Reliable computation of the eigenvalues of the discrete KdV spectrum

Prins, Peter J.; Wahls, Sander

DOI
10.1016/j.amc.2022.127361

Publication date
2022

Document Version
Final published version

Published in
Applied Mathematics and Computation

Citation (APA)
Prins, P. J., & Wahls, S. (2022). Reliable computation of the eigenvalues of the discrete KdV spectrum. *Applied Mathematics and Computation*, *433*, Article 127361. https://doi.org/10.1016/j.amc.2022.127361

Applied Mathematics and Computation 433 (2022) 127361

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Reliable computation of the eigenvalues of the discrete KdV spectrum

Peter J. Prins*, Sander Wahls

Delft Center for Systems and Control (DCSC), Delft University of Technology, Postbus 5, 2600 AA Delft, the Netherlands

ARTICLE INFO

Article history:
Received 22 October 2021
Accepted 25 June 2022

Keywords:
Korteweg–de Vries equation (KdV)
Nonlinear Fourier Transform (NFT)
Eigenvalue
Schrödinger equation
Sturm–Liouville equation
Sampled signal

ABSTRACT

We propose a numerical algorithm that computes the eigenvalues of the Korteweg–de Vries equation (KdV) from sampled input data with vanishing boundary conditions. It can be used as part of the Non-linear Fourier Transform (NFT) for the KdV equation. The algorithm that we propose makes use of Sturm Liouville (SL) oscillation theory to guaranty that all eigenvalues are found. In comparison to similar available algorithms, we show that our algorithm is more robust to numerical errors and thus more reliable. Furthermore we show that our root finding algorithm, which is based on the Newton–Raphson (NR) algorithm, typically saves computation time compared to the conventional approaches that rely heavily on bisection.

1. Introduction

The KdV is a non-linear Partial Differential Equation (PDE). It can serve al a model for a wide variety of nearly hyperbolic, weakly non-linear processes, such as surface waves in shallow water [1–5], internal waves in stratified fluids [6], acoustic waves in metals [7], electrical waves in transmission lines [8], traffic flow [9] and pressure waves in fluids [10–12]. See also [13] for a survey of some (more) applications of the KdV. The normalized form of the KdV which we consider is

$$\frac{\partial}{\partial t}q(x,t)+6q(x,t)\frac{\partial}{\partial x}q(x,t)+\frac{\partial^3}{\partial x^3}q(x,t)=0. \tag{1}$$

Equation (1) can be mapped to the various physical forms of the KdV with dimensional variables and coefficients by means of affine transformations of q, x and t [14, §1.2]. For simplicity, we can think of x and t as position and time respectively.

The KdV is the prototypical example of a Lax-integrable PDE. By that, we mean that its initial value problem can be solved with a technique that is called scattering transform or Non-linear Fourier Transform (NFT). It parallels the use of the ordinary Fourier transform for linear PDEs: The temporal evolution of $q(x,t)$ is hard to compute, but one can transform it back and forth to a so-called spectrum, of which the evolution is simple to compute [15, Sect. 1.4], [16]. The price to pay is the calculation of the direct and inverse NFT. At first, the NFT was an analytical method that allowed mathematicians to compute exact solutions of the KdV and other Lax-integrable PDEs [16–19]. Later, algorithms were developed to use the NFT

Abbreviations: FLOP, FLoating point OPeration; KdV, Korteweg–de Vries equation; NFT, Non-linear Fourier Transform; PDE, Partial Differential Equation; NR, Newton–Raphson; FFT, Fast Fourier Transform; SL, Sturm–Liouville; ODE, Ordinary Differential Equation.

* Corresponding author.

E-mail addresses: p.j.prins@tudelft.nl (P.J. Prins), s.wahls@tudelft.nl (S. Wahls).

https://doi.org/10.1016/j.amc.2022.127361
0096-3003/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license
(http://creativecommons.org/licenses/by/4.0/)

in numerical computations, when only a sampled input signal is available [20–28]. This enabled the use of the NFT as an analysis tool that reveals the physical structure of measured data from non-linear systems [2,6,29–35]. However, the early numerical methods were slow: Their computation times scaled quadratically in the number of samples. This motivated the development of fast numerical NFTs [36–38]. These scale almost linearly in the number of samples. Many recent publications present various improvements on aspects of the numerical NFT in computation time and accuracy, e.g. [39–50].

Before we can zoom in on the specific topic of this paper, we have to outline the mathematical problem that defines the NFT for the KdV, and introduce some terminology. The KdV-NFT spectrum of a signal $q(x,t)$ can be obtained at any fixed time $t = t_0$ from the one-dimensional Schrödinger equation[1,2]

$$\left(\frac{\partial^2}{\partial x^2} + q(x,t_0)\right) f(x,\kappa,t_0) = \kappa^2 f(x,\kappa,t_0). \tag{2}$$

The input signal $q(x,t_0)$ is also called *potential*, because of its role in the Schrödinger Eq. (2) in the context of quantum mechanics. We call any signal $f(x,\kappa,t_0)$ that satisfies (2) a *trajectory* of the potential $q(x,t_0)$. Here, we are concerned with real-valued potentials $q(x,t)$ that evolve according to the KdV (1) and furthermore satisfy the *vanishing boundary condition*

$$\int_{-\infty}^{\infty} |q(x,t_0)|(1+|x|)\,dx < \infty \text{ and } \lim_{|x|\to\infty} q(x,t_0) = 0. \tag{3}$$

The KdV-NFT spectrum of such a potential consists of two parts: A *continuous spectrum* and a *discrete spectrum*. The continuous spectrum describes a wave continuum called radiation. The discrete spectrum describes a countable number of wave components called solitons. It consists of *eigenvalues* and *norming constants*, one of each for every soliton. The eigenvalues are the values $\kappa = K_n > 0$ for which there exists a trajectory $f(x,\kappa,t_0)$ with finite energy[3] such that the Schrödinger Eq. (2) is satisfied. This trajectory $f(x,K_n,t_0)$ is the corresponding *eigenfunction*. Each eigenfunction is unique up to a scalar factor. The norming constants can be obtained from the eigenfunctions, as discussed e.g. in [40]. In this paper we address the computation of the eigenvalues.

In the literature two approaches can be found for the computation of the eigenvalues [51, §2.1], [28, §IV]:

The first approach is to use a *finite dimensional approximation* of (2) that turns its eigenproblem into a (large) matrix eigenproblem. Collocation methods (e.g. [26]) and rational approximations (e.g. [36]) belong to the first class. The computational complexity of these methods is at best $\mathcal{O}(D^2)$, where D is the number of degrees of freedom of the discretization. The accuracy of these methods quickly deteriorates for faster oscillating trajectories [51, Chap. 2].[4]

The second approach is known as the *shooting* approach. That is, one reduces the boundary value problem first to an initial value problem by keeping just one boundary condition. A free parameter, κ in our case, is introduced to make the initial value problem well-defined. Then one verifies with an initial value solver if the remainder of the boundary conditions is also satisfied. This procedure is iterated in a root finder that tries different values of the parameter until the boundary value problem is solved. Shooting methods can be implemented with a computational complexity of only $\mathcal{O}(DNP)$, where D is the number of samples, N is the number of eigenvalues and P is average required number of iterations per eigenvalue.

Basic implementations of the shooting approach cannot guarantee global convergence, so they may not find every eigenvalue. However, global convergence can be guaranteed by combining a shooting method with Sturm–Liouville (SL) oscillation theory. (This theory applies because the Schrödinger Eq. (2) is a specific example of a SL equation.) In short: According to the SL theory, the number of zero-crossings of the trajectory $f(x,\kappa,t_0)$ at a fixed value κ as x runs from $-\infty$ to $+\infty$, reveals the number of eigenvalues that is greater than κ. To make use of this information, we need to track the zero-crossings of the trajectory. From the Schrödinger equation Prüfer [52] derived a non-linear Ordinary Differential Equation (ODE) for the *phase* of the trajectory: the Prüfer equation. The number of zero-crossings follows trivially from the phase. The shooting method that relies on the integration of the Prüfer equation has become known as the Prüfer method [53,54]. However, the Prüfer equation is a stiff system which is hard to integrate [51, §2.1]. Alternatively, one can integrate the Schrödinger equation itself, which is simpler, and meanwhile count the number of zero-crossings [24,55,56]. In this paper we propose an algorithm that also uses this method for the computation of the eigenvalues.

The challenge within this method is to count the zero-crossings of the trajectory in a numerically robust way. Any missed or doubly counted zero-crossing can cause a significantly wrong result. Even causes for a miscount that may seem pathological at first sight, are surprisingly likely to occur in practice, because the root counting procedure is repeated D (samples) times N (eigenvalues) times P (iterations) times, and because the eigenfunctions that we search are themselves corner cases of the computation. Osborne [24] proposed to compute the number of zero-crossings by counting sign changes of the trajectory from sample to sample. However, Christov [57] observed numerical instabilities when applying Osborne's algorithm on certain simulation data. Pruess [55, §4] and Ixaru [56, §5.2] proposed a more rigorous counter, to cover the possibility that there is more than one zero-crossing between two consecutive samples. However, their approach may lead to a miscount if

[1] The reason for this is that the Schrödinger Eq. (2) is one half of a Lax-pair that constitutes the KdV equation [16–19].

[2] In most related literature one uses the parameter $\zeta \equiv \mathrm{j}\kappa$, where $\mathrm{j} := \sqrt{-1}$. Nevertheless, we use κ in this paper since it simplifies the exposition and keeps all computations in the real domain.

[3] The energy of $f(x)$ is defined as $\int_{-\infty}^{\infty} |f(x)|^2\,\mathrm{d}x$.

[4] If (2) and (3) are observed, $f(x,K_n,t_0)$ oscillates faster as K_n is closer to zero.

Table 1
Characteristics of two SL based algorithms to compute the eigenvalues of the Schrödinger equation found in the literature, and the proposed algorithm. Matslise 2.0 allows the specification of boundary conditions at $\pm\infty$, but replaces $\pm\infty$ internally by finite value before the computation.

	Osborne [24]	Matslise 2.0 [58]	Proposed
Input	Samples	Function	Samples
Boundary at $\pm\infty$	No	Only as input	Yes
Order	$\mathcal{O}(\varepsilon^2)$	$\mathcal{O}(\varepsilon^{18})$	$\mathcal{O}(\varepsilon^4)$
Root finder	Bisection	Bisection to bracket, refinement with NR	NR if possible, bisection for a new starting point

a zero-crossing occurs close to the boundary between two integration steps. In Section 3.2 we will clarify this vulnerability and explain how the proposed algorithm mends it.

Another aspect on which we improve on algorithms that are known in the literature, is the root finder. The use of SL theory provides upper and lower bounds on the eigenvalues. This information is most easily incorporated into a *bracketing* root finder, such as bisection. (A bracket consists of an upper and lower bound of a certain root.) Unfortunately, bisection has only linear convergence. Some algorithms speed up the convergence by using a two-stage approach: First, bisection is applied to obtain a (sufficiently tight) bracket for each root, which contains no other roots. Second, a root finder with a faster convergence is applied to refine the localization of the eigenvalue. The second stage could for example use regula falsi [57] or NR [24,56,58]. These algorithms are thus still limited to linear convergence in the first stage. Algorithms that apply an *open* root finder such as NR in the second stage then face the problem of finding an initial guess for which the iterations do not jump out of the just computed bracket. The algorithm that we propose in this paper uses a different approach. Since we typically need to compute all of the eigenvalues for the NFT, there is no need to isolate the eigenvalues upfront and to specify which one to localize next. Instead, we let the NR procedure converge to *any* eigenvalue. Thereafter, we select a new starting point and let NR converge to another eigenvalue. Meanwhile, we update the brackets of all the eigenvalues on every iteration of the NR procedure, based on SL theory. If the next NR iteration is not within the bracket of any of the eigenvalues, we switch to a new starting point within one of the brackets. This ensures that every iteration increases our knowledge about the location of at least one of the eigenvalues.

We briefly mention some other relevant aspects that distinguish different root finding algorithms. These aspects are summarized in Table 1 for the proposed algorithm and for the two algorithms which we use as benchmark algorithms.

- For practical applications of the KdV-NFT we need an algorithm that takes a sampled input signal. Some algorithms that were developed for other purposes, such as Matslise [58] require a functional description of the input signal instead. In our benchmark comparison we will work around this issue by fitting a Fourier series to the data.
- Some algorithms were developed for finding eigenvalues of the Schrödinger equation on a finite interval. The KdV-NFT for potentials that satisfy the vanishing boundary condition (3) requires boundary conditions at infinity. Not all algorithms support this. Matslise [58] allows the specification of boundary conditions at $\pm\infty$, but replaces $\pm\infty$ internally by a finite value before the computation. That means that not only the potential is truncated, but also the trajectories. In the NFT literature it is common practice to exploit the fact that trajectories of the Schrödinger equation can be written as the sum of two exponentials outside the support of the truncated potential. Therefore, no truncation of the trajectories is required. We also apply this in the proposed algorithm. We adapt this aspect of Osborne's algorithm [24] accordingly for the benchmark comparison in this paper.
- There exist many integrators that can be used to integrate the Schrödinger equation. If the potential is (piecewise) sufficiently smooth, integrators of higher order are more accurate, but computationally more expensive per integration step. Matslise [58] uses an 18th order integrator, but severely reduces the number of integration steps to trade a part of the accuracy gain for computational cost. We propose to use a specific fourth order integrator (see Section 3.4), because it allows for an accurate computation of the zero-crossings of the trajectory (see (14)).

The rest of this paper is organized as follows. In Section 2 we mention a few important aspects of the eigenvalues of the KdV and introduce two variables that play a fundamental role in the proposed algorithm to find the eigenvalues. These variables are the scattering parameter $a(\kappa)$ and the accounting function $s(-\infty,\infty,\kappa)$. In Section 3 we show how to compute these parameters numerically at a given value of κ. In Section 4 we propose a root finder that finds the eigenvalues by sampling these parameters, with a lower computational cost than bisection. In Section 5 we evaluate the proposed algorithm by comparing it with other methods on six different example signals. The paper is concluded in Section 6.

2. Preliminaries

Recall that the problem we aim to solve in this paper is the following. Given a uniformly sampled potential $q(x,t_0)$ that satisfies (3), find all the eigenvalues. We will search for the eigenvalues with the shooting method. That is, choose a value κ, determine if κ is an eigenvalue of $q(x,t_0)$ and repeat until all the eigenvalues are localized. In this section we will explain two strategies to determine if κ is an eigenvalue: The basic method and the one that makes use of SL oscillation theory.

We will also mention the numerical advantages and disadvantages of both strategies. In Section 3 we will explain how to perform the numerical computations for these strategies. In Section 4 we will integrate these strategies in one algorithm that combines the advantages of both.

2.1. Notation

Matrices have an upper case symbol in bold weight (e.g. $\mathbf{A}$), vectors have a lower case symbol in bold weight (e.g. $\boldsymbol{\phi}$), scalars have a normal weight (e.g. x). Constants have an upright style (e.g. π), real or imaginary variables have a slanted style (e.g. x), booleans are printed in a typewriter font (e.g. `f`). We will use super scripts l and u for respectively the lower and upper boundary value of an interval. $\mathcal{O}(\,)$ is used as the Landau 'big-O' order symbol. The symbol ':=' denotes a definition. The symbol '$\leftarrow$' means that the left hand side gets the value of the right hand side (at that point in an algorithm). The symbol '$\wedge$' denotes the logical AND operation. The notation '$|x|$' means the absolute value of x. The notation '$\lfloor x \rceil$' means rounding x towards the nearest integer,'$\lceil x \rceil$' means rounding x towards the nearest greater or equal integer, '$\lfloor x \rfloor$' means rounding x towards the nearest lesser or equal integer. We will make extensive use of the *Iverson bracket* for piecewise expressions: $[\![\mathtt{b}]\!] := 1$ if `b` is `true`, 0 otherwise. By convention, $x[\![\mathtt{false}]\!] = 0$, even if x is infinite or undefined. Single square brackets [ ] are used for the composition of vectors and matrices. Finally, both exp and $\mathrm{e}^{\mathbf{A}}$ stand for the natural scalar or matrix exponential function: $\exp(\mathbf{A}) \equiv \mathrm{e}^{\mathbf{A}} := \sum_{i=0}^{\infty} \mathbf{A}^i/(i!)$.

2.2. Eigenvalues of the Korteweg–de Vries equation

As mentioned in Section 1, the eigenvalues of the KdV equation are obtained from the Schrödinger Eq. (2). It can be shown that if $q(x,t)$ is real and evolves according to the KdV (1), then the eigenvalues are constant and isolated (with multiplicity one) [15,59], and that all eigenvalues[5] satisfy $0 < K_n^2 < sup_x\, q(x,t)$ (for all t) [60, p. 732]. We will index the eigenvalues such that $0 < K_1 < K_2 < \cdots < K_N < \sqrt{sup_x\, q(x,t)}$, where $N \geqslant 0$ is the number of eigenvalues. Since the eigenvalues are constant, we can simplify the notation by dropping the dependence on the arbitrary fixed time $t = t_0$ of other variables.

2.3. Scattering parameter $a(\kappa)$

Given any potential $q(x)$ that satisfies the vanishing boundary condition (3), it is readily verified that as $|x| \to \infty$ all trajectories of the Schrödinger Eq. (2) can be parametrized as a linear combination of $\exp(\pm\kappa x)$. We can thus define special trajectories that vanish as $x \to \pm\infty$ respectively. These trajectories are known as Jost solutions and satisfy the boundary conditions

$$\lim_{x\to-\infty} \phi(x,\kappa)\exp(-\kappa x) = 1 \tag{4}$$

$$\lim_{x\to+\infty} \psi(x,\kappa)\exp(+\kappa x) = 1. \tag{5}$$

Since the eigenfunctions have finite energy, they must vanish both as $x \to -\infty$ and as $x \to \infty$. Therefore every eigenfunction must satisfy both (4) and (5) up to an arbitrary constant factor: $\phi(x,K_n) \propto f(x,K_n) \propto \psi(x,K_n)$. We can use this insight to find the eigenvalues according to the shooting method. That is, we solve (2) for the boundary condition (4) at several values of κ. Then we check if $\phi(x,\kappa)$ satisfies 5 up to a scalar factor. To simplify this check, one defines the scattering parameter $a(\kappa)$ as follows.

$$a(\kappa) := \mathrm{W}[\psi,\phi]/(2\kappa), \quad \text{where} \quad \mathrm{W}[\psi,\phi] := \psi(x,\kappa)\frac{\partial\,\phi(x,\kappa)}{\partial x} - \phi(x,\kappa)\frac{\partial\,\psi(x,\kappa)}{\partial x}. \tag{6}$$

The Wronskian $\mathrm{W}[\psi,\phi]$, also known as the mismatch function [51, Eq. 2.14], vanishes if and only if the trajectories $\phi(x,\kappa)$ and $\psi(x,\kappa)$ are proportional. Hence, the eigenvalues K_n are the values κ for which $a(\kappa) = 0$. Usually one searches for the eigenvalues with a root finder that also makes use of the gradient $a'(\kappa) := \frac{\partial}{\partial\kappa} a(\kappa)$, for example NR. Initially, both a lower and an upper bound on the eigenvalues are known, see Section 2.2. These bounds can be used to guess suitable starting values for the root finder.

It is unreliable to compute the eigenvalues from samples of $a(\kappa)$ and $a'(\kappa)$ only, because it remains unknown how many eigenvalues there are. Suppose we have two adjacent, non-zero samples $a(\kappa_1)$ and $a(\kappa_2)$, we can only infer the parity of the number of eigenvalues between κ_1 and κ_2: The parity is odd if $a(\kappa_1)a(\kappa_2) < 0$ and even if $a(\kappa_1)a(\kappa_2) > 0$. In the odd case there must be at least one eigenvalue between these samples, which is useful information. However, in the even case the number of eigenvalues between these samples could be zero as well as any other even number. No matter how many samples are taken, there is always a possibility that one or more pairs of eigenvalues are missed. In practice, we would have to evaluate $a(\kappa)$ on a very fine κ-grid and hope that odd means one and even means zero. In that manner we can never be sure that we have localized all the eigenvalues. Additional information needs to be collected to ensure that no eigenvalues are missed. An attractive source of additional information is the accounting function, which we discuss next.

[5] By (3) $\sup_x q(x,t) \geq 0$. If $\sup_x q(x,t) = 0$, there are no eigenvalues, so the discrete spectrum is an empty set.

2.4. Accounting function

Since the Schrödinger Eq. (2) is an example of an SL equation, SL oscillation theory applies. From that theory, it is known that the number of solutions of $\phi(x,\kappa_0)=0$ for fixed κ_0 and finite real x is equal to the number of eigenvalues that is greater than κ_0 [61, Thms. 2.6.2 & 10.12.1.(4)]. We call these solutions zero-crossings for short.[6] Let us define the accounting function $s\left(x^{\mathrm{l}},x^{\mathrm{u}},\kappa\right)$ as the number of zero-crossings of $\phi(x,\kappa)$ in the open interval $x\in(x^{\mathrm{l}},x^{\mathrm{u}})$ at a fixed value κ. Then $s(-\infty,\infty,\kappa)$ is equal to the number of eigenvalues that is greater than κ. Since all eigenvalues are positive, the total number of eigenvalues is thus given by $N=s(-\infty,\infty,0)$. Since all the eigenvalues are smaller than $\sqrt{sup_x\, q(x)}$, we know a priori that $s(-\infty,\infty,\kappa)=0$ for $\kappa\geq\sqrt{sup_x\, q(x)}$. When κ is increased from zero to $\sqrt{sup_x\, q(x)}$, the value of the accounting function $s(-\infty,\infty,\kappa)$ is decremented by one whenever κ equals an eigenvalue. We can thus localize the eigenvalues by searching for the steps in $s(-\infty,\infty,\kappa)$. The value $s(-\infty,\infty,\kappa)$ for κ just above or just below each localized eigenvalue K_n, reveals the index n of that eigenvalue.

The advantage compared to searching for zero-crossings of $a(\kappa)$ is the following. Recall that if we have two adjacent, non-zero samples $a(\kappa_1)$ and $a(\kappa_2)$, we can only infer whether the number of eigenvalues between κ_1 and κ_2 is odd (if $a(\kappa_1)\,a(\kappa_2)<0$) or even (if $a(\kappa_1)\,a(\kappa_2)>0$). On the other hand, if we know $s(-\infty,\infty,\kappa_1)$ and $s(-\infty,\infty,\kappa_2)$, their difference reveals not just the parity of the number of eigenvalues between κ_1 and κ_2, but the number itself. There is no risk of missing any closely spaced eigenvalues if the κ-grid is too coarse.

Of course, the accounting function is only useful if we can reliably evaluate it in a numerical computation. To that end we have to detect the zero-crossings of $\phi(x,\kappa_0)$, where κ_0 is fixed. That may seem like a similar problem as detecting the roots of $a(\kappa)$: If we would evaluate $\phi(x,\kappa_0)$ on an x-grid and count the sign changes, we might miss pairs of zero-crossings between two adjacent samples. However, the Schrödinger equation allows us to reliably detect even multiple zero-crossings of $\phi(x,\kappa_0)$ between samples. Since $\phi(x,\kappa_0)$ is a trajectory of the Schrödinger Eq. (2), the 'speed' at which it oscillates (the slope of the Prüfer phase) is controlled by the potential $q(x)$. If the potential is well-behaved between samples, the oscillation is also well-behaved. For the proposed algorithm we will choose a *reconstruction* from the given samples, for which the Schrödinger equation has a piecewise analytic solution. This solution allows us to compute the number of zero-crossings piece by piece. We will discuss this further in Section 3.

The downside of using the accounting function is that it does not have a gradient to help finding the eigenvalues. Therefore in Section 4 we will apply a NR root finder on the scattering parameter $a(\kappa)$ and its gradient $a\prime(\kappa)$, and evaluate the accounting function in parallel in order to bracket each eigenvalue.

3. Integration of the Schrödinger equation

In this section we discuss the numerical computation of the scattering parameter $a(\kappa)$ and the accounting function $s(-\infty,\infty,\kappa)$ from the potential $q(x)$. In Section 4 we will incorporate these computations in an algorithm that finds the eigenvalues of the KdV equation efficiently and accurately.

3.1. The sampled and reconstructed potential

We assume that we do not know the true potential $q(x)$, but only a finite number of samples on a uniform x-grid. We will define a reconstruction of the potential, for which we can integrate the Schrödinger equation while keeping track of the number of zero-crossings. We denote the number of samples by D and the step size by ε. That is, if x_1 is the first grid point, then all the grid points are given by $x_d=x_1+(d-1)\varepsilon$, where $d\in\{1,2,\ldots,D\}$ and the known potential samples are $q_d:=q(x_d)$. For notational convenience, we define around each grid point an interval $(x_d^{\mathrm{l}},x_d^{\mathrm{u}}):=(x_d-\frac{\varepsilon}{2},x_d+\frac{\varepsilon}{2})$. Indeed, for $d<D$ it follows that $x_{d+1}^{\mathrm{l}}=x_d^{\mathrm{u}}$.

We will compute $a(\kappa)$ and $s(-\infty,\infty,\kappa)$ first for the simplest reconstruction of the potential from the samples q_d. That is, we use a piecewise constant reconstruction, $\hat{q}(x)$, by the midpoint rule. The known potential samples give no information about $q(x)$ for $x\notin(x_1^{\mathrm{l}},x_D^{\mathrm{u}})$. Therefore we set in the reconstruction $\hat{q}(x)=0$ for $x\notin(x_1^{\mathrm{l}},x_D^{\mathrm{u}})$. Hence, $\hat{q}(x):=\sum_{d=1}^{D}q_d[\![x_d^{\mathrm{l}}<x<x_d^{\mathrm{u}}]\!]$. (Please refer to Section 2.1 for the meaning of the Iverson bracket $[\![\ ]\!]$.) The eigenvalues of $\hat{q}(x)$ approximate those of $q(x)$ up to an error term $\mathcal{O}\left(\varepsilon^2\right)$. After demonstrating the computation for $\hat{q}(x)$, we discuss in Section 3.4 how to upgrade the method such that the error term reduces to $\mathcal{O}\left(\varepsilon^4\right)$ for sufficiently smooth potentials.

[6] All zeros are crossings, because if $\phi(x_0,\kappa_0)=0$ and $\frac{\partial}{\partial x}\phi(x,\kappa_0)\big|_{x=x_0}=0$ at the same fixed position x_0, then (2) implies $\phi(x,\kappa_0)=0\,\forall x$, which violates 4. Hence $\phi(x_0,\kappa_0)=0\Rightarrow\frac{\partial}{\partial x}\phi x,\kappa_0\big|_{x=x_0}\neq0$, so $\phi(x,\kappa_0)$ must change sign at $x=x_0$.

3.2. Numerical computation of the scattering parameter $a(\kappa)$ and the accounting function

In numerical computations, it is convenient to replace (6) by an algebraic expression. We do that by defining vector valued trajectories as $\boldsymbol{f}_{\mathrm{C}}(x,\kappa) := \left[f(x,\kappa) \quad \frac{\partial}{\partial x} f(x,\kappa)\right]^{\top}$. Then (6) is equivalent to

$$a(\kappa) \equiv \boldsymbol{\psi}_{\mathrm{C}}^{\top}(x,\kappa)\begin{bmatrix} 0 & \kappa^{-1} \\ -\kappa^{-1} & 0 \end{bmatrix}\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa)/2. \tag{7}$$

Both (6) and (7) hold for all x. Therefore, $a(\kappa)$ can be computed at any *matching point* x for which both $\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa)$ and $\boldsymbol{\psi}_{\mathrm{C}}(x,\kappa)$ are known. For simplicity, we will evaluate (7) at $x = x_D^{\mathrm{u}}$ in the analysis that follows.[7] Therefore we need to propagate $\phi(x,\kappa)$ as defined in (4) from $-\infty$ to x_D^{u}. We do that by exactly solving the Schrödinger Eq. (2) in every constant piece of the reconstruction of the potential defined in Section 3.1. Likewise, we propagate $\psi(x,\kappa)$ as defined in (5) back from ∞ to x_D^{u}.

Furthermore, we evaluate the accounting function. We will do that by summing the zero-crossings of $\phi(x,\kappa)$ in every piecewise constant interval of the reconstructed potential. That is,

$$s(-\infty,\infty,\kappa) = s\left(-\infty, x_1^{\mathrm{l}}, \kappa\right) + \left(\sum_{d=1}^{D} s\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\right)\right) + s(x_D^{\mathrm{u}}, \infty, \kappa). \tag{8}$$

When we evaluate the accounting function (8) numerically, we must treat the number zero carefully. Firstly, to determine the sign of $\phi(x,\kappa)$ near one of its zero-crossings, we must use the same intermediate value $\phi\left(x_d^{\mathrm{u}},\kappa\right)$ for the computation of $s\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\right)$ as for the computation of $\phi\left(x_{d+1}^{\mathrm{u}},\kappa\right)$ and further. Otherwise the sign of $\phi\left(x_d^{\mathrm{u}},\kappa\right)$ may differ between the two computation paths. We will come back to this in Eq. (14) and the discussion thereafter. Secondly, we must be careful to obtain a correct result in case any intermediate value $\phi\left(x_d^{\mathrm{u}},\kappa\right)$ equals exactly 0. We found that the simplest treatment is to *consider zero as a positive number*. That means that we count the crossings at 0^-, between zero and the smallest representable negative number. Since 0^- has no representation in finite precision, these crossings can numerically never lie exactly on an x grid point. This allowed us to write (8) as a summation of zero crossings in open rather than closed intervals.

We have to be careful with the case $\lim_{x\to\infty}\phi(x,\kappa) = 0$. Even if $\phi(x,\kappa)$ approaches zero from below as $x\to\infty$, we must not count this as a zero crossing. This might sound obvious at this point, but later, namely in (16), it will lead to one strict inequality '> 0', whereas we need '≥ 0' everywhere else, in accordance with the treatment of zero as a positive number. This exception is essential for the working of the algorithm when $a(K_n)$ is numerically equal to zero for any eigenvalue. It can be verified that $\lim_{x\to\infty}\phi(x,\kappa) = 0$ indicates that κ is an eigenvalue and that the eigenfunctions of K_N, K_{N-2},... approach zero from above as $x\to\infty$, whereas those of K_{N-1}, K_{N-3},... approach from below. By never counting $\lim_{x\to\infty}\phi(x,\kappa) = 0$ as a zero-crossing, we obtain a consistent behaviour of the accounting function at its steps. Namely, $s(-\infty,\infty,K_n) := \lim_{\kappa\downarrow K_n} s(-\infty,\infty,\kappa) \equiv \lim_{\kappa\uparrow K_n} s(-\infty,\infty,\kappa) - 1$, where $\downarrow$ denotes the limit from above and $\uparrow$ denotes the limit from below.

3.2.1. The lower tail: $x \in (-\infty, x_1^{\mathrm{l}})$

In the interval $x \in (-\infty, x_1^{\mathrm{l}})$ we know that $\phi(x,\kappa)$ satisfies the Schrödinger Eq. (2) and the boundary condition (4). The reconstructed potential $\hat{q}(x)$ is zero in this interval. It is readily verified that the solution in this interval is $\phi(x,\kappa) \equiv \exp(\kappa x)$. The solution has thus no zero-crossings in this interval, that is, $s\left(-\infty, x_1^{\mathrm{l}}, \kappa\right) = 0$. At the boundary of the next interval we find

$$\boldsymbol{\phi}_{\mathrm{C}}\left(x_1^{\mathrm{l}},\kappa\right) = \begin{bmatrix}1 & \kappa\end{bmatrix}^{\top} \exp\left(\kappa x_1^{\mathrm{l}}\right). \tag{9}$$

3.2.2. The support: $x \in (x_1^{\mathrm{l}}, x_D^{\mathrm{u}})$

The computations in this interval are similar to those for the periodic boundary condition algorithm of [24], but improve on it by a more robust computation of the accounting function. This interval consists of D adjacent subintervals $(x_d^{\mathrm{l}}, x_d^{\mathrm{u}})$. In each of these subintervals the potential is constant. Therefore the Schrödinger Eq. (2) in the dth subinterval simplifies to $\frac{\partial^2}{\partial x^2}\phi(x,\kappa) = \kappa^2 - q_d$. The vector-valued Jost solution thus satisfies

$$\frac{\partial}{\partial x}\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) = \mathbf{A}_{\mathrm{C}}(q_d,\kappa)\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa), \text{ where } \mathbf{A}_{\mathrm{C}}(q_d,\kappa) := \begin{bmatrix} 0 & 1 \\ \kappa^2 - q_d & 0 \end{bmatrix}. \tag{10}$$

By solving (10) subject to the boundary condition at $x = x_d^{\mathrm{l}}$, we find at $x = x_d^{\mathrm{u}} = x_{d+1}^{\mathrm{l}}$ that

$$\boldsymbol{\phi}_{\mathrm{C}}\left(x_d^{\mathrm{u}},\kappa\right) = \mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}},\kappa\right)\boldsymbol{\phi}_{\mathrm{C}}\left(x_d^{\mathrm{l}},\kappa\right), \text{where the change of state matrix } \mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}},\kappa\right) \text{ equals} \tag{11}$$

[7] If the numerical representation of the trajectories causes an overflow during the computation, a different choice of x is a possible workaround. However, one could also solve it by rescaling the trajectory by a suitable non-zero scalar factor c, i.e. $\boldsymbol{f}_{\mathrm{C}}(x,\kappa) \leftarrow c\,\boldsymbol{f}_{\mathrm{C}}(x,\kappa)$, and look for the roots of $c\,a(\kappa)$), and/or, if $\kappa \neq 0$, by choosing a more suitable basis for the representation of the trajectories. Cf. [41, §3].

$$e^{\varepsilon \mathbf{A}_{\mathrm{C}}(q_d,\kappa)} = \begin{bmatrix} \cos(\gamma\varepsilon) & \varepsilon \operatorname{sinc}(\gamma\varepsilon) \\ -\gamma\sin(\gamma\varepsilon) & \cos(\gamma\varepsilon) \end{bmatrix}, \text{where } \gamma := \sqrt{q_d-\kappa^2} \text{ and } \operatorname{sinc}(\gamma\varepsilon) := \tfrac{\sin(\gamma\varepsilon)}{\gamma\varepsilon}[\![\gamma\varepsilon \neq 0]\!] + [\![\gamma\varepsilon = 0]\!]. \tag{12}$$

We remark that γ is either real or imaginary, but $\mathbf{H}_{\mathrm{C}}\big(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\big)$ is always real.

To count the number of zero-crossings of $\phi(x,\kappa)$ in the dth subinterval, it is not sufficient in general to look only at the signs of $\phi\big(x_d^{\mathrm{u}},\kappa\big)$ and $\phi\big(x_d^{\mathrm{l}},\kappa\big)$ as in [24]. The simplest way to see this, is by looking at the structure of the closed form solution of $\phi(x,\kappa)$ in the dth subinterval. That is,

$$\phi(x,\kappa) = \begin{cases} c_1 \exp(-x\sqrt{\kappa^2-q_d}) + c_2\exp(x\sqrt{\kappa^2-q_d}) & q_d-\kappa^2 < 0, \\ c_1 + c_2 x & q_d - \kappa^2 = 0, \\ c_1\sin(c_2 + x\sqrt{q_d-\kappa^2}) & q_d-\kappa^2 > 0; \end{cases} \tag{13}$$

where c_1 and c_2 are real constants. If $q_d-\kappa^2 \le 0$ (the non-oscillatory case) we see from (13) that the number of zero-crossings in the dth interval is either zero or one. Under that condition the number of zero-crossings can be determined reliably by comparing the signs of $\phi\big(x_d^{\mathrm{l}},\kappa\big)$ and $\phi\big(x_d^{\mathrm{u}},\kappa\big)$. However, the oscillatory case $q_d-\kappa^2 > 0$ requires a more careful computation, because there could be more than one zero-crossing in the dth interval. In the oscillatory case we need to look at the propagation of the phase of the sine in (13) across the dth interval, and compute the number of zero-crossings accordingly. If $0 < \varepsilon^2(q_d-\kappa^2) < \pi^2$, the number of zero-crossings is still at most one. Hence, both ways of counting $s(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa)$ are valid in this domain. Sign comparison is computationally cheaper, but switching between the two computations exactly at $\varepsilon^2(q_d-\kappa^2) = \pi^2$ is numerically not robust. Therefore we choose to switch at $\varepsilon^2(q_d-\kappa^2) = 3^2$. Hence, we count the number of zero-crossings in the interval $(x_d^{\mathrm{l}}, x_d^{\mathrm{u}})$ as

$$s\big(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\big) = \begin{cases} \Big|[\![\phi\big(x_d^{\mathrm{u}},\kappa\big) \ge 0]\!] - [\![\phi\big(x_d^{\mathrm{l}},\kappa\big)\ge 0]\!]\Big| & (\varepsilon\gamma)^2 < 9, \\ [\![\phi\big(x_d^{\mathrm{u}},\kappa\big)\ge 0]\!] - [\![\phi\big(x_d^{\mathrm{l}},\kappa\big)\ge 0]\!] + 2\big\lfloor\big(\varepsilon\gamma - \theta\big(x_d^{\mathrm{u}}\big) + \theta\big(x_d^{\mathrm{l}}\big)\big)/(2\pi)\big\rceil & (\varepsilon\gamma)^2 \ge 9 \end{cases} \tag{14}$$

where $\gamma = \sqrt{q_d-\kappa^2}$, $\theta(x) := \operatorname{atan2}\big(\gamma\,\phi(x,\kappa), \tfrac{\partial}{\partial x}\phi(x,\kappa)\big)$, and '$\lfloor\ \rceil$' means 'round to the nearest integer'. The four-quadrant arctangent is defined for $(y,z)\neq(0,0)$ by

$$\operatorname{atan2}(y,z) := \begin{cases} \operatorname{atan}\big(\tfrac{y}{z}\big) + \pi[\![z<0]\!]([\![y\ge 0]\!] - [\![y<0]\!]) & z\neq 0, \\ \tfrac{\pi}{2}([\![y\ge 0]\!] - [\![y<0]\!]) & z = 0 \wedge y \neq 0, \end{cases} \tag{15}$$

The values that are needed to evaluate (14) are thus q_d, κ, $\boldsymbol{\phi}_{\mathrm{C}}(x_d^{\mathrm{l}},\kappa)$ and $\boldsymbol{\phi}_{\mathrm{C}}(x_d^{\mathrm{u}},\kappa)$ as computed with (11). Note that it is important to follow the definitions above carefully when either $\phi\big(x_d^{\mathrm{l}},\kappa\big) = 0$, or $\phi\big(x_d^{\mathrm{u}},\kappa\big) = 0$. That is, zero counts as a positive number.

Let us highlight the merit of (14) in comparison to the literature. When we use the accounting function $s(-\infty,\infty,\kappa)$ in the search for eigenvalues, we sample it at different values of κ. The accounting function is a non-increasing staircase function of κ. This fact is implicitly exploited while bracketing the eigenvalues. Therefore, if a numerical evaluation of the accounting function is off by only ± 1, the perception of the accounting function will be very different. The error will often remain unnoticed because the implicit assumption of a non-increasing staircase prevents the algorithm from taking suitable samples for that. Hence, *it is essential to count every zero-crossing exactly once.* As mentioned before, Osborne [24] relies on the cheap sign check in the upper line of (14), for every value of $(\varepsilon\gamma)^2$. This may lead to a miscount if $\pi^2 \le (\varepsilon\gamma)^2 < (2\pi)^2$ and will surely lead to a miscount if $(\varepsilon\gamma)^2 \ge (2\pi)^2$. Pruess [55, §4], Ixaru [56, §5.2], and Ledoux [62, §4.2] distinguish between the non-oscillatory case $(\varepsilon\gamma)^2 \le 0$ where the cheap sign check suffices, and the oscillatory case $(\varepsilon\gamma)^2 > 0$ where they all use a more expensive computation. However, the step size ε of a sampled input signal is usually small compared to the fastest oscillation in any of the trajectories in practice, in order to have a sufficiently accurate representation of that signal. Therefore, most of the oscillatory case samples will be in the interval $0 < (\varepsilon\gamma)^2 < 9$, for which we can safely use the cheap sign check. *Our computation is thus more efficient in this respect.*

Furthermore, the oscillatory case computation that is used by [55, §4] is not robust when a zero-crossing occurs close to x_d^{u}. The problem is that the sign of $\phi\big(x_d^{\mathrm{u}},\kappa\big)$ is implicitly computed twice: First from $\phi\big(x_d^{\mathrm{l}},\kappa\big)$ and $\varepsilon\gamma$ to count the zeros in the dth subinterval and second with (11) to obtain the initial condition for the next subinterval, $\phi\big(x_{d+1}^{\mathrm{l}},\kappa\big)$. If there is a zero-crossing near x_d^{u}, then due to numerical inaccuracies one computation may end up just above zero while the other ends up just below zero. The consequence is then that this zero-crossing is counted either twice or not at all. We instead compute $\phi\big(x_d^{\mathrm{u}},\kappa\big) = \phi\big(x_{d+1}^{\mathrm{l}},\kappa\big)$ once, with (11). As indicated in the bottom case of (14), we only use $\varepsilon\gamma$ to count the number of full oscillations (with two zero-crossings each) that remains after accounting for the initial phase angle $\theta(x_d^{\mathrm{l}})$ and final phase angle $\theta(x_d^{\mathrm{u}})$. *This ensures that with our computation a zero-crossing near $x_d^{\mathrm{u}} = x_{d+1}^{\mathrm{l}}$ is always counted either in the dth or in the $(d+1)$th subinterval.*

The computation for the oscillatory case that is used by [56, §5.2] and [62, §4.2] also makes use of $\phi\big(x_d^{\mathrm{u}},\kappa\big)$, but it has another vulnerability. Their idea is to use $\phi\big(x_d^{\mathrm{u}},\kappa\big)$ to add a small correction to the phase propagation $\varepsilon\gamma$. However the way their equations handle the branch cuts of the (single variable) arctangent function is not numerically robust. These branch cuts occur in their case (but in our notation) when $\frac{\partial}{\partial x}\phi\big(x_d^{\mathrm{l}},\kappa\big) = 0$ and when $\frac{\partial}{\partial x}\phi\big(x_d^{\mathrm{u}},\kappa\big) = 0$. The compensation for these branch cuts takes place when $\vartheta(x_d^{\mathrm{u}}) - (\varepsilon\gamma + \vartheta(x_d^{\mathrm{l}})) - \pi\lfloor \frac{1}{\pi}(\varepsilon\gamma + \vartheta(x_d^{\mathrm{l}}))\rceil = \pm\frac{\pi}{2}$, where $\vartheta(x) := \operatorname{atan}(\gamma\phi(x,\kappa)/(\frac{\partial}{\partial x}\phi(x,\kappa)))$. Since this requires *different* comparisons between floating point numbers, rounding errors will very likely cause glitches near the branch cuts. Equation (14) instead uses $\phi\left(x_d^{\mathrm{u}},\kappa\right)$ in essence to compute a ‘signed parity’ of the number of zero-crossings and then adds as the third term a ‘correction’ for the number of full cycles based on $\varepsilon\gamma$ minus a correction for the angles $\theta\left(x_d^{\mathrm{u}}\right) \in (-\pi,\pi]$ and $\theta\left(x_d^{\mathrm{l}}\right) \in (-\pi,\pi]$.[8] The merit is that our branch cut compensation always checks, up to multiplication by γ, the sign of the *same* floating point number that causes the branch cut.[9] Hence, if $\phi\left(x_d^{\mathrm{l}},\kappa\right)$ is perturbed near zero, then $\theta\left(x_d^{\mathrm{l}}\right)$ and $[\![\phi(x_d^{\mathrm{l}},\kappa) \geq 0]\!]$ in (14) are guaranteed to flip simultaneously. Likewise, if $\phi\left(x_d^{\mathrm{u}},\kappa\right)$ is perturbed near zero, then $\theta(x_d^{\mathrm{u}})$ and $[\![\phi(x_d^{\mathrm{u}},\kappa) \geq 0]\!]$ are guaranteed to flip simultaneously. *This ensures that with our computation, no glitches occur near the branch cuts of the four-quadrant arctangent.*

3.2.3. *The higher tail: $x \in (x_D^{\mathrm{u}},\infty)$*

In the interval $(x_D^{\mathrm{u}},\infty)$ the reconstructed potential is zero. We first solve the Schrödinger Eq. (2) for the boundary condition $\boldsymbol{\phi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa)$ in order to find the number of zero-crossings. Since $q(x) = 0 \leq \kappa^2$, we see from (13) that the number of zero-crossings in this interval is either zero or one. This number can be computed by comparing the sign of $\phi\left(x_D^{\mathrm{u}},\kappa\right)$ to the sign of $\lim_{x\to\infty}\phi(x,\kappa)$ as follows.

$$s(x_D^{\mathrm{u}},\infty,\kappa) = [\![[1 \quad 0]\boldsymbol{\phi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) < 0 \wedge [\kappa \quad 1]\boldsymbol{\phi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) > 0]\!] + [\![[1 \quad 0]\boldsymbol{\phi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) \geq 0 \wedge [\kappa \quad 1]\boldsymbol{\phi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) < 0]\!] \tag{16}$$

As discussed earlier in this section, the condition $[\kappa \quad 1]\boldsymbol{\phi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) > 0$ is the only exception we must make to counting zero as a positive number, to obtain a consistent behaviour of the accounting function at the eigenvalues.

Next, we need to compute $\boldsymbol{\psi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa)$, so that we can compute $a(\kappa)$ from (7) at $x = x_D^{\mathrm{u}}$. We thus solve the Schrödinger Eq. (2) with a potential of zero for the boundary condition (5). It is readily verified that the solution in this interval is $\psi(x,\kappa) \equiv \exp(-\kappa x)$. Hence we obtain at $x = x_D^{\mathrm{u}}$

$$\boldsymbol{\psi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) = \begin{bmatrix}1 & -\kappa\end{bmatrix}^\top \exp\left(-\kappa x_D^{\mathrm{u}}\right). \tag{17}$$

3.3. *Numerical computation of the gradient of the scattering parameter $a(\kappa)$*

In order to find the roots of $a(\kappa)$, we will make use of the gradient $a'(\kappa) := \frac{d}{d\kappa}a(\kappa)$. If we take the derivative of (7) with respect to κ, we find

$$\begin{bmatrix} a(\kappa) \\ a'(\kappa) \end{bmatrix} \equiv \frac{1}{2}\begin{bmatrix} \psi_{\mathrm{C}}^\top(x,\kappa) & \begin{bmatrix}0 & 0\end{bmatrix} \\ \frac{\partial}{\partial\kappa}\psi_{\mathrm{C}}^\top(x,\kappa) & \psi_{\mathrm{C}}^\top(x,\kappa) \end{bmatrix} \begin{bmatrix} 0 & \kappa^{-1} & 0 & 0 \\ -\kappa^{-1} & 0 & 0 & 0 \\ 0 & -\kappa^{-2} & 0 & \kappa^{-1} \\ \kappa^{-2} & 0 & -\kappa^{-1} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) \\ \frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) \end{bmatrix}, \tag{18}$$

where we have used the convenient formulation of the scalar derivative of matrix products found in [23]. For the computation of $a'(\kappa)$ we need to extend the equations in Section 3.2 as follows. The derivatives with respect to κ of (9) and (17) are respectively

$$\frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}\left(x_1^{\mathrm{l}},\kappa\right) = \begin{bmatrix}1 & \kappa\end{bmatrix}^\top x_1^{\mathrm{l}} \exp\left(\kappa x_1^{\mathrm{l}}\right); \qquad \frac{\partial}{\partial\kappa}\boldsymbol{\psi}_{\mathrm{C}}(x_D^{\mathrm{u}},\kappa) = \begin{bmatrix}-1 & \kappa\end{bmatrix}^\top x_D^{\mathrm{u}} \exp\left(-\kappa x_D^{\mathrm{u}}\right). \tag{19}$$

Then $\frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa)$ needs to be propagated from $x = x_1^{\mathrm{l}}$ to $x = x_D^{\mathrm{u}}$. Following [23,32] we augment (10) to

$$\frac{\partial}{\partial x}\begin{bmatrix} \boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) \\ \frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) \end{bmatrix} = \widetilde{\mathbf{A}}_{\mathrm{C}}(q_d,\kappa)\begin{bmatrix} \boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) \\ \frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}(x,\kappa) \end{bmatrix}, \text{ where } \widetilde{\mathbf{A}}_{\mathrm{C}}() := \begin{bmatrix} \mathbf{A}_{\mathrm{C}}() & \begin{bmatrix}0 & 0 \\ 0 & 0\end{bmatrix} \\ \frac{\partial}{\partial\kappa}\mathbf{A}_{\mathrm{C}}() & \mathbf{A}_{\mathrm{C}}() \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \kappa^2 - q_d & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 2\kappa & 0 & \kappa^2 - q_d & 0 \end{bmatrix}. \tag{20}$$

$$\text{Then } \begin{bmatrix} \boldsymbol{\phi}_{\mathrm{C}}\left(x_d^{\mathrm{u}},\kappa\right) \\ \frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}\left(x_d^{\mathrm{u}},\kappa\right) \end{bmatrix} = \widetilde{\mathbf{H}}_{\mathrm{C}}(x_d^{\mathrm{l}},x_d^{\mathrm{u}},\kappa)\begin{bmatrix} \boldsymbol{\phi}_{\mathrm{C}}\left(x_d^{\mathrm{l}},\kappa\right) \\ \frac{\partial}{\partial\kappa}\boldsymbol{\phi}_{\mathrm{C}}\left(x_d^{\mathrm{l}},\kappa\right) \end{bmatrix} \text{ (cf. (11))}, \tag{21}$$

where the augmented change of state matrix $\widetilde{\mathbf{H}}_{\mathrm{C}}(x_d^{\mathrm{l}},x_d^{\mathrm{u}},\kappa) := \exp(\varepsilon\widetilde{\mathbf{A}}_{\mathrm{C}}(q_d,\kappa))$ equals

$$\begin{bmatrix} \mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}},x_d^{\mathrm{u}},\kappa\right) & \begin{bmatrix}0 & 0 \\ 0 & 0\end{bmatrix} \\ \frac{\partial}{\partial\kappa}\mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}},x_d^{\mathrm{u}},\kappa\right) & \mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}},x_d^{\mathrm{u}},\kappa\right) \end{bmatrix} = \begin{bmatrix} \cos(\gamma\varepsilon) & \varepsilon\operatorname{sinc}(\gamma\varepsilon) & 0 & 0 \\ -\gamma\sin(\gamma\varepsilon) & \cos(\gamma\varepsilon) & 0 & 0 \\ \kappa\varepsilon^2\operatorname{sinc}(\gamma\varepsilon) & \kappa\varepsilon\gamma^{-2}(\operatorname{sinc}(\gamma\varepsilon) - \cos(\gamma\varepsilon)) & \cos(\gamma\varepsilon) & \varepsilon\operatorname{sinc}(\gamma\varepsilon) \\ \kappa\varepsilon(\operatorname{sinc}(\gamma\varepsilon) + \cos(\gamma\varepsilon)) & \kappa\varepsilon^2\operatorname{sinc}(\gamma\varepsilon) & -\gamma\sin(\gamma\varepsilon) & \cos(\gamma\varepsilon) \end{bmatrix}. \tag{22}$$

[8] That is, the number between the rounding brackets $\lfloor\rceil$ in (14) should be an integer already, up to numerical error.

[9] Numerical multiplication by γ will not change the sign, since $\gamma > 3/\varepsilon \gg 0$, unless the data is poorly normalized.

Table 2
Count of the amount of FLOPs that is required per x-sample to compute $a(\kappa)$ at one sample κ, and the extra amount for computing also the gradient $a\prime(\kappa)$. The underlined expressions should be interpreted as a single variable with a known quantity. Initially, only the potential sample q_d, the spatial step size ε, and the spectral parameter κ are known. The upper part of the table regards the computation of $\mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\right)$ according to (12), and the multiplication in (11). This appears to take 35 FLOPs. The lower part of the table regards the extra operations for the computation of $\widetilde{\mathbf{H}}_{\mathrm{C}}\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\right)$ according to (22), and the multiplication in (21). Once the values of the upper part of the table are known, this appears to take 24 additional FLOPs. Hence, the whole computation of $a(\kappa)$, $s(-\infty,\infty,\kappa)$ and $a\prime(\kappa)$ takes 59 FLOPs per x-sample, per κ sample. The overhead, that does not depend on the number of samples, is ignored. The number of FLOPs that are required for each basic operation, are obtained from [63, p. 5]. In practice FLOP counts vary between different implementations, programming languages, compilers and hardware architectures. Therefore, these results should be treated as rough estimates.

	Calculation		Operation(s)	FLOPs
$\underline{\gamma^2}$	$\leftarrow$	$\underline{q_d} - \underline{\kappa} \cdot \underline{\kappa}$	1 $\times$ and 1 $\pm$	2
$\underline{\gamma}$	$\leftarrow$	$\sqrt{\underline{\gamma^2}}$	1 $\surd$	4
$\underline{\gamma\varepsilon}$	$\leftarrow$	$\underline{\gamma} \cdot \underline{\varepsilon}$	1 $\times$	1
$\underline{\cos(\gamma\varepsilon)}$	$\leftarrow$	$\cos\left(\underline{\gamma\varepsilon}\right)$	1 cos	8
$\underline{\sin(\gamma\varepsilon)}$	$\leftarrow$	$\sin\left(\underline{\gamma\varepsilon}\right)$	1 sin	8
$\underline{\varepsilon\,\mathrm{sinc}(\gamma\varepsilon)}$	$\leftarrow$	$\underline{\sin(\gamma\varepsilon)}/\underline{\gamma}$	1 $\div$	4
$\underline{-\gamma\sin(\gamma\varepsilon)}$	$\leftarrow$	$-\underline{\gamma} \cdot \underline{\sin(\gamma\varepsilon)}$	1 $\times$ and 1 $\pm$	2
	Right-multiply a 2×2 matrix by a vector		4 $\times$ and 2 $\pm$	6
	Total FLOPs per κ-sample per x-sample to compute $a(\kappa)$:			35
$\underline{\kappa\varepsilon}$	$\leftarrow$	$\underline{\kappa} \cdot \underline{\varepsilon}$	1 $\times$	1
$\underline{\kappa\varepsilon^2\,\mathrm{sinc}(\gamma\varepsilon)}$	$\leftarrow$	$\underline{\kappa\varepsilon} \cdot \underline{\varepsilon\,\mathrm{sinc}(\gamma\varepsilon)}$	1 $\times$	1
$\underline{\kappa\varepsilon\,\mathrm{sinc}(\gamma\varepsilon)}$	$\leftarrow$	$\underline{\kappa} \cdot \underline{\varepsilon\,\mathrm{sinc}(\gamma\varepsilon)}$	1 $\times$	1
$\underline{\kappa\varepsilon\cos(\gamma\varepsilon)}$	$\leftarrow$	$\underline{\kappa\varepsilon} \cdot \underline{\cos(\gamma\varepsilon)}$	1 $\times$	1
$\underline{\kappa\varepsilon(\mathrm{sinc}(\gamma\varepsilon) + \cos(\gamma\varepsilon))}$	$\leftarrow$	$\underline{\kappa\varepsilon\mathrm{sinc}(\gamma\varepsilon)} + \underline{\kappa\varepsilon\cos(\gamma\varepsilon)}$	1 $\pm$	1
$\underline{\kappa\varepsilon\gamma^{-2}(\mathrm{sinc}(\gamma\varepsilon) - \cos(\gamma\varepsilon))}$	$\leftarrow$	$\left(\underline{\kappa\varepsilon\,\mathrm{sinc}(\gamma\varepsilon)} - \underline{\kappa\varepsilon\cos(\gamma\varepsilon)}\right)/\underline{\gamma^2}$	1 $\pm$ and 1 $\div$	5
	Right-multiply a 2×4 matrix by a vector		8 $\times$ and 6 $\pm$	14
	Extra FLOPs per κ-sample per x-sample to compute also the gradient $a'(\kappa)$:			24

A root finder that makes use of the gradient usually requires fewer iterations, but each iteration is computationally costlier. In order to determine if it pays off in this case to use the gradient, we need to determine if the first outweighs the latter. In Table 2 we have therefore counted the number of FLoating point OPerations (FLOPs) to compute respectively the two by two matrix $\mathbf{H}_{\mathrm{C}}(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa)$ and the matrix vector multiplication in (11), or the four by four matrix $\widetilde{\mathbf{H}}_{\mathrm{C}}\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\right)$ and the matrix vector multiplication in (21). Since the number of repetitions of this part of the computation scales proportionally to the (usually large) number of samples D, this part dominates the computational cost of evaluating (7) or (18) respectively. For the basic operations in Table 2 we assumed the number of FLOPs that was estimated in [63, p. 5]. In practice FLOP counts vary between different implementations, programming languages, compilers and hardware architectures. Therefore, these results should be treated as rough estimates. The results in Table 2 show that an evaluation of $a(\kappa)$ alone takes roughly $35D$ FLOPs plus overhead. The computation of $a\prime(\kappa)$ at the same value of κ takes roughly $24D$ FLOPs extra. Hence, we estimate that every iteration of a gradient based root finder is roughly 70% more expensive than an iteration of a gradient free root finder.

3.4. Upgrade to fourth order accuracy

In Section 3.2 we approximated the scattering parameter $a(\kappa)$ of a potential $q(x)$. Thereto we used an exact computation (in infinite precision) for the reconstructed potential $\hat{q}(x)$, defined in Section 3.1. This piecewise constant reconstruction enabled the relatively simple computation in (11) and (12). With respect to the true potential $q(x)$, (11) implements the exponential midpoint rule, which is also known as $\mathrm{CF}_1^{[2]}$.[10] The exponential midpoint rule provides an approximation of order two in the step size [64, p. 244]. Consequently, we obtain from (7) $a(\kappa) + \mathcal{O}(\varepsilon^3)$. The same considerations and error order apply to the computation of the gradient $a\prime(\kappa)$ according to (18) in Section 3.3.

If the potential is sufficiently smooth, a more accurate reconstruction of $a(\kappa)$ and $a\prime(\kappa)$ can be obtained by using a higher order integrator. In particular, we will use the fourth order integrator $\mathrm{CF}_2^{[4]}$ [64, Eq. 12]:

$$\mathbf{H}_{\mathrm{C}}\left(x_d^{\mathrm{l}}, x_d^{\mathrm{u}}, \kappa\right) := \mathrm{e}^{\mathbf{A}_{\mathrm{C}}(\check{q}_{2d},\kappa)\varepsilon/2}\mathrm{e}^{\mathbf{A}_{\mathrm{C}}(\check{q}_{2d-1},\kappa)\varepsilon/2}, \text{ where } \begin{bmatrix} \check{q}_{2d-1} \\ \check{q}_{2d} \end{bmatrix} := \frac{1}{2\sqrt{3}} \begin{bmatrix} \sqrt{3}+2 & \sqrt{3}-2 \\ \sqrt{3}-2 & \sqrt{3}+2 \end{bmatrix} \begin{bmatrix} q\left(x_d^{\mathrm{m}} - \varepsilon/\left(2\sqrt{3}\right)\right) \\ q\left(x_d^{\mathrm{m}} + \varepsilon/\left(2\sqrt{3}\right)\right) \end{bmatrix}. \quad (23)$$

To obtain the non-equispaced samples of $q(x)$ that are required in (23), we use band limited interpolation, as proposed in [42]. Since this particular non-equispaced grid consists of two equispaced grids, the interpolation requires only three[11] FFT computations, with a complexity of $\mathcal{O}(D\log D)$, and is thus computationally cheap.

[10] CF stands for commutator-free quasi-Magnus exponential integrators. The superscript number denotes the order of accuracy in the step size ε. The subscript denotes the number of matrix exponentials per step.

We see that the computation of $\mathbf{H}_C(x_d^l, x_d^u, \kappa)$ in (23) has the same structure as the one in (12), for two adjacent steps of step size $\varepsilon/2$. That is, the approximation of $a(\kappa)$ and $a\prime(\kappa)$ that we obtain from the $\mathrm{CF}_2^{[4]}$ integrator are (in infinite precision) the exact results for the preprocessed potential

$$\check{q}(x) := \sum_{d=1}^{D} \left(\check{q}_{2d-1}[\![x_d^l < x < x_d]\!] + \check{q}_{2d}[\![x_d < x < x_d^u]\!]\right) = \sum_{d=1}^{2D} \check{q}_d[\![\check{x}_d^l < x < \check{x}_d^u]\!], \tag{24}$$

where $\check{x}_d^l := x_1^l + (d-1)\varepsilon/2$ and $\check{x}_d^u := \check{x}_d^l + \varepsilon/2$.

Since we can interpret the approximations by the $\mathrm{CF}_2^{[4]}$ integrator as the exact results for a real piecewise constant potential, the computations of the accounting function in Section 3.2 still apply. We only have to use the preprocessed potential samples $\check{q}_d(x)$, half the step size ε, and double the number of samples D.

It is natural to ask if this approach can be extended to integrators of orders above four. Unfortunately, higher order CF integrators require complex coefficients or negative step sizes [64,65]. Therefore these break the aforementioned interpretability on which our approach relies. Integrators that assume a piecewise polynomial approximation of the potential lead to more complicated piecewise expressions for the trajectory than (13). Thus in (14) the phase propagation changes from $\varepsilon\gamma$ to $\varepsilon\gamma + \mathcal{O}(\varepsilon^2)$, cf. [62]. If these higher order terms amount to π or more, then (14) will no longer count the correct number of zero crossings. Especially when some of the accuracy gain of such a higher order integrator is traded against a larger step size ε, these higher order terms might become significant. Dealing with this effect in a numerically robust way will need further investigation.

4. Algorithm to compute the eigenvalues

In this section we present the algorithm that we propose for computing the eigenvalues of the KdV. We have seen in Section 2 that the spectral parameter $a(\kappa)$ and the accounting function $s(-\infty, \infty, \kappa)$ both contain the full information on the eigenvalues K_n. Numerically, we can only compute these functions for one sample of κ at a time. Therefore we need a strategy to choose these samples and a system to accumulate the information that we obtain at each new sample. We start with the latter in Section 4.1. In Section 4.2, 4.3 we proceed with respectively an existing and the proposed strategy to choose the samples κ. The existing strategy, bisection, both serves as a benchmark algorithm in Section 5 and as a stepping stone towards the exposition of the proposed algorithm.

4.1. Bounds on the eigenvalues

Our aim is to find the eigenvalues K_n from a numerical algorithm. More precisely, we first want to determine the number of eigenvalues N. Thereto we compute $N \leftarrow s(-\infty, \infty, 0)$. Next, we want to find each eigenvalue K_n with $n \in \{1, 2, \ldots, N\}$ up some user-selectable tolerance Δ, with respect to the eigenvalues of the discretized potential. That is, we want to find N lower bounds K_n^l and N upper bounds K_n^u, such that $\hat{K}_n \in (K_n^l, K_n^u)$ and $K_n^u - K_n^l \leq \Delta$ for all $n \in \{1, 2, \ldots, N\}$. The interval (K_n^l, K_n^u) is called a *bracket* (of the nth eigenvalue). Furthermore, we store the value of the scattering parameter $a(\kappa)$ at all bounds. In the end we will use those residuals to select for each eigenvalue a best guess between the lower and upper bound, based on a minimal residual criterion. The bounds together with the residuals form our current knowledge about the eigenvalues. The initial bounds are the same for all eigenvalues: $K_n^l \leftarrow 0$ and $K_n^u \leftarrow \sqrt{\max(0, \max_d \check{q}_d)}$. (See Section 2.2 and 24.) If $\max_d \check{q}_d \leq 0$, so all samples are non-positive, it follows immediately that the discrete spectrum is an empty set. In that case there is no need to do any other computations than this simple check.

To increase our knowledge about the eigenvalues we proceed as follows. We select a value for κ and then evaluate the scattering parameter $\alpha \leftarrow a(\kappa)$ and the accounting function $\varsigma \leftarrow s(-\infty, \infty, \kappa)$. In Fig. 1 we show how we use these results to update our knowledge about the eigenvalues. Since the accounting function signifies the number of greater eigenvalues, its value ς tells us that $K_n < \kappa$ for $n \in \{1, 2, \ldots, N - \varsigma\}$. Therefore, κ is an upper bound on this subset of the eigenvalues. If this upper bound is tighter than the previously known upper bound, we overwrite it. Similarly κ is a lower bound on K_n for $n \in \{N - \varsigma + 1, \ldots, N - 1, N\}$. If this bound is tighter than the previously known lower bound, we overwrite it.

If $\alpha = 0$, we know that κ is an eigenvalue. In that case $\phi(x, \kappa) \to 0$ as $x \to \infty$, but as emphasized in Section 3.2, we do not count this limit as a zero-crossing. Therefore $\phi(x, K_{N-\varsigma})$ has ς zero-crossings. Thus, if $\alpha = 0$ we infer that $\kappa = K_{N-\varsigma}$. We store this conclusion by setting $K_{N-\varsigma}^l \leftarrow \kappa$ and $K_{N-\varsigma}^u \leftarrow \kappa$.[12] The special case $K_n^l = K_n^u$ should thus be considered as a closed rather than an open interval.

In order to shrink a particular interval (K_n^l, K_n^u), the sample κ must lie in this interval. Hence, if κ does not lie between the currently known bounds of any eigenvalue, the computations $\alpha \leftarrow a(\kappa)$ and $\varsigma \leftarrow s(-\infty, \infty, \kappa)$ will not result in any

[11] If it is permissible to shift the truncation window from $x \in [x_1^l, x_D^u]$ to $x \in [x_1^l \pm \varepsilon/(2\sqrt{3}), x_D^u \pm \varepsilon/(2\sqrt{3})]$, the number of required Fast Fourier Transform (FFT) operations can be reduced to two, by using (instead of two shifted grids) the original grid on which the samples are known together with one grid that is shifted by $\mp\varepsilon/\sqrt{3}$.

[12] In the flowchart in Fig. 1, $K_{N-\varsigma}^u$ is already set to κ before reaching the conditional $\alpha = 0$. Therefore technically only the lower bound $K_{N-\varsigma}^l$ still has to be overwritten in the lower right block. The current representation is chosen for conceptual clarity.

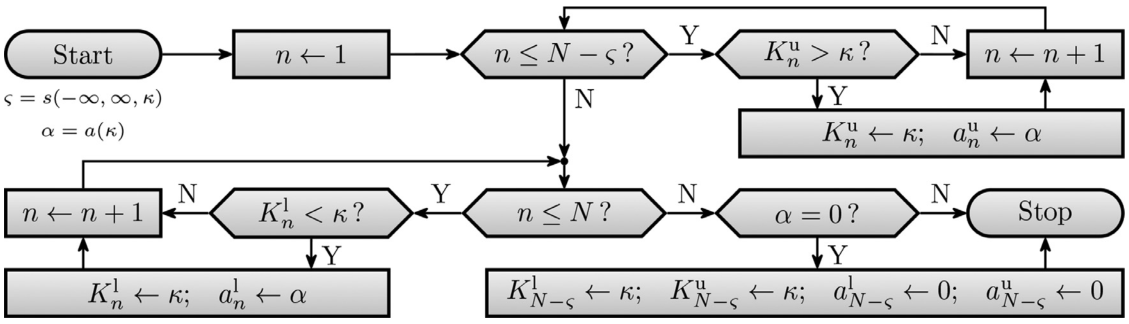
Start
$\varsigma = s(-\infty, \infty, \kappa)$
$\alpha = a(\kappa)$
$n \leftarrow 1$
$n \le N - \varsigma$?
Y
N
$K_n^{\mathrm{u}} > \kappa$?
$n \leftarrow n+1$
$K_n^{\mathrm{u}} \leftarrow \kappa; \quad a_n^{\mathrm{u}} \leftarrow \alpha$
$n \leftarrow n+1$
$K_n^{\mathrm{l}} < \kappa$?
$n \le N$?
$\alpha = 0$?
Stop
$K_n^{\mathrm{l}} \leftarrow \kappa; \quad a_n^{\mathrm{l}} \leftarrow \alpha$
$K_{N-\varsigma}^{\mathrm{l}} \leftarrow \kappa; \quad K_{N-\varsigma}^{\mathrm{u}} \leftarrow \kappa; \quad a_{N-\varsigma}^{\mathrm{l}} \leftarrow 0; \quad a_{N-\varsigma}^{\mathrm{u}} \leftarrow 0$

Fig. 1. Flow chart of the update of the brackets $(K_n^{\mathrm{l}}, K_n^{\mathrm{u}})$ and the respective residuals a_n^{l} and a_n^{u} at the interval boundaries. This flow chart is the specification of the lower left block in Fig. 2 and block 6 in Fig. 3. This update algorithm makes use of the total number of eigenvalues N, the current sample κ, the spectral parameter at this sample $\varsigma = s(-\infty, \infty, \kappa)$, and the residual at this sample $\alpha = a(\kappa)$. Every bracket is checked either in the upper loop for a tightening of the upper bound, or in the lower loop for a tightening of the lower bound. The exceptional case $\alpha = 0$ indicates that κ is an eigenvalue. This gets a special treatment, in the lower right block.

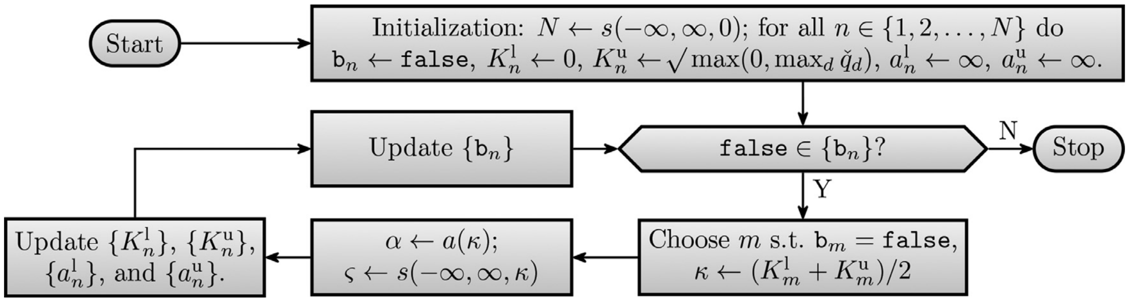
Start
Initialization: $N \leftarrow s(-\infty, \infty, 0)$; for all $n \in \{1, 2, \dots, N\}$ do $\mathtt{b}_n \leftarrow \mathtt{false}$, $K_n^{\mathrm{l}} \leftarrow 0$, $K_n^{\mathrm{u}} \leftarrow \sqrt{\max(0, \max_d \check{q}_d)}$, $a_n^{\mathrm{l}} \leftarrow \infty$, $a_n^{\mathrm{u}} \leftarrow \infty$.
Update $\{\mathtt{b}_n\}$
$\mathtt{false} \in \{\mathtt{b}_n\}$?
N
Stop
Y
Update $\{K_n^{\mathrm{l}}\}$, $\{K_n^{\mathrm{u}}\}$, $\{a_n^{\mathrm{l}}\}$, and $\{a_n^{\mathrm{u}}\}$.
$\alpha \leftarrow a(\kappa)$; $\varsigma \leftarrow s(-\infty, \infty, \kappa)$
Choose m s.t. $\mathtt{b}_m = \mathtt{false}$, $\kappa \leftarrow (K_m^{\mathrm{l}} + K_m^{\mathrm{u}})/2$

Fig. 2. Top level flow chart of the bisection benchmark algorithm.

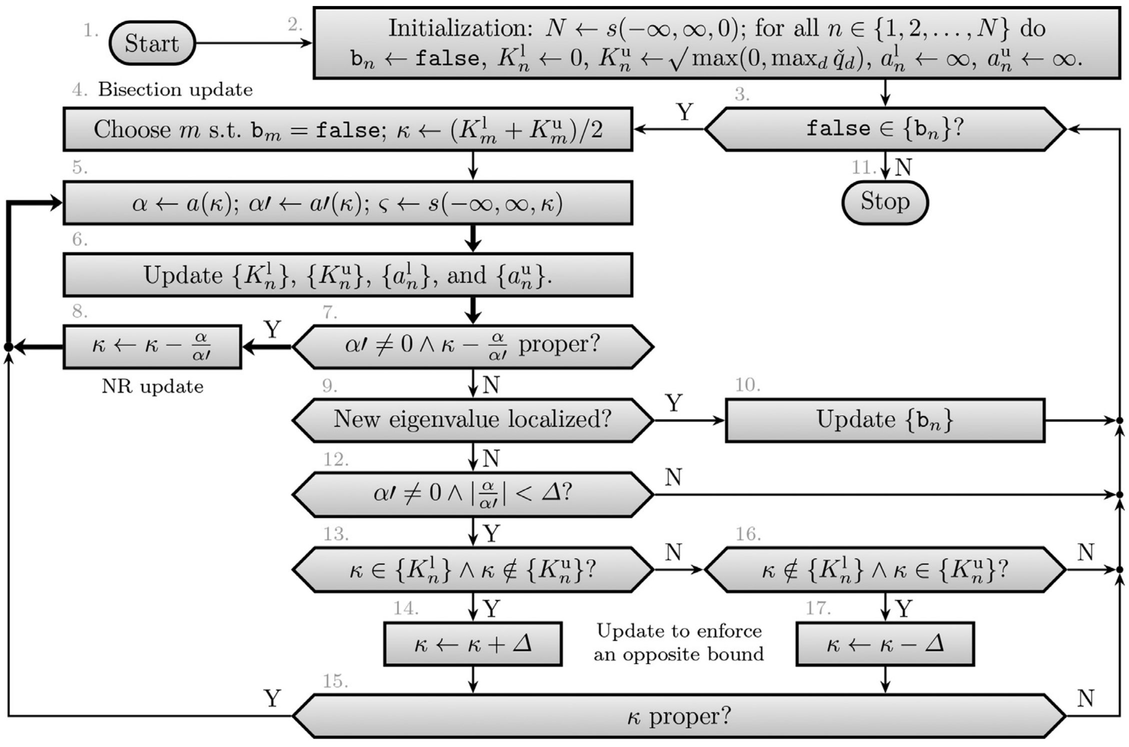
1. Start
2. Initialization: $N \leftarrow s(-\infty, \infty, 0)$; for all $n \in \{1, 2, \dots, N\}$ do $\mathtt{b}_n \leftarrow \mathtt{false}$, $K_n^{\mathrm{l}} \leftarrow 0$, $K_n^{\mathrm{u}} \leftarrow \sqrt{\max(0, \max_d \check{q}_d)}$, $a_n^{\mathrm{l}} \leftarrow \infty$, $a_n^{\mathrm{u}} \leftarrow \infty$.
4. Bisection update
Choose m s.t. $\mathtt{b}_m = \mathtt{false}$; $\kappa \leftarrow (K_m^{\mathrm{l}} + K_m^{\mathrm{u}})/2$
3. $\mathtt{false} \in \{\mathtt{b}_n\}$?
11. N Stop
5. $\alpha \leftarrow a(\kappa)$; $\alpha\prime \leftarrow a\prime(\kappa)$; $\varsigma \leftarrow s(-\infty, \infty, \kappa)$
6. Update $\{K_n^{\mathrm{l}}\}$, $\{K_n^{\mathrm{u}}\}$, $\{a_n^{\mathrm{l}}\}$, and $\{a_n^{\mathrm{u}}\}$.
8. $\kappa \leftarrow \kappa - \frac{\alpha}{\alpha\prime}$
NR update
7. $\alpha\prime \ne 0 \wedge \kappa - \frac{\alpha}{\alpha\prime}$ proper?
9. New eigenvalue localized?
10. Update $\{\mathtt{b}_n\}$
12. $\alpha\prime \ne 0 \wedge |\frac{\alpha}{\alpha\prime}| < \Delta$?
13. $\kappa \in \{K_n^{\mathrm{l}}\} \wedge \kappa \notin \{K_n^{\mathrm{u}}\}$?
16. $\kappa \notin \{K_n^{\mathrm{l}}\} \wedge \kappa \in \{K_n^{\mathrm{u}}\}$?
14. $\kappa \leftarrow \kappa + \Delta$
Update to enforce an opposite bound
17. $\kappa \leftarrow \kappa - \Delta$
15. κ proper?
Y
N

Fig. 3. Top level flow chart of the proposed algorithm.

progress regarding our knowledge about the eigenvalues. We say that κ is proper if and only if $\kappa \in \bigcup_{n=1}^{N}(K_n^{\mathrm{l}}, K_n^{\mathrm{u}})$. We use the properness of κ as a conditional in the proposed algorithm to prevent such idle iterations.

4.2. Bisection

Osborne [24] proposed to search for the eigenvalues by means of bisection. An implementation of that is shown in Fig. 2. In addition to the variables that are explained in Section 4.1, we keep track of a boolean array $\{\mathtt{b}_n\}$ that stores if the nth eigenvalue is localized. It is thus updated as $\mathtt{b}_n \leftarrow (K_n^{\mathrm{u}} - K_n^{\mathrm{l}} \le \Delta)$, for $n \in \{1, 2, \ldots, N\}$. As long as there are eigenvalues left to localize, we set κ by bisecting one of their intervals. After computing $\alpha \leftarrow a(\kappa)$ and $\varsigma \leftarrow s(-\infty, \infty, \kappa)$, we update the bounds of all the eigenvalues as described in Section 4.1 and Fig. 1. Then we update $\{\mathtt{b}_n\}$ and repeat. Finally, we select as best guess for each eigenvalue the bound with the lowest residual. That is, $\hat{K}_n := K_n^{\mathrm{l}}[\![|a_n^{\mathrm{l}}| < |a_n^{\mathrm{u}}|]\!] + K_n^{\mathrm{u}}[\![|a_n^{\mathrm{l}}| \ge |a_n^{\mathrm{u}}|]\!]$.

4.3. Proposed algorithm

Bisection is a simple technique, but it has only linear convergence. Christov [57] proposed to use an Illinois type *regula falsi* root-finder instead, to speed up the convergence. However, this technique requires initial brackets that contain exactly one eigenvalue, for which he resorted to a grid search. Root finding based on $a(\kappa)$ and $a\prime(\kappa)$ with the NR technique enjoys a quadratic convergence. However, an open root-finder like NR is more difficult to combine with SL oscillation theory than a bracketing root-finder like bisection. First, one has to find initial guesses within the basins of attraction of every eigenvalue rather than an initial bracket. Second, one can only establish convergence to the nth eigenvalue if the corresponding unit step of the accounting function is sampled both below and above the eigenvalue at a distance smaller than the tolerance Δ. A possible strategy is to apply a two-step procedure: First use bisection to find a reasonably tight bracket for each eigenvalue and refine thereafter with a NR procedure [56,58,66]. However, this combines the disadvantages of both techniques: The slow convergence of bisection and the difficulty of keeping NR iterations within a bracket.

We instead propose a hybrid algorithm that combines quadratic convergence (for most iterations) with the guarantee that all eigenvalues are localized. In short, we follow the NR procedure, unless its update results in an improper value κ. (See Section 4.1 for the definition of improper.) Only in that case and at the start we apply one bisection step to obtain a new initial value for the NR procedure. The algorithm stops when each eigenvalue is localized. Hence we do not select a priori the order in which we search for the eigenvalues. Thus we avoid the difficulty of finding an initial guess in the basin of attraction of that eigenvalue. Instead we allow the NR procedure to converge to any eigenvalue. If it happens to have started within the basin of attraction of an eigenvalue that is already localized, it will update to an improper value of κ, typically already in the first iteration. Then we use a bisection step instead.

A flowchart of the proposed algorithm is shown in Fig. 3. The blocks 1 up to and including 11 form the basis of the algorithm. Let us describe these blocks first. Thereafter we will explain the purpose and working of blocks 12 up to and including 17. **1:** The input of the proposed algorithm is the array of samples $\check{q}_d$ that results from (23). Furthermore, the tolerance on the eigenvalues, Δ, must be provided. One should keep in mind that this is the maximum deviation of the returned eigenvalues with respect to the eigenvalues of the discretized potential. The error that is caused by the sampling and reconstruction of the potential cannot be reduced by choosing Δ smaller and smaller. **2:** The initialization is the same as for the bisection algorithm. See Section 4.1 and 4.2. **3:** While there are eigenvalues left to localize, continue searching. **4:** We select any unlocalized eigenvalue and bisect its currently known bracket. It makes little difference which of the remaining brackets we choose to bisect. We choose the lowest m for which $\mathtt{b}_m = \mathtt{false}$. **5:** At the current value κ we evaluate $\alpha \leftarrow a(\kappa)$, $\alpha\prime \leftarrow a\prime(\kappa)$, and the accounting function $\varsigma \leftarrow s(-\infty, \infty, \kappa)$. This is the most computationally expensive step, so we will count every time the algorithm enters this block as one iteration. **6:** We update the brackets of all eigenvalues as described in Section 4.1 and Fig. 1. **7–8:** If κ is not a stationary point, we can compute the NR update of κ. That is, $\kappa \leftarrow \kappa - \alpha/\alpha\prime$. If the update succeeds, the NR descend continues from the updated value κ. However, if κ would become improper after this update (see Section 4.1), we break the NR cycle before the actual update. **9–10:** We consider an eigenvalue as localized if $K_n^{\mathrm{u}} - K_n^{\mathrm{l}} \le \Delta$. If that condition is met for any eigenvalue n whereas $\mathtt{b}_n$ is still `false`, we set $\mathtt{b}_n$ to `true` and return. Please note that this is not necessarily the eigenvalue m that was selected in block 4, and that in rare cases we may find more than one eigenvalue at once. **11:** When all eigenvalues are localized up to an interval of length Δ at most, we stop searching. For each eigenvalue $n \in \{1, 2, \ldots, N\}$ we return the bound with the smallest residual as the best guess. That is, $\hat{K}_n := K_n^{\mathrm{l}}[\![|a_n^{\mathrm{l}}| < |a_n^{\mathrm{u}}|]\!] + K_n^{\mathrm{u}}[\![|a_n^{\mathrm{l}}| \ge |a_n^{\mathrm{u}}|]\!]$.

If we would only use the part of the algorithm that is described in blocks 1 up to and including 11, we would already be able to localize all eigenvalues reliably. However it has a flaw that makes it fall back to linear convergence. This is because the NR algorithm tends to converge to a root monotonically: either from above or from below. Let us consider the spectral parameter $a(\kappa)$ around a root K_n. More precisely, we look at a neighbourhood of K_n where $a(\kappa)$ is either convex or concave. In the convex case, every tangent line to $a(\kappa)$ in this neighbourhood crosses zero at $\kappa > K_n$. This crossing represents the NR update. That means that once κ enters this neighbourhood, all following updates satisfy $\kappa > K_n$. Thus, ignoring finite precision effects, the algorithm converges monotonically to K_n from above. Upon termination, we will thus have found a tight upper bound of that root. However, since we require that the bracket $(K_n^{\mathrm{l}}, K_n^{\mathrm{u}})$ is smaller than Δ before we consider an eigenvalue localized, we also need a sufficiently tight lower bound. Unfortunately, this lower bound can only be found with bisection, since every nearby lower NR iteration would immediately jump over the already known upper bound. *Mutadis* *mutandis*, the same happens if $a(\kappa)$ is concave in a neighbourhood of K_n. Therefore, the algorithm as a whole falls back on linear convergence.[13]

The blocks 12 up to and including 17 in Fig. 3 are added to avoid this. The principle is that if the NR descend terminates on a tight upper bound, we decrement κ by Δ to enforce a tight enough lower bound. Vice versa, we increment κ by Δ in case of a tight lower bound. In detail: **12:** When the NR cycle terminated without localizing an eigenvalue, there are two possibilities. Firstly, we might not have started within the basin of attraction of any eigenvalue that was left to localize. In that case we need to return, to find a new starting point by bisection. Secondly, due to numerical effects, the NR update will always become improper near a root of $a(\kappa)$. When that happens, we have localized one tight bound of that root. In that case we want to enforce the opposite bound, as explained before. To distinguish between these two cases, we look at the magnitude of the NR update that would have taken place. **13–17:** If the value of κ became a lower bound for one or more eigenvalues (in block 6), we expect at this point that this bound is tight for at least one of these eigenvalues. In order to guarantee that this eigenvalue is localized, we try to put an upper bound at $\kappa + \Delta$. We expect $\kappa + \Delta$ to be proper in this case, but we test it nevertheless (in block 15), to guard against unforeseen behaviour. Similarly, if he value of κ became an upper bound for one or more eigenvalues, we expect at this point that this bound is tight for at least one of these eigenvalues. In order to guarantee that this eigenvalue is localized, we try to put a lower bound at $\kappa - \Delta$. We do not give any special treatment to the rare case in which κ is both a lower bound of one eigenvalue and an upper bound of another.

5. Examples

In this section we will demonstrate the proposed algorithm. We select six different vanishing potentials and compute their eigenvalues with the proposed algorithm and with three benchmark algorithms. We first compare the accuracy that the algorithms achieve as a function of the tolerance Δ. Next, we compare in detail the accuracy and computational cost at an ideal fixed value Δ.

5.1. Benchmark algorithms

We compare the proposed algorithm to two versions of the bisection algorithm that was described in Section 4.2 and Fig. 2. The difference between the two versions lies in the reconstruction of the potential. By using the reconstruction in Section 3.1 we obtain the second order version. We will refer to this benchmark algorithm as Alg. B2. By using the reconstruction in (23) and (24) we obtain the fourth order version. We refer to this benchmark algorithm as Alg. B4. Both Algs. B2 and B4 use the *proposed* root counter that is described in Section 3.2. We will also test one of the examples with Alg. B2s. That algorithms is equal to Alg. B2, except that it uses the root counter that was proposed by Osborne [24]. This particular example shows the problem with that root counter. We remark that we cannot use the complete algorithm from Osborne [24] as a benchmark, since we must at least adapt it to data with vanishing instead of periodic boundary conditions.

Furthermore, we compare the proposed algorithm to Matslise 2.0 [58].[14] We refer to this eighteenth order benchmark algorithm as Alg. MS18. This algorithm needs an analytic expression of the input signal $q(x)$, whereas our algorithm requires samples. Therefore we use band limited interpolation. That is, we pass an expression of the form

$$q_{\mathrm{MS}}(x) := -\Big(\alpha_0 + \textstyle\sum_{d=1}^{\lfloor D/2\rfloor} \alpha_d \cos\big(\frac{2\pi d}{\varepsilon D}(x - x_1)\big) + \sum_{d=1}^{\lceil D/2\rceil - 1} \beta_d \sin\big(\frac{2\pi d}{\varepsilon D}(x - x_1)\big)\Big)[\![x > x_1^{\mathrm{l}}]\!][\![x < x_D^{\mathrm{u}}]\!], \tag{25}$$

for which we calculate the coefficients α_d and β_d with the FFT of the sampled input signal. The minus sign is required due to the different parametrisation of the Schrödinger equation. Matslise does not make use of the step size ε of the data, but automatically chooses a mesh. Matslise only allows us to specify a tolerance for the squares of the eigenvalues, not for the eigenvalues themselves. We use the same tolerance Δ as for the other algorithms, but point out that it has to be interpreted differently for Matslise. We also pass to Matslise the jump points $x = x_1^{\mathrm{l}}$ and $x = x_D^{\mathrm{u}}$. We obtain the eigenvalues as $K_n = \sqrt{-E_n}$, where E_n are the squared eigenvalues that are returned by Matslise.

5.2. Example potentials

We will compute the eigenvalues of the six potentials that are shown in Fig. 4. The exact description of these potentials is as follows. We define $q_1(x) := 99\mathrm{sech}^2(2x)$. This is a non-reflectionless[15] potential with five eigenvalues: $K_1 = 1$, $K_2 = 3$, $K_3 = 5$, $K_4 = 7$, and $K_5 = 9$ [59, §2.5]. We define $q_2(x) := 24.99\mathrm{sech}^2(x/5)$. This is a non-reflectionless potential with 25 eigenvalues: $K_n = 0.2n - 0.1$ for $n \in \{1, 2, \ldots, 25\}$ [59, §2.5]. The reflectionless potential $q_3(x)$ shows a typical far-field pattern with six separated solitons. We selected its eigenvalues as $K_n := \sqrt{n}$, where $n \in \{1, 2, \ldots, 6\}$. The norming constants are set to $b(K_n) := (-1.0 \times 10^{12})^n$. The required samples of this potential are computed numerically with the Crum transform

[13] Thanks to finite precision effects, the NR descend sometimes finds a sufficiently small interval anyway. That happens if one of the final updates numerically jumps over the root. However, we want to enforce quadratic convergence in the majority of the cases.

[14] We considered Matslise 3.0/Pyslise [67] as well, but this package is currently still under development [68]. In particular, we found that boundary conditions at infinity are not yet supported.

[15] A reflectionless potential is a potential of which the continuous KdV-NFT spectrum is zero, such that the signal only consists of the (possibly interacting) solitions that are represented by the eigenvalues.

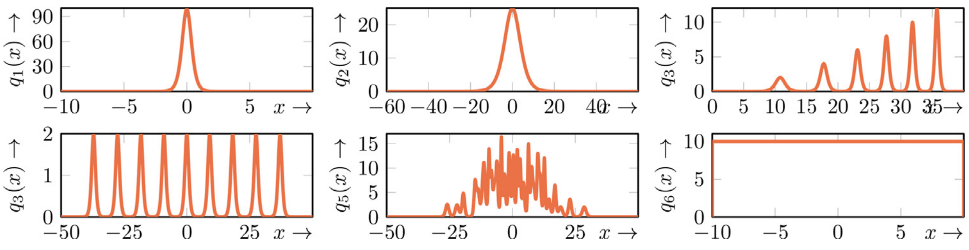

Fig. 4. Potentials of which the eigenvalues are calculated to demonstrate the proposed algorithm.

algorithm of [41]. The reflectionless potential $q_4(x)$ forms a wave train with nearly equal eigenvalues. The eigenvalues are selected as $K_n := 1 + (n-5)10^{-4}$, where $n \in \{1, 2, \dots, 9\}$. The norming constants are set to $b(K_n) := -(-1)^n$. The required samples of this potential are computed numerically with the same Crum transform algorithm [41]. Potential $q_5(x)$ is a reflectionless potential with thirty eigenvalues. The eigenvalues are selected as $\{K_n\} := \{2 - \cos(i) | i \in \{1, 2, \dots, 30\}\}$. We sort this set of eigenvalues such that $K_1 < K_2 < \cdots < K_{30}$, Then, the norming constants are set to $b(K_n) := (-1)^n 10^{6\sin(n)}$, where n is the index after sorting. The required samples of this potential are once again computed numerically with the Crum transform algorithm from Prins and Wahls [41]. Potential $q_6(x)$ is a rectangular potential that is defined as $q_6(x) := 10\,[\![-10 \le x \le 10]\!]$. It is non-reflectionless and it has 21 eigenvalues. There exists no closed form expression to compute these eigenvalues. However, they can be approximated to any desired finite precision with e.g. [41, Eq. 49].

For numerical processing, all potentials are truncated to the respective intervals shown in Fig. 4. Potentials $q_1(x)$ till $q_5(x)$ are sampled on a uniform grid of 10^4 samples on this interval, such that for example $x_1^{\mathrm{l}} = -10$ and $x_D^{\mathrm{u}} = 10$ for potential $q_1(x)$. Potential $q_6(x)$ is sampled with only two samples, at $x = -5$ and $x = 5$ respectively.

5.3. Error measures

Since we know the true eigenvalues of each of the example potentials, we can calculate the error of the computed eigenvalues. In some cases the numerical algorithms return a different number of eigenvalues than the true number of eigenvalues. We will always compare every true eigenvalue to the computed eigenvalue with the same number of zero-crossings of the eigenfunction. Thus, the greatest true eigenvalue is compared to the greatest computed eigenvalue etcetera. If the number of computed eigenvalues $\hat{N}$ is larger than the true number of eigenvalues N, we compute a separate error measure that compares the spurious $\hat{N} - N$ smallest eigenvalues to zero. The thought behind this is that an artificial eigenvalue of (nearly) zero corresponds to a part of the spectrum with (nearly) zero mass, momentum, and energy. We thus define

$$\underset{n\le N}{\mathrm{RMS}}\left\{\hat{K}_{n+\hat{N}-N} - K_n\right\} := \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\hat{K}_{n+\hat{N}-N} - K_n\right)^2}, \qquad \underset{n>N}{\mathrm{RMS}}\left\{\hat{K}_{n+\hat{N}-N}\right\} := \sqrt{\frac{1}{\hat{N}-N}\sum_{n=1}^{\hat{N}-N}\hat{K}_n^2}, \text{ if } \hat{N} > N.$$

In addition, we can indicate the error by the residual of the spectral parameter $a(\kappa)$ at the computed eigenvalues. This error measure does not account for discretization errors. The main benefit is that it can even be calculated if the true eigenvalues are not known, which is in practice typically the case. Again, we calculate this error for the N largest eigenvalues and the spurious $\hat{N} - N$ smallest eigenvalues separately:

$$\underset{n\le N}{RMS}\left\{a(\hat{K}_n)\right\} := \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(a(\hat{K}_{n+\hat{N}-N})\right)^2}, \qquad \underset{n>N}{RMS}\left\{a(\hat{K}_n)\right\} := \sqrt{\frac{1}{\hat{N}-N}\sum_{n=1}^{\hat{N}-N}\left(a(\hat{K}_n)\right)^2}, \text{ if } \hat{N} > N.$$

Unfortunately, the Matslise benchmark algorithm (MS18) does not provide information on the residual.

5.4. The effect of the tolerance

Both the proposed algorithm and the benchmark algorithms contain a tolerance parameter Δ to select the desired accuracy. However, they respond very differently to the setting of Δ. This can be seen in Fig. 5. In that figure we compare for the potential $q_1(x)$ the proposed algorithm to the benchmark algorithms for different settings of Δ.

The top right panel of Fig. 5 shows the achieved error $\mathrm{RMS}_{n\le N}\left\{\hat{K}_{n+\hat{N}-N} - K_n\right\}$ as a function of the tolerance Δ. The similar graphs for the example potentials $q_2(x)$ till $q_6(x)$ are shown in Fig. 6. We see that for high values of the tolerance Δ the bisection based benchmark algorithms (Algs. B2 and B4) achieve an error $\mathrm{RMS}_{n\le N}\left\{\hat{K}_{n+\hat{N}-N} - K_n\right\}$ just below the tolerance Δ. When the tolerance is reduced, both of these algorithms hit an error floor when the error due to the reconstruction of the potential becomes the dominant source of error. These error floors depend on the potential as well as on the sampling interval and truncation. The fourth order error floor of benchmark Alg. B4 is in most cases lower than the second order error

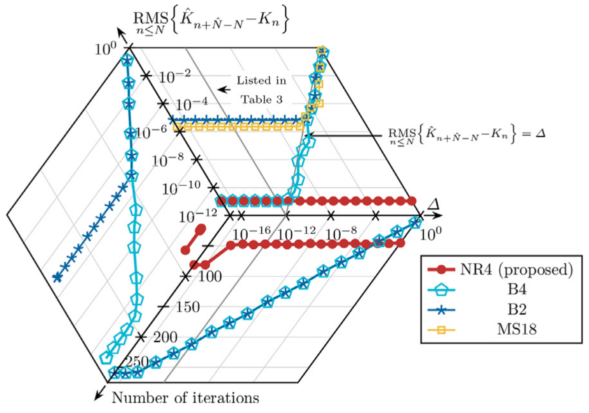

Fig. 5. The plots show a comparison between the proposed algorithm and the three benchmark algorithms for the example potential $q_1(x)$. The top right panel shows the achieved accuracy as a function of the chosen tolerance Δ. The bottom right panel shows the number of iterations that is required, as a function of the chosen tolerance Δ. The left panel shows the resulting trade-off curve. For Alg. MS18 no information is available on the number of iterations.

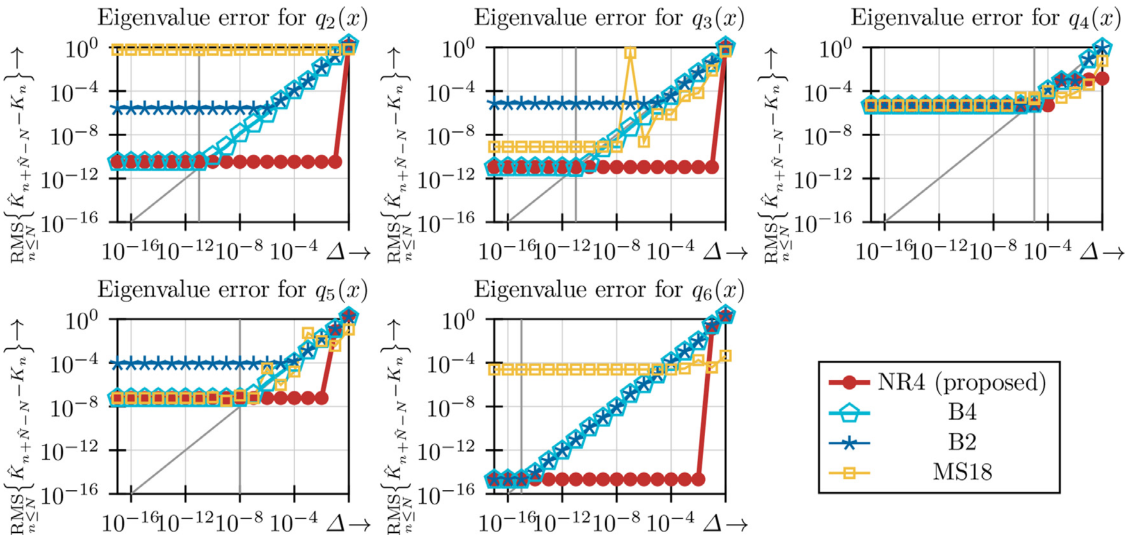

Fig. 6. The plots show the achieved accuracy as a function of the chosen tolerance Δ for the example potentials $q_2(x)$ till $q_6(x)$ when using the proposed algorithm and the three benchmark algorithms respectively. The diagonal grey lines show $\underset{n\le N}{\mathrm{RMS}}\{\hat{K}_{n+\hat{N}-N}-K_n\}=\Delta$ and the vertical grey lines show the value Δ that is listed in Table 3.

floor of benchmark Alg. B2.[16] The proposed algorithm, Alg. NR4, achieves the fourth order error floor for a much wider range of settings of the tolerance Δ. The reason is that if the NR descend is successful, it terminates at either a tight upper or lower bound. The opposite bound is enforced to ensure that the bound is tight enough, but in practice it is the termination point of the NR descend that determines the returned estimates $\hat{K}_n$. If the allowable tolerance is very high, the error of the proposed algorithm may be closer to the tolerance. In those cases one or more NR descends jumped out of the range of proper values κ, upon which the algorithm checks if the bracket is below the tolerance. (See block 9 in Fig. 3.) If a high tolerance is allowed, that may already be the case, even if the NR descend did not converge. On the other hand, if we select an extremely low tolerance, machine precision effects can show up. Among these examples this effect is only visible for the potential $q_1(x)$, in Fig. 5 for $\Delta = 10^{-16}$ and $\Delta = 10^{-17}$. In these cases the tolerance is so small that after the termination of the NR descend, the update to enforce an opposite bound (blocks 17 and 18 in Fig. 3) is too small to find an opposite bound immediately. It is thus advisable not to choose Δ lower than 10^{-15} in double precision computations.

The invariant behaviour of the proposed algorithm, Alg. NR4, with respect to a wide range of tolerances Δ is a practical advantage: We do not have to think much about an appropriate setting of Δ, since we obtain the maximum accuracy in the

[16] For example potential $q_6(x)$ the second and fourth order error floor are equal, because a rectangular potential is a truncated constant. For example potential $q_4(x)$ the error floor is equal for all algorithms. We suspect that the computation of the samples of $q_4(x)$ is the dominant source of error in that example, due to the close separation of the eigenvalues.

Table 3
Results of finding the eigenvalues of potentials $q_1(x), \ldots, q_6(x)$ (see Fig. 4) with four versions of an automatic eigenvalue finder: Algs. NR4 (proposed), B2, B4 and MS18. Additionally, the results of a fifth algorithm, Alg. B2s, are shown for the potential $q_6(x)$. Algorithm B2s uses a simple sign check root counter instead of (14) and is otherwise equal to Alg. B2. Table cells of which the value cannot be computed (because the computed number of eigenvalues $\hat{N}$ leads to an invalid expression) are left blank. The symbol $*$ means that the data are missing that are needed to compute the value in that cell.

Example	Alg.	$\hat{N}$	Iter.	$\underset{n\le N}{\mathrm{RMS}}\{\hat{K}_{n+\hat{N}-N} - K_n\}$	$\underset{n\le N}{\mathrm{RMS}}\{a(\hat{K}_{n+\hat{N}-N})\}$	$\underset{n>N}{\mathrm{RMS}}\{\hat{K}_n\}$	$\underset{n>N}{\mathrm{RMS}}\{a(\hat{K}_n)\}$
$q_1(x)$, $N=5$, $\Delta=10^{-12}$	NR4	N	47	1.09×10^{-11}	2.47×10^{-16}		
	B4	N	212	1.09×10^{-11}	3.89×10^{-14}		
	B2	N	212	6.70×10^{-6}	4.43×10^{-14}		
	MS18	N	$*$	2.26×10^{-6}	$*$		
$q_2(x)$, $N=25$, $\Delta=10^{-11}$	NR4	N	202	3.32×10^{-11}	2.54×10^{-19}		
	B4	N	881	3.38×10^{-11}	1.33×10^{-12}		
	B2	N	881	2.68×10^{-6}	1.05×10^{-12}		
	MS18	$N+3$	$*$	6.19×10^{-1}	$*$	3.68×10^{-1}	$*$
$q_3(x)$, $N=6$, $\Delta=10^{-11}$	NR4	N	52	1.09×10^{-11}	9.89×10^{-21}		
	B4	N	219	9.69×10^{-12}	2.26×10^{-15}		
	B2	$N+1$	256	7.47×10^{-6}	3.62×10^{-16}	3.05×10^{-5}	5.50×10^{-8}
	MS18	N	$*$	7.93×10^{-10}	$*$		
$q_4(x)$, $N=9$, $\Delta=10^{-5}$	NR4	$N+1$	188	4.93×10^{-6}	1.06×10^{-46}	6.45×10^{-5}	3.82×10^{-2}
	B4	$N+1$	77	6.90×10^{-6}	1.05×10^{-36}	6.47×10^{-5}	4.21×10^{-2}
	B2	$N+1$	77	5.77×10^{-6}	1.99×10^{-36}	1.03×10^{-4}	6.63×10^{-3}
	MS18	N	$*$	2.57×10^{-5}	$*$		
$q_5(x)$, $N=30$, $\Delta=10^{-8}$	NR4	N	230	6.01×10^{-8}	1.49×10^{-36}		
	B4	N	707	6.42×10^{-8}	1.30×10^{-28}		
	B2	$N+1$	733	8.79×10^{-5}	6.14×10^{-29}	5.35×10^{-4}	6.85×10^{-6}
	MS18	N	$*$	1.02×10^{-7}	$*$		
$q_6(x)$, $N=21$, $\Delta=10^{-15}$	NR4	N	258	2.13×10^{-15}	1.70×10^{-23}		
	B4	N	1002	2.07×10^{-15}	1.73×10^{-18}		
	B2	N	1002	2.07×10^{-15}	2.17×10^{-18}		
	B2s	$N-20$	52				
	MS18	N	$*$	2.44×10^{-5}	$*$		

same number of iterations for any reasonable setting. For computations in double precision arithmetic, we would simply select a default value of $\Delta = 10^{-15}$.

Algorithm MS18 does not make use of the samples that the other algorithms use. Instead it selects its own mesh based on the variation of the potential and the user input tolerance [58, §2.3]. Please recall that Matslise's tolerance is a tolerance on the squares of the eigenvalues, and therefore $\mathrm{RMS}_{n\le N}\left\{\hat{K}_{n+\hat{N}-N} - K_n\right\} < \Delta$ does not have to hold. Nevertheless, we would expect that as the tolerance Δ is tightened, the result keeps getting more accurate (at the cost of computation time) until the truncation of the potential becomes the dominant source of error. Since the other algorithms suffer likewise from this truncation error, the error floor of the fourth order algorithms (Algs. B4 and NR4) should be achievable for the eighteenth order Matslise algorithm as well. Nevertheless, we see in Figs. 5 and 6 that the error floor of Alg. MS18 can be much higher than the fourth order error floor. Furthermore, we see for some of the examples that the error curve is not a monotonically non-decreasing function of the tolerance, but shows local maxima instead. We also noticed that these error curves can change significantly if the computation is repeated with a different version of Matlab, or if the potential is obtained from a slightly different number of samples before band limited interpolation. According to the current developers, Matslise 3.0/Pyslise uses a more conservative error estimate and a finer mesh than its predecessor [67]. Therefore we surmise that the next version, if it allows boundary conditions at infinity, will return stabler and more accurate results.

In the bottom right panel of Fig. 5, we see the effect of the tolerance on the number of iterations. For the bisection based algorithms, Algs. B2 and B4, the number of iterations increases proportional to $\log(1/\Delta)$ and is the same for both versions. The proposed algorithm, Alg. NR4, needs roughly the same number of iterations for every user-selected tolerance Δ.[17] Unfortunately we do not have access to the number of iterations (and computational cost per iteration) of Alg. MS18.

The left panel of Fig. 5 shows the trade-off curve between accuracy and number of iterations that results from the other two panels. We will discuss it in the next subsection.

5.5. Comparison of error and computational cost

The left panel of Fig. 5 shows the trade-off curve between accuracy and number of iterations that results from the other two panels. We see that the proposed algorithm, Alg. NR4, achieves the best achievable accuracy of the fourth order

[17] If the error is not at the fourth order error floor for high Δ, such as in Fig. 6, the algorithm terminates earlier. Hence in that case the number of iterations also drops.

benchmark algorithm, Alg. B4, but at a much lower number of iterations. Even if we take into account that the iterations of the proposed algorithm are 70% more expensive (see Section 3.3), the proposed algorithm remains computationally cheaper.

Hereafter, we want to compare the proposed algorithm to the benchmark algorithms in more detail, considering computational cost and several kinds of error. For a fair comparison it would be preferable to fix the error and compare the computational cost, or to fix the computational cost and compare the errors. That is unfortunately not possible. We can only fix the tolerance. We set it close to the optimal value for the benchmark algorithm B4. That is, close to the maximum value Δ for which the error $\mathrm{RMS}_n\{\hat{K}_{n+\hat{N}-N} - K_n\}$ is at the error floor. For example in Fig. 5 we see that this is $\Delta = 10^{-12}$ for potential $q_1(x)$. The values for the other potentials can be read likewise from Fig. 6. A consequence of this choice is that the eigenvalue errors of Algs. NR4 and B4 will be both on the fourth order error floor, so approximately the same. We will furthermore have to keep in mind that Alg. B2 would be capable of achieving the same error as in this test at less computational cost, by selecting a higher tolerance. The same likely holds for Alg. MS18, but we cannot verify that since we have no information on its computational cost for these examples.

In Table 3 we show the results at this tolerance for the proposed algorithm, Alg. NR4, as well as for the three benchmark algorithms, Algs. B2, B4 and MS18. Besides the eigenvalue error that we have already seen in Figs. 5 and 6, it also shows the residual error, the errors which correspond to spurious eigenvalues where applicable and the iteration counts. Algorithm B2s is equal to Alg. B2, except for the root counter. It uses instead of (14) always the simple sign check, as in the upper case of (14). Algorithm B2s is thus as close as possible to the algorithm that was proposed in [24], except for the necessary adaptation to vanishing boundary conditions. We will discuss the results from this algorithm in Section 5.6.

5.5.1. Computational cost

Although we optimized the tolerance for the B4 algorithm, Table 3 shows that the proposed algorithm requires only about a fourth of the number of iterations of the bisection based algorithms for most example potentials. Even when we take into account that iterations of the proposed algorithm are approximately 70% more expensive than those of B4 (see Section 3.3) and those of Alg. B2 are 50% cheaper than those of B4, the proposed algorithm remains computationally cheapest. Indeed, Alg. B2 would have achieved approximately the same error at a lower number of computations if a higher tolerance were selected. However, the optimal choice for the tolerance is in practice unknown.

The proposed algorithm is typically computationally cheaper than bisection, but Table 3 shows one exception, namely potential $q_4(x)$. The eigenvalues of $q_4(x)$ are clustered together and while searching for the first eigenvalue, the bisection algorithms find a tight bracket for the whole cluster. Therefore the other eight eigenvalues can be localized in few extra iterations. Algorithm NR4 on the other hand converges first with a long series of NR iterations from below to the smallest eigenvalue. For the remainder of the cluster this only gives a reasonably tight lower bound. It finds the upper bound of the cluster with another long series of NR iterations that converges to the highest eigenvalue. Both of these series appear to be atypical cases in which the NR algorithm converges rather slowly.

5.5.2. Spurious eigenvalues

When an algorithm returns more eigenvalues than there should be, we refer to the surplus of eigenvalues as *spurious eigenvalues*. More precisely, we consider the $\hat{N} - N$ smallest eigenvalues, where $\hat{N}$ is the numerically calculated number of eigenvalues, as spurious. Those are the eigenvalues for which the highest numbers of zero-crossings are detected.

Regarding potential $q_4(x)$ we see in Table 3 that all algorithms except for Alg. MS18 return one spurious eigenvalue. Algorithm B2 furthermore returns one spurious eigenvalue for the potentials $q_3(x)$ and $q_5(x)$. In all cases this eigenvalue is close to zero: The eigenvalue is typically one order of magnitude larger than the *error* in the true eigenvalues. This is typical for reflectionless potentials. The reason is that the scattering parameter $a(\kappa)$ of the reconstructed potential indeed has an extra root compared to the true potential. However, this is not problematic in practice, since a near-zero eigenvalue represents in the KdV-NFT spectrum a component of near-zero 'mass', 'momentum' and 'energy'. More precisely, these are proportional to respectively the first, third and fifth power of the eigenvalue [69, §3.1], [70, §3]. Therefore, if we would compute the inverse KdV-NFT of a spectrum, the presence of a very small spurious eigenvalue will have little influence on the resulting potential.

For potential $q_2(x)$ we see in Table 3 that Alg. MS18 returns three spurious eigenvalues, while the other algorithms find the correct number of eigenvalues. The spurious and non-spurious eigenvalue errors of Alg. MS18 are also very high in this case. This is because Alg. MS18 finds the true eigenvalues to reasonable precision, plus some spurious eigenvalues *between* the true eigenvalues. The index of some of the smallest true eigenvalues is shifted by these spurious eigenvalues. This would be expected behaviour for a root finder that does not use SL oscillation theory. However, this theory makes it possible to index every single eigenvalue without knowing the others, by counting the number of zero-crossings of its eigenfunction [58]. Therefore, this behaviour is remarkable for a root finder that is based on SL theory.

5.5.3. Residual errors

We have no information on the residual error of Alg. MS18, so we can compare the proposed algorithm, Alg. NR4, only to the bisection based benchmark Algorithms B2 and B4. When we look at the residual errors of the non-spurious eigenvalues $\mathrm{RMS}_{n\le N}\left\{a(\hat{K}_{n+\hat{N}-N})\right\}$, we see that the proposed algorithm, Alg. NR4, achieves in all cases a significantly lower error than the bisection Algorithms B2 and B4. Nevertheless, the eigenvalue error $\mathrm{RMS}_{n\le N}\left\{\hat{K}_{n+\hat{N}-N}-K_n\right\}$ of the proposed algorithm is similar to the eigenvalue error of Alg. B4. This shows that the proposed algorithm gets significantly closer to the eigenvalues of the *reconstructed* potential (see (24)), but not closer to the eigenvalues of the *true* potential. Hence, the error of the proposed search algorithm is negligible compared to the discretization error itself.

5.6. Numerical robustness

The potential $q_6(x)$ is included to demonstrate two aspects of the numerical robustness of the computation of the accounting function that was presented in Section 3.2.

Firstly, the potential $q_6(x)$ demonstrates that the simple sign check (the case $(\varepsilon\gamma)^2 < 9$ in (14) alone, cf. [24]), is not sufficient. We included in this example Alg. B2s, which is equal to Alg. B2, except that Alg. B2s uses the simple sign check as a root counter, instead of (14) as a whole. Since there are only two samples plus the higher tail in this case, Alg. B2s can count maximally three sign changes in total, whereas the correct accounting function should range to $N = 21$. As we see in Table 3, it detects only one zero crossing at $\kappa = 0$, so that only $\hat{N} = 1$ eigenvalue is returned. (That one numerically computed eigenvalue differs 2.66×10^{-16} from the highest eigenvalue in this spectrum.) It needs no explanation that the inverse KdV-NFT will be significantly different from the original potential if 20 of the 21 eigenvalues are missing from its discrete spectrum. Indeed, the potential $q_6(x)$ is deliberately constructed as an extreme case, but still it illustrates why the case $(\varepsilon\gamma)^2 \geq 9$ in (14) is needed.

Secondly, the potential $q_6(x)$ demonstrates the robustness of counting zero-crossings of the trajectory $\phi(x)$ near a boundary $x = x_d^{\mathrm{u}}$. Because of the even symmetry of $q_6(x)$, 10 out of its 21 eigenfunctions are odd symmetric and thus have a zero-crossing at $x = 0$. For Algs. NR4, B2 and B4 that is at the boundary between two piecewise constant sections of the reconstructed potential. (Since Matslise chooses its own grid, we cannot effectively present Alg. MS18 with this challenge.) With the low number of samples we put this zero-crossing also numerically as close as possible to $x = 0$ and maximise the risk of counting this zero-crossing as two or not at all. We see from the results in Table 3 that Algs. NR4, B2 and B4 each find all eigenvalues, so we conclude that the computation indeed appears robust against this source of error.

The results of $q_6(x)$ furthermore show that the eigenvalue error does not improve when using a fourth order algorithm (Algs. NR4 and B4) rather than a second order algorithm (Alg. B2). This was to be expected, since in this particular case both approximations result in the exact true potential.

6. Conclusion

We presented a numerical algorithm for the computation of eigenvalues of the Non-linear Fourier Transform (NFT) spectrum with respect to the Korteweg–de Vries equation (KdV) from sampled data with vanishing boundaries. The proposed algorithm uses a shooting approach with a Newton–Raphson (NR) based root-finder. Nevertheless, because we make use of Sturm–Liouville (SL) oscillation theory, we can guarantee that our algorithm finds all the eigenvalues. This theory is in the literature usually combined with a bisection-based root-finder, because a bracketing root-finder is the natural choice for the type of information that SL oscillation theory provides. However, NR typically converges faster. Therefore we designed an algorithm that combines SL oscillation theory with an NR-based root-finder. We demonstrated that our algorithm indeed typically saves computation time compared to a bisection based root-finder. Furthermore, we showed that for sampled data with vanishing boundaries, our algorithm is more robust to numerical rounding errors than comparable algorithms that are currently available. Hence our algorithm is more reliable.

CRediT authorship contribution statement

Peter J. Prins: Methodology, Software, Validation, Writing – original draft. **Sander Wahls:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 716669).

References

[1] D.J. Korteweg, G. de Vries, On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves, The Lond. Edinb. Dublin Philos. Mag. J. Sci. Series 5 39 (240) (1895) 422–443, doi:10.1080/14786449508620739.

[2] M. Brühl, H. Oumeraci, Analysis of long-period cosine-wave dispersion in very shallow water using nonlinear Fourier transform based on KdV equation, Appl. Ocean Res. 61 (2016) 81–91, doi:10.1016/j.apor.2016.09.009.

[3] M. Brühl, M. Becker, Analysis of subaerial landslide data using nonlinear Fourier transform based on Korteweg-deVries equation (KdV-NFT), J. Earthq. Tsunami 12 (02) (2018) 1840002, doi:10.1142/s179343111840002x.

[4] A.R. Osborne, Nonlinear ocean waves and the inverse scattering transform, International Geophysics, volume 97, first ed., Academic Press, 2010.

[5] J.L. Hammack, H. Segur, The Korteweg–de Vries equation and water waves. Part 2. Comparison with experiments, J. Fluid Mech. 65 (2) (1974) 289–314, doi:10.1017/s002211207400139x.

[6] W. Zimmerman, G. Haarlemmer, Internal gravity waves: analysis using the periodic, inverse scattering transform, Nonlinear Process. Geophys. 6 (1) (1999) 11–26, doi:10.5194/npg-6-11-1999.

[7] V.V. Temnov, C. Klieber, K.A. Nelson, T. Thomay, V. Knittel, A. Leitenstorfer, D. Makarov, M. Albrecht, R. Bratschitsch, Femtosecond nonlinear ultrasonics in gold probed with ultrashort surface plasmons, Nat. Commun. 4 (1) (2013), doi:10.1038/ncomms2480.
[8] D.S. Ricketts, D. Ham, Electrical Solitons, CRC Press, 2011, doi:10.1201/9781315217802.
[9] L. Hattam, KdV cnoidal waves in a traffic flow model with periodic boundaries, Phys. D 348 (2017) 44–53, doi:10.1016/j.physd.2017.02.010.
[10] L.V. Wijngaarden, Evolving solitons in bubbly flows, in: KdV '95, Springer Netherlands, 1995, pp. 507–516, doi:10.1007/978-94-011-0017-5_29.
[11] J. Misra, M. Patra, A study of solitary waves in a tapered aorta by using the theory of solitons, Comput. Math. Appl. 54 (2) (2007) 242–254, doi:10.1016/j.camwa.2006.12.025.
[12] M. Abdou, A. Hendi, H.K. Alanzi, New exact solutions of KdV equation in an elastic tube filled with a variable viscosity fluid, Stud. Nonlinear Sci. 3 (2) (2012) 62–68, doi:10.5829/idosi.sns.2012.3.2.247. https://idosi.org/sns/3(2)12/2.pdf
[13] D.G. Crighton, Applications of KdV, Acta Appl. Math. 39 (1–3) (1995) 39–67, doi:10.1007/bf00994625.
[14] W. Eckhaus, A.M. van Harten, The Inverse Scattering Transformation and the Theory of Solitons, vol. 50, Elsevier, 1981.
[15] M.J. Ablowitz, H. Segur, Solitons and the Inverse Scattering Transform, SIAM, Philadelphia, PA, USA, 1981, doi:10.1137/1.9781611970883.
[16] C.S. Gardner, J.M. Greene, M.D. Kruskal, R.M. Miura, Method for solving the Korteweg-deVries equation, Phys. Rev. Lett. 19 (19) (1967) 1095–1097, doi:10.1103/physrevlett.19.1095.
[17] P.D. Lax, Integrals of nonlinear equations of evolution and solitary waves, Commun. Pure Appl. Math. 21 (5) (1968) 467–490, doi:10.1002/cpa.3160210503.
[18] V.E. Zakharov, A.B. Shabat, Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media, Soviet Phys. JETP 34 (1) (1972) 62.
[19] M.J. Ablowitz, D.J. Kaup, A.C. Newell, H. Segur, The inverse scattering transform - Fourier analysis for nonlinear problems, Stud. Appl. Math. 53 (4) (1974) 249–315, doi:10.1002/sapm1974534249.
[20] M.J. Ablowitz, J.F. Ladik, A nonlinear difference scheme and inverse scattering, Stud. Appl. Math. 55 (3) (1976) 213–229, doi:10.1002/sapm1976553213.
[21] A.R. Osborne, Non-linear Fourier analysis for the infinite-interval Korteweg–de Vries equation I: an algorithm for the direct scattering transform, J. Comput. Phys. 94 (1991) 284–313, doi:10.1016/0021-9991(91)90223-8.
[22] A. Provenzale, A.R. Osborne, Nonlinear Fourier analysis for the infinite-interval Korteweg–de Vries equation II : numerical tests of the direct scattering transform, J. Comput. Phys. 94 (2) (1991) 314–351, doi:10.1016/0021-9991(91)90224-9.
[23] G. Boffetta, A.R. Osborne, Computation of the direct scattering transform for the nonlinear Schroedinger equation, J. Comput. Phys. 102 (2) (1992) 252–264, doi:10.1016/0021-9991(92)90370-e.
[24] A.R. Osborne, Automatic algorithm for the numerical inverse scattering transform of the Korteweg–de Vries equation, Math. Comput. Simul. 37 (4) (1994) 431–450, doi:10.1016/0378-4754(94)00029-8.
[25] J.K. Brenne, J. Skaar, Design of grating-assisted codirectional couplers with discrete inverse-scattering algorithms, J. Lightwave Technol. 21 (1) (2003) 254–263, doi:10.1109/jlt.2003.808648.
[26] T. Trogdon, S. Olver, B. Deconinck, Numerical inverse scattering for the Korteweg–de Vries and modified Korteweg–de Vries equations, Phys. D 241 (11) (2012) 1003–1025, doi:10.1016/j.physd.2012.02.016.
[27] T. Trogdon, S. Olver, Numerical inverse scattering for the focusing and defocusing nonlinear Schrödinger equations, Proc. Royal Soc. A Math. Phys. Eng. Sci. 469 (2149) (2012) 20120330, doi:10.1098/rspa.2012.0330.
[28] M.I. Yousefi, F.R. Kschischang, Information transmission using the nonlinear Fourier transform, Part II: numerical methods, IEEE Trans. inf. Theory 60 (7) (2014) 4329–4345, doi:10.1109/tit.2014.2321151.
[29] A. Osborne, M. Petti, G. Liberatore, L. Cavaleri, Nonlinear Fourier analysis of laboratory generated, broad-banded surface waves, in: Proc. Int. Conf. Comput. Model. Ocean Eng., Held Venice, 19–23 Sept. 1988, 1988, pp. 99–105.
[30] A.R. Osborne, E. Segre, G. Boffetta, L. Cavaleri, Soliton basis states in shallow-water ocean surface waves, Phys. Rev. Lett. 67 (5) (1991) 592–595, doi:10.1103/physrevlett.67.592.
[31] A.R. Osborne, M. Petti, Numerical inverse-scattering-transform analysis of laboratory-generated surface wave trains, Phys. Rev. E 47 (2) (1993) 1035–1037, doi:10.1103/physreve.47.1035.
[32] A.R. Osborne, M. Petti, Laboratory-generated, shallow-water surface waves: analysis using the periodic, inverse scattering transform, Phys. Fluids 6 (5) (1994) 1727–1744, doi:10.1063/1.868235.
[33] M. Brühl, Direct and inverse nonlinear Fourier transform based on the Korteweg-deVries equation (KdV-NFT), Technischen Universität Carolo-Wilhelmina, Braunschweig, 2013 Ph.D. thesis. https://nbn-resolving.org/urn:nbn:de:gbv:084-14112709287
[34] S. Sugavanam, M.K. Kopae, J. Peng, J.E. Prilepsky, S.K. Turitsyn, Analysis of laser radiation using the nonlinear Fourier transform, Nat. Commun. 10 (1) (2019) 1–10, doi:10.1038/s41467-019-13265-4.
[35] S.K. Turitsyn, I.S. Chekhovskoy, M.P. Fedoruk, Nonlinear Fourier transform for analysis of optical spectral combs, Phys. Rev. E 103 (2) (2021), doi:10.1103/physreve.103.l020202.
[36] S. Wahls, H.V. Poor, Introducing the fast nonlinear Fourier transform, in: 2013 IEEE Int. Conf. on Acoust., Speech and Signal Processing (ICASSP), 2013, pp. 5780–5784, doi:10.1109/ICASSP.2013.6638772.
[37] S. Wahls, H.V. Poor, Fast numerical nonlinear Fourier transforms, IEEE Trans. inf. Theory 61 (12) (2015) 6957–6974, doi:10.1109/TIT.2015.2485944.
[38] S. Wahls, S. Chimmalgi, P.J. Prins, FNFT: A software library for computing nonlinear Fourier transforms, J. Open Source Softw. 3 (23) (2018) 597, doi:10.21105/joss.00597.
[39] P.J. Prins, S. Wahls, Higher order exponential splittings for the fast non-linear Fourier transform of the Korteweg-de Vries equation, in: 2018 IEEE Int. Conf. on Acoust., Speech and Signal Processing (ICASSP), 2018, pp. 4524–4528, doi:10.1109/ICASSP.2018.8461708.
[40] P.J. Prins, S. Wahls, Soliton phase shift calculation for the Korteweg–de Vries equation, IEEE Access 7 (2019) 122914–122930, doi:10.1109/access.2019.2932256.
[41] P.J. Prins, S. Wahls, An accurate $\wr(n^2)$ floating point algorithm for the Crum transform of the KdV equation, Commun. Nonlinear sci. Numer. Simul. 102 (105782) (2021) 1–25, doi:10.1016/j.cnsns.2021.105782.
[42] S. Chimmalgi, P.J. Prins, S. Wahls, Fast nonlinear Fourier transform algorithms using higher order exponential integrators, IEEE Access 7 (2019) 145161–145176, doi:10.1109/access.2019.2945480.
[43] S. Medvedev, I. Vaseva, I. Chekhovskoy, M. Fedoruk, Exponential fourth order schemes for direct Zakharov-Shabat problem, Opt. Express 28 (1) (2019) 20, doi:10.1364/OE.377140.
[44] S. Medvedev, I. Chekhovskoy, I. Vaseva, M. Fedoruk, Fast computation of the direct scattering transform by fourth order conservative multi-exponential scheme, 2019b, (pre-print arXiv:1909.13228v1 [math.NA]b).
[45] S. Medvedev, I. Chekhovskoy, I. Vaseva, M. Fedoruk, Fast sixth-order algorithm based on the generalized Cayley transform for the Zakharov-Shabat system associated with nonlinear schrodinger equation, J. Comput. Phys. (2021) 110764, doi:10.1016/j.jcp.2021.110764.
[46] A. Span, V. Aref, H. Bülow, S. ten Brink, Successive eigenvalue removal for multi-soliton spectral amplitude estimation, J. Lightwave Technol. (2020), doi:10.1109/jlt.2020.2994156.
[47] I. Chekhovskoy, S. Medvedev, I. Vaseva, E. Sedov, M.P. Fedoruk, Introducing phase jump tracking-a fast method for eigenvalue evaluation of the direct Zakharov-Shabat problem, Commun. Nonlinear Sci. Numer. Simul. 96 (2021) 105718, doi:10.1016/j.cnsns.2021.105718.
[48] L. Fermo, C. van der Mee, S. Seatzu, A numerical method to compute the scattering solution for the KdV equation, Appl. Numer. Math. 149 (2020) 3–16, doi:10.1016/j.apnum.2019.07.001.
[49] A. Gudko, A. Gelash, R. Mullyadzhanov, High-order numerical method for scattering data of the Korteweg—de Vries equation, in: J. Phys. Conf. Series, vol. 1677, IOP Publishing, 2020, p. 012011, doi:10.1088/1742-6596/1677/1/012011.
[50] A. Vasylchenkova, D. Salnikov, D. Karaman, O.G. Vasylchenkov, J.E. Prilepskiy, Fixed-point realization of fast nonlinear Fourier transform algorithm for

FPGA implementation of optical data processing, in: M. Bertolotti, A.V. Zayats, A.M. Zheltikov (Eds.), Nonlinear Optics and Applications XII, International Society for Optics and Photonics, vol.11770, SPIE, 2021, pp. 111–120, doi:10.1117/12.2588735.
[51] V. Ledoux, Study of special algorithms for solving Sturm-Liouville and Schrödinger equations, Ghent University, 2007 Ph.D. thesis.
[52] H. Prüfer, Neue Herleitung der Sturm-Liouvilleschen Reihenentwicklung stetiger Funktionen, Math. Ann. 95 (1) (1926) 499–518, doi:10.1007/bf01206624.
[53] NAG Library Code Contributors, NAG Library Routine Document d02kdf, Technical Report, The Numerical Algorithms Group Ltd, Oxford, UK, 2017. https://www.nag.com/numeric/fl/nagdoc_fl26/pdf/d02/d02kdf.pdf
[54] P.B. Bailey, W.N. Everitt, A. Zettl, Algorithm 810: the SLEIGN2 Sturm-Liouville code, ACM Trans. Math. Softw. 27 (2) (2001) 143–192, doi:10.1145/383738.383739.
[55] S. Pruess, C.T. Fulton, Mathematical software for Sturm-Liouville problems, ACM Trans. Math. Softw. 19 (3) (1993) 360–376, doi:10.1145/155743.155791.
[56] L. Ixaru, H.D. Meyer, G.V. Berghe, CP methods for the Schrödinger equation revisited, J. Comput. Appl. Math. 88 (2) (1997) 289–314, doi:10.1016/s0377-0427(97)00218-5.
[57] I. Christov, Internal solitary waves in the ocean: analysis using the periodic, inverse scattering transform, Math. Comput. Simul. 80 (1) (2009) 192–201, doi:10.1016/j.matcom.2009.06.005.
[58] V. Ledoux, M. Van Daele, MATSLISE 2.0: A matlab toolbox for Sturm-Liouville computations, ACM Trans. Math. Softw. 42 (4) (2016) 1–18, doi:10.1145/2839299.
[59] G.L. Lamb, Elements of Soliton Theory, Wiley, 1980.
[60] H. Segur, The Korteweg-de Vries equation and water waves. Solutions of the equation. Part 1, J. Fluid Mech. 59 (4) (1973) 721–736, doi:10.1017/s0022112073001813.
[61] A. Zettl, Sturm-Liouville theory, Mathematical Surveys and Monographs, vol. 121, American Mathematical Soc., 2005, doi:10.1090/surv/121.
[62] V. Ledoux, M. Van Daele, Solution of Sturm–Liouville problems using modified Neumann schemes, SIAM J. Sci. Comput. 32 (2) (2010) 563–584, doi:10.1137/090758398.
[63] J.E. Huss, J.A. Pennline, A Comparison of Five Benchmarks, Technical Memorandum 19870008008, NASA Lewis Research Center, Cleveland, OH, United States, 1987.
[64] S. Blanes, F. Casas, M. Thalhammer, High-order commutator-free quasi-Magnus exponential integrators for non-autonomous linear evolution equations, Comput. Phys. Commun. 220 (2017) 243–262, doi:10.1016/j.cpc.2017.07.016.
[65] A. Alvermann, H. Fehske, P.B. Littlewood, Numerical time propagation of quantum systems in radiation fields, New J. Phys. 14 (105008) (2012) 1–22, doi:10.1088/1367-2630/14/10/105008.
[66] V. Ledoux, M.V. Daele, G.V. Berghe, MATSLISE: a matlab package for the numerical solution of Sturm-Liouville and Schrödinger equations, ACM Trans. Math. Softw. 31 (4) (2005) 532–554, doi:10.1145/1114268.1114273.
[67] T. Baeyens, M. Van Daele, The fast and accurate computation of eigenvalues and eigenfunctions of time-independent one-dimensional Schrödinger equations, Comput. Phys. Commun. 258 (2020) 107568, doi:10.1016/j.cpc.2020.107568.
[68] T. Baeyens, Welcome to the documentation of pyslise. 2021, https://matslise.ugent.be.
[69] G. El, Korteweg – de Vries equation: solitons and undular bores, in: R. Grimshaw (Ed.), Solitary Waves in Fluids, Advances in Fluid Mechanics, vol. 47, WIT Press, UK, 2007, pp. 19–54.
[70] V.E. Zakharov, L.D. Faddeev, Korteweg-de Vries equation: a completely integrable Hamiltonian system, Funct. Anal. Appl. 5 (4) (1972) 280–287, doi:10.1007/bf01086739.