Algorithms for the universal decomposition algebra*

Romain Lebreton
Laboratoire d'Informatique
École Polytechnique
Palaiseau - France

Éric Schost
Computer Science Department
University of Western Ontario
London - Canada

ALGO seminar, Rocquencourt

June 11, 2012

$*$. This document has been written using the GNU TeXmacs text editor (see www.texmacs.org).

Presentation

Let k be a field of characteristic 0 or sufficiently large.

We fix $f = X^n + \sum_{i=1}^{n} (-1)^i f_i X^{n-i} \in k[X]$ separable of degree n.

We note $\alpha_1, \ldots, \alpha_n$ its roots.

Symmetric variety of roots of f		Ideal of symmetric relations
$\mathbb{V}_{\mathcal{I},\bar{k}} = \{(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)}) \mid \sigma \in \mathfrak{S}_n\}$	$\leftrightarrow$	$\mathcal{I} = \langle E_i(X_1, \ldots, X_n) - f_i \rangle_{i=1,\ldots,n} \subseteq k[X_1, \ldots, X_n]$

The universal decomposition algebra is $\mathbb{A} := k[X_1, \ldots, X_n]/\mathcal{I}$, its degree is $\delta := n!$.

For all $P \in \mathbb{A}$, let's denote its characteristic polynomial

$$\mathcal{X}_{P,\mathbb{A}}(T) := \prod_{\sigma \in \mathfrak{S}_n} (T - P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})) \in k[T].$$

State of the art : absolute resolvents

Absolute Lagrange's resolvent :

$$L_P(T) := \prod_{\sigma \in \mathfrak{S}_n // \mathrm{Stab}\, P} (T - P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})) \in k[T].$$

We have $\mathcal{X}_P = (L_P)^{\#\mathrm{Stab}\, P}$.

Symbolic methods for the computation of absolute resolvents :

- by resultants [Lagrange], [Soicher, '81], [Giusti *et al.*, '88], [Lehobey,'97]
- by symmetric functions [Lagrange], [Valibouze, '88], [Casperson, McKay, '94];
- by Groebner bases [Giusti *et al.*, '88], [Arnaudiès, Valibouze, '93];
- by invariants [Berwick, '29], [Foulkes, '31].

$\rightsquigarrow$ Little is know about complexity. Algorithm with at least quadratic complexity $\Omega(\delta^2)$.

State of the art : universal decomposition algebra

Triangular representation	Univariate representation
Cauchy modules $C_i \in k[X_1, \ldots, X_i]$ for $1 \leqslant i \leqslant n$.	Minimal polynomial Q of a primitive linear form Λ. Parametrizations $(S_i)_{1 \leqslant i \leqslant n}$.
$\mathbb{A}_1 = k[X_1]/(C_1)$ $\vdots$ $\mathbb{A}_j = k[X_1, \ldots, X_j]/(C_1, \ldots, C_j)$ $\vdots$ $\mathbb{A} = \mathbb{A}_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n)$	$\mathbb{A} \simeq k[T]/(Q)$ $X_i \longmapsto S_i(T)$ $\Lambda \longleftarrow T$

Cost of the representation of $\mathbb{A}$:

$o(\delta)$ by the recursive formula :

$$C_{i+1} = \frac{(C_i(X_1, \ldots, X_i) - C_i(X_1, \ldots, X_{i+1}))}{X_i - X_{i+1}}$$

with $C_1 := f(X_1)$.

$\tilde{\mathcal{O}}(\delta^3)$ by FGLM or RUR algorithm [Faugère et al., '93], [Rouillier, '99]

$\tilde{\mathcal{O}}(\delta^2)$ by geometric resolution [Giusti *et al.*, '01], [Heintz *et al.*, '00]

$\tilde{\mathcal{O}}(\delta^{1.69})$ by modular composition [Poteaux, Schost, '11]

State of the art : universal decomposition algebra

Triangular representation	Univariate representation
Cauchy modules $C_i \in k[X_1, \ldots, X_i]$ for $1 \leqslant i \leqslant n$.	Minimal polynomial Q of a primitive linear form Λ. Parametrizations $(S_i)_{1 \leqslant i \leqslant n}$.
$\mathbb{A}_1 = k[X_1]/(C_1)$ $\vdots$ $\mathbb{A}_j = k[X_1, \ldots, X_j]/(C_1, \ldots, C_j)$ $\vdots$ $\mathbb{A} = \mathbb{A}_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n)$	$\mathbb{A} \simeq k[T]/(Q)$ $X_i \longmapsto S_i(T)$ $\Lambda \longleftarrow T$

Cost of arithmetic operations in $\mathbb{A}$:

- multiplication $\tilde{\mathcal{O}}(\delta)$ [Bostan et *al.*, 2011], not implemented, significant constant	$\tilde{\mathcal{O}}(\delta)$, simple and efficient
- division (when possible) no quasi-optimal algorithm	$\tilde{\mathcal{O}}(\delta)$, simple and efficient

Our contribution

Let $\mathbb{A}_m = \mathbb{A} \cap k[X_1, \ldots, X_m]$ and $\delta_m = \frac{n!}{(n-m)!}$ its degree.

Theorem. *One can compute a primitive linear form $\Lambda \in \mathbb{A}_m$ and the univariate representation $\mathfrak{P} = (Q, S_1, \ldots, S_m) \in k[Z_m]^{m+1}$ with*

$$\begin{array}{ccc} \mathbb{A}_m = k[X_1, \ldots, X_m]/\mathcal{I} & \simeq & k[Z_m]/Q(Z_m) \\ X_i & \mapsto & S_i(Z_m) \\ \Lambda & \leftarrow\!\shortmid & Z_m \end{array}$$

with a Las Vegas algorithm of expected cost $\mathcal{O}(n^{(\omega+1)/2}\, m\, \mathrm{M}(\delta_m))$.

Theorem. *For all $P \in \mathbb{A}_m$, the characteristic polynomial $\mathcal{X}_{P,\mathbb{A}_m} \in k[T]$ costs*

$$\mathcal{O}(n^{(\omega+1)/2}\, m\, \mathrm{M}(\delta_m))$$

arithmetic operations in k.

Applications

- Computation of $\mathcal{X}_P$

  $\rightsquigarrow$ Symbolic computation of absolute Lagrange's resolvent in time $\tilde{\mathcal{O}}(\delta_m)$;

- Computation of $\mathbb{A} \simeq k[T]/Q(T)$

  $\rightsquigarrow$ Change of representation in time $\tilde{\mathcal{O}}(\delta)$

  $\rightsquigarrow$ Division in $\mathbb{A}$ in time $\tilde{\mathcal{O}}(\delta)$

  $\rightsquigarrow$ Efficient algorithms for trace, minimal polynomial computations...

  $\rightsquigarrow$ Dynamic splitting field, [Della Dora *et al.*, 1985]

  $\rightsquigarrow$ Effective invariant theory

Outline of the talk

Newton sums methods:

i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in \mathbb{A}_m$

ii. Change of representation : Up and Down

iii. Univariate representation of $\mathbb{A}_m$

iv. Benchmarks

Resultant methods:

i. Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$

ii. Benchmarks

iii. Generalizations

Newton sums

Definition. *Let $g \in k[X]$ monic and $\beta_1, \ldots, \beta_n$ all its root in a suitable extension. Then the i-th Newton sum of g is*

$$S_i(g) := \sum_{\ell=1}^{n} (\beta_\ell)^i \in k.$$

The Newton representation of g is $(S_i(g))_{0 \leqslant i \leqslant n}$.

Proposition. *The conversion from and to the Newton representation can be done in time $\mathcal{O}(\mathrm{M}(n))$.*

Lemma. *Multiplication in the Newton representation:*

$$S_i(f\,g) = S_i(f) + S_i(g).$$

Characteristic polynomial of linear forms

Definition. *Let* $f, g \in k[T]$ *such that* $f = \prod_{i=1,\ldots,r} (T - \alpha_i)$, $g = \prod_{j=1,\ldots,s} (T - \beta_j)$ *in* $\bar{k}$. *Then*

$$f \oplus g := \prod_{1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s} (T - (\alpha_i + \beta_j)) \in k[T]$$

$$(\text{resp.}) \quad f \otimes g := \prod_{1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s} (T - (\alpha_i \cdot \beta_j)) \quad \in k[T]$$

Proposition. *If* $\deg f, \deg g \leqslant n$, *then* $f \otimes g$ *and* $f \oplus g$ *can be computed in time* $\mathcal{O}(\mathrm{M}(n^2))$.

Proof. One has

$$S_i(f \otimes g) = S_i(f)\, S_i(g)$$

and

$$\sum_{i \in \mathbb{N}} \frac{S_i(f \oplus g)}{i!} T^i = \left(\sum_{i \in \mathbb{N}} \frac{S_i(f)}{i!} T^i \right) \left(\sum_{i \in \mathbb{N}} \frac{S_i(g)}{i!} T^i \right).$$

$\square$

Characteristic polynomial of linear forms

Our goal: Let $\Lambda = \lambda_1 X_1 + \cdots + \lambda_n X_n \in \mathbb{A}_n$, compute

$$\mathcal{X}_\Lambda(T) := \prod_{\sigma \in \mathfrak{S}_n} (T - (\lambda_1 \alpha_{\sigma(1)} + \cdots + \lambda_n \alpha_{\sigma(n)})).$$

Examples:

- $f \otimes (X - \lambda) = \prod_{i=1}^n (T - \lambda \alpha_i) = \mathcal{X}_{\lambda X_1, \mathbb{A}_1}$
- If $\mathcal{R} = \{\alpha_1, \ldots, \alpha_n\}$, then

$$\begin{aligned} f \oplus f &= \prod_{\alpha, \beta \in R} (T - (\alpha + \beta)) \\ &= \prod_{\alpha \neq \beta \in R} (T - (\alpha + \beta)) \prod_{\alpha \in R} (T - 2\alpha) \\ &= \mathcal{X}_{X_1 + X_2, \mathbb{A}_2} \cdot \mathcal{X}_{2X_1, \mathbb{A}_2} \end{aligned}$$

Characteristic polynomial of linear forms

Formula from [Casperson, McKay, '94]:

$$\mathcal{X}_{X_1+\cdots+X_m} = \prod_{h=1}^{m} \left((\mathcal{X}_{X_1+\cdots+X_{m-h}}) \oplus (f \otimes (T-h)) \right)^{(-1)^{h+1}}$$

Proposition. (Generalization)

- *One has*

$$\mathcal{X}_{\Lambda_j,\mathbb{A}_j}(T) = \frac{\mathcal{X}_{\Lambda_{j-1},\mathbb{A}_{j-1}}(T) \oplus (f \otimes (T-\lambda_j))}{\prod_{i=1}^{j-1} \mathcal{X}_{\Lambda_{j-1}+\lambda_j X_i,\mathbb{A}_{j-1}}(T)}, \qquad (1)$$

  where $\Lambda_j = \lambda_1 X_1 + \cdots + \lambda_j X_j \in \mathbb{A}_j$.

- *The associated recursive algorithm* ***NewtonSums*** *computes* $\mathcal{X}_{\Lambda,\mathbb{A}_m}$ *in time* $\mathcal{O}(2^n \, \mathrm{M}(\delta_m))$.

Advantages: Algorithm in the Newton representation, handle multiplicities, memoization

Drawback: Factor 2^n

Parametrizations

Relation characteristic polynomial / univariate representation :

- minimal polynomial : $Q_j(T) = \mathcal{X}_{\Lambda_j}(T)$;
- parametrizations :

Lemma. *Let* $K := k[T_1, \ldots, T_n]$ *and* $\Lambda := T_1 X_1 + \cdots + T_n X_n \in K[X_1, \ldots, X_n]$. *Then* $\mathcal{X}_\Lambda \in k[T, T_1, \ldots, T_n]$ *and*

$$X_i = -\left(\frac{\partial \mathcal{X}_\Lambda}{\partial T_i}\right) / \left(\frac{\partial \mathcal{X}_\Lambda}{\partial T}\right) \text{ in } \mathbb{A}_K.$$

In practice, we use tangent numbers $K := k[\varepsilon]/(\varepsilon^2)$ to compute derivatives :

– If $\Lambda^\varepsilon := (\lambda_1 + \varepsilon) X_1 + \lambda_2 X_2 + \cdots + \lambda_n X_n$ then $\mathcal{X}_{\Lambda^\varepsilon} = \mathcal{X}_\Lambda + \varepsilon \frac{\partial \mathcal{X}_\Lambda}{\partial T_1}$.

Outline of the talk

Newton sums methods:

i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in \mathbb{A}_m$

ii. Change of representation : Up and Down

iii. Univariate representation of $\mathbb{A}_m$

iv. Benchmarks

Resultant methods:

i. Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$

ii. Benchmarks

iii. Generalizations

Lift-up and push-down

Goal: Compute efficiently $\mathrm{Up}\colon\ \mathbb{A}_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n) \longrightarrow k[Z_n]/(Q_n(Z_n))$ and $\mathrm{Down} = \mathrm{Up}^{-1}$.

Elementary change of representation:

$$\begin{array}{ccccc} & \mathbb{A}_i[X_{i+1}]/(C_{i+1}) & \simeq & \mathbb{A}_{i+1} & \\ & \downarrow & & \downarrow & . \\ \mathrm{up}_i: & k[Z_i, X_{i+1}]/(Q_i(Z_i), C_{i+1}(Z_i, X_{i+1})) & \longrightarrow & k[Z_{i+1}]/(Q_{i+1}) & \end{array}$$

Example:

$$\mathrm{Up}: \quad \mathbb{A}_3 = (k[Z_1]/C_1)[X_2, X_3]/(C_2, C_3) \longrightarrow_{\mathrm{up}_1} (k[Z_2]/Q_2)[X_3]/(C_3) \longrightarrow_{\mathrm{up}_2} k[Z_3]/(Q_3)$$

with $n = 3$, $Z_1 = X_1$.

Lift-up and push-down

Goal: Compute efficiently $\mathrm{Up}\colon \mathbb{A}_n = k[X_1, \ldots, X_n]/(C_1, \ldots, C_n) \longrightarrow k[Z_n]/(Q_n(Z_n))$ and $\mathrm{Down} = \mathrm{Up}^{-1}$.

Elementary change of representation:

$$\mathrm{up}_i: \begin{array}{ccc} k[Z_i, X_{i+1}]/(Q_i(Z_i), C_{i+1}(Z_i, X_{i+1})) & \longrightarrow & k[Z_{i+1}]/(Q_{i+1}) \\ Z_i & \longmapsto & Z_{i+1} - \lambda_{i+1} S_{i+1,i+1}(Z_{i+1}) \\ X_{i+1} & \longmapsto & S_{i+1,i+1}(Z_{i+1}) \end{array}.$$

Algorithm up_i:

Input: $P \in k[Z_i, X_{i+1}]$

Ouput: $\mathrm{up}_i(P) \in k[Z_{i+1}]/(Q_{i+1})$

1. Compute $\tilde{P}(Z_i, X_{i+1}) = P(Z_i - \lambda_{i+1} X_{i+1}, X_{i+1}) \in (k[X_{i+1}]/f(X_{i+1}))[Z_i]$ $\quad \mathrm{M}(n)\mathrm{M}(\delta_i)$
2. Substitute $X_{i+1} \leftarrow S_{i+1,i+1}(Z_{i+1})$ in $\tilde{P}(Z_i, X_{i+1})$ $\quad n\,\mathrm{M}(\delta_{i+1})$

Lift-up and push-down

Goal: Compute efficiently $\mathrm{Up}\colon\ \mathbb{A}_n \longrightarrow k[Z_n]/(Q_n(Z_n))$ and its converse map $\mathrm{Down} = \mathrm{Up}^{-1}$.

Proposition.

1. *Given Q_i, Q_{i+1} and $S_{i+1,i+1}$, we can apply up_i in time $\mathcal{O}(\mathrm{M}(n)\,\mathrm{M}(\delta_{i+1}))$.*
2. *Given $(Q_i, S_{i,i})_{2\leqslant i\leqslant n}$, we can apply Up in time $\mathcal{O}(\mathrm{M}(n)\,n\,\mathrm{M}(\delta))$.*

Example:

$$\mathrm{Up}\colon\ \mathbb{A}_3 = (k[Z_1]/C_1)[X_2, X_3]/(C_2, C_3) \longrightarrow_{\mathrm{up}_1} (k[Z_2]/Q_2)[X_3]/(C_3) \longrightarrow_{\mathrm{up}_2} k[Z_3]/(Q_3)$$

with $n = 3$, $Z_1 = X_1$.

Algorithm - UnivRepNewtonSums

Input :

- $f \in k[T]$
- a primitive linear form $\Lambda := X_1 + \lambda_2 X_2 + \cdots + \lambda_n X_n$ of $\mathbb{A}$

Output :

- a univariate representation $\mathfrak{P}_i = (Q_i, S_1, \ldots, S_n)$ of $\mathbb{A}$.

Algorithm :

- Use **NewtonSums** to get for $2 \leqslant i \leqslant n$:
  - the minimal polynomials Q_i $\rightsquigarrow \mathcal{O}(2^n \,\mathrm{M}(\delta))$
  - the last parametrizations $S_{i,i}$ of $\mathbb{A}_i$ $\rightsquigarrow \mathcal{O}(2^n \,\mathrm{M}(\delta))$
- Get the other parametrizations: $S_i(Z_n) = \mathrm{Up}(X_i)$ $\rightsquigarrow \mathcal{O}(\mathrm{M}(n)\, n\, \mathrm{M}(\delta))$

Benchmarks

Advantage:

- Good timings:

Univariate representation of $\mathbb{A}_n$						
n		4	5	6	7	8
Time (sec)	Gröbner (F4) + FGLM	**0.001**	**0.03**	5.8	1500	>6h
	$\mathrm{NewtonSums}$	0.005	0.05	**0.52**	**6.8**	**100**

with MAGMA 2.17-1 over $k = \mathbb{F}_p$ with p prime number of 28 bits.

Drawbacks:

- Complexity of **UnivRepNewtonSums** not quasi-optimal: $\mathcal{O}(2^n\,\mathrm{M}(\delta))$.

- **NewtonSums** does not compute $\mathcal{X}_P$ for general $P \in \mathbb{A}$.

Outline of the talk

Newton sums methods:

i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in \mathbb{A}_m$

ii. Change of representation : Up and Down

iii. Univariate representation of $\mathbb{A}_m$

iv. Benchmarks

Resultant methods:

i. Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$

ii. Benchmarks

iii. Generalizations

Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}$

Based on the resultant approach to compute resolvents:

- $R_n := T - P(X_1, \ldots, X_n) \in k[X_1, \ldots, X_n, T]$
- For $i = n-1, \ldots, 0, \quad R_i := \mathrm{Res}_{X_{i+1}}(C_{i+1}, R_{i+1}) \in k[X_1, \ldots, X_i, T]$
- $\mathcal{X}_P = R_0 \in k[T]$.

Mathematically,

$$\mathrm{Res}_{X_{i+1}}: \quad \mathbb{A}_i[T][X_{i+1}] \times \mathbb{A}_i[T][X_{i+1}] \longrightarrow \mathbb{A}_i[T].$$

Multiplication in $\mathbb{A}_i$ in time $\tilde{\mathcal{O}}(\delta_m)$ + resultant algorithm on general ring

$\rightsquigarrow$ Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$ in time $\tilde{\mathcal{O}}(\delta_m)$

$\rightsquigarrow$ Not practical

Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}$

Algorithm - ResultantCharPol

Input : $P \in \mathbb{A}$

Output : $\mathcal{X}_{P,\mathbb{A}}$

Algorithm :

1. $G_n := \mathrm{Up}_{n-1}(Y - P)$ $\qquad G_n \in (k[Z_{n-1}]/Q_{n-1})[Y][Z_n]$
2. for $i = n-1 \ldots 1$ do

   $C'_{i+1} := \mathrm{Up}_i(C_{i+1})$ $\qquad C'_{i+1} \in (k[Z_i]/Q_i)[X_{i+1}][Y]$

   $G'_i := \mathrm{Res}_{X_{i+1}}(C'_{i+1}, G_{i+1})$ $\qquad G'_i \in (k[Z_i]/Q_i)[Y]$

   $G_i := \mathrm{down}_{i-1}(G'_i)$ $\qquad G_i \in (k[Z_{i-1}]/Q_{i-1})[Y][Z_i]$

3. return $G_i := \mathrm{down}_{i-1}(G'_i)$ $\qquad G_0 \in k[Y]$

Cost of **ResultantCharPol**: $\mathcal{O}(n^{(\omega+1)/2}\, n\, \mathrm{M}(\delta))$

Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}$

Algorithm - UnivRepResultant

Input :

- $f \in k[T]$;
- a primitive linear form $\Lambda := X_1 + \lambda_2 X_2 + \cdots + \lambda_n X_n$ of $\mathbb{A}$;

Output :

- a univariate representation $\mathfrak{P}_i = (Q_i, S_1, \ldots, S_n)$ of $\mathbb{A}$.

Algorithm :

- For $2 \leqslant i \leqslant n$, use **ResultantCharPol** to get :
  - the minimal polynomials Q_i $\mathcal{O}(n^{(\omega+1)/2} n \,\mathrm{M}(\delta))$
  - the last parametrizations $S_{i,i}$ of $\mathbb{A}_i$ $\mathcal{O}(n^{(\omega+1)/2} n \,\mathrm{M}(\delta))$
- Get the other parametrizations: $S_i(Z_n) = \mathrm{Up}(X_i)$ $\mathcal{O}(\mathrm{M}(n)\, n \,\mathrm{M}(\delta_n))$

Benchmarks

MAGMA 2.17-1 over $k = \mathbb{F}_p$ with p prime number of 28 bits.

Characteristic polynomial for any $P \in \mathbb{A}$						
	n	4	5	6	7	8
Time (sec)	Traces in $k[Z]/Q(Z)$ [SHOUP,'99]*	**0.001**	**0.01**	**0.23**	**6.8**	**200**
	ResultantCharPol*	0.03	0.24	2.6	46	1100

Characteristic polynomial for $P \in \mathbb{A}$ linear						
	n	4	5	6	7	8
Time (sec)	**NewtonSums**	0.001	0.015	0.12	**1.54**	**23**
	Traces in $k[Z]/Q(Z)$ [SHOUP,'99]*	**0.001**	**0.005**	**0.10**	2.9	83
	ResultantCharPol*	0.03	0.24	2.6	46	1100

(*) Requires the precomputation of a univariate representation

Benchmarks

MAGMA 2.17-1 on one core of a Intel Xeon @2.27GHz, 74Gb of RAM over $k = \mathbb{F}_p$ with p prime number of 28 bits.

n		5	6	7	8
Time	Up*	0.008	0.1	2	40
	Down*	0.01	0.1	1.4	25
	Univariate $\times$ *	40 μs	0.5 ms	0.006	0.06
	Univariate $\div$ *	0.002	0.03	0.29	4.5
(sec)	MAGMA triangular $\times$	0.003	0.09	4	170
	MAGMA triangular $\div$	0.1	28	>30 min	>6 h

(*) Requires the precomputation of a univariate representation

Outline of the talk

Newton sums methods:

i. Computation of $\mathcal{X}_\Lambda$ for a linear form $\Lambda \in \mathbb{A}_m$

ii. Change of representation : Up and Down

iii. Univariate representation of $\mathbb{A}_m$

iv. Benchmarks

Resultant methods:

i. Computation of $\mathcal{X}_P$ for any $P \in \mathbb{A}_m$

ii. Benchmarks

iii. Generalizations

Generalizations

Generalization: Adapt the situation to $G \subseteq \mathfrak{S}_n$.

Galoisian ideals:

$$\mathcal{I}_G = \{R \in k[X_1, \dots, X_n] \mid \forall \sigma \in G, R(\alpha_{\sigma(1)}, \dots, \alpha_{\sigma(n)}) = 0\}$$

Remark. If $G = \mathrm{Gal}(f)$, then $k[X_1, \dots, X_n]/\mathcal{I}_G$ is a decomposition field of f.

Triangular sets of Galoisian ideals:

- [Orange, Renault, Valibouze, '03]
- [Lederer, '04]
- [Renault, Yokoyama, '06 '08]
- [Orange, Renault, Yokoyama, '09]

Generalizations

From triangular sets to univariate representation:

- Resultant approach is general
- Up and Down still in good complexity, due to $\forall i, f(X_i) = 0$

Applications:

- Computation of relative resolvents

$$L_{P,G}(T) := \prod_{\sigma \in G//\mathrm{Stab}\, P} (T - P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(n)})) \in k[T],$$

where $P \in k[X_1, \ldots, X_n]^G$.

- Faster arithmetics in decomposition fields

Conclusion

Theoretical results:

- first quasi-linear algorithm for univariate representation in $\mathbb{A}_m$;
- first quasi-linear algorithm for characteristic polynomial in $\mathbb{A}_m$;
- Complexity improvement for the symbolic computation of absolute resolvents.

Practical results:

- MAGMA code;
- better timings for univariate representation of $\mathbb{A}$, Up, Down;
- as a result, better timings for arithmetic operations in $\mathbb{A}$, characteristic polynomial...

Thank you for your attention ;-)