 4.14: Convergence of the trapezoidal and linear collocation methods when computing the integral (4.10) on the surface of a torus using a Cartesian grid based triangulation as illustrated in Figure 4.13.

We compared the accuracy of both linear collocation and the trapezoidal rule by considering the integral in (4.10) for the case when $\Omega = T^2$, *i.e.* the closed surface of a torus, with minor radius $r = 2$ and major radius $R = 4.5$. As with our previous analysis, we set the unknown function $u(\theta, \phi) = w((\theta, \phi), (0, 0))$, that is the connectivity kernel given in (4.2), with the distance function d calculated numerically using the MMP algorithm. Starting from a regular, initial grid of $N_0 = 162$ nodes, obtained by applying the spatial discretisation

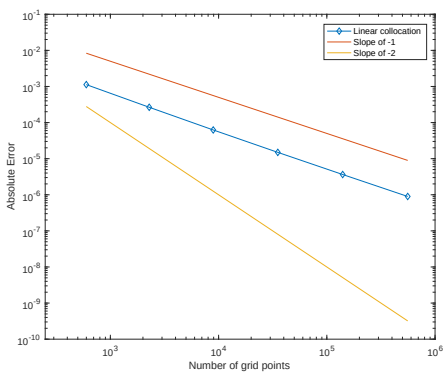
$$\begin{aligned} \theta_i &= \theta_0 + i\delta\theta, \quad i = 0, 1, \ldots, 8, \\ \phi_j &= \phi_0 + j\delta\phi, \quad j = 0, 1, \ldots, 17, \end{aligned} \tag{4.11}$$

we solved the integral on a sequence of increasingly fine meshes, in an identical manner to that described in Section 4.1.1. A regular triangulation was constructed from the rectangular tesselation (see, for example, Figure 4.13) resulting from the aforementioned grid by setting each grid point as a vertex, and subdividing each rectangular element into two triangles. The results are displayed in Figure 4.14. In particular, we see that the orders of convergence are linear for the piecewise linear collocation method and quadratic for the trapezoidal rule.

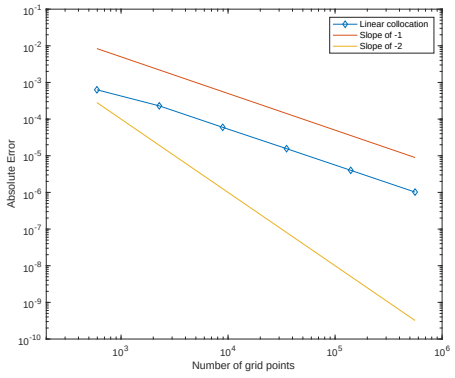
Note that we repeated the above experiment for both a general triangulation of



(a)

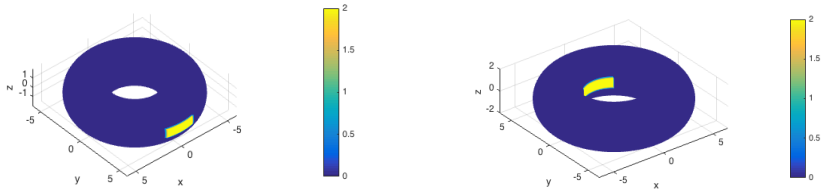


(b)



(c)

Figure 4.15: Plot (a) shows an illustration of the first three refinements of a DistMesh triangulation. Plots (b) and (c) display convergence results for linear collocation when computing the integral in (4.10) using DistMesh and random triangulations, respectively.



(a) Outside (b) Inside

Figure 4.16: Initial conditions u_0 when solving Equation (4.1) starting; (a) on the outside of the torus and (b) on the inside of the torus.

the torus, constructed using the DistMesh package, as well as a random triangulation obtained by applying a random perturbation of 20% to each of the nodes in the regular triangulation described above. Recall, that in this more general case we can not use the trapezoidal method. We performed the usual error analysis by considering a sequence of increasingly fine meshes, using the refinement procedure described earlier and illustrated in Figure 4.15(a) for the DistMesh grid. That is, we started with an initial, coarse triangulation consisting of N_0 nodes, which was equal to 164 and 162 for the DistMesh and Random triangulations, respectively; we then plotted the errors $|I_m - I_{m+1}|$ versus N_{m+1} for the two different triangulations. Our results are plotted in figures 4.15(b) and 4.15(c) for the DistMesh and random triangulation, respectively; as can be readily observed from the plots, we obtain linear convergence in both cases.

We tracked the evolution of the neural activation u from two different initial conditions: (i) a rectangular area centred on $\theta = \phi = 0$, initially set equal to 2; and (ii) a rectangular area centred on $\theta = \phi = \pi$, again, initially set equal to 2. These two initial conditions are displayed in Figure 4.16 and consist of areas of initial excitation on the outside (positive Gaussian curvature) and inside (negative Gaussian curvature) of the toroidal surface. Equation (4.1) was first solved on a regular triangulation as described above. The ODEs resulting from this spatial discretisation (see Equation (4.6) in the case of linear collocation and Appendix B in the case of the trapezoidal method) were then solved for $T = 400$ using the built-in MATLAB routine `ode45`, with absolute and relative tolerances set to $1e-06$. The model parameters were set as follows: $A = 1.5, h = 0.8$ and $\beta = 5$. Figures 4.17(i) and 4.17(ii) show stable bump solutions of Equation (4.1) centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$, respectively, for a torus with minor curvature radius $r = 2$ and major curvature radius $R = 4.5$. In both cases, figures on the right hand side show results of solving (4.1) using linear collocation on a regular grid of $n_v = 8256$ nodes, whilst

figures on the left hand side show results when employing the trapezoidal method to compute the integral in (4.1) on the same grid. These two solutions are in excellent agreement with maximal difference of the order $1e-09$.

In addition to the above experiments we also solved Equation (4.1) on both a general mesh created using the DistMesh package, and a random triangulation obtained by perturbing nodes of the uniform mesh deployed above. The grids in these general meshes required $n_v = 11094$ and $n_v = 14196$ nodes respectively in order to obtain results within $1e-07$, as measured by the infinity norm. Figure 4.17(iii) shows bump solutions on both the inside and outside of the torus, when deploying a DistMesh triangulation. Similar results found using the random triangulation can be seen in Appendix C.

Next, we considered the effect of varying the parameters h, and A on solutions of (4.1) on the torus. In addition to varying these two parameters, we also considered three different choices of the major curvature radius, R, of the torus, in order to better understand the influence of curvature on these solutions. To start, we considered a regular triangulation of the torus constructed from a Cartesian grid in θ-ϕ space and performed a comparative analysis between the trapezoidal method and linear collocation. In all of our experiments the pseudo arclength step-size was set to $\Delta s = 0.2$, and we completed 100 steps. For comparative purposes we also plot the branch obtained on the periodic square (denoted by a black line in all figures).

The results of following solutions obtained for the initial state centred at $\theta = \phi = 0$ of Equation (4.7) as h is varied are shown in Figure 4.18(i). More specifically, we performed the continuation using both trapezoidal and linear collocation for three different values of the major curvature radius:

$$R_1 = 4.5,\ R_2 = 3R_1/4 = 3.375 \ \text{ and } \ R_3 = R_1/2 = 2.25.$$

Importantly, we found that both methods were in excellent agreement as can be readily seen from the figure. In addition, we found that solutions behaved in a qualitatively similar way, regardless of the value of R, to the solutions observed on the flat, periodic domain, in that for increasing values of h the stable bump solution was destroyed via a saddle-node bifurcation. This result is confirmed by considering the plots in Figure 4.18(iii), which display the eigenvalues of the Jacobian matrix evaluated at the bifurcation point (approximately $h^* \approx 1.03$ for all values of R considered). Four different solutions, two stable and two unstable, and labelled P_1, P_2, P_3 and P_4 in Figure 4.18(i), are shown in Figure 4.18(ii); in particular, we see that as we approach the turning point along the stable branch, the bump solutions

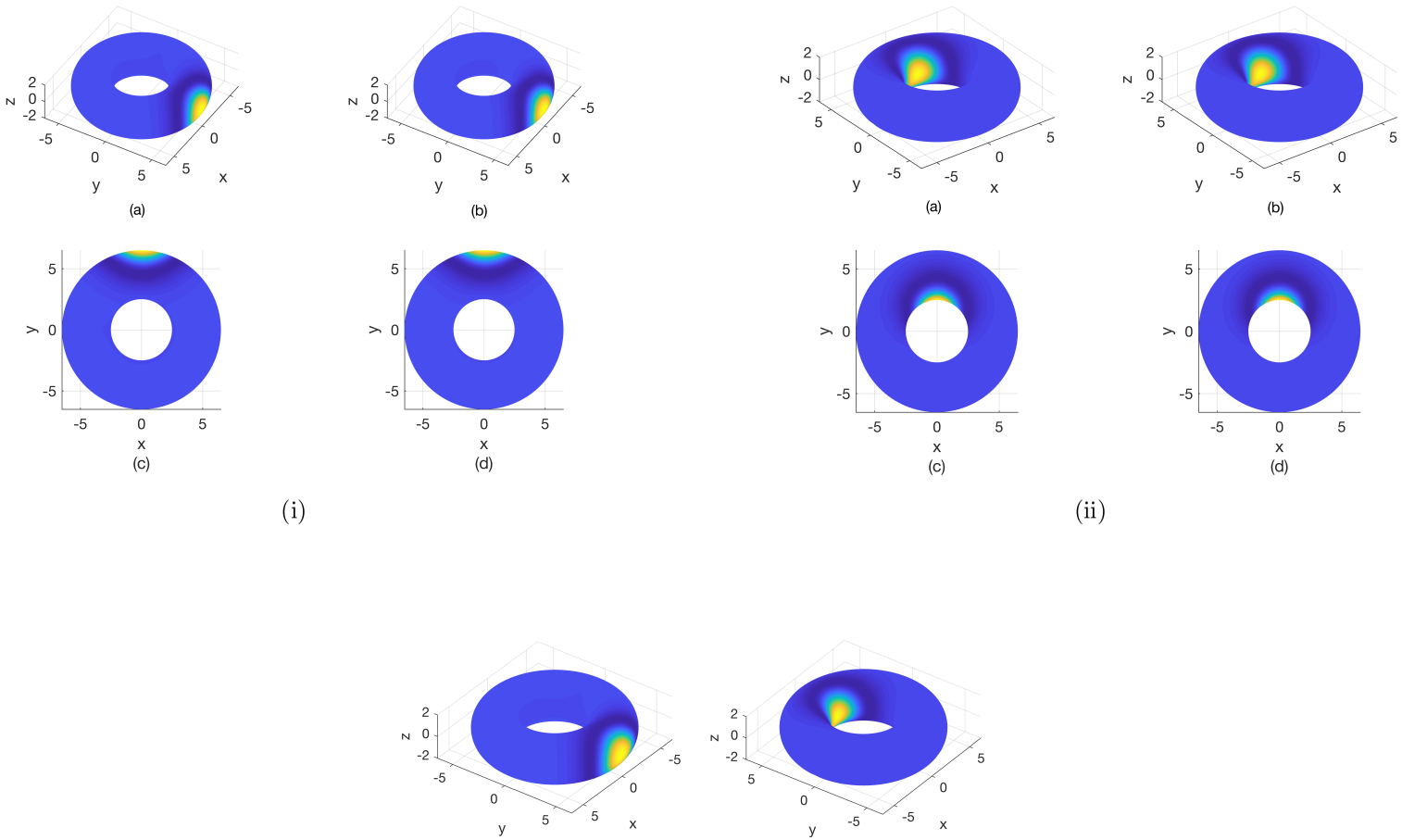


Figure 4.17: (i) Solitary bump solution found on the outside of the Cartesian grid based torus; (a)&(c) trapezoidal method, (b)&(d) linear collocation. (ii) Solitary bump solution found on the inside of the Cartesian grid based torus; (a)&(c) trapezoidal method, (b)&(d) linear collocation. (iii) Solitary bump solutions found on the DistMesh triangulation of the torus.

reduce in size before losing stability.

Next, we performed numerical continuation using the bump solution of (4.7) obtained for the initial state centred at $\theta = \phi = \pi$ as our initial condition, again as the parameter h was varied. Unlike the solutions obtained on the outer equator of the torus, the solutions on the inner equator vary depending on the value of the major curvature radius, R. Figure 4.19(i) shows solution branches for R_1 and R_2, as well as the flat, periodic square for both linear collocation and the trapezoidal rule. Again, solutions using both methods are identical, and we find that the bump solution is annihilated in a saddle-node bifurcation at $h^* \approx 1.03$. The eigenvalue plots in Figure 4.19(iv) provide further evidence for the saddle-node bifurcation. The solutions corresponding to the points P_1, P_2 displayed in figures 4.19(i) and 4.19(ii) are shown in Figure 4.19(iii). For solutions on the interior of the torus, we find that for decreasing R, the inner equator approaches a point whereby it is similar in size to the spatial extent of the connectivity function, w, at which point the stable bump solution transitions towards a stable ring solution. Note that we observe a similar behaviour on flat domains if we consider rectangular strips of width comparable to the spatial extent of the connectivity kernel and with sufficiently large aspect ratios. The bifurcation diagram for the torus with major curvature radius equal to R_3 is shown in Figure 4.19(ii) from which we observe a loss in stability of the ring solutions as we increase h; moreover, by considering the eigenvalues of the Jacobian matrix at the bifurcation point $h^* \approx 1.19$ (see Figure 4.19(v)) we see that this loss of stability is via a saddle-node bifurcation.

We then repeated the above continuation analysis for fixed h whilst varying A. Figure 4.20(i) shows the bifurcation results for solutions on the outside of the torus whilst figures 4.21(i) and 4.21(ii) show the bifurcation results for solutions on the inside of the torus. When considering solutions on the outer equator, we find that a decrease in A leads to a loss of stability for the bump solutions via the usual saddle-node bifurcation route as shown in Figure 4.20(i), as can be readily seen by the corresponding eigenvalue plot–see Figure 4.20(iii). Different outer equator solutions highlighted P_1–P_4 on the R_1 and R_3 branches in Figure 4.20(i) are shown in Figure 4.20(ii). The top row corresponds to the R_1 branch whilst the bottom row corresponds to the R_3 branch; interestingly, solutions for the R_3 branch appear to display considerably stronger neural activity. As shown the neural activity for the solutions on the R_3 branch are spreading to fill the front of the surface (this behaviour is also observed in solutions on the R_2 branch), which suggests that the curvature affects the solutions behaviour. The inner equator solutions labelled P_1

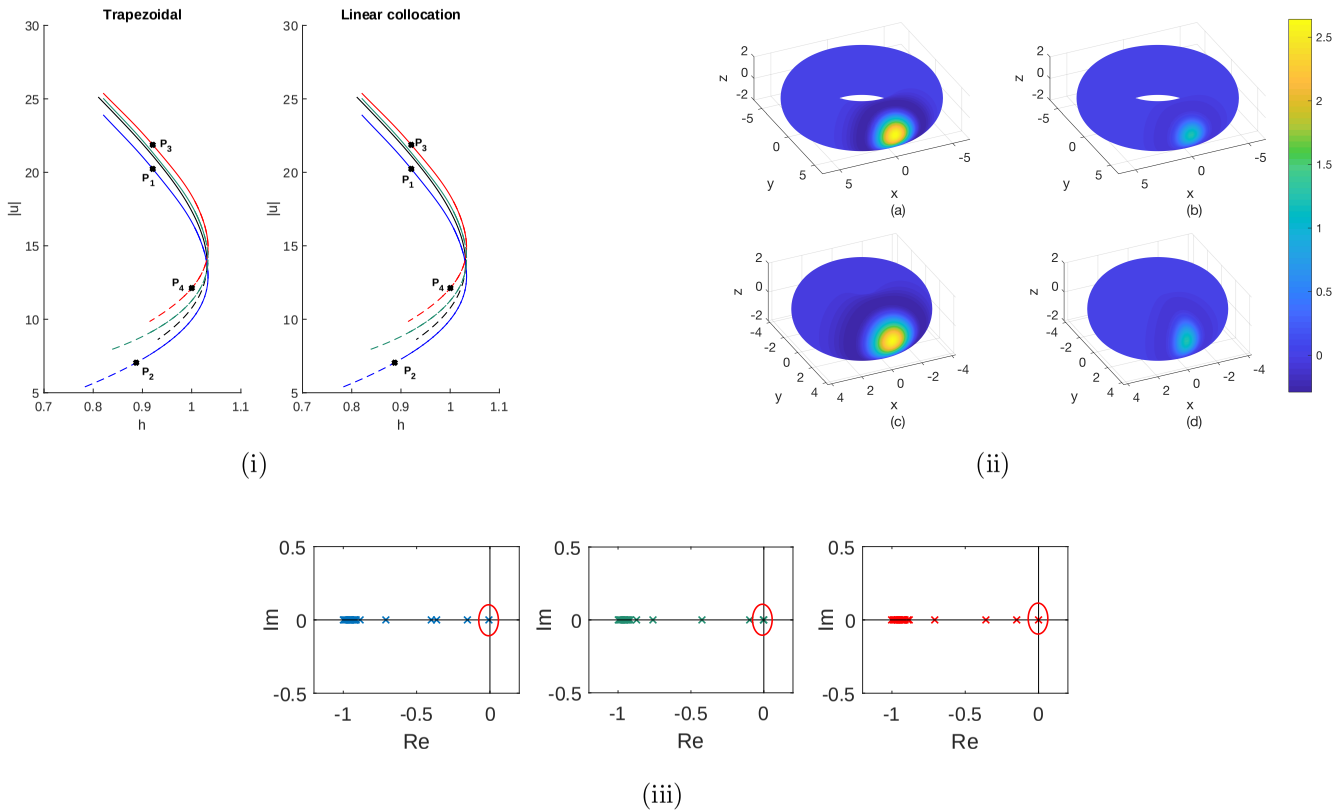


Figure 4.18: (i) Solution branches as the parameter h is varied when implementing trapezoidal and linear collocation to solve (4.7) for varying curvatures; *blue* R_1, *green* R_2, *red* R_3 and *black* the periodic square. (ii) The four solutions along the solution branch in Figure 4.18(i); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches. (iii) Plot of the eigenvalues of the Jacobian matrix evaluated at the bifurcation points h^* plotted for each branch.

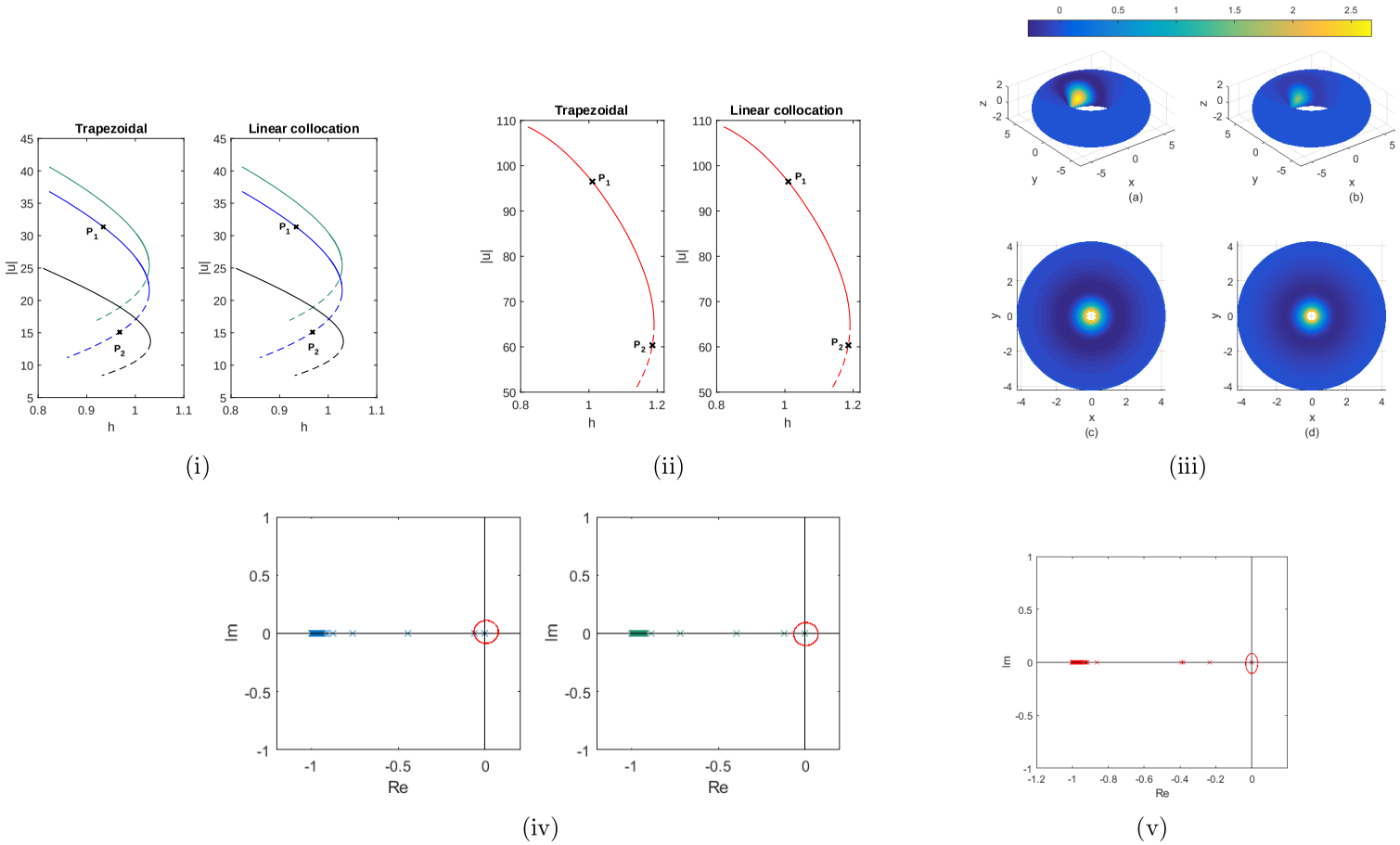


Figure 4.19: (i) Solution branches as the parameter h is varied; *blue* R_1, *green* R_2 and *black* the periodic square. (ii) Solution branches as the parameter h is varied for curvature R_3. (iii) Solutions (a)&(b) P_1 and P_2 from Figure 4.19(i) and (c)&(d) P_1 and P_2 from Figure 4.19(ii). (iv) Eigenvalues of the Jacobian matrix evaluated at the bifurcation point for each branch. (v) Plot of the eigenvalues of the Jacobian matrix evaluated at the bifurcation point.

and P_2 on the R_1 branch in Figure 4.21(i) and P_3 and P_4 on the R_3 branch in Figure 4.21(ii) are displayed in Figure 4.21(iii). The top row corresponds to the R_1 branch solutions and the bottom row to the R_3 branch solutions. We again found that ring solutions can form on the interior equator for curvature radius R_3.

4.2 Summary

In this chapter we have employed collocation techniques to solve a neural field model on both a periodic square and the curved, two-dimensional surface of a torus. Importantly, in the case of the periodic square, we found that collocation techniques are capable of replicating stationary bump solutions found using more standard techniques, such as Fourier based methods or the trapezoidal rule, using general, non-Cartesian meshes, more akin to the types of meshes derived from modern neuroimaging studies. This result, coupled with efficient numerical techniques for computing geodesic distances on triangulated surfaces, allows us to extend these algorithms with confidence to determine solutions of neural field models on curved geometries, such as the torus considered herein. Note that this is the first time that a NFM such as Equation (4.1) has been solved on a curved geometry for which no analytic formulae for geodesic distance exists. Using numerical simulations, we have explored the extent to which curvature influences the bump solutions admitted by NFMs on the torus and found the effects to be minimal. We found solutions to be qualitatively similar for all choices of major radius of curvature considered in our experiments, although there was some evidence for increased neural activity on domains with greater curvature. It is worth noting, however, that our analysis was limited to just three different choices of major radius of curvature, whilst the minor radius of curvature was held fixed, and so it would be premature to conclude that curvature has no effect on stationary bump solutions at this time. Also, whilst the results concerning curvature were rather limited, our analyses in this section provides us with considerable confidence in our numerical techniques moving forward to the next chapter.

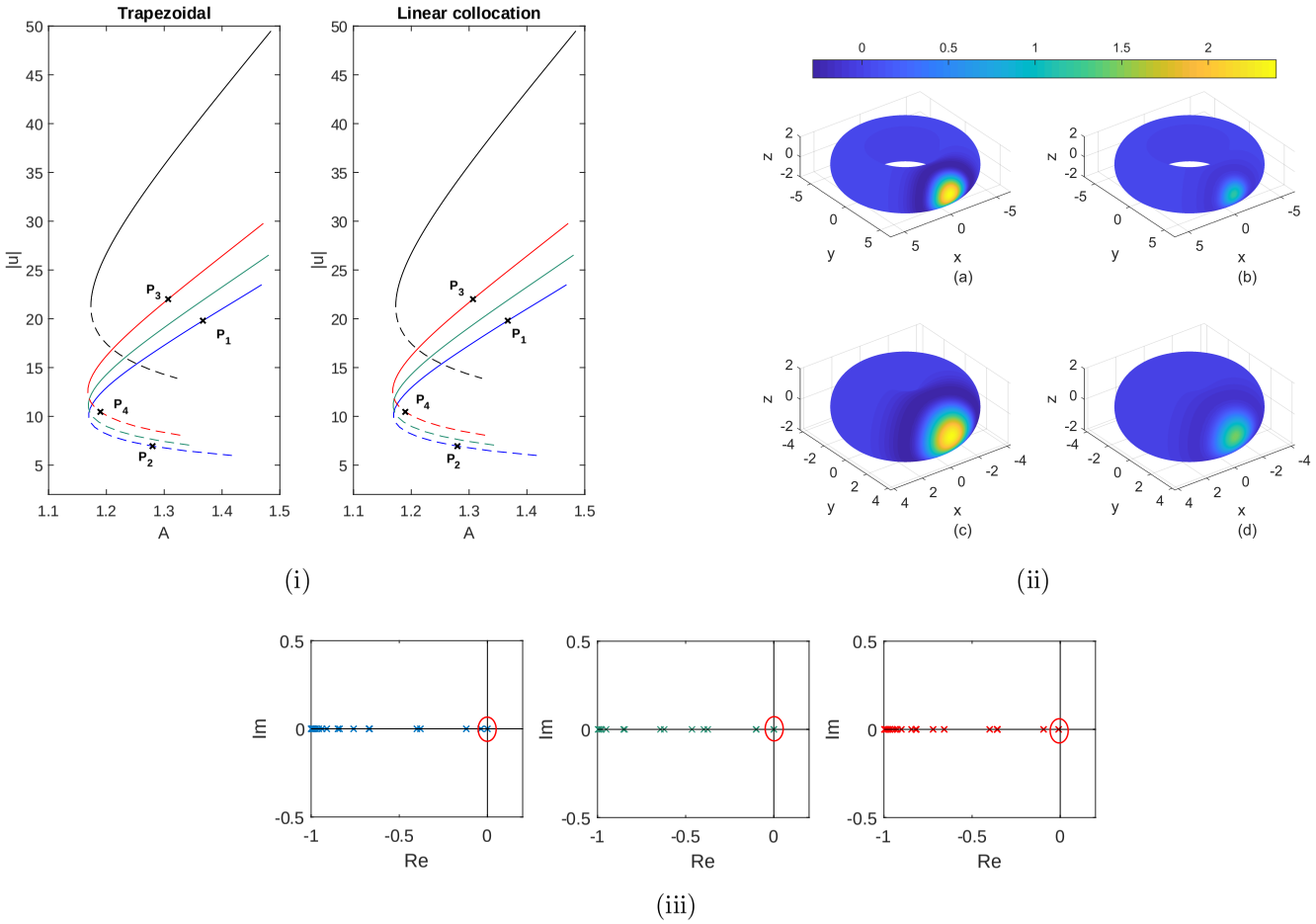


Figure 4.20: (i) Solution branches as the parameter A is varied for varying curvatures; *blue* R_1, *green* R_2, *red* R_3 and *black* the periodic square. (ii) The four solutions corresponding to points P_1–P_4 highlighted on the solution branches in Figure 4.20(i); the solutions displayed in figures (a)&(b) P_1 and P_2 on the R_1 branch (c)&(d) P_3 and P_4 on the R_3 branch. (iii) The eigenvalues of the Jacobian matrix evaluated at the bifurcation points A^*.

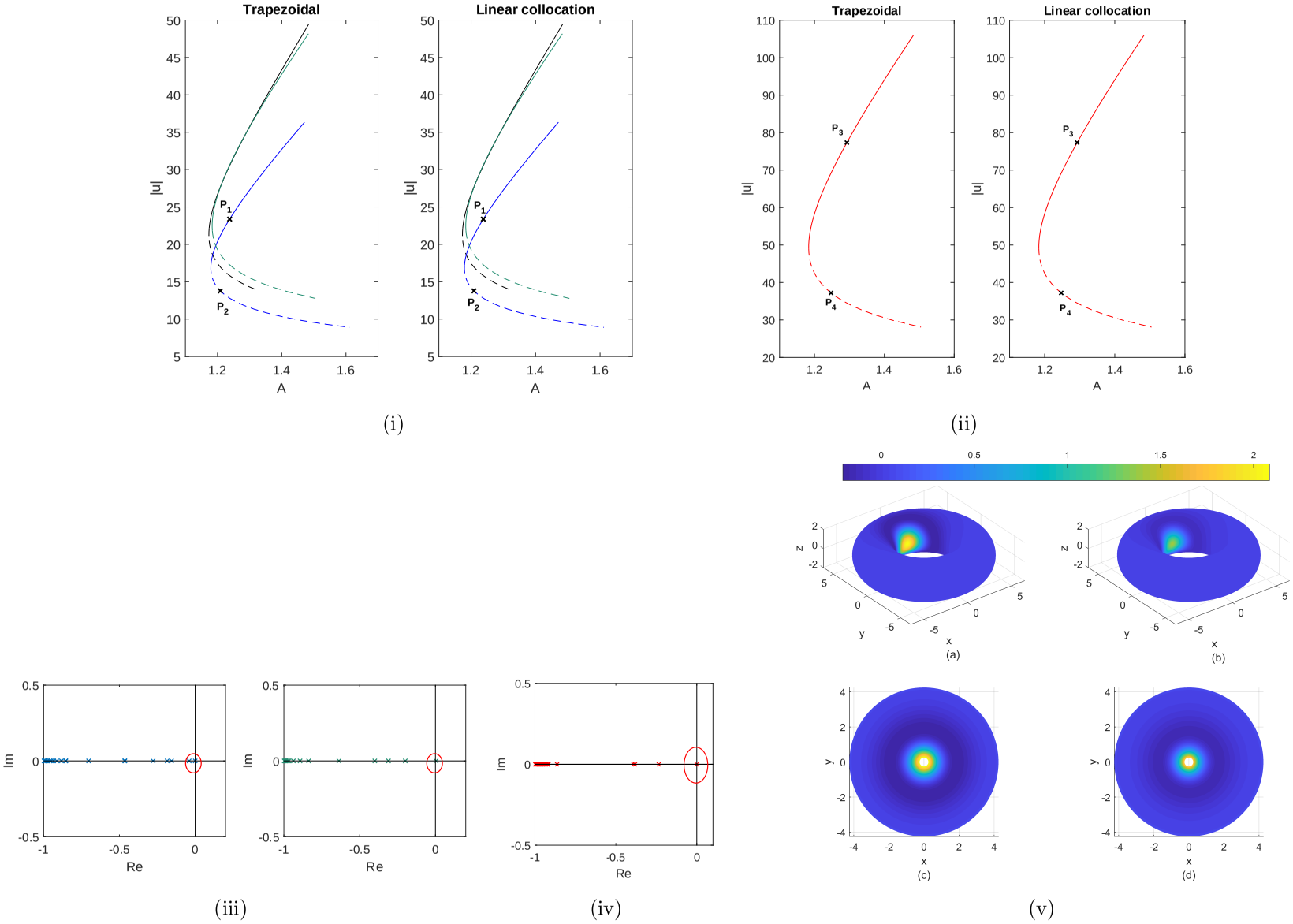


Figure 4.21: (i) Solution branches as the parameter A is varied for varying curvatures; *blue* R_1, *green* R_2 and *black* the periodic square. (ii) Solution branches as the parameter A is varied on the torus with curvature R_3. (iii) The eigenvalues of the Jacobian evaluated at the bifurcation points on the R_1 and R_2 branches. (iv) The eigenvalues of the Jacobian evaluated at the bifurcation point on the R_3 branch. (v) Solutions (a)&(b) P_1 and P_2 in Figure 4.21(i) (c)&(d) P_3 and P_4 in Figure 4.21(ii).

Chapter V

An adaptive neural field model

In this chapter we consider the following extension of the Amari equation

$$\begin{aligned}\frac{\partial u(\mathbf{x},t)}{\partial t} &= A\int_{\Omega} w(\mathbf{x},\mathbf{x}')S(u(\mathbf{x}',t))d\Omega(\mathbf{x}') - u(\mathbf{x},t) - a(\mathbf{x},t),\\ \tau\frac{\partial a(\mathbf{x},t)}{\partial t} &= Bu(\mathbf{x},t) - a(\mathbf{x},t).\end{aligned} \tag{5.1}$$

The above is an example of an adaptive neural field model, in that it includes an additional recovery variable that provides negative feedback to the system, and importantly, unlike the Amari equation studied in Chapter 4, it supports both stationary and travelling bump solutions. This will allow us to investigate the extent to which moving patterns of neural activity are influenced by the geometric features of the underlying domain Ω.

The parameters B and τ represent the strength and time-scale of the adaptation variable a. The connectivity kernel w and firing rate function S are kept the same as in the previous chapter, that is

$$w(\mathbf{x},\mathbf{x}') = e^{-d(\mathbf{x},\mathbf{x}')^2} - 0.17e^{-0.2d(\mathbf{x},\mathbf{x}')^2},$$

with d a suitably defined metric, and

$$S(u) = \frac{1}{1+e^{-\beta(u-h)}}.$$

More specifically, we present the results of applying our numerical techniques to solve a NFM on a flat, periodic square domain, the closed surface of a torus and the cortical surface of a rat brain. In the first two cases, we perform a comparative

analysis against more standard techniques, deploying either Fourier based methods and/or the trapezoidal rule to compute the integral in (5.1), and investigate the dependence of our results on the underlying mesh. We then consider solutions of our NFM on the folded structure of the rat brain, which allows us to highlight the extent to which cortical geometry influences travelling bump solutions of our NFM.

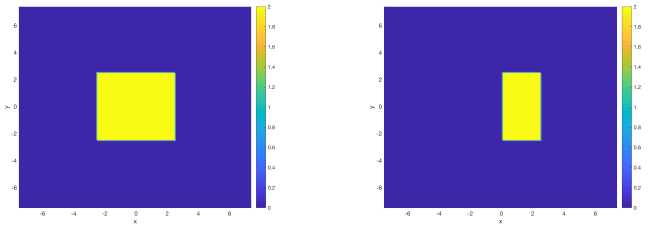
5.1 Numerical Results

In this section we present results of our numerical experiments for each of the three different domains under consideration. We start by investigating our ability to reproduce travelling bump solutions on a flat, periodic domain before moving to more general surfaces. We note here that as with the Amari equation considered in the previous chapter, the main source of error in our numerical calculations arises from the integral term in Equation (5.1); however, the analysis performed in Chapter 4 remains valid for the adapted Amari equation, at least for the periodic square and the torus. We do not conduct an error analysis on the cortical surface of the rat brain since we are restricted to the available data points; however, our investigations on both the periodic square and the torus provide us with confidence in our techniques.

5.1.1 Planar domain with periodic boundary conditions

We solved Equation (5.1) using the collocation techniques described in Chapter 3 on both regular and irregular meshes. In the case of the Cartesian mesh we also solved using trapezoidal and FFT's, for comparative purposes. In all cases the neural activation, u, was initially set equal to 2 in a rectangular area centred at the origin - see Figure 5.1(a). The initial condition for the recovery variable, a, dictates the direction in which the bump solution travels and so for the initial condition shown in Figure 5.1(b), which is set equal to 1.5 in a rectangular area shifted to the right of the initial stimulus, we obtain a bump solution that travels from right to left along the x-axis.

After spatial discretisation, we integrated the resulting system of ODEs for $T = 250$ using the built-in MATLAB routine `ode45`, with absolute and relative tolerances



(a) (b)

Figure 5.1: Initial conditions for; (a) neural activity u and (b) the recovery variable a, on the periodic square.

set to $1e-06$. In the case of linear collocation the ODEs are given by

$$\begin{aligned}\frac{\mathrm{d}u_n(\mathbf{v}_i,t)}{\mathrm{d}t} &= 2A\sum_{k=1}^{n}\mathrm{Area}(\triangle_k)\int_{\sigma} w(\mathbf{v}_i,T_k(r,s))S\left(\sum_{j=1}^{3}u(\mathbf{v}_{k,j},t)l_j(r,s)\right)\mathrm{d}r\mathrm{d}s \\ &\quad - u_n(\mathbf{v}_i,t) - a_n(\mathbf{v}_i,t), \\ \tau\frac{\mathrm{d}a_n(\mathbf{v}_i,t)}{\mathrm{d}t} &= Bu_n(\mathbf{v}_i,t) - a_n(\mathbf{v}_i,t).\end{aligned} \tag{5.2}$$

Equivalent systems of ODEs for FFTs and the trapezoidal method are given in Appendix B. In all of our simulations, model parameters were set as in [81], *i.e.*

$$A = 2.0,\ \ h = 0.8,\ \ B = 0.4,\ \ \tau = 3.0 \ \text{ and } \ \beta = 5.0.$$

This enables us to validate our solutions, at least in the case of the periodic square.

Figure 5.1.1 shows a travelling bump solution for FFTs, the trapezoidal method and linear collocation, using a regular grid on $n_v = 4225$ nodes. All three methods are in agreement, converging to the same solution up to $1e-14$. Again we find, when moving to more general meshes, that in order to accurately reproduce the solutions shown in Figure 5.1.1(i) requires considerably more node points. For example, when considering a DistMesh grid we require $n_v = 13089$ nodes, an increase of 209% on that for a regular discretisation, in order to obtain results within $1e-08$, as measured by the infinity norm. Note that the solutions on general triangulations are interpolated onto the Cartesian grid in order to conduct the error analysis. Such a solution is shown in Figure 5.1.1(iv). When considering a random mesh on $n_v = 16384$ nodes, produced by perturbing an initial Cartesian mesh by 10%, we obtain results within $1e-08$ as measured by the infinity norm, see Figure 5.1.1(iii). However, if the perturbation is increased, say to 20%, then the accuracy of the solutions can vary significantly, at least for the number mesh points we have considered. Figure

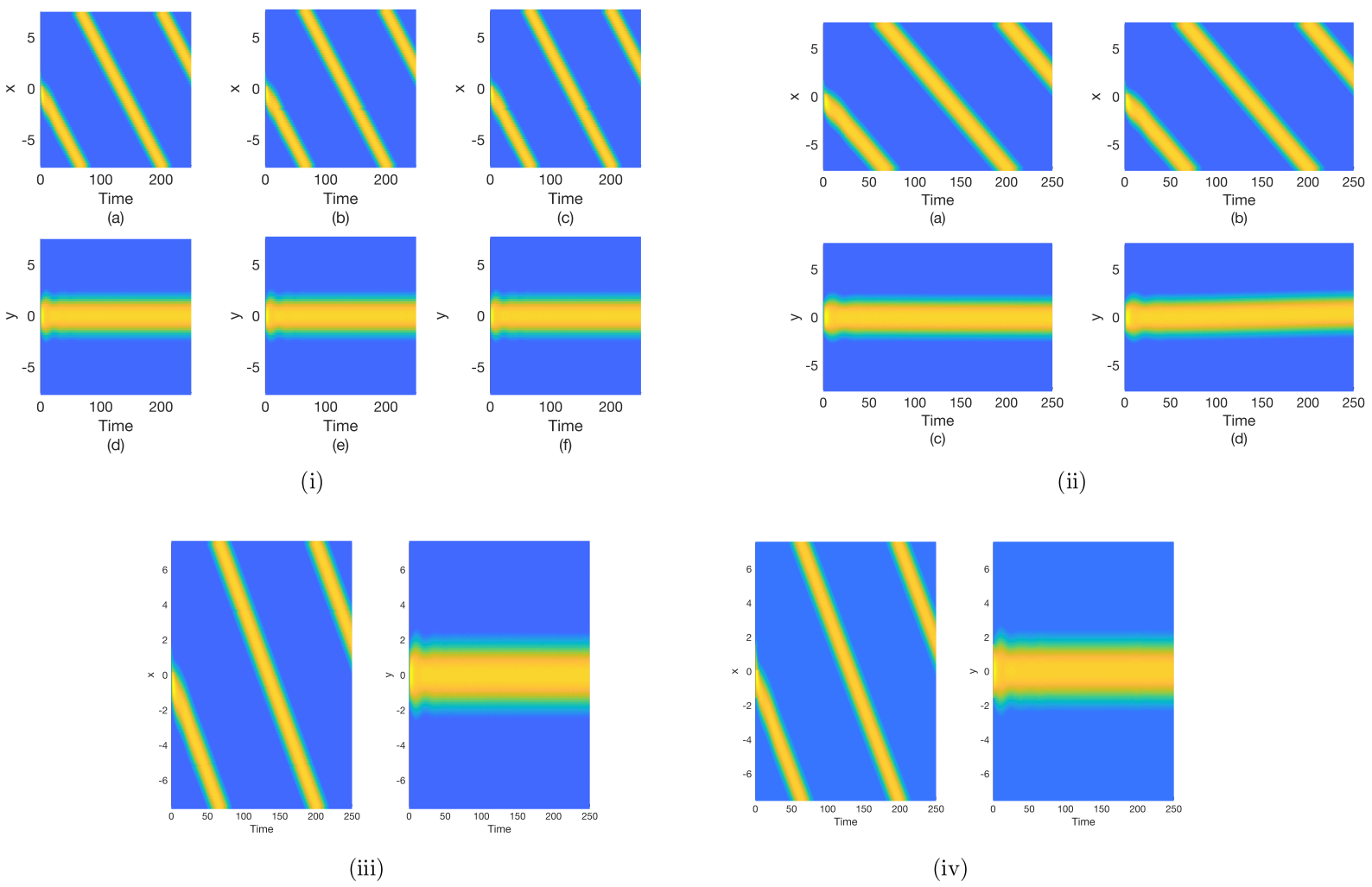


Figure 5.2: (i) Travelling bump solutions of (5.1), computed on the Cartesian grid based triangulation of the periodic square implementing; (a) & (d) FFT, (b) & (e) trapezoidal rule and (c) & (f) linear collocation. (ii) Travelling bump solutions of (5.1), solved implementing linear collocation on a random triangulation generated by perturbing the Cartesian grid based triangulation by 20%; (a) & (c) best case and (b) & (d) worst case. (iii) Travelling bump solutions of (5.1), solved implementing linear collocation on a random triangulation generated by perturbing the Cartesian grid based triangulation by 10%. (iv) Travelling bump solutions of (5.1), computed implementing linear collocation on a DistMesh triangulation of the periodic square.

5.1.1(ii) shows the 'best' and 'worst' results when running simulations on 6 different random meshes with $n_v = 16384$ nodes. The 'best' case solution achieves accuracy of $1e-08$ as measured by the infinity norm, whilst the 'worst' case scenario can be seen to drift from the line $y = 0$.

Next we performed a numerical bifurcation analysis in order to better understand the dependence of the observed travelling bump solution on certain model parameters. Note that in order to deploy the continuation procedures introduced in Chapter 3, we need to move to a travelling coordinate frame in which the bump solution is stationary. Since the bump solution we consider is travelling from right to left along the x-axis we set $\xi = x + ct$, where here c is the speed of the travelling bump. Substituting this into Equation (5.1) results in

$$\begin{aligned}\frac{\partial u(\xi,y,t)}{\partial t} &= A\int_{-L}^{L}\int_{-L}^{L} w((\xi,y),(\xi',y'))S(u(\xi',y',t))\mathrm{d}\xi'\mathrm{d}y' - u(\xi,y,t)\\ &\quad - a(\xi,y,t) - c\frac{\partial u(\xi,y,t)}{\partial \xi},\\ \frac{\partial a(\xi,y,t)}{\partial t} &= Bu(\xi,y,t) - a(\xi,y,t) - c\tau\frac{\partial a(\xi,y,t)}{\partial \xi}.\end{aligned} \tag{5.3}$$

Travelling bump solutions of (5.1) satisfy the right hand side of the above equation, that is

$$\begin{aligned}0 &= A\int_{-L}^{L}\int_{-L}^{L} w((\xi,y),(\xi',y'))S(u(\xi',y'))\mathrm{d}\xi'\mathrm{d}y' - u(\xi,y) - a(\xi,y)\\ &\quad - c\frac{\partial u(\xi,y)}{\partial \xi},\\ 0 &= Bu(\xi,y) - a(\xi,y) - c\tau\frac{\partial a(\xi,y)}{\partial \xi},\end{aligned} \tag{5.4}$$

and so solving Equation (5.4) for u, a and c results in such solutions. Note that the above situation is further complicated by the fact that translations of solutions of (5.4), in either the ξ or y direction, are also solutions. To remove these additional degrees of freedom we impose the following two conditions:

1. Solutions are required to be symmetric about $y = 0$, *i.e.*

$$u(\xi,y) = u(\xi,-y) \quad \text{and} \quad a(\xi,y) = a(\xi,-y); \tag{5.5}$$

2. We implement a scalar condition that removes translational invariance in the

ξ direction as follows:

$$u(0,0) - \frac{1}{L}\int_{-L}^{L} u(\xi,0)\mathrm{d}\xi = 0. \tag{5.6}$$

This condition ensures that the value of u at the center of the domain equals its average over ξ at $y = 0$.

Again, the motivation behind these choices are that they are the same as considered by Laing in [81] and so will allow for easier comparison of our results.

After discretisation, equations (5.4) and (5.6) result in a system of $2n_v + 1$ equations in $2n_v + 1$ unknowns – recall that the bump speed c is now included as an unknown in the problem. To perform the continuation we append the usual pseudo-arclength condition, that is

$$(\mathbf{v} - \mathbf{v}^*)^T\dot{\mathbf{v}}^* + (\lambda - \lambda^*)^T\dot{\lambda}^* - \Delta s = 0,$$

where $(\mathbf{v}^*, \lambda^*)$ is a solution of the discretisation of (5.4) and $(\mathbf{v}, \lambda)$ a nearby point on the solution branch, in order to obtain a system of $2n_v + 2$ equations

$$G(\mathbf{V}) = 0,$$

which can be solved to construct solution branches. Here we have formed the vector $\mathbf{V}$ by concatenating $\mathbf{v}$ and λ.

Perhaps the main difference between solving the Amari equation in the previous chapter and the adapted Amari equation given by (5.1) is the derivative term that appears in the right hand side of Equation (5.4). On a Cartesian grid this term is easily approximated at each point by using either finite differences or FFTs [188]; here we use the following central difference approximation:

$$\frac{\partial u(\xi,y)}{\partial \xi} \approx \frac{u(\xi+\Delta\xi,y) - u(\xi-\Delta\xi,y)}{2\Delta\xi},$$

which converges quadratically. Note, however, that when we consider more general meshes we can no longer use finite difference (or FFT) approximations of the derivative, since the classical finite difference formulae breakdown on such irregular meshes. One possible alternative is to compute the derivative directly via the projection operator

$$u_n(T_k(r,s)) = \sum_{j=1}^{f_d} u(T_k(\mathbf{q}_j))l_j(r,s), \tag{5.7}$$

that is

$$\frac{\partial u_n}{\partial \xi} = \frac{\partial u_n}{\partial r}\frac{\partial r}{\partial \xi} + \frac{\partial u_n}{\partial s}\frac{\partial s}{\partial \xi}. \tag{5.8}$$

In the case were we use linear piecewise approximations of our unknown function, such that $f_d = 3$ in (5.7), then

$$\frac{\partial u_n}{\partial r} = u(T_k(\mathbf{q}_3)) - u(T_k(\mathbf{q}_1)), \quad \frac{\partial u_n}{\partial s} = u(T_k(\mathbf{q}_2)) - u(T_k(\mathbf{q}_1)),$$

and from the inverse mapping of T_k we obtain the following relationship between r, s:

$$\begin{aligned} r(\xi_3 - \xi_1) + s(\xi_2 - \xi_1) &= \xi - \xi_1 \\ r(y_3 - y_1) + s(y_2 - y_1) &= y - y_1, \end{aligned} \tag{5.9}$$

which we can differentiate to obtain

$$\begin{aligned} \frac{\partial r}{\partial \xi} &= \alpha(y_2 - y_1), \\ \frac{\partial s}{\partial \xi} &= \alpha(y_1 - y_3), \end{aligned}$$

where

$$\alpha = \frac{1}{(\xi_2 - \xi_1)(y_2 - y_1) + (\xi_2 - \xi_1)(y_3 - y_1)}.$$

The above results in the following approximation of the derivative on a general mesh when deploying linear collocation:

$$\frac{\partial u}{\partial \xi} \approx \alpha\big[(u(T_k(\mathbf{q}_3)) - u(T_k(\mathbf{q}_1))(y_2 - y_1) + u(T_k(\mathbf{q}_2)) - (u(T_k(\mathbf{q}_1)))(y_1 - y_3)\big] \tag{5.10}$$

Unfortunately, the above approximation of the derivative converges linearly, as opposed to the quadratic convergence observed when deploying second order finite difference approximations on a Cartesian grid, and so simulations using non-Cartesian grids require a much larger number of elements in order to attain results that are in good agreement with those found on Cartesian-based grids, when using either FFTs, the trapezoidal method or linear collocation. One solution to the above problem is to consider higher-order approximations, such as quadratic collocation, and we have performed such an analysis (see Appendix D), obtaining identical results to those described below for the periodic square. However, such higher order approaches are not readily extended to curved geometries since they typically re-

quire collocation node points that are not contained on the low-order, triangulated surface approximations we consider, thus requiring higher-order surface geometry interpolation methods. Moreover, additional difficulties arise when performing a numerical bifurcation analysis on more general grids, even in the case of the flat, periodic square. For example, due to the lack of symmetry in a more general mesh, the conditions (5.5)–(5.6), that remove redundancies due to the invariance of solutions, are difficult to implement and so more general constraints are required. One way to do this is to enforce a condition that minimises the L_2 norm of u against some suitablly chosen *reference function* $\hat{u}$ [189, 190, 191, 192], that is

$$\int_{\Omega}(u-\hat{u})\frac{\partial\hat{u}}{\partial\xi}\mathrm{d}\xi = 0.$$

Of course the choice of $\hat{u}$ is crucial to the success of the method and is typically chosen so as to mimic the solutions sought [193]. For the reasons given above, we restrict the bifurcation analysis of travelling bump solutions to flat domains discretised using Cartesian-based grids.

To construct solution branches we used the Newton-Krylov algorithm, `nsoli` from [123], with the maximum number of Newton steps equal to 40 and the absolute and relative tolerances equal to $1\mathrm{e}-10$. The pseudo arclength step size is set to $\Delta s = 0.2$ in all of our experiments. The result of following solutions of (5.4) as A is varied is shown in Figure 5.3(ii), which shows A as a function of the bump speed c. Continuation was performed using both FFTs and linear collocation obtaining identical results. The plot shows that as A is varied the stable solution is destroyed in a saddle-node bifurcation; Figure 5.3(iii) plots the eigenvalues of the Jacobian matrix evaluated at the bifurcation point occurring at $A^* \approx 2.11$. Three different solutions labelled P_1, P_2 and P_3 in Figure 5.3, respectively, are shown in Figure 5.3(i); in particular, we see that as we move along the branch the bump spreads eventually splitting into two separate bumps. The above analysis was then repeated for the parameter h resulting in the bifurcation plot shown in Figure 5.4(ii). Again, the stable bump solution is annihilated at a saddle-node bifurcation, which is approximately given by $h^* \approx 0.97$. A plot of the eigenvalues of the Jacobian (see Figure 5.4(iii)) evaluated at this point provide further evidence for the existence of a saddle-node bifurcation. Three solutions along the branch are shown in Figure 5.4(i), as can be readily seen the bump solutions are shrinking as the system becomes unstable.

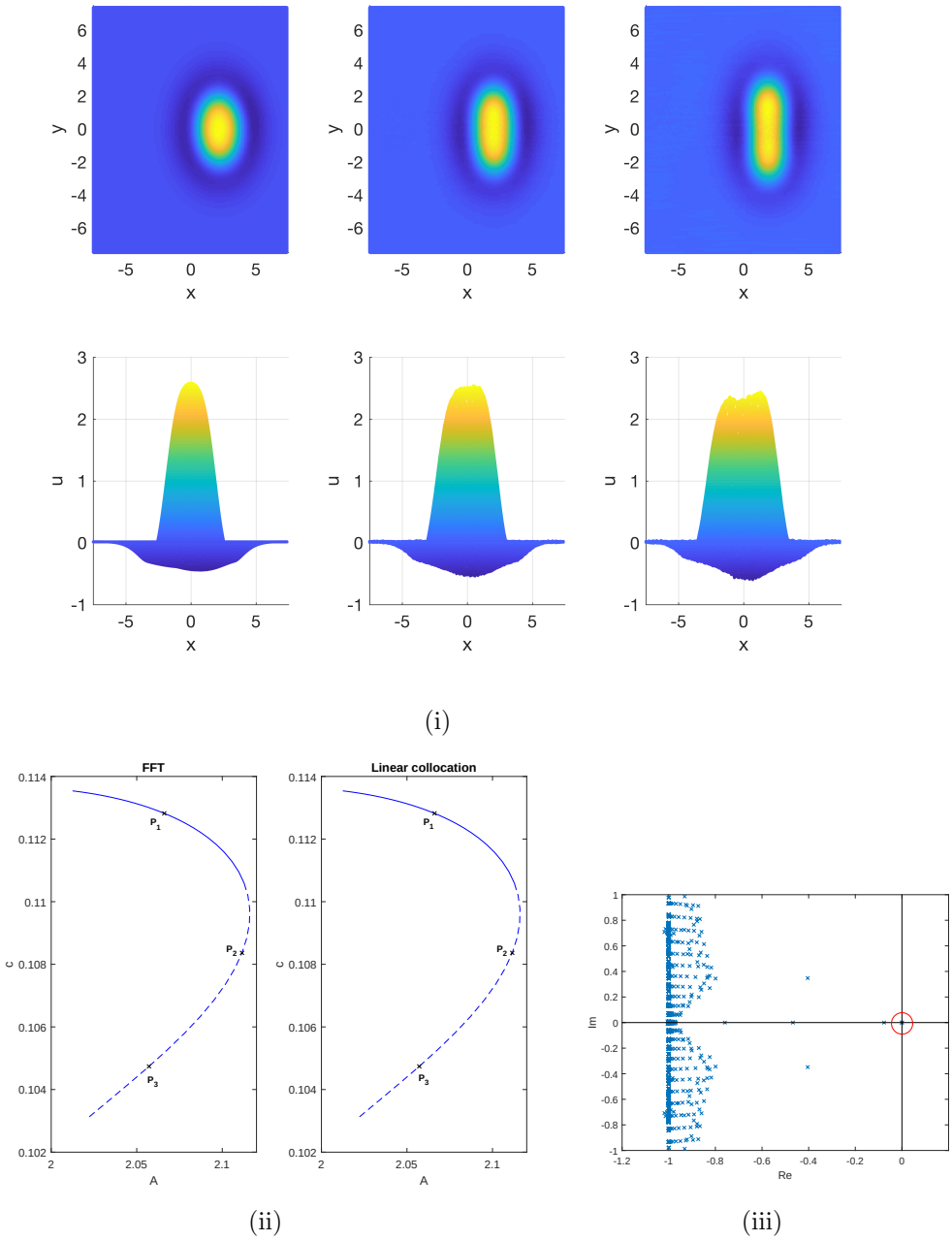


Figure 5.3: (i) Solutions from Figure (5.3)(ii); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. (ii) Solution branches as the parameter A is varied against c. (iii) The eigenvalues plotted at the saddle-node bifurcation.

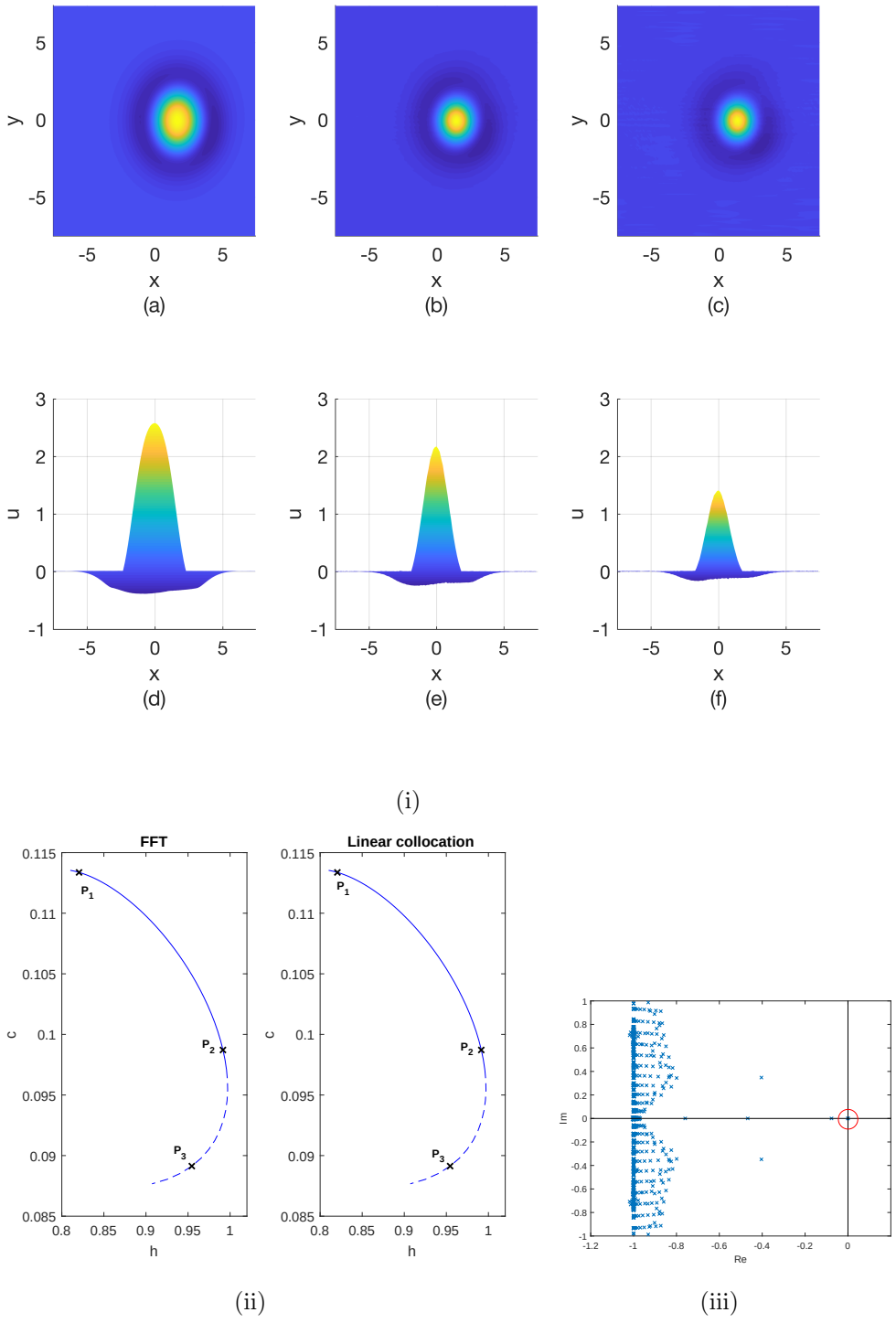


Figure 5.4: (i) Solutions from Figure (5.4)(ii); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3. (ii) Solution branches as the parameter h is varied against c. (iii) The eigenvalues plotted at the saddle-node bifurcation.

5.1.2 Torus

In this section we use the MMP algorithm in order to solve the NFM in Equation (5.1) on the curved surface of a torus. The parameterisation (4.8), described in the previous chapter, allows us to rewrite Equation (5.1) as follows

$$\begin{aligned}\frac{\partial u(\theta,\phi)}{\partial t} &= -u(\theta,\phi) - a(\theta,\phi) + \\ &\qquad A\int_0^{2\pi}\int_0^{2\pi} w((\theta,\phi),(\theta',\phi'))S(u(\theta',\phi'))r(R+r\cos\theta')\mathrm{d}\theta'\mathrm{d}\phi', \qquad (5.11)\\ \tau\frac{\partial a(\theta,\phi)}{\partial t} &= Bu(\theta,\phi) - a(\theta,\phi),\end{aligned}$$

which allows us to apply the trapezoidal rule directly in order to solve (5.1).

As before, we tracked the evolution of neural activation, u, from two different initial conditions: (i) for u initially set equal to 2 in a rectangular area centred on $\theta = \phi = 0$, and a initially set equal to 1.5 in a rectangular area shifted to the right of u (ii) for u initially set equal to 2 in a rectangular area centred on $\theta = \phi = \pi$, and a initially set equal to 1.5 in a rectangular area, shifted to the left of u. The placement of a in both cases was chosen such that the bump travelled clockwise around the torus. Equation (5.1) was first solved on a regular triangulation obtained via a uniform discretisation of the θ-ϕ plane. The resulting ODEs from this spatial discretisation (see Equation (5.2) in the case of linear collocation and Appendix B for the trapezoidal rule) were solved for $T = 400$ using the built-in MATLAB routine `ode45`, with absolute and relative tolerances set to $1e-06$. The model parameters were set as follows

$$A = 2.0,\ h = 0.8,\ B = 0.4,\ \tau = 3.0 \text{ and } \beta = 5.0.$$

Figures 5.1.2(a) and 5.1.2(b) show travelling bump solutions of Equation (5.1), for both initial conditions, implementing linear collocation on a mesh consisting of $n_v = 8256$ nodes. Comparing these solutions against those found when using the trapezoidal method (shown in Appendix C) we find that they are in excellent agreement with the maximal difference between solutions of the order $1e-08$

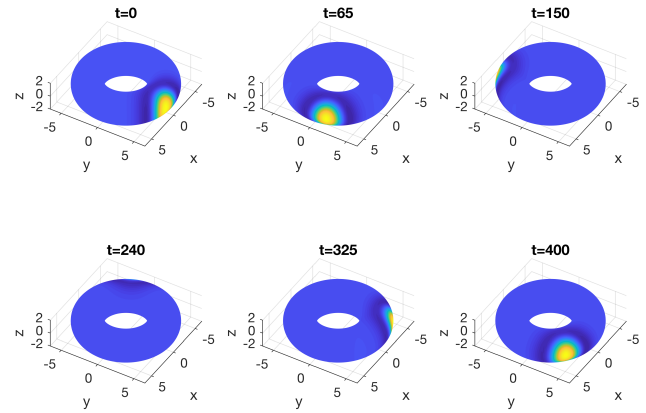
We also solved Equation (5.1) on both a general mesh generated using DistMesh, and a random triangulation obtained by perturbing the nodes of the uniform mesh by 20%. The number of nodes in each mesh increases to $n_v = 11094$ and $n_v = 17766$, respectively, in order to achieve results within $1e-07$. Note that when comparing solutions on the general triangulations to those found when implementing

the trapezoidal method we employ the MATLAB function `griddata` to interpolate the function onto the regular Cartesian based mesh. Figures 5.1.2(c) and 5.1.2(d) show travelling bump solutions on the DistMesh domain, similar results for the random domain are shown in Appendix C.

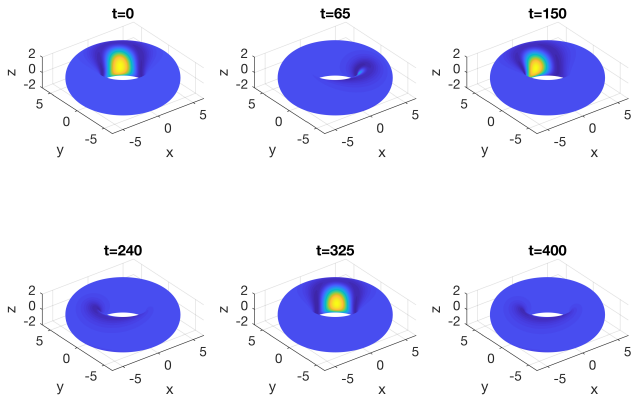
Note that the travelling bump solutions considered thus far on the torus propagate at constant speed along geodesic curves (either the inner or outer equator of the torus), and perhaps more importantly, along trajectories of constant curvature, at least along the direction of travel. However, we have also considered travelling bump solutions that propagate along non-geodesic trajectories, by considering different initial choices of the recovery variable a. Figure 5.6(a) shows the path of such a solution as it traverses the torus. In particular, we find that solutions following non-geodesic paths travel with spatially variable speed, and moreover, that the inner equator, which is the region of greatest negative curvature on the torus, acts as a barrier, in the sense that solutions travelling along non-geodesic paths are unable to pass through this region, and instead we observe oscillatory-like behaviour as the bump solution repeatedly crosses the outer equator. This behaviour is further evidenced in 5.6(b), in which we plot both the speed of the bump solution, as well as the Gaussian curvature, along the trajectory plotted in Figure 5.6(a). In Figure Figure 5.6(c) we plot the speed as a function of Gaussian curvature for each of the three tori considered in this work (*i.e.* major radii of curvature of R, $3R/4$ and $R/2$) from which it is clear that a linear relationship between curvature and bump speed exists on the torus, regardless of the radius of curvature. Note that we have also considered solutions passing through so-called meridian geodesics, *i.e.* paths of fixed azimuthal angle, and found that such bump solutions travel at constant speed and pass through the inner equator unhindered, similar to the simulations on the inner and outer equator studied earlier.

5.1.3 Cortical surface of the rat brain

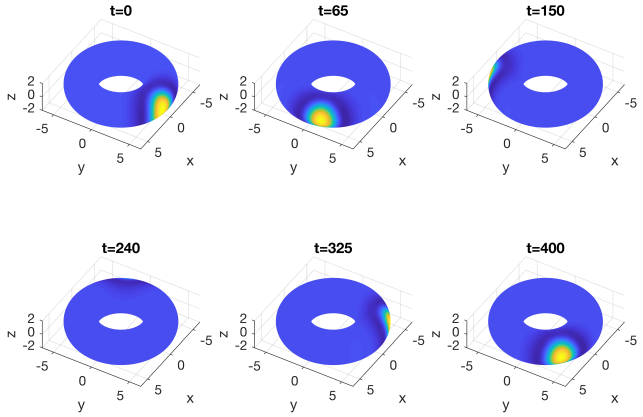
In this section we apply our methods to the curved surface of the rat brain. The spatial coordinates were obtained via the CARET software package [161] and processed using the CARET MATLAB toolbox. Restricting to the left hemisphere, we deploy the triangulation of the rat cortex provided by the CARET software with nodes positioned on the $n_v = 9623$ available data points. We tracked the evolution of neural activity u, which was initially set equal to 2 in a small region (1% of the total nodes in the mesh) surrounding a node selected at random, whilst the recovery variable a was set equal to 1.5 in an equivalently sized, partially overlapping



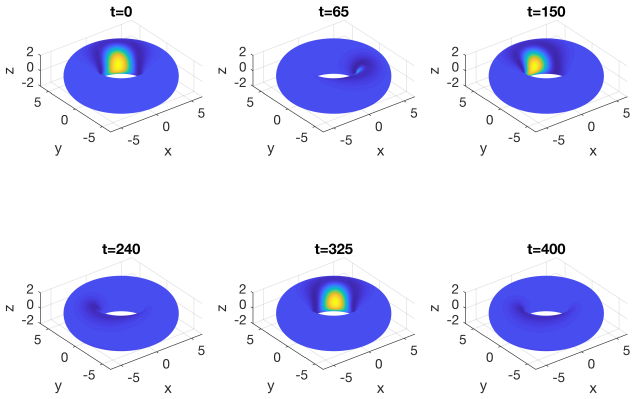
(a)



(b)

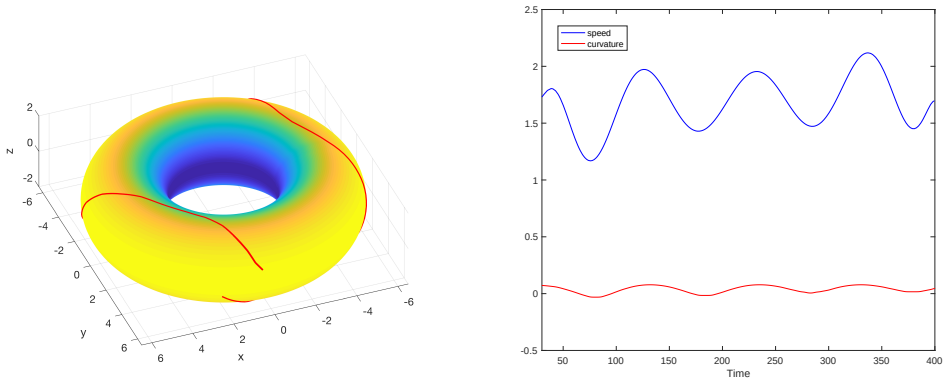


(c)

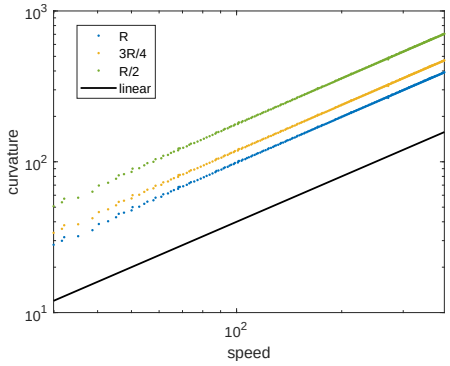


(d)

Figure 5.5: (a)Travelling bump solution found on the outside of the Cartesian grid based torus when implementing linear collocation. (b) Travelling bump solution found on the inside of the Cartesian grid based torus when implementing linear collocation. (c) The travelling bump solution found on the outside of the DistMesh triangulation of the torus implementing linear collocation. (d) The travelling bump solution found on the inside of the DistMesh triangulation of the torus implementing linear collocation.



(a) (b)



(c)

Figure 5.6: (a) The tracked path of a bump solution with non-constant speed plotted on the torus shaded with Gaussian curvature. (b) The curvature and speed of the bump plotted in time. (c) The speed plotted against curvature for the three curvatures of the torus; R, $R/2$ and $3R/4$, alongside a slope of gradient 1.

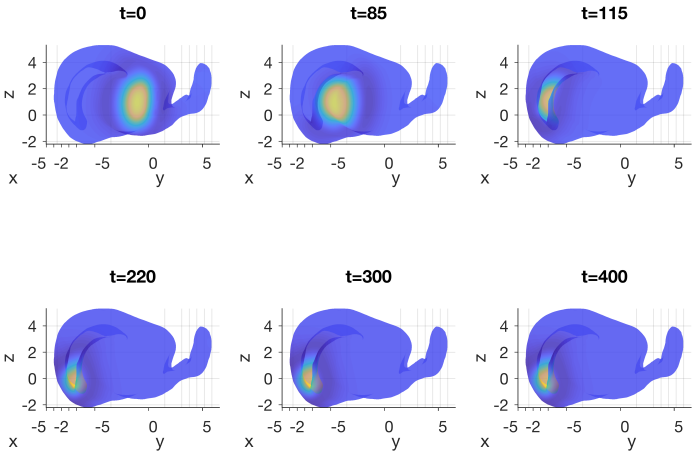
region of nodes. Since the position of the recovery variable determines the direction of propagation we have repeated this experiment a number of times, with different initial conditions, to gain an insight into how both the geometry, as well as the site, and form, of activity initiation, influences propagation travelling patterns of the localised bump solutions admitted by Equation (5.1). The ODEs in (5.2) were solved for $T = 400$ using the built-in MATLAB routine `ode45`, with absolute and relative tolerances equal to $1e-06$, and model parameters set equal to the same values as before, *i.e.*

$$A = 2.0,\ h = 0.8,\ B = 0.4,\ \tau = 3.0 \quad \text{and} \quad \beta = 5.0.$$

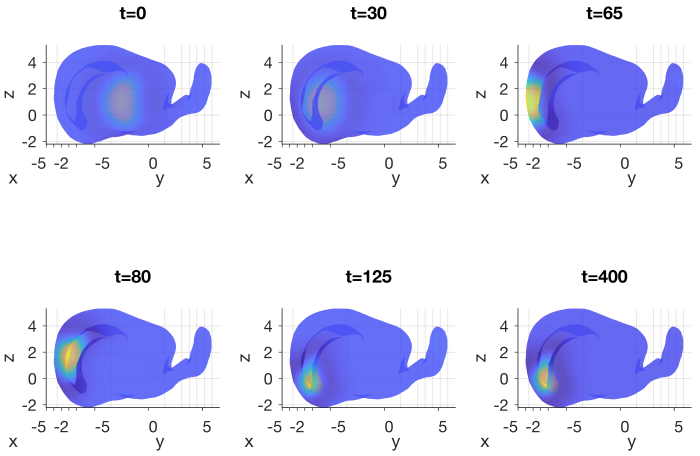
Figure 5.7 shows the progression of stable bump solutions of (5.1) at selected points for three different initial conditions. Importantly, we find regardless of the initial direction of propagation that solutions tend to one of two trapping states: either they settle on the large folded region on the underside of the rat brain (see the panel in the bottom right corner of figures 5.7(a) and 5.7(b)), or they get stuck in the transition between the main body of the brain and the tail-like structure to the rear – see Figure 5.7(c) for an example of such a solution.

We remark that unlike the solutions obtained on the torus, *all* solutions obtained for the rat brain traverse regions of both positive and negative Gaussian curvature, and as was hinted at in our experiments in the previous section, this variation in curvature would appear to have a profound affect on both the speed and direction of propagation. In Figure 5.8 we plot the path of a bump solution of (5.1) tracked at the peak (red line) alongside the geodesic path between the start and end points of the trajectory (in black) on the surface of the rat brain, which is coloured according to Gaussian curvature. Note that the initial point is taken such that transients have been removed. To compute the Gaussian curvature at each mesh point we used the Matlab Curve Fitting Toolbox to fit a cubic surface to the set of points obtained by considering the point of interest, v_i say, plus all nodes on the mesh at most a distance two (measured by counting the minimum number of edges required to step between nodes on the nearest-neighbour mesh provided by the triangulation) from v_i. We then computed the Hessian matrix (after a suitable change of coordinates) and determined the Gaussian curvature by considering its eigenvalues, which provide the principal curvatures of the fitted surface and hence an approximation to the principal curvatures of the triangulated domain at v_i (See [194] for further details on how to compute the curvature on triangulated meshes).

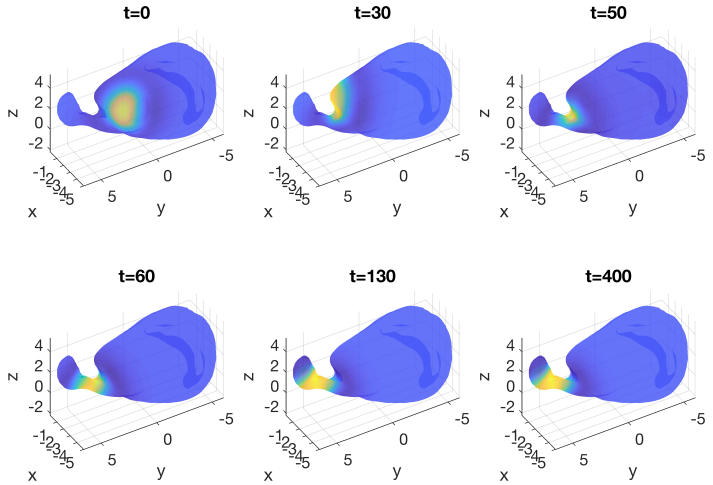
For illustrative purposes we have scaled the Gaussian curvature in Figure 5.8 to lie between minus one and one. The first point of note is that the behaviour



(a)



(b)



(c)

Figure 5.7: Travelling bump solutions found when solving (5.1) on the triangulated surface of a rat brain with different initial conditions.

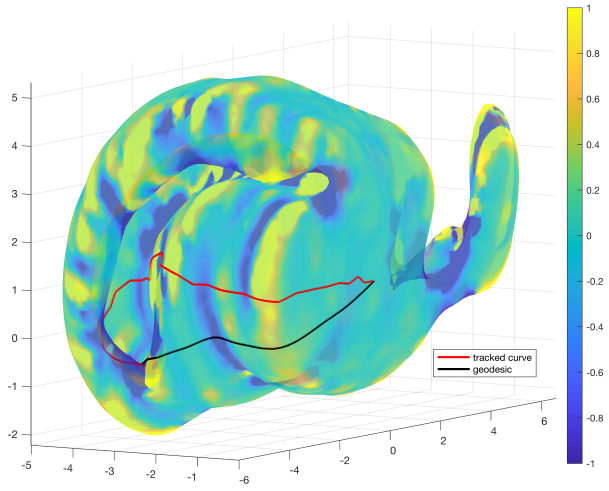


Figure 5.8: The Gaussian curvature plotted on the surface of the rat brain along with the tracked path of the solution in Figure 5.7 and the geodesic from the start point to the end point.

displayed in Figure 5.8 is typical based on our experiments, in that bump solutions do not follow geodesic paths, rather their trajectories appear to be more greatly influenced by changes in curvature. To further highlight this relation, in Figure 5.9 we have plotted the speed of the bump solution as well as the Gaussian curvature along the trajectory (the red line in Figure 5.8), from which we can see that there is an obvious correlation between peaks in bump-speed and peaks in the curvature. Our results suggest that these differences in bump-speed are due to the geometric structure of the rat brain and the presence of curved and smooth regions in the rat cortical surface. In particular, we find that areas of negative curvature cause the bump solutions to slow and even act so as to divert (solutions not shown) the path of propagation, which is reminiscent of the behaviour observed for non-geodesic trajectories on the torus, in the previous section.

As mentioned in Chapter 3, our triangulation of the rat brain includes not only the outer surface of the cortex but also the inner surface as well as subcortical regions, and so our simulations include travelling bump solutions which traverse regions outside of the cortical surface. Of course, these non-cortical regions of our surface representation do not support travelling neural activity and so such solutions are not physiologically realistic; however, from a mathematical/computational point of view these regions are perhaps the most interesting as they are the most convoluted regions of the mesh, thus they provide proof of principal of what we might expect to

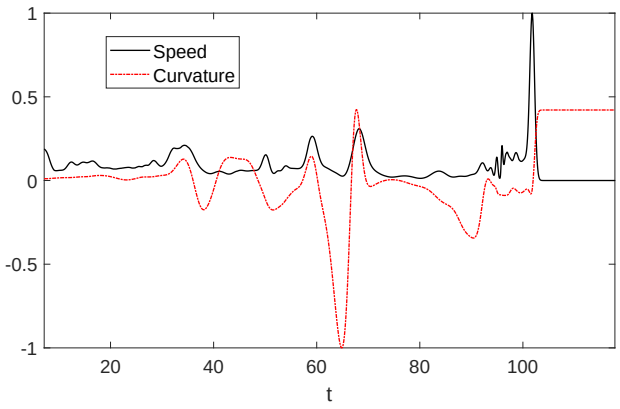


Figure 5.9: The speed of the bump solution in Figure 5.7 and the curvature along the trajectory of this solution, which is displayed by red line in Figure 5.8.

find when we extend our methods to more convoluted geometries such as the human brain.

We remark that a number of recent studies [195, 196] investigating wave propagation on cortical structures (with specific applications to cortical spreading depression) using reaction-diffusion models have forwarded Gaussian curvature as a relevant factor in both wave nucleation and propagation. Indeed, the study by Kroos *et al.* [195] found that some regions of the brain appeared to trap the propagating depolarisation waves for longer times and thus are likely to play a cruical role in CSD propagation. We note that this behaviour is reminiscent to that observed on both the torus and the rat cortex in our studies as bump solutions approach regions of negative curvature. Moreover, the study in [196] hypothesised that regions of high negative Gaussian curvature were potentially good targets for modulating pathways of localised spreading depression wave segments, and thus potentially providing stimulation protocols for clinical use. Again, early results in this direction, as discussed in this chapter, would appear to be in good agreement with these predictions.

5.2 Summary

In this chapter we have deployed our new computational technique to simulate travelling bump solutions of an adapted NFM on curved geometries and investigated the influence of the underlying mesh on these solutions. A key feature of this work is that we deploy neuroimaging data from the left hemisphere of the rat brain, alongside efficient numerical procedures for computing geodesic distances, in order

to study the behaviour of localised spot-like solutions of a non-local neural field model. Importantly, preliminary results suggest that cortical geometry influences profoundly both the propagation speed and path of such localised bump solutions, thus leading us to conclude that studies that do not account for the folded structure of the cortex risk simplifying neural activation dynamics in a potentially significant way.

Chapter VI

Conclusions and future work

6.1 Conclusions

Techniques for solving neural field models (NFMs) typically involve either transforming the problem to an equivalent differential equation [81, 83], which can be investigated using a mixture of well established analytical and/or numerical methods, or, via direct numerical simulation of the integral form of the NFM using fast Fourier transforms (FFTs) to efficiently solve the convolution integral [85]. These approaches, however, rely either on special choices for the integral kernel (*e.g.* kernels, w, whose Fourier transform is a rational function) or are restricted to uniform, periodic domains. In this thesis, we have employed collocation techniques, alongside efficient numerical procedures for computing geodesic distances on polygonal surfaces, in order to solve NFMs on realistic cortical geometries. By extending these methods to curved geometries for which no analytic formulae for geodesic distance exists, we are able to investigate, for the first time, the effect of cortical geometry on solutions of non-local field equations and thus potentially gain insight into how the cortical structure of the brain effects spreading processes and pattern formation in both healthy and diseased brains.

Despite the highly convoluted nature of the human brain, NFMs typically treat the cortex as a planar two-dimensional sheet of neurons. Indeed, current methods for directly solving NFMs usually consider a regular, Cartesian-based discretisation of a planar domain for which the analytic distance between any two points is easily computed. Thus for comparative purposes, in Chapter 4 we begin by investigating stationary bump-like solutions of perhaps the simplest NFM, namely the Amari equation, on a periodic square domain. More specifically, we investigate and

compare convergence properties of such solutions using both linear and quadratic collocation schemes on a variety of regular meshes as well as more general meshes not fixed to the Cartesian grid points. For regular grids we performed a comparative analysis against more standard techniques, in which the convolution integral is computed either by using Fourier based methods or via the trapezoidal rule. Somewhat surprisingly, we found that on regular, periodic meshes, linear collocation displays better convergence properties than quadratic collocation, and is in fact comparable with the spectral convergence displayed by both the Fourier based and trapezoidal techniques. However, for more general meshes we obtain superior convergence using higher order methods, as expected. Importantly, we found collocation techniques are capable of replicating the steady state solutions observed using more standard techniques for both regular and irregular triangulations, although additional computational costs are incurred in the latter case as a greater number of mesh points are required to achieve an accurate solution. To further improve confidence in the approach we performed a numerical bifurcation analysis of the Amari equation posed on a periodic square using both FFTs and linear collocation (the same results are obtained using quadratic collocation) and obtained identical results up to machine precision, when varying different parameters.

As a proof of principle, we then applied the new approach to solve the Amari equation on a torus. The torus was chosen as unlike the sphere (which has been studied by Coombes *et al.* [187]) no analytic formula for the distance between points exists, and moreover its Gaussian curvature is non-constant. Note that we cannot use FFTs on the torus but can deploy the trapezoidal method on a regular discretisation of the torus by moving to polar coordinates, and so a comparative analysis can be performed. As before, we considered a range of meshes (performing the comparative analysis using a regular mesh) and undertook the appropriate error analysis, again finding the different methods to be in good agreement. We considered two types of solutions: (i) a bump solution on the outer equator (*i.e.* $\theta = \phi = 0$); and (ii) a bump solution on the inner equator (*i.e.* $\theta = \phi = \pi$). By holding the minor radius of curvature constant and varying the major radius of curvature we investigated the extent to which curvature of the torus effects the stable bump solutions. In particular, we found that the bump-solutions on the outer equator behaved in a qualitatively similar way to those in the periodic square, whilst depending upon the ratio between major and minor radii, we observed either bump solutions or ring solutions on the inner equator. These differences where further highlighted by performing a numerical bifurcation analysis; note that this is the first time that

such an analysis has been performed on the torus. When varying h and A in the system we observed an increase in neural activity in the bump solutions as the major curvature radius decreased.

In Chapter 5 we considered a two-dimensional NFM that included an additional (recovery) variable and as such admits both steady state and travelling bump solutions. Here we wanted to consider the effect that curvature might have on both the path and speed of propagation for solutions of the 2D NFM. In addition to simulating this model on the periodic square and torus as before, we also deployed neuroimaging data in order to solve the model on the curved geometry of the rat cortical surface. An error analysis was again performed in the case of the periodic square and torus, and the results from the different methods were in good agreement. In the case of the periodic square we performed a numerical bifurcation analysis, providing further evidence for the accuracy of the collocation technique; however, this analysis is not currently available for travelling solutions on curved geometries due largely to fact that the bump speed now varies spatially, but also due to numerical errors that arise in this more complicated setting. Importantly, the results concerning bump propagation on the rat cortex presented in Chapter 5 suggest that cortical geometry influences profoundly both the propagation speed and path of such localised bump solutions and moreover that these propagation properties are related to changes in the Gaussian curvature of the cortical surface.

6.2 Plan of future work

Next we consider a number of possible directions for future work.

6.2.1 Computational optimisation

The use of linear collocation enables us to consider far more general geometries than the standard methods of FFTs and the trapezoidal method; however, this flexibility incurs considerable extra cost, especially in regards to FFTs. This cost increases significantly as more irregular meshes are deployed, for example, using a standard desktop computer, the computational time to solve the Amari equation on the perodic square is approximately 30 minutes for a Cartesian based triangulation, but this increases to approximately 5 hours when an irregular triangulation is used. Similar increases in computation are observed for the other geometries considered here as meshes become more irregular. There are of course a number of obvious ways in which we could speed the code up, for example, implementing Matlab's

full parallisation capabilities to take advantage of multicore processors and clusters, or even cloud based facilties such as the Amazon Web Services (AWS). To extend the ideas introduced in this thesis to human cortical geometries with physiological connectivity functions will require us to move away from Matlab and to use high level programming languages such as C/C++ or even the CUDA framework [197] which speeds numerical computations considerably by exploiting the parallelism of a GPU chip.

Another way to improve computational efficiency is to preprocess the triangulations so as to remove those triangles that are far from being equilateral; as well as making the triangulation more regular, this has the added benefit of reducing the number of mesh nodes. Of course there is a balance between the required accuracy with which we represent the geometry of the problem and the coarseness of the underlying mesh; however, more coarse representations are possible when the geometry of the problem is approximated using higher order elements – see the next section for additional details. Finally, the bifurcation analyis is particularly time consuming, taking anywhere between 10–72 hours to compute a branch, depending upon the regularity of the mesh. As well as the methods described above, which will help to reduce the computational time for the bifurcation analysis, we can consider a number of numerical tricks that have been developed for such an analysis. For example, when performing a numerical bifurcation analysis, the majority of the time is spent computing Jacobian matrices, which are required (a) in the Newton stepper; and (b) to determine the stability of the computed branch. In our work we have computed Jacobian matrices numerically using finite differences; however, in practice it is often sufficient to accurately compute the Jacobian matrix at only a few iterations and to approximate it inbetween using efficient rank-one update methods, such as Broyden's method [124]; unlike standard numerical differentiation, which requires n vector-valued function evaluations, these methods require only one vector-valued function evaluation.

6.2.2 Higher order interpolation methods

Related to the above is the concept of higher order approximation of surfaces. In our work we have considered piecewise flat approximations of our surfaces, and it is wellkown that such low order approximation can introduce a variety of errors [198, 199]. The natural next-step is to consider curved triangular elements alongside quadratic collocation, and a simple way to implement this is to use the same quadratic basis functions for the surface geometry interpolation as for the collocation method (also

known as an isoparametric mapping [200]). To obtain the curved triangle approximation to the domain we need to implement the quadratic mapping from the unit simplex σ to each $\triangle_k$ using the Lagrange quadratic basis functions. In this way we can implement quadratic collocation techniques to solve NFMs more accurately on curved domains. Importantly, such methods can attain the same levels of accuracy as the linear approximations considered in this thesis on much coarser meshes, thus reducing the computational work required to solve the NFM of interest.

Further improvements in accuracy can be attained by considering an exact representation of the surface geometry, as opposed to the typical inexact discretised representation. For example, in [201] the authors investigate the impact of considering numerical integration techniques that result in exact, curved-element-based representations of the geometry – a kind of limiting case of the quadratic-based elements described above. Importantly, the authors found that this more accurate representation required considerably fewer elements to reach the same level of accuracy in their simulations.

6.2.3 Extending the method to more convoluted geometries

Whilst we have considered curved geometries for which no analytic formula for the geodesic distance exists in this work, they are still some way off the highly convoluted nature of the human brain and so an obvious direction to extend this work is to apply it to such structures. As a first step in this direction, we have solved the neural field model in (5.1) on a down sampled version of a human cortical surface, which we obtained from the Human Connectome Project [21, 223]. Whilst this cortical structure is a rather coarse representation of the human cortex, it is significantly more convoluted than the structures considered thus far in this thesis (see Figure E.1), and so likely provides some indication as to the different types of behaviour we might expect to find when solving a NFM on such a complicated geometry.

In our preliminary investigations thus far, we have observed two different types of behaviour: the first of which is similar to that observed in the rat brain experiments, in that the travelling bump solution gets trapped in a region of large negative curvature (solutions not shown); the second behaviour that we observe happens after the bump solution enters one of the sulci, and attempts to follow its winding path. Here, we see that as the bump attempts to traverse a tight bend in the fissure, it splits into two bumps that proceed to traverse the sulcus but in opposite directions. Figure 6.1 shows snapshots of the simulation described above (additional descriptions and

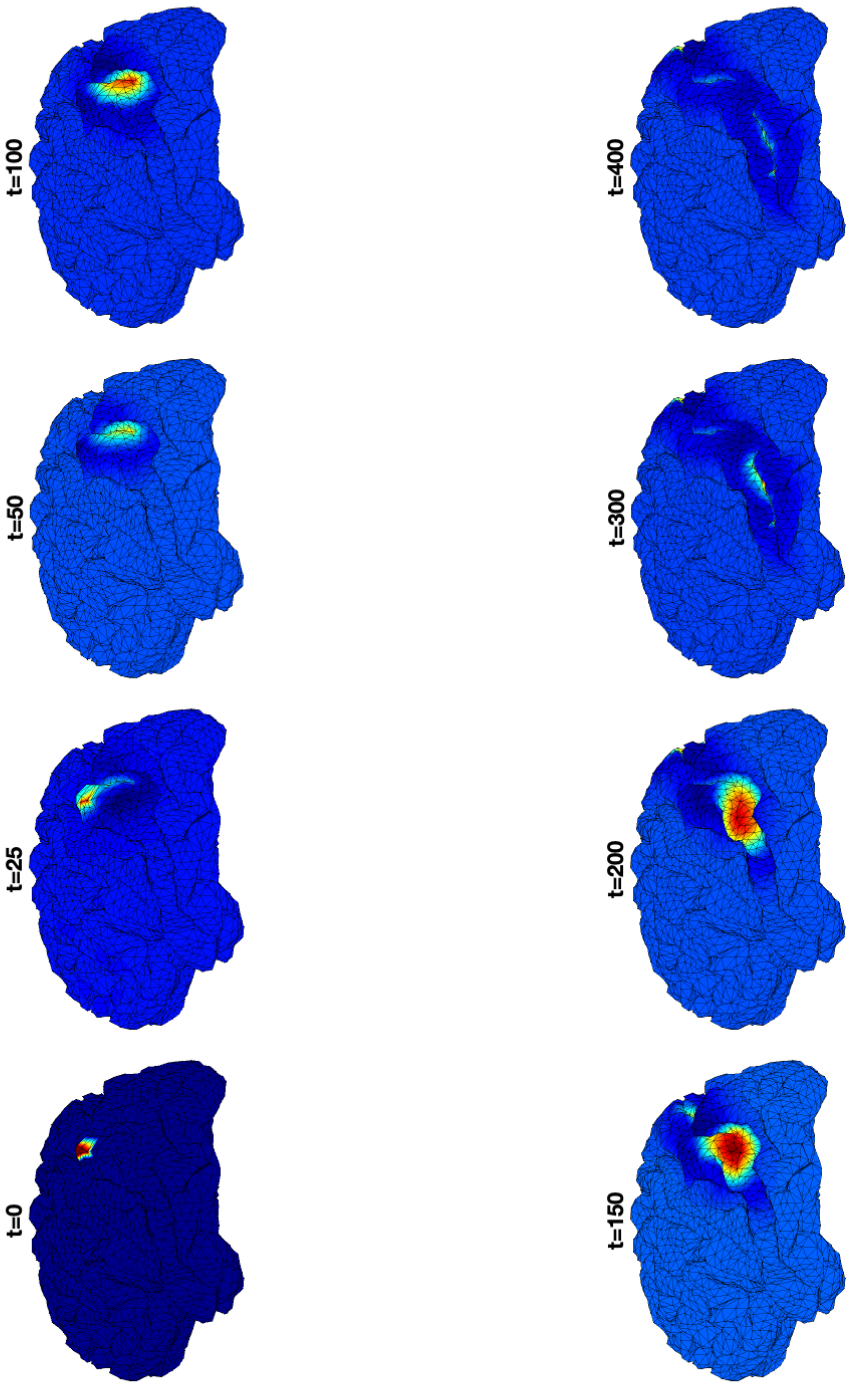


Figure 6.1: Travelling bump solution found when solving (5.1) on the triangulated surface of a human brain.

figures can be found in Appendix E).

It is worth noting that we did not observe this type of solution in our previous experiments on the torus or rat brain and so it would appear to be induced by the highly convolved architecture of the human cortex. The precise nature of these solutions and their relation to the cortical structure is an important area of future research. In addition, it would be interesting to test the dependence of the solutions observed on mesh-fineness, as well as test the effect of the connectivity function on such solutions.

6.2.4 Making the model more physiologically realistic

The NFMs considered in this thesis are relatively simplistic and various features of relevance to neural physiology have been omitted from our formulation. For example, whilst our model incorporates short-range excitatory and longer-range inhibitory connections, unlike previous models on curved geometries that required special choices of the connectivity kernels so as to reduce the NFM to a related PDE [202], the introduction of long-range white matter connections is a crucial next step. Indeed, a number of recent studies have highlighted the role of cortical geometry in shaping both local grey matter connectivity (as considered here) [153, 203], and long-range white matter connectivity architecture [154], and so including macroscale connectivity within the NFM, thus better reflecting neural mechanisms of relevance to bumps, waves and more general patterns of neural activity in the brain, is an important area of future research.

Another way to make the model more physiologically realsitic is to introduce time delays [120, 204] that arise naturally due to the finite speed of signals propagating along dendrites and axons. The models considered here assume that action potentials arrive from the presynaptic neuron to the postsynaptic neuron instantaneously, which is clearly incorrect from a biological point-of-view. Although the early formualtions of neural fields from Amari [3] and Wilson and Cowan [19] considered these delays, until more recently [108, 187, 205, 206, 207, 208] they have been disregarded due to the lack of mathematical setting [187]. An example of such a model is the time-delayed Amari equation which is given by

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = -u(\mathbf{x},t) + \int_{\Omega} w(\mathbf{x},\mathbf{x}')S(u(\mathbf{x}',t-\tau(\mathbf{x},\mathbf{x}'))\mathrm{d}\Omega(\mathbf{x}'), \tag{6.1}$$

where $\mathbf{x}$ is a point on the surface Ω, t is the time and τ denotes the corresponding delay between the fibre connecting the two points $\mathbf{x}$ and $\mathbf{x}'$. Again, w and S represent

the connectivity and firing rate functions, respectively.

The above equation reduces to a system of delay differential equations after discretisation, which can be easily solve in Matlab - although it is worth noting that this is a much more computationally challenging task than the ODE systems considered in this thesis. Also, we need to compute the delays; however, this can be done by combining distance data (geodesic in the case of a curved surface) and experimental data on the timing of signal propagation along white matter pathways.

6.2.5 Potential clinical applications

Another direction for future work is to consider the applications of NFMs to clinical experiments and neurological disorders/diseases. NFMs are capable of explaining and predicting dynamic brain activity observed in perception, behaviour and functional data [209]. For example there are emerging applications to neurological disorders such as Parkinson's disease and dementia [210]. NFMs are also able to predict bifurcations which generate primary generalized seizures [211], which has been supported by data testing [211].

Recently NFMs have been deployed to test the capabilities of both noninvasive and invasive brain stimulation techniques to amplify and accelerate recovery from stroke [217, 218]. In particular, the authors in [218] aimed to explore the alteration in brain waves due to post-stroke cortical damage, and to show that brain stimulation techniques were capable of reversing these deleterious effects. It would be an important next step to include additional physiological details into such studies, such as cortical geometry, in order to ascertain their affect on recovery and long term healthcare of stroke survivors, and also to confirm such affects by way of clinical proof of concept studies using actual patient data.

We could deploy the techniques described in this thesis to replicate the brain activity found from experimental techniques on cortical surfaces to observe any differences the geometry may have on current findings. In [212] the authors employ a NFM in order to make experimental predictions on the relationship of the excitatory-inhibitory connections which, at this larger scale, can be compared to voltage senstitive dye imaging data [213, 214, 215]. When considering techniques such as EEG and MEG, the authors in [202] provide a method for comparing the data obtained by such experimental techniques and the NFM considered. They perform their analysis on the surface of a sphere, which they then map onto the cortical surface. By replicating their analysis directly on the cortical surface using our approach, we will be able to investigate the effect geometry has on such results.

Bibliography

[1] Bard Ermentrout. Xppaut. In *Computational Systems Neurobiology*, pages 519–531. Springer, 2012.

[2] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience*, volume 10. Cambridge, MA: MIT Press, 2001.

[3] Shun-ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2):77–87, 1977.

[4] Viktor K Jirsa and Hermann Haken. Field theory of electromagnetic brain activity. *Physical Review Letters*, 77(5):960, 1996.

[5] Eric L Schwartz. *Computational neuroscience*. Mit Press, 1993.

[6] Dorian Aur and Mandar S Jog. *Neuroelectrodynamics: understanding the brain language*, volume 74. IOS Press, 2010.

[7] Terrence J Sejnowski, Christof Koch, and Patricia S Churchland. Computational neuroscience. *Science*, 241(4871):1299, 1988.

[8] Thomas Trappenberg. *Fundamentals of computational neuroscience*. OUP Oxford, 2009.

[9] Jianfeng Feng. *Computational neuroscience: a comprehensive approach*. CRC press, 2003.

[10] Louis Lapicque. Recherches quatitatives sur l'excitation electrique des nerfs traitee comme polarisation. *J. Physiol. Pathol. Gen.*, 9:620–635, 1907.

[11] Louis Édouard Lapicque. *L'excitabilité en fonction du temps: La chronaxie, sa signification et sa mesure*. Les presses universitaires de France, 1926.

[12] Anthony N Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biological cybernetics*, 95(1):1–19, 2006.

[13] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Nicholas Cain, Christof Koch, and Stefan Mihalas. Generalized leaky integrate-and-fire models classify multiple neuron types. *bioRxiv*, page 104703, 2017.

[14] Nima Soltani and Andrea Goldsmith. Directed information between connected leaky integrate-and-fire neurons. *IEEE Transactions on Information Theory*, 2017.

[15] Yann Zerlaut and Alain Destexhe. A mean-field model for conductance-based networks of adaptive exponential integrate-and-fire neurons. *arXiv preprint arXiv:1703.00698*, 2017.

[16] Raymond L Beurle. Properties of a mass of cells capable of regenerating pulses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 240(669):55–94, 1956.

[17] AL Hodgkin and AF Huxley. The components of membrane conductance in the giant axon of loligo. *The Journal of physiology*, 116(4):473–496, 1952.

[18] Hugh R Wilson and Jack D Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1, 1972.

[19] Hugh R Wilson and Jack D Cowan. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik*, 13(2):55–80, 1973.

[20] David C Van Essen, Stephen M Smith, Deanna M Barch, Timothy EJ Behrens, Essa Yacoub, Kamil Ugurbil, Wu-Minn HCP Consortium, et al. The wu-minn human connectome project: an overview. *Neuroimage*, 80:62–79, 2013.

[21] David C Van Essen, Kamil Ugurbil, E Auerbach, D Barch, TEJ Behrens, R Bucholz, Acer Chang, Liyong Chen, Maurizio Corbetta, Sandra W Curtiss, et al. The human connectome project: a data acquisition perspective. *Neuroimage*, 62(4):2222–2231, 2012.

[22] Cornelia I Bargmann and William T Newsome. The brain research through advancing innovative neurotechnologies (brain) initiative and neurology. *JAMA neurology*, 71(6):675–676, 2014.

[23] Lyric A Jorgenson, William T Newsome, David J Anderson, Cornelia I Bargmann, Emery N Brown, Karl Deisseroth, John P Donoghue, Kathy L Hudson, Geoffrey SF Ling, Peter R MacLeish, et al. The brain initiative: developing technology to catalyse neuroscience discovery. *Phil. Trans. R. Soc. B*, 370(1668):20140164, 2015.

[24] Ingo Bojak and DTJ Liley. Modeling the effects of anesthesia on the electroencephalogram. *Physical Review E*, 71(4):041902, 2005.

[25] Ole Vilhelm Larsen, Jens Haase, LR Østergaard, et al. The virtual brain project. *Stud Health Technol Inform*, 81:256–262, 2001.

[26] D Willshaw. The uk neuroinformatics node. In *Front. Neurosci. Conference Abstract: Neuroinformatics*, 2010.

[27] Michael J Keogh, Wei Wei, Ian Wilson, Jon Coxhead, Sarah Ryan, Sara Rollinson, Helen Griffin, Marzena Kurzawa-Akanbi, Mauro Santibanez-Koref, Kevin Talbot, et al. Genetic compendium of 1511 human brains available through the uk medical research council brain banks network resource. *Genome research*, 27(1):165–173, 2017.

[28] Michael Breakspear and Cornelis J Stam. Dynamics of a neural system with a multiscale architecture. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 360(1457):1051–1074, 2005.

[29] Andreas K Engel, Christian Gerloff, Claus C Hilgetag, and Guido Nolte. Intrinsic coupling modes: multiscale interactions in ongoing brain activity. *Neuron*, 80(4):867–886, 2013.

[30] Paula Sanz-Leon, Peter A Robinson, Stuart A Knock, Peter M Drysdale, Romesh G Abeysuriya, Felix K Fung, Chris J Rennie, and Xuelong Zhao. Nftsim: theory and simulation of multiscale neural field dynamics. *PLoS computational biology*, 14(8):e1006387, 2018.

[31] Krishna V Shenoy and Arto V Nurmikko. Brain enabled by next-generation neurotechnology: Using multiscale and multimodal models. *IEEE pulse*, 3(2):31–36, 2012.

[32] Lili Jiang and Xi-Nian Zuo. Regional homogeneity: a multimodal, multiscale neuroimaging marker of the human connectome. *The Neuroscientist*, 22(5):486–505, 2016.

[33] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[34] William W Lytton. Hodgkin-huxley model. *From Computer to Brain: Foundations of Computational Neuroscience*, pages 213–238, 2002.

[35] Allan L Hodgkin and Andrew F Huxley. Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of physiology*, 116(4):449, 1952.

[36] Allan L Hodgkin and Andrew F Huxley. The dual effect of membrane potential on sodium conductance in the giant axon of loligo. *The Journal of physiology*, 116(4):497, 1952.

[37] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[38] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445, 1961.

[39] Richard FitzHugh. *Mathematical models of excitation and propagation in nerve*. McGraw Hill, New York, 1966.

[40] Jinichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

[41] William Erik Sherwood. Fitzhugh–nagumo model. *Encyclopedia of Computational Neuroscience*, pages 1202–1211, 2015.

[42] Carmen Rocsoreanu, Adelina Georgescu, and Nicolaie Giurgiteanu. *The FitzHugh-Nagumo model: bifurcation and dynamics*, volume 10. Springer Science & Business Media, 2012.

[43] Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070, 2004.

[44] Catherine Morris and Harold Lecar. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical journal*, 35(1):193, 1981.

[45] Alla Borisyuk. Morris–lecar model. In *Encyclopedia of Computational Neuroscience*, pages 1758–1764. Springer, 2015.

[46] Harold Lecar. Morris-lecar model. *Scholarpedia*, 2(10):1333, 2007.

[47] Bard Ermentrout. Reduction of conductance-based models with slow synapses to neural nets. *Neural Computation*, 6(4):679–695, 1994.

[48] Kunichika Tsumoto, Hiroyuki Kitajima, Tetsuya Yoshinaga, Kazuyuki Aihara, and Hiroshi Kawakami. Bifurcations in morris–lecar neuron model. *Neurocomputing*, 69(4):293–316, 2006.

[49] Takashi Tateno and Khashayar Pakdaman. Random dynamics of the morris–lecar neural model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 14(3):511–530, 2004.

[50] Eugene M Izhikevich. Resonate-and-fire neurons. *Neural networks*, 14(6):883–894, 2001.

[51] Seung Kee Han, Christian Kurrer, and Yoshiki Kuramoto. Dephasing and bursting in coupled neural oscillators. *Physical Review Letters*, 75(17):3190, 1995.

[52] Eugene M Izhikevich et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[53] Kenneth S Cole. *Membranes, ions, and impulses: a chapter of classical biophysics*, volume 5. Univ of California Press, 1972.

[54] Jonathan Bell. Introduction to theoretical neurobiology. volume 1: Linear cable theory and dendritic structure. volume 2: Nonlinear and stochastic theories (henry c. tuckwell). *SIAM Review*, 32(1):158–160, 1990.

[55] Giacomo Indiveri. A low-power adaptive integrate-and-fire neuron circuit. In *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, volume 4, pages IV–IV. IEEE, 2003.

[56] Wulfram Gerstner. A framework for spiking neuron models: The spike response model. *Handbook of Biological Physics*, 4:469–516, 2001.

[57] Richard B Stein. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5(2):173–194, 1965.

[58] Richard B Stein. Some models of neuronal variability. *Biophysical journal*, 7(1):37–68, 1967.

[59] Thomas F Weiss. A model of the peripheral auditory system. *Biological Cybernetics*, 3(4):153–175, 1966.

[60] C Daniel Geisler and Jay M Goldberg. A stochastic model of the repetitive activity of neurons. *Biophysical journal*, 6(1):53–69, 1966.

[61] PO HSIANG CHU, John G Milton, and Jack D Cowan. Connectivity and the dynamics of integrate-and-fire neural networks. *International Journal of Bifurcation and Chaos*, 4(01):237–243, 1994.

[62] Corinna Fohlmeister, Wulfram Gerstner, Raphael Ritz, and J Leo Van Hemmen. Spontaneous excitations in the visual cortex: stripes, spirals, rings, and collective bursts. *Neural Computation*, 7(5):905–914, 1995.

[63] David Horn and Irit Opher. Solitary waves of integrate-and-fire neural fields. *Neural computation*, 9(8):1677–1690, 1997.

[64] Carlo R Laing and Carson C Chow. Stationary bumps in networks of spiking neurons. *Neural Computation*, 13(7):1473–1494, 2001.

[65] Olaf Sporns. Structure and function of complex brain networks. *Dialogues in clinical neuroscience*, 15(3):247, 2013.

[66] Olaf Sporns, Dante R Chialvo, Marcus Kaiser, and Claus C Hilgetag. Organization, development and function of complex brain networks. *Trends in cognitive sciences*, 8(9):418–425, 2004.

[67] Christopher J Honey, Jean-Philippe Thivierge, and Olaf Sporns. Can structure predict function in the human brain? *Neuroimage*, 52(3):766–776, 2010.

[68] Jaroslav Hlinka and Stephen Coombes. Using computational models to relate structural and functional brain connectivity. *European Journal of Neuroscience*, 36(2):2137–2145, 2012.

[69] Steven J Schiff. Neural control engineering, 2012.

[70] Shannon Campbell and DeLiang Wang. Synchronization and desynchronization in a network of locally coupled wilson-cowan oscillators. *IEEE transactions on neural networks*, 7(3):541–554, 1996.

[71] Jack D Cowan, Jeremy Neuman, and Wim van Drongelen. Wilson–cowan equations for neocortical dynamics. *The Journal of Mathematical Neuroscience*, 6(1):1, 2016.

[72] Jaroslav Hlinka and Stephen Coombes. Using computational models to relate structural and functional brain connectivity. *European Journal of Neuroscience*, 36(2):2137–2145, 2012.

[73] Jonathan J Crofts, Michael Forrester, and Reuben D O'Dea. Structure-function clustering in multiplex brain networks. *EPL (Europhysics Letters)*, 116(1):18003, 2016.

[74] Ben H Jansen, George Zouridakis, and Michael E Brandt. A neurophysiologically-based mathematical model of flash visual evoked potentials. *Biological cybernetics*, 68(3):275–283, 1993.

[75] Ben H Jansen and Vincent G Rit. Electroencephalogram and visual evoked potential generation in a mathematical model of coupled cortical columns. *Biological cybernetics*, 73(4):357–366, 1995.

[76] François Grimbert and Olivier Faugeras. Bifurcation analysis of jansen's neural mass model. *Neural computation*, 18(12):3052–3068, 2006.

[77] JS Griffith. A field theory of neural nets: I: Derivation of field equations. *The bulletin of mathematical biophysics*, 25(1):111–120, 1963.

[78] J So Griffith. A field theory of neural nets: Ii. properties of the field equations. *Bulletin of Mathematical Biology*, 27(2):187–195, 1965.

[79] Shun-Ichi Amari. Homogeneous nets of neuron-like elements. *Biological cybernetics*, 17(4):211–220, 1975.

[80] Paul L Nunez. The brain wave equation: a model for the eeg. *Mathematical Biosciences*, 21(3):279–297, 1974.

[81] Carlo R Laing. Numerical bifurcation theory for high-dimensional neural models. *Journal of Mathematical Neuroscience*, 4:13, 2014.

[82] Stephen Coombes, Helmut Schmidt, and Ingo Bojak. Interface dynamics in planar neural field models. *The Journal of Mathematical Neuroscience*, 2(1):1–27, 2012.

[83] Carlo R Laing and William C Troy. PDE methods for nonlocal models. *SIAM Journal on Applied Dynamical Systems*, 2(3):487–516, 2003.

[84] Carlo R Laing. PDE methods for two-dimensional neural fields. In *Neural Fields*, pages 153–173. Springer, 2014.

[85] James Rankin, Daniele Avitabile, Javier Baladron, Gregory Faye, and David JB Lloyd. Continuation of localized coherent structures in nonlocal neural field equations. *SIAM Journal on Scientific Computing*, 36(1):B70–B93, 2014.

[86] Paul C Bressloff. Spatiotemporal dynamics of continuum neural fields. *Journal of Physics A: Mathematical and Theoretical*, 45(3):033001, 2011.

[87] Reinhard Eckhorn. Neural mechanisms of scene segmentation: recordings from the visual cortex suggest basic circuits for linking field models. *Neural Networks, IEEE Transactions on*, 10(3):464–479, 1999.

[88] Harold T Kyriazi and Daniel J Simons. Thalamocortical response transformations in simulated whisker barrels. *Journal of Neuroscience*, 13(4):1601–1615, 1993.

[89] David McLaughlin, Robert Shapley, Michael Shelley, and Dingeman J Wielaard. A neuronal network model of macaque primary visual cortex (v1): Orientation selectivity and dynamics in the input layer 4cα. *Proceedings of the National Academy of Sciences*, 97(14):8087–8092, 2000.

[90] Paul C Bressloff and Stephen Coombes. Dynamics of strongly coupled spiking neurons. *Neural computation*, 12(1):91–129, 2000.

[91] Albert Compte, Nicolas Brunel, Patricia S Goldman-Rakic, and Xiao-Jing Wang. Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. *Cerebral Cortex*, 10(9):910–923, 2000.

[92] Louis Tao, Michael Shelley, David McLaughlin, and Robert Shapley. An egalitarian network model for the emergence of simple and complex cells in visual cortex. *Proceedings of the National Academy of Sciences*, 101(1):366–371, 2004.

[93] Nikola A Venkov. *Dynamics of neural field models*. University of Nottingham, 2008.

[94] Bard Ermentrout and David Hillel Terman. *Foundations of mathematical neuroscience*. Citeseer, 2010.

[95] Bard Ermentrout. Neural networks as spatio-temporal pattern-forming systems. *Reports on progress in physics*, 61(4):353, 1998.

[96] Wilfrid Rall. Cable theory for dendritic neurons. In *Methods in neuronal modeling*, pages 9–92. MIT press, 1989.

[97] Henry C Tuckwell. *Introduction to theoretical neurobiology: volume 2, nonlinear and stochastic theories*, volume 8. Cambridge University Press, 2005.

[98] André C Marreiros, Jean Daunizeau, Stefan J Kiebel, and Karl J Friston. Population dynamics: variance and the sigmoid activation function. *Neuroimage*, 42(1):147–157, 2008.

[99] Carlo R Laing. Derivation of a neural field model from a network of theta neurons. *Physical Review E*, 90(1):010901, 2014.

[100] Zachary P Kilpatrick and Paul C Bressloff. Effects of synaptic depression and adaptation on spatiotemporal dynamics of an excitatory neuronal network. *Physica D: Nonlinear Phenomena*, 239(9):547–560, 2010.

[101] Zachary P Kilpatrick and Paul C Bressloff. Spatially structured oscillations in a two-dimensional excitatory neuronal network with synaptic depression. *Journal of computational neuroscience*, 28(2):193–209, 2010.

[102] Misha Tsodyks, Klaus Pawelzik, and Henry Markram. Neural networks with dynamic synapses. *Neural Networks*, 10(4), 2006.

[103] Rodica Curtu and Bard Ermentrout. Pattern formation in a network of excitatory and inhibitory cells with adaptation. *SIAM Journal on Applied Dynamical Systems*, 3(3):191–231, 2004.

[104] Stephen Coombes, Gabriel J Lord, and Markus R Owen. Waves and bumps in neuronal networks with axo-dendritic synaptic interactions. *Physica D: Nonlinear Phenomena*, 178(3):219–241, 2003.

[105] Stephen Coombes and Markus R Owen. Evans functions for integral neural field equations with heaviside firing rate function. *SIAM Journal on Applied Dynamical Systems*, 3(4):574–600, 2004.

[106] Zachary P Kilpatrick and Paul C Bressloff. Stability of bumps in piecewise smooth neural fields with nonlinear adaptation. *Physica D: Nonlinear Phenomena*, 239(12):1048–1060, 2010.

[107] Axel Hutt and Fatihcan M Atay. Effects of distributed transmission speeds on propagating activity in neural populations. *Physical Review E*, 73(2):021906, 2006.

[108] Grégory Faye and Olivier Faugeras. Some theoretical and numerical results for delayed neural field equations. *Physica D: Nonlinear Phenomena*, 239(9):561–578, 2010.

[109] Romain Veltz. *Nonlinear analysis methods in neural field models*. PhD thesis, Paris Est, 2011.

[110] Gustavo Deco, Viktor K Jirsa, Peter A Robinson, Michael Breakspear, and Karl Friston. The dynamic brain: from spiking neurons to neural masses and cortical fields. *PLoS Comput Biol*, 4(8):e1000092, 2008.

[111] Paul C Bressloff. Stochastic neural field theory and the system-size expansion. *SIAM Journal on Applied Mathematics*, 70(5):1488–1521, 2009.

[112] Javier Baladron, Diego Fasoli, Olivier Faugeras, and Jonathan Touboul. Mean field description of and propagation of chaos in recurrent multipopulation networks of hodgkin-huxley and fitzhugh-nagumo neurons. *arXiv preprint arXiv:1110.4294*, 2011.

[113] O Faugeras, J Touboul, and B Cessac. A constructive mean-field analysis of multi population neural networks with random synaptic weights. In *Front. Syst. Neurosci. Conference Abstract: Computational and Systems Neuroscience*, 2009.

[114] Viktor K Jirsa and Hermann Haken. A derivation of a macroscopic field theory of the brain from the quasi-microscopic neural dynamics. *Physica D: Nonlinear Phenomena*, 99(4):503–526, 1997.

[115] PC Bressloff and S Coombes. Physics of the extended neuron. *International Journal of Modern Physics B*, 11(20):2343–2392, 1997.

[116] Alain Nogaret, C Daniel Meliza, Daniel Margoliash, and Henry DI Abarbanel. Automatic construction of predictive neuron models through large scale assimilation of electrophysiological data. *Scientific reports*, 6:32749, 2016.

[117] Roland Potthast and P Beim Graben. Existence and properties of solutions for neural field equations. *Math. Methods Appl. Sci*, 33(8):935–949, 2010.

[118] Kendall E Atkinson. *The numerical solution of integral equations of the second kind*, volume 4. Cambridge university press, 1997.

[119] Lokenath Debnath and Piotr Mikusiński. *Hilbert spaces with applications*. Academic press, 2005.

[120] Carlo Laing and Stephen Coombes. The importance of different timings of excitatory and inhibitory pathways in neural field models. *Network: Computation in Neural Systems*, 17(2):151–172, 2006.

[121] Carlo R Laing, William C Troy, Boris Gutkin, and G Bard Ermentrout. Multiple bumps in a neuronal model of working memory. *SIAM Journal on Applied Mathematics*, 63(1):62–97, 2002.

[122] Stephen Coombes, NA Venkov, L Shiau, Ingo Bojak, David TJ Liley, and Carlo R Laing. Modeling electrocortical activity through improved local approximations of integral neural field equations. *Physical Review E*, 76(5):051901, 2007.

[123] Carl T Kelley. *Solving nonlinear equations with Newton's method*, volume 1. Siam, 2003.

[124] Rüdiger Seydel. *Practical bifurcation and stability analysis*, volume 5. Springer Science & Business Media, 2009.

[125] Lloyd N Trefethen and JAC Weideman. The exponentially convergent trapezoidal rule. *SIAM Review*, 56(3):385–458, 2014.

[126] William H Press. *The art of scientific computing*. Cambridge university press, 1992.

[127] I Bojak, Thom F Oostendorp, Andrew T Reid, and Rolf Kötter. Towards a model-based integration of co-registered electroencephalography/functional magnetic resonance imaging data with realistic neural population meshes. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 369(1952):3785–3801, 2011.

[128] Stephen Coombes. Large-scale neural dynamics: simple and complex. *NeuroImage*, 52(3):731–739, 2010.

[129] Paul C Fife. *Mathematical aspects of reacting and diffusing systems*, volume 28. Springer Science & Business Media, 2013.

[130] Mark C Cross and Pierre C Hohenberg. Pattern formation outside of equilibrium. *Reviews of modern physics*, 65(3):851, 1993.

[131] Samuel A Ellias and Stephen Grossberg. Pattern formation, contrast control, and oscillations in the short term memory of shunting on-center off-surround networks. *Biological Cybernetics*, 20(2):69–98, 1975.

[132] Carlo R Laing. Spiral waves in nonlocal equations. *SIAM Journal on Applied Dynamical Systems*, 4(3):588–606, 2005.

[133] Xiaoying Huang, William C Troy, Qian Yang, Hongtao Ma, Carlo R Laing, Steven J Schiff, and Jian-Young Wu. Spiral waves in disinhibited mammalian neocortex. *The Journal of Neuroscience*, 24(44):9897–9902, 2004.

[134] G Bard Ermentrout, Stefanos E Folias, and Zachary P Kilpatrick. Spatiotemporal pattern formation in neural fields with linear adaptation. In *Neural Fields*, pages 119–151. Springer, 2014.

[135] MR Owen, CR Laing, and Stephen Coombes. Bumps and rings in a two-dimensional neural field: splitting and rotational instabilities. *New Journal of Physics*, 9(10):378, 2007.

[136] Paul C Bressloff, Stefanos E Folias, Alain Prat, and Y-X Li. Oscillatory waves in inhomogeneous neural media. *Physical review letters*, 91(17):178101, 2003.

[137] Mircea Steriade, Edward G Jones, and Rodolfo R Llinás. *Thalamic oscillations and signaling.* John Wiley & Sons, 1990.

[138] Philip A Schwartzkroin. *Epilepsy: models, mechanisms and concepts.* Cambridge University Press, 2007.

[139] G Bard Ermentrout and David Kleinfeld. Traveling electrical waves in cortex: insights from phase dynamics and speculation on a computational role. *Neuron*, 29(1):33–44, 2001.

[140] Y Chagnac-Amitai and BW Connors. Horizontal spread of synchronized activity in neocortex and its control by gaba-mediated inhibition. *Journal of neurophysiology*, 61(4):747–758, 1989.

[141] RD Traub, JG Jefferys, and Richard Miles. Analysis of the propagation of disinhibition-induced after-discharges along the guinea-pig hippocampal slice in vitro. *The Journal of physiology*, 472(1):267–287, 1993.

[142] David Golomb and Yael Amitai. Propagating neuronal discharges in neocortical slices: computational and experimental study. *Journal of neurophysiology*, 78(3):1199–1211, 1997.

[143] Joaquin M Fuster, Garrett E Alexander, et al. Neuron activity related to short-term memory. *Science*, 173(3997):652–654, 1971.

[144] Shintaro Funahashi, Charles J Bruce, and Patricia S Goldman-Rakic. Mnemonic coding of visual space in the monkey9s dorsolateral prefrontal cortex. *Journal of neurophysiology*, 61(2):331–349, 1989.

[145] Paul C Bressloff, Jack D Cowan, Martin Golubitsky, Peter J Thomas, and Matthew C Wiener. Geometric visual hallucinations, euclidean symmetry and the functional architecture of striate cortex. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 356(1407):299–330, 2001.

[146] Tanya I Baker and Jack D Cowan. Spontaneous pattern formation and pinning in the primary visual cortex. *Journal of Physiology-Paris*, 103(1):52–68, 2009.

[147] G Bard Ermentrout and Jack D Cowan. A mathematical theory of visual hallucination patterns. *Biological cybernetics*, 34(3):137–150, 1979.

[148] Peter Tass. Cortical pattern formation during visual hallucinations. *Journal of Biological Physics*, 21(3):177–210, 1995.

[149] Urs Braun, Sarah F Muldoon, and Danielle S Bassett. On human brain networks in health and disease. *eLS*, pages 1–9, 2001.

[150] Gustavo Deco and Morten L Kringelbach. Great expectations: using whole-brain computational connectomics for understanding neuropsychiatric disorders. *Neuron*, 84(5):892–905, 2014.

[151] Alex Fornito, Andrew Zalesky, and Michael Breakspear. The connectomics of brain disorders. *Nature Reviews Neuroscience*, 16(3):159, 2015.

[152] William W Seeley, Richard K Crawford, Juan Zhou, Bruce L Miller, and Michael D Greicius. Neurodegenerative diseases target large-scale human brain networks. *Neuron*, 62(1):42–52, 2009.

[153] Christine Ecker, Lisa Ronan, Yue Feng, Eileen Daly, Clodagh Murphy, Cedric E Ginestet, Michael Brammer, Paul C Fletcher, Edward T Bullmore, John Suckling, et al. Intrinsic gray-matter connectivity of the brain in adults with autism spectrum disorder. *Proceedings of the National Academy of Sciences*, 110(32):13222–13227, 2013.

[154] James A Henderson and Peter A Robinson. Relations between the geometry of cortical gyrification and white-matter network architecture. *Brain connectivity*, 4(2):112–130, 2014.

[155] Kendall E Atkinson. A survey of numerical methods for solving nonlinear integral equations. *The Journal of Integral Equations and Applications*, pages 15–46, 1992.

[156] Jim M Cushing. *Integrodifferential equations and delay models in population dynamics*, volume 20. Springer Science & Business Media, 2013.

[157] Stephen Coombes, Peter beim Graben, Roland Potthast, and James Wright. *Neural Fields*. Springer, 2014.

[158] JN Lyness and D Jespersen. Moderate degree symmetric quadrature rules for the triangle. *IMA Journal of Applied Mathematics*, 15(1):19–32, 1975.

[159] Arthur H Stroud. Approximate calculation of multiple integrals. pages 306–315, 1971.

[160] Per-Olof Persson and Gilbert Strang. A simple mesh generator in matlab. *SIAM review*, 46(2):329–345, 2004.

[161] David C Van Essen. Cortical cartography and caret software. *Neuroimage*, 62(2):757–764, 2012.

[162] Manfred Schroeder. The eikonal equation. *The Mathematical Intelligencer*, 5(1):36–37, 1983.

[163] Fuhao Qin, Yi Luo, Kim B Olsen, Wenying Cai, and Gerard T Schuster. Finite-difference solution of the eikonal equation along expanding wavefronts. *Geophysics*, 57(3):478–487, 1992.

[164] Anna R Bruss. The eikonal equation: Some results applicable to computer vision. *Journal of Mathematical Physics*, 23(5):890–896, 1982.

[165] James Albert Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[166] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[167] J.A. Sethian. *An Analysis of Flame Propagation*. PhD thesis, Berkeley, California, 1982.

[168] Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, 1998.

[169] James A Sethian. Curvature and the evolution of fronts. *Communications in Mathematical Physics*, 101(4):487–499, 1985.

[170] Joseph O'Rourke. Computational geometry column 35. *ACM SIGACT News*, 30(2):31–32, 1999.

[171] Joseph SB Mitchell, David M Mount, and Christos H Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.

[172] Micha Sharir and Amir Schorr. On shortest paths in polyhedral spaces. *SIAM Journal on Computing*, 15(1):193–215, 1986.

[173] Joseph O'Rourke, Subhash Suri, and Heather Booth. Shortest paths on polyhedral surfaces. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 243–254. Springer, 1985.

[174] Der-Tsai Lee. Proximity and reachability in the plane. Technical report, ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB, 1978.

[175] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[176] Eugene M Izhikevich. *Dynamical Systems in Neuroscience*. MIT Press, 2007.

[177] Bertil Hille et al. *Ion channels of excitable membranes*, volume 507. Sinauer Sunderland, MA, 2001.

[178] John Rinzel and G Bard Ermentrout. Analysis of neural excitability and oscillations. *Methods in neuronal modeling*, 2:251–292, 1998.

[179] Stefanos E Folias and Paul C Bressloff. Breathing pulses in an excitatory neural network. *SIAM Journal on Applied Dynamical Systems*, 3(3):378–407, 2004.

[180] Yuri A Kuznetsov. *Elements of applied bifurcation theory*, volume 112. Springer Science & Business Media, 2013.

[181] Linghai Zhang et al. On stability of traveling wave solutions in synaptically coupled neuronal networks. *Differential and Integral Equations*, 16(5):513–536, 2003.

[182] Todd Kapitula, Nathan Kutz, and Bjorn Sandstede. The evans function for nonlocal equations. *Indiana University mathematics journal*, 53(2004), 2004.

[183] Robert L Pego and Michael I Weinstein. Eigenvalues, and instabilities of solitary waves. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 340(1656):47–94, 1992.

[184] Björn Sandstede. Stability of travelling waves. *Handbook of dynamical systems*, 2:983–1055, 2002.

[185] CB Price. Traveling turing patterns in nonlinear neural fields. *Physical Review E*, 55(6):6698, 1997.

[186] Paul C Bressloff. Pattern formation in visual cortex. *Les Houches*, pages 477–574, 2003.

[187] Sid Visser, Rachel Nicks, Olivier Faugeras, and Stephen Coombes. Standing and travelling waves in a spherical brain model: The nunez model revisited. *Physica D: Nonlinear Phenomena*, 349:27–45, 2017.

[188] Lloyd N Trefethen. *Spectral methods in MATLAB*, volume 10. Siam, 2000.

[189] Gabriel J Lord and Vera Thümmler. Computing stochastic traveling waves. *SIAM Journal on Scientific Computing*, 34(1):B24–B43, 2012.

[190] Stephen Coombes, Carlo Laing, Helmut Schmidt, Nils Svanstedt, and John Wyller. Waves in random neural media. *Discrete and Continuous Dynamical Systems—Series A*, 32:2951–2970, 2012.

[191] Wolf-Jürgen Beyn and Vera Thümmler. Freezing solutions of equivariant evolution equations. *SIAM Journal on Applied Dynamical Systems*, 3(2):85–116, 2004.

[192] GJ Lord and Vera Thümmler. Freezing stochastic travelling waves. *arXiv preprint arXiv:1006.0428*, 2010.

[193] Clarence W Rowley, Ioannis G Kevrekidis, Jerrold E Marsden, and Kurt Lust. Reduction and reconstruction for self-similar dynamical systems. *Nonlinearity*, 16(4):1257, 2003.

[194] Timothy D Gatzke and Cindy M Grimm. Estimating curvature on triangular meshes. *International journal of shape modeling*, 12(01):1–28, 2006.

[195] Julia M Kroos, Ibai Diez, Jesus M Cortes, Sebastiano Stramaglia, and Luca Gerardo-Giorda. Geometry shapes propagation: assessing the presence and absence of cortical symmetries through a computational model of cortical spreading depression. *Frontiers in computational neuroscience*, 10:6, 2016.

[196] Markus A Dahlem, Bernd Schmidt, Ingo Bojak, Sebastian Boie, Frederike Kneer, Nouchine Hadjikhani, and Jürgen Kurths. Cortical hot spots and labyrinths: why cortical neuromodulation for episodic migraine with aura should be personalized. *Frontiers in computational neuroscience*, 9:29, 2015.

[197] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. In *ACM SIGGRAPH 2008 classes*, page 16. ACM, 2008.

[198] JC Nedelec. Curved finite element methods for the solution of singular integral equations on surfaces in r3. *Computer Methods in Applied Mechanics and Engineering*, 8(1):61–80, 1976.

[199] Meng H Lean and A Wexler. Accurate numerical integration of singular boundary element kernels over boundaries with curvature. *International journal for numerical methods in engineering*, 21(2):211–228, 1985.

[200] Siamak Amini, Paul J Harris, and David T Wilton. *Coupled boundary and finite element methods for the solution of the dynamic fluid-structure interaction problem*, volume 77. Springer Science & Business Media, 2012.

[201] Jaydeep P Bardhan, Michael D Altman, David J Willis, Shaun M Lippow, Bruce Tidor, and Jacob K White. Numerical integration techniques for curved-element discretizations of molecule-solvent interfaces. *The Journal of chemical physics*, 127(1):014701, 2007.

[202] Viktor K Jirsa, Kelly J Jantzen, Armin Fuchs, and JA Scott Kelso. Neural field dynamics on the folded three-dimensional cortical sheet and its forward eeg and meg. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 286–299. Springer, 2001.

[203] Yi-Ping Lo, Reuben O Dea, Jonathan J Crofts, Cheol E Han, and Marcus Kaiser. A geometric network model of intrinsic grey-matter connectivity of the human brain. *Scientific reports*, 5, 2015.

[204] Fatihcan M Atay and Axel Hutt. Stability and bifurcations in neural fields with finite propagation speed and general connectivity. *SIAM Journal on Applied Mathematics*, 65(2):644–666, 2004.

[205] Romain Veltz and Olivier Faugeras. Local/global analysis of the stationary solutions of some neural field equations. *SIAM Journal on Applied Dynamical Systems*, 9(3):954–998, 2010.

[206] Romain Veltz and Olivier Faugeras. Stability of the stationary solutions of neural field equations with propagation delays. *The Journal of Mathematical Neuroscience*, 1(1):1, 2011.

[207] Pedro M Lima and Evelyn Buckwar. Numerical solution of the neural field equation in the two-dimensional case. *SIAM Journal on Scientific Computing*, 37(6):B962–B979, 2015.

[208] Jian Fang and Grégory Faye. Monotone traveling waves for delayed neural field equations. *Mathematical Models and Methods in Applied Sciences*, 26(10):1919–1954, 2016.

[209] Michael Breakspear. Dynamic models of large-scale brain activity. *Nature neuroscience*, 20(3):340, 2017.

[210] Cornelis J Stam. Nonlinear dynamical analysis of eeg and meg: review of an emerging field. *Clinical neurophysiology*, 116(10):2266–2301, 2005.

[211] M Breakspear, JA Roberts, John R Terry, S Rodrigues, N Mahant, and PA Robinson. A unifying explanation of primary generalized seizures through nonlinear brain modeling and bifurcation analysis. *Cerebral Cortex*, 16(9):1296–1313, 2005.

[212] James Rankin and Frédéric Chavane. Neural field model to reconcile structure with function in primary visual cortex. *PLOS Computational Biology*, 13(10):e1005821, 2017.

[213] Francois Grimbert. Mesoscopic models of cortical structures. *Unpublished doctoral dissertation, University of Nice Sophia-Antipolis*, 2008.

[214] Valentin Markounikau, Christian Igel, Amiram Grinvald, and Dirk Jancke. A dynamic neural field model of mesoscopic cortical activity captured with voltage-sensitive dye imaging. *PLoS computational biology*, 6(9):e1000919, 2010.

[215] Sandrine Chemla and Frédéric Chavane. A biophysical cortical column model to study the multi-component origin of the vsdi signal. *Neuroimage*, 53(2):420–438, 2010.

[216] Valery L Feigin, Mohammad H Forouzanfar, Rita Krishnamurthi, George A Mensah, Myles Connor, Derrick A Bennett, Andrew E Moran, Ralph L Sacco, Laurie Anderson, Thomas Truelsen, et al. Global and regional burden of stroke during 1990–2010: findings from the global burden of disease study 2010. *The Lancet*, 383(9913):245–255, 2014.

[217] S Thomas Carmichael. Brain excitability in stroke: the yin and yang of stroke progression. *Archives of neurology*, 69(2):161–167, 2012.

[218] Nikolai Bessonov, Anne Beuter, Sergei Trofimchuk, and Vitaly Volpert. Cortical waves and post-stroke brain stimulation. *Mathematical Methods in the Applied Sciences*.

[219] Robert Pemberton Burn. *Numbers and functions: steps into analysis*. Cambridge University Press, 2015.

[220] Erwin Kreyszig. *Introductory functional analysis with applications*, volume 81. wiley New York, 1989.

[221] William H Press, William H Press, Brian P Flannery, Brian P Flannery, Saul A Teukolsky, William T Vetterling, and William T Vetterling. *Numerical recipes in Pascal: the art of scientific computing*, volume 1. Cambridge University Press, 1989.

[222] David C Van Essen and Deanna M Barch. The human connectome in health and psychopathology. *World Psychiatry*, 14(2):154, 2015.

[223] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. The minimal preprocessing pipelines for the human connectome project. *Neuroimage*, 80:105–124, 2013.

[224] Bruce Fischl. Freesurfer. *Neuroimage*, 62(2):774–781, 2012.

[225] Qianqian Fang and David A Boas. Tetrahedral mesh generation from volumetric binary and grayscale images. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1142–1145. Ieee, 2009.

Appendix I

Banach spaces

Here we present the definitions of the relevant spaces and their norms for the uniqueness and existence theorems presented in Chapter 2.

Definition A.1. *(Cauchy sequence) A sequence (u_n) is a Cauchy sequence provided that for every $\epsilon > 0$ there exists a natural number n_0 so that for $n, m > n_0$ we have that $|u_n - u_m| < \epsilon$* [219].

Definition A.2. *(Completeness) A vector space X is complete if every Cauchy sequence in X converges (to a point in X)* [220].

Definition A.3. *(Norm) A normed vector space, X, has a norm defined on it, which is defined as a real-valued function on the space,*

$$\|\mathbf{x}\|, \quad \mathbf{x} \in X.$$

It has the following properties:

(i) $\|\mathbf{x}\| \geq 0$,

(ii) $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = 0$,

(iii) $\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|$,

(iv) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$,

where $\mathbf{x}$ and $\mathbf{y}$ are arbitrary vectors in X and α is any scalar.

Definition A.4. *(Banach space) A Banach space is a complete normed vector space* [220].

N.B. The above are defined in terms of vector spaces however in the next section we consider the particular case of function spaces where $f \in X$.

A.1 Some important examples of Banach spaces

EXAMPLE A.1. *$L^p(\mathbb{R}^n)$ The function space with norm*

$$\|f\|_p = \left(\int_{\mathbb{R}^n} |f(\mathbf{x})|^p \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}.$$

EXAMPLE A.2. *$BC(\mathbb{R}^n)$ The Banach space of bounded continuous functions with norm*

$$\|f\|_\infty = \sup_{\mathbf{x}\in\mathbb{R}^n} |f(\mathbf{x})|.$$

In the thesis we also make use of the space,

EXAMPLE A.3. *$BC(\mathbb{R}^n \times [0,\rho])$ with norm*

$$\|f\|_\rho := \sup_{\mathbf{x}\in\mathbb{R}^n, t\in[0,\rho]} |f(\mathbf{x},t)|.$$

Appendix II

Standard methods

Here we present the system of ODEs resulting when implementing FFTs or the trapezoidal rule to the NFMs considered in this thesis: the Amari equation

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = -u(\mathbf{x},t) + A\int_{\Omega} w(\mathbf{x}',\mathbf{x})S(u(\mathbf{x}',t))\mathrm{d}\Omega(\mathbf{x}') \tag{B.1}$$

and the adaptive NFM

$$\begin{aligned}\frac{\partial u(\mathbf{x},t)}{\partial t} &= A\int_{\Omega} w(\mathbf{x}',\mathbf{x})S(u(\mathbf{x}',t))\mathrm{d}\Omega(\mathbf{x}') - u(\mathbf{x},t) - a(\mathbf{x},t)\\ \tau\frac{\partial a(\mathbf{x},t)}{\partial t} &= Bu(\mathbf{x},t) - a(\mathbf{x},t).\end{aligned} \tag{B.2}$$

B.1 FFTs in 2D

Let $f : \mathbb{R}^2 \mapsto \mathbb{C}$. The *fourier transform* in two-dimensions is defined to be

$$\tilde{f}(\mathbf{k}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\mathbf{x})e^{-i\mathbf{k}\cdot\mathbf{x}}\mathrm{d}x\mathrm{d}y,$$

where here $\mathbf{x} = [x, y]^T$ and $\mathbf{k} = [k_x, k_y]^T$. If this intergal exists, then $\tilde{f} = \mathcal{F}[f]$ is called the *Fourier transform* of f. In the case when $\mathcal{F}$ is bijective, the inverse transformation $\mathcal{F}^{-1} : \tilde{f} \mapsto f$ is given by

$$f(\mathbf{x}) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{f}(\mathbf{k})e^{i\mathbf{k}\cdot\mathbf{x}}\mathrm{d}k_x\mathrm{d}k_y.$$

For $f, g \in L(\mathbb{R}^2)$ define their *convolution* as:

$$(f * g)(\mathbf{x}) = \int_{\mathbb{R}^2} f(\mathbf{x} - \mathbf{x}')g(\mathbf{x}')\mathrm{d}\mathbf{x}'$$

where $\mathbf{x}' = [x', y']^T$. Note that this allows us to immediately rewrite the Amari equation in (B.1) as

$$\frac{\partial u}{\partial t} = -u + A(w * (S \circ u)).$$

Another important property of convolution integrals is their relation to the Fourier transform.

THEOREM B.1 (Convolution). *Let $f, g \in L(\mathbb{R}^2)$, then*

$$\mathcal{F}[f * g] = \mathcal{F}[f] \cdot \mathcal{F}[g].$$

Note that this theorem can be directly generalised to $f, g \in L(\mathbb{R}^n)$, for any n.

To solve the NFMs in (B.1) and (B.2) we start by discretising the spatial domain to obtain an 'equivalent' set of ODEs. The resulting two-dimensional grid consists of N^2 grid points and so the Fourier transform $\mathcal{F}$ is replaced by the *discrete Fourier transform* (DFT):

$$\begin{aligned}\tilde{f}(k_{x_i}, k_{y_j}) &= \left(F_N[f]\right)(k_{x_i}, k_{y_j}) \\ &= \Delta x \Delta y \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} e^{-i(k_{x_i}x_i + k_{y_j}y_j)} f(x_i, y_j).\end{aligned}$$

Here grid points are given by (x_i, y_j) and $\Delta x, \Delta y$ denotes the grid spacing in the x and y directions, respectively. The inverse DFT is obtained as follows

$$\begin{aligned}f(x_i, y_j) &= \left(\mathcal{F}_N^{-1}[\tilde{f}]\right)(x_i, y_j) \\ &= \frac{1}{4\pi^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} e^{i(k_{x_i}x_i + k_{y_j}y_j)} \tilde{f}(k_{x_i}, k_{y_j}).\end{aligned}$$

The above allows us to rewrite, and hence solve efficiently, equations (B.1) and (B.2) as follows:

$$\frac{\mathrm{d}u_n}{\mathrm{d}t} = -u_n + A\mathcal{F}_N^{-1}\left[\mathcal{F}_N[w] \cdot \mathcal{F}_N[S]\right]_n, \tag{B.3}$$

$$\begin{aligned}\frac{\mathrm{d}u_n}{\mathrm{d}t} &= A\mathcal{F}_N^{-1}\left[\mathcal{F}_N[w]\cdot\mathcal{F}_N[S]\right]_n - u_n - a_n,\\ \tau\frac{\mathrm{d}a_n}{\mathrm{d}t} &= Bu_n - a_n.\end{aligned} \tag{B.4}$$

The right hand side of both of the above equations can be efficiently computed using the fast Fourier transform algorithm [221].

B.2 The trapezoidal method

We implement the trapezoidal method on regular Cartesian grids of the periodic square and the surface of a torus. Considering the square domain (B.1) becomes,

$$\begin{aligned}\frac{\mathrm{d}u}{\mathrm{d}t} = -u + A\frac{\Delta x\Delta y}{4}\Bigg(& w(-L,-L)S(u(-L,-L,t)) + w(L,-L)S(u(L,-L,t))\\ & + w(-L,L)S(u(-L,L,t)) + w(L,-L)S(u(L,-L,t))\\ & + 2\sum_{i=1}^{m-1} w(x_i,-L)S(u(x_i,-L,t)) + 2\sum_{i=1}^{m-1} w(x_i,L)S(u(x_i,L,t))\\ & + 2\sum_{j=1}^{n-1} w(-L,y_j)S(u(-L,y_j,t)) + 2\sum_{j=1}^{n-1} w(L,y_j)S(u(L,y_j,t))\\ & + 4\sum_{j=1}^{n-1}\left(\sum_{i=1}^{m-1} w(x_i,y_j)S(x_i,y_j,t)\right)\Bigg).\end{aligned} \tag{B.5}$$

Since we are on a periodic domain $\Delta x = \Delta y$, the function evaluated at the corner values is equivalent *i.e.* they are all equal to $w(-L,L)S(u(-L,L,t))$ and the function evaluated along the edges are equivalent *i.e.* $(x,L) = (x,-L)$ and $(-L,y) = (L,y)$ we can rewrite (B.5) as

$$\begin{aligned}\frac{\mathrm{d}u}{\mathrm{d}t} = -u + A\Delta x^2\Bigg(& w(-L,L)S(u(-L,L,t)) + \sum_{i=1}^{m-1} w(x_i,L)S(u(x_i,L,t))\\ & + \sum_{j=1}^{n-1} w(L,y_j)S(u(L,y_j,t)) + \sum_{j=1}^{n-1}\left(\sum_{i=1}^{m-1} w(x_i,y_j)S(x_i,y_j,t)\right)\Bigg).\end{aligned} \tag{B.6}$$

When considering the torus we rewrite the (x,y,z) coordinates as (ϕ,θ) coordinates in order to implement the trapezoidal rule. In this case, following the above,

(B.1) becomes

$$\begin{aligned}\frac{\mathrm{d}u}{\mathrm{d}t} = -u + A\Delta\theta\Delta\phi\Bigg(& w(0,2\pi)S(u(0,2\pi,t))r(R + r\cos(2\pi)) \\ & + \sum_{i=1}^{m-1} w(\phi_i,2\pi)S(u(\phi_i,2\pi,t))r(R + r\cos(2\pi)) \\ & + \sum_{j=1}^{n-1} w(2\pi,\theta_j)S(u(2\pi,\theta_j,t))r(R + r\cos\theta_j) \\ & + \sum_{j=1}^{n-1}\left(\sum_{i=1}^{m-1} w(\phi_i,\theta)S(\phi_i,\theta_j,t)r(R + r\cos\theta_j)\right)\Bigg), \end{aligned} \tag{B.7}$$

and (B.2) becomes

$$\begin{aligned}\frac{\mathrm{d}u_n}{\mathrm{d}t} = & + A\Delta\theta\Delta\phi\Bigg(w(0,2\pi)S(u(0,2\pi,t))r(R + r\cos(2\pi)) \\ & + \sum_{i=1}^{m-1} w(\phi_i,2\pi)S(u(\phi_i,2\pi,t))r(R + r\cos(2\pi)) \\ & + \sum_{j=1}^{n-1} w(2\pi,\theta_j)S(u(2\pi,\theta_j,t))r(R + r\cos\theta_j) \\ & + \sum_{j=1}^{n-1}\left(\sum_{i=1}^{m-1} w(\phi_i,\theta)S(\phi_i,\theta_j,t)r(R + r\cos\theta_j)\right)\Bigg) \\ & - u_n - a_n, \\ \tau\frac{\mathrm{d}a_n}{\mathrm{d}t} = & Bu_n - a_n. \end{aligned} \tag{B.8}$$

where R and r are the major and minor curvatures of the torus respectively.

Appendix III

Additional Results

In this appendix we present results referred to in the thesis but placed here to reduce repetition.

C.1 Chapter 4

In this section we present results obtained when studying Equation (4.1) from Chapter 4.

We start by presenting the bifurcation results found when implementing linear collocation on a general DistMesh triangulation of the periodic square. The triangulation consists of $n_v = 11094$ nodes and $n = 22188$ triangles and we increase the maximum number of Newton iterations to 100 in `nsoli`. We observe near identical results to those found in Chapter 4. Figure C.1 shows the solution branches found when varying h and A and figures C.2 and C.3 show the solutions at the points labelled P_1, P_2 and P_3 for each branch respectively.

Next we present results found when integrating (4.6) for $T = 400$ using the built in MATLAB routine `ode45` on a random triangulation of the torus. The triangulation is generated by perturbing the nodes of the regular triangulation by 20%. Figure C.4 shows solitary bump solutions found from the initial states centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$. When comparing these solutions to those found when implementing trapezoidal we find that the maximal difference is within $1e-07$ as measured by the infinity norm. Note that when comparing solutions found on general triangulations we use the MATLAB function `griddata` to interpolate the solutions onto the regular grid.

The results when considering the bifurcation analysis on a DistMesh triangulation of the torus, consisting of $n_v = 11094$ nodes is shown in figure C.5 and C.8

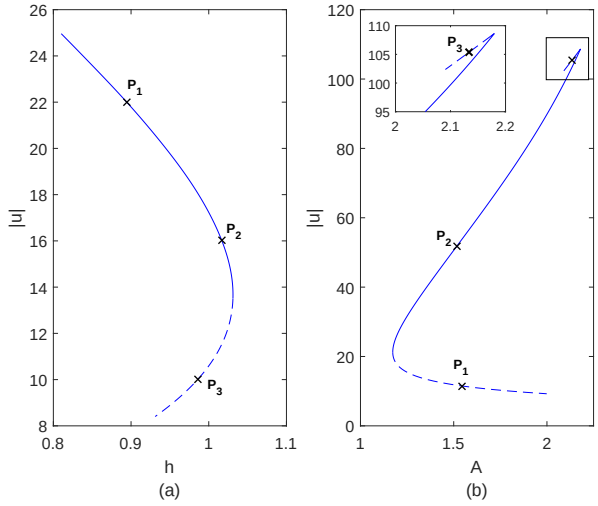


Figure C.1: Solution branches as the parameters; (a) h and (b) A are varied when implementing linear collocation on a general triangulation.

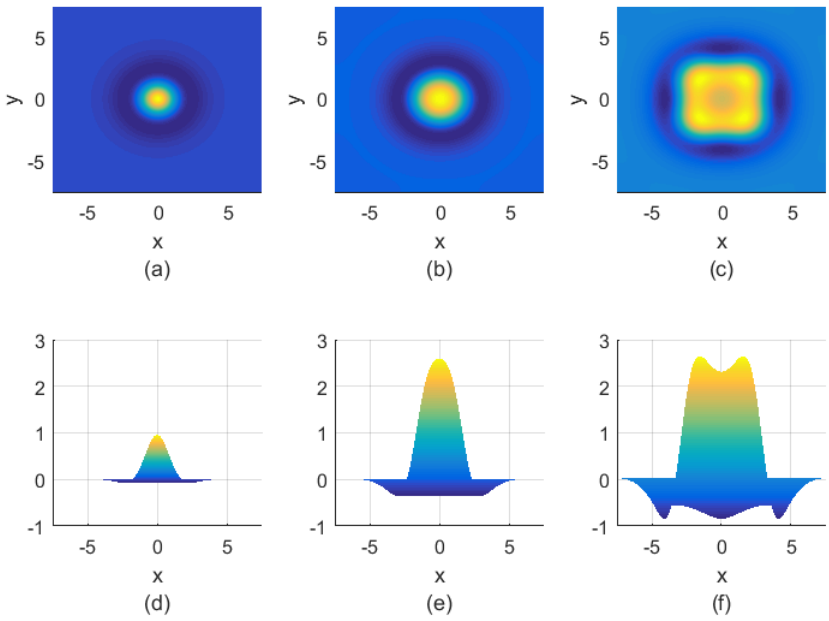


Figure C.2: The three solutions selected along the solution branch in Figure C.1(a); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3.

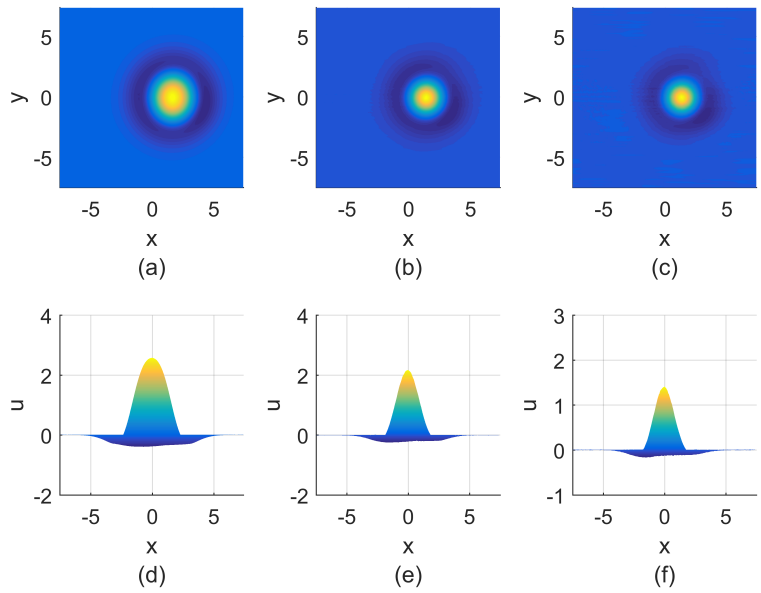


Figure C.3: The three solutions selected along the solution branch in Figure C.1(b); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3.

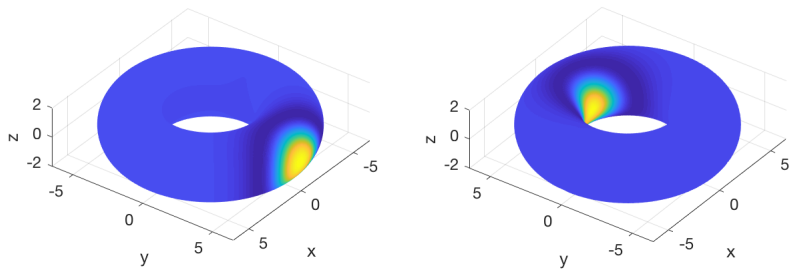


Figure C.4: Solitary bump solutions found on the random triangulation (perturbed by 20%) of the torus. *Left*, from the initial state centred at $\theta = \phi = 0$ and *right* from the initial state centred at $\theta = \phi = \pi$

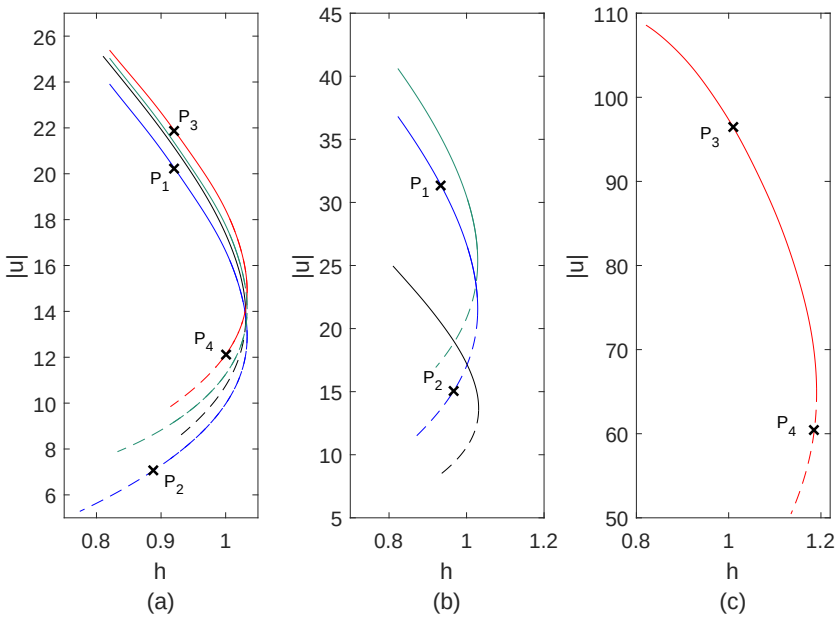


Figure C.5: Solution branches obtained when implementing linear collocation on a general triangulation of a torus when varying h. In all cases; blue - R_1, green - R_2, red -R_3 and black denotes the solution branch obtained on the periodic square. (a) Branches found when considering solutions centred at $\theta = \phi = 0$ (b) branches found when considering solutions centred at $\theta = \phi = \pi$ (c) The branch obtained considering the solution centred at $\theta = \phi = 0$ on the torus with curvature R_3.

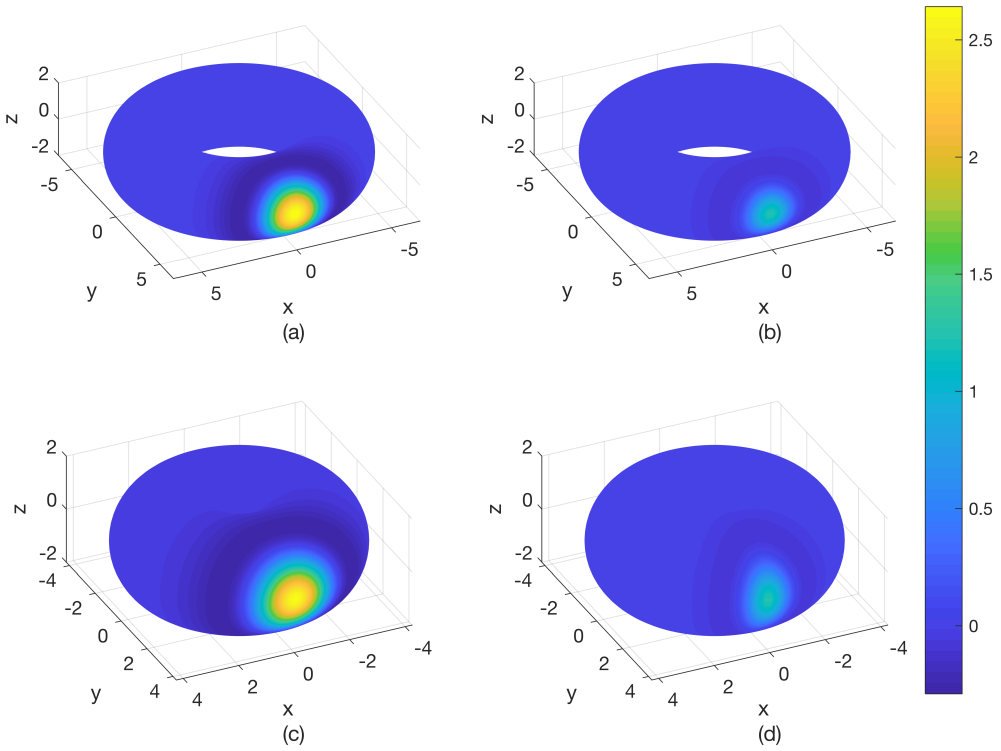


Figure C.6: The four solutions labelled P_1–P_4 along the branch in Figure C.5(a); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches.

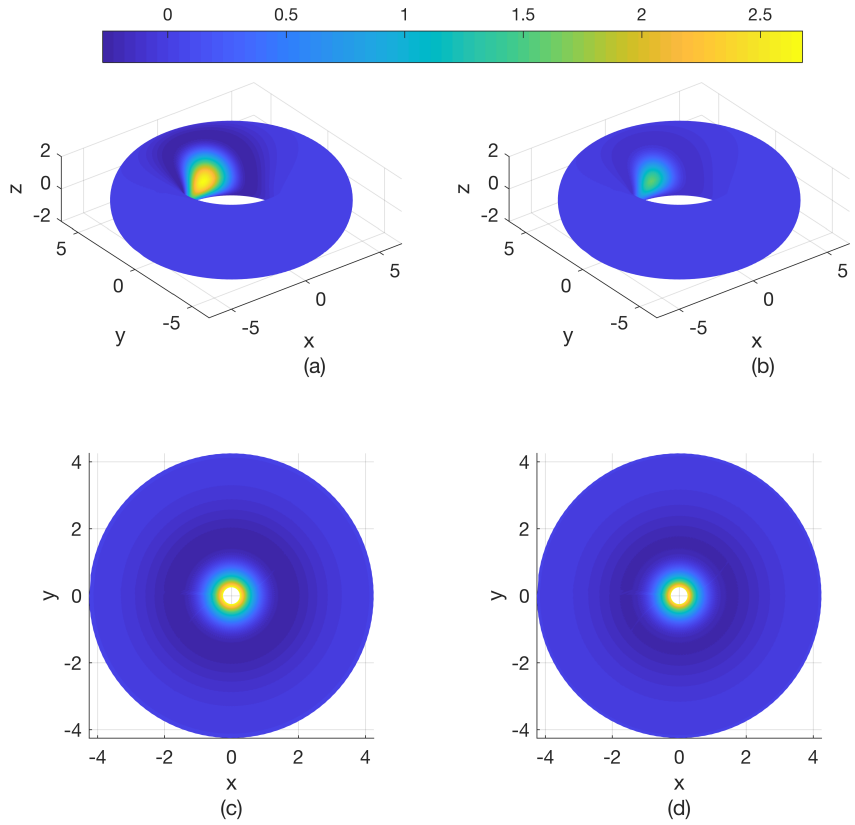


Figure C.7: The four solutions labelled P_1–P_4 along the branch in figure C.5(b) and C.5(c); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches.

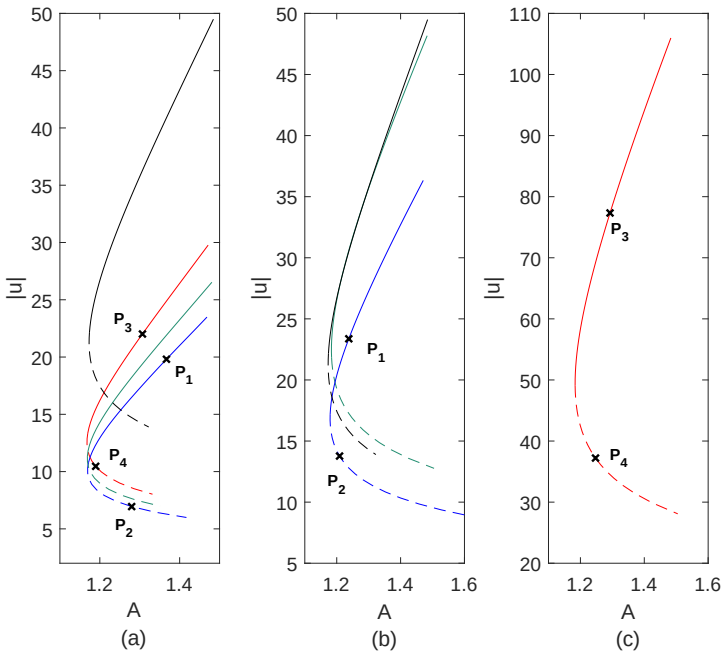


Figure C.8: Solution branches obtained when implementing linear collocation on a general triangulation of a torus when varying A. In all cases; blue - R_1, green - R_2, red -R_3 and black denotes the solution branch obtained on the periodic square. (a) Branches found when considering solutions centred at $\theta = \phi = 0$ (b) branches found when considering solutions centred at $\theta = \phi = \pi$ (c) The branch obtained considering the solution centred at $\theta = \phi = 0$ on the torus with curvature R_3.

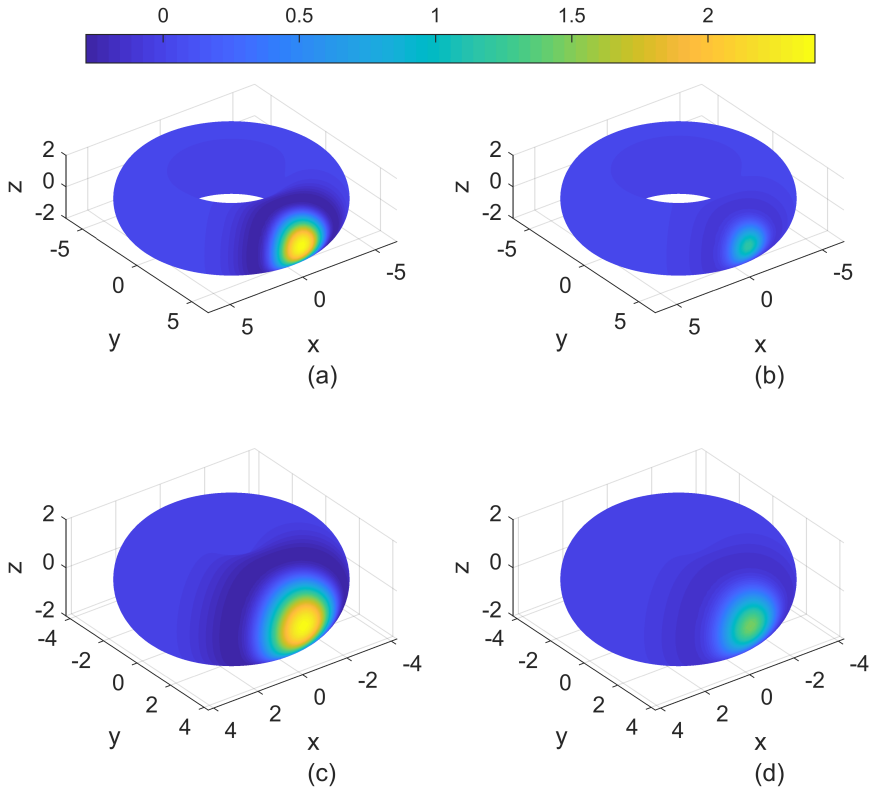


Figure C.9: The four solutions labelled P_1–P_4 along the branch in Figure C.8(a); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches.

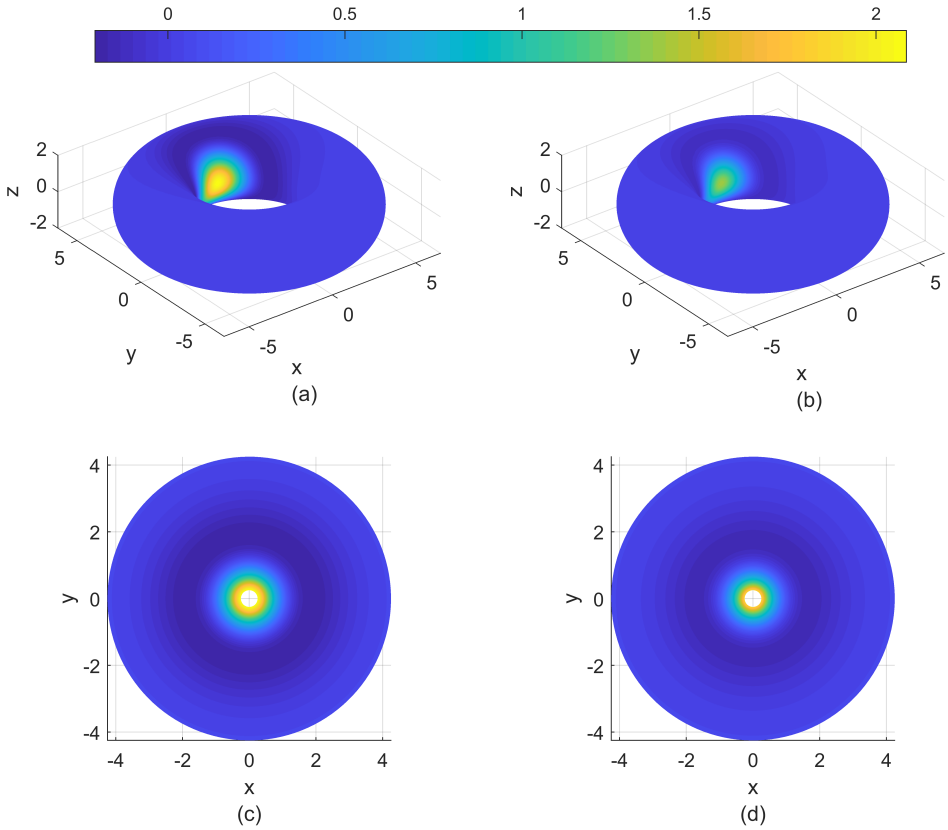


Figure C.10: The four solutions labelled P_1–P_4 along the branch in figure C.8(b) and C.8(c); (a)&(c) stable points along the R_1 and R_3 branches and (c)&(d) unstable points along the R_1 and R_3 branches.

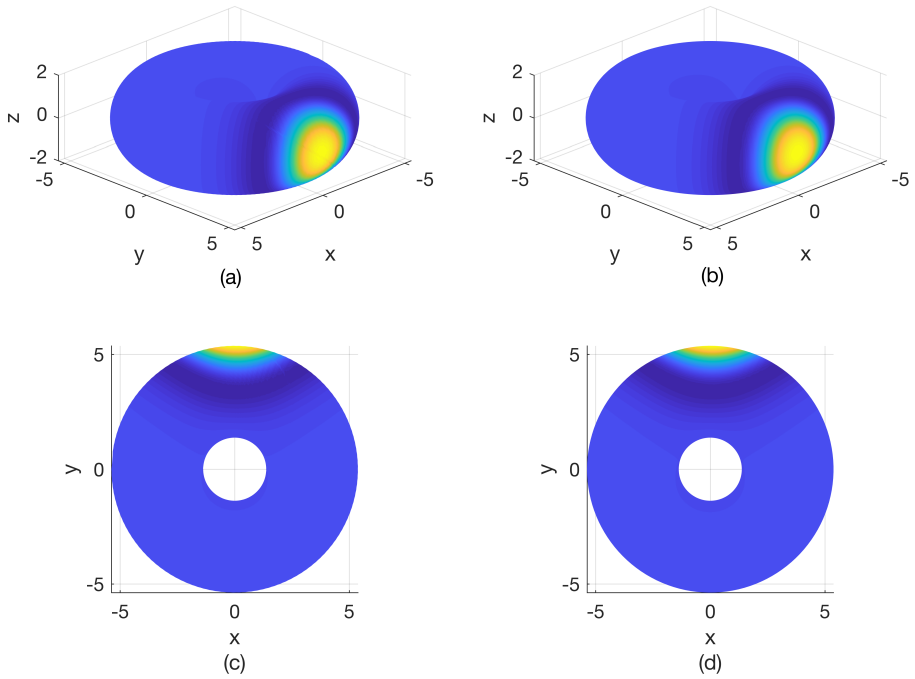


Figure C.11: Solitary bump solutions found, from the initial state centred at $\theta = \phi = 0$, on the torus with curvature R_2, when implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation.

when varying h and A respectively. Note that in all figures solid lines represent stable solutions and dashed lines unstable solutions. Each figure shows results found when considering solutions centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ on the torus for curvatures R_1, denoted by the blue line, R_2, denoted by the green line and R_3 denoted by the red line. Again the maximum number of Newton iterations is set equal to 100. Figures C.6 show the solutions at the points labelled P_1, P_2, P_3 and P_4 along the branches in Figure C.5(a). Figure C.7 show the solutions at the points labelled P_1, P_2, P_3 and P_4 on the branches in Figure C.5(b)&(c). Figures C.9 and C.10 show the solutions at the points labelled P_1, P_2, P_3 and P_4 along the branches in figures C.8(a) and C.8(b)&(c) respectively. As you can see we observe near identical results to those presented in Chapter 4 on the regular domain.

When considering a bifurcation analysis of (4.1) we compare solutions found for three different values of the major curvature radius:

$$R_1 = 4.5, \;\; R_2 = 3R_1/4 = 3.375 \;\; \text{and} \;\; R_3 = R_1/2 = 2.25,$$

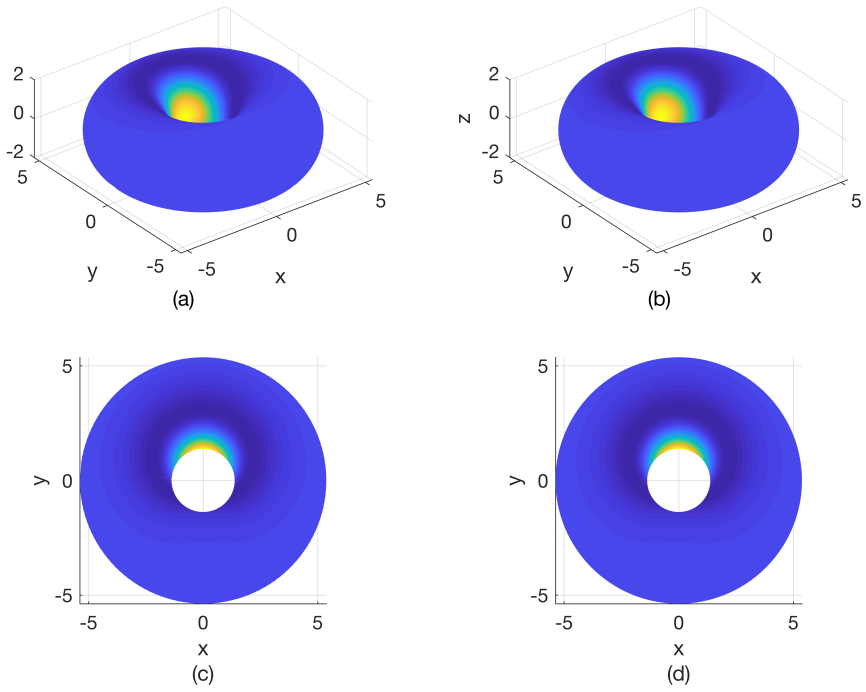


Figure C.12: Solitary bump solutions found, from an initial state centred at $\theta = \phi = \pi$, on the inside if the torus with curvature R_2, when implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation.

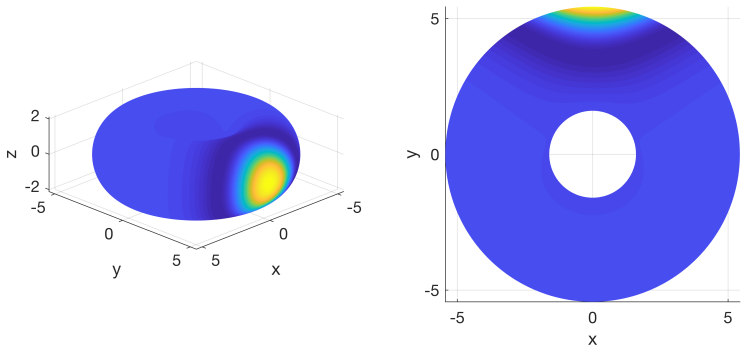


Figure C.13: The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_2, implementing linear collocation on a DistMesh triangulation.

with minor curvature fixed at $r = 2$. Here we solve (4.1) on different triangulations for curvatures R_2 and R_3. We track the evolution of the neural activation u from two different initial conditions: (i) a rectangular area centred on $\theta = \phi = 0$, initially set equal to 2; and (ii) a rectangular area centred on $\theta = \phi = \pi$, again, initially set equal to 2. We compare results found when implementing trapezoidal and linear collocation. The model parameters are set equal to $A = 1.5, h = 0.8$ and $\beta = 5.0$ and we track the evolution of neural activation u as we integrate for $T = 400$ using the built in MATLAB routine `ode45` with absolute ad relative tolerances set to $1e-06$.

C.1.1 R_2

In this section we show results found when considering triangulations of a torus with major curvature R_2. Figures C.11 and C.12 show solitary bump solutions centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively. In both cases, figures on the right hand side show results when implementing linear collocation on a regular grid of $n_v = 8256$ nodes, whilst figures on the left hand side show results when employing trapezoidal on the same grid. The two solutions are in good agreement with the maximal difference of the order $1e-08$.

Next we consider a DistMesh triangulation of the torus. Figures C.13 and C.14

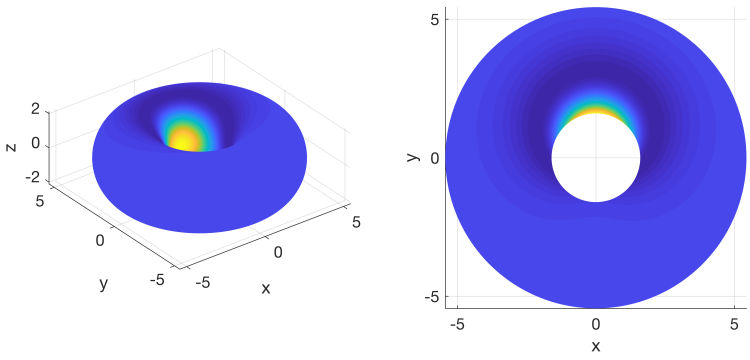


Figure C.14: The solitary bump solution found, from an initial state centred at $\theta = \phi = \pi$, on the inside of the torus with curvature R_2, implementing linear collocation on a DistMesh triangulation.

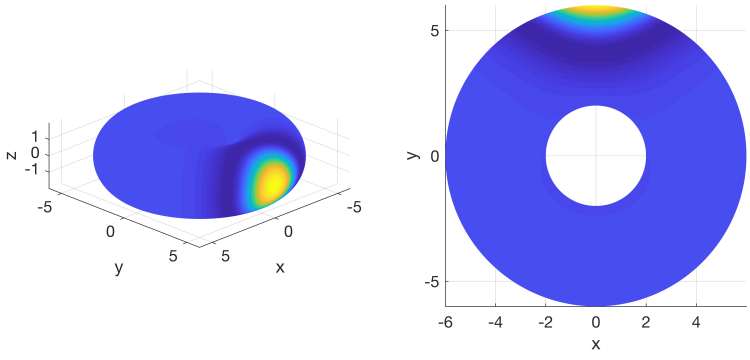


Figure C.15: The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_2, implementing linear collocation on a random triangulation.

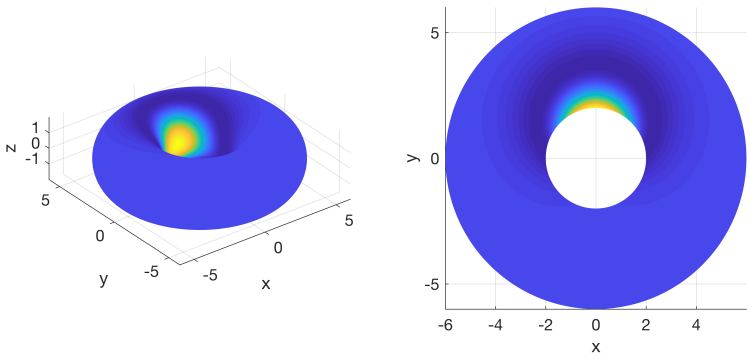


Figure C.16: The solitary bump solution found, from an initial state centred at $\theta = \phi = \pi$, on the inside of the torus with curvature R_2, implementing linear collocation on a random triangulation.

show the solitary bump solutions centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively. We increase the number of nodes to $n_v = 12496$ in order to achieve an accuracy within $1e-07$ for both solutions when comparing them to those found when implementing the trapezoidal method. Note that when comparing solutions from general meshes to those found by the trapezoidal method we use MATLAB's `griddata` to interpolate onto the regular grid. When considering a random triangulation we perturb the points of the regular Cartesian grid based triangulation by 20%. We show solitary bumps centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ in figures C.15 and C.16 respectively. We increase the number of nodes in the triangulation to $n_v = 17766$ in order to achieve a maximal difference of $1e-06$ when comparing the solutions to those found when implementing trapezoidal.

C.1.2 R_3

In this section we show results found when considering triangulations of a torus with major curvature R_3. Figures C.17 and C.18 show the solutions centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively. Again, figures on the right hand side show results when implementing linear collocation on a regular grid of $n_v = 8256$ nodes, whilst figures on the left hand side show results when employing trapezoidal on the same grid.

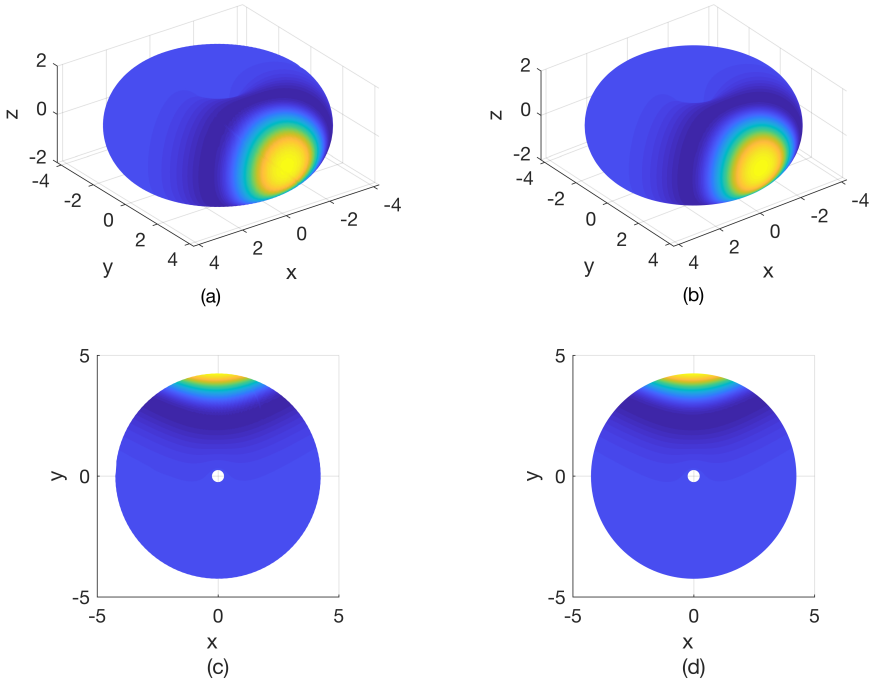


Figure C.17: Solitary bump solutions found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_3, implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation.

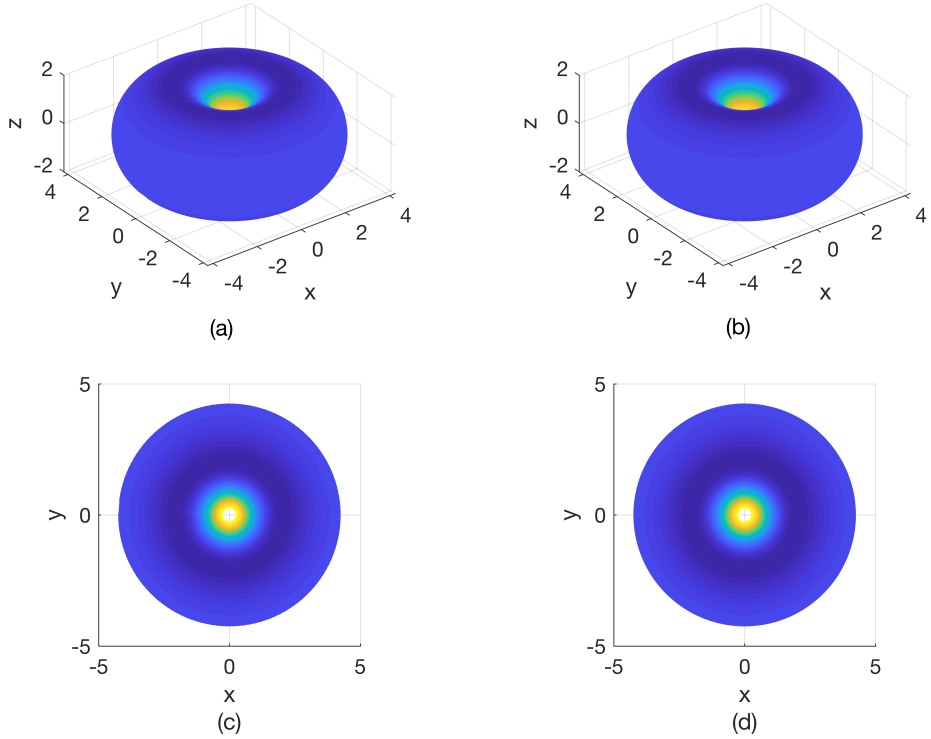


Figure C.18: Solutions found, from an initial state centred at $\theta = \phi = \pi$, on the torus with curvature R_3, implementing; (a)&(c) trapezoidal and (b)&(d) linear collocation.

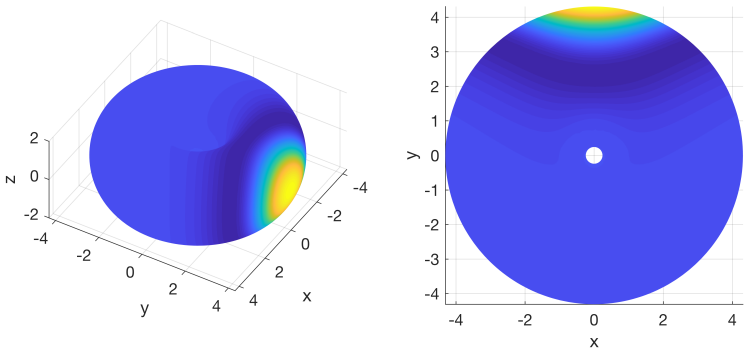


Figure C.19: The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curvature R_3, when implementing linear collocation on a DistMesh triangulation.

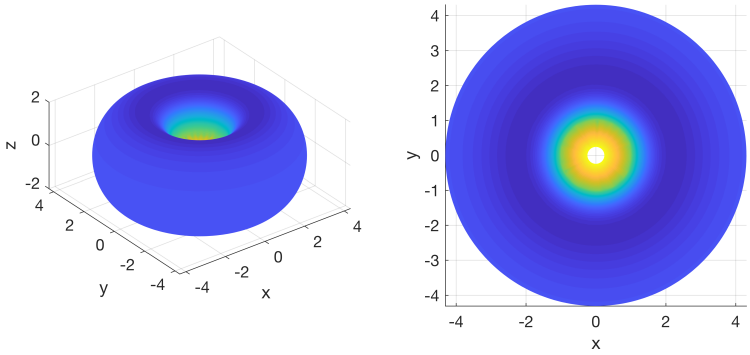


Figure C.20: The solution found, from an initial state centred at $\theta = \phi = \pi$, on the torus with curvature R_3, when implementing linear collocation on a DistMesh triangulation.

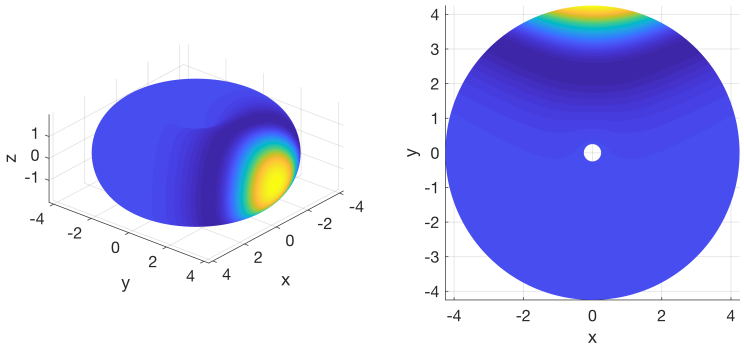


Figure C.21: The solitary bump solution found, from an initial state centred at $\theta = \phi = 0$, on the torus with curature R_3, when implementing linear collocation on a random triangulation.

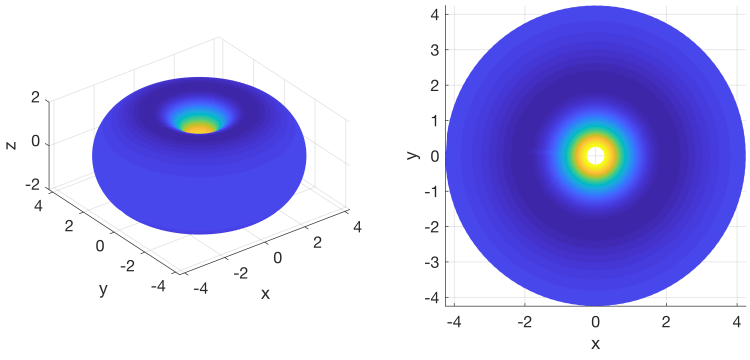


Figure C.22: The solution found, from an initial state centred at $\theta = \phi = \pi$, on the torus with curvature R_3, when implementing linear collocation on a random triangulation.

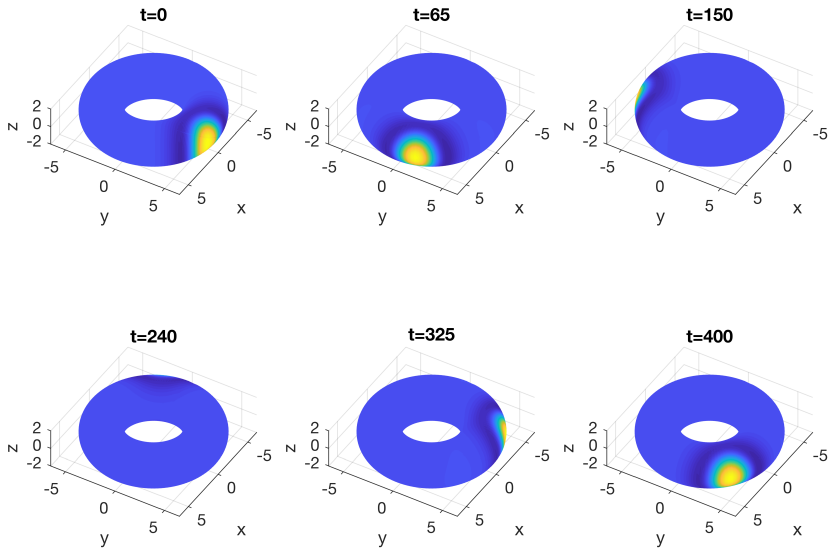


Figure C.23: The travelling bump solution found, from an initial state centred at $\theta = \phi = 0$, on the Cartesian grid based torus when implementing the trapezoidal method.

Here we observe a difference in the type of solution admitted from each initial state. For an initial state centred at $\theta = \phi = \pi$ we no longer observe a bump solution but a ring solution. When comparing solutions we find they are in good agreement with the maximal difference of the order $1e-07$.

When considering a DistMesh triangulation of the torus we must increase the number of nodes in the mesh to $n_v = 12496$ in order to achieve sufficiently accurate solutions. Figures C.19 and C.20 show solutions centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively, again, from the initial state centred at $\theta = \phi = \pi$, we observe a ring solution. When comparing results to those found when implementing trapezoidal we find a maximal difference within $1e-06$ for both solutions. For a random triangulation we perturb the points of the regular grid triangulation by 20%. Figures C.21 and C.22 show the solutions centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively. The number of nodes in the triangulation is increased to $n_v = 17766$ in order to achieve a maximal difference of $1e-06$ for both the bump and ring solution.

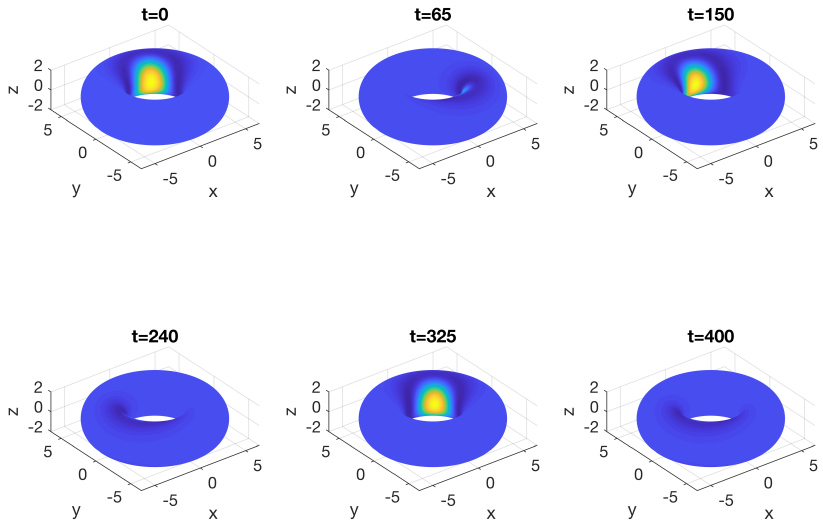


Figure C.24: The travelling bump solution found, from an initial state centred at $\theta = \phi = \pi$, on the Cartesian grid based torus when implementing the trapezoidal method.

C.2 Chapter 5

In this section we present results obtained when solving (5.1) from Chapter 5.

We begin by presenting results found when tracking the evolution of the neural activation u when integrating for $T = 400$ using MATLAB's `ode45`. Figures C.23 and C.24 show the travelling bump solutions found when implementing trapezoidal, from initial states centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively. Figures C.25 and C.26 show the travelling bump solutions found when implementing linear collocation on a random triangulation of the torus, from initial states centred at $\theta = \phi = 0$ and $\theta = \phi = \pi$ respectively. The random domain is generated by perturbing the nodes of the regular domain by 20%. When comparing these solutions to those found when implementing trapezoidal we find the maximal error to be with $1e-07$ for all solutions.

When performing the bifurcation analysis in Chapter 5 we find that in order to implement the derivatives over the triangles we must extend to consider quadratic collocation. Figure C.27 shows the solution branches found when implementing quadratic collocation on the regular triangulation when varying A and h. The solutions at the points labelled P_1, P_2 and P_3 are shown in figure C.28 and C.29

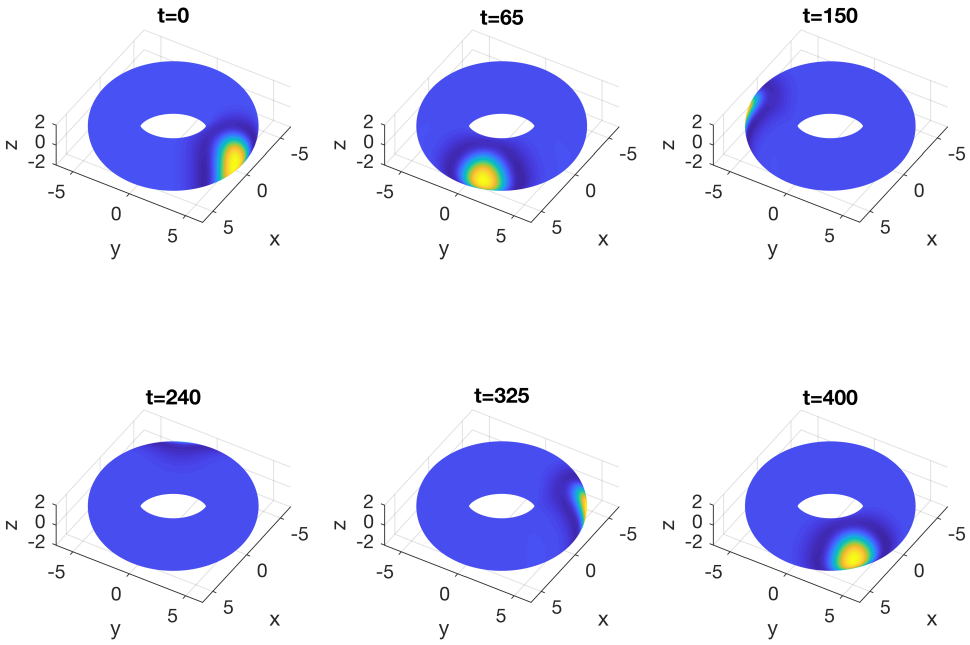


Figure C.25: Travelling bump solution found, from an initial state centred at $\theta = \phi = 0$, on the random triangulation of the torus when implementing linear collocation.

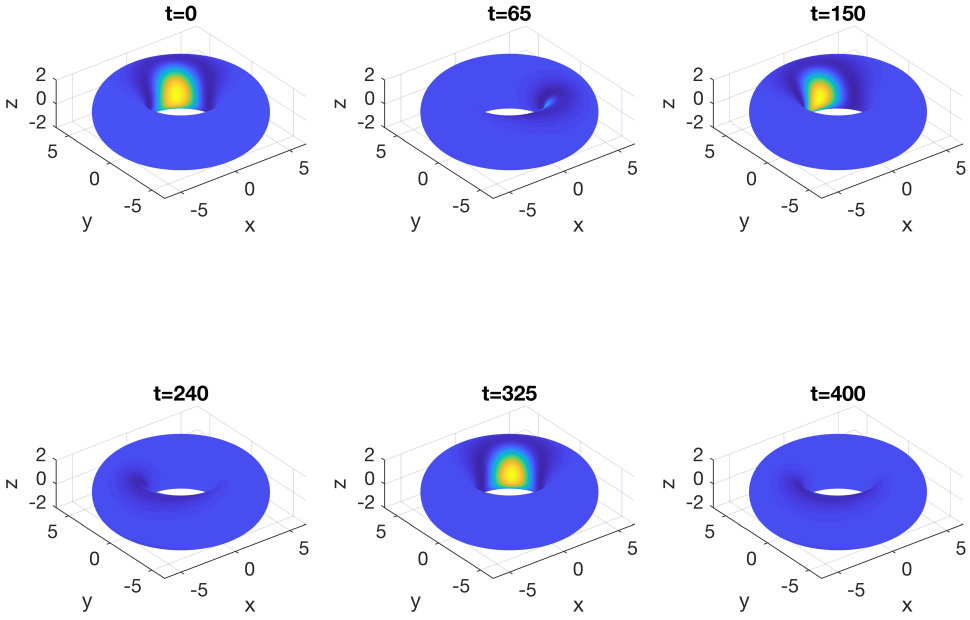


Figure C.26: Travelling bump solution found, from an initial state centred at $\theta = \phi = \pi$ on the random triangulation of the torus when implementing linear collocation.

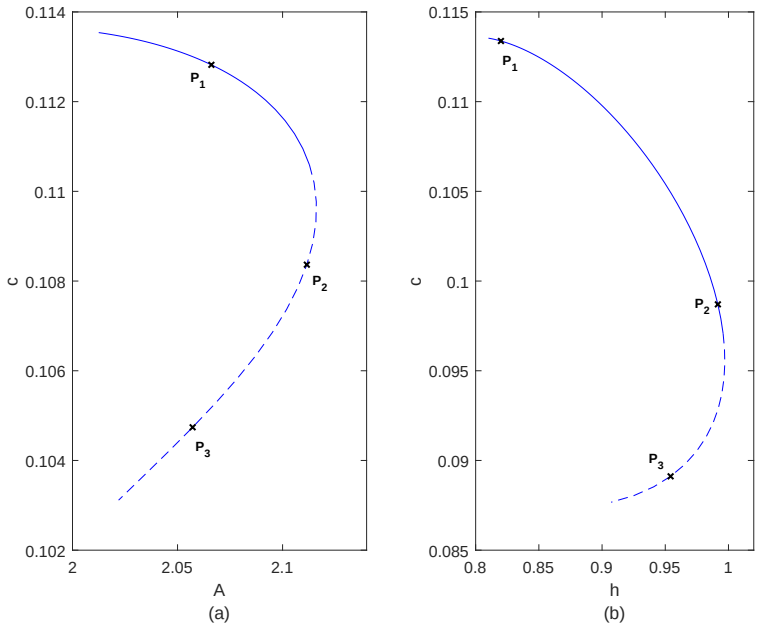


Figure C.27: Solution branches found as the parameters; (a) A and (b) h are varied against c when implementing quadratic collocation.

for the A and h branch respectively. As you can see from these results we obtain near identical results to those found when implementing FFT or linear collocation alongside finite differences for approximating the derivatives.

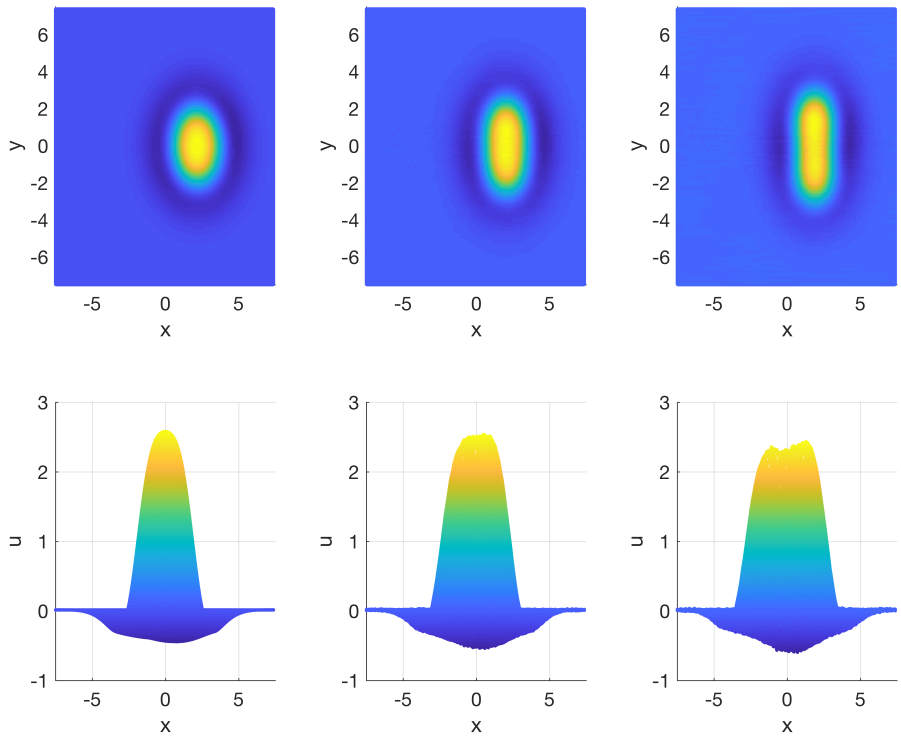


Figure C.28: The three solutions labelled along the branch in Figure C.27(a); (a) & (d) point P_1, (b) & (e) point P_2 and (c) & (f) point P_3.

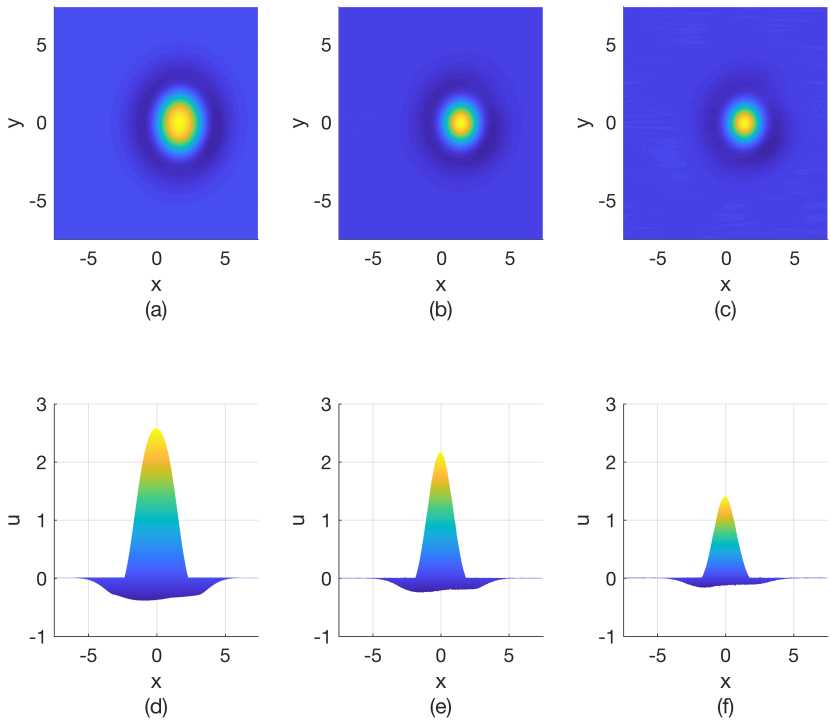


Figure C.29: The three solutions labelled along the branch in Figure C.27(b) ; (a) & (d) point P_1, (b) & (e) point P_3 and (c) & (f) point P_3.

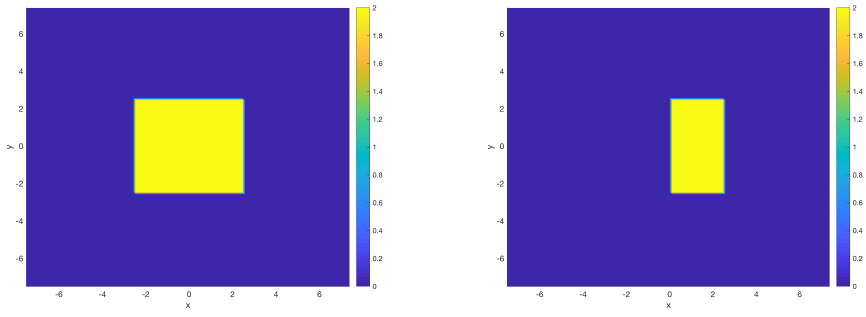
Appendix IV

Quadratic collocation

Here we implement quadratic collocation to solve Equation (4.1) and Equation (5.1). The error analysis for both equations when implementing quadratic collocation is performed in Chapter 4, alongside linear collocation, FFTs and trapezoidal. We found that when considering general triangulations, quadratic collocation outperforms linear collocation.

D.1 Amari equation

Here we show results when solving Equation (4.1) on both regular and irregular triangulations of the periodic square. In all cases the neural activation u was initially set equal to 2 in a rectangular area centred at the origin (see Figure D.1(a)). After



(a) (b)

Figure D.1: (a) Initial condition for u on the periodic square. (b) Initial condition for a on the periodic square.

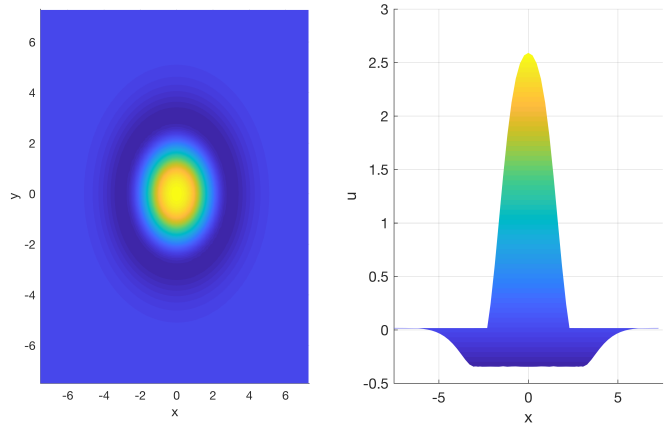


Figure D.2: A solitary bump solution obtained when implementing quadratic collocation on the Cartesian grid based triangulation of the periodic square.

spatial discretisation the resulting ODEs are given by

$$\frac{\mathrm{d}u_n(\mathbf{v}_i,t)}{\mathrm{d}t} = 2A\sum_{k=1}^{n}\mathrm{Area}(\triangle_k)\int_{\sigma} w(\mathbf{v}_i, T_k(r,s))S\left(\sum_{j=1}^{6} u(\mathbf{v}_{k,j},t)l_j(r,s)\right)\mathrm{d}r\mathrm{d}s. \tag{D.1}$$

The system of ODEs given in (D.1) is integrated for $T = 250$ using the built-in MATLAB routine `ode45`, with absolute and relative tolerances set to $1e-06$. The model parameters are set to $A = 1.5, h = 0.8$ and $\beta = 5.0$ and we compare the solutions obtained here to those found when implementing FFTs and trapezoidal. Note that when considering the general triangulations we use MATLAB's `griddata` function to interpolate solutions onto the Cartesian grid.

Figure D.2 shows the solitary bump solution on a regular Cartesian grid based triangulation consisting of $n_v = 16641$ nodes and $n = 8192$ triangles. When comparing this solution to those found when implementing FFTs and trapezoidal we find they are in excellent agreement, within $1e-12$ as measured by the infinity norm. Figure D.3 shows the solitary bump solution found when considering a DistMesh triangulation consisting of $n_v = 16129$ nodes and $n = 7936$ triangles and find that the solution is within $1e-09$ of solutions found by the more standard methods. For the random triangulations we consider again a mesh consisting of $n_v = 16641$ nodes and $n = 8192$ triangles. Figure D.4 shows solitary bump solutions found on random triangulations generated by perturbing the Cartesian triangulation by; left column by 10% and right column by 30%. We find that when comparing the solutions to

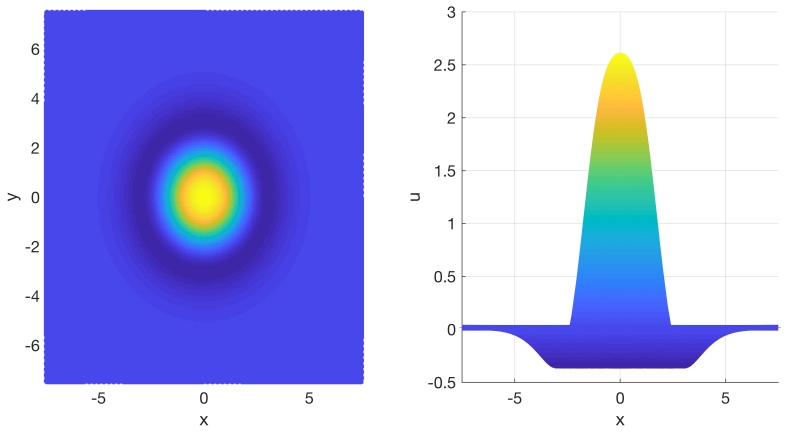


Figure D.3: A solitary bump solution obtained when implementing quadratic collocation on the DistMesh general triangulation of the periodic square.

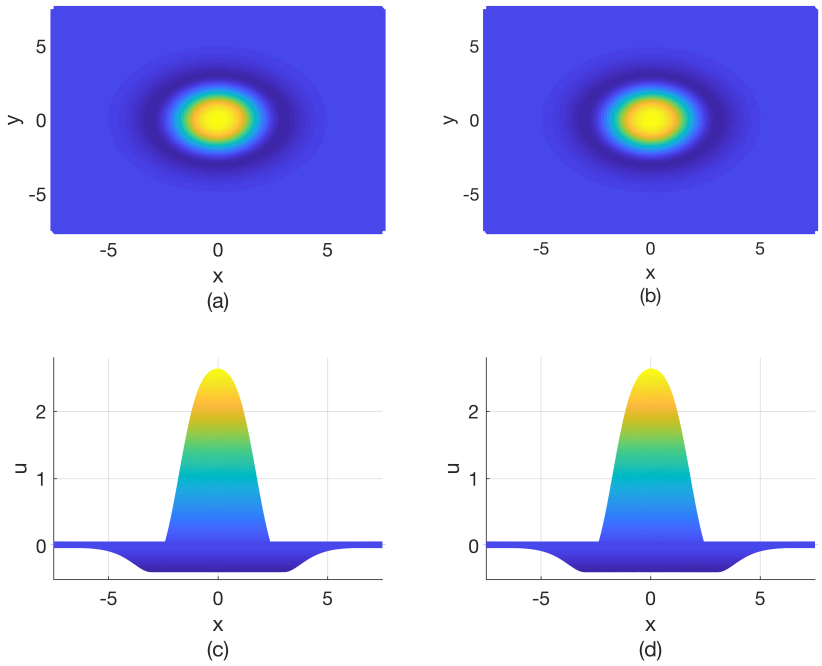


Figure D.4: Solitary bump solutions obtained when implementing quadratic collocation on the random triangulation of the periodic square with perturbation; (a)&(c) 10% and (b)&(d) 30%.

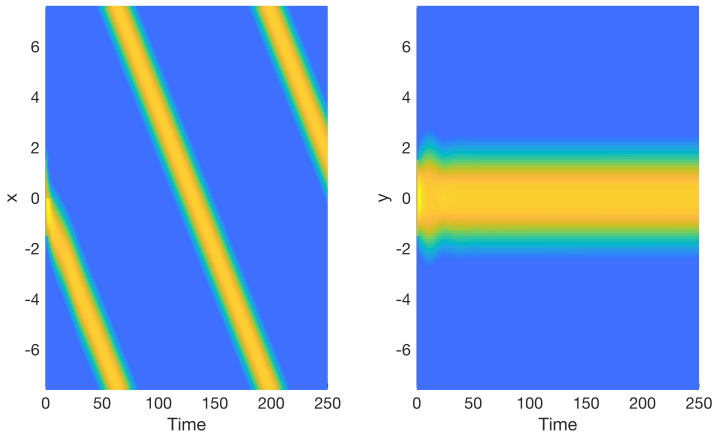


Figure D.5: Travelling bump solution computed on the Cartesian grid based triangulation of the periodic square implementing quadratic collocation.

those found by the more standard methods that they are both within $1e-08$, as measured by the infinity norm.

D.2 Adaptive NFM

Here we show results when solving Equation (5.1) on both regular and irregular triangulations of the periodic square. In all cases the neural activation u was initially set equal to 2 in a rectangular area centred at the origin (see Figure D.1(a)) and the recovery variable a was set equal to 1.5 in a rectangular area shifted to the right of u (see Figure D.1(b)). After spatial discretisation the resulting ODEs are given by

$$\begin{aligned}\frac{\mathrm{d}u_n(\mathbf{v}_i,t)}{\mathrm{d}t} &= 2A\sum_{k=1}^{n}\mathrm{Area}(\triangle_k)\int_{\sigma} w(\mathbf{v}_i,T_k(r,s))S\left(\sum_{j=1}^{6}u(\mathbf{v}_{k,j},t)l_j(r,s)\right)\mathrm{d}r\mathrm{d}s \\ &\quad - u_n(\mathbf{v}_i,t) - a_n(\mathbf{v}_i,t), \\ \tau\frac{\mathrm{d}a_n(\mathbf{v}_i,t)}{\mathrm{d}t} &= Bu_n(\mathbf{v}_i,t) - a_n(\mathbf{v}_i,t).\end{aligned} \tag{D.2}$$

The model parameters were set equal to $A = 2.0, h = 0.8, B = 0.4, \tau = 3.0$ and $\beta = 5.0$. We integrated (D.2) for $T = 250$ using the built-in MATLAB routine `ode45` with absolute and relative tolerances set to $1e-06$.

Figure D.5 shows the travelling bump solution found on a regular Cartesian grid based triangulation consisting of $n_v = 16641$ nodes and $n = 8192$ triangles. We

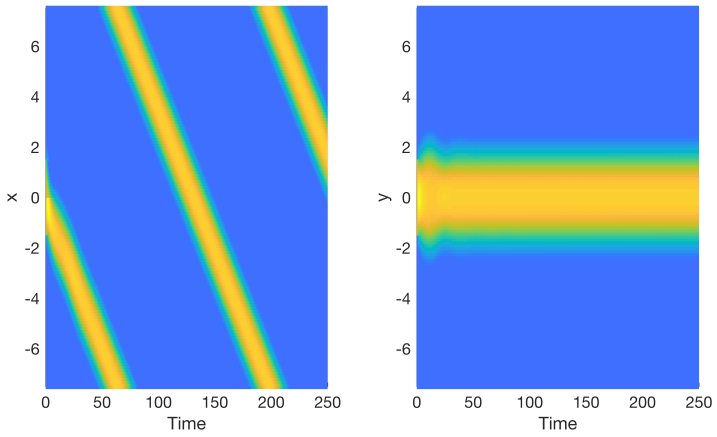


Figure D.6: Travelling bump solutions computed when implementing quadratic collocation on the DistMesh triangulation of the periodic square.

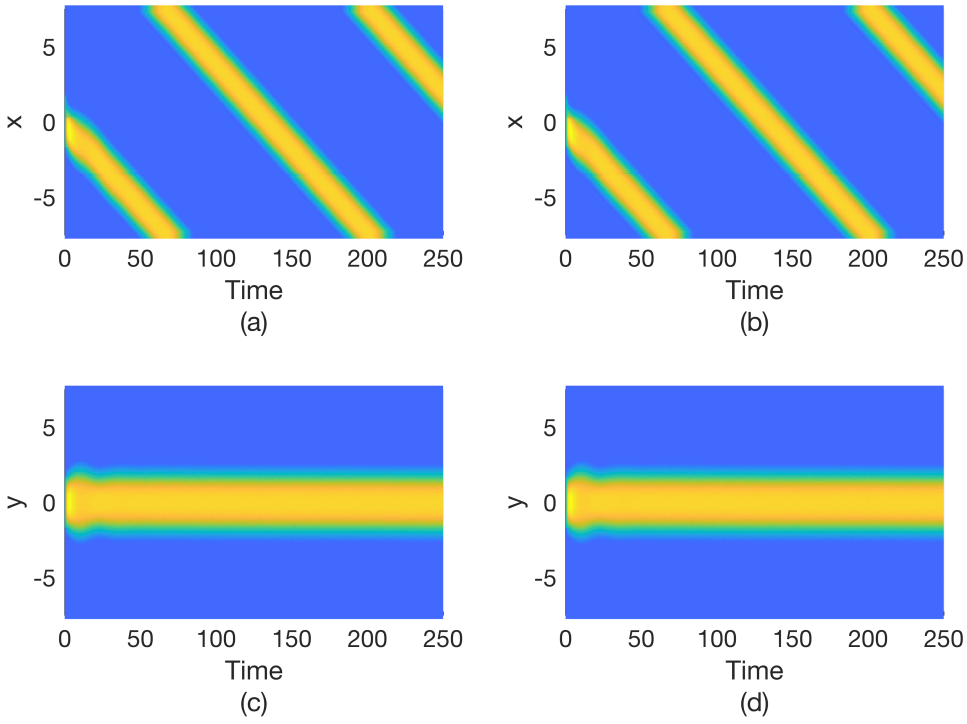


Figure D.7: Travelling bump solutions found when implementing quadratic collocation on the random triangulation of the periodic square, perturbed by; (a)&(c) 10% and (b)&(d) 30%.

find that this solution is within $1e-12$ of the solutions found by the more standard methods. Figure D.6 shows the travelling bump solution found when considering a DistMesh triangulation consisting of $n_v = 16129$ nodes and $n = 7936$ triangles. We find that the solution is in good agreement, within $1e-09$ of solutions found by the more standard methods. For the random triangulations we consider again a mesh consisting of $n_v = 16641$ nodes and $n = 8192$ triangles. Figure D.7 shows travelling bump solutions found on random triangulations generated by perturbing the Cartesian triangulation by; left column by 10% and right column by 30%. We find that when comparing the solutions to those found by the more standard methods that they are both within $1e-07$, as measured by the infinity norm.

D.3 Summary

Here we have employed quadratic collocation to solve the Amari equation and an adaptive neural field model on different triangulations of a periodic square. We found that quadratic collocation is capable of replicating travelling bump solutions found by more standard methods. Importantly we found that for the DistMesh domain we can consider a triangulation with fewer triangles than that considered when implementing linear collocation and we can consider higher perturbations in the random domain and still obtain results within $1e-07$ of the solutions found by more standard methods. The advantage of implementing higher order methods is two fold; firstly as we have found from our experiments in Chapter 5 the extension to higher order methods is crucial for the accurate computation of solution branches found when conducting a bifurcation analysis. Secondly, we can consider fewer triangles and higher perturbations in the random domain and still achieve sufficient accuracy, reducing the computational cost and whilst increasing the complexity of the domain.

Appendix V

Human Brain

In the following we present preliminary results when considering the methods described in this thesis for solving a neural field model on a representation of the human cortical surface. In particular, we solve (5.1) on a triangulated mesh representation of the left hemisphere of the human brain, which was obtained via the Human Connectome Project (HCP) [21]. More specifically, the data deployed in this section is taken from [223], which was downloaded from the HCP and preprocessed using the Freesurfer software [224] (see [223] for further details).

The resulting cortical representation consists of some $148,396$ vertices and $296,788$ triangles and so requires us to downsample the mesh for computational purposes. We used the `iso2mesh` MATLAB toolbox to process the mesh [225]. In particular, we used the function

```
% meshresample function
[P,T]=meshresample(v,f,ratio)
```

where `v` and `f` are the vertices and triangles from the original mesh and `ratio` is the fraction of the original mesh retained after sampling. For illustrative purposes we set `ratio` $= 0.04$ in our work, which results in a mesh with 5121 vertices, `P`, and 10238 triangles, `T` (Figure E.1 shows a surface plot of the down sampled brain). Note that we are also currently performing simulations on larger meshes in order to determine the effect of mesh size on the observed solutions.

We solved (5.1) on the cortical representation shown in Figure E.1, tracking the evolution of neural activity u, which was initially set equal to 2 in a small region surrounding a node selected at random, with the adaptation variable a set equal to

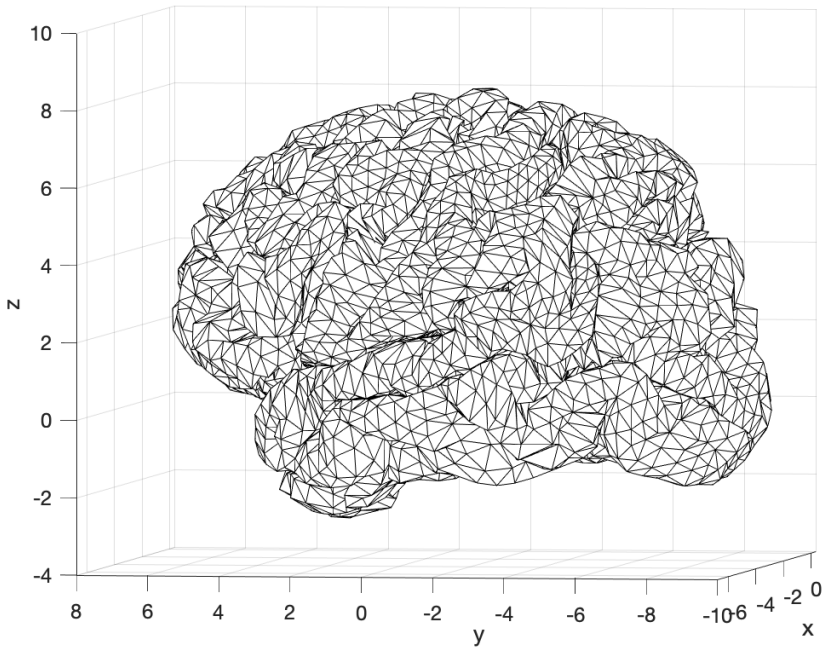


Figure E.1: Triangulation of the down sampled left hemisphere of the human cortical surface.

1.5 in an equivalently sized, partially overlapping region of nodes. We have repeated this experiment a number of times, varying both the initial position of u and a, in order to study the influence of geometry on solutions obtained by Equation (5.1). The ODEs in (5.2) were solved for $T = 400$ using MATLAB's built in `ode45` routine with absolute and relative tolerances equal to $1e-06$, and model parameters set equal to the same values as before, *i.e.*

$$A = 2.0,\ \ h = 0.8,\ \ B = 0.4,\ \ \tau = 3.0 \ \text{ and } \ \beta = 5.0.$$

The firing rate and connectivity functions were chosen the same as in Chapter 5 and the distances computed numerically using the MMP algorithm, as before.

Figures E.2 and E.3 show the progression of a travelling bump solution of (5.1) on our cortical representation. In the example shown, the bump solution travels into and along the closest sulcus until it reaches a sharp fold, at which point it splits into two travelling bumps moving in opposite directions but still remaining in the valley of the sulcus. The inability of the bump solution to leave the sulcus is likely due to the negative curvature found at these regions, reminiscent of our earlier findings. Importantly, the splitting behaviour observed in this experiment is not seen on the

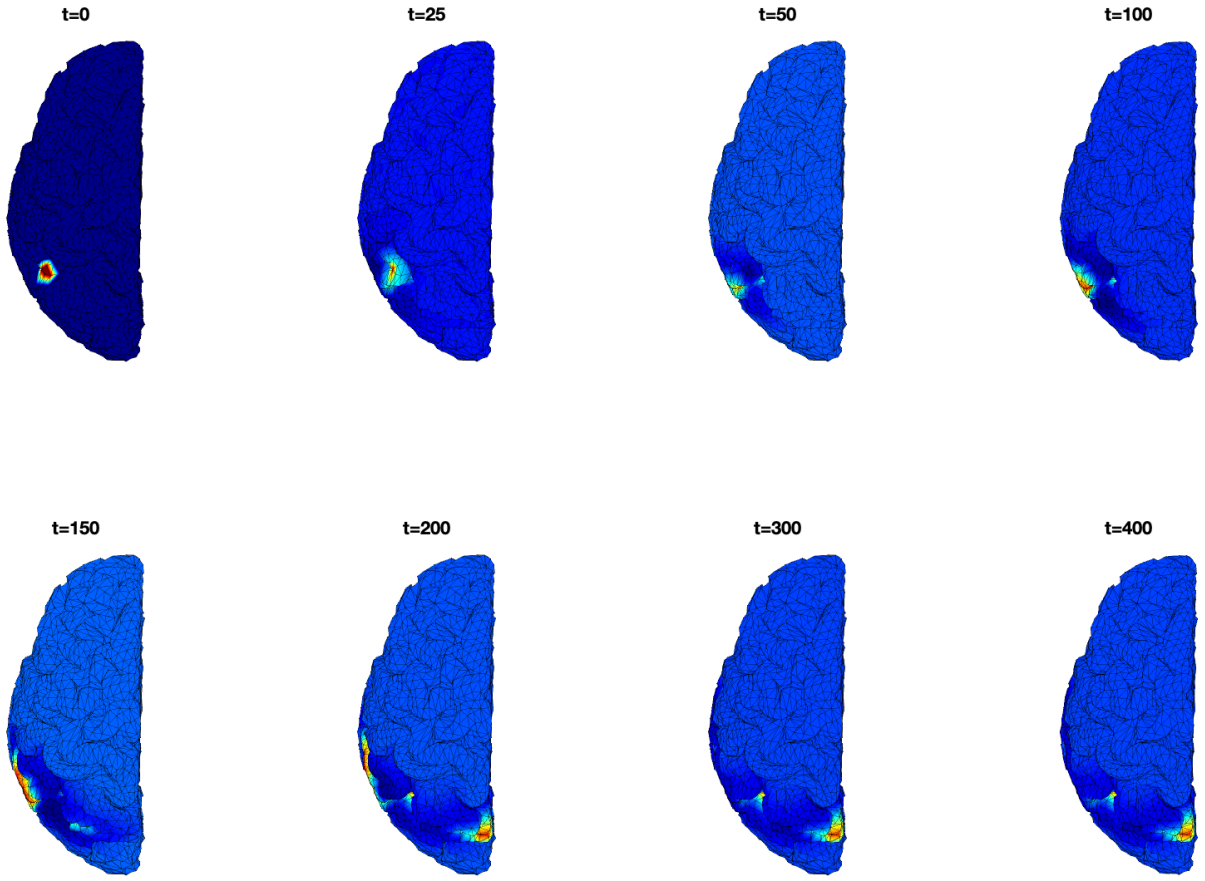
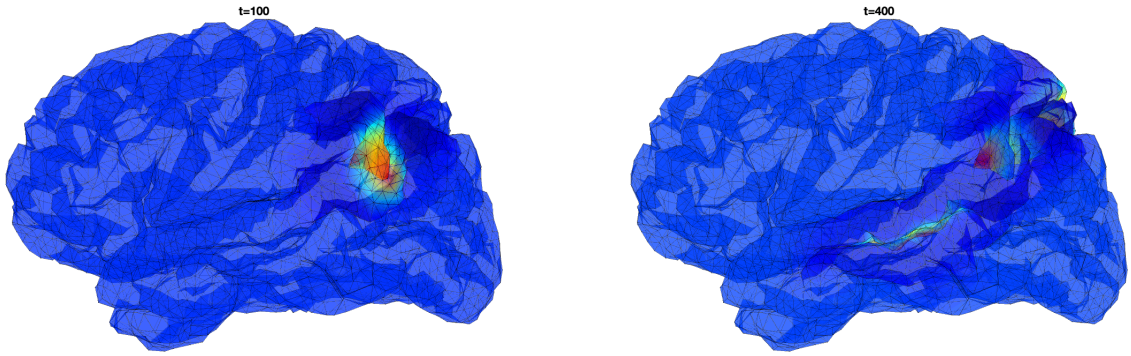


Figure E.2: Travelling bump solution found when solving (5.1) on the triangulated surface of a human brain.



(a) (b)

Figure E.3: (a) The bump solution shown in figures 6.1 and E.2 at time, $t = 100$. (b) The bump solution shown in figures 6.1 and E.2 at time, $t = 400$.

curved geometries considered thus far and is likely a result of the highly convoluted nature of the human cortex. However, the precise reasoning behind these findings will require further research.

Note that in addition to the splitting solutions shown in figures E.2 and E.3 we have also observed solutions (not shown) that do not split but rather get trapped in regions of what would appear to be large negative Gaussian curvature, much like those observed on the rat brain. Conditions under which solutions remain trapped or display splitting behaviour is an area of imminent future research.