myjournal manuscript No.
(will be inserted by the editor)

Generalized interval arithmetic on compact matrix Lie groups

Hermann Schichl, Mihály Csaba Markót, Arnold Neumaier[⋆]

Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

Received: date / Revised version: date

Abstract We introduce a generalization of intervals over compact matrix Lie groups. Interval group elements are defined using a midpoint-radius representation. Furthermore, we define the respective group operations and the action of interval group elements on vector spaces. We developed structures and operations are implemented for the special case of orthogonal matrices in the matrix library of the COCONUT Environment.

Keywords. interval arithmetic, matrix Lie group, real orthogonal group

1 Introduction

Notation and basic definitions. $\mathbb{R}^+$ denotes the set of nonnegative real numbers. $\mathbb{I}$ denotes the set of real intervals.

Throughout the paper, the symbol G will denote an arbitrary compact matrix Lie group. The identity element of G is denoted by I. The set of generalized intervals in G (to be defined later in this paper) is denoted by $\mathbb{I}G$.

For $r \in \mathbb{R}^+$, the closed ball of radius r around the identity in G is denoted by U_r, and is defined as $U_r = \{g \in G \mid \|g - I\| \le r\}$, where $\|.\|$ is an appropriate G-invariant norm.

2 Generalized intervals on compact matrix Lie groups

We define intervals on G in a way similar to the midpoint-radius representation of circular intervals over the real numbers [4].

⋆ This research was supported by the Austrian Science Found (FWF) Grant Nr. P22239-N13.

Definition 1 *A* generalized interval on G *is defined as a pair* $\langle g, r\rangle$*, where* $g \in G$ *and* $r \in \mathbb{R}^+$*.* $\langle g, r\rangle$ *represents the set* $U_r g = \{hg \mid h \in U_r\}$*.*

Less formally, given a group element g and a closed ball of radius r around the identity, $\langle g, r\rangle$ is the translation of this ball by g.

In order to ensure numerical reliability while calculating with such generalized intervals on a computer, the midpoint g must always be an *exact* representation of the group element, and the radius r must always be given by a floating point number that is an upper bound of the intended exact radius.

The following lemmas formulate the key property needed to develop the product of two generalized intervals.

Lemma 1 *For any* $\langle g, r\rangle \in \mathbb{I}G$*,* $gU_rg^{-1} = U_r$*.*

Proof

$$\begin{aligned} gU_rg^{-1} &= \{ghg^{-1} \in G \mid h \in U_r\} = \\ &= \{ghg^{-1} \in G \mid \|h - I\| \le r\} = \\ &= \{ghg^{-1} \in G \mid \|g(h - I)g^{-1}\| \le r\} = \\ &= \{ghg^{-1} \in G \mid \|ghg^{-1} - I\| \le r\} = \\ &= \{k \in G \mid \|k - I\| \le r\} = \\ &= U_r. \end{aligned}$$

(The second to last equality is easily verified by checking mutual containment: to prove $\{ghg^{-1} \in G \mid \|ghg^{-1} - I\| \le r\} \subseteq \{k \in G \mid \|k - I\| \le r\}$, set $k := ghg^{-1}$ for any fixed h, and to prove the other direction, set $h := g^{-1}kg$ for any fixed k.) $\square$

Lemma 2 *For any* $r, s \in \mathbb{R}^+$*,* $U_rU_s \subseteq U_{rs+r+s}$*.*

Proof By definition,

$$U_rU_s = \{gh \in G \mid \|g - I\| \le r, \|h - I\| \le s\},$$

so we only need to give an upper bound of $\|gh - I\|$:

$$\begin{aligned} \|gh - I\| &= \|(g - I)(h - I) + (g - I) + (h - I)\| \le \\ &\le \|(g - I)\|\|(h - I)\| + \|g - I\| + \|h - I\| \le \\ &\le rs + r + s. \quad \square \end{aligned}$$

3 Product of generalized intervals

In this section we propose a method to compute the product of two generalized intervals on G in a numerically reliable way. Let $\langle g, r\rangle, \langle h, s\rangle \in \mathbb{I}G$. Using Lemma 1, their product can be written as

$$U_rgU_sh = U_rgU_sg^{-1}gh = U_rU_sgh,$$

that, by our requirement, needs to be bounded by a product $U_t k$ for some $\langle k, t\rangle \in \mathbb{I}G$. The idea we employ here is to compute an *exactly representable* group element k that is an approximation of gh (in exact arithmetic k could of course be the exact group product, since $gh \in G$), and accumulate all rounding errors into the radius term by bounding and combining it with $U_r U_s$.

A natural way to obtain an appropriate k is to compute gh with ordinary floating point arithmetic, and search a nearby representable group element. Then,

$$U_r U_s gh = U_r U_s ghk^{-1}k,$$

where $ghk^{-1} \approx I$ needs to be bounded by a neighborhood term U_f. This can be done by computing $\boldsymbol{F} := ghk^{-1} - I$ *using interval arithmetic in all operations*, (i.e., $\boldsymbol{F} \in \mathbb{I}^{n\times n}$) and setting $f := \sup\{\|\tilde{F}\| \text{mid} \tilde{F} \in \boldsymbol{F} \cap G\}$ or to any overestimation of that number.

Using the previously derived relations and Lemma 2, we obtain

$$U_r g U_s h = U_r U_s ghk^{-1}k \subseteq U_r U_s U_f k \subseteq U_{rs+r+s} U_f k \subseteq U_{(rs+r+s)f+rs+r+s+f} k.$$

Thus, the product we derived is

$$\langle g, r\rangle\langle h, s\rangle = \langle k, (rs + r + s)f + rs + r + s + f\rangle,$$

with $k \in G, f \in \mathbb{R}^+$ defined above.

3.1 Implementation issues for the real orthogonal group

We represent our stuff in VMTL, where interval calculations are supported, but use LAPACK whenever possible to carry out the floating point parts of the computations.

We discuss the implementation of generalized intervals on the real orthogonal group $O(n)$, i.e., on the set of real orthogonal matrices of dimension n.

3.1.1 Representing real orthogonal matrices. The most often used exact representation of orthogonal matrices on a computer is a product of n Householder matrices (also called elementary reflectors): $Q = H_1 \dots H_n$. Here each Householder matrix is defined as

$$H = I - \beta vv^T, \quad \text{with} \quad \beta = \frac{2}{v^T v},$$

thus, for each reflector the vector v and the constant β is stored. LAPACK also use this kind of representation (for more on the specific implementation, see [3]). In the present study we also follow this approach with a small modification (see Sec. 3.2): we store v as a vector of floating point numbers as usual, but we always compute β with interval arithmetic and store it as an interval (of small width), to ensure mathematical rigor when interval operations are done.

Note that the H_i matrices are almost never constructed in practice. When the product of Q with an other matrix needs to be computed (or, as a special case, when Q itself is needed), the elementary reflectors are applied one-by-one.

Let us given an (arbitrary) real matrix $C \in \mathbb{R}^{n\times n}$ and a reflector H specified as $I - \beta vv^T$ with $\beta \in \mathbb{R}, v \in \mathbb{R}^n$. Then the right-multiplication of C with H, i.e., $CH = C - \beta Cvv^T$ is computed in two steps: first, the matrix-vector multiplication $w := Cv$ is computed, then the rank-1 update $C := C - \beta wv^T$ is applied. Thus, applying a reflector H on a matrix can be carried out by $O(n^2)$ operation, therefore, using this method, the cost of multiplying with an orthogonal matrix (composed of n reflectors) is cubic. For more details on the computational issues, see, e.g., [2]). The products HC, CH^T, and H^TC can be computed in similar ways.

There is one more important feature of working with orthogonal matrices, that is used in LAPACK, and utilized also in our present implementation, namely, the factored form representation of the Householder matrices. We use the LQ factorization routine `dgelqf` of LAPACK to compute the factorization of the previous section, and this routine creates Q as a product $H_n \dots H_1$, where for each $k = 1, \dots, n,\ v_{1:k-1} = 0$ and $v_k = 1$, thus, only the elements $v_{k+1:n}$ are returned. Our orthogonal matrix class implementation is prepared to deal also with such factored form representations, in order to make the interface to the LAPACK calls more efficient, and to accelerate the application of Householder reflections for the interval-type calculations by skipping the leading zeros of such vectors.

Further improvements during the multiplication with an orthogonal matrix Q can be reached if we use a factored form representation (recall that in this case $Q = H_n \dots H_1$, where for each $k = 1, \dots n,\ v_{1:k-1} = 0$ and $v_k = 1$) and utilize the presence of nonzeros for the v_i vectors. In particular, whenever it is possible, it is preferred to apply Q from the right, so that the nonzero part of w and the submatrix of C to be updated is kept minimal during the multiplications with the elementary reflectors.

3.1.2 LQ decompositions. As it is well known, every $A \in \mathbb{R}^{n\times n}$ matrix has an LQ (resp., QR) decomposition, and if A is invertible, this factorization is unique, if we require that the diagonal elements of L are all positive. In other words, for all invertible $A \in \mathbb{R}^{n\times n}$, if A is factorized as $A = LQ$, then A can be decomposed also a MDQ, where M is lower triangular with all positive diagonals and D is diagonal with entries ± 1 in its diagonal. Since D is orthogonal ($D^TD = DD = I$), so does the product DQ. In the evaluation of the previous section an LQ decomposition is needed in such a way that L is almost the identity, so we expect it to have all positive diagonal entries. However, the LQ decomposition routine of LAPACK does not quarantee to return an L with such a property, so we have to extract the negative signs as above by using the MDQ decomposition (so that $M \approx I$), and treat DQ as the orthogonal component. The slight difficulty emerging here is that, due to the representation of Q, D cannot be multiplied explicitly with it.

We resolve this by storing the D matrix (as a vector of signs) next to the Householder vectors, and whenever the Householder reflectors are applied to compute a product, we always apply D as well. (Multiplication with D can also be done in $O(n^2)$ operations, so this extra computation casues no change in the overall order of complexity.)

3.2 Our representation and the respective interval calculations

According to the above discussion, an element $\boldsymbol{\mathcal{G}}$ of the orthogonal group for our calculations (including interval support) is implemented as a triplet $\boldsymbol{\mathcal{G}} = (d, V, \boldsymbol{b})$, where

- $d \in \{-1, 1\}^n$ is the respective scaling vector;
- $V \in \mathbb{R}^{n\times n}$ is the matrix of Householder coefficients; in particular, the vector associated with the ith Householder reflector, denoted by V_i is stored at the ith row of V. **Add** how the factored form representation looks like.
- $\boldsymbol{b} \in \mathbb{I}^n$ is the vector of the interval enclosures of the exact β values.

In contexts where multiple elements of the orthogonal groups are used, we will use the notation $d(\boldsymbol{\mathcal{G}})$, $V(\boldsymbol{\mathcal{G}})$, and $\boldsymbol{b}(\boldsymbol{\mathcal{G}})$, resp., to denote the components of $\boldsymbol{\mathcal{G}}$.

The ith Householder reflector of $\boldsymbol{\mathcal{G}} = (d, V, \boldsymbol{b})$, denoted by $\boldsymbol{R}_i(\boldsymbol{\mathcal{G}})$, can be calculated as $\boldsymbol{R}_i = I - \boldsymbol{b}_i \boldsymbol{V}_i^T \boldsymbol{V}_i$, where $\boldsymbol{V}_i = V_i$ is a thin interval vector. Note that actually we will never use this explicit calculation, similarly to the real case.

Applying a Householder reflector to an interval matrix. Given an orthogonal matrix $\boldsymbol{\mathcal{G}} = (d, V, \boldsymbol{b})$, and a matrix $\boldsymbol{C} \in \mathbb{I}^{n\times n}$, the product $\boldsymbol{C} \cdot \boldsymbol{R}_i(\boldsymbol{\mathcal{G}}) \in \mathbb{I}^{n\times n}$, i.e., the right multiplication of $\boldsymbol{C}$ with $\boldsymbol{R}_i(\boldsymbol{\mathcal{G}})$ is evaluated by first computing $\boldsymbol{w} := \boldsymbol{C}\boldsymbol{V}_i^T \in \mathbb{I}^n$, then doing the update $\boldsymbol{C} := \boldsymbol{C} - \boldsymbol{b}_i \boldsymbol{w} \boldsymbol{V}_i$. The products $\boldsymbol{R}_i(\boldsymbol{\mathcal{G}}) \cdot \boldsymbol{C}$, $\boldsymbol{C} \cdot \boldsymbol{R}_i(\boldsymbol{\mathcal{G}})^T$, and $\boldsymbol{R}_i(\boldsymbol{\mathcal{G}})^T \cdot \boldsymbol{C}$ are computed in similar ways.

Multiplying an interval matrix with an orthogonal matrix. Given an orthogonal matrix $\boldsymbol{\mathcal{G}} = (d, V, \boldsymbol{b})$, and a matrix $\boldsymbol{C} \in \mathbb{I}^{n\times n}$, the product $\boldsymbol{C} \cdot \boldsymbol{\mathcal{G}} \in \mathbb{I}^{n\times n}$, i.e., the right multiplication of $\boldsymbol{C}$ with $\boldsymbol{\mathcal{G}}$ is computed as

$$\boldsymbol{C} \cdot \boldsymbol{\mathcal{G}} = \boldsymbol{C} \cdot diag(d) \cdot \boldsymbol{R}_n(\boldsymbol{\mathcal{G}}) \cdot \ldots \cdot \boldsymbol{R}_1(\boldsymbol{\mathcal{G}}) = \boldsymbol{C} \cdot diag(d) \cdot \prod_{j=n}^{1} \boldsymbol{R}_j(\boldsymbol{\mathcal{G}}),$$

by applying $n+1$ successive right-multiplications. Similarly, we have

$$\boldsymbol{\mathcal{G}} \cdot \boldsymbol{C} = diag(d) \cdot \boldsymbol{R}_n(\boldsymbol{\mathcal{G}}) \cdot \ldots \cdot \boldsymbol{R}_1(\boldsymbol{\mathcal{G}}) \cdot \boldsymbol{C} = diag(d) \cdot \prod_{j=n}^{1} \boldsymbol{R}_j(\boldsymbol{\mathcal{G}}) \cdot \boldsymbol{C},$$

by applying $n+1$ successive left-multiplications.

As for multiplying with $\boldsymbol{\mathcal{G}}^T$, we have the formulas

$$\boldsymbol{C} \cdot \boldsymbol{\mathcal{G}}^T = \boldsymbol{C} \cdot \boldsymbol{R}_1(\boldsymbol{\mathcal{G}})^T \cdot \ldots \cdot \boldsymbol{R}_n(\boldsymbol{\mathcal{G}})^T \cdot diag(d) = \boldsymbol{C} \cdot \prod_{j=1}^{n} \boldsymbol{R}_j(\boldsymbol{\mathcal{G}}) \cdot diag(d),$$

with $n+1$ successive right-multiplications, and

$$\boldsymbol{\mathcal{G}}^T \cdot \boldsymbol{C} = \boldsymbol{R}_1(\boldsymbol{\mathcal{G}})^T \cdot \ldots \cdot \boldsymbol{R}_n(\boldsymbol{\mathcal{G}})^T \cdot diag(d) \cdot \boldsymbol{C} = \prod_{j=1}^{n} \boldsymbol{R}_j(\boldsymbol{\mathcal{G}}) \cdot diag(d) \cdot \boldsymbol{C},$$

with $n+1$ successive left-multiplications, respectively.

For instance, *a guaranteed interval enclosure of the exact value of* $\boldsymbol{\mathcal{G}}$ can be computed as $\boldsymbol{I} \cdot \boldsymbol{\mathcal{G}}$ (or as $\boldsymbol{\mathcal{G}} \cdot \boldsymbol{I}$).

Note, that when using factored form representations, V_k contains $k-1$ leading zeros ($k = 1, \ldots, n$), (thus V_n contains the least number of nonzero elements). That is, for the factored form representation, the right-multiplication with $\boldsymbol{\mathcal{G}}$ is more economic than the left-multiplication with $\boldsymbol{\mathcal{G}}$, and similarly, the left-multiplication with $\boldsymbol{\mathcal{G}}^T$ is more economic than the right-multiplication with $\boldsymbol{\mathcal{G}}^T$, due to the more efficient utilization of the occurring zeros.

3.3 Computing the triple interval matrix product

Next we investigate various ways of computing the reliable interval enclosure $\boldsymbol{A} \in \mathbb{I}^{n\times n}$ of the product $\boldsymbol{\mathcal{G}} \cdot \boldsymbol{\mathcal{H}} \cdot \boldsymbol{\mathcal{K}}^T$ of three orthogonal matrices. As noted above, the usual way of computing such a product for real orthogonal matrices is to apply the respective Householder reflections successively, in order to reduce the total computing effort, and to utilize the presence of nonzeros for factored form representation.

These two observations can be carried forward to the interval case, however, in addition to reducing the computation time, in the interval case we also have to take care of reducing the interval width of the result. (As we will se below, the two criteria will not necessarily coincide.)

First we introduce two straightforward ways of computing the triple product. Algorithm 1 just uses $3(n+1)$ successive multiplications with the Householder reflectors and with the scaling matrices, resp.:

Algorithm 1:

1. Compute $\boldsymbol{A} := \boldsymbol{\mathcal{K}}^T \cdot \boldsymbol{I}$. (Observe the left-multiplication in order to be more efficient for factored form representations.)
2. Compute the update $\boldsymbol{A} := \boldsymbol{\mathcal{H}} \cdot \boldsymbol{A}$.
3. Finally, compute the update $\boldsymbol{A} := \boldsymbol{\mathcal{G}} \cdot \boldsymbol{A}$.

Another approach, utilizing the factored form representation for all three components (for the price of additional explicit matrix-matrix multiplications), is the following:

Algorithm 2:

Table 1 Comparison of the three proposed algorithms for thin interval multiplication of two $n \times n$ matrices. The first group of column shows the 2-norm of the result. The second group of column shows the CPU time.

n	Alg 1	Alg 2	Alg 3	Alg 1	Alg 2	Alg 3
5	$7.57 \cdot 10^{-13}$	$3.88 \cdot 10^{-14}$	$1.29 \cdot 10^{-14}$	0.000	0.000	0.000
10	$1.44 \cdot 10^{-9}$	$9.90 \cdot 10^{-13}$	$4.71 \cdot 10^{-14}$	0.001	0.002	0.001
20	$4.03 \cdot 10^{-3}$	$2.82 \cdot 10^{-10}$	$1.68 \cdot 10^{-13}$	0.002	0.004	0.003
50	$5.11 \cdot 10^{17}$	$3.41 \cdot 10^{-3}$	$9.44 \cdot 10^{-13}$	0.013	0.028	0.037
100	$2.37 \cdot 10^{51}$	$5.30 \cdot 10^{20}$	$3.57 \cdot 10^{-12}$	0.076	0.167	0.301
200	$1.08 \cdot 10^{118}$	$2.29 \cdot 10^{87}$	$1.38 \cdot 10^{-11}$	0.535	1.176	4.126

1. Compute the interval matrices $\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{K}$ as $\boldsymbol{G} := \boldsymbol{I} \cdot \boldsymbol{\mathcal{G}}$, $\boldsymbol{H} := \boldsymbol{I} \cdot \boldsymbol{\mathcal{H}}$, and $\boldsymbol{K} := \boldsymbol{\mathcal{K}}^T \cdot \boldsymbol{I}$, resp.
2. Compute $\boldsymbol{A} := \boldsymbol{G} \cdot \boldsymbol{H} \cdot \boldsymbol{K}$ with explicit interval matrix-matrix multiplications.

The above facts motivated us to design an alternative method of a reliable computation of $\boldsymbol{\mathcal{G}} \cdot \boldsymbol{\mathcal{H}} \cdot \boldsymbol{\mathcal{K}}^T$, using midpoint-error representation at each Householder update. First we discuss the method *assuming exact arithmetic.*

Given a matrix $\boldsymbol{C} \in \mathbb{I}^{n\times n}$, decompose it into $\boldsymbol{C} = P + \boldsymbol{E}_1$, where $P \in \mathbb{R}^{n\times n}$ is the midpoint of $\boldsymbol{C}$ and $\boldsymbol{E}_1 \in \mathbb{I}^{n\times n}$ is the interval error matrix of $\boldsymbol{C}$, resp. Then a left Householder update $\boldsymbol{R} \cdot \boldsymbol{C} = \boldsymbol{R} \cdot (P + \boldsymbol{E}_1) = \boldsymbol{R} \cdot P + \boldsymbol{R} \cdot \boldsymbol{E}_1$ can be written as

$$\boldsymbol{R} \cdot \boldsymbol{C} = P' + \boldsymbol{E}_2 + \boldsymbol{R} \cdot \boldsymbol{E}_1,$$

where $P' \in \mathbb{R}^{n\times n}$ is the midpoint of $\boldsymbol{R} \cdot P$ (i.e., the interval product of an exact Householder reflector and a thin interval matrix) and $\boldsymbol{E}_2 \in \mathbb{I}^{n\times n}$ is the interval error matrix of $\boldsymbol{R} \cdot P$, resp. Thus $\boldsymbol{E}_2$ is the newly added error term after multiplying with $\boldsymbol{R}$. As we will see later, the error term $\boldsymbol{R} \cdot \boldsymbol{E}_1$ need not be computed explicitly for our purpose. Hence, we need to apply $\boldsymbol{R}$ only to P, which means that we will face only the smallest amount of interval overestimation for each left-multiplication update.

To formulate the left-multiplication of $C = P + \boldsymbol{E}_1$ with $diag(d)$ in a similar way, we can write $diag(d) \cdot \boldsymbol{C} = diag(d) \cdot \boldsymbol{P} + diag(d) \cdot \boldsymbol{E}_1 = \boldsymbol{P}' + \boldsymbol{E}_1$, that is,

$$diag(d) \cdot \boldsymbol{C} = P' + \boldsymbol{E}_2 + diag(d) \cdot \boldsymbol{E}_1,$$

with the midpoint matrix $P' = diag(d) \cdot \boldsymbol{P}$ and the new error term $\boldsymbol{E}_2 = 0$, resp.

Using this technique the product $\boldsymbol{\mathcal{G}} \cdot \boldsymbol{\mathcal{H}} \cdot \boldsymbol{\mathcal{K}}^T$ can be written as

$$\boldsymbol{\mathcal{G}} \cdot \boldsymbol{\mathcal{H}} \cdot \boldsymbol{\mathcal{K}}^T \cdot \boldsymbol{I} = \big(diag(d(\boldsymbol{\mathcal{G}})) \cdot \prod_{j=n}^{1} \boldsymbol{R}_j(\boldsymbol{\mathcal{G}})\big) \cdot$$

$$\cdot \big(diag(d(\boldsymbol{\mathcal{H}})) \cdot \prod_{j=n}^{1} \boldsymbol{R}_j(\boldsymbol{\mathcal{H}})\big) \cdot \big(\prod_{j=1}^{n} \boldsymbol{R}_j(\boldsymbol{\mathcal{K}}) \cdot diag(d(\boldsymbol{\mathcal{K}}))\big) \cdot \boldsymbol{I} =$$

$$= P + \sum_{i=0}^{3(n+1)} \boldsymbol{E}'_i,$$

where $P \in \mathbb{R}^{n\times n}$ is the midpoint of the interval matrix resulted after the $3(n+1)$ left-multiplications, and the error terms $\boldsymbol{E}'_i$ are given as

$$\boldsymbol{E}'_i = \prod_{k=i+1}^{3(n+1)} \boldsymbol{S}_k \boldsymbol{E}_i, \qquad i = 0, \dots, 3(n+1),$$

where $\boldsymbol{S}_k$ is a shorthand notation of either a Householder reflector $\boldsymbol{R}_j$ or a diagonal matrix $diag(d)$ of one of the three orthogonal matrices, and $\boldsymbol{E}_i$, $i = 1, \dots, 3(n+1)$ is the newly added error term after the ith left-multiplication. (At $i = 1$, $i = 2n+2$ and $i = 3n+3$ we have $E_i = 0$.) The term $\boldsymbol{E}_0$ denotes the initial 0 error term of the midpoint-error representation of I.

Lemma 3 *Let $S_i \in O(n)$ for an index set i. Then for any $\boldsymbol{A} \in \mathbb{I}^{n\times n}$ matrix, $||(\prod_i S_i)\boldsymbol{A}||_2 = ||\boldsymbol{A}||_2$.*

Proof Easy, both possible types of transformations (and their compositions) are orthogonal. □

Thus, we have

$$||\boldsymbol{\mathcal{G}} \cdot \boldsymbol{\mathcal{H}} \cdot \boldsymbol{\mathcal{K}}^T - \boldsymbol{I}||_2 = ||P + \sum_{i=0}^{3(n+1)} \boldsymbol{E}'_i - \boldsymbol{I}||_2 \le$$

$$||P-\boldsymbol{I}||_2 + ||\sum_{i=0}^{3(n+1)} \boldsymbol{E}'_i||_2 \le ||P-\boldsymbol{I}||_2 + \sum_{i=0}^{3(n+1)} ||\boldsymbol{E}'_i||_2 = ||P-\boldsymbol{I}||_2 + \sum_{i=0}^{3(n+1)} ||\boldsymbol{E}_i||_2.$$

For the computer implementation of the above computation only a few things need to be taken into account. First, notice that we have not actually used that the midpoints are the exact midpoints of the given interval matrices, hence the method remains valid with approximate midpoints as well. Furthermore, for the computed approximate midpoints we need a guaranteed enclosure of the respective exact error matrices, from which the upper bounds of the 2-norms of the exact error matrices are calculated. Then the overall upper bound will also be an upper bound of the exact $\sum_{i=0}^{3(n+1)} ||\boldsymbol{E}_i||_2$ term. In addition, of course, the application of the Householder reflections to the current (approximate) midpoint and the calculation of the term $||P-\boldsymbol{I}||_2$ also have to be done in interval way. The short algorithmic description of the presented method is the following:

Algorithm 3:

1. Set the current midpoint to $P = I$ and the sum of the 2-norm of the error terms to $\boldsymbol{a} = 0$.
2. For $i = 1, \dots, 3(n+1)$, iterate the steps 2.1. to 2.4.:

Table 2 Results of multiplying N thin interval orthogonal matrices (Lie group-based representation, 'L'), and N thin interval enclosures of approximately orthogonal matrices (standard interval matrix representation, 'S')

n	N	radius		runtime	
		L	S	L	S
10	10	4.69e−13	1.66e−11	0.013	0.001
	20	9.36e−13	3.03e−07	0.019	0.002
	30	1.40e−12	5.46e−03	0.022	0.003
	40	1.87e−12	8.91e+01	0.032	0.004
	50	2.33e−12	1.48e+06	0.038	0.005
50	10	9.39e−12	1.45e−07	0.401	0.032
	20	1.88e−11	6.34e+00	0.805	0.062
	30	2.82e−11	2.79e+08	1.211	0.100
	40	3.76e−11	1.23e+16	1.610	0.130
	50	4.71e−11	5.37e+23	2.012	0.163
200	10	1.38e−10	5.00e−04	39.610	1.626
	20	2.75e−10	2.17e+07	80.954	3.350
	30	4.13e−10	9.49e+17	118.833	5.160
	40	5.51e−10	4.11e+28	162.324	6.967
	50	6.88e−10	1.78e+39	200.499	8.741

```
  2.1. Let $\boldsymbol{S}_i$ be the $i$th factor of $\boldsymbol{\mathcal{G}} \cdot \boldsymbol{\mathcal{H}} \cdot \boldsymbol{\mathcal{K}}^T$ from the right.
  2.3. If $\boldsymbol{S}_i$ is a Householder reflector $\boldsymbol{R}$, then
    2.3.1. Compute $\boldsymbol{P} = \boldsymbol{R} \cdot P$ in interval way.
    2.3.1. Decompose $\boldsymbol{P}$ into the sum of a point $P$ and an error matrix $\boldsymbol{E}_i$.
    2.3.2. Set $\boldsymbol{a} = \boldsymbol{a} + ||\boldsymbol{E}_i||_2$.
  2.4. Otherwise, if $\boldsymbol{S}_i$ is a diagonal matrix $diag(d)$, then
    2.4.1 Set $P = diag(d) \cdot P$
3. Set $\boldsymbol{a} = \boldsymbol{a} + ||P - \boldsymbol{I}||_2$.
```

References

1. COCONUT Environment. Software, University of Vienna. `http://www.mat.univie.ac.at/coconut-environment`.
2. G. H. Golub and C. F. Van Loan, *Matrix Computations* (3rd ed.), Johns Hopkins Studies in Mathematical Sciences, 1996.
3. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney and D. Sorensen, *LAPACK Users' guide* (3rd ed.), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. `http://www.netlib.org/lapack/lug`.
4. H. Ratschek and J. Rokne, *Computer Methods for the Range of Functions*, Ellis Horwood, Chichester, 1984.