The integer hull of polyhedral sets

Jürgen Gerhard[1] Marc Moreno Maza[2] Linxiao Wang[2]

[1] MapleSoft, Canada
[2] ORCCA, University of Western Ontario, Canada

October 12, 2021

Outline

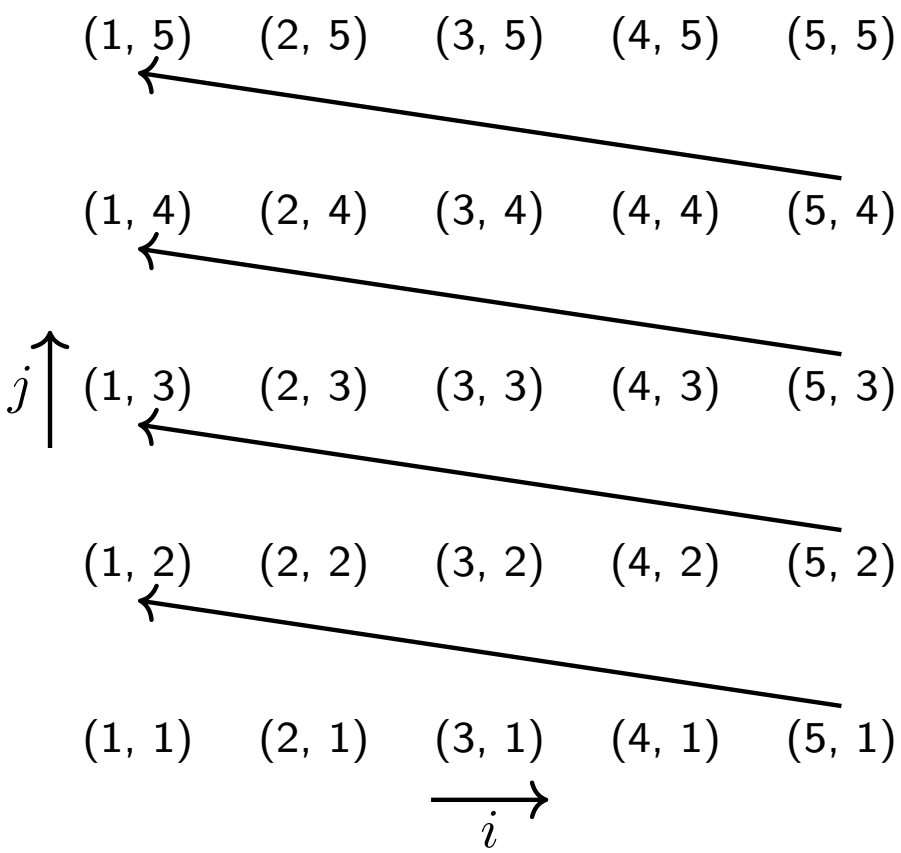
Outline

Motivation

Data dependence analysis for arrays:

for $i = 1$ to 5 do
  for $j = i$ to 5 do
    $A[i, j+1] = A[5, j]$

Can we parallel the two for loops?



Motivation

Data dependence analysis for arrays:

for $i = 1$ to 5 do
  for $j = i$ to 5 do
    $A[i, j+1] = A[5, j]$

Can we parallel the two for loops?

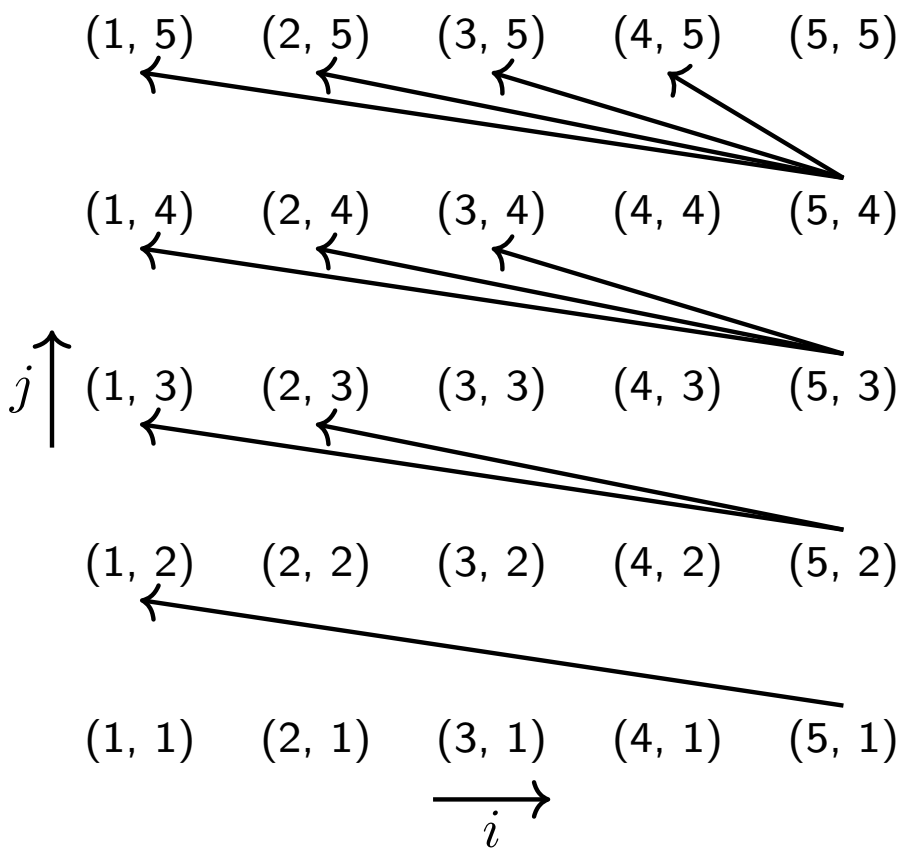
$i', j', i, j \in \mathbb{Z}$

$1 \le i \le j \le 5 \Rightarrow$ no solution!!

$1 \le i' \le j' \le 5 \Rightarrow$ no dependence exists!!

$i = 5 \Rightarrow$ can be parallelized

$j + 1 = j'$



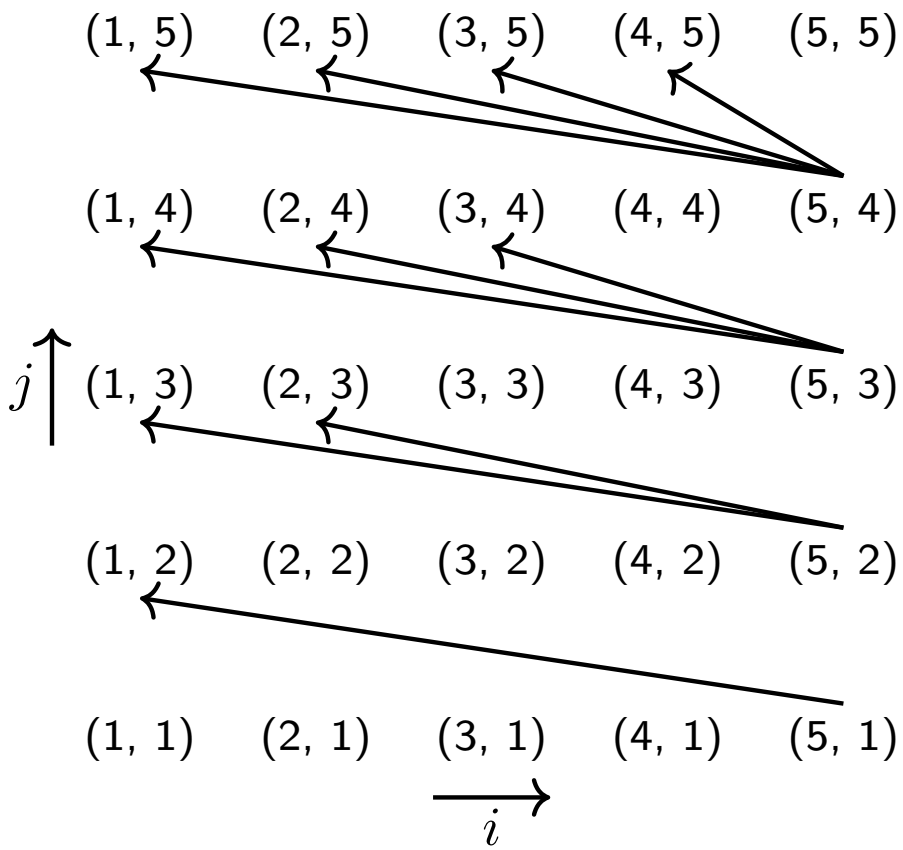
Motivation

Data dependence analysis for arrays:

for $i = 1$ to 5 do
  for $j = i$ to 5 do
    $A[i, j+1] = A[5, j]$

Can we parallel the two for loops?

$$i', j', i, j \in \mathbb{Z}$$
$$1 \le i \le j \le 5 \Rightarrow \text{no solution!!}$$
$$1 \le i' \le j' \le 5 \Rightarrow \text{no dependence exists!!}$$
$$i = 5 \Rightarrow \text{can be parallelized}$$
$$j + 1 = j'$$



Motivation

Data dependence analysis for arrays:

for $i = 1$ to 5 do
  for $j = i$ to 5 do
    $A[i,\ j+1] = A[5,\ j]$

Can we parallel the two for loops?

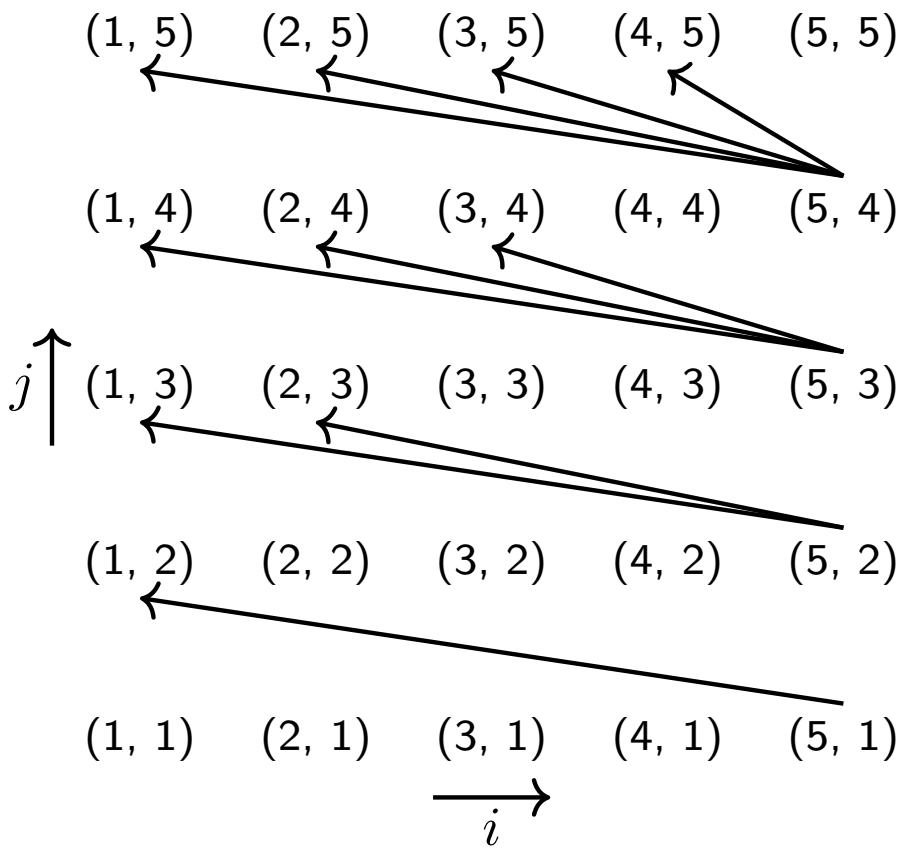
$i', j', i, j \in \mathbb{Z}$

$1 \le i \le j \le 5$ $\Rightarrow$ no solution!!

$1 \le i' \le j' \le 5$ $\Rightarrow$ no dependence exists!!

$i = 5$ $\Rightarrow$ can be parallelized

$j + 1 = j'$



Simple example

```
for(i = 0; i ≤ n; i ++)
  for(j = i; j ≤ n; j ++)
    A[i][j]...
```

$$\begin{cases} 0 \leq i \leq n \\ i \leq j \leq n \end{cases}$$

- Loop counters can only be integers
- This leads to the problem of finding the integer points of a polyhedral set, called the iteration space
- Often this space is parametric (e.g. the variable n)

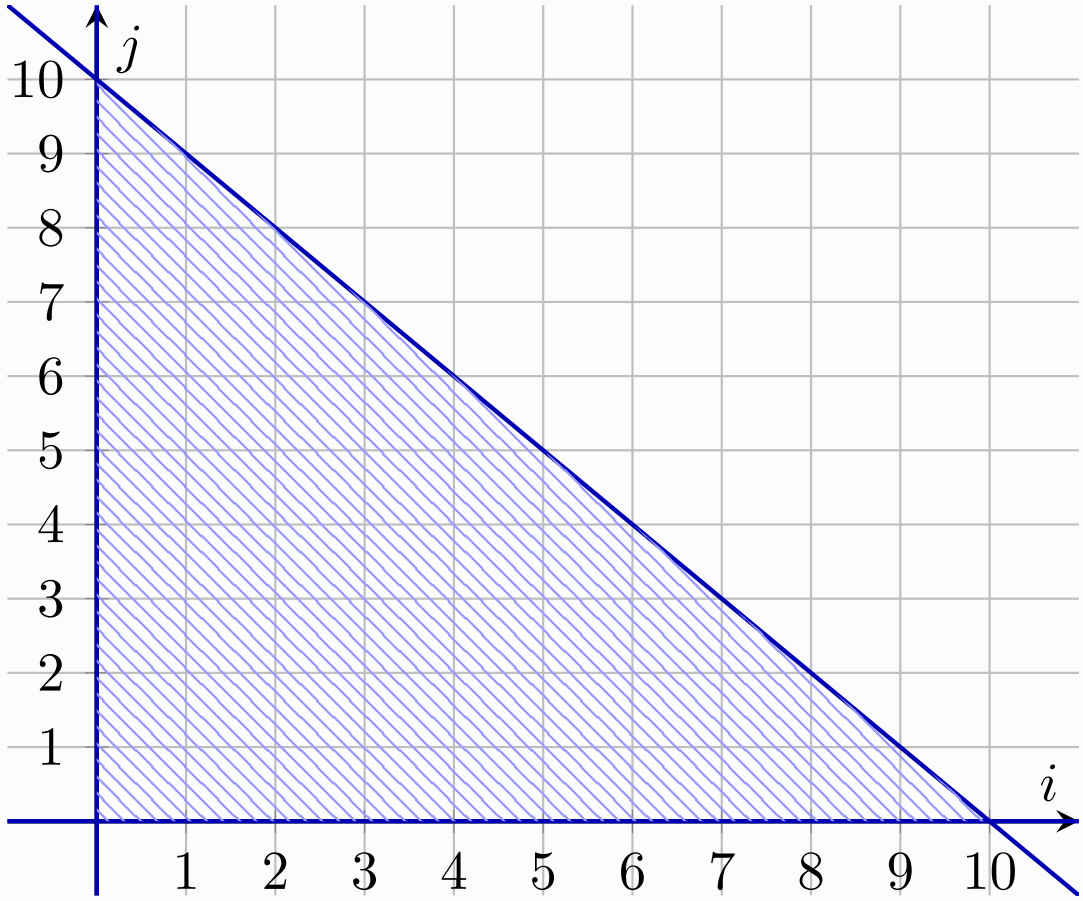
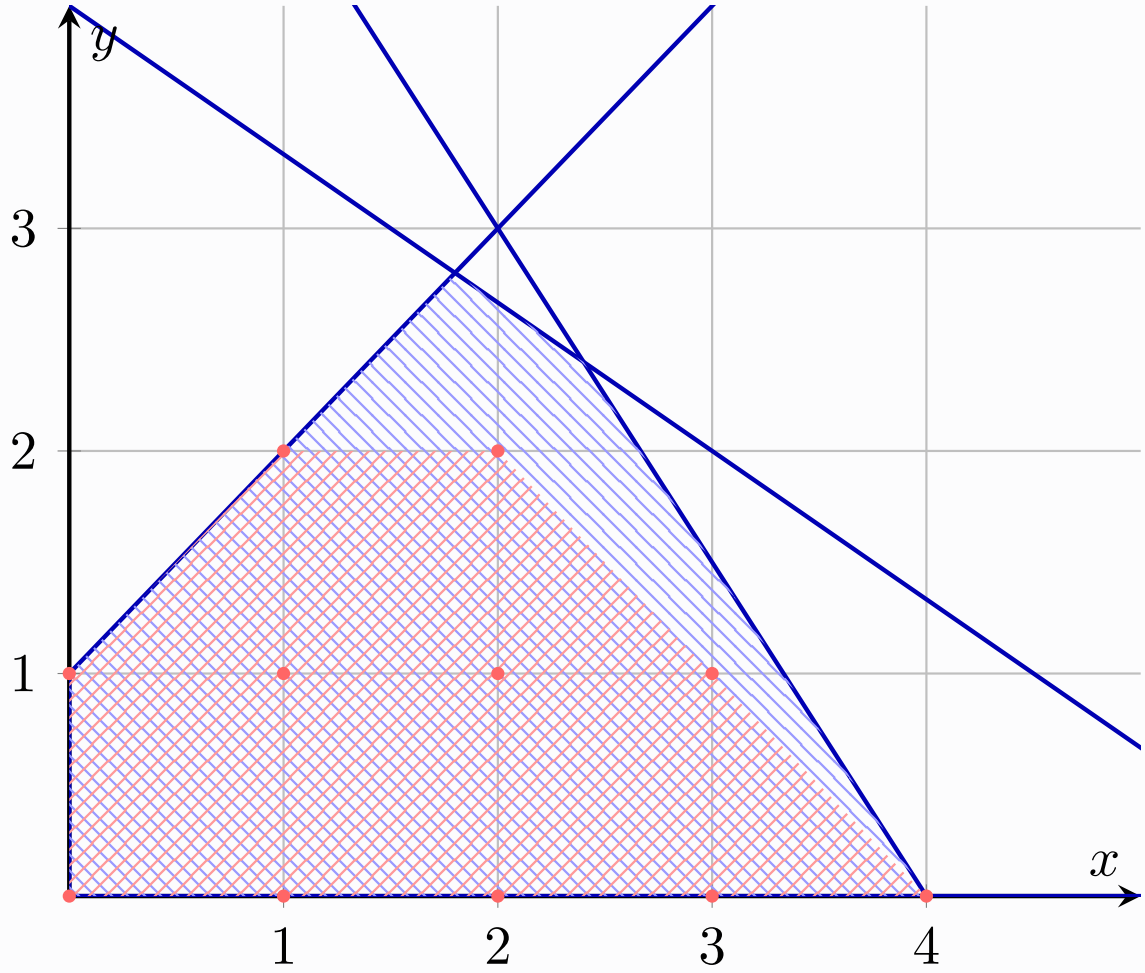


Figure: Iteration space when $n = 10$

Simple example

$$\begin{cases} 0 & \leq & x \\ 0 & \leq & y \\ 3\,x + 2\,y & \leq & 12 \\ 2\,x + 3\,y & \leq & 12 \\ -x + y & \leq & 1 \end{cases}$$

$$\begin{cases} 0 & \leq & x \\ 0 & \leq & y \\ y & \leq & 2 \\ x + y & \leq & 4 \\ -x + y & \leq & 1 \end{cases}$$

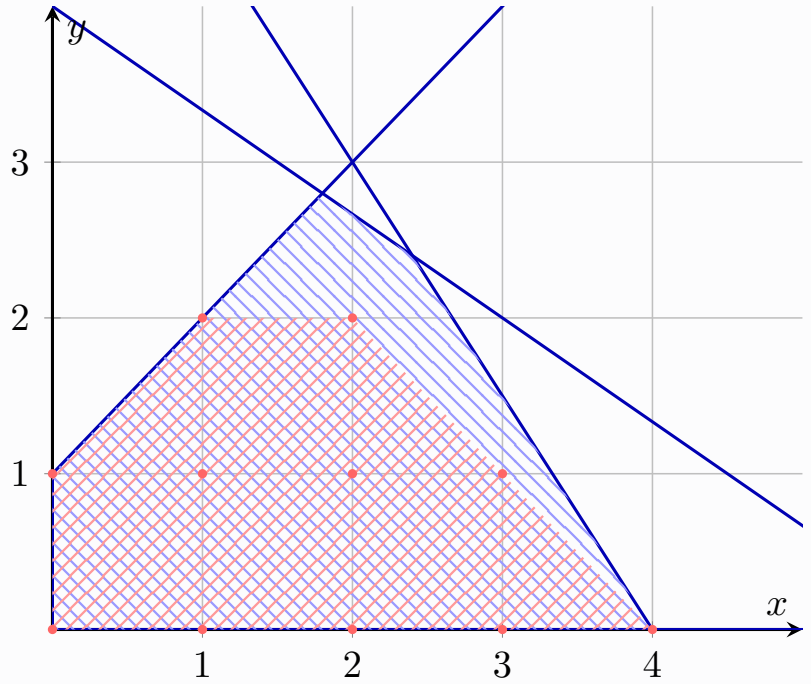


Figure

Outline

Convex Polyhedron and Integer Hull

- A subset $P \subseteq \mathbb{Q}^n$ is called a convex polyhedral set (or simply a polyhedral set) if
$$P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$$
holds, for a matrix $A \in \mathbb{Q}^{m \times n}$ and a vector $\mathbf{b} \in \mathbb{Q}^m$, where n, m are positive integers.
- We are interested in computing P_I the *integer hull* of P that is the smallest convex polyhedral set containing all the integer points of P.



The ZPolyhedralSet

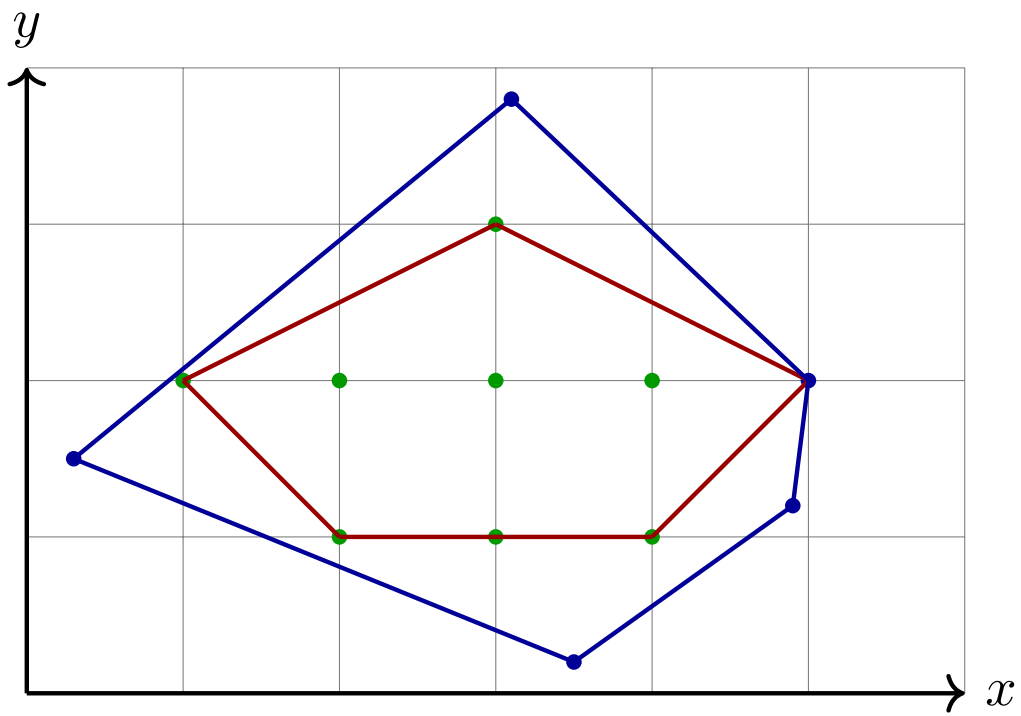
- **ZPolyhedralSet:** defined as the intersection of a PolyhedralSet and a lattice.

```
1  ZPolyhedralSets := module()
2          option package;
3
4  export
5          IntegerPointDecomposition, IsEmpty, IsIntegerPointOf,
6          PlotIntegerPoints3d, EnumerateIntegerPoints, IsContained;
7
8  $include <Lattice.mm>
9  $include <auxiliary_functions.mm>
10 $include <ZPolyhedralSet.mm>
11
12 end module;
```

- Integer hull is an ZPolyhedralSet where the lattice is the standard lattice of all points with integer coordinates.

Integer Hull

- The convex hull of all integer points inside a polyhedral set is its **integer hull**
- The complexity of computing all the integer points is related to the volume of the polyhedral set
- Vertices or Facets are enough to describe the integer hull



Vertices of Integer Hull

- $P = P_I$ if and only if every vertex of P is integer point.
- The convex hull of all the vertices of P_I is P_I itself.
- One way to represent an integer hull is to give all its vertices
- Our problem becomes:
  Given a polyhedron P, find all the vertices V of P_I.

Number of vertices

- The earlier study by Cook, Hartmann, Kannan and McDiarmid, shows that the number of vertices of P_I is related to the *size* of the coefficients of the inequalities that describe P.
- $x = p/q$ is a rational number, p and q are coprime
- $size(x) = 1 + \lceil (log(|p| + 1)) \rceil + \lceil (log(|q| + 1)) \rceil$
- The size of a linear inequality with coefficient vector $\vec{a} = (a_1, \ldots, a_n)$ is
$$size(\vec{a}) = n + size(a_1) + \ldots + size(a_n)$$
- For a polyhedron $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ where matrix $A \in \mathbb{Q}^{m \times n}$ and vector $\mathbf{b} \in \mathbb{Q}^m$, φ is the maximum size of the m inequalities, then the number of vertices of P_I is at most $2m^n(6n^2\varphi)^{n-1}$.

Vertices of Integer Hull

- $P = P_I$ if and only if every vertex of P is integer point.
- The convex hull of all the vertices of P_I is P_I itself.
- One way to represent an integer hull is to give all its vertices
- Our problem becomes:
  Given a polyhedron P, find all the vertices V of P_I.

Number of vertices

- The earlier study by Cook, Hartmann, Kannan and McDiarmid, shows that the number of vertices of P_I is related to the *size* of the coefficients of the inequalities that describe P.
- $x = p/q$ is a rational number, p and q are coprime
- $size(x) = 1 + \lceil (log(|p| + 1)) \rceil + \lceil (log(|q| + 1)) \rceil$
- The size of a linear inequality with coefficient vector $\vec{a} = (a_1, \ldots, a_n)$ is
$$size(\vec{a}) = n + size(a_1) + \ldots + size(a_n)$$
- For a polyhedron $P = \{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ where matrix $A \in \mathbb{Q}^{m \times n}$ and vector $\mathbf{b} \in \mathbb{Q}^m$, φ is the maximum size of the m inequalities, then the number of vertices of P_I is at most $2m^n(6n^2\varphi)^{n-1}$.

Outline

The `IntegerHull` procedure for 2D non-parametric case

```
> with(PolyhedralSets) :
> ineqs := [2 x + 5 y ≤ 64, 7 x + 5 y ≥ 20 , 3 x − 6 y ≤ -7] :
> poly := PolyhedralSet(ineqs, [x, y]);
```

$$poly := \begin{cases} Coordinates & : & [x, y] \\ Relations & : & \left[-x - \frac{5\,y}{7} \le -\frac{20}{7}, x - 2\,y \le -\frac{7}{3}, x + \frac{5\,y}{2} \le 32\right] \end{cases}$$

```
> IntegerHull(poly);
```

$$[[[12, 8], [-8, 16], [-7, 14], [-5, 11], [0, 4], [1, 3], [3, 3], [11, 7]], [\,]]$$

```
> IntegerHull(poly, returntype=polyhedralset);
```

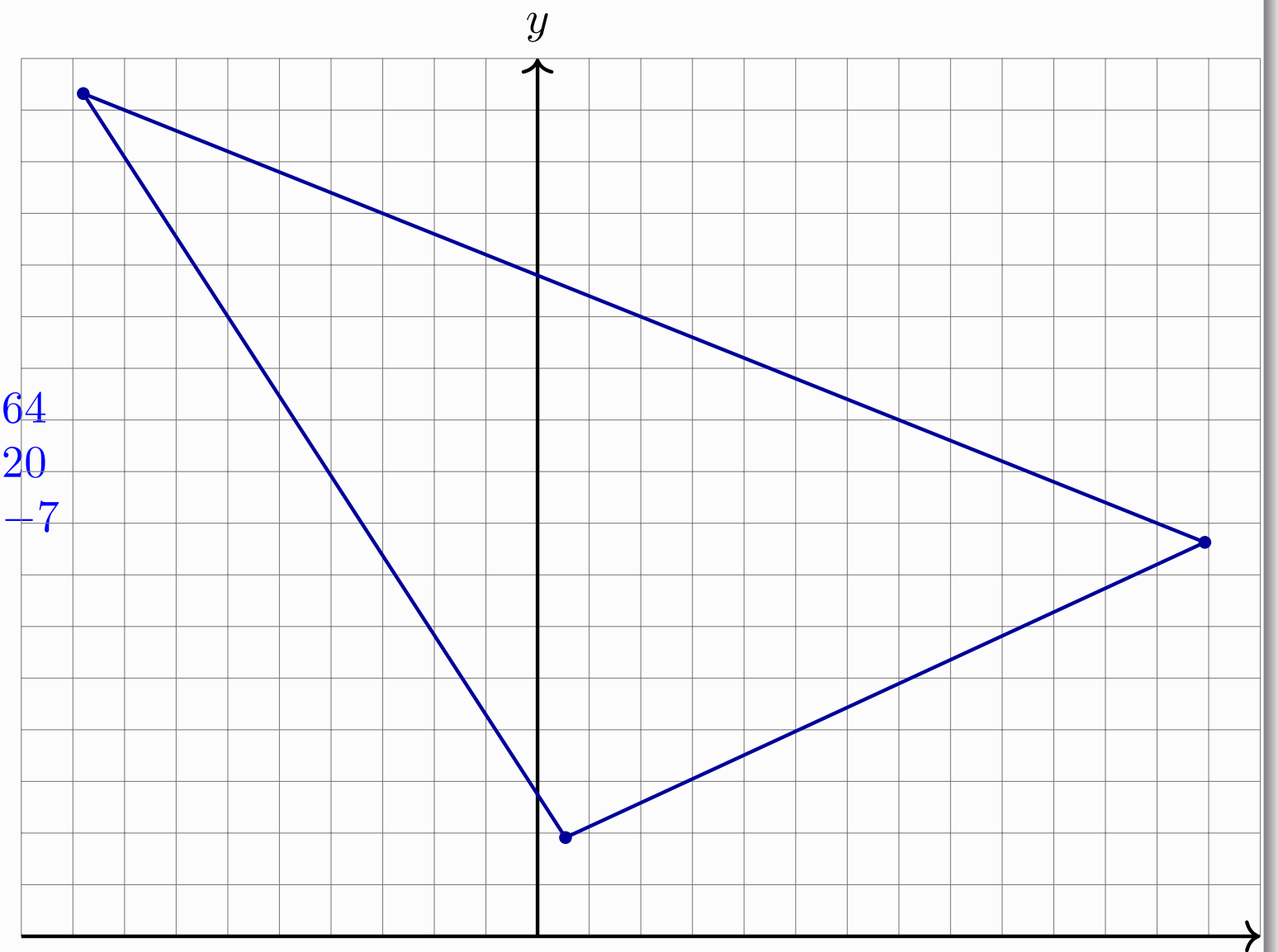
$$\begin{cases} Coordinates & : & [x, y] \\ Relations & : & \left[-y \le -3, -x - y \le -4, -x - \frac{5\,y}{7} \le -\frac{20}{7}, -x - \frac{2\,y}{3} \le -\frac{7}{3}, -x - \frac{y}{2} \le 0, x - 2\,y \le -3, x - \right. \end{cases}$$

```
>
```

Example (0/4)

Vertices: $(-44/5, 408/25), (349/27, 206/27), (85/57, 109/57)$

$$\begin{cases} 2x + 5y & \leq & 64 \\ 7x + 5y & \geq & 20 \\ 3x - 6y & \leq & -7 \end{cases}$$



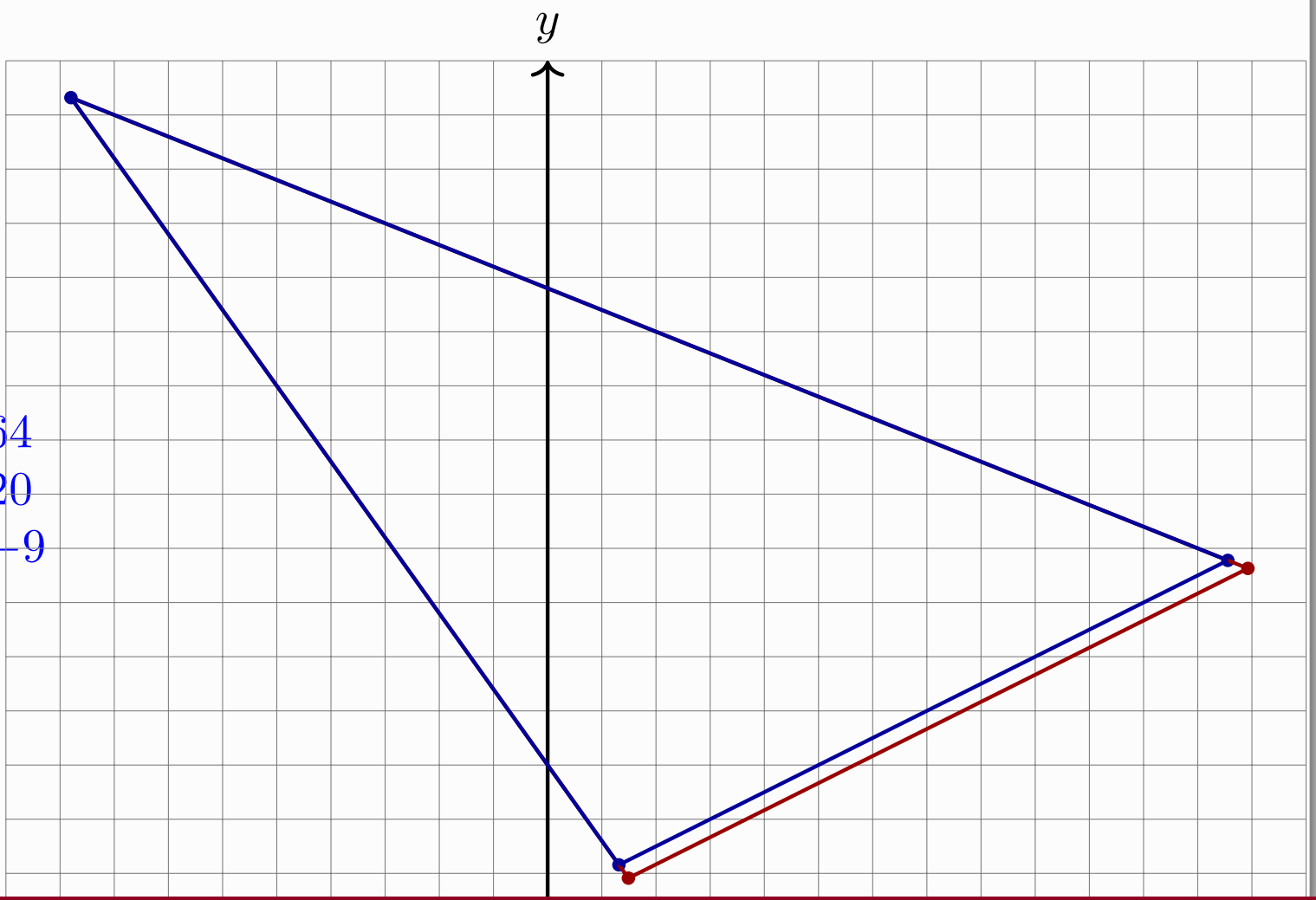
Example (1/4)

Replace the facets that could not have integer point

Vertices: $(-44/5, 408/25)$,~~$(349/27, 206/27)$~~,~~$(85/57, 109/57)$~~,
$(113/9, 70/9)$,$(25/19, 41/19)$

~~$3x - 6y \leq -7$~~

$$\begin{cases} 2x + 5y & \leq & 64 \\ 7x + 5y & \geq & 20 \\ 3x - 6y & \leq & -9 \end{cases}$$

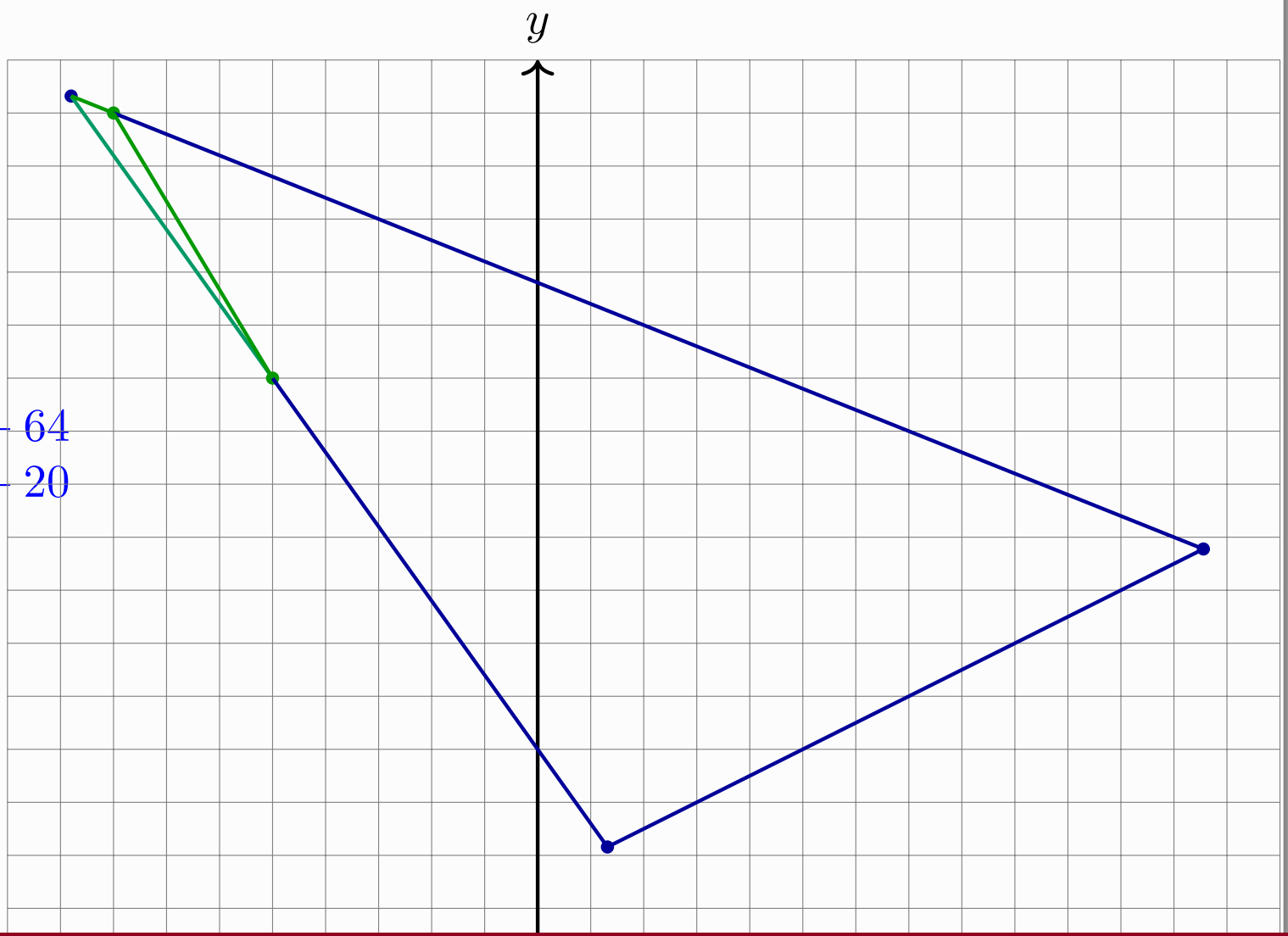


Example (2-1/4)

Vertices: $(-44/5, 408/25), (113/9, 70/9), (25/19, 41/19)$

Find the triangle with vertices: $(-8, 16), (-44/5, 408/25), (-5, 11)$

$$\begin{cases} 5y & \leq & -2x + 64 \\ 5y & \geq & -7x + 20 \\ 2y & \geq & x + 3 \end{cases}$$

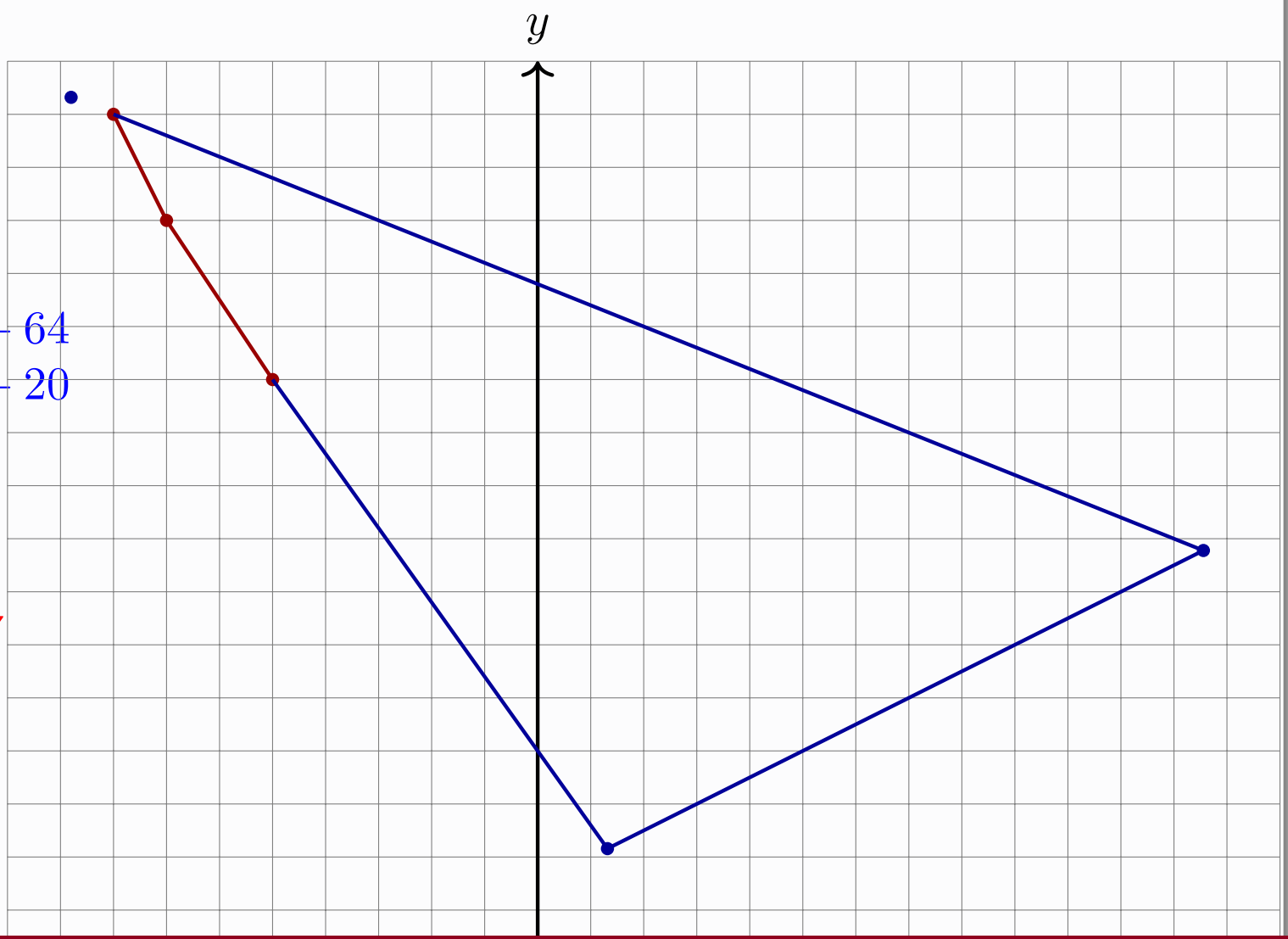


Example (2-2/4)

Compute the integer hull of the triangle
Vertices: $(-8, 16), (-7, 14), (-5, 11), (113/9, 70/9), (25/19, 41/19)$

$$\begin{cases} 5y & \leq & -2x + 64 \\ 5y & \geq & -7x + 20 \\ 2y & \geq & x + 3 \end{cases}$$

$$\begin{cases} y & \geq & -2x \\ 2y & \geq & 3x + 7 \end{cases}$$

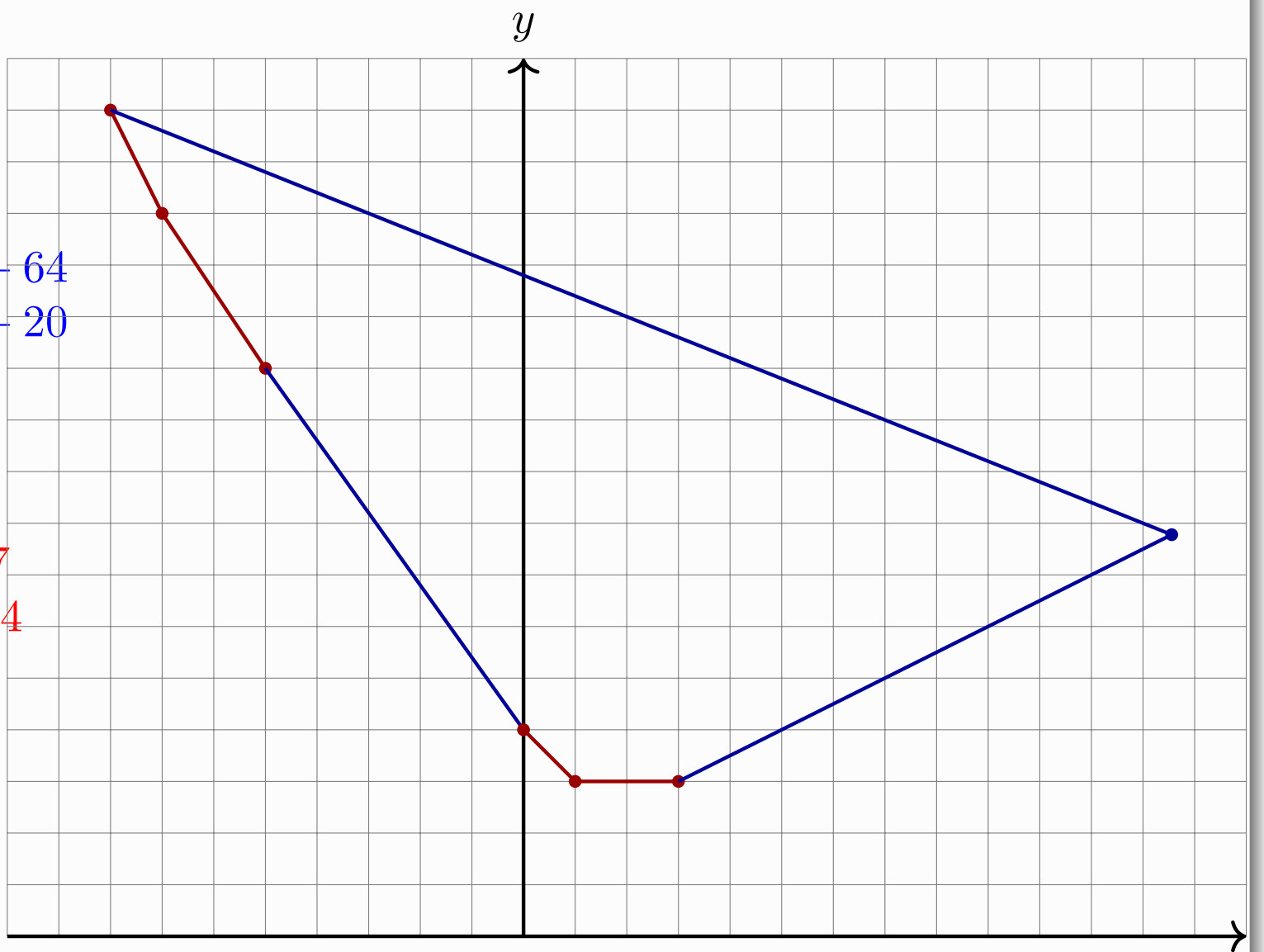


Example (3/4)

Vertices: $(-8, 16), (-7, 14), (-5, 11), (0, 4), (1, 3), (3, 3), (25/19, 41/19)$

$$\begin{cases} 5y & \leq & -2x + 64 \\ 5y & \geq & -7x + 20 \\ 2y & \geq & x + 3 \end{cases}$$

$$\begin{cases} y & \geq & -2x \\ 2y & \geq & 3x + 7 \\ y & \geq & -x + 4 \\ y & \geq & 3 \end{cases}$$

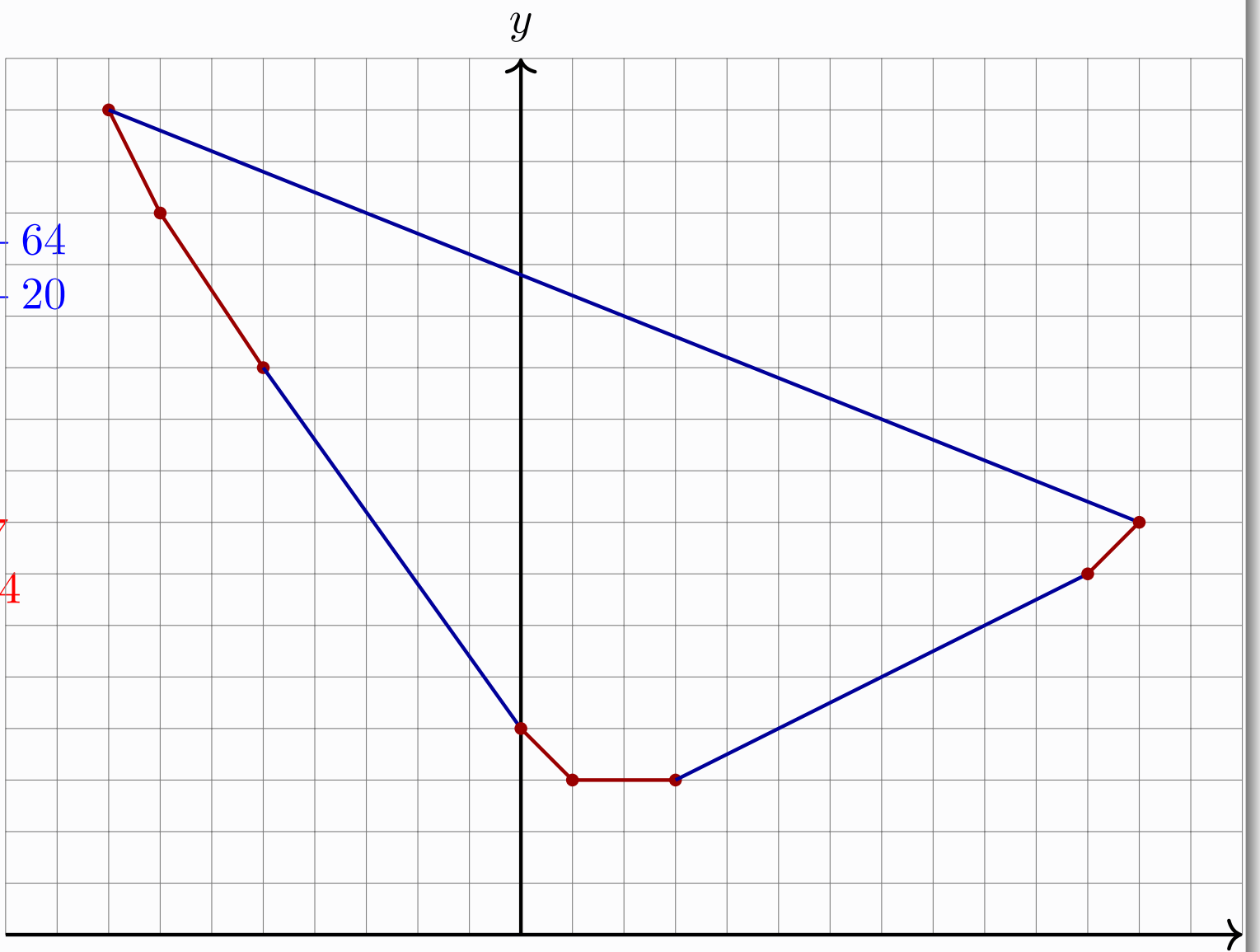


Example (4/4)

Vertices: $(-8, 16), (-7, 14), (-5, 11), (0, 4), (1, 3), (3, 3), (11, 7), (12, 8)$

$$\begin{cases} 5y & \leq & -2x + 64 \\ 5y & \geq & -7x + 20 \\ 2y & \geq & x + 3 \end{cases}$$

$$\begin{cases} y & \geq & -2x \\ 2y & \geq & 3x + 7 \\ y & \geq & -x + 4 \\ y & \geq & 3 \\ y & \geq & x - 4 \end{cases}$$



Benchmarks

What do we compare to

The following two steps together compute the same results as our algorithm

- `ZPolyhedralSets:-EnumerateIntegerPoints()`
  produces all the integer points in a polyhedral set
- `ComputationalGeometry:-ConvexHull()`
  computes the convex hull of a list of points and returns the vertices

- The output size (the number of vertices) depends on the *size* of the input
- The complexity of our algorithm also depends on the shape of the input
- The complexity of the above method depends on the volume of the input

Benchmarks

Table: Integer hulls of triangles

Volume	27.95		111.79		11179.32	
Algorithm	IntegerHull	Zpoly	IntegerHull	Zpoly	IntegerHull	Zpoly
Time(s)	0.172	0.410	0.244	0.890	0.159	58.083

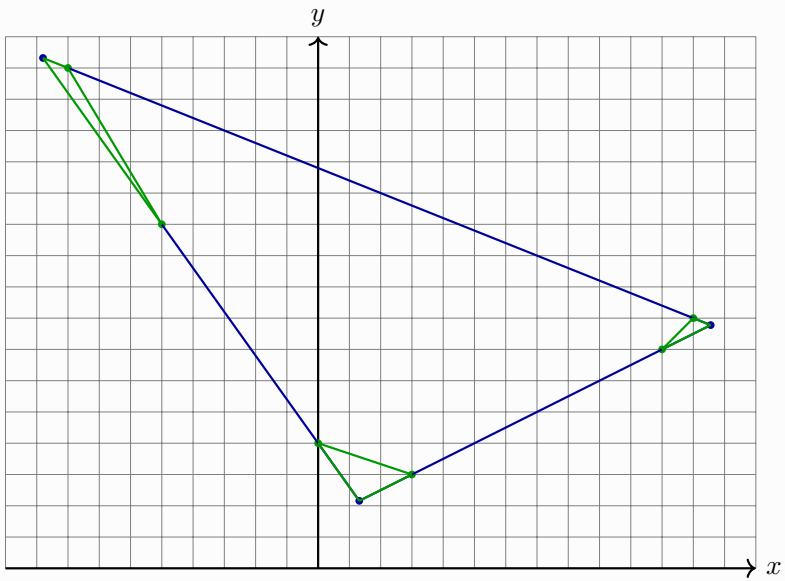
Table: Integer hulls of hexagons

Volume	58.21		5820.95		23283.82	
Algorithm	IntegerHull	Zpoly	IntegerHull	Zpoly	IntegerHull	Zpoly
Time(s)	0.303	0.752	0.275	31.357	0.304	123.159

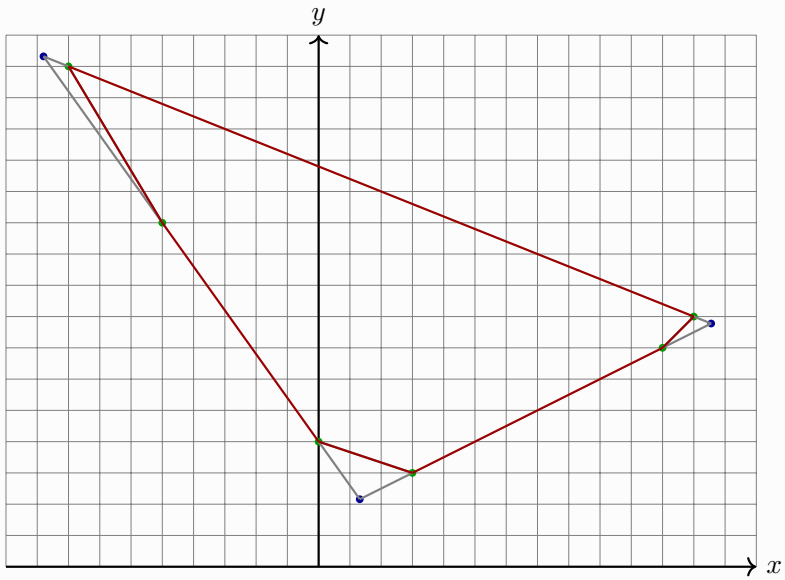
Outline

Learn from the 2D algorithm

- Pre-process
- Find the closest integer points to each vertex on its adjacent facets

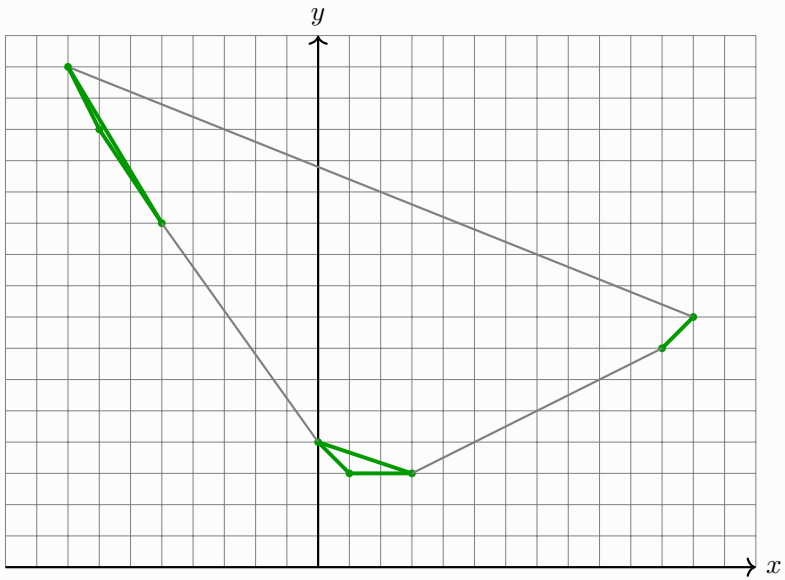


- Cut off the corners
- Center (red) part is already an integer hull

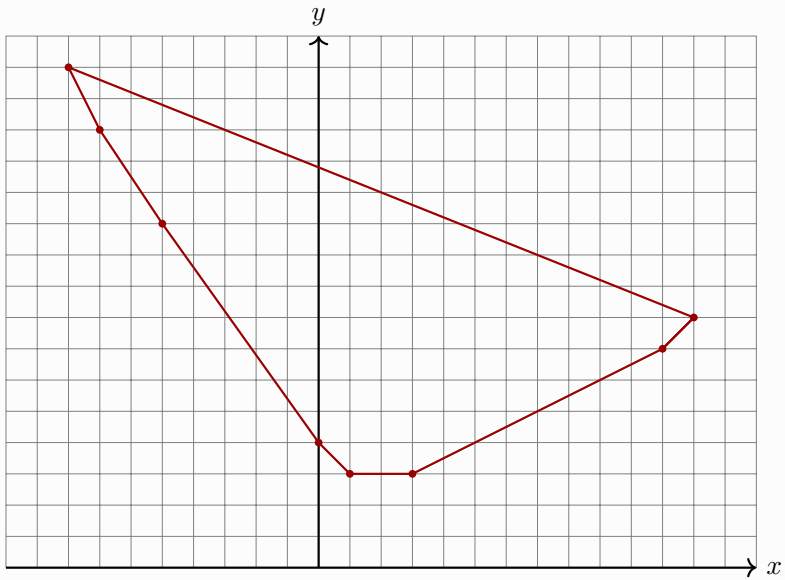


Learn from the 2D algorithm

Compute the integer hull of each corner



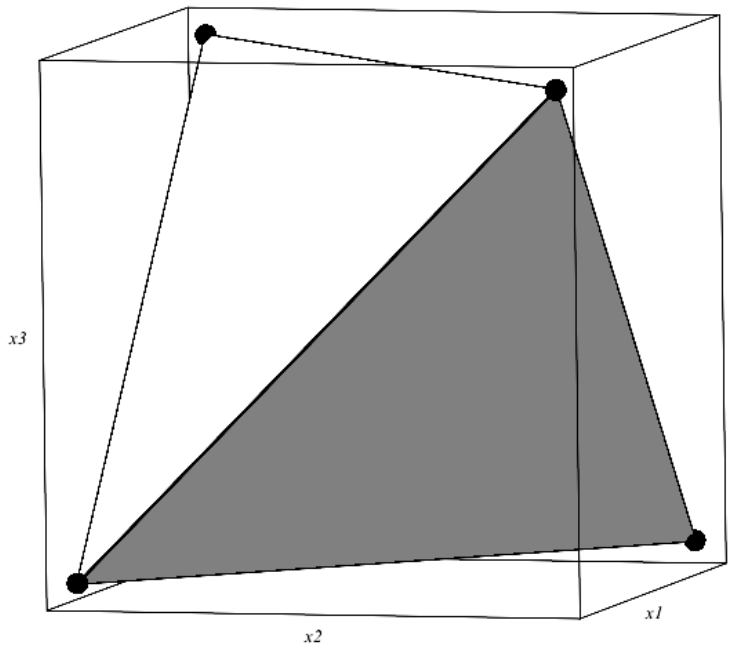
Compute the convex hull of all the integer hulls

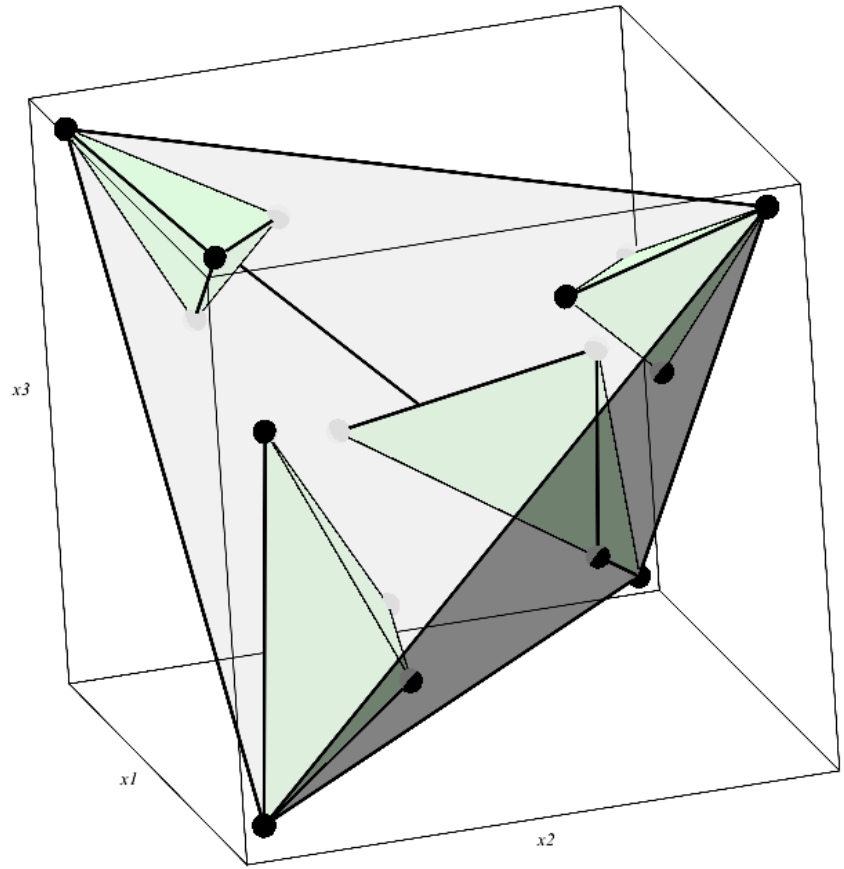


Developing the 3D algorithm

After pre-processing, find the closest integer points to each vertex on its adjacent facets

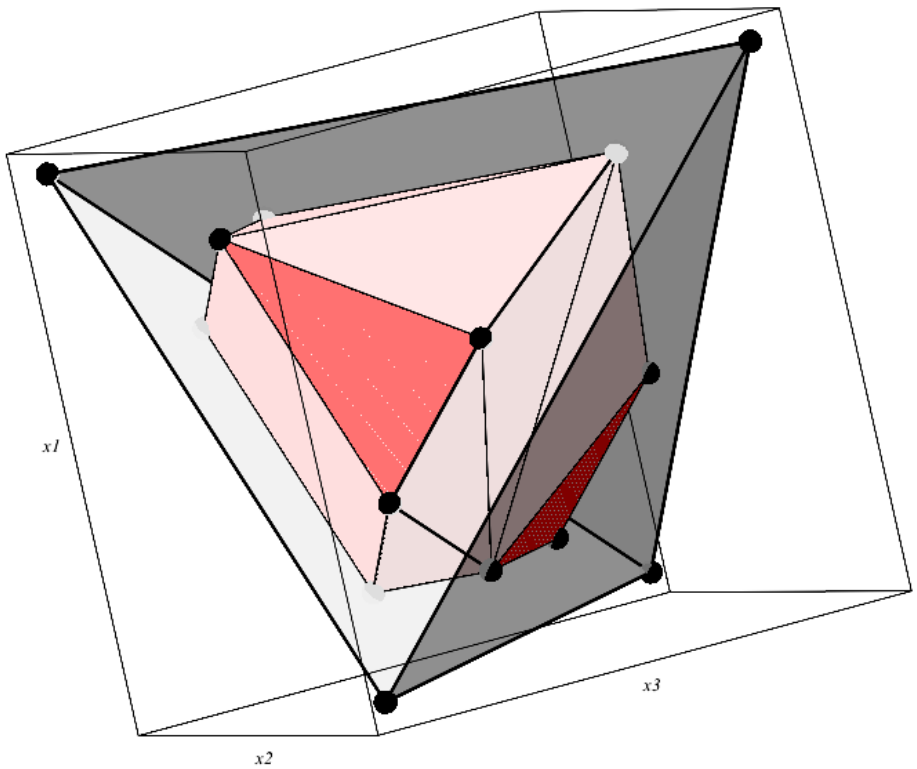
Cut off the corner area near the vertices



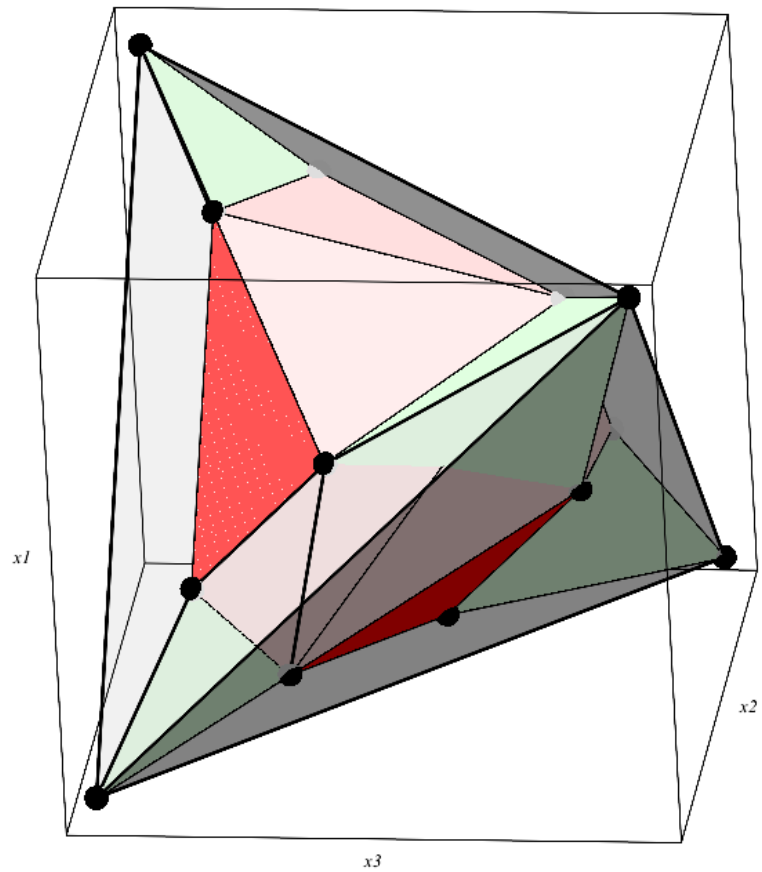


Developing the 3D algorithm

We don't need to work on the center area

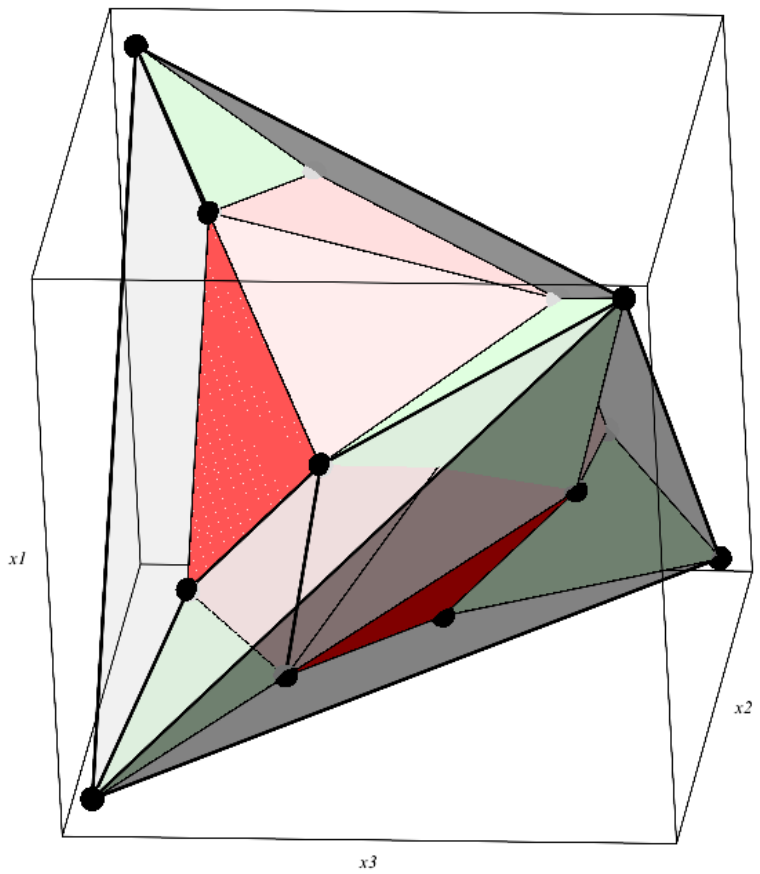


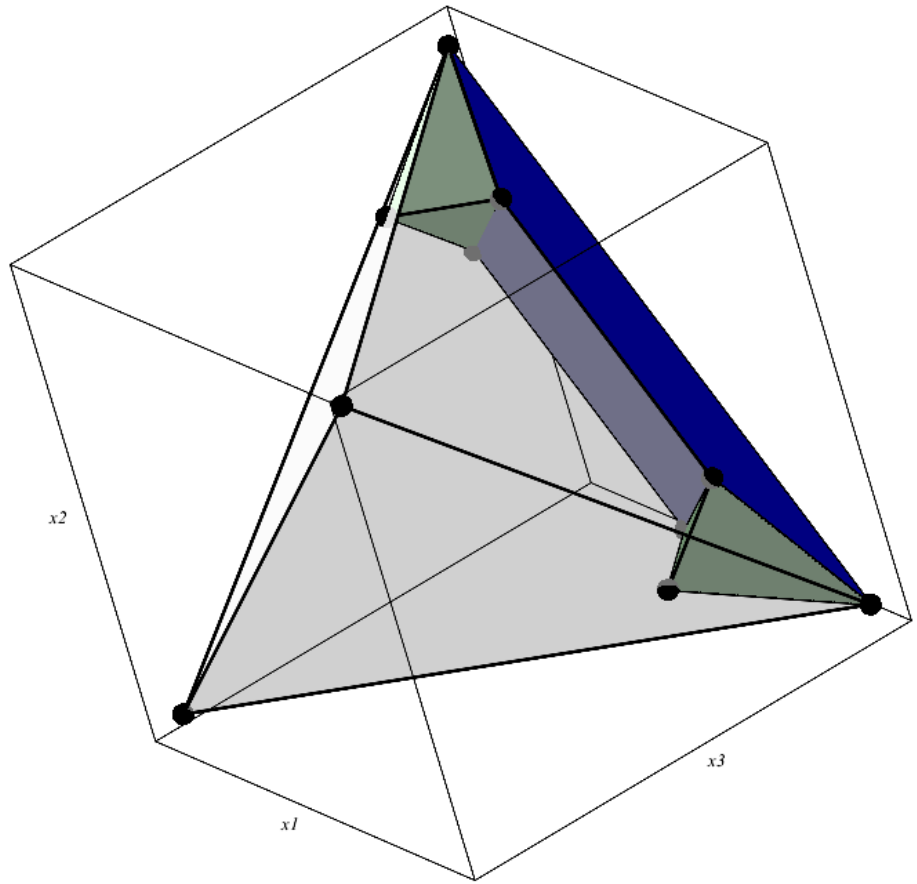
The areas near the edges are not covered



Developing the 3D algorithm

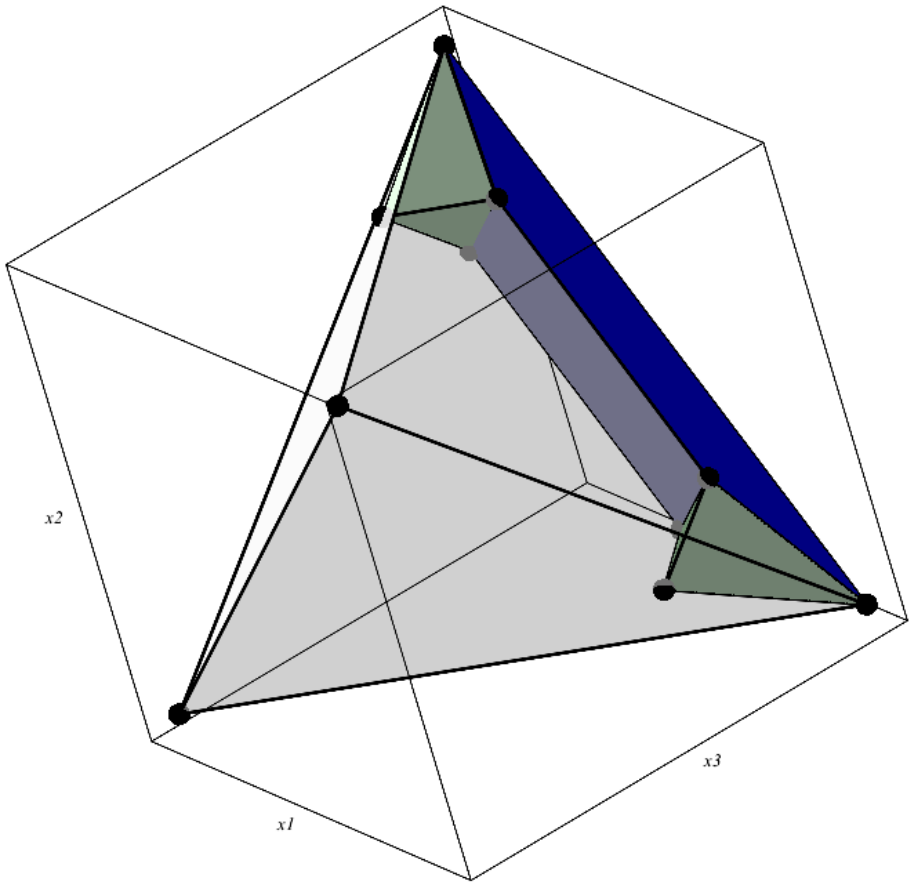
We need to compute the integer hull of the polyhedra formed near the vertices and the edges

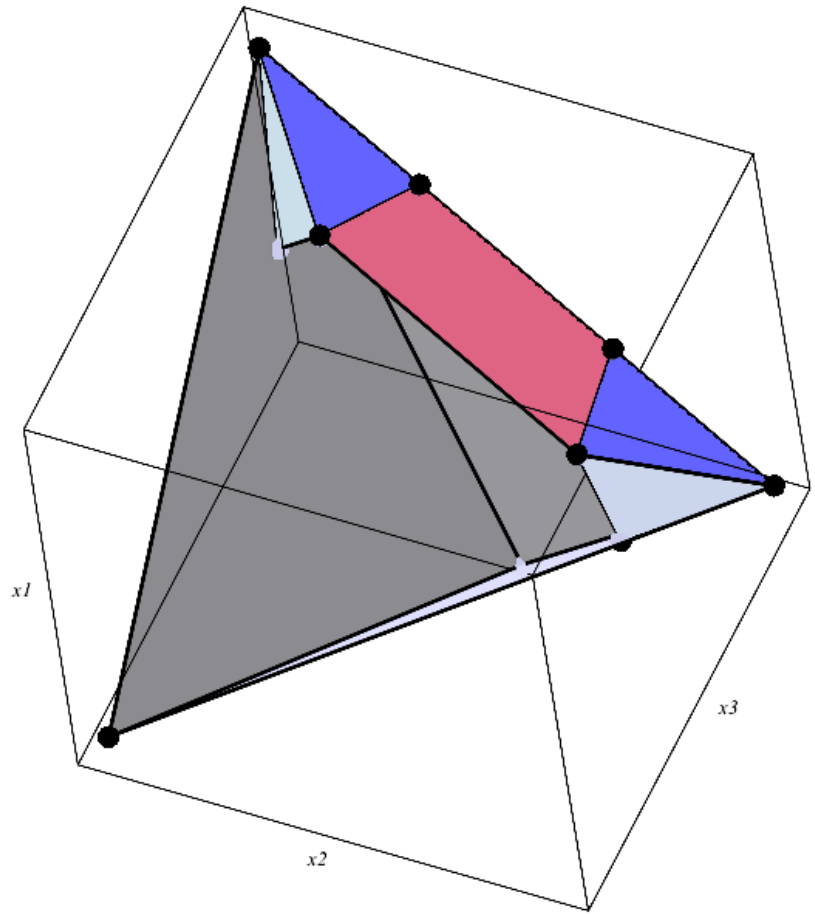




Developing the 3D algorithm

In the worst case, we need to find all the integer points near the edge.
In the best case, we can find the integer points on the edge that is closest to each fractional vertex, and cut down the searched area.





Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Main procedures

- Find the closest integer points to each vertex on its adjacent facets
  - `LinearAlgebra:-HermiteForm`
  - Make a projection of the 3D facets to 2D space
  - Hermite normal form will guarantee that the integer points in the projection would are mapped to integer points in the original space
  - Call `IntegerHull` on the projection which is a 2D polyhedral set to compute the vertices of the integer hull of the facet
  - Compute the vertices in the original space, and find the closest one to the fractional vertex
  - Note that the "closest integer hull vertex" may not be the closest integer point, but good enough that won't affect our efficiency in the later steps.
- Find the closest integer points to each vertex on its adjacent edges if any
- Create the polyhedra near vertices and edges
- Compute the integer hulls of the "cut off" polyhedra
  - `RegularChains:-FMXelim` and `LinearAlgebra:-HermiteForm`
  - Scan for all the integer points
  - `ComputationalGeometry:-ConvexHull`: find the convex hull of integer points
- Use the *Grid* package to compute the expensive steps in parallel

Benchmarks

Table: Integer hulls of tetrahedrons (4 vertices, 4 facets and 6 edges)

Volume	447.48		6991.89		55935.2	
Algorithm	IntegerHull	Zpoly	IntegerHull	Zpoly	IntegerHull	Zpoly
Time(s)	0.977	7.289	1.223	74.804	1.378	531.904

Table: Integer hulls of triangular bipyramids (5 vertices, 6 facets and 9 edges)

Volume	412.58	7050.81	60417.63
Algorithm	IntegerHull	IntegerHull	IntegerHull
Time(s)	1.476	1.573	1.728

Summary

- A new algorithm for computing the integer hulls of polyhedral sets with consideration of the geometric properties of the sets
- Implement the algorithm for 2D and 3D
- CASC 2021 paper about the periodicity of the number of the vertices of integer hulls

Future work

- Arbitrary dimensional algorithms
  - For an N-dimensional polyhedral set we may need to consider the areas near the faces that are of dimension 1 to $N-1$
- Integer hull of parametric polyhedral sets

Thank You!

Your Questions?