# Decomposition and QE algorithms over the reals and over the integers

Marc Moreno Maza

Ontario Research Center for Computer Algebra
Departments of Computer Science and Mathematics
University of Western Ontario, Canada

Special Session in honor of Professor James Davenport
SYNASC 2023, LORIA, France, September 11-13

# Acknowledgements


- Many thanks to the organizers of this workshop for their invitation.
- This talk is based on research projects with some of my former PhD students: Alexander Brandt (Dalhousie University), Changbo Chen (CIGIT Chinese Academy of Sciences), Xiaohui Chen (HUAWEI), Ruijuan Jing (Jiangsu University), Franccois Lemaire (Université de Lille), Wei Pan (NVIDIA, Delaram Talaashrafi (NVIDIA), Linxiao Wang (HUAWEI), Rong Xiao (Amazon), Ning Xie (HUAWEI), Yuzhen Xie (Scotiabank),
- As well as collaborators: James H. Davenport (University of Bath), Matthew England (Coventry University), John May (Maplesoft), Bican Xia (Peking University).
- This talk is also based on collaborations with Maplesoft, MIT/CSAIL, Intel, IBM Canada, Lawrence Livermore National Laboratory with funding support from Maplesoft, IBM Canada, and NSERC of Canada.

# Tentative Plan

1. Decomposition and QE algorithms over the reals
2. Decomposition and QE algorithms over the integers

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract information** (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract information** (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same as above**, or
  - finding **sample solutions**, or
  - performing **cylindrical algebraic decomposition (CAD)** or **quantifier elimination (QE)**.
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier elimination (QE)** (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the same as above, or
  - finding sample solutions, or
  - performing cylindrical algebraic decomposition (CAD) or quantifier elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.

- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).

- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# What does solving mean here?

- **Solving** over $\mathbb{C}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$) and inequations $h \neq 0$:
  - computing **all** its solutions symbolically, or only the **generic** ones
  - providing tools to **extract** information (dimension, degree, etc.) about those solutions and,
  - **performing** (set or geometric) **operations** on solutions sets.
- **Solving** over $\mathbb{R}$: that is, solving any system of multivariate polynomial equations (say, $f_1 = \cdots = f_m = 0$, inequations $h \neq 0$ and inequalities $g_1 > 0, \ldots, g_p > 0$
  - doing the **same** as above, or
  - finding **sample** solutions, or
  - performing **cylindrical** algebraic decomposition (CAD) or **quantifier** elimination (QE).
- **Solving** over $\mathbb{Z}$: focusing on linear inequality systems, can mean:
  - **counting** the number of solutions, or
  - **computing** all or part of the **solutions**, or
  - performing **quantifier** elimination (QE) (Presburger Arithmetic).

# Outline

# Outline

# Solving for the real solutions of polynomial systems

## Classical tools as of 2010

- Cylindrical algebraic decomposition of polynomial systems:
  `SemiAlgebraicSetTools:-CylindricalAlgebraicDecompose` (James)
- Real root classification of parametric polynomial systems:
  `ParametricSystemTools:-RealRootClassification` (Bican)
- Decomposing polynomial systems over the algebraic closure of the base field:
  `RegularChains:-Triangularize` (ORCCA)

## New tools in the RegularChains library 2011

- Triangular decomposition of semi-algebraic systems: `RealTriangularize`
- Sampling all connected components of a semi-algebraic system: `SamplePoints`
- Set-theoretical operations on semi-algebraic sets:
  `SemiAlgebraicSetTools:-Difference`

# Solving for the real solutions of polynomial systems

## Classical tools as of 2010

- Cylindrical algebraic decomposition of polynomial systems:
  `SemiAlgebraicSetTools:-CylindricalAlgebraicDecompose` (James)
- Real root classification of parametric polynomial systems:
  `ParametricSystemTools:-RealRootClassification` (Bican)
- Decomposing polynomial systems over the algebraic closure of the base field:
  `RegularChains:-Triangularize` (ORCCA)

## New tools in the RegularChains library 2011

- Triangular decomposition of semi-algebraic systems: `RealTriangularize`
- Sampling all connected components of a semi-algebraic system: `SamplePoints`
- Set-theoretical operations on semi-algebraic sets:
  `SemiAlgebraicSetTools:-Difference`

# Regular semi-algebraic system

## Notation

- Let $T \subset \mathbb{Q}[x_1 < \ldots < x_n]$ be a regular chain with $\mathbf{y} := \{\mathrm{mvar}(t) \mid t \in T\}$ and $\mathbf{u} := \mathbf{x} \setminus \mathbf{y} = u_1, \ldots, u_d$.
- Let $P$ be a finite set of polynomials, s.t. every $f \in P$ is regular modulo $\mathrm{sat}(T)$.
- Let $\mathcal{Q}$ be a quantifier-free formula of $\mathbb{Q}[\mathbf{u}]$.

## Definition

We say that $R := [\mathcal{Q}, T, P_>]$ is a regular semi-algebraic system if:

$(i)$ $\mathcal{Q}$ defines a non-empty open semi-algebraic set $S$ in $\mathbb{R}^d$,

$(ii)$ the regular system $[T, P]$ specializes well at every point $u$ of $S$

$(iii)$ at each point $u$ of $S$, the specialized system $[T(u), P(u)_>]$ has at least one real solution.

$$Z_{\mathbb{R}}(R) = \{(u, y) \mid \mathcal{Q}(u), t(u, y) = 0, p(u, y) > 0, \forall (t, p) \in T \times P\}.$$

# Regular semi-algebraic system

## Notation

- Let $T \subset \mathbb{Q}[x_1 < \ldots < x_n]$ be a regular chain with $\mathbf{y} := \{\mathrm{mvar}(t) \mid t \in T\}$ and $\mathbf{u} := \mathbf{x} \setminus \mathbf{y} = u_1, \ldots, u_d$.
- Let $P$ be a finite set of polynomials, s.t. every $f \in P$ is regular modulo $\mathrm{sat}(T)$.
- Let $\mathcal{Q}$ be a quantifier-free formula of $\mathbb{Q}[\mathbf{u}]$.

## Definition

We say that $R := [\mathcal{Q}, T, P_>]$ is a regular semi-algebraic system if:

$(i)$ $\mathcal{Q}$ defines a non-empty open semi-algebraic set $S$ in $\mathbb{R}^d$,

$(ii)$ the regular system $[T, P]$ specializes well at every point $u$ of $S$

$(iii)$ at each point $u$ of $S$, the specialized system $[T(u), P(u)_>]$ has at least one real solution.

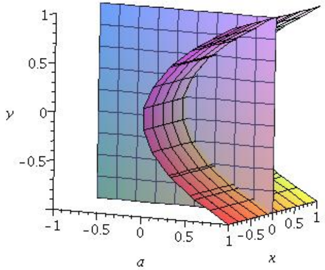
$$Z_{\mathbb{R}}(R) = \{(u, y) \mid \mathcal{Q}(u), t(u, y) = 0, p(u, y) > 0, \forall (t, p) \in T \times P\}.$$

## Example

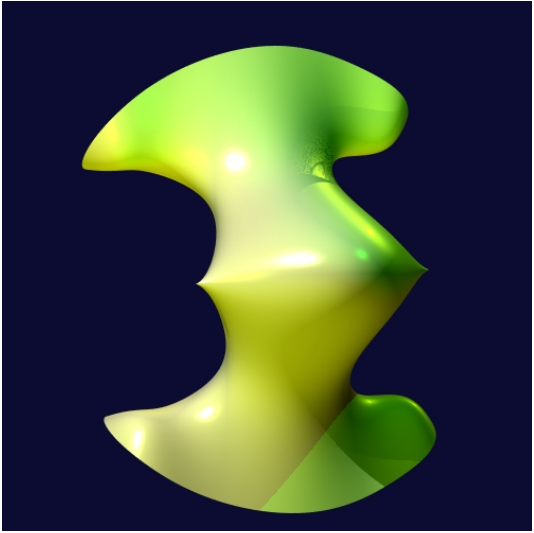
The system $[\mathcal{Q}, T, P_>]$, where

$$\mathcal{Q} := a > 0,\ T := \begin{cases} y^2 - a = 0 \\ x = 0 \end{cases},\ P_> := \{y > 0\}$$

is a regular semi-algebraic system.



# RealTriangularize applied to the Eve surface (1/2)



# RealTriangularize applied to the Eve surface (2/2)

$R := PolynomialRing([x, y, z]);\ F := [5*x^2 + 2*x*z^2 + 5*y^6 + 15*y^4 + 5*z^2 - 15*y^5 - 5*y^3];$

$$polynomial\_ring$$

$$[5x^2 + 2xz^2 + 5y^6 + 15y^4 + 5z^2 - 15y^5 - 5y^3]$$

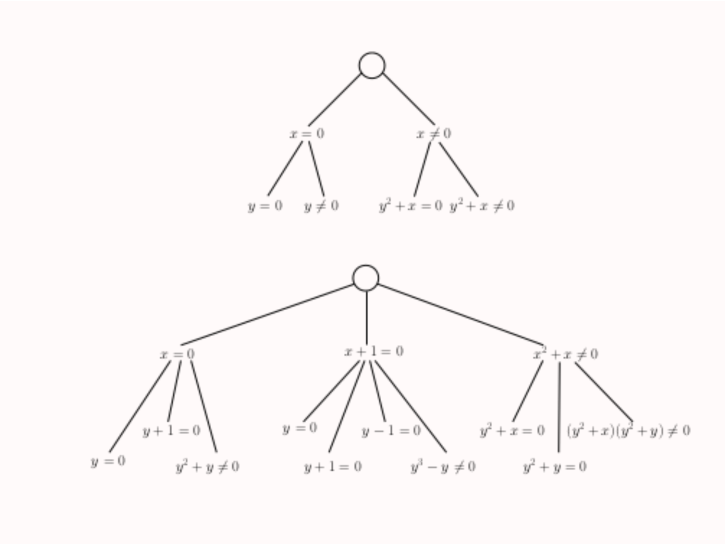
$RealTriangularize(F, R, output = record);$

$$\left\{\begin{array}{l} 5x^2 + 2z^2x + 5y^6 + 15y^4 - 5y^3 - 15y^5 + 5z^2 = 0 \\ 25y^6 - 75y^5 + 75y^4 - z^4 - 25y^3 + 25z^2 < 0 \end{array}\right.,$$

$$\left\{\begin{array}{c} 5x + z^2 = 0 \\ 25y^6 - 75y^5 + 75y^4 - 25y^3 - z^4 + 25z^2 = 0 \\ 64z^4 - 1600z^2 + 25 > 0 \\ z \neq 0 \\ z - 5 \neq 0 \\ z + 5 \neq 0 \end{array}\right., \left\{\begin{array}{c} x = 0 \\ y - 1 = 0 \\ z = 0 \end{array}\right., \left\{\begin{array}{c} x = 0 \\ y = 0 \\ z = 0 \end{array}\right., \left\{\begin{array}{c} x + 5 = 0 \\ y - 1 = 0 \\ z - 5 = 0 \end{array}\right.,$$

$$\left\{\begin{array}{c} x + 5 = 0 \\ y = 0 \\ z - 5 = 0 \end{array}\right., \left\{\begin{array}{c} x + 5 = 0 \\ y - 1 = 0 \\ z + 5 = 0 \end{array}\right., \left\{\begin{array}{c} x + 5 = 0 \\ y = 0 \\ z + 5 = 0 \end{array}\right., \left\{\begin{array}{c} 5x + z^2 = 0 \\ 2y - 1 = 0 \\ 64z^4 - 1600z^2 + 25 = 0 \end{array}\right.$$

# Outline




- A CAD of $\{y^2 + x, y^2 + y\}$ is computed **incrementally**: refining a CAD tree of $y^2 + x$ with $y^2 + y$.
- Experimental results in [5] (ASCM 2012) suggest that this approach outperforms the projection-and-lifting scheme of [7] (ISSAC 2009).

# Outline

```
> phi1 := ( ( 74 <= x ) &and ( x <= 76 ) &and ( v = 0 )
  &implies ( -v^2 - a * (x-75)^2 + b >= 0 ) ):

> phi2 := ( ( -v^2 - a * (x-75)^2 + b >= 0 )
  &implies (( 80 >= x ) &and ( x >= 70 )) ):

> phi3 := ( ( -v^2 - a * (x-75)^2 + b = 0 )
  &implies (( -2*v - a * 2 * (x-75)* v >= 0 ) &or ( 2*v - a
  * 2 * (x-75)* v >= 0 )) ):

> phi := phi1 &and phi2 &and phi3:
> t0 := time():
  psi := QuantifierElimination(&A([x,v]),phi,output=rootof);
  t1 := time() - t0;
```

$$\psi := ((0 < a \ \&\text{and}\ a \leq 1) \ \&\text{and}\ a \leq b) \ \&\text{and}\ b \leq \min\left(\frac{1}{a}, 25\, a\right)$$

$$t1 := 15.094$$

- **QE** based on regular chains and incremental CAD [6] (presented by James for us at ISSAC 2014) is illustrated above.
- This QE problem instance is related to a verification and synthesis of switched and hybrid dynamical systems (Sturm-Tiwari, ISSAC 2011).

# Outline

# Outline

# Dependence analysis

Cholesky's LU decomposition:

```
1:  for(i = 1; i <= n; i++){
        x = a[i][i];
        for(k = 1; k < i; k++)
2:          x = x - a[i][k] * a[i][k];
3:      p[i] = 1.0/sqrt(x);
        for(j = i+1; j <= n; j++){
4:          x = a[i][j];
            for(k = 1; k < i; k++)
5:              x = x - a[j][k] * a[i][k];
6:          a[j][i] = x * p[i];
        }
    }
```

system 1:

$$\begin{cases} 1 \le i \le n \\ i+1 \le j \le n \\ 1 \le k \le i-1 \\ 1 \le i' \le n \\ i'+1 \le j' \le n \\ j = j', k = i' \\ i < i' \end{cases}$$

system 2:

$$\begin{cases} 1 \le i \le n \\ i+1 \le j \le n \\ 1 \le k \le i-1 \\ 1 \le i' \le n \\ i'+1 \le j' \le n \\ j = j', k = i' \\ i = i', j < j' \end{cases}$$

system 3:

$$\begin{cases} 1 \le i \le n \\ i+1 \le j \le n \\ 1 \le k \le i-1 \\ 1 \le i' \le n \\ i'+1 \le j' \le n \\ j = j', k = i' \\ i = i', j = j' \end{cases}$$

# Dependence analysis

Cholesky's LU decomposition:

```
1:  for(i = 1; i <= n; i++){
        x = a[i][i];
        for(k = 1; k < i; k++)
2:          x = x - a[i][k] * a[i][k];
3:      p[i] = 1.0/sqrt(x);
        for(j = i+1; j <= n; j++){
4:          x = a[i][j];
            for(k = 1; k < i; k++)
5:              x = x - a[j][k] * a[i][k];
6:          a[j][i] = x * p[i];
        }
    }
```

system 1:

$$\begin{cases} 1 \leq i \leq n \\ i+1 \leq j \leq n \\ 1 \leq k \leq i-1 \\ 1 \leq i' \leq n \\ i'+1 \leq j' \leq n \\ j = j', k = i' \\ i < i' \end{cases}$$

system 2:

$$\begin{cases} 1 \leq i \leq n \\ i+1 \leq j \leq n \\ 1 \leq k \leq i-1 \\ 1 \leq i' \leq n \\ i'+1 \leq j' \leq n \\ j = j', k = i' \\ i = i', j < j' \end{cases}$$

system 3:

$$\begin{cases} 1 \leq i \leq n \\ i+1 \leq j \leq n \\ 1 \leq k \leq i-1 \\ 1 \leq i' \leq n \\ i'+1 \leq j' \leq n \\ j = j', k = i' \\ i = i', j = j' \end{cases}$$

# Outline

# Delinearization

## Linearized multi-dimensional array

```
for (int i = 0; i < n; i ++)
    for (int j = i + 1; j < n; j ++)
        A[i * n + j] =
                A[n * n - n + j - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 * n + j_1 = n^2 - n + j_2 - 1 \end{cases} \tag{1}$$

## Delinearized multi-dimensional array

```
for (int i = 0; i < n; i ++)
    for (int j = i + 1; j < n; j ++)
        A[i][j] = A[n - 1][j - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 = n - 1 \\ j_1 = j_2 - 1 \end{cases} \tag{2}$$

# Problem definition

**Input**:

$(\mathbf{i}_1 \cdots ; \ \cdots ; \ i_1{+}{+})$
$\ldots (i_d \cdots ; \cdots ; \ i_d{+}{+})$
$A[R(i_1, \ldots, i_d, m_1, \ldots, m_\delta)] \leftarrow \cdots \ \ldots$

- $i_1, \ldots, i_d$ take non-negative integer values such that
$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix},$$
- L is a lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_\delta, r_1, \ldots, r_d$: data parameters (known only at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_\delta)$ is a polynomial, the coefficients of which are known at compile time.

**Output**:

$(i_1 \cdots ; \ \cdots ; \ i_1{+}{+})$
$\ldots (i_d \cdots ; \cdots ; \ i_d{+}{+})$
$\widetilde{A}[f_1] \cdots [f_\delta] \leftarrow \cdots \ \ldots$

- $f_1, \ldots, f_\delta$ are affine forms in $i_1, \ldots, i_d$ the coefficients of which are integers to-be-determined,
- $\widetilde{A}$ is an $M_1 \times \cdots \times M_\delta$-array,
- $M_1, \ldots, M_\delta$ are affine forms in $m_1, \ldots, m_\delta$ the coefficients of which are integers TBD,

such that:
$$R = f_1 M_2 \cdots M_\delta \ + \ \cdots \ + \ f_{\delta-1} M_2 + f_\delta$$
holds and for each $(i_1, \ldots, i_d)$ in the iteration domain we have:
$$0 \leq f_1 < M_1, \ \ \ldots, 0 \leq f_\delta < M_\delta.$$

# Problem definition

**Input**:

$(\mathbf{i}_1 \cdots ;\ \cdots ;\ i_1{+}{+})$
$\ldots(i_d \cdots ; \cdots ;\ i_d{+}{+})$
$A[R(i_1, \ldots, i_d, m_1, \ldots, m_\delta)] \leftarrow \cdots\ \ldots$

- $i_1, \ldots, i_d$ take non-negative integer values such that
$$L \begin{pmatrix} i_1 \\ \vdots \\ i_d \end{pmatrix} \leq \begin{pmatrix} r_1 \\ \vdots \\ r_d \end{pmatrix},$$
- L is a lower-triangular full-rank matrix over $\mathbb{Z}$ (known at compile time) defining the iteration domain
- $m_1, \ldots, m_\delta, r_1, \ldots, r_d$: data parameters (known only at execution time)
- $R(i_1, \ldots, i_d, m_1, \ldots, m_\delta)$ is a polynomial, the coefficients of which are known at compile time.

**Output**:

$(i_1 \cdots ;\ \cdots ;\ i_1{+}{+})$
$\ldots(i_d \cdots ; \cdots ;\ i_d{+}{+})$
$\widetilde{A}[f_1] \cdots [f_\delta] \leftarrow \cdots\ \ldots$

- $f_1, \ldots, f_\delta$ are affine forms in $i_1, \ldots, i_d$ the coefficients of which are integers to-be-determined,
- $\widetilde{A}$ is an $M_1 \times \cdots \times M_\delta$-array,
- $M_1, \ldots, M_\delta$ are affine forms in $m_1, \ldots, m_\delta$ the coefficients of which are integers TBD,

such that:
$$\boxed{R = f_1 M_2 \cdots M_\delta\ +\ \cdots\ +\ f_{\delta-1} M_2 + f_\delta}$$
holds and for each $(i_1, \ldots, i_d)$ in the iteration domain we have:
$$\boxed{0 \leq f_1 < M_1,\ \ \ldots, 0 \leq f_\delta < M_\delta.}$$

# Two problems to solve

## Polynomial system solving

Find $f_1, \ldots, f_\delta$ so that

$$R = f_1 M_2 \cdots M_\delta + \cdots + f_{\delta-1} M_2 + f_\delta$$

holds.

- This part can be done off-line.

## Quantifier elimination

$\forall(i_1, \ldots, i_d)$ in the iteration domain, we have:

$$0 \leq f_1 < M_1, \ldots, 0 \leq f_\delta < M_\delta$$

- At run-time, all the parameters are known, we can solve this problem in the integer domain.
- But we would rather do it off-line (thus parametrically).

# Integer QE problem

**For each $f_k$ and $M_k$, we need to ensure $\max f_k < M_k$**

$$\begin{array}{ll} \text{maximize} & f_k \\ \text{subject to} & i_1, \ldots, i_d \in \mathbb{Z} \\ & \forall (i_1, \ldots, i_d) \ \in \mathbf{D} \end{array}$$

- At compile time, $f_k$ and $M_k$ cannot be determined numerically because of the parameters.
- Thus, the above problem becomes a **parametric** integer linear programming problem (PILP) which is very similar to a **parametric** integer hull problem.
- This has motivated what follows.

# Outline

# A for-loop nest and its associated parametric polyhedral set

```
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 $n$)

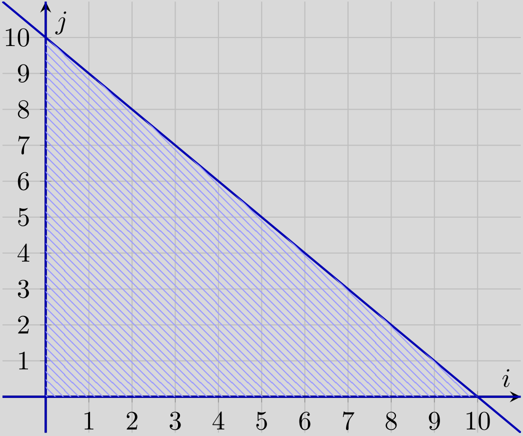
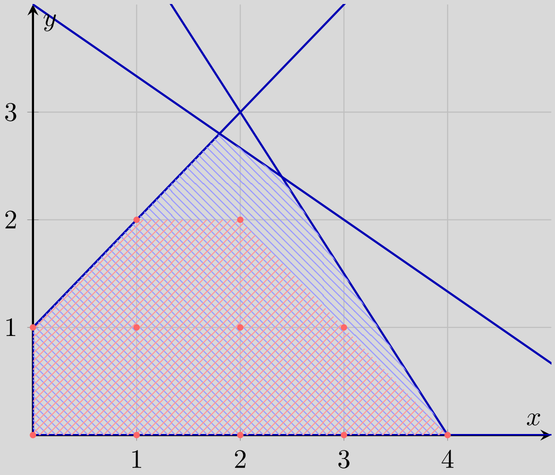



Figure: Iteration space when $n = 10$

# Integer hull: simple non-parametric 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

▸▸ skip slide

# Outline

# Decomposing the integer points of a polyhedron

## Example

Input: $K_1 : \begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ -x_2 \leq -25 \end{cases}$, assume $x_1 > x_2 > x_3$.

Output: $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ given by:

$$\begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ 2x_2 - x_3 \leq 48 \\ -5x_2 + 13x_3 \leq 67 \\ -x_2 \leq -25 \\ 2 \leq x_3 \leq 17 \end{cases}, \begin{cases} x_1 = 15 \\ x_2 = 27 \\ x_3 = 16 \end{cases}, \begin{cases} x_1 = 18 \\ x_2 = 33 \\ x_3 = 18 \end{cases}, \begin{cases} x_1 = 14 \\ x_2 = 25 \\ x_3 = 15 \end{cases}, \begin{cases} x_1 = 19 \\ x_2 = 50 + t \\ x_3 = 50 + 2t \\ -25 \leq t \leq -16. \end{cases}$$

# Decomposing the integer points of a polyhedron

Output: $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ given by:

$$\begin{cases} 3x_1 - 2x_2 + x_3 \le 7 \\ -2x_1 + 2x_2 - x_3 \le 12 \\ -4x_1 + x_2 + 3x_3 \le 15 \\ 2x_2 - x_3 \le 48 \\ -5x_2 + 13x_3 \le 67 \\ -x_2 \le -25 \\ 2 \le x_3 \le 17 \end{cases}, \begin{cases} x_1 = 15 \\ x_2 = 27 \\ x_3 = 16 \end{cases}, \begin{cases} x_1 = 18 \\ x_2 = 33 \\ x_3 = 18 \end{cases}, \begin{cases} x_1 = 14 \\ x_2 = 25 \\ x_3 = 15 \end{cases}, \begin{cases} x_1 = 19 \\ x_2 = 50 + t \\ x_3 = 50 + 2t \\ -25 \le t \le -16. \end{cases}$$

- An integer point solves $K_1$ iff it solves either $K_1^1$, $K_1^2$, $K_1^3$, $K_1^4$ or $K_1^5$.
- Each of $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ has at least one integer point.
- For each $K_1^i$, each integer point in any (standard) projection of $K_1^i$ can be lifted to an integer point in the polyhedron.

# Outline

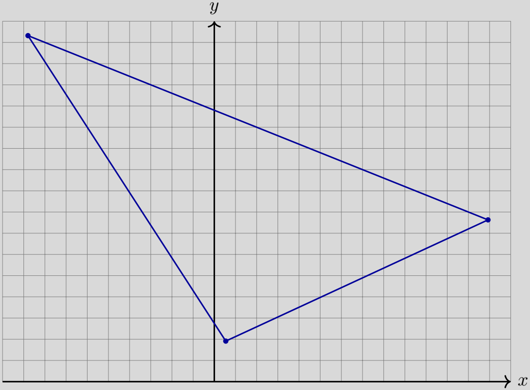
# Example (0/3)

## Input

Let's look at a simple example first.

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 (1/3)

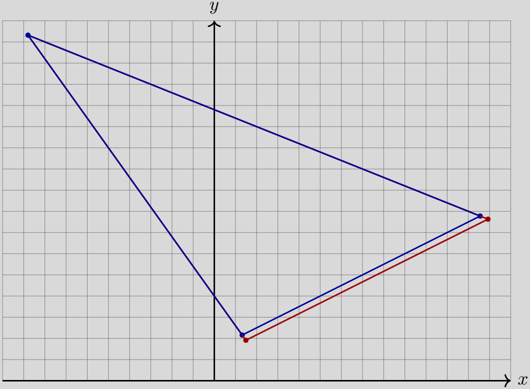
## Normalization

Replace the facets that could not have integer point

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

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

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

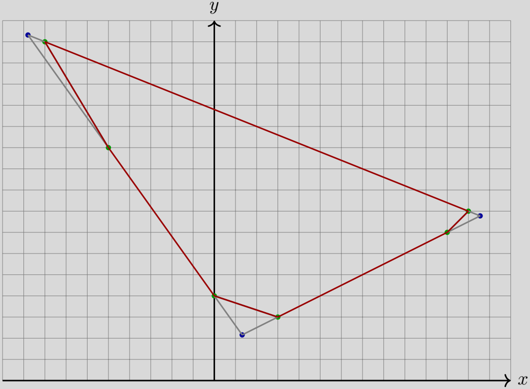


# Example (2/3)

## Partition

Vertices: $(-44/5, 408/25), (113/9, 70/9), (25/19, 41/19)$
Find the triangles with vertices: $[(-8, 16), (-44/5, 408/25), (-5, 11)]$, $[(3, 3), (25/19, 41/19), (0, 4)]$, $[(12, 8), (113/9, 70/9), (11, 7)]$

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



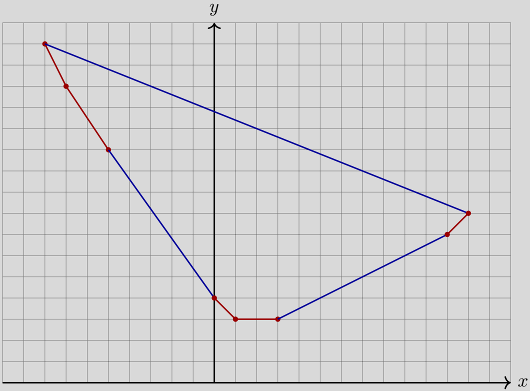
# Example (3/3)

## Merging

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

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

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



# Main steps of our algorithm

Our algorithm has 3 main steps:

- **Normalization**: construct a new polyhedral set $Q$ from $P$ as follows. Consider in turn each facet $F$ of $P$:
  1. if the hyperplane $H$ supporting $F$ contains an integer point, then $H$ is a hyperplane supporting a facet of $Q$,
  2. otherwise we slide $H$ towards the center of $P$ along the normal vector of $F$, stopping as soon as we hit a hyperplane $H'$ containing an integer point, then making $H'$ a hyperplane supporting a facet of $Q$.

  Clearly $Q_I = P_I$.
- **Partitioning**: make each part of the partition a polyhedron $R$ which:
  1. either has integer points as vertices so that $R_I = R$,
  2. or has a small volume so that any algorithm (including exhaustive search) can be applied to compute $R_I$.
- **Merging**: Once the integer hull of each part of the partition is computed and given by the list of its vertices, an algorithm for computing the convex hull of a set points, such as `QuickHull`, can be applied to deduce $P_I$.

skip slide

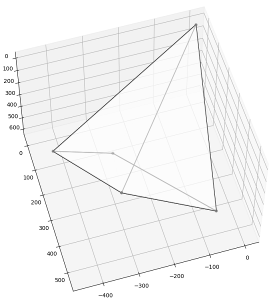
# The general algorithm on a 3D example

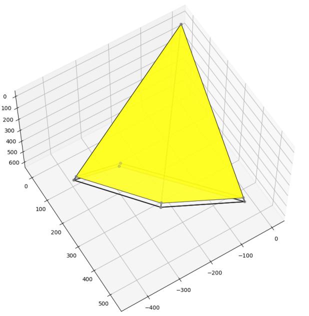
## Normalization

The integer hull of the normalized polyhedral set should be the same as that of the input

$$\left\{\begin{array}{rcl} -98877x_1 - 189663x_2 - 1798x_3 & \leq & 705915 \\ -10109x_1 - 5958x_2 - 14601x_3 & \leq & 31333 \\ -5405x_1 + 4965x_2 + 3870x_3 & \leq & 4303504 \\ 729x_1 - 117x_2 + 350x_3 & \leq & 4561 \\ 677x_1 + 465x_2 - 540x_3 & \leq & 3489 \end{array}\right.$$

$$\left\{\begin{array}{rcl} -98877x_1 - 189663x_2 - 1798x_3 & \leq & 705915 \\ -10109x_1 - 5958x_2 - 14601x_3 & \leq & 31333 \\ -1081x_1 + 993x_2 + 774x_3 & \leq & 860700 \\ 729x_1 - 117x_2 + 350x_3 & \leq & 4561 \\ 677x_1 + 465x_2 - 540x_3 & \leq & 3489 \end{array}\right.$$



# The general algorithm: building the partition

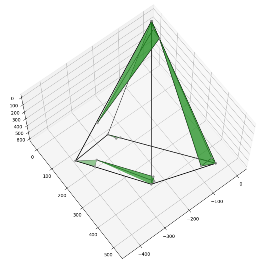
## Partition

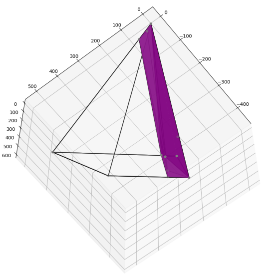
For each face $f$ of $P$:

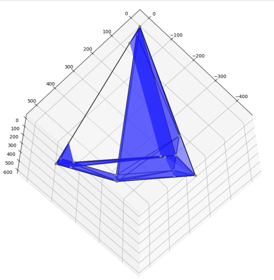
- let $\mathcal{F}$ be the set of all facets that intersect at $f$
- if there exist integer points on $f$ (which implies that the closest integer points on f to each of its vertices do exist as well), then for each vertex $v$ of $f$, a “corner” polyhedral is built as the convex hull of:
  - $v$,
  - the closest integer point to $v$ on $f$,
  - all the closest integer points to $v$ on $F$, for $F \in \mathcal{F}$.
- if there is no integer point on $f$, a single “corner” polyhedral set is built for $f$ as the convex hull of:
  - the vertex set of $f$,
  - all the closest integer points to $v$ on $F$, for $F \in \mathcal{F}$.

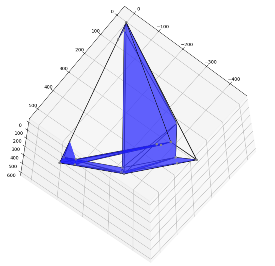
# The general algorithm on a 3D example

## Partition



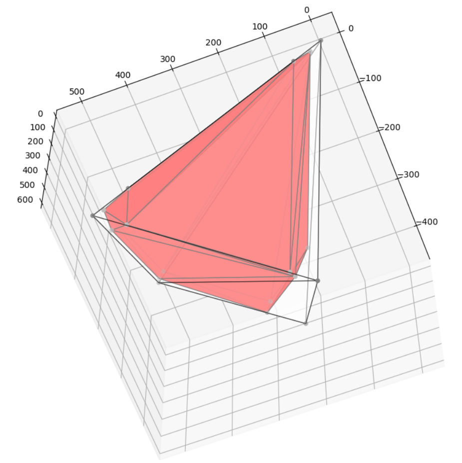



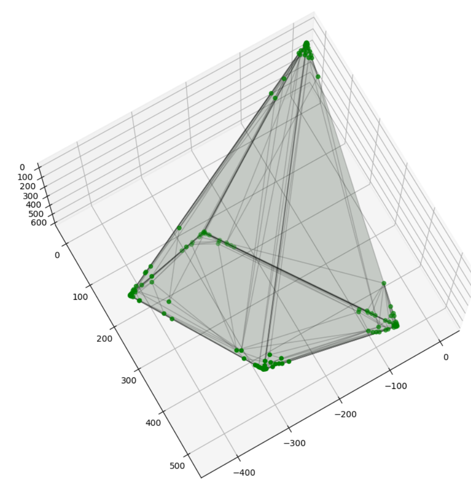



# The general algorithm on a 3D example

## Merging

The integer hull has 139 vertices



# "Closest integer points" on a facet to each of its vertices

## Projection and recursive call

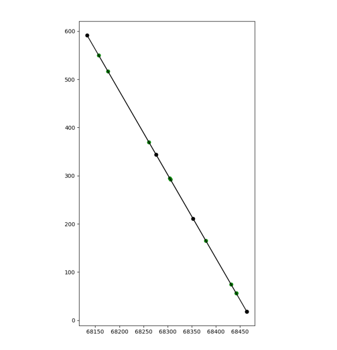
In $\mathbb{Q}^d$, for a facet $F$ of dimension $d-1 < d$, and its vertex set $V$:

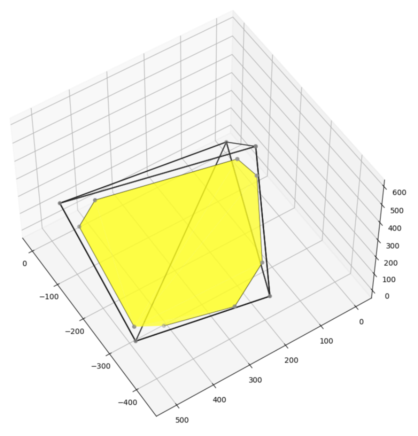
1. make a projection on a full-dimensional polyhedron $G$ using Hermite normal form $\vec{c}^t U = [\mathbf{0} H]$ (where $U = [U_L U_R]$ and $\vec{c}^t \mathbf{x} = s$ is the hyperplane supporting $F$)
2. we obtain a parametrization $R_F$ of $F$ of the form:
$$R_F : \begin{cases} \mathbb{Q}^{d-1} & \rightarrow & \mathbb{Q}^d \\ \mathbf{z} & \longmapsto & \mathbf{x} = \mathbf{v} + U_L \mathbf{z}. \end{cases} \quad (3)$$
3. thus $R_F(G) = F$. Moreover, we have
$$R_F(G_I) = F_I.$$
4. q recursive call to our integer hull algorithm computes the vertices $V_I'$ of the integer hull of $G$
5. we deduce the vertices $V_I$ of $F_I$ by $R_F(V_I') = V_I$
6. finally, we find in $V_I$ the "closest integer points" to each $v$ of $V$.

# Closest integer points on a face to one of its vertices

## Projection and recursive call

$$R_F : \begin{cases} x_1 = 993x'_1 + 573x'_2 - 67995300 \\ x_2 = 1081x'_1 + 623x'_2 - 74020200 \\ x_3 = x'_2 \end{cases}$$



# The PolyhedralSets:-IntegerHull command in Maple

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

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

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

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

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

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

```
>
```

# The PolyhedralSets:-IntegerHull command in Maple

```
> restart; with(PolyhedralSets) :
> vertices := [[10, 10, 10, 10/3], [-140/8, -220/12, -10, -10/3], [60/8, 20, -100/12, -70/3], [-10
     /4, -100/12, 70/2, 35/3], [0, 0, 0, 50/3]] :
  vars := [x1, x2, x3, x4] :
  poly := PolyhedralSet(vertices, [ ], vars);
```

$$poly := \begin{cases} Coordinates & : [x1, x2, x3, x4] \\ Relations & : \left[ -x1 + \frac{503\,x2}{694} + \frac{85\,x3}{694} + \frac{311\,x4}{2082} \leq \frac{7775}{3123}, -x1 + \frac{2715\,x2}{2234} + \frac{603\,x3}{2234} + \cdots \right. \end{cases}$$

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

[[[-15, -16, -6, -2], [-15, -15, -9, -4], [-14, -15, -4, -1], [-13, -13, -8, -1], (1
[-13, -12, -9, -5], [-12, -13, -4, 1], [-12, -13, -4, 2], [-12, -12, -3, -3], [-11, -12, -3, -1], [-11, -11, -6, -3], [-11, -11, -1, -3], [-10, -8, -8, -5], [-9, -6, -8, -8], [-7, -7, -4, 7], [-7, -6, -5, 3], [-7, -3, -8, -10], [-7, -3, -7, -10], [-6, -4, -5, 0], [-5, -9, 23, 8], [-5, -4, -4, 5], [-5, -4, -3, -2], [-4, -5, 3, 10], [-4, -5, 3, 11], [-4, -3, -2, -3], [-4, 0, -6, -9], [-3, -8, 30, 10], [-3, -7, 23, 10], [-3, -7, 23, 11], [-3, -6, 24, 6], [-3, 3, -8, -12], [-3, 3, -7, -13], [-2, -7, 31, 11], [-2, -6, 24, 8], [-2, -6, 24, 12], [-2, -6, 26, 11], [-2, -5, 25, 7], [-2, -5, 26, 6],

# The PolyhedralSets:-IntegerHull command in Maple

```
> ineqs := [-x1-(132*x2)/205-(62*x3)/205 <= -1358/123, -x1 + (34*x2)/34 + (4*x3)/4
      <= 1405/17, x1-(12*x2)/118 + (83*x3)/177 <= 3500/59]:
  poly := PolyhedralSet(ineqs, [x1, x2, x3]);
  IsBounded(poly);
```

$$poly := \begin{cases} \text{Coordinates} & : [x1, x2, x3] \\ \text{Relations} & : \left[ -x1 - \frac{132\,x2}{205} - \frac{62\,x3}{205} \leq -\frac{1358}{123},\ -x1 + x2 + x3 \leq \frac{1405}{17},\ x1 - \frac{6\,x2}{59} + \frac{8}{} \right. \end{cases}$$

$$false \quad \textbf{(14}$$

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

$$\Big[ [[-20, 36, 26], [-4, -25, 103], [-2, -30, 107], [-1, -36, 117], [0, -38, 118], [0, -36, 118], [1, -37, 112], [1, -34, 117], [2, -39, 113], [2, -38, 114], [10, -43, 95], [26, -51, 60], [399, -238, -776], [403, -240, -785], [453, -265, -897], [1544, -811, -3342]], \left[\left[\frac{101}{260}, 1, -\frac{159}{260}\right], \left[\frac{2012}{4509}, -\frac{6041}{27054}, -1\right], \left[-\frac{70}{337}, \frac{267}{337}, -1\right]\right]\Big] \quad \textbf{(15}$$

# Benchmarks 2D

E&C represents “enumeration and convex hull”, which in Maple is done by `ZPolyhedralSets:-EnumerateIntegerPoints` and `ConvexHull`. `Normaliz` is an open source tool for computations in affine monoids, vector configurations, lattice polytopes, and rational cones.

| Volume | 27.95 | | 111.79 | | 11179.32 | |
|---|---|---|---|---|---|---|
| Algorithm | IntegerHull | E&C | IntegerHull | E&C | IntegerHull | E&C |
| Maple (ms) | 172 | 410 | 244 | 890 | 159 | 58083 |
| C/C++ (ms) | 0.284 | 0.768 | 0.339 | 1.676 | 0.286 | 6.883 |
| Normaliz (ms) | 835.730 | | 462.116 | | 1559.401 | |

Table: Integer hulls of triangles

| Volume | 58.21 | | 5820.95 | | 23283.82 | |
|---|---|---|---|---|---|---|
| Algorithm | IntegerHull | E&C | IntegerHull | E&C | IntegerHull | E&C |
| Maple (ms) | 303 | 752 | 275 | 31357 | 304 | 123159 |
| C/C++ (ms) | 0.451 | 0.565 | 0.478 | 0.657 | 0.396 | 0.682 |
| Normaliz (ms) | 2.837 | | 1216.238 | | 740.559 | |

Table: Integer hulls of hexagons

# Benchmarks 3D

| Volume | 447.48 | | 6991.89 | | 55935.2 | |
|---|---|---|---|---|---|---|
| Algorithm | IntegerHull | E&C | IntegerHull | E&C | IntegerHull | E&C |
| Maple (ms) | 977 | 7289 | 1223 | 74804 | 1378 | 531904 |
| C/C++ (ms) | 4.488 | 0.826 | 4.615 | 0.923 | 4.624 | 1.527 |
| Normaliz (ms) | 851.495 | | 956.666 | | 793.192 | |

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

| Volume | 412.58 | | 7050.81 | | 60417.63 | |
|---|---|---|---|---|---|---|
| Algorithm | IntegerHull | E&C | IntegerHull | E&C | IntegerHull | E&C |
| Maple (ms) | 1476 | 5711 | 1573 | 60233 | 1728 | 512101 |
| C/C++ (ms) | 11.049 | 21.235 | 16.001 | 145.068 | 23.822 | 2082.559 |
| Normaliz (ms) | 7862.109 | | N/A | | N/A | |

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

# Outline

# Conclusions and remarks

Over the reals:

- The notion of regular semi-algebraic system is a natural generalization of that of a regular chain for isolating real solutions.
- The incremental flavor of `RealTriangularize` is experimentally more effective than its elimination approach.
- `RealTriangularize` has inspired follow-up works (CAD and QE based on regular chains and proceeding incrementally).
- The implementation of `RealTriangularize` relies on CAD but this could be relaxed. (The complexity analysis uses Renagar's work.)
- Can the notion of a regular semi-algebraic system be weakened so as to reduce the cost of the decomposition while remaining useful?

Over the integers:

- The `IntegerPointDecomposition` is also inspired by the theory of regular chains.
- It often produces more information than needed and this has a cost.
- Our `IntegerHull` solves that issue and is currently adapted to support parameters and thus QE problems.

# Thank You!

# References


[1] R. J. Bradford, C. Chen, J. H. Davenport, M. England, M. Moreno Maza, and D. J. Wilson. "Truth Table Invariant Cylindrical Algebraic Decomposition by Regular Chains". In: Computer Algebra in Scientific Computing - 16th International Workshop, CASC 2014, Warsaw, Poland, September 8 Ed. by V. P. Gerdt, W. Koepf, W. M. Seiler, and E. V. Vorozhtsov. Vol. 8660. Lecture Notes in Computer Science. Springer, 2014, pp. 44–58. doi: 10.1007/978-3-319-10515-4_4. url: https://doi.org/10.1007/978-3-319-10515-4%5C_4.

[2] C. Chen, J. H. Davenport, J. P. May, M. Moreno Maza, B. Xia, and R. Xiao. "Triangular decomposition of semi-algebraic systems". In: J. Symb. Comput. 49 (2013), pp. 3–26.

[3] C. Chen, J. H. Davenport, M. Moreno Maza, B. Xia, and R. Xiao. "Computing with semi-algebraic sets: Relaxation techniques and effective boundaries". In: J. Symb. Comput. 52 (2013), pp. 72–96. doi: 10.1016/j.jsc.2012.05.013. url: https://doi.org/10.1016/j.jsc.2012.05.013.

[4] C. Chen and M. Moreno Maza. "Algorithms for computing triangular decomposition of polynomial systems". In: J. Symb. Comput. 47.6 (2012), pp. 610–642.

[5] C. Chen and M. Moreno Maza. "An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions". In: Computer Mathematics, 9th Asian Symposium (ASCM 2009), Fukuoka, Japan, December 2009, 10th Asian Symposiu Ed. by R. Feng, W. Lee, and Y. Sato. Springer, 2012, pp. 199–221. doi: 10.1007/978-3-662-43799-5_17. url: https://doi.org/10.1007/978-3-662-43799-5%5C_17.

[6] C. Chen and M. Moreno Maza. "Quantifier elimination by cylindrical algebraic decomposition based on regular chains". In: J. Symb. Comput. 75 (2016), pp. 74–93.

[7] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. "Computing cylindrical algebraic decomposition via triangular decomposition". In: Symbolic and Algebraic Computation, International Symposium, ISSAC 2009, Seoul, Republic of Korea, July 29-31, 2 Ed. by J. R. Johnson, H. Park, and E. L. Kaltofen. ACM, 2009, pp. 95–102. doi: 10.1145/1576702.1576718. url: https://doi.org/10.1145/1576702.1576718.

[8] S. Covanov, D. Mohajerani, M. Moreno Maza, and L. Wang. "Big Prime Field FFT on Multi-core Processors". In: International Symposium on Symbolic and Algebraic Computation (ISSAC '19), Beijing, China, July 15-18, 2019. 2019, pp. 106–113.

[9] R. Jing and M. Moreno Maza. "Computing the Integer Points of a Polyhedron, I: Algorithm". In: Computer Algebra in Scientific Computing - 19th International Workshop, CASC 2017, Beijing, China, September 18- Ed. by V. P. Gerdt, W. Koepf, W. M. Seiler, and E. V. Vorozhtsov. Vol. 10490. Lecture Notes in Computer Science. Springer, 2017, pp. 225–241. doi: 10.1007/978-3-319-66320-3_17. url: https://doi.org/10.1007/978-3-319-66320-3%5C_17.

[10] R. Jing and M. Moreno Maza. "Computing the Integer Points of a Polyhedron, II: Complexity Estimates". In: Computer Algebra in Scientific Computing - 19th International Workshop, CASC 2017, Beijing, China, September 18- Ed. by V. P. Gerdt, W. Koepf, W. M. Seiler, and E. V. Vorozhtsov. Vol. 10490. Lecture Notes in Computer Science. Springer, 2017, pp. 242–256. doi: 10.1007/978-3-319-66320-3_18. url: https://doi.org/10.1007/978-3-319-66320-3%5C_18.

[11] R. Jing, M. Moreno Maza, and D. Talaashrafi. "Complexity Estimates for Fourier-Motzkin Elimination". In: Computer Algebra in Scientific Computing - 22nd International Workshop, CASC 2020, Linz, Austria, September 14-1 Ed. by F. Boulier, M. England, T. M. Sadykov, and E. V. Vorozhtsov. Vol. 12291. Lecture Notes in Computer Science. Springer, 2020, pp. 282–306. doi: 10.1007/978-3-030-60026-6_16. url: https://doi.org/10.1007/978-3-030-60026-6%5C_16.

[12] F. Lemaire, M. Moreno Maza, and Y. Xie. "The RegularChains library in MAPLE". In: ACM SIGSAM Bulletin 39.3 (2005), pp. 96–97.

[13] M. Moreno Maza and L. Wang. "Computing the Integer Hull of Convex Polyhedral Sets". In: Computer Algebra in Scientific Computing - 24th International Workshop, CASC 2022, Gebze, Turkey, August 22-26, Ed. by F. Boulier, M. England, T. M. Sadykov, and E. V. Vorozhtsov. Vol. 13366. Lecture Notes in Computer Science. Springer, 2022, pp. 246–267. doi: 10.1007/978-3-031-14788-3_14. url: https://doi.org/10.1007/978-3-031-14788-3%5C_14.

[14] M. Moreno Maza and L. Wang. "On the Pseudo-Periodicity of the Integer Hull of Parametric Convex Polygons". In: Computer Algebra in Scientific Computing - 23rd International Workshop, CASC 2021, Sochi, Russia, September 13-1 Ed. by F. Boulier, M. England, T. M. Sadykov, and E. V. Vorozhtsov. Vol. 12865. Lecture Notes in Computer Science. Springer, 2021, pp. 252–271. doi: 10.1007/978-3-030-85165-1_15. url: https://doi.org/10.1007/978-3-030-85165-1%5C_15.

[15] D. Talaashrafi, J. Doerfert, and M. Moreno Maza. "A Pipeline Pattern Detection Technique in Polly". In: Workshop Proceedings of the 51st International Conference on Parallel Processing, ICPP Workshops 2022, Bordeaux, ACM, 2022, 18:1–18:10. doi: 10.1145/3547276.3548445. url: https://doi.org/10.1145/3547276.3548445.

[16] D. Talaashrafi, M. Moreno Maza, and J. Doerfert. "Towards Automatic OpenMP-Aware Utilization of Fast GPU Memory". In: OpenMP in a Modern World: From Multi-device Support to Meta Programming - 18th International Workshop on Op Ed. by M. Klemm, B. R. de Supinski, J. Klinkenberg, and B. Neth. Vol. 13527. Lecture Notes in Computer Science. Springer, 2022, pp. 67–80. doi: 10.1007/978-3-031-15922-0_5. url: https://doi.org/10.1007/978-3-031-15922-0%5C_5.