SIAM J. SCI. COMPUT.
Vol. 43, No. 5, pp. A3372–A3392

THE AAAtrig ALGORITHM FOR RATIONAL APPROXIMATION OF PERIODIC FUNCTIONS*

PETER J. BADDOO†

Abstract. We present an extension of the AAA (adaptive Antoulas–Anderson) algorithm for periodic functions, called "AAAtrig." The algorithm uses the key steps of AAA approximation by (i) representing the approximant in (trigonometric) barycentric form and (ii) selecting the support points greedily. Accordingly, AAAtrig inherits all the favorable characteristics of AAA and is thus extremely flexible and robust, being able to consider quite general sets of sample points in the complex plane. We consider a range of applications with particular emphasis on solving Laplace's equation in periodic domains and compressing periodic conformal maps. These results reproduce the tapered exponential clustering effect observed in other recent studies. The algorithm is implemented in Chebfun.

Key words. rational functions, trigonometric rational functions, AAA algorithm, harmonic functions, conformal mapping

AMS subject classifications. 41A20, 65D15

DOI. 10.1137/20M1359316

1. Introduction.

Given a collection of sample points and data values from a function f, can we stably construct a rational function that approximates f? Can the approximation be improved by exploiting knowledge of the structure of f? The former question has recently been addressed with the AAA (adaptive Antoulas–Anderson) algorithm [37]; the present work addresses the latter question when f is known to be *periodic*. Periodic functions are ubiquitous in every area of science and there is significant scope for applications of rational approximants. For example, partial differential equations (PDEs) are commonly formulated in periodic domains, which reflects the fact that many relevant physical domains are, or can be modeled as, spatially periodic. Periodic rational approximants can thus be used to design effective spectral methods [45] or tackle elliptic PDEs directly using the recently proposed "lightning solvers" [27]. In other areas, discovering periodicities in the time series of a signal is important for analysis and diagnostics in engineering applications [6]. Periodic rational function approximations can compress such signals and uncover underlying structure [49].

The AAA algorithm was introduced as a robust, fast, and flexible method for generating rational approximants from data [37]. Given its effectiveness, it is surprisingly simple: the original article presented a MATLAB implementation in 40 lines of code. Various extensions of AAA have been developed for minimax approximation [38], matrix-valued functions [28], surrogate functions [20], and parametric dynamical systems [40]. In the present work, we adapt the AAA framework for periodic (trigonometric) functions. We call this variation "AAAtrig." A central idea of AAA algorithms is to use the barycentric representation of rational functions [10, 9]. As op-

*Submitted to the journal's Methods and Algorithms for Scientific Computing section August 13, 2020; accepted for publication (in revised form) April 29, 2021; published electronically September 23, 2021.
https://doi.org/10.1137/20M1359316
Funding: This work was supported by EPSRC grant EP/R014604/1 and an EPSRC Doctoral Prize Fellowship from Imperial College London.
†Department of Mathematics, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139 USA (baddoo@mit.edu, https://www.baddoo.co.uk).

posed to representing the rational approximant as a quotient of polynomials $p(z)/q(z)$, AAA represents the approximant as a ratio of partial fractions,

$$r(z) = \sum_{j=1}^{m} \frac{\alpha_j}{z - z_j} \Bigg/ \sum_{j=1}^{m} \frac{\beta_j}{z - z_j}, \tag{1.1}$$

where α_j, β_j, and z_j are constants chosen to optimize the approximation. This barycentric representation is better conditioned than both the aforementioned polynomial quotient and the partial fractions formula used by vector fitting [30]. The trigonometric analogue of the barycentric formula follows quite naturally as

$$r(z) = \sum_{j=1}^{m} \alpha_j \mathrm{cst}\left(\frac{z - z_j}{2}\right) \Bigg/ \sum_{j=1}^{m} \beta_j \mathrm{cst}\left(\frac{z - z_j}{2}\right), \tag{1.2}$$

where cst is a trigonometric function defined in section 2 [31]. Rational approximation can seem an intimidating nonlinear problem due to the unknown support points $\{z_j\}$. The other main step in the AAA algorithm addresses this challenge by selecting the support points $\{z_j\}$ from the sample points greedily, which has the additional effect of mitigating exponential instabilities. Again, this step generalizes naturally to the trigonometric case. Thus, AAAtrig is identical to AAA except the polynomial basis functions in (1.1) have been replaced with trigonometric basis functions (1.2). It follows that AAAtrig inherits all the favorable properties of AAA: it is fast, it can be applied to a range of domains, and it appears to give an optimal distribution of poles and zeros. It is unusual for numerical Froissart doublets (very near poles and zeros) to appear, and when they do, they can be removed using a similar "cleanup" procedure to that of AAA.

Rational function approximation is particularly useful when approximating functions with singularities. This is perhaps illustrated most clearly by the canonical result of Newman [39] who showed that $|x|$ can be approximated with root-exponential accuracy by rational functions, whereas polynomials achieve merely algebraic accuracy. Another appealing feature of rational function approximation is the excellent performance on unbounded domains; the other canonical problem of approximating e^{-x} on $[0, \infty)$ serves as a good example [14]. These observations have recently motivated the application of rational functions to solve PDEs with singularities [27]. These lightning solvers, which belong to the class of methods of fundamental solutions [21], have been successfully applied to solve the Laplace [27], Helmholtz [25], and biharmonic equations. Once the solution is computed, it can be compressed with the AAA algorithm to generate a compact representation of the solution that is extremely fast to evaluate. Typically these compressed solutions can evaluate hundreds of thousands of points in a second. In this paper we generalize these ideas to solve PDEs in periodic domains and compress the solutions using AAAtrig. We apply this theory to the potential flow past a periodic array of obstacles.

Since these lightning solvers can be used to solve Laplace's equation, they can also be used to calculate conformal maps [26, 47]. Indeed, the motivation of the present author to develop AAAtrig was to numerically compress conformal maps found using the recent periodic Schwarz–Christoffel formula [3]. This compression means that the forward and backward maps can be computed with typical speedups of between 10–1,000 times compared to evaluating the actual Schwarz–Christoffel formula, which requires evaluating numerous integrals. We also show how the AAA framework can be used to compress periodic and multivalued conformal maps.

The power of rational approximation for these tasks stems from the exponential clustering of poles and zeros near singularities, as is apparent from Newman's explicit construction [39]. It has recently been proposed that the optimal distribution of these poles and zeros follows a tapered distribution where, on a log scale, their density linearly decays to zero near the singularity [48]. Our findings support this hypothesis and we show that AAAtrig selects tapered poles and zeros in our conformal mapping application.

The remainder of the paper is arranged as follows. In section 2 we introduce the basic ideas of trigonometric interpolation and then explicate the AAAtrig algorithm. In section 3 we consider a number of applications of AAAtrig including comparisons with AAA and fast Fourier transform (FFT)-based interpolation, the removal of Froissart doublets, solutions of Laplace's equation in periodic domains, and compression of conformal maps. We conclude in section 4 with a summary and discussion of future work. The algorithm is available in Chebfun [18] as `aaatrig`.

2. The AAAtrig algorithm. In this section we will present the AAAtrig algorithm. We will first review trigonometric interpolation and justify our form of rational approximant. We then outline the key steps of AAAtrig that determine the support points and weights. Following this, we discuss how far-field behavior at $\pm\,\mathrm{i}\infty$ can be enforced, and explain how to find the poles and zeros once the support points and weights have been determined.

2.1. The form of the approximant. We begin by outlining the general theory of trigonometric polynomial and rational approximation. Thorough summaries of the underlying mathematics of computations with trigonometric functions are available in [50] and [34]. A trigonometric polynomial of degree m with period 2π is a function of the form [32]

$$r(z) = \sum_{k=-m}^{m} c_k \mathrm{e}^{\mathrm{i}kz}. \tag{2.1}$$

Alternatively, r may be expressed in terms of sines and cosines as

$$r(z) = \frac{1}{2}a_0 + \sum_{k=1}^{m} a_k \cos(kz) + b_k \sin(kz). \tag{2.2}$$

Our independent variable is represented here as z, as opposed to the more traditional θ. This choice of notation is used to emphasize the flexibility of AAAtrig: the sample points are not tied to a particular domain such as the real interval $[0, 2\pi)$.

For m function values f_j at sample points z_j, the first form of the barycentric trigonometric interpolation formula is [10]

$$r(z) = \sum_{j=1}^{m} f_j a_j l(z) \mathrm{cst}\left(\frac{z - z_j}{2}\right), \tag{2.3}$$

where l is the trigonometric node polynomial

$$l(z) = \prod_{j=1}^{m} \sin\left(\frac{z - z_j}{2}\right), \tag{2.4}$$

the coefficients are

$$a_j = \prod_{\substack{k \neq j \\ k=0}}^{m} \csc\left(\frac{z_k - z_j}{2}\right), \tag{2.5}$$

and

$$\operatorname{cst}(z) = \begin{cases} \csc(z) & \text{if } m \text{ is odd,} \\ \cot(z) + c & \text{if } m \text{ is even,} \end{cases} \tag{2.6}$$

where c is a constant discussed below. The distinction between odd and even m stems from the odd $(2m+1)$ number of unknowns in (2.1). As such, the interpolation problem for an even number of interpolation points is underdetermined. The problem is usually closed by specifying the form of the highest frequency of oscillation and requiring that the coefficient of $\sin(mz)$ vanishes [31]. This requirement produces $c = \cot(\sum z_j/2)$, and the trigonometric polynomial is called "balanced" [7]. In contrast, we take $c = 0$ in our calculations. The reason for this choice is that we wish to consider arbitrary sample data; obviously there will be a problem if, for example, $\sum z_j = 2\pi k$ for some integer k.

Dividing (2.3) by the corresponding expression for $f \equiv 1$ yields an alternative expression for r:

$$r(z) = \frac{\sum_{j=1}^{m} f_j a_j l(z) \operatorname{cst}\left(\frac{z-z_j}{2}\right)}{\sum_{j=1}^{m} a_j l(z) \operatorname{cst}\left(\frac{z-z_j}{2}\right)} = \frac{\sum_{j=1}^{m} f_j a_j \operatorname{cst}\left(\frac{z-z_j}{2}\right)}{\sum_{j=1}^{m} a_j \operatorname{cst}\left(\frac{z-z_j}{2}\right)}. \tag{2.7}$$

The above expression is the second form of the barycentric trigonometric interpolation formula [46, 41, 7]. For simplicity, we refer to (2.7) as the trigonometric barycentric form for the remainder of the paper. The (polynomial) barycentric form has several advantages over the traditional Lagrange interpolation formula, as outlined in [10]. First, evaluating the Lagrange formula for each z requires $\mathcal{O}(m^2)$ floating point operations (flops), whereas evaluating the barycentric forms (2.7) requires $\mathcal{O}(m)$ flops after the coefficients w_j have been computed. Second, including a new data pair (f_{m+1}, z_{m+1}) requires every computation to be performed again in the Lagrangian formulation; the barycentric form can be updated in $\mathcal{O}(m)$ flops. Higham [33] has proved that the (polynomial) barycentric formula is forward stable, as long as the interpolating points have a small Lebesgue constant. Third, Berrut proved that, under certain conditions, the trigonometric barycentric formula (2.7) is guaranteed to be well-conditioned for interpolation on the unit circle [8].

The barycentric form (2.7) interpolates f regardless of the choice of coefficients a_j. Accordingly, we may replace the a_j with arbitrary weights w_j to obtain

$$r(z) = \sum_{j=1}^{m} f_j w_j \operatorname{cst}\left(\frac{z-z_j}{2}\right) \Bigg/ \sum_{j=1}^{m} w_j \operatorname{cst}\left(\frac{z-z_j}{2}\right). \tag{2.8}$$

The function r is a trigonometric polynomial in the special case $w_j = a_j$. For any other choice of w_j, the function r represents a trigonometric rational function, i.e., the ratio of two trigonometric polynomials. We write the numerator and denominator of (2.8) as n and d, respectively, so that $r(z) = n(z)/d(z)$. The barycentric form is related to

the traditional form of a rational function (i.e., $r(z) = p(z)/q(z)$) by multiplication of the trigonometric node polynomial $l(z)$ in (2.4).

Before introducing the AAAtrig algorithm, we clarify the correspondence between the trigonometric and polynomial barycentric forms. The barycentric form for a (nontrigonometric) rational function interpolating f_j at z_j is [10]

$$r(z) = \sum_{j=1}^{m} \frac{f_j w_j}{z - z_j} \Bigg/ \sum_{j=1}^{m} \frac{w_j}{z - z_j}. \tag{2.9}$$

This is the form used in the original AAA algorithm [37]. The connection between the barycentric forms for trigonometric and polynomial interpolants can be made more transparent by expanding cst from (2.6) in its Mittag–Lefler representation. In particular, substituting

$$\csc(z) = \sum_{k=-\infty}^{\infty} \frac{(-1)^k}{z - k\pi}, \qquad \cot(z) = \sum_{k=-\infty}^{\infty} \frac{1}{z - k\pi} \tag{2.10}$$

into the rational approximant (2.8) yields

$$r(z) = \sum_{k=-\infty}^{\infty} \chi_m^k \left(\sum_{j=1}^{m} \frac{f_j w_j}{z - (z_j - 2\pi k)} \right) \Bigg/ \sum_{k=-\infty}^{\infty} \chi_m^k \left(\sum_{j=1}^{m} \frac{w_j}{z - (z_j - 2\pi k)} \right), \tag{2.11}$$

where $\chi_m = (-1)^m$ accounts for an odd or even number of sample points. As such, it can be seen that a AAAtrig approximant (2.9) can be viewed as a AAA approximant with periodic support points.

2.2. The AAAtrig algorithm. In this section we present the core AAAtrig algorithm. The guiding principles follow directly from the original AAA formulation and only minor changes are required.

The choice of basis functions cst merits a brief discussion. Our problem is distinct from those considered by Henrici [31] and Berrut [7] since our degree of approximation (m) varies with each iteration, whereas they considered interpolants of fixed degree. Accordingly, the definition of cst in (2.6) is not so relevant as the definition changes with each iteration. As such, in the present algorithm, the form of the approximant is specified to be "odd" or "even" by the user. Odd approximants correspond to cst $\equiv$ csc whereas even approximants correspond to cst $\equiv$ cot. Numerical experiments indicate that cot or csc are equally good choices as far as the speed of convergence is concerned.[1] In section 2.3 we show that odd and even interpolants produce different behaviors as $z \to \pm\, \mathrm{i}\infty$.

We consider a sample set $Z \in \mathbb{C}^M$ along with a function $f(z)$ that is defined for all $z \in Z$. Since the underlying function f is periodic, we project the sample points onto the strip $0 \leq \mathbb{R}\mathrm{e}[z] < 2\pi$ by

$$Z \mapsto Z - 2\pi \left\lfloor \mathbb{R}\mathrm{e}\left[\frac{Z}{2\pi} \right] \right\rfloor. \tag{2.12}$$

[1] Alternating expressions (i.e., (2.6)) were also considered but there was no discernible improvement in performance and the implementation was much more complicated.

We now come to the first essential step of the AAA framework. At step m, the next support point is selected "greedily": the support point z_m is chosen to minimize the error between the function values and the approximant evaluated at the remaining support points. In other words,

$$z_m = \underset{z \in Z^{(m-1)}}{\arg\max} \left| f(z) - \frac{n(z)}{d(z)} \right|, \tag{2.13}$$

where $Z^{(m-1)}$ is the set of sample points with all the previous support points removed:

$$Z^{(m-1)} = Z \setminus \{z_1, \ldots, z_{m-1}\} = \{Z_1^{(m-1)}, \ldots, Z_{M-m+1}^{(m-1)}\}. \tag{2.14}$$

Having found the sample point with the maximum residual error and added it to the set of support points, we now seek to calculate the weights $w_1, \ldots, w_m$ that solve the least-squares problem

$$\min \|fd - n\|_{Z^{(m)}} \qquad \text{subject to } \|\boldsymbol{w}\|_2 = 1, \tag{2.15}$$

where $\boldsymbol{w} = [w_1, \ldots, w_m]^T$ and $\|\cdot\|_{Z^{(m)}}$ represents the discrete 2-norm over $Z^{(m)}$. We assume that $m \le M/2$ so that $Z^{(m)}$ contains at least m points.

The minimization problem (2.15) can equivalently be expressed as finding $\boldsymbol{w}$ that solves

$$\min \left\| \boldsymbol{A}^{(m)} \boldsymbol{w} \right\|_2 \qquad \text{subject to } \|\boldsymbol{w}\|_2 = 1, \tag{2.16}$$

where

$$\boldsymbol{A}^{(m)} = \begin{bmatrix} \left(F_1^{(m)} - f_1\right) \operatorname{cst}\left(\dfrac{Z_1^{(m)} - z_1}{2}\right) & \cdots & \left(F_1^{(m)} - f_m\right) \operatorname{cst}\left(\dfrac{Z_1^{(m)} - z_m}{2}\right) \\ \vdots & \ddots & \vdots \\ \left(F_{M-m}^{(m)} - f_1\right) \operatorname{cst}\left(\dfrac{Z_{M-m}^{(m)} - z_1}{2}\right) & \cdots & \left(F_{M-m}^{(m)} - f_m\right) \operatorname{cst}\left(\dfrac{Z_{M-m}^{(m)} - z_m}{2}\right) \end{bmatrix}$$

is the $(M-m) \times m$ trigonometric analogue of the Loewner matrix and $F_j^{(m)} = f(Z_j^{(m)})$ for $j = 1, \ldots, M-m$. It is convenient to introduce the $(M-m) \times m$ trigonometric Cauchy matrix defined by

$$\boldsymbol{C} = \begin{bmatrix} \operatorname{cst}\left(\dfrac{Z_1^{(m)} - z_1}{2}\right) & \cdots & \operatorname{cst}\left(\dfrac{Z_1^{(m)} - z_m}{2}\right) \\ \vdots & \ddots & \vdots \\ \operatorname{cst}\left(\dfrac{Z_{M-m}^{(m)} - z_1}{2}\right) & \cdots & \operatorname{cst}\left(\dfrac{Z_{M-m}^{(m)} - z_m}{2}\right) \end{bmatrix}.$$

Then, $\boldsymbol{A}^{(m)}$ can be formed as $\boldsymbol{A}^{(m)} = \boldsymbol{S}_F\boldsymbol{C} - \boldsymbol{C}\boldsymbol{S}_f$, where

$$\boldsymbol{S}_F = \operatorname{diag}\left(F_1^{(m)}, \ldots, F_{M-m}^{(m)}\right), \qquad \boldsymbol{S}_f = \operatorname{diag}\left(f_1, \ldots, f_m\right), \tag{2.17}$$

and the rational approximant evaluated at $Z^{(m)}$ can be computed as $\boldsymbol{r} = \boldsymbol{N}/\boldsymbol{D}$, where $\boldsymbol{N} = \boldsymbol{C}(\boldsymbol{w}\boldsymbol{f})$ and $\boldsymbol{D} = \boldsymbol{C}\boldsymbol{w}$. Here $(\boldsymbol{w}\boldsymbol{f}) = [f_1 w_1\,, \cdots,\, f_m w_m]^T$ and $\boldsymbol{N}/\boldsymbol{D}$ is evaluated elementwise. The matrices $\boldsymbol{S}_F$ and $\boldsymbol{S}_f$ should be stored in sparse format to accelerate the formation of $\boldsymbol{A}^{(m)}$.

The algorithm terminates if the maximum error $\left\|\boldsymbol{r} - f(Z^{(m)})\right\|_\infty$ is below a given tolerance or if m becomes too large. These conditions can be specified by the user; by default the algorithm will terminate if the relative maximum error is below 10^{-13} or $m > 100$. If the termination criteria are not met then the algorithm proceeds to the next iterate by choosing the next support point greedily as described above.

This concludes the description of the core AAAtrig algorithm. Readers familiar with the original AAA formulation [37] will note that AAAtrig is almost identical to AAA. The only substantial difference arises in the definition of $\boldsymbol{C}$, where we have replaced the polynomial basis with the analogous trigonometric basis.

2.3. Enforcing far-field behavior. In some applications it is useful to specify the behavior of the rational approximant at infinity. In other words, we wish to specify (finite) $f_\infty^\pm$, where

$$f_\infty^\pm = \lim_{z \to \pm \mathrm{i}\infty} r(z). \tag{2.18}$$

Even approximants are of the form (2.29) so r takes the same value as $z \to \pm\,\mathrm{i}\infty$ and

$$f_\infty^\pm = f_\infty = \frac{\sum_{j=1}^m f_j w_j}{\sum_{j=1}^m w_j}. \tag{2.19}$$

The above may be rearranged into the linear constraint $\boldsymbol{g}^*\boldsymbol{w} = 0$, where

$$\boldsymbol{g} = \begin{bmatrix} f_\infty - f_1 & \cdots & f_\infty - f_m \end{bmatrix}^*. \tag{2.20}$$

Thus, the minimization problem (2.16) becomes

$$\min \left\| \boldsymbol{A}^{(m)}\boldsymbol{w} \right\|_2 \qquad \text{subject to } \|\boldsymbol{w}\|_2 = 1, \quad \boldsymbol{g}^*\boldsymbol{w} = 0, \tag{2.21}$$

which can be solved with Lagrange multipliers [24]. Specifically, the solution is $\boldsymbol{w} = \boldsymbol{P}\boldsymbol{y}$, where $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{g}\boldsymbol{g}^*/\left\|\boldsymbol{g}\right\|_2^2$ is the orthogonal projector onto the nullspace of $\boldsymbol{g}^*$ and $\boldsymbol{y}$ is the singular vector of $\boldsymbol{A}^{(m)}\boldsymbol{P}$ with smallest nonzero singular value. Additionally, $\boldsymbol{w}$ should be normalized to have unit norm.

A similar method can be used to enforce far-field behavior in odd approximants. These approximants take the form (2.24) and r attains different values as $z \to \pm\,\mathrm{i}\infty$:

$$f_\infty^\pm = \frac{\sum_{j=1}^m f_j w_j \mathrm{e}^{\mp \mathrm{i} z_j/2}}{\sum_{j=1}^m w_j \mathrm{e}^{\mp \mathrm{i} z_j/2}}. \tag{2.22}$$

The corresponding minimization problem is solved in an analogous way to the even case except $\boldsymbol{g}$ in (2.21) is now replaced with the matrix

$$\boldsymbol{G} = \begin{bmatrix} (f_\infty^+ - f_1)\,\mathrm{e}^{-\mathrm{i}z_1/2} & \cdots & (f_\infty^+ - f_m)\,\mathrm{e}^{-\mathrm{i}z_m/2} \\ (f_\infty^- - f_1)\,\mathrm{e}^{+\mathrm{i}z_1/2} & \cdots & (f_\infty^- - f_m)\,\mathrm{e}^{+\mathrm{i}z_m/2} \end{bmatrix}^*,$$

and the projection matrix becomes $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{G}\boldsymbol{G}^{-}$. Here, $\boldsymbol{G}^{-}$ is any generalized inverse that satisfies $\boldsymbol{G}\boldsymbol{G}^{-}\boldsymbol{G} = \boldsymbol{G}$ and $(\boldsymbol{G}\boldsymbol{G}^{-})^{*} = \boldsymbol{G}\boldsymbol{G}^{-}$. For example, $\boldsymbol{G}^{-}$ could represent the Moore–Penrose pseudoinverse of $\boldsymbol{G}$.

Experiments indicate that this procedure is accurate for tolerances down to around 10^{-9}. For lower tolerances, the solutions for $\boldsymbol{w}$ produce very small numerators and denominators in the fractions in (2.19) and (2.22), which result in cancellation errors. Thus, applications that require higher precision should consider a more careful implementations of the ideas in this section, possibly using the algorithms in [24] and [23].

2.4. Finding the poles and zeros. The poles and zeros of the rational approximant give insight into the AAA/AAAtrig procedure and suggest features of the underlying function that is being approximated. For example, interlaced poles and zeros may indicate that the original function possesses a branch cut. The poles and zeros computed by AAAtrig may also provide useful information for signal classification and diagnostics. Additionally, knowledge of the poles and zeros is also useful in determining whether undesirable Froissart doublets have been generated in the computation, and whether the cleanup procedure is necessary. The core AAAtrig algorithm provides only the support points and weights of the rational approximant. Determining the poles and zeros of the rational approximant requires an extra step, which we detail here.

In what follows, we will make use of the exponential representations of csc and cot:

$$\csc(z) = \frac{2\mathrm{i}\mathrm{e}^{\mathrm{i}z}}{\mathrm{e}^{2\mathrm{i}z} - 1}, \qquad \cot(z) = \mathrm{i}\left(1 + \frac{2}{\mathrm{e}^{2\mathrm{i}z} - 1}\right). \tag{2.23}$$

2.4.1. Odd approximants. In this case, the trigonometric rational approximant is

$$r(z) = \sum_{j=1}^{m} f_j w_j \csc\left(\frac{z - z_j}{2}\right) \Bigg/ \sum_{j=1}^{m} w_j \csc\left(\frac{z - z_j}{2}\right). \tag{2.24}$$

We can manipulate the trigonometric approximant (2.24) into the polynomial approximant (2.9) by a suitable transformation. In particular, inserting the substitutions

$$\hat{z} = \mathrm{e}^{\mathrm{i}z}, \qquad \hat{z}_j = \mathrm{e}^{\mathrm{i}z_j}, \qquad \hat{w}_j = w_j \mathrm{e}^{\mathrm{i}z_j/2}, \tag{2.25}$$

into (2.24) yields

$$r(z) \triangleq R(\hat{z}) = \sum_{j=1}^{m} \frac{f_j \hat{w}_j}{\hat{z} - \hat{z}_j} \Bigg/ \sum_{j=1}^{m} \frac{\hat{w}_j}{\hat{z} - \hat{z}_j}\,, \tag{2.26}$$

where we have used the identity $(2.23)_1$. The poles and zeros may now be found in the same way as the original AAA algorithm [37] using a method originally proposed by Klein [35]. For example, the zeros of R are the eigenvalues of the $(m+1) \times (m+1)$ generalized eigenvalue problem

$$\begin{pmatrix} 0 & f_1\hat{w}_1 & \cdots & f_m\hat{w}_m \\ 1 & \hat{z}_1 & & \\ \vdots & & \ddots & \\ 1 & & & \hat{z}_m \end{pmatrix} \boldsymbol{v} = \lambda \begin{pmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \boldsymbol{v} \tag{2.27}$$

with eigenvectors

$$\boldsymbol{v} = [1,\, 1/(\lambda - \hat{z}_1),\, \ldots\, , 1/(\lambda - \hat{z}_m)]^T\,. \tag{2.28}$$

At least two of the $m+1$ eigenvalues of the system (2.27) are infinite and these are discarded since a type $(m-1, m-1)$ approximant can have, at most, $m-1$ zeros. The poles are the eigenvalues of an identical problem with $f_j\hat{w}_j$ replaced with $\hat{w}_j$. Once the zeros and poles of R have been found, the zeros and poles of r can be found by inverting the substitution (2.25).

2.4.2. Even approximants. Finding the poles and zeros for even approximants follows in a similar manner. In this case, the approximant is

$$r(z) = \sum_{j=1}^{m} f_j w_j \cot\left(\frac{z - z_j}{2}\right) \Bigg/ \sum_{j=1}^{m} w_j \cot\left(\frac{z - z_j}{2}\right). \tag{2.29}$$

Now applying the substitutions

$$\hat{z} = \mathrm{e}^{\mathrm{i}z}, \qquad \hat{z}_j = \mathrm{e}^{\mathrm{i}z_j}, \qquad \hat{w}_j = w_j \mathrm{e}^{\mathrm{i}z_j}, \tag{2.30}$$

with $(2.23)_2$ converts (2.29) to

$$r(z) \triangleq R(\hat{z}) = \left(\sum_{j=1}^{m} \frac{f_j\hat{w}_j}{\hat{z} - \hat{z}_j} + c_n\right) \Bigg/ \left(\sum_{j=1}^{m} \frac{\hat{w}_j}{\hat{z} - \hat{z}_j} + c_d\right), \tag{2.31}$$

where the constants in the numerator and denominator are given by

$$c_n = \frac{1}{2}\sum_{j=1}^{m} f_j w_j, \qquad c_d = \frac{1}{2}\sum_{j=1}^{m} w_j. \tag{2.32}$$

Note that (2.31) is in the original barycentric form (2.9) used in AAA but with additional constants in the numerator and denominator. The poles and zeros of R can be found by solving a generalized eigenvalue problem in a similar vein to the previous section. It is straightforward to verify that the eigenvalues of the generalized eigenvalue problem

$$\begin{pmatrix} c_n & f_1\hat{w}_1 & \cdots & f_m\hat{w}_m \\ 1 & \hat{z}_1 & & \\ \vdots & & \ddots & \\ 1 & & & \hat{z}_m \end{pmatrix} \boldsymbol{v} = \lambda \begin{pmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \boldsymbol{v} \tag{2.33}$$

are the zeros of R, and the eigenvectors are again given by (2.28). The zeros of r are then given by inverting (2.30). Again, the poles of r can be found by solving a similar generalized eigenvalue problem where c_n is replaced by c_d and the $f_j\hat{w}_j$ are replaced by $\hat{w}_j$ in (2.33).

Once the poles and zeros have been found, the residues at each pole can be computed with the standard quotient formula. Then, the approximant can alternatively be expressed in its partial fraction representation as

$$r(z) = \sum_{k=1}^{m-1} \frac{q_k}{2} \cot\left(\frac{z - p_k}{2}\right) + d, \tag{2.34}$$

where $\{p_k\}$ are the poles, q_k is the residue at p_k, and d is a constant. Comparison with the far-field values in section 2.3 shows that, for odd approximants, the residues and constant must satisfy (2.22),

$$d \mp \mathrm{i} \sum_{k=1}^{m-1} \frac{q_k}{2} = \frac{\sum_{j=1}^{m} f_j w_j \mathrm{e}^{\mp \mathrm{i} z_j/2}}{\sum_{j=1}^{m} w_j \mathrm{e}^{\mp \mathrm{i} z_j/2}}, \tag{2.35}$$

whereas for even approximants, (2.19) implies that

$$\sum_{k=1}^{m-1} q_k = 0, \qquad d = \frac{\sum_{j=1}^{m} f_j w_j}{\sum_{j=1}^{m} w_j}. \tag{2.36}$$

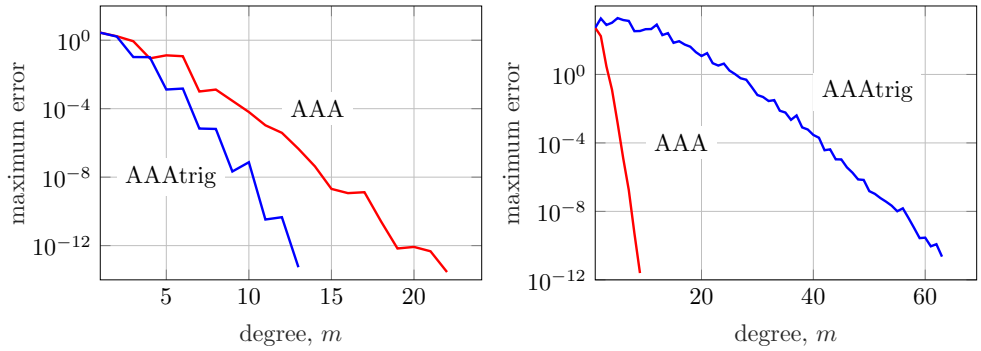
From the above we see that the even approximant is constrained such that its residues sum to zero. In contrast, the odd csc approximant is slightly more general and should therefore normally be favored in calculations. Whilst the partial fraction representation (2.34) is useful for clarifying the structure of the odd and even approximants the conversion between barycentric form and partial fractions can be numerically unstable. For example, to differentiate the approximant one should avoid differentiating (2.34) symbolically and instead use the differentiation matrices in [4].

3. Examples.

We now present a range of examples and applications of AAAtrig.

3.1. Comparison with AAA.

In Figure 1 we compare AAAtrig to the original AAA algorithm for periodic and nonperiodic functions. The functions to be approximated, $\exp(\sin(x))$ in 1(a) and $\exp(x)$ in 1(b), are sampled at 1,000 randomly distributed points in the rectangle $[0, 2\pi] \times [-\mathrm{i}/2, \mathrm{i}/2]$. We plot the maximum error, which is defined over the (discrete) sample set Z and not a continuous domain. As expected, AAAtrig outperforms AAA when approximating a periodic function in 1(a). Since AAAtrig uses a trigonometric barycentric approximant, it can simultaneously reduce the error at each end of the period window. For example, if AAAtrig chooses a support point near 0 then the error near 2π is automatically reduced. Nontrigonometric AAA does not have this built-in structure and thus requires more iterations to



(a) $f(x) = \exp(\sin(x))$ (b) $f(x) = \exp(x)$

FIG. 1. *Comparison of maximum errors for AAAtrig (blue) and AAA (red) for periodic and non-periodic functions. The functions are sampled at* 1,000 *randomly distributed points in a rectangle in the complex plane. Note that the maximum error here refers to the maximum error over the set of discrete sample points* Z.

converge. In contrast AAA substantially outperforms AAAtrig when approximating a nonperiodic function in Figure 1(b). Since the approximant used by AAAtrig is constrained to be periodic, it requires many support points to resolve the nonperiodic behavior at the endpoints of the period window. Note that we are using the discrete L^∞ norm so AAAtrig terminates with accuracy 10^{-11}; with the continuous L^∞ norm we would see that AAAtrig cannot accurately represent a nonperiodic function.

In these examples, the improvement of AAAtrig over AAA is quite modest and the advantage of AAAtrig may not be transparent. The differences become clearer when constructing an approximant that is valid throughout the complex plane; we encounter such examples in sections 3.4 and 3.5.

3.2. Removing Froissart doublets. Occasionally, an application of AAAtrig will return pairs of very close poles and zeros (e.g., separated by 10^{-14}), known as Froissart doublets. These anomalies can be removed using a similar cleanup procedure to that proposed in [37]: after AAAtrig has terminated, poles with residues below a certain tolerance (10^{-13}) are identified and the nearest support points are removed, followed by a final least-squares problem. This approach typically succeeds in drastically reducing the number of poles in the approximant.

The effect of the cleanup procedure is illustrated in Figure 2, which is comparable to Figure 5.2 of [37]. In this example, the function $f(x) = \log(2+\cos(x)^4)$ is sampled at 1,000 roots of unity and approximated with AAAtrig with the tolerance set to zero. (The default relative tolerance is 10^{-13} so this example is contrived but illustrative.) AAAtrig identifies the branch points of f and clusters poles along its branch cuts, as illustrated in Figure 2. When the cleanup procedure is disabled (Figure 2(a)), AAAtrig identifies 98 poles of which 66 poles which have very small numerical residue and thus represent Froissart doublets. These poles are mainly clustered around the unit circle. Conversely, Figure 2(b) shows the effect of applying AAAtrig with the cleanup procedure enabled. The algorithm now terminates with 32 poles of which only one is a Froissart doublet. The maximum error is $\mathcal{O}(10^{-13})$ in both cases.

3.3. Comparison to trigonometric polynomial approximation. We now investigate the performance of the rational trigonometric approximant computed by AAAtrig in comparison to the best-fit trigonometric polynomial. In the special case when the sample points are equally spaced, the FFT can be used to compute the best-fit trigonometric polynomial interpolant. For example, if we have $f_n = f(2\pi n/M)$ for $n = 0, \ldots, M-1$ then we have the discrete Fourier transform pair

$$F_j = \frac{1}{M}\sum_{k=0}^{M-1} f_k \mathrm{e}^{-2\pi \mathrm{i} kj/M}, \qquad f_j = \sum_{k=0}^{M-1} F_k \mathrm{e}^{2\pi \mathrm{i} kj/M}, \tag{3.1}$$

which can be rapidly computed using an FFT algorithm. As such, the family of trigonometric polynomials

$$f(z) = \sum_{k=0}^{M-1} F_k \mathrm{e}^{\mathrm{i}kz} \mathrm{e}^{\mathrm{i}m_k Mz} \tag{3.2}$$

interpolate the data for any integers m_k. This interpolant varies with m_k and most choices of m_k will result in an approximant that oscillates wildly between the sample points. This is the aliasing phenomenon. A suitable choice of m_k depends on the

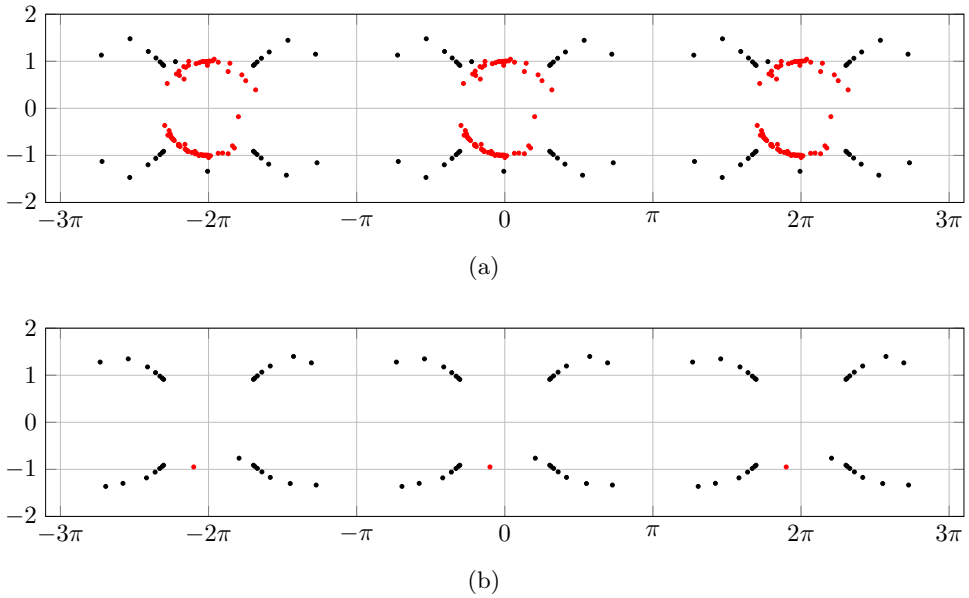


FIG. 2. *The locations of the poles of the periodic rational approximant for* $f(x) = \log(2 + \cos(x)^4)$ *for AAAtrig with a tolerance of* 0 *without* (a) *and with* (b) *the cleanup procedure. The function* f *is sampled at* 1,000 *roots of unity. Poles with a residue of magnitude less than* 10^{-13} *are plotted in red; all other poles are plotted in black. The cleanup procedure removes all but one Froissart doublet.*

application at hand, but it is generally desirable to minimize the oscillations between the sample points. Choosing the m_k that result in an approximant that oscillates as little as possible between the sample points yields

$$f(z) = F_0 + \sum_{k=1}^{\lfloor M/2 \rfloor} \left(F_k \mathrm{e}^{\mathrm{i}kz} + F_{M-k} \mathrm{e}^{-\mathrm{i}kz} \right) + F_{M/2} \cos\left(Mx/2\right). \tag{3.3}$$

The final term is set to zero if M is odd. Moreover, since the trigonometric functions form an orthogonal basis, truncating (3.3) at $m < M$ terms provides the degree m trigonometric polynomial approximant that is a best fit for the M data points in the least-squares sense. This corresponds to bandlimiting the signal in the frequency domain with a low-pass filter; see the discussion in section 10.3.2 of [42] for further details.

Figure 3 compares the best degree m trigonometric polynomial approximant to the degree m trigonometric rational approximant computed by AAAtrig. The function being approximated is $f(x) = \tanh(60\cos(x))$ and is sampled at 1,024 equispaced points in the interval $[0, 2\pi)$. The top right panel of Figure 3 plots f and illustrates the nearly discontinuous behavior at the points $x = \pi/2, 3\pi/2$. We then approximate f using (3.3) truncated at different values of m to obtain the best mth-order trigonometric polynomial in the least-squares sense. The error is compared to that of a type $(m-1, m-1)$ trigonometric rational function computed by AAAtrig in the left panel of Figure 3. The AAAtrig algorithm terminates with a tolerance of 10^{-13} at around $m = 50$. Beyond this, the algorithm stagnates and higher-order approximants show no improvements. Achieving similar accuracy with a trigonometric polynomial

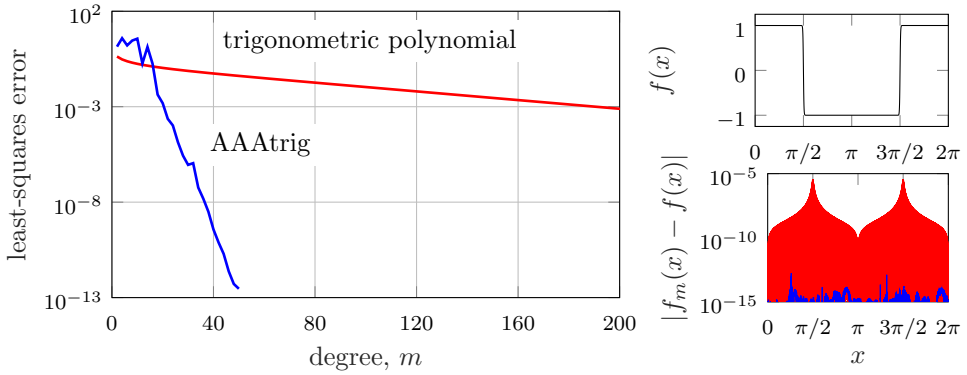


FIG. 3. *Comparison of AAAtrig (blue) to the best trigonometric polynomial approximant (red) for* $f(x) = \tanh(60\cos(x))$. *On the left we compare the least squares error for degree* m *approximants. The signal* f *is plotted at the top right; on the bottom right we plot the error for the converged AAAtrig approximant and the order* 1,024 *trigonometric interpolating polynomial evaluated at a grid* 10 *times finer than the sample points. Here* $f_m(x)$ *is the degree* m *approximant.*

interpolant requires nearly 1,024 terms: the slow decay of the red error curve in the left panel of Figure 3 illustrates the slow convergence of a trigonometric polynomial approximant for this function.

When an order 1,024 trigonometric polynomial is used, the approximant interpolates the data at each sample point with minimal oscillations in-between. The error between this interpolant and f sampled on a grid 10 times finer than the original data is compared to that of AAAtrig in the bottom right panel of Figure 3. The error of the FFT-based interpolant is relatively large (10^{-5}) near the transition points at $\pi/2$ and $3\pi/2$ where a Gibbs-like phenomenon occurs. In contrast, AAAtrig maintains an accuracy of 10^{-13} even near these points. If we were to plot the poles and zeros of the AAAtrig approximant then we would see that they cluster exponentially near $\pi/2$ and $3\pi/2$ where the gradient is almost singular. This improved accuracy is especially notable when we consider that the AAAtrig approximant only uses around 50 terms whereas the final FFT interpolant uses 1,024 terms.

This example illustrates the advantages of using a trigonometric rational approximant as opposed to a naive trigonometric polynomial approximant. The differences are most noticeable at the nearly discontinuous points where AAAtrig is free to cluster poles and zeros. Additionally, the sample points used by AAAtrig may be very general and need not be regularly spaced or real. Of course, there is no comparison in speed—the FFT can be computed in $\mathcal{O}(M\log(M))$ flops whereas AAAtrig uses $\mathcal{O}(Mm^3)$ flops—but if the approximant is to be evaluated a large number of times then AAAtrig compression may be more suitable. We emphasize that there are well-known methods to deal with singularities, jumps, and nonequispaced points within the FFT framework [11, 13, 29, 43, 22, 36]: we have only considered the most basic implementation here for the purpose of illustration. Further applications of AAA to signal processing have been developed in [49].

3.4. Compression of periodic harmonic functions. Since the initial publication of the AAA algorithm, a number of works have applied it to represent harmonic functions, especially in domains with corners. Motivated by Newman's rational approximation of $|x|$ [39], Gopal and Trefethen showed that more general corner sin-

gularities of the form x^α can be approximated with root-exponential accuracy by exponentially clustering poles near the origin [27]. These findings guide a new powerful class of methods to solve Laplace's equation (and other elliptic PDEs) called lightning solvers. The strategy is to express the solution as a rational function in the form of a combination of a polynomial (called the "Runge part") and a series of partial fractions (called the "Newman part"). The poles of the partial fractions are prescribed to cluster exponentially around each corner of the domain and the coefficients are then found by collocating on the boundary and solving a least-squares problem.

Once the coefficients have been found, the AAA algorithm can compress the solution into a more compact form that uses fewer poles [26, 47]. This compression is performed by sampling the solution on the boundary and then passing these points through AAA. If the resulting approximant contains no poles in the domain then it is also a solution of Laplace's equation and satisfies the boundary conditions to the tolerance specified by AAA.

These ideas generalize quite naturally to periodic domains—periodizing the lightning solver of [27] results in an ansatz of the form

$$f(z) = \underbrace{\sum_{j=1}^{n_1} a_j \cot\left(\frac{z - z_j}{2}\right)}_{\text{periodic Newman part}} + \underbrace{\sum_{j=1}^{n_2} b_j \cot\left(\frac{z - z^*}{2}\right)^j}_{\text{periodic Runge part}}, \tag{3.4}$$

where n_1 and n_2 represent the number of terms included in each part and z^* is a point located inside the boundary. The poles are prescribed at $z = z_j + 2\pi k$ for $k \in \mathbb{Z}$ and the coefficients a_j and b_j are then found by enforcing the boundary condition, which can be of Dirichlet or Neumann form, and solving a least-squares problem. When forming the least-squares problem, the Runge part takes a Vandermonde structure which generally leads to an ill-conditioned matrix. This issue can be circumvented by using the new "Vandermonde with Arnoldi" algorithm [12] to construct a polynomial basis that is discretely orthogonal with respect to the collocation points. Restructuring the least-squares problem in terms of this orthogonal basis dramatically improves the condition number of the problem.

After it has been calculated, the solution f can be compressed with AAAtrig. An example of AAAtrig compression is illustrated in Figure 4, which shows the potential flow past a periodic array of semicircular obstacles. Such geometries commonly arise in applications in aerodynamics, and microfluidics [2]. In this case, the complex function $f(z)$ represents the perturbation to a uniform flow induced by the boundaries and the boundary condition is enforced by requiring that the stream function vanishes on the boundaries, i.e., $\mathbb{I}\mathrm{m}[f(z) + \mathrm{i}z] = 0$. (The author has generalized this approach to a variety of potential flows including circulatory flows, point vortex dynamics, and free streamline flows in [1].) Expressing f in the form (3.4) and finding the coefficients results in a solution of accuracy 10^{-5} using 122 poles. The solution is then compressed with AAAtrig by evaluating $f(z)$ at 1,000 sample points on the boundary. The resulting trigonometric rational function has only 36 poles and is therefore much faster to evaluate than the original solution. In both cases, the poles cluster exponentially close to the corners to capture the singularities of the solution there, as illustrated in the green inset box in Figure 4. Moreover, AAAtrig selects poles that seem to represent a branch cut that connects the two corners. As we shall see in the next example, AAA and AAAtrig select poles that approximate branch cuts in some sort of optimal way such that the normal derivatives of a potential gradient are balanced [44, 48].

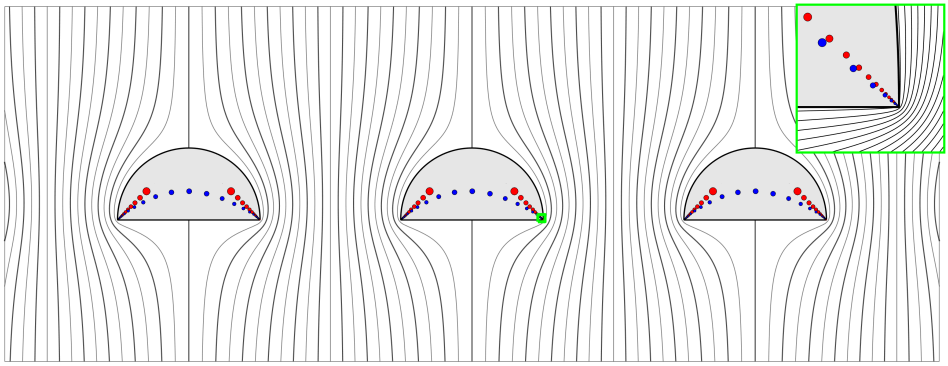


FIG. 4. *Potential flow through a periodic array of semicircles solved with the periodic lightning solver* (3.4) *then compressed with AAAtrig. The black lines indicate the streamlines of the flow. The red circles represent the poles used by the lightning solver and the blue circles represent the poles computed by AAAtrig. The size of the circles correspond to the size of the residue of each pole. The close-up view in the inset green box illuminates the exponential clustering of the poles near the corners.*

This application illustrates the need for AAAtrig over AAA in periodic applications. Unlike the example in section 3.1, here it is important that the approximant has the correct behavior away from the sample points. For example, a solution to Laplace's equation in a periodic domain generally exhibits exponential decay as $z \to \pm\,\mathrm{i}\infty$. A polynomial barycentric form computed with AAA will require many support points to approximate this behavior, which is at odds with the aim of compressing the solution. Additionally, the solution f may attain different values as $z \to \pm\,\mathrm{i}\infty$, which cannot be reproduced with polynomial rational functions. These behaviors could be approximated by including 2π-translates of the support points (i.e., by truncating (2.11)) but this introduces an additional complication of where to truncate, which would be application dependent. Moreover, the approximant would only be approximately periodic, which could result in some unexpected behavior if the approximant is used in future calculations. In summary, AAA is unsuitable for this application and AAAtrig should be favored.

3.5. Periodic and multivalued conformal maps. Conformal maps are another application amenable to AAA compression. Constructing conformal maps is challenging and, even after they have been constructed, their evaluation can require expensive integrals. For example, Schwarz–Christoffel (SC) formulas are a class of formulas that represent conformal maps from canonical domains (typically the unit disk) to polygons but must be computed using quadrature [19]. An alternative approach to representing conformal mappings was suggested by Gopal and Trefethen [26]—find the boundary correspondence function between the two domains and then approximate the mapping using AAA. This compression typically results in a speedup of the order of 10–1,000 and has the additional advantage that both the forward (from canonical domain to polygonal domain) and backward (from polygonal domain to canonical domain) mappings can be computed: the SC formulas typically only provide the forward map.

Again, these ideas generalize naturally to periodic domains with minor modifications. The periodic SC formula [3] can be used to determine the boundary correspondence function between the unit disc (the ζ-plane) and a periodic array of polygons

(the z-plane). When there is one N-sided polygon per period window, the formula takes the form

$$f(\zeta) = A \int^{\zeta} \frac{\prod_{j=1}^{N} (\zeta' - \zeta_j)^{\beta_j} \, \mathrm{d}\zeta'}{(\zeta' - a_{\infty^-})\,(\zeta' - a_{\infty^+})\,(\zeta' - 1/\overline{a_{\infty^+}})\,(\zeta' - 1/\overline{a_{\infty^-}})} + B. \tag{3.5}$$

In the above, β_j are the turning angles, ζ_j are the preimages of the corners, $a_{\infty^\pm}$ are the preimages of $\pm\,\mathrm{i}\infty$, A is a scaling and rotation constant, and B is a translation constant. Generally these parameters must be found numerically; we calculate them using an adapted version of the SC Toolbox [17]. As an aside, the periodic SC formula (3.5) generalizes to an arbitrary number of boundaries per period window by employing the transcendental Schottky–Klein prime function [15, 16].

Since the integrand in (3.5) contains simple poles at $a_{\infty^\pm}$, the function $f(\zeta)$ possesses a branch cut. Crossing this branch cut corresponds to traversing between adjacent period windows. As such, f is multivalued and cannot be represented with a polynomial AAA approximant. However, since the period of the polygonal domain is known (it is specified in the parameter problem), we know the strength of the branch points and can thus extract the branch cut from the conformal map. For example, when the period is 2π we may express the conformal map as

$$z = f(\zeta) = \tilde{f}(\zeta) + \mathrm{i} \log\left(\frac{\zeta - a_{\infty^-}}{\zeta - a_{\infty^+}}\right) \tag{3.6}$$

where $\tilde{f}$ is analytic in the unit disk and can therefore be computed with standard polynomial AAA. The result is illustrated in Figure 5(a)—AAA approximates the branch cuts near the preimages of the corners with clusters of poles. We emphasize that this compression is purely adaptive: the poles and zeros are not user specified and arise naturally from AAA's choice of support points and weights. Typical tests show that the AAA representation is around 10–100 times faster than using quadrature methods to evaluate the integrals. If AAA is applied directly without first extracting the branch cut then the resulting poles and zeros interlace between $a_{\infty^\pm}$ to approximate the branch cut, and the approximation breaks down in a neighborhood of the branch points. Extracting the branch points as in (3.6) results in a representation that is faithful throughout the entire domain.

There are typically no constructive formulas for backward SC mappings (polygon to disc) and they are generally computed with Newton iteration. This approach is usually prohibitively expensive in applications. In our case, the inverse mapping f^{-1} is periodic—every polygon is effectively mapped to the same disk (although technically they are on different sheets of an infinite, periodic Riemann surface). As such, the conformal map can be represented with AAAtrig. Since $\pm\,\mathrm{i}\infty$ are mapped to different points $a_{\infty^\pm}$, it is necessary to use the odd version of AAAtrig (2.6). The result is illustrated in Figure 5(b). Again, the poles computed by AAAtrig cluster exponentially close to the corners of the polygon. Moreover, the poles align along some approximate branch cut that connects each of the corners. We emphasize that the locations of these poles are not specified by the user but are selected adaptively and automatically by AAAtrig; we will discuss the locations of these poles in the next section. Compressing with AAAtrig resulted in speedup in the range of 100–10,000 times.

A significant issue in SC mappings is the "crowding phenomenon" which occurs when mappings of elongated regions produce very close vertex preimages and exponential distortions of density [5]. The beginnings of crowding can be seen in the nonuniform distribution of the preimages of the vertices and their close proximity to

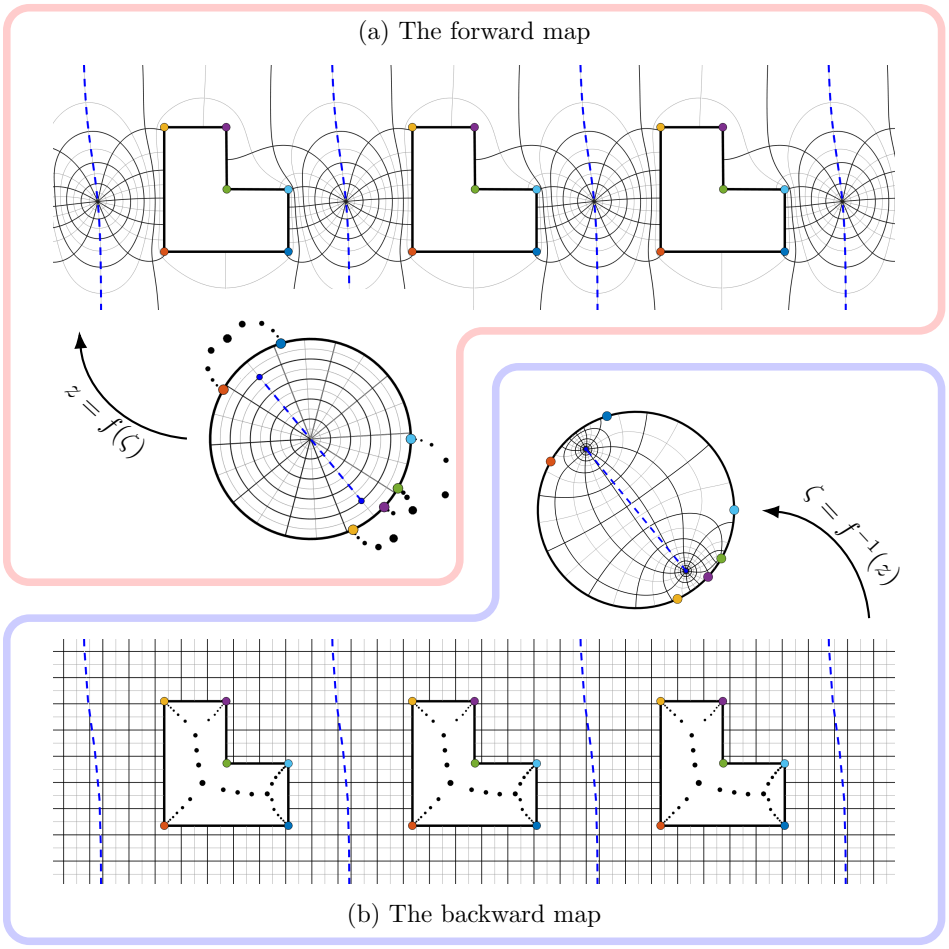
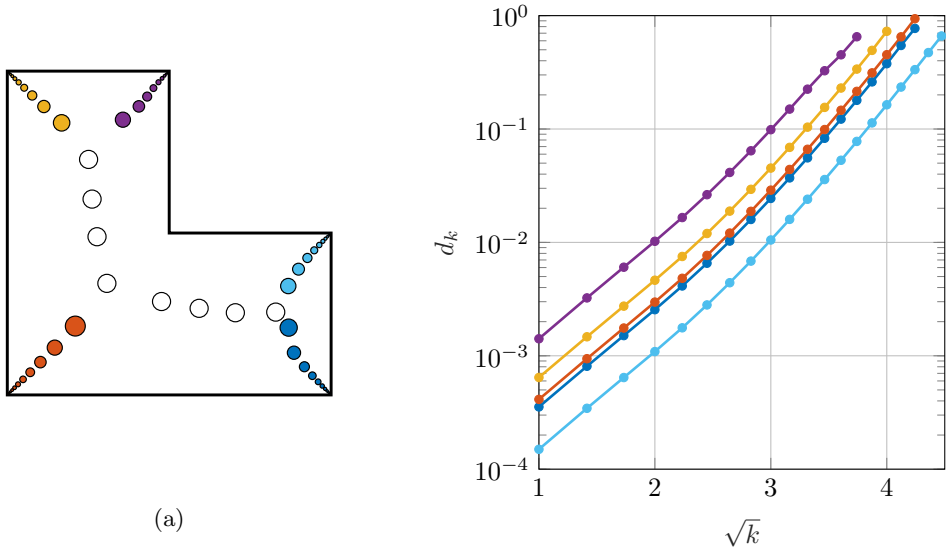


FIG. 5. *Conformal maps between the unit disc and a periodic array of L-shaped polygons. Figure* (a) *shows the (multivalued) forward map from the disc to the periodic domain; Figure* (b) *shows the (periodic) backward map from the periodic domain to the disc. The forward map is constructed using the periodic SC formula* (3.5) *and then compressed using AAA* (3.6)*. The backward map is computed by applying AAAtrig directly to the boundary correspondence function. The black dots correspond to the poles computed by AAA and AAAtrig and their size corresponds to the size of the residue at each pole. The colored dots correspond to the corners and preimages of the corners. The dashed blue line represents the branch cut and the endpoints of the branch cut are* $a_{\infty}{}^{\pm}$*. A close-up of the structure of the poles in the L-shaped domain is in Figure* 6(a).

the preimages of infinity in Figure 5. Gopal and Trefethen [26] proved that a conformal map of an elongated region (termed a "finger") necessitates a singularity or loss of univalence exponentially close to the boundary. As we have discussed, these behaviors can be approximated by rational functions, whereas polynomial approximants will fail. Thus, the procedure of this present section is suitable for elongated regions, provided the SC parameters are calculated accurately.

Periodic conformal maps between more general nonpolygonal shapes can also be constructed by using the lighting solver of (3.4) with appropriately defined boundary values following the approach of [47]. Again, the number of poles used to represent these maps can be drastically reduced using AAAtrig compression.



(a) (b)

FIG. 6. (a) *A close-up of the poles discovered by AAAtrig for the backward mapping in Figure* 5. (b) *The distance d_k from each corner to its kth nearest poles on a square-root scale. The line color corresponds to the locations of the poles in Figure* (a). *Note that the log-distance scales approximately linearly on the $\sqrt{k}$ scale for small k. Poles with white fill in Figure* (a) *are not directly associated with a particular corner and are therefore not plotted in Figure* (b). *The reentrant corner does not have an associated cluster of poles.*

3.6. Clustering of poles and zeros. We end with a brief discussion of the distribution of poles selected by AAAtrig. A recent study [48] has shown that the optimal distribution of poles and zeros near a singularity is given by the tapered distribution

$$d_k \approx \beta \exp(\sigma\sqrt{k}), \tag{3.7}$$

where d_k is the distance of the kth pole from the singularity and $\beta > 0$ and $\sigma < 0$ are constants. With this distribution, the density of poles on a log scale decays linearly as the distance to the singularity decreases. In the conformal mapping example we see that AAAtrig selects poles and zeros that obey this tapering law. Figure 6 shows the distributions of poles nearest to five corners of the L-shaped polygon in the backward map in Figure 5. Again, this distribution of poles was not specified but arose through the application of AAAtrig. The reentrant corner (green) does not have an associated distribution of poles and is thus excluded. The log-distance is seen to vary linearly with respect to $\sqrt{k}$ in Figure 6(b), showing that these poles obey the tapering law (3.7).

4. Conclusions. We have presented an extension of the AAA algorithm [37] to compute rational approximants for periodic functions. The algorithm is implemented in Chebfun. The sample points need not be equispaced and are not constrained to a particular subset of the complex plane. As with the original AAA algorithm, AAAtrig exploits a (trigonometric) barycentric representation of the approximant and uses adaptive, greedy selection of support points. Thus, AAAtrig achieves the same high levels of robustness, flexibility, and accuracy seen in the original AAA

algorithm. As one would expect, AAAtrig is an improvement over AAA when the underlying function is periodic. We have demonstrated that the periodic approximant is essential in applications to PDEs and conformal maps when the approximant must be periodic throughout the entire complex plane. We have also developed a minimax (AAA–Lawson) [38] version of AAAtrig though we did not report any of the details in this paper in the interests of brevity.

Much research is currently dedicated to understanding the theoretical underpinnings of the AAA framework and we expect that these advances will translate naturally to AAAtrig. Other lines of enquiry should develop applications to PDEs in the spirit of the new lightning solvers. We have seen how AAAtrig can be used to compress solutions of Laplace's equation, and there is significant scope to develop solvers for other PDEs in periodic domains, such as the Helmholtz, convected Helmholtz, and biharmonic equations. Signal processing is another promising application, and we showed that AAAtrig has some advantages over FFT-based interpolation in section 3.3. A parallel study has used a similar version of AAAtrig to perform signal reconstruction for noisy or incomplete data by combining a similar version of AAAtrig with Prony's method [49].

Acknowledgments. The author acknowledges insightful conversations with Profs. L. N. Trefethen, Y. Nakatsukasa, and J. A. C. Weideman at the "Complex analysis: techniques, applications and computations" programme at the Isaac Newton Institute for Mathematical Sciences, Cambridge. The author is grateful for the helpful comments from the anonymous reviewers.

REFERENCES

[1] P. J. Baddoo, *Lightning solvers for potential flows*, Fluids, 5 (2020), 227.
[2] P. J. Baddoo and L. J. Ayton, *A calculus for flows in periodic domains*, Theor. Comput. Fluid Dyn., 35 (2021), pp. 145–168, https://doi.org/10.1007/s00162-020-00551-x.
[3] P. J. Baddoo and D. G. Crowdy, *Periodic Schwarz-Christoffel mappings with multiple boundaries per period*, Proc. R. Soc. A, 475 (2019), 20190225.
[4] R. Baltensperger, *Some results on linear rational trigonometric interpolation*, Comput. Math. Appl., 43 (2002), pp. 737–746.
[5] L. Banjai, *Revisiting the crowding phenomenon in Schwarz–Christoffel mapping*, SIAM J. Sci. Comput., 30 (2008), pp. 618–636, https://doi.org/10.1137/060677392.
[6] J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures*, Wiley, Hoboken, NJ, 2010.
[7] J. P. Berrut, *Baryzentrische Formeln zur Trigonometrischen Interpolation* (I), Z. Angew. Math. Phys., 35 (1984), pp. 91–105.
[8] J. P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Comput. Math. Appl., 15 (1988), pp. 1–16.
[9] J. P. Berrut and G. Klein, *Recent advances in linear barycentric rational interpolation*, J. Comput. Appl. Math., 259 (2014), pp. 95–107.
[10] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev., 46 (2004), pp. 501–517.
[11] G. Beylkin, *On the fast Fourier transform of functions with singularities*, Appl. Comput. Harmon. Anal., 2 (1995), pp. 363–381, https://doi.org/10.1006/acha.1995.1026.
[12] P. D. Brubeck, Y. Nakatsukasa, and L. N. Trefethen, *Vandermonde with Arnoldi*, SIAM Rev., 63 (2021), pp. 405–415.
[13] E. Candès, L. Demanet, D. Donoho, and L. Ying, *Fast discrete curvelet transforms*, Multiscale Model. Simul., 5 (2006), pp. 861–899, https://doi.org/10.1137/05064182X.
[14] W. J. Cody, G. Meinardus, and R. S. Varga, *Chebyshev rational approximations to e^{-x} in $[0, +\infty)$ and applications to heat-conduction problems*, J. Approx. Theory, 2 (1969), pp. 50–65.
[15] D. G. Crowdy, *Schwarz–Christoffel mappings to unbounded multiply connected polygonal regions*, Math. Proc. Cambridge Philos. Soc., 142 (2007), pp. 319–339.

[16] D. G. Crowdy, *Solving problems in multiply connected domains*, CBMS-NSF Regional Conf. Ser. in Appl. Math., SIAM, Philadelphia, 2020.
[17] T. A. Driscoll, *Algorithm 756; a MATLAB toolbox for Schwarz-Christoffel mapping*, ACM Trans. Math. Software, 22 (1996), pp. 168–186.
[18] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*, Pafnuty Publications, Oxford, 2014.
[19] T. A. Driscoll and L. N. Trefethen, *Schwarz-Christoffel Mapping*, Cambridge University Press, Cambridge, 2002.
[20] S. Elsworth and S. Güttel, *Conversions between barycentric, RKFUN, and Newton representations of rational interpolants*, Linear Algebra Appl., 576 (2019), pp. 246–257.
[21] G. Fairweather and A. Karageorghis, *The method of fundamental solutions for elliptic boundary value problems*, Adv. Comput. Math., 9 (1998), pp. 69–95.
[22] G. X. Fan and Q. H. Liu, *Fast Fourier transform for discontinuous functions*, IEEE Trans. Antennas and Propagation, 52 (2004), pp. 461–465, https://doi.org/10.1109/TAP.2004.823965.
[23] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334, https://doi.org/10.1137/1015032.
[24] G. H. Golub and R. Underwood, *Stationary values of the ratio of quadratic forms subject to linear constraints*, Z. Angew. Math. Phys., 21 (1970), pp. 318–326, https://doi.org/10.1007/BF01627939, https://link.springer.com/article/10.1007/BF01627939.
[25] A. Gopal and L. N. Trefethen, *New Laplace and Helmholtz solvers*, Proc. Natl. Acad. Sci. USA, 116 (2019), pp. 10223–10225.
[26] A. Gopal and L. N. Trefethen, *Representation of conformal maps by rational functions*, Numer. Math., 142 (2019), pp. 359–382.
[27] A. Gopal and L. N. Trefethen, *Solving Laplace problems with corner singularities via rational functions*, SIAM J. Numer. Anal., 57 (2019), pp. 2074–2094.
[28] V. Gosea and S. Güttel, *Algorithms for the rational approximation of matrix-valued functions*, SIAM J. Sci. Comput., 43 (2021), pp. A3033–A3054.
[29] L. Greengard and J.-Y. Lee, *Accelerating the nonuniform fast Fourier transform*, SIAM Rev., 46 (2004), pp. 443–454, https://doi.org/10.1137/S003614450343200X.
[30] B. Gustavsen and A. Semlyen, *Rational approximation of frequency domain responses by vector fitting*, IEEE Trans. Power Deliv., 14 (1999), pp. 1052–1059.
[31] P. Henrici, *Barycentric formulas for interpolating trigonometric polynomials and their conjugates*, Numer. Math., 33 (1979), pp. 225–234.
[32] P. Henrici, *Applied and Computational Complex Analysis, Volume* 3: *Discrete Fourier Analysis, Cauchy Integrals, Construction of Conformal Maps, Univalent Functions*, Wiley, New York, 1986.
[33] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556.
[34] M. Javed, *Algorithms for Trigonometric Polynomial and Rational Approximation*, PhD thesis, University of Oxford, Oxford, 2016.
[35] G. Klein, *Applications of Linear Barycentric Rational Interpolation*, PhD thesis, University of Fribourg, Fribourg, Switzerland, 2012.
[36] Y. Liu, Z. Nie, and Q. H. Liu, *DIFFT: A fast and accurate algorithm for Fourier transform integrals of discontinuous functions*, IEEE Microw. Wirel. Compon. Lett., 18 (2008), pp. 716–718, https://doi.org/10.1109/LMWC.2008.2005162.
[37] Y. Nakatsukasa, O. Sète, and L. N. Trefethen, *The AAA algorithm for rational approximation*, SIAM J. Sci. Comput., 40 (2018), pp. A1494–A1522.
[38] Y. Nakatsukasa and L. N. Trefethen, *An algorithm for real and complex rational minimax approximation*, SIAM J. Sci. Comput., 42 (2020), pp. A3159–A3179.
[39] D. J. Newman, *Rational approximation to* $|x|$, Michigan Math. J., 11 (1964), pp. 11–14.
[40] A. C. Rodriguez and S. Gugercin, *The p-AAA Algorithm for Data Driven Modeling of Parametric Dynamical Systems*, preprint, arXiv:2003.06536, 2020.
[41] H. Rutishauser, *Vorlesungen über Numerische Mathematik*, Birkhäuser Basel, 1976, https://doi.org/10.1007/978-3-0348-5709-3.
[42] T. Sauer, *Numerical Analysis*, Pearson, Boston, 2nd ed., 2012.
[43] E. Sorets, *Fast Fourier transforms of piecewise constant functions*, J. Comput. Phys., 116 (1995), pp. 369–379, https://doi.org/10.1006/jcph.1995.1035.
[44] H. R. Stahl, *Sets of Minimal Capacity and Extremal Domains*. preprint, https://arxiv.org/abs/1205.3811 (2012).
[45] T. W. Tee and L. N. Trefethen, *A rational spectral collocation method with adaptively transformed Chebyshev grid points*, SIAM J. Sci. Comput., 28 (2006), pp. 1798–1811.

[46] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*, SIAM, Philadelphia, 2019.
[47] L. N. Trefethen, *Numerical conformal mapping with rational functions*, Comput. Methods Funct. Theory, 20 (2020), pp. 369–387.
[48] L. N. Trefethen, Y. Nakatsukasa, and J. A. C. Weideman, *Exponential node clustering at singularities for rational approximation, quadrature, and PDEs*, Numer. Math., 147 (2021), pp. 227–254.
[49] H. Wilber, A. Damle, and A. Townsend, *Data-driven algorithms for signal processing with rational functions*, preprint, arXiv:2105.07324v1 (2021).
[50] G. B. Wright, M. Javed, H. Montanelli, and L. N. Trefethen, *Extension of Chebfun to periodic functions*, SIAM J. Sci. Comput., 37 (2015), pp. C554–C573.