Boundary Intervals Method for Visualization of Polyhedral Solution Sets[*]

Irene A. Sharaya

Institute of Computational Technologies SB RAS,
Novosibirsk, Russia

sharaya@ict.nsc.ru

Abstract

A new "boundary intervals method" is proposed for investigation and visualization of polyhedral sets determined by systems of linear algebraic inequalities or represented as the union of solution sets to a finite number of systems of linear inequalities.

There are several visualization approaches for the solution sets of linear systems of relations (inequalities and equations), but they are unsatisfactory for unbounded and thin solution sets, and some of them can handle only square systems of relations. These approaches compute and exploit vertices of the polyhedral sets. Our new boundary intervals method uses boundary intervals instead of vertices, making it more informative and enabling us to overcome the limitations of the previous approaches.

Our boundary intervals method helps visualize AE-solution sets to interval linear relations systems that consist of equations, inequalities, or both because AE-solution sets are also polyhedral sets whose determining systems of linear inequalities can be derived from the initial interval system.

This paper describes the boundary intervals method for systems with two and three unknown variables and presents software implementations `lineq` and `IntLinIncXX`, designed for visualization of the solution sets to both interval and non-interval systems of linear relations.

Keywords: linear inequality system, polytope, polyhedron, polyhedral set, boundary interval, visualization.

AMS subject classifications: 52B10, 15A39, 65G40, 51M20

[*]Submitted: November 7, 2012; Revised: March 8, 2014; Accepted: May 17, 2015.
The work was presented at the international symposium SCAN'2012, Novosiirsk, Russia, September 23–29, 2012, and is published as a part of its proceedings.

1 Introduction

1.1 Purpose of the method

Our boundary intervals method was developed to visualize solution sets of interval linear systems of equations and inequalities. The characteristic feature of every such solution set is that it is described by a system of linear algebraic inequalities in each orthant of the space. In early 2012, the author knew of several approaches and software packages for visualization of such solution sets. However, these packages often failed to process thin (with empty interior) and unbounded solution sets,[1] and some worked only with square systems. We were also unable to find accessible software with acceptable functionality to visualize solution sets to usual systems of linear algebraic inequalities. Our wish was, having determined a rectangular matrix and a right-hand side vector and then warning about nothing, to get a picture of the solution set, no matter what it might be. The boundary intervals method has been designed to accomplish these purposes.

The *boundary intervals method* is a method for investigation and visualization of polyhedral sets in the Euclidean spaces $\mathbb{R}^2$ and $\mathbb{R}^3$ based on computation and use of a special *boundary intervals matrix* for systems of linear inequalities.

The first presentation of the new method was made at the international conference "Algebra and Linear Optimization", devoted to the centenary of the birth of Sergey Chernikov, in Yekaterinburg, Russia, May 2012. We described ideas of the method and demonstrated the package `lineq` for visualization of solution sets to systems of linear algebraic inequalities with two and three unknowns [15]. Next, at the 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numerical Computations, SCAN'2012, in Novosibirsk, Russia, in September 2012, we presented both the new method and specific results produced by the packages for visualization of the AE-solution sets to interval linear systems of equations [16, 17].

AE-solution sets to interval equations and inequalities are a natural generalization of the traditional solution sets to the interval case. The reader can find theory and applications of AE-solution sets for interval systems of equations in the survey [22]. In the context of our paper, it is important that the intersection of any AE-solution set with separate orthants of the entire space is a polyhedron for which the determining system of linear inequalities can be written easily from the initial interval system of relations. Therefore, every AE-solution set to an interval linear system of equations is the union of a finite number of polyhedra. This is why the AE-solution sets can be visualized by methods that process systems of linear inequalities.

There are many approaches to visualization of the solution sets for interval systems of linear equations and quite a few computer codes that implement them. First, `plotlinsol` from the package INTLAB [5], an interval extension of MATLAB, was developed by Siegfried Rump (Hamburg University of Technology). Several approaches to visualization of the solution sets to interval linear systems have been proposed and implemented by Walter Krämer (Bergische Universität, Wuppertal) and Evgenija Popova (Institute of Mathematics and Informatics, Bulgarian Academy of Sciences). The theoretical foundations of these approaches are presented in [8, 9, 10, 11], and the corresponding software is implemented in Java and in an interval extension of the computer algebra system Maple$^{\text{TM}}$. Based on this work, Popova also developed, in co-operation with Wolfram Research, several versions of the visualization programs within

[1]Rigorous definitions of all these terms are given at the fourth page of this paper.

the *Mathematica*$^{\text{TM}}$ system and supplemented them by a convenient web-interface.[2]

One disadvantage of all the above approaches and codes is that they process unbounded solution sets badly. Some codes cannot construct solution sets to the systems in which the number of equations does not coincide with the number of unknowns (equal to 2 or 3). Apart from the procedure `plotlinsol` from INTLAB, the remaining codes work unsatisfactorily with thin solution sets (having empty interior). Finally, the codes by Rump and by Krämer are designed for visualization of only united solution sets, that is, they process only one (although the most popular) solution set from a large family of AE-solution sets to interval linear systems of equations.

The code `AE-solset.ps` [18] developed by the author for drawing arbitrary AE-solution sets to interval linear 2×2-systems of equations stands in contrast to the above visualization techniques and packages. `AE-solset.ps` successfully copes with unbounded and thin solution sets. In addition, it may be the most portable such tool since it is implemented in the vector graphics language PostScript. On the other hand, restrictions on the accuracy of the data representation in PostScript impose constraints on the input data for which the code is guaranteed to work correctly.

The visualization methods mentioned above compute *vertices* of polyhedra that form the solution set. In contrast, our boundary intervals method uses *boundary intervals* instead of vertices, which are much more informative in the description and investigation of polyhedra than traditional vertex representations. The boundary intervals method thereby overcomes the limitations of the previous approaches.

1.2 Structure of the paper

Our paper gives the first detailed exposition of the boundary intervals method. The necessary background information on topology and polyhedra is gathered in this introductory section. In Section 2, theoretical foundations of the method are presented, while Sections 3 and 4 describe the basics of the drawing technique in the spaces $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively. The text of the paper is a revised and expanded version of [21]. The paper is accompanied by open source software packages [15, 19, 20] and their manuals, which substantially extend and enrich our exposition of the new method and can be regarded as continuations of this paper.

1.3 Necessary facts about polyhedra

In our text, we use the standard mathematical notation. In particular,

$\mathbb{N}$	is the set of positive integer (natural) numbers,
$\mathbb{R}$	is the set of real numbers (real axis),
$\mathbb{R}^n$	is the set of real n-vectors (considered as column vectors), and
$\mathbb{R}^{m\times n}$	is the set of real $m \times n$-matrices.

In geometry, convex analysis, piecewise linear topology, and other branches of mathematics, some terms used in our paper have different interpretations, so we clarify their meanings. Many definitions below coincide with those from [23], but some well-known objects appear under new names. For example, what is usually referred to as "a face exposed by supporting hyperplane" (see, e. g., [23]) is called "support" because "support" is more convenient for the description of the boundary intervals method. Our

[2]These programs were working successfully in 2002–2012 on the server of the Bulgarian Academy of Sciences at `http://cose.math.bas.bg/webComputing`.

term is laconic, it represents the geometrical sense of the object more precisely, and it does not contradict to the use of the word "face" in the elementary geometry of $\mathbb{R}^3$.

A *polyhedron* in $\mathbb{R}^n$ (after ancient Greek "polys" = many, "hédra" = base or seat) is a subset of the space $\mathbb{R}^n$ that can be represented as the solution set to a system of linear algebraic inequalities of the form

$$\left\{\begin{array}{ccccccccc} A_{11}x_1 & + & A_{12}x_2 & + & \ldots & + & A_{1n}x_n & \geq & b_1, \\ A_{21}x_1 & + & A_{22}x_2 & + & \ldots & + & A_{2n}x_n & \geq & b_2, \\ \vdots & & \vdots & & \ddots & & \vdots & & \vdots \\ A_{m1}x_1 & + & A_{m2}x_2 & + & \ldots & + & A_{mn}x_n & \geq & b_m, \end{array}\right.$$

or in matrix-vector notation,

$$Ax \geq b,$$

where $x = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^n$ is the unknown variable, $A = (A_{ij}) \in \mathbb{R}^{m\times n}$, $b = (b_j) \in \mathbb{R}^m$, and $m, n \in \mathbb{N}$.

Bounded polyhedra are called *polytopes.* The space $\mathbb{R}^n$ is itself a polyhedron, the solution to the inequality

$$(0\ldots 0)\, x \geq -1.$$

The empty set is a polytope, since the empty set is bounded by definition, and it is the solution set, e. g., to the inequality system

$$\left\{\begin{array}{rcl} x_1 & \geq & 1, \\ -x_1 & \geq & 1. \end{array}\right.$$

The union of a finite number of polyhedra will be referred to as *polyhedral set.*[3]

In $\mathbb{R}^n$ considered as a topological space, we can define interior and boundary points for a set $\Omega \subseteq \mathbb{R}^n$ (see, e. g., [6, 12]). A point x is an *interior point* of Ω if there exists an open ball centered at x which is entirely contained in the set Ω. A point x is a *boundary point* of Ω if any ball centered at x contains both points from Ω and points outside Ω. Boundary points of Ω do not necessarily belong to the set Ω itself. All the interior points form the *interior* of a set (we will also use the term *interior domain*), while all the boundary points form the *boundary*, denoted by $\partial\Omega$.

The interior and/or boundary of a polyhedron may be empty. Polyhedra with nonempty interior are called *bodily* or *solid*, while polyhedra having no interior points are called *thin.* Such a terminology originates from classical set theory and theory of convex sets, since bodily polyhedra are bodily (solid) sets, i. e., convex sets with nonempty interior, while thin polyhedra are thin (meager) sets in the set-theoretic sense [6]. Thin (meager) sets also are called *sets of first Baire category* [6].

A point x of a set Ω is called a *limit point* if every ball centered at x contains at least one point of Ω different from x itself. If a set contains all of its limit points, it is said to be *closed.* A polyhedron is a closed set.

Any polyhedron is a convex set. For every two points belonging to a polyhedron, each point on the straight line segment that connects them is also within the polyhedron. The *dimension* of a non-empty polyhedron is the dimension of its affine hull, the

[3]Our definition of polyhedra may appear not standard (see, e. g., [1, 2, 4]). Very often, "polyhedron" is used for what we refer to as "polyhedral set", while polyhedra in our sense are called "convex polyhedra". See, e. g., `https://en.wikipedia.org/wiki/Polyhedron`. Sometimes, polyhedra are defined to be bounded, see [1]. Our definition of polyhedron originates from optimization theory and its applications, especially linear programming (see, e. g., [13]).

smallest affine space that contains the polyhedron (see, e. g., [12, 13]). The dimension of the empty set is -1. The dimension of the polyhedron H will be denoted by $\dim H$. In $\mathbb{R}^n$, n-dimensional polyhedra are bodily, while those having the dimension less than n, are thin.

A further refinement of the concept of the interior is the *relative interior*. For convex sets, the relative interior of a set is defined as its interior within the affine hull of the set [12]. The relative interior is very useful in considerations about low-dimensional sets placed in higher-dimensional spaces, such as faces and edges of polyhedra and polyhedral sets. Their usual interiors are empty, while sometimes we need their relatively interior points.

Let Ψ designate a *half-space* of the space $\mathbb{R}^n$, i. e., the set of points defined by one non-strict linear algebraic inequality. A half-space Ψ is called *supporting* for a set Ω if Ψ contains Ω, and the boundary $\partial\Psi$ of the half-space has at least one common point with Ω (see [12, 23] for further details). In that case, the boundary of the supporting half-space is referred to as a *supporting hyperplane*. For the intersection of the supporting hyperplane with the set Ω, we use the term *support* (of the set Ω, of the half-space Ψ, or of the hyperplane $\partial\Psi$). In the above definition, we assume that the set Ω can belong entirely to the hyperplane $\partial\Psi$; then both closed half-spaces with the boundary $\partial\Psi$ are supporting half-spaces for Ω.

Any support S of a polyhedron H is also a non-empty polyhedron since the equality determining the support is equivalent to a pair of opposite non-strict inequalities. For a thin polyhedron H, $\dim S \leq \dim H$. If H is a bodily polyhedron, $\dim S \leq \dim H - 1$. Supports of dimension zero are called *vertices* of the polyhedron, and supports of the dimension one are its *edges*. For polyhedra in $\mathbb{R}^3$, supports of dimension two are *faces*.

A point from a polyhedron H is a vertex if and only if we cannot find a segment in H that contains the point in its relative interior. This fact is often formulated in the following words: vertices of a polyhedron are its *extreme* points. Any polytope coincides with the convex hull of its vertices, the smallest convex set that contains these vertices, i. e., the set of all convex combinations of these vertices.

Any edge has two endpoints, either finite or infinite. An endpoint all of whose coordinates are finite is a vertex of the polyhedron. We say "the endpoint is at infinity" for an endpoint with at least one infinite coordinate, either $+\infty$ or $-\infty$. The endpoint of an edge that is "at infinity" cannot be a vertex of a polyhedron.

An *orthant* of $\mathbb{R}^n$ is the set of points that have constant signs of their coordinates, taken with its boundary. An orthant is thus the set of the form $\{\, x \in \mathbb{R}^n \mid s_i x_i \geq 0,\ i = 1, \dots, n \,\}$, where the n-vector s consists of 1's and -1's. Orthants of $\mathbb{R}^2$ are also called quadrants, while orthants of $\mathbb{R}^3$ are often referred to as octants.

2 Basics of the Boundary Intervals Method

2.1 Boundary inequalities and their supports in $\mathbb{R}^n$

Consider a system of linear algebraic inequalities

$$Ax \geq b, \quad A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ m, n \in \mathbb{N}, \tag{1}$$

with the unknown variable x. We denote its solution set by H, the i-th row of the matrix A by $A_{i:}$, and the solution set to the inequality $A_{i:}x \geq b_i$, written in the i-th line of the system, by H_i.

The solution set of (1) is formed through intersection of the solution sets to the separate inequalities $A_{i:}x \geq b_i$, $i = 1, 2, \ldots, m$. Hence, the interior points of the solution set H are exactly the points which are interior for every solution set H_i to the inequalities of (1), while boundary points of H are those which lie on the boundary of at least one of H_i. We rewrite the last statement in formal mathematical language and transform it, changing the order in which the union and intersection are taken:

$$\partial H = H \cap \left(\bigcup_i \partial H_i \right) = \bigcup_i \, (H \cap \partial H_i). \tag{2}$$

Therefore, the boundary of the solution set to the system (1) consists of the contributions $H \cap \partial H_i$, $i = 1, 2, \ldots, m$, made by separate inequalities. We are going to consider these contributions in detail. For brevity, we will speak of "hypreplane $A_{i:}x = b_i$" instead of the full term "hyperplane determined by the equation $A_{i:}x = b_i$". In the two-dimensional case, "hyperplane $A_{i:}x = b_i$" is just a "straight line $A_{i:}x = b_i$". The same terminology convention is applied to half-spaces determined by non-strict inequalities $A_{i:}x \geq b_i$ (half-planes in the two-dimensional case).

If the row $A_{i:}$ has only zero elements, the contribution of the i-th inequality to the boundary of the solution set is empty. The inequality $(0 \ldots 0)\, x \geq b_i$ has no solutions for $b_i > 0$, and every point of $\mathbb{R}^n$ satisfies the inequality for $b_i \leq 0$. In both cases, the solution set to the inequality $(0 \ldots 0)\, x \geq b_i$ has no boundary, $\partial H_i = \varnothing$.

If the row $A_{i:}$ has non-zero elements, the solution set of the i-th inequality is a half-space H_i with its boundary ∂H_i being the hyperplane $A_{i:}x = b_i$. Therefore, the contribution of the i-th inequality to the boundary of the set H is the intersection of the solution set to system (1) with the hyperplane $A_{i:}x = b_i$. Formally, this contribution is described as $\{\, x \in \mathbb{R}^n \mid (Ax \geq b) \,\&\, (A_{i:}x = b_i) \,\}$. Nonemptiness of the intersection means that the half-space H_i is a supporting half-space for H.

Overall, the contribution of the i-th inequality to the boundary of the solution set is not empty if and only if the inequality determines a supporting hyperplane for the solution set. We can fix this relation in the following definitions. For system (1), an inequality $A_{i:}x \geq b_i$ written in the i-th line is called a *boundary inequality* if it determines a supporting hyperplane for the solution set H. If the i-th inequality is a boundary one, the intersection of the hyperplane $A_{i:}x = b_i$ with the set H is called the *support of the i-th inequality* and denoted by S_i. Specifically, $S_i = H \cap \partial H_i$. If the inequality $A_{i:}x \geq b_i$ from system (1) is not a boundary inequality, it has no support.

We have selected from system (1) all those inequalities that make nonempty contributions to the boundary of the solution set. We refer to such inequalities as boundary inequalities, and the contribution made by a boundary inequality is called its support. If we denote the set of indices of the boundary inequalities by I_b, then (2) implies

$$\partial H \;=\; \bigcup_{i \in I_b} (H \cap \partial H_i) \;=\; \bigcup_{i \in I_b} S_i;$$

the boundary of the solution set H is formed by supports of the boundary inequalities.

Every boundary inequality determines a half-space supporting the polyhedron of the solution set, but the reverse is not true. A boundary inequality determining a half-space cannot be found in system (1) for every supporting half-space of the solution polyhedron. For instance, the system

$$\begin{cases} x_1 \geq 0, \\ x_2 \geq 0 \end{cases} \tag{3}$$

describes the positive orthant for which the half-space $x_1 + x_2 \geq 0$ is supporting at the origin of the coordinates. However, system (3) does not have inequalities that determine this half-space. Meanwhile, there are such supporting half-spaces that we can always connect with boundary inequalities.

Proposition 1 *Let the inequality system*

$$Ax \geq b, \quad A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ m, n \in \mathbb{N}, \tag{1}$$

be given and have a non-empty solution set. If for a half-space Ψ that is supporting the solution polyhedron, its support S has dimension $n-1$, then system (1) *has a boundary inequality that determines the half-space Ψ, and S is a support of this inequality.*

For example, in system (3), the support has dimension 0, i. e., $n-2$, not $n-1 = 1$.

$\triangleright$ **Proof.** We conduct the proof *ad absurdum* and suppose that system (1) has no inequalities determining the half-space Ψ.

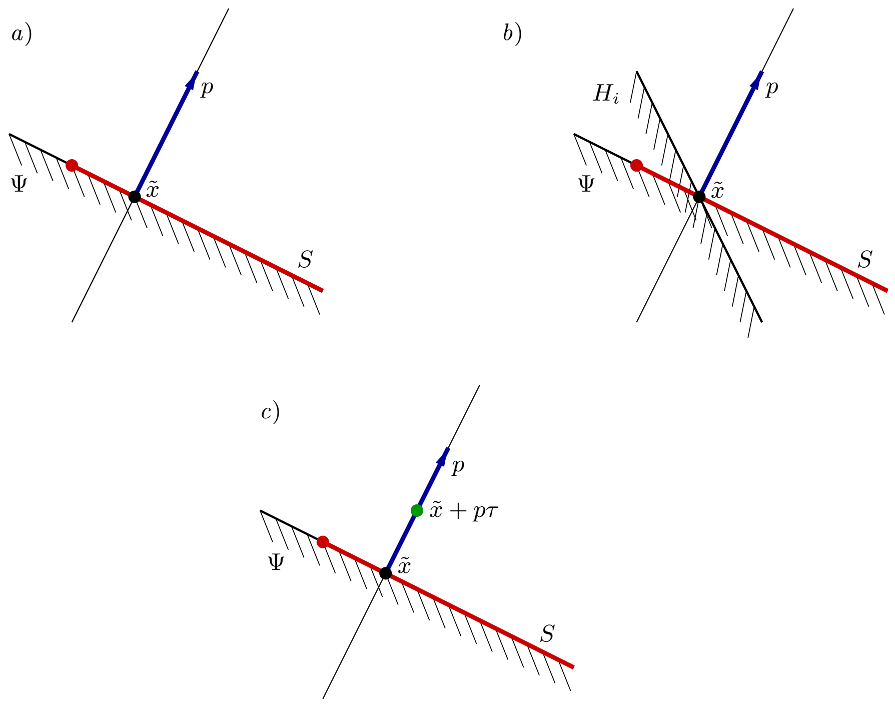


Figure 1: Illustration of the proof of Proposition 1.

Let $\tilde{x}$ be a point from the relative interior of the support S, and let p be an outward normal vector with respect to Ψ (Fig. 1, *a*). Consider the intersection of the solution set H_i, $i = 1, 2, \ldots, m$, of every inequality of system (1) with the straight line $\{\, \tilde{x} + pt \mid t \in \mathbb{R}\,\}$. The point $\tilde{x}$ lies on the support S, so it lies within the solution polyhedron H of the system (1), which means its membership in each H_i. Since the solution set H_i to the inequality $A_{i:}x \geq b_i$ is not empty, it is either a half-space containing the point $\tilde{x}$ or the entire space $\mathbb{R}^n$. Hence, the intersection of the set H_i with the line $\{\, \tilde{x} + pt \mid t \in \mathbb{R}\,\}$ is either the whole line or a ray containing the point $\tilde{x}$.

Next, we show that, apart from $\tilde{x}$, the intersection of the set H_i with the line $\{\, \tilde{x} + pt \mid t \in \mathbb{R}\,\}$ has a point of the form $\tilde{x} + p\tau_i$, with a fixed $\tau_i \in \mathbb{R}$ satisfying $\tau_i > 0$. The absence of such a point would mean that H_i intersects the line $\{\, \tilde{x} + pt \mid t \in \mathbb{R}\,\}$ along the ray with the starting point $\tilde{x}$ and direction opposite to p. By assumption, H_i cannot coincide with Ψ, while S has dimension $n-1$ due to assertion of the proposition. Finally, the point $\tilde{x}$ is taken from the relative interior of the support S. Therefore, S would have to contain points from the complement to the set H_i (see Fig. 1, *b*). But this is impossible, because S lies in the solution polyhedron for system (1) and consequently is in the solution set H_i to its every inequality.

Since the points $\tilde{x}$ and $\tilde{x} + p\tau_i$ belong to the convex set H_i, the entire line segment with the endpoints $\tilde{x}$ and $\tilde{x} + p\tau_i$ lies within H_i too. The latter is valid for each $i = 1, 2, \ldots, m$, and we denote $\tau = \min_{1 \leq i \leq m} \tau_i$. The point $\tilde{x} + p\tau$ (see Fig. 1, *c*) belongs to every H_i, $i = 1, \ldots, m$. Therefore, it is a solution to the system (1). On the other hand, for $\tau > 0$, the point lies in the complement to the supporting half-space Ψ. Hence, the point cannot be a solution to the system (1). The contradiction obtained implies that our assumption made at the beginning of the proof is wrong.

Thus in system (1), there exists an inequality that determines the supporting half-space Ψ. By definition, it is a boundary inequality and has the same support S as the half-space Ψ. ◁

Proposition 2 *The vertex set of the solution polyhedron for system* (1) *is the union of the vertex sets of the supports of the boundary inequalities from* (1).

▷ **Proof.** We show first that each vertex of the solution polyhedron H is a vertex for support of a boundary inequality. Since the vertex is a boundary point of H and $\partial H = \bigcup_{i \in I_b} S_i$, then any vertex v of the polyhedron H belongs to the support S_i of a boundary inequality. A point of a polyhedron is known to be a vertex if and only if the polyhedron does not have a line segment that contains the point in its relative interior. Since there is no such segment in H, and S_i is in H, there is no such segment in the polyhedron S_i. Hence, v is a vertex of the support S_i.

Next, the reverse is to be proven, that any vertex of support of a boundary inequality is a vertex of the solution polyhedron. Let v be a vertex of support S_i of the boundary inequality with index i. We assume that v is not a vertex of H. Therefore in H, there exists such a line segment that contains the point v in its relative interior. This segment cannot be included entirely in the supporting hyperplane $A_{i:}x = b_i$, since then it would belong to S_i, and v could not be a vertex of S_i. But the segment cannot intersect the hyperplane $A_{i:}x = b_i$ only in the point v, since the hyperplane is a supporting one, and the whole polyhedron H lies on a one side of it. Hence, our assumption that v is not a vertex of H is false. ◁

2.2 Boundary intervals in $\mathbb{R}^2$

2.2.1 Introduction of the concept "boundary interval"

We introduce the concept of "boundary interval" for systems of linear algebraic inequalities

$$Ax \geq b, \quad A \in \mathbb{R}^{m\times 2},\ x \in \mathbb{R}^2,\ b \in \mathbb{R}^m,\ m \in \mathbb{N}, \tag{4}$$

with two unknown variables x_1 and x_2, $(x_1, x_2)^\top = x$. The concept consists of three constituents: index, support, and direction.

Index of the boundary interval fixes its connection with an inequality from system (4). Boundary intervals are defined only for boundary inequalities of (4). Each boundary inequality generates one boundary interval. The *index of the boundary interval* is the number of the row from system (4) in which the boundary inequality generating the boundary interval is written.

Support represents the boundary interval as a set of points on the plane. *Support of the boundary interval with the index i* is the support of the i-th inequality of system (4), the intersection of the line $A_{i:}x = b_i$ with the solution set H.

Both the line and the set H are convex and closed. Hence, the support of a boundary interval is always a convex and closed subset of a straight line. Additionally, the support cannot be empty. As a consequence, supports of boundary intervals are one of the following four types: a point, a straight line segment, a ray, or an entire line. Fig. 2 presents examples of each type of support for boundary intervals of index 1. The solution set H is marked by hatching, while the support S_1 is shown as a thick segment of the line $A_{1:}x = b_1$.

Direction of the boundary interval with index i specifies a motion along the line $A_{i:}x = b_i$ for which the half-plane $A_{i:}x \geq b_i$ remains on the right-hand side. The opposite direction can be taken equally well; it is only important that the directions are chosen uniformly for all the boundary inequalities.

The solution set H to system (4) satisfies the inequality $A_{i:}x \geq b_i$, which means that H is entirely included in the half-plane determined by this inequality. Therefore, when we are moving along the line $A_{i:}x = b_i$ in the direction of the boundary interval with the index i, the solution set H also stays at the right-hand side.[4]

To get a numerical description of the direction chosen for a boundary interval, we turn to Fig. 3. For the inequality $A_{i:}x \geq b_i$, the vector $A_{i:}^\top = (A_{i1}, A_{i2})^\top$ is perpendicular to the line $A_{i:}x = b_i$, being directed inward to the solution half-plane for this inequality. In other words, the vector $A_{i:}^\top = (A_{i1}, A_{i2})^\top$ is an inward normal vector for the solution half-plane. After rotating this vector by 90° clockwise, we get one of two possible directions along the line $A_{i:}x = b_i$. The other direction (opposite to the first one) can be obtained after rotating the inward normal vector by 90 degrees counter-clockwise. We choose a direction of the motion along the line $A_{i:}x = b_i$ such that the half-plane $A_{i:}x \geq b_i$ remains on the right-hand side during this motion. It is not hard to understand that the required direction corresponds to the vector $A_{i:}^\top$ rotated by 90° counter-clockwise, i. e., it is $(-A_{i2}, A_{i1})^\top$.

When a direction of the line $A_{i:}x = b_i$ is fixed, we can define naturally the concepts of *start* and *finish* for the support of a boundary interval (Fig. 4). The start and finish of the support are called *endpoints* of the boundary interval.

In the term “boundary interval”, the word “boundary” displays the relation with the boundary of the solution set and boundary inequalities of system (4), and the term “interval” is chosen by analogy with the intervals over the extended real axis $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. These intervals have the form $[\underline{z}, \overline{z}] = \{\, x \in \mathbb{R} \mid \underline{z} \leq x \leq \overline{z} \,\}$ for some $\underline{z}$, $\overline{z} \in \overline{\mathbb{R}}$. They have an analog of the start — left (or lower) endpoint $\underline{z}$, and

[4]If the set H entirely lies *on* the straight line $A_{i:}x = b_i$, we say that it is simultaneously on both the right-hand side and the left-hand side in any motion along the line $A_{i:}x = b_i$.

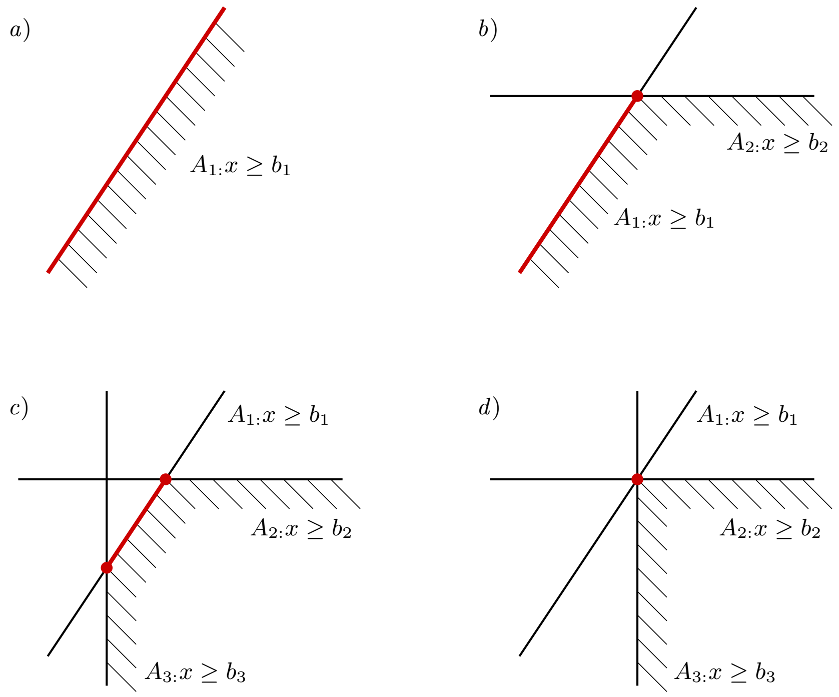


Figure 2: Supports of boundary intervals: *a*) line, *b*) ray, *c*) line segment, *d*) point.

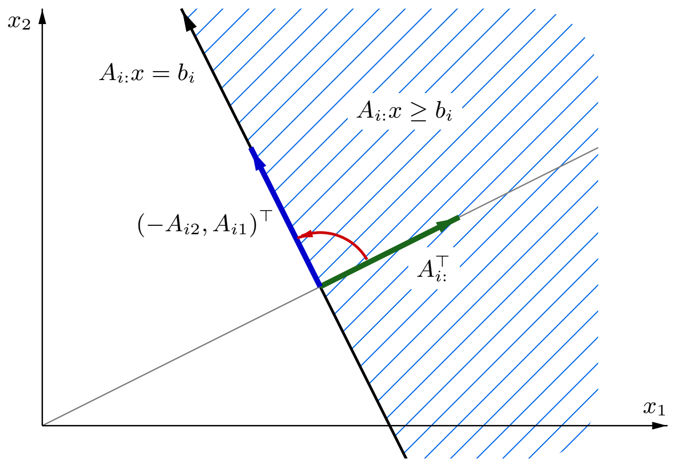


Figure 3: Choosing direction on the straight line $A_{i:}x = b_i$.

analog of the finish — right (or upper) endpoint $\overline{z}$, while their geometrical images, similar to the boundary intervals, are a point, a usual interval, a ray, or the whole real axis $\mathbb{R}$.

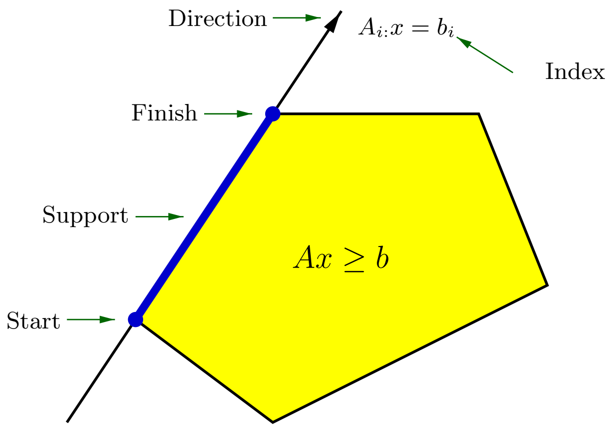


Figure 4: Visual representation of a boundary interval.

2.2.2 Numerical representation of boundary intervals

To use boundary intervals in computation, we need numerical expressions for all of their components.

General form of numerical representation of boundary intervals. First, we discuss which quantities and in what order will describe a boundary interval.

We arrange that in the coordinate system associated with the variables x_1 and x_2, the abscissa and ordinate take values from the extended real axis $\overline{\mathbb{R}}$. Then the start and finish points of any boundary interval may be considered as points of the extended real plane $\overline{\mathbb{R}}^2$.

Using the coordinates of the extended real plane, we represent a boundary interval as an ordered five-tuple of numbers: the first two numbers are the abscissa and ordinate of the start, then two coordinates of the finish, and the fifth number is the index of the boundary interval:

$$\Big(\underbrace{* \qquad *}_{\text{start} \in \overline{\mathbb{R}}^2} \quad \underbrace{* \qquad *}_{\text{finish} \in \overline{\mathbb{R}}^2} \quad \underbrace{i}_{\text{index} \in \mathbb{N}} \Big). \tag{5}$$

Such a record represents the boundary interval in numbers convenient for the boundary intervals method:

1. The index is present in the record as a separate number.
2. The support is described by coordinates of its two endpoints. (Notice that unbounded support cannot always be reconstructed from the coordinates of its endpoints. The boundary intervals method does not need such reconstruction, but the unique reconstruction becomes possible after involving the generating inequality of the corresponding index.)
3. The direction of the boundary interval is set by dividing the support endpoints to the start and finish. (If the support is a point, its relation with the direction of the corresponding straight line fails, but the direction of any point support does not matter in the boundary intervals method.)

All the information about any boundary interval can be obtained from its index and the initial data of the system of inequalities (4), i. e., from the matrix A and vector b. Still, when introducing the record (5), we had to construct a representation of the boundary interval convenient for use in our visualization technique.

Computing boundary intervals. Next, we discuss how to reveal, from the matrix A and vector b, whether the inequality $A_{i:}x \geq b_i$ generates a boundary interval of system (4), and if it does, how to compute the coordinates of the start and finish for this boundary interval. We already know (see Section 2.1) that if $A_{i:} = (0\ 0)$, the inequality $A_{i:}x \geq b_i$ cannot be boundary and does not generate a boundary interval. Therefore, we will concentrate on the case $A_{i:} \neq (0\ 0)$.

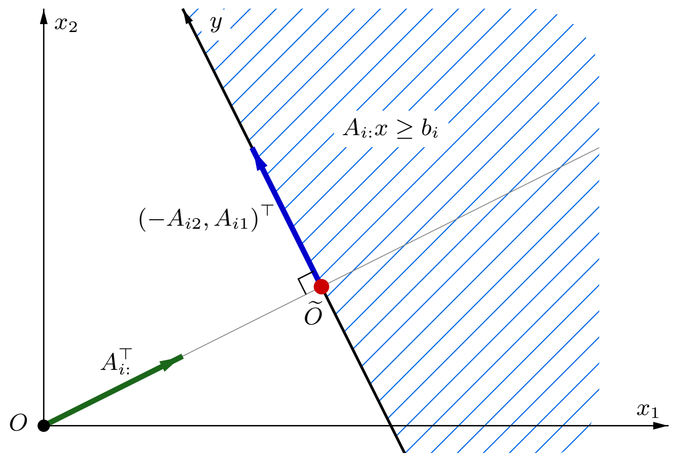


Figure 5: Choosing an internal coordinate system on the line $A_{i:}x = b_i$.

On the straight line $A_{i:}x = b_i$, we introduce an internal coordinate system. We choose the origin of coordinates, a unit vector, and an internal variable. One suitable variant of this construction is as follows (see Fig. 5):

- As the origin $\widetilde{O}$ of the internal coordinate system, we take the projection of the origin $(0, 0)$ of the "external" coordinate system Ox_1x_2 onto the line $A_{i:}x = b_i$. Then the vector $\widetilde{O}$ is proportional to its normal vector $A_{i:}^\top$, and hence, $\widetilde{O} = tA_{i:}^\top$ for some $t \in \mathbb{R}$, $t \neq 0$. On the other hand, $\widetilde{O}$ lies on the line $A_{i:}x = b_i$. This is why the value t can be computed from the equality $A_{i:}\left(tA_{i:}^\top\right) = b_i$. We get $\left(A_{i:}A_{i:}^\top\right)t = b_i$, so that $t = b_i/\|A_{i:}\|_2^2$, where $\|A_{i:}\|_2 = \sqrt{A_{i:}A_{i:}^\top} = \sqrt{A_{i1}^2 + A_{i2}^2}$ is 2-norm (Euclidean norm) of the row $A_{i:}$. Therefore,

$$\widetilde{O} = b_i A_{i:}^\top \ / \ \|A_{i:}\|_2^2.$$

- We have agreed to choose the direction on the straight line $A_{i:}x = b_i$ to coincide with the direction of the vector $(-A_{i2}, A_{i1})^\top$ (see Section 2.2.1). For this reason, we take the vector $(-A_{i2}, A_{i1})^\top$ itself as the unit vector (orth) of the internal coordinate system.
- The internal coordinate is denoted by y.

Having thus fixed the coordinate system, the straight line $A_{i:}x = b_i$ admits the parametric description

$$\frac{b_i}{\|A_{i:}\|_2^2}\, A_{i:}^{\top} + (-A_{i2}, A_{i1})^{\top} y, \qquad y \in \mathbb{R}. \tag{6}$$

To find the intersection of the line (6) with the solution set to the system $Ax \geq b$, we make the change of variables

$$x \;\longrightarrow\; \frac{b_i}{\|A_{i:}\|_2^2}\, A_{i:}^{\top} + (-A_{i2}, A_{i1})^{\top} y$$

to arrive at a system of inequalities with only one unknown variable y. The resulting mono-variable system can be solved easily by treating every inequality (with a single unknown) separately and intersecting their solutions.

The solution set to the entire system of inequalities is either the empty set or an interval $[\underline{y}, \overline{y}]$ of the extended real axis $\overline{\mathbb{R}}$. If the solution set is empty, the inequality $A_{i:}x \geq b_i$ is not the boundary one, and it generates no boundary interval. If the solution set is not empty, the inequality $A_{i:}x \geq b_i$ is a boundary one, and it generates a boundary interval with index i for which $\underline{y}$ is the start coordinate and $\overline{y}$ is the finish coordinate in the coordinate system we have constructed on the straight line $A_{i:}x = b_i$.

Next, we transform the start and finish of the boundary interval to the initial external coordinate system:

$$\text{start} = \frac{b_i}{\|A_{i:}\|_2^2}\, A_{i:} + (-A_{i2}, A_{i1})\, \underline{y}\,,$$

$$\text{finish} = \frac{b_i}{\|A_{i:}\|_2^2}\, A_{i:} + (-A_{i2}, A_{i1})\, \overline{y}\,.$$

2.2.3 How edges and vertices of solution polyhedron relate to supports and endpoints of boundary intervals

The boundary interval is an algebraic construction derived from the initial system of linear inequalities (1). Nevertheless, the boundary intervals have much in common with some well-known geometric notions. For the system (4) that determines a polyhedron in $\mathbb{R}^2$, supports and endpoints of the boundary intervals are closely related to edges and vertices of the solution polyhedron,[5] as described by Propositions 3–8.

Proposition 3 *Support of a boundary interval can be only a vertex or an edge of the solution polyhedron.*

$\triangleright$ **Proof.** Support of the boundary interval with index i is defined as support of the boundary inequality $A_{i:}x \geq b_i$ in system (4). It is the intersection of the straight line $A_{i:}x = b_i$, which is supporting the solution polyhedron H, with this polyhedron. Hence, support of the boundary interval is support for H and has dimension at most one. $\triangleleft$

Proposition 4 *If the coordinates of the start and finish for a boundary interval coincide with each other, the corresponding support is a vertex. If the coordinates of the start and finish differ, the corresponding support is an edge of the solution polyhedron.*

[5]We continue speaking of "polyhedra" for uniformity of our style, although these "polyhedra" are actually "polygons" on the plane $\mathbb{R}^2$.

▷ **Proof.** From Proposition 3, support of the boundary interval may be either a vertex or an edge of the solution polyhedron, and there are no other variants. Therefore, it suffices to prove only the one-way implication that if the support is a vertex, its start and finish have equal coordinates, and if the support is an edge, then the endpoints of the boundary interval differ in at least one coordinate in $\overline{\mathbb{R}}^2$.

Let the support be a vertex. Then a geometrical image of the support is a point from $\mathbb{R}^2$. Both the start and finish coincide with it, so their coordinates must agree.

When the support is an edge, its possible geometrical images are a line segment, a ray, or a whole straight line. If the support is a segment, its start and finish are different points of the segment, so at least one of their coordinates must differ. If the support is a ray, one of the endpoints has finite coordinates, while the other endpoint may have either $+\infty$ or $-\infty$ in its coordinates. If the support is a line parallel to a coordinate axis, the start and finish of the support differ in the corresponding coordinate by the sign at ∞'s. If the support is a line not parallel to coordinate axes, the start and finish of the support differ in both coordinates by the signs at ∞'s. ◁

Proposition 5 *Let H be a solution polyhedron for system* (4).

1. *Any edge S of the set H is support for at least one boundary interval.*
2. *If H has the dimension* 1*, its edge is support for at least two boundary intervals having opposite directions.*

▷ **Proof.**

1. An edge of a polyhedron is, by definition, a one-dimensional support S for a half-space Ψ supporting the polyhedron. In the system of inequalities (4), the number of unknowns $n = 2$, so that Ψ is a half-plane, and the dimension of S is $1 = n-1$. From Proposition 1, system (4) has a boundary inequality that determines the half-plane Ψ and has support S. This inequality, being a boundary one, generates a boundary interval with the edge S as its support and the direction that specifies motion along the edge in which the half-plane Ψ remains at the right.

2. If the dimension of the polyhedron H is one, it coincides with its single edge S. This edge is support of two half-planes Ψ_1 and Ψ_2 that lie on different sides of S. For each of these half-planes, Proposition 1 implies that system (4) has a boundary inequality describing the corresponding half-plane. The boundary intervals generated by these inequalities have edge S as support, but differ in their directions. ◁

As distinct from edges, *vertices* of the solution polyhedron to system (4) *are not necessarily supports of boundary intervals.* For example, in the system

$$\begin{cases} x_1 \geq -1, \\ -x_1 \geq -1, \\ x_2 \geq -1, \\ -x_2 \geq -1, \end{cases}$$

all four supports are edges, the sides of the square solution set $([-1,1],[-1,1])^\top$.

Notice that *any edge, as well as any vertex, can be support for several boundary intervals at the same time.* Let us consider specific examples:

1. For the system

$$\begin{cases} x_1 + x_2 \geq 0, \\ 2x_1 + 2x_2 \geq 0, \end{cases}$$

the only (infinite) edge of the solution set are supports of the boundary intervals generated by both inequalities.

2. For the system

$$\begin{cases} x_1 \geq 0, \\ -x_1 \geq 0, \\ x_2 \geq 0, \\ -x_2 \geq 0, \end{cases}$$

the solution set consists only of the point $(0,0)$. All the inequalities of the system are boundary, and the vertex $(0,0)$ is support of the boundary interval for each of them.

We emphasize that coincidence of supports does not mean that the boundary intervals are equal to each other; see example 1 above. The boundary intervals always differ in their indices, their ordinal numbers within system (4) of the generating inequalities.

The boundary of the solution polyhedron for system (4) is the union of its vertices and edges. An analogous assertion is valid for supports of the boundary intervals.

Proposition 6 *The boundary of the solution set to system* (4) *is the union of supports of boundary intervals.*

$\triangleright$ **Proof.** In Section 2.1, we showed that the boundary of the solution polyhedron to a system of linear inequalities consists of supports of the boundary intervals,

$$\partial H = \bigcup_{i \in I_b} S_i, \tag{7}$$

where I_b are all ordinal numbers (indices) of the boundary inequalities in (4), and S_i is support of the inequality i. Since the index and support of the boundary inequality are, by definition, the index and support of a boundary interval generated by the inequality, we can suppose that in representation (7), I_b is the index set of the boundary intervals, and S_i is support of boundary interval i. As a consequence, the boundary of the polyhedron H consists of supports of the boundary intervals. $\triangleleft$

Proposition 6, unlike Propositions 3–5, is valid in the general case, no matter how many unknowns the system of inequalities (1) has.

We have considered the relation of vertices and edges of the solution polyhedron with supports of the boundary intervals. Next, we discuss the connection between vertices and endpoints of the boundary intervals for systems (4) of inequalities with two unknowns.

Proposition 7 *An endpoint of the boundary interval is a vertex of the solution set to system* (4) *if and only if both its coordinates are finite.*

$\triangleright$ **Proof.** From Proposition 3, support of the boundary interval is either a vertex or an edge of the solution polyhedron. If the support of the boundary interval is a vertex, both endpoints coincide with it, and the coordinates of any vertex, abscissa and ordinate, are finite. If the support of the boundary interval is an edge, the endpoints of the boundary interval are endpoints of the edge. An endpoint of the edge with finite coordinates is a vertex of the polyhedron. An endpoint of an edge with at least one infinite coordinate is not a vertex. $\triangleleft$

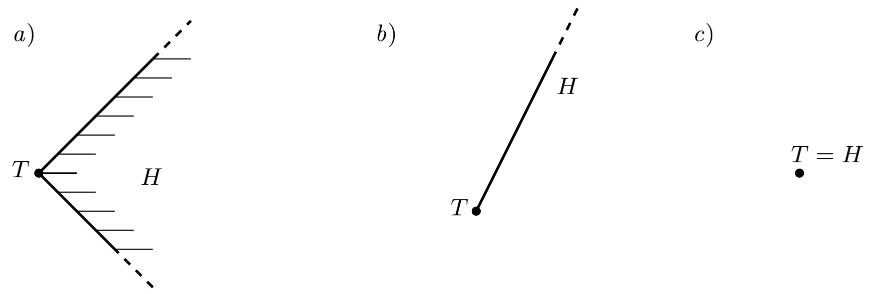


Figure 6: Position of a vertex T depending on the dimension of the polyhedron H: a) dimension 2, b) dimension 1, c) dimension 0.

Proposition 8 *Each vertex of the solution set to* (4) *is both the start of a certain boundary interval and the finish of a certain boundary interval.*

▷ **Proof.** The position of a vertex within the solution polyhedron H depends on its dimension, which can take values 2, 1, or 0 (see Fig. 6).

If $\dim H = 2$ (Fig. 6, a), two edges meet in the vertex. Proposition 5 implies that at least one boundary interval corresponds to each such edge. The boundary interval has direction so that, when moving from the start to the finish, the set H remains on the right. That is why the vertex should be the finish for a boundary interval that has one of the edges as support, and at the same time the vertex is the start for the boundary interval having the other edge as support.

If $\dim H = 1$ (Fig. 6, b), the polyhedron H coincides with its single edge. Proposition 5 implies that this edge has at least two boundary intervals as supports, and these boundary intervals have opposite directions. For the boundary intervals with one direction, the vertex is the finish, while for the boundary intervals with the opposite direction, the vertex is the start.

If $\dim H = 0$ (Fig. 6, c), the polyhedron is a point. It is both the unique vertex and the boundary of the polyhedron. Proposition 6 implies that this point is the union of supports of all the boundary intervals. Therefore, the point is both the start and the finish for every boundary interval. ◁

2.3 Boundary intervals matrix in $\mathbb{R}^2$

We arrange all the boundary intervals of system (4) as rows in the matrix

$$\left(\begin{array}{ccc} \underbrace{\begin{matrix} * & * \\ * & * \\ \vdots & \vdots \\ * & * \end{matrix}}_{\text{coordinates of starts}} & \underbrace{\begin{matrix} * & * \\ * & * \\ \vdots & \vdots \\ * & * \end{matrix}}_{\text{coordinates of finishes}} & \underset{\text{indices}}{\begin{matrix} * \\ * \\ \vdots \\ * \end{matrix}} \end{array}\right)$$

and call it the *boundary intervals matrix.* The boundary intervals matrix is a numerical expression of the solution set H. In Sections 2.3.1 – 2.3.5, we will explain what information is kept in the matrix and how to derive and use it.

2.3.1 How to compose a subsystem of boundary inequalities?

There is a one-to-one correspondence between boundary intervals and boundary inequalities. Each boundary interval is generated by only one boundary inequality, and each boundary inequality generates exactly one boundary interval. The index of the boundary interval is the ordinal number in system (4) of the boundary inequality that generates the interval. The indices of all the boundary inequalities of (4) are listed in the last fifth column of the boundary intervals matrix, and the row size of the boundary intervals matrix is equal to the total number of the boundary inequalities in (4).

2.3.2 Does the solution polyhedron have a boundary?

A support cannot be the empty set, so the equality $\partial H = \varnothing$ in (7) means the absence of the elements in the index set I_b. Further, since I_b is the set of all indices in the fifth column of the boundary intervals matrix, I_b is empty if and only if the boundary intervals matrix has no rows. Therefore, *the absence of the boundary in the solution polyhedron H is equivalent to the absence of rows in the boundary intervals matrix for the inequality system determining H.* The solution polyhedron does not have a boundary only in two cases: (i) H is the empty set, or (ii) H coincides with the entire plane $\mathbb{R}^2$. The last case is equivalent to the condition

$$A = 0, \quad b \le 0. \tag{8}$$

We can summarize the above reasoning:

- *The solution polyhedron coincides with the entire plane $\mathbb{R}^2$ if and only if the boundary intervals matrix is empty, and condition (8) is satisfied.*
- *The solution polyhedron is empty if and only if the boundary intervals matrix is empty, and condition (8) is violated.*

  In particular, in case $A \neq 0$, the emptiness of the boundary intervals matrix is equivalent to the emptiness of the solution set H.

2.3.3 Is the solution polyhedron bounded?

First, we consider the situation when the boundary intervals matrix is empty, which corresponds to a solution set with no boundary points. Then the results of the previous Section 2.3.2 imply that the boundedness of the solution set H can be tested in the following way:

(a) the solution polyhedron is unbounded (coincides with the entire plane) when condition (8) holds true,

(b) the solution polyhedron is bounded (coincides with the empty set) when condition (8) is not valid.

Next, we consider the situation when the boundary intervals matrix is not empty, i. e., when the set H has boundary points. If the solution polyhedron having boundary points is unbounded, it has an edge with an endpoint at infinity. Hence from Proposition 7, at least one boundary interval with the endpoint coordinates $+\infty$ or $-\infty$ corresponds to such an edge. If the polyhedron having boundary points is bounded, the endpoints of all its boundary intervals have finite coordinates. So for the case when the boundary intervals matrix is not empty, recognizing whether the solution set is bounded can be organized as follows:

(a) when the first four columns of the boundary intervals matrix have at least one element “$+\infty$” or “$-\infty$”, the solution polyhedron is unbounded, or

(b) when there are no elements “$+\infty$” and “$-\infty$” in the first four columns of the boundary intervals matrix, the solution polyhedron is bounded.

2.3.4 How to get matrix of vertex coordinates?

The matrix of vertex coordinates is constructed from the boundary intervals matrix in the following way:

1. From Proposition 8, each vertex is the start of a boundary interval. Take the first two columns of the boundary intervals matrix in which the coordinates of the starts are specified. Alternatively, we can use the third and fourth columns specifying the coordinates of the finishes.
2. Delete the rows having elements “$+\infty$” or “$-\infty$”, i. e., all the rows that do not correspond to vertices according to Proposition 7.
3. A vertex can be the start of several boundary intervals. To avoid unnecessary repetitions of vertices, delete duplicates of rows from the constructed matrix.

As the result, all the vertices of the solution polyhedron are written out in the matrix obtained. Each row of the matrix represents a separate vertex as a pair of its coordinates in the form “(abscissa, ordinate)”. The number of rows in the matrix is equal to the number of vertices of the solution set.

2.3.5 How to construct a path around the solution polytope?

At this point, we assume that the solution set of system (4) is bounded and non-empty. From a matrix M of the boundary intervals, we can construct a closed path P around the solution polytope. The path is a sequence of vertices, moving from one vertex to the next, visiting all edges of the boundary (if there are edges), and returning to the initial point. The primary purpose of constructing the path is to help visualizing the solution polytope by standard drawing tools used in 2D visualization (see Section 3.2).

Geometry of the path. We examine paths from two, one, and zero dimensions of the polytope (see Fig. 7) and specify a sequence of vertices T_1, T_2, ... in each case.

I) The boundary of a two-dimensional polytope (Fig. 7, *a*) is a closed broken line with a finite number of sections that do not intersect. The line being closed means that starting from an arbitrary vertex and moving along the sections of the line, we return to the initial vertex. The line not intersecting itself means that the polytope always stays on the one side chosen at the start of the line during our motion. We fix the clockwise direction of the path so that the polytope is on the right of the line. The sequence of vertices that corresponds to the path has the form $\{T_1, T_2, T_3, \dots, T_1\}$.

II) A one-dimensional polytope is a line segment (Fig. 7, *b*). A closed path around it is a sequence of vertices $\{T_1, T_2, T_1\}$. In this case, the closed broken line of the path along the boundary is composed of two subsequent sections T_1T_2 and T_2T_1. When moving along the edge in one direction, one of the half-planes supporting the edge stays on the right, while moving in the opposite direction retains the other half-plane supporting the edge on the right.

III) A zero-dimensional polytope is a point (Fig. 7, *c*). A closed path along its boundary is a sequence $\{T_1\}$ of only one vertex. Such path does not have sections, and there is no actual broken line corresponding to it.

Path can be represented by boundary intervals. The path chosen in each of these cases can be made up of the boundary intervals:

I) For a two-dimensional polytope, the sections of the broken line that specifies the path are edges of the solution set. Every edge, by virtue of Proposition 5, serves as support of a boundary interval. Additionally, moving along the section coincides with the direction of the boundary interval.

II) A one-dimensional polytope coincides with its sole edge, say, T_1T_2. Proposition 5 suggests that the edge is support of at least two boundary intervals with opposite directions. One of them corresponds to the move from T_1 to T_2, and the other corresponds to the backward move from T_2 to T_1.

III) The boundary of a zero-dimensional polytope is a single point. By Proposition 6, it coincides with support of each boundary interval for the system that determines the polytope.

Pseudocode of the algorithm. We choose boundary intervals from the boundary intervals matrix in the appropriate order to arrange a path around the polytope. Using the boundary intervals selected, we represent the path as a sequence of vertices of the polytope. In Tab. 1, we give pseudocode for one of the possible algorithm. The assignment operator is "$\leftarrow$", and the sense of the variables used in Tab. 1 should be clear from their names and further explanation. We illustrate the pseudocode using Fig. 7.

The boundary intervals matrix $M = (M_{kl})$ is input of the algorithm. Since the solution set H is non-empty and bounded, the matrix M is not empty too (see Section 2.3.2); it has at least one row. On the other hand, as shown in Section 2.3.1, the

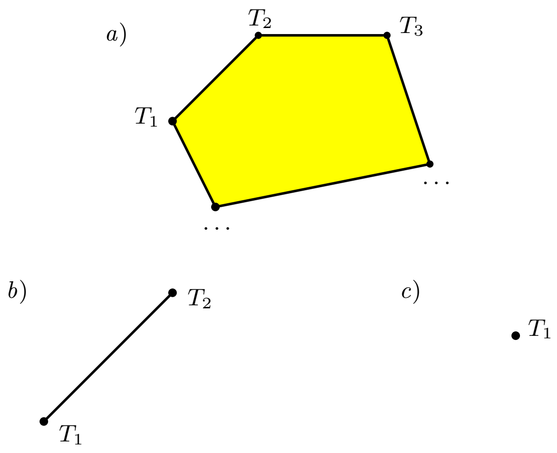


Figure 7: Non-empty polytopes in $\mathbb{R}^2$: *a*) 2-dimensional (bodily polytope), *b*) 1-dimensional (segment), *c*) 0-dimensional (point).

Table 1: Algorithm for constructing a closed path around the solution polytope

```
Input:   boundary intervals matrix $M = (M_{kl})$ of a polytope.
Output:  matrix $P$ in which the rows represent coordinates
         of subsequent verticies of a path around the polytope.
```

```
1:  WorkingMatrix $\leftarrow$ $M$;
2:  BeginningOfPath $\leftarrow$ $(M_{11}, M_{12})$;
3:  $P$ $\leftarrow$ BeginningOfPath;
4:  WorkingStart $\leftarrow$ BeginningOfPath;
5:  DO WHILE (number of rows in WorkingMatrix) $> 0$
6:      //  find the number $j$ of such row of WorkingMatrix
7:      //  for which WorkingStart coincides with the start
8:      DO $k = 1$ TO (number of rows in WorkingMatrix)
9:          IF (WorkingMatrix($k, 1$), WorkingMatrix($k, 2$)) $=$ WorkingStart
10:             $j$ $\leftarrow$ $k$;
11:             BREAK
12:         END IF
13:     END DO
14:     WorkingFinish $\leftarrow$ (WorkingMatrix($j, 3$), WorkingMatrix($j, 4$));
15:     IF WorkingFinish $\neq$ WorkingStart
16:         put WorkingFinish into the matrix $P$ as the last row;
17:         IF WorkingFinish $=$ BeginningOfPath
18:             //  a closed path $P$ has been constructed;
19:             //  end of the algorithm execution
20:             RETURN
21:         ELSE
22:             //  begin a new section of the path broken line
23:             //  from the end of the preceding section
24:             WorkingStart $\leftarrow$ WorkingFinish
25:         END IF
26:     END IF
27:     delete the $j$-th row from WorkingMatrix
28: END DO
```

number of rows in the boundary intervals matrix is equal to the number of boundary inequalities in system (4), so it does not exceed the total number of inequalities m in the system under consideration. The boundedness of the solution set also implies (see Section 2.3.3) that there are no elements "$+\infty$" or "$-\infty$" in M. Therefore from Proposition 7, all the endpoints of the boundary intervals are vertices.

The algorithm yields a two-column matrix P ("path matrix") whose rows represent subsequent vertices of the path, and each row gives two coordinates of the respective vertex.

To begin the path, we take the vertex T_1, the start in the first row of the boundary intervals matrix M. We put the coordinates (M_{11}, M_{12}) of the vertex into the first row of the matrix P. Then we assign T_1 to be the *working start* T_w, a point that we want to continue the path from it along an appropriate boundary interval. We keep processing the boundary intervals from the matrix M, storing intermediate results in a matrix `WorkingMatrix` that is initialized as M.

Each step of the algorithm attempts to find a boundary interval whose start is the working start T_w and to determine from the support of this boundary interval the next vertex T_{w+1} of the constructed path. A step of the algorithm consists of four actions:

1) Consider the rows of `WorkingMatrix` consecutively until we find a row j for which the start has the coordinates of the current working start T_w.
2) We call the finish in the j-th row, $\big($`WorkingMatrix`$(j,3)$, `WorkingMatrix`$(j,4)\big)$, the *working finish*.
3) From Proposition 4, coincidence of the working finish and working start means that the support of the boundary interval from the j-th row of `WorkingMatrix` is a vertex. Such a boundary interval does not engender a move along the boundary to a new vertex T_{w+1}. Therefore, it is not necessary in the construction of the path.

   Distinction between the working finish and the working start implies, in view of Proposition 4, that the support of the boundary interval from the j-th row of `WorkingMatrix` is an edge. In this case, we continue our path along the marked boundary interval, putting the working finish T_{w+1} as another row into the path matrix P. If in doing so, the working finish coincides with the beginning of the path T_1, the path has been closed, and the algorithm completes its work.

   If the working finish does not coincide with T_1, the path should be continued after taking the working finish as a new working start.
4) The boundary interval from the j-th row of `WorkingMatrix`, no matter whether we have used it or not, will not be involved in the further construction of the path, so we delete it from `WorkingMatrix`.

The steps of the algorithm are repeated as long as `WorkingMatrix` has rows or until the conditional operator in lines 17–20 of Tab. 1 breaks the execution when a closed path is built ahead of the natural termination of the algorithm.

Substantiation of the algorithm. To ensure that the algorithm, having performed a finite number of steps, constructs the matrix of the path vertices correctly, it is sufficient to examine which boundary intervals constitute the matrix M.

Suppose the dimension of the polytope H is zero (Fig. 7, c). All the boundary intervals from the matrix M have support T_1. It is both the start and finish for each. Therefore at every step, the algorithm chooses and deletes the first row from the current boundary intervals matrix. As the result, the number of steps taken by the algorithm equals the number of rows in the input boundary interval matrix. The algorithm completes its work, having added nothing to the beginning of the path T_1 recorded in path matrix P.

Finally, let us consider the remaining cases, when the dimension of the polytope H is either one or two (see Fig. 7, *a, b*). For each vertex T_w of the bodily polytope, there is a unique edge along which we can move to the next vertex (which we label T_{w+1}) retaining the polytope on the right. We know that the supports of the boundary intervals are all edges and possibly some vertices of the polytope (see Propositions 3 and 5), and that the boundary intervals can have equal supports. Therefore, in the boundary intervals matrix M, all the boundary intervals with the start T_w have T_w or T_{w+1} as the finish. Boundary intervals with the finish T_w need not be present, while a boundary interval with the finish T_{w+1} must be present. Hence, when processing the working start T_w, the algorithm performs one more step than the number of boundary intervals with the support T_w recorded in the initial boundary intervals matrix M before the first occurrence of the boundary interval that has T_wT_{w+1} as the supporting edge. At any rate, the algorithm finds the next path vertex T_{w+1} correctly.

In summary, the algorithm subsequently inserts all the vertices of the path into the matrix P, starting with the vertex $T_1 = (M_{11}, M_{12})$ and ending in it too, as the result of coinciding the current working finish T_{w+1} with the beginning of the path. As every step of the algorithm finds an appropriate row j and deletes it, the total number of steps executed by the algorithm, including the last incomplete step, does not exceed the number of rows in the original boundary intervals matrix M.

3 Visualization in $\mathbb{R}^2$

When designing software for visualization of polyhedral sets determined as the unions of finite numbers of the solution sets to linear inequality systems, we had to resolve several questions including

1) How to choose the drawing box?
2) How to depict a polytope?
3) How to depict an unbounded polyhedron?

Solving the first two problems resulted in the boundary intervals method. In this section, we describe our experience of resolving each of these questions in $\mathbb{R}^2$.

3.1 Choosing the drawing box

A *box* (a term borrowed from interval analysis) is a geometrical image of an interval vector in $\mathbb{R}^n$. It represents the solution set of a component-wise vector inequality $\underline{z} \le z \le \overline{z}$ with the unknown vector variable $z \in \mathbb{R}^n$ for some given $\underline{z}, \overline{z}$ from $\mathbb{R}^n$. The box is the direct (Cartesian) product of the intervals (closed segments) on the coordinate axes. Any interval can degenerate to a point. A *drawing box* is a coordinate range (direct product of the intervals over the coordinate axes) in which we will draw.

When visualizing a set, the main problem with choosing the drawing box is that our picture within the box should allow us to conceive the structure of the set clearly and unambiguously. Specifically, if the set visualized is bounded, it should be rendered entirely in the drawing box. If the set visualized is not bounded, we should be able to imagine what is outside the drawing box from what is depicted within the box.

For polyhedral sets in $\mathbb{R}^2$, the problem of choosing a drawing box is solvable because the boundary of solution sets has relatively simple structure. Except for a finite number of points, the boundary is composed of line segments. We say the *boundary is rectilinear at the point* y, if there exists a neighborhood of this point (an open ball centered at this point) where the boundary lies on a line going through y. This feature enables us to choose the drawing box so that the boundary is either absent or everywhere rectilinear outside the drawing box. The construction of the drawing box reduces to finding some special points, their total number being finite, that should be put into the box. We refer to these as "orientation points".

Orientation points. For a polyhedral set H, *orientation points* are points from H whose presence in the drawing box are sufficient to give us clear and complete information on the structure of H. The number of orientation points should not be large.

The concept of "orientation point" is not strict and formal; for the same subclass of polyhedral sets, we can take the collection of orientation points at will, depending on programming convenience, easiness of description, or any other reasons. Our experience allows us to make some recommendations about the choice of the orientation points using the boundary intervals matrix.

A collection of the orientation points of a polyhedral set should include all the points from the boundary where it is not a part of a straight line, but in general, such "non-straight-line points" are not sufficient. They cannot exhaust the collection of the orientation points, particularly since we should depict polyhedra that may not have such points (straight lines, half-spaces, and strips). We recommend the following rules for collecting the orientation points:

1. *A polyhedron with vertices.* This case corresponds to the boundary intervals matrix having a start, both of whose coordinates are finite. Ideally, the orientation points should be the set of all polyhedron vertices. In Section 2.3.4, we have explained how to get the vertex coordinates matrix.

2. *A polyhedron without vertices, but with a boundary.* This is the case when the boundary intervals matrix is not empty, but its every start has an infinite coordinate. A half-plane, a strip, and a straight line are polyhedra with boundaries, but without vertices. It is convenient to take as the orientation points projections of the origin of coordinates onto the boundary straight lines; see the derivation of the formula for such projection in Section 2.2.2. We thus find all the points $\tilde{x}_i = b_i A_{i:}^\top / \|A_{i:}\|_2^2$, where the index i runs through the numbers of the boundary inequalities recorded in the fifth column of the boundary intervals matrix. After removing repetitions, we have one orientation point for a half-plane and a straight line and two orientation points for a strip.

3. *A polyhedron without a boundary.* This is the case when the boundary intervals matrix is empty. Polyhedra without boundaries are the empty set (providing that condition (8) is violated) and the entire space $\mathbb{R}^2$ (providing that condition (8) is satisfied). It is useful to agree that the empty set has no orientation points, while the only orientation point of the space $\mathbb{R}^2$ is the origin of its coordinates. Anticipating our further explanation, under the above agreement, we say that the empty set is the only polyhedral set without orientation points, and depicting it results in a special output message "the visualized set is empty". The whole space as the solution set appears as a completely colored drawing box.

4. *A polyhedral set for which the intersection with every orthant is determined by a system of linear inequalities.* Under these conditions, the intersection of the polyhedral set with a separate orthant is either empty (see case 3) or a polyhedron with vertices (see case 1). For each orthant, we can find orientation points of the intersection and then unite them to get a collection of vertices of all the polyhedra determined by the specified systems of linear inequalities.

5. *An arbitrary polyhedral set.* In this most general case, we know only the boundary intervals matrix for each polyhedron H^k that forms the polyhedral set, and no other information is available. In this situation, constructing a collection of the orientation points is a step-by-step process. Let the polyhedron H^k be determined by a system of linear inequalities $A^k x \geq b^k$. We perform the following sequence of instructions:

Stage 1. If the entire space $\mathbb{R}^2$ is present among the polyhedra forming the polyhedral set, the origin of coordinates should be taken as the orientation point of the polyhedral set. Otherwise, go to Stage 2.

Stage 2. For each polyhedron H^k of the polyhedral set, consider the collection of its orientation points obtained according to the rules for cases 1–3 above. If the orientation point v of H^k is not an interior point of another polyhedron H^l (i. e., for every $l \neq k$, the strict component-wise inequality $A^l v > b^l$ does not hold), we add v to the collection of orientation points of the polyhedral set.

After examining all the orientation points of all the polyhedra, go to Stage 3.

Stage 3. Take every pair of polyhedra H^k and H^l forming the polyhedral set. We consider the intersection of each unbounded support of the boundary intervals of H^k with each unbounded support of the boundary intervals of H^l.

The intersection point can be found as follows. Let i be the index of a boundary interval for the polyhedron H^k, having an infinite coordinate in any of its endpoints, and let j be an analogous index for the polyhedron H^l. We compute the point $\tilde{x}$ at which the lines $A^k_{i:}x = b^k_i$ and $A^l_{j:}x = b^l_j$ intersect. If both the systems $A^k\tilde{x} \geq b^k$ and $A^l\tilde{x} \geq b^l$ are satisfied, the point $\tilde{x}$ is the intersection of the corresponding supports. Otherwise, the supports do not intersect each other.

If the point $\tilde{x}$ obtained in the intersection is not an interior point for another polyhedron H^q, $q \notin \{k, l\}$, we add the point to the collection of orientation points of the initial polyhedral set.

Interval hull of the orientation points. After the coordinates (v^i_1, v^i_2) of the orientation points v^i, $i = 1, 2, \ldots$, are found, we seek the minimal axis-aligned box that contains all the orientation points. It is known to be the interval hull of the orientation points, i. e., the interval box

$$\Big[\min_i v^i_1,\, \max_i v^i_1\Big] \times \Big[\min_i v^i_2,\, \max_i v^i_2\Big],$$

where "×" is the direct (Cartesian) product of the sets.

Drawing box. The interval hull of the orientation points itself is not suitable to be a drawing box. On one hand, it can be just a point or a line segment. It is really impossible to discern anything in such non-solid drawing boxes. On the other hand, the interval hull of the orientation points may not allow us to get an idea of how the solution set behaves "at infinity". For example, Fig. 8 shows two different unbounded polyhedra having the same vertices, which cannot be distinguished from the picture within the interval hull of their orientation points. Hence, the drawing box should be constructed so that it is essentially larger than the interval hull of all the orientation points and contains the hull in its interior.

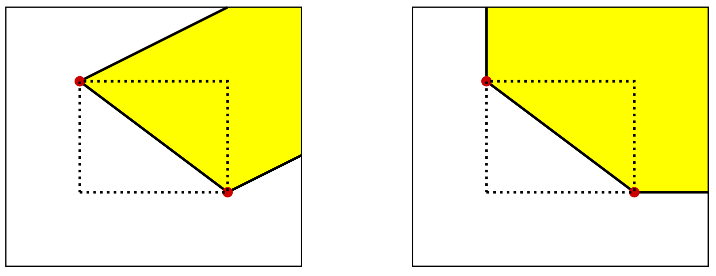
3.2 Drawing polytopes

To visualize polytopes, it is convenient to use standard graphical procedures that draw a polytope from its vertex matrix determining a closed path around the polytope. In particular, such procedures exist in most modern software systems for computer mathematics, e. g., the function `fill` in MATLAB [7] and Octave [3] and the analogous function `xfpoly` in the open-source system Scilab [14]. When using these functions, it is possible to change the color and transparency of the interior domain of the polytope.

If necessary, the vertices of the polytope can be rendered by a separate procedure that draws a set of points, such as the function `scatter` in MATLAB and Octave. Highlighting the vertices is useful in various aspects. This way, we will not lose polytopes that coincide with points. Additionally, we can clearly represent polytopes that are line segments. In the next subsection, we will show that any unbounded polyhedron can be depicted as a polytope. In doing so, highlighting the vertices helps to distinguish meager polyhedra such as line segments, rays and entire straight lines.

3.3 Drawing unbounded polyhedra

In school, we draw straight lines and rays during geometry lessons with the same ease as line segments, and drawing angles is almost as simple as drawing triangles. We depict unbounded sets as bounded, with some slight, but significant, differences. For instance,



🟨	polyhedron
●	orientation point
⋯⋯	interval hull of orientation points
——	drawing box

Figure 8: Different polyhedra with the same interval hull of their orientation points.

the origin of a ray and endpoints of a segment are marked by noticeable points. We can do the same with unbounded polyhedra, cutting an unbounded polyhedron to a polytope, but arranging our picture to give clear signals that an unbounded polyhedron is visualized.

Cut box. To be visualized within a bounded area, an unbounded polyhedron is cut by a *cut box*, chosen to contain the drawing box in its interior. For instance, we can translate each side of the drawing box from its center (see Fig. 9).

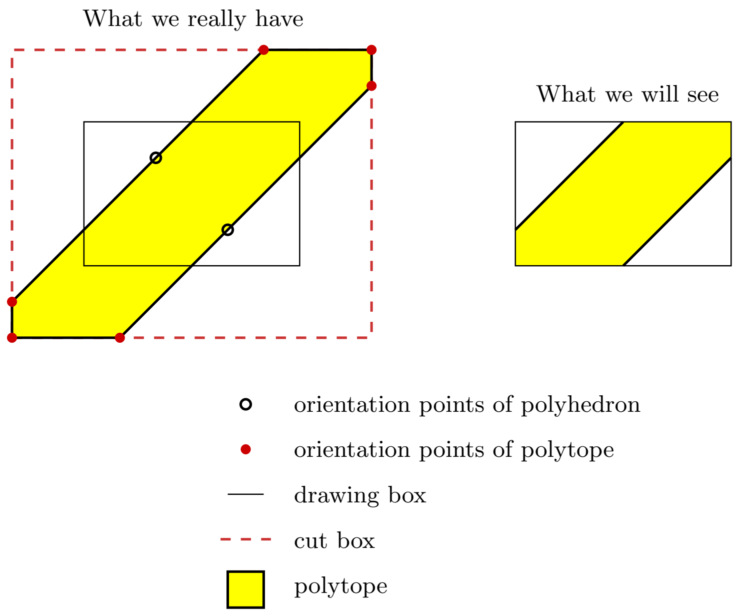

What we really have
What we will see
orientation points of polyhedron
orientation points of polytope
drawing box
cut box
polytope

Figure 9: Cutting and drawing a strip as an example of an unbounded polyhedron.

Cutting. If an unbounded polyhedron in $\mathbb{R}^2$ is determined by a system of linear inequalities $Ax \geq b$, with $A \in \mathbb{R}^{m\times 2}$, $x = (x_1, x_2)^\top$, and $b \in \mathbb{R}^m$, the intersection of the polyhedron with the cut box $[\underline{x}_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2]$ is a polytope described by the system of linear inequalities

$$\begin{cases} Ax \geq b, \\ \underline{x}_1 \leq x_1 \leq \overline{x}_1, \\ \underline{x}_2 \leq x_2 \leq \overline{x}_2, \end{cases}$$

or in canonical matrix form,

$$\begin{pmatrix} \multicolumn{2}{c}{A} \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq \begin{pmatrix} b \\ \underline{x}_1 \\ \underline{x}_2 \\ -\overline{x}_1 \\ -\overline{x}_2 \end{pmatrix}. \tag{9}$$

We can draw a polytope instead of a polyhedron. A general technique for depicting unbounded polyhedra is

Compute the boundary intervals matrix for the extended system of linear inequalities (9),

Use the boundary interval matrix to construct a vertex matrix that represents a closed path around the solution polytope for system (9), and

Draw the polytope in the drawing box found for the initial unbounded polyhedron.

Visual differences between bounded and unbounded sets. Since the cut box of an unbounded polyhedron contains the drawing box, the image of any unbounded polyhedron reaches the boundary of the drawing box.

A bounded polyhedron is included in the interval hull of its orientation points, and this hull lies strictly in the interior of the drawing box. Therefore, any bounded polyhedron cannot have points at the boundary of the drawing box.

If we have visualized a polyhedron and the picture has points at the boundary of the drawing box, then the polyhedron is unbounded. Otherwise, if a rendered polyhedron does not have points at the boundary of the drawing box, the polyhedron is bounded. Similar statements hold for general polyhedral sets.

4 Additional Facts about Polyhedra and Their Visualization in $\mathbb{R}^3$

In three-dimensional space, as two dimensions, the main questions that arise in computer visualization of polyhedral sets are how to construct the drawing box, how to draw a polytope, and how to draw unbounded polyhedra. Before discussing their solutions, we need to recall further properties of polyhedra in $\mathbb{R}^3$.

4.1 Additional facts about polyhedra

A polyhedron in $\mathbb{R}^3$ is described by a system of linear inequalities

$$Ax \geq b, \quad A \in \mathbb{R}^{m\times 3},\ x \in \mathbb{R}^3,\ b \in \mathbb{R}^m,\ m \in \mathbb{N}. \tag{10}$$

Deleting zero rows from the matrix A. In system (10), the matrix A may have zero rows. When answering various questions about the solution polyhedron, it is useful to detect zero rows in the system, which by itself offers insight into the polyhedron structure. If a question under study is not resolved yet, we delete the zero rows and address the question to the system (10) having only non-zero rows in A.

Step-by-step deletion of the zero rows is organized in the following way. Consider the rows of the matrix A. If $A_{i:} = (\,0\ 0\ 0\,)$, and $b_i > 0$, the solution set to the inequality $A_{i:}x \geq b_i$ and hence to the whole system (10) is empty. The deletion process ends when we encounter such an inequality, since then we know what the solution set is and can answer any question about it. If $A_{i:} = (0\ 0\ 0)$, and $b_i \leq 0$, the solution set to the inequality $A_{i:}x \geq b_i$ is the entire space $\mathbb{R}^3$. We can delete such inequality from the system and the respective row from A with no effect on the solution set of (10).

When the deletion process completes normally without a termination that proves the emptiness of the solution set, we look at the resulting "cleared system". If no rows (inequalities) remain, the initial systems satisfies condition (8), and its solution set is the entire space $\mathbb{R}^3$. If there are rows (inequalities) in the "cleared system", we must examine the resulting system of the form (10) without zero rows.

Contribution of a separate inequality to the boundary. Let H be a solution set to system (10) without zero rows in the matrix A. Let $\widetilde{H}_i$ denote a contribution of the i-th inequality to the boundary of H (see Section 2.1). The polyhedron $\widetilde{H}_i$ is the intersection of the plane $A_{i:}x = b_i$ with the solution set H, $\widetilde{H}_i = \{\, x \in \mathbb{R}^3 \mid A_{i:}x = b_i,\ Ax \geq b \,\}$.

The sets $\widetilde{H}_i$ play a special role in our method for the space $\mathbb{R}^3$. Many questions about the solution polyhedron H, such as testing its non-emptiness, choosing orientation points, and even drawing it, decompose into similar questions about the sets $\widetilde{H}_i$. On the other hand, for each set $\widetilde{H}_i$, these questions can be solved using the boundary intervals matrix (as shown in Section 2.3), since the set $\widetilde{H}_i$ is described in the internal coordinates of the plane $A_{i:}x = b_i$ by the system of inequalities of the form (4) with two unknowns. It only remains to select the internal coordinates in the plane $A_{i:}x = b_i$.

Constructing the internal coordinate system $\widetilde{O}y_1y_2$ in the plane $A_{i:}x = b_i$ amounts to carrying out the following steps, as in the two-dimensional case:

- The origin $\widetilde{O}$ of the internal coordinate system is the projection of the origin $(0, 0, 0)$ of the initial ("external") coordinate system onto the plane $A_{i:}x = b_i$,

$$\widetilde{O} = b_i A_{i:}^{\top} / \|A_{i:}\|_2^2, \tag{11}$$

  where $\|A_{i:}\|_2$ is the Euclidean norm of the row-vector $A_{i:}$ (the derivation of the formula (11) can be found in Section 2.2.2).

- As the unit vector e_1 corresponding to the internal coordinate y_1, we take the largest orthogonal projection of the vector $A_{i:}^{\top}$, rotated by 90° (in any direction).

  The projection of the vector $A_{i:}^{\top}$ onto the coordinate plane $x_k = 0$ has the squared length $\sum_{l \neq k} |A_{il}|^2$, so it is the longest one for k such that $|A_{ik}| = \min\{|A_{i1}|, |A_{i2}|, |A_{i3}|\}$. For example, if $\min\{|A_{i1}|, |A_{i2}|, |A_{i3}|\}$ is attained at $|A_{i1}|$ for a given i, the largest projection of the vector $A_{i:}^{\top}$ is $(0, A_{i2}, A_{i3})^{\top}$, and either of the vectors $(0, A_{i3}, -A_{i2})^{\top}$ or $(0, -A_{i3}, A_{i2})^{\top}$ can serve as the unit vector of the internal coordinate y_1.

- As the unit vector e_2 corresponding to the internal coordinate y_2, we get the cross (vector) product of the normal $A_{i:}^{\top}$ to the plane $A_{i:}x = b_i$ and the unit vector e_1. The cross product of two linearly independent vectors is a vector that is perpendicular to both, with length equal to the area of a parallelogram with the multiplied vectors as sides.

In the internal coordinates of the plane $A_{i:}x = b_i$, the polyhedron $\widetilde{H}_i$ is described by a system of inequalities

$$A\left(\widetilde{O} + e_1y_1 + e_2y_2\right) \geq b,$$

or

$$\begin{pmatrix} Ae_1 & Ae_2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \geq b - A\widetilde{O},$$

where $\widetilde{O}$ is given by (11). If in the internal coordinate system of the plane $A_{i:}x = b_i$, a point has the coordinates (y_1, y_2), its coordinates are $(\widetilde{O}+e_1y_1+e_2y_2)^\top$ in the initial system $Ox_1x_2x_3$.

Is the solution polyhedron bounded? The solution set H to the system of inequalities (10) without zero rows in A cannot coincide with the entire space $\mathbb{R}^3$ because the polyhedron H is bounded if and only if its boundary ∂H is bounded. From the representation (2), the boundary ∂H consists of the contributions made by separate inequalities, i. e., $\partial H = \bigcup_i \widetilde{H}_i$. This suggests that the boundedness of the boundary ∂H is equivalent to that all the sets $\widetilde{H}_i$ are simultaneously bounded.

Testing the boundedness of any polyhedron $\widetilde{H}_i$ can be organized as follows:

1) Pass to the internal coordinates of the plane $A_{i:}x = b_i$;
2) Compute the boundary intervals matrix for $\widetilde{H}_i$ and use the results of Section 2.3.3 to test the boundedness of the set $\widetilde{H}_i$.

4.2 Construction of the drawing box

Since the boundary of a polyhedral set in $\mathbb{R}^3$ has a finite number of points where it does not resemble a plane, a dihedral angle, or a straight line, the problem of constructing the drawing box in $\mathbb{R}^3$ appears to be solvable. However, we know of no procedure for collecting the orientation points for a general polyhedral set. For this reason, we offer recommendations for choosing orientation points only for two specific (although very important) cases: (i) an arbitrary polyhedron and (ii) a polyhedral set determined by a system of linear inequalities in each orthant of $\mathbb{R}^3$.

Orientation points of an arbitrary polyhedron. We delete from the system of inequalities (10) all the rows i for which $A_{i:} = (0\ 0\ 0)$. During the deletion process, if the solution polyhedron is empty, we say it has no orientation points. If the solution set to the initial system is $\mathbb{R}^3$, the origin of the coordinates is taken as the only orientation point. If we get a system (10) without zero rows, we search for its nontrivial orientation points by finding orientation points of the solution polyhedron $\widetilde{H}_i$ for each inequality $A_{i:}x \geq b_i$ and uniting all the sets we find. We choose orientation points of the set $\widetilde{H}_i$ following these steps:

1. Pass to the internal coordinates of the plane $A_{i:}x = b_i$;
2. Identify orientation points similar to the case of two unknowns; and
3. Transform the orientation points from the internal coordinates to the original ones.

Orientation points of a polyhedral set described in each orthant by a system of linear inequalities. For each one of the eight orthants in $\mathbb{R}^3$, we choose the orientation points of the solution polyhedron for the corresponding system of linear inequalities in the way previously specified for arbitrary polyhedra. Then we take the union of all these eight sets of the orientation points.

Drawing box. Similar to the case of two unknowns, we find orientation points and take their interval hull. The drawing box is constructed so that it is larger than the interval hull of the orientation points and contains the hull in its interior.

4.3 Drawing polytopes

In this subsection, we discuss how to draw the solution polytope for system (10) in which the matrix A has no zero rows.

In the two-dimensional case, visualization of a polytope amounts to drawing its boundary, and the interior may be colored to increase our perception of the figure. In the three-dimensional case, visualization of the interior is unnecessary in general, since when looking at a polytope, we can see only its boundary in the actual space.

From (2), the boundary of the solution polytope consists of contributions $H \cap \partial H_i$, $i = 1, 2, \ldots, m$, made by separate inequalities. If the matrix A of system (10) has no zero rows, the contribution of the i-th inequality to the boundary of the solution polytope H is the intersection of the plane $A_{i:}x = b_i$ with H. We denote this intersection by $\widetilde{H}_i$. Hence, depicting a polytope H in $\mathbb{R}^3$ reduces to depicting all the polytopes $\widetilde{H}_i$. To draw the polytope $\widetilde{H}_i = \{\, x \in \mathbb{R}^3 \mid A_{i:}x = b_i,\ Ax \geq b \,\}$, we perform the following:

1. Pass to the internal coordinates of the plane $A_{i:}x = b_i$.
2. Determine a path around the polytope $\widetilde{H}_i$, as we did for two unknowns. If the polytope $\widetilde{H}_i$ is empty, do not draw it.
3. Transform the path vertices from the internal coordinates to the original ones.
4. Draw the polytope $\widetilde{H}_i$ by any function that constructs a "flat" polytope in $\mathbb{R}^3$ from its path vertices (for instance, the function `fill3` in MATLAB).

4.4 Drawing unbounded polyhedra

As in the two-dimensional case, an unbounded polyhedron is rendered in a cut box. In principle, we draw it as a polytope. However, when drawing an unbounded polyhedron, we should add details to help convey that the polyhedron is unbounded.

A natural requirement is that the cut box include the interval hull of the orientation points in its interior. However, unlike the two-dimensional case, the cut box must lie within the drawing box so that we can see the faces of the polyhedron created by the cut. To distinguish them from actual faces, i. e., those inherent in the polyhedron itself, we mark the cut faces by another color. The difference in color between the real faces and the cut faces is the first detail that will help us to distinguish an unbounded polyhedron from a polytope. Unfortunately, this device works only for bodily (solid) polyhedra.

Yet another visualization detail especially useful for thin polyhedra is a cut box equal to the drawing box. The advantage is that only unbounded polyhedra reach the boundary of the drawing box, while polytopes lie in the interior of the drawing box. By rotating the three-dimensional picture, we can see whether the visualized polyhedron reaches the boundary of the drawing box and infer whether the polyhedron is bounded or not.

5 Conclusion

The visualization algorithms described here have been implemented by the author in MATLAB as open source software packages `lineq2` [15], `IntLinInc2D` [19], and `IntLinInc3D` [20], available at the author's web-page `http://www.nsc.ru/interval/sharaya`. Russian language versions of these packages are called `lineq`, `IntLinIncR2`, and `IntLinIncR3`, respectively. The interested reader of the manuals for these packages will find additional insights into the boundary intervals method, including examples and visualization results. These manuals and source codes continue the explanations of this paper, which gives only the foundations of the boundary intervals method.

Another implementor of the method may encounter problems relating to the algorithms and specific programming. Solving these problems depends on the experience and preferences of those who are implementing the method, while the manuals and source codes of our packages may provide helpful suggestions for particular solutions. For example, an implementor of the ideal (exact) mathematical algorithm needs to accommodate the effect of inaccuracy of the digital computation. In our boundary intervals method, the concepts of supporting hyperplane, support, boundary inequalities, and boundary intervals are based on satisfying equalities, which are unstable under data perturbations. If substantial variation in the input data (for polyhedra, the matrix and right-hand side vector of a system of linear inequalities) is allowed and the computing time does not matter, we can use symbolic computation and exact rational arithmetic. Alternatively, if we need a picture quickly, given a restriction on the range of the input data, we must manage with approximate (e. g., floating-point) computation, coarsening comparisons of not exactly computed values in the "if" statements.

Our open source visualization packages [15, 19, 20] present possible variants of the answers to many algorithmic and programming questions.

Acknowledgements. I thank Prof. Dr. Walter Krämer, Prof. Dr. Siegfried Rump, Dr. Evgenija Popova, and Prof. Zenon Kulpa, whose tools and approaches to visualization of the solution sets of interval linear algebraic systems deeply impressed me and helped me develop the boundary intervals method. I thank all those who consulted me on details and capabilities of the visualization techniques for the solution sets to linear systems of algebraic inequalities. I extend my special gratitude to Dr. Nikolai Zolotykh (Lobachevsky State University of Nizhni Novgorod, Russia).

This research was supported by the President Grant for Governmental Support of the Leading Scientific Schools (grant No. 5007.2014.9).

References

[1] E.D. Bloch. *Polygons, Polyhedra, Patterns & Beyond.* Spring 2015. Electronic lecture notes, available at `http://math.bard.edu/bloch/math107_notes.pdf`

[2] P.R. Cromwell. *Polyhedra.* Cambridge University Press, 1997. Reprinted 2001.

[3] GNU Octave. Available at `http://www.gnu.org/software/octave`

[4] B. Grünbaum. Are your polyhedra the same as my polyhedra? In: *Discrete and Computational Geometry: The Goodman-Pollack Festschrift.* B. Aronov, S. Basu, J. Pach, and M. Sharir, eds. Springer, New York, 2003, pp. 461–488.

[5] INTLAB — INTerval LABoratory, the Matlab toolbox for reliable computing. Available at `http://www.ti3.tu-harburg.de/rump/intlab/`

[6] J.L. Kelley. *General Topology.* Springer, 1975.

[7] Matlab — The language of technical computing. Available at `http://www.mathworks.com/products/matlab`

[8] W. Krämer. `intpakX` — an interval arithmetic package for Maple. In: *Scientific Computing, Computer Arithmetic and Validated Numerics, 2006. Proceedings of SCAN 2006, 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, Duisburg, Germany, September 26–29, 2006.* IEEE Computer Society Press, 2007, page 27. DOI: 10.1109/SCAN.2006.29

[9] W. Krämer. Computing and visualizing solution sets of interval linear systems, *Serdica Journal of Computing*, Vol. 1 (2007), No. 4, pp. 455–468. Available at `http://serdica-comp.math.bas.bg/index.php/serdicajcomputing/article/download/33/30`

[10] E. Popova, W. Krämer. Visualization of parametric solution sets. Preprint BUW-WRSWT 2006/10, Bergische Universität Wuppertal, 2006. Available at `http://www2.math.uni-wuppertal.de/wrswt/preprints/prep_06_10-2.pdf`

[11] E. Popova, W. Krämer. Visualizing parametric solution sets, *BIT Numerical Mathematics*, Vol. 48 (2008), Issue 1, pp. 95–115. DOI: 10.1007/s10543-007-0159-3

[12] R.T. Rockafellar. *Convex Analysis.* Princeton, NJ, Princeton University Press, 1997. (Reprint of the 1979 Princeton mathematical series 28 ed.)

[13] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1998.

[14] Scilab — free and open source software for numerical computation. Available at `http://www.scilab.org`

[15] Irene A. Sharaya. `lineq2` — a software package for visualization of the solution sets to systems of linear inequalities. Version for Matlab: Release 25.04.2012. Available at `http://www.nsc.ru/interval/sharaya/index.html#codes`

[16] I.A. Sharaya. Boundary intervals and visualization of AE-solution sets for interval system of linear equations. In: *15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics — SCAN'2012, September 23–29, 2012, Novosibirsk, Russia. Book of Abstracts*, Institute of Computational Technologies, Novosibirsk, 2012. P. 166–167. Available at `http://conf.nsc.ru/files/conferences/scan2012/140000/scan2012Abstracts.pdf`

[17] I.A. Sharaya. Boundary intervals and visualization of AE-solution sets for interval system of linear equations [online presentation] *15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics — SCAN'2012, Novosibirsk, Russia, September 23–29, 2012.* Available at `http://conf.nsc.ru/files/conferences/scan2012/142985/Sharaya-scan2012.pdf`

[18] Irene A. Sharaya. `AEsolset.ps` — a PostScript program for visualization of AE-solution sets to interval linear 2×2-systems of equations. Available at `http://www.nsc.ru/interval/Programing/AEsolset.ps`

[19] Irene A. Sharaya. `IntLinInc2D` — a software package for visualization of the solution sets to interval linear systems of relations with two unknowns. Version for Matlab: Release 01.09.2014. Available at `http://www.nsc.ru/interval/sharaya/index.html#codes`, `http://www.nsc.ru/interval/Programing`.

[20] Irene A. Sharaya. `IntLinInc3D` — a software package for visualization of the solution sets to interval linear systems of relations with three unknowns. Version for Matlab: Release 01.09.2014. Available at `http://www.nsc.ru/interval/sharaya/index.html#codes`, `http://www.nsc.ru/interval/Programing`.

[21] I.A. Sharaya. Boundary intervals method for visualization of polyhedral solution sets, *Computational Technologies*, Vol. 20 (2015), No. 1, pp. 75–103. (in Russian) Available at `http://www.nsc.ru/interval/sharaya/`

[22] S.P. Shary. A new technique in systems analysis under interval uncertainty and ambiguity, *Reliable Computing*, Vol. 8 (2002), No. 5, pp. 321–419. Available at `http://www.nsc.ru/interval/shary/Papers/ANewTech.pdf`

[23] V.A. Yemelichev, M.M. Kovalev, M.M. Kravtsov. *Polytopes, Graphs, and Optimization.* Translated by G.H. Lawden. Cambridge, Cambridge University Press, 1984.