Applied Mathematics and Computation 219 (2012) 1607–1615

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

The geometric mean algorithm

Rui Ralha

Centro de Matemática, Universidade do Minho, 4710-057 Braga, Portugal

ARTICLE INFO

Keywords:
Eigenvalues
Symmetric matrices
Geometric bisection

ABSTRACT

Bisection (of a real interval) is a well known algorithm to compute eigenvalues of symmetric matrices. Given an initial interval $[a, b]$, convergence to an eigenvalue which has size much smaller than a or b may be made considerably faster if one replaces the usual arithmetic mean (of the end points of the current interval) with the geometric mean. Exploring this idea, we have implemented geometric bisection in a Matlab code. We illustrate the effectiveness of our algorithm in the context of the computation of the eigenvalues of a symmetric tridiagonal matrix which has a very large condition number.

1. Introduction

The numerical computation of eigenvalues of large symmetric matrices is a problem of major importance in many scientific and engineering applications. See, for instance, [15, Chapter X], for an account of the origins of matrix eigenvalue problems. Depending upon the application, one may want the full spectrum or just a few eigenvalues (and possibly also the corresponding eigenvectors).

In many cases, matrices exhibit eigenvalues which have different orders of magnitude, that is, with λ_1 and λ_n the eigenvalues of larger and smaller magnitude, respectively, the condition number $cond(A) = |\lambda_1|/|\lambda_n|$ is very large. The computation of λ_n, which is certainly necessary in finding $cond(A)$, is also required, for instance, in signal processing and estimation. Given the covariance sequence of observed data, it has been proposed in [13] to determine the sinusoidal frequencies from the eigenvector associated to the smallest eigenvalue of the covariance matrix, a symmetric positive definite Toeplitz matrix.

For general symmetric matrices, there is a well known method for slicing the spectrum (see, for instance, [12, p. 46]). With K and M symmetric, let us write the triangular factorization

$$K - \sigma M = L_\sigma \Delta_\sigma L_\sigma^T, \tag{1}$$

where Δ_σ is diagonal and M is positive definite. Then the number of negative eigenvalues of $K - \sigma M$ is equal to the number of negative diagonal entries of Δ_σ. So, for each chosen value σ, the decomposition (1) gives the number of eigenvalues which are to the left of σ and we will denote this number by $count(\sigma)$. For general matrices of order n, this computation is a $O(n^3)$ process. The most popular use of $count(\sigma)$ is for the standard symmetric tridiagonal eigenvalue problem (that is, K is symmetric tridiagonal and M is the identity matrix). This is so because the computation of $count(\sigma)$ requires $O(n)$ floating point operations for tridiagonal matrices and these arise in a similarity transformation (usually with Householder reflections or Givens rotations) or in the context of the Lanczos algorithm.

In the LAPACK routines SSTEBZ and DSTEBZ [2] (for single and double precision, respectively) $count(\sigma)$ is the essential tool to compute some or all of the eigenvalues of a symmetric tridiagonal matrix, with user prescribed accuracy.

For full matrices for which the computation of $count(\sigma)$ is a $O(n^2)$ process, the reduction to tridiagonal form may be avoided. This is the case of symmetric positive definite Toeplitz matrices. For the computation of the smallest eigenvalue

E-mail address: r_ralha@math.uminho.pt

0096-3003/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.08.002

of such matrices, Cybenko and Van Loan [3] presented an algorithm which is a combination of bisection and Newton's method for the secular equation. Others have replaced the Newton's method by different acceleration techniques (see [10] and references therein). In [17,11], bisection has also been used to locate not only the smallest eigenvalue but the complete spectrum. In all the proposed methods, the most expensive part is the computation of a region in which the algorithms monotonically converge to the desired eigenvalue. This is where our proposal plays a role.

Pascal matrices, which have important applications (see [20] and references there), are another kind of structured matrices for which fast algorithms do exist. The Choleski decomposition of such matrices may be computed with only $O(nlog(n))$ flops [19], therefore $count(\sigma)$ is rather inexpensive in this case.

The central issue of this paper is to show the virtues of choosing the geometric mean $\sigma = (a \cdot b)^{1/2}$ rather than the arithmetic mean $\sigma = (a+b)/2$ in sectioning the interval $[a,b]$ which is known to contain the target eigenvalue(s). An initial interval $[a,b]$ containing all eigenvalues is usually computed from the union of the Gerschgorin "discs" (see, for instance, Theorem 2.9 in [7]). For matrices with large condition numbers, this interval will contain eigenvalue(s) of much smaller size than $max\{|a|,|b|\}$.

The use of the geometric mean has been considered in [5, pp. 9–10], in the context of computing the SVD of a dense matrix A with low relative error, in time growing like a low order polynomial in $log_2(log_2(cond(A))$. We stress out that, as compared to what has been done in [5] for geometric bisection, we do present a much more detailed analysis and original material. Of particular interest is the fact that geometric bisection (which can be much better) is never much worst than usual bisection. This is a strong argument in favor of using the geometric mean in codes where the arithmetic mean has been traditionally implemented.

2. The geometric mean

Suppose that $0 < a_0 < b_0$, with a_0 and b_0 of very different orders of magnitude. If we are looking for an eigenvalue λ that lies between a_0 and b_0 but is much closer to a_0, instead of the usual arithmetic mean (AM)

$$m_j = \frac{a_{j-1} + b_{j-1}}{2}, \quad j = 1, 2, \ldots, \tag{2}$$

it is much better, in each iteration, to use the geometric mean (GM)

$$m'_j = \left(a_{j-1} \cdot b_{j-1}\right)^{1/2}, \quad j = 1, 2, \ldots, \tag{3}$$

until the endpoints a_j and b_j have the same size. For instance, if $[a_0, b_0] = [2^{-22}, 2^{20}]$, then (2) and (3) produce $m_1 = 2^{19} + 2^{-23}$ and $m'_1 = 2^{-1}$, respectively, i.e., one single step of (3) produces an interval of much smaller size, speeding up convergence if $\lambda < 2^{-1}$. In fact, 21 iterations with (2) are needed to produce an interval with right hand side close to $m'_1 = 2^{-1}$.

To see that the geometric mean does correspond to the arithmetic mean of the exponents of the endpoints a_{j-1} and b_{j-1} (considering such exponents as floating point numbers), write

$$E(a_{j-1}) = \log_2(a_{j-1}), \quad E(b_{j-1}) = \log_2(b_{j-1})$$

and get

$$m'_j = \left(a_{j-1} \cdot b_{j-1}\right)^{1/2} = 2^{\frac{E(a_{j-1}) + E(b_{j-1})}{2}}.$$

3. Getting bounds of the same magnitude

It is clear that it is more efficient to use the geometric mean rather than the arithmetic mean when the endpoints have different sizes and the target λ is much closer to the left endpoint. At first glance, one may fear that the use of (3) is a bet whose benefit when our guess $\lambda < m'_j$ proves to be correct is completely shaded by the increase in the number of the necessary steps, relatively to the use of (2), when λ is much closer to the right endpoint. The beauty of GM is that this is not so, i.e., the gain in the best case is much bigger than the loss in the worst case. We have the following

Proposition 1. *Let $\lambda \in [a_0, b_0]$ with $0 < 2a_0 < b_0$ and $k = \lceil \log_2 \log_2 (b_0/a_0) \rceil$. Then, independently of the location of λ in $[a_0, b_0]$, after k steps with (3) we get $[a_k, b_k]$ such that*

$$\frac{b_k}{a_k} < 2. \tag{4}$$

Proof. For each $j \geqslant 1$, it is either $[a_j, b_j] = \left[a_{j-1}, (a_{j-1} \cdot b_{j-1})^{1/2}\right]$ or $[a_j, b_j] = \left[(a_{j-1} \cdot b_{j-1})^{1/2}, b_{j-1}\right]$, depending upon the location of λ. In any case, we have

$$\frac{b_j}{a_j} = \left(\frac{b_{j-1}}{a_{j-1}}\right)^{1/2}. \tag{5}$$