EXTENSION OF CHEBFUN TO PERIODIC FUNCTIONS

GRADY B. WRIGHT*, MOHSIN JAVED , HADRIEN MONTANELLI , AND LLOYD N. TREFETHEN†

Abstract. Algorithms and underlying mathematics are presented for numerical computation with periodic functions via approximations to machine precision by trigonometric polynomials, including the solution of linear and nonlinear periodic ordinary differential equations. Differences from the nonperiodic Chebyshev case are highlighted.

Key words. Chebfun, Fourier series, trigonometric interpolation, barycentric formula

AMS subject classifications. 42A10, 42A15, 65T40

1. Introduction. It is well known that trigonometric representations of periodic functions and Chebyshev polynomial representations of nonperiodic functions are closely related. Table 1 lists some of the parallels between these two situations. Chebfun, a software system for computing with functions and solving ordinary differential equations [3, 11, 25], relied entirely on Chebyshev representations in its first decade. This paper describes its extension to periodic problems initiated by the first author and released with Chebfun Version 5.1 in December 2014.

TABLE 1.1

Some parallels between trigonometric and Chebyshev settings. The row of contributors' names is just a sample of some key figures.

Trigonometric	**Chebyshev**
$t \in [0, 2\pi]$	$x \in [-1, 1]$
periodic	nonperiodic
$\exp(ikt)$	$T_k(x)$
trigonometric polynomials	algebraic polynomials
equispaced points	Chebyshev points
trapezoidal rule	Clenshaw–Curtis quadrature
companion matrix	colleague matrix
Horner's rule	Clenshaw recurrence
Fast Fourier Transform	Fast Cosine Transform
Gauss, Fourier, Zygmund, ...	Bernstein, Lanczos, Clenshaw, ...
Chebfun since 2014	Chebfun since 2003

Though Chebfun is a software product, the main focus of this paper is mathematics and algorithms rather than software *per se*. What makes this subject interesting is that the trigonometric/Chebyshev parallel, though close, is not an identity. The experience of building a software system based first on one kind of representation and then extending it to the other has given the Chebfun team a uniquely intimate view of the details of these relationships. We begin this paper by listing ten differences be-

*Dept. of Mathematics, Boise State University, Boise, ID 83725-1555, USA

†Oxford University Mathematical Institute, Oxford OX2 6GG, UK. 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.

tween Chebyshev and trigonometric formulations that we have found important. This will set the stage for presentations of the problems of trigonometric series, polynomials, and projections (Section 2), trigonometric interpolants, aliasing, and barycentric formulas (Section 3), approximation theory and quadrature (Section 4), and various aspects of our algorithms (Sections 5–7).

1. One basis or two. For working with polynomials on $[-1,1]$, the only basis functions one needs are the Chebyshev polynomials $T_k(x)$. For trigonometric polynomials on $[0,2\pi]$, on the other hand, there are two equally good equivalent choices: complex exponentials $\exp(ikt)$, or sines and cosines $\sin(kt)$ and $\cos(kt)$. (One might reserve the terms "Fourier" for $\exp(ikt)$ and "trigonometric" for $\sin(kt)$ and $\cos(kt)$, but we shall say trigonometric in all cases, except that we refer to "Fourier coefficients" instead of "trigonometric coefficients".) The former is mathematically simpler; the latter is mathematically more elementary and provides a framework for dealing with even and odd symmetries. A fully useful software system for periodic functions needs to offer both kinds of representation.

2. Complex coefficients. In the $\exp(ikt)$ representation, the expansion coefficients of a real periodic function are complex. Mathematically, they satisfy certain symmetries, and a software system needs to enforce these symmetries if it is to avoid disturbing the user with imaginary rounding errors. Polynomial approximations of real nonperiodic functions, by contrast, do not lead to complex coefficients.

3. Even and odd numbers of parameters. A polynomial of degree n is determined by $n+1$ parameters, a number that may equally be odd or even. A trigonometric polynomial of degree n, by contrast, is determined by $2n+1$ parameters, always an odd number, as a consequence of the $\exp(\pm inx)$ symmetry. For most purposes it is unnatural to speak of trigonometric polynomials with an even number of degrees of freedom. Even numbers make sense, on the other hand, in the special case of trigonometric polynomials defined by interpolation at equispaced points, if one imposes the symmetry condition that the interpolant of the $(-1)^j$ sawtooth should be real, i.e., a cosine rather than a complex exponential. In this case the new complication arises that distinct formulas are needed for the even and odd cases.

4. The effect of differentiation. Differentiation lowers the degree of an algebraic polynomial, but it does not lower the degree of a trigonometric polynomial; indeed it enhances the weight of its highest-degree components. These differences have consequences throughout the field of spectral methods for the numerical solution of differential equations, with the trigonometric functions generally performing better than the polynomials.

5. Uniform resolution across the interval. Trigonometric representations have uniform properties across the interval of approximation, but polynomials are nonuniform, with much greater resolution power near the ends of $[-1,1]$ than near the middle. (See Chapter 22 of [26].)

6. Periodicity and translation-invariance. The periodicity of trigonometric representations means that a periodic chebfun constructed on $[0,2\pi]$, say, can be perfectly well evaluated at 10π or 100π; nonperiodic chebfuns have no such global validity. Thus, whereas interpolation and extrapolation are utterly different for polynomials, they are not so different in the trigonometric case. A subtler consequence of translation invariance is explained in the footnote on p. 9.

7. Operations that break periodicity. A function that is smooth and periodic may lose these properties when restricted to a subinterval or subjected to operations like rounding or absolute value. This elementary fact has the consequence that a number

of operations on periodic chebfuns require their conversion to nonperiodic form.

8. Good and bad bases. The functions $\exp(ikt)$ or $\sin(kt)$ and $\cos(kt)$ are well-behaved by any measure, and nobody would normally think of using any other basis functions for representing trigonometric functions. For polynomials, however, many people would reach for the basis of monomials x^k before the Chebyshev polynomials $T_k(x)$. Unfortunately, the monomials are exponentially ill-conditioned on $[-1,1]$: a degree-n polynomial of size 1 on $[-1,1]$ will typically have coefficients of order 2^n when expanded in the basis $1, x, \dots, x^n$. Use of this basis will cause trouble in almost any numerical calculation unless n is very small.

9. Good and bad interpolation points. For interpolation of periodic functions, nobody would normally think of using any interpolation points other than equispaced. For interpolation of nonperiodic functions by polynomials, however, equispaced points are exponentially bad, as was first fully understood by Runge [21, 22]. The mathematically appropriate choice is not obvious until one learns it: Chebyshev points, quadratically clustered near ± 1.

10. Familiarity. All the world knows and trusts Fourier analysis. By contrast, experience with Chebyshev polynomials is often the domain of experts, and it is not as widely appreciated that numerical computations based on polynomials can be trusted. (Historically, points 8 and 9 of this list have led to this mistrust.) To quote from the appendix to [26]: "In the end this may be the biggest difference between Fourier and Chebyshev analysis, the difference in their reputations."

The book *Approximation Theory and Approximation Practice* [26] serves as a summary of the mathematics and algorithms of Chebyshev technology for nonperiodic functions. The present paper, although much more condensed, can be thought of as a trigonometric analogue. In particular, Section 2 corresponds to Chapter 3 of [26], Section 3 to Chapters 2, 4, and 5, and Section 4 to Chapters 6, 7, 8, 10, and 19.

2. Trigonometric series, polynomials, and projections. Throughout this paper, we assume f is a Lipschitz continuous periodic function on $[0, 2\pi]$. Here and in all our statements about periodic functions, the interval $[0, 2\pi]$ should be understood periodically: $t = 0$ and $t = 2\pi$ are identified, and any smoothness assumptions apply across this point in the same way as for $t \in (0, 2\pi)$. An excellent treatment of some of this mathematics can be found in Chapter I of [16].

It is known that f has a unique trigonometric series, absolutely and uniformly convergent, of the form

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{ikt}, \tag{2.1}$$

with Fourier coefficients

$$c_k = \frac{1}{2\pi} \int_0^{2\pi} f(t) e^{-ikt} dt. \tag{2.2}$$

(All coefficients in our discussions are in general complex, though in cases of certain symmetries they will be purely real or imaginary.) Equivalently, we have

$$f(t) = \sum_{k=0}^{\infty} a_k \cos(kt) + \sum_{k=1}^{\infty} b_k \sin(kt), \tag{2.3}$$

with $a_0 = c_0$ and

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(t)\cos(kt)dt, \quad b_k = \frac{1}{\pi}\int_0^{2\pi} f(t)\sin(kt)dt \qquad (k \geq 1). \tag{2.4}$$

The formulas (2.4) can be derived by matching the e^{ikt} and e^{-ikt} terms of (2.3) with those of (2.1), which yields the identities

$$c_k = \frac{a_k}{2} + \frac{b_k}{2i}, \quad c_{-k} = \frac{a_k}{2} - \frac{b_k}{2i} \qquad (k \geq 1), \tag{2.5}$$

or equivalently,

$$a_k = c_k + c_{-k}, \quad b_k = i(c_k - c_{-k}) \qquad (k \geq 1). \tag{2.6}$$

Note that if f is real, then (2.4) implies that its cosine and sine coefficients a_k and b_k are real. Its exponential coefficients c_k are generally complex, and (2.5) implies that they satisfy the symmetry condition $c_{-k} = \overline{c}_k$.

The *degree n trigonometric projection* of f is the function

$$f_n(t) = \sum_{k=-n}^{n} c_k e^{ikt}, \tag{2.7}$$

or equivalently

$$f_n(t) = \sum_{k=0}^{n} a_k \cos(kt) + \sum_{k=1}^{n} b_k \sin(kt). \tag{2.8}$$

More generally, we say that a function of the form (2.7)–(2.8) is a *trigonometric polynomial of degree n*, and we let P_n denote the $(2n+1)$-dimensional vector space of all such polynomials. The trigonometric projection f_n is the least-squares approximant to f in P_n, i.e., the unique best approximation to f in the L^2 norm over $[0, 2\pi]$.

3. Trigonometric interpolants, aliasing, and barycentric formulas. Mathematically, the simplest degree n trigonometric approximation of a periodic function f is its trigonometric projection (2.7)–(2.8). This approximation depends on the values of $f(t)$ for all $t \in [0, 2\pi]$ via (2.2) or (2.4). Computationally, a simpler approximation of f is its degree n *trigonometric interpolant*, which only depends on the values at certain interpolation points. In our basic configuration, we wish to interpolate f in equispaced points by a function $p_n \in P_n$. Since the dimension of P_n is $2n+1$, there should be $2n+1$ interpolation points. We take these *trigonometric points* to be

$$t_k = \frac{2\pi k}{N}, \qquad 0 \leq k \leq N-1 \tag{3.1}$$

with $N = 2n+1$. (It is for the sake of this definition and the subsequent formulas (3.4)–(3.5) that we have taken our fundamental interval to be $[0, 2\pi]$ rather than $[-\pi, \pi]$.) The trigonometric interpolation problem goes back at least to the young Gauss's calculations of the orbit of the asteroid Ceres in 1801 [14].

It is known that there exists a unique interpolant $p_n \in P_n$ to any set of data values $f_k = f(t_k)$. Let us write p_n in the form

$$p_n(t) = \sum_{k=-n}^{n} \tilde{c}_k e^{ikt}, \tag{3.2}$$

or equivalently

$$p_n(t) = \sum_{k=0}^{n} \tilde{a}_k \cos(kt) + \sum_{k=1}^{n} \tilde{b}_k \sin(kt), \tag{3.3}$$

for some coefficients $\tilde{c}_{-n}, \ldots, \tilde{c}_n$ or equivalently $\tilde{a}_0, \ldots, \tilde{a}_n$ and $\tilde{b}_1, \ldots, \tilde{b}_n$.[1] The coefficients $\tilde{c}_k$ and c_k are related by

$$\tilde{c}_k = \sum_{j=-\infty}^{\infty} c_{k+jN} \qquad (|k| \le n) \tag{3.4}$$

(the *Poisson summation formula*), and similarly $\tilde{a}_k/\tilde{b}_k$ and a_k/b_k are related by $\tilde{a}_0 = \sum_{j=0}^{\infty} a_{jN}$ and

$$\tilde{a}_k = a_k + \sum_{j=1}^{\infty}(a_{k+jN} + a_{-k+jN}), \quad \tilde{b}_k = b_k + \sum_{j=1}^{\infty}(b_{k+jN} - b_{-k+jN}) \tag{3.5}$$

for $1 \le k \le n$. We can derive these formulas by considering the phenomenon of *aliasing*. For all j, the functions $\exp(i[k+jN]t)$ take the same values at the trigonometric points (3.1). This implies that f and the trigonometric polynomial (3.2) with coefficients defined by (3.4) take the same values at these points. In other words, (3.2) is the degree n trigonometric interpolant to f. A similar argument justifies (3.5)–(3.5), based on the fact that the functions $\cos([\pm k + jN]t)$ or $\pm\sin([\pm k + jN]t)$ also take the same values at the trigonometric points for all j.

Another interpretation of the coefficients $\tilde{c}_k, \tilde{a}_k, \tilde{b}_k$ is that they are equal to the approximations to c_k, a_k, b_k one gets if the integrals (2.2) and (2.4) are approximated by the periodic trapezoidal quadrature rule with N points [27]:

$$\tilde{c}_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikt_j}, \tag{3.6}$$

$$\tilde{a}_k = \frac{2}{N} \sum_{j=0}^{N-1} f_j \cos(kt_j), \quad \tilde{b}_k = \frac{2}{N} \sum_{j=0}^{N-1} f_j \sin(kt_j) \qquad (k \ge 1). \tag{3.7}$$

To prove this, we note that the trapezoidal rule computes the same Fourier coefficients for f as for p_n, since they take the same values at the grid points; but these must be equal to the true Fourier coefficients of p_n, since the $N = (2n+1)$-point trapezoidal rule is exactly correct for $e^{-2int}, \ldots, e^{2int}$, hence for any trigonometric polynomial of degree $2n$, hence in particular for any trigonometric polynomial of degree n times an exponential $\exp(-ikt)$ with $|k| \le n$. From (3.6)–(3.7) it is evident that the discrete Fourier coefficients $\tilde{c}_k$, $\tilde{a}_k$, $\tilde{b}_k$ can be computed by the Fast Fourier Transform (FFT), which, in fact, Gauss invented for this purpose.

Suppose one wishes to evaluate the interpolant $p_n(t)$ at certain points t. One good algorithm is to compute the discrete Fourier coefficients and then apply them. Alternatively, another good approach is to perform interpolation directly by means

[1] Zygmund calls them *Fourier–Lagrange coefficients* [28].

of the *barycentric formula* for trigonometric interpolation, introduced by Salzer [24] and later simplified by Henrici [15]:

$$p_n(t) = \sum_{k=0}^{N-1} (-1)^k f_k \csc(\frac{t-t_k}{2}) \Bigg/ \sum_{k=0}^{N-1} (-1)^k \csc(\frac{t-t_k}{2}) \quad (N \text{ odd}). \tag{3.8}$$

(If t happens to be exactly equal to a grid point t_k, one takes $p_n(t) = f_k$.) The work involved in this formula is just $O(N)$ operations per evaluation.

In the above discussion, we have assumed that the number of interpolation points, N, is odd. However, trigonometric interpolation, unlike trigonometric projection, makes sense for an even number of degrees of freedom as well as an odd number (see e.g. [13, 17, 28]); it would be surprising indeed if FFT codes refused to accept input vectors of even lengths! Suppose $n \geq 1$ is given and we wish to interpolate f in $N = 2n$ trigonometric points (3.1) rather than $N = 2n + 1$. This is one data value less than usual for a trigonometric polynomial of this degree, and we can lower the number of degrees of freedom in (3.2) by imposing the condition

$$\tilde{c}_{-n} = \tilde{c}_n \tag{3.9}$$

or equivalently in (3.3) by imposing the condition

$$\tilde{b}_n = 0. \tag{3.10}$$

This amounts to prescribing that the trigonometric interpolant through sawtoothed data of the form $f_k = (-1)^k$ should be $\cos(nt)$ rather than some other function such as $\exp(int)$—the only choice that ensures that real data will lead to a real interpolant. An equivalent prescription is that an arbitrary number N of data values, even or odd, will be interpolated by a linear combination of the first N terms of the sequence

$$1,\ \cos(t),\ \sin(t),\ \cos(2t),\ \sin(2t),\ \cos(3t),\ \ldots. \tag{3.11}$$

In this case of trigonometric interpolation with N even, the formulas (3.1)–(3.7) still hold, except that (3.4) and (3.6) must be multiplied by $1/2$ for $k = \pm n$. FFT codes, however, do not store the information that way. Instead, following (3.9), they compute $\tilde{a}_{-n}$ by (3.6) with $2/N$ instead of $1/N$ out front—thus effectively storing $\tilde{c}_{-n} + \tilde{c}_n$ in the place of $\tilde{c}_{-n}$—and then apply (3.2) with the $k = n$ term omitted. This gives the right result for values of t on the grid, but not at points in-between.

Note that the conditions (3.9)–(3.11) are very much tied to the use of the sample points (3.1). If the grid were translated uniformly, then different relationships between c_n and c_{-n} or a_n/b_n and a_{-n}/b_{-n} would be appropriate in (3.9)–(3.10) and different basis functions in (3.11), and if the grid were not uniform, then it would be hard to justify any particular choices at all for even N. For these reasons, even numbers of degrees of freedom make sense in equispaced interpolation but not in other trigonometric approximation contexts, in general. Henrici [15] provides a modification of the barycentric formula (3.8) for the equispaced case $N = 2n$,

$$p_n(t) = \sum_{k=0}^{N-1} (-1)^k f_k \cot(\frac{t-t_k}{2}) \Bigg/ \sum_{k=0}^{N-1} (-1)^k \cot(\frac{t-t_k}{2}) \quad (N \text{ even}). \tag{3.12}$$

4. Approximation theory and quadrature. The basic question of approximation theory is, will approximants to a function f converge as the degree is increased, and how fast? The formulas of the last two sections enable us to derive theorems addressing this question for trigonometric projection and interpolation. (For finer points of trigonometric approximation theory, see [19].) The smoother f is, the faster its Fourier coefficients decrease, and the faster the convergence of the approximants. (If f were merely continuous rather than Lipschitz continuous, then the trigonometric version of the Weierstrass approximation theorem [16, Section I.2] would ensure that it could be approximated arbitrarily closely by trigonometric polynomials, but not necessarily by projection or interpolation.)

Our first theorem asserts that Fourier coefficients decay algebraically if f has a finite number of derivatives, and geometrically if f is analytic. Here and in Theorem 4.2 below, the derivative $f^{(\nu)}$ is not assumed to be continuous; if it is not, the necessary integration by parts can be carried out in the setting of Stieltjes integrals [16, Section I.4]. Thus $|\sin(t)|$ on $[0, 2\pi]$, for example, corresponds to $\nu = 1$, and $|\sin(t)|^3$ to $\nu = 3$. The total variation V of a function on $[0, 2\pi]$ is defined periodically, so that if $f^{(\nu)}(t) = t$, for example, then V takes the value 4π, not 2π. All our theorems continue to assume that f is 2π-periodic.

THEOREM 4.1. *If f is $\nu \geq 0$ times differentiable and $f^{(\nu)}$ is of bounded variation V on $[0, 2\pi]$, then*

$$|c_k| \leq \frac{V}{2\pi |k|^{\nu+1}}. \tag{4.1}$$

If f is analytic with $|f(t)| \leq M$ in the open strip of half-width α around the real axis in the complex t-plane, then

$$|c_k| \leq M e^{-\alpha |k|}. \tag{4.2}$$

Proof. The bound (4.1) can be derived by integrating (2.2) by parts $\nu + 1$ times. Equation (4.2) can be derived by shifting the interval of integration $[0, 2\pi]$ of (2.2) downward in the complex plane for $k > 0$, or upward for $k < 0$, by a distance arbitrarily close to α; see [27, Section 3]. □

To apply Theorem 4.1 to trigonometric approximations, we note that the error in the degree n trigonometric projection (2.7) is

$$f(t) - f_n(t) = \sum_{|k|>n} c_k e^{ikt}, \tag{4.3}$$

a series that converges absolutely and uniformly by the Lipschitz continuity assumption on f. Similarly, (3.4) implies that the error in trigonometric interpolation is

$$f(t) - p_n(t) = \sum_{|k|>n} c_k (e^{ikt} - e^{ik't}), \tag{4.4}$$

where $k' = \mathrm{mod}(k + n, 2n + 1) - n$ is the index that k gets aliased to on the $(2n + 1)$-point grid, i.e., the integer of absolute value $\leq n$ congruent to k modulo $2n+1$. These formulas give us bounds on the error in trigonometric projection and interpolation.

THEOREM 4.2. *If f is $\nu \geq 1$ times differentiable and $f^{(\nu)}$ is of bounded variation V on $[0, 2\pi]$, then its degree n trigonometric projection and interpolant satisfy*

$$\|f - f_n\|_\infty \leq \frac{V}{\pi \nu n^\nu}, \qquad \|f - p_n\|_\infty \leq \frac{2V}{\pi \nu n^\nu}. \tag{4.5}$$

If f is analytic with $|f(t)| \le M$ in the open strip of half-width α around the real axis in the complex t-plane, they satisfy

$$\|f - f_n\|_\infty \le \frac{2Me^{-\alpha n}}{e^\alpha - 1}, \qquad \|f - p_n\|_\infty \le \frac{4Me^{-\alpha n}}{e^\alpha - 1}. \tag{4.6}$$

Proof. The estimates (4.5) follow by bounding the tails (4.3) and (4.4) with (4.1), and (4.6) likewise by bounding them with (4.2). □

A slight variant of this argument gives an estimate for quadrature. If I denotes the integral of a function f over $[0, 2\pi]$ and I_N its approximation by the N-point periodic trapezoidal rule, then from (2.2) and (3.6), we have $I = 2\pi c_0$ and $I_N = 2\pi \tilde{c}_0$. By (3.4) this implies

$$I_N - I = 2\pi \sum_{j \ne 0} c_{jN}, \tag{4.7}$$

which gives the following result.

THEOREM 4.3. *If f is $\nu \ge 1$ times differentiable and $f^{(\nu)}$ is of bounded variation V on $[0, 2\pi]$, then the N-point periodic trapezoidal rule approximation to its integral over $[0, 2\pi]$ satisfies*

$$|I_N - I| \le \frac{4V}{N^{\nu+1}}. \tag{4.8}$$

If f is analytic with $|f(t)| \le M$ in the open strip of half-width α around the real axis in the complex t-plane, it satisfies

$$|I_N - I| \le \frac{4\pi M}{e^{\alpha N} - 1}. \tag{4.9}$$

Proof. These results follow by bounding (4.7) with (4.1) and (4.2) as in the proof of Theorem 4.2. From (4.1), the bound one gets is $2V\zeta(\nu+1)/N^{\nu+1}$, where ζ is the Riemann zeta function, which we have simplified by the inequality $\zeta(\nu+1) \le \zeta(2) < 2$ for $\nu \ge 1$. The estimate (4.9) originates with Davis [9]; see also [17, 27]. □

Finally, in a section labeled "Approximation theory" we must mention another famous candidate for periodic function approximation: best approximation in the ∞-norm. Here the trigonometric version of the Chebyshev alternation theorem holds, assuming f is real. This result is illustrated below in Figure 6.5.

THEOREM 4.4. *Let f be real and continuous (not necessarily Lipschitz continuous) on the periodic interval $[0, 2\pi]$. For each degree $n \ge 0$, f has a unique best approximant $p_n^* \in P_n$ with respect to the norm $\|\cdot\|_\infty$, and p_n^* is characterized by the property that the error curve $(f - p_n^*)(t)$ equioscillates on $[0, 2\pi]$ between at least $2n+2$ equal extrema $\pm\|f - p_n^*\|_\infty$ of alternating signs.*

Proof. See [19, Section 5.2]. □

5. Trigfun computations. Building on the mathematics of the past three sections, Chebfun was extended in 2014 to incorporate trigonometric representations of periodic functions alongside its traditional Chebyshev representations of nonperiodic functions. For years the idea of a "Fourfun" analogue of Chebfun (with "Four" short for Fourier) had been in our minds, but had not seemed important enough to be a priority. Once we'd done it, we were surprised how widely useful this extension is. Here

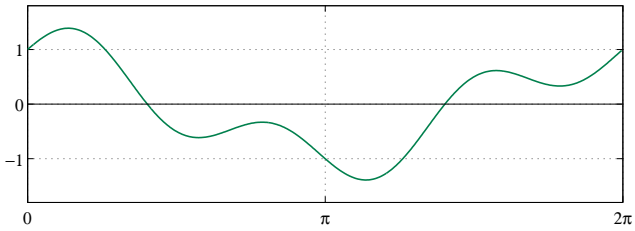

FIG. 5.1. *The trigfun representing* $f(t) = \cos(t) + \sin(3t)/2$ *on* $[0, 2\pi]$. *As always with Chebfun computations, one can evaluate* f *with* `f(t)`, *compute its definite integral with* `sum(f)` *or its maximum with* `max(f)`, *find its roots with* `roots(f)`, *and so on.*

and in the remainder of the paper, we assume the reader is familiar with Chebfun. (To get to know Chebfun, the best place to start is Chapter 1 of [11].)

After an initial period, we abandoned the prefix "four", which seemed confusing in names like "fourcoeffs" and "fourpts", and replaced it by the less ambiguous "trig". Our convention is that a *trigfun* is a representation via coefficients c_k as in (2.7) of a sufficiently smooth periodic function f on an interval by a trigonometric polynomial of adaptively determined degree, the aim always being accuracy of 15 or 16 digits relative to the ∞-norm of the function on the interval. This follows the same pattern as traditional Chebyshev-based chebfuns, which are representations of nonperiodic functions by polynomials, and a trigfun is not a distinct object from a chebfun but a particular type of chebfun. The default interval, as with ordinary chebfuns, is $[-1, 1]$, and other intervals are handled by the obvious linear transplantation.[2]

For example, we can construct and plot a trigfun for $\cos(t) + \sin(3t)/2$ on $[0, 2\pi]$ like this:

```
>> f = chebfun('cos(t) + sin(3*t)/2', [0 2*pi], 'trig')
>> plot(f)
```

The plot appears in Figure 5.1, and the following text output is produced, with the word "trig" signalling the periodic representation.

```
f =
   chebfun column (1 smooth piece)
       interval       length   endpoint values trig
[       0,     6.3]        7         1        1
Epslevel = 1.114301e-15.  Vscale = 1.388433e+00.
```

(The number `Vscale` is a measure of the maximum size of f on the interval, and `Epslevel` is a rough indication of the relative accuracy of the representation. We shall not give details, as these estimates are under ongoing discussion and development.) We see that Chebfun has determined that this function f is of length $N = 7$. This

[2]Actually, one aspect of the transplantation is not obvious, an indirect consequence of the translation-invariance of trigonometric functions. The nonperiodic function $f(x) = x$ defined on $[-1, 1]$, for example, has Chebyshev coefficients $a_0 = 0$ and $a_1 = 1$, corresponding to the expansion $f(x) = 0T_0(x) + 1T_1(x)$. Any user will expect the transplanted function $g(x) = x - 1$ defined on $[0, 2]$ to have the same coefficients $a_0 = 0$ and $a_1 = 1$, corresponding to the transplanted expansion $g(x) = 0T_0(x-1) + 1T_1(x-1)$, and this is what Chebfun delivers. By contrast, consider the periodic function $f(t) = \cos t$ defined on $[-\pi, \pi]$ and its transplant $g(t) = \cos(t - \pi) = -\cos t$ on $[0, 2\pi]$. A user will expect the expansion coefficients of g to be not the same as those of f, but their negatives! This is because we expect to use the same basis functions $\exp(ikx)$ or $\cos(kx)$ and $\sin(kx)$ on any interval of length 2π, however translated. The trigonometric part of Chebfun is designed accordingly.

means that there are 7 degrees of freedom, i.e., f is a trigonometric polynomial of degree $n = 3$, whose coefficients we can display like this:

```
>> c = trigcoeffs(f)
c =
 -0.000000000000000 + 0.250000000000000i
  0.000000000000000 + 0.000000000000000i
  0.500000000000000 + 0.000000000000000i
 -0.000000000000000 + 0.000000000000000i
  0.500000000000000 - 0.000000000000000i
  0.000000000000000 - 0.000000000000000i
 -0.000000000000000 - 0.250000000000000i
```

The coefficients in cosine/sine form are also available:

```
>> [a,b] = trigcoeffs(f)
a =
  -0.000000000000000
   1.000000000000000
   0.000000000000000
  -0.000000000000000
b =
   0.000000000000000
   0.000000000000000
   0.500000000000000
```

The absence of imaginary parts on the order of machine epsilon in these outputs is enabled by a boolean that monitors whether a function f is real:

```
>> isreal(f)
ans = 1
```

Note that the Chebfun constructor does not analyze its input symbolically, but just evaluates the function at trigonometric points (3.1), and it is from these evaluations that the degree and the values of the coefficients have been determined, as well as the fact that the function is real. A trigfun constructed in the ordinary manner is always of odd length N, corresponding to a trigonometric polynomial of degree $n = (N-1)/2$, though it is possible to make even-length trigfuns by explicitly specifying N.

To construct a trigfun, Chebfun samples the function on grids of size $16, 32, 64, \ldots$ and tests the resulting discrete Fourier coefficients for convergence down to relative machine precision. (As with standard Chebfun, the engineering details are complicated and under ongoing development.) When convergence is achieved, the series is chopped at an appropriate point and the degree reduced accordingly.

Once a trigfun has been created, computations can be carried out in the usual Chebfun fashion via overloads of familiar MATLAB commands. For example,

```
>> sum(f.^2)
ans = 3.926990816987241
```

This number is computed by integrating the trigonometric representation of f^2, i.e., by returning the number $2\pi c_0$ corresponding to the periodic trapezoidal rule applied to f^2 as described around Theorem 4.3. The default 2-norm is the square root of this result,

```
>> norm(f)
ans = 1.981663648803005
```

Derivatives of functions are computed by the overloaded command `diff`. The zeros

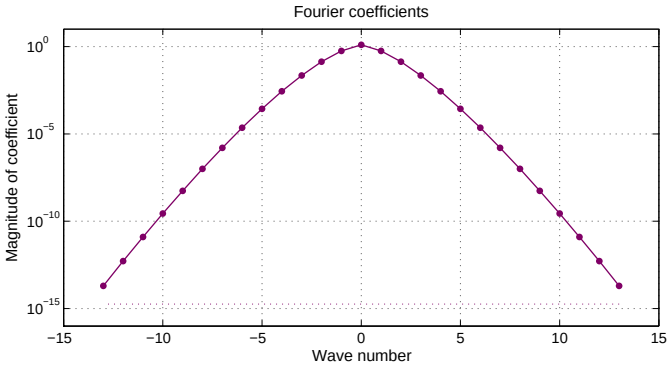

FIG. 5.2. *Absolute values of the Fourier coefficients of the trigfun for* $\exp(\sin t)$ *on* $[0, 2\pi]$. *This is an entire function (analytic throughout the complex* t*-plane), and in accordance with Theorem* 4.1, *the coefficients decrease faster than geometrically.*

of f are found with `roots`:

```
>> roots(f)
ans =
   1.263651122898791
   4.405243776488583
```

and Chebfun determines maxima and minima by first computing the derivative, then checking all of its roots:

```
>> max(f)
ans = 1.389383416980387
```

Concerning the algorithm used for periodic rootfinding, one approach would be to solve a companion matrix eigenvalue problem. However, this process requires $O(n^3)$ operations when carried out in the obvious manner, and we have not attempted to implement alternative linear algebra methods that might reduce the work to $O(n^2)$ [1]. Instead, trigfun rootfinding is done by first converting the problem to nonperiodic Chebfun form, whereupon we take advantage of Chebfun's $O(n^2)$ recursive interval subdivision strategy [5]. This shifting to subintervals for rootfinding is an example of an operation that breaks periodicity as mentioned in item 7 of the introduction.

The main purpose of the periodic part of Chebfun is to enable machine precision computation with periodic functions that are not exactly trigonometric polynomials. For example, $\exp(\sin t)$ on $[0, 2\pi]$ is represented by a trigfun of length 27, i.e., a trigonometric polynomial of degree 13:

```
g = chebfun('exp(sin(t))', [0 2*pi], 'trig')
g =
   chebfun column (1 smooth piece)
       interval       length   endpoint values trig
[       0,     6.3]       27         1        1
Epslevel = 6.672618e-16.  Vscale = 2.713687e+00.
```

The coefficients can be plotted on a log scale with the command `plotcoeffs(f)`, and the result shown in Figure 5.2 reveals the faster-than-geometric decay we expect for an entire function.

Figure 5.3 shows trigfuns and coefficient plots for the functions $f(t) = \tanh(5\cos(5t))$

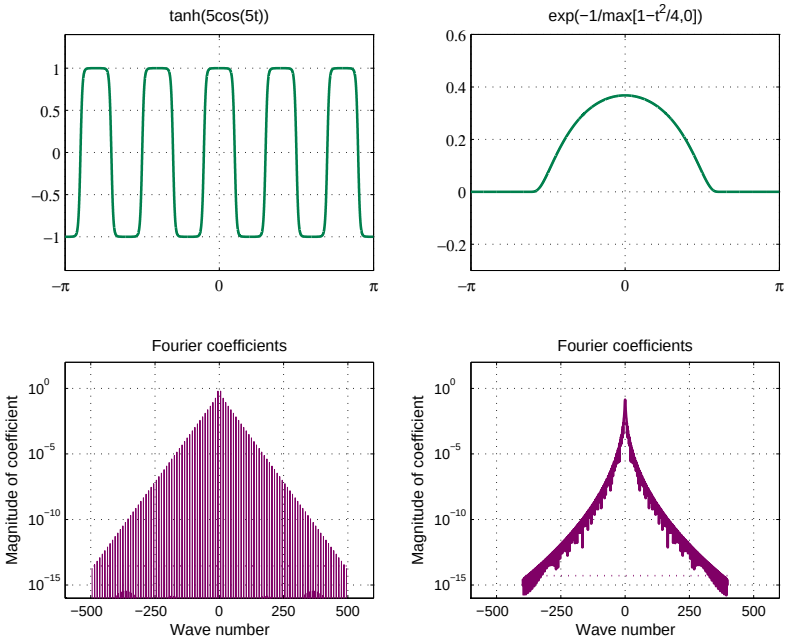

FIG. 5.3. *Trigfuns of* $\tanh(5\sin t)$ *and* $\exp(-100(t+.3)^2)$ *(upper row) and corresponding absolute values of Fourier coefficients (lower row).*

and $g(t) = \exp(-1/\max\{0, 1-t^2/4\})$ on $[-\pi, \pi]$. The latter is C^∞ but not analytic. Figure 5.4 shows a further pair of examples that we call an "AM" and an "FM signal". These are among the preloaded functions available with `cheb.gallerytrig`, the trigonometric analogue of `cheb.gallery`.

Computation with trigfuns, as with nonperiodic chebfuns, is carried out by a continuous analogue of floating point arithmetic [25]. To illustrate the "rounding" process involved, here are the lengths of the trigfuns f and g just mentioned:

```
>> length(f)
ans = 991
>> length(g)
ans = 797
```

Thus these functions are represented by trigonometric polynomials of degrees 495 and 398, respectively. Mathematically, their product is of degree 893. Numerically, however, Chebfun achieves 16-digit accuracy with degree 485:

```
>> length(f.*g)
ans = 971
```

Here is a more complicated example of Chebfun rounding adapted from [25], where it is computed with nonperiodic representations.

```
f = chebfun(@(x) sin(pi*t), 'trig')
s = f
for j = 1:15
  f = (3/4)*(1 - 2*f.^4)
  s = s + f
end
```

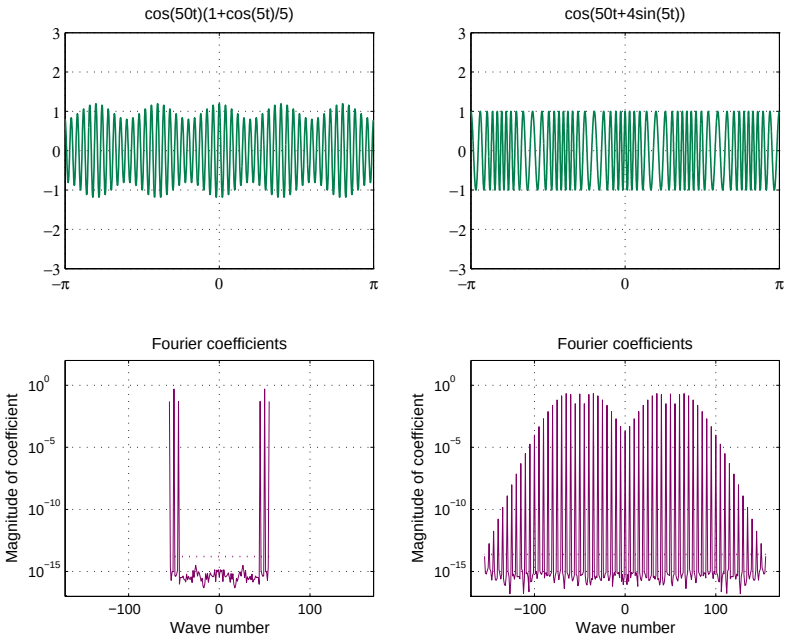

FIG. 5.4. *Trigfuns of the "AM signal"* $\cos(50t)(1+\cos(5t)/5)$ *and the "FM signal"* $\cos(50t+4\sin(5t))$ *(upper row) and corresponding absolute values of Fourier coefficients (lower row).*

This program takes 15 steps of an iteration that in principle quadruples the degree at each step, giving a function s at the end of degree $4^{15} = 1{,}073{,}741{,}824$. In actuality, however, because of the rounding to 16 digits, the degree comes out one million times smaller as 1082. This function is plotted in Figure 5.5. Following [25], we can compute the roots of $s - 8$ in half a second on a desktop machine:

```
>> roots(s-8)
ans =
  -0.992932107411882
  -0.816249934290176
  -0.798886729723434
  -0.201113270276573
  -0.183750065709826
  -0.007067892588105
   0.346696120418294
   0.401617073482085
   0.442269489632477
   0.557730510367531
   0.598382926517925
   0.653303879581729
```

The integral reported by `sum(s)` is 15.265483825826710, correct except in the last two digits.

If one tries to construct a trigfun by sampling a function that is not smoothly periodic, Chebfun will by default go up to length 2^{16} and then issue a warning:

```
>> h = chebfun('exp(t)', [0 2*pi], 'trig')
Warning: Function not resolved using 65536 pts.
```

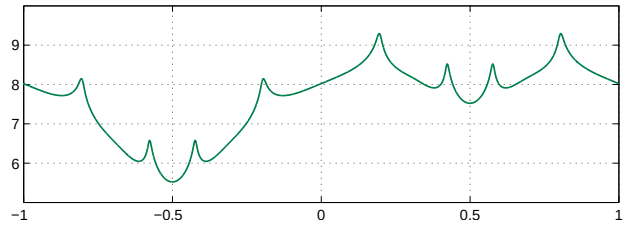

FIG. 5.5. *After fifteen steps of an iteration, this periodic function has degree* 1082 *in its Chebfun representation rather than the mathematically exact figure* 1,073,741,824.

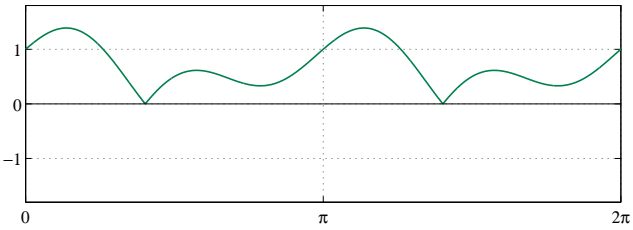

FIG. 5.6. *When the absolute value of the trigfun* f *of Figure* 5.1 *is computed, the result is a nonperiodic chebfun with three smooth pieces.*

```
          Have you tried a non-trig representation?
h =
   chebfun column (1 smooth piece)
       interval       length   endpoint values trig
[       0,     6.3]    65536   2.7e+02  2.7e+02
Epslevel = 3.631532e-12.  Vscale = 5.354917e+02.
```

On the other hand, computations that are known to break periodicity or smoothness will result in the representation being cast from a trigfun to a chebfun. For example, here we define g to be the absolute value of the function $f(t) = \cos(t) + \sin(3t)/2$ of Figure 5.1. The system detects that f has zeros, implying that g will probably not be smooth, and accordingly constructs it not as a trigfun but as an ordinary chebfun with several pieces:

```
>> f = chebfun('cos(t) + sin(3*t)/2', [0 2*pi], 'trig');
>> g = abs(f)
g =
   chebfun column (3 smooth pieces)
       interval       length   endpoint values
[       0,     1.3]       17          1  4.2e-16
[     1.3,     4.4]       25   1.8e-15  7.8e-16
[     4.4,     6.3]       20          0        1
Epslevel = 1.110223e-15.  Vscale = 1.389328e+00.  Total length = 62.
```

Similarly, if you add or multiply a trigfun and a chebfun, the result is a chebfun.

6. Applications. Analysis of periodic functions and signals is one of the oldest topics of mathematics and engineering. Here we give six examples of how a system for automating such computations may be useful.

Complex contour integrals. Smooth periodic integrals arise ubiquitously in complex analysis. For example, suppose we wish to determine the number of zeros of

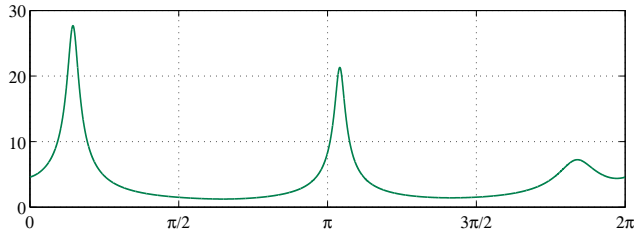

FIG. 6.1. *Resolvent norm* $\|(zI - A)^{-1}\|$ *for a* 4×4 *matrix* A *with* $z = e^{it}$ *on the unit circle.*

$f(z) = \cos(z) - z$ in the complex unit disk. The answer is given by

$$m = \frac{1}{2\pi i} \int \frac{f'(z)}{f(z)} dz = \frac{1}{2\pi i} \int \frac{1}{f(z)} \frac{df}{dt} dt \tag{6.1}$$

if $z = \exp(it)$ with $t \in [0, 2\pi]$. With periodic Chebfun, we can compute m by

```
>> z = chebfun('exp(1i*t)', [0 2*pi], 'trig');
>> f = cos(z) - z;
>> m = real(sum(diff(f)./f)/(2i*pi))
m =
  0.999999999999999
```

Changing the integrand from $f'(z)/f(z)$ to $zf'(z)/f(z)$ gives the location of the zero,

```
>> z0 = real(sum(z.*diff(f)./f)/(2i*pi))
z0 =
  0.739085133215159
```

(The exact answer ends in digits 161 rather than 159. The `real` commands are included to remove imaginary rounding errors.) For wide-ranging extensions of calculations like these, including applications to matrix eigenvalue problems, see [2].

Linear algebra. Chebfun does not work from explicit formulas: to construct a function, it is only necessary to be able to evaluate it. This is an extremely useful feature for linear algebra calculations. For example, the matrix

$$A = \frac{1}{3} \begin{pmatrix} 2 & -2i & 1 & 1 \\ 2i & -2 & 0 & 2 \\ -2 & 0 & 1 & 2 \\ 0 & i & 0 & 2 \end{pmatrix} \tag{6.2}$$

has all its eigenvalues in the unit disk. A question with the flavor of control and stability theory is, what is the maximum resolvent norm $\|(zI - A)^{-1}\|$ for z on the unit circle? We can calculate the answer with the code below, which constructs a periodic chebfun of degree $n = 477$. The maximum is 27.68851, attained with $z = \exp(0.454596i)$.

```
A = [2 -2i 1 1; 2i -2 0 2; -2 0 1 2; 0 1i 0 2]/3
I = eye(4);
ff = @(t) 1/min(svd(exp(1i*t)*I-A));
f = chebfun(ff, [0 2*pi], 'trig', 'vectorize');
[maxval,maxpos] = max(f)
```

Circular convolution and smoothing. The circular or periodic convolution of two

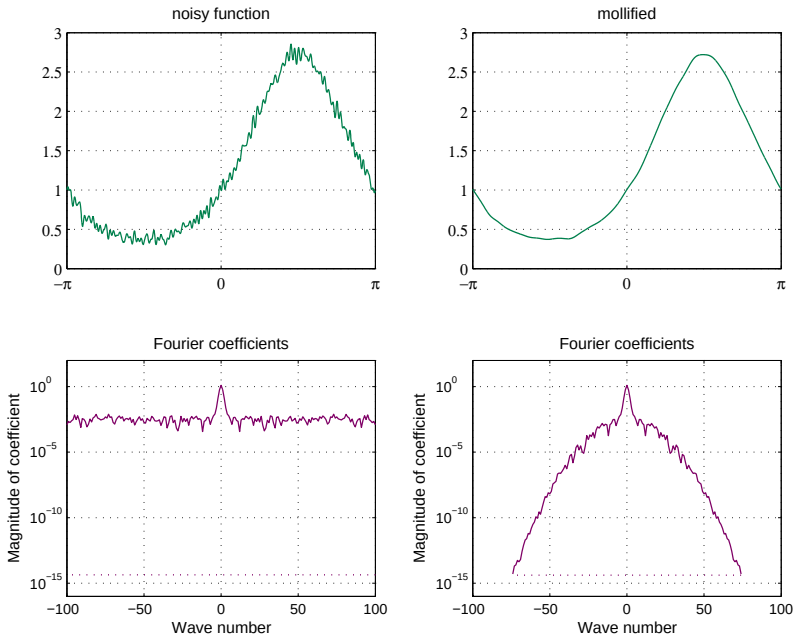

FIG. 6.2. *Circular convolution of a noisy function with a smooth mollifier.*

functions f and g with period T is defined by

$$(f * g)(t) := \int_{t_0}^{t_0+T} g(s) f(t-s) ds, \tag{6.3}$$

where t_0 is aribtrary. Circular convolutions can be computed for trigfuns with the `circconv` function, whose algorithm consists of coefficientwise multiplication in Fourier space. For example, here is a trigonometric interpolant through 201 samples of a smooth function plus noise, shown in the upper-left panel of Figure 6.2.

```
N = 201
tt = trigpts(N, [-pi pi])
ff = exp(sin(tt)) + 0.05*randn(N,1)
f = chebfun(ff, [-pi pi], 'trig')
```

The high wave numbers can be smoothed by convolving f with a mollifier. Here we use a Gaussian of standard deviation $\sigma = 0.1$ (numerically periodic for $\sigma \leq 0.35$). The result is shown in the upper-right panel of the figure.

```
gaussian = @(t,sigma) 1/(sigma*sqrt(2*pi))*exp(-0.5*(t/sigma).^2)
g = @(sigma) chebfun(@(t) gaussian(t,sigma), [-pi pi], 'trig')
h = circconv(f, g(0.1))
```

Fourier coefficients of non-smooth functions. A function f that is not smoothly periodic will at best have a very slowly converging trigonometric series, but still, one may be interested in its Fourier coefficients. These can be computed by applying `trigcoeffs` to a chebfun representation of f and specifying how many coefficients are required; the integrals (2.2) are then evaluated numerically by Chebfun's standard method of Clenshaw–Curtis quadrature. For example, Figure 6.3 shows a portrayal of

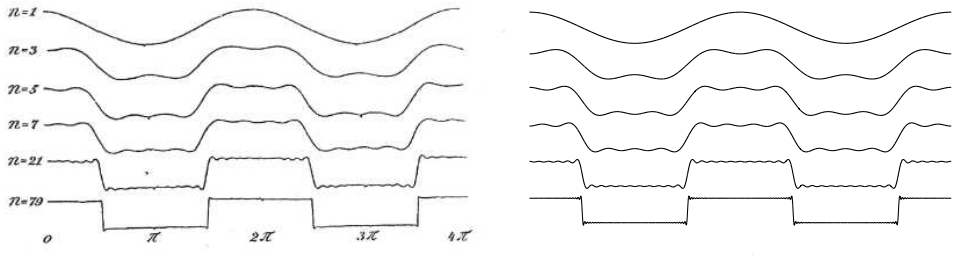

FIG. 6.3. *On the left, a figure from Runge's* 1904 *book* Theorie und Praxis der Reihen [23]. *On the right, the equivalent computed with periodic Chebfun. Among other things, this figure illustrates that a trigfun can be accurately evaluated outside its interval of definition.*

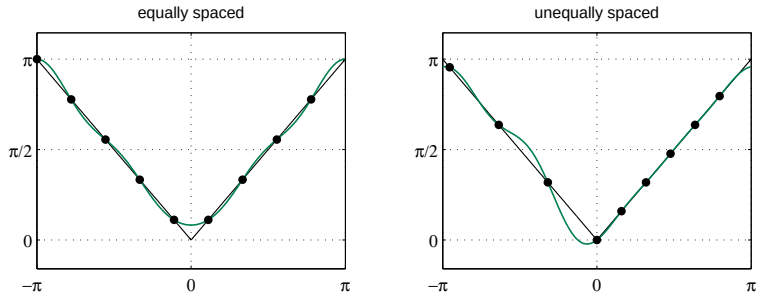

FIG. 6.4. *Trigonometric interpolation of* $|t|$ *in unequally spaced points with the generalized barycentric formula implemented in* `chebfun/interp1`.

the Gibbs phenomenon from Runge's 1904 book together with its Chebfun equivalent computed in a few seconds with the commands

```
t = chebfun('t', [-pi pi])
f = (abs(t) < pi/2)
for N = 2*[1 3 5 7 21 79] + 1
  c = trigcoeffs(f, N)
  fN = chebfun(c, [-pi pi], 'coeffs', 'trig')
  plot(fN, 'interval', [0 4*pi])
end
```

Interpolation in unequally spaced points. Very little attention has been given to trigonometric interpolation in unequally spaced points, but the barycentric formulas (3.8) and (3.12) have been generalized to this case by Salzer and Berrut [4]. Chebfun makes these formulas available through the command `chebfun/interp1`, just as has long been true for interpolation by algebraic polynomials. For example, the code

```
t = [-3 -2 -1 0 .5 1 1.5 2 2.5]
p = chebfun.interp1(t, abs(t), 'trig', [-pi pi])
```

interpolates the function $|t|$ on $[-\pi, \pi]$ in the 9 points indicated by a trigonometric polynomial of degree $n = 4$. The interpolant is shown in Figure 6.4 together with the analogous curve for equispaced points.

Best approximation, CF approximation, and rational functions. Chebfun has long had a dual role: it is a tool for computing with functions, and also a tool for demonstrating principles of approximation theory, including advanced ones. The

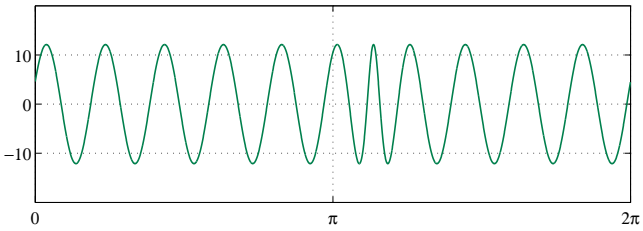

FIG. 6.5. *Error curve in degree* $n = 10$ *best trigonometric approximation to* $f(t) = 1/(1.01 - \sin(t-2))$ *over* $[0, 2\pi]$. *The curve equioscillates between* $2n + 2 = 22$ *alternating extrema.*

trigonometric side of Chebfun extends this second aspect to periodic problems. For example, Chebfun's `remez` command can compute best trigonometric approximants with equioscillating error curves as described in Theorem 4.3. Here is an example that generates the error curve displayed in Figure 6.5, with error 12.1095909.

```
f = chebfun('1./(1.01-sin(t-2))', [0 2*pi], 'trig')
p = remez(f,10)
plot(f-p)
```

Chebfun is also acquiring other capabilities for trigonometric polynomial and rational approximation, including Carathéodory–Fejér (CF) near-best approximation via singular values of Hankel matrices, and these will be described elsewhere.

7. Periodic ODEs, operator exponentials, and eigenvalue problems. One of the most important features of Chebfun is its ability to solve linear and nonlinear ordinary differential equations (ODEs), as well as integral equations, by applying the backslash command to a "chebop" object. We have extended these capabilities to periodic problems, both scalars and systems. See [12] for the theory of existence and uniqueness of solutions to periodic ODEs, which goes back to Floquet in the 1880s, a key point being the avoidance of nongeneric configurations corresponding to eigenmodes.

Chebfun's algorithm for linear ODEs amounts to an automatic spectral collocation method wrapped up so that, as always, the user need not be aware of the discretization. With standard Chebfun, these are Chebyshev spectral methods, and now with the periodic extension, they are Fourier spectral methods [8]. The problem is solved on grids of size 32, 64, and so on until the system judges that the Fourier coefficients have converged down to the level of noise, and the series is then truncated at an appropriate point.

For example, consider the problem

$$0.001(u'' + u') - \cos(t)u = 1, \qquad 0 \le t \le 6\pi \tag{7.1}$$

with periodic boundary conditions. The following Chebfun code produces the solution plotted in Figure 7.1 in half a second on a laptop. Note that the trigonometric discretizations are invoked by the flag `L.bc = 'periodic'`.

```
L = chebop(0,6*pi)
L.op = @(x,u) 0.001*diff(u,2) + 0.001*diff(u) - cos(x).*u
L.bc = 'periodic'
u = L\1
```

This trigfun is of degree 168, and the residual reported by `norm(L*u-1)` is 7×10^{-12}. As always, u is a chebfun; its maximum, for example, is `max(u)` $= 66.928$.

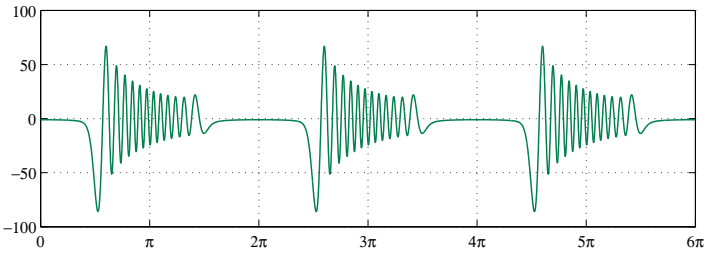

FIG. 7.1. *Solution of the linear periodic ODE* (7.1) *as a trigfun of degree* 168*, computed by an automatic Fourier spectral method.*

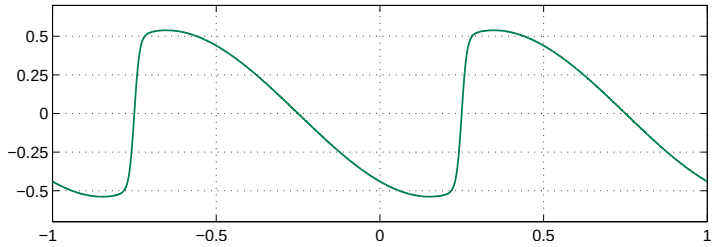

FIG. 7.2. *Solution of the nonlinear periodic ODE* (7.2) *computed by iterating the Fourier spectral method within a continuous form of Newton iteration. Executing* `max(diff(u))` *shows that the maximum of* u' *is* 32.094*.*

For periodic nonlinear ODEs, Chebfun applies trigonometric analogues of the algorithms developed by Driscoll and Birkisson in the Chebshev case [6, 7]. The whole solution is carried out by a Newton or damped Newton iteration formulated in a continuous mode ("solve then discretize" rather than "discretize then solve"), with Jacobian matrices replaced by Fréchet derivative operators implemented by means of automatic differentiation and automatic spectral discretization. For example, suppose we seek a solution of the nonlinear problem

$$0.004u'' + uu' - u = \cos(2\pi t), \qquad t \in [-1, 1] \tag{7.2}$$

with periodic boundary conditions. After seven Newton steps, the Chebfun commands below produce the result shown in Figure 7.2, of degree $n = 312$, and the residual norm `norm(N(u)-rhs,'inf')` is reported as 8×10^{-9}.

```
N = chebop(-1,1)
N.op = @(x,u) .004*diff(u,2) + u.*diff(u) - u
N.bc = 'periodic'
rhs = chebfun('cos(2*pi*t)', 'trig')
u = N\rhs
```

Chebfun's overload of the MATLAB `eigs` command solves linear ODE eigenvalue problems by, once again, automated spectral collocation discretizations [10]. This too has been extended to periodic problms, with Fourier discretizations replacing Chebyshev. For example, a famous periodic eigenvalue problem is the Mathieu equation

$$-u'' + 2q\cos(2t)u = \lambda u, \qquad t \in [0, 2\pi], \tag{7.3}$$

where q is a real parameter. The Chebfun commands below give the plot shown in Figure 7.3.

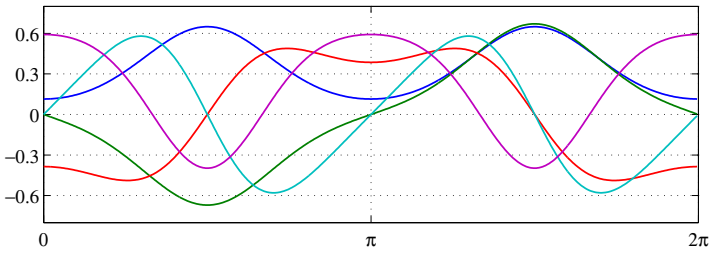

FIG. 7.3. *First five eigenfunctions of the Mathieu equation* (7.3) *with* $q = 2$*, computed with* `eigs`.

```
q = 2
L = chebop(@(x,u) -diff(u,2)+2*q*cos(2*x).*u, [0 2*pi], 'periodic')
[V,D] = eigs(L,5)
plot(V)
```

So far as we are aware, Chebfun is the only system that offers this kind of convenient solution of ODEs and related problems, now in the periodic as well as nonperiodic case.

We have also implemented a periodic analogue of Chebfun's `expm` command for computing exponentials of linear operators, respectively, which we omit discussing here for reasons of space. All the capabilities mentioned in this section can be explored with Chebgui, the graphical user interface written by Birkisson, which now invokes trigonometric spectral discretizations whenever periodic boundary conditions are specified.

8. Discussion. Chebfun is an open-source project written in MATLAB and hosted on GitHub; details and the user's guide can be found at `www.chebfun.org` [11]. About thirty people have contributed to its development over the years, and at present there are about ten developers based mainly at the University of Oxford. During 2013–2014 the code was redesigned and rewritten as version 5 (first released June 2014) in the form of about 100,000 lines of code realizing about 40 classes. The aim of this redesign was to enhance Chebfun's modularity, clarity, and extensibility, and the introduction of periodic capabilities, which had not been planned in advance, was the first big test of this extensibility. We were pleased to find that the modifications proceeded smoothly. The central new feature is a new class `@trigtech` in parallel to the existing `@chebtech1` and `@chebtech2`, which work with polynomial interpolants in first- and second-kind Chebyshev points, respectively.

About half the classes of Chebfun are concerned with representing functions, and the remainder are mostly concerned with ODE discretization and automatic differentiation for solution of nonlinear problems, whether scalar or systems, possibly with nontrivial block structure. The incorporation of periodic problems into this second, more advanced part of Chebfun was achieved by introducing a new class `@trigcolloc` matching `@chebcolloc1` and `@chebcolloc2`.

About a dozen software projects in various computer languages have been modeled on Chebfun, and a partial list can be found at `www.chebfun.org`. One of these, Fourfun, is a MATLAB system for periodic functions developed independently of the present work by Kristyn McLeod, a student of former Chebfun developer Rodrigo Platte [18]. Another that also has periodic and differential equations capabilities is ApproxFun, written in Julia by Sheehan Olver and former Chebfun developer Alex

Townsend [20].[3] We regard the business of numerical computing with functions as an enterprise destined to continue to grow, and we cannot predict what systems or languages may be dominant, say, twenty years from now. For the moment, only Chebfun offers the breadth of capabilities entailed in the vision of MATLAB-like functionality for continuous functions and operators in analogy to the long-familiar methods for discrete vectors and matrices.

In this article we have not discussed Chebfun computations with two-dimensional periodic functions, which are under development. For example, we are investigating capabilities for solution of time-dependent PDEs on a periodic spatial domain and for PDEs in two space dimensions, one or both of which are periodic. A particularly interesting prospect is to apply such representations to computation with functions on disks and spheres.

For computing with vectors and matrices, although MATLAB codes are rarely the fastest in execution, their convenience makes them nevertheless the best tool for many applications. We believe that Chebfun, including now its extension to periodic problems, plays the same role for numerical computing with functions.

Acknowledgements. This work was carried out in collaboration with the rest of the Chebfun team, whose names are listed at `www.chebfun.org`. Particularly active in this phase of the project have been Anthony Austin, Ásgeir Birkisson, Toby Driscoll, Nick Hale, Hrothgar, Alex Townsend, and Kuan Xu. We are grateful to all of these people for their suggestions in preparing this paper. The first author would like to thank the Oxford University Mathematical Institute, and in particular the Numerical Analysis Group, for hosting and supporting his sabbatical visit in 2014, during which this research was initiated.

REFERENCES

[1] J. L. Aurentz, T. Mach, R. Vandebril, and D. S. Watkins, Fast and backward stable computation of roots of polynomials, manuscript, 2014, available as preprint at `http://www.cs.kuleuven.be/publicaties/rapporten/tw/TW654.abs.html`.

[2] A. P. Austin, P. Kravanja and L. N. Trefethen, Numerical algorithms based on analytic function values at roots of unity, *SIAM J. Numer. Anal.* 52 (2014), 1795–1821.

[3] Z. Battles and L. N. Trefethen, An extension of MATLAB to continuous functions and operators, *SIAM J. Sci. Comp.* 25 (2004), 1743–1770.

[4] J.-P. Berrut, Baryzentrische Formeln zur trigonometrischen Interpolation (I), *J. Appl. Math. Phys.* 35 (1984), 91–105.

[5] J.-P. Berrut and L. N. Trefethen, Barycentric Lagrange interpolation, *SIAM Rev.* 46 (2004), 501–517.

[6] A. Birkisson and T. A. Driscoll, Automatic Fréchet differentiation for the numerical solution of boundary-value problems, *ACM Trans. Math. Softw.* 38 (2012), 1–26.

[7] A. Birkisson and T. A. Driscoll, Automatic linearity dection, preprint, `eprints.maths.ox.ac.uk`, 2013.

[8] J. P. Boyd, *Chebyshev and Fourier Spectral Methods,* 2nd ed., Dover, 2001.

[9] P. J. Davis, On the numerical integration of periodic analytic functions, in R. E. Langer, ed., *On Numerical Integration,* Math. Res. Ctr., U. of Wisconsin, 1959, pp. 45–59.

[10] T. A. Driscoll, Automatic spectral collocation for integral, integro-differential, and integrally reformulated differential equations, *J. Comput. Phys.* 229 (2010), 5980–5998.

[11] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide,* Pafnuty Publications, Oxford, 2014. Most recent version freely available at `www.chebfun.org`.

[12] M. S. P. Eastham, *The Spectral Theory of Periodic Differential Equations,* Scottish Academic Press, Edinburgh, 1973.

[3] Platte created Chebfun's edge detection algorithm for fast splitting of intervals. Townsend extended Chebfun to two dimensions.

[13] G. Faber, Über steige Funktionen, *Math. Ann.* 69 (1910), 372–443.
[14] C. F. Gauss, Theoria interpolationis methodo nova tractata, *Werke,* v. 3, Königl. Ges. Gött., 1866, pp. 265–327.
[15] P. Henrici, Barycentric formulas for interpolating trigonometric polynomials and their conjugates, *Numer. Math.,* 33 (1979), 225–234.
[16] Y. Katznelson, *An Introduction to Harmonic Analysis,* Dover, 1968.
[17] R. Kress, Ein ableitungsfreies Restglied für die trigonometrische Interpolation periodischer analytischer Funktionen, *Numer. Math.* 16 (1971), 389–396.
[18] K. McLeod, Fourfun: A new system for automatic computations using Fourier expansions, manuscript, 2014.
[19] G. Meinardus, *Approximation of Functions: Theory and Numerical Methods,* Springer, 1967.
[20] S. Olver and A. Townsend, A practical framework for infinite-dimensional linear algebra, arXiv:1409.5529, 2014.
[21] R. B. Platte, L. N. Trefethen, and A. B. J. Kuijlaars, Impossibility of fast stable approximation of analytic functions from equispaced samples, *SIAM Rev.* 53 (2011), 308–318.
[22] C. Runge, Über empirische Funktionen und die Interpolation zwischen äquisitanten Ordinaten, *Z. Math. Phys.* 46 (1901), 224–243.
[23] C. D. T. Runge, *Theorie und Praxis der Reihen,* Sammlung Schebert, 1904, reprinted by VKM Verlag, Saarbrücken, 2007.
[24] H. E. Salzer, Coefficients for facilitating trigonometric interpolation, *J. Math. Phys.* 27 (1948), 274–278.
[25] L. N. Trefethen, Numerical computation with functions instead of numbers, *Commun. ACM,* to appear.
[26] L. N. Trefethen, *Approximation Theory and Approximation Practice,* SIAM, 2013.
[27] L. N. Trefethen and J. A. C. Weideman, The exponentially convergent trapezoidal rule, *SIAM Rev.* 56 (2014), 385–458.
[28] A. Zygmund, *Trigonometric Series,* Cambridge U. Press, 1959.