Johann Radon Institute for
Computational and Applied Mathematics
Austrian Academy of Sciences (ÖAW)

A Newton's method for best uniform polynomial approximation

I. Georgieva, C. Hofreither

RICAM-Report 2021-46

A Newton's method for best uniform polynomial approximation

Irina Georgieva[1], Clemens Hofreither[2]

[1] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Acad. G. Bonchev, Bl. 8, Sofia 1113, Bulgaria, irina@math.bas.bg
[2] Johann Radon Institute for Computational and Applied Mathematics (RICAM), Altenberger Str. 69, 4040 Linz, Austria, clemens.hofreither@ricam.oeaw.ac.at

Abstract. We present a novel algorithm, inspired by the recent BRASIL algorithm [10] for rational approximation, for best uniform polynomial approximation based on a formulation of the problem as a nonlinear system of equations and barycentric interpolation. We use results on derivatives of interpolating polynomials with respect to interpolation nodes to compute the Jacobian matrix. The resulting method is fast and stable, can deal with singularities and exhibits superlinear convergence in a neighborhood of the solution.

1 Introduction

Motivated by applications in solving fractional diffusion problems [6, 7, 9], there has recently been renewed interest in the fast and stable computation of best uniform rational approximations. The classical algorithm towards this end is the rational Remez algorithm (see, e.g., [3, 4, 12]), which is based on the idea of iteratively determining the nodes in which the approximation error equioscillates. Unfortunately, this approach suffers from severe numerical instabilities, which are usually dealt with by using extended precision arithmetic (as in [12]), which in turn significantly slows down the execution of the algorithm. New approaches for stabilizing the Remez algorithm were recently proposed in [11, 5] based on so-called barycentric rational representations.

A new algorithm for computing best rational approximations was recently proposed in [10] based on a different idea: observing that the best approximation must interpolate the function to be approximated in a number of nodes, the new approach is to search for these interpolation nodes, rather than the nodes of equioscillation as is done in the Remez algorithm. The BRASIL algorithm proposed in [10] attempts to do so by a simple heuristic, iteratively applied, which attempts to rescale the lengths of the intervals between the interpolation nodes so as to equilibrate the local errors. Rational interpolation is performed using the barycentric formula. This novel approach appears to enjoy excellent numerical stability. However, as a fixed-point iteration, its convergence rate is only linear, whereas the Remez algorithm converges quadratically in a neighborhood of the exact solution. Therefore a natural question is how to construct an algorithm which seeks the interpolation nodes while converging quadratically.

The present work represents the next step towards such an algorithm. We restrict our attention to the somewhat simpler case of best uniform polynomial approximation (note that the Remez algorithm was originally formulated for polynomial approximation as well). We again treat the interpolation nodes as our unknowns and rewrite the best approximation problem as a system of nonlinear equations, for which we then formulate a Newton's method. In order to compute the Jacobian matrix, we derive expressions for the derivative of an interpolating polynomial with respect to an interpolation node.

The content of the paper is laid out as follows: we first formulate the nonlinear system of equations for best uniform polynomial approximation in Section 2. We compute derivatives of interpolating polynomials in Section 3 and use these results to obtain the Jacobian of the system of equations in Section 4. The complete approximation algorithm is formulated in Section 5, and numerical experiments are presented in Section 6.

2 Best uniform polynomial approximation as a system of nonlinear equations

We seek to determine a polynomial $p \in \mathcal{P}_n$ which best approximates a given function $f \in C[a,b]$ in the maximum norm. It is a classical result that such a best approximation exists, is unique, and that the best approximation error $f-p$ equioscillates in $n+2$ distinct nodes $(y_j)_{j=0}^{n+1}$ in $[a,b]$ (see, e.g., [1]). That is, we have

$$f(y_j) - p(y_j) = \lambda(-1)^j, \qquad j = 0, \ldots, n+1,$$

where $\lambda = \pm\|f-p\|_\infty$. Due to continuity, this implies that there are $n+1$ distinct interpolation nodes $(x_i)_{i=0}^{n}$ in (a,b) with

$$p(x_i) = f(x_i), \qquad i = 0, \ldots, n,$$

interleaving the equioscillation nodes in the sense

$$a \le y_0 < x_0 < y_1 < \ldots < y_n < x_n < y_{n+1} \le b.$$

Let $\mathbf{x} \in \mathcal{X}$ denote a vector of interpolation nodes in the admissible set

$$\mathcal{X} := \{\mathbf{x} \in (a,b)^{n+1} : x_0 < \cdots < x_n\}$$

of nodes in increasing order. We denote by $p[\mathbf{x}] \in \mathcal{P}_n$ the unique polynomial which interpolates f in the nodes $\mathbf{x}$. In each interval (x_{j-1}, x_j), $j = 0, \ldots, n+1$ (letting $x_{-1} = a$ and $x_{n+1} = b$), let

$$y_j := \operatorname*{arg\,max}_{y \in (x_{j-1}, x_j)} |f(y) - p(y)|, \qquad j = 0, \ldots, n+1,$$

denote the abscissa where the error $|f-p|$ is largest. Denote

$$\Phi(\mathbf{x}) := (f(y_j) - p[\mathbf{x}](y_j))_{j=0}^{n+1}, \qquad \mathbf{w} := ((-1)^j)_{j=0}^{n+1}.$$

Theorem 1. *If there exists $\lambda \in \mathbb{R}$ such that*

$$F(\mathbf{x}, \lambda) := \Phi(\mathbf{x}) - \lambda \mathbf{w} = 0, \qquad F : \mathbb{R}^{n+2} \to \mathbb{R}^{n+2}, \tag{1}$$

then $p[\mathbf{x}]$ is the best polynomial approximation to f with error $|\lambda| = \|f - p[\mathbf{x}]\|_\infty$.

Proof. By definition, $\|f - p[\mathbf{x}]\|_\infty = \max_{j=0}^{n+1} |f(y_j) - p[\mathbf{x}](y_j)| = |\lambda|$, and thus the error $f - p[\mathbf{x}]$ equioscillates in $(y_j)_{j=0}^{n+1}$. □

The above result shows that we can view finding the best polynomial approximation as solving the nonlinear equation (1).

In the following, we propose a Newton's method to solve this equation. Given initial guesses for the nodes $\mathbf{x}^0 \in \mathcal{X}$ and the signed error $\lambda^0 \in \mathbb{R}$, a Newton step for the solution of (1) is given by

$$\mathbf{d}^0 := (\mathbf{d}_\mathbf{x}^0, d_\lambda^0) := -(\nabla F(\mathbf{x}^0, \lambda^0))^{-1} F(\mathbf{x}^0, \lambda^0) \in \mathbb{R}^{n+2}.$$

We must make sure that the interpolation nodes remain within the interval (a, b) and in increasing order, i.e., in the admissible set $\mathcal{X}$. For this purpose, we take a damped step $2^{-k}\mathbf{d}^0$, where k is chosen according to

$$\min\{k \in \mathbb{N}_0 : \mathbf{x}^0 + 2^{-k}\mathbf{d}_\mathbf{x}^0 \in \mathcal{X}\}. \tag{2}$$

Since $\mathbf{x}^0 \in \mathcal{X}$ and $\mathcal{X}$ is an open set, such a choice always exists. The updated iterates are then given by

$$\mathbf{x}^1 := \mathbf{x}^0 + 2^{-k}\mathbf{d}_\mathbf{x}^0, \quad \lambda^1 := \lambda^0 + 2^{-k} d_\lambda^0.$$

The main challenge in realizing this Newton's method is the computation of the Jacobian matrix of F, which we discuss in the following.

3 Derivatives of polynomial interpolants

In this section, we compute the derivatives of interpolating polynomials with respect to the interpolation nodes which are required for forming the Jacobian matrix. Let

$$\pi : \mathcal{X} \times \mathbb{R} \to \mathbb{R}, \quad \pi((\xi_0, \ldots, \xi_n), y) := p[\boldsymbol{\xi}](y)$$

denote the unique polynomial of degree at most n in y which interpolates $f : C[a, b] \to \mathbb{R}$ in the nodes $\boldsymbol{\xi}$. Given $\mathbf{x} \in \mathcal{X}$, we can write

$$\pi(\mathbf{x}, y) = \sum_{k=0}^{n} \ell_k(y) f(x_k)$$

with the Lagrange basis polynomials

$$\ell_k(y) = \frac{\omega_k(y)}{\omega_k(x_k)} = \frac{\omega(y)}{(y - x_k)\omega_k(x_k)}, \qquad k = 0, \ldots, n, \tag{3}$$

where we use the notations

$$\omega(y) = \prod_{k=0}^{n} (y - x_k), \qquad \omega_i(y) = \frac{\omega(y)}{y - x_i}.$$

Theorem 2. *The derivative of the polynomial interpolant with respect to the interpolation node x_i, $i = 0, \dots, n$, is given by*

$$\frac{\partial \pi}{\partial \xi_i}(\mathbf{x}, y) = \sum_{k=0}^{n} f(x_k) \frac{\partial}{\partial x_i} \ell_k(y) + \ell_i(y) f'(x_i) = \ell_i(y) \sum_{k=0}^{n} q_{ik} \tag{4}$$

with

$$q_{ik} = \begin{cases} f'(x_k) - f(x_k) \sum_{m \neq k} \frac{1}{x_k - x_m}, & i = k, \\ f(x_k) \frac{\omega_i(x_i)}{\omega_k(x_k)(x_k - x_i)}, & i \neq k. \end{cases}$$

Proof. By elementary calculations we have

$$\begin{aligned} \frac{\partial}{\partial x_i} \ell_i(y) &= -\ell_i(y) \sum_{m \neq i} \frac{1}{x_i - x_m}, \\ \frac{\partial}{\partial x_i} \ell_k(y) &= \ell_k(y) \frac{y - x_k}{(y - x_i)(x_k - x_i)} = \ell_i(y) \frac{\omega_i(x_i)}{\omega_k(x_k)(x_k - x_i)}, \quad i \neq k. \end{aligned}$$

The statement directly follows from these identities. □

4 Computing the Jacobian

Our aim is to compute the total derivative $\frac{d}{dx_i} \pi(\mathbf{x}, y_j)$, keeping in mind that the local maxima y_j themselves depend on $\mathbf{x}$. In the following we assume sufficient smoothness, in particular, $f \in C^1[a, b]$. We have

$$\frac{d}{dx_i} \pi(\mathbf{x}, y_j) = \frac{\partial \pi}{\partial y}(\mathbf{x}, y_j) \frac{\partial y_j}{\partial x_i} + \frac{\partial \pi}{\partial \xi_i}(\mathbf{x}, y_j),$$

where in the first term the derivative of the interpolating polynomial is taken with respect to the evaluation point y_j, whereas in the second term the derivative is taken only with respect to the interpolation node x_i, but the evaluation point y_j is considered constant.

Making use of this formula, we obtain

$$\frac{\partial \Phi_j}{\partial x_i}(\mathbf{x}) = \frac{\partial y_j}{\partial x_i} \left(f'(y_j) - p[\mathbf{x}]'(y_j) \right) - \frac{\partial \pi}{\partial \xi_i}(\mathbf{x}, y_j).$$

Since the nodes y_j are local extrema of the error $f - p[\mathbf{x}]$, the term $f'(y_j) - p[\mathbf{x}]'(y_j)$ vanishes whenever $y_j \in (a, b)$. On the other hand, if $y_j \in \{a, b\}$, we have that either $\frac{\partial y_j}{\partial x_i}$ or $(f - p[\mathbf{x}])'(y_j)$ is 0 by a duality argument. Thus,

$$\frac{\partial \Phi_j}{\partial x_i}(\mathbf{x}) = -\frac{\partial \pi}{\partial \xi_i}(\mathbf{x}, y_j),$$

meaning that the dependence of the y_j on $\mathbf{x}$ can be ignored while computing the Jacobian matrix. The term on the right-hand side can be computed using (4).

We define the matrix $J(\mathbf{x}) \in \mathbb{R}^{(n+2)\times(n+1)}$ as

$$[J(\mathbf{x})]_{j,i} := -\frac{\partial \pi}{\partial \xi_i}(\mathbf{x}, y_j), \qquad j = 0, \ldots, n+1,\ i = 0, \ldots, n,$$

to be computed by means of formula (4), such that the Jacobian of $F(\mathbf{x}, \lambda)$ is given by

$$\bar{J}(\mathbf{x}, \lambda) := \begin{bmatrix} J(\mathbf{x}) & -\mathbf{w} \end{bmatrix} \in \mathbb{R}^{(n+2)\times(n+2)}. \tag{5}$$

5 The algorithm

The complete method for best uniform polynomial approximation is given in Algorithm 1.

Algorithm 1 Newton's method for best polynomial approximation

function BESTPOLY($f \in C[a,b]$, $n \in \mathbb{N}$, $\varepsilon > 0$)
  set initial nodes $\mathbf{x} \in (a,b)^{n+1}$ to Chebyshev nodes of first kind
  loop
    set $p \leftarrow$ INTERPOLATE$(f, \mathbf{x})$
    compute abscissae of local maxima

$$y_j = \operatorname*{arg\,max}_{y \in (x_{j-1}, x_j)} |f(y) - p(y)|, \qquad j = 0, \ldots, n+1$$

    if $\frac{\max_j |f(y_j) - p(y_j)|}{\min_j |f(y_j) - p(y_j)|} - 1 < \varepsilon$ **then**
      return p
    end if
    if in first iteration: set λ to the mean of the local errors $|f(y_j) - p(y_j)|$
    compute $F(\mathbf{x}, \lambda)$ and Jacobian $\bar{J}(\mathbf{x}, \lambda)$ by (5)
    compute Newton step

$$(\mathbf{d}_\mathbf{x}, d_\lambda) \leftarrow -\bar{J}(\mathbf{x}, \lambda)^{-1} F(\mathbf{x}, \lambda)$$

    determine step size 2^{-k} by rule (2)
    update

$$\mathbf{x} \leftarrow \mathbf{x} + 2^{-k}\mathbf{d}_\mathbf{x}, \quad \lambda \leftarrow \lambda + 2^{-k} d_\lambda$$

  end loop
end function

Some remarks are in order on the implementation of this algorithm:

- The function INTERPOLATE$(f, \mathbf{x})$ computes the polynomial interpolant to f in the nodes $\mathbf{x}$ by means of barycentric Lagrange interpolation [2].
- The local maxima y_j may be computed efficiently by means of a golden section search; see [10] for details.
- An initial guess for the error λ is obtained in the first iteration of the algorithm by taking the mean of the local error maxima.

– Formula (4) requires the first derivative f', which we assume to be specified along with f itself. If it is not available, finite differences could be used.
– For evaluating $\ell_i(y)$ in (4), we use the so-called barycentric form given by the last expression in (3) for reasons of numerical stability [2, 8].
– The algorithm terminates when the equioscillation property of the local maxima is valid up to a user-specified tolerance ε.

6 Numerical examples

We give numerical results for two functions,

$$f_1(x) = \frac{x^{1/4}}{1 + 10x^{1/4}}, \quad x \in [0, 1], \qquad f_2(x) = |x|, \quad x \in [-1, 1].$$

The function f_1 is a challenging example motivated by applications in fractional diffusion; cf. [10]. Using f_2 we demonstrate that the algorithm also works in the case of reduced smoothness; we use the sign function $\operatorname{sign}(x)$ in place of $f'(x)$ in this case. We approximate these functions using Algorithm 1 with varying polynomial degree n. In all cases, the tolerance for the stopping criterion is $\varepsilon = 10^{-10}$. The results were obtained on a laptop with an AMD Ryzen 5 3500U CPU.

Results for $f = f_1$ are shown in Figure 1, and for $f = f_2$ in Figure 2. In both cases, the table on the left shows the degree n, the maximum error $\|f - p\|_\infty$, the needed number of iterations, and the computation time (averaged over several runs). The plot on the right shows the convergence history for one particular run, displaying both the residual $\|F(\mathbf{x}, \lambda)\|$ and the deviation from equioscillation $\frac{\max_j |f(y_j) - p(y_j)|}{\min_j |f(y_j) - p(y_j)|} - 1$ (which is used for the stopping criterion) over the iterations. We observe that both of these error measures behave rather similarly and exhibit superlinear convergence during the final iterations. We also remark that the step sizes 2^{-k} chosen by (2) are always 1 except for a few initial iterations, thus taking full Newton steps. The convergence rates in the maximum norm are of course rather poor since the functions are not analytic and thus the polynomial approximations converge slowly.

Since f_2 is an even function, the best approximating polynomials for degrees $2n$ and $2n + 1$ are identical. Our algorithm requires the use of the odd degree $2n + 1$ in order to compute this solution.

Acknowledgments

This work was supported by the bilateral project KP-06-Austria/8/2019 (WTZ BG 03/2019), funded by Bulgarian National Science Fund and OeAD (Austria). The second author gratefully acknowledges additional support by the Austrian Science Fund (FWF) grant P 33956-NBL.

n	error	iter	time (s)
10	0.02857802	14	0.068
20	0.02472576	19	0.136
30	0.02243189	23	0.248
40	0.02081294	27	0.383
50	0.01957241	31	0.580
60	0.01857363	35	0.823
70	0.01774225	40	1.17

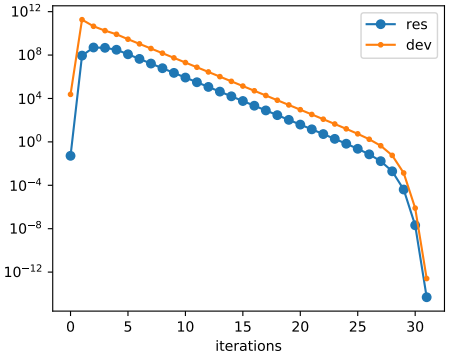

Fig. 1. Results for $f = f_1$. *Left:* Maximum error, number of iterations, and CPU time in dependence of degree n. *Right:* Convergence history for $n = 50$.

n	error	iter	time (s)
5	0.06762090	6	0.024
15	0.01994878	16	0.084
25	0.01166106	12	0.095
35	0.00823581	16	0.167
45	0.00636543	21	0.288
55	0.00518721	25	0.440
65	0.00437698	30	0.654
75	0.00378564	35	0.928

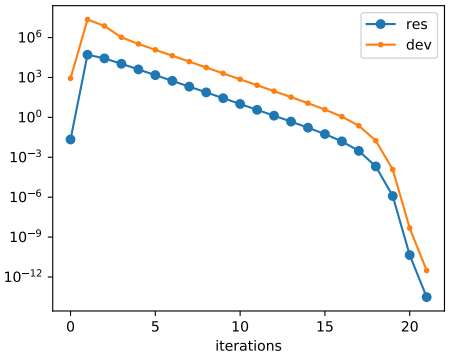

Fig. 2. Results for $f = f_2$. *Left:* Maximum error, number of iterations, and CPU time in dependence of degree n. *Right:* Convergence history for $n = 45$.

Bibliography

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

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

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

[4] 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.

[5] 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.

[6] S. Harizanov, R. Lazarov, S. Margenov, P. Marinov, and Y. Vutov. Optimal solvers for linear systems with fractional powers of sparse SPD matrices. *Numerical Linear Algebra with Applications*, 25(5):e2167, 2018. doi: 10.1002/nla.2167.

[7] 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*, 408:109285, 2020. doi: 10.1016/j.jcp.2020.109285.

[8] 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.

[9] 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.

[10] C. Hofreither. An algorithm for best rational approximation based on barycentric rational interpolation. *Numerical Algorithms*, 2021. doi: 10.1007/s11075-020-01042-0.

[11] A. C. Ioniță. *Lagrange rational interpolation and its applications to approximation of large-scale dynamical systems*. PhD thesis, Rice University, Houston, TY, 2013.

[12] 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.