Journal of Global Optimization (2019) 73:467–483
https://doi.org/10.1007/s10898-018-0718-y

# Tighter $\alpha$BB relaxations through a refinement scheme for the scaled Gerschgorin theorem

**Dimitrios Nerantzis**[1] · **Claire S. Adjiman**[1]

Received: 3 January 2018 / Accepted: 27 October 2018 / Published online: 11 January 2019


**Abstract**
Of central importance to the $\alpha$BB algorithm is the calculation of the $\alpha$ values that guarantee the convexity of the underestimator. Improvement (reduction) of these values can result in tighter underestimators and thus increase the performance of the algorithm. For instance, it was shown by Wechsung et al. (J Glob Optim 58(3):429–438, 2014) that the emergence of the cluster effect can depend on the magnitude of the $\alpha$ values. Motivated by this, we present a refinement method that can improve (reduce) the magnitude of $\alpha$ values given by the scaled Gerschgorin method and thus create tighter convex underestimators for the $\alpha$BB algorithm. We apply the new method and compare it with the scaled Gerschgorin on randomly generated interval symmetric matrices as well as interval Hessians taken from test functions. As a measure of comparison, we use the maximal separation distance between the original function and the underestimator. Based on the results obtained, we conclude that the proposed refinement method can significantly reduce the maximal separation distance when compared to the scaled Gerschgorin method. This approach therefore has the potential to improve the performance of the $\alpha$BB algorithm.

**Keywords** Global optimization · Branch-and-bound · Convex underestimators · Interval matrix

## 1 Introduction

The $\alpha$BB algorithm [1,2,5,15] is a branch-and-bound algorithm which is based on creating convex underestimators for general twice-continuously differentiable ($C^2$) functions. The tightness of the underestimator plays a key role in the efficiency of the algorithm. In the $\alpha$BB method, the underestimator of a $C^2$ term or function is obtained by adding an appropriate quadratic term to the original expression. The validity of the underestimator depends on the calculation of the so-called $\alpha$ values, which must be chosen appropriately in order to ensure

✉ Claire S. Adjiman
c.adjiman@imperial.ac.uk

Dimitrios Nerantzis
dimitrios.nerantzis10@imperial.ac.uk

[1] Department of Chemical Engineering, Imperial College London, London, UK

convexity. One must take care, however, not to be over-conservative by selecting $\alpha$ values that are larger than needed as smaller the $\alpha$ values lead to a tighter underestimator with respect to the original function.

A number of methods for the calculation of $\alpha$ values that are rigorously valid, i.e., such that the underestimator is guaranteed to be convex, have been presented in the literature [2,12,19,20]. It is usual, but not necessary, for a trade-off between tightness of the underestimator and computational cost to exist. With respect to this, a comparative study among different methods for calculating $\alpha$ values for the original $\alpha$BB underestimator as well as methods that employ different underestimators [2–4,14,16,20] has been presented by Guzman et al. [9].

One important aspect of the choice of $\alpha$ values is with respect to the so-called cluster problem [8]. The cluster problem describes the situation where a branch-and-bound algorithm creates a large number of unfathomed boxes around a solution because it creates nodes much faster than it fathoms. This effect is of course dependent on the quality of the underestimator and can significantly impact the performance of the algorithm. As shown by Wechsung et al. [21], improving the $\alpha$ values can be critical with respect to the cluster effect during execution of the $\alpha$BB algorithm.

Motivated by the above observations, we introduce a "refinement" algorithm, based on Haynsworth's theorem [6,11], to improve the $\alpha$ values given by the scaled Gerschgorin method [2]. Although the algorithm can be applied to improve the $\alpha$ values given by any of the methods used in the original $\alpha$BB method (see [2]) we choose the scaled Gerschgorin method because it usually gives good (i.e., comparatively small) $\alpha$ values, it is computationally cheap and the use of a different $\alpha$ value for each variable (non-uniform shift) allows for more flexibility than other (uniform shift) methods.

In order to test the algorithm on a number of randomly generated (symmetric) interval matrices and interval Hessians generated from test functions, we use the maximum separation distance as a measure of tightness between an $\alpha$BB underestimator and the original function.

In Sect. 2 we begin by briefly presenting the $\alpha$BB underestimator for general $C^2$ functions and the scaled Gerschgorin method for calculating $\alpha$ values for the underestimator. In Sect. 3 we state Haynsworth's theorem which is the basis of our new method. In Sect. 4 we present the refinement algorithm. We begin with an example to help the reader understand how we use Haynsworth's theorem for our purpose. We then give a pseudocode form of the algorithm and close the section with another example where we apply the refinement algorithm. In Sects. 5 and 6 we present the results of comparing the scaled Gerschgorin method and the refinement method. In Sect. 5 we present results from randomly generated symmetric interval matrices while in Sect. 6 we report results from Hessian matrices taken from test functions. Finally, we conclude in Sect. 7.

In what follows we will assume that the reader has some basic knowledge of the $\alpha$BB algorithm and is familiar with interval arithmetic and interval matrices (see [10] for an introduction to interval analysis). We will denote single intervals using lower case letters inside square brackets, for example $[x] = [\underline{x}, \overline{x}]$ and interval matrices with capital letters inside square brackets, for example $[M]$.

## 2 The $\alpha$BB underestimator and the scaled Gerschgorin method

Given a general nonlinear function, $f \in C^2$, a convex underestimator, $F(x) = f(x) + q(x)$, of $f$ over a given hyper-rectangular domain $X = \left[[\underline{x_1}, \overline{x_1}] \cdots [\underline{x_n}, \overline{x_n}]\right]^T$ is constructed within the $\alpha$BB algorithm [1,2,5,15], where

$$q(x) = \sum_{i=1}^{n} \alpha_i (\underline{x_i} - x_i)(\overline{x_i} - x_i),\ \alpha_i \geq 0,\ i = 1, \ldots, n. \tag{1}$$

Note that $q(x) \leq 0, \forall x \in X$ and thus $F(x)$ is indeed an underestimator of $f(x)$ over that domain. The $\alpha$ values have to be determined so as to ensure $F$ is convex. This is accomplished with the use of the interval Hessian matrix, $[H_f]$, over the hyper-rectangular domain of interest. The interval Hessian matrix $[H_f]$ is obtained by constructing the matrix $H_f(x)$ of second-order derivatives of $f$ and deriving an interval enclosure $\left[\underline{h_{ij}}, \overline{h_{ij}}\right]$ for each element $h_{ij}(x)$ over the domain $X$. In the scaled Gerschgorin method [2], the $\alpha$ values are calculated as

$$\alpha_i = \max\left\{0, -\frac{1}{2}\left(\underline{h_{ii}} - \sum_{j \neq i} \max\{|\underline{h_{ij}}|, |\overline{h_{ij}}|\}\frac{k_j}{k_i}\right)\right\},\ i = 1, \ldots, n \tag{2}$$

with $k_i, i = 1, \ldots, n$, being positive integers. A useful feature of the $\alpha$BB underestimator is that the maximum separation distance between $f(x)$ and the underestimator $F(x)$ over $X$ is explicitly given by

$$\max_{x \in X} D(x) = \max_{x \in X}(f(x) - F(x)) = \sum_{i=1}^{n} \alpha_i \frac{(\overline{x_i} - \underline{x_i})^2}{4}. \tag{3}$$

We can see from Eq. (3) that even if the $\alpha$ values were not to improve as we subdivide the domain, the maximum separation distance would nevertheless improve quadratically. This is an important feature of the $\alpha$BB underestimator which relates to the cluster problem [8].

A theoretical analysis of the cluster problem was first carried out in [8]. This analysis showed that the relaxations in a branch-and-bound algorithm must have at least second-order convergence to "avoid" the cluster problem. In a later paper [21], it was shown that the pre-factor of the convergence order also plays a crucial role. For the $\alpha$BB algorithm, the pre-factor corresponds to the $\alpha$ values. Therefore, an improvement of these values could have a significant effect on the performance of the $\alpha$BB algorithm.

As is evident from Eq. (3) we would like to make the $\alpha$ values as small as possible while ensuring that the Hessian matrix of second-order derivatives of $F(x)$, $H_F(x) = H_f(x) + D$ where $D$ is the diagonal matrix with diagonal entries $d_i = 2\alpha_i$, is positive semi-definite over the area of interest. With the help of Haynsworth's theorem [6,11], introduced in the next section, we can improve (reduce) the $\alpha$ values obtained by the scaled Gerschgorin method.

## 3 Haynsworth's theorem

The inertia of a symmetric matrix is defined as follows:

**Definition 3.1** (*Inertia of a symmetric scalar matrix*) Given a symmetric matrix $M$, the inertia of $M$, $In(M)$, is the triplet $(\pi(M), \nu(M), \delta(M))$ of the numbers of positive, negative and zero eigenvalues of $M$, respectively.

We now state Haynsworth's theorem which is the basis of the refinement method.

**Theorem 3.2** (Haynsworth [11]) *Given a symmetric matrix $M$ partitioned in the form $M = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$, and assuming $A$ is non-singular, then $In(M) = In(A) + In(C - B^T A^{-1} B)$.*

Theorem 3.2 can be used recursively for the complete calculation of the inertia of a scalar matrix [7] and therefore for revealing whether the matrix is positive semi-definite or not. This can be accomplished by choosing $A$ to be a single diagonal entry, noting its sign, then calculating the Schur complement, $C - B^T A^{-1} B$ and repeating the process on this newly formed matrix. Assume for example that for a given $n \times n$ symmetric matrix $M$, we repeat this procedure $n$ times and find

$$In(M) = In(A_1) + \cdots + In(A_{n-1}) + In(A_n), \tag{4}$$

with $A_i > 0$ for $i = 1, \ldots, n-1$ and $A_n \geq 0$ where $A_i$ is the entry $m_{11}^{(i)}$ of the $i$-th Schur complement, $M_i$, with $M_1$ being the initial matrix $M$. Then by Theorem 3.2 we conclude that the matrix $M$ is positive semi-definite.

For scalar matrices we can always proceed to calculate the complete inertia even if at some step there is no non-zero diagonal entry that can be chosen (see [7] for details). An extension of the recursive use of Theorem 3.2 for the calculation of the inertia of symmetric interval matrices has been presented in [18]. In this work, however, we are not interested in calculating the inertia but rather guaranteeing semi-definiteness. In a similar manner, we can extend the recursive procedure for determining the positive semi-definiteness of scalar matrices to the case of interval matrices.

For example consider a symmetric interval matrix $[M]$. We follow the same procedure as in the scalar case but use interval arithmetic for the calculation of each subsequent (interval) Schur complement. Assume we find

$$In([M]) = In([A_1]) + \cdots + In([A_{n-1}]) + In([A_n]), \tag{5}$$

with $[A_i] = \left[\underline{m_{11}^{(i)}}, \overline{m_{11}^{(i)}}\right]$ being strictly positive intervals for $i = 1, \ldots, n-1$ and $\underline{m_{11}^{(n)}} \geq 0$.

It is straightforward to show, based on Theroem 3.2, that this implies the positive semi-definiteness of the interval matrix $[M]$ (i.e. all the symmetric scalar matrices contained in $[M]$ are positive semi-definite). Nevertheless, for the purpose of completeness, we state this in Proposition 3.3, followed by the proof.

**Proposition 3.3** *Given a symmetric interval matrix $[M]$, if $[A_i]$ are strictly positive intervals for $i = 1, \ldots, n-1$ and $[A_n]$ is a non-negative interval, then $[M]$ is positive semi-definite.*

***Proof*** First, we note that when we calculate an interval Schur complement, $[C] - [B]^T [A]^{-1} [B]$, an overestimation takes place. That is, $[C] - [B]^T [A]^{-1} [B] \supseteq \{C - B^T A^{-1} B : C \in [C], B \in [B] \text{ and } A \in [A]\}$. It is therefore clear that, for any symmetric matrix $M \in [M]$ we have $A_i \in [A_i]$, for $i = 1, \ldots, n$ and thus $M$ is positive semi-definite and therefore $[M]$ is positive semi-definite. □

In the next section we begin with an example of how this can be used to calculate smaller $\alpha$ values and to help the reader understand how we utilize Theorem 3.2.

## 4 The refinement algorithm

Consider a 3-dimensional function $f : B \subset \mathbb{R}^3$. We want to construct the $\alpha$BB underestimator over an area $X \subseteq B$. After calculation of the interval Hessian $[H_f]$ over $X$ and calculation of the $\alpha$ values using Eq. (2) we consider the convex underestimator $F(x) = f(x) + q(x)$ with the corresponding Hessian matrix,

$$[H_F] = \begin{bmatrix} [h'_{11}] & [h_{12}] & [h_{13}] \\ [h_{21}] & [h'_{22}] & [h_{23}] \\ [h_{31}] & [h_{32}] & [h'_{33}] \end{bmatrix}, \tag{6}$$

where $[h_{ij}] = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}|_{x=X}\right]$ is calculated using interval arithmetic and $[h'_{ii}] = [h_{ii}] + d_i$ where $d_i = 2\alpha_i$ with $\alpha_i$ calculated using Eq. (2) for $i = 1, 2, 3$. Now assume that after applying Haynsworth's theorem recursively on $[H_F]$ we find:

at step 1,

$$\left[\underline{m_{11}^{(1)}}, \overline{m_{11}^{(1)}}\right] = [h'_{11}] > 0, \tag{7}$$

at step 2,

$$\left[\underline{m_{11}^{(2)}}, \overline{m_{11}^{(2)}}\right] = [h'_{22}] - \frac{[h_{12}]^2}{[h'_{11}]} > 0 \tag{8}$$

and finally at step 3,

$$\left[\underline{m_{11}^{(3)}}, \overline{m_{11}^{(3)}}\right] = [h'_{33}] - \frac{[h_{13}]^2}{[h'_{11}]} - \frac{\left([h_{23}] - \frac{[h_{12}]}{[h_{23}]}\right)^2}{\left([h'_{22}] - \frac{[h_{12}]^2}{[h'_{11}]}\right)} \geq 0. \tag{9}$$

Notice that the left-hand sides of inequalities (7), (8), (9) are the (interval) entries $[m_{11}^{(i)}]$, $i = 1, 2, 3$ of each subsequent (interval) Schur complement starting with matrix $[H_F]$. In the above scenario, based on Proposistion 3.3, we would know that the interval matrix is positive semi-definite.

We focus on one specific output of these calculations, $\underline{m_{11}^{(3)}}$. In the case shown here, since the original matrix $[H_F]$ is used without any pivoting, $\underline{m_{11}^{(3)}}$ is linked to the third variable $x_3$ and is therefore denoted by $r_3$. Notice that if $r_3$ is positive we can reduce $[h'_{33}]$ (i.e. reduce $d_3$) by any value in $[0, \min\{r_3, d_3\}]$ without affecting inequalities (7) and (8), thus maintaining the positive semi-definiteness of the interval Hessian $[H_F]$. We refer to the value $r_3$ (or $r_i$ in the general case) as the residual. After we reduce $[h'_{33}]$ by a certain value we can interchange the second and third rows and columns of the new matrix and repeat the same process. Since the last row and column of the matrix now correspond to variable $x_2$, the residual is denoted by $r_2$ and is used to reduce $[h'_{22}]$, if possible. Similarly we can calculate $r_1$ and reduce $[h'_{11}]$. We give a pseudocode of the refinement method in Algorithm 1. Note that the input of the algorithm is the interval Hessian, $[H_F] = [H_f] + D$, of the $\alpha$BB underestimator with $d_i = 2\alpha_i$ calculated by Eq. (2). However, as mentioned in the Introduction, the input matrix can be the interval Hessian of the underestimator where the $\alpha$ values have been calculated with any other method.

**Algorithm 1** Refinement algorithm ($O(n^4)$)

1: Inputs: $n \times n$ interval Hessian, $[H_F]$, of the $\alpha$BB underestimator, the diagonal shift matrix, $D$, where $d_i = 2\alpha_i$.
2: Initialize $m_i = 0$ for $i = 1, \ldots, n$.
3: **for** $i = 0, 1, \ldots, n-1$ **do**
4: If $i > 0$, interchange rows $n-i, n$ and columns $n-i, n$ of $[H_F]$.
5: Calculate residual $r_{n-i}$ of $[H_F]$. If $r_{n-i} \leq 0$ (or if at any step during calculation of $r_{n-i}$ the result is $\leq 0$ ) stop and exit the loop.
6: Reduce the diagonal entry $[h'_{n-i,n-i}]$ of $[H_F]$ by a value $m_{n-i} \in [0, \min\{r_{n-i}, d_{n-i}\}]$.
7: **end for**
8: The new $\alpha$ values are $\alpha'_i = (d_i - m_i)/2$, $i = 1, \ldots, n$.

Note that if Algorithm 1 terminates for some iteration $k$ at step 5, any improvements obtained up to that point $(m_n, \ldots, m_{n-(k-1)})$ to the $\alpha$ values, $\alpha_n, \ldots, \alpha_{n-(k-1)}$, are still valid. The rest of the values, $m_{n-k}, \ldots, m_1$, are still zero from the initialization at step 2. Thus, the new $\alpha$ values given at step 8 are valid.

The question arises of how to choose a value for $m_{n-i} \in [0, \min\{r_{n-i}, d_{n-i}\}]$ at step 5 of Algorithm 1. At the first iteration we can choose $m_n = \min\{r_n, d_n\}$. However, it might be wiser to "spread" the reduction to all the diagonal elements (if possible). We consider three approaches:

$$\text{Shared: } m_{n-i} = \min\left\{\frac{r_{n-i}}{n-i}, d_{n-i}\right\},\ i = 0, 1, \ldots, n-1 \tag{10}$$

$$\text{Extra-weighted: } m_{n-i} = \min\left\{\frac{r_{n-i}}{n-i} + w_{n-i}\left(r_{n-i} - \frac{r_{n-i}}{n-i}\right), d_{n-i}\right\}, \quad \text{where } w_{n-i} = \frac{d_{n-i}}{\sum_{j=1}^{n} d_j} \text{ and } i = 0, 1, \ldots, n-1 \tag{11}$$

$$\text{Weighted: } m_{n-i} = \begin{cases} \min\left\{\frac{d_{n-i}}{\sum_{j=1}^{n-i} d_j} r_{n-i}, d_{n-i}\right\}, & i = 0, 1, \ldots, n-2 \\ 0, & \text{if } d_1 = 0 \text{ and } i = n-1 \\ r_1, & \text{otherwise} \end{cases} \tag{12}$$

In the shared option, the current reduction is equal to the current residual divided by the number of remaining diagonal entries to be reduced. In the extra-weighted option the current reduction has the same value as in the shared option plus a weighted portion ($w_{n-i}$) of what remains if we subtract this value from the residual. In the weighted option the reduction value is a portion of the current residual which is given by to the ratio of $d_{n-i}$ over the sum of the remaining $d_j$ values to be reduced.

Let us now give an example of the refinement algorithm so that it may become clearer to the reader. We will use the shared reduction option for this example. Consider the $3 \times 3$ (symmetric) interval matrix:

$$[H_f] = \begin{bmatrix} -5 & [3,4] & [6,7] \\ [3,4] & -2 & [5,6] \\ [6,7] & [5,6] & -4 \end{bmatrix}. \tag{13}$$

The fact that the diagonal elements of the example matrix are scalar bears no significance. In fact, in practice, only the lower bounds of the diagonal elements need to be considered (see Lemma 1 in [13]). Calculating the $\alpha$ values using Eq. (2) (with $k_i = 1$, for $i = 1, 2, 3$) we find, $\alpha_1 = 8$, $\alpha_2 = 6$, $\alpha_3 = 8.5$. The hypothetical interval Hessian is:

$$[H_F] = \begin{bmatrix} 11 & [3,4] & [6,7] \\ [3,4] & 10 & [5,6] \\ [6,7] & [5,6] & 13 \end{bmatrix}. \tag{14}$$

Performing calculations (7)–(9) on $[H_F]$ we find, at step 1: $\left[\underline{m_{11}^{(1)}}, \overline{m_{11}^{(1)}}\right] = [11, 11] = 11$, at step 2: $\left[\underline{m_{11}^{(2)}}, \overline{m_{11}^{(2)}}\right] = [8.54, 9.18]$ and at step 3: $\left[\underline{m_{11}^{(3)}}, \overline{m_{11}^{(3)}}\right] = [6.31, 9.18]$. Therefore

$r_3 = 6.31 > 0$ and we now reduce the entry $h_{33}$ of $[H_F]$ by $m_3 = r_3/3 = 2.1$ and interchange rows and columns 2,3 resulting in the matrix,

$$[H'_F] = \begin{bmatrix} 11 & [6,7] & [3,4] \\ [6,7] & 10.9 & [5,6] \\ [3,4] & [5,6] & 10 \end{bmatrix}. \tag{15}$$

Again, using (7)–(9) we find $\left[\underline{m_{11}^{(1)}}, \overline{m_{11}^{(1)}}\right] = 11$, $\left[\underline{m_{11}^{(2)}}, \overline{m_{11}^{(2)}}\right] = [6.43, 7.62]$ and $\left[\underline{m_{11}^{(3)}}, \overline{m_{11}^{(3)}}\right] = [5.58, 8.39]$ respectively. Thus now $m_2 = r_2/2 = 2.79$ and the new matrix is

$$[H''_F] = \begin{bmatrix} 7.21 & [5,6] & [3,4] \\ [5,6] & 10.9 & [6,7] \\ [3,4] & [6,7] & 11 \end{bmatrix}. \tag{16}$$

Performing the same calculations once more we find $\left[\underline{m_{11}^{(1)}}, \overline{m_{11}^{(1)}}\right] = 7.21$, $\left[\underline{m_{11}^{(2)}}, \overline{m_{11}^{(2)}}\right] = [5.89, 742]$, $\left[\underline{m_{11}^{(3)}}, \overline{m_{11}^{(3)}}\right] = [4.67, 8.79]$ and so finally $m_1 = r_1 = 4.67$. The reduced $\alpha$ values are: $\alpha'_1 = \alpha_1 - m_1/2 = 5.665$, $\alpha'_2 = \alpha_2 - m_2/2 = 4.605$ and $\alpha'_3 = \alpha_3 - m_3/2 = 7.45$.

Although we cannot calculate actual minima in this case, since our example matrix was not derived from a specific function, we can measure the improvement obtained with the reduced values, $\alpha'_i$, using Eq. (3). More specifically, we can set $(\overline{x_i} - \underline{x_i})^2 = 1$, $i = 1, 2, \ldots, n$ and consider the percentage of improvement with respect to the (hypothetical) maximal separation distance,

$$I = 100\left(1 - \sum_{i=1}^{n} \alpha'_i \Big/ \sum_{i=1}^{n} \alpha_i\right)\%. \tag{17}$$

The value of $I$ can vary from 0% (no reduction at all in the $\alpha$ values), up to 100% (the initial matrix is identified as positive semi-definite). For our example we have $I = 21.2\%$, meaning that (by this measure) the refinement led to a 21.2% reduction in the maximal separation distance.

Note that we could simply apply the recursive procedure given in Eq. (5) on the initial Hessian matrix to determine whether it is positive semi-definite. This concept was proposed in [17]. In this work, however, we are interested in reducing the $\alpha$ values and not identifying whether the initial interval Hessian is positive semi-definite or not.

## 5 Results on random symmetric interval matrices

In this section we present results from the application of the refinement algorithm on randomly generated symmetric interval matrices. We have generated four groups of one thousand random matrices each with dimension 3, 4, 5 and 7 respectively and with the intervals in each matrix varying in $[-10, 10]$. The lower bound, $\underline{h_{ij}} = \underline{h_{ji}}$, of each non-diagonal entry is chosen randomly (uniform distribution) from $[-10, 10]$ and then the upper bound, $\overline{h_{ij}} = \overline{h_{ji}}$, is chosen from $[\underline{h_{ij}}, 10]$. The diagonal (scalar) entries are again chosen randomly from the

interval $[-10, 10]$ (as mentioned earlier, in practice we do not need to consider interval diagonal entries). For each matrix in each group we apply Algorithm 1 with all three different reduction options [Eqs. (10)–(12)] and we calculate the percentage improvement (reduction) in the maximum separation distance, $I$, given by Eq. (17). For each group of matrices we plot three histograms of the $I$ values obtained after applying the refinement algorithm with each reduction option respectively in Figs. 1, 2, 3 and 4. Furthermore, in Table 1 we give the mean values attained by $I$ for each reduction option in each group of random matrices.

Notice that cases where the $\alpha$ values calculated by the scaled Gerschgorin method were all zero have been filtered out from the results of both this and the next section. In practice, for such cases, there is no need for underestimation as the problem is already convex over the domain of interest.

We can make the following observations. First, as the matrix dimension increases the mean $I$ values improve (increase) for all cases. Second, the shared (10) and extra-weighted

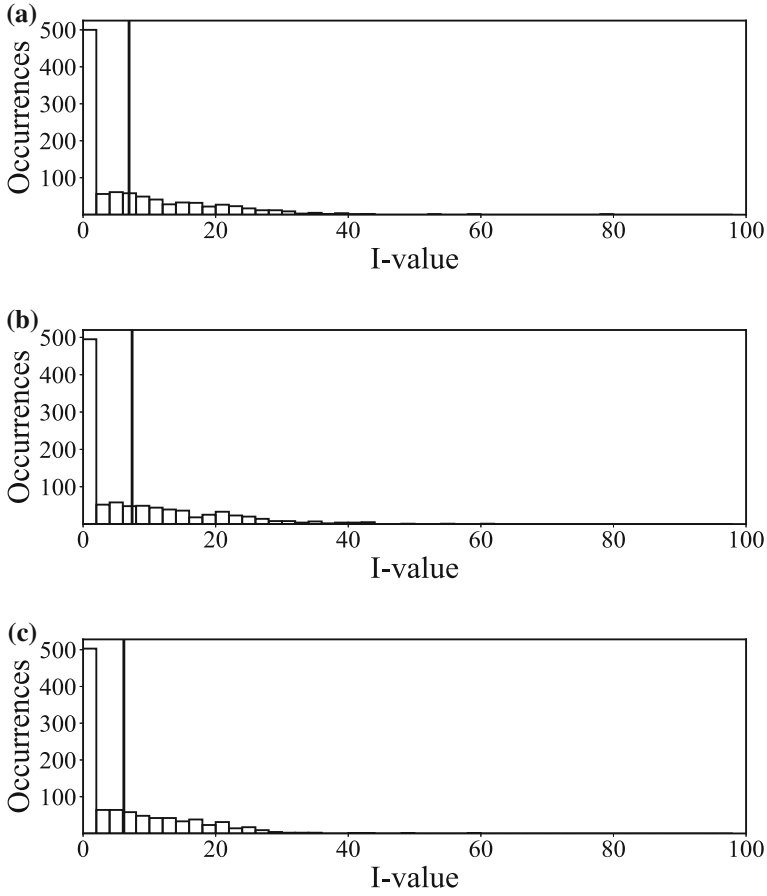


**Fig. 1** Histogram of the $I$ values [Eq. (17)] for the 1000 $3 \times 3$ random matrices using **a** the shared option (10), **b** the extra-weighted option (11) and **c** the weighted option (12). The thick vertical line indicates the mean value of $I$

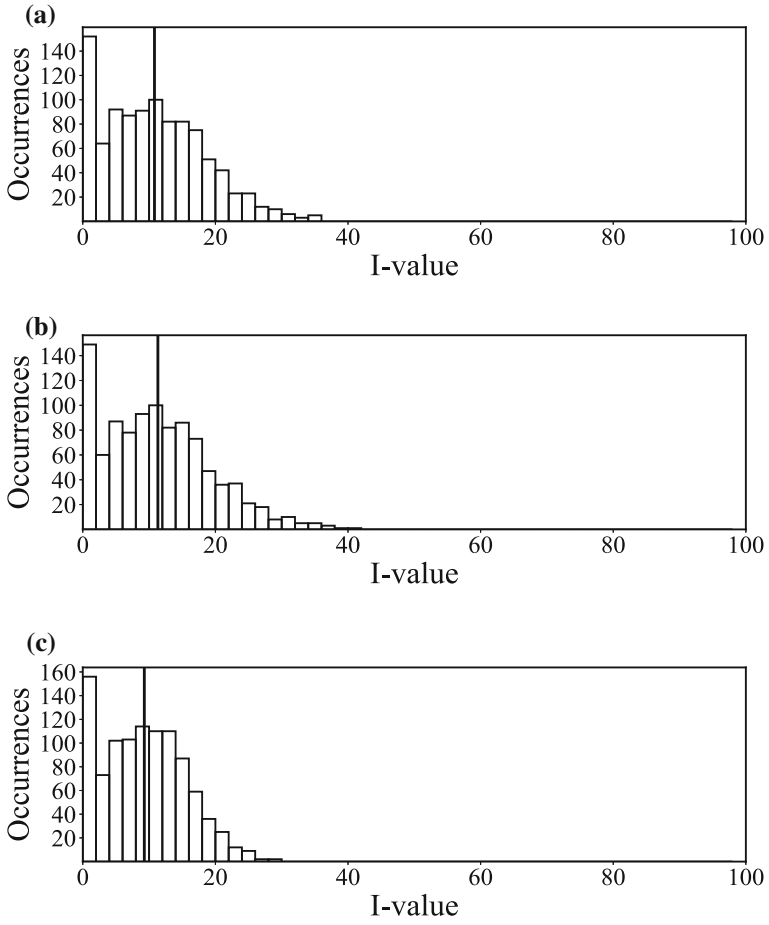


**Fig. 2** Histogram of the $I$ values [Eq. (17)] for the 1000 $4 \times 4$ random matrices using **a** the shared option (10), **b** the extra-weighted option (11) and **c** the weighted option (12). The thick vertical line indicates the mean value of $I$

(11) options perform significantly better than the weighted option (12) in all four cases while the extra-weighted option performs slightly better than the shared option. For a more detailed analysis of the performance of the shared and extra-weighted options, we plot a histogram (Fig. 5) of the value of $I_2 - I_1$, where $I_1$ and $I_2$ are the values for the shared method (Fig. 4a) and the extra-weighted method (Fig. 4b), respectively. As can be seen, in the majority of cases in Fig. 5 $I_2 - I_1$ is positive. Therefore, we can conclude that the extra-weighted option appears preferable overall.

## 6 Results on random interval Hessian matrices

In this section we present results from the application of the refinement algorithm on symmetric interval Hessian matrices calculated over random sub-domains of the following three test functions:

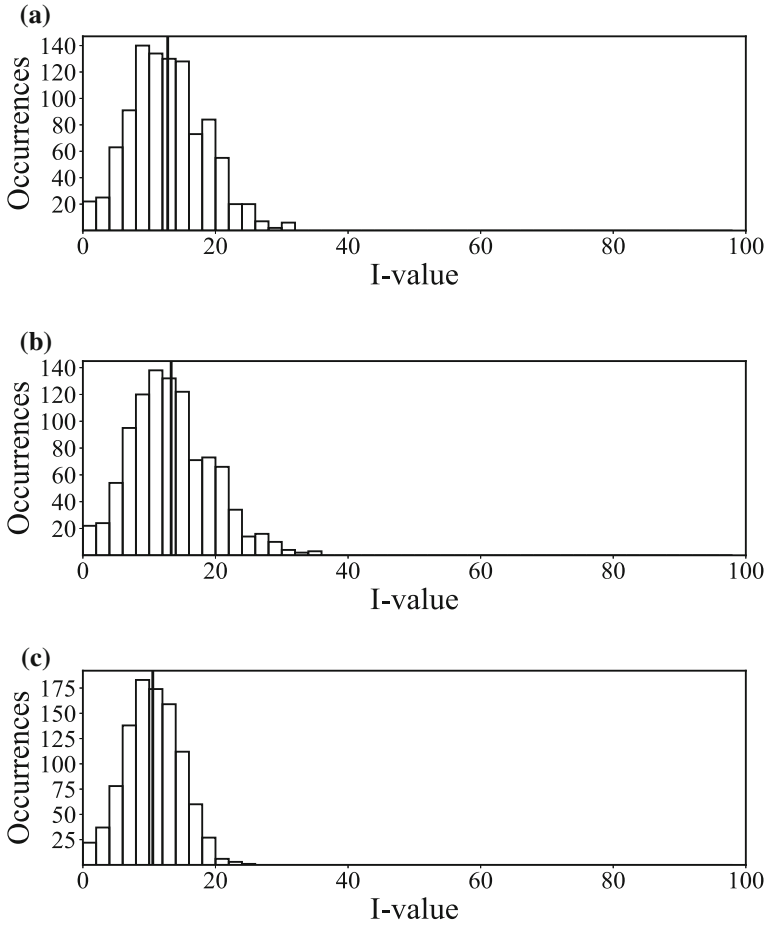


**Fig. 3** Histogram of the $I$ values [Eq. (17)] for the 1000 $5 \times 5$ random matrices using **a** the shared option (10), **b** the extra-weighted option (11) and **c** the weighted option (12). The thick vertical line indicates the mean value of $I$

Griewank:

$$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}),\ n = 4,\ x \in [-5, 5]^4. \tag{18}$$

Levy:

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2,$$
$$y_i = 1 + (x_i - 1)/4, n = 5, x \in [-5, 5]^5. \tag{19}$$

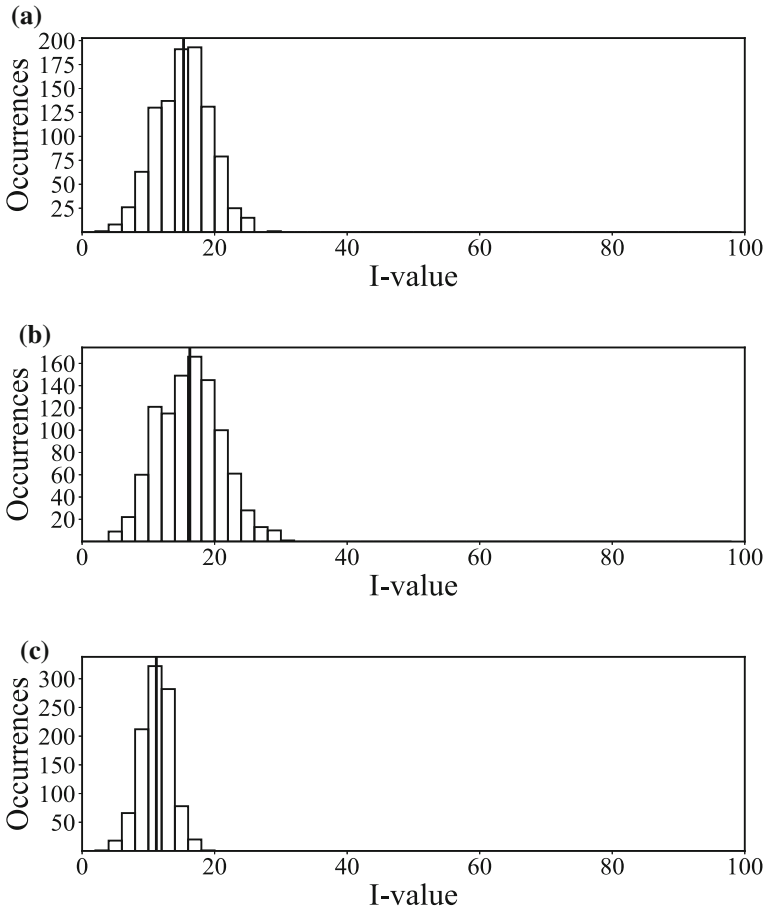


**Fig. 4** Histogram of the $I$ values [Eq. (17)] for the 1000 $7 \times 7$ random matrices using **a** the shared option (10), **b** the extra-weighted option (11) and **c** the weighted option (12). The thick vertical line indicates the mean value of $I$

Himmelblau (extension to $n$ dimensions):

$$f(x) = \sum_{i<j}^{n} \left[ (x_i^2 + x_j - 11)^2 + (x_i + x_j^2 - 7)^2 \right], \; n = 5, \; x \in [-5, 5]^5. \tag{20}$$

For each function we calculate three groups of one thousand Hessian matrices each, over random hyper-rectangular domains with randomly chosen centres and with sides of randomly varying length within $(0, L)$ for a) $L = 2$, b) $L = 1$ and c) $L = 0.2$. We then calculate the $\alpha_i$ values using the scaled Gerschgorin method [Eq. (2)] with $k_i = \overline{x_i} - \underline{x_i}$. Next we calculate the refined $\alpha_i'$ values using the extra-weighted reduction option [Eq. (11)] and we plot a corresponding histogram of the values

$$I = 100 \left( 1 - \frac{\sum_{i=1}^{n} \alpha_i' (\overline{x_i} - \underline{x_i})^2}{\sum_{i=1}^{n} \alpha_i (\overline{x_i} - \underline{x_i})^2} \right) \%. \tag{21}$$

**Table 1** Mean $I$ values for each reduction option in each group of random matrices

| Option/dimension | D = 3 (%) | D = 4 (%) | D = 5 (%) | D = 7 (%) |
|---|---|---|---|---|
| Shared (10) | 6.9 | 10.8 | 12.8 | 15.3 |
| Extra-weighted (11) | **7.4** | **11.3** | **13.3** | **16.3** |
| Weighted (12) | 6.2 | 9.3 | 10.5 | 11.2 |

The boldface indicates the option which gives the largest mean value of $I$

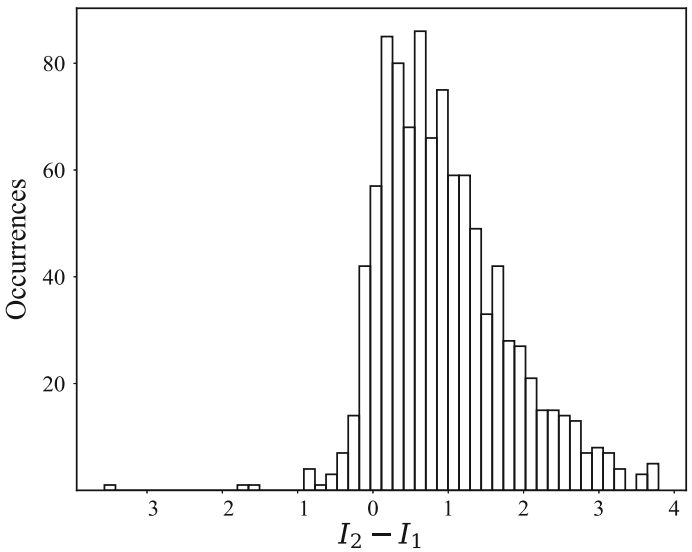


**Fig. 5** Comparison of the shared (10) and extra-weighted (11) reduction options: histogram of the values $I_2 - I_1$, where $I_1$ and $I_2$ are the values for the shared method (Fig. 4a) and the extra-weighted method (Fig. 4b), respectively

The results are given in Figs. 6, 7 and 8. We also give the mean $I$ value attained for each test function for each value of $L$ in Table 2.

As mentioned earlier, the results differ for each case since the Hessians have a certain structure and entry values. In Fig. 6 (Griewank) the refinement method results in an improvement of approximately 14% regardless of the value of $L$. In Fig. 7 (Levy) we see that the refinement method is most when $L = 0.2$ with average improvement of 11.5% percent. In Fig. 8 (extended Himmelblau) the refinement algorithm performs well for all cases with increasing improvement (21.5%, 27.4% and 32.6%) as the value of $L$ becomes smaller. Finally, in Fig. 9 we compare once more the shared reduction option and the extra-weighted reduction option by comparing the $I$ values [given by Eq. (21)] obtained for the Griewank function when $L = 2$. Again, the extra-weighted option exhibits better performance.

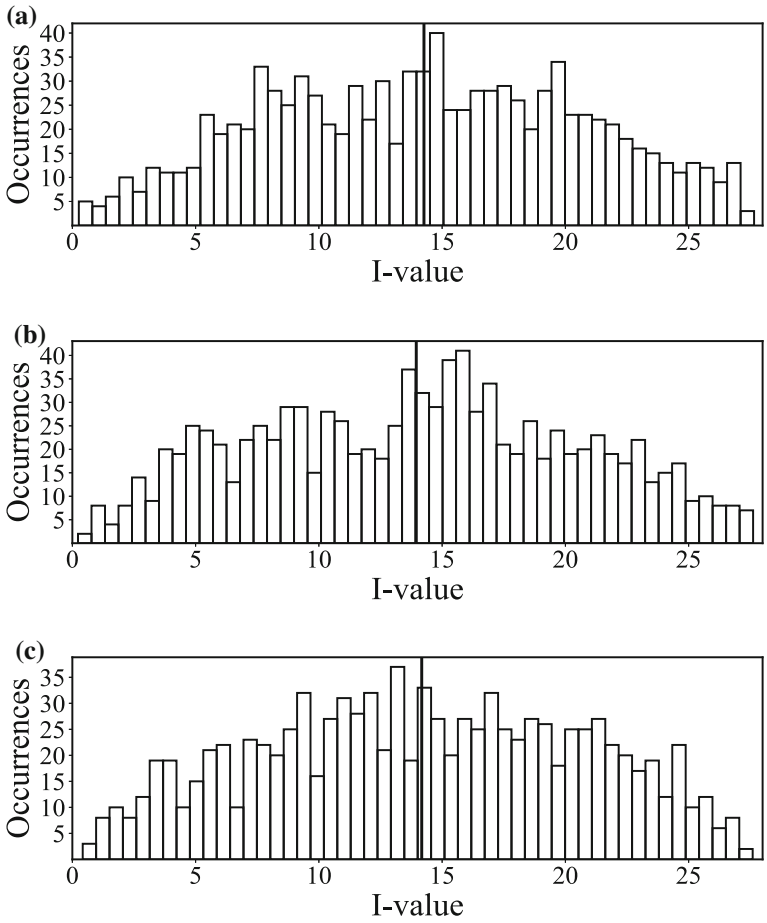


**Fig. 6** Histogram of the $I$ values [Eq. (21)] for interval Hessians of the Griewank function on randomly selected hyper-rectangular areas with **a** $L = 2$, **b** $L = 1$ and **c** $L = 0.2$. The thick vertical line indicates the mean value of $I$

## 7 Conclusions

We have presented a refinement method which we use in conjunction with the scaled Gerschgorin method in order to improve (reduce) the $\alpha$ values needed for the convex underestimator of the deterministic global optimization algorithm $\alpha$BB. The refinement method can also be utilized with other available methods for the calculation of the $\alpha$ values.

We have applied our algorithm on randomly generated symmetrical interval matrices as well as interval Hessian matrices taken from test functions. In order to compare the scaled Gerschgorin method and the refinement method we used as a measure the maximal separation distance of the underestimator.

In the experiments with the randomly generated matrices we used four groups of matrices with dimension 3, 4, 5 and 7 respectively and with each group consisting of a thousand matrices. The results showed that the refinement method improved the maximal separation distance by an average of 7.4%, 11.3%, 13.3% and 16.3% for each group respectively.

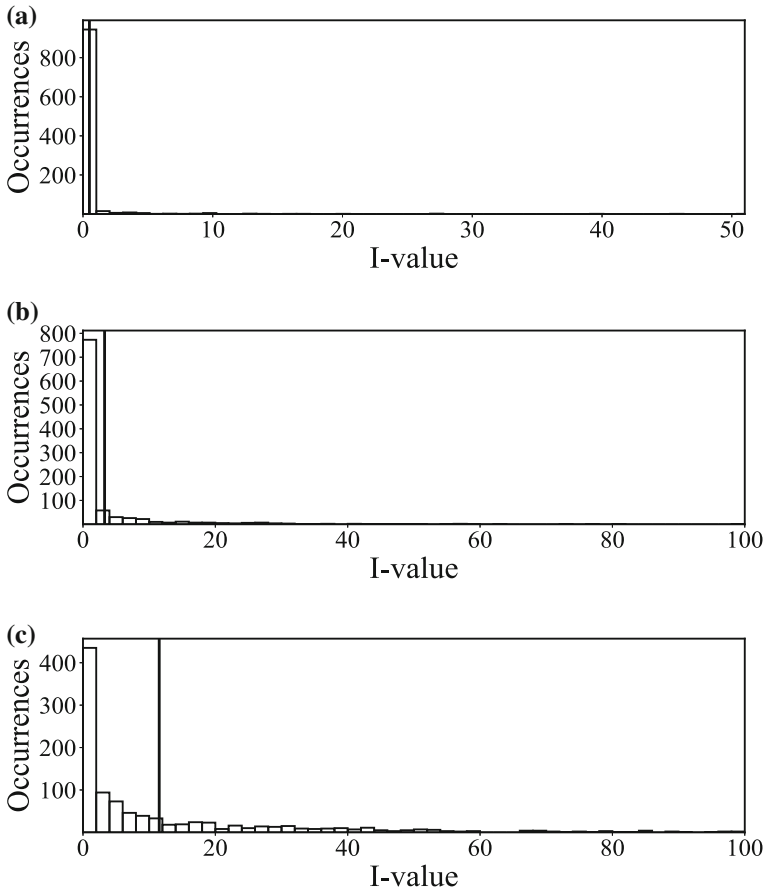


**Fig. 7** Histogram of the $I$ values [Eq. (21)] for interval Hessians of the Levy function on randomly selected hyper-rectangular areas with **a** $L = 2$, **b** $L = 1$ and **c** $L = 0.2$. The thick vertical line indicates the mean value of $I$

In the experiments with the interval Hessian matrices we used three test functions: 4D-Griewank, 5D-Levy and a 5D-extension of the Himmelblau function. For each test function we calculated three groups of a thousand interval Hessians each. The Hessians were calculated over randomly chosen hyper-rectangular areas with sides of length $(0, L)$ where $L = 2$, $L = 1$ and $L = 0.2$ for each group respectively. As is natural, the results differ for each function. For the Griewank function the results where similar regardless of the value of $L$ with an average improvement of approximately 14%. For the Levy function there was no significant improvement for $L = 2$ and $L = 1$. However for $L = 0.2$ the improvement was 11.5%. Finally, for the extended Himmelblau function we observed 21.5%, 27.4% and 32.6% improvement for $L = 2$, $L = 1$ and $L = 0.2$, respectively, with many of cases having 100% improvement.

Furthermore, we have tested three different reduction options for step 5 of the refinement algorithm and, based on our numerical results, we have concluded that the extra-weighted option (11) performs the best.

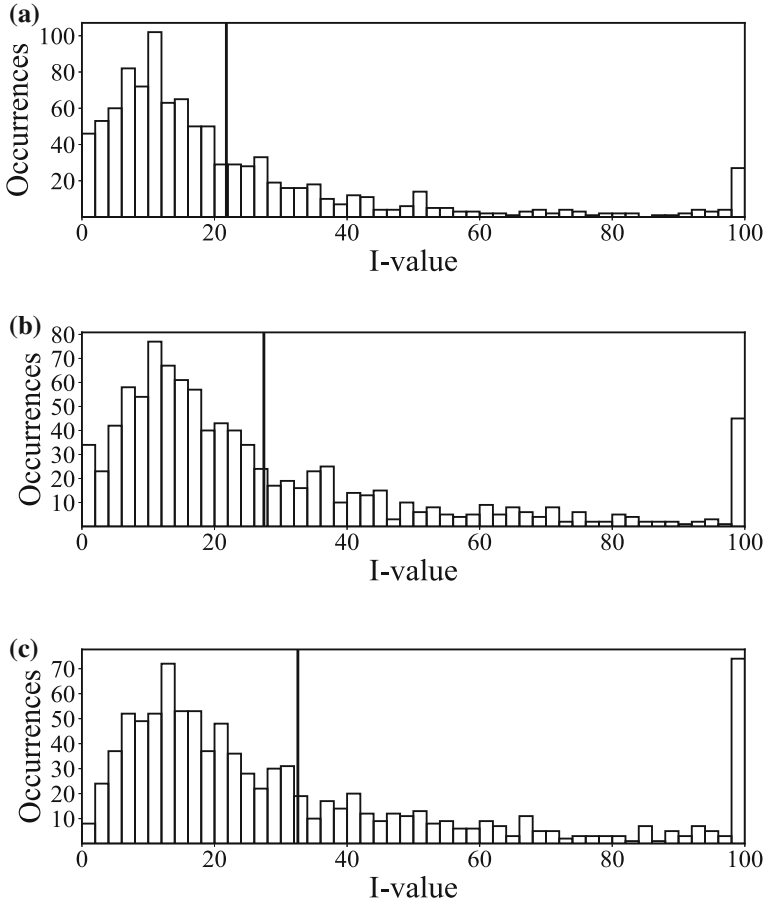


**Fig. 8** Histogram of the $I$ values [Eq. (21)] for interval Hessians of the extended Himmelblau function on randomly selected hyper-rectangular areas with **a** $L = 2$, **b** $L = 1$ and **c** $L = 0.2$. The thick vertical line indicates the mean value of $I$

**Table 2** Mean $I$ values for each combination of test function and $L$ value

| Test func./$L$-value | $L = 2$ (%) | $L = 1$ (%) | $L = 0.2$ (%) |
|---|---|---|---|
| Griewank (4D) | 14.2 | 14.0 | 14.1 |
| Levy (5D) | 0.5 | 3.3 | 11.5 |
| Himmel. (5D) | 21.5 | 27.4 | 32.6 |

From the above we conclude that the refinement method can result in a considerable improvement with respect to the the maximal separation distance. Despite the fact that this improvement comes at a (reasonable) computational cost, the improvement in the $\alpha$ values can result in an overall reduction of computational time, if nodes are fathomed at a higher rate during the execution of the Branch-and-Bound algorithm. As future work, it remains to be seen whether the refinement method is cost-effective, when integrated for use into the $\alpha$BB algorithm.

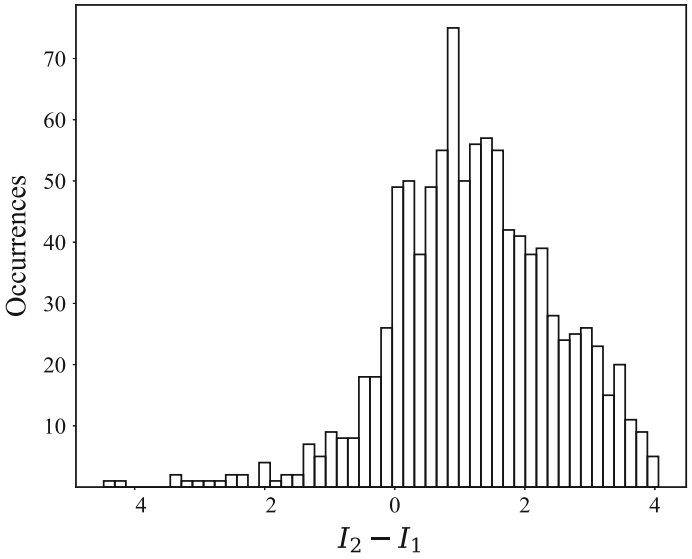


**Fig. 9** Comparison of the shared (10) and extra-weighted (11) reduction options on the random Hessians corresponding to Fig. 6 (Griewank, $L = 2$). Histogram of the values $I_2 - I_1$, where $I_1$ corresponds to the shared option $I$ values and $I_2$ to the extra-weighted option $I$ values

**Acknowledgements** Financial support from the Engineering and Physical Sciences Research Council (EPSRC) of the UK, via a Leadership Fellowship (EP/J003840/1), is gratefully acknowledged.

**Data Statement** Supporting data for this work are available in https://doi.org/10.5281/zenodo.1319047.



## References

1. Adjiman, C.S., Androulakis, I.P., Floudas, C.A.: A global optimization method, $\alpha$BB, for general twice-differentiable constrained NLPs-II. Implementation and computational results. Comput. Chem. Eng. **22**(9), 1159–1179 (1998)
2. Adjiman, C.S., Dallwig, S., Floudas, C.A., Neumaier, A.: A global optimization method, $\alpha$BB, for general twice-differentiable constrained NLPs I. Theor. Adv. Comput. Chem. Eng. **22**, 1137–1158 (1998)
3. Akrotirianakis, I.G., Floudas, C.A.: Computational experience with a new class of convex underestimators: box-constrained NLP problems. J. Glob. Optim. **29**(3), 249–264 (2004)
4. Akrotirianakis, I.G., Floudas, C.A.: A new class of improved convex underestimators for twice continuously differentiable constrained NLPs. J. Glob. Optim. **30**(4), 367–390 (2004)
5. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: $\alpha$BB: a global optimization method for general constrained nonconvex problems. J. Glob. Optim. **7**(4), 337–363 (1995)
6. Carlson, D., Haynsworth, E., Markham, T.: A generalization of the Schur complement by means of the Moore–Penrose inverse. SIAM J. Appl. Math. **26**, 169–175 (1974)
7. Cottle, R.W.: Manifestations of the Schur complement. Linear Algebra Appl. **8**, 189–211 (1974)
8. Du, K., Kearfott, R.B.: The cluster problem in multivariate global optimization. J. Glob. Optim. **5**(3), 253–265 (1994)
9. Guzman, Y.A., Hasan, M.F., Floudas, C.A.: Performance of convex underestimators in a branch-and-bound framework. Optim. Lett. **10**(2), 283–308 (2014)

10. Hansen, E., Walster, G.: Global Optimization Using Interval Analysis. Pure and Applied Mathematics. M. Dekker, New York (2003)
11. Haynsworth, E.V.: Determination of the inertia of a partitioned Hermitian matrix. Linear Algebra Appl. **1**(1), 73–81 (1968)
12. Hertz, D.: The extreme eigenvalues and stability of real symmetric interval matrices. IEEE Trans. Autom. Control **37**(4), 532–535 (1992)
13. Hladík, M., Daney, D., Tsigaridas, E.P.: Bounds on real eigenvalues and singular values of interval matrices. SIAM J. Matrix Anal. Appl. **31**(4), 2116–2129 (2010)
14. Lasserre, J.B., Thanh, T.P.: Convex underestimators of polynomials. J. Glob. Optim. **56**(1), 1–25 (2013)
15. Maranas, C.D., Floudas, C.A.: A deterministic global optimization approach for molecular structure determination. J. Chem. Phys. **100**(2), 1247–1261 (1994)
16. Meyer, C.A., Floudas, C.A.: Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: spline $\alpha$BB underestimators. J. Glob. Optim. **32**(2), 221–258 (2005)
17. Meyer, C.A., Swartz, C.L.E.: A regional convexity test for global optimization: application to the phase equilibrium problem. Comput. Chem. Eng. **22**, 1407–1418 (1998)
18. Nerantzis, D., Adjiman, C.S.: Enclosure of all index-1 saddle points of general nonlinear functions. J. Glob. Optim. **67**, 451–474 (2016). https://doi.org/10.1007/s10898-016-0430-8
19. Skjäl, A., Westerlund, T.: New methods for calculating $\alpha$BB-type underestimators. J. Glob. Optim. **58**(3), 411–427 (2014)
20. Skjäl, A., Westerlund, T., Misener, R., Floudas, C.A.: A generalization of the classical $\alpha$BB convex underestimation via diagonal and nondiagonal quadratic terms. J. Optim. Theory Appl. **154**(2), 462–490 (2012)
21. Wechsung, A., Schaber, S.D., Barton, P.I.: The cluster problem revisited. J. Glob. Optim. **58**(3), 429–438 (2014)