Moving mesh methods for problems with layer phenomena

A thesis submitted

by

Róisín Hill

to

The School of Mathematical and Statistical Sciences,

National University of Ireland Galway

in fulfilment of the requirements of the degree of

Doctor of Philosophy

May 2022

Thesis Supervisor: Niall Madden

Head of School: Aisling McCluskey

Contents

List of Tables

List of Figures

Declaration

This thesis is presented in fulfilment of the requirements for the degree of Doctor of Philosophy. It is entirely my own work and has not been submitted to any other university or institute of higher education, or for any other academic award in this university. Where use has been made of the work of other people it has been fully acknowledged and referenced.

. . . . . .
Róisín Hill
May 2022

Acknowledgement

I would first like to express my extreme gratitude to my supervisor, Niall Madden. His dedication, insights, encouragement, generosity, and patience have been invaluable. I truly felt supported, both academically and otherwise, throughout my studies. My passion remained ignited throughout even the most frustrating times of the journey, in large part due to his. I could not have imagined having a better supervisor for my PhD.

Attending the BIRS Workshop on Adaptive Numerical Methods for Partial Differential Equations and Applications in Banff, Canada in May 2018, greatly enhanced the direction of this project. A very special thank you to the organisers Weizhang Huang, Chris Budd and Ronald Haynes for their kind invitation.

Thank you to the staff of the School of Maths for their support and help during my PhD. They have created an environment that is not only conducive to research but also warm and welcoming. In particular, I would like to sincerely thank the members of my GRC, Ray Ryan, Martin Meere, Petri Piiroinen, Bharat Tripathi and Michael Hartnett (Civil Engineering), for their helpful discussions and guidance along the way. Thank you to the administrative staff, in particular Mary, for providing help whenever needed.

I will forever be grateful to my fellow PhD students for welcoming me. From an academic standpoint, the postgraduate seminar groups provided countless benefits. However, more importantly, was the friendship I received. Friday lunches, before COVID-19, were always interesting, no matter what the subject was.

I also offer my gratitude to the mathematical community that have shown their encouragement and given helpful suggestions. In particular, the FEniCS community, those involved in solving singularly perturbed differential equations and those involved in using moving mesh partial differential equations. Again, the welcome I received at the conferences and workshops I did manage to attend is unforgettable.

Finally, I gratefully acknowledge the Irish Research Council. Without their Government of Ireland Postgraduate Scholarship GOIPG/2017/463, this research would not have been possible.

Dedication

For Henry, Eimear and Aoife for your love, encouragement, and support throughout the process. From being stand-in chefs to proofreaders to rubber ducks and beyond, you each made it much more achievable. “Thank you” doesn’t begin to cover it.

Abstract

This thesis is concerned with developing, implementing, testing and refining novel algorithms for generating layer-adapted meshes for the numerical solution of singularly perturbed differential equations (SPDEs). We consider a variety of problems: ordinary and partial differential equations, time-dependent and stationary, scalar and coupled, reaction-diffusion and convection-diffusion. We also construct meshes of different types: *a priori* fitted and *a posteriori* fitted. However, there is a common thread running through the thesis: the algorithms we propose are all based around the concept of *mesh partial differential equations* (MPDEs).

We concentrate on differential equations whose solutions contain boundary or interior layers; these are particularly sensitive and require concentrated meshes to accurately capture these features. Moving meshes partial differential equations [Huang and Russell, 2011] generate meshes that evolve in time for time-dependent problems. We extend this idea to generate meshes using MPDEs for both one- and two-dimensional stationery singularly perturbed differential equations whose solutions exhibit boundary layers. The MPDE is nonlinear, and the efficiency with which it can be solved depends adversely on the magnitude of the perturbation parameter and the number of mesh intervals. We resolve this by proposing a new nonlinear solver based on h-refinement.

The main advantage of the MPDE method for generating layer-adapted grids is that, unlike conventional schemes, they are not restricted to tensor-product grids. Therefore, it is feasible to solve problems whose solutions have layers that vary in width spatially or that are posed on irregularly shaped domains. Specifically, this new approach generalises the concept of a *Bakhvalov* mesh. In that setting, *a priori* asymptotic information on the SPDE's solution, and its derivatives, is used. We further extend the framework to using *a posteriori* estimates for those derivatives, leading to more truly adaptive meshes.

All algorithms presented have been coded in FEniCS [Alnæs et al., 2015], an open-source software system for generating finite element solutions. Key aspects of the code are included, and published on `https://osf.io/dpexh/`, to demonstrate the reproducibility of our results, and to make them accessible to other researchers.

Chapter 1

Introduction

1.1 Motivation

Differential equations model how phenomena change over time or space, such as how a cancerous tumour grows, how a pollutant disperses, or how the value of a commodity changes, for example, see [Mattheij et al., 2005, Chap. 7]. Therefore, their solution is of fundamental importance to science, engineering and finance. In most cases, solutions to partial differential equations (PDEs) cannot be expressed as closed-form expressions. Therefore, we want to generate reliable approximate solutions to these problems using numerical schemes.

We use the standard, linear Galerkin, finite element method, $\mathcal{P}_1$-FEM, to generate numerical solutions to PDEs. We implement the methods in FEniCS [Alnæs et al., 2015], a Python-based open-source software system for the automated solution of PDEs. FEniCS allows access to low-level features while automating other tasks.

Our interest lies in singularly perturbed differential equations (SPDEs). These equations appear in models of semiconductor devices, ion transport across biological membranes, pollution dispersion, financial modelling, population dynamics, and many other applications. For some examples and the SPDEs that appear in their models, see [Morton, 1996, Chap. 1]. The equations are singularly perturbed in that the highest order term is multiplied by a positive perturbation parameter, usually denoted ε. When $\varepsilon \ll 1$, their solutions change very rapidly over very small regions of the domain (layer regions). Our goal is to resolve these regions without excessive computational cost, something that cannot be achieved using classical techniques [Roos et al., 2008, I.2.2].

Specifically, in this thesis, we design and implement algorithms to generate layer-adapted meshes that have a high portion of their mesh points concentrated in the layer regions, and a few points elsewhere.

There are two main classes of layer-adapted meshes:

- *a priori* fitted meshes, which are generated before the problem is solved and usually use specific information about the solution of the problem; for example, the location and width of the solution's layers. The most celebrated of these meshes are the Bakhvalov mesh, introduced in [Bakhvalov, 1969], and the Shishkin mesh of [Shishkin, 1990].
- *a posteriori* fitted meshes, which are generated using information gained by solving the problem on the initial mesh and refining that mesh.

In this thesis, we propose and investigate a collection of algorithms for generating both *a priori* and *a posteriori* fitted meshes, all based on the idea of mesh movement, based on the moving mesh PDE (MMPDE) method of [Huang et al., 1994b].

1.2 Thesis outline and organisation

This thesis is concerned with efficiently computing reliable solutions to SPDEs by generating suitable meshes for these problems using a moving mesh method.

In Chapter 2, we review the key concepts and results from the literature that are relevant to this thesis. In Section 2.2, we consider SPDEs. The solutions to SPDEs contain layers whose widths depend on the magnitude of ε. The solution changes more rapidly when ε is smaller, resulting in a narrower layer region that standard numerical methods cannot resolve. In Section 2.3, we briefly detail the FEM method that we use and its implementation in FEniCS. In Section 2.4, we review the classical fitted meshes of Shishkin and Bakhvalov, and more recent efforts towards *a posteriori* adapted meshes. These meshes condense in layer regions. We give particular attention to Bakhvalov meshes, which can be understood as solutions to certain equidistribution problems [Linß, 2010]. Section 2.5 is concerned with how to evaluate the quality of computed solutions, and includes a short investigation of numerical estimates of errors. In Section 2.6, we outline the relevant details of mesh adaptivity, including a brief investigation of the h-refinement method for SPDEs. Finally, in Section 2.7, we briefly outline the relevant history of the MMPDE method.

Chapter 3 is concerned with our implementation of a time-dependant convection-diffusion problem and Burgers' equation, both with interior layers that propagate through the domain and evolve with time. Burgers' equation is the focus of [Huang and Russell, 2011, Chap. 2], thus allowing us to present our approach to the MMPDE method in the context of a standard reference. The MMPDE is posed in terms of the errors in the numerical solution.

Chapter 4 introduces novel methods for generating Bakhvalov-style meshes using mesh PDEs (MPDEs) for one-dimensional SPDEs. Sections 4.2 to 4.4 have been published as part of [Hill and Madden, 2021], i.e.,

> Róisín Hill and Niall Madden. Generating layer-adapted meshes using mesh partial differential equations. Numer. Math. Theor. Meth. Appl., 2021, https://doi.org/10.4208/nmtma.OA-2020-0187.

We use *a priori* information about the physical problem to generate layer-adapted meshes using an MPDE. We optimise the algorithm by combining the MPDE method with uniform h-refinement. Solutions generated on these meshes are directly comparable to those generated on classic Bakhvalov meshes. The resulting meshes are similar to other layer-adapted meshes generated using the equidistribution principle. However, they can be extended to two-dimensional problems which are discussed later in the thesis. In Section 4.5 we present further investigations into the algorithms and resulting meshes of Section 4.2, but which could not be accommodated in [Hill and Madden, 2021]. Specifically, we show:

- a Bakhvalov mesh is a solution to the MPDE,
- the discretized MPDE equidistributes the mesh density function within an acceptable tolerance criterion, and
- the meshes generated are "optimal", by which we mean with respect to the order of convergence, for SPDEs (see, e.g., [Linß, 2010, p7]).

In Chapter 5, we extend the method of Chapter 4 so that, instead of using *a priori* bounds on solutions to derivatives of the SPDE, the MPDE uses numerical estimates for the derivatives. Therefore, it can be considered to derive *a posteriori* layer-adapted meshes.

Chapter 6 builds on algorithms and methods of Chapter 4 extending them to two-dimensional problems. Specifically, we design and implement algorithms based on the MPDE method using *a priori* information about the physical problem. Section 6.2 has been published as Section 3 of [Hill and Madden, 2021]. The meshes generated are not restricted to tensor-product grids, unlike classic Bakhvalov meshes in two-dimensional space, as described in [Linß, 2010, §2.3]. Although the equidistribution principle does not extend easily to two-dimensions, in Section 6.3, we show that the MPDE equidistributes the mesh density function within an appropriate tolerance criterion. We demonstrate that solutions computed on meshes generated using the MPDE method are more accurate than those generated on a relevant range of tensor-product Bakhvalov meshes for a problem whose solutions exhibit layers that vary in width. We test the algorithm for the challenging test problems from Section 6.2.3. We also investigate if there are implications to performing a fixed number of iterations, rather than satisfying some tolerance-based stopping criterion.

In Chapter 7, we extend the algorithms of Chapter 5 to two-dimensional problems. That is, the MPDEs do not depend on *a priori* information about the derivatives of the related physical problem, and instead use the derivatives of the numerical solution at the boundary to formulate the mesh density function.

Solving PDEs arising in fluid dynamics such as the Navier-Stokes equations is a major

motivation for investigating SPDEs. Therefore, in Chapter 8, we generate meshes using the algorithms of Chapters 5 and 7 for one-dimensional time-dependent SPDEs and the Oseen equations, respectively. The solutions generated on the meshes are robust and converge in line with expectations in both N and ε.

In Chapter 9, we summarise the main results of this thesis, and discuss some open problems, and avenues for further investigations.

1.3 Notation

Throughout this thesis, we use the following notation:

- $\mathbb{R}^d$ is the real coordinate space of dimension d.
- Ω is an open, bounded subset of $\mathbb{R}^d$, $\overline{\Omega}$ denotes its closure, and $\partial\Omega$ its boundary. Specifically, we denote the spatial computational domain (on which the mesh PDE is posed) as $\Omega^{[c]}$. We denote the physical domain (on which the related steady SPDE is posed) as Ω. In Chapter 3 and Section 8.2, dedicated to time-dependent problems, the physical domain is denoted as $\Omega^{[x,t]}$ and the spatial physical domain as $\Omega^{[x]}$.
- ω denotes a partition, for the purposes of finite element discretization, into simplices of the physical domain $\overline{\Omega}$ (or $\overline{\Omega}^{[x]}$, in the case of time dependant problems). In one-dimension, these are intervals. In two-dimensions, they are triangles. In addition, $\omega^{[c]}$ denotes a partition of a computational domain, $\overline{\Omega}^{[c]}$. If in a particular situation, we study a sequence of partitions of $\overline{\Omega}^{[c]}$, then the i^{th} such partition is denoted $\omega^{[c,i]}$. Similarly, if we study a sequence of partitions of $\overline{\Omega}$ then the i^{th} such partition is denoted as $\omega^{[i]}$.
- Where necessary, we use a superscript to indicate a differential operator applied on the computational domain. In particular, we define

$$\nabla^{[c]} = \left(\frac{\partial}{\partial \xi_1}, \frac{\partial}{\partial \xi_2} \right), \quad \text{for } (\xi_1, \xi_2) \in \Omega^{[c]}.$$

- ε denotes the perturbation parameter, i.e., it is the coefficient of the higher-order term in the SPDE and when $\varepsilon \ll 1$ the solution to the SPDE exhibits sharp layers.
- We will always use N to denote the mesh parameter. It is the number of mesh intervals in one dimension, and, typically, the number of mesh intervals in each coordinate direction in two dimensions.
- C is an arbitrary positive constant that may take different values in different places, and is independent of the perturbation parameter, ε and N.

- The *Sobolev space* is defined as

$$W^{s,p}(\Omega) := \{u \in L^p(\Omega); \partial^\alpha u \in L^p(\Omega) \forall |\alpha| \leq s\},$$

with

$$L^p(\Omega) := \{u : \int_\Omega u^p dx < \infty\}.$$

In particular,

$$H^1(\Omega) := \{u \in L^2(\Omega); \partial u \in L^2(\Omega)\}$$

and

$$H_0^1(\Omega) := \{u \in L^2(\Omega); \partial u \in L^2(\Omega), u\big|_{\partial\Omega} = 0\}.$$

- The inner product is

$$(u, v) := \int_\Omega uv \, d\Omega,$$

and the usual L^2-norm is

$$\|u\|_0 = \sqrt{\int_\Omega u^2 d\Omega}.$$

- We denote the exact solutions to the SPDEs as u, and its numerical approximation as $u_{p,h}$ where p is the order of the finite element used to calculate the solution. If $p = 1$ we abbreviate $u_{1,h}$ as u_h.

- The maximum pointwise norm is

$$\|u\|_{\infty,\omega} = \max_{x_i \in \omega} |u(x_i)|,$$

when the domain is in $\mathbb{R}^2$, obviously this is over (x_i, y_i) in ω. When clear from context we drop the ω.

- The H^1 semi-norm is defined as

$$|u|_1 = \|\nabla u\|_0.$$

- The energy norm is defined as

$$\|u\|_{\mathrm{E}} = \sqrt{\varepsilon^2 \|\nabla u\|_0^2 + \|u\|_0^2}. \tag{1.1}$$

- The balanced norm is defined as

$$\|u\|_{\mathrm{B}} = \sqrt{\varepsilon \|\nabla u\|_0^2 + \|u\|_0^2}. \tag{1.2}$$

Chapter 2

Background

2.1 Outline

This chapter introduces the key concepts on which this thesis is based. It does not include any novel ideas or algorithms, but serves to explain the problems we wish to solve, and the methods we will use.

In Section 2.2, we introduce singularly perturbed differential equations, and discuss why accurate numerical solutions are important. In Section 2.3, we describe the finite element method that is used in the thesis and the software we use to implement it. In Section 2.4, we review the concept of *robustness* of numerical solutions, and layer-adapted meshes, describing methods to generate these meshes and highlighting their suitability for singularly perturbed problems. Since our goal is to solve these problems accurately, we must have an understanding of how accuracy, and errors, are quantified. This leads to a discussion on norms, and now errors in these norms are estimated numerically. In Section 2.6, we discuss the concept of mesh adaptation, starting with the most common strategy: h-refinement. We present evidence to support the thesis that this approach may not be optimal for generating meshes for singular-perturbed problems. The moving mesh partial differential equation (MMPDE) method first introduced in [Huang et al., 1994b], is detailed in Section 2.7. We also introduce the idea of a *mesh partial differential equation* method based on the MMPDE method which we use to generate spatial meshes throughout this thesis.

2.2 Singularly perturbed differential equations

2.2.1 Differential equations

Differential equations arise in many aspects of science and engineering. They model how phenomena change over time or space, such as how a cancerous tumour grows, how a string

vibrates, or how a pollutant disperses. So solving them is of fundamental importance to science and engineering. Analytical solutions to differential equations are rarely available, so one must calculate reliable approximate solutions to these problems using numerical schemes.

Often there is a region of the domain where the solution is of particular interest, for example, where sharp changes occur in the solution resulting in layers, and we want to resolve these local effects without excessive computational cost.

2.2.2 Singularly perturbed differential equations

Singularly perturbed differential equations (SPDEs) are differential equations whose leading term is multiplied by a small positive parameter, usually denoted ε. They are *singularly perturbed* in the sense that they are well-posed for all positive values of ε, but ill-posed if one formally sets $\varepsilon = 0$. For a more formal definition see, e.g., [Roos et al., 2008, p.2] or [Linß, 2010, p.2-3]. Solutions to SPDEs typically exhibit boundary layers and may also contain interior layers. The elliptic singularly perturbed differential equations that we focus on in this thesis are primarily of the form

$$-\varepsilon\Delta u - \boldsymbol{q}\cdot\nabla u + ru = f \quad \text{in } \Omega \subseteq \mathbb{R}^d, \quad \text{with } u|_{\partial\Omega} = 0, \tag{2.1}$$

where $d = 1, 2$. Here $\boldsymbol{q}, r$ and f are given functions, $\varepsilon > 0$, and we assume that

$$r - \frac{1}{2}\text{div } \boldsymbol{q} > 0,$$

when $\boldsymbol{q}, r \neq 0$. This is a standard assumption in finite element analyses of convection-diffusion problems to ensure coercivity, see, e.g., [Roos et al., 2008, §III.1.1] or [Stynes and Stynes, 2018, §6.2]. (We also consider vector-valued variants of (2.1).)

When $\boldsymbol{q} \equiv \boldsymbol{0}$, (2.1) is called a *reaction-diffusion* equation. When $r = 0$, it is known as a *convection-diffusion* equation; otherwise it is a reaction-convection-diffusion equation.

The concept of a "reduced solution", which we denote u_0, is central to the study of SPDEs. It is the solution to the so-called "reduced problem" for (2.1), obtained by formally setting $\varepsilon = 0$ and neglecting certain boundary conditions. Consider (2.1) with $\boldsymbol{q} \equiv \boldsymbol{0}$. Then the reduced problem is simply $ru_0 = f$ (and is not subject to boundary conditions). The solution to (2.1) will feature layers near boundaries where u_0 does not agree with the boundary conditions. *Resolving these layers is a primary concern of the numerical solution of SPDEs.*

Throughout the thesis, our model one-dimensional scalar reaction-diffusion problem is

$$-\varepsilon^2 u'' + u = 1 - x, \quad u(0) = u(1) = 0. \tag{2.2}$$

The solution to (2.2) when $\varepsilon = 10^{-2}$ is shown in Figure 2.1. The reduced problem is $u_0 = 1 - x$. Since $u_0(0) = 1$ and $u_0(1) = 0$, the solution exhibits a single boundary layer near $x = 0$, i.e., where the value of the reduced problem does not agree with the boundary conditions applied to (2.2).

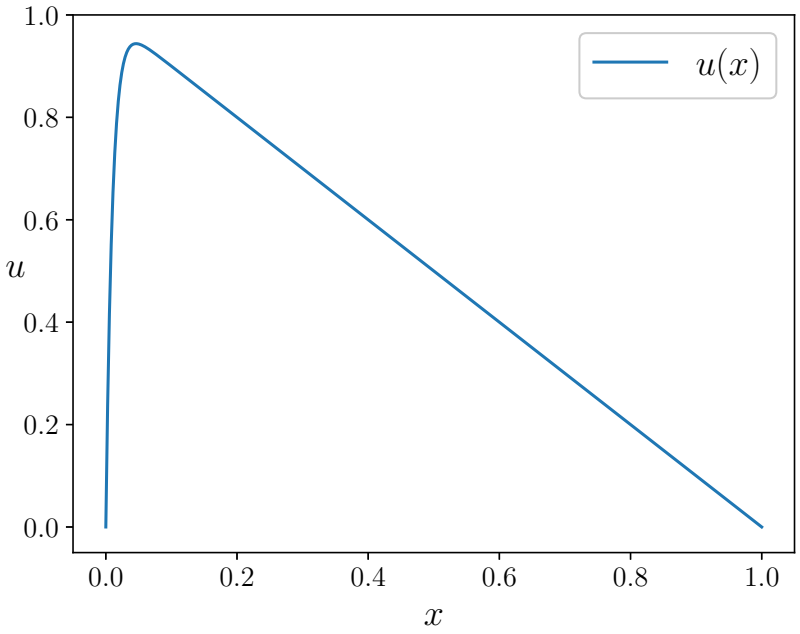


Figure 2.1: Solution to (2.2) when $\varepsilon = 10^{-2}$

2.3 Finite element methods

2.3.1 Finite element methods theory

In this thesis, we use finite element methods (FEMs) to solve the singularly perturbed differential equations. The bilinear form associated with (2.1), with $d = 1$ and $q = 0$, is

$$\mathcal{B}(u, v) := \varepsilon^2(u', v') + (ru, v).$$

Then the weak form of (2.1) is: *find* $u \in H_0^1(\Omega)$*, such that*

$$\mathcal{B}(u, v) = (f, v), \text{ for all } v \in H_0^1(\Omega). \tag{2.3}$$

From standard theory, see, e.g., [Grossmann and Roos, 2007, Chap. 3], the continuity and coercivity of the bilinear form ensure that (2.3) has a unique solution.

We obtain a finite element method by restricting the choice of u and v to a finite-dimensional subspace, V_0^h, of $H_0^1(\Omega)$. That is, we find $u_h \in V_0^h$, such that

$$\mathcal{B}(u_h, v_h) = (f, v_h), \text{ for all } v_h \in V_0^h. \tag{2.4}$$

Specifically, we compute the coefficients of the u_h, with respect to some basis, that solve (2.4).

To be more concrete we will focus on the simplest finite element method: the Galerkin method with piecewise-linear basis functions ($\mathcal{P}_1$-FEM). A mesh is a partition of the domain into intervals in one-dimension, and triangles or quadrilaterals in two-dimensions. We denote a generic one-dimensional mesh, with N intervals on $[a, b]$, as $\omega := a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b$. On an arbitrary mesh, we define the set of "hat functions", $\{\psi_i\}_{i=1}^{N-1}$, as

$$\psi_i(x) = \begin{cases} (x - x_{i-1})/(x_i - x_{i-1}) & \text{if } x_{i-1} \le x < x_i, \\ (x_{i+1} - x)/(x_{i+1} - x_i) & \text{if } x_i \le x \le x_{i+1}, \\ 0 & \text{otherwise,} \end{cases}$$

which are shown in Figure 2.2.

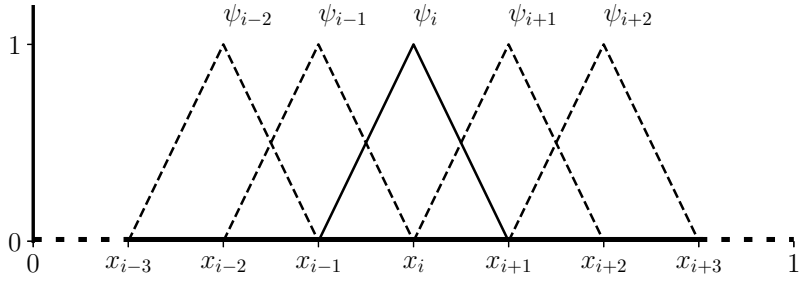


Figure 2.2: One dimensional piecewise linear basis functions (hat functions)

If we define V^h as the space spanned by these basis functions, then u_h can be written as

$$u_h = \sum_{i=1}^{N-1} \lambda_i \psi_i.$$

Taking

$$v_h = \psi_i \text{ for } i = 1, \ldots, N-1,$$

in (2.4) yields $N-1$ equations, which can be solved to give the λ_i.

Since the bilinear form is symmetric and positive definite (i.e., $\mathcal{B}(u, u) > 0$ for any non-zero u), it induces a norm, the "energy norm":

$$\|u\|_{\mathrm{E}} := \sqrt{\mathcal{B}(u, u)},$$

It is natural to analyse the error in an FEM solution with respect to this norm.

2.3.2 Finite elements methods implementation

The Galerkin FEM is very powerful, and yet can be quite complicated to implement. It is powerful in the sense that it generalises to many classes of PDEs, domains of various dimensions and shapes, and incorporates many spaces of approximating functions, all within

the same theoretical framework. Due to the "power of abstraction" [Eriksson et al., 1996], the mathematical formulation of the method is essentially the same regardless of these possible complications.

However, FEMs can be quite complicated to implement. One must discretize the domain, which usually means to triangulate it. Then one must compute transformations of quantities onto a reference element [Gockenbach, 2006, Chap. 12]; almost always quadrature is required to compute the associated integrals. Contributions from the elements are assembled to generate a system of linearly independent algebraic equations. Finally, this system of equations is solved; careful selection of suitable solvers for larger problems is usually necessary. Normally, tools to visualise the solution and calculate *a posteriori* error estimates are also needed.

It is clear that there is great potential value in general-purpose software which can automate as many of these tasks as possible. Notable examples include FEniCS [Alnæs et al., 2015, Logg et al., 2012b], deal.II [Arndt et al., 2019], MFEM [MFEM, 2019], FreeFEM [Hecht, 2012], and Firedrake [Rathgeber et al., 2017].

In this thesis we exclusively use FEniCS. It is a Python-based open-source software system that automates many aspects of implementing finite element methods for PDEs, while still allowing the user to get access to lower-level features. FEniCS is a collection of components; here we list the major ones.

- DOLFIN [Logg and Wells, 2010, Logg et al., 2012c] which is a C++/Python library that functions as the main user interface of FEniCS.
- The FEniCS Form Compiler (FFC) [Kirby and Logg, 2006, Logg et al., 2012a, Ølgaard and Wells, 2010], a compiler for finite element variational forms. In particular, a bilinear form can be assembled into a matrix and a linear form into a vector.
- The Unified Form Language (UFL) [Alnæs et al., 2014, Alnæs, 2012], which defines a flexible interface for choosing finite element spaces and defining expressions for weak forms in a notation close to mathematical notation.
- The FInite element Automatic Tabulator (FIAT) [Kirby, 2004, Kirby, 2012], a Python package for automatic generation of finite element basis functions.
- The Unified Form-assembly Code (UFC) [Alnaes et al., 2009, Alnæs et al., 2012] is a finite element code generation interface, between FFC and the DOLFIN environment.

The user specifies how to discretize the domain, and which family and degree of basis functions to use. Then, they define the weak form of the problem using a Python syntax that is very similar to the mathematical formula, and call FFC to assemble and solve the related system of algebraic equations. The code is very versatile and extends easily to higher-order elements, higher dimensions and non-regular domains.

In Listing 2.1, we show a snippet of FEniCS code, one observes that the syntax is similar to the mathematical formulation of (2.4).

Listing 2.1: The weak form of (2.4) for FEniCS.

```
f =  Expression ('1-x[0]', degree = 2) # Right-hand side of the SPDE
a = epsilon*epsilon*dot(grad(u), grad(v))*dx + dot(u,v)*dx # Bilinear form
L = f*v*dx # Linear form (right-hand side of the FEM equation)
```

In Appendix A we present the complete Python code for solving the one-dimensional singularly perturbed reaction-diffusion equation (2.2) in FEniCS on a uniform mesh.

2.4 Parameter robustness and layer-adapted meshes

In this section, we consider how to ensure the accuracy of the numerical solutions we generate and outline the necessity for layer-adapted meshes for SPDEs.

In Section 2.4.1 we discuss the importance of generating accurate solutions to SPDEs and outline some of the contributions of existing literature to this area. In Section 2.4.2 layer-adapted meshes are needed. In Sections 2.4.3 and 2.4.4 we discuss the *a priori* layer-adapted meshes of Bakhvalov and Shishkin in detail. In Section 2.4.5, we give a brief outline of the relevant history of adaptive methods for generating layer adapted meshes for SPDEs.

2.4.1 Parameter robustness

The mathematical analysis of numerical methods for SPDEs has been extensively studied over many years. Most studies focus on the concept of "*uniform convergence*" or "*parameter robustness*". We determine the robustness of our numerical solutions by establishing the independence (or otherwise) of error constants and orders of convergence with respect to the perturbation factor. The specific term "uniform convergence", sometimes called ε-"uniform convergence", is often defined with respect to pointwise convergence only. However, in [Linß, 2010, Chap. 1], Linß defines uniform convergence as follows.

Definition 2.4.1 (Parameter robustness/uniform convergence)**.** Let u be the solution of a singularly perturbed problem, and let u_h be a numerical approximation of u obtained by a numerical method with N degrees of freedom. The numerical method is said to be **parameter robust** (or **uniformly convergent**) with respect to the perturbation parameter ε in the norm $\|\cdot\|$ if

$$\|u - u_h\| \leq \vartheta(N), \qquad \text{for } N \geq N_0,$$

which satisfies a function ϑ with

$$\lim_{N\to\infty} \vartheta(N) = 0, \quad \frac{\partial \vartheta}{\partial \varepsilon} \equiv 0,$$

with some threshold value $N_0 > 0$ which is independent of ε.

Since the term "uniform convergence" is often associated with pointwise convergence, to avoid confusion, we use the term *parameter robustness* when referring to convergence as defined in Definition 2.4.1. Furthermore, ε may explicitly appear in the error bounds in L^2-type norms for certain problems. However, only positive powers of ε are present, and all ε-dependence is explicitly accounted for. Thus any constants in error bounds are independent of ε. Therefore, they are robust with respect to Definition 2.4.1.

In this section, we mention the monographs that are, arguably, the most influential in this field over the past 25 years. Together, they have almost 2,000 citations, over 100 of which are in 2021, according to `https://mathscinet-ams-org`, demonstrating that this is a vibrant area of research.

There are two types of widely studied fitted methods for solving SPDEs: fitted operator methods and fitted mesh methods. Generally, fitted operator methods involve specialised finite difference schemes, such as the Il'in method, and are typically applied on a uniform mesh [Miller et al., 1995]. However, since the 1990s, these methods have given way in popularity to *fitted mesh methods*. These usually involve classical methods, such as central finite differences or Galerkin FEMs, applied on highly specialised meshes that condense in the region of layers. The study of generating robust numerical solutions of SPDEs using fitted mesh methods is a subject of great interest over the last few decades, especially in conjunction with finite difference methods; see for example, [Miller et al., 1996] and [Shishkin and Shishkina, 2009].

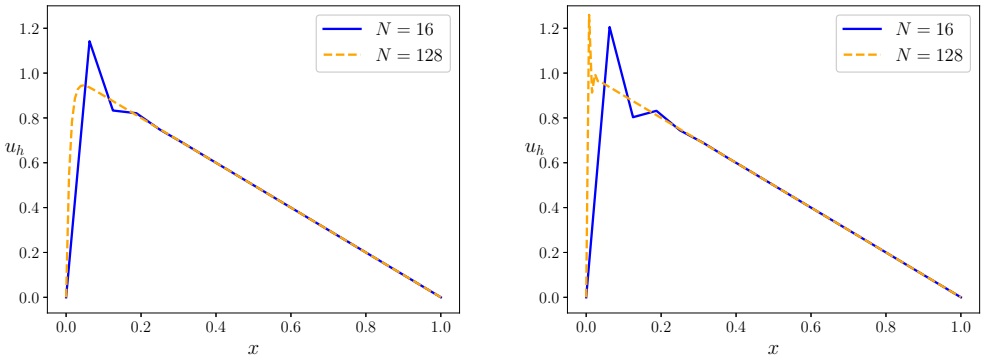
[Roos et al., 2008], which builds on the earlier edition [Roos et al., 1996], is a comprehensive *state-of-the-art* overview of robust numerical methods for SPDEs, with a particular emphasis on convection-diffusion problems. For example, in §III.5.3.2, FEMs on Shishkin meshes are studied for a two-dimensional convection-diffusion equation with exponential layers. For a monograph that focuses exclusively on the analysis of numerical methods on fitted meshes, see [Linß, 2010].

Research into the robustness of numerical solutions to SPDEs on layer-resolving meshes continues apace. Whereas the books mentioned above are aimed at a specialist audience, [Stynes and Stynes, 2018] is written to be accessible to graduate students, and gives an overview of convection-diffusion problems and their numerical solution, including FEM solutions generated on layer-adapted meshes.

Specific results on Bakhvalov and Shishkin meshes, relevant to this thesis, are discussed in Sections 2.4.3 and 2.4.4, respectively.

2.4.2 The necessity of layer-adapted meshes

A useful numerical solution to (2.2) should be both qualitatively and quantitatively representative of the exact solution. That is, it should accurately determine both the layer location and width. However, consider the numerical solutions to (2.2), shown in Figure 2.3, which were generated using uniform meshes. One can see that the solutions change rapidly near $x = 0$, but away from this layer region they closely resemble the reduced solution to (2.2), i.e., $u_0 = 1 - x$. In Figure 2.3(a), where $\varepsilon = 10^{-2}$, one can see that a mesh with $N = 16$ intervals does not resolve the layer region; taking $N = 128$ appears to be sufficient, but only because N is $\mathcal{O}(\varepsilon^{-1})$. This is demonstrated in Figure 2.3(b) where $\varepsilon = 10^{-4}$, and the numerical solution with $N = 128$ is unstable. In general, a satisfactory numerical solution is obtained with a uniform mesh only when N is $\mathcal{O}(\varepsilon^{-1})$, which is not feasible since we wish to accurately solve this problem for any positive ε, irrespective of how small it is.



(a) $\varepsilon = 10^{-2}$ (b) $\varepsilon = 10^{-4}$

Figure 2.3: FEM solutions to (2.2) on uniform meshes

Obviously, for a problem such as (2.2), we need a mesh that is very fine in the layer region. One strategy for generating such a mesh is to use asymptotic information concerning the solution (and its derivatives) and, specifically the location and width of the layer. Where this information is available, one can construct *a priori* fitted layer-resolving meshes. Among the class of such meshes that have been successfully applied to solving SPDEs, the graded mesh of Bakhvalov [Bakhvalov, 1969] and the piecewise uniform mesh of Shishkin [Shishkin, 1990] are the most widely studied. Both these methods yield parameter-robust numerical solutions, when standard discretizations are applied to them.

Numerous variations on Shishkin and Bakhvalov meshes have been proposed; see [Linß, 2010, Chap. 2] for an overview, including meshes that are graded in the layer region proposed Vulanović [Vulanović, 1987] and Gartland [Gartland, 1988].

2.4.3 Bakhvalov meshes

The first layer-adapted mesh for singularly perturbed problems was proposed in [Bakhvalov, 1969]. The mesh is graded in layer regions and uniform elsewhere. To motivate the mesh we will consider the reaction-diffusion problem (2.2), whose solution has a single boundary layer which is located near $x = 0$. Let $\exp(-\beta x/\varepsilon)$ be a function that represents this boundary layer. The mesh points x_i near $x = 0$ satisfy

$$\exp\left(\frac{-\beta x_i}{\sigma\varepsilon}\right) = 1 - \frac{i}{qN} \quad \text{for } i = 0, 1, \ldots, \tag{2.5}$$

where $q \in (0, 1)$ is roughly the proportion of mesh points used to resolve the layer, and $\sigma > 0$ is a constant that depends on the formal order of the underlying method.

One can describe a mesh by explicitly listing the coordinates of each x_i, but we shall do so in terms of a "mesh generating function".

Definition 2.4.2 (Mesh generating function)**.** A mesh generating function is a strictly monotonic bijective function $\varphi : \overline{\Omega}^{[c]} := [0, 1] \to \overline{\Omega} := [a, b]$ that maps a uniform mesh with mesh points $\xi_i = i/N$, for $i = 0, 1, \ldots, N$, to a (possibly non-uniform) mesh with mesh points $x_i = \varphi(i/N)$, for $i = 0, 1, \ldots, N$, with $\varphi(0) = a$ and $\varphi(1) = b$.

Bakhvalov meshes can be constructed using the classic construction method presented by Bakhvalov in [Bakhvalov, 1969], which we detail in Section 2.4.3. Alternatively, they can be constructed using the equidistribution principle, as described in Section 2.4.3.

Classic construction

The mesh is expressed in terms of the mesh generating function,

$$\varphi(t) = \begin{cases} \chi(t) := -\dfrac{\sigma\varepsilon}{\beta} \ln\left(1 - \dfrac{t}{q}\right), & \text{for } t \in [0, \tau], \\ \pi(t) := \chi(\tau) + (t - \tau)\varphi'(\tau), & \text{elsewhere.} \end{cases} \tag{2.6}$$

The mesh generated has mesh points $x_i = \varphi(i/N)$ for $i = 0, 1, \ldots, N$. For the standard Bakhvalov mesh on $[0, 1]$, one chooses τ in (2.6) to satisfy

$$\varphi'(\tau) = \frac{1 - \varphi(\tau)}{1 - \tau}, \tag{2.7}$$

so that $\varphi(1) = 1$, see [Roos et al., 2008, (2.136)]. An example of a Bakhvalov mesh for (2.2) is shown in Figure 2.4. It is graded in the layer region near $x = 0$ and uniform elsewhere. One observes in Figure 2.5, that $\varphi(\tau)$ is the transition point between the graded and uniform mesh. Geometrically, $\pi(t)$ is a line segment on the unique tangent to $\chi(t)$, which goes through $(1, 1)$, and, thus, $\varphi(\tau)$ is the point where $\chi(t)$ and the tangent intersect. The resulting mesh generating function is $C^1[0, 1]$. The location of the mesh points in the layer

region is determined by $\chi(t)$ for $t \in [0, \tau]$, while the location of the coarsely spaced points outside the layer region is determined by $\pi(t)$. We note that

$$\varphi(0) = -\frac{\sigma\varepsilon}{\beta}\ln(1) = 0,$$

and

$$\varphi(1) = \chi(\tau) + \frac{1-\chi(\tau)}{1-\tau}(1-\tau) = 1,$$

as required.

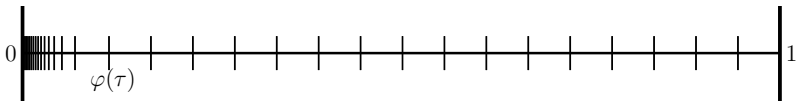


Figure 2.4: Example of a Bakhvalov mesh for (2.2) with $\varepsilon = 10^{-4}$ and $N = 32$

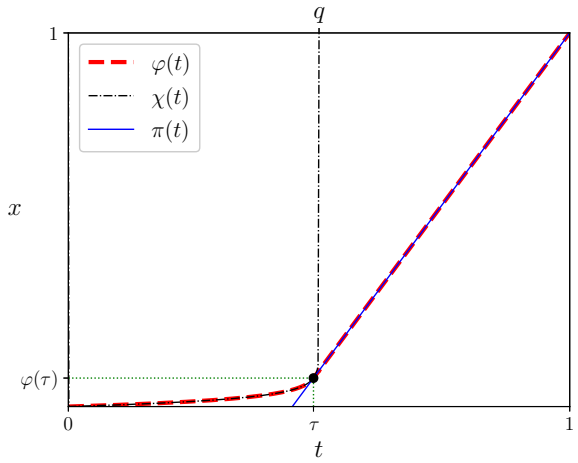


Figure 2.5: Mesh generating function for a Bakhvalov mesh

The nonlinear problem (2.7) cannot be solved explicitly. We rearrange (2.7) and get

$$0 = (1-\tau)\varphi'(\tau) - 1 + \varphi(\tau). \tag{2.8}$$

We let $f(\tau) = 0$, and rewrite (2.8) so $g(\tau) = \tau$. Then, we use a fixed-point method based on Newton's iteration:

$$g(\tau) = \tau - \frac{f(\tau)}{f'(\tau)}, \quad \text{where } g(\tau) = \tau \text{ if } f(\tau) = 0.$$

We solve for τ until $|g(\tau) - \tau| < \text{TOL}$, some suitable user chosen tolerance.

A numerical solution to (2.2) with $\varepsilon = 10^{-2}$ and $N = 32$ solved on a Bakhvalov mesh is shown in Figure 2.6. One can see that the layer region is resolved. The position of the mesh points in the layer region is dependent on ε and outside this region the mesh is uniform.

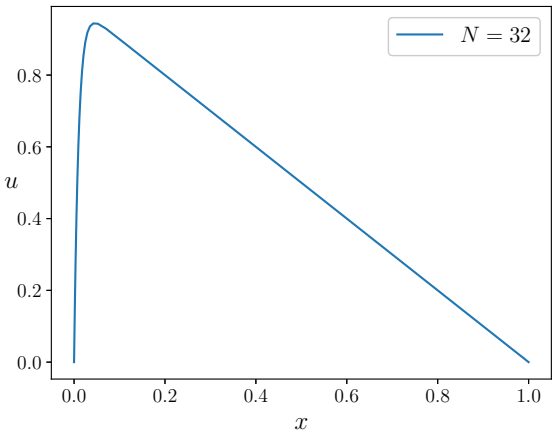


Figure 2.6: FEM solution to (2.2) with $\varepsilon = 10^{-2}$ and $N = 32$ solved on a Bakhvalov mesh

Equidistribution

The concept of equidistribution stems from spline interpolation [de Boor, 1974]. Its application to mesh generation for SPDEs dates to before 2000 in the context of adaptive refinement, for example, see [Mackenzie, 1999]. For *a priori* adaptive meshes, the idea is developed by Linß [Linß, 2010].

Definition 2.4.3 (The equidistribution principle)**.** Let $\omega := a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b$ be a mesh that discretizes $\overline{\Omega} := [a, b]$, and $\rho := \overline{\Omega} \to \mathbb{R}_{>0}$ be a strictly positive function known as the *mesh density function*. Then, ω is said to equidistribute ρ, if

$$\int_{x_{i-1}}^{x_i} \rho(x)dx = \frac{1}{N}\int_{\overline{\Omega}} \rho(x)dx \quad \text{for } i = 1, \ldots, N.$$

Specifically, when generating Bakhvalov meshes for the reaction-diffusion equation (2.2), one equidistributes

$$\rho(x) = \max\left\{1, K\frac{\beta}{\varepsilon}\exp\left(\frac{-\beta x}{\sigma\varepsilon}\right)\right\}, \tag{2.9}$$

where $K > 0$ determines the portion of mesh points in the layer region, $\sigma > 0$ is a constant that depends on the formal order of the underlying method. Away from the layer, the mesh is uniform, i.e., when $\rho(x) = 1$.

Parameter robustness

The robustness of numerical solutions to various SPDEs solved on Bakhvalov meshes has been proven in the literature. For example, in [Kellogg et al., 2008] it is proven that the errors measured in the maximum pointwise norm are bounded as

$$\|u - u_h\|_\infty \leq CN^{-2},$$

for a coupled system of two-dimensional reaction-diffusion equations, posed on the unit square, and solved on a tensor product Bakhvalov mesh using a finite difference method. In [Linß, 2010, Chap. 6], it is shown that $\mathcal{P}_1$-FEM solutions of one-dimensional reaction-diffusion equations solved on Bakhvalov meshes satisfy

$$\|u - u_h\|_{\mathrm{E}} \leq C(\varepsilon^{1/2} N^{-1} + N^{-2}). \tag{2.10}$$

An upwind finite difference method for a two-dimensional convection-diffusion on a tensor-product Bakhvalov mesh is studied in [Nhan and Vulanović, 2021], and robust convergence in the maximum pointwise norm of $\mathcal{O}(N^{-1})$ is established.

2.4.4 Shishkin meshes

A Shishkin mesh [Shishkin, 1990] is a piecewise uniform mesh (i.e., it is not graded), which is fine (but uniform) in layer regions and coarse elsewhere. When generating a Shishkin mesh for (2.2), the transition point τ_S between the fine and coarse parts of the mesh is determined by

$$\tau_S = \min\left\{q, \frac{\sigma\varepsilon}{\beta}\ln N\right\}.$$

Here q is the proportion of the mesh intervals that are allocated to the layer region $[0, \tau_S]$ and $\sigma > 0$ is a constant that depends on the formal order of the underlying method. The mesh points are uniformly distributed with $\lceil qN \rceil$ intervals in the layer region, and $\lfloor (1-q)N \rfloor$ intervals in $[\tau_S, 1]$, if $\tau_S = q$ this results in a uniform mesh. The mesh can be created using the mesh generating function (defined in Definition 2.4.2),

$$\varphi(t) = \begin{cases} \chi(t) := \dfrac{\sigma\varepsilon}{\beta}\dfrac{t}{q}\ln N, & \text{for } t \in [0, q], \\ \pi(t) := 1 - \Big(1 - \dfrac{\sigma\varepsilon}{\beta}\ln N\Big)\dfrac{1-t}{1-q}, & \text{otherwise.} \end{cases}$$

The resulting mesh will have the form shown in Figure 2.7.

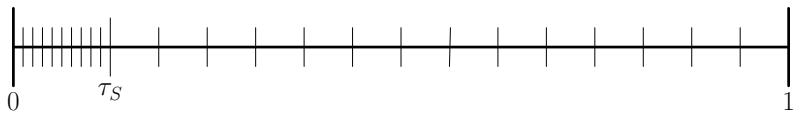


Figure 2.7: Example of a Shishkin mesh for (2.2)

An example of the mesh generating function for a Shishkin mesh is shown in Figure 2.8. One observes that unlike the Bakhvalov mesh, shown in Figure 2.5, the mesh generating function is piecewise linear and is not differentiable at the transition point. (Therefore, a Shishkin mesh cannot be generated using the MPDE methods that are the main topic of this thesis. However, very similar meshes are possible.)

The numerical solution to (2.2) with $\varepsilon = 10^{-2}$, solved on a Shishkin mesh with $N = 32$

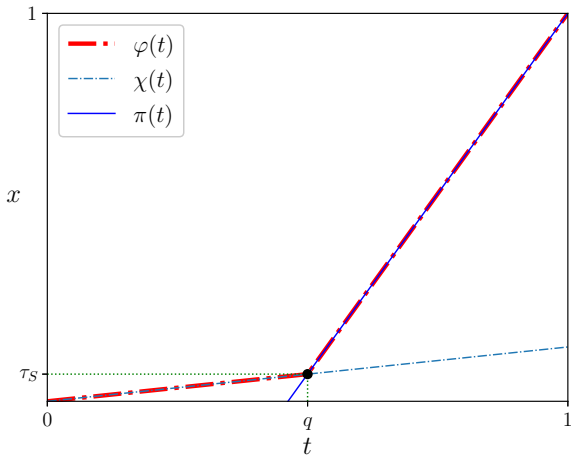


Figure 2.8: Mesh generating function for a Shishkin mesh

intervals, is shown in Figure 2.9; one can observe that the solution the layer region is resolved.

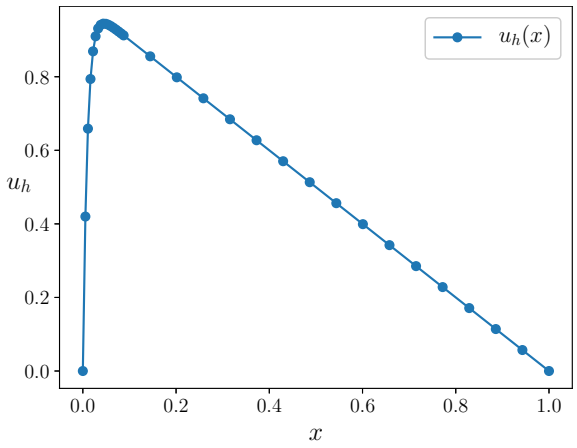


Figure 2.9: FEM solution to (2.2) with $\varepsilon = 10^{-2}$ and $N = 32$ solved on a Shishkin mesh

Parameter robustness

Since the 1990s, much research has been undertaken into the parameter robustness of numerical solutions on *a priori* fitted meshes. For example, it has been proven that the finite-difference solution to a one-dimensional reaction-diffusion problem solved on a Shishkin mesh satisfies the pointwise bound

$$\|u - u_h\|_\infty \leq C\left(N^{-2}\ln^2 N\right);$$

see [Miller et al., 2012, Chap. 6] for a full proof. This has been extended to, for example, two-dimensional problems [Clavero et al., 2005], coupled-systems [Linß and Madden, 2004], and to a time-dependent problem [Clavero and Gracia, 2010], where the backward Euler

method is used to discretize time. In this last case, robust convergence in the maximum pointwise norm, of $\mathcal{O}(M^{-1}+N^{-2}\ln N^2)$ is established, where M is the number of intervals in time.

For finite element methods, the $\mathcal{Q}_1$-FEM solution of two-dimensional reaction-diffusion equations are considered in [Liu et al., 2009] and robust convergence in the energy norm of $\mathcal{O}(\varepsilon^{1/2}N^{-1}\ln N + N^{-2})$ on a Shishkin mesh is shown. In [Wang et al., 2021], the same bounds are established for a finite volume element method for the same problem and mesh.

2.4.5 Adaptive schemes for generating layer-adapted meshes

The meshes discussed in Sections 2.4.3 and 2.4.4 are *a priori fitted meshes*. Although not studied as extensively, there are robust *adaptive* methods for constructing layer-adapted meshes for SPDEs.

The first significant effort in this direction, from the perspective of this thesis, is [Mackenzie, 1999]. In that paper, a linear convection-diffusion problem is solved using an upwind finite difference scheme. A fitted mesh is generated by equidistributing a monitor function based on the first derivative of the numerical solution. Convergence of solutions generating using the method is proven to be robust, a fact verified by the results of numerical experiments.

The method of [Mackenzie, 1999] is enhanced in [Beckett and Mackenzie, 2000] where the monitor function is based on the second derivative of the layer component of the numerical solution to a convection-diffusion equation. The monitor function used is

$$M(u_h(x),x) = \alpha + |w_h''|^{1/m}, \quad m \geq 2, \tag{2.11}$$

the positive constant $\alpha = \int_0^1 |w_h''|^{1/m} dx$ is added to prevent mesh starvation outside the layer region. The same monitor function is used in [Beckett and Mackenzie, 2001a] to generate layer-adapted meshes for a convection-diffusion problem. Robust convergence of the solutions to the SPDEs is proven and verified by the numerical results in both papers.

This concept was further developed in [Kopteva and Stynes, 2001], where the arc-length monitor function is equidistributed to generate the layer adapted meshes for convection-diffusion problems using an upwind finite difference method. It is proved that a solution to the equidistribution problem exists, that the algorithm to solve it converges, and that the resulting numerical method for the SPDEs is parameter robust.

Much of the work in this thesis is concerned with reaction-diffusion equations using FEMs. In [Beckett and Mackenzie, 2001b] robust convergence is proven for higher-order finite element solutions of one-dimensional reaction-diffusion problems on grids generated using the same monitor function as in [Beckett and Mackenzie, 2000]. The rates of convergence depend on the degree of FEM space and the value chosen for m in the monitor

function (2.11).

These papers are much cited in the literature, and continue to be the basis for solving many types of SPDE, for example in [Liu et al., 2020a, Liu et al., 2020b, Liu and Chen, 2017], where meshes are generated by equidistributing the arc-length monitor function for the Burgers-Huxley equation, an SPDE with integral boundary conditions, and a coupled system of convection-diffusion equations, respectively. In [Long et al., 2021], the equidistribution principle is used to generate meshes for the Volterra integro-differential equation. Also, in [Das, 2019] meshes are generated by equidistributing a monitor function based on the first and second derivatives of the numerical solution of singularly perturbed delay differential equations.

However, since the generation of meshes by adaptive refinement is core to this thesis, we postpone further discussion of this to Section 2.6.

2.5 Quantifying error

Our goal is to generate accurate numerical solutions to SPDEs. Therefore, we need to determine an appropriate method of measuring their accuracy. In Section 1.3 we define three norms: the maximum pointwise norm, the energy norm and the balanced norm. Of these, since it is induced directly by the bilinear forms for reaction-diffusion equations, the energy norm (1.1) is our primary concern. However, these norms have been criticised in the literature for being too "weak" to identify layers in reaction-diffusion equations [Farrell et al., 2000], leading to the proposals for so-called *"balanced norms"* (1.2), originally introduced in [Lin and Stynes, 2012], where the ε weighting on the H^1 semi-norms is adjusted so that the final error bounds are independent of ε. The maximum pointwise norm is the natural setting for analysing finite difference schemes. Since these methods do not feature in the thesis, that norm is not considered further.

2.5.1 Quantifying and calculating errors

Throughout this work, we need to assess the quality of numerical solutions computed by our algorithms, and to compare them with the accuracy of competing methods. In addition, on occasion, we need to determine the region of the domain where the largest error occurs. For both these reasons, we need accurate error estimators, particularly as, typically, the true solution to the differential equation we are approximating is not available.

Our preferred approach is to estimate the true error by comparing it to another numerical solution that, in theory, is more accurate. There are other approaches in the literature (see [Roos et al., 2008, p. 408-414] and [Verfürth, 1994, p.445-446]) for estimating the errors in finite element solutions, including

1. residual estimators, for example with respect to the strong form of the differential

equation.

2. the solution of local problems, such as residual equations posed on an element. (This is discussed further in [Roos et al., 2008, §III.3.6], with details on the Babuška-Rheinboldt, Verfürth, and modified Bank-Weiser estimators).

3. higher-order gradient recovery, i.e., if an estimate Gu_h of the gradient of the numerical solution is easy to calculate then $\|Gu_h - \nabla u\|_0$ can be used to estimate the energy norm,

4. goal-oriented estimators which rely on duality and are also known as DWR (dual weighted residuals), and

5. *a priori* error estimators.

The purpose of this section is to verify that the approach we propose is efficient and robust, which we do through extensive numerical experiments for a one-dimensional reaction-diffusion equation on Bakhvalov meshes in the energy norm, (1.1).

2.5.2 Error estimation

The requirements that a useful error estimator should satisfy are discussed by [Grätsch and Bathe, 2005]. The first of these is that the estimated error should be "close" to the actual error. Letting E_h be the estimate of the error, and $e_h = u_h - u$ be the actual error then the *efficiency index*,

$$\lambda_\star = \frac{\|E_h\|_\star}{\|e_h\|_\star},$$

should be close to 1, for the appropriate norm $\|\cdot\|_\star$. Since we consider only the energy norm (1.1) in this section, we'll investigate

$$\lambda = \lambda_{\mathrm{E}} = \frac{\|E_h\|_{\mathrm{E}}}{\|e_h\|_{\mathrm{E}}}.$$

We now evaluate three different approaches for computing E_h for our model one-dimensional reaction-diffusion test problem (2.2), whose exact solution is

$$u(x) = 1 - x + \frac{\exp(-x/\varepsilon) - \exp((x-2)/\varepsilon)}{\exp(-2/\varepsilon) - 1}.$$

We compute $\mathcal{P}_1$-FEM solutions on Bakhvalov meshes generated using the classic construction method of Section 2.4.3. The three methods of calculating E_h we consider are as follows.

1. Taking $E_h = u_{2,h} - u_h$, where $u_{2,h}$ is the $\mathcal{P}_2$-FEM solution, generated on the same mesh. The calculated errors are, in essence, the same as the hierarchical error estimations used in [Huang et al., 2010] and [Brezzi and Russo, 1994]. Results are shown

in Table 2.1.

2. Taking $E_h = u_{2,h} - u_h$, where $u_{2,h}$ is the $\mathcal{P}_1$-FEM solution generated using $2N$ intervals, i.e., on a mesh constructed by bisecting each interval in the mesh used to calculate the $\mathcal{P}_1$-FEM solution. This is sometimes called the "double mesh" or "two mesh" method [Farrell et al., 2000, Chap. 8]. Results are shown in Table 2.2.

3. Taking $E_h = u_{2,h} - u_h$, where $u_{2,h}$ is the interpolant of u_h on a function space of $\mathcal{P}_2$ elements defined on a uniformly coarsened mesh. The results are shown in Table 2.3.

Table 2.1: Values of λ for (2.2) when $u_{2,h}$ is the $\mathcal{P}_2$-FEM solution generated on the same mesh

ε^2/N	32	64	128	256	512	1024
1	0.9999719	0.9999930	0.9999982	0.9999996	0.9999999	1.0000000
10^{-2}	0.9995966	0.9998990	0.9999747	0.9999937	0.9999984	0.9999996
10^{-4}	0.9993996	0.9998507	0.9999627	0.9999907	0.9999977	0.9999994
10^{-6}	0.9993032	0.9998402	0.9999614	0.9999905	0.9999976	0.9999994
10^{-8}	0.9992040	0.9998277	0.9999598	0.9999903	0.9999976	0.9999994
10^{-10}	0.9991083	0.9998156	0.9999583	0.9999901	0.9999976	0.9999994
10^{-12}	0.9990137	0.9998037	0.9999568	0.9999899	0.9999976	0.9999994

Table 2.2: Values of λ for (2.2) when $u_{2,h}$ is the $\mathcal{P}_1$-FEM solution generated using $2N$ intervals

ε^2/N	32	64	128	256	512	1024
1	1.012623	1.012620	1.012619	1.012619	1.012619	1.012619
10^{-2}	1.012207	1.012516	1.012593	1.012612	1.012617	1.012618
10^{-4}	1.011143	1.012369	1.012570	1.012608	1.012616	1.012618
10^{-6}	1.010386	1.012277	1.012559	1.012607	1.012616	1.012618
10^{-8}	1.010376	1.012276	1.012559	1.012607	1.012616	1.012618
10^{-10}	1.010354	1.012273	1.012559	1.012607	1.012616	1.012618
10^{-12}	1.010379	1.012276	1.012559	1.012607	1.012616	1.012618

Table 2.3: Values of λ for (2.2) when $u_{2,h}$ is the $\mathcal{P}_1$-FEM solution interpolated onto $\mathcal{P}_2$ elements

ε^2/N	32	64	128	256	512	1024
1	0.9996123	0.9999031	0.9999758	0.9999939	0.9999985	0.9999996
10^{-2}	0.9662018	0.9813643	0.9902187	0.9949886	0.9974635	0.9987240
10^{-4}	0.9933297	0.9829994	0.9895221	0.9944293	0.9971495	0.9985604
10^{-6}	1.0173920	0.9859697	0.9898786	0.9944721	0.9971547	0.9985611
10^{-8}	1.0176200	0.9859982	0.9898820	0.9944725	0.9971547	0.9985611
10^{-10}	1.0183500	0.9860895	0.9898930	0.9944738	0.9971549	0.9985611
10^{-12}	1.0175100	0.9859845	0.9898803	0.9944723	0.9971547	0.9985611

Comparing Tables 2.1–2.3, one observes that all three methods have an efficiency index close to 1, but the closest is certainly in Table 2.1. Therefore, for the remainder of this

thesis, for any norm $\|\cdot\|_*$, we estimate the error in that norm as

$$\|E_h\|_\star = \|u_{2,h} - u_h\|_\star, \tag{2.12}$$

where u_h is the $\mathcal{P}_1$-FEM solution and $u_{2,h}$ is the $\mathcal{P}_2$-FEM solution, generated on the same mesh.

For completeness, and because these results are useful for later comparisons, in Tables 2.4 and 2.5 we present the values for $\|E_h\|_{\mathrm{E}}$ and $\|e_h\|_{\mathrm{E}}$ used to generate Table 2.1. Comparing Tables 2.4 and 2.5, we note that for the smallest reported value of N the estimate is correct to three digits, and for the largest N the values agree to six digits.

Table 2.4: Values of $\|E_h\|_{\mathrm{E}}$ (i.e., error estimated in the energy norm using a P_2-FEM) when (2.2) is solved on a Bakhvalov mesh with a $\mathcal{P}_1$-FEM

ε^2/N	32	64	128	256	512	1024
1	4.895123e-03	2.447683e-03	1.223857e-03	6.119302e-04	3.059653e-04	1.529827e-04
		1.00	1.00	1.00	1.00	1.00
10^{-2}	1.278415e-02	6.389206e-03	3.194262e-03	1.597089e-03	7.985394e-04	3.992690e-04
		1.00	1.00	1.00	1.00	1.00
10^{-4}	4.120288e-03	2.059119e-03	1.029426e-03	5.146968e-04	2.573463e-04	1.286729e-04
		1.00	1.00	1.00	1.00	1.00
10^{-6}	1.302916e-03	6.511507e-04	3.255347e-04	1.627622e-04	8.138043e-05	4.069013e-05
		1.00	1.00	1.00	1.00	1.00
10^{-8}	4.119961e-04	2.059106e-04	1.029430e-04	5.146991e-05	2.573475e-05	1.286735e-05
		1.00	1.00	1.00	1.00	1.00
10^{-10}	1.302789e-04	6.511426e-05	3.255342e-05	1.627621e-05	8.138043e-06	4.069013e-06
		1.00	1.00	1.00	1.00	1.00
10^{-12}	4.119645e-05	2.059084e-05	1.029429e-05	5.146990e-06	2.573475e-06	1.286735e-06
		1.00	1.00	1.00	1.00	1.00

Table 2.5: Values of $\|e_h\|_{\mathrm{E}}$ (i.e., the exact error in the energy norm) when (2.2) is solved on a Bakhvalov mesh with a $\mathcal{P}_1$-FEM

ε^2/N	32	64	128	256	512	1024
1	4.895260e-03	2.447700e-03	1.223859e-03	6.119305e-04	3.059654e-04	1.529827e-04
		1.00	1.00	1.00	1.00	1.00
10^{-2}	1.278931e-02	6.389852e-03	3.194343e-03	1.597099e-03	7.985406e-04	3.992692e-04
		1.00	1.00	1.00	1.00	1.00
10^{-4}	4.122763e-03	2.059427e-03	1.029465e-03	5.147016e-04	2.573469e-04	1.286730e-04
		1.00	1.00	1.00	1.00	1.00
10^{-6}	1.303824e-03	6.512548e-04	3.255473e-04	1.627637e-04	8.138062e-05	4.069016e-05
		1.00	1.00	1.00	1.00	1.00
10^{-8}	4.123243e-04	2.059461e-04	1.029472e-04	5.147041e-05	2.573481e-05	1.286736e-05
		1.00	1.00	1.00	1.00	1.00
10^{-10}	1.303952e-04	6.512627e-05	3.255478e-05	1.627637e-05	8.138062e-06	4.069016e-06
		1.00	1.00	1.00	1.00	1.00
10^{-12}	4.123712e-05	2.059489e-05	1.029473e-05	5.147042e-06	2.573481e-06	1.286736e-06
		1.00	1.00	1.00	1.00	1.00

In Section 2.4.3, we highlighted that the bound on the energy norm for solutions to

reaction-diffusion equations solved on Bakhvalov meshes satisfies

$$\|u_h\|_E \leq C\left(\varepsilon^{1/2}N^{-1} + N^{-2}\right).$$

Therefore, we want to verify that the errors and error estimates in our numerical solutions satisfy

$$\|e_h\|_E := \sqrt{\varepsilon^2\|\nabla e_h\|_0^2 + \|e_h\|_0^2} \leq C\big(\varepsilon^{1/2}N^{-1} + N^{-2}\big).$$

In Tables 2.6 and 2.7 one observes that the solutions convergence linearly in the H^1 semi-norm, and quadratically in the L^2-norm as expected and, as a result, the error estimates measured in the energy norm are consistent with the expected bound. One also observes that errors measured in the H^1 semi-norm term dominate in the errors measured in the energy norm. For comparison purposes, we present the exact errors measured in the

Table 2.6: Values of $\varepsilon^{1/2}\|\nabla E_h\|_0$ (i.e., error estimated in the weighted H^1 semi-norm using a P_2-FEM) when (2.2) is solved on a Bakhvalov mesh with $\mathcal{P}_1$-FEM

ε^2/N	32	64	128	256	512	1024
1	4.894906e-03	2.447656e-03	1.223853e-03	6.119298e-04	3.059653e-04	1.529827e-04
		1.00	1.00	1.00	1.00	1.00
10^{-2}	4.038326e-02	2.019899e-02	1.010046e-02	5.050354e-03	2.525193e-03	1.262598e-03
		1.00	1.00	1.00	1.00	1.00
10^{-4}	4.115354e-02	2.058510e-02	1.029351e-02	5.146873e-03	2.573452e-03	1.286728e-03
		1.00	1.00	1.00	1.00	1.00
10^{-6}	4.114937e-02	2.058490e-02	1.029354e-02	5.146896e-03	2.573463e-03	1.286733e-03
		1.00	1.00	1.00	1.00	1.00
10^{-8}	4.114646e-02	2.058468e-02	1.029352e-02	5.146895e-03	2.573463e-03	1.286733e-03
		1.00	1.00	1.00	1.00	1.00
10^{-10}	4.114480e-02	2.058456e-02	1.029351e-02	5.146894e-03	2.573463e-03	1.286733e-03
		1.00	1.00	1.00	1.00	1.00
10^{-12}	4.114375e-02	2.058449e-02	1.029351e-02	5.146893e-03	2.573463e-03	1.286733e-03
		1.00	1.00	1.00	1.00	1.00

Table 2.7: Values of $\|E_h\|_0$ (i.e., error estimated in the L^2-norm using a P_2-FEM) when (2.2) is solved on a Bakhvalov mesh with a $\mathcal{P}_1$-FEM

ε^2/N	32	64	128	256	512	1024
1	4.609607e-05	1.152503e-05	2.881320e-06	7.203341e-07	1.800859e-07	4.502512e-08
		2.00	2.00	2.00	2.00	2.00
10^{-2}	5.947170e-04	1.484952e-04	3.711286e-05	9.277577e-06	2.319355e-06	5.798388e-07
		2.00	2.00	2.00	2.00	2.00
10^{-4}	2.015722e-04	5.010957e-05	1.249554e-05	3.122226e-06	7.804195e-07	1.950967e-07
		2.01	2.00	2.00	2.00	2.00
10^{-6}	6.571414e-05	1.610281e-05	3.988501e-06	9.933844e-07	2.480695e-07	6.200646e-08
		2.03	2.01	2.01	2.00	2.00
10^{-8}	2.091989e-05	5.123828e-06	1.267615e-06	3.152369e-07	7.860525e-08	1.962752e-08
		2.03	2.02	2.01	2.00	2.00
10^{-10}	6.606684e-06	1.620153e-06	4.010492e-07	9.975330e-08	2.487345e-08	6.210165e-09
		2.03	2.01	2.01	2.00	2.00
10^{-12}	2.082990e-06	5.115767e-07	1.267392e-07	3.153706e-08	7.865291e-09	1.963895e-09
		2.03	2.01	2.01	2.00	2.00

H^1 semi-norm and L^2-norm in Tables 2.8 and 2.9, respectively. One observes that the discrepancies between $\|E_h\|_{\mathrm{E}}$ and $\|e_h\|_{\mathrm{E}}$, are due to the discrepancies in the H^1 semi-norm term, as seen by comparing Tables 2.6 and 2.8.

Table 2.8: Values of $\varepsilon^{1/2}\|\nabla e_h\|_0$ (i.e., the exact error in the weighted H^1 semi-norm) when (2.2) is solved on a Bakhvalov mesh with a $\mathcal{P}_1$-FEM

ε^2/N	32	64	128	256	512	1024
1	4.895043e-03	2.447673e-03	1.223855e-03	6.119300e-04	3.059653e-04	1.529827e-04
		1.00	1.00	1.00	1.00	1.00
10^{-2}	4.039958e-02	2.020103e-02	1.010072e-02	5.050386e-03	2.525196e-03	1.262599e-03
		1.00	1.00	1.00	1.00	1.00
10^{-4}	4.117903e-02	2.058820e-02	1.029389e-02	5.146921e-03	2.573458e-03	1.286728e-03
		1.00	1.00	1.00	1.00	1.00
10^{-6}	4.117901e-02	2.058826e-02	1.029394e-02	5.146945e-03	2.573469e-03	1.286734e-03
		1.00	1.00	1.00	1.00	1.00
10^{-8}	4.117808e-02	2.058820e-02	1.029394e-02	5.146945e-03	2.573469e-03	1.286734e-03
		1.00	1.00	1.00	1.00	1.00
10^{-10}	4.117736e-02	2.058815e-02	1.029393e-02	5.146945e-03	2.573469e-03	1.286734e-03
		1.00	1.00	1.00	1.00	1.00
10^{-12}	4.117688e-02	2.058811e-02	1.029393e-02	5.146944e-03	2.573469e-03	1.286734e-03
		1.00	1.00	1.00	1.00	1.00

Table 2.9: Values of $\|e_h\|_0$ (i.e., the exact error in the L^2-norm) when (2.2) is solved on a Bakhvalov mesh with a $\mathcal{P}_1$-FEM

ε^2/N	32	64	128	256	512	1024
1	4.609643e-05	1.152505e-05	2.881321e-06	7.203339e-07	1.800838e-07	4.502053e-08
		2.00	2.00	2.00	2.00	2.00
10^{-2}	5.947596e-04	1.484978e-04	3.711302e-05	9.277587e-06	2.319355e-06	5.798361e-07
		2.00	2.00	2.00	2.00	2.00
10^{-4}	2.001185e-04	4.999404e-05	1.248722e-05	3.121684e-06	7.803850e-07	1.950935e-07
		2.00	2.00	2.00	2.00	2.00
10^{-6}	6.516474e-05	1.601363e-05	3.977292e-06	9.922552e-07	2.479758e-07	6.199892e-08
		2.02	2.01	2.00	2.00	2.00
10^{-8}	2.116444e-05	5.136657e-06	1.267081e-06	3.149522e-07	7.855234e-08	1.962025e-08
		2.04	2.02	2.01	2.00	2.00
10^{-10}	6.867234e-06	1.646740e-06	4.035138e-07	9.995309e-08	2.488559e-08	6.210216e-09
		2.06	2.03	2.01	2.01	2.00
10^{-12}	2.228195e-06	5.280717e-07	1.285262e-07	3.172295e-08	7.883768e-09	1.965611e-09
		2.08	2.04	2.02	2.01	2.00

2.6 Introduction to mesh adaptivity

There are three primary strategies for generating adapted meshes: p-refinement, h-refinement and r-refinement. For p-refinement methods, the order of the elements is increased in areas of interest without changing the number of mesh points. However, that approach is not considered in this thesis. We now discuss h- and r-refinement in detail.

2.6.1 Mesh adaptivity using h-refinement

The most extensively used mesh adaptivity methods are the h-refinement methods. The idea is to estimate the error on every mesh interval and then create a new mesh by bisecting those intervals on which the error exceeds some threshold. In this section, we briefly examine the suitability of a h-refinement method for generating meshes on which to solve both one- and two-dimensional SPDEs, and evaluate the accuracy of the final solutions in the energy norm.

The approach we use is described in Algorithm 2.1. We initially solve the SPDE on a uniform mesh with 16 mesh intervals, and then determine the quality of the numerical solution of the SPDE by calculating the local error estimator,

$$\eta_i := \|\nabla u_{2,h} - \nabla u_h\|_{0,i} \quad \text{for } i = 1, \ldots, N,$$

on each mesh interval. Then, for all intervals where $\eta_i >$ TOL, a user-defined tolerance, we bisect the interval. We solve again on this new mesh, and recompute the local error estimators. This process is continued until $\max_{i=1,\ldots,N} \eta_i \leq$ TOL. This approach was first used in [Babuška and Rheinboldt, 1978] and is analysed in [Verfürth, 1994].

We use the convention that N is always the number of intervals in the mesh $\omega^{[k]}$ and that the $N+1$ mesh points are always ordered.
Input: $N = 16$, the number of intervals in the initial mesh.
Input: TOL.
Set $k = 0$;
Set $\omega^{[0]} := \{x_0, x_1, \ldots, x_N\}$ to be a uniform mesh discretizing $\overline{\Omega}$;
do
  Set u_h and $u_{2,h}$ to be the $\mathcal{P}_1$-FEM and $\mathcal{P}_2$-FEM solutions, respectively, to the SPDE, computed on $\omega^{[k]}$
  for i in $0 : N-1$ **do**
    calculate $\eta_i^{[k]}$ on cell $[x_i, x_{i+1}]$;
    if $\eta_i^{[k]} >$ TOL **then**
      Mark cell for refinement;
    end
  end
  Make $\omega^{[k+1]}$ by refining marked cells in $\omega^{[k]}$;
  $k \leftarrow k + 1$;
while $\max\{\eta_i^{[k]}\} >$ TOL;
Set $\omega := \omega^{[k]}$ to be the adapted mesh on $\overline{\Omega}$;

Algorithm 2.1: Generate a one-dimensional layer resolving mesh using h-refinement

In Figure 2.10, we present a plot of $\|E_h\|_E$ when the one-dimensional reaction-diffusion equation (2.2) is solved on meshes constructed as just described. For comparison purposes,

in Figure 2.11 we show the same information when the solutions are generated on Bakhvalov meshes. One can observe that the magnitude and robustness of the solutions are similar. However, we found that the number of iterations required increases as ε decreases, as shown in Table 2.10.

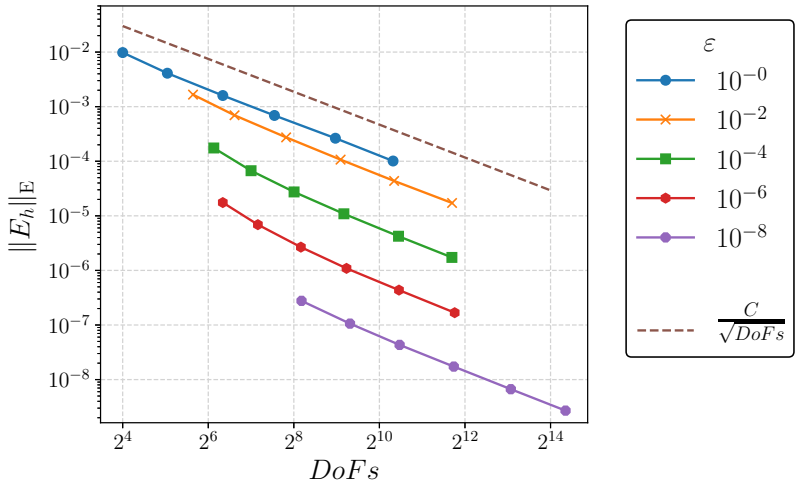


Figure 2.10: Plots of $\|E_h\|_{\mathrm{E}}$ for (2.2) when solved on h-refinement meshes generated using Algorithm 2.1

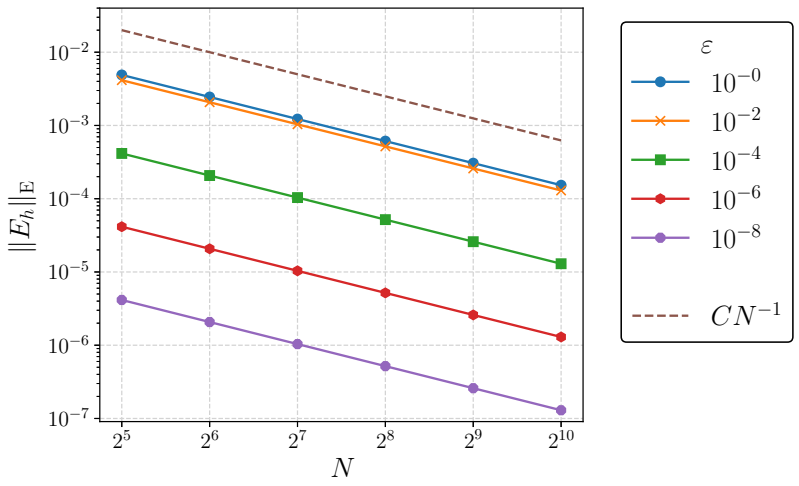


Figure 2.11: Plots of $\|E_h\|_{\mathrm{E}}$ for (2.2) when solved on Bakhvalov meshes

Then, we extend the method to the two-dimensional problem,

$$-\varepsilon^2\Delta u + u = 1 - x, \quad (x, y) \in \Omega := (0, 1)^2, \tag{2.13a}$$

with boundary conditions,

$$u(0, y) = u(1, y) = 0, \quad \frac{\partial u}{\partial y}(x, y) = \frac{\partial u}{\partial y}(x, y) = 0. \tag{2.13b}$$

Table 2.10: Number of iterations and DoFs needed to generate Figure 2.10.

ε	Iterations (DoFs)					
1	1 (16)	3 (33)	4 (81)	5 (187)	7 (502)	8 (1278)
10^{-2}	8 (50)	9 (98)	11 (226)	12 (546)	13 (1300)	15 (3323)
10^{-4}	15 (70)	16 (128)	17 (257)	19 (576)	20 (1396)	21 (3314)
10^{-6}	21 (81)	23 (143)	24 (287)	25 (601)	27 (1406)	28 (3467)
10^{-8}	31 (290)	32 (636)	33 (1421)	35 (3418)	36 (8611)	37 (20883)

The solution to (2.13) exhibits one layer near $x = 0$. Figure 2.12 shows the solution to (2.13) and the related mesh, when $\varepsilon = 10^{-2}$ with 663 degrees of freedom (DoFs).

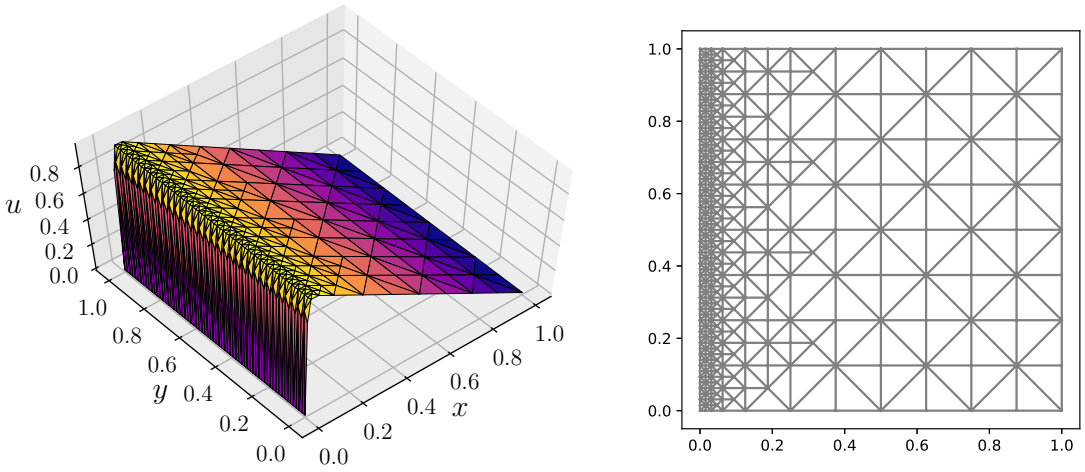


Figure 2.12: Solution and mesh generated for (2.13) using h-refinement with $\varepsilon = 10^{-2}$ with 663 DoFs

We generate the meshes for the two-dimensional SPDE (2.13) using a h-refinement method that is analogous to that outlined in Algorithm 2.1. When a cell is selected for refinement, facets are added that bisect each facet of the cell, resulting in the selected cell being divided into four cells. We add extra facets to the surrounding cells, to ensure there are no hanging nodes and the shape of all the cells, in the resulting mesh, are congruent. In Figure 2.13, we show an example of how a cell is refined.

In Figure 2.14, we present a plot of the estimate of the errors, $\|E_h\|_{\mathrm{E}}$ (2.12), for numerical solutions to (2.13) generated on these h-refinement meshes. For comparison purposes, in Figure 2.15, we present a plot of $\|E_h\|_{\mathrm{E}}$ for the numerical solutions to the same SPDE generated on Bakhvalov meshes. One observes that the magnitude of the errors in the solutions generated on the h-refinement meshes is larger than those generated on Bakhvalov meshes. Consider, for example, the results when $\varepsilon = 10^{-4}$. On a Bakhvalov mesh with 256 degrees of freedom, the error is less than 10^{-3}. The h-refinement method requires $296,915$ degrees of freedom to achieve comparable accuracy. Although this could certainly be improved using a more sophisticated approach (e.g., including mesh coarsening), it supports the argument that h-refinement is not the best choice of method to achieve robust

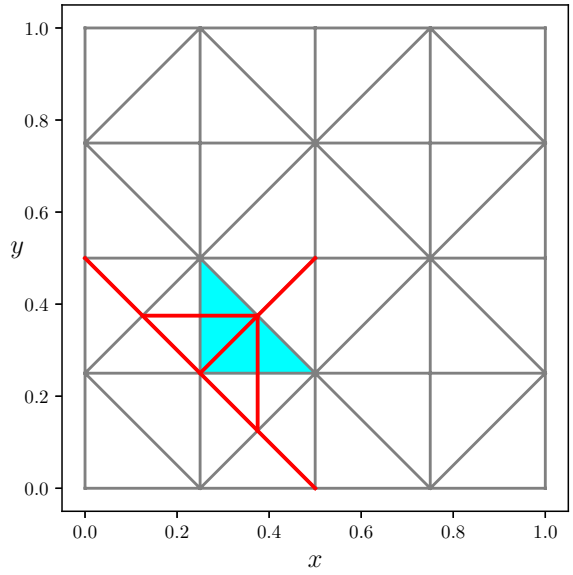


Figure 2.13: Refinement made to mesh if the highlight element is selected for refinement, new facets are shown in red

convergence for SPDEs.

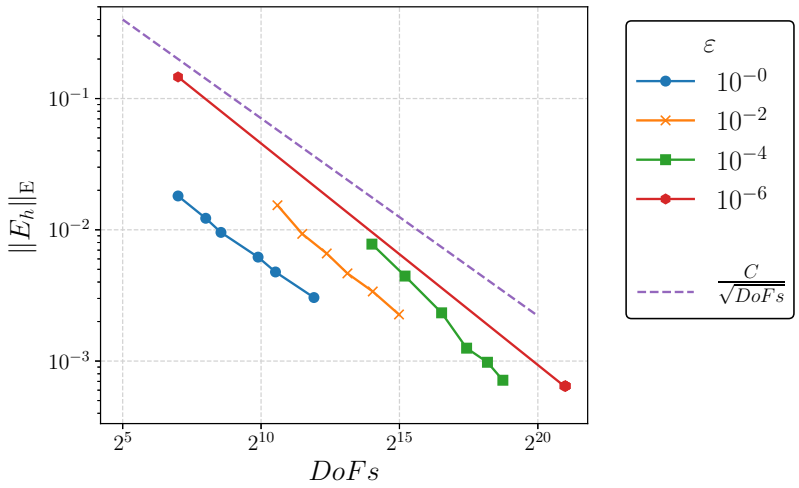


Figure 2.14: Plot of $\|E_h\|_{\mathrm{E}}$ (2.12) for (2.13) generated on h-refinement meshes

2.6.2 Mesh adaptivity using r-refinement

In contrast to the h-refinement methods, r-refinement methods, where our interest lies, relocate mesh points to control the error in the solution while fixing the mesh topology and the number of mesh points. These methods rely on the idea of equidistributing some monitor function across the domain of the related problem. The monitor function is often based on estimates of either the solution's derivatives, or errors in the numerical solution. The equidistribution method based on the spline interpolant of [de Boor, 1974] does not extend easily to two-dimensions, although some instances do exist in the literature,

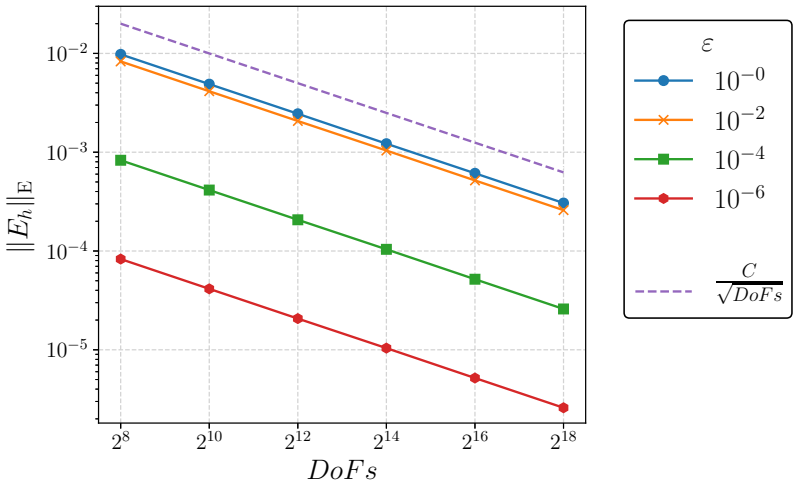


Figure 2.15: Plot of $\|E_h\|_{\mathrm{E}}$ (2.12) for (2.13) generated on Bakhvalov meshes

for example [Mackenzie, 1998], where the monitor function is equidistributed in each coordinate direction alternatively holding the other coordinate values fixed. Methods that extend more easily to two-dimensions include the moving mesh PDE (MMPDE), which have inspired the methods used throughout this thesis, and are discussed in more detail in Section 2.7.

2.7 Introduction to moving mesh PDEs

MMPDEs are presented in [Huang and Russell, 2011] and were developed for solving time-dependent PDEs where features of interest evolve through the domain in time. MMPDEs produce dynamically generated r-refinement meshes, i.e., the mesh points are relocated in time, placing a high proportion of the mesh points in areas of interest, and relatively fewer elsewhere in the domain.

Several different MMPDEs are presented in [Huang et al., 1994b], all based on the equidistribution principle discussed in Section 2.4.3. The resulting meshes are shown to only deviate slightly from an equidistributed mesh with an appropriate choice of time-step size. These MMPDEs are stated in terms of the mesh coordinates $x = x(\xi, t)$, and feature as a coefficient the nonlinear mesh density function $\rho = \rho(x(\xi, t), t)$. The MMPDEs are:

$$\frac{\partial}{\partial \xi}\left(\rho \frac{\partial \dot{x}}{\partial \xi}\right) + \frac{\partial}{\partial \xi}\left(\frac{\partial \rho}{\partial \xi}\dot{x}\right) = -\frac{\partial}{\partial \xi}\left(\frac{\partial \rho}{\partial t}\frac{\partial x}{\partial \xi}\right), \tag{MMPDE1}$$

$$\frac{\partial^2}{\partial \xi^2}(\rho \dot{x}) = -\frac{\partial}{\partial \xi}\left(\frac{\partial \rho}{\partial t}\frac{\partial x}{\partial \xi}\right) - \frac{1}{\gamma}\frac{\partial}{\partial \xi}\left(\rho \frac{\partial x}{\partial \xi}\right), \tag{MMPDE2}$$

$$\frac{\partial^2}{\partial \xi^2}(\rho \dot{x}) = -\frac{1}{\gamma}\frac{\partial}{\partial \xi}\left(\rho \frac{\partial x}{\partial \xi}\right), \tag{MMPDE3}$$

$$\frac{\partial}{\partial \xi}\left(\rho \frac{\partial \dot{x}}{\partial \xi}\right) = -\frac{1}{\gamma}\frac{\partial}{\partial \xi}\left(\rho \frac{\partial x}{\partial \xi}\right), \tag{MMPDE4}$$

$$\dot{x} = \frac{1}{\gamma}\frac{\partial}{\partial \xi}\left(\rho \frac{\partial x}{\partial \xi}\right), \tag{MMPDE5}$$

$$\dot{x} = \frac{1}{\gamma\rho}\frac{\partial}{\partial \xi}\left(\rho \frac{\partial x}{\partial \xi}\right), \tag{modified MMPDE5}$$

$$\frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{1}{\gamma}\frac{\partial}{\partial \xi}\left(\rho \frac{\partial x}{\partial \xi}\right), \tag{MMPDE6}$$

where $0 < \gamma \ll 1$ is the relaxation time and is independent of x and t. The equidistribution principle in its integral form is

$$\int_a^{x(\xi)} \rho(x,t)dx = \xi \int_a^b \rho(x,t)dx,$$

and its differential form is achieved by differentiating (2.7) twice with respect to ξ. However, this does not feature $\dot{x}$, so instead a "quasi-static" equidistribution principle is used to derive the MMPDEs, i.e., (2.7) is differentiated twice with respect to t.

MMPDE methods have been shown to be successful for generating meshes for many different types of PDE. For example, recent work in this direction includes applying the method to the shallow water equation [Zhang et al., 2022], two-dimensional Euler equations of gas dynamics [Kurganov et al., 2021], and to three-dimensional problems in weather forecasting [Browne et al., 2014].

For our part, in Chapter 3 we present an MMPDE method for time-dependent PDEs that feature layers. This then motivates the related methods, which we call "MPDEs" investigated in the rest of the thesis.

Chapter 3

Layer-adapted meshes for time-dependent problems with interior layers

3.1 Introduction

3.1.1 Motivation

In this chapter, we consider the numerical solution of one-dimensional, time-dependent problems, with interior layers (physical problems). Specifically, we look at problems with initial conditions which are such that the solution possesses layer regions that then propagate through the domain and evolve with time. Computing accurate solutions to interior layer problems usually depends on constructing a suitable layer-adapted mesh [Linß, 2010]. Therefore, we develop an algorithm to automatically generate adapted layer-resolving meshes using moving mesh partial differential equations (MMPDEs) first presented in [Huang et al., 1994b] and described in Section 2.7. By "automatic", we mean the meshes are generated without using *a priori* information about the location of the layers that they resolve. As the location of the interior layers change in time, at each time-step the MMPDE method produces meshes whose mesh points are relocated, so that the mesh points are concentrated in the layer regions at all time steps, making these meshes especially suited to these problems. We use the Galerkin finite element method implemented in FEniCS [Langtangen and Logg, 2016] to calculate the numerical solutions.

The challenges to overcome, when generating these meshes include:

- how to estimate the errors to guide the mesh refinement,
- how to project the solutions between the evolving meshes, and
- how to optimise the mesh generation process.

In this chapter, we focus on addressing these challenges for small, but specific, values of the coefficient of diffusion. We are not concerned with parameter robustness. This is because our ultimate goal is to solve realistic fluid dynamic problems, such as the Navier-Stokes equations on fully adaptive meshes, where the diffusion coefficients are related to specific properties of the fluid being modelled, or its flow.

3.1.2 Outline

The layout of the remainder of the chapter is as follows: In Section 3.2, we introduce the one-dimensional time-dependent parabolic problems that we investigate. In Section 3.3.1 we outline how the meshes are generated using the MMPDE method, and in Section 3.3.2 we present the discrete form of the MMPDE. In Section 3.3.3 we discuss the mesh density function, ρ, and how it is determined using *a posteriori* information about the physical problems being solved on the resulting meshes, and in Section 3.3.4 we outline how we optimise the mesh generating process by "smoothing" ρ. In Section 3.3.6 we describe how we project the solution to the physical problem between consecutive evolving spatial meshes. We detail the algorithm to generate the meshes in Section 3.3.7, and present examples of numerical solutions to linear and nonlinear one-dimensional time-dependent problems with interior layers and the meshes on which these solutions are computed, in Sectons 3.4.1 and 3.4.2, respectively.

3.1.3 Notation

In this chapter, we use the following notation, in addition to the notation outlined in Section 1.3.

- $\Omega^{[x,t]} = (0,1) \times (0,T]$, is the physical domain. At a given time-step, the spatial domain is denoted $\Omega^{[x]} = (0,1)$.
- M denotes the number of intervals in the time.
- $\Omega^{[c]}$ denotes the computational (spatial) domain on $(0,1)$, which is always discretized by a uniform mesh, $\omega^{[c]}$, with N intervals.
- $\tau = T/M$ denotes the length of each time-step.
- The time-steps are indexed by j, for $j = 0, 1, \ldots, M$.
- $\omega^{[j,k]} = \{x_0, x_1, \ldots, x_N\}$ is the mesh that discretizes $\Omega^{[x]}$, at the k^{th} iteration at time-step j.
- The spatial intervals are index by i, for $i = 1, 2 \ldots, N$.
- $h_i = (x_i - x_{i-1})$ is the local mesh width.
- $u_h^{[j]}$ is the numerical solution to the physical problem at time-step j.

- $x_h^{[j,k]}$ is the numerical solution to the MMPDE at the k^{th} iteration at time-step j, and $x_h^{[j]}$ is the final solution at time-step j.
- $\rho = \rho(u(x(\xi,t),t))$ is the mesh density function, and $\rho_i^{[j,k]}$ is its discretization on the i^{th} mesh interval, $[\xi_{i-1}, \xi_i]$, at the k^{th} iteration at time-step j.
- The parameter $\gamma > 0$ determines how quickly the mesh reacts to changes in ρ and is independent of ξ and t.
- $\|E_h\|_{*,\ddagger}$, is the error estimator, where $\|\cdot\|_*$ is the chosen norm and $\ddagger$ the relevant domain.
- κ is a user-defined intensity parameter.
- β is a user-defined parameter that determines how the mesh points are concentrated in the mesh, as described in [Huang and Sun, 2003].

3.2 Time-dependent problems with interior layers

We consider one-dimensional, time-dependent, parabolic problems, with first-order time derivatives and first- and second-order spatial derivatives, of the form:

$$\dot{u} - \varepsilon u'' + c(x,u)u' = 0, \text{ for } (x,t) \in \Omega^{[x,t]} := (a,b)\times(0,T], \tag{3.1a}$$

subject to the boundary and initial conditions

$$u(a,t) = q_0(t), \quad u(b,t) = q_1(t) \text{ for } t \in [0,T], \tag{3.1b}$$

$$u(x,0) = u_0(x) \text{ for } x \in \overline{\Omega}^{[x]} := [a,b], \tag{3.1c}$$

where, c, q_0, q_1 and u_0 are given smooth functions. As ever, ε is a small positive parameter. However, as discussed, we focus on the performance of the algorithms for specific values of ε, rather than arbitrarily small values. We investigate both linear and nonlinear variations of this problem. These problems are used to model advection-diffusion processes. Furthermore, solving such equations effectively is a key step towards solving the Navier-Stokes equations on fully adapted meshes.

We calculate the numerical solutions to (3.1) by combining the Galerkin $\mathcal{P}_1$-FEM and the Crank-Nicolson method [Crank and Nicolson, 1947]; see e.g., [Grossmann and Roos, 2007, §5.1.3] for a discussion of θ-methods in finite element settings. We begin by stating the continuous weak form for (3.1), find u in $H^1(\Omega^{[x,t]})$, such that,

$$(\dot{u}, v) + \varepsilon(u', v') + (cu', v) = 0, \quad \text{for all } v \in H_0^1(\Omega^{[x,t]}).$$

Since we wish to consider both linear and non-linear problems, their weak forms are different, so they are presented separately, in Sections 3.4.1 and 3.4.2.

3.3 Meshes

3.3.1 Generating the meshes

To ensure the 'moving' layer regions of the solutions are resolved when solving (3.1) numerically, the local mesh width needs to be very small, in a way that depends on ε, consequentially, uniform meshes are unsuitable. Therefore, we solve the physical problems on meshes generated using MMPDEs based on the equidistribution principle, as discussed in Section 2.7.

We implemented a number of MMPDEs, known as MMPDE2, MMPDE3, ..., MMPDE6, derived in [Huang et al., 1994b], shown to deviate only slightly from an equidistributed mesh, with an appropriate choice of time-step size. Here, we outline our implementation of the MMPDE method for MMPDE5,

$$\dot{x} = \frac{1}{\gamma} \left(\rho(u) x_\xi\right)_\xi, \text{ for } (\xi, t) \in (0,1) \times (0,T], \tag{3.2a}$$

with boundary conditions

$$x(0,t) = a, \text{ and } x(1,t) = b \text{ for } t \in (0,T], \tag{3.2b}$$

and initial condition

$$x_0(\xi, 0) = \xi, \quad \text{for } \xi \in (0,1). \tag{3.2c}$$

The constants a and b are the endpoints of the resulting mesh, and γ is a user-defined constant that determines how quickly the mesh reacts to changes in ρ. The solution to the MMPDE is a mesh generating function, as defined in Definition 2.4.2.

3.3.2 Discretization of the MMPDE

We solve the nonlinear equation (3.2) using the Galerkin $\mathcal{P}_1$-FEM for the steady part, a fixed-point iterative method to resolve the nonlinearity, and the Crank-Nicolson method [Crank and Nicolson, 1947] to discretize the time derivative.

Using M equal time-steps of length $\tau = 1/M$, we write the semi-discrete weak form of (3.2) as

$$(x^{[j]}, v) + \frac{\tau}{2\gamma} \left(\rho(u^{[j-1]})(x^{[j]})_\xi, v_\xi\right) = (x^{[j-1]}, v) - \frac{\tau}{2\gamma} \left(\rho(u^{[j-1]})(x^{[j-1]})_\xi, v_\xi\right)$$

at each time-step $j = 1, 2, \ldots, M$.

Let $x_h^{[j,k]}$ be the numerical solution to (3.2) computed on a uniform mesh at the k^{th} iteration of the MMPDE method at time-step j, $x_h^{[j-1]}$ be the solution from the previous time-step, and $\Omega^{[c]} = (0,1) \times (0,1]$, be the computational domain. The discrete problem that we solve, for a given k and j is: *find* $x_h^{[j,k]} \in V^h\left(\Omega^{[c]}\right)$, *a finite-dimensional subspace*

of $H^1\left(\Omega^{[c]}\right)$*, such that,*

$$\left(x_h^{[j,k]}, v_h\right) + \frac{\tau}{2\gamma}\left(\rho(u_h^{[j,k-1]})(x_h^{[j,k]})_\xi, (v_h)_\xi\right) = \left(x_h^{[j-1]}, v_h\right) - \frac{\tau}{2\gamma}\left(\rho(u_h^{j-1})(x_h^{[j-1]})_\xi, (v_h)_\xi\right), \quad \text{for all } v_h \in V_0^h\left(\Omega^{[c]}\right). \tag{3.3}$$

Then (3.3) is solved for $k = 1, 2, \ldots,$ at each time-step, $j = 1, 2, \ldots, M$.

3.3.3 Mesh density function ρ

We calculate the mesh density function, ρ, based on local estimates of the error in the evolving numerical solution. Where ρ is larger on the domain we expect the resulting mesh to be finer, thus equidistributing a measure of the error across each mesh interval. Alternatively, ρ, is often based on the gradient of the numerical solution, see for example in [Beckett et al., 2001].

We present two formulations for the discrete form of ρ at each iteration, $\rho_i^{[j,k]}$, for $i = 1, \ldots, N$. In the first version, each $\rho_i^{[j,k]}$ is based on estimates of errors in the L^2-norm, as in [Cao et al., 2001], and is calculated as

$$\rho_i^{[j,k]} = \left(A + \frac{1}{\kappa}\frac{\|E_h\|^2_{0,i^{[j,k]}}}{\|E_h\|^2_{0,\Omega^{[x]}}}\right)^{1/2}\frac{1}{h_i} \text{ for } i = 1, \ldots, N. \tag{3.4}$$

A is a strictly positive parameter and is included to prevent mesh starvation where the error is small and results in a mesh that is approximately uniform in these regions. We set $A = 1$ throughout this chapter.

Our second version of $\rho_i^{[j,k]}$ is based on local error estimates in the energy norm, as in [Jahandari et al., 2020], with,

$$\rho_i^{[j,k]} = \left(A + \frac{1}{\alpha}\|E_h\|^2_{\mathrm{E},i^{[j,k]}}\right)^{1/2}\frac{1}{h_i} \text{ for } i = 1, \ldots, N, \tag{3.5a}$$

and

$$\alpha = \frac{\|E_h\|^2_{\mathrm{E},\Omega^{[x]}}}{N}\frac{1-\beta}{\beta}, \tag{3.5b}$$

where β is a user-defined parameter.

3.3.4 Optimising the mesh generating process

If the numerical solution to the physical problem is changing rapidly, often some $\rho_i^{[j,k]}s$ are also changing rapidly and slow the computation of the mesh [Huang and Russell, 2011, p. 6]. To optimise the mesh generating process, therefore, we perform a number of

"smoothing" steps by recalculating $\rho_i^{[j,k]}$ on each interval, as,

$$\rho_i^{[j,k]} = \begin{cases} \frac{1}{2}\left(\rho_i^{[j,k]} + \rho_{i+1}^{[j,k]}\right) & \text{for } i = 1, \\ \frac{1}{4}\left(\rho_{i-1}^{[j,k]} + 2\rho_i^{[j,k]} + \rho_{i+1}^{[j,k]}\right) & \text{for } i = 2, 3, \ldots N-1, \\ \frac{1}{2}\left(\rho_{i-1}^{[j,k]} + \rho_i^{[j,k]}\right) & \text{for } i = N. \end{cases}$$

The number of smoothing iterations performed depends on the error estimator used and the physical problem being solved on the mesh, and are specified for our specific examples in Sections 3.4.1 and 3.4.2.

3.3.5 Quality of the meshes generated

The MMPDE method builds on the equidistribution idea and is equivalent when the MMPDE is solved exactly. The resulting adapted meshes on $\overline{\Omega}^{[x]}$ have N distinct, non-empty set of intervals, $\mathcal{T}^{[x]} := \{\mathcal{T}_1^{[x]}, \mathcal{T}_2^{[x]}, \ldots, \mathcal{T}_N^{[x]}\}$, and $\overline{\Omega}^{[x]} = \cup_{i=1}^{N} \overline{\mathcal{T}}_i^{[x]}$. When ρ is (perfectly) equidistributed

$$\max_{i=1,\ldots,N} \int_{\overline{\mathcal{T}}_i^{[x]}} \rho(x,t)dx = \frac{1}{N} \int_{\overline{\Omega}^{[x]}} \rho(x,t)dx.$$

However, in practice, we solve a discrete version of the MMPDE and since it is a nonlinear problem, we do so up to a chosen tolerance using a fixed-point iterative process. Therefore, we require a stopping criterion, which verifies that the mesh generated approximately equidistributes ρ. We introduce this by iterating until

$$\max_{i=1,\ldots,N} \int_{\overline{\mathcal{T}}_i^{[x]}} \rho(x,t)dx \text{ for } \leq \frac{\text{TOL}}{N} \int_{\overline{\Omega}^{[x]}} \rho(x,t)dx.$$

for some sufficiently small TOL. That is, at each iteration, we compute

$$\Psi = N \frac{\max_{i=1,\ldots,N} \int_{\overline{\mathcal{T}}_i^{[x]}} \rho(x,t)dx}{\int_{\overline{\Omega}^{[x]}} \rho(x,t)dx},$$

and iterate until $\Psi < \text{TOL}$.

Also, we set a limit for the number of iterations to be performed at each time-step so long as $\Psi < 2$. This improves the efficiency of the method since we observe in practice often the value of Ψ does not change significantly with multiple iterations and the meshes generated resolve the layer regions. The values for "TOL" and "Max iterations" are user defined for each problem and are specified in Sections 3.4.1 and 3.4.2.

3.3.6 Moving the solutions between meshes

Let $u_h^{[j]}$ be the final numerical solution to the SPDEs at time-step j. Since we are investigating time-dependent SPDEs, $u_h^{[j]}$ depends on $u_h^{[j-1]}$. However, $u_h^{[j]}$ is computed on a mesh that is different from the one on which $u_h^{[j-1]}$ was computed. The test problems we

will consider represent pulses or have sharp layers that evolve in time (through processes of advection and diffusion). Therefore, interpolating the solutions between consecutive meshes at all iterations may result in the numerical solution dissipating too quickly.

For example, take u to be

$$u(x) = \exp\left(-\frac{2(x-0.5)^2}{10^{-4}}\right), \tag{3.6}$$

which has a profile that is typical of an advected Gaussian pulse (see example in Section 3.4.1) and letting Π_h denote the piecewise linear interpolant of a function onto the new mesh, Figure 3.1 shows that $\|\Pi_h u\|_\infty$ can be significantly less than $\|u\|_\infty$. Since this underestimation is repeated at every time step, the error accumulates.

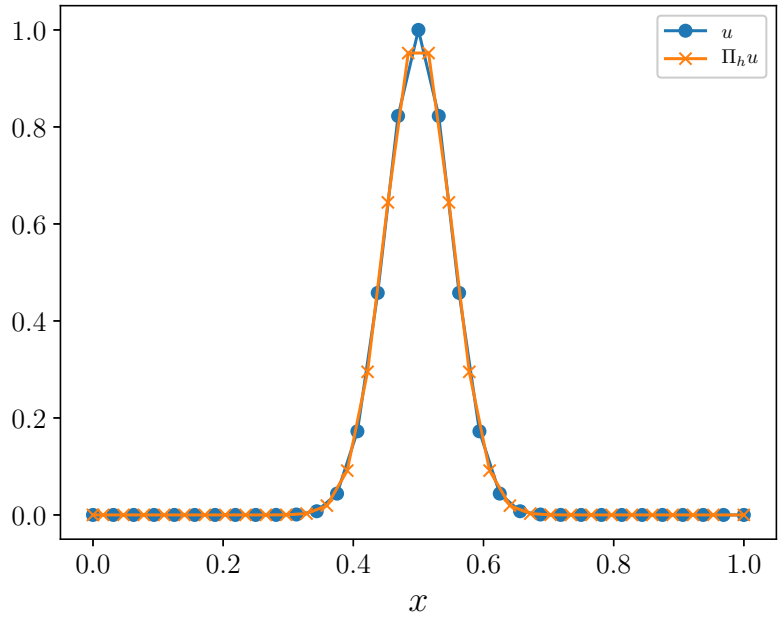


Figure 3.1: Plot of u as given in (3.6) and its interpolant $\Pi_h u$

We investigated various alternatives and found that, in practice, projecting the solution to a $\mathcal{P}_3$-FEM function space and then to the new $\mathcal{P}_1$-FEM function space works best. For example, when calculating the projection onto the $\mathcal{P}_3$-FEM function space, with basis functions $v_{3,h}$, from the $\mathcal{P}_1$-FEM function space, we solve

$$(u_h, v_3) = (\pi_{3,h} u_h, v_3),$$

where $\pi_{3,h}$ is the operator that projects a function onto the $\mathcal{P}_3$-FEM function space. An alternative approach, but which we have not investigated, is the quasi-Lagrange approach, detailed in [Huang and Russell, 2011, §2.6.1], where the mesh points are considered to move continually in time, and therefore removing the necessity to project or interpolate solutions onto a different mesh at each time-step.

3.3.7 Algorithm to generate the meshes

Our algorithm for generating MMPDE meshes for one-dimensional time-dependent problems with interior layers is given in Algorithm 3.1.

Input: N, the number of spatial mesh intervals.
Input: TOL, Max iterations, stopping criteria.
Input: γ, the reaction parameter.
Input: M, the number of time-steps and T, the final time.
Input: u_0, the initial solution to the physical problem.
Set $\omega^{[c]} := \{\xi_0, \xi_1, \ldots, \xi_N\}$ to be a uniform mesh discretizing $\overline{\Omega}^{[c]}$;
Set $x_h^{[0,0]}(\xi) = \xi$ for $\xi \in \overline{\Omega}^{[c]}$;
Set $u_h^{[0]}, u_{2,h}^{[0]} = u_0\left(x_h^{[0,0]}\right)$ in the $\mathcal{P}_1$- and $\mathcal{P}_2$-FEM function spaces, respectively;
Calculate $\rho^{[0]}$ using $u_h^{[0]}$ and $u_{2,h}^{[0]}$ (including smoothing);
$t \leftarrow 0$, $j \leftarrow 1$;
do
  $k \leftarrow 0$;
  do
    $k \leftarrow k+1$;
    Set $x_h^{[j,k]}$ to be the $\mathcal{P}_1$-FEM solution to (3.3);
    Set $\omega^{[j,k]} := x_h^{[j,k]}\left(\omega^{[c]}\right)$, the adapted mesh on $\overline{\Omega}^{[x]}$;
    Set $u_h^{[j,k]}$ and $u_{2,h}^{[j,k]}$ to be the $\mathcal{P}_1$-FEM and $\mathcal{P}_2$-FEM solutions, respectively, to the physical problem, computed on $\omega^{[j,k]}$;
    Calculate $\rho^{[j,k]}$ (including smoothing);
    Calculate $\Psi^{[j,k]}$ (3.3.5);
  while $\Psi^{[j,k]} >$ TOL and $k \leq$ Max iteration;
  $u_h^{[j]} \leftarrow u_h^{[j,k]}$, $u_{2,h}^{[j]} \leftarrow u_{2,h}^{[j,k]}$;
  $j \leftarrow j+1$;
  $t \leftarrow t + T/M$;
while $t \leq T$;

Algorithm 3.1: Generate an adapted mesh for a one-dimensional time-dependent problem

3.4 Numerical results

3.4.1 Time-dependent linear convection-diffusion problem

Our first one-dimensional, time-dependent problem with a solution that features an interior layer, is taken from [Chadha and Madden, 2016]. This linear convection-diffusion problem can be stated as

$$\dot{u} - \varepsilon u'' + \frac{u'}{4} = 0 \text{ for } (x,t) \in \Omega^{[x,t]} := (0,1) \times (0,1], \tag{3.7}$$

where the boundary and initial conditions are chosen so that the exact solution is

$$u = \frac{1}{\sqrt{1+t}} \exp\left(\frac{-(x-(1+t)/4)^2}{4\varepsilon(1+t)}\right). \tag{3.8}$$

When the strictly positive parameter ε is small, i.e., $\varepsilon \ll 1$ the solution exhibits a narrow pulse that propagates through the domain and dissipates. A typical solution is shown in Figure 3.2. The width of the pulse is dependent on the value of ε and a uniform mesh is unsuitable when $\varepsilon \ll 1$ as one observes in Figure 3.3, where oscillations occur in the solution. Moreover, the pulse is not properly resolved.

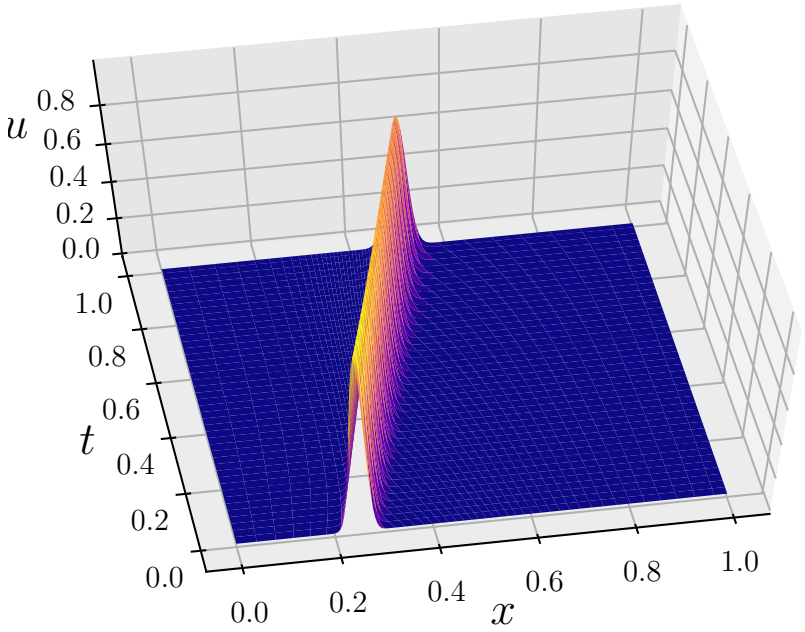


Figure 3.2: Surface plot of (3.8) with $\varepsilon = 10^{-4}$

The meshes, for (3.7), are generated using (3.2) with $\gamma = 0.01$, and $\rho_i^{[j,k]}$ as defined in (3.4) with $\kappa = 1$. We smooth $\rho_i^{[j,k]}$ twice at each iteration.

We write the weak form of the spatial operator in (3.7), at a fixed time-step j, as

$$\mathcal{B}(u^{[j]}, v) := \varepsilon\left((u')^{[j]}, v'\right) + \left(\frac{(u')^{[j]}}{4}, v\right).$$

Taking M time-steps of length $\tau = 1/M$, the semi-discrete problem is

$$\left(u^{[j]}, v\right) + \frac{\tau}{2}\mathcal{B}\left(u^{[j]}, v\right) = \left(u^{[j-1]}, v\right) - \frac{\tau}{2}\mathcal{B}\left(u^{[j-1]}, v\right), \tag{3.9}$$

for $j = 1, 2, \ldots, M$. To get the fully discrete problem, we let $u_h^{[j,k]}$ be the numerical solution to (3.9) computed on the k^{th} mesh generated at time-step j, $u_h^{[j-1]}$ be the solution from the previous time-step, and $\Omega^{[x,t]} = (0,1) \times (0,1]$, be the physical domain. Then we solve

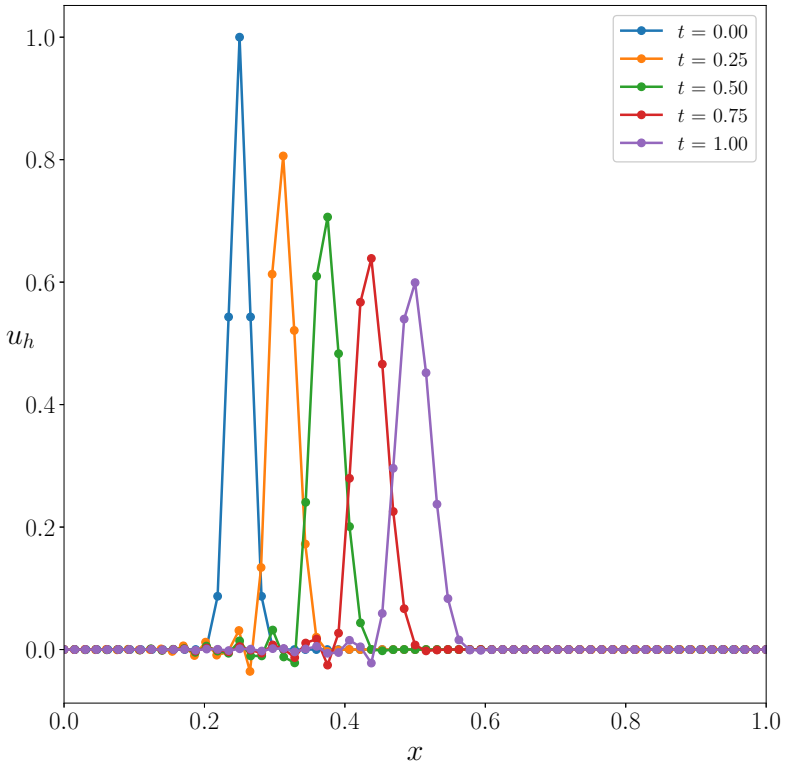


Figure 3.3: Solution to (3.7) with $\varepsilon = 10^{-4}$ and $N = 64$ on a uniform mesh

the problem: *find* $u_h^{[j,k]} \in V^h\left(\Omega^{[x,t_j]}\right)$, *a finite-dimensional subspace of* $H^1\left(\Omega^{[x,t_j]}\right)$, *such that,*

$$\left(u_h^{[j,k]}, v_h\right) + \frac{\tau}{2}\mathcal{B}\left(u_h^{[j,k]}, v_h\right) = \left(u_h^{[j-1]}, v_h\right) - \frac{\tau}{2}\mathcal{B}\left(u_h^{[j-1]}, v_h\right), \quad \text{for all } v_h \in V_0^h\left(\Omega^{[x,t_j]}\right). \tag{3.10}$$

Then (3.10) is solved for $k = 1, 2, \ldots,$ at each time-step, $j = 1, 2, \ldots, M$.

Our numerical solutions for (3.7) are calculated with $T = 1$, and $M = N$. We set TOL $= 1.05$ ("Max iterations" is not required for this example). After the initial couple of time-steps, only one or two iterations are required for the convergence criterion, $\Psi <$ TOL, to be satisfied. Plots of the solution and mesh, with $\varepsilon = 10^{-4}$ and $N = 64$ are shown at a number of time-steps in Figure 3.4. In Figure 3.5, we can see how the mesh evolves over time and that the region where the mesh points are more concentrated follows the movement of the pulse.

In Figure 3.6 we show the final solution on both the physical and computational domains, with $\varepsilon = 10^{-4}$ and $N = 64$, and one observes that no sharp layers exist on the computational domain. This view provides a useful way of thinking of suitable layer-adapted meshes. We can consider a mesh generating function as defining a physical domain $\Omega^{[x]}$: a uniform mesh on the computational domain $\Omega^{[c]}$ is equivalent to the layer-adapted mesh on

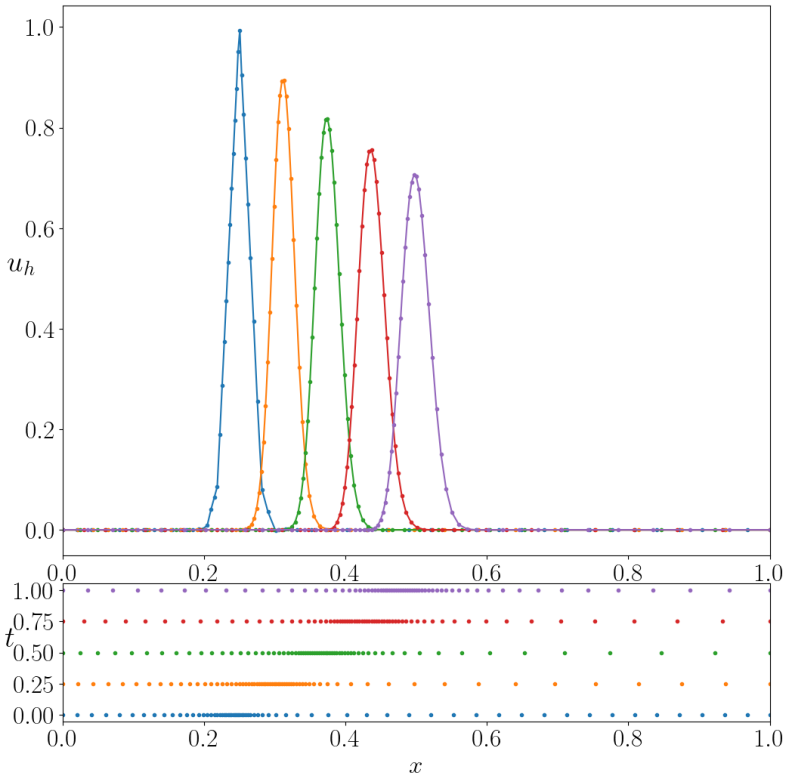


Figure 3.4: Solution to (3.7) with $\varepsilon = 10^{-4}$ and $N = 64$ using Algorithm 3.1

$\Omega^{[x]}$. Then if $u_h^{[M]}(x)$ is transformed from $\Omega^{[x]}$ onto $\Omega^{[c]}$, the layer region will be stretched and no sharp layer is exhibited in $\bar{u}_h^{[M]}(x(\xi))$.

As already discussed, in the chapter we are interested in the performance of the meshing algorithm for certain (fixed) small values of the diffusion parameter, but not arbitrarily small values. That is, we do not focus on parameter robustness *per se*. We present error tables for the final solutions of (3.7) with $\varepsilon = 10^{-3}$. For comparison purposes, we also present equivalent error tables, in Table 3.2 for the final solution to (3.7)solved on uniform meshes. One can see that the magnitude of the errors, when solved on a uniform mesh, is larger than when the problems are solved on MMPDE generated meshes.

Table 3.1: $\|E_h\|_{\mathrm{E}}$ and convergence rates for (3.7) on MMPDE generated meshes

ε/N	32	64	128	256	512
10^{-3}	8.165e-03	3.820e-03	1.828e-03	8.903e-04	4.384e-04
		1.10	1.06	1.04	1.02

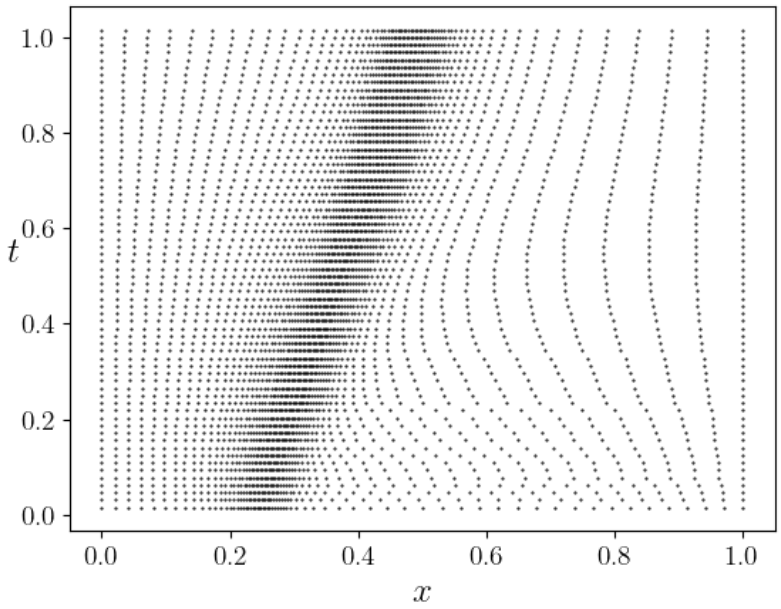


Figure 3.5: Mesh trajectory for to (3.7) with $\varepsilon = 10^{-4}$ and $N = 64$

Table 3.2: $\|E_h\|_{\mathrm{E}}$ and convergence rates for (3.7) on uniform meshes

ε/N	32	64	128	256	512
10^{-3}	1.424e-02	7.260e-03	3.643e-03	1.824e-03	9.129e-04
		0.97	0.99	1.00	1.00

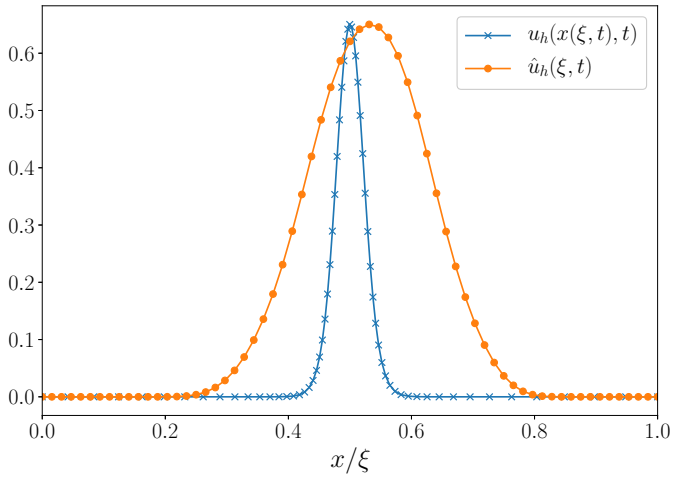


Figure 3.6: Final solution to (3.7) with $\varepsilon = 10^{-4}$ and $N = 64$ on $\Omega^{[x,t]}$, and transposed onto $\Omega^{[c]}$

3.4.2 Burgers' equation

The second one-dimensional time-dependent problem we consider is the nonlinear Burgers' equation. This is a classic test problem used for exhibiting the potential of MMPDEs; see, for example, see [Huang and Russell, 2011, Ch. 1]. It is stated as

$$\dot{u} - \varepsilon u'' + \left(\frac{u^2}{2}\right)' = 0 \quad \text{for } (x,t) \text{ in } \Omega^{[x,t]} := (0,1) \times (0,1], \tag{3.11a}$$

subject to the boundary conditions

$$u(0,t) = u(1,t) = 0, \tag{3.11b}$$

and initial conditions

$$u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \tag{3.11c}$$

When the strictly positive parameter $\varepsilon \ll 1$ the solution is initially smooth but a sharp layer develops as time evolves making uniform meshes unsuitable. For example, in Figure 3.7 we present the solution to (3.11), on a uniform mesh, with $\varepsilon = 10^{-4}$ and $N = 1024$.

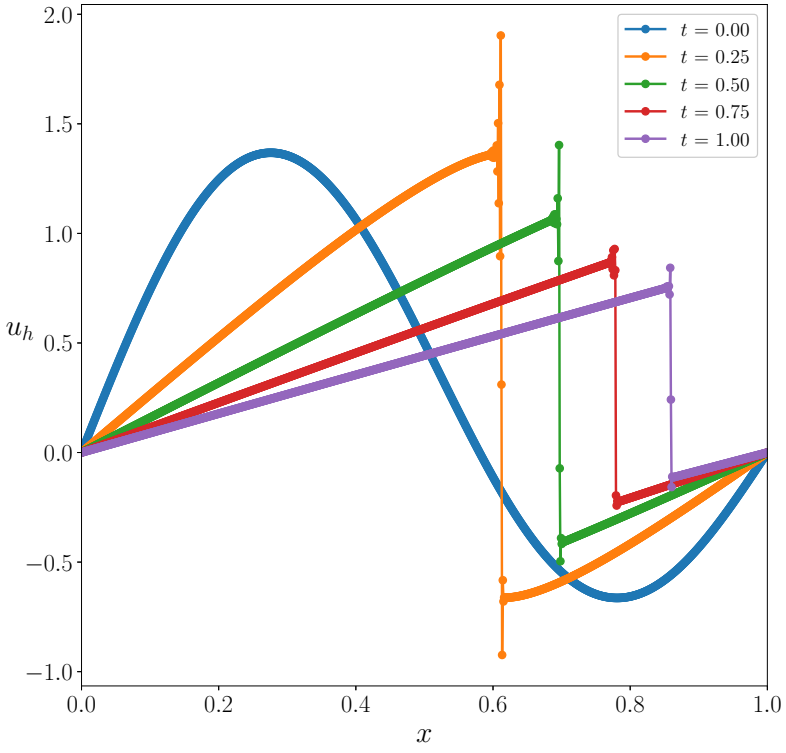


Figure 3.7: Solution to (3.11) with $\varepsilon = 10^{-4}$ and $N = 1024$

Unlike the previous example, we use (modified MMPDE5),

$$\dot{x} = \frac{1}{\gamma\rho}\left(\rho x_\xi\right)_\xi, \text{ for } (\xi,t) \in (0,1) \times (0,1],$$

rather than (MMPDE5) since the results exhibited more robust convergence for larger N for this example. In experiments, we took $M = 32N$, $\gamma = 0.01$, $\rho_i^{[j,k]}$ as defined in (3.5) with $\beta = 0.1$. We smooth $\rho_i^{[j,k]}$ $\lfloor N/5 \rfloor$ times at each iteration. TOL is not used in this example as, in practice, we found that performing one iteration at each time-step, t, was sufficient.

We write the weak form of the spatial operator in (3.11), at a fixed time-step j, as,

$$\mathcal{F}(u^{[j]}, v) := \varepsilon \left((u')^{[j]}, v' \right) + \left(\left(\frac{(u^2)^{[j]}}{2} \right)', v \right).$$

Taking M time-steps of length $\tau = 1/M$, the semi-discrete problem is

$$\left(u^{[j]}, v \right) + \frac{\tau}{2} \mathcal{F} \left(u^{[j]}, v \right) = \left(u^{[j-1]}, v \right) - \frac{\tau}{2} \mathcal{F} \left(u^{[j-1]}, v \right),$$

for $j = 1, 2, \ldots, M$. To get the fully discrete problem, we let $u_h^{[j,k]}$ be the numerical solution to (3.7) computed on the k^{th} mesh generated at time-step j, $u_h^{[j-1]}$ be the solution from the previous time-step, and $\Omega^{[x,t]} = (0,1) \times (0,1]$, be the physical domain. Then we solve the problem, *find* $u_h^{[j,k]} \in V_0^h \left(\Omega^{[x,t_j]} \right)$*, a finite-dimensional subspace of* $H_0^1 \left(\Omega^{[x,t_j]} \right)$*, such that,*

$$\left(u_h^{[j,k]}, v_h \right) + \frac{\tau}{2} \mathcal{F} \left(u_h^{[j,k]}, v_h \right) = \left(u_h^{[j-1]}, v_h \right) - \frac{\tau}{2} \mathcal{F} \left(u_h^{[j-1]}, v_h \right), \quad \text{for all } v_h \in V_0^h \left(\Omega^{[x,t_j]} \right). \tag{3.12}$$

Then (3.12) is solved for $k = 1, 2, \ldots,$ at each time-step, $j = 1, 2, \ldots, M$. This is a nonlinear problem and we solve it using Newton's method.

In Figure 3.8 we present the solution as it evolves when $\varepsilon = 10^{-4}$. Plots of the $\mathcal{P}_1$-FEM solution to (3.11), and the related mesh, with $N = 64$, $\varepsilon = 10^{-4}$ and $T = 1$, at a selection of time-steps, t, are shown in Figure 3.9.

In Figure 3.10, we show the evolution of the mesh over time. One observes that initially when $u_h^{[j]}$ is smooth the mesh remains almost uniform. As the sharp layer develops the mesh smoothly changes and concentrates the mesh points in the layer region. The location of the concentration of mesh points moves with the layer as it propagates through the domain.

In Figure 3.11, we present the numerical solution to (3.11) at $t = 1$, with $\varepsilon = 10^{-4}$ and $N = 64$, as in the previous example one observes that no sharp layers exist when $u_h^{[M]}$ is transposed onto the computational domain $\Omega^{[c]}$. In Table 3.3, we present errors estimates in the energy norm for the final solutions of (3.11) with $\varepsilon = 10^{-3}$ generated using Algorithm 3.1. For comparison purposes, we present Table 3.4, to show the equivalent errors when the solution are generated on uniform meshes. One can see that the magnitude

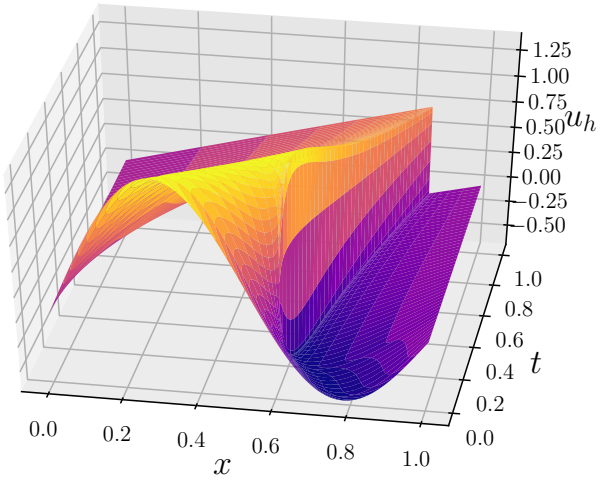


Figure 3.8: Example of the solution to (3.11) with $\varepsilon = 10^{-4}$

of the errors, when solved on a uniform mesh, is larger than when the problems are solved on MMPDE generated meshes.

Table 3.3: $\|E_h\|_{\mathrm{E}}$ and convergence rates for (3.11) on MMPDE generated meshes

ε/N	32	64	128	256	512
10^{-3}	2.998e-01	1.949e-01	8.674e-02	2.195e-02	7.10e-03
		0.62	1.17	1.98	1.63

Table 3.4: $\|E_h\|_{\mathrm{E}}$ and convergence rates for (3.11) on uniform meshes

ε/N	32	64	128	256	512
10^{-3}	2.085e-01	2.244e-01	1.175e-01	5.094e-02	2.57e-02
		-0.11	0.93	1.21	0.99

3.5 Conclusion

In this chapter, we presented meshes that were automatically generated using MMPDEs for one-dimensional, time-dependent problems with interior layers. The general approach could be considered quite classical: the example of the Burgers' equation we have presented here is from [Huang and Russell, 2011, Chap 1]. However, the presentation there focuses on discretization and the concept of an MMPDE, rather than the particular choice of the crucial mesh density function. So here we have presented a viable option, based around an error estimator. We also investigated using mesh PDEs (MPDEs) (formulations with no time-dependency), iteratively at each time-step for (3.8), based around the same error estimator. We found the meshes generated were comparable to those presented in this

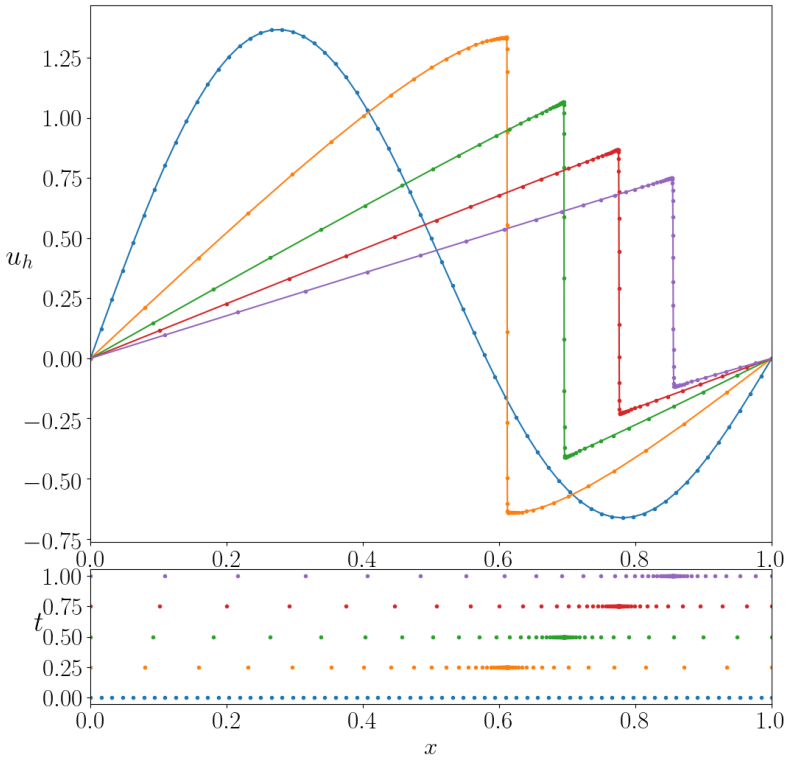


Figure 3.9: Solution to (3.11) with $\varepsilon = 10^{-4}$ and $N = 64$ using Algorithm 3.1

chapter.

This has allowed us to present our approach to MPDEs in the context of a standard reference. In subsequent chapters, we will investigate how to incorporate the underlying technology of mesh PDEs with the well-established corpus of knowledge on layer-adapted meshes for singularly perturbed problems. Initially, these algorithms use *a priori* information and can be understood as generalizations of the classical Bakhvalov meshes.

We then develop a hybrid approach, where *a priori* information on the existence and local of layers, but not their widths or strength, is exploited as the basis for more truly adaptive algorithms.

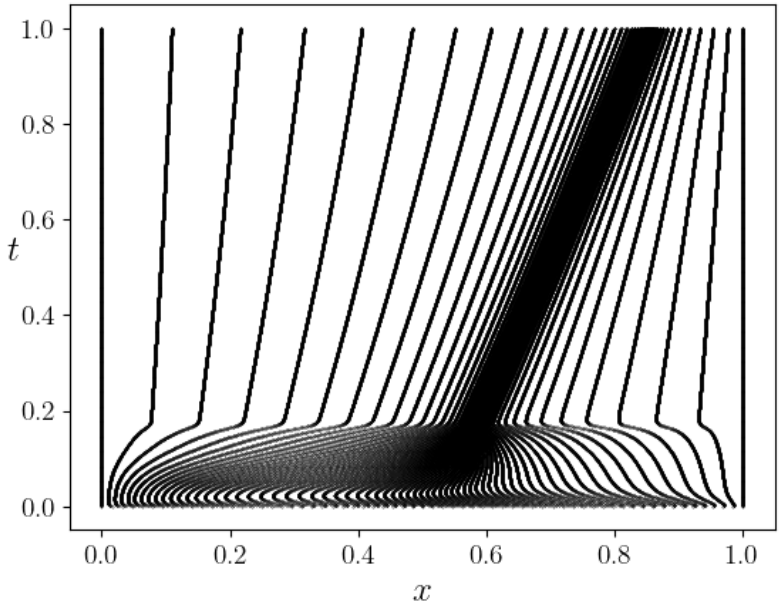


Figure 3.10: Mesh trajectory for to (3.11) with $\varepsilon = 10^{-4}$ and $N = 64$

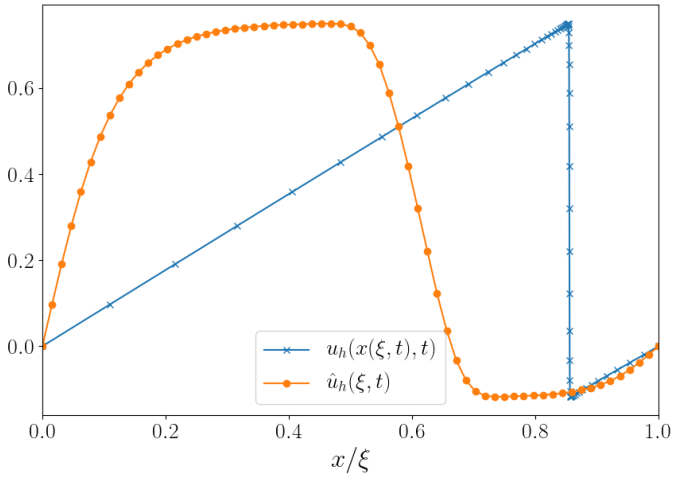


Figure 3.11: Final solution to (3.11) with $\varepsilon = 10^{-4}$ and $N = 64$ on $\Omega^{[x,t]}$ and transposed onto $\Omega^{[c]}$

Chapter 4

Generating one-dimensional Bakhvalov-style meshes using MPDEs

4.1 Introduction

In this chapter, we present a new algorithm for generating layer-adapted meshes for the finite element solution of singularly perturbed differential equations (SPDEs) based on mesh partial differential equations (MPDEs). Specifically, we concentrate on the design and implementation of the methods for one-dimensional problems. This is later extended to two dimensions in Chapter 6.

The meshes we generate are very similar to the celebrated Bakhvalov meshes presented in Section 2.4.3. That is, this new MPDE method gives an alternative algorithm for constructing these meshes. The solutions generated on these meshes are very accurate. However, the MPDE method is more flexible than the classical approach for generating Bakhvalov meshes, which we demonstrate by constructing layer-adapted meshes for systems of SPDEs whose solutions have overlapping layers of different widths.

The contents of Sections 4.2 to 4.4 have been published as part of

> Róisín Hill and Niall Madden. Generating layer-adapted meshes using mesh partial differential equations. Numer. Math. Theor. Meth. Appl., 2021, `https://doi.org/10.4208/nmtma.OA-2020-0187`.

The related FEniCS code is available at `https://osf.io/dpexh`.

4.1.1 Motivation and outline

The novel aspect of this chapter is our proposal to use "Mesh PDEs" (MPDEs) to generate suitable layer-adapted meshes. We present algorithms to generate these meshes and Python code to implement them in FEniCS [Logg et al., 2012b]. In addition, we present numerical results, demonstrating how these algorithms can be applied to one-dimensional reaction-diffusion problems.

This chapter is organised as follows. There are several ways to describe Bakhvalov meshes, two of which are outlined in Section 2.4.3, but in Section 4.2 we introduce a novel approach, based on MPDEs. The efficient solution of these MPDEs is the topic of Section 4.3. Their effectiveness is verified in Section 4.4 for scalar and coupled systems of equations (in the scalar case, with the code presented in Appendix C).

4.2 MPDEs

The core element of this chapter is to present a way of constructing a mesh generating function that is (essentially) equivalent to that described in Section 2.4.3 but expressed as the solution to a differential equation. These "mesh PDEs", introduced by Huang, Ren and Russell [Huang et al., 1994b], are usually considered as a way of performing r-refinement during the iterative *a posterior* refinement of a mesh. Here, however, the PDE features a coefficient that controls the concentration of points in the resulting mesh. Although in one dimension this is equivalent to a well-known equidistribution principle [de Boor, 1974], which we now explain, it allows for generalisation to higher dimensions.

One begins by choosing *a mesh density function*, $\rho : \overline{\Omega} \to \mathbb{R}_{>0}$. A mesh, ω, on $\overline{\Omega}$, is said to "equidistribute" ρ if

$$\int_{x_i}^{x_{i+1}} \rho(x)dx = \frac{1}{N}\int_a^b \rho(x)dx, \text{ for all } i = 0, 1, \ldots, N-1. \tag{4.1}$$

If we choose $\rho = C$, a constant, then the equidistributing mesh is uniform (see Figure 4.1(a)). More typically, ρ is not constant and the equidistributing mesh is finer where ρ is large. For example, if $\rho = 1/\xi^3$, then $x(\xi) = \xi^4$ is a solution to (4.1) and the mesh generated is as shown in Figure 4.1(b).

We can derive an MPDE by considering the equidistribution principle as a mapping $x(\xi) : \overline{\Omega}^{[c]} := [0, 1] \to \overline{\Omega} := [a, b]$ from the computational coordinate ξ to a physical coordinate x, which satisfies

$$\int_a^{x(\xi)} \rho(x)dx = \xi\int_a^b \rho(x)dx. \tag{4.2}$$

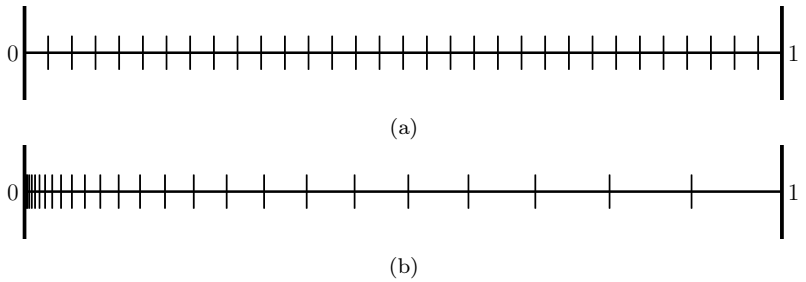


Figure 4.1: Meshes generated when $\varphi(\xi) = \xi$ in Figure 4.1(a) and when $\varphi(\xi) = \xi^4$ in Figure 4.1(b)

First, we differentiate (4.2) with respect to ξ. We can see that $x(\xi)$ satisfies

$$\rho(x)\frac{dx}{d\xi} = \int_a^b \rho(x)dx,$$

where the right-hand side is independent of ξ. Differentiating again with respect to ξ gives the MPDE,

$$(\rho(x)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, \tag{4.3a}$$

We impose the boundary conditions,

$$x(0) = a, \text{ and } x(1) = b, \tag{4.3b}$$

where a and b are the endpoints of the resulting mesh. The solution to (4.3) is a mesh generating function; we solve it numerically using a FEM. Specifically, one computes $x_h(\xi) \in V^h$ a finite-dimensional subspace of $H^1(\Omega)$, such that

$$(\rho(x_h)(x_h)_\xi(\xi), (v_h)_\xi) = (0, v_h), \text{ for all } v_h \in V_0^h, \tag{4.4a}$$

with

$$x_h(0) = a \text{ and } x_h(1) = b. \tag{4.4b}$$

Since (4.4) is a nonlinear problem, we use a fixed-point iterative method to find a numerical solution. At each iteration, k, we solve

$$\int_0^1 \rho\left(x_h^{[k-1]}\right)\left(x_h^{[k]}\right)_\xi (v_h)_\xi d\xi = 0.$$

Therefore, we require a stopping criterion, which verifies that the mesh generated approximately equidistributes ρ. We set $\text{res}^{[k]}$ to be the function on the $\mathcal{P}_1$ space, whose nodal

values are given by

$$\text{res}^{[k]}(\xi_i) = \begin{cases} 0 & i = 0 \\ \int_{\xi_{i-1}}^{\xi_{i+1}} \rho\left(x_h^{[k]}\right)\left(x_h^{[k]}\right)' \psi_i' d\xi, & i = 1, \ldots, N-1 \\ 0 & i = N. \end{cases} \tag{4.5}$$

The iteration terminates when (4.5),

$$\|\text{res}^{[k]}\|_{0,\Omega} = \left(\int_0^1 \left(\text{res}^{[k]}\right)^2 d\xi\right)^{\frac{1}{2}} \leq \text{TOL},$$

where the value of TOL is chosen for each problem (typical values are reported in the relevant sections below).

The construction of the Bakhvalov mesh presented in Section 2.4.3 is essentially that originally proposed in [Bakhvalov, 1969]. However, as indicated in Section 2.4.3, it can also be expressed as the mesh which equidistributes the mesh density function

$$\rho(x) = \max\left\{1, K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta x}{\sigma\varepsilon}\right)\right\}; \tag{4.6}$$

here one can think of ρ as representing point-wise bounds $|u'(x)|$, and, since the error in the FEM is locally proportional to $|u'(x)|$, it is desirable that the mesh equidistributes this quantity so that its contribution over any one element is minimised. Taking ρ as defined in (4.6), the (exact) solution to (4.3) generates a Bakhvalov mesh. Although, as we discuss in Section 4.3, we use the FEM solution to (4.3), we still obtain a mesh that is almost indistinguishable from that constructed using (2.6).

It remains to select K in (4.6), the positive constant that determines the proportion of mesh points that resolve the layer [Linß, 2010]. First note that, when

$$K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta\varphi(\tau)}{\sigma\varepsilon}\right) = 1, \tag{4.7}$$

the transition point, $\varphi(\tau)$, of the mesh generated is the same as in (2.6). Combining (2.6) with (4.7), yields

$$K = \varepsilon q/\big(\beta(q-\tau)\big)$$

However, this is not explicit, since τ is not known. In practice, it is sufficient to determine K using an approximation of τ. We take

$$\tau \approx q - \frac{\sigma\varepsilon(1-q)}{\beta},$$

which is similar to an approximation used in several studies (see [Roos et al., 2008, p.120]),

but comes from ensuring $\varphi(1) = 1$ as $\varepsilon \to 0$. Substituting this into (4.2) gives

$$K = \frac{q}{\sigma(1-q)}.$$

which is what we use in practice.

4.3 Algorithms and implementation

We now discuss how to implement the MPDE approach described in Section 4.2, first directly (Algorithm 4.1) and then with some optimisations with ideas involving both an MPDE and h-refinement (Algorithm 4.2).

Input: N, the number of intervals in the mesh.
Input: ρ, a mesh density function.
Input: TOL.
Input: Max iterations.
Set $\omega^{[c]} := \{\xi_0, \xi_1, \ldots, \xi_N\}$ to be a uniform mesh discretizing $\overline{\Omega}^{[c]}$;
Set $x(\xi) = \xi$ for $\xi \in \overline{\Omega}^{[c]}$;
$s \leftarrow x$;
$k \leftarrow 0$;
do
  set x to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c]}$, to

$$(\rho(s)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, \text{ with } x(0) = a, \text{ and } x(1) = b,$$

  $s \leftarrow x$;
  calculate $\|\text{res}^{[k]}\|_{0,\Omega}$;
  $k \leftarrow k + 1$;
while $\|\text{res}^{[k]}\|_{0,\Omega} >$ TOL and $k \leq$ Max iterations;
Set $\omega := x(\omega^{[c]})$ to be the adapted mesh on $\overline{\Omega}$;

Algorithm 4.1: Generate a layer resolving mesh using an MPDE

In Table 4.1 we report the number of iterations taken by Algorithm 4.1 with TOL = 0.02 and Max iterations = $N/2+5$. The simplicity of the algorithm is appealing, but it is clearly very inefficient. This is because only one mesh point is added to the layer region at each iteration, and so the number of iterations required to achieve convergence is $\mathcal{O}(qN)$, where q is the portion of mesh points used to resolve the layer region.

To speed up convergence, we refine the algorithm to alternate between solving the MPDE and h-refinement. Starting with a uniform mesh with 4 intervals, we apply three iterations of the MPDE, and then a single uniform h-refinement, which doubles the number of mesh intervals. When the desired number of mesh points is reached, we apply the MPDE until convergence is achieved; see Algorithm 4.2 for details, and Appendix C for the FEniCS implementation.

Table 4.1: Number of iterations required by Algorithm 4.1

$\varepsilon \backslash N$	32	64	128	256	512	1024
1	1	1	1	1	1	1
10^{-2}	15	14	15	15	14	13
10^{-4}	21	37	69	132	256	343
10^{-6}	21	37	69	133	261	517
10^{-8}	21	37	69	133	261	517

In Table 4.2 we report the total number of iterations required on the final mesh. Observe that only three or four iterations are ever required on the final mesh. Since (4.8) is first solved on a sequence of grids with $4, 8, 16, \ldots, N/2$, intervals, in fact, $\mathcal{O}(\log(N))$ iterations are applied, but this is still far more efficient than Algorithm 4.1. Further, we can choose a much larger tolerance for Algorithm 4.2: the results in Table 4.2 are computed with TOL $= 0.4$ with no noticeable loss of accuracy (see Section 4.4.1).

4.4 Numerical results

We now present numerical results for our model scalar one-dimensional reaction-diffusion equation in Section 4.4.1, to verify the robustness of numerical solutions computed on the mesh generated by Algorithm 4.2. Then, in Section 4.4.2, we show how the approach extends automatically to a more complex setting: a coupled system of one-dimensional problems whose solutions have multiple overlapping layers.

4.4.1 A scalar reaction-diffusion equation

We start by applying Algorithm 4.2 to generate a mesh on which one can solve

$$-\varepsilon^2 u'' + u = 1 + x, \quad \text{for } x \in (0,1), \quad \text{with } u(0) = u(1) = 0. \tag{4.9}$$

We take standard values of the parameters in (4.6): $\sigma = 2.5$, $\beta = 0.99$, $K = 0.4$, and set TOL $= 0.4$. We calculate the error in the numerical solution as

$$E_h(x) := |u_{2,h}(x) - u_h(x)|,$$

where u_h is the $\mathcal{P}_1$-FEM solution and $u_{2,h}$, our best estimate of the 'true' solution, is the $\mathcal{P}_2$-FEM solution on the same mesh; that is, it is the solution computed using a Galerkin method with quadratic basis functions.

Errors in the numerical solutions to (4.9), computed on the mesh generated by Algorithm 4.2, are shown in Table 4.3. They converge as expected: the $\mathcal{O}(\varepsilon^{1/2}N^{-1})$ term in (2.10) dominates (it is from the H^1-component of the error). The robust convergence is also clearly exhibited in Figure 4.2, which shows the error converges linearly in N and scales like $\varepsilon^{1/2}$.

Input: N, the number of intervals in the final mesh.
Input: ρ, a mesh density function.
Input: TOL.
Set $\omega^{[c;0]} := \{\xi_0, \xi_1, \ldots, \xi_4\}$ to be the uniform mesh on $\overline{\Omega}^{[c]}$ with 4 intervals;
Set $x(\xi) = \xi$ for $\xi \in \overline{\Omega}^{[c]}$;
$s \leftarrow x$;
$k \leftarrow 0$;
for i in 0:$(\log 2(N) - 3)$ **do**
  for j in 1:3 **do**
    set x to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;i]}$, to

$$(\rho(s)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, x(0) = a \text{ and } x(1) = b; \tag{4.8}$$

    $s \leftarrow x$;
  end
  $w^{[c;i+1]} \leftarrow$ uniform h-refinement of $w^{[c;i]}$;
  $s \leftarrow s$ interpolated onto $w^{[c;i+1]}$;
end
do
  set x to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;i]}$, to

$$(\rho(s)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, x(0) = a \text{ and } x(1) = b;$$

  $s \leftarrow x$;
  calculate $\|\text{res}^{[k]}\|_{0,\Omega}$;
  $k \leftarrow k + 1$;
while $\|\text{res}^{[k]}\|_{0,\Omega} > \text{TOL}$;
Set $\omega := x(\omega^{[c;i]})$ to be the adapted mesh on $\overline{\Omega}$.;

Algorithm 4.2: Generate a layer-adapted mesh using an MPDE, and h-refinement.

If one was to compare these with results obtained using a standard Bakhvalov mesh, one would see that they agree to 3 digits for small ε. Interestingly, for moderate values of ε, the results of Algorithm 4.2 out-perform the standard Bakhvalov mesh by about 30%.

4.4.2 Coupled system of reaction-diffusion equations

We now consider how to apply the MPDE approach when solving problems with solutions that contain over-lapping layers. Our test problem is a coupled system of two reaction-diffusion equations, with variable coefficients, taken from [Linß and Madden, 2009]:

$$-\begin{pmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{pmatrix}^2 \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}'' + \begin{pmatrix} 2(x+1)^2 & -(1+x^3) \\ -2\cos(\pi x/4) & (1+\sqrt{2})\exp(1-x) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 2\exp(x) \\ 10x+1 \end{pmatrix}, \text{ for } x \in \Omega := (0,1),' \tag{4.10a}$$

Table 4.2: Number of iterations required using Algorithm 4.2 on the final computational domain

$\varepsilon \backslash N$	32	64	128	256	512	1024
1	1	1	1	1	1	1
10^{-2}	1	1	1	1	1	1
10^{-4}	3	2	2	1	1	1
10^{-6}	4	4	3	3	2	1
10^{-8}	4	4	4	4	3	3

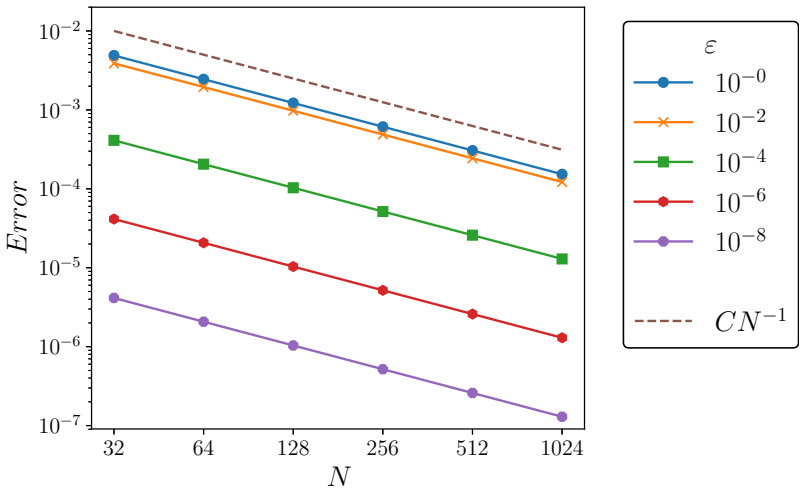


Figure 4.2: Plot of $\|E_h\|_{\mathrm{E}}$ for (4.9) using Algorithm 4.2

with boundary conditions,

$$u_1(0) = u_1(1) = 0, \text{ and } u_2(0) = u_2(1) = 0.$$

The case of interest is when ε_1 and ε_2 are both small but of different orders of magnitude. Without loss of generality, we will take $\varepsilon_1 \ll \varepsilon_2$. While both solution components exhibit layers near $x = 0$ and $x = 1$ that are of width $\mathcal{O}(\varepsilon_2)$, u_1 also has layers of width $\mathcal{O}(\varepsilon_1)$, as can be seen in Figure 4.3. The arguments in [Linß and Madden, 2004, §4.1] can be adapted to show that, if (4.10) is solved on a suitably constructed Bakhvalov mesh, then

$$\|u - u_h\|_{\mathrm{E}} \leq C \left(\left(\varepsilon_1^{1/2} + \varepsilon_2^{1/2} \right) N^{-1} + N^{-2} \right), \tag{4.11}$$

Table 4.3: $\|E_h\|_{\mathrm{E}}$ for (4.9) using Algorithm 4.2

$\varepsilon \backslash N$	32	64	128	256	512	1024
1	4.895e-03	2.448e-03	1.224e-03	6.119e-04	3.060e-04	1.530e-04
10^{-2}	3.887e-03	1.949e-03	9.749e-04	4.875e-04	2.438e-04	1.219e-04
10^{-4}	4.114e-04	2.053e-04	1.031e-04	5.163e-05	2.586e-05	1.294e-05
10^{-6}	4.139e-05	2.071e-05	1.035e-05	5.179e-06	2.590e-06	1.296e-06
10^{-8}	4.141e-06	2.071e-06	1.036e-06	5.181e-07	2.590e-07	1.295e-07

where C is a constant independent of ε_1, ε_2 and N.

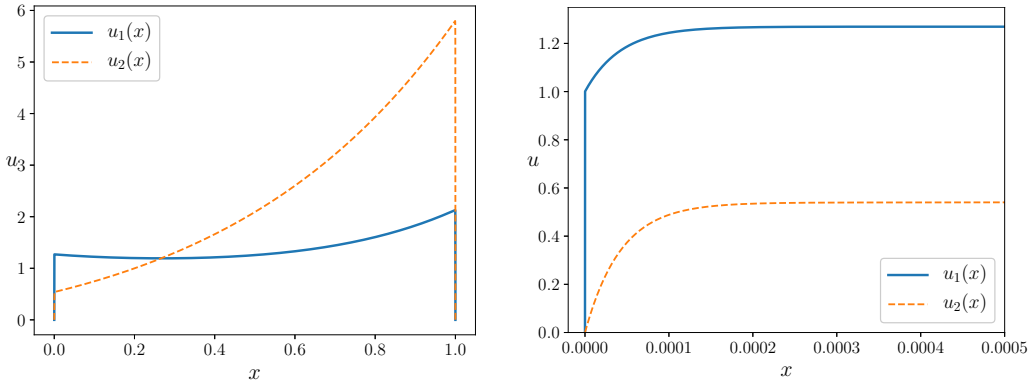


Figure 4.3: FEM solutions to (4.10) with $\varepsilon_1 = 10^{-8}$ and $\varepsilon_2 = 10^{-4}$

To generate a suitable, Bakhvalov-style, mesh for this problem, we solve the MPDE (4.3), using Algorithm 4.2, with

$$\rho(x) = 1 + K\frac{\beta}{\varepsilon_1}\left(\exp\left(-\frac{\beta x}{\sigma\varepsilon_1}\right) + \exp\left(-\frac{\beta(1-x)}{\sigma\varepsilon_1}\right)\right) + K\frac{\beta}{\varepsilon_2}\left(\exp\left(-\frac{\beta x}{\sigma\varepsilon_2}\right) + \exp\left(-\frac{\beta(1-x)}{\sigma\varepsilon_2}\right)\right), \quad (4.12)$$

with $K = 0.1$ and $\sigma = 2.5$. As per standard theory, β is chosen so that β^2 is a lower bound on the row sums of the reaction matrix; we have taken $\beta = 0.99$. Note that (4.12) is a minor variation on (4.6); this form is a little more efficient to implement since one does not have to compute the maximum of multiple terms. Although the resulting mesh is not strictly a Bakhvalov mesh—away from the layer, the mesh is not truly uniform—the difference is imperceptible in finite precision.

We implement Algorithm 4.2, taking TOL $= 0.5$ and performing five iterations of the MPDE before each h-refinement. A typical computed solution, on both physical and computational domains, is shown in Figure 4.4. These verify that the layers are successfully resolved.

The numbers of iterations performed on the final meshes are shown in Table 4.4. The errors in the numerical solution are shown in Figure 4.5, and convergence is consistent with the bound in (4.11).

4.5 Further investigations into one-dimensional MPDE-generated meshes

The remainder of this chapter includes additional analyses and results that were beyond the scope of [Hill and Madden, 2021]. In Section 2.4.3, we presented the Bakhvalov mesh in

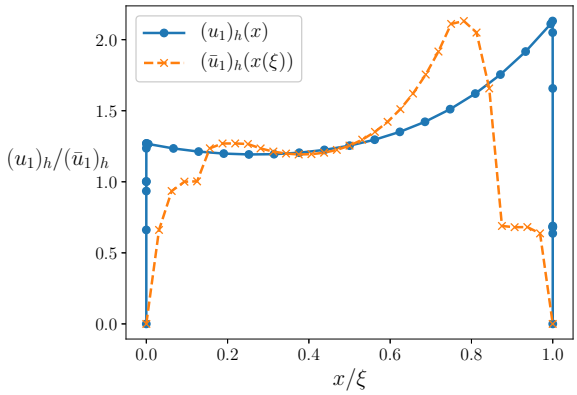


Figure 4.4: FEM solution $(u_1)_h$ to (4.10) with $\varepsilon_1 = 10^{-8}$, $\varepsilon_2 = 10^{-4}$ and $N = 32$, generated on ω, and the solution transformed onto $\omega^{[c]}$

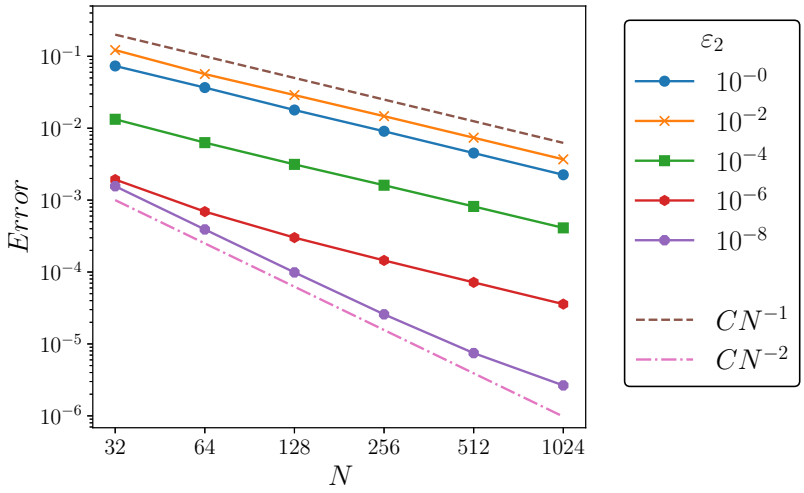


Figure 4.5: Plot of $\|E_h\|_{\mathrm{E}}$ for (4.10) with $\varepsilon_1 = 10^{-8}$

terms of a mesh generating function (2.6) and in Section 4.2 as a solution to an MPDE (4.3) In Section 4.5.1 we examine this further and demonstrate that a Bakhvalov mesh is, indeed, a solution to the continuous MPDE. In Section 4.5.2 we examine how well the MPDE method equidistributes ρ for a one-dimensional scalar reaction-diffusion equation. Finally, in Section 4.5.3 we evaluate a mesh characterization function for meshes generated for scalar reaction-diffusion equations. In particular, we show that solutions calculated on the Bakhvalov-style meshes generated using the MPDE method are comparable with those calculated on classic Bakhvalov meshes for these equations.

4.5.1 Is a Bakhvalov mesh a solution to the MPDE?

In Section 4.2 we pose the MPDE,

$$(\rho(x)x_\xi(\xi))_\xi = 0, \tag{4.13}$$

Table 4.4: Numbers of iterations for (4.10) with $\varepsilon_1 = 10^{-8}$, on the final computational domain

$\varepsilon_2 \backslash N$	32	64	128	256	512	1024
1	1	4	1	1	1	2
10^{-2}	3	2	2	1	1	1
10^{-4}	3	2	1	1	1	1
10^{-6}	3	3	3	2	1	1
10^{-8}	4	2	2	2	2	1

with appropriate boundary conditions and state that "taking ρ as defined in (4.6), the (exact) solution to (4.13) is a Bakhvalov mesh". To verify that this is indeed the case, we need to show that

$$(\rho(\varphi)\varphi_\xi(\xi))_\xi = 0,$$

when

$$\rho(x) = \max\left\{1, K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta x}{\sigma\varepsilon}\right)\right\}. \tag{4.14}$$

In the layer region

$$\rho(x) = K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta x}{\sigma\varepsilon}\right),$$

where $x = \varphi(\xi)$. In this region the mesh generating function of the Bakhvalov mesh is,

$$\varphi(\xi) = -\frac{\sigma\varepsilon}{\beta}\ln\left(1 - \frac{\xi}{q}\right).$$

Therefore,

$$\varphi_\xi = \frac{\sigma\varepsilon}{\beta}\frac{1}{q-\xi},$$

$$\rho(\varphi) = K\frac{\beta}{\varepsilon}\exp\left(\frac{\beta}{\sigma\varepsilon}\frac{\sigma\varepsilon}{\beta}\ln\left(1-\frac{\xi}{q}\right)\right) = K\frac{\beta}{\varepsilon}\left(\frac{q-\xi}{q}\right),$$

and

$$\rho(\varphi)\varphi_\xi = K\frac{\beta}{\varepsilon}\left(\frac{q-\xi}{q}\right)\frac{\sigma\varepsilon}{\beta}\frac{1}{q-\xi} = K\frac{\sigma}{q}.$$

Consequently,

$$(\rho(\varphi)\varphi_\xi)_\xi = 0,$$

in the layer region.

Outside the layer region

$$\rho(x) = 1,$$

and

$$\varphi(\xi) = -\frac{\sigma\varepsilon}{\beta}\ln\left(1-\frac{\tau}{q}\right) + (\xi-\tau)\varphi'(\tau)$$

$$= -\frac{\sigma\varepsilon}{\beta}\ln\left(1-\frac{\tau}{q}\right) + (\xi-\tau)\frac{\sigma\varepsilon}{\beta}\frac{1}{q-\tau}$$
$$= -\frac{\sigma\varepsilon}{\beta}\left(\ln\left(\frac{q-\tau}{q}\right) - \frac{\xi-\tau}{q-\tau}\right).$$

Then,

$$\varphi_\xi = \frac{\sigma\varepsilon}{\beta(q-\tau)},$$

and

$$\rho(\varphi)\varphi_\xi = \frac{\sigma\varepsilon}{\beta(q-\tau)}.$$

So, again

$$(\rho(\varphi)\varphi_\xi)_\xi = 0,$$

outside the layer region too. Hence, the Bakhvalov mesh is, indeed, a solution to the MPDE (4.13).

4.5.2 How well is ρ equidistributed?

In Sections 2.4.3 and 2.4.3, respectively, we discussed that there are two different ways of defining a Bakhvalov mesh; as originally defined by Bakhvalov, and the equidistribution idea, popularised by Linß [Linß, 2010]. We use an MPDE approach, which builds on the equidistribution idea. Although these are equivalent, when the MPDE is solved exactly, as we saw in Section 4.5.1, in practice we solve a discretized MPDE. Furthermore, since it is non-linear, the MPDE is solved only up to some chosen tolerance. Therefore, since the resulting mesh does not exactly equidistribute ρ, we investigate how significant the difference is.

Specifically, we consider the meshes generated to solve the one-dimensional reaction-diffusion equation (2.2), whose solution has one layer, near $x = 0$. The meshes were generated by solving (4.13), with ρ as defined in (4.14).

We know the transition point, x_t, between the graded and uniform meshes, satisfies

$$K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta x_t}{\sigma\varepsilon}\right) = 1. \tag{4.15}$$

By calculating the natural log of (4.15) and rearranging it, we find

$$x_t = -\frac{\sigma\varepsilon}{\beta}\log\left(\frac{\varepsilon}{K\beta}\right).$$

When generating a mesh for (2.2),

$$\rho = \begin{cases} K\dfrac{\beta}{\varepsilon}\exp\left(-\dfrac{\beta x}{\sigma\varepsilon}\right) & \text{for } 0 \le x < x_t, \\ 1 & \text{for } x_t \le x \le 1. \end{cases}$$

So,

$$\int_0^1 \rho(x)dx = \begin{cases} -K\sigma \exp\left(-\frac{\beta x}{\sigma\varepsilon}\right)\Big|_0^{x_t} & \text{for } 0 \leq x < x_t, \\ x\Big|_{x_t}^1 & \text{for } x_t \leq x \leq 1. \end{cases}$$

To check how well ρ is equidistributed, we determine the value of Ψ, for which

$$\max\left\{\int_{x_{i-1}}^{x_i} \rho(x)dx \text{ for } i = 1, \ldots, N\right\} = \frac{\Psi}{N}\int_0^1 \rho(x)dx. \tag{4.16}$$

We present values of Ψ for a range of ε and N in Table 4.5, which confirm that the solutions to (4.13) approximately equidistribute ρ. We provide the equivalent values for Bakhvalov meshes generated using the classically constructed method Section 2.4.3, in Table 4.6, which show that the classic Bakhvalov meshes equidistribute ρ for smaller values of ε. However, our analysis of errors in our numerical solutions shows that solutions calculated on Bakhvalov-style meshes generated using MPDEs are comparable to those generated on Bakhvalov meshes. Linß states, in [Linß, 2010, §4.2.4], when generating an adapted mesh using the relaxed equidistribution principle, i.e., when $\Psi > 1$ in (4.16), that "values used in various publications are 2 and 1.2". As one sees in Table 4.5, the meshes we compute yield a value of Ψ within this range.

Table 4.5: Values of Ψ that satisfy (4.16) using the MPDE method (exact integration)

ε/N	32	64	128	256	512	1024
1	1.00	1.00	1.00	1.00	1.00	1.00
10^{-1}	1.00	1.00	1.00	1.00	1.00	1.00
10^{-2}	1.08	1.05	1.02	1.01	1.00	1.00
10^{-3}	1.31	1.53	1.47	1.12	1.00	1.03
10^{-4}	1.19	1.50	1.29	1.64	1.40	1.23
10^{-5}	1.09	1.35	1.24	1.44	1.73	1.47
10^{-6}	1.14	1.12	1.41	1.25	1.44	1.71
10^{-7}	1.26	1.28	1.27	1.20	1.48	1.80
10^{-8}	1.29	1.33	1.34	1.31	1.76	1.44

4.5.3 Evaluating a mesh characterisation function for MPDE meshes

With over 200 citations, [Linß, 2010] is one of the key monographs in the field of the numerical analysis of singularly perturbed problems. It presents a novel and unifying theory for layer-adapted meshes. Rather than analysing each layer-adapted mesh separately, it proposes a *mesh characterization function* for a given problem, and then proves bounds for a given method with respect to that function. For example, in [Linß, 2010, Ch. 6], the

Table 4.6: Values of Ψ that satisfy (4.16) for Bakhvalov meshes generated for (2.2), using the classic construction (exact integration)

ε/N	32	64	128	256	512	1024
1	1.00	1.00	1.00	1.00	1.00	1.00
10^{-1}	1.43	1.43	1.43	1.43	1.43	1.43
10^{-2}	1.06	1.06	1.06	1.06	1.06	1.06
10^{-3}	1.01	1.01	1.01	1.01	1.01	1.01
10^{-4}	1.00	1.00	1.00	1.00	1.00	1.00
10^{-5}	1.00	1.00	1.00	1.00	1.00	1.00
10^{-6}	1.00	1.00	1.00	1.00	1.00	1.00
10^{-7}	1.00	1.00	1.00	1.00	1.00	1.00
10^{-8}	1.00	1.00	1.00	1.00	1.00	1.00

reaction-diffusion equation

$$-\varepsilon^2 u'' + ru = f \text{ in } (0,1), \quad u(0) = u(1) = 0, \tag{4.17}$$

is analysed, where $\varepsilon > 0$, $r \geq \beta^2$ on $[0,1]$ and $\beta > 0$ is a constant. In the case where the solution to (4.17) exhibits a single layer, near $x = 0$, the mesh characterization function is

$$\vartheta_{rd}^{[p]}(\omega) = \max_{i=1,2,\ldots,N} \int_{x_{i-1}}^{x_i} \left\{1 + \varepsilon^{-1}(e^{-\beta x/p\varepsilon})\right\} dx,$$

where p is a positive constant. Then it is proven in [Linß, 2010, Theorem 6.6] that, for the FEM we have applied, the error in the energy norm is bounded as

$$\|u - u_h\|_{\mathrm{E}} \leq C\left(\varepsilon^{1/2} + \vartheta_{rd}^{[2]}(\omega)\right)\vartheta_{rd}^{[2]}(\omega). \tag{4.18}$$

For example, if ω is a uniform mesh then $x_i - x_{i-1} = 1/N$ and

$$\int_{x_{i-1}}^{x_i} \left\{1 + \varepsilon^{-1}(e^{-\beta x/p\varepsilon})\right\} dx = x - \frac{p}{\beta}\left(e^{-\beta x/p\varepsilon}\right)\Big|_{x_{i-1}}^{x_i}$$
$$= \frac{1}{N} - \frac{p}{\beta}\left(e^{-\beta x_i/p\varepsilon} - e^{-\beta x_{i-1}/p\varepsilon}\right). \tag{4.19}$$

The mesh characterization function, $\vartheta_{rd}^{[p]}(\omega)$, i.e., the maximum value of (4.19), occurs when $x_{i-1} = 0$ and $x_i = 1/N$, giving

$$\vartheta_{rd}^{[p]}(\omega) = \frac{1}{N} - \frac{p}{\beta}\left(e^{-\beta/Np\varepsilon} - 1\right) \leq \frac{1}{N} + \frac{p}{\beta}. \tag{4.20}$$

Since p/β is independent of N, no convergence is implied by (4.20) in the case of a uniform mesh.

When ω is a Shishkin mesh, it can be shown that $\vartheta_{rd}^{[p]}(\omega) \leq CN^{-1}\ln N$, and so (4.18)

gives that the rate of convergence in the energy norm is bounded as

$$\|u - u_h\|_{\mathrm{E}} \leq C\left(\varepsilon^{1/2}N^{-1}\ln N + N^{-2}\ln^2 N\right).$$

If ω is a Bakhvalov mesh, then $\vartheta_{rd}^{[p]}(\omega) \leq CN^{-1}$, and errors in the energy norm are bounded as

$$\|u - u_h\|_{\mathrm{E}} < C\left(\varepsilon^{1/2}N^{-1} + N^{-2}\right).$$

(The same result is arrived at by other means in, for example in [Roos, 2006]). An optimal mesh for (4.17), with respect to convergence, equidistributes the function,

$$\rho(x) = \left\{1 + \varepsilon^{-1}(e^{-\beta x/p\varepsilon})\right\},$$

which is similar to the formulation

$$\rho(x) = \max\left\{1, K\frac{\beta}{\varepsilon}e^{-\beta x/\sigma\varepsilon}\right\},$$

with $\sigma > p$.

In this section, we want to verify that $\vartheta_{rd}^{[p]}(\omega) < CN^{-1}$, when ω is a Bakhvalov-style mesh generated using the MPDE method outlined in Chapter 4 for our model reaction-diffusion equation

$$-\varepsilon^2 u'' + u = 1 - x \text{ in } (0,1), \qquad \text{and} \qquad u(0) = u(1) = 0, \tag{4.21}$$

whose solution exhibits one layer near $x = 0$. We calculate

$$C_\vartheta = \max_{i=1,2,\ldots,N} N\int_{x_{i-1}}^{x_i}\left\{1 + \varepsilon^{-1}(e^{-\beta x/p\varepsilon})\right\}dx,$$

and present the results in Table 4.7.

Table 4.7: Values of C_ϑ for (4.21) when solved on an MPDE Bakhvalov-style mesh

$\varepsilon\backslash N$	32	64	128	256	512	1024
1	1.99	2.00	2.00	2.00	2.00	2.00
10^{-2}	4.76	4.78	4.79	4.79	4.80	4.80
10^{-4}	5.00	5.00	5.02	5.03	5.04	5.05
10^{-6}	5.01	5.03	5.04	5.05	5.05	5.05
10^{-8}	5.01	5.03	5.04	5.05	5.05	5.05

For comparison purposes, we include Table 4.8, which shows the values of C_ϑ when (4.21) is solved on a classic Bakhvalov mesh. We can see that the values of C_ϑ generated are of similar magnitude for both mesh types and are approximately 25% smaller for larger

values of ε when generated on the MPDE Bakhvalov-style meshes.

Table 4.8: Values of C_ϑ for (4.21) solved on a classic Bakhvalov mesh

$\varepsilon \backslash N$	32	64	128	256	512	1024
1	1.99	2.00	2.00	2.00	2.00	2.00
10^{-2}	5.64	6.17	6.36	6.52	6.73	6.75
10^{-4}	5.10	5.13	5.19	5.28	5.41	5.60
10^{-6}	5.05	5.05	5.05	5.06	5.06	5.07
10^{-8}	5.05	5.05	5.05	5.05	5.05	5.05

We can see from Table 4.7 that we can choose a constant, C, such that $C_\vartheta < C$ for the range of ε and N presented so that $\vartheta_{rd}^{[p]}(\omega) < CN^{-1}$. It then follows from (4.18) that errors in the solutions to (4.21) when solved on meshes generated using the MPDE method outlined in Chapter 4 satisfy the same bound as given in (4.5.3).

4.6 Conclusion

In this chapter, we have introduced a novel method of generating Bakhvalov-style meshes using an MPDE method. By applying it to a coupled system, we have demonstrated that suitable layer-adapted meshes can be constructed for complicated situations, once explicit (and sharp) pointwise bounds for the solution's derivatives are available. Thus, this approach can be applied to problems for which the conventional Bakhvalov mesh does not in reality generalise.

We have shown, by experiment, that the numerical solutions of the related SPDEs are robust, and the rates of convergence are consistent with the theory for solutions on Bakhvalov meshes.

Also, we have presented additional information about the one-dimensional meshes of Section 4.2, which validate the assertion that the Bakhvalov mesh can be generated by solving exactly an MPDE. We have also demonstrated that any perturbation of the mesh caused by solving the MPDE by an FEM is of little consequence.

In subsequent chapters, we generalise the ideas presented in the chapter. In Chapter 5 we again consider one-dimensional problems, but with a new variation on the MPDE approach that adaptively determines the derivative estimates in an *a posteriori* way. In Chapter 6 we extend the method to generate meshes for two-dimensional SPDEs. We demonstrate that the method is more flexible than the classic method of generating two-dimensional Bakhvalov meshes, in particular, it is not limited to generating tensor-product grids.

Parts of this chapter can also be thought of as a prelude to Section 6.3, where we perform similar investigations for two-dimensional problems. For example, in Section 4.5.2

we have shown that the numerical solutions to (4.13) approximately equidistribute ρ, and in Section 6.3.1 we will extend this to determine if the numerical solutions to the two-dimensional MPDE also approximately equidistribute the mesh density function, $M(\boldsymbol{x})$.

Chapter 5

A posteriori fitted meshes for one-dimensional problems

5.1 Introduction

In Chapter 4, we presented new algorithms for generating layer-adapted meshes for the finite element solution of one-dimensional SPDEs based on the MPDE

$$(\rho(x)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, \text{ with } x(0) = a \text{ and } x(1) = b, \tag{5.1}$$

where a and b are the endpoints of the resulting mesh. The formulation of, the mesh density function, ρ depends on *a priori* information on the physical differential equation. In a typical example, we take

$$\rho(x) = \max\left\{1, K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta x}{\sigma\varepsilon}\right)\right\},$$

for a one-dimensional reaction-diffusion equation whose solution has one boundary layer near $x(0) = 0$. Here ρ can be thought of as representing point-wise bounds $|u'(x)|$.

In this chapter, we use the same MPDE but we develop the formulation of ρ, now $\rho(x, u_h)$, based on derivatives of numerical solutions to the SPDEs solved on the resulting meshes. That is, the algorithm anticipates that there may be boundary layers present, but does not require any of the *a priori* knowledge used to construct fitted meshes, such as the width of these layers, or bounds on the magnitude of the solutions' derivatives at boundaries.

This chapter is organised as follows: In Section 5.2 we discuss the MPDE that we use, specifically showing how ρ is derived. In Section 5.3, we outline our algorithm and its implementation. In Section 5.4 we verify the effectiveness and robustness, for a scalar reaction-diffusion equation in Section 5.4.1 and a coupled system of reaction-diffusion equa-

tions in Section 5.4.2.

5.2 MPDEs

Our MPDE is (5.1) but now ρ is dependent on the numerical solution to the physical differential equation. For a one-dimensional reaction-diffusion problem, posed on the unit interval,

$$\rho(x, u_h) = \max\left\{1, K\left(|u_h'(x_0^+)| \exp\left(-\frac{|u_h'(x_0^+)|x}{\sigma}\right) + |u_h'(x_N^-)| \exp\left(-\frac{|u_h'(x_N^-)|(1-x)}{\sigma}\right)\right)\right\}, \quad (5.2)$$

where K and σ are user-defined parameters. It should be noted that this formulation is independent of whether the solution to the related SPDE has one or two boundary layers or their location. For example, if the solution has only one layer near $x = 0$ then $u_h'(x_0^+) \gg 1$ and the first term in (5.2) will dominate and the second term will be of $\mathcal{O}(1)$ resulting in a mesh that is only graded near $x = 0$.

Since (5.1) is a nonlinear problem we again use a fixed-point iterative method to find a numerical solution. At each iteration k, we calculate

$$\Delta^{[k]} = \left|\frac{\left((u_h^{[k]})'(x_0^+) - (u_h^{[k-1]})'(x_0^+)\right)}{(u_h^{[k]})'(x_0^+)}\right| + \left|\frac{\left((u_h^{[k]})'(x_N^-) - (u_h^{[k-1]})'(x_N^-)\right)}{(u_h^{[k]})'(x_N^-)}\right|.$$

The iterations terminate when $\Delta^{[k]} \leq \text{TOL}$, some user-defined tolerance (typical values are reported in Section 5.4).

5.3 Algorithm and implementation

We now discuss the implementation of the method outlined in Section 5.2. As in Chapter 4, we use a combination of the MPDE and uniform h-refinement to optimise the algorithm.

In contrast to Algorithm 4.2, we need a "good" numerical solution to the SPDE to calculate an approximation of the derivative of the numerical solution that is suitable to drive the adaption. If the initial mesh, $\omega^{[c,0]}$, has 4 or 8 mesh intervals the derivative of the numerical solution does not approximate the derivative of the true solution to the SPDE until the mesh has 16 intervals, even when solving the scalar reaction-diffusion equation (2.2) with $\varepsilon = 10^{-1}$. Therefore, in Algorithm 5.1 $\omega^{[c,0]}$ is a uniform mesh with 16 intervals.

Input: N, the number of intervals in the final mesh.
Input: ρ, a mesh density function.
Input: TOL.
Set $\omega^{[c;0]} := \{\xi_0, \xi_1, \ldots, \xi_{16}\}$ to be the uniform mesh on $\overline{\Omega}^{[c]}$ with 16 intervals;
Set $x(\xi) = \xi$ for $\xi \in \overline{\Omega}^{[c]}$;
Set $(u_{h+}^{[0]})'(x_0) = (u_{h-}^{[0]})'(x_N) = 1$;
$s \leftarrow x$;
$k \leftarrow 1$;
for i in 0:$(\log_2(N)-4)$ **do**
  do
    set x to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;i]}$, to

$$(\rho(s,v)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, x(0) = a \text{ and } x(1) = b;$$

    Set $\omega^{[k]} := x(\omega^{[c;i]})$ to be the adapted mesh on $\overline{\Omega}$ at step k;
    $u_h^{[k]} \leftarrow \mathcal{P}_1$-FEM solution to the SPDE on $\omega^{[k]}$;
    Calculate $\Delta^{[k]}$;
    $s \leftarrow x$;
    $v \leftarrow u_h^{[k]}$;
    $k \leftarrow k+1$;
  while $\Delta^{[k]} > \text{TOL}$;
  if $|w^{[c;i]}| < N+1$ **then**
    $w^{[c;i+1]} \leftarrow$ uniform h-refinement of $w^{[c;i]}$;
    $s \leftarrow s$ interpolated onto $w^{[c;i+1]}$;
  end
end
Set $\omega := x(\omega^{[c;i]})$ to be the (final) adapted mesh on $\overline{\Omega}$.;

Algorithm 5.1: Generate an *a posteriori* layer-adapted mesh using an MPDE h-refinement information.

5.4 Numerical results

Firstly, in Section 5.4.1 we present numerical results for a scalar one-dimensional reaction-diffusion equation to verify the robustness of the numerical solutions computed on the meshes generated by Algorithm 5.1. Then in Section 5.4.2, we verify that the method can construct appropriate meshes for a coupled system of reaction-diffusion equations with overlapping layers of different widths.

We estimate the errors in our numerical solutions using

$$|E_h| = |u_{2,h}(x) - u_h(x)|,$$

where u_h is the $\mathcal{P}_1$-FEM solution, and $u_{2,h}$, our best estimate of the true solution, is the $\mathcal{P}_2$-FEM solution on the same mesh.

5.4.1 Scalar reaction-diffusion equation

As in Chapter 4, we consider the scalar one-dimensional reaction-diffusion equation,

$$\varepsilon^2 u'' + u = 1 - x, \text{ for } x \in \Omega := (0,1), \text{ with } u(0) = u(1) = 0, \tag{5.3}$$

to verify that numerical solutions computed on the mesh generated by Algorithm 5.1 are accurate and robust. To generate the meshes, the MPDE is as defined in (5.1), ρ is as stated in (5.2), $K = 0.4$, $\sigma = 2.5$ and we set TOL $= 10^{-3}$. In Figure 5.2, we see an example of a numerical solution to (5.3) on the physical domain, Ω, and computational domain, $\Omega^{[c]}$, solved on a mesh generated using (5.1).

In Figure 5.1 we show how the derivative of the numerical solution near $x = 0$, when $N = 32$ and $\varepsilon = 10^{-4}$, evolves and compare it with ε^{-1}. It is proven in [Linß, 2010, Theorem 3.35] that the first derivative of a reaction-diffusion problem is bounded by $C\varepsilon^{-1}$. One observes, that when $\omega^{[c,0]}$ has 16 intervals, that after 5 iterations of Algorithm 5.1 $(u_h^{[k]})'(x_0^+)$ approximates the derivative of the exact solution. The set TOL requires more iterations on $\omega^{[c,0]}$ for this example to ensure Algorithm 5.1 is efficient for the range of N and ε investigated. A typical computed solution, on both physical and computational domains, is shown in Figure 5.2. These verify that the layers are successfully resolved.

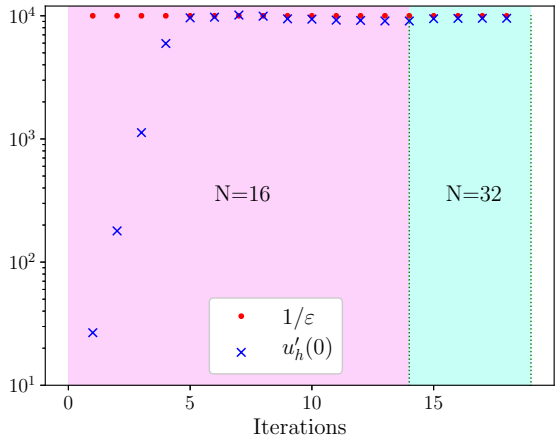


Figure 5.1: Evolution of $(u_{h+})'(x_0)$ for (5.3) with $N = 32$ and $\varepsilon = 10^{-4}$

We present plots of errors in the solutions to (5.3), in the energy norm (1.1), in Figure 5.3. Robust convergence is observed, which is linear in N and scales like $\varepsilon^{-1/2}$. The number of iterations required on the final mesh to achieve the set tolerance are shown in Table 5.1 and are independent of ε and only weakly depend on N.

5.4.2 A coupled system of reaction-diffusion equations

We now consider how to apply the *a posteriori* MPDE approach when solving problems with solutions that contain over-lapping layers. As in Chapter 4, our test problem is a cou-

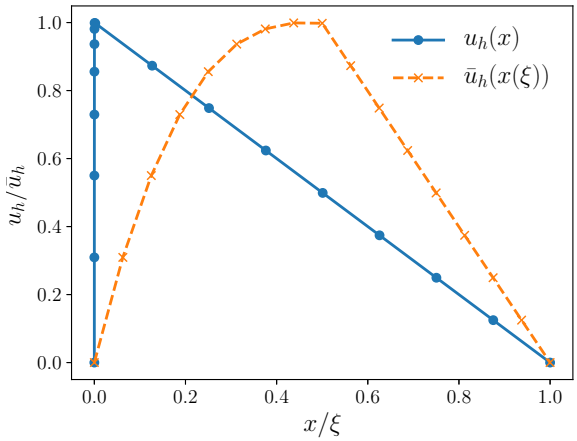


Figure 5.2: Solution of (5.3) on Ω and $\Omega^{[c]}$, with $N = 16$ and $\varepsilon = 10^{-4}$

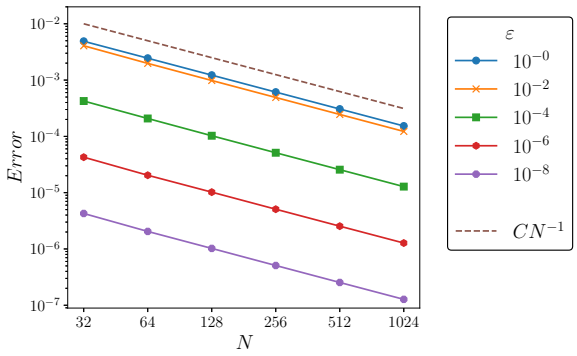


Figure 5.3: Plot of $\|E_h\|_{\mathrm{E}}$ for (5.3) derivative-based stopping criterion

pled system of two reaction-diffusion equations, with variable coefficients, taken from [Linß and Madden, 2009]:

$$-\begin{pmatrix}\varepsilon_1 & 0\\ 0 & \varepsilon_2\end{pmatrix}^2\begin{pmatrix}u_1\\ u_2\end{pmatrix}'' + \boldsymbol{A}\begin{pmatrix}u_1\\ u_2\end{pmatrix} = \boldsymbol{f}, \text{ for } x \in \Omega := (0,1), \tag{5.4a}$$

where

$$\boldsymbol{A} = \begin{pmatrix}a_{1,1} & a_{1,2}\\ a_{2,1} & a_{2,2}\end{pmatrix} = \begin{pmatrix}2(x+1)^2 & -(1+x^3)\\ -2\cos(\pi x/4) & (1+\sqrt{2})\exp(1-x)\end{pmatrix}, \tag{5.4b}$$

Table 5.1: The number of iterations taken to achieve the derivative-based stopping criterion for (5.1)

$\varepsilon\backslash N$	32	64	128	256	512	1024
1	2	2	2	2	2	1
10^{-2}	7	3	2	2	2	2
10^{-4}	4	3	2	2	2	2
10^{-6}	4	3	2	2	2	2
10^{-8}	4	3	2	2	2	2

and

$$\boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} 2\exp(x) \\ 10x + 1 \end{pmatrix}, \tag{5.4c}$$

with boundary conditions,

$$u_1(0) = u_1(1) = 0, \text{ and } u_2(0) = u_2(1) = 0. \tag{5.4d}$$

The case of interest is when ε_1 and ε_2 are both small but of different magnitude. We take $\varepsilon_1 \leq \varepsilon_2$ without loss of generality. The solutions have overlapping layers of different widths near both $x = 0$ and $x = 1$.

To generate suitable layer resolving meshes for (5.4) we solve (5.1) using Algorithm 5.1, with

$$\begin{aligned}
\rho = 1 + K \Bigg(& \frac{\left| \left((u_1)_h^{[k]} \right)' \right| (x_0^+)}{\lambda_{1,0}} \exp\left(- \frac{\left| \left((u_1)_h^{[k]} \right)' \right| (x_0^+) x}{\sigma \lambda_{1,0}} \right) \\
& + \frac{\left| \left((u_2)_h^{[k]} \right)' \right| (x_0^+)}{\lambda_{2,0}} \exp\left(- \frac{\left| \left((u_2)_h^{[k]} \right)' \right| (x_0^+) x}{\sigma \lambda_{2,0}} \right) \\
& + \frac{\left| \left((u_1)_h^{[k]} \right)' \right| (x_N^-)}{\lambda_{1,1}} \exp\left(- \frac{\left| \left((u_1)_h^{[k]} \right)' \right| (x_N^-)(1-x)}{\sigma \lambda_{1,1}} \right) \\
& + \frac{\left| \left((u_2)_h^{[k]} \right)' \right| (x_N^-)}{\lambda_{2,1}} \exp\left(- \frac{\left| \left((u_2)_h^{[k]} \right)' \right| (x_N^-)(1-x)}{\sigma \lambda_{2,1}} \right) \Bigg)
\end{aligned}$$

where

$$\lambda_{1,0} = (a_{2,2}(0) f_1(0) - a_{1,2}(0) f_2(0)) / (a_{1,1}(0) a_{2,2}(0) - a_{1,2}(0) a_{2,1}(0)), \tag{5.5a}$$

$$\lambda_{2,0} = (a_{2,1}(0) f_1(0) - a_{1,1}(0) f_2(0)) / (a_{1,2}(0) a_{2,1}(0) - a_{1,1}(0) a_{2,2}(0)), \tag{5.5b}$$

$$\lambda_{1,1} = (a_{2,2}(1) f_1(1) - a_{1,2}(1) f_2(1)) / (a_{1,1}(1) a_{2,2}(1) - a_{1,2}(1) a_{2,1}(1)), \tag{5.5c}$$

$$\lambda_{2,1} = (a_{2,1}(1) f_1(1) - a_{1,1}(1) f_2(1)) / (a_{1,2}(1) a_{2,1}(1) - a_{1,1}(1) a_{2,2}(1)), \tag{5.5d}$$

$K = 0.2$, $\sigma = 0.25$ and $\text{TOL} = 0.01$. Although, outside the layer region the resulting mesh is not truly uniform—the difference is imperceptible in finite precision. A typical computed solution, on both the physical and computational domains, is shown in Figure 5.4. The numbers of iterations performed on the final meshes are shown in Table 5.2 and are independent of ε and have a weak dependency on N.

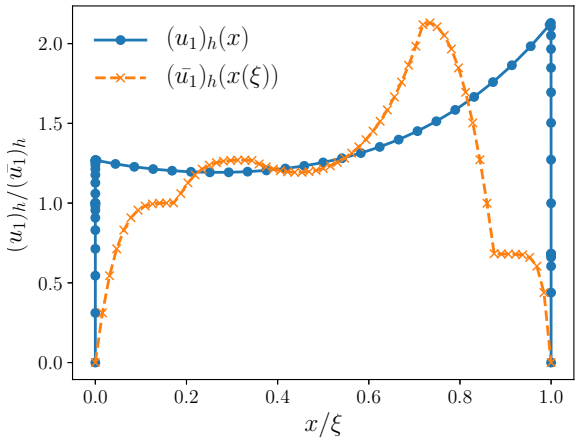


Figure 5.4: FEM solution $(u_1)_h$ to (5.4) with $\varepsilon_1 = 10^{-8}$, $\varepsilon_2 = 10^{-4}$ and $N = 64$, generated on ω, and the solution transformed onto $\omega^{[c]}$

Table 5.2: Number of iterations required to achieve the derivative-based stopping criterion for (5.4) with $\varepsilon_1 = 10^{-8}$

$\varepsilon_2 \backslash N$	32	64	128	256	512	1024
1	65	4	3	2	2	2
10^{-2}	41	14	3	2	2	2
10^{-4}	18	10	3	2	2	2
10^{-6}	11	5	3	2	2	2
10^{-8}	3	5	2	2	2	1

The errors in the numerical solution, measured in the energy norm, are shown in Figure 5.5, and convergence is consistent with the bound in (4.11).

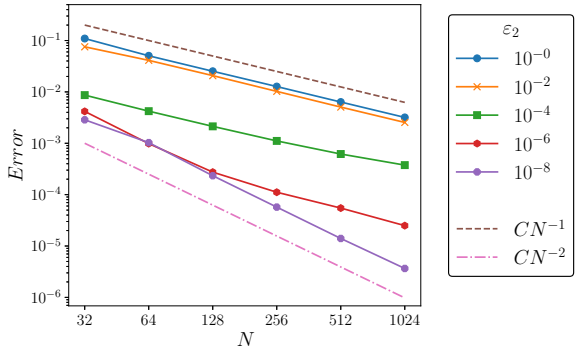


Figure 5.5: Plot of $\|E_h\|_{\mathrm{E}}$ for (5.4) `with` $\varepsilon_1 = 10^{-8}$, using derivative-based stopping criterion

5.5 Conclusion

The MPDEs discussed in this chapter are based on the derivatives of the $\mathcal{P}_1$-FEM solution to the related SPDE at the boundaries. This is consistent with existing literature, for example, in [O'Riordan and Pickett, 2019] it is shown that first-order differences are good

estimates for the derivatives of the true solution.

It would be interesting to study how well higher-order methods behave on MPDE meshes. As an initial investigation, we compared our results with those for a $\mathcal{P}_3$-FEM presented in [Beckett and Mackenzie, 2001b]. Using a test problem presented in Eqn. 24 of that paper, we have observed third-order convergence (in the energy norm) using the MPDE approach and the same discretization, compared with the fourth-order convergence reported in that study. However, these results are quite preliminary, and further tuning is possible.

The *a posteriori* MPDE method detailed in this chapter is only suitable for boundary layer problems. Further work includes amending the algorithm to make it suitable for problems with interior layers too.

Chapter 6

Generating two-dimensional Bakhvalov-style meshes using MPDEs

6.1 Introduction

In Chapter 4, we presented a novel method to generate one-dimensional Bakhvalov-style meshes using the solution to an MPDE. The solutions to the related SPDEs are robust and converge appropriately. The technique described in Section 4.2 demonstrates the applicability of the method to one-dimensional problems. However, one can argue that it presents no advantage over any other method for equidistributing a chosen mesh density function. The same would be true for two-dimensional problems, if we restricted our interest to problems for which tensor-product grids are appropriate. Therefore, in this section, we consider scenarios where tensor-product grids are *not* appropriate.

The contents of Section 6.2 have been published as Section 3 of

> Róisín Hill and Niall Madden. Generating layer-adapted meshes using mesh partial differential equations. Numer. Math. Theor. Meth. Appl., 2021, `https://doi.org/10.4208/nmtma.OA-2020-0187`.

The related FEniCS code is available at `https://osf.io/dpexh`.

In Section 6.2 we consider two-dimensional SPDEs, with a focus on problems for which tensor product grids are not appropriate. In Section 6.2.1 we describe two-dimensional adapted meshes. We present MPDEs and an algorithm for generating these meshes in Section 6.2.2, and numerical results in Section 6.2.3. FEniCS code for generating two-dimensional layer-adapted meshes using an MPDE is in Appendix D. As such, the contributions of this chapter are both *expository* (motivating MPDEs for mesh generation,

and showing how they can be implemented) and *novel* (presenting a new approach for constructing layer-adapted meshes, and for solving MPDEs for SPDEs).

6.2 Two-dimensional problems

Our two test problems are, again, reaction-diffusion problems, and posed on convex quadrilateral domains. Their solutions may exhibit layers near the boundaries of the domain, depending on boundary conditions and the right-hand side of the problem. (Interior layers are also possible, but we do not consider such a scenario). In Section 6.2.2 we introduce generalisations of the MPDE in (4.3) to two distinct two-dimensional settings. The first involves solving a reaction-diffusion problem on a non-rectangular domain, and requires a mesh that is non-uniform in one coordinate direction. So the MPDE is required only for mesh generation in that direction. In the second example, the solution features layers that vary in width throughout the boundary, and a fully two-dimensional MPDE is applied. In Section 6.2.2, we also present an extension of Algorithm 4.2 to two-dimensions.

In Section 6.2.3 we present numerical results when the problems are solved using the $\mathcal{P}_1$ Galerkin FEM on the MPDE-generated meshes of Section 6.2.2. It has been proven that the error estimate in (2.10) holds when this FEM is applied on a Bakhvalov-type triangular mesh for solving reaction-diffusion problems with constant diffusion on a unit square [Zhang and Liu, 2020]. Rigorous error estimates for the problems we present in this section are beyond the scope of this thesis. However, numerical results suggest that the error estimate in [Zhang and Liu, 2020] extends to these cases.

The associated FEniCS code is given in Appendix D.

6.2.1 Layer-adapted M-uniform meshes

Describing an adapted mesh in two-dimensions is somewhat more complicated than in one dimension; in the latter case the mesh connectivity is automatic, and so the solution to the MPDE, for a particular mesh density function, uniquely defines the mesh. In two dimensions a set of points does not lead to a uniquely defined mesh, since many different connectivities are possible. There are several ways this can be addressed; here we consider the simplest approach, that of an "*M-uniform mesh*" [Huang and Russell, 2011, §4.1] (alternatives involve so-called "mesh control" and "function approximation", but are not considered here).

The idea of an M-uniform mesh is that our adapted mesh can be considered as a uniform mesh in some metric space on Ω, on which, $M = M(\boldsymbol{x})$, a matrix-valued ("monitor") function is defined. Roughly, the MPDE is responsible for determining how mesh points move between the uniform and adapted meshes, and the adapted mesh inherits the connectivity of the uniform mesh.

Equipped with this idea, we will investigate how the MPDE-based algorithms for generating one-dimensional layer-adapted meshes can be extended to two-dimensional settings.

6.2.2 Two-dimensional MPDEs and their solution

We present suitable MPDEs for generating two-dimensional layer-adapted meshes for singularly perturbed problems, whose solution exhibits layers. If one requires a two-dimensional mesh that is layer-adapted only in one direction, for example, in the y-direction (see Section 6.2.3), then, a suitable MPDE, for $\boldsymbol{x}(\xi_1, \xi_2) = (x, y)^T$, is

$$(\rho(\boldsymbol{x}(\xi_1, \xi_2))\boldsymbol{x}_{\xi_2}(\xi_1, \xi_2)))_{\xi_2} = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{6.1}$$

with appropriate boundary conditions.

Alternatively, a problem may demand a mesh that is layer-adapted in both directions (see Section 6.2.3). Then we propose that a suitable MPDE, is the two-dimension vector-valued Poisson equation, for $\boldsymbol{x}(\xi_1, \xi_2) = (x, y)^T$,

$$\nabla^{[c]} \cdot (M(\boldsymbol{x}(\xi_1, \xi_2))\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{6.2}$$

with appropriate boundary conditions. This is similar to that proposed in [Huang and Russell, 1998, Eq:(21)].

Remark 6.2.1. It has been noted that a formulation such as in (6.2) may not easily extend to non-convex domains; see, e.g., [Huang and Russell, 2011, §6.1.1]. In that case, a formulation in terms of an inverse coordinate mapping may be more appropriate.

Since (6.2) is the more complicated problem, we discuss how the techniques shown in Algorithm 4.2 can be extended for this problem. Starting with a uniform mesh with four mesh intervals in each direction, we first apply four iterations of the MPDE, and then a single uniform h-refinement which doubles the number of mesh intervals in each direction. This process is repeated until the desired mesh size is reached. Finally, we apply five iterations of the MPDE. (Computational experience has shown this is sufficient; so we do not iterate until a certain tolerance is satisfied).

6.2.3 Numerical results

Reaction-diffusion equation on an irregular domain

Our first example features a reaction-diffusion problem posed on an irregular domain:

$$-\varepsilon^2 \Delta u + u = x, \text{ for } (x, y) \in \Omega, \tag{6.3a}$$

Input: N, the number of intervals in both directions on $\Omega^{[c]}$.
Input: M, a mesh density function.
Set $\omega^{[c;0]} := \{(\xi_1, \xi_2)_i\}_{i=0}^{4}$ to be the uniform mesh on $\overline{\Omega}^{[c]}$ with 5 mesh points in each direction;
Set $\boldsymbol{x}(\xi_1, \xi_2) = (\xi_1, \xi_2)$ for $(\xi_1, \xi_2) \in \overline{\Omega}^{[c]}$;
$\boldsymbol{s} \leftarrow \boldsymbol{x}$;
for i in $0{:}(\log_2(N) - 3)$ **do**
  for j *in* $1:4$ **do**
    set $\boldsymbol{x}$ to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;i]}$, to

$$\nabla^{[c]} \cdot (M(\boldsymbol{s})\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0,0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]},$$

    with boundary conditions as defined in (6.6);
    $\boldsymbol{s} \leftarrow \boldsymbol{x}$;
  end
  $w^{[c;i+1]} \leftarrow$ refine $w^{[c;i]}$;
  $s \leftarrow s$ interpolated onto $w^{[c;i+1]}$;
end
for k *in* $1:5$ **do**
  set $\boldsymbol{x}$ to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;i]}$, to

$$\nabla^{[c]} \cdot (M(\boldsymbol{s})\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0,0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]},$$

  with boundary conditions as defined in (6.6);
  $\boldsymbol{s} \leftarrow \boldsymbol{x}$;
end
Set $\omega := x(\omega^{[c;i]})$ to be the adapted mesh on $\overline{\Omega}$;

Algorithm 6.1: Generate a two-dimensional layer-adapted mesh using an MPDE and h-refinement.

where Ω is the polygon with vertices $(0,0), (0,1), (1,1)$ and $(1,1/2)$, and subject to the boundary conditions

$$u(0,y) = u\left(x, \frac{x}{2}\right) = u(x,1) = 0 \quad \text{and} \quad \frac{\partial u}{\partial x}(1,y) = 0.$$

The solution to (6.3) exhibits boundary layers, whose widths are $\mathcal{O}(\varepsilon)$, near $y = x/2$ and $y = 1$, as shown in Figure 6.1. There are no boundary layers at $x = 0$ because of the right-hand of (6.3) or at $x = 1$ because of the boundary conditions.

A suitable mesh for this is uniform in the x-direction, but layer-adapted in the y-direction. For this scenario, an appropriate MPDE, for $\boldsymbol{x}(\xi_1, \xi_2) = (x, y)^T$, is (6.1), where $\Omega^{[c]}$ is the polygon with vertices $(0,0), (0,1), (1,1)$ and $(1,1/2)$, and apply boundary con-

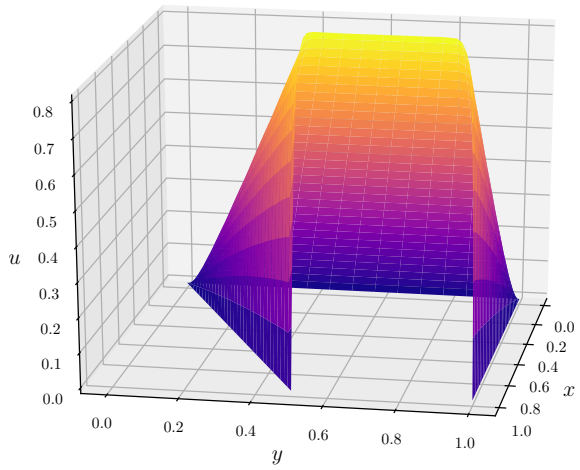


Figure 6.1: FEM solution to (6.4) with $N = 32$ and $\varepsilon = 10^{-2}$

ditions,

$$x(0,\xi_2) = 0, \quad x(1,\xi_2) = 1, \quad x\left(\xi_1, \frac{\xi_1}{2}\right) = 2\xi_2, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1, 1) = 0,$$
$$y\left(\xi_1, \frac{\xi_1}{2}\right) = \xi_1/2, \quad y(\xi_1, 1) = 1, \quad \frac{\partial y}{\partial \mathbf{n}}(0,\xi_2) = 0, \quad \frac{\partial y}{\partial \mathbf{n}}(1,\xi_2) = 0.$$

The boundary conditions on x (for example) ensure that the x-coordinate of the mesh points are fixed at $\xi_1 = 0$ and $\xi_1 = 1$, while allowing them to vary along the boundary $\xi_1 = 1$, yet remaining on the boundary at $\xi_1 = 2\xi_2$. To solve (6.3), we generate a mesh that is concentrated near $y = x/2$ and $y = 1$ using Algorithm 6.1. The initial mesh $\omega^{[c,0]}$ has the mesh points $(\xi_1, \xi_2) = (\bar{\xi}_1, (\bar{\xi}_2(2 - \bar{\xi}_1) + \bar{\xi}_1)/2)$, where $(\bar{\xi}_1, \bar{\xi}_2)$ are the mesh points of a uniform mesh on $[0,1]^2$, with $N = 4$ intervals in each direction. Then the MPDE (6.1) is solved with

$$\rho(\boldsymbol{x}) = \max\left\{1, K\frac{\beta}{\varepsilon}\exp\left(\frac{-\beta(y - x/2)}{\sigma\varepsilon}\right) + K\frac{\beta}{\varepsilon}\exp\left(\frac{-\beta(1-y)}{\sigma\varepsilon}\right)\right\},$$

where $K = 0.28$, $b = 0.99$ and $\sigma = 2.5$. An example of the resulting mesh is shown in Figure 6.2. In Figure 6.3, one observes that no sharp layer exists in the numerical solution when transformed onto the computational domain $\Omega^{[c]}$. Figure 6.4 shows the errors in the numerical solution of (6.3); one observes that they converge linearly in N and scale like $\varepsilon^{1/2}$, as per (2.10).

Reaction-diffusion equation with varying diffusion

We studied problems where diffusion varies on the domain, so the boundary layers vary in width, spatially. Consequently, even when posed on a regular domain (such as a unit square) optimal meshes are not tensor-product in nature. Whereas standard Bakhvalov meshes are tensor-product, our MPDE approach has no such restriction.

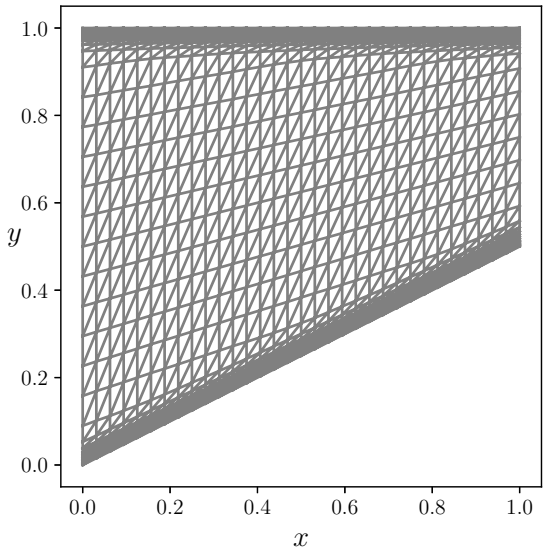


Figure 6.2: Mesh generated using (6.1), for (6.3), with $N = 32$ and $\varepsilon = 10^{-2}$

Our chosen two-dimensional reaction-diffusion equation, with varying diffusion, is

$$-\nabla \cdot \left(\begin{pmatrix} \varepsilon(1+2y)^2 & 0 \\ 0 & \varepsilon(3-2x)^2 \end{pmatrix}^2 \nabla u(x,y) \right) + u(x,y) = (e^x - 1)\,(e^y - 1)\,, \quad \text{for } (x,y) \in \Omega = (0,1)^2, \tag{6.4a}$$

with boundary conditions

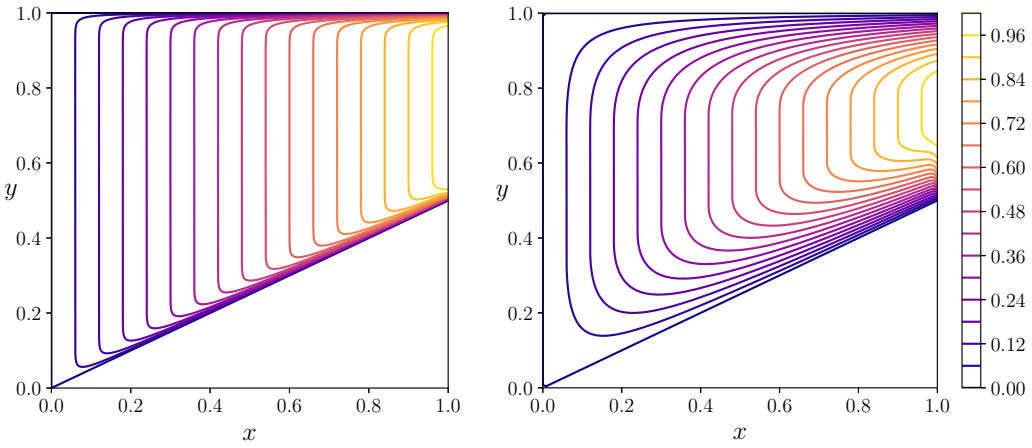
$$u = 0 \text{ on } \partial\Omega. \tag{6.4b}$$

The solution to (6.4) exhibits layers, whose widths vary, near $x = 1$ and $y = 1$, and a corner layer near $(x, y) = (1, 1)$, as can be seen in Figure 6.5(a).

The MPDE mesh for (6.4), is constructed by solving (6.2), on the domain $\Omega^{[c]} = (0,1)^2$, with

$$M(\boldsymbol{x}) = \begin{pmatrix} \max\left\{1, K_1 \dfrac{\beta}{\varepsilon_1} \exp\left(-\dfrac{\beta(1-x)}{\sigma\varepsilon_1}\right)\right\} & 0 \\ 0 & \max\left\{1, K_2 \dfrac{\beta}{\varepsilon_2} \exp\left(-\dfrac{\beta(1-y)}{\sigma\varepsilon_2}\right)\right\} \end{pmatrix}, \tag{6.5}$$

with $\varepsilon_1(x,y) = \varepsilon(1+2y)^2$, $\varepsilon_2(x,y) = \varepsilon(3-2x)^2$, $\beta = 0.99$, $\sigma = 2.5$ and $K_1 = K_2 = 0.4$.



(a) (b)

Figure 6.3: Contour plots of FEM solution to (6.4) with $\varepsilon = 10^{-2}$ and $N = 32$, on Ω in Figure 6.3(a) and on $\Omega^{[c]}$ in Figure 6.3(b)

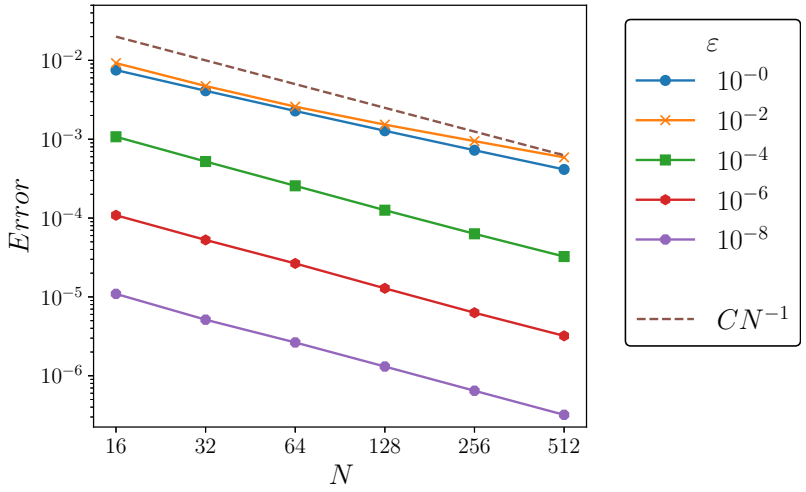


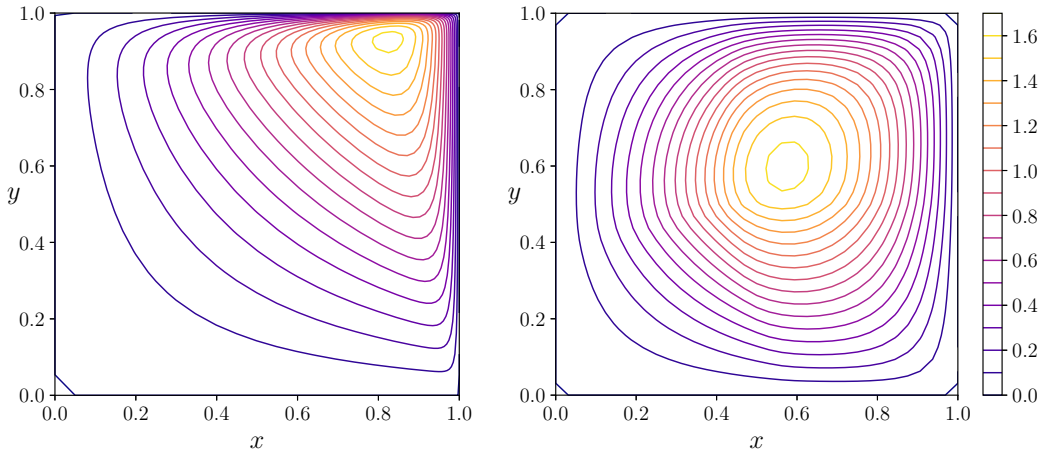
Figure 6.4: Plot of $\|E_h\|_{\mathrm{E}}$ for (6.3) using Algorithm 6.1

The boundary conditions are

$$\begin{aligned} &x(0,\xi_2) = 0, \quad x(1,\xi_2) = 1, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1,0) = 0, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1,1) = 0, \\ &y(\xi_1,0) = 0, \quad y(\xi_1,1) = 1, \quad \frac{\partial y}{\partial \mathbf{n}}(0,\xi_2) = 0, \quad \frac{\partial y}{\partial \mathbf{n}}(1,\xi_2) = 0. \end{aligned} \tag{6.6}$$

That is, the Dirichlet boundary conditions on x (for example) ensure that the x-coordinate of the mesh points are fixed at $\xi_1 = 0$ and $\xi_1 = 1$, while the Neumann (natural) boundary condition allows them to vary along the boundaries at $\xi_2 = 0$ and $\xi_1 = 1$. We then solve (6.2) using Algorithm 6.1. An example of the resulting non tensor-product mesh is shown in Figure 6.6.

The errors in the numerical solutions to (6.4) are shown in Figure 6.7. One observes robust convergence, the $\varepsilon^{1/2}N^{-1}$ term dominates when $N^{-1} < \varepsilon^{1/2}$, otherwise, the N^{-2}



(a) (b)

Figure 6.5: Contour plots of FEM solution to (6.4) with $\varepsilon = 10^{-2}$ and $N = 32$, on Ω in Figure 6.5(a) and on $\Omega^{[c]}$ in Figure 6.5(b)

term dominates.

6.3 Further study of two-dimensional MPDE meshes

This section provides some deeper investigations into the two-dimensional algorithms of [Hill and Madden, 2021].

Following the presentation in Section 4.5, which focused on one-dimensional problems, in Section 6.3.1 we study how well ρ is equidistributed by our mesh generated by the MPDE. This requires some thought since the concept of equidistribution is often considered to be one-dimensional in nature. We present a two-dimensional interpretation, and evaluate how well the two-dimensional MPDE method equidistributes ρ in Section 6.3.1 for three SPDEs.

In Section 6.2 we propose that a non-tensor product grid is preferred to solve problems on irregular domains or whose solutions have layers that vary in width spatially. In order to verify that assertion, in Section 6.3.2 we investigate if a tensor-product Bakhvalov is suitable for (6.4), a reaction-diffusion equation with spatially varying diffusion.

In Section 6.2.2 we claim that performing 5 iterations of the MPDE on the final mesh size is sufficient, so in Section 6.3.3 we present a tolerance based example of generating Bakhvalov-style meshes using the MPDE method for (6.4) and show the method is efficient and the solutions calculated on the meshes are robust and comparable with those presented in Section 6.2.3.

6.3.1 How well is ρ equidistributed?

As discussed in Section 6.2.1, our adapted mesh can be considered as a uniform mesh is some metric space on Ω, on which $M(\boldsymbol{x})$, a matrix-valued function is defined. The layer-

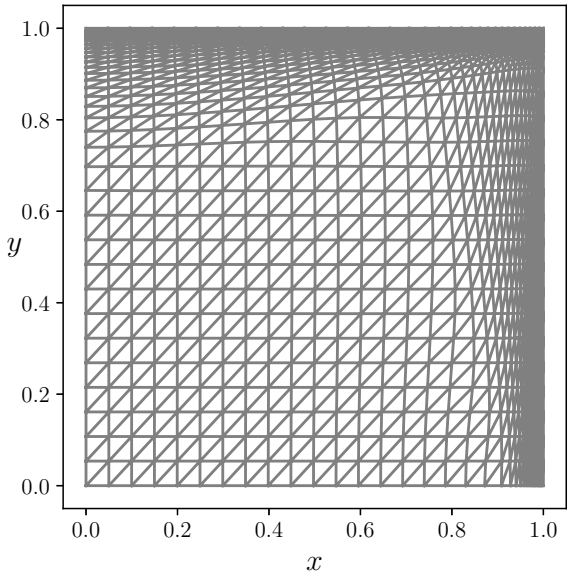


Figure 6.6: Mesh generated using (6.2) for (6.4) with $N = 32$ and $\varepsilon = 10^{-2}$

adapted meshes on $\overline{\Omega}$ have a set of distinct, non-empty simplices (i.e., triangles, in this context), which we denote $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{2N^2}\}$, and $\overline{\Omega} = \cup_{i=1}^{2N^2} \overline{\mathcal{T}_i}$. When $\rho = \sqrt{\det M(\boldsymbol{x})}$ is equidistributed, it is inferred that all mesh simplices in this metric space have constant volume, as detailed in [Huang and Russell, 2011, Ch. 4], i.e.,

$$\int_{\overline{\mathcal{T}_i}} \rho(x,y)dA = \frac{1}{|\mathcal{T}|}\int_{\overline{\Omega}} \rho(x,y)dA, \qquad \text{for } i = 1,2,\ldots,2N^2. \tag{6.7}$$

In practice, (6.7) is not satisfied exactly. Our solution to the MPDE has two sources of error: it is approximated by an FEM, rather than being solved exactly, and the nonlinearity is resolved only to some chosen tolerance. Therefore, we investigate how well ρ is equidistributed for three two-dimensional SPDEs solved on meshes using the MPDE approach outlined in Section 6.2. Specifically, we determine the value of Ψ such that

$$\max_{i=1,2,\ldots,2N^2} \int_{\overline{\mathcal{T}_i}} \rho(x,y)dA \leq \frac{\Psi}{|\mathcal{T}|}\int_{\overline{\Omega}} \rho(x,y)dA.$$

If Ψ is exactly 1, then ρ is perfectly equidistributed. If the computed Ψ is "moderately" larger than 1, and not adversely dependent on N or ε, then we can be confident the mesh is equidistributed sufficiently.

To quantify what might be considered "moderate", recall that in Section 4.5.2 we show that for a one-dimensional SPDE (2.2) the MPDE method approximately equidistributes ρ, and all computed values of Ψ are less than 2 (see Table 4.5). For that investigation, we integrate ρ exactly, but that is not practical for the two-dimensional cases studied here, so we use Gaussian quadrature to estimate the integrals of ρ. However, the effect of quadrature error is not insignificant, and must be quantified. Therefore, in Appendix E, we

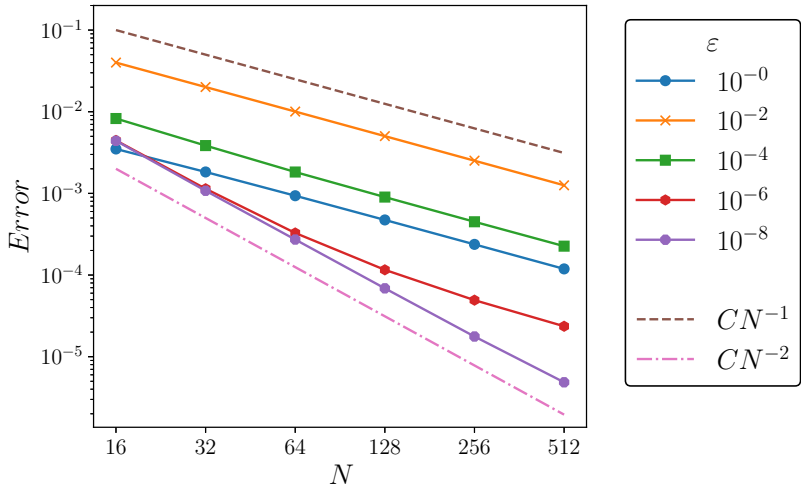


Figure 6.7: Plot of $\|E_h\|_{\mathrm{E}}$ for (6.4) using Algorithm 6.1

have included the values of Ψ for the one-dimensional SPDE (2.2) in Table E.1, where the integrals of ρ are calculated using Gaussian quadrature. Now we observe that the values of Ψ are in the range $1 \leq \Psi \leq 15.26$, indicating that quadrature errors dominate these calculations, but that the process can still be relied on to demonstrate robustness.

In practice, we shall solve

$$\nabla^{[c]} \cdot (M(\boldsymbol{x}(\xi_1, \xi_2)) \nabla^{[c]} \boldsymbol{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{6.8a}$$

with boundary conditions,

$$\begin{aligned} x(0, \xi_2) &= 0, \quad x(1, \xi_2) = 1, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1, 0) = 0, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1, 1) = 0, \\ y(\xi_1, 0) &= 0, \quad y(\xi_1, 1) = 1, \quad \frac{\partial y}{\partial \mathbf{n}}(0, \xi_2) = 0, \quad \frac{\partial y}{\partial \mathbf{n}}(1, \xi_2) = 0. \end{aligned} \tag{6.8b}$$

In (6.8) M is specific to the physical problems being solved, see Sections 6.3.1, 6.3.1 and 6.3.1 for examples. We use Algorithm 6.1 as described in Section 6.2.2. We determine how well the resulting mesh equidistributes ρ by calculating

$$\Psi = \frac{|\mathcal{T}|}{\int_{\overline{\Omega}} \rho(x, y) dA} \max_{i=1,2,\ldots,|\mathcal{T}|} \int_{\overline{\mathcal{T}_i}} \rho(x, y) dA, \tag{6.9}$$

for those examples.

Meshes for an SPDE with one layer

Our first two-dimensional example is the reaction-diffusion equation,

$$-\varepsilon^2 \Delta u + u = 1 - x, \quad \text{for } (x, y) \in \Omega := (0, 1)^2, \tag{6.10}$$

with homogenous Dirichlet boundary conditions at $(0, y)$ and $(1, y)$ and Neumann boundary conditions at $(x, 0)$ and $(x, 1)$. Its solution exhibits one layer near $x = 0$, as shown in Figure 6.8.

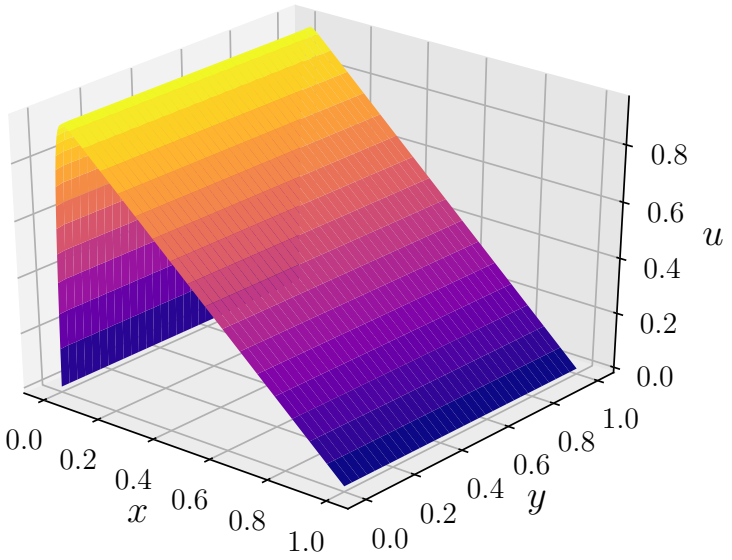


Figure 6.8: Plot of solution to (6.10) with $\varepsilon = 10^{-2}$ and $N = 32$

The mesh density function for (6.10) is

$$M(\boldsymbol{x}) = \begin{pmatrix} 1 + K\dfrac{\beta}{\varepsilon}\exp\left(-\dfrac{\beta x}{\sigma\varepsilon}\right) & 0 \\ 0 & 1 \end{pmatrix}, \tag{6.11}$$

where, following standard theory we take $\beta = 1$ and $\sigma = 2.5$, and we chose $K = 0.4$ based on numerical experience. In Table 6.1, one observes that the values of Ψ compare favourably with those achieved for the one-dimensional case shown in Table E.1 and range from 1.02 to 7.24.

Table 6.1: Values of Ψ for meshes generated using MPDE (6.8) with $M(\boldsymbol{x})$ as in (6.11), when solving (6.10)

ε/N	16	32	64	128	256	512
1	1.02	1.02	1.02	1.02	1.02	1.02
10^{-1}	1.31	1.29	1.29	1.28	1.28	1.28
10^{-2}	3.72	3.66	3.67	3.65	3.64	3.64
10^{-3}	5.34	5.06	5.58	5.37	5.37	5.36
10^{-4}	5.94	5.91	5.95	5.89	6.02	6.05
10^{-5}	6.85	6.53	6.11	6.03	6.04	6.06
10^{-6}	7.24	7.01	6.61	6.28	6.07	6.25

Meshes for an SPDE with two layers and a corner layer

Our second two-dimensional example is the reaction-diffusion equation

$$-\varepsilon^2 \Delta u + u = (1-x)(1-y), \quad \text{for } (x,y) \in \Omega := (0,1)^2, \tag{6.12}$$

with homogenous Dirichlet boundary conditions. Its solution exhibits layers near $x = 0$ and $y = 0$ and a corner layer near $(0,0)$, as shown in Figure 6.9.

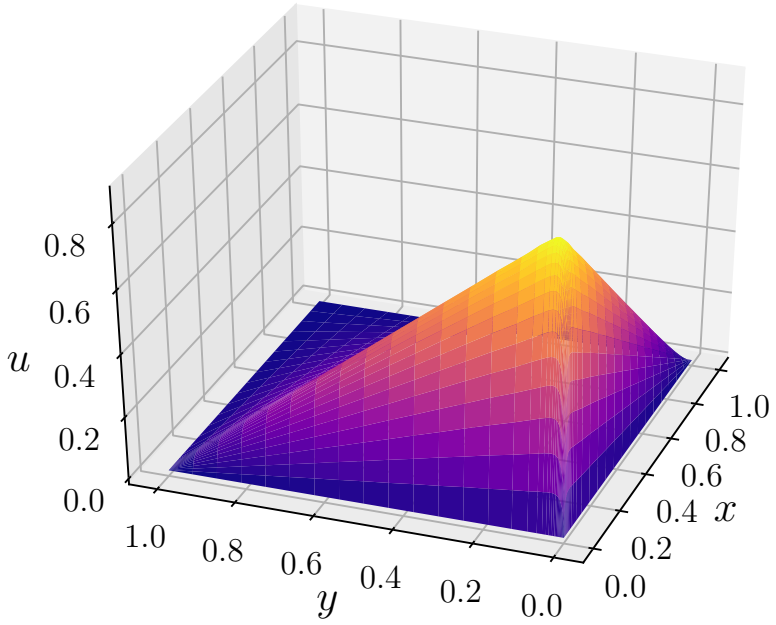


Figure 6.9: Plot of solution to (6.12) with $\varepsilon = 10^{-2}$ and $N = 32$

For (6.12) the MPDE used to generate the meshes is (6.8) with

$$M(\boldsymbol{x}) = \begin{pmatrix} 1 + K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta x}{\sigma\varepsilon}\right) & 0 \\ 0 & 1 + K\frac{\beta}{\varepsilon}\exp\left(-\frac{\beta y}{\sigma\varepsilon}\right) \end{pmatrix}. \tag{6.13}$$

In our calculations we take $\beta = 1$ and $\sigma = 2.5$ (as prescribed by theory), and $K = 0.4$. In Table 6.2, we present the values we calculated for Ψ, and one observes that these values are roughly the square of the values of Ψ given in Table 6.1 for an SPDE with one layer, i.e., the equidistribution discrepancy is the square of the discrepancy when there is one layer. Also, in Figure 6.10 we show $\int_{\overline{\mathcal{T}_i}} \rho dA$, with $\varepsilon = 10^{-2}$ and $N = 32$ and observe that the maximum values occur, just outside the layer regions, on the elements where the mesh transitions between a graded and almost uniform mesh.

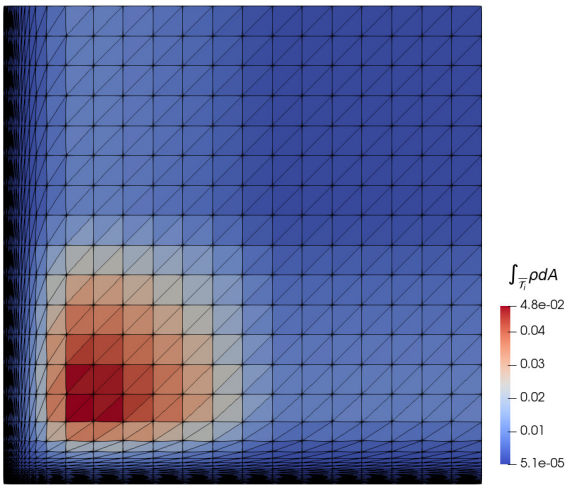


Figure 6.10: Plot of $\int_{\overline{T_i}} \rho dA$ when $N = 32$ and $\varepsilon = 10^{-2}$ for (6.12)

Table 6.2: Values of Ψ for meshes generated using MPDE (6.8) with $M(\boldsymbol{x})$ as in (6.13) when solving (6.12)

ε/N	16	32	64	128	256	512
1	1.05	1.05	1.05	1.05	1.05	1.05
10^{-1}	1.63	1.63	1.64	1.64	1.64	1.64
10^{-2}	12.45	12.61	13.13	13.16	13.19	13.19
10^{-3}	25.02	24.31	28.78	28.36	28.59	28.69
10^{-4}	30.23	31.68	32.89	33.47	34.78	35.39
10^{-5}	32.45	33.40	34.23	34.95	35.55	35.88
10^{-6}	33.06	34.09	34.74	35.22	35.52	35.80

Meshes for (6.4) with spatially varying diffusion

Our final example features spatially varying diffusion:

$$-\nabla\cdot\left(\begin{pmatrix}\varepsilon(1+2y)^2 & 0\\ 0 & \varepsilon(3-2x)^2\end{pmatrix}^2\nabla u(x,y)\right)+u(x,y)=\left(e^x-1\right)\left(e^y-1\right),$$

$$\text{for } (x,y)\in\Omega=(0,1)^2, \quad (6.14a)$$

with boundary conditions

$$u=0 \quad \text{on } \partial\Omega. \quad (6.14b)$$

As shown in Figure 6.11, the solution to (6.14) exhibits two edge layers, near $x = 1$ and $y = 1$, whose widths vary spatially, and also a corner layer.

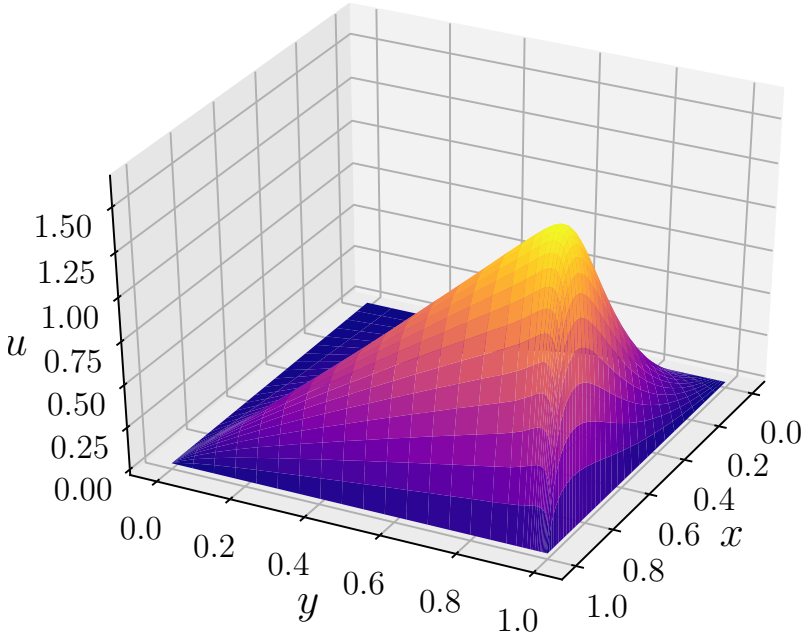


Figure 6.11: Plot of a solution to (6.14) with $\varepsilon = 10^{-2}$ and $N = 32$

We generate suitable meshes for (6.14) using the MPDE in (6.8) with

$$M(\boldsymbol{x}) = \begin{pmatrix} 1 + K_1 \frac{\beta}{\varepsilon_1} \exp\left(-\frac{\beta(1-x)}{\sigma\varepsilon_1}\right) & 0 \\ 0 & 1 + K_2 \frac{\beta}{\varepsilon_2} \exp\left(-\frac{\beta(1-y)}{\sigma\varepsilon_2}\right) \end{pmatrix}, \tag{6.15}$$

where $\varepsilon_1(x, y) = \varepsilon(1 + 2y)^2$, $\varepsilon_2(x, y) = \varepsilon(3 - 2x)^2$ come from the problem data of SPDE (6.14) and $\sigma = 2.5$ depends on the numerical scheme, while $K_1 = K_2 = 0.4$, and $\beta = 0.99$ have been chosen following numerical experimentation. In Table 6.3, we present the values of Ψ calculated and observe that they are similar to the values of Ψ presented in Table 6.2. In Figure 6.12, we show a plot of the values of $\int_{\overline{\mathcal{T}}_i} \rho dA$ and again observe the maximum values of $\int_{\overline{\mathcal{T}}_i} \rho dA$ occur where the mesh transitions between a graded and almost uniform mesh.

Table 6.3: Values of Ψ for meshes generated using (6.8) with $M(\boldsymbol{x})$ as in (6.15), when solving (6.14)

ε/N	16	32	64	128	256	512
1	1.17	1.18	1.18	1.18	1.18	1.18
10^{-1}	1.37	1.39	1.39	1.40	1.40	1.40
10^{-2}	3.50	3.52	3.49	3.48	3.47	3.47
10^{-3}	18.48	19.15	19.81	19.84	19.81	19.79
10^{-4}	27.57	29.79	29.27	28.20	27.55	27.41
10^{-5}	34.03	35.14	33.14	33.81	31.97	31.29
10^{-6}	36.07	40.39	34.19	35.45	35.67	35.35

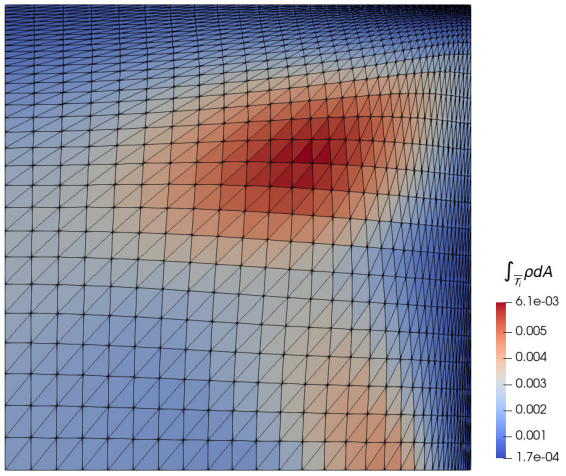


Figure 6.12: Plot of $\int_{\overline{T}_i} \rho dA$ when $N = 32$ and $\varepsilon = 10^{-2}$ for (6.14)

Summary

In Section 4.5.2, we have shown for the one-dimensional case that when we use exact integration that ρ is approximately equidistributed. The evidence from these sections supports the argument that this is also the case for a broad class of two-dimensional SPDEs.

6.3.2 Is a tensor-product Bakhvalov mesh appropriate for (6.14)?

The classical Bakhvalov mesh is a tensor product grid, and is designed with problems that have constant diffusion in mind, and thus for which layer widths may be assumed to be constant. However, in Section 6.2.3 we solve (6.4), a two-dimensional reaction-diffusion equation with spatially varying diffusion and, thus, whose solution exhibits layers that vary in width. While it is intuitive that the non-tensor generated by our MPDE approach is more suitable, it is worth considering if standard tensor-product grids would suffice.

In Section 6.3.2, we present images that visually compare the numerical solutions to (6.14), when generated on tensor-product Bakhvalov meshes, with the numerical solution generated using the MPDE method outlined in Section 6.2.3 for $\varepsilon = 10^{-3}$ and $N = 32$. Also, we present error estimates in the energy norm for $\varepsilon = 10^{-6}$ and $N = 2^j$ for $j = 5, \ldots, 9$.

Tensor-product Bakhvalov meshes

The Bakhvalov meshes are generated using the classic construction method, as described in Section 2.4.3, and the resulting grids, $\omega = \omega_x \times \omega_y$, are tensor-product in nature. The

Bakhvalov meshes can be stated in terms of the mesh generating function,

$$\varphi(t) = \begin{cases} \chi(t) := 1 + \dfrac{\sigma\varepsilon_B}{\beta} \ln\left(1 - \dfrac{1-t}{q}\right), & \text{for } t \in [1-\tau, 1], \\ \pi(t) := \chi(1-\tau) + (t-1+\tau)\varphi'(1-\tau), & \text{elsewhere,} \end{cases} \tag{6.16}$$

where the resulting mesh ω_t has the mesh points $t_i = \varphi(i/N)$ for $i = 0, 1, \ldots, N$. In experiments we take $\sigma = 2.5$, $\beta = 0.99$, $q = 0.5$ (which is consistent with the parameters used in Section 6.2.3). We present results for different values for ε_B, specifically, 10^{-3}, 5×10^{-3}, 9×10^{-3}, the minimum, average and maximum values of the diffusion coefficient in (6.14), respectively.

Results

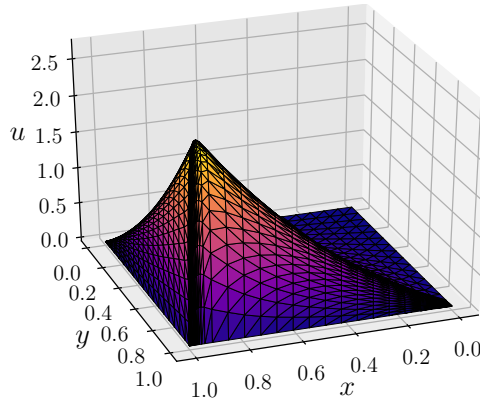
In Figure 6.13 we display numerical solutions to (6.4) with $\varepsilon = 10^{-3}$ and $N = 32$. We show the numerical solution generated using the MPDE method of Section 6.2.3 in Figure 6.13(a), and numerical solutions generated on tensor-product Bakhvalov meshes in Figures 6.13(c) to 6.13(g). Considering all the layer regions one observes that the allocation of mesh points is best in Figure 6.13(a). To affirm this, one observes in Figure 6.13(b) that when the numerical solutions are transformed onto a uniform mesh all the layer regions are stretched, which does not occur in Figures 6.13(d) to 6.13(h).

In Figure 6.14, we compare error estimates in the energy-norm, $\|E_h\|_{\mathrm{E}}$, for (6.14) when the SPDE, with $\varepsilon = 10^{-4}$, is solved on meshes generated using the MPDE method to $\|E_h\|_{\mathrm{E}}$ when the solutions are generated on tensor product Bakhvalov meshes generated using (6.16) with $\varepsilon_B = \varepsilon, 5\varepsilon, 9\varepsilon$. First, note that when the Bakhvalov mesh is based on $\varepsilon_B = \varepsilon$, the order of convergence suffers greatly. For $\varepsilon_B = 5\varepsilon$ and $\varepsilon_B = 9\varepsilon$, first-order convergence is observed on the tensor product meshes, but the results are outperformed by those on the MPDE mesh by factors of 20%, and 40%, respectively.

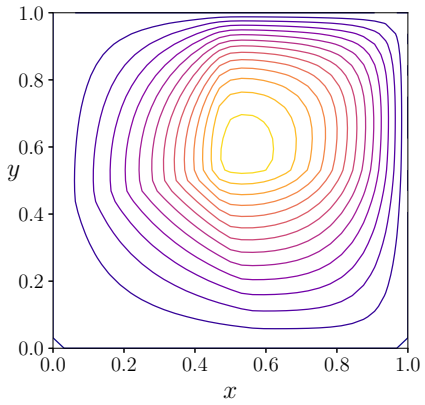
From these experiments, we can conclude that the MPDE meshes yield numerical solutions that are superior to those computed on tensor-product Bakhvalov meshes. Their closest competitor is the one associated with $\varepsilon_B = 5\varepsilon$, but that was found only after careful experimentation, whereas the optimal MPDE mesh was generated automatically; a minor deviation from the choice $\varepsilon_B = 5\varepsilon$ yielded vastly inferior results.

6.3.3 Tolerance based MPDE Bakhvalov-Style meshes

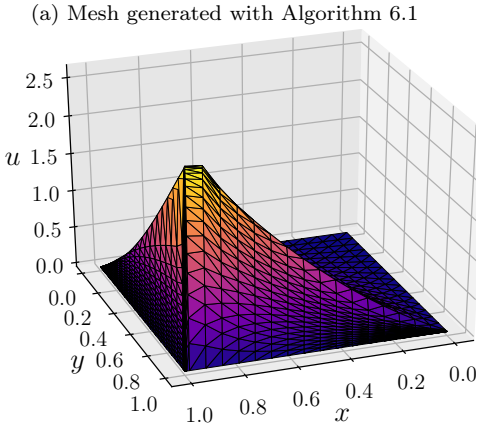
In Section 6.2.2, when generating Bakhvalov-style meshes for two-dimensional problems using an MPDE, we apply five iterations of the MPDE method once the desired mesh size is reached. Computational experience has shown this is sufficient, and that it is not necessary to iterate until a certain tolerance is satisfied. The purpose of this section is to verify that this is indeed the case. Therefore, we now present numerical results for (6.14), a



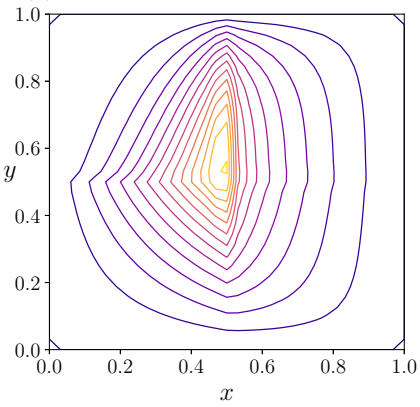
(a) Mesh generated with Algorithm 6.1



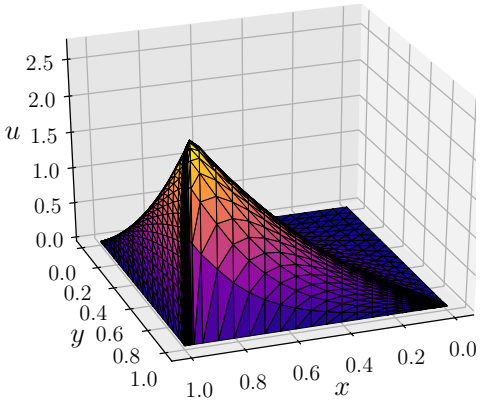
(b) Mesh generated with Algorithm 6.1



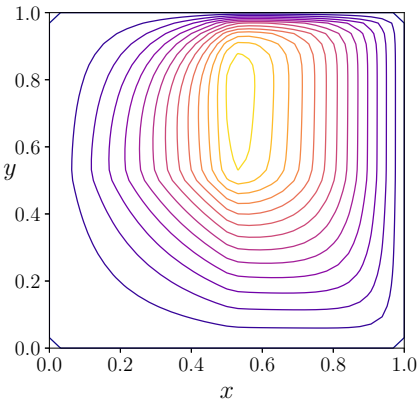
(c) Bakhvalov mesh with $\varepsilon_B = 10^{-3}$



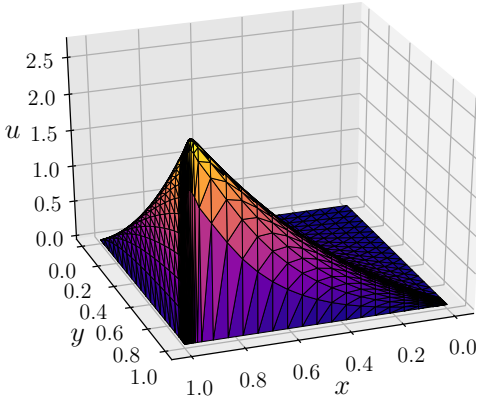
(d) Bakhvalov mesh with $\varepsilon_B = 10^{-3}$



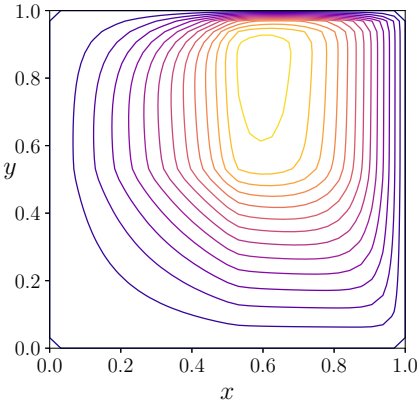
(e) Bakhvalov mesh with $\varepsilon_B = 5{\times}10^{-3}$



(f) Bakhvalov mesh with $\varepsilon_B = 5{\times}10^{-3}$



(g) Bakhvalov mesh with $\varepsilon_B = 9{\times}10^{-3}$



(h) Bakhvalov mesh with $\varepsilon_B = 9{\times}10^{-3}$

Figure 6.13: Plots and contour plots of solutions to (6.4) with $\varepsilon = 10^{-3}$ and $N = 32$

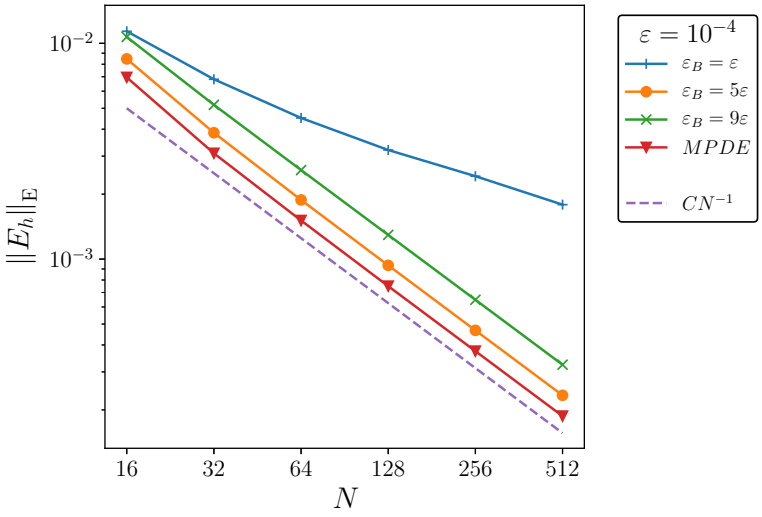


Figure 6.14: $\|E_h\|_{\mathrm{E}}$ for (6.14) with $\varepsilon = 10^{-4}$

reaction-diffusion problem with spatially varying diffusion, where the number of iterations performed when the desired mesh size is reached is determined by achieving a criterion that verifies that the meshes generated approximately equidistribute the mesh density function, $M(\boldsymbol{x})$.

The stopping criterion is chosen in the same manner as for the one-dimensional problems in Section 4.2. The uniform meshes on $\overline{\Omega}^{[c]}$, have "crossed" diagonals, an example of which is shown in Figure 6.15 with $N = 4$. The mesh has $R = (N+1)^2 + N^2$ vertices. In the weak form of (6.8), we denote the basis functions $v_h \in V_h$, a finite dimension subspace of $H_0^1(\overline{\Omega}^{[c]})$, as

$$v_h = \psi_i \text{ for } i = 1, \ldots, R.$$

At each iteration, k, we set $\mathrm{res}^{[k]}$ to be the function on the $\mathcal{P}_1$ space, whose value at each mesh point is given by

$$\mathrm{res}_i^{[k]} = \int M\left(x_h^{[k]}\right)\left(x_h^{[k]}\right)' \psi_i' dA \text{ for } i = 1, \ldots, R$$

subject to

$$\mathrm{res}^{[k]} = 0 \text{ on } |_{\partial\Omega^{[c]}}.$$

The iteration terminates when

$$\|\mathrm{res}^{[k]}\|_{0,\Omega^{[c]}} = \left(\int_{\overline{\Omega}^{[c]}} \left(\mathrm{res}^{[k]}\right)^2 dA\right)^{\frac{1}{2}} \leq \mathrm{TOL},$$

we set the value of TOL as $10^{-5}/(2N\varepsilon)$. The meshes are generated in the same way and

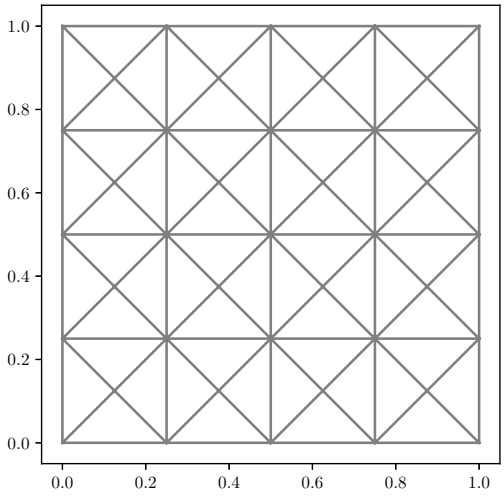


Figure 6.15: Example of mesh with crossed diagonals with $N = 4$

using the same parameters as in Section 6.2.2 except for the change outlined above.

The estimate of the error, $\|E_h\|_{\mathrm{E}}$, in the numerical solution to (6.14) are shown in Figure 6.16. One observes that Figure 6.16 is indistinguishable from Figure 6.7, the plot of the error estimator when we perform 5 iterations when the final mesh size is reached. The number of iterations required on the final meshes are shown in Table 6.4, and are independent of ε and have only a weak dependency on N.

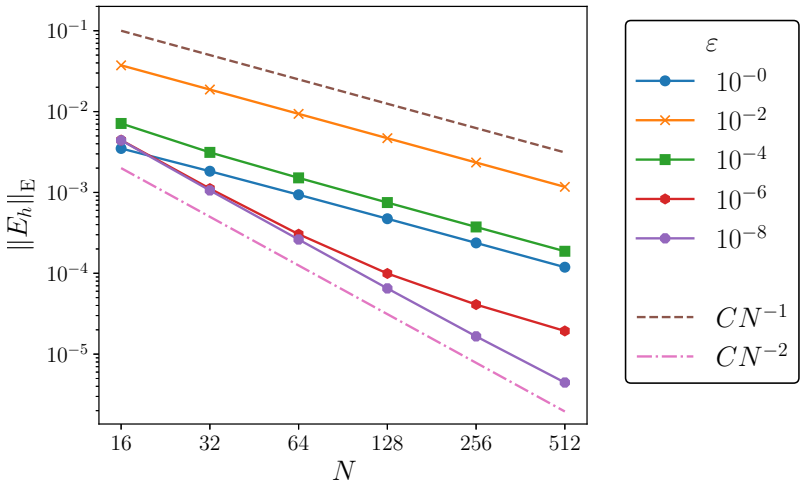


Figure 6.16: $\|E_h\|_{\mathrm{E}}$ for (6.14) when the meshes are generated using a residual-based stopping criterion

Table 6.4: Number of iterations on the final mesh for (6.14)

ε^2/N	16	32	64	128	256	512
1	1	1	1	1	1	1
10^{-2}	8	6	3	2	1	1
10^{-4}	12	10	8	6	3	1
10^{-6}	8	7	7	5	3	1
10^{-8}	8	6	5	3	3	1

6.4 Conclusions

We have introduced a new "Mesh PDE" approach for constructing layer-adapted meshes for singularly perturbed problems. It is novel in that it generates meshes suitable for two-dimensional problems where tensor-product meshes are not appropriate. This is different to the classical approaches for constructing Bakhvalov meshes, which can lead only to tensor-product grids.

We have extended the efficient non-linear solver, based on h-refinement, from Section 4.3, to the two-dimensional setting. Although not discussed, our experiments have shown that the run-time required is independent of ε; this is to be expected since, essentially, so too are the iterations counts.

We have reported on numerical experiments which demonstrate the accuracy of the method. Furthermore, we include full FEniCS implementations of the method for (6.14) in Appendix D.

For the one-dimensional problems considered in Chapter 4, the numerical results we have presented are underpinned by known theory. However, proving convergence for two-dimensional problems would rely on detailed analysis of the solutions to (6.3) and (6.4), which are not currently available.

In spite of this, we are confident that the approach could successfully apply to a wide range of problems, including those posed on more complicated domains, and ones whose solutions feature interior layers. We elaborate on this briefly in Section 9.2.

In this chapter, we have verified that the MPDE method approximately equidistributes ρ for a range of two-dimensional SPDEs. We have shown that numerical solutions to an SPDE with spatially varying diffusion, computed on meshes generated using the MPDE method, are more robust than those computed on tensor product Bakhvalov meshes. Finally, we have confirmed that iterating the MPDE a fixed number of times results in meshes that are equivalent to those generated when iterating the method until an appropriate tolerance criterion is achieved. This supports our argument that the MPDE method could be successfully applied to a wide range of problems.

In subsequent chapters, we generalise the idea presented in Section 6.2 in a number

of ways including extending the *a posteriori* method from Chapter 5 to two dimensions (Chapter 7), and further to the Oseen equations (Chapter 8).

Chapter 7

Two-dimensional *a posteriori* fitted meshes via MPDEs

7.1 Introduction

In the chapter, our goal is to extend the algorithms of Chapter 6 so that, instead of relying on *a priori* information to construct meshes, we use *a posteriori* computed quantities. That is, we extend the ideas from Chapter 5 to two-dimensions.

More precisely, in Chapter 6 we presented novel algorithms for generating meshes for the numerical solution of two-dimensional SPDEs. In the case where the appropriate mesh is uniform in one coordinate direction but layer-adapted in the other, we use the MPDE, for $\boldsymbol{x}(\xi_1, \xi_2) = (x, y)^T$,

$$\left(\rho(\boldsymbol{x}(\xi_1, \xi_2))\boldsymbol{x}_{\xi_2}(\xi_1, \xi_2)\right)\right)_{\xi_2} = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{7.1}$$

with appropriate boundary conditions. Alternatively, if a problem demands a mesh that is layer-adapted in both directions, the MPDE is formulated as the two-dimension vector-valued Poisson equation, for $\boldsymbol{x}(\xi_1, \xi_2) = (x, y)^T$,

$$\nabla^{[c]} \cdot (M(\boldsymbol{x}(\xi_1, \xi_2))\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{7.2}$$

with appropriate boundary conditions.

In this chapter, we propose mesh-construction algorithms that rely on (7.1) and (7.2), but for which $\rho = \rho(\boldsymbol{x}, u_h)$ and $M = M(\boldsymbol{x}, u_h)$. That is, the mesh density functions depend on the numerical solutions – specifically, they depend on the derivatives of the numerical solution near the boundaries of the domain, $\overline{\Omega}$. Therefore, the adaptivity does not rely entirely on *a priori* information about the physical problem. It is constructed anticipating the presence of boundary layers, but not at which boundaries, and without knowing the

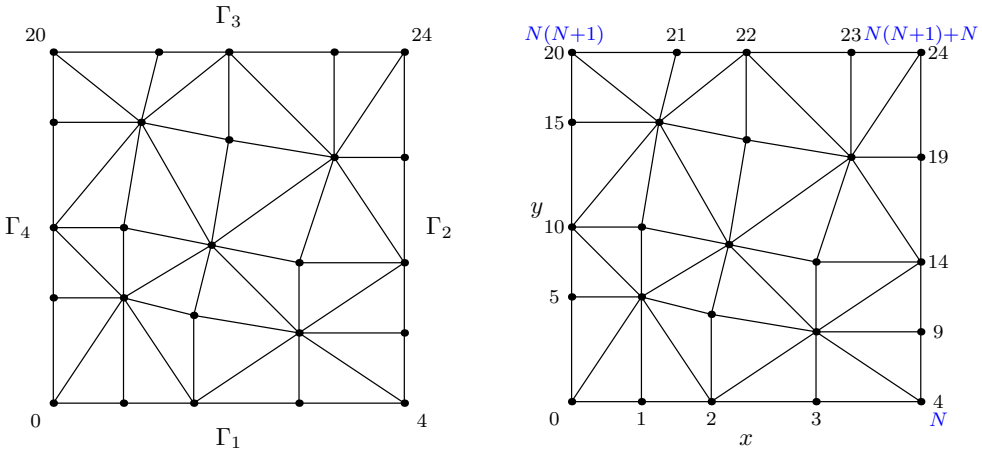
width of these layers.

This chapter is organised as follows. In Section 7.2 we discuss the MPDEs that we use, specifically showing how $\rho(\boldsymbol{x}, u_h)$ and $M(\boldsymbol{x}, u_h)$ are calculated. We present an algorithm for generating the meshes in Section 7.3. In Section 7.4 we present the results of numerical experiments that demonstrate the accuracy and robustness of the algorithm.

7.2 MPDEs

Our first MPDE is (7.1) and $\rho(\boldsymbol{x}, u_h)$ is based on the derivatives of the numerical solution near the boundaries of the domain, $\overline{\Omega}$, which are labelled as in Figure 7.1(a). Our second MPDE is (7.2) and again $M(\boldsymbol{x}, u_h)$ is now based on the derivatives of the numerical solution on the boundaries of the domain.

Although the meshes we construct are not tensor-product, they are M-uniform. Therefore, we can denote the mesh points as $(x, y)_k$, for $k = 0, 1, \ldots, (N+1)^2 - 1$, indexed using standard lexicographic ordering. For example, in Figure 7.1(b) we show the indices of the mesh points when $N = 4$.



(a) Boundaries (b) Indices of the mesh points

Figure 7.1: Example of boundaries and indices of mesh points on a non-tensor product grid with $N = 4$

The formulation of ρ and M presented in Chapter 6 can be thought of as representing point-wise bounds of $|u'(x, y)|$. Since the derivatives are largest in the layer regions we use the derivatives of the numerical solutions near each boundary to estimate these bounds. If no layer exists near a boundary the value of the numerical derivatives are of $\mathcal{O}(1)$ and have no significant effect on the formulation of ρ and M. We set

$$\upsilon_1(x, y)_k = \frac{\partial u_h}{\partial y}(x, y)_l^+, \text{ where } l = k \bmod N{+}1, \tag{7.3a}$$

$$\upsilon_2(x, y)_k = \frac{\partial u_h}{\partial x}(x, y)_l^-, \text{ where } l = N + \left\lfloor \frac{k}{N{+}1} \right\rfloor (N{+}1), \tag{7.3b}$$

$$\upsilon_3(x,y)_k = \frac{\partial u_h}{\partial y}(x,y)_l^-, \text{ where } l = N(N+1)+k \bmod N+1, \tag{7.3c}$$

$$\upsilon_4(x,y)_k = \frac{\partial u_h}{\partial x}(x,y)_l^+, \text{ where } l = \left\lfloor \frac{k}{N+1} \right\rfloor (N+1). \tag{7.3d}$$

That is, these functions propagate the derivatives of u_h on their respective boundaries across the domain. For example, if $\partial u_h/\partial x(x,y)_0^+ = 1$, then $\upsilon_4(x,y)_k = 1$ for $k = 0,\ldots,N$. Alternatively, for example, we could set $\upsilon_1(x,y) = \partial u_h/\partial y(x,0)$, this would be computationally expensive on a non-tensor product grid as the derivatives at the boundary would need to be interpolated onto each line of the grid i.e, N times. Since υ is only relevant near the related boundary calculating υ as detailed in (7.3) has no significant effect on the quality of the solutions calculated on the meshes generated.

7.3 Algorithm and implementation

We discuss the implementation of the method presented in Section 7.2 and present Algorithm 7.1 for (7.2) since it is a more complicated problem. As in Algorithm 6.1, we use a combination of the MPDE method and uniform h-refinement to optimise the algorithm. As in Algorithm 5.1, we found that the derivatives of the numerical solutions generated on meshes when $N < 16$ were not a good estimation for the derivatives of the true solution, so our initial mesh has 16 mesh intervals in each direction. The implementation steps of Algorithm 7.1 are as follows.

1. We start with a uniform tensor-product mesh with 16 mesh intervals in each coordinate direction and perform $\lceil 6\log_{10}\left(\varepsilon^{-1}\right)\rceil$ iterations of the MPDE method. Since only one mesh point is added to the layer region at each iteration and we require the minimum mesh width to be $\mathcal{O}(\varepsilon/N)$, one can deduce that $\mathcal{O}(\log\varepsilon^{-1})$ iterations will be required. Experiments indicate that $\lceil 6\log_{10}\left(\varepsilon^{-1}\right)\rceil$ iterations suffice.

2. We then perform a uniform h-refinement and 4 iterations of the MPDE on subsequent meshes until the mesh has the required number of mesh points. This requires solving u and calculating u_x and u_y to update the MPDE at each iteration.

As with Algorithm 6.1, computational experimentation indicated that this is sufficient and we do not need to iterate until a specified tolerance is achieved.

7.4 Numerical Results

7.4.1 Reaction-diffusion equation on an irregular domain

In Section 6.2.3 we presented numerical results for (6.3) on an irregular domain. Here we present a slight variation on that reaction-diffusion problem posed on the same domain:

$$-\varepsilon^2\Delta u + u = 1, \text{ for } (x,y) \in \Omega, \tag{7.6a}$$

where Ω is the polygon with vertices $(0,0),(0,1),(1,1)$ and $(1,1/2)$, and is subject to the boundary conditions

$$u\left(x,\frac{x}{2}\right)=u(x,1)=0, \quad \text{and} \quad \frac{\partial u}{\partial x}(0,y)=\frac{\partial u}{\partial x}(1,y)=0. \tag{7.6b}$$

The solution to (7.6) exhibits boundary layers, whose widths are $\mathcal{O}(\varepsilon)$, near $y=x/2$ and $y=1$, as shown in Figure 7.2. There are no boundary layers at $x=0$ or $x=1$ because of the boundary conditions.

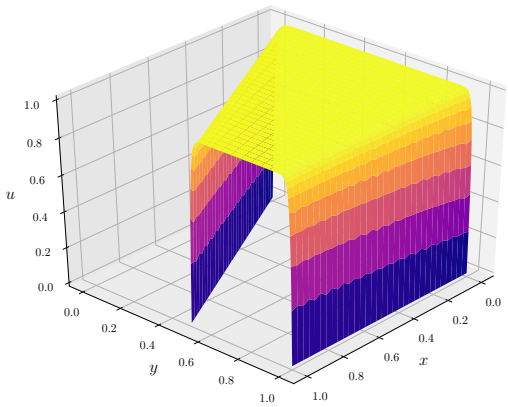


Figure 7.2: FEM solution to (7.6) with $N=32$ and $\varepsilon=10^{-2}$

For this problem, a mesh that is layer-adapted only in the y-direction is adequate. For this scenario, an appropriate MPDE, for $\boldsymbol{x}(\xi_1,\xi_2)=(x,y)^T$, is (7.1), where $\Omega^{[c]}$ is the polygon with vertices $(0,0),(0,1),(1,1)$ and $(1,1/2)$, and subject to the boundary conditions

$$x(0,\xi_2)=0, \quad x(1,\xi_2)=1, \quad x\left(\xi_1,\frac{\xi_1}{2}\right)=2\xi_2, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1,1)=0,$$
$$y\left(\xi_1,\frac{\xi_1}{2}\right)=\xi_1/2, \quad y(\xi_1,1)=1, \quad \frac{\partial y}{\partial \mathbf{n}}(0,\xi_2)=0, \quad \frac{\partial y}{\partial \mathbf{n}}(1,\xi_2)=0.$$

The boundary conditions on x (for example) ensure that the x-coordinate of the mesh points are fixed at $\xi_1=0$ and $\xi_1=1$, while allowing them to vary along the boundary $\xi_1=1$, yet remaining on the boundary at $\xi_1=2\xi_2$. To solve (7.6), we generate a mesh that is concentrated near $y=x/2$ and $y=1$ using Algorithm 7.1. The initial mesh $\omega^{[c,0]}$ has the mesh points $(\xi_1,\xi_2)=(\bar{\xi}_1,(\bar{\xi}_2(2-\bar{\xi}_1)+\bar{\xi}_1)/2)$, where $(\bar{\xi}_1,\bar{\xi}_2)$ are the mesh points of a uniform mesh on $[0,1]^2$, with $N=16$ intervals in each direction.

Then the MPDE (7.1) is solved with

$$\rho(\boldsymbol{x},u_h(x,y))=\max\left\{1,K|\upsilon_1|\exp\left(\frac{|\upsilon_1|(y-x/2)}{\sigma}\right)+K|\upsilon_3|\exp\left(\frac{|\upsilon_3|(1-y)}{\sigma}\right)\right\},$$

where $\upsilon_1=\upsilon_1(u_h)$ and $\upsilon_3=\upsilon_3(u_h)$ are as defined in (7.3). Of the other parameters,

$K = 0.28$ is chosen based on numerical experiments, $\sigma = 2.5$ depends on the formal order of the underlying method. An example of the resulting mesh is shown in Figure 7.3. In Figure 7.4, one observes that no sharp layer exists in the numerical solution when transformed onto the computational domain $\Omega^{[c]}$ which demonstrates that the physical boundary layers are resolved.

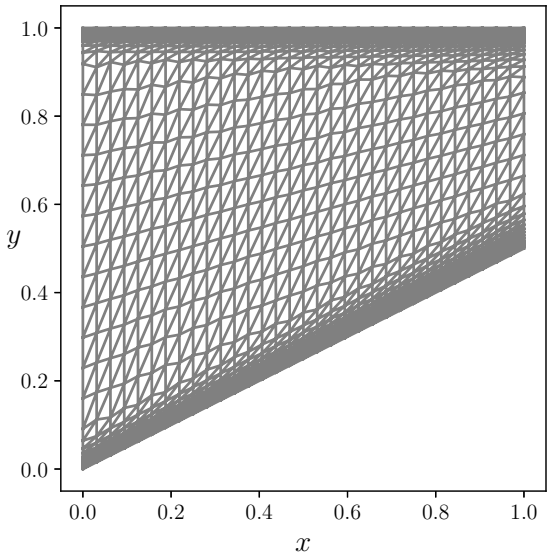


Figure 7.3: Mesh generated using (7.1), for (7.6), with $N = 32$ and $\varepsilon = 10^{-2}$

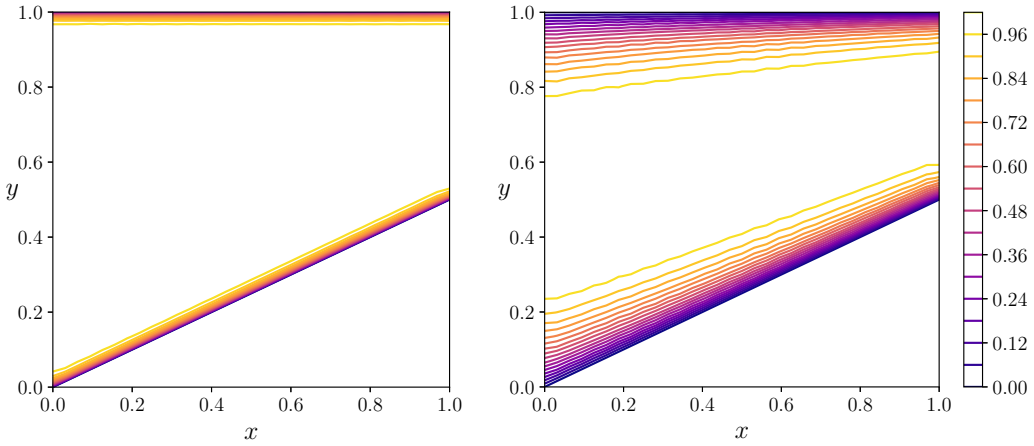


Figure 7.4: Contour plots of FEM solution to (7.6) with $\varepsilon = 10^{-2}$ and $N = 32$, on Ω in Figure 7.4(a), and on $\Omega^{[c]}$ in Figure 7.4(b)

Figure 7.5 shows the error estimates in the numerical solution of (7.6), measured in the energy norm; one observes that they converge linearly in N and scale like $\varepsilon^{1/2}$, as per (1.1). The errors have the same order of magnitude and convergence as those shown in Figure 6.4 for (6.3) on the same domain.

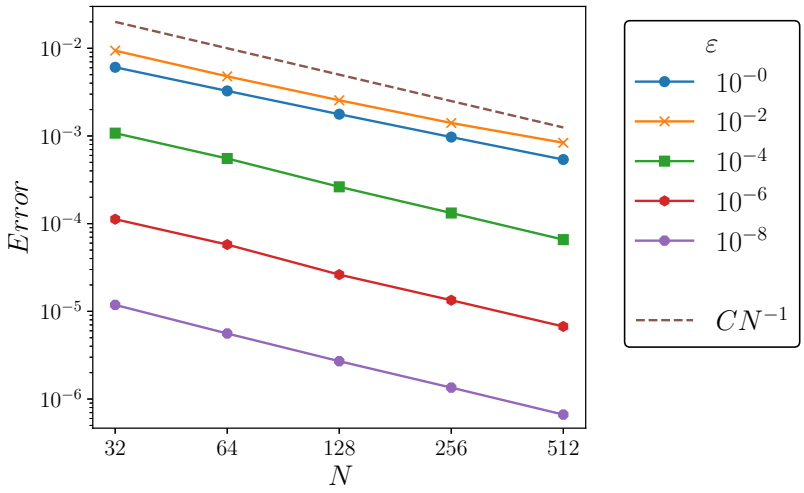


Figure 7.5: Plot of $\|E_h\|_{\mathrm{E}}$ for (7.6) using Algorithm 7.1

7.4.2 Reaction-diffusion equation with varying diffusion

In this section, we focus on solving the reaction-diffusion equation (7.7). This was previously studied in Section 6.2.3. Its key feature is that the diffusion term varies on the domain, so the boundary layers vary in width. Consequently, even when posed on a regular domain (such as a unit square) optimal meshes are not tensor-product in nature. The MPDE approach can generate such meshes.

The two-dimensional reaction-diffusion equation, with varying diffusion is,

$$-\nabla\cdot\left(\begin{pmatrix}\varepsilon(1+2y)^2 & 0\\ 0 & \varepsilon(3-2x)^2\end{pmatrix}^2\nabla u(x,y)\right)+u(x,y)=(e^x-1)\,(e^y-1)\,,$$

$$\text{for } (x,y)\in\Omega=(0,1)^2, \quad (7.7a)$$

with boundary conditions

$$u=0 \text{ on } \partial\Omega. \quad (7.7b)$$

To generate the meshes using the MPDE method, we solve (7.2) with boundary conditions,

$$\begin{aligned} &x(0,\xi_2)=0, \quad x(1,\xi_2)=1, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1,0)=0, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1,1)=0,\\ &y(\xi_1,0)=0, \quad y(\xi_1,1)=1, \quad \frac{\partial y}{\partial \mathbf{n}}(0,\xi_2)=0, \quad \frac{\partial y}{\partial \mathbf{n}}(1,\xi_2)=0. \end{aligned} \quad (7.8)$$

We take

$$M(\boldsymbol{x})=\begin{pmatrix}m_{1,1} & 0\\ 0 & m_{2,2}\end{pmatrix},$$

where

$$m_{1,1} = \max\left\{1, K_1\left(|\upsilon_4|\exp\left(-\frac{|\upsilon_4|\,x}{\sigma}\right) + |\upsilon_2|\exp\left(-\frac{|\upsilon_2|\,(1-x)}{\sigma}\right)\right)\right\} \tag{7.9}$$

$$m_{2,2} = \max\left\{1, K_2\left(|\upsilon_1|\exp\left(-\frac{|\upsilon_1|\,y}{\sigma}\right) + |\upsilon_3|\exp\left(-\frac{|\upsilon_3|\,(1-y)}{\sigma}\right)\right)\right\}, \tag{7.10}$$

with $K_1 = K_2 = 0.4$ chosen based on numerical experience. As before, since we use a $\mathcal{P}_1$-FEM we take $\sigma = 2.5$. The terms $\upsilon_1, \upsilon_2, \upsilon_3$ and υ_4 are as defined in (7.3).

A plot of errors in the energy norm for (7.7) solved using Algorithm 7.1 is shown in Figure 7.6. One observes robust convergence, the $\varepsilon^{1/2}N^{-1}$ term dominates when $N^{-1} < \varepsilon^{1/2}$, otherwise, the N^{-2} term dominates.

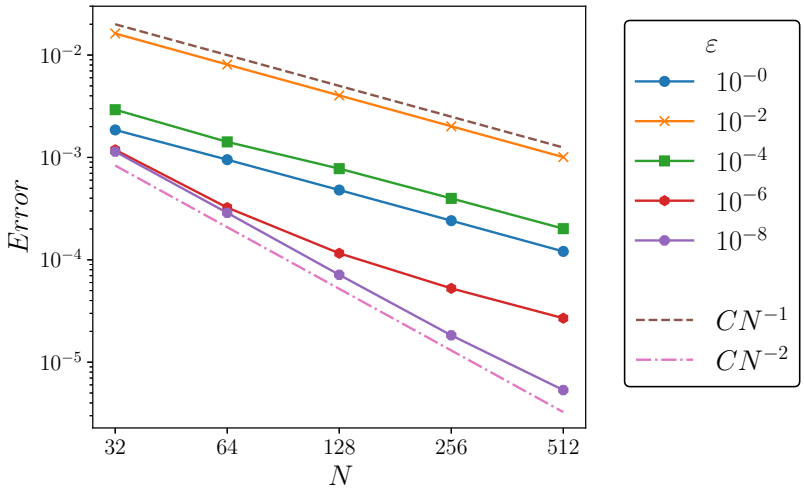


Figure 7.6: Plot of $\|E_h\|_{\mathrm{E}}$ for (7.7) using Algorithm 7.1

7.5 Conclusion

The MPDEs presented in this chapter use *a posteriori* information about the related SPDE. More precisely, using only the knowledge that the solution possesses boundary layers (but not their location or width) layer-adapted meshes are generated. The solutions are robust and the errors converge as expected. The magnitude and rates of convergence of the errors are similar to when the solutions are generated using the MPDE method based on *a priori* information, as in Chapter 6. However, the method presented is only suitable for problems whose solutions exhibit boundary layers. Further work would be required to extend the method so that it is suitable for problems whose solutions contain interior layers. Nonetheless, we believe that the method shows promise. In the next chapter, the final substantial one of this thesis, we will demonstrate applicability to problems in computational fluid dynamics.

Input: N, the number of intervals in both directions in the final mesh on $\Omega^{[c]}$.
Input: M, a mesh density function.
Set $\omega^{[c;0]} := \{(\xi_1, \xi_2)_i\}_{i=0}^{16}$ to be a uniform tensor-product mesh on $\overline{\Omega}^{[c]}$ with 16 mesh intervals in each coordinate direction;
Set $\boldsymbol{x}(\xi_1, \xi_2) = (\xi_1, \xi_2)$ for $(\xi_1, \xi_2) \in \overline{\Omega}^{[c]}$;
Set $v_l^{[0]} = 1$ for $l = 1, 2, 3, 4$;
$\boldsymbol{r} \leftarrow \boldsymbol{x}$;
$k \leftarrow 1$;
for j *in* $0 : \lceil 6 \log_{10}\left(\varepsilon^{-1}\right) \rceil$ **do**
    set $\boldsymbol{x}$ to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;0]}$, to

$$\nabla^{[c]} \cdot (M(\boldsymbol{r}, s)\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{7.4}$$

    with boundary conditions as defined in (7.8);
    Set $\omega^{[k]} := \boldsymbol{x}(\omega^{[c;0]})$ to be the adapted mesh on $\overline{\Omega}$ at step k;
    $u_h^{[k]} \leftarrow \mathcal{P}_1$-FEM solution to the SPDE on $\omega^{[k]}$;
    Calculate $v_l^{[k]}$ for $l = 1, 2, 3, 4$;
    $\boldsymbol{r} \leftarrow \boldsymbol{x}$;
    $s \leftarrow u_h^{[k]}$;
    $k \leftarrow k+1$;
end
for i *in* $1 : (\log_2 N - 4)$ **do**
    $w^{[c;i]} \leftarrow$ refine $w^{[c;i-1]}$;
    $s \leftarrow s$ interpolated onto $w^{[c;i]}$;
    for j *in* $1 : 4$ **do**
        set $\boldsymbol{x}$ to be the $\mathcal{P}_1$-FEM solution, on $\omega^{[c;i]}$, to

$$\nabla^{[c]} \cdot (M(\boldsymbol{r}, s)\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{7.5}$$

        with boundary conditions as defined in (7.8);
        Set $\omega^{[k]} := \boldsymbol{x}(\omega^{[c;i]})$ to be the adapted mesh on $\overline{\Omega}$ at step k;
        $u_h^{[k]} \rightarrow \mathcal{P}_1$-FEM solution to the SPDE on $\omega^{[k]}$;
        Calculate $v_l^{[k]}$ for $l = 1, 2, 3, 4$;
        $\boldsymbol{r} \leftarrow \boldsymbol{x}$;
        $s \leftarrow u_h^{[k]}$;
        $k \leftarrow k+1$;
    end
end
Set $\omega := x(\omega^{[c;i]})$ to be the adapted mesh on $\overline{\Omega}$;

Algorithm 7.1: Generate a two-dimensional layer-adapted mesh using an MPDE using *a posteriori* information and h-refinement.

Chapter 8

Applications of MPDEs to problems in fluid dynamics

8.1 Introduction

In this chapter, we consider the application of algorithms developed in earlier chapters to two classes of PDEs that have applications in fluid dynamics. The first class features (scalar) time-dependent SPDEs, and we consider both reaction-diffusion and convection-diffusion variants. The second of these arises in fluid dynamics as a transport model (sometimes called an "advection-diffusion" problem in that context), see [Morton, 1996, Chap. 1]. The second class of problems is steady but vector-valued and two-dimensional in space. Formulated by Carl Wilhelm Oseen in 1910, the Oseen equations are of interest for fluid flow problems. For example, if one uses an implicit time discretization when solving the Navier-Stokes equations and applies a fixed-point iterative method, e.g., Picard's method, it leads to solving the Oseen equations at each iteration [John, 2016, Chap. 5].

This chapter is organised as follows. In Section 8.2 we discuss our implementation of the MPDE method outlined in Chapter 5 for time-dependent SPDEs. We present our numerical findings for two time-dependent SPDEs in Section 8.2.2, one a reaction-diffusion equation and the other a convection-diffusion equation. In Section 8.3 we outline how to extend the MPDE method using *a posteriori* information for the two-dimensional Oseen equations. Specifically, in Section 8.3.1 we discuss the related MPDE, detailing how $M(\boldsymbol{x}(\xi_1, \xi_2), \boldsymbol{u}_h(x, y))$ is formulated, and in Section 8.3.2 we present results of numerical experiments.

8.2 One-dimensional unsteady SPDEs

The general one-dimensional time-dependent SPDEs that we consider in this section is

$$u_t - \varepsilon^2 u'' + q(x,t)u' + r(x,t)u = f(x,t), \quad \text{for } (x,t) \in \Omega^{[x,t]} := (0,1) \times (0,T], \tag{8.1a}$$

with boundary and initial conditions,

$$u(0,t) = u(1,t) = u(x,0) = 0. \tag{8.1b}$$

The numerical method we employ combines the Crank-Nicolson method (for integration in time) with the Galerkin $\mathcal{P}_1$-FEM.

We write the weak form of the spatial operator in (8.1), at a fixed time-step j, as

$$\mathcal{B}(u^{[j]}, v) := \varepsilon^2((u^{[j]})', v') + (q(x,t_j)(u^{[j]})', v) + (r(x,t_j)u^{[j]}, v).$$

Taking M time-steps of length $\tau = T/M$, the semi-discrete problem is

$$(u^{[j]}, v) + \frac{\tau}{2}\mathcal{B}(u^{[j]}, v) = (u^{[j-1]}, v) - \frac{\tau}{2}\mathcal{B}(u^{[j-1]}, v) + \frac{\tau}{2}(f(x,t_j) + f(x,t_{j-1}), v),$$

at each time-step, $j = 1, 2, \ldots, M$.

To get the fully discrete problem, we set $u_h^{[j,k]}$ be the numerical solution to (8.1) computed on the k^{th} mesh generated by Algorithm 5.1 at time-step j with $u_h^{[j]}$ denoting the solution on the final mesh. Then we solve the problem: *find* $u_h^{[j,k]} \in V_0^h\left(\Omega^{[x,t_j]}\right)$, *a finite-dimensional subspace of* $H_0^1\left(\Omega^{[x,t_j]}\right)$, *such that*

$$\begin{aligned}(u_h^{[j,k]}, v_h) + \frac{\tau}{2}\mathcal{B}(u_h^{[j,k]}, v_h) = (u_h^{[j-1]}, v_h) - \frac{\tau}{2}\mathcal{B}(u_h^{[j-1]}, v_h) \\ + \frac{\tau}{2}(f(x,t_j) + f(x,t_{j-1}), v_h) \text{ for all } v_h \in V_0^h\left(\Omega^{[x,t_j]}\right).\end{aligned} \tag{8.2}$$

Then (8.2) is solved for $k = 1, 2, \ldots,$ at each time-step, $j = 1, 2, \ldots, M$.

Note that (8.2) involves both $u_h^{[j]}$ and $u_h^{[j-1]}$, which are computed on different meshes. Therefore, we use linear interpolation to map the solutions between the evolving meshes. The meshes generated in this chapter are graded near the boundaries. Therefore, using projection onto a cubic function space, as used in Chapter 3 for problems with interior layers, is not suitable as it can result in mesh points occurring outside the domain since such a projection is not necessarily monotonic. Potentially, a monotonic cubic interpolation method, such as a cubic Hermite spline, could be used instead, which would ensure solution accuracy is not lost and the method remains robust. However, currently only Lagrange interpolation is implemented in FEniCS.

In Chapter 5, we propose new algorithms for generating layer-adapted meshes for the finite element solution of one-dimensional SPDEs of the form

$$-\varepsilon^2 u'' + q(x)u' + r(x)u = f(x), \text{ for } x \in (0,1),$$

based on the MPDE

$$(\rho(x,u_h)x_\xi(\xi))_\xi = 0, \text{ for } \xi \in \Omega^{[c]}, \text{ with } x(0) = a \text{ and } x(1) = b, \tag{8.3}$$

where a and b are the endpoints of the spatial domain. The mesh density function, ρ, depends on *a posteriori* information on the physical differential equation. In this chapter we use (8.3) to generate the meshes for one-dimensional time-dependent SPDEs at each time-step, with a slight variation on the formulation of $\rho(x,u_h)$, which we describe in Section 8.2.1.

8.2.1 Mesh density function

The solution to the problems we consider may feature boundary layers near either of the boundaries at $x = 0$ or $x = 1$. To solve the time-dependent SPDEs, the meshes at each time step are generated from the solutions of (8.3), with

$$\rho(x,u_h) = \max\left\{1, K\left(|bu_h'(x_0^+)|\exp\left(-\frac{|bu_h'(x_0^+)|x}{\sigma}\right) + |bu_h'(x_N^-)|\exp\left(-\frac{|bu_h'(x_N^-)|(1-x)}{\sigma}\right)\right)\right\}, \tag{8.4}$$

where σ depends on the numerical scheme, and K and $b(x,t)$ are problem-specific. At each time step, the meshes are generated using Algorithm 5.1. The formulation of ρ is independent of the location or number of boundary layers in the solution to the related SPDE, as shown in Section 8.2.2.

8.2.2 Numerical results

We present numerical results for a time-dependent reaction-diffusion equation in Section 8.2.3, and for a time-dependent convection-diffusion equation in Section 8.2.4.

8.2.3 One-dimensional unsteady reaction-diffusion equation

Our chosen one-dimensional reaction-diffusion equation is (8.1) with $q \equiv 0$, $r = (1+3t)^2$, $f \equiv 1$, and $T = 1$, i.e.,

$$\dot{u} - \varepsilon^2 u'' + (1+3t)^2 u = 1, \quad \text{for } (x,t) \in \Omega^{[x,t]} := (0,1)\times(0,1], \tag{8.5a}$$

with boundary and initial conditions,

$$u(0,t) = u(1,t) = u(x,0) = 0. \tag{8.5b}$$

As shown in Figures 8.1 and 8.2, the solution to (8.5) exhibits boundary layers near both $x = 0$ and $x = 1$ at each time step. The widths of the layers are of $\mathcal{O}(\varepsilon \ln(1/\varepsilon))$, but vary in time.

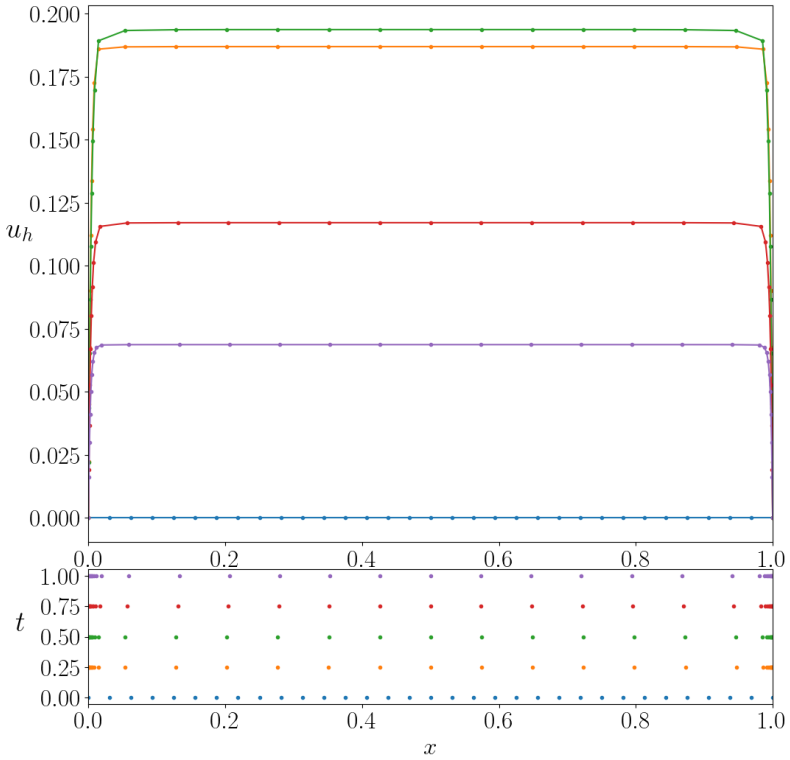


Figure 8.1: Example of a solution to (8.5) at a number of time-steps with $\varepsilon = 10^{-2}$ and $N = 32$

When generating the meshes to solve (8.5), ρ is formulated as in (8.4), with $K = 0.28$ chosen based on numerical experiments, and $b = r(x,y) = (1+3t)^2$ taken from the physical problem, i.e., it is the reaction coefficient.

In this section and Section 8.2.4, our error estimator is defined as

$$\|E_h\|_{\mathrm{E}} := \left\| u_{2,h}^{[M]} - u_h^{[M]} \right\|_{\mathrm{E}},$$

i.e, we compare the $\mathcal{P}_1$-FEM and $\mathcal{P}_2$-FEM solutions at $T = 1$.

In Figure 8.3, we present error estimates for (8.5) using Algorithm 5.1. One observes robust convergence which is linear in N and scales like $\varepsilon^{1/2}$, which is consistent with the error bound given in (2.10) for the time-independent problem.

For comparison purposes, in Table 8.1 we present the error estimates for solutions

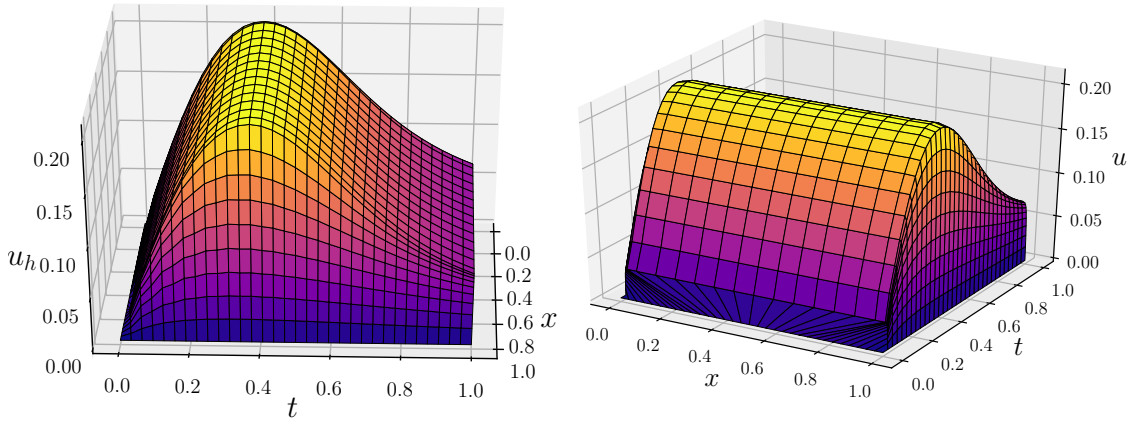


Figure 8.2: Surface plots and layer-adapted grid for a solution to (8.5) with $\varepsilon = 10^{-2}$ and $N = 32$

computed using the method of Section 8.2.1, and in Table 8.2 we present the error estimates calculated on classic Bakhvalov meshes. They show that the two approaches yield the same order of convergence, and exhibit the same dependency in ε. However, the solutions computed using Algorithm 5.1 outperform those on the Bakhvalov meshes by a factor of 3. Note that when $\varepsilon = 1$, the Bakhvalov mesh is a uniform mesh, however the MPDE mesh adapts to the solution even when no layer is present.

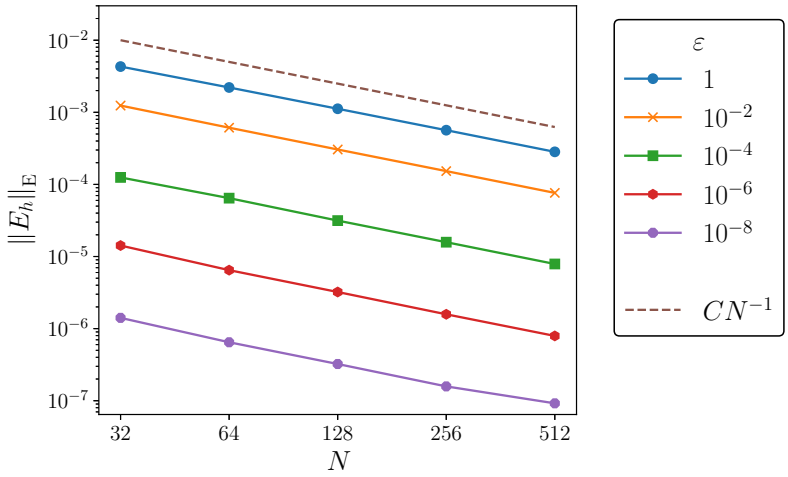


Figure 8.3: Plot of $\|E_h\|_{\mathrm{E}}$ for (8.5) using Algorithm 5.1

8.2.4 One-dimensional unsteady convection-diffusion equation

Our chosen one-dimensional convection-diffusion test problem is (8.1) with $q = (1+3t)^2$, $r \equiv 0$ and $f \equiv 1$, i.e.,

$$\dot{u} - \varepsilon^2 u'' + (1+3t)^2 u' = 1, \quad \text{for } (x,t) \in \Omega^{[x,t]} := (0,1) \times (0,1], \tag{8.6a}$$

Table 8.1: $\|E_h\|_{\mathrm{E}}$ for (8.5) using Algorithm 5.1

ε/N	32	64	128	256	512
1	4.308e-03	2.212e-03	1.121e-03	5.646e-04	2.833e-04
		0.96	0.98	0.99	0.99
10^{-2}	1.244e-03	6.115e-04	3.052e-04	1.526e-04	7.628e-05
		1.02	1.00	1.00	1.00
10^{-4}	1.253e-04	6.449e-05	3.146e-05	1.578e-05	7.880e-06
		0.96	1.04	1.00	1.00
10^{-6}	1.417e-05	6.471e-06	3.220e-06	1.582e-06	7.916e-07
		1.13	1.01	1.03	1.00
10^{-8}	1.407e-06	6.472e-07	3.230e-07	1.582e-07	9.203e-08
		1.12	1.00	1.03	0.78

Table 8.2: $\|E_h\|_{\mathrm{E}}$ for (8.5) on classic Bakhvalov meshes

ε/N	32	64	128	256	512
1	4.580e-03	2.351e-03	1.191e-03	5.993e-04	3.006e-04
		0.96	0.98	0.99	1.00
10^{-2}	3.836e-03	1.985e-03	1.002e-03	5.024e-04	2.514e-04
		0.95	0.99	1.00	1.00
10^{-4}	3.836e-04	1.985e-04	1.002e-04	5.024e-05	2.514e-05
		0.95	0.99	1.00	1.00
10^{-6}	3.836e-05	1.985e-05	1.002e-05	5.024e-06	2.514e-06
		0.95	0.99	1.00	1.00
10^{-8}	3.836e-06	1.985e-06	1.002e-06	5.024e-07	2.514e-07
		0.95	0.99	1.00	1.00

with boundary and initial conditions,

$$u(0,t) = u(1,t) = u(x,0) = 0. \tag{8.6b}$$

As shown in Figures 8.4 and 8.5, the solution to (8.6) exhibits a boundary layer near $x = 1$, which varies in width over time. In general, the width of the layer in the solution to (8.6) is of order $\mathcal{O}\left(\varepsilon^2 \ln(1/\varepsilon^2)\right)$ and so is much sharper than layers present in the solution to the reaction-diffusion problem (8.5).

To solve (8.6), ρ is formulated as in (8.4), with $K = 0.4$ chosen based on numerical experiments and $b = q(x,t) = (1+3t)^2$ taken from the physical problem. Although the solution to (8.6) exhibits only one boundary layer, the formulation of ρ is the same as for (8.5) whose solution has two boundary layers.

Again, for comparison purposes, in Table 8.3 we present the error estimates for solutions computed using the method of Section 8.2.1, and in Table 8.4 we present the error estimates calculated on classic Bakhvalov meshes. They show that the two approaches are robust

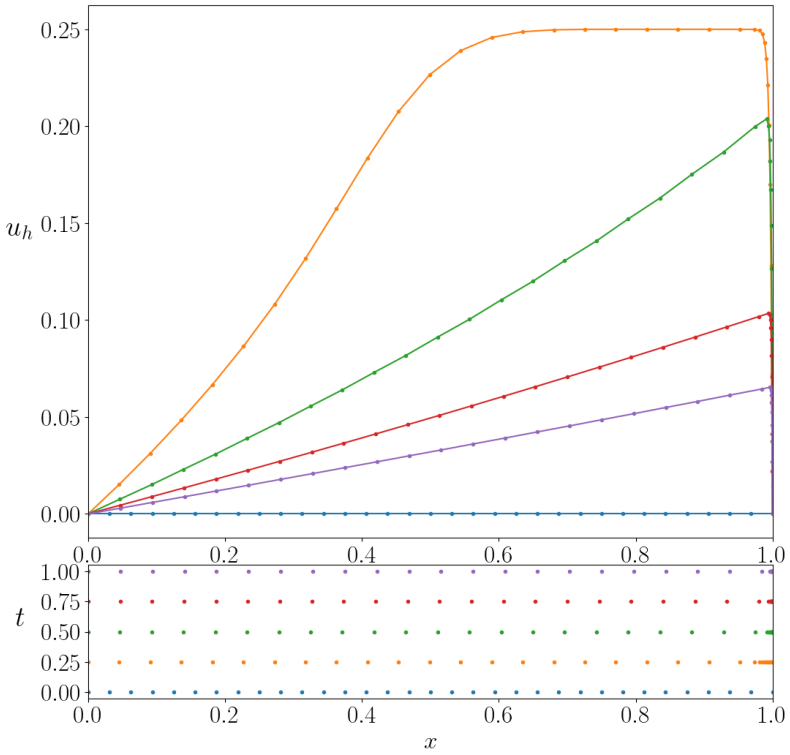


Figure 8.4: Example of a solution to (8.1) at a number of time-steps with $\varepsilon^2 = 10^{-2}$ and $N = 32$

in ε and yield similar orders of convergence. The results in Table 8.4 show the solution on a classic Bakhvalov mesh is fully first-order which is consistent with the bounds proven in [Roos, 2006]. For small ε the MPDE approach shows slightly more irregular rates of convergence. Nonetheless, it outperforms those on the Bakhvalov meshes by a factor of 10.

8.3 Oseen equations

The Oseen equations are

$$-\varepsilon\Delta\boldsymbol{u} - (\boldsymbol{q}\cdot\nabla)\,\boldsymbol{u} + r(x,t)\boldsymbol{u} + \nabla p = \boldsymbol{f} \quad \text{in } \Omega := (0,1)^2, \tag{8.7a}$$

with

$$\operatorname{div}\boldsymbol{u} = 0 \quad \text{in } \Omega, \tag{8.7b}$$

and boundary conditions

$$\boldsymbol{u} = 0 \text{ on } \quad \partial\Omega. \tag{8.7c}$$

Here $\boldsymbol{u}$ represents a velocity field and p is the pressure term. In the most general setting, one solves (8.7) for both $\boldsymbol{u}$ and p. However, in this section we consider the function p to be given. (The situation for when p is unknown is discussed briefly in Section 9.1)

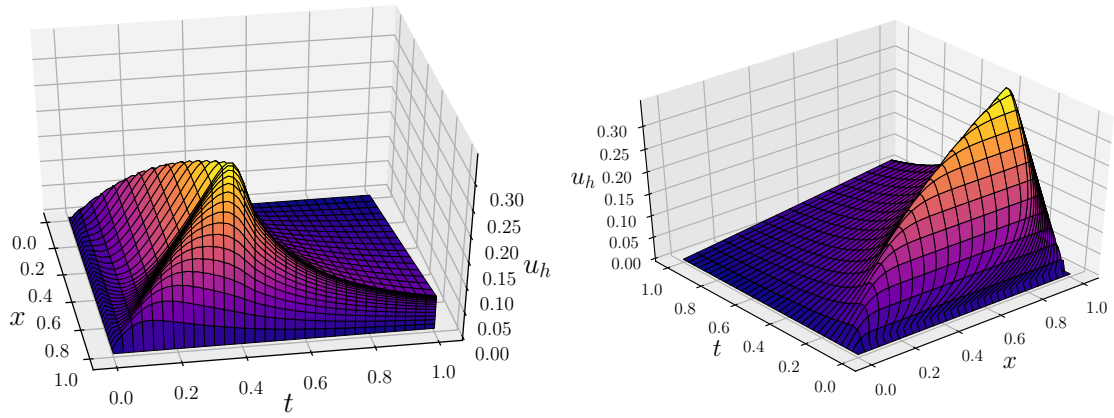


Figure 8.5: Surface plots and layer-adapted grid for a solution to (8.6) with $\varepsilon = 10^{-2}$ and $N = 64$

Table 8.3: $\|E_h\|_{\mathrm{E}}$ for (8.6) using Algorithm 5.1

ε^2/N	32	64	128	256	512
1	8.443e-03	4.215e-03	2.102e-03	1.053e-03	5.255e-04
		1.00	1.00	1.00	1.00
10^{-2}	1.595e-02	5.919e-03	2.653e-03	1.311e-03	6.536e-04
		1.43	1.16	1.02	1.00
10^{-4}	1.673e-02	1.034e-02	5.262e-03	3.403e-03	1.868e-03
		0.69	0.97	0.63	0.87
10^{-6}	2.018e-02	1.180e-02	5.941e-03	3.190e-03	1.762e-03
		0.77	0.99	0.90	0.86
10^{-8}	2.442e-02	1.026e-02	5.638e-03	3.458e-03	2.210e-03
		1.25	0.86	0.71	0.65

In Chapter 7, when a two-dimensional problem required a mesh that is layer-adapted in both directions, we propose an MPDE that is the two-dimension vector-valued Poisson equation,

$$\nabla^{[c]} \cdot (M(\boldsymbol{x}(\xi_1, \xi_2), u_h(x, y))\nabla^{[c]}\boldsymbol{x}(\xi_1, \xi_2)) = (0, 0)^T, \text{ for } (\xi_1, \xi_2) \in \Omega^{[c]}, \tag{8.8}$$

for $\boldsymbol{x}(\xi_1, \xi_2) = (x, y)^T$, with appropriate boundary conditions. In this chapter, for the Oseen equations, we propose a mesh construction algorithm that relies on (8.8) but with $M = M(\boldsymbol{x}(\xi_1, \xi_2), \boldsymbol{u}_h(x, y))$ and $\boldsymbol{u}_h = ((u_1)_h, (u_2)_h)$. As in Section 8.2, M does not require *a priori* information about the location or width of boundary layers in the solution to the related physical problem but is dependent on the derivatives of its numerical solution. Since the implementation of the method and the algorithm are the same as discussed in Section 7.3 we omit them from this chapter.

Table 8.4: $\|E_h\|_{\mathrm{E}}$ for (8.6) on classic Bakhvalov meshes

ε^2/N	32	64	128	256	512
1	5.217e-02	2.636e-02	1.323e-02	6.624e-03	3.314e-03
		0.98	0.99	1.00	1.00
10^{-2}	2.284e-01	1.291e-01	6.699e-02	3.382e-02	1.696e-02
		0.82	0.95	0.99	1.00
10^{-4}	2.307e-01	1.296e-01	6.709e-02	3.384e-02	1.696e-02
		0.83	0.95	0.99	1.00
10^{-6}	2.307e-01	1.297e-01	6.711e-02	3.384e-02	1.696e-02
		0.83	0.95	0.99	1.00
10^{-8}	2.307e-01	1.297e-01	6.711e-02	3.384e-02	1.696e-02
		0.83	0.95	0.99	1.00

8.3.1 MPDE

Our MPDE is (8.8) and $M(\boldsymbol{x}, \boldsymbol{u}_h)$ is based on the derivatives of the numerical solution on the boundaries of the domain, $\overline{\Omega}$, which are labelled as in Figure 7.1(a).

As in Chapter 7, we denote the mesh points as $(x, y)_k$, for $k = 0, 1, \ldots, (N+1)^2-1$, indexed using standard lexicographic ordering. (For example, in Figure 7.1 we show the indices of the mesh points when $N = 4$.)

We set

$$\begin{aligned}
\upsilon_1(x,y)_k &= \frac{\partial (u_1)_h}{\partial y}(x,y)_l^+ + \frac{\partial (u_2)_h}{\partial y}(x,y)_l^+, \text{ where } l = k \bmod N(N+1),\\
\upsilon_2(x,y)_k &= \frac{\partial (u_1)_h}{\partial x}(x,y)_l^- + \frac{\partial (u_2)_h}{\partial x}(x,y)_l^-, \text{ where } l = N + \left\lfloor \frac{k}{N+1} \right\rfloor (N+1),\\
\upsilon_3(x,y)_k &= \frac{\partial (u_1)_h}{\partial y}(x,y)_l^- + \frac{\partial (u_2)_h}{\partial y}(x,y)_l^-, \text{ where } l = N(N+1) + k \bmod N+1,\\
\upsilon_4(x,y)_k &= \frac{\partial (u_1)_h}{\partial x}(x,y)_l^+ + \frac{\partial (u_2)_h}{\partial x}(x,y)_l^+, \text{ where } l = \left\lfloor \frac{k}{N+1} \right\rfloor (N+1).
\end{aligned}$$

That is, these functions propagate the derivatives of $\boldsymbol{u}_h = ((u_1)_h, (u_2)_h)$ on their respective boundaries across the domain as detailed in Section 7.2 (described in more detail in Section 7.2). We use Algorithm 7.1 to implement the method with $u_h = \boldsymbol{u}_h$, $s = \boldsymbol{s}$ and perform one additional iteration of the MPDE on the initial mesh.

8.3.2 Numerical results

Our two-dimensional Oseen equations are

$$-\varepsilon \Delta \boldsymbol{u} - \left(\begin{pmatrix} 2+3y \\ 3+2x^2 \end{pmatrix} \cdot \nabla \right) \boldsymbol{u} + (1+2xy)\boldsymbol{u} + \nabla p = \begin{pmatrix} 1+x \\ 1+y \end{pmatrix} \quad \text{in } \Omega := (0,1)^2, \quad (8.9a)$$

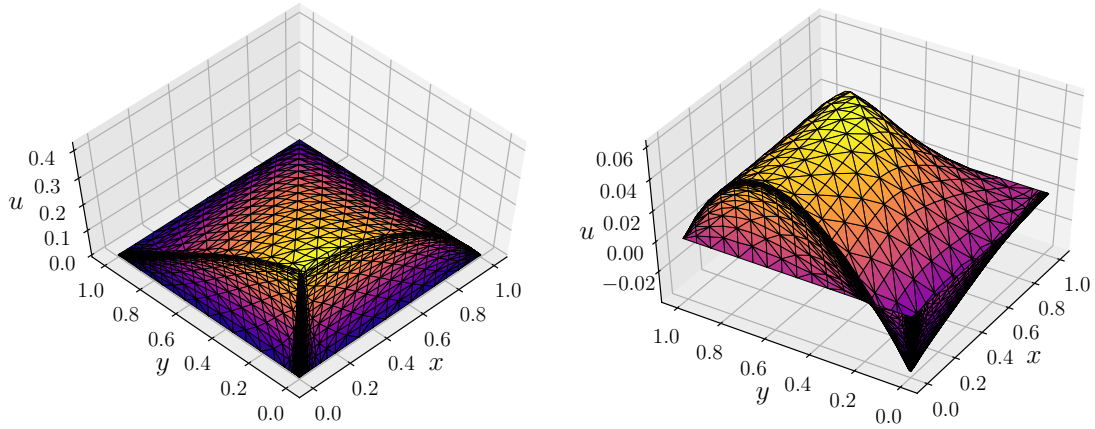
with

$$p(x, y) = 2\cos(x)\sin(y) - 2\sin(1)(1 - \cos(1)), \tag{8.9b}$$

and boundary conditions

$$\boldsymbol{u} = 0 \quad \text{on } \partial\Omega, \tag{8.9c}$$

which is motivated by [Franz et al., 2018, Eq.5.1]. As shown in Figure 8.6, the solutions u_1 and u_2 both have layers near $x = 0$ and $y = 0$ and a corner layer near $(x, y) = (0, 0)$.



(a) Velocity component u_1

(b) Velocity component u_2

Figure 8.6: Plot of a solution to (8.9) with $\varepsilon = 10^{-2}$ and $N = 32$

To generate the meshes for (8.9) using the MPDE method, we solve (8.8) with boundary conditions,

$$\begin{aligned} &x(0, \xi_2) = 0, \quad x(1, \xi_2) = 1, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1, 0) = 0, \quad \frac{\partial x}{\partial \mathbf{n}}(\xi_1, 1) = 0, \\ &y(\xi_1, 0) = 0, \quad y(\xi_1, 1) = 1, \quad \frac{\partial y}{\partial \mathbf{n}}(0, \xi_2) = 0, \quad \frac{\partial y}{\partial \mathbf{n}}(1, \xi_2) = 0. \end{aligned}$$

Here

$$M(\boldsymbol{x}, \boldsymbol{u}_h) = \begin{pmatrix} m_{1,1} & 0 \\ 0 & m_{2,2} \end{pmatrix},$$

where

$$\begin{aligned} m_{1,1} &= \max\left\{1, K_1\left(\left|\frac{b_1 v_4}{f_1}\right| \exp\left(-\frac{|b_1 v_4|\, x}{f_1 \sigma}\right) + \left|\frac{b_1 v_2}{f_1}\right| \exp\left(-\frac{|b_1 v_2|\,(1-x)}{f_1 \sigma}\right)\right)\right\}, \\ m_{2,2} &= \max\left\{1, K_2\left(\left|\frac{b_2 v_1}{f_2}\right| \exp\left(-\frac{|b_2 v_1|\, y}{f_2 \sigma}\right) + \left|\frac{b_2 v_3}{f_2}\right| \exp\left(-\frac{|b_2 v_3|\,(1-y)}{f_2 \sigma}\right)\right)\right\}, \end{aligned}$$

with $K_1 = K_2 = 0.4$ chosen based on numerical experience, $\sigma = 2.5$ depends on the numerical method, and $b_1 = q_1 = 2 + 3y$, $b_2 = q_1 = 3 + 2x^2$, $f_1 = 1 + x$, $f_2 = 1 + y$ are taken from the physical problem (8.9). An example of a mesh generated with $\varepsilon = 10^{-2}$

and $N = 32$ is shown in Figure 8.7.

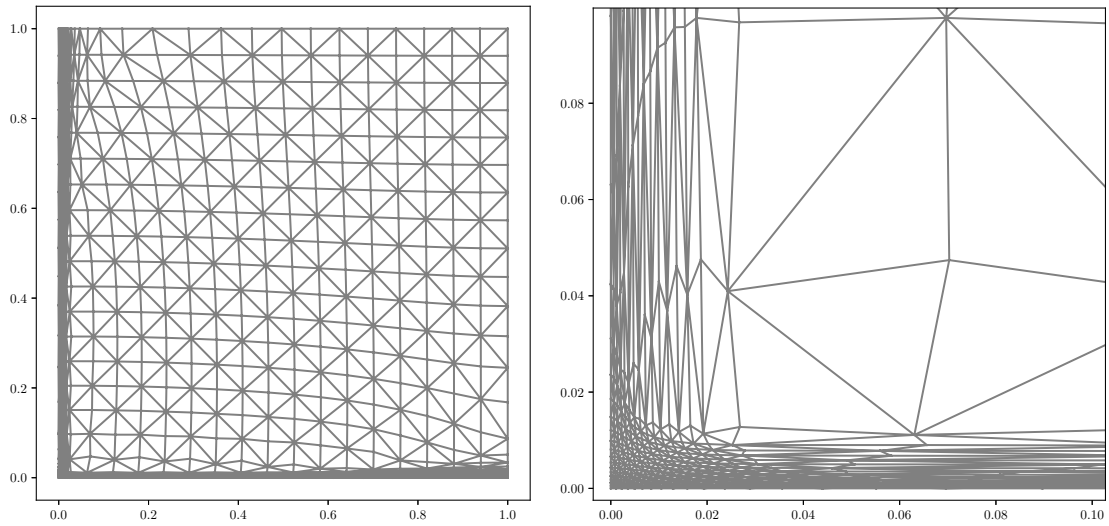


Figure 8.7: Mesh generated for (8.9) using Algorithm 7.1 with $\varepsilon = 10^{-2}$ and $N = 32$

Error estimates, $\|E_h\|_{\mathrm{E}}$, (2.12), in the numerical solution to (8.9) computed on meshes generated using Algorithm 7.1 are presented in Table 8.5. One observes linear convergence which is consistent with bounds for convection-diffusion equations solved on two-dimensional Bakhvalov meshes proven in [Roos and Schopf, 2012]. For comparison purposes we also present the error estimates in the numerical solution to (8.9) computed on tensor-product Bakhvalov meshes, in Table 8.6. We can see that, for smaller values of ε, the results for Algorithm 7.1 outperform the results for the Bakhvalov meshes by about 30%.

Table 8.5: $\|E_h\|_{\mathrm{E}}$ for (8.9) using Algorithm 7.1

ε/N	32	64	128	256	512
1	1.897e-02	9.474e-03	4.731e-03	2.363e-03	1.181e-03
		1.00	1.00	1.00	1.00
10^{-2}	3.572e-02	1.812e-02	8.985e-03	4.489e-03	2.244e-03
		0.98	1.01	1.00	1.00
10^{-4}	3.724e-02	1.823e-02	9.107e-03	4.561e-03	2.271e-03
		1.03	1.00	1.00	1.01
10^{-6}	3.707e-02	1.831e-02	9.142e-03	4.565e-03	2.277e-03
		1.02	1.00	1.00	1.00
10^{-8}	3.721e-02	1.830e-02	9.146e-03	4.569e-03	2.279e-03
		1.02	1.00	1.00	1.00

Table 8.6: $\|E_h\|_{\mathrm{E}}$ for (8.9) computed on tensor product classic Bakhvalov meshes

ε/N	32	64	128	256	512
1	2.115e-02	1.059e-02	5.299e-03	2.650e-03	1.325e-03
		1.00	1.00	1.00	1.00
10^{-2}	4.973e-02	2.489e-02	1.245e-02	6.227e-03	3.114e-03
		1.00	1.00	1.00	1.00
10^{-4}	5.100e-02	2.551e-02	1.276e-02	6.383e-03	3.192e-03
		1.00	1.00	1.00	1.00
10^{-6}	5.102e-02	2.552e-02	1.277e-02	6.385e-03	3.193e-03
		1.00	1.00	1.00	1.00
10^{-8}	5.102e-02	2.552e-02	1.277e-02	6.385e-03	3.193e-03
		1.00	1.00	1.00	1.00

8.4 Conclusion

In this chapter, we show that meshes generated using the MPDE methods are suitable for both a time-dependent one-dimensional reaction-diffusion and time-dependent convection-diffusion equation. Our numerical results show that the solutions generated on these meshes are robust and outperform those generated on classic Bakhvalov meshes. We have not optimised the iteration counts for the time-dependent problems and include this in further work. Also, we show for the Oseen equation that meshes generated using the MPDE method are competitive with tensor-product Bakhvalov meshes, while not requiring *a priori* information.

As noted by John,

> Oseen equations, which are linear equations, show up as an auxiliary problem in many numerical approaches for solving Navier-Stokes equations [John, 2016, Chap. 5].

Therefore, we feel confident that the MPDE method is suitable to generate meshes for the solution of the Navier-Stokes equations.

Chapter 9

Conclusion

9.1 Summary of thesis

Our goal has been to design fitted-mesh methods for singularly perturbed problems that are more general than those currently in the literature, and to implement these methods in a robust, efficient, and reproducible manner. To achieve this goal we have designed and implemented novel mesh generating algorithms based around the idea of Moving Mesh PDEs. We have presented two major variants of these algorithms: one which, like, the popular meshes of Shishkin and Bakhvalov, are built around detailed *a priori* estimates for the solution's derivatives, and a second which estimates those derivatives in an *a posteriori* way. We have demonstrated that these meshes are layer-resolving, and yield parameter robust solutions for a variety of steady-state and time-dependent SPDEs with solutions that exhibit boundary layers.

In Chapter 4 we present novel algorithms based on MPDEs that generate Bakhvalov-style meshes for one-dimensional SPDEs. We took inspiration from the MMPDE methods of [Huang et al., 1994b] and a method to generate the graded meshes of [Bakhvalov, 1969] using the equidistribution principle presented in [Linß, 2010]. The MPDE is a nonlinear problem and converges slowly for larger values of N, we solve this by combining the MPDE method with uniform h-refinement, then the number of iterations only depend very weakly on N and is independent of the perturbation factor, ε. To verify the MPDE method we demonstrate that solutions computed on these meshes are indistinguishable from those generated on traditional Bakhvalov meshes for one-dimensional problems. Like classic Bakhvalov meshes we use *a priori* information about the SPDEs, so this is not an adaptive method. The specific information concerns the location of layers and sharp bounds on the derivatives of the solution to the SPDEs.

In Chapter 5 we further develop the formulation of the mesh density function presented in Chapter 4, so it is based on *a posteriori* information about the derivatives of the nu-

merical solutions to the related SPDE near the boundaries of the domain. Therefore, no *a priori* information about the location or width of the boundary layers in the solution to the SPDE is required. The numerical solutions to the SPDEs calculated on these meshes are robust and the error estimates are similar in magnitude to those for SPDEs computed on meshes generated using the method presented in Chapter 4 using *a priori* information.

In Chapter 6 we extend the algorithms and methods presented in Chapter 4 to generate meshes for two-dimensional SPDEs. We present results for two two-dimensional reaction-diffusion problems, one posed on an irregular domain and one with spatially varying diffusion, rather than simple two-dimensional problems. Therefore, the meshes generated are not tensor-product in nature. We demonstrate that numerical solutions of these SPDEs computed on the resulting meshes are superior to those computed on tensor-product Bakhvalov meshes.

In Chapter 7 we combine elements of Chapters 5 and 6 to generate two-dimensional meshes for use when solving SPDEs numerically. In particular, the algorithm depends on *a posteriori* information about the derivatives of the numerical solution to the related SPDE. The method relies on the knowledge that the solution to the SPDE has boundary layers, but not the number, location or width of these layers. Again, we chose to investigate two two-dimensional reaction-diffusion problems, one posed on an irregular domain and one with spatially varying diffusion. The solutions computed on the meshes generated are comparable to those computed in Chapter 6, and therefore, are more accurate than those solved on tensor-product grids.

Our interest is in the application of the methods to challenging problems that arise in computational modelling, e.g., in computational fluid dynamics. In Section 8.2, we extend the *a posteriori* method of Chapter 5, to compute numerical solutions to time-dependent reaction-diffusion and convection-diffusion equations. The solutions to the problems investigated have boundary layers whose widths vary in time. The numerical solutions computed on these meshes are more accurate than those generated on Bakhvalov meshes. In Section 8.3, we apply the *a posteriori* method of Chapter 7 to the two-dimensional Oseen equations. Again, the solutions are more accurate than those generated on tensor-product Bakhvalov meshes.

9.2 Further work and open questions

The methods introduced in this thesis are quite novel in the field of SPDEs. Therefore, naturally, many related open problems would be worthy of further investigation. Here we list a selection of these that we think are particularly interesting.

How can the MPDE approach be applied to generate meshes to solve SPDEs on more complicated domains?

Most of the two-dimensional problems considered here are for problems posed on the unit square. The only exceptions to this are in Sections 6.2.3 and 7.4.1, where we have shown that the algorithms are suitable for generating appropriate meshes for SPDEs on an irregular domain. Part of the challenge to applying the MPDE approach on more complicated domains is the lack of *a priori* analysis to motivate fitted meshes. However, there is some work in the literature in this direction, for example, in [Andreev and Kopteva, 2008] robust convergence is proven for a two-dimensional reaction-diffusion equation using a finite-difference method on a Shishkin mesh on an L-shaped domain. It is natural to consider how Algorithms 6.1 and 7.1 could be successfully adapted to this scenario. Noted challenges would include meshing this non-convex domain. One possible approach is to follow the example in the `MMPDELAB` software suite [Huang, 2020]. It includes a sample MATLAB script for application of an MMPDE method to meshing an L-shaped domain, though not for a singularly perturbed problem. As noted in Remark 6.2.1, the formulations we have used in two dimensions may not extend so readily to non-convex domains [Huang and Russell, 2011, §6.1.1.], and that an alternative formulation in terms of an inverse coordinate mapping may be more appropriate. Alternatively, domain decompositions methods, detailed in [Quarteroni and Valli, 1999], may also be applicable if the resulting meshes can be successfully merged.

In a series of papers, Kopteva has investigated the solution of a semi-linear reaction-diffusion problem on a smooth, irregular domain (e.g., [Kopteva, 2007]). The computational method analysed in [Kopteva and Pickett, 2012] may be a promising starting point. They apply a domain decomposition method, with a uniform Delaunay triangulation on the interior, and a layer-adapted mesh on the annulus containing the boundary layer, which is then mapped to a regular rectangular domain with periodic boundary conditions. We expect that Algorithms 6.1 and 7.1 should perform well for this example by transforming the SPDE posed on the annulus to one posed on a rectangular domain with periodic boundary conditions.

In [Hegarty and O'Riordan, 2020] the Hemker problem is posed on a domain that is a composite of an annulus and rectangle. For a workshop dedicated to Pieter W. Hemker, we solved this problem on an annulus to generate an animation of its numerical solution. The mesh was generated by mapping a Bakhvalov mesh from a unit square to an annulus and applying periodic boundary conditions where required. Given that this is considered an important test problem in the literature, investigating if our MPDE approach could be usefully applied to this problem seems worthwhile.

How can the algorithms be developed to make them appropriate for problems with interior layers?

In this thesis, we have applied the MPDE methods only to problems with boundary layers but it is natural to consider how the method could be adapted to problems with interior layers. The test problems introduced in [Madden and Stynes, 1997] would be a good starting point since they feature interior layers whose location is known, *a priori*. The Shishkin meshes used in that paper are M-uniform: the nature of the layers are such that fitted meshes may be mapped to tensor product grids. We have had some success with extending the MPDE method of Chapter 6 to these problems, but further investigation is needed to determine the robustness of the method and to ensure that mesh tangling cannot occur.

As our meshes are M-uniform, some technical issues could arise when generating MPDE meshes for SPDEs for which a suitable mesh is not M-uniform, for example, where a layer enters and leaves the domain at adjacent boundaries as in [Stynes, 2005, Example 6.4].

In addition, it would be interesting to determine if the derivative of the numerical solutions to SPDEs could be used to automatically identify the location of interior layers to drive an *a posteriori* MPDE method.

How can one extend the algorithms presented to generate layer-adapted meshes for three-dimensional problems?

It is natural to consider how the MPDE mesh generating algorithms could be extended to construct layer-adapted meshes for problems in three dimensions. Some of the necessary analysis is already in place. In particular, the derivative bounds, on which our approach relies, are available (see, e.g., [Shishkin and Shishkina, 2009, §3.2] and, especially, [Russell and Madden, 2020] for the problem on the unit cube). No doubt, extending our MPDE methods from two to three dimensions is not straightforward, but recent work in this direction (see, e.g., [Duan and Tang, 2021, Huang and Kamenski, 2018]) suggests it is feasible. One challenge, for example, is that the MPDEs themselves would be three-dimensional, and so the computational cost of their solution could be prohibitive. Nonetheless, there is some scope for optimism. For example, when we extended the one-dimensional algorithms to two-dimensions, the number of iterations required did not increase. Hopefully, this would be the case for three-dimensional problems.

Furthermore, since the MPDE is solved on a uniform mesh there may be scope for solving them with specialist fast solvers, such as sparse grid techniques, for example, that of [Russell and Madden, 2017], could be a fitting option. An alternative approach, as presented in [Huang, 2001], but not investigated in this thesis, is to solve the SPDE on a mesh obtained by interpolating a coarser moving mesh.

Are the MPDE methods developed suitable for convection-diffusion equations?

Most of the focus in this thesis has been on the solution of reaction-diffusion equations, other than in Chapter 8 where we briefly investigate generating meshes suitable for a time-dependent convection-diffusion problem and the Oseen equation. However, further tuning is needed to optimise the algorithms to reduce the number of iterations used to generate the meshes.

A particular difficulty one may encounter, for the methods that depend on derivatives estimates, is that oscillations present in numerical solutions to convection-diffusion problems on uniform meshes lead to inaccurate estimation of derivatives. One possible solution is to employ stabilised methods when computing the initial solutions.

Can our algorithms be adapted to automatically generate meshes suitable for the numerical solution of the Navier-Stokes equations?

Solution of the full Navier-Stokes equations was beyond the scope of this research project. However, we took some steps towards this goal. In Chapter 8, we demonstrated that the *a posteriori* MPDE method presented in this thesis is suitable to generate meshes for the solution of some problems that arise in fluid dynamics. One of these problems is the Oseen equations which are often solved at each time-step if using an iterative method when solving the Navier-Stokes equations. In Section 8.3, we solve the Oseen equations where the pressure term is known. When the pressure term is unknown, numerical solutions to the physical differential equation may be unstable (perhaps related to the absence of a unique solution). In addition, we have preliminary results which show that our proposed approach is effective when solving (linear) convection-diffusion problems. However, these results are not yet ready for dissemination: we have obtained good results in some instances, but not for a wide range of ε.

How does our MPDE compare, in practice, with other MMPDE formulations in the literature?

In [Huang and Russell, 2011, §2.3.1] they state that "There are numerous ways of formulating MMPDEs (e.g., see [Huang et al., 1994b, Huang et al., 1994a]), and indeed, it can be unclear which MMPDEs to prefer without careful analysis and computational comparison". The algorithms presented in this thesis rely on MPDEs framed around the Poisson equation. As well as the MMPDEs of [Huang et al., 1994b], listed in Section 2.7, other possibilities on which to base an MPDE exist, including the Monge-Ampère equation used in [Weller et al., 2016]. It would be valuable to make a systematic investigation of our MPDE methods in the settings of these other MMPDEs, with a view to comparing them for efficiency and parameter robustness.

Other interesting problems

In Chapter 4, we have shown that solving the MPDE is equivalent to equidistribution and thus to the classical Bakhvalov mesh. This essentially shows the method is provably robust. However, how do we know that the method for solving the MPDE converges? For equidistribution, this is done by [Kopteva and Stynes, 2001] with fixed-point iteration. Since we observed that the method converged to the stopping criterion monotonically for (4.9) then perhaps fixed-point analysis may also be applicable here.

As shown in Section 4.5.1, the Bakhvalov mesh is a solution to the MPDE (4.13). However, the MPDE is not solved exactly, and its numerical solution contains both a discretization and iteration error. As shown in Section 4.5.2 this is not a practical issue. However, can one prove that? For example, can it be determined by how much a Bakhvalov mesh can be perturbed by the fact that the MPDE is solved numerically, rather than exactly? This also has a practical consideration: can the MPDE be solved at a lower resolution without impacting the accuracy of the approximation solution to the physical problem?

Chapter 5 depends on accurate derivative estimates. In Figure 5.1, one observes that the derivative of the numerical solution is a good approximation of the derivative of the related reaction-diffusion problem. However, how accurate are the estimates we use and could they be improved? In [O'Riordan and Pickett, 2019], they prove that the numerical (first) derivatives of upwind finite-difference solutions to reaction-convection diffusion equations, robustly converge to the true derivatives on a Shishkin mesh. Can their methods be used to prove bounds on the derivatives for SPDEs solved on meshes generated using MPDEs? Finally, there may be some practical benefit to estimating these derivatives using higher-order schemes.

Appendices

Appendix A

FEniCS code to solve (4.9) on a uniform mesh

```
## 1DRD_FEM_uniform_mesh.py
## FEniCS version: 2019.2.0.dev0
"""
Solve the singularly perturbed differential equation (SPDE)
 -epsilon^2 u" + u =  f = 1-x, for x in (0,1)  with  u(0) = u(1) = 0
on a uniform mesh.
This code is part of
Generating layer-adapted meshes using mesh partial differential equations,
by Róisín Hill and Niall Madden. DOI: 10.17605/OSF.IO/DPEXH
Contact: Róisín Hill <Roisin.Hill@NUIGalway.ie>
"""

from fenics import *

# Define problem parameters
epsilon = 1E-2           # perturbation factor
N = 128                  # mesh intervals

# Create mesh and function space with linear element on that mesh
mesh = UnitIntervalMesh(N)
V = FunctionSpace(mesh, 'P', 1)

u = TrialFunction(V)     # Trial function on V
v = TestFunction(V)      # Test function on V
uN = Function(V)         # Numerical solution in V

# Define the Dirichlet boundary conditions
g = Expression('x[0]*(1-x[0])', degree=2)
bc = DirichletBC(V, g, 'on_boundary')

# Define right hand side of the SPDE, and the weak form
f = Expression ('1-x[0]', degree=2)
a = epsilon*epsilon*inner(grad(u), grad(v))*dx + inner(u,v)*dx
L = f*v*dx
solve(a==L, uN, bc)      # Solve, applying the boundary conditions strongly
```

Appendix B

A note on quadrature

FEniCS uses Gaussian quadrature as its default method, and so approximates integrals by sampling the integrands at certain points in the interior of each interval. For a boundary layer problem, the maximum value of the mesh density function will occur at the boundary and, when $\varepsilon \ll 1$, it decays rapidly away from that boundary. Consequently, one should use a quadrature method, such as Gauss-Lobatto, that includes quadrature points at mesh points. To force FEniCS to select Gauss-Lobatto quadrature, we use a "monkey patch" which we incorporate in our Python code when generating one-dimensional layer-adapted meshes.

Appendix C

FEniCS code to generate a 1D layer-adapted mesh using an MPDE and h-refinement

This FEniCS code generates a one-dimensional layer-adapted mesh, similar to a Bakhvalov mesh, that is appropriate for an SPDE whose solution exhibits a layer near $x = 0$. A combination of solving an MPDE and h-refinement is used. Then (4.9) is solved on the resulting mesh.

```
## 1D_layer_adapted_mesh_MPDE_h_refinement.py
## FEniCS version: 2019.2.0.dev0
'''
Generate 1-sided Bakhvalov-type mesh, for a singularly perturbed ODE
whose solution has a layer near x = 0, by solving the MPDE
   -(rho(x) x'(xi))' = 0 for xi in (0,1) with x(0) = 0 and x(1) = 1,
using uniform h-refinement.

Note: selection of Gauss-Lobatto quadrature rule is executed in lines 22-30.

This code is part of
Generating layer-adapted meshes using mesh partial differential equations,
by Róisín Hill and Niall Madden. DOI: 10.17605/OSF.IO/DPEXH
Contact: Róisín Hill <Roisin.Hill@NUIGalway.ie>
'''

from fenics import *
import matplotlib.pyplot as plt
import math
import numpy as np
plt.rc('font',**{'family':'serif','serif':['DejaVu␣Sans']})
plt.rc('text', usetex=True)


# Change quadrature rule to Gauss Lobatto
from FIAT.reference_element import *
from FIAT.quadrature import *
```

```
from FIAT.quadrature_schemes import create_quadrature
def create_GaussLobatto_quadrature(ref_el, degree, scheme="default"):
    if degree < 3: degree = 3
    return GaussLobattoLegendreQuadratureLineRule(ref_el, degree)
import FIAT
FIAT.create_quadrature = create_GaussLobatto_quadrature

# Problem parameters
epsilon = 1E-2        # perturbation factor of the physical PDE
N = 1024              # mesh intervals in final mesh

# parameters for rho(x)
sigma = 2.5           # related to degree of elements
b = 0.99              # lower bound on reaction coefficient
q = 0.5               # proportion of mesh points in layer
K = q/(sigma*(1-q))

# Parameters for h-refinement steps
N_start = 4           # initial number of mesh intervals
step_index = 0        # initial step number
N_steps = 2**np.arange(round(math.log(N_start,2)),int(math.log(N,2)))

f = Expression('0.0', degree=2)    # Right-and side of MPDE
x_0 = Expression('x[0]', degree=2) # Initial solution

residual_norm_TOL = 0.4   # Stopping criterion tolerance
residual_norm = residual_norm_TOL+1

# Lists for meshes and computational function spaces
mesh_list = []
V_list = []
list_length = (N_steps.size+1)

for i in range(0,list_length):
    mesh_list.append("mesh%2d" %(i))
    V_list.append("V2D%2d" %(i))

# Computational function space parameters
def comp_space(N):
    meshc = UnitIntervalMesh(N)
    V = FunctionSpace(meshc, 'P', 1) # Function space with P1-elements
    v = TestFunction(V)              # Test function on V
    x = TrialFunction(V)             # Trial function on V
    return meshc, V, v, x

# FEM problem for the MPDE
def MPDE_FEM(rho, xN, x, v, f, meshz, V):
    bc = DirichletBC(V, x_0, 'on_boundary') # Dirichlet boundary condition
    a = rho(xN)*inner(grad(x),grad(v))*dx(meshz) # Left side of weak form
    L = f*v*dx                       # Right side of weak form
    xN = Function(V)                 # Numerical solution in V
    solve(a==L, xN, bc)              # Solve, applying the BCs strongly
    return xN

# Define rho(x) for MPDE
def rho(xN):
    return conditional((K*(b/epsilon)*exp(-b*xN/(sigma*epsilon)))>1.0, \
```

```
                        K*(b/epsilon)*exp(-b*xN/(sigma*epsilon)), 1.0)

# The norm of the residual to use as stopping criterion
def calc_residual(rho, V, meshN):
    residual = assemble((rho(xN)*xN.dx(0)*v.dx(0))*dx) # residual of xN
    bcr = DirichletBC(V, '0.0', 'on_boundary')
    bcr.apply(residual)                                # apply Dirichlet BCs
    residual_func = Function(V)
    residual_func.vector()[:] = residual.get_local() # Residual fn on V
    residual_norm = norm(residual_func, 'L2', meshN) # L2-norm of residual
    return residual_norm

# Initial function space parameters
mesh_list[0], V_list[0], v, x = comp_space(N_start)
xN = interpolate(x_0, V_list[0])                       # Initial value for xN

# Iterate through uniform h-refinements
for N_step in N_steps:
    for i in range(0,3):    # Calculate solution three times on each mesh size
         xN = MPDE_FEM(rho, xN, x, v, f, mesh_list[step_index], \
                           V_list[step_index])

    step_index = round(math.log(N_step*2/N_start,2))
    # Generate computational function space on the finer mesh
    mesh_list[step_index], V_list[step_index], v, x = comp_space(N_step*2)

    # Interpolate the solution onto the new computational function space
    xN = interpolate(xN, V_list[step_index])

# Calculate solution and L2 norm of the residual on final function space
while residual_norm > residual_norm_TOL:
    xN = MPDE_FEM(rho, xN, x, v, f, mesh_list[step_index], V_list[step_index])
    residual_norm = calc_residual(rho, V_list[step_index], \
                                  mesh_list[step_index])

# Generate the physical mesh
meshp = UnitIntervalMesh(N)
meshp.coordinates()[:,0]  = xN.compute_vertex_values()[:]

# DEFINE REACTION-DIFFUSION PROBLEM TO SOLVE ON MESH GENERATED
# Solve -epsilon^2 u" + u = 1-x for x in (0,1) with u(0) = u(1) = 0
# Physical function space parameters
Vp = FunctionSpace(meshp, 'P', 1)            # Function space with P1-elements
vp = TestFunction(Vp)                        # Test function on V
u = TrialFunction(Vp)                        # Trial function on V
gp = Expression('x[0]*(1-x[0])', degree=2)   # Boundary values
bcp = DirichletBC(Vp, gp, 'on_boundary')     # Dirichlet boundary conditions
fp = Expression('1.0-x[0]', degree=2)        # RHS of PDE

# FEM problem for the reaction-diffusion equation
a = epsilon*epsilon*dot(grad(u), grad(vp))*dx + u*vp*dx
L = fp*vp*dx
uN = Function(Vp)
solve(a==L, uN, bcp)
fig1 = plt.figure("Solution")
plt.plot(meshp.coordinates()[:], uN.compute_vertex_values()[:], label = r'$u(x)$')
plt.xlabel(r'$x$',fontsize = 18)
```

```
plt.ylim(-0.05, 1.0)
plt.ylabel(r'$u$',rotation =0,fontsize = 18)
plt.tick_params( labelsize=14)
plt.legend(fontsize = 18)
fig1.savefig('Sol_1DRD_model.eps', bbox_inches = 'tight', pad_inches = 0.01 ,
    format='eps')
plt.show()
```

Appendix D

FEniCS code to generate a 2D layer-adapted mesh using an MPDE

This FEniCS code generates a two-dimensional layer-adapted mesh for an SPDE whose diffusion varies spatially and has layer regions near $x = 1$ and $y = 1$ using a combination of MPDE and h-refinement.

```
## 2D_layer_adapted_mesh_MPDE_h_refinement.py
## FEniCS version: 2019.2.0.dev0
'''
Generate a non-tensor product layer-adapted mesh, for a singularly
perturbed PDE whose solution has layers that vary spatially near x = 1
and y = 1, and meet at (1,1), by solving the MPDE
    -grad(M(x) grad(x(xi1, xi2))) = (0,0)^T for (xi1,xi2) in (0,1)^2
where x is a vector, with
    x(0,xi2) = (0,x_xi2=0), x(1,xi2) = (1,x_xi2=0),
    x(xi1,0) = (x_xi1=0,0), x(xi1,1) = (x_xi1=0,1),
and using uniform h-refinements.

This code is part of
Generating layer-adapted meshes using mesh partial differential equations,
by Róisín Hill and Niall Madden. DOI: 10.17605/OSF.IO/DPEXH
Contact: Róisín Hill <Roisin.Hill@NUIGalway.ie>
'''

from fenics import *
import math
import numpy as np
import matplotlib.pyplot as plt

# Problem parameters
epsilon = 1E-2      # perturbation factor of the physical PDE
N = 32              # N+1 mesh points in each direction in final mesh

# Parameters for h refinement steps
N_start = 4         # initial mesh size
step_index = 0      # initial step number
N_steps = 2**np.arange(round(math.log(N_start,2)),int(math.log(N,2)))
```

```
# Parameters for M(x)
sigma = 2.5 # dependant on method, often degree of elements +1 or +1.5
b = 0.99    # minimum coefficient of reaction/convection term as appropriate
q = 0.5     # proportion of mesh points in layer
K = q/(sigma*(1-q))

f = Expression(('0.0','0.0'), degree=2) # Right-hand side of MPDE

# Lists for meshes and computational function spaces
mesh_list = []
V_list = []
list_length = int(math.log(N/N_start,2)+1)
for i in range(0,list_length):
    mesh_list.append("mesh%2d" %(i))
    V_list.append("V2D%2d" %(i))

# generate computational function space
def comp_space(N):
    mesh = UnitSquareMesh(N, N, diagonal='crossed') # uniform mesh on (0,1)^2
    V = VectorFunctionSpace(mesh, 'P', 1)   # Function space V with P1 elements
    v = TestFunction(V)                     # Test function on V
    x = TrialFunction(V)                    # Trial function on V
    return mesh, V, v, x

# Define boundaries
TOL = 1E-14
def left_boundary( x, on_boundary):
    return  abs(x[0]) < TOL
def right_boundary(x, on_boundary):
    return abs(1-x[0]) < TOL
def bottom_boundary( x, on_boundary):
    return abs(x[1]) < TOL
def top_boundary( x, on_boundary):
    return abs(1-x[1]) < TOL

# Define Dirichlet boundary conditions
def boundary_conditions(V):
    bcl = DirichletBC(V.sub(0), '0.0', left_boundary)
    bcr = DirichletBC(V.sub(0), '1.0', right_boundary)
    bcb = DirichletBC(V.sub(1), '0.0', bottom_boundary)
    bct = DirichletBC(V.sub(1), '1.0', top_boundary)
    bcs = [bcl, bcr, bcb, bct]
    return bcs

# Define the matrix M(x) for MPDE
def M(xN):
    epsilon_x = Expression('epsilon*(1+2*xi2)*(1+2*xi2)',\
        epsilon=epsilon, xi2=xN.sub(1), degree=2)
    epsilon_y = Expression('epsilon*(3-2*xi1)*(3-2*xi1)',\
        epsilon=epsilon, xi1=xN.sub(0), degree=2)
    M = Expression((('K*(b/epsilon_x)*exp(-b*(1-xi1)/(sigma*epsilon_x))>1␣?␣K*(b/
        epsilon_x)*exp(-b*(1-xi1)/(sigma*epsilon_x))␣:␣1','0'),\
        ('0','K*(b/epsilon_y)*exp(-b*(1-xi2)/(sigma*epsilon_y))>1␣?␣K*(b/epsilon_y)
            *exp(-b*(1-xi2)/(sigma*epsilon_y))␣:1')),\
        K=K, b=b, sigma=sigma, epsilon_x=epsilon_x, epsilon_y=epsilon_y,\
                    xi1=xN.sub(0), xi2=xN.sub(1), degree=4)
```

```
    return M

# Define the FEM problem for the MPDE
def MPDE_FEM(M, xN, x, v, f, bcs, meshz, V):
    a = inner(M(xN)*grad(x), grad(v))*dx(meshz) # Left side of weak form
    L = dot(f,v)*dx                 # Right side of weak form
    xN = Function(V)                # Numerical solution in V
    solve(a==L, xN, bcs)            # Solve, applying the DirichletBC strongly
    return xN

# Initial function space parameters
mesh_list[0], V_list[0], v, x = comp_space(N_start)

# Set intial value for xN: xN(xi1, xi2) = (xi1, xi2)
a = inner(grad(x),grad(v))*dx
L = dot(f,v)*dx
xN = Function(V_list[0])
solve(a==L, xN, boundary_conditions(V_list[0]))

# Iterate through uniform h-refinements
for N_step in N_steps:
    for i in range (0,4): # Solve MPDE 4 times on each mesh size
        xN = MPDE_FEM(M, xN, x, v, f, boundary_conditions(V_list[step_index]),
            mesh_list[step_index], V_list[step_index])

    step_index = round(math.log(N_step*2/N_start,2))
    # Generate computational function space on the finer mesh
    mesh_list[step_index], V_list[step_index], v, x = comp_space(N_step*2)

    # Interpolate solution onto the new computational function space
    xN = interpolate(xN,V_list[step_index])

# Calculate solution in final function space
for i in range(0,5):
    xN = MPDE_FEM(M, xN, x, v, f, boundary_conditions(V_list[step_index]),
        mesh_list[step_index], V_list[step_index])

# Generate the physical mesh
xiX, xiY = xN.split(True)
meshp = UnitSquareMesh(N,N)
meshp.coordinates()[:,0] = xiX.compute_vertex_values()[0:(N+1)*(N+1)]
meshp.coordinates()[:,1] = xiY.compute_vertex_values()[0:(N+1)*(N+1)]

plot(meshp)
plt.show()
```

Appendix E

Effects of quadrature in evaluating equidistribution

In Table E.1 we present the values of Ψ, such that,

$$\max\left\{\int_{x_{i-1}}^{x_i}\rho(x)dx \text{ for } i=1,\ldots,N\right\}=\frac{\Psi}{N}\int_0^1\rho(x)dx, \tag{E.1}$$

where the MPDE (4.13) has been solved with the mesh density function

$$\rho(x)=\max\left\{1,K\frac{\beta}{\varepsilon}\exp\left(\frac{-\beta x}{\sigma\varepsilon}\right)\right\}, \tag{E.2}$$

The resulting meshes are suitable for solving a one-dimensional reaction-diffusion equation whose solution has one layer near $x=0$. Here, the integrals of ρ are calculated using Gaussian quadrature on linear elements. The results in Table E.1 should be compared with Table 4.5, where integrals are calculated exactly. In particular, note that the data in Table 4.5 show $1\leq\Psi\leq 1.76$, whereas Table E.1 shows $1\leq\Psi\leq 15.26$. We can conclude that the effects of quadrature are significant but, nonetheless, the results can be relied upon to verify robustness. This is useful in the investigations of two-dimensional problems in Section 6.3.1, where using quadrature is the only viable approach.

Table E.1: Values of Ψ that satisfy (4.16) using the MPDE method (using quadrature)

ε/N	32	64	128	256	512	1024
1	1.00	1.00	1.00	1.00	1.00	1.00
10^{-1}	1.00	1.00	1.00	1.00	1.00	1.00
10^{-2}	1.34	1.09	1.04	1.04	1.01	1.01
10^{-3}	3.67	3.62	1.79	1.34	1.06	1.06
10^{-4}	4.13	6.78	4.92	5.39	2.56	2.04
10^{-5}	3.52	6.04	4.97	7.27	7.83	3.75
10^{-6}	3.85	3.84	6.94	5.20	7.71	8.23
10^{-7}	4.89	5.46	5.51	4.79	8.27	15.26
10^{-8}	5.12	5.95	6.39	6.06	15.15	7.67

Bibliography

[Alnaes et al., 2009] Alnaes, M., Logg, A., Mardal, K.-A., Skavhaug, O., and , H. L. (2009). Unified framework for finite element assembly. *International Journal of Computational Science and Engineering*, 4(4):231–244.

[Alnæs, 2012] Alnæs, M. S. (2012). *UFL: a Finite Element Form Language*, chapter 17. Springer.

[Alnæs et al., 2015] Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N. (2015). The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100).

[Alnæs et al., 2012] Alnæs, M. S., Logg, A., and Mardal, K.-A. (2012). *UFC: a Finite Element Code Generation Interface*, chapter 16. Springer.

[Alnæs et al., 2014] Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., and Wells, G. N. (2014). Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, 40(2).

[Andreev and Kopteva, 2008] Andreev, V. B. and Kopteva, N. (2008). Pointwise approximation of corner singularities for a singularly perturbed reaction-diffusion equation in an L-shaped domain. *Math. Comp.*, 77(264):2125–2139.

[Arndt et al., 2019] Arndt, D., Bangerth, W., Clevenger, T. C., Davydov, D., Fehling, M., Garcia-Sanchez, D., Harper, G., Heister, T., Heltai, L., Kronbichler, M., Kynch, R. M., Maier, M., Pelteret, J.-P., Turcksin, B., and Wells, D. (2019). The `deal.II` library, version 9.1. *J. Numer. Math.*, 27(4):203–213.

[Babuška and Rheinboldt, 1978] Babuška, I. and Rheinboldt, W. C. (1978). Error estimates for adaptive finite element computations. *SIAM J. Numer. Anal.*, 15(4):736–754.

[Bakhvalov, 1969] Bakhvalov, N. S. (1969). On the optimization of the methods for solving boundary value problems in the presence of a boundary layer. *Ž. Vyčisl. Mat i Mat. Fiz.*, 9:841–859.

[Beckett and Mackenzie, 2000] Beckett, G. and Mackenzie, J. A. (2000). Convergence analysis of finite difference approximations on equidistributed grids to a singularly perturbed boundary value problem. *Appl. Numer. Math.*, 35(2):87–109.

[Beckett and Mackenzie, 2001a] Beckett, G. and Mackenzie, J. A. (2001a). On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution. *J. Comput. Appl. Math.*, 131(1-2):381–405.

[Beckett and Mackenzie, 2001b] Beckett, G. and Mackenzie, J. A. (2001b). Uniformly convergent high order finite element solutions of a singularly perturbed reaction-diffusion equation using mesh equidistribution. *Appl. Numer. Math.*, 39(1):31–45.

[Beckett et al., 2001] Beckett, G., Mackenzie, J. A., Ramage, A., and Sloan, D. M. (2001). On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution. *J. Comput. Phys.*, 167(2):372–392.

[Brezzi and Russo, 1994] Brezzi, F. and Russo, A. (1994). Choosing bubbles for advection-diffusion problems. *Math. Models Methods Appl. Sci.*, 4(4):571–587.

[Browne et al., 2014] Browne, P. A., Budd, C. J., Piccolo, C., and Cullen, M. (2014). Fast three dimensional r-adaptive mesh redistribution. *J. Comput. Phys.*, 275:174–196.

[Cao et al., 2001] Cao, W., Huang, W., and Russell, R. D. (2001). An error indicator monitor function for an r-adaptive finite-element method. *J. Comput. Phys.*, 170(2):871–892.

[Chadha and Madden, 2016] Chadha, N. M. and Madden, N. (2016). An optimal time-stepping algorithm for unsteady advection-diffusion problems. *J. Comput. Appl. Math.*, 294:57–77.

[Clavero and Gracia, 2010] Clavero, C. and Gracia, J. L. (2010). On the uniform convergence of a finite difference scheme for time dependent singularly perturbed reaction-diffusion problems. *Appl. Math. Comput.*, 216(5):1478–1488.

[Clavero et al., 2005] Clavero, C., Gracia, J. L., and O'Riordan, E. (2005). A parameter robust numerical method for a two dimensional reaction-diffusion problem. *Math. Comp.*, 74(252):1743–1758.

[Crank and Nicolson, 1947] Crank, J. and Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Proc. Cambridge Philos. Soc.*, 43:50–67.

[Das, 2019] Das, P. (2019). An a posteriori based convergence analysis for a nonlinear singularly perturbed system of delay differential equations on an adaptive mesh. *Numer. Algorithms*, 81(2):465–487.

[de Boor, 1974] de Boor, C. (1974). Good approximation by splines with variable knots. II. In *Conference on the Numerical Solution of Differential Equations (Univ. Dundee, Dundee, 1973)*, pages 12–20. Lecture Notes in Math., Vol. 363. Springer.

[Duan and Tang, 2021] Duan, J. and Tang, H. (2021). Entropy stable adaptive moving mesh schemes for 2D and 3D special relativistic hydrodynamics. *J. Comput. Phys.*, 426:109949.

[Eriksson et al., 1996] Eriksson, K., Estep, D., Hansbo, P., and Johnson, C. (1996). *Computational Differential Equations*. Cambridge University Press, Cambridge.

[Farrell et al., 2000] Farrell, P. A., Hegarty, A. F., Miller, J. J. H., O'Riordan, E., and Shishkin, G. I. (2000). *Robust Computational Techniques for Boundary Layers*, volume 16 of *Applied Mathematics (Boca Raton)*. Chapman & Hall/CRC, Boca Raton, FL.

[Franz et al., 2018] Franz, S., Höhne, K., and Matthies, G. (2018). Grad-div stabilized discretizations on S-type meshes for the Oseen problem. *IMA J. Numer. Anal.*, 38(1):299–329.

[Gartland, 1988] Gartland, Jr., E. C. (1988). Graded-mesh difference schemes for singularly perturbed two-point boundary value problems. *Math. Comp.*, 51(184):631–657.

[Gockenbach, 2006] Gockenbach, M. S. (2006). *Understanding and Implementing the Finite Element Method*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[Grätsch and Bathe, 2005] Grätsch, T. and Bathe, K.-J. (2005). A posteriori error estimation techniques in practical finite element analysis. *Comput. & Structures*, 83(4-5):235–265.

[Grossmann and Roos, 2007] Grossmann, C. and Roos, H.-G. (2007). *Numerical Treatment of Partial Differential Equations*. Universitext. Springer, Berlin. Translated and revised from the 3rd (2005) German edition by Martin Stynes.

[Hecht, 2012] Hecht, F. (2012). New development in FreeFEM++. *J. Numer. Math.*, 20(3-4):251–265.

[Hegarty and O'Riordan, 2020] Hegarty, A. F. and O'Riordan, E. ([2020] ©2020). A numerical method for the Hemker problem. In *Boundary and interior layers, computational and asymptotic methods—BAIL 2018*, volume 135 of *Lect. Notes Comput. Sci. Eng.*, pages 97–111. Springer, Cham.

[Hill and Madden, 2021] Hill, R. and Madden, N. (2021). Generating layer-adapted meshes using mesh partial differential equations. *Numer. Math. Theory Methods Appl.*, 14(3):559–588.

[Huang, 2001] Huang, W. (2001). Practical aspects of formulation and solution of moving mesh partial differential equations. *J. Comput. Phys.*, 171(2):753–775.

[Huang, 2020] Huang, W. (2020). An introduction to MMPDElab. `https://arxiv.org/abs/1904.05535`.

[Huang and Kamenski, 2018] Huang, W. and Kamenski, L. (2018). On the mesh nonsingularity of the moving mesh PDE method. *Math. Comp.*, 87(312):1887–1911.

[Huang et al., 2010] Huang, W., Kamenski, L., and Lang, J. (2010). A new anisotropic mesh adaptation method based upon hierarchical a posteriori error estimates. *J. Comput. Phys.*, 229(6):2179–2198.

[Huang et al., 1994a] Huang, W., Ren, Y., and Russell, R. D. (1994a). Moving mesh methods based on moving mesh partial differential equations. *J. Comput. Phys.*, 113(2):279–290.

[Huang et al., 1994b] Huang, W., Ren, Y., and Russell, R. D. (1994b). Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. *SIAM J. Numer. Anal.*, 31(3):709–730Adaptive Moving Mesh Methods.

[Huang and Russell, 1998] Huang, W. and Russell, R. D. (1998). A high-dimensional moving mesh strategy. In *Proceedings of the International Centre for Mathematical Sciences Conference on Grid Adaptation in Computational PDEs: Theory and Applications (Edinburgh, 1996)*, volume 26, pages 63–76.

[Huang and Russell, 2011] Huang, W. and Russell, R. D. (2011). *Adaptive Moving Mesh Methods*, volume 174 of *Applied Mathematical Sciences*. Springer, New York.

[Huang and Sun, 2003] Huang, W. and Sun, W. (2003). Variational mesh adaptation. II. Error estimates and monitor functions. *J. Comput. Phys.*, 184(2):619–648.

[Jahandari et al., 2020] Jahandari, H., MacLachlan, S., Haynes, R. D., and Madden, N. (2020). Finite element modelling of geophysical electromagnetic data with goal-oriented *hr*-adaptivity. *Comput. Geosci.*, 24(3):1257–1283.

[John, 2016] John, V. (2016). *Finite element methods for incompressible flow problems*, volume 51 of *Springer Series in Computational Mathematics*. Springer, Cham.

[Kellogg et al., 2008] Kellogg, R. B., Linß, T., and Stynes, M. (2008). A finite difference method on layer-adapted meshes for an elliptic reaction-diffusion system in two dimensions. *Math. Comp.*, 77(264):2085–2096.

[Kirby, 2004] Kirby, R. C. (2004). Algorithm 839: FIAT, a new paradigm for computing finite element basis functions. *ACM Trans. Math. Softw.*, 30(4):502–516.

[Kirby, 2012] Kirby, R. C. (2012). *FIAT: Numerical Construction of Finite Element Basis Functions*, chapter 13. Springer.

[Kirby and Logg, 2006] Kirby, R. C. and Logg, A. (2006). A compiler for variational forms. *ACM Trans. Math. Softw.*, 32(3):417–444.

[Kopteva, 2007] Kopteva, N. (2007). Maximum norm error analysis of a 2D singularly perturbed semilinear reaction-diffusion problem. *Math. Comp.*, 76(258):631–646.

[Kopteva and Pickett, 2012] Kopteva, N. and Pickett, M. (2012). A second-order overlapping Schwarz method for a 2D singularly perturbed semilinear reaction-diffusion problem. *Math. Comp.*, 81(277):81–105.

[Kopteva and Stynes, 2001] Kopteva, N. and Stynes, M. (2001). A robust adaptive method for a quasi-linear one-dimensional convection-diffusion problem. *SIAM J. Numer. Anal.*, 39(4):1446–1467.

[Kurganov et al., 2021] Kurganov, A., Qu, Z., Rozanova, O. S., and Wu, T. (2021). Adaptive moving mesh central-upwind schemes for hyperbolic system of PDEs: applications to compressible Euler equations and granular hydrodynamics. *Commun. Appl. Math. Comput.*, 3(3):445–479.

[Langtangen and Logg, 2016] Langtangen, H. P. and Logg, A. (2016). *Solving PDEs in Minutes–The FEniCS Tutorial Volume I.* Springer.

[Lin and Stynes, 2012] Lin, R. and Stynes, M. (2012). A balanced finite element method for singularly perturbed reaction-diffusion problems. *SIAM J. Numer. Anal.*, 50(5):2729–2743.

[Linß, 2010] Linß, T. (2010). *Layer-Adapted Meshes for Reaction-Convection-Diffusion Problems*, volume 1985 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin.

[Linß and Madden, 2004] Linß, T. and Madden, N. (2004). A finite element analysis of a coupled system of singularly perturbed reaction-diffusion equations. *Appl. Math. Comput.*, 148(3):869–880.

[Linß and Madden, 2009] Linß, T. and Madden, N. (2009). Layer-adapted meshes for a linear system of coupled singularly perturbed reaction-diffusion problems. *IMA J. Numer. Anal.*, 29(1):109–125.

[Liu et al., 2009] Liu, F., Madden, N., Stynes, M., and Zhou, A. (2009). A two-scale sparse grid method for a singularly perturbed reaction-diffusion problem in two dimensions. *IMA J. Numer. Anal.*, 29(4):986–1007.

[Liu and Chen, 2017] Liu, L.-B. and Chen, Y. (2017). A-posteriori error estimation in maximum norm for a strongly coupled system of two singularly perturbed convection-diffusion problems. *J. Comput. Appl. Math.*, 313:152–167.

[Liu et al., 2020a] Liu, L.-B., Liang, Y., Zhang, J., and Bao, X. (2020a). A robust adaptive grid method for singularly perturbed Burger-Huxley equations. *Electron. Res. Arch.*, 28(4):1439–1457.

[Liu et al., 2020b] Liu, L.-B., Long, G., and Cen, Z. (2020b). A robust adaptive grid method for a nonlinear singularly perturbed differential equation with integral boundary condition. *Numer. Algorithms*, 83(2):719–739.

[Logg et al., 2012a] Logg, A., B., O. K., E., R. M., and Wells, G. N. (2012a). *FFC: the FEniCS form compiler*, chapter 11. Springer.

[Logg et al., 2012b] Logg, A., Mardal, K.-A., and Wells, G. N., editors (2012b). *Automated Solution of Differential Equations by the Finite Element Method*, volume 84 of *Lecture Notes in Computational Science and Engineering*. Springer, Heidelberg. The FEniCS book.

[Logg and Wells, 2010] Logg, A. and Wells, G. N. (2010). Dolfin: Automated finite element computing. *ACM Trans. Math. Softw.*, 37(2).

[Logg et al., 2012c] Logg, A., Wells, G. N., and Hake, J. (2012c). *DOLFIN: a C++/Python Finite Element Library*, chapter 10. Springer.

[Long et al., 2021] Long, G., Liu, L.-B., and Huang, Z. (2021). Richardson extrapolation method on an adaptive grid for singularly perturbed Volterra integro-differential equations. *Numer. Funct. Anal. Optim.*, 42(7):739–757.

[Mackenzie, 1999] Mackenzie, J. (1999). Uniform convergence analysis of an upwind finite-difference approximation of a convection-diffusion boundary value problem on an adaptive grid. *IMA J. Numer. Anal.*, 19(2):233–249.

[Mackenzie, 1998] Mackenzie, J. A. (1998). The efficient generation of simple two-dimensional adaptive grids. *SIAM J. Sci. Comput.*, 19(4):1340–1365.

[Madden and Stynes, 1997] Madden, N. and Stynes, M. (1997). Efficient generation of oriented meshes for solving convection-diffusion problems. *International Journal for Numerical Methods in Engineering*, 40(3):565–576.

[Mattheij et al., 2005] Mattheij, R. M., Rienstra, S. W., and Boonkkamp, J. T. T. (2005). *Partial differential equations: modeling, analysis, computation*. SIAM.

[MFEM, 2019] MFEM (2019). MFEM: Modular finite element methods library. `mfem.org`.

[Miller et al., 1995] Miller, J. J. H., O'Riordan, E., and Shishkin, G. I. (1995). On the use of fitted operator methods for singularly perturbed partial differential equations. In *Advanced mathematics: computations and applications (Novosibirsk, 1995)*, pages 518–531. NCC Publ., Novosibirsk.

[Miller et al., 1996] Miller, J. J. H., O'Riordan, E., and Shishkin, G. I. (1996). *Fitted numerical methods for singular perturbation problems*. World Scientific Publishing Co., Inc., River Edge, NJ. Error estimates in the maximum norm for linear problems in one and two dimensions.

[Miller et al., 2012] Miller, J. J. H., O'Riordan, E., and Shishkin, G. I. (2012). *Fitted Numerical Methods for Singular Perturbation Problems*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, revised edition. Error estimates in the maximum norm for linear problems in one and two dimensions.

[Morton, 1996] Morton, K. W. (1996). *Numerical solution of convection-diffusion problems*, volume 12 of *Applied Mathematics and Mathematical Computation*. Chapman & Hall, London.

[Nhan and Vulanović, 2021] Nhan, T. A. and Vulanović, R. (2021). The Bakhvalov mesh: a complete finite-difference analysis of two-dimensional singularly perturbed convection-diffusion problems. *Numer. Algorithms*, 87(1):203–221.

[Ølgaard and Wells, 2010] Ølgaard, K. B. and Wells, G. N. (2010). Optimizations for quadrature representations of finite element tensors through automated code generation. *ACM Trans. Math. Softw.*, 37(1).

[O'Riordan and Pickett, 2019] O'Riordan, E. and Pickett, M. L. (2019). Numerical approximations to the scaled first derivatives of the solution to a two parameter singularly perturbed problem. *J. Comput. Appl. Math.*, 347:128–149.

[Quarteroni and Valli, 1999] Quarteroni, A. and Valli, A. (1999). *Domain decomposition methods for partial differential equations*. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York. Oxford Science Publications.

[Rathgeber et al., 2017] Rathgeber, F., Ham, D. A., Mitchell, L., Lange, M., Luporini, F., McRae, A. T. T., Bercea, G.-T., Markall, G. R., and Kelly, P. H. J. (2017). Firedrake: automating the finite element method by composing abstractions. *ACM Trans. Math. Software*, 43(3):Art. 24, 27.

[Roos, 2006] Roos, H.-G. (2006). Error estimates for linear finite elements on Bakhvalov-type meshes. *Appl. Math.*, 51(1):63–72.

[Roos and Schopf, 2012] Roos, H.-G. and Schopf, M. (2012). Analysis of finite element methods on Bakhvalov-type meshes for linear convection-diffusion problems in 2D. *Appl. Math.*, 57(2):97–108.

[Roos et al., 1996] Roos, H.-G., Stynes, M., and Tobiska, L. (1996). *Numerical methods for singularly perturbed differential equations*, volume 24 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin. Convection-diffusion and flow problems.

[Roos et al., 2008] Roos, H.-G., Stynes, M., and Tobiska, L. (2008). *Robust Numerical Methods for Singularly Perturbed Differential Equations*, volume 24 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition. Convection-Diffusion-Reaction and Flow Problems.

[Russell and Madden, 2017] Russell, S. and Madden, N. (2017). An introduction to the analysis and implementation of sparse grid finite element methods. *Comput. Methods Appl. Math.*, 17(2):299–322.

[Russell and Madden, 2020] Russell, S. and Madden, N. (2020). Analysis of a Galerkin finite element method applied to a singularly perturbed reaction-diffusion problem in three dimensions. *Int. J. Numer. Anal. Model.*, 17(3):297–315.

[Shishkin, 1990] Shishkin, G. I. (1990). Grid approximation of singularly perturbed boundary value problems with convective terms. *Soviet J. Numer. Anal. Math. Modelling*, 5(2):173–187.

[Shishkin and Shishkina, 2009] Shishkin, G. I. and Shishkina, L. P. (2009). *Difference Methods for Singular Perturbation Problems*, volume 140 of *Chapman & Hall/CRC Monographs and Surveys in Pure and Applied Mathematics*. CRC Press, Boca Raton, FL.

[Stynes, 2005] Stynes, M. (2005). Steady-state convection-diffusion problems. *Acta Numer.*, 14:445–508.

[Stynes and Stynes, 2018] Stynes, M. and Stynes, D. (2018). *Convection-diffusion problems*, volume 196 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI; Atlantic Association for Research in the Mathematical Sciences (AARMS), Halifax, NS. An introduction to their analysis and numerical solution.

[Verfürth, 1994] Verfürth, R. (1994). A posteriori error estimates for nonlinear problems. Finite element discretizations of elliptic equations. *Math. Comp.*, 62(206):445–475.

[Vulanović, 1987] Vulanović, R. (1987). Nonequidistant generalizations of the Gushchin-Shchennikov scheme. *Z. Angew. Math. Mech.*, 67(12):625–632.

[Wang et al., 2021] Wang, Y., Meng, X., and Li, Y. (2021). The finite volume element method on the Shishkin mesh for a singularly perturbed reaction-diffusion problem. *Comput. Math. Appl.*, 84:112–127.

[Weller et al., 2016] Weller, H., Browne, P., Budd, C., and Cullen, M. (2016). Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge-Ampère type equation. *J. Comput. Phys.*, 308:102–123.

[Zhang and Liu, 2020] Zhang, J. and Liu, X. (2020). Convergence of a finite element method on a Bakhvalov-type mesh for singularly perturbed reaction-diffusion equation. *Appl. Math. Comput.*, 385:125403, 9.

[Zhang et al., 2022] Zhang, M., Huang, W., and Qiu, J. (2022). A well-balanced positivity-preserving quasi-Lagrange moving mesh DG method for the shallow water equations. *Commun. Comput. Phys.*, 31(1):94–130.