Reducing the Computation of Convex Hulls to Linear Programming

Chirantan Mukherjee *joint with S. Bhatt, R. J. Jing, Y. Lei and M. Moreno Maza*

Western Science

ORCCA
Ontario Research Centre for Computer Algebra

Plan

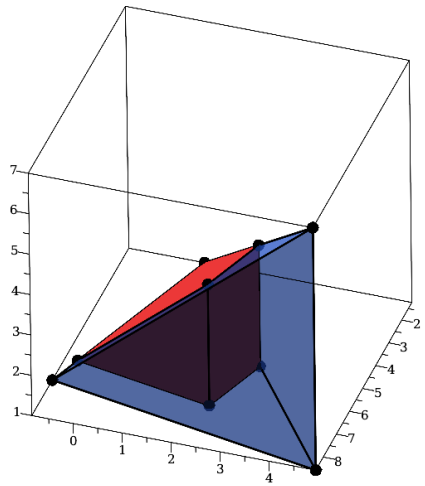
Overview

The wrapping procedure

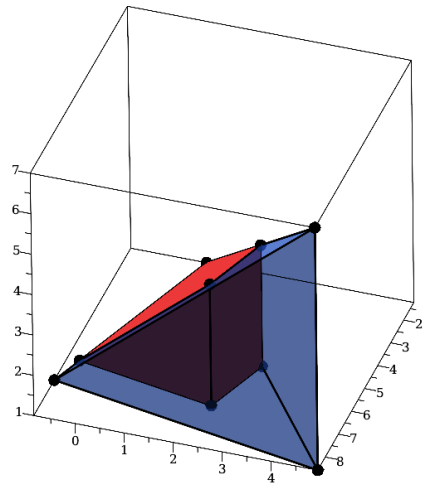
Computing convex hulls

Plan

Motivating example



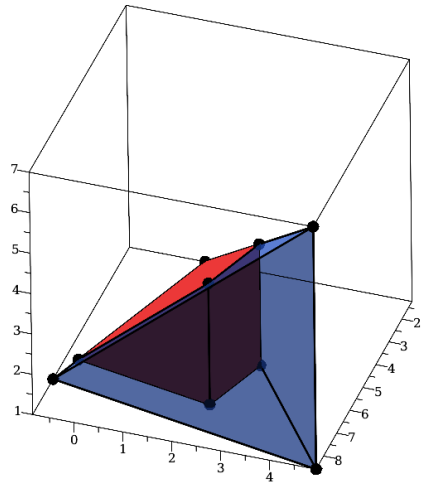
Motivating example



$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

$$np := \begin{cases} \left\{\frac{1}{3}\,n + \frac{1}{2}\,n^2 + \frac{1}{6}\,n^3\right\} & \mathrm{And}(-m + n = 0, 0 \le -2 + n) \\ \{1\} & \mathrm{And}(m - 1 = 0, n - 1 = 0) \\ \left\{\frac{1}{3}\,n + \frac{1}{2}\,n^2 + \frac{1}{6}\,n^3\right\} & \mathrm{And}(0 \le -2 + n, 0 \le m - n - 1) \\ \{1\} & \mathrm{And}(n - 1 = 0, 0 \le m - 2) \\ \left\{\frac{1}{3}\,m + \frac{1}{2}\,m\,n + \frac{1}{2}\,n\,m^2 - \frac{1}{3}\,m^3\right\} & \mathrm{And}(0 \le m - 2, 0 \le n - 3, 0 \le -m + n - 1) \\ \{4 + n\} & \mathrm{And}(m - 1 = 0, 0 \le -2 + n) \end{cases}$$

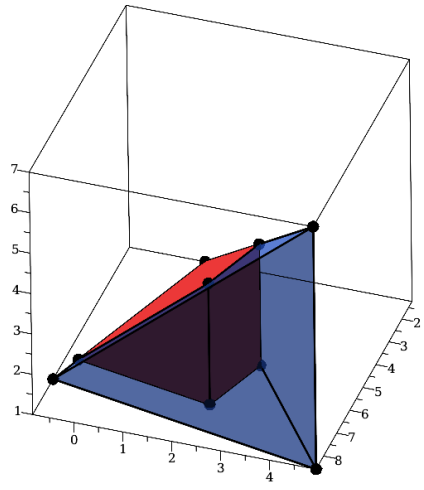
Counting the integer points of $1 \le i \le n, 1 \le j \le m, j \le i, 1 \le k \le j$.



$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m + n = 0, 0 \le -2 + n) \\ \{1\} & \text{And}(m - 1 = 0, n - 1 = 0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2 + n, 0 \le m - n - 1) \\ \{1\} & \text{And}(n - 1 = 0, 0 \le m - 2) \\ \left\{\frac{1}{3}m + \frac{1}{2}m\,n + \frac{1}{2}n\,m^2 - \frac{1}{3}m^3\right\} & \text{And}(0 \le m - 2, 0 \le n - 3, 0 \le -m + n - 1) \\ \{4 + n\} & \text{And}(m - 1 = 0, 0 \le -2 + n) \end{cases}$$

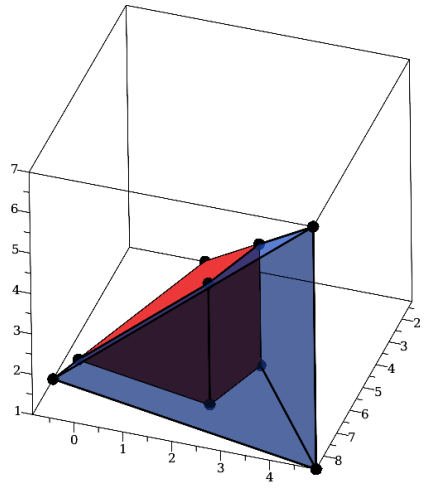
- Problems involving parametric polyhedra e.g. counting the integer points of $1 \le i \le n, 1 \le j \le m, j \le i, 1 \le k \le j$ tends to split more than necessary.



$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m + n = 0, 0 \le -2 + n) \\ \{1\} & \text{And}(m - 1 = 0, n - 1 = 0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2 + n, 0 \le m - n - 1) \\ \{1\} & \text{And}(n - 1 = 0, 0 \le m - 2) \\ \left\{\frac{1}{3}m + \frac{1}{2}m\,n + \frac{1}{2}n\,m^2 - \frac{1}{3}m^3\right\} & \text{And}(0 \le m - 2, 0 \le n - 3, 0 \le -m + n - 1) \\ \{4 + n\} & \text{And}(m - 1 = 0, 0 \le -2 + n) \end{cases}$$

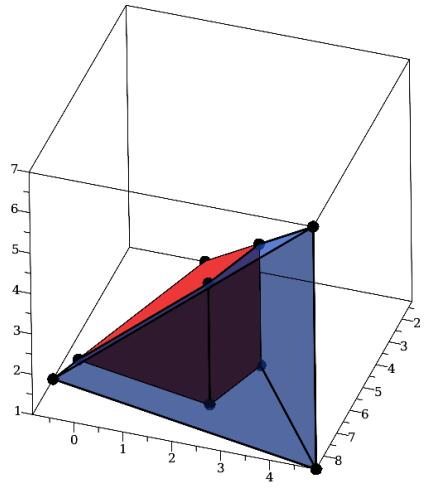
- Problems involving parametric polyhedra e.g. counting the integer points of $1 \le i \le n, 1 \le j \le m, j \le i, 1 \le k \le j$ tends to split more than necessary.
- Clearly, some cases are special cases of other cases.



$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

$$np := \begin{cases} \left\{\frac{1}{3}\,n + \frac{1}{2}\,n^2 + \frac{1}{6}\,n^3\right\} & \mathrm{And}(-m + n = 0,\ 0 \le -2 + n) \\ \{1\} & \mathrm{And}(m - 1 = 0,\ n - 1 = 0) \\ \left\{\frac{1}{3}\,n + \frac{1}{2}\,n^2 + \frac{1}{6}\,n^3\right\} & \mathrm{And}(0 \le -2 + n,\ 0 \le m - n - 1) \\ \{1\} & \mathrm{And}(n - 1 = 0,\ 0 \le m - 2) \\ \left\{\frac{1}{3}\,m + \frac{1}{2}\,m\,n + \frac{1}{2}\,n\,m^2 - \frac{1}{3}\,m^3\right\} & \mathrm{And}(0 \le m - 2,\ 0 \le n - 3,\ 0 \le -m + n - 1) \\ \{4 + n\} & \mathrm{And}(m - 1 = 0,\ 0 \le -2 + n) \end{cases}$$

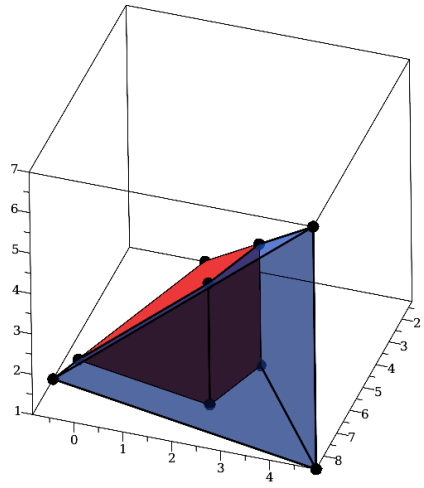
- Problems involving parametric polyhedra e.g. counting the integer points of $1 \le i \le n, 1 \le j \le m, j \le i, 1 \le k \le j$ tends to split more than necessary.
- Clearly, some cases are special cases of other cases.
- To combine two cases, the disjunction of their systems of constraints should define a polyhedral set, which must be the convex hull of the union of the zero sets of these systems of constraints.



$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & And(-m + n = 0, 0 \le -2 + n) \\ \{1\} & And(m - 1 = 0, n - 1 = 0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & And(0 \le -2 + n, 0 \le m - n - 1) \\ \{1\} & And(n - 1 = 0, 0 \le m - 2) \\ \left\{\frac{1}{3}m + \frac{1}{2}m\,n + \frac{1}{2}n\,m^2 - \frac{1}{3}m^3\right\} & And(0 \le m - 2, 0 \le n - 3, 0 \le -m + n - 1) \\ \{4 + n\} & And(m - 1 = 0, 0 \le -2 + n) \end{cases}$$

- Problems involving parametric polyhedra e.g. counting the integer points of $1 \le i \le n, 1 \le j \le m, j \le i, 1 \le k \le j$ tends to split more than necessary.
- Clearly, some cases are special cases of other cases.
- To combine two cases, the disjunction of their systems of constraints should define a polyhedral set, which must be the convex hull of the union of the zero sets of these systems of constraints.
- Therefore, computing the convex hull of finitely many polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$ is a first step to us.



$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m + n = 0, 0 \le -2 + n) \\ \{1\} & \text{And}(m - 1 = 0, n - 1 = 0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2 + n, 0 \le m - n - 1) \\ \{1\} & \text{And}(n - 1 = 0, 0 \le m - 2) \\ \left\{\frac{1}{3}m + \frac{1}{2}mn + \frac{1}{2}nm^2 - \frac{1}{3}m^3\right\} & \text{And}(0 \le m - 2, 0 \le n - 3, 0 \le -m + n - 1) \\ \{4 + n\} & \text{And}(m - 1 = 0, 0 \le -2 + n) \end{cases}$$

- Problems involving parametric polyhedra e.g. counting the integer points of $1 \le i \le n, 1 \le j \le m, j \le i, 1 \le k \le j$ tends to split more than necessary.
- Clearly, some cases are special cases of other cases.
- To combine two cases, the disjunction of their systems of constraints should define a polyhedral set, which must be the convex hull of the union of the zero sets of these systems of constraints.
- Therefore, computing the convex hull of finitely many polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$ is a first step to us.
- Our ultimate goal is to decide whether or not the union of finitely many $\mathbb{Z}$-polyhedral sets is a $\mathbb{Z}$-polyhedra set, extending Verdoolaege's 2D constructions in [21].

Computing convex hulls (I/II)

- Convex hulls of objects, that are not necessarily points, are used in the fields of autonomous vehicles and LiDAR [5, 8, 10], image processing [12, 13], robotics and motion planning [11, 19, 20], and geographic information systems [1], among others.

Computing convex hulls (I/II)

- Convex hulls of objects, that are not necessarily points, are used in the fields of autonomous vehicles and LiDAR [5, 8, 10], image processing [12, 13], robotics and motion planning [11, 19, 20], and geographic information systems [1], among others.
- In computational geometry:
  - Various algorithms have been proposed to compute the convex hull ConvexHull(V) of a finite set V of points.

Computing convex hulls (I/II)

- Convex hulls of objects, that are not necessarily points, are used in the fields of autonomous vehicles and LiDAR [5, 8, 10], image processing [12, 13], robotics and motion planning [11, 19, 20], and geographic information systems [1], among others.
- In computational geometry:
  - Various algorithms have been proposed to compute the convex hull $\mathsf{ConvexHull}(V)$ of a finite set V of points.
  - In 2D, with $n = |V|$ and $h = \mathsf{size}(V)$, the algorithm of Kirkpatrick and Seidel [17], as well as that of Chan [6] run in time $O(n \log(h))$, which is optimal.

Computing convex hulls (I/II)

- Convex hulls of objects, that are not necessarily points, are used in the fields of autonomous vehicles and LiDAR [5, 8, 10], image processing [12, 13], robotics and motion planning [11, 19, 20], and geographic information systems [1], among others.
- In computational geometry:
  - Various algorithms have been proposed to compute the convex hull $\mathsf{ConvexHull}(V)$ of a finite set V of points.
  - In 2D, with $n := |V|$ and $h := \mathsf{size}(V)$, the algorithm of Kirkpatrick and Seidel [17], as well as that of Chan [6] run in time $\mathcal{O}(n \log(h))$, which is optimal.
- But if k input polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$ are given in H-representation rather than V-representation, reducing to the latter makes no sense since $|V| \in \mathcal{O}(m^{\lfloor d/2 \rfloor})$ where $m := |H|$, see [18].

Computing convex hulls (II/II)

- For k polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$
  - given in H-representation,
  - bounded,
  - full-dimensional,
  - in general position,

Computing convex hulls (II/II)

- For k polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$
  - given in H-representation,
  - bounded,
  - full-dimensional,
  - in general position,
- Fukuda, Liebling and Lütolf proposes in [9] an algorithm (ECH)
  - computing the H-representation of $\mathsf{ConvexHull}(P_1 \cup \cdots \cup P_k)$, and
  - running in polynomial time in the size of the input and output.

Computing convex hulls (II/II)

- For k polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$
  - given in H-representation,
  - bounded,
  - full-dimensional,
  - in general position,
- Fukuda, Liebling and Lütolf proposes in [9] an algorithm (ECH)
  - computing the H-representation of $\mathsf{ConvexHull}(P_1 \cup \cdots \cup P_k)$, and
  - running in polynomial time in the size of the input and output.
- They rely on two core ideas:
  - the wrapping procedure of Chand and Kapur [7], and
  - the reverse search enumeration developed by Avis and Fukuda [2–4].

Our contribution and work in progress

Recall that the Authors of [9] assume that input polyhedra are *1.* full-dimensional, *2.* bounded, and *3.* in general position.

Our contribution and work in progress

Recall that the Authors of [9] assume that input polyhedra are *1.* full-dimensional, *2.* bounded, and *3.* in general position.

In our CASC 2026 paper [14]

- we drop the general position assumption,
- we relax the full-dimensionalily assumption,
- using H- and V-representation, we drop boundedness.

Our contribution and work in progress

Recall that the Authors of [9] assume that input polyhedra are *1.* full-dimensional, *2.* bounded, and *3.* in general position.

In our CASC 2026 paper [14]

- we drop the general position assumption,
- we relax the full-dimensionalily assumption,
- using H- and V-representation, we drop boundedness.

In this SYNASC 2026 paper,

- we no longer use the V-representation,
- we treat bounded and unbounded polyhedra uniformly.

Our contribution and work in progress

Recall that the Authors of [9] assume that input polyhedra are *1.* full-dimensional, *2.* bounded, and *3.* in general position.

In our CASC 2026 paper [14]

- we drop the general position assumption,
- we relax the full-dimensionalily assumption,
- using H- and V-representation, we drop boundedness.

In this SYNASC 2026 paper,

- we no longer use the V-representation,
- we treat bounded and unbounded polyhedra uniformly.

In what follows:

1. we start by illustrating a core procedure: the wrapping,
2. we explain the top-level procedure for convex hull,
3. we explain the wrapping procedure,
4. all our algorithms require LP in exact arithmetic (GMP),
5. experimentation is work in progress.

Plan

Recap on linear programming

For $P := \{x \in \mathbb{R}^d \mid Ax \leq b\}$, consider the linear program Π with $x \mapsto c^t x$ as objective function and P as feasible region.

Recap on linear programming

For $P := \{x \in \mathbb{R}^d \mid Ax \leq b\}$, consider the linear program Π with $x \mapsto c^t x$ as objective function and P as feasible region.

- Π admits an optimal solution if and only if $P \neq \emptyset$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.

Recap on linear programming

For $P := \{x \in \mathbb{R}^d \mid Ax \leq b\}$, consider the linear program Π with $x \mapsto c^t x$ as objective function and P as feasible region.

- Π admits an optimal solution if and only if $P \neq \emptyset$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.
- We denote by $\mathsf{LP}(d, h)$ an upper bound for the number of bit operations required for solving a linear program with total bit size h and with d variables. For instance, in the case of Karmarkar's algorithm [16], we have $\mathsf{LP}(d, h) \in O(d^{3.5} h^2 \cdot \log h \cdot \log\log h)$.

Recap on linear programming

For $P := \{x \in \mathbb{R}^d \mid Ax \leq b\}$, consider the linear program Π with $x \mapsto c^t x$ as objective function and P as feasible region.

- Π admits an optimal solution if and only if $P \neq \emptyset$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.
- We denote by $\mathsf{LP}(d, h)$ an upper bound for the number of bit operations required for solving a linear program with total bit size h and with d variables. For instance, in the case of Karmarkar's algorithm [16], we have $\mathsf{LP}(d, h) \in O(d^{3.5} h^2 \cdot \log h \cdot \log\log h)$.
- As a result, if h is the bit size of Π, one can decide
  - whether P is empty (using $c = 0$) in time $\mathsf{LP}(d, h)$, and
  - in time $2\mathsf{LP}(d, h)$ whether $c^t x = 0$ supports P (using $x \mapsto c^t x$ and then $x \mapsto -c^t x$, as objective functions).

Valid facets and wrapping

- We consider k polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their H-representation and want to compute $P_0 := \mathsf{ConvexHull}(P_1, \cdots, P_k)$.

Valid facets and wrapping

- We consider k polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their H-representation and want to compute $P_0 := \mathsf{ConvexHull}(P_1, \cdots, P_k)$.
- Assume one of P_i is full-dimensional. (A pre-processing step reduces to that situation.) Hence, P_0 is full-dimensional.

Valid facets and wrapping

- We consider k polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their H-representation and want to compute $P_0 := \mathsf{ConvexHull}(P_1, \cdots, P_k)$.
- Assume one of P_i is full-dimensional. (A pre-processing step reduces to that situation.) Hence, P_0 is full-dimensional.
- A facet F of any full-dimensional P_i is valid for P_0 if all $P_j, 1 \leq j \leq k$ are on the same side of F.

Valid facets and wrapping

- We consider k polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their H-representation and want to compute $P_0 := \mathsf{ConvexHull}(P_1, \cdots, P_k)$.
- Assume one of P_i is full-dimensional. (A pre-processing step reduces to that situation.) Hence, P_0 is full-dimensional.
- A facet F of any full-dimensional P_i is valid for P_0 if all $P_j, 1 \leq j \leq k$ are on the same side of F.
- Recall that any ridge (that is, a facet of a facet) of a full-dimensional polytope belongs to exactly two facets.

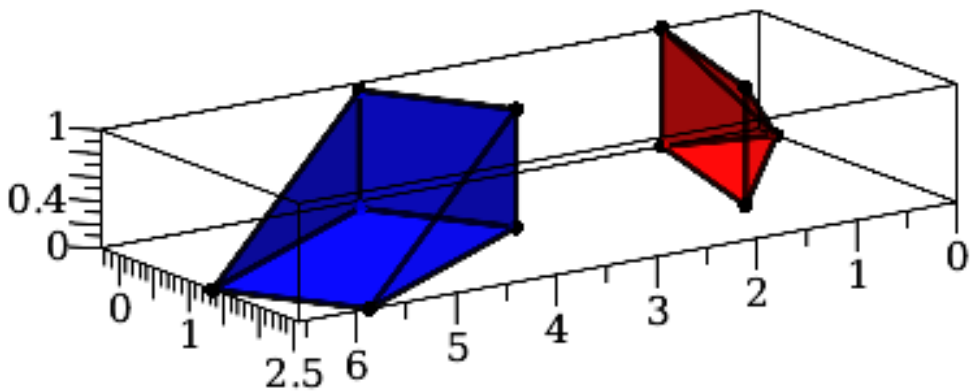
Valid facets and wrapping

- We consider k polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their H-representation and want to compute $P_0 := \mathsf{ConvexHull}(P_1, \cdots, P_k)$.
- Assume one of P_i is full-dimensional. (A pre-processing step reduces to that situation.) Hence, P_0 is full-dimensional.
- A facet F of any full-dimensional P_i is valid for P_0 if all $P_j, 1 \leq j \leq k$ are on the same side of F.
- Recall that any ridge (that is, a facet of a facet) of a full-dimensional polytope belongs to exactly two facets.
- What is wrapping? If a ridge R belongs to exactly one valid facet F, one can rotate its other facet F' along R into a supporting hyperpplane of a facet of P_0.

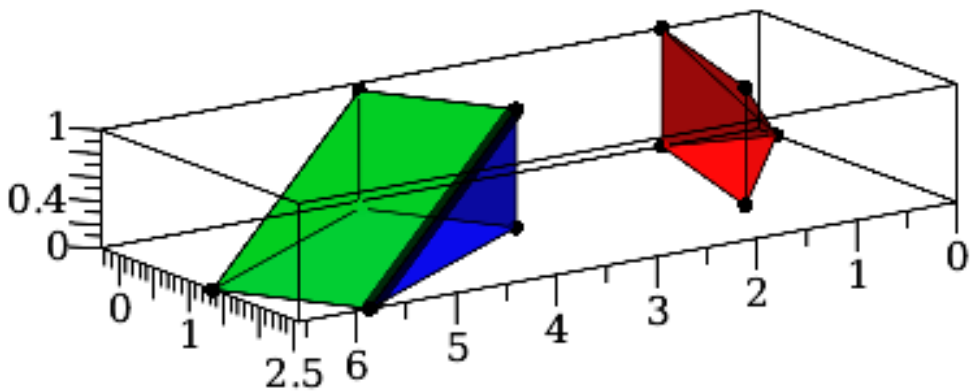
Valid facets and wrapping

- We consider k polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their H-representation and want to compute $P_0 := \mathsf{ConvexHull}(P_1, \cdots, P_k)$.
- Assume one of P_i is full-dimensional. (A pre-processing step reduces to that situation.) Hence, P_0 is full-dimensional.
- A facet F of any full-dimensional P_i is valid for P_0 if all $P_j, 1 \leq j \leq k$ are on the same side of F.
- Recall that any ridge (that is, a facet of a facet) of a full-dimensional polytope belongs to exactly two facets.
- What is wrapping? If a ridge R belongs to exactly one valid facet F, one can rotate its other facet F' along R into a supporting hyperpplane of a facet of P_0.
- How to wrap? The rotation angle can be obtained by solving an LP problem.

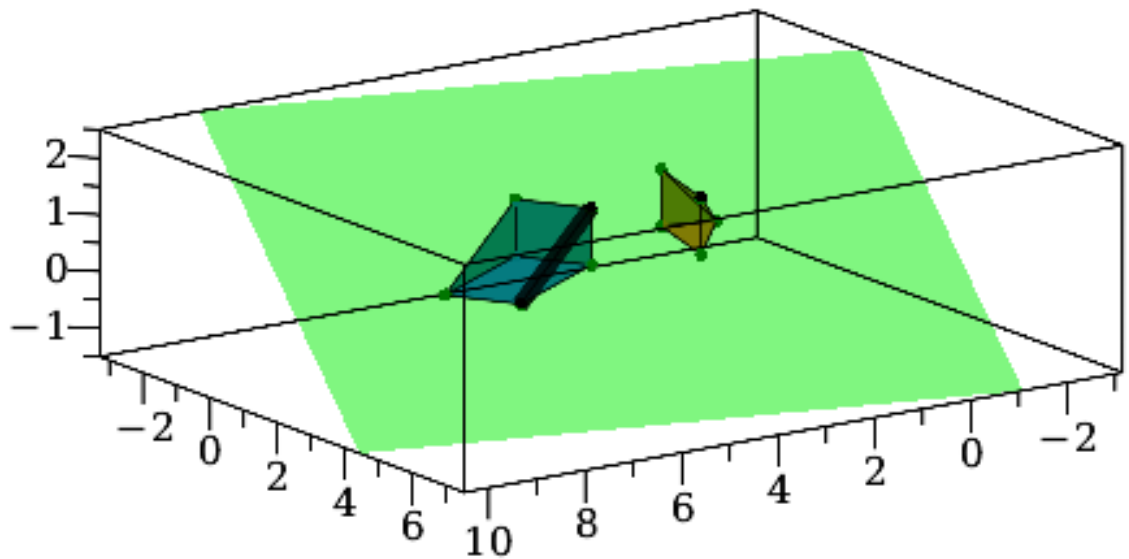
The wrapping procedure: example (I/V)



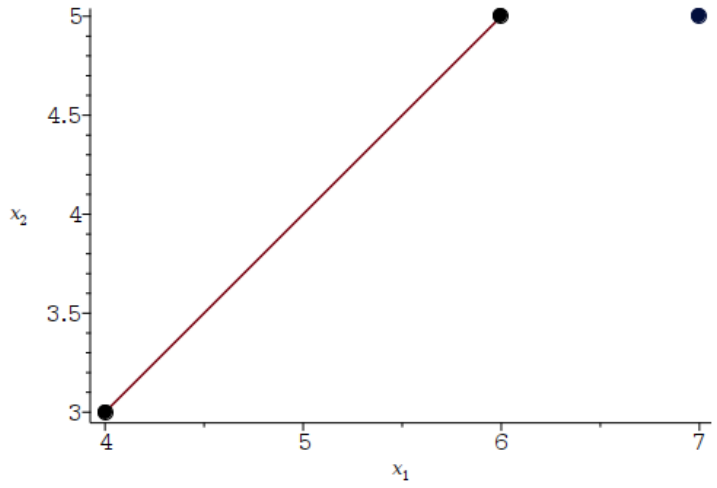
The wrapping procedure: example (II/V)

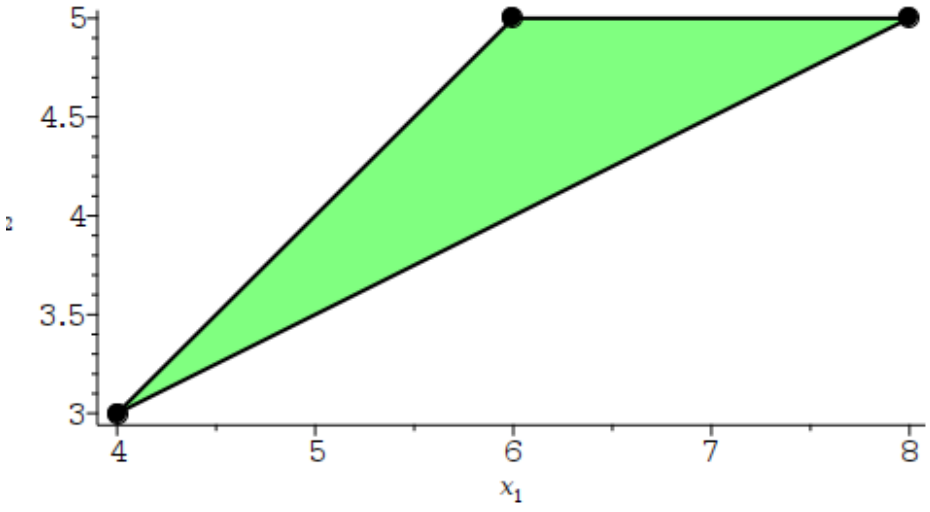


The wrapping procedure: example (III/V)

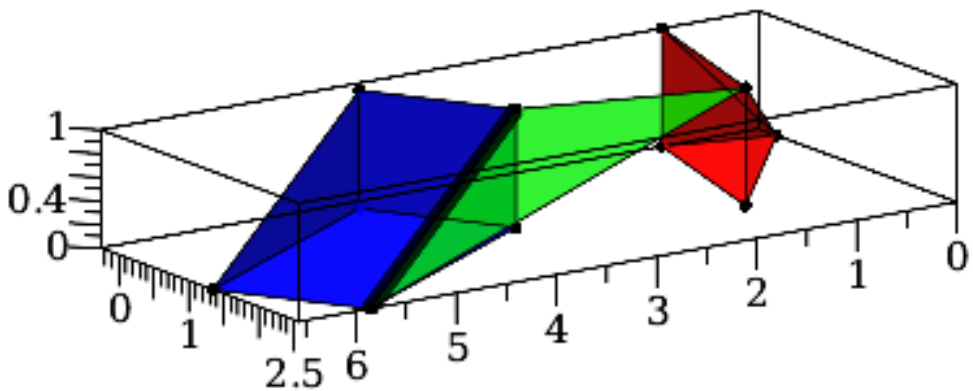


The wrapping procedure: example (IV/V)





The wrapping procedure: example (V/V)



Key features of our wrapping procedure

- On the contrary of the wrapping procedure of [9, 14]
  1. no projection to 2D is required,
  2. a rotation angle rather than a point is computed.

Key features of our wrapping procedure

- On the contrary of the wrapping procedure of [9, 14]
  1. no projection to 2D is required,
  2. a rotation angle rather than a point is computed.
- Once this rotation angle is known, the corresponding facet of P_0 is deduced by a recursive call to the convex hull procedure.

Key features of our wrapping procedure

- On the contrary of the wrapping procedure of [9, 14]
  1. no projection to 2D is required,
  2. a rotation angle rather than a point is computed.
- Once this rotation angle is known, the corresponding facet of P_0 is deduced by a recursive call to the convex hull procedure.
- This avoids the simplicial decomposition of [14] and relaxes the limitations of [9].

Plan

Specifications of our convex hull procedure

We make three assumptions:

1. P_0 is full-dimensional,
2. at least one of the input polyhedra P_i has a face F of co-dimension 1 (w.r.t the ambient space), and
3. that face F is contained in a facet of P_0.

Specifications of our convex hull procedure

We make three assumptions:

1. P_0 is full-dimensional,
2. at least one of the input polyhedra P_i has a face F of co-dimension 1 (w.r.t the ambient space), and
3. that face F is contained in a facet of P_0.

Then, we denote by ExtendedConvexHull a function so that the call ExtendedConvexHull$(P_1, \cdots, P_k)$ returns the H-representation of the convex hull P_0 of $P_1, \cdots, P_k$.

Specifications of our convex hull procedure

We make three assumptions:

1. P_0 is full-dimensional,
2. at least one of the input polyhedra P_i has a face F of co-dimension 1 (w.r.t the ambient space), and
3. that face F is contained in a facet of P_0.

Then, we denote by ExtendedConvexHull a function so that the call ExtendedConvexHull$(P_1, \cdots, P_k)$ returns the H-representation of the convex hull P_0 of $P_1, \cdots, P_k$. Preprocessing steps actually relax the assumptions.

The homogenization cone of P_0

- polyhedral sets P_i in H-rep: $A_i\vec{x_i} \leq \vec{b_i}$, $1 \leq i \leq k$.
- Assume these H-reps are minimal [15]
- C_i homogen. cone of P_i: $A_i\vec{x_i} \leq \vec{b_i}\lambda_i, \lambda_i \geq 0$, $1 \leq i \leq k$.
- C_0 homogen. cone of $P_0 = \mathsf{ConvexHull}(P_1, \ldots P_k)$.

The homogenization cone of P_0

- polyhedral sets P_i in H-rep: $A_i\vec{x_i} \leq \vec{b_i}$, $1 \leq i \leq k$.
- Assume these H-reps are minimal [15]
- C_i homogen. cone of P_i: $A_i\vec{x_i} \leq \vec{b_i}\lambda_i, \lambda_i \geq 0$, $1 \leq i \leq k$.
- C_0 homogen. cone of $P_0 = \mathsf{ConvexHull}(P_1, \ldots P_k)$.

Lemma (1)

We have $C_0 = C_1 + \ldots + C_k$.

The homogenization cone of P_0

- polyhedral sets P_i in H-rep: $A_i\vec{x_i} \leq \vec{b_i}$, $1 \leq i \leq k$.
- Assume these H-reps are minimal [15]
- C_i homogen. cone of P_i: $A_i\vec{x_i} \leq \vec{b_i}\lambda_i, \lambda_i \geq 0$, $1 \leq i \leq k$.
- C_0 homogen. cone of $P_0 = \mathsf{ConvexHull}(P_1, \ldots P_k)$.

Lemma (1)

We have $C_0 = C_1 + \ldots + C_k$.

Therefore, C_0 is given by the vectors $(\vec{x}, \lambda)$ satisfying the following system of Minkowski sums of C_i for $i = 1, \ldots, k$:

$$C_0 = \begin{cases} A_i^{\leq}\vec{x_i} \leq \vec{b_i}^{\leq}\lambda_i, & 1 \leq i \leq k \\ A_i^{=}\vec{x_i} = \vec{b_i}^{=}\lambda_i, & 1 \leq i \leq k \\ 0 \leq \lambda_i, & 1 \leq i \leq k \\ (\vec{x}, \lambda) = \sum_{i=1}^{k}(\vec{x_i}, \lambda_i). \end{cases} \tag{1}$$

where $A_i^{\leq}, \vec{b_i}^{\leq}$ (resp. $A_i^{=}, \vec{b_i}^{=}$) come from the inequalities (resp. equations) of P_i's H-rep.

When valid facets give rise to facets of P_0

- Recall P_i in H-rep: $A_i\vec{x}_i \leq \vec{b}_i$, $1 \leq i \leq k$.
- Let m_i be the number of rows of A_i.

When valid facets give rise to facets of P_0

- Recall P_i in H-rep: $A_i\vec{x_i} \leq \vec{b_i}$, $1 \leq i \leq k$.
- Let m_i be the number of rows of A_i.

Lemma (2)

Fix $1 \leq i \leq k$. *Assume* P_i *is full-dimensional. Let* $1 \leq c \leq m_i$. *If the half-space* $A_{i,c}x \leq b_{i,c}$ *is valid for* $P_1, \ldots, P_k$, *then the facet of* P_i *corresponding to* $A_{i,c}x \leq b_{i,c}$ *is contained in a facet of* P_0.

When valid facets give rise to facets of P_0

- Recall P_i in H-rep: $A_i\vec{x_i} \leq \vec{b_i}$, $1 \leq i \leq k$.
- Let m_i be the number of rows of A_i.

Lemma (2)

Fix $1 \leq i \leq k$. *Assume* P_i *is full-dimensional. Let* $1 \leq c \leq m_i$. *If the half-space* $A_{i,c}x \leq b_{i,c}$ *is valid for* $P_1, \ldots, P_k$, *then the facet of* P_i *corresponding to* $A_{i,c}x \leq b_{i,c}$ *is contained in a facet of* P_0.

Lemma (3)

Assume, P_i *is of co-dimension* 1 *and let* $A_{i,1}x = b_{i,1}$ *be the equation of* $A_i\vec{x_i} \leq \vec{b_i}$. *If either* $A_{i,1}x \leq b_{i,1}$ *or* $-A_{i,1}x \leq -b_{i,1}$ *is valid for* $P_1, \ldots, P_k$, *then* P_i *is contained in a facet of* P_0.

Determining the initial ridges

- Assume P_i is full-dimensional
- Recall P_i's H-rep is $A_i\vec{x_i} \leq \vec{b_i}$, with m_i rows
- Let $1 \leq j < \ell \leq m_i$.

Determining the initial ridges

- Assume P_i is full-dimensional
- Recall P_i's H-rep is $A_i\vec{x_i} \leq \vec{b_i}$, with m_i rows
- Let $1 \leq j < \ell \leq m_i$.

Lemma (4)

Let ε be a new variable. Consider the linear program Π whose feasible region is given by

$$\begin{cases} A_{i,j}x &= b_{i,j} & (H_{i,j}) \\ A_{i,\ell}x &= b_{i,\ell} & (H_{i,\ell}) \\ A'_i x &\leq b'_i - (\varepsilon, \ldots, \varepsilon)^t, \end{cases}$$

with ε as objective function to be maximized.

Determining the initial ridges

- Assume P_i is full-dimensional
- Recall P_i's H-rep is $A_i\vec{x_i} \leq \vec{b_i}$, with m_i rows
- Let $1 \leq j < \ell \leq m_i$.

Lemma (4)

Let ε be a new variable. Consider the linear program Π whose feasible region is given by

$$\begin{cases} A_{i,j}x &= b_{i,j} & (H_{i,j}) \\ A_{i,\ell}x &= b_{i,\ell} & (H_{i,\ell}) \\ A'_i x &\leq b'_i - (\varepsilon, \ldots, \varepsilon)^t, \end{cases}$$

with ε as objective function to be maximized.
Then, $H_{i,j} \cap H_{i,\ell} \cap P_i$ is a ridge of P_i if and only if Π's feasible region is not empty and the maximum value of Π's objective function is strictly positive.

High-level of view of our convex hull algorithm

- The algorithm manages two data-structures:
  1. a set $\mathcal{F}$, under construction, of half-spaces S valid for $P_1, \ldots, P_k$ so that a facet of P_0 has S as supporting hyperplane; $\mathcal{F}$ is initialized with Lemmas (2) and (3),

High-level of view of our convex hull algorithm

- The algorithm manages two data-structures:
  1. a set $\mathcal{F}$, under construction, of half-spaces S valid for $P_1, \ldots, P_k$ so that a facet of P_0 has S as supporting hyperplane; $\mathcal{F}$ is initialized with Lemmas (2) and (3),
  2. a set $\mathcal{R}$ of ridge objects R so that $R.\text{facet}_1$ is valid for $P_1, \ldots, P_k$ and $R.\text{facet}_2$ is not valid; $\mathcal{R}$ is initialized by considering the ridges of $P_1, \ldots, P_k$ with exactly one adjacent facet whose supporting hyperplane is in $\mathcal{F}$.
- while $\mathcal{R} \neq \emptyset$,
  1. we choose $R \in \mathcal{R}$ and apply the wrapping to it,

High-level of view of our convex hull algorithm

- The algorithm manages two data-structures:
  1. a set $\mathcal{F}$, under construction, of half-spaces S valid for $P_1, \ldots, P_k$ so that a facet of P_0 has S as supporting hyperplane; $\mathcal{F}$ is initialized with Lemmas (2) and (3),
  2. a set $\mathcal{R}$ of ridge objects R so that $R.\text{facet}_1$ is valid for $P_1, \ldots, P_k$ and $R.\text{facet}_2$ is not valid; $\mathcal{R}$ is initialized by considering the ridges of $P_1, \ldots, P_k$ with exactly one adjacent facet whose supporting hyperplane is in $\mathcal{F}$.
- while $\mathcal{R} \neq \emptyset$,
  1. we choose $R \in \mathcal{R}$ and apply the wrapping to it,
  2. this produces a half-space T incident on R whose supporting hyperplane H is that of a facet of P_0,

High-level of view of our convex hull algorithm

- The algorithm manages two data-structures:
  1. a set $\mathcal{F}$, under construction, of half-spaces S valid for $P_1, \ldots, P_k$ so that a facet of P_0 has S as supporting hyperplane; $\mathcal{F}$ is initialized with Lemmas (2) and (3),
  2. a set $\mathcal{R}$ of ridge objects R so that $R.\text{facet}_1$ is valid for $P_1, \ldots, P_k$ and $R.\text{facet}_2$ is not valid; $\mathcal{R}$ is initialized by considering the ridges of $P_1, \ldots, P_k$ with exactly one adjacent facet whose supporting hyperplane is in $\mathcal{F}$.
- while $\mathcal{R} \neq \emptyset$,
  1. we choose $R \in \mathcal{R}$ and apply the wrapping to it,
  2. this produces a half-space T incident on R whose supporting hyperplane H is that of a facet of P_0,
  3. we deduce a facet F'' of P_0 (possibly a subset of a facet of P_0), see next slide,

High-level of view of our convex hull algorithm

- The algorithm manages two data-structures:
  1. a set $\mathcal{F}$, under construction, of half-spaces S valid for $P_1, \ldots, P_k$ so that a facet of P_0 has S as supporting hyperplane; $\mathcal{F}$ is initialized with Lemmas (2) and (3),
  2. a set $\mathcal{R}$ of ridge objects R so that $R.\text{facet}_1$ is valid for $P_1, \ldots, P_k$ and $R.\text{facet}_2$ is not valid; $\mathcal{R}$ is initialized by considering the ridges of $P_1, \ldots, P_k$ with exactly one adjacent facet whose supporting hyperplane is in $\mathcal{F}$.
- while $\mathcal{R} \neq \emptyset$,
  1. we choose $R \in \mathcal{R}$ and apply the wrapping to it,
  2. this produces a half-space T incident on R whose supporting hyperplane H is that of a facet of P_0,
  3. we deduce a facet F'' of P_0 (possibly a subset of a facet of P_0), see next slide,
  4. using T and F'' we update $\mathcal{F}$ and $\mathcal{R}$ so as to maintain their invariants.

How the wrapping procedure yields a new facet p_0

- Recall that applying the wrapping to a ridge R,
- We obtain a half-space T incident on R whose supporting hyperplane H is that of a facet of P_0.
- Define $P_i' := H \cap P_i$.

How the wrapping procedure yields a new facet p_0

- Recall that applying the wrapping to a ridge R,
- We obtain a half-space T incident on R whose supporting hyperplane H is that of a facet of P_0.
- Define $P_i' := H \cap P_i$.

Lemma (6)

The polyhedral sets $P_1', \ldots, P_k', R$ satisfy the input specifications of our convex hull algorithm.

How the wrapping procedure yields a new facet p_0

- Recall that applying the wrapping to a ridge R,
- We obtain a half-space T incident on R whose supporting hyperplane H is that of a facet of P_0.
- Define $P_i' := H \cap P_i$.

Lemma (6)

The polyhedral sets $P_1', \ldots, P_k', R$ satisfy the input specifications of our convex hull algorithm.

Lemma (5)

We have
$H \cap \mathsf{ConvexHull}(P_1, \ldots, P_k) = \mathsf{ConvexHull}(P_1', \ldots, P_k')$.

Therefore, this recursive call to the convex hull algorithm returns the desired facet F''.

Wrapping without Projection

- The procedure input consists of
  - H_1: $a_1^t x - b_2 \leq 0$ a half space containing $P_1, \ldots, P_k$,
  - H_2: $a_2^t x - b_2 \leq 0$ a half space not containing all of $P_1, \ldots, P_k$,
  - the system of Equations (1) corresponding to the homogen. cone C_0.

Wrapping without Projection

- The procedure input consists of
  - H_1: $a_1^t x - b_2 \leq 0$ a half space containing $P_1, \ldots, P_k$,
  - H_2: $a_2^t x - b_2 \leq 0$ a half space not containing all of $P_1, \ldots, P_k$,
  - the system of Equations (1) corresponding to the homogen. cone C_0.
- The desired output is a half-space H_3 containing $P_1, \ldots, P_k$ and such that the pair H_1, H_3 define the same ridge as the pair H_1, H_2.

Wrapping without Projection

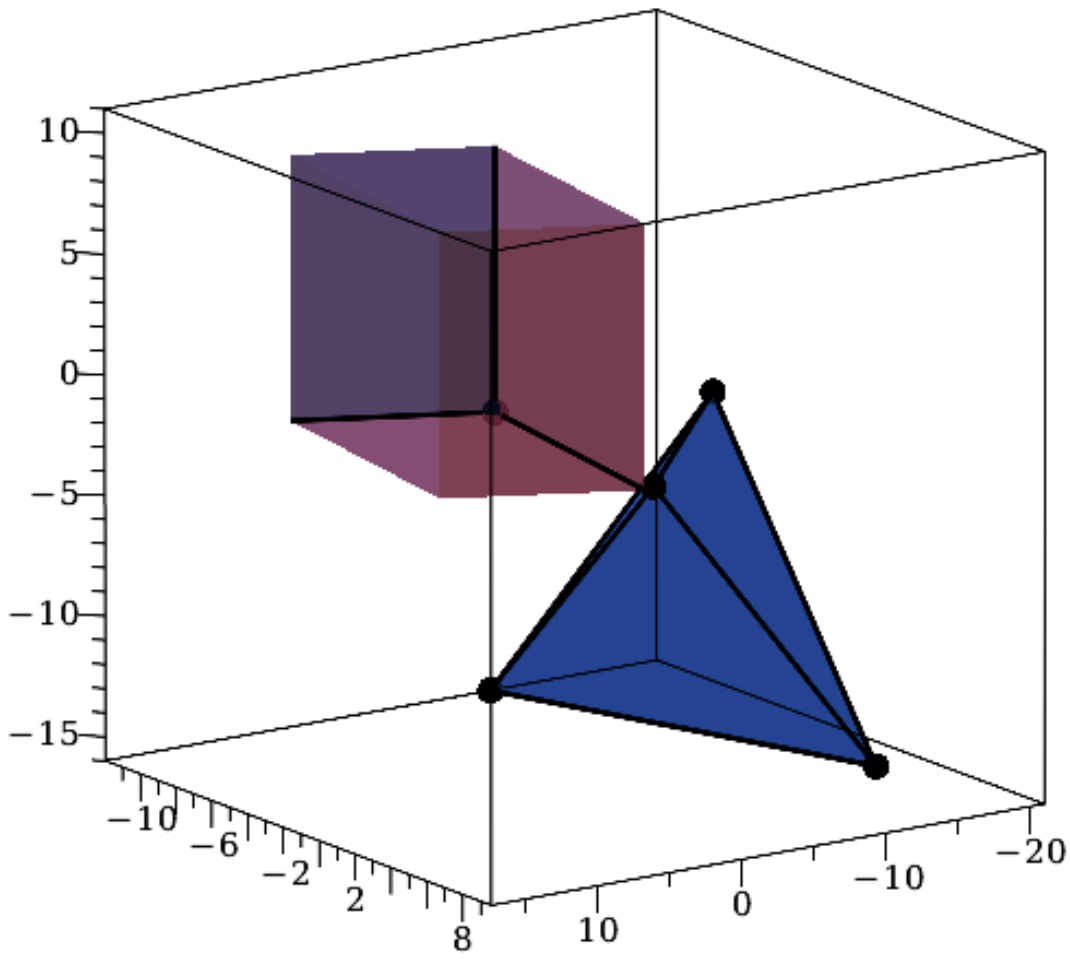
- The procedure input consists of
  - H_1: $a_1^t x - b_2 \leq 0$ a half space containing $P_1, \ldots, P_k$,
  - H_2: $a_2^t x - b_2 \leq 0$ a half space not containing all of $P_1, \ldots, P_k$,
  - the system of Equations (1) corresponding to the homogen. cone C_0.
- The desired output is a half-space H_3 containing $P_1, \ldots, P_k$ and such that the pair H_1, H_3 define the same ridge as the pair H_1, H_2.
- Hence, we search for H_3 in the form $H_3 = H_1 + kH_2$, with k real.

Wrapping without Projection

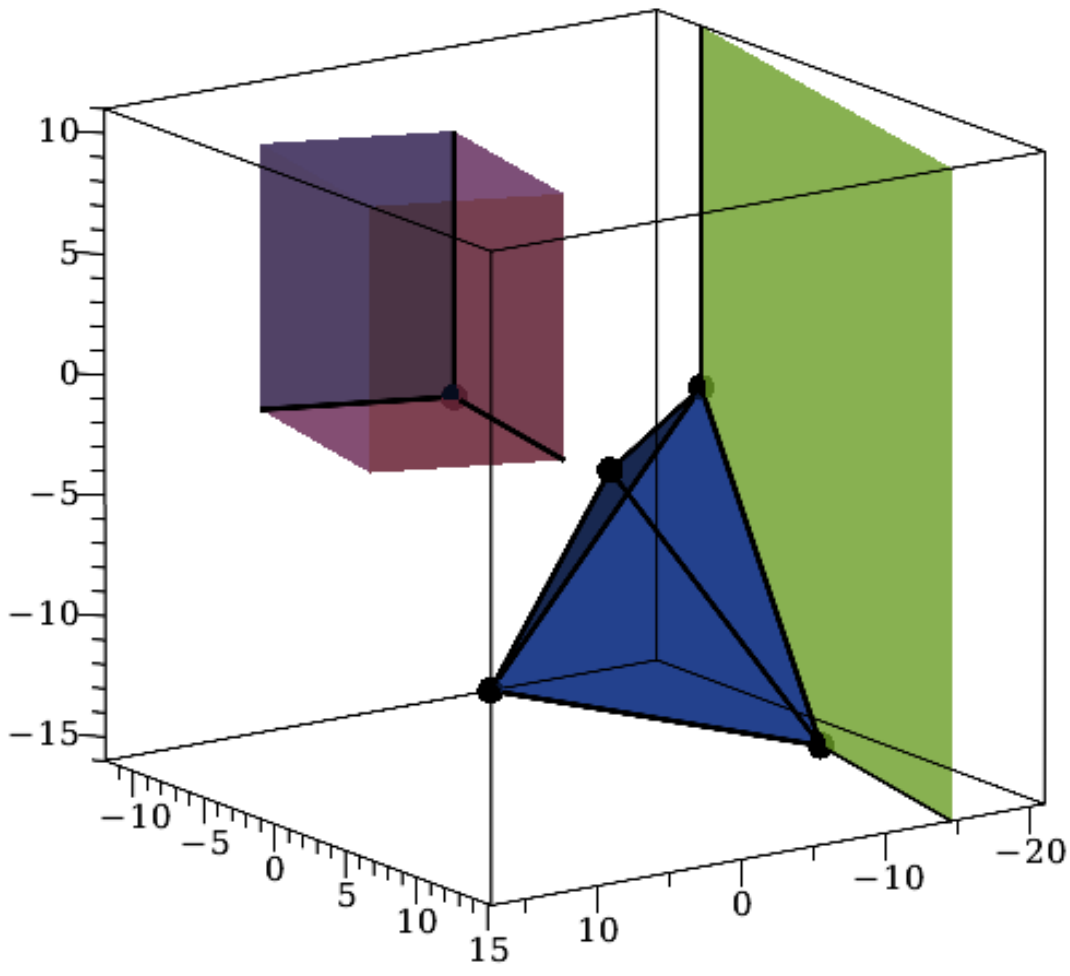
- The procedure input consists of
  - H_1: $a_1^t x - b_2 \leq 0$ a half space containing $P_1, \ldots, P_k$,
  - H_2: $a_2^t x - b_2 \leq 0$ a half space not containing all of $P_1, \ldots, P_k$,
  - the system of Equations (1) corresponding to the homogen. cone C_0.
- The desired output is a half-space H_3 containing $P_1, \ldots, P_k$ and such that the pair H_1, H_3 define the same ridge as the pair H_1, H_2.
- Hence, we search for H_3 in the form $H_3 = H_1 + kH_2$, with k real.
- Applying the Charnes-Cooper transformation, the desired k is obtained as

$$\begin{aligned} &\min -(a_1^t x - b_1 \lambda) \\ &\text{Subject to: Conditions 1} \\ &a_2^t x - b_2 \lambda = 1. \end{aligned} \tag{2}$$

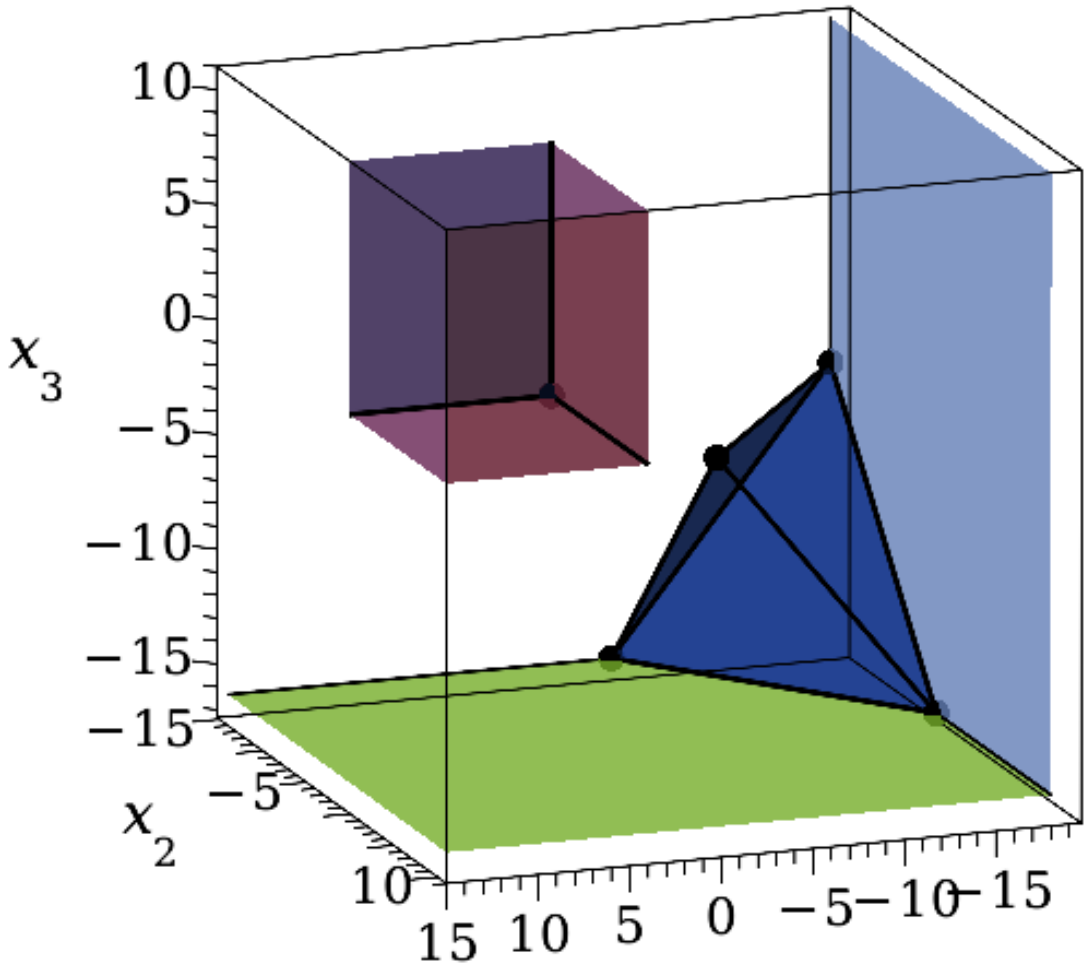
The extended convex hull (I/V)



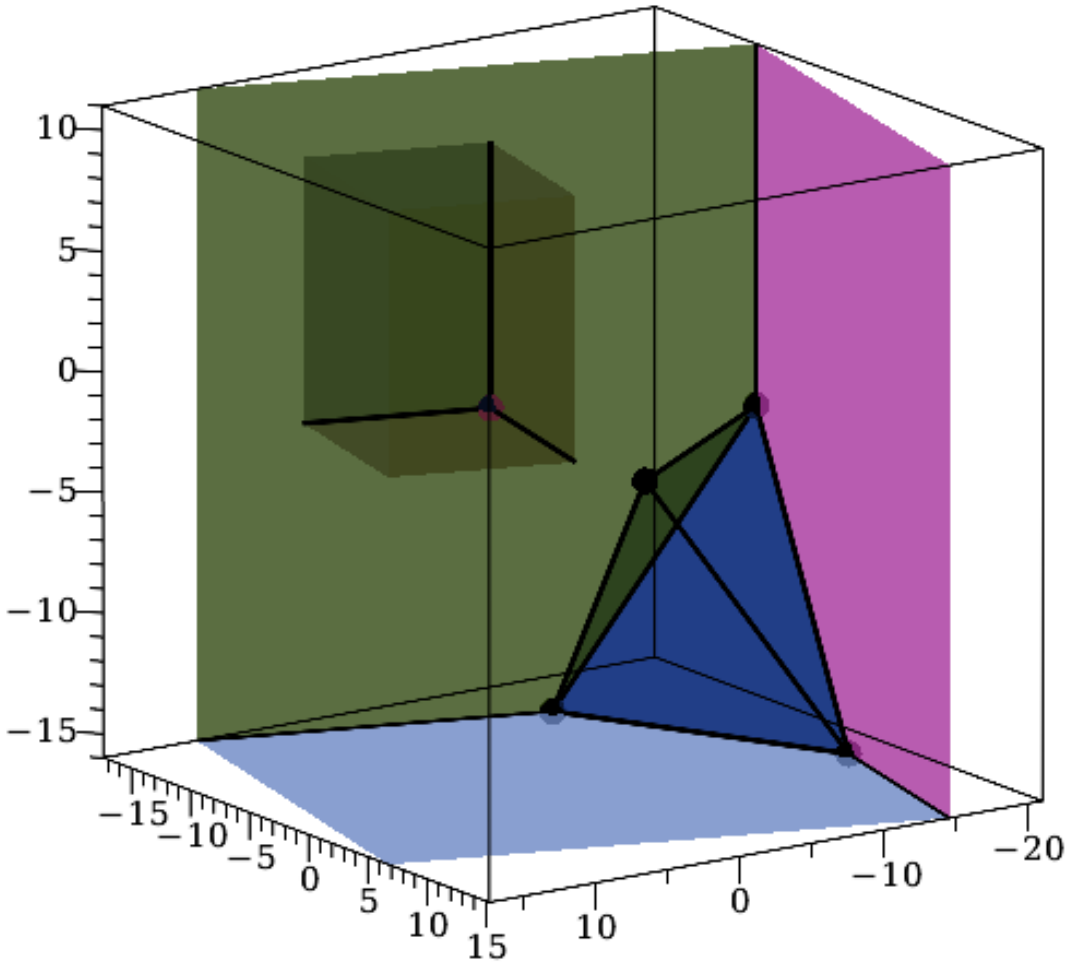
The extended convex hull (II/V)



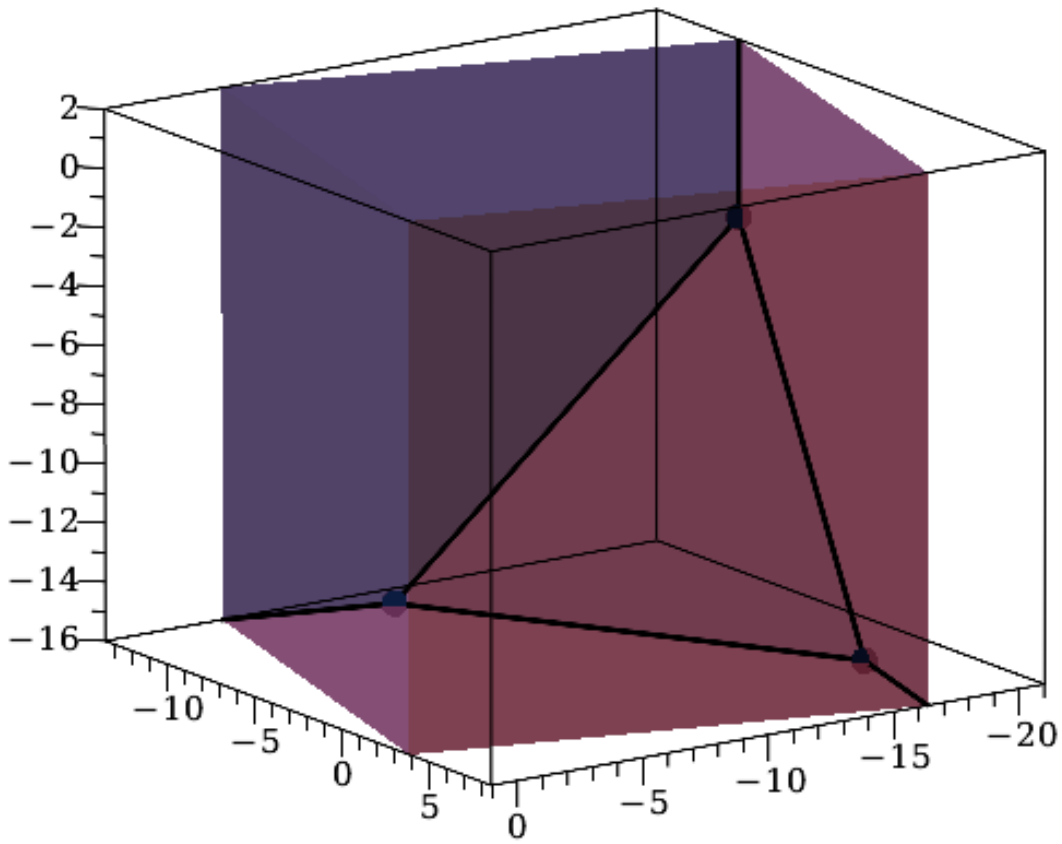
The extended convex hull (III/V)



The extended convex hull (IV/V)



The extended convex hull (V/V)



References I

[1] J. Antony, M. K. Mukundan, M. Thomas, and R. Muthuganapathy. "Convex Hull Computation in a Grid Space: A GPU Accelerated Parallel Filtering Approach". In: *Pacific Graphics Conference Papers and Posters*. Ed. by R. Chen, T. Ritschel, and E. Whiting. The Eurographics Association, 2024. ISBN: 978-3-03868-250-9.

[2] D. Avis and K. Fukuda. "A basis enumeration algorithm for linear systems with geometric applications". In: *Applied Mathematics Letters* 4.5 (1991), pp. 39–42.

[3] D. Avis and K. Fukuda. "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra". In: *Proceedings of the seventh annual symposium on Computational geometry*. 1991, pp. 98–104.

References II

[4] D. Avis and K. Fukuda. "Reverse search for enumeration". In: *Discrete applied mathematics* 65.1-3 (1996), pp. 21–46.

[5] E. Bernardi, S. Masi, P. Xu, and P. Bonnifait. "High Integrity Lane-level Occupancy Estimation of Road Obstacles Through LiDAR and HD Map Data Fusion". In: *IEEE Intelligent Vehicles Symposium*. IEEE, 2020, pp. 1873–1878.

[6] T. M. Chan. "Optimal output-sensitive convex hull algorithms in two and three dimensions". In: *Discrete & computational geometry* 16.4 (1996), pp. 361–368.

[7] D. R. Chand and S. S. Kapur. "An algorithm for convex polytopes". In: *Journal of the ACM (JACM)* 17.1 (1970), pp. 78–86.

References III

[8] N. Ding. "An Efficient Convex Hull-Based Vehicle Pose Estimation Method for 3D LiDAR". In: *CoRR* abs/2302.01034 (2023). arXiv: 2302.01034.

[9] K. Fukuda, T. M. Liebling, and C. Lütolf. "Extended convex hull". In: *Computational Geometry* 20.1-2 (2001), pp. 13–23.

[10] M. Gouda, Z. Pawliuk, J. Mirza, and K. El-Basyouny. "Using convex hulls with octree/voxel representations of point clouds to assess road and roadside geometric design for automated vehicles". In: *Automation in Construction* 154 (2023), p. 104967. ISSN: 0926-5805.

[11] M. Greer. *Memory Optimization for Convex Hull Support Point Queries*. 2025. arXiv: 2509.03753 [cs.GR]. URL: https://arxiv.org/abs/2509.03753.

References IV

[12] Z. Guo, C. Liu, X. Zhang, J. Jiao, X. Ji, and Q. Ye. "Beyond Bounding-Box: Convex-Hull Feature Adaptation for Oriented and Densely Packed Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 8792–8801.

[13] M. A. Jayaram and H. Fleyeh. "Convex Hulls in Image Processing: A Scoping Review". In: *American Journal of Intelligent Systems* 6.2 (2016), pp. 48–58.

[14] R. Jing, Y. Lei, M. Moreno Maza, and C. Mukherjee. "On the Computation of Extended Convex Hulls". In: *Computer Algebra in Scientific Computing*. Cham: Springer Nature Switzerland, 2027, pp. 189–209. ISBN: 978-3-032-34586-8.

References V

[15] R. Jing, M. Moreno Maza, C. Mukherjee, Y. Xie, and C. Yuan. "Efficient detection of redundancies in systems of linear inequalities". In: *J. Symb. Comput.* 135 (2026), p. 102527.

[16] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. STOC '84. New York, NY, USA: ACM, 1984, pp. 302–311. ISBN: 0-89791-133-4. DOI: 10.1145/800057.808695. URL: http://doi.acm.org/10.1145/800057.808695.

[17] D. G. Kirkpatrick and R. Seidel. "The ultimate planar convex hull algorithm?" In: *SIAM journal on computing* 15.1 (1986), pp. 287–299.

References VI

[18] P. McMullen. "On the upper-bound conjecture for convex polytopes". In: *Journal of Combinatorial Theory, Series B* 10.3 (1971), pp. 187–200.

[19] S. Meeran and A. Share. "Optimum path planning using convex hull and local search heuristic algorithms". In: *Mechatronics* 7.8 (1997), pp. 737–756. ISSN: 0957-4158.

[20] J. Schulman, Y. Duan, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. "Motion planning with sequential convex optimization and convex collision checking". In: *Int. J. Robotics Res.* 33.9 (2014), pp. 1251–1270.

[21] S. Verdoolaege. "Integer set coalescing". In: *International Workshop on Polyhedral Compilation Techniques, Date: 2015/01/19-2015/01/19, Location: Amsterdam, The Netherlands*. 2015.