Computing the Volume of a Convex Polytope Using Interval Arithmetic *

Hoon Hong Franz Kriftner Volker Stahl

Research Institute for Symbolic Computation
Johannes Kepler University Linz
A-4040 Linz, Austria
Hoon.Hong@risc.uni-linz.ac.at
Franz.Kriftner@risc.uni-linz.ac.at
Volker.Stahl@risc.uni-linz.ac.at

Thomas Fahringer

Institute for Software Technology and Parallel Systems
University of Vienna
Bruenner Str. 72, A-1210 Vienna, Austria
tf@par.univie.ac.at

January 10, 1994

Abstract

We present a new algorithm for computing the volume of a convex polytope in a box. More precisely, given a set of linear inequalities and an n-dimensional interval, we compute the volume of the intersection of the interval and the common solution set of the inequalities. The output of the algorithm is exact up to some admissible error which is specified by the user.

1 Introduction

Computing the volume of convex polytopes is an important problem in geometric modeling, but arises also in completely different areas [4], [1], [2], [3].

*The research was sponsored by the Austrian Science Foundation FWF (S5302-PHY).

In this paper we assume that we are given a set of linear inequalities (constraints) whose common solution set forms a convex polytope $\mathcal{P}$. For an n-dimensional interval $\boldsymbol{X}$ and an admissible error $\varepsilon > 0$ we want to find an upper bound u and a lower bound l for the volume of $\mathcal{P} \cap \boldsymbol{X}$, such that $u - l \leq \varepsilon$.

We propose an interval arithmetic approach to solve this problem. Roughly, the idea is to enclose the polytope recursively by boxes. If for some box there exists at most one constraint, whose solution set does not fully cover the box, then the volume computation in that box is trivial. Similarly, if for some box there exists at least one constraint whose solution set is disjoint from the box, then the volume of the polytope in that box is zero. A box is split in such a way that one of these special cases is likely to arise for the sub boxes. Before splitting a box, it is reduced to the smallest sub box which has the same intersection with the polytope. The algorithm keeps track of an upper and a lower bound of the volume of $\mathcal{P} \cap \boldsymbol{X}$ during execution. If the bounds are closer than the admissible error, the algorithm terminates.

In Section 2 we state the problem formally and fix some notation. The general idea and an initial algorithm are given in Section 3, improvements are subject of Section 4. Finally, in Section 5 we mention some alternative approaches and directions for future research.

2 Problem Statement and Notation

Problem Statement. Let $f_i : \Re^n \to \Re$

$$f_i(x_1, \ldots, x_n) = \sum_{j=1}^{n} a_{ij} x_j + b_i, \quad i = 1, \ldots, m,$$

be linear functionals and let

$$\mathcal{P} = \bigcap_{i=1}^{m} \{\boldsymbol{x} \in \Re^n \mid f_i(\boldsymbol{x}) \geq 0\}. \tag{1}$$

Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a box in $\Re^n$, i.e. a real n-dimensional interval. Finally, let $\varepsilon > 0$ be an admissible error. We want to find u, l such that $u - l \leq \varepsilon$ and

$$l \leq \text{volume}(\mathcal{P} \cap \boldsymbol{X}) \leq u.$$

Notation. We use lower case letters for real numbers and functions, upper case letters for real intervals and boldface for vectors. The lower and upper endpoint of an interval X is denoted by $\underline{X}$ respectively $\overline{X}$. For the set theoretic difference of two intervals X, X' we write $X \backslash X'$.

3 General Idea and Initial Algorithm

We begin this section by discussing some special cases where the volume computation is trivial. Next, we show how to reduce the general case to such special cases. Finally, we give an initial algorithm which incorporates the main ideas.

There are several special cases when the volume of $\mathcal{P} \cap \boldsymbol{X}$ can be computed easily. For example, if $\boldsymbol{X} \subseteq \mathcal{P}$ or $\boldsymbol{X} \cap \mathcal{P} = \emptyset$, then the volume of $\mathcal{P} \cap \boldsymbol{X}$ is equal to the volume of $\boldsymbol{X}$ respectively zero. These cases can be tested for example by evaluating the f_i on $\boldsymbol{X}$ using interval arithmetic [8]: If $\underline{f_i(\boldsymbol{X})} \geq 0$ for all i, then $\boldsymbol{X} \subseteq \mathcal{P}$, if $\overline{f_i(\boldsymbol{X})} < 0$ for at least one i, then $\boldsymbol{X} \cap \mathcal{P} = \emptyset$.

Another special case where the volume computation is straight forward is a situation where at most one constraint is active in $\boldsymbol{X}$. We call a constraint f_i *active* in $\boldsymbol{X}$ if there exist $\boldsymbol{x}^+, \boldsymbol{x}^- \in \boldsymbol{X}$ such that $f_i(\boldsymbol{x}^+) > 0$ and $f_i(\boldsymbol{x}^-) < 0$. The geometric interpretation is that the line $f_i = 0$ crosses the interior of $\boldsymbol{X}$. If there is a unique active constraint in $\boldsymbol{X}$, the volume can be computed as follows: Let $\boldsymbol{X}'$ be the smallest sub box of $\boldsymbol{X}$ such that $\boldsymbol{X}$ and $\boldsymbol{X}'$ have the same intersection with the set $f_i = 0$. No constraint is active in $\boldsymbol{X} \backslash \boldsymbol{X}'$ and the volume of $\mathcal{P} \cap (\boldsymbol{X} \backslash \boldsymbol{X}')$ can be computed easily. Further, the volume of $\mathcal{P} \cap \boldsymbol{X}'$ is half of the volume of $\boldsymbol{X}'$. The computation of $\boldsymbol{X}'$ can be done by the formula

$$X'_{\hat{\jmath}} = \begin{cases} X_{\hat{\jmath}} \cap (-R_{i\hat{\jmath}}/a_{i\hat{\jmath}}) & \text{if } a_{i\hat{\jmath}} \neq 0 \\ X_{\hat{\jmath}} & \text{if } a_{i\hat{\jmath}} = 0 \text{ and } 0 \in R_{i\hat{\jmath}} \\ \emptyset & \text{if } a_{i\hat{\jmath}} = 0 \text{ and } 0 \notin R_{i\hat{\jmath}}, \end{cases} \tag{2}$$

where

$$R_{i\hat{\jmath}} = \sum_{j \neq \hat{\jmath}} a_{ij} X_j + b_i, \quad \hat{\jmath} = 1, \ldots, n.$$

One can imagine several other situations when the volume computation is simple, but in this paper we restrict our considerations to the cases described above.

The basic idea of the following algorithm is to reduce the given volume computation problem to one or several volume computation problems which can either be solved immediately or which are at least "easier" than the original problem. This reduction can be done in two ways:

Splitting. Split the box $\boldsymbol{X}$ into several sub boxes. If possible, the splitting should be done in such a way that the number of active constraints in the sub boxes is smaller than the number of active constraints in the original box. In many cases this can be achieved as follows: Choose an active constraint $f_i \geq 0$ and find the smallest sub box $\boldsymbol{X}'$ of $\boldsymbol{X}$ such that $\boldsymbol{X}$ and $\boldsymbol{X}'$ have the same intersection with the set $f_i = 0$, using (2). If there exists a face of $\boldsymbol{X}'$ which is not contained in any face of $\boldsymbol{X}$, then split $\boldsymbol{X}$ into two sub boxes, such that the sub boxes share that face of $\boldsymbol{X}'$. In this case f_i is active in only one of the sub boxes.

Tightening. The notion of tightening was introduced for nonlinear equation systems in [7]. Tightening means reducing the box $\boldsymbol{X}$ to a sub box $\boldsymbol{X}' \subseteq \boldsymbol{X}$ such that

$$\mathcal{P} \cap \boldsymbol{X} = \mathcal{P} \cap \boldsymbol{X}'. \tag{3}$$

Such a reduction can be achieved by choosing a constraint $f_{\hat{\imath}} \geq 0$ and a variable $x_{\hat{\jmath}}$ and "solving" $f_{\hat{\imath}} \geq 0$ for $x_{\hat{\jmath}}$. More formally, we define the tightening operation as

$$\boldsymbol{X}' = \text{tight}(\boldsymbol{X}, f_{\hat{\imath}}, x_{\hat{\jmath}})$$

where $X'_j = X_j$ for all $j \neq \hat{\jmath}$,

$$X'_{\hat{\jmath}} = \begin{cases} X_{\hat{\jmath}} \cap (-R_{\hat{\imath}\hat{\jmath}}/a_{\hat{\imath}\hat{\jmath}}) & \text{if } a_{\hat{\imath}\hat{\jmath}} \neq 0 \\ X_{\hat{\jmath}} & \text{if } a_{\hat{\imath}\hat{\jmath}} = 0 \text{ and } 0 \in R_{\hat{\imath}\hat{\jmath}} \\ \emptyset & \text{if } a_{\hat{\imath}\hat{\jmath}} = 0 \text{ and } 0 \notin R_{\hat{\imath}\hat{\jmath}}, \end{cases} \tag{4}$$

and

$$R_{\hat{\imath}\hat{\jmath}} = \sum_{j \neq \hat{\jmath}} a_{\hat{\imath}j} X_j + [-\infty, b_{\hat{\imath}}].$$

One checks that $X'_{\hat{\jmath}}$ is the smallest interval such that $\boldsymbol{X}$ and $\boldsymbol{X}'$ have the same intersection with the set $f_{\hat{\imath}} \geq 0$ and therefore (3) holds. Note the similarity of (2) and (4). Both formulas can be viewed as interval Gauss–Seidel steps, see for example [5].

Algorithm 1 incorporates the main concepts discussed so far. During execution it keeps track of an upper and a lower bound of the volume of the polytope in the starting box. If the difference of these bounds is less than ε, the algorithm terminates. In the next section we give a refined and optimized version.

Algorithm 1 [Volume of Polytope]

In: Linear inequalities $f_1 \geq 0, \ldots, f_m \geq 0$ over $x_1, \ldots x_n$.
Error bound $\varepsilon > 0$.
Starting box $\boldsymbol{X}$.

Out: u, l such that $u - l \leq \varepsilon$ and $l \leq \text{volume}(\mathcal{P} \cap \boldsymbol{X}) \leq u$, where $\mathcal{P}$ is defined by (1).

(1) [Initialize.] $u \longleftarrow \text{volume}(\boldsymbol{X})$, $l \longleftarrow 0$, $\mathcal{X} \longleftarrow \{\boldsymbol{X}\}$.

(2) [Test termination.] If $u - l \leq \varepsilon$ then return u, l.

(3) [Get box.] Remove a box $\boldsymbol{X}$ from $\mathcal{X}$.

(4) [Transform.] Do one of the following:
(i) [Test whether box is completely inside polytope.]
If $\boldsymbol{X} \subseteq \mathcal{P}$ then
$l \longleftarrow l + \text{volume}(\boldsymbol{X})$.

(o) [Test whether box is completely outside polytope.]
If $\boldsymbol{X} \cap \mathcal{P} = \emptyset$ then
$u \longleftarrow u - \text{volume}(\boldsymbol{X})$.

(h) [Test for unique active constraint.]
If there is only one active constraint in $\boldsymbol{X}$, then
$v \longleftarrow \text{volume}(\mathcal{P} \cap \boldsymbol{X})$,
$u \longleftarrow u - \text{volume}(\boldsymbol{X}) + v$, $l \longleftarrow l + v$.

(t) [Tighten.]
Choose a constraint f_i and a variable x_j.
$\boldsymbol{X}' \longleftarrow \text{tight}(\boldsymbol{X}, f_i, x_j)$.
If $\boldsymbol{X}' \neq \emptyset$ then insert $\boldsymbol{X}'$ into $\mathcal{X}$.
$u \longleftarrow u - \text{volume}(\boldsymbol{X}) + \text{volume}(\boldsymbol{X}')$.

(s) [Split.] Split $\boldsymbol{X}$ into sub boxes and insert them into $\mathcal{X}$.

(5) [Loop.] Goto Step 2.

4 Improved Algorithm

The following observations lead to an improvement of Algorithm 1.

Elimination of inactive constraints. It suffices to carry out tightening only on active constraints. Further, if a constraint is inactive in $\boldsymbol{X}$, then it is also inactive in any sub box of $\boldsymbol{X}$. The set of (possibly) active constraints of a box should therefore be stored in a list $\mathcal{F}$ together with the box. Initially, all constraints are assumed to be active. If during the computation some of them are detected to be inactive, they are removed from $\mathcal{F}$.

Tightening tests whether box is completely outside. If there exists x_j such that $\text{tight}(\boldsymbol{X}, f_i, x_j) = \emptyset$ then $\mathcal{P} \cap \boldsymbol{X} = \emptyset$. Similarly, if there exists x_j such that $\text{tight}(\boldsymbol{X}, f_i, x_j) \neq \emptyset$ then $\mathcal{P} \cap \boldsymbol{X} \neq \emptyset$. This means that the test whether $\boldsymbol{X}$ is outside the polytope is redundant after tightening.

Algorithm 2 [Volume of Polytope]

In: Linear inequalities $f_1 \geq 0, \ldots, f_m \geq 0$ over $x_1, \ldots x_n$.
Error bound $\varepsilon > 0$.
Starting box $\boldsymbol{X}$.

Out: u, l such that $u - l \leq \varepsilon$ and $l \leq \text{volume}(\mathcal{P} \cap \boldsymbol{X}) \leq u$, where $\mathcal{P}$ is defined by (1).

(1) [Initialize.] $u \longleftarrow \text{volume}(\boldsymbol{X})$, $l \longleftarrow 0$, $\mathcal{X} \longleftarrow \{(\boldsymbol{X}, \{f_1, \ldots, f_m\})\}$

(2) [Test termination.] If $u - l \leq \varepsilon$ then return u, l.

(3) [Get box.] Remove an element $(\boldsymbol{X}, \mathcal{F})$ from $\mathcal{X}$.

(4) [Fully Tighten.]
$u \longleftarrow u - \text{volume}(\boldsymbol{X})$.
Repeat
$\quad \boldsymbol{X}' \longleftarrow \boldsymbol{X}$.
$\quad$ For every f in $\mathcal{F}$
$\quad\quad$ For every variable x occuring in f
$\quad\quad\quad \boldsymbol{X} \longleftarrow \text{tight}(\boldsymbol{X}, f, x)$
$\quad\quad\quad$ If $\boldsymbol{X} = \emptyset$ then goto Step 2.
Until $\boldsymbol{X}' = \boldsymbol{X}$.
$u \longleftarrow u + \text{volume}(\boldsymbol{X})$.

(5) [Test whether box is completely inside some constraint or entire polytope.]
For every f in $\mathcal{F}$
$\quad$ If $f(\boldsymbol{X}) \geq 0$ then remove f from $\mathcal{F}$.
If $\mathcal{F} = \emptyset$ then $l \longleftarrow l + \text{volume}(\boldsymbol{X})$, goto Step 2.

(6) [Test for unique active constraint.]
If $|\mathcal{F}| = 1$ then
$\quad v \longleftarrow \text{volume}(\mathcal{P} \cap \boldsymbol{X})$,
$\quad u \longleftarrow u - \text{volume}(\boldsymbol{X}) + v$, $l \longleftarrow l + v$,
Goto Step 2.

(7) [Split.]
Split $\boldsymbol{X}$ into $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$ and store $(\boldsymbol{X}^{(1)}, \mathcal{F})$ and $(\boldsymbol{X}^{(2)}, \mathcal{F})$ in $\mathcal{X}$.

(8) [Loop.] Goto Step 2.

In order to keep Algorithm 2 simple we did not incorporate the following "low–level" optimizations:

- The test whether the box is completely inside the polytope in Step 5 is almost for free if intermediate results from the last tighten iteration in Step 4 are used.

- Sometimes it is possible to determine in advance that $\text{tight}(\boldsymbol{X}, f_i, x_j) = \boldsymbol{X}$. For example this is the case if $\boldsymbol{X}$ did not shrink in any of the variables where f_i depends on since the previous iteration.

- If there exists a unique active constraint in $\boldsymbol{X}$, then the volume computation in Step 6 can be simplified by taking into account that full tightening was carried out before. Let $\boldsymbol{X}'$ be as in (2), then $\boldsymbol{X} \backslash \boldsymbol{X}' \subseteq \mathcal{P}$ and the volume of $\mathcal{P} \cap \boldsymbol{X}$ is simply $\text{volume}(\boldsymbol{X}) - \text{volume}(\boldsymbol{X}')/2$.

- The termination criterion $\boldsymbol{X}' = \boldsymbol{X}$ in Step 4 can lead to slow convergence in the case of ill conditioned linear *equation* systems. It seems that this problem will arise only in special cases of linear inequalities.

Strategy issues. Algorithm 2 is non–deterministic in the following two steps:

- Step 3: Which box to choose? Choosing a box with largest volume guarantees termination independent of the splitting strategy and seems intuitively reasonable. Another promising strategy might be choosing a box with smallest number of active constraints.

- Step 7: Where to split? As mentioned in Section 3 it is advantageous to split in such a way that the number of active constraints in the sub boxes is reduced. If this is not possible, then we split in the middle of the longest coordinate direction. This strategy guarantees termination of the algorithm.

5 Alternative Approaches

Preconditioning. A linear inequality can be viewed as a particular linear equation with interval coefficients. For linear interval equation systems, preconditioning methods can improve efficiency significantly, see for example [6], [9], [10]. However, the problem is that the interval equation system which arises from the inequality system has unbounded interval coefficients on the right side. This could be handled by enclosing the polytope in a larger polytope which has pairs of parallel faces.

Alternative box selection strategy. One might consider doing depth first search in order to minimize memory consumption and overhead for searching the largest box. But this can easily lead to an infinite loop if splitting does not reduce the number of the active constraints in all sub boxes. Further, it can cause that much computation time is spent on very small boxes. One could overcome this by finding an a priori bound δ such that, if all boxes whose maximum edge length is less than δ are discarded, the error bound ε is still achieved. Such a bound is for example

$$\delta = \frac{\varepsilon}{2\, s\, \sqrt{n}},$$

where s is the surface of the starting box. This idea will be fully developed in a different report.

References

[1] T. Fahringer. *Automatic Performance Prediction for Parallel Programs on Massively Parallel Computers.* PhD thesis, University of Vienna, Department of Software Technology and Parallel Systems, October 1993.

[2] T. Fahringer. Using the P^3T to Guide the Parallelization and Optimization Effort under the Vienna Fortran Compilation System. In *IEEE Proc. of the Scalable High Performance Computing Conference 1994, Knoxville, TN*, May 1994.

[3] T. Fahringer and H. Hong. On Using Volume Computation to Estimate the Work Distribution for Parallel Programs. Technical Report 95-01, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, 1995.

[4] T. Fahringer and H. Zima. A Static Parameter based Performance Prediction Tool for Parallel Programs. In *Invited Paper, In Proc. of the 7th ACM International Conference on Supercomputing 1993*, Tokyo, Japan, July 1993.

[5] E. Hansen and S. Sengupta. Bounding Solutions of Systems of Equations using Interval Analysis. *BIT*, 21:203–211, 1981.

[6] E. Hansen and R. R. Smith. Interval Arithmetic in Matrix Computations, Part 2. *SIAM J. Numerical Analysis*, 4(1):1–9, 1967.

[7] H. Hong and V. Stahl. Safe Starting Regions by Fixed Points and Tightening. Technical Report 94-29, Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria, 1994.

[8] R. E. Moore. *Interval Anaysis*. Prentice-Hall, Englewood Cliffs, N.J., 1966.

[9] A. Neumaier. New Techniques for the Analysis of Linear Interval Equations. *Linear Algebra Appl.*, 58:273–325, 1984.

[10] A. Neumaier. Overestimation in Linear Interval Equations. *SIAM J. Numerical Analysis*, 24:207–214, 1987.