Accelerated Solution of the Tridiagonal Symmetric Eigenvalue Problem

Victor Pan

TR-93-016

March 1993

Accelerated solution of the tridiagonal symmetric eigenvalue problem

Victor Pan

Mathematics and Computer Science Department

Lehman College, City University of New York

Bronx, NY 10468

and International Computer Science Institute

Suite 600, 1947 Street

Berkeley, CA 94704

(Supported by NSF Grant CCR 9020690 and

by CUNY University Award #662478)

Summary. We present new algorithms that accelerate the bisection method for the symmetric eigenvalue problem. The algorithms rely on some new techniques, which include acceleration of Newton's iteration and can also be further applied to acceleration of some other iterative processes, in particular, of iterative algorithms for approximating polynomial zeros.

Key words: symmetric eigenvalue problem, bisection algorithm, Newton's iteration, convergence acceleration, polynomial zeros.

1991 Mathematics subject classification: 65F15, 65Y20, 65B99.

1. Introduction.

In this paper, we propose some new techniques in order to accelerate the convergence of the bisection method for the eigenvalues of a real symmetric tridiagonal (hereafter rst) matrix A (compare [Par], [GL], [B], [LPS], [PR]).

We recall (see our section 3) that the bisection algorithm approximates all the eigenvalues $\lambda_1, \ldots, \lambda_n$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, of an $n \times n$ rst matrix A within an error bound t, $0 \leq t \leq \lambda_1 - \lambda_n$, by using

$$4n^2 \lceil H(t) \rceil + O(n) \tag{1.1}$$

arithmetic operations where

$$H(t) = \log_2((\lambda_1 - \lambda_n)/t) \; . \tag{1.2}$$

Our new algorithms involve

$$(4n^2 + O(n))\left[(\log_2 H)^2 + \log_2 n + \nu \log_2 H\right] \tag{1.3}$$

arithmetic operations where ν varies from 4 to $5.5/\log_2 3 \approx 3.47$. (Note the decrease of the complexity bound from $O(H(t))$ in (1.1) to $O(\log H(t))$ in (1.3); see the derivation of the estimate (1.3) and some further comments in section 10 and, for a further improvement, see section 5 and table 10.6.) Furthermore, some features of our algorithms suggest that (unlike the bisection method) they tend to perform substantially better on the average input than the estimate (1.3) indicates.

The techniques used may be of some independent interest. In particular, by employing Newton's iteration for a k-fold zero of a polynomial, our algorithm 6.1 ensures a nearly quadratic convergence, *right from the start*, to single and multiple eigenvalues of A or to their clusters, as soon as each of these eigenvalues or, respectively, of their clusters is sufficiently well isolated from the other eigenvalues of A. The desired isolation is ensured by using the bisection procedure and the double exponential sieve algorithm of [BOT] with its new improvement (see some alternative techniques in [P87], [P89a], [PD]). And we show in section 9 a simple but novel extension of Newton's iteration techniques, leading to a nearly cubic convergence rate, right from the start. This makes our techniques potentially useful for the study and design of other iterative algorithms. Actually, in this paper, we develop the earlier approach of [P87], originally applied to approximating complex polynomial zeros and based on Weyl's construction; our present techniques can in turn be easily extended in order to improve this approach of [P87] to the latter problem. This direction seems to be even more promising from the application point of view, because the polynomial zeros are usually sought with a much higher precision than the eigenvalues of a symmetric matrix. Another promising extension of this work is apparently to improving the known divide-and-conquer algorithms for the symmetric tridiagonal eigenvalue problem (see remark 4.1 in section 4).

Our paper is organized as follows: We recall fast methods for the evaluation of the characteristic polynomial of A and of its derivatives, in section 2, and the bisection algorithm, in section 3. In section 4, we present the structure of our main algorithm, give its informal description and also define the two basic concepts, of separation and isolation. In section 5, we recall the double exponential sieve process of [BOT] for the eigenvalue isolation. In section 6 we describe our algorithm 6.1 for approximating the well-isolated (clusters of) eigenvalues of A. Rapid convergence of this algorithm is proved in section 7, and the algorithm is further accelerated in section 9. In sections 8–10, we summarize our study by presenting (in sections 8 and 9) our two algorithms for approximating the eigenvalues of an rst matrix and (in section 10) their computational complexity estimates, shown both in formal expressions and in tables (for two samples of specific problem sizes).

2. Definitions and auxiliary results.

Hereafter, A denotes the $n \times n$ rst matrix, α_i denotes its entry (i,i), β_j denotes its entries $(j-1,j)$ and $(j,j-1)$; $p_i(x) = \det(xI_i - A_i)$ denotes the characteristic polynomial of the $i \times i$ leading principal submatrix A_i of A,

$$A_n = A \ , \quad p_n(x) = p(x) = \det(xI_n - A) \ ,$$

$$p_0(x) = 1 \ , \qquad p_1(x) = x - \alpha_1 \ ,$$

$$p_j(x) = (x - \alpha_j)\, p_{j-1}(x) - \beta_{j-1}^2 p_{j-2}(x) \ , \tag{2.1}$$

where $i = 1, \ldots, n$; $j = 2, \ldots, n$. Due to (2.1), it suffices to use $2n - 3$ multiplications and $2n - 1$ additions to evaluate the sequence $p_1(x), \ldots, p_n(x)$ for any fixed x (assuming that the values β_h^2 have been precomputed for all h, in $n - 1$ multiplications). The number of sign agreements in this sequence equals the number $n_-(x)$ of the eigenvalues of A that are less than x ([GL, p. 438], [Par, p. 131]).

Furthermore, $4n - 8$ multiplications and $4n - 5$ additions suffice to evaluate both $p(x)$ and its derivative $p'(x)$ ([BP]). Indeed, introduce an auxiliary variable z, replace $p_h(x)$, for $h = j, j-1, j-2$, by $p_h(x+z) \bmod z^2 = p_h(x) + z\, p_h'(x)$ in (2.1), and recursively compute $p_h(x)$ and $p_h'(x)$ from the resulting expressions for $p_h(x+z) \bmod z^2$. Likewise, we may replace $p_h(x)$ for $h = j, j-1, j-2$ in (2.1) by $p_h(x+z) \bmod z^3$ and then recover $p(x)$, $p'(x)$ and $p''(x)$ from $p_h(x+z) \bmod z^3 = p_h(x) + zp_h'(x) + z^2 p_h''(x)/2$.

Remark 2.1. The above approach can be extended to the evaluation of higher order derivatives of the polynomial $p(x)$ and to computing its coefficients.

We will measure the computational complexity by the number of the evaluations of $n_-(x)$, $p(x)$ and $p'(x)$ and will deal with real semi-open intervals of the form

$$[a, b) = \{x, \, a \leq x < b\} \ .$$

Hereafter, all logarithms are to the base 2, and $t > 0$ denotes the tolerance to the absolute errors of the output approximations to $\lambda_1, \ldots, \lambda_n$, the eigenvalues of A, where

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \ .$$

3. Bisection algorithm.

Next, we will recall the bisection algorithm for approximating the eigenvalues of A (compare [B], [PR], [LPS]).

Algorithm 3.1.

Input: an $n \times n$ rst matrix A, two real numbers λ_{n+1} and λ_0, such that $\lambda_{n+1} < \lambda_n \leq \lambda_1 < \lambda_0$, and an error tolerance $t > 0$.

Output: approximations (within the absolute error bound t) to the n eigenvalues of A in the interval $[\lambda_{n+1}, \lambda_0)$.

Initialize: call the interval $[\lambda_{n+1}, \lambda_0)$ *suspect.*

Recursive step: for every suspect interval $[q, r)$ such that $r - q > 2t$, compute $n_-(\frac{q+r}{2})$; call the subinterval $[\frac{q+r}{2}, r)$ *suspect* if $n_-(r) > n_-((q+r)/2)$; call the subinterval $[q, \frac{q+r}{2})$ *suspect* if $n_-(\frac{q+r}{2}) > n_-(q)$; remove the label "suspect" for the interval $[q, r)$.

Stopping criterion: end the computation when all the suspect intervals have length at most $2t$; for each of these intervals, output its midpoint and the number of the eigenvalues of A lying in it.

To estimate the computational cost of the bisection algorithm, note that in each its recursive step, there is at least one eigenvalue in each suspect interval, and thus, there are at most n suspect intervals at each recursive step.

On the other hand, in step s, each suspect interval has length $(\lambda_0 - \lambda_{n+1})/2^s$, so that $S = \lceil \log(\frac{\lambda_0 - \lambda_{n+1})}{t}) \rceil - 1$ steps suffice to arrive at the intervals of length at most $2t$, at which point we output the soluiton. Each s-th of these S steps requires $k(s) \leq n$ evaluations of $n_-(x)$, at the midpoints of each of the $k(s) \leq n$ suspect intervals. This leads to (1.1) since the extremal eigenvalues λ_1 and λ_n of an rst matrix can be easily approximated ([GL], [P90]), so that we may assume, say, that $\lambda_{n+1} - \lambda_0 \leq 2(\lambda_1 - \lambda_n)$.

4. Separation and isolation.

Our objective is to modify the bisection algorithm to accelerate its convergence. We will still rely on the *recursive partitioning* of the input interval $[\lambda_{n+1}, \lambda_0)$ by its interior points, but not necessarily by the midpoints of suspect intervals. One of our two major tools will be Newton's iteration algorithm, which is well known to have local quadratic convergence. In fact, we will show (in sections 7 and 9) its superlinear (nearly quadratic or nearly cubic) convergence right from the start, provided that we start with an approximation to a single eigenvalue (or to a cluster of eigenvalues) sufficiently well isolated from all the other eigenvalues of A. We will quantitatively specify the term "sufficiently well" by using the concept of an *isolation ratio* ([P87], [P89]), abbreviated as ir. For an interval $J = [m - \ell, m + \ell)$, its isolation ratio, $\text{ir}(J)$, equals H/ℓ, H being the distance from m to the nearest eigenvalue of A not lying in J itself, so that $H \geq \ell$, $\text{ir}(J) \geq 1$.

Now, if we have a cluster of the eigenvalues of A in a fixed real interval J, containing no other eigenvalues of A and having its isolation ratio at least $8n$, then we will prove (near) quadratic convergence of Newton's iteration algorithm 6.1 of section 6 to this cluster, right from the start. Furthermore, our modified algorithm 9.1 of section 9 reaches nearly cubic convergence, right from the start, if $ir(J) \geq 2 + 4\sqrt{n}$. Moreover, the same algorithms remain effective under the above bound on $ir(J)$, no matter whether all the eigenvalues of A in J form a single cluster or not. In the latter case, the algorithms converge (with the same speed) not to a common approximation point for all the eigenvalues of A in J (such a point cannot exist in this case), but to a *splitting point* defined as follows:

A partition of an interval $[a, b)$ by its internal point x is called a *separation step* if

$$n_-(a) < n_-(x) < n_-(b) , \tag{4.1}$$

and then x is called a *splitting point.*

Clearly, there can be at most $n-1$ separation steps of the bisection or of any recursive partition algorithm. Due to this bound $n-1$, we just need to devise a recursive partition algorithm whose every sufficiently long sequence of successive recursive steps, including no separation steps, rapidly converges to the eigenvalues of A. We will indeed show such an algorithm in the next sections (where "sufficiently long" will be interpreted as the order of $\log n$), and in fact, the only remaining problem is to ensure high isolation ratios of the initial approximation intervals. Indeed, if the isolation ratios are high enough, already the Newton iteration algorithms rapidly converge either to an approximation point or to a splitting point. Since there can be at most $n-1$ separation steps, it follows that at most $2n-1$ (in the worst case!) applications of these algorithms will give us the desired approximations to the n eigenvalues of A.

Let us next turn to the problem of isolation. Looking for the increase of the isolation ratio of the original input interval, we will employ, in particular, the following simple result:

Fact 4.1. *Let $ir(J) = 1 + u$ for a suspect interval J (see algorithm 3.1). Then the bisection of J either outputs two suspect intervals of half length (and then the bisection is a separation step) or else defines a single suspect subinterval J^* of J (of half length) such that*

$$\text{ir}(J^*) \geq 1 + 2u \ .$$

If h bisection steps have been successively applied to the interval J of fact 4.1 such that $\text{ir}(J) \geq 1 + u$ and if neither of them turned out to be a separation step, then their output suspect subinterval of J has length $|J|/2^h$ and has an isolation ratio of at least $1 + 2^h u$.

In the next section we will recall an algorithm from [BOT] that, for any interval J of fact 4.1, rapidly computes either a splitting point in J or a subinterval J_0 of J such that all the eigenvalues of A in J lie in J_0 and $\text{ir}(J_0) \geq 3$, in which case $u \geq 2$, $1 + 2^h u \geq 1 + 2^{h+1}$. We will also show a further improvement of this algorithm.

Summarizing, our approach has the following structure (compare section 8).

1) apply improved algorithm of [BOT], to ensure (after sufficiently many separation steps) an isolation ratio of at least $1 + 2v$ for a fixed $v \geq 1$ ($v = 1$ in the version of [BOT]), for every eigenvalue or every cluster of the eigenvalues of A;

2) apply the bisection algorithm to increase the isolation ratios to at least $8n$ or to at

least $2+4\sqrt{n}$;

3) apply Newton's iteration (algorithm 6.1) or its refinement (algorithm 9.1) to obtain approximations to the eigenvalues of A within the fixed tolerance to the errors.

The process may be interrupted (at most $n-1$ times) by the separation steps, and then it is recursively repeated.

Remark 4.1. Steps 1) and 2) can be included in all the known divide-and-conquer (d.-c.) eigenvalue algorithms ([BP], [C], [DS]), to ensure faster convergence of the subsequent iterative processes (which vary for various d.-c. algorithms). Such an inclusion has been elaborated in [BP], but now we may also incorporate new improvements, of step 1 (shown in the next section) and of the subsequent Newton iteration (by using our algorithm of section 9). The power of the application of our approach is accentuated in the case of the d.-c. algorithms because they immediately furnish us with approximate separation of the eigenvalues of A.

5. Double exponential sieve.

For simplicity, we will study the case where a (single or multiple) eigenvalue λ lies in the interval $[0,2)$ and is the only eigenvalue of A lying in this interval, and we will only comment on the extension to the general case at the end of this section.

By applying one bisection step, we will first specify in which interval, $[0,1)$ or $[1,2)$, λ lies. Without loss of generality, let $0 \leq \lambda < 1$. At this point, to see the main difficulty, assume that λ lies very near 0 and that so do some negative eigenvalues of A. (This situation so far has no perfect solution in the known implementations of the divide-and-conquer algorithms, [C], [DS].) In this case, the bisection process yields the isolation too slowly. Indeed, it is much more efficient in this case to compute $n_-(x_i)$ not for $x_i = 2^{-i}$ but for $x_i = 2^{-2^i}$, $i = 0, 1, \ldots$. This is precisely the strategy of our next algorithm, and this strategy (after its refinement) turned out to be the most effective means for rapidly ensuring higher isolation.

Let us next formally describe the algorithm.

For a fixed tolerance t, $0 < t < 1$, we will seek two real values a and b such that

$$0 \leq a \leq \lambda < b \leq 1$$

and either

$$b - a \leq 2t \tag{5.1}$$

(in which case $|\lambda - (a+b)/2| \leq t$, so that the point $(a+b)/2$ approximates λ within the error bound t) or else

$$b \leq 2a \tag{5.2}$$

(in which case the isolation ratio of the interval $[a, b)$ is at least 3).

The *double exponential sieve* algorithm of [BOT] computes the desired values a and b in $\lceil \log\log(1/(2t)) \rceil^2$ evaluations of $n_-(x)$. More precisely, the algorithm of [BOT] has been devised for approximating real polynomial zeros; we will restate it for the symmetric eigenvalue problem and will then improve it a little.

The algorithm first successively evaluates $n_-(x_i)$ for $x_i = 2^{-2^i}$, $i = 1, \ldots, g_1$, where

$$g_1 = \min\{\lceil \log\log(1/(2t)) \rceil \ , \ \min_{i=1,2,\ldots,} \{i,\, n_-(x_i) < n_-(1)\}\} \ .$$

If $g_1 = \lceil \log\log(1/(2t)) \rceil$, then $0 \leq \lambda \leq 2t$, and we satisfy (5.1) by setting $a = 0$, $b = x_{g_1}$. Otherwise, λ lies in the interval $J = \{\lambda,\, a_1 = x_{g_1} \leq \lambda < x_{g_1 - 1} = b_1\}$. If $g_i = 1$, then $\text{ir}(J) \geq 5/3$, and we increase this ratio to at least $11/3$ in at most 2 applications of fact 4.1. Otherwise, we apply the same double exponential sieve procedure to the latter interval. Let g_2 denote the number of the evaluations of $n_-(x)$ in these applications. Then we either satisfy (5.1) by setting $a = a_1$, $b = a_2 = a_1 + (b_1 - a_1)2^{-2^{g_2}}$ or else obtain that

$$a_2 = a_1 + (b_1 - a_1)2^{-2^{g_2}} \leq \lambda \leq a_1 + (b_1 - a_1)2^{-2^{g_2 - 1}} = b_2 \ .$$

Since $b_1 - a_1 < 1$, the length of the latter interval, $b_2 - a_2$, is less than a_1 if $g_2 \geq g_1$, and in this case, we set $a = a_2$, $b = b_2$, and satisfy (5.2). Thus, it remains to consider the case where $g_2 < g_1$. Recursively, we arrive at a decreasing sequence of positive integers $\{g_1, g_2, g_3, \ldots\}$ ending with a term g_u such that setting

$$a = a_u \ , \qquad b = b_u$$

satisfies (5.1) and/or (5.2). Since the sequence $\{g_1, g_2, \ldots\}$ strictly decreases, we have $u \le g_1$, so that the overall number of the evaluations of $n_-(x)$ in the entire process is at most

$$T_0 = \sum_{i=1}^{u} g_i \le 1 + 2 + \cdots + g_1 = (g_1 + 1)g_1/2 \ , \quad g_1 = \lceil \log\log(1/(2t)\,) \rceil \ . \tag{5.3}$$

Extension. If the input interval $[0, 2)$ has been replaced by any interval J of length $2L$, which may contain several eigenvalues of A, then the above process can be immediately extended so that, in at most $(g_1^*+1)g_1^*/2$ (for $g_1^* = \lceil \log\log(L/(2t)\,) \rceil^2$) evaluations of $n_-(x)$, we either compute a splitting point x in J [compare (4.1)], thus defining a separation of the eigenvalues of A from each other, or else output a subinterval $\widetilde{J}$ of J containing the same eigenvalues of A as J and such that $|\widetilde{J}| \le 2t$ (which ensures the desired approximation to all the $2L$ eigenvalues of A in J) and/or $\mathrm{ir}(\widetilde{J}) \ge 3$ (which ensures good initial isolation of these $2L$ eigenvalues from the other eigenvalues of A).

Improvement. The double exponential sieve process can be applied with any base $1 + v > 1$, rather than with the base $1 + v = 2$, as in [BOT], that is, we may initially set $x_i = (1+v)^{-2^i}$ and compute $n_-(x_i)$ as above, for $i = 1, \ldots, g_1(v)$; then we shall recursively apply a similar process to the intervals $\{\lambda, x_{g_k(v)} \le \lambda < x_{g_k(v)-1}\}$, $k = 1, 2, \ldots, u(v)$. In this case, the complexity estimate (5.3) (extended to any interval of length $2L$) changes into the estimate

$$T_0^*(v) = (g_1(v) + 1)\, g_1(v)/2, \; g_1(v) \le \lceil \log\left(\log(L/(2t))/\log(1+v)\right) \rceil \ , \tag{5.4}$$

and the lower bound on the isolation ratio of the output interval of the double exponential sieve process remains equal to 3, except for the case where $g_1 = 1$, and then this ratio changes from 3 to $1+2/v$. In the latter case, h applications of fact 4.1 increase the ratio to at least $1+2^{1+h}/v$, which is at least 3 if $h \geq \log v$. Thus, we shall extend the complexity bound (5.3) (adjusted to the interval L) to the following estimate:

$$T_0(v) = \max\{T_0^*(v),\ \lceil \log v \rceil\}\ , \tag{5.5}$$

which we may optimize by choosing an appropriate value for the parameter v. In particular, consider the two cases:

(a) $\log(L/(2t)) = 69$, $g_1(1) = 7$,

(b) $\log(L/(2t)) = 34$, $g_1(1) = 6$.

Then we may set $v = 512$ in case (a), $v = 64$ in case (b) and obtain from (5.3), (5.5) that

$$T_0 = T_0(1) = 28\ , \qquad T_0(v) = 9\ , \qquad \text{in case (a)},$$

$$T_0 = T_0(1) = 15\ , \qquad T_0(v) = 6\ , \qquad \text{in case (b)},$$

Due to the influence of the choice of v on the subsequent computations, it turns out to be more effective to set

$$v = 19\ , \quad T_0(v) = 10\ , \quad \text{in case a)}, \tag{5.6}$$

and

$$v = 3.4\ , \quad T_0(v) = 6\ , \quad \text{in case b)}. \tag{5.7}$$

Remark 5.1. The equations (5.3) and even (5.5) give us overly pessimistic estimates, because for a random λ, chosen under the uniform probability distribution on the interval from 0 to 1, we have that

$$\text{Probability}\{g_i(v) > i\} = (1+v)^{-2^i}\ ,$$

and similarly, Probability$\{g_k(v) > i\}$ also rapidly decreases to 0 as i grows, for all k. Moreover, the same effect can be achieved by means of the randomization of the choice of v, rather than λ.

6. Accelerated computation in the case of good initial isolation.

In this section we will propose an algorithm that, for a given interval J having a sufficiently high isolation ratio N, either separates the eigenvalues of A in J or approximates all of them (within a fixed tolerance t to the errors) at the cost of $O(\log\log(|J|/t))$ evaluations of $n_-(x)$. The choice of N will be specified in sections 7–10.

We will first formalize our task as the following computational problem:

Problem 6.1. Input: real a, b, N, t and integers j, k, $n_-(a)$, $n_-(b)$ such that

$$t > 0\ , \quad b > a\ , \quad j \geq 0\ , \quad k \geq 1\ , \quad j+k \leq n\ ,$$

$$\lambda_{j+k+1} \leq a - C < a \leq \lambda_{j+k} \leq \cdots \leq \lambda_{j+1} < b < b + C \leq \lambda_j \tag{6.1}$$

where $C = (b-a)(N-1)/2$.

(Due to the above relations, the interval $[a, b)$ contains exactly k eigenvalues of A, that is, $\lambda_{j+1}, \ldots, \lambda_{j+k}$, and has an isolation ratio of at least N.)

Output: a real μ such that

$$\lambda_{j+k} - t \leq \mu \leq \lambda_{j+1} + t \tag{6.2}$$

and the integers $n_-(\mu - t)$, $n_-(\mu + t)$.

Once these output values are available, we have that either

$$n_-(\mu + t) = n_-(b)\ , \quad n_-(\mu - t) = n_-(a)$$

(in which case

$$\mu - t \leq \lambda_{j+k} \leq \cdots \leq \lambda_{j+1} < \mu + t\ ,$$

so that μ approximates all the eigenvalues $\lambda_{j+1}, \ldots, \lambda_{j+k}$ within the error bound t) or $n_-(\mu + t) < n_-(b)$ [and then $\mu + t$ is a splitting point,

$$\lambda_{j+k} \leq \mu + t < \lambda_{j+1}\,]$$

or $n_-(\mu - t) > n_-(a)$ [and then $\mu - t$ is a splitting point,

$$\lambda_{j+k} < \mu - t \le \lambda_{j+1}] \, .$$

To solve problem 6.1, we will apply Newton's iteration for computing the k-fold zero of a function. Since we deal with a cluster of the zeros of $p(x)$, well isolated from the other zeros of $p(x)$ (because we assume that N and C are large), this cluster should behave like a k-fold multiple zero of $p(x)$, and Newton's iteration is expected to converge rapidly. Our analysis in the next section confirms this expectation.

Let us next formally describe a recursive algorithm that realizes this idea in order to solve problem 6.1.

Algorithm 6.1. Initialization: set $a_0 = a$, $b_0 = b$, $N_0 = N$, $t_0 = (b_0 - a_0)/2$.

Recursive step i, $i = 0, 1, \ldots$. Choose a nonnegative h_i (according to remark 6.1 below) and compute

$$c_i = b_i + (b_i - a_i)h_i \, , \tag{6.3}$$

$$\mu_i = c_i - k\, p(c_i)/p'(c_i) \, , \tag{6.4}$$

$$t_{i+1} = 2(b_i - a_i)(h_i + 1)^2 M_i \,/\, (k - 2(h_i + 1)\, M_i) \tag{6.5}$$

where

$$M_i = \max \left\{ \frac{n - j - k}{N_i + 1 + 2h_i} \, , \, \frac{j}{N_i - 1 - 2h_i} \right\} . \tag{6.6}$$

[Note that (6.4) represents Newton's iteration for approximating a k-fold zero of $p(x)$.] If $t_{i+1} \le t$, then output $\mu = \mu_i$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$ and end. Otherwise compute $a_{i+1} = \mu_i - t_{i+1}$, $b_{i+1} = \mu_i + t_{i+1}$, $n_-(a_{i+1})$ and $n_-(b_{i+1})$. If

$$n_-(a) < n_-(a_{i+1}) \, ,$$

output $\mu = a_{i+1}$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$ and end. Otherwise, if

$$n_-(b_{i+1}) < n_-(b) \, ,$$

output $\mu = b_{i+1}$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$ and end. Otherwise, set

$$N_{i+1} = \frac{(N_i - 1)(k - 2(h_i + 1)M_i)}{4(h_i + 1)^2 M_i} - 1 , \tag{6.7}$$

where M_i is defined by (6.6), and go to step $i + 1$.

Remark 6.1. Estimating the complexity of our computations in sections 7, 8 and 9, we will focus on the cases where $h_i = 0$ and $h_i = 1$, providing also the analysis in the case of any positive $h_i < (N_i - 1)/2$ (in section 7) and $h_i < (N_i - 2)/2$ (in section 9). The two choices of $h_i = 0$ and $h_i = 1$ for all i lead to about the same complexity estimates for our algorithms. Due to the symmetry, this analysis can be extended to the choice of $h_i \leq -1$ [after the respective adjustment of the expressions (6.5) and (6.6) for t_i and M_i]. The choice of $h_i \leq -1$ in the cases where $a_i - \lambda_{j+k+1} > \lambda_j - b_i$ and of $h_i \geq 0$ otherwise should slightly improve the convergence of the algorithm.

7. Analysis of algorithm 6.1.

To analyze algorithm 6.1, we will assume that $2h_i < N_i - 1$ and will first recall that

$$-p'(x)/p(x) = \text{trace}(\,(A - xI)^{-1}) = \sum_{r=1}^{n} 1/(\lambda_r - x) ,$$

$$a \leq \lambda_r < b , \qquad r = j + 1, \ldots, j + k ,$$

$$\lambda_r \leq a - C , \qquad r = j + k + 1, \ldots, n ,$$

$$\lambda_r \geq b + C , \qquad r = 1, \ldots, j ,$$

where $C = (b - a)(N - 1)/2$ [as in (6.1)].

For large C and N and for smaller positive h_0, the reciprocals of the eigenvalues of $A - I$ from the interval $[a, b)$ dominate in the entire sum $-p'(x)/p(x) = \sum_{r=1}^{n} 1/(\lambda_r - x)$ if $a \leq x < b$. Therefore, $-p'(x)/(kp(x))$ well approximates the average value S_0/k of these k largest reciprocals. On the other hand, k/S_0 must lie between λ_{j+1} and λ_{j+k}, and consequently, $-kp(x)/p'(x)$ must lie in or near the interval $[\lambda_{j+1}, \lambda_{j+k})$. When we compute $-kp(x)/p'(x)$, we may include the latter unknown interval in a small subinterval of $[a, b)$ lying about μ_0 of (6.4) and having a higher isolation ratio than $[a, b)$. We will next formalize this argument and will apply it recursively, to prove both rapid growth of the isolation ratio and rapid decrease of the interval length in the recursive process.

Examine step i, for $i = 0$; denote that $h = h_0$, $M_0 = M$ (to simplify the notation). Since $c_0 = b + (b-a)h$ [see (6.3)], the latter inequalities imply that

$$(a-b)(h+1) \le \lambda_r - c_0 < (a-b)h \ , \qquad r = j+1, \ldots, j+k \ ,$$

$$\lambda_r - c_0 \le (N+1+2h)(a-b)/2 \ , \qquad r = j+k+1, \ldots, n \ ,$$

$$\lambda_r - c_0 \ge (N-1-2h)(b-a)/2 \ , \qquad r = 1, \ldots, j \ .$$

Therefore,

$$S_0 = \sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - c_0} \le \frac{k}{(a-b)(h+1)} \ , \tag{7.1}$$

$$|S_0| \ge \frac{k}{(b-a)(h+1)} \ ; \tag{7.1a}$$

$$\frac{2(n-j-k)}{(N+1+2h)(a-b)} \le S_1 = \sum_{r=1+j+k}^{n} \frac{1}{\lambda_r - c_0} < 0 \ ,$$

$$0 < S_2 = \sum_{r=1}^{j} \frac{1}{\lambda_r - c_0} \le \frac{2j}{(N-1-2h)(b-a)} \ ,$$

$$|S_1 + S_2| \le 2M/(b-a) \ ,$$

$$M = \max\left\{\frac{n-j-k}{N+1+2h} \ , \ \frac{j}{N-1-2h}\right\} \le \frac{n-1}{N-1-2h}$$

[compare (6.6)]. It follows that

$$-\frac{p'(c_0)}{k\,p(c_0)} = \frac{1}{k}\sum_{r=1}^{n} \frac{1}{\lambda_r - c_0} = \frac{S_0}{k}\left(1 + \frac{S_1 + S_2}{S_0}\right) = \frac{S_0}{k}(1+\rho) \ ,$$

$$|\rho| = |S_1 + S_2|/|S_0| \le 2(h+1)\,M/k < \frac{2(h+1)(n-1)}{k(N-1-2h)} \ . \tag{7.2}$$

Consequently,

$$-\frac{k\,p(c_0)}{p'(c_0)} = \frac{k}{(1+\rho)S_0}\,. \tag{7.3}$$

We now deduce from (6.1) and (7.1) that

$$\frac{1}{\lambda_{j+1}-c_0} \le \frac{S_0}{k} = \frac{1}{k}\sum_{r=j+1}^{j+k} \frac{1}{\lambda_j - c_0} \le \frac{1}{\lambda_{j+k}-c_0}\,,$$

and therefore,

$$(a-b)(h+1) \le \lambda_{j+k} - c_0 \le k/S_0 \le \lambda_{j+1} - c_0 < (a-b)h\,, \tag{7.4}$$

$$\lambda_{j+k} \le (k/S_0) + c_0 \le \lambda_{j+1}\,. \tag{7.5}$$

On the other hand, recall from (6.4) and (6.5), for $i = 0$, that

$$\mu_0 = c_0 - k\,p(c_0)/p'(c_0)\,, \tag{7.6}$$

$$t_1 = 2(b-a)(h+1)^2\,M/\big(k - 2(h+1)\,M\big)\,. \tag{7.7}$$

Our choice of h and N will guarantee that $|\rho| < 1$, and we deduce from (7.2) that

$$|\rho/(1+\rho)| \le \frac{2(h+1)M}{k(1-|\rho|)} \le \frac{2(h+1)M}{k(1-2(h+1)M/k)} = \frac{2(h+1)M}{k-2(h+1)M}\,.$$

Combining the latter inequality with the bound $k/|S_0| \le (b-a)(h+1)$, of (7.4), and with the equations

$$\frac{k|\rho|}{|(1+\rho)S_0|} = k\Big|\frac{p(c_0)}{p'(c_0)} + \frac{1}{S_0}\Big|$$

[implied by (7.3)] and (7.7), we obtain that

$$t_1 \ge \frac{k|\rho|}{|(1+\rho)S_0|} = k\Big|\frac{p(c_0)}{p'(c_0)} + \frac{1}{S_0}\Big|\,. \tag{7.8}$$

From (7.5) and (7.8), we now deduce that

$$\lambda_{j+1} + t_1 \ge (k/S_0) + c_0 + k\Big|\frac{p(c_0)}{p'(c_0)} + \frac{1}{S_0}\Big| \ge c_0 - kp(c_0)/p'(c_0)\,,$$

and due to (7.6), we conclude that

$$\lambda_{j+1} + t_1 \geq \mu_0 \ .$$

Similarly, we deduce from (7.5), (7.6) and (7.8) that

$$\lambda_{j+k} - t_1 \leq \mu_0 \ .$$

If $t_1 \leq t$, we satisfy (6.2) by setting $\mu = \mu_0$. Otherwise, we have 3 cases:

Case a). $n_-(\mu_0 - t_1) > n_-(a)$. Then

$$\lambda_{j+k} < \mu_0 - t_1 \leq \lambda_{j+1} \ ,$$

and $\mu = \mu_0 - t_1$ satisfies (6.2).

Case b). $n_-(\mu_0 - t_1) = n_-(a)$, $n_-(\mu_0 + t_1) < n_-(b)$. Then

$$\lambda_{j+k} \leq \mu_0 + t_1 < \lambda_{j+1} \ ,$$

and $\mu = \mu_0 + t_1$ satisfies (6.2).

Case c). $n_-(\mu_0 - t_1) = n_-(a)$, $n_-(\mu_0 + t_1) = n_-(b)$. Then

$$\mu_0 - t_1 \leq \lambda_{j+k} \leq \lambda_{j+1} < \mu_0 + t_1 \ ,$$

and by setting

$$a_1 = \mu_0 - t_1 \ , \qquad b_1 = \mu_0 + t_1 \ , \tag{7.9}$$

we bracket the eigenvalues $\lambda_{j+k}, \ldots, \lambda_{j+1}$ in the interval $[a_1, b_1)$ of length $2t_1$. The distance from μ_0 to the nearest eigenvalue of A lying outside the interval $[a_1, b_1)$ is at least $C - t_1$ [for C of (6.1)]. Therefore, the isolation ratio of this interval is at least

$$N_1 = (C/t_1) - 1 = \frac{b-a}{2t_1}(N-1) - 1 = \frac{(N-1)(k - 2(h+1)M)}{4(h+1)^2 M} - 1 \ , \tag{7.10}$$

which is (6.7) for $i = 0$, and we will choose $N = \gamma n$ so as to ensure that $N_1 + 1 > N - 1$ and, therefore, $2t_1 < b - a$.

Thus, the first step of algorithm 6.1 either solves problem 6.1 or reduces its input interval $[a, b)$ to a shorter interval $[a_1, b_1)$, containing exactly the same eigenvalues of A. In the latter case, we again arrive at problem 6.1, with a, b, N replaced by a_1, b_1, N_1, respectively [according to (7.9), (7.10)], and recursively repeat step i of algorithm 6.1, for $i = 1, 2, \ldots$.

From (6.6), we obtain that

$$M_i \leq \frac{n-1}{N_i - 1 - 2h_i} \, .$$

Substitute this bound into (6.7) and (6.5) and deduce that

$$N_{i+1} \geq (N_i - 1)(N_i - 1 - 2h_i)\frac{k - 2(h_i+1)M_i}{4(h_i+1)^2(n-1)} - 1 \, ,$$

$$b_{i+1} - a_{i+1} = 2t_{i+1} \leq 2t_i \frac{N_i - 1}{N_{i+1}} \leq \frac{4(b_i - a_i)(h_i+1)^2(n-1)}{(N_i - 1 - 2h_i)(k - 2(h_i+1)M_i)} \, ,$$

for $i = 0, 1, \ldots$. It follows that

$$\frac{b_{i+1} - a_{i+1}}{b_i - a_i} = O(\frac{n}{kN_i}) \, ,$$

$$N_i^2 / N_{i+1} = O(n/k) \, .$$

For a sufficiently large N, this implies a superlinear (and actually almost quadratic) growth of N_i (with the growth of i) and a respective decrease of t_i.

To give specific estimates, let us next set $h_i = 0$ for all i (compare remark 6.1), apply the above lower bound on N_{i+1}, and obtain that

$$\begin{aligned}(4n-4)(N_{i+1}+1) &\geq (N_i - 1)^2(k - 2M_i) \\ &\geq (N_i - 1)^2(k - (2n-2)/(N_i - 1)) = (N_i - 1)(N_i - 2n + 1) \, .\end{aligned}$$

Assume that $N = \gamma_0 n$, $\gamma_0 > 6$, and recursively deduce that

$$N_{i+1} > \gamma_{i+1} n \, ; \quad \gamma_{i+1} = (\gamma_i - 2)\gamma_i / 4 \, , \quad i = 0, 1, \ldots$$

Set $\gamma_i = 4\gamma_i^* + 2$, $i = 0, 1, \ldots$, and obtain that $\gamma_{j+i}^* > (\gamma_j^*)^{2^i}$ for all integers $i > 0$ and $j \geq 0$. It follows that, for any fixed $j \geq 0$,

$$N_{i+j} > (4(\gamma_j^*)^{2^i} + 2)n \ ,$$

$$t_{i+j+1}/t_{i+j} \leq \frac{4n-4}{N_{i+j} - 2n + 1} < (\gamma_j^*)^{-2^i} \ ,$$

$$t_{i+j+1} < (\gamma_j^*)^{1-2^{i+1}} t_j \ , \qquad i = 0, 1, \ldots \tag{7.11}$$

In particular, for $j = 0$, we obtain that

$$t_{i+1} < (\gamma_0^*)^{1-2^{i+1}} (b-a)/2 \ .$$

Hereafter, we will denote that

$$H = \log((b-a)/(2t))$$

[which corresponds to (1.2) with $b - a = 2(\lambda_1 - \lambda_n)$], and we now obtain that

$$\widehat{T}(\gamma_0^*) = \lceil \log(1 + H/\log \gamma_0^*) \rceil - 1 \tag{7.12}$$

recursive steps of algorithm 6.1 suffice to solve problem 6.1 assuming that $N = \gamma n$, $\gamma > 6$, $\gamma_0^* > 1$, and $h_i = 0$ for all i.

In particlar, we have:

$$\widehat{T}(2) = \lceil \log(H+1) \rceil - 1 \ , \tag{7.12a}$$

$$\widehat{T}(4) = \lceil \log(H+2) \rceil - 2 \ , \tag{7.12b}$$

$$\widehat{T}(8) = \lceil \log(1 + H/3) \rceil - 1 \ , \tag{7.12c}$$

$$\widehat{T}(16) = \lceil \log(H+4) \rceil - 3 \ , \tag{7.12d}$$

and we need to set $N = 10n$, $N = 18n$, $N = 34n$ and $N = 66n$ in order to arrive at (7.12a), (7.12b), (7.12c) and (7.12d), respectively.

Let us now set $h_i = 1$ for all i. Then

$$M_i < (n-1)/(N_i - 3) ,$$

$$16(n-1)(N_{i+1}+1)/(N_i - 1) \geq (N_i - 3)(k - 4M_i) \geq (N_i - 3)(k - 4(n-1)/(N_i - 3))$$
$$= k(N_i - 3) - 4(n-1) \geq N_i - 4n + 1$$

since $k \geq 1$.

It follows that, for $N = \gamma n$ and for a sufficiently large γ, we have:

$$N_{i+1} > \gamma_{i+1} n ,$$

$$\gamma_{i+1} = (\gamma_i - 4)\gamma_i/16 , \qquad i = 0, 1, \ldots$$

Set $\gamma_i = 16\gamma_i^* + 4$ and deduce that

$$\gamma_{i+1}^* > (\gamma_i^*)^2 , \qquad \gamma_{i+j}^* > (\gamma_j^*)^{2^i} ,$$

for all integers $i > 0$ and $j \geq 0$. Therefore, for any fixed integer $j \geq 0$, we have:

$$N_{i+j} > (16(\gamma_j^*)^{2^i} + 4)n ,$$

$$t_{i+j+1}/t_{i+j} \leq \frac{16n - 16}{N_{i+j} - 4n + 1} < (\gamma_j^*)^{-2^i} ,$$

which again gives us (7.11), (7.12), (7.12a), (7.12b) and (7.12c), although this time for a slightly distinct expression of γ_0^* through $\gamma_0 = \gamma$, that is, for $\gamma_0^* = (\gamma_0 - 4)/16$. In particular (assuming $h_i = 1$ for all i), we now need to set $N = 36n$, $N = 68n$, $N = 132n$ and $N = 260n$ in order to arrive at (7.12a), (7.12b), (7.12c) and (7.12d), respectively.

Remark 7.1. The same techniques of the analysis would improve the bound (7.12) for $k > 1$. A small improvement (with a more difficult analysis job) could also be obtained based on choosing positive j in (7.11).

8. The eigenvalue algorithm.

Combining the algorithms of the previous sections, we will now accelerate the bisection algorithm 3.1. We will first restate the eigenvalue problem for any fixed interval $[a, b)$ as follows:

Problem 8.1. Input: an $n \times n$ rst matrix A, real a, b, t and the integers k, $n_-(a)$, $n_-(b)$ such that $t > 0$, $a < b$, $k = n_-(b) - n_-(a) > 0$, $0 \leq n_-(a) < n_-(b) \leq n$.

Output: a splitting point x, such that (4.1) holds, or an approximation, μ, within the tolerance t, to all the eigenvalues of A lying in $[a, b)$.

In the latter case, μ serves as a desired solution to the eigenvalue problem on $[a, b)$. In the former case, we reduce the eigenvalue problem on $[a, b)$ to two smaller eigenvalue problems on two subintervals. Proceeding recursively, we will arrive (in at most $k - 1$ steps) at the former case for problems 8.1 on all the subintervals; the solutions on these subintervals combined define a solution of the original eigenvalue problem on the input interval $[a, b)$.

An algorithm for the solution of problem 8.1 now follows, with the choice of the parameters v, N and h_i according to the recipes of sections 5–7.

Algorithm 8.1.

1°. Fix an appropriate positive v and apply the double exponential sieve process of section 5 to the interval $J = [a, b)$. The process either solves problem 8.1 or outputs a subinterval $\widehat{J}$ of $[a, b)$ containing the same eigenvalues as the input interval $[a, b)$, and in this case, the process also outputs ir_-, a lower bound 3 or $1 + 2/v$ on the isolation ratio of $\widehat{J}$.

2°. In the latter case, fix a sufficiently large N and apply $\lceil \log((N - 1)(ir_- - 1)) \rceil$ bisection steps to the interval $\widehat{J}$. This either solves problem 8.1 or else outputs a subinterval J^* of $\widehat{J}$ containing the same eigenvalues as $\widehat{J}$ and having an isolation ratio of at least N (see fact 4.1).

3°. In the latter case, choose nonnegative h_i, $i = 0, 1, \ldots$, and apply algorithm 6.1 to the interval J^*.

Since there can be at most $k - 1$ separation steps, approximating all the k eigenvalues of A in $[a, b)$ requires at most $2k-1$ calls for algorithm 8.1, that is, in a pessimistic estimate, at most $(2k-1)(T_0(v) + T_1(v, N))$ evaluations of $n_-(x)$, for $k \leq n$, and $(2k-1)T_2(N)$ evaluations of $p(x)/p'(x)$ where

$$T_0(v) = \max\{T_0^*(v), \lceil \log v \rceil\}\ , \quad T_0^*(v) = (g_1(v)+1)g_1(v)/2\ ,\ g_1(v) = \lceil \log \frac{H-1}{\log(1+v)} \rceil\ , \tag{8.1}$$

[compare (5.4) and (5.5)],

$$T_1(v, N) = \lceil \log((N-1)v/2) \rceil\ , \tag{8.2}$$

$$T_2(N) = \lceil \log(cH + d) \rceil\ . \tag{8.3}$$

$H = \log((b-a)/(2t))$; $[a, b)$ is the input interval, $t > 0$ is the tolerance to the output errors, the constants c and d should be defined depending on the choice of the values N and h_i. In particular, by choosing $N = (4\gamma_0^* + 2)n$, $h_i = 0$, for all i, or $N = (16\gamma_0^* + 4)n$, $h_i = 1$, for all i, we arrive at $T_2(N) = \widehat{T}(\gamma_0^*)$, where $\widehat{T}(\gamma_0^*)$ satisfies (7.12).

Remark 8.1. The worst case factor $2k-1$ in the above estimates is overly pessimistic. Indeed, the number of recursive calls for each stage 1°, 2°, 3° of algorithm 8.1 actually varies from k to $2k-1$, and if it reaches the value $2k-1$, then the number of iteration steps in each call to stages 1° and 3° must be substantially less than the estimates (8.1), (8.3) shows (these estimates would have implied the output error of $t/(2\lceil \log k \rceil)$, and thus the algorithm should stop earlier).

9. Acceleration of algorithm 6.1

When we apply algorithm 6.1 at step 3° of algorithm 8.1, we may have available some approximations to all the eigenvalues of A [not only to $\lambda_{j+1}, \ldots, \lambda_{j+k}$, lying in the input interval $[a, b)$]. Next, we will use such approximations in order to accelerate the convergence of algorithm 6.1 and to decrease the value of N used in this agorithm. For this purpose, we will further weaken the already decayed influence of the reciprocals $1/(\lambda_r - c_i)$ of the remote eigenvalues λ_r, for $r \le j$ and $r > j+k$, on the value $p'(c_i)/p(c_i)$. We yield this simply by subtracting (from this value) $1/(\lambda_r^* - c_i)$, the approximation to such reciprocals, readily available since λ_r^* are available. Our further analysis shows that, indeed, this well serves our purpose. To simplify the presentation, we will only show (in some detail) the first recursive step (accelerating step 0 of algorithm 6.1) and will assume that we are given approximations λ_r^* to all the eigenvalues λ_r of A, for $r = 1, \dots, n$, such that

$$|\lambda_r^* - \lambda_r| \le t^* = (b-a)/2 \ . \tag{9.1}$$

In fact, we may ensure this assumption by arranging the steps of algorithm 6.1 so as to always work with the *largest* of the available suspect intervals output at stage 2°.

We now modify algorithm 6.1 by replacing the values μ_i, involved in the expression (6.4) of the recursive step i, by the values μ_i^* defined as follows:

$$\mu_i^* = c_i + k/q_i \ ,$$

$$q_i = -\frac{p'(c_i)}{p(c_i)} - \sum_{r=1}^{j} \frac{1}{\lambda_r^* - c_i} - \sum_{r=j+k+1}^{n} \frac{1}{\lambda_r^* - c_i} \ ,$$

$i = 0, 1, \dots$ [The evaluation of the two latter sums and their subtraction from $-p'(c_i)/p(c_i)$ requires $3(n-k)$ extra arithmetic operations for each i.] Hereafter, we will refer to this modification of algorithm 6.1 as to **algorithm 9.1** and will refer to the respective modification of algorithm 8.1 as to **algorithm 8.1a.**

We need to specify the choice of the parameters h_i and N in this algorithm. We may set $h_i = 0$ for all i, which should lead to faster convergence of the algorithm, but leaves the value $r(c_i) = -p'(c_i)/p(c_i)$ unbounded. To avoid overflow, we should first compute the reciprocal $1/r(c_i)$ and end the computation detecting some eigenvalues of A in the interval $(b_i - t, b_i)$ if this reciprocal value is close to 0.

Another option is to assign small positive values to h_i, which should prevent the computer from overflow since

$$|p'(c_i)/p(c_i)| = \sum_{r=1}^{n} \frac{1}{\lambda_r - c_i} < \frac{k}{h_i} + \frac{n-k}{C - h_i} \ , \quad C = (b-a)(N-1)/2 \ .$$

If V denotes the minimum value that causes overflow, then the bound $|p'(c_i)/p(c_i)| < V$ is guaranteed for any h_i exceeding $k/(V - \frac{n-k}{C-h_i})$, and thus it is sufficient to choose positive h_i of the above order.

To ensure rapid convergence of algorithm 9.1, we need a milder restriction on the parameter N than in the case of algorithm 6.1, namely, we will set

$$N = \Theta n^{1/2} + 2 ,$$

for a constant Θ to be specified later on.

Next, we will analyze algorithm 9.1 extending our analysis from section 7. Setting again $S_0 = \sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - c_0}$, we obtain that

$$q_0 = S_0 + \sum_{r=1}^{j} d_r + \sum_{r=j+k+1}^{n} d_r , \qquad (9.2)$$

$$d_r = \frac{1}{\lambda_r - c_0} - \frac{1}{\lambda_r^* - c_0} = \frac{\lambda_r^* - \lambda_r}{(\lambda_r - c_0)(\lambda_r^* - c_0)} , \quad \text{for all } r . \qquad (9.3)$$

We recall that, unless $j < r \leq j+k$, we have the bound,

$$\frac{1}{|\lambda_r - c_0|} \leq \frac{2}{(N-1-2h)(b-a)} ,$$

which we extend to the bound

$$\frac{1}{|\lambda_r^* - c_0|} \leq \frac{2}{(N-2-2h)(b-a)} ,$$

due to (9.1).

Combine these bounds with (9.1)–(9.3) and obtain that

$$|q_0 - S_0| \leq (n-k)/((N-1-2h)(N-2-2h)t^*) .$$

Denote

$$\rho^* = (q_0 - S_0)/S_0 ,$$

so that $q_0 = (1+\rho^*)S_0$. Deduce from the latter relations and from (7.1a) and (9.1) that

$$k/|S_0| \le (b-a)(h+1) ,$$

$$|\rho^*| \le 2(h+1)(n-k)/((N-1-2h)(N-2-2h)k) . \tag{9.4}$$

Our choice of (small) h and (large) N will guarantee that $|\rho^*| < 1$, and we will obtain the following bound:

$$|k/S_0 - k/q_0| = |\rho^*(k/S_0)/(1+\rho^*)| \le t_1^* ,$$

for

$$t_1^* = 2(h+1)^2(n-k)(b-a)/((N-1-2h)(N-2-2h)k(1+\rho^*)) .$$

We have from (7.5) that

$$\frac{1}{\lambda_{j+1}-c_0} \le \frac{S_0}{k} \le \frac{1}{\lambda_{j+k}-c_0} , \quad \lambda_{j+k} \le \frac{k}{S_0} + c_0 \le \lambda_{j+1} ,$$

and it follows that

$$\lambda_{j+k} - t_1^* \le \mu_0^* = c_0 + k/q_0 \le \lambda_{j+1} + t_1^* .$$

Thus, step 0 of our modification of algorithm 6.1 [based on the replacement μ_0 of (6.4) by μ_0^*] either solves problem 8.1 or else brackets the eigenvalues $\lambda_{j+k}, \dots, \lambda_{j+1}$ in the interval (a_1^*, b_1^*) of length $2t_1^*$ where

$$a_1^* = \mu_0^* - t_1^* , \qquad b_1^* = \mu_0^* + t_1^* .$$

The isolation ratio of this interval is at least

$$\begin{aligned} N_1^* &= (b-a)(N-1)/(2t_1^*) - 1 \\ &= |1+\rho^*|(N-1)(N-1-2h)(N-2-2h)k/(4(h+1)^2(n-k)) - 1 , \end{aligned} \tag{9.5}$$

which is substantially larger than N_{i+1} of (7.11) for $i=0$ and for large $N = N_0$.

Replacing a, b, N by a_1^*, b_1^*, N_1^*, we may recursively repeat the computations. The above estimates for the growth of N and for the respective decrease of the interval length

$b - a$ can be recursively extended if we follow the (already pointed out) policy of always applying our algorithm to the currently largest approximation interval, thus improving the currently worst approximations to the eigenvalues of A. For $N = \Theta\, n^{0.5+v}$, $\Theta \geq 1$, we have that

$$\Theta\, n^{0.5}/N_1^* = O(1) \ ,$$

which shows a *nearly cubic growth* of the isolation ratio in the transition from $[a, b)$ to $[a_1^*, b_1^*)$. $b - a$, the length of the interval $[a, b)$, decreases at a similar rate in the transition to $b_1 - a_1^*$, the length of $[a_1^*, b_1^*)$.

Let us now set $h = 0$, denote $N = N_0^*$, and obtain that

$$(4n-4)(N_1^*+1) \geq (N_0^*-1)[(N_0^*-1)(N_0^*-2)-2(N_0^*-1)] = (N_0^*-1)((N_0^*)^2-3N_0^*-2n+4] \ .$$

Similarly, we may bound the isolation ratios N_{i+1}^* in terms of N_i^* at the next recursive steps $i = 1, 2, \ldots$ provided that $h_i = 0$ for all i.

Setting $N = N_0^* = \Theta_0 n^{1/2} + 2$, for $\Theta_0 > 4$, we may recursively deduce from these bounds that

$$N_{i+1}^* > \Theta_{i+1} n^{1/2} + 2 \ , \quad \Theta_{i+1} = (\Theta_i^2 - 2)\Theta_i/4 \ , \qquad i = 0, 1, \ldots \ .$$

Denote $\Theta_i^* = (\Theta_i/2) - 1$, and obtain that

$$\Theta_i = 2\Theta_i^* + 2 \ ,$$

$$N_{i+1} > (2\Theta_{i+1}^* + 2)n^{1/2} + 2 \ ,$$

$$\Theta_{i+1}^* > (\Theta_i^*)^3 \ , \qquad i = 0, 1, \ldots \ .$$

Therefore, for all integers $i > 0$, $j \geq 0$, we have that

$$\Theta_{i+j}^* > (\Theta_j^*)^{3^i} \ ,$$

$$N_{i+j} > (2(\Theta_j^*)^{3^i} + 2)n^{1/2} + 2 \ .$$

For any fixed $j \geq 0$, we have:

$$t_{i+j+1}/t_{i+j} = (N^*_{i+j} - 1)/(N^*_{i+j+1} + 1) \leq \frac{4n-4}{(N^*_{i+j})^2 - 3N^*_{i+j} - 2n + 4} < (\Theta^*_j)^{-3^i} ,$$

$i = 0, 1, \ldots$, and consequently,

$$t_{i+j+1} < (\Theta^*_j)^{(1-3^{i+1})/2} t_j , \quad i = 0, 1, \ldots \tag{9.6}$$

In particular, for $j = 0$, we obtain that

$$t_{i+1} < (\Theta^*_0)^{(1-3^{i+1})/2} (b-a)/2 ,$$

and therefore, for H denoting $\log((b-a)/(2t))$, we have that

$$\widetilde{T}(\Theta^*_0) = \lceil (\log(1 + 2H/\log\Theta^*_0))/\log 3 \rceil - 1 \tag{9.7}$$

recursive steps of algorithm 9.1 suffice to solve problem 6.1 provided that $h_i = 0$ for all i and $N = (2\Theta^*_0 + 2)n^{1/2} + 2$, $\Theta^*_0 > 1$, $\Theta_0 > 4$.

Remark 9.1. Setting $j > 1$, we may extend (9.7) to similar bounds for any $\Theta^*_0 > \sqrt{3/2} - 1$.

In particular, we obtain that

$$\widetilde{T}(2) = \lceil (\log(1 + 2H))/\log 3 \rceil - 1 , \tag{9.7a}$$

$$\widetilde{T}(4) = \lceil (\log(1 + H))/\log 3 \rceil - 1 , \tag{9.7b}$$

$$\widetilde{T}(8) = \lceil (\log(3 + 2H))/\log 3 \rceil - 2 , \tag{9.7c}$$

$$\widetilde{T}(16) = \lceil (\log(2 + H))/\log 3 \rceil - 1 , \tag{9.7d}$$

and we need to set $N = \Theta n^{1/2} + 2$, with Θ taking the values 4, 10, 18 and 34 in order to arrive at (9.7a), (9.7b), (9.7c) and (9.7d), respectively.

Let us also supply the estimates in the case where $h_i = 1$ for all i. In this case, we deduce from (9.4) and (9.5) that

$$16(n-1)N^*_1 + 1 \geq (N^*_0 - 1)[(N^*_0 - 3)(N^*_0 - 4) - 4n + 4] = (N^*_0 - 1)[(N^*_0)^2 - 7N^*_0 - 4n + 16]$$

and similarly bound the isolation ratios N_i^* of the intervals $[a_i, b_i)$ computed at the next recursive steps i of algorithm 9.1, for $i = 2, 3, \ldots$.

Setting $N_0 = \Theta_0 n^{1/2} + 2$, for $\Theta_0 > \sqrt{20}$, we deduce that

$$N_i > \Theta_i n^{1/2} + 2 \ , \quad \Theta_{i+1} = (\Theta_i^2 - 4)\Theta_i/16 \ , \quad i = 1, 2, \ldots$$

Now, we denote that

$$\Theta_i^* = (\Theta_i/4) - 2 \ , \quad \Theta_i = 4\Theta_i^* + 2$$

and deduce that

$$\Theta_{i+1}^* > (\Theta_i^*)^3 \ , \qquad i = 0, 1, \ldots \ .$$

Then, using the equations

$$\frac{t_{i+1}}{t_i} = \frac{N_i^* - 1}{N_{i+1}^* + 1} = \frac{16n - 16}{(N_i^*)^2 - 7N_i^* - 4n + 16} \ ,$$

we again deduce the bound (9.6), (9.7), (9.7a)–(9.7d), although this time we assume a distinct expression for Θ_0 through Θ_0^*, that is, $\Theta_0 = 4\Theta_0^* + 2$. In particular, we need to set that $N = N_0^* = \Theta_0 n^{1/2} + 2$ (for Θ_0 taking the values 10, 18, 34 and 66) in order to arrive at (9.7a), (9.7b), (9.7c) and (9.7d), respectively.

Remark 9.1. Seeking convergence acceleration at the expense of performing a little more work per iteration, we may generalize algorithms 6.1 and 9.1 by replacing (6.4) by more general expressions, such as

$$\tilde{\mu}_i = c_i - (k/\tilde{q}_i)^{1/d} \ ,$$

$$\tilde{q}_i = \sum_{r=j+1}^{n} \frac{1}{(\lambda_r - c_i)^d} - \sum_{r=1}^{j} \frac{1}{(\lambda_r^* - c_i)^d} - \sum_{r=j+k+1}^{n} \frac{1}{(\lambda_r^* - c_i)^d} \ ,$$

for some fixed natural $d > 1$, say, for $d = 3$. Note that the value

$$\sum_{r=1}^{n} \frac{1}{(\lambda_r - c_i)^d} = \operatorname{trace}\left((A_i - c_i I)^{-d}\right)$$

can be computed by extending the techniques of section 2. [In the extension of the same approach to approximating polynomial zeros, pointed out in the introduction, the latter value can be easily obtained from the d leading coefficients of the input polynomial $p(x)$, by using Newton's identities.]

10. Summary of the complexity estimates.

We are now ready to summarize our previous analysis into the estimates for the arithmetic complexity of the solution of problem 8.1, assuming that $k = n$, $a \le \lambda_n \le \lambda_1 \le b$, $b - a \le 2(\lambda_1 - \lambda_n)$. We recall (8.1)–(8.3), recall the need for $3n$ extra arithmetic operations in every iteration of algorithm 9.1 (versus algorithm 6.1), apply the operation count for the evaluation of $p(x)$ and $p'(x)$ from section 2, and obtain that

$$(4n^2 + O(n))(2T_0(v) + 2T_1(v, N) + \nu T_2(N)) \tag{10.1}$$

arithmetic operations suffice for the solution (for any choice of $v > 0$ and N, according to sections 5, 7 and 9) provided that either $\nu T_2(N) = 4\widehat{T}(\gamma_0^*)$ [compare (7.12)] or $\nu T_2(N) = 5.5\widetilde{T}(\Theta_0^*)$ [compare (9.7)], depending on which of the algorithms 8.1 or 8.1a we apply. The estimate (10.1) implies the estimate (1.3) of the introduction.

Next, we will calculate the value

$$T(N, 1) = 2T_0(1) + 2T_1(1, N) + \nu T_2(N) \tag{10.2}$$

for both our policies of choosing h_i in algorithms 6.1 and 9.1 (that is, for setting $h_i = 0$ or $h_i = 1$ for all i). We will consider the two cases:

a) $n = 1000$, $H = \log((b - a)/(2t)) = 70$,

b) $n = 500$, $H = \log((b - a)/(2t)) = 35$.

We first obtain from (8.1) that

$$g_1(1) = \lceil \log 69 \rceil = 7\ , \quad T_0(1) = 28\ , \text{ in the case a)}, \tag{10.3a}$$

$$g_1(1) = \lceil \log 34 \rceil = 6\ , \quad T_0(1) = 15\ , \text{ in the case b)}, \tag{10.3b}$$

The values of $T_1(1, N) = \lceil \log(N-1) \rceil - 1$ (as the functions in γ and Θ) and the values of $\nu T_2(N)$ (as the functions in γ_0^* and Θ_0^*), associated with the values of N of our interest, are displayed in Tables 10.1, 10.2 and 10.3. Tables 10.4 and 10.5 relate γ to γ_0^* and Θ to Θ_0^*. Table 10.6 collects the values $2T_0(1)$, $2T_1(1, N)$ and $\nu T_2(N)$ for all choices of γ_0^*, Θ_0^*, including also subdivision of the values $2T_1(1, N)$ into the two cases, where $h_i = 0$ and $h_i = 1$, respectively. In the two bottom lines of Table 10.6, we display the values $T(1, N)$, defined by (10.2), for each of the two cases (where $h_i = 0$ and $h_i = 1$).

The values shown as the numerators of fractions correspond to the case a), and ones shown as the denominators correspond to the case b).

The data for $T(1, N)$ can be compared to the values 128 (in case a)) and 64 (in case b)), which represent the complexity of the bisection algorithm. Table 10.6 shows that with a successful choice of the values γ_0^* and Θ_0^*, we may obtain superior worst case estimates. In parentheses in the third and in the two last lines of Table 10.6, we show these estimates decreased, due to optimizing the value v [compare (5.6) and (5.7)].

Furthermore, unlike the case of the bisection algorithm (whose worst and average case complexity is about the same, algorithms 8.1 and 8.1a, for a large class of input matrices, may actually perform substantially faster than the worst case bounds suggest (compare remarks 5.1, 7.1 and 8.1).

Table 10.1. $T_1(1,N)$ and $T_1(v,N)$ for $N=\gamma n$ and $N=\Theta n^{1/2}+2$, $v=19/3.4$.

γ	10	18	34	66	36	68	132	260					
Θ									4	10	18	34	66
$T_1(1,N)$ $(T_1(v,N))$	$\frac{13}{12}$ $(\frac{17}{15})$	$\frac{14}{13}$ $(\frac{18}{15})$	$\frac{15}{14}$ $(\frac{19}{16})$	$\frac{16}{15}$ $(\frac{20}{17})$	$\frac{15}{14}$ $(\frac{19}{16})$	$\frac{16}{15}$ $(\frac{20}{17})$	$\frac{17}{16}$ $(\frac{21}{18})$	$\frac{17}{16}$ $(\frac{22}{19})$	$\frac{6}{6}$ $(\frac{11}{8})$	$\frac{8}{7}$ $(\frac{12}{10})$	$\frac{9}{8}$ $(\frac{13}{10})$	$\frac{10}{9}$ $(\frac{14}{11})$	$\frac{11}{10}$ $(\frac{15}{12})$

Table 10.2. $\nu T_2(N)=4\widehat{T}(\gamma_0^*)$.

γ_0^*	2	4	8	16
$4\widehat{T}(\gamma_0^*)$	24/20	20/16	16/12	16/12

Table 10.3. $\nu T_2(N)=5.5\widetilde{T}(\Theta_0^*)$.

Θ_0^*	2	4	8	16
$5.5\widetilde{T}(\Theta_0^*)$	22/16.5	16.5/16.5	16.5/11	16.5/11

Table 10.4.

γ_0^*	2	4	8	16
γ (for $h_i = 0$)	10	18	34	66
γ (for $h_i = 1$)	36	68	132	260

Table 10.5.

Θ_0^*	2	4	8	16
Θ (for $h_i = 0$)	4	10	18	34
Θ (for $h_i = 1$)	10	18	34	66

Table 10.6.

(the case of $v = 19/3.4$ is shown in parentheses)

γ_0^*	2	4	8	16				
Θ_0^*					2	4	8	16
$2T_0(1)(2T_0(v))$ [(10.3a), (10.3b)]	56/30 (20/12)							
$2T_1(1,N)$ $(2T_1(v,N))$ $(h_i = 0)$ (Table 10.1)	26/24 (34/30)	28/26 (36/30)	30/28 (38/32)	32/30 (40/34)	12/12 (22/16)	16/14 (24/20)	18/16 (26/20)	20/18 (28/22)
$2T_1(1,N)$ $(2T_1(v,N))$ $(h_i = 1)$ (Table 10.1)	30/28 (38/32)	32/30 (40/34)	34/32 (42/36)	34/32 (44/38)	16/14 (24/20)	18/16 (26/20)	20/18 (28/22)	22/20 (30/24)
$\nu T_2(N)$ (Tables 10.2, and 10.3)	$\frac{24}{20}$	$\frac{20}{16}$	$\frac{16}{12}$	$\frac{16}{12}$	$\frac{22}{16.5}$	$\frac{16.5}{16.5}$	$\frac{16.5}{11}$	$\frac{16.5}{11}$
$T(1,N)$ $(h_i = 0)$ $(T(v,N))$	$\frac{106}{74}$ $(\frac{78}{62})$	$\frac{104}{72}$ $(\frac{76}{58})$	$\frac{102}{70}$ $(\frac{74}{56})$	$\frac{104}{72}$ $(\frac{76}{58})$	$\frac{90}{58.5}$ $(\frac{64}{44.5})$	$\frac{88.5}{60.5}$ $(\frac{60.5}{48.5})$	$\frac{90.5}{57}$ $(\frac{62.5}{43})$	$\frac{92.5}{59}$ $(\frac{64.5}{45})$
$T(1,N)$ $(h_i = 1)$ $(T(v,N))$	$\frac{110}{78}$ $(\frac{82}{64})$	$\frac{108}{76}$ $(\frac{80}{62})$	$\frac{106}{74}$ $(\frac{78}{60})$	$\frac{106}{74}$ $(\frac{80}{62})$	$\frac{94}{60.5}$ $(\frac{66}{48.5})$	$\frac{90.5}{62.5}$ $(\frac{62.5}{48.5})$	$\frac{92.5}{59}$ $(\frac{64.5}{45})$	$\frac{94.5}{61}$ $(\frac{66.5}{47})$

References

[B] H. J. Bernstein, An Accelerated Bisection Method for the Calculation of Eigenvalues of Symmetric Tridiagonal Matrices, *Numer. Math.*, 43, 153–160, 1984.

[BP] D. Bini, V. Pan, Practical Improvement of the Divide-and-Conquer Eigenvalue Algorithms, *Computing*, 48, 109–123, 1992.

[BOT] M. Ben-Or, P. Tiwari, Simple Algorithm for Approximating All Roots of a Polynomial with Real Roots, *J. of Complexity*, 6, 4, 417–442, 1990.

[C] J. J. M. Cuppen, A Divide and Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem, *Numer. Math.*, 36, 177–195, 1981.

[DS] J. J. Dongarra, D. C. Sorensen, A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem, *SIAM J. Sci. Stat. Computing*, 8, 2, 139–154, 1987.

[GL] G. M. Golub, C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1989.

[H] P. Henrici, *Applied and Computational Complex Analysis,* Wiley, 1974.

[LPS] S.-S. Lo, B. Philippe, A. Sameh, A Multiprocessor Algorithm for the Symmetric Tridiagonal Eigenvalue Problem, *SIAM J. Sci. Stat. Computing,* 8, 155–165, 1987.

[P87] V. Pan, Sequential and Parallel Complexity of Approximate Evaluation of Polynomial Zeros, *Computers and Math. (with Applications),* 14, 8, 591–622, 1987.

[P89] V. Pan, Fast and Efficient Parallel Evaluation of the Zeros of a Polynomial having only Real Zeros, *Computer and Math. (with Applications),* 17, 11, 1475–1480, 1989.

[P89a] V. Pan, A New Algorithm for the Symmetric Eigenvalue Problem, Tech. Report TR 89-3, *Computer Science Dept., SUNYA,* Albany, NY, 1989.

[P90] V. Pan, Estimating Extremal Eigenvalues of a Symmetric Matrix, *Computers and Math. (with Applications),* 20, 2, 17–22, 1990.

[PR] B. N. Parlett, J. K. Reid, Tracking the Process of the Lanczos Algorithmn for Large Symmetric Eigenproblems, *IMA J. Num. Anal.,* 1, 135–155, 1981.

[PD] V. Pan, J. Demmel, A New Algorithm for the Symmetric Eigenvalue Problem, preprint, 1991.

[Par] B. N. Parlett, *The Symmetric Eigenvalue Problem,* Prentice-Hall, 1980.