Math.Comput.Sci. (2021) 15:373–405
https://doi.org/10.1007/s11786-021-00510-7

A Symbolic-Numeric Validation Algorithm for Linear ODEs with Newton–Picard Method

Florent Bréhard

Received: 17 October 2020 / Revised: 4 March 2021 / Accepted: 16 March 2021 / Published online: 15 May 2021

Abstract A symbolic-numeric validation algorithm is developed to compute rigorous and tight uniform error bounds for polynomial approximate solutions to linear ordinary differential equations, and in particular D-finite functions. It relies on an a posteriori validation scheme, where such an error bound is computed afterwards, independently from how the approximation was built. Contrary to Newton–Galerkin validation methods, widely used in the mathematical community of computer-assisted proofs, our algorithm does not rely on finite-dimensional truncations of differential or integral operators, but on an efficient approximation of the resolvent kernel using a Chebyshev spectral method. The result is a much better complexity of the validation process, carefully investigated throughout this article. Indeed, the approximation degree for the resolvent kernel depends linearly on the magnitude of the input equation, while the truncation order used in Newton–Galerkin may be exponential in the same quantity. Numerical experiments based on an implementation in C corroborate this complexity advantage over other a posteriori validation methods, including Newton–Galerkin.

Keywords Validated numerics · A posteriori validation · Chebyshev spectral methods · Linear differential equations · D-finite functions

Mathematics Subject Classification 65G20 · 65L60 · 65L70 · 65Y20

1 Introduction

Many functions arising in scientific computing are implicitly defined by functional equations, of which *differential equations* are probably the most commonly encountered ones. Yet, one frequently requires to explicitly manipulate and evaluate these functions. Polynomials (in a broad sense) often provide accurate, global and smooth approximants, and are easy to manipulate on a computer. Moreover, in some applicative contexts, like computer-assisted proofs in mathematics, notably in dynamical systems, or safety-critical industrial applications (e.g., aerospace challenges, automatic vehicles or remote medicine), those computations must be accompanied with strong reliability guarantees.

F. Bréhard (✉)
Department of Mathematics, Uppsala University, Box 480, 751 06 Uppsala, Sweden
e-mail: florent.brehard@univ-lille.fr

1.1 Background and Related Works

A wide range of techniques in computer algebra allow for a symbolic, exact manipulation of certain classes of functions defined implicitly. A major example are the *D-finite* functions, that is the solutions of linear ordinary differential equations (ODEs) with polynomial coefficients [35]. Despite this seemingly restricted definition, D-finite functions account for about 60% [32] of the functions listed in Abramowitz and Stegun's *Handbook of Mathematical Functions* [1]. Yet, many applications require to move from symbolic representations, based on linear differential operators [33], to numeric or symbolic-numeric ones providing quantitative information. While numerical analysis focuses on algorithms to compute approximations, error estimates are most of the time asymptotic. For example, the Chebfun library[1] in Matlab allows for powerful numerical computations with functions using polynomials obtained by truncating Chebyshev expansions, but it does not provide rigorous error bounds. On the other side, *validated numerics* [37] (also called *rigorous* or *set-valued* numerics) is concerned with the computation of such error bounds, encompassing both method and rounding errors. Such a challenge, and more specifically the following problem, is at the core of the present work.

Problem 1.1 *Given a polynomial y° approximating the solution y^* of the Initial Value Problem (IVP):*

$$\begin{cases} y^{(r)}(x) + a_{r-1}(x)y^{(r-1)}(x) + \cdots + a_0(x)y(x) = h(x), & x \in [x_l, x_r], \\ y^{(i)}(x_0) = v_i \in \mathbb{R}, \quad 0 \leqslant i < r, & x_0 \in [x_l, x_r], \end{cases} \tag{1}$$

compute a bound $\varepsilon \geqslant \|y^\circ - y^*\|_\infty = \sup_{x\in[x_l,x_r]} |y^\circ(x) - y^*(x)|$.

The branch of validated numerics dedicated to the validation of functional problems has its origins in the early 80s with *ultra-arithmetics* [12,13,20]. By analogy with floating-point numbers and interval arithmetics [26] to rigorously approximate real numbers on computers, ultra-arithmetics proposes a quite general framework to provide similar operations on functions. Solutions are approximated using *polynomials* (in a broad sense), and rigorously enclosed in balls or other domains in function spaces using *rigorous polynomial approximations* (RPAs), typically pairs (p, ε) with an approximation p and a rigorous error bound ε. Notable implementations of RPAs include Taylor models [5,28] and Chebyshev models [10,19], based on Taylor and Chebyshev expansions, respectively.

Numerous techniques have been developed over past decades to compute rigorous enclosures for solutions of ODEs [17,21,24,27,30,41,42] and of wider classes of equations (e.g., partial differential equations, delay equations), thereby contributing to the rise of computer-assisted proofs in dynamics [38]. More recently, several works [2,6,8,25] paved the way towards an algorithmic and complexity approach to such methods, aiming at the development of general-purpose reliable and efficient libraries for symbolic-numeric computations. This article brings its own contributions to this approach. The counterpart is the restriction to more limited classes of ODEs, notably D-finite functions, that is, linear ODE (1) with polynomial or rational coefficients $a_i(x)$. However, we emphasize the fact that many of the ODE validation methods cited above proceed by linearization of the ODE after bounding the nonlinear terms. Also, analytic coefficients $a_i(x)$ in (1) can be rigorously represented with RPAs. Hence, the application scope of these algorithmic works, including this article in particular, goes far beyond D-finite functions: they provide an algorithmic core for the validation of differential equations in a broad perspective.

1.2 Setting and Contributions

A powerful representation of D-finite functions y in computer algebra is via differential operators $\mathbf{L}$ for which $\mathbf{L}\{y\} = 0$ (together with sufficiently many initial conditions), where:

$$\mathbf{L} = \boldsymbol{\partial}^r + a_{r-1}(\mathbf{x})\boldsymbol{\partial}^{r-1} + \cdots + a_1(\mathbf{x})\boldsymbol{\partial} + a_0(\mathbf{x}),$$

[1] https://www.chebfun.org/.

Table 1 Notation

x	Independent variable	$\\|\cdot\\|_\infty$	Uniform norm over $[x_l, x_r]$
$[x_l, x_r]$	Real compact interval of interest	$\mathcal{C}^0$	Space of continuous functions over $[x_l, x_r]$
x_0	Initial time in $[x_l, x_r]$	$\mathbb{R}_N[x]$	Space of univariate polynomials with real
T	$= \max(x_r - x_0, x_0 - x_l)$ maximum time		Coefficients of degree $\leqslant N$
$\boldsymbol{\partial}$	Differentiation operator w.r.t. x	T_n	n-th Chebyshev polynomial of the first
$\mathbf{L}$	Linear differential operator		kind, rescaled over $[x_l, x_r]$
$\mathbf{L}^*$	adjoint differential operator	$[f]_n$	n-th Chebyshev coefficient of f
r	Order of $\mathbf{L}$	$\boldsymbol{\Pi}_N$	$: \mathcal{C}^0 \to \mathbb{R}_N[x]$ N-th Chebyshev truncation
a_i	Coefficients of $\mathbf{L}$	$Ч^1$	Banach space of absolutely summable
s	$= \max_{0 \leqslant i < r} \deg a_i$ maximum degree		Chebyshev series
h	Right-hand side of differential equation	$\\|\cdot\\|_{Ч^1}$	Norm associated to $Ч^1$
y	Unknown function in differential equation	$\mathbf{K}^{[N]}$	Truncated integral operator
$y^{(i)}$	Iterated derivative of y	$\mathbf{I}$	Identity operator
$y^{[i]}$	$y^{[0]} = y$ and $y^{[i+1]} = a_{r-1-i}\, y - \boldsymbol{\partial}\{y^{[i]}\}$	I_N	Order N identity matrix
$\mathfrak{K}(x, t)$	Bivariate kernel	$\mathfrak{K} * \mathfrak{L}$	Composition of kernels
$[x_l, x_r \| x_0]$	$= \{(x, t)$ s.t. $x_l \leqslant x \leqslant t \leqslant x_0$	$\mathfrak{R}(x, t)$	Resolvent kernel
	or $x_0 \leqslant t \leqslant x \leqslant x_r\}$	$\mathfrak{R}^\circ(x, t)$	Polynomial approximation of $\mathfrak{R}(x, t)$
$\mathbf{K}$	Integral operator of kernel $\mathfrak{K}(x, t)$	$\mathcal{E}_\rho$	Bernstein ellipse of parameter $\rho > 1$
g	Right-hand side of integral equation	$N_{\mathcal{G}}$	Truncation order in Newton–Galerkin
f	Unknown function in integral equation	$N_{\mathfrak{R}}$	Degree of $\mathfrak{R}^\circ(x, t)$ in Newton–Picard

with a_i polynomials or rational functions, belongs to the *Ore algebra* of differential operators [11,22], spanned by $\mathbf{x}$ ($\mathbf{x}\{y\} = x \mapsto xy(x)$) and $\boldsymbol{\partial}$ ($\boldsymbol{\partial}\{y\} = y'$), with the skew commutation rule $\boldsymbol{\partial}\mathbf{x} - \mathbf{x}\boldsymbol{\partial} = 1$. In the theoretical developments of this article, the a_i may actually be analytic functions over the real compact interval $[x_l, x_r]$ of interest. For the algorithmic parts, they are supposed to be polynomials (with rational, floating-point or interval coefficients) for the sake of clarity, even if adapting the algorithms to RPA representations of the $a_i(x)$ is rather straightforward (see the example of Sect. 4.3). We thus propose a fully algorithmic validation method for Problem 1.1 with analytic coefficients a_i representable by RPAs. In particular, iterating this construction allows for the rigorous numerical treatment of D^n-finite functions [18]. Also, in the case of non-homogeneous linear ODEs, the same remarks hold for the right-hand side h in (1).

Our validation method is based on an automatic reformulation of IVP (1) into an equivalent Volterra linear integral equation of the second kind [23] (simply referred to as *integral equation* in this article):

$$f(x) + \mathbf{K}\{f\}(x) = g(x), \qquad \text{where} \quad \mathbf{K}\{f\}(x) = \int_{x_0}^{x} \mathfrak{K}(x, t) f(t) \mathrm{d}t. \tag{2}$$

The bivariate kernel $\mathfrak{K}(x, t)$ associated to the integral operator $\mathbf{K}$ is analytic, and even polynomial if the a_i are. Therefore, representations through integral equations are equally algorithmic. Moreover, such integral equations are particularly suited for numerical resolution via Galerkin spectral methods [7,15,29], which compute a polynomial approximation y° for Problem 1.1 using truncated expansions (e.g., Taylor, Fourier, Chebyshev), by transforming functional equation (2) into a linear system on the unknown coefficients.

Several works, e.g. [17,24,41], proposed ODE validation methods based on *a posteriori* Newton-like methods in well-chosen coefficient spaces. The main idea is to approximate the (infinite-dimensional) inverse Jacobian by finite-dimensional truncations, as done by spectral methods in numerical analysis, in order to construct a *contracting* Newton-like operator and apply the Banach fixed-point theorem or a variant of it. Such methods, which we propose to call *Newton–Galerkin* validation methods, have been extensively used in the mathematical community of computer-assisted proofs. In [8], we developed a detailed analysis of a Newton–Galerkin validation algorithm in the Chebyshev basis, by enhancing this general scheme with the linear algebra algorithms of [29] for so-called *almost-banded* linear systems. In the present article, we propose an alternative validation algorithm, which we call *Newton–Picard*, to overcome the complexity shortcomings of Newton–Galerkin validation methods, acknowledged in [8]. The leitmotif is to approximate the inverse Jacobian by computing polynomial approximations of the so-called *resolvent kernel* associated to (1) which arises from the well-known Picard iterations. More or less related ideas have already been used in previous works, notably [2,21,30]. However, the algorithmic description and complexity analysis we propose here lead to a robust and efficient validation algorithm, together with a fair comparison with Newton–Galerkin. Our conclusions are the following.

1. We prove that Newton–Picard runs in polynomial time with respect to the magnitude of the coefficients in the input equation, while Newton–Galerkin's complexity can be exponential in the same parameters, even, surprisingly, in the analytic case we consider here. The result is that moderate to hard instances of Problem 1.1 (e.g., exponentially increasing or highly oscillating functions) may be intractable with Newton–Galerkin, but solved within a few seconds with Newton–Picard on a modern computer.
2. The Newton–Picard validation method is widely independent of the underlying approximation tools, although presented in a Chebyshev approximation theory flavor in this article. Indeed, the algorithms could be parametrized by a generic type of RPAs to rigorously enclose functions without major modifications. Instead of Chebyshev models, one could use Fourier approximations (for trigonometric functions), splines (for $\mathcal{C}^k$ functions), depending on the context. Validation of solutions with singularities or on unbounded intervals also belongs to future applications of Newton–Picard method.
3. A challenging goal we want to address in the future is the formalization in a proof assistant, for example Coq [4], of an ODE a posteriori validation algorithm. To this purpose, Newton–Picard turns out to be a better candidate than Newton–Galerkin. Indeed, the correctness of the algorithms presented in this article requires very limited functional analysis background, contrary to Newton–Galerkin which involves more Chebyshev approximation theory. Moreover, the resolvent kernel approximation can be safely computed *outside* the proof assistant without compromising the final reliability.

Throughout this article, we consider an arithmetic model for the complexity, where each arithmetic operation ($+$, $-$, $\times$, $\div$), typically on floating-point numbers or intervals, is assumed to have a constant cost $\mathcal{O}(1)$. Therefore, the claimed complexity estimates do not take into account the fact that particularly stiff ODEs may require a larger floating-point bit precision to provide sufficiently accurate results.

1.3 Outline

First, Sect. 2 provides the necessary background (a posteriori Newton-like validation principle, reformulation of IVP (1) into integral Eq. (2), Chebyshev approximation theory and spectral methods) and summarizes the main lines of [8] about the Newton–Galerkin validation method. Then, Sect. 3 introduces Newton–Picard validation method, with the main algorithm NewtonPicardValid and the approximation routine ResolventKernel for the resolvent kernel approximation, together with detailed complexity results. After that, Sect. 4 presents how the method is implemented in the C library ChebValid[2] and illustrates its performances on practical examples, in comparison with other methods. We finally discuss the obtained results and the future perspectives of this work.

[2] https://gitlab.inria.fr/brisebar/tchebyapprox/-/releases/v0.0.1.

2 A Posteriori Validation and Newton–Galerkin Method

This section lays out the main prerequisites to this article: *a posteriori validation* paradigm using Newton-like fixed-point operators (Sect. 2.1), transformation(s) of linear IVPs (1) into integral equations (2) (Sect. 2.2), some highlights of Chebyshev approximation theory and in particular *spectral methods* (Sect. 2.3). Finally, Sect. 2.4 gives an overview of the principles and limitations of Newton–Galerkin validation method, thoroughly investigated in [8]. This serves as a basis for comparison with the novel Newton–Picard algorithm presented in Sect. 3.

2.1 Principle of A Posteriori Validation Using Newton-Like Fixed-Point Operators

In an a posteriori validation scheme, an approximation x° to the exact solution x^* is first constructed by whatever numerical routine of choice, and then a rigorous error bound between x° and x^* is computed in a totally independent second step. Some advantages of this paradigm include the possibility to use external non-trusted numerical routines for the approximation step without compromising the final reliability; the need for rigorous numerics (e.g., interval arithmetics) in the validation step only; and a rather elegant mathematical formulation that eases the verification by a formal proof assistant.

A wide range of such methods involve *fixed-point* based validation, in which x^* is expressed as the fixed point of a well chosen equation $\mathbf{T}(x) = x$, with $\mathbf{T}$ a contracting operator. Then a rigorous error bound is obtained using a suitable fixed-point theorem, for example this formulation of the Banach fixed-point theorem [3, Thm. 2.1].

Theorem 2.1 *Let* $\mathbf{T} : X \to X$ *with* $(X, \|\cdot\|)$ *a Banach space and* $D \subseteq X$ *closed, such that:*

- *D is $\mathbf{T}$-stable:* $\mathbf{T}(D) \subseteq D$;
- *$\mathbf{T}$ is contracting over D: there exists* $\lambda \in [0, 1)$ *such that:*

$$\|\mathbf{T}(x) - \mathbf{T}(y)\| \leqslant \lambda \|x - y\|, \qquad \textit{for all } x, y \in D.$$

Then:

- *$\mathbf{T}$ has a unique fixed point x^* in D;*
- *For any $x^\circ \in D$,*

$$\frac{d}{1+\lambda} \leqslant \|x^\circ - x^*\| \leqslant \frac{d}{1-\lambda}, \qquad \textit{where} \quad d = \|\mathbf{T}(x^\circ) - x^\circ\|.$$

Most of the problems encountered in scientific computing are certainly not of this form. In this context, *Newton-like validation methods* (also called *Krawczyk methods*) reuse the ideas behind Newton-Raphson iterations in numerical analysis to transform the initial problem into an instance on which Theorem 2.1 applies. Assume that the original equation is of the form $\mathbf{F}(x) = 0$, with $\mathbf{F} : X \to Y$ a Fréchet differentiable map between Banach spaces X and Y. The exact solution x^* is supposed to be simple root of $\mathbf{F}$, so that the Fréchet derivative $\mathrm{D}\mathbf{F}(x)$ is invertible in a neighborhood of x^*. We are given an approximation x° of x^*.

1. Compute a bounded injective linear operator $\mathbf{A}$ approximating $\mathrm{D}\mathbf{F}(x^\circ)^{-1}$, the inverse of the Fréchet derivative of $\mathbf{F}$ at x°. Then $\mathbf{F}(x) = 0$ is equivalent to the fixed-point equation associated to the Newton-like operator $\mathbf{T}$:

$$\mathbf{T}(x) = x, \qquad \text{where} \quad \mathbf{T}(x) = x - \mathbf{A}(\mathbf{F}(x)).$$

2. Compute *rigorously* a closed subset $D \subseteq X$ containing x° (e.g. a ball centered at x° with suitable radius r) and a contraction ratio $\lambda < 1$ satisfying the assumptions of Theorem 2.1.
3. Return the error bound $\varepsilon = \|\mathbf{A}(\mathbf{F}(x^\circ))\|/(1 - \lambda)$ as in Theorem 2.1. If $\mathbf{F}$ has several roots in X, also ensure that the unique fixed-point of $\mathbf{T}$ inside D is necessarily x^*.

Remark 2.2 The IVP (1) (or the equivalent integral equation (2)) we consider in this article are linear, which somewhat simplifies the scheme above. The Newton-like operator $\mathbf{T}$ is indeed affine, and hence it is locally contracting if and only if it is globally contracting (i.e. $D = X$), and λ is given by the operator norm of its linear part. Moreover, the solution being unique by the Picard-Lindelöf theorem, no additional verification is required in step 3 above.

2.2 Reformulation of Linear ODE into Integral Equation

Several reasons may lead us to prefer integral equations rather than differential ones in numerical analysis. From a theoretical point of view, the differentiation operator $\boldsymbol{\partial}$ can rarely be seen as a bounded linear endomorphism in one function space, since it typically *reduces* the regularity. Manipulating differential equations thus requires to consider a separate function space for each iterated application of $\boldsymbol{\partial}$ (which is for example the approach of [29] using ultraspherical polynomials). On the contrary, the operation of integration *increases* the regularity, which allows us to work in a single function space. Moreover, the compactness property, necessary for the projection approach of spectral methods, easily follows (see Sect. 2.3). On the practical side also, integral equations often lead to better conditioned numerical problems [16].

There are several possibilities to transform a linear ODE into an equivalent integral equation, acting on the same function y or one of its iterated derivatives. In this article, we focus on two such transforms, giving integral equations acting on y and $y^{(r)}$, respectively. The obvious "duality" between them will play an important role later in Sect. 3.

2.2.1 Integral Equation on Highest-Order Derivative

This transformation, considered for example in [8], is the simplest one, acting on $f = y^{(r)}$. We here consider a differential equation $\mathbf{L}\{y\} = h$ with a linear differential operator $\mathbf{L}$ given in a "standard form", which we call $\mathbf{x} - \boldsymbol{\partial}$ form:

$$\mathbf{L} = \boldsymbol{\partial}^r + a_{r-1}(x)\boldsymbol{\partial}^{r-1} + \cdots + a_1(x)\boldsymbol{\partial} + a_0(x). \tag{3}$$

Proposition 2.3 *The linear initial value problem:*

$$\begin{cases} \mathbf{L}\{y\}(x) = h(x), & x \in [x_l, x_r], \\ y^{(i)}(x_0) = v_i, & 0 \leqslant i < r, \end{cases} \quad \textit{with} \quad \mathbf{L} \textit{ in } \mathbf{x} - \boldsymbol{\partial} \textit{ form } (3),$$

is equivalent to the integral equation on $f = y^{(r)}$:

$$f(x) + \mathbf{K}\{f\}(x) = g(x), \quad \textit{where} \quad \mathbf{K}\{f\}(x) = \int_{x_0}^{x} \mathfrak{K}(x,t) f(t) \mathrm{d}t,$$

$$\textit{with} \quad \mathfrak{K}(x,t) = \sum_{i=0}^{r-1} a_i(x) \frac{(x-t)^{r-1-i}}{(r-1-i)!}, \quad \textit{and} \quad g(x) = h(x) - \sum_{i=0}^{r-1} a_i(x) \sum_{j=i}^{r-1} \frac{(x-x_0)^{j-i}}{(j-i)!} v_j.$$

Proof By a repeated use of the integration by part formula, one has for $0 \leqslant i < r$:

$$\begin{aligned} y^{(i)}(x) &= \int_{x_0}^{x} \int_{x_0}^{t_1} \cdots \int_{x_0}^{t_{r-1-i}} f(t_{r-i}) \mathrm{d}t_{r-1} \ldots \mathrm{d}t_1 + \sum_{j=i}^{r-1} \frac{(x-x_0)^{j-i}}{(j-i)!} v_j \\ &= \int_{x_0}^{x} \frac{(x-t)^{r-1-i}}{(r-1-i)!} f(t) \mathrm{d}t + \sum_{j=i}^{r-1} \frac{(x-x_0)^{j-i}}{(j-i)!} v_j. \end{aligned}$$

Doing the substitutions in $\mathbf{L}\{y\} = h$, regrouping the integral terms and moving the initial condition terms to the right, one obtains the expected kernel $\mathfrak{K}(x, t)$ and the right-hand side $g(x)$. □

2.2.2 *Integral Equation on the Same Function*

Another possible reformulation, used for example in [2], is to define an integral equation directly on $f = y$. To do so, we need to have $\mathbf{L}$ in a $\boldsymbol{\partial} - \mathbf{x}$ form, with the polynomial (or analytic) coefficients $a_i(x)$ located to the right of the $\boldsymbol{\partial}$ symbols:

$$\mathbf{L} = (-\boldsymbol{\partial})^r + (-\boldsymbol{\partial})^{r-1} a_{r-1}(x) + \cdots - \boldsymbol{\partial} a_1(x) + a_0(x). \tag{4}$$

We systematically use $-\boldsymbol{\partial}$ instead of simply $\boldsymbol{\partial}$ in $\boldsymbol{\partial} - \mathbf{x}$ forms to be consistent with the adjunction $\mathbf{L} \mapsto \mathbf{L}^*$ that will play a crucial role in Sect. 3.3.

While a $\mathbf{x} - \boldsymbol{\partial}$ form linear ODE relates together the iterated derivatives $y^{(i)}$ where $y^{(i+1)} = \boldsymbol{\partial}\{y^{(i)}\}$, it is more natural with a $\boldsymbol{\partial} - \mathbf{x}$ form to consider functions $y^{[i]}$ for $0 \leqslant i \leqslant r$, with $y^{[0]} = y$, $y^{[i+1]} = a_{r-i-1} y - \boldsymbol{\partial}\{y^{[i]}\}$, and then equating $y^{[r]}(x) = h(x)$. By doing so, we have $y^{[i]} = \mathbf{L_{[i]}}\{y\}$ where:

$$\mathbf{L_{[i]}} = (-\boldsymbol{\partial})^i + (-\boldsymbol{\partial})^{i-1} a_{r-1}(x) + \cdots - \boldsymbol{\partial} a_{r-i+1}(x) + a_{r-i}(x),$$

so that the "natural" $\boldsymbol{\partial} - \mathbf{x}$ initial conditions at x_0 are $y^{[i]}(x_0) = \mathbf{L_{[i]}}\{y\}(x_0) = w_i$ for $0 \leqslant i < r$.

Remark 2.4 It is clear that, as long as one can differentiate the $a_i(x)$ symbolically sufficiently many times, one easily converts between $\mathbf{x} - \boldsymbol{\partial}$ and $\boldsymbol{\partial} - \mathbf{x}$ operators $\mathbf{L}$ and initial conditions. Note however that for large r, this may result in a differential operator with enormous coefficients, potentially leading to numerical stability issues.

The integral equation on $f = y$ follows by merely integrating the linear ODE r times.

Proposition 2.5 *The linear initial value problem:*

$$\begin{cases} \mathbf{L}\{y\}(x) = h(x), & x \in [x_l, x_r], \\ \mathbf{L_{[i]}}\{y\}(x_0) = w_i, & 0 \leqslant i < r, \end{cases} \qquad \textit{with} \quad \mathbf{L} \textit{ in } \boldsymbol{\partial} - \mathbf{x} \textit{ form } (4),$$

is equivalent to the integral equation on y:

$$y(x) + \mathbf{K}\{y\}(x) = g(x), \qquad \textit{where} \quad \mathbf{K}\{y\}(x) = \int_{x_0}^{x} \mathfrak{K}(x, t) y(t) \mathrm{d}t,$$

$$\textit{with} \quad \mathfrak{K}(x, t) = -\sum_{i=0}^{r-1} \frac{(t - x)^{r-1-i}}{(r - 1 - i)!} a_i(t), \qquad \textit{and} \qquad g(x) = -\int_{x_0}^{x} \frac{(t - x)^{r-1}}{(r - 1)!} h(t) \mathrm{d}t + \sum_{i=0}^{r-1} w_i \frac{(x_0 - x)^i}{i!}.$$

Proof We first prove by induction on $0 \leqslant k \leqslant r$:

$$\int_{x_0}^{x} \ldots \int_{x_0}^{t_{k-1}} y^{[k]}(t_k) \mathrm{d}t_k \ldots \mathrm{d}t_1 = (-1)^k \left(y - \int_{x_0}^{x} \sum_{i=r-k}^{r-1} \frac{(t - x)^{r-1-i}}{(r - 1 - i)!} a_i(t) y(t) \mathrm{d}t - \sum_{i=0}^{k-1} \frac{(x_0 - x)^i}{i!} w_i \right). \tag{5}$$

This is trivially true for $k = 0$. If it holds for $k < r$, since $y^{[k+1]} = a_{r-k-1}y - \partial\{y^{[k]}\}$, then:

$$\begin{aligned}\int_{x_0}^{x}\!\!\ldots\!\int_{x_0}^{t_k} y^{[k+1]}(t_{k+1})\mathrm{d}t_{k+1}\ldots\mathrm{d}t_1 &= \int_{x_0}^{x}\!\!\ldots\!\int_{x_0}^{t_k} a_{r-k-1}(t)y(t)\mathrm{d}t_{k+1}\ldots\mathrm{d}t_1 - \int_{x_0}^{x}\!\!\ldots\!\int_{x_0}^{t_{k-1}} (y^{[k]}(t_k) - w_k)\mathrm{d}t_k\ldots\mathrm{d}t_1 \\ &= \int_{x_0}^{x} \frac{(x-t)^k}{k!} a_{r-k-1}(t)y(t)\mathrm{d}t - \int_{x_0}^{x}\!\!\ldots\!\int_{x_0}^{t_{k-1}} y^{[k]}(t_k)\mathrm{d}t_k\ldots\mathrm{d}t_1 + \frac{(x-x_0)^k}{k!}w_k \\ &= (-1)^{k+1}\left(y - \int_{x_0}^{x} \sum_{i=r-k-1}^{r-1} \frac{(t-x)^{r-1-i}}{(r-1-i)!} a_i(t)y(t)\mathrm{d}t - \sum_{i=0}^{k} \frac{(x_0-x)^i}{i!} w_i\right).\end{aligned}$$

This completes the proof of (5). Taking this equality with $k = r$ allows us to rewrite:

$$(-1)^r \int_{x_0}^{x}\!\!\ldots\!\int_{x_0}^{t_{k-1}} y^{[r]}(t_k)\mathrm{d}t_k\ldots\mathrm{d}t_1 = (-1)^r \int_{x_0}^{x}\!\!\ldots\!\int_{x_0}^{t_{k-1}} h(t_k)\mathrm{d}t_k\ldots\mathrm{d}t_1,$$

into the desired integral equation with expected $\mathfrak{K}(x, t)$ and $g(x)$. □

2.3 Chebyshev Approximation Theory and Spectral Methods

When dealing with analytic functions over a compact real interval $[x_l, x_r]$, Chebyshev approximation theory turns out to be an efficient generic tool to provide global, smooth and remarkably accurate approximants. Here we simply recall some fundamentals of this beautiful and rich theory, to which numerous introductory books are dedicated [7,14,36].

The Chebyshev polynomials form a graded family of univariate polynomials, commonly defined over $[-1, 1]$ by the three-term recurrence:

$$T_0(x) = 1, \qquad T_1(x) = x, \qquad T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x) \quad \text{for } n \geqslant 0.$$

The mantra from a complexity point of view is that manipulating polynomials in the Chebyshev basis is asymptotically equivalent to the same operations in the monomial basis. Indeed, multiplication and differentiation/integration are given by the formulas:

$$T_n(x)T_m(x) = \frac{1}{2}(T_{n+m}(x) + T_{n-m}(x)), \qquad \text{for } 0 \leqslant m \leqslant n, \tag{6}$$

$$T_n(x) = \frac{1}{2}\left(\frac{T_{n+1}(x)}{n+1} - \frac{T_{n-1}(x)}{n-1}\right)', \qquad \text{for } n \geqslant 2. \tag{7}$$

Remark 2.6 Multiplication of two degree n polynomials in the Chebyshev basis can be computed in $\mathcal{O}(n \log n)$ floating-point arithmetic operations thanks to fast algorithms for Discrete Cosine Transform (DCT), which is the analog to the Fast Fourier Transform (FFT). However, designing a numerically stable algorithm for fast multiplication of polynomials with interval coefficients (in the monomial or Chebyshev basis) is still an open and debated topic.[3] This is why in Sect. 3 we will assume a quadratic complexity when interval arithmetics is needed.

From an analysis point of view, the relation $T_n(\cos\vartheta) = \cos(n\vartheta)$ strongly connects Chebyshev polynomials with Fourier approximation theory. In particular, $|T_n(x)| \leqslant 1$ for $-1 \leqslant x \leqslant 1$ and they form an orthogonal family

[3] see, e.g., https://www.texmacs.org/joris/stablemult/stablemult-abs.html.

w.r.t. the scalar product:

$$\langle f, g\rangle = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\mathrm{d}x = \int_{0}^{\pi} f(\cos\vartheta)g(\cos\vartheta)\mathrm{d}\vartheta,$$

which motivates the definition of *Chebyshev coefficients* for a continuous function f over $[-1,1]$ by orthogonal projection:

$$[f]_n = \frac{\langle f, T_n\rangle}{\langle T_n, T_n\rangle} = \begin{cases} \frac{1}{\pi}\langle f, 1\rangle & \text{for } n = 0, \\ \frac{2}{\pi}\langle f, T_n\rangle & \text{for } n \geqslant 1. \end{cases}$$

Remark 2.7 By an affine change of variable, the above definitions are transposed from $[-1,1]$ to any compact interval. We assume in this article that the Chebyshev polynomials T_n are defined over $[x_l, x_r]$.

Chebyshev approximations are defined by truncated Chebyshev series using the *truncation* (or *projection*) operator:

$$\mathbf{\Pi}_N \quad : \quad \mathcal{C}^0 \to \mathbb{R}_N[x], \quad f \mapsto \mathbf{\Pi}_N\{f\} = \sum_{n=0}^{N} [f]_n T_n.$$

The key point of Chebyshev approximation theory is that the more regular f is, the faster $\mathbf{\Pi}_N\{f\}$ converges to f as $N \to \infty$. In the analytic case we consider—and contrary to Taylor approximations—convergence on $[x_l, x_r]$ always holds, at exponential rate with ratio ρ determined by the location of singularities in the complex plane (Lemma 3.15 in Sect. 3 gives a quantitative estimate).

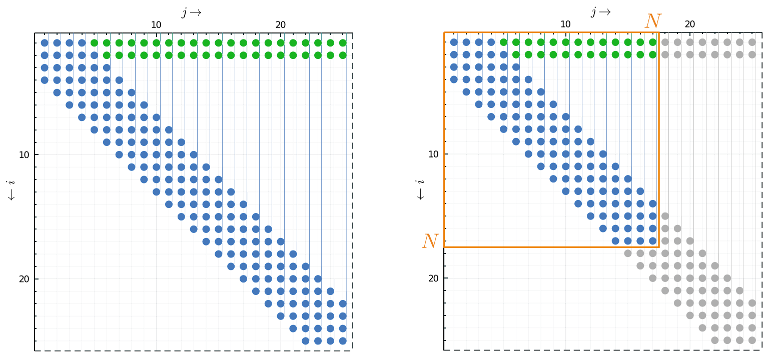
When working with infinite sequences of Chebyshev coefficients, a relevant subspace of $\mathcal{C}^0$ is the space Ч^1 of absolutely summable Chebyshev series [8, Sec. 2.2], which is the analog of the Wiener algebra $A(\mathbb{T})$ for Fourier series.

Definition 2.8 The Ч^1 space is the space of continuous functions f over $[x_l, x_r]$ for which the quantity,

$$\|f\|_{\text{Ч}^1} = \sum_{n=0}^{\infty} |[f]_n|,$$

is finite. This makes $(\text{Ч}^1, \|\cdot\|_{\text{Ч}^1})$ a Banach space, with the property $\|f\|_{\text{Ч}^1} \geqslant \|f\|_\infty$.

Obviously, Chebyshev coefficients $[f]_n$ cannot be directly computed most of the time. Turning back to the integral equation (2), a spectral method approach [7,15] is to cast this problem into an infinite-dimensional linear system in the Chebyshev space of coefficients. More specifically, the integral operator $\mathbf{K}$ of polynomial kernel $\mathfrak{K}(x,t)$ becomes a bounded endomorphism of Ч^1, acting on the vector of Chebyshev coefficients of a function f as an infinite-dimensional matrix (depicted in Fig. 1a). The multiplication (6) and integration (7) formulas imply that $\mathbf{K}$ has a sparse structure called *almost-banded* (following the terminology of [29]), that is, nonzero coefficients are located in a finite number of rows and diagonals. Moreover, $\mathbf{K}$ is a *compact* endomorphism of Ч^1, where coefficients in the matrix tend to zero. Therefore, spectral methods relax this infinite-dimensional problem by projecting it onto a finite-dimensional subspace, using the truncated integral operator $\mathbf{K}^{[N]} = \mathbf{\Pi}_N\, \mathbf{K}\, \mathbf{\Pi}_N$ (see Fig. 1b). The resulting almost-banded linear system of dimension $N+1$ is eventually solved in linear time w.r.t. the approximation degree N using the QR-like algorithms of [29].



(A) $\mathbf{K}$ **(B)** $\mathbf{K}^{[N]}$

Fig. 1 Almost-banded matrix representations in the Chebyshev basis of an integral operator $\mathbf{K}$ and its truncation $\mathbf{K}^{[N]}$

Proposition 2.9 (see [29, Sec. 5]) *The solution of the integral equation* (2) *can be approximated by a degree* N *polynomial in the Chebyshev basis using a spectral method requiring* $\mathcal{O}(N(r+s)^2)$ *arithmetic operations. More precisely:*

- *A first algorithm* AlmostBandedQRFactor *computes a (structured) QR factorization of* $I_{N+1} + \mathbf{K}^{[N]}$ *in* $\mathcal{O}(N(r+s)^2)$ *arithmetic operations.*
- *Given this QR factorization, a second algorithm* AlmostBandedBackSubs *performs a back-substitution to solve:*

$$(\mathbf{I}_{N+1} + \mathbf{K}^{[N]})\{y^\circ\} = \mathbf{\Pi}_N\{g\},$$

of unknown $y^\circ = \sum_{n=0}^{N} c_n T_n$, *in* $\mathcal{O}(N(r+s))$ *arithmetic operations.*

2.4 A Brief Summary of the Newton–Galerkin Validation Method

Problem 1.1 typically falls within the class of function space problems that can be solved using the fixed-point based a posteriori validation scheme presented in Sect. 2.1. One needs to compute a rigorous error bound between an approximation f° and the exact solution f^* of the affine equation:

$$\mathbf{F}\{f\} = 0, \qquad \text{where} \quad \mathbf{F}\{f\} = f + \mathbf{K}\{f\} - g \; : \; \text{Ч}^1 \to \text{Ч}^1.$$

We recall the main ideas of Newton–Galerkin validation method, in the way it is presented in [8], with complexity analysis results and discussion about its shortcomings.

2.4.1 Newton–Galerkin Fixed-Point Operator and Validation Algorithm

The principle of the Newton–Galerkin validation algorithm is to construct the linear operator $\mathbf{A} \approx (\mathbf{I} + \mathbf{K})^{-1}$ (required by step 1 of Newton-like validation) using finite dimensional projections. Indeed, $\mathbf{K}^{[N]}$ converges to $\mathbf{K}$, so the algorithm chooses a suitable truncation order $N_{\mathcal{G}}$ (which is totally independent of the approximation f° to be validated) and computes a numerical inverse A of the $(N_{\mathcal{G}}+1)$-square matrix $I_{N_{\mathcal{G}}+1} + \mathbf{K}^{[N_{\mathcal{G}}]}$. $\mathbf{A}$ is defined by extending the matrix A over Ч^1 with identity (see Fig. 2a).

After that, in step 2, a contraction ratio λ for the resulting Newton–Galerkin fixed-point operator $\mathbf{T}$ is automatically computed by rigorously bounding the operator norm (w.r.t. the $\mathrm{Ч}^1$-norm) of the linear part, simply denoted by D$\mathbf{T}$:

$$\|\mathrm{D}\mathbf{T}\|_{\mathrm{Ч}^1} = \|\mathbf{I} - \mathbf{A}(\mathbf{I} + \mathbf{K})\|_{\mathrm{Ч}^1} \leqslant \|\mathbf{I} - \mathbf{A}(\mathbf{I} + \mathbf{K}^{[N_\mathcal{G}]})\|_{\mathrm{Ч}^1} + \|\mathbf{A}(\mathbf{K}^{[N_\mathcal{G}]} - \mathbf{K})\|_{\mathrm{Ч}^1}. \tag{8}$$

- The first quantity $\|\mathbf{I} - \mathbf{A}(\mathbf{I} + \mathbf{K}^{[N_\mathcal{G}]})\|_{\mathrm{Ч}^1}$ is due to the fact that A is computed as a numerical inverse (either dense, or even approximated by an almost-banded matrix, see [8, sec. 4.2]). Bounding this quantity only requires straightforward operations on finite-dimensional matrices.
- The second quantity $\|\mathbf{A}(\mathbf{K}^{[N_\mathcal{G}]} - \mathbf{K})\|_{\mathrm{Ч}^1}$, where the two matrices of this product are depicted in Fig. 2, is a consequence of the fact that $\mathbf{K}^{[N_\mathcal{G}]}$ is only an approximation of $\mathbf{K}$. The difficulty, addressed in details in [8, Sec. 5.1], consists in rigorously computing a tight bound for the resulting infinite-dimensional tail, in reasonable complexity. Indeed, gross overestimations would force us to use a larger truncation index $N_\mathcal{G}$ to ensure a contraction ratio $\lambda < 1$, thus strongly impacting the overall efficiency.

Finally, the defect $d = \|\mathbf{T}\{f^\circ\} - f^\circ\|_{\mathrm{Ч}^1} = \|\mathbf{A}\{f^\circ + \mathbf{K}\{f^\circ\} - g\}\|_{\mathrm{Ч}^1}$ (step 3) is rigorously computed in a straightforward manner, yielding the final bound $d/(1-\lambda)$.

Theorem 2.10 (see [8, Prop. 5.3]) *Let f° be an approximate solution for the integral equation* (2)*. Then, given a validation truncation index $N_\mathcal{G}$, algorithms in [8], if they do not fail, realize the following*[4]*:*

- *A Newton–Galerkin fixed-point validation operator* $\mathbf{T}$ *with truncation index $N_\mathcal{G}$ is computed and bounded in*

  $$\mathcal{O}\left(N_\mathcal{G}^2(r+s)\right) \quad \textit{arithmetic operations},$$

  where r is the order of $\mathbf{L}$ *and $s = \max_i \deg a_i$ in Problem 1.1.*
- *Having this Newton–Galerkin operator and if the rigorously computed contraction ratio $\lambda < 1$, then a candidate approximation f° (with right-hand side g) is validated in*

  $$\mathcal{O}\left(N_\mathcal{G}(n+m+r+s) + n(r+s)^2\right) \quad \textit{arithmetic operations},$$

  where $n = \deg f^\circ$ and $m = \deg g$.

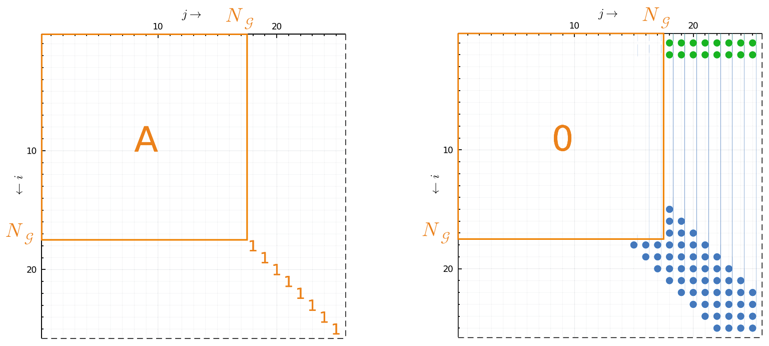
Obviously, the overall efficiency of the procedure depends on how large $N_\mathcal{G}$ must be to ensure a contracting operator $\mathbf{T}$. This crucial question is discussed in the next section.

2.4.2 Limitations of the Newton–Galerkin Validation Method

As pointed out in [8], the Newton–Galerkin validation method, although widely used in the mathematical community of computer-assisted proofs, suffers from several limitations, especially when targeting certified implementation in a proof assistant.

The first shortcoming is the difficulty to process "harder" instances, e.g. LODEs with highly oscillating or rapidly increasing solutions, due to the value of the truncation index $N_\mathcal{G}$. Indeed, $\mathbf{K}^{[N_\mathcal{G}]}$ converges to $\mathbf{K}$ in $\mathcal{O}(1/N_\mathcal{G})$ only, while the $\mathrm{Ч}^1$-norm of $(\mathbf{I}+\mathbf{K})^{-1}$ (and hence the approximation $\mathbf{A}$) is exponentially bounded by the magnitude $\|a_i\|_{\mathrm{Ч}^1}$ of the input LODE coefficients. This explains why the minimum $N_\mathcal{G}$ to ensure a contracting operator $\mathbf{T}$ can grow exponentially fast with the $\|a_i\|_{\mathrm{Ч}^1}$ [8, Sec. 5.2]. Although other norms for the space of Chebyshev coefficients have been investigated, e.g. in [17,24], it seems that none of them avoid this exponential bound.

[4] Refined complexity estimates are given in [8, Prop. 5.3] when using an almost-banded A instead of a dense one.



(A) A **(B)** $\mathbf{K} - \mathbf{K}^{[N_{\mathcal{G}}]}$

Fig. 2 Bounding the truncation error $\|\mathbf{A}(\mathbf{K} - \mathbf{K}^{[N_{\mathcal{G}}]})\|_{\mho^1}$ in Newton–Galerkin method

Another limitation of Newton–Galerkin validation methods is that implementations can hardly be generic w.r.t. the approximation setting. Indeed, changing the approximation basis from (T_n) to another (φ_n) (e.g., Fourier, Legendre) or the norm implies substantial modifications in the technical formulas used to bound the operator norm of $\mathbf{T}$. Moreover, not all norms are suitable when working with coefficient spaces. This is for example why we use $\| \cdot \|_{\mho^1}$ to overapproximate $\| \cdot \|_\infty$ here.

Finally, the need to consider infinite-dimensional matrix representations of the involved operators, as well as the rather technical formulas that are used to bound them, makes Newton–Galerkin not the best candidate for a certified implementation in a proof assistant. The Newton–Picard validation algorithm presented in the next section not only offers a far more satisfying solution to the above-mentioned issues of complexity and genericness, it is also in essence more "algebraic" and adapted to formal proof.

3 Newton–Picard Validation Algorithm

The conclusion to draw from Newton–Galerkin's pessimistic complexity estimates is that $(\mathbf{I} + \mathbf{K})^{-1}$ is poorly approximated by finite-dimensional truncations $(\mathbf{I}+\mathbf{K}^{[N_{\mathcal{G}}]})^{-1}$. By contrast, numerical experiments (see for example Fig. 3c related to the Airy function example in Sect. 4.2) show that $(\mathbf{I} + \mathbf{K})^{-1}$ is "asymptotically" almost-banded, meaning that it is very well approximated by almost-banded operators of moderate band width. This phenomenon is investigated in Sect. 3.1 from the point of view of integral operators. More specifically, we prove that $(\mathbf{I}+\mathbf{K})^{-1} = \mathbf{I} + \mathbf{R}$, where $\mathbf{R}$ is an integral operator of kernel $\mathfrak{R}(x, t)$ called *resolvent kernel*. This elegant framework is the cornerstone of the Newton–Picard validation algorithm presented in Sect. 3.2.

Contrary to the validation method [2] which performs explicit Picard iterations to construct a contracting operator, our novel approach relies on the Chebyshev spectral method to compute a polynomial approximation of the resolvent kernel $\mathfrak{R}(x, t)$ (Sect. 3.3). The resulting complexity is carefully investigated in Sect. 3.4 and compared with Newton–Galerkin.

3.1 Kernel Composition, Picard Iterations and the Resolvent Kernel

The essence of Newton–Picard validation is to represent an integral operator $\mathbf{K}$ not as an infinite-dimensional matrix in some coefficient space, but using its kernel $\mathfrak{K}(x, t)$ defined over $[x_l, x_r | x_0]$:

$$\begin{aligned} \mathbf{K}\{f\}(x) &= \int_{x_0}^{x} \mathfrak{K}(x, t) f(t) \mathrm{d}t, \qquad f \in \mathcal{C}^0, \\ [x_l, x_r | x_0] &= \{(x, t) \in \mathbb{R}^2 \quad \text{s.t.} \quad x_l \leqslant x \leqslant t \leqslant x_0 \quad \text{or} \quad x_0 \leqslant t \leqslant x \leqslant x_r\}. \end{aligned} \tag{9}$$

The following lemma shows the bijectivity of this correspondence.

Lemma 3.1 *Let* $\mathbf{K}$ *be an integral operator with a continuous kernel* $\mathfrak{K}(x,t)$ *as in* (9). *Then* $\mathbf{K}\{f\} = 0$ *for all* $f \in \mathcal{C}^0$ *if and only if* $\mathfrak{K}(x,t) = 0$ *for all* $(x,t) \in [x_l, x_r | x_0]$.

Proof The "if" part of the proof is straightforward. For the "only-if" part, let $x \in [x_l, x_r]$ and suppose, without loss of generality, that $x \geqslant x_0$. Define $f \in \mathcal{C}^0$ by $f(t) = \mathfrak{K}(x,t)$ for $t \in [x_0, x]$ and extend it constantwise over $[x_l, x_0]$ and $[x, x_r]$. Then:

$$\mathbf{K}\{f\}(x) = \int_{x_0}^{x} \mathfrak{K}(x,t) f(t) \mathrm{d}t = \int_{x_0}^{x} \mathfrak{K}(x,t)^2 \mathrm{d}t = 0,$$

which implies $\mathfrak{K}(x,t) = 0$ for all $t \in [x_0, x]$. Therefore, $\mathfrak{K}(x,t) = 0$ for all $(x,t) \in [x_l, x_r | x_0]$. □

Another nice property of the class of integral operators is its closedness under composition, where the resulting kernel is given by a "convolution-like" operation, denoted by $*$ throughout this article.

Lemma 3.2 *Let* $\mathbf{K}$ *and* $\mathbf{L}$ *be integral operators with respective continuous kernels* $\mathfrak{K}(x,t)$ *and* $\mathfrak{L}(x,t)$. *Then the composition* $\mathbf{K}\,\mathbf{L}$ *is again an integral operator, with kernel* $\mathfrak{K} * \mathfrak{L}$ *given by the associative law:*

$$(\mathfrak{K} * \mathfrak{L})(x,t) = \int_{t}^{x} \mathfrak{K}(x,s)\mathfrak{L}(s,t)\mathrm{d}s, \qquad (x,t) \in [x_l, x_r | x_0].$$

In particular, if $\mathfrak{K}$ *and* $\mathfrak{L}$ *are polynomials, then so is* $\mathfrak{K} * \mathfrak{L}$, *with:*

$$\deg \mathfrak{K} * \mathfrak{L} \leqslant \deg \mathfrak{K} + \deg \mathfrak{L} + 1.$$

Proof The kernel composition formula simply follows from an application of Fubini's theorem:

$$\mathbf{K}\{\mathbf{L}\{f\}\} = \int_{x_0}^{x} \mathfrak{K}(x,s) \int_{x_0}^{s} \mathfrak{L}(s,t) f(t) \mathrm{d}t \mathrm{d}s = \int_{x_0}^{x} \int_{t}^{x} \mathfrak{K}(x,s)\mathfrak{L}(s,t) f(t) \mathrm{d}s \mathrm{d}t = \int_{x_0}^{x} (\mathfrak{K} * \mathfrak{L})(x,t) f(t) \mathrm{d}t.$$

Moreover, since the composition $\circ$ of operators is associative and by injectivity given by Lemma 3.1, we deduce that the operation $*$ of kernel composition is associative too. □

Historically, the expression of $(\mathbf{I}+\mathbf{K})^{-1}$ involving an integral operator arose from the *Picard iterations* [23,31], which build a sequence (f_n) of functions converging to the solution f^* of (2):

$$\begin{cases} f_0 = g, \\ f_{n+1} = g - \mathbf{K}\{f_n\}, \quad n \geqslant 0. \end{cases}$$

This indeed motivates to write $(\mathbf{I}+\mathbf{K})^{-1}$ (at least symbolically) as the Neumann series:

$$(\mathbf{I}+\mathbf{K})^{-1} = \mathbf{I} - \mathbf{K} + \mathbf{K}^2 - \cdots + (-1)^n \mathbf{K}^n + \ldots \tag{10}$$

This matches with the observation that $(\mathbf{I}+\mathbf{K})^{-1}$ is "asymptotically" almost-banded, since $\mathbf{K}^n$ is almost-banded, of band width n times larger than $\mathbf{K}$. The classic Theorem 3.3 below, for which we provide a proof for the sake of completeness, makes it rigorous by establishing the convergence of this series. This gives rise to the *resolvent kernel* $\mathfrak{R}(x,t)$ that plays a central role in Newton–Picard validation method.

Theorem 3.3 *Let* $\mathbf{K}$ *of continuous kernel* $\mathfrak{K}(x,t)$ *act as an endomorphism of* $\mathcal{C}^0$. *Then* $(\mathbf{I}+\mathbf{K})^{-1}=\mathbf{I}+\mathbf{R}$, *where the resolvent kernel* $\mathfrak{R}(x,t)$ *is defined using the iterated kernels* $\mathfrak{K}^{*n}$*:*

$$\mathfrak{R}=\sum_{n=1}^{\infty}(-1)^n\mathfrak{K}^{*n}, \qquad \textit{where} \quad \begin{cases}\mathfrak{K}^{*1}=\mathfrak{K},\\ \mathfrak{K}^{*n+1}=\mathfrak{K}*\mathfrak{K}^{*n}, \quad n\geqslant 1.\end{cases} \tag{11}$$

We have the following estimates:

$$|\mathfrak{R}(x,t)|\leqslant Me^{M|x-t|}, \qquad \textit{for } (x,t)\in[x_l,x_r|x_0],$$
$$\|(\mathbf{I}+\mathbf{K})^{-1}\|_\infty=\|\mathbf{I}+\mathbf{R}\|_\infty\leqslant e^{MT}, \qquad \textit{where } M=\sup_{(x,t)\in[x_l,x_r|x_0]}|\mathfrak{K}(x,t)|.$$

Proof First, we prove by induction on $n\geqslant 1$ that:

$$|\mathfrak{K}^{*n}(x,t)|\leqslant M^n\frac{|x-t|^{n-1}}{(n-1)!} \qquad \text{for } (x,t)\in[x_l,x_r|x_0].$$

This clearly holds for $n=1$. Suppose that this is true for n. Take w.l.o.g. $x_0\leqslant t\leqslant x\leqslant x_r$:

$$|\mathfrak{K}^{*n+1}(x,t)|\leqslant\int_t^x|\mathfrak{K}(x,s)||\mathfrak{K}^{*n}(s,t)|\mathrm{d}s\leqslant\int_t^x M^{n+1}\frac{(s-t)^{n-1}}{(n-1)!}\mathrm{d}s=M^{n+1}\frac{(x-t)^n}{n!}.$$

It follows therefrom that the series (11) defining $\mathfrak{R}(x,t)$ converges in norm, that $\|\mathbf{K}^n\|_\infty\leqslant\frac{(MT)^n}{n!}$, and that the Neumann series (10) makes sense, with the expected estimates for $|\mathfrak{R}(x,t)|$ and $\|\mathbf{I}+\mathbf{R}\|_\infty$. □

The notion of resolvent kernel is closely related to that of Green's function associated to a linear differential operator [31]. The validation methods proposed in [21] and [30] rigorously bound the Green's function to provide a bound on the exact inverse operator associated to $\mathbf{L}$. Our validation algorithm, presented in next section, proceeds differently: it *approximates* the resolvent kernel and *rigorously bounds* the operator norm of the resulting Newton-like operator $\mathbf{T}$. This allows for sharper bounds and a more convenient complexity analysis.

3.2 The Validation Algorithm

Algorithm NewtonPicardValid below implements the three steps of the Newton-like a posteriori validation scheme presented in Sect. 2.1, by approximating the inverse derivative $(\mathbf{I}+\mathbf{K})^{-1}$ as $\mathbf{I}+\mathbf{R}^\circ$, where $\mathbf{R}^\circ$ is an integral operator of polynomial kernel $\mathfrak{R}^\circ(x,t)$ approximating the analytic resolvent kernel $\mathfrak{R}(x,t)$. Similarly to the truncation index $N_{\mathcal{G}}$ in Newton–Galerkin validation procedure, a parameter $N_{\mathfrak{R}}$ (the approximation degree for $\mathfrak{R}^\circ(x,t)$) is passed to the validation algorithm. In general, $N_{\mathfrak{R}}$ is at least $r+s$, where r is the order of $\mathbf{L}$ and $s=\max_{0\leqslant i<r}\deg a_i$, so that we make the assumption $r+s=\mathcal{O}(N_{\mathfrak{R}})$ in the complexity estimates.

1. First, the subroutine ResolventKernel (postponed in Sect. 3.3) computes degree $N_{\mathfrak{R}}$ polynomials α_i°, β_i° and sets:

$$\mathfrak{R}^\circ(x,t)=\sum_{i=0}^{r-1}\alpha_i^\circ(x)\beta_i^\circ(t)\approx\mathfrak{R}(x,t), \qquad \deg\alpha_i^\circ,\deg\beta_i^\circ\leqslant N_{\mathfrak{R}}. \tag{12}$$

This defines the Newton–Picard fixed-point operator $\mathbf{T}\{f\}=f-(\mathbf{I}+\mathbf{R}^\circ)\{f+\mathbf{K}\{f\}-g\}$.

2. The linear part of affine operator $\mathbf{T}$:

$$\mathbf{E} = \mathbf{I} - (\mathbf{I} + \mathbf{R}^\circ)(\mathbf{I} + \mathbf{K}) = -\mathbf{K} - \mathbf{R}^\circ - \mathbf{R}^\circ\mathbf{K},$$

is an integral operator whose polynomial kernel $\mathfrak{E} = -\mathfrak{K} - \mathfrak{R}^\circ - \mathfrak{R}^\circ * \mathfrak{K}$ is explicitly computed as $\sum_{i,j} \varepsilon_{ij} T_i(x) T_j(t)$. This yields the bound λ:

$$\|\mathbf{E}\|_\infty = \sup_{x \in I} \ \operatorname{sgn}(x - x_0) \int_{x_0}^{x} |\mathfrak{E}(x,t)| \mathrm{d}t \leqslant T \sum_{i,j} |\varepsilon_{ij}| = \lambda.$$

3. If $\lambda \geqslant 1$, then go back to step 1 using a larger approximation degree $N_{\mathfrak{R}}$. Otherwise, compute and bound the defect as d, using, for instance, the $\mathrm{Ч}^1$ norm (or any other more refined, yet rigorous overestimation of the uniform norm):

$$\|\mathbf{T}\{f^\circ\} - f^\circ\|_\infty \leqslant \|(\mathbf{I} + \mathbf{R}^\circ)\{f^\circ + \mathbf{K}\{f^\circ\} - g\}\|_{\mathrm{Ч}^1} = d,$$

and return $d/(1 - \lambda)$.

For the sake of efficiency, kernels are represented using *low-rank decompositions*.

Definition 3.4 A bivariate kernel $\mathfrak{K}(x,t)$ is said to be of rank k if there are functions $\alpha_0(x), \dots, \alpha_{k-1}(x)$ and $\beta_0(t), \dots, \beta_{k-1}(t)$ over $[x_l, x_r]$ such that:

$$\mathfrak{K}(x,t) = \sum_{i=0}^{k-1} \alpha_i(x)\beta_i(t), \qquad (x,t) \in [x_l, x_r | x_0].$$

The kernel $\mathfrak{K}(x,t)$ in Propositions 2.3 and 2.5 is clearly of rank r. Also, we shall see in the next section that the resolvent kernel $\mathfrak{R}(x,t)$ is of rank r too (which is not obvious from its definition using iterated kernels in Theorem 3.3). This explains why we compute a $\mathfrak{R}^\circ(x,t)$ of rank r to approximate $\mathfrak{R}(x,t)$ in (12). Therefore, elementary operations on low rank kernels (evaluation on a polynomial and composition with $*$) are necessary to implement these three steps in Algorithm NEWTONPICARDVALID. Their complexity is investigated in the subsequent Lemmas 3.7 and 3.8.

Theorem 3.5 *Let f° be a degree n polynomial approximation of the solution f^* of the integral equation of Proposition 2.3 or 2.5, and $N_{\mathfrak{R}}$ the degree used for $\mathfrak{R}^\circ(x,t)$. If* NEWTONPICARDVALID$(\mathfrak{K}, g, f^\circ, N_{\mathfrak{R}})$ *does not fail, then it outputs a rigorous error bound for $\|f^\circ - f^*\|$. More precisely:*

- *It computes a Newton–Picard fixed-point operator* $\mathbf{T}$ *and a rigorous contraction ratio λ in*

  $$\mathcal{O}(N_{\mathfrak{R}}^2 r^2) \quad \textit{arithmetic operations.}$$

- *Given* $\mathbf{T}$ *and λ, it computes for any right-hand side g and any approximation f° a rigorous error bound for $\|f^\circ - f^*\|$ in*

  $$\mathcal{O}((N_{\mathfrak{R}} + n + m) N_{\mathfrak{R}} r) \quad \textit{arithmetic operations,}$$

  where $n = \deg f^\circ$ and $m = \deg g$.

Algorithm 1 NewtonPicardValid($\mathfrak{K}$, g, f°, $N_{\mathfrak{R}}$)

Input: The polynomial kernel $\mathfrak{K}(x,t)$ and polynomial right-hand side $g(x)$ coming from $\mathbf{L}\{y\} = h$ using Proposition 2.3 or 2.5, a polynomial approximation $f^\circ(x)$ and a validation degree $N_{\mathfrak{R}}$.
Output: A rigorous error bound $\varepsilon \geqslant \|f^\circ - f^*\|_\infty$.

```
▷ Step 1: approximating the resolvent kernel $\mathfrak{R}(x,t)$ using floating-point arithmetics
1: Write $\mathfrak{K}(x,t)$ as $\begin{cases} \sum_{i=0}^{r-1} \kappa_i(x)T_i(t) & \text{if } \mathbf{L} \text{ is in } \mathbf{x}-\boldsymbol{\partial} \text{ form} \\ \text{or } -\sum_{i=0}^{r-1} T_i(x)\kappa_i(t) & \text{if } \mathbf{L} \text{ is in } \boldsymbol{\partial}-\mathbf{x} \text{ form} \end{cases}$
2: $\mathfrak{R}^\circ(x,t) = \sum_{i=0}^{r-1} \alpha_i^\circ(x)\beta_i^\circ(t) \leftarrow$ ResolventKernel($\mathfrak{K}$, $N_{\mathfrak{R}}$)
▷ Step 2: Bounding the contraction ratio of T using interval arithmetics
3: $\mathfrak{M}(x,t) \leftarrow$ KernelComp($\mathfrak{R}^\circ$, $\mathfrak{K}$)
4: $\mathfrak{E}(x,t) = \sum_{i,j} \varepsilon_{ij} T_i(x) T_j(t) \leftarrow -\mathfrak{R}^\circ(x,t) - \mathfrak{K}(x,t) - \mathfrak{M}(x,t)$
5: $\lambda \leftarrow T \sum_{i,j} |\varepsilon_{ij}|$
▷ Step 3: Compute the final bound using interval arithmetics
6: if $\lambda < 1$ then
7:     $\delta(x) \leftarrow f^\circ(x) + \mathbf{K}\{f^\circ\}(x) - g(x)$
8:     $\delta(x) \leftarrow \delta(x) + \mathbf{R}^\circ\{\delta\}(x)$
9:     $d \leftarrow \|\delta\|_{\mathcal{U}^1}$
10:    return $\frac{d}{1-\lambda}$
11: else
12:    return "FAIL"
13: end if
```

Remark 3.6 Theorem 3.5 gives a separate complexity estimate for Step 3, since Steps 1 and 2 are completely independent of f° and g. This means that the same Newton–Picard fixed-point validation operator $\mathbf{T}$ can be reused to validate another approximation f° in Problem 1.1, possibly with a different right-hand side h and/or different initial conditions v_i.

Lemma 3.7 *If* $\mathbf{K}$ *is an integral operator with polynomial kernel* $\mathfrak{K}(x,t) = \sum_{i=0}^{k-1} \alpha_i(x)\beta_i(t)$ *of rank* k *and total degree* n*, and* $f(x)$ *a univariate polynomial of degree* m*, then* $\mathbf{K}\{f\} = \int_{x_0}^{x} \mathfrak{K}(x,t) f(t)\mathrm{d}t$ *is a polynomial of degree at most* $n+m+1$ *that can be computed in* $\mathcal{O}(kn(n+m))$ *arithmetic operations.*

Proof For each $0 \leqslant i < k$, $\alpha_i(x)\int_{x_0}^{x} \beta_i(t)f(t)\mathrm{d}t$ is a univariate polynomial of degree at most $\deg \alpha_i + \deg \beta_i + \deg f + 1 \leqslant n+m+1$, that requires $\mathcal{O}(\deg \beta_i \deg f)$ arithmetic operations for the inner multiplication, $\mathcal{O}(\deg \beta_i + \deg f)$ ones for the integration, and finally $\mathcal{O}(\deg \alpha_i(\deg \beta_i + \deg f))$ ones for the outer multiplication, giving a total of $\mathcal{O}(n(n+m))$ arithmetic operations. The claimed overall complexity follows from the iteration for each $0 \leqslant i < k$ and the additions. □

Lemma 3.8 *Let* $\mathfrak{K}(x,t) = \sum_{i=0}^{k-1} \alpha_i(x)\beta_i(t)$ *and* $\mathfrak{L}(x,t) = \sum_{j=0}^{l-1} \gamma_j(x)\delta_j(t)$ *be polynomial kernels of respective ranks* k *and* l*, and respective total degrees* n *and* m*. Then* KernelComp($\mathfrak{K}$, $\mathfrak{L}$) *computes a rank* $k+l$ *decomposition of* $\mathfrak{K} * \mathfrak{L}$ *with degree at most* $n+m+1$*, in* $\mathcal{O}(kl(n+m)^2)$ *arithmetic operations.*

Algorithm 2 KernelComp($\mathfrak{K}$, $\mathfrak{L}$)

Input: Polynomial kernels $\mathfrak{K}(x,t) = \sum_{i=0}^{k-1} \alpha_i(x)\beta_i(t)$ and $\mathfrak{L}(x,t) = \sum_{j=0}^{l-1} \gamma_j(x)\delta_j(t)$.

Output: Polynomial kernel $\mathfrak{M}(x,t) = \sum_{i=0}^{k+l-1} \mu_i(x)\nu_i(t)$ equal to $\mathfrak{K} * \mathfrak{L}$.

```
1: for $i = 0$ to $k-1$ and $j = 0$ to $l-1$ do $\zeta_{ij}(s) \leftarrow \int \beta_i(s)\gamma_j(s)\mathrm{d}s$ end for
2: for $i = 0$ to $k-1$ do
3:     $\mu_i(x) \leftarrow \alpha_i(x)$
4:     $\nu_i(t) \leftarrow -\sum_{j=0}^{l-1} \zeta_{ij}(t)\delta_j(t)$
5: end for
6: for $j = 0$ to $l-1$ do
7:     $\mu_{k+j}(x) \leftarrow \sum_{i=0}^{k-1} \alpha_i(x)\zeta_{ij}(x)$
8:     $\nu_{k+j}(t) \leftarrow \delta_j(t)$
9: end for
10: return $\mathfrak{M}(x,t) \leftarrow \sum_{i=0}^{k+l-1} \mu_i(x)\nu_i(t)$
```

Proof Algorithm KernelComp first computes primitives in line 1:

$$\zeta_{ij}(s) = \int \beta_i(s)\gamma_j(s)\mathrm{d}s, \qquad 0 \leqslant i < k, \quad 0 \leqslant j < l,$$

in $\mathcal{O}(klnm)$ arithmetic operations, giving polynomials of degree at most $n+m+1$.

The computation of $\mathfrak{M}(x,t)$ then follows:

$$\begin{aligned}(\mathfrak{K} * \mathfrak{L})(x,t) &= \int_t^x \mathfrak{K}(x,s)\mathfrak{L}(s,t)\mathrm{d}s = \sum_{i=0}^{k-1}\sum_{j=0}^{l-1} \alpha_i(x)(\zeta_{ij}(x) - \zeta_{ij}(t))\delta_j(t) \\ &= \sum_{j=0}^{l-1} \underbrace{\left[\sum_{i=0}^{k-1} \alpha_i(x)\zeta_{ij}(x)\right]}_{\mu_{k+j}(x)} \underbrace{\delta_j(t)}_{\nu_{k+j}(t)} - \sum_{i=0}^{k-1} \underbrace{\alpha_i(x)}_{\mu_i(x)} \underbrace{\left[\sum_{j=0}^{l-1} \zeta_{ij}(t)\delta_j(t)\right]}_{\nu_i(t)}.\end{aligned}$$

More precisely, $\nu_i(t)$ (line 4) requires $\mathcal{O}(l(n+m)m)$ arithmetic operations, and similarly $\mu_{k+j}(x)$ (line 7) requires $\mathcal{O}(kn(n+m))$ ones. This concludes the proof of the claimed complexity. □

Proof of Theorem 3.5 First, for step 1, rewriting $\mathfrak{K}(x,t)$ given in Propositions 2.3 or 2.5 as a rank r expression (line 1) requires $\mathcal{O}(r^2 s)$ arithmetic operations. Then, anticipating the complexity estimate for ResolventKernel given in Proposition 3.13, line 2 requires $\mathcal{O}(N_{\mathfrak{R}}(r+s)^2)$ arithmetic operations. Moving on to step 2, the kernel composition in line 3 runs in complexity $\mathcal{O}((N_{\mathfrak{R}}+r+s)^2 r^2)$, according to Lemma 3.8. After that, the computation of a bound λ for $\mathfrak{E}(x,t)$ (lines 4 and 5) runs in $\mathcal{O}((N_{\mathfrak{R}}+r+s)^2 r)$. This justifies the overall complexity for steps 1 and 2, assuming $r+s = \mathcal{O}(N_{\mathfrak{R}})$.

Concerning step 3, lines 7 and 8 computing $\delta(x)$ respectively require $\mathcal{O}(r(r+s)(r+s+n)+m)$ and $\mathcal{O}(2N_{\mathfrak{R}}(2N_{\mathfrak{R}}+r+s+n+m)r)$ arithmetic operations, using Lemma 3.7. Finally, the bound $d \geqslant \|\delta\|_{\mathrm{Y}^1} \geqslant \|\delta\|_\infty$ computed line 9 requires $\deg \delta(x) = \mathcal{O}(2N_{\mathfrak{R}}+r+s+n+m)$ arithmetic operations. Thus, $\mathcal{O}((N_{\mathfrak{R}}+n+m)N_{\mathfrak{R}}r)$ is a compact majorant for the complexity, assuming $r+s = \mathcal{O}(N_{\mathfrak{R}})$. □

Remark 3.9 The total complexity can be made quasi-linear w.r.t. $N_{\mathfrak{R}}$ by using a fast DCT-based algorithm for polynomial multiplication in the kernel composition (line 3), and bounding the low-rank kernel $\mathfrak{E}(x,t)$ (lines 4–5) using an adequate orthogonalization process (to reduce overestimation). However, as explained in Remark 2.6, we choose to keep the quadratic estimate in this article.

Remark 3.10 The use of Chebyshev approximations for $\mathfrak{R}^\circ(x,t)$ is not crucial to the principle of the validation algorithm. The reason of this choice is that they are asymptotically excellent candidates and allow for an elegant complexity analysis in Sect. 3.4. However, for intermediate precision, other approximants (e.g., splines) may perform better.

3.3 Resolvent Kernel Approximation

Algorithm NEWTONPICARDVALID above requires a polynomial approximation $\mathfrak{R}^\circ(x,t)$ of $\mathfrak{R}(x,t)$. We first show in Sect. 3.3.1 that $\mathfrak{R}$ admits an analytic rank r expression, involving the solutions of the homogeneous linear ODE $\mathbf{L}\{y\}=0$, together with those of the *adjoint* equation $\mathbf{L}^*\{y\}=0$. The adjunction $\mathbf{L}\mapsto\mathbf{L}^*$, defined as:

$$\mathbf{x}^*=\mathbf{x},\qquad \boldsymbol{\partial}^*=-\boldsymbol{\partial},\qquad (\mathbf{L}_1\mathbf{L}_2)^*=\mathbf{L}_2^*\mathbf{L}_1^*,\qquad (\mathbf{L}_1+\mathbf{L}_2)^*=\mathbf{L}_1^*+\mathbf{L}_2^*,$$

plays a crucial role to this purpose. Clearly, the $\mathbf{x}-\boldsymbol{\partial}$ form differential operator in (3) and the one in $\boldsymbol{\partial}-\mathbf{x}$ form in (4) are mutually adjoint. For the sake of clarity, we opt for the convention to refer to the former as $\mathbf{L}$, and the latter as $\mathbf{L}^*$. We also call $\mathbf{K}$ and $\mathfrak{K}(x,t)$ the integral operator and kernel associated to $\mathbf{L}$ by Proposition 2.3, and similarly $\mathbf{K}^*$ and $\mathfrak{K}^*(x,t)$ for $\mathbf{L}^*$ in Proposition 2.5. Note the following adjunction relation on integral operators[5]:

$$\mathfrak{K}^*(x,t)=-\mathfrak{K}(t,x),\qquad (x,t)\in[x_l,x_r]^2.$$

The formulas given in Proposition 3.12 below for the resolvent kernels $\mathfrak{R}$ and $\mathfrak{R}^*$ associated to $\mathfrak{K}$ and $\mathfrak{K}^*$ also satisfy $\mathfrak{R}^*(x,t)=-\mathfrak{R}(t,x)$.

Based on this, Sect. 3.3.2 presents the approximation routine RESOLVENTKERNEL which numerically solves the homogeneous equations using the Chebyshev spectral method presented in Sect. 2.3. This completes the algorithmic description of the Newton–Picard validation procedure.

3.3.1 Formulas for the Resolvent Kernel

The scalar linear ODE (3) can be transformed into the equivalent r-dimensional first-order vectorial linear ODE $Y'(x)=A(x)Y(x)+H(x)$, where:

$$Y(x)=\begin{pmatrix} y(x)\\ y'(x)\\ \vdots\\ \vdots\\ y^{(r-1)}(x)\end{pmatrix},\qquad A(x)=\begin{pmatrix} 0 & 1 & 0 & \cdots & 0\\ 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \cdots & 1\\ -a_0(x) & -a_1(x) & -a_2(x) & \cdots & -a_{r-1}(x)\end{pmatrix},\qquad H(x)=\begin{pmatrix} 0\\ \vdots\\ \vdots\\ 0\\ h(x)\end{pmatrix}.$$

[5] The adjunction relation requires the kernels to be defined over $[x_l,x_r]^2$ rather than $[x_l,x_r|x_0]$ only, which is always the case in this article.

It is well known that the solution with initial conditions $v = (v_0, \dots, v_{r-1})^T$ can be expressed using an integral operator with kernel $\Phi(x)\Phi(t)^{-1}$:

$$Y(x) = \Phi(x) \left[v + \int_{x_0}^{x} \Phi(t)^{-1} H(t) \mathrm{d}t \right],$$

where $\Phi(x)$ is the *fundamental matrix*, satisfying the matrix ODE $\Phi'(x) = A(x)\Phi(x)$ with $\Phi(x_0) = I_r$. In other words, $\Phi(x)$ contains the canonical basis of solutions for the homogeneous linear ODE $\mathbf{L}\{y\} = 0$ and their derivatives up to order $r - 1$. In particular, its columns are linearly independent for all x. Hence, $\Phi(x)$ is invertible and we have:

$$(\Phi(x)^{-1})' = -\Phi(x)^{-1}\Phi'(x)\Phi(x)^{-1} = -\Phi(x)^{-1}A(x)\Phi(x)\Phi(x)^{-1} = -\Phi(x)^{-1}A(x),$$

which means that the inverse transpose $x \mapsto \Phi(x)^{-T}$ satisfies the matrix ODE $M'(x) = -A(x)^T M(x)$.

On the other hand, the scalar linear ODE $\mathbf{L}^*\{y\} = h$ with $\mathbf{L}^*$ given in $\boldsymbol{\partial} - \mathbf{x}$ form as in (4) is equivalent to the vectorial ODE $Y'(x) = B(x)Y(x) + H(x)$, where:

$$Y(x) = \begin{pmatrix} y(x) \\ y^{[1]}(x) \\ \vdots \\ \vdots \\ y^{[r-1]}(x) \end{pmatrix}, \qquad B(x) = \begin{pmatrix} a_{r-1}(x) & -1 & 0 & \dots & 0 \\ a_{r-2}(x) & 0 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1(x) & 0 & 0 & \dots & -1 \\ a_0(x) & 0 & 0 & \dots & 0 \end{pmatrix}, \qquad H(x) = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ -h(x) \end{pmatrix}.$$

Similarly, the fundamental matrix $\Psi(x)$ is defined as the solution of $\Psi'(x) = B(x)\Psi(x)$ with $\Psi(x_0) = I_r$. A key observation is that:

$$J_r B(x) J_r = -A(x)^T, \qquad \text{where} \quad J_r = \begin{pmatrix} 0 & \dots & 1 \\ \vdots & ⋰ & \vdots \\ 1 & \dots & 0 \end{pmatrix}, \tag{13}$$

leading to the following crucial orthogonality relations.

Lemma 3.11 *The canonical basis $\varphi_0, \dots, \varphi_{r-1}$ for the solution space of $\mathbf{L}$:*

$$\mathbf{L}\{\varphi_i\} = 0, \qquad \varphi_i^{(j)}(x_0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise}, \end{cases}$$

and the dual basis $\psi_0, \dots, \psi_{r-1}$ for the solution space of the adjoint $\mathbf{L}^*$:

$$\mathbf{L}^*\{\psi_i\} = 0, \qquad \psi_i^{[j]}(x_0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise}, \end{cases}$$

satisfy the orthogonality relations for $0 \leqslant j, k < r$:

$$\sum_{i=0}^{r-1} \varphi_i^{(j)}(x)\, \psi_{r-1-i}^{[k]}(x) = \begin{cases} 1 & \text{if } k + j = r - 1, \\ 0 & \text{otherwise}, \end{cases} \qquad \text{for all } x \in [x_l, x_r]. \tag{14}$$

Proof The matrices $\Phi(x) = (\varphi_j^{(i)}(x))_{0 \leqslant i,j<r}$ and $\Psi(x) = (\psi_j^{[i]}(x))_{0 \leqslant i,j<r}$ satisfy $\Phi(x_0) = \Psi(x_0) = I_r$, and are solutions to the matrix linear ODEs $\Phi'(x) = A(x)\Phi(x)$ and $\Psi'(x) = B(x)\Psi(x)$, respectively. From identity (13), we have that:

$$(J_r\Psi(x)J_r)' = J_r B(x)\Psi(x)J_r = -A(x)^T(J_r\Psi(x)J_r).$$

Hence, $x \mapsto J_r\Psi(x)J_r$ satisfies the same ODE $M'(x) = -A(x)^T M(x)$ as $x \mapsto \Phi(x)^{-T}$, and both are equal to I_r at x_0. It follows therefrom that $\Phi(x)^{-T} = J_r\Psi(x)J_r$, and in particular:

$$\Phi(x)J_r\Psi(x)^T J_r = I_r \qquad \text{for all } x \in [x_l, x_r],$$

which unfolds as the orthogonality relations (14). □

Proposition 3.12 *The resolvent kernels $\mathfrak{R}$ and $\mathfrak{R}^*$ are given by:*

$$\mathfrak{R}(x,t) = \sum_{i=0}^{r-1} \varphi_i^{(r)}(x)\psi_{r-1-i}(t), \tag{15}$$

$$\mathfrak{R}^*(x,t) = -\sum_{i=0}^{r-1} \psi_i(x)\varphi_{r-1-i}^{(r)}(t), \tag{16}$$

with the φ_i, ψ_i defined in Lemma 3.11.

Proof • For $\mathfrak{R}$, call $\mathfrak{R}'(x,t)$ the right-hand side of (15) and $\mathbf{R'}$ the integral operator defined by this kernel. Since, by Theorem 3.3, $\mathbf{I}+\mathbf{K}$ is already known to be invertible, of inverse $\mathbf{I}+\mathbf{R}$, it is sufficient to prove that $(\mathbf{I}+\mathbf{K})(\mathbf{I}+\mathbf{R'}) = \mathbf{I}$, that is $\mathfrak{K} * \mathfrak{R}' + \mathfrak{R}' + \mathfrak{K} = 0$. For $0 \leqslant i < r$, $\mathbf{L}\{\varphi_i\} = 0$ implies from Proposition 2.3 by renaming x_0 into t:

$$\varphi_i^{(r)}(x) + \int_t^x \mathfrak{K}(x,s)\varphi_i^{(r)}(s)\mathrm{d}s = -\sum_{j=0}^{r-1} a_j(x) \sum_{k=j}^{r-1} \frac{(x-t)^{k-j}}{(k-j)!}\varphi_i^{(k)}(t).$$

Multiplying the left-hand side by $\psi_{r-1-i}(t)$ and summing over i gives $\mathfrak{R}'(x,t) + (\mathfrak{K} * \mathfrak{R}')(x,t)$. Doing the same operations on the right-hand side and commuting the summation symbols allows us to isolate the term $\sum_{i=0}^{r-1} \varphi_i^{(k)}(t)\psi_{r-1-i}(t)$, which by Lemma 3.11 is nonzero (and equal to 1) if and only if $k = r-1$. We thus obtain the expected equality:

$$\mathfrak{R}'(x,t) + (\mathfrak{K} * \mathfrak{R}')(x,t) = -\sum_{j=0}^{r-1} a_j(x)\frac{(x-t)^{r-1-j}}{(r-1-j)!} = -\mathfrak{K}(x,t).$$

• For $\mathfrak{R}^*$, call $\mathfrak{R}''(x,t)$ the right-hand side of (16). Clearly, $\mathfrak{R}''(x,t) = -\mathfrak{R}(t,x)$. We have:

$$\begin{aligned}
&(\mathfrak{K}^* * \mathfrak{R}'')(x,t) + \mathfrak{R}''(x,t) + \mathfrak{K}^*(x,t)\\
&= \int_t^x \mathfrak{K}^*(x,s)\mathfrak{R}''(s,t)\mathrm{d}s + \mathfrak{R}''(x,t) + \mathfrak{K}^*(x,t)\\
&= \int_t^x \mathfrak{K}(s,x)\mathfrak{R}(t,s)\mathrm{d}s - \mathfrak{R}(t,x) - \mathfrak{K}(t,x) = -(\mathfrak{R} * \mathfrak{K})(t,x) - \mathfrak{R}(t,x) - \mathfrak{K}(t,x) = 0,
\end{aligned}$$

since $-\mathbf{R}\,\mathbf{K} - \mathbf{R} - \mathbf{K} = \mathbf{I} - (\mathbf{I}+\mathbf{R})(\mathbf{I}+\mathbf{K}) = 0$. Hence the corresponding operator $\mathbf{R}''$ satisfies $(\mathbf{I}+\mathbf{K^*})(\mathbf{I}+\mathbf{R''}) = \mathbf{I}$, which implies $\mathbf{R''} = \mathbf{R^*}$ and hence $\mathfrak{R}'' = \mathfrak{R}^*$. □

3.3.2 Resolvent Kernel Approximation Algorithm

A major advantage of the formulas in Proposition 3.12 from the computational point of view, is that in both $\mathbf{x}-\boldsymbol{\partial}$ and $\boldsymbol{\partial}-\mathbf{x}$ cases, one requires the ψ_i, solutions to an ODE in $\boldsymbol{\partial}-\mathbf{x}$ form, and the $\varphi_i^{(r)}$ where the φ_i are solutions to an ODE in $\mathbf{x}-\boldsymbol{\partial}$ form. Hence, no conversion from $\mathbf{x}-\boldsymbol{\partial}$ and $\boldsymbol{\partial}-\mathbf{x}$ forms is needed here.

Algorithm ResolventKernel below computes degree $N_{\mathfrak{R}}$ polynomial approximations of these analytic functions. To do so, the Chebyshev spectral method of Sect. 2.3 is applied on the following integral equations, equivalent by Propositions 2.3 and 2.5 to the IVPs defining φ_i and ψ_i in Lemma 3.11:

$$\begin{aligned}
&(\mathbf{I}+\mathbf{K})\{\varphi_i^{(r)}\} = g_i, &&\text{with}\quad g_i(x) = -\sum_{j=0}^{i} a_j(x)\frac{(x-x_0)^{i-j}}{(i-j)!},\\
&(\mathbf{I}+\mathbf{K}^*)\{\psi_i\} = g_i^*, &&\text{with}\quad g_i^*(x) = \frac{(x_0-x)^i}{i!}.
\end{aligned}\tag{17}$$

Algorithm 3 ResolventKernel($\mathfrak{K}$, $N_{\mathfrak{R}}$)

Input: Kernel $\mathfrak{K}(x,t)$ obtained from $\mathbf{L}$ using Proposition 2.3 or 2.5, and approximation degree $N_{\mathfrak{R}}$.

Output: Polynomial kernel $\mathfrak{R}^\circ(x,t) = \sum_{i=0}^{r-1} \alpha_i^\circ(x)\beta_i^\circ(t)$ approximating the resolvent kernel $\mathfrak{R}(x,t)$.

```
1: $\mathfrak{K}^*(x,t) \leftarrow -\mathfrak{K}(t,x)$
2: if $\mathbf{L}$ is in $\boldsymbol{\partial}-\mathbf{x}$ form then
3:     $\mathfrak{K}(x,t) \leftrightarrow \mathfrak{K}^*(x,t)$      ▷ Exchange the role of $\mathbf{L}$ and its adjoint $\mathbf{L}^*$
4: end if
▷ Compute approximations $\varphi_i^{(r)\circ}$ to the $\varphi_i^{(r)}$ associated to $\mathbf{L}\{\varphi\}=0$
5: $M \leftarrow I_{N_{\mathfrak{R}}+1} + \mathbf{K}^{[N_{\mathfrak{R}}]}$
6: $(Q, R) \leftarrow$ AlmostBandedQRFactor($M$)
7: for $i = 0$ to $r-1$ do
8:     $g_{ii} \leftarrow -a_i$
9:     $g_{ij} \leftarrow g_{i-1,j}\frac{x-x_0}{i-j}$    for $j = 0$ to $i-1$
10:    $g_i \leftarrow g_{i0} + \cdots + g_{ii}$
11:    $\varphi_i^{(r)\circ} \leftarrow$ AlmostBandedBackSubs($Q$, $R$, $g_i$)
12: end for
▷ Compute approximations $\psi_i^\circ$ to the $\psi_i$ associated to $\mathbf{L}^*\{\psi\}=0$
13: $M^* \leftarrow I_{N_{\mathfrak{R}}+1} + \mathbf{K}^{*[N_{\mathfrak{R}}]}$
14: $(Q^*, R^*) \leftarrow$ AlmostBandedQRFactor($M^*$)
15: $g_0^* \leftarrow 1$
16: for $i = 0$ to $r-1$ do
17:    $\psi_i^\circ \leftarrow$ AlmostBandedBackSubs($Q^*$, $R^*$, $g_i^*$)
18:    $g_{i+1}^* \leftarrow g_i^*\frac{x_0-x}{i+1}$
19: end for
▷ Return polynomial kernel $\mathfrak{R}^\circ(x,t)$
20: if $\mathbf{L}$ is in $\mathbf{x}-\boldsymbol{\partial}$ form then
21:    $(\alpha_i^\circ, \beta_i^\circ) \leftarrow (\varphi_i^{(r)\circ}, \psi_{r-1-i}^\circ)$    for $i = 0$ to $r-1$
22: else if $\mathbf{L}$ is in $\boldsymbol{\partial}-\mathbf{x}$ form then
23:    $(\alpha_i^\circ, \beta_i^\circ) \leftarrow (-\psi_i^\circ, \varphi_{r-1-i}^{(r)\circ})$    for $i = 0$ to $r-1$
24: end if
25: return $\mathfrak{R}^\circ(x,t) \leftarrow \sum_{i=0}^{r-1} \alpha_i^\circ(x)\beta_i^\circ(t)$
```

Proposition 3.13 *Algorithm* ResolventKernel *computes a rank r, degree $2N_{\mathfrak{R}}$ polynomial kernel $\mathfrak{R}^\circ(x,t)$ approximating the resolvent kernel $\mathfrak{R}(x,t)$ in $\mathcal{O}(N_{\mathfrak{R}}(r+s)^2)$ arithmetic operations.*

Proof First, the computation of M (line 5) and M^* (line 13) only requires $\mathcal{O}(N_{\mathfrak{R}} r(r+s))$ arithmetic operations, due to the specific form of $\mathfrak{K}(x,t)$ built in line 1 of Algorithm NewtonPicardValid. Indeed, for $0 \leqslant i < r$ and $0 \leqslant j \leqslant N_{\mathfrak{R}}$, the computation of $\kappa_i(x) \int_{x_0}^x T_i(t)T_j(t)\mathrm{d}t$ or $T_i(t)\int_{x_0}^x \kappa_i(t)T_j(t)\mathrm{d}t$ involves two multiplications where at most one operand has $\mathcal{O}(r+s)$ nonzero coefficients ($\mathcal{O}(1)$ for the other one), and one integral where the integrand has at most $\mathcal{O}(r+s)$ nonzero coefficients. Hence, for $0 \leqslant j \leqslant N_{\mathfrak{R}}$, the nonzero coefficients of $\mathbf{K}\{T_j\}$ and $\mathbf{K}^*\{T_j\}$ (forming the horizontal and diagonal bands of M and M^*) are computed in $\mathcal{O}(r(r+s))$ arithmetic operations each.

Then, according to Proposition 2.9, the computation of the QR factorizations in lines 6 and 14 using Algorithm AlmostBandedQRFactor requires $\mathcal{O}(N_{\mathfrak{R}}(r+s)^2)$ arithmetic operations.

After that, for each $0 \leqslant i < r$, the computation of the right-hand side g_i (lines 8–10) requires $\mathcal{O}(r(r+s))$ arithmetic operations, while the updating of the right-hand side g_i^* of the adjoint equation (line 18) needs $\mathcal{O}(r+s)$ ones. Finally, solving the resulting almost-banded linear systems using AlmostBandedBackSubs in lines 11 and 17 with the previously computed QR factorizations requires $\mathcal{O}(N_{\mathfrak{R}}(r+s))$ arithmetic operations. Iterating over $0 \leqslant i < r$, this yields a complexity of $\mathcal{O}(N_{\mathfrak{R}} r(r+s))$ for lines 7–12 and 16–19, assuming $r+s = \mathcal{O}(N_{\mathfrak{R}})$.

Hence, the overall complexity of ResolventKernel is $\mathcal{O}(N_{\mathfrak{R}}(r+s)^2)$. □

Remark 3.14 Instead of computing a fundamental matrix for the adjoint equation $\mathbf{L}^*\{y\} = 0$ (lines 16–19 in ResolventKernel), we can also directly approximate the inverse of the fundamental matrix of $\mathbf{L}\{y\} = 0$ (already computed in lines 7–12). To do so, sample the fundamental matrix at $N_{\mathfrak{R}} + 1$ Chebyshev nodes (with complexity $\mathcal{O}(N_{\mathfrak{R}} \log N_{\mathfrak{R}} r^2)$ using fast DCT), invert the matrix at each node (with a total of $\mathcal{O}(N_{\mathfrak{R}} r^\omega)$ arithmetic operations), and finally re-interpolate in complexity $\mathcal{O}(N_{\mathfrak{R}} \log N_{\mathfrak{R}} r^2)$.

3.4 Newton–Galerkin vs Newton–Picard Validation Complexity

The overall complexity of NewtonPicardValid depends on how large the degree $N_{\mathfrak{R}}$ of the approximate resolvent kernel $\mathfrak{R}^\circ(x,t)$ has to be to ensure a contracting Newton–Picard operator $\mathbf{T}$. This, in turn, is determined by the approximation errors $\varepsilon_i^{(N_{\mathfrak{R}})} = \|\varphi_i^{(r)\circ} - \varphi_i^{(r)}\|_\infty$ and $\eta_i^{(N_{\mathfrak{R}})} = \|\psi_i^\circ - \psi_i\|_\infty$ of the $\varphi_i^{(r)\circ}$ and ψ_i° computed by ResolventKernel.

The use of spectral methods is motivated by both efficiency and accuracy, and in practice the resulting error is close to the one from the mere truncation of the exact Chebyshev series. In the analytic framework we consider (where the coefficients of $\mathbf{L}$ are polynomials or analytic functions over a complex neighborhood of $[x_l, x_r]$, represented using RPAs), Chebyshev series enjoy far better convergence properties than the simple summability condition defining the Ч^1 space. Let $\mathcal{E}_\rho$ denote the *Bernstein ellipse* [36, Chap. 8] of parameter $\rho > 1$ rescaled over $[x_l, x_r]$, i.e. the ellipse in the complex plane of foci x_l and x_r and eccentricity $2/(\rho+\rho^{-1})$. It is the typical domain of convergence of Chebyshev series, which play the same role as the convergence disks do for Taylor series. The following lemma (see e.g. [36, Thm. 8.1]) asserts the exponential decay of Chebyshev coefficients for analytic functions.

Lemma 3.15 *Let f be an analytic function over $\mathcal{E}_\rho$. Then its Chebyshev coefficients $[f]_n$ satisfy:*

$$|[f]_n| \leqslant \begin{cases} \|f\|_{\mathcal{E}_\rho} & \text{for } n = 0, \\ 2\|f\|_{\mathcal{E}_\rho} \rho^{-n} & \text{for } n \geqslant 1, \end{cases}$$

where $\|f\|_{\mathcal{E}_\rho} = \sup_{z \in \mathcal{E}_\rho} |f(z)|$ is the uniform norm over $\mathcal{E}_\rho$. In particular,

$$\|f - \mathbf{\Pi}_n\{f\}\|_\infty \leqslant \|f - \mathbf{\Pi}_n\{f\}\|_{\text{Ч}^1} \leqslant 2\|f\|_{\mathcal{E}_\rho} \frac{\rho^{-n}}{\rho - 1}, \qquad n \geqslant 0.$$

Let us fix $\rho > 1$ such that all coefficients of $\mathbf{L}$ (and consequently the IVP solutions) are analytic over $\mathcal{E}_\rho$. Theorem 3.16 provides an estimate for $N_{\mathfrak{R}}$ under the simplifying assumption that exact truncated Chebyshev series of degree $N_{\mathfrak{R}}$ for $\varphi_i^{(r)}$ and ψ_i can be computed. We discuss later why including the approximation error of the Chebyshev spectral method used in ResolventKernel does not significantly affect it.

Theorem 3.16 *Under the hypothesis that $\mathfrak{R}(x,t)$ is approximated using exact truncated Chebyshev series for $\varphi_i^{(r)}$ and ψ_i:*

$$\varphi_i^{(r)\circ} = \mathbf{\Pi}_{N_{\mathfrak{R}}}\{\varphi_i^{(r)}\}, \qquad \psi_i^\circ = \mathbf{\Pi}_{N_{\mathfrak{R}}}\{\psi_i\}, \qquad 0 \leqslant i < r, \tag{18}$$

the minimum value for $N_{\mathfrak{R}}$ to ensure a contracting Newton–Picard fixed-point operator is bounded by the asymptotic estimate:

$$N_{\mathfrak{R}} = \mathcal{O}\left(\frac{\log\left(1+\sum_{i=0}^{r-1}\|a_i\|_\infty \frac{T^{r-i}}{(r-i)!}\right) + \log\left(T\sum_{i=0}^{r-1}\|\varphi_i^{(r)}\|_{\mathcal{E}_\rho}\|\psi_{r-1-i}\|_{\mathcal{E}_\rho}\right) - \log(\rho-1)}{\log\rho}\right).$$

Proof • We first bound the operator norm of the linear part $\mathbf{E} = \mathbf{I} - (\mathbf{I}+\mathbf{R}^\circ)(\mathbf{I}+\mathbf{K}) = (\mathbf{R}-\mathbf{R}^\circ)(\mathbf{I}+\mathbf{K})$ of the Newton–Picard fixed-point operator $\mathbf{T}$ as:

$$\|\mathbf{E}\|_\infty \leqslant T\left(\sum_{i=0}^{r-1}(\|\varphi_i^{(r)}\|_\infty \eta_{r-1-i}^{(N_{\mathfrak{R}})} + \|\psi_{r-1-i}\|_\infty \varepsilon_i^{(N_{\mathfrak{R}})} + \varepsilon_i^{(N_{\mathfrak{R}})}\eta_{r-1-i}^{(N_{\mathfrak{R}})}\right)\left(1+\sum_{i=0}^{r-1}\|a_i\|_\infty\frac{T^{r-i}}{(r-i)!}\right). \tag{19}$$

Indeed, the definition of $\mathfrak{K}(x,t)$ in Proposition 2.3 or 2.5 yields:

$$\|\mathbf{K}\{f\}\|_\infty \leqslant \left(\sum_{i=0}^{r-1}\|a_i\|_\infty \sup_{x\in[x_l,x_r]} \operatorname{sgn}(x-x_0)\int_{x_0}^{x}\frac{|x-t|^{r-1-i}}{(r-1-i)!}\mathrm{d}t\right)\|f\|_\infty \leqslant \left(\sum_{i=0}^{r-1}\|a_i\|_\infty\frac{T^{r-i}}{(r-1)!}\right)\|f\|_\infty,$$

from which we have the expected bound for $\|\mathbf{K}\|_\infty$. Moreover, in the $\mathbf{x}-\boldsymbol{\partial}$ case:

$$\begin{aligned}\mathfrak{R}^\circ(x,t) - \mathfrak{R}(x,t) = \sum_{i=0}^{r-1}\Big(&\varphi_i^{(r)}(x)(\psi_{r-1-i}^\circ - \psi_{r-1-i})(t) + (\varphi_i^{(r)\circ}-\varphi_i^{(r)})(x)\psi_{r-1-i}(t)\\ &+(\varphi_i^{(r)\circ}-\varphi_i^{(r)})(x)(\psi_{r-1-i}^\circ-\psi_{r-1-i})(t)\Big),\end{aligned}$$

and a similar decomposition holds in the $\boldsymbol{\partial}-\mathbf{x}$ case. This gives the bound:

$$\|\mathbf{R}^\circ - \mathbf{R}\|_\infty \leqslant T\left(\sum_{i=0}^{r-1}(\|\varphi_i^{(r)}\|_\infty\eta_{r-1-i}^{(N_{\mathfrak{R}})} + \varepsilon_i^{(N_{\mathfrak{R}})}\|\psi_{r-1-i}\|_\infty + \varepsilon_i^{(N_{\mathfrak{R}})}\eta_{r-1-i}^{(N_{\mathfrak{R}})}\right).$$

• Instantiating $\varepsilon_i^{(N_{\mathfrak{R}})} = 2\|\varphi_i^{(r)}\|_{\mathcal{E}_\rho}\rho^{-N_{\mathfrak{R}}}/(\rho-1)$ and $\eta_i^{(N_{\mathfrak{R}})} = 2\|\psi_i\|_{\mathcal{E}_\rho}\rho^{-N_{\mathfrak{R}}}/(\rho-1)$ in (19) yields:

$$\|\mathbf{E}\| \leqslant 4T\left(\frac{\rho^{-N_{\mathfrak{R}}}}{\rho-1}+\frac{\rho^{-2N_{\mathfrak{R}}}}{(\rho-1)^2}\right)\left(\sum_{i=0}^{r-1}\|\varphi_i^{(r)}\|_{\mathcal{E}_\rho}\|\psi_{r-1-i}\|_{\mathcal{E}_\rho}\right)\left(1+\sum_{i=0}^{r-1}\|a_i\|_\infty\frac{T^{r-i}}{(r-i)!}\right).$$

Taking the logarithm and reordering the terms gives the claimed asymptotic lower bound for $N_{\mathfrak{R}}$. □

This estimate depends on the magnitude of the transition matrix associated to the linear ODE, which is consistent with the intuition that stiff ODEs with large amplitude solutions are harder to validate. We can however provide a more conservative estimate depending on the initial data only.

Corollary 3.17 *Let $T_\rho = (x_r - x_l)(\rho + \rho^{-1})/2 \geqslant T$ be the major axis of $\mathcal{E}_\rho$ and $M_\rho = \sum_{i=0}^{r-1} \|a_i\|_{\mathcal{E}_\rho} \frac{T_\rho^{r-1-i}}{(r-1-i)!}$. Then, under the same simplifying assumption (18), the minimum $N_{\mathfrak{R}}$ is given by:*

$$N_{\mathfrak{R}} = \mathcal{O}\left(\frac{T_\rho M_\rho - \log(\rho - 1)}{\log \rho}\right).$$

Proof Reusing the notations of (17), the formulas $\varphi_i^{(r)} = (\mathbf{I} + \mathbf{R})\{g_i\}$ and $\psi_i = (\mathbf{I} + \mathbf{R}^*)\{g_i^*\}$ extend over the Bernstein ellipse $\mathcal{E}_\rho$ in the complex plane. An adaptation of the inverse bound estimate of Theorem 3.3 yields $\|\mathbf{I}+\mathbf{R}\|_{\mathcal{E}_\rho}, \|\mathbf{I}+\mathbf{R}^*\|_{\mathcal{E}_\rho} \leqslant e^{T_\rho M_\rho}$. Also, for $0 \leqslant i < r$:

$$\|g_i\|_{\mathcal{E}_\rho} \|g_{r-1-i}^*\|_{\mathcal{E}_\rho} \leqslant \sum_{j=0}^{i} \|a_j\|_{\mathcal{E}_\rho} \frac{T_\rho^{r-1-j}}{(i-j)!(r-1-i)!} \leqslant M_\rho.$$

Hence, in the estimate of Theorem 3.16:

$$\log\left(T_\rho \sum_{i=0}^{r-1} \|\varphi_i^{(r)}\|_{\mathcal{E}_\rho} \|\psi_{r-1-i}\|_{\mathcal{E}_\rho}\right) \leqslant \log(r T_\rho M_\rho e^{2T_\rho M_\rho}) = \mathcal{O}(T_\rho M_\rho),$$

from which we obtain the claimed asymptotic estimate. □

This estimate confirms the significant efficiency gap between the Newton–Picard and the Newton–Galerkin validation methods for moderate to hard instances. Both algorit0hms run in quadratic time w.r.t. the parameter $N_{\mathcal{G}}$ (truncation index in Newton–Galerkin) or $N_{\mathfrak{R}}$ (resolvent kernel approximation degree in Newton–Picard). However, contrary to the linear estimate for $N_{\mathfrak{R}}$ given above, $N_{\mathcal{G}}$ can grow exponentially fast w.r.t. the magnitude of the input ODE, as explained in Sect. 2.4.2.

This conclusion must be tempered by noticing that the quantity M_ρ characterizes the magnitude of the input ODE over the Bernstein ellipse $\mathcal{E}_\rho$. In fact, $\|a_i\|_{\mathcal{E}_\rho}$ may be in theory way larger than $\|a_i\|_{\mathbb{Y}^1}$ (and consequently than $\|a_i\|_\infty$). Indeed, for $n \geqslant 0$, $\|T_n\|_\infty = \|T_n\|_{\mathbb{Y}^1} = 1$, while $\|T_n\|_{\mathcal{E}_\rho} = \mathcal{O}(\rho^n)$. However, most of the time in practice, the a_i are polynomials of reasonable degrees or analytic functions with fast decaying Chebyshev coefficients, so that this ratio still makes $N_{\mathfrak{R}}$ way smaller than $N_{\mathcal{G}}$. If not, then both Newton–Galerkin and Newton–Picard perform poorly for the mere reason that even the numerical spectral method computing a candidate Chebyshev approximation will require a very high degree, since the Chebyshev coefficients of the solution may be large and decrease slowly.

Including the Spectral Method Error. The estimates for $N_{\mathfrak{R}}$ given in Theorem 3.16 and Corollary 3.17 above rely on the simplifying assumption (18) that $\mathfrak{R}^\circ(x,t)$ is built from the exact truncated Chebyshev series for the $\varphi_i^{(r)}$ and ψ_i defining the resolvent kernel $\mathfrak{R}(x,t)$. In practice, such truncations cannot be directly obtained, and Algorithm ResolventKernel resorts to spectral methods to compute the desired polynomial approximations $\varphi_i^{(r)\circ}$ and ψ_i°. Therefore, we prove in Theorem 3.18 below that the same asymptotic estimate for $N_{\mathfrak{R}}$ holds in this more realistic setting where method errors of spectral methods are taken into account.

Chebyshev spectral methods perform roughly as well as merely truncating the Chebyshev series (see e.g. [29, Thm. 4.5]), with a convergence in $\mathcal{O}(\rho^{-N})$ with N the approximation degree in the analytic case. However, the proof is asymptotic, and valid as long as $\varphi + \mathbf{K}^{[N]}\{\varphi\} = g$ has a solution φ with $\deg \varphi \leqslant N$. Obviously, since $\mathbf{I}+\mathbf{K}$ is itself invertible, so is $\mathbf{I}+\mathbf{K}^{[N]}$ for N sufficiently large, e.g. if $\|(\mathbf{I}+\mathbf{K})^{-1}(\mathbf{K}-\mathbf{K}^{[N]})\|_{\mathbb{Y}^1} \leqslant 1$. But this quantity, already encountered in (8), precisely leads to an exponential bound for $N = N_{\mathcal{G}}$ in the Newton–Galerkin validation method. To avoid this phenomenon for the degree $N = N_{\mathfrak{R}}$ used in Newton–Picard, we propose a slightly modified procedure ResolventKernel, by replacing the linear system with a least-squares problem to approximate the exact solution $\varphi^* = (\mathbf{I}+\mathbf{K})^{-1}\{g\}$. It is important to note that the following development is essentially theoretical and aims at justifying the linear asymptotic estimate for $N_{\mathfrak{R}}$ given in Corollary 3.17. In practice however, ResolventKernel is still implemented as presented in Sect. 3.3.2.

Let $m = N - r - s$, and call $\mathbf{K}^{[N,m]} = \mathbf{\Pi}_N\, \mathbf{K}\, \mathbf{\Pi}_m$ the $(N+1) \times (m+1)$ rectangular projection of $\mathbf{K}$. The key observation is that $\mathbf{K}^{[N,m]}\{\varphi\} = \mathbf{K}^{[N]}\{\varphi\} = \mathbf{K}\{\varphi\}$ for all φ with $\deg \varphi \leqslant m$. To solve the following least-squares problem, we replace $\|\cdot\|_{\mathcal{U}^1}$ by an Euclidean norm, namely $\|h\|_2 = \sqrt{\sum_{i=0}^N a_i^2}$ for $h = \sum_{i=0}^N a_i T_i$, keeping in mind the classical norm equivalence $\|h\|_2 \leqslant \|h\|_{\mathcal{U}^1} \leqslant \sqrt{N+1}\|h\|_2$.

$$\begin{aligned} \min_{\varphi} \quad & \|(\mathbf{I} + \mathbf{K}^{[N,m]})\{\varphi\} - g\|_2, \\ \text{s.t.} \quad & \deg \varphi \leqslant m. \end{aligned} \tag{20}$$

Let $\widetilde{\varphi}$ be the optimal solution to (20). By comparing it with suboptimal $\overline{\varphi} = \mathbf{\Pi}_m\{\varphi^*\}$, we get:

$$\begin{aligned} \|(\mathbf{I} + \mathbf{K}^{[N,m]})\{\widetilde{\varphi}\} - g\|_2 &\leqslant \|(\mathbf{I} + \mathbf{K}^{[N,m]})\{\overline{\varphi}\} - g\|_2 \leqslant \|(\mathbf{I} + \mathbf{K})\{\overline{\varphi}\} - g\|_{\mathcal{U}^1} \\ &\leqslant \|\mathbf{I} + \mathbf{K}\|_{\mathcal{U}_1} \|\overline{\varphi} - \varphi^*\|_{\mathcal{U}^1} \leqslant \|\mathbf{I} + \mathbf{K}\|_{\mathcal{U}_1} 2\|\varphi^*\|_{\mathcal{E}_\rho} \frac{\rho^{-N+r+s}}{\rho - 1}. \end{aligned}$$

We therefore deduce the bound:

$$\begin{aligned} \|\widetilde{\varphi} - \varphi^*\|_\infty &\leqslant \|(\mathbf{I} + \mathbf{K})^{-1}\|_\infty \|(\mathbf{I} + \mathbf{K}^{[N,m]})\{\widetilde{\varphi}\} - g\|_\infty \\ &\leqslant \|(\mathbf{I} + \mathbf{K})^{-1}\|_\infty \sqrt{N+1} \|(\mathbf{I} + \mathbf{K}^{[N,m]})\{\widetilde{\varphi}\} - g\|_2 \\ &\leqslant \|(\mathbf{I} + \mathbf{K})^{-1}\|_\infty \|\mathbf{I} + \mathbf{K}\|_{\mathcal{U}_1} 2\|\varphi^*\|_{\mathcal{E}_\rho} \sqrt{N+1} \frac{\rho^{-N+r+s}}{\rho - 1}. \end{aligned} \tag{21}$$

Theorem 3.18 *With the polynomial approximations $\varphi_i^{(r)\circ}$ and ψ_i° computed by* ResolventKernel *using the modified spectral method presented above, the minimum value for $N_{\mathfrak{R}}$ to ensure a contracting Newton–Picard fixed-point operator is bounded by the asymptotic estimate:*

$$N_{\mathfrak{R}} = \mathcal{O}\left(\frac{T_\rho M_\rho - \log(\rho - 1)}{\log \rho}\right).$$

Proof Combining the method error for the (modified) Chebyshev spectral method (21) with $\|(\mathbf{I} + \mathbf{K})^{-1}\|_\infty \leqslant e^{MT} \leqslant e^{M_\rho T_\rho}$ and $\|\mathbf{I} + \mathbf{K}\|_{\mathcal{U}^1} \leqslant 1 + \frac{2M_\rho T_\rho}{\rho - 1}$, we get for $\varphi = \varphi_i^{(r)}$ or ψ_i and $N = N_{\mathfrak{R}}$:

$$\|\varphi^\circ - \varphi\|_\infty \leqslant e^{M_\rho T_\rho}\left(1 + \frac{2M_\rho T_\rho}{\rho - 1}\right) 2\|\varphi\|_{\mathcal{E}_\rho} \sqrt{N_{\mathfrak{R}} + 1} \frac{\rho^{-N_{\mathfrak{R}}+r+s}}{\rho - 1}.$$

Instantiating the error bounds $\varepsilon_i^{(N_{\mathfrak{R}})}$ and $\eta_i^{(N_{\mathfrak{R}})}$ with these values instead of the mere truncation bounds in the proof of Theorem 3.16 yields the same asymptotic estimate for $N = N_{\mathfrak{R}}$ in Corollary 3.17. □

To conclude, we also note that this modified procedure still runs in $\mathcal{O}(N_{\mathfrak{R}}(r+s)^2)$ arithmetic operations. Let $M = I_{N+1,m+1} + \mathbf{K}^{[N,m]}$. The solutions $\widetilde{\varphi}$ to the least squares problem (20) are the solutions of:

$$M^T M \widetilde{\varphi} = M^T g. \tag{22}$$

Since M is (rectangular) almost-banded, adapting the algorithms of [29] leads to a QR decomposition of M in complexity $\mathcal{O}(N_{\mathfrak{R}}(r+s)^2)$ and a solution $\widetilde{\varphi}$ of (22) in complexity $\mathcal{O}(N_{\mathfrak{R}}(r+s))$.

Remark 3.19 The overall complexity analysis in this article does not include the effect of rounding errors of floating-point arithmetics and the growth of interval width in the validation part. This could be avoided by using rational numbers instead. However, in most examples we tried, including those presented in the following section, rounding errors do not seem to play an important role. Also, the Ч^1 norm used in line 9 of NewtonPicardValid could be replaced by a polynomial extrema isolation technique to get sharper bounds.

4 Implementation, Examples and Discussion

We briefly present our implementation of the Newton–Picard validation algorithm presented in this paper, and draw up more detailed observations based on practical examples.

4.1 Implementation in the ChebValid library

Algorithm NewtonPicardValid presented in this article is implemented in the C library ChebValid[6] dedicated to rigorous numerics with RPAs in the Chebyshev basis (a.k.a. Chebyshev models). This library, which relies on MPFR[7] for floating-point arithmetics in arbitrary precision and its extension MPFI[8] for interval arithmetics, was initially designed to implement the Newton–Galerkin validation method of [8]. It implements data types as polynomials with double floating-point numbers, arbitrary precision floating-point numbers or interval coefficients, under the name `double_chebpoly_t`, `mpfr_chebpoly_t` and `mpfi_chebpoly_t`. The type `chebmodel_t` implements Chebyshev models, that is a pair consisting of a `mpfi_chebpoly_t` and a floating-point bound for the remainder.

For the purpose of this article, we enriched ChebValid with low-rank integral operators, implemented as the type `prefix_lrintop_t` (where `prefix_ = double_chebpoly_`, `mpfr_chebpoly_` or `chebmodel_`), together with relevant operations, in particular evaluation on a polynomial or RPA, composition (that is, operation $*$ on the kernels), and rigorous bound. The validation method is implemented in `mpfi_chebpoly_newtonpicard/`. In Sect. 4.2 below, we test it on a D-finite example, namely the Airy function of the first kind, and discuss the performances compared to other a-posteriori or self-validating methods. Also, Sect. 4.3 exemplifies the extension of NewtonPicardValid to non-polynomial coefficients, represented using RPAs (i.e., `chebmodel_t`), and implemented in `chebmodel_newtonpicard/`.

4.2 A D-finite Example: The Airy Function

The Airy function of the first kind Ai is a special function, notably used in different branches of physics (e.g., quantum mechanics and optics). It is a D-finite function, satisfying the following second order linear ODE:

$$y''(x) - xy(x) = 0, \tag{23}$$

6 https://gitlab.inria.fr/brisebar/tchebyapprox/-/releases/v0.0.1

7 https://www.mpfr.org/

8 https://gforge.inria.fr/projects/mpfi/

with transcendental initial conditions $v_0 = \mathrm{Ai}(0) = 3^{-2/3}/\Gamma(\frac{2}{3})$ and $v_1 = \mathrm{Ai}'(0) = -3^{-1/3}/\Gamma(\frac{1}{3})$. While Ai converges exponentially fast as $x \to +\infty$, namely $\mathrm{Ai}(x) \sim x^{-\frac{1}{4}} e^{-\frac{2}{3}x^{3/2}}/2\sqrt{\pi}$, the Airy function of the second kind Bi, which together with Ai forms a basis of the solution space of (23), is exponentially divergent, with $\mathrm{Bi}(x) \sim x^{-\frac{1}{4}} e^{\frac{2}{3}x^{3/2}}/\sqrt{\pi}$ as $x \to +\infty$ (see Fig. 3a). This means that even slight perturbations of the (non-rational) initial conditions v_0 and v_1 result in exponentially growing errors when evaluating $\mathrm{Ai}(x)$ using the D-finite equation (23). This justifies the use of rigorous validation methods.

First, this IVP for Ai is transformed into an equivalent integral equation on $f = \mathrm{Ai}''$ using Proposition 2.3:

$$f(x) + \int_0^x (xt - x^2) f(t)\mathrm{d}t = v_0 x + v_1 x^2. \tag{24}$$

A polynomial approximation y° is computed using the Chebyshev spectral method described in Sect. 2.3. The magnitude of the Chebyshev coefficients of Ai over the symmetric interval $[-a, a] = [-15, 15]$ is depicted in Fig. 3b. The observed super-linear convergence in log-scale (that is, in $o(\rho^{-n})$ for any $\rho > 1$) is typical for entire functions. According to the plot, a Chebyshev polynomial approximation y° with double-precision accuracy ($\approx 10^{-16}$) needs to be of degree roughly 90. However, such a heuristic truncation, as done for example in the Chebfun library, is not a rigorous proof.

As clearly evidenced by Fig. 3c, the coefficients in the matrix representation in the Chebyshev basis of $(\mathbf{I}+\mathbf{K})^{-1}$ converge exponentially fast to 0 when moving off a few diagonals or upper rows, in accordance with the logarithmic bound for $N_{\mathfrak{R}}$ claimed in Theorem 3.16. On the contrary, it also appears that finite-dimensional truncations are, by far, less efficient approximations. This phenomenon is reflected by the minimum value for Newton–Galerkin's $N_{\mathcal{G}}$ and Newton–Picard's $N_{\mathfrak{R}}$ needed to ensure a contracting fixed-point operator. More precisely, Fig. 3d compares the evolution in function of $[x_l, x_r] = [-a, a]$ of $N_{\mathcal{G}}$ (Newton–Galerkin), $N_{\mathfrak{R}}$ (Newton–Picard), and $N_{\mathcal{P}}$, the number of Picard iterations needed in the algorithm of [2]. In addition, we provide the a priori conservative estimate for $N_{\mathfrak{R}}$ from Corollary 3.17, computed over a Bernstein ellipse with $\rho = 2$ (the estimate can be made effective rather than asymptotic by using the bounds in the proof).

The first observation is that Newton–Galerkin validation method can hardly run with $a \geqslant 10$, due to the exponential blow-up of the truncation index $N_{\mathcal{G}}$. This indeed requires to perform rigorous computations with matrices with several billions entries... The second observation is that although $N_{\mathfrak{R}}$ and $N_{\mathcal{P}}$ should be similar in theory (which is reflected by the similar behavior of $N_{\mathcal{P}}$ and the a priori estimate for $N_{\mathfrak{R}}$), in practice $N_{\mathcal{P}}$ is significantly larger, because it is computed as a conservative, purely a priori estimate, contrary to $N_{\mathfrak{R}}$ where the actual contraction ratio of $\mathbf{T}$ is explicitly computed.

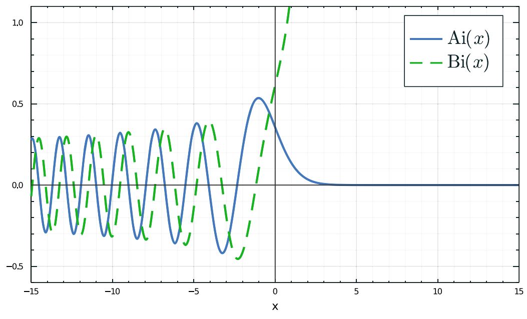
Timing Comparison. To conclude the Airy example, we compare timings of computing RPAs in the Chebyshev basis using several a posteriori validation methods:

- The Newton–Picard validation algorithm NEWTONPICARDVALID presented in this article and implemented in the C library ChebValid.
- The Newton–Galerkin validation method detailed in [8], for which the ChebValid library was initially designed.
- The Picard iteration based a posteriori validation algorithm of [2], which we implemented (without any optimization) in ChebValid.

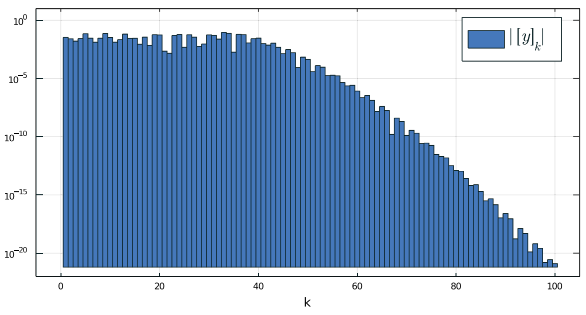
Although the application scope of self-validating methods slightly differs (the validation method constructs its own approximant, which is no longer freely provided by the user), we also compared for the sake of completeness with two implementations compatible with multiprecision:

- The CAPD library[9] computes rigorous Taylor forms with remainders using a higher-order Lohner method [40]. The result is a piecewise analytic representation of the solution, as opposed to uniform polynomial approximants considered in this article.

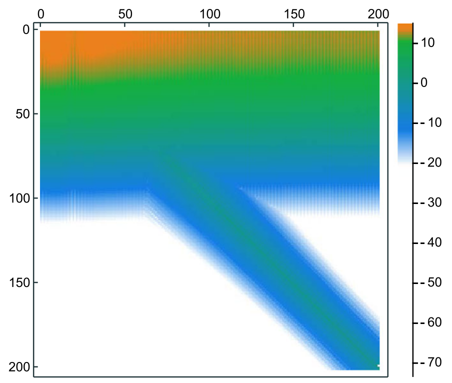
[9] http://capd.ii.uj.edu.pl/



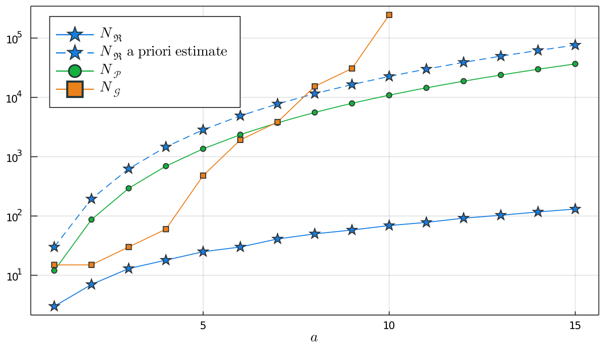
(A) Airy functions Ai and Bi.



(B) Chebyshev coefficients of $y = \mathrm{Ai}$ for $a = 15$.



(C) $(\mathbf{I} + \mathbf{K})^{-1}$ with $\mathbf{K}$ in (24) and $a = 15$.



(D) Comparison of Newton-Picard's $N_{\mathfrak{R}}$ (real value and a priori estimate) with Newton-Galerkin's $N_{\mathcal{G}}$ and the number $N_{\mathcal{P}}$ of Picard iterations in [2].

Fig. 3 Validation of the Airy function Ai over $[x_l, x_r] = [-a, a]$

- The algorithm developed in [25] and implemented in the SageMath `ore_algebra` package[10] combines a majorant series approach with a binary splitting algorithm to evaluate pointwise D-finite functions. The scope is even more restricted, but we fairly acknowledge the remarkable performances of this implementation.

We first note that in the category of Newton-like a posteriori validation methods, the Newton–Picard algorithm presented in this paper outperforms Newton–Galerkin and Picard iterations, which are both unable to deal with the instances where $a \geqslant 10$ in reasonable time. For the former, the main reason is that the required truncation index $N_{\mathcal{G}}$ becomes very large and rapidly exceeds the memory available on a standard machine. For the latter, the number of iterations remains reasonable (although larger than the degree $N_{\mathfrak{R}}$ used in Newton–Picard), but the iteration process using interval arithmetics suffers from numerical instability, which in the simple case $a = 5$ forced us to use MPFR floating-point numbers with 20,000 bits of precision.

The comparison with self-validating methods used in CAPD or SageMath is more delicate, since those implementations are by far more mature than our prototypes. However, the timings highlight the "cost to pay" to construct a contracting fixed-point operator as a common feature of a posteriori validation methods. For very high precision, this constant cost is amortized by the linear asymptotic complexity w.r.t. the degree of the candidate approximation. In particular, NewtonPicardValid performs better than CAPD when high precision is required. Note however that we did not manage to use CAPD with Taylor models of degree more than 70, which may explain the poorer performance of this reputed library in high precision.

Finally, when it comes to the simple pointwise rigorous evaluation of Ai, the algorithm implemented in SageMath outperforms all the other methods. One reason for this is that it uses a *binary splitting* technique to evaluate the

10 https://github.com/mkauers/ore_algebra/.

Table 2 Timings comparison for the rigorous computation of Ai over $[-a, a]$ with prescribed target accuracy, between Newton–Picard (N–P), Newton–Galerkin (N–G), Picard iterations (Pit) and Taylor forms in the CAPD library. Timings for the rigorous evaluation of $\mathrm{Ai}(a)$ using the SageMath `ore_algebra` package are also provided. "mpfr prec" denotes the MPFR precision in bits and "deg" the degree of the Chebyshev polynomial approximation used in (N-P), (N-G) and (Pit)

a	Accuracy	mpfr prec	deg	Timings (s)				
				N-P	N-G	Pit	CAPD	(Sage)
5	1e–16	128	45	**0.016**	0.306	107.	0.027	(0.008)
5	1e–32	256	65	**0.021**	0.382	108.	0.037	(0.009)
5	1e–64	512	105	**0.032**	0.520	110.	0.073	(0.010)
5	1e–128	512	165	**0.038**	0.523	113.	0.465	(0.011)
10	1e–16	256	85	0.060	–	–	**0.044**	(0.010)
10	1e–32	256	110	**0.064**	–	–	0.069	(0.010)
10	1e–64	512	155	**0.091**	–	–	0.137	(0.012)
10	1e–128	1024	235	**0.210**	–	–	1.554	(0.013)
15	1e–16	256	140	0.198	–	–	**0.066**	(0.012)
15	1e–32	512	165	0.268	–	–	**0.097**	(0.012)
15	1e–64	512	215	0.287	–	–	**0.251**	(0.014)
15	1e–128	1024	300	**0.602**	–	–	2.945	(0.017)

truncated Taylor series without computing explicitly the Taylor coefficients up to the truncation order. This is quite different from Problem 1.1 where we want to compute an RPA with an explicit polynomial approximation that can be manipulated. However, some techniques used in this implementation could certainly be reused to improve the existing algorithms for RPAs.

4.3 A Boundary Value Problem with Non-Polynomial Coefficients

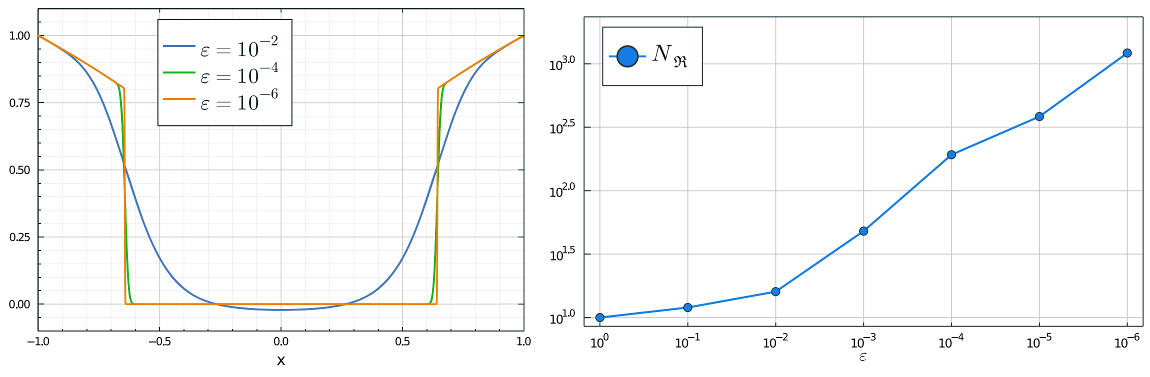
As mentioned in the introduction, Algorithm NewtonPicardValid easily extends to the case of linear ODEs with non-plynomial coefficients, represented using RPAs. Indeed, one just needs to provide routines to manipulate low-rank kernels defined with univariate RPAs rather than pure polynomials, namely evaluation on another RPA, composition and rigorous bound. This is implemented in the ChebValid library, with the prefix `chebmodel_` replacing `mpfi_chebpoly_` in the names of the corresponding functions.

To illustrate this, let us consider the following boundary layer problem considered in [29] and reused in [8]:

$$\begin{cases} y''(x) - \dfrac{2x}{\varepsilon}\left(\cos x - \dfrac{8}{10}\right) y'(x) + \dfrac{1}{\varepsilon}\left(\cos x - \dfrac{8}{10}\right) y(x) = 0, \qquad x \in [-1, 1], \\ y(-1) = y(1) = 1. \end{cases} \tag{25}$$

This is a Boundary Value Problem (BVP) rather than an Initial Value Problem (IVP). However, since the problem is linear, this amounts to solving two IVPs on the same ODE, namely computing $y = y_0 + \lambda y_1$ with y_0, y_1 solutions to the IVP with initial conditions (1, 0) and (0, 1), respectively, and with $\lambda = (1 - y_0(1))/y_1(1)$.

We first use Algorithm NewtonPicardValid to compute an RPA for cos over $[-1, 1]$, using the differential equation $y'' + y = 0$. This allows us to provide RPAs of arbitrary accuracy for the two coefficients in (25). Then we call this algorithm again on the resulting IVP, overloaded with RPAs. The smaller ε is, the steeper the slope of the solution y is, as it can be observed in Fig. 4a. This results in an increasing degree $N_{\mathfrak{R}}$ to approximate the resolvent kernel (see Fig. 4b). Note however that $N_{\mathfrak{R}}$ is way smaller than the truncation order $N_{\mathcal{G}}$ required by



(A) Solution for different values of ε. **(B)** Required degree $N_{\mathfrak{R}}$ in function of ε.

Fig. 4 Validation of the boundary layer example using NewtonPicardValid

Newton–Galerkin validation method (see [8, Sec. 7.3]). As a consequence, we can rigorously deal with instances with really small ε (up to 10^{-6} within a few seconds), while Newton–Galerkin is limited to $\varepsilon = 0.005$ roughly.

On a last note, one remarks that y_0 and y_1 have very high amplitude compared to the BVP solution y. With the algorithm of [29], one can in fact directly numerically solve this BVP by replacing the first two rows of the almost-banded system by the corresponding equations representing the boundary conditions. An interesting future work is to extend Newton–Picard validation algorithm to BVPs, by investigating the resolvent kernel of the associated integral equation.

4.4 High Order D-finite Equations

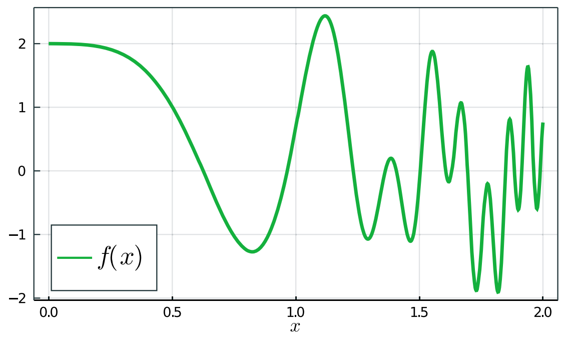
In order to check that the proposed method successfully handles linear ODEs of order greater than 2, 21 random D-finite equations $\mathbf{L}_{10}, \mathbf{L}_{11}, \dots, \mathbf{L}_{30}$ were generated, with order $r_i = i$ and degree 2 polynomial coefficients with integer coefficients uniformly chosen between -100 and 100, like the following one:

$$\begin{aligned}\mathbf{L}_{10} = \boldsymbol{\partial}^{10} &+ (-93\mathbf{x}^2 + 5\mathbf{x} + 73)\boldsymbol{\partial}^9 + (-51\mathbf{x}^2 + 22\mathbf{x} + 74)\boldsymbol{\partial}^8 + (-8\mathbf{x}^2 - 41\mathbf{x} + 41)\boldsymbol{\partial}^7 \\ &+ (-18\mathbf{x}^2 - 76\mathbf{x} - 11)\boldsymbol{\partial}^6 + (79\mathbf{x}^2 - 45\mathbf{x} + 70)\boldsymbol{\partial}^5 + (-11\mathbf{x}^2 + 61\mathbf{x} - 83)\boldsymbol{\partial}^4 \\ &+ (-74\mathbf{x}^2 - 5\mathbf{x} - 84)\boldsymbol{\partial}^3 + (-\mathbf{x}^2 - 63\mathbf{x} + 96)\boldsymbol{\partial}^2 + (-95\mathbf{x}^2 + 86\mathbf{x} - 57)\boldsymbol{\partial} + (-8\mathbf{x}^2 + 21\mathbf{x} - 56),\end{aligned}$$

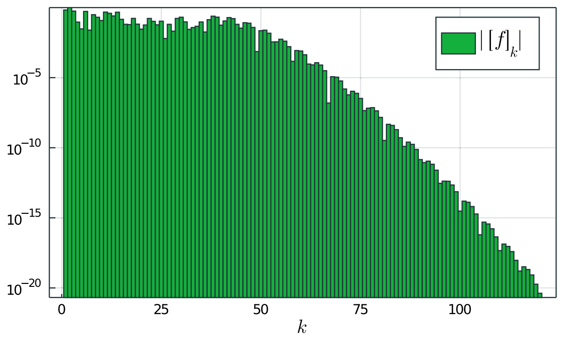
together with random integer initial conditions between -100 and 100.

Despite the high order and dense coefficients of the $\mathbf{L}_i$, Algorithm NewtonPicardValid requires quite moderate degrees $N_{\mathfrak{R}}$ to construct the Newton–Picard fixed-point operator (always less than 700 for those 21 randomly generated instances). Hence, we see in practice that the remarkable approximation properties of Chebyshev spectral methods are reflected in the validation procedure.

Examples of "natural" high order ODEs are hard to find in the literature and applications. Yet, they frequently appear in the context of D-finite functions, e.g. when performing closure operations like addition, multiplication or algebraic substitution on simpler functions, and their properties may be quite different from those of random equations. Consider for example the D-finite function $f(x) = \cos x + \cos 5x^2 - \sin 3x^4$ over $[0, 2]$, plotted in Fig. 5a. The `holexprtodiffeq` command of Maple's Gfun package [34] automatically computes an order 6, degree 37 differential equation $\mathbf{L}\{f\} = 0$, with a 6-dimensional solution space spanned by $\{\cos x, \sin x, \cos 5x^2, \sin 5x^2, \cos 3x^4, \sin 3x^4\}$. Unfortunately, the leading coefficient,



(A) $f(x) = \cos x + \cos 5x^2 - \sin 3x^4$ over $[0, 2]$.



(B) Chebyshev coefficients of f over $[0, 2]$.

Fig. 5 An example of a D-finite function with high-order, ill-conditioned differential equation

$$\begin{aligned}a_6(\mathbf{x}) = {} & 268738560000\mathbf{x}^{29} - 5374771200\mathbf{x}^{27} - 365158969344\mathbf{x}^{25} + 3732480000\mathbf{x}^{23} + 463694284800\mathbf{x}^{21} \\ & - 6366325248\mathbf{x}^{19} - 66390306048\mathbf{x}^{17} + 2217859200\mathbf{x}^{15} + 27057529296\mathbf{x}^{13} - 2477301120\mathbf{x}^{11} \\ & + 3908641720\mathbf{x}^9 - 134485400\mathbf{x}^7 + 1100233\mathbf{x}^5 + 15403262\mathbf{x}^3,\end{aligned}$$

is non-constant, and in particular singular at $x = 0$. To circumvent this difficulty, the `Desingularize` command can be used to compute a non-singular, higher order equation. The result is an operator of order 9 and degree 35, with significantly larger coefficients, which is generated in approximately 10 seconds in Maple. Our experiments on this equation show the following:

- The Chebyshev spectral method provides remarkably accurate numerical approximations of this function, despite the considerable size of the equation. Indeed, the Chebyshev coefficients (see Fig. 5b) converge at a super-linear rate, since f is entire.
- Unfortunately, NewtonPicardValid fails to validate this example in reasonable time, due to a too large degree $N_{\mathfrak{R}}$ needed to approximate the resolvent kernel so as to get a contracting operator. Indeed, the functions φ_i and ψ_i defining $\mathfrak{R}(x, t)$ are *way* larger than f, and cannot be accurately approximated with a moderate degree. The reason is that the desingularization step produces a much bigger and ill-conditioned equation (in this example, $\max_{0 \leqslant i < 9} \|a_i\|_{\mathbb{U}^1} = 1.26 \cdot 10^{22}$). Hence, the extra 3 functions in the new 9-dimensional solution space most probably have Chebyshev coefficients of size exponential in this bound, and are therefore not even representable in MPFR.

The key lesson of this example is that while the Chebyshev spectral method can perform well on specific initial conditions associated to a well-behaved particular function of the solution space of the differential equation, the Newton–Picard validation method needs to inspect the whole solution space (of $\mathbf{L}$ and $\mathbf{L}^*$) to guarantee the maximum error "in every direction". Still, future improvements could help in taming the contribution of the ill-conditioned but non-relevant solutions. Also, in this example and more generally for D-finite functions, handling directly singular equations is an interesting future challenge.

4.5 Discussion and Future Work

Newton–Galerkin a posteriori validation methods have been widely used over recent years to provide successful computer-assisted proofs for the existence (and the effective approximation) of solutions to various kinds of functional equations, notably (linear or nonlinear) ODEs with initial or boundary value conditions, partial differential equations, delay-differential equations, etc. Elaborating on the simpler case of linear ODEs with initial values, we presented a new validation algorithm (which we propose to call Newton–Picard) and provided a thorough complexity analysis to explain why it performs better than other a posteriori validation algorithms in the literature. This algorithm is implemented in the prototype C library ChebValid.

This work is to be seen as part of a long term effort to combine the field of rigorous numerics towards computer-assisted proofs in mathematics with the algorithmic and complexity approach of computer algebra, in order to design efficient and fully trusted symbolic-numeric libraries for function space computations. In particular, several future goals can be identified:

- Combining the Newton–Picard fixed-point operator with the *radii polynomial* approach of [17] to bound non-linear terms will allow for the efficient treatment of nonlinear ODEs, where even proving the existence of a solution on a given interval is a nontrivial problem. In particular, investigating the complexity and comparing with the results of [6] is an interesting challenge.
- Deriving formulas for the resolvent kernel in the case of boundary conditions is the key to adapting this algorithm to BVPs.
- Even in the case of D-finite functions, dealing with singular points (that is, when the leading coefficient, instead of being 1, vanishes at some point of the interval) is a difficult problem. While rigorous evaluation of D-finite functions near regular singular points has already been investigated, e.g. in [25,39], rigorous uniform approximation (with a suitable norm) is, up to our knowledge, still an open problem. A similar question is the case of unbounded intervals.
- Computing rigorous approximants for PDE solutions is a major challenge due to the (time and space) complexity issues related to the several variable case. Hence, proposing efficient and reliable algorithms is crucial in the development of computer-assisted proofs for such problems.
- Finally, we propose to implement this validation algorithm in the Coq proof assistant in order to provide a fully reliable tool to engineers and mathematicians. More specifically, we want to integrate it into our experimental development of certified RPAs in Coq using an a posteriori (or certificate-based) approach [9]. The algebraic framework of Newton–Picard with integral operators and polynomial kernels makes this new validation algorithm a relevant choice for that task.

Acknowledgements I would like to thank the anonymous referees for their careful reviewing work and their useful suggestions, which helped improve the clarity of this article.

References

1. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, volume 55 of National Bureau of Standards Applied Mathematics Series. Courier Corporation (1964)
2. Benoit, A., Joldes, M., Mezzarobba, M.: Rigorous uniform approximation of D-finite functions using Chebyshev expansions. Math. Comput. **86**(305), 1303–1341 (2017). https://doi.org/10.1090/mcom/3135
3. Berinde, V.: Iterative Approximation of Fixed Points. Lecture Notes in Mathematics, vol. 1912. Springer, Berlin (2007)
4. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development: Coq'Art: the Calculus of Inductive Constructions. Springer (2013)
5. Berz, M., Makino, K.: Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. Reliable Comput. **4**(4), 361–369 (1998). https://doi.org/10.1023/A:1024467732637
6. Bournez, O., Graça, D.S., Pouly, A.: On the complexity of solving initial value problems. In: Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation, pp. 115–121 (2012). https://doi.org/10.1145/2442829.2442849
7. Boyd, J.P.: Chebyshev and Fourier spectral methods. Dover Publications (2001)
8. Bréhard, F., Brisebarre, N., Joldes, M.: Validated and numerically efficient Chebyshev spectral methods for linear ordinary differential equations. ACM Trans. Math. Softw. **44**(4), 44:1–44:42 (2018). https://doi.org/10.1145/3208103
9. Bréhard, F., Mahboubi, A., Pous, D.: A certificate-based approach to formally verified approximations. In: 10th International Conference on Interactive Theorem Proving (ITP 2019), vol. 141, pp. 8:1–8:19 (2019). https://doi.org/10.4230/LIPIcs.ITP.2019.8
10. Brisebarre, N., Joldes, M.: Chebyshev interpolation polynomial-based tools for rigorous computing. In: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation, pp. 147–154. ACM (2010). URL: https://doi.org/10.1145/1837934.1837966
11. Chyzak, F.: The ABC of Creative Telescoping: Algorithms, Bounds, Complexity. Memoir of accreditation to supervise research (HDR), Université d'Orsay (2014). https://tel.archives-ouvertes.fr/tel-01069831

12. Epstein, C., Miranker, W.L., Rivlin, T.J.: Ultra-arithmetic. I. Function data types. Math. Comput. Simul. **24**, 1–18 (1982). https://doi.org/10.1016/0378-4754(82)90045-3
13. Epstein, C., Miranker, W.L., Rivlin, T.J.: Ultra-arithmetic. II. Intervals of polynomials. Math. Comput. Simul. **24**, 19–29 (1982). https://doi.org/10.1016/0378-4754(82)90046-5
14. Fox, L., Parker, I.B.: Chebyshev Polynomials in Numerical Analysis. Oxford University Press, London-New York-Toronto, Ont. (1968)
15. Gottlieb, D., Orszag, S.A.: Numerical Analysis of Spectral Methods: Theory and Applications, vol. 26. SIAM (1977)
16. Greengard, L.: Spectral integration and two-point boundary value problems. SIAM J. Numer. Anal. **28**(4), 1071–1080 (1991). https://doi.org/10.1137/0728057
17. Hungria, A., Lessard, J.-P., Mireles James, J.D.: Rigorous numerics for analytic solutions of differential equations: the radii polynomial approach. Math. Comput. **85**(299), 1427–1459 (2016). https://doi.org/10.1090/mcom/3046
18. Jiménez-Pastor, A., Pillwein, V., Singer, M.F.: Some structural results on D^n-finite functions. Adv. Appl. Math. **117**, 102027 (2020). https://doi.org/10.1016/j.aam.2020.102027
19. Joldes, M.: Rigorous Polynomial Approximations and Applications. PhD thesis, École normale supérieure de Lyon – Université de Lyon, Lyon, France (2011). https://tel.archives-ouvertes.fr/tel-00657843
20. Kaucher, E., Miranker, W.L.: Validating computation in a function space. In: Reliability in computing: the role of interval methods in scientific computing, pages 403–425. Academic Press Professional, Inc. (1988). https://doi.org/10.1016/B978-0-12-505630-4.50028-6
21. Kedem, G.: A posteriori error bounds for two-point boundary value problems. SIAM J. Numer. Anal. **18**(3), 431–448 (1981). https://doi.org/10.1137/0718028
22. Koutschan, C.: Advanced Applications of the Holonomic Systems Approach. PhD thesis, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria (2009). https://www3.risc.jku.at/research/combinat/software/ergosum/RISC/HolonomicFunctions.html
23. Lalescu, T.: Introduction à la Théorie des Équations Intégrales (Introduction to the Theory of Integral Equations). Hermann, Librairie Scientifique A (1911)
24. Lessard, J.-P., Reinhardt, C.: Rigorous numerics for nonlinear differential equations using Chebyshev series. SIAM J. Numer. Anal. **52**(1), 1–22 (2014). https://doi.org/10.1137/13090883X
25. Mezzarobba, M.: Autour de l' évaluation numérique des fonctions D-finies. PhD thesis, École polytechnique (2011). URL: https://tel.archives-ouvertes.fr/pastel-00663017/
26. Moore, R.E.: Interval Analysis. Prentice-Hall (1966)
27. Nakao, M.T.: Numerical verification methods for solutions of ordinary and partial differential equations. Numer. Funct. Anal. Optim. **22**(3–4), 321–356 (2001). https://doi.org/10.1081/NFA-100105107
28. Neumaier, A.: Taylor forms—use and limits. Reliable Comput. **9**(1), 43–79 (2003). https://doi.org/10.1023/A:1023061927787
29. Olver, S., Townsend, A.: A fast and well-conditioned spectral method. SIAM Rev. **55**(3), 462–489 (2013). https://doi.org/10.1137/120865458
30. Plum, M.: Computer-assisted existence proofs for two-point boundary value problems. Computing **46**(1), 19–34 (1991). https://doi.org/10.1007/BF02239009
31. Rall, L.B.: Resolvent kernels of Green's function kernels and other finite-rank modifications of Fredholm and Volterra kernels. J. Optim. Theory Appl. **24**, 59–88 (1978). https://doi.org/10.1007/BF00933182
32. Salvy, B.: D-finiteness: Algorithms and applications. In: M. Kauers (ed.) ISSAC 2005: Proceedings of the 18th International Symposium on Symbolic and Algebraic Computation, Beijing, China, July 24-27, 2005, pages 2–3. ACM Press (2005). https://doi.org/10.1145/1073884.1073886
33. Salvy, B.: Linear differential equations as a data-structure. Foundations of Computational Mathematics, pp. 1071–1012, (2019). https://doi.org/10.1007/s10208-018-09411-x
34. Salvy, B., Zimmermann, P.: Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. ACM Trans. Math. Softw. (TOMS) **20**(2), 163–177 (1994). https://doi.org/10.1145/178365.178368
35. Stanley, R.P.: Differentiably finite power series. Eur. J. Combinat. **1**, 175–188 (1980). https://doi.org/10.1016/S0195-6698(80)80051-5
36. Trefethen, L.N.: Approximation Theory and Approximation Practice. SIAM (2013). http://www.chebfun.org/ATAP/
37. Tucker, W.: Validated Numerics: A Short Introduction to Rigorous Computations. Princeton University Press (2011)
38. van den Berg, J.B., Lessard, J.-P.: Rigorous numerics in dynamics. Not. AMS **62**(9), 1 (2015). https://doi.org/10.1090/noti1276
39. van der Hoeven, J.: Fast evaluation of holonomic functions near and in regular singularities. J. Symbol. Comput. **31**(6), 717–743 (2001). https://doi.org/10.1006/jsco.2000.0474
40. Wilczak, D., Zgliczyński, P.: C^r-Lohner algorithm. Schedae Informaticae **20**, 9–42 (2011)
41. Yamamoto, N.: A numerical verification method for solutions of boundary value problems with local uniqueness by Banach's fixed-point theorem. SIAM J. Numer. Anal. **35**(5), 2004–2013 (1998). https://doi.org/10.1137/S0036142996304498
42. Zgliczyński, P.: C^1 Lohner algorithm. Found. Comput. Math. **2**(4), 429–465 (2002). https://doi.org/10.1007/s102080010025