



# **ALGORITHMIC SOLUTION OF HIGH ORDER PARTIAL DIFFERENTIAL EQUATIONS IN JULIA VIA THE FOKAS TRANSFORM METHOD**

**XIAO LINFAN**

**Capstone Final Report for BSc (Honours) in  
Mathematical, Computational, and Statistical Sciences**

**Supervised by: Prof. Dave Smith**

**AY 2018/2019**

## **Yale-NUS College Capstone Project**

### **DECLARATION & CONSENT**

1. I declare that the product of this Project, the Thesis, is the end result of my own work and that due acknowledgement has been given in the bibliography and references to ALL sources be they printed, electronic, or personal, in accordance with the academic regulations of Yale-NUS College.
2. I acknowledge that the Thesis is subject to the policies relating to Yale-NUS College Intellectual Property ([Yale-NUS HR 039](#)).

### **ACCESS LEVEL**

3. I agree, in consultation with my supervisor(s), that the Thesis be given the access level specified below: [check one only]

Unrestricted access

Make the Thesis immediately available for worldwide access.

Access restricted to Yale-NUS College for a limited period

Make the Thesis immediately available for Yale-NUS College access only from \_\_\_\_\_  
(mm/yyyy) to \_\_\_\_\_ (mm/yyyy), up to a maximum of 2 years for the following reason(s): (please specify; attach a separate sheet if necessary):  
\_\_\_\_\_

After this period, the Thesis will be made available for worldwide access.

Other restrictions: (please specify if any part of your thesis should be restricted)  
\_\_\_\_\_  
\_\_\_\_\_

Linfan Xiao Elm College  
Name & Residential College of Student

Linfan Xiao  
Signature of Student

David A Smith

R. S.  
Name & Signature of Supervisor

2019/04/15  
Date

2019/04/15  
Date

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation	1
1.2	The initial-boundary value problem	5
<b>2</b>	<b>Preliminaries</b>	<b>7</b>
2.1	Homogeneous boundary value problems	7
2.2	Adjoints of homogeneous boundary value problems	8
2.2.1	Adjoint differential operator	8
2.2.2	Adjoint boundary conditions	9
	Green's formula	9
	Boundary-form formula	15
	Characterization of adjoint boundary conditions	23
<b>3</b>	<b>Algorithm</b>	<b>29</b>
3.1	Constructing the adjoint of a set of homogeneous boundary conditions	30
3.1.1	Checking input	31
3.1.2	Finding a candidate adjoint	31
3.1.3	Checking the validity of the candidate adjoint	32
3.2	Constructing the Fokas transform pair	33
	Definition of the integrands	33
	Definition of the integration contours	34
	Definition of the Fokas transform pair	35
3.2.1	Constructing the integration contours	36
	Tracing contour backbone	36
	Deforming contour backbone to avoid poles of the integrands	37
	Approximating poles of the integrands	45
3.2.2	Constructing the integrands	45

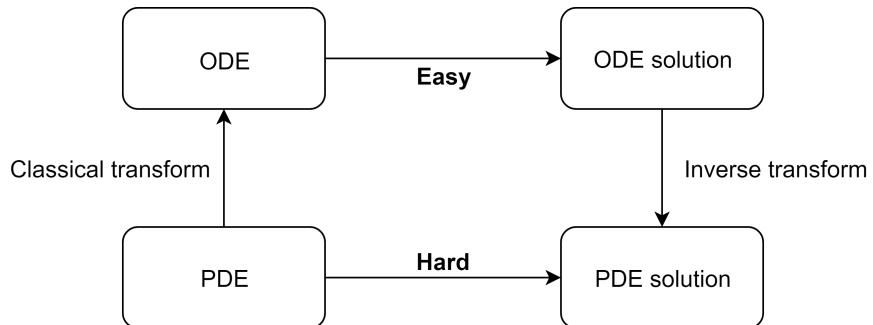
Approximating the inner integral using Chebyshev polynomials . . . . .	47
<b>4 Implementation . . . . .</b>	<b>52</b>
4.1 Overview . . . . .	52
4.2 Sample usage . . . . .	53
4.2.1 Solving IBVP . . . . .	53
4.2.2 Symbolic computation . . . . .	57
<b>5 Discussion . . . . .</b>	<b>59</b>
5.1 Extension to other problems . . . . .	59
5.2 Possible improvements . . . . .	59
5.2.1 Symbolic computation . . . . .	59
5.2.2 Automatically approximating poles of the integrands . . . . .	60
5.2.3 Computation speed . . . . .	61
<b>References . . . . .</b>	

# 1 Introduction

Evolution partial differential equations (PDEs) relate a quantity to its rates of change with respect to both time and position. Evolution PDEs can model a variety of phenomena, such as heat flow, wave propagation, and particle motion. This project concerns implementing an algorithmic procedure to solve a class of initial-boundary value problems (IBVPs) for linear evolution PDEs of arbitrary spatial order in the finite interval [1] based on the Fokas transform method [2][3][4][5][6][7]. The implementation is developed in Julia [8], featuring symbolic computation along with numeric evaluation.

## 1.1 Motivation

IBVPs arise naturally in the study of physical systems. Certain IBVPs (henceforth referred to as type I problems) can be solved algorithmically via classical transform pairs such as the Fourier transform (Figure 1).



**Figure 1:** Solving type I IBVPs using classical transform pairs.

For example, the heat equation [9]

$$u_t = Ku_{xx}, \quad (x, t) \in (0, L) \times (0, \infty)$$

with periodic boundary conditions

$$u(0, t) = u(L, t), \quad t \in (0, \infty)$$

$$u_x(0, t) = u_x(L, t), \quad t \in (0, \infty)$$

and initial datum

$$u(x, 0) = f(x), \quad x \in (0, L)$$

can be turned into an ordinary differential equation (ODE) in the temporal variable  $t$

$$U_t = -K\lambda^2 U$$

with initial datum

$$U(\lambda, 0) = F(\lambda), \quad \lambda \in (0, L).$$

where  $U(\lambda, t) := \mathcal{F}[u]$  and  $F(\lambda) := \mathcal{F}[f]$  are the Fourier transforms of  $u(x, t)$  and  $f(x)$  with respect to  $x$ , respectively. This initial-value problem (IVP) for the ODE can be solved directly as

$$U(\lambda, t) = F(\lambda)e^{-\lambda^2 Kt},$$

which, when applying the inverse Fourier transform, gives the IBVP solution  $u(x, t)$ .

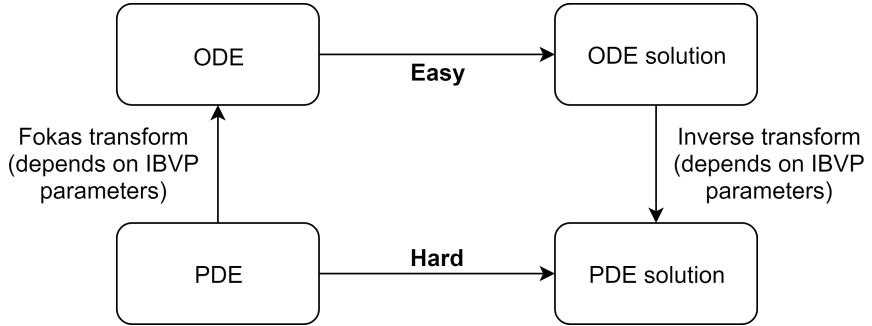
For more complicated IBVPs (henceforth referred to as type II problems), no such classical transform pairs exist. The linearized Korteweg-de Vries (linearized KdV) equation which describes shallow water waves [10],

$$u_t + u_{xxx} = 0, \tag{1.1}$$

is one such example. Solving type II problems typically requires a combination of ad-hoc methods, which are often specific to the given problem and cannot be generalized to related problems with different parameters (e.g., IBVPs involving PDEs of different orders or different boundary conditions).

The Fokas transform method extends the idea of classical transform pairs to solving type II problems by constructing non-classical transform pairs (henceforth referred to as the Fokas transform pairs) based on the IBVP parameters. In other words, the Fokas method “customizes” appropriate transform pairs for a class of

related IBVPs with different parameters; this class of IBVPs can then be solved algorithmically in a manner similar to Figure 1, as illustrated in Figure 2.



**Figure 2:** Solving type II IBVPs using the Fokas transform pairs.

A thorough presentation of the Fokas method can be found in [4], and a pedagogical introduction in [6]. Yet the history of the Fokas method can be traced back to a general approach to solving IBVPs for linear and integrable nonlinear PDEs, introduced in 1997 [11]. Research efforts thus inspired motivated the discovery of a new transform method to solve boundary-value problems (BVPs) for linear evolution PDEs in two dimensions with arbitrary spatial order (i.e., arbitrary order in the derivative of the spatial variable) in 2000 [2]. The Fokas method for two-point BVPs for linear evolution PDEs was proposed in [3]. The discussion was expanded to a complete analysis of third and higher-order two-point BVPs in [5]. Ongoing research is extending the Fokas method to various settings, such as PDEs of fractional order [12].

This project focuses on the Fokas method for IBVPs [1]. Examples of IBVPs that can be solved by the Fokas method include the following IBVPs for the linearized KdV equation [1]:

$$q_t(x, t) + q_{xxx}(x, t) = 0, \quad (x, t) \in (0, 1) \times (0, T) \quad (1.2a)$$

$$q(x, 0) = f(x), \quad x \in [0, 1] \quad (1.2b)$$

$$q(0, t) = q(1, t) = 0, \quad t \in [0, T] \quad (1.2c)$$

$$q_x(1, t) = \frac{1}{2}q_x(0, t) \quad t \in [0, T], \quad (1.2d)$$

and

$$q_t(x, t) + q_{xxx}(x, t) = 0, \quad (x, t) \in (0, 1) \times (0, T) \quad (1.3a)$$

$$q(x, 0) = f(x), \quad x \in [0, 1] \quad (1.3b)$$

$$q(0, t) = q(1, t) = q_x(1, t) = 0, \quad t \in [0, T]. \quad (1.3c)$$

As mentioned earlier and will be shown later, beyond the two examples above, the Fokas method is in fact applicable to IBVPs with arbitrary spatial order. PDEs of higher order are generally not as well understood as their first and second order counterparts. The Fokas method thus advances the understanding of higher order PDEs by providing an algorithmic procedure to solve an entire class of IBVPs of arbitrary spatial order. Applying the Fokas method by hand, however, is laborious. It is thus of interest to implement the Fokas method as a software library that aims to supply as much computer aid as possible in the computation process. The types of computer aid include visualizations of integration contours, numeric evaluation, and more importantly, derivation of explicit symbolic formulas for important mathematical objects which would help proving technical lemmas that arise in applying the Fokas method to IBVPs [5] and beyond [13].

In this project, the implementation of the Fokas method is developed in Julia, a free open-source, high-performance language for numerical computing [8]. Among tools for numerical computing, commercial software packages may provide functionalities to execute sophisticated computational tasks; yet due to being closed-source, there exists a lack of transparency which limits users' understanding of the nature of the software output. As a result, there has been a trend in the mathematical community that encourages the use of open-source languages like Julia, which allows users to verify the correctness of the output. Therefore, we choose Julia to be the language of implementation. At the start of this project, the latest release of Julia is 1.0; however, existing graphical libraries in Julia were

not yet made compatible with Julia 1.0 at that time. Thus, this project uses the next latest release of Julia that is stable at that time, namely Julia 0.6.4.

To the best of our knowledge, despite the presence of various differential equation solvers in mathematical software such as Julia, Mathematica, and Matlab, this is the first time that the Fokas method is implemented computationally in any generality, and it allows solving a particular class of IBVPs for which no other general analytic solver algorithm yet exists.

## 1.2 The initial-boundary value problem

To formally characterize the class of IBVPs that can be solved by the Fokas method [1, p.9], we first introduce the following definitions.

Define linearly independent boundary forms  $B_j : C^\infty[0, 1] \rightarrow \mathbb{C}$  (where  $C^\infty[0, 1]$  denotes the class of real-valued functions differentiable for all orders on  $[0, 1]$ ) by

$$B_j \phi := \sum_{k=0}^{n-1} (b_{jk} \phi^{(k)}(0) + \beta_{jk} \phi^{(k)}(1)), \quad j \in \{1, 2, \dots, n\} \quad (1.4)$$

where the boundary coefficients  $b_{jk}, \beta_{jk} \in \mathbb{R}$ .

Define the set of smooth functions  $\phi$  that satisfy the homogeneous boundary conditions  $B_j \phi = 0$  for  $j \in \{1, 2, \dots, n\}$  as

$$\Phi := \{\phi \in C^\infty[0, 1] : B_j \phi = 0 \forall j \in \{1, 2, \dots, n\}\}. \quad (1.5)$$

Define a spatial differential operator (i.e., a differential operator in the spatial variable  $x$ ) of order  $n$  to be

$$S := (-i)^n \frac{d^n}{dx^n}. \quad (1.6)$$

The Fokas method allows solving any well-posed IBVP that can be written in the form

$$(\partial_t + aS)q(x, t) = 0 \quad \forall (x, t) \in (0, 1) \times (0, T) \quad (1.7a)$$

$$q(x, 0) = f(x) \in \Phi \quad \forall x \in [0, 1] \quad (1.7b)$$

$$q(\cdot, t) \in \Phi \quad \forall t \in [0, T], \quad (1.7c)$$

where  $a$  is a complex constant. For the IBVP to be well-posed, we require at least that  $a = \pm i$  if  $n$  is odd and  $\operatorname{Re}(a) \geq 0$  if  $n$  is even. Equation (1.7a) is a PDE in the spatial variable  $x$  and the temporal variable  $t$  with arbitrary spatial order  $n$ . Equation (1.7b) is an initial datum where the temporal variable  $t = 0$ . Equation (1.7c) corresponds to the homogeneous spatial boundary conditions, specifying that for any fixed  $t \in [0, T]$ ,  $q(x, t) =: \phi(x)$  as a function of  $x$  satisfies  $B_j \phi = 0$  for  $j \in \{1, 2, \dots, n\}$ .

The linearized KdV equations introduced in equation (1.1) are examples of such IBVPs. Specifically, IBVP (1.2) can be written in the form of (1.7) with  $a = -i$ ,

$$S = (-i)^3 \frac{d^3}{dx^3} = i \frac{d^3}{dx^3},$$

$$B_1 \phi = 1 \cdot \phi(0) + 0 \cdot \phi(1) + 0 \cdot \phi^{(1)}(0) + 0 \cdot \phi^{(1)}(1) + 0 \cdot \phi^{(2)}(0) + 0 \cdot \phi^{(2)}(1)$$

$$B_2 \phi = 0 \cdot \phi(0) + 1 \cdot \phi(1) + 0 \cdot \phi^{(1)}(0) + 0 \cdot \phi^{(1)}(1) + 0 \cdot \phi^{(2)}(0) + 0 \cdot \phi^{(2)}(1)$$

$$B_3 \phi = 0 \cdot \phi(0) + 0 \cdot \phi(1) + 1 \cdot \phi^{(1)}(0) - 2 \cdot \phi^{(1)}(1) + 0 \cdot \phi^{(2)}(0) + 0 \cdot \phi^{(2)}(1),$$

and

$$\Phi = \{\phi \in C^\infty[0, 1] : B_j \phi = 0 \forall j \in \{1, 2, 3\}\}.$$

Similarly, IBVP (1.3) can be written in the form of (1.7) with  $a = -i$ ,

$$S = (-i)^3 \frac{d^3}{dx^3} = i \frac{d^3}{dx^3},$$

$$B_1 \phi = 1 \cdot \phi(0) + 0 \cdot \phi(1) + 0 \cdot \phi^{(1)}(0) + 0 \cdot \phi^{(1)}(1) + 0 \cdot \phi^{(2)}(0) + 0 \cdot \phi^{(2)}(1)$$

$$B_2 \phi = 0 \cdot \phi(0) + 1 \cdot \phi(1) + 0 \cdot \phi^{(1)}(0) + 0 \cdot \phi^{(1)}(1) + 0 \cdot \phi^{(2)}(0) + 0 \cdot \phi^{(2)}(1)$$

$$B_3\phi = 0 \cdot \phi(0) + 0 \cdot \phi(1) + 0 \cdot \phi^{(1)}(0) + 1 \cdot \phi^{(1)}(1) + 0 \cdot \phi^{(2)}(0) + 0 \cdot \phi^{(2)}(1),$$

and

$$\Phi = \{\phi \in C^\infty[0, 1] : B_j\phi = 0 \forall j \in \{1, 2, 3\}\}.$$

For appropriate transform  $F_\lambda$  and inverse transform  $f_x$  pair found using the Fokas method (to be explained in detail later), the solution to the IVP characterized above is given by [1, p.15]

$$q(x, t) = f_x(e^{-a\lambda^n t} F_\lambda(f)), \quad (1.8)$$

thus completing the procedure summarised in Figure 2.

## 2 Preliminaries

We now begin to introduce the preliminary definitions and results used in the algorithmic procedure that finds (1.8). The key to the algorithm is the construction of the Fokas transform pair, which in turn depends on finding a valid adjoint of the given boundary conditions. Thus, we devote the following sections to characterizing the adjoint, and by doing so, outline its construction. This section presents material from chapter 11 of [14], with proofs expanded and completed, and the utility of the theorems expounded.

### 2.1 Homogeneous boundary value problems

**Definition 2.1.** [14, p.81] A linear differential operator  $L$  of order  $n$  ( $n > 1$ ) on interval  $[a, b]$  is defined by

$$Lx = p_0 x^{(n)} + p_1 x^{(n-1)} + \cdots + p_{n-1} x' + p_n x, \quad (2.1)$$

where the  $p_k$  are complex-valued functions of class  $C^{n-k}$  on  $[a, b]$  and  $p_0(t) \neq 0$  on  $[a, b]$ .

**Definition 2.2.** [14, p.284] **Homogeneous boundary conditions** refer to a set of equations of the type

$$\sum_{k=1}^n (M_{jk}x^{(k-1)}(a) + N_{jk}x^{(k-1)}(b)) = 0 \quad (j = 1, \dots, m) \quad (2.2)$$

where  $M_{jk}, N_{jk}$  are complex constants.

**Definition 2.3.** [14, p.284] A **homogeneous boundary value problem** concerns finding the solutions of

$$Lx = 0$$

on some interval  $[a, b]$  which satisfy some homogeneous boundary conditions.

## 2.2 Adjoint of homogeneous boundary value problems

Recall from linear algebra that, given a linear map  $T$  from inner product spaces  $V$  to  $W$ , the adjoint of  $T$  is the function  $T^* : W \rightarrow V$  with

$$\langle Tv, w \rangle = \langle v, T^*w \rangle \quad (2.3)$$

for  $v \in V, w \in W$ , where  $\langle \cdot, \cdot \rangle$  denotes the inner product defined on  $V$  and  $W$  [15, p.204]. There exists a similar notion for boundary value problems, which we begin to formulate below.

### 2.2.1 Adjoint differential operator

**Definition 2.4.** [14, p.84] Given a linear differential operator  $L$  of order  $n$  as in Definition 2.1, the operator  $L^+$  given by

$$L^+x = (-1)^n(\bar{p}_0x)^{(n)} + (-1)^{n-1}(\bar{p}_1x)^{(n-1)} + \cdots + \bar{p}_nx$$

(where  $\bar{p}_k$  denotes the complex conjugate of  $p_k$  for  $k \in \{0, \dots, n\}$ ) is the **adjoint** of  $L$ .

### 2.2.2 Adjoint boundary conditions

For a homogeneous boundary value problem  $\pi$  with linear differential operator  $L$  and some homogeneous boundary conditions, an adjoint problem  $\pi^+$  involves the adjoint linear differential operator  $L^+$  and a set of homogeneous adjoint boundary conditions. The adjoint boundary conditions are such that an equation similar to (2.3) holds for solutions of  $\pi$  and those of  $\pi^+$ , with the inner product  $\langle \cdot, \cdot \rangle$  defined as  $\langle u, v \rangle := \int_a^b u \bar{v} dt$  for  $u, v \in C^n$  on  $[a, b]$ . In the following sections, we seek to characterize the adjoint boundary conditions and their construction.

The construction depends on two important results, namely Green's formula and the boundary-form formula. Green's formula allows characterizing a form, which, when used in the boundary-form formula, gives rise to the desired construction.

We begin with Green's formula.

#### Green's formula

**Theorem 2.5.** [14, p.284] (Green's formula) For  $u, v \in C^n$  on  $[a, b]$ ,

$$\int_{t_1}^{t_2} (Lu)\bar{v} dt - \int_{t_1}^{t_2} u(\overline{L^+v}) dt = [uv](t_2) - [uv](t_1) \quad (2.4)$$

where  $a \leq t_1 < t_2 \leq b$  and  $[uv](t)$  is the form in  $(u, u', \dots, u^{(n-1)})$  and  $(v, v', \dots, v^{(n-1)})$  given by

$$[uv](t) = \sum_{m=1}^n \sum_{j+k=m-1} (-1)^j u^{(k)}(t) (p_{n-m} \bar{v})^{(j)}(t) \quad (2.5)$$

Using the form  $[uv](t)$ , we define an important  $n \times n$  matrix  $B$  whose entries  $B_{jk}$  satisfy

$$[uv](t) = \sum_{j,k=1}^n B_{jk}(t) u^{(k-1)}(t) \bar{v}^{(j-1)}(t). \quad (2.6)$$

Note that

$$\begin{aligned}
[uv](t) &= \sum_{m=1}^n \sum_{j+k=m-1} (-1)^j u^{(k)}(t) (p_{n-m} \bar{v})^{(j)}(t) \\
&= \sum_{m=1}^n \sum_{j+k=m-1} (-1)^j u^{(k)}(t) \left( \sum_{\ell=0}^j \binom{j}{\ell} p_{n-m}^{(j-\ell)}(t) \bar{v}^{(\ell)}(t) \right) \\
&= \sum_{m=1}^n \sum_{k=0}^{m-1} (-1)^{m-1-k} u^{(k)}(t) \left( \sum_{\ell=0}^{m-1-k} \binom{m-1-k}{\ell} p_{n-m}^{(m-1-k-\ell)}(t) \bar{v}^{(\ell)}(t) \right) \\
&= \sum_{m=1}^n \sum_{k=1}^m (-1)^{m-k} \left( \sum_{\ell=0}^{m-k} \binom{m-k}{\ell} p_{n-m}^{(m-k-\ell)}(t) \bar{v}^{(\ell)}(t) \right) u^{(k-1)}(t) \quad (\text{shifting } k \text{ to } k+1) \\
&= \sum_{k=1}^n \sum_{m=k}^n (-1)^{m-k} \left( \sum_{\ell=0}^{m-k} \binom{m-k}{\ell} p_{n-m}^{(m-k-\ell)}(t) \bar{v}^{(\ell)}(t) \right) u^{(k-1)}(t).
\end{aligned}$$

To find  $B_{jk}$ , we need to extract the coefficients of  $u^{(k-1)} \bar{v}^{(j-1)}$ . We first note that, fixing  $m$  and  $k$ , when  $\ell = j-1$ , the coefficient of  $\bar{v}^{(j-1)}$  is

$$\binom{m-k}{j-1} p_{n-m}^{(m-k-j+1)}(t).$$

To find the coefficient of  $u^{(k-1)} \bar{v}^{(j-1)}$ , we need to fix  $k$  and collect the above coefficient across all values of  $m$ . Since  $m$  goes up to  $n$ ,  $m-k$  goes up to  $n-k$ . Since  $\ell \leq m-k$ ,  $\ell = j-1$  implies  $j-1 \leq m-k$ . Thus,  $m-k$  ranges from  $j-1$  to  $n-k$ . Let  $\ell' := m-k$ , then  $m = k+\ell'$ , and the above equation becomes

$$\begin{aligned}
[uv](t) &= \sum_{k=1}^n \sum_{j=1}^n (-1)^{\ell'} \left( \sum_{\ell'=j-1}^{n-k} \binom{\ell'}{j-1} p_{n-(k+\ell')}^{(\ell'-(j-1))}(t) \right) \bar{v}^{(j-1)}(t) u^{(k-1)}(t) \\
&= \sum_{j,k=1}^n \left( \sum_{\ell=j-1}^{n-k} \binom{\ell}{j-1} p_{n-k-\ell}^{(\ell-j+1)}(t) (-1)^\ell \right) u^{(k-1)}(t) \bar{v}^{(j-1)}(t) \quad (\text{replace } \ell' \text{ by } \ell).
\end{aligned}$$

Thus,

$$B_{jk}(t) = \sum_{\ell=j-1}^{n-k} \binom{\ell}{j-1} p_{n-k-\ell}^{(\ell-j+1)}(t) (-1)^\ell.$$

We note that for  $j+k > n+1$ , or  $j-1 > n-k$ ,  $\ell$  is undefined, for the terms  $u^{(k-1)}(t) \bar{v}^{(j-1)}(t)$  with  $j+k > n+1$  do not exist in  $[uv](t)$ . Thus,  $B_{jk}(t) = 0$ .

Also, for  $j + k = n + 1$ , or  $j - 1 = n - k$ ,

$$B_{jk}(t) = \binom{j-1}{j-1} p_{j-1-(j-1)}^{(j-1-j+1)}(t) (-1)^{j-1} = (-1)^{j-1} p_0(t).$$

Thus, the matrix  $B$  has the form

$$B(t) = \begin{bmatrix} B_{11} & B_{12} & \cdots & \cdots & B_{1n-1} & p_0(t) \\ B_{21} & B_{22} & \cdots & \cdots & -p_0(t) & 0 \\ \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ (-1)^{n-1} p_0(t) & 0 & \cdots & \cdots & 0 & 0 \end{bmatrix}. \quad (2.7)$$

We note that because  $p_0(t) \neq 0$  on  $[a, b]$  (as required in Definition 2.1),  $B(t)$  is square with  $\det B(t) = (p_0(t))^n \neq 0$  on  $[a, b]$ . Thus,  $B(t)$  is nonsingular for  $t \in [a, b]$ .

Now we seek another matrix  $\hat{B}$  that embodies both the characteristics of  $B$  and those of the interval  $[a, b]$ . This concerns writing the right-hand side of Green's formula in matrix form. We begin by introducing the following definitions.

**Definition 2.6.** [14, p.285] For vectors  $f = (f_1, \dots, f_k)$  and  $g = (g_1, \dots, g_k)$ , define the product

$$f \cdot g := \sum_{i=1}^k f_i \bar{g}_i.$$

Note that  $f \cdot g = g^* \cdot f$  where  $*$  denotes the conjugate transpose.

**Definition 2.7.** [14, p.285] A **semibilinear form** is a complex-valued function  $\mathcal{S}$  defined for pairs of vectors  $f = (f_1, \dots, f_k)$ ,  $g = (g_1, \dots, g_k)$  satisfying

$$\mathcal{S}(\alpha f + \beta g, h) = \alpha \mathcal{S}(f, h) + \beta \mathcal{S}(g, h)$$

$$\mathcal{S}(f, \alpha g + \beta h) = \bar{\alpha} \mathcal{S}(f, g) + \bar{\beta} \mathcal{S}(f, h)$$

for any complex numbers  $\alpha, \beta$  and vectors  $f, g, h$ .

We note that if

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1k} \\ \vdots & \ddots & \vdots \\ s_{k1} & \cdots & s_{kk} \end{bmatrix},$$

then  $Sf \cdot g$  is given by

$$\begin{aligned} \mathcal{S}(f, g) := Sf \cdot g &= \begin{bmatrix} s_{11} & \cdots & s_{1k} \\ \vdots & \ddots & \vdots \\ s_{k1} & \cdots & s_{kk} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_k \end{bmatrix} \cdot \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^k s_{1j} f_j \\ \vdots \\ \sum_{j=1}^k s_{kj} f_j \end{bmatrix} \cdot \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} \\ &= \sum_{i=1}^k \left( \sum_{j=1}^k s_{ij} f_j \right) \bar{g}_i = \sum_{i,j=1}^k s_{ij} f_j \bar{g}_i. \end{aligned} \quad (2.8)$$

To see that this is a semibilinear form:

$$\begin{aligned} \mathcal{S}(\alpha f + \beta g, h) &= \sum_{i,j=1}^k s_{ij} (\alpha f_j + \beta g_j) \bar{h}_i = \alpha \sum_{i,j=1}^k s_{ij} f_j \bar{h}_i + \beta \sum_{i,j=1}^k g_j \bar{h}_i \\ &= \alpha Sf \cdot h + \beta Sg \cdot h = \alpha \mathcal{S}(f, h) + \beta \mathcal{S}(g, h); \end{aligned}$$

and similarly,

$$\begin{aligned} \mathcal{S}(f, \alpha g + \beta h) &= \sum_{i,j=1}^k s_{ij} f_j (\overline{\alpha g_i + \beta h_i}) = \bar{\alpha} \sum_{i,j=1}^k s_{ij} f_j \bar{g}_i + \bar{\beta} \sum_{i,j=1}^k f_j \bar{h}_i \\ &= \bar{\alpha} Sf \cdot g + \bar{\beta} Sf \cdot h = \bar{\alpha} \mathcal{S}(f, g) + \bar{\beta} \mathcal{S}(f, h). \end{aligned}$$

Under a similar matrix framework, we see that  $[uv](t)$  is a semibilinear form with matrix  $B(t)$ : Let  $\vec{u} = (u, u', \dots, u^{(n-1)})$  and  $\vec{v} = (v, v', \dots, v^{(n-1)})$ . Then we

have

$$\begin{aligned}
[uv](t) &= \sum_{j,k=1}^n B_{jk}(t) u^{(k-1)}(t) \bar{v}^{(j-1)}(t) \quad (\text{by equation (2.6)}) \\
&= \sum_{i,j=1}^n (B_{ij} u^{(j-1)} \bar{v}^{(i-1)})(t) \\
&= (B\vec{u} \cdot \vec{v})(t) \quad (\text{by equation (2.8)}) \\
&=: \mathcal{S}(\vec{u}, \vec{v})(t).
\end{aligned} \tag{2.9}$$

With this notation, we can rewrite the right-hand side of Green's formula as a

semibilinear form below:

$$\begin{aligned}
[uv](t_2) - [uv](t_1) &= \sum_{j,k=1}^n B_{jk}(t_2) u^{(k-1)}(t_2) \bar{v}^{(j-1)}(t_2) - \sum_{j,k=1}^n B_{jk}(t_2) u^{(k-1)}(t_1) \bar{v}^{(j-1)}(t_1) \\
&= B(t_2) \vec{u}(t_2) \cdot \vec{v}(t_2) - B(t_1) \vec{u}(t_1) \cdot \vec{v}(t_1) \\
&= \begin{bmatrix} B_{11}(t_2) & \cdots & B_{1n}(t_2) \\ \vdots & \ddots & \vdots \\ B_{n1}(t_2) & \cdots & B_{nn}(t_2) \end{bmatrix} \begin{bmatrix} u(t_2) \\ \vdots \\ u^{(n-1)}(t_2) \end{bmatrix} \cdot \begin{bmatrix} \bar{v}(t_2) \\ \vdots \\ \bar{v}^{(n-1)}(t_2) \end{bmatrix} - \\
&\quad \begin{bmatrix} B_{11}(t_1) & \cdots & B_{1n}(t_1) \\ \vdots & \ddots & \vdots \\ B_{n1}(t_1) & \cdots & B_{nn}(t_1) \end{bmatrix} \begin{bmatrix} u(t_1) \\ \vdots \\ u^{(n-1)}(t_1) \end{bmatrix} \cdot \begin{bmatrix} \bar{v}(t_1) \\ \vdots \\ \bar{v}^{(n-1)}(t_1) \end{bmatrix} \\
&= \begin{bmatrix} -B_{11}(t_1) & \cdots & -B_{1n}(t_1) \\ \vdots & \ddots & \vdots \\ -B_{n1}(t_1) & \cdots & -B_{nn}(t_1) \end{bmatrix} \begin{bmatrix} u(t_1) \\ \vdots \\ u^{(n-1)}(t_1) \end{bmatrix} \cdot \begin{bmatrix} \bar{v}(t_1) \\ \vdots \\ \bar{v}^{(n-1)}(t_1) \end{bmatrix} + \\
&\quad \begin{bmatrix} B_{11}(t_2) & \cdots & B_{1n}(t_2) \\ \vdots & \ddots & \vdots \\ B_{n1}(t_2) & \cdots & B_{nn}(t_2) \end{bmatrix} \begin{bmatrix} u(t_2) \\ \vdots \\ u^{(n-1)}(t_2) \end{bmatrix} \cdot \begin{bmatrix} \bar{v}(t_2) \\ \vdots \\ \bar{v}^{(n-1)}(t_2) \end{bmatrix} \\
&= \begin{bmatrix} -B_{11}(t_1) & \cdots & -B_{1n}(t_1) & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -B_{n1}(t_1) & \cdots & -B_{nn}(t_1) & 0 & \cdots & 0 \\ 0 & \cdots & 0 & B_{11}(t_2) & \cdots & B_{1n}(t_2) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & B_{n1}(t_2) & \cdots & B_{nn}(t_2) \end{bmatrix} \begin{bmatrix} u(t_1) \\ \vdots \\ u^{(n-1)}(t_1) \\ u(t_2) \\ \vdots \\ u^{(n-1)}(t_2) \end{bmatrix} \cdot \begin{bmatrix} \bar{v}(t_1) \\ \vdots \\ \bar{v}^{(n-1)}(t_1) \\ \bar{v}(t_2) \\ \vdots \\ \bar{v}^{(n-1)}(t_2) \end{bmatrix} \\
&= \begin{bmatrix} -B(t_1) & 0_n \\ 0_n & B(t_2) \end{bmatrix} \begin{bmatrix} u(t_1) \\ \vdots \\ u^{(n-1)}(t_1) \\ u(t_2) \\ \vdots \\ u^{(n-1)}(t_2) \end{bmatrix} \cdot \begin{bmatrix} \bar{v}(t_1) \\ \vdots \\ \bar{v}^{(n-1)}(t_1) \\ \bar{v}(t_2) \\ \vdots \\ \bar{v}^{(n-1)}(t_2) \end{bmatrix} \\
&=: \widehat{B} \begin{bmatrix} \vec{u}(t_1) \\ \vec{u}(t_2) \end{bmatrix} \cdot \begin{bmatrix} \vec{v}(t_1) \\ \vec{v}(t_2) \end{bmatrix} \quad 14.
\end{aligned}$$

(2.10)

Recall that  $\det(\lambda A) = \lambda^n \det(A)$  for  $n \times n$  matrix  $A$ . Thus, since  $B(t_1)$  is  $n \times n$ , we have

$$\det \widehat{B} = \det(-B(t_1)) \det(B(t_2)) = (-1)^n \det B(t_1) \det B(t_2).$$

Since  $B(t)$  is nonsingular for  $t \in [a, b]$  (as shown before),  $\widehat{B}$  is nonsingular for  $t_1, t_2 \in [a, b]$ .

To recapitulate, given a linear differential operator  $L$  defined on an interval  $[a, b]$  with coefficients  $p_0(t), \dots, p_n(t)$ , from the Green's formula, we have defined a matrix  $B$  which depends on  $p_0, \dots, p_n$  and a matrix  $\widehat{B}$  which depends on  $B$  and  $[a, b]$ . These objects will be important in characterizing an adjoint boundary condition using the boundary-form formula, which we now turn to.

### Boundary-form formula

Before introducing the boundary-form formula, we need the following definitions and results concerning boundary conditions.

**Definition 2.8.** [14, p.286] Given any set of  $2mn$  complex constants  $M_{ij}, N_{ij}$  ( $i = 1, \dots, m; j = 1, \dots, n$ ), define  $m$  **boundary operators (boundary forms)**  $U_1, \dots, U_m$  for functions  $x$  on  $[a, b]$ , for which  $x^{(j)}$  ( $j = 1, \dots, n - 1$ ) exists at  $a$  and  $b$ , by

$$U_i x = \sum_{j=1}^n (M_{ij} x^{(j-1)}(a) + N_{ij} x^{(j-1)}(b)) \quad (i = 1, \dots, m) \quad (2.11)$$

$U_i$  are **linearly independent** if the only set of complex constants  $c_1, \dots, c_m$  for which

$$\sum_{i=1}^m c_i U_i x = 0$$

for all  $x \in C^{n-1}$  on  $[a, b]$  is  $c_1 = c_2 = \dots = c_m = 0$ .

**Definition 2.9.** [14, p.286] A **vector boundary form**  $U = (U_1, \dots, U_m)$  is a vector whose components are boundary forms. When  $U_1, \dots, U_m$  are linearly

independent, we say that  $U$  has full rank  $m$ . We assume  $U$  has full rank below.

With the above definitions, we can now write a set of homogeneous boundary conditions in matrix form. Define

$$\xi := \begin{bmatrix} x & x^{(1)} & \cdots & x^{(n-1)} \end{bmatrix}^\top, \quad (2.12)$$

$$U := \begin{bmatrix} U_1 & U_2 & \cdots & U_m \end{bmatrix}^\top, \quad (2.13)$$

and

$$M := \begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{m1} & \cdots & M_{mn} \end{bmatrix}, \quad N := \begin{bmatrix} N_{11} & \cdots & N_{1n} \\ \vdots & \ddots & \vdots \\ N_{m1} & \cdots & N_{mn} \end{bmatrix}. \quad (2.14)$$

Then the set of homogeneous boundary conditions in equation (2.2) can be written as

$$Ux = M\xi(a) + N\xi(b). \quad (2.15)$$

Indeed:

$$\begin{aligned} M\xi(a) + N\xi(b) &= \begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{m1} & \cdots & M_{mn} \end{bmatrix} \begin{bmatrix} x(a) \\ x'(a) \\ \vdots \\ x^{(n-1)}(a) \end{bmatrix} + \begin{bmatrix} N_{11} & \cdots & N_{1n} \\ \vdots & \ddots & \vdots \\ N_{m1} & \cdots & N_{mn} \end{bmatrix} \begin{bmatrix} x(b) \\ x'(b) \\ \vdots \\ x^{(n-1)}(b) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^n M_{1j}x^{(j-1)}(a) \\ \vdots \\ \sum_{j=1}^n M_{mj}x^{(j-1)}(a) \end{bmatrix} + \begin{bmatrix} \sum_{j=1}^n N_{1j}x^{(j-1)}(b) \\ \vdots \\ \sum_{j=1}^n N_{mj}x^{(j-1)}(b) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^n (M_{1j}x^{(j-1)}(a) + N_{1j}x^{(j-1)}(b)) \\ \vdots \\ \sum_{j=1}^n (M_{mj}x^{(j-1)}(a) + N_{mj}x^{(j-1)}(b)) \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} U_1x \\ \vdots \\ U_m x \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_m \end{bmatrix} x = Ux.$$

Based on the above, we propose another way to write  $Ux$ . Define the  $m \times 2n$  matrix

$$(M : N) := \begin{bmatrix} M_{11} & \cdots & M_{1n} & N_{11} & \cdots & N_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{m1} & \cdots & M_{mn} & N_{m1} & \cdots & N_{mn} \end{bmatrix}.$$

Then  $U_1, \dots, U_m$  are linearly independent if and only if  $\text{rank}(M : N) = m$ , or equivalently,  $\text{rank}(U) = m$ . Moreover,  $Ux$  can also be written as

$$\begin{aligned} Ux &= \begin{bmatrix} \sum_{j=1}^n (M_{1j}x^{(j-1)}(a) + N_{1j}x^{(j-1)}(b)) \\ \vdots \\ \sum_{j=1}^n (M_{mj}x^{(j-1)}(a) + N_{mj}x^{(j-1)}(b)) \end{bmatrix} \\ &= \begin{bmatrix} M_{11} & \cdots & M_{1n} & N_{11} & \cdots & N_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{m1} & \cdots & M_{mn} & N_{m1} & \cdots & N_{mn} \end{bmatrix} \begin{bmatrix} x(a) \\ \vdots \\ x^{(n-1)}(a) \\ x(b) \\ \vdots \\ x^{(n-1)}(b) \end{bmatrix} \\ &= (M : N) \begin{bmatrix} \xi(a) \\ \xi(b) \end{bmatrix}. \end{aligned}$$

Having proposed a compact way to represent a set of homogeneous boundary conditions, we begin building our way to characterizing the notion of adjoint boundary condition. First, we need the notion of a complementary boundary form.

**Definition 2.10.** [14, p.287] If  $U = (U_1, \dots, U_m)$  is any boundary form with

$\text{rank}(U) = m$  and  $U_c = (U_{m+1}, \dots, U_{2n})$  is any form with  $\text{rank}(U_c) = 2n - m$  such that  $(U_1, \dots, U_{2n})$  has rank  $2n$ , then  $U$  and  $U_c$  are **complementary boundary forms**.

Note that extending  $U_1, \dots, U_m$  to  $U_1, \dots, U_{2n}$  is equivalent to embedding the matrix  $(M : N)$  in a  $2n \times 2n$  nonsingular matrix (recall that a square matrix is nonsingular if and only if it has full rank).

The characterization of adjoint boundary conditions is given by the boundary-form formula. The boundary-form formula is motivated by writing the right-hand side of Green's formula (2.4) as the linear combination of a boundary form  $U$  and a complementary form  $U_c$ . Before finally getting to it, we need the following propositions.

**Proposition 2.11.** [14, p.287] *In the context of the semibilinear form (2.8), we have*

$$Sf \cdot g = f \cdot S^*g, \quad (2.16)$$

where  $*$  denotes the conjugate transpose.

*Proof.*

$$\begin{aligned} Sf \cdot g &= \sum_{i,j=1}^k s_{ij} f_j \bar{g}_i \quad (\text{by equation (2.8)}); \\ f \cdot S^*g &= \begin{bmatrix} f_1 \\ \vdots \\ f_k \end{bmatrix} \cdot \begin{bmatrix} \bar{s}_{11} & \cdots & \bar{s}_{k1} \\ \vdots & \ddots & \vdots \\ \bar{s}_{1k} & \cdots & \bar{s}_{kk} \end{bmatrix} \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix} \\ &= \begin{bmatrix} f_1 \\ \vdots \\ f_k \end{bmatrix} \cdot \begin{bmatrix} \sum_{j=1}^k \bar{s}_{j1} g_j \\ \vdots \\ \sum_{j=1}^k \bar{s}_{jk} g_j \end{bmatrix} \\ &= \sum_{i=1}^k f_i \cdot \left( \sum_{j=1}^k \bar{s}_{ji} g_j \right) \\ &= \sum_{i=1}^k f_i \cdot \left( \sum_{j=1}^k s_{ji} \bar{g}_j \right) \end{aligned}$$

$$= \sum_{i,j=1}^k s_{ji} f_i \bar{g}_j = Sf \cdot g.$$

**Proposition 2.12.** [14, p.287] Let  $\mathcal{S}$  be the semibilinear form associated with a nonsingular matrix  $S$ . Suppose  $\bar{f} := Ff$  where  $F$  is a nonsingular matrix. Then there exists a unique nonsingular matrix  $G$  such that if  $\bar{g} = Gg$ , then  $\mathcal{S}(f, g) = \bar{f} \cdot \bar{g}$  for all  $f, g$ .

*Proof.* Let  $G := (SF^{-1})^*$ , then

$$\begin{aligned} \mathcal{S}(f, g) &= Sf \cdot g \\ &= S(F^{-1}F)f \cdot g \\ &= SF^{-1}(Ff) \cdot g \\ &= SF^{-1}\bar{f} \cdot g \\ &= \bar{f} \cdot (SF^{-1})^*g \quad (\text{by equation (2.16)}) \\ &= \bar{f} \cdot G * g \\ &= \bar{f} \cdot \bar{g}. \end{aligned}$$

To see that  $G$  is nonsingular, note that  $\det G = \det((\overline{SF^{-1}})^T) = \det(\overline{SF^{-1}}) = \overline{\det(SF^{-1})} = \overline{\det(S)\det(F)^{-1}} \neq 0$  since  $S, F$  are nonsingular.  $\square$

**Proposition 2.13.** [14, p.287] Suppose  $\mathcal{S}$  is associated with the unit matrix  $E$ , i.e.,  $\mathcal{S}(f, g) = f \cdot g$ . Let  $F$  be a nonsingular matrix such that the first  $j$  ( $1 \leq j < k$ ) components of  $\bar{f} = Ff$  are the same as those of  $f$ . Then the unique nonsingular matrix  $G$  such that  $\bar{g} = Gg$  and  $\bar{f} \cdot \bar{g} = f \cdot g$  (as in Proposition 2.12) is such that the last  $k-j$  components of  $\bar{g}$  are linear combinations of the last  $k-j$  components of  $g$  with nonsingular coefficient matrix.

*Proof.* We note that for the condition on  $F$  to hold,  $F$  must have the form

$$\begin{bmatrix} E_j & 0_+ \\ F_+ & F_{k-j} \end{bmatrix}_{k \times k}$$

where  $E_j$  is the  $j \times j$  identity matrix,  $0_+$  is the  $j \times (k-j)$  zero matrix,  $F_+$  is a  $(k-j) \times j$  matrix, and  $F_{k-j}$  a  $(k-j) \times (k-j)$  matrix. Let  $G$  be the unique nonsingular matrix in Proposition 2.12. Write  $G$  as

$$\begin{bmatrix} G_j & G_- \\ G_= & G_{k-j} \end{bmatrix}_{k \times k}$$

where  $G_j, G_-, G_=, G_{k-j}$  are  $j \times j, j \times (k-j), (k-j) \times j, (k-j) \times (k-j)$  matrices, respectively. By the definition of  $G$ ,

$$f \cdot g = Ff \cdot Gg = \bar{f} \cdot Gg = G^* \bar{f} \cdot g = G^* F f \cdot g,$$

(where the third equality follows from a reverse application of equation (2.16) with  $\bar{f}$  as  $f$ ,  $G^*$  as  $S$ ) which implies

$$G^* F = E_k.$$

Since

$$\begin{aligned} G^* F &= \begin{bmatrix} G_j^* & G_=^* \\ G_-^* & G_{k-j}^* \end{bmatrix} \begin{bmatrix} E_j & 0_+ \\ F_+ & F_{k-j} \end{bmatrix} \\ &= \begin{bmatrix} G_j^* + G_=^* F_+ & G_=^* F_{k-j} \\ G_-^* + G_{k-j}^* F_+ & G_{k-j}^* F_{k-j} \end{bmatrix} \\ &= \begin{bmatrix} E_j & 0_{j \times (k-j)} \\ 0_{(k-j) \times j} & E_{k-j} \end{bmatrix}. \end{aligned}$$

Thus,  $G_=^* F_{k-j} = 0_+$ , the  $j \times (k-j)$  zero matrix. But  $\det F = \det(E_j) \cdot \det(F_{k-j}) \neq 0$ , so  $\det F_{k-j} \neq 0$  and we must have  $G_=^* = 0_+$ , i.e.,  $G_= = 0_{(k-j) \times j}$ . Thus,  $G$  is upper-triangular, and so  $\det G = \det G_j \cdot \det G_{k-j} \neq 0$ , which implies  $\det G_{k-j} \neq 0$

and  $G_{k-j}$  is nonsingular. Hence,

$$\bar{g} = Gg = \begin{bmatrix} G_j & G_- \\ 0_{(k-j) \times j} & G_{k-j} \end{bmatrix} \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix}$$

where  $G_{k-j}$  is the nonsingular coefficient matrix such that

$$\begin{bmatrix} \bar{g}_{j-1} \\ \vdots \\ \bar{g}_k \end{bmatrix} = G_{k-j} \begin{bmatrix} g_{j-1} \\ \vdots \\ g_k \end{bmatrix}.$$

□

We are finally ready to introduce the boundary-form formula, the theorem central to the construction of adjoint boundary conditions.

**Theorem 2.14.** [14, p.288] (*Boundary-form formula*) *Given any boundary form  $U$  of rank  $m$ , and any complementary form  $U_c$ , there exist unique boundary forms  $U_c^+$ ,  $U^+$  of rank  $m$  and  $2n - m$ , respectively, such that*

$$[xy](b) - [xy](a) = Ux \cdot U_c^+y + U_cx \cdot U^+y. \quad (2.17)$$

If  $\tilde{U}_c$  is any other complementary form to  $U$ , and  $\tilde{U}_c^+$ ,  $\tilde{U}^+$  the corresponding forms of rank  $m$  and  $2n - m$ , then

$$\tilde{U}^+y = C^*U^+y \quad (2.18)$$

for some nonsingular matrix  $C$ .

*Remark 2.15.*  $U^+$  is the key object which will be defined later as an adjoint boundary condition to  $U$ .

Note that the existence of  $[xy](t)$  implies that a linear differential operator is involved (see equation (2.5)). The matrices  $\widehat{B}$  and  $B$  in the proof also depend on

this linear differential operator.

Also note that the second statement in the theorem reflects the fact that adjoint boundary conditions are unique only up to linear transformation. This is why, when the need for rigor is greater than that for convenience, we take care to use “an” instead of “the” in referring to adjoint boundary condition.

*Proof.* Recall from equation (2.10) that the left-hand side of equation (2.17) can be considered as a semibilinear form  $\mathcal{S}(f, g) = \widehat{B}f \cdot g$  for vectors

$$f = \begin{bmatrix} x(a) \\ \vdots \\ x^{(n-1)}(a) \\ x(b) \\ \vdots \\ x^{(n-1)}(b) \end{bmatrix}, \quad g = \begin{bmatrix} y(a) \\ \vdots \\ y^{(n-1)}(a) \\ y(b) \\ \vdots \\ y^{(n-1)}(b) \end{bmatrix}$$

with the nonsingular matrix

$$\widehat{B} = \begin{bmatrix} -B(a) & 0_n \\ 0_n & B(b) \end{bmatrix},$$

where  $B$  is as in equation (2.7). Recall from equation (2.15) that

$$Ux = M\xi(a) + N\xi(b) = (M : N) \begin{bmatrix} \xi(a) \\ \xi(b) \end{bmatrix}$$

for  $\xi, M, N$  defined in equation (2.12) and equation (2.14). With the definition of  $f$ , we have  $f = \begin{bmatrix} \xi(a) \\ \xi(b) \end{bmatrix}$  and thus

$$Ux = (M : N)f.$$

By Definition 2.10,  $U_c x = (\tilde{M} : \tilde{N})f$  for two appropriate matrices  $\tilde{M}, \tilde{N}$  for which

$$H = \begin{bmatrix} M & N \\ \tilde{M} & \tilde{N} \end{bmatrix}_{2n \times 2n}$$

has rank  $2n$ . Thus,

$$\begin{bmatrix} Ux \\ U_c x \end{bmatrix} = \begin{bmatrix} (M : N)f \\ (\tilde{M} : \tilde{N})f \end{bmatrix} = \begin{bmatrix} M & N \\ \tilde{M} & \tilde{N} \end{bmatrix} f = Hf.$$

By Proposition 2.12, there exists a unique  $2n \times 2n$  nonsingular matrix  $J$  (in fact, with  $S = \hat{B}$ ,  $F = H$ ,  $J = G$ , and  $G = (SF^{-1})^*$ , we have  $J = (\hat{B}H^{-1})^*$ ) such that  $\mathcal{S}(f, g) = Hf \cdot Jg$ . Let  $U^+, U_c^+$  be such that

$$Jg = \begin{bmatrix} U_c^+ y \\ U^+ y \end{bmatrix},$$

then

$$[xy](b) - [xy](a) = \mathcal{S}(f, g) = Hf \cdot Jg = \begin{bmatrix} Ux \\ U_c x \end{bmatrix} \cdot \begin{bmatrix} U_c^+ y \\ U^+ y \end{bmatrix} = Ux \cdot U_c^+ y + U_c x \cdot U^+ y.$$

Thus, equation (2.17) holds.

The second statement in the theorem follows from Proposition 2.13 with  $Hf$  and  $Jg$  corresponding to  $f$  and  $g$ .  $\square$

### Characterization of adjoint boundary conditions

With the boundary-form formula, we are now able to fully characterize the notion of “adjoint” for boundary value problems. In this section, we begin by defining adjoint boundary condition and adjoint boundary value problem. We then explore some properties of these adjoints as relevant to their construction.

**Definition 2.16.** [14, p.288-89] For any boundary form  $U$  of rank  $m$  there is

associated the homogeneous boundary condition

$$Ux = 0 \quad (2.19)$$

for functions  $x \in C^{n-1}$  on  $[a, b]$ . If  $U^+$  is any boundary form of rank  $2n - m$  determined as in Theorem 2.14, then the homogeneous boundary condition

$$U^+x = 0 \quad (2.20)$$

is an **adjoint boundary condition** to (2.19).

Putting together  $L, L^+$  and  $U, U^+$ , we have the definition of boundary value problem and its adjoint boundary value problem. Specifically:

**Definition 2.17.** [14, p.291] If  $U$  is a boundary form of rank  $m$ , the problem of finding solutions of

$$\pi_m : Lx = 0 \quad Ux = 0$$

on  $[a, b]$  is a **homogeneous boundary value problem of rank  $m$** . The problem

$$\pi_{2n-m}^+ : L^+x = 0 \quad U^+x = 0$$

on  $[a, b]$  is the **adjoint boundary value problem to  $\pi_m$** .

In connection with the notion of adjoint problem introduced after Definition 2.3, we now have the following property of the adjoint analogous to equation (2.3).

**Proposition 2.18.** *By Green's formula (2.4) and the boundary-form formula (2.17),*

$$\langle Lu, v \rangle = \langle u, L^+v \rangle$$

*for all  $u \in C^n$  on  $[a, b]$  satisfying equation (2.19) and all  $v \in C^n$  on  $[a, b]$  satisfying equation (2.20).*

*Proof.*

$$\begin{aligned}
\langle Lu, v \rangle - \langle u, L^+v \rangle &= \int_a^b Lu \bar{v} dt - \int_a^b u(\overline{L^+v}) dt \\
&= [uv](a) - [uv](b) \quad (\text{by Green's formula (2.4)}) \\
&= Uu \cdot U_c^+v + U_c u \cdot U^+v \quad (\text{by boundary-form formula (2.17)}) \\
&= 0 \cdot U_c^+v + U_c u \cdot 0 \quad (\text{by equation (2.19) and equation (2.20)}) \\
&= 0.
\end{aligned}$$

□

Here, we treat the above result as a property of the adjoint after defining  $L^+$  and constructing  $U^+$ . In some cases, it is treated as the definition of the adjoint problem, especially if the adjoint problem is introduced using linear algebra, as described in the introduction to section 2.2. Historically, though, it is the adjoint problem that motivated the inner product characterization.

Now, we turn to one last result that would help us in the last step of the construction algorithm, namely checking whether an adjoint boundary condition is valid.

Just like how  $U$  is associated with two  $m \times n$  matrices  $M, N$ ,  $U^+$  is associated with two  $n \times (2n - m)$  matrices  $P, Q$  such that  $(P^* : Q^*)$  has rank  $2n - m$  and

$$U^+x = P^*\xi(a) + Q^*\xi(b). \quad (2.21)$$

The following theorem is motivated by characterizing the homogeneous boundary condition and its adjoint ((2.20) – (2.19)) in terms of the matrices  $M, N, P, Q$ . For our purpose, it provides a means to check whether the adjoint boundary condition we found via the boundary-form formula (Theorem 2.14) is indeed valid.

**Theorem 2.19.** [14, p.289] *The boundary condition  $U^+x = 0$  is adjoint to  $Ux = 0$  if and only if*

$$MB^{-1}(a)P = NB^{-1}(b)Q \quad (2.22)$$

where  $B(t)$  is the  $n \times n$  matrix associated with the form  $[xy](t)$  in equation (2.7).

*Proof.* Let  $\eta := (y, y', \dots, y^{(n-1)})$ , then  $[xy](t) = B(t)\xi(t) \cdot \eta(t)$  by equation (2.9).

Suppose  $U^+x = 0$  is adjoint to  $Ux = 0$ . By definition of adjoint boundary condition (2.20),  $U^+$  is determined as in Theorem 2.14. But by Theorem 2.14, in determining  $U^+$ , there exist boundary forms  $U_c, U_c^+$  of rank  $2n - m$  and  $m$ , respectively, such that equation (2.17) holds.

Put

$$U_c x = M_c \xi(a) + N_c \xi(b) \quad \text{rank}(M_c : N_c) = 2n - m$$

$$U_c^+ y = P_c^* \eta(a) + Q_c^* \eta(b) \quad \text{rank}(P_c^* : Q_c^*) = m.$$

Then by the boundary-form formula (2.17),

$$\begin{aligned} B(b)\xi(b) \cdot \eta(b) - B(a)\xi(a) \cdot \eta(a) &= (M\xi(a) + N\xi(b)) \cdot (P_c^*\eta(a) + Q_c^*\eta(b)) + \\ &\quad (M_c\xi(a) + N_c\xi(b)) \cdot (P^*\eta(a) + Q^*\eta(b)). \end{aligned}$$

By equation (2.16),

$$M\xi(a) \cdot P_c^*\eta(a) = P_c M \xi(a) \cdot \eta(a).$$

Thus,

$$\begin{aligned} B(b)\xi(b) \cdot \eta(b) - B(a)\xi(a) \cdot \eta(a) &= (P_c M + P M_c)\xi(a) \cdot \eta(a) + (Q_c M + Q M_c)\xi(a) \cdot \eta(b) \\ &\quad (P_c N + P N_c)\xi(b) \cdot \eta(a) + (Q_c N + Q N_c)\xi(b) \cdot \eta(b). \end{aligned}$$

Thus, we have

$$P_c M + P M_c = -B(a) \quad P_c N + P N_c = 0_n$$

$$Q_c M + Q M_c = 0_n \quad Q_c N + Q N_c = B(b).$$

Since  $\det B(t) \neq 0$  on  $t \in [a, b]$ ,  $B^{-1}(a)$ ,  $B^{-1}(b)$  exist, and thus

$$\begin{bmatrix} -B^{-1}(a)P_c & -B^{-1}(a)P \\ B^{-1}(b)Q_c & B^{-1}(b)Q \end{bmatrix} \begin{bmatrix} M & N \\ M_c & N_c \end{bmatrix} = \begin{bmatrix} E_n & 0_n \\ 0_n & E_n \end{bmatrix}.$$

Recall that  $\begin{bmatrix} M & N \\ M_c & N_c \end{bmatrix}$  has full rank, which means that it is nonsingular. Thus, the two matrices on the left are inverses of each other. So we also have

$$\begin{bmatrix} M & N \\ M_c & N_c \end{bmatrix} \begin{bmatrix} -B^{-1}(a)P_c & -B^{-1}(a)P \\ B^{-1}(b)Q_c & B^{-1}(b)Q \end{bmatrix} = \begin{bmatrix} E_m & 0_+ \\ 0_- & E_{2n-m} \end{bmatrix}.$$

Therefore,

$$-MB^{-1}(a)P + NB^{-1}(b)Q = 0_+,$$

which is equation (2.22).

Conversely, let  $U_1^+$  be a boundary form of rank  $2n - m$  such that

$$U_1^+y = P_1^*\eta(a) + Q_1^*\eta(b)$$

for appropriate  $P_1^*$ ,  $Q_1^*$  with  $\text{rank}(P_1^* : Q_1^*) = 2n - m$ . Suppose

$$MB^{-1}(a)P_1 = NB^{-1}(b)Q_1 \tag{2.23}$$

holds.

By the fundamental theorem of linear maps [15, p.63], if  $V$  is finite-dimensional and  $T \in \mathcal{L}(V, W)$ , then  $\dim \ker T = \dim V - \dim \text{range } T$ . Suppose  $A$  is a  $n \times k$  matrix, then  $A \in \mathcal{L}(\mathbb{R}^n, \mathbb{R}^k)$ . Thus, in a homogeneous system of linear equations  $Ax = 0$ , we have  $\dim \ker A = \dim A - \dim \text{range } A$ . That is, the dimension of solution space  $\ker A$  is the difference between the number of unknown variables and the rank of the coefficient matrix, or  $\dim \text{range } A$ . Therefore, letting  $u$  be a  $2n \times 1$  vector, there exist exactly  $2n - m$  linearly independent solutions of the

homogeneous linear system  $(M : N)_{m \times 2n} u = 0$ . By equation (2.23),

$$MB^{-1}(a)P_1 - NB^{-1}(b)Q = 0,$$

and thus

$$(M : N)_{m \times 2n} \begin{bmatrix} B^{-1}(a)P_1 \\ -B^{-1}(b)Q_1 \end{bmatrix}_{2n \times (2n-m)} = 0_{m \times (2n-m)}.$$

So the  $2n - m$  columns of the matrix

$$H_1 := \begin{bmatrix} B^{-1}(a)P_1 \\ -B^{-1}(b)Q_1 \end{bmatrix}$$

are solutions of this system. Since  $\text{rank}(P_1^* : Q_1^*) = 2n - m$ ,

$$\text{rank} \begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} = 2n - m.$$

Since  $B(a)$ ,  $B(b)$  are nonsingular,  $\text{rank}(H_1) = 2n - m$ .

If  $U^+x = P^*\xi(a) + Q^*\xi(b) = 0$  is a boundary condition adjoint to  $Ux = 0$ ,

then the matrix

$$\begin{bmatrix} -B^{-1}(a)P_c & -B^{-1}(a)P \\ B^{-1}(b)Q_c & B^{-1}(b)Q \end{bmatrix}_{2n \times 2n}$$

is nonsingular (because it has inverse  $\begin{bmatrix} M & N \\ M_c & N_c \end{bmatrix}$ ), i.e., it has full rank. Thus, if

$$H = \begin{bmatrix} -B^{-1}(a)P \\ B^{-1}(b)Q \end{bmatrix}_{n \times (2n-m)},$$

then  $\text{rank}(H) = 2n - m$ . Therefore, by equation (2.22), the  $2n - m$  columns of  $H$  also form  $2n - m$  linearly independent solutions of  $(M : N)u = 0$ , as in the case of  $H_1$ . Hence, there exists a nonsingular  $(2n - m) \times (2n - m)$  matrix  $A$  such

that  $H_1 = HA$  (change of basis in the solution space). Thus we have

$$\begin{bmatrix} B^{-1}(a)P_1 \\ -B^{-1}(b)Q_1 \end{bmatrix} = H_1 = HA = \begin{bmatrix} B^{-1}(a)PA \\ -B^{-1}(b)QA \end{bmatrix},$$

or  $P_1 = PA$ ,  $Q_1 = QA$ . Thus,

$$U_1^+y = P_1^*\eta(a) + Q_1^*\eta(b) = A^*P^*\eta(a) + A^*Q^*\eta(b) = A^*U^+y.$$

Since  $A^*$  is a linear map,  $U^+y = 0$  implies  $U_1^+y = A^*U^+y = 0$ . Since  $A^*$  is nonsingular,  $A^{*-1}$  is also a linear map, and  $A^{*-1}U_1^+y = U^+y$ . Thus,  $U_1^+y = 0$  implies  $U^+y = A^{*-1}U_1^+y = 0$ . Therefore,  $U^+y = 0$  if and only if  $U_1^+y = 0$ . Since  $U^+y = 0$  is adjoint to  $Ux = 0$ ,  $U_1^+y = 0$  is adjoint to  $Ux = 0$ .  $\square$

To recapitulate, the boundary-form formula (Theorem 2.14) provides the existence and construction of adjoint boundary condition, and Theorem 2.19 supplies a means to check whether a proposed adjoint boundary condition is valid. With these results, we are ready to formulate an algorithmic procedure to solve (1.7) using the Fokas method.

### 3 Algorithm

Consider an IBVP of the form (1.7). We now outline the algorithm to find its solution. The solution is given in equation (3.8), via explicit construction of an integral transform and inverse transform pair. In this section, we provide that explicit mathematical construction, and adapt it into an algorithm that can be implemented in software. The transform pair is displayed in equation (3.7), using notations developed throughout section 3.2.

### 3.1 Constructing the adjoint of a set of homogeneous boundary conditions

The adjoint boundary condition associated with the adjoint problem relates to that of the original problem in a way that is important to making the Fokas transform pairs work as we would like them to. In fact, the first step in implementing the Fokas method is to construct the adjoint of a set of homogeneous boundary conditions. To this end, an algorithm to construct adjoint boundary conditions has been devised and implemented in Julia, outlined as follows.

Define the differential operator

$$L := S$$

to be the spatial differential operator in equation (1.6), the interval

$$[a, b] := [0, 1]$$

to be the unit interval in (1.7), and  $U$  to be the vector boundary form associated with two  $n \times n$  matrices  $M, N$  given entry-wise by

$$M_{jk} := b_{jk}, \quad N_{jk} := \beta_{jk}$$

for  $b_{jk}, \beta_{jk}$  in equation (1.4). Then we have defined a boundary-value problem in the spatial variable from the original IBVP, namely

$$Lx = 0 \quad Ux = 0.$$

We seek to construct a valid adjoint boundary condition  $U^+x = 0$ .

### 3.1.1 Checking input

We first check that  $U_1, \dots, U_n$  are linearly independent. Recall from equation (2.15) that

$$Ux = M\xi(a) + N\xi(b).$$

Thus, it suffices to check whether  $\text{rank}(M : N) = n$ .  $U$  would be considered an invalid input if  $\text{rank}(M : N) \neq n$ .

### 3.1.2 Finding a candidate adjoint

Recall from Definition 2.10 that extending  $U_1, \dots, U_n$  to  $U_1, \dots, U_{2n}$  (where  $(U_{n+1}, \dots, U_{2n})$  is a complementary boundary form  $U_c$ ) is equivalent to embedding  $(M : N)$  in a  $2n \times 2n$  nonsingular matrix (where the newly added rows constitute  $(\tilde{M}, \tilde{N})$  associated with  $U_c$ ). Thus, we can construct this  $2n \times 2n$  nonsingular matrix as follows: Given the identity matrix  $E_{2n}$  and the  $n \times 2n$  matrix  $(M : N)$ , we append the rows of  $E_{2n}$  one by one to  $(M : N)$  and retain only those rows that make the rank of the resulting matrix increase.

The output from the above construction is a  $2n \times 2n$  matrix of the form

$$\begin{bmatrix} M & N \\ E' & E'' \end{bmatrix}$$

where the rows of  $E', E''$  are the first  $n$  entries and the last  $n$  entries of the retained rows of  $E_{2n}$ , respectively. We identify this matrix with the matrix

$$H = \begin{bmatrix} M & N \\ \tilde{M} & \tilde{N} \end{bmatrix}$$

in Theorem 2.14 (where  $\tilde{M}, \tilde{N}$  are associated with the complementary boundary form  $U_c x = \tilde{M}\xi(a) + \tilde{N}\xi(b)$ ).

Recall that the linear differential operator  $L$  is characterized by the functions  $p_0, \dots, p_n$  and the interval  $[a, b]$ . Let  $B$  be as in (2.7) which depends on  $p_0, \dots, p_n$ .

Construct  $\widehat{B}$  from  $B$  and  $[a, b]$  as in Theorem 2.14. Let  $J := (\widehat{B}H^{-1})^*$ . By Theorem 2.14 and Proposition 2.12, the matrix  $J$  is of the form

$$J = \begin{bmatrix} M' & N' \\ \tilde{M}' & \tilde{N}' \end{bmatrix}$$

where  $\tilde{M}', \tilde{N}'$  are associated with an adjoint  $U^+$  and  $M', N'$  with its complement  $U_c^+$ . Thus, we can identify  $(P^* : Q^*)$  with the last  $n$  rows of  $J$ . That is, identify  $P^*$  with  $\tilde{M}'$ , the lower-left  $n \times n$  submatrix of  $J$ , and  $Q^*$  with  $\tilde{N}'$ , the lower-right  $n \times n$  submatrix of  $J$ . Define  $U^+$  by

$$U^+x = P^*\xi(a) + Q^*\xi(b),$$

then we have found an adjoint  $U^+$  to  $U$ .

### 3.1.3 Checking the validity of the candidate adjoint

By Theorem 2.19, with

$$Ux = M\xi(a) + N\xi(b), \quad U^+x = P^*\xi(a) + Q^*\xi(b) \tag{3.1}$$

we can check whether the  $U^+$  found above is indeed a valid adjoint to  $U$  by checking

$$MB^{-1}(a)P = NB^{-1}(b)Q$$

where  $B$  is as in (2.7).

It may be worth noting that although the spatial differential operator which the above procedure will be applied to when using the Fokas method assumes the specific form given by equation (1.6), this procedure in fact works for any general linear differential operator of the form in equation (2.1). Systematic unit tests have been developed to test the procedure's implementation in Julia [16], accounting for linear differential operators with constant coefficients and

polynomial coefficients.

### 3.2 Constructing the Fokas transform pair

Having constructed the adjoint boundary conditions, we can now construct the Fokas transform pair. The Fokas transform pair is a pair of contour integral transform and inverse transform. The key to its construction is thus its integrands and the integration contours. This section presents material from [1], with definitions expanded and adapted to the algorithm.

#### Definition of the integrands

Define the matrices

$$b^* := P^*, \quad \beta^* := Q^*,$$

where  $P^*, Q^*$  are the matrices associated with the adjoint vector boundary form  $U^+$  in equation 3.1.

Let  $\alpha := e^{2\pi i/n}$ , where  $n$  is the spatial order of the IBVP in (1.7).

For complex variable  $\lambda$ , define  $W^+(\lambda)$ ,  $W^-(\lambda)$  as the  $n \times n$  matrices whose  $kj$ -entries are given by

$$W_{kj}^+(\lambda) := \sum_{r=0}^{n-1} (-i\alpha^{k-1}\lambda)^r b_{jr}^* \quad (3.2a)$$

$$W_{kj}^-(\lambda) := \sum_{r=0}^{n-1} (-i\alpha^{k-1}\lambda)^r \beta_{jr}^*, \quad (3.2b)$$

where  $b_{jr}^*$ ,  $\beta_{jr}^*$  are indexed from 0 to  $n - 1$  in  $r$ .

Define  $W$  as the  $n \times n$  matrix whose  $kj$ -entry is given by

$$W_{kj}(\lambda) := W_{kj}^+(\lambda) + W_{kj}^-(\lambda)e^{-\alpha^{k-1}\lambda}. \quad (3.3)$$

Let

$$\Delta(\lambda) := \det W(\lambda). \quad (3.4)$$

Note that by the definition of  $W(\lambda)$ ,  $\Delta$  is an exponential polynomial in  $\lambda$ . That

is,

$$\Delta(\lambda) = \sum_{k=1}^m a_k \lambda^k e^{b_k \lambda},$$

where  $a_k, b_k \in \mathbb{C}$  for  $k \in \{1, \dots, m\}$ .

Let  $X^{lj}$  be the  $(n-1) \times (n-1)$  submatrix of the block matrix  $\mathbb{W} := \begin{bmatrix} W & W \\ W & W \end{bmatrix}$ , where  $X_{11}^{lj}$  is  $\mathbb{W}_{l+1,j+1}$ .

For  $\lambda \in \mathbb{C}$  such that  $\Delta(\lambda) \neq 0$ , define the integrands

$$F_\lambda^+(f) := \frac{1}{2\pi\Delta(\lambda)} \sum_{l=1}^n \sum_{j=1}^n (-1)^{(n-1)(l+j)} \det X^{lj}(\lambda) W_{1j}^+(\lambda) \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x) dx \quad (3.5a)$$

$$F_\lambda^-(f) := \frac{-e^{-i\lambda}}{2\pi\Delta(\lambda)} \sum_{l=1}^n \sum_{j=1}^n (-1)^{(n-1)(l+j)} \det X^{lj}(\lambda) W_{1j}^-(\lambda) \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x) dx. \quad (3.5b)$$

### Definition of the integration contours

We now define the integration contours. By the theory of exponential polynomials [17], the infimum of the distances between every two distinct zeros of  $\Delta(\lambda)$  is positive. Let  $5\epsilon$  be that infimum. Let  $\partial S$  denote boundary of set  $S$ . Let  $\text{cl } S$  denote the closure of set  $S$ . Let  $\mathbb{C}^\pm$  denote the open upper and open lower halves of the complex plane, respectively. Let  $D(x_0, \epsilon)$  denote the disk  $\{x \in \mathbb{C} : |x - x_0| < \epsilon\}$  centered at  $x_0$  with radius  $\epsilon$ . Let  $C(x_0, \epsilon)$  denote the circle  $\{x \in \mathbb{C} : |x - x_0| = \epsilon\}$  centered at  $x_0$  with radius  $\epsilon$ . Define the contours

$$\Gamma_a^\pm := \partial \left( \{\lambda \in \mathbb{C}^\pm : \operatorname{Re}(a\lambda^n) > 0\} \setminus \bigcup_{\substack{\sigma \in \mathbb{C}; \\ \Delta(\sigma)=0}} D(\sigma, 2\epsilon) \right) \quad (3.6a)$$

$$\Gamma_a := \Gamma_a^+ \cup \Gamma_a^- \quad (3.6b)$$

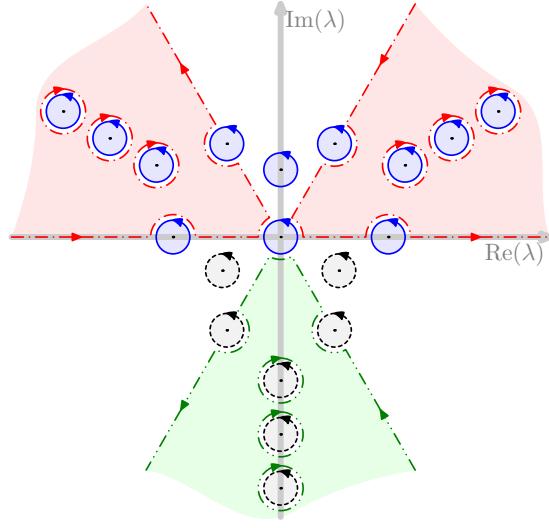
$$\Gamma_0^+ := \bigcup_{\substack{\sigma \in \text{cl } \mathbb{C}^+; \\ \Delta(\sigma)=0}} C(\sigma, \epsilon) \quad (3.6c)$$

$$\Gamma_0^- := \bigcup_{\substack{\sigma \in \mathbb{C}^-; \\ \Delta(\sigma) = 0}} C(\sigma, \epsilon) \quad (3.6d)$$

$$\Gamma_0 := \Gamma_0^+ \cup \Gamma_0^- \quad (3.6e)$$

$$\Gamma := \Gamma_0 \cup \Gamma_a. \quad (3.6f)$$

A sample contour is shown in Figure 3.



**Figure 3:** Hypothetical  $\Gamma$  contours corresponding to the case  $a = -i$ , taken from [1]. The red dashed lines correspond to  $\Gamma_a^+$ , and the green dashed lines correspond to  $\Gamma_a^-$ . The blue circles correspond to  $\Gamma_0^+$ , and the black dashed circles correspond to  $\Gamma_0^-$ . The arrows indicate the order of integration in equation (3.8).

### Definition of the Fokas transform pair

With the integrands  $F_\lambda^+$ ,  $F_\lambda^-$  and the contours  $\Gamma$ , the Fokas transform pair is given by

$$F_\lambda : f(x) \mapsto F(\lambda) \quad F_\lambda(f) = \begin{cases} F_\lambda^+(f) & \text{if } \lambda \in \Gamma_0^+ \cup \Gamma_a^+, \\ F_\lambda^-(f) & \text{if } \lambda \in \Gamma_0^- \cup \Gamma_a^-, \end{cases} \quad (3.7a)$$

$$f_x : F(\lambda) \mapsto f(x) \quad f_x(F) = \int_{\Gamma} e^{i\lambda x} F(\lambda) d\lambda, \quad x \in [0, 1], \quad (3.7b)$$

which allows computing the IBVP solution  $q(x, t)$  in (1.8) by

$$\begin{aligned} q(x, t) &= f_x(e^{-a\lambda^n t} F_\lambda(f)) \\ &:= \int_{\Gamma_0^+} e^{i\lambda x} e^{-a\lambda^n t} F_\lambda^+(f) d\lambda + \int_{\Gamma_a^+} e^{i\lambda x} e^{-a\lambda^n t} F_\lambda^+(f) d\lambda \\ &\quad + \int_{\Gamma_0^-} e^{i\lambda x} e^{-a\lambda^n t} F_\lambda^-(f) d\lambda + \int_{\Gamma_a^-} e^{i\lambda x} e^{-a\lambda^n t} F_\lambda^-(f) d\lambda. \end{aligned} \quad (3.8)$$

### 3.2.1 Constructing the integration contours

All objects defined above except for the contour  $\Gamma$ , namely  $W^+(\lambda)$ ,  $W^-(\lambda)$ ,  $W(\lambda)$ ,  $\Delta(\lambda)$ ,  $X^{lj}$ ,  $F_\lambda^+(f)$ ,  $F_\lambda^-(f)$ , can be constructed directly from their explicit expressions. Thus, to construct the Fokas transform pair  $F_\lambda$  and  $f_x$ , we only need to find an explicit characterization of the contour  $\Gamma$ .

We note that by (3.6),  $\Gamma_a^+$  and  $\Gamma_a^-$  are boundaries of the region  $\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}$  in the upper and lower half planes, respectively, deformed to avoid poles of the integrands  $F_\lambda^+$  and  $F_\lambda^-$ , i.e., zeros of  $\Delta(\lambda)$ . On the other hand,  $\Gamma_0^+$  and  $\Gamma_0^-$  consist of circles of radius  $\epsilon$  around the integrands' poles, or zeros of  $\Delta(\lambda)$ , in the closed upper half plane and the open lower half plane, respectively. Thus, to find  $\Gamma$ , we need to find the region  $\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}$  and the zeros of  $\Delta(\lambda)$ .

#### Tracing contour backbone

We first find the region

$$\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}.$$

For  $z \in \mathbb{C}$ , define  $\theta_z := \arg(z)$  and  $r_z := |z|$ . Let  $\lambda \in \mathbb{C}$ . Then

$$a\lambda^n = r_a e^{i\theta_a} \cdot r_\lambda^n e^{ni\theta_\lambda} = r_a r_\lambda^n e^{i(\theta_a + n\theta_\lambda)} = r_a r_\lambda^n (\cos(\theta_a + n\theta_\lambda) + i \sin(\theta_a + n\theta_\lambda)).$$

Thus,

$$\operatorname{Re}(a\lambda^n) = r_a r_\lambda^n \cos(\theta_a + n\theta_\lambda),$$

where

$$\operatorname{Re}(a\lambda^n) > 0 \iff \cos(\theta_a + n\theta_\lambda) > 0.$$

But

$$\cos(\theta_a + n\theta_\lambda) > 0 \iff \theta_a + n\theta_\lambda \in \bigcup_{k \in \mathbb{Z}} \left( 2\pi k - \frac{\pi}{2}, 2\pi k + \frac{\pi}{2} \right),$$

and so

$$\theta_\lambda \in \bigcup_{k \in \mathbb{Z}} \left( \frac{2\pi k - \frac{\pi}{2} - \theta_a}{n}, \frac{2\pi k + \frac{\pi}{2} - \theta_a}{n} \right).$$

For example, when  $a = -i$  and  $n = 3$ ,

$$\theta_\lambda \in \bigcup_{k \in \mathbb{Z}} \left( \frac{2\pi k - \frac{\pi}{2} - (-\frac{\pi}{2})}{3}, \frac{2\pi k + \frac{\pi}{2} - (-\frac{\pi}{2})}{3} \right) = \bigcup_{k \in \mathbb{Z}} \left( \frac{2\pi k}{3}, \frac{2\pi k + \pi}{3} \right).$$

Thus, in  $[0, 2\pi)$ ,

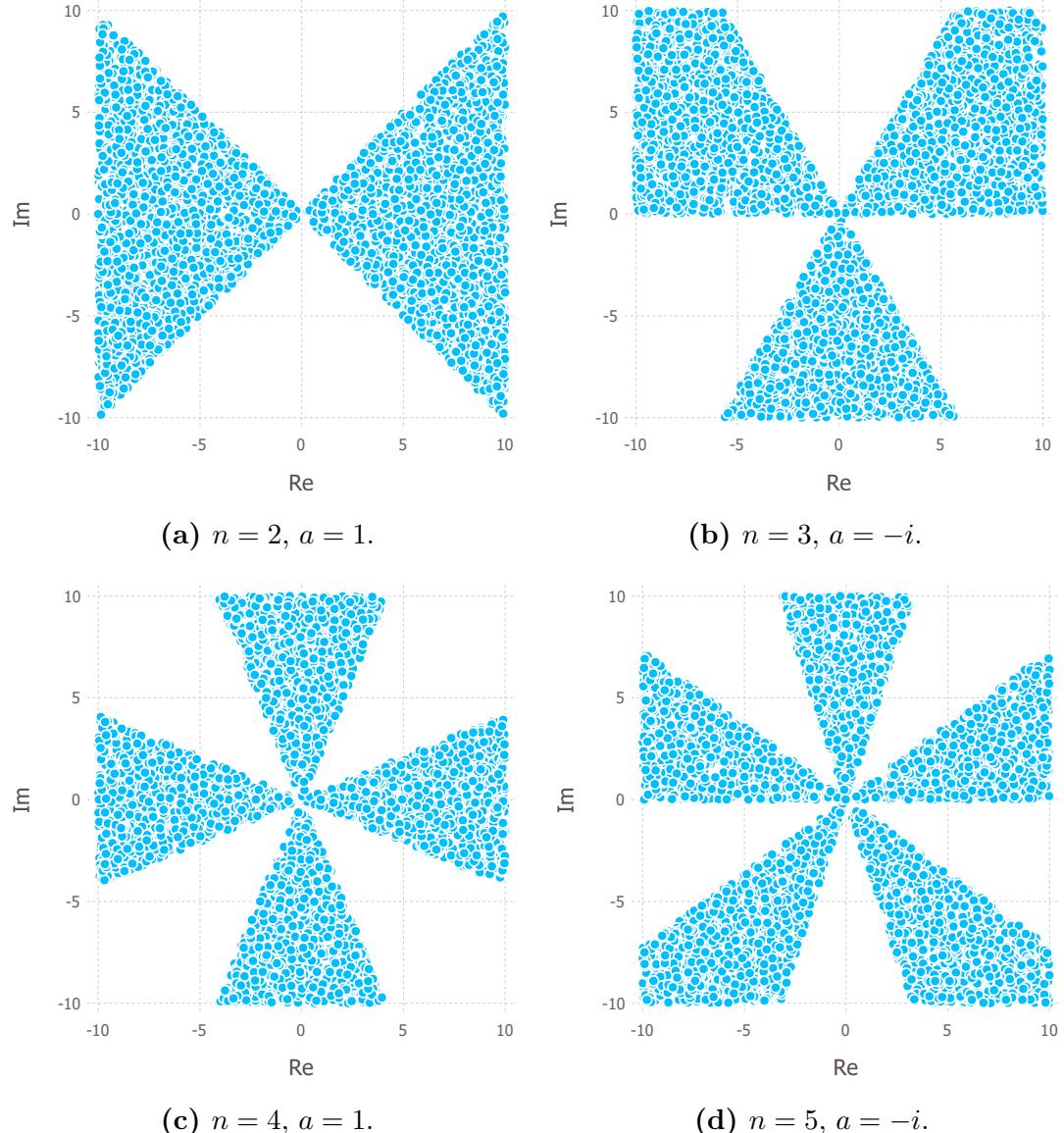
$$\theta_\lambda = \arg(\lambda) \in \bigcup_{k \in \{0,1,2\}} \left( \frac{2\pi k}{3}, \frac{2\pi k + \pi}{3} \right) = \left( 0, \frac{\pi}{3} \right) \cup \left( \frac{2\pi}{3}, \pi \right) \cup \left( \frac{4\pi}{3}, \frac{5\pi}{3} \right).$$

Figure 4 shows some simulations of the region  $\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}$ . Note that the region consists of a number of connected components, which we henceforth refer to as sectors.

### Deforming contour backbone to avoid poles of the integrands

The boundaries of the sectors found above form the “backbones” of the contours  $\Gamma_a^+$  and  $\Gamma_a^-$ . We proceed to turn these “backbones” into  $\Gamma_a^+$ ,  $\Gamma_a^-$  by deforming them to avoid the poles of the integrands  $F_\lambda^+$  and  $F_\lambda^-$ , i.e., the zeros of  $\Delta(\lambda)$ .

Before going into the technicalities of the contour deformation, we first draw attention to a modification of the mathematical construction of  $\Gamma$  in the actual implementation of the algorithm. In (3.6),  $\Gamma$  is defined on the entire complex plane; that is, we can have  $\lambda$  with arbitrarily large modulus. In the actual implementation, however, we choose a sufficiently large real number  $d$  to represent the modulus of complex infinity in the algorithm’s context. That is, we restrict the mathematical construction to the disk  $D(0, d)$ . As the choice of  $d$  potentially



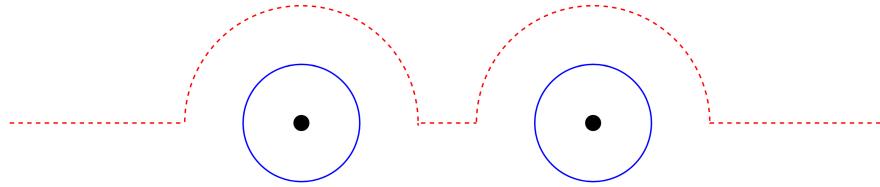
**Figure 4:** Simulation of the region  $\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}$  by randomly sampling  $10^4$  points  $\lambda$  and keeping those that satisfy  $\operatorname{Re}(a\lambda^n) > 0$ .

implies a tradeoff between computational cost and numerical accuracy, it should be chosen with regard to the error tolerance of the computational task at hand. The figures in this report and the documentation of the implementation of the Fokas method [16] are obtained with  $d = 10$  for clearer illustration. Note that the original infinite contour integrals are understood as improper integrals which, whenever they exist, are equal to their principal values, namely the limits of the respective integrals as the integration bounds tend to infinity. Truncating within  $D(0, d)$  and expecting convergence as  $d \rightarrow \infty$  is therefore legitimate.

By the theory of exponential polynomials [17],  $\Delta(\lambda)$  has infinitely many zeros, yet any finite region contains only finitely many of these zeros. Since we restrict the construction to the finite region  $D(0, d)$ , we would only be dealing with finitely many zeros of  $\Delta(\lambda)$ . This is an important fact: One complication that arises in practice is that each circular contour around a zero of  $\Delta(\lambda)$  not only cannot overlap with the circle around another zero but also cannot overlap with the boundary of any sector. That is, in Figure 3, the blue and black circles not only cannot overlap with each other, but they also cannot overlap with the red or green dashed lines if the zeros of  $\Delta(\lambda)$  at their centers do not lie on the boundaries of the red or green sectors. This implies that the  $5\epsilon$  introduced in section 3.2 in fact needs to be the minimum of the infimum of the pairwise distances between distinct zeros of  $\Delta(\lambda)$  and the infimum of distances between each zero that is exterior to all sectors and the boundary of each sector. The theory of exponential polynomials [17] guarantees that the former infimum is positive, but there is no such guarantee for the latter infimum. However, precisely because we are working with a finite number of zeros of  $\Delta(\lambda)$  in  $D(0, d)$ , the latter infimum is indeed positive, and so is  $\epsilon$ .

To see that our choice of  $\epsilon$  is valid, suppose  $5\epsilon$  is the minimum of the two infima described above. Then the circle of radius  $\epsilon$  around a zero of  $\Delta(\lambda)$  does not overlap with any sector boundary. Moreover, the contours around any two distinct zeros of  $\Delta(\lambda)$  do not overlap. Indeed: In Figure 5, suppose the radius of

the blue circle around a zero of  $\Delta(\lambda)$  is  $\epsilon$  and the distance between the blue circle and the red dashed curve around it is again  $\epsilon$ . Then since the distance between any two distinct zeros of  $\Delta(\lambda)$  is no less than  $5\epsilon$ , the blue circle together with the red dashed curve around it is still of distance  $5\epsilon - 2\epsilon \cdot 2 = \epsilon$  from the red dashed curve and the blue circle surrounding the nearest zero of  $\Delta(\lambda)$ . Thus, our choice of  $\epsilon$  ensures that no contour around a zero of  $\Delta(\lambda)$  is “overwritten” by the contour around another zero.



**Figure 5:** Two distinct zeros of  $\Delta(\lambda)$  (black dots) that are  $5\epsilon$  apart. The blue circles and red dashed contours are as in Figure 3. The radius of the blue circles, the distance between each blue circle and the red dashed curve around it, and the distance between the two red dashed curves, are all  $\epsilon$ .

Suppose for now that we have been given finitely many zeros of  $\Delta(\lambda)$  in  $D(0, d)$  (we will show how to approximate these zeros at the end of this section).

In the complex plane, a ray (straight half-line) emanating from the origin is characterized by the argument of any point on the ray. From Figure 4, we note that each sector in the region  $\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}$  is characterized by the arguments of the two rays that mark its boundaries, namely the starting ray and the ending ray, such that if the sector is traversed counterclockwise from one end to the other, the starting ray will be traversed before the ending ray. For example, with  $n = 3$  and  $a = -i$ , as shown in section 3.2.1, the region  $\{\lambda \in \mathbb{C} : \operatorname{Re}(a\lambda^n) > 0\}$  is characterized by the rays with arguments  $0, \frac{\pi}{3}, \frac{2}{3}\pi, \pi, \frac{4}{3}\pi, \frac{5}{3}\pi$ , where  $0, \frac{2}{3}\pi, \frac{4}{3}\pi$  are the angles of the starting rays, and  $\frac{\pi}{3}, \pi, \frac{5}{3}\pi$  are the angles of the ending rays.

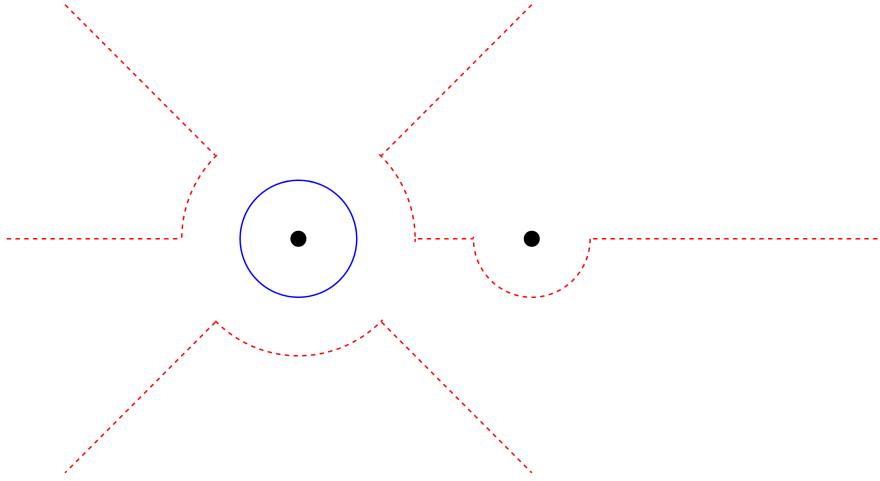
Note that each sector is assigned to  $\Gamma_a^+$  or  $\Gamma_a^-$  depending on whether it belongs to the upper or the lower half plane. As shown in Figure 8, if a sector overlaps with the real line, we divide it into the upper and lower half planes, which can then be assigned to  $\Gamma_a^+$  and  $\Gamma_a^-$ , respectively. In the implementation, this is done

by modifying the starting and ending rays that characterize the sector so that the sector is split into two sub-sectors, one ending at the real line and the other starting at it.

The contours  $\Gamma_a^\pm$  are built from these rays with possible deformations to avoid zeros of  $\Delta(\lambda)$ . Here we make another modification of the mathematical construction in (3.6) to save computational cost. We note that the contours  $\Gamma_a^\pm$  and  $\Gamma_0^\pm$  defined in (3.6) lead to partial cancellation, and deformation is in fact only necessary if a zero lies on the boundary of some sector. Indeed: As illustrated in Figure 3, the red and blue dashed lines do not need to be deformed to avoid zeros exterior to all sectors because they are already outside the sectors. Moreover, zeros interior to any sector can be ignored, since integrating over the blue circle around the zero counterclockwise cancels with integrating over the interior red dashed circle around the zero clockwise. For a zero that lies on the boundary of the sector, half of the blue circle around it that is interior to the sector cancels out with the red dashed curve around it, but the other half that is exterior to the sector does not cancel with anything. In this case, we deform the ray on which the zero lies to include the exterior half of the blue circle that does not cancel.

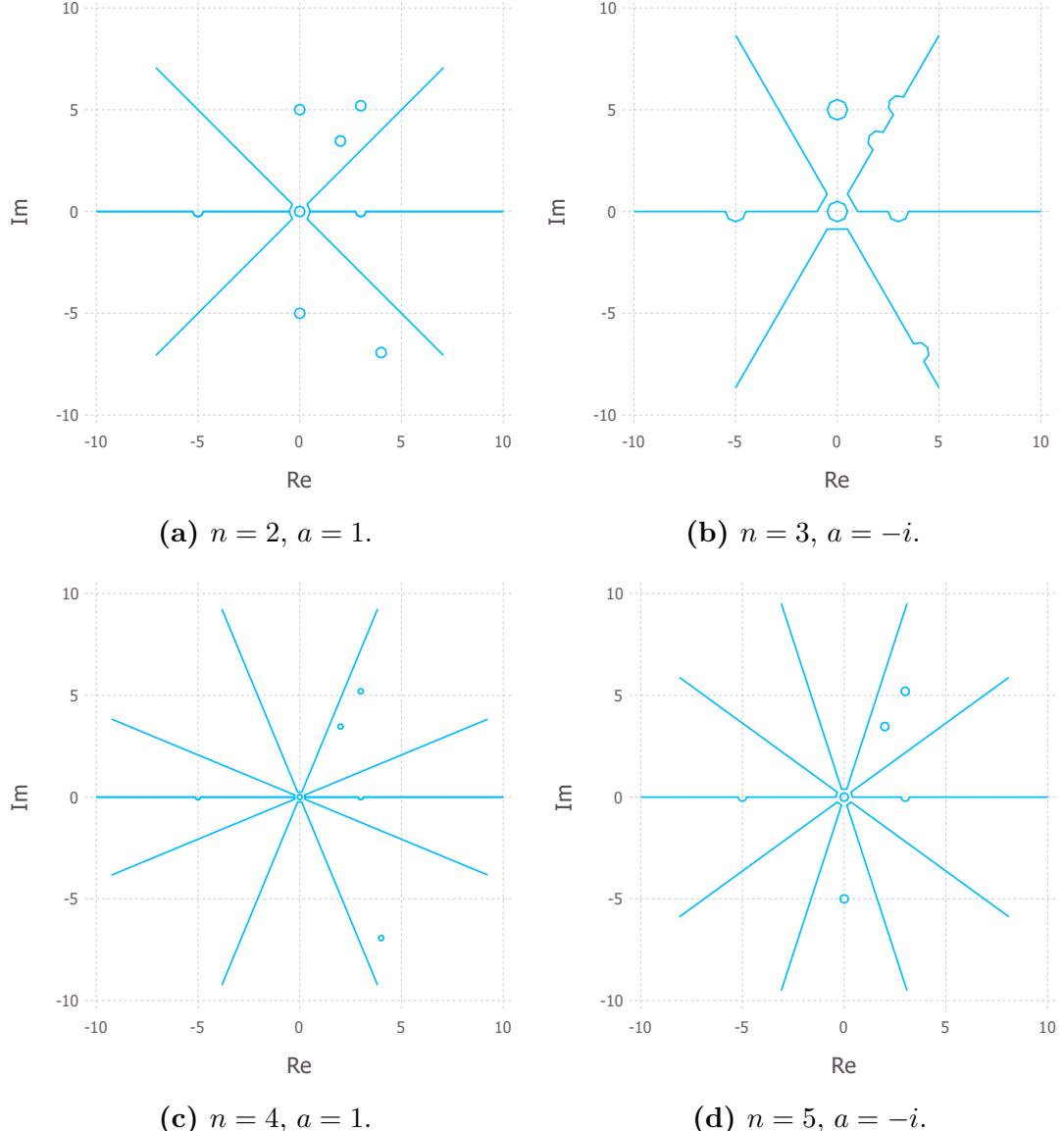
On a side note, the way we deform the contours described above implies that we can choose  $\epsilon$  to be  $\frac{1}{4}$  the minimum of the pairwise distances between distinct zeros of  $\Delta(\lambda)$  and the distances between any zero to any sector boundary, instead of  $\frac{1}{5}$  as in section 3.2. Indeed: Since we have merged the blue circle into the red dashed curve in the previous paragraph, the scenario in Figure 5 would not occur. Instead, the scenario that puts the most constraint on  $\epsilon$  occurs when a zero is at the origin, and as shown in Figure 6, this scenario only requires that the two distinct zeros are  $4\epsilon$  apart. Besides mathematical clarity, there is a subtlety in the implementation that actually argues for using  $4\epsilon$  over  $5\epsilon$ : Since zeros of  $\Delta(\lambda)$  are poles of the integrands  $F_\lambda^+$ ,  $F_\lambda^-$  in (3.5), the values of  $F_\lambda^+$ ,  $F_\lambda^-$  easily blow up near the zeros, thereby creating numerical instability that could make integration very slow. Thus, we use the new choice of  $\epsilon$  in the actual implementation to make the

contours stay further away from the zeros. Figure 7 shows the contour  $\Gamma$  for some examples, and Figure 8 shows each component of  $\Gamma$  of Figure 7(a) separately.

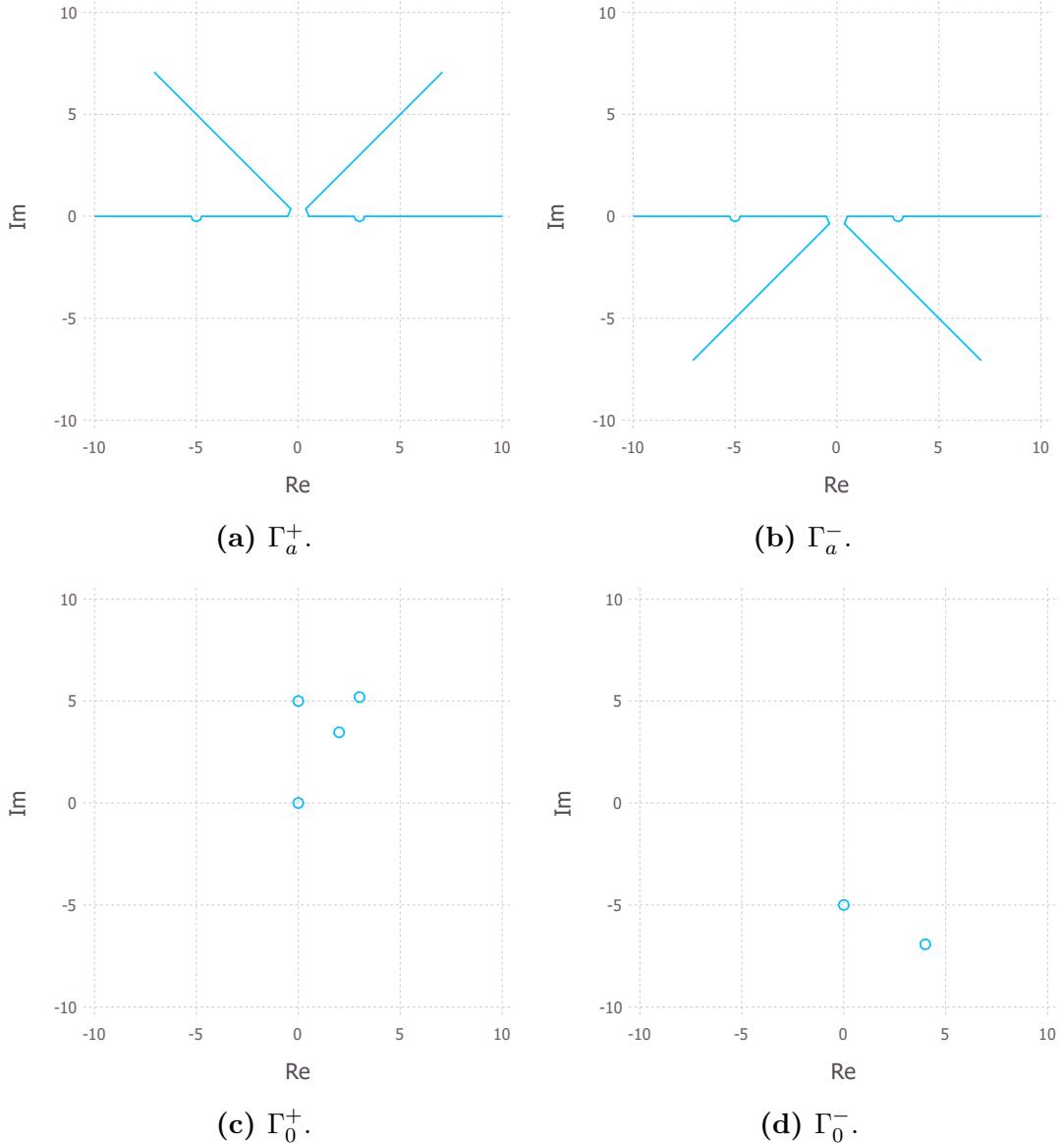


**Figure 6:** Two distinct zeros of  $\Delta(\lambda)$  (black dots) that are  $4\epsilon$  apart. The zero on the left is at the origin. The blue circles and red dashed contours are as in Figure 3 (though not drawn to scale here). The radius of the blue circle, the distance between the blue circle and the red dashed curve around it, and the distance between the red dashed curve around the blue circle and the red dashed curve around the zero on the right, are all  $\epsilon$ .

The construction of  $\Gamma_a^+$ ,  $\Gamma_a^-$ ,  $\Gamma_0^+$ , and  $\Gamma_0^-$  is described as follows. In the implementation, each contour in  $\Gamma$  is encoded by an array of points listed in the order of integration. As shown in Figure 3, we integrate over each contour in  $\Gamma_a^\pm$  from the ending ray to the starting ray, and each contour in  $\Gamma_0^\pm$  counterclockwise. Thus, in constructing the contours, for each sector, we initialize its boundary contour by an array of three points, namely the point on the ending ray with modulus  $d$ , the origin, and the point on the starting ray with modulus  $d$ . We add these initial sector boundary contours to  $\Gamma_a^\pm$  depending on whether the sectors are in the upper or lower half plane. Then, for each zero of  $\Delta(\lambda)$  that is not at the origin, if it is on the boundary of some sector, we deform the boundary contour of that sector to include a “bulge” that is the exterior half of an  $N$ -gon of radius  $\epsilon$  around the zero, with the vertices listed counterclockwise. If the zero is exterior to all sectors, we add an  $N$ -gon contour around it to  $\Gamma_0^\pm$ , depending on whether the zero is in the upper or lower half plane. If the zero is interior to any sector, we ignore it, since its contribution to the contour integral will be cancelled out,



**Figure 7:**  $\Gamma$  with with zeros of  $\Delta(\lambda)$  at  $3 + 3\sqrt{3}i, 2 + 2\sqrt{3}i, 0 + 0i, 0 + 5i, 0 - 5i, 3, -5,$  and  $4 - 4\sqrt{3}i.$  Note that  $\Gamma_a^+$  are deformed to avoid zeros on the sectors' boundaries.



**Figure 8:** Components of  $\Gamma$  with  $n = 2$ ,  $a = 1$ , and the same zeros of  $\Delta(\lambda)$  as in Figure 7. Note that the sectors are split at the real line into the upper and lower half planes.

as explained before. Finally, if there is a zero at the origin, we draw an  $N$ -gon around it and deform the boundary contours of all sectors to avoid the  $N$ -gon.

### Approximating poles of the integrands

As promised, we now describe a method to approximate the poles of  $F_\lambda^+$  and  $F_\lambda^-$ , i.e., the zeros of  $\Delta(\lambda)$ , in the finite region  $D(0, d)$ .

Writing  $\Delta(\lambda) = \text{Re}(\Delta)(\lambda) + i\text{Im}(\Delta)(\lambda)$  for  $\text{Re}(\Delta), \text{Im}(\Delta) : \mathbb{C} \rightarrow \mathbb{R}$ , we note that  $\lambda_0$  is a root of  $\Delta(\lambda)$  if and only if it is both a root of  $\text{Re}(\Delta)(\lambda)$  and a root of  $\text{Im}(\Delta)(\lambda)$ . Thus, to find zeros of  $\Delta(\lambda)$ , we aim to solve simultaneously

$$\text{Re}(\Delta)(\lambda) = 0, \tag{3.9a}$$

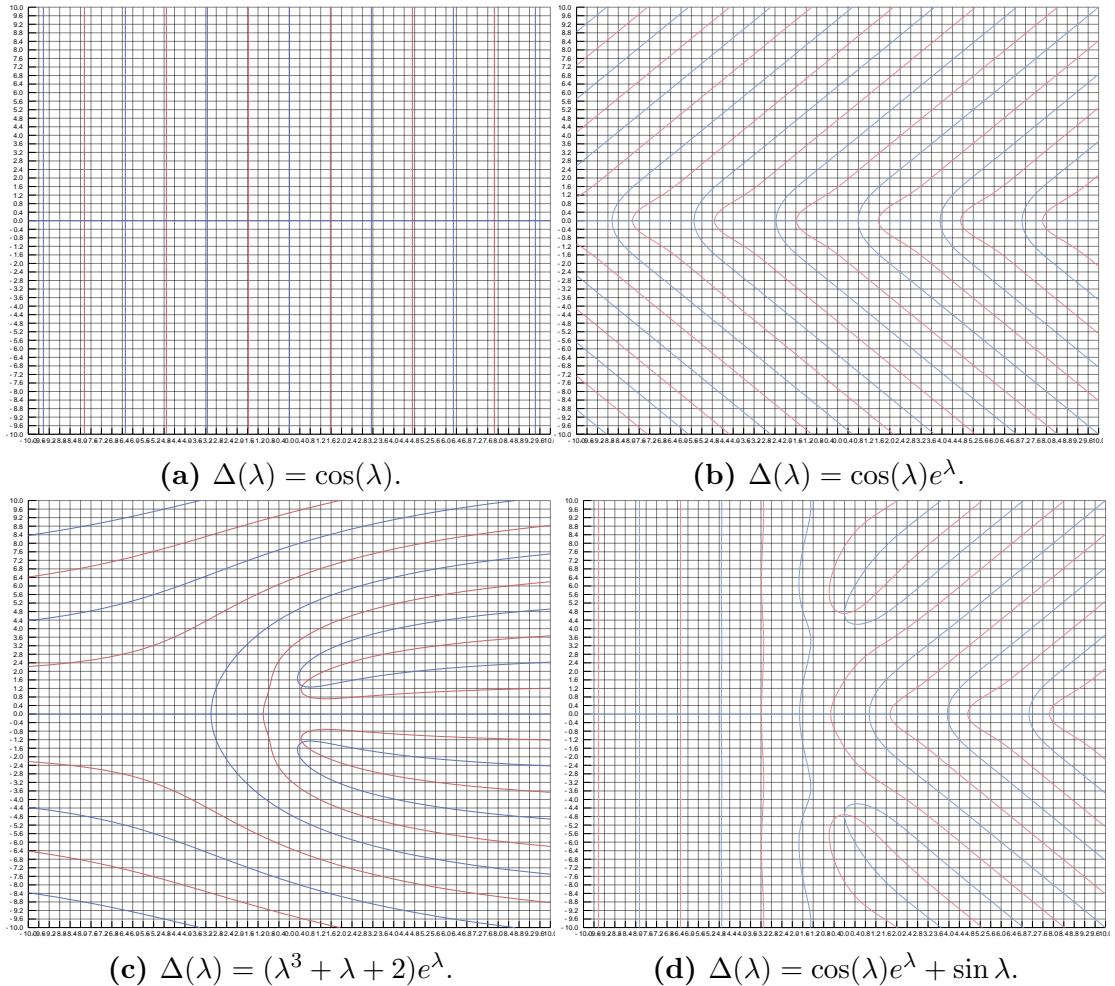
$$\text{Im}(\Delta)(\lambda) = 0. \tag{3.9b}$$

We approximate the solutions to (3.9) by visualizing their corresponding level curves. In the examples shown in Figure 9, the red lines correspond to the zeros of  $\text{Re}(\Delta)(\lambda)$ , and the blue lines correspond to those of  $\text{Im}(\Delta)(\lambda)$ . Thus, We can locate the zeros of  $\Delta(\lambda)$  at the intersections of the red and blue lines.

The above method still requires a human to read the level curve plots, visually locate the zeros of  $\Delta(\lambda)$ , and record them in an array, which can then be used as an input to the algorithm that produces  $\Gamma$ . Yet this does not introduce much inaccuracy: Recall that when constructing  $\Gamma$ , we need to deform the contours to avoid zeros of  $\Delta(\lambda)$  with  $N$ -gons of radius  $\epsilon$ . Thus, there is relatively high tolerance for our approximation of the locations of the zeros.

#### 3.2.2 Constructing the integrands

Having constructed the integration contours as above, theoretically, we can already compute the Fokas transform pair  $F_\lambda$  and  $f_x$  using equation (3.7) and solve for  $q(x, t)$  using equation (3.8). Yet there is one more complication in practice. As shown in equation (3.8), the solution  $q(x, t)$  is a sum of integrals of  $F_\lambda^+$ ,  $F_\lambda^-$  over the contours  $\Gamma$ , but as shown in (3.5a)–(3.5b),  $F_\lambda^+$  and  $F_\lambda^-$  themselves in-



**Figure 9:** Level curves corresponding to (3.9).

volve an integral of the initial datum  $f(x)$ . Double integrals are computationally expensive, and so to improve algorithm efficiency, we replace the inner integral involving  $f(x)$  with an approximation of it that has explicit formula, developed as follows.

### Approximating the inner integral using Chebyshev polynomials

The inner integral in (3.5) that we wish to approximate is

$$\int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x) dx,$$

where  $f(x)$  is the initial datum of the IBVP in (1.7). Note that this is the Fourier transform of  $f(x)$  on  $[0, 1]$  evaluated at  $\alpha^{l-1}\lambda$ . The idea is to approximate  $f(x)$  using Chebyshev polynomials and compute the integral using explicit formulas of the Fourier transform of Chebyshev polynomials [18].

In this section,  $n$  refers to the index of terms in the Chebyshev approximation of  $f(x)$  instead of the spatial order of the IBVP in (1.7).

The Chebyshev polynomials are given by

$$T_n(x) = \begin{cases} \cos(n \arccos(x)) & |x| \leq 1 \\ \frac{1}{2} ((x - \sqrt{x^2 - 1})^n + (x + \sqrt{x^2 - 1})^n) & |x| > 1, \end{cases}$$

for  $n = 0, 1, 2, \dots$ . Given a function  $f_0(x)$  on the interval  $[a, b]$ , we can approximate  $f_0(x)$  using Chebyshev polynomial expansion. The Chebyshev approximation is best on  $[-1, 1]$ . Thus, we scale  $[a, b]$  to  $[-1, 1]$  via  $g : [a, b] \rightarrow [-1, 1]$  given by  $g(x) = 2(x - a)/(b - a) - 1$ , obtain the Chebyshev coefficients  $\{b_n\}_{n=0}^N$  there, and write the approximation of  $f_0$  on  $[a, b]$  as

$$f_0(x) \approx \sum_{n=0}^N b_n T_n(g(x)), \quad (3.10)$$

which is a finite sum of Chebyshev polynomials.

As shown in (1.7), the IBVP's initial datum  $f(x)$  is defined on  $[0, 1]$ . Thus,

we define  $g(x) : [0, 1] \rightarrow [-1, 1]$  by  $g(x) = 2x - 1$ . With the change of variable  $t = g(x) = 2x - 1$ , we have  $x = g^{-1}(t) = \frac{t+1}{2}$ . Let  $h(t) := f \circ g^{-1}(t)$ . Then  $h(t)$  is defined on  $[-1, 1]$ . Let  $c := \frac{\alpha^{l-1}\lambda}{2}$ . Then

$$\begin{aligned} \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x) dx &= \int_0^1 e^{-i2c\frac{t+1}{2}} f\left(\frac{t+1}{2}\right) d\left(\frac{t+1}{2}\right) \\ &= \int_{-1}^1 e^{-ict} e^{-ic} f\left(\frac{t+1}{2}\right) \frac{1}{2} dt \\ &= \frac{1}{2e^{ic}} \int_{-1}^1 e^{-ict} f\left(\frac{t+1}{2}\right) dt \\ &= \frac{1}{2e^{ic}} \int_{-1}^1 e^{-ict} h(t) dt, \end{aligned}$$

Thus, it suffices to find

$$\int_{-1}^1 e^{-ict} h(t) dt.$$

Suppose that on  $[-1, 1]$ , the Chebyshev approximation of  $h(t)$  is given by

$$h(t) \approx \sum_{n=0}^N b_n T_n(t).$$

Note that since  $t = g(x)$ , we have  $f(x) = f(g^{-1}(g(x))) = h(g(x)) = h(t)$ , and the above is equivalent to

$$f(x) \approx \sum_{n=0}^N b_n T_n(g(x)),$$

which is the same form as equation (3.10). Thus, the Chebyshev coefficients of  $h(t)$  are exactly the Chebyshev coefficients of  $f(x)$  obtained on  $[-1, 1]$  after scaling.

Since  $t \in [-1, 1]$ , the Chebyshev polynomials  $T_n(t)$  is given by  $\cos(n \arccos(t))$ , and thus

$$h(t) \approx \sum_{n=0}^N b_n \cos(n \arccos(t)).$$

Therefore,

$$\int_{-1}^1 e^{-ict} h(t) dt = \int_{-1}^1 e^{-ict} \sum_{n=0}^N b_n \cos(n \arccos(t)) dt$$

$$= \sum_{n=0}^N b_n \int_{-1}^1 e^{-ict} \cos(n \arccos(t)) dt.$$

With the change of variable  $t = \cos \theta$ , we have

$$\begin{aligned} \int_{-1}^1 e^{-ict} \cos(n \arccos(t)) dt &= \int_{-1}^1 e^{-ic \cos \theta} \cos(n \arccos(\cos \theta)) d \cos \theta \\ &= \int_{\pi}^0 e^{-ic \cos \theta} \cos(n\theta) (-\sin \theta) d\theta \\ &= \int_0^\pi e^{-ic \cos \theta} \cos(n\theta) \sin \theta d\theta. \end{aligned}$$

The remaining section presents material from [18] with proofs expanded.

Define

$$\hat{T}_n(c) := \int_0^\pi e^{-ic \cos \theta} \cos(n\theta) \sin \theta d\theta.$$

Then

$$\begin{aligned} \hat{T}_n(0) &= \int_0^\pi \cos(n\theta) \sin \theta d\theta = \begin{cases} 2 & \text{if } n = 0 \\ [\frac{1}{2} \sin^2 \theta + \frac{0}{2} \cos^2 \theta]_0^\pi & \text{if } n = 1 \\ \left[ \frac{n}{n^2-1} \sin \theta \sin(n\theta) + \frac{1}{n^2-1} \cos \theta \cos(n\theta) \right]_0^\pi & \text{if } n \geq 2 \end{cases} \\ &= \begin{cases} 2 & \text{if } n = 0 \\ \frac{1}{2} \sin^2(\pi) - \frac{1}{2} \sin^2(0) & \text{if } n = 1 \\ \frac{n}{n^2-1} \sin(\pi) \sin(n\pi) + \frac{1}{n^2-1} \cos(\pi) \cos(n\pi) \\ \quad - \frac{n}{n^2-1} \sin(0) \sin(0) - \frac{1}{n^2-1} \cos(0) \cos(0) & \text{if } n \geq 2 \end{cases} \\ &= \begin{cases} 2 & \text{if } n = 0, \\ 0 & \text{if } n = 1, \\ \frac{(-1)^{n+1}-1}{n^2-1} & \text{if } n \geq 2. \end{cases} \end{aligned}$$

Note that the formula for  $n \geq 2$  agrees with the case for  $n = 0$ , but for clarity,

we write them separately. For  $c \in \mathbb{C} \setminus \{0\}$ , using integration by parts, we have

$$\begin{aligned}
\hat{T}_n(c) &= \int_0^\pi e^{-ic\cos\theta} \cos(n\theta) \sin\theta d\theta \\
&= \left[ \cos(n\theta) \frac{1}{ic} e^{-ic\cos\theta} \right]_0^\pi - \int_0^\pi \frac{1}{ic} e^{-ic\cos\theta} n(-\sin(n\theta)) d\theta \\
&= \left[ \cos(n\pi) \frac{1}{ic} e^{-ic\cos(\pi)} - \cos(0) \frac{1}{ic} e^{-ic\cos 0} \right] + \frac{n}{ic} \int_0^\pi e^{-ic\cos\theta} \sin(n\theta) d\theta \\
&= \frac{1}{ic} \left( n \int_0^\pi e^{-ic\cos\theta} \sin(n\theta) d\theta + (-1)^n e^{ic} - e^{-ic} \right).
\end{aligned}$$

Let  $z \in \mathbb{C} \setminus \{0\}$ , define

$$K_n(z) := \int_0^\pi e^{z\cos\theta} \sin(n\theta) d\theta.$$

Then for  $c \in \mathbb{C} \setminus \{0\}$ ,

$$\begin{aligned}
\hat{T}_n(c) &= \frac{1}{ic} \left( n \int_0^\pi e^{-ic\cos\theta} \sin(n\theta) d\theta + (-1)^n e^{ic} - e^{-ic} \right) \\
&= \frac{1}{ic} (n K_n(-ic) + (-1)^n e^{ic} - e^{-ic}).
\end{aligned}$$

We note that

$$\begin{aligned}
K_0(z) &= \int_0^\pi e^{z\cos\theta} \sin 0 d\theta = 0 \\
K_1(z) &= \int_0^\pi e^{z\cos\theta} \sin\theta d\theta \\
&= \left[ -\frac{1}{z} e^{z\cos\theta} \right]_0^\pi \\
&= -\frac{1}{z} e^{z\cos(\pi)} + \frac{1}{z} e^{z\cos 0} \\
&= \frac{e^z - e^{-z}}{z}.
\end{aligned}$$

To compute  $K_n(z)$  for  $n \geq 2$ , we first note that

$$\begin{aligned}
\sin((n+1)\theta) - \sin((n-1)\theta) &= \frac{e^{i(n+1)\theta} - e^{-i(n+1)\theta}}{2i} - \frac{e^{i(n-1)\theta} - e^{-i(n-1)\theta}}{2i} \\
&= \frac{e^{in\theta} e^{i\theta} - e^{in\theta} e^{-\theta} - e^{-in\theta} e^{-i\theta} + e^{-in\theta} e^{i\theta}}{2i}
\end{aligned}$$

$$\begin{aligned}
&= \frac{(e^{i\theta} - e^{-i\theta})(e^{in\theta} + e^{-in\theta})}{2i} \\
&= 2 \left( \frac{e^{i\theta} - e^{-i\theta}}{2i} \right) \left( \frac{e^{in\theta} + e^{-in\theta}}{2} \right) \\
&= 2 \sin \theta \cos(n\theta).
\end{aligned}$$

So

$$\sin((n+1)\theta) = \sin((n-1)\theta) + 2 \sin \theta \cos(n\theta).$$

Thus,

$$\begin{aligned}
K_{n+1}(z) &= \int_0^\pi e^{z \cos \theta} \sin((n+1)\theta) d\theta \\
&= \int_0^\pi e^{z \cos \theta} (\sin((n-1)\theta) + 2 \sin \theta \cos(n\theta)) d\theta \\
&= \int_0^\pi e^{z \cos \theta} \sin((n-1)\theta) d\theta + 2 \int_0^{\frac{\pi}{2}} e^{z \cos \theta} \sin \theta \cos(n\theta) d\theta \\
&= K_{n-1}(z) + 2 \int_0^\pi \cos(n\theta) \sin \theta e^{z \cos \theta} d\theta.
\end{aligned}$$

Using integration by parts,

$$\begin{aligned}
\int_0^\pi \cos(n\theta) \sin \theta e^{z \cos \theta} d\theta &= \left[ -\frac{1}{z} e^{z \cos \theta} \cos(n\theta) \right]_0^\pi - \int_0^\pi \frac{1}{z} e^{z \cos \theta} n \sin(n\theta) d\theta \\
&= -\frac{1}{z} e^{z \cos(\pi)} \cos(n\pi) + \frac{1}{z} e^{z \cos 0} \cos 0 - \frac{n}{z} \int_0^\pi e^{z \cos \theta} \sin(n\theta) d\theta \\
&= \frac{e^z}{z} + (-1)^{n+1} \frac{e^{-z}}{z} - \frac{n}{z} K_n(z).
\end{aligned}$$

Thus,  $K_n(z)$  satisfies the recurrence relation

$$\begin{aligned}
K_0(z) &= 0 \\
K_1(z) &= \frac{e^z - e^{-z}}{z} \\
K_n(z) &= K_{n-2}(z) + 2 \left( \frac{e^z}{z} + (-1)^n \frac{e^{-z}}{z} - \frac{n-1}{z} K_{n-1}(z) \right), \quad n \geq 2.
\end{aligned}$$

Further solving the recurrence relation [18] gives

$$\hat{T}_n(c) = \int_0^\pi e^{-ic\cos\theta} \cos(n\theta) \sin\theta d\theta = \begin{cases} 2 & \text{if } n = 0, \\ 0 & \text{if } n = 1, \\ \frac{(-1)^{n+1}-1}{n^2-1} & \text{if } n \geq 2, \end{cases} \quad \text{if } c = 0,$$

$$= \sum_{m=1}^{n+1} \alpha(m, n) \left[ \frac{e^{i\lambda}}{(i\lambda)^m} + (-1)^{m+n} \frac{e^{-i\lambda}}{(i\lambda)^m} \right] \quad \text{if } c \neq 0,$$

where

$$\alpha(m, n) = \begin{cases} (-1)^n & \text{if } m = 1, \\ (-1)^{n+1}n^2 & \text{if } m = 2, \\ (-1)^{n+m-1}2^{m-2}n \sum_{k=1}^{n-m+2} \binom{m+k-3}{k-1} \prod_{j=k}^{m+k-3} (n-j) & \text{if } m = 3, 4, \dots, n+1. \end{cases}$$

Thus, in summary,

$$\begin{aligned} \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x) dx &= \frac{1}{2e^{ic}} \int_{-1}^1 e^{-ict} h(t) dt \\ &\approx \frac{1}{2e^{ic}} \sum_{n=0}^N b_n \int_{-1}^1 e^{-ict} \cos(n \arccos(t)) dt \\ &= \frac{1}{2e^{ic}} \sum_{n=0}^N b_n \hat{T}_n(t), \end{aligned}$$

with  $\hat{T}_n(t)$  given above.

## 4 Implementation

### 4.1 Overview

The above algorithm is implemented in Julia 0.6.4. Library dependencies are SymPy [19], sympy [20], Distributions [21], ApproxFun [22], Roots [23], QuadGK [24], Plots [25], Gadfly [26], and PyPlot [27]. Code, unit tests, and documentation with examples can be found in [16], with possibly minor differences in

notation from section 3.2.

As mentioned near the end of section 1.1, the implementation aims at providing computer aid alongside human judgment in the process of using the Fokas method to solve IBVPs.

## 4.2 Sample usage

We illustrate the utilities of the implementation using two cases of sample usage below. More details can be found in the documentation [16].

### 4.2.1 Solving IBVP

In this section, we show how the implementation provides computer aid in the process of applying the Fokas method.

Consider the heat equation IBVP

$$q_t - q_{xx} = 0 \quad \forall (x, t) \in (0, 1) \times (0, T) \quad (4.1a)$$

$$q(x, 0) = f(x) \quad \forall x \in [0, 1] \quad (4.1b)$$

$$q(0, t) - q(1, t) = 0 \quad \forall t \in [0, T] \quad (4.1c)$$

$$q_x(0, t) - q_x(1, t) = 0 \quad \forall t \in [0, T], \quad (4.1d)$$

where  $f(x)$  also satisfies equations (4.1c)–(4.1d).

Rewriting the IBVP in the form (1.7), we have  $n = 2$ ,  $a = 1$ ,

$$S = (-i)^2 \frac{d^2}{dx^2} = -\frac{d^2}{dx^2},$$

$$B_1\phi = 1 \cdot \varphi(0) + (-1) \cdot \varphi(1) + 0 \cdot \varphi^{(1)}(0) + 0 \cdot \varphi^{(1)}(1)$$

$$B_2\phi = 0 \cdot \varphi(0) + 0 \cdot \varphi(1) + 1 \cdot \varphi^{(1)}(0) + (-1) \cdot \varphi^{(1)}(1),$$

and

$$\Phi = \{\phi \in C^\infty[0, 1] : B_j\phi = 0 \forall j \in \{1, 2\}\}.$$

Using the notation in section 3.1, the two matrices associated with the vector boundary form corresponding to the above boundary conditions are

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad N = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

We choose  $f(x) = \sin(2\pi x)$  to be the initial datum. Then  $f(x)$  satisfies

$$Uf = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} f(0) \\ f'(0) \\ f(1) \\ f'(1) \end{bmatrix} = 0.$$

By (2.15),  $f \in \Phi$ , as required.

The adjoint vector boundary form found by the algorithm outlined in section 3.1 is characterized by the matrices

$$b^* = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \beta^* = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \quad (4.2)$$

From (4.2), we compute  $\Delta(\lambda)$  in (3.4) to be

$$\Delta(\lambda) = 8i\lambda \sin^2\left(\frac{\lambda}{2}\right), \quad (4.3)$$

and  $F_\lambda^+(f)$ ,  $F_\lambda^-(f)$  in (3.5) to be

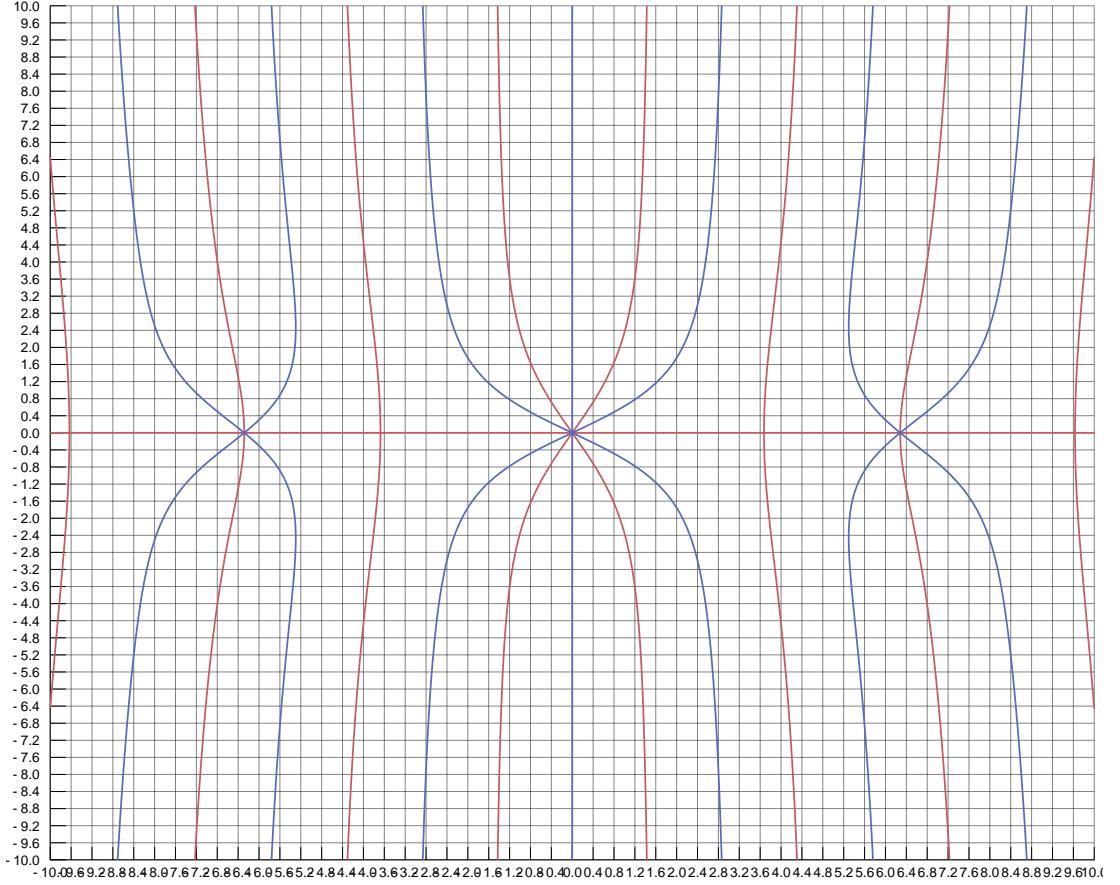
$$F_\lambda^+(f) = \frac{\hat{f}(\lambda)e^{i\lambda}}{2\pi(e^{i\lambda} - 1)} \quad (4.4a)$$

$$F_\lambda^-(f) = \frac{\hat{f}(\lambda)}{2\pi(e^{i\lambda} - 1)}, \quad (4.4b)$$

where  $\hat{f}(\lambda) = \int_0^1 e^{i\lambda x} f(x) dx$  is approximated using the method in section 3.2.2.

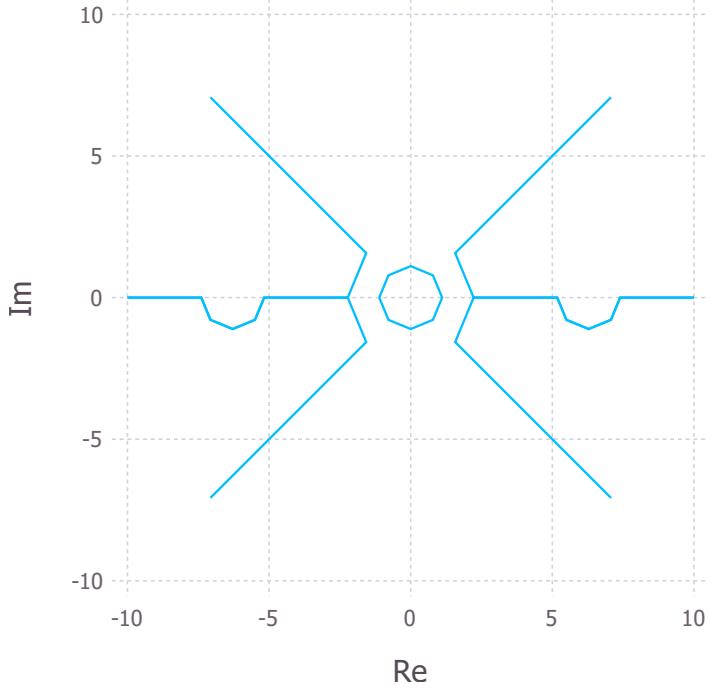
From the level curves corresponding to (3.9) in Figure 10, we see that the zeros of  $\Delta(\lambda)$  are  $-2\pi, 0, 2\pi$  (with the modulus of complex infinity  $d$  set as 10).

With this information, we find the contours  $\Gamma$  to be as in Figure 11.



**Figure 10:** Level curves corresponding to (3.9) with  $\Delta(\lambda)$  defined in (4.3).

As shown in equation (3.8), to compute  $q(x, t)$ , we need to evaluate  $e^{i\lambda x} e^{-a\lambda^n t} F_\lambda^\pm(f)$  over the contours  $\Gamma_a^\pm$  and  $\Gamma_0^\pm$ , respectively; call this the outer integral. We note that the inner integral  $\int_0^1 e^{i\alpha^{l-1}\lambda x} f(x) dx$  in (3.5) depends on both  $\lambda$  and  $f$ . One consequence of this is that if  $F_\lambda^+$ ,  $F_\lambda^-$  are to be implemented as functions without symbolic expressions, they each need to take both  $\lambda$  and  $f$  as arguments. But doing so results in unnecessary computational cost: The outer integral is over  $\lambda$  only, meaning that, when evaluating it,  $\lambda$  changes while  $f$  does not. But once  $\lambda$  changes, the inner integral also needs to be re-calculated, and because it depends on  $f$  and  $\lambda$ , the procedure to approximate the integral described in section 3.2.2 would need to be repeated. As a result, when evaluating  $q(x, t)$ , the Chebyshev approximation of  $f$  is re-calculated at each point in the  $\Gamma$  contour even though  $f$  does not change throughout the process. Current implementation of the Fokas method in this project avoids such unnecessary computational cost



**Figure 11:** The  $\Gamma$  contour.

by computing the explicit symbolic formulas of  $F_\lambda^+$ ,  $F_\lambda^-$  that depend on  $\lambda$  only; these symbolic formulas are derived by replacing  $\hat{f}(\lambda)$  in (4.4) with the symbolic expression of the approximation of  $\int_0^1 e^{i\lambda x} f(x) dx$  obtained in section 3.2.2. In this way, components of the outer integral that depend on  $f$  are evaluated only once.

In practice, we would convert the symbolic formulas into function objects, because in Julia, function objects are evaluated much faster than symbolic objects. More recent versions of Julia have parsers that perform the conversion. In Julia 0.6.4 in which the current implementation is based (for reasons explained in the last paragraph in section 1.1), however, no such parser exists, and conversion needs to be done manually, namely to create the corresponding function objects by typing in the symbolic formulas in the function definition. Having done so, current implementation of the Fokas method in this project allows evaluating  $q(x, t)$  at a given  $(x_0, t_0) \in (0, 1) \times (0, T)$  within 20 seconds with the modulus of complex infinity set as 10, and within 1 minute with the modulus of complex

infinity set as 50 (which is a more reasonable value regarding numerical accuracy, as explained in section 3.2.1).

As a side note, the solution  $q(x, t)$  computed above using equation (3.8) corresponds to the separated solution of the heat equation [9] in the following sense. Using Cauchy's theorem and Jordan's lemma from complex analysis, it can be shown that integration over the  $\Gamma$  contour in Figure 11 reduces to integration over closed contours around the poles of  $F_\lambda^+$  and  $F_\lambda^-$ , i.e., the zeros of  $\Delta(\lambda)$  in Figure 10. That is,  $q(x, t)$  reduces to a sum of contour integrals around zeros of  $\Delta(\lambda)$ , and using the residue theorem, these contour integrals can be shown to correspond to the eigenfunctions of Sturm-Liouville BVP related to the heat equation IBVP. Since  $\Delta(\lambda)$  have infinitely many zeros on the real line, as the estimate of the modulus of complex infinity  $d$  tends to  $\infty$ ,  $\Gamma$  would effectively consist of infinitely many closed contours centered on the real line (much like the blue circles on  $\text{Re}(\lambda)$  in Figure 3, but extending to  $\infty$  in both directions). As a result,  $q(x, t)$  would become an infinite series of contour integrals, which is none other than the separated solution of the heat equation of the form

$$q(x, t) = \sum_{n=1}^{\infty} A_n \phi_n(x) e^{-\lambda_n Kt},$$

where  $A_n$  are Fourier coefficients of the initial datum  $f(x)$ ,  $\lambda_n$  the eigenvalues of the Sturm-Liouville problem, and  $\phi_n(x)$  the corresponding eigenfunctions. Such correspondence to the separated solution is no surprise since the Fokas transform method originated from a deeper form of separation of variables [11][28]. This is also reflected in the fact that methods based on transform pairs essentially separate the spatial and temporal components of the IBVP.

#### 4.2.2 Symbolic computation

In this section, we illustrate how the symbolic computation feature of the implementation can be useful.

Applying the Fokas method to IBVPs and beyond depends on two technical

lemmas, one concerning the positions of the poles of  $F_\lambda^+$ ,  $F_\lambda^-$  (i.e., zeros of  $\Delta(\lambda)$ ), the other concerning the asymptotic decay of  $F_\lambda^+$ ,  $F_\lambda^-$ . These lemmas are first described in [5] for IBVPs. Their precise roles are more clearly articulated in [13], which also describes the extension of the Fokas method to a broader class of problems. These lemmas may assume variations for different types of problems such as those described in the beginning of section 5, but some version of them are always required when applying the Fokas method. Because of the way these lemmas make statements about  $F_\lambda^+$  and  $F_\lambda^-$ , in proving them or their variations, it would be helpful to have the symbolic formulas for  $F_\lambda^+$ ,  $F_\lambda^-$ .

For the IBVPs (1.2) and (1.3), the formulas for  $F_\lambda^+$  and  $F_\lambda^-$  have been computed by hand in [1, p. 1-4]. Symbolic formulas for the same objects have also been computed by the implementation of the Fokas method following the steps outlined in section 4.2.1, with  $S$ ,  $a$  as described in section 1.2, and

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad N = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -2 & 0 \end{bmatrix}$$

in (1.2), and

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad N = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

in (1.3). The symbolic formulas found by the implementation, though not simplified to forms pleasant to the human eye due to limitations of the Julia libraries used, have been verified to be equivalent to those derived by hand in [1] by comparing the coefficients of  $\hat{f}(\alpha^{l-1}\lambda) = \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x) dx$  in (3.5); the coefficients (which depend on  $\lambda$ ) are evaluated at 100 random values of  $\lambda \in \mathbb{C}$  avoiding any zeros of  $\Delta(\lambda)$ .

## 5 Discussion

### 5.1 Extension to other problems

This project concerns applying the Fokas method to solving IBVPs. With appropriate modifications, the Fokas method can be extended to other types of problems. These include multipoint problems with conditions that relate the solution’s values at different spatial locations [29]; nonlocal problems with conditions that specify a weighted average of the solution over the spatial interval [13]; interface problems where solution in one domain specifies boundary conditions for neighbouring domains [30], involving the heat equation [31][32], the linear Schrödinger equation [33], and the linear KdV equation [34]; and PDEs of fractional orders [12], as briefly mentioned before in section 1.1. Correspondingly, the implementation of the Fokas method in this project may be extended to problems beyond IBVPs. While modifications need to be made, the design and usefulness of the implementation’s functionalities such as those described in section 4.2.2 should remain generalizable to applications of the Fokas method in other settings.

### 5.2 Possible improvements

Currently, this project’s implementation of the Fokas method [16] supplies utilities such as contour visualization, symbolic computation, and numeric evaluation that significantly reduce the time and effort in applying the Fokas method compared to working by hand. Possible directions for further improvement involve better symbolic computation, wider coverage of automated functionalities, and faster computation speed.

#### 5.2.1 Symbolic computation

As mentioned in section 4.2.2, the symbolic formulas derived by the implementation of the Fokas method are not always simplified to a satisfying form due to

limitations of the Julia libraries used. This issue may be improved by migrating the implementation to later releases of Julia with better library support for symbolic computation.

### 5.2.2 Automatically approximating poles of the integrands

As mentioned in section 3.2.1, to approximate the poles of  $F_\lambda^+$  and  $F_\lambda^-$ , i.e., zeros of  $\Delta(\lambda)$ , in the region  $D(0, d)$  (where  $d$  is the number representing the modulus of complex infinity), human judgement is still needed to visually locate the zeros as the intersections of the level curves of  $\text{Re}(\Delta)(\lambda) = 0$  and  $\text{Im}(\Delta)(\lambda) = 0$ . This procedure may be automated based on a numerical method to find zeros of an analytic function in a given region using Cauchy's argument principle [35].

By Cauchy's argument principle, if  $C$  is a closed curve,  $R$  is the interior of  $C$ , and  $f(z)$  is analytic in and on  $C$ , then the number of zeros of  $f(z)$  in  $R$  is given by

$$N_z := \frac{1}{2\pi i} \int_C \frac{f'(z)}{f(z)} dz.$$

As mentioned in section 3.2,  $\Delta(\lambda)$  is an exponential polynomial, and so it is analytic. By the above result, the number of zeros of  $\Delta(\lambda)$  in  $D(0, d)$  is given by

$$\frac{1}{2\pi i} \int_{C(0,d)} \frac{\Delta'(\lambda)}{\Delta(\lambda)} d\lambda.$$

Thus, to approximate the zeros of  $\Delta(\lambda)$ , we may divide the region  $D(0, d)$  into sub-regions, find the number of zeros in each region, discard those sub-regions that do not contain any zeros, and proceed iteratively, until we obtain  $N_z$  sub-regions whose radii are within a certain pre-determined tolerance level, each trapping exactly one zero of  $\Delta(\lambda)$ . Alternatively, one may strictly follow the method in [35], which starts by dividing the region into sub-regions until each sub-region contains no more than a pre-determined number  $n_z$  of zeros. Then, zeros in any given sub-region (if any) are found by first constructing an up-to- $n_z$ -degree polynomial with the same zeros as  $\Delta(\lambda)$  in that sub-region, and then finding

the zeros of this constructed polynomial by exploiting the rich collection of root-finding algorithms for polynomials.

### 5.2.3 Computation speed

Although evaluation of the solution  $q(x, t)$  at a single point can now be done in less than a minute with a reasonable estimate of the modulus of complex infinity (as explained at the end of section 4.2.1), analyzing the solution in depth usually involves visualizing it, which poses a much higher demand on the speed of evaluating the solution. To plot in reasonable time, evaluating the solution at a single point should ideally take no more than a fraction of a second. The implementation as it is now evaluates the solution  $q(x, t)$  as follows. Given  $(x_0, t_0) \in (0, 1) \times (0, T)$ , the integrand  $e^{i\lambda x_0} e^{-\alpha\lambda^n t_0} F_\lambda^\pm(f)$  in equation (3.8) is defined as a function of  $\lambda$ , which is then integrated over the four contours  $\Gamma_a^+$ ,  $\Gamma_a^-$ ,  $\Gamma_0^+$ , and  $\Gamma_0^-$  as in equation (3.8). This means that when attempting to plot  $q(x, t)$  over a domain  $D$ , at each point  $(x_0, t_0) \in D$ , a new function corresponding to the integrand is defined, compiled, and evaluated. The problem here is that in Julia, compilation time could be several orders of magnitude higher than evaluation time, and the more so for complicated functions. For instance, the compilation time issue would be prominent if the Chebyshev approximation of  $f(x)$  used in section 3.2.2 has many coefficients; in this case the symbolic formulas for  $F_\lambda^+$ ,  $F_\lambda^-$  would be complicated, meaning that they would take much time to compile, making it impractical to plot the solution  $q(x, t)$  in reasonable time.

Another problem that contributes to the speed issue is that, if the circular contours around zeros of  $\Delta(\lambda)$  have very small diameter  $\epsilon$  (given our choice of  $\epsilon$  in section 3.2.1, this would be the case if two distinct zeros are very close to each other, or there is a zero that is close to the boundary of some sector), the integration contours would be very close to the zeros which they are supposed to avoid. As discussed in section 3.2.1, zeros of  $\Delta(\lambda)$  are poles of the integrands  $F_\lambda^+$  and  $F_\lambda^-$ , and so the integrands easily become numerically unstable near them,

yielding very large numbers when evaluated. As a result, the integration itself becomes slow.

Currently, the implementation of the Fokas method in this project uses the native Julia integrator `quadgk`. One possible solution to the speed issue is to develop a custom integrator tailored to evaluating the contour integrals in equation (3.8) by exploiting the mathematical properties of the integrands  $F_\lambda^+$ ,  $F_\lambda^-$  and the contours  $\Gamma$ . Analysis of the behaviour of  $F_\lambda^+$  and  $F_\lambda^-$  over  $\Gamma$  could help identifying the components of the integrands that contribute significantly to the integral at different parts of the contour  $\Gamma$ . In this way, we could reduce the time it takes to evaluate  $q(x, t)$  by dynamically “trimming” the integrands as we move along the contours.

## References

- [1] D. A. Smith and A. S. Fokas, “Evolution PDEs and augmented eigenfunctions. Finite interval,” *Advances in Differential Equations* **21** no. 7/8, (Mar, 2016) 735–766, [arXiv:1303.2205](https://arxiv.org/abs/1303.2205).  
<http://www.ems-ph.org/doi/10.4171/JST/123>  
<http://arxiv.org/abs/1303.2205>.
- [2] A. S. Fokas, “On the integrability of linear and nonlinear partial differential equations,” *Journal of Mathematical Physics* **41** no. 6, (2000) 4188–4237.
- [3] A. S. Fokas and B. Pelloni, “Two-point boundary value problems for linear evolution equations,” *Mathematical Proceedings of the Cambridge Philosophical Society* **131** (2001) 521–543.
- [4] A. S. Fokas, *A Unified Approach to Boundary Value Problems*. Society for Industrial and Applied Mathematics, Jan, 2008.  
<http://pubs.siam.org/doi/book/10.1137/1.9780898717068>.
- [5] D. A. Smith, “Well-posedness and conditioning of 3rd and higher order two-point initial-boundary value problems,” [arXiv:1212.5466v1](https://arxiv.org/abs/1212.5466v1).
- [6] B. Deconinck, T. Trogdon, and V. Vasan, “The Method of Fokas for Solving Linear Partial Differential Equations,” *SIAM Review* **56** no. 1, (Jan, 2014) 159–186. <http://pubs.siam.org/doi/10.1137/110821871>.
- [7] A. S. Fokas and B. Pelloni, eds., *Unified Transform for Boundary Value Problems*. Society for Industrial and Applied Mathematics, Philadelphia, 2014. <http://pubs.siam.org/doi/book/10.1137/1.9781611973822>.
- [8] “The Julia Language.” <https://julialang.org/>.
- [9] M. A. Pinsky, *Partial Differential Equations and Boundary-Value Problems with Applications*. American Mathematical Society, third ed., 1991.  
[papers://44fb6834-50a5-40c9-9a74-c1e60ac88924/Paper/p752](https://papers.ssrn.com/sol3/papers/f44fb6834-50a5-40c9-9a74-c1e60ac88924/Paper/p752.pdf).

- [10] D. J. Korteweg and G. de Vries, “XLI. On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **39** no. 240, (May, 1895) 422–443.  
<https://www.tandfonline.com/doi/full/10.1080/14786449508620739>.
- [11] A. S. Fokas, “A unified transform method for solving linear and certain nonlinear PDEs,” *Proceedings: Mathematical, Physical and Engineering Sciences* **453** no. 1962, (1997) 1411–1443.
- [12] A. Fernandez, D. Baleanu, and A. S. Fokas, “Solving PDEs of fractional order using the unified transform method,” *Applied Mathematics and Computation* **339** (2018) 738–749.  
<https://doi.org/10.1016/j.amc.2018.07.061>.
- [13] P. D. Miller and D. A. Smith, “The diffusion equation with nonlocal data,” *Journal of Mathematical Analysis and Applications* **466** no. 2, (2018) 1119–1143, arXiv:1708.00972. <http://arxiv.org/abs/1708.00972>.
- [14] E. A. Coddington and N. Levinson, *Theory of Ordinary Differential Equations*. McGraw-Hill Publishing, New York, 1977.
- [15] S. Axler, *Linear Algebra Done Right*. Undergraduate Texts in Mathematics. Springer New York, New York, NY, third ed., 1997.  
arXiv:arXiv:1011.1669v3.  
<http://link.springer.com/10.1007/b97662>.
- [16] L. Xiao, “Implementation of the Fokas method.”  
<https://gitlab.com/linfanxiaolinda/capstone>.
- [17] R. E. Langer, “On the zeros of exponential sums and integrals,” *Bulletin of the American Mathematical Society* **37** no. 4, (Apr, 1931) 213–240. <http://www.ams.org/journal-getitem?pii=S0002-9904-1931-05133-8>.

- [18] A. S. Fokas and S. A. Smitheman, “The Fourier Transforms of the Chebyshev and Legendre Polynomials,” [arXiv:1211.4943](https://arxiv.org/abs/1211.4943).  
<http://arxiv.org/abs/1211.4943>.
- [19] “SymPy.jl.” <https://github.com/JuliaPy/SymPy.jl>.
- [20] “sympy.py.” <https://www.sympy.org/en/index.html>.
- [21] “Distributions.jl.” <https://github.com/JuliaStats/Distributions.jl>.
- [22] “ApproxFun.jl.”  
<https://github.com/JuliaApproximation/ApproxFun.jl>.
- [23] “Roots.jl.” <https://github.com/JuliaMath/Roots.jl>.
- [24] “QuadGK.jl.” <https://github.com/JuliaMath/QuadGK.jl>.
- [25] “Plots.jl.” <https://github.com/JuliaPlots/Plots.jl>.
- [26] “Gadfly.jl.” <https://github.com/GiovineItalia/Gadfly.jl/blob/master/docs/src/index.md>.
- [27] “PyPlot.jl.” <https://github.com/JuliaPy/PyPlot.jl>.
- [28] A. S. Fokas, “Lax pairs: A novel type of separability,” *Inverse Problems* **25** no. 12, (2009) 123007.
- [29] B. Pelloni and D. A. Smith, “Nonlocal and multipoint boundary value problems for linear evolution equations,” *Studies in Applied Mathematics* **141** no. 1, (2018) 46–88, [arXiv:1511.07244](https://arxiv.org/abs/1511.07244).  
<http://arxiv.org/abs/1511.07244>.
- [30] N. E. Sheils, *Interface Problems usnig the Fokas Method*. PhD thesis, University of Washington, 2015.
- [31] N. E. Sheils and B. Deconinck, “Heat conduction on the ring: Interface problems with periodic boundary conditions,” *Applied Mathematics Letters* **37** (2014) 107–111. <http://dx.doi.org/10.1016/j.aml.2014.06.006>.

- [32] N. E. Sheils and D. A. Smith, “Heat equation on a network using the Fokas method,” *Journal of Physics A: Mathematical and Theoretical* **48** no. 33, (2015) 335001.
- [33] N. E. Sheils and B. Deconinck, “Interface Problems for Dispersive equations,” *Studies in Applied Mathematics* **134** no. 3, (May, 2014) 253–275, arXiv:1405.3307. <http://arxiv.org/abs/1405.3307> <http://dx.doi.org/10.1111/sapm.12070>.
- [34] B. Deconinck, N. E. Sheils, and D. A. Smith, “The Linear KdV Equation with an Interface,” *Communications in Mathematical Physics* **347** no. 2, (2016) 489–509, arXiv:arXiv:1508.03596v1.
- [35] L. M. Delves and J. N. Lyness, “A Numerical Method for Locating the Zeros of an Analytic Function,” *Mathematics of Computation* **21** no. 100, (1967) 543–560.