Exact Computation of the Inertia of Symmetric Integer Matrices

Steven Fortune
Bell Laboratories

Abstract

The *inertia* of a symmetric matrix M is the triple (p, z, n), where p, z, and n are the number of positive, zero, and negative eigenvalues of M, respectively. We give an algorithm that computes the inertia of a symmetric integer matrix exactly. The algorithm couples a floating-point eigenvalue subroutine (such as the QR algorithm) to an integer conditioning step. The algorithm requires at most about $\log \mathrm{spread}(M)/\log(1/\epsilon)$ conditioning steps, where ϵ is floating-point precision and the *spread* of M, $\mathrm{spread}(M)$, is the ratio of the maximum magnitude of any eigenvalue to the minimum magnitude of any nonzero eigenvalue. If M is an $n \times n$ matrix, each conditioning step requires a floating-point eigenvalue evaluation of an $n \times n$ matrix, and two multiplies of integer $n \times n$ matrices of norm about $||M||$. In the worst case $\mathrm{spread}(M) \approx ||M||^n$, and the number of conditioning steps is about $n \log ||M||/\log(1/\epsilon)$.

1 Introduction

The *inertia* of a symmetric matrix M is the triple (p, z, n), where p, z, and n are the number of positive, zero, and negative eigenvalues of M, respectively. We describe an algorithm that computes the inertia of M exactly, provided M has integer entries. The algorithm uses both a floating-point eigenvalue subroutine and an exact integer conditioning step.

The *spread* of M, $\mathrm{spread}(M)$, is the ratio of the maximum magnitude of any eigenvalue to the minimum magnitude of any nonzero eigenvalue. Of course, the maximum magnitude of any eigenvalue is the (Euclidean) norm of M. Standard analysis of floating-point eigenvalue evaluation indicates that the absolute error in a computed eigenvalue is about $\epsilon\delta(n)||M||$, where ϵ is floating-point precision and $\delta(n)$ is a slowly-growing function of the matrix size n. Hence if $\mathrm{spread}(M)$ is less than about $1/\epsilon$, the inertia of M can be computed exactly in floating-point arithmetic.

Sylvester's law of inertia states that any matrix congruent to M, i.e. any matrix of the form C^TMC with C nonsingular, has the same inertia as M. (Recall that a matrix similar to M, i.e. a matrix of the form $C^{-1}MC$, has the same eigenvalues as M.) Section 6 below gives a quantitative version of Sylvester's law that bounds $\mathrm{spread}(C^TMC)$ in terms of $\mathrm{spread}(M)$ and properties of C. Using this result and the floating-point eigendecomposition of M, it is possible to construct an integer matrix C so that $\mathrm{spread}(C^TMC)$ is about ϵ times the $\mathrm{spread}(M)$, as long as $\mathrm{spread}(M) > 1/\epsilon$. In particular, $||C^TMC|| \approx ||M||$ and the small eigenvalues of C^TMC are about $1/\epsilon$ times larger than corresponding eigenvalues of M.

This construction immediately suggests an iterative inertia algorithm. The algorithm first evaluates the eigenvalues of M using floating-point arithmetic. Using standard techniques it is easy to compute an error bound on an eigenvalue, and hence certify the correctness of its sign. If the eigenvalue signs cannot be certified as correct, then the algorithm iterates with M replaced with C^TMC. Clearly, the number of required iterations is about $\log \mathrm{spread}(M)/\log(1/\epsilon)$.

The product of the eigenvalues is the determinant of the matrix and each eigenvalue magnitude is at most the norm of M. If M is nonsingular, the magnitude of its determinant is at least 1, so the smallest eigenvalue of M has magnitude at least $1/||M||^{n-1}$ and the spread of M is at most $||M||^n$. (If M is singular, a slightly more complex argument yields a similar bound.) Hence the worst-case number of iterations is $n \log ||M|| / \log(1/\epsilon)$.

Other exact algorithms. It is possible to give a rough comparison of this algorithm with other exact algorithms for inertia. We assume a 'digit' model, where a digit is fixed bit-length integer or floating-point number (i.e. a number fitting into a single computer word). An arithmetic operation on two single-digit numbers costs 1; multiplying a k-digit number by an l-digit number costs kl. For the following M is an $n \times n$ matrix with b-digit integer entries and the spread of M is bounded by an s-digit integer, $s \geq b$. In the worst case s is about nb.

- Conditioned QR (this algorithm): Each conditioning step requires an eigenvalue computation and the matrix product $C^T M C$. The QR algorithm is usually estimated as $O(n^3)$ floating-point operations [8]. The conditioning matrix C has only single-digit entries, so each matrix product requires $O(bn^3)$. The total for all conditioning steps is $O(sbn^3)$, or $O(b^2 n^4)$ in the worst case.
- Characteristic polynomial: The inertia of M can be determined from the coefficients of the characteristic polynomial $\det(M - \lambda I)$ in the indeterminate λ, using the Descartes rule of signs [11]. The characteristic polynomial can be computed by interpolation; this requires a determinant evaluation for n distinct values of λ. Using Chinese remaindering, each determinant evaluation requires about $O(n^4 b + n^3 b^2)$ arithmetic operations, for a total of about $O(n^5 b^2 + n^4 b^3)$. There is no obvious way to improve the algorithm for small s.
- Extended-precision floating point: The QR algorithm will find all eigenvalue signs if implemented with roughly s-digit floating-point arithmetic. This costs $O(n^3 s^2)$, or $O(b^2 n^5)$ in the worst case.

The QR algorithm in extended-precision floating-point either requires an *a priori* estimate of the matrix spread, or it needs to be repeated with increasing precision until eigenvalue signs are determined. The algorithm in this paper does not require an estimate of the matrix spread.

Other work. The computation of eigenvalues is a central theme of numerical analysis [8]. However there has been relatively little emphasis on exact computation. For some symmetric tridiagonal matrices, it is possible to compute the eigenvalues with small relative error [1]. Of course, this implies that the signature can be computed exactly. For certain structured symmetric matrices, the singular value decomposition can also be evaluated with small relative error [5, 6]; this may extend to eigenvalue calculation.

Clarkson [2] gives an algorithm that computes the sign of the determinant of an integer matrix exactly. His algorithm combines floating-point modified Gram-Schmidt orthogonalization with exact integer conditioning steps, and hence has similar flavor to the algorithm described here.

Geometric algorithms on curved objects. Geometric algorithms require predicates that determine properties of geometric objects. A typical predicate is the orientation test on planes, which decides if the intersection point of three planes is above, on, or below a fourth oriented plane. The orientation test can be expressed as the sign of a polynomial in the plane coordinates. Floating-point evaluation of polynomial signs can introduce inconsistencies, because sign-determination may be incorrect due to floating-point roundoff. Exact sign evaluation avoids such inconsistencies but requires software extended-precision arithmetic. Fortunately, the requisite increase in precision is relatively modest if the predicates involve linear geometric objects. Various techniques are known that minimize the performance cost of extended-precision arithmetic [2, 7].

Curved objects are often represented implicitly, as the zeroes of a multivariate polynomial. Geometric predicates on curved objects are much more complicated and expensive than on linear objects, and require much more sophisticated algorithms [10]. Using connections with symbolic algebra, the inertia test may eventually lead to a reasonably efficient analog of the orientation test on curved objects.

The following illustrates the situation in $\mathbb{R}^3$; for more background see [3]. Suppose p_1, p_2, p_3 are real polynomials in x, y, z. Then for each i, the set $\{(x, y, z) : p_i(x, y, z) = 0\}$ forms a surface in $\mathbb{R}^3$. Suppose the ideal $J =< p_1, p_2, p_3 >$ is zero-dimensional, i.e. the three surfaces meet in a finite set of points $V(J)$. Then $\mathbb{R}[x, y, z]/J$ is finite-dimensional over R. Multiplication in $\mathbb{R}[x, y, z]/J$ by a polynomial q is a linear map, and can be expressed as a matrix m_q. A standard result from symbolic algebra is that the eigenvalues of m_q are exactly the values of q on $V(J)$. The matrix m_q is not symmetric, and may have complex eigenvalues corresponding to complex points of $q(V(J))$. However, it is possible to construct a real symmetric matrix M_q whose eigenvalues give information about the real points of $V(J)$. In particular, the rank of M_q is the number of real points in $V(J)$ for which q is not zero, and the signature of M_q is the number of real points of $V(J)$ for which q is positive less the number of real points of $V(J)$ for which q is negative. (The signature of M_q is the number of positive eigenvalues less the number of negative eigenvalues.) Thus the inertia of M_q gives information about the sign sequence of q evaluated at the intersection points of the three surfaces. (However, an efficient "orientation test for surfaces" does not immediately result. For example, the signature calculation does not refer to a specific real point of $V(J)$; furthermore, M_q is rational if the polynomial coefficients are rational, but not necessarily integer.)

The rest of the paper. Section 2 reviews some basic definitions and gives some elementary results about rounding integer matrices. Section 3 outlines the standard error analysis of floating-point eigenvalue evaluation. The actual algorithm to compute inertia appears in section 4. Sections 5 through 7 prove the correctness of the algorithm; in particular section 6 gives a quantitative version of Sylvester's law of inertia. Finally section 8 gives some experimental results with the algorithm.

2 Definitions

Extensive information about the floating-point evaluation of eigenvalues can be found in various standard references [8, 12, 13, 14]. Throughout the paper $||\cdot||$ denotes the Euclidean norm; e_i is the unit vector parallel to the ith coordinate axis; ϵ denotes floating-point precision; and $\#S$ is the number of elements of a finite set S.

Let $M \in \mathbb{R}^{n\times n}$. An *eigenpair* of M is a pair (q, λ), $q \in \mathbb{R}^n$ with $||q|| = 1$ and $\lambda \in \mathbb{R}$ so that $Mq = \lambda q$; q is an *eigenvector* and λ is an *eigenvalue*. An *eigendecomposition* of M is a pair (Q, Λ), $Q, \Lambda \in \mathbb{R}^{n\times n}$, Q orthogonal, $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ with $\lambda_1 \geq \lambda_2 \dots \geq \lambda_n$ so that $Q^T M Q = \Lambda$. $Q = [q_1 \dots q_n]$ means that the columns of Q are $q_1, \dots, q_n$. $\lambda_i(M)$ is the ith largest eigenvalue of M. It is useful to observe that if $x \in \mathbb{R}^n$ is expressed as $x = \sum_i \alpha_i q_i$, then $Mx = \sum_i \alpha_i \lambda_i q_i$.

For real number a, $\lfloor a \rceil$ is the closest integer to a (with half-integer values rounded arbitrarily). For matrix $A = (a_{ij})$, $\lfloor A \rceil = (\lfloor a_{ij} \rceil)$. A is *expanding* if $||Ax|| \geq ||x||$; A is *t-expanding* if $||Ax|| \geq t||x||$.

Lemma 2.1 *Let matrix $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$, and set $\Delta A = A - \lfloor A \rceil$. Then*

1. $||\Delta A|| \leq (\sqrt{n}/2)$.

2. *An expanding matrix is nonsingular.*

3. *If A is $\sqrt{n}$-expanding, then $\lfloor A \rceil$ is $(\sqrt{n}/2)$-expanding.*

4. *If A is t-expanding, then for all x, $||A^{-1}x|| \leq ||x||/t$.*

5. *If A is $\sqrt{n}$-expanding, then $||\lfloor A \rceil^{-1}x|| \leq 2||A^{-1}x||$.*

Proof: Only the last statement is nontrivial. We have

$$\begin{aligned} ||\lfloor A \rceil^{-1}x|| &= ||\lfloor A \rceil^{-1}(\lfloor A \rceil + \Delta A)A^{-1}x|| \\ &\leq ||A^{-1}x|| + ||\lfloor A \rceil^{-1} \cdot \Delta A \cdot A^{-1}x|| \\ &\leq ||A^{-1}x|| + (2/\sqrt{n})(\sqrt{n}/2)||A^{-1}x||. \end{aligned}$$

□

3 Floating-point eigenvalue computation

The floating-point computation of eigenvalues has been a central topic of numerical analysis for several decades [8]. However, there seems to be no complete analysis of, say, the QR algorithm in floating-point arithmetic. This section outlines the assumptions made about the subroutine `eig`. The assumptions mirror the conventional error analysis of the QR algorithm [1].

> *Assumption:* There is a function $\delta(n)$ so that for any symmetric $n \times n$ floating-point matrix M, the call
>
> $$Q, \Lambda := \texttt{eig}(M);$$
>
> returns computed floating-point matrices $\hat{Q}$ (the approximate eigenvectors) and $\hat{\Lambda}$ (the approximate eigenvalues) so that there is a perturbed matrix $M' \in \mathbb{R}^{n\times n}$ and an orthogonal matrix $Q' \in \mathbb{R}^{n\times n}$ satisfying
>
> $$\begin{aligned} (Q')^T M' Q' &= \Lambda' = \hat{\Lambda} = \mathrm{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_n), \quad \hat{\lambda}_1 \geq \ldots \geq \hat{\lambda}_n \\ Q' &= \hat{Q} + \Delta Q, \quad ||\Delta Q|| \leq \delta(n)\epsilon \\ M' &= M + \Delta M, \quad ||\Delta M|| \leq \delta(n)\epsilon||M|| \end{aligned}$$
>
> where $\delta(n)$ satisfies
>
> $$\delta(n) \geq 1 \quad \text{and} \quad 64n^{5/2}\delta(n)(\delta(n){+}4)\,\epsilon < 1.$$

For the QR algorithm, $\delta(n)$ is usually described as a modestly-growing function of the matrix size n, e.g. $\delta(n) \leq 10n$ [1]. It seems difficult to obtain a worst-case upper bound on $\delta(n)$ (it appears to be $O(nt)$, where t is the number of iterations of the QR algorithm [12], but there is no obvious bound on t).

The rather odd condition on $\delta(n)$ comes from thereom 4.1 below. It can be interpreted either as the requirement that ϵ vary as a function of n, or as an upper bound on the allowed values of n, for ϵ fixed. For example, the values $\delta(n) = 10n$ and $\epsilon = 2^{-53}$ imply that $n \leq 500$. Experimental evidence (section 8) suggests that the condition on $\delta(n)$ is too conservative.

Let

$$Q^T M Q = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \quad \lambda_1 \geq \ldots \geq \lambda_n$$

```
// Input:
//     M      n × n symmetric integer matrix
// Output:
//     the inertia of M
//
procedure inertia(M)
      r := rank(M);
      Q,Λ := eig(M);
      while there are fewer than r sign-certified eigenvalues
            M := condition(M, Q, Λ);
            Q,Λ := eig(M);
      end while
      return (#{i : λ_i > 0}, n − r, #{i : λ_i < 0});
```

Figure 1: Procedure `inertia`

be the true eigendecomposition of matrix M. Standard perturbation theory [8] implies that

$$|\lambda_i - \hat{\lambda}_i| \leq ||\Delta M||.$$

The relation between the computed eigenvector $\hat{q}_i$ and the true eigenvector q_i is more complicated; roughly, the relative error between the two is inversely proportional to the separation of the eigenvalue λ_i from the rest of the eigenvalues. The effect of this error forms a significant portion of the analysis in section 7.

Proposition 3.1 *Let $(\hat{q}, \hat{\lambda})$ be an eigenpair computed by* `eig`*(M). Then $||M\hat{q} - \hat{\lambda}\hat{q}|| < 4\epsilon\delta(n)||M||$.*

Proof: Straightforward; omitted. □

4 The inertia algorithm

Procedure `inertia` appears in figure 1. It uses the floating-point subroutine `eig` (section 3), a matrix-conditioning subroutine `condition` (figure 3 and section 4.2), and a test to certify that the sign of a computed eigenvalue is correct (section 4.1). Procedure `condition`(M) returns a matrix with the same inertia as M and approximately the same norm as M, but with small eigenvalues increased in magnitude. Clearly, `condition` is repeatedly called until it returns a matrix whose nonzero eigenvalues are large enough so that their signs can be certified as correct.

The *spread* of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is defined by

$$\text{spread}(A) = \min_{\lambda_i(A)\neq 0} \frac{||A||}{|\lambda_i(A)|}.$$

Let $q(n) = n(\delta(n)+4)$ and $p(n) = 64n^{3/2}\delta(n)q(n)$. The assumptions on `eig` imply that $\epsilon p(n) < 1$.

Theorem 4.1 *Procedure* `inertia`*(M) correctly computes the inertia of M, calling* `condition` *at most*

$$c = \frac{\log(\text{spread}(M))}{-\log(\epsilon p(n))}$$

times. The norm of any matrix returned by `condition` *is at most $q(n)^c||M||$.*

```
// Input:
//      M       n × n symmetric integer matrix
//      x, μ    candidate eigenpair
// Output:
//      true if sign of λ is correct (see Lemma 4.2)
procedure certify(M, x, μ)
        ρ := ||Mx − μx||;
        d := ρ√(n+1) ;
        return 0 ∉ [λ−d, λ+d];
```

Figure 2: Procedure `certify`. The arithmetic is assumed to be performed exactly (see text).

4.1 Eigenvalue sign certification

This section discusses a test that can certify the correctness of the sign of a computed eigenvalue $\hat{\lambda}_i$.

Clearly, if $|\hat{\lambda}_i| > ||\Delta M||$, then the sign of λ_i is the same as the sign of $\hat{\lambda}_i$. Of course, $||\Delta M||$ is unknown. For the QR algorithm, the bound $||\Delta M|| < c(t+1)n\epsilon||M||$ probably holds, where t is the number of iterations of the algorithm. Even if the constant c could be determined, this test would be quite conservative.

A standard result [12, page 79] is that for any real μ and any unit vector $x \in \mathbb{R}^n$, there is an eigenvalue of M in the interval $[\mu-\rho, \mu+\rho]$, where $\rho = ||Mx - \mu x||$. This result can sometimes be used to certify the signs of a set of eigenvalues. Suppose $(\hat{q}_1, \hat{\lambda}_1), \ldots, (\hat{q}_k, \hat{\lambda}_k)$ are some or all of the computed eigenpairs. Choose $\rho_i > ||M\hat{q}_i - \hat{\lambda}_i\hat{q}_i||$, $1 \le i \le k$. If none of the intervals $[\hat{\lambda}_i - \rho_i, \hat{\lambda}_i + \rho_i]$ contains zero and all of them are pairwise disjoint, then clearly there are k nonzero eigenvalues of M with the same sign sequence as $\hat{\lambda}_1, \ldots, \hat{\lambda}_k$. Unfortunately, if two of the intervals overlap, then there is no guarantee that there are two eigenvalues (or a repeated eigenvalue) within the union of the two intervals. Fortunately, the situation can be rectified by widening the intervals by a factor of about $\sqrt{n}$ (see procedure `certify` in figure 2).

Lemma 4.2 *Let (x_i, μ_i), $1 \le i \le k$, be a subset of the eigenpairs computed by* `eig`*. If* `certify`*(M, x_i, μ_i) returns true for $1 \le i \le k$, then there are k eigenvalues of M with the same sign sequence as $\mu_1, \ldots, \mu_k$ (each eigenvalue of M can be repeated up to its multiplicity). If* $\mathrm{spread}(M) \le 1/(5\epsilon\delta(n)\sqrt{n+1})$*, then* `certify` *returns true for $n - \mathrm{rank}(M)$ computed eigenpairs.*

The proof of lemma 4.2 requires the following two lemmas. For $R \subseteq \mathbb{R}$ and M with eigendecomposition (Q, Λ), define the subspace $S_M(R) = \mathrm{span}(\{q_i : \lambda_i \in R\})$.

Lemma 4.3 *Let $\mu \in \mathbb{R}$, $x \in \mathbb{R}^n$ with $||x|| = 1$, $\rho = ||Mx - \mu x||$, and $s > 1$. Then there is $w \in S_M([\mu - s\rho, \mu + s\rho])$ with $||w - x|| < 1/s$.*

Proof: Write $x = \sum_{i=1}^n \alpha_i q_i$ with $\sum_{i=1}^n \alpha_i^2 = 1$. Then

$$\rho = ||Mx - \mu x|| = ||\sum_{i=1}^n \alpha_i(\lambda_i - \mu)q_i|| = \left(\sum_{i=1}^n (\alpha_i^2(\lambda_i - \mu)^2\right)^{1/2}.$$

Let $K = \{i : |\lambda_i - \mu| < s\rho\}$, then $\rho^2 > \sum_{i \notin K} \alpha_i^2 (s\rho)^2$, so $\sum_{i \notin K} \alpha_i^2 < 1/s^2$. Setting $v = \sum_{i \in K} \alpha_i q_i$, we have $||v - x|| = ||\sum_{i \notin K} \alpha_i q_i|| < 1/s$. □

```
// Input:
//     M       n × n symmetric integer matrix
//     Q,Λ     eigendecomposition computed by eig
// Output:
//     a conditioned matrix congruent to M
procedure condition(M, Q, Λ)
        m := max(λ_1, −λ_n); // ||M||
        for i := 1 to n
            s_i := sqrt(min(1/ε, m/λ_i));
        S := diag(s_1, ..., s_n);
        B := Q√n S;
        C := ⌊B⌉;
        return C^T M C;
```

Figure 3: Procedure `condition`. All computation uses floating-point arithmetic, except the computation of C^TMC, which is done exactly with integer arithmetic.

Lemma 4.4 *Suppose* $v_1, \ldots, v_k \in \mathbb{R}^n$ *are orthonormal and for* $1{\leq}i{\leq}k$, $w_i = v_i{+}\delta_i \in \mathbb{R}^n$ *with* $||\delta_i||{<}1/\sqrt{n}$. *Then the* $\{w_i\}$ *are linearly independent.*

Proof: Without loss of generality $k{=}n$ and $v_i{=}e_i$, $1{\leq}i{\leq}n$. Then $\det([w_1 \ldots w_n]) = \det(I + [\delta_1 \ldots \delta_n]) > 0$, as $\det([\delta_1 \ldots \delta_n]) < 1$. □

Proof of Lemma 4.2: By the analysis of `eig`, there is an orthonormal set $\{r_i : 1{\leq}i{\leq}k\}$ so that $||r_i - x_i|| < \delta(n)\epsilon$ (the $\{r_i\}$ are a reindexing of a subset of the $\{q_i'\}$). Let $\rho_i = ||Mx_i - \mu_i x_i||$ and $I_i = [\mu_i - \rho_i\sqrt{n{+}1},\ \mu_i + \rho_i\sqrt{n{+}1}]$. Let $P = \{i : \mu_i > 0\}$ and $I = \cup_{i \in P} I_i$. By lemma 4.3, for each $i{\in}P$, there is $w_i{\in}S_M(I)$ with $||w_i - x_i|| < 1/\sqrt{n{+}1}$, so $||q_i - w_i|| < \delta(n)\epsilon + 1/\sqrt{n{+}1} < 1/\sqrt{n}$. By lemma 4.1, the $\{x_i\}$ are linearly independent, hence $\dim(S_M(I)) > \#P$, and there are at least $\#P$ positive eigenvalues, counted up to multiplicity. A similar argument counts the negative eigenvalues.

Suppose $\text{spread}(M) \leq 1/(5\epsilon\delta(n)\sqrt{n{+}1})$, then any nonzero eigenvalue λ_i of M has magnitude at least $5\epsilon\delta(n)||M||\sqrt{n{+}1}$. Let $(\hat{q}_i, \hat{\lambda}_i)$ be the corresponding computed eigenpair; then $|\lambda_i - \hat{\lambda}_i| < \epsilon\delta(n)||M||$ so $|\hat{\lambda}_i| \geq 4\epsilon\delta(n)||M||\sqrt{n{+}1}$. By proposition 3.1, $\rho = ||M\hat{q}_i - \hat{\lambda}_i\hat{q}_i|| < 4\epsilon\delta(n)||M||$. Hence `certify` returns true. □

It is easy to obtain a version of `certify` that uses floating-point arithmetic. The computed interval radius d must be increased by an estimate of the error in computing ρ. Lemma 4.2 would be essentially unchanged; the details are slightly tedious and are omitted.

4.2 Matrix conditioning

Procedure `condition` appears in figure 3. It uses the floating-point eigendecomposition of M to determine a floating-point matrix B and an integer matrix $C = \lfloor B \rceil$. It is easy to verify that C is expanding, hence C is invertible by lemma 2.1. The computation of C^TMC is exact, hence C^TMC has the same inertia as M, by Sylvester's law of inertia.

To understand the behavior of `condition`, it is helpful to consider its behavior if the eigendecomposition were exact and C were replaced by QS. Since Λ is diagonal, its eigenvalues are its diagonal entries. The

$$
\begin{array}{lll}
\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) & Q = [q_1 \ldots q_n] & \text{true eigendecomposition of } M \\
\Lambda' = \mathrm{diag}(\lambda'_1, \ldots, \lambda'_n) & Q' = [q'_1 \ldots q'_n] & \text{true eigendecomposition of } M' \\
\hat{\Lambda} = \Lambda' & \hat{Q} = [\hat{q}_1 \ldots \hat{q}_n] & \text{computed eigendecomposition} \\
s_j = \min\left(\frac{1}{\epsilon}, \frac{||M||}{||\lambda_j||}\right)^{1/2} & S = \mathrm{diag}(s_1, \ldots, s_n) & \text{true scaling matrix of } M \\
s'_j = \min\left(\frac{1}{\epsilon}, \frac{||M'||}{||\lambda'_j||}\right)^{1/2} & S' = \mathrm{diag}(s'_1, \ldots, s'_n) & \text{true scaling matrix of } M' \\
B' = Q'\sqrt{n}S' & \hat{B} = \mathrm{float}(\hat{Q}\sqrt{n}\hat{S}) &
\end{array}
$$

Figure 4: Summary of notation. $\hat{B}$ and $\hat{S}$ are the values computed in floating-point within `condition`.

transformation from Λ to $S^T \Lambda S$ replaces diagonal entries with magnitude larger than $\epsilon||M||$ by $||M||$ and multiplies small entries by $1/\epsilon$. Clearly M and Λ have the same eigenvalues and norm. Hence the transformation from M to $(QS)^T M(QS) = S^T \Lambda S$ increases small eigenvalues by a factor of $1/\epsilon$, but does not increase norm.

Of course, the eigendecomposition is computed in floating-point arithmetic, and C is rounded to have integer entries. However, the following two lemmas hold; proofs are given below. These two lemmas together with lemma 4.2 imply theorem 4.1 (using $5\delta(n)\sqrt{n+1} < p(n)$).

Lemma 4.5 *Let* $R =$ `condition`(M). *Then* $||R|| \leq q(n)||M||$.

Lemma 4.6 *Let* $R =$ `condition`(M). *Then*

$$\frac{\mathrm{spread}(R)}{\mathrm{spread}(M)} \;\leq\; p(n)\, \max(\epsilon,\, 1/\mathrm{spread}(M)).$$

5 Proof of Lemma 4.5

This section proves lemma 4.5 and section 7 proves lemma 4.6. The proofs require careful distinction among true, computed, and perturbed quantities. An unadorned symbol, e.g. q_i, is usually a true value; a hatted symbol, e.g. $\hat{q}_i$, is a computed value; a primed symbol, e.g. q'_i, is a perturbed value. For convenience, the various symbols are summarized in figure 5.

Proof of Lemma 4.5: We have

$$
\begin{array}{rcl}
||B'|| & \leq & \sqrt{n/\epsilon} \\
||(B')^T M' B'|| & = & n\,||(S')^T \Lambda' S'|| \;=\; n\,||M'|| \\
||M'B'|| & = & \sqrt{n}\,||Q'\Lambda' S'|| \;=\; \sqrt{n}\,||M'||
\end{array}
$$

since $(S')^T \Lambda' S'$ and $\Lambda' S'$ are diagonal matrixes with all entries at most $||M'||$. Similarly $||(B')^T M'|| = \sqrt{n}||M'||$. Define Δ by

$$\lfloor \hat{B} \rceil = B' + \Delta$$

Then by using lemma 2.1 and by estimating the floating-point error in the computation of $\hat{B}$ from $\hat{Q}$ and $\hat{\Lambda}$ as well as the difference between $\hat{Q}$ and Q', we get

$$\begin{aligned} ||\Delta|| &\leq ||B' - \hat{B}|| + ||\hat{B} - \lfloor \hat{B} \rceil|| \\ &\leq (\delta(n){+}2)\epsilon||B'|| \;+\; \sqrt{n}/2 \\ &\leq \sqrt{n}, \end{aligned}$$

where the last step uses the inequality $(\delta(n)+2)\sqrt{\epsilon} < 1/2$, which follows easily from the condition on $\delta(n)$ in the description of `eig`. The matrix R returned by `condition` satisfies

$$\begin{aligned} R &= \lfloor \hat{B} \rceil^T M \lfloor \hat{B} \rceil \\ &= (B' + \Delta)^T M\,(B' + \Delta) \\ &= (B')^T M B' \;+\; (B')^T M\,\Delta \;+\; \Delta^T M\,B' + \;\Delta^T M\,\Delta \\ &= (B')^T (M' - \Delta M) B' \;+\; (B')^T M\,\Delta \;+\; \Delta^T M\,B' + \;\Delta^T M\,\Delta \\ &= (B')^T M' B' \;-\; (B')^T \Delta M\,B' \;+\; (B')^T M\,\Delta \;+\; \Delta^T M\,B' \;+\; \Delta^T M\,\Delta \\ ||R|| &\leq n||M'|| + \sqrt{n/\epsilon} \cdot \epsilon\delta(n)||M|| \cdot \sqrt{n/\epsilon} + \sqrt{n}||M|| \cdot \sqrt{n} + \sqrt{n} \cdot \sqrt{n}||M|| + \sqrt{n} \cdot ||M|| \cdot \sqrt{n} \\ &\leq (4{+}\delta(n))n||M||. \end{aligned}$$

□

6 Congruences

Sylvester's law of inertia [8, 9] states that symmetric $A \in \mathbb{R}^{n\times n}$ and C^TAC have the same inertia as long as $C \in \mathbb{R}^{n\times n}$ is nonsingular. This section gives a version of Sylvester's law which bounds the change in eigenvalue magnitude. In the following standard result[8, 14], S varies over all k-dimensional subspaces of $\mathbb{R}^{n\times n}$.

Theorem 6.1 *(The Courant-Fischer Minimax Theorem) If $A \in \mathbb{R}^{n\times n}$ is symmetric, then*

$$\lambda_k(A) = \max_{\dim(S)=k} \; \min_{0 \neq y \in S} \frac{y^T A y}{y^T y}.$$

Lemma 6.2 *Let $C \in \mathbb{R}^{n\times n}$ be nonsingular and let $A \in \mathbb{R}^{n\times n}$ be symmetric with p positive and q negative eigenvalues. Index the eigenvalues $\mu_1, \ldots, \mu_n$ and corresponding unit eigenvectors $v_1, \ldots, v_n$ so that*

$$\frac{\mu_1}{\rho_1^2} \geq \frac{\mu_2}{\rho_2^2} \geq \ldots \geq \frac{\mu_p}{\rho_p^2} \geq 0 \geq \frac{\mu_{n-q+1}}{\rho_{n-q+1}^2} \geq \ldots \geq \frac{\mu_n}{\rho_n^2}$$

where $\rho_i = ||C^{-1}v_i||$. Then for $1 \leq k \leq p$,

$$\lambda_k(C^TAC) \geq \frac{\mu_k}{n\rho_k^2}$$

and similarly for $n-q+1 \leq k \leq n$,

$$-\lambda_k(C^TAC) \geq -\frac{\mu_k}{n\rho_k^2}.$$

Proof: We show the case $1 \le k \le p$; the other case is similar. Let $S = \mathrm{span}(C^{-1}v_i)$; then S is a k-dimensional subspace of $\mathbb{R}^n$ since C is nonsingular and the $\{v_i\}$ form a basis. We show $y^T C^T A C y / y^T y \ge \mu_k/(n\rho_k^2)$ for any nonzero $y \in S$; the lemma then follows by the Courant-Fischer Minimax theorem.

For any nonzero $y \in S$ and some $\alpha_i \in \mathbb{R}$ not all zero we have

$$\begin{aligned} y &= \sum_{i=1}^{k} \alpha_i (C^{-1} v_i) \\ Cy &= \sum_{i=1}^{k} \alpha_i v_i \\ y^T C^T A C y &= (\sum_{i=1}^{k} \alpha_i v_i)^T A (\sum_{i=1}^{k} \alpha_i v_i) = (\sum_{i=1}^{k} \alpha_i v_i)^T (\sum_{i=1}^{k} \alpha_i \mu_i v_i) = \sum_{i=1}^{k} \alpha_i^2 \mu_i, \end{aligned}$$

where the last equality uses the orthonormality of the eigenvectors $\{v_i\}$. Also we have

$$y^T y = ||\sum_{i=1}^{k} \alpha_i (C^{-1} v_i)||^2 \le n \sum_{i=1}^{k} \alpha_i^2 ||C^{-1} v_i||^2 = n \sum_{i=1}^{k} \alpha_i^2 \rho_i^2.$$

Hence

$$\frac{y^T C^T A C y}{y^T y} = \frac{1}{n} \frac{\sum_{i=1}^{k} \alpha_i^2 \mu_i}{\sum_{i=1}^{k} \alpha_i^2 \rho_i^2} \ge \frac{1}{n} \min_{1 \le i \le k} \left(\frac{\mu_i}{\rho_i^2} \right) = \frac{\mu_k}{n \rho_k^2}.$$

□

Corollary 6.3 *Let $R = C^T A C$ with A, C, $\{\mu_i\}$, and $\{\rho_i\}$ as in lemma 6.2. Then*

$$\min_{\lambda_k(R) \ne 0} |\lambda_k(R)| \ \ge \ \frac{1}{n} \min_{\lambda_k(A) \ne 0} \frac{|\lambda_k(A)|}{\rho_k^2}.$$

7 Proof of lemma 4.6

This section uses the notation as summarized in figure 5. By Corollary 6.3, for $C^T M C$ to have smaller spread than M, $||C^{-1} q_j||$ should be small for each eigenvector q_j of M with $|\lambda_j|$ small. If $B = Q\sqrt{n}S$ were computed exactly with the exact eigendecomposition (Q, Λ) of M, then $||B^{-1} q_j|| = 1/(s_j \sqrt{n})$ would be appropriately small. Furthermore, the $\sqrt{n}$ term in the formula for B guarantees that B is $\sqrt{n}$-expanding, so by lemma 2.1(5), $||\lfloor B \rceil^{-1} q_j|| < 2/(s_j \sqrt{n})$. Unfortunately the computed eigenvectors $\{\hat{q}_j\}$ can be rather far from the true eigenvectors, and this simple argument only shows that $||\hat{B}^{-1} \hat{q}_j||$ is small, where $\hat{B} = \mathrm{float}(\hat{Q}\sqrt{n}\hat{S})$. It is more involved to prove that that $||\hat{B}^{-1} q_j||$ is small.

Let S and T be subspaces of $\mathbb{R}^n$ of the same dimension, then define

$$\mathrm{dist}(S, T) = \max_{s \in S, ||s|| = 1} \min_{t \in T} ||s - t||$$

(S and T can be interchanged without change; see [8] for an alternate definition). If unit vector q is orthogonal to S, then for any vector $t \in T$, $|q \cdot t| \le \mathrm{dist}(S, T)$.

Theorem 7.1 *Let $I \subset \{1,\ldots,n\}$, $I \neq \emptyset$. As long as*

$$\sigma = \min_{i\in I, j\notin I} |\lambda_i - \lambda_j| > 4||\Delta M||$$

then

$$\operatorname{dist}(\operatorname{span}_{i\in I}(q_i), \operatorname{span}_{i\in I}(q'_i)) < \frac{4||\Delta M||}{\sigma}.$$

Proof: This is essentially a restatement of theorem 2.8 of [13, page 238], with slightly weakened bounds on σ allowing a slightly simpler statement. □

Lemma 7.2 *If $\lambda_j \neq \lambda_k$ then $|q'_k \cdot q_j| < 4n||\Delta M||/|\lambda_k - \lambda_j|$.*

Proof: Without loss of generality we can assume $k < j$, so $\lambda_k > \lambda_j$. If $\lambda_k - \lambda_j < 4n||\Delta M||$, then $4n||\Delta M||/(\lambda_k - \lambda_j) > 1 \geq |q'_k \cdot q_j|$, as q'_k and q_j are unit vectors.

Otherwise we can choose an m so that $k \leq m < j$ and $\lambda_m - \lambda_{m+1} \geq (\lambda_k - \lambda_j)/n \geq 4||\Delta M||$. Set $I = \{m+1,\ldots,n\}$. By theorem 7.1,

$$\operatorname{dist}(\operatorname{span}_{i\in I}(q_i), \operatorname{span}_{i\in I}(q'_i)) < \frac{4||\Delta M||}{\lambda_m - \lambda_{m+1}}$$

Since q'_k is orthogonal to $\operatorname{span}_{i\in I}(q'_i)$ and $q_j \in \operatorname{span}_{i\in I}(q_i)$,

$$|q'_k \cdot q_j\ | < \frac{4||\Delta M||}{\lambda_m - \lambda_{m+1}} < \frac{4n||\Delta M||}{\lambda_k - \lambda_j}.$$

□

Lemma 7.3 *Let $\beta(n) = 4n^{3/2}\delta(n)$. Then for $1{\leq}j{\leq}n$, $||(Q'S')^{-1}q_j|| < 3\sqrt{\beta(n)}/s_j$.*

Proof (sketch): Assume $\lambda_i \approx \lambda'_i$ and hence $s_i \approx s'_i$. Write q_j as $\sum \alpha_k q'_k$ where $\alpha_j = q'_k \cdot q_j$ and $\sum \alpha_j^2 = 1$. We have

$$||(Q'S')^{-1}q_j||^2 = ||\sum\left(\frac{\alpha_k}{s_k}\right)e_k||^2 = \sum\left(\frac{\alpha_k}{s_k}\right)^2$$

and we want an upper bound of about $1/s_j^2$. If $s_k{\gtrsim}s_j$ then the magnitude of α_k does not matter. If $s_k \ll s_j$ then $(1/s_k)^2 \gg (1/s_j)^2$ and the magnitude of α_k does matter. However then $|\lambda_k| \gg |\lambda_j|$ and by lemma 7.2, $\alpha_k \approx ||\Delta M||/|\lambda_k - \lambda_j| \approx ||\Delta M||/|\lambda_k|$ so

$$\left(\frac{\alpha_k}{s_k}\right)^2 = \alpha_k\frac{\alpha_k}{s_k^2} \lesssim \alpha_k\frac{||\Delta M||/|\lambda_k|}{||M||/|\lambda_k|} \leq \alpha_k\epsilon\delta(n),$$

which is (almost) negligibly small. The detailed proof, given in the appendix, accounts for the difference between s_k and s'_k as well as optimizing the provable bound on $||(Q'S')^{-1}q_j||$ in the case that $|\lambda_j| \approx \epsilon||M||$ and $|\lambda_k| \approx ||\Delta M||$. □

The following lemma accounts for the difference between $Q'\sqrt{n}S'$ and $\hat{B}$, resulting primarily from the difference between $\hat{Q}$ and Q'.

Lemma 7.4 *Let $\hat{B}$ be the value of B computed in* `condition` *and $\beta(n)$ as in lemma 7.3. Then for $1 \le j \le n$, $||\hat{B}^{-1}q_j|| < (4\sqrt{\beta(n)/n})/s_j$.*

Proof: For simplicity we ignore the floating-point error in the computation of $\hat{B}$ from $\hat{Q}$ and $\hat{S}$ (which is small relative to other errors anyway). Since $||Q' - \hat{Q}|| \le \delta(n)\epsilon$, we have

$$Q'\hat{Q}^{-1}q_j = q_j + \phi, \quad ||\phi|| \le \delta(n)\epsilon$$

(in fact such a relationship holds for any unit vector instead of q_j), so

$$\begin{aligned}
\hat{B}^{-1} &= (\hat{Q}\sqrt{n}\hat{S})^{-1} \\
&= \frac{1}{\sqrt{n}}(\hat{S}^{-1}S')\ (Q'S')^{-1}\ Q'\hat{Q}^{-1} \\
||\hat{B}^{-1}q_j|| &\le \frac{||\hat{S}^{-1}S'||\ (||(Q'S')^{-1}q_j|| + ||(Q'S')^{-1}\phi||)}{\sqrt{n}} \\
&\le \frac{(1 + 2\sqrt{n}\epsilon)\ \left(3\sqrt{\beta(n)}/s_j + \delta(n)\epsilon\right)}{\sqrt{n}} \\
&\le \frac{4\sqrt{\beta(n)}}{s_j\sqrt{n}}
\end{aligned}$$

where the second-to-last inequality uses both lemma 7.3 and $||(Q'S')^{-1}|| = 1$ while the last inequality uses $\epsilon \le 1/s_j^2 < 1/s_j$. $\square$

Proof of Lemma 4.6: We have

$$\frac{\mathrm{spread}(R)}{\mathrm{spread}(M)} = \frac{||R||}{||M||} \frac{\min\limits_{\lambda_i(M)\ne 0} |\lambda_i(M)|}{\min\limits_{\lambda_i(R)\ne 0} |\lambda_i(R)|}$$

Set $\rho_i = ||C^{-1}q_i||$, then using $C = \lfloor \hat{B} \rceil$ and lemmas 2.1(5) and 7.4

$$\rho_i^2 = ||C^{-1}q_i||^2 \le 4||\hat{B}^{-1}q_i||^2 \le \frac{16\beta(n)}{n}\frac{1}{s_i^2}$$

so by Corollary 6.3

$$\begin{aligned}
\min_{\lambda_i(R)\ne 0} |\lambda_i(R)| &\ge \frac{1}{n} \min_{\lambda_i(M)\ne 0} \frac{|\lambda_i(M)|}{\rho_i^2} \\
&\ge \frac{1}{n}\frac{n}{16\beta(n)} \min_{\lambda_i(M)\ne 0} |\lambda_i(M)| s_i^2 \\
&= \frac{1}{16\beta(n)} \min_{\lambda_i(M)\ne 0} |\lambda_i(M)| \min\left(\frac{||M||}{|\lambda_i(M)|}, \frac{1}{\epsilon}\right) \\
&= \frac{1}{16\beta(n)} \min\left(||M||, \frac{1}{\epsilon} \min_{\lambda_i(M)\ne 0} |\lambda_i(M)|\right) \\
\frac{\min\limits_{\lambda_i(M)\ne 0} |\lambda_i(M)|}{\min\limits_{\lambda_i(R)\ne 0} |\lambda_i(R)|} &\le 16\beta(n) \max(\epsilon, 1/\mathrm{spread}(M)).
\end{aligned}$$

The lemma follows since $||R||/||M|| \le q(n)$ by Lemma 4.5 and $p(n) = 16\beta(n)q(n)$. $\square$

type	n	init spread	init norm	max $\frac{\|\|R\|\|}{\|\|M\|\|}$	max $\frac{\text{spread}(R)}{\text{spread}(M)}$	max $\Delta\lambda$
C^TBC	10	10^{36}	10^{18}	13	4.8ϵ	$7\epsilon\|\|M\|\|$
C^TBC	20	10^{36}	10^{18}	26	1.3ϵ	$22\epsilon\|\|M\|\|$
C^TBC	30	10^{36}	10^{18}	42	1.4ϵ	$13\epsilon\|\|M\|\|$
C^TBC	40	10^{36}	10^{18}	55	1.1ϵ	$15\epsilon\|\|M\|\|$
C^TBC	50	10^{36}	10^{18}	66	1.4ϵ	$27\epsilon\|\|M\|\|$
U^TU	10	10^{115}	10^{12}	16	7.0ϵ	$21\epsilon\|\|M\|\|$
U^TU	20	10^{230}	10^{13}	35	9.0ϵ	$19\epsilon\|\|M\|\|$
U^TU	30	10^{336}	10^{13}	56	1.4ϵ	$17\epsilon\|\|M\|\|$

Figure 5: Statistics for execution of `inertia` on sample matrices. The first four columns are matrix type, size, approximate initial spread, and approximate initial norm. The last three columns are the maximum over all iterations of the change in norm, change in spread (relative to ϵ), and maximum absolute error in any eigenvalue computed in floating point (relative to $\epsilon||M||$).

8 Experiments

A prototype implementation of the algorithm was coded using both Maple for integer matrix operations and Matlab for eigenvalue computation in hardware floating-point. The algorithm was tested using two classes of symmetric integer unimodular matrices M:

- $M = U^TU$, where U is upper triangular with unit diagonal and upper off-diagonal elements chosen as random integers in the range $1 \ldots b$. These matrices tend to have one very small eigenvalue and the rest of magnitude about b.
- $M = C^TBC$, where B is block diagonal with 2×2 blocks of the form $\begin{bmatrix} r-1 & r \\ r & r+1 \end{bmatrix}$ with r a randomly chosen integer, and C a unimodular matrix similarly composed of 2×2 blocks. These matrices tend to have about $n/2$ eigenvalues of magnitude $||M||$ and the rest of magnitude about $1/||M||$.

Figure 5 summarizes the behavior of the algorithm on sample matrices. The last three columns are the most interesting. For each matrix, the last column gives the observed absolute error in all of the floating-point eigenvalue computations, normalized to ϵ and matrix norm. (Extended-precision floating-point in Maple was used to obtain relatively accurate estimates of all eigenvalues.) For these matrices, the estimate $\delta(n) \lesssim 2n$ is reasonable. The second-to-last column indicates that `condition` always reduced spread by a factor close to ϵ. Hence the number of calls to `condition` is about $\log \text{spread}(M)/\log 1/\epsilon$. The third-to-last column indicates that `condition` increases matrix norm by a factor of at most $2n$. Thus the factors $p(n)$ and $q(n)$ that occur in lemmas 4.5 and 4.6 appear to be very conservative.

9 Open problems

1. The `inertia` algorithm computes the sign but not the magnitude of each eigenvalue. It would be interesting to obtain an efficient algorithm that could compute eigenvalues with good relative accuracy, using arithmetic with precision substantially less than $O(1/\text{spread}(M))$.

2. The presentation in this paper carefully separates floating-point eigenvalue computation from integer conditioning. This leads to a clean algorithm with fairly simple analysis. However, with a more tight coupling of the two parts, it might be possible to obtain an algorithm that is faster in the worst case.

3. The floating-point error analysis of the asymmetric eigenvalue problem is significantly more complicated than the symmetric case. In particular, the absolute error in computed eigenvalues is about $\kappa_{M'}\epsilon||M||$, where $\kappa_{M'}$ is the condition number of an appropriate associated matrix M' [8]. For some matrices M, $\kappa_{M'}$ can be very large. Assuming the matrix M has integer entries, it might be possible to eliminate $\kappa_{M'}$ from the error bound with an algorithm in the same style as `inertia`, i.e., by combining integer conditioning steps with floating-point eigenvalue computation.

Acknowledgements. Thanks to Ken Clarkson, Lakshman Yagati, Margaret Wright, and Steve Vavasis for helpful comments.

References

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

[2] K. Clarkson, Safe and effective determinant evaluation, *Proc. 33rd Annual IEEE Symp. on Found. of Computer Science*, pp. 387–395, 1992.

[3] D. Cox, J. Little, D. O'Shea, *Using Algebraic Geometry*, Springer, 1998.

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

[5] J. Demmel, *Computing the Singular Value Decomposition with High Relative Accuracy*, manuscript, 1998.

[6] J. Demmel, *Accurate SVDs of Structured Matrices*, manuscript, 1998.

[7] Static Analysis yields efficient exact integer arithmetic for computational geometry, with Chris Van Wyk, *ACM Trans. Graphics*, 15(3), pp. 223–248, 1996.

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

[9] I.N. Herstein, *Topics in Algebra*, Xerox College Publishing, 1964.

[10] J. Keyser, T. Culver, D. Manocha, S. Krishnan, MAPC: a library for efficient and exact manipulation of algebraic points and curves, *Proc. 15th Annual Symp. Comp. Geom*, pp. 360–369, 1999.

[11] M. Mignotte, *Mathematics for Computer Algebra*, Springer-Verlag, New York, 1992.

[12] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall Series in Computational Mathematics, 1980.

[13] G.W. Stewart, J. Sun, *Matrix Perturbation Theory*, Academic Press, 1990.

[14] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Claredon Press, Oxford, England, 1965.

A Proof of lemma 7.3

Proof of lemma 7.3: We can express

$$q_j = \sum_{i=1}^{n} \alpha_k q'_k$$

where $\alpha_k = q'_k \cdot q_j$ and $\sum \alpha_k^2 = 1$, since the $\{q'_k\}$ form an orthonormal basis and $||q_j|| = 1$. Then since $(Q')^{-1} q'_k = e_k$, we have

$$||(Q'S')^{-1} q_j|| \;=\; ||\sum_{k=1}^{n} \left(\frac{\alpha_k}{s'_k}\right) e_k|| \;=\; \left(\sum_{k=1}^{n} \left(\frac{\alpha_k}{s'_k}\right)^2\right)^{1/2}.$$

Writing simply β for $\beta(n)$, we partition the index set of the last summation into two subsets, $G = \{k : s'_k > s_j/\sqrt{\beta}\}$ and $L = \{k : s'_k \leq s_j/\sqrt{\beta}\}$, and bound each of the subsets separately.

Case 1: $k \in G$, so $s'_k > s_j/\sqrt{\beta}$. Then $\alpha_k/s'_k < \alpha_k\sqrt{\beta}/s_j$ and

$$\left(\sum_{k\in G} \left(\frac{\alpha_k}{s'_k}\right)^2\right)^{1/2} \;<\; \frac{\sqrt{\beta}}{s_j}\left(\sum_{k\in G} \alpha_k^2\right)^{1/2} \leq \frac{\sqrt{\beta}}{s_j}$$

since $\sum_{k=1}^{n} \alpha_k^2 = 1$.

Case 2: $k \in L$. To simplify the exposition slightly, we write for example $r \lesssim s$ to mean $r \leq s(1+\phi)$ where $\phi < 1/q(n)$, for some polynomial $q(n)$.

We have $s'_k \leq s_j/\sqrt{\beta} \leq 1/\sqrt{\epsilon\beta} < 1/\sqrt{\epsilon}$, so $s'_k = \sqrt{||M'||/|\lambda_k|}$. Combining with $s'_k \leq \sqrt{1/(\epsilon\beta)}$ we get

$$|\lambda_k| \geq \epsilon||M||\beta \geq 4n^{3/2}\epsilon\delta(n) \geq 4n^{3/2}||\Delta M||.$$

Since $|\lambda_k - \lambda'_k| < ||\Delta M||$, $\lambda_k \approx \lambda'_k$; similarly using $||M|| \geq |\lambda_k|$

$$s_k \approx s'_k.$$

Using the case hypothesis we obtain

$$\beta \lesssim \left(\frac{s_j}{s_k}\right)^2.$$

From the definition we have

$$\lambda_k = \left(\frac{s_j}{s_k}\right)^2 \max(\epsilon||M||, \lambda_j)$$

from which

$$|\lambda_k - \lambda_j| \approx |\lambda_k|$$

follows. We can now bound α_k/s'_k. From lemma 7.2 we have

$$\begin{aligned}
\frac{\alpha_k}{s'_k} &\leq \frac{4n||\Delta M||}{|\lambda_k - \lambda_j|\; s'_k} \\
&\approx \frac{4n||\Delta M||}{|\lambda_k|\; s_k} \\
&\leq \frac{4n^{3/2}\;||\Delta M||}{(s_j/s_k)^2\epsilon||M||\;(s_k/s_j)s_j\;\sqrt{n}}
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{\beta}{(s_j/s_k)}\frac{1}{s_j\sqrt{n}} \\
&\lesssim \frac{\sqrt{\beta}}{s_j\sqrt{n}}.
\end{aligned}
$$

It is tedious but straightforward to verify that this argument actually implies

$$\frac{\alpha}{s'_k} \leq \frac{2\sqrt{\beta}}{s_j\sqrt{n}}.$$

To finish the proof, we have

$$\left(\sum_{k\in L}\left(\frac{\alpha_k}{s'_k}\right)^2\right)^{1/2} \leq \frac{2\sqrt{\beta}}{s_j}\left(\sum_{k\in L}\frac{1}{n}\right)^{1/2} \leq \frac{2\sqrt{\beta}}{s_j}.$$

□