# An Introduction on Polynomial System Solving

# #1
# Setting and basic properties

System $\mathcal{P}$ of polynomial equations

$$f_1 = 0, \ldots, f_m = 0$$

where $f_i \in k[x_1, \ldots, x_n]$ for some field $k$.

Throughout the talk we (hopefully) assume $m$ **equations** in $n$ **variables**.

Solutions? Go to algebraic closure $K$ of $k$

Let $I = \langle f_1, \ldots, f_m \rangle \subset R := K[x_1, \ldots, x_n]$.

Compute

$$\begin{aligned} V(I) &= \{z \in K^n \mid f_i(z) = 0 \text{ for } i = 1, \ldots, m\} \\ &= \{z \in K^n \mid f(z) = 0 \ \forall f \in I\}. \end{aligned}$$

$\mathcal{P}$ is **inconsistent** if it has no solution.

$\mathcal{P}$ is **overdetermined** if $m > n$.
Not all overdetermined systems are inconsistent, e.g.
$\mathcal{P} = \left(x^3 - 1 = 0, x^2 - 1 = 0\right)$ has solution $x = 1$.

$\mathcal{P}$ is **underdetermined** if $m < n$.
An underdetermined system is either inconsistent
or it has infinitely many solutions.

$\mathcal{P}$ is **positive-dimensional** if it has infinitely many solutions.

$\mathcal{P}$ is **zero-dimensional** if it has finitely many solutions. (Corresponds to $\dim \mathrm{V}(\mathrm{I}) = 0$.)

Let $\mathcal{P}$ be zero-dimensional and $m = n$:

**Bézout's theorem** gives us:
$\deg f_i = d_i \Rightarrow$ at most $\prod_{i=1}^{m} d_i$ solutions.

Bound is sharp and exponential in number of variables.

In general: **solving is difficult**.

# What does solving mean?

If $\mathcal{P}$ is **positive-dimensional** then counting solutions is meaningless.

Try to find a description of the solutions from which we can **easily** extract the **relevant data**.

Algebraic geometry, here we go!

"*Does $\mathcal{P}$ over $\mathbb{Q}$ has a finite number of* ***real*** *solutions? If so, compute them.*"

**Cylindrical algebraic decomposition (CAD)**:

Complexity: doubly exponential in $n$.

A **semi-algebraic set / cell** is a finite union of subsets of $\mathcal{R}^n$ ($\mathcal{R}$ is a real closed field) defined by a finite sequence of polynomial equations or inequalities.

A CAD is a decomposition of $\mathcal{R}^n$ into connected semi-algebraic sets on which each polynomial has constant sign $(+, -, 0)$.

When projecting $\pi : \mathcal{R}^n \to \mathcal{R}^{n-k}$ then for cells $C$ and $D$, either $\pi(C) = \pi(D)$ or $\pi(C) \cap \pi(D) = \emptyset$.

$\Rightarrow$ Images of $\pi$ give CAD of $\mathcal{R}^{n-k}$.

## Algorithmic idea

Sequence of **projections** $\mathcal{R}^n \to \mathcal{R}^{n-1} \to \ldots \to \mathcal{R}$.

Take $f = \prod_{i=1}^{m} f_i$, let $g = \mathrm{gcd}(f, f')$ (w.r.t. $x_n$).
Zeroes of $g$ and intersections of $f_i$ give cell boundaries (no local variation of $f = 0$ when perturbing $x_n$).

Zeroes of univariate polynomials provide critical points for cell decomposition of $\mathcal{R}$ in zero- and one-dimensional cells.

**Lift** them back up, get a cylinder of cells from $\mathcal{R}$ in $\mathcal{R}^2$.

Go on till $\mathcal{R}^n$.

Let's restrict to **zero-dimensional** systems in the following.

Solving means to compute all solutions.

There are two main ways to output the solutions:

## Numerical representation

For real/complex solutions one in general uses numeric approximations.

A **certified** solution provides a bound on the error of the approximations in order to separate the different solutions.

## Algebraic representation

Several different ways (we talk about them).

All boil down to a representation of the solution set by **univariate** equations.

Then compute a numerical approximation of the solutions by solving this univariate system.

# #2
# Numerical solving – quick & dirty

One can use general solvers for **non-linear** systems.

**Problems**

- ▷ In general one cannot find all solutions.
- ▷ If the method does not find a solution there is no certificate that there really exists no solution.

**Notably mentions**

- ▷ Newton's method (fast if we start near a solution)
- ▷ Optimization (meeh)

**Homotopy continuation method**

Semi-numerical, supposes $m = n$.

The algorithm consists of four main steps:

## Step 1

An **upper bound** on the number of solutions is computed.

This step is critical, the bound $B$ should be as sharp as possible.

## Step 2

Another polynomial system

$$g_1 = 0, \ldots, g_n = 0$$

is generated with exactly $B$ **easily computable** solutions.

## Step 3

We construct a **homotopy** between both systems:

$$(1-t)g_1 + tf_1 = 0, \ldots, (1-t)g_n + tf_n = 0.$$

Not only straight lines, but also other paths,
in order to avoid singularities and other trouble.

## Step 4

Now we **follow** the solutions of the $g_i$s $(t = 0)$ to the $f_i$s $(t = 1)$.

If $t_k < t_l$ then we get the solutions for $t = t_l$ from those for $t_k$ using **Newton**'s method.

**Difficult task**: How to choose $t_l - t_k$?

▷ If **too large**, convergence is too slow, even jumps from one solution path to a different one is possible.

▷ If **too small**, then too many steps may slow down the computation.

There is a recent paper by **Verschelde** on using parallel approaches.

**Main idea**

Different solution paths are independent of each other.

## #3
Algebraic representations of solutions

A **triangular set** is a

non-empty set $T = \{g_1, \ldots, g_s\} \subset K[x_1, \ldots, x_n]$ such that

- ▷ no $g_i$ is constant,
- ▷ all main variables are different,
- ▷ $|T| \leq n$.

(The main variable mvar$(g)$ of a poly $g$ is the greatest appearing variable.)

A **regular chain**

$T = \{g_1, \ldots, g_s\}$ is a triangular set such that

- ▷ $\mathsf{mvar}(g_1) < \ldots < \mathsf{mvar}(g_s)$.
- ▷ Let $h = \prod_{i=1}^{s} \mathsf{lm}(g_i)$. Then $\mathsf{resultant}(h, T) \neq 0$ where each internal resultant is computed w.r.t. the main variable of $g_i$.

**Main idea** by **Kalkbrenner**

Every irreducible variety is uniquely determined by one of its generic points.

Regular chains give us exactly these generic points.

## Example

Take $R = \mathbb{Q}[x, y, z]$ such that $x < y < z$.

Then $T = \{y^2 - x^2, y(z - x)\}$ is a triangular set and a regular chain.

Two generic points given by $T$ are $(t, t, t)$ and $(t, -t, t)$ for $t$ transcendental over $\mathbb{Q}$.

Thus we have two irreducible components: $\{y - x, z - x\}$ and $\{y + x, z - x\}$.

## Note

$y$ is the content of the second polynomial of $T$ and can be removed.

The dimension of each component is one, the number of free variables.

Let T be a regular chain.

The **quasi-component** of T: $W(\mathrm{T}) = V(\mathrm{T}) \setminus V(\mathrm{h})$.

Corresponding algebraic structure:
The **saturated ideal** $\mathrm{sat}(\mathrm{T}) = (\mathrm{T}) : \mathrm{h}^{\infty}$.

We have $\overline{W(\mathrm{t})} = V(\mathrm{sat}(\mathrm{T}))$.

Some properties of a regular chain $T$:

- $\triangleright$ $\text{sat}(T)$ is an unmixed ideal of dimension $n - |T|$.
- $\triangleright$ $\text{sat}\,(T \cap K[x_1, \ldots, x_i]) = \text{sat}(T) \cap K[x_1, \ldots, x_i]$.
- $\triangleright$ A triangular set is a regular chain iff
  it is *Ritt characteristic set* of its saturated ideal.

**Triangular decomposition** of a polynomial system $\mathcal{P}$:

**Kalkbrenner** style, lazy decomposition:

$\sqrt{(\mathcal{P})} = \cap_{i=1}^{k} \sqrt{\mathsf{sat}(T_i)}$.

**Lazard** style, describe all zeroes:

$V(\mathcal{P}) = \cup_{i=1}^{k} W(T_i)$.

## Zero-dimensional **regular chains**

Sequence of polys $g_1(x_1), g_2(x_1, x_2), \ldots, g_n(x_1, \ldots, x_n)$ such that for all $1 \leq i \leq n$

- ▷ $g_i$ poly in $x_1, \ldots, x_i$ such that $d_{x_i} := \deg_{x_i} g_i > 0$.
- ▷ Coefficient of $x_i^{d_{x_i}}$ is a poly in $x_1, \ldots, x_{i-1}$ that has no common zero with $g_1, \ldots, g_{i-1}$.

Thus we have a **triangular system**

$$g_1(x_1) = 0$$
$$g_2(x_1, x_2) = 0$$
$$\dots$$
$$g_n(x_1, \dots, x_n) = 0.$$

Solve first equation, make thus second univariate, ...

Working over a finite field this is wonderful.

# Over the rationals?

**Problem 1**

Coefficients might explode.

**Idea**

**Equiprojectable decomposition** by **Dahan** and **Schost**

- ▷ Bound on coefficients w.r.t. size of the input system.
- ▷ Depends only on choice of coordinates.
- ▷ Allows modular computation.

## Problem 2

Solving univariate polys with approximate coefficients is quite unstable.

## Ideas

Get regular chains in special form: **shape lemma**.

Use **rational univariate representation** starting from a general regular chain or a Gröbner basis.

**Shape lemma**
Up to a linear change of coordinates any zero-dimensional radical ideal $I$ has a LEX Gröbner basis in **shape position**, i.e.

$$G = \{x_1 - h_1(x_n), \ldots, x_{n-1} - h_{n-1}(x_n), h_n(x_n)\},$$

such that

- ▷ $D = \dim_K(R/I)$,
- ▷ $\deg h_n = D$ and
- ▷ $\deg h_i < D$ for all $1 \leq i < n$.

## **Rational univariate representation (RUR)** by **Rouillier**

Connected to the shape lemma.

Uses **separating variable** $t$, a linear combination of the other variables.

We get a system

$$
\begin{aligned}
&h(t) = 0,\\
&x_1 = h_1(t)/q(t),\\
&\dots\\
&x_n = h_n(t)/q(t),
\end{aligned}
$$

where $D = \deg h$ and $\deg q, \deg h_i < D$.

## Example

Let $\mathcal{P} = \{x^2 - 1, (x-1)(y-1), y^2 - 1\}$.

Besides $\lambda x$, $\lambda y$, and $\lambda(x+y)$ we can use any linear combination of $x$ and $y$ as separating variable. For example, take $t = \frac{x-y}{2}$. Then we get as RUR

$$t^3 - t = 0,\ x = \frac{t^2 + 2t - 1}{3t^2 - 1},\ y = \frac{t^2 - 2t - 1}{3t^2 - 1}.$$

**Properties** of a RUR

- ▷ Only defined in the zero-dimensional case.
- ▷ Only finitely many linear combinations do **not** lead to a separating variable.
- ▷ Once a separating variable is chosen **the** corresponding RUR exists and is unique.
- ▷ 1-to-1 correspondence between roots of $h$ and solutions of the system. (multiplicities coincide; triangular decompositions in general do not preserve this information.)
- ▷ If $h$ has no multiple root then $q = h'$.

Factorizing $h$ gives a RUR for each irreducible factor.

We get a **prime decomposition**
i.e. primary decomposition of the radical.

Especially if $\mathcal{P}$ has a high multiplicity we
thus get an output with much smaller coefficients.

Getting a **RUR** from a LEX Gröbner basis:

If $I$ is **radical**, take smallest variable from LEX GB as separating variable $t$.
Check that $h(t)$ is squarefree and get a RUR.

In the general case, there also exist algorithms.

If the separating variable is already known **and** if the multiplication matrices are already given then we can compute a RUR in $O\left(D^3 + nD^2\right)$.

**#4**

# Numerical solving once having the RUR

Seems easy, but evaluating one poly at the roots of another one is highly unstable.

Compute roots of $h$ with high precision.
(This may change for different roots).

- ▷ **Aberth**'s method,
- ▷ **Laguerre**'s method (Singular),
- ▷ other improved algorithms by Rouillier, Zimmermann, etc.,
- ▷ other algorithms I know nothing about.

**Laguerre**'s method

Find approximation for **one** root of a polynomial $f(x)$ of degree $d$:

Initial guess $z_0$.

For $k = 0, 1, 2, \ldots,$ some upper bound

If $f(z_k)$ is small enough, exit loop.

$G = f'(z_k)/f(z_k)$.

$H = G^2 - f''(z_k)/f(z_k)$.

$a = d/\left(G \pm \sqrt{(d-1)(dH - G^2)}\right)$ (Choose sign to get bigger absolute value of denominator.)

$z_{k+1} = z_k - a$.

## **Aberth**'s method

Find approximation for **all** roots of a polynomial $f(x) \subset \mathbb{C}[x_1, \ldots, x_n]$ of degree $d$ simultaneously:

Compute upper and lower bounds of absolute values for the $d$ roots from the coefficients of the polynomial.

Now pick randomly or evenly distributed **distinct** complex numbers $z_1, \ldots, z_d$ with absolute values within the same bounds.

For some number of iterations / until values are small enough do:

> For current approximations $z_1, \ldots, z_d$ compute
>
> $$w_k = -\frac{\frac{f(z_k)}{f'(z_k)}}{1 - \frac{f(z_k)}{f'(z_k)} \cdot \sum_{l \neq k} \frac{1}{z_k - z_l}}.$$
>
> Calculate next approximations $z'_k = z_k + w_k$ for all $1 \leq k \leq d$.

Both methods share the following properties:

If $z$ is a **simple** root then convergence is **cubically**.

Over a finite field enumerating all the roots can be done in $\tilde{O}(D)$.

In characteristic 0 finding an approximation of all real roots can also be done in $\tilde{O}(D)$.

**Overall complexity** of multivariate solving lies in the computation of a LEX Gröbner basis resp. a RUR.

## #5

How to get the RUR / LEX Gröbner basis in shape position

**F4 Algorithm** for computing DRL Gröbner basis

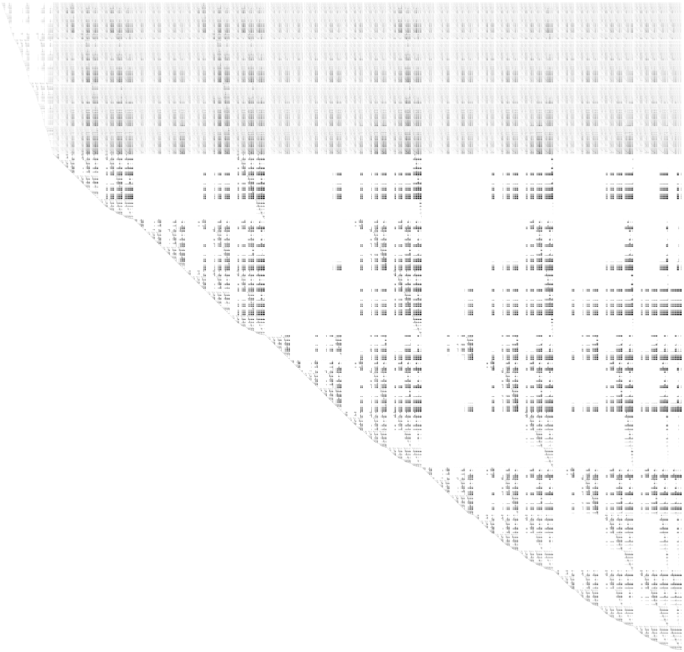
**gb** package, plain C code

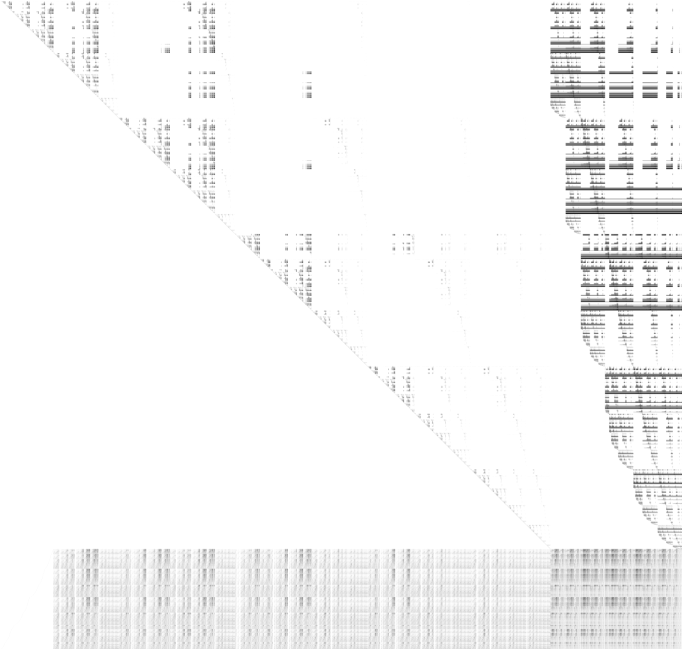
**GB.jl** for **OSCAR**

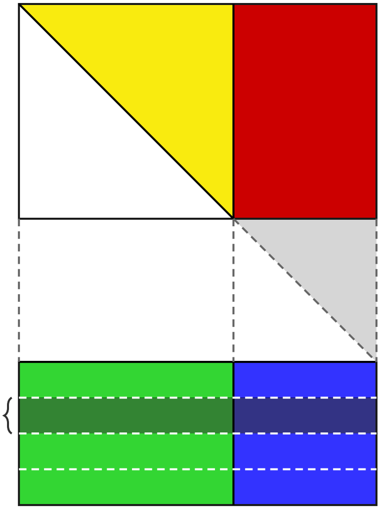
Start your **julia** session. Then

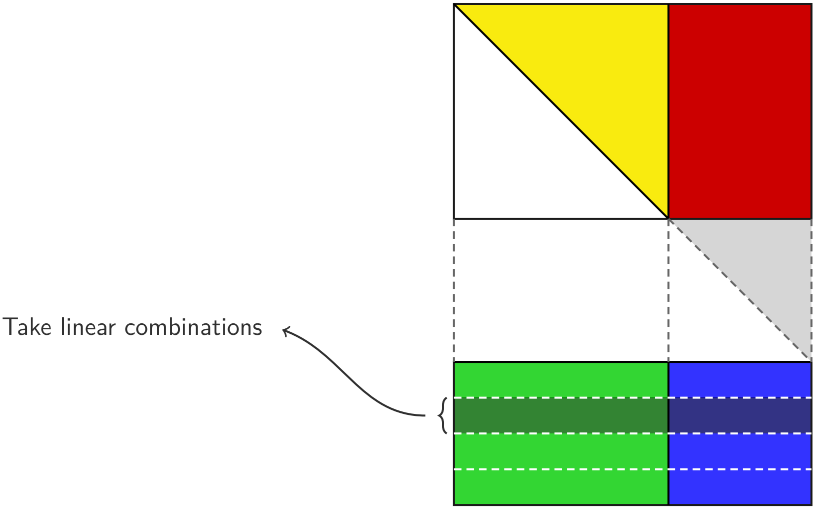
```
//Load the GB.jl library, also loads Singular.jl.
using GB

// Next we define a ring R of characteristic 2^31-1
// with DRL order and the ideal I in R generated by the
// cyclic generators with 10 variables.
R,I = GB.cyclic_10(2^31-1, :degrevlex)

// Compute Groebner basis G for I using standard
// settings of GB's F4 implementation.
G = Gb.f4(I)

// Same computation, but with specialized setting:
// hash table size = 2^21, 8 threads,
// max.  2500 s-polynomials, probabilistic linear algebra
G = Gb.f4(I, 21, 8, 2500, 42)

// Further process G using Singular stuff
H = Singular.fglm(G, :lex) // TODO as a first step?
```

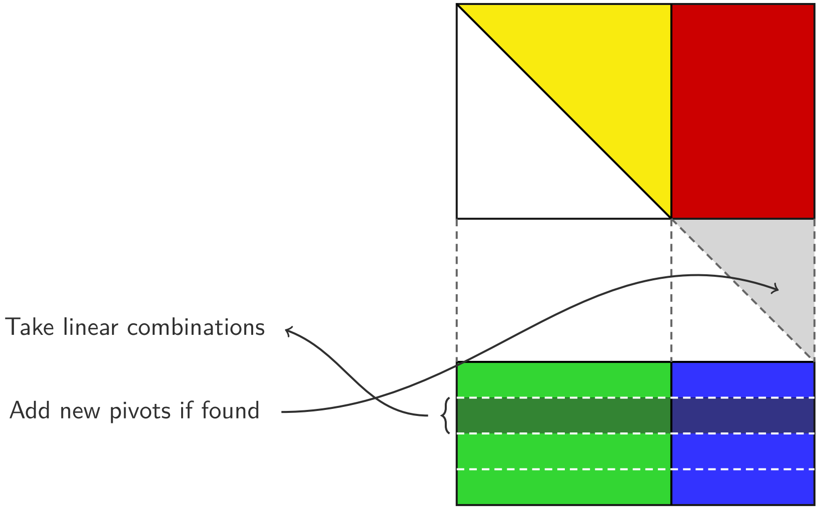
**Magma / Maple** performance for 31-bit prime fields using **probabilistic linear algebra** for reduction.



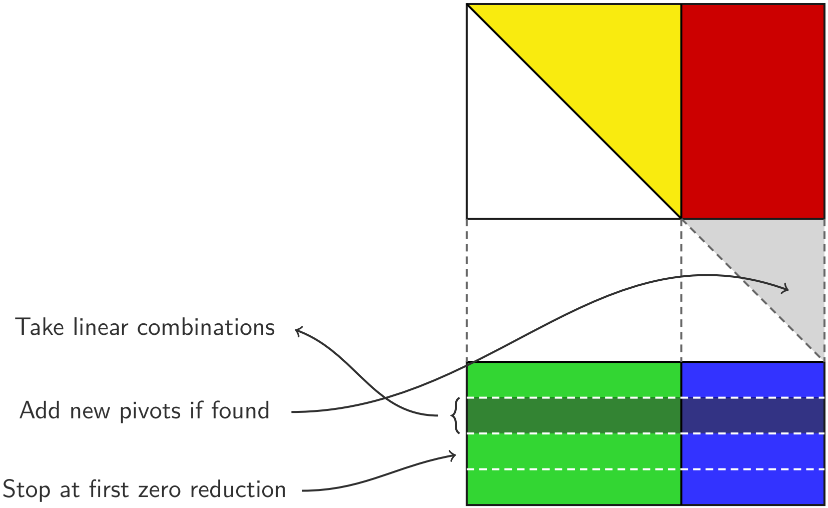





Take linear combinations



Take linear combinations
Add new pivots if found


Take linear combinations
Add new pivots if found
Stop at first zero reduction

## Todo

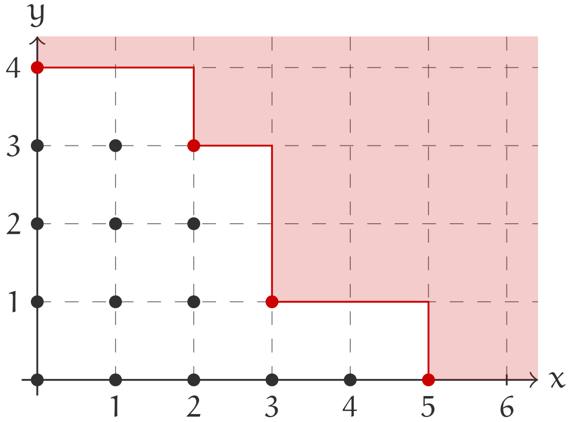
Implementation over $\mathbb{Q}$

Multimodular implementation

**Conversion** from DRL to LEX Gröbner basis using the **FGLM** algorithm.

Complexity: $O\left(nD^3\right)$

Use zero-dimensional structure of $R/I$.
DRL Gröbner basis gives us a **finite basis** $B$ for $R/I$ as vector space.



## Step 1

Generate **multiplication matrices**

$M_{x_i} : R/I \to R/I,\ \overline{p} \mapsto \overline{x_i p}$

where reduction is done w.r.t. the DRL Gröbner basis.

We have $O(nD)$ matrix-vector products of size $D \times D$ times $D \times 1$, thus a complexity of $O\left(nD^3\right)$ for this step.

## Step 2

Test **linear dependency** of $O(nD)$ vectors of size $D \times 1$, done in $O\left(nD^3\right)$ arithmetic operations:

Add 1 to $B'$ and to $C$. Multiply 1 by all variables, add them to $L$.

Take $m \in L$ minimal w.r.t. LEX and reduce $m$ w.r.t. $G$.

- ▷ If $\overline{m}$ is linearly independent w.r.t. $C$ then add $m$ to $B'$, $\overline{m}$ to $C$ and add multiples of $m$ to $L$.
- ▷ If $\overline{m}$ is linearly dependent w.r.t. $C$ then $\overline{m - \sum_i \lambda_i b_i} = \overline{0}$, i.e. $m - \sum_i \lambda_i b_i \in I$. Thus add $m - \sum_i \lambda_i b_i$ to $G'$.

Method by **Mourrain**, **Telen** and **van Barel** (2018)

They propose a new method for constructing the multiplication matrices.

Allows **finite precision** computation.

Gröbner bases are unstable, border bases need a good initial choice of basis taking global numerical properties into account.

Idea of **truncated normal forms**.

## Setting

Let $I = \langle f_1, \ldots, f_m \rangle \subset R = \mathbb{C}[x_1, \ldots, x_n]$ be zero-dimensional, say $\dim_{\mathbb{C}}(R/I) = D$.

Let $V, W$ be finite dimensional vector spaces of $R$ such that $V \to \mathbb{C}^D$ is surjective and $x_i W \subset V$ for all $i$.

Denote

$$\begin{array}{cccccccc} \phi_I : & V_1 & \times & \ldots & \times & V_m & \to & V, \\ & (q_1 & , & \ldots & , & q_m) & \mapsto & \sum_{i=1}^{m} q_i f_i. \end{array}$$

such that $f_i V_i \subset V$ for all $i$.

**Algorithm**

Compute $\phi_I$ and let $N \leftarrow \ker(\phi_I^T)^T$.

Choose $h = h_0 + \sum_{i=1}^{n} h_i x_i$ such that $hW \subset V$.

Set $N_0 : W \mapsto N(hw)$ for $w \in W$.

$N' \leftarrow$ an invertible submatrix of $N_0$.

$B \leftarrow$ monomials corresponding to columns of $N'$.
(This gives us an isomorphism between the basis $B$ and $R/I$.)

For $i = 1, \ldots, n$ do

$N_i \leftarrow$ columns of $N$ corresponding to $x_i B$.

$M_{x_i} \leftarrow (N')^{-1} N_i$.

Return $(M_{x_1}, \ldots, M_{x_n})$.

$\mathrm{N}'$ must be chosen such that an **inverse** (last step) can be computed **accurately**.

Use column pivoted **QR factorization** on $\mathrm{N}_0$.

$\Rightarrow$ We get a monomial basis with good numerical properties.

## Comparison to GB approach

| Mourrain et al | Gröbner bases |
| --- | --- |
| Construct $\phi_I$ and compute $N$. | Compute reduced DRL GB $G$ with induced normal form NF. |
| QR factorization with pivoting on $N\|_W$ to get $N'$ corresponding to a basis $B$ for $R/I$. | Find a normal set $B$ from $G$. |
| Compute $N_i$ and set $M_{x_i} = (N')^{-1}N_i$. | Compute multiplication matrices $M_{x_i}$ by applying NF to $x_iB$. |

# Thanks

Questions? Remarks?