Electronic Transactions on Numerical Analysis.
Volume 38, pp. 146-167, 2011.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

ROBUST RATIONAL INTERPOLATION AND LEAST-SQUARES*

PEDRO GONNET†, RICARDO PACHÓN†, AND LLOYD N. TREFETHEN†

Abstract. An efficient and robust algorithm and a Matlab code `ratdisk` are presented for rational interpolation or linearized least-squares approximation of a function based on its values at points equally spaced on a circle. The use of the singular value decomposition enables the detection and elimination of spurious poles or Froissart doublets that commonly complicate such fits without contributing to the quality of the approximation. As an application, the algorithm leads to a method for the stable computation of certain radial basis function interpolants in the difficult case of smoothness parameter ε close to zero.

Key words. Rational interpolation, spurious poles, Froissart doublets, Padé approximation, radial basis functions, ratdisk, singular value decomposition

AMS subject classifications. 41A20, 41A21, 65D05

1. Introduction.

Polynomial interpolants and least-squares fits are used all the time, rational ones more rarely. The potential advantage of rational approximations is that they may behave better in the presence of singularities, and, in particular, may be used to extrapolate or interpolate a function beyond poles that would block the convergence of a polynomial. The disadvantage is that they are more fragile. Certain approximants do not exist, or are nonunique, or depend discontinuously on the data, issues that are not just of theoretical importance as they tend to arise whenever one approximates a function that is even or odd. More challenging in practice is the fact that even when a unique and well-posed approximant exists in theory, especially for higher degree numerators and denominators, it may be difficult to compute numerically in finite precision arithmetic. A symptom of this situation is the common appearance of poles with residues close to machine precision—"spurious poles" or "Froissart doublets"—which contribute negligibly to the quality of the approximation while still causing difficulty in its application. For these reasons, despite many interesting contributions going back to Cauchy and Jacobi in the first half of the 19th century, rational interpolation and least-squares fitting have not become a robust tool of numerical computation that is widely relied upon.

In this article we propose an algorithm that we hope is a step in the right direction, together with an implementation in the form of a 58-line Matlab code `ratdisk`. This algorithm is an outgrowth of the "PGV method" described recently in [19], but goes further in removing spurious poles and in producing linearized least-squares approximants as well as interpolants. A wide range of computed examples are presented to illustrate some of the properties of the algorithm.

The application to radial basis functions (Section 8) is what originally led to the writing of this paper, especially through discussions of the third author with Bengt Fornberg of the University of Colorado and Grady Wright of Boise State University.

For an excellent presentation of rational interpolation and approximation we recommend Chapter 5 of [3].

2. Notation.

Throughout this paper we use the following notation. The symbol P_m denotes the set of polynomials of degree $\le m$, and R_{mn} is the set of rational functions of type (m, n), that is, functions that can be written as the quotient of a polynomial in P_m and a

*Received February 10, 2011. Accepted for publication February 28, 2011. Published online May 18, 2011. Recommended by L. Reichel. P. G. was supported by Swiss National Science Foundation Individual Support Fellowships Nr. PBEZP2-127959 and Nr. PA00P2-134146.

†Oxford University Mathematical Institute, 25-29 St Giles, Oxford OX1 3LB, UK (`Pedro.Gonnet,Nick.Trefethen@maths.ox.ac.uk`).

polynomial in P_n. Our purpose is to use functions in R_{mn} to approximate a function f defined on the unit circle $\{z \in \mathbb{C} : |z| = 1\}$. The approximations will be based on the values taken by f at the $(N+1)$st roots of unity on the unit circle, where $N \geq m + n$. We define the roots of unity by $z_j = \exp(2\pi ij/(N+1))$, $0 \leq j \leq N$, where $i = \sqrt{-1}$, and the corresponding values of f by $f_j = f(z_j)$. (Our algorithms can also be applied to arbitrary data $\{f_j\}$, which may not come from an underlying function f, but the applications we are concerned with involve such functions.) Finally, we define $\|\mathbf{v}\|$ to be the usual 2-norm of a vector $\mathbf{v}$, and $\|p\|_N$ to be the root-mean-square norm of a function p defined at the $(N+1)$st roots of unity. That is, if $\mathbf{p}$ is the $(N+1)$-vector with entries $p_j = p(z_j)$, then

$$\|p\|_N = (N+1)^{-1/2}\|\mathbf{p}\|.$$

Note that for the particular case of the function z^k for some k we have $\|z^k\|_N = 1$, and since different powers of z are orthogonal over the roots of unity, this implies

$$\|p\|_N = \|\mathbf{a}\|$$

whenever $p \in P_N$ and $\mathbf{a}$ is its vector of coefficients, i.e., $p(z) = \sum_{k=0}^{N} a_k z^k$.

We summarize these notations as follows:

P_m:	set of polynomials of degree at most m
R_{mn}:	set of rational functions of type (m, n)
f:	function defined on the unit circle
N:	number of sample points, minus 1
$\{z_j\}$:	$(N+1)$st roots of unity
$\{f_j\}$:	values of f at these points
$\|\mathbf{v}\|$:	2-norm of a vector $\mathbf{v}$
$\|p\|_N$:	root-mean-square norm of a function p over the roots of unity.

Note that we have not yet prescribed which approximation $r \in R_{mn}$ to f we are looking for. That is because several ideas are in play here, and robustness of the algorithm requires clarity about the different possibilities. A first possibility, in the case $N = m + n$, is to look for an interpolant satisfying

$$r(z_j) = f_j, \quad 0 \leq j \leq N, \tag{2.1}$$

but such a function does not always exist. For example, there is no $r \in R_{11}$ that takes the same value at two of the 3rd roots of unity and a different value at the other. Instead one may linearize the problem and look for polynomials $p \in P_m$ and $q \in P_n$ such that

$$p(z_j) = f_j q(z_j), \quad 0 \leq j \leq N. \tag{2.2}$$

Obviously, at least one solution to this problem exists, with $p = q = 0$; to make the problem meaningful, some kind of normalization is needed. The normalization we shall employ is the condition

$$\|q\|_N = 1. \tag{2.3}$$

Existence of polynomials satisfying (2.2) and (2.3) is guaranteed, as follows from the linear algebra discussed in the next section. Generically, such polynomials will correspond to a rational function $r = p/q$ satisfying (2.1), but not always. A potentially more robust approach is to take $N > m+n$ and find polynomials $p \in P_m$ and $q \in P_n$ that solve the least-squares problem

$$\|p - fq\|_N = \text{minimum} \tag{2.4}$$

again normalized by (2.3). As before, existence of polynomials p and q is guaranteed (though not, as we shall see, uniqueness). This problem, which we call the *linearized least-squares problem* for rational interpolation, is the starting point of the algorithm we recommend in this article, and we describe the mathematical basis of how we solve it in the next section. In Section 4 we show that this basic approach to rational interpolation and least-squares, however, is susceptible to difficulties in machine computation. Section 5 is then the heart of the article, where we present a more practical algorithm whose key feature is the removal of contributions from minimal singular values that lie below a relative tolerance ($\texttt{tol} = 10^{-14}$ works well in many applications), or that match the next higher singular value to within such a tolerance.

In the discussion of Section 9 we comment on an iterative algorithm for solving the true nonlinear least-squares problem

$$\|r - f\|_N = \text{minimum.} \tag{2.5}$$

3. Linearized interpolation and least-squares. Following [19], we now describe a method for computing a solution to the least-squares problem (2.3)–(2.4); Figure 3.1 provides a helpful reference. Let $\mathbf{b}$ be an $(n+1)$-vector with $\|\mathbf{b}\| = 1$

$$\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix},$$

defining coefficients of a polynomial $q \in P_n$ satisfying (2.3), i.e., $q(z) = \sum_{k=0}^{n} b_k z^k$. Let $\mathbf{z}$ be the $(N+1)$-vector of the $(N+1)$st roots of unity

$$\mathbf{z} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_N \end{bmatrix},$$

and let powers such as $\mathbf{z}^2, \mathbf{z}^3$ be defined componentwise. Then the vector

$$\mathbf{p} = \begin{bmatrix} f_0 & & & \\ & f_1 & & \\ & & \ddots & \\ & & & f_N \end{bmatrix} \left[\begin{array}{c|c|c} \mathbf{z}^0 & \cdots & \mathbf{z}^n \end{array} \right] \mathbf{b}$$

is the $(N+1)$-vector with entries

$$p_j = f_j q(z_j), \quad 0 \le j \le N.$$

Multiplying on the left by $(N+1)^{-1}$ times the $(N+1) \times (N+1)$ matrix of conjugate transposes of vectors $\mathbf{z}^j$ gives the $(N+1)$-vector

$$\hat{\mathbf{a}} = \hat{Z}\mathbf{b}, \tag{3.1}$$

ETNA
Kent State University
http://etna.math.kent.edu

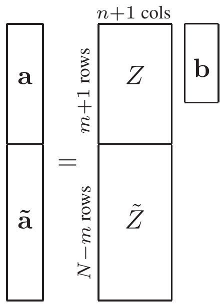

FIG. 3.1. *Structure of the Toeplitz matrix equation* $\hat{\mathbf{a}} = \hat{Z}\mathbf{b}$ *of (3.1), summarizing the notation of Section 3. In terms of function values rather than coefficients, this corresponds to equation (3.4),* $\hat{p}(z_j) = f_j q(z_j)$ *for all* $0 \le j \le N$*. The least-squares problem is to minimize* $\|\tilde{\mathbf{a}}\|$ *subject to the constraint* $\|\mathbf{b}\| = 1$*. The minimum value of* $\|\tilde{\mathbf{a}}\|$ *is* $\sigma_{\min}$*, the smallest singular value of* $\tilde{Z}$*.*

as shown in Figure 3.1, where $\hat{Z}$ is the $(N+1) \times (n+1)$ matrix

$$\hat{Z} = \frac{1}{N+1} \begin{bmatrix} (\mathbf{z}^0)^* \\ \hline \vdots \\ \hline (\mathbf{z}^N)^* \end{bmatrix} \begin{bmatrix} f_0 & & & \\ & f_1 & & \\ & & \ddots & \\ & & & f_N \end{bmatrix} \left[\begin{array}{c|c|c} \mathbf{z}^0 & \cdots & \mathbf{z}^n \end{array} \right]. \tag{3.2}$$

We can interpret this product of three matrices as follows. To find the polynomial coefficients of the product fq, we could convolve the coefficients of q with those of f. Equation (3.2) takes a discrete Fourier transform to convert this convolution to a multiplication by values of f, then returns to coefficient space by the inverse discrete Fourier transform.

Explicitly, the entries of $\hat{Z}$ are given by

$$z_{jk} = \frac{1}{N+1} \sum_{\ell=0}^{N} z_\ell^{k-j} f_\ell. \tag{3.3}$$

Thus, we see that $\hat{Z}$ is a nonsymmetric Toeplitz matrix whose first column is the discrete Fourier transform of the data $\{f_j\}$. The vector $\hat{\mathbf{a}}$ is the vector of coefficients of the unique polynomial $\hat{p} \in P_N$ taking the values

$$\hat{p}(z_j) = f_j q(z_j), \quad 0 \le j \le N. \tag{3.4}$$

Let us now define p to be the best approximation to $\hat{p}$ in P_m with respect to the norm $\|\cdot\|_N$, that is, the truncation of $\hat{p}$ to degree m, and let $\mathbf{a}$ be its vector of coefficients, that is, the truncation of $\hat{\mathbf{a}}$ to length $m+1$. Then $\hat{p} - p$ is the polynomial in P_N with coefficients $a_{m+1}, \dots, a_N$, implying

$$\|p - \hat{p}\|_N = \|p - fq\|_N = \left(\sum_{j=m+1}^{N} |a_j|^2 \right)^{1/2}.$$

In other words $\|p - fq\|_N = \|\tilde{\mathbf{a}}\|$, where

$$\tilde{\mathbf{a}} = \tilde{Z}\mathbf{b}, \tag{3.5}$$

ETNA
Kent State University
http://etna.math.kent.edu

and $\tilde{Z}$ is the $(N-m)\times(n+1)$ matrix consisting of the last $N-m$ rows of $\hat{Z}$, as shown in Figure 3.1. The norm $\|\tilde{\mathbf{a}}\|$ will be as small as possible if and only if $\mathbf{b}$ is a minimal singular vector of $\hat{Z}$. The corresponding vector $\mathbf{a}$ is then

$$\mathbf{a} = Z\mathbf{b},$$

where Z is the $(m+1)\times(n+1)$ matrix consisting of the first $m+1$ rows of $\hat{Z}$. Notice that since the minimal singular value of a matrix may be multiple, there is no reason to expect that $\mathbf{b}$ will always be unique. We shall return to this matter in Section 5.

Case $N=m+n$: interpolation. The matrix $\tilde{Z}$ is of dimension $(N-m)\times(n+1)$. An important special case is that in which $N=m+n$, corresponding to interpolation rather than least squares. In this case $\tilde{Z}$ has dimension $n\times(n+1)$, so it must have a nonzero null vector $\mathbf{b}$ for which the approximation error is $\|p-fq\|_N = 0$. In other words, the linearized rational interpolation problem (3.4) is guaranteed to have a nontrivial solution in this case, and we can compute a null vector $\mathbf{b}$ numerically with the singular value decomposition (SVD). The amount of work is $O(n^3)$.

Case $N>m+n$: least-squares. For larger N, $\tilde{Z}$ will be square or more usually rectangular with more rows than columns. We now have a true least-squares problem, which again can be solved with the SVD. The amount of work is $O(n^2N)$.

Idealized Matlab code segment. In Matlab, suppose m,n,N are given together with a column $(N+1)$-vector `fj` of data values $\{f_j\}$. The following code segment produces the coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ of the polynomials p and q. (In Section 5 we shall improve this in many ways.) The roots of q can be found afterward by `roots(b(end:-1:1))`, and similarly for p.

```
col = fft(fj)/(N+1);                % column of Toeplitz matrix
row = conj(fft(conj(fj)))/(N+1);    % row of Toeplitz matrix
Z = toeplitz(col,row(1:n+1));       % the Toeplitz matrix
[U,S,V] = svd(Z(m+2:N+1,:),0);      % singular value decomposition
b = V(:,n+1);                       % coefficients of q
qj = ifft(b,N+1);                   % values of q at zj
ah = fft(qj.*fj);                   % coefficients of p-hat
a = ah(1:m+1);                      % coefficients of p
pj = ifft(a,N+1);                   % values of p at zj
```

Evaluation of the rational function. Once the coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ have been determined, there are two good methods for evaluating $r(z)=p(z)/q(z)$: direct use of the coefficients, or barycentric interpolation. Suppose a vector `zz` of numbers z is given and we wish to find the vector `rr` of corresponding values $r(z)$. The following command computes them in the direct fashion:

```
rr = polyval(a(end:-1:1),zz)./polyval(b(end:-1:1),zz)
```

Alternatively, they can be computed without transforming to coefficient space by the following process of rational barycentric interpolation described in [19].

```
rr = zeros(size(zz));
for i = 1:length(zz)
   dzinv = 1./(zz(i)-zj(:));
   ij = find(~isfinite(dzinv));
   if length(ij)>0, rr(i) = pj(ij)/qj(ij);
   else rr(i) = ((pj.*zj).'*dzinv)/((qj.*zj).'*dzinv); end
end
```

ETNA
Kent State University
http://etna.math.kent.edu

For our purposes both of these evaluation methods are fast and stable, and in the remainder of this paper, the experiments and discussion are based on the simpler direct method. (The advantages of barycentric interpolation become important for approximations in sets of points other than roots of unity.)

4. Spurious poles or Froissart doublets. To illustrate various approximations, this paper presents a number of figures in a uniform format. Each plot corresponds to the approximation of a function f in the $(N+1)$st roots of unity by a rational function of type (m, n) computed in standard IEEE double precision arithmetic. The unit circle is marked, with the roots of unity shown as black dots. The triplet (m, n, N) is listed on the upper-right, and a label on the upper-left reads "Interpolation" if $N = m+n$ and "Least-squares" if $N > m+n$. Our standard choice in the latter case is $N = 4(m + n) + 1$. The advantage of having N odd is discussed in the next section.

The lower-left of each plot lists the exact type (μ, ν), to be explained in the next section, and the elapsed time for computing this approximation on a 2010 desktop computer.

Each plot also lists a number `Err`, equal to the maximum of $|f(z) - r(z)|$ over the discrete grid of 7860 points in the unit disk whose real and imaginary coordinates are odd multiples of 0.01. How to choose a single scalar like this to measure the accuracy of r as an approximation to f is not in the least bit clear. Different rational functions will be constructed for different purposes and can be expected to have very different approximation properties. If f is meromorphic in the disk, for example, then one may hope that r will approximate it closely throughout the disk, at least away from a small region around each pole. (A function is meromorphic if it is analytic apart from poles.) For type (n, n) approximation there is no difference in principle between the interior and the exterior of the disk however, so one could also measure error outside the disk, or in an annulus centered on the unit circle. Another issue is that if there are branch points as opposed to poles, or essential singularities, one cannot expect close approximation near them. We have settled on the quantity `Err` defined above as a simple indicator that makes sense for many problems, and have added comments in the captions of figures where this measure does not work so well (Figures 6.6 and 6.8). Interestingly, `Err` is meaningful even in many cases where f has poles in the disk, though it would diverge to ∞ in such cases if the grid were infinitely fine.

Finally, each plot also shows poles or essential singularities of f, marked by crosses, and poles of r, marked by dots. The absolute value of the residue at each pole of r, evaluated by a finite difference, is indicated by a color code (a scheme suggested to us by Grady Wright):

$$|\text{residue}| \in \begin{cases} [10^{-3}, \infty) & \text{blue} \\ [10^{-6}, 10^{-3}) & \text{light blue} \\ [10^{-9}, 10^{-6}) & \text{green} \\ [10^{-12}, 10^{-9}) & \text{light green} \\ [10^{-14}, 10^{-12}) & \text{pink} \\ (0, 10^{-14}) & \text{red.} \end{cases}$$

Thus, a blue or green pole has a good chance of being genuine and useful for approximation, whereas pink and red poles are likely to be artifacts introduced by rounding errors.

In this section we consider just one example function,

$$f(z) = \tan(4z),$$

with poles at odd multiples of $\pi/8$; many more examples are presented in Section 6. Figure 4.1 shows rational interpolants and least-squares fits to f of type $(8, 8)$, $(80, 8)$, and

ETNA
Kent State University
http://etna.math.kent.edu

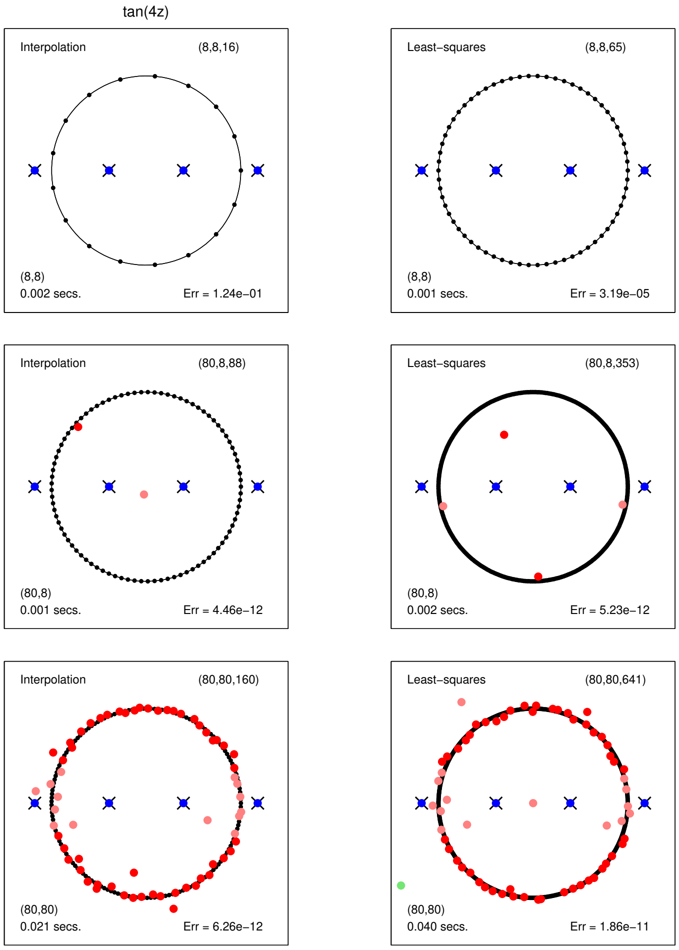

FIG. 4.1. *Approximations to* $\tan(4z)$ *by the algorithm of Section 3. For the type* $(8, 8)$ *approximations of the top row, both approximations successfully place four poles where one would expect them, and least-squares improves the quality of the fit by orders of magnitude. For type* $(80, 8)$*, in the second row, some spurious poles have appeared, and least-squares no longer makes much difference. With type* $(80, 80)$*, there are dozens of spurious poles clustering along the unit circle. In both the second and third rows, the spurious poles would make the* `Err` *measure infinite if the grid on which* $|f(z) - r(z)|$ *was measured were infinitely fine, but the grid has just* 7860 *points and poles at arbitrary points with residues below* 10^{-12} *usually slip through undetected.*

$(80, 80)$. The type $(8, 8)$ fits are trouble-free, with four poles of r closely matching poles of f. In the type $(80, 8)$ fits, however, a few pink and red dots have appeared at seemingly arbitrary locations. With type $(80, 80)$, the pink and red dots have become numerous, and most are located near the unit circle. These poles with very small residues, introduced by rounding errors, are what we call *spurious poles* or *Froissart doublets* [11, 12]. The word doublet alludes to the fact that near each pole one will normally find an associated zero, the pole and zero effectively cancelling each other except locally.

One can explain the appearance of spurious poles as follows. For low degrees m and n, all the available parameters may be needed to achieve a good approximation, and thus poles tend to be placed in a manner well adapted to the function being approximated. As m and n increase, on the other hand, or even if m increases with n held fixed, we begin to have more

ETNA
Kent State University
http://etna.math.kent.edu

parameters available than are needed to approximate f to machine precision. In this regime we are fitting the rounding errors rather than the data, and this is when spurious poles appear. Note that the pink and red dots in Figure 4.1 show neither of the symmetries one would expect for this function, namely up-down (since $f(\overline{z}) = \overline{f(z)}$) or left-right (since $f(-z) = -f(z)$). These losses of symmetry are further evidence of the dependence of these approximations on rounding error (symmetries are discussed further in the next section).

In the literature of rational approximation, spurious poles have been investigated mainly in two contexts. One are situations like our own, where finite precision effects or other perturbations introduce poles that in an exact analysis should not be there. Authors on this topic include Bessis, Fournier, Froissart, Gammel, Gilewicz, Kryakin, Pindor, and Truong-Van; see, for example, [12]. The other is in more theoretical studies on convergence of Padé and Padé-like approximants to functions f in the complex plane, especially the case of type (n, n) approximants with $n \to \infty$. In such cases it has been known at least since Perron in the 1920s that poles with small residues may appear at seemingly arbitrary places, preventing the Padé approximants $\{r_{nn}\}$ from approaching f pointwise as $n \to \infty$. Instead, the standard theorem of convergence of diagonal Padé approximants, the *Nuttall–Pommerenke Theorem,* asserts that $\{r_{nn}\}$ converges to a meromorphic function *in capacity*, which means away from exceptional sets that may vary from one value of n to the next and whose capacities decrease exponentially to 0 as $n \to \infty$ [1, 18, 20, 25]. (The capacity of a set is a standard notion of potential theory, and is greater than or equal to π^{-1} times the area measure, so convergence in capacity also implies convergence in measure.) If f is not meromorphic but has branch points, a theorem of Stahl makes an analogous statement about convergence in capacity away from certain arcs in the complex plane [25].

Simpler than the Nuttall–Pommerenke theorem is an earlier theorem of de Montessus de Ballore, concerning rows of the Padé table rather than diagonals, which asserts that as $m \to \infty$ with fixed n, the poles of approximants r_{mn} must approach those of a meromorphic function like $\tan(4z)$ [1, 17]. (The original theorem applies to Padé approximation, but rational interpolation in roots of unity is closely related, and indeed rational inpterpolation also goes by the name of multipoint Padé approximation [22].) This theorem asserts true pointwise convergence, not just convergence in capacity, but in the second row of Figure 4.1 we can see that this convergence is evidently not taking place as we go from type $(8, 8)$ to type $(80, 8)$; it is undone by the rounding errors.

Figure 4.2 plots the singular values of $\tilde{Z}$ for the same six cases as in Figure 4.1. Below about 10^{-14}, these are clearly artifacts of rounding error, which introduces effectively random contributions of order machine epsilon in the coefficients of the numerator and denominator polynomials. This observation explains why the spurious poles in Figure 4.1 tend to cluster near the unit circle: it is because the roots of random polynomials tend to cluster near the unit circle [13, 16, 23, 24]. Figure 4.3 illustrates this effect by comparing the roots of a random polynomial of degree 100 with the poles of a type $(100, 100)$ rational interpolant to random data in 201 roots of unity.

5. A more robust algorithm and code. We now propose a collection of modifications to the algorithm and code segment of Section 3 to make rational interpolation and least-squares fitting more robust and useful for applications. The robust code is listed with line numbers in Figure 5.1, and our algorithmic proposals can be summarized as follows:

1. Check if $\{f_j\}$ is real symmetric,
2. If N is odd, check if $\{f_j\}$ is even or odd,
3. Remove contributions from negligible singular values of $\tilde{Z}$,
4. Remove the degeneracy if the smallest singular value of $\tilde{Z}$ is multiple.
5. Discard negligible trailing coefficients of p and q.

ETNA
Kent State University
http://etna.math.kent.edu

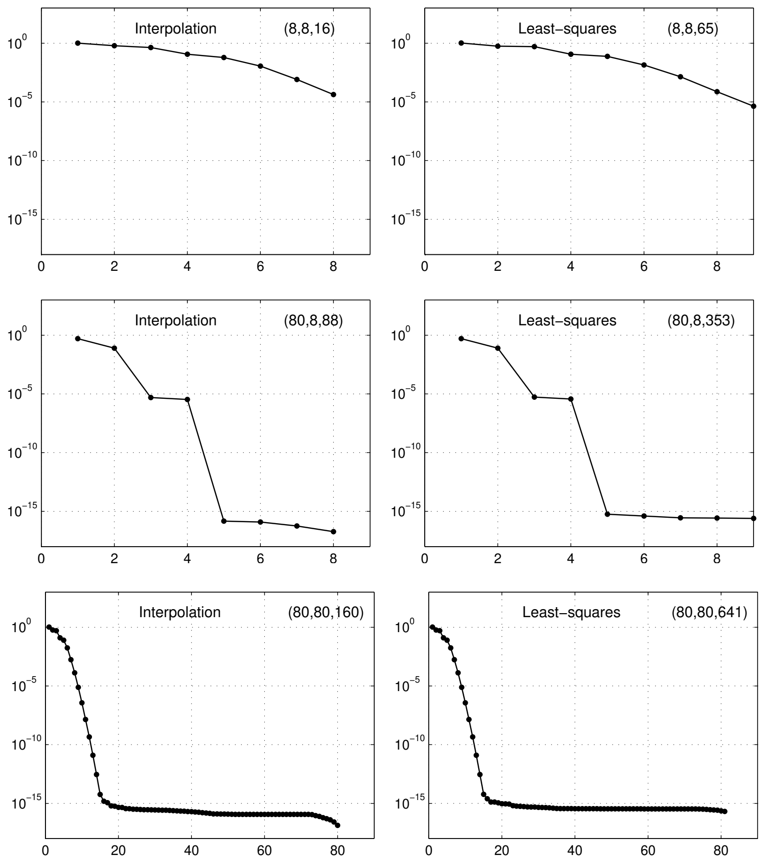

FIG. 4.2. *Nonzero singular values of $\tilde{Z}$ for the same six problems as in Figure 4.1. In the top row, rounding errors have little effect and the singular values are all genuine. The second row shows four genuine singular values but the rest of order 10^{-15} rather than decreasing toward zero as would happen in exact arithmetic. The bottom row shows about 15 singular values that could contribute to the quality of the approximation, plus dozens more at the level of machine precision.*

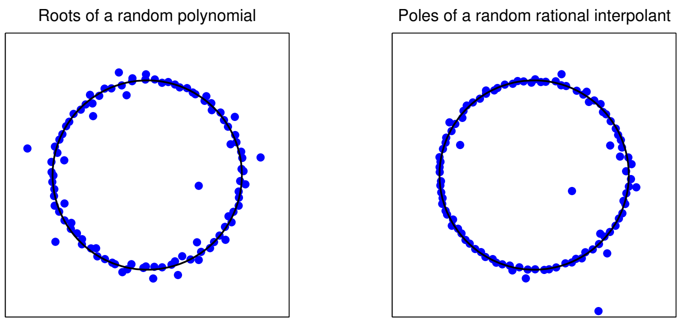

FIG. 4.3. *On the left, the roots of a random polynomial p of degree 100 with coefficients from a complex normal distribution of mean 0. The black line marks the unit circle. On the right, the poles of a rational interpolant of type $(100, 100)$ to random data from the same distribution at 201 roots of unity. The close match of the images illustrates that the appearance of spurious poles near the unit circle in numerical rational approximation is related to randomness in the computed denominators.*

ETNA
Kent State University
http://etna.math.kent.edu

All of these ideas involve a tolerance `tol`, which by default we set at 10^{-14}. This is an effective choice for many problems in which the only perturbations are the rounding errors of floating-point arithmetic. If other perturbations are present, for example if one is approximating a function f that is only known to a certain precision, then `tol` can be increased accordingly.

1. Check if $\{f_j\}$ is real symmetric. Our first improvement involves the common case in which f is real symmetric, by which we mean that $\overline{f(z)} = f(\overline{z})$ for each f_j. Such a function should have a real discrete Fourier transform, a real matrix $\hat{Z}$, and approximations p and q with real coefficient vectors $\mathbf{a}$ and $\mathbf{b}$. However, rounding errors will typically break this symmetry, e.g., if one computes the roots of unity as `zj = exp(2i*pi*(0:N)/(N+1))`. The resulting approximations will be complex, typically with poles and zeros located nonsymmetrically with respect to the real axis, as in Figure 4.1.

One could insist that a user wishing to approximate a real symmetric function should supply a data vector $\{f_j\}$ which is itself exactly real symmetric. Such a requirement, however, is likely to confuse users. Instead our algorithm checks if $\{f_j\}$ is symmetric up to relative tolerance `tol` (lines 8–10). If it passes this test, then the imaginary parts of entries of $\tilde{Z}$ are discarded as well as the imaginary parts of the computed vector $\mathbf{a}$, which are introduced by the FFT (lines 16–17).

2. If N is odd, check if $\{f_j\}$ is even or odd. A similar issue arises when f is even or odd. In these cases one might expect q should be even, while p should have the same parity as f. In fact, however, this expectation is only valid when N is odd, so that $N + 1$ is even. On the 3-point grid associated with $N = 2$, for example, one could hardly expect that an even function f must have an even interpolant.

Accordingly, we take no steps to enforce even or odd symmetry when N is even, but if N is odd, we follow a procedure similar to the the one before. The data vector $\{f_j\}$ is tested for even or odd symmetry up to a relative tolerance `tol` (lines 12–13), and if it passes the test, the appropriate structure is forced upon $\tilde{Z}$, as follows from (3.3): if $\{f_j\}$ is even, then odd diagonals of $\tilde{Z}$ are zero, and if $\{f_j\}$ is odd, then even diagonals are zero (lines 25–26 and 34–35). In both of these cases the row and column dimensions of $\tilde{Z}$ can be cut in half.

For many practical applications the user will want a least-squares fit with $N \gg m + n$, and here the loss of symmetry would be disturbing even though in principle, $\{f_j\}$ can only be truly symmetric when N is odd. We address this issue by including a recommendation in the comment lines of the code that if $N \gg m + n$, then N should be chosen to be odd.

3. Remove contributions from negligible singular values of $\tilde{Z}$. For larger values of m and n, the matrix $\tilde{Z}$ of (3.5) often has a number of singular values at a level close to machine precision. We make this notion precise by defining a singular value of $\tilde{Z}$ to be *negligible* if it is smaller than `tol` times $\max_j |f_j|$, where `tol` is a number set by default to 10^{-14}. Let τ be the number of negligible singular values of $\tilde{Z}$ (lines 28 and 40). If $\tau = 0$, then the minimal singular vector defines a denominator polynomial q and then a numerator polynomial p for which the error $\|p - fq\|_N$ of (2.4) is equal to the smallest singular value and thus minimal, and we have solved the linearized least-squares problem. If $\tau = 1$, then the same solution has a negligible error $\|p - fq\|_N$, and we have solved the interpolation problem. If $\tau \geq 2$, then there are τ different linearly independent denominator polynomials q for which the error $\|p - fq\|_N$ can be made negligible. By considering linear combinations, we see that in this case there must exist a denominator polynomial of degree $n - (\tau - 1)$ with the same property, and for robustness it is a good idea to find such a solution so as to prevent the appearance of spurious poles.

ETNA
Kent State University
http://etna.math.kent.edu


```
function [r,a,b,mu,nu,poles,residues] = ratdisk(f,m,n,N,tol)
%  Input: Function f or vector of data at zj = exp(2i*pi*(0:N)/(N+1))
%           for some N>=m+m.  If N>>m+n, it is best to choose N odd.
%         Maximal numerator, denominator degrees m,n.
%         An optional 5th argument specifies relative tolerance tol.
%           If omitted, tol = 1e-14.  Use tol=0 to turn off robustness.
% Output: function handle r of exact type (mu,nu) approximant to f
%         with coeff vectors a and b and optional poles and residues.
% P. Gonnet, R. Pachon, L. N. Trefethen, January 2011

if nargin<4, if isfloat(f), N=length(f)-1;
  else N=m+n; end, end                       % do interpolation if no N given
N1 = N+1;                                    % no. of roots of unity
if nargin<5, tol = 1e-14; end                % default rel tolerance 1e-14
if isfloat(f), fj = f(:);                    % allow for either function
else fj = f(exp(2i*pi*(0:N)'/(N1))); end     %   handle or data vector
ts = tol*norm(fj,inf);                       % absolute tolerance
M = floor(N/2);                              % no. of pts in upper half-plane
f1 = fj(2:M+1); f2 = fj(N+2-M:N1);           % fj in upper, lower half-plane
realf = norm(f1(M:-1:1)-conj(f2),inf)<ts;    % true if fj is real symmetric
oddN = mod(N,2)==1;                          % true if N is odd
evenf = oddN & norm(f1-f2,inf)<ts;           % true if fj is even
oddf  = oddN & norm(f1+f2,inf)<ts;           % true if fj is odd
row = conj(fft(conj(fj)))/N1;                % 1st row of Toeplitz matrix
col = fft(fj)/N1; col(1) = row(1);           % 1st column of Toeplitz matrix
if realf, row = real(row);                   % discard negligible imag parts
  col = real(col); end
d = xor(evenf,mod(m,2)==1);                  % either 0 or 1
while true                                   % main stabilization loop
  Z = toeplitz(col,row(1:n+1));              % Toeplitz matrix
  if ~oddf & ~evenf                          % fj is neither even nor odd
    [U,S,V] = svd(Z(m+2:N1,:),0);            % singular value decomposition
    b = V(:,n+1);                            % coeffs of q
  else                                       % fj is even or odd
    [U,S,V] = svd(Z(m+2+d:2:N1,1:2:n+1),0);  % special treatment for symmetry
    b = zeros(n+1,1); b(1:2:end) = V(:,end); % coeffs of q
  end
  if N > m+n && n>0, ssv = S(end,end);       % smallest singular value
  else ssv = 0; end                          % or 0 in case of interpolation
  qj = ifft(b,N1); qj = qj(:);               % values of q at zj
  ah = fft(qj.*fj);                          % coeffs of p-hat
  a = ah(1:m+1);                             % coeffs of p
  if realf a = real(a); end                  % discard imag. rounding errors
  if evenf a(2:2:end) = 0; end               % enforce even symmetry of coeffs
  if  oddf a(1:2:end) = 0; end               % enforce odd symmetry of coeffs
  if tol>0                                   % tol=0 means no stabilization
    ns = n;                                  % no. of singular values
    if oddf|evenf, ns = floor(n/2); end
    s = diag(S(1:ns,1:ns));                  % extract singular values
    nz = sum(s-ssv<=ts);                     % no. of sing. values to discard
    if nz == 0, break                        % if no discards, we are done
    else n=n-nz; end
  else break                                 % no iteration if tol=0.
  end
end                                          % end of main loop
nna = abs(a)>ts; nnb = abs(b)>tol;           % nonnegligible a and b coeffs
kk = 1:min(m+1,n+1);                         % indices a and b have in common
a = a(1:find(nna,1,'last'));                 % discard trailing zeros of a
b = b(1:find(nnb,1,'last'));                 % discard trailing zeros of b
if length(a)==0 a=0; b=1; end                % special case of zero function
mu = length(a)-1; nu = length(b)-1;          % exact numer, denom degrees
r = @(z) polyval(a(end:-1:1),z)...           % function handle for r
      ./polyval(b(end:-1:1),z);
if nargout>5                                 % only compute poles if necessary
  poles = roots(b(end:-1:1));                % poles
  t = max(tol,1e-7);                         % perturbation for residue estimate
  residues = t*(r(poles+t)-r(poles-t))/2;    % estimate of residues
end
```

FIG. 5.1. *Matlab code* `ratdisk` *for robust rational interpolation and linearized least squares.*

To achieve this, our strategy in cases with $\tau \geq 2$ is to reduce the denominator degree from n to $n-(\tau-1)$ and start the approximation process again, now inevitably as a least-squares problem rather than interpolation since N is unchanged (line 42). The shrinking of the denominator degree can be justified quantitatively by noting that the contribution of a singular value of size ε can only affect the error norm (2.4) by ε. Consequently, discarding such a contribution can at worse increase the error by ε, with the great benefit of eliminating a probably spurious pole.

As the examples of the next section will show, the effect of discarding these negligible singular values is an elimination of most of the Froissart doublets that otherwise appear in plots such as those of Figure 4.1.

4. Remove the degeneracy if the smallest singular value of $\tilde{Z}$ is multiple. In approximation cases where the error $\|p - fq\|_N$ of (2.4) cannot be brought down to machine precision, the smallest singular value of $\tilde{Z}$ will not be negligible. Nevertheless, it may be multiple, and in this case the choice of q is again nonunique. Such a case corresponds to p and q potentially sharing a common factor. For example, if the best type $(1,1)$ approximation to a function f on the $(N+1)$st roots of unity is the constant 1, then $p=q=1$ and $p=q=z$ are both solutions to the least-squares problem. Some situations like this are avoided by the special steps described above that are taken if f is even or odd, but other degeneracies are not caught by those tests, such as a function like $\cos(z)+z^7$ (not even, but its low-order approximations should be even if N is odd) or $\log(2+z^3)$ (Taylor series containing only powers of z divisible by 3).

To remove such degeneracies we apply a procedure just like the one desribed above for negligible singular values. If the smallest singular value of $\hat{Z}$ has multiplicity $\tau \geq 2$, we reduce the denominator degree from n to $n-(\tau-1)$ and start the approximation process again (lines 28, 40, and 42).

5. Discard negligible trailing coefficients of p and q. Sometimes the numerator or denominator polynomials generated by the methods we have described have one or more highest-order coefficients that are zero or negligible. For example, this will be true if f is even and (m,n) is not of the form (even,even), or if f is odd and (m,n) is not of the form (odd,even). In this case it is appropriate to delete the negligible coefficient(s) and return polynomials of lower order (lines 46–49).

As we have seen, Figure 5.1 lists our robust Matlab program `ratdisk`. The user provides a function f vector of data `fj` and nonnegative integers m and n, and optionally a tolerance `tol` to override the default value of 10^{-14}. Setting `tol` $=0$ leads to a pure interpolation calculation as in the code segment of Section 3, with no robustness features. The program computes a rational approximant r and returns a function handle `r` to evaluate this function together with its coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ and exact numerator and denominator degrees $\mu \leq m$ and $\nu \leq n$. We say that the rational function r returned is of *exact type* (μ, ν). The poles and residues are also optionally returned for plots such as those of this paper. Computing poles takes $O(\nu^3)$ operations, so one would normally not request this output.

The style of `ratdisk` is very compressed, with fewer comments and tests than one would expect in fully developed software, but this program includes all our robustness strategies and should work in many applications.

6. Examples. Figures 6.1–6.9 show examples spanning a wide range of functions and approximation orders. This time, each figure presents four plots instead of two. The first row in each case corresponds to `ratdisk` with `tol` $=0$, that is, to the idealized algorithm of Section 3, just as in Figure 4.1, while the second row corresponds to `ratdisk` in its robust mode with the default value `tol` $=10^{-14}$.

ETNA
Kent State University
http://etna.math.kent.edu

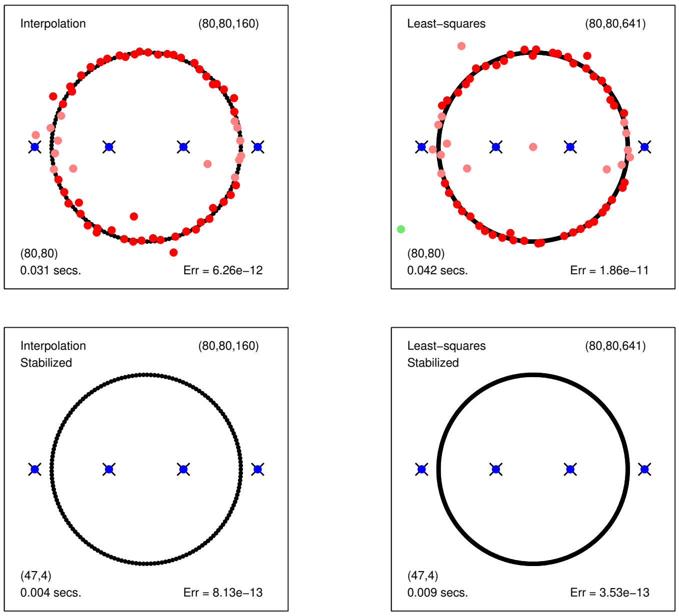

FIG. 6.1. *Type* $(80, 80)$ *approximation of* $\tan(4z)$ *again, now including robust* `ratdisk` *approximation in the second row. The spurious poles are gone. Note that the exact type has shrunk to* $(47, 4)$*, which is of the (odd, even) form appropriate in the approximation of an odd function.*

The discussion of the examples is given in the captions of the figures. We see that in almost every case, the `ratdisk` algorithm removes the spurious poles.

7. The limit $N \to \infty$ and an analogue of the Padé table. This paper concerns rational approximation of a function f in $N + 1$ points on a circle, whose radius τ can of course be varied. If f is analytic at $z = 0$, then in the limit $\tau \to 0$ and $N \to \infty$ one would expect the approximants to approach Padé approximants, at least generically [27]. Recall that the type (m, n) *Padé approximant* to f is the unique function $r \in R_{mn}$ whose Taylor series matches that of f as far as possible [1]:

$$f(z) - r(z) = O(z^{\text{maximum}}).$$

Generically the degree of matching is exactly $O(z^{m+n+1})$, but in special cases it can be higher or lower. For example, if f is even or odd, then r will have the same symmetry regardless of m and n, so the first nonzero term in the expansion of $f(z) - r(z)$ will be even or odd, respectively.

Padé approximation is an elegant and fundamental notion of mathematics, but for computation, approximation on circles of finite radius may be more convenient. To calculate the coefficients of a Padé approximation, a straightforward method is to set up a set of linear equations involving the Taylor coefficients of p and q, and these equations have a Toeplitz structure analogous to that of $\hat{Z}$ of (3.2). Instead of the values $\{f_j\}$ at roots of unity that appear in (3.2), however, such a calculation requires the Taylor coefficients of f. If one is starting from a function f rather than from Taylor coefficients, then these must be computed in one way or another, and a standard method is to sample f in equispaced points on a circle about 0 and then use the FFT [7, 14, 15]. In this paper, since we are approximating on a circle rather than at a point, the steps of computing coefficients by the FFT and generating the rational approximation are combined into one.

ETNA
Kent State University
http://etna.math.kent.edu

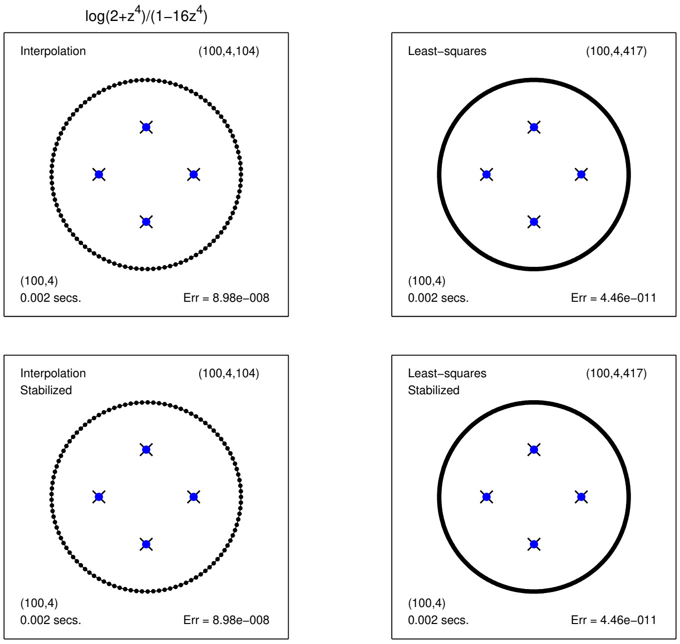

FIG. 6.2. *Type* $(100, 4)$ *approximation of the even function* $\log(2 + z^4)/(1 - 16z^4)$*. Here, since* n *has been specified as low as* 4*, the robustness features make no significant difference. The next figure, Figure 6.3, shows what happens if* n *is increased.*

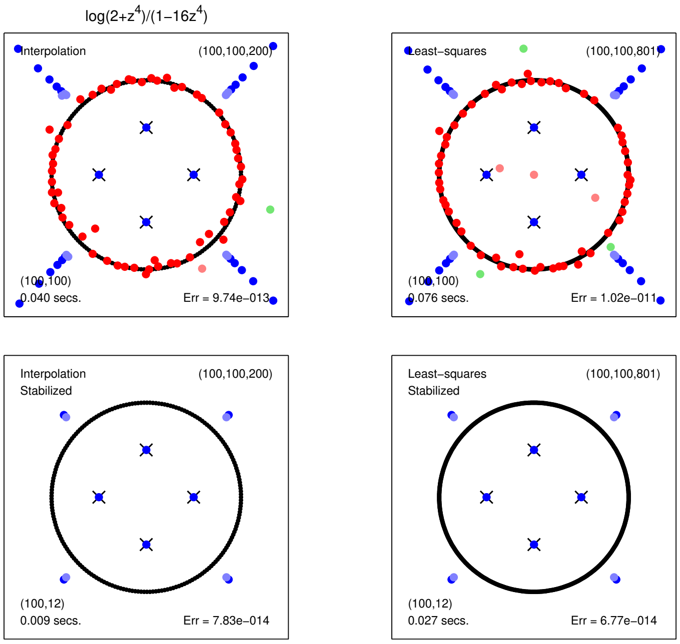

FIG. 6.3. *For type* $(100, 100)$ *approximation of* $\log(2 + z^4)/(1 - 16z^4)$*, there are many spurious poles in addition to the useful poles tracking the fourfold symmetric branch cuts. Note that as in Figures 4.1 and 6.1, the spurious poles are not symmetric, even though the function is even. In the lower plots the symmetries are enforced and the type is reduced, with both* μ *and* ν *being divisible by* 4 *because of the fourfold symmetry.*

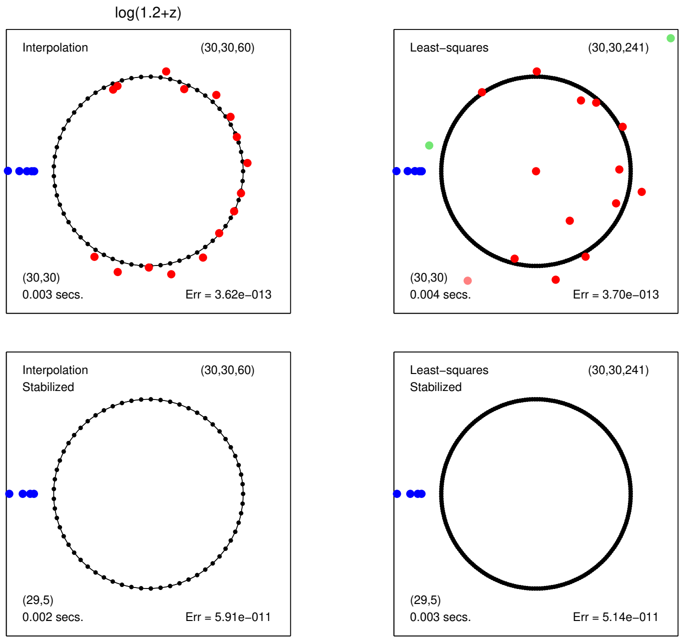

FIG. 6.4. *Type* $(30,30)$ *approximation of a function with a single branch cut* $(-\infty,-1.2]$. *As in the last figure, we see "green and blue poles" with significant residues lining up along the branch cut and performing a useful approximation function.*

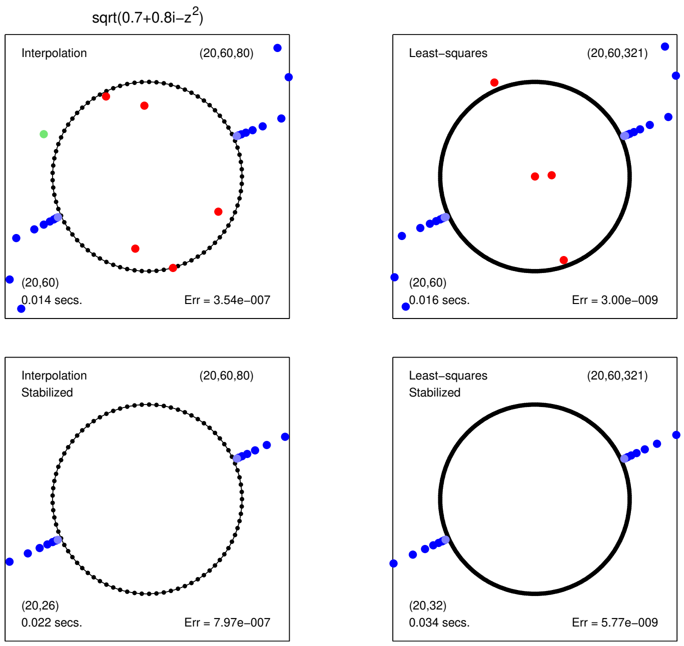

FIG. 6.5. *Type* $(20,60)$ *approximation of* $\sqrt{0.7+0.8i-z^2}$, *a complex even function with two branch cuts.*

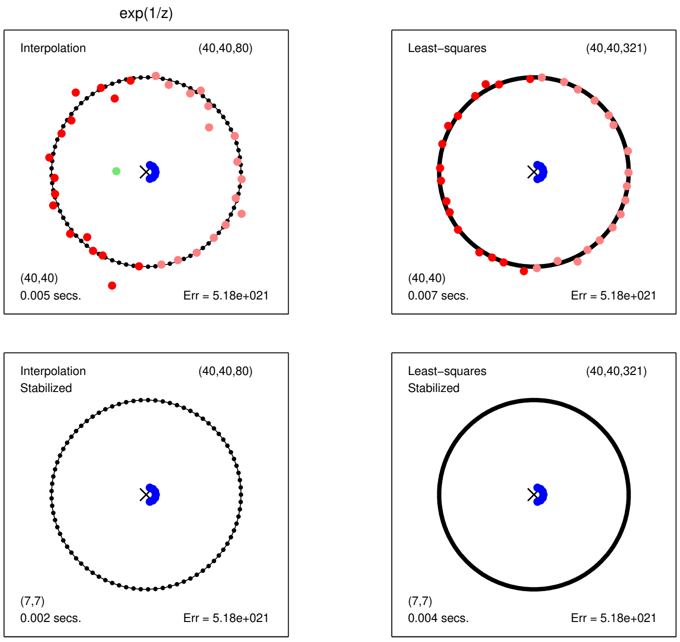

FIG. 6.6. *Type* $(40, 40)$ *approximation of* $\exp(1/z)$*, with an essential singularity at the origin. The* `Err` *measures come out nearly infinite. If* $|f(z) - r(z)|$ *is measured just at the grid points in the disk with* $|z| > 0.5$*, however, they shrink to* $1.04e{-}11$*,* $4.26e{-}11$*,* $3.94e{-}11$*, and* $3.82e{-}11$*.*

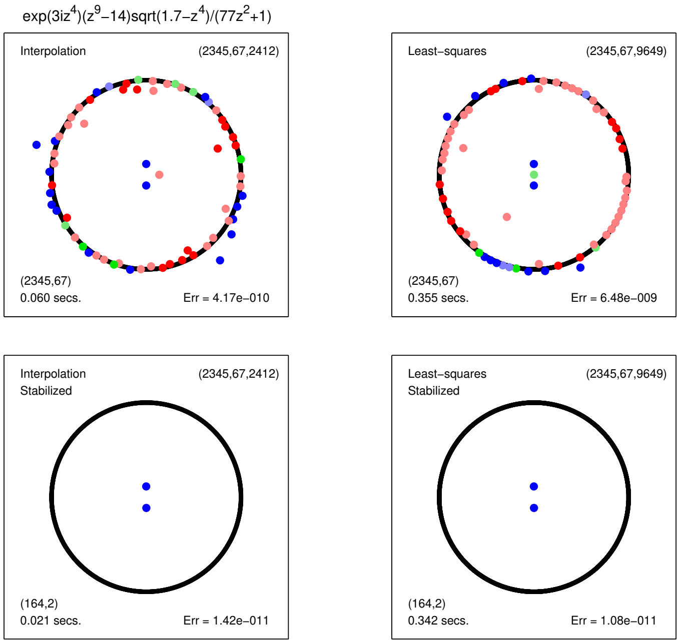

FIG. 6.7. *Approximation of type* $(2345, 67)$ *to a complex function with two poles and four branch cuts. The polynomial degree is so high that the robust algorithm does not use the denominator at all except to capture the poles.*

ETNA
Kent State University
http://etna.math.kent.edu

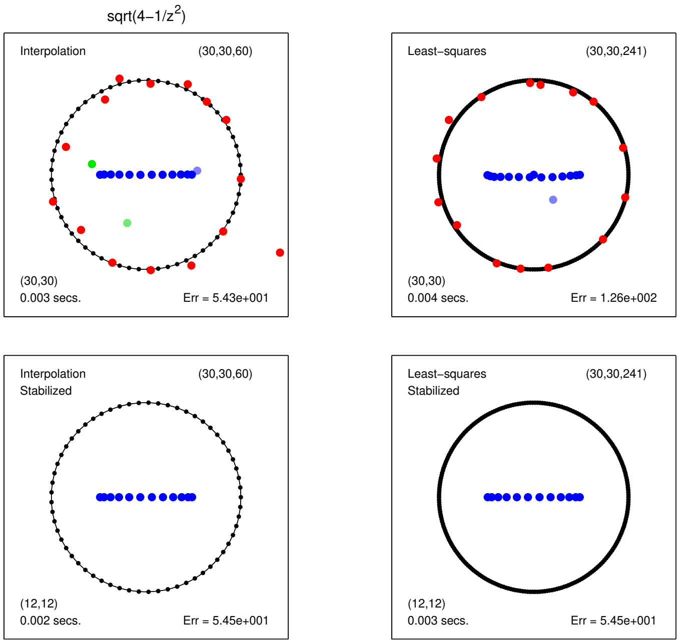

FIG. 6.8. *Type* $(30, 30)$ *approximation of* $\sqrt{4 - z^{-2}}$*, which has a branch cut* $[-1/2, 1/2]$*. Poles are placed along the branch cut. In the upper row the poles are far from symmetric, but the lower row shows the expected symmetries enforced by* `ratdisk`*. If* $|f(z) - r(z)|$ *is measured just at grid points with* $|\mathrm{Im}\, z| > 0.25$*, the* `Err` *values shrink to* $1.27e{-5}$*,* $2.51e{-5}$*,* $1.36e{-5}$*, and* $1.38e{-5}$*.*

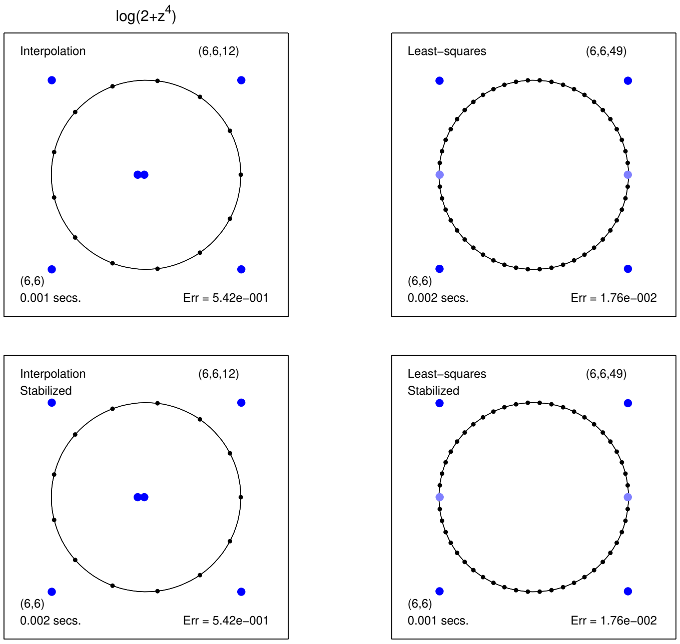

FIG. 6.9. *Type* $(6, 6)$ *approximation of* $\log(2 + z^4)$*. Notice that although* f *has four-fold symmetry, the exact type* (μ, ν) *is not divisible by* 4*, even in the least-squares computation of the bottom-right. This is because* $N = 49$ *is fairly small. For larger* N *one gets the expected symmetry, as shown in the final panel of Figure 7.1.*

ETNA
Kent State University
http://etna.math.kent.edu

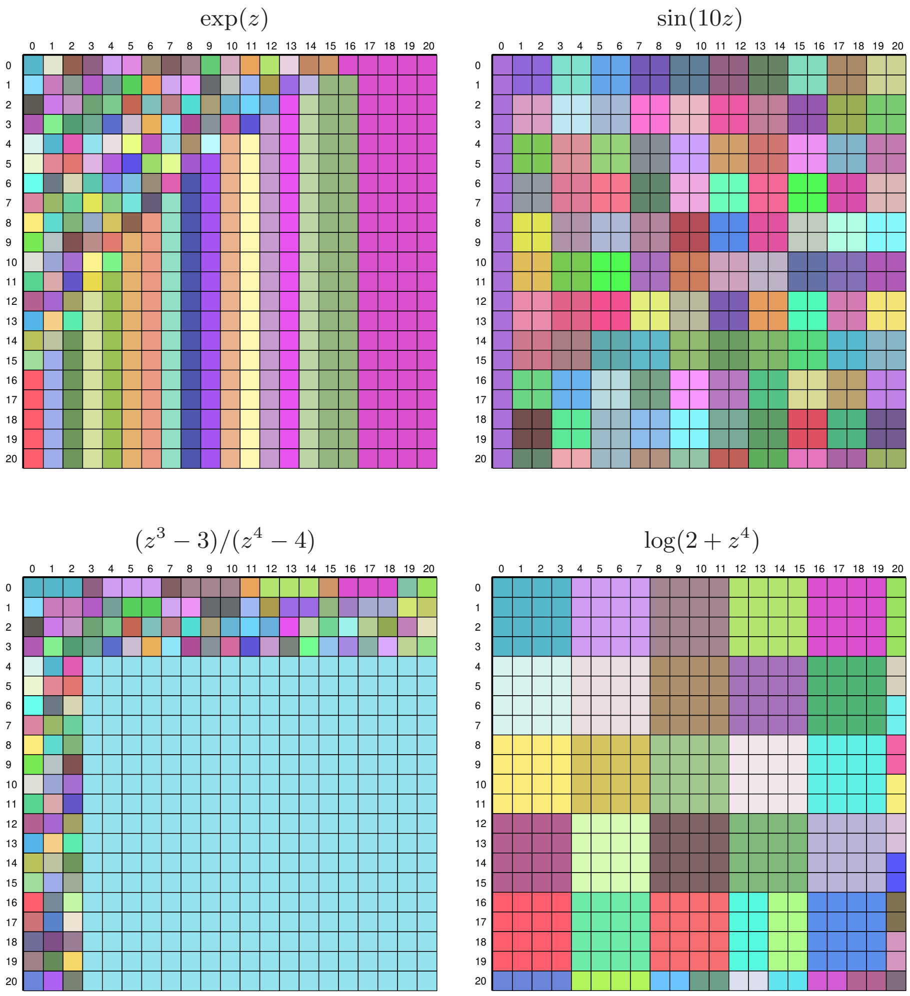

FIG. 7.1. *Tables of linearized least-squares approximants to four functions, with m on the horizontal axis and n on the vertical axis. Each color corresponds to the exact type (μ,ν) of a* `ratdisk` *approximation computed with* $\mathtt{tol}=10^{-14}$ *and $N=1023$, so that colors reveal blocks of identical entries or at least entries of identical exact types. For $\exp(z)$ all the blocks are distinct until the function is resolved to machine precision, after which the denominator degrees are systematically reduced as far as possible. For the odd function $\sin(10z)$, 2×2 square blocks appear in the table; because of the factor 10, this function is never resolved with $m,n\le 20$ to machine precision, so no further degeneracies appear in the table. For $(z^3-3)/(z^4-4)$ we get an infinite square block since the function is rational and thus resolved exactly for $m\ge 3$ and $n\ge 4$. Finally, for $\log(2+z^4)$ we get 4×4 blocks until m and n get large enough for the approximations to have accuracy close to machine precision; these anomalies go away if* `tol` *is increased to 10^{-12}.*

A particularly natural finite-radius analogue of Padé approximation emerges if we consider the limit $N\to\infty$, in which the linearized least-squares problem (2.3)–(2.4) is posed on the unit circle rather than a discrete set of points. If r_{mn} is the type (m,n) approximation to a fixed function f determined in this way, then we may imagine a table of approximations to f, analogous to the usual Padé table, with m displayed horizontally and n vertically. This notion would apply both as a mathematical abstraction, and also in numerical form as computed in floating-point arithmetic with a tolerance $\mathtt{tol}>0$.

ETNA
Kent State University
http://etna.math.kent.edu

Figure 7.1 gives a graphical illustration of these Padé-like tables for approximation on the unit circle of $\exp(z)$, $\sin(z)$, $(z^3-3)/(z^4-4)$, and $\log(2+z^4)$. Each square is given a random color associated not with its allowed type (m,n) but with its exact type (μ,ν) as obtained in a `ratdisk` computation with $N=1023$. The computation of the whole table takes less than a second on a desktop computer. For $\exp(z)$ we see distinct approximations in each square for smaller m and n, then numerical degeneracy as machine precision is reached and further increase of m and n serves no purpose. For $\sin(z)$ we see an approximate 2×2 square block structure caused by the fact that the function is odd [1, 26]. The third example is rational, and this is reflected in the infinite block for $m\geq 3$, $n\geq 4$. Finally, the last function is fourfold symmetric, and we see see 4×4 blocks down to the level where machine precision begins to be felt.

8. Evaluating radial basis function interpolants. Radial basis functions (RBFs) are a flexible tool for representing scattered data in multiple dimensions by smooth interpolants [4, 6, 28]. When applied to the solution of partial differential equations, they offer the prospect of combining the high accuracy of spectral methods with great freedom with respect to the geometry. Following ideas of Fornberg and his coauthors, we shall show by an example that robust rational interpolation may play a role in such calculations. The RBFs used in the example are Gaussians.

Suppose we have an M-vector $\mathbf{g}$ of data values $\{g_j\}$ at distinct points $\mathbf{u}_j$ in a region of the $\mathbf{u}$-plane. We wish to find an *RBF interpolant* of the form

$$s^{(\varepsilon)}(\mathbf{u})=\sum_{j=1}^{M}\lambda_j^{(\varepsilon)}e^{-\varepsilon\|\mathbf{u}-\mathbf{u}_j\|^2},$$

where $\varepsilon>0$ is a fixed number called the *shape parameter* (usually written ε^2 in the RBF literature) and $\boldsymbol{\lambda}^{(\varepsilon)}$ is a vector of coefficients. The interpolation conditions take the form of an $M\times M$ linear system of equations for $\boldsymbol{\lambda}^{(\varepsilon)}$,

$$A^{(\varepsilon)}\boldsymbol{\lambda}^{(\varepsilon)}=\mathbf{g},\qquad a_{ij}^{(\varepsilon)}=e^{-\varepsilon\|\mathbf{u}_i-\mathbf{u}_j\|^2}. \tag{8.1}$$

It was proved by Bochner in 1933 that $A^{(\varepsilon)}$ is always nonsingular, so a unique solution to the interpolation problem exists [2, 4, 28].

The dependence on ε is a key point in RBF fits. When ε is large, the basis functions $\exp(-\varepsilon\|\mathbf{u}-\mathbf{u}_j\|^2)$ are narrowly localized and the matrix $A^{(\varepsilon)}$ is well conditioned. Much greater interpolating accuracy, however, is potentially obtained for smaller values of ε, for which the RBFs are less localized. The difficulty is that in this regime the condition number of $A^{(\varepsilon)}$ reaches huge values, easily exceeding the inverse of machine epsilon in floating point arithmetic. The challenge is to evaluate $s^{(\varepsilon)}(\mathbf{u})$, which is a well-behaved function of $\mathbf{u}$, despite the ill-conditioning of the matrix. With their "Contour-Padé algorithm," Fornberg and his coauthors have proposed that one method for this is to regard ε as a complex parameter. For values of ε on the unit circle, for example, the matrix $A^{(\varepsilon)}$ may be reasonably well conditioned, whereas perhaps it is $\varepsilon=0.1$ that one cares about, corresponding to an impossibly ill-conditioned matrix. For each fixed point $\mathbf{u}$, Cramer's rule shows that $s^{(\varepsilon)}(\mathbf{u})$ is a meromorphic function of ε, and the idea is to evaluate $s^{(\varepsilon)}(\mathbf{u})$ for values of ε on the unit circle, then use a rational interpolant or least-squares fit to extrapolate in to $\varepsilon=0.1$. This idea was first proposed in [10].

We give just one example. Let $\{\mathbf{u}_j\}$ be the set of points scattered in the unit disk

$$|\mathbf{u}_j|=\sqrt{j/M}e^{ij},\quad 1\leq j\leq M$$

ETNA
Kent State University
http://etna.math.kent.edu

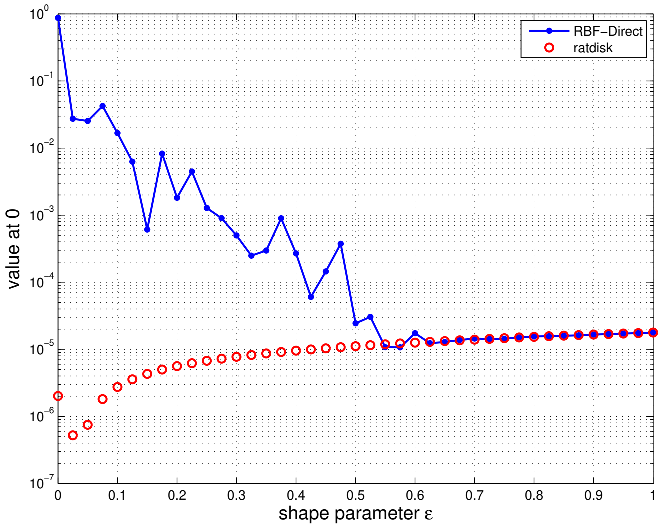

FIG. 8.1. *Evaluation of an RBF interpolant through* 80 *data points by direct linear algebra (blue) and* `ratdisk` *(red circles). Rational least-squares approximation circumvents the ill-conditioning of the matrices in the linear algebra formulation.*

with $M = 80$. Let $g(\mathbf{u})$ be the function

$$g(\mathbf{u}) = \frac{\cos(\mathbf{u}^{(1)})\tanh(\mathbf{u}^{(2)})}{\|\mathbf{u} - (1,1)\|}$$

where $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$ denote the two components of $\mathbf{u}$. At $\mathbf{u} = \mathbf{0}$, g takes the value 0, and we wish to evaluate the RBF interpolant at the same point, $s^{(\varepsilon)}(\mathbf{0})$, for various ε. For small ε, the exact value of $s^{(\varepsilon)}$ is very close to $g(\mathbf{0}) = 0$. Figure 8.1 shows values of $s^{(\varepsilon)}$ by the "RBF-Direct" method of solving the ill-conditioned system (8.1) and by `ratdisk` with $(m, n, N) = (60, 20, 127)$.

As Fornberg and coauthors have pointed out, the range of RBF problems for which rational interpolation or least-squares is effective may be rather limited. When the number of data points is much higher than in the example of Figure 8.1, one is faced with meromorphic functions of ε with so many poles that these techniques may break down. For such problems an alternative known as the RBF-QR method is sometimes effective [8, 9].

9. Discussion. We have presented a robust numerical method and the Matlab code `ratdisk` for rational approximation on the unit circle, with examples and an application to the evaluation of radial basis function interpolants. We believe the method can be useful for many practical problems and mathematical explorations. We conclude with some remarks about various issues.

Increasing `tol`. In all our experiments, the relative tolerance parameter `tol` has been set to 0 (for pure interpolation/least-squares) or 10^{-14} (for robust computation with rounding errors). In applications, users may want to adjust this parameter. Even when the only perturbations are rounding errors, there might be advantages to increasing `tol` in applications where robustness is more important then very high accuracy. If other perturbations in the data are present, then a correspondingly larger value of `tol` will almost certainly be appropriate.

Ill-conditioning. However robust our algorithm, rational approximation remains an ill-conditioned problem. For example, suppose one uses an (m, n) approximant to attempt to locate some poles of a meromorphic function numerically. As various practitioners have noted over the years, type $(10, 10)$ approximations will often yield more accurate poles than type $(40, 40)$, and the reason is simple—as more coefficients become available to achieve a fit, it becomes less necessary for the approximation to locate the poles exactly right to achieve optimality. We have seen this effect at several points in this paper, such as the imperfect 4×4 block structure in the final plot of Figure 7.1. For another example, it is interesting to approximate a function like $f(z) = (z^2 - 2)/(z^3 + 3)$ by rational functions with $m, n \to \infty$. For small m and n, the denominator q may come out just as expected; for large enough m and n our robustness procedures will reduce q to a constant; but for in-between values of m and n, q will typically be a cubic with coefficient far from the "correct" ones. Nevertheless the rational function will be an excellent approximation to f.

Nonlinear least-squares. True rational approximations defined by (2.5), as opposed to their linearized analogues defined by (2.4), are well known to pose difficulties in some circumstances. Nevertheless they are of interest, and we have successfully experimented with the computation of nonlinear approximations by a sequence of iteratively reweighted linear ones using a variant of `ratdisk` modified to incorporate a weight vector. Such an iteration is the basis of the differential correction algorithm for rational best approximation [5]. This work will be reported elsewhere.

Beyond roots of unity. Roots of unity are beautifully convenient: the basis of monomials z^k is numerically stable, formulas written in this basis have a familiar appearance, and function values are linked to coefficients by the FFT. For rational interpolants and least-squares approximants on an interval $[a, b]$, however, one would need to use a different set of interpolation points, and a good choice would be scaled and translated Chebyshev points $x_j = a + (b - a)\cos(j\pi/N)$, $0 \le j \le N$. The monomials would no longer be a good basis and a good alternative would be scaled and translated Chebyshev polynomials $T_k(-1 + 2(x - a)/(b - a))$ [21]. These tools of Chebyshev polynomials and Chebyshev points have the same mathematical advantages on $[a, b]$ as roots of unity and monomials on the unit circle, though they are conceptually more complicated since most scientists and engineers are less familiar with them. The methods we have described also generalize to arbitrary point sets, though here one loses the FFT. Also, unless a good basis is known, it becomes crucial to use barycentric interpolation to evaluate r, as described at the end of Section 3. For details see [19].

Availability of code. The Matlab program `ratdisk` is available from the third author at `http://people.maths.ox.ac.uk/trefethen/other.html`.

A theoretical challenge. We would like to close by proposing a theoretical opportunity. As was mentioned in Section 4, the Nuttall–Pommerenke and Stahl theorems reflect the fact that as $n \to \infty$, type (n, n) Padé approximations to certain functions do not converge everywhere where one might expect, because Froissart doublets may appear at wandering locations [18, 20, 25]. Instead, these theorems only guarantee convergence in capacity. However, we have proposed methods for eliminating Froissart doublets through the use of the singular value decomposition. Our method has involved a fixed tolerance such as 10^{-14}, since our focus is on rounding errors, but from a theoretical point of view, in exact arithmetic, one might consider a similar algorithm with tolerance decreasing to zero at a prescribed rate as $n \to \infty$. Some form of this idea might lead to a precise notion of a *robust Padé approximation* that might be guaranteed to converge pointwise, not just in capacity. A theorem establishing such a fact would be a beautiful link between rational approximation theory and practice.

Acknowledgments. This work arose from discussions with radial basis function experts Bengt Fornberg and Grady Wright. In particular, it was Wright who persuaded us of the importance in practical applications of eliminating spurious poles. We also thank Ed Saff for pointing us to some of the literature on Froissart doublets.

REFERENCES

[1] G. A. BAKER, JR. AND P. R. GRAVES-MORRIS, *Padé Approximants*, 2nd ed., Cambridge U. Press, Cambridge, 1996.
[2] S. BOCHNER, *Monotone Funktionen, Stieltjes Integrale und harmonische Analyse*, Math. Ann., 108 (1933), pp. 378–410.
[3] D. BRAESS, *Nonlinear Approximation Theory,* Springer, Berlin, 1986.
[4] M. D. BUHMANN, *Radial Basis Functions: Theory and Implementations,* Cambridge U. Press, Cambridge, 2003.
[5] E. W. CHENEY AND H. L. LOEB, *On rational Chebyshev approximation*, Numer. Math., 4 (1962), pp. 124–127.
[6] G. E. FASSHAUER, *Meshfree Approximation Methods with Matlab,* World Scientific, Hackensack, 2007.
[7] B. FORNBERG, *ALGORITHM 579 CPSC: Complex power series coefficients*, ACM Trans. Math. Softw., 7 (1981), pp. 542–547.
[8] B. FORNBERG, E. LARSSON, AND N. FLYER, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput., 33 (2011), pp. 869–892.
[9] B. FORNBERG AND C. PIRET, *A stable algorithm for flat radial basis functions on a sphere*, SIAM J. Sci. Comput., 30 (2007), pp. 60–80.
[10] B. FORNBERG AND G. WRIGHT, *Stable computation of multiquadric interpolants for all values of the shape parameter*, Comput. Math. Appl., 48 (2004), 853–867.
[11] M. FROISSART, *Approximation de Padé: application à la physique des particules élémentaires*, RCP Programme No. 25, v. 9, CNRS, Strasbourg, 1969, pp. 1–13.
[12] J. GILEWICZ AND M. PINDOR, *Padé approximants and noise: rational functions*, J. Comput. Appl. Math., 105 (1999), pp. 285–297.
[13] J. M. HAMMERSLEY, *The zeros of a random polynomial*, in Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1954–1955, J. Neyman, ed., U. California Press, 1956, pp. 89–111.
[14] P. HENRICI, *Fast Fourier methods in complex analysis*, SIAM Rev., 21 (1979), pp. 481–527.
[15] L. N. LYNESS AND G. SANDE, *Algorithm 413: ENTCAF and ENTCRE: evaluation of normalized Taylor coefficients of an analytic function*, Comm. ACM, 14 (1971), pp. 669–675.
[16] G. A. MEZINCESCU et al., *Distribution of roots of random real generalized polynomials*, J. Stat. Phys., 86 (1997), pp. 675–705.
[17] R. DE MONTESSUS DE BALLORE, *Sur les fractions continues algébriques*, Bull. Soc. Math. de France, 30 (1902), pp. 28–36.
[18] J. NUTTALL, *The convergence of Padé approximants of meromorphic functions*, J. Math. Anal. Appl., 31 (1970), pp. 147–153.
[19] R. PACHÓN, P. GONNET AND J. VAN DEUN, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, SIAM J. Numer. Anal., in press.
[20] CH. POMMERENKE, *Padé approximants and convergence in capacity*, J. Math. Anal. Appl., 41 (1973), pp. 775–780.
[21] T. J. RIVLIN, *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*, 2nd ed., Wiley, New York, 1990.
[22] E. B. SAFF, *An extension of Montessus de Ballore's theorem on the convergence of interpolating rational functions*, J. Approximation Theory, 6 (1972), pp. 63–67.
[23] L. A SHEPP AND R. J. VANDERBEI, *The complex zeros of random polynomials*, Trans. Amer. Math. Soc., 347 (1995), pp. 4365–4384.
[24] B. SHIFFMAN AND S. ZELDITCH, *Equilibrium distribution of zeros of random polynomials*, Int. Math. Res. Not., 2003, pp. 25–49.
[25] H. STAHL, *The convergence of Padé approximants to functions with branch points*, J. Approximation Theory, 91 (1997), pp. 139–204.
[26] L. N. TREFETHEN, *Square blocks and equisoscillation in the Padé, Walsh, and CF tables,* in *Rational Approximation and Interpolation,* P. R. Graves-Morris, E. B. Saff, and R. S. Varga, eds., Lect. Notes in Math. 1105, Springer, Berlin, 1984, pp. 170–181.
[27] L. N. TREFETHEN AND M. H. GUTKNECHT, *On convergence and degeneracy in rational Padé and Chebyshev approximation*, SIAM J. Math. Anal., 16 (1985), pp. 198–210.
[28] H. WENDLAND, *Scattered Data Approximation,* Cambridge U. Press, Cambridge, 2005.