NUMERICAL ALGORITHMS BASED ON ANALYTIC FUNCTION VALUES AT ROOTS OF UNITY*

ANTHONY P. AUSTIN†, PETER KRAVANJA‡, AND LLOYD N. TREFETHEN§

Abstract. Let $f(z)$ be an analytic or meromorphic function in the closed unit disk sampled at the nth roots of unity. Based on these data, how can we approximately evaluate $f(z)$ or $f^{(m)}(z)$ at a point z in the disk? How can we calculate the zeros or poles of f in the disk? These questions exhibit in the purest form certain algorithmic issues that arise across computational science in areas including integral equations, partial differential equations, and large-scale linear algebra. We analyze the possibilities and emphasize a basic distinction between algorithms based on polynomial or rational interpolation and those based on trapezoidal rule approximations of Cauchy integrals. We then show how these developments apply to the problem of computing the eigenvalues in the disk of a matrix of large dimension.

Key words. polynomial interpolation, barycentric formula, Cauchy integral formula, trapezoidal rule, rational functions, eigenvalues, roots of unity, FEAST

AMS subject classifications.

1. Introduction and the unit disk filter function. The title of this paper suggests a narrow topic, but in fact, our aim is a broad one: to present a set of ideas underlying certain numerical algorithms used across computational science. In each application, each geometry, it is easy to get lost in details particular to the problem at hand. But the common threads are remarkable, and concentrating our attention on analytic functions at roots of unity on the unit disk will help us to see them. Table 1.1 summarizes the problems and algorithms we shall discuss. From analytic functions on the unit disk one can reach out to generalizations including harmonic functions [12], matrix-valued functions [62], noncircular geometries [19], irregular sample points [34], non-smooth functions (periodic function with discontinuities) [20], dimensions greater than 2 [70], Helmholtz problems [45], integral equations [30], radial basis functions [22], nonlinear partial differential equations [44], and linear operators [29]. (The references just given are a mix of mathematical classics and numerical state-of-the-art.) We offer this list with the thought in mind that researchers may find it fruitful, in investigating some of these variants, to use the present paper as a template.

Throughout this paper, n is a positive integer and $\{z_k\}$, $0 \le k \le n-1$, are the nth roots of unity, $z_k = \exp(2\pi ik/n)$. We let S denote the unit circle, D the open unit disk, and D_R the open disk $\{z \in C : |z| < R\}$. The function f is assumed to be analytic in D_R but not $\overline{D}_R$ for some $R > 1$, and $\{f_k\}$ are its sampled values $f_k = f(z_k)$. The symbol P_{n-1} denotes the set of polynomials of degree at most $n-1$.

A certain simple function will have special importance in our discussion, which

*APA and LNT were supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement no. 291068. The views expressed in this article are not those of the ERC or the European Commission, and the European Union is not liable for any use that may be made of the information contained here.

†Oxford U. Mathematical Institute, Oxford OX2 6GG, UK, austin@maths.ox.ac.uk

‡Dept. of Comp. Sci., Katholieke U. Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium, peterkravanja@gmail.com

§Oxford U. Mathematical Institute, Oxford OX2 6GG, UK, trefethen@maths.ox.ac.uk

FIG. 1.1. *An outline of the problems and algorithms considered in this article, showing a selection of key publications followed by Matlab names of algorithms in typewriter font. (Some of the classifications are arguable, and some of the entries do not discuss roots of unity or discretizations of Cauchy integrals explicitly.) One of the observations of this paper is the mathematical equivalence of algorithm* `[Cz2]`*, based on discretized Cauchy integrals defining certain moments, and the more stable algorithm* `[ratdisk]`*, based on linearized rational interpolation.*

	Polynomial interpolation	**Discrete Cauchy integral**	**Linearized rational interpolation**
Function values	Méray 1884 Runge 1904 Fejér 1918 Walsh 1935 Henrici 1982 Gutknecht 1986 `[P],[P*]`	Lyness-Delves 1967 Gutknecht 1986 Helsing-Ojala 2008 `[C]`	Jacobi 1846 Eğecioğlu-Koç 1989 Fornberg-Wright 2004 Gonnet-Pachón-Tref. 2011 Pachón-Gonnet-V. Deun 2012
Derivatives or Taylor coefficients	Schneider-Werner 1986 Ioakimidis-Pap.-Perd. 1991 `[P'],[P'*]`	Lyness-Moler 1967 Lyness-Sande 1971 Henrici 1979 Fornberg 1981 Bornemann 2011 Ioakimidis-Pap.-Perd. 1991 `[C']`	Schneider-Werner 1986
Zeros and poles	Fortune 2001 Corless 2004 Amiraslani 2006 Townsend 2012 `[Pz],[Pz*]`	Jackson 1917 McCune 1966 Delves-Lyness 1967 Burniston-Siewert 1973 Henrici 1979 Ioakimidis 1985 Anastasselou 1986 Kravanja-Sak.-V. Barel 1999 Kravanja-Van Barel 1999 Kravanja-Van Barel 2000 Luck-Stevens 2002 `[Cz1],[Cz2],[Cp]`	Gonnet-Pachón-Tref. 2011 `[ratdisk]`
Matrix eigenvalues	`[det]`	Goedecker 1999 Sakurai-Sugiura 2003 Sakurai-Tadano 2007 Polizzi 2009 Ikegami-Sakurai 2010 Asakura et al. 2009 Beyn 2012	`[res]`

one might call the *unit disk filter function*:[1]

$$b(z) = \frac{1}{1-z^n} = \prod_{k=0}^{n-1} \frac{z_k}{z_k - z}. \tag{1.1}$$

[1]The function b is related to the Butterworth filter of electrical engineering. The Butterworth filter takes the product over just half of the roots of unity, those in one half-plane, and approximates constant modulus 1 over a diameter of D rather than constant value 1 over all of D. See Figure 1.9 of [77].

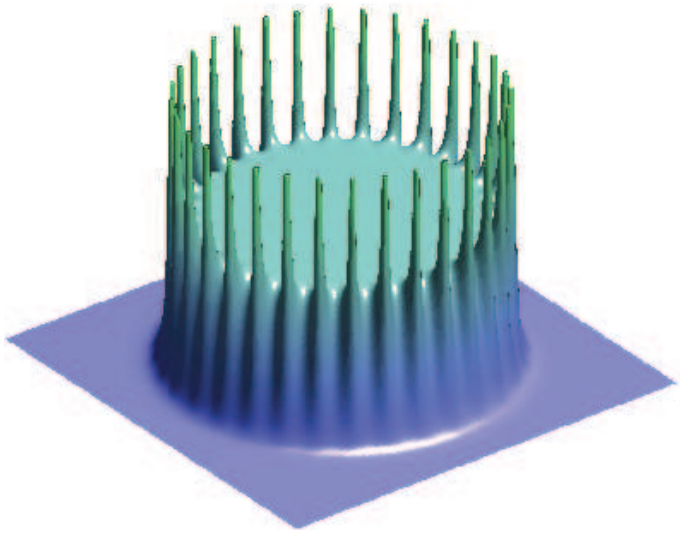

FIG. 1.2. *Absolute value of the unit disk filter function $b(z)$ of (1.1) for $n = 32$. As $n \to \infty$, the function converges to 1 for $|z| < 1$ and 0 for $|z| > 1$, but there is always a ring of poles along the unit circle.*

This is a rational function with n poles and no zeros, apart from a zero of multiplicity n at $z = \infty$. Figure 1 illustrates the shape of $b(z)$ in the complex plane, with values $b(z) \approx 1$ inside D and $b(z) \approx 0$ outside. The separation between the two regions is effected by a cage of poles along the unit circle.[2] The pole at z_k has residue

$$\text{Res}_k = -\frac{z_k}{n}, \tag{1.2}$$

and thus each individual pole gets weaker as $n \to \infty$. In the limit, pointwise for each z with $|z| \neq 1$, we have

$$\lim_{n\to\infty} b(z) = \begin{cases} 1 & |z| < 1, \\ 0 & |z| > 1. \end{cases} \tag{1.3}$$

2. Evaluating a function and its derivatives in the unit disk.

2.1. Two algorithms for evaluating a function. We begin with the most basic of all computational problems. Suppose $z \in D$ is given and we wish to evaluate $f(z)$ approximately based on the values $\{f_k\}$. Here are two good algorithms, whose interplay will resonate throughout this paper.

ALGORITHM P: POLYNOMIAL INTERPOLATION. *Approximate $f(z)$ by $p(z)$, where $p \in P_{n-1}$ is the unique polynomial interpolant to f in the points $\{z_k\}$.*

ALGORITHM C: DISCRETE CAUCHY INTEGRAL. *Approximate $f(z)$ by the trapezoidal rule discretization in the points $\{z_k\}$ of the integral $(2\pi i)^{-1} \int_S (\zeta - z)^{-1} f(\zeta)\, d\zeta$.*

Both algorithms make use of exactly the same data $\{f_k\}$, and both, as we shall see in §2.4, deliver results that converge geometrically to $f(z)$ as $n \to \infty$. Indeed, one's first thought might be, are these two different descriptions of the same method?

[2] This is related mathematically to the Faraday cage of electromagnetism. See [76, Sec. 3].

But they are not the same. We shall show that algorithm C approximates $f(z)$ not by a polynomial but by the rational function

$$r(z) = \frac{p(z)}{1 - z^n} = b(z)p(z). \tag{2.1}$$

Thus the result of algorithm C is the same as that of algorithm P, but multiplied by the filter function (1.1), which introduces poles at all the roots of unity (except if a sample value f_k happens to be zero). In this paper we shall consider the significance of this relationship from several angles.

In the special case $z = 0$, $b(z)$ takes the value 1, and the results of algorithms P and C are identical. This is the case that has gotten the most attention in the literature, beginning with work in the 1960s by Lyness and his coauthors, which may partly explain why, amid many publications that discuss algorithms of type C, it is hard to find many that discuss algorithms of type P. Yet for $z \neq 0$, not only are P and C different, but P is much more accurate for $|z| \approx 1$ (Theorem 2.1). At the data points $\{z_k\}$ themselves, P returns exactly correct results whereas C makes errors of infinite magnitude.

A practical situation where evaluation of a function f from samples on a circle is useful is in problems where accurate direct computation of $f(z)$ in floating-point arithmetic is impossible because of rounding errors. For an elementary example, consider the evaluation of a "phi function" like $(e^z - 1 - z)/z^2$, a problem of practical importance not only for scalars z but also matrices [44, 68].

2.2. Barycentric formulas for the two algorithms. To derive formulas for the polynomial interpolant p of algorithm P, one may first define the node polynomial

$$\ell(z) = \prod_{k=0}^{n-1} (z - z_k) = z^n - 1 \tag{2.2}$$

and the *barycentric weights*

$$\lambda_k = \frac{1}{\ell'(z_k)} = \frac{z_k}{n}. \tag{2.3}$$

These give the degree $n - 1$ *cardinal polynomial*

$$\ell_k(z) = \ell(z)\frac{\lambda_k}{z - z_k} = \frac{1}{n}\,\ell(z)\frac{z_k}{z - z_k} \tag{2.4}$$

for any k, taking the values 1 at $z = z_k$ and 0 at the other roots of unity. From (2.4) it follows that the interpolant p can be written in Lagrange form as

$$p(z) = \frac{1}{n}\,\ell(z) \sum_{k=0}^{n-1} \frac{z_k f_k}{z - z_k}, \tag{2.5}$$

and this is the *barycentric formula of the first kind.* Here it is to be understood that if $z = z_k$ for some k, then the use of the formula is replaced by the exact value f_k. A similar qualification applies to (2.4) and other formulas in this paper involving divisions by $z - z_k$. The resulting polynomial evaluation algorithms are known to be numerically stable [33].

In particular, if $f \equiv 1$, then its polynomial interpolant is the constant function $p \equiv 1$, which we can write in the form (2.5) as

$$1 = \frac{1}{n}\,\ell(z) \sum_{k=0}^{n-1} \frac{z_k}{z - z_k}. \tag{2.6}$$

If (2.5) is divided by (2.6), the factors $n^{-1}\ell(z)$ in the numerator and denominator cancel, giving the elegant identity

$$p(z) = \sum_{k=0}^{n-1} \frac{z_k f_k}{z - z_k} \Bigg/ \sum_{k=0}^{n-1} \frac{z_k}{z - z_k}, \tag{P}$$

the *barycentric formula of the second kind,* or in Matlab,

```
[P]        pz = mean(zk.*fk./(z-zk))/mean(zk./(z-zk)).
```

The work involved is $O(n)$ operations. Here and in all the Matlab formulas of this article, `zk` is a column vector containing the nth roots of unity, $z_k = \exp(2\pi i k/n)$, $0 \le k \le n-1$, and `fk` is the corresponding column vector of values f_k.

The use of the barycentric formula falls in the category of a calculation "by values." It is equally possible to evaluate $p(z)$ "by coefficients." The Taylor coefficients can be computed by the fast Fourier transform (FFT), and the algorithm for evaluating $p(z)$ becomes

```
[P*]        c = fft(fk)/n,  pz = polyval(flipud(c),z),
```

requiring $O(n \log n)$ operations, with $O(n)$ additional operations for each point z after the first one. The formulas behind `[P*]` appear in the next section.

We now turn to algorithm C. The trapezoidal rule formula is [56]

$$r(z) = -\frac{1}{n} \sum_{k=0}^{n-1} \frac{z_k f_k}{z - z_k}, \tag{C}$$

or in Matlab,

```
[C]        rz = -mean(zk.*fk./(z-zk)),
```

again with $O(n)$ operations. Comparing (C) with (2.5), we see that the discrete Cauchy integral approximation to $f(z)$ is

$$r(z) = -\frac{p(z)}{\ell(z)}, \tag{2.7}$$

and since $-1/\ell(z) = b(z)$, this confirms (2.1). Note that r is a rational function of type $(n-1, n)$, meaning that it has at most $n-1$ finite zeros (unless it is identically zero) and at most n finite poles, counted with multiplicity, with a zero of order at least 1 at $z = \infty$.

2.3. Taylor projections and polynomial interpolants. Suppose f is analytic in D_R for some $R > 1$. Then it has a Taylor series

$$f(z) = \sum_{k=0}^{\infty} a_k z^k, \tag{2.8}$$

with coefficients given by the Cauchy integrals

$$a_k = \frac{1}{2\pi i} \int_S \zeta^{-k-1} f(\zeta)\, d\zeta. \tag{2.9}$$

Since f is bounded in D_ρ for any $\rho < R$, Cauchy's estimate gives

$$|a_k| = O(\rho^{-k}), \quad k \to \infty. \tag{2.10}$$

If we truncate the series to the degree $n-1$ Taylor section

$$f_{n-1}(z) = \sum_{k=0}^{n-1} a_k z^k, \tag{2.11}$$

it follows that for any $z \in D_R$ and ρ with $|z| < \rho < R$ we have

$$|f_{n-1}(z) - f(z)| = O((|z|/\rho)^n), \quad n \to \infty, \tag{2.12}$$

uniformly for all z in each closed disk of radius $<\rho$ about the origin.

In this paper our interest is not Taylor sections f_{n-1} but polynomial interpolants p_{n-1}. (We use the labels p and p_{n-1} interchangeably.) If we write p_{n-1} in the form

$$p_{n-1}(z) = \sum_{k=0}^{n-1} c_k z^k, \tag{2.13}$$

then the coefficients $\{a_k\}$ and $\{c_k\}$ are related by the aliasing identity

$$c_k = a_k + a_{k+n} + a_{k+2n} + \cdots, \quad 0 \le k \le n-1. \tag{2.14}$$

In words, p_{n-1} is obtained from f by replacing each term $a_k z^k$ in the series (2.8) by its alias $a_k z^{k(\mathrm{mod}\ n)}$. By (2.10) this implies

$$|p_{n-1}(z) - f(z)| = O(\rho^{-n}), \quad |z| \le 1 \tag{2.15}$$

as $n \to \infty$ for any $\rho < R$, since the leading term of the error is the first of the aliases, $a_n z^0$, and $a_n = O(\rho^{-n})$. For $1 < |z| < R$ the leading term of the error becomes $-a_n z^n$, of order $O(|z|^n \rho^{-n})$, so we get

$$|p_{n-1}(z) - f(z)| = O(|z|^n \rho^{-n}), \quad 1 \le |z| < \rho \tag{2.16}$$

for any $\rho < R$. Again these bounds hold uniformly for all z in each closed disk of radius $<\rho$ about the origin.

If the Taylor coefficients $\{a_k\}$, $0 \le k \le n-1$ are approximated by applying the trapezoidal rule to (2.9), the same aliasing occurs, and so the resulting coefficients are $\{c_k\}$. It is this property that led to the FFT-based algorithm `[P*]`, which utilizes the fact that the FFT computes all n trapezoidal rule approximations at once.

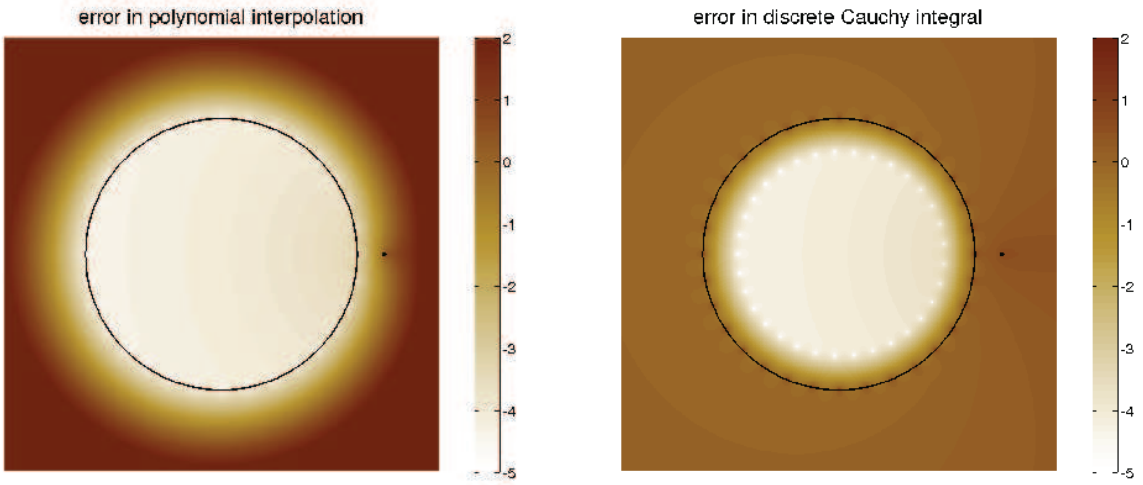

FIG. 2.1. *Error plot for approximations with $n = 32$ to $f(z) = \log(1.2 - z)$ by algorithms P (left) and C (right), with the unit circle marked by a black curve and the branch point at $z = 1.2$ marked by a dot. The color bars represent base 10 logarithms. As established in Theorem 2.1, algorithm P converges throughout $D_{1.2}$ as $n \to \infty$, whereas algorithm C converges only in D. Faint white dots reflect exact interpolation at the $n = 32$ roots of unity on the left, and on the right, at $n + 1 = 33$ points close to a smaller circle (see §2.9). At the roots of unity on the right, the errors are infinite because of the poles introduced by the trapezoidal rule.*

2.4. Accuracy of the two algorithms. The observations of the foregoing pages are summarized in the following theorem. For a general discussion of the geometric convergence of the trapezoidal rule for periodic analytic integrands, see [76].

THEOREM 2.1. *Let f be analytic in D_R for some $R > 1$. Then as $n \to \infty$, for any ρ with $1 < \rho < R$, the approximation of algorithm* P *has accuracy*

$$|p(z) - f(z)| = \begin{cases} O(\rho^{-n}) & |z| \le 1, \\ O(|z|^n \rho^{-n}) & 1 \le |z| < \rho, \end{cases} \tag{2.17}$$

whereas algorithm C gives

$$|r(z) - f(z)| = \begin{cases} O(\rho^{-n}) & |z| \le R^{-1}, \\ O(|z|^n) & R^{-1} \le |z| < 1. \end{cases} \tag{2.18}$$

These bounds hold uniformly for all z in each closed disk of radius $< \rho$ about the origin.

Proof. Equation (2.17) was already given as (2.15) and (2.16). Equation (2.18) follows from this together with (2.1), since $1 - (1 - z^n)^{-1} = O(|z|^n)$ as $n \to \infty$ for each z with $|z| < 1$. □

Let us look at these conclusions in a numerical example. In Figure 2.1, the function $f(z) = \log(1.2 - z)$ has been approximated by algorithms P and C with $n = 32$. The figure plots the error $|f(z) - p(z)|$ or $|f(z) - r(z)|$ in each approximation, revealing that P is accurate in a region considerably larger than the unit disk. This phenomenon, which is predicted by (2.17), is known as *overconvergence*. Method C, on the other hand, only converges in the unit disk, and near the unit circle it loses accuracy, as predicted in (2.18). To be precise, the estimates indicate that P is more accurate than C for $|z| > R^{-1}$.

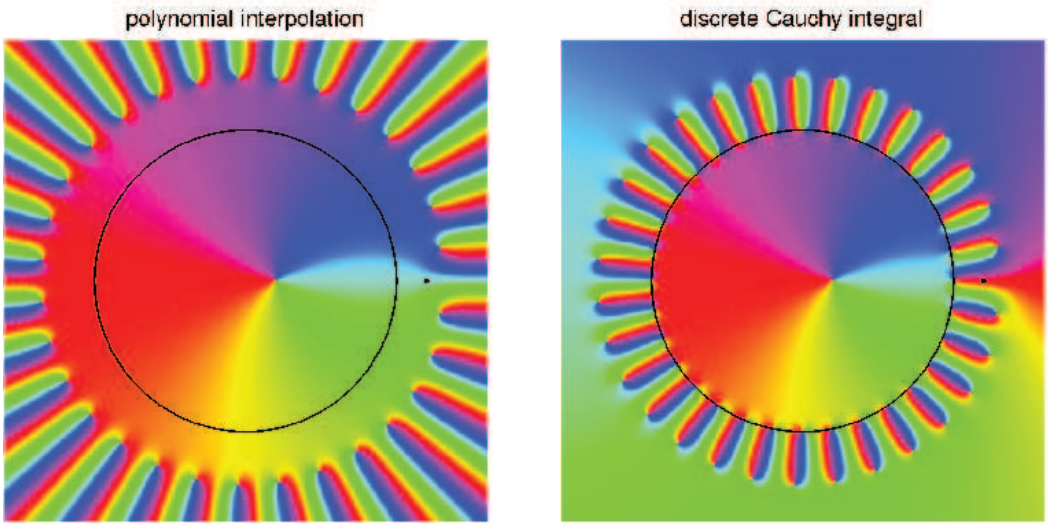

FIG. 2.2. *Phase portraits (see* [77]*) for the same two approximations as in Fig.* 2.1. *The image on the left shows* $n-2$ *zeros of* p *converging as* $n \to \infty$ *to* $|z| = R = 1.2$*, the circle of analyticity for this function, in keeping with Jentzsch's theorem and its generalization by Walsh (*§2.5*). (The* $(n-1)$*st zero converges to the zero of* f *at* $z = 0.2$*.) The image on the right shows exactly the same zeros and in addition the* n *poles at the* n*th roots of unity of the approximation* $r(z) = b(z)p(z)$*.*

Figure 2.2 displays the approximations $p(z)$ and $r(z)$ themselves, rather than the associated errors. These images are *phase portraits,* depicting the phase but not the amplitude of each function: red for positive real, green for positive imaginary, cyan for negative real, and violet for negative imaginary [77]. Within the unit disk, it is apparent that both algorithms approximate the same function f, whose zero at $z = 0.2$ shows up as a point where all the colors meet. The differences between the two figures near and outside the unit circle, on the other hand, are pronounced.

The theorems and figures of this section highlight an effect that was foreshadowed in Figure 1: a Cauchy integral over a contour Γ does not just approximate a function in the region enclosed by Γ; it acts to *separate* that region from the exterior. If such a separation is not part of one's computational purpose, then the discrete Cauchy integral may not be the best algorithm.

2.5. Zeros, poles, and the Jentzsch and Walsh theorems. A striking feature of Figure 2.2 is the ring of colorful stripes in each image outside the unit disk. In both cases, these stripes end at $n-2 = 30$ zeros lying approximately on the circle $|z| = 1.3$, which will shrink to $|z| = 1.2$ as $n \to \infty$. For the polynomial approximant $p(z)$ on the left, the stripes extend outward to $z = \infty$, where they meet at the pole of multiplicity $n-1 = 31$. For the rational approximant $r(z)$ on the right, on the other hand, they point inward and terminate at the $n = 32$ poles on the unit circle.

The appearance of a ring of zeros near the circle of convergence of a Taylor series is a well-known phenomenon. According to *Jentzsch's theorem* [42], if f is analytic in D_R but not $\overline{D}_R$, the zeros of Taylor sections f_{n-1} cluster at every point on the circle $|z| = R$ as $n \to \infty$. (This is a generalization of the phenomenon that the zeros of the partial sum $1 + z + \cdots + z^{n-1}$ of the Taylor series of $(1-z)^{-1}$ are the nth roots of unity except $z = 1$.) In the present case Jentzsch's theorem is inapplicable, because the polynomials p_{n-1} are interpolants in roots of unity rather than Taylor sections. However, a generalization due to Walsh asserts the same conclusion for any sequence of polynomials that is *maximally convergent* on D, meaning $\limsup_{n\to\infty} \|p_{n-1} - f\|_D^{1/n} = R^{-1}$, where $\|\cdot\|_D$ is the supremum norm on D [78]. By Theorem 2.1, polynomial

interpolants in roots of unity are maximally convergent, and thus Walsh's theorem applies to Figure 2.2. As $n \to \infty$, zeros will cluster near every z with $|z| = 1.2$.

2.6. Cauchy integrals of the second kind. The developments of §2.2 suggest a curious observation. In analogy to our progression from (2.5) to (P), we could consider the "Cauchy integral of the second kind"

$$f(z) = \int_S \frac{f(\zeta)}{z-\zeta}\,d\zeta \,\Big/ \int_S \frac{1}{z-\zeta}\,d\zeta\ , \tag{2.19}$$

obtained by dividing the usual Cauchy integral formula for f by the same formula for the case $f \equiv 1$. Discretizing both the numerator and the denominator of this quotient by the trapezoidal rule then suggests a "discrete Cauchy integral of the second kind,"

$$\tilde{r}(z) = \sum_{k=0}^{n-1} \frac{z_k f_k}{z-z_k} \,\Big/ \sum_{k=0}^{n-1} \frac{z_k}{z-z_k}\ . \tag{2.20}$$

But this is exactly (P)! Thus the discrete Cauchy integral and the polynomial interpolation ideas give the same result after all, if one realizes the former by a nonstandard "second kind" discretization. As Helsing and Ojala put it in the context of an analogous formula for integral equations [30], "One could say that the denominator in this formula compensates for the error in the numerator." This kind of compensation is familiar to devotees of barycentric formulas, which look numerically unstable at first sight because of their poles but are in fact stable [33].

This multiplicative derivation of (P) may appear rather magical, but a third and more straightforward additive realization of the Cauchy integral idea leads to the same formula. Note that unlike $f(s)/(z-s)$, the quotient $(f(z)-f(s))/(z-s)$ has no pole at $s = z$, so it is an analytic function of $s \in D_R$, whose Cauchy integral must be equal to zero. A trapezoidal rule discretization of that Cauchy integral accordingly gives

$$-\frac{1}{n}\sum_{k=0}^{n-1} \frac{z_k(f(z)-f(z_k))}{z-z_k} \approx 0, \tag{2.21}$$

and now the summands do not diverge as $z \to z_k$, so one can expect the trapezoidal rule to be more accurate than in (C). Solving for $f(z)$ in (2.21) gives (2.20) again. This argument has been employed by various authors to derive improved discretizations of integrals near contours [30, 38]. Ioakimidis calls it "the use of the Cauchy theorem instead of the Cauchy formula" [37, 38].

2.7. Derivatives of f or Taylor coefficients. The ideas we have considered for the evaluation of $f(z)$ extend readily to the evaluation of derivatives $f^{(m)}(z)$ for any $m \ge 0$, or equivalently, the computation of Taylor coefficients. Algorithms P and C have the following analogues:

ALGORITHM $\mathrm{P}^{(m)}$. *Approximate $f^{(m)}(z)$ by $p^{(m)}(z)$, where $p \in P_{n-1}$ is the polynomial interpolant to f in $\{z_k\}$.*

ALGORITHM $\mathrm{C}^{(m)}$. *Approximate $f^{(m)}(z)$ by the trapezoidal rule discretization in $\{z_k\}$ of the integral $m!(2\pi i)^{-1}\int_S (\zeta - z)^{-m-1} f(\zeta)\,d\zeta$.*

The same convergence bounds hold as in Theorem 2.1, with $\mathrm{P}^{(m)}$ more accurate than $\mathrm{C}^{(m)}$ and converging in a larger disk.

Matlab generalizations of [P], [P*], and [C] can be written as follows for the case $m = 1$, the first derivative. Formula [P'] evaluates the derivative by a barycentric formula, a technique that originates with [69].

```
[P']    pz = mean(zk.*fk./(z-zk))/mean(zk./(z-zk))
        ppz = mean(zk.*(pz-fk)./(z-zk).^2)/mean(zk./(z-zk)),

[P'*]   c = fft(fk)/n,  cp = (1:n-1)'.*c(2:end)
        ppz = polyval(flipud(cp),z),

[C']    rpz = mean(zk.*fk./(z-zk).^2).
```

We shall not discuss these methods at any length, since the mathematics is so close to the case $m = 0$. However, it is worth noting that it is in the context of derivatives, not function values, that numerical methods based on complex samples are best known, since they eliminate the instabilities that afflict real finite differences. In the words of Lyness and Moler [57], "Once complex arguments are allowed, the principal difficulties encountered in numerical differentiation simply disappear."

2.8. Scaling of the radius. In this paper we work always with the unit disk and unit circle, but in practice, it is often desirable to vary the radius. According to (2.17), algorithm P converges approximately at the rate $O(R^{-n})$. It follows that if z is rescaled by a factor $\tau < 1$, so that f is sampled on the circle $|z| = \tau$, the convergence rate will improve to $O((\tau/R)^n)$. On the other hand, as τ gets smaller, problems appear of ill-conditioning and rounding errors in floating-point arithmetic. The computation is particularly problematic when one wants to obtain higher Taylor coefficients of f and do so with high relative as opposed to just absolute accuracy. These effects have been noted in this subject from the start, and in the 1980s Fornberg developed an algorithm for determining a good radius adaptively [21]. More recently Bornemann has published a theory showing that an optimal choice of τ can often eliminate ill-conditioning completely even for very high order derivatives [9].

An interesting extreme case in the matter of choice of radius arises in the method of "complex step differentiation" for computing the first derivative [72]. Here the grid size is $n = 2$ (using the midpoint rather than trapezoidal rule, i.e., an imaginary rather than real finite difference), and the radius τ is taken smaller than machine epsilon.

2.9. The method of fundamental solutions. Let us look again at the faint white spots in Fig. 2.1, showing points of interpolation. On the left, f is approximated by a polynomial $p \in P_{n-1}$, which we can think of as a rational function with $n-1$ poles at $z = \infty$. The particular approximation in this class is determined by the condition of interpolation of f at the nth roots of unity. On the right, f is a approximated by a rational function with n poles at the roots of unity, and the white spots show that it interpolates f in a ring of $n + 1$ points lying approximately on a circle of radius 0.8.

These considerations suggest a unifying view of algorithms P and C: choose a set of basis functions $\{(z - \zeta_k)^{-1}\}$ defined by a fixed set of poles $\{\zeta_k\}$, and approximate f by a linear combination of these basis functions determined by interpolation. To fit the framework of this paper, the interpolation points must be the roots of unity. The two algorithms now begin to look like extremes of a continuum, with poles at ∞ or on the unit circle. In-between choices could also be considered, such as poles distributed along a circle such as $|z| = 1.1$. This is the *method of fundamental solutions,* also

known as the *charge simulation method* [4] (and related to the *Trefftz method*). If one generalizes the idea to the Helmholtz equation, for example, then the appropriate fundamental solutions become Hankel functions rather than simple poles.

The reason why it may be advantageous to place poles near S rather than at $z = \infty$ has to do with ill-conditioning. If one places poles on the circle $|z| = \tau$ for some τ, the interpolation problem becomes exponentially ill-conditioned as τ increases, making large finite values of τ unworkable. It happens that $\tau = \infty$ is well-behaved, because we can represent polynomials by other means than as linear combinations of simple poles, but such well-behaved representations are not always available in more general situations, such as problems on non-circular domains. And so the idea of placing singularities near the boundary of the problem domain is a powerful one. There is always a tradeoff between wanting singularities well separated from the boundary, for accuracy, and close to the boundary, for conditioning.

Related issues of ill-conditioning arise in the use of radial basis functions (RBFs) for approximating functions f or solving differential equations. RBFs differ from fundamental solutions in that they do not exactly satisfy the differential equation under consideration; on the other hand they need not have singular points, and interpolation points can be distributed inside the problem domain as well as outside it. In analogy to the radius τ for the method of fundamental solutions, there is now a smoothness parameter ε, and the limit $\varepsilon \to 0$ is associated with the better approximations but worse conditioning.

3. Zerofinding in the unit disk.

3.1. Two algorithms for zerofinding. We now turn to the second fundamental problem of this paper, the calculation of the zeros of f within D. This is an inverse of the former problem: instead of determining the value of f at a given point, now we want to find points where f takes a given value. For clarity we will use the letter w to denote a zero. Once again, we begin with two basic ideas, polynomial interpolation and discretized Cauchy integrals. We shall see that this time, the Cauchy integral has an advantage when n is large.

The first idea can be stated immediately. Curiously, there seems to be no literature on the following simple algorithm.

ALGORITHM P^Z: POLYNOMIAL INTERPOLATION. *Approximate f by the polynomial interpolant $p(z)$ in $\{z_k\}$ and compute the zeros of p that lie in D.*

The zeros of p can be calculated by computing its coefficients with the discrete Fourier transform, then finding its zeros by solving a companion matrix eigenvalue problem. In Matlab, the eigenvalue computation is invoked by the `roots` command:

```
[Pz*]    c = fft(fk)/n, w = roots(flipud(c)), w = w(abs(w)<1).
```

Alternatively, working from values rather than coefficients [24], one can solve an $(n+1) \times (n+1)$ generalized eigenvalue problem with arrowhead structure [1, 11]. (This approach produces two spurious eigenvalues at ∞ as well as the $n-1$ eigenvalues corresponding to zeros of p.) The following particularly elegant symmetric form has been suggested by Townsend [74]:

```
[Pz]     A = [0 fk.'; fk diag(fk)], B = diag([0; fk./zk])
         w = eig(A,B),  w = w(abs(w)<1).
```

Both `[Pz*]` and `[Pz]` require $O(n^3)$ operations.

Like algorithms P and $\mathrm{P}^{(m)}$, algorithm P^Z converges geometrically, at least when the zeros are simple.

THEOREM 3.1. *Let f be analytic in D_R for some $R > 1$ with exactly K zeros in D, all of them simple, and no zeros with $|z| = 1$. Then for all sufficiently large n, Algorithm P^Z produces exactly K zeros in D, which converge to the zeros of f at the rate $O(\rho^{-n})$ as $n \to \infty$ for any $\rho < R$.*

Proof. This follows from (2.17) and the fact that simple zeros of analytic functions vary smoothly with analytic perturbations. □

Another approach to zerofinding is to use a discretized contour integral. Many different integrals are suitable, and we begin with the one that has had the most attention in the literature, involving the logarithmic derivative f'/f. This approach has roots in the work of Delves and Lyness and was developed more fully by Kravanja and Van Barel, in collaboration with Sakurai [47], [49, §1.2]. The next subsection considers the alternative choice $1/f$, a derivative-free variant developed by Kravanja and Van Barel [48], [49, §1.6].

To simplify the discussion, we begin by assuming that f has exactly one zero w in D and no zeros with $|w| = 1$. In this case, w is given by the integral

$$w = \frac{1}{2\pi i} \int_S \frac{\zeta f'(\zeta)}{f(\zeta)}\, d\zeta. \tag{3.1}$$

This formula goes back to the 19th century and was proposed in the computer era by Delves and Lyness and McCune [14, 59]. In analogy to (C) we get

$$w \approx \frac{1}{n} \sum_{k=0}^{n-1} \frac{z_k^2 f_k'}{f_k}. \tag{3.2}$$

For some problems, samples of f_k' are available and this formula can be used as written. Our convention in this paper, however, is that just $\{f_k\}$ are known, in which case $\{f_k'\}$ can be approximated by differentiating the polynomial interpolant as in `[P']` or `[P'*]`. This gives the following algorithm:

[McCune]
```
c = fft(fk)/n, cp = (1:n-1)'.*c(2:end)
ppzk = n*ifft([cp;0]), w = mean(zk.^2.*ppzk./fk)
```

Algorithms `[Pz]`/`[Pz*]` and `[McCune]` both work excellently in certain circumstances and converge geometrically, with `[Pz]`, as in §2.4 and §2.7, having the better convergence constant. A new feature arises now, however, that gives `[McCune]` an advantage. As mentioned above, `[Pz]` and `[Pz*]` require $O(n^3)$ operations when implemented by general-purpose solvers, whereas the operation count for `[McCune]` is just $O(n \log n)$. For modest values of n like 32 or 64, this difference may not be very important, but when n is in the hundreds, it becomes significant. For example, the function $f(z) = \log(1.1 - z)$ has a zero $w = 0.1$. For 10-digit accuracy, algorithms `[Pz]`/`[Pz*]` require $n = 190$ sample points and 0.5 secs. on a 2012 desktop computer, whereas `[McCune]` requires $n = 240$ points but less than 0.001 sec.

This difference between algorithms `[Pz]`/`[Pz*]` and `[McCune]` is rooted in the distinction mentioned in the Introduction, Cauchy integrals' property of separating the region inside the contour from the region outside. Algorithm `[Pz]` computes all the zeros of a polynomial interpolant, whereas algorithm `[McCune]` first projects the

problem onto the unit disk and then computes only the zeros there. For a problem that is complicated in the large but relatively simple in the disk, this may be much more efficient.

It remains to generalize [McCune] to functions with more than one zero in D. Following [49, §1.2], we begin with a generalization of (3.1). Suppose f has K zeros $w_1, \ldots, w_K$ in D, which we assume to be simple, and no zeros on S.[3] Then for any integer $m \geq 0$,

$$s_m := \frac{1}{2\pi i} \int_S \frac{\zeta^m f'(\zeta)}{f(\zeta)}\, d\zeta = \sum_{j=1}^{K} w_j^m. \tag{3.3}$$

Taking $m = 2$ gives the sum of the squares of the zeros, $m = 3$ gives the sum of the cubes, and so on; also $m = 0$ recovers a familiar formula for counting the number of zeros. (These "moment integrals" are the starting point of a theory of *formal orthogonal polynomials* [15] with respect to the weight function $f'(\zeta)/f(\zeta)$ [47].) Now consider the $K \times K$ Hankel matrices

$$H = \begin{pmatrix} s_0 & \cdots & s_{K-1} \\ \vdots & & \vdots \\ s_{K-1} & \cdots & s_{2K-2} \end{pmatrix}, \quad H^< = \begin{pmatrix} s_1 & \cdots & s_K \\ \vdots & & \vdots \\ s_K & \cdots & s_{2K-1} \end{pmatrix}. \tag{3.4}$$

The roots $w_1, \ldots, w_K$ are precisely the eigenvalues λ of the generalized eigenvalue problem

$$H^< x = \lambda H x. \tag{3.5}$$

To show this, note that the Hankel matrices can be factored as

$$H = VV^T, \quad H^< = V \mathrm{diag}(w_1, \ldots, w_K) V^T \tag{3.6}$$

where V is the Vandermonde matrix

$$V = \begin{pmatrix} 1 & \cdots & 1 \\ w_1 & \cdots & w_K \\ \vdots & & \vdots \\ w_1^{K-1} & \cdots & w_K^{K-1} \end{pmatrix}. \tag{3.7}$$

It is readily verified that the jth column of V^{-T} is an eigenvector x in (3.5) with eigenvalue $\lambda = w_j$. If the integrals are approximated by the trapezoidal rule in roots of unity, with f'_k approximated by polynomial interpolation as in [McCune], this gives us an algorithm for approximating K distinct roots of f in D. The following is a Matlab realization. Because of the use of the FFT, this algorithm assumes $n \geq 2K$, which is not much of a restriction since we would normally expect to have $n \gg K$ when applying any of the algorithms of this article. The same condition arises in the error analysis of this algorithm published in [65].

[3] In practice one would need to estimate K, e.g. via the identity $K = (2\pi i)^{-1} \int_S f'(\zeta)/f(\zeta) d\zeta$. There is a literature on this problem going back at least to [71]. See §1.1.1 of [49].

```
                c = fft(fk)/n, cp = (1:n-1)'.*c(2:end)
                ppzk = n*ifft([cp;0]), s = ifft(ppzk./fk)
[Cz1]           H = hankel(s(2:K+1), s(K+1:2*K))
                H2 = hankel(s(3:K+2), s(K+2:2*K+1))
                w = eig(H2,H)
```

Apart from its use of approximations to f_k' rather than exact values, this algorithm is due to Kravanja and Van Barel in collaboration with Sakurai [47], based on earlier work by Delves and Lyness [14].

Although we have presented `[Cz1]` as a generalization of `[McCune]` to multiple roots, in fact it is different in the case $K = 1$, and more accurate. This is because its generalized eigenvalue problem gives it the "second kind" nature of a quotient of two trapezoidal approximations, as in (2.20), rather than a single trapezoidal approximant, as in (2.5). For $K = 1$, the estimate of w delivered by `[Cz1]` is equal to the estimate delivered by `[McCune]` divided by a constant close to 1, namely the trapezoidal approximation to $s_0 = 1$, the number of zeros inside the unit disk.

As we shall see in the numerical experiments of §4.2, the assumption of distinct zeros used in deriving `[Cz1]` is an essential one. As explained in [49], nearly-equal zeros cause instability for this algorithm, and additional formulas must be used to treat cases with confluent zeros.

3.2. Cauchy integrals involving $1/f$ instead of f'/f. In the Cauchy integral method just described, values of the derivative f' were in principle required and approximated in practice by polynomial interpolation. Other contour integrals can be used used for zerofinding, however, that do not involve f'. For the case where f has just one zero in D, an integral formulation analogous to (3.1) is

$$w = \int_S \frac{\zeta}{f(\zeta)}\, d\zeta \,\Big/ \int_S \frac{1}{f(\zeta)}\, d\zeta\ , \tag{3.8}$$

proposed by Luck and Stevens [54]. The trapezoidal discretization takes this form:

```
[LuckStevens]        w = mean(zk.^2./f)/mean(zk./f).
```

Note that unlike `[McCune]`, `[LuckStevens]` contains a denominator sum, giving it the flavor of a "type 2" discretization. As a consequence, the generalization to the case of K zeros will reduce exactly to `[LuckStevens]` in the case $K = 1$.

That generalization is surprisingly easy: we just delete the f' terms from the algorithm of §3.1! Following [49, §1.6], let us redefine the numbers s_m by replacing the integral (3.3) by

$$s_m := \frac{1}{2\pi i} \int_S \frac{\zeta^m}{f(\zeta)}\, d\zeta = \sum_{j=1}^{K} \frac{w_j^m}{f'(w_j)}, \tag{3.9}$$

assuming the roots are simple. From here the developments proceed as before. Again the zeros of f are the eigenvalues of the generalized eigenvalue problem (3.5); the derivation is the same as before, except with the jth column of V divided by $(f'(w_j))^{1/2}$. (Any choice of square roots will do. There is a much more general structure to these problems; see [5] and [49].) The trapezoidal rule in roots of unity gives accurate approximations, and can be realized as follows in Matlab:

```
                s = ifft(1./fk)
[Cz2]           H = hankel(s(2:K+1), s(K+1:2*K))
                H2 = hankel(s(3:K+2), s(K+2:2*K+1))
                w = eig(H2,H)
```

Unlike [Cz1], algorithm [Cz2] is able to calculate nearly equal or multiple zeros, as we shall see in the numerical experiments of §4.2. Both algorithms involve Hankel matrices that may be ill-conditioned, however, so they are not stable in all cases.

4. Polefinding in the unit disk. To find zeros of an analytic function f, algorithms [Cz1] and [Cz2] applied Cauchy integrals to the functions f'/f and $1/f$, respectively. However, one might say that at bottom, any algorithm utilizing Cauchy integrals is really a "polefinding" algorithm. Following this line of thinking, suppose we are given a function g that is meromorphic in D_R for some $R > 1$, having exactly K poles there, which we assume are all in D. We could then consider the problem of calculating these K poles from samples of g at roots of unity. Note that this problem is not quite equivalent to the problem of the last section with $g = 1/f$, since g may have both poles and zeros, whereas f was assumed to have no poles.

4.1. Two algorithms for polefinding. Again we shall distinguish two approaches: interpolation and Cauchy integrals. The new feature that appears is that for polefinding, it is natural to interpolate by rational functions rather than polynomials.

Algorithm R^P: Linearized rational interpolation. *Assuming g has K poles in D_R, approximate it by $r(z) = p(z)/q(z)$, where $p \in P_{n-1-K}$, $q \in P_K$ is monic, and the equation $g(z_k)q(z_k) = p(z_k)$ holds at each of the nth roots of unity; then compute the zeros of q in D.*

(Generically, the linearized rational interpolant is a true rational interpolant, but interpolation may fail in cases where $q(z_k) = 0$.) Algorithms for computing r and a corresponding Matlab code `ratdisk` are given in [27], based on a singular value decomposition of a $K \times (K+1)$ rectangular Toeplitz matrix of discrete Fourier coefficients of g. Using these tools, polefinding by linearized rational interpolation reduces to a single call to `ratdisk`:

```
[ratdisk]       [r,a,b,mu,nu,w] = ratdisk(gk, n-1-K, K)
```

(The same effect is also available via `ratinterp(gk,n-1-K,K,[],'unitroots')` in Chebfun.) Here `gk` represents a column vector of samples of g at the roots of unity. If our aim is to find zeros of an analytic function f, we precede this command by `gk = 1./fk`.

Alternatively, we can compute poles by Cauchy integrals. The natural approach here is to use algorithm [Cz2], with $1/f$ relabeled as g. For completeness we give this formulation of the algorithm the new name [Cp]:

```
                s = ifft(gk)
[Cp]            H = hankel(s(2:K+1), s(K+1:2*K))
                H2 = hankel(s(3:K+2), s(K+2:2*K+1))
                w = eig(H2,H)
```

4.2. Numerical illustration of five algorithms for zerofinding. We have presented five algorithms that can be used for zerofinding, with Matlab names `[Pz]` and `[Pz*]` (these two are equivalent), `[Cz1]`, `[Cz2]` (or `[Cp]`), and `[ratdisk]`. To illustrate the behavior of these algorithms, we apply them with $n = 50$ to the function $f(z) = \sin(z - 0.3)\log(1.2 - z)$, which has two nearby zeros in D at 0.2 and 0.3. The first four algorithms find zeros of f directly, and `[ratdisk]` finds poles of $g = 1/f$. Here are the results:

```
                  Arrowhead matrix, Algorithm [Pz]: 0.20002100, 0.29997794
                 Companion matrix, Algorithm [Pz*]: 0.20002100, 0.29997794
          Cauchy integrals in f'/f, Algorithm [Cz1]: 0.20468368, 0.30432215
           Cauchy integrals in 1/f, Algorithm [Cz2]: 0.20000116, 0.29999865
Rational interpolation of 1/f, Algorithm [ratdisk]: 0.20000116, 0.29999865
```

These numbers reveal several interesting properties. First, `[Pz]` and `[Pz*]` give the same results, since they are mathematically equivalent. Second, `[Cz1]` gives far worse results. This is a consequence of the nearly-equal zeros; this algorithm must be modified in such cases, as explained in [49]. Third, `[Cz2]` and `[ratdisk]` also give the same results. This is a new observation (although it can be found between the lines in §2.3 of [49]), which we explain in the next section. Fourth, this last pair of algorithms do not suffer from the nearly-equal zeros, and indeed their results are slightly more accurate than those of `[Pz]`/`[Pz*]`. As a rule, we believe that linearized rational interpolation is quite a good method for approximating zeros of analytic functions, by regarding them as poles of their reciprocals.

4.3. Equivalence of linearized rational interpolation and a discretized contour integral algorithm. The two polefinding algorithms we have described, `[ratdisk]` and `[Cp]`, are mathematically equivalent. We now outline this connection.

Given $n \geq 1$ and $K < n$, let g be a function defined at the nth roots of unity, let $q(z) = \sum_{k=0}^{K} b_k z^k$ be a polynomial in P_K, and let $p(z) = \sum_{k=0}^{n-1} a_k z^k$ be the polynomial in P_{n-1} that interpolates $g(z)q(z)$ at the roots of unity. We know that p/q corresponds to a linearized rational interpolant to g if p belongs to P_{n-1-K}.

Since $p \in P_{n-1}$, its coefficients $a_0, \ldots, a_{n-1}$ can be calculated by interpolation in the roots of unity: a discrete Fourier transform of the data $g(z_k)q(z_k)$. The condition for p/q to be a linearized rational interpolant to g is $a_{n-K} = a_{n-K+1} = \cdots = a_{n-1} = 0$, that is, K equations in the $K + 1$ unknowns $b_0, \ldots, b_K$. This rectangular linear system of equations always has a nontrivial solution, and `ratdisk` finds such a solution by computing a singular value decomposition.

Algorithm `[Cp]` is based on an alternative interpretation of the same conditions $a_{n-K} = a_{n-K+1} = \cdots = a_{n-1} = 0$. If p has degree $< n - 1$, its discrete contour integral in the nth roots of unity must be zero: $\sum_{k=0}^{n-1} z_k p(z_k) = \sum_{k=0}^{n-1} z_k\, g(z_k)q(z_k) = 0$. But for any $j < K$, $z^j\, p(z)$ still has degree $< n - 1$, so we actually have

$$\sum_{k=0}^{n-1} z_k^{j+1} g(z_k)q(z_k) = 0, \quad 0 \leq j \leq K - 1. \tag{4.1}$$

That is, q is orthogonal to $1, z, \ldots, z^{K-1}$ with respect to a discrete symmetric bilinear form supported at the roots of unity (not quite an inner product, since it lacks positive-definiteness and conjugate-symmetry). We say that q is the degree K *discrete formal orthogonal polynomial* defined by the weights $z_k\, g(z_k)$ at the nth roots of unity. From here, standard manipulations bring us to a generalized eigenvalue problem involving

Hankel matrices of discrete moments, as described in equations (3.4)–(3.7) above, with the eigenvalues being the roots of q.

In a word, `[ratdisk]` determines q by its coefficients and `[Cp]` by its zeros.

5. Computing the eigenvalues of a matrix in the unit disk. Let A be a square matrix of dimension N. How can we find its eigenvalues in the unit disk? This question is a natural extension of the topics addressed up to now in this paper, and it points toward major challenges of computational science, where A will typically be an approximation with $N \gg 1$ to a partial differential or other linear operator. For simplicity we speak of the standard eigenvalue problem $Ax = \lambda x$, though most of the ideas carry over to the generalized eigenvalue problem $Ax = \lambda Bx$.

One idea with a zerofinding flavor would be to compute zeros of the determinant, i.e., zeros of the function

$$f(z) = \det(zI - A). \tag{5.1}$$

To do this, we could evaluate $f(z)$ at the nth roots of unity and then apply any of the algorithms of §3 or §4 (with $g = 1/f$). In Matlab, one could write

```
[det]          for k = 1:n, fk(k) = det(zk(k)*I-A), end
```

followed by, say, `[Pz]`. (The operation count could be improved from $O(nN^3)$ to $O(N^3 + nN^2)$ by a preliminary factorization.) To illustrate, consider the 4×4 matrix

$$A = \begin{pmatrix} 3.2 & 1.5 & 0.5 & -0.5 \\ -1.6 & 0.0 & -0.4 & 0.6 \\ -2.1 & -2.2 & 0.2 & -0.1 \\ 20.7 & 9.3 & 3.9 & -3.4 \end{pmatrix},$$

with eigenvalues $0.2, 0.3, 1.5, -1.9$; thus the eigenvalues in D are 0.2 and 0.3. Here are the computed eigenvalue estimates with $n = 30$:

```
Polynomial interpolation, Algorithm [Pz]/[Pz*]: 0.20000000, 0.30000000
     Cauchy integrals in f'/f, Algorithm [Cz1]: 0.19901287, 0.29895203
Cauchy integrals in 1/f, Algorithm [Cz2]/[ratdisk]: 0.19994887, 0.30005345
```

In this case polynomial interpolation gives the exact result, since the determinant is a polynomial of degree $N \le n$. For dimensions $N \gg 1$, of course, one would not be prepared to compute enough determinants for this property to hold. The other two algorithms give geometrically accurate results, with linearized rational interpolation performing particularly well.

Though this small-scale experiment was successful, we are not sure whether finding zeros of the determinant is likely to be an attractive option in many large-scale computations, though a related algorithm has been proposed in [43].

The other approach for computing eigenvalues we will now consider is certainly an important one: computing poles of the resolvent. Strengthening the foundations of algorithms of this kind was a significant motivation for us to write this paper.

The eigenvalues of A are the poles of the resolvent function, $(zI - A)^{-1}$. This function is matrix-valued, but suppose u and v are fixed N-vectors. Then the rational function

$$g(z) = u^*(zI - A)^{-1}v \tag{5.2}$$

is scalar-valued. If u and v are random vectors, then with probability 1, g will have poles at all of the eigenvalues of A (though the residues may be very small). The same applies in the special case where u and v are taken to be equal.

This immediately gives us a first idea for approximating eigenvalues by polefinding: evaluate the scalarized resolvent function g at roots of unity, then use `[Cz2]` or equivalently `[ratdisk]` to estimate its poles. Each point evaluation of g requires the solution of a linear system of equations involving a shift of A, which may be carried out in parallel:

```
[res]        for k = 1:n, gk(k) = u'*((zk(k)*I-A)\v), end
```

As an example, consider the same 4×4 matrix as before. Here are the computed eigenvalue estimates with $n = 30$ and $u = v = (1, 1, 1, 1)^T$:

```
Rational interpolation, Algorithm [Cz2]/[ratdisk]: 0.20021799, 0.29990926
```

In the past decade, algorithms of this kind have begun to be widely used. There are two main schools of research in this area: Sakurai and his collaborators in Japan [3, 35, 36, 66, 67], and Polizzi and his collaborators [25, 46, 50, 51, 63, 73], who have given their software package the name FEAST. The Sakurai school began with a trapezoidal discretization of Cauchy integrals, effectively `[Cp]` applied to the scalarized resolvent function g [66]. Polizzi chose instead a Gauss quadrature discretization over semicircles. Both groups have applied their algorithms to large-scale problems in physics, particularly the calculation of electronic structure, where earlier related algorithms were presented by Goedecker [26].

For these eigenvalue computations to be fully effective, one must move beyond scalar functions $g(z)$. To explain this we note that although algorithm `[ratdisk]` will find multiple poles, not every multiple eigenvalue of a matrix A produces a multiple pole of the function g; this will happen only if the eigenvalues correspond to just a single Jordan block (i.e., the algebraic multiplicity is > 1 but the geometric multiplicity is 1). The Hermitian matrices that often arise in physics are very far from this situation. To get around this problem, rather than using a single pair of vectors u and v, one can use a set of such vectors, so that g effectively becomes a matrix function of dimension greater than 1 but much less than N. Polizzi utilized this device from the start [63], and Sakurai (the "block Sakurai–Sugiura method") from 2010 [35, 36]. A further important practical modification is that the sampling at roots of unity may be imbedded in an outer loop, so that in the language of our Introduction, the unit disk filter function is applied not just once but several times to improve the projection onto the unit disk, damping eigenvector components outside the disk relative to those inside. Many other practical variations have been considered, including use of regions other than a disk, and this is a rapidly moving area of research. Generalizations to nonlinear eigenvalue problems have been published by Asakura, et al. and Beyn [3, 8].

6. Further notes on the literature. In 2008, the third author started to write a book called *Neoclassical Numerics,* which was to begin with a chapter on the unit disk. Later he decided that the work should concentrate instead on $[-1, 1]$, and it evolved into *Approximation Theory and Approximation Practice (ATAP)* [75]. The present paper now returns to that project of 2008, informed by the subsequent years' experience with *ATAP* and also the Chebfun software project: one may think of this paper as a kind of "*ATAP* for the unit disk," or as the mathematical basis for a "Diskfun" project (not planned). There are close links between monomials, roots of

unity, and the disk D_R as considered here and Chebyshev polynomials, Chebyshev points, and Bernstein ellipses as discussed in [75].

A feature of both that project and this one is that the mathematics involved is in good part classical, depending on results well distributed across the past century. In particular, when it comes to approximation of functions in roots of unity, a great expert was Joseph Walsh (1895–1973), who served on the mathematics faculty at Harvard from 1921 to 1966. Walsh had little interest in numerical algorithms, but much of what we have done here would be familiar to him, and we wish he were here to comment on every line of this paper. His book [79] is an important resource, though difficult to read.

§2.1. *Two algorithms for evaluating a function.* Algorithm P is not discussed much in the numerical analysis literature, but algorithm C has received plenty of attention, beginning with the work of Lyness and his coauthors in the 1960s [14], [55]–[58]. We do not know how often it has been noticed that algorithms P and C are related by (2.1); this observation appears as eq. (5.12) of [28].

§2.2. *Barycentric formulas for the two algorithms.* For general point sets $\{z_k\}$, the first barycentric formula (2.5) is due to Jacobi in 1825 [40], and the second to Dupuy in 1948 [16]. For the particular case of roots of unity, formula (P) was perhaps first written down by Henrici and can be found in print in [7, 28].

§2.3. *Taylor projections and polynomial interpolants.* A parallel treatment of Chebyshev projections and interpolants can be found in Chapter 4 of [75].

§2.4. *Accuracy of the two algorithms.* It was Runge in 1904 who first showed convergence in D of polynomial interpolants in roots of unity [64, §II.15, pp. 136–137], though Méray had the necessary tools in hand two decades earlier [60]. Geometric convergence in the context of the trapezoidal rule for quadrature was analyzed by Davis in 1959 [13].

§2.5. *Zeros, poles, and the Jentzsch and Walsh theorems.* Beautiful phase portraits illustrating Jentzsch's theorem and discretized contour integrals can be found in Figures 3.9–3.14 and 4.18 of [77].

§2.6. *Cauchy integrals of the second kind.* In the integral equations literature, series-related reformulations of the standard Nyström-type algorithms to achieve better accuracy near contours have been developed lately. An example in the context of the Fast Multipole Method is the QBX method of Klöckner, et al. [45].

§2.7. *Derivatives of f or Taylor series.* The first proposal of numerical calculation of derivatives via the trapezoidal rule applied to Cauchy integrals may have been in [55, 56]. The FFT was introduced into the algorithm in [58], and barycentric formulas for derivatives were introduced in [69].

§2.8. *Scaling of the radius.* The question of adjusting the radius τ goes back to [55], where the problem of numerical instability for small τ was noted.

§2.9. *The method of fundamental solutions.* The tradeoff between accuracy and conditioning in the smooth, high-accuracy regime is a central issue in the field of radial basis functions [18, 23].

§3.1. *Two algorithms for zerofinding.* Equation (3.3) goes back to [39]. The classic paper on numerical zerofinding algorithms based on trapezoidal discretization of Cauchy integrals is [14], whose §6 proposes the use of the polynomial interpolant to approximate $\{f_k'\}$ as in `[Cz1]` (i.e., Taylor polynomials with coefficients calculated by the trapezoidal rule). Delves and Lyness use Newton identities to find the roots; Kravanja, Sakurai, and Van Barel [47] use a generalized eigenvalue problem based on modified moments. The "discrete Cauchy integral of the second kind" idea is applied

to zerofinding in [37]. The special case of (3.3) with $m = 0$ is the subject of [71].

§4.1. *Two algorithms for polefinding.* There does not appear to be much literature on polefinding based on data at roots of unity, though such problems get some attention in [49, Chapter 3] and [27].

§5. *Computing the eigenvalues of a matrix in the unit disk.* For more on the interplay of contour integrals, spectral projectors, and electronic structure, see [6, 53].

7. Conclusions. The central theme of this article is that there are two kinds of algorithms for the problems at hand: those based on function approximations (polynomial or rational), and those based on discretized Cauchy integrals (which implicitly also construct rational approximations). The former have the advantage that they introduce no singularities on the boundary contour and may therefore be more accurate, while the latter have the advantage that they restrict the problem to the interior region and may therefore be faster. Sometimes the two concepts lead to the same result, as in `[P]` and (2.20), which are identical, or `[Cp]` and `[ratdisk]`, which are mathematically equivalent but have different stability properties. It is rare in the literature for the two types of algorithms to be distinguished or compared. We believe such comparisons may reveal new computational possibilities, not just for analytic functions on the unit disk, but in many other areas that may be regarded as variations on this theme. A list of such variations with references was given in the introduction.

The main algorithms we have discussed can be summarized as follows, listed with the names of their Matlab realizations:

`[P]`	Evaluation of $f(z)$ by polynomial interpolation (p. 5)
`[P*]`	Same, coefficient-based algorithm (p. 5)
`[C]`	Evaluation of $f(z)$ by discretized Cauchy integral (p. 5)
`[P']`	Evaluation of $f'(z)$ by polynomial interpolation (p. 10)
`[P'*]`	Same, coefficient-based algorithm (p. 10)
`[C']`	Evaluation of $f'(z)$ by discretized Cauchy integral (p. 10)
`[Pz]`	Zerofinding for f via polynomial interpolation (p. 11)
`[Pz*]`	Same, coefficient-based algorithm (p. 11)
`[Cz1]`	Zerofinding for f via discretized Cauchy integral for f'/f (p. 14)
`[Cz2]`	Zerofinding for f via discretized Cauchy integral for $1/f$ (p. 14)
`[ratdisk]`	Polefinding for g via rational interpolation (p. 15)
`[Cp]`	Polefinding for g via discretized Cauchy integral (p. 15)

We close by summarizing some of the main points of this paper.

- For evaluating $f(z)$, polynomial interpolants and discretized Cauchy integrals are equivalent for $z = 0$, otherwise distinct. The latter method approximates f by a rational function equal to the polynomial interpolant divided by $1-z^n$, making it less accurate for $|z| \approx 1$.
- On the other hand, the barycentric formula for polynomial interpolation is identical to a "discretized Cauchy integral of the second kind," or as Ioakimidis puts it, a discretization of "the Cauchy theorem rather than the Cauchy formula."
- For zerofinding, the Cauchy integral approach has an $O(n \log n)$ vs. $O(n^3)$ complexity advantage over polynomial interpolation, because the integral serves to project the problem onto the unit disk. It is not clear that zerofinding via polynomial interpolation has ever been proposed in the literature.

- The polefinding method we have called `[Cp]`, based on discretized Cauchy integrals, is mathematically equivalent to computing the poles of a linearized rational interpolant, the algorithm we have called `[ratdisk]`. However, the latter is numerically more stable.
- The originators of algorithm `[Cp]`, Kravanja and Van Barel, recognized its instability and proposed a more stable algorithm based on modified moments and formal orthogonal polynomials [48, 49]. A challenge for future research is to compare the stability of that algorithm and `[ratdisk]`.
- FEAST and related algorithms for finding eigenvalues of a matrix in a disk are derived from the idea of polefinding for the resolvent function. In the simplest case the resolvent is reduced to a scalar; more robust block algorithms reduce it to an intermediate dimension. In addition an outer iteration may be applied to improve the projection onto the unit disk.

Acknowledgments. We are grateful for corrections and suggestions from Jean-Paul Berrut, Martin Gutknecht, Georges Klein, Edward Saff, Alex Townsend, Elias Wegert, and André Weideman.

REFERENCES

[1] A. Amiraslani, *New Algorithms for Matrices, Polynomials, and Matrix Polynomials,* PhD diss., U. Western Ontario, 2006.
[2] E. G. Anastasselou, *A formal comparison of the Delves–Lyness and Burniston–Siewert methods for locating the zeros of analytic functions,* IMA J. Numer. Anal., 6 (1986), pp. 337–341.
[3] J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura, *A numerical method for nonlinear eigenvalue problems using contour integrals,* JSIAM Letters, 1 (2009), pp. 52–55.
[4] A. H. Barnett and T. Betcke, *Stability and convergence of the method of fundamental solutions for Helmholtz problems on analytic domains,* J. Comp. Phys., 227 (2008), pp. 7003–7026.
[5] B. Beckermann, G. H. Golub, and G. Labahn, *On the numerical condition of a generalized Hankel eigenvalue problem,* Numer. Math., 106 (2007), pp. 41–68.
[6] M. Benzi, P. Boito, and N. Razouk, *Decay properties of spectral projectors with applications to electronic structure,* SIAM Rev., 55 (2013), pp. 3–64.
[7] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation,* SIAM Rev., 46 (2004), pp. 501–517.
[8] W.-J. Beyn, *An integral method for solving nonlinear eigenvalue problems,* Lin. Alg. Applics., 436 (2012), 3839–3863.
[9] F. Bornemann, *Accuracy and stability of computing high-order derivatives of analytic functions by Cauchy integrals,* Found. Comput. Math., 11 (2011), pp. 1–63.
[10] E. E. Burniston and C. E. Siewert, *The use of Riemann problems in solving a class of transcendental equations,* Proc. Camb. Phil. Soc, 73 (1973), pp. 111–118.
[11] R. Corless, *Generalized companion matrices for the Lagrange basis,* Proceedings EACA, 2004, pp. 317–322.
[12] J. H. Curtiss, *Interpolation by harmonic polynomials,* J. SIAM, 10 (1962), pp. 709–736.
[13] P. J. Davis, On the numerical integration of periodic analytic functions, in R. E. Langer, ed., *On Numerical Integration: Proceedings of a Symposium, Madison, April* 21–23*,* 1958*,* Math. Res. Ctr., U. of Wisconsin, 1959, pp. 45–59.
[14] L. M. Delves and J. N. Lyness, *A numerical method for locating the zeros of an analytic function,* Math. Comput., 21 (1967), pp. 543–560.
[15] A. Draux, *Formal orthogonal polynomials revisited. Applications,* Numer. Algorithms, 11 (1996), pp. 143–158.
[16] M. Dupuy, *Le calcul numérique des fonctions par l'interpolation barycentrique,* Comptes Rend. Acad. Sci., 226 (1948), 158–159.
[17] Ö. Eğecioğlu and Ç. Koç, *A fast algorithm for rational interpolation via orthogonal polynomials*, Math. Comp., 53 (1989), pp. 249–264.
[18] G. E. Fasshauer, *Meshfree Approximation Methods with Matlab,* World Scientific, Singapore, 2007.
[19] L. Fejér, *Interpolation und konforme Abbildung,* Göttinger Nachrichten (1918), pp. 319–331.

[20] M. S. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation,* Numer. Math., 107 (2007), pp. 315–331.
[21] B. Fornberg, *Numerical differentiation of analytic functions,* ACM Trans. Math. Softw., 7 (1981), pp. 512–526.
[22] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions,* SIAM J. Sci. Comput., 33 (2011), pp. 869–892.
[23] B. Fornberg and G. Wright, *Stable computation of multiquadric interpolants for all values of the shape parameter,* Comput. Math. Appl., 48 (2004), pp. 853–867.
[24] S. Fortune, *Polynomial root finding using iterated eigenvalue computation,* Proc. 2001 Intl. Symp. Symb. Alg. Comput., ACM, 2001, pp. 121–128.
[25] B. Gavin and E. Polizzi, *Non-linear-eigensolver-based alternative to traditional SCF methods,* J. Chem. Phys., 138 (2013), pp. 194101:1–8.
[26] S. Goedecker, *Linear scaling electronic structure methods,* Rev. Modern Phys., 71 (1999), pp. 1085–1123.
[27] P. Gonnet, R. Pachón, and L. N. Trefethen, *Robust rational interpolation and least-squares,* Elect. Trans. Numer. Anal., 38 (2011), pp. 146–167.
[28] M. H. Gutknecht, *Numerical conformal mapping methods based on function conjugation,* J. Comput. Appl. Math, 14 (1986), pp. 31–77.
[29] N. Hale, N. J. Higham, and L. N. Trefethen, *Computing A^α, $\log(A)$, and related matrix functions by contour integrals,* SIAM J. Numer. Anal., 46 (2008), pp. 2505–2523.
[30] J. Helsing and R. Ojala, *On the evaluation of layer potentials close to their sources,* J. Comp. Phys., 227 (2008), pp. 2899–2921.
[31] P. Henrici, *Fast Fourier methods in complex analysis,* SIAM Rev., 21 (1979), pp. 481–527.
[32] P. Henrici, *Essentials of Numerical Analysis with Pocket Calculator Demonstrations,* Wiley, 1982.
[33] N. J. Higham, *The numerical stability of barycentric Lagrange interpolation,* IMA J. Numer. Anal., 24 (2004), pp. 547–556.
[34] E. Hlawka, *Interpolation analytischer Funktionen auf dem Einheitskreis,* in P. Turán, ed., Number Theory and Analysis, Plenum, New York, 1969, pp. 99–118.
[35] T. Ikegami and T. Sakurai, *Contour integral eigensolver for non-Hermitian systems: A Rayleigh–Ritz-type approach,* Taiwan J. Math., 14 (2010), pp. 825–837.
[36] T. Ikegami, T. Sakurai, and U. Nagashima, *A filter diagonalization for generalized eigenvalue problems based on the Sakurai–Sugiura projection method,* J. Comput. Appl. Math., 223 (2010), pp. 1927–1936.
[37] N. I. Ioakimidis, *A modification of the classical quadrature method for locating zeros of analytic functions,* BIT, 25 (1985), pp. 681–686.
[38] N. I. Ioakimidis, K. E. Papadakis, and E. A. Perdios, *Numerical evaluation of analytic functions by Cauchy's theorem,* BIT, 31 (1991), pp. 276–285.
[39] D. Jackson, *Roots and singular points of semi-analytic functions,* Annals Math., 19 (1917), pp. 142–151.
[40] C. G. J. Jacobi, *Disquisitiones Analyticae de Fractionibus Simplicibus,* dissertation, Berlin, 1825.
[41] C. G. J. Jacobi, *Über die Darstellung einer Reihe gegebner Werthe durch eine gebrochne rationale Function,* J. Reine Angew. Math., 30 (1846), pp. 127–156.
[42] R. Jentzsch, *Untersuchungen zur Theorie Analytischer Funktionen,* dissertation, Berlin, 1914.
[43] E. Kamgnia and B. Philippe, *Counting eignevalues in domains of the complex field,* Elect. Trans. Numer. Anal., 40 (2013), pp. 1–16.
[44] A.-K. Kassam and L. N. Trefethen, *Fourth-order time-stepping for stiff PDEs,* SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.
[45] A. Klöckner, A. Barnett, L. Greengard, and M. O'Neil, *Quadrature by expansion: a new method for the evaluation of layer potentials,* J. Comput. Phys., to appear.
[46] L. Krämer, E. Di Napoli, M. Galgon, B. Lang, and P. Bientinesi, *Dissecting the FEAST algorithm for generalized eigenproblems,* J. Comput. Appl. Math., 244 (2013), pp. 1–9.
[47] P. Kravanja, T. Sakurai, and M. Van Barel, *On locating clusters of zeros of analytic functions,* BIT, 39 (1999), pp. 646–682.
[48] P. Kravanja and M. Van Barel, *A derivative-free algorithm for computing zeros of analytic functions,* Computing, 63 (1999), pp. 69–91.
[49] P. Kravanja and M. Van Barel, *Computing the Zeros of Analytic Functions,* Springer Lect. Notes Math. v. 1727, 2000.
[50] S. E. Laux, *Solving complex band structure problems with the FEAST eigenvalue algorithm,* Phys. Rev. B, 86 (2012), pp. 075103:1–10.
[51] A. R. Levin, D. Zhang, and E. Polizzi, *FEAST fundamental framework for electronic struc-*

ture calculations: Reformulation and solution of the muffin-tin problem, Comput. Phys. Commun., 183 (2012), pp. 2370–2375.
[52] T.-Y. Li, *On locating all zeros of an analytic function within a bounded domain by a revised Delves/Lyness method,* SIAM J. Numer. Anal., 20 (1983), pp. 865–871.
[53] L. Lin, J. Lu, L. Ying, and W. E, *Pole-based approximation of the Fermi–Dirac function,* Chin. Ann. Math., 30B (2009), pp. 729–742.
[54] R. Luck and J. W. Stevens, *Explicit solutions for transcendental equations,* SIAM Rev., 44 (2002), pp. 227–233.
[55] J. N. Lyness, *Numerical algorithms based on the theory of complex variable,* Proceedings of the 1967 22nd National Conference (1967), pp. 125–133.
[56] J. N. Lyness and L. M. Delves, *On numerical contour integration round a closed contour,* Math. Comput., 21 (1967), 561–577.
[57] J. N. Lyness and C. B. Moler, *Numerical differentiation of analytic functions,* SIAM J. Numer. Anal., 4 (1967), pp. 202–210.
[58] J. N. Lyness and G. Sande, *Algorithm* 413*: ENTCAF and ENTCRE: evaluation of normalized Taylor coefficients of an analytic function,* Commun. ACM, 14 (1971), pp. 669–675.
[59] J. E. McCune, *Exact inversion of dispersion relations,* Phys. Fluids, 9 (1966), pp. 2082–2084.
[60] C. Méray, *Observations sur la légitimité de l'interpolation,* Annal. Scient. de l'École Normale Supérieure, 3 (1884), pp. 165–176.
[61] R. Pachón, P. Gonnet, and J. Van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points,* SIAM J. Numer. Anal., 50 (2012), pp. 1713–1734.
[62] C. Pearcy, *An elementary proof of the power inequality for the numerical radius,* Mich. Math. J., 13 (1966), pp. 289–291.
[63] E. Polizzi, *Density-matrix-based algorithm for solving eigenvalue problems,* Phys. Rev. B, 79 (2009), pp. 115112:1–6.
[64] C. Runge, *Theorie und Praxis der Reihen,* G. J. Göschen, Leipzig, 1904.
[65] T. Sakurai, P. Kravanja, H. Sugiura, and M. Van Barel, *An error analysis of two related quadrature methods for computing zeros of analytic functions,* J. Comp. Appl. Math., 152 (2003), pp. 467–480.
[66] T. Sakurai and H. Sugiura, *A projection method for generalized eigenvalue problems using numerical integration,* J. Comput. Appl. Math, 159 (2003), pp. 119–128.
[67] T. Sakurai and H. Tadano, *CIRR: A Rayleigh–Ritz type method with contour integral for generalized eigenvalue problems,* Hokkaido Math. J., 36 (2007), pp. 745–757.
[68] T. Schmelzer and L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals,* Electron. Trans. Numer. Anal., 29 (2007), pp. 1–18.
[69] C. Schneider and W. Werner, *Some new aspects of rational interpolation,* Math. Comput., 47 (1986), pp. 285–299.
[70] I. H. Sloan and R. S. Womersley, *Extremal systems of points and numerical integration on the sphere,* Advances Comp. Math., 21 (2004), pp. 107–125.
[71] R. Spira, *Zeros of approximate functional approximations,* Math. Comput., 21 (1967), pp. 41–48.
[72] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions,* SIAM Rev., 40 (1998), pp. 110–112.
[73] P. T. P. Tang and E. Polizzi, *Subspace iteration with approximate spectral projection,* arXiv:1302.0432, 2013.
[74] A. Townsend, personal communication, 2012.
[75] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013.
[76] L. N. Trefethen and J. A. C. Weideman, *The exponentially convergent trapezoidal rule,* in preparation.
[77] E. Wegert, *Visual Complex Functions,* Springer–Birkhäuser, 2012.
[78] J. L. Walsh, *The analogue for maximally convergent polynomials of Jentzsch's theorem,* Duke Math. J., 26 (1959), pp. 605–616.
[79] J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Domain,* 5th ed., Amer. Math. Soc., 1969 (1st edition published in 1935).