An Exchange Algorithm for Optimizing both Approximation and Finite-Precision Evaluation Errors in Polynomial Approximations

Denis Arzelier, Florent Bréhard, Tom Hubrecht, Mioara Joldes

An Exchange Algorithm for Optimizing both Approximation and Finite-Precision Evaluation Errors in Polynomial Approximations

DENIS ARZELIER, LAAS-CNRS, Université de Toulouse, CNRS, France
FLORENT BRÉHARD, Univ. de Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, France
TOM HUBRECHT, CNRS, LIP, INRIA AriC, France
MIOARA JOLDES, LAAS-CNRS, Université de Toulouse, CNRS, France

The finite precision implementation of mathematical functions frequently depends on polynomial approximations. A key characteristic of this approach is that rounding errors occur both when representing the coefficients of the polynomial on a finite number of bits, and when evaluating it in finite precision arithmetic. Hence, to find a best polynomial, for a given fixed degree, norm and interval, it is necessary to account for both the approximation error and the floating-point evaluation error. 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 a 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 exchange algorithm. An open-source C implementation using the Sollya library is presented and tested on several examples, which are then analyzed and compared against state-of-the-art Sollya routines.

CCS Concepts: • **Mathematics of computing → Mathematical software**; **Linear programming**; **Approximation**; • **Theory of computation → Rounding techniques**; *Numeric approximation algorithms.*

Additional Key Words and Phrases: polynomial approximation, finite-precision, evaluation error, semi-infinite linear programming, Remez exchange algorithm, mathematical library

1 INTRODUCTION

Polynomials are often used for approximating functions on computers [2, 39, 40]. 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 754-2019 [1] 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 for such functions and are collected in mathematical libraries named *libm*s (see for instance [24, 37] or [12] for further details on the issues of correct rounding and existing libms). In the last two decades, sophisticated tools were developed, which allow nowadays for almost automatically generated and tuned libms [10, 14, 33, 34].

In general, 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 [41]. Otherwise, code generating techniques extended to larger classes of special functions, which are widely used in scientific and technical applications (like Bessel, Airy, Erf, etc.), are mostly based on piecewise polynomial approximations [35].

Then, the implementation task becomes: given a description of a function f, an input interval I, and a target accuracy $\varepsilon > 0$, it 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:

Authors' addresses: Denis Arzelier, LAAS-CNRS, Université de Toulouse, CNRS, 07 Avenue Colonel Roche, BP 54200, Toulouse, France, 31031, arzelier@laas.fr; Florent Bréhard, Univ. de Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, Bâtiment Esprit, Lille, France, F59000, florent.brehard@univ-lille.fr; Tom Hubrecht, CNRS, LIP, INRIA AriC, 46, Allée d'Italie, Lyon, France, F69364, tom.hubrecht@ens-lyon.fr; Mioara Joldes, LAAS-CNRS, Université de Toulouse, CNRS, 07 Avenue Colonel Roche, BP 54200, Toulouse, France, 31031, mmjoldes@laas.fr.

a) 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 $\|(f - p)/f\|_I \leqslant \varepsilon_{\text{approx}}$.

When the coefficients of the polynomial are constrained to fit on a specific (sometimes small) number of bits, efficient algorithms based on parametric integer linear programming were developed in [13], whose goal is to enumerate the set of integer points which lie inside a convex polyhedron (this polyhedron can depend linearly on one or more integer parameters). While this technique worked for precisions up to 32 bits, another idea based on Euclidean lattice reduction [11] proved to be very efficient for higher precision (binary64, double-double, or higher) and was successfully implemented in Sollya [22], a state-of-the-art tool for obtaining such approximations.

In the simpler case of polynomials p with real coefficients and given degree n, $p = \sum_{i=0}^{n} a_i t^i$, this problem is known as the *minimax* approximation:

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

which can be solved by the Remez exchange algorithm [46] (see also [11, 21, 53] and references therein). The convergence of this iterative algorithm is quadratic under certain conditions [54] and it has 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.

b) 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 $\varepsilon_{\text{eval}}$.

Some heuristics like those presented in [15, 34] 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 [22], Gappa[1] [26], Rosa[2] [25] or Real2Float[3] [47].

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 [52].

In this sense, the approach developed in [36, 37] for the purpose of obtaining correctly rounded elementary functions exploits linear programming (LP) in a different manner for the approximation step. Specifically, one considers constraints on the coefficients of the form $l \leqslant p(t) \leqslant h$, where for an input t, all the values in the corresponding interval $[l, h]$ round to the same value as $f(t)$ (in the specific given precision and rounding mode). By discretizing the input, an exact rational LP problem is solved. The evaluation step is done by a search-and-refine technique. Initially, the candidate polynomial satisfies the set of constraints when evaluated in real numbers. If an posteriori finite-precision evaluation fails to do so, the constraints are restricted iteratively. This strategy is

[1] https://gappa.gitlabpages.inria.fr/
[2] https://github.com/malyzajko/rosa/
[3] https://github.com/afd/real2float/

reported to successfully work up to binary32 implementation, but it is unclear whether this scales beyond this precision.

However, as explicitly mentioned in [13], *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 main goal of this work is to make progress on the following open problem: provide and algorithm which computes 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 for a given evaluation scheme, over a given input interval I, with respect to a given function f. We consider a *black-box* description of f i.e., one disposes of values $f(t)$, up to any required accuracy [34]. This allows for handling very general functions (elementary, special, etc.), but also implies that no argument reduction step is usually possible. In the absolute error case the corresponding optimization problem is stated as follows:

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

This article provides an extended version of our prior work [4].

In Section 2, a model for the evaluation error $|\tilde{p}(t) - p(t)|$ is proposed. Based on [42], the accuracy of a given arbitrary polynomial evaluation scheme is recursively assessed by bounding the rounding error of each elementary operation. The proposed implementation can handle different precisions and both linearized and higher order error terms. This leads to the formulation in Section 3 of Problem P_{general} as a linear semi-infinite programming (LSIP) problem [45, 50]. We provide a concise, yet standalone description of the duality theory employed, in particular, the theoretical formulation of a dual linear programming problem defined in the space of positive measures. We revisit theoretical results which show that this problem is reducible to a discretization of finite size, which allows for adapting a general*exchange algorithm* for LSIP [16–18, 55] to our specific case. The correctness of this tailored algorithm is proved in Section 4. Then, we turn to more practical aspects and discuss in Section 5 the case of the relative error, which is similar from a theoretical standpoint, but requires a careful implementation when the function f cancels for certain values in the considered interval. In Section 6, we present our new implementation in C, using in particular the Sollya library, which can handle many practical cases and both high and low precision constraints on the coefficients. Finally, in Section 7, we conclude this first attempt at optimizing with respect to the sum of both errors: practical examples show that in some cases the evaluation error can be improved, while in some other cases, the minimax polynomial solution of problem P_{minimax} is very close to the solution of P_{general}.

2 EVALUATION ERROR

Consider first some basic notation used for error analysis [41]. 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 $\mathrm{RN}(t)$ as t rounded to nearest. This is the default rounding mode in IEEE-754 arithmetic [32], 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 $c = \mathrm{RN}(a \top b)$, for any arithmetic operation $\top \in \{+, -, \times, \div\}$. We have

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

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

Moreover, there exists a real number ϵ such that

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

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 [42], which is detailed below as an example. More recently, several works insisted on the automatic algorithmic approach via operator overloading similar to automatic differentiation [9, 51]. Based on this, we propose, for completeness[4], 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 how the linearized

Algorithm 1 HORNER(p,t).

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

evaluation error is computed for the polynomial $p(t) = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_0$ 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:

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

Ignoring the higher order terms, the linear approximation θ_{lin} of the evaluation error, function of ϵ_i^{+} and $\epsilon_i^{\times}$ is:

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

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.

[4]While the general ideas are the same as in [9, 51] and references therein, we could not find the exact pseudo-code in literature, so it is stated in order to provide a complete algorithmic solution for problem P_{general}.

In general, an automated error analysis is provided, for which one can consider higher order terms. This case is of potential interest for evaluation schemes, where operations have different precisions. As exemplified in Table 1, to automate the evaluation error analysis, we firstly associate to a mathematical expression e^*, a given symbolic evaluation scheme e, composed of terms $\mathrm{RN}(e', u)$. This means that e' is rounded with a relative error $|\epsilon_{e'}^{[u]}| \leqslant u$. This formulation models rounding errors occuring for both input variables ($e' \in \mathcal{V}$, where $\mathcal{V}$ denotes the set of input variables) and for 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 $\tilde{e}'(1 + \epsilon_{e'}^{[u]})$. This formulation allows for modeling equal relative errors corresponding to multiple occurences of the same subexpression $\mathrm{RN}(e', u)$ with one common variable $\epsilon_{e'}^{[u]}$. Furthermore, one can also consider different precisions for roundings, which translate to different bounds u_i. For instance, in Table 1, one considers both binary64 and binary32 operations by instantiating $u_1 = 2^{-53}$ and $u_2 = 2^{-24}$.

▹ *Example A. Arithmetic operations in binary64, with* $u = 2^{-53}$.	
▹ *Mathematical expression:*	$e_1^* = a + bc$
▹ *Evaluation scheme:*	$e_1 = \mathrm{RN}(a + \mathrm{RN}(b \times c, u), u)$
▹ *Roundings expression:*	$\tilde{e}_1 = (a + bc(1 + \epsilon_{b\times c}^{[u]}))(1 + \epsilon_{a+\mathrm{RN}(b\times c,u)}^{[u]})$
▹ *Linearized error:*	$\theta_{\mathrm{lin}}^{(e_1)} = bc\epsilon_{b\times c}^{[u]} + (a + bc)\epsilon_{a+\mathrm{RN}(b\times c,u)}^{[u]}$
▹ *Linearized error bound:*	$\lvert\theta_{\mathrm{lin}}^{(e_1)}\rvert \leqslant (\lvert bc\rvert + \lvert a + bc\rvert)u$
▹ *Example B. Fused multiply-add (FMA) in double precision* ($u_1 = 2^{-53}$)*, with input a rounded to binary32* ($u_2 = 2^{-24}$)*.*	
▹ *Mathematical expression:*	$e_2^* = a + ba$
▹ *Evaluation scheme:*	$e_2 = \mathrm{RN}(\mathrm{RN}(a, u_2) + b \times \mathrm{RN}(a, u_2), u_1)$
▹ *Roundings expression:*	$\tilde{e}_2 = (a(1 + \epsilon_a^{[u_2]}) + ba(1 + \epsilon_a^{[u_2]}))(1 + \epsilon_{\mathrm{RN}(a,u_2)+b\times\mathrm{RN}(a,u_2)}^{[u]})$
▹ *Eval. error:*	$\theta^{(e_2)} = (a + ba)\epsilon_a^{[u_2]} + (a + ba)\epsilon_{\mathrm{RN}(a,u_2)+b\times\mathrm{RN}(a,u_2)}^{[u_1]} + (a + ba)\epsilon_a^{[u_2]}\epsilon_{\mathrm{RN}(a,u_2)+b\times\mathrm{RN}(a,u_2)}^{[u_1]}$
▹ *Error bound:*	$\lvert\theta^{(e_2)}\rvert \leqslant \lvert a + ba\rvert(u_1 + u_2 + u_1u_2)$
▹ *Example C. Horner scheme with both single* ($u_2 = 2^{-24}$) *and double* ($u_1 = 2^{-53}$) *precision operations.*	
▹ *Mathematical expression:*	$e_3^* = a_0 + t(a_1 + ta_2)$
▹ *Evaluation scheme:*	$e_2 = \mathrm{RN}(a_0 + \mathrm{RN}(t \times \mathrm{RN}(a_1 + \mathrm{RN}(t \times a_2, u_2), u_2), u_1), u_1)$
▹ *Roundings expression:*	$\tilde{e}_3 = a_0(1 + \epsilon_+^{[u_1]}) + a_1t(1 + \epsilon_+^{[u_1]})(1 + \epsilon_\times^{[u_1]})(1 + \epsilon_+^{[u_2]}) + a_2t^2(1 + \epsilon_+^{[u_1]})(1 + \epsilon_\times^{[u_1]})(1 + \epsilon_+^{[u_2]})(1 + \epsilon_\times^{[u_2]})$
▹ *Eval. error:*	$\theta^{(e_3)} = (a_0 + a_1t + a_2t^2)\epsilon_+^{[u_1]} + (a_1t + a_2t^2)(\epsilon_\times^{[u_1]} + \epsilon_+^{[u_2]} + \epsilon_+^{[u_1]}\epsilon_+^{[u_2]} + \epsilon_\times^{[u_1]}\epsilon_+^{[u_2]}) + a_2t^2(\epsilon_\times^{[u_2]} + \epsilon_+^{[u_2]}\epsilon_\times^{[u_2]} + \epsilon_+^{[u_1]}\epsilon_\times^{[u_2]} + \epsilon_\times^{[u_1]}\epsilon_\times^{[u_2]}) + \ldots$
▹ *Error bound:*	$\lvert\theta^{(e_3)}\rvert \leqslant \lvert a_0 + a_1t + a_2t^2\rvert u_1 + \lvert a_1t + a_2t^2\rvert(u_1 + u_2 + 2u_1u_2) + \lvert a_2t^2\rvert(u_2 + u_2^2 + 2u_1u_2) + O(u_1^2)$

Table 1. Evaluation error examples.

Automatic linearized evaluation error expressions θ_{lin}, such as in (6) and Example A in Table 1, 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:*

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

However, in some cases, when several different precisions are used, higher order error terms may need to be considered. A basic example (Example C) is shown in Table 1. The proposed implementation (similar to the linear case algorithm) is discussed in Section 6.2.

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

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

$$|\theta_{\text{lin}}(\boldsymbol{a}, t)| \leqslant \sum_{i=1}^{k} |\boldsymbol{\pi}_i(t)^T\boldsymbol{a}| := \theta(\boldsymbol{a}, t). \tag{8}$$

Example 1. *For Horner evaluation, equation (7) gives:*

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

3 SEMI-INFINITE PROGRAMMING FORMULATION

3.1 Problem (P_{general}) as a linear continuous 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 [8, Section 4.3.1] for instance), with the compact index set $I = [t_l, t_r]$ and the monomial basis $\boldsymbol{\pi_0}(t) = (1, \dots, t^n)^T$.

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

Problem (P'_{general}) is a convex Semi-Infinite Programming (SIP) problem (see [45] 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. The evaluation error is as in equation (8), and define:

$$\boldsymbol{x} = (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2},\ \boldsymbol{z} = (1, 0, \dots, 0) \in \mathbb{R}^{n+2},$$

$$\boldsymbol{\alpha}(t, \sigma_0, \dots, \sigma_k) = (1, \sigma_0 \boldsymbol{\pi_0}^T(t) + \sum_{i=1}^{k} \sigma_i \boldsymbol{\pi_i}^T(t))^T \in \mathbb{R}^{n+2},$$

$$\mathfrak{S} = \{-1, 0, 1\}^{k+1},\ \omega = (t, \sigma_0, \dots, \sigma_k) \in \Omega := I \times \mathfrak{S}.$$

Then, Problem (P'_{general}) is exactly 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 $\mathbb{R}$. As a result of the definition of the set Ω and of the continuity properties of the functions $\boldsymbol{\alpha}(\cdot)$ and $c(\cdot)$ on Ω, the LSIP (P'_{general}) is called *continuous* [30, Section 5.2]. 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:*

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

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

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

In Section 4 an exchange algorithm which solves Problem (P'_{general}) is presented. It can be seen as a generalization, in the above framework, of the Remez exchange algorithm, which solves Problem (P_{minimax}). To prove its correctness, important duality and discretization properties of continuous linear SIP problems are recalled, closely following the survey [50] and the book [30].

About the Haar condition. In the classical setting of Problem (P_{minimax}), the convergence of the Remez exchange algorithm as well as the uniqueness of the optimal solution is guaranteed by the so-called *Haar condition* satisfied by the system of $(n+1)$ linearly independent functions $\boldsymbol{\varphi}(t) = (\varphi_0(t), \dots, \varphi_n(t))^T$ used for the approximation over $I = [t_\ell, t_r]$. This condition states that for any set of $n+1$ distinct points $t_\ell \leqslant t_1 < t_2 < \cdots < t_{n+1} \leqslant t_r$, the $(n+1) \times (n+1)$ determinant $\det\left(\varphi_i(t_j)\right)_{0\leqslant i\leqslant n}^{1\leqslant j\leqslant n+1}$ is nonzero. Equivalently, any nontrivial combination $\boldsymbol{\varphi}(t)^T \boldsymbol{a} = a_0\varphi_0(t) + \cdots + a_n\varphi_n(t)$ (i.e., at least one a_i is nonzero) has at most n distinct zeros over I (see, e.g., [44, §7.3] or [19, §3.8] for more details). This condition is notably satisfied by the monomial basis $\boldsymbol{\varphi}(t) = \boldsymbol{\pi_0}(t) = (1, t, \dots, t^n)^T$ considered in this article.

For Problem (P_{general}), however, the index set Ω is no longer a real segment but a compact set isomorphic to 3^{k+1} copies of I. The Haar condition is not satisfied over it in general for the basis considered here, $\boldsymbol{\varphi}(\omega) = \boldsymbol{\varphi}(t, \sigma_0, \dots, \sigma_k) = \sum_{i=0}^k \sigma_i \boldsymbol{\pi_i}(t)$. To illustrate this issue, consider the case $n = 1$ with the Horner evaluation scheme. Using the evaluation error formulas of Example 1, we have $\boldsymbol{\varphi}(\omega) = (\sigma_0 + u\sigma_1, t(\sigma_0 + u\sigma_1 + u\sigma_2))$. It is then possible to choose two distinct points $\omega, \omega' \in \Omega$ so that the associated 2×2 determinant vanishes. For example, take $t, t' \in I$ with $t' = t(1+2u)$, and define $\omega = (t, 1, 1, 1)$, $\omega' = (t', 1, 1, -1)$. Then $\boldsymbol{\varphi}(\omega) = \boldsymbol{\varphi}(\omega') = (1+u, t(1+2u))$, hence the determinant is zero.

For this reason, the iterative exchange algorithm presented in Section 4 relies on an additional hypothesis (Assumption 1) to prove its convergence.

3.2 Duality and Discretization for Continuous LSIP Problems

For a continuous LSIP Problem (P), we denote respectively by $\mathrm{val}(P)$ and $\mathrm{Sol}(P)$, its optimal value and the set of its optimal solutions. The concept of duality has a central role in the design of exchange algorithms for SIP problems and will stand at the core of the specific exchange algorithm presented in Section 4. Associated with (P'_{general}), different dual problems can be specified depending of the choice of the primal constraint and variable spaces [3]. The problem (P'_{general}) is a continuous LSIP problem and if we define the continuous mapping $G : \boldsymbol{x} \mapsto g(\boldsymbol{x}, \cdot)$ from $\mathbb{R}^d$ to the Banach space of continuous functions $C(\Omega)$, equipped with the uniform norm $\|h\|_\Omega = \sup_{\omega\in\Omega} |h(\omega)|$, it is natural to embed the constraints of (P'_{general}) into the functional Banach space $C(\Omega)$. Its topological dual is the space $C(\Omega)^*$ of signed Borel measures μ over $(\Omega, \mathcal{B})$ where $\mathcal{B}$ is the Borel sigma algebra of Ω (see the Riesz representation theorem in [38, Section 5.5] and [49, Section 21.5]). The dual norm of $C(\Omega)^*$ is $|\mu|(\Omega)$, where $|\mu|$ is the total variation of μ and the bilinear form pairing $C(\Omega)$ and $C(\Omega)^*$ is defined by the duality bracket:

$$\langle h, \mu \rangle = \int_\Omega h(\omega) d\mu(\omega). \tag{9}$$

For a measure $\mu \in C(\Omega)^*$, its support is the smallest closed subset Γ of Ω such that $|\mu|(\Omega \setminus \Gamma) = 0$. A positive measure μ is denoted by $\mu \geq 0$. A classical example of a positive measure with discrete support is the Dirac measure of support $\{\omega_j\}$:

$$\delta_{\omega_j}(A) = \begin{cases} 0 & \text{if} \quad \omega_j \notin A, \\ 1 & \text{if} \quad \omega_j \in A, \end{cases} \tag{10}$$

for any set $A \subseteq \Omega$..

The dual problem of (P'_{general}) is now derived by using the usual Lagrangian approach. The Lagrangian of (P'_{general}) is defined as:

$$\mathcal{L}(x, \mu) := z^T x + \langle G(x), \mu \rangle = z^T x + \int_\Omega [c(\omega) - \alpha(\omega)^T x] \mathrm{d}\mu(\omega), \quad x \in \mathbb{R}^d, \mu \geq 0. \tag{11}$$

It is straightforward that:

$$\sup_{\mu\geq 0}\mathcal{L}(x,\mu)=\begin{cases}z^Tx & \text{if } \alpha(\omega)^Tx\geqslant c(\omega) \text{ for all } \omega\in\Omega,\\ +\infty & \text{otherwise.}\end{cases}$$

Hence, (P) is equivalent to $\inf_{x\in\mathbb{R}^d}\sup_{\mu\geq 0}\mathcal{L}(x,\mu)$. Noticing that:

$$\inf_{x\in\mathbb{R}^d}\mathcal{L}(x,\mu)=\begin{cases}\int_\Omega c(\omega)\mathrm{d}\mu(\omega) & \text{if } \int_\Omega \alpha(\omega)\mathrm{d}\mu(\omega)=z,\\ -\infty & \text{otherwise}\end{cases}$$

the *Lagrangian dual* problem (D) related to the *primal* problem (P) is obtained as the relaxation $\sup_{\mu\geq 0}\inf_{x\in\mathbb{R}^d}\mathcal{L}(x,\mu)$, which reads:

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

Problem (D) is an LP problem defined in the space of positive measures which is hard to solve in general. If the *weak duality*, that is $\mathrm{val}(D)\leqslant\mathrm{val}(P)$ always holds, the central issue pertaining to duality theory is to find conditions on the primal and/or dual to guarantee that $\mathrm{val}(D)=\mathrm{val}(P)$.

Definition 1. *[50] It is said that the 'no duality gap' property holds if* $\mathrm{val}(D)=\mathrm{val}(P)$*, and the 'strong duality' property holds if* $\mathrm{val}(D)=\mathrm{val}(P)$ *and the dual problem has an optimal solution.*

These two properties of interest are strongly connected to the reducibility or discretizability properties of both problems (P) and (D). A discretization (P_m) of (P) for a set $\boldsymbol{\omega}=\{\omega_1,\ldots,\omega_m\}\subseteq\Omega$ is the following linear program:

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

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 2. *[50]* (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* $\mathrm{val}(P_m)=\mathrm{val}(P)$.

Concerning the dual problem (D), a discretized counterpart (D_m) of (D) is obtained, by restricting the support of $\mu\succeq 0$ to $\{\omega_1,\ldots,\omega_m\}$, that is considering positive discrete measures of the form $\mu=\sum_{j=1}^m y_j\delta_{\omega_j}$ with $y_j\geqslant 0$:

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

with $\mathrm{val}(D_m)\leqslant\mathrm{val}(D)$ since the feasible set of (D_m) is included in the feasible set of (D). It is important to note that the LP dual of the discretized problem (P_m) is exactly (D_m) which implies that $\mathrm{val}(D_m)=\mathrm{val}(P_m)$ 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 [50, Theorems 2.3, 3.1 and 3.2] and [7, Thm 5.99].

Theorem 1. *[50, Thm. 2.3, 3.1, 3.2], [7, Thm 5.99] 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 *(primal Slater's condition): there exists* $\boldsymbol{x}^\circ$ *such that:*

$$\boldsymbol{\alpha}(\omega)^T \boldsymbol{x}^\circ > c(\omega), \quad \text{for all } \omega \in \Omega; \tag{12}$$

A4 *(dual Slater's condition): there exist* $\omega_1, \dots, \omega_d \in \Omega$ *with* $(\boldsymbol{\alpha}(\omega_1), \cdots, \boldsymbol{\alpha}(\omega_d))$ *linearly independent such that:*

$$\exists\, y_1, \dots, y_d > 0, \quad \boldsymbol{z} = \sum_{j=1}^{d} y_j \boldsymbol{\alpha}(\omega_j), \tag{13}$$

the following statements are true:

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

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

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

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

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 $\mathrm{val}(P) = +\infty$ 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 $\mathrm{val}(P) < +\infty$. In addition, $\mathrm{val}(P) > -\infty$ 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 explicit instance for $\{\omega_1, \dots, \omega_d\}$ is provided by Algorithm InitPoints in Section 4 (see Lemma 1 for its correctness). □

Remark 3.1. *The dual Slater's condition A4 is equivalent to the usual cone condition given in [50, Eq. (2.20)], which is the regularity condition involving the moment cone of the LSIP Problem* (P) *[29]. By the fundamental result of Rogosinski [48], this cone is equivalently represented in the space of positive Borel measures. The condition A4 is therefore equivalent to the regularity condition given in [7, Eq. (5.259)].*

The fact that both (P) and (D) are reducible to a discretization of size of at most d, allows for recasting the problem (P'_{general}) as the problem of finding the right discretization $\{\omega_1, \dots, \omega_m\}$, $(m \leqslant d)$, 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 [16] to our specific case.

This algorithm can be seen as a particular instance of the dual-simplex primal-exchange algorithm [30]. The main idea is to keep a set of $m = d = n + 2$ elements $\boldsymbol{\omega}^{(\ell)} = \{\omega_j{}^{(\ell)}\}_{j=1}^m$, which is updated at each iteration ℓ by a one-element exchange rule. This exchange rule requires finding the solution $\boldsymbol{y}^{(\ell)}$ of the discretized dual problem $(D_m^{(\ell)})$, with $\boldsymbol{\omega}^{(\ell)}$. 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 $(P_m^{(\ell)})$ and (P) for the instance $\boldsymbol{x}^{(\ell)} := (\overline{a}^{(\ell)}, \boldsymbol{a}^{(\ell)})$, obtained by solving the primal $(P_m^{(\ell)})$, is equal to the objective value of (D) for the instance $\boldsymbol{y}^{(\ell)}$[5]. 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 the violated constraints (the worst one in terms of feasibility) 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.

4 ITERATIVE EXCHANGE ALGORITHM

Algorithm Eval&ApproxOptimize computes a degree-n best polynomial approximation with respect to both evaluation and approximation errors, i.e. it solves (P'_{general}) based on developments from Section 3.2. Regarding the main steps of the new algorithm, an analogy with Remez is as follows:
- InitPoints 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.

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.
- InitPoints and Exchange need the solution of the dual problem.

A running example for this algorithm is given in Section 6. 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 exchange 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 $(D_m^{(\ell)})$ is an interior point, that is $y_j^{(\ell)} > 0$ for all $j \in [1\,..\,m]$.*

Remark 4.1. *Assumption 1 guarantees that at each iteration, the exchange rule of Exchange is well-defined and that the linear systems spanned by $\boldsymbol{\alpha}(\omega_j)$ and solved in the routines InitPoints, SolvePrimal and Exchange are always invertible. It is also crucial to prove the convergence of the overall Algorithm Eval&ApproxOptimize (Theorem 2 below). Although this hypothesis may seem an ad-hoc assumption, it is justified by the following intuition. As noted at the end of Section 3.1, the system of $n+1$ functions $\boldsymbol{\varphi}(\omega) = (\varphi_0(\omega), \ldots, \varphi_n(\omega)) = \sum_{i=0}^{k} \sigma_i \boldsymbol{\pi_i}(t)$ does not satisfy, in general the Haar condition on I, which would be the key to guarantee that this assumption always holds. However, since $u = 2^{-p} \ll 1$, this system may be viewed as a slight perturbation of $\boldsymbol{\pi_0}(t) = (1, t, \ldots, t^n)$, which satisfies the Haar condition on I. Hence, we expect that the $(n+1)$-tuples of points that violate the Haar condition by cancelling the corresponding determinant, contain at least two very close points, within a distance proportional to u (this was notably the case for the counter-example given previously). On the other hand, we expect that at each iteration, similarly to the classical Remez exchange algorithm (see, e.g., [44, Thm. 9.2]), the points t_j associated to ω_j are spaced far enough apart, so that the determinant of the Haar condition remains nonzero. However, note that Assumption 1 never failed in our practical experiments.*

In addition, one needs preliminary correctness proofs of InitPoints, SolvePrimal, FindNewIndex, and Exchange.

Theorem 2. *Let f be a continuous function over an interval $I = [t_\ell, t_r]$, 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*

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

and returns a degree-n polynomial approximation for f with a total error ε (approximation and evaluation) satisfying:

$$\varepsilon^* \leqslant \varepsilon \leqslant (1+\tau)\varepsilon^*, \tag{14}$$

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

Proof. • It is proven 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) $\omega_*^{(\ell)} = \arg\max_{\omega\in\Omega} \left(c(\omega) - \boldsymbol{\alpha}(\omega)^T \boldsymbol{x}^{(\ell)}\right)$.

For $\ell = 0$, InitPoints(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 $\boldsymbol{a}^{(\ell)T}\boldsymbol{\pi_0}(t)$ and therefore, it is greater or equal to the optimal error ε^*. In addition, [5, Lemma 5, Appendix] proves $\overline{a}^{(\ell)} \leqslant \overline{a}^{(\ell+1)}$.
• Finally, the convergence of this iterative process is proved by [16, 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 (14). □

Algorithm 3 Eval&ApproxOptimize(f, n, I, θ, τ)

Input: function f, $n \geqslant 0$, I, $\theta(\boldsymbol{a}, t)$ as in (8), $\tau > 0$.
Output: $(\overline{a}, \boldsymbol{a})$ solution of Problem (P) within accuracy τ.

▷ *Initialization*
$(\boldsymbol{\omega}^{(0)}, \boldsymbol{y}^{(0)}) \leftarrow$ InitPoints(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 InitPoints. Algorithm InitPoints essentially initializes the dual problem (D_{n+2}) with Chebyshev nodes for heuristic efficiency and signatures corresponding to the sign alternation of the classical Remez exchange

algorithm (without the evaluation error term). This heuristic is commonly used in the classical setting [11]. It produces an initial dual optimal solution $\boldsymbol{y}$ for this particular discretization.

Algorithm 4 InitPoints(n, I)

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

▷ *Initialize with Chebyshev nodes and Remez constraints*
for j **in** $[1\,..\,n+2]$ **do**
  $t_j \leftarrow \frac{t_\ell+t_r}{2} + \cos\left(\frac{(j-1)\pi}{n+1}\right)\frac{t_\ell-t_r}{2}$
  $\boldsymbol{\sigma}_j \leftarrow ((-1)^j, 0, \ldots, 0)$
  $\omega_j \leftarrow (t_j, \boldsymbol{\sigma}_j)$
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 $(\boldsymbol{\omega}, \boldsymbol{y})$

Lemma 1 (Correctness of InitPoints). InitPoints(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_{n+2}) *for* $\boldsymbol{\omega}$*;*
- $y_j > 0$ *for all* $j \in [1\,..\,n+2]$.

Proof. Let $\boldsymbol{A}(\boldsymbol{\omega})$ denote the $(n+2)$ square matrix whose columns are the $\boldsymbol{\alpha}(\omega_j)$. Since $\boldsymbol{\sigma_j} = \big((-1)^j, 0, \ldots, 0\big)$, 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_{n+2}) for $\boldsymbol{\omega}$, that is $\boldsymbol{A}(\boldsymbol{\omega})\boldsymbol{y} = \boldsymbol{z}$ and $\boldsymbol{y} \geqslant 0$. From Farkas' lemma [23, Section 2.4.2.], if such a vector $\boldsymbol{y}$ 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_{n+2}) 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 $\mathcal{I} \subset [1\,..\,n+2]$ of size $n+1$, and $\tilde{\boldsymbol{y}} \in \mathbb{R}^{n+1}$ s.t. $\sum_{i\in\mathcal{I}} \tilde{y}_i \boldsymbol{\alpha}(\omega_i) = \boldsymbol{z}$. The $(n+1)\times(n+1)$ subsystem extracted by removing the first line of this system of linear equations clearly shows that the family $\{\boldsymbol{\pi_0}(t_i)\}_{i\in\mathcal{I}}$ is linearly dependent (remember that $\boldsymbol{z} = [1\ 0 \cdots\ 0]^T$). But the Vandermonde determinant of this system cannot vanish since the t_i are pairwise distinct. Therefore, $\{\boldsymbol{\alpha}(\omega_i)\}_{i=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$, and $y_i > 0$ for all i.

Finally, since $\boldsymbol{A}(\boldsymbol{\omega})$ is invertible, $\boldsymbol{y}$ is the unique optimal feasible point of (D_{n+2}). □

Algorithm SolvePrimal. This algorithm computes the optimal solution $\boldsymbol{x} = (\overline{a}, \boldsymbol{a})$ of Problem (P_{n+2}) corresponding to the optimal dual solution $\boldsymbol{y}$ of (D_{n+2}). This means that the optimal primal and dual solutions are related to the same discretization $\boldsymbol{\omega}$ and that they satisfy the complementary slackness property of the associated finite-dimensional LP problems [29].

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_{n+2}) 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** $(\overline{a}, \boldsymbol{a})$

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_{n+2}) is feasible for $\boldsymbol{\omega}$, then* SolvePrimal$(f, n, \theta, \boldsymbol{\omega})$ *computes the optimal solution $\boldsymbol{x} = (\overline{a}, \boldsymbol{a})$ of Problem (P_{n+2}).*

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_{n+2}) for $\boldsymbol{\omega}$.

Let $\tilde{\boldsymbol{x}}$ be any feasible point in (P_{n+2}). Since the dual Problem (D_{n+2}) for $\boldsymbol{\omega}$ is feasible and bounded, the primal Problem (P_{n+2}) is bounded. The feasibility of (D_{n+2}) means that 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 $\boldsymbol{x}$. □

Algorithm FindNewIndex. In this algorithm, an index ω_* is chosen so that the best possible increase in the objective of the dual problem is achieved when adding it to the actual discretization $\boldsymbol{\omega}$. This index is precisely the one corresponding to the most violated constraint of the primal problem (P).

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

Input: function f, $n \geqslant 0$, $I = [t_\ell, t_r]$, θ the evaluation error as in (8), 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: $t_* \leftarrow \underset{t_\ell \leqslant t \leqslant t_r}{\arg\max} |\boldsymbol{a}^T\boldsymbol{\pi}_0(t) - f(t)| + \sum_{i=1}^{k} |\boldsymbol{a}^T\boldsymbol{\pi}_i(t)|$

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

▷ *Reconstruct signature*

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

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

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

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

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 of the primal problem (P):*

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

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

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

Although the formulation of FindNewIndex theoretically requires the values of f over the whole interval $[t_l, t_r]$, which would contradict a black-box approach, in practice FindNewIndex is implemented via a discretization of $[t_l, t_r]$, evaluating f on it, and then picking t_* among these grid points.

Algorithm Exchange. Given an index set $\boldsymbol{\omega} = \left\{\omega_j\right\}_{j=1}^{n+2}$, the optimal solution $\boldsymbol{y}$ of the associated discretized dual problem (D_{n+2}) and the index ω_* computed by algorithm FindNewIndex and corresponding to the most violated constraint of the primal problem (P), the objective of Algorithm Exchange is twofold. Firstly, the new index ω_* will replace one element ω_{j_0} of $\boldsymbol{\omega}$ so that the discretized dual problem (D'_{n+2}) associated to the resulting index set $\boldsymbol{\omega'} = \boldsymbol{\omega} \setminus \left\{\omega_{j_0}\right\} \cup \{\omega_*\}$ improves the approximation of the optimal value $\mathrm{val}(D)$ i.e. $\mathrm{val}(D'_{n+2}) \geqslant \mathrm{val}(D_{n+2})$. Secondly, the index ω_{j_0} is selected in order to guarantee the feasibility of the new solution $\boldsymbol{y'}$ of the linear system $\sum_{j=1}^{n+2} y'_j \alpha(\omega'_j) = z$, $(y'_i \geqslant 0$, for $i = 1, \cdots, n+2)$ and therefore the optimality of $\boldsymbol{y'}$ for the problem (D'_{n+2}). Keeping this last point in mind and denoting by $\gamma \in \mathbb{R}^{n+2}$ the vector of coordinates of the vector $\boldsymbol{\alpha}(\omega_*)$ in the basis $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$, a new dual vector $\tilde{\boldsymbol{y}} \in \mathbb{R}^{n+3}$ is defined as $\tilde{y}_j = y_j - \lambda \gamma_j, \quad j \in [1\,..\,n+2]$ and $\tilde{y}_{n+3} = \lambda$ for any $\lambda \in \mathbb{R}$. It is clearly feasible for the dual Problem on $n+3$ points ω_j, $j \in [1\,..\,n+2] \cup \{*\}$. The maximum possible value for λ that preserves the positivity (feasibility) conditions $y'_j = \tilde{y}_j \geqslant 0$ is

$$\lambda = \min_j \left\{ \frac{y_j}{\gamma_j} \;\middle|\; \gamma_j > 0 \right\} = \frac{y_{j_0}}{\gamma_{j_0}} = \tilde{y}^*.$$

Finally, the index ω_{j_0} is replaced by the index ω_* to build the new index set $\boldsymbol{\omega'}$ of the dual Problem (D_{n+2}).

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 $\boldsymbol{y'} \in \mathbb{R}^{n+2}$.

1: 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*
2: $j_0 \leftarrow \arg\min \left\{ \frac{y_j}{\gamma_j} \;\middle|\; \gamma_j > 0 \right\}$
▷ *Update dual solution*
3: $\tilde{y}_* \leftarrow \frac{y_{j_0}}{\gamma_{j_0}}$
4: $\tilde{y}_j \leftarrow y_j - \gamma_j \tilde{y}_*, \quad j \in [1\,..\,n+2]$
5: $\{(\boldsymbol{\omega}'_j, y'_j)\} \leftarrow \left\{(\omega_j, \tilde{y}_j), j \in [1\,..\,n+2] \setminus \{j_0\} \cup \{*\}\right\}$
6: **return** $(\boldsymbol{\omega'}, \boldsymbol{y'})$

LEMMA 4 (CORRECTNESS OF EXCHANGE). *If $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$ and $\boldsymbol{y}$ is the optimal solution of Problem (D_{n+2}) for $\boldsymbol{\omega}$, then* EXCHANGE$(n, \theta, \boldsymbol{\omega}, \boldsymbol{y}, \omega_*)$ *computes a new index set $\boldsymbol{\omega'} = \boldsymbol{\omega} \setminus \{\omega_{j_0}\} \cup \{\omega_*\}$ for some $j_0 \in [1..n+2]$, and $\boldsymbol{y'} \in \mathbb{R}^{n+2}$ such that:*

(1) *$\boldsymbol{y'}$ is the optimal solution of the discretized dual problem (D'_{n+2}) associated to $\boldsymbol{\omega'}$;*
(2) *the optimal value of (D'_{n+2}) is no less than the one of (D_{n+2}):*

$$\mathrm{val}(D'_{n+2}) = \sum_{j=1}^{n+2} y'_j c(\omega'_j) \geqslant \sum_{j=1}^{n+2} y_j c(\omega_j) = \mathrm{val}(D_{n+2});$$

(3) *$\{\boldsymbol{\alpha}(\omega'_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$.*

PROOF.

(1) First, note that since $\{\boldsymbol{\alpha}(\omega_j)\}_{j=1}^{n+2}$ is a basis of $\mathbb{R}^{n+2}$, there exist some $\gamma_j \in \mathbb{R}$ such that $\boldsymbol{\alpha}(\omega_*) = \sum_{j=1}^{n+2} \gamma_j \boldsymbol{\alpha}(\omega_j)$.

The solution $\gamma \in \mathbb{R}^{n+2}$ of the linear system in line 1 of EXCHANGE satisfies the equation $\sum_{j=1}^{n+2} \gamma_j = 1$ (first row of the linear system). Hence, at least one of the γ_j is strictly positive, so that j_0 exists (line 2), though it is not necessarily unique.

Let us now show that $\boldsymbol{y'}$ is a feasible solution of Problem (D'_{n+2}) for $\boldsymbol{\omega'}$. As $\tilde{y}_* = \dfrac{y_{j_0}}{\gamma_{j_0}}$ with $y_{j_0} \geqslant 0$ and $\gamma_{j_0} > 0$, we have that $\tilde{y}_* \geqslant 0$. Similarly, the definition of $\tilde{y}_j = y_j - \gamma_j \tilde{y}_* = y_j - \gamma_j \min \left\{ \frac{y_j}{\gamma_j} \;\middle|\; \gamma_j > 0 \right\}$ clearly shows that $\tilde{y}_j = y'_j \geqslant 0,\ \forall\, j \in [1\,..\,n+2] \setminus \{j_0\}$ and $\tilde{y}_{j_0} = 0$. In addition, we have that

$$\begin{aligned}
\sum_{j=1}^{n+2} y'_j \alpha(\omega'_j) &= \sum_{j=1,\ j\neq j_0}^{n+2} y'_j \alpha(\omega'_j) + \tilde{y}_* \alpha(\omega_*) \\
&= \sum_{j=1,\ j\neq j_0}^{n+2} y_j \alpha(\omega_j) - \sum_{j=1,\ j\neq j_0}^{n+2} \gamma_j \tilde{y}_* \alpha(\omega_j) + \sum_{j=1,\ j\neq j_0}^{n+2} \gamma_j \tilde{y}_* \alpha(\omega_j) + \gamma_{j_0} \alpha(\omega_{j_0}) \tilde{y}_* \\
&= \sum_{j=1,\ j\neq j_0}^{n+2} y_j \alpha(\omega_j) + \alpha(\omega_{j_0}) y_{j_0} = \sum_{j=1}^{n+2} y_j \alpha(\omega_j) = z.
\end{aligned}$$

It follows that $\boldsymbol{y}'$ is a feasible solution of Problem (D'_{n+2}) for $\boldsymbol{\omega}'$, and actually the optimal solution since the feasible set is reduced to a singleton (uniqueness of the decomposition of $\boldsymbol{z}$ in a basis).

(2) Following similar steps as for point (1), $\mathrm{val}(D'_{n+2})$ for $\boldsymbol{\omega}'$ and $\boldsymbol{y}'$ can be written as

$$\begin{aligned}\mathrm{val}(D'_{n+2}) &= \sum_{j=1}^{n+2} y'_j c(\omega'_j) = \sum_{j=1,\ j\neq j_0}^{n+2} \tilde{y}_j c(\omega'_j) + \tilde{y}_* \alpha(\omega_*) \\ &= \sum_{j=1}^{n+2} y_j \alpha(\omega_j) + \frac{y_0}{\gamma_{j_0}} \left[c(\omega_*) - \sum_{j=1}^{n+2} c(\omega_j)\gamma_j \right].\end{aligned}$$

Note that $c(\omega_*) - \sum_{j=1}^{n+2} c(\omega_j)\gamma_j = c(\omega_*) - \alpha(\omega_*)x \geqslant 0$ where $\alpha^T(\omega_j)x = c(\omega_j)$ from SolvePrimal and the positivity is obtained from FindNewIndex. It is then concluded that $\mathrm{val}(D'_{n+2}) = \sum_{j=1}^{n+2} y'_j c(\omega'_j) \geqslant \sum_{j=1}^{n+2} y_j c(\omega_j)$.

(3) Finally, $\{\boldsymbol{\alpha}(\omega'_j)\}_{j\in[1..n+2]} = \{\boldsymbol{\alpha}(\omega_j)\}_{j\in[1..n+2]\setminus\{j_0\}\cup\{*\}}$ is a basis of $\mathbb{R}^{n+2}$ since $\gamma_{j_0} \neq 0$, i.e. $\boldsymbol{\alpha}(\omega_*)$ does not belong to the linear subspace spanned by $\{\boldsymbol{\alpha}(\omega_j)\}_{j\in[1..n+2]\setminus\{j_0\}}$. □

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

$$\overline{a}^{(\ell)} = \sum_{j=1}^{n+2} y_j^{(\ell)} c(\omega_j^{(\ell)}), \quad \text{and}$$

$$\overline{a}^{(\ell+1)} = \sum_{j=1}^{n+2} y_j^{(\ell+1)} c(\omega_j^{(\ell+1)}),$$

we have $\overline{a}^{(\ell+1)} - \overline{a}^{(\ell)} = \sum_{j=1}^{n+2} y_j^{(\ell+1)} c(\omega_j^{(\ell+1)}) - \sum_{j=1}^{n+2} y_j^{(\ell)} c(\omega_j^{(\ell)}) \geqslant 0$ by Lemma 4. □

5 OPTIMIZING THE RELATIVE ERROR

Since many interesting practical cases appearing in the implementation of `libms` involve the relative error, we also describe the modifications needed for the previously presented algorithms to handle these cases. When considering the relative error instead of of the absolute error, Problem (P_{general}) is replaced by

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

Two cases are worth mentioning. On the one hand, if f has constant sign over I (f does not cancel in I), Problem $(P^{\text{rel}}_{\text{general}})$ is equivalent to multiplying the error variable $\overline{a}$ by $|f|$ in Problem (P'_{general}), yielding the

following new linear SIP formulation

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

with

$$\begin{gathered} \boldsymbol{x} = (\overline{a}, \boldsymbol{a}) \in \mathbb{R}^{n+2},\ \boldsymbol{z} = (1, 0, \dots, 0) \in \mathbb{R}^{n+2}, \\ \boldsymbol{\alpha_r}(t, \sigma_0, \dots, \sigma_k) = (f(t), \sigma_0 \boldsymbol{\pi_0}^T(t) + \sum_{i=1}^{k} \sigma_i \boldsymbol{\pi_i}^T(t))^T \in \mathbb{R}^{n+2}, \\ \mathfrak{S} = \{-1, 0, 1\}^{k+1},\ \omega = (t, \sigma_0, \dots, \sigma_k) \in \Omega := I \times \mathfrak{S}. \end{gathered} \tag{15}$$

Therefore, replacing $\boldsymbol{\alpha}(\omega)$ by $\boldsymbol{\alpha_r}(\omega)$ 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 a relative setting.

On the other hand, f may cancel for certain values in I. In this case, the relative error may sometimes be defined by continuity, when the numerator $|f(t) - p(t)| + |\tilde{p}(t) - p(t)|$ also cancels for the same values in I (these values are often called apparent singularities). Interesting practical examples involve the case when f has a unique zero t_z, of finite order s in I. For instance, consider the function `exmp1`, $f(t) = \exp(t) - 1$, which has a unique zero $t_z = 0$, of order 1, in the interval $I = [-1/4, 1/4]$. Not only should we find a way to keep the relative error between the polynomial and the function bounded in a neighborhood of t_z but one also should particularly take care of discretizations $\boldsymbol{\omega}$ including $t = 0$, in the SolvePrimal and Exchange algorithms. For this particular example, a simple way to tackle this issue is to cancel the numerator of the relative error, by approximating f by a polynomial $p(t) = \sum_{i=1}^{n} a_i t^i$ (the constant coefficient is set to 0). More generally, when the function f has a unique zero t_z, of finite order s in I, it is suggested in the reference [20, Example 2.11], to work with the modified form

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}\in\mathbb{R}^d} & \boldsymbol{z}^T\boldsymbol{x} \\ \text{s.t.} & \tilde{\boldsymbol{\alpha}}_{\boldsymbol{r}}(\omega)^T\boldsymbol{x} \geqslant \tilde{c}(\omega), \quad \omega \in \Omega, \end{array} \tag{P^{rel2}}$$

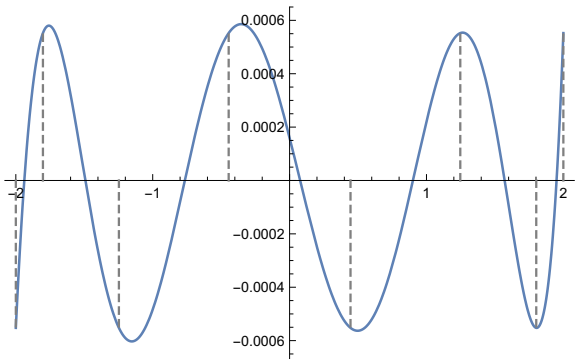
with

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

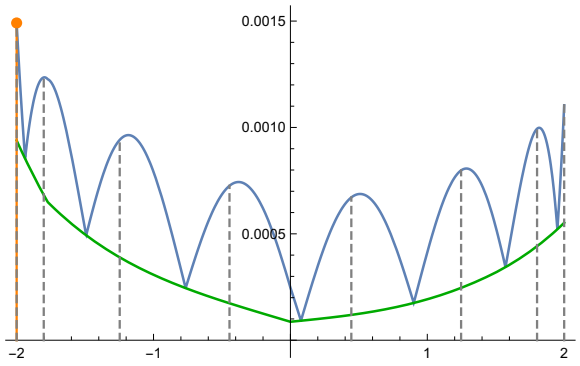
Problem P^{rel2} is indeed the equivalent formulation for the evaluation and approximation problem of the function $t \mapsto 1$ by the system of basis functions $(t - t_z)^s/f(t), (t - t_z)^{s+1}/f(t), \dots, (t - t_z)^n/f(t)$, considering the minimization of the absolute error. It is important to notice that the system $(t-t_z)^s/f(t), (t-t_z)^{s+1}/f(t), \dots, (t-t_z)^n/f(t)$ satisfies the so-called Haar condition [20, Example 2.11] for the initialization step, ensuring that the routine InitPoints works.

6 IMPLEMENTATION AND EXAMPLES

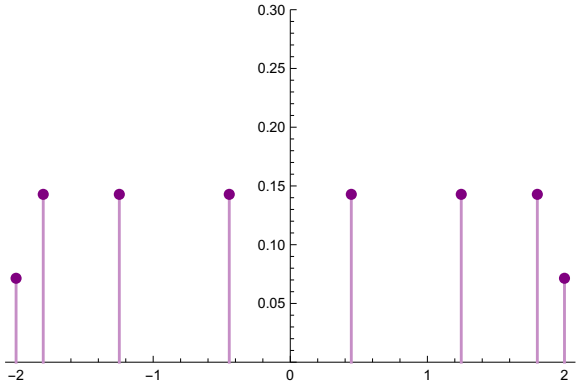
Algorithm Eval&ApproxOptimize is firstly illustrated by a tutorial example for Airy special function. Then, we detail our implementation and exemplify it on approximations with higher precision (binary64, double-extended, double double) coefficients of elementary functions like arcsin and the sigmoid (logistic) function.



(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

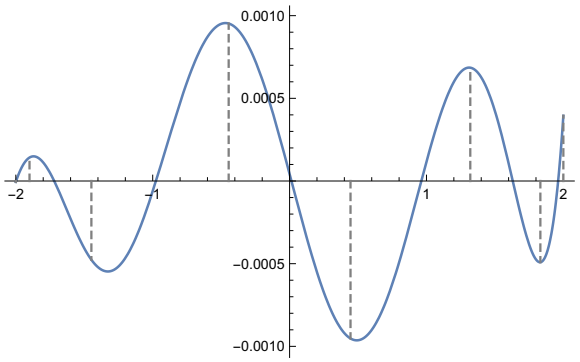
Fig. 1. Approximation of Ai over $[-2, 2]$: iteration 0

6.1 Example 1 - Airy function

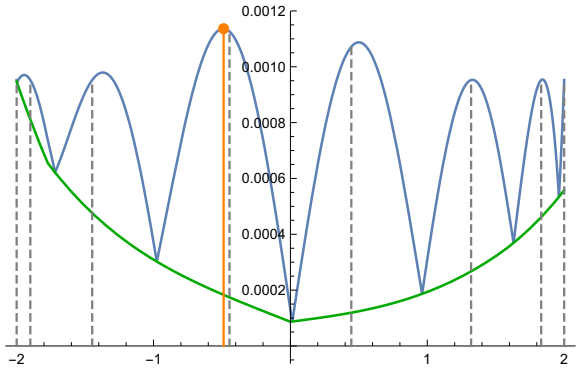
The Airy function Ai is a special function frequently used in theoretical physics. In particular, some applications require to compute $\mathrm{Ai}(t)$ for possibly large negative values of t, where the function exhibits a highly oscillatory behavior (see Figure 4a). However, contrary to elementary functions, 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.

6.1.1 Toy example in small precision. Let us firstly utilize a toy example for approximating Ai over $I = [-2, 2]$, 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 $\tau = 0.01$. The proof of concept code in Mathematica, used to produce the graphics in this example is available at https://gitlab.laas.fr/mmjoldes/xatom.

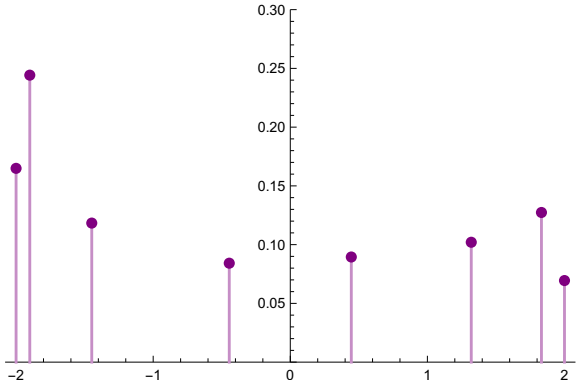
At iteration 0 (Figure 1), the points $t_j^{(0)}$ are initialized with the Chebyshev nodes and the signatures $\boldsymbol{\sigma_j}^{(0)}$ define a Remez-like system of linear equations on the coefficients of the polynomial (Figure 1d). Its solution $\boldsymbol{x}^{(0)} = (\overline{a}^{(0)}, \boldsymbol{a}^{(0)})$ 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



(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

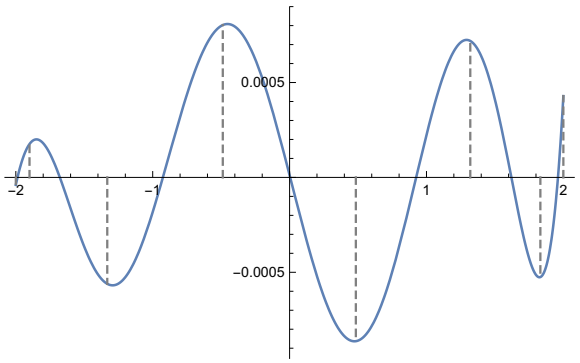
Fig. 2. Approximation of Ai over $[-2, 2]$: iteration 6

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 $\boldsymbol{\omega}^{(0)}$.

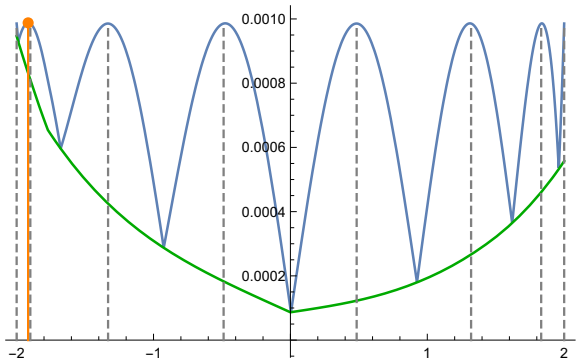
Moving forward to iteration 6 (Figure 2), the total error is more balanced, though still not optimal. Both the signatures and the approximation error are now completely different from the Remez solution.

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.

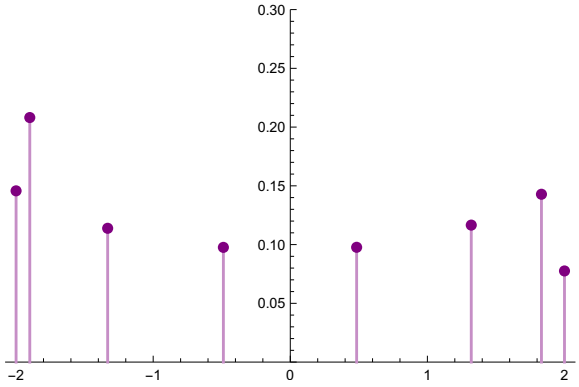
6.1.2 Further comparisons in single precision. We here consider the approximation of the Airy function over $I = [-4, 0]$ with polynomials evaluated using the Horner scheme with binary32 floating-point numbers, that is, single precision ($u = 2^{-24}$). Figure 4b shows the total error (approximation 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



(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

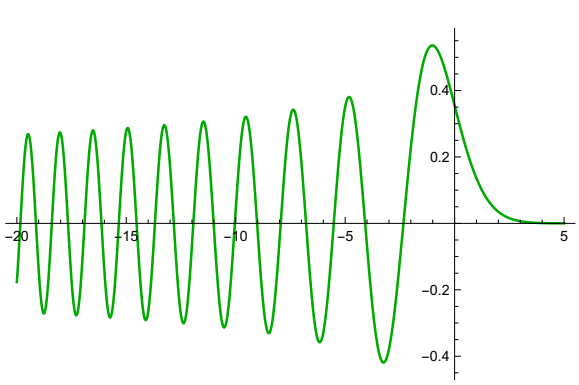
Fig. 3. Approximation of Ai over $[-2, 2]$: iteration 9

of polynomials obtained with EvalApproxOptimize (in yellow) continues to decrease 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.

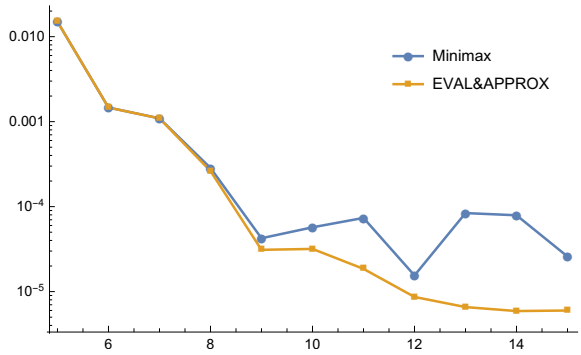
6.2 Implementation using the Sollya Library

We propose a `C` implementation of the described algorithms available at https://gitlab.laas.fr/mmjoldes/xatom. It handles multiple precision computations, based on the MPFR library [28] which allows for obtaining polynomial approximations with coefficients represented in various formats (including binary32, binary64, double-double, double-extended or any custom-precision floating-point format). The `C` code is interfaced via a so-called `externalprocedure` with the Sollya software tool [22], which is a state-of-the-art environment for safe floating-point code development, particularly targeted for `libm` implementations. This allows for direct comparisons with the available routines `remez` and `fpminimax`, which provide tuned polynomials optimized with respect to the approximation error and coefficients formats (without taking the evaluation error into account).

In order to handle the evaluation error, we provide additional `C` code which computes, for a polynomial evaluation scheme composed of terms $\mathrm{RN}(e', u_i)$ (as exemplified in Section 2 and Table 1), an evaluation error bound of the form $\theta(\boldsymbol{a}, t)$, required for Problem P'_{general}. Generic evaluation schemes can be handled, provided that the corresponding mathematical expression is a univariate polynomial (that is, the coefficients $\boldsymbol{a}$ appear



(a) Airy function Ai



(b) Total error for Minimax and EvalApproxOptimize polynomials, in function of degree

Fig. 4. Approximation of Ai over $[-4, 0]$

linearly in arithmetic expressions involving additions and multiplications, which additionally, depend only on one variable t).

To exemplify the code usage, we consider as in [34, Section 4.1.1.], the function $\mathrm{expm1}(x) = \exp(x) - 1$, on the interval $I = [-1/4, 1/4]$. We search for the single-precision coefficients of a polynomial of degree 5, which should minimize the total relative error, with respect to the Horner evaluation scheme. Firstly, an *evaluation scheme* input file contains on the first line: the evaluation model, followed on subsequent lines by specifications regarding which error terms should be considered in the order-i (linear, quadratic, etc.) error. For instance in Figure 5, the file `hornerIn.txt`, specifies that all linear terms are considered and the rounding error is bounded by $|u_i| \leqslant 2^{-24}$, for $1 \leqslant i \geqslant 9$. Note also that we consider a polynomial whose constant coefficient is zero by default, since we approximate with respect to the relative error which cancels at zero (cf. Section 5).

In Sollya, the corresponding code given in Figure 5 runs the external procedure `evalApproxOptim` which parses the evaluation scheme information and runs the proposed algorithm. The second parameter of the `evalApproxOptim` corresponds to a weight by which the absolute error is divided. In this case, this is f, which directly gives the relative error. The resulted polynomial is stored in p and, the total relative error estimated by our model as the sum of the relative approximation error $|1 - p/f|$ and the relative evaluation error model $\theta(\boldsymbol{a}, t)/f$ is plotted in Figure 6a. Its maximum is $2.7965 \cdot 10^{-7}$. Figure 6b provides a plot of a discretization of the actual total relative error obtained after the implementation. This shows that the mathematical model of the error may overestimate the actual attained error, as it is often the case for this kind of rounding error modeling.

For completeness, let us compare with the classical method, where the coefficients are obtained in two steps: the `remez` routine, followed by rounding of the coefficients to the requested format. In this case, our total relative mathematical error model (where only the coefficients are changed) as well as its corresponding actual estimation via a discretization are plotted in Figure 6c and 6d. The plot shows that the coefficients obtained in this case are not optimal with respect to the error model. Its maximum $2.973 \cdot 10^{-7}$ is higher than in the previous case. However, the discretizations seem to give similar numeric errors in this example, whose primary goal is to show the features of the implementation. Beside handling generic evaluation schemes and relative errors, our code allows for generating corresponding implementations in Gappa, C wih Mpfr or interpreted Sollya language. This in turn allows for generating all the plots in Figure 6, but the whole file was omitted for brevity.

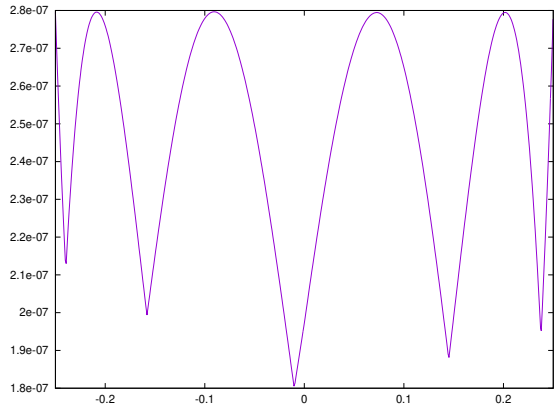
```
R(t*R(R(c1,u10)+R(t*R(R(c2,u11)+R(t*R(R(c3,u12)+R(t*R(R(c4,u13)+
R(t*R(c5,u14),u9),u8),u7),u6),u5),u4),u3),u2),u1)

1 all
--
24 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
> externalproc(evalApproxOptimize, "./result/evalApproxOptim",
(function, function, range, constant, integer, string, string) ->object);
> I = [-0.25, 0.25];
> tau = 0.001;
> fileIn = "hornerIn.txt";
> fileOut= "hornerOut.txt";
> f = expm1(x);
> prec = 100;
> maxIter=30;
> res = evalApproxOptim(f, f, I, tau, maxIter, fileIn, fileOut);
> p=head(res);
> approxError=res[1];
> evalError=res[2];
> a_star = res[3];
> plot(approxError+evalError, I);
```

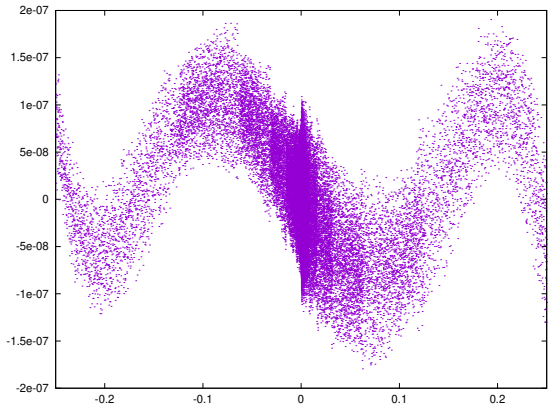
Fig. 5. The evaluation scheme file `hornerIn.txt` and the corresponding Sollya code.

6.3 Example - Sigmoid function

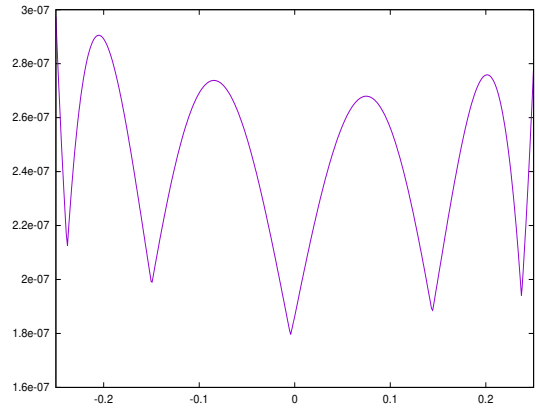
The sigmoid function is defined as $\texttt{sigmoid}(x) = 1/(1+\exp(-x))$ and is often used as an activation function for recurrent neural networks. Half-precision and single-precision implementations based on polynomial approximations are needed in practice, especially in the context of custom architectures on FPGAs [43], where implementations based on composition of $1/x$ and $\exp(x)$ are less efficient both in terms of circuit area and latency. This example is based on the features presented in [43]: for a single-precision implementation, the interval of interest is $[0, 16)$ (the values for the negative range are easily reconstructed from these); two interesting implementation choices of the authors were to consider either degree-3 polynomials on 256 equal-size subintervals, or respectively, degree-2 polynomials on 512 subintervals. Furthermore, they observed that evaluating these polynomials in single-precision arithmetic results in very low accuracies: the total relative error, that is the sum of the relative approximation and evaluation error for each of these polynomials is much worse than the approximation error alone, which renders these approximations less practical. We note for completeness, that the Taylor approximation also suffers from this phenomenon. In Figure 7 we compare the total errors obtained with our algorithm with respect to two state of the art techniques. Firstly, we consider the classical Remez exchange algorithm which solves Problem (P_{minimax}) and then rounds the coefficients to the requested format. Secondly, we compare with the improved algorithm, which uses euclidean lattices-based techniques in order to take into account the imposed coefficients formats [11] (but does not handle the evaluation error). Both algorithms are available in Sollya, under the so-called `remez` and `fpminimax` routines. The total obtained relative accuracy in b bits is plotted for each of the subintervals considered. That is, both the relative approximation and evaluation errors are taken into account, such that $|1-\tilde{p}_i(x)/\texttt{sigmoid}(x)| \leqslant 2^{-b}$ on each interval I_i, $(i = 1, \ldots, 256)$, where $\tilde{p}_i$



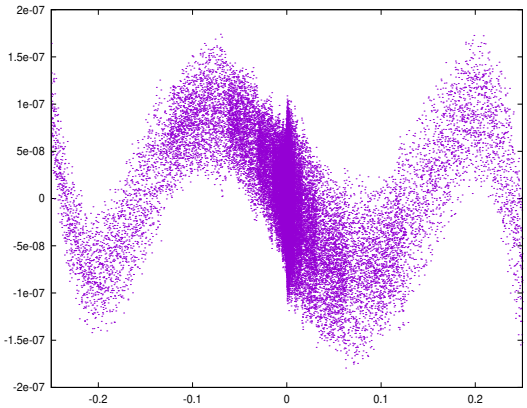
(a) Mathematical model of the total relative error; coefficients obtained by `evalApprox`.



(b) Discretization of the total relative error; coefficients obtained by `evalApprox`.



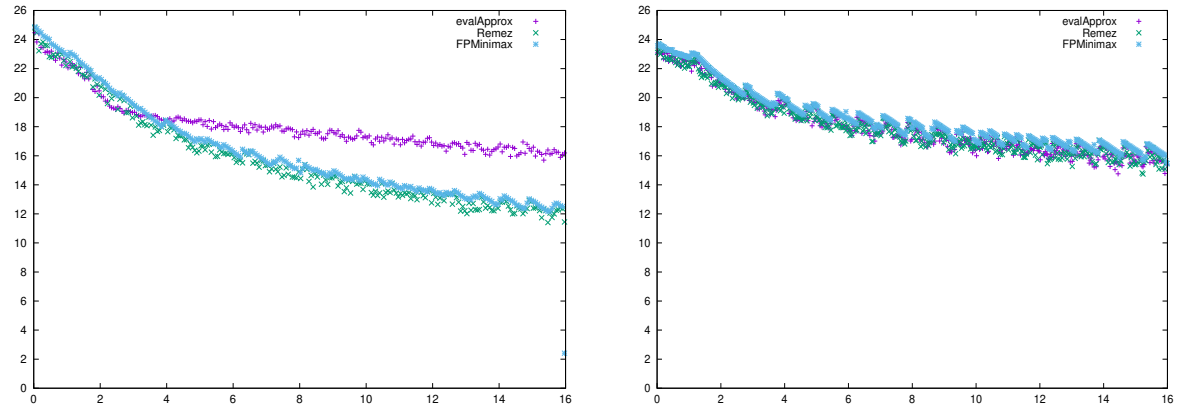
(c) Mathematical model of the total relative error; coefficients obtained by `remez` followed by rounding.



(d) Discretization of the total relative error; coefficients obtained by `remez` followed by rounding.

Fig. 6. Total relative error plots: the mathematical model versus the actual numerical discretization are analyzed. Two methods are compared: evalApprox vs. Remez with rounded coefficients.

represents the Horner scheme implementing the evaluation in single precision of the approximation polynomials p_i, obtained with the three algorithms. One observes an accuracy improvement when using our algorithm for the first case (Fig. 7a, degree-3 polynomials, coarser subdivision) on intervals which are further away from zero, while for intervals close to zero, `fpminimax` routine is slightly more accurate. For the second case (Fig. 7b, degree-2 polynomials, finer subdivision) all the three methods perform similarly, with `fpminimax` being slightly more accurate. While one could argue that both cases lead to the same accuracy in average, from the hardware implementation point of view, we note that the second case requires however twice as much storage (for the coefficients on each interval), while the first case demands an extra addition and multiplication – a final trade-off depending on the architecture has to be considered.



(a) Subdivision with 256 subintervals and degree-3 polynomials.

(b) Subdivision with 512 subintervals and degree-2 polynomials.

Fig. 7. Final accuracy (number of correct bits) on each subinterval, when using degree-2 or degree-3 polynomials (on each subinterval) and single-precision evaluation. Three methods are compared:present paper aka. evalApprox, Remez with rounded coefficients and FPMinimax.

6.4 Example 2 - Arcsine function

Consider $f = \arcsin$, over the interval $I = [b; 1]$, where the lower bound $b \approx 0.78$ is the binary64 floating number 0x3FE8F5C200000000[6]. This example is particularly insightful because f is ill-conditioned over I due to its singularity at 1. In this particular case, argument reduction techniques make it possible to tackle the singularity (see, e.g., the documentation of CRLibm [24] or the reference [11]). For illustrative purposes, however, we assume no such technique is available, as it would be the case for other singular or ill-conditioned functions known only through sampling via a black-box approach.

Our goal is to represent f over I using a degree 21 polynomial with binary64 coefficients, evaluated using a Horner's scheme with binary64 arithmetic operations as well. Since the range of f over I is roughly $[0.89, 1.57]$, the absolute or relative error treatment is similar, and we choose the former for the sake of simplicity.

Figure (8a) shows the absolute approximation error between f and the best degree 21 polynomial approximation p_{Remez} with *real* coefficients, for which $\max_I |f - p_{\text{Remez}}| \approx 4.42 \cdot 10^{-3}$. However, when rounding the coefficients of p_{Remez} to the nearest binary64 representable number, the resulting polynomial $\text{roundCoeff}(p_{\text{Remez}})$ becomes completely irrelevant, namely $\max_I |f - \text{roundCoeff}(p_{\text{Remez}})| \approx 1.85 \cdot 10^{12}$ (cf. Figure (8b)). This is due to the high magnitude of the coefficients. To search for better coefficients, one could use the competitive FPMinimax routine of Sollya [22], which optimizes on the coefficient space of representable FP numbers. The approximation error in this case is shown in Figure (8c) and one has $\max_I |f - p_{\text{FPMinimax})}| \approx 1.57$. Although this seems much better (even if still not sufficient to obtain at least one correct digit), when evaluating these polynomials with a Horner's scheme in binary64 floating-point arithmetic, the actual error sampled in Figures (8d) and (8e) using MPFR is catastrophic in both cases.

In contrast, our Algorithm EvalApproxOptimize computes a degree 21 polynomial $p_{\text{E\&A}}$ with binary64 coefficients. The total error found by our algorithm and verified in practice is $\max_I |f - p_{\text{E\&A}}| \approx 8.00 \cdot 10^{-3}$ (see Figures (8f) and (8g)). Hence the algorithm was able to find an optimal solution by moving away from the minimax polynomial to drastically reduce the evaluation error, at the expense of a factor less than 2 in the total

[6]This is one of the sub-intervals of $[-1, 1]$ considered for the implementation of the arcsin function in CRLibm [24].

error. One can then actually certify with Gappa [26] that the Horner scheme evaluation of $p_{\text{E\&A}}$ has an absolute error less than $1.17 \cdot 10^{-4}$.

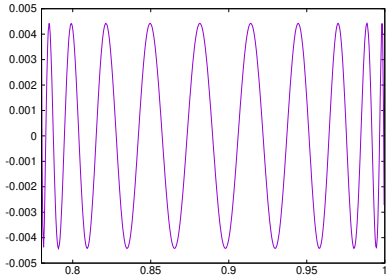
Finally, let us consider the same function on the interval $I_2 = [0.5; b]$ and approximate it with polynomials for a target relative error of 2^{-60}. Using the `guessdegree` routine of `sollya`, the minimum required degree is $d = 23$. The `remez` routine provides a polynomial with *real* (or very high precision coefficients, e.g. 300 bits in our experiments) with an error bounded by $2^{-62.77}$. When rounding the coefficients to `double extended` (DE, 64 bits mantissa), the approximation error quality drops to $2^{-30.67}$. The `fpminimax` routine allows for a very good tuning of the coefficients to fit the DE format, since we obtain a relative error bounded by $2^{-61.39}$. However when evaluating with a Horner scheme in DE operations, the resulting total error attains $2^{-29.34}$ (see Fig. 9). The Eval&ApproxOptimize routine provides a polynomial with DE coefficients which evaluates with the same DE scheme with an error of at most $2^{-48.56}$. It is interesting to note that increasing the approximation degree corresponds to decreasing the approximation error when *real* (high precision) coefficients are considered. However, when the coefficient format is imposed (as mentioned before), the total error does not improve, as shown in Table 2. In fact, for the `fpminimax` routine, the precision deteriorates towards no significant bit, while for our algorithm, the total error seems to slowly converge towards 52 precision bits. For completeness, we also mention in the same table the results obtained with the Estrin evaluation scheme [27] (which can be handled by our code): both evaluation schemes provide similar accuracy, with Estrin being slighly less accurate.

Method	Degree				
	23	24	25	26	27
Rel. Approx. Error (2^{-p})					
Remez	62.77	65.16	67.55	69.94	72.32
Rounded Remez	30.67	27.04	28.15	23.85	21.24
FPMinimax	61.39	61.39	61.39	61.39	0
Total Rel. Error with Horner Scheme (2^{-p})					
Rounded Remez	29.30	25.90	24.25	22.66	20.08
FPMinimax	29.34	26.90	24.25	22.66	0
Eval&ApproxOptimize	48.56	49.45	50.40	51.10	51.67
Total Rel. Error with Estrin Scheme (2^{-p})					
Rounded Remez	28.31	25.43	24.4	21.93	20.08
FPMinimax	27.89	26.43	23.76	22.05	0
Eval&ApproxOptimize	48.43	49.35	50.11	51.04	51.51

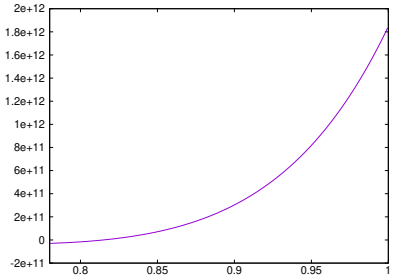
Table 2. Relative errors for successive degrees, when approximating $f = \arcsin$ over $I = [0.5, 0\text{x}3\text{FE}8\text{F}5\text{C}200000000]$ and Horner/Estrin scheme in `double extended` (DE) arithmetic: Comparing the minimax polynomial (a.k.a. Remez), the rounded minimax polynomial aka. Rounded Remez, the polynomial obtained avec FPMinimax and polynomial obtained using our Algorithm Eval&ApproxOptimize.

7 CONCLUSION

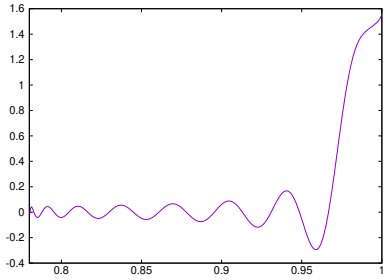
In this article, we proposed a contribution to the field of fixed-precision implementation of mathematical functions that relies on the powerful and versatile framework of Linear Semi-Infinite Programming (LSIP). Specifically, we proposed a first formulation of an LSIP optimization problem for the total error (approximation plus evaluation



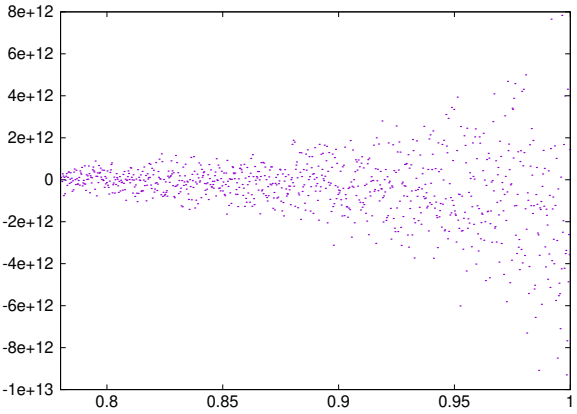
(a) Approx. error $f - p_{\text{Remez}}$



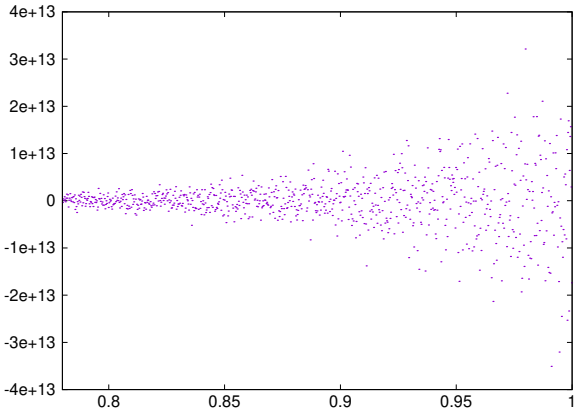
(b) Approx. error $f - \text{roundCoeff}(p_{\text{Remez}})$



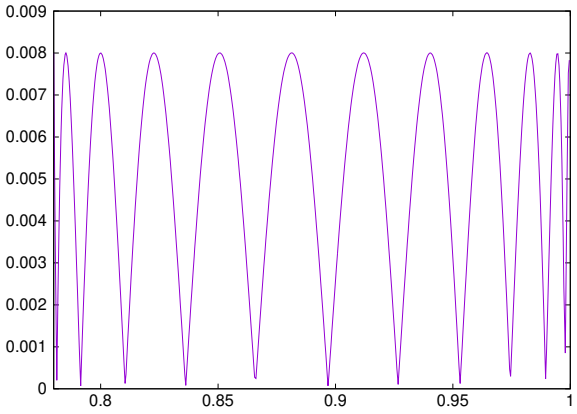
(c) Approx. error $f - p_{\text{FPMinimax}}$



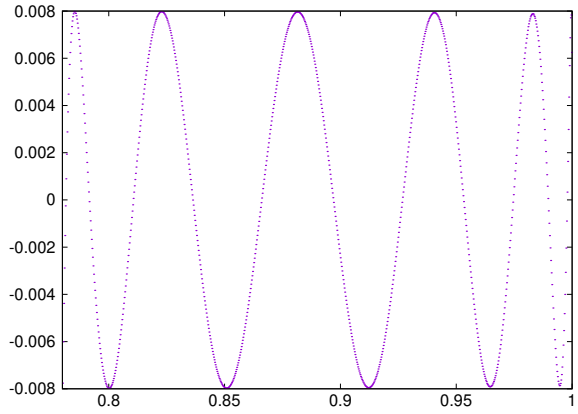
(d) Error between roundCoeff(p_{Remez}) evaluated in binary64 and f evaluated using MPFR



(e) Error between $p_{\text{FPMinimax}}$ evaluated in binary64 and f evaluated using MPFR

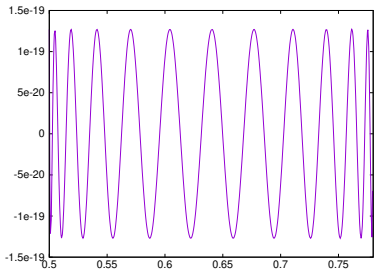


(f) Total (= approximation + evaluation) error $|f - p_{\text{E\&A}}|$, in absolute value

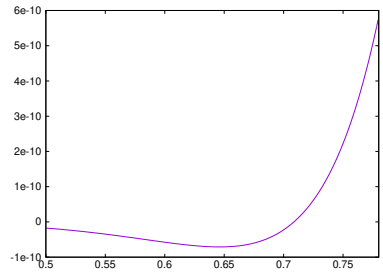


(g) Error between $p_{\text{E\&A}}$ evaluated in binary64 and f evaluated using MPFR

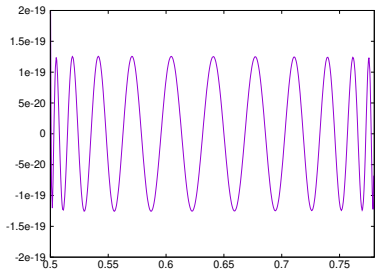
Fig. 8. Approximating $f = \arcsin$ over $I = [0\text{x}3\text{FE}8\text{F}5\text{C}200000000, 1.]$ using a degree 21 polynomial and a Horner's scheme in binary64 arithmetic: Comparing the minimax polynomial p_{Remez}, the rounded minimax polynomial roundCoeff(p_{Remez}) and the polynomial $p_{\text{E\&A}}$ obtained using our Algorithm EvalApproxOptimize.



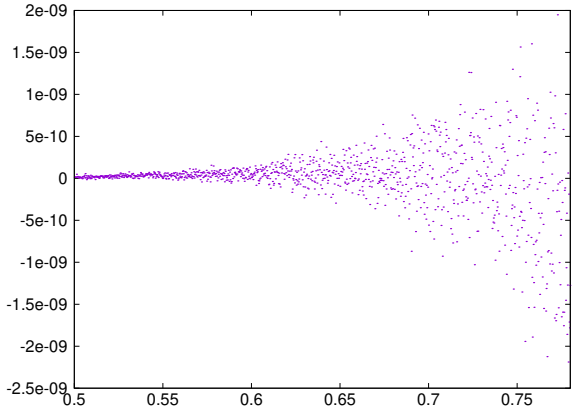
(a) Approx. error $f - p_{\text{Remez}}$



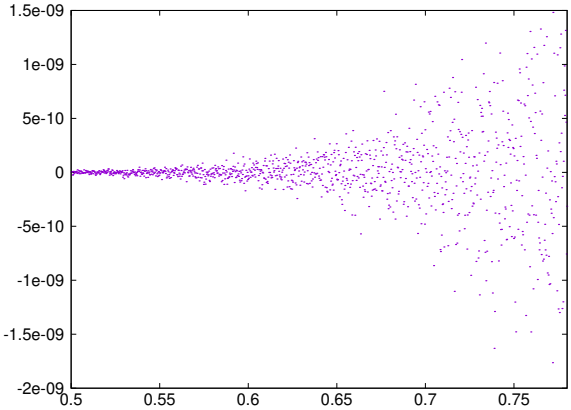
(b) Approx. error $f - \text{roundCoeff}(p_{\text{Remez}})$



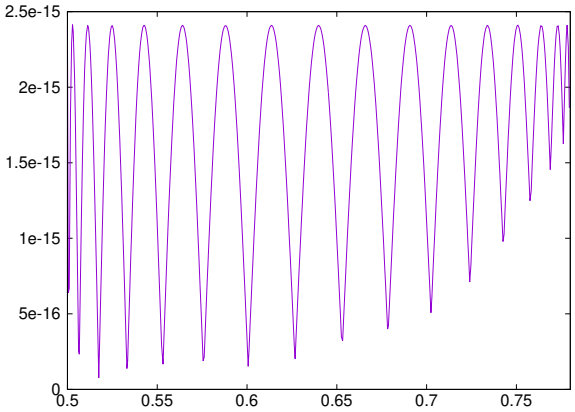
(c) Approx. error $f - p_{\text{FPMinimax}}$



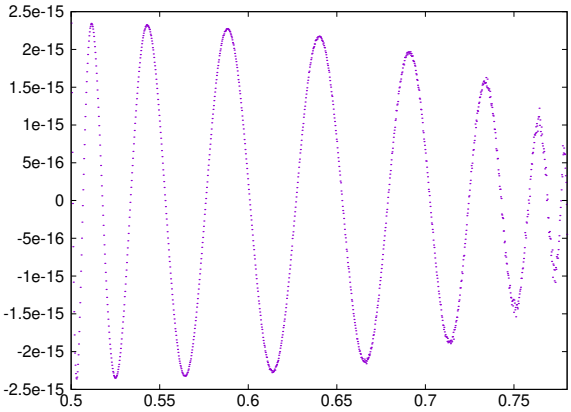
(d) Relative error between $\text{roundCoeff}(p_{\text{Remez}})$ evaluated in DE and f



(e) Relative error between $p_{\text{FPMinimax}}$ evaluated in DE and f



(f) Total (= approximation + evaluation) error $|1 - p_{\text{E\&A}}/f|$



(g) Relative error between $p_{\text{E\&A}}$ evaluated in DE and f

Fig. 9. Approximating $f = \arcsin$ over $I = [0.5, 0\text{x}3\text{FE}8\text{F}5\text{C}200000000]$ using a degree 23 polynomial and a Horner's scheme in `double extended` (DE) arithmetic: Comparing the minimax polynomial p_{Remez}, the rounded minimax polynomial $\text{roundCoeff}(p_{\text{Remez}})$ and the polynomial $p_{\text{E\&A}}$ obtained using our Algorithm EvalApproxOptimize.

errors) of polynomials used for the evaluation of mathematical functions, with a *black-box* description. This allows for handling very general functions (elementary, special, etc.), but also implies that no argument reduction step is usually possible. To reformulate the problem as an LSIP instance, we bounded the evaluation error using the classical relative rounding error model of floating-point arithmetic. Then, we described an exchange algorithm to solve this particular problem, which can be seen as an extension of Remez' exchange algorithm that optimizes the approximation error only. This work was fully implemented in C, using the Sollya library and various numerical experiments were carried out to assess its practical efficiency.

The conclusion is twofold. On certain examples with potentially high evaluation error, like for instance Horner schemes on non-zero-centered intervals, approximations on intervals close to function's singularities, etc.), our algorithm is able to improve the total error, in order to keep the constrained precision(s) in the evaluation scheme reasonable. Furthermore, the proposed algorithm scales to higher precisions imposed on the coefficients (examples with double, double-extended, double-double were presented) compared to other linear programming inspired approaches available. On the other hand, for examples where the coefficients of the approximation polynomial are highly constrained (typical cases are smooth functions on small, zero-centered intervals), our method provides polynomials that are very similar to the ones obtained with the classical Remez algorithm. The main observation is that, in such cases, the continuous relative rounding error model employed may be too conservative (as seen in Example 6.2) since the bound in u is not attained in practice for each operation.

Therefore, this opens the question for several future extensions. Concerning the modeling of the evaluation error, one can take into account a *stochastic* error model like in [31], in order to optimize the *average* total error. In addition, while our current implementation directly allows for the estimation of evaluation errors for other numerical schemes, a further extension could also take other polynomial bases [6] into account. Finally, our work opens the way for a mixed-integer linear programming formulation in the provided optimization framework. However, the theoretical properties of the obtained problems need to be further studied and a similar exchange procedure in this case is not direct.

Acknowledgments This research was partly supported by ANR NuSCAP project funded by French Research Agency. We would like to thank Nicolas Brisebarre for helpful remarks that led to major improvements in our work.

REFERENCES

[1] 2019. *IEEE Standard for Floating-Point Arithmetic*. 1–84 pages. https://doi.org/10.1109/IEEESTD.2019.8766229

[2] M. Abramowitz and I.A. Stegun. 1964. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. National Bureau of Standards Applied Mathematics Series, Vol. 55. Courier Corp. xiv+1046 pages.

[3] E. J. Anderson and P. Nash. 1987. *Linear Programming in Infinite-Dimensional Spaces*. John Wiley and Sons, Chichester, Great Britain.

[4] D. Arzelier, F. Bréhard, and M. Joldes. 2019. Exchange algorithm for evaluation and approximation error-optimized polynomials. In *ARITH 2019 - 26th IEEE Symposium on Computer Arithmetic*. IEEE, Kyoto, Japan, 1–8. https://hal.science/hal-02006606

[5] D. Arzelier, F. Bréhard, and M. Joldes. 2019. Exchange algorithm for evaluation and approximation error-optimized polynomials. (2019). Research report, https://hal.archives-ouvertes.fr/hal-02006606.

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

[7] J.-F. Bonnans and A. Shapiro. 2000. *Perturbations Analysis of Optimization Problems*. Springer Verlag, New York, NY, USA.

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

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

[10] I. Briggs, Y. Lad, and P. Panchekha. 2024. Implementation and Synthesis of Math Library Functions. *Proc. ACM Program. Lang.* 8, POPL, Article 32 (jan 2024), 28 pages. https://doi.org/10.1145/3632874

[11] N. Brisebarre and S. Chevillard. 2007. Efficient polynomial L^∞ approximations. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*. Montpellier, France, 169–176. https://doi.org/10.1109/ARITH.2007.17

[12] N. Brisebarre, G. Hanrot, J.-M. Muller, and P. Zimmermann. 2024. Correctly-rounded evaluation of a function: why, how, and at what cost? (May 2024). https://hal.science/hal-04474530 working paper or preprint.

[13] N. Brisebarre, J.-M. Muller, and N. Tisserand. 2006. Computing machine-efficient polynomial approximations. *ACM Trans. Math. Software* 32, 2 (June 2006), 236–256.
[14] N. Brunie, F. De Dinechin, O. Kupriianova, and C. Lauter. 2015. Code Generators for Mathematical Functions. In *2015 IEEE 22nd Symposium on Computer Arithmetic*. 66–73. https://doi.org/10.1109/ARITH.2015.22
[15] N. Brunie, C. Lauter, and G. Revy. 2019. Precision adaptation for fast and accurate polynomial evaluation generation. In *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, Vol. 2160. IEEE, 41–41.
[16] C. Carasso. 1973. *L'algorithme d'échange en optimisation convexe.* Ph. D. Dissertation. Université Joseph-Fourier-Grenoble I.
[17] C. Carasso and P.-J. Laurent. 1976. Un algorithme général pour l'approximation au sens de Tchebycheff de fonctions bornées sur un ensemble quelconque.. In *Approximation Theory (Lecture Notes in Mathematics, 556)*. Springer Berlin Heidelberg, 99–121.
[18] B.L. Chalmers. 1976. The Remez exchange algorithm for approximation with linear restrictions. *Trans. Amer. Math. Soc.* 223 (1976), 103–131.
[19] E.W. Cheney. 1982. *Introduction to Approximation Theory.* AMS Chelsea Publishing, Providence, RI.
[20] S. Chevillard. 2009. *Évaluation efficace de fonctions numériques. Outils et exemples.* Ph. D. Dissertation. École Normale Supérieure de Lyon.
[21] S. Chevillard, J. Harrison, M. Joldeş, and C.Q. Lauter. 2011. Efficient and accurate computation of upper bounds of approximation errors. *Theoret. Comput. Sci.* 412, 16 (2011), 1523–1543. https://doi.org/10.1016/j.tcs.2010.11.052
[22] S. Chevillard, M. Joldeş, and C.Q. Lauter. 2010. Sollya: An environment for the development of numerical codes. In *International Congress on Mathematical Software*. Springer, 28–31.
[23] G.B. Dantzig and M.N. Thapa. 2003. *Linear programming 2: theory and extensions.* Springer, New York, NY, USA.
[24] C. Daramy-Loirat, D. Defour, F. de Dinechin, M. Gallet, N. Gast, C.Q. Lauter, and J.-M. Muller. 2006. *CR-LIBM, A library of correctly-rounded elementary functions in double-precision.* Technical Report. LIP Laboratory, Arenaire team.
[25] E. Darulova and V. Kuncak. 2017. Towards a Compiler for Reals. *ACM Trans. Program. Lang. Syst.* 39, 2, Article 8 (March 2017), 28 pages.
[26] M. Daumas and G. Melquiond. 2010. Certification of bounds on expressions involving rounded operators. *ACM Trans. Math. Software* 37, 1 (2010), Art. 2, 20. https://doi.org/10.1145/1644001.1644003
[27] G. Estrin. 1960. Organization of computer systems: the fixed plus variable structure computer. In *Papers Presented at the May 3-5, 1960, Western Joint IRE-AIEE-ACM Computer Conference* (San Francisco, California) *(IRE-AIEE-ACM '60 (Western))*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/1460361.1460365
[28] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. 2007. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw.* 33, 2 (2007), 13–es.
[29] K. Glashoff and S.-Å. Gustafson. 1983. *Linear Optimization and Approximation.* Springer, New York, NY, USA.
[30] M.A. Goberna and M.A. López. 1998. *Linear Semi-Infinite Optimization.* Wiley, West Sussex, England.
[31] N.J. Higham and T. Mary. 2019. A new approach to probabilistic rounding error analysis. *SIAM journal on scientific computing* 41, 5 (2019), A2815–A2835.
[32] IEEE Computer Society. 2008. *IEEE Standard for Floating-Point Arithmetic.* IEEE Standard 754-2008. Available at http://ieeexplore.ieee.org/servlet/opac?punumber=4610933.
[33] O. Kupriianova and C.Q. Lauter. 2014. Metalibm: A mathematical functions code generator. In *International Congress on Mathematical Software*. Springer, 713–717.
[34] C.Q. Lauter. 2008. *Arrondi Correct de Fonctions Mathématiques.* Ph. D. Dissertation. ÉNS de Lyon, Lyon, France.
[35] C.Q. Lauter and M. Mezzarobba. 2015. Semi-Automatic Floating-Point Implementation of Special Functions. In *ARITH 22*, J.-M. Muller, A. Tisserand, and J. Villalba (Eds.). IEEE, Lyon, France.
[36] J.P. Lim. 2021. *Novel polynomial approximation methods for generating correctly rounded elementary functions.* Ph. D. Dissertation. Rutgers The State University of New Jersey, School of Graduate Studies.
[37] J.P. Lim, M. Aanjaneya, J. Gustafson, and S. Nagarakatte. 2021. An approach to generate correctly rounded math libraries for new floating point variants. *Proc. ACM Program. Lang.* 5, POPL, Article 29 (jan 2021), 30 pages. https://doi.org/10.1145/3434310
[38] D. G. Luenberger. 1969. *Optimization by Vector Space Methods.* John Wiley and Sons, New York, NY, USA.
[39] J.-M. Muller. 2016. *Elementary Functions, Algorithms and Implementation* (3rd ed.). Birkhäuser, Boston.
[40] J.-M. Muller. 2020. Elementary Functions and Approximate Computing. *Proc. IEEE* 108, 12 (2020), 2136–2149. https://doi.org/10.1109/JPROC.2020.2991885
[41] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldeş, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. 2018. *Handbook of Floating-Point Arithmetic.* Birkhäuser Boston, Boston, MA,USA.
[42] J. Oliver. 1979. Rounding error propagation in polynomial evaluation schemes. *J. Comput. Appl. Math.* 5, 2 (1979), 85–97.
[43] B. Pasca and M. Langhammer. 2018. Activation function architectures for FPGAs. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 43–437.
[44] M.J.D. Powell. 1981. *Approximation Theory and Methods.* Cambridge University Press. https://doi.org/10.1017/CBO9781139171502
[45] R. Reemtsen and J.-J. Rückmann. 1998. *Semi-Infinite Programming*. Vol. 25. Springer Science & Business Media.

[46] E. Remez. 1934. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation: Comptes Rendus de l'Académie des sciences. 198 (1934), 2063–2065.
[47] A. Rocca, V. Magron, and T. Dang. 2017. Certified Roundoff Error Bounds using Bernstein Expansions and Sparse Krivine-Stengle Representations. In *24th IEEE Symposium on Computer Arithmetic*. IEEE.
[48] W.W. Rogosinsky. 1958. Moments of non-negative mass. *Proceedings of the Royal Society London, Series A* 245 (1958), 1–27.
[49] H. Royden and P. Fitzpatrick. 2010. *Real Analysis*. Prentice Hall, Boston, MA, USA.
[50] A. Shapiro. 2009. Semi-infinite programming, duality, discretization and optimality conditions. *Optimization* 58, 2 (2009), 133–161.
[51] A. Solovyev, M.S. Baranowski, I. Briggs, C. Jacobsen, Z. Rakamarić, and G. Gopalakrishnan. 2018. Rigorous Estimation of Floating-Point Round-Off Errors with Symbolic Taylor Expansions. *ACM Trans. Program. Lang. Syst.* 41, 1, Article 2 (Dec. 2018), 39 pages.
[52] A. Tisserand. 2007. High-performance hardware operators for polynomial evaluation. *International Journal of High Performance Systems Architecture (IJHPSA)* 1, 1 (2007), 14–23.
[53] L.N. Trefethen. 2019. *Approximation theory and approximation practice, extended edition*. SIAM.
[54] L. Veidinger. 1960. On the numerical determination of the best approximations in the Chebyshev sense. *Numer. Math.* 2 (1960), 99–105.
[55] G.A. Watson. 1974. The calculation of best restricted approximations. *SIAM J. Numer. Anal.* 11, 4 (1974), 693–699.