Convergence analysis of an iterated-eigenvalue polynomial root-finding algorithm

Steven Fortune
Bell Laboratories
`sjf@research.bell-labs.com`

Abstract

We provide a partial convergence analysis of an iterative polynomial root-finding algorithm [7, 14]. The iteration in the algorithm is based upon floating-point eigenvalue computation. The algorithm is quite efficient, particularly for approximating the zeroes of ill-conditioned, high-degree polynomials. A previous analysis of the algorithm [7] depended upon a technical assumption, that eigenvalue computation is componentwise backwards stable for a particular class of matrices. We present a new analysis that eliminates this assumption, using instead the usual normwise backwards stability of eigenvalue computation and certain additional genericity conditions.

1 Introduction

The algorithmic problem of approximating the roots of a univariate polynomial, presented by its coefficients, is classic in numerical analysis and symbolic algebra [15, 16]. Existing algorithms can be crudely and unexhaustively classified either as iterative algorithms (Newton's method, the Durand-Kerner method, Aberth's method [2, 16]) or as theoretically complete algorithms (Weyl's algorithm and the Schönhage splitting-circle algorithm [16]). The iterative algorithms usually have local convergence proofs, but are not known to converge globally and hence have no running time analysis.

In practice the iterative algorithms can perform very well. Bini and Fiorentino [2, 3] compared their implementation `mpsolve`, based on Aberth's iterative method, with the solvers in Maple$^{\text{TM}}$ and Mathematica$^{\text{TM}}$ and with Gourdon's implementation [9] of the Schönhage splitting-circle algorithm. Their implementation is much the fastest, beating the other implementations by factors ranging from roughly 10 to 5000, depending upon instance.

An alternative iterative algorithm `eigensolve` [7] is based on floating-point eigenvalue computation. (A version of the algorithm is also described in [14].) At a high level, the algorithm is very simple (see figure 3). Let P be a polynomial and S the current estimate of the roots of P, represented as floating-point numbers. The iteration is to compute the eigenvalues, in hardware floating-point, of a generalized companion matrix $C(P,S)$; these computed eigenvalues then form the next estimate of the roots of P. The matrix $C(P,S)$ is an eigenvalue formulation of Lagrange interpolation. This matrix is familiar; a standard way to find inclusion regions that are guaranteed to contain the roots of P is to apply Gershgorin's theorem to this matrix [2, 11]. Computation of $C(P,S)$ needs the values of P at the points of S to floating-point precision, requiring extended precision arithmetic if evaluation of P is ill-conditioned. Figure 4 documents the performance of `eigensolve`; compared with `mpsolve`, it is particularly efficient for high-degree, ill-conditioned polynomials.

It is not obvious that this iteration converges—indeed it might seem highly implausible, since the heart of the iteration is a floating-point eigenvalue computation on a matrix whose (eigenvalue) condition number may be vastly larger than $1/\epsilon$, ϵ the relative accuracy of hardware floating-point arithmetic. Figure 1 depicts the behavior of `eigensolve` on the Wilkinson polynomial of degree 60.

Why does the iteration converge? For a vector $S = (s_1, \ldots, s_n)$, $s_i \in \mathbb{C}$, let $Q_S(x) = \prod(x - s_i)$, and let $S_1, S_2, \ldots,$ be the successive iterate vectors produced by `eigensolve`. For a polynomial P such as a Wilkinson polynomial, it can be observed experimentally that for each root λ of P,

$$|Q_{S_{t+1}}(\lambda)| \lesssim \epsilon \max(|Q_{S_t}(\lambda)|, |\lambda P'(\lambda)|) \tag{1}$$

(see figure 2). Here ϵ is the relative error of floating-point arithmetic ($\epsilon \approx 10^{-16}$ for IEEE double-precision floating-point). The condition $|Q_{S_k}(\lambda)| \approx |\epsilon \lambda P'(\lambda)|$, for all roots λ, is sufficient to guarantee that the iterate S_k approximates all the roots of P to relative error about ϵ [7]. It follows that the number of required iterations is bounded by about $\log_{(1/\epsilon)} \chi(P)$, where $\chi(P)$ is the condition number of the polynomial [7].

Previous analysis [7] proved a version of equation (1), with an additional factor of $q(n, 1/\rho, \tau)$ in the upper bound, where q is a polynomial in the degree n of P, ρ measures the relative separation of the roots of P, and τ measures certain distribution characteristics of the iterate S_i. The proof required a technical assumption about floating-point eigenvalue computation. Floating-point eigenvalue computation is backwards stable [8], that is, the computed eigenvalues for a matrix M are the true eigenvalues for a matrix $M + \Delta M$, where $||\Delta M|| \leq \epsilon\, t(n)\, ||M||$ and $t(n)$ is a slowly-growing function of the matrix size n. The previous proof required the additional assumption that eigenvalue computation be componentwise backwards stable, that is, $|\Delta M_{ij}| \leq \epsilon\, t(n)\, |M_{ij}|$, for the very structured matrices used in the algorithm.

The contribution of this paper is to give a proof of (1), this time without the assumption of componentwise backwards stability, though again with an additional factor in the upper bound. The additional factor depends upon n and ρ as before and also a parameter α that measures the genericity of the iterate S_i (see section 3.1). S_i is the vector of roots of $Q_{S_i} = \mathrm{char}(M + \Delta M) = P + \Delta P$, where $\mathrm{char}\, M = P$. The perturbation matrix ΔM, and hence the perturbation polynomial ΔP, are consequences of floating-point rounding during the eigenvalue computation. The genericity of S_i is determined from P and ΔP; roughly, α is large if ΔP is "exceptional" and $\alpha \approx 1$ if ΔP is "random" or "unexceptional". All experimental observations are that $\alpha \approx 1$. Section 3.3 below argues why it is plausible that α should be small, though it is far from clear how to prove a rigorous upper bound on α.

A complete and efficient implementation of the iterated-eigenvalue algorithm requires careful attention to many details. These include heuristics for coping with roots that have are clustered, repreated, or have widely varying magnitudes. After roots have been isolated, Newton iteration can be used to approximate roots to arbitrary precision. Adaptive-precision arithmetic is required for efficient evaluation of a polynomial at a root estimate s_i. Some discussion of these details can be found in the previous paper [7]; information on `eigensolve` can be found at the web site `http://cm.bell-labs.com/who/sjf/eigensolve.html`.

2 Background

Let P be a polynomial of degree n. If P has simple root λ then the *root separation* of P at λ is $\mathrm{sep}(P, \lambda) = \min_{\lambda'} |\lambda - \lambda'|$ where λ' varies over all roots distinct from λ. If λ is a simple root of P and $\alpha > 0$, s is *α-close* to λ if $|s - \lambda| \leq \alpha \min_{\lambda' \neq \lambda} |\lambda - \lambda'|$, where λ' varies over all roots of P distinct from λ.

Proposition 2.1 *If s is $1/n$-close to root λ of P, then $|P(s)| \leq e\, |s - \lambda|\; |P'(\lambda)|$.*

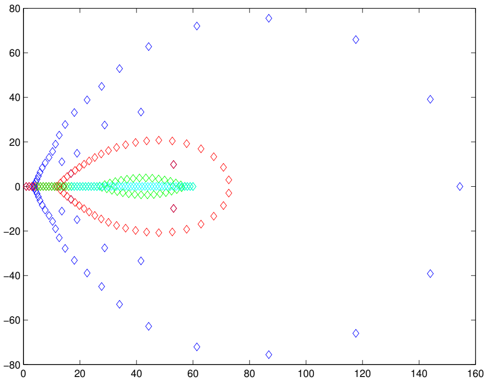

Figure 1: Iterates of the algorithm on the Wilkinson polynomial of degree 60.

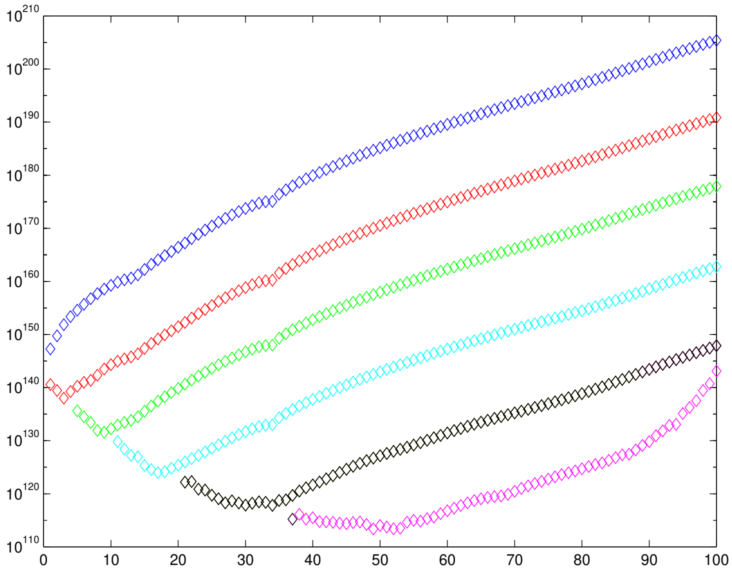

Figure 2: The x-axis are the roots 1..100 of the Wilkinson polynomial W_{100} of degree 100; dots are the values of $Q_i(\lambda)$, i varying over the iteration. The lower envelope of the dots closely matches $\epsilon|\lambda W'_{100}(\lambda)|$.

Proof: Suppose $P = \prod (x-\lambda_i)$ and s is α-close to λ_1. Then for $i \neq 1$, $|s-\lambda_i| = |s-\lambda_1 + \lambda_1-\lambda_i| \leq (1+\alpha)\,|\lambda_1-\lambda_i|$ and

$$|P(s)| \;=\; |(s-\lambda_1)| \prod_{i\neq 1} |s-\lambda_i| \;\leq\; |s-\lambda_1|\; (1+\alpha)^{n-1} |P'(\lambda_1)|.$$

The lemma follows for $\alpha = 1/n$ using $(1+1/n)^{n-1} \leq e$. □

Proposition 2.2 *Suppose λ is the root of P closest to s, and $|s-t| < |s-\lambda|/(n+1)$. Then $|P(t)|/e \leq |P(s)| \leq e\,|P(t)|$.*

Proof: Similar to the proof of proposition 2.1 □

For $S = (s_1, \ldots, s_k)$ a vector of complex numbers, define

$$Q_S(x) = \prod_{i=1}^{k} (x-s_i).$$

Notice that $Q'(s_j) = \prod_{i\neq j} (s_j - s_i)$.

For M a matrix, $\lambda(M)$ denotes the vector of eigenvalues of M, $\det M$ the determinant of M, and $\operatorname{char} M$ the characteristic polynomial of M. For monic $P = x^n + p_{n-1}x^{n-1} + \ldots + p_1 x + p_0$, the *companion matrix* is

$$C(P) = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ & \vdots & & \ddots & \\ -p_0 & -p_1 & -p_2 & \ldots & -p_{n-1} \end{bmatrix}.$$

Standard results are that $\lambda(C(P))$ are the roots of P and that $\lambda(M)$ are the roots of $\operatorname{char} M$.

2.1 Some matrix facts

Throughout this section, $D = \operatorname{diag}(d_1, \ldots, d_n)$ is a diagonal matrix, $R = (r_{ij})$ is a matrix of rank one, $\det_{ij} M$ is the determinant of the matrix obtained from M by deleting the ith row and the jth column, and similarly for $\operatorname{char}_{ij} M$. For the proofs of the following lemmas, see [7].

Lemma 2.3 $\det(D+R) = \prod_{i=1}^{n} d_i + \sum_{i=1}^{n} r_{ii} \prod_{j\neq i} d_j.$

Corollary 2.4 $(-1)^n \operatorname{char}(D+R) \;=\; \prod_{i=1}^{n} (x-d_i) \;+\; \sum_{i=1}^{n} r_{ii} \prod_{j\neq i} (x-d_i).$

Lemma 2.5

$$\det_{ij}(D+R) = \begin{cases} r_{ji} \prod_{k\neq i,j} d_k & i \neq j \\[2ex] \prod_{k\neq i} d_k + \sum_{l\neq i} r_{ll} \prod_{k\neq i,l} d_k & i = j. \end{cases}$$

Corollary 2.6

$$(-1)^{n-1}\mathrm{char}_{ij}\,(D{+}R) = \begin{cases} r_{ji} \prod\limits_{k\neq i,j} (x{-}d_k) & i \neq j \\[2ex] \prod\limits_{k\neq i}(x{-}d_k) + \sum\limits_{l\neq i} r_{ll} \prod\limits_{k\neq i,l} (x{-}d_k) & i = j. \end{cases}$$

2.2 Generalized companion matrices

Let $S = (s_1, \ldots, s_n)$ with s_i distinct complex numbers, and let P be a monic polynomial of degree n. The *Lagrange coefficients* $l_1, \ldots, l_n$ for P and S are defined by

$$l_i = \left| \frac{P(s_i)}{Q'_S(s_i)} \right|.$$

The *generalized companion matrix* $C(P, S)$ is the $n \times n$ matrix

$$C(P,S) \;=\; \begin{bmatrix} s_1 & 0 & \ldots & 0 \\ 0 & s_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & s_n \end{bmatrix} - \begin{bmatrix} l_1 & l_2 & \ldots & l_n \\ l_1 & l_2 & \ldots & l_n \\ \vdots & \vdots & & \vdots \\ l_1 & l_2 & \ldots & l_n \end{bmatrix}$$

By Corollary 2.4,

$$(-1)^n \,\mathrm{char}\, C(P,S) = \prod (x{-}s_i) \;+\; \sum_{i=1}^{n} l_i \prod_{j\neq i} (x{-}s_j).$$

The second term on the righthand side of this equation is the Lagrange interpolating polynomial [20] of degree $n{-}1$ that has value $P(s_i)$ at s_i. Since this polynomial is unique, and P is monic of degree of n, we have a result apparently first proved by Smith [11].

Corollary 2.7 *If P is monic, then $(-1)^n \,\mathrm{char}(C(P,S)) = P$; hence the eigenvalues of $C(P,S)$ are the roots of P.*

For complex s and real $r{>}0$, let $D(s,r) \subset \mathbb{C}$ be the closed disc of radius r centered at s. The following lemma follows easily from Gershgorin's theorem [8] and corollary 2.7 (see also [2, 6, 11]).

Lemma 2.8 *Let P be a monic polynomial of degree n, $S = (s_1, \ldots, s_n)$ be distinct complex numbers and $l_1, \ldots, l_n$ be the Lagrange coefficients. Then the union of discs $\bigcup_{i=1}^{n} D(s_i, nl_i)$ contains all the roots of P. Furthermore, each connected component made up of k discs contains k roots. In particular, if the discs are pairwise disjoint, each disc contains exactly one root.*

2.3 Floating-point eigenvalue computation

The computation of eigenvalues has been a central topic of numerical analysis for several decades. The QR algorithm[4, 8] is the standard for dense matrices. The conventional analysis is that the QR algorithm is

```
// P is a monic polynomial of degree n
// S = (s_1, ..., s_n) is a vector of length n of floating-point complex numbers
// C(P) and C(P,S) are companion matrices
// fl is conversion to floating-point
// eig is floating-point eigenvalue calculation (section 2.3)
procedure eigensolve (P)
    S := eig(fl(C(P)))
    while ¬converged(S)
            S := eig(fl(C(P,S)))
    return S
```

Figure 3: The iterated eigenvalue algorithm.

backwards stable: for M an $n \times n$ matrix with floating-point entries, a call to the implementation `eig` of the QR algorithm

$$\Lambda := \texttt{eig}(M)$$

computes a vector of eigenvalues $\Lambda = (\lambda_1, \ldots, \lambda_n)$ that are the true eigenvalues of a perturbed matrix $M + \Delta M$:

$$\begin{aligned} \Lambda &= \lambda(M + \Delta M) \\ ||\Delta M|| &\leq t(n)\,\epsilon\,||M||, \end{aligned}$$

where ϵ is the relative error of floating-point arithmetic. The upper bound $t(n)$ is usually quoted as a slow-growing function of n, say $t(n) \leq 10n$ [1]. Apparently, there is no known rigorous convergence proof of the QR algorithm. Assuming that the QR algorithm requires only $O(n)$ iterations, $t(n)$ can be rigorously bounded by $O(n^3)$ [17]. We assume that the QR algorithm is globally convergent and that $t(n)$ is bounded by a polynomial in n.

3 The algorithm and its analysis

For this section we assume that P is monic of degree n, with distinct nonzero roots $\lambda_1, \ldots, \lambda_n$. The algorithm for root finding is given in figure 3. Lemma 2.8 provides a possible basis for a convergence test; more sophisticated strategies are discussed in [7].

3.1 Generic root estimates

Let $S = \{s_1, \ldots, s_n\}$, $s_i \in \mathbb{C}$. Let $\lambda_{\sigma(i)}$ be the root of P closest to s_i, and let $G_S = Q_S - P$. For real $\alpha > 0$, S is *α-generic* (with respect to P) if the following four conditions hold:

1. If $i \neq j$, then $s_i \neq s_j$.
2. $\max_i |s_i| \;\leq\; \alpha \max_i |\lambda_i|$.
3. $|l_i| \;\leq\; \alpha\,|s_i - \lambda_{\sigma(i)}|$, $i = 1, \ldots, n$, where l_i is the Lagrange coefficient as in section 2.2.

4. If s_i is $1/n$-close to $\lambda_{\sigma(i)}$, then $|G_S(\lambda_{\sigma(i)})| \;\leq\; \alpha\;|G_S(s_i)|$, $i = 1, \ldots, n$.

Our interest is in the case that S is the result of a floating-point eigenvalue computation on a matrix whose characteristic polynomial is P. In this case, $||G_S||$ should be small with respect to $||P||$; Section 3.3 below gives nonrigorous plausibility arguments that S should be α-generic for some small α.

Proposition 3.1 *Suppose S is α-generic and s_i is $1/n$-close to λ_k. Then $|Q_S(\lambda_k)| \leq \alpha\;|P(s_i)|$.*

Proof: Using condition (4), $|Q_S(\lambda_k)| = |G_S(\lambda_k)| < \alpha\;|G_S(s_i)| = \alpha\;|P(s)|$. □

3.2 Analysis

Let S_0 be the values computed by the initial call `eig(fl(`$C(P)$`))`, and for $t = 1, 2, \ldots$, let S_t be the values computed at the tth call to `eig` within the `while` loop. With

$$S_t = \{s_{t1}, \ldots, s_{tn}\}$$

define

$$Q_t(x) = Q_{S_t}(x) = \prod_{i=1}^{n}(x - s_{ti}).$$

Lemma 3.2 *There is a polynomial q in n and α so that if S_t is α-generic, for every root λ of P,*

$$|Q_{t+1}(\lambda)| \;\leq\; \epsilon\; q(n,\alpha) \max\left(\frac{|\lambda_{\max}|}{\mathrm{sep}(P,\lambda)}|Q_t(\lambda)|,\; |\lambda_{\max} P'(\lambda)|\right),$$

where $\lambda_{\max}$ is the root of P of largest magnitude.

Proof of Lemma 3.2: Consider a root λ_k of P. For simplicity drop the double subscripts and write $S_t = (s_1, \ldots, s_n)$. Let $M = C(P, S_t)$. Recall that $M_{ij} = -l_j$ if $i \neq j$, and $M_{jj} = s_j - l_j$, where $l_j = P(s_j)/Q_t'(s_j)$ is the Lagrange coefficient.

Using $|l_j| \leq \alpha\,|s_j - \lambda_{\sigma(j)}|$, $\lambda_{\sigma(j)}$ the root closest to s_j, it is easy to get

$$||M|| \leq 2n\alpha^2\;|\lambda_{\max}|. \tag{2}$$

where $||\cdot||$ (here and elsewhere) is the ∞-norm. By the assumptions on `eig`, there is a perturbation matrix Δ so that

$$\begin{aligned} |\,\Delta_{ij}\,| &\leq \epsilon\, n^{O(1)}\,||M|| \qquad (3)\\ S_{t+1} &= \lambda(M + \Delta)\\ Q_{t+1}(x) &= \mathrm{char}(M + \Delta). \end{aligned}$$

By expanding the determinant of $\mathrm{char}(M + \Delta)$ and letting $O(\Delta^2)$ denote the sum of all terms with at least two factors from Δ, we get

$$Q_{t+1}(x) \;=\; \mathrm{char}\, M \;+\; \sum_{i,j}(-1)^{i+j}\Delta_{ij}\,\mathrm{char}_{ij}\, M \;+\; O(\Delta^2);$$

of course $\operatorname{char} M = P$. We henceforth ignore the $O(\Delta^2)$ terms; this can be justified rigorously by a somewhat tedious argument similar to the argument below. Substituting λ_k for x, we get

$$|Q_{t+1}(\lambda_k)| \;\leq\; \sum_{i,j} |\Delta_{ij}\ (\operatorname{char}_{ij} M)(\lambda_k)|. \tag{4}$$

We first consider the case $i \neq j$. By corollary 2.6 and using $M_{ij} = -l_j$,

$$\begin{aligned} \operatorname{char}_{ij} M &= (-1)^n\, l_j \prod_{p\neq i,j} (x - s_p) \\ \Big|\,(\operatorname{char}_{ij} M)(\lambda_k)\,\Big| &= \Big|\,\frac{l_j}{\lambda_k - s_j} \prod_{p\neq i} (\lambda_k - s_p)\,\Big|. \end{aligned} \tag{5}$$

Since S_t is α-generic, $|l_j/(\lambda_k - s_j)| \leq \alpha$. There are now two subcases. First, if s_i is $1/n$-close to λ_k, then using propositions 3.1 and 2.1 followed by (2), (3) and (5)

$$\begin{aligned} \Big|\prod_{p\neq i}(\lambda_k - s_p)\Big| &= \left|\frac{Q_S(\lambda_k)}{\lambda_k - s_i}\right| \;\leq\; \alpha \left|\frac{P(S)}{\lambda_k - s_i}\right| \;\leq\; \alpha e |P'(\lambda_k)| \\ |\Delta_{ij}\ (\operatorname{char}_{ij} M)(\lambda_k)| &\leq\; \epsilon\, \alpha^3\, n^{O(1)}\, |\lambda_{\max}\ P'(\lambda_k)| \end{aligned} \tag{6}$$

Second, if s_i is not $1/n$-close to λ_k, then easily $|\lambda_{\max}| \leq n|s_i - \lambda_k|/\operatorname{sep}(P, \lambda_k)$. Again using (2), (3), and (5)

$$\begin{aligned} |\Delta_{ij}| &\leq\; \epsilon\, n^{O(1)} \left|\frac{s_i - \lambda_k}{\operatorname{sep}(P, \lambda_k)}\right| \\ |(\Delta_{ij}\ \operatorname{char}_{ij} M)(\lambda_k)| &\leq\; \epsilon\, \alpha^2\, n^{O(1)} \left|\frac{Q_t(\lambda_k)}{\operatorname{sep}(P, \lambda_k)}\right|. \end{aligned} \tag{7}$$

Now suppose $i = j$. By lemma 2.6

$$\begin{aligned} (-1)^{n-1} \operatorname{char}_{ii} M &= \prod_{j\neq i}(x - s_j) \;+ \sum_{p\neq i} l_p \prod_{j\neq i,p} (x - s_j) \\ &= \Big(1 + \sum_{p\neq i} \frac{l_p}{x - s_p}\Big) \prod_{j\neq i}(x - s_j). \end{aligned}$$

The analysis in this case is essentially identical to the analysis following equation (5) above. Lemma 3.2 now follows from (4), (6), (7), and the corresponding bounds in the case $i = j$. □

3.3 The genericity assumption

Each iterate $S = \{s_1, \ldots, s_n\}$ is the result of a floating-point eigenvalue computation on a matrix M so that $\operatorname{char} M = P$. Hence $Q_S = \operatorname{char}(M + \Delta M)$ for some perturbation matrix ΔM determined by the rounding during the floating point computation. Let $G_S = Q_S - P$. It is plausible that G_S should be both "generic," in the sense that it should have no unusual properties, and "tiny" in norm relative to the norm of P. We give plausibility arguments why each iterate S is α-generic, for some small $\alpha > 0$, given that G_S is "generic" and "tiny". We remark that it is far from obvious how to make these plausibility arguments rigorous. Each case below corresponds to a case in the definition in section 3.1.

(1) The manifold of nonsimple polynomials has codimension 1 in the space of polynomials. Hence for almost all choices of G_S, $P{+}G_S$ is simple.

(2) By standard results on containment discs for roots of polynomials [13, Section 6.4], for any polynomial R of degree $n > 0$ there is a real $r(R) > 0$ so that the annulus $A(R) = \{x \in \mathbb{C} : r(R) \geq |x| \geq r(R)/(ne)\}$ contains the root of largest magnitude of R. The value $r(R)$ is given by a continuous and well-conditioned function of the coefficients of R. As long as G_S is actually a coefficientwise small perturbation of P (i.e. the ith coefficient of G_S is small relative to the ith coefficient of P), then the annuli $A(P)$ and $A(P{+}G_S)$ are similar, and $\max_i |s_i| \lesssim ne \max_i |\lambda_i|$. Unfortunately, error analysis only guarantees that $||\Delta M||$ is small relative to $||M||$. There is some theoretical work relating $||\Delta M||$ and $||G_S||$ [5, 19]. Furthermore, experimental work by Toh and Trefethen [18] suggests that eigenvalue computation of the ordinary companion matrix $C(P)$ is indeed coefficientwise backwards stable, at least if the companion matrix is balanced [8] before the floating-point eigenvalue computation.

(3) The Lagrange coefficient l_i satisfies $l_i = P(s_i)/Q'_S(s_i) = P(s_i)/(P'(s_i){+}G'_S(s_i))$. If G_S is "generic", then so is G'_S. Hence $P'(s_i)$ and $G'_S(s_i)$ are uncorrelated, and $|l_i| \lesssim |P(s_i)/P'(s_i)|$. Let λ_k be the root of P closest to s_i (the argument works for any root). Let $t_0{=}0$, t_1, $\ldots$, t_n be the coefficients of the Taylor expansion of P about λ_k. For $\Delta = s_i - \lambda_k$, a little manipulation yields

$$\frac{P(s_i)}{P'(s_i)} = \Delta \frac{t_1 + t_2\Delta + \ldots + t_n\Delta^{n-1}}{t_1 + 2t_2\Delta + \ldots + (n{-}1)t_n\Delta^{n-1}} = \Delta \frac{T_1(\Delta)}{T_2(\Delta)}.$$

Since G_S is "generic" and s_i is a root of $P{+}G_S$, Δ is "generic," and $|T_2(\Delta)| \approx |T_2|(|\Delta|)$, where $|T_2|$ is the polynomial obtained from T_2 by replacing each coefficient by its absolute value. Hence $|T_1(\Delta)/T_2(\Delta)| \lesssim 1$, and $|l_i| \lesssim |\Delta|$.

(4) Let λ_k and γ be the roots of P and G_S, respectively, closest to s_i. As long as γ is not too close to s_i, in particular as long as $(n{+}1)|\lambda_k{-}s_i| < |\gamma - s_k|$, Proposition 2.2 guarantees $|G_S(\lambda_k)| < e\,|G_S(s_i)|$. (The condition that s_i is $1/n$-close to λ_k is not needed, except in the informal sense that $|\lambda_k{-}s_i|$ is small.)

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorenson, *LAPACK User's Guide*, Second Edition, SIAM, 1995.

[2] D.A. Bini, G. Fiorentino, Design, analysis, and implementation of a multiprecision polynomial rootfinder, *Numerical Algorithms*, **23:**127–173, 2000.

[3] D.A. Bini, G. Fiorentino, Numerical computation of polynomial roots using MPSolve, manuscript, 1999. Software and paper available for download from `ftp://ftp.dm.unipi.it/pub/mpsolve/`.

[4] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

[5] A. Edelman, H. Murakami, Polynomial roots from companion matrix eigenvalues, 1994. *Mathematics of Computation*, **64**:763–776, 1995.

[6] L. Elsner, A remark on simultaneous inclusions of the zeros of a polynomial by Gershgorin's theorem, *Numer. Math.*, **21**:425–427, 1973.

[7] S. Fortune, Polynomial root finding using iterated eigenvalue computation, to appear, *ISSAC 2001.*

Polynomial	deg	mpsolve	eigen	ratio
Wilkinson	40	1.1	.2	5.5
	80	12.3	1.9	6.5
	160	144.0	12.1	11.9
	320	2138.0	87.6	24.4
	640		1448.0	
	1000		5938.0	
Laguerre	80	5.1	1.6	3.2
	160	59.0	9.9	6.0
	320	806.0	86.4	9.3
	1000		5110.0	
Chebyshev	160	39.0	10.0	3.9
	320	450.0	86	5.2
Chromatic a	342	1150.0	67.8	17.0
Chromatic ad	340	1848.0	81.5	22.7
Chromatic cd	171	123.1	11.5	10.7
	683	$>$15750.0	610.5	$>$25.8
Hermite	160	20.4	8.1	2.5
	320	230.0	78.8	2.9
Mandelbrot	127	11.4	4.3	2.6
	255	134.0	30.9	4.3
	511	1874.0	246.6	7.6
	1023		3048.0	
Sendra	160	291.0	10.9	26.7
	320	5200	78.4	66.3
Exponential	50	.3	.5	.6
	100	3.8	2.5	1.5
	200	43.0	17.2	2.5
	400	484.7	125.4	3.9
Toeplitz 1	128	5.8	4.2	1.4
	256	60.2	31.6	1.9
Toeplitz 2	128	16.3	6.2	2.6
	256	190.4	49.25	3.8
Easy	800	4.5	244.5	0.018
Shifted Wilk.	100	508.0	5.3	95.8
$(300x-1)^{100}$	100	260.0	.3	866.6

Figure 4: Comparison of `eigensolve` and `mpsolve` (version 2.1, January 2000). Times are seconds on a 250MHz SGI R10000; roots reported to 15 decimal digits. Most of the polynomials are from the `mpsolve` web site. The "Shifted Wilkinson" polynomial has integer roots $10^4+1, \ldots, 10^4+100$.

[8] G.H. Golub, C.F. Van Loan, *Matrix Computations*, 2nd edition, The Johns Hopkins University Press, 1990.

[9] X. Gourdon, *Combinatoire, Algorithmique et Géométrie des Polynômes*, Ph.D. thesis, L'École Polytechnique, 1996, available at `pauillac.inria.fr/algo/gourdon/`.

[10] T. Granlund, GNU MP: the GNU multiple precision arithmetic library, Edition 2.0, 1996. Code available for download from `www.swox.com/gmp/`.

[11] B.T. Smith, Error bounds for the zeroes of a polynomial based upon Gershgorin's theorem, *JACM*, **17**:661-674, 1970.

[12] B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V.C. Klema, C.B. Moler, Matrix Eigensystem Routines - EISPACK Guide, volume 6 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1976. Source code is available from netlib at `www.netlib.org`.

[13] P. Henrici, *Applied and computational complex analysis: Vol. 1*, John Wiley and sons, 1974.

[14] F. Malek, R. Vaillancourt, A composite polynomial zerofinding matrix algorithm, *Computers Math. Applic.* **30**(2):37–47, 1995.

[15] J. McNamee, A bibliography on roots of polynomials, *J. Comput. Appl. Math* **47** pp. 391–394, 1993. An updated bibliography is available at `www.elsevier.nl/homepage/sac/cam/mcnamee/index.html`.

[16] V. Pan, Solving a polynomial equation: some history and recent progress, *SIAM Review* **39**:2, pp. 187–220, 1997.

[17] F. Tisseur, *Backward stability of the QR algorithm*, Technical Report No 239, UMR 5585, Lyon Saint-Etienne, October 1996.

[18] K.C. Toh, L. N. Trefethen, Pseudozeroes of polynomials and pseudospectra of companion matrices, *Numerische Mathematik*, **68**:403–425, 1994.

[19] P. Van Dooren, P. Dewilde, The eigenstructure of an arbitrary polynomial matrix: computational aspects, *Lin. Alg. Appl*, **50**:545–579, 1983.

[20] C.K. Yap, *Fundamental problems of algorithmic algebra*, Oxford University Press, Oxford, 2000.