An algorithm for best rational approximation based on barycentric rational interpolation

C. Hofreither

RICAM-Report 2020-37

An algorithm for best rational approximation based on barycentric rational interpolation

Clemens Hofreither[*]

August 28, 2020

Abstract

We present a novel algorithm for computing best uniform rational approximations to real scalar functions in the setting of zero defect. The method, dubbed BRASIL (best rational approximation by successive interval length adjustment), is based on the observation that the best rational approximation r to a function f must interpolate f at a certain number of interpolation nodes (x_j). Furthermore, the sequence of local maximum errors per interval (x_{j-1}, x_j) must equioscillate. The proposed algorithm iteratively rescales the lengths of the intervals with the goal of equilibrating the local errors. The required rational interpolants are computed in a stable way using the barycentric rational formula.

The BRASIL algorithm may be viewed as a fixed-point iteration for the interpolation nodes and converges linearly. We demonstrate that a suitably designed rescaled and restarted Anderson acceleration (RAA) method significantly improves its convergence rate.

The new algorithm exhibits excellent numerical stability and computes best rational approximations of high degree to many functions in a few seconds, using only standard IEEE double-precision arithmetic. A free and open-source implementation in Python is provided.

We validate the algorithm by comparing to results from the literature. We also demonstrate that it converges quickly in some situations where the current state-of-the-art method, the `minimax` function from the `Chebfun` package which implements a barycentric variant of the Remez algorithm, fails.

1 Introduction

1.1 Problem and prior work

Given a continuous function $f \in C[a, b]$ in a finite interval, our goal is to determine a rational function

$$r \in \mathcal{R}_n = \left\{ \frac{p}{q} : \; p, q \in \mathcal{P}_n, q \neq 0 \right\},$$

[*]Johann Radon Insitute for Computational and Applied Mathematics (RICAM), Altenbergerstr. 69, 4040 Linz, Austria. <clemens.hofreither@ricam.oeaw.ac.at>

where $\mathcal{P}_n$ is the space of real algebraic polynomials with degree at most n, such that r minimizes the maximum norm error

$$r = \operatorname*{arg\,min}_{\tilde{r} \in \mathcal{R}_n} \|f - \tilde{r}\|_{L_\infty[a,b]}. \tag{1}$$

This problem is of course classical, and best rational approximations play an important role in many applications, such as filter design in signal processing and the efficient evaluation of special functions. An application to the solution of fractional diffusion problems is sketched in Section 1.2. Analytic expressions for best rational approximations are rarely known. All this makes it quite surprising that the fast and stable computation of best rational approximations is still a challenging task in many situations.

The main workhorse for the numerical computation of best rational approximations is the rational Remez algorithm (see, e.g., [5]). It attempts to determine the points at which the error of the best rational approximation equioscillates. Starting with a suitable initial guess, it iteratively determines a rational approximation which passes through these points and shifts one or more points towards a nearby local maximum. The Remez algorithm generally converges quadratically in a neighborhood of the exact solution. While the basic idea of the algorithm is simple, there are several subtleties involved in its implementation. Furthermore, the method has notoriously poor robustness; namely, convergence is not guaranteed unless the initial guess is sufficiently close to the solution, and numerical stability is often very poor. The latter problem usually makes it necessary to resort to multi-precision floating point arithmetic, for which there is typically no hardware support, slowing down the method dramatically. For some works describing implementations of rational Remez algorithms, see [7, 27].

Big strides towards improving the stability of the rational Remez algorithm have recently been made [12] by using the barycentric representation of rational functions (see, e.g., [4]) with adaptively chosen support points. This approach made it possible to quickly compute best rational approximations using only standard double-precision arithmetic in many interesting examples. This method also has the advantage of having a high-quality implementation in the freely available `Chebfun` software package for Matlab [10].

There exist also other approaches for solving the rational best approximation problem, perhaps most prominently the differential correction (DC) algorithm by Cheney and Loeb [8]. Again, a more stable version of this method based on barycentric representations has been proposed recently [12]. Although this method is quite robust, it is less attractive in practice since it converges only linearly and involves the costly solution of a linear programming problem in each iteration.

Before we outline the novel contribution of the present work, we sketch a particular application which motivated the work on this problem. Although the problem of best rational approximation is universal enough not to require any further motivation, this application will provide a justification why the particular class of functions to which the novel algorithm is applicable are interesting. As we will see in the numerical tests in Section 6, existing methods do not perform well for these functions.

1.2 Motivation: fractional diffusion equations

Recently there has been significant activity in the development of numerical methods for solving elliptic fractional partial differential equations of the type

$$\mathcal{L}^\alpha u = f \qquad \text{in } \Omega,$$

where $\alpha \in (0,1)$, Ω is a bounded domain, $\mathcal{L}$ an elliptic diffusion operator augmented with homogeneous Dirichlet boundary conditions on $\partial\Omega$, f the right-hand side and u the sought solution. After discretization, a discrete solution $\mathbf{u} \in \mathbb{R}^m$ can be found as $A^\alpha \mathbf{u} = \mathbf{f}$, where $A \in \mathbb{R}^{m\times m}$ is a (typically large and sparse) matrix with real and positive spectrum arising from any number of popular discretization techniques [20]. Fractional matrix powers of sparse matrices are however dense in general and cannot be directly realized for large problems. An attractive approach due to its simplicity and excellent convergence properties is to compute

$$\tilde{\mathbf{u}} = \tilde{r}(A)\mathbf{f}$$

with a rational function $\tilde{r}(x)$ which is a good approximation to the function $x \mapsto x^{-\alpha}$ for $x \in [\lambda_1, \lambda_m]$, the spectral interval of A. The approximation $\tilde{\mathbf{u}}$ can be efficiently evaluated using the partial fraction decomposition of r. The author has recently shown [20] that many published methods for solving the fractional diffusion problem can be interpreted as such rational approximation methods.

The most recent method in this class is described in [15], where the choice

$$\tilde{r}(x) = \lambda_1^{-\alpha} r(\lambda_1 x^{-1})$$

is made with $r(x)$ being the best rational approximation with a chosen degree of the function $x \mapsto x^\alpha$ in $[0,1]$. The error converges asymptotically like $\sim e^{-\sqrt{n}}$, where n is the degree of r (cf. [24]). However, until now a major obstacle to the use of these methods in practice has been the computation of the best approximations $r(x) \approx x^\alpha$. In the recent report [14], the authors provide extensive tables with coefficients and errors of best rational approximations for $\alpha \in \{0.25, 0.5, 0.75\}$ and degrees up to $n = 8$. These results were produced using a Remez algorithm using quadruple-precision floating point arithmetic and in many cases required the use of computing time on the order of hours.

Enabling the computation of these approximations in a fast way was the main driving force behind the present work. Therefore, we are here most interested in approximation of functions of the type

$$f(x) = x^\alpha, \qquad f(x) = \frac{x^\alpha}{1 + qx^\alpha}, \qquad \alpha \in (0,1),\ q \geq 0,\ x \in [0,1],$$

both of which have applications in fractional diffusion-type problems [14]. Nevertheless, the proposed method works equally well for other functions with similar characteristics, as detailed later: best approximations with zero defect of functions with no interior singularities, but possible boundary singularities.

1.3 A novel algorithm for best rational approximation

As noted earlier, the most popular approach for computing best rational approximations, the rational Remez algorithm, is based on the idea of finding the

$2n+2$ equioscillation points at which the error assumes its local extrema. Here, n is the degree of the rational approximation. For a simple example, these nodes are shown as stars in the right plot of Figure 1. Between each consecutive pair of such points there lies a point (shown as dots in Figure 1) where the rational approximant interpolates the exact function.

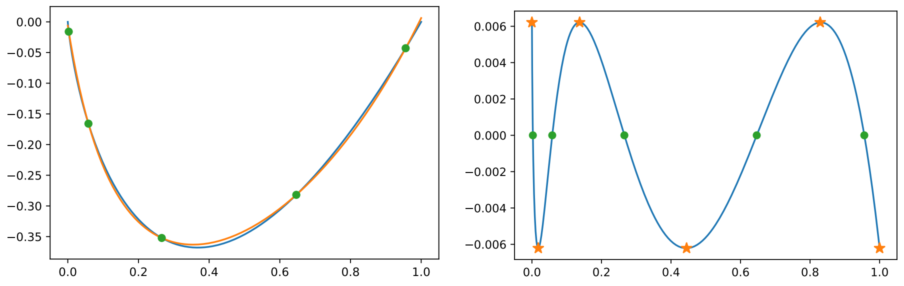


Figure 1: The function $f(x) = x \log x$, $x \in [0, 1]$, together with its best rational approximation r of degree 2 (left) and the resulting error $f - r$ (right). Equioscillation nodes are marked with stars, and interpolation nodes with dots.

The basic idea of the BRASIL algorithm is to determine the locations $(x_j)_{j=0}^{2n}$ of these $2n+1$ interpolation nodes and compute the best rational approximation by rational interpolation in these nodes. The algorithm first determines suitable initial guesses for the interpolation nodes $(x_j)_{j=0}^{2n}$ and then in each iteration

1. computes the rational interpolant through these nodes in barycentric representation,
2. determines the maximum error in each interval (x_j, x_{j+1}),
3. simultaneously rescales all intervals such that intervals where the error is too large are shrunk and intervals where the error is too small are enlarged.

These steps are repeated until the maximum errors are equilibrated to a desired tolerance. This idea results in a fast and robust method with excellent stability properties. The method is also significantly easier to implement than the Remez algorithm.

The remainder of this paper is structured as follows. Some important preliminaries on best rational approximations as well as barycentric rational interpolation are given in Section 2. The novel BRASIL algorithm is described in Section 3, where we also discuss a method for initializing the interpolation nodes and analyze the computational complexity. The convergence properties of BRASIL are investigated in Section 4, and a rescaled and restarted Anderson acceleration method is presented which can significantly improve the convergence rates. An open-source software implementation of the proposed algorithm is briefly discussed in Section 5, and numerical experiments demonstrating the accuracy, performance and robustness of the method are given in Section 6. At the end of that section, we also briefly discuss the excellent numerical stability of the new method by way of an example.

2 Preliminaries

2.1 Properties of best rational approximations

Consider again the best rational approximation problem (1). It is a classical result that the minimizer exists and is unique (see, e.g., [1, 26]). For the representation $r = p/q$ with polynomials p and q of minimal degree, we will denote by $\deg r := \max\{\deg p, \deg q\}$ the degree of the rational function. If $r \in \mathcal{R}_n$ is the best rational approximation to f of degree at most n, its *defect* is the number

$$d := n - \deg r \geq 0.$$

The defect plays an important role in the following classical equioscillation characterization of the best approximation (see [1] as well as [26] for historical references).

Theorem 1. *A unique best uniform rational approximation $r \in \mathcal{R}_n$ to $f \in C[a, b]$ exists. A rational function $\tilde{r} \in \mathcal{R}_n$ is equal to r if and only if the error $f - \tilde{r}$ equioscillates between at least $2n+2-d$ extreme points, where $d = n - \deg r$ is the defect of r.*

Equioscillation at k extreme points means that there exist k distinct points, $a \leq z_1 < \ldots < z_k \leq b$, such that

$$(f - \tilde{r})(z_j) = (-1)^{j+\delta} \|f - \tilde{r}\| \qquad \forall j = 1, \ldots, k$$

with $\delta \in \{0, 1\}$. Here we write $\|\cdot\|$ for the maximum norm in $[a, b]$.

In the remainder of this paper, we will assume zero defect, $d = 0$. This will generally hold true for the class of functions we are interested in. In particular, for the approximation of functions of the type x^α, this has been proven by Stahl [24]. Note, however, that some problems of practical interest, such as computing best rational approximations to $|x|$ in $[-1, 1]$, result in nonzero defects, and the proposed method cannot be directly applied to such problems. Some problems can be reformulated in order to satisfy this condition. For instance, in the particular case of the function $|x|^{2\alpha}$ in $[-1, 1]$, by a simple argument we obtain that its rational best-approximation error of degree $2n$ is equal to that of x^α in $[0, 1]$ of degree n [24].

2.2 Barycentric rational interpolation

Since the algorithm proposed in this work relies heavily on rational interpolation, we require a robust way of computing rational interpolants. Our main tool towards this end will be the so-called barycentric rational formula. This formula has a long history, and a comprehensive list of the related literature is beyond the scope of the present work; instead, we refer to [3, 4, 21] and the references therein for an overview. The main advantage of the barycentric formula is its well-documented superior numerical stability [23, 3, 18].

For nodes, values, and weights, respectively,

$$x_i \in \mathbb{R}, \quad f_i \in \mathbb{R}, \quad w_i \in \mathbb{R}, \quad i = 0, \ldots, n,$$

where the nodes (x_i) are pairwise distinct, the barycentric formula is given by

$$r(x) = \frac{\sum_{i=0}^{n} \frac{w_i}{x - x_i} f_i}{\sum_{i=0}^{n} \frac{w_i}{x - x_i}}. \tag{2}$$

It describes a rational function r of degree at most n with the interpolation property $r(x_i) = f_i$ for each $i \in \{0, \dots, n\}$ where $w_i \neq 0$. All rational functions with this interpolation property are parameterized by varying the weights (w_i). Note also that rescaling the vector of weights by a nonzero scalar does not change the function r.

The particular choice of the weights

$$w_i = \frac{1}{\prod_{k=0,k\neq i}^{n}(x_i - x_k)}, \qquad i = 0, \dots, n$$

leads to polynomial interpolants and is generally superior to the classical Lagrange interpolation formula in terms of numerical stability [18]. Different choices of the weights lead to different, in general rational, interpolants through the $n+1$ nodes (x_i). In our case, we will choose the weights in such a way as to enforce interpolation conditions in n additional nodes, yielding rational interpolation in $2n+1$ nodes.

Assume the nodes (x_i) are given in increasing order and introduce additional nodes $\hat{x}_i$, $i = 1, \dots, n$ with the interlacing property

$$x_0 < \hat{x}_1 < x_1 < \cdots < x_{n-1} < \hat{x}_n < x_n. \tag{3}$$

It is clear that $2n+1$ arbitrary, pairwise distinct given nodes can always be arranged in this way. Given nodal values $(\hat{f}_i)$ at the nodes $(\hat{x}_i)$, our aim is to enforce the additional interpolation conditions

$$r(\hat{x}_i) = \hat{f}_i, \qquad i = 1, \dots, n. \tag{4}$$

Following [21], we observe that inserting (2) into (4) leads to the conditions

$$\hat{f}_j \sum_{i=0}^{n} \frac{w_i}{\hat{x}_j - x_i} - \sum_{i=0}^{n} \frac{w_i}{\hat{x}_j - x_i} f_i = 0, \qquad j = 1, \dots, n.$$

It is easy to see that they are satisfied by choosing the weight vector (w_i) to lie in the nullspace of the *Löwner matrix*

$$B \in \mathbb{R}^{n\times(n+1)}, \quad B_{k\ell} = \frac{\hat{f}_k - f_\ell}{\hat{x}_k - x_\ell}, \quad k = 1, \dots, n, \ \ell = 0, \dots, n. \tag{5}$$

A nonzero weight vector with this property always exists due to the rectangular shape of B. However, this does not necessarily mean that the interpolation property is satisfied in all $2n+1$ nodes: for instance, individual weights w_i may still be zero. This is related to the issue of *unattainable points*, a general fact of rational interpolation; see [23] for more details. Nevertheless, unattainable points are the exceptional case, and we can usually hope to attain interpolation in all $2n+1$ nodes.

The resulting rational interpolation routine is summarized in Algorithm 1.

3 The BRASIL algorithm

3.1 The algorithmic idea

Theorem 1 states that the best rational approximation r of degree n with zero defect d to a function f has an error which equioscillates in $2n+2$ points $(z_j)_{j=0}^{2n+1}$.

Algorithm 1 Barycentric rational interpolation with degree n in $2n+1$ nodes.

```
function INTERPOLATE($f \in C[a,b]$, $(z_0,\dots,z_{2n})$)
    arrange the nodes $(z_k)$ into the vectors $(x_i)_{i=0}^n$ and $(\hat{x}_i)_{i=1}^n$ as in (3)
    evaluate $f_i = f(x_i)$, $\hat{f}_i = f(\hat{x}_i)$
    compute the Löwner matrix $B \in \mathbb{R}^{n\times(n+1)}$ as in (5)
    compute $(w_i)_{i=0}^n$ as a nonzero vector in the nullspace of $B$
    return $r$ from (2) with nodes $(x_i)$, values $(f_i)$, and weights $(w_i)$
end function
```

Due to continuity, the error must attain zero between each pair of neighboring points (z_j, z_{j+1}). This means that there must exist at least $2n+1$ points $(x_j)_{j=0}^{2n}$ in the interior of (a,b) where the best rational approximation interpolates the function itself,

$$r(x_j) = f(x_j), \qquad j = 0,\dots,2n. \tag{6}$$

This observation has also been exploited theoretically; see Stahl [24] for an application of this idea to the case where $f(x) = x^\alpha$ in $[0,1]$. Note also that this implies that for the particular choice of interpolation nodes $(x_0,\dots,x_{2n})$, the above rational interpolation problem does not have any unattainable points as described in Section 2.2, and thus Algorithm 1 will successfully compute an interpolant through all nodes in this case.

Refer again to Figure 1 for an example, where $n = 2$ and there exist 6 nodes of equioscillation (orange stars) and 5 nodes of interpolation (green dots).

To take another point of view, assume that we take $2n+1$ arbitrary nodes $x_0 < \dots < x_{2n}$ in (a,b) and succeed in computing a rational interpolant r of degree n, $r(x_j) = f(x_j)$, through these nodes. The above considerations make it clear that r is the best rational approximation of degree n if and only if the interval-wise errors,

$$\delta_i := \max_{x\in(x_{i-1},x_i)} |f(x) - r(x)|, \qquad i = 0,\dots,2n+1$$

where we set $x_{-1} = a$, $x_{2n+1} = b$, are all equal. The BRASIL algorithm is based on the simple idea of equilibrating these errors. Starting with a suitable guess for the interpolation nodes (x_j), it performs an iteration by simultaneously rescaling the interval lengths such that intervals where the error δ_i is small are enlarged and intervals where the error δ_i is large are shrunk.

Implicitly, this idea makes the assumption that the error of the rational interpolant varies smoothly with the interpolation nodes. Therefore, the proposed method is not well suited to functions which have singularities within the interval (a,b). On the other hand, singularities at the boundary of the interval pose no problem.

Note the distinction to the Remez algorithm: there, the quantities of interest are the nodes where the error is maximal, whereas in BRASIL, we work with the nodes where the error is zero.

3.2 Description of the algorithm

A complete description of BRASIL is given in Algorithm 2. An explanation of the various quantities used in this algorithm is in order.

Algorithm 2 BRASIL algorithm for best rational approximation

```
function BRASIL($f \in C[a,b]$, $n \in \mathbb{N}$, $\varepsilon > 0$, $\sigma_{\max} \in (0,1)$, $\tau > 0$)
    initialize interpolation nodes $x_0 < \ldots < x_{2n} \in (a,b)$
    loop
        $r \leftarrow$ INTERPOLATE($f$, $(x_i)_{i=0}^{2n}$)
        compute interval-wise errors ($x_{-1} = a$, $x_{2n+1} = b$)
            $$\delta_i = \max_{x \in (x_{i-1}, x_i)} |f(x) - r(x)|, \qquad i = 0, \ldots, 2n+1$$
        if $\frac{\max_i \delta_i}{\min_i \delta_i} - 1 < \varepsilon$ then
            return $r$
        end if
        compute ($i$ ranges over $\{0, \ldots, 2n+1\}$)
            $\overline{\delta} = \frac{1}{2n+2} \sum_{i=0}^{2n+1} \delta_i$   (mean error)
            $\overline{\gamma} = \max_{i=0,\ldots,2n+1} |\delta_i - \overline{\delta}|$   (max. deviation from mean error)
            $\gamma_i = \frac{\delta_i - \overline{\delta}}{\overline{\gamma}}$   (signed normalized deviation)
            $\sigma = \min\{\sigma_{\max},\ \tau\overline{\gamma}/\overline{\delta}\}$   (step size)
            $c_i = (1-\sigma)^{\gamma_i}$   (interval length correction factor)
            $\ell_i = c_i(x_i - x_{i-1})$   (rescaled interval lengths)
            $\omega = \sum_{i=0}^{2n+1} \ell_i$   (normalization factor)
        update the interpolation nodes:
            $$x_j \leftarrow a + \frac{b-a}{\omega} \sum_{k=0}^{j} \ell_k \ (j = 0, \ldots, 2n)$$
    end loop
end function
```

We first initialize the interpolation nodes (x_j) in a way which will be discussed in Section 3.3. Then, in each iteration we compute the interpolant r by Algorithm 1, as well as the interval-wise errors (δ_i). Due to the interpolation property, $(f - r)(x_i) = 0$ for $i = 0, \ldots, 2n$, and therefore we we know that the local maximum is bracketed between the two endpoints of the interval (except in the first and last intervals). This makes it convenient to use a standard golden-section search (cf. [22]) to find it. Brent's method (described in the same reference) may require fewer iterations to achieve the same accuracy. If only a few correct digits of the best approximation error are required, computing the maximum error by sampling the error function $|f - r|$ over an equispaced mesh of, say, 100 points in (x_{i-1}, x_i) is a quite competitive approach that is also very easy to implement.

If the errors (δ_i) are equilibrated to the desired tolerance ε, we can conclude from Theorem 1 that we have found a rational function that is sufficiently close to the best rational approximation, and the program terminates successfully.

Otherwise, we compute factors $c_i = (1 - \sigma)^{\gamma_i}$ by which the length of the i-th interval will be scaled. Here $\sigma \in (0, 1)$ is an adaptively chosen step size and $\gamma_i \in [-1, 1]$ is a normalized deviation which is negative if the error δ_i is smaller than the mean error $\overline{\delta}$ and positive if δ_i is larger. Thus, we have that $c_i < 1$ for intervals with large errors and $c_i > 1$ for intervals with small errors. Then the old interval lengths are scaled by these factors, normalized such that they add up to $b - a$ again, and the interpolation nodes updated accordingly.

The choice of the step size $\sigma \in (0, 1)$ is a delicate issue. It may be viewed as the maximum percentage by which an interval may be scaled in a single iteration. Choosing it too large can make the algorithm stagnate without reaching the desired tolerance, whereas choosing it too small slows down convergence significantly. Experimentally, it turns out that a good choice is to set it in dependence of the current error via the formula $\sigma = \min\{\sigma_{\max},\ \tau\overline{\gamma}/\overline{\delta}\}$, where $\sigma_{\max} > 0$ is a maximum step size (usually set to 0.1), τ is a scaling factor (usually set to 0.1), and $\overline{\gamma}/\overline{\delta}$ is the maximum relative deviation from the interval-wise errors to the mean error. Note that $\overline{\gamma}/\overline{\delta}$ tends to 0 as the algorithm converges, and thus so does the step size σ.

Remark 1. It is a simple matter to derive a variant of BRASIL which computes best polynomial approximations of degree n instead. Indeed, only two changes to Algorithm 2 are necessary: the number of interpolation nodes has to be reduced from $2n + 1$ to $n + 1$, and the rational interpolation routine INTERPOLATE has to be replaced by a polynomial barycentric interpolation routine (cf. [3]). The software implementation described in Section 5 offers this variant as an option.

3.3 The initialization method

An issue not yet discussed is the initialization of the interpolation nodes before the algorithm starts. A good choice of these initial nodes has significant benefits both with respect to robustness of the algorithm (i.e., ensuring convergence to the best rational approximation) and convergence speed.

Good initial nodes are obtained by performing a fixed number of iterations (say, $K = 100$) of the following algorithm. The basic structure is similar to the BRASIL algorithm itself, but instead of rescaling the intervals, we simply move, in each iteration, one node from an area of small error to the point where the error is largest, thus forcing the error to zero at that point in the next step.

This procedure is described in Algorithm 3. We first choose the nodes in an arbitrary way, e.g., as the Chebyshev nodes in $[a, b]$ or equispaced (the concrete choice does not appear to matter much). As in the main BRASIL algorithm, we then interpolate through these nodes in each iteration and compute the interval-wise errors (δ_i). In addition, we determine the abscissa $x_{\max}$ where the error is maximal (but shifting it slightly inwards if it happens to lie on the boundary of the interval $[a, b]$; recall that the interpolation nodes always lie in the interior of the interval). We then choose the node x_k which borders on the interval with the smallest error and is farther away from $x_{\max}$ and relocate it to $x_{\max}$. The nodes are then re-sorted and the process repeated.

Algorithm 3 The initialization algorithm for obtaining an initial guess for the interpolation nodes (x_j).

```
function INITIALIZE($f \in C[a,b]$, $n \in \mathbb{N}$, $K \in \mathbb{N}$)
    choose initial nodes $x_0, \dots, x_{2n} \in (a,b)$ (e.g., Chebyshev nodes)
    for $K$ times do
        determine rational interpolant $r$ with $r(x_j) = f(x_j)$, $j = 0, \dots, 2n$
        compute interval-wise errors ($x_{-1} = a$, $x_{2n+1} = b$)

            $$\delta_i = \max_{x \in (x_{i-1}, x_i)} |f(x) - r(x)|, \qquad i = 0, \dots, 2n+1$$

        find $x_{\max} = \arg\max_{x \in [a,b]} |f(x) - r(x)|$
        if $x_{\max} = a$ then
            $x_{\max} \leftarrow \frac{3a + x_0}{4}$
        else if $x_{\max} = b$ then
            $x_{\max} \leftarrow \frac{x_{2n} + 3b}{4}$
        end if
        find $i^* = \arg\min_{i \in \{0,\dots,2n+1\}} \delta_i$ (interval with smallest error)
        choose the node $x_k \in \{x_{i^*-1}, x_{i^*}\} \cap \{x_{j=0}^{2n}\}$ which is farther from $x_{\max}$
        update $x_k \leftarrow x_{\max}$ and re-sort the nodes $(x_j)$
    end for
    return $(x_0, \dots, x_{2n})$
end function
```

This process does not converge to the best rational approximation and usually stagnates rapidly, but tends to be successful in establishing a distribution of the interpolation nodes that asymptotically resembles the correct one. For instance, for a function f with a singularity at the left side a of the interval, it will cluster the nodes toward that end of the interval.

3.4 Computational complexity

The main computational effort during one iteration of BRASIL is spent in computing the new interpolant r as well as the error maxima (δ_i); the remaining steps have negligible cost.

In computing the interpolant r via Algorithm 1, the main effort is in finding the weight vector in the nullspace of $B \in \mathbb{R}^{n \times (n+1)}$. We use the singular value decomposition (SVD) to do this, which has computational complexity $\mathcal{O}(n^3)$ [13].

Referring to the barycentric formula (2), it is easy to see that evaluating r at a given point x requires $\mathcal{O}(n)$ operations. Using golden-section search with k_{GS} iterations for one interval (x_{i-1}, x_i) to determine the maximum error δ_i requires $\mathcal{O}(k_{\text{GS}})$ evaluations of r. Since there are $\mathcal{O}(n)$ intervals, we arrive at the total cost of $\mathcal{O}(k_{\text{GS}} n^2)$ operations for computing the error maxima $(\delta_i)_{i=0}^{2n+1}$. If we used sampling with k_{S} points per interval instead of golden-section search to estimate the maxima, we would instead obtain $\mathcal{O}(k_{\text{S}} n^2)$. Golden-section search has significantly better convergence rate than equispaced sampling for determining the maxima and should be preferred unless the required accuracy is low. Usually, a choice in the range $10 \leq k_{\text{GS}} \leq 30$ is more than sufficient.

Thus, assuming that the evaluation of the given function f is $\mathcal{O}(1)$, we can conclude that a single iteration of BRASIL has computational complexity

$$\mathcal{O}(n^3 + k_{\mathrm{GS}} n^2).$$

Note however that dense linear algebra routines are exceedingly well optimized on modern hardware, and thus the implicit constant in front of the cubic term may be much smaller than the one in front of the quadratic term. In the software implementation described in Section 5, finding the maxima is usually the dominant cost.

A crucial factor in determining the overall computational cost is of course the total number of iterations required to bring the deviation below the tolerance ε. We discuss the convergence rate, as well as ways to improve it, in Section 4. Although the BRASIL algorithm as given converges only with linear rate in practice, the low computational costs per iteration make it competitive with the rational Remez algorithm [12], which exhibits local quadratic convergence but requires more costly computations per iteration. In particular, BRASIL seems much more attractive than the differential correction algorithm [8, 12], which too converges only linearly but requires the solution of a linear programming problem at each step.

4 Accelerating convergence of BRASIL

4.1 Formulation as a fixed-point iteration

The structure of the BRASIL algorithm can be viewed as a fixed-point iteration. Indeed, denoting the vector of interpolation nodes at iteration k as $\mathbf{x}^k = (x_0^k, \dots, x_{2n}^k)$, we obtain a new set of nodes by the nonlinear map

$$\mathbf{x}^{k+1} = \Phi(\mathbf{x}^k), \tag{7}$$

where the operator $\Phi : \mathbb{R}^{2n+1} \to \mathbb{R}^{2n+1}$ denotes the action of one pass through the main loop in Algorithm 2. In particular, Φ does not depend on k. Here and in the following we implicitly assume that no unattainable points occur in the rational interpolation problems (cf. Section 2.2).

Let

$$\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^{2n+1} : a < x_0 < \ldots < x_{2n} < b\}$$

denote the set of admissible nodes. The following result shows that the interpolation nodes $\mathbf{x}^*$ associated with the best rational approximation are a fixed point of Φ in $\mathcal{X}$, and any fixed point yields the best rational approximation.

Theorem 2. *1. The operator Φ has the mapping properties*

$$\Phi : \mathcal{X} \to \mathcal{X}.$$

2. Assume zero defect. The interpolation nodes $\mathbf{x}^* \in \mathcal{X}$ associated with the best rational approximation according to (6) are a fixed point of Φ.

3. If the defect is zero and some $\mathbf{x} \in \mathcal{X}$ satisfies

$$\Phi(\mathbf{x}) = \mathbf{x},$$

then $r[\mathbf{x}] = r[\mathbf{x}^*] = r^*$, where $r[\mathbf{x}]$ denotes the rational interpolant with $r(x_i) = f(x_i)$ for $i = 0, \dots, 2n$, and r^* is the unique best rational approximation to f.

Proof. 1. Let $\mathbf{x} \in \mathcal{X}$ and $\hat{\mathbf{x}} := \Phi(\mathbf{x})$. In the notation of Algorithm 2, we have for all $i = 0, \dots, 2n+1$ that $c_i \geq (1 - \sigma_{\max})^{\gamma_i} > 0$. Hence $\ell_i > 0$ and $\omega > 0$, and it follows $\hat{x}_i - \hat{x}_{i-1} = \frac{b-a}{\omega}\ell_i > 0$. That all nodes are contained in (a, b) is a consequence of the normalization by ω.

2. Consider the application $\Phi(\mathbf{x}^*)$. Since the interval-wise error equioscillates according to Theorem 1, we have $\overline{\delta} = \delta_i$ for all i, the step size $\sigma = 0$, and the correction factors $c_i = 1$. Hence, $\ell_i = x_i - x_{i-1}$ and $\Phi(\mathbf{x}^*) = \mathbf{x}^*$.

3. Let $\hat{\mathbf{x}} = \Phi(\mathbf{x})$. From the update formula, we have

$$\hat{x}_{i-1} - \hat{x}_i = \frac{b-a}{\omega} c_i (x_i - x_{i-1}) \qquad \forall i = 0, \dots, 2n+1,$$

where $\hat{x}_{-1} = x_{-1} = a$ and $\hat{x}_{2n+1} = x_{2n+1} = b$. If $\hat{\mathbf{x}} = \mathbf{x}$, it follows that

$$\frac{\omega}{b-a} = c_i = (1-\sigma)^{\gamma_i} \qquad \forall i = 0, \dots, 2n+1,$$

and hence γ_i and then also δ_i must be constant. But this means that the error $|f - r[\mathbf{x}]|$ equioscillates in at least $2n+2$ extreme points, and therefore $r[\mathbf{x}]$, the rational interpolant through $\mathbf{x}$, must be the unique best rational approximation by Theorem 1. □

Remark 2. To conclude uniqueness of the fixed point, i.e., $\Phi(\mathbf{x}) = \mathbf{x} \implies \mathbf{x} = \mathbf{x}^*$, we would need to establish that the interpolation nodes $\mathbf{x}^*$ resulting in r^* are unique. In other words, we would need to exclude the case that $f - r^*$ has additional zeros, either because it equioscillates in more than $2n+2$ nodes or because there exist additional zeros between two consecutive equioscillation nodes. Results on the exact number of extreme points for best rational approximations to the function x^α are given by Stahl [24]. Also in practice we observe in all tested cases that the error function has exactly $2n+1$ zeros and thus $\mathbf{x}^*$ is unique.

In any case, the third statement of the above theorem makes it clear that any fixed point $\mathbf{x}$ will yield the unique best rational approximation $r[\mathbf{x}] = r^*$, and therefore uniqueness of the fixed point itself is a lesser concern.

The BRASIL fixed point iteration converges in practice with a linear rate, that is, we observe convergence of the form

$$|\mathbf{x}^k - \Phi(\mathbf{x}^k)| \leq C\rho^k, \qquad k = 0, 1, 2, \dots$$

The observed rate $\rho \in (0, 1)$ depends essentially on the function f to be approximated, the degree n, and the step factor parameter τ. In Table 1 we compare estimated convergence rates ρ for varying functions f and degrees n. We observe that ρ depends only very weakly on n, but strongly on the function f. Whereas the approximations for $x^{0.5}$ and $x^{0.75}$ converge rather quickly, the method requires roughly 1200 iterations to reduce the residual from 10^{-2} to 10^{-11} in the case $f(x) = x^{0.1}$. The deviation from equioscillation $\frac{\max_i \delta_i}{\min_i \delta_i} - 1$, which we use for the stopping criterion, converges at a similar rate as the fixed-point residual in all examples.

n	α	ρ	α	ρ	α	ρ
10	0.1	0.980	0.5	0.920	0.75	0.904
20		0.980		0.925		0.891
30		0.980		0.928		0.897
40		0.981		0.927		0.904

Table 1: Estimated convergence rates for the functions x^α in $[0, 1]$ with varying rational degree n and exponent α. We used $\tau = 0.1$ throughout.

4.2 Improving convergence via Anderson acceleration

Each iteration of BRASIL is quite fast and thus the overall computation times are good even with the high iteration numbers reported in the previous subsection. The timings given in the numerical examples in Section 6 confirm this. Nevertheless, accelerating the convergence could further reduce the computation times. Increasing the step factor parameter τ (which was set at 0.1 for all examples) can achieve this, but it may also spoil the robustness of the method for more difficult problems since choosing τ too large results in lack of convergence. Instead, we consider here a general method for accelerating fixed point iterations which is known as *Anderson acceleration* or *Anderson mixing*. Originating from the work of D. G. Anderson [2] in 1965, it has only relatively recently attracted wider attention in the numerical analysis community as a general-purpose acceleration method [11, 29, 19, 6, 25]. It has been pointed out that in the linear case, Anderson acceleration is essentially equivalent to GMRES [29], whereas in the nonlinear case it can be viewed as a generalized Broyden method [11].

Let $F(\mathbf{x}) := \mathbf{x} - \Phi(\mathbf{x})$ denote the non-fixed point formulation of the nonlinear problem such that $F(\mathbf{x}^*) = 0$. The Anderson acceleration procedure of the fixed-point iteration (7) proceeds as follows. We choose a (usually small) integer parameter $m \in \mathbb{N}$ which describes the number of previous iterates to take into account for acceleration. Having computed previous iterates $\mathbf{x}^0, \ldots, \mathbf{x}^k$, $k \geq m$, we find the real parameters (α_j) which minimize

$$\min_{\sum_{j=0}^m \alpha_j = 1} \left| \sum_{j=0}^m \alpha_j F(\mathbf{x}^{k-j}) \right|^2$$

in the Euclidean norm and set

$$\mathbf{x}^{k+1} = \sum_{j=0}^m \alpha_j \Phi(\mathbf{x}^{k-j}) = \sum_{j=0}^m \alpha_j (\mathbf{x}^{k-j} - F(\mathbf{x}^{k-j})).$$

For the first few iterations where $k < m$, we simply perform the same procedure with reduced order $m_k = \min\{m, k\}$. In particular, Anderson acceleration with $m = 0$ is just fixed-point iteration, and therefore $\mathbf{x}^1 = \Phi(\mathbf{x}^0)$ is a standard fixed-point step.

In practice, the minimization problem is reformulated as an unconstrained least-squares problem and solved efficiently using a QR factorization of a $(2n + 1) \times m$ matrix (where $2n + 1$ is the length of each vector $\mathbf{x}^k$). The factorization can be updated, rather than recomputed, in each iteration in order to further save on computational costs; see, e.g., [19, 29] for details.

In our particular application, we must take care that the nodes $\mathbf{x}^k$ remain in the feasible set $\mathcal{X}$. We achieve this by damping the Anderson acceleration by interpolating between a standard fixed-point step (which is guaranteed to maintain the feasibility of the vector due to Theorem 2) and the accelerated vector,

$$\mathbf{x}^{k+1} = (1-\mu)\Phi(\mathbf{x}^k) + \mu \sum_{j=0}^{m} \alpha_j \Phi(\mathbf{x}^{k-j})$$

with a damping parameter $\mu \in [0,1]$. For μ sufficiently small, we obtain $\mathbf{x}^{k+1} \in \mathcal{X}$ since $\mathcal{X}$ is an open set. Note that this damping is different from the *mixing parameter* β usually described in the literature, which modifies Anderson acceleration according to

$$\mathbf{x}^{k+1} = (1-\beta)\sum_{j=0}^{m} \alpha_j \mathbf{x}^{k-j} + \beta \sum_{j=0}^{m} \alpha_j \Phi(\mathbf{x}^{k-j}) = \sum_{j=0}^{m} \alpha_j (\mathbf{x}^{k-j} - \beta F(\mathbf{x}^{k-j})).$$

A further improvement is obtained as follows. We first perform a number of steps of the standard fixed-point iteration to obtain a good estimate $\widetilde{\mathbf{x}}$ for the location of the correct interpolation nodes. We then define the rescaled fixed point iteration

$$\widetilde{\Phi}(\mathbf{y}) := \Phi(\mathbf{y} * \widetilde{\mathbf{x}})/\widetilde{\mathbf{x}}, \qquad \mathbf{y}^0 = (1,\ldots,1), \tag{8}$$

where the vector multiplication and division are to be understood elementwise. This ensures that the components of the new fixed point $\mathbf{y}^*$ have roughly equal order of magnitude, whereas the components of the original fixed point $\mathbf{x}^*$ often vary by many orders of magnitude (cf. Figure 3 for an example). Whereas the results of the standard fixed-point iteration are not changed by passing to the rescaled formulation as one easily sees, the results using Anderson acceleration are significantly more robust using this transformation.

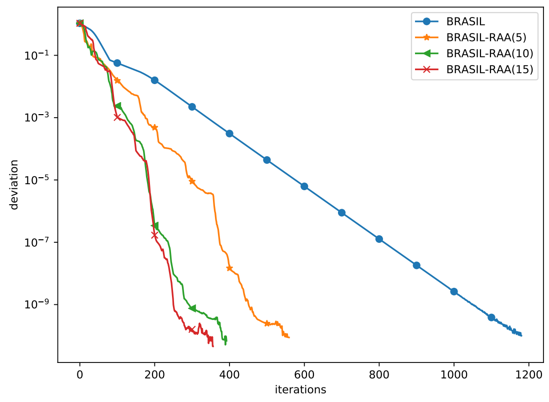
The robustness is further increased by periodically (say, after every 50 iterations) updating the scaling vector $\widetilde{\mathbf{x}}$ by multiplying it with the current iterate $\mathbf{y}^k$ and resetting the iterate to a constant vector of ones. At this point, the Anderson acceleration should also be restarted as the old iterates are no longer accurate for the rescaled problem. We can consider this procedure as a general form of rescaled and restarted Anderson acceleration (RAA). It turns out that the successive rescaling and restarting significantly improves the robustness and convergence speed of Anderson acceleration for our use case of accelerating BRASIL. The same may be true for other applications of Anderson acceleration where the components of the solution vary significantly in order of magnitude.

4.3 Results

We consider best rational approximation of the function $f(x) = x^{0.1}$ since the results of Subsection 4.1 have shown that this is the most difficult of the three tested examples, and we wish to compute the best rational approximation of degree $n = 40$ to a deviation tolerance of $\varepsilon = 10^{-10}$.

As discussed above, we initialize the problem by performing 100 steps of the initialization method, Algorithm 3, and then another 100 steps of standard BRASIL fixed-point iteration, yielding an initial guess $\widetilde{\mathbf{x}}$ for the location of the interpolation nodes, which we use to define the rescaled problem (8).

We then perform 50 steps of Anderson-accelerated fixed-point iteration for $\widetilde{\Phi}$ with a given order m. Unless the desired tolerance has been reached, we use the result to update the scaling, $\widetilde{\mathbf{x}} \leftarrow \widetilde{\mathbf{x}} * \mathbf{y}^{50}$, and restart the iteration with a new starting vector $\mathbf{y}^0 = (1, \dots, 1)$, again iterating for 50 steps, and so on.



m	μ	iter	time
0	n/a	1181	8.6 s
5	0.3	560	4.4 s
10	0.3	393	3.1 s
15	0.25	357	2.7 s

Figure 2: BRASIL with rescaled and restarted Anderson acceleration (BRASIL-RAA). We compare AA order $m = 0$ (fixed point iteration) and $m = 5, 10, 15$. Left: deviation plotted over number of iterations. Right: table displaying the order m, the chosen damping parameter μ, the number of iterations to reduce the deviation below 10^{-10}, and the computation time.

The results of this BRASIL-RAA(m) (BRASIL with rescaled and restarted Anderson acceleration of order m) method are shown in Figure 2. We observe that increasing the order m of Anderson acceleration significantly reduces the number of iterations, but with diminishing returns past a certain point. The computation times show that the added overhead of performing Anderson acceleration is essentially negligible, and the reduction in iterations translates directly into a corresponding reduction in the time. The used implementation did not update the QR factorization, but recomputed it from scratch in each iteration.

Overall, we achieve a reduction of the computation time by a factor of about 3.2x by using RAA(15).

5 Software implementation

A software implementation of the BRASIL algorithm is contained in the `baryrat` open-source Python package for barycentric rational approximation and interpolation, which is developed by the author[1]. It is available on the Python Package Index and therefore can be easily installed on any existing Python 3 distribution using the command `pip install baryrat`. The package relies on the `numpy` and `scipy` packages for fast linear algebra routines, as well as optionally the `mpmath` package for extended-precision arithmetic. Some simple examples of how to use the package can be found on the author's software homepage[2]. A one-line example of how to compute a best rational approximation of degree 12

[1]`https://github.com/c-f-h/baryrat`

[2]`https://people.ricam.oeaw.ac.at/c.hofreither/software/` – see `https://orcid.org/0000-0002-6616-5081` if link is out of date.

to the function $x^{0.5}$ in the interval $[0, 1]$ is given as follows:

```
from baryrat import brasil
r = brasil(lambda x: x**0.5, [0,1], 12)
```

The barycentric rational function object `r` returned by the `brasil` function can be evaluated at arbitrary points and also allows computation of the poles, residues, and zeros, optionally with extended precision by using the `mpmath` arbitrary-precision package.

The implementation of BRASIL in `baryrat` supports both sampling and golden-section search for determining the local maxima δ_i. This is controlled by the `npi=<n>` keyword argument, which selects sampling in `n` nodes if `n` is positive and golden-section search using `-n` iterations if negative.

The interested reader is referred to the package documentation[3] for further details and additional options.

At the time of writing, `baryrat` v1.2.0 does not include the Anderson acceleration method described in Section 4. The used Python implementation of Anderson acceleration can be found on the author's software homepage listed above.

6 Numerical results

In this section, we discuss several numerical experiments with the goal of validating the correctness of the BRASIL algorithm, demonstrating its performance, and highlighting its robustness in comparison to other methods. All tests were performed in a Linux environment using the Python implementation of BRASIL described in Section 5. The used hardware was a laptop with an AMD Ryzen 5 3500U CPU as well as a workstation with an Intel Xeon W3680 CPU.

Throughout all examples, we have left the parameters $\sigma_{\max} = 0.1$, $\tau = 0.1$ at their standard settings, demonstrating the robustness of the method with respect to these parameters.

6.1 Comparison with the results of Varga et al.

In order to confirm the correctness of the results computed with the BRASIL algorithm, we compare them to the published best-approximation errors from [28, 27]. In these works, best rational approximation errors for the functions $\sqrt{x}$ and x^α, respectively, in the interval $[0, 1]$ were computed using an extended-precision implementation of the rational Remez algorithm to at least 200 significant digits.

To achieve high accuracy, we run the BRASIL algorithm with a very small tolerance of $\varepsilon = 10^{-11}$ and perform 30 steps of golden-section search to determine the local error maxima. Note that the desired tolerance could not be achieved for some cases with higher degree due to numerical error, in which case the algorithm was terminated after 1500 iterations. Anderson acceleration was not used for this test.

The results for approximation of $\sqrt{x}$ are shown in Table 2, and for x^α with varying choices of α in Table 3. Correct significant digits according to Varga

[3]`https://baryrat.readthedocs.io/en/latest/`

n	time	BRASIL error	error from Varga et al. [28]
1	0.947 s	**4.3689012692**2871e-02	4.3689012692076361570855971e-02
2	1.01 s	**8.501484704**11732e-03	8.5014847040738294902974113e-03
3	1.10 s	**2.2821060097**3698e-03	2.2821060097252594879063105e-03
4	1.00 s	**7.3656361403**4811e-04	7.3656361403070305616249126e-04
5	1.12 s	**2.6895706008**6703e-04	2.6895706008518350996178760e-04
6	1.21 s	**1.0747116229**5158e-04	1.0747116229451284948608235e-04
7	1.28 s	**4.6036592662**8982e-05	4.6036592662634959571292708e-05
8	1.18 s	**2.0851586406**4118e-05	2.0851586406330327171110359e-05
9	1.57 s	**9.8893346452**9662e-06	9.8893346452814243884404320e-06
10	2.31 s	**4.8759575126**6778e-06	4.8759575126319132435883035e-06
11	3.42 s	**2.4855902684**9726e-06	2.4855902684782111169206258e-06
12	3.61 s	**1.3043775913**8236e-06	1.3043775913430736526687704e-06
13	3.98 s	**7.02231997**995462e-07	7.0223199787397756951998002e-07
14	3.73 s	**3.86755771**630831e-07	3.8675577147259020291010816e-07
15	3.70 s	**2.17398782**198508e-07	2.1739878201697943205320496e-07
16	3.78 s	**1.24477088**414565e-07	1.2447708820895071928214596e-07
17	3.93 s	**7.2478633**9834083e-08	7.2478633767555369698557389e-08
18	3.97 s	**4.2854645**7209578e-08	4.2854645582735082156977870e-08
19	4.03 s	**2.5698967**8426578e-08	2.5698967632180816149049674e-08
20	4.12 s	**1.5613288**7729754e-08	1.5613288569948668163944414e-08
21	4.29 s	**9.6011226**7467395e-09	9.6011226128422364808987184e-09
22	4.37 s	**5.970823**61331047e-09	5.9708233987055580552986137e-09
23	4.50 s	**3.752381**51661961e-09	3.7523813816413163690864502e-09
24	4.51 s	**2.381499**77516144e-09	2.3814996907217830892279694e-09
25	4.57 s	**1.525473**41425446e-09	1.5254732895109793748147207e-09
26	4.79 s	**9.85676**429365867e-10	9.8567633494963529958137413e-10
27	4.86 s	**6.4213**6011030914e-10	6.4213580507266246923653248e-10
28	4.89 s	**4.21588**763899194e-10	4.2158848429927145758285061e-10
29	4.95 s	**2.78832**512634608e-10	2.7883241651339275411060214e-10
30	5.09 s	**1.85707**338395957e-10	1.8570720011628217953125707e-10
31	5.15 s	**1.24507**959498033e-10	1.2450783250744235910902360e-10
32	5.22 s	**8.400**62464035896e-11	8.4005997557762786343216049e-11
33	6.71 s	**5.7022**3868123776e-11	5.7022115757288620263774447e-11
34	6.48 s	**3.8929**7483138762e-11	3.8929505815993459443909823e-11
35	6.65 s	**2.6724**5114926595e-11	2.6724435566456537363975894e-11
36	6.71 s	**1.844**31359073756e-11	1.8442995092525441602503777e-11
37	6.89 s	**1.2792**6558235458e-11	1.2792448409247089881993010e-11
38	7.01 s	**8.916**53417767202e-12	8.9163582949186860871201939e-12
39	7.16 s	**6.243**94980164311e-12	6.2438281549962812624730424e-12
40	7.25 s	**4.392**15330771958e-12	4.3920484091817861898391037e-12

Table 2: Results for the BRASIL algorithm for the best rational approximation of $x^{1/2}$ in $[0, 1]$. Columns: degree of the rational approximation, computation time, obtained error of the approximation computed with BRASIL, high-precision error reported in Varga et al. [28]. Matching significant digits are displayed in bold.

α	n	time	BRASIL error	error from Varga et al. [27]
1/8	5	3.34 s	**1.13217759654**848e-02	1.13217759654301431474e-02
	10	3.96 s	**1.49835131984**965e-03	1.49835131984212814870e-03
	15	4.47 s	**3.13680880237**8988e-04	3.13680880237474018841e-04
	20	4.97 s	**8.36130293638**017e-05	8.36130293633923681498e-05
	25	5.93 s	**2.60369628069**693e-05	2.60369628068729899861e-05
	30	7.88 s	**9.058**76319928778e-06	9.05888497658656444007e-06
1/4	5	1.84 s	**2.73477892547**939e-03	2.73477892546592632673e-03
	10	2.18 s	**1.61000182085**681e-04	1.61000182084826634400e-04
	15	2.35 s	**1.786811229**50487e-05	1.78681122949941574509e-05
	20	6.28 s	**2.77649653**213086e-06	2.77649653194622424338e-06
	25	6.90 s	**5.36229110892**350e-07	5.36229110816195252824e-07
	30	7.63 s	**1.20976855**311206e-07	1.20976855182212349779e-07
3/8	5	1.49 s	**8.17554102624**347e-04	8.17554102620063791819e-04
	10	1.49 s	**2.55069505017**713e-05	2.55069505016840959874e-05
	15	5.60 s	**1.729037823**80551e-06	1.72903782372985456924e-06
	20	6.32 s	**1.7693969**1565181e-07	1.76939690978831627329e-07
	25	6.90 s	**2.3627687**9173047e-08	2.36276876897326311568e-08
	30	7.80 s	**3.816213**67839813e-09	3.81621345151530366382e-09
5/8	5	1.03 s	**9.047584360**96580e-05	9.04758436081231726004e-05
	10	5.46 s	**1.010915774**09118e-06	1.01091577400034953766e-06
	15	5.70 s	**3.10847**401152614e-08	3.10847399216645831784e-08
	20	6.20 s	**1.631831**61427938e-09	1.63183149718337342321e-09
	25	6.98 s	**1.20955**134796930e-10	1.20955031570929120091e-10
	30	7.61 s	**1.147**01581566123e-11	1.14699901451277581199e-11
3/4	5	1.05 s	**2.867552082**60766e-05	2.86755208259350891777e-05
	10	5.40 s	**2.05844565**615010e-07	2.05844565554256321726e-07
	15	5.77 s	**4.514385**64308404e-09	4.51438552412578873109e-09
	20	6.38 s	**1.7830**4149223152e-10	1.78303904874895610620e-10
	25	6.88 s	**1.0287**7706353866e-11	1.02875517502182814096e-11
	30	7.82 s	**7.7**8044295657310e-13	7.77898317234545956241e-13
7/8	5	1.1 s	**7.08939**667066425e-06	7.08939067065246124013e-06
	10	5.16 s	**3.3728331**6831360e-08	3.37283314949851048359e-08
	15	5.62 s	**5.4111**7151087178e-10	5.41116988976654311353e-10
	20	6.21 s	**1.6435**1865450385e-11	1.64350070140677763822e-11
	25	6.93 s	**7.52**731210695856e-13	7.52610355198006348346e-13
	30	9.48 s	4.62963001268690e-14	3.61891911270669288013e-14

Table 3: Results for best rational approximation of x^α in $[0, 1]$. Columns: exponent α, degree of the rational approximation, computation time, obtained error of the approximation computed with BRASIL, high-precision error reported in Varga et al. [27]. Matching significant digits are displayed in bold.

et al. [28, 27] are shown in bold. We see that, despite working only with standard double precision, BRASIL can often match the high-precision results to several significant digits. Accuracy usually begins to suffer once the absolute maximum error is smaller than 10^{-12}, which is not surprising given that machine epsilon is roughly 10^{-16} in the IEEE double precision system. For many practical applications, rational approximations with an error of this magnitude are sufficient.

6.2 Results with lower accuracy

In practice, many applications do not require computing the best rational approximation to many significant digits; instead, it is often sufficient to have an approximation which has an error within a few percent of the best approximation. For example, in the application to fractional diffusion problems sketched in Section 1.2, it suffices that the rational approximation error lies below the discretization error of the spatial discretization of the diffusion problem (cf. [20]).

Therefore, we perform a few tests with lower tolerance $\varepsilon = 10^{-4}$ and using sampling with $k_S = 100$ nodes per interval to determine the local maxima. The results are shown in Table 4. The computation times are lower using these settings, and the results are still in good agreement with the high-precision computations from the previous subsection.

n	iter	error	time
5	335	2.735e-03	0.259 s
10	337	1.610e-04	0.425 s
20	327	2.777e-06	0.794 s
40	331	8.568e-09	1.83 s
60	301	1.002e-10	2.96 s
80	479	2.347e-12	7.30 s

n	iter	error	time
5	209	2.868e-05	0.152 s
10	202	2.059e-07	0.242 s
15	204	4.515e-09	0.354 s
20	201	1.783e-10	0.476 s
24	201	1.776e-11	0.572 s
28	254	2.126e-12	0.881 s

Table 4: Results for best rational approximation of $x^{0.25}$ (left) and $x^{0.75}$ (right) in $[0, 1]$ with tolerance $\varepsilon = 10^{-4}$. Columns: degree, number of iterations, error $\|f - r\|$, computation time.

6.3 Comparison to `Chebfun`'s `minimax` and Remez

The current state-of-the-art implementation of a rational Remez algorithm is the `minimax` routine contained in the `Chebfun` package for Matlab [10]. It is based on a barycentric formulation of the Remez algorithm with adaptively chosen support points [12]. Like the tested implementation of BRASIL, it uses only IEEE double-precision arithmetic.

Recent results for best-approximation using a modified Remez algorithm in quadruple-precision arithmetic were given in [14]. The used software is however not available, and therefore we have to rely on the published numbers.

We compare these approaches in terms of the highest achievable degree n such that the method still converges to a desired tolerance ε. For BRASIL and `minimax`, the deviation tolerance $\varepsilon = 10^{-4}$ was used. The used tolerance for the results from [14] was not specified, but the results produced using BRASIL match them to all significant digits given therein. We use the function $f(x) =$

$\frac{x^{1/4}}{1+qx^{1/4}}$ with varying $q \geq 0$ and $x \in [0,1]$ as a test case since it proved the most challenging for the Remez algorithm in [14].

	BRASIL	`minimax` [12]	Remez [14]
$q = 0$	80	11	at least 8
$q = 1$	82	5	at least 8
$q = 100$	97	2	7
$q = 200$	97	1	6
$q = 400$	93	1	6

Table 5: Highest achievable degrees n for best rational approximation of $f(x) = \frac{x^{1/4}}{1+qx^{1/4}}$ in $[0,1]$ with varying q using three different algorithms.

The results are shown in Table 5. BRASIL far outperforms previously published algorithms in terms of the highest achievable degree n of the best rational approximation. Note that "at least 8" for some of the results from [14] means that the authors provided results up to $n = 8$, but did not specify if higher degrees n were still successfully computed. BRASIL again converged again within a few seconds for each of the tested problems.

In the interest of fairness, it should be noted that for other examples, `minimax` is superior to BRASIL. In particular, many of the examples given in [12] are quite challenging and have nonzero defect or interior singularities. At present, BRASIL cannot be used for these functions.

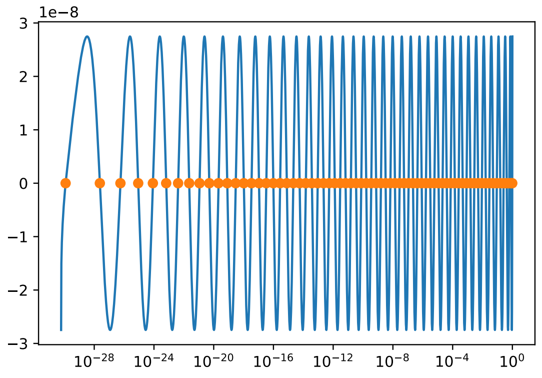


Figure 3: Error $f - r$ of the best rational approximation of degree $n = 35$ for the function $f(x) = x^{1/4}/(1 + x^{1/4})$ in $[0,1]$ computed using BRASIL. The computed interpolation nodes $(x_0, \ldots, x_{70})$ are marked as dots.

Nevertheless, Table 5 confirms the exceptional numerical stability of the BRASIL algorithm. Consider Figure 3, which shows the equioscillation property of the error $f(x) - r(x)$ for $f(x) = x^{1/4}/(1+x^{1/4})$ in $[0,1]$ with a degree $n = 35$ best rational approximation. The 71 computed interpolation nodes (x_j), shown as dots, range in order of magnitude from around 10^{-31} to 1. It seems surprising that this computation could be completed in IEEE double-precision arithmetic, where machine epsilon is roughly 10^{-16}. In essence, this is due to a combination of several factors:

- the backwards stability of the barycentric rational formula [18, 12];
- the fact that the computation of the SVD, which is used to determine the weight vector for the barycentric rational interpolant in Algorithm 1, can be performed in an accurate way [9, 16];
- the fact that the main computations in the BRASIL algorithm, in particular the computation of the error maxima (δ_i), are completely local and thus do not suffer from loss of significant digits since all involved quantities have similar order of magnitude;
- in this particular example, the fact that the singularity lies at $x = 0$. Were it instead at $x = 1$, the interpolation nodes could not be represented to sufficient accuracy since the smallest non-zero difference between 1 and a neighboring double-precision floating point number is on the order of 10^{-16}, whereas the nodes in our example range in magnitude down to the order of 10^{-31}. This poses no problem since IEEE double precision can represent numbers of magnitude down to roughly $2^{-1023} \approx 10^{-308}$ without loss of significant digits (see, e.g., Higham [17]). It may be possible to exploit this observation in different examples by shifting the function such that the singularity lies at $x = 0$, but care must be taken that the shifted function f can be accurately evaluated at these nodes.

References

[1] N.I. Achieser. *Theory of Approximation.* Dover books on advanced mathematics. Dover Publications, 1992. ISBN 9780486671291.

[2] D. G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965. doi:10.1145/321296.321305.

[3] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004. doi:10.1137/s0036144502417715.

[4] 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. Birkhäuser Basel, 2005. doi:10.1007/3-7643-7356-3_3.

[5] D. Braess. *Nonlinear Approximation Theory.* Springer Berlin Heidelberg, 1986. ISBN 978-3-642-64883-0. doi:10.1007/978-3-642-61609-9.

[6] C. Brezinski, M. Redivo-Zaglia, and Y. Saad. Shanks sequence transformations and Anderson acceleration. *SIAM Review*, 60(3):646–669, 2018. doi:10.1137/17m1120725.

[7] A. J. Carpenter, A. Ruttan, and R. S. Varga. Extended numerical computations on the 1/9 conjecture in rational approximation theory. In *Rational Approximation and Interpolation*, pages 383–411. Springer Berlin Heidelberg, 1984. doi:10.1007/bfb0072427.

[8] E. W. Cheney and H. L. Loeb. Two new algorithms for rational approximation. *Numerische Mathematik*, 3(1):72–75, 1961. doi:10.1007/bf01386002.

[9] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. *Linear Algebra and its Applications*, 299(1-3):21–80, 1999. doi:10.1016/s0024-3795(99)00134-2.

[10] T. A Driscoll, N. Hale, and L. N. Trefethen. *Chebfun Guide*. Pafnuty Publications, 2014. URL `http://www.chebfun.org/docs/guide/`.

[11] H. Fang and Y. Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009. doi:10.1002/nla.617.

[12] S.-I. Filip, Y. Nakatsukasa, L. N. Trefethen, and B. Beckermann. Rational minimax approximation via adaptive barycentric representations. *SIAM Journal on Scientific Computing*, 40(4):A2427–A2455, 2018. doi:10.1137/17m1132409.

[13] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, fourth edition, 2012. ISBN 9781421408590.

[14] S. Harizanov, R. Lazarov, S. Margenov, and P. Marinov. The best uniform rational approximation: Applications to solving equations involving fractional powers of elliptic operators, 2019. URL `https://arxiv.org/abs/1910.13865`. arXiv:1910.13865.

[15] S. Harizanov, R. Lazarov, S. Margenov, P. Marinov, and J. Pasciak. Analysis of numerical methods for spectral fractional elliptic equations based on the best uniform rational approximation. *Journal of Computational Physics*, 2020. doi:10.1016/j.jcp.2020.109285. Available online.

[16] N. J. Higham. QR factorization with complete pivoting and accurate computation of the SVD. *Linear Algebra and its Applications*, 309(1-3):153–174, 2000. doi:10.1016/s0024-3795(99)00230-x.

[17] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2002. ISBN 0-89871-521-0.

[18] N. J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA Journal of Numerical Analysis*, 24(4):547–556, 2004. doi:10.1093/imanum/24.4.547.

[19] N. J. Higham and N. Strabić. Anderson acceleration of the alternating projections method for computing the nearest correlation matrix. *Numerical Algorithms*, 72(4):1021–1042, 2015. doi:10.1007/s11075-015-0078-3.

[20] C. Hofreither. A unified view of some numerical methods for fractional diffusion. *Computers & Mathematics with Applications*, 80(2):332–350, 2020. doi:10.1016/j.camwa.2019.07.025.

[21] L. Knockaert. A simple and accurate algorithm for barycentric rational interpolation. *IEEE Signal Processing Letters*, 15:154–157, 2008. doi:10.1109/lsp.2007.913583.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, USA, second edition, 1992.

[23] C. Schneider and W. Werner. Some new aspects of rational interpolation. *Mathematics of Computation*, 47(175):285–285, 1986. doi:10.1090/s0025-5718-1986-0842136-8.

[24] H. R. Stahl. Best uniform rational approximation of x^α on $[0, 1]$. *Acta Mathematica*, 190(2):241–306, 2003. doi:10.1007/bf02392691.

[25] A. Toth and C. T. Kelley. Convergence analysis for Anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015. doi:10.1137/130919398.

[26] L.N. Trefethen. *Approximation Theory and Approximation Practice.* Other Titles in Applied Mathematics. SIAM, 2013. ISBN 9781611972405.

[27] R. S. Varga and A. J. Carpenter. Some numerical results on best uniform rational approximation of x^α on $[0, 1]$. *Numerical Algorithms*, 2(2):171–185, 1992. doi:10.1007/bf02145384.

[28] R. S. Varga, A. Ruttan, and A. D. Carpenter. Numerical results on best uniform rational approximation of $|x|$ on $[-1, 1]$. 74(2):271–290, 1993. doi:10.1070/sm1993v074n02abeh003347.

[29] H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011. doi:10.1137/10078356x.