# RELIABLE AUTOMATIC MODEL REDUCTION COMBINING A POSTERIORI ERROR ANALYSIS AND A POSTERIORI MODEL REDUCTION

Thesis submitted to

THE UNIVERSITY OF OXFORD

in partial fulfilment of the requirements
of the award of the degree

MASTER OF SCIENCE BY RESEARCH

by

**Antonietta Ambuehl–Guglielmino**

ST HUGH'S COLLEGE

DEPARTMENT OF COMPUTER SCIENCE




Hilary Term 2019

**Abstract**

Mathematical models are used for simulations and predictions of various phenomena and processes that can be translated into mathematical language. Nowadays, with easy access to powerful computers and specialised software, even complicated models can be handled. Model output can be generated even by users that may not have specialised mathematical training. A consequence of the increase in complexity of the models on the one hand, and the relative ease of numerical output generation on the other hand, can be a loss of insight into the mathematical model. One way to regain this is to reduce the original model to a simpler form that will nevertheless capture the key features of the model and elucidate the processes that are modelled, especially those that are of interest to the user.

In this thesis I develop a model reduction algorithm that is based on a technique borrowed from finite element methods. An *a posteriori* error estimate, where the error is defined as the difference between the unknown solution and a FE approximation thereof, is used to ensure that the mesh on the computational domain is fine enough such that the discretised scheme approximates the differential operator sufficiently accurately. Model reduction on the other hand uses an *a posterioi error estimate*, where the error is defined as the difference between the solutions to the original model and a reduced model, to generate a differential operator that is a close enough approximation of the original operator.

The main research chapter focuses on the mathematical analysis that this automated method entails. The model reduction technique is then applied to model problems to validate the numerical methods that are employed. An enzyme kinetic reaction model is discussed in detail as the application of the developed model reduction method satisfies the aim of this thesis to a great extent. In particular, I can show with that example that the reduced models which were automatically generated correspond to those obtained by asymptotic analysis. The same elucidation of the processes that are modelled could therefore be achieved with the automatic model reduction technique as with methods that usually need considerable mathematical expertise.

# Contents

**1 Introduction** **1**

1.1 Aim of Thesis . . . . . 2

1.2 Model Solutions and Interpretation . . . . . 3

1.3 Overview of Thesis . . . . . 4

**2 The Finite Element Method** **6**

2.1 Function Spaces . . . . . 6

2.1.1 Continuous Functions . . . . . 7

2.1.2 Lebesgue Spaces . . . . . 8

2.1.3 Sobolev Spaces . . . . . 10

2.2 Space of Polynomials . . . . . 11

2.2.1 Polynomial Approximation in one Dimension . . . . . 12

2.3 The Finite Element Method . . . . . 14

2.3.1 System of Ordinary Differential Equations . . . . . 17

2.4 *A Posteriori* Error Estimation . . . . . 23

2.4.1 Linear Functionals of the Solution . . . . . 24

2.4.2 The Dual Problem . . . . . 24

2.4.3 Galerkin Orthogonality . . . . . 26

2.4.4 Adaptive Mesh Refinement . . . . . 27

**3 Model Reduction Review** **28**

3.1 Approaches to Model Reduction . . . . . 29

3.1.1 Perturbation Methods . . . . . 29

3.1.2 Model Order Reduction . . . . . 31

3.1.3 Other Methods . . . . . 34

3.2 Model Reduction by Duality . . . . . 35

3.2.1 Model Reduction for Systems of ODEs . . . . . 38

3.2.2 Model Reduction Combined with Mesh Refinement . . . . . 43

**4 The Model Reduction Method** **49**

4.1 Model Reduction Technique . . . . . 51

4.1.1 Derivation of the Induced Error Estimate . . . . . 52

4.1.2 The Reduced Model . . . . . 54

4.1.3 Model Reduction Algorithm . . . . . 57

4.1.4 Initialisation and Rules for Model Reduction Algorithm . . . . . 59

4.2 Application of Model Reduction Algorithm . . . . . 62

4.2.1 Linear System of ODEs . . . . . 63

4.2.2 Nonlinear System of ODEs . . . . . 71

4.2.3 Summary and Outlook . . . . . 75

**5 Combination of Model Reduction and Mesh Refinement** **76**

5.1 Mesh Refinement Based on Linear Functionals of the Solution . . . . . 77

5.1.1 Mesh Refinement Algorithm . . . . . 79

5.1.2 Combined Algorithm . . . . . 80

5.2 Application of Combined Method . . . . . 80

5.2.1 Nonlinear Model Problem . . . . . 80

5.2.2 Enzyme Kinetics . . . . . 83

5.2.3 Neuronal Spike Dynamics . . . . . 92

**6 Conclusion** **97**

# CHAPTER 1

# Introduction

Mathematical tools are widely used to model phenomena arising in the life sciences and other fields, such as engineering and physics for instance. Such models often take the form of systems of algebraic or differential equations, derived from physical laws like conservation of mass or energy or the law of mass action, and important parameters are determined from experimental data. A mathematical model can be used to gain a better understanding of a process that is being modelled and also as a tool for prediction. Before the advent of computers, mathematical models had to be kept simple from the start or otherwise simplified to be amenable for finding a solution or at least an approximation thereof. The increase in computing power on the one hand and the availability of data on the other hand allows for more complex models.

Computational power, specialised software, and advanced mathematical tools thus enable modellers to compute solutions to their mathematical models. Strictly speaking, what is computed is not a solution but a numerical approximation to the usually unknown 'true' or analytic solution. Any numerical method that is employed to compute this approximation involves numerical errors. We usually call the difference between the true solution and the numerical approximation thereof the computational error. If we could control this error then it should in principle be possible to make accurate predictions with an appropriate mathematical model. However, there are other factors which contribute to the 'accuracy' of the model output, such as initial data input and the reliability of the data that the model is based on, amongst other factors. Thus, in addition to the computational error there is also a data modelling error to be considered [1]. In this thesis however we will disregard all other possible errors and only concentrate on the use and control of the computational error and on ways to use this to gain insight into the processes the system of equations seeks to model.

## 1.1 Aim of Thesis

Our goal is to gain some insight into the mathematical model, what its key features are, which components of the solution influence others, and which are important for an outcome of interest. There exist established methods to achieve just this, some of which are reviewed in the next chapter. However, we strive for an 'automatic' or 'algorithmic' technique without having to resort to advanced mathematical methods. At the same time we would like a measure for the accuracy of the computed numerical approximation to the solution, be that the solution of the original model or be that the solution of an alternative model that captures the essence of the original one. The first part of the aforementioned goal can be achieved by model reduction[1], meaning a simplification of the original system of equations. A measure of the accuracy of the numerical approximation to the solution is achieved by analysing the error which is either an *a priori* or an *a posteriori* error. We work with the latter since we seek a measure for the accuracy of the numerical approximation in relation to the unknown true solution, whereas *a priori* error estimates rely on just this unknown quantity [2].

*A priori* and *a posteriori* error analysis is used in finite element methods which will be employed when computing approximations to solutions to the systems of differential equations constituting the mathematical models we seek to simplify. *A posteriori* error bounds were only introduced in the late 1970s, more than three decades after the inception of the finite element techniques [3]. Bounds on the error are estimated in a user-defined norm or functional of the solution. Global bounds, meaning estimates thereof based on the whole computational domain, are usually sufficient for a prescribed accuracy. Hoverer, the computation of local, or element-wise, error estimates can be used to control the error further by refining the mesh, that is to say the discretisation of the computational domain. It is this technique of error estimation, with contributions over each finite element to an upper bound for a functional of the solution, that can also be used to generate a reduced model [4].

[1]This should not be confused with 'model order reduction' which is widely denoted by the shorter 'model reduction' in the literature, see Chapter 3 where model reduction techniques are reviewed.

## 1.2 Model Solutions and Interpretation

When a mathematical model is set up which describes a process 'in real life' then a reduction has already taken place because it is but an abstraction. This is so because only those parts of the system under consideration are included and translated into mathematical language which are identified as being (the most) important. Small alterations in the formulation of the model may make it untractable for mathematical manipulation and thus finding solutions, but now that we have computers this is no longer really a problem. Numerical approximations to solutions of even very complicated models can be found and the effect of changes in a system can be tested quickly. One might however lose a 'feel' for the model as analysing it (and the behaviour of its solutions) is either difficult or in fact not possible, even if one possesses strong mathematical expertise.

We are not concerning ourselves with the mechanisms used and assumptions that were made in order to arrive at a particular model. Thus, whenever a model is being discussed in this thesis, we will not question the validity of the model. We are only concerned with the question whether a given model can be reduced[2]. This seems to contradict what we have said above about a model already being reduced. Since only the most important aspects or processes have been included in the model it seems that therefore these cannot be neglected. This may well be the case if one looks at a model and its output as a whole, which usually means that the accuracy of a numerical approximation to a solution is measured by some (global) norm of the error defined as the difference between the true and the numerical solution. If instead the accuracy were measured by a (linear) functional of the solution which reflects a user-defined outcome of interest, then it may well be possible to get a similar outcome with a reduced (in the sense of simplified) model.

[2]Reduction is meant in the sense that the (differential) operator that acts on the solution (vector) could be simplified further such that the reduced model solution approximates the true solution with satisfactory accuracy.

## 1.3 Overview of Thesis

**Chapter 2** introduces the finite element method (FEM) which is applied in this thesis to find numerical approximations of solutions to initial value systems of first order ordinary differential equations (ODEs). It also contains the mathematical terminology and notation as well as a summary of the concepts used later. These include polynomial interpolation, the Galerkin FEM, linear functionals of the solution, and *a posteriori* error estimation using a duality argument.

**Chapter 3** is a review of various model reduction methods where the second part of the chapter focuses on model reduction by duality. That section is important for subsequent chapters as our model reduction method is an extension of the technique introduced by Whiteley in [4] which we describe there. The chapter concludes with a derivation of bounds and a broad description of model reduction combined with mesh refinement as part of preliminary work done for this thesis. Results that are derived there will be referred to frequently in the next chapter.

**Chapter 4** forms the main technical part of this thesis. In it we present our model reduction method that extends the work of Whiteley [4] in a novel way that allows for the generation of more reduced (in the sense of 'simpler') models via the model reduction algorithm than was previously possible. The latter part of this chapter is used to verify that the numerical methods do indeed produce reliable results by applying the model reduction method to two model problems for which the solutions to both the original and the reduced models can be found.

**Chapter 5** is dedicated to the combination of model reduction with adaptive mesh refinement. This thesis focuses on model reduction as this is novel, we will however show that the combination of the two algorithms is more efficient than the original method that relied on 'true' solutions (meaning approximations on very fine meshes). The second part of this chapter sees the application of the combined algorithm to mathematical models. We can show that at least for one of the examples (modelling enzyme kinetics) the model reduction method works remarkably well. This means that the aim

of this thesis – to gain insight into a mathematical model through 'automatic' model reduction without having to resort to methods that require mathematical expertise – can clearly be shown.

**Chapter 6** concludes this thesis with a discussion of the model reduction method and its sensitivities to various choices that can be made with its implementation. It also lists some of the issues that were raised for the combined method and a suggested way to resolve these. We also propose there how tackling these sensitivities and issues could form the basis for future work.

# CHAPTER 2

# The Finite Element Method

Here we introduce the vector spaces as well as terminology and notation although, as the title suggests, the main part will focus on the finite element method which will be applied in subsequent chapters. The various spaces are introduced as they are needed for the FE method where the solutions and approximations to these belong to different spaces and subspaces, each with their own metric and properties. Then, as FE approximations are linear combinations of polynomial interpolants, we give a short outline on function interpolation. The main section focuses on the Galerkin FE method for systems of first order ODEs and introduces continuous and discontinuous piecewise polynomial spaces which will be used for the examples in Chapters 4 and 5. This chapter concludes with a brief outline on linear functionals of the solution of a system of differential equation and how this is used for *a posteriori* error estimation using a duality argument.

## 2.1 Function Spaces

We denote a connected subset of $\mathbb{R}^d$ by $\Omega$ if the dimension $d > 1$ and by $I$ for $d = 1$; if a function defined on $\Omega$ is a function of $d$-dimensional space, then we denote the independent variable by $\mathbf{x} = (x_1, \ldots, x_d)^\mathsf{T}$, and for $d = 1$, if a function defined on the interval $I$ is a function of time, then the independent variable will be denoted by $t$.

Furthermore, if $\mathcal{V}$ denotes a vector space of functions $v(t, \mathbf{x})$ say, then we use the bold font notation $\boldsymbol{\mathcal{V}}$ for the space of vector-valued functions, that is

$$\mathbf{v}(t, \mathbf{x}) = (v_1(t, \mathbf{x}), \ldots, v_N(t, \mathbf{x}))^\mathsf{T} \in \boldsymbol{\mathcal{V}} = \underbrace{\mathcal{V} \times \cdots \times \mathcal{V}}_{N \text{ times}}.$$

As usual, for two vectors $\mathbf{v}$, $\mathbf{w}$ in an $N$-dimensional vector space, the scalar product is

$$\mathbf{v} \cdot \mathbf{w} = \sum_{n=1}^{N} v_n w_n.$$

### 2.1.1 Continuous Functions

We define a multi-index $\kappa$ as a $d$-tuple of non-negative integers, $\kappa = (\kappa_1, \ldots, \kappa_d)$, and its length as $|\kappa| = \kappa_1 + \cdots + \kappa_d$. Now we can define the generalised differential operator of order $k = |\kappa|$ by

$$D^{\kappa} = \frac{\partial^{|\kappa|}}{\partial x_1^{\kappa_1} \cdots \partial x_d^{\kappa_d}}.$$

Note that for $d = 1$ this is the $k^{th}$ ordinary derivative operator $D^k = \dfrac{\mathrm{d}^k}{\mathrm{d}x^k}$.

Let $\Omega \subset \mathbb{R}^d$ be open, then the space of continuous functions on $\Omega$ is denoted by

$$\mathcal{C}(\Omega) = \{u(\mathbf{x})\colon u \text{ is continuous on } \Omega\},$$

and the space of $k$-times continuously differentiable functions on $\Omega$ is

$$\mathcal{C}^k(\Omega) = \{u\colon D^{\kappa} u \in \mathcal{C}(\Omega) \text{ for all } \kappa \text{ with } |\kappa| = 0, \ldots, k\}.$$

Assume that $\Omega \in \mathbb{R}^d$ is bounded and open and that $u \in \mathcal{C}^k(\Omega)$, then $u \in \mathcal{C}^k(\overline{\Omega})$ if $u$ can be extended to a continuous function on the closure $\overline{\Omega}$ of $\Omega$ for all $\kappa$ with $|\kappa| = 0, \ldots, k$ where for $k = 0$ we write $\mathcal{C}(\overline{\Omega})$ rather than $\mathcal{C}^0(\overline{\Omega})$. A norm associated with the vector space $\mathcal{C}^k(\overline{\Omega})$ is the supremum norm which is defined as

$$\|u\|_{\mathcal{C}^k(\overline{\Omega})} = \sum_{|\kappa|=0}^{k} \sup\{|D^{\kappa} u(\mathbf{x})|\colon \mathbf{x} \in \Omega\}.$$

We denote the support of a continuous function $u$ on an open set $\Omega \in \mathbb{R}^d$ by supp $u$, and it is defined as the closure of the set $\{\mathbf{x} \in \Omega\colon u(\mathbf{x}) \neq 0\}$. We can now define the

sets $\mathcal{C}_0^k(\Omega)$ and $\mathcal{C}_0^\infty(\Omega)$ by

$$\mathcal{C}_0^k(\Omega) = \big\{u \in \mathcal{C}^k(\Omega) \colon \text{supp } u \text{ is a bounded subset of } \Omega\big\},$$
$$\text{and} \quad \mathcal{C}_0^\infty(\Omega) = \bigcap_{k\geq 0} \mathcal{C}_0^k(\Omega).$$

Note that if $v \in \mathcal{C}_0^\infty(\Omega)$ then $D^\kappa v$ vanishes at the boundary of $\Omega$ for all $|\kappa| \geq 0$. Therefore, if we integrate the product $(D^\kappa u(\mathbf{x}))\, v(\mathbf{x})$ over $\Omega$, where $u \in \mathcal{C}^k(\Omega)$ with $k = |\kappa|$ and $v \in \mathcal{C}_0^\infty(\Omega)$, and use integration by parts ($k$ times), then all arising integrals over the boundary of $\Omega$ vanish and we have

$$\int_\Omega (D^\kappa u(\mathbf{x}))\, v(\mathbf{x})\, \mathrm{d}\mathbf{x} = (-1)^{|\kappa|} \int_\Omega u(\mathbf{x})\, (D^\kappa v(\mathbf{x}))\, \mathrm{d}\mathbf{x}.$$

### 2.1.2 Lebesgue Spaces

Assume that $\Omega \subset \mathbb{R}^d$ is open and let $\mathcal{L}_1(\Omega)$ denote the space of Lebesgue integrable functions, that is,

$$\text{if } u(\mathbf{x}) \in \mathcal{L}_1(\Omega) \text{ then } \int_\Omega |u(\mathbf{x})|\, \mathrm{d}\mathbf{x} < \infty.$$

Note that two functions which are equal except on a set of measure zero, have the same (Lebesgue) integral. Thus the space $\mathcal{L}_1(\Omega)$ of Lebesgue integrable functions consists of equivalence classes of functions.

If $\omega$ is a bounded open set whose closure $\overline{\omega}$ is a subset of $\Omega$ and $u \in \mathcal{L}_1(\omega)$ for all such $\omega$, then we say that $u$ is a locally integrable function defined on $\Omega$.

For $p > 1$ we define the space $\mathcal{L}_p(\Omega)$ similarly by

$$\mathcal{L}_p(\Omega) = \left\{u(\mathbf{x}) : \Omega \to \mathbb{R} \colon \int_\Omega |u(\mathbf{x})|^p\, \mathrm{d}\mathbf{x} < \infty\right\}.$$

For any positive integer $p$ we can define the associated $\mathcal{L}_p$ norm as

$$\|u\|_{\mathcal{L}_p(\Omega)} = \left( \int_\Omega |u(\mathbf{x})|^p \, \mathrm{d}\mathbf{x} \right)^{1/p}.$$

Assume that $u$ is a function defined on an open subset $\Omega \in \mathbb{R}^d$. The essential supremum of $u$ on $\Omega$, $\operatorname{ess\,sup}\{|u(\mathbf{x})| \colon \mathbf{x} \in \Omega\}$ is defined as the smallest positive constant $c$ such that $|u(\mathbf{x})| \leq c$ for almost every $\mathbf{x} \in \Omega$ (meaning that the set $\{\mathbf{x} \in \Omega \colon |u(\mathbf{x})| > c\}$ has measure zero). Now we can define the space $\mathcal{L}_\infty(\Omega)$ by

$$\mathcal{L}_p(\Omega) = \{u(\mathbf{x}) : \Omega \to \mathbb{R} \colon \operatorname{ess\,sup}\{|u(\mathbf{x})| \colon \mathbf{x} \in \Omega\} < \infty\},$$

and the associated $\mathcal{L}_\infty$ norm by

$$\|u\|_{\mathcal{L}_\infty(\Omega)} = \operatorname{ess\,sup}\{|u(\mathbf{x})| \colon \mathbf{x} \in \Omega\}.$$

The case $p = 2$ is of special importance. The space $\mathcal{L}_2(\Omega)$, often called the space of square integrable functions, can be equipped with an inner product. It is defined as

$$\langle u \, , v \rangle = \int_\Omega u(\mathbf{x}) v(\mathbf{x}) \, \mathrm{d}\mathbf{x} \quad \text{for all} \quad u, \, v \in \mathcal{L}_2(\Omega),$$

and we note that $\langle u \, , u \rangle = \|u\|^2_{\mathcal{L}_2(\Omega)}$.

If we are dealing with vector-valued functions defined on $\Omega$, then we will use the notation

$$\boldsymbol{\mathcal{L}}_2(\Omega) = \big\{ \mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), \ldots, u_N(\mathbf{x}))^\mathsf{T} \colon u_n \in \mathcal{L}_2(\Omega), \; n = 1, \ldots, N \big\},$$

and write the $\boldsymbol{\mathcal{L}}_2$ inner product as

$$\langle \mathbf{u} \, , \mathbf{v} \rangle = \int_\Omega \mathbf{u}(\mathbf{x}) \cdot \mathbf{v}(\mathbf{x}) \, \mathrm{d}\mathbf{x}.$$

### 2.1.3 Sobolev Spaces

Suppose that $u$ is a locally integrable function defined on an open set $\Omega \subset \mathbb{R}^d$. Then we say that the function $u$ has a weak derivative of order $k = |\kappa|$ if there exists a locally integrable function $w_k$ defined on $\Omega$ such that

$$\int_\Omega w_k(\mathbf{x})\, v(\mathbf{x})\, \mathrm{d}\mathbf{x} = (-1)^{|\kappa|} \int_\Omega u(\mathbf{x})\, (D^\kappa v(\mathbf{x}))\, \mathrm{d}\mathbf{x} \quad \text{for all } v \in \mathcal{C}_0^\infty(\Omega).$$

Note that if a weak derivative exists, then it is unique. For if we assume that also $\tilde{w}_k$ is a weak derivative of $u$ of order $k$, then $\int_\Omega (w_k(\mathbf{x}) - \tilde{w}_k(\mathbf{x}))\, v(\mathbf{x})\, \mathrm{d}\mathbf{x} = 0$ for all $v \in \mathcal{C}_0^\infty(\Omega)$, but then (by the DuBois Reynold's lemma) it follows that $w_k(\mathbf{x}) - \tilde{w}_k(\mathbf{x}) = 0$ for almost every $\mathbf{x} \in \Omega$. This means that the two functions are equivalent which proves that the weak derivative is unique if it exists. It is because of this uniqueness that the weak derivative of a function $u \in \mathcal{C}^k(\Omega)$ coincides with its generalised derivative $D^\kappa u$ of order $k = |\kappa|$, if it exists, and it is quite common to use the same notation for both. We adopt this practice in the definition of Sobolev spaces, that is we denote a weak derivative of $u$ of order $k = |\kappa|$ by $D^\kappa u$.

For any integer $p \in [1\,, \infty]$ we define the Sobolev space $\mathcal{S}_p^k(\Omega)$ of order $k$ by

$$\mathcal{S}_p^k(\Omega) = \{u \in \mathcal{L}_p(\Omega) \colon D^\kappa u \in \mathcal{L}_p(\Omega) \text{ for all } \kappa \text{ with } |\kappa| = 1, \ldots, k\}.$$

The Sobolev norms are

$$\|u\|_{\mathcal{S}_p^k(\Omega)} = \left( \sum_{|\kappa|=0}^{k} \|D^\kappa u\|_{\mathcal{L}_p(\Omega)}^p \right)^{1/p} \qquad \text{for } p \in [1\,, \infty),$$

$$\text{and} \quad \|u\|_{\mathcal{S}_\infty^k(\Omega)} = \sum_{|\kappa|=0}^{k} \|D^\kappa u\|_{\mathcal{L}_\infty(\Omega)}.$$

As with the Lebesgue spaces, also the Sobolev spaces with $p = 2$ are of special impor-

tance. The space $\mathcal{S}_2^k(\Omega)$ is a Hilbert space with the inner product defined as

$$\left\langle u\,,v\right\rangle_{\mathcal{S}_2^k(\Omega)} = \sum_{|\kappa|=0}^{k} \left\langle D^\kappa u\,, D^\kappa v\right\rangle,$$

where $\left\langle \cdot\,,\cdot\right\rangle$ denotes the $\mathcal{L}_2$ inner product.

The spaces $\mathcal{S}_2^k(\Omega)$ are therefore often denoted by $\mathcal{H}^k(\Omega)$. In one dimension for instance we thus have, in accordance with the definitions above,

$$\mathcal{H}^1(I) = \{u(t) \in \mathcal{L}_2(I) \colon \mathrm{d}u/\mathrm{d}t \in \mathcal{L}_2(I)\},$$
$$\text{and} \quad \|u\|_{\mathcal{H}^1(I)} = \left(\|u\|^2_{\mathcal{L}_2(I)} + \|\mathrm{d}u/\mathrm{d}t\|^2_{\mathcal{L}_2(I)}\right)^{1/2}.$$

## 2.2 Space of Polynomials

The finite element method, which will be introduced in the next section, provides a technique for computing the numerical approximation to the solution of a differential equation. This method approximates the solution by first partitioning the computational domain into small regions known as elements. The solution on each element is then approximated by a polynomial function which is usually of low degree. Suppose we seek a numerical approximation to a function $u$ defined on an interval $I \in \mathbb{R}$ or a bounded connected domain $\Omega \in \mathbb{R}^d$ for $d = 2, 3$. In one dimension the domain is divided into subintervals whose intersection is empty and whose union is the whole interval. In two and three dimensions we discretise the computational domain, meaning that the boundary is approximated by a polygon ($d = 2$) or a polyhedron ($d = 3$), and the enclosed area or volume is divided into (finite) elements. These elements are again polygons, respectively polyhedra, that do not overlap and cover the whole discretised domain. Now we define a basis of the finite-dimensional approximation space consisting of piecewise polynomials in $d$ variables, whose support is an element. Depending on the choice of elements these polynomials are either a linear combination of basis functions of spaces of polynomials ($d \geq 1$) or bilinear forms ($d > 1$). The numerical

approximation to $u$ can now be represented as a linear combination of these basis functions, and is thus a piecewise polynomial approximation. We will not go into more detail here as this should suffice to motivate the introduction of spaces of polynomials. We will now introduce (piecewise) polynomial approximation in one dimension. We will not discuss higher dimensions here as we will not need this for our thesis, though the principle can be easily extended to more dimensions, too. See for instance Chapter 14 on 'Piecewise Polynomials in Several Dimensions' in Eriksson et al [5] for higher dimensional methods.

### 2.2.1 Polynomial Approximation in one Dimension

Let $\mathcal{P}^q(I)$ denote the vector space of polynomial functions of one variable of degree at most $q$ defined on the interval $I \subset \mathbb{R}$, that is

$$\mathcal{P}^q(I) = \left\{ v \colon I \to \mathbb{R} : \ v(x) = \sum_{k=0}^{q} c_k x^k \right\}$$

Suppose now that we would like to find a polynomial approximation to a function $u$ defined on $I$. We say that a polynomial $\pi_q u$ of degree at most $q$ is a polynomial interpolant if $\pi_q u(\xi_j) = u(\xi_j)$ for $q+1$ distinct values $\xi_j \in I$. The dimension of $\mathcal{P}^q(I)$ is $q+1$ and a common basis is the set of monomials $\{1, x, \ldots, x^q\}$. A more convenient basis is the set of Lagrangian functions, also called a nodal basis, which makes the determination of a polynomial interpolant easier. Given $q+1$ distinct 'nodes' $\xi_j$, $j = 0, \ldots, q$, then these define the Lagrangian functions

$$\lambda_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{q} \frac{x - \xi_k}{\xi_j - \xi_k}, \quad j = 0, \ldots, q.$$

The polynomial interpolant of degree $q$ can thus be written as

$$\pi_q u(x) = \sum_{j=0}^{q} u(\xi_j)\lambda_j(x), \quad \text{since } \lambda_j(\xi_k) = \delta_{jk}.$$

Linear and quadratic nodal basis functions and interpolants are shown in Figure 2.1.

We can achieve a more 'accurate' approximation, where 'accuracy' here means some norm of the error $\pi_q u - u$ or some other measure thereof, by increasing the degree $q$ of the polynomial space $\mathcal{P}^q(I)$. Better accuracy can however also be achieved with low degree polynomials. To this end we divide the domain $I = (a\,, b]$ into $M$ subintervals $I_m = (x_{m-1}\,, x_m]$ where the $M-1$ nodes $x_m$ satisfy $a = x_0 < x_1 < \cdots < x_{M-1} < x_M = b$. These in turn define a mesh $\mathcal{I}_h = \{I_m\}_{m=1}^M$, where the subscript $h$ stands for the mesh function $h(x)$ which is defined via the lengths of the subintervals, as $h(x) := h_m = x_m - x_{m-1}$ for $x \in I_m$.

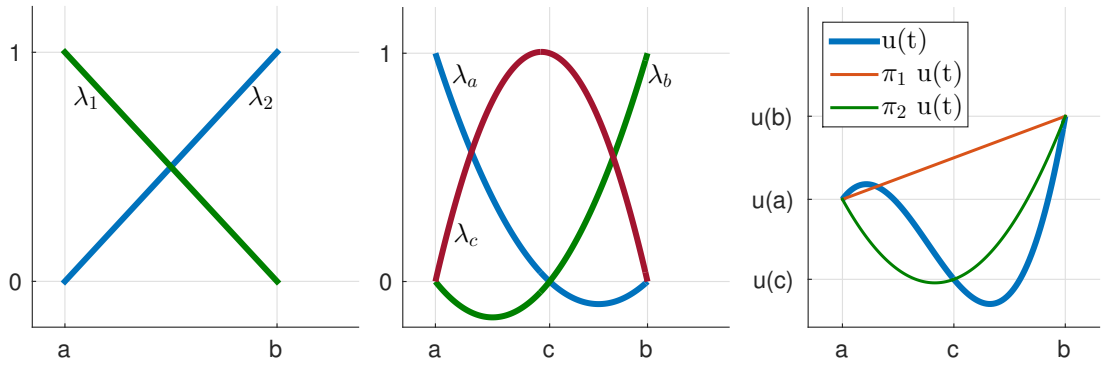



Figure 2.1: Nodal basis functions for $\mathcal{P}^1(I)$ and $\mathcal{P}^2(I)$, and linear and quadratic interpolants on $I = [a\,, b]$.

A function can thus be approximated by piecewise polynomials, that is, polynomials of the same degree $q$ defined on each 'piece' $I_m$, and these can be continuous or discontinuous. Spaces of piecewise polynomials are vector spaces with bases consisting of functions whose restriction on an element of the mesh are polynomials of a given degree $q$.

We distinguish between the space of continuous piecewise polynomials $\mathcal{S}^h_{c(q)}$ and the space of discontinuous piecewise polynomials $\mathcal{S}^h_{d(q)}$, that is,

$$\mathcal{S}^h_{c(q)} = \Big\{ w(x) \colon w\big|_{I_m} \in \mathcal{P}^q(I_m),\ w \text{ is continuous on } I \Big\}, \tag{2.1}$$

$$\mathcal{S}^h_{d(q)} = \Big\{ w(x) \colon w\big|_{I_m} \in \mathcal{P}^q(I_m),\ w \text{ may be discontinuous at } x_0, \ldots, x_M \Big\}. \tag{2.2}$$

Note that these spaces are dependent on the mesh function which we indicate by the superscript $h$. Figure 2.2 shows piecewise polynomial interpolants in $\mathcal{S}^h_{d(0)}$ (constant),

$\mathcal{S}^h_{c(1)}$ (linear) and $\mathcal{S}^h_{c(2)}$ (quadratic).

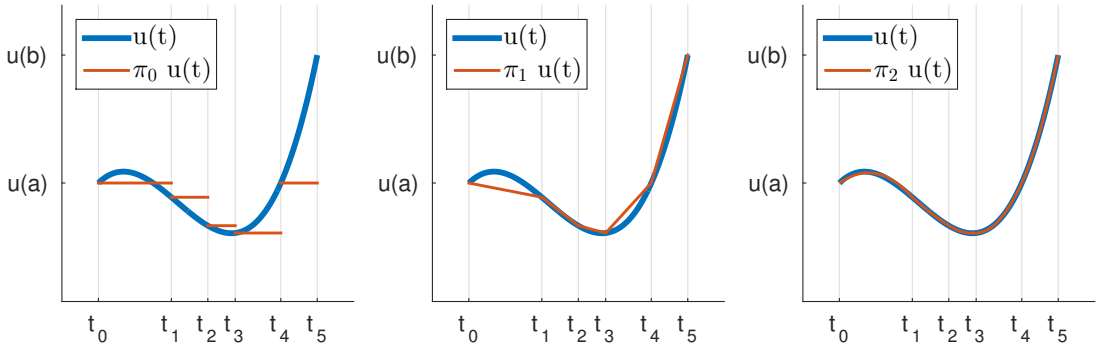



Figure 2.2: Piecewise constant, linear and quadratic interpolants.

## 2.3 The Finite Element Method

The finite element method has its beginnings in structural engineering. It is a way of solving structural problems, particularly to compute stresses, for example in the structure of bridges when a load is applied, or stresses on the frame of tall buildings under harsh weather conditions [6]. The underlying idea is to be able to estimate the effect caused by small changes on the components of a structure. For applied mathematicians this translates to solving differential equations, systems of equations that try to describe processes arising in the world around us. These equations model a continuum in the sense that time and space coordinates can take any value, not only discrete ones. This entails that possible solutions are continuous, but most of the time such 'true' solutions cannot be found or do not even exist. The next best non-discrete approximation to a solution would be a combination of but a few functions.

This was indeed Ritz's approach at the beginning of the $20^{th}$ century, to find a finite set of 'trial' functions whose linear combination approximates the solution. Both Ritz and Rayleigh used a variational formulation to compute the approximation to the solution of a differential equation, and Galerkin made use of an orthogonality condition to compute the coefficients in the linear combination used for the approximation [7]. Galerkin and his contemporaries had to make a good guess of the solution in order to find a very small set of trial functions, typically only two or three, in order to obtain a 'good'

approximation.

With the advent of the computer the size of the set of trial functions could be increased dramatically. Also it was no longer necessary to carefully choose special trial functions to minimize the (human) computational effort. Instead, simple ones like piecewise linear, quadratic or even piecewise constant functions could suffice, the more so if there are hundreds if not thousands of trial basis functions at our disposal. This idea was Courant's, a German mathematician who emigrated to the USA. The same idea was conceived by Chinese mathematician Feng from Beijing [8], but unlike his American counterparts he had no computer at his disposal to test his idea. The idea also appeared in Europe and Russia, all around the middle of the $20^{th}$ century, so it seems that it was "the idea of the time" [6].

The basic principle of Galerkin's method can be summarised as follows (see [9], Chapter 2). Suppose we are given a differential equation

$$\boldsymbol{\mathcal{D}}\mathbf{u} = \mathbf{f}, \tag{2.3}$$

where $\boldsymbol{\mathcal{D}}$ is a differential operator (including any boundary conditions), the unknown is $\mathbf{u}$, and $\mathbf{f}$ is some (forcing) function independent of $\mathbf{u}$, defined on a domain $\Omega$.

1. Multiplying equation (2.3) by a test function $\mathbf{v}$ and integrating over the domain $\Omega$ yields

$$\left\langle \boldsymbol{\mathcal{D}}\mathbf{u}\,, \mathbf{v} \right\rangle = \left\langle \mathbf{f}\,, \mathbf{v} \right\rangle, \tag{2.4}$$

   where $\left\langle\,,\right\rangle$ is the $\boldsymbol{\mathcal{L}}_2$ inner product. Equation (2.4) should hold for all $\mathbf{v}$ in a test space $\boldsymbol{\mathcal{V}}$. Informally, the idea is that if (2.4) is 'tested' at every point of the domain $\Omega$, then if $\mathbf{u}$ satisfies (2.4) it should also satisfy (2.3).

2. Integrating (2.4) by parts shifts some or all of the derivatives off $\mathbf{u}$ and onto $\mathbf{v}$, depending on the order of the operator $\boldsymbol{\mathcal{D}}$ and also on the test space $\boldsymbol{\mathcal{V}}$. Generally speaking, if the order of $\boldsymbol{\mathcal{D}}$ is $m$, say, and the test space is a Hilbert

space $\boldsymbol{\mathcal{V}} = \boldsymbol{\mathcal{H}}^s(\Omega)$ with $m \geq s$, then the solution $\mathbf{u}$ would be sought in the space $\boldsymbol{\mathcal{H}}^{m-s}(\Omega)$ as $s$ of the derivatives could be 'shifted' onto $\mathbf{v}$. This has a bearing on any boundary conditions, too. If none of the derivatives are shifted, so $s = 0$, then the full set of boundary conditions is imposed on $\mathbf{u}$. If $s = m$ on the other hand, none of the boundary conditions on any of the derivatives apply to the solution any longer. Between these extremes, so $0 < s < m$, only those boundary conditions on derivatives of order less than $m - s$ make sense to be imposed on $\mathbf{u}$. When $m$ is even we can have the special case $s = m/2$. This implies symmetry between trial and test space, which are both Hilbert spaces of the same order.

3. Galerkin's method starts with the weak form which is either (2.4) or what follows from that from integration by parts. This is then discretised, meaning that an approximation $\mathbf{u}^{\mathsf{h}}$ is sought in a finite-dimensional subspace $\boldsymbol{\mathcal{S}}^h$ of the trial space. This means that the Galerkin approximation satisfies

$$\left\langle \boldsymbol{\mathcal{D}}\mathbf{u}^{\mathsf{h}}\,, \mathbf{v}^{\mathsf{h}} \right\rangle = \left\langle \mathbf{f}\,, \mathbf{v}^{\mathsf{h}} \right\rangle \tag{2.5}$$

for all $\mathbf{v}^{\mathsf{h}}$ in a finite-dimensional subspace $\boldsymbol{\mathcal{V}}^h$ of the test space $\boldsymbol{\mathcal{V}}$.

4. When the spaces $\boldsymbol{\mathcal{S}}^h$ and $\boldsymbol{\mathcal{V}}^h$ have equal dimension $M$ say, then (2.5) can ultimately be written as a system of algebraic equations. To this end, choose bases $\{\chi_1 \ldots, \chi_M\}$ for $\mathcal{S}^h$ and $\{\psi_1, \ldots, \psi_M\}$ for $\mathcal{V}^h$ and write the Galerkin approximation as a linear combination $\mathbf{u}^{\mathsf{h}} = \sum_{j=1}^{M} \mathbf{U}_j \chi_j$, so each of the $N$ components of $\mathbf{u}^{\mathsf{h}} = (u_1^h, \ldots, u_N^h)^{\mathsf{T}}$ is $u_i^h = \sum_{j=1}^{M} U_{i,j}\chi_j$. As (2.4) should hold for all $\mathbf{v}^{\mathsf{h}} = (v_1^h, \ldots, v_N^h)^{\mathsf{T}} \in \boldsymbol{\mathcal{V}}^h$, it has to therefore hold for each and every one of the basis functions $\psi_k \in \mathcal{V}^h$ for $k = 1, \ldots, M$. This means that

$$\int_\Omega \left( \boldsymbol{\mathcal{D}} \sum_{j=1}^{M} \mathbf{U}_j \chi_j \right) \psi_k \, \mathrm{d}\mathbf{x} = \int_\Omega \mathbf{f}\psi_k \, \mathrm{d}\mathbf{x} \quad \text{for } k = 1, \ldots M. \tag{2.6}$$

If $\boldsymbol{\mathcal{D}}$ is linear and we have just one differential equation instead of a system of $N$

equations, then (2.6) can be written as an $M \times M$ matrix equation

$$\mathbf{BU} = \mathbf{F}, \quad \text{where } \mathbf{B} \in \mathbb{R}^{M\times M} \text{ and } \mathbf{U}, \mathbf{F} \in \mathbb{R}^{M}, \tag{2.7}$$

with $B_{kj} = \int_\Omega (\boldsymbol{\mathcal{D}}\chi_j)\psi_k \, \mathrm{d}\mathbf{x}$ and $F_k = \int_\Omega f\psi_k \, \mathrm{d}\mathbf{x}$, $1 \leq j, k \leq M$. Solving (2.7) yields $\mathbf{U} = (U_1, \ldots, U_M)^\mathsf{T} \in \mathbb{R}^M$ where the $U_j$ are the coefficients in the linear combination of the trial functions $\chi_j$ that make up $u^h$.

We will conclude this chapter with a description of the Galerkin method for ordinary differential equations. Systems of ODEs, in particular initial value problems, will again be encountered in the following chapters to show the model reduction technique in its application. Our description of ODEs in this thesis follows the above summary of the basic principle of Galerkin's method.

### 2.3.1 System of Ordinary Differential Equations

We will discuss the Galerkin FEM in a bit more detail here and will be looking at a general initial value problem of the form:

Find $\mathbf{u}(t) = (u_1(t), \ldots, u_N(t))^\mathsf{T} \in \boldsymbol{\mathcal{C}}^1(I)$ satisfying $\mathbf{u}(0) = \mathbf{u}_0$ such that

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}) = \mathbf{f}(t), \quad \text{for } 0 < t \leq T, \tag{2.8}$$

where $\mathbf{f}(t) = (f_1(t), \ldots, f_N(t))^\mathsf{T}$ and $\mathbf{a}(t, \mathbf{u}) = (a_1(t, \mathbf{u}), \ldots, a_N(t, \mathbf{u}))^\mathsf{T}$ are vector-valued functions of functions $f_n(t)$, $a_n(t, \mathbf{u}) \in \mathcal{L}_2(I)$, and $\mathbf{u}_0 \in \mathbb{R}^N$. We note here that all mathematical models in this thesis will have the form (2.8).

The first step is to multiply (2.8) with a test function $\mathbf{v}(t)$ to get the weak form:

Find $\mathbf{u}$ in a trial space $\boldsymbol{\mathcal{S}}$ satisfying the initial condition $\mathbf{u}(0) = \mathbf{u}_0$ such that

$$\int_0^T \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}) \right) \cdot \mathbf{v} \, \mathrm{d}t = \int_0^T \mathbf{f} \cdot \mathbf{v} \, \mathrm{d}t \tag{2.9}$$

for all $\mathbf{v}$ in a test space $\boldsymbol{\mathcal{V}}$.

The next step is to change to finite-dimensional subspaces $\boldsymbol{\mathcal{S}}^h$ and $\boldsymbol{\mathcal{V}}^h$ of trial and test spaces respectively. We differentiate between two piecewise polynomial methods, the continuous and discontinuous Galerkin methods cG($q$) and dG($q$), where the spaces are spanned by piecewise polynomials of degree $q$. Recall that piecewise polynomial spaces are defined by a mesh $\mathcal{I}_h$ or the mesh function $h$ and are denoted by $\mathcal{S}^h_{c(q)}$ and $\mathcal{S}^h_{d(q)}$ respectively, as defined previously, see (2.1) and (2.2). Both methods start with a global formulation, meaning that the entire computational domain $I = [0, T]$ is being treated as being (partitioned into) a single element, as we will explain now.

## The Continuous Galerkin Method

The continuous Galerkin cG($q$) method for the primal problem (2.8) is based on the following global Galerkin method:

Find $\mathbf{u}^{\mathsf{h}} = (u_1^h, \ldots, u_N^h)^{\mathsf{T}}$ with $u_n^h \in \mathcal{P}^q(I)$ satisfying $\mathbf{u}^{\mathsf{h}}(0) = \mathbf{u}_0$ such that

$$\int_0^T \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}) \right) \cdot \mathbf{v}^{\mathsf{h}} \, \mathrm{d}t = \int_0^T \mathbf{f} \cdot \mathbf{v}^{\mathsf{h}} \, \mathrm{d}t \tag{2.10}$$

for all $\mathbf{v}^{\mathsf{h}} = (v_1^h, \ldots, v_N^h)^{\mathsf{T}}$ with $v_n^h \in \mathcal{P}^{q-1}(I)$.

We apply (2.10) on each element $I_m$ to find the coefficients in the linear expansion for $\mathbf{u}^{\mathsf{h}} \in \mathcal{S}^h_{c(q)}$. For the following description we will set $q = 1$ since for $q > 1$ there are more than just the one nodal value on each subinterval $I_m$. We set $\mathbf{u}_0^{\mathsf{h}} = \mathbf{u}^{\mathsf{h}}(t_0) = \mathbf{u}_0$ and apply (2.10) to $I_1 = (0\,, t_1]$ instead of $I = (0\,, T]$, which when solved yields $\mathbf{u}_1^{\mathsf{h}} = \mathbf{u}^{\mathsf{h}}(t_1)$. Successively, assuming that $\mathbf{u}_{m-1}^{\mathsf{h}} = \mathbf{u}^{\mathsf{h}}(t_{m-1})$ has already been computed on $I_{m-1} = (t_{m-2}\,, t_{m-1}]$, apply (2.10) again to $I_m = (t_{m-1}\,, t_m]$ and solve for the next value $\mathbf{u}_m^{\mathsf{h}} = \mathbf{u}^{\mathsf{h}}(t_m)$, continuing until $m = M$. This means that the cG($q$) approximation to the primal solution can be computed efficiently by sequential computation on each element.

For $q = 1$ for instance, the trial space $\boldsymbol{\mathcal{S}}^h_{c(1)}$ is the space of continuous piecewise linear functions, and the test space $\boldsymbol{\mathcal{V}}^h_{d(0)}$ therefore the space of piecewise constant functions.

Basis functions $\chi_m$ for $\mathcal{S}^h_{c(1)}$ and $\psi_m$ for $\mathcal{V}^h_{d(0)}$, shown in Figure 2.3, are defined as

$$\chi_m(t) = \begin{cases} \frac{t-t_{m-1}}{t_m-t_{m-1}} & t \in I_m, \\ \frac{t-t_{m+1}}{t_m-t_{m+1}} & t \in I_{m+1}, \\ 0 & \text{otherwise.} \end{cases} \qquad \psi_m(t) = \begin{cases} 1 & t \in I_m, \\ 0 & \text{otherwise.} \end{cases}$$

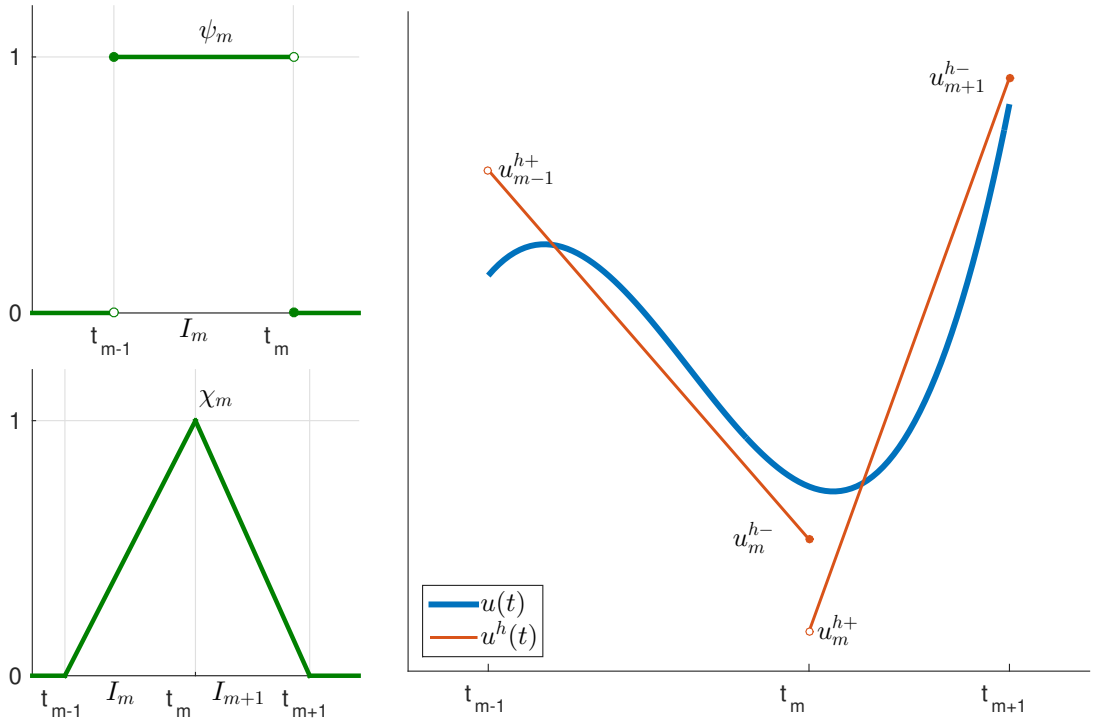



Figure 2.3: Basis functions for spaces of piecewise constant and continuous piecewise linear functions on the left. On the right is shown a function $u(t)$ on the interval $(t_{m-1}\,, t_{m+1}]$ and a dG(1) approximation $u^h(t)$ thereof.

This means that $\mathbf{u}^{\mathsf{h}} \in \boldsymbol{\mathcal{S}}^h_{c(1)}$ restricted on $I_m$ is

$$\begin{aligned} \mathbf{u}^{\mathsf{h}}(t)\big|_{I_m} &= \mathbf{u}^{\mathsf{h}}_{m-1}\frac{t-t_m}{t_{m-1}-t_m} + \mathbf{u}^{\mathsf{h}}_m\frac{t-t_{m-1}}{t_m-t_{m-1}}, \\ &= \frac{1}{h_m}\left(\mathbf{u}^{\mathsf{h}}_m(t-t_{m-1}) - \mathbf{u}^{\mathsf{h}}_{m-1}(t-t_m)\right), \end{aligned}$$

where $h_m = |I_m| = t_m - t_{m-1}$, and therefore

$$\frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}(t)\big|_{I_m}}{\mathrm{d}t} = \frac{1}{h_m}\left(\mathbf{u}^{\mathsf{h}}_m - \mathbf{u}^{\mathsf{h}}_{m-1}\right).$$

Thus (2.10) applied to $I_m$ becomes

$$\int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}\big|_{I_m}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m}) \right) \psi_m \, \mathrm{d}t = \int_{t_{m-1}}^{t_m} \mathbf{f}\psi_m \, \mathrm{d}t.$$

Note that the first part of the integral on the left-hand-side is simply $\mathbf{u}_m^{\mathsf{h}} - \mathbf{u}_{m-1}^{\mathsf{h}}$ and that $\psi_m \equiv 1$ on $I_m$. The cG(1) approximation $\mathbf{u}^{\mathsf{h}}(t)$ is therefore determined by the nodal values $\mathbf{u}_m^{\mathsf{h}} = \mathbf{u}^{\mathsf{h}}(t_m)$ for $m = 1, \ldots, M$, where $\mathbf{u}_0^{\mathsf{h}} := \mathbf{u}_0$, that satisfy

$$\mathbf{u}_m^{\mathsf{h}} - \mathbf{u}_{m-1}^{\mathsf{h}} + \int_{t_{m-1}}^{t_m} \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m}) \, \mathrm{d}t = \int_{t_{m-1}}^{t_m} \mathbf{f} \, \mathrm{d}t. \tag{2.11}$$

We will return to the continuous Galerkin method later when discussing convergence and stability properties. First however, we shall introduce the discontinuous Galerkin dG($q$) method.

## The Discontinuous Galerkin Method

The dG($q$) approximation $\mathbf{u}^{\mathsf{h}}$ is in $\boldsymbol{\mathcal{S}}_{d(q)}^h$ and is therefore no longer guaranteed to be continuous as 'jumps' are permitted over the element boundaries, the nodes $t_m$. To account for these discontinuities we introduce the following notation:

$$\begin{aligned}
\mathbf{u}_m^{\mathsf{h}^+} &= \lim_{t \searrow t_m} \mathbf{u}^{\mathsf{h}}(t) \;\text{ for the right-hand limit,} \\
\mathbf{u}_m^{\mathsf{h}^-} &= \lim_{t \nearrow t_m} \mathbf{u}^{\mathsf{h}}(t) \;\text{ for the left-hand limit,} \\
[\mathbf{u}^{\mathsf{h}}]_m &= \mathbf{u}_m^{\mathsf{h}^+} - \mathbf{u}_m^{\mathsf{h}^-} \;\text{ for the 'jump' over the element boundary.}
\end{aligned}$$

Similar to the continuous case the starting point is a global Galerkin method:

Find $\mathbf{u}^{\mathsf{h}} = (u_1^h, \ldots, u_N^h)^{\mathsf{T}}$ with $u_n^h \in \mathcal{P}^q(I)$ satisfying $\mathbf{u}_0^{\mathsf{h}^-} := \mathbf{u}_0$ such that

$$\int_0^T \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}) \right) \cdot \mathbf{v}^{\mathsf{h}} \, \mathrm{d}t + \gamma [\mathbf{u}^{\mathsf{h}}]_0 \cdot \mathbf{v}_0^{\mathsf{h}^+} = \int_0^T \mathbf{f} \cdot \mathbf{v}^{\mathsf{h}} \, \mathrm{d}t \tag{2.12}$$

for all $\mathbf{v}^{\mathsf{h}} = (v_1^h, \ldots, v_N^h)^{\mathsf{T}}$ with $v_n^h \in \mathcal{P}^q(I)$.

Note the extra term $\gamma[\mathbf{u}^{\mathsf{h}}]_0 \cdot \mathbf{v}_0^{\mathsf{h}^+}$ where $[\mathbf{u}^{\mathsf{h}}]_0$ is the 'jump' at the initial time $t_0$. If this 'jump' $[\mathbf{u}^{\mathsf{h}}]_0$ carries equal weight as the residual $\mathrm{d}\mathbf{u}^{\mathsf{h}}/\mathrm{d}t + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}})$ then $\gamma = 1$ and it turns out that this weighting gives "the best accuracy and stability" [5]. Based on this we take the same sequential approach as for the continuous case, starting again by applying (2.12) to $I_1 = (0\,, t_1]$ instead of $I = (0\,, T]$, then to $I_2 = (t_1\,, t_2]$ and so forth. Thus we can again compute the Galerkin approximation sequentially:

$$\begin{aligned}\text{For } m = 1, \ldots, M \text{ find } \mathbf{u}^{\mathsf{h}}\big|_{I_m} \text{ where } u_n^h\big|_{I_m} \in \mathcal{P}^q(I_m) \text{ with } \mathbf{u}_0^{\mathsf{h}^-} := \mathbf{u}_0 \text{ such that} \\ \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}\big|_{I_m}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m}) \right) \cdot \mathbf{v}^{\mathsf{h}}\big|_{I_m} \,\mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \mathbf{v}_{m-1}^{\mathsf{h}^+} \\ = \int_{t_{m-1}}^{t_m} \mathbf{f} \cdot \mathbf{v}^{\mathsf{h}}\big|_{I_m} \,\mathrm{d}t \\ \text{for all } \mathbf{v}^{\mathsf{h}} = (v_1^h, \ldots, v_N^h)^{\mathsf{T}} \text{ with } v_n^h \in \mathcal{P}^q(I_m).\end{aligned} \tag{2.13}$$

Note that in the discontinuous Galerkin dG($q$) method the trial and test spaces are identical, which is not the case in the continuous cG($q$) method. On the other hand the dG($q$) approximation can no longer satisfy the initial condition due to discontinuities. Still, in comparison with the cG($q$) method, the dG($q$) method has generally better stability properties, see below.

Let us write out (2.13) for $q = 0$ first, so both the trial and the test space are the space of piecewise constant functions. This is the simplest case where $\mathbf{u}^{\mathsf{h}}\big|_{I_m} = \mathbf{u}_m^{\mathsf{h}^-}$, which we will denote by $\mathbf{u}_m^{\mathsf{h}}$ for simplicity's sake since $\mathbf{u}_{m-1}^{\mathsf{h}^+} = \mathbf{u}_m^{\mathsf{h}^-}$. The derivative with respect to $t$ vanishes and as $\{1\}$ is a basis for $\mathcal{P}^0(I_m)$, then (2.13) becomes

$$\int_{t_{m-1}}^{t_m} \mathbf{a}(t, \mathbf{u}_m^{\mathsf{h}}) \,\mathrm{d}t + \mathbf{u}_m^{\mathsf{h}} = \mathbf{u}_{m-1}^{\mathsf{h}} + \int_{t_{m-1}}^{t_m} \mathbf{f} \,\mathrm{d}t \quad \text{where} \quad \mathbf{u}_0^{\mathsf{h}} := \mathbf{u}_0. \tag{2.14}$$

The dG(0) approximation $\mathbf{u}^{\mathsf{h}}$ is determined by the $M$ nodal values $\mathbf{u}_m^{\mathsf{h}}$ which can be computed sequentially for $m = 1, \ldots, M$ using (2.14). We point out that if $\mathbf{f}$ is constant and also $\mathbf{a}$ is independent of $t$ then (2.14) becomes

$$\frac{\mathbf{u}_m^{\mathsf{h}} - \mathbf{u}_{m-1}^{\mathsf{h}}}{t_m - t_{m-1}} = \mathbf{f} - \mathbf{a}(\mathbf{u}_m^{\mathsf{h}}), \tag{2.15}$$

which is equivalent to the backward Euler method. The left-hand-side of (2.15) is indeed the (backward Euler) approximation of the first ordinary derivative $\mathrm{d}\mathbf{u}/\mathrm{d}t$.

Note that with increasing degree $q$ of the polynomial space $\mathcal{P}^q$ the number of nodal (and inter-element) values also increases. Already for $q = 1$ there are two values to be determined at each node because the requirement of continuity over element-boundaries has been dropped. As $\{1, (t - t_{m-1})/(t_m - t_{m-1})\}$ forms a basis for $\mathcal{P}^1(I_m)$ we now have

$$\mathbf{u}^{\mathsf{h}}\big|_{I_m} = \mathbf{u}_{m-1}^{\mathsf{h}^+} + (\mathbf{u}_m^{\mathsf{h}^-} - \mathbf{u}_{m-1}^{\mathsf{h}^+})\frac{t - t_{m-1}}{h_m}, \text{ where as before } h_m = t_m - t_{m-1},$$

see Figure 2.3. Hence (2.13) now becomes a system of $2N$ equations which is actually composed of $N$ separate systems of two equations each. This has to be solved for the unknowns $\mathbf{u}_{m-1}^{\mathsf{h}^+}$ and $\mathbf{u}_m^{\mathsf{h}^-}$, that is the $2N$ unknowns $u_{n,m-1}^{\mathsf{h}^+}$ and $u_{n,m}^{\mathsf{h}^-}$ where $n = 1, \ldots, N$. Explicitly equation (2.13) becomes

$$\begin{aligned}
\int_{t_{m-1}}^{t_m} \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}\big|_{I_m}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m})\,\mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} &= \int_{t_{m-1}}^{t_m} \mathbf{f}\,\mathrm{d}t, \\
\int_{t_{m-1}}^{t_m} \left(\frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}\big|_{I_m}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m})\right)\frac{t - t_{m-1}}{h_m}\,\mathrm{d}t &= \int_{t_{m-1}}^{t_m} \mathbf{f}\frac{t - t_{m-1}}{h_m}\,\mathrm{d}t.
\end{aligned}$$

We can compute the integrals involving the derivative of $\mathbf{u}^{\mathsf{h}}\big|_{I_m}$, rearrange the terms and conclude:

The dG(1) approximation $\mathbf{u}^{\mathsf{h}}$ is determined by the $2M$ nodal values

$\mathbf{u}_{m-1}^{\mathsf{h}^+}$ and $\mathbf{u}_m^{\mathsf{h}^-}$ that satisfy for $m = 1, \ldots, M$ where $\mathbf{u}_0^{\mathsf{h}^-} := \mathbf{u}_0$,

$$\begin{aligned}
\mathbf{u}_m^{\mathsf{h}^-} + \int_{t_{m-1}}^{t_m} \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m})\,\mathrm{d}t &= \mathbf{u}_{m-1}^{\mathsf{h}^-} + \int_{t_{m-1}}^{t_m} \mathbf{f}\,\mathrm{d}t, \\
-\mathbf{u}_{m-1}^{\mathsf{h}^+} + \mathbf{u}_m^{\mathsf{h}^-} + \frac{2}{h_m}\int_{t_{m-1}}^{t_m} \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}\big|_{I_m})(t - t_{m-1})\,\mathrm{d}t &= \frac{2}{h_m}\int_{t_{m-1}}^{t_m} \mathbf{f}(t - t_{m-1})\,\mathrm{d}t.
\end{aligned}$$

### Stability of continuous and discontinuous Galerkin methods

In this thesis we will deal with initial value problems whose solutions will be approximated by FE numerical solutions. The error, defined as the difference between the solution and its approximation, is expected to get larger as time progresses since the error is accumulative and sensitive to earlier perturbations. Since a numerical solution such as a Galerkin approximation is also a perturbation of the solution we speak of the stability property of the numerical method. Whether the discontinuous Galerkin method is chosen or the continuous Galerkin method ultimately depends on the properties of the continuous system of ODEs and the choice of discretisation. It can be shown that for specific cases the cG($q$) method is more accurate, as measured by size of the perturbation that discretisation has introduced, but that the dG($q$) method has better stability properties. We will not go into detail here and refer to the literature, see Eriksson [5] (Section 9.2.1) or Strang [9] (Chapter 2).

## 2.4 *A Posteriori* Error Estimation

As we have to work with numerical approximations to the unknown true solution, we would like to estimate how 'good' an approximation the computed solution is. Measuring this as a norm of the error, defined as the difference between the true solution and the approximate solution[1], is widespread in the literature, see for instance the introduction to [3] and references therein, especially to the work by Babuška and Rheinboldt from the late 1970's and the 1980's. In many applications however one is more interested in an observed output than in all the details of the solution. This can be measured by an 'output' functional or, in more technical terms, a (linear) functional of the solution. Giles and Süli give a nice overview in [10] where they also describe its use for *a posteriori* error estimation based on a duality argument which involves an adjoint (also called dual) problem.

[1] Although the error involves the unknown solution, the norm (to be defined) of the error is nevertheless an *a posteriori* estimate since it is estimated using the known (computed) numerical approximation.

### 2.4.1 Linear Functionals of the Solution

A linear functional is a linear map $J$ from the solution space, a Hilbert space $\boldsymbol{\mathcal{H}}$, into the real numbers, so

$$J\colon \boldsymbol{\mathcal{H}} \to \mathbb{R} \text{ with } J(\eta\mathbf{u}+\mathbf{v}) = \eta J(\mathbf{u}) + J(\mathbf{v}) \text{ for all } \mathbf{u},\mathbf{v} \in \boldsymbol{\mathcal{H}},\ \eta \in \mathbb{R}.$$

Thus $J$ belongs to the dual space $\boldsymbol{\mathcal{H}}^*$, and by the Riesz representation theorem (see for instance [11], Section 2.4) we can always write a linear functional as an inner product, meaning that there exists $\mathbf{g} \in \boldsymbol{\mathcal{H}}$ such that

$$J(\mathbf{u}) = \big\langle \mathbf{u}\,, \mathbf{g} \big\rangle. \tag{2.16}$$

This can be seen as a weighted average of $\mathbf{u}$ over the computational domain with weighting function $\mathbf{g}$. Now, instead of computing $\|\mathbf{u}-\tilde{\mathbf{u}}\|$, where $\mathbf{u}$ is the solution and $\tilde{\mathbf{u}}$ is an approximation thereof, we look at the quantity

$$|J(\mathbf{u}) - J(\tilde{\mathbf{u}})| = |\big\langle \mathbf{u}\,, \mathbf{g} \big\rangle - \big\langle \tilde{\mathbf{u}}\,, \mathbf{g} \big\rangle| = |\big\langle \mathbf{u}-\tilde{\mathbf{u}}\,, \mathbf{g} \big\rangle|, \tag{2.17}$$

which we call the error induced in the linear functional of the solution when $\mathbf{u}$ is replaced bu $\tilde{\mathbf{u}}$, or simply the 'induced error' if it is clear from the context what we mean. Although the right-hand-side of (2.17) still depends on (the usually unknown) $\mathbf{u}$ we can make this dependency 'disappear' with an appropriate dual problem as we show (in sketch form) in the next section.

### 2.4.2 The Dual Problem

Say we computed a FE approximation $\mathbf{u}^{\mathsf{h}}$ to the solution $\mathbf{u}$ of the differential equation

$$\boldsymbol{\mathcal{D}}\mathbf{u} = \mathbf{f},$$

see (2.3) above. As we approximate $J(\mathbf{u})$ by $J(\mathbf{u}^\mathsf{h})$ since the solution $\mathbf{u}$ is unknown, we want an estimate of the induced error

$$|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| = |\langle \mathbf{u} - \mathbf{u}^\mathsf{h}, \mathbf{g} \rangle|,$$

where $\mathbf{g}$ is the vector-valued function that arises from the Riesz representation (2.16) of the linear functional as an inner product. This $\mathbf{g}$ is used[2] to define the dual problem:

$$\boldsymbol{\mathcal{D}}^* \boldsymbol{\varphi} = \mathbf{g}, \tag{2.18}$$

where $\boldsymbol{\mathcal{D}}^*$ is the adjoint of $\boldsymbol{\mathcal{D}}$ (including appropriate boundary conditions). With the dual problem (2.18) we can write

$$\begin{aligned} J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h}) &= \langle \mathbf{u}, \mathbf{g} \rangle - \langle \mathbf{u}^\mathsf{h}, \mathbf{g} \rangle, \\ &= \langle \mathbf{u} - \mathbf{u}^\mathsf{h}, \mathbf{g} \rangle, \\ &= \langle \mathbf{u} - \mathbf{u}^\mathsf{h}, \boldsymbol{\mathcal{D}}^* \boldsymbol{\varphi} \rangle, \\ &= \langle \boldsymbol{\mathcal{D}}(\mathbf{u} - \mathbf{u}^\mathsf{h}), \boldsymbol{\varphi} \rangle, \\ &= \langle \mathbf{f} - \boldsymbol{\mathcal{D}} \mathbf{u}^\mathsf{h}, \boldsymbol{\varphi} \rangle. \end{aligned}$$

We therefore found that the error induced in the linear functional can be written as

$$|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| = |\langle \mathrm{Res}(\mathbf{u}^\mathsf{h}), \boldsymbol{\varphi} \rangle|, \tag{2.19}$$

where the residual error is defined as $\mathrm{Res}(\mathbf{u}^\mathsf{h}) := \boldsymbol{\mathcal{D}} \mathbf{u}^\mathsf{h} - \mathbf{f}$. Note that therefore the induced error is small not only when $\mathbf{u}^\mathsf{h}$ is a good approximation to $\mathbf{u}$, meaning that $\mathrm{Res}(\mathbf{u}^\mathsf{h})$ is small, but also if $\boldsymbol{\varphi}$ is 'almost orthogonal' to the residual error.

It seems that (2.19) is now independent of the primal solution $\mathbf{u}$. We will see however that often (meaning for nonlinear problems) the adjoint operator does involve $\mathbf{u}$ and

[2] This is not true in all cases; one exception is for instance when $\mathbf{g}$ contains (Dirac) delta functions. In that case the dual problem has to be slightly altered but the outcome will ultimately be the same. We only give a rough outline here as we will come back to the subject matter in greater detail in Chapter 3, Section 3.2 in particular, and also in Chapters 4 and 5 where the duality argument is used.

therefore has to be approximated. Furthermore, just as $\mathbf{u}$ has to be approximated by $\mathbf{u}^\mathsf{h}$ so does the dual solution $\boldsymbol{\varphi}$ by an appropriate FE approximation $\boldsymbol{\varphi}^\mathsf{h}$. For these reasons we cannot compute the induced error exactly as in (2.19) but just approximately,

$$|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| \approx |\langle \mathrm{Res}\,(\mathbf{u}^\mathsf{h})\,, \boldsymbol{\varphi}^\mathsf{h} \rangle|, \tag{2.20}$$

using $\boldsymbol{\varphi}^\mathsf{h}$ instead of the true solution $\boldsymbol{\varphi}$ of an appropriate dual problem that is in turn also just an approximation of the 'true' dual problem. Even with all these approximations it turns out that the *a posteriori* error estimate using this duality argument is a useful tool for mesh refinement and, as we will show in this thesis, also for model reduction.

### 2.4.3 Galerkin Orthogonality

Recall that the Galerkin FE approximation $\mathbf{u}^\mathsf{h}$ in a finite-dimensional subspace $\boldsymbol{\mathcal{S}}^h$ of the trial space $\boldsymbol{\mathcal{H}}$ satisfies

$$\langle \boldsymbol{\mathcal{D}}\mathbf{u}^\mathsf{h}\,, \mathbf{v}^\mathsf{h} \rangle = \langle \mathbf{f}\,, \mathbf{v}^\mathsf{h} \rangle \tag{2.21}$$

for all $\mathbf{v}^\mathsf{h}$ in a finite-dimensional subspace $\boldsymbol{\mathcal{V}}^h$ of the test space $\boldsymbol{\mathcal{V}}$. Clearly (2.21) also holds if we substitute $\mathbf{u}^\mathsf{h}$ for $\mathbf{u}$, meaning that $\mathbf{u}$ satisfies

$$\langle \boldsymbol{\mathcal{D}}\mathbf{u}\,, \mathbf{v}^\mathsf{h} \rangle = \langle \mathbf{f}\,, \mathbf{v}^\mathsf{h} \rangle \tag{2.22}$$

also for all $\mathbf{v}^\mathsf{h} \in \boldsymbol{\mathcal{V}}^h$. Since the right-hand-sides of (2.21) and (2.22) are identical, and because $\boldsymbol{\mathcal{D}}\mathbf{u} = \mathbf{f}$, subtracting the former from the latter yields

$$\langle \boldsymbol{\mathcal{D}}\mathbf{u}^\mathsf{h} - \mathbf{f}\,, \mathbf{v}^\mathsf{h} \rangle = \langle \mathrm{Res}\,(\mathbf{u}^\mathsf{h})\,, \mathbf{v}^\mathsf{h} \rangle = 0. \tag{2.23}$$

This is saying that the residual error is (Galerkin) orthogonal to the test space $\boldsymbol{\mathcal{V}}^h$.

Thus, if we would compute the finite element approximation $\boldsymbol{\varphi}^{\mathsf{h}}$ to the solution $\boldsymbol{\varphi}$ of the dual problem (2.18) in the same space $\boldsymbol{\mathcal{V}}^h$, then the right-hand-side of (2.20) would vanish due to Galerkin orthogonality. To prevent this we have to choose a trial space to compute a FE approximation $\boldsymbol{\varphi}^{\mathsf{h}}$ to the dual problem's solution that is different to the test space used to compute a FE approximation $\mathbf{u}^{\mathsf{h}}$ to the primal problem's solution. Two common choices are either to use different piecewise polynomial spaces and use the same mesh, or to change the underlying mesh but use the same degree of piecewise polynomials. What choice will ultimately be made will be stated when need arises.

### 2.4.4 Adaptive Mesh Refinement

The *a posteriori* error (estimate) can also be used to adaptively refine the mesh to improve the accuracy of the FE approximation $\mathbf{u}^{\mathsf{h}}$. The underlying idea involves writing the right-hand-side of (2.19) or (2.20) as a sum over the elements whose union is the discretised computational domain. The elements that make the largest contribution to that sum are located and the mesh is then refined accordingly, meaning that locally (where the contributions were largest) the new mesh will contain more elements. Then one computes $\mathbf{u}^{\mathsf{h}}$ anew based on this refined mesh and repeats the process if necessary until one is satisfied with the estimated accuracy of the induced error (2.20).

When FE approximations are computed on adaptively refined meshes with varying element sizes (given by the 'size function' $h$) then this is generally denoted by $h-$FEM. This distinguishes it from $p-$FEM where $p$ stands for the polynomial degree of the piecewise polynomials that form a (elementwise) basis for the finite-dimensional solution space. The former seeks to increase the accuracy of the FE approximation by computation on a (locally) finer mesh whereas the latter seeks to achieve this by (locally) increasing the degree of the interpolating polynomials. Both variations can be combined to give $hp-$FEM which was first introduced in the early 1990's by Babuška and Guo [12]. We will use a form of $h-$FEM in this thesis in Chapter 5, but our main contribution which is model reduction is independent of mesh refinement.

# CHAPTER 3

# Model Reduction Review

A mathematical model is the result of the translation of a description of a natural phenomenon into mathematical language. This model can then be analysed and interpreted, and checked against the data arising from the process it is supposed to model. The process of translation into mathematical language is not straightforward, since we usually do not know what causes an outcome. We can roughly speak of two approaches to arrive at a mathematical model, the 'top-down' and 'bottom-up' approach. The former will include every possible mechanism that could potentially have a bearing on the outcome [13], the latter will start with a simple model and consecutively include more terms in some of the equations until the outcome is satisfactory [14]. No matter which approach is taken, the goal is to end up with a model that captures the important underlying features of the phenomenon and produces reliable output for further analysis.

Not only the 'top-down' but also the 'bottom-up' approach can lead to fairly large systems of differential equations, since it might be necessary to include many variables in order to model and explain observed behaviours in the most general terms. But this can render the model opaque and may obscure its key features. It is of course true that even very large systems of equations can be solved numerically, but a numerical solution won't necessarily identify what terms and therefore what physical phenomena are responsible for a given behaviour. In this case, some form of model reduction would be welcome, to enable the user to analyse the model and gain further insights about the processes the model seeks to emulate. We will now review some of the standard approaches for model simplification or reduction, ending with a section on model reduction by a duality argument. This last section includes an outline of preliminary work and the original idea by Whiteley as presented in [4].

The underlying idea is to use a duality argument similar to the one described at the

end of the previous chapter in Section 2.4. There we gave a rough description how the error induced in the linear functional when the FE approximation is substituted for the true solution is used to ultimately get a better approximation by refining the mesh. One can apply the same principle using the error induced in the linear functional when the reduced model solution is substituted for the true solution to ultimately get a good approximation to the original differential operator. Whiteley's method put a lot of restrictions not only on the reduced model solution (such as continuity requirements) but also on the form the reduced model operator can take. It was furthermore assumed that the true solutions of the (full and the reduced) models are 'known' by using very fine meshes (and then using these numerical 'solutions' as the 'true' solutions).

We extended the model reduction method by removing the restrictions imposed on the form of the reduced model operator. Also the assumptions about the nature of the original and reduced model solutions were relaxed in as far that discontinuous FE approximations were admitted and by working with FE approximations throughout. This made very fine meshes not only no longer necessary, which already reduces the computational effort, but also increased the 'efficiency' of the model reduction method through combination of model reduction with mesh refinement. These extensions led to novel analysis which is presented in detail in the main research chapter which is Chapter 4. As some of it is based on preliminary work we expanded the analysis of the original method and also of a first extension thereof at the end of this chapter.

## 3.1 Approaches to Model Reduction

### 3.1.1 Perturbation Methods

Perturbation methods are an established tool to analyse mathematical models and elucidate their key features, and are a standard methodology in applied mathematics. These methods are often used to approximate multi-scale problems, examples from mathematical biology include models for cell motility [15], signalling pathways [16], tissue growth [17], tumour growth [18], heart and nerve excitability [19] and many

others. Models are simplified in the first instance by non-dimensionalisation, and then, including only dominant terms and possible boundary layers, through asymptotic expansions. Even though these methods yield the desired information about a mathematical model, they are nevertheless not easy to apply, especially with increasing complexity of the model. Furthermore the process needs advanced mathematical expertise, requires significant attention to the finer details, and each problem may require different expansions depending on the nature of the perturbed problem. It is clearly not automatic in the sense that it could be translated into an algorithm when given a generic model. Even simple initial value problems like the following,

$$\varepsilon u' + u = 1 \quad \text{with} \quad u(0) = 0, \quad \text{and}$$
$$u' + \varepsilon u = 1 \quad \text{with} \quad u(0) = 0,$$

suffice to make it clear that asymptotic expansions differ in almost every case. The naïve approach, setting $\varepsilon = 0$ in the first problem, yields the constant solution $u(x) \equiv 1$ which can never satisfy the initial condition. The correct assumption would be rather, that $u \approx 0$ near the origin and thus $\varepsilon u' \approx 1$ there, whence $u(x) \approx x/\varepsilon$ for $0 < x < \varepsilon \ll 1$. The second example on the other hand is a regular perturbation problem and the asymptotic expansion in integral powers of $\varepsilon$, namely $u(x) \approx \sum_{k=0}^{\infty} (-1)^k \frac{x^{k+1}}{(k+1)!} \varepsilon^k$ converges to the true solution $u(x) = (1 - \exp(-\varepsilon x))/\varepsilon$. These simple examples both have an exact solution, but most of the differential equations encountered in applied mathematics do not have solutions that can be written down. When an exact solution is not available we would still like to have some information, at least about the local behaviour in certain regions of the computational domain. Asymptotic analysis can often give this information, but at a price. One of our goals is, to find an automated way that yields the same information as obtained through asymptotic analysis, thus informing the user about key aspects modelled by the system at hand.

### 3.1.2 Model Order Reduction

We speak of order reduction if the number of equations of a system of (algebraic or differential) equations is reduced. A result of the reduction of the order is either a phenomenological model, that is a model that still captures a feature of interest of the phenomenon but loses a real-life correspondence, or it still retains the key processes and real-life correspondence while discarding less important processes.

A famous example for order reduction resulting in a phenomenological model starts with the Hodgkin–Huxley equations [20] that model the propagation of an impulse in a nerve fibre. The electric potential difference across the nerve cell membrane at each point on the fibre is due to ions entering and leaving via ion channels. Hodgkin and Huxley's model thus includes an equation for the action potential, that is the time evolution of that electric potential difference, and three equations are required to model ion transport through the membrane's ion channels. The FitzHugh–Nagumo model is a reduced order model of the Hodgkin–Huxley model, where the ion exchange dynamics has been simplified by a single equation.

The original order four model captures the excitability property, that is the ability of a nerve cell to respond to a stimulus. Say if $V$ denotes the action potential with $V_0$ being the steady state, then there is a value $V_1 > V_0$ such that if $V$ is perturbed to $V^*$ with $V_0 < V^* < V_1$, then the solution decays to the steady state. If the perturbation is only a little bit larger, that is $V^* > V_1$ then this small perturbation in the membrane potential instigates a large sequence of biochemical reactions before returning to the steady state. The order two FitzHugh–Nagumo model still captures the excitability behaviour of the action potential but will no longer be able to feature chaotic dynamics and neuron bursting. The Hindmarsh–Rose model [21] on the other hand reduces the order of the Hodgkin–Huxley only by one, but it can capture rapid neuron firing.

A description of the various mathematical models of neurons, including Hodgkin–Huxley and the aforementioned reduced order models can be found in [22]. We will encounter the Hodgkin–Huxley model again in Chapter 5 where we attempt to reduce

the model further with our model reduction algorithm.

Order reduction can also be achieved in an automated way, but is usually reserved for systems of linear ordinary differential equations. The method is used in systems control theory, but it could also be tailored to the needs of computational biologists when confronted with a large order mathematical model that consists of linear ODEs. Although hardly any models in biology are linear, it could still apply to some form of linearised models. The method can be described as 'transform and truncate' [23] in the following sense. We start with an original system with $n$ equations, where the solution $\mathbf{u}$ belongs to the $n$-dimensional space of vector-valued functions. We can write this as

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{A}\mathbf{u} = \mathbf{f}, \quad \mathbf{u}(0) = \mathbf{u}_0, \tag{3.1}$$

where $\mathbf{u}$ is the $n$-dimensional solution vector, $\mathbf{f}$ is the input (or we can regard $\mathbf{f}$ as the forcing function), and $\mathbf{A}$ is a $n \times n$ matrix. To arrive at a reduced model we choose $n \times r$ and $n \times s$ matrices $\mathbf{T}_r$, $\mathbf{R}_r$ and $\mathbf{T}_s$, $\mathbf{R}_s$ respectively with $r + s = n$ and $s \geq 1$, such that

$$\mathbf{T}_r\mathbf{R}_r^\mathsf{T} + \mathbf{T}_s\mathbf{R}_s^\mathsf{T} = \mathbf{I}_n, \tag{3.2}$$

which yields the similarity transformation

$$\mathbf{T}_r\mathbf{u}_r + \mathbf{T}_s\mathbf{u}_s = \mathbf{u}, \quad \text{and} \quad \mathbf{u}_r = \mathbf{R}_r^\mathsf{T}\mathbf{u},\ \mathbf{u}_s = \mathbf{R}_s^\mathsf{T}\mathbf{u}. \tag{3.3}$$

There are several ways to define such matrices meeting this condition. Most algorithms seem to be based on singular value decomposition, some of which are described in [23].

Transforming $\mathbf{u}$ also transforms the original system (3.1) and we get

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{u}_r\\ \\ \mathbf{u}_s\end{bmatrix} + \begin{bmatrix}\mathbf{R}_r^\mathsf{T}\mathbf{A}\mathbf{T}_r & \mathbf{R}_r^\mathsf{T}\mathbf{A}\mathbf{T}_s\\ \\ \mathbf{R}_s^\mathsf{T}\mathbf{A}\mathbf{T}_r & \mathbf{R}_s^\mathsf{T}\mathbf{A}\mathbf{T}_s\end{bmatrix}\begin{bmatrix}\mathbf{u}_r\\ \\ \mathbf{u}_s\end{bmatrix} = \begin{bmatrix}\mathbf{R}_r^\mathsf{T}\\ \\ \mathbf{R}_s^\mathsf{T}\end{bmatrix}\mathbf{f}.$$

Now we truncate the transformed system and end up with a system of order $r < n$,

$$\frac{\mathrm{d}\mathbf{u}_r}{\mathrm{d}t} + (\mathbf{R}_r^\top \mathbf{A}\mathbf{T}_r)\mathbf{u}_r = \mathbf{R}_r^\top \mathbf{f}, \quad \mathbf{u}_r(0) = \mathbf{R}_r^\top \mathbf{u}_0. \tag{3.4}$$

The magnitude of the reduction is dependent on the choice of similarity transformation and results in the different algorithms described in the literature, see for instance [24] for an introductory overview.

Control theory is widely used in the engineering sciences to design efficient systems that are composed of many components. A system might be an industrial plant, or an electronic device composed of many circuits, or a power supply network. The credo of systems design seems to be 'that good component models do not imply good models for system design purposes' (Skelton et al. in [23]). In systems biology, say in modelling signalling pathways, biologists have traditionally studied the components of the network separately. Introducing control theory into the field of systems biology would enable life scientists to understand the whole signalling system. There have been efforts to introduce control theory into systems biology and it is being used by researchers in this area [25–27].

Note that the method described above, reducing (3.1) to (3.4), uses the fact that the solution vector can be written as a linear combination of basis vectors for the solution space $\boldsymbol{\mathcal{V}}$. In the particular case above, $\boldsymbol{\mathcal{V}}$ was decomposed into two subspaces and only the projection onto one of the subspaces was retained to gain a reduced order model. Reduced Basis Methods and Proper Orthogonal Decomposition are amongst the emerging fields to cope with high-order models and offer ways of reducing the order for more general classes of differential equations. The approach is not too different for a system of non-linear ODEs, say for a system given as

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}) = \mathbf{f}, \quad \mathbf{u}(0) = \mathbf{u}_0. \tag{3.5}$$

We can now choose a trial set of basis vectors $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\} \subset \boldsymbol{\mathcal{V}}$ and express the

solution of (3.5) as a linear combination of these vectors,

$$\mathbf{u} \approx \sum_{i=1}^{k} p_i(t)\mathbf{w}_i, \quad \text{or in matrix form} \quad \mathbf{u} \approx \mathbf{W}\mathbf{p}(t). \tag{3.6}$$

The vector $\mathbf{p}(t) = (p_1(t), \ldots, p_k(t))^{\mathsf{T}}$ is called the reduced solution vector, and now we would just like to find a projection to obtain the reduced order model

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{p}) = \mathbf{f}_r, \quad \text{with appropriate initial conditions.} \tag{3.7}$$

There have been several papers published recently on this emerging topic as well as textbooks and symposia dedicated to it [28–31]. It is certainly worthwhile to explore this avenue of model reduction further.

### 3.1.3 Other Methods

A dynamical systems approach has been used to automatically simplify chemical kinetics [32]. It is a method to analyse the long-term qualitative behaviour whereby no assumptions have to be made with regard to the quasi-steady state of the solution components. In contrast to this approach is linearisation theory [33] where one assumes that the non-linear system behaves locally just as its linearised counterpart at the equilibrium points. Models have also been simplified by substituting algebraic equations for some of the differential equations in the system [34, 35]. It is often the case that a model's equations contain many parameters whose values are uncertain. A model output might change significantly depending on parameter estimation. Sensitivity analysis is a commonly used tool to study the effect of parameter choices on the outcome of interest [36–38] and it can in principle lead to a simplified model via reduction of the number of parameters.

All these are of course valid approaches and do usually make the solution of large systems more tractable. On the other hand they usually lack the feature of being automatic and accessible for an algorithmic approach, with the exception of a computational tool described in [39] that reduces multi-scale systems of ODEs. One alternative to

these authors' approach might be used to simplify a given mathematical model while at the same time retaining the original 'input-output' features. This idea cropped up while researching how mathematical models can be identified from measurements and experimental data. The method known as systems identification is a tool that recovers, or more accurately, gives insight into the structure of the underlying mathematical model. We gain information whether it is a linear process, or highly non-linear for instance, and this from given input and measured output. See for instance [40] for such a method description and [41] for some examples of model recovery. Although the idea seems a bit far-fetched, there might be a way to use model recovery for model simplification. The 'reverse engineered' idea is, that if it is the case that systems cannot be uniquely determined from given input-output data, then we could maybe 'recover' or reconstruct a simpler or reduced system with approximately the same input-output behaviour as the original given model.

## 3.2 Model Reduction by Duality

An alternative method to achieve simplification and elucidation of key features of a mathematical model is the method of *a posteriori* error analysis using a duality argument. As previously mentioned, the method has been developed in the framework of FE methods to estimate the error between the FE approximation and the true solution. It has also been taken out of this context and has been used to automatically simplify a model, first by Whiteley [4] for ODE systems and extended for simple parabolic PDEs by Maybank and Whiteley [42], so that the error induced in the linear functional when replacing the solution to the full model with the solution to the simplified model is less than some prescribed tolerance. Whereas the FE approach involved the bounding of an error defined as the difference between the true solution of the full model and its FE approximation, these authors worked with an error defined as the difference between the true solution of the full model and the numerical approximation of the solution to a reduced model.

That is, both techniques use *a posteriori* analysis in the same spirit, although with

different motivations. *A posteriori* error analysis is used in FE methods to ensure that the mesh on the computational domain is fine enough such that the discretised scheme approximates the differential operator sufficiently accurately. Model reduction on the other hand generates a differential operator that is a 'close enough' approximation of the original operator.

The model reduction algorithm introduced by Whiteley achieves a reduction of systems of differential equations, such that only a subset of the original set of equations in the system has to be solved on subsets of the computational domain. This subset of equations and subsets of the computational domain are chosen in such a way that a specified quantity of interest can be calculated to within a given accuracy without having to solve the full system of equations everywhere. A key feature of the method is that this accuracy can be automated using *a posteriori* error analysis. This allows for different reduced models to be generated for various observed behaviours via a user-defined (linear) functional of the quantity of interest.

The general framework for this model reduction technique applied to a

$$\text{primal problem} \quad \boldsymbol{\mathcal{D}}\mathbf{u} = \mathbf{f}, \tag{3.8}$$

where $\mathbf{u}$ denotes the true solution (not a numerical approximation), is as follows. Let $\mathbf{U}$ denote the solution of a

$$\text{reduced problem} \quad \widehat{\boldsymbol{\mathcal{D}}}\mathbf{U} = \mathbf{f}, \tag{3.9}$$

which is a system of differential equations that is a simplification of the primal problem. Suppose that we are interested in a

$$\text{linear functional} \quad J(\mathbf{u}) = \big\langle \mathbf{u}\,, \mathbf{g} \big\rangle, \tag{3.10}$$

which is a functional of the solution and defines the user-defined quantity of interest. Now, to have a measure for how well the reduced model solution $\mathbf{U}$ approximates the

solution $\mathbf{u}$ of the full model, we define the 'error' not as $\mathbf{u}-\mathbf{U}$ but as $J(\mathbf{u})-J(\mathbf{U})$. This is justifiable on the grounds that we are interested in a linear functional of the solution and hence we would like to know how 'close' the linear functionals of the respective solutions are. We call this error 'the error induced in the linear functional when the true solution is replaced by an approximation' which we sometimes abbreviate in the text as 'the functional error' or 'the induced error' when it is clear from the context what we mean by it. To compute this we suppose furthermore that $\boldsymbol{\varphi}$ solves the

$$\text{dual problem} \quad \boldsymbol{\mathcal{D}}^*\boldsymbol{\varphi} = \mathbf{g}, \tag{3.11}$$

where the adjoint operator is defined such that $\left\langle \mathbf{u}-\mathbf{U}\,, \boldsymbol{\mathcal{D}}^*\boldsymbol{\varphi} \right\rangle = \left\langle \boldsymbol{\mathcal{D}}\mathbf{u}-\boldsymbol{\mathcal{D}}\mathbf{U}\,, \boldsymbol{\varphi} \right\rangle$. This usually means that the adjoint $\boldsymbol{\mathcal{D}}^*$ is dependent on $\mathbf{u}$ and also on $\mathbf{U}$ and as such a change in $\mathbf{U}$ entails a change in the dual problem[1]. Then

$$\begin{aligned} J(\mathbf{u})-J(\mathbf{U}) &= \left\langle \mathbf{u}-\mathbf{U}\,, \mathbf{g} \right\rangle, \\ &= \left\langle \mathbf{u}-\mathbf{U}\,, \boldsymbol{\mathcal{D}}^*\boldsymbol{\varphi} \right\rangle, \\ &= \left\langle \boldsymbol{\mathcal{D}}\mathbf{u}-\boldsymbol{\mathcal{D}}\mathbf{U}\,, \boldsymbol{\varphi} \right\rangle, \\ &= \left\langle \mathbf{f}-\boldsymbol{\mathcal{D}}\mathbf{U}\,, \boldsymbol{\varphi} \right\rangle, \\ &= -\left\langle \mathrm{Res}\,(\mathbf{U})\,, \boldsymbol{\varphi} \right\rangle. \end{aligned}$$

Thus $|J(\mathbf{u})-J(\mathbf{U})|$ is small if the residual error $\mathrm{Res}\,(\mathbf{U})$ is 'almost orthogonal' to $\boldsymbol{\varphi}$ or if $\mathrm{Res}\,(\mathbf{U})$ is small which means that $\mathbf{U}$ approximates $\mathbf{u}$ well. The model reduction technique by Whiteley [4] is based on this *a posteriori* bound on the error induced in the linear functional. We will now explain in more detail how the model reduction technique works.

[1]This happens if and only if $\boldsymbol{\mathcal{D}}$ is non-linear. If $\boldsymbol{\mathcal{D}}$ is linear then $\boldsymbol{\mathcal{D}}^*$ is independent of $\mathbf{u}$ and therefore also of $\mathbf{U}$.

### 3.2.1 Model Reduction for Systems of ODEs

We will start with an outline of what has been done previously. Whiteley in [4] has been looking at systems of ordinary differential equations of the form

$$\text{Find } \mathbf{u}(t) = (u_1(t), \ldots, u_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T) \text{ satisfying } \mathbf{u}(0) = \mathbf{u}_0 \text{ such that}$$
$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t,\mathbf{u}) = \mathbf{0} \quad \text{for} \quad 0 < t \leq T. \tag{3.12}$$

Suppose we have a user-defined linear functional of the solution $J(\mathbf{u}) = \langle \mathbf{u}\,, \mathbf{g} \rangle$. Let $\mathbf{U}$ denote the solution (actually a numerical approximation on a very fine mesh) of a simplification of the primal problem (3.12). Our interest lies with the error that is induced in the linear functional when using this approximation (solution of the reduced model) instead of the solution to the (original) full model, so we want a bound on the quantity $|J(\mathbf{u}) - J(\mathbf{U})|$. The original model reduction technique in [4] requires that $\mathbf{U}(t) \in \boldsymbol{\mathcal{C}}(0,T)$ and also that $\mathrm{d}\mathbf{U}/\mathrm{d}t$ is continuous except for a finite set of values $\{t_0, \ldots, t_M \colon 0 = t_0 < t_1 < \cdots t_{M-1} < t_M = T\}$. Furthermore, $\mathbf{U}$ should also satisfy the initial condition $\mathbf{U}(0) = \mathbf{u}_0$ of the primal problem. The bound on the error $|J(\mathbf{u}) - J(\mathbf{U})|$ is based on a duality argument, that is, the analysis makes use of a dual (sometimes also called adjoint) problem associated with the primal problem.

Depending on whether the vector-valued function $\mathbf{g}(t)$ in the definition of the linear functional is finite or not[2] at $t = T$, the dual problem has slightly different formulations. As these ultimately lead to the same error bound we will assume for the moment that $\mathbf{g}(T)$ is finite. In this case the dual problem associated with (3.12) is defined as:

$$\text{Find } \boldsymbol{\varphi}(t) = (\varphi_1(t), \ldots, \varphi_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T) \text{ satisfying } \boldsymbol{\varphi}(T) = \mathbf{0} \text{ such that}$$
$$-\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} = \mathbf{g} \quad \text{for} \quad T > t \geq 0. \tag{3.13}$$

The vector-valued function $\mathbf{g}(t)$ on the right-hand-side is the same one defining the

[2] An example for finite $\mathbf{g}$ is if one or more components of $\mathbf{g}$ are defined as $g_n = 1/T$. This means that the linear functional of the solution $\mathbf{u}$ is the average of component $u_n$ over the time interval $[0\,, T]$. On the other hand, if $J(\mathbf{u}) = u_n(T)$ meaning that we are interested in the value of the solution component $n$ at a fixed point $T$, then $\mathbf{g}$ contains a delta-function, so not all components of $\mathbf{g}$ are finite.

linear functional $J(\mathbf{u})$. The $N \times N$ entries of the adjoint matrix $\mathbf{A}^*(t)$ are functions $A^*_{ij}(t)$ defined as

$$A^*_{ij}(t) = \int_0^1 \frac{\partial a_j}{\partial u_i}(\underbrace{s\mathbf{u}(t) + (1-s)\mathbf{U}(t)}_{=:\,\boldsymbol{\alpha}(s,t)})\,\mathrm{d}s \quad \text{for } i,j = 1,\ldots,N. \tag{3.14}$$

Using the dual problem, the definition of the dual matrix, and the fact that $\mathbf{u}$, $\mathbf{U}$, $\boldsymbol{\varphi}$ and their derivatives are continuous on $(t_{m-1}\,, t_m]$ for $m = 1,\ldots,M$, we can write the error that is induced in the linear functional when $\mathbf{u}$ is replaced by $\mathbf{U}$ as:

$$\begin{aligned}
J(\mathbf{u}) - J(\mathbf{U}) &= \big\langle \mathbf{u} - \mathbf{U}\,, \mathbf{g} \big\rangle, && (3.15)\\
&= \Big\langle \mathbf{u} - \mathbf{U}\,, -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} \Big\rangle, && (3.16)\\
&= \left\langle \left(\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t,\mathbf{u})\right) - \left(\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + \mathbf{a}(t,\mathbf{U})\right)\,, \boldsymbol{\varphi} \right\rangle, && (3.17)\\
&= -\big\langle \mathrm{Res}\,(\mathbf{U})\,, \boldsymbol{\varphi} \big\rangle. && (3.18)
\end{aligned}$$

Equation (3.15) holds as per linearity of the functional $J$, and equation (3.16) by replacing $\mathbf{g}$ by the left-hand-side of the dual problem (3.13). Also (3.17) and (3.18) are equivalent as per definition of the residual error and bearing in mind that $\mathrm{d}\mathbf{u}/\mathrm{d}t + \mathbf{a}(t,\mathbf{u}) = \mathbf{0}$. To get from (3.16) to (3.17) we use the definition of the adjoint matrix. First note that

$$\begin{aligned}
\sum_{n=1}^{N} A^*_{nk}(u_n - U_n) &= \sum_{n=1}^{N} \int_0^1 \frac{\partial a_k}{\partial u_n}(t, \boldsymbol{\alpha}(s,t))\,(u_n - U_n)\,\mathrm{d}s,\\
&= \int_0^1 \sum_{n=1}^{N} \frac{\partial a_k}{\partial \alpha_n}\frac{\partial \alpha_n}{\partial s}\,\mathrm{d}s,\\
&= \int_0^1 \frac{\mathrm{d}a_k}{\mathrm{d}s}\,\mathrm{d}s,\\
&= a_k(t,\mathbf{u}) - a_k(t,\mathbf{U}). && (3.19)
\end{aligned}$$

Now we can easily get from (3.16) to (3.17), as

$$J(\mathbf{u}) - J(\mathbf{U}) = \Big\langle \mathbf{u} - \mathbf{U}\,, -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} \Big\rangle,$$

$$
\begin{aligned}
&= \int_0^T -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} \cdot (\mathbf{u} - \mathbf{U})\, \mathrm{d}t + \int_0^T (\mathbf{u} - \mathbf{U}) \cdot \mathbf{A}^* \boldsymbol{\varphi}\, \mathrm{d}t, \\
&= \sum_{m=1}^{M} \left[ -\boldsymbol{\varphi} \cdot (\mathbf{u} - \mathbf{U}) \right]_{t_{m-1}}^{t_m} + \sum_{m=1}^{M} \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} \right) \cdot \boldsymbol{\varphi}\, \mathrm{d}t \\
&\qquad + \int_0^T \sum_{n=1}^{N} (u_n - U_n)(\mathbf{A}^* \boldsymbol{\varphi})_n\, \mathrm{d}t,
\end{aligned}
$$

but as $\mathbf{u}$, $\mathbf{U}$, $\boldsymbol{\varphi}$ are continuous on $[0, T]$, and as $\mathbf{u}(0) = \mathbf{u}_0 = \mathbf{U}(0)$ and $\boldsymbol{\varphi}(T) = \mathbf{0}$, the first term that results from integration by parts vanishes and we are left with

$$
J(\mathbf{u}) - J(\mathbf{U}) = \int_0^T \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} \right) \cdot \boldsymbol{\varphi}\, \mathrm{d}t + \int_0^T \sum_{k=1}^{N} \varphi_k \sum_{n=1}^{N} A_{nk}^* (u_n - U_n)\, \mathrm{d}t.
$$

This, with (3.19), simplifies to

$$
\begin{aligned}
J(\mathbf{u}) - J(\mathbf{U}) &= \int_0^T \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} \right) \cdot \boldsymbol{\varphi}\, \mathrm{d}t + \int_0^T \sum_{k=1}^{N} \varphi_k (a_k(t, \mathbf{u}) - a_k(t, \mathbf{U}))\, \mathrm{d}t, \\
&= \int_0^T \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} \right) \cdot \boldsymbol{\varphi}\, \mathrm{d}t + \int_0^T (\mathbf{a}(t, \mathbf{u}) - \mathbf{a}(t, \mathbf{U})) \cdot \boldsymbol{\varphi}\, \mathrm{d}t, \\
&= \left\langle \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}) \right) - \left( \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{U}) \right), \boldsymbol{\varphi} \right\rangle, \\
&= -\left\langle \mathrm{Res}\,(\mathbf{U}), \boldsymbol{\varphi} \right\rangle.
\end{aligned}
$$

Now we can also deal with the case when $\mathbf{g}(t) = (g_1(t), \ldots, g_N(t))^{\mathsf{T}}$ has an entry which is not finite at $t = T$. Say there exists an $n \in \{1, \ldots, N\}$ for which $g_n(t) = \delta(t - T)$ and $g_k(t) \equiv 0$ for all $k \in \{1, \ldots, N\} \setminus \{n\}$. Then

$$
J(\mathbf{u}) = \left\langle \mathbf{u}, \mathbf{g} \right\rangle = \int_0^T u_n(t) \delta(t - T)\, \mathrm{d}t = u_n(T).
$$

In that case the dual problem associated with (3.12) is defined as:

Find $\boldsymbol{\varphi}(t) = (\varphi_1(t), \ldots, \varphi_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0, T)$ satisfying $\boldsymbol{\varphi}(T) = (\varphi_1, \ldots, \varphi_N)^{\mathsf{T}}$

where $\varphi_n(T) = 1$ and $\varphi_k(T) = 0$ for all $k \in \{1, \ldots, N\} \setminus \{n\}$ such that

$$
-\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^* \boldsymbol{\varphi} = \mathbf{0} \quad \text{for} \quad T > t \geq 0. \tag{3.20}
$$

With this we will also get (3.18) but in a slightly different way. We now have

$$\begin{aligned}
0 &= \left\langle \mathbf{u} - \mathbf{U}\,, \mathbf{0} \right\rangle, \\
&= \left\langle \mathbf{u} - \mathbf{U}\,, -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} \right\rangle, \\
&= \int_0^T -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} \cdot (\mathbf{u} - \mathbf{U})\,\mathrm{d}t + \int_0^T (\mathbf{u} - \mathbf{U}) \cdot \mathbf{A}^*\boldsymbol{\varphi}\,\mathrm{d}t, \\
&= \sum_{m=1}^{M} \left[ -\boldsymbol{\varphi} \cdot (\mathbf{u} - \mathbf{U}) \right]_{t_{m-1}}^{t_m} + \sum_{m=1}^{M} \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} \right) \cdot \boldsymbol{\varphi}\,\mathrm{d}t \\
&\qquad + \int_0^T \sum_{n=1}^{N} (u_n - U_n)(\mathbf{A}^*\boldsymbol{\varphi})_n\,\mathrm{d}t,
\end{aligned}$$

but now the continuity of $\mathbf{u}$, $\mathbf{U}$, $\boldsymbol{\varphi}$ and the values of $\mathbf{u}(0)$, $\mathbf{U}(0)$ and $\boldsymbol{\varphi}(T)$ do not make the first term that results from integration by parts vanish but make it simpler, and the integral terms simplify as before, so we get

$$0 = -(u_n - U_n) + \int_0^T \left( \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} - \frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} \right) \cdot \boldsymbol{\varphi}\,\mathrm{d}t + \int_0^T \sum_{k=1}^{N} \varphi_k (a_k(t, \mathbf{u}) - a_k(t, \mathbf{U}))\,\mathrm{d}t,$$

and therefore, using the definition of the linear functional and the residual error,

$$0 = -(J(\mathbf{u}) - J(\mathbf{U})) - \left\langle \mathrm{Res}\,(\mathbf{U})\,, \boldsymbol{\varphi} \right\rangle.$$

Thus in either case we derive the following bound,

$$\begin{aligned}
|J(\mathbf{u}) - J(\mathbf{U})| &= |\left\langle \mathrm{Res}\,(\mathbf{U})\,, \boldsymbol{\varphi} \right\rangle|, \\
&= \left| \sum_{n=1}^{N} \sum_{m=1}^{M} \int_{t_{m-1}}^{t_m} Res_{\,n}(\mathbf{U}) \varphi_n\,\mathrm{d}t \right|, \\
&\leq \sum_{n=1}^{N} \sum_{m=1}^{M} \left| \int_{t_{m-1}}^{t_m} Res_{\,n}(\mathbf{U}) \varphi_n\,\mathrm{d}t \right|. \qquad (3.21)
\end{aligned}$$

The quantity (3.21) is now used to construct a reduced (in the sense of simplified) model which is defined as follows:

Find $\mathbf{U}$ (satisfying the conditions above including $\mathbf{U}(0) = \mathbf{u}_0$) such that

$$\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} + \widehat{\mathbf{a}}(t, \mathbf{U}) = \mathbf{0} \text{ for } 0 < t \leq T, \qquad (3.22)$$

$$\text{where } \hat{a}_n(t, \mathbf{U}) = \begin{cases} a_n(t, \mathbf{U}) & \text{if } t \in S_n, \\ 0 & \text{otherwise.} \end{cases}$$

For $n = 1, \ldots, N$ the $S_n$ are unions of subsets $I_m = (t_{m-1}, t_m]$ of the computational domain which are as yet to be defined. Initially all $S_n$ are empty, and it is assumed that the solution $\mathbf{u}$ to the primal problem (3.12) is known. Now set a tolerance `TOL` for the bound on $|J(\mathbf{u}) - J(\mathbf{U})|$ and iterate the following algorithm:

1. Solve the reduced model (3.22) to get $\mathbf{U}$.
2. Solve the dual problem (3.13) or (3.20) to get $\boldsymbol{\varphi}$.
3. Compute $|J(\mathbf{u}) - J(\mathbf{U})|$. Stop if $|J(\mathbf{u}) - J(\mathbf{U})| \leq$ `TOL`.
4. Find the $l$ largest contributions to the error bound (3.21) over all components $n$ and all elements $m$. Include these $I_m$ in the appropriate $S_n$.
5. Repeat from 1.

Let us explain the fourth step in a bit more detail. In 4. we determine a set of tuples $L = \{(n_1, m_1), \ldots, (n_l, m_l)\}$ such that if $(\nu, \mu) \in L$ then

$$\left| \int_{t_{\mu-1}}^{t_\mu} Res_\nu(\mathbf{U}) \varphi_\nu \, \mathrm{d}t \right| > \left| \int_{t_{m-1}}^{t_m} Res_n(\mathbf{U}) \varphi_n \, \mathrm{d}t \right|$$

for all $(n, m) \in \{1, \ldots, N\} \times \{1, \ldots, M\} \setminus L$. In this case we would include $I_\mu$ in the set $S_\nu$, so $S_\nu$ becomes $S_\nu \cup I_\mu$. Once the iteration stops, not only will we have generated a reduced model, we will at the same time have computed the reduced model solution and a bound on $|J(\mathbf{u}) - J(\mathbf{U})|$. It is also the case that the iteration will always terminate, even if `TOL` is set to a (too) small value. The worst that can happen then is that eventually all $S_n$ include all subintervals $I_m$, meaning that they correspond to the computational domain. In that case $\widehat{\mathbf{a}}(t, \mathbf{U}) = \mathbf{a}(t, \mathbf{U})$ whence $\mathbf{U} = \mathbf{u}$. This simply means that the generated model is actually the primal problem (the full model), so no model reduction has been achieved.

In any other case however the model reduction algorithm generates a simplified model in

the sense that only a subset of the full set of equations constituting the mathematical model has to be solved on certain subintervals of the computational domain. The example that Whiteley took in [4] to illustrate the model reduction algorithm at work is a cardiac electro-physiology model consisting of more than twenty ODEs. Researchers are often interested in the upstroke of the trans-membrane potential, thus a reduced model can be generated based on a linear functional which only takes into account that quantity of interest. The result of the model reduction algorithm is that the generated reduced model includes only two more quantities besides the one of interest (two of the fast sodium current gating variables). This yields a great simplification of course in as far as only three equations are needed to compute the upstroke in the trans-membrane potential sufficiently accurately. If on the other hand one were interested in the calcium transient (as calcium is a key quantity for generating forces in fibres) then the linear functional has to be defined accordingly and as a result a reduced model will be generated that includes a different subset of the original equations.

This method of model reduction has so far been applied to initial and boundary value parabolic ODEs and time-dependent boundary value PDEs in one dimension. Apart from simple reaction-diffusion problems the method has also been applied to a cell-level cardiac electro-physiology model (as mentioned above, see [4]) and to Fisher's equation (see [42]).

### 3.2.2 Model Reduction Combined with Mesh Refinement

The work by Whiteley served as a basis for a first extension of the method as part of a short project in the first year of the DTC (the university of Oxford's doctoral training centre). As was already mentioned, the technique is borrowed from FE methods where it is used to adaptively refine the mesh in order to increase the accuracy of the numerical approximation to the solution. It seemed therefore appropriate to combine model reduction with adaptive mesh refinement instead of relying on knowledge of the true solution (or rather assuming that such is known). Thus, instead of using the bound (3.21) on $|J(\mathbf{u}) - J(\mathbf{U})|$ for the model reduction algorithm, we worked only with FE

approximations and wrote

$$|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})| = |J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}}) + J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|,$$
$$\leq |J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| + |J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|.$$

Here $\mathbf{u}^{\mathsf{h}}$ and $\mathbf{U}^{\mathsf{h}}$ are FE approximations to the solutions $\mathbf{u}$ and $\mathbf{U}$ of the full and reduced models respectively. Then we derived bounds on each of $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$ and $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ similar to (3.21) and used the former to adaptively refine the mesh and the latter to generate a reduced model. One advantage of not making use of $\mathbf{u}$ but rather to work with an approximation $\mathbf{u}^{\mathsf{h}}$ is that we do not necessarily need a very fine mesh to obtain the same computational accuracy and thus can speed up computation. Further to that we dropped the restrictions imposed on the reduced model solution $\mathbf{U}$ or approximation $\mathbf{U}^{\mathsf{h}}$ thereof. This means in particular that neither do $\mathbf{u}$, $\mathbf{U}$ have to be continuous on the computational domain, nor do they have to satisfy the initial condition of the primal problem. This means that the approximations $\mathbf{u}^{\mathsf{h}}$, $\mathbf{U}^{\mathsf{h}}$ can be computed by the dG($q$) method which has (in most cases) better stability properties than cG($q$) as was briefly discussed in Chapter 2 at the end of Section 2.3.1. We will briefly outline the model reduction technique combined with mesh refinement here, as some of the analysis will be re-used in subsequent chapters.

We will first derive a bound on $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ as this is novel in as far that both $\mathbf{u}^{\mathsf{h}}$ and $\mathbf{U}^{\mathsf{h}}$ are discontinuous over the element boundaries. The bound on $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$, where $\mathbf{u}$ satisfies the continuity conditions set out before, then follows easily. Say that we choose the dG($q$) method, on a mesh $\mathcal{I}_h = \{I_m\}_{m=1}^{M}$ with elements $I_m = (t_{m-1}, t_m]$, to compute the FE approximations $\mathbf{u}^{\mathsf{h}}$ and $\mathbf{U}^{\mathsf{h}}$ to the solutions of the primal and reduced problems respectively. In order to compute the error induced in the linear functional when $\mathbf{u}$ is replaced by $\mathbf{u}^{\mathsf{h}}$ and $\mathbf{U}$ is replaced by $\mathbf{U}^{\mathsf{h}}$ we use an appropriate dual problem that coincides with (3.13) or (3.20), where only the definition of the adjoint matrix $\mathbf{A}^*$ changes. For model reduction the entries of $\mathbf{A}^*$, the functions $A^*_{ij}(t)$, should be

$$A^*_{ij}(t) = \int_0^1 \frac{\partial a_j}{\partial u_i}(s\mathbf{u}^{\mathsf{h}}(t) + (1-s)\mathbf{U}^{\mathsf{h}}(t))\,\mathrm{d}s \quad \text{for } i,j = 1,\ldots,N. \tag{3.23}$$

As the only change is the replacement of $\mathbf{u}$ by $\mathbf{u}^\mathsf{h}$ and $\mathbf{U}$ by $\mathbf{U}^\mathsf{h}$ it follows that the derivation of a bound for $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})|$ is analogous to the derivation of a bound for $|J(\mathbf{u}) - J(\mathbf{U})|$ above, up to the point where continuity is required. That is, for the dual problem (3.13) with adjoint matrix defined as in (3.23) instead of (3.14), we can easily derive the following:

$$\begin{aligned} J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h}) &= \big\langle \mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}, \mathbf{g} \big\rangle, \\ &= \big\langle \mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}, -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^* \boldsymbol{\varphi} \big\rangle, \\ &= \sum_{m=1}^{M} \big[ -\boldsymbol{\varphi} \cdot (\mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}) \big]_{t_{m-1}}^{t_m} \\ &\quad + \left\langle \left( \frac{\mathrm{d}\mathbf{u}^\mathsf{h}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^\mathsf{h}) \right), \boldsymbol{\varphi} \right\rangle - \left\langle \left( \frac{\mathrm{d}\mathbf{U}^\mathsf{h}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{U}^\mathsf{h}) \right), \boldsymbol{\varphi} \right\rangle, \end{aligned} \tag{3.24}$$

whereas for (3.20) we similarly get:

$$\begin{aligned} 0 &= \big\langle \mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}, \mathbf{0} \big\rangle, \\ &= \big\langle \mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}, -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^* \boldsymbol{\varphi} \big\rangle, \\ &= \sum_{m=1}^{M} \big[ -\boldsymbol{\varphi} \cdot (\mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}) \big]_{t_{m-1}}^{t_m} \\ &\quad + \left\langle \left( \frac{\mathrm{d}\mathbf{u}^\mathsf{h}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^\mathsf{h}) \right), \boldsymbol{\varphi} \right\rangle - \left\langle \left( \frac{\mathrm{d}\mathbf{U}^\mathsf{h}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{U}^\mathsf{h}) \right), \boldsymbol{\varphi} \right\rangle, \end{aligned} \tag{3.25}$$

The only difference here to Section 3.2.1 is the extra term arising from integration by parts. Because $\mathbf{u}^\mathsf{h}$ and $\mathbf{U}^\mathsf{h}$ are discontinuous over the element boundaries, this extra term is not equal to $-\boldsymbol{\varphi}(T) \cdot (\mathbf{u}^\mathsf{h}(T) - \mathbf{U}^\mathsf{h}(T))$ as in the continuous case, but we also have to include all 'jumps'. Note that as $\boldsymbol{\varphi}$ is continuous, we can write this as

$$\begin{aligned} \sum_{m=1}^{M} \big[ -\boldsymbol{\varphi} \cdot (\mathbf{u}^\mathsf{h} - \mathbf{U}^\mathsf{h}) \big]_{t_{m-1}}^{t_m} &= -\boldsymbol{\varphi}(t_M) \cdot (\mathbf{u}^\mathsf{h}(t_M^-) - \mathbf{U}^\mathsf{h}(t_M^-)) \\ &\quad + \boldsymbol{\varphi}(t_{M-1}) \cdot (\mathbf{u}^\mathsf{h}(t_{M-1}^+) - \mathbf{U}^\mathsf{h}(t_{M-1}^+)) \\ &\quad - \boldsymbol{\varphi}(t_{M-1}) \cdot (\mathbf{u}^\mathsf{h}(t_{M-1}^-) - \mathbf{U}^\mathsf{h}(t_{M-1}^-)) \\ &\quad + \boldsymbol{\varphi}(t_{M-2}) \cdot (\mathbf{u}^\mathsf{h}(t_{M-2}^+) - \mathbf{U}^\mathsf{h}(t_{M-2}^+)) \\ &\quad - \boldsymbol{\varphi}(t_{M-2}) \cdot (\mathbf{u}^\mathsf{h}(t_{M-2}^-) - \mathbf{U}^\mathsf{h}(t_{M-2}^-)) \end{aligned}$$

$$
\begin{aligned}
&\vdots \\
&+ \boldsymbol{\varphi}(t_1) \cdot (\mathbf{u}^{\mathsf{h}}(t_1^+) - \mathbf{U}^{\mathsf{h}}(t_1^+)) \\
&- \boldsymbol{\varphi}(t_1) \cdot (\mathbf{u}^{\mathsf{h}}(t_1^-) - \mathbf{U}^{\mathsf{h}}(t_1^-)) \\
&+ \boldsymbol{\varphi}(t_0) \cdot (\mathbf{u}^{\mathsf{h}}(t_0^+) - \mathbf{U}^{\mathsf{h}}(t_0^+)) \\
&- \boldsymbol{\varphi}(t_0) \cdot (\mathbf{u}^{\mathsf{h}}(t_0^-) - \mathbf{U}^{\mathsf{h}}(t_0^-)),
\end{aligned}
$$

where we added the last line which is zero since $\mathbf{u}^{\mathsf{h}}(t_0^-) = \mathbf{u}_0 = \mathbf{U}^{\mathsf{h}}(t_0^-)$, so

$$
\begin{aligned}
\sum_{m=1}^{M} \left[-\boldsymbol{\varphi} \cdot (\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}})\right]_{t_{m-1}}^{t_m} &= -\boldsymbol{\varphi}(T) \cdot (\mathbf{u}^{\mathsf{h}}(T^-) - \mathbf{U}^{\mathsf{h}}(T^-)) \\
&\quad + \sum_{m=1}^{M} [\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}).
\end{aligned}
$$

Now note that for dual problem (3.13) we have $\boldsymbol{\varphi}(T) = \mathbf{0}$, whence

$$
\sum_{m=1}^{M} \left[-\boldsymbol{\varphi} \cdot (\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}})\right]_{t_{m-1}}^{t_m} = \sum_{m=1}^{M} [\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}), \tag{3.26}
$$

whereas per definition of $\boldsymbol{\varphi}(T)$ in dual problem (3.20) we get

$$
\begin{aligned}
\sum_{m=1}^{M} \left[-\boldsymbol{\varphi} \cdot (\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}})\right]_{t_{m-1}}^{t_m} &= -(u_n(T) - u_n^h(T^-)) + \sum_{m=1}^{M} [\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}), \\
&= -(J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})) + \sum_{m=1}^{M} [\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}).
\end{aligned} \tag{3.27}
$$

Substitution of (3.26) into (3.24) or (3.27) into (3.25) yields in both cases

$$
\begin{aligned}
J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}}) &= \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}) \right) \cdot \boldsymbol{\varphi} \, \mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}) \right) \\
&\quad - \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{U}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{U}^{\mathsf{h}}) \right) \cdot \boldsymbol{\varphi} \, \mathrm{d}t + [\mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}) \right), \\
&= \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \mathrm{Res}\,(\mathbf{u}^{\mathsf{h}}) \cdot \boldsymbol{\varphi} \, \mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}) \right) \\
&\quad - \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \mathrm{Res}\,(\mathbf{U}^{\mathsf{h}}) \cdot \boldsymbol{\varphi} \, \mathrm{d}t + [\mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}) \right).
\end{aligned} \tag{3.28}
$$

Equality (3.28) of course only holds if $\boldsymbol{\varphi}$ is the true solution of the dual problem associated with the primal problem (3.12). If we use a FE approximation to the solution of the dual problem then we get an approximation to the functional error

$$|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| \approx \left| \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left(\mathrm{Res}\,(\mathbf{u}^{\mathsf{h}}) - \mathrm{Res}\,(\mathbf{U}^{\mathsf{h}})\right) \cdot \boldsymbol{\varphi}^{\mathsf{h}} \,\mathrm{d}t \right.\right. \\ \left.\left. + [\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1}) \right) \right|, \tag{3.29}$$

$$\lessapprox \sum_{n=1}^{N} \sum_{m=1}^{M} \left| \int_{t_{m-1}}^{t_m} \left(Res\,_n(\mathbf{u}^{\mathsf{h}}) - Res\,_n(\mathbf{U}^{\mathsf{h}})\right) \varphi^h \,\mathrm{d}t \right. \\ \left. + [u_n^h - U_n^h]_{m-1}\, \varphi^h(t_{m-1}) \right|. \tag{3.30}$$

Note that since the derivation of (3.28) relied on continuity of the dual solution $\boldsymbol{\varphi}$ we also require its FE approximation $\boldsymbol{\varphi}^{\mathsf{h}}$ to be continuous. The dual solution approximation will therefore be computed with the continuous Galerkin method.

Now to compute the error induced into the linear functional when $\mathbf{u}$ is replaced by $\mathbf{u}^{\mathsf{h}}$ we only have to change the entries of the adjoint matrix $\mathbf{A}^*$ to

$$A^*_{ij}(t) = \int_0^1 \frac{\partial a_j}{\partial u_i}(s\mathbf{u}(t) + (1-s)\mathbf{u}^{\mathsf{h}}(t)) \,\mathrm{d}s \quad \text{for } i,j = 1,\ldots,N. \tag{3.31}$$

However, as $\mathbf{u}$ is unknown we approximate these by evaluating the Jacobian at $\mathbf{u}^{\mathsf{h}}$,

$$A^*_{ij}(t) \approx \left.\frac{\partial a_j}{\partial u_i}\right|_{\mathbf{u}(t)=\mathbf{u}^{\mathsf{h}}(t)} \quad \text{for } i,j = 1,\ldots,N. \tag{3.32}$$

The analysis above leading to (3.29) and (3.30) is followed accordingly where $\mathbf{u}$ takes the place of $\mathbf{u}^{\mathsf{h}}$ and $\mathbf{u}^{\mathsf{h}}$ that of $\mathbf{U}^{\mathsf{h}}$. As $\mathrm{Res}\,(\mathbf{u}) = \mathbf{0}$ because $\mathbf{u}$ solves the primal problem, and also $[\mathbf{u}]_{m-1} = \mathbf{0}$ because $\mathbf{u}$ is continuous, the equation corresponding to (3.28) simplifies to

$$J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}}) \approx -\sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \mathrm{Res}\,(\mathbf{u}^{\mathsf{h}}) \cdot \boldsymbol{\varphi} \,\mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}(t_{m-1}) \right).$$

This is not an equality because the adjoint matrix is an approximation to the 'true' ad-

joint (3.31). With $\boldsymbol{\varphi}^{\mathsf{h}}$ denoting a continuous FE approximation to either dual problem (3.13) or (3.20) with adjoint matrix entries (3.32) we can write

$$|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| \approx \left| \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \mathrm{Res}\,(\mathbf{u}^{\mathsf{h}}) \cdot \boldsymbol{\varphi}^{\mathsf{h}} \, \mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1}) \right) \right|, \tag{3.33}$$

$$\lessapprox \sum_{n=1}^{N} \sum_{m=1}^{M} \left| \int_{t_{m-1}}^{t_m} Res_{\,n}(\mathbf{u}^{\mathsf{h}}) \varphi^h \, \mathrm{d}t + [u_n^h]_{m-1} \, \varphi^h(t_{m-1}) \right|. \tag{3.34}$$

Thus we derived a bound on $|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})|$ which is the sum of the right-hand-sides of (3.34) and (3.30) which differs of course from the bound (3.21) in Section 3.2.1. We are not assuming that the true solution is known (or computed on a very fine mesh which would warrant that it be called the 'solution'), and are working with dG($q$) approximations. The model reduction algorithm introduced in the previous section did not change to begin with, only the way the 'solutions' are computed, and therefore also the computation of the error induced in the linear functional which is used as a stopping criterion for the algorithm. These changes served as a starting point to develop the model reduction technique further, as presented in the next chapter.

# CHAPTER 4

# The Model Reduction Method

We will describe our extension of the model reduction method in this chapter for systems of initial value ordinary differential equations. By way of an introduction let us recapitulate in general terms.

1. Given is a primal problem (the full model) and its solution or FE approximation thereof.

2. Model reduction generates a reduced model based on a user defined linear functional of the solution to the primal problem.

3. The (initial) reduced problem solution or FE approximation thereof together with the definition of the linear functional determine an appropriate dual problem.

4. The solution or FE approximation thereof to this dual problem is used in each step of the model reduction algorithm to drive model reduction further.

This does not look at all different to the original model reduction technique introduced in Section 3.2.1 of the preceding chapter, other than working with discontinuous Galerkin approximations instead of continuous solutions.

The big change lies with the form that the reduced model can take: where originally the reduced model always consisted of a system of differential equations (DEs), we now also allow differential-algebraic equations (DAEs)[1]. To clarify, say that we denote the reduced model operator by $\widehat{\boldsymbol{\mathcal{D}}}$ and its $n^{th}$ component by $\widehat{\mathcal{D}}_n$ (where the subscript $n$ means $\widehat{\mathcal{D}}_n\mathbf{U} = (\widehat{\boldsymbol{\mathcal{D}}}\mathbf{U})_n$ as was introduced earlier). Then the original model reduction

[1]In this context 'always' means that on each and every sub-domain of the computational domain all equations that constitute the system are differential equations as opposed to one or more being algebraic equations.

method allowed $\widehat{\boldsymbol{\mathcal{D}}}$ to take the form

$$\widehat{\mathcal{D}}_n\mathbf{U}(t) = \begin{cases} \dfrac{\mathrm{d}U_n}{\mathrm{d}t} & \text{if } t \notin S_n, \\ \dfrac{\mathrm{d}U_n}{\mathrm{d}t} + a_n(t,\mathbf{U}) & \text{if } t \in S_n, \end{cases} \tag{4.1}$$

Although this generates a reduced model it may however not be suitable if the original system of ODEs models processes that involve very slow or rapid changes of the solution components. One such example is an enzyme kinetics model where the concentrations of the reactants vary significantly over small and large timescales. This is reflected in their rates of change with respect to time which can be captured by analytical methods, such as have been discussed in Section 3.1.1. If we only allow model reduction in the original form however, then a reduced model resulting from the algorithm cannot reflect that. We therefore introduce a further option that the reduced model can take which is the possibility to exclude the derivative term from an equation in the system. Our thus extended model reduction method allows the reduced model operator $\widehat{\boldsymbol{\mathcal{D}}}$ to take this form:

$$\widehat{\mathcal{D}}_n\mathbf{U}(t) = \begin{cases} \dfrac{\mathrm{d}U_n}{\mathrm{d}t} & \text{if } t \in R_n \text{ and } t \notin S_n, \\ a_n(t,\mathbf{U}) & \text{if } t \notin R_n \text{ and } t \in S_n, \\ \dfrac{\mathrm{d}U_n}{\mathrm{d}t} + a_n(t,\mathbf{U}) & \text{if } t \in R_n \text{ and } t \in S_n. \end{cases} \tag{4.2}$$

Thus in addition to the $N$ sets $S_n$ which determine whether the non-derivative term $a_n(t,\mathbf{U})$ is included in the $n^{th}$ equation, we need another $N$ sets $R_n$ to determine whether the derivative $\mathrm{d}U_n/\mathrm{d}t$ is included or not. The 'loss of a solution's history' however is not so straightforward as this can result in inconsistent initial conditions or discontinuities in the solution when the rate of change term is neglected. The algorithm has to take this into account as well as keeping track of the terms that are included or excluded on each element when generating the reduced model. These extra features require novel analysis.

This chapter is dedicated to the extended model reductiontechnique and to tackling the

issues raised above. It is divided into two parts, where the first part (Section 4.1), the main section of this chapter, contains our (extended) model reduction algorithm and subsections dealing with the prerequisites that are necessary for the application of said algorithm. The second part (Section 4.2) is used to validate the numerical methods that are employed with two model examples. The first one has analytic solutions to primal, dual, and reduced models which makes it possible to compare the estimates for the *a posteriori* errors with the exact values. The second example is of the form usually obtained through asymptotic model reduction as described shortly in Section 3.1.1. We chose this model example not only for validation purposes but also to show that our extended method is especially suitable to reduce models of this form, which could not be achieved by the original model reduction method.

## 4.1 Model Reduction Technique

We are once again looking at an initial value system of ODEs of the form

$$\begin{aligned}&\text{Find } \mathbf{u}(t) = (u_1(t), \ldots, u_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T) \text{ satisfying } \mathbf{u}(0) = \mathbf{u}_0 \text{ such that} \\ &\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t,\mathbf{u}) = \mathbf{f}(t) \quad \text{for} \quad 0 < t \leq T. \end{aligned} \tag{4.3}$$

Note that the right-hand-side is not the zero vector as in example (3.12) of Whiteley [4] which was discussed in Section 3.2.1. We included a possibly non-zero right-hand-side as any general system of ODEs, $\mathrm{d}\mathbf{u}/\mathrm{d}t + \mathbf{b}(t,\mathbf{u}) = \mathbf{0}$, can be written in this form by splitting the vector-valued function $\mathbf{b}(t,\mathbf{u})$ into a sum of two functions $\mathbf{a}(t,\mathbf{u}) - \mathbf{f}(t)$. This is akin to the convention for linear ODEs, where $\mathrm{d}\mathbf{u}/\mathrm{d}t + \mathbf{A}(t)\mathbf{u} = \mathbf{0}$ is the homogeneous system that corresponds to the inhomogeneous initial value system of first order ODEs $\mathrm{d}\mathbf{u}/\mathrm{d}t + \mathbf{A}(t)\mathbf{u} = \mathbf{f}(t)$. Splitting the function $\mathbf{b}(t,\mathbf{u})$ into solution dependent and independent parts did actually arise from a more general form for the reduced model operator[2]. Suffice to note that we will adopt the 'inhomogeneous'

[2]Whiteley [4] employed the use of $N$ sets $S_n$ that determine which equations $\mathrm{d}u_n/\mathrm{d}t + \widehat{b}_n(t,\mathbf{u}) = 0$ have $\widehat{b}_n(t,\mathbf{u}) = b_n(t,\mathbf{u})$ $(t \in S_n)$ or $\widehat{b}_n(t,\mathbf{u}) = 0$ $(t \notin S_n)$. Thus one set is needed to determine the inclusion of one term of an equation and therefore $k$ sets to include or exclude $k$ terms of an equation. If we were to write equation $\mathrm{d}\mathbf{u}/\mathrm{d}t + \mathbf{b}(t,\mathbf{u}) = \mathbf{0}$ as $\mathrm{d}\mathbf{u}/\mathrm{d}t + \mathbf{a}^{(\mathbf{1})}(t,\mathbf{u}) + \mathbf{a}^{(\mathbf{2})}(t,\mathbf{u}) = \mathbf{0}$ then we would

notation here and in the remainder of this chapter and the following chapters as this follows through with the notation already adopted in Chapter 2 where the FE method was introduced.

We assume that a FE approximation $\mathbf{u}^{\mathsf{h}}$ of the solution $\mathbf{u}$ to the primal problem (4.3), which we also call the full model, has been computed on a mesh $\mathcal{I}_h = \{I_m\}_{m=1}^{M}$ with elements $I_m = (t_{m-1}\,, t_m]$ that partition the computational domain. It is irrelevant at this stage whether the mesh has been 'chosen' as such or is the result of an adaptive mesh refinement algorithm (which will be addressed in Chapter 5) . Although the mesh does have an effect on the induced error $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$ of course, we are at this stage only concerned with model reduction. We do not use a sufficiently fine mesh in this chapter that we could assume $\mathbf{u} \approx \mathbf{u}^{\mathsf{h}}$ as is the case for the original model reduction method that was discussed in Section 3.2. There the full model was reduced using the error induced into the linear functional when the solution $\mathbf{u}$ is replaced by the reduced model solution $\mathbf{U}$. Our model reduction algorithm however works with the induced error $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ precisely because we do not want to have to rely on 'true' solutions or approximations on very fine meshes. An estimate for this error will be derived in the following section.

### 4.1.1 Derivation of the Induced Error Estimate

The dual problem associated with the primal problem (4.3) is dependent on the definition of the linear functional

$$J(\mathbf{u}) = \big\langle \mathbf{u}\,, \mathbf{g} \big\rangle = \int_0^T \sum_{n=1}^{N} u_n(t) g_n(t)\, \mathrm{d}t,$$

and, as before, we distinguish between the two cases:

need $2N$ sets $S_n^{(1)}$, $S_n^{(2)}$ to determine the inclusion of $a_n^{(1)}(t, \mathbf{u})$ or $a_n^{(2)}(t, \mathbf{u})$ in equation $n$. Furthermore, if we also allow the exclusion of $\mathrm{d}U_n/\mathrm{d}t$ then $N$ more sets $R_n$ are needed to keep track of that. This makes a total of $3N$ sets and up to 3! cases that the operator $\widehat{\mathcal{D}}_n$ could take. The introduction of the sets $R_n$ in addition to the $S_n$ in the new definition (4.2) already gives rise to issues pertaining to 'loss of a solution's history' and possible discontinuities of the reduced model solution. We therefore decided in favour of the 'minimal' split of $\mathbf{b}(t, \mathbf{u})$ into a term $\mathbf{a}(t, \mathbf{u})$ that is dependent on the solution $\mathbf{u}$ and an independent term $\mathbf{f}(t)$.

case (1) $g_n(T)$ is finite for all $n = 1, \ldots, N$; and

case (2) there exists $n \in \{1, \ldots, N\}$ such that $g_n(t) = \delta(t - T)$ and $g_k(t) = 0$ for all other $k = \{1, \ldots, N\} \setminus \{n\}$.

The dual problem can now be defined as

Find $\boldsymbol{\varphi}(t) = (\varphi_1(t), \ldots, \varphi_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0, T)$ where $\boldsymbol{\varphi}$ satisfies

case (1) $\boldsymbol{\varphi}(T) = \mathbf{0}$ such that

$$-\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} = \mathbf{g} \quad \text{for} \quad T > t \geq 0, \tag{4.4}$$

case (2) $\varphi_n(T) = 1$ and $\varphi_k(T) = 0$ for all $k \in \{1, \ldots, N\} \setminus \{n\}$ such that

$$-\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} = \mathbf{0} \quad \text{for} \quad T > t \geq 0, \tag{4.5}$$

where in either case the entries of the adjoint matrix $\mathbf{A}^*$ are defined as

$$A^*_{ij}(t) = \int_0^1 \frac{\partial a_j}{\partial u_i}(s\mathbf{u}^{\mathsf{h}}(t) + (1-s)\mathbf{U}^{\mathsf{h}}(t))\,\mathrm{d}s \quad \text{for } i, j = 1, \ldots, N. \tag{4.6}$$

The vector-valued function $\mathbf{U}^{\mathsf{h}}$ appearing in the definition (4.6) is a FE approximation to the solution of a reduced model, which we will specify shortly, computed on the same mesh as $\mathbf{u}^{\mathsf{h}}$, the approximation to the solution of the full model[3]. Suppose now that $\boldsymbol{\varphi}^{\mathsf{h}}$ is a continuous Galerkin FE approximation to (either of) the dual problem(s). Then we can readily write down a bound on the functional error $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$, referring back to Section 3.2.2 and the derivations leading to (3.29) and (3.30) in particular:

$$|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| = \int_0^T (\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}) \cdot \mathbf{g}\,\mathrm{d}t, \tag{4.7}$$

$$\begin{aligned} &\approx \left| \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}) \right) \cdot \boldsymbol{\varphi}^{\mathsf{h}}\,\mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1}) \right) \right. \\ &\quad \left. - \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{U}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{U}^{\mathsf{h}}) \right) \cdot \boldsymbol{\varphi}^{\mathsf{h}}\,\mathrm{d}t + [\mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1}) \right) \right|, \\ &\lessapprox \sum_{n=1}^{N} \sum_{m=1}^{M} \left| \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}u_n^h}{\mathrm{d}t} + a_n(t, \mathbf{u}^{\mathsf{h}}) \right) \varphi_n^h\,\mathrm{d}t + [u_n^h]_{m-1}\varphi_n^h(t_{m-1}) \right) \right. \\ &\quad \left. - \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}U_n^h}{\mathrm{d}t} + a_n(t, \mathbf{U}^{\mathsf{h}}) \right) \varphi_n^h\,\mathrm{d}t + [U_n^h]_{m-1}\varphi_n^h(t_{m-1}) \right) \right|. \end{aligned} \tag{4.8}$$

[3]If the true solution were known, then we would replace $\mathbf{U}^{\mathsf{h}}$ by $\mathbf{U}$ of course. This is hardly ever the case however, with few exceptions like the examples at the end of this chapter.

We see that the quantity $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ is small whenever $\boldsymbol{\varphi}^{\mathsf{h}}$ is 'nearly orthogonal' to the residuals. On the other hand it can be just as small when $\mathbf{U}^{\mathsf{h}}$ is 'very close' to $\mathbf{u}^{\mathsf{h}}$ and does vanish if they coincide. This is only the case however if no model reduction took place, meaning that the 'reduced' model is the full model.

### 4.1.2 The Reduced Model

Finally we can turn our attention to the reduced model, for which exclusion of the derivative term is now also allowed. For $n = 1, \dots, N$ we define subsets $R_n$ and $S_n$ of $I = \bigcup_{m=1}^{M} I_m$ that are unions of elements $I_m$ such that for any given $n$ their union covers the whole interval $(0\,, T]$. The sets $R_n$ determine whether the time derivative is included in equation $n$ and the sets $S_n$ whether $a_n(t, \cdot)$ is included, as the original model reduction algorithm has it. We also define an operator $\widehat{\boldsymbol{\mathcal{D}}}\colon \boldsymbol{\mathcal{H}}^1(I) \to \boldsymbol{\mathcal{L}}_2(I)$ whose $N$ components are the operators $\widehat{\mathcal{D}}_n\colon \mathcal{H}^1(I) \to \mathcal{L}_2(I)$, and write the reduced model as:

Find $\mathbf{U}(t) = (U_1(t), \dots, U_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{H}}^1(I)$ such that for all $n = 1, \dots, N$

$$\widehat{\mathcal{D}}_n \mathbf{U}(t) = f_n(t) \quad \text{for } 0 < t \leq T \quad \text{and } U_n(0) = u_n(0),$$

$$\text{with} \quad \widehat{\mathcal{D}}_n \mathbf{U}(t) = \begin{cases} \dfrac{\mathrm{d}U_n}{\mathrm{d}t} & \text{if } t \in R_n \text{ and } t \notin S_n, \\ a_n(t, \mathbf{U}) & \text{if } t \notin R_n \text{ and } t \in S_n, \\ \dfrac{\mathrm{d}U_n}{\mathrm{d}t} + a_n(t, \mathbf{U}) & \text{if } t \in R_n \text{ and } t \in S_n, \end{cases} \tag{4.9}$$

where $\mathbf{f}(t) = (f_1(t), \dots, f_N(t))^{\mathsf{T}}$ and $\mathbf{a}(t, \mathbf{u}) = (a_1(t, \mathbf{u}), \dots, a_N(t, \mathbf{u}))^{\mathsf{T}}$ are the vector-valued functions in $\boldsymbol{\mathcal{L}}_2(I)$ of the primal problem (4.3).

Note that by definition of the sets $S_n$ and $R_n$ the case '$t \notin R_n$ and $t \notin S_n$' does not exist. To make this a bit more transparent, we can re-write the reduced model (4.9) as a 'sequence' of local reduced problems:

For $m = 1, \dots, M$ find $\mathbf{U}(t)\big|_{I_m} \in \boldsymbol{\mathcal{H}}^1(I_m)$ with appropriate initial condition such that for $n = 1, \dots, N$

$$\widehat{\mathcal{D}}_n \mathbf{U}(t)\big|_{I_m} = f_n(t) \quad \text{for } t_m < t \leq t_{m+1}, \tag{4.10}$$

$$\text{where} \quad \widehat{\mathcal{D}}_n \mathbf{U}(t)\big|_{I_m} = \begin{cases} \dfrac{\mathrm{d}U_n\big|_{I_m}}{\mathrm{d}t} & \text{if } I_m \subset R_n \text{ and } I_m \not\subset S_n, \\ a_n(t, \mathbf{U}\big|_{I_m}) & \text{if } I_m \not\subset R_n \text{ and } I_m \subset S_n, \\ \dfrac{\mathrm{d}U_n\big|_{I_m}}{\mathrm{d}t} + a_n(t, \mathbf{U}\big|_{I_m}) & \text{if } I_m \subset R_n \text{ and } I_m \subset S_n. \end{cases}$$

Later, with the examples at the end of this chapter and also in the next, we will use a graphical representation of the reduced model as shown in Figure 4.1 and explained in the figure legend.

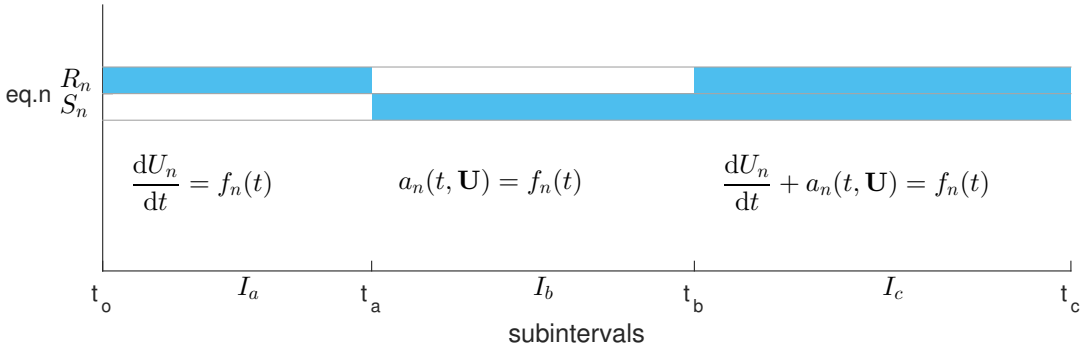



Figure 4.1: Depicted is equation $n$ of a reduced model where $R_n = I_a \cup I_c$ and $S_n = I_b \cup I_c$, where a coloured bar signifies that this subinterval is included in $R_n$ or $S_n$. Below the bar we wrote the 'local' reduced model $\widehat{\mathcal{D}}_n \mathbf{U}(t)\big|_{I_d}$ where $d \in \{a, b, c\}$.

With 'appropriate initial condition' referred to in the local reduced problem (4.10) we mean initial conditions appropriate to the type of (local) reduced model. Depending on whether the interval $I_m$ is included in the sets $R_n$ or $S_n$, equation $n$ is either an ordinary differential equation or an algebraic equation. A first order ODE requires an initial condition whereas an algebraic equation does not. Let us look at a local reduced problem of the form (4.10) for a particular $m$. If $I_m \subset R_n$ for all $n = 1, \ldots, N$ then the local reduced model is a system of $N$ ODEs (whether $I_m$ is included in any or all $S_n$) and the 'appropriate' initial condition is simply $\mathbf{U}(t_{m-1}) = \mathbf{U}(t_{m-1}^-)$.

Suppose now that the interval $I_m$ is included in at least one of the sets $R_n$ but not in all. Then we have a system of $N$ equations consisting of $N - P$ first order ODEs

coupled with $P$ algebraic equations[4]:

$$\begin{aligned} \frac{\mathrm{d}U_n}{\mathrm{d}t} + \widehat{a}_n(t, \mathbf{U}) &= f_n(t), \qquad n \in \{n_N, n_{N-1}, \ldots, n_{P+1}\}, \\ a_p(t, \mathbf{U}) &= f_p(t), \qquad p \in \{n_P, n_{P-1}, \ldots, n_1\}, \end{aligned} \tag{4.11}$$

where $P$ is an integer with $0 < P < N$. We used the notation $\widehat{a}_n(t, \mathbf{U})$ to signify that the term $a_n(t, \mathbf{U})$ might be included in equation $n$ depending on whether $I_m$ is a subset of $S_n$ or not, but this is not important in this discussion. For the $N - P$ ODEs we are given the initial condition

$$(U_{n_N}(t_{m-1}), \ldots, U_{n_{P+1}}(t_{m-1}))^{\mathsf{T}} = (U_{n_N}(t_{m-1}^-), \ldots, U_{n_{P+1}}(t_{m-1}^-))^{\mathsf{T}}. \tag{4.12}$$

If a solution to the system (4.11) subject to equation (4.12) exists then we say that the initial condition is consistent. If no solution can be found then we say that the initial condition is inconsistent or that no consistent initial condition exists. In this case one can try to increase the number of ODEs and decrease the number of algebraic equations in (4.11) accordingly, until a consistent initial condition can be found. We can justify this approach by noting that the primal problem is a system of ODEs and only the reduced model could become a system of ODEs coupled with algebraic equations, by neglecting the derivative terms of some of the equations of the primal problem. Thus if this should not work, meaning that no consistent initial condition can be found for that case, then we can simply include more of the derivative terms. Reducing the number $P$ of algebraic equations by one in each attempt to find a consistent initial condition can of course lead to $P = 0$ ultimately. This simply means that a reduced model cannot be a system of DAEs locally because that would be inconsistent and the (local) reduced model will still be a system of ODEs. This can nevertheless lead to a (globally) reduced or simplified model as it is possible that a model reduction can be achieved on other elements.

To increase the number of ODEs in order to get a system with consistent initial condi-

[4]If $P = N$ then we have a system of $N$ algebraic equations which may not have a solution. We will treat this case separately after the case where at least one of the $N$ equations is an ODE.

tion, proceed as follows. Assume that the set $\{n_P, n_{P-1}, \ldots, n_1\}$ is ordered according to the contribution to (4.8), meaning that for $p = P, P-1, \ldots, 2$

$$\begin{aligned}
&\left| \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}u^h_{n_p}}{\mathrm{d}t} + a_{n_p}(t, \mathbf{u}^\mathsf{h}) \right) \varphi^h_{n_p} \, \mathrm{d}t + [u^h_{n_p}]_{m-1} \varphi^h_{n_p}(t_{m-1}) \right. \\
&\qquad \left. - \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}U^h_{n_p}}{\mathrm{d}t} + a_{n_p}(t, \mathbf{U}^\mathsf{h}) \right) \varphi^h_{n_p} \, \mathrm{d}t + [U^h_{n_p}]_{m-1} \varphi^h_{n_p}(t_{m-1}) \right| \\
&\qquad\qquad \geq \left| \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}u^h_{n_{p-1}}}{\mathrm{d}t} + a_{n_{p-1}}(t, \mathbf{u}^\mathsf{h}) \right) \varphi^h_{n_{p-1}} \, \mathrm{d}t + [u^h_{n_{p-1}}]_{m-1} \varphi^h_{n_{p-1}}(t_{m-1}) \right. \\
&\qquad\qquad\qquad \left. - \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}U^h_{n_{p-1}}}{\mathrm{d}t} + a_{n_{p-1}}(t, \mathbf{U}^\mathsf{h}) \right) \varphi^h_{n_{p-1}} \, \mathrm{d}t + [U^h_{n_{p-1}}]_{m-1} \varphi^h_{n_{p-1}}(t_{m-1}) \right| .
\end{aligned}$$

Now include the set $I_m$ in $R_{n_P}$, and exclude $n_P$ from the set $\{n_P, n_{P-1}, \ldots, n_1\}$, which is the same as changing the value of $P$ to $P-1$. If the set is now empty, so $P = 0$, then we have a system of ODEs for which we have an 'appropriate' initial condition, and if $P \neq 0$ we again try to find consistent initial conditions. If that is possible it means that a solution to the system (4.11) subject to equation (4.12) exists. Otherwise we repeat this process until consistent initial conditions can be found. For a system of algebraic equations we proceed just the same as this corresponds to (4.11) with $P = N$.

This ensures that in any case consistent initial conditions can be found or in other words that a (local) reduced model solution or an approximation thereof exists[5]. Now that we have introduced all the equations, notations and bounds needed, we can at last formulate the model reduction algorithm in its general form.

### 4.1.3 Model Reduction Algorithm

Given is a discontinuous Galerkin approximation $\mathbf{u}^\mathsf{h}$ to the solution $\mathbf{u}$ of the primal problem (4.3) computed on a mesh $\mathcal{I}_h = \{I_m\}_{m=1}^M$, and also an initial reduced model. This means that the sets $R_n$ and $S_n$ have to be 'initialised' to include certain subsets $I_m$ of the computational domain $I = [0\,, T]$ as we require the union of $R_n$ and $S_n$ to

[5]Provided of course that the system of (nonlinear) initial value ODEs has a solution, even with changing initial conditions on each subinterval $I_m$, for which there is no guarantee. The models that we discuss in this dissertation do have a solution so this is not an issue. The subroutine to find consistent initial conditions is still needed to ensure the working of the model reduction algorithm.

cover $I$ for each $n \in \{1, \ldots, N\}$. We assume that we have an initial reduced model (initialisation is discussed in Section 4.1.4). We also need the FE approximation $\mathbf{U}^\mathsf{h}$ of the solution to that initial reduced model[6], computed with the same method on the same mesh $\mathcal{I}_h$ as was used to compute $\mathbf{u}^\mathsf{h}$.

We now set a value for the tolerance `TOL` and run the following algorithm, which generates a reduced model via the sets $R_n$ and $S_n$, and a FE approximation to the solution thereof.

1. Determine the entries of the adjoint matrix $\mathbf{A}^*$ with $\mathbf{u}^\mathsf{h}$ and $\mathbf{U}^\mathsf{h}$ as per definition (4.6) and compute the continuous Galerkin FE approximation $\boldsymbol{\varphi}^\mathsf{h}$ to the solution of dual problem (4.4) or (4.5).
2. Compute $\left|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})\right|$ with (4.7). Stop if this quantity does not exceed `TOL`.
3. Find the equation index $\nu$ and the element index $\mu$ of the largest contribution to the double sum (4.8).
4. Change the reduced model locally via a given 'rule'[7] that determines whether $I_\mu$ is included or excluded from $R_\nu$ or $S_\nu$.
5. Compute the discontinuous Galerkin approximation $\mathbf{U}^\mathsf{h}$ to the solution of the thus changed reduced model (4.9) by computing the approximations to the local reduced models (4.10).
6. Repeat from 1.

Once the algorithm stops we have $\left|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})\right| \leq \texttt{TOL}$ and, depending on the set tolerance as well as the choice of initialisation, a more or less simplified model. We would like to add here that in order to speed up computation, Step 3. could be altered such that more than just one equation on one element is changed:

[6]Note that this deviates from the original model reduction algorithm in Section 3.2.1 where initially all $S_n$ were empty. This would correspond to setting $R_n = I$ and $S_n = \varnothing$ for all $n$ initially and thereafter only the sets $S_n$ might change by inclusion of subsets $I_m$ whereas all the $R_n$ remain unchanged throughout. The choice of 'initialisation' does of course have a bearing on the outcome, meaning that there is a direct correlation between the initial and the generated reduced model, which we will discuss in a subsequent section.

[7]The rules that determine the local reduced model are dependent on the choice of initial reduced model; initialisations and accompanying rules are described in Section 4.1.4.

3*. Find the set $L = \{(n, m)\} \subset \{1, \ldots, N\} \times \{1, \ldots, M\}$ which contains all tuples $(\nu, \mu)$ of equation indices $\nu$ and element indices $\mu$ of the $l = |L|$ largest contributions to the double sum (4.8).

Although increasing $l$ (originally $l = 1$) does certainly speed up the algorithm there is nevertheless a drawback. For if $l$ is chosen too large then too many equations could be changed on too many elements and as a result we could get a reduced model that is less 'simple' as the one we could have got with a smaller $l$. This is but one of the sensitivities that the model reduction algorithm exhibits. The outcome of the model reduction algorithm is thus dependent on the choices that are made:

- Choice of initial reduced model (initialisation of sets $R_n$ and $S_n$).
- Choice of the rule for updating the sets $R_n$ and $S_n$ (inclusion or exclusion of $I_m$).
- Choice of linear functional of the solution.
- Choice of mesh for FE approximations.
- Choice of size of set $L$ of largest contributions.

This is not an exhaustive list but comprises the choices which we found to have a non-negligible impact. Change in any of these choices results in a different reduced model which means that it ultimately depends on what insight one wishes to gain through model reduction. Therefore we will always state the choices we make before applying the model reduction algorithm to a mathematical model. Before we end this chapter with two examples to validate the numerical methods employed, we specify different 'rules' which are used in Step 4. of the algorithm in the next section, and we will see that these are linked to the initialisation of the reduced model.

### 4.1.4 Initialisation and Rules for Model Reduction Algorithm

We now describe the 'initialisation' of the reduced problem and the 'rule' of including or excluding elements $I_\mu$ in the subsets $R_\nu$ and $S_\nu$, where $\nu$ is the equation number and $\mu$ the element index of the largest contribution to the double sum (4.8) that defines a

bound on the quantity $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$. We consider the following:

**(1)** initially $R_n = I$ and $S_n = \varnothing$ for all $n = 1, \ldots, N$, so $\dfrac{\mathrm{d}\mathbf{U}}{\mathrm{d}t} = \mathbf{f}$.

Rule: if $I_\mu \subset R_\nu$ and $I_\mu \not\subset S_\nu$ then exclude $I_\mu$ from $R_\nu$ and include $I_\mu$ in $S_\nu$;

if $I_\mu \not\subset R_\nu$ and $I_\mu \subset S_\nu$ then include $I_\mu$ in $R_\nu$

**(2)** initially $R_n = \varnothing$ and $S_n = I$ for all $n = 1, \ldots, N$, so $\mathbf{a}(t, \mathbf{U}) = \mathbf{f}$.

Rule: if $I_\mu \not\subset R_\nu$ and $I_\mu \subset S_\nu$ then include $I_\mu$ in $R_\nu$ and exclude $I_\mu$ from $S_\nu$,

if $I_\mu \subset R_\nu$ and $I_\mu \not\subset S_\nu$ then include $I_\mu$ in $S_\nu$.

Thus locally, for $t \in I_\mu = (t_{\mu-1}, t_\mu]$, the $\nu^{th}$ equation looks like one of the following options (i), (ii), or (iii):

$$\text{(i)} \qquad \frac{\mathrm{d}U_\nu}{\mathrm{d}t} = f_\nu \quad (\text{if } I_\mu \subset R_\nu \text{ and } I_\mu \not\subset S_\nu),$$

$$\text{(ii)} \qquad a_\nu(t, \mathbf{U}) = f_\nu \quad (\text{if } I_\mu \not\subset R_\nu \text{ and } I_\mu \subset S_\nu),$$

$$\text{(iii)} \qquad \frac{\mathrm{d}U_\nu}{\mathrm{d}t} + a_\nu(t, \mathbf{U}) = f_\nu \quad (\text{if } I_\mu \subset R_\nu \text{ and } I_\mu \subset S_\nu).$$

The rule for initialisation (1) changes equation $\nu$ on $I_\mu$ from (i) to (ii) if it is the first change and from (ii) to (iii) if it is the second change. Similarly, the rule for initialisation (2) changes equation $\nu$ on $I_\mu$ from (ii) to (i) if it is the first change and from (i) to (iii) if it is the second change. The main difference between the two rules is the initialisation, as we either start with a system of ODEs or a system of AEs. For the former we are given the initial value $\mathbf{U}^{\mathsf{h}}(t_0^-) = \mathbf{u}_0$ but for the latter it could be the case that there is no solution.

We cannot apply the strategy outlined in Section 4.1.2 as the initial reduced model solution $\mathbf{U}^{\mathsf{h}}$ is as yet unknown, but we can do something similar. For any $m \in \{1, \ldots, M\}$ where the algebraic system

$$\mathbf{a}(t, \mathbf{U}) = \mathbf{f}(t) \quad \text{with } t \in I_m \tag{4.13}$$

has no solution, we will succesively include derivative terms until the system, now consisting of ODEs and algebraic equations, has a solution. Without loss of generality

we assume that the equations in (4.13) are ordered according to the contribution to the double sum (3.34) bounding (approximately) the induced error $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$, meaning that for $n = 2, \ldots, N$

$$\left| \int_{t_{m-1}}^{t_m} Res_{\,n-1}(\mathbf{u}^{\mathsf{h}}) \varphi_{n-1}(t) \, \mathrm{d}t + [u_{n-1}^h]_{m-1} \varphi_{n-1}(t_{m-1}) \right| \geq \left| \int_{t_{m-1}}^{t_m} Res_{\,n}(\mathbf{u}^{\mathsf{h}}) \varphi_n(t) \, \mathrm{d}t + [u_n^h]_{m-1} \varphi_n(t_{m-1}) \right|,$$

where $\boldsymbol{\varphi}$ is the solution (or FE approximation thereof) of the dual problem associated with the primal problem used for mesh refinement[8]. Now we include $I_m$ in $R_1$ and exclude $I_m$ from $S_1$ which turns the first equation in (4.13) into an ODE:

$$\left.\begin{aligned} \frac{\mathrm{d}U_1}{\mathrm{d}t} &= f_1(t), \\ a_2(t, \mathbf{U}) &= f_2(t), \\ \vdots \qquad & \qquad \vdots \\ a_N(t, \mathbf{U}) &= f_N(t), \end{aligned}\right\} \text{ system of } N-1 \text{ AEs} \tag{4.14}$$

with initial condition $U_1(t_{m-1}) = U_1(t_{m-1}^-)$ for the ODE. If (4.14) has no solution subject to that initial condition then we include $I_m$ in $R_2$ and exclude $I_m$ from $S_2$, thus increasing the number of ODEs and decreasing the number of algebraic equations yet again by one. We continue to do this until consistent initial condition can be found. Note that in general this initialisation has to be done for each $m = 1, \ldots, M$. Only in the case where there is no time dependency do we get away with only one application of the process.

[8]This dual solution does not coincide with the solution of the dual problem used for model reduction since the adjoint matrices $\mathbf{A}^*$ differ, except if the primal problem is linear.

## 4.2 Application of Model Reduction Algorithm

We will apply the model reduction technique that has just been introduced to systems of ODEs that have the form of primal problem (4.3), which is:

$$\begin{aligned}&\text{Find } \mathbf{u}(t) = (u_1(t), \ldots, u_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T) \text{ satisfying } \mathbf{u}(0) = \mathbf{u}_0 \text{ such that}\\ &\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t,\mathbf{u}) = \mathbf{f}, \quad \text{where } \mathbf{f} \in \mathbb{R}^N, \quad \text{for} \quad 0 < t \leq T.\end{aligned}$$

We will look at a linear and a nonlinear system of ODEs for which the analytic solutions of the primal, dual, and reduced problems are known, for various linear functionals of the solution $J(\mathbf{u}) = \langle \mathbf{u}\,, \mathbf{g} \rangle$. We assume that a dG(0) approximation $\mathbf{u}^{\mathsf{h}}$ of the solution $\mathbf{u}$ of the full model (4.3) has been computed on a given mesh $\mathcal{I}_h = \{I_m\}_{m=1}^M$. Now we choose an initial reduced model and the rule that goes with it as to which terms of each equation we include or exclude in each iteration of the algorithm, as was introduced in Section 4.1.4 above. We can then apply the model reduction algorithm described in 4.1.3 and obtain a dG(0) approximation $\mathbf{U}^{\mathsf{h}}$ as a result.

We can compare the quantities $|J(\mathbf{u}) - J(\mathbf{U})|$ and $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ where the latter is computed in two different ways, directly and indirectly using the cG(1) approximation $\boldsymbol{\varphi}^{\mathsf{h}}$ of the dual problem solution $\boldsymbol{\varphi}$. That is, we compute and compare the following errors induced in the linear functional of the solution:

$$|J(\mathbf{u}) - J(\mathbf{U})| = \left| \int_0^T (\mathbf{u} - \mathbf{U}) \cdot \mathbf{g} \, \mathrm{d}t \right|, \tag{4.15}$$

$$|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| = \left| \sum_{m=1}^{M} \int_{t_{m-1}}^{t_m} \left(\mathbf{u}^{\mathsf{h}} - \mathbf{U}^{\mathsf{h}}\right) \cdot \mathbf{g} \, \mathrm{d}t \right|, \tag{4.16}$$

$$\begin{aligned}&\approx \left| \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{u}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{u}^{\mathsf{h}}) \right) \cdot \boldsymbol{\varphi}^{\mathsf{h}} \, \mathrm{d}t + [\mathbf{u}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1}) \right) \right.\\ &\quad \left. - \sum_{m=1}^{M} \left( \int_{t_{m-1}}^{t_m} \left( \frac{\mathrm{d}\mathbf{U}^{\mathsf{h}}}{\mathrm{d}t} + \mathbf{a}(t, \mathbf{U}^{\mathsf{h}}) \right) \cdot \boldsymbol{\varphi}^{\mathsf{h}} \, \mathrm{d}t + [\mathbf{U}^{\mathsf{h}}]_{m-1} \cdot \boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1}) \right) \right|,\end{aligned} \tag{4.17}$$

This is to demonstrate that the numerical methods do work, by comparing analytic

predictions with the numerical approximations. Once our method has been thus validated we will apply the technique to other examples in the next chapter, where we also combine model reduction with mesh refinement.

### 4.2.1 Linear System of ODEs

Our first example is the following linear initial value problem:

$$\begin{aligned}&\text{Find } \mathbf{u}(t) = (u_1(t), u_2(t), u_3(t))^{\mathsf{T}} \in \mathcal{C}^1(0,T) \text{ satisfying } \mathbf{u}(0) = \mathbf{u}_0^{\mathsf{T}} \text{ such that}\\ &\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{A}\mathbf{u} = \mathbf{f} \quad \text{for} \quad 0 < t \leq T. \end{aligned} \tag{4.18}$$

$$\text{where} \quad \mathbf{A} = \begin{pmatrix} -1 & -2 & 0 \\ 1 & 4 & -1 \\ 0 & 4 & -1 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} -1 \\ 3 \\ 4 \end{pmatrix}, \quad \text{and} \quad \mathbf{u}_0 = \begin{pmatrix} 0 \\ 2 \\ 4 \end{pmatrix}.$$

For this example we will use the dG(0) method to compute the FE approximations to the solutions of the primal and reduced problems and the cG(1) method when computing the approximation to the solution of the respective dual problem. We can readily write down the true solution $\mathbf{u}$ and the FE approximation $\mathbf{u}^{\mathsf{h}}$ thereof, which is determined by the values $\mathbf{u}_m^{\mathsf{h}}$ at each of the $M+1$ nodes $t_m \in (0\,, T]$. As the entries of the matrix $\mathbf{A}$ and the vector $\mathbf{f}$ are constant, the nodal values are simply

$$\mathbf{u}_m^{\mathsf{h}} = (h_m \mathbf{A} + \mathbf{I})^{-1}(\mathbf{u}_{m-1}^{\mathsf{h}} + h_m \mathbf{f}) \quad \text{for } m = 1, \ldots, M \tag{4.19}$$

with $\mathbf{u}_0^{\mathsf{h}} = \mathbf{u}_0$ and $h_m = t_m - t_{m-1}$, whereas the true solution to the primal problem can be checked to be

$$\begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix} = \begin{pmatrix} 2\exp(t) - \exp(-t) - 1 \\ \exp(-t) + 1 \\ 2\exp(t) + 2\exp(-t) \end{pmatrix}. \tag{4.20}$$

Suppose we are interested in the average value of the third component of the solution

over the interval $[0\,,1]$, whence $T=1$, so with $\mathbf{g}=(0,0,1)^{\mathsf{T}}$ for all $t\in[0\,,1]$,

$$J(\mathbf{u})=\big\langle \mathbf{u}\,,\mathbf{g}\big\rangle=\int_0^1 u_3(t)\,\mathrm{d}t=\int_0^1 4\cosh(t)\,\mathrm{d}t=4\sinh(1)\approx 4.7008. \tag{4.21}$$

Therefore the dual problem associated with (4.18) is

$$\begin{aligned}&\text{Find } \boldsymbol{\varphi}(t)=(\varphi_1(t),\varphi_2(t),\varphi_3(t))^{\mathsf{T}}\in\boldsymbol{\mathcal{C}}^1(0,1) \text{ satisfying } \boldsymbol{\varphi}(1)=\mathbf{0} \text{ such that}\\ &\quad -\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t}+\mathbf{A}^{\mathsf{T}}\boldsymbol{\varphi}=\mathbf{g}\quad\text{for}\quad 1>t\geq 0.\end{aligned} \tag{4.22}$$

Again we can write down the true solution and the FE approximation thereof, which is determined by the nodal values $\boldsymbol{\varphi}^{\mathsf{h}}_m=\boldsymbol{\varphi}^{\mathsf{h}}(t_m)$, which can be computed via

$$\boldsymbol{\varphi}^{\mathsf{h}}_{m-1}=(\mathbf{I}+\frac{h_m}{2}\mathbf{A}^{\mathsf{T}})^{-1}(\mathbf{I}-\frac{h_m}{2}\mathbf{A}^{\mathsf{T}}\boldsymbol{\varphi}^{\mathsf{h}}_m+h_m\mathbf{g})\quad\text{for } m=M,\ldots,1 \tag{4.23}$$

with $\boldsymbol{\varphi}^{\mathsf{h}}_M=\boldsymbol{\varphi}(1)$ and $\mathbf{g}=(0,0,1)^{\mathsf{T}}$. The true solution is

$$\begin{pmatrix}\varphi_1(t)\\ \varphi_2(t)\\ \varphi_3(t)\end{pmatrix}=\begin{pmatrix}-\frac{2}{3}\exp(2(t-1))+2\exp(t-1)+\frac{2}{3}\exp(-(t-1))-2\\ -2\exp(2(t-1))+4\exp(t-1)-2\\ \frac{2}{3}\exp(2(t-1))-2\exp(t-1)+\frac{1}{3}\exp(-(t-1))+1\end{pmatrix}. \tag{4.24}$$

As we are not yet concerned with mesh refinement we just choose a mesh to compute the dG(0) approximation $\mathbf{u}^{\mathsf{h}}$ to the primal problem. To compare performance we choose equally spaced meshes with $M+1$ nodes where $M=10,20,40$. This gives element sizes of $h=0.1,0.05,0.025$ which are all rather 'coarse'. We made this choice because we found that reducing the mesh size for equally spaced meshes, while clearly reducing the induced error $|J(\mathbf{u})-J(\mathbf{u}^{\mathsf{h}})|$, does not 'pay off' for model reduction. We will get back to this performance issue after having discussed the outcome using this setup.

We apply the model reduction algorithm described in 4.1.3 with tolerance $\texttt{TOL}=10^{-1}$ for either of the two initialisations and accompanying rules set out in Section 4.1.4. This 'generous' choice of the model reduction tolerance is justified since the linear functional (4.21) gives an average of $u_3(t)=4\cosh(t)$ over the interval $[0\,,1]$ which is

quite large. The relative error $|(J(\mathbf{u}^{\mathrm{h}}) - J(\mathbf{U}^{\mathrm{h}}))/J(\mathbf{u}^{\mathrm{h}})|$ is however only of order $10^{-2}$ which is satisfactory.

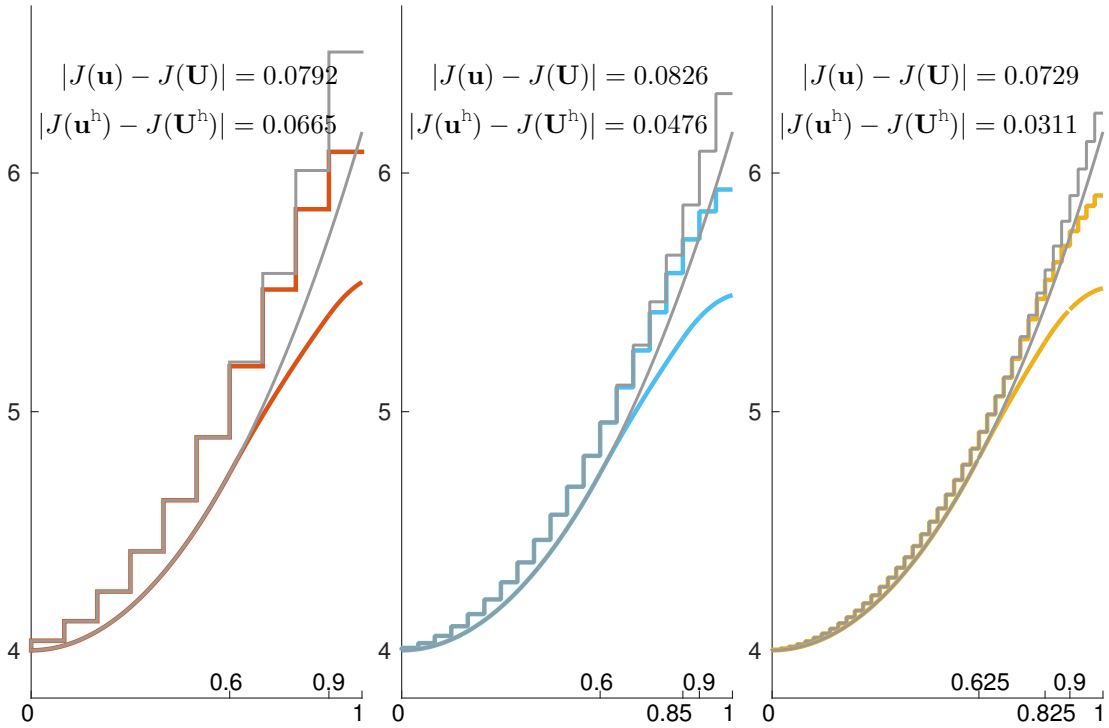



Figure 4.2: The coloured lines depict the reduced model solution $U_3(t)$ and dG(0) approximation $U_3^h(t)$ for meshes with $M+1$ equally spaced nodes, where $M = 10$ (orange), $M = 20$ (blue), $M = 40$ (yellow). The grey lines depict the original full model solution $u_3(t)$ and dG(0) approximation $u_3^h(t)$. Also shown are the values for $|J(\mathbf{u}) - J(\mathbf{U})|$ computed via (4.15) and $|J(\mathbf{u}^{\mathrm{h}}) - J(\mathbf{U}^{\mathrm{h}})|$ computed via (4.16) or (4.17), and only one value is shown as the computations correspond to a high degree.

Note also that we employ the model reduction algorithm using (4.16) as a stopping criterion and (4.17) to drive model reduction. The exact induced error (4.15) is only used for comparative purposes once the reduced model has been generated. Approximate and exact values are shown in Figure 4.2 together with the third component of the solutions and FE approximations to the full and reduced models.

Figure 4.3 shows a graphical representation of the reduced models for the two rules and different meshes. We observe that the reduced models for the two rules and initialisations are similar since, as predicted, the third equation of the reduced model is still present in full for most if not all the interval $[0\,,1]$, and on the subinterval $[0\,,0.6]$ the reduced model does correspond to the full model. This means that the model is

only reduced in the second half of the computational domain and even there only the first two equations are somewhat simplified. This was to be expected as $u_3(t)$ contains a non-negative exponential which means that $\mathrm{d}u_3/\mathrm{d}t$ is already large even when $t$ is of order 1.

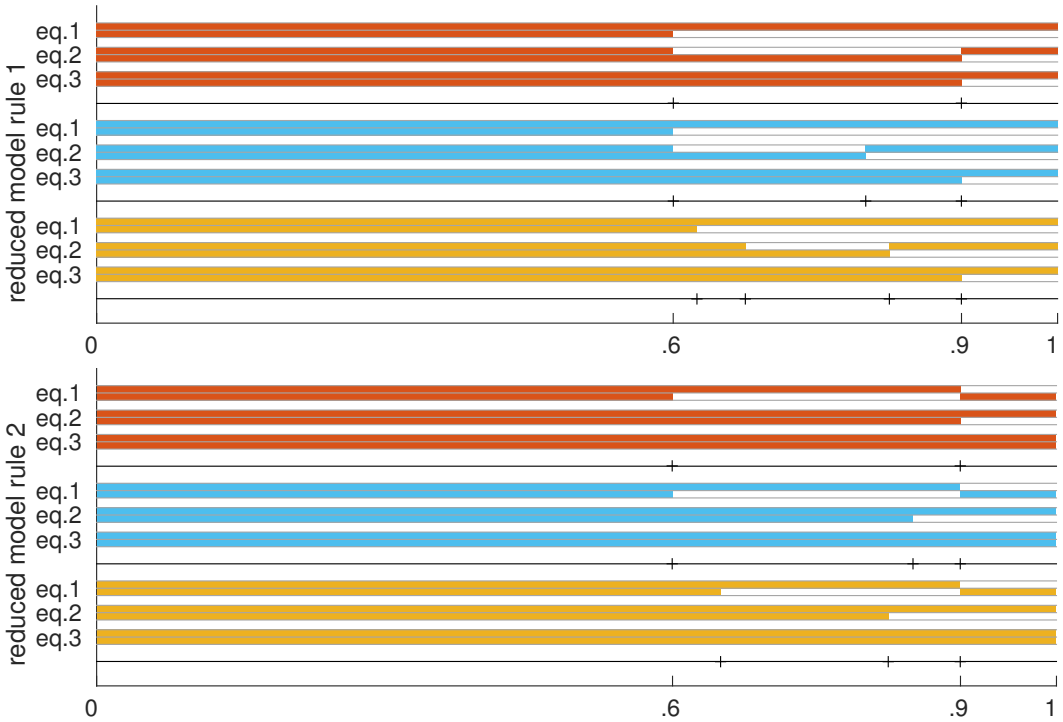



Figure 4.3: Representation of reduced models for meshes with $M+1$ equally spaced nodes, where $M = 10$ (orange), $M = 20$ (blue), $M = 40$ (yellow). Each equation $n$ has two bars, the top bar is coloured whenever $t \in R_n$, meaning that on that subinterval $\mathrm{d}U_n/\mathrm{d}t$ is included, likewise the bottom bar is coloured whenever $t \in S_n$, meaning that on that subinterval $a_n(t, \mathbf{U})$ is included.

However, if we look at the number of iterations that are needed to achieve very similar reduced models we notice that 'rule 2' performs better. We deduce this from Figure 4.4 by comparing the number of iterations of the model reduction algorithm that are needed to achieve the given accuracy. These are marked on the 'iterations' axis in the form *no.iter.rule1(no.iter.rule2)*. This means that with increasing number of elements fewer iterations of the model reduction algorithm are needed to achieve comparable accuracy when using rule 2 instead of rule 1. While it is clear that the finer the mesh the more iterations are needed, since there are more intervals to be included in or excluded from the relevant sets that define the reduced model, we do observe that

model reduction performs best on a coarse mesh, independent of the rule that is used in the algorithm.

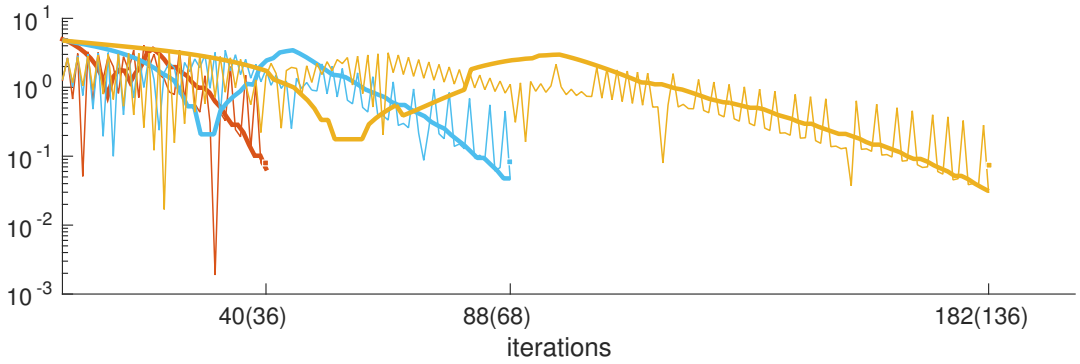



Figure 4.4: The induced error $|J(\mathbf{u}^h) - J(\mathbf{U}^h)|$ is shown at each iteration of the model reduction algorithm for three meshes, $h = 0.100$ (orange), $h = 0.050$ (blue), $h = 0.025$ (yellow), where the thin line is for model reduction using rule 1 and the thick line using rule 2. The dots represent the values of $|J(\mathbf{u}) - J(\mathbf{U})|$ for rule 2 (those for rule 1 are similar and therefore omitted) where $\mathbf{u}$, $\mathbf{U}$ are the true solutions of the full model and respective reduced models.

By 'better performance' of the model reduction algorithm we do not mean that fewer iterations are needed since this is obviously so as there are fewer elements. Finer meshes have more elements that can be included or excluded in each step of the algorithm and this can lead to 'splintering' of the computational domain. What we mean by this can best be shown with Figure 4.3 by observation of the subintervals that are or are not contained in the sets $R_n$ and $S_n$. The subintervals common to all three meshes are $[0\,, 0.6]$ and $[0.9\,, 1]$. What we call 'splintering' happens in the interval $[0.6\,, 0.9]$ which is only present for the mesh with size $h = 0.1$ (reduced model depicted in orange). For the meshes with $h = 0.05$ (blue) and $h = 0.025$ (yellow) that interval 'splinters' into two or even three smaller intervals. Since the sets $R_n$ and $S_n$ define the reduced model this means that the reduced models differ, but not by much, since the additional subintervals are of negligible length in comparison to the 'coarser' subdomains and do not significantly change the value of the induced error $|J(\mathbf{u}^h) - J(\mathbf{U}^h)|$. This we can deduce from Figure 4.2 as well as by comparison of $|J(\mathbf{u}^h) - J(\mathbf{U}^h)|$, computed via (4.16) or (4.17), with $|J(\mathbf{u}) - J(\mathbf{U})|$. To compute the latter via the integral (4.15) we need the solution $\mathbf{U}$ of the reduced model which is given below.

Our goal is ultimately to achieve simplification and elucidation of key features of a mathematical model. Thus our interest is more of a 'global' nature, meaning that we would like to be able to capture time scales of different order for instance, or rapid changes of solution components or asymptotic behaviour. Therefore it seems more appropriate to apply the model reduction algorithm based on 'coarse' rather than 'fine' FE approximations which avoids the 'splintering' issue raised above. One drawback of this approach is though that the induced error $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$ can be quite large as is the case for our example: $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| = 0.26$ for $M = 10$ as opposed to $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| = 0.06$ for $M = 40$. Thus, even though the induced error $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ is of the same order for all three meshes and indeed comparable with $|J(\mathbf{u}) - J(\mathbf{U})|$, it means that $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| + |J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ is overestimating the induced error $|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})|$. We could get around this by first generating a reduced model using a 'coarse' mesh and then compute the FE approximation $\mathbf{U}^{\mathsf{h}}$ to the solution of that reduced model on a much finer mesh. Then, computing also the FE approximation $\mathbf{u}^{\mathsf{h}}$ to the solution of the full model on that finer mesh will reduce both the induced errors $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$ and $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$ and as a consequence lower the estimate for the approximate bound on $|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})|$. Alternatively, one could switch back and forth between mesh refinement and model reduction as is briefly discussed in the Conclusion.

## Solutions to Reduced Models

We only give the ones for rule 2 here as the reduced models are very similar anyhow. The solution $\mathbf{U}(t)$ satisfies $\mathbf{U}(0) = \mathbf{u}_0$ and for $0 < t \leq 1$ it is

$$\mathbf{U}(t) = \begin{cases} \mathbf{U}^{(a)}(t) & \text{for } 0 < t \leq t_a, \\ \mathbf{U}^{(b)}(t) & \text{for } t_a < t \leq t_b, \\ \mathbf{U}^{(c)}(t) & \text{for } t_b < t \leq t_c, \\ \mathbf{U}^{(d)}(t) & \text{for } t_c < t \leq 1, \end{cases}$$

where the reduced model is composed of the following 'locally' reduced models.

**(a)** For $0 < t \leq t_a$ the reduced model is

$$\begin{aligned}
\mathrm{d}U_1/\mathrm{d}t - U_1 - 2\,U_2 &= -1, \;\; U_1(0) = -1,\\
\mathrm{d}U_2/\mathrm{d}t - U_1 + 4\,U_2 - U_3 &= \;\; 3, \;\; U_2(0) = \;\; 2,\\
\mathrm{d}U_3/\mathrm{d}t + 4\,U_2 - U_3 &= \;\; 4, \;\; U_3(0) = \;\; 4,
\end{aligned}$$

which has solution $\mathbf{U}^{(a)}(t) = (U_1^{(a)}(t), U_2^{(a)}(t), U_3^{(a)}(t))^{\mathsf{T}}$, where

$$\begin{aligned}
U_1^{(a)}(t) &= \exp(t) - \exp(-t) - 1,\\
U_2^{(a)}(t) &= \exp(-t) + 1,\\
U_3^{(a)}(t) &= 2\exp(t) + 2\exp(-t).
\end{aligned}$$

**(b)** For $t_a < t \leq t_b$ the reduced model is

$$\begin{aligned}
\mathrm{d}U_1/\mathrm{d}t &= -1, \;\; U_1(t_a) = U_1^{(a)}(t_a) =: U_{a1},\\
\mathrm{d}U_2/\mathrm{d}t - U_1 + 4\,U_2 - U_3 &= \;\; 3, \;\; U_2(t_a) = U_2^{(a)}(t_a) =: U_{a2},\\
\mathrm{d}U_3/\mathrm{d}t + 4\,U_2 - U_3 &= \;\; 4, \;\; U_3(t_a) = U_3^{(a)}(t_a) =: U_{a3},
\end{aligned}$$

which has solution $\mathbf{U}^{(b)}(t) = (U_1^{(b)}(t), U_2^{(b)}(t), U_3^{(b)}(t))^{\mathsf{T}}$, where

$$\begin{aligned}
U_1^{(b)}(t) &= U_{a1} + t_a - t,\\
U_2^{(b)}(t) &= (p_1 + p_2 t - 9t^2/2 + r\exp(3(t_a - t))/27,\\
U_3^{(b)}(t) &= (q_1 + q_2 t - 18t^2 + r\exp(3(t_a - t))/27,
\end{aligned}$$

with coefficients $p_1$, $p_2$, $q_1$, $q_2$, and $r$ given by

$$\begin{aligned}
p_1 &= 9U_{a3} - 9U_{a2} - 12U_{a1} - 21t_a - 9U_{a1}t_a - 9t_a^2/2 + 20,\\
q_1 &= 36U_{a3} - 36U_{a2} - 12U_{a1} - 48t_a - 36U_{a1}t_a - 18t_a^2 + 20,\\
p_2 &= 9U_{a1} + 9t_a + 21,\\
q_2 &= 36U_{a1} + 36t_a + 48,\\
r &= 12U_{a1} + 36U_{a2} - 9U_{a3} - 20.
\end{aligned}$$

**(c)** For $t_b < t \leq t_c$ the reduced model is

$$\begin{aligned}
\mathrm{d}U_1/\mathrm{d}t &= -1, \;\; U_1(t_b) = U_1^{(b)}(t_b) =: U_{b1},\\
\mathrm{d}U_2/\mathrm{d}t &= \;\; 3, \;\; U_2(t_b) = U_2^{(b)}(t_b) =: U_{b2},\\
\mathrm{d}U_3/\mathrm{d}t + 4\,U_2 - U_3 &= \;\; 4, \;\; U_3(t_b) = U_3^{(b)}(t_b) =: U_{b3},
\end{aligned}$$

which has solution $\mathbf{U}^{(c)}(t) = (U_1^{(c)}(t), U_2^{(c)}(t), U_3^{(c)}(t))^{\mathsf{T}}$, where

$$\begin{aligned}
U_1^{(c)}(t) &= U_{b1} - (t - t_b),\\
U_2^{(c)}(t) &= U_{b2} + 3(t - t_b),\\
U_3^{(c)}(t) &= 4(U_{b2} + 2) + 12(t - t_b) - (4U_{b2} - U_{b3} + 8)\exp(t - t_b).
\end{aligned}$$

**(d)** For $t_c < t \leq 1$ the reduced model is

$$\begin{aligned}
U_1 &= 1 - 2\,U_2, \;\; U_1(t_c) = U_1^{(c)}(t_c) =: U_{c1},\\
\mathrm{d}U_2/\mathrm{d}t &= \;\; 3 \;\; , \;\; U_2(t_c) = U_2^{(c)}(t_c) =: U_{c2},\\
\mathrm{d}U_3/\mathrm{d}t - U_3 &= 4 - 4\,U_2, \;\; U_3(t_c) = U_3^{(c)}(t_c) =: U_{c3},
\end{aligned}$$

which has solution $\mathbf{U}^{(d)}(t) = (U_1^{(d)}(t), U_2^{(d)}(t), U_3^{(d)}(t))^{\mathsf{T}}$, where

$$\begin{aligned}
U_1^{(d)}(t) &= 1 - 2U_{c2} - 6(t - t_c),\\
U_2^{(d)}(t) &= U_{c2} + 3(t - t_c),\\
U_3^{(d)}(t) &= 4(U_{c2} + 2) + 12(t - t_c) - (4U_{c2} - U_{c3} + 8)\exp(t - t_c).
\end{aligned}$$

The start and end point of the subintervals $[0\,, t_a]$, $[t_a\,, t_b]$, $[t_b\,, t_c]$, $[t_c\,, 1]$ vary with the element size which we took to be constant $h$. The subintervals generated by the model reduction algorithm are depicted in Figure 4.2 (note that for the 'orange' reduced model we only have three subintervals, meaning that $t_b = t_c$). Also there are displayed the values for $|J(\mathbf{u}) - J(\mathbf{U})|$ and $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})|$.

### 4.2.2 Nonlinear System of ODEs

Our second example to which we apply the model reduction technique is:

$$\begin{aligned}&\text{Find } \mathbf{u}(t) = (u_1(t), u_2(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T) \text{ satisfying } \mathbf{u}(0) = (0,1)^{\mathsf{T}} \text{ such that}\\ &\begin{aligned}\epsilon^2 \frac{\mathrm{d}u_1}{\mathrm{d}t} + \exp(t/\epsilon)(\epsilon u_1 - 1)u_2 &= 0,\\ \frac{\mathrm{d}u_2}{\mathrm{d}t} + u_1 u_2 &= 0,\end{aligned} \quad \text{for } 0 < t \leq T, \text{ where } \epsilon^2 \ll 1. \end{aligned} \tag{4.25}$$

We chose this example because of the small parameter multiplying a derivative which is often to be found in model reduction using asymptotic methods. As was mentioned in the introduction to this chapter, the original model reduction method cannot capture a 'quasi-steady state solution', meaning that the derivative term that is multiplied by the small parameter cannot be dropped. Our extended method allows this, thus making it possible to capture long term and short term behaviour of the solution. We chose this toy example rather than a 'real' one because we can find solutions to the full model and also to the reduced models that are generated as a result of the model reduction algorithm in Section 4.1.3. In the next chapter we will encounter a similar nonlinear system that models enzyme kinetics.

The solution to (4.25) is $\mathbf{u}(t) = (u_1(t), u_2(t))^{\mathsf{T}}$, whose entries can be checked to be

$$u_1(t) = \tanh(t/\epsilon)/\epsilon \text{ and } u_2(t) = 1/\cosh(t/\epsilon).$$

Applying the dG(0) method to compute the FE approximation to the solution of the primal problem yields a system of nonlinear algebraic equations for the nodal values $\mathbf{u}_m^{\mathsf{h}} = (u_{1,m}^h, u_{2,m}^h)^{\mathsf{T}}$ at each of the $M$ nodes $t_m \in (0\,, T]$. That is, the nodal values are determined by solving

$$\begin{aligned}(\epsilon u_{1,m}^h - 1)u_{2,m}^h(\exp(t_m/\epsilon) - \exp(t_{m-1}/\epsilon))/\epsilon^2 + u_{1,m}^h &= u_{1,m-1}^h,\\ (t_m - t_{m-1})u_{1,m}^h u_{2,m}^h + u_{2,m}^h &= u_{2,m-1}^h,\end{aligned}$$

for $m = 1, \ldots, M$ with $u_{1,0}^h = 0$ and $u_{2,0}^h = 1$.

For this example we try a different type of linear functional. Suppose we are interested in the $\mathcal{L}_2$ norm of the $n^{th}$ component of the error $\mathbf{u} - \mathbf{U}$. Although the $\mathcal{L}_2$ norm is nonlinear, we can still use an appropriately defined functional to get a good approximation of the norm. For note that with $g_n = (u_n - U_n)/\|u_n - U_n\|$ and $g_k = 0$ for $k \neq n$ we have

$$\begin{aligned} J(\mathbf{u}) - J(\mathbf{U}) &= \int_0^T (\mathbf{u} - \mathbf{U}) \cdot \mathbf{g} \, \mathrm{d}t, \\ &= \int_0^T (u_n - U_n) g_n \, \mathrm{d}t, \\ &= \int_0^T (u_n - U_n)^2 / \|u_n - U_n\| \, \mathrm{d}t, \\ &= \|u_n - U_n\|. \end{aligned}$$

Thus, to approximate $\left\|u_n^h - U_n^h\right\|$ we define the dual problem for model reduction as

Find $\boldsymbol{\varphi}(t) = (\varphi_1(t), \varphi_2(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,1)$ satisfying $\boldsymbol{\varphi}(T) = \mathbf{0}$ such that

$$\begin{aligned} -\epsilon^2 \frac{\mathrm{d}\varphi_1}{\mathrm{d}t} + A_{11}^*(t)\varphi_1 + A_{12}^*(t)\varphi_2 &= g_1, \\ -\frac{\mathrm{d}\varphi_1}{\mathrm{d}t} + A_{21}^*(t)\varphi_1 + A_{22}^*(t)\varphi_2 &= g_2, \end{aligned} \qquad \text{for} \quad T > t \geq 0. \tag{4.26}$$

The entries of the adjoint matrix $\mathbf{A}^*$ and the right-hand-side vector $\mathbf{g}$ are defined as

$$\begin{aligned} A_{11}^* &= \epsilon \exp(t/\epsilon) \frac{u_2^h + U_2^h}{2}, & A_{12}^* &= \frac{u_2^h + U_2^h}{2}, \\ A_{21}^* &= \exp(t/\epsilon)(\epsilon \frac{u_1^h + U_1^h}{2} - 1), & A_{22}^* &= \frac{u_1^h + U_1^h}{2}, \end{aligned} \qquad g_k = \begin{cases} \dfrac{u_n^h - U_n^h}{\|u_n - u_n^h\|} & k = n, \\ 0 & k \neq n. \end{cases}$$

We want to find a reduced model to the primal problem (4.25) where $T = 10$, $\epsilon = 0.1$, and $n = 1$. As $u_1(t) = \tanh(t/\epsilon)/\epsilon$ grows rapidly initially and comes close to the asymptotic value $1/\epsilon$ for $t \approx 0.5$, we can expect a reduced model that reflects this. Suppose now that we have computed $\mathbf{u}^{\mathsf{h}}$ on a given mesh with the dG(0) method. Because $\epsilon^2 \mathrm{d}u_1/\mathrm{d}t$ is negligible once $\mathrm{d}u_1/\mathrm{d}t$ is of order 1 we apply the model reduction algorithm with just the one initialisation, $R_n = \varnothing$ and $S_n = (0\,, T]$ for $n = 1, 2$. This

initial reduced model is therefore a system of two algebraic equations which has the constant solution $\mathbf{U}^{\mathsf{h}} = (0, 0)^{\mathsf{T}}$.

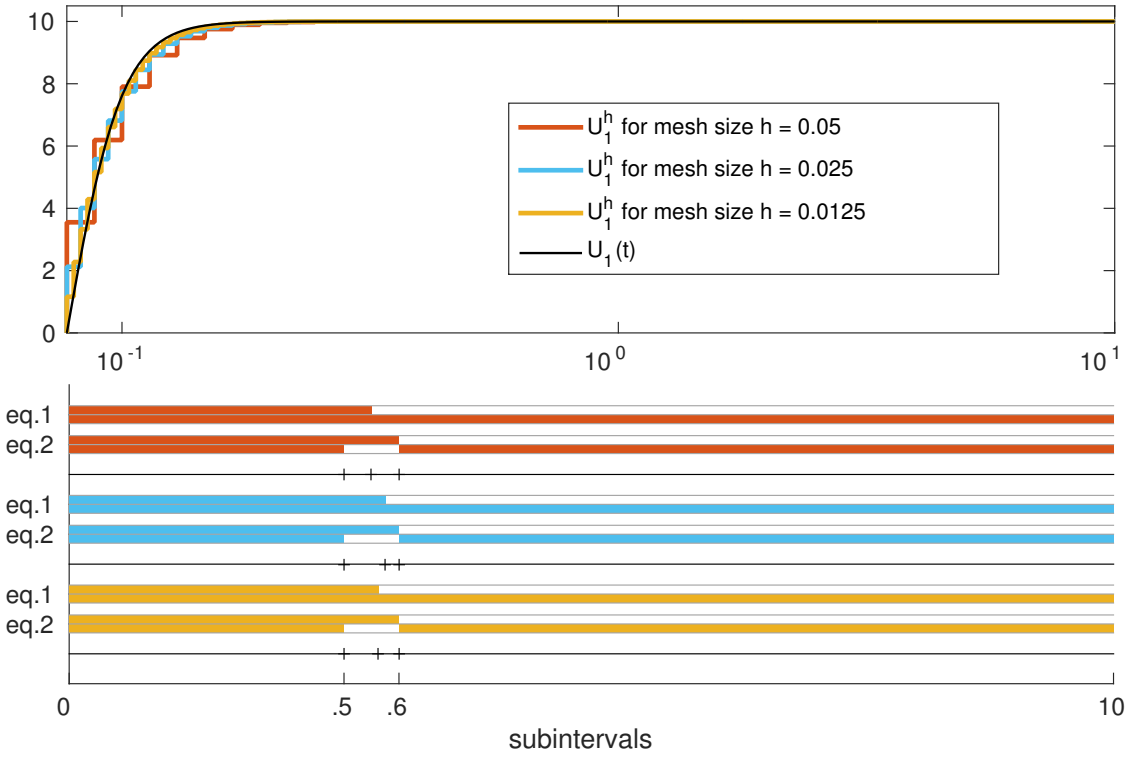



Figure 4.5: Top: first component $U_1(t)$ of full model (black) and approximations $U_1^h$ of reduced model solution computed on uniform meshes with $M = 200$ (orange), $M = 400$ (blue), $M = 800$ (yellow). Bottom: Representation of reduced models for the three meshes shown on top, where $t$ is on a logarithmic scale.

Application of the model reduction algorithm with this initialisation and corresponding rule (rule 2) yields the reduced model solution $\mathbf{U}(t) = (U_1(t), U_2(t))^{\mathsf{T}}$, where

$$U_1(t) = \begin{cases} \frac{1}{\epsilon}\tanh(t/\epsilon) & \text{for } 0 \le t < 0.5, \\ \frac{1}{\epsilon}\left((\tanh(0.5/\epsilon) - 1)\exp(\frac{\exp(0.5/\epsilon)-\exp(t/\epsilon)}{\cosh(0.5/\epsilon)}) + 1\right) & \text{for } 0.5 \le t \le t_\epsilon, \\ \frac{1}{\epsilon} & \text{for } t_\epsilon < t \le 10, \end{cases}$$

$$U_2(t) = \begin{cases} 1/\cosh(t/\epsilon) & \text{for } 0 \le t < 0.5, \\ 1/\cosh(0.5/\epsilon) & \text{for } 0.5 \le t \le 0.6, \\ 0 & \text{for } 0.6 < t \le 10, \end{cases}$$

and $t_\epsilon \in \{0.55,\, 0.575,\, 0.5625\}$. The reduced model solution and approximations thereof are displayed in Figure 4.5 as well as a graphical representation of the reduced models for three equally spaced meshes. Note also that

$$(\tanh(0.5/\epsilon) - 1)\exp(\frac{\exp(0.5/\epsilon) - \exp(t/\epsilon)}{\cosh(0.5/\epsilon)}) + 1 \approx 0.9999$$

for any $t \in [0.5\,, t_\epsilon]$. Thus, when computing $J(\mathbf{u}) - J(\mathbf{U})$ we will do this with

$$u_1(t) = \frac{1}{\epsilon}\tanh(t/\epsilon) \quad \text{for } 0 \le t \le 10,$$

$$\text{and } U_1(t) = \begin{cases} \frac{1}{\epsilon}\tanh(t/\epsilon) & \text{for } 0 \le t \le 0.5, \\ \frac{1}{\epsilon} & \text{for } 0.5 < t \le 10, \end{cases}.$$

The error induced in the linear functional when $\mathbf{u}$ is replaced by $\mathbf{U}$ is

$$\begin{aligned} |J(\mathbf{u}) - J(\mathbf{U})| &= \frac{1}{\epsilon}\left(\int_{0.5}^{10} (\tanh(t/\epsilon) - 1)^2 \,\mathrm{d}t\right)^{\frac{1}{2}}, \\ &= \frac{1}{\epsilon}\Big(\big[2\epsilon\log(\tanh(t/\epsilon) + 1) - \epsilon\tanh(t/\epsilon)\big]_{0.5}^{10}\Big)^{\frac{1}{2}}, \\ &= \sqrt{10}\left(2\log\left(\frac{\tanh(100) + 1}{\tanh(5) + 1}\right) - \tanh(100) + \tanh(5)\right)^{\frac{1}{2}}, \end{aligned}$$

which is of the same order ($10^{-4}$) as

$$|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| = \left(\sum_{m=1}^{M} (u_m^h - U_m^h)^2 h_m\right)^{\frac{1}{2}},$$

where the exact values are given in the legend to Figure 4.6.

That $|J(\mathbf{u}) - J(\mathbf{U})|$ is of the same order as $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})|$ only means that $\mathbf{U}(t)$ is 'as good an approximation to' $\mathbf{u}(t)$ as $\mathbf{U}^\mathsf{h}$ is to $\mathbf{u}^\mathsf{h}$. It does not however mean that $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| = \left\|u_1(t) - u_1^h(t)\right\|$ is small. We can compute the latter directly as we have both $\mathbf{u}$ and $\mathbf{u}^\mathsf{h}$. The square of the norm is

$$|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})|^2 = \sum_{m=1}^{M} \int_{t_{m-1}}^{t_m} \left(\frac{1}{\epsilon}\tanh^2(t/\epsilon) - u_{1,m}^h\right)^2 \mathrm{d}t,$$

$$
\begin{aligned}
&= \sum_{m=1}^{M} \left[ (t - \epsilon \tanh(t/\epsilon))/\epsilon^2 - 2u_m^h \log(\cosh(t/\epsilon)) + (u_m^h)^2 t \right]_{t_{m-1}}^{t_m}, \\
&= (t_M - \epsilon \tanh(t_M/\epsilon))/\epsilon^2 \\
&\qquad - \sum_{m=1}^{M} \left( 2u_m^h \log(\frac{\cosh(t_m/\epsilon)}{\cosh(t_{m-1}/\epsilon)}) + (u_m^h)^2 h_m \right),
\end{aligned}
\tag{4.27}
$$

and thus $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| = 0.4853$ ($M = 200$), $0.2676$ ($M = 400$), $0.1413$ ($M = 800$). This suggests that if model reduction is 'measured' by $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| + |J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})|$ instead of just $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})|$ then the estimate for $|J(\mathbf{u}) - J(\mathbf{U})|$ will generally be a much larger value.

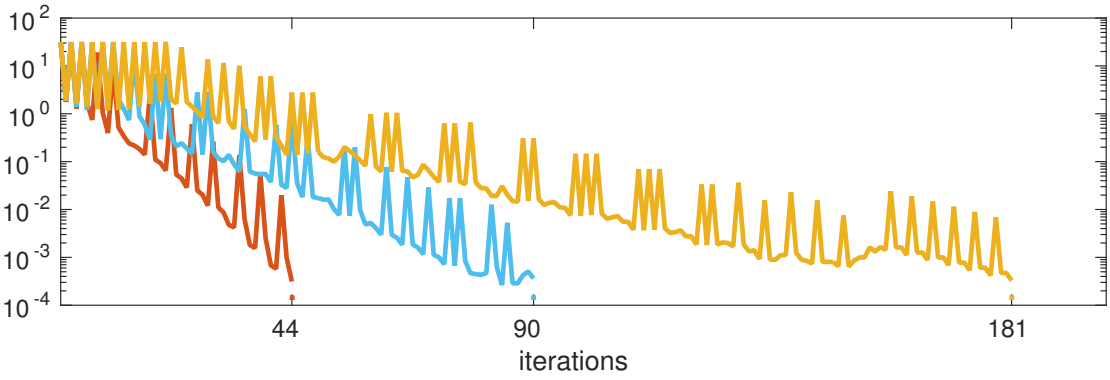



Figure 4.6: The induced error $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| \approx \left\| u_1^h - U_1^h \right\|$ is shown at each iteration of the model reduction algorithm for the three meshes, $M = 200$ (orange), $M = 400$ (blue), $M = 800$ (yellow). The dots represent the value of $\|u_1 - U_1\| = 0.000144$, whereas $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| \approx 0.0003$, the exact values are $0.000313$ ($M = 200$), $0.000363$ ($M = 400$), and $0.000327$ ($M = 800$).

### 4.2.3 Summary and Outlook

We were able to show that the model reduction method can give satisfactory results, and it seems to be particularly suited for systems of ODEs that contain different timescales. Indeed, we will discuss an enzyme kinetics model in the next chapter for which the method produces results that fulfil the aim of this thesis. However, due to the relaxation of the conditions imposed on the solutions or their approximations and also the allowances for the form of the reduced model operator, we have identified a set of sensitivities pertaining to the choices that are made for the algorithm's implementation. We will refer to these whenever these choices might influence the outcome.

# CHAPTER 5

# Combination of Model Reduction and Mesh Refinement

When reviewing various model reduction techniques in Chapter 3, in particular model reduction by duality, we have already mentioned the potential for combining model reduction with adaptive mesh refinement in Section 3.2.2. We said there that it seemed appropriate to combine the two techniques instead of relying on knowledge of the true solution (or rather assuming that such is known). In the last chapter we introduced the model reduction algorithm which not only approximates $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})|$ but also yields a reduced model and approximation to the solution thereof. In this chapter we will first present a mesh refinement algorithm to get an estimate of a bound on $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})|$ but the major part will deal with the combination of the two algorithms to ultimately yield a bound on the quantity $|J(\mathbf{u}) - J(\mathbf{U}^\mathsf{h})|$.

Note that for any given $\texttt{TOL} > 0$ and $\vartheta$ with $0 < \vartheta < 1$:

$$\begin{aligned} &\text{if } |J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| \leq \vartheta \times \texttt{TOL} \text{ and } |J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| \leq (1-\vartheta) \times \texttt{TOL}, \\ &\text{then } \quad |J(\mathbf{u}) - J(\mathbf{U}^\mathsf{h})| \leq |J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| + |J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| \leq \texttt{TOL}. \end{aligned} \tag{5.1}$$

This splitting allows allocation of the 'error' to mesh refinement ($\vartheta$ close to 1) or to model reduction ($\vartheta$ closer to 0). This can be useful when, say, $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})|$ is orders of magnitude larger than $|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})|$, as was the case for the toy example at the end of the preceding chapter. If we are not so much concerned about the overall error not exceeding TOL but are rather interested in generating a 'good' reduced model then 'error allocation' to model reduction would make sense. If on the other hand the 'accuracy' of the approximation to the solution of the full model (and as a consequence also of the reduced model) is of more importance then mesh refinement should carry much greater weight.

We will now proceed with the introduction of the mesh refinement algorithm and the combination with the model reduction algorithm of Chapter 4. Thereafter we will apply the combined algorithm first to the nonlinear model problem from Section 4.2 of the previous chapter and then to 'real' (as opposed to 'model') examples which will form the main part of this chapter.

## 5.1 Mesh Refinement Based on Linear Functionals of the Solution

Mesh refinement made its first appearance in Section 3.2.2 but the algorithm was not introduced there. We therefore restate the setup and results here, for derivations of (estimations of) bounds however we refer back to that section.

Let $\mathbf{u}(t) = (u_1(t), \ldots, u_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T)$ denote the solution of the primal problem

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{a}(t,\mathbf{u}) = \mathbf{f} \quad \text{for} \quad 0 < t \leq T \quad \text{with} \quad \mathbf{u}(0) = \mathbf{u}_0. \tag{5.2}$$

We choose the dG($q$) method to compute the FE approximation $\mathbf{u}^{\mathsf{h}}$ to $\mathbf{u}$ on a mesh $\mathcal{I}_h = \{I_m\}_{m=1}^M$ with $M$ elements $I_m = (t_{m-1}\,, t_m]$. We are given a linear functional of the solution $J(\mathbf{u}) = \big\langle \mathbf{u}\,, \mathbf{g} \big\rangle$ and want to get an estimate of the bound on the error induced in the linear functional when $\mathbf{u}$ is replaced by $\mathbf{u}^{\mathsf{h}}$. To this end we use an appropriate dual problem which differs depending on the entries $g_n$ of $\mathbf{g}(t) = (g_1(t), \ldots, g_N(t))^{\mathsf{T}}$, distinguishing between the two cases:

case (1) $g_n(T)$ is finite for all $n = 1, \ldots, N$, and

case (2) there exists $n \in \{1, \ldots, N\}$ such that $g_n(t) = \delta(t-T)$ and $g_k(t) = 0$ for all other $k \in \{1, \ldots, N\} \setminus \{n\}$.

The dual problem can now be defined as:

Find $\boldsymbol{\varphi}(t) = (\varphi_1(t), \ldots, \varphi_N(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T)$ where $\boldsymbol{\varphi}$ satisfies

case (1) $\boldsymbol{\varphi}(T) = \mathbf{0}$ such that

$$-\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} = \mathbf{g} \quad \text{for} \quad T > t \geq 0, \tag{5.3}$$

case (2) $\varphi_n(T) = 1$ and $\varphi_k(T) = 0$ for all $k \in \{1, \ldots, N\} \setminus \{n\}$ such that

$$-\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}t} + \mathbf{A}^*\boldsymbol{\varphi} = \mathbf{0} \quad \text{for} \quad T > t \geq 0, \tag{5.4}$$

where in either case the entries of the adjoint matrix $\mathbf{A}^*$ are defined as

$$A^*_{ij}(t) = \int_0^1 \frac{\partial a_j}{\partial u_i}(s\mathbf{u}(t) + (1-s)\mathbf{u}^{\mathsf{h}}(t))\,\mathrm{d}s \;\text{ for } i,j = 1,\ldots,N, \tag{5.5}$$

which we though have to approximate, as $\mathbf{u}$ is unknown, by

$$A^*_{ij}(t) \approx \left.\frac{\partial a_j}{\partial u_i}\right|_{\mathbf{u}(t)=\mathbf{u}^{\mathsf{h}}(t)} \quad \text{for } i,j = 1,\ldots,N. \tag{5.6}$$

In either case we can express the error induced into the linear functional when replacing $\mathbf{u}$ by $\mathbf{u}^{\mathsf{h}}$, as derived in Section 3.2.2 for (3.33) and (3.34), as

$$|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| \approx \left|\sum_{m=1}^{M}\left(\int_{t_{m-1}}^{t_m} \mathrm{Res}\,(\mathbf{u}^{\mathsf{h}})\cdot\boldsymbol{\varphi}^{\mathsf{h}}\,\mathrm{d}t - [\mathbf{u}^{\mathsf{h}}]_{m-1}\cdot\boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1})\right)\right|, \tag{5.7}$$

$$\lessapprox \sum_{m=1}^{M}\left|\int_{t_{m-1}}^{t_m} \mathrm{Res}\,(\mathbf{u}^{\mathsf{h}})\cdot\boldsymbol{\varphi}^{\mathsf{h}}\,\mathrm{d}t - [\mathbf{u}^{\mathsf{h}}]_{m-1}\cdot\boldsymbol{\varphi}^{\mathsf{h}}(t_{m-1})\right|, \tag{5.8}$$

$$\lessapprox \sum_{n=1}^{N}\sum_{m=1}^{M}\left|\left(\int_{t_{m-1}}^{t_m} Res_{\,n}(\mathbf{u}^{\mathsf{h}})\varphi_n^h\,\mathrm{d}t + [u_n^h]_{m-1}\varphi_n^h(t_{m-1})\right)\right| \tag{5.9}$$

where $\boldsymbol{\varphi}^{\mathsf{h}}$ is a continuous Galerkin approximation of the solution to dual problem (5.3) or (5.4). Even if we would replace $\boldsymbol{\varphi}^{\mathsf{h}}$ with the true solution $\boldsymbol{\varphi}$ the right-hand-side of (5.7) would nevertheless just be an estimate of the bound of $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$. This is because (either of) the dual problem(s) is not the 'true' one as the adjoint matrix is already an approximation, see (5.6). This is a standard approach since if $\mathbf{u}$ were known then one would not need to resort to approximations anyway. In most applications estimates are enough to drive mesh refinement without the need for exact bounds. We will use all the above results for adaptive mesh refinement using a duality argument.

### 5.1.1 Mesh Refinement Algorithm

Set iteration counter $i = 0$ and define initial mesh $\mathcal{I}_h^{(i)} = \{I_m^{(i)}\}_{m=1}^{M_{(i)}}$ with $M_{(i)}$ elements $I_m^{(i)} = (t_{m-1}^{(i)}, t_m^{(i)}]$.

1. Compute discontinuous Galerkin approximation $\mathbf{u}^\mathsf{h}$ to the solution of the primal problem (5.2) on mesh $\mathcal{I}_h^{(i)}$. Use this to get adjoint matrix $\mathbf{A}^*(t)$ as defined in (5.6) and then compute continuous Galerkin approximation $\boldsymbol{\varphi}^\mathsf{h}$ to the solution of the dual problem, either (5.3) or (5.4).

2. Compute approximation to $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})|$ via right-hand-side of (5.7). Stop if this does not exceed a given tolerance `TOL`.

3. Generate a new mesh $\mathcal{I}_h^{(i+1)}$ by inserting a mesh point $(t_{m-1}^{(i)} + t_m^{(i)})/2$ into each $I_m^{(i)} = (t_{m-1}^{(i)}, t_m^{(i)}]$ for all those $m$ for which the contributions to the sum (5.8) are largest.

4. Set $i \mapsto i + 1$ and return to step 1.

Once the iteration stops, we will have computed an approximation $\mathbf{u}^\mathsf{h}$ to $\mathbf{u}$ on a mesh $\mathcal{I}_h^{(i)}$ for which the induced error $|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})|$ is estimated to be less than or equal to the given tolerance.

Before we go on we should look at Step 3. again and clarify which elements we are refining. As it is presented above it implies finding the set $K \subseteq \{1, \ldots, M\}$ that contains all element indices $m$ for which the contribution to the sum (5.8) exceeds $\texttt{TOL}/M_{(i)}$. This is but one way of locating the elements to be refined. One could also just go one step at a time and find the equation index $n$ and element index $m$ such that the contribution to the double sum (5.9) is largest and then just refine the element $I_m^{(i)} = (t_{m-1}^{(i)}, t_m^{(i)}]$, or include more than just one. Of all the possibilities we chose the one for the examples in this thesis that we described first.

### 5.1.2 Combined Algorithm

Now that both algorithms have been introduced, we just have to choose an overall tolerance $\texttt{TOL} > 0$ and a 'weighting' factor $\vartheta$ with $0 < \vartheta < 1$, as discussed at the start of this chapter, and simply combine the two:

- Run mesh refinement algorithm as described in 5.1.1 with tolerance $\vartheta \times \texttt{TOL}$.
- Run model reduction algorithm as described in 4.1.3 with tolerance $(1-\vartheta) \times \texttt{TOL}$.

This yields a reduced model and an approximation $\mathbf{U}^{\mathsf{h}}$ to the solution thereof such that $|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})| \approx \texttt{TOL}$.

## 5.2 Application of Combined Method

### 5.2.1 Nonlinear Model Problem

We apply the combined algorithm to the example already discussed in Section 4.2.2 when validating the model reduction technique. We saw there that

$$\mathbf{u}(t) = \left( \frac{1}{\epsilon} \tanh(t/\epsilon)\,, \frac{1}{\cosh(t/\epsilon)} \right)^{\mathsf{T}}$$

satisfies the initial condition $\mathbf{u}(0) = (0\,,1)^{\mathsf{T}}$ and solves the nonlinear ODE

$$\begin{aligned} \epsilon^2 \frac{\mathrm{d}u_1}{\mathrm{d}t} &+ \exp(t/\epsilon)(\epsilon u_1 - 1)u_2 &&= 0, \\ \frac{\mathrm{d}u_2}{\mathrm{d}t} &+ u_1 u_2 &&= 0, \end{aligned}$$

where we took $\epsilon = 0.1$. We applied the model reduction algorithm based on a linear functional of the solution that approximates the $\mathcal{L}_2$ norm of the error of the first component. That is, we defined $\mathbf{g}(t) = (g_1(t), g_2(t))^{\mathsf{T}}$ in $J(\mathbf{u}) = \left\langle \mathbf{u}\,, \mathbf{g} \right\rangle$ with components $g_1(t) = (u_1^h(t), U_1^h(t))/\left\|u_1^h - U_1^h\right\|$ and $g_2(t) = 0$ to compute the induced error $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| = \left\|u_1^h - U_1^h\right\|$. This was possible since both dG(0) approximations to the full and reduced problems were known. For mesh refinement however

we usually cannot define the $n^{th}$ component of the vector $\mathbf{g} = (g_1(t), \ldots, g_N(t))^{\mathsf{T}}$ as $g_n(t) = (u_n(t), u_n^h(t))/\left\|u_n - u_n^h\right\|$ since the true solution $\mathbf{u}(t)$ is generally not known. Although $\mathbf{u}(t)$ is known for this specific example, we will nevertheless not use it for mesh refinement but rather present an estimate of the error $\mathbf{u} - \mathbf{u}^{\mathsf{h}}$ for the general case where $\mathbf{u}$ is unknown.

To estimate the error $\mathbf{u} - \mathbf{u}^{\mathsf{h}}$ we could replace $\mathbf{u}$ with $\mathbf{u}^{\mathsf{2h}}$ which denotes the dG(0) approximation to $\mathbf{u}$ computed by doubling the mesh. This means that if the original mesh has $M + 1$ nodes and $M$ elements $I_m = (t_{m-1}\,, t_m]$, then the new mesh has $2M + 1$ nodes and $2M$ elements, $M$ of each the left and right halves of $I_m$ which are $I_{m_l} = (t_{m-1}\,, (t_{m-1} + t_m)/2]$ and $I_{m_r} = ((t_{m-1} + t_m)/2\,, t_m]$. The dG(0) approximation $\mathbf{u}^{\mathsf{2h}}$ computed on that mesh is determined by the nodal values $\mathbf{u}_{m_l}^{\mathsf{2h}}$ and $\mathbf{u}_{m_r}^{\mathsf{2h}}$ for $m = 1, \ldots, M$ with $\mathbf{u}_0^{\mathsf{h}} = \mathbf{u}_0^{\mathsf{2h}}$. Thus we could approximate the solution by

$$\mathbf{u}(t) \approx \tilde{\mathbf{u}}^{\mathsf{h}}(t) \text{ where } \tilde{\mathbf{u}}^{\mathsf{h}}\big|_{I_m} := \frac{1}{2}(\mathbf{u}_{m_l}^{\mathsf{2h}} + \mathbf{u}_{m_r}^{\mathsf{2h}})$$

and hence approximate the error $\mathbf{u} - \mathbf{u}^{\mathsf{h}}$ by $\tilde{\mathbf{u}}^{\mathsf{h}} - \mathbf{u}^{\mathsf{h}}$. We will use this estimate for the error not only for this example but also whenever the linear functional of the solution should approximate an $\mathcal{L}_2$ error of a solution component.

We choose an overall tolerance of $\texttt{TOL} = 0.01$ with most of the weight given to mesh refinement, choosing $\vartheta = 0.9$, whence tolerance for mesh refinement is $\vartheta \times \texttt{TOL} = 0.009$. We start the mesh refinement algorithm described in Section 5.1.1 with an initial mesh of $M = 100$ equally spaced nodes discretising the computational domain $I = [0\,, 10]$, so initial mesh size is $h_m = 0.1$ for all $m = 1, \ldots, M$. As the solution $u_1(t)$ changes rapidly for $t \in (0\,, 0.5)$ we expect the refined mesh to reflect this. Indeed we see in Figure 5.1 that as $u_1^h(t)$ increases so do the interval lengths $h_m = |I_m|$. The smallest element size $h_m = t_m - t_{m-1}$ is smaller than $10^{-3}$ for nodes $0 < t_m \leq 0.075$, and the largest are unchanged at $h_m = 0.1$ for nodes $0.5 \leq t_m \leq 10$. A mesh of equally spaced nodes such that the mesh size would have that smallest element size would have 12800 nodes as opposed to our refined mesh which has 260 nodes which is fifty times less.

There is also not much 'gain' in accuracy with a finer mesh, as we can see by comparing the values we get for $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| \approx \left\|u_1 - u_1^h\right\|$ for $\mathbf{u}^{\mathsf{h}}$ computed on the very fine mesh or the refined mesh (taking the square root of (4.27)) and the approximate value for $|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})|$ given once the mesh refinement algorithm stops:

$$|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| = \begin{cases} 0.009348 & \text{mesh with 12800 equally spaced nodes,} \\ 0.012173 & \text{refined mesh with 260 nodes,} \\ 0.008378 & \text{estimate of mesh refinement algorithm.} \end{cases}$$

We see that all have a value of about $10^{-2}$ which is actually what we chose for the overall tolerance `TOL`.

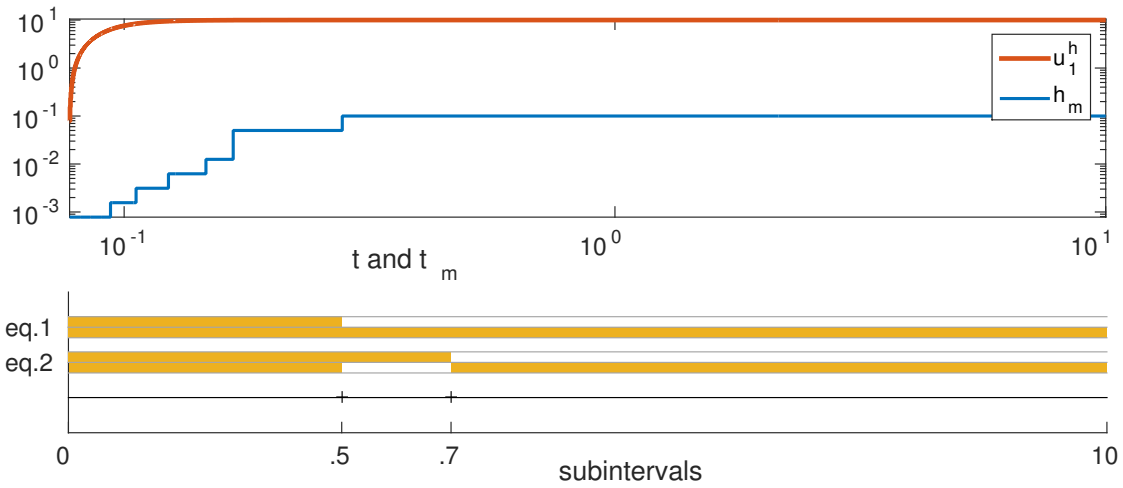



Figure 5.1: Top: Mesh size $h_m$ and $u_1^h(t)$ on the same plot where both axes have logarithmic scale. Bottom: graphical representation of the reduced model where the subintervals are also on a logarithmic scale to correspond with the plot of the reduced model solution on the plot shown on top. The top bar of equation $n$ is coloured whenever $t \in R_n$ which means that the local reduced model includes the term $\mathrm{d}U_n/\mathrm{d}t$, likewise the bottom bar is coloured when $t \in S_n$ and therefore the term $a_n(t, \mathbf{U})$ is included locally.

Now we use the dG(0) approximation $\mathbf{u}^{\mathsf{h}}$ computed on the refined mesh for the model reduction algorithm described in Section 4.1.3 with dual problem as defined in Section 4.2.2 and tolerance $(1 - \vartheta) \times \mathtt{TOL} = 0.001$. It should not be surprising that many more iterations are needed for the refined mesh than the equally spaced meshes seen there to arrive at a very similar reduced model, as depicted in Figure 5.1. This is so because

there are now over 160 elements $I_m$ in the 'critical' interval $[0\,,0.5]$ as opposed to only 10, 20, or 40 for the equally spaced meshes used in the last chapter and therefore that many more choices for inclusion or exclusion of these in the relevant $R_n$ and $S_n$. Once the model reduction algorithm stops we also get a dG(0) approximation $\mathbf{U}^{\mathsf{h}}$ of the solution to the reduced model and $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| = 0.000423$.

This is comparable to $|J(\mathbf{u}) - J(\mathbf{U})| = 0.000144$ and also $|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| \approx 0.0003$ which we computed for the various equally spaced meshes in Section 4.2.2. What differs though is the overall induced error $|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})|$ which for the refined mesh is estimated to be within the tolerance $\texttt{TOL} = 0.01$ unlike for the meshes with equally spaced nodes used in Section 4.2.2 for which the estimates are at least 40, 20, and 10 times larger (see there). This stresses the importance of the combination of model reduction with adaptive mesh refinement. For were we to rely on a very fine mesh to approximate the unknown solution then this would add greatly to the computational effort whilst not really contributing much to model reduction.

### 5.2.2 Enzyme Kinetics

A simple model for a chemical reaction sequence, involving only one reactant and an enzyme as a catalyst, is

$$E + S \underset{k_{-1}}{\overset{k_{+1}}{\rightleftharpoons}} C \xrightarrow{k_{+2}} P + E, \tag{5.10}$$

where $E$, $S$, $C$, $P$ denote the enzyme, substrate, complex, and product respectively. Enzymes act as catalysts and bind to the substrate (the reactant) to form a complex and are subsequently released once a product is formed. The subscripted $k$'s are reaction rate constants, a '+' denoting a 'forward' reaction and '-' a 'backward reaction, where the first forward reaction is a very fast reaction with rate constant $k_{+1}$ several orders of magnitude faster than the other two and so $k_{+1} \gg k_{-1}$, $k_{+2}$. Applying the Law of Mass Action to (5.10) enables us to formulate a mathematical model that describes

the rates of change of the various reactants:

$$
\begin{aligned}
&\text{Find } \mathbf{u}(t) = (s(t), c(t), e(t), p(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T) \text{ satisfying } \mathbf{u}(0) = (s_0, 0, e_0, 0)^{\mathsf{T}} \\
&\text{such that} \\
&\begin{aligned}
\frac{\mathrm{d}s}{\mathrm{d}t} + \quad & k_{+1}es - k_{-1}c \quad = 0, \\
\frac{\mathrm{d}c}{\mathrm{d}t} - & k_{+1}es + (k_{-1} + k_{+2})c = 0, \\
\frac{\mathrm{d}e}{\mathrm{d}t} + & k_{+1}es - (k_{-1} + k_{+2})c = 0, \\
\frac{\mathrm{d}p}{\mathrm{d}t} - \quad & k_{+2}c \quad = 0,
\end{aligned}
\qquad \text{for } 0 < t \leq T.
\end{aligned}
\tag{5.11}
$$

It is quite straightforward to reduce this model to a two equation model by noting that the fourth equation decouples and that $e = e_0 - c$ since $\mathrm{d}e/\mathrm{d}t + \mathrm{d}c/\mathrm{d}t = 0$. The model (5.11) can therefore also be written as

$$
\begin{aligned}
&\frac{\mathrm{d}s}{\mathrm{d}t} + k_{+1}e_0 s - (k_{+1}s + k_{-1})c = 0, \qquad s(0) = s_0, \\
&\frac{\mathrm{d}c}{\mathrm{d}t} - k_{+1}e_0 s + (k_{+1}s + k_{-1} + k_{+2})c = 0, \qquad c(0) = 0, \\
&\text{with } e(t) = e_0 - c(t) \text{ and } p(t) = k_{+2}\int_0^T c(t)\,\mathrm{d}t.
\end{aligned}
\tag{5.12}
$$

## Asymptotic Model Reduction

We introduce the non-dimensional variables and parameters, following the exposition in Chapter 6 of Murray [43]:

$$
u_s := \frac{s}{s_0},\ u_c := \frac{c}{e_0},\ \kappa := \frac{k_{-1} + k_{+2}}{k_{+1}s_0},\ \lambda := \frac{k_{+2}}{k_{+1}s_0},\ \text{ and } \epsilon := \frac{e_0}{s_0}. \tag{5.13}
$$

In enzyme-kinetic reactions it is often the case that the initial substrate concentration by far exceeds the initial enzyme concentration. This means that $e_0 \ll s_0$ and that the rescaled time variables $\tau = k_{+1}e_0 t$ and $\hat{\tau} = k_{+1}s_0 t$ stand for the 'long' and for the

'short' timescales. Substitution of the non-dimensional variables and parameters (5.13) with $\tau = k_{+1}e_0 t$ into (5.12) leads, after suitable rearranging, to the model

$$\begin{aligned} \frac{\mathrm{d}u_s}{\mathrm{d}\tau} + u_s - (u_s + \kappa - \lambda)u_c &= 0, \qquad & u_s(0) &= 1, \\ \epsilon\frac{\mathrm{d}u_c}{\mathrm{d}\tau} - u_s + \quad (u_s + \kappa)u_c \quad &= 0, \qquad & u_c(0) &= 0. \end{aligned} \tag{5.14}$$

When the (non-dimensional) complex concentration $u_c$ is almost at equilibrium after the (fast) transient stage, we have $\epsilon \mathrm{d}u_c/\mathrm{d}\tau \approx 0$ since $\epsilon \ll 1$. This means that the full model (5.14) reduces to

$$\begin{aligned} \frac{\mathrm{d}U_s}{\mathrm{d}\tau} \;+\; U_s \;-\; (U_s + \kappa - \lambda)U_c \;&=\; 0, \qquad & U_s(0) \;&=\; 1, \\ -\; U_s \;+\; \quad (U_s + \kappa)U_c \quad \;&=\; 0, \end{aligned} \tag{5.15}$$

which has solution $\mathbf{U}(\tau) = (U_s(\tau), U_c(\tau))^{\mathsf{T}}$ where $U_s(\tau)$ and $U_c(\tau)$ are (implicity for the substrate concentration) given by

$$U_s(\tau) + \kappa \log(U_s(\tau)) = 1 - \lambda\tau, \quad U_c(\tau) = \frac{U_s(\tau)}{U_s(\tau) + \kappa}. \tag{5.16}$$

The solution component $U_c(\tau)$ in (5.16) is called the quasi-steady state approximation for the complex concentration. The asymptotic model (5.15) has no dependency on the initial complex concentration as the second equation has become an algebraic equation and as such the 'past' of the complex concentration has been 'lost'. This does not matter however, as the quasi-steady state approximation is only valid for times larger than $\epsilon$, and for those times $u_c \approx U_c$ as can be clearly seen in Figure 5.2.

To recover the transient we rescale time by $\hat{\tau} = k_{+1}s_0 t$ and substitution into (5.12) together with the non-dimensional variables and parameters (5.13) now yields, after

suitable rearranging, the non-dimensional model

$$\begin{aligned} \frac{\mathrm{d}u_s}{\mathrm{d}\hat{\tau}} + \epsilon(u_s - (u_s + \kappa - \lambda)u_c) &= 0, \qquad & u_s(0) &= 1, \\ \frac{\mathrm{d}u_c}{\mathrm{d}\hat{\tau}} - u_s + (u_s + \kappa)u_c &= 0, \qquad & u_c(0) &= 0. \end{aligned} \tag{5.17}$$

Since $\epsilon \ll 1$ we have $\mathrm{d}u_c/\mathrm{d}\hat{\tau} \approx 0$ for small times and (5.17) can therefore be approximated by the reduced model

$$\begin{aligned} \frac{\mathrm{d}U_s}{\mathrm{d}\hat{\tau}} &= 0, & U_s(0) &= 1, \\ \frac{\mathrm{d}U_c}{\mathrm{d}\hat{\tau}} - U_s + (U_s + \kappa)U_c &= 0, & U_c(0) &= 0. \end{aligned} \tag{5.18}$$

which has solution $\mathbf{U}(\hat{\tau}) = (U_s(\hat{\tau}), U_c(\hat{\tau}))^{\mathsf{T}}$ with $U_s(\hat{\tau})$ and $U_c(\hat{\tau})$ given by

$$U_s(\hat{\tau}) = 1, \quad U_c(\hat{\tau}) = \frac{1}{1+\kappa}(1 - \exp(-(1+\kappa)\hat{\tau})). \tag{5.19}$$

We refer to Figure 5.2 for a comparison of the complex concentrations $U_c(\tau)$ from (5.15) and $u_c(t)$ which is part of the solution (approximation on a very fine mesh) to (5.14) with parameters $\epsilon = 10^{-3}$, $\kappa = 11$, and $\lambda = 10$.

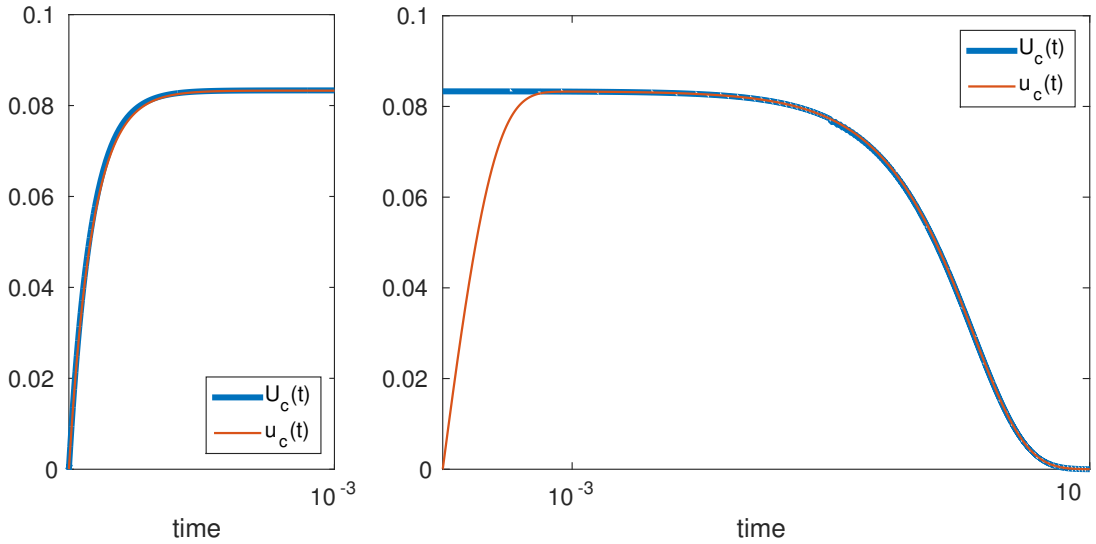



Figure 5.2: Complex concentration $u_c(t)$ of full model (5.14) and $U_c(t)$ of the reduced model: on the left the short time behaviour (solution (5.19)), on the right the long time behaviour (quasi-steady state approximation (5.16)).

## Automatic Model Reduction

We would like to get reduced models similar to the ones obtained through asymptotic model reduction with our combined algorithm, starting with the dimensional model (5.12) without the need for non-dimensionalisation. The model reduction algorithm, and therefore also the mesh refinement algorithm, relies on a linear functional of the solution. This in turn influences the outcome, a reduced model, whence it is important to choose the 'right' functional[1].

First we would like to recover the quasi-steady state approximation which is the (asymptotically) reduced model (5.15). As this does neglect the initial transient, it seems appropriate to choose $J$ as the average of the complex concentration. We therefore define the linear functional of the solution $\mathbf{u}(t) = (s(t), c(t))^{\mathsf{T}}$ as

$$J(\mathbf{u}) = \big\langle \mathbf{u}\,, \mathbf{g} \big\rangle := \frac{1}{T} \int_0^T c(t)\ \mathrm{d}t, \ \text{ whence } \ \mathbf{g} = (0, 1/T)^{\mathsf{T}}.$$

We apply the combined algorithm described in Section 5.1.2 to the primal problem (5.12) with time scale $0 \leq t \leq T = 10$ sec, reaction rates and initial values as follows:

| const. | $k_{+1}$ | $k_{-1}$ | $k_{+2}$ | $s_0$ | $c_0$ | $e_0$ | $p_0$ |
|---|---|---|---|---|---|---|---|
| unit | $\mathrm{Mol}^{-1}\mathrm{sec}^{-1}$ | $\mathrm{sec}^{-1}$ | $\mathrm{sec}^{-1}$ | Mol | Mol | Mol | Mol |
| | $10^5$ | $10^3$ | $10^4$ | $10^{-2}$ | 0 | $10^{-5}$ | 0 |

We point out that this choice of initial values and reaction rates yields the non-dimensional parameter values $\epsilon = 10^{-3}$, $\kappa = 11$, and $\lambda = 10$ which we used for the non-dimensional models in the preceding section. This should facilitate the comparison of the asymptotic with the automatic model reduction.

[1] This wording seems to contradict the original aim of generating a reduced model with guaranteed accuracy based on a user-defined linear functional of the solution. But we can go 'both ways' so to speak, which boils down to why and how we choose a linear functional of the solution. So far we argued that a linear functional should be chosen to capture a user-defined outcome of interest. This is then used to generate a reduced model and a solution that approximates the full model and its solution satisfactorily. Conversely we could choose a linear functional in such a way that the generated reduced model and solution capture the 'user-defined essence' of the full model, like for instance a reduced model that reflects an asymptotic model. It is in this sense that we speak of choosing the 'right' or 'correct' functional of the solution.

As the complex concentration $c$ is of order $10^{-6}$ we want the overall induced error $|J(\mathbf{u}) - J(\mathbf{U}^\mathsf{h})|$ to be much smaller and put the weight on the induced error on model reduction. We choose $\vartheta = 1/10$ and $\texttt{TOL} = 10^{-10}$ and run the combined algorithm. It turns out that the refined mesh has only 100 nodes when starting with an initial mesh for the mesh refinement algorithm with just 11 equally spaced nodes. The model reduction algorithm with initial reduced model $\mathbf{a}(t, \mathbf{U}) = \mathbf{0}$ with 'rule 2' (see 4.1.4) yields a reduced model which is the dimensional analogue of (5.15) for $t \in [0\,, 8.25]$,

| for $0 < t \leq 8.25$ | for $8.25 < t \leq 10$ |
|---|---|
| $\dfrac{\mathrm{d}S}{\mathrm{d}t} + k_{+1}e_0S - (k_{+1}S + k_{-1})C = 0$ | $\dfrac{\mathrm{d}S}{\mathrm{d}t} = 0$ |
| $-k_{+1}e_0S + (k_{+1}S + k_{-1} + k_{+2})C = 0$ | $-k_{+1}e_0S + (k_{+1}S + k_{-1} + k_{+2})C = 0$ |

where $S(0) = s_0$. The long time behaviour corresponds to the quasi-steady state assumption $\mathrm{d}C/\mathrm{d}t \approx 0$ which is valid for times greater than $10^{-3}$ seconds. This is similar to the (non-dimensional) asymptotic model (5.15), with solution (implicit for the substrate):

$$S(t) + \frac{k_{-1} + k_{+2}}{k_{+1}} \log(S) = s_0 - k_{+2}e_0t + \frac{k_{-1} + k_{+2}}{k_{+1}} \log(s_0), \quad C(t) = \frac{e_0S(t)}{S(t) + \frac{k_{-1}+k_{+2}}{k_{+1}}}.$$

For $t > 8.25$ seconds our reduced model has the substrate concentration at the constant value of $S(8.25)$ and accordingly also the complex concentration at the constant value of $e_0S(8.25)/(S(8.25) + \frac{k_{-1}+k_{+2}}{k_{+1}})$. This change in the reduced model on that last subinterval $[8.25\,, 10]$ does not really make a difference though since the complex concentration is again nearly zero there anyway, see Figure 5.3.

The estimates for the induced errors are

$$\begin{aligned}
|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| &\approx 7.5 \times 10^{-12} \leq \vartheta \times \texttt{TOL} = 10^{-11}, \text{ and} \\
|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| &\approx 7.2 \times 10^{-11} \leq (1 - \vartheta) \times \texttt{TOL} = 9 \times 10^{-11}, \text{ and thus} \\
|J(\mathbf{u}) - J(\mathbf{U}^\mathsf{h})| &\approx 8 \times 10^{-11} \leq \texttt{TOL} = 10^{-10}.
\end{aligned}$$

The choice of linear functional to get the reduced model we hoped for was therefore the 'correct' one in as far as we could generate a reduced model with a solution for the complex concentration that corresponds to the (non-dimensional) quasi-steady state solution.

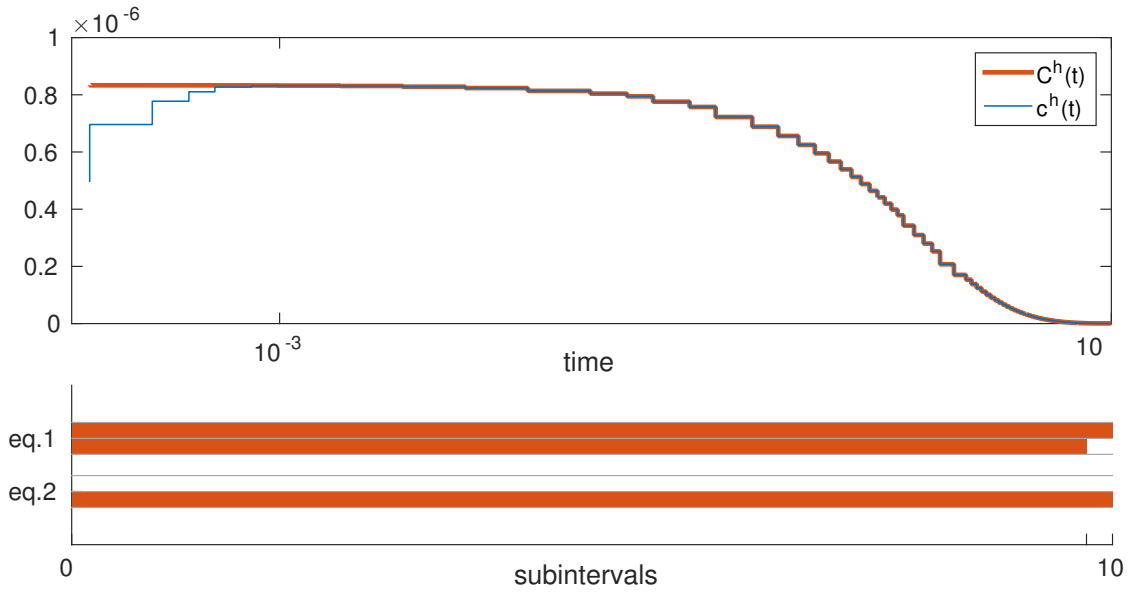



Figure 5.3: Top: approximations $C^h$ and $c^h$ of the complex concentrations of full and reduced models. Bottom: reduced model with linear functional being the average of the complex concentration over $[0\,,10]$. This corresponds to the quasi-steady state model (5.15).

What we could not retrieve with that linear functional is the initial fast transient, which means the 'boundary layer' near 0 of size $\epsilon = e_0/s_0 = 10^{-3}$. Would we want this then we not only have to choose a different linear functional, but we also need a mesh with elements at the left boundary each of which has size smaller than the boundary layer width. We could achieve this by choosing $J$ such that $|J(\mathbf{u}) - J(\mathbf{U}^h)|$ approximates the $\mathcal{L}_2$ error $\left\|c - C^h\right\|$ similarly to what we have used for the nonlinear example in Section 4.2.2 of the preceding chapter. So for the dual problem needed for model reduction we define the vector $\mathbf{g} = (0, g_2)^\mathsf{T}$ with $g_2 = (c^h - C^h)/\left\|c^h - C^h\right\|$ as this can be computed since both dG(0) approximations to the primal and reduced problems are known. For mesh refinement we resort to the approximation to the error $c - c^h$ as introduced for the nonlinear example in Section 5.2.1 above, with $\mathbf{g}$'s second component defined as $(\tilde{c}^h - c^h)/\left\|\tilde{c}^h - c^h\right\|$.

Again we want the overall induced error $|J(\mathbf{u}) - J(\mathbf{U}^\mathsf{h})|$ to be of order $10^{-10}$, setting $\texttt{TOL} = 4 \times 10^{-10}$ and giving more weight to model reduction, choosing $\vartheta = 1/4$. The mesh refinement algorithm generates a mesh with 374 nodes where the smallest elements $I_m$ in $[0\,, 10^{-3}]$ are of order smaller than $10^{-3}$ as is needed to capture the fast transient, and the element size $h_m$ then increases with $m$ since a 'high resolution' is only needed close to 0. The initial reduced model was again $\mathbf{a}(t, \mathbf{U}) = \mathbf{0}$ and 'rule 2' (see Section 4.1.4) was chosen for updating the sets $R_n$ and $S_n$. After applying the combined algorithm we found the estimates for the induced errors to be

$$
\begin{aligned}
|J(\mathbf{u}) - J(\mathbf{u}^\mathsf{h})| &\approx 1.8 \times 10^{-11} \leq \vartheta \times \texttt{TOL} = 10^{-10}, \text{ and} \\
|J(\mathbf{u}^\mathsf{h}) - J(\mathbf{U}^\mathsf{h})| &\approx 2.5 \times 10^{-10} \leq (1 - \vartheta) \times \texttt{TOL} = 3 \times 10^{-10}, \text{ and thus} \\
|J(\mathbf{u}) - J(\mathbf{U}^\mathsf{h})| &\approx 2.7 \times 10^{-10} \leq \texttt{TOL} = 4 \times 10^{-10}.
\end{aligned}
$$

We did indeed recover the boundary layer which is of order $\epsilon = e_0/s_0 = 10^{-3}$ (model reduction algorithm gives an actual value of $5 \times 10^{-4}$) as can be seen in Figure 5.4. The reduced model is actually very simple, splitting into two parts:

| for $0 < t \leq 5 \times 10^{-4}$ | for $5 \times 10^{-4} < t \leq 10$ |
|---|---|
| $\dfrac{\mathrm{d}S}{\mathrm{d}t} = 0$ | $\dfrac{\mathrm{d}S}{\mathrm{d}t} + k_{+1}e_0S - (k_{+1}S + k_{-1})C = 0$ |
| $\dfrac{\mathrm{d}C}{\mathrm{d}t} - k_{+1}e_0S + (k_{+1}S + k_{-1} + k_{+2})C = 0$ | $-k_{+1}e_0S + (k_{+1}S + k_{-1} + k_{+2})C = 0$ |

with $S(0) = s_0$ and $C(0) = 0$. The short time behaviour for $0 < t \leq 5 \times 10^{-4}$ has a closed form solution for both the substrate $S$ and the complex $C$. The substrate concentration stays constant throughout the boundary layer, whereas the complex concentration grows rapidly:

$$
S = s_0, \quad C(t) = \frac{k_{+1}e_0s_0}{k_{+1}s_0 + k_{-1} + k_{+2}}(1 - \exp(-(k_{+1}s_0 + k_{-1} + k_{+2})t).
$$

This is the dimensional analogue to the fast transient solution (5.19), whereas the long

time behaviour corresponds to the quasi-steady state approximation (5.16) which is valid for times greater than $10^{-3}$ seconds.

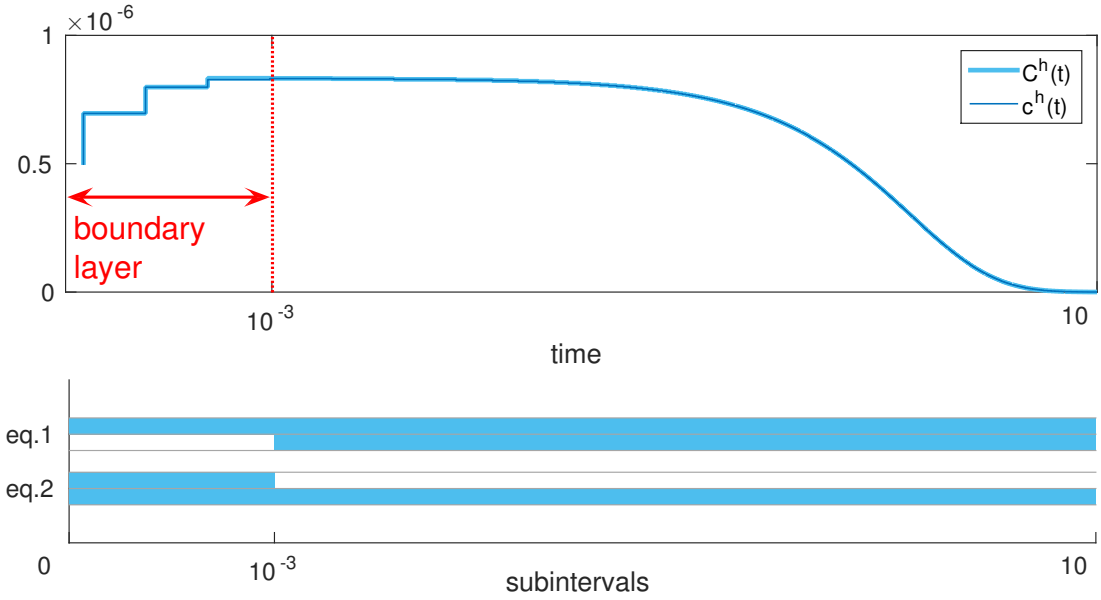



Figure 5.4: Top: approximations $C^h$ and $c^h$ of the complex concentrations of full and reduced models showing the boundary layer. Bottom: graphical representation of the reduced model with functional approximating the $\mathcal{L}_2$ error $\left\|c^h - C^h\right\|$.

We saw with this example that model reduction did indeed produce insights and model simplification which could otherwise only be gained through mathematical techniques such as non-dimensionalisation and asymptotic analysis. We would also like to stress again that the model reduction that we have seen so far could only be achieved with the extended algorithm that allows also for derivative terms to be dropped from some or all equations in a system. The combination with mesh refinement does also contribute to model reduction by reducing the computational effort. Since the accuracy of the numerical solutions are measured with a linear functional of the solution, this entails that meshes are only refined 'where necessary' which in turn leads to far fewer elements that discretise the computational domain.

### 5.2.3 Neuronal Spike Dynamics

Neurons transmit signals through short voltage pulses. Signals are 'collected' by a neuron's dendrites, 'processed' by the soma and 'transmitted' by the axon. The signal takes the form of an action potential that travels along the axon. Hodgkin and Huxley gathered experimental data using nerve axons of giant squid and were able to formulate a mathematical model which reflects experimental data fairly closely. The model we reproduce here is taken from their original paper [20] with constants as given there, rearranged to our format (5.2):

Find $\mathbf{u}(t) = (u_1(t), u_2(t), u_3(t), u_4(t))^{\mathsf{T}} \in \boldsymbol{\mathcal{C}}^1(0,T)$ satisfying $\mathbf{u}(0) = \mathbf{u}_0$ such that

$$\begin{aligned}
C\frac{\mathrm{d}u_1}{\mathrm{d}t} + \; & \bar{g}_K u_2^4(u_1 - V_K) + \bar{g}_{Na} u_3^3 u_4(u_1 - V_{Na}) + \bar{g}_l(u_1 - V_l) &&= I(t), \\
\frac{\mathrm{d}u_2}{\mathrm{d}t} + \; & \frac{(10-u_1)/100}{\exp((10-u_1)/10) - 1}(u_2 - 1) + \frac{\exp(-u_1/80)}{8} u_2 &&= 0, \\
\frac{\mathrm{d}u_3}{\mathrm{d}t} + \; & \frac{(25-u_1)/10}{\exp((25-u_1)/10) - 1}(u_3 - 1) + 4\exp(-u_1/18) u_3 &&= 0, \\
\frac{\mathrm{d}u_4}{\mathrm{d}t} + \; & \frac{7}{100}\exp(-u_1/20)(u_4 - 1) + \frac{1}{\exp((30-u_1)/10) + 1} u_4 &&= 0,
\end{aligned} \tag{5.20}$$

for $0 < t \leq T$, where time is measured in milliseconds.

Hodgkin and Huxley denoted components of $\mathbf{u}$ by $V$, $n$, $m$, $h$ where $V (= u_1)$ is the transmembrane potential and the variables determine the potassium activation ($n = u_2$), sodium activation ($m = u_3$) and inactivation ($h = u_4$). The potassium conductance is $g_K = \bar{g}_K u_2^4$ and the sodium conductance $g_{Na} = \bar{g}_K u_2^4$, and the ionic current $g_{\text{ion}}(V - V_{\text{ion}})$. Thus the first equation expresses the total current $I(t)$ as the sum of the ionic currents passing through the axon membrane and the time variation in transmembrane potential (multiplied by capacitance $C$). Hodgkin and Huxley mainly considered potassium and sodium ions and any other ions passing through the membrane are accounted for by a 'leakage' current, hence the subscript $l$ for the third ionic current.

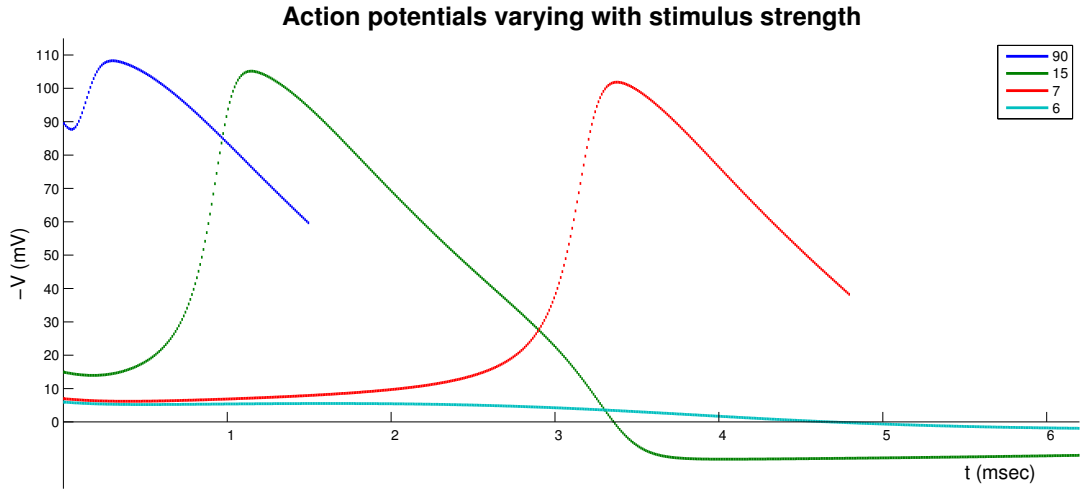



Figure 5.5: This figure corresponds to Fig. 12 in [20] and shows the membrane action potentials $u_1(t)$ for four different initial stimuli. Thus the initial conditions are $\mathbf{u}_0 = (V_0, 0.3177, 0.0529, 0.5961)^\mathsf{T}$ with $V_0 \in \{90, 15, 7, 6\}$.

They chose the values for the constants (capacitance $C$, constant equilibrium potentials $V_{\text{ion}}$ and conductances $\bar{g}_{\text{ion}}$) around the mean experimental values (Table 3. in [20]), which are

| const. | $C$ | $V_K$ | $V_{Na}$ | $V_l$ | $\bar{g}_K$ | $\bar{g}_{Na}$ | $\bar{g}_l$ |
|---|---|---|---|---|---|---|---|
| unit | $\mu$F/cm$^2$ | mV | mV | mV | m.mho/cm$^2$ | m.mho/cm$^2$ | m.mho/cm$^2$ |
| | 1.0 | $-12$ | 115 | 10.613 | 36 | 120 | 0.3 |

Although the Hodgkin–Huxley model is a realistic physiological model it is nevertheless quite complex. Since its inception other models of neuron spiking have been developed, the most widespread is probably the FitzHugh–Nagumo model which captures the qualitative features with only two equations, see Figure 5.6. FitzHugh studied the Hodgkin–Huxley model but the 'model reduction' that he achieved cannot be reproduced by our model reduction method since it is a phenomenological model. He started out with the van der Pol equations and furthermore the goal was to find a less complex model that captures the behaviour qualitatively rather than quantitatively. We can still apply our model reduction algorithm to the original Hodgkin–Huxley model which may elicit an insight into the activation or inactivation of the ion channels for instance.

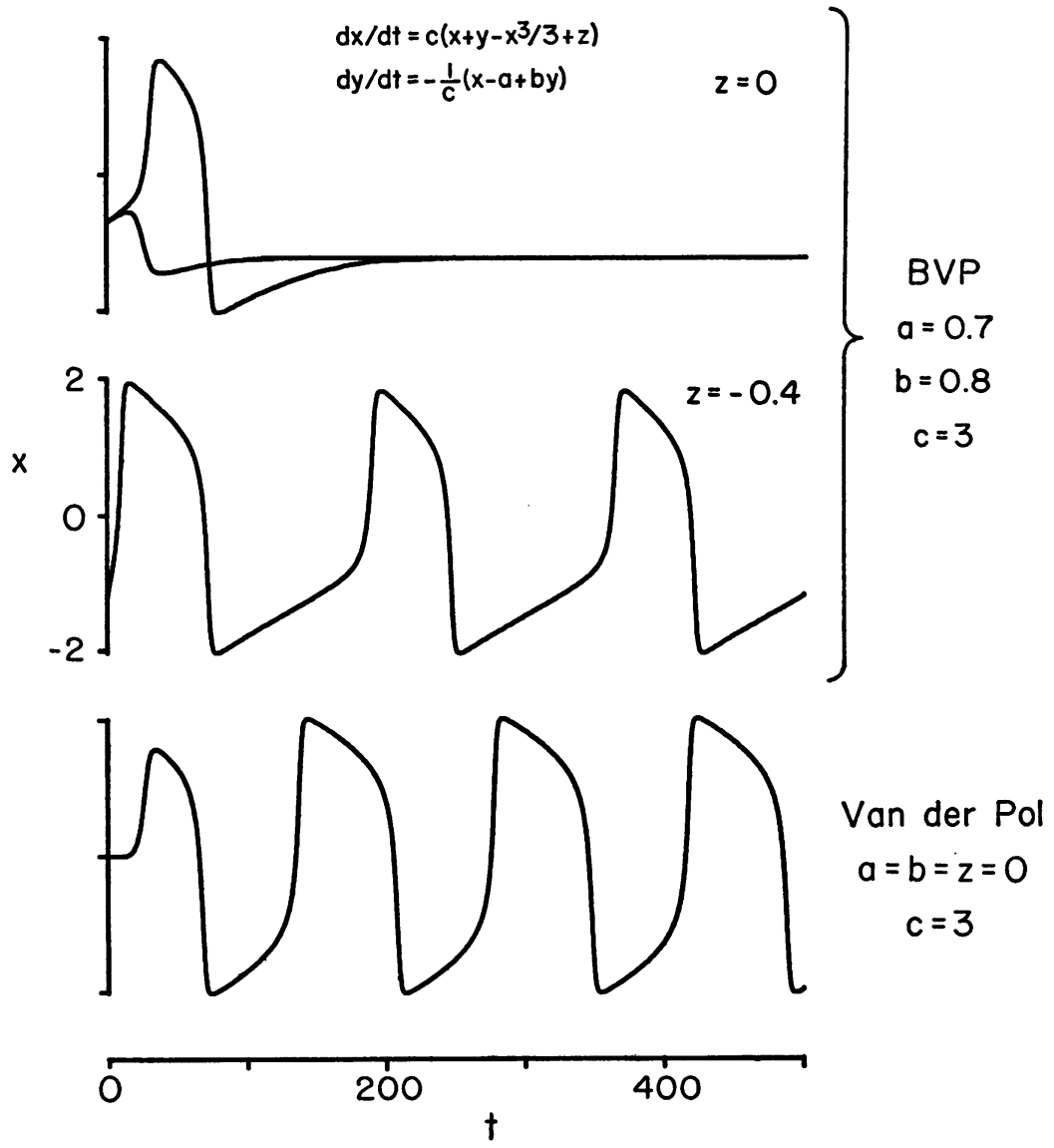



Figure 5.6: This is Fig. 2 taken from [44] bearing the legend: Single impulses and trains from BVP model and special van der Pol case.

## Ion Channels Model Reduction

We can deduce from Figure 5.5 that the action potential spikes at around 1.2msec for an original stimulus of $V_0 = -15\text{mV}$. We are interested in the contributions of the various ion channels to this spiking and so we choose the linear functional of the solution to be the value of $u_1(t)$ at $t = 1.2$, whence we define

$$J(\mathbf{u}) = \langle \mathbf{u}\,, \mathbf{g} \rangle = \int_0^{1.2} u_1(t)\delta(t-1.2)\,\mathrm{d}t, \text{ so } \mathbf{g}(t) = (\delta(t-1.2), 0, 0, 0)^\mathsf{T}.$$

As we want to discover which of the ion channels influence the spiking behaviour, we will use an initialisation that reflects this. The initial reduced model has $R_1 = [0\,, 1.2] = S_1$, and for $n = 2, 3, 4$ $R_n = \varnothing$ and $S_n = [0\,, 1.2]$. This means that the first equation governing the action potential will always be present in the reduced model, and that any model reduction just takes place for the ionic currents.

Since the estimated value of the action potential at $t = 1.2$msec is about $-105$mV we can choose the overall tolerance generously as $\texttt{TOL} = 3/5$ which gives an estimated relative error of about $5 \times 10^{-3}$. We also set $\vartheta = 5/6$ which gives mesh refinement five times more weight than model reduction.

The combined algorithm generates dG(0) approximations to the solutions of primal and reduced problems on a mesh with 95 nodes (with element sizes that vary from $\min_m h_m = 1/400$ to $\max_m h_m = 1/25$) and a reduced model, depicted in Figure 5.7.

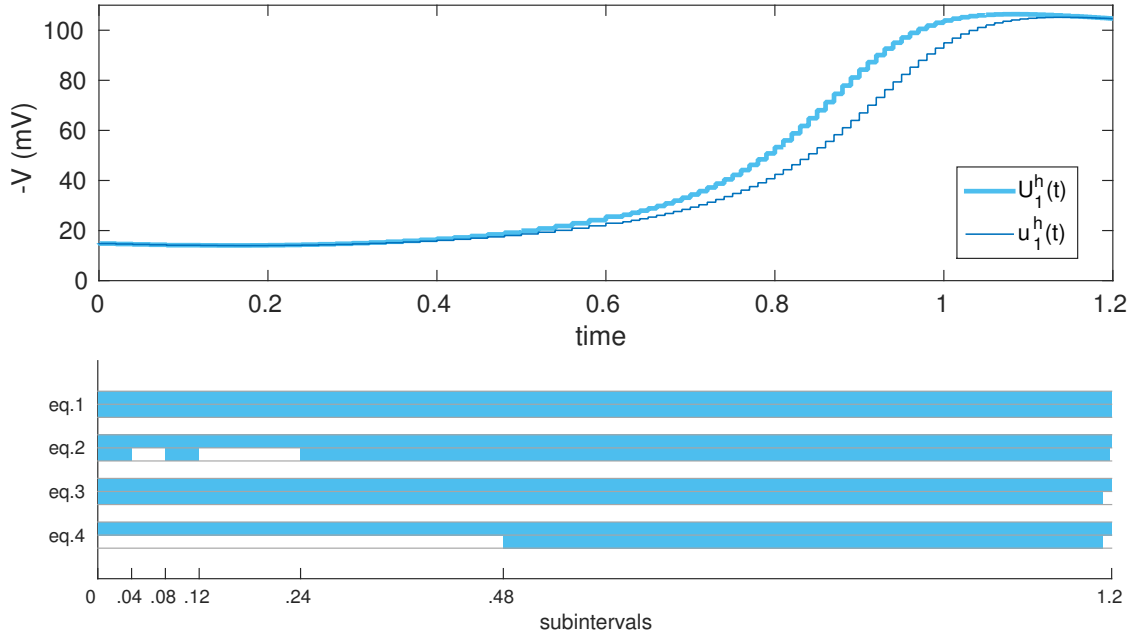



Figure 5.7: Top: Spiking of the action potential, where the reduced model component $U_1^h(t)$ is the light blue curve and the full model component $u_1^h(t)$ is dark blue. Bottom: graphical representation of the reduced model, showing the spike contribution of the ion channels.

Estimates for the induced errors after completion of the combined algorithm are

$$
\begin{aligned}
|J(\mathbf{u}) - J(\mathbf{u}^{\mathsf{h}})| &\approx 0.4831 \leq \vartheta \times \texttt{TOL} = 1/2, \text{ and} \\
|J(\mathbf{u}^{\mathsf{h}}) - J(\mathbf{U}^{\mathsf{h}})| &\approx 0.0632 \leq (1-\vartheta) \times \texttt{TOL} = 1/10, \text{ and thus} \\
|J(\mathbf{u}) - J(\mathbf{U}^{\mathsf{h}})| &\approx 0.5463 \leq \texttt{TOL} = 3/5.
\end{aligned}
$$

We deduce from the graphical representation of the reduced model in Figure 5.7 that there are roughly three time intervals that are of interest, the first fifth of $[0\,,1.2]$, the second fifth and the remaining three fifths. For $t \in [0\,,0.24]$ potassium activation ($U_2(t)$) and sodium inactivation ($U_4(t)$) do not take place meaning that these ionic currents stay at the initial constant value, whereas sodium is activated ($U_3(t)$) throughout. After that for $t \in [0.24\,,0.48]$ also potassium activation kicks in and finally from $t \geq 0.48$ also sodium inactivation. It would be interesting to see whether these deductions are actual 'facts' in a real physiological setting but this is not possible. We also do think that the Hodgkin–Huxley model is not really suitable for this type of model reduction as it is already the simplest physiologically realistic model. Any further reduction can probably only be achieved by employing a model order reduction such as we described in the review chapter in Section 3.1.2.

# CHAPTER 6

# Conclusion

The model reduction method which forms the main part of this thesis has been shown to produce results that fulfil the aim set out in the introduction. The goal was to develop an automatic method that, when applied to a mathematical model in the form of an initial value system of ODEs, generates a reduced (simpler) model that captures key features of the model. One can gain insight in this way which otherwise often requires significant mathematical expertise and might not even be tractable if the system is complex.

As already mentioned in Chapter 4, the model reduction algorithm exhibits sensitivities to the following choices and which change the outcome of the model reduction algorithm, even more so if in combination with the mesh refinement algorithm:

- Choice of initial reduced model.
- Choice of the rule for updating the sets that define the reduced model.
- Choice of linear functional.
- Choice of mesh for FE approximations.
- Choice of size of set of largest contributions to the induced errors.

We have not been able to resolve this sentitivity issue in this thesis as our focus was on the development of an appropriate algorithm for model reduction and its implementation. Of special interest for us is the correlation of mesh size and 'simplicity' of a reduced model or 'better performance' of the model reduction algorithm. We already introduced these qualitative terms in Section 4.2.1 where we conjectured that it is more appropriate to apply the model reduction algorithm based on a 'coarse' (not much refined) rather than a 'fine' (very refined) mesh respectively FE approximations. For this reason a weighting factor $\vartheta$ was introduced in Chapter 5 to allow allocation

of the induced error to either mesh refinement or to model reduction. However the overall tolerance TOL, for the error induced in the linear functional when the true solution is replaced by the reduced model solution, does not really take our conjecture into account. One way to resolve this would be to 'adaptively' reduce a model without setting an overall tolerance TOL for $|J(\mathbf{u}) - J(\mathbf{U}^h)|$ beforehand and starting with a given (non-refined) initial mesh:

1. Compute FE approximations to solutions of primal and associated dual problem to get an estimate for the induced error $|J(\mathbf{u}) - J(\mathbf{u}^h)|$.
2. Apply model reduction algorithm which generates a reduced model and a FE approximation of the solution thereof.
3. Stop if satisfied with reduced model and with the induced error $|J(\mathbf{u}^h) - J(\mathbf{U}^h)|$.
4. If not, apply mesh refinement algorithm once and repeat from 1.

Also this has drawbacks, one of which is that it is not automatic, meaning that at least Step 3. needs user judgement and supervision, but it might well be worth exploring. Other avenues to pursue are looking at different ways of initialisation of the reduced model and rules for updating the sets that determine inclusion of the different parts of an equation in the system of ODEs which we referred to in Section 4.1 (see also footnote there). We only discussed initial value systems of ODEs in this thesis which already gave rise to several issues. Nevertheless the method could easily be extended to also include boundary value problems and then possibly extended for PDEs.

We conclude by remarking that the model reduction method (combined with mesh refinement) worked remarkably well for an enzyme kinetics model as we have seen in Section 5.2.2. The combined algorithm automatically generated reduced models that captured short term (fast transient) and long term (quasi-steady state) behaviour. Furthermore, these reduced models corresponded to a large extent with reduced models gained through asymptotic analysis. This gives reason to believe that our model reduction technique can work effectively for systems of ODEs that model quantities that act on different time scales.

# Bibliography


[1] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson. Introduction to adaptive methods for differential equations. *Acta Numerica*, 4:105–158, 1995.

[2] S. Repin. Lectures on *a Posteriori* Estimates. `http://www.math.uni-sb.de/ag/fuchs/Menupkte/SB01.pdf`. DAAD program 2008.

[3] M. Ainsworth and J. T. Oden. *A posteriori* error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(1–2):1–88, 1997.

[4] J. P. Whiteley. Model reduction using *a posteriori* analysis. *Mathematical Biosciences*, 225:44–52, 2010.

[5] K. Erikkson, D. Estep, C. Johnson, and P. Hansbo. *Computational Differential Equations*. Cambridge University Press, 1996.

[6] G. Strang. Finite Element Method. `http://serious-science.org/finite-element-method-36`, 2013. Serious Science Project.

[7] V. Thomée. From finite differences to finite elements: A short history of numerical analysis of partial differential equations. *Journal of Computational and Applied Mathematics*, 128:1–54, 2001.

[8] Y. Yuan. In memory of professor Feng Kang. `http://lsec.cc.ac.cn/~yyx/fk.html`, 1993. Chinese Academy of Sciences.

[9] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, $2^{nd}$ edition, 2008.

[10] M. B. Giles and E. Süli. Adjoint methods for pdes: *a posteriori* error analysis and postprocessing by duality. *Acta Numerica*, 11:145–236, 2002.

[11] S. C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer–Verlag New York, 1994.

[12] I. Babuška and B. Q. Guo. The $h$, $p$ and $h$-$p$ version of the finite element method; basis theory and applications. *Advances in Engineering Software*, 15:159–174, 1992.

[13] A. C. Fowler. *Mathematical models in the applied sciences*. Cambridge University Press, 1997.

[14] S. Schnell, R. Grima, and P. K. Maini. Multiscale modeling in biology: New insights into cancer illustrate how mathematical tools are enhancing the understanding of life from the smallest scale to the grandest. *American Scientist*, 95(2):134–142, 2007.

[15] L. S. Kimpton, J. P. Whiteley, S. L. Waters, J. R. King, and J. M. Oliver. Multiple travelling-wave solutions in a minimal model for cell motility. *Mathematical Medicine and Biology*, 30(3), 2013.

[16] G. Mirams, J. King, and H. Byrne. A multiple timescale analysis of a mathematical model of the Wnt/Beta-catenin signalling pathway. *Journal of Mathematical Biology*, 60(1):131–160, 2010.

[17] L. J. Cummings and S. L. Waters. Tissue growth in a rotating bioreactor. Part II: fluid flow and nutrient transport problems. *Mathematical Medicine and Biology*, 24:169–208, 2007.

[18] H. M. Byrne. A weakly nonlinear analysis of a model of avascular solid tumour growth. *Journal of Mathematical Biology*, 39:59–89, 1999.

[19] R. Suckley and V. N. Biktashev. Comparison of asymptotics of heart and nerve excitability. *Physical Review Letters E*, 68:011902, 2003.

[20] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[21] J. L. Hindmarsh and R. M. Rose. A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations. *Proceedings of the Royal Society of London, Series B, Biological Sciences*, 221(1222):87–102, 1984.

[22] W. Gerstner and W. M. Kistler. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[23] R. Skelton, M. Oliveira, and J. Han. Systems modelling and model reduction. In V. Varadan, editor, *Hanbook of smart systems and materials*. Institute of Physics, Bristol, 2004.

[24] W. A. H. Schilders, H. A. van der Vorst, and J. Rommes. *Model Order Reduction*. Springer, 2008.

[25] P. A. Iglesias and B. P. Ingalls, editors. *Control Theory and Systems Biology*. MIT press, 2009.

[26] P. Wellstead, D. Bullinger, E. Kalamatianos, O. Mason, and M. Verwoerd. The Role of Control and System Theory in Systems Biology. *Annual Reviews in Control*, 32:33–47, 2008.

[27] M. Cloutier and P. Wellstead. The Control Systems Structures of Energy Metabolism. *Interface Journal of the Royal Society*, 7:651–665, 2010.

[28] D. Amsallem, M. Zahr, and C. Farhat. Nonlinear Model Order Reduction Based on Local Reduced-Order Bases. *IJNME*, 92:891–916, 2012.

[29] D. Wirtz, D. C. Sorensen, and B. Haasdonk. A Posteriori Error Estimation for DEIM Reduced Nonlinear Dynamical Systems. *SIAM Journal on Scientific Computing*, 36(2):311–338, 2014.

[30] I. Maier and B. Haasdonk. A Dirichlet–Neumann reduced basis method for homogeneous domain decomposition problems. *Applied Numerical Mathematics*, 78:31–48, 2014.

[31] A. Quarteroni and G. Rozza, editors. *Reduced Order Methods for Modeling and Computational Reduction*. Springer, 2014.

[32] U. Maas and S. B. Pope. Simplifying Chemical Kinetics: Intrinsic Low-Dimensional Manifolds in Composition Space. *Combustion and Flame*, 88:239–264, 1992.

[33] H. K. Khalil. *Nonlinear Systems*. Prentice–Hall, 1996.

[34] T. B. Kepler, L. F. Abbott, and E. Marder. Reduction of conductance-based neuron models. *Biological Cybernetics*, 66:381–387, 1992.

[35] C. Y. Cha, Y. Himeno, T. Shimayoshi, A. Amano, and A. Noma. A Novel Method to Quantify Contribution of Channels and Transporters to Membrane Potential Dynamics. *Biophysical Journal*, 97(12):3086–3094, 2009.

[36] D. Estep and D. Neckels. Fast and reliable methods for determining the evolution of uncertain parameters in differential equations. *Journal of Computational Physics*, 213(2):530–556, 2006.

[37] D. Estep and D. Neckels. Fast methods for determining the evolution of uncertain parameters in reaction-diffusion equations. *Computer Methods in Applied Mechanics and Engineering*, 196(37–40):3967–3979, 2007.

[38] E. A. Sobie. Parameter sensitivity analysis in electrophysiological models using multivariable regression. *Biophysical Journal*, 96(4):1264–1274, 2009.

[39] R. Clewey, H. G. Rotstein, and N. Kopell. A computational tool for the reduction of nonlinear ODE systems possessing multiple scales. *SIAM Multiscale Modeling and Simulation*, 4(3):732–759, 2005.

[40] S. A. Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. Wiley, 2013.

[41] L. Todorovski, S. Dzeroski, A. Srinivasan, J. P. Whiteley, and D. J. Gavaghan. Discovering the Structure of Partial Differential Equations from Example Behaviour. Proceedings of the International Conference on Machine Learning, 2000.

[42] P. J. Maybank and J. P. Whiteley. Automatic simplification of systems of reaction–diffusion equations by *a posteriori* analysis. *Mathematical Biosciences*, 248:146–157, 2014.

[43] J. D. Murray. *Mathematical Biology, I: An Introduction*. Springer, $3^{rd}$ edition, 2002.

[44] R. FitzHugh. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal*, 1:445–466, 1961.