SIAM REVIEW
Vol. 62, No. 1, pp. 231–243

The Runge Example for Interpolation and Wilkinson's Examples for Rootfinding*

Robert M. Corless†
Leili Rafiee Sevyeri†

Abstract. We look at two classical examples in the theory of numerical analysis, namely, the Runge example for interpolation and Wilkinson's example (actually two examples) for rootfinding. We use the modern theory of backward error analysis and conditioning, as instigated and popularized by Wilkinson but refined by Farouki and Rajan. By this means, we arrive at a satisfactory explanation of the puzzling phenomena encountered by students when they try to fit polynomials to numerical data, or when they try to use numerical rootfinding to find polynomial zeros. Computer algebra, with its controlled, arbitrary precision, plays an important didactic role.

Key words. interpolation, rootfinding, conditioning, sensitivity

AMS subject classification. 97N40

DOI. 10.1137/18M1181985

The function $y = \frac{1}{1+25x^2}$ on $-1 \leq x \leq 1$, called the Runge example,[1] is used in many numerical analysis textbooks to show why high-degree polynomial interpolation *using equally spaced nodes* is bad: see, for instance, any of [16], [15], [4], [20], [1], [8], [11], [19], [12], and [17]. Some textbooks, even the iconic [14], do not emphasize the crucial qualification "using equally spaced nodes," or, perhaps, many readers skip over it[2] or don't retain it for other reasons, leaving only the false impression that high-degree interpolation is *always* bad.

Similarly, Wilkinson's first example polynomial[3]

$$p_{20}(x) = \prod_{k=1}^{20}(x-k) \tag{1}$$

is widely discussed as an example—maybe *the* canonical example—of polynomial perfidy, this time for rootfinding rather than interpolation. As we will discuss below,

*Received by the editors January 17, 2018; accepted for publication (in revised form) February 7, 2019; published electronically February 11, 2020.
https://doi.org/10.1137/18M1181985

†Ontario Research Center for Computer Algebra and The School of Mathematical and Statistical Sciences, University of Western Ontario, London, ON N6A 5B7, Canada (rcorless@uwo.ca, lrafiees@uwo.ca).

[1]Carl David Tolmé Runge (30 August 1856 – 3 January 1927) was a German mathematician, physicist, and spectroscopist.

[2]"tl;dr" as the kids say nowadays.

[3]James Hardy Wilkinson. Born: 27 September 1919 in Strood, Kent, England. Died: 5 October 1986 in Teddington, Middlesex, England. He worked with Turing in 1946, won the Chauvenet Prize in 1970 for mathematical exposition for his paper "The Perfidious Polynomial" [22], and his book, "The Algebraic Eigenvalue Problem," was foundational in the field of numerical linear algebra.

both examples are better explained using the theory of *conditioning* as developed, for instance, by Farouki and Rajan [10]. One considers a polynomial

$$p(x) = \sum_{k=0}^{n} c_k \phi_k(x) \tag{2}$$

expressed in some polynomial basis $\{\phi_k(x)\}_{k=0}^{n}$ for polynomials of degree at most n. The usual monomial basis $\phi_k(x) = x^k$ is the most common, but is by no means the best example for all purposes. Farouki and Goodman [9] point out that the Bernstein basis $\phi_k(x) = \binom{n}{k} x^k (1-x)^{n-k}$ has the best conditioning in general, out of all polynomial bases satisfying $\phi_k(x) \geq 0$ on the interval $0 \leq x \leq 1$. Surprisingly, Corless and Watt [7] show that Lagrange bases can be better; see also Carnicer, Khiar, and Peña [5].

We now discuss Farouki and Rajan's formulation. The idea is to investigate the effects of small relative changes to the coefficients c_k, such as might arise from data error or perhaps approximation or computational error. The model is

$$p(x) + \Delta p(x) = \sum_{k=0}^{n} c_k (1+\delta_k) \phi_k(x), \tag{3}$$

where each $|\delta_k| \leq \varepsilon$, usually taken to be small. For instance, if $\varepsilon = 0.005$, each coefficient can be in error by no more than 0.5%. In particular, zero coefficients are not allowed to be disturbed at all.

Then

$$|\Delta p(x)| = \left| \sum_{k=0}^{n} c_k (1+\delta_k) \phi_k(x) - \sum_{k=0}^{n} c_k \phi_k(x) \right| \tag{4}$$

$$= \left| \sum_{k=0}^{n} c_k \delta_k \phi_k(x) \right|. \tag{5}$$

By the triangle inequality, this is

$$\leq \sum_{k=0}^{n} |c_k \delta_k \phi_k(x)| \tag{6}$$

$$\leq \left(\sum_{k=0}^{n} |c_k| |\phi_k(x)| \right) \max_{0 \leq k \leq n} |\delta_k| \tag{7}$$

$$\leq B(x) \cdot \varepsilon, \tag{8}$$

where

$$B(x) = \sum_{k=0}^{n} |c_k| |\phi_k(x)| \tag{9}$$

serves, for each x, as a *condition number* for polynomial evaluation.

This can be contrasted with the definition of "evaluation condition number" that arises when thinking of error Δx in the input: If x changes to $x + \Delta x$, then $y = f(x)$ changes to $y + \Delta y$, where calculus tells us that

$$\frac{\Delta y}{y} \doteq \frac{x f'(x)}{f(x)} \cdot \frac{\Delta x}{x}\,. \tag{10}$$

Here $C = xf'(x)/f(x)$ is the condition number, instead of $B(x)$, and B and C measure the responses to different types of error (coefficients and input). We consider B here.

Remark 0.1. There are many theorems[4] in numerical analysis that say, effectively, that when evaluating equation (2) in IEEE floating-point arithmetic, the computed result is exactly of the form of equation (3) for $|\delta_k| < Ku$, where K is a modest constant and u is the unit roundoff — in double precision, $2^{-53} \doteq 10^{-16}$. This is one motivation to study the effects of such perturbations, but there are others.

The Runge Example. If the equally spaced nodes $x_k = -1 + 2k/n$, $k = 0, \ldots, n$, are used to interpolate a function with a single polynomial of degree at most n, and the basis functions

$$\ell_k(x) = \frac{\prod_{\substack{j=0\\ j\neq k}}^{n} (x - x_j)}{\prod_{\substack{j=0\\ j\neq k}}^{n} (x_k - x_j)}, \tag{11}$$

which are the Lagrange interpolation basis functions, are used, then the coefficients are the values for the Runge function

$$y_k = f(x_k) = \frac{1}{1 + 25x_k^2}\,. \tag{12}$$

Then the condition number of the interpolant is

$$B(x) = \sum_{k=0}^{n} \frac{1}{1 + 25x_k^2} |\ell_k(x)|\,. \tag{13}$$

Choosing $n = 5, 8, 13, 21, 34, 55,$ and 89, we plot $B(x)$ on a logarithmic vertical scale for $-1 \le x \le 1$. The result is shown in Figure 1. The Maple code used to generate that figure is as follows (similar code using MATLAB can be developed, but the Maple code below avoids numerical issues in the construction of $B(x)$ by using exact rational arithmetic). We see that the maximum values for $B(x)$ occur near $x = \pm 1$, and that for any n there is an interval over which $B(x)$ is small.

```
Digits := 15:
Ns := [seq(combinat[fibonacci](k), k=5..11)]:
f := x -> 1/(1 + 25*x^2):

for N in Ns do
    tau := [seq(-1 + 2*k/N, k=0..N)]:
    rho := [seq(y[k], k=0..N)]:

    p := CurveFitting[PolynomialInterpolation](tau, rho, z, form=
        Lagrange):
```

[4]See, for instance, those cited in Chapter 2 of [6], or equation (2) of [17], which gives a backward error result for an inner product; in that case, the modest "constant" K is proportional to the dimension of the vector. See [18] for better, probabilistic error bounds.

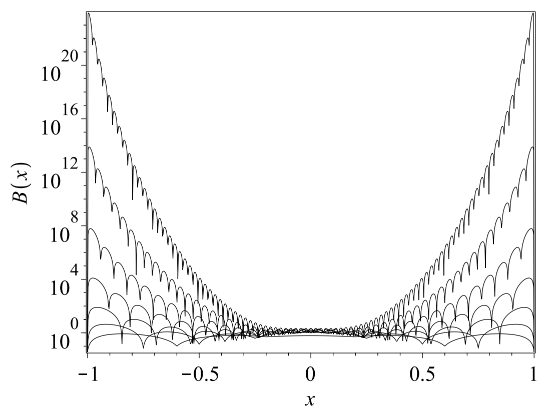

Fig. 1 *The condition number of the Runge example on equally spaced nodes with degrees* $n = 5, 8, 13, 21, 34, 55,$ *and* 89.

```
    B := map(abs, p):
    BRunge := eval(B, [seq(y[k] = f(tau[k+1]), k=0..N)]):

    pl[N] := plots[logplot](BRunge, z=-1..1, color=black):

end do:

plots[display]([seq(pl[N], N in Ns)]);
```

This experiment nicely illustrates that interpolation of the Runge example function on equally spaced nodes is a bad idea. But, really, it is the nodes that are bad.

> Generations of textbooks have warned readers that polynomial interpolation is dangerous. In fact, if the interpolation points are clustered and a stable algorithm is used, it is bulletproof.
>
> — L.N. Trefethen [21]

For a full explanation of the Runge phenomenon, see Chapter 13 of [21].

The Runge Example with Chebyshev Nodes. If instead we use $x_k = \cos(\pi k/n)$, replacing the line

```
    tau := [seq(-1 + 2*k/N, k=0..N)];
```

with

```
 tau := [seq(evalf[2*Digits](cos(Pi*k/N)), k=0..N)];
```

then $B(x)$ climbs no higher than about 2. Indeed, we can replace `plots[logplot]` by just `plot`. See Figure 2.

This is an improvement, for $n = 89$, of a factor of about 10^{22}. For a detailed exposition of why this works, and when, see [6] and the Chebfun project at www.chebfun.org.

Concluding Remarks on the Runge Example. An instructor of numerical analysis has to walk a tightrope: the students need to be taught caution (maybe bordering

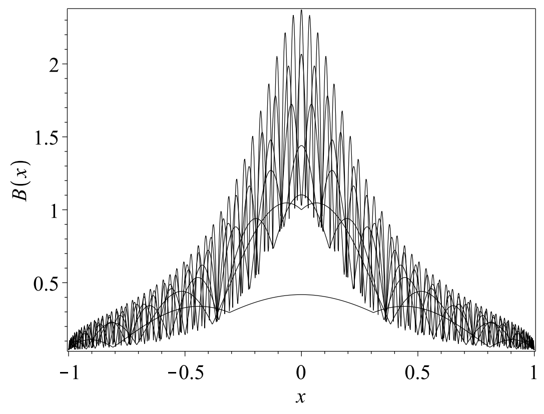

Fig. 2 *The Runge example with Chebyshev nodes with degrees* $n = 5, 8, 13, 21, 34, 55,$ *and* 89.

on paranoia), but they also need to learn when to trust their results. Learning to assess the sensitivity of their expression (as programmed) to realistic changes in data values is an important objective. The Runge example is a very clear case where these ideas can be usefully and thoroughly explored.

One can go further and replace $B(x)$ by its upper bound in terms of the Lebesgue function $B(x) \leq L(x)||c||_\infty$, where $L(x) = \sum_{k=0}^{n} |\phi_k(x)|^k$. This more general analysis is useful, as in [21], but in our opinion it loses the chance to make a special retrospective diagnostic of the problem at hand. Moreover, there are cases where $B(x) \ll L(x)||c||_\infty$ and this overestimation could lead to the wrong conclusion.

The bad behavior of the Runge example shows up in other ways, notably in the ill-conditioning of the Vandermonde matrix on those nodes. However, the Vandermonde matrix is ill-conditioned on Chebyshev nodes too [3], so that can't be the whole story. The explanation offered here seems more apt.

Wilkinson's First Polynomial. Let us now consider rootfinding. Suppose r is a simple zero of $p(x)$; that is, $p'(x) \neq 0$ and

$$0 = p(r) = \sum_{k=0}^{n} c_k \phi_k(r) \,. \tag{14}$$

Suppose $r + \Delta r$ is the corresponding zero of $p + \Delta p$. This really only makes sense if Δp is sufficiently small. Otherwise, the roots get mixed up. Then

$$0 = (p + \Delta p)(r + \Delta r) = p(r + \Delta r) + \Delta p(r + \Delta r) \tag{15}$$

$$\approx p(r) + p'(r)\Delta r + \Delta p(r) + \mathcal{O}(\Delta^2) \tag{16}$$

to first order; since $p(r) = 0$, we also have

$$p'(r)\Delta r \approx -\Delta p(r) \tag{17}$$

or

$$|\Delta r| \approx \left| \frac{-\Delta p(r)}{p'(r)} \right| \leq \frac{B(r) \cdot \varepsilon}{|p'(r)|}, \tag{18}$$

where

$$B(r) = \sum_{k=0}^{n} |c_k||\phi_k(r)| \tag{19}$$

is the condition number as before. For nonzero roots, the number

$$A(r) = \left|\frac{rB(r)}{p'(r)}\right| \tag{20}$$

has $\left|\frac{\Delta r}{r}\right| \le A(r)\varepsilon$, giving a kind of mixed relative/absolute conditioning. This analysis can be made more rigorous by using "pseudozeros" as follows. Define, for given $w_k \ge 0$ not all zero,

$$\Lambda_\varepsilon(p) := \left\{ z \,:\, \exists\, \Delta c_k \text{ with } |\Delta c_k| \le w_k\varepsilon \text{ and } \sum_{k=0}^{n}(c_k + \Delta c_k)\phi_k(z) = 0 \right\} . \tag{21}$$

Normally, we take $w_k = |c_k|$, in which case we may write $\Delta c_k = c_k\delta_k$.

This is the set of all complex numbers that are zeros of "nearby" polynomials—nearby in the sense that we allow the coefficients to change. This definition is inconvenient to work with. Luckily, there is a useful theorem, which can be found, for instance, in [6, Theorem 5.3]; see also [13] and [2].

THEOREM 0.2. *Given weights $w_k \ge 0$, not all zero, and a basis $\phi_k(z)$, define the weighted ε-pseudozero set of $p(z)$ as in* (21)*. Suppose also that*

$$\delta p(z) = \sum_{k=0}^{n} \Delta c_k \phi_k(z).$$

Moreover, let

$$B(\lambda) = \sum_{k=0}^{n} w_k |\phi_k(\lambda)|.$$

Then the pseudozero set of $p(z)$ may be alternatively characterized as

$$\Lambda_\varepsilon(p) = \{z \,:\, |p(z)| \le B(z)\cdot\varepsilon\} = \left\{ z \,:\, \left|\frac{zp(z)}{p'(z)}\right| \le \left|\frac{zB(z)}{p'(z)}\right|\varepsilon \right\}. \tag{22}$$

This is again a condition number, the same as in (9) if $w_k = |c_k|$.
Wilkinson's first polynomial is, with $N = 20$,

$$W_N(x) = \prod_{k=1}^{N}(x-k) \tag{23}$$

$$= (x-1)(x-2)(x-3)\cdots(x-N) \,. \tag{24}$$

In this form, it is "bulletproof." However, if we are foolish enough to expand it into its expression in the monomial basis, namely,

$$W_N(x) = x^N - \frac{1}{2}N(N+1)x^{N-1} + \cdots + (-1)^N \cdot N! \tag{25}$$

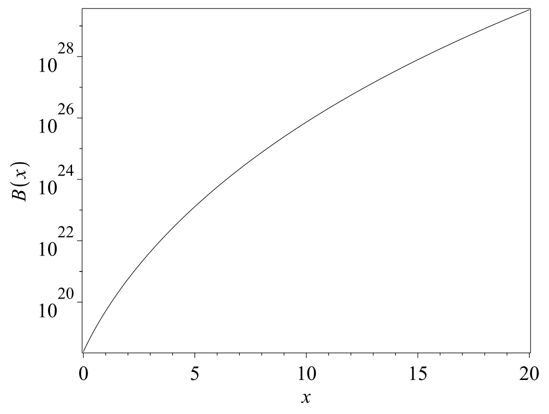

Fig. 3 *The condition number of Wilkinson's first polynomial ($N = 20$).*

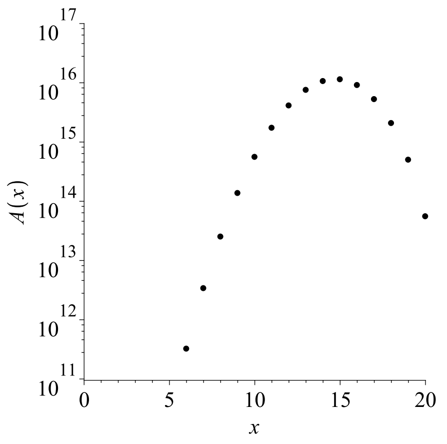

Fig. 4 *The condition number for rootfinding.*

(for $N = 20$ this is $x^{20} - 210x^{19} + \cdots + (20!)$) and in this basis, $\phi_k = x^k$, the condition number for evaluation

$$B_N(x) = |x|^N + \frac{1}{2}N(N+1)|x|^{N-1} + \cdots + |N!| \tag{26}$$

is very large. See Figure 3.

When we plot the condition number for rootfinding, $A(r) = |\frac{rB_N(r)}{W_N'(r)}|$, we find that for $N = 20$ (Wilkinson's original choice), the maximum value occurs at $r = 16$ and $rB_{20}(r)/|W_{20}'(r)| \approx 10^{16}$. See Figure 4.

Working in single precision would give no digits of accuracy; double precision (that is, $u \approx 10^{-16}$) also does not guarantee any accuracy. For $N = 30$ we sometimes find $\frac{rB_{30}(r)}{|W_{30}'(r)|} > 10^{21}$; for $N = 40$, it's 10^{28}. Working with the monomial basis for this polynomial is surprisingly difficult; Wilkinson himself was surprised, since the polynomial was intended to be a simple test problem for his program for the ACE computer. His investigations led to the modern theory of conditioning [22].

However, there's something a little unfair about the scaling: when taken to the 20th power, the interval $0 \le x \le 20$ covers quite a range of values. One wonders if matters can be improved by a simple change of variable.

The Scaled Wilkinson Polynomial. If we move the roots $1, 2, 3, \ldots, 20$ to the roots $-1+2k/21, k = 1, \ldots, 30$, then they become symmetrically placed in $-1 < x < 1$, and this improves matters quite dramatically, as we can see in Figure 5.

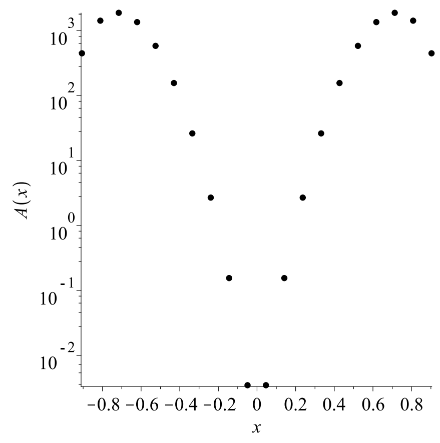

Fig. 5 *The condition number for the scaled Wilkinson polynomial,* $A(r) = \left|\frac{rB_N(r)}{W_N'(r)}\right|$.

The condition number 10^{13} becomes just 10^3, and we have to go to $N = 60$ to find condition numbers as high as 10^{13}. The scaling seems to matter. However, nearly all of the improvement comes from the symmetry; W_N will be even if N is even, and odd if N is odd, and this means that half the coefficients are zero and therefore not subject to (relative) perturbation.

If instead we scale to the interval $[0, 2]$ we have a different story: for roots $2-2k/21$ the condition number $B(x)$ reaches nearly the same heights as it did on $0 \le x \le 20$. See Figure 6. Similar results are obtained if we use $[0, 1]$. Thus, we conclude that symmetry matters. See Figure 7 for the pseudozeros of $W_N(x)$, where the contour levels are 10^{-14} and 10^{-18}. The roots are visibly changed by extremely tiny perturbations.

Wilkinson's Second Example Polynomial. The story of Wilkinson's second example is even stranger. The polynomial is

$$C_{20}(x) = \prod_{k=1}^{20} (x - 2^{-k}) \tag{27}$$

and the roots are $1/2, 1/4, 1/8, 1/16, \ldots, 1/2^{20}$. Wilkinson expected that the clustering of roots near zero would cause difficulty for his rootfinder, once the polynomial was expanded as

$$C_{20}(x) = x^{20} - \left(\sum_{k=1}^{20} \frac{1}{2^k}\right) x^{19} + \cdots + \prod_{k=1}^{20} 2^{-k} \,. \tag{28}$$

However, his program had no difficulty at all! This is because the monomial basis is, in fact, quite well-conditioned near $x = 0$, and the condition number for this polynomial can be seen in Figure 8 on $0 \le x \le 1$.

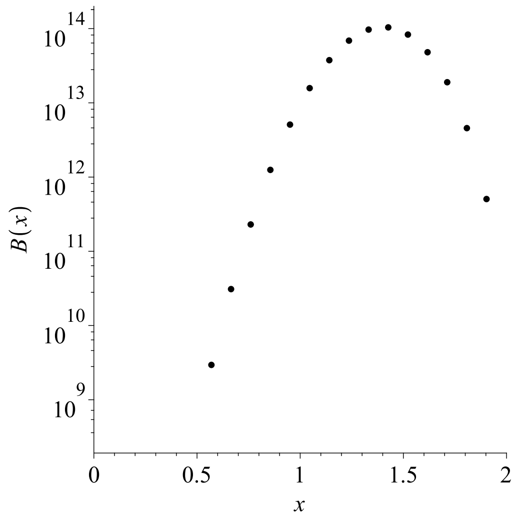

Fig. 6 *The condition number of the scaled Wilkinson polynomial.*

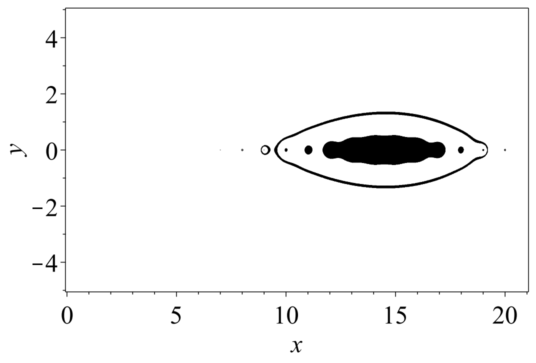

Fig. 7 *The pseudozeros of* $W_N(x)$. *The contour levels are* 10^{-14} *and* 10^{-18}. *The interior is blacked out because contours are difficult to draw at such sizes in floating-point arithmetic.*

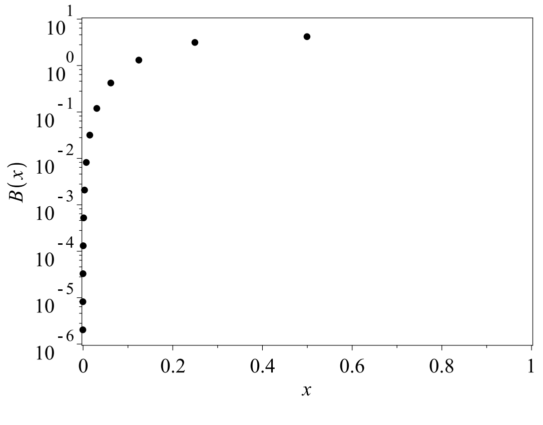

Fig. 8 *The condition number for Wilkinson's second example polynomial* (C_{20}). *In contrast to his first test problem, it is well-conditioned.*

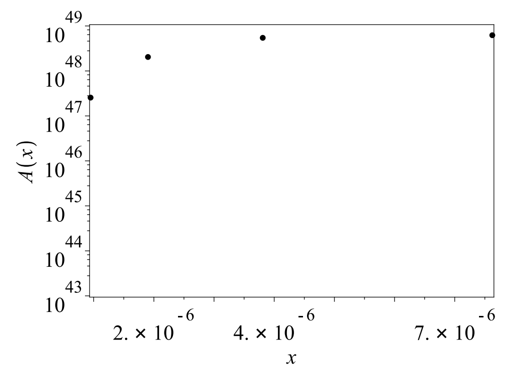

Fig. 9 *A portion of the condition number of C_{20} in the Lagrange basis on the nodes $k/20$, $0 \le k \le 20$.*

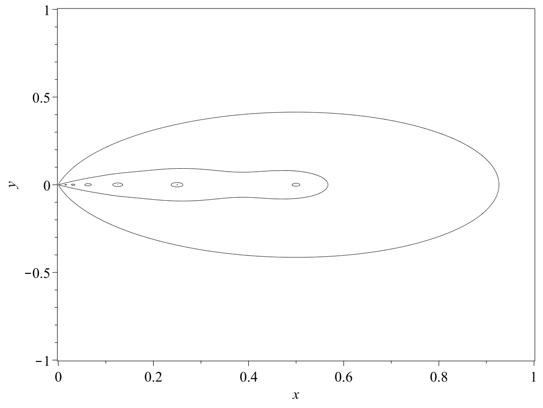

Fig. 10 *The pseudozeros of C_{20}. The contour levels are 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-6}, and 10^{-8}.*

In contrast, the condition number for evaluation using the Lagrange basis on equally spaced nodes in $[0, 1]$, plus either $x_0 = 0$ or $x_0 = 1$, is horrible: for $N = 20$ it is already 10^{48}; see Figure 9. This computation conforms to Wilkinson's intuition that things can go wrong if roots are clustered. Also, observe the pseudozeros of C_{20} in Figure 10. The perturbations required to make visible changes are quite large: these roots are not very sensitive to changes in the monomial basis coefficients. Another way to see this is to look at a problem where the roots are clustered at 1, rather than 0:

$$S_{20} = \prod_{k=1}^{N} \left(x - (1 - 2^{-k})\right) = \sum_{k=0}^{N} s_k x^k. \tag{29}$$

In this case, the condition number is presented in Figure 11 and is huge. This polynomial is very sensitive to changes in the monomial basis coefficients. The condition number for evaluation using a Lagrange basis for S_{20} is shown in Figure 12 (zoomed in for emphasis). Here the Lagrange basis is also very sensitive. Now consider the pseudozeros of S_{20} in Figure 13, in which contour levels are (from the outside in) 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, and 10^{-15}. In order to obtain a better view of the pseudozeros,

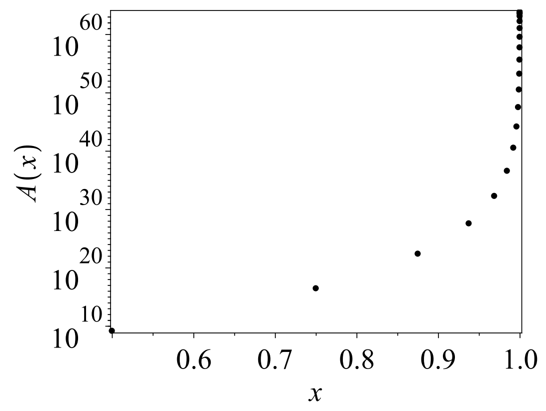

Fig. 11 *The condition number of* S_{20}.

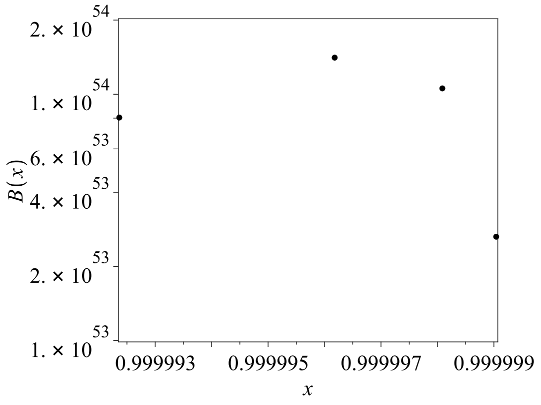

Fig. 12 *The condition number of* S_{20} *in a Lagrange basis.*

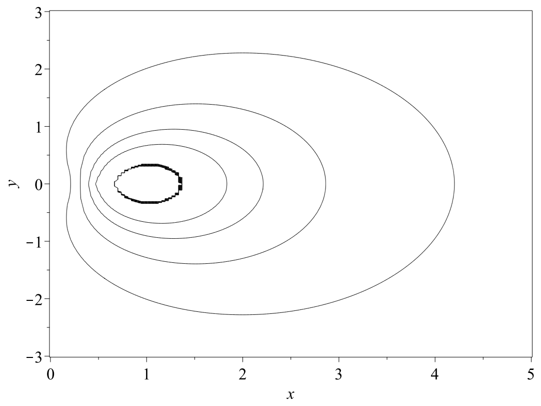

Fig. 13 *The pseudozeros of* S_{20}. *The contour levels are* 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, *and* 10^{-15}.

let's consider the first contour, 10^{-4}, which is the biggest curve in Figure 13. We know that

$$S_{20} + \Delta S = \sum_{k=0}^{20} s_k(1+\delta_k)x^k, \tag{30}$$

where $\Delta S = s_0\delta_0 + s_1\delta_1 x + s_2\delta_2 x^2 + \cdots + s_{20}\delta_{20}x^{20}$. Now if we choose a point between contour levels 10^{-4} and 10^{-6}, for example, $p = 3 - 1.5i$, we can see that p is a zero of some $S_{20} + \Delta S(x)$ where all coefficients of ΔS have $|\delta_k| < 10^{-4}$. These are all small relative perturbations, which means everything inside the contour level 10^{-4} is a zero of a polynomial that is reasonably close to S_{20}. This is somehow backward error. Therefore, everything inside the contour level 10^{-4} is a zero of a polynomial closer to S_{20} (in this sense) than 10^{-4}. Everything inside the contour level 10^{-6} is a zero of a polynomial closer that 10^{-6} to S_{20}, and so on. Notice that the innermost contour, corresponding to 10^{-15}, is visible to the eye. This means that trivial (unit roundoff level in double precision) changes in the coefficients lead to visible changes in the root.

Concluding Remarks on the Wilkinson Rootfinding Examples. The first example polynomial, $\prod_{k=1}^{20}(x-k)$, is nearly universally known as a surprising example. Yet there are very few places where one sees an elementary exposition of Wilkinson's theory of conditioning using this example, which is itself surprising because the theory was essentially born from it. We have illustrated here Wilkinson's theory, as refined by Farouki and Rajan, for students.

> For accidental historical reasons therefore backward error analysis is always introduced in connexion with matrix problems. In my opinion the ideas involved are much more readily absorbed if they are presented in connexion with polynomial equations. Perhaps the fairest comment would be that polynomial equations narrowly missed serving once again in their historical didactic role and rounding error analysis would have developed in a more satisfactory way if they had not.
>
> — James H. Wilkinson [22]

A Final Word for the Instructor. Backward error analysis is difficult at first for some students. The conceptual problem is that people are trained to think of mathematical problems as being exact; some indeed are, but many come from physical situations and are only models with uncertain data. The success of backward error analysis for floating point is to put rounding errors on the same footing as data or modeling errors, which have to be studied anyway. This is true even if the equations are solved exactly, using computer algebra! The conditioning theory for polynomials discussed here allows this to be done quite flexibly, and is a useful part of the analyst's repertoire. Students need to know this.

REFERENCES

[1] F. Acton, *Numerical Methods that Work*, MAA Spectrum, Mathematical Association of America, 1990. (Cited on p. 231)

[2] A. Amiraslani, *New Algorithms for Matrices, Polynomials and Matrix Polynomials*, Ph.D. thesis, Western University, Ontario, Canada, 2006. (Cited on p. 236)

[3] B. Beckermann, *The condition number of real Vandermonde, Krylov and positive definite Hankel matrices*, Numer. Math., 85 (2000), pp. 553–577. (Cited on p. 235)

[4] R. Burden, D. Faires, and A. Burden, *Numerical Analysis*, 10th ed., Cengage Learning, Boston, MA, 2016. (Cited on p. 231)

[5] J. Carnicer, Y. Khiar, and J. Peña, *Optimal stability of the Lagrange formula and conditioning of the Newton formula*, J. Approx. Theory, 238 (2019), pp. 52–66. (Cited on p. 232)

[6] R. M. Corless and N. Fillion, *A Graduate Introduction to Numerical Methods: From the Viewpoint of Backward Error Analysis*, Springer, 2013, https://doi.org/10.1007/978-1-4614-8453-0. (Cited on pp. 233, 234, 236)

[7] R. M. Corless and S. M. Watt, *Bernstein bases are optimal, but, sometimes, Lagrange bases are better*, in Proceedings of SYNASC, Timisoara, MIRTON Press, 2004, pp. 141–153. (Cited on p. 232)

[8] T. A. Driscoll and R. J. Braun, *Fundamentals of Numerical Computation*, SIAM, 2018. (Cited on p. 231)

[9] R. Farouki and T. Goodman, *On the optimal stability of the Bernstein basis*, Math. Comp., 65 (1996), pp. 1553–1566. (Cited on p. 232)

[10] R. Farouki and V. Rajan, *On the numerical condition of polynomials in Bernstein form*, Comput. Aided Geom. Design, 4 (1987), pp. 191–216. (Cited on p. 232)

[11] W. Gautschi, *Numerical Analysis*, Birkhäuser Boston, 2011. (Cited on p. 231)

[12] T. Gowers, J. Barrow-Green, and I. Leader, *The Princeton Companion to Mathematics*, Princeton University Press, 2010. (Cited on p. 231)

[13] K. Green and T. Wagenknecht, *Pseudospectra and delay differential equations*, J. Comput. Appl. Math., 196 (2006), pp. 567–578. (Cited on p. 236)

[14] R. Hamming, *Numerical Methods for Scientists and Engineers*, Dover, 2012. (Cited on p. 231)

[15] P. Henrici, *Elements of Numerical Analysis*, Tech. report, 1964. (Cited on p. 231)

[16] P. Henrici, *Essentials of Numerical Analysis with Pocket Calculator Demonstrations*, John Wiley & Sons, 1982. (Cited on p. 231)

[17] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, 2002, https://doi.org/10.1137/1.9780898718027. (Cited on pp. 231, 233)

[18] N. J. Higham and T. Mary, *A new approach to probabilistic rounding error analysis*, SIAM J. Sci. Comput., 41 (2019), pp. A2815–A2835, https://doi.org/10.1137/18M1226312. (Cited on p. 233)

[19] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice-Hall, 1989. (Cited on p. 231)

[20] T. Sauer, *Numerical Analysis*, Pearson, 2011. (Cited on p. 231)

[21] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, 2013. (Cited on pp. 234, 235)

[22] J. H. Wilkinson, *The perfidious polynomial*, in Studies in Numerical Analysis, MAA Stud. Math. 24, MAA, 1984, pp. 1–28. (Cited on pp. 231, 237, 242)