Bisection Acceleration for the Symmetric Tridiagonal Eigenvalue Problem *

Victor Y. Pan[1] and Elliot Linzer[2]

[1] Department of Mathematics and Computer Science
Lehman College, City University of New York
Bronx, NY 10468, USA
Internet: vpan@lcvax.lehman.cuny.edu

[2] DiviCom
1 Water Street
White Plains, NY 10601
Internet: elliotl@divi.com

February 23, 1999

Abstract

We present new algorithms that accelerate the bisection method for the symmetric tridiagonal eigenvalue problem. The algorithms rely on some new techniques, including a new variant of Newton's iteration that reaches cubic convergence (right from the start) to the well separated eigenvalues and can be further applied to acceleration of

* The results of this paper (apart from numerical experiments, designed and performed by the second author) have been presented (in the preliminary form of [P92]) at the Copper Mountain Conference on Iterative Methods, Copper Mountain, Colorado, 1992.

some other iterative processes, in particular, of the divide-and-conquer methods for the symmetric tridiagonal eigenvalue problem.

Key words: symmetric eigenvalue problem, bisection algorithm, divide-and-conquer eigenvalue algorithms, Newton's iteration, convergence acceleration, approximating polynomial zeros.

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

1 Introduction

The classical bisection method is a reliable approach to the solution of the symmetric eigenvalue problem [Par], pp. 53-54; [GL], pp. 437-443, particularly effective for approximating a few eigenvalues confined to a fixed interval. Successful acceleration of the convergence of this method usually attracts a lot of interest; celebrated examples are given in [B], [LPS], and [PR]. In this paper, we propose new acceleration techniques, extending the ones of [P87], [P89], [P94], originally developed for approximating polynomial zeros (also compare [PD], [P92]). We analyze the resulting algorithms and give formal rigorous proofs of their complexity estimates. The analysis and proofs are quite involved, but the implementation and coding of our algorithms are sufficiently simple. The results of our numerical experiments, presented in section 13, confirm the expected behavior of the algorithms. The algorithms substantially accelerate bisection for higher precision approximation of the eigenvalues.

Our bisection acceleration relies on some general techniques of convergence acceleration, already proved to be effective for approximating polynomial zeros [P87], [P94] and matrix eigenvalues [BP91], [BP92], [BP98]. These techniques are less known in the numerical linear algebra community but have some attractive features, such as global quadratic or cubic convergence of iterations (right from the start). In particular, such a convergence is guaranteed if we have a real symmetric input matrix A whose invariant subspace associated with the eigenvalues of A from a fixed real interval J is well-conditioned. The condition of such a subspace is quantitatively measured by $ir(J)$, the *isolation ratio* of J (according to our definition 4.1, which we borrow from [P87]), and we supply some effective techniques, which either define an interval associated with a well-conditioned subspace or enable us to partition the original problem for approximating k eigenvalues of A into two subproblems of smaller sizes. The techniques of this paper can be also combined with some other known methods, to improve their performance, in particular, with the divide-and-conquer method for the symmetric eigenproblem (see our remark 10.1).

Our paper is organized as follows: We introduce some definitions and recall some fast methods for the evaluation of the characteristic polynomial of A and its derivative, in section 2. We recall the bisection algorithm, in section 3. In section 4, we outline our main algorithm and define the two basic concepts, of a splitting point and an isolation ratio. To ensure stronger isolation of the eigenvalues, we use bisection in section 5 and the double exponential sieve process of [BOT] in section 6. Our algorithm 7.1 of section 7 approximates well-isolated (clusters of) the eigenvalues of A. In section 8, we summarize our first eigenvalue algorithm, which combines the two stages: of isolation (via the algorithms of sections 5 and 6) and subsequent rapid approximation (via algorithm 7.1). The algorithms of sections 7 and 8 are further accelerated in section 10. In sections 9 and 11, we analyze the algorithms of sections 7 and 10, respectively, and prove their global superlinear convergence. In section 12, we summarize the resulting computational complexity estimates. In section 13, we show the results of numerical experiments.

Acknoweldgements. The authors are greatful to the referee for a series of helpful comments and suggestions. The research by Victor Y. Pan was supported by NSF Grants CCR 9020690 and CCR 9625344 and by Professional Staff Congress of CUNY Awards Nos. 665301, 666327, 667340 and 668365.

2 Definitions and auxiliary results on the evaluation of the characteristic polynomial and its derivative

We will start with some basic definitions. Hereafter, $A = (a_{i,j},\ i,\ j = 1, \dots, n)$ denotes an $n \times n$ real symmetric tridiagonal (rst) matrix, with n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$; $\alpha_i = a_{i,i}$, $\beta_j = a_{j-1,j} = a_{j,j-1}$, A_i denotes the $i \times i$ leading principal submatrix of $A = A_n$, $p_i(x) = \det(\ xI_i - A_i\ )$ denotes the characteristic polynomial of A_i, $p(x) = p_n(x) = \det(\ xI_n - A\ )$. We have

$$p_0(x) = 1, \quad p_1(x) = x - \alpha_1, \quad p_j(x) = (x - \alpha_j)\, p_{j-1}(x) - \beta_j^2 p_{j-2}(x)\ , \quad j = 2, \dots, n\ . \tag{2.1}$$

Next we will recall the known algorithms for the basic steps of both classical bisection algorithm and our acceleration, that is, for the evaluation of $p(x)$ and its derivative. We may precompute the values β_h^2 for all h in $n-1$ multiplications and then apply (2.1) in order to evaluate the sequence $p_1(x), \dots, p_n(x)$, for any fixed x, in $2n-3$ multiplications and $2n-1$ additions. The number of sign agreements in this sequence (provided that $\text{sign}(p_j(\lambda)) = \text{sign}(p_{j-1}(\lambda))$ if $p_j(\lambda) = 0$) equals the number $n_-(x)$ of the eigenvalues of A that are less than x ([GL], p. 438; [Par], p. 131).

We will employ a nonlinear recurrence ([PS], [BP92], p. 120) to speed-up the computation and to avoid overflow. Write $R_i = R_i(x) = p_i(x)/p_{i-1}(x)$, precompute β_i^2 for all i (this stage does not depend on x), compute $\alpha_i - x$ for all i, and then recursively compute

$$R_{j+1} = x - \alpha_{j+1} - \beta_{j+1}^2 / R_j \quad \text{if} \quad R_j \neq 0, \qquad R_{j+2} = x - \alpha_{j+2} \quad \text{if} \quad R_j = 0, \quad j = 1, \dots, n-1.$$

By the definition of R_j, $n_-(x)$ equals the number of indices j with $R_j \geq 0$. In $3n \pm O(1)$ arithmetic operations, we obtain all $x - \alpha_i$ and all R_j for each x, and then $n_-(x)$ is recovered in n comparisons of R_j with 0, that is, we need a total of $4n \pm O(1)$ operations. If we only need to evaluate R_j, for $j = 1, \dots, n$, and if we ignore the tests that determine if $R_j \neq 0$, then the total number of operations used decreases to $3n \pm O(1)$.

We also extend this approach to the computation of $p'(x)/p(x)$, by recursively computing the ratios $R_j^* = R_j^*(x) = p_j'(x)/p_j(x)$ as follows ([PS], [BP92], p.120):

$$\begin{aligned} &R_0^* = 0, \qquad R_1^* = 1/(x - \alpha_1)\ , \\ &R_j^* = R_{j-2}^* + ((x - \alpha_j)(R_{j-1}^* - R_{j-2}^*) + 1)/R_j\ , \quad j = 2, 3, \dots, n. \end{aligned}$$

This will give us $p'(x)/p(x) = p_n'(x)/p_n(x)$ in $5n \pm O(1)$ additional arithmetic operations. Unlike the computation of $n_-(x)$, the latter evaluation involves no comparisons.

We will measure the computational complexity by the number of operations involved and also by the number of the evaluations of $n_-(x)$, $p(x)$, and $p'(x)/p(x)$.

Hereafter, all logarithms are to the base 2.

3 Bisection algorithm

In this section we will recall the classical bisection algorithm for the eigenvalues of an rst matrix.
Algorithm 3.1, *bisection algorithm.*
Input: an $n \times n$ rst matrix A, three real numbers, a, b, and t, $0 < 2t < b - a$.
Output: approximations (within the absolute error bound t) to all the eigenvalues of A in the semi-open interval $[a, b) = \{ x : a \le x < b \}$.
Initialize: compute $n_-(a)$ and $n_-(b)$; if $n_-(a) = n_-(b)$ then exit; otherwise call the interval $[a, b)$ *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 them, output its midpoint and the number of the eigenvalues of A lying in it. □ Let $\lambda_{j+1}, \ldots, \lambda_{j+k}$ denote the eigenvalues of A in $[a, b)$,

$$\lambda_{j+k+1} < a \le \lambda_{j+k} \le \ldots \le \lambda_{j+1} < b \le \lambda_j \ . \tag{3.1}$$

By definition, a suspect interval contains at least one eigenvalue. Therefore, at each recursive step, there are at most k suspect intervals, and at most k evaluations of $n_-(x)$ are required, at the midpoints of all the suspect intervals. In the s-th recursive step, each suspect interval has length $(b-a)/2^s$, so that in $\lceil H \rceil$ steps, for

$$H = \log((b-a)/(2t)) \ , \tag{3.2}$$

we output the solution. Therefore, the bisection algorithm outputs desired approximations to the k eigenvalues of A in $[a, b)$ by using $4kn\lceil H \rceil + O(k)$ arithmetic operations and comparisons.

4 Accelerated algorithms (preliminaries)

We next outline two algorithms that converge with nearly quadratic or cubic rates and support the complexity bounds of order $kn \log^2 H$ (versus $4kn\lceil H \rceil + O(k)$ above). We need two basic definitions.

Definition 4.1 *Let $J = [\ \mu - \ell, \mu + \ell\ )$ be a real interval. Let $D = \min_{\lambda_i \notin J} |\ \mu - \lambda_i\ |$ denote the distance from the midpoint μ to the closest eigenvalue of A lying outside J. Then we write $|J| = 2\ell$ and define (cf. [P87], [P89]) the* isolation ratio *of J, $ir(J) = 2D/|J| = D/\ell$.*

Definition 4.2 *$x \in J$ is called a* splitting point *of $J = [\mu - \ell, \mu + \ell)$ if*

$$n_-(\mu - \ell) < n_-(x) < n_-(\mu + \ell) \ .$$

Whenever our algorithm applied to $J = [\mu - \ell, \mu + \ell)$ computes a splitting point x, we interrupt the computations and restart them separately for each of the two subintervals $[\mu - \ell, x)$ and $[x, \mu + \ell)$, each of them containing fewer eigenvalues of A. Since $[a, b)$ contains k eigenvalues, we encounter at most $k - 1$ splitting points and need at most $k - 1$ such interruptions.

We may restate our task as follows:
Problem 4.1, *splitting or approximation of the eigenvalue in a fixed interval.*

Input: an $n \times n$ rst matrix A, real a, b, t, and the integers k, $n_-(a)$, $n_-(b)$ such that $t > 0$, $a + 2t < b$, $n \geq k = n_-(b) - n_-(a) > 0, n > 3$.

Output: a splitting point x satisfying the relations of definition 4.2 or an approximation μ, within t, to all the eigenvalues of A lying in $[a, b)$.

As our first step towards the solution of problem 4.1, we are going to compute either a splitting point of the interval $J = [a, b)$ or its subinterval J^* containing the same k eigenvalues of A as J but having a greater isolation ratio. This will be our goal in the next two sections.

5 Linear increase of an isolation ratio via bisection

The next result is immediately verified.

Lemma 5.1 *Let $ir(J_G) = 1 + u$ for a suspect interval J_G computed at the G-th recursive step of algorithm* 3.1. *Then the bisection of J_G either outputs two suspect intervals of half length (and then the midpoint of J_G is a splitting point) or defines a single suspect subinterval J_{G+1} of J_G such that*

$$2\ |J_{G+1}| = |J_G|\ , \qquad ir(J_{G+1}) \geq 1 + 2u\ .$$

If h bisection steps have been applied to the interval J_G of lemma 5.1 such that $ir(J_G) \geq 1 + u$ and if neither of them computes a splitting point for J_G, then their output suspect subinterval J_{G+h} has length $|J_G|/2^h$ and has an isolation ratio of at least $1 + 2^h u$. If

$$ir(J_G) \geq 3 \text{ or , equivalently, } u \geq 2, \tag{5.1}$$

then $1 + 2^h u \geq 1 + 2^{h+1}$, so that $ir(J_Q) \geq N$ for $Q = G + h$, $h = \lceil \log(N-1) \rceil - 1$.

6 Double exponential sieve for increasing an isolation ratio

In this section, we will yield an interval J_G with its isolation ratio of at least 3, by applying the *double exponential sieve* algorithm of [BOT], originally proposed for the approximation of polynomial zeros. For simplicity, let $\mu = \ell = 1$, let A have a unique (single or multiple) eigenvalue λ in the interval $J = [0, 2)$, and see our comments on the extension to the general case at the end of this section.

One bisection step specifies if λ lies in $[0, 1)$ or $[1, 2)$. Without loss of generality, let $0 \leq \lambda < 1$; then A has no eigenvalues in [1,2). Now, the idea is to compute $n_-(x_i)$ not for $x_i = 2^{-i}$ but for $x_i = 2^{-2^i}$, $i = 1, 2, \ldots$, and this is clearly more effective if λ lies near 0. Formally, for a fixed t, $\frac{1}{2} > t > 0$, we seek two real values $\overline{a}$ and $\overline{b}$ such that

$$0 \leq \overline{a} \leq \lambda < \overline{b} \leq 1$$

and either

$$\overline{b} - \overline{a} \leq 2t\ , \tag{6.1}$$

(in which case $|\ \lambda - (\ \overline{a} + \overline{b}\ )/2\ | \leq t$) or else

$$\overline{b} \leq 2\overline{a} \tag{6.2}$$

(in which case the isolation ratio of the interval $[\overline{a}, \overline{b})$ is at least 3). We first successively evaluate $n_-(x_i)$ for $x_i = 2^{-2^i}$, $i = 1, \ldots, g_1$, where

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

If $g_1 = \lceil \log\log(1/(2t)) \rceil$, then $0 \le \lambda \le 2^{-2^{g_1}} \le 2t$, and we satisfy (6.1) for $\overline{a} = 0$, $\overline{b} = x_{g_1}$. Otherwise, λ lies in the interval $J_{g_1} = \{ \lambda, \overline{a}_1 = x_{g_1} \le \lambda < x_{g_1 - 1} = \overline{b}_1 \}$. If $g_1 = 1$, then we write $x_0 = 1$, $J_{g_1} = [1/4, 1)$, so that $ir(J_{g_1}) \ge 5/3$, and we increase this ratio to at least 11/3 in at most two applications of lemma 5.1. Otherwise, if $g_1 > 1$, we apply the same (double exponential sieve) procedure to the interval J_{g_1}. Let g_2 denote the number of the evaluations of $n_-(x)$ in this application. Then we either satisfy (6.1) by setting $\overline{a} = \overline{a}_1$, $\overline{b} = \overline{a}_2 = \overline{a}_1 + (\overline{b}_1 - \overline{a}_1)2^{-2^{g_2}}$ or else obtain that

$$\overline{a}_2 = \overline{a}_1 + (\overline{b}_1 - \overline{a}_1)2^{-2^{g_2}} \le \lambda < \overline{a}_1 + (\overline{b}_1 - \overline{a}_1)2^{-2^{g_2 - 1}} = \overline{b}_2 \ .$$

If $g_2 > g_1$, then $\overline{b}_2 - \overline{a}_2 < \overline{a}_1$ since $\overline{b}_1 - \overline{a}_1 < \overline{b}_0 - \overline{a}_0 = 1$, and we satisfy (6.2) for $\overline{a} = \overline{a}_2$, $\overline{b} = \overline{b}_2$. If $g_2 = g_1$, we argue as follows: According to the definition of $\overline{a}_1$ and $\overline{b}_1$ we have $0 < \overline{b}_1 < 1$ and $\overline{a}_1 = \overline{b}_1^2$. This yields (for $g_1 = g_2$):

$$\overline{b}_2 - \overline{a}_2 = (\overline{b}_1 - \overline{a}_1)(2^{-2^{g_1 - 1}} - 2^{-2^{g_1}}) = (\overline{b}_1 - \overline{a}_1)^2$$

$$= (\overline{b}_1 - \overline{b}_1^2)^2 = \overline{b}_1^2(1 - \overline{b}_1)^2 < \overline{b}_1^2 = \overline{a}_1.$$

It remains to consider the case where $g_2 < g_1$, in which we apply for the third time the double exponential sieve procedure, this time to the interval $[\overline{a}_2, \overline{b}_2)$. Recursively, we arrive at a decreasing sequence of positive integers $\{g_1, g_2, \ldots, g_u\}$, such that (6.1) and/or (6.2) are satisfied for $\overline{a} = \overline{a}_u$, $\overline{b} = \overline{b}_u$. Then $u \le g_1$, and the overall number of the evaluations of $n_-(x)$ in the entire process, including the initial evaluation of $n_-(1)$, is at most

$$G = 1 + \sum_{i=1}^{u} g_i \le 1 + (g_1 + 1)g_1/2 \ , \quad g_1 = \lceil \log\log(1/(2t)\,) \rceil \ . \tag{6.3}$$

Extension. We may immediately extend the above process from the input interval $[0, 2)$ to any interval J of length $2l$, which may contain several eigenvalues of A, so that, in at most $G^* = (g_1^* + 1)g_1^*/2 + 1$ evaluations of $n_-(x)$, for $g_1^* = \lceil \log\log(l/(2t)\,) \rceil$, we either compute a splitting point x in J or output a subinterval $\widetilde{J} = J_{G^*}$ of J containing exactly the same eigenvalues of A as J and such that $|\widetilde{J}| \le 2t$ and/or $ir(\widetilde{J}) \ge 3$.

Remark 6.1 *The equation (6.3) gives us overly pessimistic estimates. Indeed, under the uniform probability distribution of random λ on [0,1), one may easily deduce that Probability$\{g_1 > i\} = 2^{-2^i}$, and similarly, Probability$\{g_k > i\}$ rapidly decreases to 0 as i grows, for all k.*

7 Accelerated computation in the case of very good initial isolation

The algorithms of sections 5 and 6 enable us to isolate the eigenvalues in a fixed interval and thus to reduce problem 4.1 to the following problem:

Problem 7.1, *splitting or approximation of the eigenvalues lying in a fixed isolated interval.*

Input: an $n \times n$ rst matrix A, $n > 1$, real a^*, b^*, C, N, t and integers j, k, $n_-(a^*)$, $n_-(b^*)$ such that

$$t > 0, b^* > a^* + 2t, j \ge 0, n \ge k \ge 1, j + k \le n, C = (b^* - a^*)(N - 1)/2, N > 6n - 7, \tag{7.1}$$

$$\lambda_{j+k+1} \le a^* - C < a^* \le \lambda_{j+k} \le \ldots \le \lambda_{j+1} < b^* < b^* + C \le \lambda_j, \tag{7.2}$$

where $\lambda_0 = +\infty$, $\lambda_{n+1} = -\infty$. (The interval $J^* = [a^*, b^*)$ contains exactly k eigenvalues of A, and we have $ir([a^*, b^*)) \geq N$.)

Output: a real μ such that

$$\lambda_{j+k} - t \leq \mu \leq \lambda_{j+1} + t \tag{7.3}$$

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

Once these output values are available, we have either

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

(in which case $\mu - t \leq \lambda_{j+k} \leq \ldots \leq \lambda_{j+1} < \mu + t$) 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, $\lambda_{j+k} < \mu - t \leq \lambda_{j+1}$).

For large N and C, the k eigenvalues of A in [a,b) can be viewed as a cluster well isolated from the other eigenvalues of A. Our next goal is to show that *Newton's iteration* for computing the k-fold zero of $p(x)$ rapidly converges to such a cluster, defining a recursive algorithm for problem 7.1. (We will analyze this algorithm in section 9 and, in particular, will estimate how many its recursive steps are sufficient to solve problem 7.1.)

Algorithm 7.1

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

Recursive step i, $i = 0, 1, \ldots$. Compute

$$\mu_i = b_i - k\, p(b_i)/p^{'}(b_i) \ , \tag{7.4}$$

$$M_i = \max\left\{\frac{n - j - k}{N_i + 1} \ , \ \frac{j}{N_i - 1}\right\} \ , \tag{7.5}$$

$$t_{i+1} = 2(b_i - a_i)M_i \,/\, (k - 2\,M_i). \tag{7.6}$$

If $t_{i+1} \leq t$, then output $\mu = \mu_i$, compute and output $n_-(\mu - t)$, $n_-(\mu + t)$, and end the computation. 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 $\mu = a_{i+1}$, $n_-(\mu - t)$, $n_-(\mu + t)$ and stop. Otherwise, if $n_-(b_{i+1}) < n_-(b^*)$, output $\mu = b_{i+1}$, $n_-(\mu - t)$, $n_-(\mu + t)$ and stop. Otherwise, set

$$N_{i+1} = \frac{(N_i - 1)(k - 2M_i)}{4M_i} - 1 \ , \tag{7.7}$$

where M_i is defined by (7.5), and go to step $i + 1$. □

Note that until the computation stops, the equation

$$2t_i = b_i - a_i \tag{7.8}$$

is maintained for all i. Furthermore, equations (7.6)-(7.9) together imply that

$$t_{i+1} = (N_i - 1)t_i/(N_{i+1} + 1) \ \textit{for all } i. \tag{7.9}$$

8 Summary of an eigenvalue algorithm

By combining our algorithms of sections 4-7, we specify our first algorithm for problem 4.1.

Algorithm 8.1

1) Apply the double exponential sieve process of section 6 to the interval $J = [a, b)$. The process either solves problem 4.1 or outputs a subinterval $\widetilde{J}$ of $[a, b)$ containing the same eigenvalues as the input interval $[a, b)$ and having an isolation ratio of at least 3.

2) In the latter case, apply $h = \lceil \log((N-1)/2) \rceil$ bisection steps (supporting lemma 5.1) to the interval $\widetilde{J}$. This either solves problem 4.1 or outputs a subinterval J_Q of $\widetilde{J}$ sharing all the eigenvalues with $\widetilde{J}$ and having an isolation ratio of at least N.

3) In the latter case, apply algorithm 7.1 to the interval $J_Q = [a^*, b^*)$.

There can be at most $k-1$ splitting points of J; therefore, approximating all the k eigenvalues of A in $[a, b)$ requires at most $2k-1$ calls for algorithm 8.1.

Remark 8.1 *The latter (worst case) estimate of $2k-1$ is overly pessimistic on the average case, according to our experiments and some theoretical consideration.*

9 Analysis of algorithm 7.1

Analyzing algorithm 7.1, we will simplify the notation by replacing a^* by a and b^* by b, and will recall from (7.2) that

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

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

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

where $C = (b-a)(N-1)/2$ (as in (7.1)). We also recall that

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

We will start with a high level argument and will supply more details later on. For large N and C, the eigenvalues of A lying in the interval $[a, b)$ are strongly isolated from all other eigenvalues of A. It follows that the sum of the reciprocals of the former eigenvalues, which are the eigenvalues of $A - xI$ lying in the interval $[a-x, b-x)$, dominate in the sum $-p'(x)/p(x) = \sum_{r=1}^{n} 1/(\lambda_r - x)$ if $a \le x < b$. Therefore, $-p'(x)/(kp(x))$ closely approximates the average value $\frac{S_0}{k} = \frac{1}{k}\sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - x}$ of these k absolutely largest reciprocals. On the other hand, the value k/S_0 lies between $\lambda_{j+k} - x$ and $\lambda_{j+1} - x$; consequently, the value $-kp(x)/p'(x)$ lies in or near the interval $[\lambda_{j+k} - x, \lambda_{j+1} - x)$. Knowing $-kp(x)/p'(x)$ for $x = b_0$, we will cover the latter unknown interval by a small interval lying about $\mu = \mu_0$ of (7.3) and having a higher isolation ratio than $[a, b)$.

We are going to formalize this argument and to apply it recursively, proving both rapid growth of the isolation ratio and rapid decrease of the interval length in this recursive process until it ends with the solution of Problem 4.1, that is, with computing a splitting point or approximating all the eigenvalues of A lying in the interval $[a, b)$.

Let us next examine step i, for $i = 0$. Write $a = a_0$, $b = b_0$, $M_0 = M$ (to simplify the notation). The inequalities listed in the beginning of this section imply that

$$a - b \le \lambda_r - b < 0 \ , \qquad r = j+1, \ldots, j+k, \tag{9.1}$$

$$\lambda_r - b \le (N+1)(a-b)/2\ , \qquad r = j+k+1,\dots,n, \tag{9.2}$$

$$\lambda_r - b \ge (N-1)(b-a)/2, \qquad r = 1,\dots,j. \tag{9.3}$$

We begin with proving some auxiliary relations.

Lemma 9.1 *Let* $S_0 = \sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - b}$, $S_1 = \sum_{r=j+k+1}^{n} \frac{1}{\lambda_r - b}$, $S_2 = \sum_{r=1}^{j} \frac{1}{\lambda_r - b}$, $\rho = \frac{S_1+S_2}{S_0}$, $M = \max\{\frac{n-j-k}{N+1}\ ,\ \frac{j}{N-1}\}$ *(cf. (7.5)). Then we have*

$$0 < -S_0 < k/(b-a), \tag{9.4}$$

$$\frac{1}{S_0} + \frac{p(b)}{p'(b)} = \frac{\rho}{(1+\rho)S_0}, \tag{9.5}$$

$$|\rho| \le \frac{2M}{k} \le \frac{2n-2}{(N-1)k} < 1, \tag{9.6}$$

$$\lambda_{j+k} \le (k/S_0) + b \le \lambda_{j+1}. \tag{9.7}$$

Proof. (9.4) follows from (9.1). Observe that

$$-\frac{p'(b)}{p(b)} = \sum_{r=1}^{n} \frac{1}{\lambda_r - b_0} = S_0 + S_1 + S_2 = (1+\rho)S_0,$$

deduce that
$\frac{p(b)}{p'(b)} = -\frac{1}{(1+\rho)S_0}$, $\frac{1}{S_0} + \frac{p(b)}{p'(b)} = \frac{1}{S_0}(1 - \frac{1}{1+\rho}) = \frac{\rho}{(1+\rho)S_0}$, and we arrive at (9.5). Next deduce from (9.2) and (9.3) that

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

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

and consequently,

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

Combine the latter bound on $|S_1+S_2|$ with the definitions of M and ρ and with (9.4) and obtain (9.6). Finally, deduce from (7.2) that

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

and obtain (9.7). □

For convenience, let us display (7.4) and (7.6) for $i = 0,\ a_0 = a,\ b_0 = b:$

$$\mu_0 = b - kp(b)/p'(b), \tag{9.8}$$

$$t_1 = 2(b-a)M/(k-2M). \tag{9.9}$$

Our next goal is to prove the following result.

Lemma 9.2 $\lambda_{j+k} - t_1 \le \mu_0 \le \lambda_{j+1} + t_1$, *that is, (7.3) holds for* $\mu = \mu_0, t = t_1$.

Proof: Deduce from (9.6) that

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

By combining the latter inequality with the bound $k/|S_0| \le b-a$ of (9.4) and with equations (9.5) and (9.9), we obtain that

$$t_1 \ge \frac{k|\rho|}{|(1+\rho)S_0|} = k \ \left| \ \frac{p(b)}{p'(b)} + \frac{1}{S_0} \ \right| \ . \tag{9.10}$$

Now, we deduce from (9.7) and (9.10) that

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

and due to (9.8), we conclude that $\lambda_{j+1} + t_1 \ge \mu_0$. Similarly, we deduce from (9.7), (9.8) and (9.10) that $\lambda_{j+k} - t_1 \le \mu_0$. □

Let us deduce from lemma 9.2 that the first step of algorithm 7.1 either solves problem 7.1 correctly or reduces it to the same problem, but with a, b, N replaced by a_1, b_1, N_1, respectively. Indeed, if $t_1 \le t$, then (7.3) holds for $\mu = \mu_0$, $t = t_1$. Otherwise, we have three cases:

Case a). $n_-(\mu_0 - t_1) > n_-(a)$. Then $\lambda_{j+k} < \mu_0 - t_1 \le \lambda_{j+1}$, and $\mu = \mu_0 - t_1$ satisfies (7.3) for $t = 0$.

Case b). $n_-(\mu_0 - t_1) = n_-(a)$, $n_-(\mu_0 + t_1) < n_-(b)$. Then $\lambda_{j+k} \le \mu_0 + t_1 < \lambda_{j+1}$, and $\mu = \mu_0 + t_1$ satisfies (7.3) for $t = 0$.

Case c). $n_-(\mu_0 - t_1) = n_-(a)$, $n_-(\mu_0 + t_1) = n_-(b)$. Then $\mu_0 - t_1 \le \lambda_{j+k} \le \lambda_{j+1} < \mu_0 + t_1$, and the eigenvalues $\lambda_{j+k}, \ldots, \lambda_{j+1}$ are bracketted in the interval $[a_1, b_1)$ of length $2t_1$ for $a_1 = \mu_0 - t_1, b_1 = \mu_0 + t_1$.

It remains to assume that case c) occurs in the i-th step of algorithm 7.1, for $i = 0, 1, \ldots$. (Otherwise, we yield the desired splitting of the eigenvalues lying in the interval $[a, b)$ and solve problems 4.1 and 7.1 correctly as lemma 9.2 is easily extended to the pairs of μ_i and t_{i+1} for all i.) If case c) occurs recursively for all i then we will deduce that the algorithm 7.1 converges quadraticly until it yields either the desired splitting or the approximation of the eigenvalues lying in $[a, b)$. Let us first show that the isolation ratio of the interval $[a_i, b_i)$ is at least N_i, for $i = 1$ (and similar property actualy holds for $i = 2, 3, \ldots$). We observe that the distance from μ_0 to the closest eigenvalue of A lying outside the interval $[a_1, b_1)$ is at least $C - t_1$ (for C of (7.1)). Therefore, the isolation ratio of this interval is at least

$$(C/t_1) - 1 = \frac{b-a}{2t_1}(N-1) - 1 = \frac{(N-1)(k-2M)}{4M} - 1 = N_1. \tag{9.11}$$

(The latter equation holds due to (7.7) for $i = 0$.)

Due to (7.1), (7.5), we have $k > 6M$ and, therefore, $N_1 + 1 > N - 1$, $2t_1 = b_1 - a_1 < b - a$ (cf. (9.9), (9.11)). Our next result gives us a stronger bound on N_{i+1} via N_i and n.

Lemma 9.3 *Suppose that algorithm 7.1 is performed, and case c) occurs recursively for* $i = 0, 1 \ldots$. *Then the computed values* N_i *satisfy the following relations:* $N_{i+1}+1 \ge (N_i-1)(N_i-2n+1)/(4n-4)$, *i=0, 1, ...* .

Proof. Substitute the bound $M_i \le \frac{n-1}{N_i - 1}$, implied by (7.5), into (7.7) and deduce that

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

and, consequently,

$$\begin{array}{rcl} (4n-4)(N_{i+1}+1) & \ge & (N_i - 1)^2 (k - 2M_i) \\ & \ge & (N_i - 1)^2 (k - (2n-2)/(N_i - 1)) \\ & = & (N_i - 1)(kN_i - 2n - k + 2) \\ & \ge & (N_i - 1)(N_i - 2n + 1) \ . \end{array}$$

□

Remark 9.1 *Note that the latter inequality becomes stronger for $k > 1$, and so are the estimates of lemmas 9.4 amd 9.5 and equation (9.12) below.*

By combining lemma 9.3 with equations (7.8), (7.9), we obtain the next result.

Lemma 9.4 *Under the assumptions of lemma 9.3, we have*

$$t_{i+1}/t_i = (b_{i+1} - a_{i+1})/(b_i - a_i) \le (4n-4)/(N_i - 2n + 1), i = 0, 1, \ldots .$$

Lemmas 9.3 and 9.4 show the quadratic growth of N_i and (consequently) quadratic decrease of the length $|b_i - a_i| = 2t_i$ of the intervals $[a_i, b_i)$ as $i \to \infty$ (assuming that N is large enough). Let us next show some specific estimates.

Lemma 9.5 *Under the assumptions of lemma 9.3, let $N = \gamma_0 n$, $\gamma_0 > 6$, $\gamma_u = 4\gamma_u^* + 2$, $u = 0, 1, \ldots$. Then $N_{i+u} > (4(\gamma_u^*)^{2^i} + 2)n$, $t_{i+u+1} < (\gamma_u^*)^{1-2^{i+1}} t_u$, $\gamma_{u+i+1}^* > (\gamma_u^*)^{2^{i+1}}$, $i = 0, 1, \ldots; u = 0, 1, \ldots$.*

Proof. Assuming that $N = \gamma_0 n$, $\gamma_0 > 6$, recursively deduce that

$$N_{i+1} > \gamma_{i+1} n \ , \quad \gamma_{i+1} = (\gamma_i - 2)\, \gamma_i / 4 > 6 \ , \quad i = 0, 1, \ldots \ .$$

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

$$N_{i+u} > (4(\gamma_u^*)^{2^i} + 2)n \ , \quad \frac{t_{i+u+1}}{t_{i+u}} = \frac{N_{i+u} - 1}{N_{i+u+1} + 1} \le \frac{4n - 4}{N_{i+u} - 2n + 1} < (\gamma_u^*)^{-2^i} \ ,$$

$$t_{i+u+1} < (\gamma_u^*)^{1-2^{i+1}} t_u \ , \qquad i = 0, 1, \ldots \ . \qquad □$$

For $u = 0$, we obtain that $t_{i+1} < (\gamma_0^*)^{1-2^{i+1}}(b-a)/2$ (and a little stronger bound can be obtained for $u > 0$). Therefore,

$$t_i < t \quad \text{if} \quad i \ge \lceil \log(1 + \log(\frac{b-a}{2t})/\log \gamma_0^*) \rceil \ ,$$

or, equivalently, if $i \ge \widehat{T}(\gamma_0^*)$ for

$$\widehat{T}(\gamma_0^*) = \lceil \log(1 + H/\log \gamma_0^*) \rceil \qquad (9.12)$$

and for H of (3.2). In particular, we have

$$\widehat{T}(2) = \lceil \log(H+1) \rceil \ , \quad \text{for} \quad N = 10n \ ,$$

$$\widehat{T}(16) = \lceil \log(H+4) \rceil - 2 \ , \quad \text{for} \quad N = 66n \ .$$

Summarizing, we obtain the following result.

Theorem 9.1 *Let $N = \gamma_0 n$, $\gamma_0 > 6$, $\gamma_i = 4\gamma_i^* + 2$, $i = 0, 1, \ldots$, and let $\widehat{T}(\gamma_0^*)$ be defined by (9.12). Then at most $\widehat{T}(\gamma_0^*)$ recursive steps of algorithm 7.1 suffice in order to solve problem 7.1.*

10 Acceleration of algorithms 7.1 and 8.1

According to our comments given early in section 9, a splitting point for the eigenvalues of A lying in the interval $[a, b)$ would have been approximated more closely, if we had available a better approximation to k/S_0 than one by the value $-p'(x)/(kp(x))$. Such an improved approximation to k/S_0 can be easily computed if some approximations λ_r^* to the eigenvalues λ_r of A for $r \leq j$ and $r > j+k$ are available when we apply algorithm 7.1 at stage 3 of algorithm 8.1. Indeed, in this case, we may subtract the reciprocals $1/(\ \lambda_r^* - b_i\ )$ for $r \leq j$ and $r > j+k$ from the value $-p'(b_i)/p(b_i)$ and substitute the resulting value into (7.4), thus decreasing the influence of these remote eigenvalues on the sum of the reciprocals, $-p'(b_i)/p(b_i)$. Our further analysis in section 11 and our experiments show that this accelerates the convergence and enables us to decrease the lower bound on N from $6n$ to $2 + \sqrt{6n}$. Next, we will specify the first recursive step of the modified algorithm (accelerating step 0 of algorithm 7.1) assuming for simplicity that the available approximations λ_r^* to all the eigenvalues λ_r of A, for $r = 1, \ldots, j, j+k+1, \ldots, n$, satisfy the bound

$$|\lambda_r^* - \lambda_r| \leq t^* = (b-a)/2 \ . \tag{10.1}$$

In fact, we may ensure this assumption by arranging the steps of algorithm 7.1 to work always with the *largest* of the available suspect intervals output at stage 2.

We now modify algorithm 7.1 by replacing the values μ_i of (7.4) by the values

$$\mu_i^* = b_i + k/q_i, \quad q_i = -\frac{p'(b_i)}{p(b_i)} - \sum_{r=1}^{j} \frac{1}{\lambda_r^* - b_i} - \sum_{r=j+k+1}^{n} \frac{1}{\lambda_r^* - b_i} \ , \quad i = 0, 1, \ldots \ .$$

(This requires $3(n-k)$ extra arithmetic operations for each i.) We will cite this modification of algorithm 7.1 as **algorithm 10.1** and the respective modification of algorithm 8.1 as **algorithm 8.1a**.

In the next section, we will show rapid convergence of algorithm 10.1 for $N = \Theta n^{1/2} + 2$, $\Theta > 4$, and will indicate an extension to any $\Theta > \sqrt{6}$.

Remark 10.1 *[BP91], [BP92] use Newton's iteration to ensure faster convergence of the divide-and-conquer (d.-c.) eigenvalue algorithms [C], [DS]. We may further improve this iteration, by extending the above acceleration techniques applied in algorithm 10.1 (cf. also [P87], [P94]). Their power is accentuated in the case of the d.-c. algorithms, which readily furnish us with splitting points for the eigenvalues of A (cf. [BP91], [BP92], [BP98]).*

11 Analysis of algorithm 10.1

We will analyze algorithm 10.1 by extending our analysis from section 9. As we will follow the line of section 9, we will only specify the changes due to the transition to algorithm 10.1. At first, we will extend lemma 9.1. For $S_0 = \sum_{r=j+1}^{j+k} \frac{1}{\lambda_r - b}$, we write

$$q_0 = S_0 + \sum_{r=1}^{j} d_r + \sum_{r=j+k+1}^{n} d_r \ , \tag{11.1}$$

$$d_r = \frac{1}{\lambda_r - b} - \frac{1}{\lambda_r^* - b} = \frac{\lambda_r^* - \lambda_r}{(\lambda_r - b)(\lambda_r^* - b)} \quad \text{for all } r, \tag{11.2}$$

$$\hat{\rho} = (q_0 - S_0)/S_0, \tag{11.3}$$

$$\rho^* = 2(n-k)/(\,(N-1)(N-2)k), \tag{11.4}$$

$$t_1^* = 2(n-k)(b-a)/(\ (N-1)(N-2)k(1+\rho^*)). \tag{11.5}$$

Instead of the bound $N > 6n - 7$ of (7.1), we now impose a weaker lower bound on N, that is,

$$N > 1.5 + \sqrt{2n - 1.75}. \tag{11.6}$$

Now, in addition to lemma 9.1, we have the following relations.

Lemma 11.1

$$|q_0 - S_0| \le (n-k)/((N-1)(N-2)t^*), \tag{11.7}$$

$$|\hat{\rho}| \le \rho^* < 1, \tag{11.8}$$

$$|(k/S_0) - (k/q_0)| = |(k/S_0)\hat{\rho}/(1+\hat{\rho})| \le t_1^*. \tag{11.9}$$

Proof Recall that

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

for $r \le j$ and for $r > j + k$. Combine the latter bounds with (10.1) and extend them as follows:

$$\frac{1}{|\lambda_r^* - b|} \le \frac{2}{(N-2)(b-a)}.$$

Combine these inequalities with (10.1), (11.1), and (11.2) and obtain (11.7).

Deduce from equations (9.4), (10.1), (11.3), (11.4) and (11.7) that

$$|\hat{\rho}| \le \rho^*.$$

Combine (11.4) and (11.6) to obtain that $\rho^* < 1$, thus completing the proof of (11.8).

The equation of (11.9) is easily obtained from (11.3). To deduce the inequality of (11.9), combine (9.4), (10.1), (11.3)-(11.5), (11.7) and (11.8). □

We deduce from (9.7) and (11.9) the following extension of lemma 9.2:

Lemma 11.2 $\lambda_{j+k} - t_1^* \le \mu_0^* = b + k/q_0 \le \lambda_{j+1} + t_1^*$.

By this lemma, step 0 of algorithm 10.1 either solves problem 4.1 or, else, brackets the eigenvalues $\lambda_{j+k}, \ldots, \lambda_{j+1}$ in the interval $[a_1^*, b_1^*)$ of length $2t_1$, where $a_1^* = \mu_0^* - t_1^*$, $b_1^* = \mu_0^* + t_1^*$, and t_1^* is defined by (11.5).

The isolation ratio of this interval is at least

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

Write $N = N_0^*$, substitute (11.4), recall that $k > 1$, and obtain the following extension of Lemma 9.3 for $i = 0$:

$$\begin{aligned} (4n-4)(N_1^*+1) &\ge (N_0^*-1)((N_0^*-1)(N_0^*-2) - 2(n-1)) \\ &= (N_0^*-1)((N_0^*)^2 - 3N_0^* + 2n) \end{aligned} \tag{11.10}$$

This defines a lower bound on N_1^* that is substantially greater than N_1 of (9.11) for large $N = N_0$.

It remains to examine algorithm 10.1 in the latter case. Replacing a, b, N by a_1, b_1, N_1^*, we may recursively repeat the computations until we yield a splitting point or $t_i^* \le t$. To complete the analysis, let us estimate the convergence rate and bound the overall number of iterations in the latter case. We immediately observe that for $N = \Theta\, n^{0.5+v}$, $\Theta \ge 1$, we have

$$\Theta\, n^{0.5+3v}/N_1^* = O(1) \text{ as } v \longrightarrow \infty\ ,$$

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

Similarly to (11.10), we may bound the isolation ratios N_{i+1}^* in terms of N_i^* at the next recursive steps, for $i = 1, 2, \ldots$, to arrive at the following extension of lemma 9.3:

Lemma 11.3 *Let the j-th recursive step of algorithm 10.1 produce no spitting points for $j = 0, \ldots, i$. Let N_{j+1}^* denote the isolation ratio of the interval $[a_{j+1}^*, b_{j+1}^*)$ computed in the j-th recursive step of the algorithm. Then $N_{i+1}^* + 1 \ge (N_i^* - 1)((N_i^*)^2 - 3N_i^* + 2n)/(4n-4)$.*

The next lemma extends lemmas 9.4 and 9.5.

Lemma 11.4 *Under the assumptions of lemma 11.3, let $2t_{j+1}^* = b_{j+1}^* - a_{j+1}^*$, $j = 0, 1, \ldots, i$. Let $N = N_0^* = \Theta_0 n^{1/2} + 2$, for $\Theta_0 > 4$, $\Theta_i = 2\Theta_i^* + 2, i = 0, 1, \ldots$. Then $N_{i+u}^* > (2(\Theta_u^*)^{3^i} + 2)n^{1/2} + 2$, $\Theta_{i+u}^* > (\Theta_u^*)^{3^i}$ and, furthermore, $t_{i+u+1}^*/t_{i+u}^* \le 2(\Theta_u^*)^{-3^i}, t_{i+u+1}^* \le (\Theta_u^*)^{1-3^{i+1}/2} t_u^*, i = 0, 1, \ldots; u = 0, 1, \ldots$.*

Proof. Recursively deduce from lemma 11.3 that $N_{i+1}^* > \Theta_{i+1} n^{1/2} + 2$, $\Theta_{i+1} = (\Theta_i^2 - 2)\Theta_i/4, i = 0, 1, \ldots$, and obtain that $N_{i+1}^* > (2\Theta_{i+1}^* + 2)n^{1/2} + 2$, where $\Theta_i = 2\Theta_i^* + 2$, $\Theta_{i+1}^* > (\Theta_i^*)^3, i = 0, 1, \ldots$. This implies the first two inequalities of lemma 11.4. Next, from the definition of t_j^* and N_j^*, we deduce that $t_{i+1}^*/t_i^* = (N_i^* - 1)/(N_{i+1}^* + 1)$. Replace i by $i + u$, apply lemma 11.3 and the first two inequalities of lemma 11.4 and obtain that $t_{i+u+1}^*/t_{i+u}^* \le \frac{4n-4}{(N_{i+u}^*)^2 - 3N_{i+u}^* + 2n} < (\Theta_u^*)^{-3^i}$, $i = 0, 1, \ldots$, for any fixed $u > 0$. This implies the last two bounds of lemma 11.4. □

For $u = 0$, we obtain from lemma 11.4 that

$$t_{i+1}^* < (\Theta_0^*)^{(1-3^{i+1})/2}(b-a)/2$$

(and a little stronger bound can be decuced for $u > 0$). Therefore, for H of (3.2), denoting $\log((b-a)/(2t))$, we obtain that

$$\widetilde{T}(\Theta_0^*) = \lceil (\log(1 + 2H/\log\Theta_0^*))/\log 3 \rceil \tag{11.11}$$

recursive steps of algorithm 10.1 suffice in order to solve problem 7.1 provided that $N = (2\Theta_0^* + 2)n^{1/2} + 2$, $\Theta_0^* > 1$, $\Theta_0 = 2\Theta_0^* + 2 > 4$. In particular, we obtain that

$$\widetilde{T}(2) = \lceil (\log(1 + 2H))/\log 3 \rceil \ , \quad \text{for} \quad N = 6n^{1/2} + 2 \ ,$$

$$\widetilde{T}(16) = \lceil (\log(2 + H) - 1)/\log 3 \rceil \ , \quad \text{for} \quad N = 34n^{1/2} + 2 \ .$$

Remark 11.1 *By choosing $u > 1$ and writing $N = \Theta_0 n^{1/2} + 2$, we may extend (11.11) to similar bounds for any $\Theta_0^* > \sqrt{3/2} - 1$ or, equivalently, for any $\Theta_0 > \sqrt{6}$.*

Remark 11.2 *To accelerate convergence at the expense of performing a little more work per iteration, we may generalize algorithms 7.1 and 10.1 by replacing (7.4) by more general expressions, such as*

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

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

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

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

can be computed by extending the techniques of section 2. (In the extension of the same approach to approximating polynomial zeros, cited in the introduction, the latter value can be easily obtained from the d leading coefficients of the input polynomial $p(x)$, by using either Newton's identities or a simpler known algorithm, cf. pp. 34–35 of [BP94].)

12 Summary of the complexity estimates

We are ready to summarize our previous analysis to estimate the arithmetic complexity of the approximation of the k eigenvalues $\lambda_{j+k}, \dots, \lambda_{j+1}$, assuming (3.1). We recall that the approximation of all the k eigenvalues in the input interval $J = [a, b)$ requires at most $2k-1$ calls for algorithms 8.1 or 8.1a, that is, at most $(2k-1)(T_0 + T_1(N))$ evaluations of $n_-(x)$, for $k \le n$, and $(2k-1)T_2(N)$ evaluations of $p(x)/p'(x)$, where $T_0 = 1 + (g_1 + 1)g_1/2$, for $g_1 = \lceil \log(H-1) \rceil$, and $T_1(N) = \lceil \log((N-1)/2) \rceil$ are the numbers of the evaluations of $n_-(x)$ in each application of the algorithms of sections 5 and 6, respectively, whereas $T_2(N) = \lceil \log(cH + d) \rceil$ is the number of the evaluations of $p(x)/p'(x)$ in each application of one of algorithms 8.1 or 8.1a (compare (3.2) and (6.3)). The constants c and d depend on the choice between algorithms 8.1 and 8.1a and on the choice of the value N. In particular, by choosing algorithm 8.1 and choosing $N = (4\gamma_0^* + 2)n$ for all i, we arrive at $T_2(N) = \widehat{T}(\gamma_0^*)$, where $\widehat{T}(\gamma_0^*)$ satisfies (9.12). Applying algorithms 8.1a and 10.1, we recall about the need for $3n$ extra arithmetic operations in every iteration. By using the worst case operation count for the evaluation of $p(x)$ and $R_n^*(x) = p'(x)/p(x)$ from section 2, we obtain that

$$(2k-1)(n \pm O(1))(4T_0 + 4T_1(N) + \nu T_2(N)) + n - 1 \tag{12.1}$$

arithmetic operations and comparisons suffice for the approximation (within an error tolerance t) of the k eigenvalues of A in $J = [a, b)$ (cf. (3.1)) provided that either $N > 6n$ and $\nu T_2(N) = 8\widehat{T}(\gamma_0^*)$ (compare (9.12)) or $N > \sqrt{6n} + 2$ and $\nu T_2(N) = 11\widetilde{T}(\Theta_0^*)$ (compare (11.11) and remark 11.1), depending on which of algorithms 8.1 or 8.1a we apply. (12.1) implies the two following estimates for the numbers of arithmetic operations involved when our two algorithms are applied in order to approximate the k eigenvalues of (3.1), respectively:

$$(2k-1)(n \pm O(1))[2\lceil\log(H-1)\rceil(1+\lceil\log(H-1)\rceil) + 4\log n + 8\log H \pm O(1)], \tag{12.2}$$

$$(2k-1)(n \pm O(1))\left[2\lceil\log(H-1)\rceil(1+\lceil\log(H-1)\rceil) + 2\log n + 11\log H/\log 3 \pm O(1)\right], \tag{12.3}$$

where $11/\log 3 \approx 6.94$. The bound (12.3) is superior to (12.2), thus suggesting a lower complexity of our second algorithm, which, however, requires some additional assumptions and also is built on the top of the first one. The operation count above includes

$$(2k-1)(n \pm O(1))(T_0 + T_1(N)) = (2k-1)(n \pm O(1))(1 + (g_1+1)g_1/2 + \lceil\log((N-1)/2)\rceil\ )$$

comparisons needed to compute the sign agreements (see section 2), where $g_1 = \lceil\log(H-1)\rceil$ and N is required to exceed a fixed bound (that is, $6n$ or $\sqrt{6n}+2$). This count, however, does not include the cost of the $(2k-1)nT_2(N)$ tests determining whether $R_j = 0$ or $R_j \neq 0$ and involved in the computation of $p'(x)/p(x)$. The estimates (12.1)$-$(12.3) are the worst case estimates, which tend to be overly pessimistic for the average input (cf. remarks 6.1, 8.1 and 9.1).

13 Numerical Experiments

Two sets of numerical experiments have been performed to compare the bisection algorithm with algorithms 8.1 and 8.1a. In the first set of experiments we computed all of the eigenvalues of symmetric tridiagonal matrices, and in the second set we computed a single eigenvalue.

For algorithm 8.1 we set $N = 8n$, and for algorithm 8.1a we set $N = 4\sqrt{n}+2$. (Even though the latter choice of N corresponds to the choice of $\Theta_0 = 4$, $\Theta_0^* = 1$, the cubic convergence was expected and actually observed due to the *strict* inequality $N_{i+1} > (2\Theta_{i+1}+2)n^{1/2}+2$.) For all algorithms, the endpoints of an interval containing all eigenvalues were found by using the Gershgorin circle theorem [GL]. The experiments were performed in standard IEEE double precision arithmetic.

Even though in this paper we described our algorithms performed with infinite precision, they turned out to be robust enough so that their implementation with finite (double) precision only required a single and very minor change and only in the case of algorithm 8.1a. The analysis in section 11 allows us to choose t_{i+1} according to the formula

$$t_{i+1} = \frac{2((n-k)(b_i - a_i)}{(N_i-1)(N_i-2)k(1+\rho_i^*)} \quad .$$

However, with such a choice of t_{i+1} the numerical value of $n_-(\mu_i + t_{i+1})$ was often equal to the numerical value of $n_-(\mu_i - t_{i+1})$. We, therefore, computed t_{i+1} according to the formula

$$t_{i+1} = \max\left(0.9t\ ,\ \frac{2(n-k)(b_i-a_i)}{(N_i-1)(N_i-2)k(1+\rho_i^*)}\right),$$

$$\rho_i^* = \frac{2M_i}{k}$$

(cf. (11.5)), which did not allow the suspect intervals to get too small. This was, of course, just a translation of the above stopping criteria into a language understood by a computer: when t_{i+1} is set to $0.9t$ the algorithm should terminate.

In order to improve the performance of our algorithms in practice, we made some small changes versus sections 7–12 when we implemented algorithms 8.1 and 8.1a. These changes do not affect the worst case complexity bounds for those algorithms.

We stored the suspect intervals in an ordered doubly linked list, allowing us to quickly compute a lower bound for the isolation ratio of a suspect interval from the length of the interval and the distance between the interval and its neighbors.

We used this approach instead of relying on (7.7) when we computed N_{i+1} (lower bounds on the isolation ratios). These calculated lower bounds are at least as large as the lower bound error (7.7).

Secondly, if during the double exponential sieve process we could bound the isolation ratio of the suspect interval by 2, we stopped the double exponential sieve process and then performed a single bisection step to bring the isolation ratio to at least 3. The double exponential sieve process would have anyway used at least one additional evaluation of $n_-(x)$.

Thirdly, in step 2 of algorithm 8.1 (or its modification), we stopped the bisection steps when we computed a lower bound on the isolation ratio that was greater than N; this clearly never required more, and in fact sometimes used fewer, bisection steps than the $\lceil \log((N-1)/2) \rceil$ steps that would otherwise be required for step 2 of Algorithm 8.1.

To describe the final change, let the input interval to the double exponential sieve process be an interval $J = [a, b)$. If we know that there are no eigenvalues in $[a - (b - a), a)$, or that there are no eigenvalues in $[a, b + (b - a))$, then the actual purpose of performing the bisection step at the beginning of the double exponential sieve process is already achieved, and we can skip this step. Because we stored the suspect intervals in an ordered doubly linked list, we could sometimes make that determination and skip the bisection step at the beginning of the double exponential sieve process.

For the numerical experiments, we have chosen t by setting the *relative error*, $t/(\lambda_1 - \lambda_n)$, equal to 10^{-15}, 10^{-11}, and 10^{-7}. We computed t by first computing λ_1 and λ_n with the LAPACK subroutine dstev [A], these eigenvalues could have been alternatively estimated with Gershgorin's circle theorem.

We compared the results computed with the bisection algorithm and algorithms 8.1 and 8.1a with those computed by the LAPACK subroutine. The differences between the eigenvalues computed with algorithms 8.1 and 8.1a and the eigenvalues computed with the LAPACK subroutine were always smaller than the requested precision. The differences between the eigenvalues computed with the bisection algorithm and the eigenvalues computed with the LAPACK subroutine were sometimes up to 44% larger than the requested precision because the LAPACK subroutine does not, of course, compute the exact eigenvalues.

We measured the costs of using the various algorithms both by the actual running time of the algorithms (on an IBM RISC System / 6000 model 530) and by the number of computations of $n_-(x)$ ($4n - O(1)$ operations) and equivalents. The advantage of the first method is that it takes all overheads into account, while the second method is machine and code independent. (The code we wrote was not highly optimized.) We counted computing $p(x)/p'(x)$ [$8n - O(1)$ arithmetic operations] as equivalent to 2 computations of $n_-(x)$, and we counted computing

$$-\sum_{r=1}^{j} \frac{1}{\lambda_r^* - c_i} - \sum_{r=j+k+1}^{n} \frac{1}{\lambda_r^* - c_i}$$

(in $3n - O(k+1)$ operations) as equivalent to 3/4 of the cost of computing $n_-(x)$.

We ran the numerical experiments described in this section on the ten matrices discussed in [PS]. In Tables 13.1–13.3 we give the costs (both in terms of computations of $n_-(x)$ and equivalents and in milliseconds) of computing all the eigenvalues of these ten matrices with the bisection algorithm, algorithm 8.1 and algorithm 8.1a.

When we measured the cost of performing the algorithms by the number of computations of $n_-(x)$ and equivalents, algorithm 8.1a was almost always more efficient than algorithm 8.1. For a relative error of 10^{-15}, algorithm 8.1a required only about 57% of the computations of $n_-(x)$ and equivalents used by the bisection algorithm for all ten matrices. The improvement over the bisection algorithm became smaller as the requested precision was made larger, until for a requested relative precision of 10^{-7} the bisection algorithm used fewer computations of $n_-(x)$ and equivalents than algorithm 8.1a.

When we used actual timings, we have found out that algorithm 8.1 was overall more efficient than algorithm 8.1a. For all ten matrices, algorithm 8.1 used 68% of the time used by the bisection algorithm when the requested relative error was 10^{-15}. The improvement over the bisection algorithm became smaller as the requested precision was made larger, until for a requested relative precision of 10^{-7} the bisection algorithm used less time than algorithm 8.1 for all ten matrices.

These experimental results do not show the quadratic or cubic convergence in the Newton iteration stage of algorithms 8.1 and 8.1a, of course, since besides this stage the algorithms generally include the slower bisection stages (needed for increasing the isolation ratios) and splitting steps. However, the high-order convergence of the new algorithms ($O((\log H)^2)$ operations, as compared to $O(H)$ for the bisection algorithm) can be seen from the relatively slow increase of the cost of performing these algorithms – when compared to the cost of performing the classical bisection algorithm – as the allowed tolerance to the errors is reduced and higher output precision is required; in fact, for matrix 1 the cost of performing algorithm 8.1a was the same for relative tolerances of 10^{-7}, and 10^{-11}. Indeed, the numerical experiments presented in this section do suggest that algorithms 8.1 and 8.1a can give a significant improvement over bisection when the desired output precision is high, but that this improvement disappears for low output precision. This is consistent with the high order convergence properties of algorithms 8.1 and 8.1a.

In order to judge the performance of algorithm 8.1 for finding a subset of the set of all the eigenvalues of a symmetric tridiagonal matrix, we used both algorithm 8.1 and the bisection algorithm to find λ_1 for the ten test matrices discussed in this paper. (Algorithm 8.1a can only be used to find a proper subset of the eigenvalues without actually computing all of the eigenvalues if those eigenvalues not in the subset are already known.) Let i be the smallest integer for which $\lambda_1 - \lambda_i > 10^{-14} \times \mid \lambda_1 \mid$. For both algorithms, we began with a suspect interval with the left endpoint set to $(\lambda_1 + \lambda_i)/2$. The right endpoint was set by calculating an upper bound on λ_1 with Gershgorin's circle theorem. For algorithm 8.1 we did not use knowledge of the method that defined the original suspect interval with bounded isolation ratio (which would have accelerated convergence); we assume only that there could be eigenvalues anywhere in $(-\infty, (\lambda_1 + \lambda_i)/2)$.

For the same values for requested relative tolerances that we used in the previous set of experiments, neither of the two tested algorithms had any splitting steps with this choice for an initial suspect interval. Thus the number of iterations that the bisection algorithm used varied little from matrix to matrix – it depended only on the ratio of the size of the initial suspect interval to the requested precision, and this ratio did not vary much between matrices.

The results of this second set of experiments are shown in Tables 13.4–13.6. We again see acceleration when comparing our methods to bisection where the requested precsion is small. However, algorithm 8.1 performed even better in this set of experiments than in the previous set of experiments; we see a greater decrease in the cost of using algorithm 8.1 to find only a single eigenvalue (over finding all the eigenvalues) than we do with the bisection algorithm.

For desired relative tolerances of 10^{-11} and larger, algorithm 8.1 used significantly fewer iteration to caluclate λ_1 for matrix 7 than for the other matrices. The reason why so few iterations were needed is that the largest two eigenvalues of matrix 7 are closely clustered, so the suspect interval with which we began algorithm 8.1 (as well as the bisection algorithm) had its left endpoint very close to λ_1. This resulted in the double exponential sieve algorithm converging quickly to a fairly small interval.

Table 13.1: Cost of three symmetric tridiagonal eigenvalue solvers for the test matrices from [PS] with relative error of 10^{-15}. The cost is given in the number of computations of $n_-(x)$ or equivalents and in milliseconds. The timing results were obtained by averaging over 10,000 runs.

Matrix	Bisection		Alg 8.1		Alg 8.1a	
	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.
1	194	1.277	103	0.912	86.25	1.062
2	1347	59.507	884	35.248	682	38.077
3	1923	142.080	1412	92.526	1096.25	97.389
4	1246	75.144	972	54.444	769.75	52.542
5	691	30.997	569	24.542	474.25	26.276
6	240	1.935	135	1.395	110.5	1.594
7	201	1.934	233	3.491	175.75	2.918
8	626	19.541	512	16.372	391	15.282
9	1292	57.205	879	35.516	677.75	37.606
10	99	4.350	53	2.562	33.75	1.984
All	7859	393.970	5752	267.007	4497.25	274.730

Table 13.2: Cost of symmetric tridiagonal eigenvalue solvers with relative error of 10^{-11}.

Matrix	Bisection		Alg 8.1		Alg 8.1a	
	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.
1	138	0.924	91	0.845	72	0.945
2	927	40.860	796	32.553	620.25	34.785
3	1223	89.655	1240	81.778	1010.75	89.604
4	853	51.019	793	44.096	649.75	43.953
5	372	16.792	371	16.416	320.75	17.225
6	170	1.383	123	1.305	96.25	1.431
7	117	1.153	150	2.781	116.75	2.181
8	429	13.415	395	12.333	308	11.635
9	902	39.838	787	32.255	620.75	34.417
10	71	3.183	49	2.400	33.75	1.977
All	5202	258.221	4795	226.761	3849	238.154

Table 13.3: Cost of symmetric tridiagonal eigenvalue solvers with relative error 10^{-7}.

Matrix	Bisection		Alg 8.1		Alg 8.1a	
	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.
1	86	0.598	79	0.781	72	0.944
2	537	23.824	656	27.486	539.5	30.373
3	573	42.274	695	50.999	729.5	65.533
4	487	29.482	555	32.027	500.5	33.941
5	163	7.532	194	9.019	174	9.133
6	105	0.881	107	1.202	91.5	1.387
7	65	0.667	56	0.756	42.5	0.736
8	248	7.851	270	8.802	224	8.763
9	512	22.811	602	26.102	521	29.280
10	45	2.136	45	2.243	33.75	1.978
All	2821	138.056	3259	159.418	2928.25	182.068

Table 13.4: Cost of computing one eigenvalue with two symmetric tridiagonal eigenvalue solvers for the test matrices from [PS] with relative error of 10^{-15}. The cost is given in the number of computations of $n_-(x)$ or equivalents and in milliseconds. The timing results were obtained by averaging over 10000 runs.

Matrix	Bisection		Alg 8.1	
	Comp $n_-(x)$	ms.	Comp $n_-(x)$	ms.
1	48	0.347	27	0.275
2	47	2.182	32	1.454
3	48	3.675	30	2.092
4	45	2.837	29	1.677
5	48	2.225	29	1.261
6	48	0.423	27	0.305
7	46	0.479	32	0.571
8	45	1.482	28	0.890
9	45	2.085	29	1.263
10	49	2.087	30	1.422
All	469	17.823	293	11.210

Table 13.5: Cost of computing one eigenvalue with two symmetric tridiagonal eigenvalue solvers with relative error of 10^{-11}.

Matrix	Bisection		Alg 8.1	
	Comp $n_{-}(x)$	ms.	Comp $n_{-}(x)$	ms.
1	35	0.265	23	0.251
2	34	1.593	28	1.317
3	35	2.693	30	2.089
4	32	2.035	25	1.491
5	35	1.636	29	1.262
6	35	0.320	27	0.306
7	33	0.357	16	0.445
8	32	1.069	24	0.796
9	32	1.499	25	1.131
10	35	1.504	26	1.267
All	338	12.971	253	10.355

Table 13.6: Cost of computing one eigenvalue with two symmetric tridiagonal eigenvalue solvers with relative error of 10^{-7}.

Matrix	Bisection		Alg 8.1	
	Comp $n_{-}(x)$	ms.	Comp $n_{-}(x)$	ms.
1	22	0.183	23	0.251
2	20	0.961	24	1.180
3	22	1.721	26	1.865
4	18	1.175	21	1.310
5	22	1.050	25	1.127
6	22	0.217	23	0.280
7	20	0.234	7	0.250
8	19	0.656	20	0.699
9	19	0.917	21	0.995
10	22	0.971	26	1.265
All	206	8.084	216	9.221

References

[A] E. Anderson, et al., *LAPACK Users' Guide*, SIAM, Philadelphia, PA, 1992.

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

[BMW] W. Barth, R. S. Martin, J. H. Wilkinson, Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection, *Numer. Math.*, 9, 386–393, 1967.

[BP91] D. Bini, V. Y. Pan, Parallel Complexity of Tridiagonal Symmetric Eigenvalue Problem, *Proc. 2nd Ann. ACM-SIAM Symposium on Discrete Algorithms*, 384-393, ACM Press, New York, and SIAM Publications, Philadelphia, Pennsylvania, 1991.

[BP92] D. Bini, V. Y. Pan, Practical Improvement of the Divide-and-Conquer Eigenvalue Algorithms, *Computing*, 48, 109–123, 1992.

[BP94] D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[BP98] D. Bini, V. Y. Pan, Computing Matrix Eigenvalues and Polynomial Zeros Where the Output Is Real, *SIAM J. on Computing*, 27, 4, 1099-1115 (1998).

[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.

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

[P89] V. Y. 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.

[P92] V. Y. Pan, Accelerated Solution of the Symmetric Eigenvalue Problem, Tech. Report TR 92-06, *Computer Science Dept., SUNYA*, Albany, NY, 1992, and Tech. Report 93-016, *International Computer Science Institute*, Berkeley, CA, 1993.

[P94] V. Y. Pan, New Techniques for Approximating Complex Polynomial Zeros, *Proc. 5-th Ann. Symp. on Discrete Algorithms*, 260-270, ACM Press, New York, and SIAM Publications, Philadelphia, 1994.

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

[PD] V. Y. Pan, J. Demmel, A New Algorithm for the Symmetric Eigenvalue Problem, *J. of Complexity*, 9, 387-405, 1993.

[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.

[PS] V. Pereyra, G. Scherer, Eigenvalues of Symmetric Tridiagonal Matrices: a Fast, Accurate and Reliable Algorithm, *J. Inst. Maths. Applics.*, 12:209–222, 1973.

A Some Variations of Our Algorithms

Together with our algorithms 7.1, 8.1, 8.1a and 10.1, we studied some variations, in particular, in the expressions for μ_i, M_i, t_{i+1} and N_{i+1} of (7.4)-(7.7) of algorithm 7.1 and similarly for algorithm 10.1. Namely, we considered the following expressions, generalizing equations (7.4)-(7.7):

$$\mu_i = c_i - kp(c_i)/p^{'}(c_i),$$

$$c_i = b_i + (b_i - a_i)h_i,$$

$$M_i = \max\{\frac{n-j-k}{N_i+1+2h_i}\ ,\ \frac{j}{N_i-1-2h_i}\}\ ,$$

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

.

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

for a nonnegative parameter h_i. Equations (7.4)-(7.7) and all our all algorithms of the previous sections were obtained by using the above equations for $h_i = 0$.

We experimented with the above variation of our algorithms for both $h_i = 0$ and $h_i = 1$. The results of the experiments were not different in both cases. We also extended our theoretical analysis of the algorithms to $h_i > 0$ (cf. [P92]). In fact, further extension to $h_i < -1$ is also straightforward.

Theoretically, choosing small positive h_i has an advantage of enabling us to bound the ratio $p'(c_i)/p(c_i)$. This enables us to exclude overflow, which is plausible theoretically for $h_i = 0$ but never occurred in our extensive experiments. Even with setting $h_i = 0$, we could have excluded overflow simply by first computing the ratio $p(c_i)/p'(c_i)$. If it is close to zero, we would have ended the computations by detecting some eigenvalues in the interval $[b_i - t, b_i) = [c_i - t, c_i)$.