Towards Machine-Efficient Rational L^∞-Approximations of Mathematical Functions

Nicolas Brisebarre
Univ. Lyon, CNRS, LIP
F-69342 Lyon, France
Email: Nicolas.Brisebarre@ens-lyon.fr

Silviu-Ioan Filip
Univ. Rennes, Inria, CNRS, IRISA
F-35000 Rennes, France
Email: silviu.filip@inria.fr

***Abstract*—Software implementations of mathematical functions often use approximations that can be either polynomial or rational in nature. While polynomials are the preferred approximation in most cases, rational approximations are nevertheless an interesting alternative when dealing with functions that have a pronounced "nonpolynomial behavior" (such as poles close to the approximation domain, asymptotes or finite limits at $\pm\infty$).**

The major challenge is that of computing good rational approximations with machine number coefficients (*e.g.* floating-point or fixed-point) with respect to the supremum norm, a key step in most procedures for evaluating a mathematical function. This is made more complicated by the fact that even when dealing with real-valued coefficients, optimal supremum norm solutions are sometimes difficult to obtain. Here, we introduce flexible and fast algorithms for computing such rational approximations with both real and machine number coefficients. Their effectiveness is explored on several examples.

I. Introduction

While software and hardware evaluation of mathematical functions in floating-point (FP) arithmetic is often done using polynomial approximations, rational functions can sometimes be a viable alternative, especially when dealing with special functions whose behavior is not easily captured by a polynomial proxy. An example of this is the SPECFUN [1] FORTRAN package, which used rational approximations in the computation of exponential integrals, error, and gamma-related functions on the real line.

Evaluating a function using FP arithmetic is usually reduced to determining a set of approximations on (small) compact domains. For each domain B and function f to compute with a target accuracy $\varepsilon > 0$, the task is to find an implementation $\tilde{f}$ such that $\left\|(f-\tilde{f})/f\right\|_B \leqslant \varepsilon$, where $\|g\|_B = \max_{x\in B}|g(x)|$ is the L^∞ norm of g on B. This process is typically handled in two steps: (a) approximation, and (b) evaluation.

In the first step, an approximation r (polynomial or rational) is searched for, with coefficients that are representable in a certain target format (such as IEEE-754-2019 [2] `binary32` and `binary64`, or unevaluated sums of terms in these formats) and approximation error that is below a threshold $\varepsilon_{\text{approx}}$, either in an absolute $\|f-r\|_B \leqslant \varepsilon_{\text{approx}}$ or a relative $\|(f-r)/f\|_B \leqslant \varepsilon_{\text{approx}}$ sense. Depending on the evaluation scheme chosen, one must then ensure that the computed value $\tilde{r}$ satisfies $\|r-\tilde{r}\|_B \leqslant \varepsilon_{\text{eval}}$ or $\|(r-\tilde{r})/f\|_B \leqslant \varepsilon_{\text{eval}}$, for a given threshold $\varepsilon_{\text{eval}}$. This is usually certified a posteriori, with various methods and tools being available [3]–[5].

This work is partly sponsored by the ANR Nuscap (ANR-20-CE48-0014) project of the French Agence Nationale de la Recherche and the COMINLabs (10-LABX-0007) LeanAI project.

Despite their potential, rational approximations seem to be relied on less and less in the design of mathematical libraries (*libms*). There are several reasons for this. Whereas polynomial approximations use only low latency floating-point addition ($+$) and multiplication ($\times$) operations, rational function evaluation requires division ($/$), whose latency can be up to ten times larger on modern processors [6, Table 3.5] with respect to $+$ and $\times$. Polynomials also benefit from powerful tools such as Sollya [3] and its `fpminimax` algorithm [7] for generating machine-coefficient approximations (see also [8], [9] for other methods), whereas equally flexible and general tools for rational functions have been missing in the community. The `fpminimax` approach in particular was only adapted to rational approximations with bounded magnitude coefficients [10], [11]. Determining such rational approximations is more difficult than in the polynomial case.

The problems start in the real-coefficient setting. While the Remez algorithm [6, Ch. 3.6] and its generalizations (*e.g.* the Remez-Stiefel version from Sollya [12, Ch. 2.2]) have good convergence properties (usually guaranteed and quadratic [13]), rational L^∞ approximation can be hindered by degeneracy [14] problems. This makes rational extensions of the Remez algorithm more fragile (see [15] and references therein), with no guarantee of convergence in all cases. To our knowledge, there are no implementations that are sufficiently robust to handle the kind of problems that would need to be solved in a libm context. Existing codes (*e.g.* the `minimax` commands in Maple and Chebfun [15]) are restricted to full monomial basis approximations and/or cannot produce approximations with errors smaller than machine precision.

The availability of tools such as Sollya have also made possible the development of powerful automatic code generators for function implementations that are primarily based on polynomials [16], [17], giving even less incentive to the libm designer to explore the use of rational approximations.

Our goal is therefore two-fold. On the algorithmic side, we introduce a highly robust method that is capable of producing generalized real-coefficient rational approximations (*e.g.* with incomplete monomial bases that frequently occur in practice). The result of this algorithm serves as input to an `fpminimax`-like approach for generating machine-coefficient approximations. On the practical side, we have developed a C++ library that should make experimenting with rational function approximations in a libm context much easier and close to par with polynomial-based offerings. We showcase its capabilities on several examples and discuss certain limitations and directions for future work.

II. An overview of L^∞ approximation problems

We start with a general presentation (using notation introduced in [18]) of the kind of problems we want to solve and the various approaches that have been devised to address them.

A. The problem

Let $(X, \|\cdot\|_X)$ be an N-dimensional normed linear space and let A be a nonempty subset of X. For a compact set $B \subset \mathbb{R}^s, s \geqslant 1$, let $C(B)$ be the space of all real-valued continuous functions on B. Given $f \in C(B)$ and a family $T(A)$ of functions for the approximation of f defined by an operator $T : A \to C(B)$ that maps A continuously into $C(B)$, we want to solve problems of the form

$$P[D] : \text{minimize}\,\{\|f - T\mathbf{a}\|_D, \mathbf{a} \in A\}$$

for $D \subseteq B$. We denote with $\mu(D) = \min_{\mathbf{a}\in A} \|f - T\mathbf{a}\|_D$ the L^∞ error associated with a solution $\mathbf{a}$ to $P[D]$.

The most prevalent use cases that interest us here, for $s = 1$ and a closed interval domain $B = [a, b]$, are:

(a) linear approximations of the form

$$T\mathbf{a} = \sum_{i=1}^{N} a_i\phi_i,$$

where $\phi_1, \dots, \phi_N \in C(B)$ are linearly independent and $\mathbf{a} = [a_1, \dots a_N] \in A = \mathbb{R}^N$. The monomial basis of $\mathbb{R}_{N-1}[x]$ is a classic example and is the default basis for polynomial approximations of elementary functions.

(b) rational approximations of the form

$$T\mathbf{a} = \frac{\sum_{i=1}^{m} p_i\phi_i}{\sum_{i=1}^{n} q_i\psi_i},$$

where $N = m + n, \mathbf{a} = [p_1, \dots, p_m, q_1, \dots, q_n] \in A = \mathbb{R}^N$ and $\{\phi_i\}_{i=1}^m, \{\psi_i\}_{i=1}^n$ are families of linearly independent functions from $C(B)$. Again, the default examples are $\mathbb{R}_{m-1}[x]$ and $\mathbb{R}_{n-1}[x]$, respectively.

B. The algorithmic landscape

Iterative methods that discretize the approximation domain are generally used to approach a (global) solution to $P[B]$.

A first family of algorithms (see [19] and the references therein) successively solves discrete problems $P[B_k]$, where $\{B_k\}_{k\geqslant 0}$ is a sequence of grids in B with density tending to zero. While they can work well under mild constraints, the discretizations can grow quite large, making them slow and prone to numerical issues.

A second family of methods, of so-called Remez-type, require only the solution of relatively small discrete problems, but with the added complexity of having to determine the global extremum of the error function over B at each iteration, which can be hard to do (*e.g.* in multivariate approximation contexts). There are two main flavours of Remez algorithms [20, Ch. 3.8], *first* and *second*. Algorithms of the first type increase the size of B_k at each iteration k with points where the extremum of $P[B_k]$ is attained, whereas methods of the second kind keep the size of B_k constant, *exchanging* the elements of B_k at each iteration. While fast (quadratic convergence is frequently attainable [13], [21]), the use of such exchange algorithms is usually dependent on an alternation theorem [20, p. 75], which for linear T states that there exist $N + 1$ distinct points in B where the error function $f - T\widehat{\mathbf{a}}$ attains its extremum $\mu(B)$ with alternating sign. In the linear case, recent highly scalable implementations of the Remez exchange algorithm are described in [22], [23].

This alternating error result can be degenerate [20, p. 161] (*i.e.,* the error sometimes equioscillates at strictly less than N points) in the rational approximation case, causing the algorithm to fail in such scenarios. Failure can also occur if the starting approximation is too far from an optimal one.

C. A general method

To cope with such failures and have a more generalizeable approach, over the years there have been attempts to combine the two families of methods. In particular, Reemtsen [18, Sec. 2] derived a modified version of the first Remez algorithm that does not necessarily require the computation of global extrema over B. It considers that A is a closed subset of X. Taking the closed level sets $L_\alpha(D) = \{\mathbf{a} \in A, \|T\mathbf{a}\|_D \leqslant \alpha\}$ for a compact $D \subseteq B$ and $\alpha \geqslant 0$, it assumes:

(A1) $\{B_k\}_{k\geqslant 0}$ is a given sequence of finite sets in B, with $B_k \subseteq B_{k+1} \subseteq B, \forall k \geqslant 0$, and $\lim_{k\to\infty} h(B_k, B) = 0$, where $h(B_k, B) = \sup_{x\in B} \inf_{y\in B_k} \|x-y\|_\infty$ and $\|\cdot\|_\infty$ is the infinity norm in $\mathbb{R}^s$;

(A2) For $\alpha_0 = \|f - T\mathbf{a}_0\|_B + \|f\|_B$, where $\mathbf{a}_0$ is an arbitrary element of A, the level set $L_{\alpha_0}(B_0)$ is bounded (and hence compact) in X.

Defining the set of extremal points of $g \in C(B)$ on a compact $D \subseteq B$ to be $E(g, D) = \{x \in D, |g(x)| = \|g\|_D\}$, the algorithm is as follows:

Algorithm 1 Generalized First Remez Algorithm [18]

```
Step 1. $k \leftarrow 0$ and $D_0 = B_0$.
Step 2. Find $\mu(D_k)$ and a solution $\widehat{\mathbf{a}}_k \in A$ of $P[D_k]$. Set
        $e_k = f - T\widehat{\mathbf{a}}_k$.
Step 3. Compute $x_k \in E(e_k, B_{k+1})$ and a set $D_{k+1} \subseteq$
        $B_{k+1}$ with $D_{k+1} \supseteq D_k \cup \{x_k\}$.
Step 4. $k \leftarrow k + 1$ and go to Step 2.
```

The choice of B_0, how **Steps 2 & 3** are performed, and stopping criteria are generally dependent on the family of approximations $T(A)$ and on the dimension of B. We address these aspects for the rational case in Sec. III.

While in the classic setting of the first Remez algorithm $x_k \in E(e_k, B)$ must be determined, this relaxed algorithm only requires the evaluation of e_k on B_{k+1} at each iteration. Provided (A1) and (A2) hold, we know [18, Thm. 1] that (a) both $P[B]$ and $P[D_k], k \geqslant 0$ possess solutions $\widehat{\mathbf{a}}, \widehat{\mathbf{a}}_k \in A$, (b) $\mu(D_k) \leqslant \mu(D_{k+1}) \leqslant \mu(B)$ and $\lim_{k\to\infty} \mu(D_k) = \mu(B)$, (c) $\{\widehat{\mathbf{a}}_k\}_{k\geqslant 0}$ has at least one accumulation point in A that solves $P[B]$ and (d) if $\widehat{\mathbf{a}} \in A$ is the unique solution of $P[B]$, then $\lim_{k\to\infty} \|\widehat{\mathbf{a}} - \widehat{\mathbf{a}}_k\|_X = 0$. This result is also true if **Step 3** is replaced with the established version

Step 3'. Compute $x_k \in E(e_k, B)$ and a set $D_{k+1} \subseteq B$ with $D_{k+1} \supseteq D_k \cup \{x_k\}$.

and without requiring (A1).

For linear operators T, it can be shown that (A2) corresponds to the classic condition of the first Remez algorithm that the $M \times N$ system matrix $[\phi_j(x_i)]_{i=1,\ldots,M, j=1,\ldots,N}$ has rank N, where B_0 consists of the $M \geqslant N$ pairwise distinct points $\{x_k\}_{k=1}^{M}$. The monomial basis of $\mathbb{R}_{N-1}[X]$ satisfies this condition[1] for any $B_0 \subseteq B$ with $M \geqslant N$.

In the rational T case, (A2) is not necessarily true for any $B_0 \subseteq B$, even in the case where $P[B]$ has a unique solution. This can be restrictive. Nevertheless, the above algorithm can converge even when (A2) is not satisfied. In some cases, it is possible to ensure convergence if certain assumptions regarding $T(A)$ hold [18, Thm. 2]:

(A3) $P[B]$ has a solution $\widehat{\mathbf{a}} \in A$;
(A4) There exists $\varepsilon > 0$ such that the set $\Lambda = \{T\mathbf{a}, \mathbf{a} \in A$ and $\|T\widehat{\mathbf{a}} - T\mathbf{a}\|_B \leqslant \varepsilon\}$ is compact;
(A5) There is a $\delta > 0$ such that for every finite set $D \subseteq B$ with $h(D, B) \leqslant \delta$, $P[D]$ has a solution $\mathbf{a}_D \in A$ and $T\mathbf{a}_D \in \Lambda$. In particular, we assume that $h(B_0, B) \leqslant \delta$.

Examples of nonlinear $T(A)$ families that satisfy these properties are derived in [25] for the case where $B = [0, 1]$.

III. A Generalized First Remez Algorithm Tailored for Rational Approximation Problems

A particular problem in practice can be the non-existence of a best approximation (*i.e.,* (A3) does not hold). One simple example, taken from [26], is that of approximating $f(x) = x$ over $[0, 1]$ by the family $p_1 x/(q_1 + q_2 x)$. To avoid it, we consider approximation spaces $T(A)$ such as

$$\mathcal{R}_L(B) = \left\{ \frac{P}{Q} := \frac{p_1\phi_1 + \ldots + p_m\phi_m}{q_1\psi_1 + \ldots + q_n\psi_n} \,\middle|\, \begin{array}{l} Q \geqslant L > 0 \text{ on } B, \\ \max\limits_{1\leqslant i\leqslant n} |q_i| \leqslant 1. \end{array} \right\},$$

where $L \in C(B)$ is strictly positive over B. Besides ensuring existence of a best approximation [27, Thm. 1], L also limits the magnitude in the denominator from becoming too small and, together with the normalizing condition $\max_{1\leqslant i\leqslant n} |q_i| \leqslant 1$, Q from having a large dynamic range over B, which otherwise might incur numerical issues (at both the approximation and evaluation levels). For L sufficiently small over B, the result will be practically indistinguishable from the case with positive denominator $Q > 0$. In this work, we take L to be a constant function, namely 10^{-20}.

[1]The notion attached to this condition is that of a Haar system [24, Ch. 3.3].

The solution to $P[D]$ in this setting (with D a finite set like in **Step 2** of Algorithm 1), can be determined using a version of the so-called *differential correction* (DC) algorithm. It consists of a sequence of linear programming (LP) calls [27, Sec. 4], as shown in Algorithm 2. Convergence is guaranteed and is usually quadratic [27, Thm. 9]. Stopping in practice occurs when the relative change in Δ_k is sufficiently small (*e.g.* in our case when $(\Delta_k - \Delta_{k+1})/\Delta_k < 10^{-16}$).

Algorithm 2 Differential Correction (DC) [27]

```
Step 1. $k \leftarrow 0$ and choose $P_0/Q_0 \in \mathcal{R}_L(D)$.
Step 2. Given $P_k/Q_k \in \mathcal{R}_L(D)$ and $\Delta_k = \|f - P_k/Q_k\|_D$,
        find $P_{k+1} = P, Q_{k+1} = Q$ that minimizes
            $\max_{x\in D} \frac{|f(x)Q(x) - P(x)| - \Delta_k Q(x)}{Q_k(x)}$,
        with $\max_{1\leqslant i\leqslant n} |q_i| \leqslant 1$ and $Q(x) \geqslant L(x), \forall x \in D$.
Step 3. $k \leftarrow k + 1$ and go to Step 2.
```

Algorithm 3 Adaptive Differential Correction (ADC) [28]

```
Step 1. $k \leftarrow 0$, $\Delta_{-1} \leftarrow 0$, choose $S_0 \subseteq D$ s.t. $|S_0| \geqslant m+n$.
Step 2. Solve $P[S_k]$ using Algorithm 2, getting $\Delta_k = \mu(S_k)$
        and the solution $R_k \in \mathcal{R}_L(S_k)$. Set $e_k = f - R_k$.
Step 3. Compute the following sets ($T_k$ and $A_k$):
        $T_k = \{x \in S_k, |e_k(x)| \geqslant \Delta_k(1 - \varepsilon_{\text{tol}})\}$,
        $A_k \subseteq D$ by searching for local extrema of $e_k$ over
        $D$ starting from all the elements of $S_k$ s.t. $|e_k(x)| \geqslant$
        $\Delta_k(1 + \varepsilon_{\text{tol}}), \forall x \in A_k$.
Step 4. if $A_k$ is empty then terminate else $k \leftarrow k + 1$.
Step 5. if $k = 1$ then $S_1 = T_0(R_0) \cup A_0$ else
            $X_k = \begin{cases} S_k, & \Delta_k \leqslant \max(\Delta_{k-1}, \Delta_{k-2})(1 + \varepsilon_{\text{tol}}) \\ T_k, & \text{otherwise} \end{cases}$
            $S_{k+1} = \begin{cases} X_k \cup A_k, & \Delta_k \geqslant \Delta_{k-1} \\ X_k \cup A_k \cup S_{k-1}, & \text{otherwise} \end{cases}$
            $k \leftarrow k + 1$ and go to Step 2.
```

If the size of D is too big, then the LP calls cost during the DC algorithm can be quite large. We can however reduce runtime drastically in this case. The idea [28], [29] is, as part of an iterative procedure, to construct a small *active* subset $S \subseteq D$ with $|S| = O(m + n)$ and apply DC on it. Based on the obtained result, either S gets updated (see Algorithm 3) for a new iteration or we stop if the solution over S is sufficiently close to the one over D (controlled with $\varepsilon_{\text{tol}} > 0$, which we take in practice as 10^{-8}). This *adaptive* procedure generally executes much faster than applying DC over D when $|D| \gg |S|$ (see [28, Sec. 3] for convergence results).

To derive an $\mathcal{R}_L(B)$ solution to $P[B]$, we have devised the following variant of Algorithm 1 (see Algorithm 4). At each inner iteration of the DC algorithm during the overall procedure (in **Step 2**), the problem is solved only on a small subset of the current D_k. Taking D_0 to be sufficiently dense in B (we start with $100(m+n)$ scaled Chebyshev nodes [30, Ch. 2]) usually leads to fast convergence in practice (2 to 5 iterations) and reasonable runtime, ranging from a couple of seconds to several minutes on complicated functions. The stopping criterion used in practice is similar to the one for Algorithm 2: when $(|e_k(x^*)| - \mu(D_k))/|e_k(x^*)| < 10^{-4}$, where $x^* = \arg\max_{x \in E_k} |e_k(x)|$.

Algorithm 4 Generalized Rational Approximation Algorithm

```
Step 1. $k \leftarrow 0$ and $D_0 \subseteq B$ a finite set with $|D_0| \geqslant m+n$.
        Initialize active subset $S \subseteq D_0$ s.t. $|S| = m+n$.
Step 2. Solve $P[D_k]$ using Algorithm 3, with solution $R_k \in \mathcal{R}_L(D_k)$ and let $S \subseteq D_k$ now be the final active
        subset. Set $e_k = f - R_k$.
Step 3. Compute $E_k = \{x \in B, x$ local extrema of $e_k$ over $B$ with $|e_k(x)| > \mu(D_k)\}$ and let $D_{k+1} = D_k \cup E_k$.
Step 4. $k \leftarrow k+1$ and go to Step 2 with starting active
        subset $S$.
```

Compared to Algorithm 1, at each iteration k we take $B_k = B$ and do not consider successively finer discretizations of B. This is due to the fact that we can determine accurate approximations of E_k in **Step 3** quite fast using Chebyshev-proxy rootfinding methods (see [30, Ch. 18] and [23, Sec. 6]). Using discretizations B_k is nevertheless important in multivariate approximation contexts, where finding all local extrema over B is usually more challenging.

Algorithm 4 converges under the same conditions as Algorithm 1. While not true for any approximation space $T(A)$, due to the restricted denominator condition we are imposing (which ensures existence of a best approximation), we have not had any convergence issues with the choice of basis functions $\{\phi_i\}_{i=1}^m$ and $\{\psi_i\}_{i=1}^n$ that occur in practice.

IV. Rational approximations with machine-number coefficients

The $\mathcal{R}_L(B)$ solution $r(x) = \frac{\sum_{i=1}^m p_i \phi_i(x)}{\sum_{i=1}^n q_i \psi_i(x)}$ to $P[B]$ has real-valued coefficients, but the evaluation in a libm context will require FP coefficients. Obtaining these values in a naive way, for instance through rounding, can degrade the approximation error significantly. To obtain a better approximation in this setting, we adapt the method presented in [7], [10], [11] based on central algorithms in the study of Euclidean lattices. We review it in Sec. IV-A and discuss the challenges of applying it in the nonlinear setting. We then propose changes needed to address them in Sec. IV-B and Sec. IV-C.

A. A Closest Vector Problem formulation

Up to multiplying both P and Q by a constant such that $\max_{1 \leqslant i \leqslant n} |q_i| = 1$ and reordering $\{\psi_i\}_{i=1}^n$ such that $q_1 = 1$, we want to find

$$\widehat{r}(x) = \frac{\sum_{i=1}^m \widehat{p}_i \phi_i(x)}{\psi_1(x) + \sum_{i=2}^n \widehat{q}_i \psi_i(x)} \tag{1}$$

with coefficients $\{\widehat{p}_i\}_{i=1}^m$ and $\{\widehat{q}_i\}_{i=2}^n$ in some desired FP format(s) that minimizes $\|f - \widehat{r}\|_B$. Actually minimizing this error is difficult even in the polynomial case except for small degrees [8], which is why we use a fast heuristic that will hopefully still give a good result.

Each coefficient will be of the form $I 2^e$, with bounded $I, e \in \mathbb{Z}$ (and bounds defined by the FP format the coefficient is stored in). Following [7, Sec. 3], we tentatively set the exponent e to coincide with the exponent of the corresponding coefficient of r and update it if necessary later.

Equipped with a set of integer exponents $\{u_i\}_{i=1}^m$ and $\{v_i\}_{i=2}^n$, we now need to determine $m+n-1$ integers $a_i (= \widehat{p}_i 2^{-u_i})$ and $b_i (= \widehat{q}_i 2^{-v_i})$ such that

$$\widehat{r}(x) = \frac{\sum_{i=1}^m a_i \widehat{\phi}_i(x)}{\psi_1(x) + \sum_{i=2}^n b_i \widehat{\psi}_i(x)},$$

with $\widehat{\phi}_i(x) = 2^{-u_i}\phi_i(x)$ and $\widehat{\psi}_i(x) = 2^{-v_i}\psi_i(x)$, is a good approximation to r (and f), *i.e.*, $\|r - \widehat{r}\|_B$ is small.

To address this new problem, we discretize and linearize it as follows: (a) choose $N_r \geqslant m+n-1$ distinct points $\{x_k\}_{k=1}^{N_r}$ from B and (b) find integers a_i and b_i such that

$$\sum_{i=1}^m a_i \underbrace{\begin{bmatrix} \widehat{\phi}_i(x_1) \\ \widehat{\phi}_i(x_2) \\ \vdots \\ \widehat{\phi}_i(x_{N_r}) \end{bmatrix}}_{\boldsymbol{\alpha}_i} + \sum_{i=2}^n b_i \underbrace{\begin{bmatrix} -r(x_1)\widehat{\psi}_i(x_1) \\ -r(x_2)\widehat{\psi}_i(x_2) \\ \vdots \\ -r(x_{N_r})\widehat{\psi}_i(x_{N_r}) \end{bmatrix}}_{\boldsymbol{\beta}_i}$$

and

$$\boldsymbol{r} = \begin{bmatrix} r(x_1)\psi_1(x_1) & r(x_2)\psi_1(x_2) & \cdots & r(x_{N_r})\psi_1(x_{N_r}) \end{bmatrix}^T$$

are as close as possible to one another with respect to the $\|\cdot\|_\infty$ norm ($\|\mathbf{x}\|_\infty = \max_{1 \leqslant i \leqslant N_r} |x_i|$), *i.e.*,

$$\left\| \sum_{i=1}^m a_i \boldsymbol{\alpha}_i + \sum_{i=2}^n b_i \boldsymbol{\beta}_i - \boldsymbol{r} \right\|_\infty \tag{2}$$

is minimized. The solution is hopefully also good over B.

Solving (2) exactly, which is a type of Closest Vector Problem (CVP) in the theory of Euclidean lattices, CVP$_\infty$, is in general hard from a complexity-theoretic point of view. Both it and its ℓ_2 equivalent, CVP$_2$, are known to be NP-hard. Fast approximate algorithms exist for CVP$_2$. While they have an exponential worst case approximation factor, when applied to lattices appearing in function approximation (especially using polynomials) they tend to be extremely well behaved. The overall use of these algorithms is identical in the rational case and we point the reader to [7], [31] for details. Concretely, we search for integers a_i and b_i that minimize the quantity

$$\left\| \sum_{i=1}^m a_i \boldsymbol{\alpha}_i + \sum_{i=2}^n b_i \boldsymbol{\beta}_i - \boldsymbol{r} \right\|_2. \tag{3}$$

B. Discretizing the approximation domain

How to choose the N_r points is also a nontrivial matter. In the polynomial case, the suggestion [7, Sec. 3.1] is to take N_r close to N (an interpolation-like problem), either the points where the error $f - r$ vanishes or Chebyshev nodes. Since in $\mathbb{R}^{N_r}$, $\|\cdot\|_\infty \leqslant \|\cdot\|_2 \leqslant \sqrt{N_r}\,\|\cdot\|_\infty$, the larger N_r is, the more the ℓ_2 solution of (3) can differ from the ℓ_∞ and L^∞ ones. On the other hand, for nonlinear r (or $\widehat{r}$), we have no guarantee that the error will vanish at N points. Even if that happens, there is always the risk that *spurious poles* appear inside B. These can be both mathematical (discontinuous dependence of parameters on the data in some cases) and numerical. They frequently come with a corresponding close by zero and are known as *Froissart doublets*.

A denser grid of points tends to alleviate this problem to a considerable extent when dealing with real coefficient solutions of (3) (see [30, Ch. 26] for a discussion regarding this). We apply a similar approach in the machine-coefficient setting, taking around $10N$ points in B, with good choices being Chebyshev nodes or points that follow the distribution of the zeros of $f-r$. It nevertheless remains an open question how to best discretize B and guarantee pole-free machine-coefficient L^∞ solutions.

C. Searching for an optimized normalization factor

One may also wonder about the normalization choice in (1). While it has no impact on the approximation error of a real coefficient solution r, it does change the approximation error of $\widehat{r}$ and can potentially impact the evaluation error as well. We introduce an optional sampling-based heuristic to optimize the choice of the normalization factor. It consists of:

(a) scale (1) such that the coefficient of ψ_1 takes machine-representable values between $[1, 2)$ (*e.g.* 128 equidistant points starting at 1);
(b) solve the CVP$_2$ problem (3) in each case;
(c) take the approximation with smallest $\|f - \widehat{r}\|_B$.

This can significantly increase runtime, and in the future we plan to investigate optimizations to this part of the process.

V. Examples

Two examples highlight the behaviour of our algorithms in practical situations. The first one comes from the CORE-MATH project [32], where our approach has already been used to produce the implementation of the `binary32` version of the arctan function, showcasing excellent results at both the approximation and evaluation steps. The second example is on the inverse Langevin function, which has uses in fields such as polymer science, magnetism and biomechanics. While rational approximations are suitable here and our algorithms are able to produce quality approximations, the use of monomial bases is highly ill-conditioned, significantly degrading evaluation error. Evaluations for both examples are performed in `binary64` arithmetic, without the use of fused multiply-add operations. Since we are targetting errors below machine precision, we had to use multiprecision arithmetic (200 bits here) in the computations done in Algorithm 4 and the lattice-based approach of Sec. IV.

Our method is implemented in C/C++ as an open source project[2], having been tested on Linux and MacOS-based systems. It can be used as a library, but we also provide an executable that parses command line arguments to compute generalized rational approximations.

[2]See https://gitlab.inria.fr/sfilip/rminimax for details.

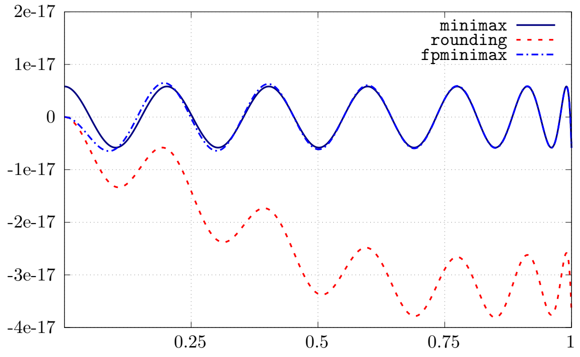

Fig. 1. Relative errors between $\arctan(x)$ and approximations of the form $r(x) = \sum_{i=1}^{7} p_i x^{2i-1} / \sum_{i=1}^{7} q_i x^{2i-2}$ on $[0.00012207, 1]$.

A. An example from the CORE-MATH project

The CORE-MATH project intends to provide a collection of mathematical functions implementations with correct rounding (*i.e.,* given a function f and a floating-point input x, the correct rounding of $f(x)$ is the floating-point value y that is closest to $f(x)$ according to the rounding mode in use) in both `binary32` and `binary64` contexts.

The `binary32` arctangent function (`atanf`) in CORE-MATH is evaluated using a rational approximation for inputs x with $|x| \in B = [2^{-13}, 1]$. Since $f(x) = \arctan(x)$ is an odd function, we consider rational approximations of the form

$$r(x) := \frac{\sum_{i=1}^{m} p_i \phi_i(x)}{\sum_{i=1}^{n} q_i \psi_i(x)} = \frac{\sum_{i=1}^{m} p_i x^{2i-1} / f(x)}{\sum_{i=1}^{n} q_i x^{2i-2}}$$

such that $\|1 - r(x)\|_B$ is minimized. Using our approach from Sec. III on a slightly larger domain $B = [0.000127, 1]$ and $m = n = 7$, we obtain an approximation with error $2^{-57.26}$ (the `minimax` error curve in Fig. 1). Taking the rounded version of this approximation with `binary64` coefficients gives a $2^{-54.54}$ error (the `rounding` error curve in Fig. 1), which proves to be insufficient for a correct rounding implementation. The `fpminimax`-like approach from Sec. IV with the default normalization choice gives a much better $2^{-57.09}$ error (the `fpminimax` error curve in Fig. 1).

This is still not sufficient to ensure correct rounding in all situations however (one hard to round case remains for $|x| =$`0x1.1ad646p-4` and rounding to nearest), but slightly reducing the value of q_0 eliminates the issue. Performing a normalization search as presented at the end of Sec. IV also eliminates the problem, resulting in a smaller $2^{-57.10}$ error.

A polynomial approximation ($n = 1$) with similar error requires at least degree 20 ($m = 21$), resulting in a possible tradeoff of 6 additions and 6 multiplications for one division.

B. Rational approximations to the inverse Langevin function

The Langevin function is defined by

$$y = \mathcal{L}(y) = \coth(y) - 1/y$$

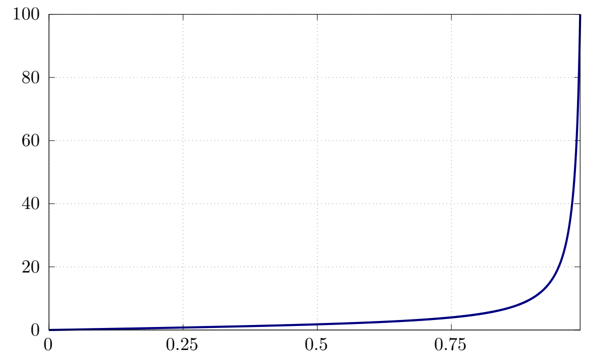

Fig. 2. Plot of the inverse Langevin function $\mathcal{L}^{-1}(x)$ between $[0, 0.99]$. It has two simple poles at $x = \pm 1$, each with residue -1.

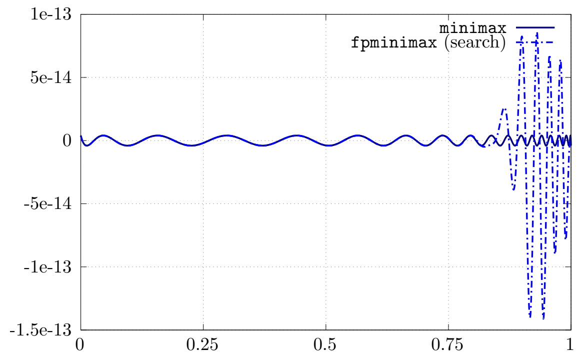

Fig. 3. Relative errors between the inverse Langevin function and approximations of the form $\mathcal{R}_{17,17}$ on the interval $[0, 1)$.

for $y \in \mathbb{R}$ and has a removable singularity at $y = 0$. Its inverse is defined by $y = \mathcal{L}^{-1}(x)$ for $x \in (-1, 1)$. Both of these are odd functions and can thus be studied in the restricted case where x and y are positive. While the Langevin function is well-behaved and can be easily handled both numerically and algebraically, $\mathcal{L}^{-1}$ does not have a similar closed-form expression in terms of elementary functions. It has simple pole singularities at $x = \pm 1$ (see Fig. 2) and an infinite number of complex singularities with moduli tending to 1 (see [33, Sec. 6]), making it a challenging candidate for numerical computation.

There have been several attempts in recent years to provide reliable approximations to this function using various techniques with different target accuracies. For example, [34] presents a method for computing the Taylor series of $\mathcal{L}^{-1}$ up to 500 terms, which has the form

$$\mathcal{L}^{-1}(x) = 3x + \frac{9}{5}x^3 + \frac{297}{175}x^5 + \frac{1539}{875}x^7 + \cdots$$

and radius of convergence $0.904643679\ldots$, making it unusable close to 1. Rational functions are the most widely used form of approximating $\mathcal{L}^{-1}$, with several authors introducing Padé-based [35], [36], multipoint Padé [37], and L^∞-approximations [38]. The relative error in these aforementioned works never descends below 10^{-6}.

Nevertheless, having fast access to a highly accurate (close to machine precision) approximation is desirable in practice [39]. There seems to be little work in this direction. Piece-wise cubic splines at 10000 points in [40] enable a maximum relative error of around 10^{-11}, but require 320KB for coefficient storage which could make it difficult to store them all in L1 level cache on many processors. A different approach is taken in [41], which (iteratively) uses linearization and error approximation to generate high accuracy proxys of $\mathcal{L}^{-1}$ with stated relative errors that go as low as 10^{-46}, but computational complexity (in terms of number of basic operations) and IEEE-754 finite precision arithmetic impact on evaluation error are not explored.

To compute high accuracy rational approximations of $\mathcal{L}^{-1}$, we take advantage of its asymptotic behavior and Taylor expansion to remove the pole at $x = 1$ and focus on

$$f(x) = \begin{cases} \dfrac{\mathcal{L}^{-1}(x)(1-x)}{x}, & x \in (0,1), \\ 3, & x = 0, \\ 1, & x = 1, \end{cases}$$

using rational approximations from the sets

$$\mathcal{R}_{m,n} = \left\{ r(x) := \frac{\sum_{i=1}^{m+1} p_i x^{i-1}}{\sum_{i=1}^{n+1} q_i x^{i-1}} \right\}$$

and

$$\mathcal{J}_{m,n} = \left\{ r(x) := \frac{\sum_{i=1}^{m+1} p_i x^{2i-2}}{\sum_{i=1}^{n+1} q_i x^{i-1}} \right\}$$

such that $\|r/f - 1\|_{[0,1]}$ is minimized. This translates back to L^∞-approximations to $\mathcal{L}^{-1}$ using rational functions of the form $xr(x)/(1-x)$.

The set $\mathcal{R}_{m,n}$ corresponds to ordinary rational functions. This proves to be a highly ill-conditioned choice. Using the algorithm of Sec. III, a type $(m, n) = (17, 17)$ approximation is required in order to reach an error level of $4.0 \cdot 10^{-15}$ (the `minimax` error curve in Fig. 3). By contrast, a polynomial approximation of degree 70 or more is required to reach a similar target error. Rounding the coefficients to `binary64` values leads to an unusable approximation with $1.14 \cdot 10^{-2}$ relative error. This sensitivity also manifests itself in the `fpminimax`-like approach, with the default result having two spurious poles close to $x = 0.9$. Normalization search removes them, but the final error is on the order of $1.15 \cdot 10^{-13}$ (the `fpminimax` error curve in Fig. 3). Evaluating this last approximation using Horner's rule in `binary64` arithmetic is still ill-conditioned, especially in the hard to evaluate region $[0.8, 1)$ where the relative error jumps to at least $1.5 \cdot 10^{-3}$.

The numerator basis in $\mathcal{J}_{m,n}$ was chosen because it leads to better conditioned approximations. Taking again $(m, n) = (17, 17)$ we obtain a real coefficient L^∞-approximation with error $3.5 \cdot 10^{-14}$ (the `minimax` curve in Fig. 4). If the coefficients are rounded to `binary64` values, the error drops to $1.07 \cdot 10^{-9}$, whereas the `fpminimax` approach greatly reduces the error to $4.05 \cdot 10^{-14}$. Normalization search further

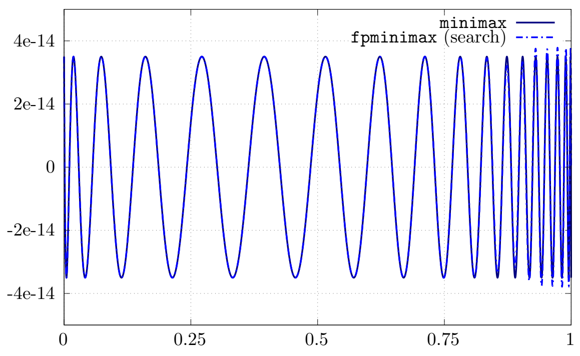

Fig. 4. Relative errors between the inverse Langevin function and approximations of the form $\mathcal{J}_{17,17}$ on the interval $[0, 1)$.

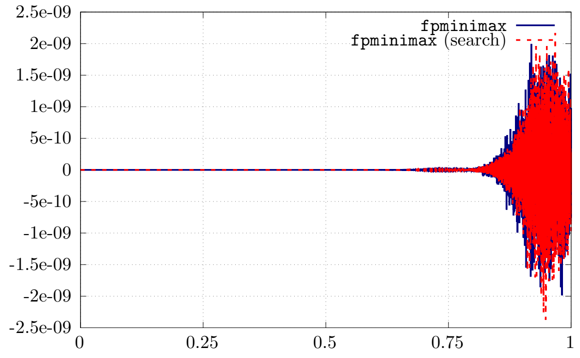

Fig. 5. Relative errors between the inverse Langevin function and `binary64` evaluations of $\mathcal{J}_{17,17}$ approximations using Horner's rule. The evaluations are done at 20000 equispaced points inside $[0, 1)$.

improves this to $3.81 \cdot 10^{-14}$ (see `fpminimax` error curve in Fig. 4). The evaluation error (see Fig. 5), while better than in the $\mathcal{R}_{17,17}$ setting, is still quite large on $[0.8, 1)$.

To improve evaluation error in this setting, one can use extended precision arithmetic (using `extended double` formats, or double-`double` and triple-`double` formats consisting of unevaluated sums of two or three regular `binary64` values). Another alternative is to use a better conditioned representation. For instance, barycentric forms have been shown to be a robust and well-conditioned choice for evaluating rational functions (see for instance [15], [42]), but they come at the cost of a large number of divisions, which might be unacceptable in an optimized implementation. Factoring the numerator and denominator into eight degree 2 and one degree 1 irreducible components each, and rounding their coefficients to `binary64` is another option that is better conditioned and faster (but still at the cost of more floating-point multiplications with respect to the monomial representation). This is shown in Fig. 6, where the quality in evaluation error seems greatly improved, especially for the $\mathcal{J}_{17,17}$ approximation.

Interval subdivision with polynomial approximations is another alternative that might be worth considering, but the appeal of global rational approximations such as the ones produced here is that they are relatively compact, fast, and

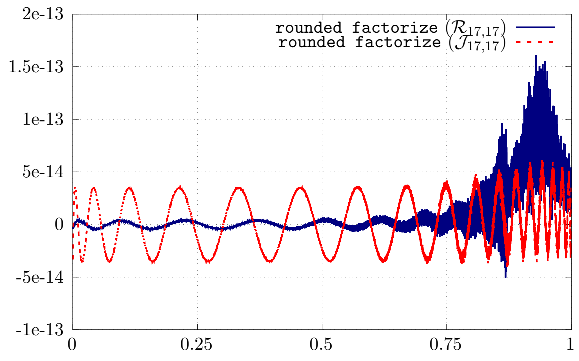

Fig. 6. Relative errors between the inverse Langevin function and `binary64` evaluations for factorizations of $\mathcal{R}_{17,17}$ and $\mathcal{J}_{17,17}$. The evaluations are done at 20000 equispaced points inside $[0, 1)$.

straightforward to vectorize.

VI. Conclusion and Future Work

We have introduced two new algorithms for computing efficient generalized rational approximations with real and machine-number coefficients (such as floating-point values), respectively. The real-coefficient method builds on several results [18], [28] that have been proposed in the approximation theory literature over the years, whereas the machine-coefficient method extends a Euclidean lattice basis reduction approach that has been succesfully used in polynomial approximation [7]. A robust open-source C++ implementation makes it easy to experiment with these algorithms.

While not shown here, both methods can be adapted to incorporate additional constraints such as fixed coefficient values for some of the terms in the numerator and denominator.

If the resulting real-coefficient rational approximation is not too ill-conditioned (*e.g.* highly sensitive to perturbations of the coefficient values), the generated machine number coefficient approximation tends to be almost as accurate, with a good evaluation error as well (see the example in Sec. V-A).

In case of extremely poor conditioning, spurious poles can sometimes appear and for the moment we have no guaranteed approach to avoid them in the Euclidean lattice-based results. Even if no spurious poles appear, the quality of the results in such cases can still be significantly degraded compared to the real-coefficient baseline.

This leads to several interesting directions for future work. The value of the coefficient normalization parameter can significantly impact accuracy. The grid-based search we proposed at the end of Sec. IV could stand to be improved using a mathematical optimization-based method. Exploring various rewritings that are better conditioned (*e.g.* like (partial) factorization in irreducible terms of both numerator and denominator or having an approximation with both a polynomial and rational part) is also relevant, but optimizing their coefficients seems to require different tools from the ones used here, possibly stemming from the nonlinear mixed integer optimization literature. Finally, exploring a way to generate rational approximations whose machine number coefficients are jointly optimized for minimizing the sum of evaluation and approximation errors (similar to [43] in the polynomial case) could lead to even better results, but looks complicated due to the nonlinear nature of the rational approximation problem.

In theory, the algorithmic ideas discussed in this paper are not restricted to univariate problems. Multivariate rational approximations seem to be good surrogate models for high-energy physics applications [44] and it would be interesting to investigate if and how the methods introduced here extend in practice to such problems.

References

[1] W. J. Cody, "Algorithm 715: SPECFUN – A Portable FORTRAN Package Of Special Function Routines And Test Drivers," *ACM TOMS*, vol. 19, no. 1, pp. 22–30, 1993.

[2] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[3] S. Chevillard, M. Joldeş, and C. Lauter, "Sollya: An Environment for the Development of Numerical Codes," in *Mathematical Software - ICMS 2010*, ser. Lecture Notes in Computer Science, K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, Eds., vol. 6327. Heidelberg, Germany: Springer, September 2010, pp. 28–31.

[4] M. Daumas and G. Melquiond, "Certification of Bounds on Expressions Involving Rounded Operators," *ACM TOMS*, vol. 37, no. 1, pp. 1–20, 2010.

[5] V. Magron, A. Rocca, and T. Dang, "Certified Roundoff Error Bounds Using Bernstein Expansions and Sparse Krivine-Stengle Representations," *IEEE Trans. on Computers*, vol. 68, no. 7, pp. 953–966, 2018.

[6] J.-M. Muller, *Elementary Functions*, 3rd ed. Springer, 2016.

[7] N. Brisebarre and S. Chevillard, "Efficient Polynomial L^∞-Approximations," in *18th IEEE Symposium on Computer Arithmetic (ARITH'07)*. IEEE, 2007, pp. 169–176.

[8] N. Brisebarre, J.-M. Muller, and A. Tisserand, "Computing Machine-Efficient Polynomial Approximations," *ACM TOMS*, vol. 32, no. 2, pp. 236–256, 2006.

[9] J. P. Lim, M. Aanjaneya, J. Gustafson, and S. Nagarakatte, "An Approach to Generate Correctly Rounded Math Libraries for New Floating Point Variants," *Proceedings of the ACM on Programming Languages*, vol. 5, no. POPL, 2021.

[10] N. Brisebarre, S. Chevillard, M. D. Ercegovac, J.-M. Muller, and S. Torres, "An Efficient Method For Evaluating Polynomial And Rational Function Approximations," in *2008 Int. Conf. on Application-Specific Systems, Architectures and Processors*. IEEE, 2008, pp. 233–238.

[11] N. Brisebarre, G. Constantinides, M. Ercegovac, S.-I. Filip, M. Istoan, and J.-M. Muller, "A High Throughput Polynomial And Rational Function Approximations Evaluator," in *2018 IEEE 25th Symposium on Computer Arithmetic*. IEEE, 2018, pp. 99–106.

[12] S. Chevillard, "Évaluation efficace de fonctions numériques – outils et exemples," Ph.D. dissertation, École Normale Supérieure de Lyon, 2009.

[13] L. Veidinger, "On the Numerical Determination of the Best Approximations in the Chebyshev Sense," *Numer. Math.*, vol. 2, no. 1, pp. 99–105, 1960.

[14] A. Ralston, "Some Aspects of Degeneracy in Rational Approximations," *IMA J. Numer. Anal.*, vol. 11, no. 2, pp. 157–170, 1973.

[15] S.-I. Filip, Y. Nakatsukasa, L. N. Trefethen, and B. Beckermann, "Rational Minimax Approximation via Adaptive Barycentric Representations," *SIAM J. on Sci. Comp.*, vol. 40, no. 4, pp. A2427–A2455, 2018.

[16] O. Kupriianova and C. Lauter, "Metalibm: A Mathematical Functions Code Generator," in *Mathematical Software–ICMS 2014: 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings 4*. Springer, 2014, pp. 713–717.

[17] C. Lauter and M. Mezzarobba, "Semi-Automatic Floating-Point Implementation of Special Functions," in *2015 IEEE 22nd Symposium on Computer Arithmetic*. IEEE, 2015, pp. 58–65.

[18] R. Reemtsen, "Modifications of the First Remez Algorithm," *SIAM J. Numer. Anal.*, vol. 27, no. 2, pp. 507–518, 1990.

[19] ——, "On Discretization Errors In Nonlinear Approximation Problems," *J. Approx. Theory*, vol. 49, no. 3, pp. 256–273, 1987.

[20] E. W. Cheney, *Introduction to Approximation Theory*. AMS Chelsea Pub., 1982.

[21] A. Curtis and M. R. Osborne, "The Construction of Minimax Rational Approximations to Functions," *Comput. J.*, vol. 9, no. 3, pp. 286–293, 1966.

[22] R. Pachón and L. N. Trefethen, "Barycentric-Remez algorithms for best polynomial approximation in the chebfun system," *BIT*, vol. 49, pp. 721–741, 2009.

[23] S.-I. Filip, "A Robust and Scalable Implementation of the Parks-McClellan Algorithm for Designing FIR Filters," *ACM TOMS*, vol. 43, no. 1, pp. 1–24, 2016.

[24] R. A. DeVore and G. G. Lorentz, *Constructive Approximation*. Springer Science & Business Media, 1993, vol. 303.

[25] M. E. Burke, "Nonlinear Best Approximations on Discrete Sets," *J. Approx. Theory*, vol. 16, no. 2, pp. 133 – 141, 1976.

[26] H. L. Loeb, "Approximation by Generalized Rationals," *SIAM J. Numer. Anal.*, vol. 3, no. 1, pp. 34–55, 1966.

[27] E. H. Kaufman Jr. and G. D. Taylor, "Uniform Approximation by Rational Functions Having Restricted Denominators," *J. Approx. Theory*, vol. 32, no. 1, pp. 9–26, 1981.

[28] E. H. Kaufman Jr., S. F. McCormick, and G. D. Taylor, "An Adaptive Differential-Correction Algorithm," *J. Approx. Theory*, vol. 37, no. 3, pp. 197–211, 1983.

[29] ——, "Uniform Rational Approximation on Large Data Sets," *Int. J. for Numer. Methods in Engineering*, vol. 18, no. 10, pp. 1569–1575, 1982.

[30] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[31] N. Brisebarre, S.-I. Filip, and G. Hanrot, "A Lattice Basis Reduction Approach for the Design of Finite Wordlength FIR Filters," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2673–2684, 2018.

[32] A. Sibidanov, P. Zimmermann, and S. Glondu, "The CORE-MATH Project," in *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*. IEEE, 2022, pp. 26–34.

[33] S. R. Rickaby and N. H. Scott, "On the complex singularities of the inverse Langevin function," *IMA J. Numer. Anal.*, vol. 83, no. 6, pp. 1092–1116, 2018.

[34] M. Itskov, R. Dargazany, and K. Hörnes, "Taylor expansion of the inverse function with application to the Langevin function," *Math. Mech. Solids*, vol. 17, no. 7, pp. 693–701, 2012.

[35] A. Cohen, "A Padé approximant to the inverse Langevin function," *Rheol. Acta*, vol. 30, pp. 270–273, 1991.

[36] B. C. Marchi and E. M. Arruda, "Generalized error-minimizing, rational inverse Langevin approximations," *Math. Mech. Solids*, vol. 24, no. 6, pp. 1630–1647, 2019.

[37] R. Jedynak, "Approximation of the inverse Langevin function revisited," *Rheol. Acta*, vol. 54, no. 1, pp. 29–39, 2015.

[38] ——, "A comprehensive study of the mathematical methods used to approximate the inverse Langevin function," *Math. Mech. Solids*, vol. 24, no. 7, pp. 1992–2016, 2019.

[39] A. Ammar, "Effect of the inverse Langevin approximation on the solution of the Fokker–Planck equation of non-linear dilute polymer," *J. Non-Newton. Fluid Mech.*, vol. 231, pp. 1–5, 2016.

[40] J. M. Benítez and F. J. Montáns, "A simple and efficient numerical procedure to compute the inverse Langevin function with high accuracy," *J. Non-Newton. Fluid Mech.*, vol. 261, pp. 153–163, 2018.

[41] R. M. Howard, "Analytical approximations for the inverse Langevin function via linearization, error approximation, and iteration," *Rheol. Acta*, vol. 59, no. 8, pp. 521–544, 2020.

[42] Y. Nakatsukasa, O. Sète, and L. N. Trefethen, "The AAA algorithm for rational approximation," *SIAM J. Sci. Comput.*, vol. 40, no. 3, pp. A1494–A1522, 2018.

[43] D. Arzelier, F. Bréhard, and M. Joldeş, "Exchange Algorithm for Evaluation and Approximation Error-Optimized Polynomials," in *2019 IEEE 26th Symp. on Comp. Arith. (ARITH)*. IEEE, 2019, pp. 30–37.

[44] A. P. Austin, M. Krishnamoorthy, S. Leyffer, S. Mrenna, J. Müller, and H. Schulz, "Practical algorithms for multivariate rational approximation," *Computer Physics Communications*, vol. 261, p. 107663, 2021.