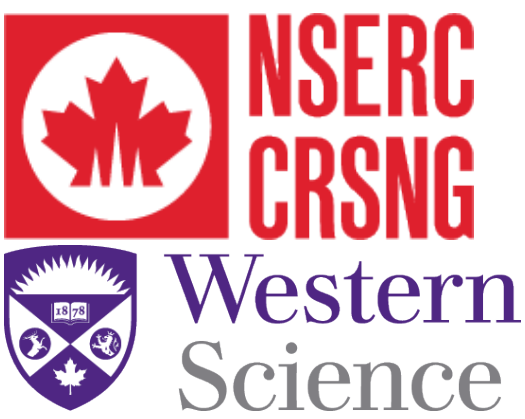
# On the Computation of Convex Hulls

CASC 2026 · Aug. 31 - Sept. 4, Bath, UK

Marc Moreno Maza
Ontario Research Center for Computer Algebra (ORCCA), UWO, London, Ontario.
Joint work with:
Rui-Juan Jing, Yuzhuo Lei and Chirantan Mukherjee

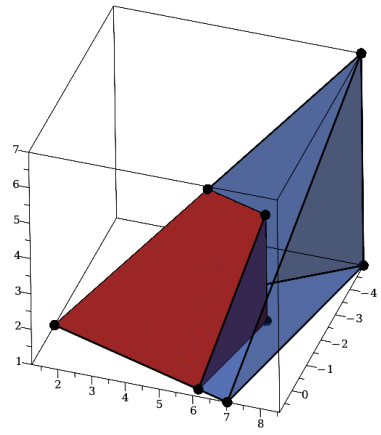
Sept 1, 2026




# Plan

1. Overview
2. The wrapping procedure
3. Computing ECH in the bounded case
4. Reduction to the bounded case

# A motivating example



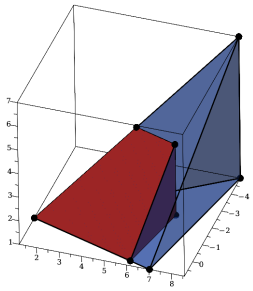
$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$

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

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

# A motivating example

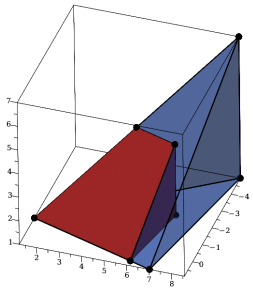
$np := NumberOfIntegerPoints(parametric\_polytope\_2, [i, j, k], [m, n], output = piecewise);$



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

# A motivating example

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

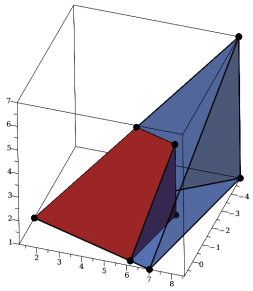


$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m+n=0,\ 0 \le -2+n) \\ \{1\} & \text{And}(m-1=0,\ n-1=0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2+n,\ 0 \le m-n-1) \\ \{1\} & \text{And}(n-1=0,\ 0 \le m-2) \\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

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

# A motivating example

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

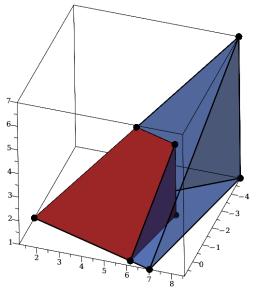


$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m+n=0,\ 0 \le -2+n) \\ \{1\} & \text{And}(m-1=0,\ n-1=0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2+n,\ 0 \le m-n-1) \\ \{1\} & \text{And}(n-1=0,\ 0 \le m-2) \\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

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

# A motivating example

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

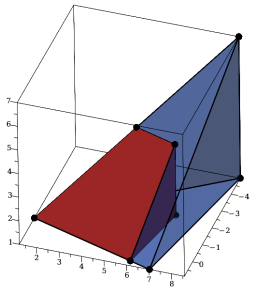


$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m+n=0,\ 0 \le -2+n) \\ \{1\} & \text{And}(m-1=0,\ n-1=0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2+n,\ 0 \le m-n-1) \\ \{1\} & \text{And}(n-1=0,\ 0 \le m-2) \\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

1. Problems involving **parametric polyhedra** e. g. counting the integer points of $1 \le i \le n, \quad 1 \le j \le m, \quad j \le i, \quad 1 \le k \le j$ tend to split more necessary.
2. Clearly, some cases are **special cases** of other cases.
3. 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.

# A motivating example

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

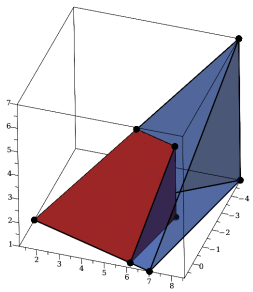


$$np := \begin{cases} \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(-m+n=0,\ 0 \le -2+n) \\ \{1\} & \text{And}(m-1=0,\ n-1=0) \\ \left\{\frac{1}{3}n + \frac{1}{2}n^2 + \frac{1}{6}n^3\right\} & \text{And}(0 \le -2+n,\ 0 \le m-n-1) \\ \{1\} & \text{And}(n-1=0,\ 0 \le m-2) \\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

1. Problems involving **parametric polyhedra** e. g. counting the integer points of $1 \le i \le n, \quad 1 \le j \le m, \quad j \le i, \quad 1 \le k \le j$ tend to split more necessary.
2. Clearly, some cases are **special cases** of other cases.
3. 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.
4. 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.

# A motivating example

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



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

1. Problems involving **parametric polyhedra** e. g. counting the integer points of $1 \le i \le n, \quad 1 \le j \le m, \quad j \le i, \quad 1 \le k \le j$ tend to split more necessary.
2. Clearly, some cases are **special cases** of other cases.
3. 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.
4. 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.
5. 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 [19].

# Computing convex hulls (1/2)

1. 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, 17, 18], and geographic information systems [1], among others.

# Computing convex hulls (1/2)

1. 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, 17, 18], and geographic information systems [1], among others.
2. In computational geometry:

# Computing convex hulls (1/2)

1. 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, 17, 18], and geographic information systems [1], among others.
2. In computational geometry:
   a. Various algorithms have been proposed to compute the convex hull ConvexHull($V$) of a **finite set $V$ of points**.

# Computing convex hulls (1/2)

1. 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, 17, 18], and geographic information systems [1], among others.
2. In computational geometry:
   a. Various algorithms have been proposed to compute the convex hull $\mathrm{ConvexHull}(V)$ of a **finite set $V$ of points**.
   b. In 2D, with $n := |V|$ and $h := \mathrm{size}(V)$, the algorithm of Kirkpatrick and Seidel [15], as well as that of Chan [6] run in time $\mathcal{O}(n \log(h))$, which is **optimal**.

# Computing convex hulls (1/2)

1. 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, 17, 18], and geographic information systems [1], among others.
2. In computational geometry:
   a. Various algorithms have been proposed to compute the convex hull $\mathrm{ConvexHull}(V)$ of a **finite set $V$ of points**.
   b. In 2D, with $n := |V|$ and $h := \mathrm{size}(V)$, the algorithm of Kirkpatrick and Seidel [15], as well as that of Chan [6] run in time $\mathcal{O}(n \log(h))$, which is **optimal**.
3. 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 [16].

# Computing convex hulls (2/2)

1. For $k$ polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$

# Computing convex hulls (2/2)

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

# Computing convex hulls (2/2)

1. For $k$ polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$
   a. given in H-representation,
   b. bounded,
   c. full-dimensional,
   d. in general position,
2. Fukuda, Liebling and Lütolf proposes in [9] an algorithm (ECH)

# Computing convex hulls (2/2)

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

# Computing convex hulls (2/2)

1. For $k$ polyhedral sets $P_1, \ldots, P_k$ of $\mathbb{R}^d$
   a. given in H-representation,
   b. bounded,
   c. full-dimensional,
   d. in general position,
2. Fukuda, Liebling and Lütolf proposes in [9] an algorithm (ECH)
   a. computing the H-representation of $\mathrm{ConvexHull}(P_1 \cup \cdots \cup P_k)$, and
   b. running in polynomial time in the size of the **input and output**.
3. They rely on two core ideas:

# Computing convex hulls (2/2)

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

# Our contributions and work in progress

1. We allow the ECH to work with polyhedra of **arbitrary dimension**, but still requiring boundedness

# Our contributions and work in progress

1. We allow the ECH to work with polyhedra of **arbitrary dimension**, but still requiring boundedness
2. We develop an algorithm computing ConvexHull$(P_1 \cup \cdots \cup P_k)$ **dropping the boundedness requirement**, but using both H-representation and V-representation

# Our contributions and work in progress

1. We allow the ECH to work with polyhedra of **arbitrary dimension**, but still requiring boundedness
2. We develop an algorithm computing ConvexHull$(P_1 \cup \cdots \cup P_k)$ **dropping the boundedness requirement**, but using both H-representation and V-representation
3. More recently (SYNASC 2026) we further improve that latter algorithm and totally **avoid the use of V-representation**

# Our contributions and work in progress

1. We allow the ECH to work with polyhedra of **arbitrary dimension**, but still requiring boundedness
2. We develop an algorithm computing ConvexHull$(P_1 \cup \cdots \cup P_k)$ **dropping the boundedness requirement**, but using both H-representation and V-representation
3. More recently (SYNASC 2026) we further improve that latter algorithm and totally **avoid the use of V-representation**
4. All our algorithms require linear programming (LP) in **exact arithmetic** (GMP)

# Our contributions and work in progress

1. We allow the ECH to work with polyhedra of **arbitrary dimension**, but still requiring boundedness
2. We develop an algorithm computing ConvexHull$(P_1 \cup \cdots \cup P_k)$ **dropping the boundedness requirement**, but using both H-representation and V-representation
3. More recently (SYNASC 2026) we further improve that latter algorithm and totally **avoid the use of V-representation**
4. All our algorithms require linear programming (LP) in **exact arithmetic** (GMP)
5. Experimentation is work in progress.

# Our contributions and work in progress

1. We allow the ECH to work with polyhedra of **arbitrary dimension**, but still requiring boundedness
2. We develop an algorithm computing ConvexHull$(P_1 \cup \cdots \cup P_k)$ **dropping the boundedness requirement**, but using both H-representation and V-representation
3. More recently (SYNASC 2026) we further improve that latter algorithm and totally **avoid the use of V-representation**
4. All our algorithms require linear programming (LP) in **exact arithmetic** (GMP)
5. Experimentation is work in progress.
6. In fact, our plan is to reducing everything in polyhedral geometry to LP.

# Plan

1. Overview
2. The wrapping procedure
3. Computing ECH in the bounded case
4. Reduction to the bounded case

# Recap on linear programming

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

1. $\Pi$ admits an optimal solution if and only if $P \neq \varnothing$ 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 $\Pi$ with $x \longmapsto c^t x$ as objective function and $P$ as feasible region.

1. $\Pi$ admits an optimal solution if and only if $P \neq \varnothing$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.
2. We denote by $\mathrm{LP}(d, h)$ 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 [14], we have $\mathrm{LP}(d, h) \in O(d^{3.5} h^2 \cdot \log h \cdot \log\log h)$.

# Recap on linear programming

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

1. $\Pi$ admits an optimal solution if and only if $P \neq \varnothing$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.
2. We denote by $\mathrm{LP}(d, h)$ 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 [14], we have $\mathrm{LP}(d, h) \in O(d^{3.5} h^2 \cdot \log h \cdot \log\log h)$.
3. As a result, if $h$ is the bit size of $\Pi$, one can decide

# Recap on linear programming

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

1. $\Pi$ admits an optimal solution if and only if $P \neq \varnothing$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.
2. We denote by $\mathrm{LP}(d, h)$ 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 [14], we have $\mathrm{LP}(d, h) \in O(d^{3.5} h^2 \cdot \log h \cdot \log\log h)$.
3. As a result, if $h$ is the bit size of $\Pi$, one can decide
   a. whether $P$ is empty (using $c = 0$) in time $\mathrm{LP}(d, h)$, and

# Recap on linear programming

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

1. $\Pi$ admits an optimal solution if and only if $P \neq \varnothing$ and $c \notin \{y \in \mathbb{R}^d \mid Ay \leq 0\}$.
2. We denote by $\mathrm{LP}(d, h)$ 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 [14], we have $\mathrm{LP}(d, h) \in O(d^{3.5} h^2 \cdot \log h \cdot \log\log h)$.
3. As a result, if $h$ is the bit size of $\Pi$, one can decide
   a. whether $P$ is empty (using $c = 0$) in time $\mathrm{LP}(d, h)$, and
   b. in time $2\mathrm{LP}(d, h)$ whether $c^t x = 0$ supports $P$ (using $x \longmapsto c^t x$ and then $x \longmapsto -c^t x$, as objective functions).

# Valid and non-valid facets

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

# Valid and non-valid facets

1. We consider $k$ polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their **H-representation** and want to compute $P_0 := \text{ConvexHull}(P_1, \cdots, P_k)$.
2. At least one of $P_1, \ldots, P_k$ is full-dimensional. Hence, $P_0$ is full-dimensional.

# Valid and non-valid facets

1. We consider $k$ polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their **H-representation** and want to compute $P_0 := \text{ConvexHull}(P_1, \cdots, P_k)$.
2. At least one of $P_1, \ldots, P_k$ is full-dimensional. Hence, $P_0$ is full-dimensional.
3. A facet $F$ of a full-dimensional $P_i$, for any $1 \leq i \leq k$ is **valid** for $P_0$ if this is a facet of $P_0$, that is, if all $P_j, 1 \leq j \leq k$ are on the same side of $F$.

# Valid and non-valid facets

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

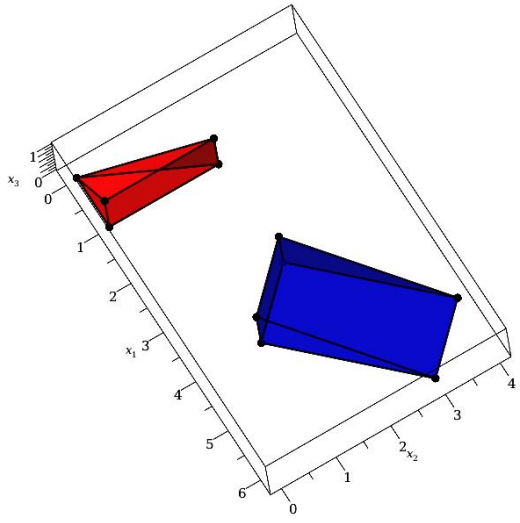
# Valid and non-valid facets

1. We consider $k$ polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their **H-representation** and want to compute $P_0 := \text{ConvexHull}(P_1, \cdots, P_k)$.
2. At least one of $P_1, \ldots, P_k$ is full-dimensional. Hence, $P_0$ is full-dimensional.
3. A facet $F$ of a full-dimensional $P_i$, for any $1 \leq i \leq k$ is **valid** for $P_0$ if this is a facet of $P_0$, that is, if all $P_j, 1 \leq j \leq k$ are on the same side of $F$.
4. Recall that any ridge (that is, a facet of a facet) of a full-dimensional polytope belongs to exactly two facets.
5. **Key observation 1:** 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 $P_0$.

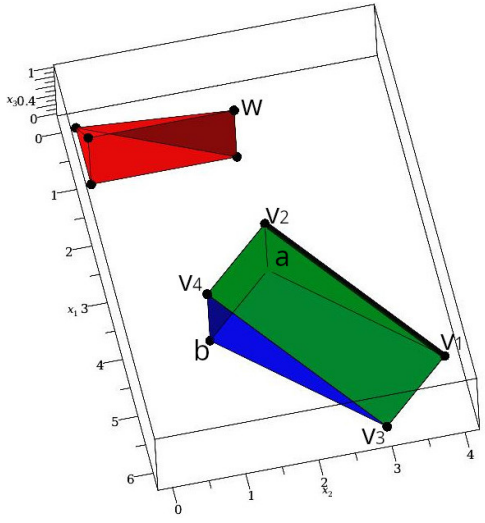
# Valid and non-valid facets

1. We consider $k$ polytopes $P_1, \ldots, P_k$ of $\mathbb{R}^d$ given by their **H-representation** and want to compute $P_0 := \text{ConvexHull}(P_1, \cdots, P_k)$.
2. At least one of $P_1, \ldots, P_k$ is full-dimensional. Hence, $P_0$ is full-dimensional.
3. A facet $F$ of a full-dimensional $P_i$, for any $1 \leq i \leq k$ is **valid** for $P_0$ if this is a facet of $P_0$, that is, if all $P_j, 1 \leq j \leq k$ are on the same side of $F$.
4. Recall that any ridge (that is, a facet of a facet) of a full-dimensional polytope belongs to exactly two facets.
5. **Key observation 1:** 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 $P_0$.
6. **Key observation 2:** the rotation angle can be obtained by solving an **LP** problem.

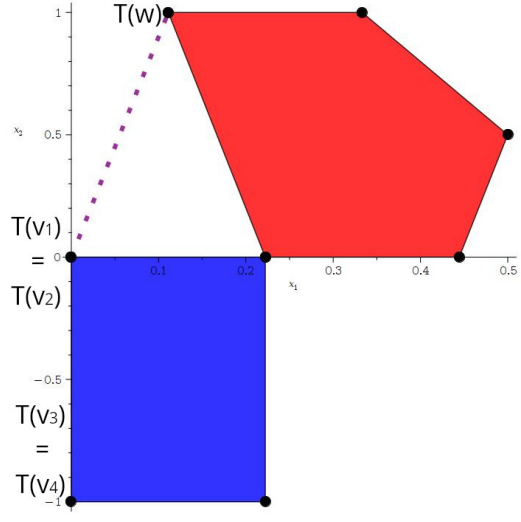
# The wrapping procedure: an example



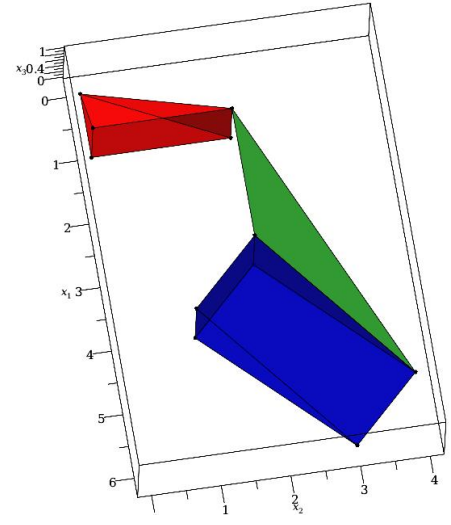
(a) Step 1




(b) Step 2




(c) Step 3



(d) Step 4

# The wrapping procedure: the formulas (1/2)

1. Let $F$ be a facet of $P_0$ and a ridge $R$ of $F$ for which the facet $F'$ adjacent to $F$ on $R$ is not known.

# The wrapping procedure: the formulas (1/2)

1. Let $F$ be a facet of $P_0$ and a ridge $R$ of $F$ for which the facet $F'$ adjacent to $F$ on $R$ is not known.
2. Wrapping$(R, F, P_1, \ldots, P_k)$ computes a vertex $w$ of $F'$ so that $w \notin R$, thus, so that ConvexHull$(R \cup \{w\})$ is a $(d-1)$-simplex of $\mathbb{R}^d$.

# The wrapping procedure: the formulas (1/2)

1. Let $F$ be a facet of $P_0$ and a ridge $R$ of $F$ for which the facet $F'$ adjacent to $F$ on $R$ is not known.
2. Wrapping$(R, F, P_1, \ldots, P_k)$ computes a vertex $w$ of $F'$ so that $w \notin R$, thus, so that ConvexHull$(R \cup \{w\})$ is a $(d-1)$-simplex of $\mathbb{R}^d$.
3. we assume that $d-1$ vertices $v_1, \ldots, v_{d-1}$ of $R$, and a vertex $v_d$ of $F$ so that $v_d \notin R$ are known.

# The wrapping procedure: the formulas (1/2)

1. Let $F$ be a facet of $P_0$ and a ridge $R$ of $F$ for which the facet $F'$ adjacent to $F$ on $R$ is not known.
2. Wrapping$(R, F, P_1, \ldots, P_k)$ computes a vertex $w$ of $F'$ so that $w \notin R$, thus, so that ConvexHull$(R \cup \{w\})$ is a $(d-1)$-simplex of $\mathbb{R}^d$.
3. we assume that $d-1$ vertices $v_1, \ldots, v_{d-1}$ of $R$, and a vertex $v_d$ of $F$ so that $v_d \notin R$ are known.
4. Let $n_1$ be the outer normal of $F$ and $r_1, \ldots, r_{d-2}$ a basis for the linear space spanned by $v_2 - v_1, \ldots, v_{d-1} - v_1$, satisfying $n_1 \perp r_j$ for $j = 1, \ldots, d-2$.

# The wrapping procedure: the formulas (1/2)

1. Let $F$ be a facet of $P_0$ and a ridge $R$ of $F$ for which the facet $F'$ adjacent to $F$ on $R$ is not known.
2. Wrapping$(R, F, P_1, \ldots, P_k)$ computes a vertex $w$ of $F'$ so that $w \notin R$, thus, so that ConvexHull$(R \cup \{w\})$ is a $(d-1)$-simplex of $\mathbb{R}^d$.
3. we assume that $d-1$ vertices $v_1, \ldots, v_{d-1}$ of $R$, and a vertex $v_d$ of $F$ so that $v_d \notin R$ are known.
4. Let $n_1$ be the outer normal of $F$ and $r_1, \ldots, r_{d-2}$ a basis for the linear space spanned by $v_2 - v_1, \ldots, v_{d-1} - v_1$, satisfying $n_1 \perp r_j$ for $j = 1, \ldots, d-2$.
5. Let $n_2$ be another vector satisfying $n_2 \perp n_1$, $n_2 \perp r_j$ for all $j$, and $n_2^t(v_d - v_1) < 0$. The vectors $r_1, \ldots, r_{d-2}, n_1, n_2$ a basis of $\mathbb{R}^d$, assembled into the $d \times d$ matrix $B := [r_1 \cdots r_{d-2}\ n_1\ n_2]$.

# The wrapping procedure: the formulas (1/2)

1. Let $F$ be a facet of $P_0$ and a ridge $R$ of $F$ for which the facet $F'$ adjacent to $F$ on $R$ is not known.
2. Wrapping$(R, F, P_1, \ldots, P_k)$ computes a vertex $w$ of $F'$ so that $w \notin R$, thus, so that ConvexHull$(R \cup \{w\})$ is a $(d-1)$-simplex of $\mathbb{R}^d$.
3. we assume that $d-1$ vertices $v_1, \ldots, v_{d-1}$ of $R$, and a vertex $v_d$ of $F$ so that $v_d \notin R$ are known.
4. Let $n_1$ be the outer normal of $F$ and $r_1, \ldots, r_{d-2}$ a basis for the linear space spanned by $v_2 - v_1, \ldots, v_{d-1} - v_1$, satisfying $n_1 \perp r_j$ for $j = 1, \ldots, d-2$.
5. Let $n_2$ be another vector satisfying $n_2 \perp n_1$, $n_2 \perp r_j$ for all $j$, and $n_2^t(v_d - v_1) < 0$. The vectors $r_1, \ldots, r_{d-2}, n_1, n_2$ a basis of $\mathbb{R}^d$, assembled into the $d \times d$ matrix $B := [r_1 \cdots r_{d-2}\ n_1\ n_2]$.
6. The projection then maps a point $(x_1, \ldots, x_d)^t \in \mathbb{R}^d$ to $(y_1, y_2)^t \in \mathbb{R}^2$ via

$$y = T(x - v_1), \qquad T := \begin{bmatrix} 0 & \cdots & 0 & -1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} B^{-1} \in \mathbb{R}^{2 \times d},$$

# The wrapping procedure: the formulas (2/2)

1. Let us assume that no vertices of $P_1, \ldots, P_k$ project to a point $(y_1, y_2)^t$ satisfying $y_1 = 0$ and $y_2 > 0$.

# The wrapping procedure: the formulas (2/2)

1. Let us assume that no vertices of $P_1, \ldots, P_k$ project to a point $(y_1, y_2)^t$ satisfying $y_1 = 0$ and $y_2 > 0$.
2. $x \in P_0 = \text{ConvexHull}(P_1 \cup \cdots \cup P_k)$ can be written as a convex combination

$$x = \sum_{i=1}^{k} \lambda_i x^i, \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_i \geq 0, \quad x^i \in P_i.$$

# The wrapping procedure: the formulas (2/2)

1. Let us assume that no vertices of $P_1, \ldots, P_k$ project to a point $(y_1, y_2)^t$ satisfying $y_1 = 0$ and $y_2 > 0$.
2. $x \in P_0 = \text{ConvexHull}(P_1 \cup \cdots \cup P_k)$ can be written as a convex combination
$$x = \sum_{i=1}^{k} \lambda_i x^i, \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_i \geq 0, \quad x^i \in P_i.$$
3. The bilinear terms $\lambda_i x^i$ are linearized via the substitution $\bar{x}^i := \lambda_i x^i$.

# The wrapping procedure: the formulas (2/2)

1. Let us assume that no vertices of $P_1, \ldots, P_k$ project to a point $(y_1, y_2)^t$ satisfying $y_1 = 0$ and $y_2 > 0$.
2. $x \in P_0 = \text{ConvexHull}(P_1 \cup \cdots \cup P_k)$ can be written as a convex combination
$$x = \sum_{i=1}^{k} \lambda_i x^i, \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_i \geq 0, \quad x^i \in P_i.$$
3. The bilinear terms $\lambda_i x^i$ are linearized via the substitution $\bar{x}^i := \lambda_i x^i$.
4. To avoid maximizing the fractional objective function $y_2/y_1$ directly, we fix $y_1 = 1$ as a normalization constraint

# The wrapping procedure: the formulas (2/2)

1. Let us assume that no vertices of $P_1, \ldots, P_k$ project to a point $(y_1, y_2)^t$ satisfying $y_1 = 0$ and $y_2 > 0$.
2. $x \in P_0 = \text{ConvexHull}(P_1 \cup \cdots \cup P_k)$ can be written as a convex combination
$$x = \sum_{i=1}^{k} \lambda_i x^i, \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_i \geq 0, \quad x^i \in P_i.$$
3. The bilinear terms $\lambda_i x^i$ are linearized via the substitution $\bar{x}^i := \lambda_i x^i$.
4. To avoid maximizing the fractional objective function $y_2/y_1$ directly, we fix $y_1 = 1$ as a normalization constraint
5. The wrapping procedure thus reduces to solving:

$$\begin{aligned}
\max \quad & y_2 \\
\text{s.t.} \quad & A^i \bar{x}^i \leq b^i \lambda_i, \quad i = 1, \ldots, k, \\
& \lambda_i \geq 0, \quad i = 1, \ldots, k, \\
\text{null} \quad & x = \sum_{i=1}^{k} \bar{x}^i, \\
& y = T\left(x - v_1 \sum_{i=1}^{k} \lambda_i\right), \\
& y_1 = 1.
\end{aligned} \tag{LP}$$

# Plan

1. Overview
2. The wrapping procedure
3. Computing ECH in the bounded case
4. Reduction to the bounded case

# The bounded case (1/3)

(1) Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

**(3 a)** Choose a ridge $R \in \mathcal{R}$ and let $F$ be its facet from $\mathcal{F}$.

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

- **(3 a)** Choose a ridge $R \in \mathcal{R}$ and let $F$ be its facet from $\mathcal{F}$.
- **(3 b)** Let $w$ be the vertex returned by Wrapping$(R, F, P_1, \ldots, P_k)$. Recall that $w$ and $F$ determine a $(d-1)$-dimensional simplex $S$ which is either a facet of $P_0$ or a subset of a facet of $P_0$.

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

- **(3 a)** Choose a ridge $R \in \mathcal{R}$ and let $F$ be its facet from $\mathcal{F}$.
- **(3 b)** Let $w$ be the vertex returned by Wrapping$(R, F, P_1, \ldots, P_k)$. Recall that $w$ and $F$ determine a $(d-1)$-dimensional simplex $S$ which is either a facet of $P_0$ or a subset of a facet of $P_0$.
- **(3 c)** If $S$ and an element $F$ of $\mathcal{F}$ have the same supporting hyperplane then $F$ is replaced by $F \cup \{S\}$, otherwise $\mathcal{F}$ is replaced by $\mathcal{F} \cup \{\{S\}\}$.

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

- **(3 a)** Choose a ridge $R \in \mathcal{R}$ and let $F$ be its facet from $\mathcal{F}$.
- **(3 b)** Let $w$ be the vertex returned by Wrapping$(R, F, P_1, \ldots, P_k)$. Recall that $w$ and $F$ determine a $(d-1)$-dimensional simplex $S$ which is either a facet of $P_0$ or a subset of a facet of $P_0$.
- **(3 c)** If $S$ and an element $F$ of $\mathcal{F}$ have the same supporting hyperplane then $F$ is replaced by $F \cup \{S\}$, otherwise $\mathcal{F}$ is replaced by $\mathcal{F} \cup \{\{S\}\}$.
- **(3 d)** Add to $\mathcal{R}$ any edge of $S$ which is not already in $\mathcal{R}$

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

- **(3 a)** Choose a ridge $R \in \mathcal{R}$ and let $F$ be its facet from $\mathcal{F}$.
- **(3 b)** Let $w$ be the vertex returned by Wrapping$(R, F, P_1, \ldots, P_k)$. Recall that $w$ and $F$ determine a $(d-1)$-dimensional simplex $S$ which is either a facet of $P_0$ or a subset of a facet of $P_0$.
- **(3 c)** If $S$ and an element $F$ of $\mathcal{F}$ have the same supporting hyperplane then $F$ is replaced by $F \cup \{S\}$, otherwise $\mathcal{F}$ is replaced by $\mathcal{F} \cup \{\{S\}\}$.
- **(3 d)** Add to $\mathcal{R}$ any edge of $S$ which is not already in $\mathcal{R}$
- **(3 e)** Remove from $\mathcal{R}$ any ridge which, thanks to $S$, is now on two facets in $\mathcal{F}$.

# The bounded case (1/3)

**(1)** Initialize the set $\mathcal{F}$ to the $(d-1)$-dimensional facets of $P_1, \ldots, P_k$ that are valid for $P_1, \ldots, P_k$.

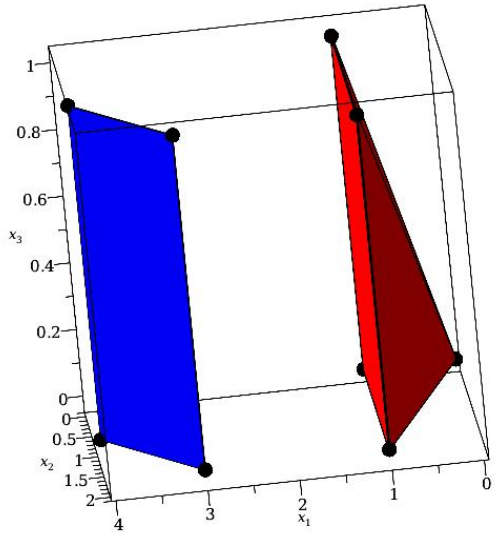
**(2)** Determine the set $\mathcal{R}$ of the ridges that belong to only one facet of $\mathcal{F}$. Note that, in the implementation, each ridge must remember the facet(s) it belongs to.

**(3)** While $\mathcal{R}$ is not empty do the following:

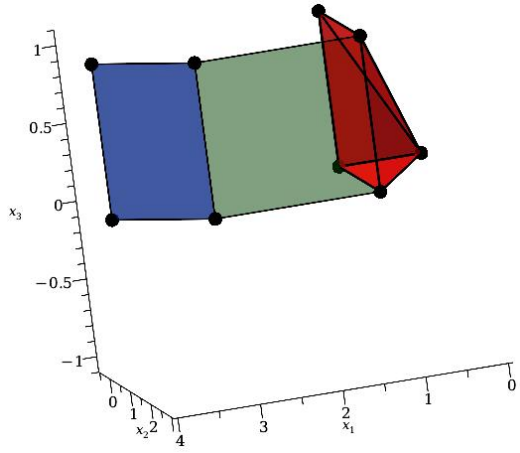
- **(3 a)** Choose a ridge $R \in \mathcal{R}$ and let $F$ be its facet from $\mathcal{F}$.
- **(3 b)** Let $w$ be the vertex returned by Wrapping$(R, F, P_1, \ldots, P_k)$. Recall that $w$ and $F$ determine a $(d-1)$-dimensional simplex $S$ which is either a facet of $P_0$ or a subset of a facet of $P_0$.
- **(3 c)** If $S$ and an element $F$ of $\mathcal{F}$ have the same supporting hyperplane then $F$ is replaced by $F \cup \{S\}$, otherwise $\mathcal{F}$ is replaced by $\mathcal{F} \cup \{\{S\}\}$.
- **(3 d)** Add to $\mathcal{R}$ any edge of $S$ which is not already in $\mathcal{R}$
- **(3 e)** Remove from $\mathcal{R}$ any ridge which, thanks to $S$, is now on two facets in $\mathcal{F}$.

**(4)** return $\mathcal{F}$.

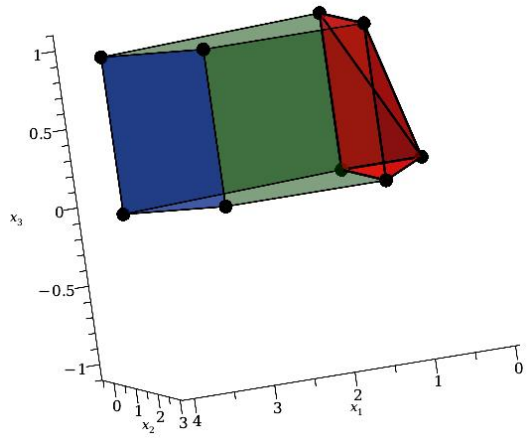
# The bounded case (2/3)



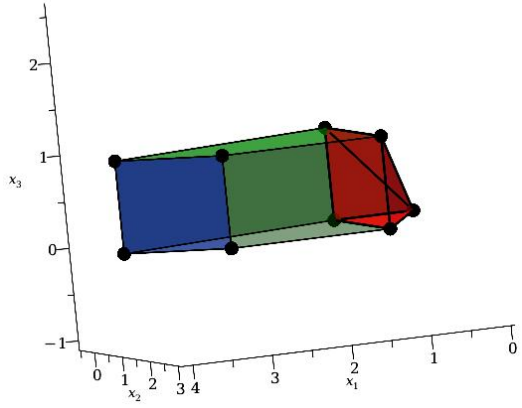

(a) Input



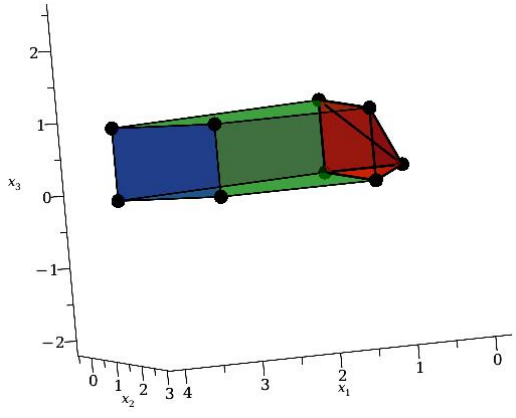

(b) Step 1




(c) Step 2

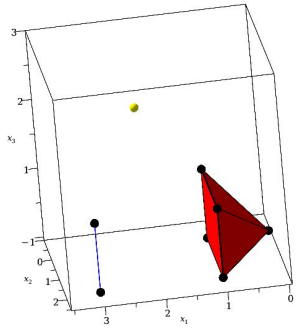



(d) Step 3

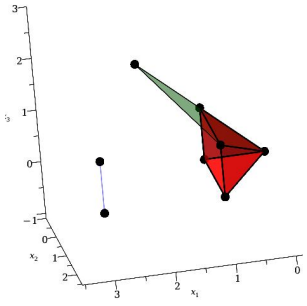



(e) Step 4

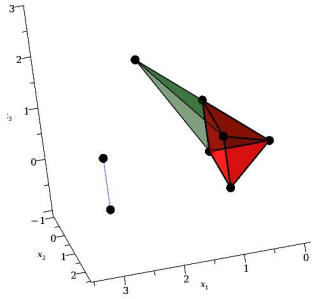
# The bounded case (3/3)



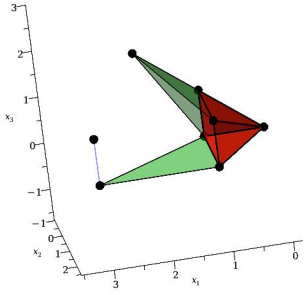
(a) Input



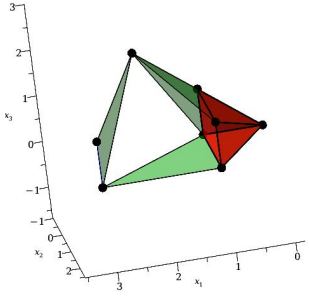
(b) Step 1



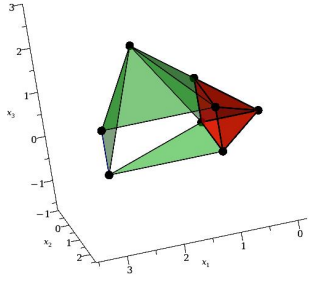
(c) Step 2



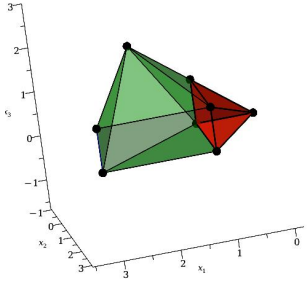
(d) Step 3



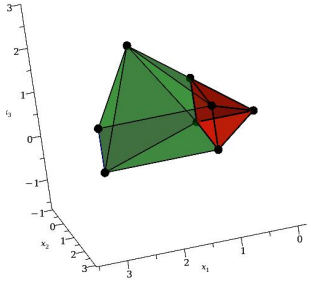
(e) Step 4



(f) Step 5



(g) Step 6



(h) Step 7

# Plan

1. Overview
2. The wrapping procedure
3. Computing ECH in the bounded case
4. Reduction to the bounded case

# Mapping an unbounded to a bounded polyhedron (1/3)

1. Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ be an unbounded polyhedron.

# Mapping an unbounded to a bounded polyhedron (1/3)

1. Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ be an unbounded polyhedron.
2. Let $\{v_1, \ldots, v_t\} \subseteq \mathbb{R}^d$ be the vertices of $P$ and let $\{r_1, \ldots, r_s\} \subseteq \mathbb{R}^d$ be its extreme rays.

# Mapping an unbounded to a bounded polyhedron (1/3)

1. Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ be an unbounded polyhedron.
2. Let $\{v_1, \ldots, v_t\} \subseteq \mathbb{R}^d$ be the vertices of $P$ and let $\{r_1, \ldots, r_s\} \subseteq \mathbb{R}^d$ be its extreme rays.
3. Assume that the homogeneization cone of $P$

$$T = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \in \mathbb{R}^{d+1} \mid Ax - \lambda b \leq 0, \lambda \geq 0 \right\}.$$

is pointed, that is, it does not contain any line.

# Mapping an unbounded to a bounded polyhedron (1/3)

1. Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ be an unbounded polyhedron.
2. Let $\{v_1, \ldots, v_t\} \subseteq \mathbb{R}^d$ be the vertices of $P$ and let $\{r_1, \ldots, r_s\} \subseteq \mathbb{R}^d$ be its extreme rays.
3. Assume that the homogeneization cone of $P$

$$T = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \in \mathbb{R}^{d+1} \mid Ax - \lambda b \leq 0, \lambda \geq 0 \right\}.$$

   is pointed, that is, it does not contain any line.
4. We transform $P$ into a bounded polyhedron $Q$ given in V-representation, using an invertible transformation matrix $S$.

# Mapping an unbounded to a bounded polyhedron (1/3)

1. Let $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ be an unbounded polyhedron.
2. Let $\{v_1, \ldots, v_t\} \subseteq \mathbb{R}^d$ be the vertices of $P$ and let $\{r_1, \ldots, r_s\} \subseteq \mathbb{R}^d$ be its extreme rays.
3. Assume that the homogeneization cone of $P$
$$T = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \in \mathbb{R}^{d+1} \mid Ax - \lambda b \leq 0, \lambda \geq 0 \right\}.$$
is pointed, that is, it does not contain any line.
4. We transform $P$ into a bounded polyhedron $Q$ given in V-representation, using an invertible transformation matrix $S$.
5. We then show that the H-representation of $P$ can easily be recovered from the H-representation of $Q$.

# Mapping an unbounded to a bounded polyhedron (2/3)

1. Homogenization$(P)$ is the conical hull of the columns of the matrix

$$M = \begin{pmatrix} V & R \\ \mathbf{1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(d+1)\times(t+s)},$$

where the columns of $V$ and $R$ are of the vertices and extreme rays of $P$.

# Mapping an unbounded to a bounded polyhedron (2/3)

1. Homogenization$(P)$ is the conical hull of the columns of the matrix
$$M = \begin{pmatrix} V & R \\ \mathbf{1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(d+1)\times(t+s)},$$
where the columns of $V$ and $R$ are of the vertices and extreme rays of $P$.
2. Since Homogenization$(P)$ is pointed, the interior of its dual cone is not empty and we can compute via LP a vector $s \in \mathbb{R}^{d+1}$ so that $M^t s > \mathbf{0}$ holds.

# Mapping an unbounded to a bounded polyhedron (2/3)

1. Homogenization$(P)$ is the conical hull of the columns of the matrix
$$M = \begin{pmatrix} V & R \\ \mathbf{1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(d+1)\times(t+s)},$$
where the columns of $V$ and $R$ are of the vertices and extreme rays of $P$.
2. Since Homogenization$(P)$ is pointed, the interior of its dual cone is not empty and we can compute via LP a vector $s \in \mathbb{R}^{d+1}$ so that $M^t s > \mathbf{0}$ holds.
3. Next, we compute a basis $(b_1, \ldots, b_d)$ of the linear space given by $s^t x = 0$ in $\mathbb{R}^{d+1}$.

# Mapping an unbounded to a bounded polyhedron (2/3)

1. Homogenization$(P)$ is the conical hull of the columns of the matrix
$$M = \begin{pmatrix} V & R \\ \mathbf{1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(d+1)\times(t+s)},$$
where the columns of $V$ and $R$ are of the vertices and extreme rays of $P$.
2. Since Homogenization$(P)$ is pointed, the interior of its dual cone is not empty and we can compute via LP a vector $s \in \mathbb{R}^{d+1}$ so that $M^t s > \mathbf{0}$ holds.
3. Next, we compute a basis $(b_1, \ldots, b_d)$ of the linear space given by $s^t x = 0$ in $\mathbb{R}^{d+1}$.
4. We form the $(d+1) \times (d+1)$ invertible matrix $S$ whose first $d$ rows are $b_1^t, \ldots, b_d^t$ and whose last row is $s^t$. All entries of the last row of the matrix $SM$ are positive.

# Mapping an unbounded to a bounded polyhedron (2/3)

1. Homogenization$(P)$ is the conical hull of the columns of the matrix
$$M = \begin{pmatrix} V & R \\ \mathbf{1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(d+1)\times(t+s)},$$
where the columns of $V$ and $R$ are of the vertices and extreme rays of $P$.
2. Since Homogenization$(P)$ is pointed, the interior of its dual cone is not empty and we can compute via LP a vector $s \in \mathbb{R}^{d+1}$ so that $M^t s > \mathbf{0}$ holds.
3. Next, we compute a basis $(b_1, \ldots, b_d)$ of the linear space given by $s^t x = 0$ in $\mathbb{R}^{d+1}$.
4. We form the $(d+1) \times (d+1)$ invertible matrix $S$ whose first $d$ rows are $b_1^t, \ldots, b_d^t$ and whose last row is $s^t$. All entries of the last row of the matrix $SM$ are positive.
5. Let $c_1, \ldots, c_{t+s}$ be the entries of that last row of $SM$. We form the diagonal matrix $D$ whose diagonal entries are $1/c_1, \ldots, 1/c_{t+s}$. Now the entries on the last row of the matrix $SMD$ are all equal to 1.

# Mapping an unbounded to a bounded polyhedron (2/3)

1. Homogenization$(P)$ is the conical hull of the columns of the matrix
$$M = \begin{pmatrix} V & R \\ \mathbf{1} & \mathbf{0} \end{pmatrix} \in \mathbb{R}^{(d+1)\times(t+s)},$$
where the columns of $V$ and $R$ are of the vertices and extreme rays of $P$.
2. Since Homogenization$(P)$ is pointed, the interior of its dual cone is not empty and we can compute via LP a vector $s \in \mathbb{R}^{d+1}$ so that $M^t s > \mathbf{0}$ holds.
3. Next, we compute a basis $(b_1, \ldots, b_d)$ of the linear space given by $s^t x = 0$ in $\mathbb{R}^{d+1}$.
4. We form the $(d+1) \times (d+1)$ invertible matrix $S$ whose first $d$ rows are $b_1^t, \ldots, b_d^t$ and whose last row is $s^t$. All entries of the last row of the matrix $SM$ are positive.
5. Let $c_1, \ldots, c_{t+s}$ be the entries of that last row of $SM$. We form the diagonal matrix $D$ whose diagonal entries are $1/c_1, \ldots, 1/c_{t+s}$. Now the entries on the last row of the matrix $SMD$ are all equal to 1.
6. Removing that last row, we obtain a matrix, the columns of which contain the vertices of the desired polytope $Q$.

# Mapping an unbounded to a bounded polyhedron (3/3)

1. If $Bx \leq c$ is an H-representation of $Q$, then, by definition, we have

$$\text{Homogenization}(Q) = \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \;\middle|\; \begin{pmatrix} B & -c \\ \mathbf{0} & -1 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \right\}. \tag{4.1}$$

# Mapping an unbounded to a bounded polyhedron (3/3)

1. If $Bx \leq c$ is an H-representation of $Q$, then, by definition, we have

$$\text{Homogenization}(Q) = \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \middle| \begin{pmatrix} B & -c \\ \mathbf{0} & -1 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \right\}. \tag{4.1}$$

2. In order to relate $Q$ to $P$, we define another polyhedral cone in $\mathbb{R}^{d+1}$ by

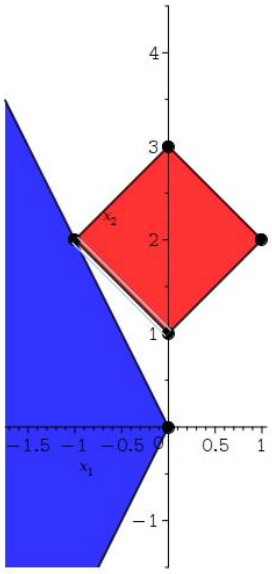
$$H := \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \middle| \begin{pmatrix} A & -b \\ \mathbf{0} & -1 \end{pmatrix} S^{-1} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \right\}, \tag{4.2}$$

where $Ax \leq b$ is an H-representation of $P$.

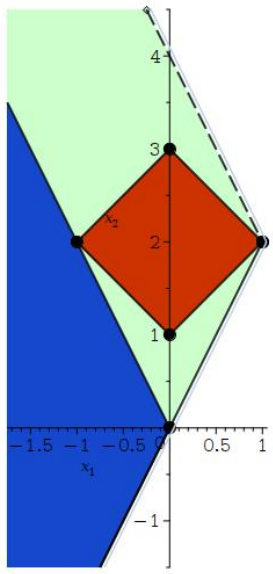
# Mapping an unbounded to a bounded polyhedron (3/3)

1. If $Bx \leq c$ is an H-representation of $Q$, then, by definition, we have
$$\text{Homogenization}(Q) = \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \middle| \begin{pmatrix} B & -c \\ \mathbf{0} & -1 \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \right\}. \tag{4.1}$$
2. In order to relate $Q$ to $P$, we define another polyhedral cone in $\mathbb{R}^{d+1}$ by
$$H := \left\{ \begin{pmatrix} x \\ t \end{pmatrix} \middle| \begin{pmatrix} A & -b \\ \mathbf{0} & -1 \end{pmatrix} S^{-1} \begin{pmatrix} x \\ t \end{pmatrix} \leq \mathbf{0} \right\}, \tag{4.2}$$
where $Ax \leq b$ is an H-representation of $P$.
3. **Theorem 1:** We have $\text{Homogenization}(Q) = H$.

# The convex hull of two closed sets may not be closed




(a) Input




(b) Output

Bounded polyhedral set:

$$\text{null} \begin{cases} -x_1 - x_2 <= -1 \\ -x_1 + x_2 <= 3 \\ x_1 - x_2 <= -1 \\ x_1 + x_2 <= 3 \end{cases}$$

Unbounded polyhedral set:

$$\begin{cases} x_1 - \frac{x_2}{2} <= 0 \\ x_1 + \frac{x_2}{2} <= 0 \end{cases}$$

The green region is contained in the convex hull of the blue and the red polyhedra. But the dotted line is no.

**Theorem 2:** $\text{ConvexHull}(V) + \text{ConicalHull}(R) = \overline{P_0}$.

# Reduction to the bounded case: algorithm (1/2)

1. Let $t_i$ and $s_i$ be the numbers of vertices and the extreme rays of $P_i$. We define the following matrix

$$M = \begin{pmatrix} V_1 & R_1 & \cdots & V_k & R_k \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \end{pmatrix}$$

# Reduction to the bounded case: algorithm (1/2)

1. Let $t_i$ and $s_i$ be the numbers of vertices and the extreme rays of $P_i$. We define the following matrix
$$M = \begin{pmatrix} V_1 & R_1 & \cdots & V_k & R_k \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \end{pmatrix}$$
2. Then, we follow Steps (**2**) to (**5**) of the previous algorithm. Again, the entries of the last row of the matrix product $SMD$ are all equal to 1.

# Reduction to the bounded case: algorithm (1/2)

1. Let $t_i$ and $s_i$ be the numbers of vertices and the extreme rays of $P_i$. We define the following matrix
$$M = \begin{pmatrix} V_1 & R_1 & \cdots & V_k & R_k \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \end{pmatrix}$$
2. Then, we follow Steps (**2**) to (**5**) of the previous algorithm. Again, the entries of the last row of the matrix product $SMD$ are all equal to 1.
3. Removing the last row, the first $t_1 + s_1$ columns correspond to a bounded polyhedron $Q_1$, the next $t_2 + s_2$ columns correspond to a bounded polyhedron $Q_2$, etc.

# Reduction to the bounded case: algorithm (1/2)

1. Let $t_i$ and $s_i$ be the numbers of vertices and the extreme rays of $P_i$. We define the following matrix
$$M = \begin{pmatrix} V_1 & R_1 & \cdots & V_k & R_k \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \end{pmatrix}$$
2. Then, we follow Steps (**2**) to (**5**) of the previous algorithm. Again, the entries of the last row of the matrix product $SMD$ are all equal to 1.
3. Removing the last row, the first $t_1 + s_1$ columns correspond to a bounded polyhedron $Q_1$, the next $t_2 + s_2$ columns correspond to a bounded polyhedron $Q_2$, etc.
4. Next, we compute $Q_0 :=$ ConvexHull$(Q_1, \ldots, Q_k)$.

# Reduction to the bounded case: algorithm (1/2)

1. Let $t_i$ and $s_i$ be the numbers of vertices and the extreme rays of $P_i$. We define the following matrix
$$M = \begin{pmatrix} V_1 & R_1 & \cdots & V_k & R_k \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \end{pmatrix}$$
2. Then, we follow Steps (**2**) to (**5**) of the previous algorithm. Again, the entries of the last row of the matrix product $SMD$ are all equal to 1.
3. Removing the last row, the first $t_1 + s_1$ columns correspond to a bounded polyhedron $Q_1$, the next $t_2 + s_2$ columns correspond to a bounded polyhedron $Q_2$, etc.
4. Next, we compute $Q_0 := \text{ConvexHull}(Q_1, \ldots, Q_k)$.
5. Let $K_0$ be the homogenization cone of $Q_0$ and $C_0$ be the result of inverting the map $x \longmapsto SxD$ on $K_0$.

# Reduction to the bounded case: algorithm (1/2)

1. Let $t_i$ and $s_i$ be the numbers of vertices and the extreme rays of $P_i$. We define the following matrix
$$M = \begin{pmatrix} V_1 & R_1 & \cdots & V_k & R_k \\ \mathbf{1} & \mathbf{0} & \cdots & \mathbf{1} & \mathbf{0} \end{pmatrix}$$
2. Then, we follow Steps (**2**) to (**5**) of the previous algorithm. Again, the entries of the last row of the matrix product $SMD$ are all equal to 1.
3. Removing the last row, the first $t_1 + s_1$ columns correspond to a bounded polyhedron $Q_1$, the next $t_2 + s_2$ columns correspond to a bounded polyhedron $Q_2$, etc.
4. Next, we compute $Q_0 :=$ ConvexHull$(Q_1, \ldots, Q_k)$.
5. Let $K_0$ be the homogenization cone of $Q_0$ and $C_0$ be the result of inverting the map $x \longmapsto SxD$ on $K_0$.
6. Finally, dehomogenizing $C_0$, we obtain $\overline{P_0}$,

# Reduction to the bounded case: algorithm (2/2)

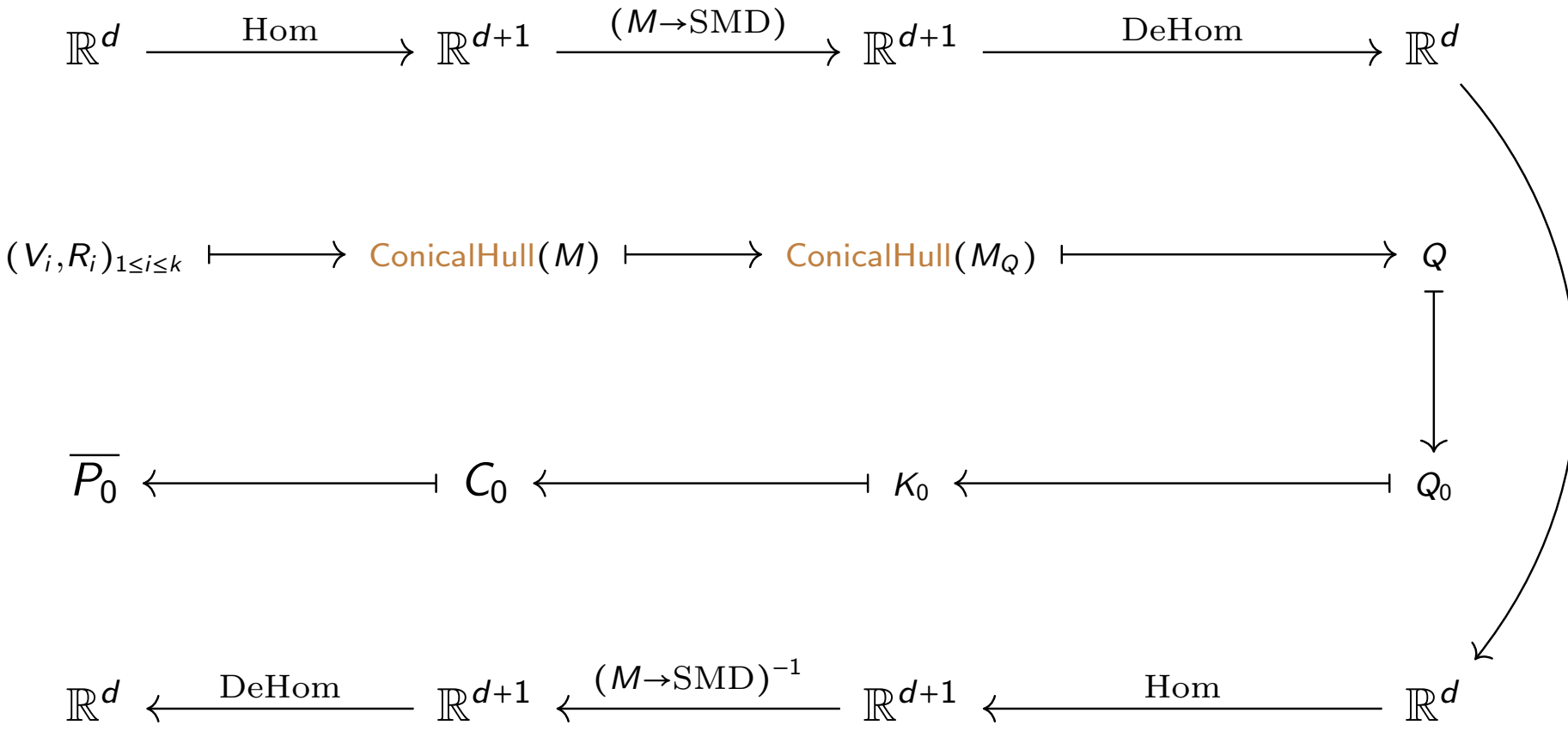
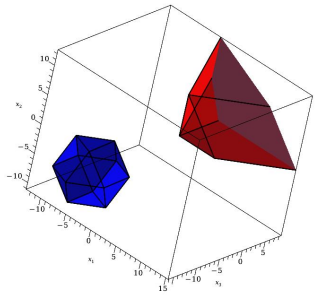
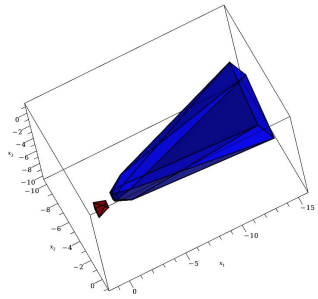



Figure: Reduction to the bounded case in 7 steps.

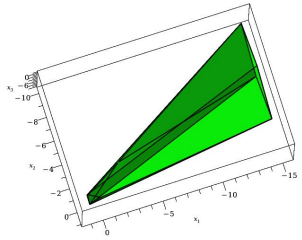
# Reduction to the bounded case: example (1/2)



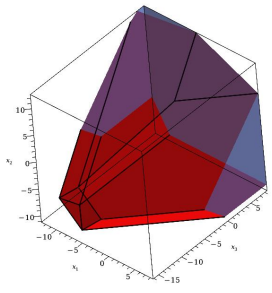
(a) Input



(b) Step 1



(c) Step 2



(d) Step 3

Octahedron:

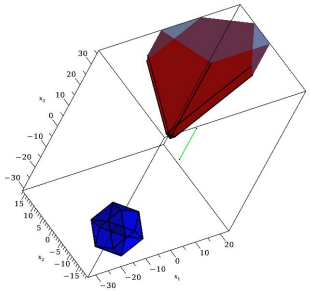
$$\text{null}\begin{cases} -x_3 \le 15 \\ x_3 \le -7 \\ -x_2 \le 10 \\ x_2 \le -2 \\ -x_1 - x_2 - x_3 \le 31 \\ -x_1 - x_2 + x_3 \le 9 \\ -x_1 \le 10 \\ -x_1 + x_2 - x_3 \le 19 \\ -x_1 + x_2 + x_3 \le -3 \\ x_1 - x_2 - x_3 \le 19 \\ -x_2 + x_3 + x_1 \le -3 \\ x_1 \le -2 \\ x_1 + x_2 - x_3 \le 7 \\ x_2 + x_3 + x_1 \le -15 \end{cases}$$

Unbounded polyhedral set:

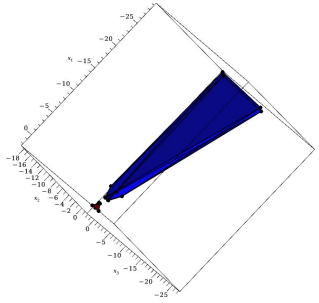
$$\begin{cases} -x_3 \le 0 \\ -x_2 \le 0 \\ -x_1 - \frac{x_2}{2} \le 0 \\ -x_1 - \frac{x_3}{3} \le 0 \\ x_1 + x_2 - x_3 \le 7 \\ x_1 + 2x_2 - x_3 \le 14 \end{cases}$$

Figure: Computation of the convex hull for a truncated octahedron and an unbounded polyhedral set.

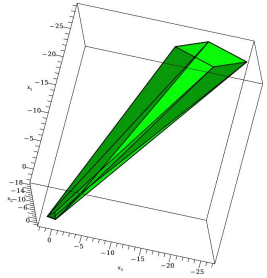
# Reduction to the bounded case: example (2/2)



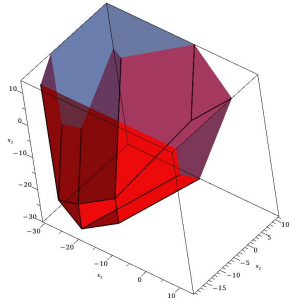
(a) Input



(b) Step 1



(c) Step 2



(d) Step 3

Octahedron:

$$\begin{cases} -x_3 \leq 28 \\ x_3 \leq -16 \\ -x_2 \leq 18 \\ x_2 \leq -6 \\ -x_1 - x_2 - x_3 \leq 66 \\ -x_1 - x_2 + x_3 \leq 22 \\ -x_1 \leq 26 \\ -x_1 + x_2 - x_3 \leq 42 \\ -x_1 + x_2 + x_3 \leq -2 \\ x_1 - x_2 - x_3 \leq 26 \\ -x_2 + x_3 + x_1 \leq -18 \\ x_1 \leq -14 \\ x_1 + x_2 - x_3 \leq 2 \\ x_2 + x_3 + x_1 \leq -42 \end{cases}$$

null

Unbounded polyhedral set:

$$\begin{cases} -x_3 \leq 0 \\ -x_2 \leq 0 \\ -x_1 - \frac{x_2}{2} \leq 0 \\ -x_1 - \frac{x_3}{3} \leq 0 \\ x_1 + x_2 - x_3 \leq 2 \\ x_1 + 2x_2 - x_3 \leq 4 \end{cases}$$

Line segment:

$$\begin{cases} -x_3 \leq 0 \\ x_3 \leq 15 \\ x_2 = -8 \\ x_1 = -2 \end{cases}$$

Figure: Computation of the convex hull for a truncated octahedron, an unbounded polyhedral set and a lower dimensional polytope which is a line segment.

## References


[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 geom 1991, pp. 98–104.

[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.

[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.

[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 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] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: Proceedings of the sixteenth annual ACM symposium on Theory of cor STOC '84. New York, NY, USA: ACM, 1984, pp. 302–311. ISBN: 0-89791-133-4.

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

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

[17] 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.

[18] 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.

[19] S. Verdoolaege. “Integer set coalescing”. In: International Workshop on Polyhedral Compilation Techniques, Date: 2 2015.