The geometric mean algorithm

Rui Ralha

Centro de Matemática
Universidade do Minho
4710-057 Braga, Portugal
email: r_ralha@math.uminho.pt

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.

Key words: Eigenvalues, symmetric matrices, geometric bisection.

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 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] and [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)

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

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

$$m'_j = (a_{j-1} \cdot b_{j-1})^{1/2}, \qquad j = 1, 2, \cdots \tag{3}$$

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

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

and get

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

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

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

Therefore, the condition (4) may be written as

$$\left(\frac{b_0}{a_0} \right)^{1/2^k} < 2 \tag{6}$$

which is equivalent to

$$k > \log_2 \log_2 (b_0/a_0)$$

so that for

$$k' = \lceil \log_2 \log_2 (b_0/a_0) \rceil \tag{7}$$

the condition (4) is true.

More generally, to compute the smallest integer k for which the following condition holds

$$\frac{b_k - a_k}{a_k} < \varepsilon \tag{8}$$

we write

$$\frac{b_k}{a_k} < 1 + \varepsilon \tag{9}$$

and

$$\left(\frac{b_0}{a_0} \right)^{1/2^k} < 1 + \varepsilon \tag{10}$$

to get

$$k > \log_2 \log_2 (b_0/a_0) - \log_2 \log_2 (1+\varepsilon) \tag{11}$$

which shows that the number of iterations required to satisfy a relative error bound is independent of the location of the eigenvalue within the bounds a_0 and b_0.

The same is not true for arithmetic bisection. To satisfy (4), one single step may be enough, that is when λ is in the right half of $[a_0, b_0]$; on the other hand, if the left half of the interval is always chosen in each iteration, then the number of steps necessary for (4) to hold is

$$k = \left\lceil \log_2 \left(\frac{b_0 - a_0}{a_0} \right) \right\rceil \tag{12}$$

which we take to be

$$k = \left\lceil \log_2 \left(\frac{b_0}{a_0} \right) \right\rceil \tag{13}$$

since this exceeds the true value by one unit only when

$$\frac{b_0}{a_0} - 1 \leq 2^{k-1} < \frac{b_0}{a_0} \tag{14}$$

holds. Therefore, we may say that the average number of AM steps to satisfy condition (4) is $k/2$, with k given in (13), and we may write, with k' given in (7),

$$k = 2^{k'-1}.$$

This simple relation expresses the average gain of using GM, as compared to AM, for endpoints of different sizes.

4 When bounds are of the same magnitude

Having produced an interval $[a_k, b_k]$ that satisfies (4), one may switch from GM to AM since further iterations to accomplish the condition (8) are almost the same, independently of the choice of GM or AM. A first approach to this is to write [1]

$$(a_j \cdot b_j)^{1/2} = \frac{a_j + b_j}{2} - \frac{(b_j - a_j)^2}{8a_j} + \cdots$$

from where it is clear that the two means tend to the same value as $b_j - a_j$ approaches zero, but we will analyze this in further detail. First note that, in

[1] This follows from $(a_j \cdot b_j)^{1/2} = a_j \left(1 + \frac{b_j - a_j}{a_j}\right)^{1/2}$ and the expansion $(1+\delta)^{1/2} = 1 + \frac{1}{2}\delta - \frac{1}{8}\delta^2 + \cdots$

the pursuit of (4), there are two extreme cases for AM: the best case corresponds to the situation in which the right hand half of the interval is always chosen: from $[a_j, b_j]$ to $[a_{j+1}, b_{j+1}]$, with $b_{j+1} = b_j$ and $a_{j+1} = m_j$, the ratio

$$\frac{b_j - a_j}{a_j}$$

decreases to less that half, since we have

$$\frac{b_{j+1} - a_{j+1}}{a_{j+1}} = \frac{b_j - a_j}{a_j + b_j} < \frac{b_j - a_j}{2a_j}.$$

The worst case for AM does correspond to the situation in which the left hand half of the interval is always chosen since with $a_{j+1} = a_j$ and $b_{j+1} = m_j$ we get

$$\frac{b_{j+1} - a_{j+1}}{a_{j+1}} = \frac{b_j - a_j}{2a_j}.$$

In each one of these extreme cases, the GM iterates follow closely the AM iterates according to an "interlacing property". We have the following

Proposition 2 *Let a_k and b_k be such that $0 < a_k < b_k < 2a_k$ and, for $j \geq 1$, $a_{k+j} = a_k$ and $b_{k+j} = m_{k+j}$. Then, for each $j = 1, 2, \cdots$, we have*

$$m_{k+j+1} < m'_{k+j} < m_{k+j}. \tag{15}$$

PROOF. The condition we wish to prove is

$$a_k + \frac{b_k - a_k}{2^{j+1}} < a_k \left(\frac{b_k}{a_k}\right)^{1/2^j} < a_k + \frac{b_k - a_k}{2^j}. \tag{16}$$

With

$$x = \frac{b_k - a_k}{a_k} = \frac{b_k}{a_k} - 1 < 1$$

we have

$$\left(\frac{b_k}{a_k}\right)^{1/2^j} = (1+x)^{1/2^j} = 1 + \frac{1}{2^j}x - \frac{2^j - 1}{2^{2j+1}}x^2 + O(x^3)$$

and, taking into account that $a_k x = b_k - a_k$, we get

$$\begin{aligned} m'_{k+j} &= a_k + \frac{b_k - a_k}{2^j} - \frac{2^j - 1}{2^{2j+1}} a_k x^2 + a_k \cdot O(x^3) \\ &= m_{k+j} - R. \end{aligned}$$

Since the series is alternate (x is positive), we may write

$$0 < R < \frac{2^j - 1}{2^{2j+1}} a_k x^2 \tag{17}$$

and conclude immediately that $m'_{k+j} < m_{k+j}$. It is easy to verify that for $m_{k+j+1} < m_{k+j} - R$ to hold, it must be

$$R < \frac{b_k - a_k}{2^{j+1}} \tag{18}$$

and we now have the following equivalencies

$$\begin{aligned} \frac{2^j - 1}{2^{2j+1}} a_k x^2 &< \frac{b_k - a_k}{2^{j+1}} \\ \frac{2^j - 1}{2^{2j+1}} x^2 &< \frac{b_k - a_k}{2^{j+1} a_k} \\ \frac{2^j - 1}{2^j} x^2 &< x, \end{aligned}$$

the last condition being clearly true since $0 < x < 1$.

For the other extreme case we have the following

Proposition 3 *Let a_k and b_k be such that $0 < a_k < b_k < 2a_k$ and, for $j \geq 1$, $a_{k+j} = m_{k+j}$ and $b_{k+j} = b_k$. Then, for each $j = 1, 2, \cdots$, we have*

$$m_{k+j-1} < m'_{k+j} < m_{k+j}. \tag{19}$$

PROOF. Since, in this case, it is

$$m_{k+j} = b_k - \frac{b_k - a_k}{2^j}, \tag{20}$$

$$m'_{k+j} = b_k \left(a_k / b_k\right)^{1/2^j}, \tag{21}$$

we need to prove the following

$$b_k - \frac{b_k - a_k}{2^{j-1}} < b_k \left(\frac{a_k}{b_k}\right)^{1/2^j} < b_k - \frac{b_k - a_k}{2^j}. \tag{22}$$

With

$$x = 1 - \frac{a_k}{b_k} < \frac{1}{2}$$

we have

$$\left(\frac{a_k}{b_k}\right)^{1/2^j} = (1 - x)^{1/2^j} = 1 - \frac{1}{2^j} x - \frac{2^j - 1}{2^{2j+1}} x^2 + \cdots \tag{23}$$

and

$$b_k \left(\frac{a_k}{b_k}\right)^{1/2^j} = b_k - \frac{b_k - a_k}{2^j} - \frac{2^j - 1}{2^{2j+1}} b_k x^2 - \cdots \tag{24}$$

$$= m_{k+j} - R. \tag{25}$$

This time, R is the sum of a series of positive terms and can not be bounded in the same way as before. Straightforward calculations show that, with $f(x) = (1-x)^{1/2^j}$, we have

$$\frac{f^{(i+1)}(0)}{f^{(i)}(0)} = i - 2^{-j}$$

for each $i \geq 2$. Therefore,

$$R = \frac{2^j - 1}{2^{2j+1}} b_k x^2 + .. \cdots + \frac{f^{(i)}(0)}{i!} b_k x^i + \left(\frac{f^{(i)}(0)}{i!} b_k x^i \right) \frac{i - 2^{-j}}{i+1} x + \cdots$$

is bounded by the sum of a geometric series of ratio x, i.e., we have

$$R < \frac{\frac{2^j-1}{2^{2j+1}} b_k x^2}{1-x}.$$

It is easy to verify that for $m_{k+j-1} < m_{k+j} - R$ to hold, it must be

$$R < \frac{b_k - a_k}{2^j} \tag{26}$$

and we now have the following equivalencies

$$\begin{aligned} \frac{\frac{2^j-1}{2^{2j+1}} b_k x^2}{1-x} &< \frac{b_k - a_k}{2^j} \\ \frac{\frac{2^j-1}{2^{j+1}} x^2}{1-x} &< \frac{b_k - a_k}{b_k} \\ \frac{2^j - 1}{2^{j+1}} x &< 1 - x \end{aligned}$$

the last condition being true since $0 < x < \frac{1}{2}$.

We now use the previous results to prove the following

Proposition 4 *Let a_k and b_k be such that $0 < a_k < b_k < 2a_k$ and $\lambda \in [a_k, b_k]$. For any $\varepsilon \in]0,1[$, the number of AM steps and the number of GM steps required to locate λ in an interval $[a_{k+j}, b_{k+j}]$ such that*

$$\frac{b_{k+j} - a_{k+j}}{a_{k+j}} < \varepsilon$$

can not differ by more than one.

PROOF. The number of GM steps is independent of the location of λ. For the cases considered in Proposition 2 and Proposition 3, the conclusion follows

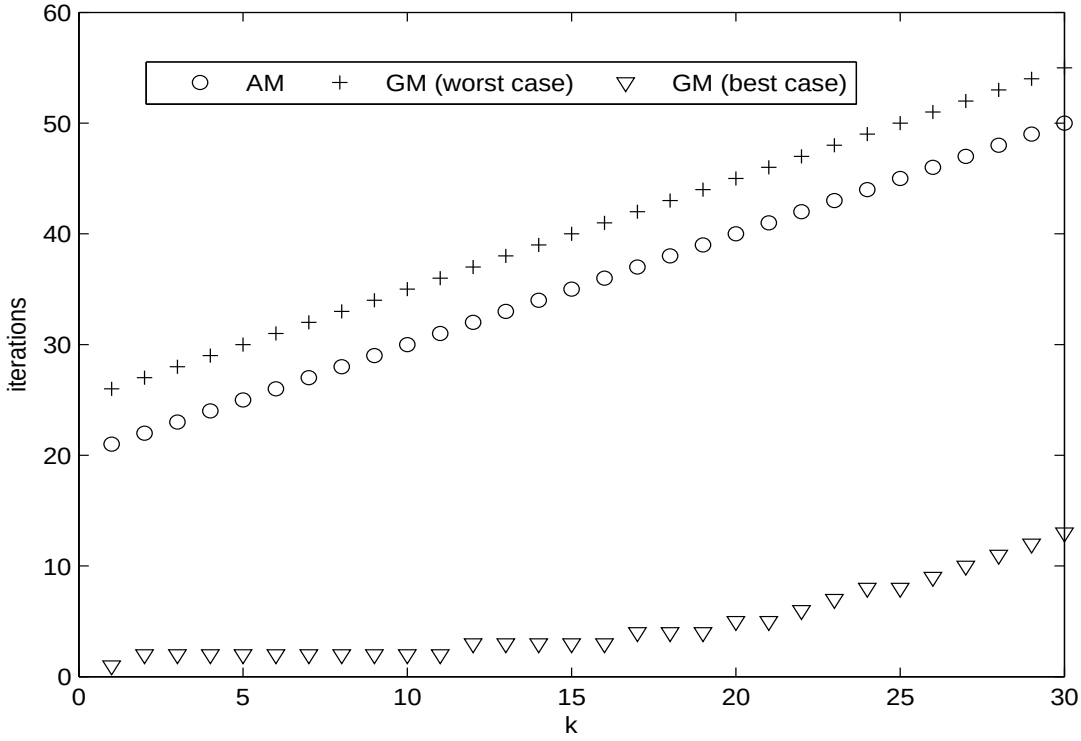

Fig. 1. Iterations of AM and GM to get $b_k - a_k < 2^{-k}$.

from the respective "interlacing property". Since these are extreme cases for AM, the result is true independently of the location of λ.

Of course, we do not need to aim at high relative accuracy to ripe the benefits of using the geometric mean. To illustrate this point, in Fig. 1 we plot the number of iterations required to satisfy the stopping criteria $b_k - a_k < \varepsilon$, with $\varepsilon = 2^{-k}$ for $k = 1, \cdots, 30$ and $[a_0, b_0] = [2^{-22}, 2^{20}]$. The straight line for AM does correspond to the number $\left\lceil \log_2 \left(\frac{b_0 - a_0}{\varepsilon} \right) \right\rceil = k + \lceil \log_2 (b_0 - a_0) \rceil$; as compared to this, in the worst case, GM takes an extra $\left\lceil \log_2 \log_2 \left(\frac{2^{20}}{2^{-22}} \right) \right\rceil = 6$ iterations. In the best case, the gain of GM over AM increases as $\varepsilon = 2^{-k}$ decreases until ε is of the size of $a_0 = 2^{-22}$. For $k > 22$ (i.e., for $\varepsilon < a_0$), the logarithmic curve gives place to another straight line, parallel to the other two lines.

5 The harmonic mean

So far, we have considered only two of the three Pythagorean means. We now consider also the harmonic mean $H(a, b)$ of two positive numbers a and b,

which is

$$H(a,b) = \frac{2}{\frac{1}{a} + \frac{1}{b}}. \tag{27}$$

It is natural to ask whether the harmonic mean has some interest in the context of our problem which is that of locating λ inside $[a_0, b_0]$ with the endpoints of different size. The answer is quite simple: if λ is very close to a_0 then the harmonic mean is better than the other two means. In fact, we have

$$H(a_0, b_0) = \frac{2a_0 b_0}{a_0 + b_0} < 2a_0$$

and see that if $\lambda < H(a_0, b_0)$ then one single step is enough to produce an interval $[a_1, b_1]$ such that $\frac{b_1}{a_1} < 2$. In comparison, as seen before, GM and AM take a number of steps which is equal to $\lceil \log_2 \log_2 (b_0/a_0) \rceil$ and $\lceil \log_2 (b_0/a_0) \rceil$, respectively, to produce intervals whose endpoints satisfy the same condition.

From (27) we see that the use of the harmonic mean with $[a_j, b_j]$ to find λ is equivalent to use the arithmetic mean with $[\alpha_j, \beta_j] = [1/b_j, 1/a_j]$ to find $1/\lambda$. From section 3, we know that the number of AM steps required to satisfy $\frac{\beta_k}{\alpha_k} < 2$ varies between 1 and

$$k = \left\lceil \log_2 \left(\frac{\beta_0}{\alpha_0} \right) \right\rceil = \left\lceil \log_2 \left(\frac{1/a_0}{1/b_0} \right) \right\rceil = \left\lceil \log_2 \left(\frac{b_0}{a_0} \right) \right\rceil \tag{28}$$

and so the same is true for the harmonic mean. From this we conclude that of the three Pythagorean means, the geometric mean is the winner in minimizing the average number of steps necessary to locate λ.

6 Implementation of the geometric mean

So far, we have restricted our analysis to the case of a_0 and b_0 being positive. When they are both negative, we simply take

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

and our previous analysis on the number of iterations required does apply to this situation. If $a_0 \neq 0$ and $b_0 \neq 0$ have different signs, then we take $m'_1 = 0$. If $a_0 = 0$, we replace it with $realmin$, the smallest normalized number, which is equal to 2^{-1022} in the IEEE double precision format (this ensures high relative accuracy for the eigenvalues which are well defined by the matrix entries and larger than $realmin$). If $a_1 = 0$ (note that when $a_0 b_0 < 0$ it is either $a_1 = 0$ or $b_1 = 0$) we also replace it $realmin$. When $b_0 = 0$ or $b_1 = 0$ we replace it with

$-realmin$. We summarize this in the following table.

CASE	ITERATE
$0 < a_0 < b_0$	$m_1' = G(a_0, b_0)$
$a_0 < b_0 < 0$	$m_1' = -G(a_0, b_0)$
$a_0 < 0 < b_0$	$m_1' = 0$
$a_j = 0$ ($j = 0$ or $j = 1$)	$m_{j+1}' = G(realmin, b_j)$
$b_j = 0$ ($j = 0$ or $j = 1$)	$m_{j+1}' = -G(a_j, -realmin)$

Next, we illustrate the use of the algorithm.

Example 5 *The matrix*

$$T = \begin{bmatrix} 1 & 0.15 \cdot 10^{-16} & 0 \\ 0.15 \cdot 10^{-16} & 10^{-32} & 0.15 \cdot 10^{-16} \\ 0 & 0.15 \cdot 10^{-16} & 1 \end{bmatrix}$$

has eigenvalues $\lambda_1 = 0.955 \cdot 10^{-32}$ *and* $\lambda_2 = \lambda_3 = 1$*, to 16 decimal digits of accuracy (see [4] and [14]). The Gerschgorin "discs" for* T *are*

$$\mathcal{D}_1 = \mathcal{D}_3 = \left[1 - 0.15 \cdot 10^{-16}, 1 + 0.15 \cdot 10^{-16}\right] \tag{29}$$

and

$$\mathcal{D}_2 = \left[10^{-32} - 0.3 \cdot 10^{-16}, 10^{-32} + 0.3 \cdot 10^{-16}\right]. \tag{30}$$

Since $\mathcal{D}_2$ *does not intersect the other intervals, we immediately conclude that it contains the smallest eigenvalue and also that* $\lambda_2 = \lambda_3 = 1$ *to 16 digits. However, our code does not see this and takes the initial interval to be* $[a_0, b_0] = [10^{-32} - 0.3 \cdot 10^{-16}, 1 + 0.15 \cdot 10^{-16}]$*. This is a good example to compare GM with AM since the eigenvalues are close to both ends of* $[a_0, b_0]$*. The algorithm proceeds with* $m_1' = 0$ *and produces* $a_1 = 0$ *by finding that all the pivots for* T *are positive, that is,* $count(0) = 0$*. For the next iterate,* a_1 *will be replaced by* 2^{-1022}*, that is*

$$m_2' = \left(2^{-1022} \cdot b_0\right)^{1/2}$$

and then takes 10 more iterations to produce an interval $[a_{11}, b_{11}]$ *that contains* λ_1 *and satisfies* $b_{11} < 2a_{11}$*. Compared to this, the usual bisection algorithm takes 107 steps to satisfy the same criterion. For* λ_2*, AM is only a little faster: the right half of* $[a_0, b_0]$ *satisfies the condition* (4)*, so no further bisection step is required to this end; GM takes an extra 6 iterations for this. From this point, both AM and GM pay one step for each bit of accuracy. In the following table we display the number of iterates necessary to satisfy the condition* (8)*, for*

some relative tolerances ε (since the eigenvalues are well defined by the matrix entries, the accuracy increases as we diminish ε):

ε	$AM(\lambda_1, \lambda_2)$	$GM(\lambda_1, \lambda_2)$
1	*(107,0)*	*(11,6)*
2^{-10}	*(117,10)*	*(21,16)*
2^{-50}	*(157,50)*	*(61,56)*

We see that the number of steps to locate all the eigenvalues of the matrix is significantly smaller in GM than it is in AM.

7 Geometric multi-section

The parallel computation of eigenvalues continues to be an active area of research. See, for instance, [1] and references therein. The bisection algorithm is adequate for parallel processing since independent tasks are created as soon as one gets several intervals containing different eigenvalues of a given matrix. For parallel processing, care must be taken to ensure the correctness of the results. The logic of the bisection algorithm depends on $count(\sigma)$ being a monotonic increasing function of σ. However, depending upon the features of the arithmetic, monotonicity can fail and incorrect eigenvalues may be computed, because of rounding or as a result of using networks of heterogeneous parallel processors (see [6]). A different source of parallelism is the use of multi-section (see [8], [9] and [16]). Multi-section consists upon inspecting simultaneously the $p-1$ points

$$a + k \cdot \frac{b-a}{p}, \text{ for } k = 1, \cdots, p-1. \tag{31}$$

Multi-section of $[a, b]$ can be an effective way for producing several intervals containing eigenvalues. However, it is not an efficient algorithm if $[a, b]$ contains only a single eigenvalue. This is because the simultaneous calculation at $p-1$ points reduces the number of bisection steps by a factor equal to $\lceil \log_2 p \rceil$ as it follows from

$$\frac{b_0 - a_0}{p^k} < \varepsilon.$$

So, the speedup of a parallel multi-section code is bounded by $\lceil \log_2 p \rceil$ and for the efficiency E we have

$$E \leq \frac{\lceil \log_2 p \rceil}{p-1}$$

which tends rapidly to zero. Nevertheless, if many processors are readily available, then, in despite of our theoretical considerations, they may be used to

produce non-negligible speedup. See [18] for results of an implementation of multi-section on GPUs (Graphics Processing Units).

Again, for end points a and b of very different sizes, the use of geometric multi-section plays a role. With $0 < a < b$, it consists upon dividing the interval $[a, b]$ in p sections through the points

$$M'_k = a \left(\frac{b}{a}\right)^{k/p} \text{ for } k = 1, \cdots, p-1. \tag{32}$$

Again, with $E(a) = \log_2 a$ and $E(b) = \log_2 b$, we get

$$M'_k = 2^{E(a)} \left(\frac{2^{E(b)}}{2^{E(a)}}\right)^{k/p} \tag{33}$$

$$= 2^{E(a)+k\cdot\frac{E(b)-E(a)}{p}}, \text{ for } k = 1, \cdots, \ p-1 \tag{34}$$

which shows that formula (32) does multi-section of the interval of the exponents $E(a)$ and $E(b)$. As it is the case with bisection, the number of arithmetic multi-section steps may be significantly reduced if geometric multi-section is used instead.

8 Conclusions

Usual bisection codes use the arithmetic mean of the end points of an interval which is known to contain a target eigenvalue λ. We have shown that there are cases for which it is much better to use the geometric mean instead, as this may reduce significantly the number of required steps. This will be the case when the interval $[a, b]$ is such that $0 < a \ll b$ and λ is much closer to a than it is to b (similarly, when $a \ll b < 0$ and λ is much closer to b). The interesting point about geometric bisection is that, although it can be much better than usual bisection, depending upon the location of λ, it is never much worst. This fact is a strong argument in favor of using the geometric mean in codes where the arithmetic mean is traditionally implemented. We have illustrated the advantages of geometric bisection with the computation of the eigenvalues of a tridiagonal matrix. However, its use may be even more interesting for matrices where the cost of one iteration is more expensive than it is in the tridiagonal case. This is the case of symmetric positive definite Toeplitz matrices, for which the computation of the smallest eigenvalue has important applications. For parallel processing, geometric multi-section may also be a useful alternative to usual multi-section.

Acknowledgements

This research was financed by FEDER Funds through "Programa Operacional Factores de Competitividade - COMPETE" and by Portuguese Funds through FCT - "Fundação para a Ciência e a Tecnologia", within the Project PEst-C/MAT/UI0013/2011.

References

[1] J. Aliaga, P. Bientinesi, D. Dadidović, E. Di Napoli, F. Igual, E. Quintana-Orti, Solving dense generalized eignproblems on multi-threaded architectures, Appl. Math. Comput., 218(2012), pp.11279-11289.

[2] E. Anderson et al, LAPACK Users' Guide, SIAM, 1999.

[3] G. Cybenko and C. F. Van Loan, Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix, SIAM J. Sci. Stat. Comput. 7 (1986), 123-131.

[4] J. Demmel, The inherent inaccuracy of implicit tridiagonal QR, LAPACK Working Note 15, 1992.

[5] J. Demmel and P. Koev, Necessary and Sufficient Conditions for Accurate and Efficient Singular Value Decompositions of Structured Matrices, in Computer Science and Engineering, Vol. II, Amer. Math. Soc. 2001, pp.117-145.

[6] J. Demmel, I. Dhillon and H. Ren, On the correctness of some bisection-like parallel eigenvalue algorithms in floating point arithmetic, Electronic Trans. on Numerical Analysis, 3(1995), pp.116-149.

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

[8] T. Katagiri, C. Vomel and J. Demmel, Automatic performance tuning for the multi-section with multiple eigenvalues method for the symmetric eigenproblem, in PARA'06, Umea, Sweden, 2006.

[9] S. Lo, B. Philippe and A. Sameh, A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem, SIAM J. Sci. Stat. Comp., vol. 8, 2 (1987), pp. s155-s165.

[10] N. Mastronardi, M. Van Barel and R. Vandebril, A Schur-based algorithm for computing bounds to the smallest eigenvalue of a symmetric positive definite Toeplitz matrix, Linear Algebra and its Applications 428 (2008), pp.479-491.

[11] F. Noor and S. Morgera, Recursive and itrative algorithms for computing eigenvalues of Hermitian Toeplitz matrices, IEEE Trans. Signal Processing 41 (1993), pp.1271-1280.

[12] B. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, 1980.

[13] V. Pisarenko, The retrieval of harmonics from a covariance function, Geophysical J. Royal Astronomical Soc., 33 (1973), pp. 347-366.

[14] R. Ralha, Perturbation splitting for more accurate eigenvalues, SIAM J. Matrix Anal. Appl., 31(2009), pp. 75-91.

[15] Y. Saad, Numerical Methods for Large Eigenvalue Problems, 2nd ed., SIAM, 2011.

[16] H. Simon, Bisection is not optimal on vector processors, SIAM J. Sci. Stat. Comp., vol.10, 1(1989), pp. 205-209.

[17] W. F. Trench, Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices, SIAM J. Matrix Anal. Appl., 10(1989), pp. 135-146.

[18] V. Volkov and J. Demmel, Using GPUs to accelerate the bisection algorithm for finding eigenvalues of symmetric tridiagonal matrices, LAPACK Working Note 197, 2008.

[19] X. Wang and Z. Jituan, A fast eigenvalue algorithm for Pascal matrices, Appl. Math. Comput., 183(2006), pp.711-716.

[20] X. Wang and L. Lu, A fast algorithm for solving linear systems of the Pascal type, Appl. Math. Comput., 175(2006), pp.441-451.