Algorithmic Solution of High Order Partial Differential Equations in Julia via the Fokas Transform Method

Linfan Xiao

Advisor: Prof. Dave Smith
Mathematical, Computational, and Statistical Sciences
Yale-NUS College

April 9, 2019

Algorithmic Solution of High Order Partial Differential Equations in Julia via the Fokas Transform Method

Algorithmic Solution of
High Order Partial Differential Equations*
in Julia
via the Fokas Transform Method

* Equations of the form

$$f\left(x_1, \ldots, x_n, u, u_{x_1}, \ldots, u_{x_n}, u_{x_1 x_1}, \ldots, u_{x_1 x_n}, \ldots\right) = 0,$$

which relate an unknown function $u(x_1, \ldots, x_n)$ to its partial derivatives $u_{x_i \cdots x_j} := \frac{\partial}{\partial x_i} \cdots \frac{\partial}{\partial x_j} u$.

Algorithmic Solution of High Order Partial Differential Equations in Julia via the Fokas Transform Method*

* A method to solve a certain class of PDE problems algorithmically[1].

[1] Athanassios S. Fokas. *A Unified Approach to Boundary Value Problems*. Society for Industrial and Applied Mathematics, 2008. ISBN: 978-0-89871-651-1. DOI: 10.1137/1.9780898717068. URL: http://epubs.siam.org/doi/book/10.1137/1.9780898717068.

Algorithmic Solution of
High Order Partial Differential Equations
in Julia*
via the Fokas Transform Method

* A free, open-source, high-performance language for numerical computing[2].

[2] *The Julia Language*. https://julialang.org/. URL: `https://julialang.org/` (visited on 03/19/2019).

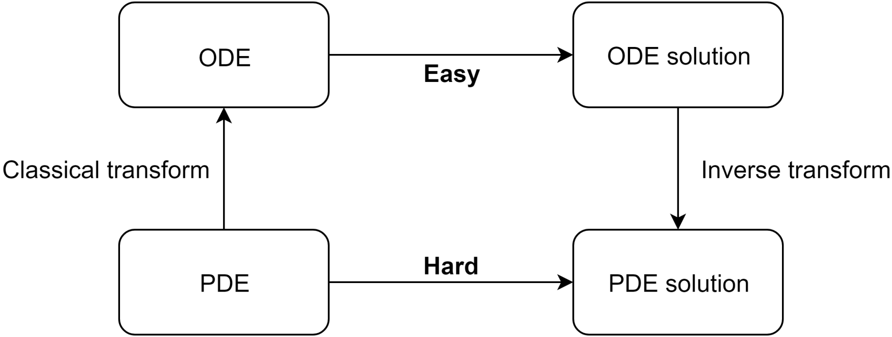
Table of Contents

Motivation

- PDEs can model various physical phenomena. E.g.,
  - heat equation $u_t = K u_{xx}$
  - wave equation $u_{tt} = c^2(u_{xx} + u_{yy})$
  - (linear) Schrödinger equation $ihw_t + \frac{h}{2m} w_{xx} = 0$
- Problems involving PDEs are usually formulated as initial-boundary value problems (IBVPs), consisting of
  - a PDE in temporal and spatial variables defined over a domain,
  - boundary conditions, and
  - initial condition.

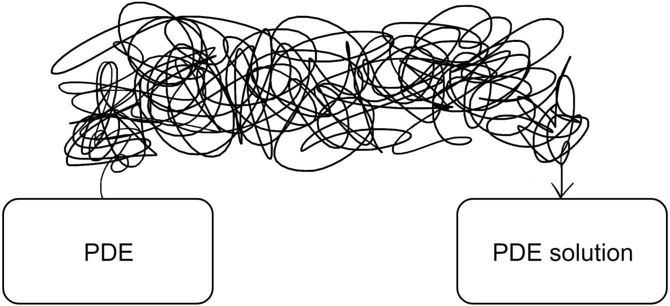
Motivation

- Some IBVPs can be solved using classical transform pairs, e.g., the Fourier transform.



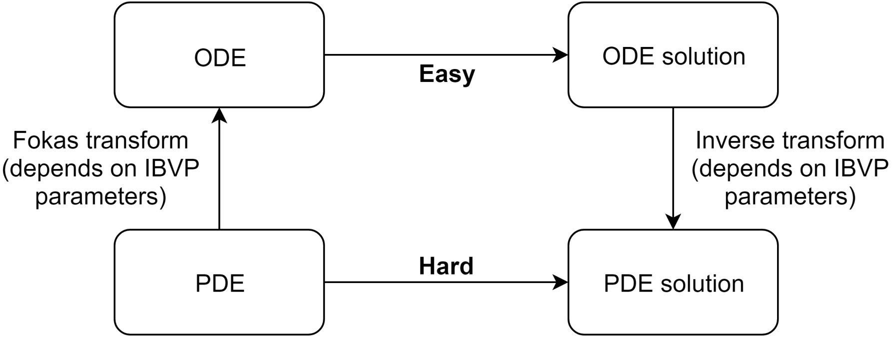
Motivation

- For more complicated IBVPs, no such classical transform pairs exist.
  - Resort to ad-hoc methods that are specific to the given IBVP.



Motivation

- The Fokas method extends the idea of classical transform pair to a class of complicated IBVPs by constructing transform pairs depending on the problems' parameters.
  - Even better, this class of IBVPs can have arbitrary spatial order.



Motivation

- The Fokas method advances the understanding of high order PDEs by providing an algorithmic procedure to solve an entire class of IBVPs of arbitrary spatial order.
- However, it is still laborious to use the Fokas method by hand.

Motivation

- The Fokas method advances the understanding of high order PDEs by providing an algorithmic procedure to solve an entire class of IBVPs of arbitrary spatial order.
- However, it is still laborious to use the Fokas method by hand.

Thus, capstone:

> Implement the Fokas method[1] as a software library[2] that supplies various computer aid[3] in the process of solving IBVPs[4].

[1]This is the first time that the Fokas method is implemented computationally in any generality.

[2]In Julia: Open-source allows for checking correctness.

[3]Numeric and symbolic features.

[4]For this class of IBVPs, no other solver algorithm yet exists.

IBVPs solvable by the Fokas method

The Fokas method allows solving any well-posed IBVP of the form

$$(\partial_t + a(-i)^n \partial_x^n)\, q(x,t) = 0 \qquad \forall (x,t) \in (0,1) \times (0,T) \tag{2.1a}$$

$$q(x,0) = f(x) \in \Phi \qquad \forall x \in [0,1] \tag{2.1b}$$

$$q(\cdot,t) \in \Phi \qquad \forall t \in [0,T], \tag{2.1c}$$

where

- $a \in \mathbb{C}$ (for the IBVP to be well-posed, we require at least that $a = \pm i$ if n is odd and $\mathrm{Re}(a) \geq 0$ if n is even),
- Φ is a set of functions that satisfy a set of homogeneous boundary conditions

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

  where B_j-s are boundary forms with boundary coefficients b_{jk}, $\beta_{jk} \in \mathbb{R}$,

  $$B_j\phi := \sum_{k=0}^{n-1} \left(b_{jk}\phi^{(k)}(0) + \beta_{jk}\phi^{(k)}(1) \right),\, j \in \{1,2,\ldots,n\}.$$

IBVPs solvable by the Fokas method

E.g., the linearized Korteweg-de Vries (KdV) equations:

Problem 1

$$\begin{aligned} q_t(x,t) + q_{xxx}(x,t) &= 0, && (x,t) \in (0,1) \times (0,T) \\ q(x,0) &= f(x), && x \in [0,1] \\ q(0,t) = q(1,t) &= 0, && t \in [0,T] \\ 2q_x(1,t) &= q_x(0,t) && t \in [0,T]. \end{aligned}$$

Problem 2

$$\begin{aligned} q_t(x,t) + q_{xxx}(x,t) &= 0, && (x,t) \in (0,1) \times (0,T) \\ q(x,0) &= f(x), && x \in [0,1] \\ q(0,t) = q(1,t) = q_x(1,t) &= 0, && t \in [0,T]. \end{aligned}$$

Algorithm: Overview

Consider an IBVP of the form

$$\begin{aligned}
(\partial_t + a(-i)^n \partial_x^n)\, q(x,t) &= 0 && \forall (x,t) \in (0,1) \times (0,T) \\
q(x,0) &= f(x) \in \Phi && \forall x \in [0,1] \\
q(\cdot,t) &\in \Phi && \forall t \in [0,T].
\end{aligned}$$

$$\begin{tikzcd}
\boxed{\begin{array}{c}(\partial_t + a\lambda^n)\hat{q}(\lambda,t) = 0 \\ \hat{q}(\lambda,0) = \hat{f}(\lambda)\end{array}} \arrow[r, "\text{Solve}"] & \boxed{\hat{q}(\lambda,t) = e^{-a\lambda^n t} F_\lambda(f)} \arrow[d, "\text{Inverse transform } f_x"] \\
\boxed{\begin{array}{c}(\partial_t + a(-i)^n \partial_x^n) q(x,t) = 0 \\ q(\cdot,t) \in \Phi\end{array}} \arrow[u, "\text{Transform } F_\lambda"] \arrow[r, dashed] & \boxed{q(x,t) = f_x(e^{-a\lambda^n t} F_\lambda(f))}
\end{tikzcd}$$

Algorithm: Overview

- Input: An IBVP of the form

$$
\begin{aligned}
(\partial_t + a(-i)^n \partial_x^n)\, q(x,t) &= 0 && \forall (x,t) \in (0,1) \times (0,T) \\
q(x,0) &= f(x) \in \Phi && \forall x \in [0,1] \\
q(\cdot,t) &\in \Phi && \forall t \in [0,T].
\end{aligned}
$$

- Construct adjoint boundary conditions of the spatial ordinary BVP.
- Using the spatial adjoint boundary conditions and other information about the IBVP, construct the Fokas transform pair F_λ and f_x.
- Output: Solution of the IBVP $q(x,t) = f_x(e^{-a\lambda^n t} F_\lambda(f))$.

Algorithm: Constructing adjoint boundary conditions

An algorithm to find adjoint boundary conditions is developed based on literature[3], which includes (among definitions of various relevant objects)

- a constructive existence theorem on the adjoint, and
- a theorem to check whether a candidate adjoint is indeed valid.

These results are expanded and adapted to an algorithm to construct valid adjoint boundary conditions.

[3] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. New York: McGraw-Hill Publishing, 1977, p. 429. ISBN: 9780070992566.

Algorithm: Constructing Fokas transform pair

Integrand construction

Let $b^\star, \beta^\star$ be the matrices associated with the adjoint boundary conditions. Let $\alpha := e^{2\pi i/n}$. For complex variable λ, define $n \times n$ matrices $W^+(\lambda)$, $W^-(\lambda)$ entry-wise by

$$W^+_{kj}(\lambda) := \sum_{r=0}^{n-1} (-i\alpha^{k-1}\lambda)^r b^\star_{jr}$$

$$W^-_{kj}(\lambda) := \sum_{r=0}^{n-1} (-i\alpha^{k-1}\lambda)^r \beta^\star_{jr}.$$

Define $n \times n$ matrix W entry-wise by

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

Define

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

Algorithm: Constructing Fokas transform pair

Integrand construction

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^{lj}_{11} is $\mathbb{W}_{l+1,j+1}$.

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

$$F^+_\lambda(f) := \frac{1}{2\pi\Delta(\lambda)} \sum_{j=1}^{n} \sum_{l=1}^{n} (-1)^{(n-1)(l+j)} \det X^{lj}(\lambda) W^+_{1j}(\lambda) \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x)\, dx$$

$$F^-_\lambda(f) := \frac{-e^{-i\lambda}}{2\pi\Delta(\lambda)} \sum_{j=1}^{n} \sum_{l=1}^{n} (-1)^{(n-1)(l+j)} \det X^{lj}(\lambda) W^-_{1j}(\lambda) \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x)\, dx.$$

Algorithm: Constructing Fokas transform pair

Contour construction

Define the contours

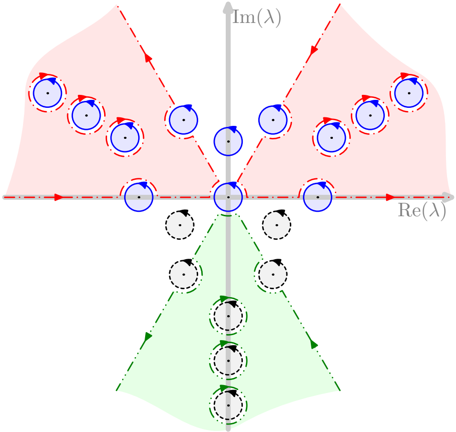
$$\Gamma_a^{\pm} := \partial(\{\lambda \in \mathbb{C}^{\pm} : \operatorname{Re}(a\lambda^n) > 0\} \setminus \bigcup_{\substack{\sigma\in\mathbb{C};\\ \Delta(\sigma)=0}} D(\sigma, 2\epsilon))$$

$$\begin{aligned}
\Gamma_a &:= \Gamma_a^+ \cup \Gamma_a^- \\
\Gamma_0^+ &:= \bigcup_{\substack{\sigma\in\operatorname{cl}\mathbb{C}^+;\\ \Delta(\sigma)=0}} C(\sigma, \epsilon) \\
\Gamma_0^- &:= \bigcup_{\substack{\sigma\in\mathbb{C}^-;\\ \Delta(\sigma)=0}} C(\sigma, \epsilon) \\
\Gamma_0 &:= \Gamma_0^+ \cup \Gamma_0^- \\
\Gamma &:= \Gamma_0 \cup \Gamma_a.
\end{aligned}$$

Algorithm: Constructing Fokas transform pair

Contour construction

Sample contour drawn by hand[4]:

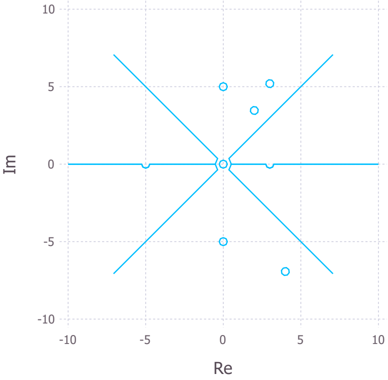


[4] David Andrew Smith and Athanassios S. Fokas. "Evolution PDEs and augmented eigenfunctions. Finite interval". In: *Journal of Spectral Theory* 6.1 (2013), pp. 185–213. ISSN: 1664-039X. DOI: 10.4171/JST/123. arXiv: 1303.2205. URL: http://www.ems-ph.org/doi/10.4171/JST/123http://arxiv.org/abs/1303.2205.

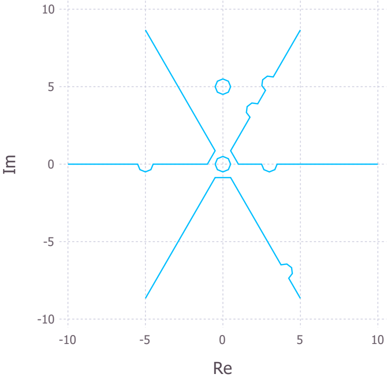
Algorithm: Constructing Fokas transform pair

Contour construction

Sample contours drawn by the implemented library[5]:



(a) $n = 2$, $a = 1$.



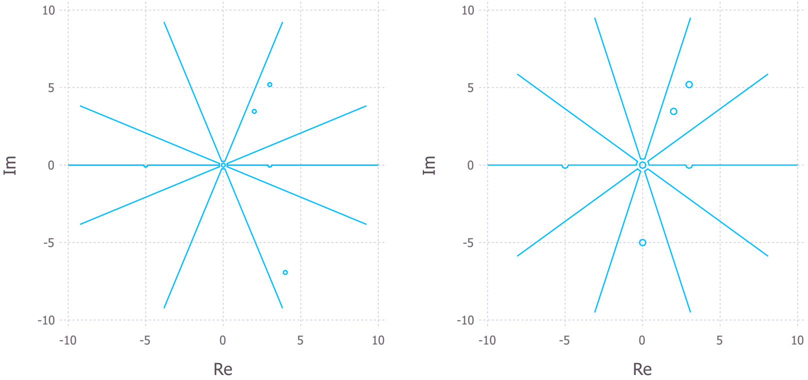
(b) $n = 3$, $a = -i$.

[5]Linfan Xiao. *Documentation of the Fokas method implementation*. URL: `https://gitlab.com/linfanxiaolinda/capstone{\_}repo/blob/master/work{\_}in{\_}julia/TheFokasmethoddocumentation.ipynb`.

Algorithm: Constructing Fokas transform pair

Contour construction

Sample contours drawn by the implemented library[6]:



(a) $n = 4$, $a = 1$. (b) $n = 5$, $a = -i$.

[6] Xiao, *Documentation of the Fokas method implementation*.

Algorithm: Constructing Fokas transform pair

Define the Fokas transform pair

$$F_\lambda : f(x) \mapsto F(\lambda) \quad F_\lambda(f) = \begin{cases} F^+_\lambda(f) & \text{if } \lambda \in \Gamma^+_0 \cup \Gamma^+_a, \\ F^-_\lambda(f) & \text{if } \lambda \in \Gamma^-_0 \cup \Gamma^-_a, \end{cases}$$

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

Algorithm: IBVP solution

The solution to the IBVP is given by

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

Sample usage: Solving IBVPs

IBVP formulation

Consider the IBVP

$$
\begin{aligned}
q_t - q_{xx} &= 0 && \forall (x,t) \in (0,1) \times (0,T) && \text{(4.1a)}\\
q(x,0) &= \sin(2\pi x) && \forall x \in [0,1] && \text{(4.1b)}\\
q(0,t) - q(1,t) &= 0 && \forall t \in [0,T] && \text{(4.1c)}\\
q_x(0,t) - q_x(1,t) &= 0 && \forall t \in [0,T]. && \text{(4.1d)}
\end{aligned}
$$

Rewriting the IBVP in the standard form, we have $n = 2$, $a = 1$,

$$
\begin{aligned}
B_1\phi &= 1 \cdot \varphi(0) + (-1) \cdot \varphi(1) + 0 \cdot \varphi^{(1)}(0) + 0 \cdot \varphi^{(1)}(1)\\
B_2\phi &= 0 \cdot \varphi(0) + 0 \cdot \varphi(1) + 1 \cdot \varphi^{(1)}(0) + (-1) \cdot \varphi^{(1)}(1),
\end{aligned}
$$

and

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

Sample usage: Solving IBVPs

Adjoint boundary conditions construction

The matrices associated with the spatial adjoint boundary conditions are

$$b^\star = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \beta^\star = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Fokas transform pair construction

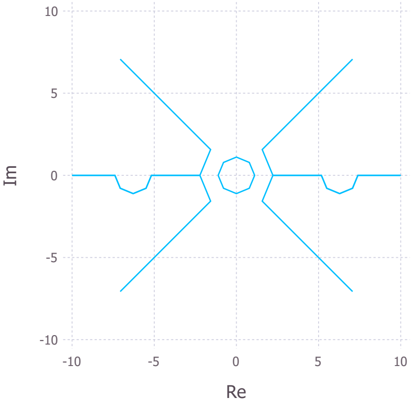
We compute the integrands $F_\lambda^+(f)$, $F_\lambda^-(f)$ to be

$$F_\lambda^+(f) = \frac{\left(\int_0^1 e^{i\lambda x} f(x)\, dx\right) e^{i\lambda}}{2\pi(e^{i\lambda} - 1)}, \quad F_\lambda^-(f) = \frac{\int_0^1 e^{i\lambda x} f(x)\, dx}{2\pi(e^{i\lambda} - 1)}.$$

Sample usage: Solving IBVPs

Fokas transform pair construction

We compute the contours to be:



IBVP solution

$q(x,t)$ can be evaluated within 20s.

Sample usage: Symbolic features

Applying the Fokas method requires two technical lemmas concerning the poles of the integrands F_λ^+, F_λ^- and their asymptotic behaviour[7].

Thus, it is necessary to obtain symbolic formulas for the integrands.

In the case of complicated integrands, it is of interest to compute their symbolic formulas using a computer.

We verified that the library computes the correct symbolic formulas for F_λ^+, F_λ^- using the following two problems.

[7] Peter D. Miller and David A. Smith. "The diffusion equation with nonlocal data". In: *Journal of Mathematical Analysis and Applications* 466.2 (2017), pp. 1119–1143. DOI: 10.1016/j.jmaa.2018.05.064. arXiv: 1708.00972. URL: http://arxiv.org/abs/1708.00972.

Sample usage: Symbolic features

Problem 1

$$\begin{aligned}
q_t(x,t) + q_{xxx}(x,t) &= 0, & (x,t) &\in (0,1)\times(0,T) \\
q(x,0) &= f(x), & x &\in [0,1] \\
q(0,t) &= 0 = q(1,t), & t &\in [0,T] \\
2q_x(1,t) &= q_x(0,t) & t &\in [0,T],
\end{aligned}$$

with

$$\begin{aligned}
F_\lambda^+(f) = \frac{1}{2\pi\Delta(\lambda)} &\Big[\hat{f}(\lambda)(e^{i\lambda} + 2\alpha e^{-i\alpha\lambda} + 2\alpha^2 e^{-i\alpha^2\lambda}) + \hat{f}(\alpha\lambda)(\alpha e^{i\alpha\lambda} - 2\alpha e^{-i\lambda}) \\
&+\hat{f}(\alpha^2\lambda)(\alpha^2 e^{i\alpha\lambda} - 2\alpha^2 e^{-i\lambda})\Big] \\
F_\lambda^-(f) = \frac{e^{-i\lambda}}{2\pi\Delta(\lambda)} &\Big[-\hat{f}(\lambda)(2 + \alpha^2 e^{-\alpha\lambda} + \alpha e^{-\alpha^2\lambda}) - \alpha\hat{f}(\alpha\lambda)(2 - e^{-i\alpha^2\lambda}) \\
&-\alpha^2\hat{f}(\alpha^2\lambda)(2 - e^{-i\alpha\lambda})\Big],
\end{aligned}$$

where $\Delta(\lambda) = e^{i\lambda} + \alpha e^{i\alpha\lambda} + \alpha^2 e^{i\alpha^2\lambda} + 2(e^{-i\lambda} + \alpha e^{-i\alpha\lambda} + \alpha^2 e^{-i\alpha^2\lambda})$ and $\hat{f}(\alpha^{l-1}\lambda) = \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x)\,dx$.

Sample usage: Symbolic features

Problem 2

$$\begin{aligned}
q_t(x,t) + q_{xxx}(x,t) &= 0, & (x,t) &\in (0,1)\times(0,T)\\
q(x,0) &= f(x), & x &\in [0,1]\\
q(0,t) &= 0, & t &\in [0,T]\\
q(1,t) &= 0 & t &\in [0,T]\\
q_x(1,t) &= 0 & t &\in [0,T],
\end{aligned}$$

with

$$\begin{aligned}
F_\lambda^+(f) &= \frac{1}{2\pi\Delta(\lambda)}\left[\hat{f}(\lambda)(\alpha e^{-\alpha\lambda} + \alpha^2 e^{-i\alpha^2\lambda}) - (\alpha\hat{f}(\alpha\lambda) + \alpha^2\hat{f}(\alpha^2\lambda))e^{-i\lambda}\right]\\
F_\lambda^-(f) &= \frac{e^{-i\lambda}}{2\pi\Delta(\lambda)}\left[-\hat{f}(\lambda) - \alpha\hat{f}(\alpha\lambda) - \alpha^2\hat{f}(\alpha^2\lambda)\right],
\end{aligned}$$

where $\Delta(\lambda) = e^{-i\lambda} + \alpha e^{-i\alpha\lambda} + \alpha^2 e^{-i\alpha^2\lambda}$ and $\hat{f}(\alpha^{l-1}\lambda) = \int_0^1 e^{-i\alpha^{l-1}\lambda x} f(x)\,dx$.

Discussion

- Next step: Speed up
  - Measures have been taken to bring down computation time to a reasonable range (e.g., by replacing double integral with explicit formulas).
  - Yet to analyze the IBVP solution in-depth, evaluating it at a given (x_0, t_0) should take less than milliseconds to allow graphing.
- Idea: Develop a custom integrator tailored to the efficient evaluation of these contour integrals by exploiting the mathematical properties of the integrand and the contour.

Summary

- The Fokas method allows solving an entire class of IBVPs of arbitrary spatial order.
- A library has been developed in Julia to provide computer aid in the process of applying the Fokas method to solve IBVPs. These include symbolic formulas of important mathematical objects, their numeric approximations, and contour visualization. Code, unit tests, and documentations are available on a public GitLab repository[8].
- Looking ahead, it is of interest to develop a custom integrator to bring the computation time from dozens of seconds to the order of milliseconds.

[8] Xiao, *Documentation of the Fokas method implementation*.