N ROOTS OF THE SECULAR EQUATION IN $O(N)$ OPERATIONS *

OREN E. LIVNE †‡ AND ACHI BRANDT †§

Abstract. We present a novel multilevel algorithm which computes N roots of the secular equation in $O(CN)$ computer operations, where C depends on the desired accuracy. Since current methods of solution require $O(N^2)$ operations, this algorithm can drastically reduce the computational effort in various applications, including updating the singular value decomposition and symmetric eigenvalue problems, and solving constrained least squares problems. The algorithm is based on the multilevel approach for fast evaluation of integral transforms. It has been adapted for the efficient solution of the secular equation. We have also incorporated discontinuous kernel softening, a technique which improves the implementation of multilevel summation algorithms toward theoretical optimality. We present and discuss numerical results, parallelization, and other related applications of the multilevel approach, including a possible substitute for current symmetric tridiagonal eigenbasis solvers (such as the Divide and Conquer method).

Key words. secular equation, fast multilevel summation, root-search

AMS subject classifications. 15A18, 65F15, 65H17, 65R10, 65R20, 65Y05, 65Y20, 68Q25

1. The secular equation.

We consider the computational task of finding all the roots $\{\lambda_k^*\}_{k=1}^N$ of the secular equation

$$f(\lambda) := 1 + \sigma v(\lambda) = 0, \qquad v(\lambda) := \sum_{k=1}^{N} \frac{u_k}{d_k - \lambda}, \tag{1.1}$$

which are strictly separated by the values $\{d_k\}_{k=1}^N$, namely [21, 25, 32]

$$d_1 < \lambda_1^* < d_2 < \lambda_2^* < \ldots < d_N < \lambda_N^* < d_N + \sigma \sum_{k=1}^{N} u_k^2, \tag{1.2}$$

assuming $d_1 < d_2 < \ldots < d_N$ are real, $\sigma > 0$ and $u_k > 0$ for all k. This problem has various applications in numerical linear algebra, such as

1. Updating the singular value decomposition of matrices [1, 10].
2. Modifying the symmetric eigenvalue problem [11, 14, 15, 16, 24, 25, 27].
3. Solving constrained least squares type problems [13, 18, 20, 21, 23, 28, 36, 37, 44].
4. Computing the eigenvalues of a matrix using the escalator method [19].
5. Invariant subspace computations [17].

A thorough literature survey may be found in [32, 33, 34, 35].

1.1. Current methods of solution.

Secular equations are often a "subproblem of a larger one" [34], as in the Divide and Conquer method [26, 16]. Consequently, they "typically have to be solved to high accuracy many times, which requires fast and stable methods" [34]. Many root-searching algorithms for solving (1.1) have been extensively studied and developed, among these are the following:

*This research was supported by Grant No. 696/97 from the Israel Science Foundation, by AFOSR Contract F33615-97-D5405, and by the Carl F. Gauss Minerva Center for Scientific Computation at the Weizmann Institute of Science.

†Department of Computer Science and Applied Mathematics, the Weizmann Institute of Science, Rehovot 76100, Israel.

‡Email address: livneo@wisdom.weizmann.ac.il

§Email address: achi@wisdom.weizmann.ac.il

1. The quadratic BNS methods [10, 11, 38], based on a rational interpolation.
2. Melman's methods [32, 33, 34, 35], that use a change of coordinates transforming the original equation into an equivalent problem for which Newton's method exhibits global quadratic convergence.
3. Gragg's third order zero-finder [24] and other high-order methods [34, 35].

These methods (e.g., Melman's) can compute any root of (1.1) to machine accuracy using a small number of direct evaluations of v and its derivative ($O(\log(\log(1/\varepsilon)))$ iterations are needed to obtain an ε-accuracy). Since each such evaluation costs $O(N)$ operations, N roots are computed in $O(N^2)$.

1.2. Objectives. Our goal is to design a linear complexity algorithm for computing N roots of (1.1) in only $O(CN)$ operations, where C depends on the desired accuracy ε, $C = O((\log(1/\varepsilon))^q)$ for some small $q \in \mathbb{R}_+$. This is achieved in a two-stage procedure:

(a) Designing an algorithm for *evaluating* v at N values of λ in $O(N)$ operations.
(b) Adapting this fast evaluation to the solution of (1.1) in $O(N)$, using any of the root-search methods mentioned in §1.1.

Both stages are handled efficiently and naturally by the multilevel approach presented in [5]. In §2 we present our fast multilevel evaluation algorithm (stage (a)), for uniformly dense $\{d_k\}_{k=1}^N$. §3 discusses the fast solution of (1.1) (stage (b)). We conclude in §4 by discussing non-uniform density, generalizations, parallelization, and other related applications of the multilevel approach, including a possible substitute for current symmetric tridiagonal eigenbasis solvers (such as the Divide and Conquer method [14, 16]).

2. Fast evaluation of $v(\lambda)$. A necessary stage toward the fast solution of (1.1) is the fast evaluation of v. Let $\{\lambda_j\}_{j=1}^N$ be any sequence satisfying (1.2) (e.g., approximations to $\{\lambda_j^*\}_{j=1}^N$ at a certain root-searching step); we wish to calculate

$$v(\lambda_j) = \sum_{k=1}^{N} G(d_k - \lambda_j)u(d_k), \quad j = 1, \dots N; \quad u(d_k) := u_k, \quad G(r) := \frac{1}{r} \tag{2.1}$$

in $O(N)$ operations. The algorithm for computing $\{v'(\lambda_j)\}_{j=1}^N$, if desired, is discussed in §3.2. For simplicity let us first assume that $\{d_k\}_{k=1}^N$ have a uniform density α, i.e., it is possible to place a uniform grid $\{D_K^1\}_{K=1}^{N_1}$ with meshsize H over $[d_1, d_N]$, so that in each interval $[D_K^1, D_{K+1}^1]$ there lies a uniformly bounded number (about $\alpha H =: m$) of d_ks. The interlacement property (1.2) implies that $\{\lambda_j\}_{j=1}^N$ are also uniformly dense (for nonuniform densities, see §4.1).

Our algorithm is a straightforward application of the general multilevel approach for fast evaluation of integral transforms with asymptotically smooth kernels, which is described in detail in [5, 7, 8, 9]. We also incorporate a technical modification (discontinuous softening) that improves the work-accuracy relation of multilevel summation algorithms toward optimality. This may be of interest in practical implementations.

2.1. Kernel softening. The kernel $G(r) = 1/r$ is *asymptotically smooth*, that is, increasingly smooth for larger r. As in [5, 7, 9] it can be decomposed into

$$G(r) = G_S(r) + G_{\text{local}}(r), \tag{2.2}$$

so that

(i) $G_S(r) = G(r)$ (or $G_{\text{local}}(r) = 0$) for all $|r| \geq S$.

(ii) G_S is *suitably smooth* on the scale S, namely, for any $\varepsilon > 0$ there exists $p = O(\log(1/\varepsilon)) \in \mathbb{N}$ such that G_S can be uniformly approximated to an accuracy ε by a p-order interpolation from its values on any uniform grid with a meshsize comparable with S [9].

Traditional multilevel algorithms [5, 8, 9, 40] used a polynomial softened kernel

$$G_S(r) = \frac{1}{S} G^*(\frac{r}{S}), \qquad G^*(r) := \begin{cases} \sum_{n=0}^{2p-1} a_n r^n, & |r| \le 1, \\ G(r), & |r| \ge 1, \end{cases} \tag{2.3}$$

that fits $G, G', \ldots, G^{(p)}$ at $r = \pm S$. In this paper we propose a novel *piecewise* smooth kernel softening in the form

$$G_S(r) = \frac{1}{S} \tilde{G}(\frac{r}{S}), \qquad \tilde{G}(r) := \begin{cases} 0, & |r| \le 1, \\ G(r), & |r| > 1, \end{cases} \tag{2.4}$$

which is suitably smooth only for $r \in \mathbb{R}\backslash\{-S, S\}$. Nevertheless, the discontinuous softening (2.4) has the following advantages over the continuous softening (2.3):

1. The derivative $\tilde{G}^{(p)}(r)$ vanishes for $|r| \le 1$; hence its magnitude is certainly less than $(G^*)^{(p)}(r)$ for all $r \ne \pm 1$. Moreover, $(G^*)^{(p)}$ may have a large magnitude (typically, $(G^*)^{(p)} \sim O((p!)^{1+\nu}) \sim O(p! p^{\nu p})$ for some $\nu > 0$). This is observed especially in kernels that fully depend on r, rather than on $|r|$ only (in [8, 9, 40], $G = G(|r|)$). For instance, in the secular problem, G's sign flip across $r = 0$ causes a "fold" in G^* (see Figure 1), consequently causing a large $\|(G^*)^{(p)}\|_{L^\infty(\mathbb{R})}$ (see Table 1).

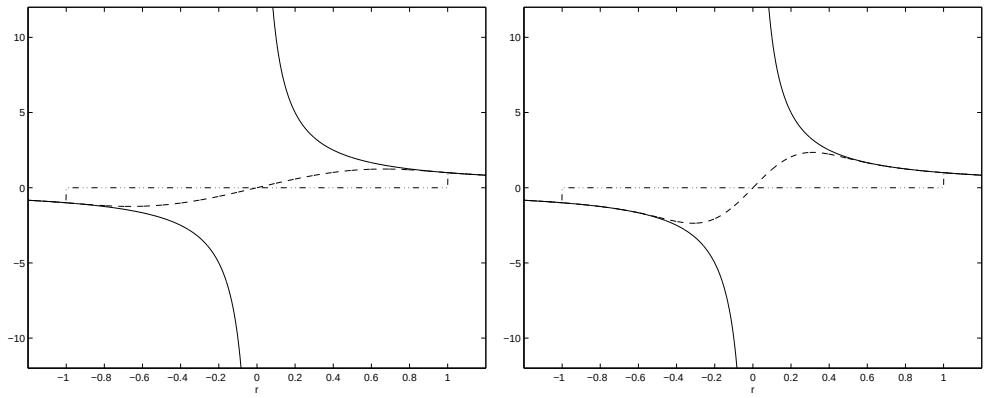

FIG. 1. *The kernel $G(r) = 1/r$ (solid line) and its softenings, $G^*(r)$ (dashed line) and $\tilde{G}(r)$ (dashed-dotted line), for $p = 2$ (left picture) and $p = 12$ (right picture).*

TABLE 1

The powers ν corresponding to the magnitudes of the p-order derivatives of $G^*, \tilde{G}$, which determine the interpolation error ε_I, versus p.

p	$\log(\|(G^*)^{(p)}\|_{L^\infty(\mathbb{R})}/p!)/(p\log(p))$	$\log(\|\tilde{G}^{(p)}\|_{L^\infty(\mathbb{R}\backslash\{-S,S\})}/p!)/(p\log(p))$
2	0.8582	0
4	0.4275	0
6	0.3357	0
8	0.2934	0
10	0.2681	0

The relative error ε_I in approximating the scale-S softened kernel $G_S(r) := \tilde{G}(r/S)/S$ by a p-order central interpolation from its values on a meshsize-H uniform grid (when the discontinuities are not straddled by the interpolation interval) satisfies

$$\varepsilon_I \lesssim 2\left(\frac{pH}{2eS}\right)^p, \tag{2.5}$$

as explained in Appendix A.

2. An evaluation of G^* costs $O(p)$ operations, versus none per $\tilde{G}$ evaluation.

3. Computing $\{a_n\}_{n=0}^{2p}$ of (2.3) requires $O(p^3)$ operations, whereas $\tilde{G}$ requires no "preparatory work". This is usually a pre-processing step, but if the softened kernel needs to be repeatedly updated, this would mean a major saving of work.

On the other hand, $\tilde{G}$'s jumps at $r = \pm 1$ require additional correction steps, which are described in §2.2. Overall, the cost efficiency of the multilevel summation algorithm is improved by using (2.4) instead of (2.3), because of the first two advantages. This is shown in §2.4 for the secular equation, and in §4.3, for general integral transforms in higher dimensions.

2.2. Derivation of the algorithm. Following the terminology of [5, §3–4], observe that

$$v(\lambda_j) = v_S^0(\lambda_j) + v_{\text{local}}^0(\lambda_j), \qquad j = 1, \ldots, N, \tag{2.6}$$

where

$$v_S^0(\lambda_j) := \sum_{k=1}^{N} G_S(d_k - \lambda_j)u(d_k), \qquad j = 1, \ldots, N \tag{2.7}$$

and

$$v_{\text{local}}^0(\lambda_j) := \sum_{k:|d_k-\lambda_j|\le S} G(d_k - \lambda_j)u(d_k), \qquad j = 1, \ldots, N. \tag{2.8}$$

The sum (2.8) extends over $O(s)$ points d_k, if we choose $S = sH$. The softened kernel can be represented as

$$G_S(d_k - \lambda_j) = \sum_{K\in\sigma_k} \omega_{kK}^{1,0} G_S(D_K^1 - \lambda_j) + O(\varepsilon_I), \tag{2.9}$$

where $\sigma_k := \{K : |D_K^1 - d_k| < pH/2\}$, $\omega_{kK}^{1,0}$ are the weights of interpolation from the gridpoints D_K^1 to d_k, and ε_I is bounded by (2.5). The grid $\{D_K^1\}_{K=1}^{N_1}$ may include $O(p)$ points to the left of d_1 and to the right of d_n to keep the interpolation central; from now on, p is assumed to be even. In fact, for a given j, (2.9) holds for all $k = 1, \ldots, N$ except the set

$$\Omega_j^{\text{bad}} := \{k : \exists K, K+1 \in \sigma_k, b \in \{-1, 1\}, \quad \operatorname{sgn}(D_K^1 - \lambda_j - bS) \neq \operatorname{sgn}(D_{K+1}^1 - \lambda_j - bS)\},$$

since $G_S(d_k - \cdot)$ is not continuous in the interpolation stencil for $k \in \Omega_j^{\text{bad}}$. Neglecting $O(\varepsilon_I)$ terms, it follows that

$$v_S^0(\lambda_j) = \sum_{k=1}^{N} \sum_{K\in\sigma_k} \omega_{kK}^{1,0} G_S(D_K^1 - \lambda_j)u(d_k) + \omega^0(\lambda_j) = V_S^0(\lambda_j) + \omega^0(\lambda_j), \tag{2.10}$$

where

$$V_S^0(\lambda_j) := \sum_{K=1}^{N_1} G_S(D_K^1 - \lambda_j) U^1(D_K^1), \qquad j = 1, \dots N, \tag{2.11}$$

$$U^1(D_K^1) := \sum_{k \in \tau_K} \omega_{kK}^{1,0} u(d_k), \quad \tau_K := \{k : K \in \sigma_k\}, \quad K = 1, \dots, N_1, \tag{2.12}$$

$$\omega^0(\lambda_j) := \sum_{k \in \Omega_j^{\text{bad}}} G_S(d_k - \lambda_j) u(d_k) - \sum_{K \in \Omega_j^{\text{BAD}}} G_S(D_K^1 - \lambda_j) \tilde{U}_j^1(D_K^1), \tag{2.13}$$

$$\tilde{U}_j^1(D_K^1) := \sum_{k \in \tilde{\tau}_{jK}} \omega_{kK}^{1,0} u(d_k), \quad \tilde{\tau}_{jK} := \Omega_j^{\text{bad}} \cap \tau_K, \quad K \in \Omega_j^{\text{BAD}}, \tag{2.14}$$

$\Omega_j^{\text{BAD}} := \{K : \tilde{\tau}_{jK} \neq \emptyset\}$, and (2.13), (2.14) are defined for all $j = 1, \dots, N$. Note that the sums in (2.12), (2.13), and (2.14) extend over $O(p)$ points; hence, they are *local*. $\{U_K^1\}_K$ is the "aggregation" of $\{u_k\}_k$ from the non-uniform fine locations $\{d_k\}_k$ (denoted "level $l = 0$") to the uniform coarse locations $\{D_K\}_K$ (denoted "level $l = 1$"), a procedure referred to as *anterpolation* in [5], since it is the adjoint of interpolation. Similarly, we can use the smoothness of $G_S(d - \lambda)$ in λ to write

$$G_S(D_K^1 - \lambda_j) = \sum_{J \in \bar{\sigma}_j} \bar{\omega}_{jJ}^{1,0} G_S(D_K^1 - \Lambda_J^1) + O(\varepsilon_I), \quad j = 1, \dots, N, \tag{2.15}$$

for all $K = 1, \dots, N_1$ except the set

$$\bar{\Omega}_j^{\text{BAD}} := \{K : \exists J, J+1 \in \bar{\sigma}_j, b \in \{-1, 1\}, \operatorname{sgn}(D_K^1 - \Lambda_J^1 - bS) \neq \operatorname{sgn}(D_K^1 - \Lambda_{J+1}^1 - bS)\},$$

where $\bar{\sigma}_j := \{J : |\Lambda_J^1 - \lambda_j| < pH/2\}$, $\bar{\omega}_{jJ}^{1,0}$ are the λ-interpolation weights, and $\{\Lambda_J^1\}_{J=1}^{\bar{N}_1}$ is a uniform grid with meshsize H over $[\lambda_1, \lambda_N]$ (again including $O(p)$ points to the left of λ_1 and to the right of λ_N), from which we can use p-order central interpolation to all points $\lambda_1, \dots, \lambda_N$. Up to an $O(\varepsilon_I)$ error,

$$V_S^0(\lambda_j) = \bar{V}_S^0(\lambda_j) + z^0(\lambda_j), \qquad j = 1, \dots, N, \tag{2.16}$$

where

$$\bar{V}_S^0(\lambda_j) := \sum_{J \in \bar{\sigma}_j} \bar{\omega}_{jJ}^{1,0} V_S^1(\Lambda_J^1), \tag{2.17}$$

$$V_S^1(\Lambda_J^1) := \sum_{K=1}^{\bar{N}_1} G_S(D_K^1 - \Lambda_J^1) U^1(D_K^1), \qquad J = 1, \dots, \bar{N}_1, \tag{2.18}$$

$$z^0(\lambda_j) := \sum_{K \in \bar{\Omega}_j^{\text{BAD}}} G_S(D_K^1 - \lambda_j) U^1(D_K^1) - \sum_{J \in \bar{\sigma}_j} \bar{\omega}_{jJ}^{1,0} \tilde{V}_j^1(\Lambda_J^1), \tag{2.19}$$

and

$$\tilde{V}_j^1(\Lambda_J^1) := \sum_{K\in\bar{\Omega}_j^{\mathrm{BAD}}} G_S(D_K^1 - \Lambda_J^1)U^1(D_K^1), \qquad J = 1,\ldots,\bar{N}_1. \tag{2.20}$$

The sums in (2.17), (2.19) are over local sets and defined for all $j = 1,\ldots,N$; (2.18) is a uniform coarser version of (2.1). We have reduced the original evaluation of v at the non-uniform fine level ($l = 0$) to the evaluation of V_S^1 at the uniform coarse level ($l = 1$). In order to keep the evaluation of (2.8) inexpensive, the coarsening ratio m cannot be too large (e.g., $m = 2$ [9]) and s should not increase with N. To sum up, the multi-summation (2.1) is replaced by the following:

(i) *Anterpolation:* calculate the "aggregated" $\{U_K^1\}_K$ from (2.12).
(ii) *Coarse grid summation:* carry out the task (2.18).
(iii) *Interpolation:* interpolate $\{V_S^1(\Lambda_J^1)\}_J$ to $\{\bar{V}_S^0(\lambda_j)\}_j$ using (2.17).
(iv) *Local corrections:* add the local correction $v_{\mathrm{local}}(\lambda_j)$ defined by (2.8) to $\bar{V}_S^0$.
(v) *w-correction:* compute w^0 from (2.13),(2.14) and add it to $\bar{V}_S^0$.
(vi) *z-correction:* compute z^0 from (2.19), (2.20) and add it to $\bar{V}_S^0$.

The number of nodes at level 1 is roughly $N/2$, which may still be too large to calculate directly. Instead, the task (2.18) can be further reduced to summation at level $l = 2$ on twice as coarse (meshsize $2H$) λ- and d-grids, using the same algorithm [(i)–(vi)]: decomposition of G_S into G_{2S} plus a local part, anterpolation of U^1 to level 2, level 2 summation, interpolation of V_{2S}^2 to level 1, and addition of the three local corrections. The above-described procedure can be repeated recursively until a grid is reached at which direct summation can be done in at most $O(N)$ operations.

2.3. Computational cost and evaluation error. The local correction (iv) costs $O(sN)$ operations, since each G-evaluation costs $O(1)$. However, it is less obvious to implement the w-correction in $O(pN)$ operations. It may seem that for any given j it takes $O(p)$ points to compute every $\tilde{U}_j^1(D_K), K \in \bar{\Omega}^{\mathrm{BAD}}$; hence, $O(p^2N)$ operations are required for evaluating the right-hand term in the right-hand-side of (2.13). Instead, we can use a "sliding window" approach (see for example [41, 42, 45]): $\{\tilde{U}_1^1(D_K^1)\}_K$ are calculated in $O(p^2)$ and then repeatedly updated in $O(p)$ operations to obtain $\{\tilde{U}_2^1(D_K^1)\}_K$, and so on. This is possible since the sets $(\tau_{jK}\cup\tau_{j+1,K})\backslash(\tau_{jK}\cap\tau_{j+1,K})$ contains only $O(1)$ points for every $j = 1,\ldots,N-1$. The same approach can be applied to the z-correction, interpolations, and anterpolations. Thus, the total computational complexity of steps (i),(iii)-(vi) is $W = O((p+s)N)$, which is smaller than the $O(psN)$ cost of the multilevel summation with the "traditional" softening [5, 8, 9, 40]. Generally, if the order of anterpolation/interpolation from/to level l to/from $l-1$ is denoted by p_l and the softening scale is denoted by $S_l := 2^{l-1}Hs_{l-1}$, the total work W per fine grid point in evaluating (2.1) (omitting some constants and neglecting the direct evaluation at the coarsest level) is given by

$$\frac{W}{N} = \sum_{l=0}^{t-1} 2^{-l}(p_l + As_l), \tag{2.21}$$

where $t = O(\log N)$ is the number of levels and $A > 0$ is a constant. The error ε_v in evaluating v satisfies (as implied by (2.5))

$$\varepsilon_v \lesssim 2\sum_{l=0}^{t-1}\left(\frac{p_l}{2es_l}\right)^{p_l}.$$

2.4. Parameter optimization. The values of s, p at each of the levels $l = 0, \ldots, t-1$ should be determined to minimize the computational work under the constraint of a controlled evaluation error, $\varepsilon_v = \varepsilon$.

2.4.1. Two-level parameter optimization. Let us first consider the case $t = 1$. Discarding the coarse level summation portion of the work and omitting constants, the constrained minimization problem for $p := p_0, s := s_0$ is

$$\begin{cases} W/N & \propto p(1 + A\kappa/(2e)) & \longrightarrow \min., \qquad \kappa := 2es/p, \\ \varepsilon_v & \propto \kappa^{-p} & = \varepsilon. \end{cases}$$

The optimum is attained if and only if

$$\left[\frac{d}{d\kappa}\left(\frac{1 + A\kappa/(2e)}{\log(\kappa)}\right)\right]_{k=k_{\rm opt}} = 0, \quad p_{\rm opt} = \frac{\log(1/\varepsilon)}{\log(\kappa_{\rm opt})}.$$

This implies

$$\kappa_{\rm opt}(\log(\kappa_{\rm opt}) - 1) = (2e)/A \qquad \Longrightarrow \qquad \text{(e.g.)} \quad \begin{array}{ll} \kappa_{\rm opt} \approx 6.376, & A = 1, \\ \kappa_{\rm opt} \approx 9.045, & A = 0.5. \end{array}$$

Thus,

$$p_{\rm opt} = p_{\rm opt}(\varepsilon) = K_1 \log\left(\frac{1}{\varepsilon}\right), \quad s_{\rm opt} = s_{\rm opt}(\varepsilon) = K_2 \log\left(\frac{1}{\varepsilon}\right), \tag{2.22}$$

where for instance $K_1 \approx 0.54, K_2 \approx 0.63$ for $A = 1$ and $K_1 \approx 0.45, K_2 \approx 0.75$ for $A = 0.5$. Consequently, the computational complexity of evaluating (2.1) to accuracy ε is

$$W = (K_1 + AK_2)N \log(1/\varepsilon) =: KN \log(1/\varepsilon). \tag{2.23}$$

2.4.2. Multilevel parameter optimization. Clearly, if we use $p_l = p_{\rm opt}(\varepsilon)$, $s_l := s_{\rm opt}(\varepsilon)$ at all levels $l = 0, \ldots, t-1$, the error ε_v would be $t\varepsilon$. Instead, we use

$$p_l = p_{\rm opt}(2^{-l-1}\varepsilon), \quad s_l = s_{\rm opt}(2^{-l-1}\varepsilon), \qquad l = 0, \ldots, t-1, \tag{2.24}$$

so that

$$\varepsilon_v = \sum_{l=0}^{t-1} 2^{-l-1}\varepsilon \le \varepsilon$$

and

$$W \le KN \sum_{l=0}^{t-1} 2^{-l} \log\left(\frac{2^{l+1}}{\varepsilon}\right) \le 2KN \left(\log\left(\frac{1}{\varepsilon}\right) + 4\log(2)\right), \tag{2.25}$$

using (2.21). This cost is smaller than the total cost of the multilevel summation algorithm with continuous softening. Indeed, with the latter we get $W = O(N(\log(1/\varepsilon))^2)$.

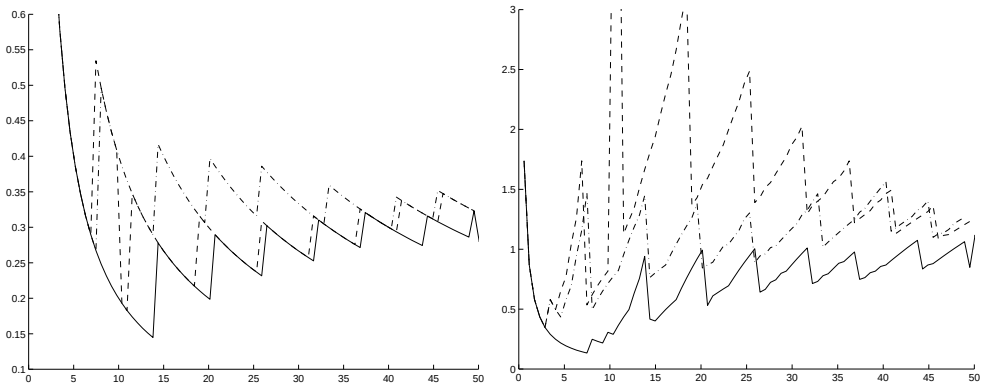

FIG. 2. *The optimal interpolation order* ($p_{\rm opt}(\varepsilon)/\log(1/\varepsilon)$, *left picture) and the optimal softening distance* ($s_{\rm opt}(\varepsilon)/\log(1/\varepsilon)$, *right picture) versus* $\log(1/\varepsilon)$, *for* $N = 64$ *(dashed line),* $N = 256$ *(dashed-dotted line), and* $N = 1024$ *(solid line). For small* ε, $p_{\rm opt} \approx 0.3\log(1/\varepsilon)$ *and* $s_{\rm opt} \approx 1.1\log(1/\varepsilon)$.

2.5. Numerical results. First, we performed two-level ($t = 1$) evaluation experiments of (2.1) for different values of N, to show that the optimal p, s indeed satisfy (2.22). The pair $(p_{\rm opt}(\varepsilon), s_{\rm opt}(\varepsilon))$ corresponding to the minimal W (out of all $0 \le p \le 16, 0 \le s \le 64$) was computed for various ε values, and stored in a table. The values of W were averaged over 20 experiments, each using a uniformly random sequence pair $\{d_k\}_{k=1}^N, \{\lambda_j\}_{j=1}^N \subset [0,1]$ that satisfied (1.2). Figure 2 shows that $p_{\rm opt}(\varepsilon)/\log(1/\varepsilon), s_{\rm opt}(\varepsilon)/\log(1/\varepsilon)$ are indeed bounded independently of N.
Second, we performed the multilevel evaluation of (2.1) for various N and ε values ($t = O(\log N)$ being the maximum possible, so that level $l = t$ grids contained $O(p_t)$ points) using $\{p_l, s_l\}_{l=0}^{t-1}$, which were computed using the table generated at the two-level stage and (2.24). Table 2 summarizes the computational cost of evaluating v in these experiments; each experiment was averaged over 20 uniformly random sequence pairs $\{d_k\}_{k=1}^N, \{\lambda_j\}_{j=1}^N \subset [0,1]$ satisfying (1.2) (this was a sufficiently large sample). The l_∞ error $\tilde{\varepsilon}$ of the differences between the directly computed v values and the values computed using the fast evaluation algorithm was always less than the desired ε. It can be observed that W behaves according to the desired (2.25).

TABLE 2

The computational cost $W/(N\log(1/\tilde{\varepsilon}))$ *versus* N *and* ε. *Each column (starting from the second) corresponds to a different* $\log_{10}(\varepsilon)$ *value, that is,* $\varepsilon = 10^{-2}$ *through* 10^{-12}; W *is the arithmetic operations count. It can be observed that* $W/(N\log(1/\tilde{\varepsilon}))$ *is practically uniformly bounded, as claimed by (2.25).*

N	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
64	36.0	33.1	31.7	30.0	26.9	21.6	8.1	5.5	5.5	5.5	5.5
128	41.0	40.4	37.0	40.3	45.3	51.0	37.4	42.3	38.2	33.6	30.1
256	46.8	46.0	40.4	43.1	59.6	61.3	57.0	60.3	64.6	62.4	60.4
512	46.2	44.0	40.6	40.4	51.6	62.8	61.4	65.3	72.7	71.4	71.5
1024	42.5	41.1	40.4	38.7	56.5	49.4	62.9	64.0	68.2	75.0	74.2
2048	44.1	43.2	42.3	39.8	58.3	48.1	64.9	64.6	69.2	77.7	77.1
4096	43.0	41.8	40.8	36.8	51.1	45.2	63.2	64.0	67.6	77.3	76.9

3. Fast solution of $f(\lambda) = 0$. The fast evaluation algorithm presented in §2 can be naturally adapted to any root-search method for solving $f(\lambda) = 0$. For demon-

stration purposes, we used Melman's improved Newton method [32]. Let $1 \le j < N$ (for simplicity we avoid the case $j = N$, which is also treated in [32]). The iterations take the form

$$\lambda_j^{(n+1)} = d_j + \frac{(\lambda_j^{(n)} - d_j)^2 f'(\lambda_j^{(n)})}{f(\lambda_j^{(n)}) + (\lambda_j^{(n)} - d_j) f'(\lambda_j^{(n)})}, \qquad n = 0, 1, 2, \ldots. \tag{3.1}$$

It was proved that these iterations converge quadratically to λ_j^*, provided that the starting point is

$$\lambda_j^{(0)} = d_j + \frac{2A}{-B - \sqrt{B^2 - 4AC}}, \tag{3.2}$$

where

$$\begin{array}{lll} a := 1 + \Delta_j, & \delta := d_{j+1} - d_j, & \\ A := -\frac{u_j}{\delta}, & B := a\delta + u_j, & C := \frac{u_{j+1}}{\delta} - a, \end{array}$$

and

$$\Delta_j := \sum_{k:k\neq j} \frac{u_k}{d_k - d_j}, \qquad j = 1, \ldots, N. \tag{3.3}$$

Our algorithm for finding the roots $\{\lambda_j^*\}_{j=1}^{n-1}$ to an accuracy ε consists of the following steps:

(i) Compute $\{\Delta_j\}_{j=1}^{N-1}$ of (3.3) using the fast evaluation algorithm (§2).
(ii) For $j = 1$ to $N - 1$, set λ_j to the expression of (3.2).
(iii) Compute $\{V_S^1(\Lambda_J^1)\}_{J=1}^{N_1}$ using the fast evaluation algorithm (§2).
(iv) For $j = 1$ to $N - 1$, do steps (v)–(viii).
(v) While (STOP–CRITERION$_j$ = FALSE) do steps (vi)–(viii).
(vi) Compute $f(\lambda_j)$ (see §3.1).
(vii) Compute $f'(\lambda_j)$ (see §3.2).
(viii) Set $\lambda_j \leftarrow d_j + ((\lambda_j - d_j)^2 f'(\lambda_j))/(f(\lambda_j) + (\lambda_j - d_j) f'(\lambda_j))$.

Step (i) is executed in $O(N \log(1/\varepsilon))$ using the fast evaluation algorithm of §2 for computing $\{v(\lambda_j)\}_{j=1}^N$ to accuracy ε, with one modification: the kernel $G(r)$ is defined to be 0 at $r = 0$. Here we can accept a low accuracy, since we only provide initial conditions for the roots.

The initialization of $\{\lambda_j\}_j$ (step (ii)) requires $O(N)$ operations.

Step (iii), using an accuracy ε, is a preparatory step for the fast evaluation of f, f' in steps (vi),(vii) (see §3.1,3.2). We execute the algorithm for evaluating $\{v(\lambda_j)\}_{j=1}^N$ to accuracy ε, except the last four steps (i.e., the steps before interpolating $\{V_S^1(\Lambda_J^1)\}_J$ to level 0). This takes $O(N \log(1/\varepsilon))$ operations.

The stopping criterion may be chosen in different ways. We use the criterion

$$|\lambda_j^{(n+1)} - \lambda_j^{(n)}| \le \varepsilon |d_{j+1} - d_j|.$$

Provided that each evaluation of f or f' at steps (vi),(vii) costs $O(\log(1/\varepsilon))$ operations (see §3.1,3.2), the total cost of the algorithm (i)–(viii) is $O(N \log(1/\varepsilon))$. The numerical stability of algorithm (i)–(viii) depends solely on the stability of the root-search methods; the fast evaluation introduces in addition central interpolation, which is a numerically stable process.

3.1. $O(\log(1/\varepsilon))$ **evaluation of** v**.** Once $\{V_S^1(\Lambda_J^1)\}_J$ is computed and stored (step (ii)), a value $f(\lambda)$ for a given λ may be calculated in additional $O(\log(1/\varepsilon))$ operations, using the last four steps of the evaluation algorithm: interpolation of V_S^1 to the point λ, followed by the three local corrections to $v_s(\lambda)$. Here a "sliding window" (see §2.3) is used (for every j) to update $\tilde{U}_j^1(D_K^1), \tilde{V}_j^1(\Lambda_J^1)$ from their values at the previous root-search step. Since the approximations $\lambda_j^{(n)}$ to the jth root λ_j^* remain in the interval $[d_j, d_{j+1}]$ for all n (see melman1) the interpolation stencils in the w- and z-corrections can "move" only by at most $O(1)$ meshsizes in every Newton step. Hence, each correction costs only $O(p = \log(1/\varepsilon))$ operations per Newton step for a single root.

3.2. $O(\log(1/\varepsilon))$ **evaluation of** v'**.** If we also want to evaluate f', we can again use the pre-computed values of $\{V_S^1(\Lambda_J^1)\}_J$. As in §3.1, we perform the last four steps of the evaluation algorithm, with two modifications:

1. In the interpolation step, we use different interpolation coefficients $\{\xi_J^{1,0}(\lambda)\}_J$ for interpolating V_S^1 from $\{\Lambda_J^1\}_{J\in\bar{\sigma}_j}$ to λ, instead of $\{\bar{\omega}_J^{1,0}(\lambda)\}_J$ (used for interpolating V_S^1 from $\{\Lambda_J^1\}_{J\in\bar{\sigma}_j}$ to λ in the v-evaluation step; $\{\bar{\omega}_J^{1,0}(\lambda_j)\}_J = \{\bar{\omega}_{j,J}^{1,0}\}_J$). These coefficients are computed from differentiating the interpolation polynomial for $G_S(D_K^1 - \cdot)$ (see also [40]), so that (except when discontinuities are straddled by the interpolation stencil)

$$-G_S'(D_K^1 - \lambda) = \sum_{J\in\bar{\sigma}_j} \xi_J^{1,0}(\lambda)\left[-G_S(D_K^1 - \Lambda_J^1)\right] + O(\varepsilon_I). \tag{3.4}$$

2. The three local corrections are executed with the kernel $-G'$ instead of G (note that $(d/d\lambda)[G_S(d-\cdot)] = -G'(d-\cdot)$).

We remark that we can evaluate v' to a lower accuracy than the one required for v, without spoiling the convergence of the Newton iterations (3.1). In fact, we can avoid computing the derivative by switching to the secant root-search method, thereby reducing the overall computing time by a factor 1.8.

3.3. Numerical results. Table 3 compares the computational cost of evaluating the roots $\{\lambda_j^*\}_{j=1}^N$ of (1.1), using a direct evaluation of v (with $\varepsilon = 10^{-10}$) versus a fast evaluation of v with $\varepsilon = 10^{-20}, 10^{-10}$. The results were averaged over 20 uniformly random sequence pairs $\{d_k\}_{k=1}^N, \{\lambda_j\}_{j=1}^N \subset [0,1]$ satisfying (1.2) (this was a sufficiently large sample). Indeed, the average cost per root for the direct evaluation method increases linearly with N, whereas it remains constant for our proposed method, as desired. The cross-over (using direct evaluation versus the fast evaluation, the roots being computed to the same accuracy ε) was detected at $N \approx 200$ for $\varepsilon = 10^{-10}$ and at $N \approx 450$ for $\varepsilon = 10^{-20}$ (for $\varepsilon = 10^{-5}$ at $N \approx 70$).

4. Concluding remarks. In the previous sections we described the basic elements of the fast evaluation of v and the fast solution of (1.1) for uniformly dense $\{d_k\}_k$ (N roots are computed in only $O(N)$ operations). The following are some important insights and generalizations of these algorithms, that can be further explored in a future research.

4.1. Non-uniform d**-density.** Recursive local grid refinement is essential to maintain the above work-accuracy relationship wherever the number of d_k points per meshsize is large, including pathologically high concentrations (for instance, $d_k = 1/k, k = 1, \ldots, N$).

TABLE 3

The computational cost (number of arithmetic operations) of the proposed novel algorithm versus current algorithms: the fourth column is the number of Newton steps (3.1) in the algorithm (i)–(viii) for $\varepsilon = 10^{-10}$, and its cost per root (number of arithmetic operations for computing a single root) is given in column 5. *Columns* 2 *and* 3 *are the corresponding measurements when f, f' in the algorithm (i)–(viii) are directly computed from (1.1), (2.1) to accuracy $\varepsilon = 10^{-10}$. Columns* 6,7 *are the corresponding values to columns* $4, 5$, *for $\varepsilon = 10^{-20}$.*

N	Direct (-10) # iter.	Direct (-10) cost/N	Fast (-10) # iter.	Fast (-10) cost/N	Fast (-20) # iter.	Fast (-20) cost/N
64	5.43	$2.66 \cdot 10^3$	5.50	$4.01 \cdot 10^3$	6.52	$4.70 \cdot 10^3$
128	5.59	$5.44 \cdot 10^3$	5.53	$8.29 \cdot 10^3$	6.62	$1.16 \cdot 10^4$
256	5.58	$1.08 \cdot 10^4$	5.55	$8.37 \cdot 10^3$	6.66	$2.09 \cdot 10^4$
512	5.56	$2.15 \cdot 10^4$	5.66	$8.09 \cdot 10^3$	6.71	$2.36 \cdot 10^4$
1024	5.57	$4.30 \cdot 10^4$	5.69	$7.34 \cdot 10^3$	6.73	$2.50 \cdot 10^4$
2048	5.56	$8.59 \cdot 10^4$	5.68	$7.49 \cdot 10^3$	6.76	$2.61 \cdot 10^4$
4096	5.56	$1.72 \cdot 10^5$	5.68	$7.45 \cdot 10^3$	6.75	$2.65 \cdot 10^4$

Importantly, the algorithm will be based on patches of *uniform* grids; therefore interpolations are highly efficient compared with those involving non-uniform meshsizes. In the rest of this section we first explain where these patches should be introduced, and then we discuss the adaptation of the evaluation algorithm to such patches.

4.1.1. Refinement strategy. Since in the secular problem the local average density of $\{\lambda_j\}_j$ is the same as the $\{d_k\}_k$'s density, local refinements are introduced in the same regions for both d and λ spaces; in general, we may need to construct different patches for the λs (see §4.3).

A direct application of the evaluation algorithm described in §2.2 does not efficiently address the v-evaluation task on non-uniform sets $\{d_k\}_k, \{\lambda_j\}_j$. In regions where the number α of d-points per meshsize H of the finest grid employed in the evaluation algorithm is large, the work involved in the local corrections increases like $O(\alpha^2)$. To avoid this, we introduce a patch of a twice finer grid, defined only over these regions. It is possible to construct an optimal quantitative criterion to decide where to introduce such a local refinement, based on its cost effectiveness. For example: a twice finer patch should be introduced in any region with length of at least s cells (where $S = sH$ is the softening distance, see §2.1), that includes more than $\alpha_c S$ d-points, where α_c is a small integer whose optimal value can be determined experimentally. Clearly, if two close regions need to be locally refined, it is more efficient to unify them into just one patch.

If yet more dense regions exist within the twice finer patches, we create yet finer patches within the former patches, using the same criterion. This is recursively repeated until no further refinement is needed.

4.1.2. The evaluation algorithm with patches. Here we start the algorithm on the finest patches, where we anterpolate u from the original $\{d_k\}_k$ which lie within the region of these patches and have the highest local density, to the equally spaced gridpoints of the finest patches. Thus we have eliminated the regions of the highest density from the original evaluation task. By recursively anterpolating u to yet coarser and larger patches, we finally arrive at the original everywhere uniform grid covering the full domain of the original $\{d_k\}_k$, where the algorithm of §2.2 can be directly applied.

Note that the local refinement create intermediate levels with kernels $G(d, \lambda)$ which no longer depend only on $d - \lambda$; but this changes only their local part, G_{local}, and the

local corrections still cost $O(s)$ per λ-point.

4.2. General kernels. The multilevel approach for evaluating (2.1) provides the same efficiency for computing integral transforms involving any asymptotically smooth kernel $G(r)$, as shown in [5]. In particular, other secular equations such as

$$1+\sigma\sum_{k=1}^{N}\frac{u_k}{(d_k-\lambda_j)^{\zeta}}=0$$

can be solved in linear time (N roots in $O(N)$ operations), for any $\{u_k\}_k \subset \mathbb{R}$ and $\zeta > 0$. Importantly, the same multilevel approach can be used to address other multi-summation tasks with other types of kernels (e.g., oscillatory) [5].

4.3. Higher dimensions. The secular equation (1.1) does not admit a higher dimensional analogue; nevertheless, it may still be interesting to extend the discontinuous softening technique to the multilevel summation of the transform

$$v(\underline{\lambda_j})=\sum_{k=1}^{N}G(\underline{d_k},\underline{\lambda_j})u(\underline{d_k}),\quad \{\underline{d_k}\}_{k=1}^{N},\{\underline{\lambda_j}\}_{j=1}^{\bar{N}}\subset\mathbb{R}^d, \tag{4.1}$$

with an asymptotically smooth $G(\underline{x},\underline{y})$. The discontinuous softening (2.4) can be extended to this case, specifically,

$$\tilde{G}(\underline{x},\underline{y})=\begin{cases}0, & |\underline{x}-\underline{y}|<1,\\ G(\underline{x},\underline{y}), & |\underline{x}-\underline{y}|>1,\end{cases}\qquad |\underline{x}-\underline{y}|:=\max_{\mu}|x_\mu-y_\mu|. \tag{4.2}$$

This kernel is singular on the sphere $|\underline{x}-\underline{y}|=S$; hence the w- and z-corrections involve points in a high-dimensional ring including $|\underline{x}-\underline{y}|=S$ (for kernels $G=G(|\underline{x}-\underline{y}|_2)$, it might be more efficient to use $|\underline{x}-\underline{y}|_2:=\sqrt{\sum_{\mu=1}^{d}(x_\mu-y_\mu)^2}$ in $\tilde{G}$ instead of $|x-y|$). It can be shown that they can be implemented in $O(p^d)$ operations per λ_j, again using the "sliding window" technique (see §2.3). In addition, the local correction costs here only $O(s^d)$ per λ_j, versus $O(s^d p)$ for the usual softening, e.g., (2.3). Substituting the optimal $p=s=O(\log(1/\varepsilon))$, the total work amounts to

$$W=O\left(N\left(\log\left(\frac{1}{\varepsilon}\right)\right)^d\right),$$

whereas it is $O(N(\log(1/\varepsilon))^q)$ when using everywhere-smooth softened kernels, with $q=d+1+\eta$, $\eta\geq 0$ depending on the magnitude of the p-order derivatives of the softened kernel for $r\leq S$ (see §2.1).

Importantly, discontinuous softening cannot replace the original smooth softening of [5] because

1. The original smooth softening is much simpler to implement, and the cost per node (for a fixed accuracy) may be smaller. It is also easier to adapt local refinements to it (see 4.1) than to discontinuous softening.

2. In many physical problems (e.g., evaluation of potential or dipole fields [8, 9, 40]), η is small and $d=2,3$, hence the relative loss $(q-d)$ is not too large.

3. An important advantage of continuous softening is that it gives the kind of multiscale description of the interactions needed in dynamic situations, where the particles carrying these interactions participate in multiscale movements, as in molecular dynamics [3, 5, 6, 39].

However, the cost-efficiency of the multilevel summation algorithms may be significantly improved by discontinuous softening when

1. The dimension d is low (in the extreme case, when $d = 1$);
2. The kernel G is inherently "hard-to-soften", as in the secular problem or in some problems where G is not a function of $r := |\underline{x} - \underline{y}|$ (i.e., unlike the cases of [8, 9, 40], where $G(r) = \log(r)$ or $1/r$).

The power d of $\log(1/\varepsilon)$ in the total work seems to be generally the smallest possible, since the local corrections involve $O(p^d)$ points per evaluation; in this sense, discontinuous softening leads to the optimal work-accuracy relation.

4.4. Parallelization. Our presented algorithms can be efficiently parallelized. The number of unparallelizable steps is theoretically only $O(\log(N))$, since we mostly rely on interpolations and local corrections, which can be fully parallelized (see also [5]).

4.5. Eigenbasis computation. The Divide and Conquer method [14, 16, 26] for finding the full spectrum (eigenvalues and eigenvectors) of an $N \times N$ symmetric tridiagonal matrix requires $O(N^2)$ computer operations (although it runs in $O(N \log N)$ for some very special matrices, the storage always increases as $O(N^2)$). Even if we incorporate our fast evaluation algorithm for the secular equation, the computational cost remains the same. Instead, the recently developed approach of *multiscale eigenbasis* (MEB) [6, 29, 30, 31] seems more reasonable and efficient in addressing such eigenbasis computations. For instance, this method can be directly applied for computing N eigenvectors and eigenvalues of the symmetric tridiagonal eigenproblem, as well as many other sparse eigenproblems, in $O(N \log N)$ operations and storage. This cost can be reduced to $O(N)$ in special cases, e.g., for discretizations of constant coefficient differential operators. Of course, singular cases (e.g., when the ratio of two adjacent diagonal elements is very large) need to be treated (e.g., deflated), as demonstrated in [10, 11, 14, 26] for $O(N^2)$ eigenbasis solvers.

5. Acknowledgement. The authors are thankful to the referees of this paper. In particular, they brought to our attention that the idea of evaluating the secular equation in $O(N)$ operations is not new. In [26, §5], Gu and Eisenstat describe how the fast multipole method [12] can be used to compute all the roots of the secular equation in $O(N)$ operations. The resulting Divide and Conquer algorithm for the symmetric tridiagonal eigenproblem computes all the eigenvalues in $O(N \log N)$ operations, and all the eigenvectors in $O(N^2 \log N)$ operations.
We can add that the idea of fast multilevel evaluation of integral transforms with very general kernels, on which the method presented here is based, appears already in [4, §8.6], although the specific case of the secular equation is not mentioned there. Also, the recent multiscale eigenbasis approach (see §4.5) shows how to reduce the $O(N^2 \log N)$ cost of calculating the entire eigenbasis, to only $O(N \log N)$ operations.

Appendix A. The error of a *pth* order central interpolation of $\tilde{G}$ defined by (2.4) from its values on a grid of meshsize H satisfies [43]

$$\varepsilon_I \leq \frac{1}{p!}\|\tilde{G}^{(p)}\|_{L^\infty(\mathbb{R})} \cdot \left(\frac{H}{2} \cdot \frac{3H}{2} \cdot \ldots \cdot \frac{(p-1)H}{2}\right)^2 = \frac{1}{p!} p! H^p \left(\frac{p!}{2^p (\frac{p}{2})!}\right)^2,$$

for any positive even p. Simplifying the expression and using Stirling's formula, we

get

$$\varepsilon_I \lesssim H^p \left(\frac{\sqrt{2\pi p}(\frac{p}{e})^p}{2^p \sqrt{\pi p}(\frac{p}{2e})^{\frac{p}{2}}} \right)^2 = 2 \left(\frac{pH}{2e} \right)^p .$$

When we scale $G_S(r) = \tilde{G}(r/S)/S$, the *pth* derivative is scaled by S^{-p-1}, and the relative error by S^{-p}. This implies (2.5).

REFERENCES

[1] Arbenz, P. and Golub, G. H., *On the spectral decomposition of Hermitian matrices modified by low rank perturbations with applications*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 40–58.

[2] Bai, D. and Brandt, A., *Local mesh refinement multilevel techniques*, SIAM J. Stat. Comput., 8 (1986), pp. 109–134.

[3] Bai, D. and Brandt, A., *Multiscale computation of polymer models*, in Brandt, A., Bernholc, J., and Binder, K. (Eds.), Multiscale Computational Methods in Chemistry and Physics, NATO Science Series III: Computer and Systems Sciences, NATO Science Series, Vol. 177, IOS Press, The Netherlands, 2001.

[4] Brandt, A., *Guide to multigrid development*, in Hackbusch, W. and Trottenberg, U. (Eds.), Multigrid Methods, Proceedings of Conference Held at Köln-Porz, November 23–27, 1981, Lecture Notes in Mathematics, Vol. 960, Springer-Verlag, Heidelberg, 1982.
Comp. Phys. Comm., 65 (1991), pp. 471–476.

[5] Brandt, A., *Multilevel computations of integral transforms and particle interactions with oscillatory kernels*, Comp. Phys. Comm., 65 (1991), pp. 471–476.

[6] Brandt, A., *Multiscale scientific computation: review 2001*, in Barth, T.J., Chan, T.F. and Haimes, R. (Eds.), Multiscale and Multiresolution Methods: Theory and Applications, Lecture Notes in Computational Science and Engineering, Vol. 20, Springer-Verlag, Heidelberg, 2001.

[7] Brandt, A. and Lubrecht, A. A., *Multilevel matrix multiplication and the fast solution of integral equations*, J. Comput. Phys., 90 (1990), pp. 348–370.

[8] Brandt, A. and Venner, C. H., *Multilevel evaluation of integral transforms on adaptive grids*, Report WI/GC–5, Weizmann Institute of Science, Rehovot, 1996.

[9] ———, *Multilevel evaluation of integral transforms with asymptotically smooth kernels*, SIAM J. Sci. Comput., 19 (1998), pp. 468–492.

[10] Bunch, J. R. and Nielsen, C. P.. *Updating the singular value decomposition*, Numer. Math., 31 (1978), pp. 111–129.

[11] Bunch, J. R., Nielsen, C. P., and Sorensen, D. C., *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[12] Carrier, J., Greengard, L., and Rokhlin, V., *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 669–686.

[13] Chan, T. F., Olkin, J. A., and Cooley, D. W., *Solving quadratically constrained least squares using black box solvers*, BIT 32 (1992), pp. 481–495.

[14] Cuppen, J. J. M., *A divide and conquer method for the tridiagonal eigenproblem*, Numer. Math., 36 (1981) pp. 177–195.

[15] Dongarra, J. J. and Sorensen, D. C., *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. s139–s154.

[16] Doron, G. and Tadmor, E., *An $O(N^2)$ method for computing the eigensystem of $N \times N$ symmetric tridiagonal matrices by the divide and conquer approach*, SIAM J. Sci. Stat. Comput., 1 (1990), pp. 161–173.

[17] Faddeeva, V. N., *Computational Methods of Linear Algebra*, Dover Publications, New York, 1959.

[18] Forsythe, G. E. and Golub, G. H., *On the stationary values of a second-degree polynomial on the unit sphere*, SIAM J. Appl. Math., 13 (1965), pp. 1050–1068.

[19] Fuhrmann, D. R., *An algorithm for subspace computation with applications in signal processing*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 213–220.

[20] Gander, W., *Least squares with a quadratic constraint*, Numer. Anal., 36 (1981), pp. 291–307.

[21] Gander, W., Golub, G. H., and von Matt, U., *A constrained eigenvalue problem*, Linear Algebra Appl., 114/115 (1989), pp. 815–839.

[22] Golub, G. H., *Some modified matrix eigenvalue problems*, SIAM Review, 15 (1973) pp. 318–334.
[23] Golub, G. H. and von Matt, U., *Quadratically constrained least squares and quadratic problems*, Numer. Math., 59 (1991), pp. 561–580.
[24] Gragg, W. B. and Reichel, L., *A divide and conquer method for unitary and orthogonal eigenproblems*, Numer. Math., 57 (1990), pp. 695–718.
[25] Gu, M. and Eisenstat, S. C., *A stable and efficient algorithm for the rank-one modifications of the symmetric eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276.
[26] Gu, M. and Eisenstat, S. C., *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.
[27] Handy, S. L. and Barlow, J. L. , *Numerical solution of the eigenproblem for banded, symmetric Toeplitz matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 205–214.
[28] Hanson, R. and Phillips, J., *An adaptive numerical method for solving linear Fredholm integral equations of the first kind*, Numer. Math., 24, pp. 291–307.
[29] Livne, O. E., *Multiscale Eigenbasis Algorithms*, Ph. D. Thesis, Weizmann Institute of Science, Rehovot, 2000.
[30] Livne, O. E. and Brandt, A., *$O(N \log N)$ multilevel calculation of N eigenfunctions*, in Brandt, A., Bernholc, J., and Binder, K. (Eds.), Multiscale Computational Methods in Chemistry and Physics, NATO Science Series III: Computer and Systems Sciences, NATO Science Series, Vol. 177, IOS Press, The Netherlands, 2001.
[31] Livne, O. E. and Brandt, A., *Multiscale Eigenbasis Calculations: N eigenfunctions in $O(N \log N)$*, in Barth, T.J., Chan, T.F. and Haimes, R. (Eds.), Multiscale and Multiresolution Methods: Theory and Applications, Lecture Notes in Computational Science and Engineering, Vol. 20, Springer-Verlag, Heidelberg, 2001.
[32] Melman, A., *Numerical solution of a secular equation*, Numer. Math., 69 (1995), pp. 483–493.
[33] ———, *A unifying convergence analysis of second-order methods for secular equations*, Math. Comp., 66 (1997), pp. 333–344.
[34] ———, *A numerical comparison of methods for solving secular equations*, J. Comp. Appl. Math., 86 (1997), pp. 237–249.
[35] ———, *Analysis of third-order methods for secular equations*, Math. Comp., 67 (1998), pp. 271–286.
[36] Reinsch, C. H., *Smoothing by spline functions*, Num. Math., 10 (1967), pp. 167–183.
[37] ———, *Smoothing by spline functions. II*, Num. Math., 16 (1971), pp. 451–454.
[38] Li, R. C., *Solving the secular equation stably and efficiently*, Technical report, Department of Mathematics, University of California, Berkeley, CA, USA, April 1993. LAPACK Working Note 89.
[39] Ron, D. and Brandt, A., *Renormalization multigrid (RMG): Coarse-to-fine Monte Carlo acceleration and optimal derivation of macroscopic actions*, in Brandt, A., Bernholc, J., and Binder, K. (Eds.), Multiscale Computational Methods in Chemistry and Physics, NATO Science Series III: Computer and Systems Sciences, NATO Science Series, Vol. 177, IOS Press, The Netherlands, 2001.
[40] Sandak, B. and Brandt, A., *Multiscale fast summation of long range charge and dipolar interactions*, J. Comp. Chem. (in press).
[41] Skolnik, M. I. (ed.), *Radar Handbook*, McGraw-Hill Book Company, New York, 1970.
[42] Skolnik, M. I. (ed.), *Introduction to Radar Systems*, 2*nd* ed., McGraw-Hill Book Company, Tokyo, 1983.
[43] Stoer, J. and Bulirsch, R., *Introduction to Numerical Analysis*, Translation of *Einführung in die Numerische*, Springer-Verlag, New York, 1980.
[44] von Matt, U., *Large Constrained Quadratic Problems*, Verlag dr Fachvereine, Zürich, 1993.
[45] Weiss, M., *Analysis of some modified cell-averaging CFAR processes in multiple-target situations*, IEEE Trans. Aero. Elec. Sys., AES–18 (1982), pp. 102–114.