Rational Minimax Approximation via Adaptive Barycentric Representations

Silviu-Ioan Filip, Yuji Nakatsukasa, Lloyd Nicholas Trefethen, Bernhard Beckermann

RATIONAL MINIMAX APPROXIMATION VIA ADAPTIVE BARYCENTRIC REPRESENTATIONS

SILVIU-IOAN FILIP*, YUJI NAKATSUKASA†, LLOYD N. TREFETHEN‡, AND BERNHARD BECKERMANN§

Abstract. Computing rational minimax approximations can be very challenging when there are singularities on or near the interval of approximation — precisely the case where rational functions outperform polynomials by a landslide. We show that far more robust algorithms than previously available can be developed by making use of rational barycentric representations whose support points are chosen in an adaptive fashion as the approximant is computed. Three variants of this barycentric strategy are all shown to be powerful: (1) a classical Remez algorithm, (2) a "AAA-Lawson" method of iteratively reweighted least-squares, and (3) a differential correction algorithm. Our preferred combination, implemented in the Chebfun MINIMAX code, is to use (2) in an initial phase and then switch to (1) for generically quadratic convergence. By such methods we can calculate approximations up to type $(80, 80)$ of $|x|$ on $[-1, 1]$ in standard 16-digit floating point arithmetic, a problem for which Varga, Ruttan, and Carpenter required 200-digit extended precision.

Key words. barycentric formula, rational minimax approximation, Remez algorithm, differential correction algorithm, AAA algorithm, Lawson algorithm

AMS subject classifications. 41A20, 65D15

1. Introduction. The problem we are interested in is that of approximating functions $f \in \mathcal{C}([a, b])$ using type (m, n) rational approximations with real coefficients, in the L^∞ setting. The set of feasible approximations is

$$\mathcal{R}_{m,n} = \left\{ \frac{p}{q} : \ p \in \mathbb{R}_m[x], \ q \in \mathbb{R}_n[x] \right\}. \tag{1.1}$$

Given f and prescribed nonnegative integers m, n, the goal is to compute

$$\min_{r \in \mathcal{R}_{m,n}} \|f - r\|_\infty, \tag{1.2}$$

where $\|\cdot\|_\infty$ denotes the infinity norm over $[a, b]$, i.e., $\|f - r\|_\infty = \max_{x\in[a,b]} |f(x) - r(x)|$. The minimizer of (1.2) is known to exist and to be unique [58, Ch. 24].

Let the *minimax* (or *best*) approximation be written $r^* = p/q \in \mathcal{R}_{m,n}$, where p and q have no common factors. The number $d = \min\{m - \deg p, m - \deg q\}$ is called the *defect* of r^*. It is known that there exists a so-called *alternant* (or *reference*) set consisting of ordered nodes $a \leqslant x_0 < x_1 < \cdots < x_{m+n+1-d} \leqslant b$, where $f - r^*$ takes

*Univ Rennes, Inria, CNRS, IRISA, F-35000 Rennes, France (silviu.filip@inria.fr).

† National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. (nakatsukasa@nii.ac.jp)

‡ Mathematical Institute, University of Oxford, Oxford, OX2 6GG, UK (trefethen@maths.ox.ac.uk). SF and LNT were supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement 291068. The views expressed in this article are not those of the ERC or the European Commission, and the European Union is not liable for any use that may be made of the information contained here. YN was supported by Japan Society for the Promotion of Science as an Overseas Research Fellow.

§ Laboratoire Paul Painleve UMR 8524, Dept. Mathématiques, Univ. Lille, F-59655 Villeneuve d'Ascq CEDEX, France (bernhard.beckermann@univ-lille1.fr). Supported in part by the Labex CEMPI (ANR-11-LABX-0007-01).

its global extremum over $[a, b]$ with alternating signs. In other words, we have the beautiful *equioscillation* property [58, Theorem 24.1]

$$f(x_\ell) - r^*(x_\ell) = (-1)^{\ell+1}\lambda, \qquad \ell = 0, \ldots, m+n+1-d, \tag{1.3}$$

where $|\lambda| = \|f - r^*\|_\infty$. Minimax approximations with $d > 0$ are called *degenerate*, and they can cause problems for computation. Accordingly, unless otherwise stated, we make the assumption that $d = 0$ for (1.2). In practice, degeneracy most often arises due to symmetries in approximating even or odd functions, and we check for these cases explicitly to make sure they are treated properly. Other degeneracies can usually be detected by examining in succession the set of best approximations of types $(m-k, n-k), (m-k+1, n-k+1), \ldots, (m, n)$ with $k = \min\{m, n\}$ [11, p. 161].

In the approximation theory literature [11, 15, 40, 50, 63], two algorithms are usually considered for the numerical solution of (1.2), the *rational Remez* and *differential correction* (DC) algorithms. The various challenges that are inherent in rational approximations can, more often than not, make the use of such methods difficult. Finding the best polynomial approximation, by contrast, can usually be done robustly by a standard implementation of the linear version of the Remez algorithm [47]. This might explain why the current software landscape for minimax rational approximations is rather barren. Nevertheless, implementations of the rational Remez algorithm are available in some mathematical software packages: the Mathematica `MiniMaxApproximation` function, the Maple `numapprox[minimax]` routine and the MATLAB Chebfun [24] `remez` code. The Boost C++ libraries [1] also contain an implementation.

Over the years, the applications that have benefited most from minimax rational approximations come from recursive filter design in signal processing [13, 23] and the representation of special functions [18, 19]. Apart from such practical motivations, we believe it worthwhile to pursue robust numerical methods for computing these approximations because of their fundamental importance to approximation theory. A new development of this kind has already resulted from the algorithms described here: the discovery that type (k, k) rational approximations to x^n, for $n \gg k$, converge geometrically at the rate $O(9.28903\cdots^{-k})$ [44].

In this paper we present elements that greatly improve the numerical robustness of algorithms for computing best rational approximations. The key idea is the use of barycentric representations with adaptively chosen basis functions, which can overcome the numerical difficulties frequently encountered when f has nonsmooth points. For instance, when trying to approximate $f(x) = |x|$ on $[-1, 1]$ using standard IEEE double precision arithmetic in MATLAB, our barycentric Remez algorithm can compute rational approximants of type up to $(82, 82)$—higher than that obtained by Varga, Ruttan and Carpenter in [62] using 200-digit arithmetic[1].

A similar Remez iteration using the barycentric representation was described by Ioniță [35, Sec. 3.2.3] in his PhD thesis. We adopt the same set of *support points* (see Section 4.3), and our analysis justifies its choice: we prove its optimality in a certain sense. A difference from Ioniță's treatment is that we reduce the core computational task to a symmetric eigenvalue problem, rather than a generalized eigenproblem as in [35]. The bigger difference is that Ioniță treated just the core iteration for approximations of type (n, n), whereas we generalize the approach to

[1] Chebfun's previous `remez` command (until version 5.6.0 in December 2016) could only go up to type $(8, 8)$.

type (m, n) and include the initialization strategies that are crucial for making the entire procedure into a fully practical algorithm.

This work is motivated by the recent *AAA algorithm* [43] for rational approximation, which uses adaptive barycentric representations with great success. A large part of the text is focused on introducing a robust version of the rational Remez algorithm, followed by a discussion of two other methods for discrete ℓ_∞ rational approximation: the AAA-Lawson algorithm (efficient at least in the early stages, but non-robust) and the DC algorithm (robust, but not very efficient). We shall see how all three algorithms benefit from an adaptive barycentric basis. In practice, we advocate using the Remez algorithm, mainly for its convergence properties (usually quadratic [21], unlike AAA-Lawson, which converges linearly at best), practical speed (an eigenvalue-based Remez implementation is usually much faster than a linear programming-based DC method), and its ability to work with the interval $[a, b]$ directly rather than requiring a discretization (unlike both AAA-Lawson and DC). AAA-Lawson is used mainly as an efficient approach to initialize the Remez algorithm.

The paper is organized as follows. In Section 2 we review the barycentric representation for rational functions. Sections 3 to 6 are the core of the paper; here we develop the barycentric rational Remez algorithm with adaptive basis functions. Numerical experiments are presented in Section 7. We describe the AAA-Lawson algorithm in Section 8, and in Section 9 we briefly present the barycentric version of the differential correction algorithm. Section 10 presents a flow chart of `minimax` and an example of how to compute a best approximation in Chebfun.

2. Barycentric rational functions. All of our methods are made possible by a barycentric representation of r, in which both the numerator and denominator are given as partial fraction expansions. Specifically, we consider

$$r(z) = \frac{N(z)}{D(z)} = \sum_{k=0}^{n} \frac{\alpha_k}{z - t_k} \Bigg/ \sum_{k=0}^{n} \frac{\beta_k}{z - t_k}, \tag{2.1}$$

where $n \in \mathbb{N}$, $\alpha_0, \dots, \alpha_n$ and $\beta_0, \dots, \beta_n$ are sets of real coefficients and $t_0, \dots, t_n$ is a set of distinct real *support points*. The names N and D stand for "numerator" and "denominator".

If we denote by ω_t the *node polynomial* associated with $t_0, \dots, t_n$,

$$\omega_t(z) = \prod_{k=0}^{n} (z - t_k),$$

then $p(z) = \omega_t(z)N(z)$ and $q(z) = \omega_t(z)D(z)$ are both polynomials in $\mathbb{R}_n[x]$. We thus get $r(z) = p(z)/q(z)$, meaning that r is a type (n, n) rational function. (This is not necessarily sharp; r may also be of type (μ, ν) with $\mu < n$ and/or $\nu < n$.) At each point t_k with nonzero α_k or β_k, formula (2.1) is undefined, but this is a removable singularity with $\lim_{z \to t_k} r(z) = \alpha_k/\beta_k$ (or a simple pole in the case $\alpha_k \neq 0, \beta_k = 0$), meaning r is a *rational interpolant* to the values $\{\alpha_k/\beta_k\}$ at the support points $\{t_k\}$.

Much of the literature on barycentric representations exploits this interpolatory property [7,8,10,12,27,55] by taking $\alpha_k = f(t_k)\beta_k$, so that r is an interpolant to some given function values $f(t_0), \dots, f(t_n)$ at the support points. In this case

$$r(z) = \sum_{k=0}^{n} \frac{f(t_k)\beta_k}{z - t_k} \Bigg/ \sum_{k=0}^{n} \frac{\beta_k}{z - t_k}, \tag{2.2}$$

with the coefficients $\{\beta_k\}$ commonly known as *barycentric weights*; we have $r(t_k) = f(t_k)$ as long as $\beta_k \neq 0$. While such a property is useful and convenient when we want to compute good approximations to f (see in particular the AAA algorithm), for a best rational approximation r^* we do not know a priori where r^* will intersect f, so enforcing interpolation is not always an option. (We use interpolation for Remez but not for AAA-Lawson or DC.) Formula (2.1), on the other hand, has $2n+1$ degrees of freedom and can be used to represent any rational function of type (n,n) by appropriately choosing $\{\alpha_k\}$ and $\{\beta_k\}$ [43, Theorem 2.1]. We remark that variants of (2.1) also form the basis for the popular vector fitting [30, 31] method used to match frequency response measurements of dynamical systems. A crucial difference is that the support points $\{t_k\}$ in vector fitting are selected to approximate poles of f, whereas, as we shall describe in detail, we choose them so that our representation uses a numerically stable basis.

2.1. Representing rational functions of nondiagonal type. Functions r expressed in the barycentric form (2.1) range precisely over the set of all rational functions of (not necessarily exact) type (n,n). When one requires rational functions of type (m,n) with $m \neq n$, additional steps are needed to enforce the type.

The approach we have followed, which we shall now describe, is a linear algebraic one based on previous work by Berrut and Mittelmann [9], where we make use of Vandermonde matrices to impose certain conditions that limit the numerator or denominator degree. An alternative might be to avoid such matrices and constrain the barycentric representation more directly to have a certain number of poles or zeros at $z = \infty$. This is a matter for future research.

To examine the situation, we first suppose $m < n$ and convert r into the conventional polynomial quotient representation

$$r(z) = \frac{\omega_t(z)N(z)}{\omega_t(z)D(z)} = \frac{\prod_{k=0}^{n}(z-t_k)\sum_{k=0}^{n}\dfrac{\alpha_k}{z-t_k}}{\prod_{k=0}^{n}(z-t_k)\sum_{k=0}^{n}\dfrac{\beta_k}{z-t_k}} =: \frac{p(z)}{q(z)}. \tag{2.3}$$

The numerator p is a polynomial of degree at most n. Further, it can be seen (either via direct computation or from [9, eq. (1)]) that p is of degree m $(< n)$ if and only if the vector $\alpha = [\alpha_0, \ldots, \alpha_n]^T$ lies in a subspace spanned by the null space of the (transposed) Vandermonde matrix

$$V_m = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ t_0 & t_1 & \cdots & t_n \\ \vdots & \vdots & & \vdots \\ t_0^{n-1-m} & t_1^{n-1-m} & \cdots & t_n^{n-1-m} \end{bmatrix}. \tag{2.4}$$

That is, to enforce $r \in \mathcal{R}_{m,n}$ with $m < n$, we require $\alpha \in \operatorname{span}(P_m)$, where $P_m \in \mathbb{R}^{(n+1)\times(m+1)}$ has orthonormal columns, obtained by taking the full QR factorization $V_m^T = \begin{bmatrix} P_m^{\perp} & P_m \end{bmatrix} \begin{bmatrix} R_m \\ 0 \end{bmatrix}$, where $P_m^{\perp} \in \mathbb{R}^{(n+1)\times(n-m)}$, $R_m \in \mathbb{R}^{(n-m)\times(n-m)}$. Note that R_m is nonsingular if the support points $\{t_k\}$ are distinct.

Similarly, for $m > n$, we need to take $m+1$ terms in (2.1), that is, $r(z) = \sum_{k=0}^{m} \alpha_k (z-t_k)^{-1} / \sum_{k=0}^{m} \beta_k (z-t_k)^{-1}$, and force $\beta \in \operatorname{span}(P_n)$, where $\operatorname{span}(P_n)$ is

the null space of the matrix

$$V_n = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ t_0 & t_1 & \cdots & t_m \\ \vdots & \vdots & & \vdots \\ t_0^{m-1-n} & t_1^{m-1-n} & \cdots & t_m^{m-1-n} \end{bmatrix}, \tag{2.5}$$

obtained by the QR factorization $V_n^T = \begin{bmatrix} P_n^\perp & P_n \end{bmatrix} \begin{bmatrix} R_n \\ 0 \end{bmatrix}$, where $P_n^\perp \in \mathbb{R}^{(m+1)\times(m-n)}$, $R_n \in \mathbb{R}^{(m-n)\times(m-n)}$.

In Section 4.4 we describe how to use the matrices P_m, P_n in specific situations. Since these matrices are obtained via V_m, V_n in (2.4)–(2.5) and real-valued Vandermonde matrices are usually highly ill-conditioned [4,5,48], care is needed when computing their null spaces, as extracting the orthogonal factors in QR (or SVD) is susceptible to numerical errors. Berrut and Mittelmann [9] suggest a careful elimination process to remedy this (for a slightly different problem). Here, in view of the Krylov-type structure of the matrices V_m^T and V_n^T, we propose the following simpler approach, based on an Arnoldi-style orthogonalization:

1. Let $Q = [1, \dots, 1]^T$ when $m > n$, and $Q = [f(t_0), \dots, f(t_n)]^T$ when $m < n$, and normalize to have Euclidean norm 1.
2. Let q be the last column of Q. Take the projection of $\mathrm{diag}(t_0, \dots, t_{\max(m,n)})q$ onto the orthogonal complement of Q, normalize, and append it to the right of Q. Repeat this $|m-n|$ times to obtain $Q \in \mathbb{C}^{(\max(m,n)+1)\times(|m-n|)}$. In MATLAB, this is `q = Q(:,end); q = diag(t)*q; for i = 1:size(Q,2), q = q-Q(:,i)*(Q(:,i)'*q); end, q = q/norm(q); Q = [Q,q];`.
3. Take the orthogonal complement $Q^\perp$ of Q via computing the QR factorization of Q. $Q^\perp$ is the desired matrix, P_m or P_n.

Note that the matrix Q in the final step is well conditioned ($\kappa_2(Q) = 1$ in exact arithmetic), so the final QR factorization is a stable computation.

2.2. Why does the barycentric representation help?. The choice of the support points $\{t_k\}$ is very important numerically, and indeed it is the flexibility of where to place these points that is the source of the power of barycentric representations. If the points are well chosen, the basis functions $1/(x - t_k)$ lead to a representation of r that is much better conditioned (often exponentially better) than the conventional representation as a ratio of polynomials. We motivate and explain our adaptive choice of $\{t_k\}$ for the Remez algorithm in Sections 4.3 and 4.5. The analogous choices for AAA-Lawson and DC are discussed in Sections 8.5 and 9.2.

To understand why a barycentric representation is preferable for rational approximation, we first consider the standard quotient representation p/q. It is well known that a polynomial will vary in size by exponentially large factors over an interval unless its roots are suitably distributed (approximating a minimal-energy configuration). If p/q is a rational approximation, however, the zeros of p and q will be positioned by approximation considerations, and if f has singularities or near-singularities they will be clustered near those points. In the clustering region, p and q will be exponentially smaller than in other parts of the interval and will lose much or all of their relative accuracy. Since the quotient p/q depends on that relative accuracy, its accuracy too will be lost.

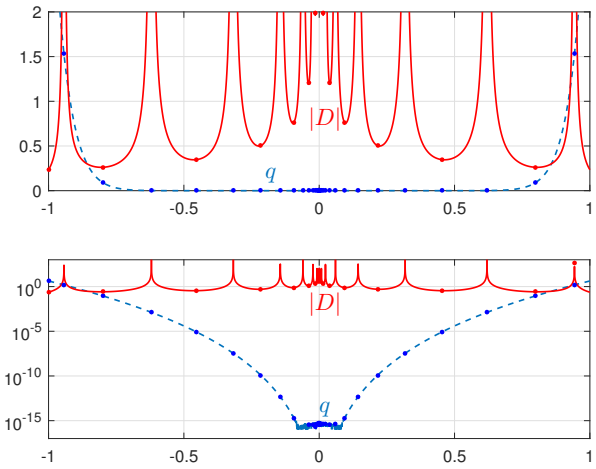

Fig. 2.1: Linear (top) and semilogy (bottom) plots of q and $|D|$ in $r^* = p/q = N/D$, the best rational approximation for $|x|$ of type $(m, n) = (20, 20)$. Here p, q are the polynomials in the classical quotient representation (1.1), and D is the denominator in the barycentric representation (2.1). The dots are the equioscillation points $\{x_\ell\}$, while the set of support points $\{t_k\}$ consists of every other point in $\{x_\ell\}$.

A barycentric quotient N/D, by contrast, is composed of terms that vary in size just algebraically across the interval, not exponentially, so this effect does not arise. If the support points are suitably clustered, N and D may have approximately uniform size across the interval (away from their poles, which cancel in the quotient), as illustrated in Figure 2.1.

2.3. Numerical stability of evaluation. Regarding the evaluation of r in the barycentric representation, Higham's analysis in [34, p. 551] (presented for barycentric polynomial interpolation, but equally valid for (2.1)) shows that evaluating $r(x)$ is backward stable in the sense that the computed value $\widehat{r}(x)$ satisfies

$$\widehat{r}(x) = \sum_{k=0}^{n} \frac{\alpha_k(1+\epsilon_{\alpha_k})}{x - t_k} \Bigg/ \sum_{k=0}^{n} \frac{\beta_k(1+\epsilon_{\beta_k})}{x - t_k}, \tag{2.6}$$

where $\epsilon_{\alpha_k}, \epsilon_{\beta_k}$ denote quantities of size $O(u)$, or more precisely, bounded by $(1 + u)^{3n+4}$. In other words, $\widehat{r}(x)$ is an exact evaluation of (2.1) for slightly perturbed $\{\alpha_k\}, \{\beta_k\}$. Note that when r represents a polynomial (as assumed in [34]), (2.6) does not imply backward stability. However, as a rational function for which we allow for backward errors in the denominator, (2.6) does imply backward stability.

For the forward error, we can adapt the analysis of [14, Proposition 2.4.3]. Assume that the computed coefficients $\widehat{\alpha}, \widehat{\beta}$ are obtained through a backward stable process,

$$\widehat{\alpha}_k = \alpha_k(1+\delta_{\alpha_k}),\ \delta_{\alpha_k} = O(\kappa_\alpha u),\ \widehat{\beta}_k = \beta_k(1+\delta_{\beta_k}),\ \delta_{\beta_k} = O(\kappa_\beta u), \quad k = 0, \ldots, n,$$

where κ_α and κ_β are condition numbers associated with the matrices used to determine $\widehat{\alpha}$ and $\widehat{\beta}$. Then, if x (the evaluation point) and $\{t_k\}$ are considered to be floating point numbers, we have

LEMMA 2.1. *The relative forward error for the computed value* $\widehat{r}(x)$ *of* (2.1) *satisfies*

$$\left|\frac{r(x)-\widehat{r}(x)}{r(x)}\right| \le u(n+3+O(\kappa_\alpha))\frac{\sum_{k=0}^n \left|\frac{\alpha_k}{x-t_k}\right|}{\left|\sum_{k=0}^n \frac{\alpha_k}{x-t_k}\right|}+u(n+2+O(\kappa_\beta))\frac{\sum_{k=0}^n \left|\frac{\beta_k}{x-t_k}\right|}{\left|\sum_{k=0}^n \frac{\beta_k}{x-t_k}\right|}+O(u^2). \tag{2.7}$$

Proof. This follows from [14, Prop. 2.4.3]. ∎

If the functions $|D(x)|$ and $|N(x)|$ appearing in the denominators of the right-hand side of (2.7) do not become too small over $[a,b]$, then we can expect the evaluation of $\widehat{r}$ to be accurate. Note that $|D(x)|$ is precisely the quantity examined in Section 2.2, where we argued that it takes values $O(1)$ or larger across the interval. Further, since $r(x) \approx f(x)$ implies $|N(x)| \approx |D(x)f(x)|$, we see that $|N(x)|$ is not too small unless $|f(x)|$ is small. Put together, we expect the barycentric evaluation phase to be stable unless $|f(x)|$ (and hence $|r(x)|$) is small. Note that since (2.7) measures the relative error, we usually cannot expect it to be $O(u)$ when $|r(x)| \approx |f(x)| \ll 1$.

3. The rational Remez algorithm. Initially developed by Werner [64, 65] and Maehly [38], the rational Remez algorithm extends the ideas of computing best polynomial approximations due to Remez [53, 54]. It can be summarized as follows:

Step 1 Set $k=1$ and choose $m+n+2$ distinct reference points

$$a \le x_0^{(k)} < \cdots < x_{m+n+1}^{(k)} \le b.$$

Step 2 Determine the *levelled error* $\lambda_k \in \mathbb{R}$ (positive or negative) and $r_k \in \mathcal{R}_{m,n}$ such that r_k has no pole on $[a,b]$ and

$$f(x_\ell^{(k)}) - r_k(x_\ell^{(k)}) = (-1)^{\ell+1}\lambda_k, \qquad \ell = 0,\dots,m+n+1. \tag{3.1}$$

Step 3 Choose as the next reference $m+n+2$ local maxima $\{x_\ell^{(k+1)}\}$ of $|f-r_k|$ such that

$$s(-1)^\ell \left(f(x_\ell^{(k+1)}) - r_k(x_\ell^{(k+1)})\right) \ge |\lambda_k|, \qquad \ell = 0,\dots,m+n+1, \tag{3.2}$$

with $s \in \{\pm 1\}$ and such that for at least one $\ell \in \{0,\dots,m+n+1\}$, the left-hand side of (3.2) equals $\|f-r_k\|_\infty$. If r_k has converged to within a given threshold $\varepsilon_t > 0$ (i.e., $(\|f-r_k\|_\infty - \lambda_k)/\|f-r_k\|_\infty \le \varepsilon_t$ [50, eq. (10.8)]) return r_k, else go to **Step 2** with $k \leftarrow k+1$.

If Step 2 is always successful, then convergence to the best approximation is assured [63, Theorem 9.14]. It might happen that Step 2 fails, namely when all rational solutions satisfying the equations (3.1) have poles in $[a,b]$. If the best approximation is non-degenerate and the initial reference set is already sufficiently close to optimal, then the algorithm will converge [11, §V.6.B]. To our knowledge, there is no effective way in general to determine when degeneracy is the cause of failure.

We note that the rational Remez algorithm can also be adapted to work in the case of *weighted* best rational approximation. An early account of this is given in [22]. Given a positive weight function $w \in \mathcal{C}([a,b])$, the goal is to find $r^* \in \mathcal{R}_{m,n}$ such that the weighted error $\|f-r^*\|_{w,\infty} = \max_{x\in[a,b]} |w(x)(f(x)-r^*(x))|$ is minimal. Equations (3.1) and (3.2) get modified to

$$w(x_\ell^{(k)})\left(f(x_\ell^{(k)}) - r_k(x_\ell^{(k)})\right) = (-1)^{\ell+1}\lambda_k, \qquad \ell = 0,\dots,m+n+1$$

and

$$s(-1)^{\ell} w(x_\ell^{(k+1)}) \left(f(x_\ell^{(k+1)}) - r_k(x_\ell^{(k+1)}) \right) \geq |\lambda_k|, \qquad \ell = 0, \ldots, m+n+1,$$

while the norm computations in Step 3 are taken with respect to w. Notice that the ability to work with the weighted error immediately allows us to compute the best approximation in the *relative* sense, by taking $w(x) = 1/|f(x)|$, assuming that f is nonzero over $[a, b]$.

We discuss each step of the rational Remez algorithm in the following sections. We first address Step 2, as this is the core part where the barycentric representation is used. We then discuss initialization (Step 1) in Section 5, and finding the next reference set (Step 3) in Section 6. Our focus is on the unweighted setting, but we comment on how our ideas can be extended to the weighted case as well.

4. Computing the trial approximation. For notational simplicity, in this section we drop the index k referring to the iteration number, the analysis being valid for any iteration of the rational Remez algorithm. We begin with the case $m = n$.

4.1. Linear algebra in a polynomial basis. We first derive the Remez algorithm in an (arbitrary) polynomial basis. At each iteration, we search for $r = p/q \in \mathcal{R}_{n,n}, p, q \in \mathbb{R}_n[x]$ such that

$$f(x_\ell) - r(x_\ell) = (-1)^{\ell+1}\lambda, \qquad \ell = 0, \ldots, 2n+1 \tag{4.1}$$

and assume that we represent p and q using a basis of polynomials $\varphi_0, \ldots, \varphi_n$ such that $\operatorname{span}_{\mathbb{R}} (\varphi_i)_{0 \leq i \leq n} = \mathbb{R}_n[x]$:

$$p(x) = \sum_{k=0}^{n} c_{p,k}\varphi_k(x), \quad q(x) = \sum_{k=0}^{n} c_{q,k}\varphi_k(x).$$

The linearized version of (4.1) is then given by

$$p(x_\ell) = q(x_\ell) \left(f(x_\ell) - (-1)^{\ell+1}\lambda \right),$$

which, in matrix form, becomes

$$\Phi_x c_p = \left(\begin{bmatrix} f(x_0) & & & \\ & f(x_1) & & \\ & & \ddots & \\ & & & f(x_{2n+1}) \end{bmatrix} - \lambda \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & -1 & \\ & & & \ddots \end{bmatrix} \right) \Phi_x c_q, \tag{4.2}$$

where $\Phi_x \in \mathbb{R}^{(2n+2)\times(n+1)}$ is the basis matrix $(\Phi_x)_{\ell,k} = \varphi_k(x_\ell), 0 \leq \ell \leq 2n+1, 0 \leq k \leq n$, and $c_p = [c_{p,0}, c_{p,1}, \ldots, c_{p,n}]^T$ and $c_q = [c_{q,0}, c_{q,1}, \ldots, c_{q,n}]^T$ are the coefficient vectors of p and q. Note that in this paper, vector and matrix indices always start at zero. Up to multiplying both sides on the left by a nonsingular diagonal matrix $D = \operatorname{diag}(d_0, \ldots, d_{2n+1})$, (4.2) can also be written as a generalized eigenvalue problem

$$\begin{bmatrix} D\Phi_x & -FD\Phi_x \end{bmatrix} \begin{bmatrix} c_p \\ c_q \end{bmatrix} = \lambda \begin{bmatrix} 0 & -SD\Phi_x \end{bmatrix} \begin{bmatrix} c_p \\ c_q \end{bmatrix}, \tag{4.3}$$

with $F = \operatorname{diag}(f(x_0), \ldots, f(x_{2n+1}))$ and $S = \operatorname{diag}\left((-1)^{k+1}\right)$.

As described in Powell [50, Ch. 10.2], solving (4.3) is usually done by eliminating c_p. His presentation considers the monomial basis, but the approach is valid for any basis of $\mathbb{R}_n[x]$. By taking the full QR decomposition of $D\Phi_x$, we get

$$D\Phi_x = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R.$$

Since $D\Phi_x$ is of full rank, we have $Q_1, Q_2 \in \mathbb{R}^{(2n+2)\times(n+1)}$ and $Q_2^T Q_1 = 0$. By multiplying (4.3) on the left by $Q^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T$, we obtain a block triangular eigenvalue problem with lower-right $(n+1)\times(n+1)$ block

$$Q_2^T F Q_1 R c_q = \lambda Q_2^T S Q_1 R c_q. \tag{4.4}$$

(The top-left $(n+1)\times(n+1)$ block has all eigenvalues at infinity, and is thus irrelevant.) In terms of polynomials, $(Q_1)_{\ell,k} = d_\ell \psi_k(x_\ell)$, $0 \le k \le n, 0 \le \ell \le 2n+1$, where $(\psi_k)_{0\le k\le n}$ is a family of orthonormal polynomials with respect to the discrete inner product $\langle f, g\rangle_x = \sum_{k=0}^{2n+1} d_k^2 f(x_k)\overline{g(x_k)}$. Moreover, if $(\varphi_k)_{0\le k\le n}$ is a degree-graded basis with $\deg \varphi_k = k$, then we have $\deg \psi_k = k, 0 \le k \le n$.

Let ω_x be the node polynomial associated with the reference nodes $x_0, \dots, x_{2n+1}$, and $\Omega_x = \operatorname{diag}(1/\omega_x'(x_0), \dots, 1/\omega_x'(x_{2n+1}))$. We have [50, p. 114]

$$V_x^T \Omega_x V_x = 0, \tag{4.5}$$

where $V_x \in \mathbb{R}^{(2n+2)\times(n+1)}$ is the Vandermonde matrix associated with $x_0, \dots, x_{2n+1}$, that is, $(V_x)_{i,j} = x_i^j$. Indeed,

$$\left(V_x^T \Omega_x V_x\right)_{i,j} = \sum_{\ell=0}^{2n+1} x_\ell^{i+j} \frac{1}{\omega_x'(x_\ell)} = (x^{i+j})[x_0, \dots, x_{2n+1}] = 0, \quad i,j \in \{0,\dots,n\},$$

the divided differences of order $2n+1$ of the function x^{i+j} at the $\{x_\ell\}$ nodes, hence 0 if $i + j \le 2n$.

By using the appropriate change of basis matrix in (4.5), we have

$$\Phi_x^T \Omega_x \Phi_x = 0. \tag{4.6}$$

Now, by multiplying (4.3) on the left by $\Phi_x^T \Omega_x D^{-1}$ and using (4.6), we can eliminate the c_p term to obtain

$$\Phi_x^T \Omega_x F \Phi_x c_q = \lambda \Phi_x^T \Omega_x S \Phi_x c_q. \tag{4.7}$$

Equation (4.7) is the extension of [50, Eq. (10.13)] from the monomial basis to $\varphi_0, \dots, \varphi_n$. Moreover, we have:

LEMMA 4.1. *The matrix $\Phi_x^T \Omega_x S \Phi_x$ is symmetric positive definite.*

Proof. Since $\Omega_x S = |\Omega_x|$, it means that $\Omega_x S$ is symmetric positive definite, and the conclusion follows. See also [50, Theorem 10.2]. ∎

Since $\Phi_x^T \Omega_x F \Phi_x$ is also symmetric, it follows that all eigenvalues of (4.7) are real and at most one eigenvector c_q corresponds to a pole-free solution r (i.e., q has no root on $[a,b]$). To see this, suppose to the contrary that there exists another pole-free solution r'. Then, from (4.1), it follows that either $r(x_k) - r'(x_k)$ are all zero or they alternate in sign at least $2n+1$ times. In both cases, $r - r' \in \mathcal{R}_{2n,2n}$ has at least $2n+1$ zeros inside $[a,b]$, leading to $r = r'$.

We can in fact transform (4.4) into a symmetric eigenvalue problem (an observation which seems to date to [49]) by considering the choice $D = |\Omega_x|^{1/2}$, which leads to $Q_2 = SQ_1$ in view of (4.6). The system becomes $Q_1^T SFQ_1 Rc_q = \lambda Q_1^T S^2 Q_1 Rc_q$, which, by the change of variables $y = Rc_q$, gives

$$Q_1^T SFQ_1 y = \lambda y. \tag{4.8}$$

To get c_p, from (4.2), we have $|\Omega_x|^{1/2} \Phi_x c_p = (F - \lambda S) |\Omega_x|^{1/2} \Phi_x c_q$, or equivalently (by multiplication on the left by Q_1^T),

$$Rc_p = Q_1^T (F - \lambda S) |\Omega_x|^{1/2} \Phi_x c_q = Q_1^T F Q_1 y.$$

The vectors Rc_p and Rc_q can be seen as vectors of coefficients of the numerator and denominator of r in the orthogonal basis $\psi_0, \ldots, \psi_n$. The (scaled) values of the denominator at each x_k corresponding to an eigenvector y can be recovered by computing

$$|\Omega_x|^{1/2} \Phi_x c_q = Q_1 y. \tag{4.9}$$

From this we can confirm the uniqueness of the pole-free solution: since the eigenvectors are orthogonal, there is at most one generating a vector of denominator values of the same sign, making it the only pole-free solution candidate.

4.2. Linear algebra in a barycentric basis. An equivalent analysis is valid if we take r in the barycentric form (2.1). Namely, (4.1) becomes

$$C\alpha = \left(\begin{bmatrix} f(x_0) & & & \\ & f(x_1) & & \\ & & \ddots & \\ & & & f(x_{2n+1}) \end{bmatrix} - \lambda \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & -1 & \\ & & & \ddots \end{bmatrix} \right) C\beta, \tag{4.10}$$

where C is now a $(2n+2) \times (n+1)$ Cauchy matrix with entries $C_{\ell,k} = 1/(x_\ell - t_k)$ (we assume for the moment $\{x_\ell\} \cap \{t_k\} = \varnothing$) and $\alpha = [\alpha_0, \alpha_1, \ldots, \alpha_n]^T$ and $\beta = [\beta_0, \beta_1, \ldots, \beta_n]^T$ are the column vectors of coefficients $\{\alpha_k\}$ and $\{\beta_k\}$. Again, this can be transformed into a generalized eigenvalue problem

$$\begin{bmatrix} C & -FC \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} 0 & -SC \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{4.11}$$

To reduce (4.11) to a symmetric eigenvalue problem as in (4.8), we form a link between the monomial and barycentric representations in terms of the basis matrices V_x and C. We have:

LEMMA 4.2. *Let V_x, ω_t be as defined above, and $V_t \in \mathbb{R}^{(n+1)\times(n+1)}$ be the Vandermonde matrix corresponding to the support points, i.e., $(V_t)_{i,j} = t_i^j$. Then*

$$\operatorname{diag}\left(\frac{1}{\omega_t(x_0)}, \ldots, \frac{1}{\omega_t(x_{2n+1})} \right) V_x = C \operatorname{diag}\left(\frac{1}{\omega_t'(t_0)}, \ldots, \frac{1}{\omega_t'(t_n)} \right) V_t.$$

Proof. If we look at an arbitrary element of the right-hand side matrix, we have

$$\left(C \operatorname{diag}\left(\frac{1}{\omega_t'(t_0)}, \ldots, \frac{1}{\omega_t'(t_n)} \right) V_t \right)_{j,\ell} = \sum_{k=0}^{n} \frac{1}{(x_j - t_k)\omega_t'(t_k)} t_k^\ell = \frac{x_j^\ell}{\omega_t(x_j)},$$

where the second equality is a consequence of the Lagrange interpolation formula. ∎

In place of Ω_x we will use the following matrix Δ:

LEMMA 4.3. *If* $\Delta = \operatorname{diag}\left(\omega_t(x_0)^2, \ldots, \omega_t(x_{2n+1})^2\right)\Omega_x$, *then* $C^T\Delta C = 0$.

Proof. We apply Lemma 4.2 and use the fact that $V_x^T\Omega_x V_x = 0$. Namely, $C^T\Delta C = \operatorname{diag}\left(\omega_t'(t_0), \ldots, \omega_t'(t_n)\right) V_t^{-T} V_x^T \Omega_x V_x V_t^{-1} \operatorname{diag}\left(\omega_t'(t_0), \ldots, \omega_t'(t_n)\right) = 0$. ∎

We now take the full QR decomposition of $|\Delta|^{1/2} C = (S\Delta)^{1/2} C$. We have

$$|\Delta|^{1/2} C = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R.$$

Based on Lemma 4.3, we can again take $Q_2 = SQ_1$. From (4.11) we get

$$\begin{bmatrix} |\Delta|^{1/2} C & -F\,|\Delta|^{1/2} C \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} 0 & -S\,|\Delta|^{1/2} C \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}.$$

Multiplying this expression on the left by $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T$ gives a block triangular matrix pencil, whose $(n+1)\times(n+1)$ lower-right corner is the barycentric analogue of (4.4): $Q_2^T F Q_1 R\beta = \lambda Q_2^T S Q_1 R\beta$. After substituting $Q_2^T = Q_1^T S$, we get

$$Q_1^T (SF) Q_1 R\beta = \lambda Q_1^T S^2 Q_1 R\beta, \tag{4.12}$$

which, by the change of variable $y = R\beta$, becomes a standard symmetric eigenvalue problem in λ with eigenvector y (recall that S, F are diagonal):

$$Q_1^T (SF) Q_1 y = \lambda y. \tag{4.13}$$

Hence, computing its eigenvalues is a well-conditioned operation. The values of the denominator of the rational interpolant corresponding to each eigenvector y can be recovered by computing

$$\operatorname{diag}\left(\omega_t(x_0), \ldots, \omega_t(x_{2n+1})\right) C\beta = \operatorname{diag}\left(\omega_t(x_0), \ldots, \omega_t(x_{2n+1})\right) |\Delta|^{-1/2} Q_1 y. \tag{4.14}$$

As in the polynomial case, there is at most one solution such that $q(x) = D(x)\omega_t(x)$ has no root in $[a, b]$; indeed, (4.9) and (4.14) represent the values of $q(x_\ell)$ for $r = p/q$ and x_ℓ satisfying equation (4.1). We use this sign test involving (4.14) to determine the levelled error λ that gives a pole-free r in Step 2 of our rational Remez algorithm. The appropriate β is then taken by solving $R\beta = y$. From (4.10), we have

$$|\Delta|^{1/2} C\alpha = (F - \lambda S)\,|\Delta|^{1/2} C\beta,$$

or equivalently (by multiplication on the left by Q_1^T)

$$R\alpha = Q_1^T (F - \lambda S)\,|\Delta|^{1/2} C\beta = Q_1^T (F - \lambda S) Q_1 y = Q_1^T F Q_1 y, \tag{4.15}$$

which allows us to recover α (and thus r).

Most of the derivations in this section can be carried over to the weighted approximation setting as well. In particular, the reader can check that the weighted versions of Equations (4.11) and (4.13) correspond to

$$\begin{bmatrix} C & -FC \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} 0 & -SW^{-1}C \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

and

$$Q_1^T(SF)Q_1 y = \lambda Q_1^T W^{-1} Q_1 y,$$

where $W = \operatorname{diag}(w(x_0), \ldots, w(x_{2n+1}))$ and all the other quantities are the same as before. While not leading to a symmetric eigenvalue problem, the symmetric and symmetric positive definite matrices appearing in the second pencil seem to suggest that the eigenproblem computations will again correspond to well-conditioned operations. Our experiments support this statement and we leave it as future work to make this rigorous. To recover α, (4.15) becomes $R\alpha = Q_1^T(F - \lambda S W^{-1})Q_1 y$.

4.3. Conditioning of the QR factorization. Since the above discussion makes heavy use of the matrix Q_1, it is desirable that computing the (thin) QR factorization $|\Delta|^{1/2} C = Q_1 R$ is a well-conditioned operation.

Here we examine the conditioning of Q_1, the orthogonal factor in the QR factorization of $|\Delta|^{1/2}C$, as this is the key matrix for constructing (4.12). We use the fact that the standard Householder QR algorithm is invariant under column scaling, that is, it computes the same Q_1 for both $|\Delta|^{1/2} C$ and $|\Delta|^{1/2} C\Gamma$ for diagonal Γ [33, Ch. 19]. We thus consider

$$\min_{\Gamma \in \mathcal{D}_{n+1}} \kappa_2(|\Delta|^{1/2} C\Gamma), \tag{4.16}$$

where $\mathcal{D}_{n+1}$ is the set of $(n+1) \times (n+1)$ diagonal matrices. We have

THEOREM 4.4. *Let $t_k \in (x_{2k}, x_{2k+1})$ for $k = 0, \ldots, n$ and $s_k \in (x_{2k+1}, x_{2k+2})$ for $k = 0, \ldots, n-1$, $s_n \in (x_{2n+1}, \infty)$, and define $\omega_s(x) = \prod_{k=0}^{n}(x - s_k)$. Then*

$$\min_{\Gamma \in \mathcal{D}_{n+1}} \kappa_2(|\Delta|^{1/2} C\Gamma) \le \max_{\ell} \sqrt{\left|\frac{\omega_s(x_\ell)}{\omega_t(x_\ell)}\right|} \cdot \max_{k} \sqrt{\left|\frac{\omega_t(x_k)}{\omega_s(x_k)}\right|}. \tag{4.17}$$

Proof. Let $\{y_j\}$ be a $(2n+2)$-element set such that $y_j \in (x_j, x_{j+1}), j = 0, \ldots, 2n$, $y_{2n+1} > x_{2n+1}$ and let $C_{x,y} \in \mathbb{R}^{(2n+2)\times(2n+2)}$ be the Cauchy matrix with elements $(C_{x,y})_{j,k} = 1/(x_j - y_k)$. If we consider $D_1 = \operatorname{diag}(\sqrt{|\omega_y(x_j)/\omega_x'(x_j)|})$ and $D_2 = \operatorname{diag}(\sqrt{|\omega_x(y_j)/\omega_y'(y_j)|})$, then the matrix $D_1 C_{x,y} D_2$ is orthogonal. This follows, for instance, if we examine the elements of its associated Gram matrix G and use divided differences. Indeed, for an arbitrary element $(G)_{j,k}$ with $j \neq k$, we have

$$\begin{aligned} -(G)_{j,k} &= \sqrt{\left|\frac{\omega_x(y_j)\omega_x(y_k)}{\omega_y'(y_j)\omega_y'(y_k)}\right|} \sum_{\ell=0}^{2n+1} \frac{\omega_y(x_\ell)}{(x_\ell - y_j)(x_\ell - y_k)\omega_x'(x_\ell)} \\ &= \sqrt{\left|\frac{\omega_x(y_j)\omega_x(y_k)}{\omega_y'(y_j)\omega_y'(y_k)}\right|} \left(\frac{\omega_y(x)}{(x - y_j)(x - y_k)}\right)[x_0, \ldots, x_{2n+1}] = 0. \end{aligned}$$

Similarly, since $\prod_{j \neq k}(x - y_j) = q(x)(x - y_k) + \omega_y'(y_k)$, with $q \in \mathbb{R}_{2n}[x]$, we have,

$$\begin{aligned} -(G)_{k,k} &= \frac{\omega_x(y_k)}{\omega_y'(y_k)} \sum_{\ell=0}^{2n+1} \frac{\omega_y(x_\ell)}{(x_\ell - y_k)^2 \omega_x'(x_\ell)} = \frac{\omega_x(y_k)}{\omega_y'(y_k)} \left(\frac{\prod_{j \neq k}(x - y_j)}{x - y_k}\right)[x_0, \ldots, x_{2n+1}] \\ &= \frac{\omega_x(y_k)}{\omega_y'(y_k)} \left(q(x) + \frac{\omega_y'(y_k)}{x - y_k}\right)[x_0, \ldots, x_{2n+1}] \\ &= \omega_x(y_k) \left(\frac{1}{x - y_k}\right)[x_0, \ldots, x_{2n+1}] = \omega_x(y_k) \frac{-1}{\omega_x(y_k)} = -1. \end{aligned}$$

Now, if we take $t_k = y_{2k}$, $s_k = y_{2k+1}$, for $k = 0, \dots, n$, there exist $D \in \mathcal{D}_{2n+2}$ and $\Gamma \in \mathcal{D}_{n+1}$ such that $|\Delta|^{1/2} C\Gamma = DD_1C_{x,y}D_2I_t$, where $D = \operatorname{diag}(\sqrt{|\omega_t(x_j)/\omega_s(x_j)|})$ and I_t is obtained by removing every second column from I_{2n+2}. In particular, $\Gamma = I_t^T D_2 I_t$. It follows that

$$\kappa_2(|\Delta|^{1/2} C\Gamma) \le \kappa_2(D) = \max_\ell \sqrt{\left|\frac{\omega_s(x_\ell)}{\omega_t(x_\ell)}\right|} \cdot \max_k \sqrt{\left|\frac{\omega_t(x_k)}{\omega_s(x_k)}\right|}.$$

□

Let $\Gamma = I_t^T D_2 I_t$ be as in the proof of Theorem 4.4. It turns out that for the choice $t_k = x_{2k+1} - \varepsilon, s_k = x_{2k+1} + \varepsilon$, for $k = 0, \dots, n$, as $\varepsilon \to 0$, the matrix $|\Delta|^{1/2} C$ has a finite limit $\widetilde{C}$ of full column rank, and similarly Γ tends to some diagonal matrix $\widetilde{\Gamma}$ with positive diagonal entries. From Theorem 4.4 and its proof we know that $\widetilde{C}\widetilde{\Gamma}$ has condition number 1, and, more precisely, orthonormal columns. We thus obtain an explicit thin QR decomposition of $\widetilde{C}$ (by direct calculation):

COROLLARY 4.5. *In the limit $t_k \nearrow x_{2k+1}$, for $k = 0, \dots, n$, the matrix $|\Delta|^{1/2} C$ converges to $\widetilde{C}$, with entries*

$$(\widetilde{C})_{j,k} = \begin{cases} \frac{|w_t'(t_k)|}{\sqrt{|w_x'(t_k)|}} & \text{if } j = 2k+1, \\ 0 & \text{if } j = 2\ell+1, \ell \ne k, \\ \frac{|w_t(x_j)|}{\sqrt{|w_x'(x_j)|}}/(x_j - t_k) & \text{if } j = 2\ell, \end{cases}$$

and explicit thin QR decomposition $\widetilde{C} = Q_1 R$, where

$$(Q_1)_{j,k} = \begin{cases} 1/\sqrt{2} & \text{if } j = 2k+1, \\ 0 & \text{if } j = 2\ell+1, \ell \ne k, \\ \left|\frac{w_t(x_j)}{w_t'(t_k)}\right| \sqrt{\left|\frac{w_x'(t_k)}{2w_x'(x_j)}\right|}/(x_j - t_k) & \text{if } j = 2\ell, \end{cases}$$

and $R = \sqrt{2}\, \operatorname{diag}\left(\frac{|w_t'(t_0)|}{\sqrt{|w_x'(t_0)|}}, \dots, \frac{|w_t'(t_n)|}{\sqrt{|w_x'(t_n)|}}\right)$.

Corollary 4.5 suggests the choice

$$t_k = x_{2k+1} \text{ for } k = 0, \dots, n. \tag{4.18}$$

This takes us back to the interpolatory mode of barycentric representations (2.2), in which we take $\alpha_k = \beta_k(f(t_k) - \lambda)$ for all k, instead of solving the system (4.15). This interpolatory mode formulation is used in [35, Sec. 3.2.3]. Our derivation provides a theoretical justification by showing that it is optimal with respect to the conditioning of $|\Delta|^{1/2} C\Gamma$. Moreover, since $\min_{\Gamma \in \mathcal{D}_{n+1}} \kappa_2(\widetilde{C}\Gamma) = 1$ in (4.16), forming the QR factorization of $|\Delta|^{1/2} C$ via a standard algorithm (e.g. Householder QR) to obtain Q_1 is actually unnecessary, as the explicit form of Q_1 is given in Corollary 4.5. In addition, we reduce the problem to a symmetric eigenvalue problem (4.13), resulting in well-conditioned eigenvalues, with β being obtained by solving the diagonal system $R\beta = y$ with y as in (4.13). Compared to (4.1), where we want q to have the same sign over $\{x_\ell\}$, we similarly require that β and thus y have components alternating in sign, which uniquely fixes the norm 1 eigenvector y in (4.13). Our approach also allows for nondiagonal types, as we describe next.

4.4. The nondiagonal case $m \neq n$. As pointed out in Section 2.1, when searching for a best approximant with $m \neq n$, we need to force the coefficient vector α or β to lie in a certain subspace. This results in modified versions of (4.11). Namely,

$$\begin{bmatrix} C & -FCP_n \end{bmatrix} \begin{bmatrix} \alpha \\ \widehat{\beta} \end{bmatrix} = \lambda \begin{bmatrix} 0 & -SCP_n \end{bmatrix} \begin{bmatrix} \alpha \\ \widehat{\beta} \end{bmatrix}, \qquad \text{when } m > n, \tag{4.19}$$

for $\widehat{\beta} \in \mathbb{C}^{n+1}$, and we take $\beta = P_n \widehat{\beta}$. Similarly,

$$\begin{bmatrix} CP_m & -FC \end{bmatrix} \begin{bmatrix} \widehat{\alpha} \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} 0 & -SC \end{bmatrix} \begin{bmatrix} \widehat{\alpha} \\ \beta \end{bmatrix}, \qquad \text{when } m < n, \tag{4.20}$$

for $\widehat{\alpha} \in \mathbb{C}^{m+1}$, and we take $\alpha = P_m \widehat{\alpha}$.

Below we describe the reduction of the generalized eigenvalue problems (4.19) and (4.20) to standard symmetric eigenvalue problems.

Case $m > n$. In this case, $C \in \mathbb{R}^{(m+n+2)\times(m+1)}$. Since $\det |\Delta|^{1/2} \neq 0$, (4.19) is equivalent to the generalized eigenvalue problem

$$\begin{bmatrix} |\Delta|^{1/2} C & -F |\Delta|^{1/2} CP_n \end{bmatrix} \begin{bmatrix} \alpha \\ \widehat{\beta} \end{bmatrix} = \lambda \begin{bmatrix} 0 & -S |\Delta|^{1/2} CP_n \end{bmatrix} \begin{bmatrix} \alpha \\ \widehat{\beta} \end{bmatrix}. \tag{4.21}$$

Consider the (thin) QR decomposition of $|\Delta|^{1/2} C \begin{bmatrix} P_n & P_n^\perp \end{bmatrix} = (S\Delta)^{1/2} C \begin{bmatrix} P_n & P_n^\perp \end{bmatrix}$:

$$|\Delta|^{1/2} C \begin{bmatrix} P_n & P_n^\perp \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} R = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 & R_{12} \\ 0 & R_2 \end{bmatrix}.$$

Then we have the identity $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T (SQ_1) = 0$, as can be verified analogously to (4.5) using divided differences. This implies $(SQ_1)^T |\Delta|^{1/2} C = 0$, so by left-multiplying (4.21) by $\begin{bmatrix} (SQ_1)^\perp & SQ_1 \end{bmatrix}^T$ we obtain a block upper-triangular eigenvalue problem with lower-right $(n+1) \times (n+1)$ block

$$(SQ_1)^T F Q_1 R_1 \widehat{\beta} = \lambda (SQ_1)^T S Q_1 R_1 \widehat{\beta},$$

which again reduces to the standard symmetric eigenvalue problem (setting $y = R_1 \widehat{\beta}$)

$$Q_1^T (SF) Q_1 y = \lambda y. \tag{4.22}$$

From (4.21), we have $|\Delta|^{1/2} C\alpha = (F - \lambda S) |\Delta|^{1/2} CP_n \widehat{\beta}$. Left-multiplying by $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T$ and using $\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T S |\Delta|^{1/2} CP_n = 0$, we obtain

$$\begin{aligned} R \begin{bmatrix} P_n & P_n^\perp \end{bmatrix}^T \alpha &= \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T F |\Delta|^{1/2} CP_n \widehat{\beta} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T F Q_1 R_1 \widehat{\beta} \\ &= \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T F Q_1 y. \end{aligned}$$

Therefore

$$\alpha = \begin{bmatrix} P_n & P_n^\perp \end{bmatrix} R^{-1} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T F Q_1 y,$$

which is obtained by computing the vector $\widehat{y} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T F Q_1 y$, then solving $R\widetilde{y} = \widehat{y}$ for $\widetilde{y}$, then $\alpha = \begin{bmatrix} P_n & P_n^\perp \end{bmatrix} \widetilde{y}$.

Case $m < n$. This case is analogous to the previous one; we highlight the differences. C is a $(m+n+2) \times (n+1)$ matrix. Equation (4.20) is equivalent to

$$\begin{bmatrix} |\Delta|^{1/2} CP_m & -F\,|\Delta|^{1/2} C \end{bmatrix} \begin{bmatrix} \widehat{\alpha} \\ \beta \end{bmatrix} = \lambda \begin{bmatrix} 0 & -S\,|\Delta|^{1/2} C \end{bmatrix} \begin{bmatrix} \widehat{\alpha} \\ \beta \end{bmatrix}. \tag{4.23}$$

Consider the (thin) QR decompositions

$$|\Delta|^{1/2} C = (S\Delta)^{1/2} C = Q_1 R, \qquad |\Delta|^{1/2} CP_m = \widehat{Q}_1 \widehat{R}.$$

Here $Q_1 \in \mathbb{R}^{(m+n+2)\times(n+1)}, \widehat{Q}_1 \in \mathbb{R}^{(m+n+2)\times(m+1)}$. We have $\widehat{Q}_1^T (SQ_1) = 0$, which again can be established using divided differences. This implies $(SQ_1)^T\,|\Delta|^{1/2} CP_m = 0$, so left-multiplying equation (4.23) by $\begin{bmatrix} (SQ_1)^\perp & SQ_1 \end{bmatrix}^T$ results in a block upper-triangular eigenvalue problem with lower-right block

$$(SQ_1)^T F Q_1 R\beta = \lambda (SQ_1)^T S Q_1 R\beta,$$

which also reduces to the standard symmetric eigenvalue problem (setting $y = R\beta$)

$$Q_1^T (SF) Q_1 y = \lambda y. \tag{4.24}$$

From (4.23), we have $|\Delta|^{1/2} CP_m \widehat{\alpha} = (F - \lambda S)\,|\Delta|^{1/2} C\beta$. Left-multiplying by $\widehat{Q}_1^T$ and using $\widehat{Q}_1^T S\,|\Delta|^{1/2} C = 0$, we obtain

$$\widehat{R}\widehat{\alpha} = \widehat{Q}_1^T F\,|\Delta|^{1/2} C\beta = \widehat{Q}_1^T F Q_1 R\beta = \widehat{Q}_1^T F Q_1 y.$$

Therefore

$$\widehat{\alpha} = \widehat{R}^{-1} \widehat{Q}_1^T F Q_1 y,$$

obtained via $\widehat{y} = \widehat{Q}_1^T F Q_1 y$, then solving the linear system $\widehat{R}\widehat{\alpha} = \widehat{y}$.

Analogously to our comments at the end of Section 4.2, the analysis for nondiagonal approximation presented here carries over to the weighted setting. In both the $m > n$ and $m < n$ scenarios, the standard symmetric eigenproblems (4.22) and (4.24) become

$$Q_1^T (SF) Q_1 y = \lambda Q_1^T W^{-1} Q_1 y,$$

where $y = R_1 \widehat{\beta}$ when $m > n$ and $y = R\beta$ when $m < n$. Recovering the set of barycentric coefficients in the numerator corresponds to solving the systems

$$\alpha = \begin{bmatrix} P_n & P_n^\perp \end{bmatrix} R^{-1} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}^T (F - \lambda S W^{-1}) Q_1 y, \qquad m > n$$

and

$$\widehat{\alpha} = \widehat{R}^{-1} \widehat{Q}_1^T (F - \lambda S W^{-1}) Q_1 y, \qquad m < n.$$

Stability and conditioning. We have just shown that the matrices arising in our rational Remez algorithm have explicit expressions, and the eigenvalue problem reduces to a standard symmetric problem. Indeed, our experiments corroborate that we have greatly improved the stability and conditioning of the rational Remez algorithm using the barycentric representation. However, the algorithm is still not

guaranteed to compute r^* to machine precision. Let us summarize the situation for the unweighted case. As shown in Corollary 4.5, the computation of Q_1 can be done explicitly, and the linear system $y = R\beta$ is diagonal, hence can be solved with high relative accuracy. The main source of numerical errors is therefore in the symmetric eigenvalue problem (4.13), (4.22) or (4.24). As is well known, by Weyl's bound [57, Cor. IV.4.9], eigenvalues of symmetric matrices are well conditioned with condition number 1; thus λ is computed with $O(u)$ accuracy, assuming for simplicity that $\|f\|_\infty = 1$ (without loss of generality). The eigenvector, on the other hand, has conditioning $O(1/\text{gap})$ [57, Ch. V], where gap is the distance between the desired λ and the rest of the eigenvalues. These eigenvalues are equal to those of the nonzero eigenvalues of the generalized eigenproblem (4.3), and are inherent in the Remez algorithm, i.e., they cannot be changed e.g. by a change of bases. For a fixed f, gap tends to decrease as m, n increase, and we typically have gap $= O(|\lambda|)$. Hence the computed eigenvector tends to have accuracy $O(u/|\lambda|)$, and if the eigenvector y has small elements, the componentwise relative accuracy may be worse. The computation therefore breaks down (perhaps as expected) when $|\lambda| = O(u)$, that is, when the error curve has amplitude of size machine precision.

4.5. Adaptive choice of the support points. Theorem 4.4 gives an optimal choice of support points $t_k = x_{2k+1}$ in terms of optimizing $\min_{\Gamma\in\mathcal{D}_{n+1}} \kappa_2(|\Delta|^{1/2} C\Gamma)$. In Section 2.2 we discussed another desideratum for the support points $\{t_k\}$: the resulting $|D(x_\ell)| = |q(x_\ell)\prod_{k=0}^{n}(x_\ell - t_k)|$ should take uniformly large values for all ℓ. Fortunately, this requirement is also met with this choice, as was illustrated in Figure 2.1.

When $m \neq n$, (4.18) does not determine enough support points. We take the remaining $|m - n|$ support points from the rest of the reference points in Leja style, i.e., to maximize the product of the differences (see for instance [52, p. 334]). This is a heuristic strategy, and the optimal choice is a subject of future work: indeed, in this case $\min_{\Gamma\in\mathcal{D}_{n+1}} \kappa_2(|\Delta|^{1/2} CP_{m,n}\Gamma) > 1$.

5. Initialization. An indispensable component of a successful Remez algorithm implementation is a method for finding a good set of initial reference points $\{x_\ell\}$. A key element of our approach is the AAA-Lawson algorithm, which can efficiently find an approximate solution to the minimax problem (1.2) (to low accuracy).

5.1. Carathéodory-Fejér (CF) approximation. We first attempt to compute the CF approximant [59, 61] to f, and use it to find the initial reference points (as explained in Section 6). The dominant computation is an SVD of a Hankel matrix of Chebyshev coefficients, which usually does not cause a computational bottleneck. This method was also used in the previous Chebfun `remez` code. When f is smooth, the result produced by CF approximation is often indistinguishable from the best approximation, but nonsmooth cases may be very different.

5.2. AAA-Lawson approximation. This approach is based on the AAA algorithm [43] followed by an adaptation of the Lawson algorithm. The resulting algorithm is also based crucially on the barycentric representation. To keep the focus on Remez, we defer the details to Section 8.

The output of the AAA-Lawson iteration typically has a nearly equioscillatory error curve $e = f - r$, from which we find the initial set of reference points as the extrema of e. For the prototypical example $f = |x|$, AAA-Lawson initialization lets our barycentric `minimax` code converge for type up to $(40, 40)$. The entire process relies on a moderate number of SVDs (say $\max(m, n) + 10$).

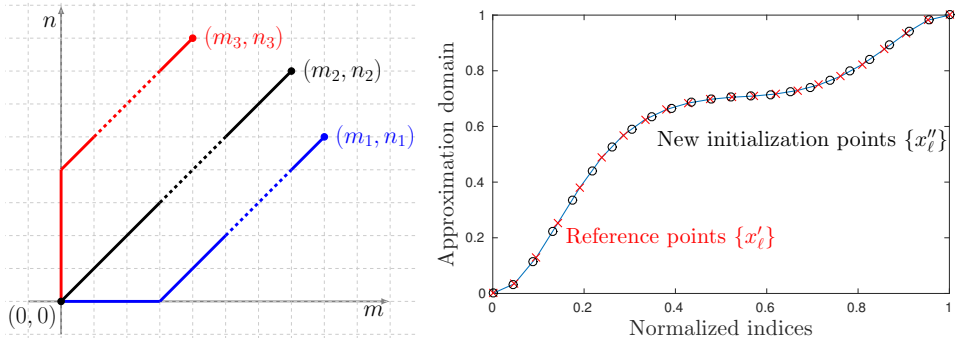

Fig. 5.1: Initialization with lower degree approximations. The left plot shows the three possible paths for updating the degrees (assuming the increment is $j = 1$): $m < n$ (red), $m = n$ (black) and $m > n$ (blue). The right plot shows how initialization is done at an intermediate step. The function is f_1 from Table 7.1, with a singularity at $x = 1/\sqrt{2}$. The y components of the red crosses correspond to the final references $\{x'_\ell\}$ for the $(m', n') = (10, 10)$ best approximation, while the y components of the black circles are the initial guess $\{x''_\ell\}$ for the $(m'', n'') = (11, 11)$ problem, taken based on the piecewise linear fit at $\{x'_\ell\}$. Note how the y components of both sets of points cluster near the singularity.

5.3. Using lower degree approximations. We resort to this strategy if CF and AAA-Lawson fail to produce a sufficiently good initial guess. For functions f with singularities in $[a, b]$, the reference sets $\{x_\ell\}$ corresponding to best approximations in (1.3) tend to cluster near these singularities as m and n increase.

It is sensible to expect that first computing a type (m', n') best approximation to f with $m' \ll m$ and $n' \ll n$ is easier (with convergence achieved if necessary with the help of CF or AAA-Lawson). We then proceed by progressively increasing the values of m' and n' by small increments j, typically $j \in \{1, 2, 4\}$. The steps taken follow a diagonal path, as explained in Figure 5.1. Note that in addition to improving the robustness of the Remez algorithm, this strategy can help detect degeneracy; recall the discussion after (1.3). It proves useful for many examples, including some of those shown in Section 7: type (n, n) approximations to $f(x) = |x|, x \in [-1, 1]$ for $n > 40$ and the f_1, f_2 and f_4 specifications in Table 7.1.

6. Searching for the new reference. We now turn to the updating strategy for the reference points $x_0 \ldots, x_{m+n+1}$ during the Remez iterations. These are a subset of the local extrema of the error function $e(x) = f(x) - r(x)$. To find them, we decompose the domain $[a, b]$ into subintervals of the form $[\tilde{x}_\ell, \tilde{x}_{\ell+1}]$ (and $[a, \tilde{x}_0]$ and $[\tilde{x}_{m+n+1}, b]$, if non-degenerate; here $\{\tilde{x}_\ell\}$ are the old reference points) and then compute Chebyshev interpolants $p_e(x)$ of $e(x)$ on each subinterval. In addition, if f has singularities (identified by Chebfun's `splitting on` functionality [46]), then we further divide the subintervals at those points. Since $e(x)$ is then smooth and each subinterval is small, typically a low degree suffices for $p_e = \sum_{i=0}^{k} c_i T_i(x)$: we start with $2^3 + 1$ points (degree $k = 8$), and resample if necessary (determined by examining the decay of the Chebyshev coefficients). We then find the roots of $p'_e(x) = \sum_{i=1}^{k} i c_i U_{i-1}(x)$ (using the formula $T'_n(x) = nU_{n-1}(x)$) via the eigenvalues of the colleague matrix for Chebyshev polynomials of the second kind [28]. Typically, one local extremum per subinterval is found, resulting in $m + n + 2$ points, including the endpoints. If more extrema are found, we evaluate the values of $|e(x)|$ at those points

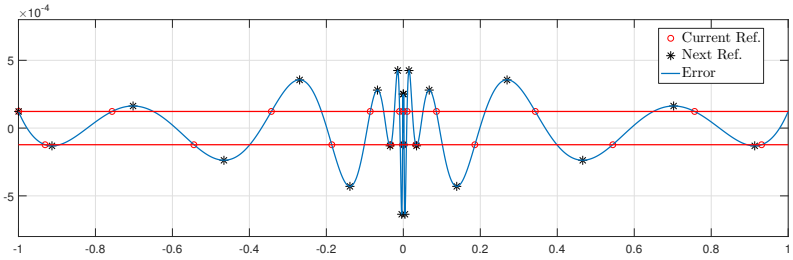

Fig. 6.1: Illustration of how a new set of reference points (black stars) is found from the current error function $e = f - r$ (blue curve). Shown here is the error curve after three Remez iterations in finding the best type $(10, 10)$ approximation to $f(x) = |x|$ on $[-1, 1]$. We split this interval into subintervals separated by the previous reference points (red circles), and approximate e on each subinterval by a low-degree polynomial. We then find the roots of its derivative.

and select those with the largest values that satisfy (3.2).

7. Numerical results. All computations in this section were done using Chebfun's new `minimax` command in standard IEEE double precision arithmetic.

Let us start with our core example of approximating $|x|$ on $[-1, 1]$, a problem discussed in detail in [58, Ch. 25]. For more than a century, this problem has attracted interest. The work of Bernstein and others in the 1910s led to the theorem that degree $n \geq 0$ polynomial approximations of this function can achieve at most $O(n^{-1})$ accuracy, whereas Newman in 1964 showed that rational approximations can achieve root-exponential accuracy [45]. The convergence rate for best type (n, n) approximations was later shown by Stahl [56] to be $E_{n,n}(|x|, [-1, 1]) \sim 8e^{-\pi\sqrt{n}}$.

This result had in fact been conjectured by Varga, Ruttan and Carpenter [62] based on a specialized multiple precision (200 decimal digits) implementation of the Remez algorithm. Their computations were performed on the square root function, using the fact that $E_{2n,2n}(|x|, [-1, 1]) = E_{n,n}(\sqrt{x}, [0, 1])$, as follows from symmetry. They went up to $n = 40$. In both settings, the equioscillation points cluster exponentially around $x = 0$ (see second plot of Figure 7.1), making it extremely difficult to compute best approximations. Our barycentric Remez algorithm in double precision arithmetic is able to match their performance, in the sense that we obtain the type $(80, 80)$ best approximation to $|x|$ in less than 15 seconds on a desktop machine. The results are showcased in Figure 7.1, where our levelled error computation for the type $(80, 80)$ approximation (value $4.39\ldots \times 10^{-12}$) matches the corresponding error of [62, Table 1] to two significant digits, even though the floating point precision is no better than 10^{-16}.

Running the other non-barycentric codes (Maple's `numapprox[minimax]`, Mathematica's `MiniMaxApproximation` (which requires f to be analytic on $[a, b]$), and Chebfun's previous `remez`) on the same example resulted in failures at very small values of n (all for $n \leq 8$).

The robustness of our algorithm is also illustrated by the examples of Table 7.1 and Figure 7.2, which is a highlight of the paper. Computing these five approximations takes in total less than 50 seconds with `minimax`. Example f_4 is taken from [60, §5], while f_5 is inspired by [51]. The difficulty of approximating f_5 is even more pronounced than for $|x|$, since best type (n, n) approximations to f_5 offer at most $O(n^{-1})$ accu-

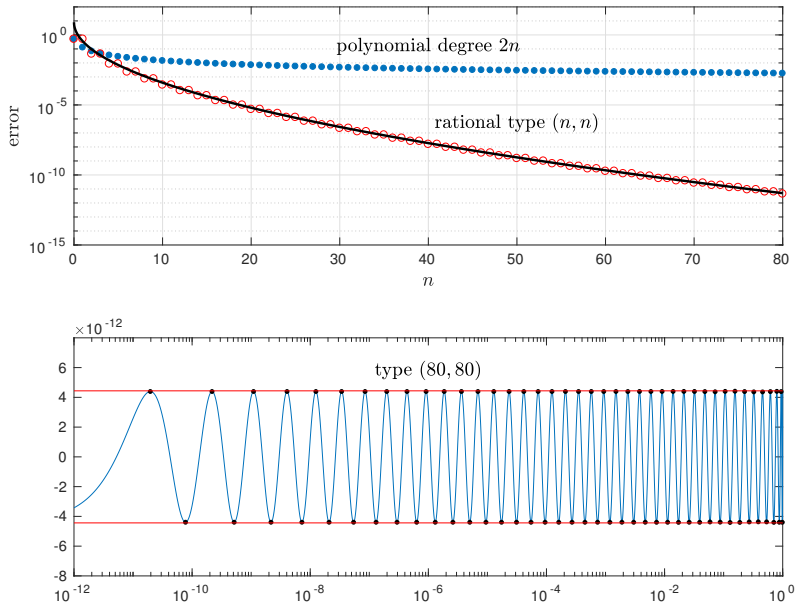

Fig. 7.1: In the first plot, the upper dots show the best approximation errors for the degree $2n$ best polynomial approximations of $|x|$ on $[-1,1]$, while the lower ones correspond to the best type (n,n) rational approximations, superimposed on the asymptotic formula from [56]. The bottom plot shows the minimax error curve for the type $(80,80)$ best approximation to $|x|$. Note that the horizontal axis has a log scale: the alternant ranges over 11 orders of magnitude. The positive part of the domain $[-1,1]$ is shown (by symmetry the other half is essentially the same).

Table 7.1: Best approximation to five difficult functions by the barycentric rational Remez algorithm. f_1'' is discontinuous at $x = 1/\sqrt{2}$, f_2' is discontinuous at $x = 0$, f_3' is unbounded as $x \to 0$, f_4 has two sharp peaks at $x = \pm 0.6$, and f_5 has a logarithmic singularity at $x = 0$.

i	f_i	$[a,b]$	(m,n)	$\|f - r^*\|_\infty$
1	$\begin{cases} x^2, & x < \frac{1}{\sqrt{2}} \\ -x^2 + 2\sqrt{2}x - 1, & \frac{1}{\sqrt{2}} \le x \end{cases}$	$[0,1]$	$(22,22)$	2.439×10^{-9}
2	$\lvert x\rvert\sqrt{\lvert x\rvert}$	$[-0.7,2]$	$(17,71)$	4.371×10^{-8}
3	$x^3 + \frac{\sqrt[3]{x}e^{-x^2}}{8}$	$[-0.2,0.5]$	$(45,23)$	2.505×10^{-5}
4	$\frac{100\pi(x^2-0.36)}{\sinh(100\pi(x^2-0.36))}$	$[-1,1]$	$(38,38)$	1.780×10^{-12}
5	$-\frac{1}{\log\lvert x\rvert}$	$[-0.1,0.1]$	$(8,8)$	1.52×10^{-2}

racy (a stark contrast to the root-exponential behavior of $E_{n,n}(|x|,[-1,1])$) and the reference points cluster even more strongly, quickly falling below machine precision.

In Figures 7.3 and 7.4, we further illustrate `minimax` and its weighted variant,

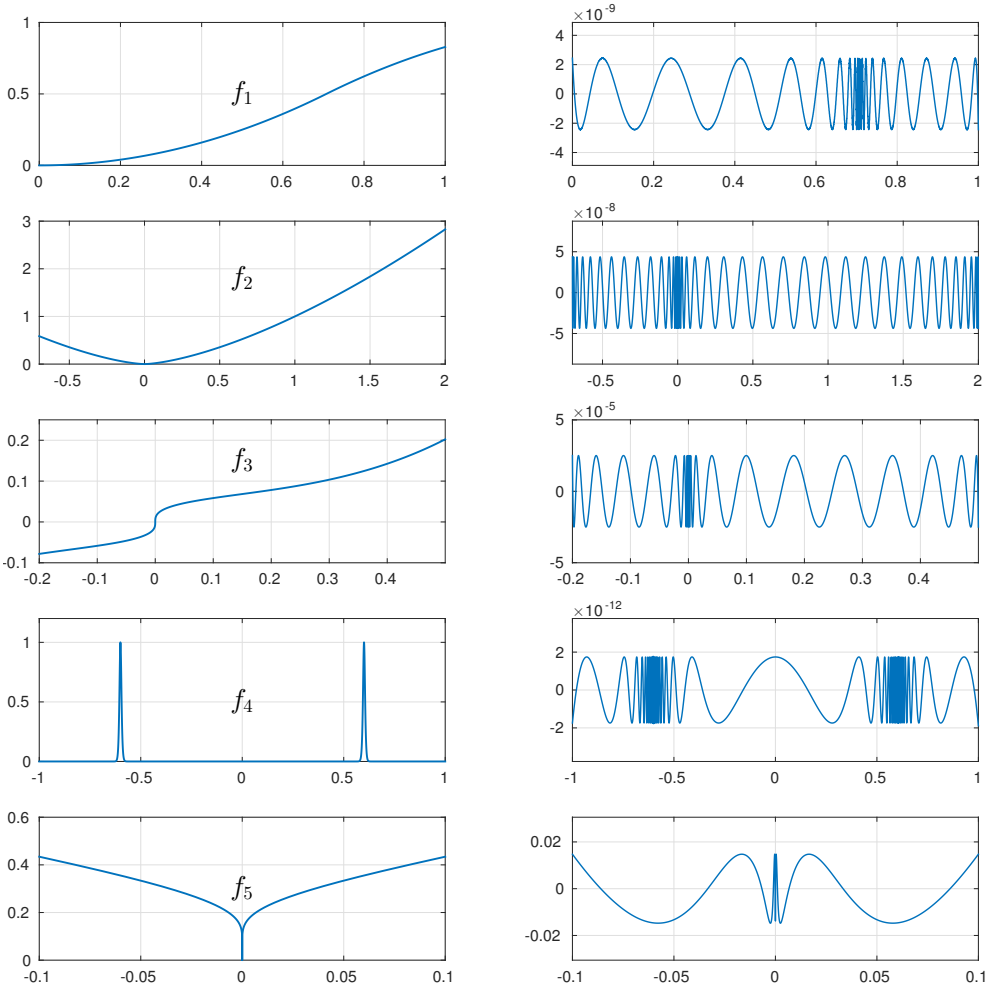

Fig. 7.2: Error curves for the best rational approximations of Table 7.1.

by revisiting some classical problems in rational approximation: the Zolotarev problems [2, Ch. 9]. Among other questions, Zolotarev asked what are the best rational approximants to the sign function (on the union of intervals $[-b,-a]\cup[a,b]$ for scalars $0<a<b$) and the $\sqrt{x}$ function (in the relative sense, i.e., minimizing $\|1-r/\sqrt{x}\|_\infty$) on $[1/b^2, 1/a^2]$. Zolotarev proved these problems are mathematically equivalent through the identity $\mathrm{sign}(x) = x\sqrt{1/x^2}$: if r is the type (m,m) best approximant to $\sqrt{x}$ on $[1/b^2, 1/a^2]$, then $\mathrm{sign}(x) - xr(1/x^2)$ is found to equioscillate at $4m+4$ points on $[-b,-a]\cup[a,b]$, so $xr(1/x^2)$ is the best approximant to $\mathrm{sign}(x)$ of type $(2m+1,2m)$ on $[-b,-a]\cup[a,b]$. Furthermore, Zolotarev gave explicit solutions involving Jacobi's elliptic functions. These rational functions have the remarkable property of preserving optimality under appropriate composition [42]. In Figure 7.3 we compute the best relative error approximant of type (m,m) to $\sqrt{x}$ using the weighted variant of our rational Remez algorithm. We then compute $xr(1/x^2)$, the type $(2m+1,2m)$ best approximant to the sign function. The error function is shown in Figure 7.4,

confirming Zolotarev's results.

We emphasize that the examples presented in this section are extraordinarily challenging, far beyond the capabilities of most codes for minimax approximation. Chebfun `minimax` not only solves them but does so quickly. For smoother functions such as analytic functions (with singularities, if any, lying far from the interval), we find that `minimax` usually easily computes r^* so long as $\|f - r^*\|_\infty$ is a digit or two larger than $u\|f\|_\infty$.

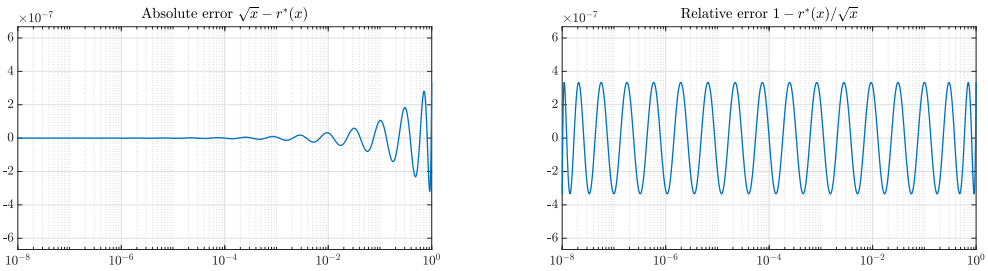

Fig. 7.3: Result of the weighted version of our barycentric Remez algorithm for the function $f(x) = \sqrt{x}, x \in [10^{-8}, 1]$ with $w(x) = 1/\sqrt{x}$ and a type $(17, 17)$ rational approximation. We plot the absolute error curve on the left, while the relative error (right), matching our choice of w, gives an expected equioscillating curve. This is Zolotarev's third problem.

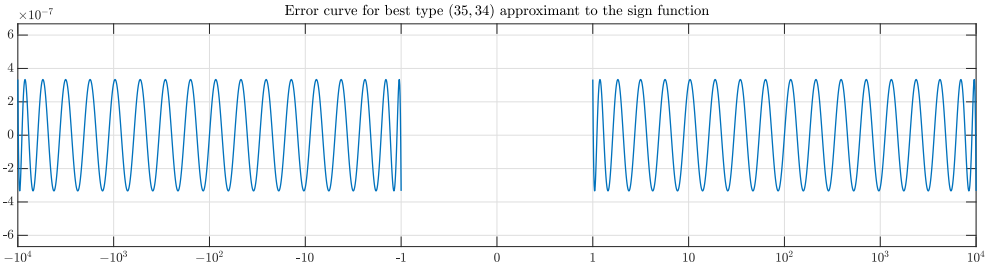

Fig. 7.4: The error in type $(35, 34)$ best approximation to the sign function on $[-10^4, -1] \cup [1, 10^4]$, computed via $xr(1/x^2)$, where $r(x) \approx \sqrt{x}$ as obtained in Figure 7.3. This is Zolotarev's fourth problem.

8. AAA-Lawson algorithm. Here we describe a new algorithm for rational approximation that we call the AAA-Lawson algorithm; in practice we recommend this for computing an initial guess for the Remez iteration. It applies on a finite, discrete set rather than the continuous interval $[a, b]$ as in (1.2). Specifically, we consider the problem

$$\underset{r \in \mathcal{R}_{m,n}}{\text{minimize}} \, \|f(Z) - r(Z)\|_\infty, \tag{8.1}$$

where $Z = \{z_1, \ldots, z_M\}$ is a set of distinct points (*sample points*) in $[a, b]$. The number M is usually large, e.g. 10^5, and in particular much bigger than m and n. The idea is that the solution for the discrete problem (8.1) should converge to the continuous one (1.2) if we discretize the interval densely enough.

AAA-Lawson proceeds as follows:

1. Use the AAA algorithm to find an approximant (2.2), in particular the support points $\{t_k\}$ for a rational approximation r to f. This step is not tied to a particular norm.
2. Use a variant of Lawson's algorithm to obtain a refined (near-best) rational approximant in the ℓ_∞ norm.

Below we first review the AAA algorithm, introduced in [43], then the Lawson algorithm, and then we present the AAA-Lawson combination.

8.1. The AAA algorithm. Given a function f and sample points $Z \in \mathbb{C}^M$, the AAA algorithm finds a rational approximant of type (n, n) represented as in (2.2) by $r(z) = \widetilde{N}(z)/\widetilde{D}(z) := \sum_{k=0}^n f(t_k)\beta_k(z-t_k)^{-1} / \sum_{k=0}^n \beta_k(z-t_k)^{-1}$. Here, the support points $\{t_k\}$ are a subset of Z chosen in an adaptive, greedy manner so as to improve the approximation as we increase n, exploiting the interpolatory property $\widetilde{N}(t_k)/\widetilde{D}(t_k) = f(t_k)$ for all k (unless $\beta_k = 0$). AAA takes only β_k as the unknowns, which are found by solving a linearized least-squares problem of the form $\text{minimize}_{\|\beta\|_2=1} \|f\widetilde{D} - \widetilde{N}\|_{\widetilde{Z}}$, where the subscript $\widetilde{Z}$ denotes the discrete 2-norm at points $\widetilde{Z} := Z \setminus \{t_0, \dots, t_n\}$. For details, see [43].

Noninterpolatory AAA. As we discussed in Section 2, the representation $\widetilde{N}(z)/\widetilde{D}(z)$ is unsuitable when the goal is to represent r^*: it is necessary to use the representation $r(z) = N(z)/D(z) = \sum_{k=0}^n \alpha_k(z-t_k)^{-1} / \sum_{k=0}^n \beta_k(z-t_k)^{-1}$ as in (2.1). This leads to a noninterpolatory variant of AAA, discussed briefly in [43, Section 10]. The resulting least-squares problem $\text{minimize}_{\|\alpha\|_2^2+\|\beta\|_2^2=1} \|fD - N\|_{\widetilde{Z}}$ has unknowns α and β. Written in matrix form, it takes the form

$$\underset{\|\alpha\|_2^2+\|\beta\|_2^2=1}{\text{minimize}} \left\| \begin{bmatrix} C & -FC \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \right\|_2, \tag{8.2}$$

where $F = \text{diag}(f(\widetilde{Z}))$, and $C_{\ell,k} = 1/(z_\ell - t_k)$ is the Cauchy (basis) matrix as in (4.11), but with rows corresponding to $z_\ell \in \{t_0, \dots, t_n\}$ removed. We take the same support points $\{t_k\}$ as in AAA. We solve (8.2) by computing the SVD of the matrix $\begin{bmatrix} C & -FC \end{bmatrix}$ and finding the right singular vector $v = \left[\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix}\right] \in \mathbb{R}^{2n+2}$ corresponding to the smallest singular value. As in Section 4.4, the case $m \neq n$ also uses the projection matrices P_m, P_n.

8.2. Lawson's algorithm. Lawson's algorithm [37] computes the best polynomial (linear) approximation based on an iteratively reweighted least-squares process. During the iteration, a set of weights is updated according to the residual of the previous solution.

Specifically, suppose that f is to be approximated on $Z = \{z_1, \dots, z_M\}$ in a linear subspace $\text{span}(g_i)_{i=0}^n$. With an initial set of weights $\{w_j\}_{j=1}^M$ such that $w_j \geq 0$ and $\sum_{j=1}^M w_j = 1$, one solves (using a standard solver) the weighted least-squares problem

$$\underset{c_0,\dots,c_n}{\text{minimize}} \|f - \sum_{i=0}^n c_i g_i\|_w = \sqrt{\sum_{j=1}^M w_j (f(Z_j) - \sum_{i=0}^n c_i g_i(Z_j))^2}, \tag{8.3}$$

and computes the residual $r_j = f(Z_j) - \sum_{i=0}^n c_i g_i(Z_j)$. The weights are then updated by $w_j := w_j|r_j|$, followed by the re-normalization $w_j := w_j / \sum_{i=1}^M w_i$. Iterating this process is known to converge linearly to the best polynomial approximant (with nontrivial convergence analysis [17]), and an acceleration technique is presented in [26].

8.3. AAA-Lawson. We now propose a rational variant of Lawson's algorithm. (A similar attempt was made in [20, § 6.5], though the formulation there is not the same: most notably, adjusting the exponent γ as done below appears to improve robustness significantly.) The idea is to incorporate Lawson's approach into nonin-terpolatory AAA, replacing (8.3) with a weighted version of (8.2), and updating the weights as in Lawson.

Specifically, given an initial set of weights $w \in \mathbb{R}^{M-(\max(m,n)+1)}$, usually all ones, and initializing the *Lawson exponent* $\gamma = 1$, we proceed as follows:

1. Solve the weighted linear least-squares problem

$$\underset{\|\alpha\|_2^2+\|\beta\|_2^2=1}{\text{minimize}} \|f(\widetilde{Z})D(\widetilde{Z}) - N(\widetilde{Z})\|_w, \tag{8.4}$$

   via the SVD of the matrix $\text{diag}(\sqrt{w})\begin{bmatrix} C & -FC \end{bmatrix}$ (recall (8.2)). If the resulting $\|f(Z) - N(Z)/D(Z)\|_\infty$ is not smaller than before, then set $\gamma := \gamma/2$.

2. Update w by

$$w_j \leftarrow w_j \left| f(Z_j) - \frac{N(Z_j)}{D(Z_j)} \right|^\gamma, \quad \forall j, \quad \text{then} \quad w_j := \frac{w_j}{\sum_i w_i} \tag{8.5}$$

   and return to step 1.

Note the exponent γ in (8.5). In the linear case, this is $\gamma = 1$. In the rational (nonlinear) case, for which experiments suggest convergence is a delicate issue, we have found that taking γ to be smaller makes the algorithm much more robust. We repeat the steps until w undergoes small changes, e.g. 10^{-3}, or a maximum number of iterations (e.g. 30) is reached.

We refer to this algorithm as AAA-Lawson. Each iteration is computed by an SVD of an $(M - \max(m,n) - 1) \times (m + n + 2)$ matrix, so the cost for k iterations is $O(kM(m+n)^2)$. Convergence analysis appears to be highly nontrivial and is out of our scope. We simply note here that if equioscillation of $f - N/D$ is achieved at $m+n+2$ points in $Z_* \subset Z$, then by defining w^* as $w_j^* = 1/\sqrt{|D(Z_j)|}$ for $j \in Z_*$ and 0 otherwise, we see that $w^*/\sum w^*$ (together with $N^*/D^* = r^*$, the solution of (1.2)) is a fixed point of the iteration.

8.4. Experiments with AAA-Lawson. Figure 8.1 compares AAA and AAA-Lawson (run for ten Lawson steps) for type (10,10) and (20,20) approximation of $f(x) = |x|$. The sample points are 10^4 equispaced points on $[-1, 1]$. Observe that the Lawson update significantly reduces the error and brings the error curve close to equioscillation.

AAA-Lawson is a new algorithm for rational minimax approximation. However, we do not recommend it as a practical means to obtain r^* over the classical Remez or differential correction algorithms. The reason is that its convergence is far from understood, and even when it does converge, the rate is slow (linear at best). We illustrate this in Figure 8.2. In our Remez algorithm context, we take a small number (say 10) of AAA-Lawson steps to obtain a set of initial reference points, thereby taking advantage of the initial stage of the AAA-Lawson convergence.

We note that other approaches for rational approximation are available, which can be used for initializing Remez. These include the Loewner approach presented in [39] and RKFIT [6]. In particular, the Loewner approach is well suited when approximating smooth functions (and sometimes non-smooth functions like f_4 [36]),

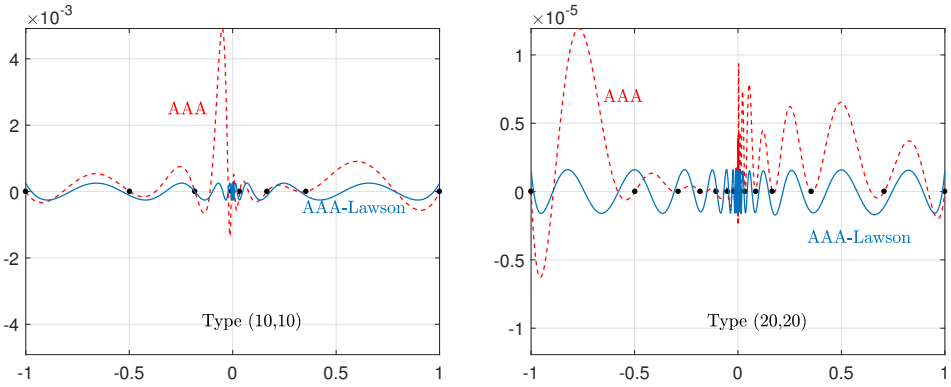

Fig. 8.1: Error of rational approximants to $f(x) = |x|$ by the AAA and AAA-Lawson algorithms. The black dots are the support points. They are also interpolation points for AAA, but not for AAA-Lawson.

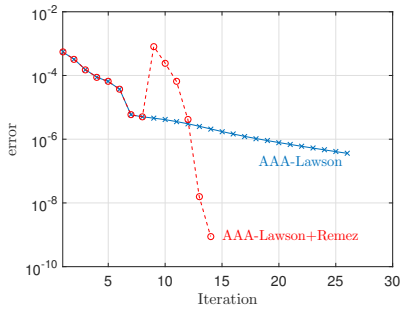

Fig. 8.2: Convergence of AAA-Lawson alone and AAA-Lawson followed by Remez, for $f(x) = |x|$, $m = n = 10$. The error is measured by $\|r^* - r_k\|_\infty$, where r_k is the kth iterate. AAA-Lawson converges linearly, whereas Remez converges quadratically.

often achieving an error of the same order of magnitude as the best approximation. Our experiments suggest that AAA-Lawson is at least as efficient and robust as these alternatives.

8.5. Adaptive choice of support points. At an early stage of the AAA-Lawson iteration, we usually do not have the correct number $(m+n+2)$ of reference (oscillation) points in the error curve. Therefore, choosing the support points $\{t_k\}$ as in (4.18) is not an option. Instead, we use the same support points chosen by the AAA algorithm, which is typically a good set. Once convergence sets in and the error curve of the AAA-Lawson iterates has at least $m+n+2$ alternation points, we can switch to the adaptive choice (4.18) as in Remez. We note, however, that adaptively changing the support points may further complicate the convergence, since it changes the linear least-squares problem (8.4).

8.6. Adaptive choice of the sample points. For solving the continuous problem (1.2), we take the sample point set Z to be M points uniformly distributed on $[a, b]$ ($M \lesssim 10^5$, chosen to keep the run time under control). Generally, it is necessary to sample more densely near a singularity if there is one; this is important e.g. for

$f(x) = |x|$. We incorporate this need as follows: use AAA to find the support points $\{t_k\}$ (assume they are sorted), and take M/n points between $[t_k, t_{k+1}]$.

9. A barycentric version of the differential correction algorithm. The DC algorithm, due to Cheney and Loeb [16], has the great advantage of guaranteed global convergence in theory [3,25], which applies whether the approximation domain X is an interval $[a, b]$ or a finite set. It can also be extended to multivariate approximation problems [32]. In practice, however, it may suffer greatly from rounding errors, and its speed is often disappointing on larger problems. As we shall now describe, we have found that the first of these difficulties can be largely eliminated by the use of barycentric representations with adaptively chosen support points. The second problem of speed, however, remains, which is why ultimately we prefer the Remez algorithm for most problems.

9.1. The barycentric formulation. For an effective implementation, X needs to be a finite set (e.g. obtained by discretizing $[a, b]$) to reduce each iteration to a linear programming (LP) problem. Considering the diagonal case $m = n$, a barycentric version of the DC algorithm can be defined recursively as follows. (We assume the support points are fixed to the values $t_0, \dots, t_n$, which do not belong to X.) Given $r_k = N_k/D_k \in \mathcal{R}_{n,n}(X)$, choose the partial fraction decompositions N and D of (2.1) that minimize the expression

$$\max_{x \in X} \left\{ \frac{|f(x)D(x) - N(x)| - \delta_k |D(x)|}{|D_k(x)|} \right\}, \tag{9.1}$$

subject to

$$\operatorname{sign}(\omega_t(x)D(x)) = \operatorname{sign}(\omega_t(y)D(y)), \qquad \forall x, y \in X, \quad x \neq y, \tag{9.2}$$

and

$$\max_{0 \le j \le n} |\beta_j| \le 1, \tag{9.3}$$

where $\delta_k = \max_{x \in X} |f(x) - r_k(x)|$. If $r = N/D$ is not good enough, continue with $r_{k+1} = r$. By imposing (9.3), we can establish convergence using an argument analogous to [3, Theorem 2]. In the polynomial basis setting, we know that the rate of convergence will ultimately be at least quadratic if the best approximation is non-degenerate [3, Theorem 3]. Non-diagonal approximations can be computed by adding the appropriate null space constraints as described in Section 4.4.

9.2. Choice of support points. Compared to the case of the barycentric Remez algorithm, changing the support points at each iteration of the DC algorithm makes it hard to impose a normalization condition similar to (9.3) or do a convergence analysis of the method. We therefore fix $\{t_k\}$ throughout the execution. The strategy we have adopted is based on Section 5.3: recursively construct type (ℓ, ℓ) approximations with $\ell \le n$. We take the set of support points of the (ℓ, ℓ) problem based on a piecewise linear fit of the final reference points of the $(\ell - 1, \ell - 1)$ problem (similar to what is shown in Figure 5.1).

9.3. Experiments. We have implemented[2] the barycentric DC algorithm in MATLAB using CVX [29] to specify the LP problems corresponding to (9.1)–(9.3),

[2]The prototype code used is available at https://github.com/sfilip/barycentricDC.

Table 9.1: Best type $(16, 16)$ approximations to four functions using the barycentric DC algorithm. X consists of 20000 equispaced points inside $[-1, 1]$.

i	f_i	$\|f_i - r^*\|_{X,\infty}$
1	$\sum_{k=0}^{\infty} 2^{-k} \cos(3^k x)$	0.1377
2	$\min\{\mathrm{sech}(3\sin(10x)), \sin(9x)\}$	0.0610
3	$\sqrt{\lvert x^3\rvert} + \lvert x + 0.5\rvert$	$1.2057 \cdot 10^{-4}$
4	$\left(\frac{1}{2}\mathrm{erf}\frac{x}{\sqrt{0.0002}} + \frac{3}{2}\right) e^{-x}$	$6.2045 \cdot 10^{-6}$

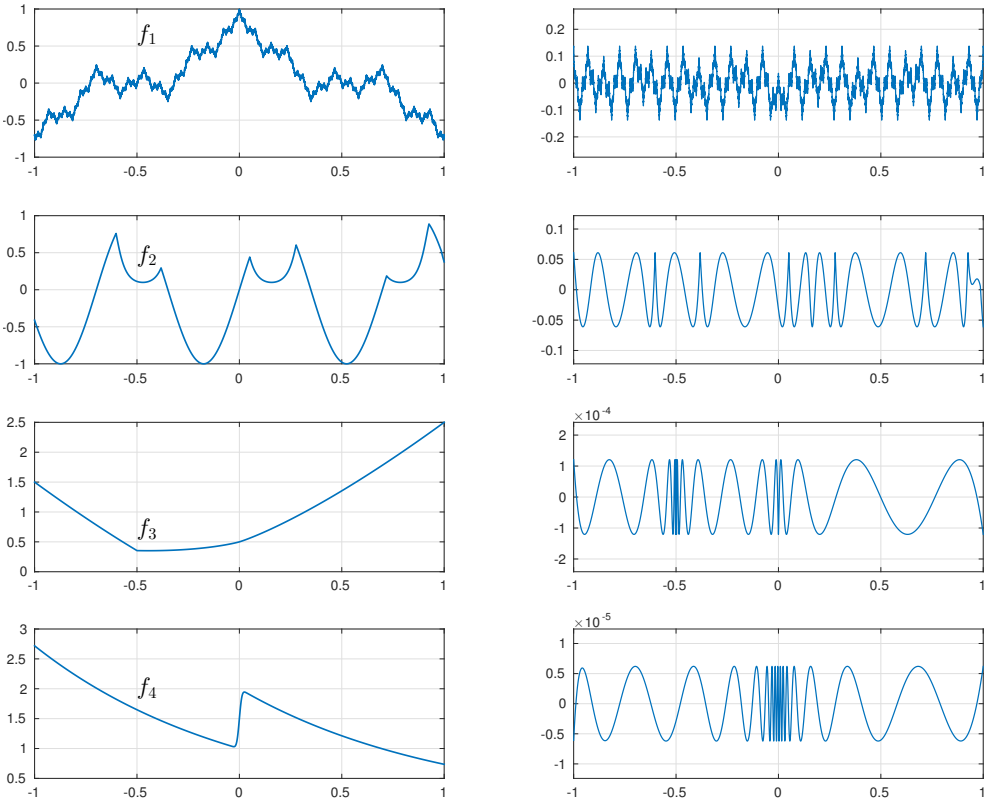

Fig. 9.1: The functions of Table 9.1 with error curves for best rational approximations computed by the barycentric DC algorithm.

which are then solved using MOSEK's [41] state-of-the-art LP optimizers. The four examples in Table 9.1 and Figure 9.1, for instance, demonstrate the effectiveness of the algorithm. For comparison, the sensitivity to the initial reference set prevented the convergence of our barycentric Remez implementation on *all four of* these examples. Function f_1 is particularly interesting since it is a version of Weierstrass's classic example of a continuous but nowhere differentiable function.

Using a monomial or Chebyshev basis representation for the LP formulations

quickly failed due to numerical errors, illustrating that the barycentric representation is crucial for the DC algorithm just as for the Remez algorithm.

We nevertheless echo the statement in the beginning of the section of the downsides of using the DC approach:

- Its overall cost. Producing the approximations in Figure 9.1 took several minutes in MATLAB on a desktop machine for each example.
- Numerical optimization tools for solving the corresponding LP problems break down at lower values of m and n than the ones we achieved with the barycentric Remez algorithm. We were usually able to go up to about type $(20, 20)$.

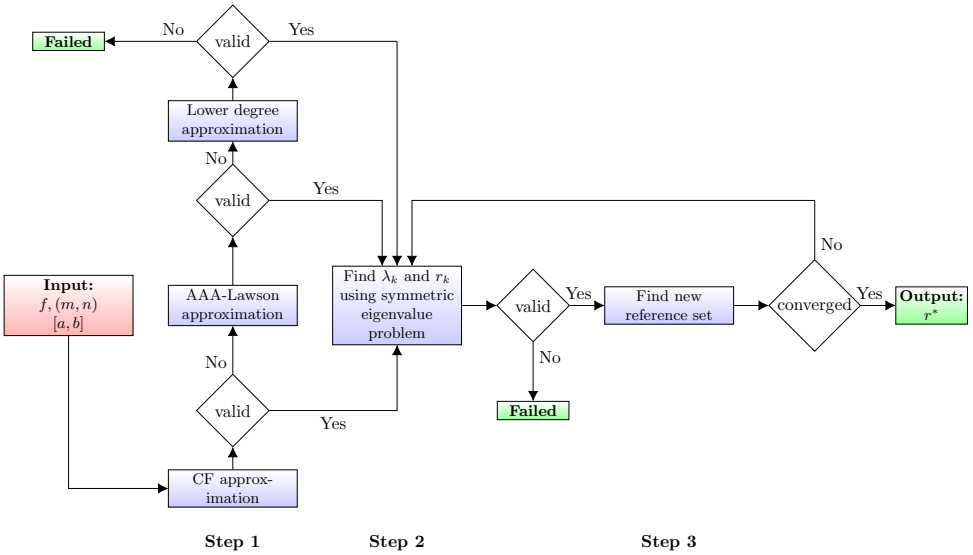

Fig. 10.1: Flowchart summarizing the `minimax` implementation of the rational Remez algorithm in the unweighted case. It follows the steps outlined at the start of Section 3. Step 1 consists of picking the initial reference set. This is done by applying in succession (if needed) the strategies discussed in Sections 5.1, 5.2 and 5.3. Next up in Step 2 is computing the current approximant r_k and alternation error λ_k. We do this by solving a symmetric eigenvalue problem (4.13), (4.22) or (4.24), depending on $m = n$, $m > n$ or $m < n$. We then pick, if possible, the eigenpair leading to a rational approximant with no poles in $[a, b]$ (see discussion around equation (4.14)). The next reference set is determined in Step 3 as explained in Section 6. If convergence is successful, the routine outputs a numerical approximant of r^*.

10. Minimax approximation in Chebfun. We have presented many algorithmic details that have enabled the design of a fast and robust Remez implementation. In closing we remind readers that all this is available in Chebfun and readily explored in a few lines of code. Download Chebfun version 5.7.0 or later from GitHub or `www.chebfun.org`, put it in your MATLAB path, and then try for example

```
[p,q,r] = minimax(@(x) abs(x),60,60);
fplot(@(x) abs(x)-r(x),[-1 1])
```

In a few seconds a beautiful curve with 123 exponentially clustered equioscillation points will appear. Figure 10.1 summarizes our algorithm in a flowchart.

REFERENCES

[1] Boost C++ Libraries. http://www.boost.org.

[2] N. I. Akhiezer. *Elements of the Theory of Elliptic Functions*, volume 79 of *Translations of Mathematical Monographs*. American Mathematical Society, 1990.

[3] I. Barrodale, M. J. D. Powell, and F. K. Roberts. The differential correction algorithm for rational ℓ_∞-approximation. *SIAM J. Numer. Anal.*, 9(3):493–504, 1972.

[4] B. Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numer. Math.*, 85(4):553–577, 2000.

[5] B. Beckermann and A. Townsend. On the singular values of matrices with displacement structure. *SIAM J. Matrix Anal. Appl.*, 38(4):1227–1248, 2017.

[6] M. Berljafa and S. Güttel. The RKFIT algorithm for nonlinear rational approximation. *SIAM J. Sci. Comp.*, 39(5):A2049–A2071, 2017.

[7] J.-P. Berrut. Rational functions for guaranteed and experimentally well-conditioned global interpolation. *Comput. Math. Appl.*, 15(1):1–16, 1988.

[8] J.-P. Berrut, R. Baltensperger, and H. D. Mittelmann. Recent developments in barycentric rational interpolation. In *Trends and Applications in Constructive Approximation*, pages 27–51. Springer, 2005.

[9] J.-P. Berrut and H. D. Mittelmann. Matrices for the direct determination of the barycentric weights of rational interpolation. *J. Comput. Appl. Math.*, 78(2):355–370, 1997.

[10] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Rev.*, 46(3):501–517, 2004.

[11] D. Braess. *Nonlinear Approximation Theory*. Springer, 1986.

[12] C. Brezinski and M. Redivo-Zaglia. Padé–type rational and barycentric interpolation. *Numer. Math.*, 125(1):89–113, 2013.

[13] F. Brophy and A. Salazar. Synthesis of spectrum shaping digital filters of recursive design. *IEEE T. Circuits Syst.*, 22(3):197–204, 1975.

[14] O. S. Celis. *Practical Rational Interpolation of Exact and Inexact Data*. PhD thesis, Universiteit Antwerpen, 2008.

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

[16] E. W. Cheney and H. L. Loeb. Two new algorithms for rational approximation. *Numer. Math.*, 3(1):72–75, 1961.

[17] A. K. Cline. Rate of convergence of Lawson's algorithm. *Math. Comp.*, 26(117):167–176, 1972.

[18] W. J. Cody. The FUNPACK package of special function subroutines. *ACM Trans. Math. Softw.*, 1(1):13–25, 1975.

[19] W. J. Cody. Algorithm 715: SPECFUN—a Portable FORTRAN Package of Special Function Routines and Test Drivers. *ACM Trans. Math. Softw.*, 19(1):22–30, 1993.

[20] P. Cooper. *Rational Approximation of Discrete Data with Asymptomatic Behaviour*. PhD thesis, University of Huddersfield, 2007.

[21] A. Curtis and M. R. Osborne. The construction of minimax rational approximations to functions. *Comput. J.*, 9(3):286, 1966.

[22] A. R. Curtis. Theory and calculation of best rational approximations. In *Methods of Numerical Approximation*, pages 139–148. Elsevier, 1966.

[23] A. Deczky. Equiripple and minimax (Chebyshev) approximations for recursive digital filters. *IEEE T. Acoust., Speech, Signal Process.*, 22(2):98–111, 1974.

[24] T. A. Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, Oxford, 2014.

[25] S. N. Dua and H. L. Loeb. Further remarks on the differential correction algorithm. *SIAM J. Numer. Anal.*, 10(1):123–126, 1973.

[26] S. Ellacott and J. Williams. Linear Chebyshev approximation in the complex plane using Lawson's algorithm. *Math. Comp.*, 30(133):35–44, 1976.

[27] M. S. Floater and K. Hormann. Barycentric rational interpolation with no poles and high rates of approximation. *Numer. Math.*, 107(2):315–331, 2007.

[28] I. J. Good. The colleague matrix, a Chebyshev analogue of the companion matrix. *Q. J. Math.*, 12(1):61–68, 1961.

[29] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar. 2014.

[30] B. Gustavsen. Improving the pole relocating properties of vector fitting. *IEEE Trans. Power Del.*, 21(3):1587–1592, 2006.

[31] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Trans. Power Del.*, 14(3):1052–1061, 1999.

[32] R. Hettich and P. Zencke. An algorithm for general restricted rational Chebyshev approxima-

tion. *SIAM J. Numer. Anal.*, 27(4):1024–1033, 1990.
[33] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, USA, second edition, 2002.
[34] N. J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, 24(4):547–556, 2004.
[35] A. C. Ioniță. *Lagrange Rational Interpolation and its Applications to Approximation of Large-Scale Dynamical Systems*. PhD thesis, Rice University, 2013.
[36] D. S. Karachalios. Hyperbolic function and the Loewner framework. private communication.
[37] C. L. Lawson. *Contributions to the Theory of Linear Least Maximum Approximations.* PhD thesis, University of California, Los Angeles, 1961.
[38] H. J. Maehly. Methods for fitting rational approximations, Parts II and III. *J. ACM*, 10(3):257–277, 1963.
[39] A. J. Mayo and A. C. Antoulas. A framework for the solution of the generalized realization problem. *Linear Algebra Appl.*, 425(2-3):634–662, 2007.
[40] G. Meinardus. *Approximation of Functions: Theory and Numerical Methods*. Springer, 1967.
[41] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28)*, 2015.
[42] Y. Nakatsukasa and R. W. Freund. Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev's functions. *SIAM Rev.*, 58(3):461–493, 2016.
[43] Y. Nakatsukasa, O. Sète, and L. N. Trefethen. The AAA algorithm for rational approximation. Technical report, 2016. To appear in *SIAM J. Sci. Comp.*
[44] Y. Nakatsukasa and L. N. Trefethen. Rational approximation of x^n. Technical report, 2018. To appear in *Proc. AMS*.
[45] D. J. Newman. Rational approximation to $|x|$. *Michigan Math. J.*, 11(1):11–14, 03 1964.
[46] R. Pachón, R. B. Platte, and L. N. Trefethen. Piecewise-smooth chebfuns. *IMA J. Numer. Anal.*, 30(4):898–916, 2010.
[47] R. Pachón and L. N. Trefethen. Barycentric-Remez algorithms for best polynomial approximation in the Chebfun system. *BIT Numer. Math.*, 49(4):721–741, 2009.
[48] V. Y. Pan. How bad are Vandermonde matrices? *SIAM J. Matrix Anal. Appl.*, 37(2):676–694, 2016.
[49] A. Pelios. Rational function approximation as a well-conditioned matrix eigenvalue problem. *SIAM J. Numer. Anal.*, 4(4):542–547, 1967.
[50] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, 1981.
[51] A. Pushnitski and D. Yafaev. Best rational approximation of functions with logarithmic singularities. *Constr. Approx.*, 46(2):243–269, 2017.
[52] L. Reichel. Newton interpolation at Leja points. *BIT Numer. Math.*, 30(2):332–346, 1990.
[53] E. Remes. Sur le calcul effectif des polynômes d'approximation de Tchebichef. *C. r. hebd. séances Acad. Sci.*, 199:337–340, 1934.
[54] E. Remes. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. *C. r. hebd. séances Acad. Sci.*, 198:2063–2065, 1934.
[55] C. Schneider and W. Werner. Some new aspects of rational interpolation. *Math. Comp.*, 47(175):285–299, 1986.
[56] G. Stahl. Best uniform approximation of $|x|$ on $[-1, 1]$. *Russian Acad. Sci. Sb. Math.*, 76(2):461–487, 1993.
[57] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory (Computer Science and Scientific Computing)*. Academic Press, 1990.
[58] L. N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013.
[59] L. N. Trefethen and M. H. Gutknecht. The Carathéodory-Fejér method for real rational approximation. *SIAM J. Numer. Anal.*, 20(2):420–436, 1983.
[60] J. Van Deun. Computing near-best fixed pole rational interpolants. *J. Comput. Appl. Math.*, 235(4):1077–1084, 2010.
[61] J. Van Deun and L. N. Trefethen. A robust implementation of the Carathéodory–Fejér method for rational approximation. *BIT Numer. Math.*, 51(4):1039–1050, 2011.
[62] R. S. Varga, A. Ruttan, and A. D. Carpenter. Numerical results on best uniform rational approximation of $|x|$ on $[-1, +1]$. *Mathematics of the USSR-Sbornik*, 74(2):271, 1993.
[63] G. A. Watson. *Approximation Theory and Numerical Methods*. Wiley, 1980.
[64] H. Werner. Die konstruktive Ermittlung der Tschebyscheff-Approximierenden im Bereich der rationalen Funktionen. *Arch. Ration. Mech. An.*, 11(1):368–384, 1962.
[65] H. Werner. Tschebyscheff-Approximation im Bereich der rationalen Funktionen bei Vorliegen einer guten Ausgangsnäherung. *Arch. Ration. Mech. An.*, 10(1):205–219, 1962.