



Path-Following Methods for Calculating Linear Surface Wave Dispersion Relations on Vertical Shear Flows

Peter Maxwell¹ · Simen Å. Ellingsen¹

Received: 31 May 2019 / Accepted: 23 March 2020 / Published online: 30 April 2020
© The Author(s) 2020

Abstract

The path-following scheme in Loisel and Maxwell (SIAM J Matrix Anal Appl 39(4):1726–1749, 2018) is adapted to efficiently calculate the dispersion relation curve for linear surface waves on an arbitrary vertical shear current. This is equivalent to solving the Rayleigh stability equation with linearized free-surface boundary condition for each sought point on the curve. Taking advantage of the analyticity of the dispersion relation, a path-following or continuation approach is adopted. The problem is discretized using a collocation scheme, parametrized along either a radial or angular path in the wave vector plane, and differentiated to yield a system of ODEs. After an initial eigenproblem solve using QZ decomposition, numerical integration proceeds along the curve using linear solves as the Runge–Kutta $F(\cdot)$ function; thus, many QZ decompositions on a size $2N$ companion matrix are exchanged for one QZ decomposition and a small number of linear solves on a size N matrix. A piecewise interpolant provides dense output. The integration represents a nominal setup cost whereafter very many points can be computed at negligible cost whilst preserving high accuracy. Furthermore, a two-dimensional interpolant suitable for scattered data query points in the wave vector plane is described. Finally, a comparison is made with existing numerical methods for this problem, revealing that the path-following scheme is the most competitive algorithm for this problem whenever calculating more than circa 1,000 data points or relative normwise accuracy better than 10^{-4} is sought.

Keywords Path-following method · Rayleigh stability equation · Free surface · Quadratic eigenproblem · Dispersion relation · Numerical continuation

This work was funded by the Norwegian Research Council (FRINATEK) #249740.

✉ Peter Maxwell
peter@allicient.co.uk

¹ Department of Energy and Process Engineering, Norwegian University of Science and Technology, 7491 Trondheim, Norway

1 Introduction

We develop a path-following method to numerically calculate the dispersion relation curve for linear surface waves atop a horizontal current of arbitrary depth dependence, an adaptation of the scheme developed by Loisel and Maxwell [29].

1.1 Background and Motivation

We consider a classical problem from the study of wave-current interactions [38–40], [36, sec. IV], [12,34,35,43], that of linear surface waves upon a vertical shear current. In other words, waves are considered as perturbations to first-order in wave steepness $\epsilon = ka$ upon some zeroth-order depth-dependent horizontal background flow that, in general, has non-constant vorticity. The problem geometry is shown in Fig. 1. Waves in this regime are dispersive with their behaviour being entirely characterized by the dispersion relation, i.e. the dependence of phase velocity $c(\mathbf{k})$ on the wave vector \mathbf{k} whose modulus k is the wavenumber.

When a vertically varying shear current is present beneath a water surface, the dispersion of water waves riding atop it is affected in a complicated way. The well-known expression for the phase velocity of a linear surface wave in inviscid, initially quiescent water of depth h , $c_0(\mathbf{k}) = c_0(k) = \sqrt{(g/k) \tanh(kh)}$, where g is gravitational acceleration, does not hold when considering general horizontal depth-dependent currents of the form $\mathbf{U}(z)$ where z is the depth. In the general case, $c(\mathbf{k})$ is an eigenvalue of the boundary value problem formed by the Rayleigh stability equation (inviscid Orr–Sommerfeld equation) with a bottom Dirichlet boundary condition and a free-surface boundary condition. Only in the cases of quiescent water, uniform current, and strictly linearly varying current is a closed-form expression for $c(\mathbf{k})$ known for all permissible values of \mathbf{k} (and vice versa). Figure 2a, b illustrates quiescent water,

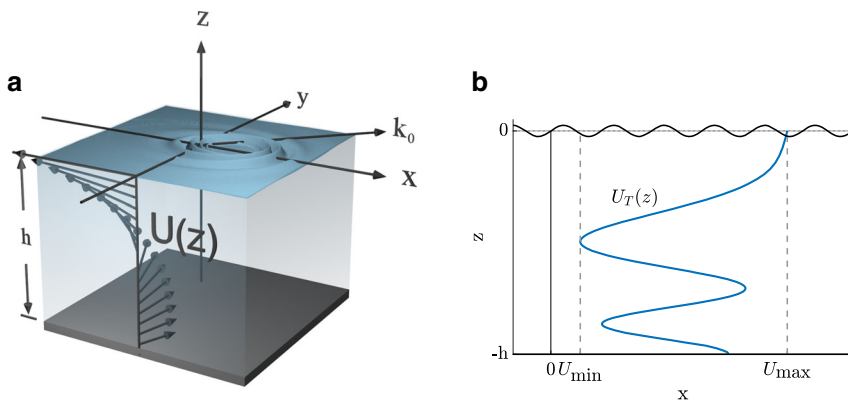


Fig. 1 The problem setting. **a** An illustrative example of dispersive ring waves atop a shear current. An example velocity profile is indicated with z -dependent arrows changing direction in the horizontal plane and an example \mathbf{k} vector as \mathbf{k}_0 . **b** Two-dimensional geometry along x -axis (see also, ‘reduced problem’ defined in Sect. 2.4). Velocity profile shown is $U_T(z)$ as defined in Sect. 2.5

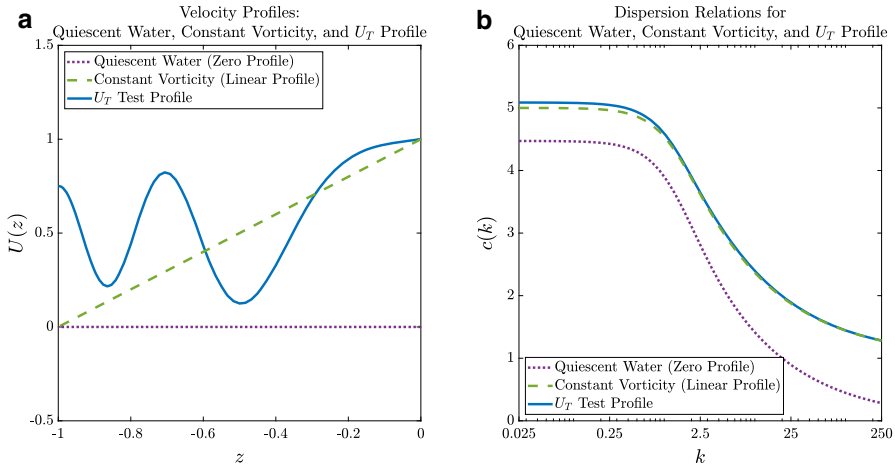


Fig. 2 **a** Plot and dispersion relations for quiescent water, linear velocity profile, and a general velocity profile. Velocity profiles for quiescent water (no background current) in dotted magenta, a linearly varying background current (constant vorticity) in dashed green, and a more general profile U_T in solid blue. **b** Dispersion relations $c(\mathbf{k})$ corresponding to the three velocity profiles in **(a)**

a linear velocity profile, and a general velocity profile, along with the corresponding dispersion relations.

In a theory setting, the typical situation is that $c(\mathbf{k})$ is desired for a given \mathbf{k} , such as when initial value problems are solved via a Fourier transformation in horizontal spatial directions. In a laboratory setting, however, wave-makers operate at a given frequency $\omega = kc$, with k as the dependent parameter. Likewise, when a monochromatic wave propagates in an environment of varying current and bathymetry, its frequency is the conserved quantity whereas the wave vector \mathbf{k} changes according to local conditions. The former setting of determining $c(\mathbf{k})$ from \mathbf{k} , we refer to as the ‘forward problem’; the latter, finding $\mathbf{k}(c)$, we refer to as the ‘backward problem’ (distinct from the ‘inverse problem’ which aims to infer the current profile when $c(\mathbf{k})$ is known). Our primary interest in this paper is in the forward problem but, for purposes of completeness, we also describe how to solve for $k(c)$ using a collocation scheme.

1.2 Existing Methods to Calculate the Dispersion Relation

Unsurprisingly, a number of approaches exist for determining $c(\mathbf{k})$. The oldest approach was to approximate a depth-varying unidirectional velocity profile $U(z)$, where z is the vertical coordinate, by a piecewise linear profile [13,38,39,50]. $U(z)$ could realistically be divided only into a small number of pieces before the formalism would become impractical for manual calculation. The availability of modern computing permits extending the piecewise linear approximation to arbitrarily many pieces [45,57]. Despite the procedure being physically transparent, it converges only as N^{-2} [57], where N is the number of pieces.

Another popular approach is integral approximation methods. An expression was derived by Stewart and Joy [47] (infinite depth) and then generalized by Skop [44]

(finite depth) and further extended by Kirby and Chen [26] (finite depth to second order); these approaches are usually valid to within a few percent for typical oceanographic current profiles. An alternative able to tackle cases with extreme shear was presented recently by Ellingsen and Li [21].

Numerical techniques are applicable to the problem, for example, a shooting method was used by Dong and Kirby [14]. The problem is also amenable to solution with a naive shooting method using a Runge–Kutta numerical integrator and Newton’s method. A recent addition is the so-called Direct Integration Method (DIM) due to Li and Ellingsen [27] which is versatile and arguably the best available method when not so many points are sought on the dispersion relation curve and only moderate accuracy is required. There are various ways to apply spectral methods; example usage of a collocation method is described later in Sect. 3.

It is also possible to calculate a high-order interpolant of the curve using the collocation method for pointwise sampling; this is used as a comparison in our performance tests in Sect. 7 using Chebfun [17] to construct the interpolant.

The path-following method presented in Sect. 4 calculates not only a piecewise Hermite interpolant but also exploits the local analyticity by integrating along the curve. This confers a significant efficiency advantage in most cases.

1.3 Methodology and Approach

Engineering and the natural sciences are replete with eigenproblems for ordinary differential operators which depend on a finite set of parameters. We are interested in problems which are parametrized by a single real variable.

The canonical solution approach involves conversion to an algebraic problem via spatial discretization, which often leads to polynomial or even nonlinear eigenproblems of potentially large dimension. These can be solved using classical techniques for each sought parameter value. This strategy may become prohibitively expensive when the computation must be repeated many times. It can also be difficult to take advantage of the nearness of solutions for small parameter variations, forcing full calculations for each point in the parameter space. An alternative approach is to solve the discretized eigenproblem once for a fixed parameter value then use the local piecewise analyticity of the eigenvalue and eigenvector [4,24,37] to calculate along the solution curve using a path-following or numerical continuation algorithm.

In a more general setting, this comprises a numerical continuation method whereby the parameter-dependent solution is calculated as an implicitly defined curve [3]. Homotopy methods have a similar philosophy but introduce an artificial parameter to parametrize a convex homotopy to map from the solution of an ‘easy’ problem to the solution of the actual problem [28]. These methods tend to use predictor–corrector schemes such as pseudo-arclength continuation or similar approaches. We make reference to the homotopy method in [30] and the invariant subspace methods in [8], [10] as relevant examples. For a recent approach that shares a strong philosophical similarity with the material herein for working with time-varying matrix eigenproblems, albeit using different techniques (look-ahead finite difference formulas), see [54] and [53].

This paper is concerned with repurposing the path-following technique used in [29] to solve for the dispersion relation associated with the geometry described in Sect. 1.1. This problem is particularly suited as a motivating example of the technique: it is conceptually simple, it has an eigenvalue-dependent boundary condition, it is well known from both the waves literature and hydrodynamic stability, and there is a practical requirement for efficient numerical solution.

We summarize our approach as follows. The original eigenproblem is spatially discretized using a collocation method, implicitly incorporating the boundary conditions, to obtain a parameter-varying system of equations that are then differentiated to yield an under-determined system of ODEs. An additional constraint is then included. After performing an initial eigenproblem solve, numerical integration can proceed along the solution curve using linear solves as the Runge–Kutta $F(\cdot)$ function. Specifically, the Dormand–Prince RK5(4)7M method [15, p. 23], which incorporates adaptive step-size control, is used. A piecewise polynomial interpolant provides dense output.

This approach is in contrast to standard predictor–corrector style numerical continuation, see, for example, the description in [3]. Rather than applying a correction at each step, reliance is placed entirely on the adaptive step-size control in the Dormand–Prince 5th-order scheme to ensure that the error is maintained within the desired tolerance. As demonstrated later in Sects. 5 and 7, this approach produces an algorithm that efficiently computes the solution with high accuracy.

1.4 Outline of the Paper

We begin by formally describing the geometry of the physical problem and some problem-specific background in Sect. 2. The collocation scheme used is briefly described in Sect. 3. The path-following method is described in Sect. 4 along radial and angular one-dimensional paths in the \mathbf{k} wave vector plane, and also for general scattered data points in the two-dimensional \mathbf{k} plane. In Sect. 5, we provide numerical results to determine the expected accuracy of the collocation and path-following methods. In Sect. 6, we describe an adaptive partition of unity method to address the problem of the eigenvector becoming numerically singular at the surface with increasing k . In Sect. 7, we evaluate the relative performance characteristics of the various methods and in Sect. 8, we describe how to choose optimal parameters. Finally, in Sect. 9, we provide some conclusions.

2 Preamble

2.1 Problem Description

Wave–current interaction problems are often studied by adopting a modal linear perturbative approach: waves are considered as first-order perturbations of a stationary, incompressible, and inviscid bulk fluid flow. In this context, waves are dispersive with the phase velocity of a wave dependent on the wave vector in a nonlinear manner. The relationship between the wave vector, \mathbf{k} , and the phase velocity, \mathbf{c} , is termed the

dispersion relation and is determined by factors such as water depth and background current.

In our context of first-order free-surface waves atop a vertical shear flow, the problem reduces to finding solutions of the eigenvalue problem formed from the Rayleigh stability equation and appropriate boundary conditions. The Rayleigh equation is a second-order ODE that is equivalent to the Orr–Sommerfeld equation when viscous effects are neglected. A solution to the eigenproblem will yield a $\{k, c, w(z)\}$ triplet where $w(z)$ is the associated eigenfunction. The eigenfunction can be used to reconstruct the velocity and pressure field for the corresponding wave vector [27, sec. 4.4]. Notably, there is substantial overlap with the literature on linear stability theory, e.g. [41, ch. 2] or [16], albeit with different boundary conditions.

Closed-form expressions for the dispersion relation for free-surface waves exist only in specific scenarios such as quiescent water [33], [48, ch. 1]; uniform current [48, ch. 6]; or, a linear velocity profile (constant vorticity) [36, p. 78], [18, eqn. 3.8], [19,20].

To make matters more concrete, we adopt the general approach used in [27,43] and refer the reader to the derivations therein for full detail. For expediency, the approach is only summarized here. The physical model is depicted in Fig. 1a. Dimensional quantities are denoted with an acute accent, e.g. \acute{k} .

The background flow is specified by a velocity profile $\acute{U}(\acute{z}) = (\acute{U}_x(\acute{z}), \acute{U}_y(\acute{z}))$: a two-dimensional vector field describing the bulk fluid velocity in the horizontal plane for a given $\acute{z} \in [-\acute{h}, 0]$ where \acute{h} is the constant fluid depth and the unperturbed surface is at $\acute{z} = 0$. Let

$$\acute{U}_m := \max \left\{ \left| \sup_{\acute{z} \in [-\acute{h}, 0]} \acute{U}_x(\acute{z}) - \inf_{\acute{z} \in [-\acute{h}, 0]} \acute{U}_x(\acute{z}) \right|, \left| \sup_{\acute{z} \in [-\acute{h}, 0]} \acute{U}_y(\acute{z}) - \inf_{\acute{z} \in [-\acute{h}, 0]} \acute{U}_y(\acute{z}) \right| \right\}. \tag{1}$$

We use \acute{h} as a characteristic length scale and \acute{U}_m as a characteristic velocity, to arrive at the following nondimensionalization:

$$\mathbf{U}(z) = \acute{U}(\acute{z}/\acute{h})/\acute{U}_m, \quad z = \acute{z}/\acute{h}, \quad \mathbf{k} = \acute{k}\acute{h}, \quad \mathbf{c} = \acute{c}/\acute{U}_m, \quad h = 1, \tag{2}$$

so that a notional shear strength can be expressed with Froude number, $F = \acute{U}_m/\sqrt{g\acute{h}}$. The velocity profile must be suitably regular, so we impose that $U_x, U_y \in C^2([-1, 0], \mathbb{R})$. We also require \mathbf{c} parallel to \mathbf{k} and we henceforth only refer to scalar $c = |\mathbf{c}|$. It is assumed that the current can influence the waves but not conversely and, for clarity of exposition, we neglect surface tension. We adopt the Ansatz that perturbations are plane waves proportional to $\exp[i(\mathbf{k} \cdot \mathbf{x} - \omega t)]$, where the (scalar) wave angular frequency is $\omega = kc$ with $k = |\mathbf{k}|$, and use a Fourier transform in the horizontal plane (coordinate space quantities indicated with a tilde),

$$\{\tilde{u}, \tilde{v}, \tilde{w}, \tilde{p}\} = \frac{1}{(2\pi)^2} \int_{\mathbb{R}^2} \{u, v, w, p\} e^{i(\mathbf{k} \cdot \mathbf{x} - \omega t)} d\mathbf{k}. \tag{3}$$

The velocity perturbations along the x , y , and z axes are, respectively, $u = u(\mathbf{k}, z)$, $v = v(\mathbf{k}, z)$, and $w = w(\mathbf{k}, z)$, whilst the pressure is $p = p(\mathbf{k}, z)$. The governing equations are the linearized Euler equations and incompressibility condition, e.g. recall [43]:

$$\begin{aligned} ik_x u + ik_y v + w' &= 0, & i(\mathbf{k} \cdot \mathbf{U} - \omega)u + U'_x w &= -ik_x p / \rho, \\ i(\mathbf{k} \cdot \mathbf{U} - \omega)w &= -p' / \rho, & i(\mathbf{k} \cdot \mathbf{U} - \omega)v + U'_y w &= -ik_y p / \rho, \end{aligned}$$

using shorthand $U_x = U_x(z)$, $U_y = U_y(z)$. After rearranging, the Rayleigh equation (4a) is obtained which we write along with the relevant free-surface condition Eq. (4b) (combined kinematic and dynamic boundary condition) and Dirichlet boundary condition for the fluid bottom Eq. (4c),

$$w'' - \frac{\mathbf{k} \cdot \mathbf{U}''}{\mathbf{k} \cdot \mathbf{U} - kc} w - \mu w = 0, \quad z \in (-h, 0); \quad (4a)$$

$$(\mathbf{k} \cdot \mathbf{U} - kc)^2 w' - [(\mathbf{k} \cdot \mathbf{U} - kc)\mathbf{k} \cdot \mathbf{U}' + F^{-2}k^2]w = 0, \quad z = 0; \quad (4b)$$

$$w = 0, \quad z = -h; \quad (4c)$$

for $\mu = k^2$. The velocity and pressure field for a specific wave vector can be recovered by substituting the appropriate eigenpair into [27, sec. 4.4],

$$\begin{aligned} (\mathbf{k} \cdot \mathbf{U} - \omega)w' - \mathbf{k} \cdot \mathbf{U}' w &= ik^2 p / \rho, \\ ik_x [\mathbf{k} \cdot \mathbf{U}' w - (\mathbf{k} \cdot \mathbf{U} - \omega)w'] - ik^2 U'_x w &= k^2 (\omega - \mathbf{k} \cdot \mathbf{U})u, \\ ik_y [\mathbf{k} \cdot \mathbf{U}' w - (\mathbf{k} \cdot \mathbf{U} - \omega)w'] - ik^2 U'_y w &= k^2 (\omega - \mathbf{k} \cdot \mathbf{U})v. \end{aligned} \quad (5)$$

Often, the physical problem is posed so that a scalar $U(z)$ is aligned along the x -axis as $\mathbf{U}(z) = (U(z), 0)$, with scalar k and c , see Fig. 1b. However, there is no difficulty in solving with a general $\mathbf{U}(z)$ by simply projecting $U(z)$ along \mathbf{k} as described in Sect. 2.4. Full three-dimensional considerations only come to the fore when calculating the velocity and pressure field; the eigenvalue problem described by Eq. (4) is always one-dimensional in the sense that the eigenfunctions are a function of one variable, z . This is all straightforward but it is worth stating explicitly to avoid the inherent ambiguities and possible confusion caused by ‘1d’ / ‘2d’ or ‘2d’ / ‘3d’ descriptions, e.g. [21, sec. 1.1]. In particular, we refer to the problem with scalar $U(z)$ as the reduced problem and with vector $\mathbf{U}(z)$ as the general problem. In Sect. 2.4, we note that the reduced problem is equivalent to solving the general problem in a radial ‘slice’ at some fixed angle θ_0 .

So far, we have deliberately avoided specifying which variable is the sought eigenvalue in Eq. (4): it can be chosen as either μ or c , with its counterpart parametrizing the problem and always chosen to be real valued, in a similar manner to [11, sec. 7.4]. Since we are always choosing the parameter to be real valued, we are concerned with a subset of the parametrized spectrum in each case and can plot this arrangement as a function of the parameter.

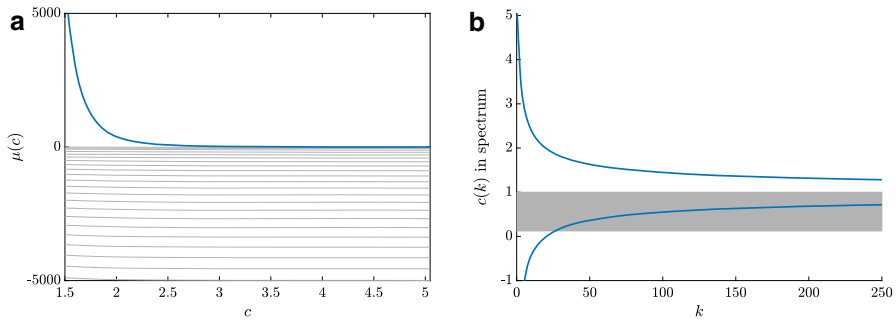


Fig. 3 Parametrized plots of the real spectrum for both the backward and forward problem. **a** $\mu(c)$ spectrum for backward problem. Notice only one positive eigenvalue and series of negative eigenvalues distributed $\sim n^2$. **b** Plot showing the sought positive and also negative solution branches along with the essential spectrum

- The spectrum for $\mu(c)$, for c in some suitable interval, is comprised of a countably infinite set of eigenvalues. The dominant eigenvalue, $\mu_1 = k^2$, in this case is the only positive eigenvalue, and corresponds to a propagating wave (for $\pm k$). The negative eigenvalues correspond to the countably infinite set of discrete k arranged along the imaginary axis and are not mentioned further. See Fig. 3a.
- The spectrum for $c(k)$, for $k > 0$, has both discrete and essential parts (c such that $U(z) - c = 0$, regular singular points of the ODE). In particular, there are two discrete eigenvalues usually labelled $c_+(k)$ and $c_-(k)$. Further discussion is available in [16, p. 137]; in particular, our interest shall be in computing the dominant ‘regular inviscid solution’, the smooth $c_+(k)$ curve. For some k , $c_+(k)$ may be located within the same interval as the essential spectrum and, therefore, can be difficult to identify within numerical solution sets, see Fig. 3b.

The c_{\pm} exhibit odd symmetry with respect to k : $-c_+(k) = c_-(-k)$; the associated eigenfunctions are also odd symmetric, $-w_+(k, z) = w_-(-k, z)$. Furthermore, it should be noted that the one-sided limits $\lim_{k \rightarrow 0^+} c(k)$ and $\lim_{k \rightarrow 0^-} c(k)$ exist but depend on angle θ . So $c_{\pm}(k)$ is, in general, undefined at the origin.

2.2 Problem Types: Forward, Backward, and Inverse

We distinguish three types of problem.

1. For velocity profile, $U(z)$, and collection of wavenumbers, $\{k^{(j)}\}_{j=1}^J$, calculate associated velocities $\{c(k^{(j)})\}_{j=1}^J$. We denote this as the forward problem.
2. For velocity profile, $U(z)$, and collection of velocities, $\{c^{(j)}\}_{j=1}^J$, calculate associated wavenumbers $\{k(c^{(j)})\}_{j=1}^J$. We denote this as the backward problem.
3. For collection of wavenumber and phase velocity pairs, $\{(k^{(j)}, c^{(j)})\}_{j=1}^J$, determine velocity profile, $U(z)$. We denote this as the inverse problem, which is of an entirely different nature and herein not considered further.

Both the forward and backward problems usually amount to calculating sufficiently many $\{k^{(j)}, c^{(j)}\}$ pairs as to adequately specify the full dispersion relation for a given

velocity profile. For practical purposes, these problems are almost always posed as Eq. (3) with exponent of the form $i(kx - \omega(k)t)$ (see [36, p. 77, eqn. 4.6], [26, eqn. 1], [43, eqn. 2.4], etc). This, by default, presupposes solving the forward problem. Furthermore, handling of critical layers as in Sect. 3.4 is, from a numerical standpoint, easier for the forward problem. Therefore, most of this paper concerns solution of the forward problem. Unless otherwise specified, we shall compute c_+ since c_- follows automatically from symmetry.

There are a few exceptions when solving the backward problem is more suitable such as for wave problems with periodic or stationary time dependence, e.g. in ship waves. Hence, for purposes of completeness, we also describe solving the backward problem using a rudimentary collocation scheme and the basic properties of the spectrum.

2.3 Summary of Numerical Schemes

For ease of reference, we denote the various numerical schemes used or described in this paper:

- *CL-c* Collocation scheme for the forward problem, see Sect. 3.3.
- *CL-k* Collocation scheme for the backward problem, see Sect. 3.2.
- *PF-R-r-c* A path-following scheme with dense output to solve the forward reduced problem along a fixed angle θ_0 in the \mathbf{k} -plane, see Sect. 4.2.1.
- *PF-R-a-c* A path-following scheme with dense output to solve the forward reduced problem for $c_{k_0}(\theta)$ along a fixed circle of radius $k = k_0$ with varying θ in the \mathbf{k} -plane, which we term the angular solution, see Sect. 4.2.2.
- *PFmp-R-{r,a}-c* An illustrative scheme using a single high-precision QZ solve to improve accuracy of PF-R-{r,a}-c, detailed in Sect. 4.2.5.
- *PF-G-c* A scheme which solves the forward general problem using PF-R-r-c and PF-R-a-c to allow rapid interpolation with two-dimensional scattered data query points in the \mathbf{k} -plane, see Sect. 4.3.

The CL-c and CL-k schemes incur an eigenvalue calculation for each query point, so the computational cost will increase linearly with the number of query points. The arrangement of points in the \mathbf{k} -plane for the CL-schemes can be random (scattered data) without affecting computational cost.

The PF-R-r-c path-following algorithm is two staged: it first performs numerical integration to calculate control points along a radial ‘slice’ at fixed angle θ_0 , which incurs a nominal initial computational cost; query points on that slice are then calculated using a Hermite interpolant. Although the computational cost of interpolation is linear in the number of query points, it is so lightweight as to be of almost negligible cost in most situations: so after the initial computation, very many query points can be calculated efficiently. The angular PF-R-a-c scheme is similar but instead calculates along a circular path at a fixed radius k_0 .

The PF-G-c scheme is more involved because we accept query points in the \mathbf{k} -plane with no assumption on arrangement, i.e. scattered data. A naive approach would incur a complete first stage calculation of PF-R-r-c for every query point, which is

unacceptable. The PF-G-c scheme instead precalculates a two-dimensional polar grid of suitable control points and then can interpolate for query points at negligible cost.

Note that all methods presented can also make available the eigenfunction $w(z)$ so that the velocity and pressure field can be reconstructed using Eq. (5).

2.4 General to Reduced Problem

The general problem Eq. (4) can be simplified to a reduced problem by projecting U along \mathbf{k} , see, for example, [36, p. 77], [43, p. 566]. Define the scalar velocity profile for the reduced problem as $U_{\mathbf{k}}(z) = (1/k)\mathbf{k} \cdot \mathbf{U} = \cos(\theta)U_x(z) + \sin(\theta)U_y(z)$ where θ is taken to be the standard angular polar coordinate for \mathbf{k} .

2.5 Velocity Profiles and Parameters Used

For later numerical tests, we define a test velocity profile U_T for the reduced problem, as shown in Fig. 1b,

$$U_T(z) := \frac{\gamma}{2}(1 + \delta z) \cos(\beta(-z)^\alpha) + 1/2, \quad \alpha = 2, \beta = 4\pi, \gamma = 1, \delta = 1/2. \quad (6)$$

We choose the physical depth $\hat{h} = 20$ and shear Froude number as $F^2 = 0.05$. We choose nondimensional $k \in I_k := [0.025, 250]$. This corresponds broadly to gravity waves in the air–water interface regime [33, p. 4] with shortest period ≈ 0.2 s. The function is chosen as a suitable test candidate because it has several stationary points and cannot be approximated exactly over a finite-dimensional polynomial basis.

For the figures produced from PF-G-c shown later in Fig. 7, we use $U_T(z)$ along the x axis and an approximation of a flow in the Columbia River on the y axis, which we denote U_{CR} . U_{CR} is defined by a sixth-degree polynomial as in [32]. For the plots in Sect. 4.3, U_{CR} is scaled to have $F^2 \approx 0.01$; it is also used in Sect. 3.4 with $F^2 = 0.05$.

For use later, define

$$U_{\min} := \inf_{z \in [-1, 0]} \{U(z)\} \text{ and } U_{\max} := \sup_{z \in [-1, 0]} \{U(z)\}.$$

3 Collocation Method for Solving the Dispersion Relation

3.1 Discretization of the Equations

We use sans serif notation to indicate matrices (uppercase) and vectors (lowercase), e.g. U or w , to distinguish from their continuous counterparts.

Let $\zeta_j = \cos((j-1)\pi/N_z)$, $j = 1 \dots N_z + 1$ be the Chebyshev–Gauss–Lobatto collocation points on $[-1, 1]$ (second-kind points). We use the change of variable $z = (h/2)(\zeta - 1)$ to map ζ_j to $z_j \in [-1, 0]$ in nondimensional coordinates and let $\mathbf{z} := [z_1, \dots, z_{N_z+1}]^T$ be the associated column vector. Let D be a corresponding square differentiation matrix (in practice, we calculate the first- and second-order

differentiation matrices, D and D^2 , using `polldif.m` from the Weideman–Reddy library [55] and then apply the ‘negative sum trick’ as detailed in [6]). We define vector discretizations of the velocity profile,

$$\mathbf{u} = [u_1, \dots, u_{N_z+1}]^T := [U(z_1), \dots, U(z_{N_z+1})]^T.$$

Quantities u' and u'' are similarly defined. Let $w := w(z)$. Define the diagonal matrices $U := \text{diag}(\mathbf{u})$, $U' := \text{diag}(u')$, and $U'' := \text{diag}(u'')$.

The problem is a two-point boundary value problem so is amenable to the standard ‘row-replacement’ strategy, see, for example, [52, ch. 7]. Specifically, we aim to construct eigenvalue equations which discretize the governing equation Eq. (4a) using the ‘interior’ rows 2 through N_z of the differentiation and coefficient matrices. The free-surface boundary condition Eq. (4b) is incorporated in the first row of the matrices. The bottom Dirichlet Eq. (4c) boundary condition is accounted for by eliminating the last row and column in the matrices, as in [52, ch. 7]. For notational convenience, we define ‘interior’ differentiation and velocity profile matrices as $D_{\text{int}} = D_{lm}$, $U_{\text{int}} = U_{lm}$, $U'_{\text{int}} = U'_{lm}$, and $U''_{\text{int}} = U''_{lm}$ with $l = 2 \dots N_z$, $m = 1 \dots N_z$ (in other words, eliminating the first and last rows, and last column). We also define a free-surface differentiation vector as the first row of D , $\mathbf{d}_f := D_{1m}$, $m = 1 \dots N_z$, again without the last column.

3.2 Backward Reduced Problem (CL-k)

Treating c as a parameter and k as the eigenvalue then using the projection from Sect. 2.4, we obtain a regular Sturm–Liouville problem on $z \in [-1, 0]$ with eigenvalue $\mu = k^2$,

$$\left(\frac{d^2}{dz^2} - \frac{U''(z)}{U(z) - c} \right) w(z) = \mu w(z), \quad z \in (-1, 0); \quad (7a)$$

$$(U - c)^2 w' - [(U - c)U' + F^{-2}]w = 0, \quad z = 0; \quad (7b)$$

$$w = 0, \quad z = -1. \quad (7c)$$

Let $q_j := u''_j / (u_j - c)$, $\mathbf{q}_{\text{int}} := [q_2 \dots q_{N_z}]$, and $Q_{\text{int}} = \text{diag}(\mathbf{q}_{\text{int}})$. The discretization of Eq. (7a) proceeds in an obvious manner,

$$\mu \mathbf{w} = (D_{\text{int}}^2 - Q_{\text{int}})\mathbf{w} = \mathbf{R}\mathbf{w}, \quad \text{for } \mathbf{R} := D_{\text{int}}^2 - Q_{\text{int}}. \quad (8)$$

Discretizing Eq. (7b) into a row vector gives,

$$\mathbf{f} := (u_1 - c)^2 \mathbf{d}_f - [((u_1 - c)u'_1 + F^{-2}), 0, \dots, 0]. \quad (9)$$

Write

$$\mathbf{A} = \begin{bmatrix} \mathbf{f} \\ \mathbf{R} \end{bmatrix}, \quad \text{and } \mathbf{B} = \text{diag}(0, 1, \dots, 1), \quad (10)$$

to obtain the generalized eigenvalue problem,

$$Aw = \mu Bw. \tag{11}$$

Note that the only effect of B is to ensure that the row of A with the free-surface boundary condition is set equal to zero and is not dependent on the eigenvalue. Equation (11) can be solved in several ways, e.g. using MATLAB's implementation of QZ as $\text{eig}(A, B)$.

For a given c , the solution of Eq. (11) yields the greatest N_z eigenvalues from the countably infinite set of discrete μ_j eigenvalues, ordered $\mu_1 > \mu_2 > \dots$, associated with the infinite-dimensional operator. The only positive eigenvalue is μ_1 . Since $k = \sqrt{\mu}$, μ_1 gives the only real k . Our interest is in propagating waves; therefore, $k_{\pm} = \pm\sqrt{\mu_1}$ is the sought solution. This is shown in Fig. 3a.

3.3 Forward Reduced Problem (CL-c)

Now, treating k as a parameter and c as the eigenvalue, we rewrite the reduced problem Eq. (7) to emphasize the quadratic eigenvalue dependence in the free-surface boundary condition,

$$\left(U \left(\frac{d^2}{dz^2} - k^2 \right) - U'' - c \left(\frac{d^2}{dz^2} - k^2 \right) \right) w = 0, \quad z \in (-1, 0); \tag{12a}$$

$$c^2 w' + c(-2Uw' + U'w) + (U^2 w' - UU'w - F^{-2}w) = 0, \quad z = 0; \tag{12b}$$

$$w = 0, \quad z = -1. \tag{12c}$$

We initially discretize Eq. (12a) as

$$(U_{\text{int}}(D_{\text{int}}^2 - k^2) - U''_{\text{int}} - c(D_{\text{int}}^2 - k^2))w = 0. \tag{13}$$

We proceed by expressing the free-surface condition as coefficients of the powers of c ,

$$f_2 = d_f \tag{14a}$$

$$f_1 = -2u_1 d_f + [u'_1, 0, \dots, 0] \tag{14b}$$

$$f_0 = u_1^2 d_f - [u'_1 u_1 + F^{-2}, 0, \dots, 0]. \tag{14c}$$

In the same manner, we now separate Eq. (13) into powers of c ,

$$R_2 = 0 \tag{15a}$$

$$R_1 = -(D_{\text{int}}^2 - k^2)l \tag{15b}$$

$$R_0 = U_{\text{int}}(D_{\text{int}}^2 - k^2 I) - U_{\text{int}}'' \quad (15c)$$

Define

$$A_2 = \begin{bmatrix} f_2 \\ R_2 \end{bmatrix}, A_1 = \begin{bmatrix} f_1 \\ R_1 \end{bmatrix}, \text{ and } A_0 = \begin{bmatrix} f_0 \\ R_0 \end{bmatrix}. \quad (16)$$

To obtain the sought solution, we solve the quadratic eigenproblem,

$$(c^2 A_2 + c A_1 + A_0)w = 0. \quad (17)$$

There are several techniques to solve the quadratic eigenvalue problem. A linearization and then using a QZ decomposition works well in this setting. MATLAB's `polyeig(A2, A1, A0)` implements such a linearization although some care must be taken. In particular, the A_2 matrix is badly rank-deficient. As a consequence, the QZ algorithm will return infinite and large-but-finite eigenvalues which are merely artefacts of the numerical method and must be removed.

From the solution of Eq. (17), it is the c_+ eigenvalue that is sought. The result set will also include an eigenvalue for c_- and other finite eigenvalues that are regular singular points of the underlying ODE. These regular singular points, or critical layer points, are sampled from the essential spectrum: c such that $U(z) - c = 0$ for some $z \in [-1, 0]$, see [16] and Fig. 3b.

Let α_j for $j = 1, \dots, N_j \leq 2N_z$ be the eigenvalues obtained from the solution of Eq. (17) and w_j be the corresponding eigenvectors. The infinite and very large in magnitude α_j are discarded. The solution is found by determining the j such that $\alpha_j = c_+$ can be identified (to within numerical error). There are two possible cases, either

- (i) c_+ is the greatest element of α_j such that $c_+ \notin [U_{\min}, U_{\max}]$, or
- (ii) $c_+ \in [U_{\min}, U_{\max}]$ (a critical layer has been encountered).

The former case is trivial: merely pick the largest eigenvalue that is not in $[U_{\min}, U_{\max}]$. The latter case may be handled using the methodology described in Sect. 3.4 below.

Although the collocation method converges exponentially fast with respect to increasing N_z , the effects of roundoff error and ill-conditioning in the differentiation matrices soon start to dominate [11]. As shown later in Sect. 5, in `double` precision, the relative error in calculated c is at best around 10^{-10} for this problem. Techniques such as domain decomposition may be used to reduce the error further but since these techniques are relatively well known, implementation details have been omitted.

3.4 Solving with Critical Layers in the Forward Problem

In most cases, one of the α_j corresponds to c_+ and another to c_- . The remaining α_j eigenvalues are sampled from the essential spectrum and shall be described as spurious eigenvalues to match the terminology used in [11, ch. 7] and [45]. The underlying eigenvalue problem is quadratic and, as might be expected, the spurious eigenvalues

occasionally manifest as complex conjugate pairs due to collisions with other spurious eigenvalues. In pathological situations, an eigenvalue collision between c_+ and a spurious eigenvalue will create a complex conjugate pair causing the sought c_+ to disappear from the solution set. This is dealt with in more detail in Sects. 3.4.1 and 3.4.2 below.

A similar approach to filtering of spurious eigenvalues by looking for singular behaviour in the eigenfunction away from the endpoints was used in [45]. However, it does not perform robustly when the order of discretization is too large or if the sought c_+ is too close to a spurious eigenvalue.

3.4.1 Eigenvector Filtering

The sought α_j is usually within a good error margin of the exact c_+ but identifying this from the spurious eigenvalues is nontrivial. The eigenvectors corresponding to these spurious eigenvalues tend to exhibit features that permit identification aside from singular behaviour in the interior of the interval. These include

1. nonzero near $z = -h$;
2. nonzero imaginary component;
3. for suitably large k , the decay near the surface is not sufficiently ‘fast’;
4. dependence on the order of discretization; and,
5. singular behaviour in the interior of the domain (detected using MATLAB’s `findpeaks()`).

The approach taken is to successively eliminate eigenvectors from the set $\{w_j\}_j$ until the only eigenvectors remaining are those corresponding to the sought α_j . Computationally cheap methods are used as a first-pass filter (everything in the above list except peak-finding). If there are still spurious eigenvalues remaining then the using the peak-finding strategy will give the sought solution.

This process is demonstrated using the negative of the Columbia River profile, $-U_{CR}$ and $N_z = 80$: Fig. 4a shows an example of the full set of eigenvectors for all finite eigenvalues at a fixed k ; Fig. 4b shows the remaining eigenvectors after the initial filtering step; finally, Fig. 4c shows the sought eigenvector after all filtering steps are completed.

3.4.2 Eigenvalue Collisions

Although the spurious eigenvalues are sampled from the essential spectrum, when solving the finite-dimensional Eq. (17), this distinction is essentially lost. Eigenvalues can collide forming complex conjugate pairs. In particular, the sought c_+ eigenvalue may collide with a spurious eigenvalue causing c_+ to disappear from the result set and so the method in Sect. 3.4.1 would be ineffectual. By detecting these points and temporarily adjusting the size of the discretization so that the spurious eigenvalues are shifted, the sought eigenvalue reappears (similar in spirit to [11, sec. 7.5]). The detection is done by looking for complex conjugate pairs and matching-up against the eigenvector filtering, i.e. there should be an eigenvector corresponding to c_+ , if it is missing then a collision should be expected. This is demonstrated in Fig. 5.

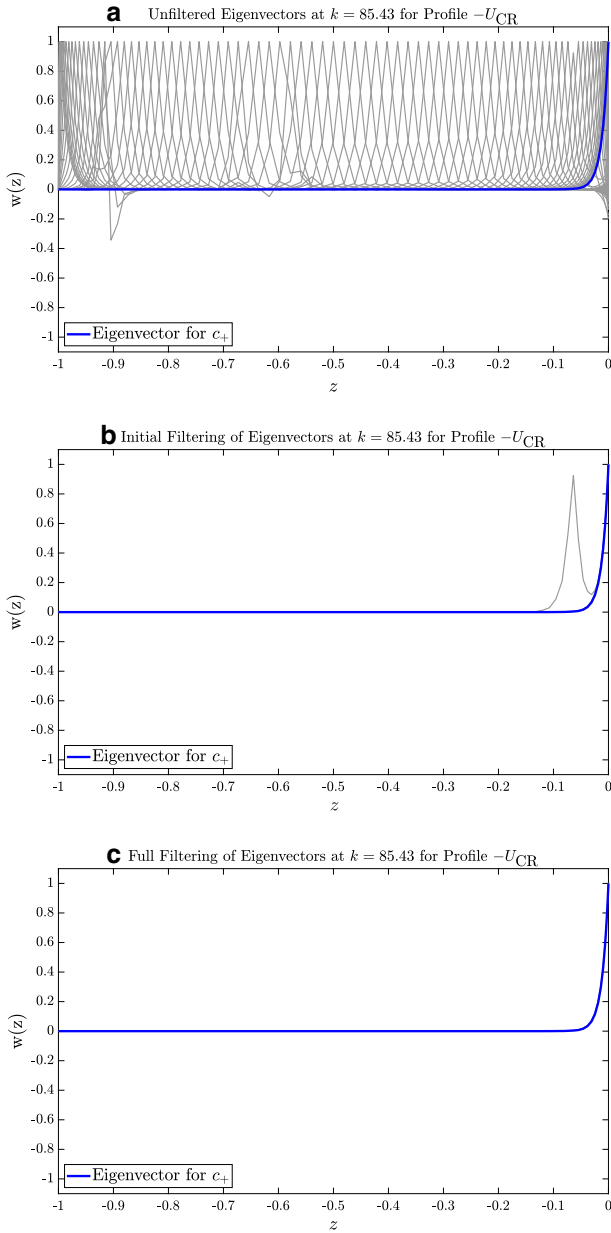


Fig. 4 Eigenvector filtering for $-U_{CR}$ profile at $k \approx 85$. **a** Eigenvectors for all finite eigenvalues. **b** Eigenvectors remaining after first-pass filtering. **c** Eigenvectors after filtering using peak-detection

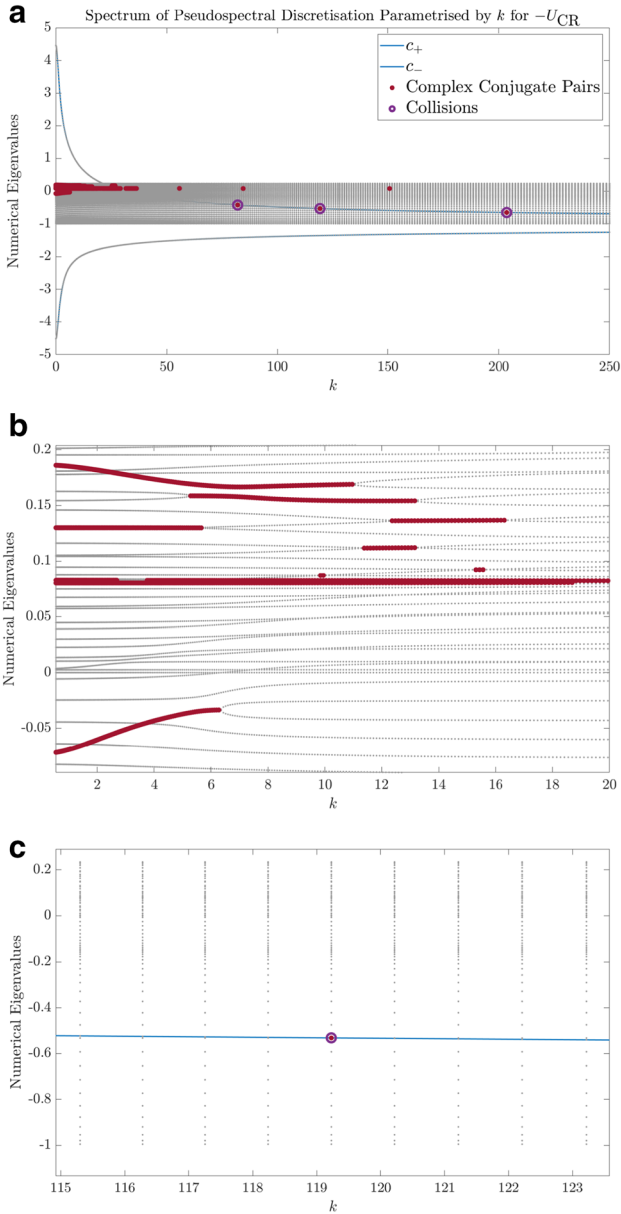


Fig. 5 $-U_{CR}$ profile. **a** Parametrised finite-dimensional spectrum indicating discrete c_{\pm} in solid blue lines. The grey dots are spurious ‘eigenvalues’ that really belong to the essential spectrum. The red dots indicate complex conjugate pairs (only real part shown). The purple circles indicate detected collisions (when the eigenvector corresponding to c_+ has disappeared). **b** Zoomed-in plot of eigenvalue paths. Red indicates complex conjugate pair (only real part shown). **c** Zoomed-in plot of a collision where the eigenvector corresponding to c_+ eigenvalue disappears from the solution set

4 Path-Following Method for Calculating the Dispersion Relation Curve

4.1 Review of Loisel–Maxwell Path-Following Method for the Field of Values

In [29], the authors describe a path-following method to calculate the field of values boundary of a matrix, which we now briefly summarize. It concerns the solution of a parametrized Hermitian eigenvalue problem (which bounds the projection of the field of values onto the real axis),

$$H(e^{i\tau}A)u(\tau) = \lambda(\tau)u(\tau) \text{ for } A \in \mathbb{C}^{n \times n}, 0 \neq u \in \mathbb{C}^n, \lambda \in \mathbb{R}, \tau \in [0, 2\pi), \quad (18)$$

taking $(\lambda(\tau), u(\tau))$ as the dominant eigenpair, where $H(A) := (1/2)(A + A^*)$ is the Hermitian part, $S(A) := (1/2)(A - A^*)$ is the skew-Hermitian part of the given matrix A , and A^* is the conjugate-transpose. Here, and in the remainder of the paper, the overdot notation is used to indicate derivatives with respect to the problem parameter. This is to emphasize the parameter-varying or “time-varying” nature of the problems.

Note that Eq. (18) is well-defined except perhaps for a finite number of τ_j due to elementwise analyticity of the elements of $H(e^{i\tau}A)$ and the analyticity, up to ordering, of the eigenvalue and eigenvectors. Differentiating Eq. (18) gives

$$H(e^{i\tau}A)\dot{u}(\tau) - \dot{\lambda}(\tau)u(\tau) - \lambda(\tau)\dot{u}(\tau) = -iS(e^{i\tau}A)u(\tau). \quad (19)$$

Since the system is under-determined, an additional constraint that $u(\tau)$ must be perpendicular to its (elementwise) derivative is included: $u(\tau)^* \dot{u}(\tau) = 0$. In [29], this was derived by differentiating the algebraic normalization constraint $u(\tau)^* u(\tau) = 1$ and fixing the ‘phase-factor’. In retrospect, this appears to have been the same rationale applied in [49, p. 57] and similar to the normalization considerations in [9]. However, the normalization constraint is violated in the numerical integration and must be manually enforced at each step. In this sense, the derivation of the constraint does not offer a satisfactory explanation of its action.

An anonymous referee points out that upon application of the Fredholm alternative, it is possible to express the derivative of the eigenvalue as

$$\dot{\lambda}(\tau) = i \frac{u(\tau)^* S(e^{i\tau}A)u(\tau)}{u(\tau)^* u(\tau)} = i \frac{(u(\tau), S(e^{i\tau}A)u(\tau))}{(u(\tau), u(\tau))}, \quad (20)$$

where (\cdot, \cdot) is the usual inner product. Note that this expression is none other than the Hadamard first variation formula from [49, p. 57, eqn. 1.72].

Including the constraint gives the fully determined system,

$$\begin{cases} H(e^{i\tau}A)\dot{u}(\tau) - \dot{\lambda}(\tau)u(\tau) - \lambda(\tau)\dot{u}(\tau) = -iS(e^{i\tau}A)u(\tau) \\ u(\tau)^* \dot{u}(\tau) = 0, \end{cases} \quad (21)$$

which can be rewritten in matrix form,

$$\begin{bmatrix} H(e^{i\tau}A) - \lambda(\tau)I & -u(\tau) \\ -u(\tau)^* & 0 \end{bmatrix} \begin{bmatrix} \dot{u}(\tau) \\ \dot{\lambda}(\tau) \end{bmatrix} = \begin{bmatrix} -iS(e^{i\tau}A)u(\tau) \\ 0 \end{bmatrix}. \tag{22}$$

The system described by Eq. (22) can be solved for $[\dot{u}(\tau)^* \dot{\lambda}(\tau)^*]^*$ using a linear solver and used as the $F(\cdot)$ function for a Runge–Kutta numerical integrator, which then generates control points along the curve. The authors use the Dormand–Prince RK5(4)7M method [15, p. 23] and interpolation method of Shampine [42, p. 148]. The near-interpolant solution from this method is 5th-order accurate. There is comprehensive discussion presented of sharp points and flat segments on the boundary, and relevant event processing to detect these exceptional points in the path-following algorithm.

4.2 Path-Following Method for Forward Reduced Problem

We now extend the same process to the quadratic eigenvalue problem posed in Sect. 3.3. Recall Eq. (17),

$$\left(c^2(k)A_2(k) + c(k)A_1(k) + A_0(k) \right) w(k) = 0, \tag{23}$$

which upon differentiation (indicated with overdot) with respect to k gives

$$\begin{aligned} & \left(c^2(k)\dot{A}_2(k) + c(k)\dot{A}_1(k) + \dot{A}_0(k) + 2c(k)\dot{c}(k)A_2(k) + \dot{c}(k)A_1(k) \right) w(k) \\ & + \left(c^2(k)A_2(k) + c(k)A_1(k) + A_0(k) \right) \dot{w}(k) = 0. \end{aligned} \tag{24}$$

We further impose that $w(k)^* \dot{w}(k) = 0$. Writing in matrix form,

$$\underbrace{\begin{bmatrix} \left(c^2(k)A_2(k) + c(k)A_1(k) + A_0(k) \right) & \left(2c(k)A_2(k) + A_1(k) \right) w(k) \\ w(k)^* & 0 \end{bmatrix}}_M \begin{bmatrix} \dot{w}(k) \\ \dot{c}(k) \end{bmatrix} = \begin{bmatrix} -\left(c^2(k)\dot{A}_2(k) + c(k)\dot{A}_1(k) + \dot{A}_0(k) \right) w(k) \\ 0 \end{bmatrix}. \tag{25}$$

This is the general form in which the structure is clear. In the subsections below, we perform the same derivation but include the specific expressions for the radial and angular paths including boundary conditions.

The computational approach taken is analogous to [29]: an initial eigenpair $\{c_0, w_0\}$ is calculated using CL-c. Then using Eq. (25) to solve for $[\dot{w}(k)^* \dot{c}(k)^*]^*$, numerical integration can proceed along the curve in both directions. Hermite interpolation can then be used to query at arbitrary k .

Care must be taken to ensure that M remains nonsingular throughout the parameter ranges required. This may be done computationally at each Runge–Kutta step by, for example, checking the reciprocal condition number. Due to the bordered matrix structure, M only becomes singular under certain degenerate circumstances such as encountering nonsimple eigenvalues. For background on use of bordered matrices in numerical continuation, see, for example, [25] or [10]. From knowledge of the problem at hand, we know that the only region in which there may be nonsimple eigenvalues is for k such that $c(k) \in [U_{\min}, U_{\max}]$ (whereupon a critical layer is encountered). This is discussed further in Sect. 4.2.4. For other problems, more careful handling of points of higher algebraic multiplicity would likely be required.

4.2.1 System of Equations Along Radial Slice at Fixed θ (PF-R-r-c)

For PF-R-r-c, we fix angle $\theta = \theta_0$ and parametrize by k . Thus, we are in the setting of the reduced problem with the constant velocity profile being the relevant reduced velocity profile, $U_k(z)$. Writing Eq. (7a) in matrix form with c as eigenvalue and explicit dependence on parameter k ,

$$\left(U_{\text{int}} - c(k)I_{\text{int}} \right) D_{\text{int}}^2 w(k) - U_{\text{int}}'' w(k) = k^2 \left(U_{\text{int}} - c(k)I_{\text{int}} \right) w(k). \quad (26)$$

For notational succinctness, we use the shorthand $c = c(k)$ and $w = w(k)$. Differentiating Eq. (26) with respect to k (indicated by an overdot) gives

$$\left(U_{\text{int}} - cI_{\text{int}} \right) D_{\text{int}}^2 \dot{w} - \dot{c} D_{\text{int}}^2 w - U_{\text{int}}'' \dot{w} = 2k \left(U_{\text{int}} - cI_{\text{int}} \right) w - \dot{c} k^2 w + k^2 \left(U_{\text{int}} - cI_{\text{int}} \right) \dot{w}. \quad (27)$$

The free-surface condition can be written as

$$(u_1 - c)^2 d_f w + [(-u_1 u_1' + c u_1' - F^{-2}), 0, \dots, 0] w = 0. \quad (28)$$

Differentiating Eq. (28) with respect to k ,

$$\begin{aligned} & -2\dot{c}(u_1 - c)d_f w + (u_1 - c)^2 d_f \dot{w} \\ & + [(\dot{c}u_1'), 0, \dots, 0] w + [-(u_1'(u_1 - c) + F^{-2}), 0, \dots, 0] \dot{w} = 0. \end{aligned} \quad (29)$$

Upon rearranging terms, we define the block matrices:

$$P(k, c) := \begin{bmatrix} (u_1 - c)^2 d_f + [-(u_1'(u_1 - c) + F^{-2}), 0, \dots, 0] \\ (U_{\text{int}} - cI_{\text{int}}) D_{\text{int}}^2 - U_{\text{int}}'' - k^2 (U_{\text{int}} - cI_{\text{int}}) \end{bmatrix}, \quad (30a)$$

$$Q(k, c) := \begin{bmatrix} -2(u_1 - c)d_f + [(u_1'), 0, \dots, 0] \\ -D_{\text{int}}^2 + k^2 I_{\text{int}} \end{bmatrix}, \text{ and} \quad (30b)$$

$$R(k, c) := \begin{bmatrix} 0 \\ 2k(U_{\text{int}} - cI_{\text{int}}) \end{bmatrix}, \quad (30c)$$

so that the system of ODEs can now be written in matrix form as

$$\underbrace{\begin{bmatrix} P(k, c) & Q(k, c)w(k) \\ -w(k)^* & 0 \end{bmatrix}}_{M(k, c, w)} \underbrace{\begin{bmatrix} \dot{w}(k) \\ \dot{c}(k) \end{bmatrix}}_{\dot{v}(k)} = \underbrace{\begin{bmatrix} R(k, c)w(k) \\ 0 \end{bmatrix}}_{b(k, c, w)}. \quad (31)$$

In the same manner as in Sect. 3.1, the bottom boundary condition is imposed by eliminating the corresponding row and columns [52, ch. 7], only the free-surface is included explicitly.

4.2.2 System of Equations Along Angular Circle at Fixed k (PF-R-a-c)

For PF-R-r-c, the angle θ and hence the velocity profile was held constant. For PF-R-a-c, we instead hold k constant and seek to use a θ angular dependence. Therefore, we must also specify the parametrization of the velocity profile.

$$U(\theta, z) = \cos(\theta)U_x(z) + \sin(\theta)U_y(z).$$

So that in matrix form

$$U(\theta) = \cos(\theta)U_x + \sin(\theta)U_y,$$

and, upon differentiation with respect to θ (indicated with an overdot),

$$\dot{U}(\theta) = -\sin(\theta)U_x + \cos(\theta)U_y.$$

Our starting point is the same, we use Eq. (26) but instead hold k constant and take the derivative with respect to θ . Temporarily adopting the abbreviated notation $U_{\text{int}} = U_{\text{int}}(\theta)$, $w = w(\theta)$, and $c = c(\theta)$:

$$\begin{aligned} & (\dot{U}_{\text{int}} - \dot{c}_{\text{int}})D_{\text{int}}^2 w + (U_{\text{int}} - c_{\text{int}})D_{\text{int}}^2 \dot{w} - \dot{U}_{\text{int}}'' w - U_{\text{int}}'' \dot{w} \\ & = k^2((\dot{U}_{\text{int}} - \dot{c}_{\text{int}})w + (U_{\text{int}} - c_{\text{int}})\dot{w}). \end{aligned} \quad (32)$$

As before, the free-surface condition is Eq. (28), which we take the derivative of with respect to θ using the shorthand $\dot{u}_1 = \dot{u}_1(\theta)$,

$$\begin{aligned} & 2(u_1 - c)(\dot{u}_1 - \dot{c})d_f w + (u_1 - c)^2 d_f \dot{w} \\ & + [-(\dot{u}'_1(u_1 - c) + u'_1(\dot{u}_1 - \dot{c})), 0, \dots, 0]w \\ & + [-(u'_1(u_1 - c) + F^{-2}), 0, \dots, 0]\dot{w} = 0. \end{aligned} \quad (33)$$

In a similar manner as before, we define the block matrices:

$$P(\theta, c) := \begin{bmatrix} (u_1 - c)^2 d_f + [-(u'_1(u_1 - c) + F^{-2}), 0, \dots, 0] \\ (U_{\text{int}} - c_{\text{int}})D_{\text{int}}^2 - U_{\text{int}}'' - k^2(U_{\text{int}} - c_{\text{int}}) \end{bmatrix}, \quad (34a)$$

$$Q(\theta, c) := \begin{bmatrix} -2(u_1 - c)\mathbf{d}_f + [(u_1'), 0, \dots, 0] \\ -D_{\text{int}}^2 + k^2 l_{\text{int}} \end{bmatrix}, \text{ and} \quad (34b)$$

$$R(\theta, c) := \begin{bmatrix} -2(u_1 - c)\dot{u}_1 \mathbf{d}_f + [(\dot{u}_1 u_1' + \dot{u}_1'(u_1 - c)), 0, \dots, 0] \\ -\dot{U}_{\text{int}} D_{\text{int}}^2 + \dot{U}_{\text{int}}'' + k^2 \dot{U}_{\text{int}} \end{bmatrix}, \quad (34c)$$

so that the system of ODEs can now be written in matrix form as

$$\underbrace{\begin{bmatrix} P(\theta, c) & Q(\theta, c)\mathbf{w}(\theta) \\ -\mathbf{w}(\theta)^* & 0 \end{bmatrix}}_{M(\theta, c, \mathbf{w})} \underbrace{\begin{bmatrix} \dot{\mathbf{w}}(\theta) \\ \dot{c}(\theta) \end{bmatrix}}_{\dot{\mathbf{v}}(\theta)} = \underbrace{\begin{bmatrix} R(\theta, c)\mathbf{w}(\theta) \\ 0 \end{bmatrix}}_{\mathbf{b}(\theta, c, \mathbf{w})}. \quad (35)$$

Note that the P and Q matrix have the same structure as in Eq. (31), it is R that changes.

4.2.3 Path-Following Algorithm Specification for Reduced Problem

We describe the algorithm for PF-R-r-c, the algorithm for PF-R-a-c follows in the obvious manner. Using the definitions of P, Q, R from Eq. (30a,30b,30c) define matrix and vector functions,

$$M(k, c, \mathbf{w}) := \begin{bmatrix} P(k, c) & Q(k, c)\mathbf{w}(k) \\ -\mathbf{w}(k)^* & 0 \end{bmatrix}, \quad \mathbf{b}(k, c, \mathbf{w}) := \begin{bmatrix} R(k, c)\mathbf{w}(k) \\ 0 \end{bmatrix}. \quad (36)$$

Given a candidate $\mathbf{v}(k) := [\mathbf{w}(k)^* \ c(k)^*]^*$, define the Runge–Kutta $F(\cdot)$ function as

$$F \left(k, \begin{bmatrix} \mathbf{w}(k) \\ c(k) \end{bmatrix} \right) = \begin{bmatrix} \dot{\mathbf{w}}(k) \\ \dot{c}(k) \end{bmatrix} = M(k, c, \mathbf{w})^{-1} \mathbf{b}(k, c, \mathbf{w}). \quad (37)$$

$F(\cdot)$ can be easily obtained using a linear solver such as LU decomposition. The inverse M^{-1} is never calculated explicitly.

The algorithm requires an initial $\mathbf{v}_0 = \mathbf{v}(k_0)$ calculated using CL-c. As in [29], the Dormand–Prince RK5(4)7M method [15, p. 23] and Hermite interpolation strategy of Shampine [42, p. 148] are used. We use automatic step-size control as described in [22, p. 167]. For an interval $[k^{(j)}, k^{(j+1)}]$ with midpoint $k^{(\text{mid})}$, the integrator produces control points $\{\mathbf{v}^{(j)}, \dot{\mathbf{v}}^{(j)}, \mathbf{v}^{(j+1/2)}, \mathbf{v}^{(j+1)}, \dot{\mathbf{v}}^{(j+1)}\}$ where $\mathbf{v}^{(j)} = \mathbf{v}(k^{(j)})$ and $\mathbf{v}^{(j+1/2)} = \mathbf{v}(k^{(\text{mid})})$. Thus, after numerical integration, a solution set of $\mathbf{v}^{(j)}, \dot{\mathbf{v}}^{(j)}$, and $\mathbf{v}^{(j+1/2)}$ is obtained upon which piecewise Hermite interpolation can be performed. If both $c(k)$ and the eigenvector $\mathbf{w}(k)$ is required then interpolation is over N_z length vectors; if only $c(k)$ is required then interpolation is only one-dimensional. Example output is shown in Fig. 6 (one-dimensional output).

4.2.4 Nonsimple Eigenvalues and Critical Layer Processing for PF-R-{r,a}-c

As alluded to in Sect. 4.2, for $c \notin [U_{\min}, U_{\max}]$ then we know that for this problem the eigenvalues remain simple. However, there is essential spectrum in the interval

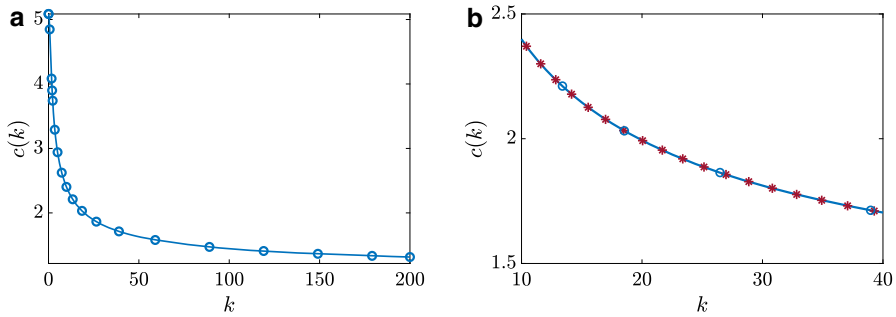


Fig. 6 Numerical integration of dispersion relation curve for velocity profile U_T . Tolerance for integrator was 10^{-6} . **a** Dormand-Prince control points indicated by blue circles. To make the plot slightly clearer, an interval $k \in [0, 200]$ was used instead of $k \in I_k = [0.025, 250]$ as stipulated earlier in Sect. 2.5. **b** Zoomed-in. Sample interpolant query points shown with red asterisks

$[U_{\min}, U_{\max}]$, and so if the sought discrete c is in that interval then it poses practical difficulties. This was addressed for the collocation method in Sect. 3.4.

For the path-following method, if $c \in [U_{\min}, U_{\max}]$ then M may become nearly singular and it is possible to encounter nonsimple eigenvalues. Furthermore, the eigenvalues in this interval are sampled from the essential spectrum so they can be located very closely together; standard techniques to reliably process through this region would require a very small step-size indeed. Therefore, if within an integration step it is found that $c \in [U_{\min}, U_{\max}]$ then the path-following algorithm uses CL- c with the critical layer filtering described in Sect. 3.4 to calculate $v(k)$ at each stage of the Runge–Kutta integration and then a linear solve to calculate the derivative as usual. This is obviously slower but a large step-size may be used and it ensures that the path-following algorithm tracks the sought $c_+(k)$ curve.

4.2.5 PFmp-R- $\{r,a\}$ - c : Improving Accuracy for PF-R- $\{r,a\}$ - c

As shall be described in Sect. 5, the error in the CL- methods is determined almost entirely by roundoff error incurred during the solution of the quadratic eigenvalue problem in `double` precision. In our numerical tests, the path-following algorithm essentially maintains the same error as is present in the initial v_0 . By calculating $v(k_0)$ in high-precision arithmetic then executing the path-following schemes in `double` precision as normal, an improvement in accuracy of two to three orders of magnitude is obtained. This is discussed further in Sect. 5.1.

4.3 Path-Following Method for Forward General Problem (PF-G- c)

The PF-R- r - c and PF-R- a - c algorithms can be combined to create an efficient algorithm that can process scattered data query points, which we describe below.

- (i) First, PF-R- a - c is executed at some nominal $k = k_0$ and interpolation points at angles $\{\theta^{(j)}\}_{j=1}^J$ are calculated. See Fig. 7a, b.

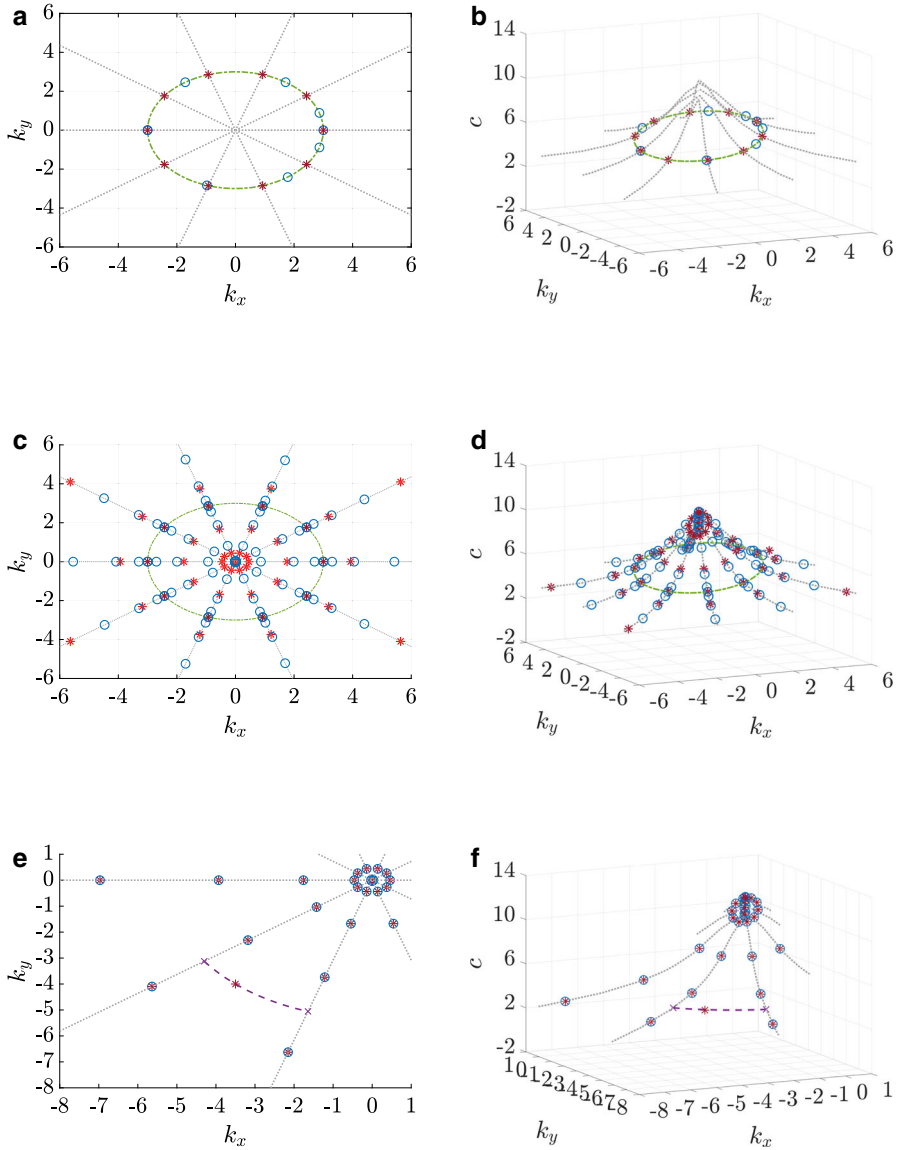


Fig. 7 Steps of PF-G. Intervals for k_x and k_y were chosen for purposes for visual clarity only; in practice, these are chosen as is appropriate for the problem. **a** Planar plot of PF-R-a used at radius k_0 , interpolated at angles $\theta^{(j)}$. Blue circles are control points, red asterisks are interpolation points. Angles indicated in dotted grey. **b** 3d plot as panel (a). **c** Planar plot of PF-R-r used along each $\theta^{(j)}$. Blue circles are control points, red asterisks are interpolation points (the $k^{(i)}$). **d** 3d plot as panel (c). **e** Planar plot showing interpolation of query point. Magenta crosses indicate the k_q radius on each radial slice at angles $\theta^{(l)}$, $\theta^{(l+1)}$. Red asterisk indicates the query point (k_q, θ_q) . **f** 3d plot as panel (e)

- (ii) The results from step i. are used as the initial v_0 values for PF-R-r-c calculating radially along each $\theta^{(j)}$. The curve on each radial slice is then interpolated at predefined $\{k^{(i)}\}_{i=1}^I$ points. The control points for each radial slice are then replaced with the control points at these fixed $k^{(i)}$ (we do not calculate new midpoint values). So there is now a two-dimensional polar grid at angles $\theta^{(j)}$ and radii $k^{(i)}$. See arrangement in Fig. 7c, d.
- (iii) For an arbitrary query point (k_q, θ_q) , the nearest angles $\theta^{(l)}, \theta^{(l+1)}$ and radii $k^{(m)}, k^{(m+1)}$ are identified. The interpolant on radial slices at angles $\theta^{(l)}, \theta^{(l+1)}$ are calculated at radius k_q . Equation (35) is then used to calculate the angular derivatives. Finally, (cubic) interpolation is performed in an angular direction for angle θ_q to obtain the solution. See Fig. 7e, f.

Note that after steps i. and ii. are calculated once, only step iii. needs to be performed for further query points, in a similar manner as PF-R-{r,a}-c.

There is a loss in accuracy because of the required use of cubic interpolation—due to not having the midpoint—rather than the 4th-order interpolation used in PF-R-{r,a}-c. However, for these purposes, it is not particularly significant. For clarity, we omit further analysis of PF-G: it is broadly similar to PF-R-r-c and does not add anything to the discussion.

5 Convergence and Error Estimates

It is well known that for sufficiently smooth solutions, spectral methods converge exponentially fast or with ‘spectral accuracy’ [11, ch. 1,2]. However, roundoff error poses a significant challenge for collocation methods due to the interaction of ill-conditioned matrices with commonly used `double` precision calculations [6,11]. We adopt a heuristic strategy to estimate the accuracy of each algorithm.

5.1 Dependence of Eigenvalue Accuracy on Order N_z

To determine accuracy depending on N_z , we first calculate a reference dispersion relation $R_{\text{ref}} = \{(k_{\text{ref}}^{(i)}, c_{\text{ref}}^{(i)})\}_{i=1}^I$ for the $k_{\text{ref}}^{(i)}$ values distributed along the test interval I_k . This is done in high-precision arithmetic, using the `Advanpix` library [1], for $N_{\text{ref}} = 384$; this size of matrix exceeds what would be used in practice.

We calculate the relative normwise error in a candidate dispersion relation R_{cand} (with $k_{\text{cand}}^{(i)} = k_{\text{ref}}^{(i)}$) as

$$\epsilon := \frac{\| [c_{\text{cand}}^{(1)} - c_{\text{ref}}^{(1)}, \dots, c_{\text{cand}}^{(J)} - c_{\text{ref}}^{(J)}] \|_{\infty}}{\| [c_{\text{ref}}^{(1)}, \dots, c_{\text{ref}}^{(J)}] \|_{\infty}}. \quad (38)$$

This is done for for the U_T velocity profile using the CL-c, PF-R-r-c, PFmp-R-r-c, and DIM algorithms. The CL and PF methods reduce error with spectral accuracy until roundoff error starts to dominate. The high-precision initial calculation for the PFmp algorithm avoids this roundoff error and it can be seen that the path-following method

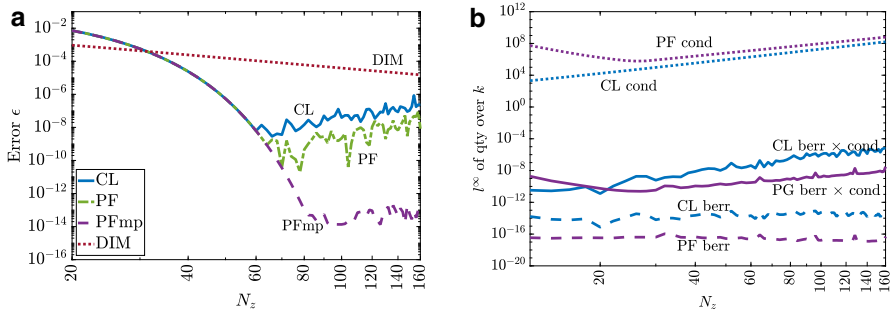


Fig. 8 Plots of error in eigenvalue computations and backwards error + condition number estimates. **a** Log-log plot of normwise relative error in candidate algorithms depending on N_z . The collocation method and path-following algorithm reduce error with spectral accuracy until roundoff error starts to dominate around $N_z \approx 65$. Path following+MP which uses a high-precision initial result maintains broadly the same low error despite the actual path-following calculations being performed in double precision (as is expected, the tolerance for the adaptive step-size control in the Dormand–Prince integration had to be set to around 10^{-15} to achieve this). DIM reduces error as predicted, as $\mathcal{O}(N_z^{-2})$. **b** Backwards error, condition number, and error estimate (backwards error \times condition number). The time series for the quadratic eigenproblem solves for CL shown in blue, the linear solves for PF in magenta. From the condition numbers, shown with dotted lines, it can be seen that CL solves are of the same order of magnitude. The backwards error, shown in dashed lines, shows the linear solves in PF are appreciably more backwards stable. The error estimate clearly favours PF

itself retains this improved accuracy even in double precision. DIM is included for indicative purposes. See Fig. 8a.

A possible explanation for this can be found in comparison of the backwards error and conditioning of the quadratic eigenvalue solve used for the CL-schemes compared to the linear solves predominantly used in PF-schemes. Although it is not a direct comparison—the linear solves are used to calculate a derivative, not the value itself—it may lend some insight. The backward error for the linear solve can be calculated with [23, eqn. 1.2] and the condition number in the usual manner:

$$\eta_L = \frac{\|\mathbf{b} - \mathbf{M}\mathbf{x}\|_2}{\|\mathbf{M}\|_2 \|\mathbf{v}\|_2 + \|\mathbf{b}\|_2}, \quad \kappa_L = \|\mathbf{M}^{-1}\|_2 \|\mathbf{M}\|_2. \quad (39)$$

The backward error for the quadratic eigenproblem solve can be calculated using [51, thm. 1, eqn. 2.3] and the condition number using [51, thm. 5, eqn. 2.15]:

$$\eta_Q = \frac{\|(\mathbf{A}_2 c^2 + \mathbf{A}_1 c + \mathbf{A}_0)\mathbf{w}\|_2}{(\|\mathbf{A}_2\|_2 |c|^2 + \|\mathbf{A}_1\|_2 |c| + \|\mathbf{A}_0\|_2) \|\mathbf{w}\|_2}, \quad (40a)$$

$$\kappa_Q = \frac{(\|\mathbf{A}_2\|_2 |c|^2 + \|\mathbf{A}_1\|_2 |c| + \|\mathbf{A}_0\|_2) \|\mathbf{w}_l\|_2 \|\mathbf{w}\|_2}{|c| |\mathbf{w}_l^* (2\mathbf{A}_2 c + \mathbf{A}_1) \mathbf{w}|}, \quad (40b)$$

where \mathbf{w}_l is a corresponding left-eigenvector. Noting the usual inequality [51, eqn. 1.3],

$$\text{forward error} \leq \text{condition number} \times \text{backward error}. \quad (41)$$

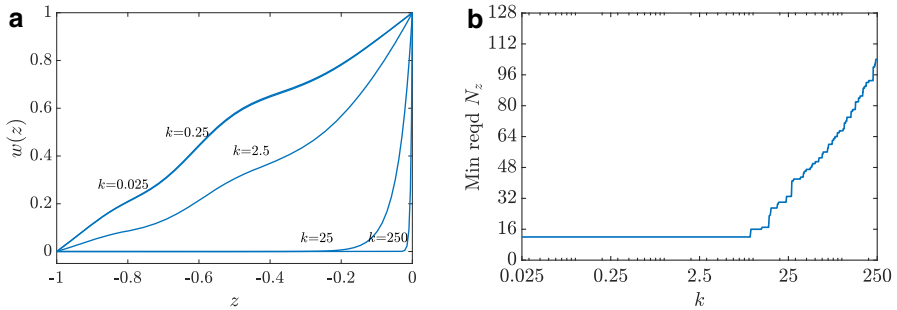


Fig. 9 Plot of eigenvectors for various k and convergence properties. **a** Eigenvector plot for $k = \{0.025, 0.25, 2.5, 25, 250\}$. As k increases, the solution becomes numerically singular near $z = 0$. **b** Lin-log plot showing required N_z for Chebyshev series to converge depending on k . The increasingly numerically singular behaviour of the eigenvector requires much larger N_z to reach convergence

For a range of N_z , we calculate the $\|\cdot\|_\infty$ of the backwards errors η_L , η_Q and condition numbers κ_L , κ_Q over the k vector. This then permits calculating the product from Eq. (41). This is shown in Fig. 8b. The condition numbers are of the same magnitude, $\kappa_L \approx \kappa_Q$, but the backwards error for the linear solves in PF-R-r-c clearly smaller, $\eta_L \ll \eta_Q$. Although a direct comparison cannot be made, this suggests the path-following method has favourable numerical properties.

5.2 k -Dependent Convergence

As can be seen from Fig. 9a, the eigenvectors become numerically singular at the surface as k increases, implying that increasingly many basis polynomials are required to approximate the solution. This can be tested using a similar algorithm as in [5] to determine when the Chebyshev series has converged. Specifically, we calculate an envelope then use a histogram to locate the plateau convergence region. We then determine the required N_z to reach convergence for a range of k values, as shown in Fig. 9b. For shorter wavelengths, much higher N_z is required to reach convergence and so requiring more computational resources. This problem can be entirely ameliorated, as described in Sect. 6.

6 Adaptive Depth and Partition of Unity

It is clear from the results in Sect. 5.2 that as k increases, the required N_z becomes infeasibly large due to the singular behaviour of the eigenfunction. This can be avoided by using a smaller h so that $h \ll 1$ for higher k , on the following rationale. We expect that the eigenfunction decays roughly as e^{kz} . Therefore, we can estimate the depth below which the eigenfunction is effectively zero, for numerical purposes. Let δ be the tolerance below which numerical values are considered zero, e.g. the “machine epsilon”. Let $h_\delta(k) := \min\{1, -\log(\delta)/k\}$ be an estimate of the depth, taking into account the finite depth, at which the eigenfunction decays below tolerance δ for a given k .

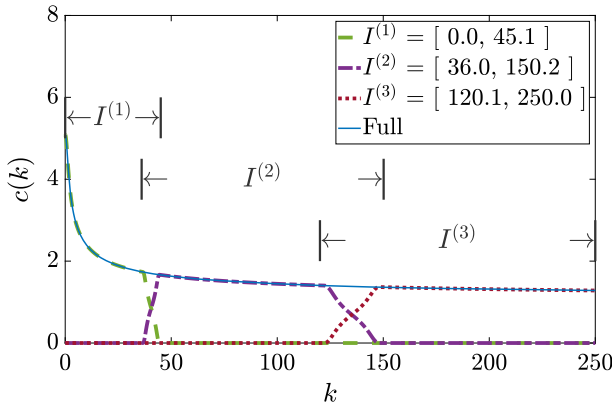


Fig. 10 Partition of unity method for dispersion relation

The CL-r scheme can be adapted for each calculated k . For a given k , we can set $h = h_\delta(k)$. The calculated eigenvalue for the phase velocity will be correct automatically. The eigenvector may be mapped back onto the original interval on any suitably large set of z points chosen on the $[-1, 0]$ interval using barycentric interpolation [7]; the eigenvector will be zero for $z < -h_\delta(k)$.

This procedure becomes less obvious when considering the PF-r scheme because it would require mapping the entire system at each Runge–Kutta step. To avoid this, we instead split the k domain into several, partially overlapping, subintervals for which the chosen depth is suitable for all k in that subinterval. The path-following algorithm is then used on each subinterval independently with the appropriate depth. To combine the subintervals and avoid loss of smoothness in the computed dispersion relation, a partition of unity method is used on the overlaps.

We seek a scheme to choose subintervals $I_k^{(j)} = [k_a^{(j)}, k_b^{(j)}]$ and corresponding depths $h^{(j)}$ that is both simple and easy to implement. For some $k^{(j)} \in I_k^{(j)}$, we seek that $h_{\min}^{(j)} \leq h_\delta(k^{(j)}) \leq h_{\max}^{(j)}$ for $h_{\min}^{(j)} = C_{\min} h^{(j)}$ and $h_{\max}^{(j)} = C_{\max} h^{(j)}$, where C_{\min}, C_{\max} are constants controlling the proportion of the $[-h^{(j)}, 0]$ interval that $-h_\delta(k^{(j)})$ should be within. In our computations, we found that $C_{\min} = 0.3$ and $C_{\max} = 0.8$ worked well. The subintervals and associated depths are then calculated as

$$I_k^{(0)} = \left[0, \frac{\log(\delta)}{C_{\max}} \right], \quad h^{(0)} = 1,$$

$$I_k^{(j)} = \left[\frac{\log(\delta)}{C_{\min}^{j-1}}, \frac{\log(\delta)}{C_{\min}^j C_{\max}} \right], \quad h^{(j)} = \min \left\{ 1, \frac{1}{2} \left(C_{\min}^{j-1} + C_{\min}^j C_{\max} \right) \right\}.$$

This generates intervals $I_k^{(j)}$ in such a manner that $k_b^{(j)} > k_a^{(j+1)}$, i.e. there is some overlap in the intervals. We use the partition of unity method described in [2] to join the subintervals. This is demonstrated in Fig. 10.

7 Performance Analysis

We compare the performance of the path-following algorithm against the collocation scheme CL-c from Sect. 3.3 and also some methods that were referenced in Sect. 1.2: DIM [27], the naive shooting method (SH), and the high-order interpolant method (CHEB) using Chebfun [17] with collocation for pointwise sampling.

7.1 Asymptotic Performance Analysis

There are two variables which control the expected computation time for the candidate algorithms: the number of z evaluation points, N_z , and the number of query points N_q . Since N_z determines accuracy and is dependent on algorithm choice, we assume N_z is set appropriately for each algorithm to achieve similar accuracy. Therefore, our primary concern shall be how the algorithms scale with N_q .

DIM will incur a fixed per-point computational cost that depends on N_z , which we denote $\sigma_{\text{DIM}}(N_z)$. So, the expected cost is $\mathcal{O}(\sigma_{\text{DIM}}(N_z)N_q)$. Similarly, the collocation algorithm incurs a fixed per-point computational cost which also depends on N_z , $\sigma_{\text{CL}}(N_z)$. So, the expected cost is $\mathcal{O}(\sigma_{\text{CL}}(N_z)N_q)$. A naive shooting method would also have an expected cost of the form $\mathcal{O}(\sigma_{\text{SH}}(N_z)N_q)$. These estimates are valid for both the reduced and general problems.

In contrast, the reduced path-following algorithm incurs an initial computational cost dependent on N_z , $\sigma_{\text{PF-NI}}(N_z)$, whereafter there is a very light-weight per-point cost, $\sigma_{\text{PF-Q}} \ll \sigma_{\text{PF-NI}}(N_z)$. Therefore, the expected computational cost is $\mathcal{O}(\sigma_{\text{PF-NI}}(N_z) + \sigma_{\text{PF-Q}}N_q)$. The general path-following algorithm is similar with only the coefficients changed.

Finally, constructing a high-order interpolant using Chebfun with the collocation method as the pointwise sampling would incur an initial setup cost dependent on both N_z and required number of sampling points N_s , $\sigma_{\text{CHEB-S}}(N_z, N_s)$; thereafter, there is a very low per-point cost, $\sigma_{\text{CHEB-Q}} \ll \sigma_{\text{CHEB-S}}(N_z, N_s)$. So the expected computation cost is $\mathcal{O}(\sigma_{\text{CHEB-S}}(N_z, N_s) + \sigma_{\text{CHEB-Q}}N_q)$.

This is summarized in the following table, assuming the eigenvector output is not required:

Algorithm	Computational Cost
DIM	$\mathcal{O}(\sigma_{\text{DIM}}(N_z)N_q)$
CL-c & CL-G-c	$\mathcal{O}(\sigma_{\text{CL}}(N_z)N_q)$
SH (Shooting)	$\mathcal{O}(\sigma_{\text{SH}}(N_z)N_q)$
PF-R-r-c	$\mathcal{O}(\sigma_{\text{PF-NI}}(N_z) + \sigma_{\text{PF-Q}}N_q)$
CHEB (Chebfun + CL-c)	$\mathcal{O}(\sigma_{\text{CHEB-S}}(N_z, N_s) + \sigma_{\text{CHEB-Q}}N_q)$

It immediately becomes clear that if $\sigma_{\text{PF-NI}}(N_z)$ is not too large and $\sigma_{\text{PF-Q}}$ is sufficiently small then as N_q increases, the path-following algorithms are the most efficient.

Analogous behaviour can be expected from the high-order interpolant with Chebfun but numerical experiments show that the initial setup cost $\sigma_{\text{CHEB-S}}(N_z, N_s)$ is prohibitively high. This is discussed at further length in Sect. 7.2 below.

7.2 Numerical Experiments to Determine Algorithm Performance

The asymptotic complexity claims are confirmed by practical testing. For clarity, we only test with the reduced problem. By measuring the time taken for each algorithm to compute the dispersion relation for differing N_q , we can determine the computational complexity in relation to N_q as shown in the log–log plot, Fig. 11. Each algorithm can be executed with different parameter choices that influence accuracy. As such, we calibrated algorithms to produce output at three different accuracies (measured as relative normwise error using Eq. (38)): $\epsilon \approx 10^{-4}$ ('low' accuracy), $\epsilon \approx 10^{-7}$ ('medium' accuracy), and $\epsilon \approx 10^{-10}$ ('high' accuracy). The naive shooting algorithm (SH) is tested only at the medium and high accuracy levels. The high-order interpolant method (CHEB) is only performed for the high accuracy level.

As seen in the results, the path-following algorithm is asymptotically at least two orders of magnitude faster than DIM, SH, and the collocation scheme. The break-even point in N_q at which the path-following scheme becomes faster than DIM is $N_q \approx 1000$ for the low accuracy solution; for the medium and high accuracy levels, the path-following method is always faster.

Comparing the path-following algorithm against the only other scheme that uses interpolation, CHEB, it can be seen that they share similar characteristics. However, the initial setup cost for the path-following scheme is much lower than for the high-order interpolant scheme. This is because, in this setup, a quadratic eigenvalue problem must be solved for each point on the curve used to construct the high-order interpolant. Each quadratic eigenproblem solve incurs a QZ decomposition on a size $2N_z$ matrix. The path-following algorithm avoids most of this cost by swapping the eigenproblem solves for linear solves on size N_z matrices, which is considerably faster.

There is an issue with the interpolation for the path-following scheme that can be seen after around 10^5 points: the computation time starts to increase linearly. This is due to having to determine what the relevant control points are to use for the Hermite interpolation for a given query point. In the implementation used, this was done using MATLAB's `histcounts()` function. If some ordering of the query points were assumed, this cost could be eliminated but we do not assume order in the query points.

8 Guidance on Optimal Parameter Choices

Optimal parameter choices are predicated on two key properties: the required accuracy and the anticipated number of query points.

As can be observed from Fig. 11, the path-following algorithm is most effective when higher accuracy and at least a moderate number of query points are required. The nominal setup cost caused by the initial quadratic eigenproblem solve and numerical integration is dependent on the order of differentiation matrix used, N_z . So this should

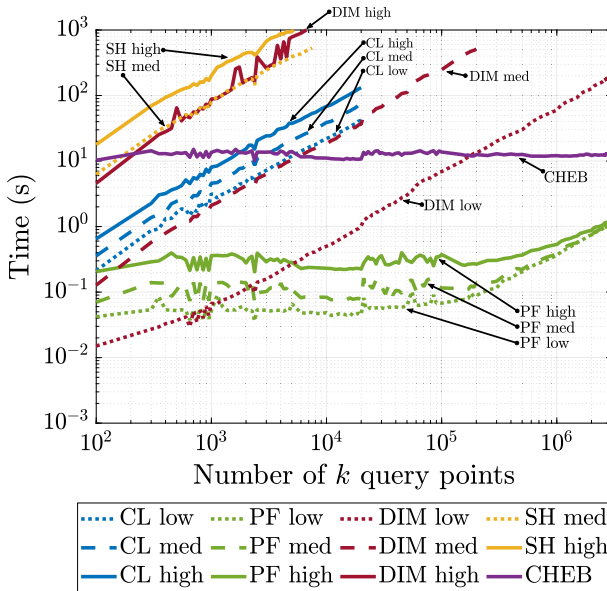


Fig. 11 Performance plot for reduced problem: log–log plot of computational time against number of query points. The algorithms were tested at various relative normwise accuracies: $\varepsilon \approx 10^{-4}$ (‘low’ accuracy), $\varepsilon \approx 10^{-7}$ (‘medium’ accuracy), and $\varepsilon \approx 10^{-10}$ (‘high’ accuracy). Note that the DIM, shooting (SH), and collocation algorithms are clearly linear in complexity with respect to N_q . The high-order interpolant using Chebfun with collocation sampling (CHEB) is almost constant time but has a large initial computational cost. The path-following algorithm is also almost constant time but has a much lower initial computational cost due to swapping quadratic eigenproblem solves for linear solves

be kept at the lowest value possible that maintains required accuracy. We found N_z between 48 and 64 is optimal for the cases we tested. Furthermore, using N_z too high risks roundoff error causing deleterious effects, cf. Fig. 8a.

The Dormand–Prince integrator requires a tolerance for the adaptive step-size control. We suggest that 10^{-11} is the smallest value to use when the initial eigenproblem solve is performed in double precision. If the initial eigenvalue solve can be performed more accurately, for example in high-precision arithmetic, then the tolerance can be set around 10^{-15} . In any case, if using a smaller N_z then the tolerance should be adjusted to match the accuracy from the collocation solution.

9 Conclusions

By considering the boundary value eigenproblem posed by the Rayleigh stability equation with linearized free-surface boundary condition as parameterized by wave number k , we can adapt the path-following scheme in [29] to efficiently calculate the dispersion relation at high accuracy. This efficiency is achieved by first exchanging many expensive QZ decompositions on a size $2N$ matrix for one QZ decomposition and some linear solves on a size N matrix; second, we ‘front load’ the computational

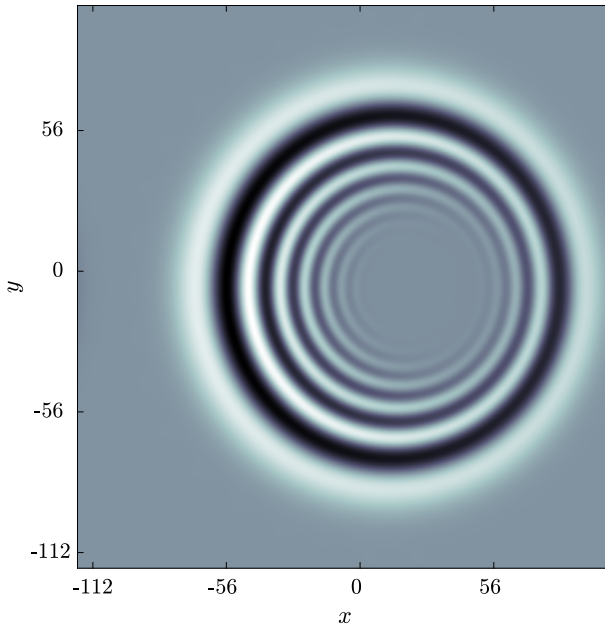


Fig. 12 Example initial value problem: propagation of surface waves from initial Gaussian disturbance atop fluid with velocity profile $U(z) = (U_T(z), 0)$. Note that the background vorticity causes visible anisotropy in the ring waves

cost into the numerical integration with light-weight Hermite interpolation being used to compute the sought solution points.

The accuracy tests in Sect. 5 suggest that the path-following algorithm can maintain the accuracy of the initial eigenpair v_0 and appears to be numerically more stable than the QZ decomposition used to obtain the initial eigenpair.

The algorithm is extended to permit calculation in the two-dimensional k -plane with scattered data. A method for processing through critical layers is given.

In other tests, not included here, it is clear the same approach works well for other problems from physics and engineering, assuming the problem is parametrized by a real scalar. Additional difficulties arise when there are exceptional points or bifurcations in the solution curve, or if the ODEs become stiff. These challenges may form the basis of future work.

The MATLAB library used to perform the calculations used in this paper is maintained at [31].

9.1 Example Usage

To demonstrate a possible use of the path-following algorithm, we calculate the result of a simple initial value problem of the form in [56, sec. 11.2] using the shear profile $U(z) = (U_T(z), 0)$ with $F^2 = 0.15$. Upon a sheared current, the anisotropic dispersion

relation results in a ring wave pattern which is visibly asymmetric, predicted for a linear current [19] and recently observed experimentally for the first time [46].

Following from Eq. (3), the general solution can be written as

$$\tilde{\zeta} = \frac{1}{(2\pi)^2} \int_{\mathbb{R}^2} \zeta_+ e^{i(\mathbf{k}\cdot\mathbf{x} - \omega_+(\mathbf{k})t)} d\mathbf{k} + \frac{1}{(2\pi)^2} \int_{\mathbb{R}^2} \zeta_- e^{i(\mathbf{k}\cdot\mathbf{x} - \omega_-(\mathbf{k})t)} d\mathbf{k}. \quad (42)$$

This requires two initial conditions, which are imposed as $\tilde{\zeta}(\mathbf{x}, t)|_{t=0} = \tilde{\zeta}_0(\mathbf{x})$ and $\dot{\tilde{\zeta}}(\mathbf{x}, t)|_{t=0} = \dot{\tilde{\zeta}}_0(\mathbf{x})$ and their Fourier transformed counterparts $\zeta_0(\mathbf{k})$ and $\dot{\zeta}_0(\mathbf{k})$.

Solving for ζ_+ and ζ_- :

$$\zeta_+(\mathbf{k}) = \frac{\omega_-(\mathbf{k})\zeta_0(\mathbf{k}) - i\dot{\zeta}_0(\mathbf{k})}{\omega_-(\mathbf{k}) - \omega_+(\mathbf{k})}, \quad \zeta_-(\mathbf{k}) = \frac{\omega_+(\mathbf{k})\zeta_0(\mathbf{k}) - i\dot{\zeta}_0(\mathbf{k})}{\omega_+(\mathbf{k}) - \omega_-(\mathbf{k})}. \quad (43)$$

In our example, we impose an initial Gaussian of the form $\tilde{\zeta}_0(\mathbf{x}) = e^{-\frac{x^2}{2\sigma^2} - \frac{y^2}{2\sigma^2}}$ and $\dot{\tilde{\zeta}}_0(\mathbf{x}) = 0$. Specifically, we choose $\sigma^2 \approx 5/2$ and plot for $t = 16$; this is shown in Fig. 12. The dispersion relation, $\omega_{\pm}(\mathbf{k})$ is calculated using the path-following algorithm, which allows us to pick a large number of grid points in \mathbf{x} coordinates and still compute the result quickly.

Acknowledgements Open Access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital).

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Advanpix LLC.: Advanpix: Multiprecision computing toolbox for MATLAB. <https://www.advanpix.com/>
2. Aiton, K., Driscoll, T.: An adaptive partition of unity method for Chebyshev polynomial interpolation. *SIAM J. Sci. Comp.* **40**(1), A251–A265 (2018). <https://doi.org/10.1137/17M112052X>
3. Allgower, E., Georg, K.: Introduction to numerical continuation methods. *Soc. Ind. Appl. Math.* (2003). <https://doi.org/10.1137/1.9780898719154>
4. Andrew, A., Chu, K., Lancaster, P.: Derivatives of eigenvalues and eigenvectors of matrix functions. *SIAM J. Matrix Anal. Appl.* **14**(4), 903–926 (1993). <https://doi.org/10.1137/0614061>
5. Aurentz, J.L., Trefethen, L.N.: Chopping a Chebyshev series. *ACM Trans. Math. Softw.* **43**(4), 33:1–33:21 (2017). <https://doi.org/10.1145/2998442>

6. Baltensperger, R., Trummer, M.R.: Spectral differencing with a twist. *SIAM J. Sci. Comput.* **24**(5), 1465–1487 (2003). <https://doi.org/10.1137/S1064827501388182>
7. Berrut, J., Trefethen, L.N.: Barycentric Lagrange interpolation. *SIAM Rev.* **46**(3), 501–517 (2004). <https://doi.org/10.1137/S0036144502417715>
8. Beyn, W.J., Thümmler, V.: Continuation of invariant subspaces for parameterized quadratic eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **31**(3), 1361–1381 (2010). <https://doi.org/10.1137/080723107>
9. Beyn, W.J., Kleß, W., Thümmler, V.: Continuation of low-dimensional invariant subspaces in dynamical systems of large dimension. In: Fiedler, B. (ed.) *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*, pp. 47–72. Springer, Berlin (2001)
10. Beyn, W.J., Effenberger, C., Kressner, D.: Continuation of eigenvalues and invariant pairs for parameterized nonlinear eigenvalue problems. *Numer. Math.* **119**(3), 489 (2011). <https://doi.org/10.1007/s00211-011-0392-1>
11. Boyd, J.: *Chebyshev and Fourier Spectral Methods: Second Edition*. Dover Books on Mathematics, Revised edn. Dover Publications, New York (2001)
12. Craik, A.D.D.: *Wave Interactions and Fluid Flows*. Cambridge Monographs on Mechanics. Cambridge University Press (1986). <https://doi.org/10.1017/CBO9780511569548>
13. Dalrymple, R.: Water waves on a bilinear shear current. In: *Proc. 14th Intl. Conf. Coastal Engn.*, pp. 24–28. Am. Soc. Civil Engineers (1974)
14. Dong, Z., Kirby, J.: Theoretical and numerical study of wave-current interaction in strongly-sheared flows. *Coast. Eng. J.* **1**(33), 2 (2012). <https://doi.org/10.9753/icce.v33.waves.2>
15. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. *J. Comput. Appl. Math.* **6**(1), 19–26 (1980). [https://doi.org/10.1016/0771-050X\(80\)90013-3](https://doi.org/10.1016/0771-050X(80)90013-3)
16. Drazin, P.G., Reid, W.H.: *Parallel Shear Flows*, 2 edn. Cambridge Mathematical Library. Cambridge University Press, pp. 124–250 (2004). <https://doi.org/10.1017/CBO9780511616938.006>
17. Driscoll, T.A., Hale, N., Trefethen, L.N.: *Chebfun guide* (2014). <http://www.chebfun.org/docs/guide/>. (Also see *Chebfun* GitHub repository at <https://github.com/chebfun/chebfun>)
18. Ehrnström, M., Villari, G.: Linear water waves with vorticity: rotational features and particle paths. *J. Differ. Equ.* **244**(8), 1888–1909 (2008). <https://doi.org/10.1016/j.jde.2008.01.012>
19. Ellingsen, S.Å.: Initial surface disturbance on a shear current: the Cauchy-Poisson problem with a twist. *Phys. Fluids* **26**(8), 082104 (2014). <https://doi.org/10.1063/1.4891640>
20. Ellingsen, S.Å.: Ship waves in the presence of uniform vorticity. *J. Fluid Mech.* (2014). <https://doi.org/10.1017/jfm.2014.28>
21. Ellingsen, S.Å., Li, Y.: Approximate dispersion relations for waves on arbitrary shear flows. *J. Geophys. Res.* **122**(12), 9889–9905 (2017). <https://doi.org/10.1002/2017JC012994>
22. Hairer, E., Nørsett, S.P., Wanner, G.: *Solving Ordinary Differential Equations, Nonstiff Problems I*, 2nd edn. Springer, Berlin (1993)
23. Higham, D., Higham, N.: Backward error and condition of structured linear systems. *SIAM J. Matrix Anal. Appl.* **13**(1), 162–175 (1992). <https://doi.org/10.1137/0613014>
24. Kato, T.: Perturbation theory for linear operators. In: *Classics in Mathematics*. In: Springer, Berlin (1995). <https://doi.org/10.1007/978-3-642-66282-9>
25. Keller, H.B.: *Numerical methods in bifurcation problems*. Lectures on Mathematics and Physics, Tata Institute of Fundamental Research (Bombay), 1987 (1987). <https://ci.nii.ac.jp/naid/10026075958/en/>
26. Kirby, J.T., Chen, T.M.: Surface waves on vertically sheared flows: approximate dispersion relations. *J. Geophys. Res. Oceans* **94**(C1), 1013–1027 (1989). <https://doi.org/10.1029/JC094iC01p01013>
27. Li, Y., Ellingsen, S.Å.: A framework for modelling linear surface waves on shear currents in slowly varying waters. *J. Geophys. Res. Oceans* (2019). <https://doi.org/10.1029/2018JC014390>
28. Liao, S.: *Beyond Perturbation: Introduction to the Homotopy Analysis Method*. Modern Mechanics and Mathematics. Taylor & Francis, Oxford (2003)
29. Loisel, S., Maxwell, P.: Path-following method to determine the field of values of a matrix with high accuracy. *SIAM J. Matrix Anal. Appl.* **39**(4), 1726–1749 (2018). <https://doi.org/10.1137/17M1148608>
30. Lui, S., Keller, H., Kwok, T.: Homotopy method for the large, sparse, real nonsymmetric eigenvalue problem. *SIAM J. Matrix Anal. Appl.* **18**(2), 312–333 (1997). <https://doi.org/10.1137/S0895479894273900>
31. Maxwell, P.: *Nessie Water Waves Library*. <https://github.com/PeterMaxwell/nessie-water-waves>
32. Maxwell, P., Smeltzer, B.K., Ellingsen, S.Å.: The error in predicted phase velocity of surface waves atop a shear current with uncertainty. *Water Waves* (2019). <https://doi.org/10.1007/s42286-019-00012-x>

33. Mei, C., Stiassnie, M., Yue, D.: *Theory and Applications of Ocean Surface Waves: Linear aspects*. Advanced series on ocean engineering. World Scientific, Pennsylvania (2005)
34. Miles, J.: Gravity waves on shear flows. *J. Fluid Mech.* **443**, 293–299 (2001). <https://doi.org/10.1017/S0022112001005043>
35. Miles, J.: A note on surface waves generated by shear-flow instability. *J. Fluid Mech.* **447**, 173–177 (2001). <https://doi.org/10.1017/S0022112001005833>
36. Peregrine, D.H.: Interaction of water waves and currents. *Adv. Appl. Mech.* **16**, 9–117 (1976)
37. Qian, J., Chu, D., Tan, R.: Analyticity of semisimple eigenvalues and corresponding eigenvectors of matrix-valued functions. *SIAM J. Matrix Anal. Appl.* **36**(4), 1542–1566 (2015). <https://doi.org/10.1137/151003799>
38. Rayleigh, Lord: On the stability, or instability, of certain fluid motions. *Proc. Lond. Math. Soc.* **s1-11**(1), 57–72 (1879). <https://doi.org/10.1112/plms/s1-11.1.57>
39. Rayleigh, Lord: On the stability or instability of certain fluid motions, II. *Proc. Lond. Math. Soc.* **s1-19**(1), 67–75 (1887). <https://doi.org/10.1112/plms/s1-19.1.67>
40. Rayleigh, Lord: On the stability or instability of certain fluid motions III. *Proc. Lond. Math. Soc.* **s1-27**(1), 5–12 (1895). <https://doi.org/10.1112/plms/s1-27.1.5>
41. Schmid, P.J., Henningson, D.S.: *Stability and Transition in Shear Flows*. Applied Mathematical Sciences, vol. 142. Springer, New York (2001). <https://doi.org/10.1007/978-1-4613-0185-1>
42. Shampine, L.F.: Some practical Runge–Kutta formulas. *Math. Comput.* **46**, 135–150 (1986). <https://doi.org/10.2307/2008219>
43. Shrira, V.I.: Surface waves on shear currents: solution of the boundary-value problem. *J. Fluid Mech.* **252**, 565–584 (1993). <https://doi.org/10.1017/S002211209300388X>
44. Skop, R.A.: Approximate dispersion relation for wave-current interactions. *J. Waterw. Port Coast* **113**(2), 187–195 (1987). [https://doi.org/10.1061/\(ASCE\)0733-950X\(1987\)113:2\(187\)](https://doi.org/10.1061/(ASCE)0733-950X(1987)113:2(187))
45. Smeltzer, B.K., Ellingsen, S.Å.: Surface waves on currents with arbitrary vertical shear. *Phys. Fluids* **29**(4), 047102 (2017). <https://doi.org/10.1063/1.4979254>
46. Smeltzer, B.K., Æsøy, E., Ellingsen, S.Å.: Observation of surface wave patterns modified by sub-surface shear currents. *J. Fluid Mech.* **873**, 508–530 (2019). <https://doi.org/10.1017/jfm.2019.424>
47. Stewart, R.H., Joy, J.W.: HF radio measurements of surface currents. *Deep-Sea Res. Oceanogr. Abstr.* **21**(12), 1039–1049 (1974). [https://doi.org/10.1016/0011-7471\(74\)90066-7](https://doi.org/10.1016/0011-7471(74)90066-7)
48. Stoker, J.J.: *Water Waves: The Mathematical Theory with Applications*. Wiley Classics Library Edition. Wiley. (1992 [1958]). <https://www.wiley.com/en-us/Water+Waves+%3A+The+Mathematical+Theory+with+Applications-p-9780471570349>
49. Tao, T.: *Topics in Random Matrix Theory*, *Graduate Studies in Mathematics*, vol. 132. American Mathematical Society (2012). <https://terrytao.files.wordpress.com/2011/02/matrix-book.pdf>
50. Taylor, G.: The action of a surface current used as a breakwater. *Proc. R. Soc. Lond. A* **231**, 466–478 (1955)
51. Tisseur, F.: Backward error and condition of polynomial eigenvalue problems. *Linear Algebra Appl.* **309**(1), 339–361 (2000). [https://doi.org/10.1016/S0024-3795\(99\)00063-4](https://doi.org/10.1016/S0024-3795(99)00063-4)
52. Trefethen, L.N.: *Spectral methods in MATLAB*. Software, environments, and tools. Soc. Ind. Appl. Math. (2000). <https://doi.org/10.1137/1.9780898719598>
53. Uhlig, F.: Zhang neural networks for fast and accurate computations of the field of values. *Linear and Multilinear Algebra* pp. 1–17 (2019). <https://doi.org/10.1080/03081087.2019.1648375>
54. Uhlig, F., Zhang, Y.: Time-varying matrix eigenanalyses via Zhang neural networks and look-ahead finite difference equations. *Linear Algebra Appl.* **580**, 417–435 (2019). <https://doi.org/10.1016/j.laa.2019.06.028>
55. Weideman, J.A., Reddy, S.C.: A MATLAB Differentiation Matrix Suite. *ACM Trans. Math. Softw.* **26**(4), 465–519 (2000). <https://doi.org/10.1145/365723.365727>
56. Whitham, G.B.: *Linear and nonlinear waves*. Wiley, Oxford. (1999 [1974]). <https://doi.org/10.1002/9781118032954>
57. Zhang, X.: Short surface waves on surface shear. *J. Fluid Mech.* **541**, 345–370 (2005). <https://doi.org/10.1017/S0022112005006063>