J Eng Math (2006) 56:203–219
DOI 10.1007/s10665-006-9087-5

ORIGINAL PAPER

Computing the zeros, maxima and inflection points of Chebyshev, Legendre and Fourier series: solving transcendental equations by spectral interpolation and polynomial rootfinding

John P. Boyd

Received: 3 January 2006 / Accepted: 26 July 2006 /
Published online: 18 November 2006
© Springer Science+Business Media B.V. 2006

Abstract Recently, both companion-matrix methods and subdivision algorithms have been developed for finding the zeros of a truncated spectral series. Since the Chebyshev or Legendre coefficients of derivatives of a function $f(x)$ can be computed by trivial recurrences from those of the function itself, it follows that finding the maxima, minima and inflection points of a truncated Chebyshev or Fourier series $f_N(x)$ is also a problem of finding the zeros of a polynomial when written in truncated Chebyshev series form, or computing the roots of a trigonometric polynomial. Widely scattered results are reviewed and a few previously unpublished ideas sprinkled in. There are now robust zerofinders for all species of spectral series. A transcendental function $f(x)$ can be approximated arbitrarily well on a real interval by a truncated Chebyshev series $f_N(x)$ of sufficiently high degree N. It follows that through Chebyshev interpolation and Chebyshev rootfinders, it is now possible to easily find all the real roots on an interval for any smooth transcendental function.

Keywords Companion-matrix methods · Gegenbauer · Spectral series · Spherical harmonics · Trigonometric polynomial

1 Introduction

The book review [1] catalogues no fewer than 18 books on Chebyshev, Legendre and Fourier spectral methods. However, none of these books discusses how to analyze these spectral series for zeros, stationary points and inflection points. One can of course apply the two old standbys: (1) the MAGS (Make-A-Graph–Stupid) method, which is to plot $f_N(x)$ on a fine grid and observe where it crosses the axis and (2) Newton's iteration, which can refine MAGS-generated approximations to machine precision. For finding the roots of a polynomial in the usual "monomial" or "powers-of-x" form, however, neither is necessary: every software library has robust "black box" solvers that require nothing from the user except the coefficients of the polynomial. In recent years, equally effective "black box" methods have been developed for spectral series. The rest of this article will briefly explain them.

J. P. Boyd (✉)
Department of Atmospheric, Oceanic and Space Science, University of Michigan, 2455 Hayward Avenue, Ann Arbor, MI 48109-2143, USA
e-mail: jpboyd@umich.edu

There are two families of methods of general applicability: companion-matrix methods and subdivision algorithms. Frobenius showed more than a century ago that the roots of a polynomial are the eigenvalues of the "Frobenius companion matrix" whose elements are trivial functions of the coefficients of the polynomial. The default rootsolver in Matlab, for example, via the "roots" command, is a companion-matrix method. In the next section, we review companion-matrix methods for orthogonal polynomials and Fourier series.

The other family of algorithms subdivide the target interval, and compute the zeros as the roots of local Chebyshev approximations on each subdomain. These are harder to program, but faster for polynomials of high degree. Because the target function is a polynomial, one can derive rigorous error bounds that make the subdivision methods very robust.

Since spectral series converge much slower off the expansion interval, the emphasis will be on *real* roots *on* the *expansion interval*. Companion-matrix methods furnish *all* roots of the polynomial $f_N(x)$. However, roots in the complex plane where the spectral series is *diverging* will only be spurious approximations to zeros of the transcendental function $f(x)$ that the truncated spectral series is approximating. Even at points only slightly off the real axis where the spectral series is still converging, the *rate* of convergence of $f_N(x)$ to $f(x)$ will be much poorer than for real x. So, in the rest of this article, we shall discuss real roots on the interval only.

For many years, software libraries have contained reliable polynomial rootfinders that compute all roots without any user input except the coefficients of the polynomial. Unfortunately, it has not been possible to extend these techniques to computing zeros of transcendental functions $f(x)$, which must be computed one-at-a-time by Newton's method, assuming a sufficiently good first guess is available, or by bisection, which often misses roots. In Sect. 5, we explain that Chebyshev interpolation can be combined with the Chebyshev rootfinders to create a "blackbox" for computing all zeros of a smooth transcendental function on a real interval [2–4].

2 Companion-matrix methods

2.1 Chebyshev polynomials

Theorem 1 (Chebyshev companion matrix) *Let $f_N(x)$ denote a polynomial of degree N written in "Chebyshev form" as*

$$f_N(x) = \sum_{j=0}^{N} a_j\, T_j(x). \tag{1}$$

Then all roots of $f_N(x)$, both on and off the canonical expansion interval, $x \in [-1, 1]$, *are eigenvalues of the $N \times N$ matrix $\vec{\vec{A}}$ whose elements are*

$$A_{jk} = \begin{cases} \delta_{2,k}, & j = 1, \;\; k = 1, 2, \ldots, N \\ \frac{1}{2}\left\{\delta_{j,k+1} + \delta_{j,k-1}\right\}, & j = 2, \ldots, (N-1), \;\; k = 1, 2, \ldots, N, \\ (-1)\frac{a_{j-1}}{a_N} + (1/2)\delta_{k,N-1}, & j = N, \;\; k = 1, 2, \ldots, N, \end{cases} \tag{2}$$

where δ_{jk} is the usual Kronecker delta function such that $\delta_{jk} = 0$ if $j \neq k$ while $\delta_{jj} = 1$ for all j. ([13] *and earlier sources.*)

For a quintic polynomial, for example,

$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ (1/2) & 0 & (1/2) & 0 & 0 \\ 0 & (1/2) & 0 & (1/2) & 0 \\ 0 & 0 & (1/2) & 0 & (1/2) \\ (-1)\frac{a_0}{2a_5} & (-1)\frac{a_1}{2a_5} & (-1)\frac{a_2}{2a_5} & (-1)\frac{a_3}{2a_5}+(1/2) & (-1)\frac{a_4}{2a_5} \end{vmatrix}. \tag{3}$$

To make a practical algorithm, one must balance the matrix before computing its eigenvalues, as is done automatically in the Matlab "eig" command, and then sort the roots so as to discard all those that fail to lie within a small tolerance ϵ of the interval $x \in [-1,1]$. The reason for the tolerance is that the usual roundoff error may perturb the eigenvalues slightly off the interval, especially for multiple roots, which are notoriously ill-conditioned.

It is easy to polish the eigenvalues by one or two Newton's iterations:

$$x_{\text{root}}^{(m+1)} = x_{\text{root}}^{(m)} - \frac{f_N(x_{\text{root}}^{(m)})}{\mathrm{d}f/\mathrm{d}x(x_{\text{root}}^{(m)})} \tag{4}$$

where m is the iteration number. Note that it is trivial to compute the necessary derivative by recurrence as explained in Sect. 4.

An alternative is to convert the Chebyshev form to the usual powers-of-x form. This is easy, but the condition number of the transformation matrix grows as $0.04\,N^{2.4} = 0.04N^{1+\sqrt{2}}$, which limits this strategy to rather modest N [3]. The Chebyshev companion matrix given here is the better way.

Companion matrices for Legendre, Gegenbauer, Hermite and general orthogonal polynomials are given in Appendix A.

2.2 Trigonometric polynomials and Fourier series

At present, it is not known how to construct a matrix whose eigenvalues are the roots of a truncated Fourier series, $f_N(t)$, as given by (8) below. However, the transformation

$$z = \exp(\mathrm{i}t) \tag{5}$$

converts a *trigonometric* polynomial $f_N(t)$ with $2N+1$ terms into an *ordinary* polynomial $h(z)$ of degree $2N$ as independently discovered several times [3, 5, 6]. The "associated polynomial", $h(z[t]) \equiv \exp(\mathrm{i}Nt)f_N(t)$, is

$$h(z) = \frac{1}{2}\sum_{k=0}^{2N} h_k\, z^k \;\equiv\; z^N f(t[z]), \tag{6}$$

where

$$h_j = \begin{cases} a_{N-j} + \mathrm{i}\, b_{N-j}, & j = 0,1,\ldots,(N-1), \\ 2\,a_0, & j = N \\ a_{j-N} - \mathrm{i}\, b_{j-N}, & j = N+1, N+2, \ldots (2N). \end{cases} \tag{7}$$

From this transformation comes the following.

Theorem 2 (Fourier companion matrix) *Define the trigonometric polynomial*

$$f_N(t) \equiv \sum_{j=0}^{N} a_j \cos(jt) + \sum_{j=1}^{N} b_j \sin(jt). \tag{8}$$

The matrix elements B_{jk} of the Frobenius matrix for a trigonometric polynomial of general degree N (and therefore $(2N+1)$ terms) are

$$B_{jk} = \begin{cases} \delta_{j,k-1}, & j = 1,2,\ldots,(2N-1), \\ (-1)\frac{h_{k-1}}{a_N - ib_N}, & j = 2N, \end{cases} \tag{9}$$

where δ_{jk} is the usual Kronecker delta function such that $\delta_{jk} = 0$ if $j \neq k$, while $\delta_{jj} = 1$ for all j and k and the h_j are defined by (7).

The roots t_k of $f_N(t)$ are the negative of $\sqrt{-1}$ times the logarithm *of the matrix eigenvalues z_k:*

$$t_{k,m} \equiv arg(z_k) + 2\pi m - \mathrm{i}\log(|z_k|), \quad k = 1,2,\ldots,2N;\ m = \text{integer}. \tag{10}$$

In particular, the real-valued roots of $f_N(t)$ for real $t \in [-\pi,\pi]$ are the angles of the roots of $h(z)$ on the unit circle. Equivalently, each real-valued root t_k of $f(t)$ on $t \in (-\pi,\pi]$ is connected to a root z_k of the associated polynomial through $t_k = arg(z_k)\ \forall\, k$ such that $|z_k| = 1$ where $arg(z)$ is the usual complex argument function such that, for $z = |z|\exp(\mathrm{i}\theta)$, $arg(z) = \theta$. (Previously unpublished.)

For $N = 2$, the Fourier–Frobenius matrix is explicitly

$$\begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ (-1)\frac{a_2+ib_2}{a_2-ib_2} & (-1)\frac{a_1+ib_1}{a_2-ib_2} & (-1)\frac{2\,a_0}{a_2-ib_2} & (-1)\frac{a_1-ib_1}{a_2-ib_2} \end{vmatrix}. \tag{11}$$

The transformation $t = -\mathrm{i}\ \log(z)$ between the trigonometric polynomial in t and the ordinary polynomial in z does not amplify errors [5].

A useful corollary is that $f_N(t)$ has exactly $2N$ roots, when the roots are counted according to their multiplicity, in the strip of the complex plane such that $-\pi < \Re(t) \leq \pi$. (Note that roots with $\Re(t) = -\pi$ are excluded.)

2.3 Costs of the companion-matrix method

The companion matrix/QR method has a cost which grows as N^3 as illustrated in Fig. 1 for both Chebyshev series (left) and Fourier series (right). The Chebyshev method has a cost which is $O(12N^3)$; the Fourier cost is $O(500N^3)$. Part of the reason that the Fourier method is so much more costly for a given N is that a Fourier polynomial of degree N generates a companion matrix of size $2N \times 2N$, which increases the cost by a factor of eight over the Chebyshev case. The other difference is that, because the Fourier matrix elements are *complex-valued*, the matrix balancing/QR algorithm does a lot more complex arithmetic than in the Chebyshev case.

2.4 Errors in companion matrix methods

Toh and Trefethen have analyzed the numerical conditioning of the companion matrix [7]. They found that, if the matrix is "balanced" by a diagonal similarity transform in the sense of [8], then the condition number of the balanced companion matrix was only a little worse than that of the underlying polynomial zeros. We shall not discuss balancing further because this is automatic in `EISPACK` and in the Matlab "roots" command.

For the Fourier companion matrix, the transformation $z = \exp(\mathrm{i}t)$ amplifies errors in complex-valued roots of $f_N(t)$ because roots at a moderate distance $|t_{\mathrm{im}}|$ below the real axis are mapped into roots of large modulus $\exp(|t_{\mathrm{im}}|)$. However, the error in the approximation of a function $f(t)$ by the truncated Fourier

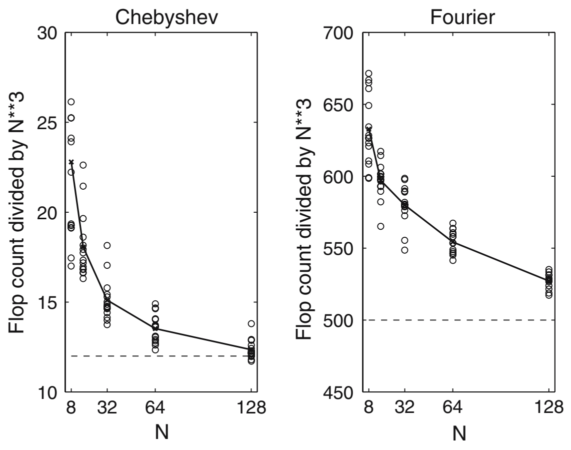

Fig. 1 Floating-point operation counts, divided by N^3, for an ensemble of polynomials with random coefficients and various N. Left panel: Chebyshev. Right panel: Fourier. (Note that a Fourier polynomial of degree N actually has $2N + 1$ terms, and has an associated polynomial of degree $2N$.) The amplitudes of the coefficients were multiplied by $1/(1.2)^j$ where j is the degree to mimic the exponential decay of Chebyshev coefficients of functions analytic on $x \in [-1, 1]$ and of Fourier coefficients of periodic, analytic functions. Circles: cost for a particular polynomial. The thick solid curve connects the average of an ensemble of 15 random polynomials for each polynomial degree N. The thin dashed line on the left is at 12; the right guideline is at 500. Computations were performed in Matlab 5.2, which by default "balances" the matrix before applying the QR algorithm to find the eigenvalues

series $f_N(t)$ grows as $\exp(N|t_{\text{im}}|)$, which is the Nth power of the growth in errors of the companion matrix. Thus, the exponential change-of-coordinate in the associated polynomial method does not cause difficulties, except for those roots, namely those of large negative t_{im}, which would be poor approximations to those of $f(t)$ even if the zeros of $f_N(t)$ were computed exactly.

Thus, the theoretical analysis asserts that the companion matrix/associated polynomial scheme is very well-conditioned for finding the *real* roots of $f_N(t)$ [and the function it approximates, $f(t)$] and, with less accuracy, roots of *small* imaginary part. Similar remarks apply to the Chebyshev companion matrix: highly accurate and well-conditioned for roots on the interval, much less accurate for roots far from the interval $x \in [-1, 1]$ which are likely to be rubbish.

To numerically support these arguments, we tested the algorithm on an ensemble of trigonometric polynomials with random coefficients. (Similar experiments for polynomials in Chebyshev form are given in [9].) For the first set of experiments, we employed what in signal processing would be called a "white noise" spectrum: the coefficients had the same range of values regardless of degree. The Fourier series of periodic, analytic functions fall exponentially fast with degree [10], however, so we also performed tests in which the random coefficients were multiplied by a factor that decays exponentially fast with degree, a "red noise" spectrum in the sense that most of the amplitude is in coefficients of low degree. We then evaluated $f(t)$ at each of the numerically generated roots and took the L_∞ norm of the vector of the residuals thus obtained.

Figure 2 shows that the errors grow with N — but not very fast. The "white noise" polynomials are the worst case: the dashed fitting curve shows that the errors grow proportional to N^2, roughly. However, the proportionality constant is machine epsilon, 2×10^{-16}, so the error is extremely small unless N is huge. Note that the roots from the companion matrix can always be "Newton–polished" by iterating once or twice with the Newton–Raphson iteration for each root to refine the zeros to near machine precision.

For the red-noise case, which is more typical of applications, the error grows only as $O(\sqrt{N})$. Though one must be cautious about extrapolating from these examples to the Fourier series that arise in applications, which have *non-random* coefficients, the numerical evidence is fully consistent with the theoretical analysis

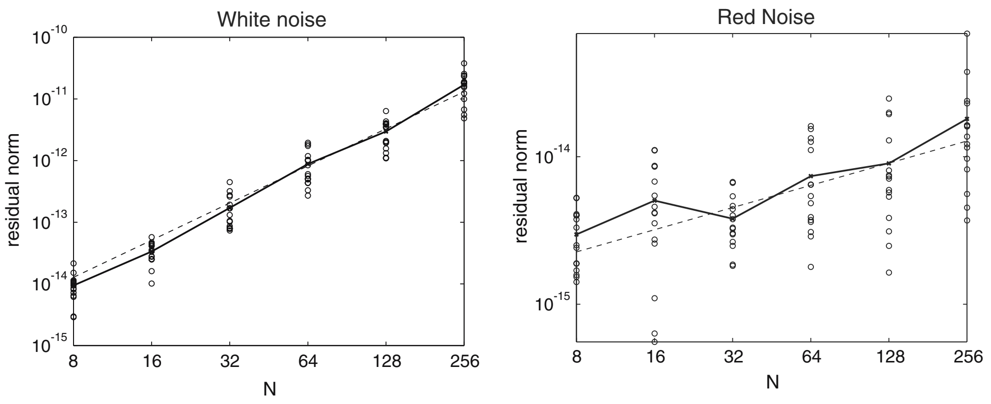

Fig. 2 Circles: maximum absolute value of the residual at the numerically-computed REAL roots for individual trigonometric polynomials with random coefficients. (The "residual" is just the value of $f(t)$ at the numerically computed root, which of course is zero if the root is exact.) The thick solid curve connects the average of an ensemble of fifteen random polynomials for each polynomial degree N. Left panel: a "white noise" set of coefficients. The thin dashed curve is a fitting line, $2 \times 10^{-16} N^2$. Right panel: Same as the left panel except that the random coefficients a_j, b_j were multiplied by $1/(1.2)^j$ to make the coefficients a "red noise" spectrum. The thin dashed curve is a fitting line, $8 \times 10^{-16}\sqrt{N}$

of the preceding section and strongly suggests that the associated polynomial method is very accurate and reliable for *real* roots of trigonometric polynomials.

Comprehensive tests of this sort are difficult because the Fourier coefficients in applications are not random, but rather have considerable structure. But what structure? Jónsson and Vavasis did not merely randomly vary the coefficients between ± 1, but randomly varied the magnitude of the range, too [11]. However, the magnitude is not random either, but rather decreases exponentially fast with superimposed oscillations. It is probably best to blend both random examples and structured examples drawn from applications, such as the Mathieu functions treated in [12]. It is heartening that for both random polynomials and applications [13], the companion matrix methods are remarkably stable and accurate.

One final worry is that for typical spectral series, the exponential decrease of the spectral coefficients with degree implies that the highest coefficient is very tiny. This is worrisome because this coefficient is the *divisor* of all the terms in the last row of the companion matrix. However, the numerical experiments of the next section and the extensive experiments for the related case of Chebyshev polynomial companion matrices [9], applied to trigonometric rootfinding in later sections, suggest that the smallness of the highest coefficient is rarely a problem. Perhaps the only exception is when $f_N(x)$ is of definite parity so that the highest coefficient is *identically zero*; the fix is to simply decrease N by one.

If the smallness of the highest coefficient is a problem, somehow, a remedy is merely to solve a generalized eigenvalue problem [11], but this remedy seems rarely needed here.

2.5 Orthogonal polynomials and Fourier series with parity symmetries

In applications, one often encounters functions which are symmetric with respect to $x = 0$, that is $f(x) = f(-x)$, or antisymmetric such that $f(x) = -f(x)$. The Chebyshev, Legendre, Hermite, Gegenbauer and Fourier series for such functions of "definite parity" then contain only half as many non-zero terms as a general series of the same degree. The cost can be reduced by a *factor of eight* by special methods that yield companion matrices or reduced problems with half the size of the general case. Boyd and Gally developed a parity-exploiting reduction for Chebyshev polynomials [9]. A similar treatment for orthogonal polynomials in general and for Fourier series is given in [12].

Because these methods for symmetric polynomials are somewhat specialized and have been published elsewhere, we omit them here. However, when applicable, the savings are always at least an order-of-magnitude in computer time.

3 Subdivision methods

In their downloadable "chebfun" library, Battles and Trefethen have extended Matlab from vectors and matrices to functions and operators by replacing functions by their approximations as Chebyshev interpolants — i.e., polynomials in Chebyshev form where the degree N may be as high as the thousands. This is "Chebyshevization" on a grand scale, and rootfinding-in-Chebyshev-form is an essential component [6].

Obviously, when N is in the thousands, a companion-matrix method, with its cost growing as N^3, is extremely expensive. The cheapness of subdivision is that, if the interval is divided into N_s parts, and a polynomial of degree $M \approx N/N_s$ is applied on each subdomain, then the total cost of computing the eigenvalues of N_s smaller matrices will be smaller than finding the eigenvalues of a single large $N \times N$ matrix by a factor of N_s^2!

In reality, this estimate is overly optimistic because to retain the same accuracy, it is usually necessary to use a local degree M which is larger, perhaps much larger, than N/N_s where N is assumed to be a satisfactory degree for the "global" Chebyshev series that is accurate over the entire target interval. Nevertheless, subdivision can be very much cheaper than applying a single companion matrix. Indeed, a simple subdivision strategy is now incorporated into the "chebfun" library.

Broadly speaking, there are four main options. One is to use a moderate number of subdivisions with polynomials of moderate degree; another is to use a huge number of subdivisions with very-low-degree approximations, such as cubic polynomials, on each subdomain [14]. The third option is to make a rather coarse initial subdivision and then deploy various theorems to identify subdomains that are provably "zero-free". One can then subdivide zero-possible intervals, as in bisection, until each subdomain is zero-free, or the surviving intervals are so narrow that a low-degree approximation is provably accurate, and can be cheaply solved to find the roots [15]. All these options are based on rigorous theoretical error bounds.

Some observations greatly improve the robustness and efficiency of these three varieties of subdivision schemes. First, the biggest challenge for the local, low-degree approximations is to represent the highest term included in the truncation, $T_N(x)$: if we can prove that the local approximation is accurate for $T_N(x)$, then it will be good for all the lower and slower-varying terms in the polynomial $f_N(x)$. Second, Chebyshev polynomials oscillate very non-uniformly with extremely rapid oscillations near the endpoints and much slower oscillations near the center of the interval. This non-uniformity can be removed by making the change of variable $x = \cos(t)$ which transforms a polynomial in Chebyshev form in x into a *cosine* polynomial in t. One can prove much tighter error bounds for the approximation of $f_N(\cos(t))$ than for the original polynomial in x. It is then comparatively easy, as explained in [14, 15], to use Cauchy's Interpolation Error Theorem to come up with errors in approximating $\cos(Nt)$, which then are upper bounds for approximating $f_N(\cos(t))$.

Boyd recommmends as a high-order scheme the "Tredecic" algorithm, which is to subdivide the interval into N subintervals that are equally spaced in the *trigonometric* coordinate t and then deploying a local approximation of degree thirteen on each subdomain at a cost of roughly $22{,}000N + 70\log_2(N)$ floating point operations [14] . Using $500N$ subdivisions with cubic polynomials on each also guarantees an absolute approximation error to $f_N(\cos(t))$ which is no worse than 10^{-12}, but this "Megacubes" method is always more expensive than the Tredecic. If, however, one can tolerate a higher error tolerance such as 10^{-8}, then the number of subdivisions drops to $N_s = 50N$, and "Megacubes" is faster for large N. The best subdivision method is faster than the Chebyshev companion-matrix method for $N \geq 50$ at the higher error tolerance and for $N > 25$ at an approximation error tolerance of 10^{-8}.

The fourth class of subdivision schemes is to arbitrarily choose an initial number of subdivisions N_s and an initial local degree M, and then rely on an adaptative Chebyshev interpolation scheme to increase M, and further subdivide individual subdomains, until a sufficiently accurate approximation has been obtained over the whole domain, and finally apply the companion-matrix method to generate the roots for each subdomain. Adaptation strategies are discussed further in Sect. 6 below; a new Bernstein-polynomial-based test for a "zero-free" interval was recently published in [16]. Such adaptive subdivision is easier to program and potentially much less expensive than the schemes of [14] and [15], which are very safe (being based on rigorous theoretical bounds) but also very conservative (designed to defend against possible but improbable worst cases.)

In reality, assessing the relative merits of the Chebyshev rootfinders is not straightforward. The Chebyshev companion-matrix method is ridiculously easy to program: a sample Matlab code containing just five lines and eleven statements is given as Table 2 below. It may be better to use a slow-but-easy-to-program algorithm than to waste human labor on a fast-but-elaborate method; as the old arithmurgical proverb goes, "A good workstation never sleeps." Furthermore the subdivision algorithms contain a lot of loops, conditionals and other overhead which is hardware-dependent and not captured by formal floating-point-operation counts.

4 Maxima, minima and inflection points

The maxima and minima of a function $f(x)$ are always located at roots of the first derivative. Similarly, the inflection points where the curvature of a function changes sign are the zeros of the second derivative. If $f(x)$ is in fact a truncated Chebyshev series, then the necessary derivatives can be computed by recurrence.

Let $a_k^{(q)}$ denote the coefficients of the qth derivative:

$$\frac{\mathrm{d}^q f}{\mathrm{d}x^q} = \sum_{k=0}^{N-q} a_k^{(q)}\, T_k(x). \tag{12}$$

These may be computed from the Chebyshev coefficients of the $(q-1)$st derivative by the recurrence relation (in descending order)

$$\begin{aligned} &a_{N-q+2}^{(q)} = a_{N-q+1}^{(q)} = 0, \\ &a_{k-1}^{(q)} = \frac{1}{c_{k-1}} \left\{ 2\,k\,a_k^{(q-1)} + a_{k+1}^{(q)} \right\}, \quad k = N-q+1, N-q,\ N-q-1, \ldots, 1, \end{aligned} \tag{13}$$

where $c_k = 2$ if $k = 0$ and $c_k = 1$ for $k > 0$. Similar recurrences for Legendre and other spectral series can be found in [10, Appendix A] and numerous other sources.

Thus, maxima, minima and inflection points may be computed by first computing the Chebyshev coefficients of as many derivatives of $f_N(x)$ as necessary, and then computing the roots of the derivative series. Computing the maxima and minima and inflection points are all problems in rootfinding.

5 Finding the roots of a transcendental function

If a function $f(x)$ is smooth on an interval $[a, b]$, it can always be "Chebyshevized" on that interval by Chebyshev interpolation. One can then find the roots of a transcendental function $f(x)$ on the interval by finding the zeros of its polynomial proxy, $f_N(x)$. Similarly, periodic functions can be approximated by trigonometric polynomials, and the roots of these proxies can be computed by the methods described earlier. It is thus possible to extend the existence of robust black-box rootfinders from polynomials to all smooth transcendental functions.

The details are described in [2–4, 9, 12, 13, 15–17], so we shall content ourselves with an amusing example and a brief analysis of the remaining theoretical and practical difficulties.

The amusing example is Kepler's equation from celestial mechanics:

$$E - \epsilon \sin(E) - M = 0, \tag{14}$$

where the unknown $E(M, \epsilon)$ is the "eccentric anomaly" of an orbit and ϵ is the eccentricity of the orbit. At least five hundred papers have been published on this problem over the last four centuries, and fresh ones still appear at an average of one or two a year [17]. Inserting the Chebyshev series

$$\sin(\pi x) = 2 \sum_{k=0}^{\infty} (-1)^k J_{2k+1}(\pi)\, T_{2k+1}(x), x \in [-1, 1] \tag{15}$$

converts the *transcendental* equation into a *polynomial* equation. If the series is truncated at fifteenth order and the result is rearranged into an ordinary polynomial, then the Kepler equation becomes approximately

$$\begin{aligned} M = E - \epsilon \Bigg\{ & 3.14\frac{E}{\pi} - 5.16\left(\frac{E}{\pi}\right)^3 + 2.55\left(\frac{E}{\pi}\right)^5 - 0.599\left(\frac{E}{\pi}\right)^7 \\ & + 0.082\left(\frac{E}{\pi}\right)^9 - 0.0073\left(\frac{E}{\pi}\right)^{11} + 0.00046\left(\frac{E}{\pi}\right)^{13} - 0.000018\left(\frac{E}{\pi}\right)^{15} \Bigg\}, \end{aligned} \tag{16}$$

where the numbers have been truncated to the first three significant figures only to simplify the display. The maximum absolute error in computing the root of Kepler's equation by solving this proxy is only 4×10^{-10} for all $\epsilon \in [0, 1]$ and for all $M \in [-\pi, \pi]$, which is the entire range of interest for elliptical and parabolic orbits. The simplicity of the polynomial approximation seems rather an anticlimax after four centuries of relentless and highly variegated ingenuity.

The Chebyshev-transcendental-rootfinding method has five issues requiring further discussion:

1. Iterative refinement
2. Choice of approximation interval.
3. Smoothness of $f(x)$
4. Dynamic range of $f(x)$
5. Adaptive Chebyshev interpolation, perhaps with subdivision

The companion matrix and subdivision methods are all global methods in the sense that they approximately compute all the roots on the canonical interval, which is $x \in [-1, 1]$ for Chebyshev polynomials. Although these algorithms are capable of achieving near-machine precision, this is not really necessary, especially when the polynomial $f_N(x)$ is used as a proxy for a transcendental function $f(x)$. One can use Newton's iteration to refine the approximate roots of $f_N(x)$ to machine-precision-accurate roots of $f(x)$.

One drawback of the Chebyshev-proxy method is that the user must choose an approximation interval, $x \in [a, b]$, which is to be searched for roots. (Through a trivial change of variable, this can be mapped to the canonical interval for Chebyshev polynomials, $x \in [-1, 1]$.) To find all zeros on the *whole* real axis, even if there are an infinite number of them, does not necessarily defeat the Chebyshev-proxy strategy: one can often find asymptotic approximations for large roots, thus requiring the user to numerically compute only a finite number of roots on a finite interval, or if the number of roots is finite, one can use a coordinate change to map the infinite interval into a finite one, as illustrated in [3]. Still, when $f(x)$ is transcendental, the user cannot be entirely spared the obligation of knowing at least a little about the target function and its roots.

The third issue is obvious to anyone familiar with Chebyshev interpolation: the approximation converges rapidly only if $f(x)$ is infinitely differentiable everywhere on the approximation interval $x \in [a, b]$. The method will not be accurate for a non-smooth function like $\sin(10|x|)$ on $x \in [-1, 1]$, for example, because the first derivative is discontinuous at the origin.

The fourth complication is that a function $f(x)$ may be huge on some subintervals of interest but very tiny on others. An example is

$$f(x) = \exp(-100x^2)\ \sin(50x) \tag{17}$$

on $x \in [-1, 1]$. Chebyshev interpolants are highly uniform in *absolute* error, but for a function like (17), this translates into rather bad *relative* errors near the ends of the interval where the function is oscillating between extremely small peaks and valleys. In photography, this is the "dynamic range" problem; in a scene with a high contrast, a photograph focused on features in bright sunlight will capture areas in shadow as solid black, whereas a photograph focused on the shadows will "wash out" the bright regions as patches of solid white. One remedy as discussed in [2] is to multiply $f(x)$ by a positive function that removes the variations in range; for (17), one can define

$$g(x) \equiv \exp(100x^2) f(x) = \sin(50x) \tag{18}$$

and then all the peaks and valleys of the rescaled function are one, and the Chebyshev-proxy rootfinder is extremely accurate for all roots. Another strategy is to subdivide the interval into multiple subintervals and find the roots of separate, low-degree Chebyshev interpolants on each subdomain. As noted above, subdivision is often a very good strategy for *reducing cost*; for functions with a large dynamic range, it also is an excellent strategy for *improving accuracy*.

Lastly, when a Chebyshev rootfinder is applied to find the roots of a polynomial $f_N(x)$ that is really a sort of mathematical stunt double for a transcendental function $f(x)$, the accuracy of the roots can be no better than the accuracy of the approximation of $f(x)$ by its Chebyshev interpolant $f_N(x)$. Furthermore, the goal is not merely to obtain an approximation with a small *absolute* error, but rather one in which the *relative* errors are small over the entire target interval.

To meet this goal, a wide variety of rather simple-minded adaptive Chebyshev strategies have been employed. Since the "adaptation issue" has not been discussed in any of the rootfinding articles mentioned above, it is appropriate to delve a little deeper here.

6 Adaptive Chebyshev interpolation

"However it has been the authors' experience that $|a_N|$ in general provides a satisfactory error estimate even when the convergence of the corresponding Fourier–Chebyshev expansion is slow, and that it dominates the error when convergence is rapid."

W. Fraser and M. W. Wilson [18].

If $f(x)$ is analytic on the expansion interval, then its Chebyshev coefficients will fall geometrically. If the coefficients are of the asymptotic form

$$a_n \sim p\ \exp(-qn), \tag{19}$$

where p and q are positive constants, then when the series is truncated after a_N, the truncation error E_N is bounded by [[11], Chapter 2]

$$E_N \le p \exp(-qN) \frac{\exp(-q)}{1 - \exp(-q)}. \tag{20}$$

Unless $\exp(-q)$ is very close to one, the order-of-magnitude of the error is the same as $|a_N|$, which is the "Last Coefficient Rule-of-Thumb" of [10].

Other authors have used more sophisticated criteria as catalogued in Table 1. Some have tried to fit the asymptotic form (19). However, as explained with examples in [[10], Chapter 2] it is common for Chebyshev coefficients to *oscillate* with degree. Convergence theory guarantees that the coefficients a_n can be bounded from above by $p\ \exp(-qn)$, the "envelope" of the coefficients, for some p and q, but the decay often is non-monotonic. A function with parity such as $\cos(x)$ is an extreme example: all its odd coefficients

Table 1 Adaptation criteria for Chebyshev series ϵ is a user-chosen tolerance and N is the highest retained Chebyshev degree

Reference	Error test
[21]	$k^2\lvert a_{N-2}\rvert, k\lvert a_{N-1}\rvert, \lvert a_N\rvert \le \epsilon$ k typically 1/8
[18]	$\lvert a_N\rvert \le \epsilon$
[19]	Separate treatment of even and odd coefficients $p\ \exp(-qn)/(1-\exp(-q)) < \epsilon$ where p, q, n fitted to the last three coefficients of each parity
[22]	$\lvert a_{N-2}\rvert, \lvert a_{N-1}\rvert, \lvert a_N\rvert \le \epsilon\ \max_n \lvert a_n\rvert$
[6]	$\lvert a_{N-1}\rvert, \lvert a_N\rvert \le 2$ machine epsilon
[23]	Fitting of last few coefficients by decaying exponential $p\ \exp(-qn)$: $p\ \exp(-qn)/(1-\exp(-q)) < \epsilon$
This work	$\max_{j\in[0,2N]} \lvert f(x_j) - f_N(x_j)\rvert \le \epsilon$ x_j are the points of a $(2N+1)$-point Lobatto grid

are zero. For this reason, Sloan and Smith fit the even-degree and odd-degree coefficients separately, and then increase the number of grid points if the larger of the two errors exceeds the user-chosen tolerance [19]. Similarly, other authors such as Jacobs and Battles and Trefethen examine the *last three* or *last two* coefficients.

However, it is easy to construct examples which can defeat such schemes using the following.

Theorem 3 (Lacunary Chebyshev Series) *Suppose*

$$f(x) \equiv g(T_k(x)), \tag{21}$$

where k is an integer and $g(x)$ is analytic everywhere on $x \in [-1,1]$. Then all Chebyshev coefficients of $f(x)$ whose degrees are not multiples of k are zero, that is,

$$f(x) \equiv \sum_{n=0}^{\infty} a_{nk} T_{nk}(x). \tag{22}$$

Proof

With the trigonometric change of coordinate, $x = \cos(t)$, $f(\cos(t)) = g(\cos(kt))$. With the further change of coordinate $T \equiv kt$, this becomes $g(\cos(T))$ which, because of the analyticity of g has a nice Fourier-cosine series in T. Converting this series back to t gives a Fourier series in which all terms, except those whose degrees are multiples of k, are missing. Converting from t back to x gives the theorem. QED

Thus, a function like $f(x) = (3/4)/(5/4 - T_7(x)) = 1 + 2\sum_{n=1}^{\infty}(1/2)^n T_{7n}(x)$ can fool adaptive schemes that examine just the last three coefficients. However, it is unclear if such functions with "runs" of three or more zero coefficients ever arise in applications.

Indeed the situation for simple Chebyshev error tests is probably rather like that of Gaussian elimination [20]: the worst case is very bad, but in a probalistic sense, the chances of trouble are actually rather small. Unfortunately, at present, no way is known to quantify "typical" Chebyshev series.

We shall therefore briefly describe a "super-safe" strategy. First, compute the Chebyshev interpolant $f_N(x)$ on the Lobatto (endpoints-and-extrema) grid where N, to allow the use of the fast cosine transform, is a power of two. Second, calculate the maximum error of the interpolant at the points of a Lobatto grid with $2N + 1$ points:

$$E_{\text{interstices}}(N) \equiv \max_{j\in[0,2N]} |f(x_j) - f_N(x_j)|. \tag{23}$$

Third, compare $E_{\text{interstices}}(N)$ with the user-chosen error tolerance ϵ. If $E_{\text{interstices}}(N) > \epsilon$, then compute $f_{2N}(x)$ by interpolation at the points of the $(2N+1)$-point Lobatto grid and repeat. The choice of N as a power of two, and the decision to evaluate the error on a grid of $2N+1$ points, has the advantage that all Chebyshev computations can be performed by the fast cosine transform, and since every point on the interpolation grid for N is also part of the Lobatto grid when N is doubled, all evaluations of $f(x)$ are recycled each time N is doubled. Of course, the error is zero at the $N+1$ points where $f_N(x)$ interpolates $f(x)$, so the extra N points lie, in the trigonometric coordinate $t = \arccos(x)$, at the midpoints of each subinterval between points where $f_N(x) = f(x)$. Thus, the error is evaluated at those points where it is expected to be worse.

This "super-safe" strategy is still not infallible. It will be fooled, for example, if N is chosen to be initially sixteen, say, and $f(x)$ happens to be zero at each of the points of a 33 point Lobatto grid: the adaptation will stop with a Chebyshev approximation that is identically zero! However, it is obvious that $f(x)$ which satisfy 31 conditions are much less likely than $f(x)$ whose Chebyshev series satisfies the three conditions that a_{N-2}, a_{N-1} and a_N are all zero for some N, even though higher coefficients are non-zero. Like a kindly financial advisor, we shall leave each reader to find his or her own comfort point on the risk-versus-cost scale. The probabilities of being tripped up by the various strategies catalogued in the table are at present as uncertain as the future performance of the stock market.

7 History

The companion-matrix idea was generalized to orthogonal polynomials independently in [13, 24–28]. The early work, however, died out like *homo neanderthalensis*. The papers on solving transcendental equations by Chebyshev interpolation of $f(x)$ [2, 3] and Battle and Trefethen's development of a Chebyshev-interpolation-based extension of Matlab to functions of a continuous variable inspired the flurry of recent papers cited above.

Similarly, there was much activity on finding the roots of trigonometric polynomials between 1982 and 1994, but no papers in the following 11 years.

The first trio of trigonometric papers employed a generalization of the Durand–Kerner iteration for *simultaneously* finding all roots of a polynomial [29–31]. Frommer gave a more abstract justification of the trigonometric Durand–Kerner method that simplified the proof that the algorithm converges quadratically near the roots and slightly simplified the algorithm [32]. Another Durand–Kerner article is [5], which is the earliest appearance of the $z = \exp(it)$ transformation for converting a trigonometric polynomial into an ordinary polynomial.

Schweikard introduces three new ideas. The first is the conversion of a trigonometric polynomial to an algebraic polynomial through a different (and less simple) means than Weidner: the transformation [33]

$$t = 2\arctan(x) \qquad \leftrightarrow \qquad x = \tan(t/2). \tag{24}$$

This implies the identities

$$\cos(t) = \frac{1-x^2}{1+x^2}, \quad \sin(t) = \frac{2x}{1+x^2}. \tag{25}$$

Since $\cos(jt)$ and $\sin(jt)$ can be expanded as series of powers of $\cos(t)$ or as $\sin(t)$ times a cosine polynomial, an arbitrary trigonometric polynomial can always be transformed into a rational function of x, which becomes an ordinary polynomial after multiplying through by the appropriate power of $1+x^2$. This transformation is more cumbersome than Weidner's, but it has the advantage that trigonometric polynomials with real coefficients are transformed to algebraic polynomials with real coefficients.

The second is the observation that the concept of a "square-free" polynomial, that is, one that has only simple roots, can be extended to trigonometric polynomials [33]. The algorithm is to convert to an algebraic polynomial, calculate the square-free algebraic polynomial, and then convert back to trigonometric form. This can be done without error in a computer-algebra system if the coefficients are rational or algebraic numbers.

The third idea is interval arithmetic [34]. In a trio of papers, Carstensen and collaborators developed the interval arithmetic ideas at greater length and generalized the Durand–Kerner simultaneous rootfinding method to higher-order iterations, that is, those converging faster than quadratically [35–37]. Carstensen and Petkovic note that Weidner's transformation, which yields an algebraic polynomial with complex coefficients, even when the original trigonometric polynomial has only real coefficients, is undesirable in interval arithmetic methods because the intervals are forced to become bounding rectangles in the complex plane.

The preoccupation with the Durand–Kerner iteration is a bit surprising. For ordinary polynomials, experience shows that, if the starting points are chosen on a circle in the complex plane which encloses all the roots, the algorithm usually converges. Although convergence is guaranteed to be quadratic very close to the roots, the Durand–Kerner method often converges quite slowly for the early iterates. For multiple roots, the Durand–Kerner method converges linearly, that is, as δ^m for some $\delta < 1$ where m is the iteration number. All these remarks carry over to applications to trigonometric polynomials. In particular, there is no *global* convergence theory — only theorems about rates of convergence *near* the roots. No comparisons with the Fourier companion-matrix method have yet been made.

8 Conclusions

There are now robust algorithms for finding the zeros of truncated Fourier series and orthogonal polynomials. Companion-matrix methods, which are very easy to program and extremely well-conditioned, should be the first choice, second choice and third choice for computing the roots of polynomials in Chebyshev form and trigonometric polynomials.

However, when the polynomial $f_N(x)$ is of high degree N, subdividing the search interval into subdomains and finding the roots of local Chebyshev interpolants of moderate degree on each subdomain, will be much cheaper than applying the companion matrix method to $f_N(x)$ directly: the cost of the companion matrix method is $O(12N^3)$ floating-point operations. Furthermore, subdivision is accuracy-increasing, too, when $f(x)$ is oscillating between wildly different ranges in different parts of the interval [2], i.e. exhibits the "dynamic range" problem.

Adaptive Chebyshev interpolation is a key ingredient both in solving transcendental equations by Chebyshev interpolation-followed-by-Chebyshev-companion-matrix-eigensolving, and also in subdivision methods for solving a polynomial of high degree. Good *ad hoc* strategies are available, as described above, but no perfect strategy is possible, because, except for the approximation of one polynomial by another, one can always devise exotic examples to mislead any particular error test [14, 15].

Acknowledgements Supported by the NSF through grant OCE0451951. I thank the guest editors, C. Pozrikidis, Jan Hesthaven and George Em Karniadakis, for inviting me to contribute to this special issue. I thank the reviewers for helpful comments. I thank H. K. Kuiken for his very careful proofing and editing.

Appendix. Companion matrices for Legendre, Hermite, Gegenbauer and general orthogonal polynomials

A.1 Legendre companion matrix

Theorem 4 (Legendre companion matrix) *Let $f_N(x)$ denote a polynomial of degree N written in "Legendre form" as*

Table 2 Matlab code for the Legendre polynomial companion matrix

```
% Input: coefficients s(1), ..., s(N + 1) for truncated Legendre series of degree N
P = zeros(N, N); P(1, 2) = 1; P(N, 1) = −(s(1)/s(N + 1)) * (N/(2 * N − 1));
for j = 2 : (N − 1), P(j, j − 1) = (j − 1)/(2 * j − 1); P(j, j + 1) = j/(2 * j − 1);
P(N, j) = −(s(j)/s(N + 1)) * (N/(2 * N − 1)); end
P(N, N − 1) = P(N, N − 1) + (N − 1)/(2 * N − 1); P(N, N) = −(N * s(N)/(s(N + 1)*
(2 * N − 1))); rootsfromeig = eig(P); % compute Legendre roots
```

$$f_N(x) = \sum_{j=0}^{N} s_j\, P_j(x). \tag{26}$$

Then all roots of $f_N(x)$, both on and off the canonical expansion interval, $x \in [-1, 1]$, are eigenvalues of the $N \times N$ matrix $\vec{\vec{P}}$ whose elements are

$$P_{jk} = \begin{cases} \delta_{2,k}, \quad j = 1, \;\; k = 1, 2, \ldots, N, \\ \frac{j-1}{2j-1}\delta_{j,k+1} + \frac{j}{2j-1}\delta_{j,k-1}, \qquad j = 1, \ldots, (N-1), \;\; k = 1, \ldots, N, \\ (-1)\frac{N}{2N-1}\frac{s_{j-1}}{s_N} + \frac{N-1}{2N-1}\delta_{k,N-1}, \qquad j = N, \;\; k = 1, \;\; , N, \end{cases} \tag{27}$$

where δ_{jk} is the Kronecker delta function such that $\delta_{jk} = 0$ if $j \neq k$, while $\delta_{jj} = 1$ for all j.

For a quintic polynomial, we have for example,

$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ (1/3) & 0 & (2/3) & 0 & 0 \\ 0 & (2/5) & 0 & (3/5) & 0 \\ 0 & 0 & (3/7) & 0 & (4/7) \\ (-1)\frac{5}{9}\frac{s_0}{s_5} & (-1)\frac{5}{9}\frac{s_1}{s_5} & (-1)\frac{5}{9}\frac{s_2}{s_5} & (-1)\frac{5}{9}\frac{s_3}{s_5} + (4/9) & (-1)\frac{5}{9}\frac{s_4}{s_5} \end{vmatrix}. \tag{28}$$

The complete Matlab code to generate the companion matrix is given as Table 2.

A.2 Gegenbauer polynomials and spherical harmonics

The Gegenbauer polynomials, also known as the ultraspherical polynomials, depend on an order parameter m which is the subscript on the functions. The polynomials include the Chebyshev and Legendre polynomials as the special cases $m = 0$ and $m = 1/2$, respectively. The Gegenbauer polynomials for half-integer m are, modulo a non-negative factor, the latitudinal parts of spherical harmonics. A multiplicity of notations and normalizations are employed. We shall specify the Gegenbauer polynomials uniquely, as in [10, Appendix A] by their recurrence

$$C_0^{(m)} \equiv 1 \qquad ; \qquad C_1^{(m)}(x) \equiv 2\, m\, x, \tag{29}$$
$$(n+1)\; C_{n+1}^{(m)}(x) = 2\,(n+m)\, x\; C_n^{(m)}(x) - (n + 2m - 1)\; C_{n-1}^{(m)}(x).$$

Theorem 5 *The roots of a series of Gegenbauer polynomials,*

$$f_N(x) = \sum_{j=0}^{N} s_j\, C_j^{(m)}(x) \tag{30}$$

are the eigenvalues of the matrix whose elements are

$$G_{1k} = \delta_{2k}(1/[2m]), \qquad j = 1 \tag{31}$$

$$G_{jk} = \frac{(j-1+2m-1)}{2\,(j-1+m)}\delta_{j-1,k} + \frac{j}{2\,(j-1+m)}\delta_{j+1,k}, \quad j = 2, \ldots, (N-1), \tag{32}$$

$$G_{Nk} = -\frac{s_{k-1}\,\frac{N)}{2\,(N-1+m)}}{s_N} + \delta_{N-1,k}\,\frac{(N-2+2m)}{2\,(N-1+m)}, \qquad j = N, \tag{33}$$

For a polynomial of degree four in Gegenbauer form,

$$\begin{vmatrix} 0 & 1/[2m] & 0 & 0 \\ \frac{2m}{2\,(1+m)} & 0 & \frac{2}{2\,(1+m)} & 0 \\ 0 & \frac{2m+1}{2\,(2+m)} & 0 & \frac{3}{2\,(2+m)} \\ \frac{-4}{2\,(3+m)}\frac{s_0}{s_4} & \frac{-4}{2\,(3+m)}\frac{s_1}{s_4} & \left[\frac{-4}{2\,(3+m)}\frac{s_2}{s_4} + \frac{(2m+2)}{2\,(3+m)})\right] & \frac{-4}{2\,(3+m)}\frac{s_3}{s_4} \end{vmatrix}. \tag{34}$$

A.3 Hermite polynomials and Hermite functions

Theorem 6 (Hermite companion matrix) *Let $f_N(x)$ denote a polynomial of degree N written in "Hermite form" as*

$$f_N(x) = \sum_{j=0}^{N} h_j\, H_j(x) \tag{35}$$

or a function that is a truncated series of Hermite functions,

$$f_N(x) = \sum_{j=0}^{N} h_j\, \psi_j(x), \tag{36}$$

where $\psi_j(x) = \exp(-[1/2]x^2)\, H_j(x)$. (Note that the Gaussian factor that is part of the definition of the Hermite functions has no effect on the zeros of the sum of the series.) Then all roots of $f_N(x)$, both on and off the canonical expansion interval, $x \in [-\infty, \infty]$, are eigenvalues of the $N \times N$ matrix $\vec{\vec{H}}$ whose elements are

$$H_{jk} = \begin{cases} (1/2)\delta_{2,k}, \quad j = 1, & k = 1, 2, \ldots, N, \\ (j-1)\delta_{j,k+1} + \frac{1}{2}\delta_{j,k-1}, & j = 2, \ldots, (N-1), \\ (-1)\frac{1}{2}\frac{h_{j-1}}{h_N} + (N-1)\delta_{k,N-1}, & j = N, \end{cases} \tag{37}$$

where δ_{jk} is the usual Kronecker delta function.

For a quintic polynomial, the matrix is

$$\begin{vmatrix} 0 & (1/2) & 0 & 0 & 0 \\ 1 & 0 & (1/2) & 0 & 0 \\ 0 & 2 & 0 & (1/2) & 0 \\ 0 & 0 & 3 & 0 & (1/2) \\ (-1)\frac{h_0}{2\,h_5} & (-1)\frac{h_1}{2h_5} & (-1)\frac{h_2}{2h_5} & (-1)\frac{h_3}{2h_5} + 4 & (-1)\frac{h_4}{2\,h_5} \end{vmatrix}. \tag{38}$$

A.4 General orthogonal polynomials

Theorem 7 (General polynomials) *If a set of polynomials $\psi_n(x)$ satisfy a three-term recurrence relation such that*

$$x\psi_n = B(n)\psi_{n+1} + D(n)\psi_n + E(n)\,\psi_{n-1} \tag{39}$$

then the roots of the polynomial

$$f_N(x) \equiv \sum_{j=0}^{N} s_j\,\psi_j(x) \tag{40}$$

are the eigenvalues of the companion matrix with elements

$$Q_{jk} = \begin{cases} D(0)\delta_{1k} + B(0)\delta_{1,k-1}, & j=1 \\ E(j-1)\delta_{j,k+1} + D(j-1)\delta_{jk} + B(j-1)\,\delta_{j,k-1}, & \textit{otherwise} \\ -B(N-1)s_{k-1}/s_N + E(N-1)\delta_{N-1,k} + D(N-1)\delta_{Nk}, & j=N, \end{cases} \tag{41}$$

where δ_{jk} is the Kronecker delta function.

For a quintic polynomial, the companion matrix is

$$\left|\begin{matrix} D(0) & B(0) & 0 & 0 \\ E(1) & D(1) & B(1) & 0 \\ 0 & E(2) & D(2) & B(2) \\ \dfrac{-B(3)\,s_0}{s_4} & \dfrac{-B(3)\,s_1}{s_4} & \dfrac{-B(3)\,s_2}{s_4} + E(3) & \dfrac{-B(3)\,s_3}{s_4} + D(3) \end{matrix}\right|. \tag{42}$$

References

1. Boyd JP (2004) A review of high-order methods for incompressible fluid flow by M. O. Deville, P. F. Fischer and E. H. Mund. SIAM Rev 46:151–157
2. Boyd JP (1995) A Chebyshev polynomial interval-searching method ("Lanczos economization") for solving a nonlinear equation with application to the nonlinear eigenvalue problem. J Comput Phys 118:1–8
3. Boyd JP (2002) Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding. SIAM J Num Anal 40:1666–1682
4. Boyd JP (2006) Finding the roots of a transcendental equation: how Chebyshev polynomials, algebraic geometry and matrix theory have solved a quest begun in Babylon. Math Intell submitted
5. Weidner P (1988) The Durand–Kerner method for trigonometric and exponential polynomials. Computing 40:175–179
6. Battles Z, Trefethen LN (2004) An extension of Matlab to continuous functions and operators. SIAM J Sci Comp 25:1743–1770
7. Toh K, Trefethen LN (1994) Pseudozeros of polynomials and pseudospectra of companion matrices. Num Math 68:403–425
8. Parlett BN, Reinsch C (1969) Balancing a matrix for calculation of eigenvalues and eigenvectors. Num Math 13:293–304
9. Boyd JP, Gally DH (2005) Numerical experiments on the accuracy of the Chebyshev–Frobenius companion matrix method for finding the zeros of a truncated series of Chebyshev polynomials. J Comp Appl Math (in proof)
10. Boyd JP (2001) Chebyshev and Fourier spectral methods. 2nd edn. Dover, Mineola, New York
11. Jónsson GF, Vavasis S (2004) Solving polynomials with small leading coefficients. SIAM J Matrix Anal Appl 26:400–414
12. Boyd JP (2006) Computing the zeros of a Fourier series or a Chebyshev series or general orthogonal polynomial series with parity symmetries. Comput Math Appl submitted
13. Day D, Romero L (2005) Roots of polynomials expressed in terms of orthogonal polynomials. SIAM J Num Anal 43:1969–1987
14. Boyd JP (2006) Computing real roots of a polynomial in Chebyshev series form through subdivision. Appl Num Math in proof
15. Boyd JP (2006) Computing real roots of a polynomial in Chebyshev series form through subdivision with linear testing and cubic solves. Appl Math Comp 174:1642–1648

16. Boyd JP (2006) A test for an interval to be free of zeros for polynomials in Chebyshev form through conversion to the Bernstein polynomial basis. Appl Math Comp submitted
17. Boyd JP (2006) Rootfinding for a transcendental equation without a first guess: Polynomialization of Kepler's equation through Chebyshev polynomial expansion of the sine. Appl Num Math 52:12–18
18. Fraser W, Wilson MW (1966) Remarks on the Clenshaw–Curtis quadrature scheme. SIAM Rev 8:322–327
19. Sloan IH, Smith WE (1980) Product integration with the Clenshaw–Curtis quadrature scheme. SIAM Rev 8:322–327
20. Trefethen LN, Schreiber RS (1990) Average-case stablity of Gaussian elimination. SIAM J Matrix Anal Appl 11: 335–360
21. Clenshaw CW, Curtis AR (1960) A method for numerical integration on an automatic computer. Num Math 2:197–205
22. Jacobs SJ (1990) A variable order pseudospectral method for two-point boundary value problems. J Comp Phys 88:169–182
23. Mason JC, Handscomb DC (2003) Chebyshev polynomials. Chapman and Hall/CRC press, Boca Raton, Florida
24. Specht W (1957) Die Lage der Nullstellen eines Polynom III. Math Nach 16:369–389
25. Specht W (1960) Die Lage der Nullstellen eines Polynom IV. Math Nach 21:201–222
26. Barnett S (1975) Companion matrix analog for orthogonal polynomials. Linear Algebra Appl 12:197–208
27. Stetter HJ (2004) Numerical polynomial algebra, Society for Industrial and Applied Mathematics. Philadelphia, Pennsylvania
28. Gol'berg EM, Malozemov VN (1979) Estimates for the zeros of certain polynomials. Vestmol Leningrad Univ Math 6:127–135. Translated from Vestnik Leningrad Univ Mat Mekh Astr 7(1973), 18–24
29. Angelova ED, Semerdzhiev KI (1982) Methods for the simulataneous approximate derivation of the roots of algebraic, trigonometric and exponential equations. USSR Comp Maths Math Phys 22:226–232
30. Makrelov IV, Semerdzhiev HI (1985) Methods for the simultaneous determination of all zeros algebraic, trigonometric and exponential equations. USSR Comp Maths Math Phys 24:1443–1453
31. Makrelov IV Semerdzhiev HI (1985) On the convergence of two methods for the simultaneous find of all roots of exponential equations. IMA J Num Anal 5:191–200
32. Frommer A (1988) A unified approach to methods for the simulataneous computation of all zeros of generalized polynomials. Num Math 54:105–116
33. Schweikard A (1991) Trigonmetric polynomials with simple roots. Inform Process Lett 39:231–236
34. Schweikard A (1992) Real zero isolation for trigonmetric polynomials. ACM Trans Math Software 18:350–359
35. Carstensen C, Petkovi'c MS (1993) On some interval methods for algebraic, exponential and trigonomertric polynomials. Computing 51:313–326
36. Carstensen C, Reinders M (1993) On a class of higer order methods for simultaneous rootfinding of generalized polynomials. Num Math 64:69–84
37. Carstensen C (1994) A note on simulataneous rootfinding for algebraic, exponential, and trigonomertric polynomials Comp Math Appl 27:7–14