UPPSALA
UNIVERSITET

Adjoint-based inverse modelling of an acoustic material

Andreas Eriksson, Erik Thermaenius

Project in Computational Science: Report

February 2022

PROJECT REPORT

Department of Information Technology

Abstract

Inverse modelling is the study of finding unknown causal parameters in models using measured data. This project aims to develop an algorithm for finding the material parameters in the second order acoustic wave model. Mathematically, these parameters correspond to the wave speed inside the medium. The model is described by a partial differential equation (PDE), which when solved, gives the magnitude of the acoustic waves. We formulate a minimisation problem with respect to the wave speed parameter. The cost function of this problem is defined by the norm of the residual to the PDE solution and measured data.

The inverse problem is solved using a gradient based optimisation algorithm. Both the Quasi-Newton BFGS method and the Steepest descent method are tested. Direct evaluation of the initial objective cost function is very expensive numerically, especially if the gradient of the objective isn't provided. A Lagrangian cost function is therefore introduced as the new objective. The free Lagrangian multiplier parameter is used to simplify the gradient expression by solving the adjoint problem to the original PDE.

The finite difference method, summation-by-parts simultaneous-approximation-terms (SBP-SAT), is used for the spatial discretisation of the PDE. The schemes tested here are second, fourth and sixth order accurate. To treat the boundaries we derive SAT with homogeneous Neumann conditions. For the temporal part of the PDE, both the fourth order explicit Runge-Kutta and central differences are used.

The resulting algorithm is tested using two different source functions, a standard Gaussian function and the Ricker-wavelet function. The measured data is simulated using a target parameter of three different profiles to test the algorithms behaviour. A solution satisfying the set tolerance is found in all cases and the result show some oscillations when the profile is discontinuous or if the mesh resolution is coarse. For a convergent solution below larger tolerances, it is crucial to use a dense mesh or an SBP operator with high order of accuracy. Comparing the Steepest descent method which is of linear convergence speed to the Quasi-Newton method of quadratic convergence speed, the latter method is as expected, the better alternative for a fast algorithm. The result show an effective method with low computational cost.

Contents

1 Introduction

In this section, we introduce the problem formulation of this project and its background. We showcase the difficulties we face with the problem and introduce the method we use to solve these. Finally, we state and specify our goal and aims of this project.

1.1 Background

Inverse modelling is the study of using measured data of some natural phenomenon to compute the causal parameters in a model of the observed phenomenon. The term *inverse* refers to the formulation of the problem, which is initiated from the observed end time and solved backwards to some initial state. The problem is formulated as an optimisation problem, where the error between the model output and measured data is minimised with respect to some parameter in the model. For models of transient physical processes, partial differential equations are used extensively and optimisation with those models are known as PDE-constrained optimisation problems.

1.2 Problem formulation

In this project, we consider the acoustic wave equation in the domain Ω with boundaries $\partial\Omega$ and the unknown parameter $a = K\rho^{-1}$, which is the bulk modulus K times the inverse of the density field of the domain $\rho(\boldsymbol{x})$. The optimisation problem reads

$$\underset{a}{\text{minimise}} \quad J(u,a) = \int_0^T \Big(u(\boldsymbol{x}_r,t,a) - u_{\text{data}}(t)\Big)^2 \,\mathrm{d}t \tag{1.1}$$

$$\text{subject to} \quad \frac{\partial^2 u}{\partial t^2} - \boldsymbol{\nabla}\cdot(a(\boldsymbol{x})\boldsymbol{\nabla}u) - g(t)\delta(\boldsymbol{x}-\boldsymbol{x}_s) = 0, \quad (\boldsymbol{x},t) \in \Omega\times(0,T], \tag{1.2}$$

$$\boldsymbol{n}\cdot\boldsymbol{\nabla}u = 0, \qquad (\boldsymbol{x},t) \in \partial\Omega\times(0,T], \tag{1.3}$$

$$u = 0, \qquad \boldsymbol{x}\in\Omega,\ t=0, \tag{1.4}$$

$$\frac{\partial u}{\partial t} = 0, \qquad \boldsymbol{x}\in\Omega,\ t=0, \tag{1.5}$$

where $u(\boldsymbol{x}_r)$ is the solution to the forward problem (1.2)-(1.5) at the receiver point $\boldsymbol{x}_r$, u_{data} is measured data at the receiver point and $g(t)$ a time-dependent source term. The Dirac delta function is written as $\delta(\boldsymbol{x}-\boldsymbol{x}_s)$, this sets the source at location $\boldsymbol{x}_s$. See Figure 1.1.

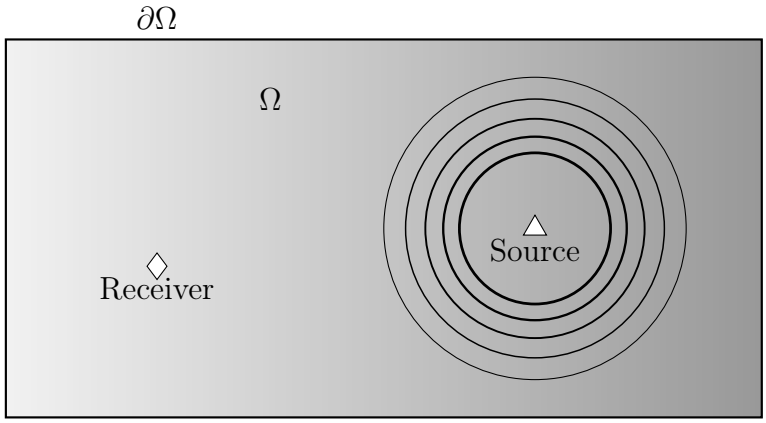


Figure 1.1: The computational domain Ω with boundaries $\partial\Omega$. The solution u is measured by a receiver at the position $\boldsymbol{x}_r$ marked by $\Diamond$. The receiver measures waves caused by a pulsating source at position $\boldsymbol{x}_s$ marked by $\triangle$. The density field of the domain $\rho(\boldsymbol{x})$ is spatially variant, indicated by the colour gradient.

We aim to solve this inverse problem numerically. Denoting the cost functional integral argument $u(\boldsymbol{x}_r, t, a) - u(t)_{\text{data}} = \mathcal{R}(u, a)$ as the residual, which is expressed as

$$\mathcal{R}(u,a) = \int_\Omega \left(u(\boldsymbol{x},t,a) - u(t)_{\text{data}}\right) \delta(\boldsymbol{x} - \boldsymbol{x}_r)\, \mathrm{d}\boldsymbol{x} = \left(u - u_{\text{data}},\, \delta(\boldsymbol{x} - \boldsymbol{x}_r)\right)_\Omega , \tag{1.6}$$

where $(\cdot,\, \cdot)_\Omega = \int_\Omega \, \mathrm{d}\boldsymbol{x}$ is the expression for the inner product between two arguments. We define the Lagrangian cost functional as

$$\mathcal{L}(u,a,\lambda) = J(u,a) + \int_0^T \left(\lambda,\, A(u)\right)_\Omega \, \mathrm{d}t, \tag{1.7}$$

where

$$A(u) = \frac{\partial^2 u}{\partial t^2} - \boldsymbol{\nabla} \cdot (a(\boldsymbol{x}) \boldsymbol{\nabla} u) - g(t)\delta(\boldsymbol{x} - \boldsymbol{x}_s). \tag{1.8}$$

The gradient of the cost functional is computed as

$$\frac{\mathrm{d}}{\mathrm{d}a} J(u,a) = \frac{\mathrm{d}}{\mathrm{d}a} \int_0^T \mathcal{R}^2 \, \mathrm{d}t = \int_0^T 2\left(u(\boldsymbol{x}_r, a) - u_{\text{data}}(t)\right) \frac{\partial}{\partial a} u(\boldsymbol{x}_r, a)\, \mathrm{d}t, \tag{1.9}$$

where

$$\frac{\partial}{\partial a} u(\boldsymbol{x}_r, a) = \left(\delta(\boldsymbol{x} - \boldsymbol{x}_r),\, \frac{\partial u}{\partial a}\right)_\Omega , \tag{1.10}$$

which gives $\frac{\mathrm{d}}{\mathrm{d}a}J(u,a)$ as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}a}J(u,a) &= \int_0^T \left(2\left(u(\boldsymbol{x}_r,a)-u_{\text{data}}(t)\right)\delta(\boldsymbol{x}-\boldsymbol{x}_r),\ \frac{\partial u}{\partial a}\right)_\Omega \mathrm{d}t \\ &= \int_0^T \left(2\,\mathcal{R}(u,a)\,\delta(\boldsymbol{x}-\boldsymbol{x}_r),\ \frac{\partial u}{\partial a}\right)_\Omega \mathrm{d}t.\end{aligned} \tag{1.11}$$

The gradient of the Lagrangian is similarly

$$\frac{\mathrm{d}}{\mathrm{d}a}\mathcal{L} = \int_0^T \left(\left(2\,\mathcal{R}(u,a)\,\delta(\boldsymbol{x}-\boldsymbol{x}_r),\ \frac{\partial u}{\partial a}\right)_\Omega + \frac{\mathrm{d}}{\mathrm{d}a}\left(\lambda,\ A(u)\right)_\Omega\right)\mathrm{d}t. \tag{1.12}$$

The main issue comes down to the term $\frac{\partial u}{\partial a}$, which is very expensive to compute numerically. This is because we must first discretise the PDE to get an approximate solution of u, then we must approximate the partial derivative of our approximation of u with respect to a. Because a is a function with an unique value everywhere in the domain, the partial derivative with respect to a in the discrete case is a component wise partial derivative in every grid point. Essentially, we must discretise the problem $m+1$ times, where m is the number of grid points, to make a single iteration with an optimisation algorithm.

We resolve this issue by rewriting the expression of the gradient such that the dependence of $\frac{\partial u}{\partial a}$ disappears. We do this using the second term of the Lagrangian cost functional gradient (1.12) and the adjoint solution of u.

1.3 Goal and aims

The goal of this project is to implement an algorithm for the inverse problem (1.1-1.5) in one spatial dimension, with the aims to measure the performance and accuracy of the algorithm. Given the complexity of the forward problem, we use the SBP-SAT finite difference method for the spatial discretisation of the forward problem. One of many advantages of this method is the definition and construction of the SBP-operator, which exactly mimics the properties of the continuous integration-by-parts. This gives that the discrete adjoint problem is mathematically consistent with the continuous adjoint, which was one of the primary motivations of choosing this method. In turn this enables to have an explicit expression for the gradient of the cost functional, which makes it suitable when paired with a gradient based optimisation algorithm.

2 Theory

In this section, we introduce the SBP-SAT method and apply it on the forward problem. We then give a short review of the stability of the forward problem, both the continuous case and the semi-discrete SBP-SAT approximation. We continue with the theory of adjoint problems and adjoint operators and finally optimisation, specifically gradient-based optimisation.

2.1 Definitions

We introduce the following definitions, used extensively in the following sections.

Definition 1. *The L^2 inner product between two scalar functions u and v is written as*

$$\int_\Omega u\, v \,\mathrm{d}x = (u,\, v)_\Omega \tag{2.1}$$

where Ω is the domain. For vector valued functions, $\boldsymbol{u}$ and $\boldsymbol{v}$, it is

$$\int_\Omega \boldsymbol{u}^{\mathrm{T}}\, \boldsymbol{v} \,\mathrm{d}x = (\boldsymbol{u},\, \boldsymbol{v})_\Omega \tag{2.2}$$

where $\boldsymbol{u}^{\mathrm{T}}$ is the transpose of $\boldsymbol{u}$.

Definition 2. *The energy norm of a function u is the inner product of u with itself as*

$$(u,\, u)_\Omega = \|u\|^2. \tag{2.3}$$

Definition 3. *The discrete inner product between two vector valued functions $\boldsymbol{u}$ and $\boldsymbol{v}$ is denoted by*

$$\boldsymbol{u}^{\mathrm{T}}\mathbf{H}\boldsymbol{v} = (\boldsymbol{u},\, \boldsymbol{v})_{\mathbf{H}}\,, \tag{2.4}$$

where the weighted matrix $\mathbf{H}$ must be a diagonal, symmetric and positive definite matrix such that $\mathbf{H} = \mathbf{H}^{\mathrm{T}}$.

Definition 4. *The discrete energy norm of a vector valued function $\boldsymbol{u}$ is the discrete inner product of itself as*

$$(\boldsymbol{u},\, \boldsymbol{u})_{\mathbf{H}} = \|\boldsymbol{u}\|^2_{\mathbf{H}}. \tag{2.5}$$

Definition 5. *An initial boundary value problem (IBVP) is said to be well-posed if an unique solution exists and*

$$\|u\|^2 \leq \exp(\eta t)\|f(\cdot)\| \tag{2.6}$$

holds for the homogeneous version of the IBVP, where η is some constant and $f(\cdot)$ is the initial data [3].

Definition 6. *Let $\mathbb{X}$ and $\mathbb{Y}$ be inner product spaces over the same field and let*

$$L : \mathbb{X} \mapsto \mathbb{Y} \tag{2.7}$$

be a continuous linear map. Then a map

$$L^* : \mathbb{Y} \mapsto \mathbb{X} \tag{2.8}$$

is called an adjoint of L if the following holds

$$(Lx,\, y)_\Omega = (x,\, L^* y)_\Omega\,, \quad \forall x \in \mathbb{X}, y \in \mathbb{Y}. \tag{2.9}$$

2.2 The SBP-SAT method

2.2.1 Discretisation of the forward problem

The forward problem (1.2-1.5) is discretised in space using summation-by-parts (SBP) operators with boundary treatment using simultaneous-approximation-terms (SAT). SBP operators are narrow stencil finite difference operators introduced by Kreiss and Scherer in 1974 [3], mimicking the properties of integration-by-parts in the discrete case. They have been proven to be efficient for long time computations and having high-order accuracy. Furthermore they have been especially popular for applications with hyperbolic equations, such as conservation laws and wave-dominated problems [6, 5]. Given the acoustic wave equation in one spatial dimension,

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right) + f(x,t), \tag{2.10}$$

the semi-discrete form using a second order SBP differential operator $\mathbf{D}_2$ is written as

$$\frac{\partial^2 \boldsymbol{v}}{\partial t^2} = \mathbf{D}_2^{(a)}\boldsymbol{v} + f(\boldsymbol{x},t), \tag{2.11}$$

where $\boldsymbol{v}$ is the discrete solution vector. The wave speed a is incorporated in the $\mathbf{D}_2$ operator since it is allowed to vary in space. The $\mathbf{D}_2$ operator is constructed as

$$\mathbf{D}_2^{(a)} = -\mathbf{H}^{-1}(\mathbf{M}^{(a)} - \bar{\mathbf{B}}\mathbf{S}), \tag{2.12}$$

where

$$\mathbf{M}^{(a)} = \mathbf{D}_1^{\mathrm{T}}\mathbf{H}\mathbf{A}^{(a)}\mathbf{D}_1 + \mathbf{R}^{(a)}. \tag{2.13}$$

All the matrices are found in [5] and the matrices $\mathbf{H}$, $\mathbf{M}^{(a)}$ and $\mathbf{R}^{(a)}$ are all symmetric, positive definite. The matrix $\mathbf{S}$ contains the column vectors $\boldsymbol{d}_b$ which are first order differential operators used for the boundary points. For treatment of the boundary values, SAT was

introduced by Carpenter et al. in 1993 [4], using this method the boundary terms are applied weakly into the scheme. For Neumann conditions, SAT are given by

$$\text{SAT} = \tau \mathbf{H}^{-1} \boldsymbol{e}_b (\boldsymbol{d}_b^{\mathrm{T}} \boldsymbol{v} - g_b(t)), \tag{2.14}$$

where the subscript b denotes the boundary indices. The vector e_b is a column vector with only non-zero elements at the boundary positions, i.e.

$$e_1 = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix}^{\mathrm{T}} \tag{2.15}$$

for left hand side boundaries and

$$e_m = \begin{bmatrix} 0 & \dots & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}} \tag{2.16}$$

for right hand side boundaries. The boundary data is here given by $g_b(t)$ and τ is a penalty parameter governing the stability. The full semi-discrete equation with the SBP-SAT method is then the sum of equation (2.11) and SAT for each boundary

$$\frac{\partial^2 \boldsymbol{v}}{\partial t^2} = \mathbf{D}_2^{(a)} \boldsymbol{v} + \text{SAT} + f(\boldsymbol{x}, t). \tag{2.17}$$

2.2.2 Stability of the forward problem

Lemma 1. *The well-posedness of the IBVP (1.2-1.5) is governed by the boundary conditions of the problem.*

Proof. The proof is done through an energy estimate of the problem. The source term and boundary data is set to zero such that the problem is homogeneous. The analysis is done in one spatial dimension. The equation is multiplied with the time derivative of u and integrated over the spatial domain. This give

$$\left(\frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial t^2} \right)_\Omega = \left(\frac{\partial u}{\partial t}, \frac{\partial}{\partial x} \left(a(x) \frac{\partial u}{\partial x} \right) \right)_\Omega . \tag{2.18}$$

Applying integration by parts in space for the right hand side and adding the equation with its transpose results in

$$\begin{aligned} \left(\frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial t^2} \right)_\Omega + \left(\frac{\partial^2 u}{\partial t^2}, \frac{\partial u}{\partial t} \right)_\Omega = &- \left(\frac{\partial}{\partial x} \left(\frac{\partial u}{\partial x} \right), a(x) \frac{\partial u}{\partial x} \right)_\Omega - \left(a(x) \frac{\partial u}{\partial x}, \frac{\partial}{\partial x} \left(\frac{\partial u}{\partial t} \right) \right)_\Omega \\ &+ 2 \, \frac{\partial u}{\partial t} \, a(x) \frac{\partial u}{\partial x} \Big|_{x_1}^{x_m} . \end{aligned} \tag{2.19}$$

This is then simplified with the chain rule and Definition 2 to

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\left\|\frac{\partial u}{\partial t}\right\|^2 + \left\|\frac{\partial u}{\partial x}\right\|_a^2\right) = 2\,\frac{\partial u}{\partial t}\,a(x)\frac{\partial u}{\partial x}\bigg|_{x_1}^{x_m}. \tag{2.20}$$

From Definition 5, the right hand side of equation (2.20) needed to be lesser or equal to zero for well posedness to be fulfilled. The stability is therefore dependent on the boundaries of the problem. Given the homogeneous Neumann condition (1.3), the forward problem is stable. □

Lemma 2. *The semi-discrete problem (2.17) is stable given the correct choice of the penalty parameters τ.*

Proof. Stability for the semi-discrete problem is proved using the same technique but in discrete form. Continuing in one spatial dimension, the semi-discrete problem with Neumann boundary conditions is

$$\frac{\partial^2 \boldsymbol{v}}{\partial t^2} = \mathbf{D}_2^{(a)}\boldsymbol{v} + \tau_1 \mathbf{H}^{-1} e_1 d_1^{\mathrm{T}}\boldsymbol{v} + \tau_m \mathbf{H}^{-1} e_m d_m^{\mathrm{T}}\boldsymbol{v} \tag{2.21}$$

where the subscripts 1 and m are the first and mth indices of the vectors, i.e. the left and right hand boundary indices. Setting the source term to zero, the discrete energy analysis is done by multiplying the problem with $\frac{\partial \boldsymbol{v}}{\partial t}^{\mathrm{T}}\mathbf{H}$ from the left. Then adding the transpose which, combined with equation (2.12) and Definition 3, 4 gives

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\left\|\frac{\partial \boldsymbol{v}}{\partial t}\right\|_{\mathbf{H}}^2 + \|\boldsymbol{v}\|_{\mathbf{M}^{(a)}}^2\right) =& 2\frac{\partial v_m}{\partial t}a_m \boldsymbol{d}_m\boldsymbol{v} - 2\frac{\partial v_1}{\partial t}a_1\boldsymbol{d}_1\boldsymbol{v} \\ &- \frac{\partial \boldsymbol{v}}{\partial t}\left(\mathbf{M}^{(a)} + \mathbf{M}^{(a)\mathrm{T}}\right)\boldsymbol{v} \\ &+ 2\tau_1\frac{\partial v_1}{\partial t}\boldsymbol{d}_1^{\mathrm{T}}\boldsymbol{v} + 2\tau_m\frac{\partial v_m}{\partial t}\boldsymbol{d}_m^{\mathrm{T}}\boldsymbol{v},\end{aligned} \tag{2.22}$$

where $\|\boldsymbol{v}\|_{\mathbf{M}^{(a)}}^2 = \boldsymbol{v}^{\mathrm{T}}\mathbf{M}^{(a)}\boldsymbol{v}$. Since $\mathbf{M}^{(a)} = \mathbf{M}^{(a)\mathrm{T}} \geq 0$, we rewrite the terms

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\left\|\frac{\partial \boldsymbol{v}}{\partial t}\right\|_{\mathbf{H}}^2 + \|\boldsymbol{v}\|_{\mathbf{M}^{(a)}}^2\right) = 2(\tau_m + a)\frac{\partial v_m}{\partial t}\boldsymbol{d}_m\boldsymbol{v} - 2(\tau_1 - a)\frac{\partial v_1}{\partial t}\boldsymbol{d}_1\boldsymbol{v}. \tag{2.23}$$

To fulfil well-posedness according to Definition 5, the penalty parameters needs to be set to $\tau_1 = a$ and $\tau_m = -a$ [5]. □

2.3 Adjoint operators and adjoint problems

Following Definition 6, we establish the basic properties of an adjoint operator. For a differential operator L and functions u, v, an adjoint fulfils the following relationship

$$(Lu,\, v)_\Omega = (u,\, L^* v)_\Omega\,, \quad \forall u \in V_1, v \in V_2. \tag{2.24}$$

The spaces V_1, V_2 are equipped with an inner product, such as Hilbert spaces, and defined over the same field which enables the mapping between the two spaces. The operator L^* is called the adjoint operator of L and, for basic differential operators, the adjoint is found by performing integration-by-parts to move the differentials from the different arguments in the inner product. We illustrate with an example.

Example 1. *The differential operator L is given by*

$$Lu \equiv \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u) + \boldsymbol{\nabla} \cdot (\boldsymbol{b}u), \tag{2.25}$$

for the function $u \in \mathbb{R}^n$. We assume $u = 0$ on $\partial\Omega$, then the adjoint of L is found as

$$\begin{aligned}(Lu,\, v)_\Omega &= (\boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u) + \boldsymbol{\nabla} \cdot (\boldsymbol{b}u),\, v)_\Omega = (\boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u),\, v)_\Omega + (\boldsymbol{\nabla} \cdot (\boldsymbol{b}u),\, v)_\Omega \\ &= -\,(a(\boldsymbol{x})\boldsymbol{\nabla} u,\, \boldsymbol{\nabla} v)_\Omega - (\boldsymbol{b}u,\, \boldsymbol{\nabla} v)_\Omega = (u,\, \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} v))_\Omega - (u,\, \boldsymbol{b}\boldsymbol{\nabla} v)_\Omega \\ &= (u,\, \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} v) - \boldsymbol{b}\boldsymbol{\nabla} v)_\Omega = (u,\, L^* v)_\Omega\,.\end{aligned} \tag{2.26}$$

We observe that the adjoint of the first term in the differential operator L is itself, those operators are called self-adjoint operators. We also observe that we did not specify any boundary terms for the second argument v, but all boundary terms are cancelled in (2.26) despite that the condition $u = 0$ on $\partial\Omega$ would not suffice this. To cancel the remaining boundary terms, we specify the appropriate boundaries on v. This problem L^*v is called the adjoint problem to Lu. We redo the computation of (2.26), but specifically specify the boundaries and obtain the adjoint problem.

Example 2. *A boundary value problem (BVP) in $\mathbb{R}^n$ is given by*

$$\begin{aligned} Lu \equiv \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u) + \boldsymbol{\nabla} \cdot (\boldsymbol{b}u) = 0 \qquad & \boldsymbol{x} \in \Omega, \\ u = 0, \qquad & \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{2.27}$$

The adjoint problem of Lu is then found as

$$\begin{aligned}(Lu,\, v)_\Omega &= (\boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u) + \boldsymbol{\nabla} \cdot (\boldsymbol{b}u),\, v)_\Omega = (\boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u),\, v)_\Omega + (\boldsymbol{\nabla} \cdot (\boldsymbol{b}u),\, v)_\Omega \\ &= -\,(a(\boldsymbol{x})\boldsymbol{\nabla} u,\, \boldsymbol{\nabla} v)_\Omega - (\boldsymbol{b}u,\, \boldsymbol{\nabla} v)_\Omega + \int_{\partial\Omega} a(\boldsymbol{x})\boldsymbol{n} \cdot \boldsymbol{\nabla} u\, v \,\mathrm{d}S + \underbrace{\int_{\partial\Omega} \boldsymbol{b}u\, v \,\mathrm{d}S}_{=0} \\ &= (u,\, \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} v))_\Omega - (u,\, \boldsymbol{b}\boldsymbol{\nabla} v)_\Omega + \int_{\partial\Omega} a(\boldsymbol{x})\boldsymbol{n} \cdot \boldsymbol{\nabla} u\, v \,\mathrm{d}S - \underbrace{\int_{\partial\Omega} u\, a(\boldsymbol{x})\boldsymbol{n} \cdot \boldsymbol{\nabla} v \,\mathrm{d}S}_{=0}.\end{aligned} \tag{2.28}$$

To cancel the remaining boundary term, we specify the boundary on v as $v = 0$ on $\partial\Omega$. This gives the adjoint operator of L and corresponding adjoint problem to (2.27) as

$$\begin{aligned} L^* v \equiv \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} v) - \boldsymbol{b}\boldsymbol{\nabla} v = 0 \qquad & \boldsymbol{x} \in \Omega, \\ v = 0, \qquad & \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{2.29}$$

We extend this discussion to include time-dependent operators, and the so-called initial boundary value problems (IBVP). The method to find the adjoint of such operators is the same as for BVPs, using partial integration and choosing appropriate conditions for the adjoint problem. Unlike the original problem where the initial conditions are defined at the initial time the conditions of the adjoint problem are specified on the final time, so-called terminal time. We illustrate this with an example as well.

Example 3. *An initial boundary value problem (IBVP) in $\mathbb{R}^n$ is given by*

$$\begin{aligned} Lu \equiv \frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} u) + \boldsymbol{\nabla} \cdot (\boldsymbol{b}u) = 0 \qquad & (\boldsymbol{x}, t) \in \Omega \times (0, T], \\ u = 0, \qquad & (\boldsymbol{x}, t) \in \partial\Omega \times (0, T], \\ u = f(\boldsymbol{x}), \quad & \boldsymbol{x} \in \Omega,\ t = 0. \end{aligned} \tag{2.30}$$

We want to find the adjoint problem of (2.30), and we know from (2.29) the adjoint for the spatial operators and the boundary condition for the problem. We only evaluate the time derivative, which is found to be

$$\left(\frac{\partial u}{\partial t},\, v\right)_\Omega = -\left(u,\, \frac{\partial v}{\partial t}\right)_\Omega + \int_0^T (u,\, v)_\Omega \,\mathrm{d}t = -\left(u,\, \frac{\partial v}{\partial t}\right)_\Omega + (u,\, v)_\Omega \Big|_{t=T} - (u,\, v)_\Omega \Big|_{t=0}. \tag{2.31}$$

From the initial condition in (2.30), we know the last term to be $(u,\, v)_\Omega \big|_{t=0} = f(x)$. To cancel the other term, the terminal value of v must be $v\big|_{t=T} = u\big|_{t=0}$. The corresponding adjoint problem for (2.30) is then

$$\begin{aligned} L^* v \equiv -\frac{\partial v}{\partial t} + \boldsymbol{\nabla} \cdot (a(\boldsymbol{x})\boldsymbol{\nabla} v) - \boldsymbol{b}\boldsymbol{\nabla} v = 0 \qquad & (\boldsymbol{x}, t) \in \Omega \times (0, T], \\ v = 0, \qquad & (\boldsymbol{x}, t) \in \partial\Omega \times (0, T], \\ v = f(\boldsymbol{x}), \quad & \boldsymbol{x} \in \Omega,\ t = T. \end{aligned} \tag{2.32}$$

Adjoint problems are also known as dual problems, and for time-dependent problems we see that the corresponding adjoint is solved backwards in time, starting from the final time and solved to the initial time [1].

2.4 Optimisation

The brief explanation of the optimisation problems in the introduction, i.e. (1.1)-(1.5), is now extended further. A basic optimisation problem consist of an objective function $f(x)$ optimised with respect to some variable x subject to any number of constraints,

$$\begin{aligned} \underset{x}{\text{minimize}} \qquad & f(x) \\ \text{subject to} \qquad & \sum_{i=1}^{N} g_i(x) = 0, \end{aligned} \tag{2.33}$$

where $0 \leq i \leq N$ is the number of active constraints. In the case of (1.1)-(1.5), the constraints are part of the solution of an IBVP and therefore the problem is known as a PDE-constrained optimisation problem.

An optimal point is found by searching for the smallest value of $f(x)$ with the optimisation parameter x still being in the feasible region defined by the constraints. Denoting an optimal point as x^*, a basic optimisation algorithm is derived by Taylor expansion of the objective function near the region of a specified point. The optima is found locally as

$$f(x^*) < f(x) \quad \forall x \in S \text{ such that } x^* \neq x \text{ and } \|x - x^*\| < \text{TOL}, \tag{2.34}$$

where S is the feasible region defined by the constraints and TOL is a given tolerance. We will not discuss more exact definitions of optima, instead referring further reading to the book by Griva et. al [2].

Optimisation algorithms are usually iterative methods, starting by guessing a solution x_0. If x_0 satisfies the optimum, the algorithm terminates. Otherwise, a new solution x_1 is determined by moving from the initial solution with some step length α along a direction p_0, i.e. the next iteration is given by

$$x_1 = x_0 + \alpha_0 p_0 \tag{2.35}$$

where α_0 is the step length and p_0 is the descent direction. The algorithm repeats itself until an optimum satisfying (2.34) is found. The choice of a search direction p is central for these types of algorithms and a popular subgroup of these algorithms uses the gradient of the objective function as the search direction. These methods are known as gradient based optimisation algorithms. The most simple gradient based method is known as *Steepest descent*, which follows the direction on which the function decreases in value. We write as

$$x_{i+1} = x_i - \alpha_i \boldsymbol{\nabla} f(x_i) \tag{2.36}$$

with the step length computed by

$$\alpha_i = \frac{|(x_i - x_{i-1})\left(\boldsymbol{\nabla} f(x_i) - \boldsymbol{\nabla} f(x_{i-1})\right)|}{\left\|\boldsymbol{\nabla} f(x_i) - \boldsymbol{\nabla} f(x_{i-1})\right\|^2} \tag{2.37}$$

using line search and the Wolfe conditions, see [2] for further details. Steepest descent have, however, only linear convergence rate and there are better alternatives which converge faster.

A popular method with quadratic convergence rate is Newton's method, which is also a gradient based algorithm. This method is used for finding the root x^* to the equation $g(x^*) = 0$. If the function $g(x)$ defines the gradient of the objective function, its root in turn leads us to a stationary point of the objective. Making an initial guess to the solution as x_0 similar to Steepest descent, the next iteration is found as $x = x_0 + \Delta x$. The Taylor expansion of $g(x_0)$ reads

$$g(x) = g(x_0) + \Delta x \boldsymbol{\nabla} g(x_0) + \mathcal{O}\big(\Delta x^2\big). \tag{2.38}$$

As we are searching for a point where $g(x) = 0$, this results in an expression for Δx as

$$\Delta x = -\frac{g(x_0) + \mathcal{O}(\Delta x^2)}{\boldsymbol{\nabla} g(x_0)}. \tag{2.39}$$

Assuming $\mathcal{O}(\Delta x^2)$ is very small, this can be discarded and the approximation for the point x is found as

$$x = x_0 + \Delta x \approx x_0 - \frac{g(x_0)}{\boldsymbol{\nabla} g(x_0)}. \tag{2.40}$$

Given that $g(x)$ is the expression for the gradient to our objective function, the full expression for finding the minimum of $f(x)$ is given as follows

$$x_{i+1} = x_i - \frac{\boldsymbol{\nabla} f(x_i)}{\boldsymbol{\nabla}^2 f(x_i)} \tag{2.41}$$

where $\boldsymbol{\nabla}^2$ is the Hessian matrix. This is the classic Newton's method with quadratic convergence rate. Unfortunately, the Hessian is generally quite expensive to compute which affects the efficiency of the algorithm. This has led to the development of similar methods, which uses the fast convergence of the Newton's method but approximates the expensive Hessian to reduce the computational complexity. These methods are known as Quasi-Newton methods. One of the more popular Quasi-Newton methods is the BFGS algorithm which approximates the Hessian with a finite difference stencil in each iteration. The Hessian approximation is denoted by

$$\boldsymbol{B}_k \approx \boldsymbol{\nabla}^2 f(x_k) \tag{2.42}$$

and the next iteration of $\boldsymbol{x}$ according to equation (2.41) is therefore given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{B}_k^{-1} \boldsymbol{\nabla} f(\boldsymbol{x}_k). \tag{2.43}$$

The Hessian approximation at $\boldsymbol{x}_{k+1}$ is calculated using the BFGS formulation as

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^{\mathrm{T}}}{\boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k (\boldsymbol{B}_k \boldsymbol{s}_k)^{\mathrm{T}}}{\boldsymbol{s}_k^{\mathrm{T}} \boldsymbol{B}_k \boldsymbol{s}_k}, \tag{2.44}$$

where $\boldsymbol{y}_k = \boldsymbol{\nabla} f(\boldsymbol{x}_{k+1}) - \boldsymbol{\nabla} f(\boldsymbol{x}_k)$ and $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$ [2].

3 Numerical solution method

In this section, we derive the adjoint problem in both the continuous and semi-discrete case. We show our problem algorithm, using the forward and adjoint problem as well as a temporal integrator and optimisation method. We finally present the problem conditions which we use in a benchmark implementation to measure the efficiency and accuracy of our algorithm.

3.1 Adjoint problem

3.1.1 Continuous analysis

Recall the Lagrangian functional from Equation (1.7). Since the gradient of the cost functional includes the term $\frac{\partial u}{\partial a}$ and the Lagrangian by definition is equal to the cost function, the goal is to minimise the Lagrangian. The Lagrangian is then rewritten using the second term with the Lagrange multipliers λ. We want to rewrite

$$\frac{\mathrm{d}}{\mathrm{d}a}\int_0^T (\lambda,\, A(u))_\Omega \,\mathrm{d}t \tag{3.1}$$

such that

$$\int_0^T \left(2\,\mathcal{R}\,\delta(x-x_r),\, \frac{\partial u}{\partial a}\right)_\Omega \mathrm{d}t \tag{3.2}$$

is cancelled. This is done by finding the adjoint of the second argument $A(\cdot)$ which satisfies the relation

$$(\lambda,\, A(u))_\Omega = (A^*(\lambda),\, u)_\Omega\,. \tag{3.3}$$

We first derive the adjoint in the continuous case. We expand $A(u)$ and obtain

$$\begin{aligned}(\lambda,\, A(u))_\Omega &= \left(\lambda,\, \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right) + g(t)\delta(x-x_s)\right)_\Omega \\ &= \left(\lambda,\, \frac{\partial^2 u}{\partial t^2}\right)_\Omega - \left(\lambda,\, \frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right)\right)_\Omega + (\lambda,\, g(t)\delta(x-x_s))_\Omega\,.\end{aligned} \tag{3.4}$$

Assuming that λ is smooth and continuously dependent on x but not dependent on a, such that the gradient with respect to a on the last term is zero, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}a}\int_0^T (\lambda,\, g(t)\delta(x-x_s))_\Omega \,\mathrm{d}t = 0. \tag{3.5}$$

Therefore We can neglect the last term and evaluate the remaining terms separately using integration-by-parts. That is, for the first term

$$\begin{aligned}\int_0^T \left(\lambda,\, \frac{\partial^2 u}{\partial t^2}\right)_\Omega \mathrm{d}t &= \int_0^T \left(-\left(\frac{\partial \lambda}{\partial t},\, \frac{\partial u}{\partial t}\right)_\Omega + \left(\lambda,\, \frac{\partial u}{\partial t}\right)_\Omega \Big|_0^T\right) \mathrm{d}t \\ &= \int_0^T \left(\left(\frac{\partial^2 \lambda}{\partial t^2},\, u\right)_\Omega + \left(\lambda,\, \frac{\partial u}{\partial t}\right)_\Omega \Big|_0^T - \left(\frac{\partial \lambda}{\partial t},\, u\right)_\Omega \Big|_0^T\right) \mathrm{d}t.\end{aligned} \tag{3.6}$$

We want to cancel the initial and terminal conditions, only leaving the single inner product $\left(\frac{\partial^2 \lambda}{\partial t^2},\, u\right)_\Omega$. As we know the conditions for the forward problem (1.4-1.5), we assign the appropriate conditions for the adjoint problem such that these cancel. The conditions are

$$\lambda\Big|_{t=T} = 0, \tag{3.7}$$

$$\frac{\partial \lambda}{\partial t}\Big|_{t=T} = 0, \tag{3.8}$$

when combined with (1.4-1.5), left (3.6) as

$$\int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2},\, u\right)_\Omega \mathrm{d}t. \tag{3.9}$$

When taking the gradient of (3.9), only the second argument depends on a, namely,

$$\frac{\mathrm{d}}{\mathrm{d}a} \int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2},\, u\right)_\Omega \mathrm{d}t = \int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2},\, \frac{\mathrm{d}u}{\mathrm{d}a}\right)_\Omega \mathrm{d}t. \tag{3.10}$$

Using the product rule and chain rule to expand $\mathrm{d}u/\mathrm{d}a$ gives

$$\frac{\mathrm{d}}{\mathrm{d}a} u(x,t,a) = \frac{\partial x}{\partial a}\frac{\partial u}{\partial x} + \frac{\partial t}{\partial a}\frac{\partial u}{\partial t} + \frac{\partial u}{\partial a} = \frac{\partial u}{\partial a}. \tag{3.11}$$

As both t and x are not functions of a, their derivatives are zero. The final form of the first term after taking the gradient with respect to a becomes

$$\frac{\mathrm{d}}{\mathrm{d}a} \int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2},\, u\right)_\Omega \mathrm{d}t = \int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2},\, \frac{\partial u}{\partial a}\right)_\Omega \mathrm{d}t. \tag{3.12}$$

Now, for the second term of (3.4), we obtain

$$\int_0^T -\left(\lambda,\, \frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right)\right)_\Omega \mathrm{d}t = \int_0^T \left(\left(\frac{\partial \lambda}{\partial x},\, a(x)\frac{\partial u}{\partial x}\right)_\Omega - \lambda\, a(x)\frac{\partial u}{\partial x}\Big|_{x_1}^{x_m}\right) \mathrm{d}t. \tag{3.13}$$

From the boundary condition on the forward problem, (1.3), the boundary term cancels. When taking the full derivative with respect to a on this term, the derivative only acts on the second argument as λ does not depend on a, thus

$$\frac{\mathrm{d}}{\mathrm{d}a} \int_0^T \left(\frac{\partial \lambda}{\partial x},\, a(x)\frac{\partial u}{\partial x}\right)_\Omega \mathrm{d}t = \int_0^T \left(\frac{\partial \lambda}{\partial x},\, \frac{\mathrm{d}}{\mathrm{d}a} a(x)\frac{\partial u}{\partial x}\right)_\Omega \mathrm{d}t. \tag{3.14}$$

Evaluating the second argument using the product and chain rule reads as follows

$$\frac{\mathrm{d}}{\mathrm{d}a}a(x)\frac{\partial u}{\partial x} = \frac{\mathrm{d}a}{\mathrm{d}a}\frac{\partial u}{\partial x} + a(x)\frac{\mathrm{d}}{\mathrm{d}a}\left(\frac{\partial u}{\partial x}\right) = \frac{\partial u}{\partial x} + a(x)\frac{\partial}{\partial x}\left(\frac{\mathrm{d}u}{\mathrm{d}a}\right) = \frac{\partial u}{\partial x} + a(x)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial a}\right). \quad (3.15)$$

Here, we use the result (3.11), $\frac{\mathrm{d}a}{\mathrm{d}a} = 1$ and the assumption that the differential order between $\frac{\partial}{\partial a}$ and $\frac{\partial}{\partial x}$ does not matter. Using this and inserting back into (3.14), we get

$$\int_0^T \left(\left(\frac{\partial \lambda}{\partial x}, \frac{\partial u}{\partial x} \right)_\Omega + \left(\frac{\partial \lambda}{\partial x}, a(x)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial a}\right) \right)_\Omega \right) \mathrm{d}t \quad (3.16)$$

and continuing using partial integration on the second term we obtain

$$\int_0^T \left(\frac{\partial \lambda}{\partial x}, a(x)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial a}\right) \right)_\Omega \mathrm{d}t = \int_0^T \left(- \left(\frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x} \right), \frac{\partial u}{\partial a} \right)_\Omega + a(x)\frac{\partial \lambda}{\partial x}\frac{\partial u}{\partial a}\Big|_{x_1}^{x_m} \right) \mathrm{d}t. \quad (3.17)$$

Again, we want the boundary integral to cancel which we achieve by choosing boundary conditions on λ. We chose

$$\frac{\partial \lambda}{\partial x} = 0 \quad \forall x \in \partial\Omega \quad (3.18)$$

which cancels the term. The gradient with respect to a takes the form

$$\int_0^T \left(\left(\frac{\partial \lambda}{\partial x}, \frac{\partial u}{\partial x} \right)_\Omega - \left(\frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x} \right), \frac{\partial u}{\partial a} \right)_\Omega \right) \mathrm{d}t \quad (3.19)$$

and leaves us with the following for the gradient of the Lagrangian cost functional (1.7)

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}a}\mathcal{L} = &\int_0^T \left(2\,\mathcal{R}\,\delta(x - x_r), \frac{\partial u}{\partial a} \right)_\Omega \mathrm{d}t + \int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2}, \frac{\partial u}{\partial a} \right)_\Omega \mathrm{d}t \\ &+ \int_0^T \left(\frac{\partial \lambda}{\partial x}, \frac{\partial u}{\partial x} \right)_\Omega \mathrm{d}t - \int_0^T \left(\frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x} \right), \frac{\partial u}{\partial a} \right)_\Omega \mathrm{d}t. \end{aligned} \quad (3.20)$$

Next, this is rearranged using the linearity property of the inner product as

$$\frac{\mathrm{d}}{\mathrm{d}a}\mathcal{L} = \int_0^T \left(2\,\mathcal{R}\,\delta(x - x_r) + \frac{\partial^2 \lambda}{\partial t^2} - \frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x} \right), \frac{\partial u}{\partial a} \right)_\Omega \mathrm{d}t + \int_0^T \left(\frac{\partial \lambda}{\partial x}, \frac{\partial u}{\partial x} \right)_\Omega \mathrm{d}t. \quad (3.21)$$

Since we want to avoid the term $\frac{\partial u}{\partial a}$, the inner product is set to zero by choosing λ as the solution to the adjoint problem formulation. That is

$$\begin{aligned} \int_0^T \left(\frac{\partial^2 \lambda}{\partial t^2} - \frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x} \right) + 2\,\mathcal{R}\,\delta(x - x_r), \frac{\partial u}{\partial a} \right)_\Omega \mathrm{d}t = 0, \\ \frac{\partial^2 \lambda}{\partial t^2} - \frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x} \right) + 2\,\mathcal{R}\,\delta(x - x_r) = 0, \end{aligned} \quad (3.22)$$

thus, the adjoint problem reads as follows,

$$\frac{\partial^2 \lambda}{\partial t^2} - \frac{\partial}{\partial x}\left(a(x)\frac{\partial \lambda}{\partial x}\right) + 2\,\mathcal{R}\,\delta(x - x_r) = 0, \qquad (x,t) \in \Omega \times (0,T], \tag{3.23}$$

$$\frac{\partial \lambda}{\partial x} = 0, \qquad (x,t) \in \partial\Omega \times (0,T], \tag{3.24}$$

$$\lambda = 0, \qquad x \in \Omega,\, t = T, \tag{3.25}$$

$$\frac{\partial \lambda}{\partial t} = 0, \qquad x \in \Omega,\, t = T. \tag{3.26}$$

The final result for the gradient of the Lagrangian is therefore given by

$$\frac{\mathrm{d}}{\mathrm{d}a}\mathcal{L}(u,a,\lambda) = -\int_0^T \left(\lambda,\, \frac{\partial^2 u}{\partial x^2}\right)_\Omega \mathrm{d}t. \tag{3.27}$$

3.1.2 Semi-discrete analysis

We use the SBP-SAT method to discretise our optimisation problem (1.1-1.5). The semi-discrete problem reads

$$\underset{\boldsymbol{a}}{\text{minimise}} \quad J_h(\boldsymbol{v},\boldsymbol{a}) = \int_0^T \delta_h(\boldsymbol{x} - \boldsymbol{x_r})^{\mathrm{T}} \mathcal{R}_h(\boldsymbol{v},\boldsymbol{a})^2 \,\mathrm{d}t \tag{3.28}$$

$$\text{subject to} \quad \frac{\partial^2 \boldsymbol{v}}{\partial t^2} - \mathbf{D}_w^{(a)}\boldsymbol{v} - g(t)\delta_h(\boldsymbol{x} - \boldsymbol{x}_s) = 0, \qquad (x,t) \in \Omega \times (0,T], \tag{3.29}$$

$$\boldsymbol{v} = 0, \qquad x \in \Omega,\, t = 0, \tag{3.30}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} = 0, \qquad x \in \Omega,\, t = 0. \tag{3.31}$$

Here, $\boldsymbol{v}$ is the discrete solution vector for u, and the subscript h denoted the discrete equivalent for the cost functional J_h, residual $\mathcal{R}_h$ and Dirac delta δ_h. The residual is defined as

$$\mathcal{R}_h(\boldsymbol{v},\boldsymbol{a}) = \boldsymbol{v}(\boldsymbol{a}) - u_{\text{data}} \tag{3.32}$$

and the SBP operator $\mathbf{D}_w^{(a)}$ has the form

$$\mathbf{D}_w^{(a)} = -\mathbf{H}^{-1}\mathbf{M}^{(a)} = -\mathbf{H}^{-1}\left(\mathbf{D}_1^{\mathrm{T}}\mathbf{H}\mathbf{A}^{(a)}\mathbf{D}_1 + \mathbf{R}^{(a)}\right). \tag{3.33}$$

This follows from the construction of the operator, equation (2.12) and the homogeneous Neumann condition which cancels the term $\bar{\mathbf{B}}\mathbf{S}$ in $\mathbf{D}_2^{(a)}$. Similarly, the semi-discrete Lagrangian becomes

$$\mathcal{L}_h(\boldsymbol{v},\boldsymbol{a},\boldsymbol{\lambda}) = J_h(\boldsymbol{v},\boldsymbol{a}) + \int_0^T \left(\boldsymbol{\lambda},\, \frac{\partial^2 \boldsymbol{v}}{\partial t^2} - \mathbf{D}_w^{(a)}\boldsymbol{v} - g(t)\delta_h(\boldsymbol{x} - \boldsymbol{x}_r)\right)_{\mathbf{H}} \mathrm{d}t. \tag{3.34}$$

In the discrete case, the discrete density parameter is vector valued for every value of x compared to a function of x in the continuous case. As such, when we optimise with respect to $\boldsymbol{a}$, the gradient of the cost functional becomes a component-wise partial derivative of $\boldsymbol{a}$ instead of a full derivative. That is,

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{a}} = \begin{bmatrix} \dfrac{\partial}{\partial a_1} & \dfrac{\partial}{\partial a_2} & \cdots & \dfrac{\partial}{\partial a_m} \end{bmatrix}^{\mathrm{T}} \tag{3.35}$$

and following equation (1.9) and (1.11) the discrete gradient of the cost functional takes the form

$$\frac{\partial}{\partial a_i} J_h(\boldsymbol{v}, \boldsymbol{a}) = \int_0^T \left(\frac{1}{\mathbf{H}_{rr}} 2\mathcal{R}_h \delta_h(\boldsymbol{x} - \boldsymbol{x_r}),\, \partial_{a_i} \boldsymbol{v} \right)_{\mathbf{H}} \mathrm{d}t. \tag{3.36}$$

The addition of the factor $\frac{1}{\mathbf{H}_{rr}}$ in front of the residual comes from rewriting the expression in the discrete norm, where the subscript r is the index for the receiver location. As in the continuous case, we want to cancel all terms containing the expression $\frac{\partial \boldsymbol{v}}{\partial a_i}$ by finding the adjoint to the semi-discrete problem. Following the steps in the previous section, starting with equation (3.5) we once again neglect the source term when evaluating the derivative. Using the linearity of the discrete inner product, we separate the two terms of the forward problem

$$\left(\boldsymbol{\lambda},\, \frac{\partial^2 \boldsymbol{v}}{\partial t^2} - \mathbf{D}_w^{(a)} \boldsymbol{v} \right)_{\mathbf{H}} = \left(\boldsymbol{\lambda},\, \frac{\partial^2 \boldsymbol{v}}{\partial t^2} \right)_{\mathbf{H}} - \left(\boldsymbol{\lambda},\, \mathbf{D}_w^{(a)} \boldsymbol{v} \right)_{\mathbf{H}}. \tag{3.37}$$

As we were not discretising the temporal part, the discrete adjoint of the time derivative is the same as the continuous. That is

$$\int_0^T \left(\boldsymbol{\lambda},\, \frac{\partial^2 \boldsymbol{v}}{\partial t^2} \right)_{\mathbf{H}} \mathrm{d}t = \int_0^T \left(\frac{\partial^2 \boldsymbol{\lambda}}{\partial t^2},\, \boldsymbol{v} \right)_{\mathbf{H}} \mathrm{d}t \tag{3.38}$$

with the temporal conditions

$$\boldsymbol{\lambda}\Big|_{t=T} = 0, \qquad \frac{\partial \boldsymbol{\lambda}}{\partial t}\Big|_{t=T} = 0. \tag{3.39}$$

As in the continuous case, the discrete $\boldsymbol{\lambda}$ also does not dependent on $\boldsymbol{a}$ and the discrete total derivative of (3.38) is given by

$$\frac{\partial}{\partial a_i} \int_0^T \left(\frac{\partial^2 \boldsymbol{\lambda}}{\partial t^2},\, \boldsymbol{v} \right)_{\mathbf{H}} \mathrm{d}t = \int_0^T \left(\frac{\partial^2 \boldsymbol{\lambda}}{\partial t^2},\, \frac{\partial \boldsymbol{v}}{\partial a_i} \right)_{\mathbf{H}} \mathrm{d}t. \tag{3.40}$$

Continuing with the discrete SBP operator, similarly as (3.14), the total derivative was only applied on the second argument. Since both the SBP operator and the discrete solution vector does not depend on $\boldsymbol{a}$, applying the product rule gives

$$\frac{\partial}{\partial a_i} \left(\boldsymbol{\lambda},\, \mathbf{D}_w^{(a)} \boldsymbol{v} \right)_{\mathbf{H}} = \left(\boldsymbol{\lambda},\, \frac{\partial}{\partial a_i} \mathbf{D}_w^{(a)} \boldsymbol{v} \right)_{\mathbf{H}} = \left(\boldsymbol{\lambda},\, \frac{\partial}{\partial a_i} \mathbf{D}_w^{(a)} \boldsymbol{v} \right)_{\mathbf{H}} + \left(\boldsymbol{\lambda},\, \mathbf{D}_w^{(a)} \frac{\partial}{\partial a_i} \boldsymbol{v} \right)_{\mathbf{H}}. \tag{3.41}$$

The component wise partial derivative on the SBP operator becomes

$$\frac{\partial}{\partial a_i}\mathbf{D}_w^{(a)} = \mathbf{D}_w^{(e_i)} \tag{3.42}$$

where e_i is the ith column of an identity matrix with the appropriate size. Due to the construction of $\mathbf{D}_w^{(a)}$ and $\mathbf{D}_w^{(e_i)}$ by the homogeneous Neumann conditions of the continuous problem the boundary terms are zero. Using the symmetric, positive definite properties of the matrices $\mathbf{H}$ and $\mathbf{D}_w^{(a)}$, we can rearrange and move our discrete operator from the second to the first argument in the inner product as

$$\begin{aligned}\left(\boldsymbol{\lambda},\, \mathbf{D}_w^{(a)}\frac{\partial \boldsymbol{v}}{\partial a_i}\right)_{\mathbf{H}} =& \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{H}\mathbf{D}_w^{(a)}\frac{\partial \boldsymbol{v}}{\partial a_i} = \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{D}_w^{(a)}\mathbf{H}\frac{\partial \boldsymbol{v}}{\partial a_i} = \left(\mathbf{D}_w^{(a)}\boldsymbol{\lambda}\right)^{\mathrm{T}}\mathbf{H}\frac{\partial \boldsymbol{v}}{\partial a_i} \\ =& \left(\mathbf{D}_w^{(a)}\boldsymbol{\lambda},\, \frac{\partial \boldsymbol{v}}{\partial a_i}\right)_{\mathbf{H}}.\end{aligned} \tag{3.43}$$

We now obtain an expression for the derivative of the semi-discrete Lagrangian,

$$\begin{aligned}\frac{\partial}{\partial a_i}\mathcal{L}_h(\boldsymbol{v}, a, \boldsymbol{\lambda}) =& \int_0^T \left(\frac{1}{\mathbf{H}_{rr}}2\mathcal{R}_h\delta_h(\boldsymbol{x}-\boldsymbol{x_r}) + \frac{\partial^2\boldsymbol{\lambda}}{\partial t^2} - \mathbf{D}_w^{(a)}\boldsymbol{\lambda},\, \frac{\partial \boldsymbol{v}}{\partial a_i}\right)_{\mathbf{H}} \mathrm{d}t \\ &- \int_0^T \left(\boldsymbol{\lambda},\, \mathbf{D}_w^{(e_i)}\boldsymbol{v}\right)_{\mathbf{H}} \mathrm{d}t.\end{aligned} \tag{3.44}$$

The semi-discrete adjoint problem now read as follows

$$\frac{\partial^2\boldsymbol{\lambda}}{\partial t^2} - \mathbf{D}_w^{(a)}\boldsymbol{\lambda} + \frac{1}{\mathbf{H}_{rr}}2\mathcal{R}_h\delta_h(\boldsymbol{x}-\boldsymbol{x_r}) = 0, \qquad (x,t)\in\Omega\times(0,T], \tag{3.45}$$

$$\boldsymbol{\lambda} = 0, \qquad x\in\Omega,\, t=T, \tag{3.46}$$

$$\frac{\partial\boldsymbol{\lambda}}{\partial t} = 0, \qquad x\in\Omega,\, t=T. \tag{3.47}$$

The resulting gradient of the semi-discrete Lagrangian for one component is found to be

$$\frac{\partial}{\partial a_i}\mathcal{L}_h(\boldsymbol{v}, \boldsymbol{a}, \boldsymbol{\lambda}) = -\int_0^T \left(\boldsymbol{\lambda},\, \mathbf{D}_w^{(e_i)}\boldsymbol{v}\right)_{\mathbf{H}} \mathrm{d}t \tag{3.48}$$

and the full vector of the semi-discrete Lagrangian is then

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{a}}\mathcal{L}_h(\boldsymbol{v}, \boldsymbol{a}, \boldsymbol{\lambda}) = \begin{bmatrix}\frac{\partial}{\partial a_1}\mathcal{L}_h & \frac{\partial}{\partial a_2}\mathcal{L}_h & \dots & \frac{\partial}{\partial a_m}\mathcal{L}_h\end{bmatrix}^{\mathrm{T}}. \tag{3.49}$$

3.2 Solution algorithm

The optimisation algorithm works iteratively, updating the searched for parameter using the semi-discrete gradient of the cost functional (3.48-3.49). In each iteration, four main steps are performed

(i) solve the forward problem u,

(ii) solve the adjoint problem λ,

(iii) compute the gradient of the cost function $\frac{\mathrm{d}J}{\mathrm{d}a}$,

(iv) compute the unknown parameter a.

We solve the forward and adjoint problem to find the gradient of the cost functional. The parameter a is then updated. The algorithm terminates when either the cost functional value becomes less than the tolerance or if it can not find a descent direction. We describe the process using Algorithm (1)-(2). The first Algorithm (1) shows how the optimisation function was called to give a solution to a.

Algorithm 1: Main algorithm.

Data: Initial guess of a at every spatial point
Result: Final solution of a

Set up the domain
Store measured receiver data
Set up the optimisation method such that the gradient is provided by an explicit function
Run the optimisation method using the initial guess of a

Algorithm (2) shows the computational steps for finding the gradient of the cost functional used in the optimiser. Note that this is most often called more than once to achieve a cost function value lower than the specified tolerance.

Algorithm 2: Cost functional gradient.

Data: Current value of a
Result: Gradient of cost function $\frac{\mathrm{d}J}{\mathrm{d}a}$

Solve the forward problem u in x for all t_i
Compute the residual in x_r for all t_i
Solve the adjoint problem λ for all $\tau_i \leftarrow T - t_i$
for *every point of* a **do**
  Compute the gradient of cost function $\frac{\mathrm{d}J}{\mathrm{d}a} \leftarrow -\int_0^T \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{H} \mathbf{D}_w^{(e_i)} \boldsymbol{v} \, \mathrm{d}t$
end

As the adjoint problem is solved backwards in time, a backward time step τ is defined, possibly different from that, used in the forward problem. The temporal integrals is computed with the fourth order Simpsons method combined with Richardson extrapolation.

3.3 Problem conditions

We test our proposed algorithm for different problem conditions to measure efficiency and accuracy. For all benchmark cases, we restrict our analysis in one spatial dimension and always use the homogeneous initial and Neumann boundary conditions prescribed in (1.3-1.5). The computational domain Ω is restricted in $\Omega = x \in [0, 1]$ and defined by the grid

$$x_i = (i-1)h, \quad 0 \leq i \leq m, \quad h = \frac{1}{m-1} \tag{3.50}$$

where m is the number of grid points. We use two different temporal integration methods, the fourth order explicit Runge Kutta (ERK) scheme and the explicit central difference (CD) scheme. To use the ERK scheme, the semi-discrete forward and adjoint problems, (3.29) and (3.45), are rewritten in first order form as

$$\frac{\partial}{\partial t}\begin{bmatrix}\frac{\partial \boldsymbol{v}}{\partial t}\\ \boldsymbol{v}\end{bmatrix} = \begin{bmatrix}0 & \mathbf{D}_w\\ \mathbf{I} & 0\end{bmatrix}\begin{bmatrix}\frac{\partial \boldsymbol{v}}{\partial t}\\ \boldsymbol{v}\end{bmatrix} + \begin{bmatrix}g(t)\delta_h(x-x_s)\\ 0\end{bmatrix}, \tag{3.51}$$

$$\frac{\partial}{\partial t}\begin{bmatrix}\frac{\partial \boldsymbol{\lambda}}{\partial t}\\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}0 & \mathbf{D}_w\\ \mathbf{I} & 0\end{bmatrix}\begin{bmatrix}\frac{\partial \boldsymbol{\lambda}}{\partial t}\\ \boldsymbol{\lambda}\end{bmatrix} - \begin{bmatrix}\frac{1}{\mathbf{H}_{rr}}2\mathcal{R}_h\delta_h(x-x_s)\\ 0\end{bmatrix}, \tag{3.52}$$

equivalent to

$$\frac{\partial \boldsymbol{v}^n}{\partial t} = F\left(\boldsymbol{v}_h^n, t^n\right). \tag{3.53}$$

The ERK time stepping is then computed as

$$\begin{aligned}
\boldsymbol{w}^{(1)} &= F\left(\boldsymbol{v}^n, t^n\right),\\
\boldsymbol{w}^{(2)} &= F\left(\boldsymbol{v}^n + \frac{1}{2}k\boldsymbol{w}^{(1)}, t^n + \frac{1}{2}k\right),\\
\boldsymbol{w}^{(3)} &= F\left(\boldsymbol{v}^n + \frac{1}{2}k\boldsymbol{w}^{(2)}, t^n + \frac{1}{2}k\right),\\
\boldsymbol{w}^{(4)} &= F\left(\boldsymbol{v}^n + k\boldsymbol{w}^{(3)}, t^n + k\right),\\
\boldsymbol{v}^{n+1} &= \boldsymbol{v}^n + \frac{1}{6}\left(\boldsymbol{w}^{(1)} + 2\boldsymbol{w}^{(2)} + 2\boldsymbol{w}^{(3)} + \boldsymbol{w}^{(4)}\right),
\end{aligned} \tag{3.54}$$

for some time step k. The CD scheme is applied directly on the semi-discrete form as it can be written as a direct approximation of the second derivative. The fourth order accurate CD scheme for the problems is derived from the second order accurate CD scheme which reads as follows,

$$\begin{aligned}
\frac{\boldsymbol{v}^{n+1} - 2\boldsymbol{v}^n + \boldsymbol{v}^{n-1}}{k^2} &= \mathbf{D}_w\boldsymbol{v}^n + g^n\delta_h(x-x_s),\\
\frac{\boldsymbol{\lambda}^{n+1} - 2\boldsymbol{\lambda}^n + \boldsymbol{\lambda}^{n-1}}{k^2} &= \mathbf{D}_w\boldsymbol{\lambda}^n - \frac{1}{\mathbf{H}_{rr}}2\mathcal{R}_h\delta_h(x-x_s).
\end{aligned} \tag{3.55}$$

Because a higher order method is needed, we use Taylor expansion to derive such a scheme. Expanding the terms $\boldsymbol{v}^{n+1}$ and $\boldsymbol{v}^{n-1}$ around $\boldsymbol{v}^n$ we obtain,

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + k\frac{\partial \boldsymbol{v}}{\partial t}^n + \frac{k^2}{2!}\frac{\partial^2 \boldsymbol{v}}{\partial t^2}^n + \frac{k^3}{3!}\frac{\partial^3 \boldsymbol{v}}{\partial t^3}^n + \frac{k^4}{4!}\frac{\partial^4 \boldsymbol{v}}{\partial t^4}^n + \mathcal{O}\big(k^5\big), \tag{3.56}$$

$$\boldsymbol{v}^{n-1} = \boldsymbol{v}^n - k\frac{\partial \boldsymbol{v}}{\partial t}^n + \frac{k^2}{2!}\frac{\partial^2 \boldsymbol{v}}{\partial t^2}^n - \frac{k^3}{3!}\frac{\partial^3 \boldsymbol{v}}{\partial t^3}^n + \frac{k^4}{4!}\frac{\partial^4 \boldsymbol{v}}{\partial t^4}^n + \mathcal{O}\big(k^5\big). \tag{3.57}$$

Inserting (3.56) and (3.57) into equation (3.55) the higher order terms are

$$\frac{\boldsymbol{v}^{n+1} - 2\boldsymbol{v}^n + \boldsymbol{v}^{n-1}}{k^2} - \frac{k^2}{12}\frac{\partial^4 \boldsymbol{v}^n}{\partial t^4} + \mathcal{O}\big(k^4\big) = \mathbf{D}_w \boldsymbol{v}^n + g(t^n)\delta_h(x - x_s). \tag{3.58}$$

We now resolve the unknown term $\frac{\partial^4 \boldsymbol{v}}{\partial t^4}^n$. We use the fact that $\frac{\partial^2 \boldsymbol{v}^n}{\partial t^2}$ is known from the original equation, and apply the partial derivative with respect to t two times for our original equation. This result in

$$\frac{\partial^4 \boldsymbol{v}^n}{\partial t^4} = \mathbf{D}_w \frac{\partial^2 \boldsymbol{v}^n}{\partial t^2} + \frac{\partial^2 g(t^n)}{\partial t^2}\delta_h(x - x_s). \tag{3.59}$$

The scheme is then expanded into

$$\frac{\boldsymbol{v}^{n+1} - 2\boldsymbol{v}^n + \boldsymbol{v}^{n-1}}{k^2} - \frac{k^2}{12}\mathbf{D}_w \frac{\partial^2 \boldsymbol{v}^n}{\partial t^2} = \mathbf{D}_w \boldsymbol{v}^n + \left(g(t^n) + \frac{k^2}{12}\frac{\partial^2 g(t^n)}{\partial t^2}\right)\delta_h(x - x_s), \tag{3.60}$$

and as $\frac{\partial^2 \boldsymbol{v}^n}{\partial t^2}$ is given by the model, this is inserted one more time. Then, the fourth order accurate scheme takes the form

$$\begin{aligned}
\frac{\boldsymbol{v}^{n+1} - 2\boldsymbol{v}^n + \boldsymbol{v}^{n-1}}{k^2} &= \mathbf{D}_w \left(1 + \frac{k^2}{12}\mathbf{D}_w\right)\boldsymbol{v}^n + \left(g^n + \frac{k^2}{12}\left(\frac{\partial^2 g^n}{\partial t^2} + \mathbf{D}_w g^n\right)\right)\delta_h(x - x_s), \\
\frac{\boldsymbol{\lambda}^{n+1} - 2\boldsymbol{\lambda}^n + \boldsymbol{\lambda}^{n-1}}{k^2} &= \mathbf{D}_w \left(1 + \frac{k^2}{12}\mathbf{D}_w\right)\boldsymbol{\lambda}^n - \frac{1}{\mathbf{H}_{rr}}\left(2\mathcal{R}_h^n + \frac{k^2}{6}\left(\frac{\partial^2 \mathcal{R}_h^n}{\partial t^2} + \mathbf{D}_w \mathcal{R}_h^n\right)\right)\delta_h(x - x_r).
\end{aligned} \tag{3.61}$$

We use two different source functions $g(t)$, a Gaussian function and a Ricker wavelet which is the negative normalised second derivative of a Gaussian function. The source functions read

$$g(t)_{\text{Gauss}} = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(t - t_c)^2}{2\sigma^2}\right), \tag{3.62}$$

$$g(t)_{\text{Ricker}} = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}\left(1 - \frac{(t - t_c)^2}{\sigma^2}\right)\exp\left(-\frac{(t - t_c)^2}{2\sigma^2}\right), \tag{3.63}$$

for some standard deviation, or width, σ and activation time t_c. In the numerical experiments, we use $\sigma = 0.1$ and $t_c = 5\sigma$.

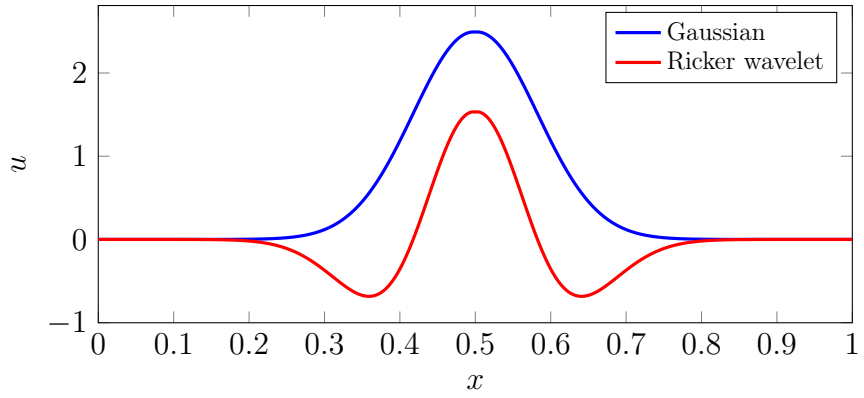


Figure 3.1: The solution of (1.2-1.5) with the two source functions with $\sigma = 0.1$, $t_c = 0.5$ and source position $x_s = 0.5$.

We solve the minimisation problem using two different gradient based optimisation methods. Using the optimisation toolbox in Matlab, the problem is solved using the second order Quasi-Newton method with the BFGS Hessian update. We also test the first order Steepest descent method, implemented following (2.36)-(2.37). The accepted tolerance is set to TOL $\leq 10^{-6}$ for both optimisation methods. The algorithms are also provided with an initial guess for a at every spatial point and it is assumed we always know the exact value on the boundaries for this guess. Finally, we test the algorithm using three different configurations of $a(x)$, either as a smooth, constant or discontinuous profile. The discontinuous profile is expected to be especially challenging for the algorithm as the numerical schemes assumes the parameter a to be a smooth function.

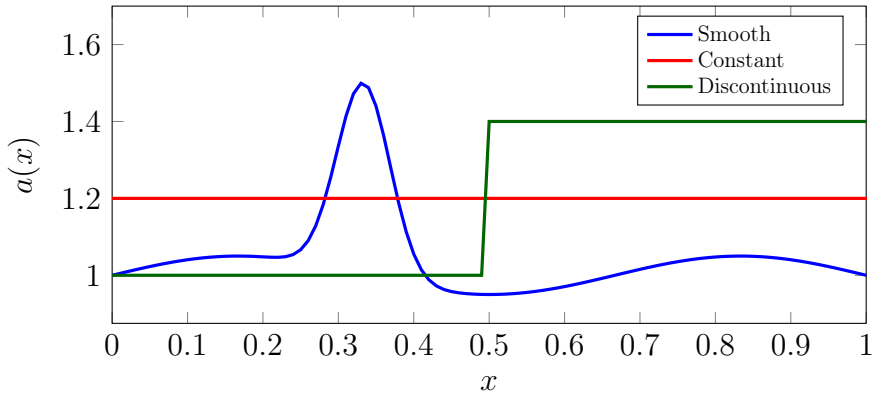


Figure 3.2: The three different profiles of $a(x)$ used as the exact solution.

4 Results

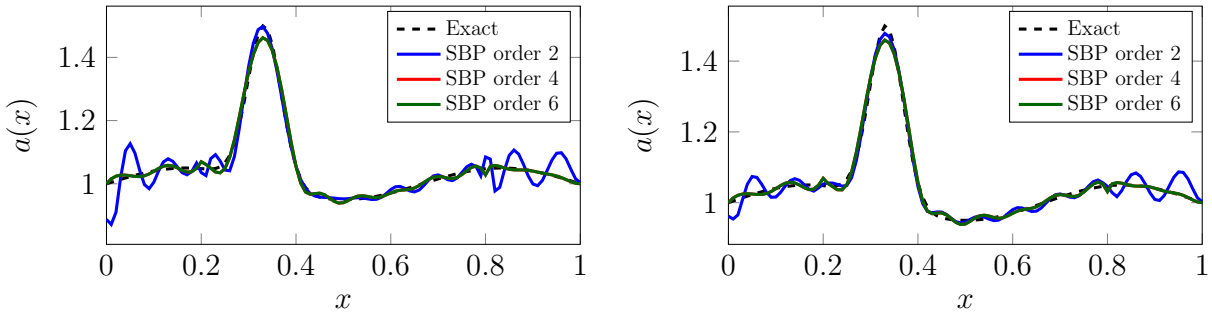
In this section, we present our results. The algorithm is tested for benchmark cases with the different profiles of the parameter, source function, optimisation method, approximation order of the SBP operator and choice of temporal integrator. The overall performance and efficiency of the algorithm are evaluated to identify potential bottlenecks and synergies.

4.1 Benchmark cases

We perform our proof-of-concept implementations for the three different parameter profiles (Figure 3.2), two different source functions (Figure 3.1) and two different optimisers. We use two receivers at the positions $x_r = 0.2, 0.8$ and the source at the left boundary $x_s = 0$. Applying the second, fourth and sixth order accurate SBP operators with $m = 101$ grid points and the ERK temporal integrator for this analysis. All the tests are initiated with a guess of one, for all a_i except at the boundaries a_1, a_m which are set to the correct parameter values.

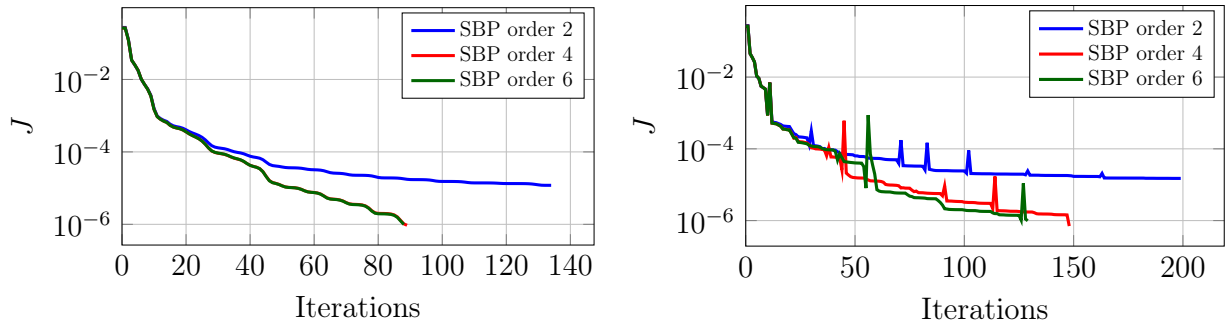
4.1.1 Gauss source

4.1.1.1 Smooth profile



(a) BFGS **(b)** Steepest descent

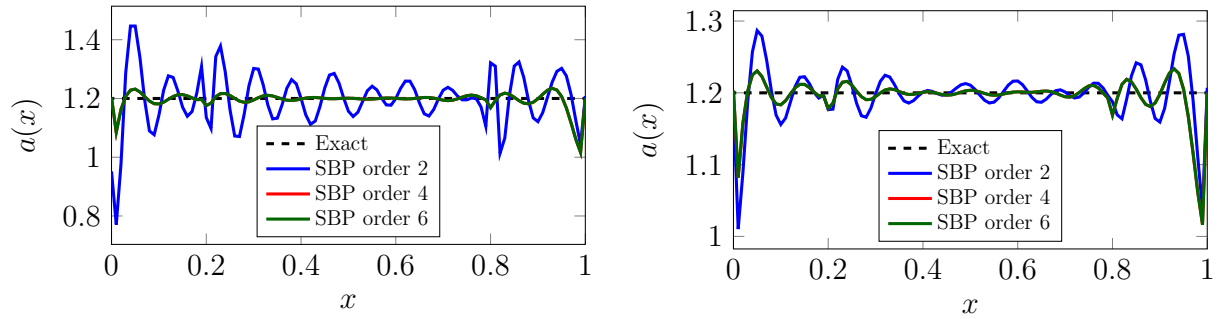
Figure 4.1: Solution for the smooth profile of a, using the Gaussian source function, ERK integrator and the two optimisers.



(a) BFGS **(b)** Steepest descent

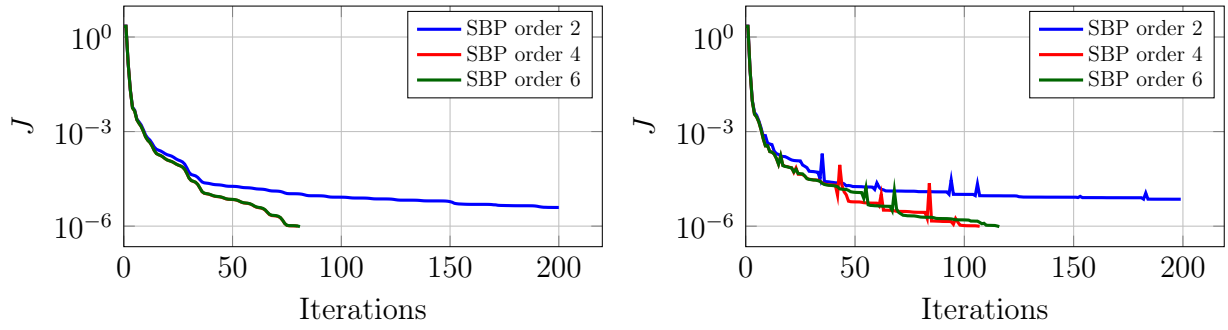
Figure 4.2: Number of iterations for the two optimisers using the smooth profile of a, Gaussian source function and ERK integrator.

4.1.1.2 Constant profile



(a) BFGS **(b)** Steepest descent

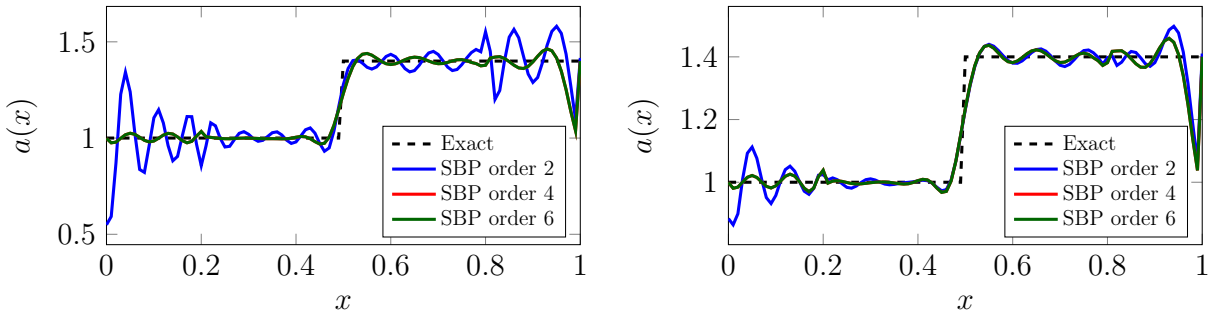
Figure 4.3: Solution for the constant profile of a, using the Gaussian source function, ERK integrator and the two optimisers.



(a) BFGS **(b)** Steepest descent

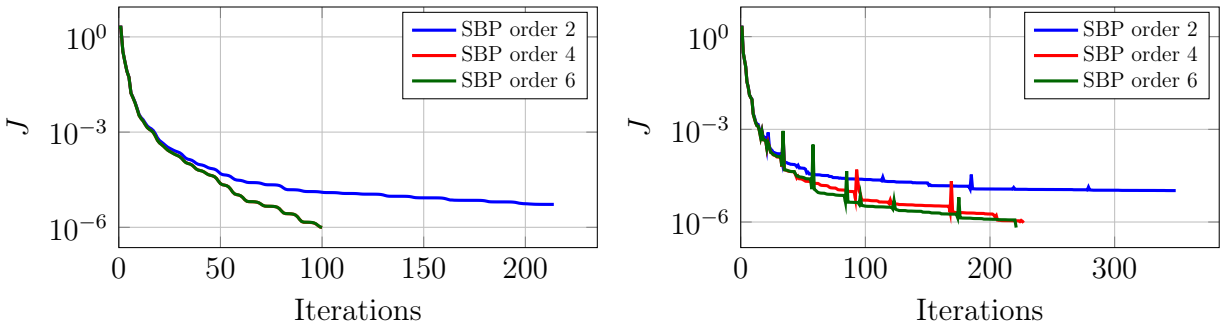
Figure 4.4: Number of iterations for the two optimisers using the constant profile of a, Gaussian source function and ERK integrator.

4.1.1.3 Discontinuous profile



(a) BFGS **(b)** Steepest descent

Figure 4.5: Solution for the discontinuous profile of a, using the Gaussian source function, ERK integrator and the two optimisers.

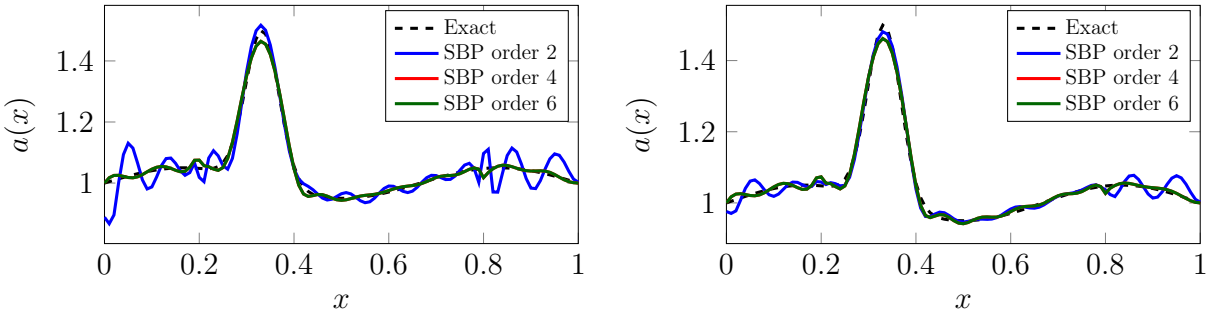


(a) BFGS **(b)** Steepest descent

Figure 4.6: Number of iterations for the two optimisers using the discontinuous profile of a, Gaussian source function and ERK integrator.

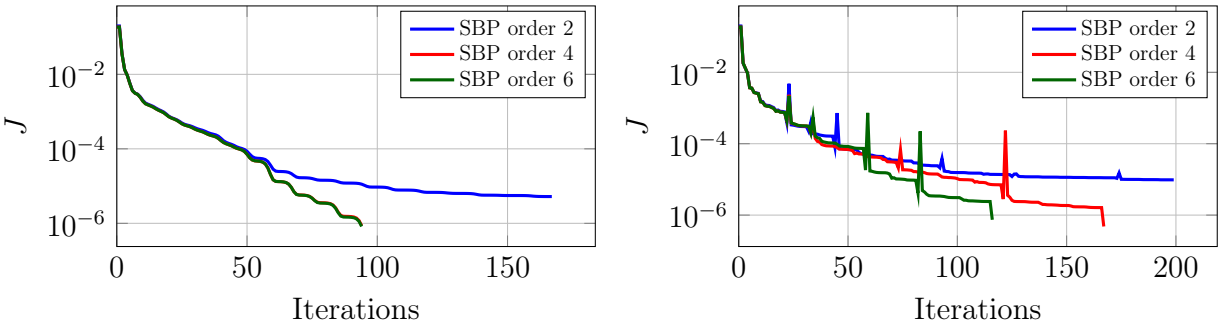
4.1.2 Ricker wavelet source

4.1.2.1 Smooth profile



(a) BFGS **(b)** Steepest descent

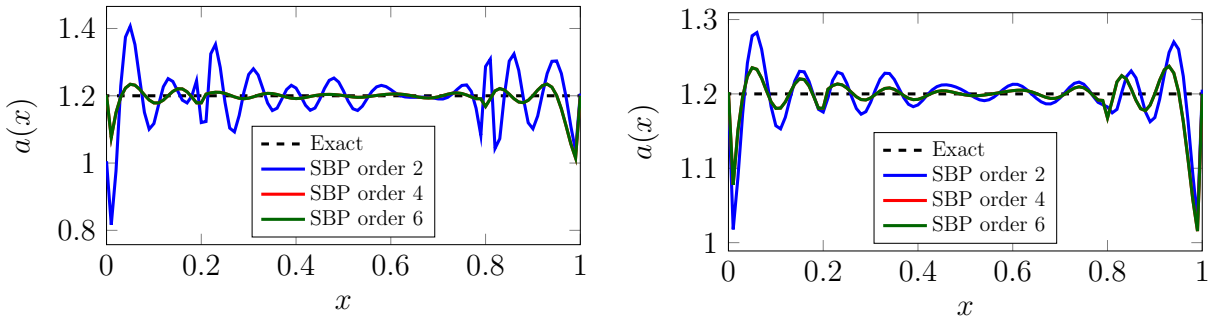
Figure 4.7: Solution for the smooth profile of a, using the Ricker-wavelet source function, ERK integrator and the two optimisers.



(a) BFGS **(b)** Steepest descent

Figure 4.8: Number of iterations for the two optimisers using the smooth profile of a, Ricker-wavelet source function and ERK integrator.

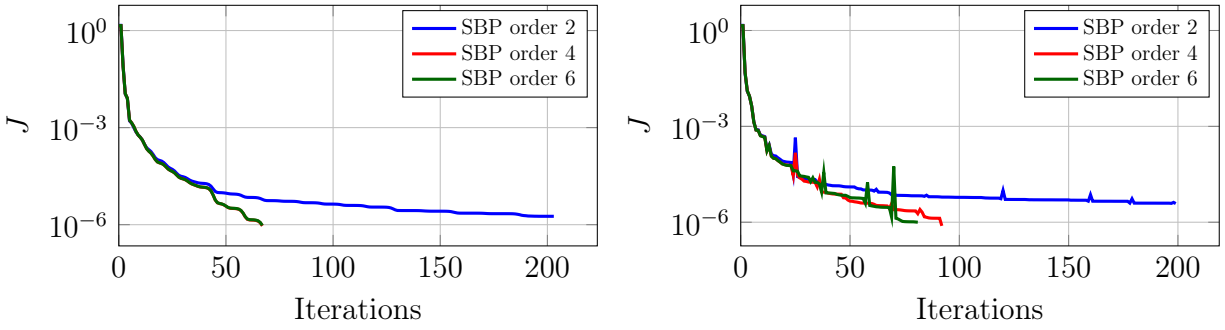
4.1.2.2 Constant profile



(a) BFGS

(b) Steepest descent

Figure 4.9: Solution for the constant profile of a, using the Ricker-wavelet source function, ERK integrator and the two optimisers.

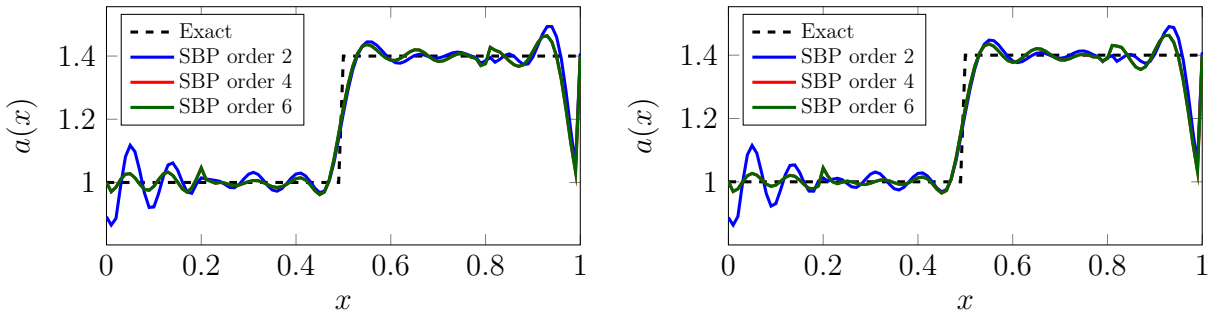


(a) BFGS

(b) Steepest descent

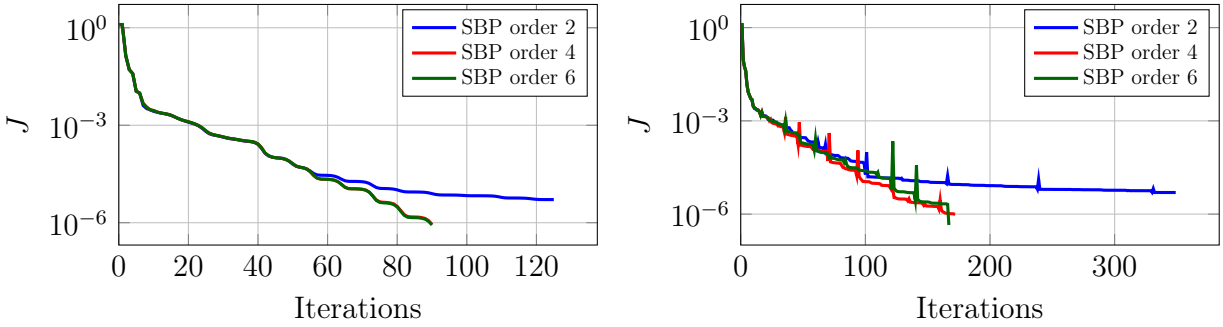
Figure 4.10: Number of iterations for the two optimisers using the constant profile of a, Ricker-wavelet source function and ERK integrator.

4.1.2.3 Discontinuous profile



(a) BFGS **(b)** Steepest descent

Figure 4.11: Solution for the discontinuous profile of a, using the Ricker-wavelet source function, ERK integrator and the two optimisers.



(a) BFGS **(b)** Steepest descent

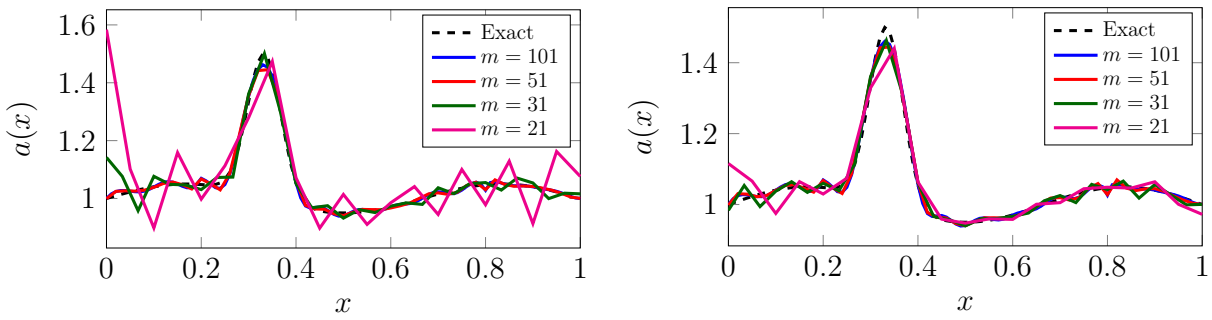
Figure 4.12: Number of iterations for the two optimisers using the discontinuous profile of a, Ricker-wavelet source function and ERK integrator.

4.2 Performance and accuracy

4.2.1 Order of accuracy of the SBP discretisation

The initial results from the benchmark cases, where we solve for a and compare to the exact solution from Figure 3.2, strongly indicate that the spatial accuracy of the method is essential for a convergent result given the set tolerance. The fourth and sixth order SBP operators result in identical solutions for every case, and the second order SBP operator performs worst in every case for our moderately sized mesh of $m = 101$ grid points.

In the case with the Gaussian source, smooth profile of $a(x)$ and the BFGS method with tolerance $\text{TOL} = 10^{-6}$, we compare the performance of the SBP operators for varying mesh sizes.



(a) SBP order 4 **(b)** SBP order 6

Figure 4.13: The solution for the smooth profile with the BFGS method with tolerance TOL $= 10^{-6}$ for decreasing mesh sizes.

For moderately larger mesh sizes, the second order SBP operator converged to a solution given the tolerance. The full result for all three orders of the operator are compiled in Table 4.1.

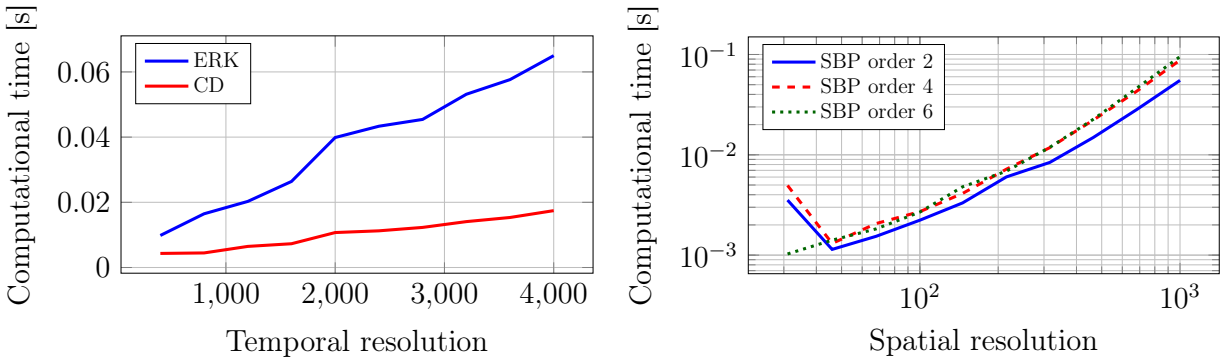
Table 4.1: The number of iterations for the BFGS method for different SBP operator orders given the number of spatial grid points. Iterations marked with "-" did not converge to the accepted tolerance of TOL $= 10^{-6}$.

Grid points	**Iterations**		
	SBP order 2	SBP order 4	SBP order 6
$m = 251$	126	97	97
$m = 201$	172	96	96
$m = 151$	-	95	95
$m = 101$	-	88	87
$m = 51$	-	85	83
$m = 31$	-	-	78
$m = 21$	-	-	-

4.2.2 Time complexity

The computational time for the problem depends on several factors. Among these are the implementation of the methods, the number of iterations for the optimisation algorithm to converge, the computational complexity of the temporal integrator and the differential operators. The computational time also depends on the performance of the system. For the following results we test the code on a system running Matlab R2021a on Windows 10 with

an Intel Core i5-7500 CPU at 3.40Ghz and 16GB RAM working at 2133MHz. The implementation is done according to Algorithm 1 and 2. Calculation and storage of variables is done using matrices as much as possible but in some cases loops are needed, for example in the temporal integration and gradient calculation. Since the point of this section is to give an indication of the overall performance of the base algorithm, details of the implementation is not needed.



(a) Temporal time complexity.

(b) Spatial time complexity.

Figure 4.14: Time complexity of the temporal and spatial resolutions. The ERK temporal integrator was on average 3.64 times slower than the CD temporal integrator.

Figure 4.14a and 4.14b show the execution time complexity when increasing the resolution in both the temporal and spatial part of the computational domain. Using a domain with 101 spatial points and 401 temporal points, the average time for one iteration using the BFGS method with the Gaussian source, smooth parameter profile and the two temporal integrators is measured. The average is found by running the code and repeating the computations fifty times with the same problem conditions. Each of the four parts of the algorithm is tested and the results are shown in Table 4.2 and 4.3.

Table 4.2: Average execution time per iteration for a domain of 101 spatial points and 401 temporal points using ERK temporal integrator.

	SBP		
	Order 2	Order 4	Order 6
Forward solver	0.0103s	0.0102s	0.0103s
Adjoint solver	0.3202s	0.2987s	0.2985s
Gradient calculation	0.0296s	0.0581s	0.0362s
Function evaluation	0.0006s	0.0006s	0.0006s
Total time	0.3606s	0.3677s	0.3456s

Table 4.3: Average execution time per iteration for a domain of 101 spatial points and 401 temporal points using the CD temporal integrator.

	SBP		
	Order 2	Order 4	Order 6
Forward solver	0.0045s	0.0053s	0.0048s
Adjoint solver	0.0894s	0.0951s	0.0883s
Gradient calculation	0.0278s	0.0603s	0.0345s
Function evaluation	0.0006s	0.0006s	0.0006s
Total time	0.1223s	0.1612s	0.1282s

5 Discussion and Conclusions

In this section, we discuss the behaviour of the algorithm and its performance, as well as the limitations of the algorithm and the advantages of using this type of gradient-based optimisation. Finally we give some conclusion and recommendations for future work.

The aim of this project is to solve the inverse model of the acoustic wave equation to find the unknown, spatially varying wave speed. This is done by minimising the cost functional defined by the residual between the PDE solution and measured data with respect to the wave speed parameter of the model. We find the gradient of the cost functional with respect to the wave speed, and through iteration, update the wave speed value with a gradient based optimisation algorithm. The computation of this gradient is done by defining the equivalent Lagrangian and reducing the complexity of the gradient expression by finding the adjoint to our original problem. The original problem is discretised and solved in space using the SBP-SAT method and the adjoint problem is derived in this semi-discrete form by using the summation-by-parts property of the SBP operator. Two different explicit, high order temporal integration schemes are then tested.

The algorithm is able to find the optimum value for the wave speed parameter with great speed. As seen in the result, both Steepest descent and the BFGS method are effective. Yet, for larger tolerances the BFGS outperforms Steepest descent in terms of convergence rate and therefore also computational time. It is also concluded that convergence can only be achieved with a moderately sized discrete mesh. The second order SBP-SAT scheme needs at least 201 spatial grid points to give a convergent solution for the wave speed parameter given by a smooth profile at a tolerance of TOL $= 10^{-6}$. As with the fourth order scheme, it needs at least 51 spatial grid points and the sixth order scheme only 31. This is a clear demonstration of the importance of using a high order spatial approximation, especially if computer resources are limited.

The original problem is tested with two different source term functions, but without any significance between the results. Some differences are however found in terms of performance. The algorithm using the Ricker wavelet source function need fewer iterations for convergence with an irregular profile of the parameter, as with the constant and discontinuous profile, compared to the Gaussian source function. For the smooth parameter profile, the sources have no large differences in performance.

When we looked at the performance of the code, the order of the spatial integration scheme does not affect the computational time substantially. Increasing or decreasing the resolution in time and space has as expected a larger effect, as well as using the CD temporal integration scheme instead of ERK. The CD scheme is almost four times faster on average than the ERK scheme, probably a result of the larger memory usage required to run the ERK scheme. Both schemes are of the fourth order and give equally accurate results, but the choice of temporal

integrator is a subject which could be analysed further. Especially comparing implicit versus explicit solvers in terms of accuracy and computational resources.

Another question is whether any particular bottlenecks slowed down the computations. This is related to implementation and the reader might want to review the code used in particular. As many different approaches for implementation are possible and the use of SBP-SAT is fairly straight forward following this report, the authors do not believe it is necessary to attach the actual code here. The choice of implementation do however affect the performance and this should be kept in consideration when comparing any other work to our results. The code was analysed with Matlab's own performance analyser which did not indicate any mayor bottlenecks. For any interest in this we therefore refer to Table (4.2) and (4.3) which shows which parts was most time consuming.

In conclusion, a successful algorithm for solving an inverse problem has been constructed. It utilises the adjoint problem of the original model to decrease the complexity of the computations and reduce computational time without affecting the accuracy of the results. Of course, these statements must be compared with results of other algorithms to give this algorithm a proper foundation. We therefore propose further studies with comparisons with this model to others. Other questions of interest are, optimal positions and optimal number of sources and receivers in the domain. In this project we found that at least two receivers are needed for our problem through trial and error, but this is not studied to any larger degree. We can therefore not motivate our choices through any good arguments. A natural follow-up of this project would be to extend the algorithm to solve two and three spatial dimensional problems and see if any other type of spatial discretisation scheme could be equally as effective.

Bibliography

[1] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, 2010.

[2] Stephen G. Nash Igor Griva and Ariela Sofer. *Linear and Nonlinear optimization*. Society for Industrial and Applied Mathematics, 2009.

[3] Heinz-Otto Kreiss and Godela Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. In Carl de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 195–212. Academic Press, Inc, Cambridge, Massachusetts, 1974.

[4] David Gottlieb Mark H. Carpenter and Saul Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes. *Journal of Scientific Computing*, 111(2):220–236, 1994.

[5] Ken Mattsson. Summation by parts operators for finite difference approximations of second-derivatives with variable coefficients. *Journal of Scientific Computing*, 51(51):650–682, 2012.

[6] Ken Mattsson and Jan Nordström. Summation by parts operators for finite difference approximations of second derivatives. *Journal of Scientific Computing*, 199(2):503–540, 2004.