Exchange algorithm for evaluation and approximation error-optimized polynomials

Denis Arzelier, Florent Bréhard, Mioara Joldes

Exchange algorithm for evaluation and approximation error-optimized polynomials

Denis Arzelier[1], Florent Bréhard[2], and Mioara Joldes[3]

[1]LAAS-CNRS, Toulouse, France, arzelier@laas.fr
[2]ENS de Lyon & LAAS-CNRS, Lyon & Toulouse, France, florent.brehard@ens-lyon.fr
[3]LAAS-CNRS, Toulouse, France, joldes@laas.fr

February 4, 2019

Abstract

Machine implementation of mathematical functions often relies on polynomial approximations. The particularity is that rounding errors occur both when representing the polynomial coefficients on a finite number of bits, and when evaluating it in finite precision. Hence, for finding the best polynomial (for a given fixed degree, norm and interval), one has to consider both types of errors: approximation and evaluation. While efficient algorithms were already developed for taking into account the approximation error, the evaluation part is usually a posteriori handled, in an ad-hoc manner. Here, we formulate a semi-infinite linear optimization problem whose solution is the best polynomial with respect to the supremum norm of the sum of both errors. This problem is then solved with an iterative exchange algorithm, which can be seen as an extension of the well-known Remez algorithm. A discussion and comparison of the obtained results on different examples are finally presented.

1 Introduction

Polynomials are often used for approximating functions on computers [1,21]. Their evaluation only requires additions and multiplications, which are efficiently implemented in hardware floating-point (FP) arithmetic units. FP

operations are specified by the IEEE 754-2008 [17] standard, which requires, among others, correctly rounded basic arithmetic operations $(+, -, *, /, \sqrt{})$ for several precision formats, and recommends correctly rounded elementary functions like exp, sin, cos. Very efficient fixed FP precision implementations exist [15, 20] for such functions and are collected in mathematical libraries (*libm*s), which can be nowadays almost automatically generated and tuned [12, 18]. Recently, in [19], such code generating techniques were extended to larger classes of special functions, which are widely used in scientific and technical applications (like Bessel, Airy, Erf, etc.).

The problem of evaluating a function for the whole FP input range is firstly reduced to the evaluation of an approximation valid in a rather small compact domain I. This can be done for instance, by argument reduction techniques, which are available only for specific elementary functions, and/or by piecewise polynomial approximations [22]. Then, the implementation task becomes: given a description of a function f, an input interval I, and a target accuracy $\varepsilon > 0$, one is requested a source code which provides a function $\tilde{f}$, such that: $\left\|(f - \tilde{f})/f\right\|_I \leqslant \varepsilon$, where we denote by $\|g\|_I := \sup_{t \in I} |g(t)|$ the supremum norm of g on I.

Typically, this is handled in two main steps:

Approximation an approximation polynomial p is searched for, such that two main requirements are met: its coefficients are representable with a specified fixed precision format (usually, binary32, binary64, or an unevaluated sum of such formats) and the approximation error is less than a target $\varepsilon_{\text{approx}}$, whether absolute $\|f - p\|_I \leqslant \varepsilon_{\text{approx}}$ or relative $\|(f - p)/f\|_I \leqslant \varepsilon_{\text{approx}}$.

For that, efficient algorithms were developed in [5, 6]. In the simpler case of polynomials p with real coefficients and given degree n, $p = \sum_{i=0}^{n} a_i t^i$, this boils down to the so-called *minimax* problem:

$$\min_{\substack{a_i \in \mathbb{R}, \\ i \in [0..n]}} \max_{t \in I} |f(t) - p(t)|, \qquad (P_{\text{minimax}})$$

which can be solved by the Remez algorithm (see [5, 10] and references therein). This iterative algorithm has quadratic convergence and rather low complexity, since it involves solving a linear system of size $n+2$ at each step, together with numerically computing the extrema of $f - p$ over I.

Evaluation an efficient evaluation scheme for p is searched for; since after each addition or multiplication, rounding errors occur, one must ensure that

the computed value $\tilde{p}$ satisfies $\|p - \tilde{p}\|_I \leqslant \varepsilon_{\text{eval}}$ (or $\|(p - \tilde{p})/p\|_I \leqslant \varepsilon_{\text{eval}}$) for a given threshold $\varepsilon_{\text{eval}}$.

Heuristics presented in [20] extend the precision of the *important* coefficients, such that the evaluation error remains below $\varepsilon_{\text{eval}}$. For instance, Sollya command *implementpoly* uses a Horner-based evaluation scheme, which behaves rather well when the evaluation interval is sufficiently small and contains zero. Otherwise, consider step i of Horner evaluation $a_i + t\tilde{p}_i(t)$, where $\tilde{p}_i$ is the already computed partial polynomial evaluation: when the argument $|t| >> 1$, the accumulated evaluation error is much amplified when multiplying by t. Another heuristic is a ratio test between a_i and $t\tilde{p}_i(t)$, to check for cancellation issues which appear when both terms have the same order of magnitude and opposite signs.

Once the coefficients have been chosen, the approximation and the evaluation error can *a posteriori* be certified by several existing algorithms and tools, like Sollya [11], Gappa [14], Rosa [13] or Real2Float [25].

It is important to note that steps *a)* and *b)* are usually independently considered. An exception occurs for the case of very small precisions or polynomial degrees, where an exhaustive search on the rounded coefficients is possible [28].

However, as explicitly mentioned in [6], *one would like to take into account the roundoff error that occurs during polynomial evaluation: getting the polynomial, with constraints on the size of the coefficients, that minimizes the total (approximation plus roundoff) error would be extremely useful.*

The purpose of this article is to make progress on this open question: we search for the coefficients of a polynomial $p(t) = \sum_{i=0}^{n} a_i t^i$, of given degree n, which minimizes the maximum of the sum of both approximation and evaluation errors over an input interval I, with respect to f. We consider a *black-box* description of f i.e., one disposes of values $f(t)$, up to any required accuracy [20]. This allows for handling very general functions (elementary, special, etc.), but also implies that no argument reduction step is usually possible. For simplicity we state the problem for the absolute error case:[1]:

$$\min_{\substack{a_i \in \mathbb{R}, \\ i \in [0..n]}} \max_{t \in I} \left(|f(t) - p(t)| + |\tilde{p}(t) - p(t)| \right) \qquad (P_{\text{general}})$$

In Section 2, we give a linearized bound for the evaluation error $|\tilde{p}(t) - p(t)|$. Based on [23], the performance of a given arbitrary evaluation scheme is recursively assessed by bounding the rounding error of each elementary

[1]The relative error can be similarly handled from a theoretical standpoint (cf. Appendix .2).

operation. This leads to the formulation in Section 3 of Problem P_{general} as a linear semi-infinite programming (SIP) problem [24, 26].

In this context, we show two results: on the theoretical side, based on the duality theory, we revisit, explain and extend an *exchange algorithm* [7–9, 29], which solves this problem in Section 4. On the practical side, the solution of this problem provides a first attempt on simultaneously optimizing over both errors: we show that in some cases the evaluation error can be improved. We also show that in some other cases, the minimax polynomial solution of problem P_{minimax} is very close to the solution of P_{general}. Numerical examples and a discussion are provided in Section 5.

2 Evaluation error

Consider first some basic notation used for error analysis [22]. Firstly, assume radix-2, precision-p, floating-point arithmetic with unbounded exponent range i.e, provided that overflows and underflows do not occur. If $t \in \mathbb{R}$, define $\text{RN}(t)$ as t rounded to nearest. This is the default rounding mode in IEEE-754 arithmetic [17], so that, given two FP numbers a and b, when the instruction $c = a \top b$ appears in a program, what is effectively computed is $c = \text{RN}(a \top b)$, for any arithmetic operation $\top \in \{+, -, \times, \div\}$. We have

$$\frac{|t - \text{RN}(t)|}{|t|} \leqslant \frac{u}{1+u} < u, \tag{1}$$

where $u = 2^{-p}$ is called the *rounding unit*.

Moreover, there exists a real number ϵ such that

$$\text{RN}(a \top b) = (a \top b)(1+\epsilon), \quad |\epsilon| \leqslant u. \tag{2}$$

Based on the previous property, the error of any arithmetic expression can be recursively bounded. Firstly, for specific evaluation schemes, like Horner, bounds date back to the work of Oliver [23], which is detailed below as an example. More recently, several works insisted on the automatic algorithmic approach via operator overloading similar to automatic differentiation [4, 27]. Based on this, we propose, for completeness[2], Algorithm 2, which automatically computes linearized expressions for the evaluation error (like in (6)), for any given symbolic expression tree e, provided with symbolic rounding errors for each tree node. Let us exemplify on the evaluation of the polyno-

[2]While the general ideas are the same as in [4, 27] and references therein, we could not find the exact pseudo-code in literature, so we state it in order to provide a complete algorithmic solution for problem P_{general}.

Algorithm 1 HORNER(p,t).

1: $r_n \leftarrow a_n$
2: **for** $i = n-1$ **downto** 0 **do**
3: $\quad r_i \leftarrow \mathrm{RN}\big(\mathrm{RN}(r_{i+1} \times t) + a_i\big)$
4: **end for**
5: **return** r_0

mial $p(t) = a_n t^n + a_{n-1}t^{n-1} + \cdots + a_0$ using Horner's rule, assuming that a Fused Multiply Add (FMA) instruction is not employed. The actual machine operations are recalled in Algorithm 1. We have:

$$r_n = a_n, \tag{3a}$$
$$r_{n-1} = \big(t r_n(1+\epsilon_{n-1}^{\times}) + a_{n-1}\big)(1+\epsilon_{n-1}^{+}), \tag{3b}$$

where $\epsilon_{n-1}^{\times}$ and ϵ_{n-1}^{+} model the rounding errors for multiplication and addition at step $n-1$. By induction, one obtains:

$$r_k = \sum_{i=k}^{n} \left((1+\epsilon_i^{+}) \prod_{j=k}^{i-1} (1+\epsilon_j^{+})(1+\epsilon_j^{\times}) \right) a_i t^{i-k}, \tag{4}$$

where we define $\epsilon_n^{+} := 0$ and $\prod_{j=k}^{k-1}(1+\epsilon_j^{+})(1+\epsilon_j^{\times}) := 1$. This implies that the total evaluation error is:

$$r_0 - \sum_{i=0}^{n} a_i t^i = \sum_{i=0}^{n} \left((1+\epsilon_i^{+}) \prod_{j=k}^{i-1} (1+\epsilon_j^{+})(1+\epsilon_j^{\times}) - 1 \right) a_i t^i. \tag{5}$$

Here, we consider only a linear approximation θ_{lin} of the evaluation error, function of ϵ_i^{+} and $\epsilon_i^{\times}$, in what follows. This gives, for our Horner example:

$$\theta_{\mathrm{lin}}^{(\mathrm{Horner})} := \sum_{j=0}^{n-1} \left(\sum_{i=j+1}^{n} a_i t^i \right) \epsilon_j^{\times} + \sum_{j=0}^{n-1} \left(\sum_{i=j}^{n} a_i t^i \right) \epsilon_j^{+}. \tag{6}$$

Moreover, provided bounds are specified for each rounding error, depending on the precision employed, one obtains upper bounds for the linearized absolute evaluation error. For instance, if binary64 is used for all the computations in Algorithm 1, with $u = 2^{-53}$, one has:

$$\left| \theta_{\mathrm{lin}}^{(\mathrm{Horner})} \right| \leqslant 2u \sum_{j=0}^{n}{}'' \left| \sum_{i=j}^{n} a_i t^i \right|, \tag{7}$$

where the double superscript indicates that the first and last terms in the summation are to be halved.

In order to automate this evaluation error analysis for the general case, we firstly associate to a "pure" mathematical expression e^*, a given symbolic evaluation scheme with roundings e, composed of terms $\mathrm{RN}(e', u)$. This means that e' is rounded with a relative error bounded by u. This formulation models both possible rounding errors on an input variable ($e' \in \mathcal{V}$, where $\mathcal{V}$ denotes the set of input variables) and the rounding errors of arithmetic operations (if $e' = a_1 \top e_2$). Then, we build an expression $\tilde{e}$, as in (4), by recursively replacing terms $\mathrm{RN}(e', u)$ in e, with $\tilde{e}'(1 + \epsilon_{e'}^{[u]})$ where $|\epsilon_{e'}^{[u]}| \leqslant u$. Several examples are shown in Table 1.

$e_1 = \mathrm{RN}(a + \mathrm{RN}(b \times c, u), u)$ $\qquad e_1^* = a + bc$
$\tilde{e}_1 = (a + bc(1 + \epsilon_{b\times c}^{[u]}))(1 + \epsilon_{a+\mathrm{RN}(b\times c,u)}^{[u]})$
$\theta_{\mathrm{lin}}^{(e_1)} = bc\epsilon_{b\times c}^{[u]} + (a + bc)\epsilon_{a+\mathrm{RN}(b\times c,u)}^{[u]}$
$\lvert\theta_{\mathrm{lin}}^{(e_1)}\rvert \leqslant (\lvert bc\rvert + \lvert a + bc\rvert)u$
▷ *Arithmetic operations in double precision.*
$e_2 = \mathrm{RN}(a + b \times c, u)$ $\qquad e_2^* = a + bc$
$\tilde{e}_2 = (a + bc)(1 + \epsilon_{a+b\times c}^{[u]})$
$\theta_{\mathrm{lin}}^{(e_2)} = (a + bc)\epsilon_{a+b\times c}^{[u]}$
$\lvert\theta_{\mathrm{lin}}^{(e_2)}\rvert \leqslant \lvert a + bc\rvert u$
▷ *Fused multiply-add (FMA) in double precision.*
$e_3 = \mathrm{RN}(\mathrm{RN}(a, u') + b \times \mathrm{RN}(a, u'), u)$ $\qquad e_3^* = a + ba$
$\tilde{e}_3 = (a(1 + \epsilon_a^{[u']}) + ba(1 + \epsilon_a^{[u']}))(1 + \epsilon_{\mathrm{RN}(a,u')+b\times\mathrm{RN}(a,u')}^{[u]})$
$\theta_{\mathrm{lin}}^{(e_3)} = (a + ba)\epsilon_a^{[u']} + (a + ba)\epsilon_{\mathrm{RN}(a,u')+b\times\mathrm{RN}(a,u')}^{[u]}$
$\lvert\theta_{\mathrm{lin}}^{(e_3)}\rvert \leqslant \lvert a + ba\rvert(u + u')$
▷ *Fused multiply-add (FMA) in double precision, with input a rounded to single precision.*

Table 1: Evaluation error examples ($u = 2^{-53}$, $u' = 2^{-24}$).

Finally, automatic linearized evaluation error expressions θ_{lin}, such as in (6), are obtained using Algorithm 2. Specifically, for an arithmetic expression with roundings e, this algorithm recursively computes an expression of the form $\theta_{\mathrm{lin}} = \sum_{i=1}^{k} \theta_{\mathrm{lin},i}\epsilon_{e_i}^{[u_i]}$, with symbolic $\epsilon_{e_i}^{[u_i]}$ ($i \in [1\,..\,k]$) for each term $\mathrm{RN}(e_i, u_i)$ in e. The coefficients $\theta_{\mathrm{lin},i}$ are arithmetic expressions depending only on the input variables in e. Note that $\mathrm{RN}(e_i, u_i)$ may occur several

times in e, but the error variable $\epsilon_{e_i}^{[u_i]}$ is unique since the rounding operation is deterministic. This allows to bound the (linearized) evaluation error as in (7).

Proposition 1 (Correctness of Algorithm 2)**.** *Let e be an arithmetic expression with roundings, and $\theta_{lin} = \sum_{i=1}^{k} \theta_{lin,i} \epsilon_{e_i}^{[u_i]}$ the linearized expression for the evaluation error returned by Algorithm 2. If $u_i \leqslant u$ for all $i \in [1\,..\,k]$, then:*

$$|\tilde{e} - e^*| \leqslant \sum_{i=1}^{k} |\theta_{lin,i}| u_i + \mathcal{O}\left(u^2\right), \qquad \text{as } u \to 0.$$

Usually, for polynomial evaluation schemes, the functions $\theta_{\mathrm{lin},i}$ are linear with respect to the coefficients $\boldsymbol{a}$ of $p(t)$, that is $u_i \theta_{\mathrm{lin},i} = \boldsymbol{\pi}_i(t)^T \boldsymbol{a}$ for some $\boldsymbol{\pi}_i(t) \in \mathbb{R}^{n+1}$. Hence we obtain a linearized bound of the evaluation error of the form:

$$|\theta_{\mathrm{lin}}(\boldsymbol{a}, t)| \leqslant \sum_{i=1}^{k} |\boldsymbol{\pi}_i(t)^T \boldsymbol{a}|, \quad \boldsymbol{a} \in \mathbb{R}^{n+1},\ t \in \mathbb{R}. \tag{8}$$

We denote the right-hand side of equation (9) by

$$\theta(\boldsymbol{a}, t) = \sum_{i=1}^{k} |\boldsymbol{\pi}_i(t)^T \boldsymbol{a}|, \quad \boldsymbol{a} \in \mathbb{R}^{n+1},\ t \in \mathbb{R}. \tag{9}$$

Example 1 (Bound evaluation error for Horner)**.** *In particular, for Horner evaluation, we have from equation* (7)*:*

$$\begin{aligned}
\boldsymbol{\pi}_1(t)^T &= (u, ut, \ldots, ut^{n-1}, ut^n) \\
\boldsymbol{\pi}_2(t)^T &= (0, 2ut, \ldots, 2ut^{n-1}, 2ut^n) \\
&\ldots \\
\boldsymbol{\pi}_n(t)^T &= (0, 0, \ldots, 2ut^{n-1}, 2ut^n) \\
\boldsymbol{\pi}_{n+1}(t)^T &= (0, 0, \ldots, 0, ut^n)
\end{aligned}$$

Algorithm 2 LinEvalError(e)

Input: e an arithmetic expression with explicit roundings.
Output: θ_{lin} the linearized evaluation error of e.

if $e \in \mathcal{V}$ **then**
  return 0
else if $e = \text{RN}(f, u)$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  return $\theta'_{\text{lin}} + f^* \epsilon_f^{[u]}$
else if $e = -f$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  return $-\theta'_{\text{lin}}$
else if $e = f + g$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  $\theta''_{\text{lin}} \leftarrow$ LinEvalError(g)
  return $\theta'_{\text{lin}} + \theta''_{\text{lin}}$
else if $e = f \times g$ **then**
  $\theta'_{\text{lin}} \leftarrow$ LinEvalError(f)
  $\theta''_{\text{lin}} \leftarrow$ LinEvalError(g)
  return $g^* \theta'_{\text{lin}} + f^* \theta''_{\text{lin}}$
end if

3 Semi-Infinite Programming formulation

3.1 Problem (P_{general}) as a linear SIP

Noting that Problem (P_{general}) is a piecewise-linear optimization problem and using the convex evaluation error formula $\theta(\boldsymbol{a}, t)$ at point $t \in [t_l, t_r]$ obtained in Section 2, Problem (P_{general}) becomes Problem (P'_{general}) (see [3, Section 4.3.1] for instance), with the compact index set $I = [t_l, t_r]$ and the monomial basis $\boldsymbol{\pi_0}(t) = (1, \ldots, t^n)^T$.

$$\begin{array}{ll} \min\limits_{(\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}} & \overline{a} \\ \text{s.t.} & |f(t) - \boldsymbol{\pi_0}(t)^T \boldsymbol{a}| + \theta(\boldsymbol{a}, t) - \overline{a} \leqslant 0, \quad t \in I. \end{array} \tag{P'_{general}}$$

Problem (P'_{general}) is a convex Semi-Infinite Programming (SIP) problem (see [24] which provides a comprehensive overview of SIP) that can be reformulated as a linear SIP problem, at the expense of a different index set Ω replacing the previous index set I. Here, the set of constraints of (P'_{general}) involving absolute values is replaced by as many linear constraints as required

to represent all possible sign combinations. More precisely, the evaluation error is as in equation (9), $\theta(\boldsymbol{a},t)=\sum_{i=1}^{k}|\boldsymbol{a}^T\boldsymbol{\pi_i}(t)|$, and define:

$$\begin{gathered}\boldsymbol{x}=(\overline{a},\boldsymbol{a})\in\mathbb{R}^{n+2},\ \boldsymbol{z}=(1,0,\ldots,0)\in\mathbb{R}^{n+2},\\ \boldsymbol{\alpha}(t,\sigma_0,\ldots,\sigma_k)=(1,\sigma_0\boldsymbol{\pi_0}^T(t)+\sum_{i=1}^{k}\sigma_i\boldsymbol{\pi_i}^T(t))^T\in\mathbb{R}^{n+2},\\ \mathfrak{S}=\{-1,0,1\}^{k+1},\ \omega=(t,\sigma_0,\ldots,\sigma_k)\in\Omega:=I\times\mathfrak{S}.\end{gathered}\tag{10}$$

Then, Problem (P'_{general}) is exactly given by the following linear SIP:

$$\begin{array}{ll}\min\limits_{\boldsymbol{x}\in\mathbb{R}^d} & \boldsymbol{z}^T\boldsymbol{x}\\ \text{s.t.} & \boldsymbol{\alpha}(\omega)^T\boldsymbol{x}\geqslant c(\omega),\quad \omega\in\Omega,\end{array}\tag{P}$$

where $d=n+2$, $c(\omega)=\sigma_0 f(t)$, Ω is a compact metric space and the function $g(\boldsymbol{x},\omega)=c(\omega)-\boldsymbol{\alpha}(\omega)^T\boldsymbol{x}\leqslant 0$ defining the feasible set is a continuous function from $\mathbb{R}^{n+2}\times\Omega$ into $\mathbb{R}$. Note that for $\mathfrak{S}'=\{-1,1\}\times\{0\}^k$ and $\Omega'=I\times\mathfrak{S}'\subseteq\Omega$, ($P_{\text{minimax}}$) is exactly retrieved as shown in the next example.

Example 2. *For $n=5$, Problem (P_{minimax}) is:*

$$\begin{array}{ll}\min\limits_{(\overline{a},\boldsymbol{a})\in\mathbb{R}^7} & \overline{a}\\ \text{s.t.} & (1,\sigma_0 1,\sigma_0 t,\ldots,\sigma_0 t^5)(\overline{a},a_0,a_1,\ldots,a_5)^T\geqslant\sigma_0 f(t),\\ & \sigma_0=\mp 1,\quad t\in I.\end{array}\tag{Example 2 (a))}$$

while Problem (P'_{general}), assuming Horner evaluation is:

$$\begin{array}{ll}\min\limits_{(\overline{a},\boldsymbol{a})\in\mathbb{R}^7} & \overline{a}\\ \text{s.t.} & (1,\sigma_0+\sigma_1 u,(\sigma_0+\sigma_1 u+\sigma_2 2u)t,\ldots,\\ & (\sigma_0+\sigma_1 u+\ldots+\sigma_5 u)t^5)(\overline{a},a_0,a_1,\ldots,a_5)^T\\ & \qquad\geqslant\sigma_0 f(t),\\ & \sigma_0=\mp 1,\sigma_1=\mp 1,\ldots,\sigma_5=\mp 1,\quad t\in I.\end{array}\tag{Example 2 (b))}$$

We propose in Section 4 an exchange algorithm which solves Problem (P'_{general}) and can be seen as a generalization (in the above framework) of the classical Remez algorithm, which solves Problem (P_{minimax}). In order to prove its correctness, we need some important discretization properties of linear SIP problems, which are based on conjugate duality theory. Our presentation closely follows the survey [26].

3.2 Duality and Discretization for SIP

For a Problem (P), we denote respectively by $\mathrm{val}(P)$ and $\mathrm{Sol}(P)$, its optimal value and the set of its optimal solutions.

A discretization (P_m) of (P) for a set $\boldsymbol{\omega} = \{\omega_1, \ldots, \omega_m\} \subseteq \Omega$ is the following linear program:

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}\in\mathbb{R}^d} & \boldsymbol{z}^T\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{\alpha}(\omega_j)^T\boldsymbol{x} \geqslant c(\omega_j), \quad j = 1, \cdots m. \end{array} \tag{P_m}$$

Since the feasible set of (P) is included in the feasible set of (P_m), we have that $\mathrm{val}(P_m) \leqslant \mathrm{val}(P)$. The existence of a discretization (P_m) such that the equality holds is a particularly appealing feature of some linear SIPs since the solution of (P) may be obtained by the solution of (P_m) if we are able to find the corresponding set $\boldsymbol{\omega}$.

Definition 1. *[26]* (P) *is said to be* reducible *if there exists a discretization* (P_m) *defined by the subset* $\{\omega_1, \ldots, \omega_m\} \subseteq \Omega$ *such that* $\mathrm{val}(P_m) = \mathrm{val}(P)$.

The characterization of reducible SIP problems relies on the central notion of *duality* that rules the interplay between two optimization problems. This notion has its roots in the interrelations between a normed linear space and its topological dual. Let us define the continuous mapping $h : \boldsymbol{x} \mapsto g(\boldsymbol{x}, \cdot)$ from $\mathbb{R}^d$ to the Banach space of continuous functions $\mathcal{C}(\Omega)$, equipped with the uniform norm $\|h\|_\Omega = \sup_{\omega\in\Omega} |h(\omega)|$, then the topological dual of $\mathcal{C}(\Omega)$ is the space $\mathcal{C}(\Omega)^*$ of signed Borel measures μ over $(\Omega, \mathcal{B}(\mathbb{R}^{k+2}))$ [16, Section 21.5]. For a measure $\mu \in \mathcal{C}(\Omega)^*$, its support is the smallest closed subset Γ of Ω such that $|\mu|(\Omega \setminus \Gamma) = 0$. A positive measure μ is denoted by $\mu \succeq 0$. A classical example of a positive measure with discrete support is the Dirac measure of support $\{\omega_j\}$:

$$\delta_{\omega_j}(A) = \begin{cases} 0 & \text{if} \quad \omega_j \notin A, \\ 1 & \text{if} \quad \omega_j \in A. \end{cases} \qquad A \subseteq \Omega. \tag{11}$$

Defining the bilinear form pairing $\mathcal{C}(\Omega)$ and $\mathcal{C}(\Omega)^*$ by the duality bracket:

$$\langle h, \mu \rangle = \int_\Omega h(\omega) d\mu(\omega), \tag{12}$$

the *dual* problem (D) associated to the *primal* problem (P) is obtained as:

$$\begin{array}{ll} \max\limits_{\mu\succeq 0} & \displaystyle\int_\Omega c(\omega)\mathrm{d}\mu(\omega) \\ \text{s.t.} & \displaystyle\int_\Omega \boldsymbol{\alpha}(\omega)\mathrm{d}\mu(\omega) = \boldsymbol{z}. \end{array} \tag{D}$$

The *weak duality* property, that is $\mathrm{val}(D) \leqslant \mathrm{val}(P)$ always holds. Problem (D) is an LP problem defined in the space of positive measures which is hard to solve. By restricting the support of $\mu \succeq 0$ to $\{\omega_1, \ldots, \omega_m\}$, that is $\mu = \sum_{j=1}^m y_j \delta_{\omega_j}$ with $y_j \geqslant 0$, a discretized counterpart (D_m) of (D) is obtained:

$$\begin{array}{ll} \max\limits_{\substack{y_j \geqslant 0 \\ j \in [1..m]}} & \sum\limits_{j=1}^m c(\omega_j) y_j \\ \text{s.t.} & \sum\limits_{j=1}^m y_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{z}, \end{array} \tag{D_m}$$

with $\mathrm{val}(D_m) \leqslant \mathrm{val}(D)$. It is important to note that the LP dual of the discretized problem (P_m) is exactly (D_m) which implies that $\mathrm{val}(D_m) = \mathrm{val}(P_m)$ (strong duality holds) provided that none of (P_m) or (D_m) is infeasible.

So far, under these mild assumptions, we have that $\mathrm{val}(D_m) = \mathrm{val}(P_m) \leqslant \mathrm{val}(D) \leqslant \mathrm{val}(P)$ and conditions for having only equalities (respectively reducibility and strong duality properties) may be obtained by using conjugate duality theory as developed in [26, Theorems 2.2, 2.3 and 3.2].

Theorem 1. *[26, Thm. 2.2, 2.3, 3.2] Under the assumptions:*

A1 *Ω is a compact metric space, $\boldsymbol{\alpha} : \Omega \to \mathbb{R}^d$ and $c : \Omega \to \mathbb{R}$ are continuous functions;*

A2 *$\mathrm{val}(P)$ is finite;*

A3 *(Slater's condition): there exists $\boldsymbol{x}^\circ$ such that:*

$$\boldsymbol{\alpha}(\omega)^T \boldsymbol{x}^\circ > c(\omega), \quad \textit{for all } \omega \in \Omega; \tag{13}$$

A4 *There exist $\omega_1, \ldots, \omega_d \in \Omega$ with $(\boldsymbol{\alpha}(\omega_1), \cdots, \boldsymbol{\alpha}(\omega_d))$ linearly independent such that:*

$$\exists\ y_1, \ldots, y_d > 0, \quad \boldsymbol{z} = \sum_{j=1}^d y_j \boldsymbol{\alpha}(\omega_j), \tag{14}$$

the following statements are true:

(i) *$\mathrm{Sol}(P) \neq \emptyset$ and bounded;*

(ii) *$\mathrm{Sol}(D) \neq \emptyset$ and bounded;*

(iii) *Problem (P) is reducible to a Problem (P_m) with $m \leqslant d$;*

(iv) $\text{val}(P) = \text{val}(D) = \text{val}(P_m) = \text{val}(D_m)$.

Proposition 2. *Assumptions A1-A4 are satisfied for our Problem* (P'_{general}) *and therefore results (i)-(iv) of Theorem 1 apply.*

Proof.

A1 By construction, our set Ω is a compact metric space and α and c are polynomials and therefore continuous on Ω;

A2 $\text{val}(P) = +\infty$ means that the primal problem (P'_{general}) is not feasible but $\boldsymbol{x} = (\max_{t\in I} |f(t)|, 0 \cdots, 0)$ is a feasible point for (P'_{general}), therefore $\text{val}(P) < +\infty$. In addition, $\text{val}(P) > -\infty$ since $\overline{a} \geqslant 0$ by construction and for all feasible points of (P'_{general});

A3 It may be easily deduced from the proof of A2 that $\boldsymbol{x}^\circ = (\max_{t\in I} |f(t)| + \varsigma, 0 \cdots, 0)$ is a strictly feasible point for (P'_{general}) for any $\varsigma > 0$;

A4 An instance for $\{\omega_1, \ldots, \omega_{n+2}\}$ is provided by Algorithm INIT in Section 4 (see Lemma 1 in Appendix .1).

□

The conclusion of this theorem, that is both (P) and (D) are reducible to a discretization of size of at most d, allows us to recast the problem of solving (P'_{general}) as the problem of finding the right discretization $\{\omega_1, \ldots, \omega_d\}$ such that item (iv) of Theorem 1 applies and to solve the associated (P_m) and/or (D_m). This goal may be reached by tailoring the general exchange algorithm for semi-infinite linear programs presented in [7] to our specific case.

This algorithm can be seen as a generalization of the dual simplex algorithm for Problem (D). The main idea consists first in finding at each iteration ℓ, the solution $\boldsymbol{y}^{(\ell)}$ of $(D^{(\ell)}_{n+2})$, with $\boldsymbol{\omega}^{(\ell)} = \{\omega_j{}^{(\ell)}\}_{j=1}^{n+2}$. Such a solution is a feasible (but not necessarily optimal) point of the dual Problem (D). Moreover, the objective value $\mathbf{z}^T\mathbf{x}^{(\ell)}$ of $(P^{(\ell)}_m)$ and (P) for the instance $\boldsymbol{x}^{(\ell)} := (\overline{a}^{(\ell)}, \boldsymbol{a}^{(\ell)})$ is equal to the objective value of (D) for the instance $\boldsymbol{y}^{(\ell)}$[3]. Hence, either $\boldsymbol{x}^{(\ell)}$ is a feasible solution of Problem (P) by Theorem 1, or it is an infeasible point of Problem (P). In the latter case, one of these constraints is replaced by a new one, indexed by $\omega_*{}^{(\ell)}$, in an exchange step in order to increase the objective value of the dual and works towards primal feasibility.

[3]The feasible set of (P) is included in the feasible set of $(P^{(\ell)}_m)$, for all ℓ.

4 Iterative exchange algorithm

Algorithm Eval&ApproxOptimize computes the degree-n best polynomial approximation with respect to both evaluation and approximation errors, i.e. it solves (P'_{general}) based on the theoretical developments from Section 3.2. Before entering the details, we provide an analogy with Remez algorithm. Roughly speaking, the new algorithm consists of the same main steps:

- Init provides *a good set of initial points*.
- At each step, SolvePrimal solves a linear system of equations (built w.r.t. the current set of points), where the variables are the polynomial coefficients.
- Then, FindNewIndex finds a new point where the *total error* is maximal.
- Finally, Exchange replaces one point from the current set with this new point.
- This process repeats until a certain tolerance threshold is reached concerning the *total error*.

However, when considering both errors, one can not only rely on the primal problem (coefficients reconstruction), but also needs the dual problem. This implies:

- Besides classical points, a combination of signs (signatures) is required at each step.
- Init and Exchange need the solution of the dual problem.

A running example for this algorithm is given in Section 5. We focus now on its correctness, which is stated in Theorem 2. For this, one needs an assumption on the dual solution, which always holds in the Remez algorithm. It is not proven in our setting, but it never failed in practice.

Assumption 1. *At each iteration ℓ, the solution $\boldsymbol{y}^{(\ell)}$ of the dual discretized Problem $(D^{(\ell)}_{n+2})$ is an interior point, that is $y^{(\ell)}_j > 0$ for all $j \in [1\,..\,n+2]$.*

Theorem 2. *Let f be a continuous function over an interval $I = [t_l, t_r]$, a degree $n \geqslant 0$, a linearized evaluation error bound θ and a tolerance parameter*

$\tau > 0$. *Under Assumption 1,* EVAL&APPROXOPTIMIZE(f, n, I, θ, τ) *terminates and returns a degree-n polynomial approximation for* f *with a total error* ε *(approximation and evaluation) satisfying:*

$$\varepsilon^* \leqslant \varepsilon \leqslant (1+\tau)\varepsilon^*, \tag{15}$$

where ε^* *is the total error of the best degree-n polynomial approximation of* f*.*

Proof. • First, we prove by induction that the following properties hold at each iteration $\ell \geqslant 0$:

(i) $\{\boldsymbol{\alpha}(\omega_j^{(\ell)})\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$;

(ii) $\boldsymbol{y}^{(\ell)}$ is the optimal solution of Problem (D_m) for $\boldsymbol{\omega}^{(\ell)}$;

(iii) $\boldsymbol{x}^{(\ell)}$ is the optimal solution of Problem (P_m) for $\boldsymbol{\omega}^{(\ell)}$;

(iv) $\omega_*^{(\ell)} = \arg\max_{\omega\in\Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T \boldsymbol{x}^{(\ell)}\right)$;

using the correctness Lemmas given in Appendix .1. For $\ell = 0$, INIT(n, I) returns $\boldsymbol{\omega}^{(0)}$, $\boldsymbol{y}^{(0)}$ satisfying (i) and (ii). Then SOLVEPRIMAL$(f, n, \theta, \boldsymbol{\omega}^{(0)})$ computes $\boldsymbol{x}^{(0)} = (\overline{a}^{(0)}, \boldsymbol{a}^{(0)})$ satisfying (iii). Finally, FINDNEWINDEX$(f, n, I, \theta, \boldsymbol{a}^{(0)})$ gives $\omega_*^{(0)}$, $\overline{a}_*^{(0)}$ satisfying (iv).

For the inductive step, EXCHANGE$(n, \theta, \boldsymbol{\omega}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \omega_*^{(\ell)})$ computes $\boldsymbol{\omega}^{(\ell+1)}$, $\boldsymbol{y}^{(\ell+1)}$ satisfying (i) and (ii), by induction hypothesis on $\boldsymbol{\omega}^{(\ell)}$, $\boldsymbol{y}^{(\ell)}$, $\omega_*^{(\ell)}$. Then, SOLVEPRIMAL$(f, n, \theta, \boldsymbol{\omega}^{(\ell+1)})$ and FINDNEWINDEX$(f, n, I, \theta, \boldsymbol{a}^{(\ell+1)})$ compute $\boldsymbol{x}^{(\ell+1)}$, $\omega_*^{(\ell+1)}$, $\overline{a}_*^{(\ell+1)}$ satisfying (iii) and (iv).

• Moreover, at each iteration ℓ, we have $\overline{a}^{(\ell)} \leqslant \varepsilon^* \leqslant \overline{a}_*^{(\ell)}$. Indeed, $\boldsymbol{x}^{(\ell)}$ is the optimal solution of the discretized Problem (P_m) for $\boldsymbol{\omega}^{(\ell)}$, whose objective value $\overline{a}^{(\ell)}$ is less or equal to the optimal value ε^* of Problem (P). On the other side, $\overline{a}_*^{(\ell)}$ is the total error of degree-n polynomial $\boldsymbol{a}^{(\ell)T}\boldsymbol{\pi}_{\boldsymbol{0}}(t)$ and therefore, it is greater or equal to the optimal error ε^*. In addition, Lemma 5 in the Appendix proves $\overline{a}^{(\ell)} \leqslant \overline{a}^{(\ell+1)}$.

• Finally, the convergence of this iterative process is proved by [7, Theorem 2.1], relying on Assumption 1. Hence, Algorithm EVAL&APPROXOPTIMIZE terminates at some iteration ℓ, with $\overline{a}_*^{(\ell)} \leqslant (1+\tau)\overline{a}^{(\ell)}$, yielding the enclosure (15). □

Algorithm 3 EVAL&APPROXOPTIMIZE(f, n, I, θ, τ)

Input: function f, $n \geqslant 0$, I, $\theta(\boldsymbol{a}, t)$ as in (9), $\tau > 0$.
Output: $(\overline{a}, \boldsymbol{a})$ solution of Problem (P) within accuracy τ.

▷ *Initialization*
$(\boldsymbol{\omega}^{(0)}, \boldsymbol{y}^{(0)}) \leftarrow$ INIT(n, I)
$(\overline{a}^{(0)}, \boldsymbol{a}^{(0)}) \leftarrow$ SOLVEPRIMAL$(f, n, \theta, \boldsymbol{\omega}^{(0)})$
$({\omega_*}^{(0)}, \overline{a}_*^{(0)}) \leftarrow$ FINDNEWINDEX$(f, n, I, \theta, \boldsymbol{a}^{(0)})$
$\ell \leftarrow 0$
▷ *Iterate while accuracy τ not reached*
while $\overline{a}_*^{(\ell)} / \overline{a}^{(\ell)} > 1 + \tau$ **do**
    $(\boldsymbol{\omega}^{(\ell+1)}, \boldsymbol{y}^{(\ell+1)}) \leftarrow$ EXCHANGE$(n, \theta, \boldsymbol{\omega}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \omega_*^{(\ell)})$
    $(\overline{a}^{(\ell+1)}, \boldsymbol{a}^{(\ell+1)}) \leftarrow$ SOLVEPRIMAL$(f, n, \theta, \boldsymbol{\omega}^{(\ell+1)})$
    $({\omega_*}^{(\ell+1)}, \overline{a}_*^{(\ell+1)}) \leftarrow$ FINDNEWINDEX$(f, n, I, \theta, \boldsymbol{a}^{(\ell+1)})$
    $\ell \leftarrow \ell + 1$
end while
return $(\overline{a}^{(\ell)}, \boldsymbol{a}^{(\ell)})$

Algorithm 4 INITPOINTS(n, I)

Input: $n \geqslant 0$, $I = [t_l, t_r]$.
Output: $\boldsymbol{\omega} \in \Omega^{n+2}$ and solution $\boldsymbol{y}$ of Problem (D_m).

▷ *Initialize with Chebyshev nodes and Remez constraints*
for j **in** $[1\,..\,n+2]$ **do**
    $t_j \leftarrow \frac{t_l + t_r}{2} + \cos\left(\frac{(j-1)\pi}{n+1}\right) \frac{t_l - t_r}{2}$
    $\boldsymbol{\sigma}_j \leftarrow ((-1)^j, 0, \ldots, 0)$
    $\omega_j \leftarrow (t_j, \boldsymbol{\sigma}_j)$
end for
▷ *Compute dual solution*
Solve for $\boldsymbol{y}$ the linear system $\sum_{j=1}^{n+2} y_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{z}$
return $(\boldsymbol{\omega}, \boldsymbol{y})$

Algorithm 5 SolvePrimal$(f, n, \theta, \boldsymbol{\omega})$

Input: function f, $n \geqslant 0$, θ the evaluation error, $\boldsymbol{\omega} \in \Omega^{n+2}$.
Output: $(\overline{a}, \boldsymbol{a})$ solution of Problem (P_m) for $\boldsymbol{\omega}$.

1: Solve for $(\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}$ the linear system:

$$\boldsymbol{\alpha}(\omega_j)^T(\overline{a}, \boldsymbol{a}) = c(\omega_j), \quad j \in [1\,..\,n+2]$$

2: **return** $(\overline{a}, \boldsymbol{a})$

Algorithm 6 FindNewIndex$(f, n, I, \theta, \boldsymbol{a})$

Input: function f, $n \geqslant 0$, $I = [t_l, t_r]$, θ the evaluation error as in (9), coefficients $\boldsymbol{a} \in \mathbb{R}^{n+1}$.
Output: $(\omega_*, \overline{a}_*)$ with $\omega_* = (t_*, \boldsymbol{\sigma}_*) \in \Omega$

▷ *Compute maximal error in absolute value*

1: $t_* \leftarrow \underset{t_l \leqslant t \leqslant t_r}{\arg\max} |\boldsymbol{a}^T \boldsymbol{\pi}_0(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T \boldsymbol{\pi}_i(t)|$

2: $\overline{a}_* \leftarrow \underset{t_l \leqslant t \leqslant t_r}{\max} |\boldsymbol{a}^T \boldsymbol{\pi}_0(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T \boldsymbol{\pi}_i(t)|$

▷ *Reconstruct signature*

3: $\sigma_{*0} \leftarrow -\text{sign}(\boldsymbol{a}^T \boldsymbol{\pi}_0(t_*) - f(t_*))$

4: $\sigma_{*i} \leftarrow -\text{sign}(\boldsymbol{a}^T \boldsymbol{\pi}_i(t_*)), \quad i \in [1\,..\,k]$

5: $\omega_* \leftarrow (t_*, \boldsymbol{\sigma}_*)$

6: **return** $(\omega_*, \overline{a}_*)$

Algorithm 7 EXCHANGE$(n, \theta, \boldsymbol{\omega}, \boldsymbol{y}, \omega_*)$

Input: $n \geqslant 0$, θ the evaluation error, $\boldsymbol{\omega} \in \Omega^{n+2}$, dual solution $\boldsymbol{y} \in \mathbb{R}^{n+2}$, new index $\omega_* \in \Omega$.

Output: new set $\boldsymbol{\omega'} \in \Omega^{n+2}$ and dual solution $\boldsymbol{y'} \in \mathbb{R}^{n+2}$.

Solve for $\boldsymbol{\gamma} \in \mathbb{R}^{n+2}$ the linear system:

$$\sum_{j=1}^{n+2} \gamma_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{\alpha}(\omega_*)$$

▷ *Exiting index*

$j_0 \leftarrow \arg\min \left\{ \frac{y_j}{\gamma_j} \;\middle|\; \gamma_j > 0 \right\}$

▷ *Update dual solution*

$\tilde{y}_* \leftarrow \frac{y_{j_0}}{\gamma_{j_0}}$

$\tilde{y}_j \leftarrow y_j - \gamma_j \tilde{y}_*, \quad j \in [1\,..\,n+2]$

$\{(\boldsymbol{\omega'_j}, y'_j)\} \leftarrow \{(\omega_j, \tilde{y}_j), j \in [1\,..\,n+2] - \{j_0\} \cup \{*\}\}$,

return $(\boldsymbol{\omega'}, \boldsymbol{y'})$

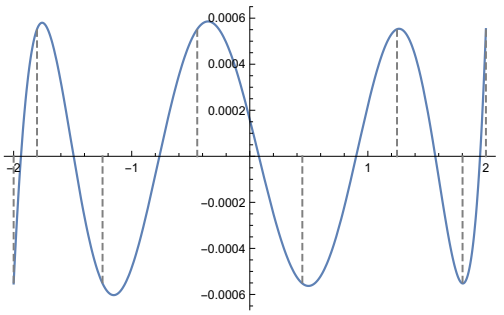
5 Examples and conclusion

To illustrate Algorithm EVAL&APPROXOPTIMIZE, a tutorial example for Airy special function is discussed (further comparisons in binary32 case are given in Appendix .3). Then approximations with binary64 coefficients of arcsin are presented.

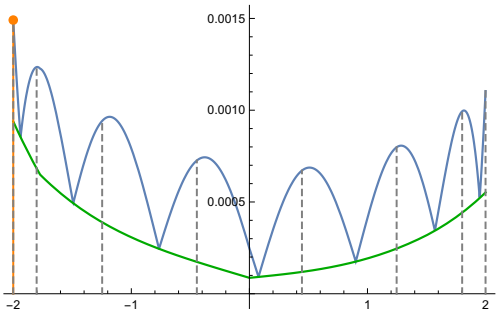
Example 3 (Airy function). Let Ai over $I = [-2, 2]$, approximated by a polynomial of degree $n = 6$, evaluated using the Horner scheme with $u = 2^{-12}$. The terms $\{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_7\}$ defining the evaluation error θ are given in Example 1. We fix a tolerance $\tau = 0.01$.

At iteration 0 (Figure 1), the points $t_j^{(0)}$ are initialized with the Chebyshev nodes and the signatures $\boldsymbol{\sigma_j}^{(0)}$ define a Remez-like system of linear equations on the coefficients of the polynomial (Figure 1d). Its solution $\boldsymbol{x}^{(0)} = (\overline{a}^{(0)}, \boldsymbol{a}^{(0)})$ defines a polynomial $p^{(0)}(t) = \boldsymbol{a}^{(0)T}\boldsymbol{\pi_0}(t)$, whose approximation error is depicted in Figure 1a. It exhibits quasi-equioscillations indicating that $\boldsymbol{p}^{(0)}$ is rather close to the degree-6 minimax approximation of Ai over I. However, the total error is more important near -2 and 2 (Figure 1b), due to the evaluation depicted in green. In particular, the algorithm detects the maximum error at $t_*^{(0)} = -2$ (in orange). Note that $t_1^{(0)}$ was already equal to -2, but $\omega_1^{(0)} \neq \omega_*^{(0)}$ since the signatures are different. To perform the exchange, the dual solution is needed (Figure 1c). It is a positive

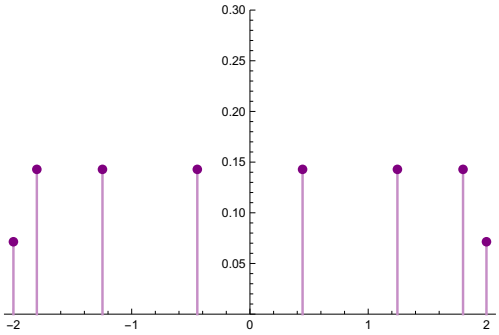
combination of Dirac measures supported on the finite set $\boldsymbol{\omega}^{(0)}$.



(a) Approximation error



(b) Total error

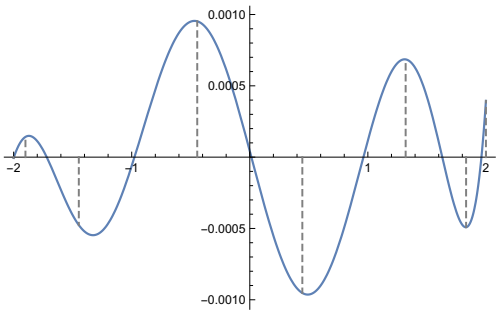


(c) Dual solution

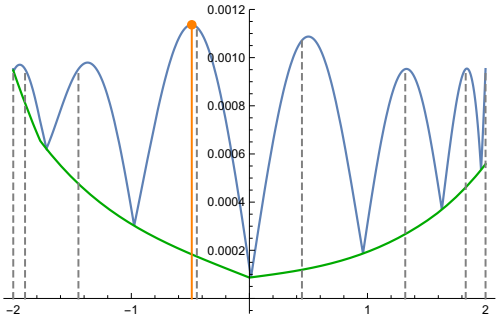
t_j		σ_{j0}	σ_{j1}	σ_{j2}	σ_{j3}	σ_{j4}	σ_{j5}	σ_{j6}	σ_{j7}
-2.	+	o	o	o	o	o	o	o	
-1.803	-	o	o	o	o	o	o	o	
-1.247	+	o	o	o	o	o	o	o	
-0.445	-	o	o	o	o	o	o	o	
0.445	+	o	o	o	o	o	o	o	
1.247	-	o	o	o	o	o	o	o	
1.802	+	o	o	o	o	o	o	o	
2.	-	o	o	o	o	o	o	o	

(d) Points and signatures

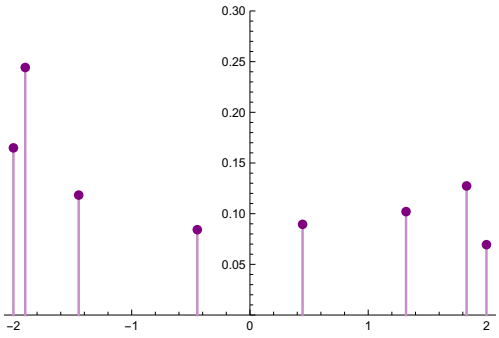
Figure 1: Approximation of Ai over $[-2, 2]$: iteration 0



(a) Approximation error



(b) Total error

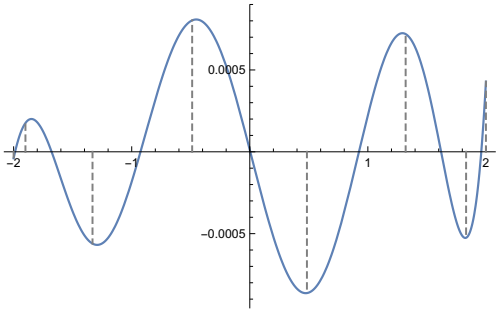


(c) Dual solution

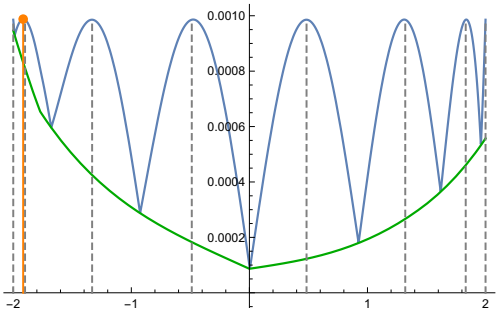
t_j	σ_{j0}	σ_{j1}	σ_{j2}	σ_{j3}	σ_{j4}	σ_{j5}	σ_{j6}	σ_{j7}
-2.	+	-	-	+	+	+	+	-
-1.9	-	-	-	+	+	+	+	-
-1.448	+	-	-	+	+	+	-	-
-0.445	-	o	o	o	o	o	o	o
0.445	+	o	o	o	o	o	o	o
1.32	-	-	+	+	-	-	+	-
1.832	+	-	-	+	-	-	+	-
2.	-	-	-	+	-	-	+	-

(d) Points and signatures

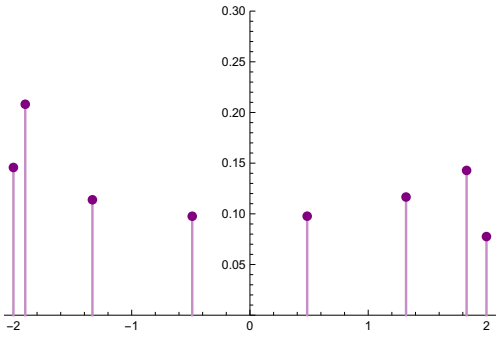
Figure 2: Approximation of Ai over $[-2, 2]$: iteration 6



(a) Approximation error



(b) Total error



(c) Dual solution

t_j	σ_{j0}	σ_{j1}	σ_{j2}	σ_{j3}	σ_{j4}	σ_{j5}	σ_{j6}	σ_{j7}
-2.	+	-	-	+	+	+	+	-
-1.9	-	-	-	+	+	+	+	-
-1.332	+	-	-	+	+	+	-	-
-0.488	-	-	-	+	+	+	-	-
0.484	+	-	+	+	-	-	+	-
1.32	-	-	+	+	-	-	+	-
1.832	+	-	-	+	-	-	+	-
2.	-	-	-	+	-	-	+	-

(d) Points and signatures

Figure 3: Approximation of Ai over $[-2, 2]$: iteration 9

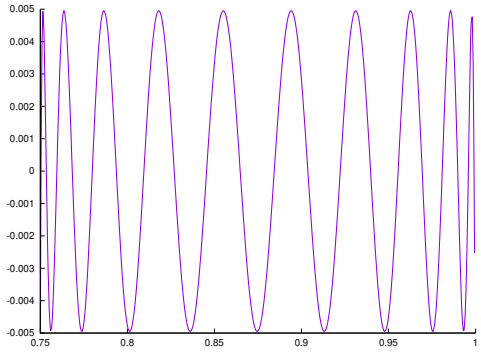
Moving forward to iteration 6 (Figure 2), the total error is more balanced, though still not optimal. Both the signatures and the approximation error are now completely different from the Remez solution.

Eventually, the algorithm stops at iteration 9 (Figure 3). Indeed, the maximum total error $\overline{a}_*^{(9)}$ (in orange) is less than 1% higher than the error $\overline{a}^{(9)}$ over the discrete set $\boldsymbol{\omega}^{(9)}$. Note that the total error reaches its maximum at $n+2 = 8$ points. This became possible by unbalancing the approximation error, namely reducing the amplitude of the oscillations near -2 and 2, at the cost of higher oscillations in the middle of I.

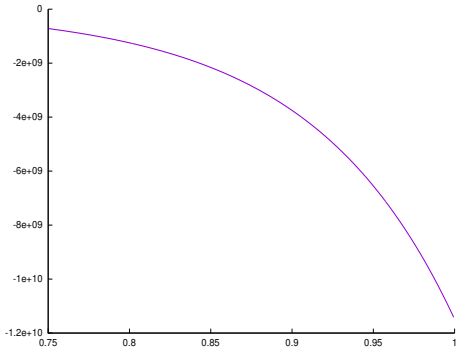
Example 4 (Arcsine function). Consider $f = \arcsin$, over the interval $I = [0.75; 1]$. This example is particularly insightful because f is ill-conditioned over I, and also because its values are close to 1, so absolute or relative error treatment is similar. Firstly, assume that argument reduction techniques are not available (this can be the case for any ill-conditioned function, known only through sampling via a black-box approach), operations are in binary64 and an approximation polynomial of degree 20 with binary64 coefficients is searched for.

Figure 4(a) shows the absolute approximation error between arcsin and the best approximation polynomial p_{Remez} with *real* coefficients, for which

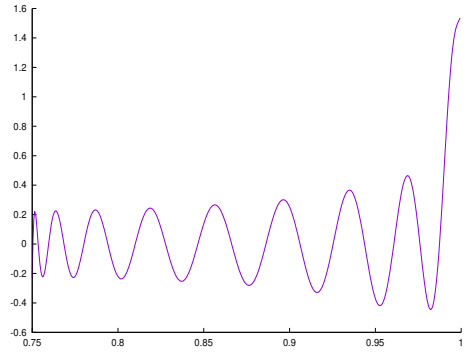
$\max_I |f - p_{\text{Remez}}| \leqslant 0.00496$. Then, the coefficients of p_{Remez} are directly rounded to binary64, which results in $\max_I |f - \text{roundCoeff}(p_{\text{Remez}})| \simeq 0.004961.15 \cdot 10^{10}$ (cf. Figure 4(b)). This is due to the high magnitude of the coefficients. To search for better coefficients, one could use the proficient FPMinimax routine of Sollya [11], which optimizes on the coefficient space of representable FP numbers. The approximation error in this case is shown in Figure 4(c) and one has $\max_I |f - p_{\text{FPMinimax})}| \simeq 1.58$. Finally, in Figure 4(d) we provide the plot of the total error between f and the obtained Eval&ApproxOpt polynomial (with binary64 coefficients), which provides the best bound in this case $\max_I |f - p_{\text{E\&A})}| \simeq 0.0081$. One can then actually prove with Gappa [14] that the Horner scheme evaluation of $p_{\text{E\&A}}$ has an absolute error less than 0.00035.



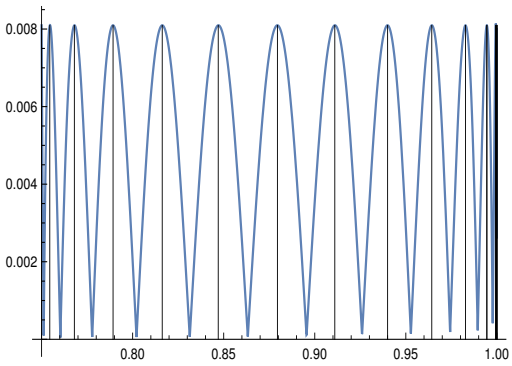
(a) $f - p_{\text{Remez}}$.



(b) $f - \text{roundCoeff}(p_{\text{Remez}})$.



(c) $f - p_{\text{FPMinimax}}$.



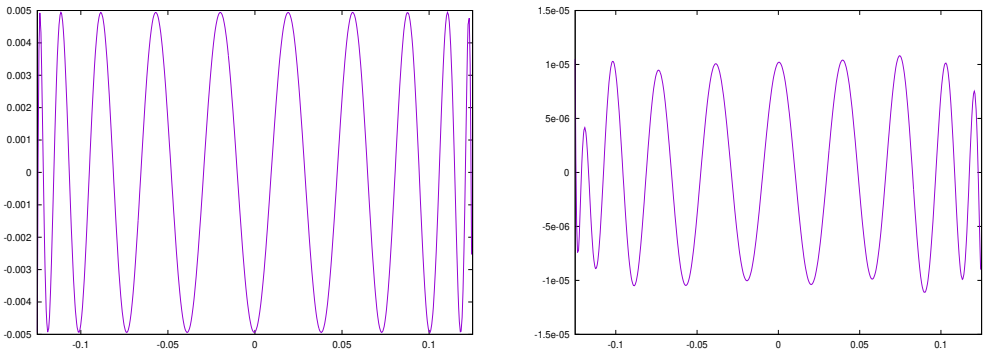
(d) Eval&ApproxOpt absolute error.

Figure 4: Error plots for different approximation polynomials of degree 20 for $f = \arcsin$ over $I = [0.75; 1]$.

Secondly, suppose that I is re-centered to $I_c = [-0.125; 0.125]$ and the other parameters remain unchanged. Thus, the function to be approximated is $f = \arcsin(x + c)$, where c is the middle point of I. In this case, it turns

out that all the methods presented above provide almost the same accuracy. In Figure 5(a), the plot of the absolute error between f and p_{Remez} is given: $\max_{I_c} |f - p_{\text{Remez}}| \leqslant 0.00496$; at the drawing scale, the plot is the same for the other errors and one has that $\max_{I_c} |f - \text{roundCoeff}(p_{\text{Remez}})|$, $\max_{I_c} |f - p_{\text{FPMinimax})}|$ and $\max_{I_c} |f - p_{\text{E\&A})}|$ are all very close in magnitude $\simeq 0.00496$. Similarly, one can prove with Gappa that the Horner scheme evaluation has an absolute error of $7.16227 \cdot 10^{-11}$ for every above-mentioned polynomial. Hence in this case, our algorithm does not improve the error bound. Yet, it shows (at least numerically) that if only binary64 computations and a Horner scheme are used, there is no other polynomial which performs better w.r.t. both evaluation and approximation errors. For completeness, in Figure 5(b) we show that FPMinimax and Eval&ApproxOpt polynomials are different. However, this difference is very small and this does not influence the number of correct bits provided $-\log_2(0.00496) \simeq 7.65$.

Finally, we mention that when other argument reduction techniques exist, and when the evaluation error is not an issue (very small intervals around zero), the FPMinimax method still provides better tuned FP coefficients. So this opens the question for several future extensions. A mixed-integer linear programming problem could be formulated in the provided optimization framework. However, a similar exchange procedure in this case is not obvious. Concerning precisions of the coefficients and operations, they can be variable, as mentioned in Section 2, but a more detailed study is needed to eventually take into account higher order error terms for the error estimation formula. The polynomial coefficients stay linear in such a formula, so the algorithm presented can be straightforwardly used in such a case. In addition, this formula directly allows for the estimation of evaluation errors for other numerical schemes and eventually polynomial bases [2] for which a practical study is necessary.



(a) $f - p_{\text{Remez}}$ (b) $p_{\text{FPMinimax}} - p_{\text{E\&A}}$.

Figure 5: Error plots for different approximation polynomials of degree 20 for $f = \arcsin(x + c)$ over $I_c = [-0.125, 0.125]$.

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. Courier Corp., 1964.

[2] R. Barrio, H. Jiang, and S. Serrano. A general condition number for polynomials. *SIAM J. Numer. Anal.*, 51(2):1280–1294, 2013.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[4] T. Braconnier and P. Langlois. From rounding error estimation to automatic correction with automatic differentiation. In *Automatic differentiation of algorithms*, pages 351–357. Springer, 2002.

[5] N. Brisebarre and S. Chevillard. Efficient polynomial L^∞ approximations. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 169–176, Montpellier, France, 2007.

[6] N. Brisebarre, J.-M. Muller, and A. Tisserand. Computing machine-efficient polynomial approximations. *ACM Trans. Math. Software*, 32(2):236–256, June 2006.

[7] C. Carasso. *L'algorithme d'échange en optimisation convexe*. PhD thesis, Université Joseph-Fourier-Grenoble I, 1973.

[8] C. Carasso and P. J. Laurent. Un algorithme général pour l'approximation au sens de Tchebycheff de fonctions bornées sur un ensemble quelconque. In *Approximation Theory*, number 556 in Lecture Notes in Mathematics, pages 99–121. Springer Berlin Heidelberg, 1976.

[9] B. L. Chalmers. The Remez exchange algorithm for approximation with linear restrictions. *Trans. Amer. Math. Soc.*, 223:103–131, 1976.

[10] S. Chevillard, J. Harrison, M. Joldeş, and C. Lauter. Efficient and accurate computation of upper bounds of approximation errors. *Theoret. Comput. Sci.*, 412(16):1523–1543, 2011.

[11] S. Chevillard, M. Joldeş, and C. Lauter. Sollya: An environment for the development of numerical codes. In *International Congress on Mathematical Software*, pages 28–31. Springer, 2010.

[12] C. Daramy-Loirat, D. Defour, F. de Dinechin, M. Gallet, N. Gast, C. Q. Lauter, and J.-M. Muller. CR-LIBM, a library of correctly-rounded elementary functions in double-precision. Technical report, LIP Laboratory, Arenaire team, Dec. 2006.

[13] E. Darulova and V. Kuncak. Towards a compiler for reals. *ACM Trans. Program. Lang. Syst.*, 39(2):8:1–8:28, Mar. 2017.

[14] M. Daumas and G. Melquiond. Certification of bounds on expressions involving rounded operators. *ACM Trans. Math. Software*, 37(1):Art. 2, 20, 2010.

[15] F. de Dinechin, C. Q. Lauter, and J.-M. Muller. Fast and correctly rounded logarithms in double-precision. *Theor. Inform. Appl.*, 41(1):85–102, 2007.

[16] R. Halsey and F. Patrick. *Real Analysis*. Prentice Hall, 2010.

[17] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-2008, Aug. 2008. Available at `http://ieeexplore.ieee.org/servlet/opac?punumber=4610933`.

[18] O. Kupriianova and C. Lauter. Metalibm: A mathematical functions code generator. In *International Congress on Mathematical Software*, pages 713–717. Springer, 2014.

[19] C. Lauter and M. Mezzarobba. Semi-automatic floating-point implementation of special functions. In J.-M. Muller, A. Tisserand, and J. Villalba, editors, *ARITH 22*, Lyon, France, 2015. IEEE.

[20] C. Q. Lauter. *Arrondi Correct de Fonctions Mathématiques*. PhD thesis, ÉNS de Lyon, Lyon, France, Oct. 2008.

[21] J.-M. Muller. *Elementary Functions, Algorithms and Implementation*. Birkhäuser, Boston, 3rd edition, 2016.

[22] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2018.

[23] J. Oliver. Rounding error propagation in polynomial evaluation schemes. *J. Comput. Appl. Math.*, 5(2):85–97, 1979.

[24] R. Reemtsen and J.-J. Rückmann. *Semi-Infinite Programming*, volume 25. Springer Science & Business Media, 1998.

[25] A. Rocca, V. Magron, and T. Dang. Certified Roundoff Error Bounds using Bernstein Expansions and Sparse Krivine-Stengle Representations. In *24th IEEE Symposium on Computer Arithmetic*. IEEE, 2017.

[26] A. Shapiro. Semi-infinite programming, duality, discretization and optimality conditions. *Optimization*, 58(2):133–161, 2009.

[27] A. Solovyev, M. S. Baranowski, I. Briggs, C. Jacobsen, Z. Rakamarić, and G. Gopalakrishnan. Rigorous estimation of floating-point round-off errors with symbolic taylor expansions. *ACM Trans. Program. Lang. Syst.*, 41(1):2:1–2:39, Dec. 2018.

[28] A. Tisserand. High-performance hardware operators for polynomial evaluation. *International Journal of High Performance Systems Architecture (IJHPSA)*, 1(1):14–23, 2007.

[29] G. A. Watson. The calculation of best restricted approximations. *SIAM J. Numer. Anal.*, 11(4):693–699, 1974.

.1 Complementary proofs for the exchange algorithm

This appendix provides auxiliary lemmas for the proof of Theorem 2.

Lemma 1 (Correctness of INIT)**.** INIT(n, I) *computes* $\boldsymbol{\omega} = \{\omega_j\}_{j=1}^{n+2} \in \Omega^{n+2}$ *and* $\boldsymbol{y} \in \mathbb{R}^{n+2}$ *satisfying:*

- $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ *is a basis of* $\mathbb{R}^{n+2}$*;*
- $\boldsymbol{y}$ *is the optimal solution of Problem* (D_m) *for* $\boldsymbol{\omega}$*;*
- $y_j > 0$ *for all* $j \in [1\,..\,n+2]$.

Note that Algorithm INIT essentially initializes the problem with Chebyshev nodes for heuristic efficiency and signatures corresponding to the classical Remez algorithm, without the evaluation error term.

Proof. Let $\boldsymbol{A}(\boldsymbol{\omega})$ denote the $(n+2)$ square matrix whose columns are the $\boldsymbol{\alpha}(\omega_j)$. Since $\boldsymbol{\sigma_j} = ((-1)^j, 0, \ldots, 0)$, we have

$$\boldsymbol{A}(\boldsymbol{\omega}) = \begin{pmatrix} 1 & \ldots & \ldots & 1 \\ -1 & \ldots & \ldots & (-1)^{n+2} \\ -t_1 & \ldots & \ldots & (-1)^{n+2} t_{n+2} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ -t_1^n & \ldots & \ldots & (-1)^{n+2} t_{n+2}^n \end{pmatrix}.$$

First, we prove the existence of a feasible point $\boldsymbol{y}$ in Problem (D_m) for $\boldsymbol{\omega}$, that is $\boldsymbol{A}(\boldsymbol{\omega})\boldsymbol{y} = \boldsymbol{z}$ and $\boldsymbol{y} \geqslant 0$. From Farkas' lemma [26], if such a $\boldsymbol{y}$ does not exist, then there exists $\boldsymbol{x} = (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}$ s.t. $\boldsymbol{z}^T\boldsymbol{x} = \overline{a} < 0$ and $\boldsymbol{A}(\boldsymbol{\omega})^T\boldsymbol{x} \geqslant 0$, that is

$$\overline{a} + (-1)^j \boldsymbol{a}^T \boldsymbol{\pi_0}(t_j) \geqslant 0, \qquad j \in [1\,..\,n+2].$$

Since $\overline{a} < 0$, this implies that $\operatorname{sign}(\boldsymbol{a}^T\boldsymbol{\pi_0}(t_j)) = (-1)^j$. But $\boldsymbol{a}^T\boldsymbol{\pi_0}(t)$ is a polynomial of degree at most n, hence it cannot strictly change signs $n+2$ times. Consequently, Problem (D_m) has a feasible point $\boldsymbol{y}$.

Now, suppose that the columns of $\boldsymbol{A}(\boldsymbol{\omega})$ are not linearly independent, or that $y_j = 0$ for some j. Both cases imply that there exists $J \subset [1\,..\,n+2]$ of size $n+1$, and $\tilde{\boldsymbol{y}} \in \mathbb{R}^{n+1}$ s.t. $\sum_{j \in J} \tilde{y}_j \boldsymbol{\alpha}(\omega_j) = \boldsymbol{z}$. In particular, by canceling the first component, the family $\{\boldsymbol{\pi_0}(t_j)\}_{j \in J}$ is linearly dependent. But the Vandermonde determinant of this system cannot vanish since the t_j

are pairwise distinct. Therefore, $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$, and $y_j > 0$ for all j.

Finally, since $\boldsymbol{A}(\boldsymbol{\omega})$ is invertible, $\boldsymbol{y}$ is the unique optimal feasible point of (D_m). □

Lemma 2 (Correctness of SolvePrimal)**.** *If $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$ and Problem (D_m) for $\boldsymbol{\omega}$ is feasible, then* SolvePrimal$(f, n, \theta, \boldsymbol{\omega})$ *computes the optimal solution $\boldsymbol{x} = (\overline{a}, \boldsymbol{a})$ of Problem (P_m).*

Proof. Algorithm SolvePrimal computes the solution $\boldsymbol{x} \in \mathbb{R}^{n+2}$ of $\boldsymbol{\alpha}(\omega_j)^T\boldsymbol{x} = c(\omega_j)$ for $j \in [1\,..\,n+2]$. We show that $\boldsymbol{x}$ is the optimal solution of Problem (P_m) for $\boldsymbol{\omega}$.

Let $\tilde{\boldsymbol{x}}$ be any feasible point in (P_m). Since the dual Problem (D_m) for $\boldsymbol{\omega}$ is feasible, there exists $\boldsymbol{y} \geqslant 0$ s.t. $\boldsymbol{z} = \sum_{j=1}^{n+2} y_j\boldsymbol{\alpha}(\omega_j)$. Then

$$\boldsymbol{z}^T\tilde{\boldsymbol{x}} \;=\; \sum_{j=1}^{n+2} y_j\boldsymbol{\alpha}(\omega_j)^T\tilde{\boldsymbol{x}} \;\geqslant\; \sum_{j=1}^{n+2} y_j c(\omega_j) \;=\; \sum_{j=1}^{n+2} y_j\boldsymbol{\alpha}(\omega_j)^T\boldsymbol{x} \;=\; \boldsymbol{z}^T\boldsymbol{x},$$

thereby establishing optimality of $\boldsymbol{x}$. □

Lemma 3 (Correctness of FindNewIndex)**.** *Given $\boldsymbol{x} = (\overline{a}, \boldsymbol{a})$,* FindNewIndex$(f, n, I, \theta, \boldsymbol{a})$ *computes ω_* and $\overline{a}_*$ corresponding to the most violated constraint:*

$$\omega_* = \arg\max_{\omega\in\Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T\boldsymbol{x}\right),$$
$$\overline{a}_* - \overline{a} = \max_{\omega\in\Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T\boldsymbol{x}\right).$$

Proof. We have

$$\begin{aligned}
&\max_{\omega\in\Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T\boldsymbol{x}\right) \\
&= \max_{\omega=(t,\boldsymbol{\sigma})\in\Omega} \left(\sigma_0 \left(f(t) - \boldsymbol{a}^T\boldsymbol{\pi_0}(t)\right) - \sum_{i=1}^{k} \sigma_i \boldsymbol{a}^T\boldsymbol{\pi_i}(t)\right) - \overline{a} \\
&= \max_{t_l\leqslant t\leqslant t_r} \left(|\boldsymbol{a}^T\boldsymbol{\pi_0}(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T\boldsymbol{\pi_i}(t)|\right) - \overline{a}.
\end{aligned}$$

Therefore, by computing t_* (line 1) and $\boldsymbol{\sigma_*}$ (lines 3-4), Algorithm FindNewIndex ensures that $\omega_* := (t_*, \boldsymbol{\sigma_*})$ is the index of the most violated constraint, with

$$c(\omega_*) - \boldsymbol{\alpha}(\omega_*)^T\boldsymbol{x} = \overline{a}_* - \overline{a} \geqslant 0.$$

□

Lemma 4 (Correctness of EXCHANGE)**.** *If* $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ *is a basis of* $\mathbb{R}^{n+2}$ *and* $\boldsymbol{y}$ *the optimal solution of Problem* (D_m) *for* $\boldsymbol{\omega}$*, then* EXCHANGE$(n, \theta, \boldsymbol{\omega}, \boldsymbol{y}, \omega_*)$ *computes new* $\boldsymbol{\omega'} \in \Omega^{n+2}$ *and* $\boldsymbol{y'} \in \mathbb{R}^{n+2}$ *such that:*

- $\boldsymbol{\omega'} = \{\omega_j\}_{j\in J} \cup \{\omega_*\}$ *for a subset* $J \subset [1\,..\,n+2]$ *of size* $n+1$*;*
- $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ *is a basis of* $\mathbb{R}^{n+2}$*;*
- $\boldsymbol{y'}$ *is the optimal solution of Problem* (D_m) *for* $\boldsymbol{\omega'}$*.*

Proof. In order to increase the objective value in the dual Problem (D), a measure supported on $\boldsymbol{\omega} \cup \{\omega_*\}$ is looked for, requiring nonnegative coefficients $\tilde{y}_j$ for $j \in [1\,..\,n+2] \cup \{*\}$ s.t.

$$\boldsymbol{z} = \sum_{j=1}^{n+2} \tilde{y}_j \boldsymbol{\alpha}(\omega_j) + \tilde{y}_* \boldsymbol{\alpha}(\omega_*) = \sum_{j=1}^{n+2} \left(\tilde{y}_j + \gamma_j \tilde{y}_*\right) \boldsymbol{\alpha}(\omega_j),$$

while maximizing $\tilde{y}_*$. But $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ being a basis of $\mathbb{R}^{n+2}$ implies

$$\tilde{y}_j + \gamma_j \tilde{y}_* = y_j, \qquad j \in [1\,..\,n+2].$$

The nonnegativity constraints on the $\tilde{y}_j$ induces the choice of the exiting index ω_{j_0} (line 2) and the values of the $\tilde{y}_j$ (lines 3-4). Note that the first line of the linear system in line 1 says that the coefficients γ_j sum to 1. Hence, at least one of them is strictly positive, so that j_0 exists (line 2), though it is not necessarily unique.

Finally, $\{\boldsymbol{\alpha}(\omega_j)\}_{j\in[1..n+2]-\{j_0\}\cup\{*\}}$ remains a basis of $\mathbb{R}^{n+2}$ since $\gamma_{j_0} \neq 0$, that is $\boldsymbol{\alpha}(\omega_*)$ is not in the linear subspace spanned by $\{\boldsymbol{\alpha}(\omega_j)\}_{j\in[1..n+2]-\{j_0\}}$. □

Lemma 5. *The total error* $\overline{a}^{(\ell)}$ *computed over the discrete set* $\boldsymbol{\omega}^{(\ell)}$ *increases at each iteration.*

Proof. Let $\tilde{y}_*^{(\ell)}$ and $\gamma_j^{(\ell)}$ denote the variables $\tilde{y}_*$ and γ_j for $j \in [1\,..\,n+2]$ in EXCHANGE$(n, \theta, \boldsymbol{\omega}^{(\ell)}, \boldsymbol{y}^{(\ell)}, \omega_*^{(\ell)})$. By strong duality in linear programming, $\overline{a}^{(\ell)}$ is also the objective value of the optimal solution $\boldsymbol{y}^{(\ell)}$ in the discretized dual problem (D_{n+2}) for $\boldsymbol{\omega}^{(\ell)}$. Hence, by writing

$$\begin{aligned}
\overline{a}^{(\ell)} &= \sum_{j=1}^{n+2} y_j^{(\ell)} c(\omega_j^{(\ell)}), \quad \text{and} \\
\overline{a}^{(\ell+1)} &= \sum_{j=1}^{n+2} y_j^{(\ell+1)} c(\omega_j^{(\ell+1)}) \\
&= \sum_{j=1}^{n=2} \left(y_j^{(\ell)} - \gamma_j^{(\ell)} \tilde{y}_*^{(\ell)}\right) c(\omega_j^{(\ell)}) + \tilde{y}_*^{(\ell)} c(\omega_*^{(\ell)}),
\end{aligned}$$

we have

$$\begin{aligned}\overline{a}^{(\ell+1)} - \overline{a}^{(\ell)} &= \tilde{y}_*^{(\ell)} \left(c(\omega_*^{(\ell)}) - \sum_{j=1}^{n+2} \gamma_j^{(\ell)} c(\omega_j^{(\ell)}) \right) \\ &= \tilde{y}_*^{(\ell)} \left(c(\omega_*^{(\ell)}) - \sum_{j=1}^{n+2} \gamma_j^{(\ell)} \boldsymbol{\alpha}(\omega_j^{(\ell)})^T \boldsymbol{x}^{(\ell)} \right) \\ &= \tilde{y}_*^{(\ell)} (c(\omega_*^{(\ell)}) - \boldsymbol{\alpha}(\omega_*^{(\ell)})^T \boldsymbol{x}^{(\ell)}) \geqslant 0,\end{aligned}$$

because $\tilde{y}_*^{(\ell)} \geqslant 0$ and the constraint $\boldsymbol{\alpha}(\omega_*^{(\ell)})^T \boldsymbol{x}^{(\ell)} \geqslant c(\omega_*^{(\ell)})$ is violated at the beginning of iteration ℓ. □

.2 Optimizing with the relative error

When considering the relative error in place of the absolute error, Problem (P_{general}) is replaced by

$$\min_{\substack{a_i \in \mathbb{R}, \\ i \in [0..n]}} \max_{t \in I} \left(\frac{|f(t) - p(t)| + |\tilde{p}(t) - p(t)|}{|f|} \right) \qquad (P^{\text{rel}}_{\text{general}})$$

where f is assumed to be strictly positive over I. This is equivalent to multiplying the error variable $\overline{a}$ by f in Problem (P), yielding the following new linear SIP formulation

$$\begin{array}{ll} \min\limits_{\boldsymbol{x} \in \mathbb{R}^d} & \boldsymbol{z}^T \boldsymbol{x} \\ \text{s.t.} & \tilde{\boldsymbol{\alpha}}(\omega)^T \boldsymbol{x} \geqslant c(\omega), \quad \omega \in \Omega, \end{array} \qquad (P^{\text{rel}})$$

with

$$\begin{aligned} \boldsymbol{x} = (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2}, \ \boldsymbol{z} = (1, 0, \ldots, 0) \in \mathbb{R}^{n+2}, \\ \tilde{\boldsymbol{\alpha}}(t, \sigma_0, \ldots, \sigma_k) = (f(t), \sigma_0 \boldsymbol{\pi_0}^T(t) + \sum_{i=1}^{k} \sigma_i \boldsymbol{\pi_i}^T(t))^T \in \mathbb{R}^{n+2}, \qquad (16) \\ \mathfrak{S} = \{-1, 0, 1\}^{k+1}, \ \omega = (t, \sigma_0, \ldots, \sigma_k) \in \Omega := I \times \mathfrak{S}. \end{aligned}$$

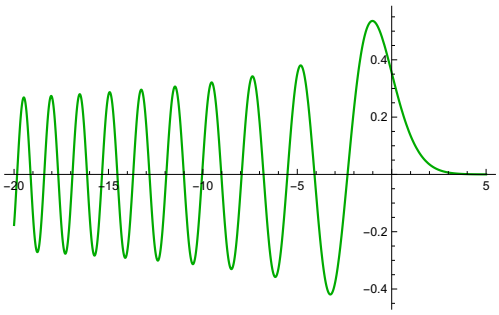
Therefore, replacing $\boldsymbol{\alpha}(\omega)$ by $\tilde{\boldsymbol{\alpha}}(\omega)$ in Eval&ApproxOptimize and its subroutines provides an algorithm to compute the best degree-n polynomial w.r.t. both approximation and evaluation errors in relative setting.

.3 Airy function (continued)

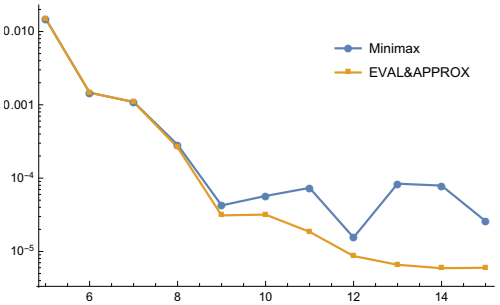
The Airy function Ai is a special function frequently used in theoretical physics. In particular, some applications require to compute $\text{Ai}(t)$ for possibly large negative values of t, where the function exhibits a highly oscillatory

behavior (see Figure 6a). However, contrary to elementary functions, there exists no simple argument reduction for Ai. Therefore, one polynomial approximation is needed for each interval of the domain subdivision, and these intervals cannot be assumed to be small. Hence, controlling the evaluation error is essential.

We here consider the approximation of the Airy function over $I = [-4, 0]$ with polynomials evaluated using the Horner scheme with binary32 floating-point numbers, that is, single precision ($u = 2^{-24}$). Figure 6b shows the total error (approximation and evaluation) of the degree-n minimax polynomial (in blue), in function of n. Specifically, the total error starts to decrease when n increases, thanks to the improvement of the approximation error. However, at some point (here $n = 9$), the total error starts again to increase, due to the evaluation error of higher degree polynomials. On the contrary, the evaluation error of polynomials obtained with EVAL&APPROXOPTIMIZE (in yellow) continues decreasing to some asymptotic threshold (around $5 \cdot 10^{-6}$). In such a case, reducing the evaluation error by using a higher degree can be more efficient than increasing the floating-point precision.



(a) Airy function Ai



(b) Total error for Minimax and EVAL&APPROXOPTIMIZE polynomials, in function of degree

Figure 6: Approximation of Ai over $[-4, 0]$