Efficient computation of spectral bounds for Hessian matrices on hyperrectangles for global optimization

Moritz Schulze Darup[†], Martin Kastsian[†], Stefan Mross[†], *and* M. Mönnigmann[†]

Abstract

We compare two established and a new method for the calculation of spectral bounds for Hessian matrices on hyperrectangles by applying them to a large collection of 1522 objective and constraint functions extracted from benchmark global optimization problems. Both the tightness of the spectral bounds and the computational effort of the three methods, which apply to C^2 functions $\varphi : \mathbb{R}^n \to \mathbb{R}$ that can be written as codelists, are assessed. Specifically, we compare eigenvalue bounds obtained with the interval variant of Gershgorin's circle criterion [2, 8], Hertz and Rohn's [9, 20] method for tight bounds of interval matrices, and a recently proposed Hessian matrix eigenvalue arithmetic [16], which deliberately avoids the computation of interval Hessians. The eigenvalue arithmetic provides tighter, as tight, and less tight bounds than the interval variant of Gershgorin's circle criterion in about 15%, 61%, and 24% of the examples, respectively. Hertz and Rohn's method results in bounds that are always as tight as or tighter than those from Gershgorin's circle criterion, and as tight as or tighter than those from the eigenvalue arithmetic in 96% of the cases. In 4% of the examples, the eigenvalue arithmetic results in tighter bounds than Hertz and Rohn's method. This result is surprising, since Hertz and Rohn's method provides tight bounds for interval matrices. The eigenvalue arithmetic provides tighter bounds in these cases, since it is not based on interval matrices.

Keywords. eigenvalue bounds, spectral bounds, Hessian, interval matrix, global optimization.

1 Introduction

We compare a recently proposed method [14] for the calculation of spectral bounds for Hessian matrices on hyperrectangles to existing ones. We begin with a concise problem

[†] M. Schulze Darup, M. Kastsian, S. Mross, and M. Mönnigmann are with Automatic Control and Systems Theory, Department of Mechanical Engineering, Ruhr-Universität Bochum, 44801 Bochum, Germany. E-mail: `martin.moennigmann@rub.de`.

statement. Let $\varphi : U \subseteq \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function on an open set $U \subseteq \mathbb{R}^n$ and let $B = [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_n, \overline{x}_n]$ be a closed hyperrectangle in U. The problem of interest reads as follows.

$$\begin{aligned} &\text{Find } \underline{\lambda} \in \mathbb{R}, \overline{\lambda} \in \mathbb{R} \text{ such that} \\ &\underline{\lambda} \leq \lambda \leq \overline{\lambda} \text{ for all eigenvalues } \lambda \text{ of all matrices } H \in \mathcal{H}(\varphi, B), \end{aligned} \tag{1}$$

where $\mathcal{H}(\varphi, B)$ is the set of Hessian matrices of φ on B

$$\mathcal{H}(\varphi, B) = \left\{ \nabla^2 \varphi(x) \,|\, x \in B \right\}. \tag{2}$$

A bound $\overline{\lambda}$ (resp. $\underline{\lambda}$) is called *tight* if there exists at least one matrix H in the matrix set with an eigenvalue $\lambda = \overline{\lambda}$ (resp. $\lambda = \underline{\lambda}$). Note that the bounds $\underline{\lambda}$, $\overline{\lambda}$ in (1) may or may not be tight.

Problem (1) appears in various applications. It is crucial, for example, to establish the convexity of nonlinear functions in nonlinear optimization, since methods for solving nonconvex optimization problems are much less efficient than those for their convex counterparts. If (1) results in $\underline{\lambda} \geq 0$ then φ is convex on the interior of the hyperrectangle B [4,19]. If, in contrast, $\underline{\lambda} < 0$ results from (1), then $\varphi(x)$ may or may not be convex on B, but

$$\breve{\varphi}(x) = \varphi(x) - \frac{1}{2} \underline{\lambda} \sum_{i=1}^{n} (\underline{x}_i - x_i)(\overline{x}_i - x_i) \tag{3}$$

is a convex function that underestimates φ on B and coincides with φ at the vertices of B [3,11,12]. Underestimators of this type are employed in nonconvex global optimization to bound the global minimum from below. Essentially, B is bisected into smaller and smaller hyperrectangles $\tilde{B}$ in these approaches to obtain tighter and tighter convex underestimators. This requires solving (1) repeatedly for different domains $\tilde{B} \subset B$ but the same function φ. As a result, a considerable fraction of the total computational time is spent on the calculation of convex underestimators [1]. Consequently, fast methods for solving (1) are of interest in this field. Problem (1) also arises in automatic control and systems theory. We refer to [15] for a simple example, where eigenvalue bounds for Hessian matrix sets are used to prove the positive or negative invariance of regions in the state space of nonlinear dynamical systems.

Problem (1) is commonly solved in two steps: (i) The *interval Hessian matrix* is calculated. (ii) One out of several existing methods that provide bounds on the eigenvalues of *symmetric interval matrices* [2,9,20] is applied. Interval Hessian matrices can efficiently be computed by combining interval arithmetics (IA for short; see, e.g., [17]) and automatic differentiation (AD for short; see, e.g., [7,18]). This results in intervals $[\underline{\nabla^2 \varphi_{ij}}, \overline{\nabla^2 \varphi_{ij}}] \subset \mathbb{R}$, $i = 1, \ldots, n$, $j = 1, \ldots, n$, such that

$$\left(\nabla^2 \varphi(x) \right)_{ij} \in [\underline{\nabla^2 \varphi_{ij}}, \overline{\nabla^2 \varphi_{ij}}] \tag{4}$$

for all $x \in B$, where $\underline{\nabla^2 \varphi_{ij}} = \underline{\nabla^2 \varphi_{ji}}$ and $\overline{\nabla^2 \varphi_{ij}} = \overline{\nabla^2 \varphi_{ji}}$ due to symmetry of $\nabla^2 \varphi(x)$. We refer to the set of matrices

$$\mathcal{H}^{\mathrm{IA}}(\varphi, B) = \left\{ H \in \mathbb{R}^{n \times n} \,\middle|\, H_{ij} \in [\underline{\nabla^2 \varphi_{ij}}, \overline{\nabla^2 \varphi_{ij}}],\, H = H^T \right\} \tag{5}$$

as the *interval Hessian* of φ on B. After calculating $\mathcal{H}^{\mathrm{IA}}(\varphi, B)$, the spectral bounds can be found by solving the following problem.

$$\begin{aligned} &\text{Find } \underline{\lambda} \in \mathbb{R}, \overline{\lambda} \in \mathbb{R} \text{ such that} \\ &\underline{\lambda} \leq \lambda \leq \overline{\lambda} \text{ for all eigenvalues } \lambda \text{ of all matrices } H \in \mathcal{H}^{\mathrm{IA}}(\varphi, B). \end{aligned} \tag{6}$$

The calculation of $\mathcal{H}^{\mathrm{IA}}(\varphi, B)$ requires $\mathcal{O}(n^2)\,N(\varphi)$ operations if the forward mode of automatic differentiation [7] is used, where $N(\varphi)$ denotes the number of operations needed to evaluate φ at a point in its domain. With the backward mode of automatic differentiation, this complexity can be reduced to $\mathcal{O}(n)\,N(\varphi)$ [7].

There exist a number of approaches to solving (6). Assuming the interval Hessian $\mathcal{H}^{\mathrm{IA}}(\varphi, B)$ is available, the computational complexity of these methods varies between $\mathcal{O}(n^2)$ for the interval variant of Gershgorin's circle criterion [2,8] and $\mathcal{O}(2^n\, n^3)$ for Hertz and Rohn's method [9,20], which provides *tight* spectral bounds for $\mathcal{H}^{\mathrm{IA}}(\varphi, B)$ (see Sect. 2.1 and 2.3 for details). However, since $\mathcal{H}(\varphi, B) \subseteq \mathcal{H}^{\mathrm{IA}}(\varphi, B)$, problem (6) is conservative compared to the original problem (1). In [16], we introduced a method for solving (1) that does not require the interval Hessian $\mathcal{H}^{\mathrm{IA}}(\varphi, B)$ and therefore avoids the conservatism inherent in (6). The major advantage of this method is the low computational complexity, which was shown to be of order $\mathcal{O}(n)\,N(\varphi)$ [16]. Note that the total numerical effort of the approaches mentioned before is the sum of the complexity for calculating $\mathcal{H}^{\mathrm{IA}}(\varphi, B)$ *and* solving (6). At least $\mathcal{O}(n)\,N(\varphi) + \mathcal{O}(n^2)$ operations are needed in these cases, and implementations based on forward mode automatic differentiation and Gershgorin's circle criterion require $\mathcal{O}(n^2)\,N(\varphi) + \mathcal{O}(n^2)$ operations.

It is the purpose of this paper to compare spectral bounds obtained with the recently proposed method [16] to those calculated by applying Gershgorin's circle criterion and Hertz and Rohn's method to the interval Hessian. Since the motivation for developing a new method was the application of Hessian eigenvalue bounds in global optimization, we apply the three compared methods to a large set of test functions generated from a collection of benchmark global optimization problems. Specifically, we extract 1522 objective and constraint functions from the COCONUT collection [22]. For each function, we randomly generate 100 hyperrectangles in its domain and compute the associated lower and upper eigenvalue bounds with the three methods. We compare both the resulting spectral bounds and the number of operations required by each of the approaches.

After introducing some notation in the remainder of this section, the three methods are summarized in Sect. 2. The central benchmark, which constitutes the main result of the paper, is stated in Sect. 3. Finally, conclusions are given in Sect. 4.

Notation. Pairs of lower and upper bounds such as $\underline{\lambda} \leq \lambda \leq \overline{\lambda}$ are denoted by intervals, i.e. $\lambda \in [\underline{\lambda}, \overline{\lambda}] \subset \mathbb{R}$, for short. Intervals $[\underline{a}, \overline{a}]$ are further abbreviated by $[a] := [\underline{a}, \overline{a}]$. Interval equality $[a] = [b]$ is understood as $\underline{a} = \underline{b}$ and $\overline{a} = \overline{b}$. Calculations with intervals are carried out with standard interval arithmetics rules, which are collected in Fact 1 without proof (see, e.g., [17]).

Fact 1: *(basic interval arithmetics)*

Let $[a]$ and $[b]$ be intervals and $a \in [a]$, $b \in [b]$ and $c \in \mathbb{R}$ be arbitrary real numbers. Then

$$\begin{aligned}
a + b &\in [a] + [b] := [\underline{a} + \underline{b}, \overline{a} + \overline{b}], && (7)\\
a\,b &\in [a]\,[b] := [\min\left(\underline{a}\,\underline{b}, \underline{a}\,\overline{b}, \overline{a}\,\underline{b}, \overline{a}\,\overline{b}\right), \max\left(\underline{a}\,\underline{b}, \underline{a}\,\overline{b}, \overline{a}\,\underline{b}, \overline{a}\,\overline{b}\right)]. && (8)\\
1/b &\in 1/[b] := [1/\overline{b}, 1/\underline{b}] && (9)\\
a + c &\in [a] + c := [\underline{a} + c, \overline{a} + c] && (10)\\
c\,a &\in c\,[a] := \begin{cases} [c\,\underline{a}, c\,\overline{a}] & \textit{if } c \geq 0 \\ [c\,\overline{a}, c\,\underline{a}] & \textit{if } c < 0, \end{cases} && (11)
\end{aligned}$$

where $0 \notin [\underline{b}, \overline{b}]$ is assumed (9). Furthermore, the power of natural numbers $m \in \mathbb{N}$, the *square root, the exponential and the natural logarithm of an interval are defined as follows.*

$$
\begin{aligned}
a^m &\in [a^m] := \begin{cases} [\underline{a}^m, \overline{a}^m] & \text{if } \underline{a} > 0 \text{ or } m \text{ odd} \\ [\overline{a}^m, \underline{a}^m] & \text{if } \overline{a} < 0 \text{ and } m \text{ even} \\ [0, \max(-\underline{a}, \overline{a})^m] & \text{if } 0 \in [a] \text{ and } m \text{ even} \end{cases} && (12)\\
\sqrt{a} &\in \left[\sqrt{[a]}\right] := [\sqrt{\underline{a}}, \sqrt{\overline{a}}], && (13)\\
\exp(a) &\in [\exp([a])] := [\exp(\underline{a}), \exp(\overline{a})], && (14)\\
\ln(a) &\in [\ln([a])] := [\ln \underline{a}, \ln \overline{a}], && (15)
\end{aligned}
$$

where $\underline{a} \geq 0$ are assumed in (13) and (15), respectively.

By a slight abuse of notation we denote both a real interval $[x] = [\underline{x}, \overline{x}] \subset \mathbb{R}$ and a hyperrectangle $[x] = [\underline{x}, \overline{x}] = [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_n, \overline{x}_n] \subset \mathbb{R}^n$, $n \geq 2$ by a lower case letter surrounded by brackets. As a generalization of Eqs. (12)–(15), interval extensions of functions $f(x)$, $f : U \subseteq \mathbb{R}^n \to \mathbb{R}$, $n \geq 1$, are denoted by $[f([x])]$. We denote gradients and the Hessian matrices of a function $f : U \subseteq \mathbb{R}^n \to \mathbb{R}$ by $\nabla f(x)$ and $\nabla^2 f(x)$, respectively, if they exist. Whenever $\underline{\nabla f}_i$, $\overline{\nabla f}_i \in \mathbb{R}$, $i = 1, \ldots, n$ are known, then these bounds define an interval vector denoted by $[\nabla f] = [\underline{\nabla f}, \overline{\nabla f}]$. Lower and upper bounds $\underline{\nabla^2 f}_{ij}$, $\overline{\nabla^2 f}_{ij} \in \mathbb{R}$, $i = 1, \ldots, n$ and $j = 1, \ldots, n$ define an interval matrix of the type (5), which is denoted by $[\nabla^2 f] = [\underline{\nabla^2 f}, \overline{\nabla^2 f}]$. Interval vectors and matrices are added component by component. The multiplication of an interval vector or matrix by an interval is understood componentwise. Finally, let $e^{(i)} \in \mathbb{R}^n$ be defined by $e^{(i)}_j = \delta_{ij}$, where δ_{ij} is Kronecker's δ, and let Z denote the zero matrix of dimension $n \times n$.

2 Numerical calculation of eigenvalue bounds of Hessian matrices on hyperrectangles

In this section we introduce the methods for the calculation of eigenvalue bounds that are applied to the collection of test cases in Sect. 3. We give only a short introduction, since these methods have been explained in detail elsewhere [2,8,9,14,16].

The compared methods have in common that they are based on a codelist. A codelist results if a function φ is broken down into a sequence of elementary unary and binary operations. More specifically, let $\varphi : U \subseteq \mathbb{R}^n \to \mathbb{R}$ denote a twice continuously differentiable function. Assume φ can be evaluated at an arbitrary point $x \in U$ by carrying out a finite sequence of operations of the form

$$
\begin{aligned}
y_1 &= x_1 \\
&\vdots \\
y_n &= x_n \\
y_{n+1} &= \Phi_{n+1}(y_1, \ldots, y_n) \\
y_{n+2} &= \Phi_{n+2}(y_1, \ldots, y_n, y_{n+1}) \\
&\vdots \\
y_{n+t} &= \Phi_{n+t}(y_1, \ldots, y_n, y_{n+1}, \ldots, y_{n+t-1}) \\
\varphi &= y_{n+t}
\end{aligned}
\qquad (16)
$$

where each Φ_{n+k}, $k = 1, \ldots, t$, represents one of the elementary operations listed in the first column of Tab. 1. The codelist lines y_k for the example $\varphi(x_1, x_2, x_3) = \exp(x_1 - 2\,x_2^2 + 3\,x_3^3)$ are given in the second column in (17) below. The interval extension of a function φ can be evaluated by replacing the operations in each line of (16) by their interval variants listed in Fact 1. For the example this results in replacing the y_k from the second column by the $[y_k]$ from the third column of (17).

Table 1: Rules for the calculation of y_k, $[y_k]$, $[y'_k]$ and $[y''_k]$ in the k-th line of the codelist (16). $[y'_k]$ refers to the interval gradient of line k with respect to x. The operations Φ_k shown here have been selected more or less arbitrarily to accommodate a reasonably large collection of examples φ treated in Sect. 3. The list of Φ_k can easily be extended [16].

`op` Φ_k	y_k	$[y_k]$	$[y'_k]$	$[y''_k]$
`var`	x_k	$[x_k]$	$[e^{(k)}, e^{(k)}]$	$[Z, Z]$
`add`	$y_i + y_j$	$[y_i] + [y_j]$	$[y'_i] + [y'_j]$	$[y''_i] + [y''_j]$
`mul`	$y_i\, y_j$	$[y_i]\,[y_j]$	$[y_i]\,[y'_j] + [y_j]\,[y'_i]$	$[y_i]\,[y''_j] + [y_j]\,[y''_i] + [y'_i]\,[y'_j]^T + [y'_j]\,[y'_i]^T$
`addConst`	$y_i + c$	$[y_i] + [c, c]$	$[y'_i]$	$[y''_i]$
`mulByConst`	$c\, y_i$	$c\,[y_i]$	$c\,[y'_i]$	$c\,[y''_i]$
`powNat`	y_i^m	$[y_i]^m$	$m\,[y_i]^{m-1}\,[y'_i]$	$m\,[y_i]^{m-2}\,((m-1)\,([y'_i]\,[y'_i]^T) + [y_i]\,[y''_i])$
`oneOver`	$1/y_i$	$1/[y_i]$	$-[y_k]^2\,[y'_i]$	$[y_k]^2\,(2\,[y_k]\,([y'_i]\,[y'_i]^T) - [y''_j])$
`square`	y_i^2	$[y_i]^2$	$2\,[y_i]\,[y'_i]$	$2\,([y'_i]\,[y'_i]^T + [y_i]\,[y''_i])$
`cube`	y_i^3	$[y_i]^3$	$3\,[y_i]^2\,[y'_i]$	$3\,[y_i]\,(2\,[y'_i]\,[y'_i]^T + [y_i]\,[y''_i])$
`sqrt`	$\sqrt{y_i}$	$[\sqrt{[y_i]}]$	$1/(2\,[y_k])\,[y'_i]$	$1/(2\,[y_k])([y''_i] + 1/(-2\,[y_i])\,([y'_i]\,[y'_i]^T))$
`exp`	$\exp(y_i)$	$[\exp([y_i])]$	$[y_k]\,[y'_i]$	$[y_k]\,([y'_i]\,[y'_i]^T + [y''_i])$
`ln`	$\ln(y_i)$	$[\ln([y_i])]$	$1/[y_i]\,[y'_i]$	$1/[y_i]\,([y''_i] - 1/[y_i]\,([y'_i]\,[y'_i]^T))$

2.1 Interval Hessians, Hertz and Rohn's method, and Gershgorin's circle criterion

Just as for the calculation of the interval extension of a function, a codelist can be extended to calculate gradients $\nabla\varphi$ and Hessians $\nabla^2\varphi$ and their interval extensions $[\nabla\varphi]$ and $[\nabla^2\varphi]$ by combining automatic differentiation (see, e.g., [7,18]) and interval arithmetics (see, e.g., [17]). The required results are summarized in the following algorithm, which summarizes results from [7] (see also [5, 18]). We recall that Z denotes the zero matrix of dimension $n \times n$.

Algorithm 1: *[5, 7] Assume φ is twice continuously differentiable on U and can be written as a codelist. Let $B \subset U$ be a hyperrectangle. Then, for all $x \in B$, we have $\varphi(x) \in [\varphi]$, $\nabla\varphi(x) \in [\nabla\varphi]$, and $\nabla^2\varphi(x) \in [\nabla^2\varphi]$, where $[\varphi]$, $[\nabla\varphi]$, and $[\nabla^2\varphi]$ are calculated by the following algorithm.*

1. *For $k = 1, \dots, n$, set $[y_k] = [\underline{x}_k, \overline{x}_k]$, $[y'_k] = [e^{(k)}, e^{(k)}]$, and set $[y''_k] = [Z, Z]$.*
2. *For $k = n+1, \dots, n+t$, calculate $[y_k]$, $[y'_k]$ and $[y''_k]$ according to columns 3–5 of Tab. 1, respectively.*
3. *Set $[\varphi] = [y_{n+t}]$, $[\nabla\varphi] = [y'_{n+t}]$, and $[\nabla^2\varphi] = [y''_{n+t}]$.*

Algorithm 1 implements the forward mode of automatic differentiation. We state the corresponding backward mode algorithm in the appendix for completeness and ease of reference in the comparisons to follow. We refer to a codelist (16), that has been extended by additional operations for the calculation of interval extensions or derivatives, as an *extended codelist* for short. The codelist (16) and the extended codelist that results from Alg. 1 are illustrated with an example.

Example 1: *(interval Hessian for* $\exp(x_1 - 2\,x_2^2 + 3\,x_3^3)$*) Let* $B \subset \mathbb{R}^3$ *be an arbitrary closed hyperrectangle and consider* $\varphi : B \to \mathbb{R}$, $\varphi(x_1, x_2, x_3) = \exp(x_1 - 2\,x_2^2 + 3\,x_3^3)$. *Alg. 1 results in the following expressions for* $[y_k]$, $[y'_k]$, *and* $[y''_k]$, *which are first stated in a table for brevity. The expressions for* y_k *stated in* (17) *do not result from Alg. 1, but are given for illustration of the codelist (16) of* φ.

k	y_k	$[y_k]$	$[y'_k]$	$[y''_k]$
1	x_1	$[x_1]$	$([1,1],[0,0],[0,0])^T$	$[Z,Z]$
2	x_2	$[x_2]$	$([0,0],[1,1],[0,0])^T$	$[Z,Z]$
3	x_3	$[x_3]$	$([0,0],[0,0],[1,1])^T$	$[Z,Z]$
4	y_2^2	$[y_2]^2$	$2\,[y_2]\,[y'_2]$	$2\,([y'_2]\,[y'_2]^T + [y_2]\,[y''_2])$
5	y_3^3	$[y_3]^3$	$3\,[y_3]^2\,[y'_3]$	$3\,[y_3]\,(2\,([y'_3]\,[y'_3])^T + [y_3]\,[y''_3])$
6	$-2\,y_4$	$-2\,[y_4]$	$-2\,[y'_4]$	$-2\,[y''_4]$
7	$3\,y_5$	$3\,[y_5]$	$3\,[y'_5]$	$3\,[y''_5]$
8	$y_1 + y_6$	$[y_1] + [y_6]$	$[y'_1] + [y'_6]$	$[y''_1] + [y''_6]$
9	$y_7 + y_8$	$[y_7] + [y_8]$	$[y'_7] + [y'_8]$	$[y''_7] + [y''_8]$
10	$\exp(y_9)$	$[\exp([y_9])]$	$[y_{10}]\,[y'_9]$	$[y_{10}]\,([y'_9]\,[y'_9]^T + [y''_9])$
	$\varphi = y_{10}$	$[\varphi] = [y_{10}]$	$[\nabla\varphi] = [y'_{10}]$	$[\nabla^2\varphi] = [y''_{10}]$

(17)

The codelist for φ *of the form (16) results from rewriting the second column of (17) as* $y_1 = x_1$, $y_2 = x_2$, $y_3 = x_3$, $y_4 = y_2^2, \ldots, y_{10} = \exp(y_9), \varphi = y_{10}$. *The extended codelist for* $[y''_k]$ *can be constructed by carrying out the expressions for* $[y_k]$, $[y'_k]$, *and* $[y''_k]$ *and storing the results line by line, i.e.,*

$$\begin{array}{rclrclrcl} [y_1] & = & [x_1]\,, & [y'_1] & = & ([1,1],[0,0],[0,0])^T, & [y''_1] & = & [Z,Z], \\ & \vdots & & & \vdots & & & \vdots & \\ [y_{10}] & = & [\exp([y_9])]\,, & [y'_{10}] & = & [y_{10}]\,[y'_9], & [y''_{10}] & = & [y_{10}]\,([y'_9]\,[y'_9]^T + [y''_9]), \end{array} \tag{18}$$

where the interval Hessian $[\nabla^2\varphi]$ *reads as* $[y''_{10}]$ *in the codelist notation. Note that the intermediate interval function values* $[y_k]$ *and the derivatives* $[y'_k]$ *are needed to calculate* $[\nabla^2\varphi]$, *while the intermediate function values* y_k *of the original codelist for* φ *are not.*

After calculating the interval Hessian $[\nabla^2\varphi]$ with Alg. 1, the relaxed problem (6) can be solved with a number of methods (see [10] for an overview). As pointed out in Sect. 1, we choose Gershgorin's circle criterion for its favorable computational complexity (see Sect. 2.3). In addition, we apply Hertz and Rohn's method, because it provides the *tight* eigenvalue bounds that solve (6). The interval variant of Gershgorin's circle criterion and Hertz and Rohn's method are summarized in the following two theorems.

Theorem 1 (interval Gershgorin [2,8])**:** *Let* $[\underline{\nabla^2\varphi}_{ij}, \overline{\nabla^2\varphi_{ij}}]$, $i, j = 1, \ldots, n$ *be intervals that define a symmetric interval matrix of the form (5). Then*

$$\underline{\lambda} = \min_{i \in \{1,\ldots,n\}} \underline{\nabla^2\varphi}_{ii} - r_i\,, \qquad \overline{\lambda} = \max_{i \in \{1,\ldots,n\}} \overline{\nabla^2\varphi_{ii}} + r_i \tag{19}$$

where the Gershgorin-radii r_i *are defined by* $r_i = \sum_{j=1, j\neq i}^{n} \max(-\underline{\nabla^2\varphi}_{ij}, \overline{\nabla^2\varphi_{ij}})$, *solve problem (6).*

Theorem 2 (Hertz [9] and Rohn [20])**:** *Let* $[\underline{\nabla^2\varphi}_{ij}, \overline{\nabla^2\varphi_{ij}}]$, $i, j = 1, \ldots, n$ *be intervals that define a symmetric interval matrix of the form (5). Define the matrices* $S^{(k)} \in \mathbb{R}^{n \times 2^k}$ *for* $k = 1, \ldots, n$ *recursively by*

$$S^{(k)} = \begin{pmatrix} & S^{(k-1)} & & & S^{(k-1)} & \\ 1 & \ldots & 1 & -1 & \ldots & -1 \end{pmatrix}, \qquad S^{(1)} = \begin{pmatrix} 1 & -1 \end{pmatrix}.$$

Define the symmetric matrices $L^{(k)} \in \mathbb{R}^{n\times n}$ *and* $U^{(k)} \in \mathbb{R}^{n\times n}$ *for* $k = 1,\ldots,2^{n-1}$ *according to*

$$L_{ij}^{(k)} = \begin{cases} \underline{\nabla^2\varphi}_{ij} & \textit{if } i=j \textit{ or } S_{ik}^{(n)}\cdot S_{jk}^{(n)} = 1 \\ \overline{\nabla^2\varphi}_{ij} & \textit{otherwise} \end{cases}, \quad U_{ij}^{(k)} = \begin{cases} \overline{\nabla^2\varphi}_{ij} & \textit{if } L_{ij}^{(k)} = \underline{\nabla^2\varphi}_{ij} \\ \underline{\nabla^2\varphi}_{ij} & \textit{otherwise} \end{cases}.$$

Then

$$\underline{\lambda} = \min_{k\in\{1,\ldots,2^{n-1}\}} \lambda_{\min}(L^{(k)}), \qquad \overline{\lambda} = \max_{k\in\{1,\ldots,2^{n-1}\}} \lambda_{\max}(U^{(k)}), \tag{20}$$

where $\lambda_{\min}(A)$ *and* $\lambda_{\max}(A)$ *denote the smallest and largest (real) eigenvalue of any symmetric real matrix* $A = A^T$*, respectively, solve problem (6).*

We briefly illustrate Thms. 1 and 2 by applying them to the sample function from Example 1.

Example 2: *(Gershgorin and Hertz/Rohn applied to* $\exp(x_1 - 2\,x_2^2 + 3\,x_3^3)$*) Without detailing the calculations we claim that substituting* $B = [-0.3, 0.2] \times [-0.1, 0.6] \times [-0.4, 0.5]$ *into the extended codelist* (18) *yields*

$$[\nabla^2\varphi] = [y_{10}''] = \begin{pmatrix} [0.298, 1.777] & [-4.265, 0.7109] & [0.000, 3.999] \\ [-4.265, 0.7109] & [-7.109, 3.128] & [-9.597, 1.599] \\ [0.000, 3.999] & [-9.597, 1.599] & [-12.795, 24.991] \end{pmatrix}.$$

Applying Thm. 1 yields the $n = 3$ *Gershgorin radii* $r_1 = 4.265 + 3.999 = 8.264$*,* $r_2 = 4.265 + 9.597 = 13.862$ *and* $r_3 = 3.999 + 9.597 = 13.596$*. Upon substitution into (19) the spectral bounds*

$$[\lambda_{\mathrm{G}}] = [\underline{\lambda}_{\mathrm{G}}, \overline{\lambda}_{\mathrm{G}}] = [-12.795 - 13.596, 24.991 + 13.596] = [-26.391, 38.587] \tag{21}$$

result, where the subscript G is short for Gershgorin. Hertz and Rohn's method requires to calculate $2\cdot 2^{n-1} = 2^n = 8$ *vertex matrices* $L^{(1)},\ldots,L^{(4)}$ *and* $U^{(1)},\ldots,U^{(4)}$ *and the sign matrix* $S^{(3)}$ *defined in Thm. 2. Equation (20) yields*

$$[\lambda_{\mathrm{H}}] = [\underline{\lambda}_{\mathrm{H}}, \overline{\lambda}_{\mathrm{H}}] = [\lambda_{\min}(L^{(2)}), \lambda_{\max}(U^{(3)})] = [-20.597, 29.603] \tag{22}$$

with $L^{(2)} = \begin{pmatrix} 0.298 & 0.711 & 3.999 \\ 0.711 & -7.109 & -9.597 \\ 3.999 & -9.597 & -12.795 \end{pmatrix}$ *and* $U^{(3)} = \begin{pmatrix} 1.777 & -4.265 & 3.999 \\ -4.265 & 3.128 & -9.597 \\ 3.999 & -9.597 & 24.991 \end{pmatrix}$,
where the subscript H is short for Hertz and Rohn. We only list the matrices $L^{(2)}$ *and* $U^{(3)}$ *that are selected in the minimization and maximization in (20) and omit the remaining six vertex matrices for brevity.*

2.2 Eigenvalue arithmetic

We summarize the eigenvalue arithmetic in Alg. 2 and Tab. 2 and refer the reader to [14,16] for details. Algorithm 2 implies that the eigenvalue arithmetic does *not* require the interval Hessian, but interval gradients suffice. Some recurring calculations to be carried out with the interval gradients can conveniently be summarized with the functions Λ_s and Λ_t defined by

$$[\Lambda_s([a])] = \begin{cases} \left[0, \sum_{i=1}^n \max(\underline{a}_i^2, \overline{a}_i^2)\right], & \text{if } n > 1 \\ [a]^2, & \text{if } n = 1 \end{cases} \tag{23}$$

and

$$[\Lambda_t([a],[b])] = \begin{cases} [-\beta,\beta] + \sum_{i=1}^{n} [\underline{a}_i, \overline{a}_i]\,[\underline{b}_i, \overline{b}_i], & \text{if } n > 1 \\ 2\,[a]\,[b], & \text{if } n = 1 \end{cases} \tag{24}$$

where $\beta = \sqrt{(\sum_{i=1}^{n} \max(\underline{a}_i^2, \overline{a}_i^2))(\sum_{i=1}^{n} \max(\underline{b}_i^2, \overline{b}_i^2))}$. We refer to [16] for a detailed discussion of the meaning of $[\Lambda_s([a])]$ and $[\Lambda_t([a],[b])]$.

Table 2: Rules for the calculation of $[\lambda_k]$ in the k-th line of the codelist (16). We assume that $[y_i]$, $[y_i']$ and $[y_j]$, $[y_j']$ for all previous lines $i \le k$, $j \le k$ have been calculated according to the rules from Tab. 1 and can be reused in line k. Rules for y_k are repeated here for convenience. The functions $[\Lambda_s]([a])$ and $[\Lambda_t]([a],[b])$ are defined in (23) and (24).

`op` Φ_k	y_k	$[\lambda_k]$
`var`	x_k	$[0,0]$
`add`	$y_i + y_j$	$[\lambda_i] + [\lambda_j]$
`mul`	$y_i\, y_j$	$[y_j]\,[\lambda_i] + [y_i]\,[\lambda_j] + [\Lambda_t([y_i'],[y_j'])]$
`addConst`	$y_i + c$	$[\lambda_i]$
`mulByConst`	$c\, y_i$	$c\,[\lambda_i]$
`powNat`	y_i^m	$m[y_i]^{m-2}((m-1)[\Lambda_s([y_i'])] + [y_i]\,[\lambda_i])$
`oneOver`	$1/y_i$	$[y_k]^2\,(2\,[y_k]\,[\Lambda_s([y_i'])] - [\lambda_i])$
`square`	y_i^2	$2\,([\Lambda_s([y_i'])] + [y_i]\,[\lambda_i])$
`cube`	y_i^3	$3\,[y_i]\,(2\,[\Lambda_s([y_i'])] + [y_i]\,[\lambda_i])$
`sqrt`	$\sqrt{y_i}$	$1/(2\,[y_k])([\lambda_i] + 1/(-2\,[y_i])[\Lambda_s([y_i'])])$
`exp`	$\exp(y_i)$	$[y_k]\,([\Lambda_s([y_i'])] + [\lambda_i])$
`ln`	$\ln(y_i)$	$1/[y_i]\,([\lambda_i] - 1/[y_i]\,[\Lambda_s([y_i'])])$

Algorithm 2 (eigenvalue arithmetic [16])**:** *Assume φ is twice continuously differentiable on U and can be written as a codelist. Let $B \subset U$ be a hyperrectangle. Then, for all $x \in B$, we have $\varphi(x) \in [\varphi]$, $\nabla\varphi(x) \in [\nabla\varphi]$, and $\lambda_\varphi \in [\lambda_\varphi]$ for all eigenvalues of $\nabla^2\varphi(x)$, where $[\varphi]$, $[\nabla\varphi]$, and $[\lambda_\varphi]$ are calculated by the following algorithm.*

1. *For $k = 1, \ldots, n$, set $[y_k] = [\underline{x}_k, \overline{x}_k]$, $[y_k'] = [e^{(k)}, e^{(k)}]$, and set $[\lambda_k] = [0,0]$.*
2. *For $k = n+1, \ldots, n+t$, calculate $[y_k]$, $[y_k']$ and $[\lambda_k]$ according to columns 3 and 4 of Tab. 1 and column 3 of Tab. 2, respectively.*
3. *Set $[\varphi] = [y_{n+t}]$, $[\nabla\varphi] = [y_{n+t}']$, and $[\lambda_\varphi] = [\lambda_{n+t}]$.*

Algorithm 2 is illustrated with the sample function from Example 1 and 2.

Example 3: *Let B and $\varphi : B \to \mathbb{R}$ be as in Example 1. Applying Alg. 2 to φ results in the expressions for $[y_k]$, $[y_k']$, and $[\lambda_k]$ listed in* (25)*, which we state in a table for brevity. Note that $[y_k]$ and $[y_k']$ are equal to those in* (17)*. These expressions are repeated here,* *since the* $[\lambda_k]$ *depend on them.*

k	$[y_k]$	$[y'_k]$	$[\lambda_k]$
1	$[x_1]$	$([1,1],[0,0],[0,0])^T$	$[0,0]$
2	$[x_2]$	$([0,0],[1,1],[0,0])^T$	$[0,0]$
3	$[x_3]$	$([0,0],[0,0],[1,1])^T$	$[0,0]$
4	$[y_2]^2$	$2\,[y_2]\,[y'_2]$	$2\,([\Lambda_s([y'_2])] + [y_2]\,[\lambda_2])$
5	$[y_3]^3$	$3\,[y_3]^2\,[y'_3]$	$3\,[y_3]\,(2\,[\Lambda_s([y'_3])] + [y_3]\,[\lambda_3])$
6	$-2\,[y_4]$	$-2\,[y'_4]$	$-2\,[\lambda_4]$
7	$3\,[y_5]$	$3\,[y'_5]$	$3\,[\lambda_5]$
8	$[y_1] + [y_6]$	$[y'_1] + [y'_6]$	$[\lambda_1] + [\lambda_6]$
9	$[y_7] + [y_8]$	$[y'_7] + [y'_8]$	$[\lambda_7] + [\lambda_8]$
10	$[\exp([y_9])]$	$[y_{10}]\,[y'_9]$	$[y_{10}]\,([\Lambda_s([y'_9])] + [\lambda_9])$
	$[\varphi] = [y_{10}]$	$[\nabla\varphi] = [y'_{10}]$	$[\lambda_\varphi] = [\lambda_{10}]$

(25)

The extended codelist for $[\lambda_\varphi]$ *results from evaluating and storing the expressions listed in* (25) *line by line, i.e.,*

$$\begin{array}{rclrclrcl} [y_1] &=& [x_1]\,, & [y'_1] &=& ([1,1],[0,0],[0,0])^T, & [\lambda_1] &=& [0,0], \\ &\vdots& & &\vdots& & &\vdots& \\ [y_{10}] &=& [\exp([y_9])]\,, & [y'_{10}] &=& [y_{10}]\,[y'_9], & [\lambda_{10}] &=& [y_{10}]\,([\Lambda_s([y'_9])] + [\lambda_9]). \end{array} \tag{26}$$

Without detailing the calculations we claim that applying (26) *to the particular hyperrectangle* $B = [-0.3, 0.2] \times [-0.1, 0.6] \times [-0.4, 0.5]$ *from Example 2 results in*

$$[\lambda_\mathrm{A}] = [\lambda_\varphi] = [-19.904, 37.004], \tag{27}$$

where the subscript A is short for arithmetic.

By comparing the spectral bounds (21), (22) and (27), we find the relations $\underline{\lambda}_\mathrm{G} < \underline{\lambda}_\mathrm{H} < \underline{\lambda}_\mathrm{A}$ and $\overline{\lambda}_\mathrm{H} < \overline{\lambda}_\mathrm{A} < \overline{\lambda}_\mathrm{G}$ for the discussed example. Note that the lower bound from the eigenvalue arithmetic is tighter than the tight bound for the interval Hessian obtained with Hertz and Rohn's method. We recall this result may arise, because the interval Hessian $\mathcal{H}^{IA}$ is in general a superset of the actual matrix set $\mathcal{H}$ of interest defined in (2). Since the eigenvalue arithmetic is not based on interval Hessians, it may result in tighter eigenvalue bounds than the tight eigenvalue bounds for the interval Hessian.

We stress that the relations between $\underline{\lambda}_\mathrm{G}$, $\underline{\lambda}_\mathrm{H}$, $\underline{\lambda}_\mathrm{A}$ and $\overline{\lambda}_\mathrm{H}$, $\overline{\lambda}_\mathrm{A}$, $\overline{\lambda}_\mathrm{G}$ found in Examples 1–3 do not hold in general. It is the very point of Section 3 to analyze these relations for a large collection of examples.

2.3 Computational complexities

The discussed methods do not only differ with respect to the tightness of the eigenvalue bounds, but also with respect to computational cost. Calculating the interval Hessian matrix with forward or backward mode automatic differentiation and applying Hertz and Rohn's method requires

$$\mathcal{O}(n^2)\,N(\varphi) + \mathcal{O}(2^n n^3) \quad \text{or} \quad \mathcal{O}(n)\,N(\varphi) + \mathcal{O}(2^n n^3) \tag{28}$$

operations [16], respectively, where $N(\varphi)$ denotes the number of operations needed for the evaluation of φ at a point. Calculating the interval Hessian with forward or backward mode automatic differentiation and applying Gershgorin's circle criterion takes

$$\mathcal{O}(n^2)\,N(\varphi) + \mathcal{O}(n^2) \quad \text{or} \quad \mathcal{O}(n)\,N(\varphi) + \mathcal{O}(n^2) \tag{29}$$

operations [16], respectively. Calculating eigenvalue bounds with the arithmetic from [16] does not involve the interval Hessian and requires

$$\mathcal{O}(n)\, N(\varphi) \tag{30}$$

operations [16]. Due to the $\mathcal{O}(2^n\, n^3)$ term in (28) the computational cost of Hertz and Rohn's grows drastically compared to (29) and (30). The complexities (29) and (30), however, are very similar and a detailed comparison is necessary. We therefore compare the numbers of operations associated with the eigenvalue arithmetic and the forward and backward variants of the interval Hessian computation in combination with Gershgorin's circle criterion. These numbers are denoted by $N_{\mathrm{A}}(\varphi)$, $N_{\mathrm{G}}(\varphi)$ and $N_{\mathrm{bG}}(\varphi)$, respectively. $N_{\mathrm{A}}(\varphi)$, $N_{\mathrm{G}}(\varphi)$ and $N_{\mathrm{bG}}(\varphi)$ can be determind for any specific φ by counting operations in the extended codelist of φ. Table 3 lists the number of operations needed in each line of the extended codelist by line type.

Table 3: Number of operations necessary to calculate y_k, $[y_k]$, $[y'_k]$, $[\lambda_k]$ and $[y''_k]$ introduced in Algs. 1 and 2 for each type of line of a codelist (16). The last column lists the number of operations for the evaluation of the set of updates $\mathcal{U}_k$ needed to calculate $[\nabla^2\varphi]$ with the backward mode of automatic differentiation. See Alg. 3 and Rem. 1 in the appendix for details. $N([y'_k])$ denotes the number of operations necessary to compute $[y'_k]$ assuming that $[y_k]$ is already available. $N([\lambda_k])$, $N([y''_k])$, and $N(\mathcal{U}_k)$ denote the number of operations necessary to compute $[\lambda_k]$, $[y''_k]$ and the updates of $[u]$ in line k, respectively, assuming $[y_k]$ and $[y'_k]$ are available. Listed numbers apply for $n > 1$.

`op` Φ_k	$N(y_k)$	$N([y_k])$	$N([y'_k])$	$N([\lambda_k])$	$N([y''_k])$	$N(\mathcal{U}_k)$
`var`	1	0	0	0	0	0
`add`	1	2	$2\,n$	2	$n\,(n+1)$	$8\,n$
`mul`	1	8	$18\,n$	$18\,n+21$	$19\,n\,(n+1)$	$60\,n$
`addConst`	1	2	0	0	0	$4\,n$
`mulByConst`	1	2	$2\,n$	2	$n\,(n+1)$	$8\,n$
`powNat`	1	5	$8\,n+7$	$4\,n+26$	$14\,n\,(n+1)+7$	$59\,n$
`oneOver`	1	2	$8\,n+7$	$4\,n+26$	$14\,n\,(n+1)+7$	$56\,n$
`square`	1	5	$8\,n+2$	$4\,n+11$	$10\,n\,(n+1)$	$36\,n$
`cube`	1	2	$8\,n+7$	$4\,n+21$	$14\,n\,(n+1)+2$	$54\,n$
`sqrt`	1	2	$8\,n+4$	$4\,n+25$	$13\,n\,(n+1)+8$	$52\,n$
`exp`	1	2	$8\,n$	$4\,n+9$	$9\,n\,(n+1)$	$30\,n$
`ln`	1	2	$8\,n+2$	$4\,n+21$	$13\,n\,(n+1)+4$	$51\,n$

An operation counted towards $N(y_k)$, $N([y_k])$, $N([y'_k])$, $N([\lambda_k])$, $N([y''_k])$ or $N(\mathcal{U}_k)$ in Tab. 3 may either be an addition, multiplication or comparison of two real numbers, or the application of one of the functions `oneOver`, `square`, `cube`, `pow`, `sqrt`, `exp` or `ln`. Note that this way of counting operations is coarse but a standard approach in the field of automatic differentiation [7, 18]. The use of Tab. 3 is illustrated with an example.

Example 4: *(number of operations for Example 1) Table 4 lists the numbers of operations necessary to evaluate $[y_k]$, $[y'_k]$, $[\lambda_k]$ and $[y''_k]$ with the codelists (17) and (25) for the sample function $\varphi(x) = \exp(x_1 - 2\,x_2^2 + 3\,x_3^3)$ ($n = 3$). The last column states the number of operations needed to calculate $[\nabla^2\varphi]$ by backward mode AD with the updates $\mathcal{U}_k$ introduced in Rem. 1 and Alg. 3. The numbers of operations to calculate the eigenvalue bounds with* *the eigenvalue arithmetic and the foward and backward AD Gershgorin variants can be obtained from*

$$N_{\mathrm{A}}(\varphi) = \sum_{k=n+1}^{n+t} N([y_k]) + N([y'_k]) + N([\lambda_k]), \tag{31}$$

$$N_{\mathrm{G}}(\varphi) = N([\lambda_{\mathrm{G}}]) + \sum_{k=n+1}^{n+t} N([y_k]) + N([y'_k]) + N([y''_k]), \text{ and} \tag{32}$$

$$N_{\mathrm{bG}}(\varphi) = N([\lambda_{\mathrm{G}}]) + \sum_{k=n+1}^{n+t} N([y_k]) + N([y'_k]) + N([\mathcal{U}_k]). \tag{33}$$

We find $N_{\mathrm{A}}(\varphi) = N([\lambda_\varphi]) = 17 + 105 + 85 = 207$, $N_{\mathrm{G}}(\varphi) = N([\nabla^2\varphi]) + N([\lambda_{\mathrm{G}}]) = 17 + 105 + 446 + 28 = 596$ and $N_{\mathrm{bG}}(\varphi) = 17 + 105 + 456 + 28 = 608$, where $N([\lambda_{\mathrm{G}}]) = 3\,n^2 + n - 2 = 28$ is the number of operations necessary to calculate $[\lambda_{\mathrm{G}}]$ assuming $[\nabla^2\varphi]$ has already been determined. For later use we note that the number of operations needed to calculate $[\lambda_{\mathrm{A}}]$ assuming $[y_k]$ and $[y'_k]$ have already been calculated is $N_{\Delta\mathrm{A}}(\varphi) = 85$. $N_{\Delta\mathrm{A}}(\varphi)$ is introduced more precisely in Eq. (36) below.

Table 4: Numbers of operations for the codelist lines of Example 1.

k	op Φ_k	$N([y_k])$	$N([y'_k])$	$N([\lambda_k])$	$N([y''_k])$	$N(\mathcal{U}_k)$
4	`square`	5	26	23	120	108
5	`cube`	2	31	33	170	162
6	`mulByConst`	2	6	2	12	24
7	`mulByConst`	2	6	2	12	24
8	`add`	2	6	2	12	24
9	`add`	2	6	2	12	24
10	`exp`	2	24	21	108	90
$\sum$		17	105	85	446	456

Before applying Tab. 3 to the collection of examples in Sect. 3, we derive some general statements. From the last three columns of Tab. 3 we infer

$$N([\lambda_k]) < N([y''_k]) \text{ and } N([\lambda_k]) < N(\mathcal{U}_k) \tag{34}$$

for all codelist line types and all $n > 1$. Combining (34) and (31)–(33) we find

$$N_{\mathrm{A}}(\varphi) < N_{\mathrm{G}}(\varphi) \qquad \text{and} \qquad N_{\mathrm{A}}(\varphi) < N_{\mathrm{bG}}(\varphi) \tag{35}$$

for any function φ that can be stated as a codelist with lines of the types from Tab. 3. Furthermore, inspection of Tab. 3 shows that the eigenvalue arithmetic can be applied at little additional computational effort, whenever eigenvalue bounds are calculated with the interval Hessian. This statement holds, since the $[y'_k]$ required for the arithmetic are available as an intermediate result in both the foward and backward AD variant. More specifically,

$$N_{\Delta\mathrm{A}}(\varphi) = \sum_{k=n+1}^{n+t} N([\lambda_k]) \tag{36}$$

additional operations are needed to calculate eigenvalue bounds with the arithmetic, if they are calculated by applying Gershgorin's circle criterion to the interval Hessian matrix. We infer from Tab. 3 that $N_{\Delta\mathrm{A}}(\varphi)$ as defined in (36) amount to $\mathcal{O}(n)$ operations.

While the number of operations can be reduced with the backward mode of automatic differentiation[1], the resulting interval enclosures are in general *not tighter* than for the forward mode [21]. This result, which is surprising at first sight, can be traced to the subdistributivity of interval arithmetics. A discussion is beyond the paper, and we refer the reader to [21]. In order to simplify the comparisons reported here, we count operations for *both the forward and backward* variant of the interval Hessian calculation in combination with Gershgorin's circle criterion, but we calculate eigenvalue bounds *only with the forward variant.* We stress this results in a comparison in favour of Gershgorin's circle criterion, since we compare our method to the tighter out of two results and the smaller out of two operation counts that can be obtained with the Gershgorin variants.

3 Benchmark: Arithmetic versus Gershgorin and Hertz

We apply the eigenvalue arithmetic (A) to a large collection of examples and compare results to those obtained by applying Gershgorin's circle criterion (G) and Hertz and Rohn's method (H) to the interval Hessian. While all interval Hessians are calculated with forward mode automatic differentiation, we report the number of operations both for forward and backward mode. See the end of Sect. 2 for an explanation. Sections 3.1 and 3.2 describe the test examples and the scheme of comparison. The actual results are summarized in Sect. 3.3.

3.1 Collection of test cases

The test cases are extracted from the COCONUT collection of optimization problems [22]. We consider all COCONUT problems with $1 < n \leq 10$ variables and extract those cost and constraint functions that can be decomposed into the operations listed in Tabs. 1 and 2 respectively. This results in a set of 1522 sample functions $\varphi : \mathbb{R}^n \to \mathbb{R}$ with $1 < n \leq 10$. For each φ, we generate 100 random hyperrectangles $B \subseteq D \subset \mathbb{R}^n$ in the domain D of φ specified in the respective COCONUT problem. Each of the three methods (A, G, and H) introduced in Sect. 2 is applied to the resulting $1522 \cdot 100$ sample problems.

We omit examples with $n = 1$, since the three methods yield identical spectral bounds in this case[2]. The upper bound $n \leq 10$ is arbitrary. The comparison in Sect. 3.3 corroborates that the eigenvalue arithmetic benefits more and more from its favorable computational complexity as n increases, which was anticipated in the comparison of computational complexities in Sect. 2.3. While the eigenvalue arithmetic and Gershgorin's circle criterion could be applied well beyond $n = 10$, it becomes tedious to calculate the exact Hessian matrix eigenvalue bounds for comparison, due to the $\mathcal{O}(2^n\, n^3)$ complexity of this problem.

Table 5 lists three sample functions from the COCONUT collection for illustration. We refer to all examples by their COCONUT name, for example `ex8_1_6`. The suffix `-i`, as in `ex8_1_6-1` for example, uniquely identifies the function in the respective COCONUT optimization problem, where $i = 1$ corresponds to the objective function and $i = 2, \ldots, m$ corresponds to the $(i-1)$-th constraint function[3].

[1] In fact this holds only for $n > \bar{n}$, where $\bar{n}$ depends on the codelist line type and can be computed from solving $N([y_k'']) = N(\mathcal{U}_k)$, where $N([y_k''])$ and $N(\mathcal{U}_k)$ are as in Tab. 3.

[2] To see $[\lambda_{\mathrm{A}}] = [\lambda_{\mathrm{G}}] = [\lambda_{\mathrm{H}}]$ for $n = 1$, first note that, according to (19) and (20), $[\lambda_{\mathrm{G}}] = [\lambda_{\mathrm{H}}] = [\varphi'']$ in this case. Moreover, for $n = 1$, we have $[\lambda_1] = [0, 0] = [Z, Z] = [y_1'']$ and, according to (23) and (24), $[\Lambda_s]([y_i']) = [y_i']^2 = [y_i']\,[y_i']^T$ and $[\Lambda_t]([y_i'], [y_j']) = 2\,[y_i']\,[y_j']$. Therefore, $[\lambda_k] = [y_k'']$ for every $k = 1, \ldots, 1 + t$ which yields $[\lambda_{\mathrm{A}}] = [\lambda_\varphi] = [\varphi'']$.

[3] Ordering is as in the GAMS code provided in the COCONUT library.

Table 5: Excerpt of the set of examples taken from the COCONUT-benchmark.

name	n	function φ
`ex8_1_6-1`	2	$\frac{1}{(x_1-4)^2+(x_2-4)^2+0.1} + \frac{1}{(x_1-1)^2+(x_2-1)^2+0.2} + \frac{1}{(x_1-8)^2+(x_2-8)^2+0.2}$
`ex7_2_6-2`	3	$1 - 0.01\,\frac{x_2}{x_3} - 0.01\,x_1 - 0.0005\,x_1\,x_3$
`ex14_2_2-6`	4	$10.208 - \frac{2755.642}{x_3+219.161} - \frac{0.192\,x_1}{x_1+0.192\,x_2} - \frac{x_2}{0.316\,x_1+x_2} - \ln(0.316\,x_1+x_2) + x_4$

We stress that we use the described set of test functions without further modifications. There exist functions in the collection treated here that contain convex terms, or terms for which tight convex under- or tight concave overestimators are known (e.g., bilinear, trilinear, linear fractional terms) [2,13] . Depending on the application it may be advisable to separate these terms from the given function φ, and to calculate Hessian eigenvalue bounds only for the remaining terms of φ. Here we choose not to apply any preprocessing for the sake of an unbiased comparison.

3.2 Evaluation of results

We introduce a simple rating scheme that assigns each result to one of a finite set of classes. Specifically, we distinguish the cases listed in Tab. 6, which reflect that a bound from the eigenvalue arithmetic may be

$(-)$	worse than the bounds from the other two methods,
$(\circ)$	equal to the one from Gershgorin's method, but equal to or worse than the one from Hertz and Rohn's method,
$(+)$	better than the one from Gershgorin's method, and equal to or worse than the one from Hertz and Rohn's method,
$(++)$	better than the one from Hertz and Rohn's method.

Note that bounds calculated with Hertz and Rohn's method are never worse than those from Gershgorin's circle criterion, since Hertz and Rohn's method provides the tight eigenvalue bounds for an interval matrix. Consequently, the bounds from Gershgorin's method do not play a role in our definition of the (++) category. Furthermore note that we do not distinguish between $\underline{\lambda}_{\mathrm{G}} = \underline{\lambda}_{\mathrm{H}}$ and $\underline{\lambda}_{\mathrm{G}} < \underline{\lambda}_{\mathrm{H}}$ (resp. $\overline{\lambda}_{\mathrm{G}} = \overline{\lambda}_{\mathrm{H}}$ and $\overline{\lambda}_{\mathrm{G}} > \overline{\lambda}_{\mathrm{H}}$) in the case $\underline{\lambda}_{\mathrm{A}} \leq \underline{\lambda}_{\mathrm{G}}$ (resp. $\overline{\lambda}_{\mathrm{A}} \geq \overline{\lambda}_{\mathrm{G}}$).

Table 6: Classes used to aggregate results in Sect. 3.3. Symbols $[\lambda_{\mathrm{A}}] = [\underline{\lambda}_{\mathrm{A}}, \overline{\lambda}_{\mathrm{A}}]$, $[\lambda_{\mathrm{G}}] = [\underline{\lambda}_{\mathrm{G}}, \overline{\lambda}_{\mathrm{G}}]$, and $[\lambda_{\mathrm{H}}] = [\underline{\lambda}_{\mathrm{H}}, \overline{\lambda}_{\mathrm{H}}]$ denote the eigenvalue bounds calculated with the eigenvalue arithmetic, Gershgorin's circle criterion, and Hertz and Rohn's method, respectively.

bound	class			
	$(-)$	$(\circ)$	$(+)$	$(++)$
upper ($\overline{\lambda}_{\mathrm{A}}$)	$\overline{\lambda}_{\mathrm{A}} > \overline{\lambda}_{\mathrm{G}} \geq \overline{\lambda}_{\mathrm{H}}$	$\overline{\lambda}_{\mathrm{A}} = \overline{\lambda}_{\mathrm{G}} \geq \overline{\lambda}_{\mathrm{H}}$	$\overline{\lambda}_{\mathrm{G}} > \overline{\lambda}_{\mathrm{A}} \geq \overline{\lambda}_{\mathrm{H}}$	$\overline{\lambda}_{\mathrm{G}} \geq \overline{\lambda}_{\mathrm{H}} > \overline{\lambda}_{\mathrm{A}}$
lower ($\underline{\lambda}_{\mathrm{A}}$)	$\underline{\lambda}_{\mathrm{A}} < \underline{\lambda}_{\mathrm{G}} \leq \underline{\lambda}_{\mathrm{H}}$	$\underline{\lambda}_{\mathrm{A}} = \underline{\lambda}_{\mathrm{G}} \leq \underline{\lambda}_{\mathrm{H}}$	$\underline{\lambda}_{\mathrm{G}} < \underline{\lambda}_{\mathrm{A}} \leq \underline{\lambda}_{\mathrm{H}}$	$\underline{\lambda}_{\mathrm{G}} \leq \underline{\lambda}_{\mathrm{H}} < \underline{\lambda}_{\mathrm{A}}$

Table 7 lists some numerical results. These examples illustrate that the classes introduced in Tab. 6 are meaningful. In particular it is evident that eigenvalue bounds of the same function may fall into different classes for different hyperrectangles B.

Table 7: Illustration of the classes $(-)$, $(\circ)$, $(+)$, $(++)$ introduced in Tab. 6.

example		`illustrative-1`		`illustrative-2`	
φ		$\exp(x_1 - 2\,x_2^2 + 3\,x_3^3)$		$\frac{x_1}{x_1+0.2\,x_2^2} - 2\,\frac{x_2}{x_2+0.3\,x_3^3}$	
B	$[\underline{x}_1, \overline{x}_1]$	$[-0.3, 0.2]$	$[-0.198, 0.177]$	$[1.043, 1.535]$	$[1.5, 1.6]$
	$[\underline{x}_2, \overline{x}_2]$	$[-0.1, 0.6]$	$[-0.473, 0.2]$	$[0.6, 1.969]$	$[0.6, 1.1]$
	$[\underline{x}_3, \overline{x}_3]$	$[-0.4, 0.5]$	$[-0.392, 0.39]$	$[0.555, 0.772]$	$[1.0, 1.6]$
A	$[\underline{\lambda}_\mathrm{A}, \overline{\lambda}_\mathrm{A}]$	$[-19.904, 37.004]$	$[-15.767, 19.27]$	$[-43.934, 27.391]$	$[-45.014, 17.624]$
G	$[\underline{\lambda}_\mathrm{G}, \overline{\lambda}_\mathrm{G}]$	$[-26.391, 38.587]$	$[-15.767, 18.443]$	$[-44.907, 27.391]$	$[-40.725, 19.507]$
H	$[\underline{\lambda}_\mathrm{H}, \overline{\lambda}_\mathrm{H}]$	$[-20.597, 29.603]$	$[-12.603, 14.278]$	$[-34.743, 26.399]$	$[-33.691, 18.897]$
class	$\overline{\lambda}_\mathrm{A}$	$(+)$	$(-)$	$(\circ)$	$(++)$
	$\underline{\lambda}_\mathrm{A}$	$(++)$	$(\circ)$	$(+)$	$(-)$

3.3 Results

Table 8 summarizes the results obtained for the 1522 sample functions[4]. The numbers listed in the columns labeled $\underline{\lambda}_\mathrm{A}$ state for how many of 100 randomly generated hyperrectangles the lower bounds calculated with the three methods fall into the classes $(-)$, $(\circ)$, $(+)$ and $(++)$ defined in Tab. 6. The numbers listed in the columns labeled $\overline{\lambda}_\mathrm{A}$ state the corresponding results for the upper bounds.

The examples are ranked in Tab. 8 by, loosely speaking, the quality of the bounds found with the eigenvalue arithmetic. More precisely, the higher an example is ranked, the higher the sum of the figures in its two $(++)$ columns. If this sum is equal for several examples, they are sorted according to the sum of the figures in their two $(+)$ columns. Subsequently, the sums of the two $(\circ)$ columns and the two $(-)$ columns are used for the ranking whenever necessary.

Table 8 shows the 20 best and 5 worst rated examples and characteristic ranks in between. Ranks 213−209, for example, are shown, because they mark the boundary between those φ for which some eigenvalue bounds still fall into class $(++)$, and the highest ranking examples for which class $(++)$ no longer occurs. These transitions in the ranking are marked with horizontal lines and shaded areas.

Just as for the illustrative examples given in Tab. 7, all classes $(-)$, $(\circ)$, $(+)$, $(++)$ occur for the sample functions from the COCONUT collection. In particular there exist cases for which the eigenvalue arithmetic provides tighter bounds than the tight bounds for the interval Hessian. An analysis of the data given in Tab. 8 reveals that 2.50% and 5.52% of the examples belong to the $(++)$ class for the lower bound and upper eigenvalue bound, respectively. Furthermore, in 12.48% of the cases the eigenvalue arithmetic provides tighter lower bounds than Gershgorin's circle criterion applied to the interval Hessian matrix. In 10.63% of the cases the upper bound from the eigenvalue arithmetic is tighter than the upper Gershgorin bound. In 60.85% (60.33%) of the cases the lower (upper) bounds from the eigenvalue arithmetic and those from Gershgorin's circle criterion are equal. We stress, however, that the Gershgorin bounds outperform those of the eigenvalue arithmetic in 24.17% (lower bound) and 23.52% (upper bound) of the cases, respectively. Overall, there exist 40 examples for which the eigenvalue arithmetic provides better bounds than

[4] All results were obtained with Jcodegen, a code generator written by the authors available on www.rus.rub.de/software/jcodegen for non-commercial use. Given a function φ, Jcodegen can generate ANSI C-code for the calculation of $[\nabla\varphi]$, $[\lambda_A]$, $[\nabla^2\varphi]$, etc., where the hyperrectangle B is a parameter to be passed at time of execution. Generated code needs to be linked to a simple IA implementation provided with Jcodegen.

Table 8: Summary of results for the 1522 sample functions extracted from the COCONUT collection. For each example, lower and upper bounds on Hessian matrix eigenvalues were calculated with the methods introduced in Sect. 2 for 100 random hyperrectangles. Numbers state for how many out of the 100 random hyperrectangles the bounds belong to the classes $(-)$, $(\circ)$, $(+)$ and $(++)$ defined in Tab. 6. Shaded cells highlight empty classes. Horizontal lines divide characteristic groups, e.g. examples with an empty class $(++)$ (rank $212 \le r \le 854$). The averages listed in the last row take all 1522 examples into account, including the ones not shown here.

example			$\underline{\lambda}_{\mathrm{A}}$				$\overline{\lambda}_{\mathrm{A}}$			
r	name	n	$(-)$	$(\circ)$	$(+)$	$(++)$	$(-)$	$(\circ)$	$(+)$	$(++)$
1	`box3-1`	3	2	50	32	16	0	0	2	98
2	`box2-1`	3	0	56	26	18	0	0	6	94
3	`cliff-1`	2	22	54	1	23	0	10	1	89
4	`chaconn1-1`	3	29	45	0	26	0	14	0	86
5	`chaconn2-1`	3	19	63	0	18	0	11	0	89
6	`cb3-1`	3	18	62	0	20	0	14	0	86
7	`polak6-1`	5	0	0	100	0	0	0	0	100
8	`polak6-2`	5	0	0	100	0	0	0	0	100
9	`polak6-3`	5	0	0	100	0	0	0	0	100
10	`polak6-4`	5	0	0	100	0	0	0	0	100
11	`growth-1`	3	0	0	96	4	0	0	4	96
12	`alsotame-1`	2	0	0	100	0	0	0	0	100
13	`vardim-1`	10	0	0	0	100	0	0	100	0
14	`vardim-2`	10	0	0	100	0	0	0	0	100
15	`alsotame-2`	2	0	0	100	0	0	0	0	100
16	`brownden-1`	4	1	0	99	0	0	0	0	100
17	`price-1`	2	3	0	97	0	0	0	0	100
18	`vanderm1-10`	10	75	0	25	0	0	0	0	100
19	`ex8_1_7-1`	5	99	0	1	0	0	0	0	100
20	`hs026-2`	3	0	100	0	0	0	0	0	100
⋮										
209	`ex14_1_7-5`	10	4	0	95	1	25	0	75	0
210	`ex14_1_7-9`	10	25	0	75	0	4	0	95	1
211	`nonmsqrt-1`	9	100	0	0	0	96	0	3	1
212	`brkmcc-1`	2	0	0	100	0	0	0	100	0
213	`ship-15`	10	1	0	99	0	1	0	99	0
⋮						⋮				⋮
852	`butcher-4`	7	100	0	0	0	99	0	1	0
853	`i5-3`	10	100	0	0	0	99	0	1	0
854	`cohn3-1`	4	100	0	0	0	99	0	1	0
855	`ex4_1_8-1`	2	0	100	0	0	0	100	0	0
856	`sample-3`	4	0	100	0	0	0	100	0	0
⋮					⋮	⋮			⋮	⋮
1391	`womflet-1`	3	100	0	0	0	97	3	0	0
1392	`reimer5-2`	5	98	2	0	0	99	1	0	0
1393	`reimer5-5`	5	98	2	0	0	99	1	0	0
1394	`ex7_2_9-4`	10	100	0	0	0	100	0	0	0
1395	`ex7_2_9-2`	10	100	0	0	0	100	0	0	0
⋮			⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1518	`cohn2-2`	4	100	0	0	0	100	0	0	0
1519	`cohn2-3`	4	100	0	0	0	100	0	0	0
1520	`cohn2-4`	4	100	0	0	0	100	0	0	0
1521	`boon-2`	6	100	0	0	0	100	0	0	0
1522	`boon-4`	6	100	0	0	0	100	0	0	0
arithmetic average			24.17	60.85	12.48	2.50	23.52	60.33	10.63	5.52

Gershgorin's circle criterion for all random boxes (e.g. $7 \leq r \leq 15$ or $r = 212$). On the other hand, there exist 129 examples for which the arithmetic results in less tight spectral bounds for all random boxes ($r \geq 1394$). For 854 out of the 1522 examples (i.e., 56.11%) the eigenvalue arithmetic provides tighter bounds than Gershgorin for at least one of the random boxes.

One of the anonymous reviewers suggested to compare the three methods as a function of the size of the hyperrectangles. In particular, sequences of hyperrectangles that converge to a point in $\mathbb{R}^n$ are of interest, because bisection algorithms may result in a large number of small boxes if first order interval extensions are used (a phenomenon known as the cluster problem, see, e.g., [6]). Hertz and Rohn's method, if applicable despite its computational complexity, will converge to tight bounds in the limit of hyperrectangles that collapse to a point. Neither the eigenvalue arithmetic nor the method based on Gershgorin's circle criterion will in general converge in this sense[5]. A closer comparison of Gershgorin's circle criterion and the eigenvalue arithmetic would be interesting but is beyond the paper.

Finally, we note that the ranking r does not correlate with the dimension n, i.e., we find both low and high values of n in any part of the ranking shown in Tab. 8. The dependency on n is analyzed in more detail at the end of this section with Tab. 10.

We discussed in Sect. 2.3 that the methods do not only differ with respect to the tightness of eigenvalue bounds, but also with respect to their computational complexity. Table 9 lists the numbers of operations necessary to evaluate the eigenvalue bounds for the examples from Tab. 8. All figures in Tab. 9 are based on the total number of operations needed for the respective method. Specifically, $N_{\mathrm{A}}(\varphi)$ denotes the total number of operations for calculating $[\lambda_{\mathrm{A}}]$ with the eigenvalue arithmetic, including the operations for the intermediate results $[y_k]$ and $[y'_k]$. $N_{\mathrm{G}}(\varphi)$ denotes the total number of operations for calculating $[\lambda_{\mathrm{G}}]$ with Gershgorin's circle criterion and forward automatic differentiation, including the operations for the intermediate results $[y_k]$, $[y'_k]$ and $[y''_k]$. $N_{\mathrm{bG}}(\varphi)$ refers to the same figure but backward automatic differentiation. Note again that all interval Hessians are calculated with forward mode automatic differentiation, but we list the numbers of operations for both the forward and backward mode. This results in a comparison in favour of the methods based on interval Hessians; see the explanation at the end of Sect. 2.

We also list $N_{\Delta\mathrm{A}}(\varphi)$ defined in (36), i.e. the additional effort to calculate $[\lambda_{\mathrm{A}}]$, if $[\lambda_{\mathrm{G}}]$ and its intermediate results $[y_k]$ and $[y'_k]$ have been determined. Note that $N_{\Delta\mathrm{A}}(\varphi)$ is independent of the mode of automatic differentiation. As predicted by relations (35), the eigenvalue arithmetic always requires fewer operations than the interval variant of Gershgorin's circle criterion. This result holds regardless of the mode of automatic differentiation. When forward mode is used, the computational effort for the eigenvalue arithmetic on average amounts to 39.17% of that of applying Gershgorin's circle criterion to the interval Hessian, where values range from 10.45% (example $r = 18$ in Tab. 9) to 60.52% (example $r = 161$). These figures read 43.30%, 22.86% (example $r = 1455$), and 51.87% (example $r = 1498$), respectively, for backward mode AD. The relative additional effort $N_{\Delta\mathrm{A}}(\varphi)/N_{\mathrm{G}}(\varphi)$ is 18.29% on average, with a minimum of 3.31% and maximum of 30.49%. The respective mean reads 20.14%, with a minimum of 6.13% and maximum of 26.94% for $N_{\Delta\mathrm{A}}(\varphi)/N_{\mathrm{bG}}(\varphi)$, i.e., the case where backward mode AD is used.

$N_{\mathrm{A}}(\varphi)$, $N_{\Delta\mathrm{A}}(\varphi)$, $N_{\mathrm{G}}(\varphi)$, and $N_{\mathrm{bG}}(\varphi)$ do not depend on the particular hyperrectangle B, but can be determined for any function φ before eigenvalue bounds are actually calculated. It may therefore be an option to determine these operation counts beforehand and to decide which method to use. This may be of interest in applications in which eigenvalue bounds

[5] Recall Gershgorin's circle criterion does in general not provide tight bounds for a real, i.e. non-interval, matrix.

Table 9: Computational effort for the evaluation of eigenvalue bounds for the examples taken from the COCONUT-benchmark. $N_{\mathrm{A}}(\varphi)$ and $N_{\mathrm{G}}(\varphi)$ denote the total number of operations necessary to calculate $[\lambda_{\mathrm{A}}]$ and $[\lambda_{\mathrm{G}}]$, respectively. $N_{\Delta\mathrm{A}}(\varphi)$ denotes the additional number of operations needed to calculate $[\lambda_{\mathrm{A}}]$ assuming the Gershgorin bounds have already been computed. The examples are listed in the same order as in Tab. 8. Shaded cells highlight minima and maxima of $\frac{N_{\mathrm{A}}(\varphi)}{N_{\mathrm{G}}(\varphi)}$ and $\frac{N_{\Delta\mathrm{A}}(\varphi)}{N_{\mathrm{G}}(\varphi)}$. The averages stated in the last row take all 1522 examples into account, including those not shown here.

example			abs. complexity				rel. complexity (%)			
r	name	n	N_{A}	$N_{\Delta\mathrm{A}}$	N_{G}	N_{bG}	$\frac{N_{\mathrm{A}}}{N_{\mathrm{G}}}$	$\frac{N_{\Delta\mathrm{A}}}{N_{\mathrm{G}}}$	$\frac{N_{\mathrm{A}}}{N_{\mathrm{bG}}}$	$\frac{N_{\Delta\mathrm{A}}}{N_{\mathrm{bG}}}$
1	`box3-1`	3	4584	1616	11584	12784	39.57	13.95	35.86	12.64
2	`box2-1`	3	4584	1616	11584	12784	39.57	13.95	35.86	12.64
3	`cliff-1`	2	318	112	554	806	57.40	20.22	39.45	13.90
4	`chaconn1-1`	3	174	58	428	488	40.65	13.55	35.66	11.89
5	`chaconn2-1`	3	174	58	428	488	40.65	13.55	35.66	11.89
6	`cb3-1`	3	174	58	428	488	40.65	13.55	35.66	11.89
7	`polak6-1`	5	2604	912	10896	9112	23.90	8.37	28.58	10.01
8	`polak6-2`	5	2600	912	10892	9068	23.87	8.37	28.67	10.06
9	`polak6-3`	5	2604	912	10896	9112	23.90	8.37	28.58	10.01
10	`polak6-4`	5	2604	912	10896	9112	23.90	8.37	28.58	10.01
11	`growth-1`	3	6940	3092	16208	16040	42.82	19.08	43.27	19.28
12	`alsotame-1`	2	118	46	216	288	54.63	21.30	40.97	15.97
13	`vardim-1`	10	5656	1442	41848	21794	13.52	3.45	25.95	6.62
14	`vardim-2`	10	6236	1490	45020	24326	13.85	3.31	25.64	6.13
15	`alsotame-2`	2	122	46	220	308	55.45	20.91	39.61	14.94
16	`brownden-1`	4	11392	3792	37120	34992	30.69	10.22	32.56	10.84
17	`price-1`	2	834	384	1502	1890	55.53	25.57	44.13	20.32
18	`vanderm1-10`	10	36316	13236	347420	153800	10.45	3.81	23.61	8.61
19	`ex8_1_7-1`	5	1364	458	5678	4746	24.02	8.07	28.74	9.65
20	`hs026-2`	3	396	150	1004	1152	39.44	14.94	34.38	13.02
⋮										
161	`gold-1`	2	1922	894	3176	3956	60.52	28.15	48.58	22.60
⋮										
1452	`desc.f.-2`	2	828	444	1456	1648	56.87	30.49	50.24	26.94
⋮										
1455	`more10-1`	10	4000	1300	37940	17500	10.54	3.43	22.86	7.43
⋮										
1498	`ex2-1`	2	360	186	630	694	57.14	29.52	51.87	26.80
⋮										
1518	`cohn2-2`	4	6240	3028	18468	14092	33.79	16.40	44.28	21.49
1519	`cohn2-3`	4	6240	3028	18468	14092	33.79	16.40	44.28	21.49
1520	`cohn2-4`	4	10122	4926	29044	22028	34.85	16.96	45.95	22.36
1521	`boon-2`	6	1424	700	6360	3604	22.39	11.01	39.51	19.42
1522	`boon-4`	6	1424	700	6360	3604	22.39	11.01	39.51	19.42
arithmetic average			*irrelevant*				39.17	18.29	43.30	20.14

need to be calculated for the same function φ for many B such as branch-and-bound global optimization.

While we did not recognize a dependency of the tightness of the bounds on n, the ratios $N_{\mathrm{A}}(\varphi)/N_{\mathrm{G}}(\varphi)$ and $N_{\Delta\mathrm{A}}(\varphi)/N_{\mathrm{G}}(\varphi)$ associated with forward mode automatic differentiation clearly depend on n. Table 10 shows that the relative number of operations for the eigenvalue arithmetic $N_{\mathrm{A}}(\varphi)/N_{\mathrm{G}}(\varphi)$ improves from about 56% for $n = 2$ to about 14% for $n = 10$. Similarly, the additional effort for the eigenvalue arithmetic decreases from about 25% for $n = 2$ to about 6% for $n = 10$. Note that examples with $n = 1$ would yield $N_{\mathrm{A}}(\varphi)/N_{\mathrm{G}}(\varphi) = 1 = 100\%$. Equations (29) and (30) predict a weaker dimensional dependency for the corresponding ratios $N_{\mathrm{A}}(\varphi)/N_{\mathrm{bG}}(\varphi)$ and $N_{\Delta\mathrm{A}}(\varphi)/N_{\mathrm{bG}}(\varphi)$. This is confirmed by the numerical experiments. In fact, Tab. 10 shows that $N_{\mathrm{A}}(\varphi)/N_{\mathrm{bG}}(\varphi)$ ($N_{\Delta\mathrm{A}}(\varphi)/N_{\mathrm{bG}}(\varphi)$) decreases from about 44% (20%) for $n = 2$ to about 35% (15%) for $n = 10$. Finally, we note that the large number of examples φ for $n = 3$ results from the COCONUT optimization problem `oet2`. We did not omit any of these φ in order not to introduce bias.

Table 10: Results by dimension n.

	num. of	mean: (%)				mean: (%)			
n	examples	$(-)$	$(\circ)$	$(+)$	$(++)$	$\frac{N_{\mathrm{A}}}{N_{\mathrm{G}}}$	$\frac{N_{\Delta\mathrm{A}}}{N_{\mathrm{G}}}$	$\frac{N_{\mathrm{A}}}{N_{\mathrm{bG}}}$	$\frac{N_{\Delta\mathrm{A}}}{N_{\mathrm{bG}}}$
2	62	57.89	15.47	14.68	11.97	55.55	25.29	43.54	19.90
3	1078	10.88	79.32	8.79	1.01	44.08	20.96	46.17	21.98
4	67	61.29	19.34	8.45	10.92	31.74	14.18	37.84	17.00
5	88	56.86	15.85	12.48	14.81	25.53	10.68	34.58	14.61
6	95	35.05	14.21	36.88	13.86	23.18	9.42	34.53	14.16
7	27	65.81	34.17	0.02	0.00	19.08	7.76	33.54	13.80
8	15	94.23	4.50	1.27	0.00	17.75	7.97	37.51	16.89
9	24	65.60	4.21	18.71	11.48	14.71	6.32	33.35	14.46
10	66	57.06	9.34	23.74	9.86	14.32	6.06	34.87	14.90
all	1522	23.84	60.59	11.55	4.01	39.17	18.29	43.30	20.14

4 Conclusion and Outlook

Our numerical experiments corroborate that the eigenvalue arithmetic always requires fewer operations than applying Gershgorin's circle criterion to the interval Hessian matrix. This result also holds if the interval Hessian is obtained with backward instead of forward mode AD. While this result has been established by comparing the complexity classes of the methods (see Sect. 2.3 and [16]), it was analyzed quantitatively with a large set of examples here for the first time. Specifically, 10.45% (example $r = 18$) to 60.52% (example $r = 161$) of the number of operations of the Gershgorin based approach are necessary for the eigenvalue method. The average over all examples amounts to 39.17% for the forward mode and 43.30% for the backward mode variant. As anticipated in the complexity analysis in Sect. 2.3, the eigenvalue method benefits from its favorable complexity as n increases (see Tab. 9 for details). We recall that a comparison to the computational effort of Hertz and Rohn's method is not reasonable, since Hertz and Rohn's method belongs to a very different complexity class (see Sect. 2.3).

While we determined the numbers of operations for both the forward and backward mode variants of Gershgorin's circle criterion, we calculated all interval Hessian matrices only with forward mode automatic differentiation (see the end of Sect. 2 for an explanation.) Gershgorin's circle criterion provides tighter lower (upper) bounds in 24.17% (23.52%) of the examples. In 60.85% (60.33%) of the cases the lower (upper) bounds from both methods are equal. In 14.98% (16.15%) of the examples the eigenvalue arithmetic results in tighter lower (upper) bounds than the Gershgorin based approach. Finally, our tests reveal that the number of cases in which the eigenvalue arithmetic results in tighter bounds than the tight bounds of the interval Hessian, which are obtained with Hertz and Rohn's method, is small (2.50% and 5.52% for lower and upper bounds, respectively). On the other hand, these figures indicate that these cases are not anecdotal or constructed, but they appear in global optimization problems.

The eigenvalue arithmetic provides a tighter lower or upper bound than Gershgorin's circle criterion for at least one random box in 56.11% of the examples, where these occurrences are not correlated with n. This figure suggests to combine the two methods. We claim the eigenvalue arithmetic can be applied at an attractive additional cost for, say, $n > 5$, whenever the Gershgorin bounds have already been calculated, since both methods involve the same intermediate quantities ($[y_k]$ and $[y'_k]$, see Sect. 2). Specifically, the additional effort for applying the eigenvalue method after the intermediate quantities have been calculated in the Gershgorin based approach ranges from 25% (20%) for $n = 2$ to about 6% (15%) for $n = 10$ using forward (backward) mode automatic differentiation (see Tab. 10). These figures decreases for increasing n as anticipated from the abstract complexity analysis in Sect. 2.3. Note that this combination of Gershgorin's circle criterion and the eigenvalue method will provide tighter bounds than Hertz and Rohn's method for the interval Hessian whenever the eigenvalue method does.

Acknowledgment

Funding by Deutsche Forschungsgemeinschaft is gratefully acknowledged (MO 1086/9-1).

References

[1] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice-differentiabe constrained NLPs-II. Implementation and computational results. *Computers and Chemical Engineering*, 22(9):1159–1179, 1998.

[2] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs-I. Theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.

[3] I.P. Androulakis, C.D. Maranas, and C.A. Floudas. αbb: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambrige University Press, 2004.

[5] G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, editors. *Automatic Differentiation of Algorithms: From Simulation to Optimization.* Springer, 2002.

[6] K. Du and R.B. Kearfott. The cluster problem in multivariate global optimization. *Journal of Global Optimization*, 5(3):253–265, 1994.

[7] H. Fischer. Automatisches Differenzieren. In J. Herzberger, editor, *Wissenschaftliches Rechnen: eine Einführung in das Scientific Computing*, pages 53–103. Akademie Verlag Berlin, 1995.

[8] S. Gershgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk SSSR, Ser. fizmat.*, 6:749–754, 1931.

[9] D. Hertz. The extreme eigenvalues and stability of real symmetric interval matrices. *IEEE Transactions on Automatic Control*, 37:532–535, 1992.

[10] M. Hladík, D. Daney, and E. Tsigaridas. Bounds on real eigenvalues and singular values of interval matrices. *SIAM J. Matrix Anal. Appl.*, 31(4):2116–2129, 2010.

[11] C.D. Maranas and C.A. Floudas. A deterministic global optimization approach for molecular-structure determination. *Journal of Chemical Physics*, 100(2):1247–1261, 1994.

[12] C.D. Maranas and C.A. Floudas. Global minimum potential energy conformations of small molecules. *Journal of Global Optimization*, 4(2):135–170, 1994.

[13] G.P. McCormick. Computability of global solutions of factorable nonconvex programs – 1 convex understimating problems. *Mathematical Programming*, 10(2):147–175, 1976.

[14] M. Mönnigmann. Efficient calculation of bounds on spectra of Hessian matrices. *SIAM Journal on Scientific Computing*, 30:2340–2357, 2008.

[15] M. Mönnigmann. Positive invariance tests with efficient Hessian matrix eigenvalue bounds. In *Proc. of 17th IFAC World Congress*, 2008.

[16] M. Mönnigmann. Fast Calculation of Spectral Bounds for Hessian Matrices on Hyperrectangles. *SIAM Journal on Scientific Computing*, 2011.

[17] A. Neumaier. *Interval Methods for Systems of Equations.* Encyclopedia of Mathematics and ist Applications. Cambrige University Press, 2008.

[18] L. B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science.* Springer, Berlin, 1981.

[19] R.T. Rockafellar. *Convex Analysis.* Princeton University Press, 1997.

[20] J. Rohn. Positive definiteness and stability of interval matrices. *SIAM Journal on Matrix Analysis and Applications*, 15(1):175–184, 1994.

[21] H. Schichl and M.C. Markot. Algorithmic differentiation techniques for global optimization in the coconut environment. *Journal of Optimization Methods and Software*, 27(2):359–372, 2012.

[22] O. Shcherbina, A. Neumaier, D. Sam-Haroud, X.-H. Vu, and T.-V. Nguyen. Benchmarking global optimization and constraint satisfaction codes. In: C. Bliek, C. Jermann, and A. Neumaier (eds.), *Global Optimization and Constraint Satisfaction*, pp. 211–222. Springer, Berlin, 2003.

Appendix

Interval Hessians can be calculated by backward mode AD according to the following algorithm.

Algorithm 3: *[7] Assume φ is twice continuously differentiable on U and can be written as a codelist. Let $B \subset U$ be a hyperrectangle. Then, for all $x \in B$, we have $\varphi(x) \in [\varphi]$, $\nabla\varphi(x) \in [\nabla\varphi]$, and $\nabla^2\varphi(x) \in [\nabla^2\varphi]$, where $[\varphi]$, $[\nabla\varphi]$, and $[\nabla^2\varphi]$ are calculated by the following algorithm.*

1. *For $k = 1, \dots, n$, set $[y_k] = [\underline{x}_k, \overline{x}_k]$ and $[y'_k] = [e^{(k)}, e^{(k)}]$.*
2. *For $k = n+1, \dots, n+t$, calculate $[y_k]$ and $[y'_k]$ according to columns 3–4 of Tab. 1, respectively.*
3. *Set $[\varphi] = [y_{n+t}]$ and $[\nabla\varphi] = [y'_{n+t}]$.*
4. *For $d = 1, \dots, n$,*
   a) *set $[u] = [e^{2(n+t)}, e^{2(n+t)}]$.*
   b) *for $k = n+t, \dots, n+1$, update $[u_i]$, $[u_j]$, $[u_{l+i}]$ and $[u_{l+j}]$ with $l = n+t$ according to columns 3–4 of Tab. 11, respectively.*
   c) *for $k = n+t, \dots, n+1$, update $[u_{l+i}]$ and $[u_{l+j}]$ with $l = 0$ according to column 4 of Tab. 11, respectively.*
   d) *set the d-th row of $[\nabla^2\varphi]$ equal to $([u_1], \dots, [u_n])$.*

Table 3 states the number of operations $N(\mathcal{U}_k)$ needed in Alg. 3 for each codelist line type Φ_k. Remark 1 summarizes how these values for $N(\mathcal{U}_k)$ can be obtained.

Table 11: Rules for the update of $[u]$ associated with the k-th line of the codelist (16) [7].

op Φ_k	$[u_i]$, $[u_j]$	$[u_{l+i}]$, $[u_{l+j}]$
`add`	$-, -$	$[u_{l+i}] + [u_{l+k}]$, $[u_{l+j}] + [u_{l+k}]$
`mul`	$[u_i] + [(y'_j)_d]\,[u_{n+t+k}]$, $[u_j] + [(y'_i)_d]\,[u_{n+t+k}]$	$[u_{l+i}] + [y_j]\,[u_{l+k}]$, $[u_{l+j}] + [y_i]\,[u_{l+k}]$
`addConst`	$-, -$	$[u_{l+i}] + [u_{l+k}]$, $-$
`mulByConst`	$-, -$	$[u_{l+i}] + c\,[u_{l+k}]$, $-$
`powNat`	$[u_i] + (m-1)\,m\,[y_i]^{m-2}\,[(y'_i)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] + m\,[y_i]^{m-1}\,[u_{l+k}]$, $-$
`oneOver`	$[u_i] + 2\,[y_k]^3\,[(y'_i)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] - [y_k]^2\,[u_{l+k}]$, $-$
`square`	$[u_i] + 2\,[(y'_i)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] + 2\,[y_i]\,[u_{l+k}]$, $-$
`cube`	$[u_i] + 6\,[y_i]\,[(y'_i)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] + 3\,[y_i]^2\,[u_{l+k}]$, $-$
`sqrt`	$[u_i] - 0.25\,(1/[y_k]^3)\,[(y'_i)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] + 1/(2\,[y_k])\,[u_{l+k}]$, $-$
`exp`	$[u_i] + [(y'_k)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] + [y_k]\,[u_{l+k}]$, $-$
`ln`	$[u_i] - (1/[y_i])^2\,[(y'_i)_d]\,[u_{n+t+k}]$, $-$	$[u_{l+i}] + 1/[y_i]\,[u_{l+k}]$, $-$

Remark 1: *First note that steps 1-3 of Alg. 3 calculate the interval function value $[\varphi]$ and the interval gradient $[\nabla\varphi]$. The additional operations needed to calculate the interval Hessian $[\nabla^2\varphi]$ arise in step 4.*

We recall that in forward mode AD the second order derivative $[y''_k]$ is calculated for each line of the codelist. The number of operations needed for the forward AD Hessian calculation can be found by determining the additional number of operations needed for $[y''_k]$ in each line k, and summing over all lines of a codelist. Similarly, the number of operations *needed in the eigenvalue arithemetic can be found by determining the additional operations for* $[\lambda_k]$ *in each line k of the codelist, and summing over all lines. In order to compare the backward mode case to the other two cases, we need to assign a number of operations to each line k needed for the calculation* $[\nabla^2\varphi]$*. Unfortunately, there is no intermediate result in the backward mode that can be interpreted as easily as* $[y_k'']$ *or* $[\lambda_k]$*, but the operations are hidden in the updates in lines 4(b) and 4(c) of Alg. 3. Specifically, we have to evaluate the four updates of* $[u_i]$*,* $[u_j]$*,* $[u_{l+i}]$ *and* $[u_{l+j}]$ *in line 4(b) and the two updates of* $[u_{l+i}]$ *and* $[u_{l+j}]$ *in step 4(c). Let*

$$\mathcal{U}_k^d := \left\{ \begin{array}{ll} \textit{updates of } [u_i],\ [u_j],\ [u_{l+i}],\ \textit{and } [u_{l+j}] & \textit{according to 4(b),} \\ \textit{updates of } [u_{l+i}] \textit{ and } [u_{l+j}] & \textit{according to 4(c)} \end{array} \right\},$$

denote the updates for the k-th line of the codelist and the d-th row of the interval Hessian $[\nabla^2\varphi]$*, and let* $\mathcal{U}_k$ *denote the set of all updates for line k, i.e.* $\mathcal{U}_k = \cup_{d=1}^n \mathcal{U}_k^d$*. Then the number of operations required to carry out the updates* $\mathcal{U}_k$ *is given by* $N(\mathcal{U}_k) = \sum_{d=1}^n N(\mathcal{U}_k^d)$*. Without giving details we claim that it follows from Tab. 11 that* $N(\mathcal{U}_k^d)$ *does not depend on d, which implies* $N(\mathcal{U}_k) = n\, N(\mathcal{U}_k^d)$*. Table 3 lists* $N(\mathcal{U}_k)$ *for each codelist line type.*

Finally, we illustrate the calculation of $N(\mathcal{U}_k)$ with a specific example.

Example 5: *The following updates are required in a codelist line of type* `mul` *in lines 4(b) and 4(c) of Alg. 3:*

$$\mathcal{U}_k^d = \left\{ \begin{array}{ll} [u_i] \leftarrow [u_i] + [(y_j')_d]\,[u_{n+t+k}], & [u_j] \leftarrow [u_j] + [(y_i')_d]\,[u_{n+t+k}] \\ [u_{n+t+i}] \leftarrow [u_{n+t+i}] + [y_j]\,[u_{n+t+k}], & [u_{n+t+j}] \leftarrow [u_{n+t+j}] + [y_i]\,[u_{n+t+k}] \\ [u_i] \leftarrow [u_i] + [y_j]\,[u_k], & [u_j] \leftarrow [u_j] + [y_i]\,[u_k] \end{array} \right\}.$$

These updates require six interval additions and six interval multiplications. Each interval addition requires two additions of real numbers according to (7). Each interval multiplication requires eight operations, since four multiplications of real numbers are required according to (8) and four comparisons of two real numbers are necessary to find the minimum and maximum among the four products. Consequently, $N(\mathcal{U}_k^d) = 6 \cdot (2 + 8) = 60$ *and* $N(\mathcal{U}_k) = n\, N(\mathcal{U}_k^d) = 60\, n$*. This is the number of operations stated in the last column of Tab. 3.*