An Introduction to Singular

Yang Zhang
University of Science and Technology of China

Sep 9, 2021

This is an abbreviated version of

the lecture notes

http://staff.ustc.edu.cn/~yzhphy/teaching/summer2021/CAG.pdf

mini courses taught in

Sagex 2021 winter school
HangZhou Amplitude Summer School

As theoretical physicists,
we frequently met computation with
polynomials, rational functions
and matrices with polynomials, rational functions …

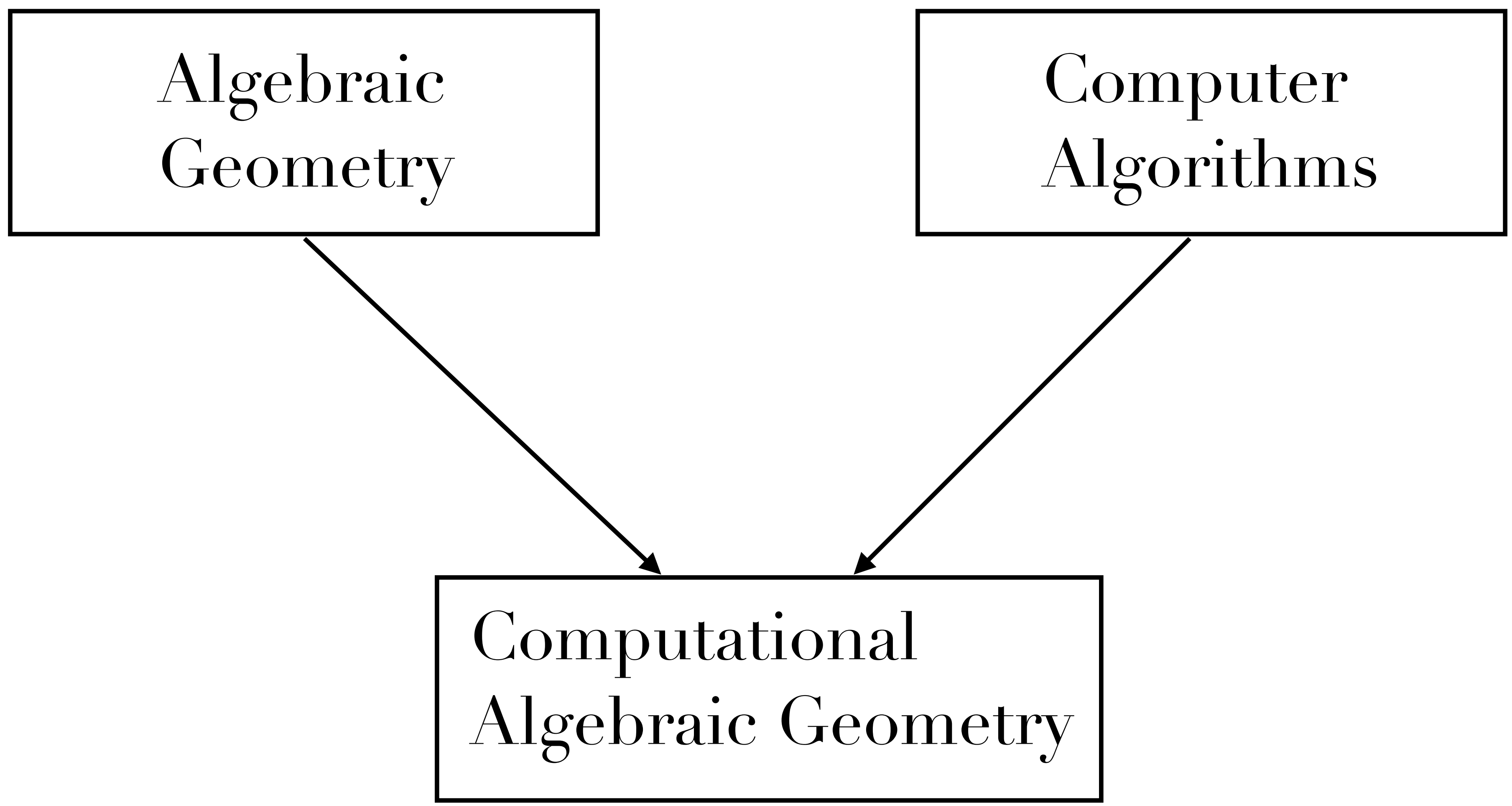
- Feynman integral reduction
- Differential equation for Feynman integrals
- Solve Bethe-Ansatz equation
- Find the minima of a super-potential …

It is not surprising that
in many cases, the polynomial/rational function computation
is the most time and RAM consuming step

in many cases, the polynomial/rational function part
is the longest component of an analytic result

The key to polynomial/rational function problems

is computational algebraic geometry



originated in 1970s
thrive from 2000s

Bruno Buchberger, Frank-Olaf Schreyer, Jean-Charles Faugère
David Eisenbud, Michael Stillman, Daniel Grayson, Wolfram Decker …

Overview

Concept

Basic commutative algebra
and algebraic geometry

Software

Mathematica, Maple
Singular, Macaulay2, Bertini

Reference

Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, *David A. Cox, Donal O'Shea, and John Little*

Using algebraic geometry, *David A. Cox, Donal O'Shea, and John Little*

A Singular Introduction to Commutative Algebra, *G. Pfister and Gert-Martin Greuel*

Algebraic geometry, *Robin Hartshorne*
Principles of Algebraic geometry, Phillip Griffiths and Joe Harris

---

Lecture Notes on Multi-loop Integral Reduction and Applied Algebraic Geometry
YZ, arXiv: 1612.02249

Concept: Polynomial ring and ideal

Polynomial ring $R = \mathbb{F}[x_1, \ldots, x_n]$

↑

field, $\mathbb{C}, \mathbb{Q}, \mathbb{Z}/p, \mathbb{Q}[c_1, \ldots c_m], \ldots$

An ideal I in the polynomial ring $R = \mathbb{F}[z_1, \ldots z_n]$ is a linear subspace of R such that, For $\forall f \in I$ and $\forall h \in R$, $hf \in I$.

The ideal in the polynomial ring generated by a polynomial set S is the collection of all such polynomials,

$$\sum_i h_i f_i, \quad h_i \in R, \quad f_i \in S.$$

This ideal is denoted as $\langle S \rangle$.

Concept: Monomial Ordering

Let M be the set of all monomials in the ring $R = \mathbb{F}[x_1, \ldots x_n]$. A monomial order $\prec$ of R is an ordering on M such that,

1. $\prec$ is a total ordering.
2. $\prec$ respects monomial products, i.e., if $u \prec v$ then for any $w \in M$, $uw \prec vw$.
3. $1 \prec u$, if $u \in M$ and u is not constant.

We use the convention $z_n \prec z_{n-1} \prec \ldots \prec z_1$ for all monomial orders. Given $g_1 = z_1^{\alpha_1} \ldots z_n^{\alpha_n}$ and $g_2 = z_1^{\beta_1} \ldots z_n^{\beta_n}$,

- Lexicographic order (*lex*). First compare α_1 and β_1. If $\alpha_1 < \beta_1$, then $g_1 \prec g_2$. If $\alpha_1 = \alpha_2$, we compare α_2 and β_2. Repeat this process the tie is broken.
- Degree lexicographic order (*grlex*). First compare the total degrees. If $\sum_{i=1}^{n} \alpha_i < \sum_{i=1}^{n} \beta_i$, then $g_1 \prec g_2$. If total degrees are equal, we compare (α_1, β_1), (α_2, β_2) ... until the tie is broken.
- Degree reversed lexicographic order (*grevlex*). First compare the total degrees. If $\sum_{i=1}^{n} \alpha_i < \sum_{i=1}^{n} \beta_i$, then $g_1 \prec g_2$. If total degrees are equal, we compare α_n and β_n. If $\alpha_n < \beta_n$, then $g_1 \succ g_2$ (reversed!). If $\alpha_n = \beta_n$, then we further compare $(\alpha_{n-1}, \beta_{n-1})$, $(\alpha_{n-2}, \beta_{n-2})$ ... until the tie is broken, and use the reversed result.
- Block order. We separate the variables into k blocks, say,

$$\{z_1, z_2, \ldots z_n\} = \{z_1, \ldots z_{s_1}\} \cup \{z_{s_1+1}, \ldots z_{s_2}\} \ldots \cup \{z_{s_{k-1}+1}, \ldots z_n\}.$$

  Define the monomial order in each block. To compare g_1 and g_2, first we compare the first block. If it is a tie, we compare the second block... until the tie is broken.

All monomials are sorted.

Concept: Polynomial division

Algorithm 2 Multivariate division algorithm

Input: F, $f_1 \dots f_k$, $\succ$
$q_1 := \dots := q_k = 0$, $r := 0$
while $F \neq 0$ **do**
  $reductionstatus := 0$
  for $i = 1$ to k **do**
    if $\mathrm{LT}(f_i) | \mathrm{LT}(F)$ **then**
      $q_i := q_i + \frac{\mathrm{LT}(F)}{\mathrm{LT}(f_i)}$
      $F := F - \frac{\mathrm{LT}(F)}{\mathrm{LT}(f_i)} f_i$
      $reductionstatus := 1$
      break
    end if
  end for
  if $reductionstatus = 0$ **then**
    $r := r + \mathrm{LT}(F)$
    $F := F - \mathrm{LT}(F)$
  end if
end while
return $q_1 \dots q_k$, r

Divide a polynomial over a set of polynomial

It seems that it can solve the ideal membership problem …
But it does not …

Concept: Groebner basis

For an ideal I in $\mathbb{F}[x_1, \ldots, x_n]$ with a monomial order, a Groebner basis $G(I) = \{g_1, \ldots g_m\}$ is a generating set for I such that for each $f \in I$, there always exists $g_i \in G(I)$ such that,

$$\mathrm{LT}(g_i) | \mathrm{LT}(f)\,.$$

invented by B. Buchburger, in the namesake of his supervisor, W.Groebner

- Polynomial division over a Groebner basis, provide a unique remainder, independent of the polynomial order.
- If $f \in I$, then the remainder of f over the Groebner basis is zero.
  Ideal membership problem is solved.
- The remainder provides a canonical representation of $F[x_1, \ldots, x_n]/I$.
- With a fixed monomial order, the reduced Groebner basis is unique.
  Ideal identification problem is solved.

Concept: Companion matrix

$$R/I = \text{span}\{b_1, \ldots b_k\}$$

We consider the linear representation of R/I,

$$[f \cdot b_i] = a_{ij}[b_j], \quad [f] \in R/I$$

companion matrix

If $p \in \mathcal{Z}(I)$, then $f(p)$ is an eigenvalue of the companion matrix $m_{[f]}$. On the other hand, any eigenvalue λ of m_f corresponds to some $p \in \mathcal{Z}(I)$ and $\lambda = f(p)$.

$$\text{tr} M_{[f]} = \sum_{p \in \mathcal{Z}(I)} f(p)$$

In physics, we frequently evaluate a function over a solution set
, and then take the sum.

Cachazo-Yuan-He Equation
Bethe Ansatz Equation

Concept: Modules

A module M over a ring $R = \mathbb{F}[x_1, \ldots, x_n]$ is an abelian group, such that

- $f(m_1 + m_2) = fm_1 + fm_2$, for $f \in R$ and $m_1, m_2 \in M$,
- $(f_1 + f_2)m = f_1 m + f_2 m$, for $f_1, f_2 \in R$ and $m \in M$,
- $(f_1 f_2)m = f_1(f_2)m$, for $f_1, f_2 \in R$ and $m \in M$,
- $1m = m$, for $1 \in R$, $m \in M$.

Clearly, R^m is a module. Any ideal of R is a module. We mainly consider a sub-module of R^m.

A module is an analogy of linear space, in algebraic geometry. The biggest difference is that for $m \in M$ and $f \in R$, $\frac{1}{f}m$ is not defined.

A basis of a module is a set $\{m_1, \ldots, m_k\}$ in M, such that $m_1, \ldots, m_k$ generate M, and if

$$f_1 m_1 + \ldots + f_k m_k = 0, \quad f_i \in R$$

then $f_1 = \ldots = f_k = 0$.

In most cases, a module does not have a basis. If it has, then such a module is a *free module*. R^m is a free module.

Concept: Syzygy

Consider $\{m_1, \ldots, m_k\}$ in a module M over R. All tuples $\{f_1, \ldots f_k\}$ such that

$$f_1 m_1 + \ldots + f_k m_k = 0, \quad f_i \in R$$

form the *syzygy* of $\{m_1, \ldots, m_k\}$. The syzygy is a sub-module of R^k.

If M is a sub-module of R^l, then each m_i can be written as a column vector with polynomial components. Define $A = \{m_1, \ldots, m_k\}$ as an $l \times k$ matrix, then the syzygy is,

$$\ker A \cap R^k$$

Syzygy can be understood as solving homogenous equations with polynomial solutions.

Concept: Lift

Lift computation can be understood as
an inhomogeneous version of syzygy computation

For $f_1, \ldots f_k, g \in R^l$, find a list of polynomials $\alpha_1, \ldots, \alpha_k$

$$\alpha_1 f_1 + \ldots + \alpha_n f_k = g$$

One solution of the lift problem, can be computed from the Groebner basis.

All solutions of the lift problem
can be obtained from one lift solution + syzygy

Singular

Singular

https://www.singular.uni-kl.de

A computer algebra system for polynomial-related problems

Open source (easy to install in Mac/Linux)

has been used for

- UT integral determination
- IBP reduction
- Spin chains
- N=4 Super-Yang-Mills

in physics

Warm-up: Interface to Singular in Mathematica

https://www3.risc.jku.at/research/combinat/software/Singular

Interface written by M. Kauers and V. Levandovskyy

```
Get["~/packages/singular_m/Singular.m"]

Singular -- Interface to Mathematica   Package by Manuel Kauers (mkauers@risc.uni-linz.ac.at) and Viktor Levandovskyy (levandov@risc.uni
http://www.risc.uni-linz.ac.at/research/combinat/software/Singular/ – © RISC Linz – V 0.11 (2008-04-18)
```

1. Install Singular (it is difficult to install Singular on Windows)
2. Modify Singular.m to get the binary path to Singular correct

SingularStd: Compute Groebner basis in Singular with Buchberger algorithm
SingularSlimgb: Compute Groebner basis in Singular with a better S-pair reduction strategy
SingularNF: Compute the remainder of a polynomial/module division
SingularSyz: Compute the syzygy
SingularLift: Compute the lift

SingularSlimgb can also be used as a linear equation solver with parameters for mid-size problems

Warm-up: Groebner basis computation in Singular

```
Cyclic6={x + y + z + t + u + v,
x*y + y*z + z*t + t*u + u*v + v*x-c,
x*y*z + y*z*t + z*t*u + t*u*v + u*v*x + v*x*y,
x*y*z*t + y*z*t*u + z*t*u*v + t*u*v*x + u*v*x*y + v*x*y*z,
x*y*z*t*u + y*z*t*u*v + z*t*u*v*x + t*u*v*x*y + u*v*x*y*z + v*x*y*z*t,
x*y*z*t*u*v - d};
```

```
AbsoluteTiming[Gr1=SingularStd[Cyclic6,{x,y,z,t,u,v},MonomialOrder→DegreeReverseLexicographic];]
```

```
{20.9969, Null}
```

```
AbsoluteTiming[Gr=GroebnerBasis[Cyclic6,{x,y,z,t,u,v},MonomialOrder→DegreeReverseLexicographic,CoefficientDomain→RationalFunct
```

```
{184.908, Null}
```

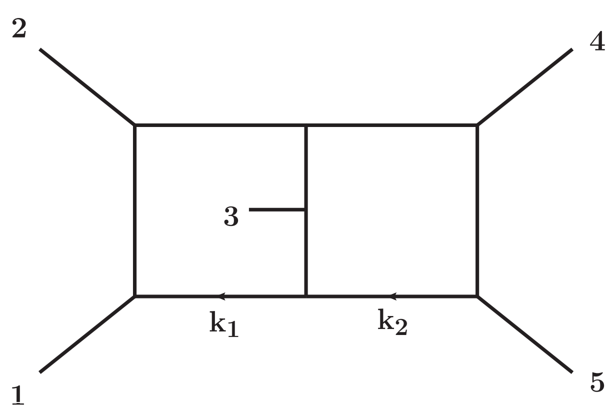
Warm-up: Lift, applications for Feynman integrals with uniformly transcendental weights

Consider the (D-dim) residues at the points ξ_α, $\alpha = 1, \ldots n$.

$$\sum_i \mathrm{Res}_{\xi_\alpha}(I_i) c_i = b_i$$

```
SingularLift[NormalizedResidueMatrix, {denFactor {1, 0, 0, -1, 0, 0, 0, 0}}, {s12, s23, s34, s45, s15}] // Factor
{{0, 4 s12 s23 (s12 - 2 s45), 4 s12 s34 s45, 4 s12 s15 (s12 - s45), 0, 0, -4 s12 s15, -4 s12 (s12 - s45), -4 s12 (s23 + s45), -4 s12 (s12 - s34)}}
```

Solved in seconds with Singular interface,
all UT in this sector obtained



"*All master integrals for three-jet production at NNLO*", PhysRevLett. 123 (2019), no. 4 041603
Chicherin, Gehrmann, Henn, Wasser, **YZ**, Zoia

Singular programing

Singular language has a C-like syntax

Run a Singular code as a script

Easy to parallelise

A first example

reduce the “tail” during
the Groebner basis computation

show the progress

Ring definition

```
option(redSB);
option(redTail);
option(prot);
//ring r=0,(s0,s1,s2),dp;
ring r=0,(s0,s1,s2),(dp(2),dp(1));
//ring r=42013,(s0,s1,s2),dp;
ideal I=18*(s0 - s2 - 3*s1*s2 + 2*s2^3), 384*(-3 + 3*s1 - s2^2)*(-s0 + s2 + 3*s1*s2 - 2*s2^3), 64*(s0 - s2 - 3*s1*s2 + 2*s2^
3)*(27*s0 + 6*s2 - 15*s1*s2 + 4*s2^3), -160*s0 + 2304*s0^3 + 672*s0*s1 + 768*s0*s1^2 - 1280*s0*s1^3 + s2 - 256*s0^2*s2 - 819
2*s0^2*s1*s2 - 2304*s0*s1*s2^2 + 5376*s0*s1^2*s2^2 + 3840*s0^2*s2^3 + 768*s0*s2^4 - 3328*s0*s1*s2^4 + 512*s0*s2^6, -13 + 102
4*s0^2 - 144*s1 + 5888*s0^2*s1 + 672*s1^2 + 768*s1^3 - 1280*s1^4 + 1280*s0*s1*s2 - 11264*s0*s1^2*s2 - 3328*s0^2*s2^2 - 2304*
s1^2*s2^2 + 5376*s1^3*s2^2 - 768*s0*s2^3 + 6656*s0*s1*s2^3 + 768*s1*s2^4 - 3328*s1^2*s2^4 - 512*s0*s2^5 + 512*s1*s2^6, 16*(-
14*s0 + 16*s0*s1 + 304*s0*s1^2 - 15*s2 + 320*s0^2*s2 + 42*s1*s2 + 144*s1^2*s2 - 272*s1^3*s2 + 32*s0*s2^2 - 864*s0*s1*s2^2 -
192*s1*s2^3 + 512*s1^2*s2^3 + 272*s0*s2^4 + 48*s2^5 - 240*s1*s2^5 + 32*s2^7);
ideal gb=std(I);
write(":w gb_6_3.txt",string(gb));
//write("ssi:w gb_6_3.ssi",gb);
exit;
```

Groebner basis computation

Output file: .ssi is the Singular format (fast)
.txt is readable by other softwares (Maple, Mathematica)

C-like features

Library for matrix

```
LIB "matrix.lib";
option(redSB);
option(redTail);
option(prot);
ring r=0,(s0,s1,s2),(dp(2),dp(1));
poly varprod=s0*s1*s2;
proc coeflist(poly p, poly varp, ideal mlist) // Find the coefficient of monomial list "mlist" in p
  {
        int i,j;
        list clist;
        matrix matrixA=coef(p,varp);
        for(i=1;i<=size(mlist);i++)
                {
                        clist[i]=0;
                        for(j=1;j<=ncols(matrixA);j++)
                                {
                                 if(matrixA[1,j]==mlist[i])                                                       {
                                                clist[i]=matrixA[2,j];
}
                                }


   }
        return(clist);
 }

ideal I=18*(s0 - s2 - 3*s1*s2 + 2*s2^3), 384*(-3 + 3*s1 - s2^2)*(-s0 + s2 + 3*s1*s2 - 2*s2^3), 64*(s0 - s2 - 3*s1*s2 + 2*s2^
3)*(27*s0 + 6*s2 - 15*s1*s2 + 4*s2^3), -160*s0 + 2304*s0^3 + 672*s0*s1 + 768*s0*s1^2 - 1280*s0*s1^3 + s2 - 256*s0^2*s2 - 819
2*s0^2*s1*s2 - 2304*s0*s1*s2^2 + 5376*s0*s1^2*s2^2 + 3840*s0^2*s2^3 + 768*s0*s2^4 - 3328*s0*s1*s2^4 + 512*s0*s2^6, -13 + 102
4*s0^2 - 144*s1 + 5888*s0^2*s1 + 672*s1^2 + 768*s1^3 - 1280*s1^4 + 1280*s0*s1*s2 - 11264*s0*s1^2*s2 - 3328*s0^2*s2^2 - 2304*
s1^2*s2^2 + 5376*s1^3*s2^2 - 768*s0*s2^3 + 6656*s0*s1*s2^3 + 768*s1*s2^4 - 3328*s1^2*s2^4 - 512*s0*s2^5 + 512*s1*s2^6, 16*(-
14*s0 + 16*s0*s1 + 304*s0*s1^2 - 15*s2 + 320*s0^2*s2 + 42*s1*s2 + 144*s1^2*s2 - 272*s1^3*s2 + 32*s0*s2^2 - 864*s0*s1*s2^2 -
192*s1*s2^3 + 512*s1^2*s2^3 + 272*s0*s2^4 + 48*s2^5 - 240*s1*s2^5 + 32*s2^7);
ideal gb=std(I);
write(":w gb_6_3.txt",string(gb));
//write("ssi:w gb_6_3.ssi",gb);
int i;
for(i=1;i<=size(gb);i++)
        {
                if(variables(gb[i])==s2)
                {
                        print(gb[i]);
                }
        }
print(coeflist(gb[1],varprod,s2^7));
exit;
```

A simple function to find the monomial coefficient

“for” loop for finding generators in s2 only

With algebraic numbers

```
ring R=(0,r),(x,y,z),dp;
minpoly=r^2+1;
ideal I=x^2+y^2,z^2-x^3-r*x-r,z^2-x*y;
ideal gb=std(I);
```

r is the imaginary unit

Several algebraic numbers ? Find the primitive generator

$$a = \sqrt{2}, b = \sqrt{3} \quad \Rightarrow \quad r^4 - 20r^2 + 16 = 0$$

$$a = \frac{1}{8}(-16r + r^3), \quad b = \frac{1}{8}(24r - r^3)$$

Here r means $\sqrt{2} + \sqrt{3}$, the primitive element

Heavy computation, progress

progress indicator

read an input file (an ideal)

```
option(redSB);
option(redTail);
option(prot);
ring r=0,(s0,s1,s2,s3,s4),dp;
execute("ideal I="+read("ideal_15_5.txt")+";");
ideal gb=std(I);
write(":w gb_15_5.txt",string(gb));
exit;
```

number of unreduced S-paris

```
[1048575:3]6(14)s8(13)-9---12-s13(8)s14s(9)-s(8)s(10)-ss(12)s15(15)s(16)s(17)s(20)s(22)s16(25)-s--s(24)s(25)-s(27)s(30)s(31)s(33
)s(34)s(31)s(33)s(35)s(38)s17(41)s(43)s(46)s(47)-s(48)s(50)s(52)-s(54)s(57)s(58)s(61)s(64)s(68)s(70)s(73)s(76)s(78)s(81)s(80)s(8
3)s(85)s18(86)s(87)s(88)s(87)s(88)s(91)s(92)s(95)ss(96)-s(98)--s-s(99)s(101)s(104)s(107)s(110)--s(111)s(114)s(117)s(118)s(122)s(
124)19-ss(125)---s(122)--s(123)s(125)s(127)s(131)s(133)s(134)s(137)s(140)---ss(143)--s(144)s(147)s(149)s(151)s(154)s(157)s(160)s
(161)s(163)s(165)s(168)s(171)s(173)s(176)s(178)-s(179)----s(177)s(180)-s(182)---s(181)s(184)s(187)s(189)s(190)-s(191)s(194)s(198
)s(199)s(202)s(205)s(207)s-s(209)s(212)--s(213)s(215)s(219)s(222)-s(224)s(226)s(229)s(232)--s(233)s(236)s(239)--s(240)-s20(242)-
----------s(233)s(235)s(237)---s(234)-------s(230)---ss(232)----s(231)-s(232)--s--s---s(231)s(234)s(237)---s(236)-s(239)-s(241)s(2
44)s(246)s(249)----------s(242)--s(243)-s(244)----ss(247)s(250)s(252)-s(254)s(258)--s(259)-s(260)---s-s(262)s(265)s(268)s(271)-s(
272)-s(274)---ss(277)s(280)s(283)s(286)s(288)--s--s(289)-s(290)s(293)-s(295)s(298)s(301)-(300)---s-s(302)s(306)--s(307)-s(309)s(
312)s(315)s(318)s(321)s(324)s(327)s(329)s(332)s(334)-s(336)s(339)s(342)s(345)s21(348)--s-ss(350)--s---s(349)------------s(341)s(3
44)s(347)-s(349)s(351)--s--s-------s(348)----ss(350)s(352)--s--ss(354)------s(349)s(351)s(354)s(357)-s(359)s(362)-s(363)-s(364)s(3
66)s(369)---------s(364)-------s(361)s(364)-----------------s(350)s(354)s(356)------s(354)s(357)-s(358)---ss(361)s(363)--s(364)--s(3
65)s(368)s(372)s(375)s(377)s(380)s(383)s(386)s(388)s(391)---s--s(392)s(395)s(398)s(401)s(403)s(406)s(409)-----s(408)s(411)-----s(4
10)s(412)--s(413)--s(414)s(417)s(420)s(423)s(425)s(427)s(430)-s(432)--s(433)s(436)s(439)s(442)-s(444)s(447)s(450)s(452)s(455)s(4
58)s(461)-s(463)s(465)s(468)s(471)s(474)s(476)-s(478)-s(480)s(483)s(485)s(488)s(491)s(494)s22(497)---s-s(498)--ss(501)s(503)---(
500)--------------------s(482)---------s(476)---ss(479)s(482)s(485)s(488)---------s(482)-----------s(475)--s(476)-s(477)--s(478)--s
(479)s(482)s(485)s(488)s(491)s(494)s(497)--------s(493)--ss(495)-------s(491)-s(492)s(494)-s(496)-----s(495)s(498)s(501)s(504)s(507
)s(510)s(513)s(515)------s(513)-------------(500)-----------s(493)--s(494)----s(493)-s(495)----------s(489)s(492)-s(494)---s(495)--
-ss(498)------s(495)s(498)-----s(495)-s(497)-s(499)s(502)-s(504)s(507)s(509)-s(511)--s(512)---ss(515)-s(517)-s(519)-s(521)s(523)
s(525)s(528)s(531)-s(533)-s(535)-s(537)s(540)s(543)s(546)------s(544)s(546)-----s(545)--s(546)---s--s(547)s(550)s(553)s(556)s(559)
s(562)s(565)s(568)s(570)s(573)-s(575)s(578)-s(580)--s(581)------s(579)s(582)s(584)-s(586)s(589)s(591)s(594)s23(597)---------------
-----s(581)-s(583)--------------------s(566)--s(567)----------s(561)----------------s(549)-----s(548)------s(545)-s(547)s(550)-s(552)-
--s---s-s(554)s(557)s(560)s(563)-s(565)s(568)s(571)s(574)s(577)--s------s(574)-s(576)s(580)s(583)s(586)s(589)s(592)s(595)s(599)s(
```

Finite-field methods

library for finite-field lift

```
LIB "modstd.lib"
option(redSB);
option(redTail);
option(prot);
ring r=0,(s0,s1,s2,s3,s4),dp;
execute("ideal I="+read("ideal_15_5.txt")+";");
ideal gb=modStd(I,0);
write(":w gb_15_5.txt",string(gb));
exit;
```

Use multiple prime numbers to compute Groebner basis

Automatically parallelized

0 means to get the result with high probability

1 means to check the result over Q definitively

Module

position over term
first entry first

position over term
last entry first

```
option(redSB);
option(redTail);
option(prot);

ring r=0,(x,y,z),(c,dp);
//ring r=0,(x,y,z),(C,dp);
//ring r=0,(x,y,z),(dp,c);
//ring r=0,(x,y,z),(dp,C);

vector v1=x*gen(1) +x*gen(2);
vector v2=y*gen(1)+gen(3);
vector v3=(x^2-y)*gen(2)+z*gen(3);
module M=v1,v2,v3;
std(M);
```

term over position
last entry first

term over position
first entry first

```
1(2)ss2ss3s4s
(S:5)------
product criterion:0 chain criterion:1
_[1]=[0,0,x3+xyz-xy]
_[2]=[0,y2,-x2-yz]
_[3]=[0,xy,-x]
_[4]=[0,x2-y,z]
_[5]=[y,0,1]
_[6]=[x,x]
```

to eliminate the first entry

Syzygy

```
option(redSB);
option(redTail);
option(prot);

ring r=0,(x,y,z),(dp,c);


vector v1=x*gen(1) +y*gen(3);
vector v2=y*gen(1)+gen(3);
vector v3=(y^2-1)*gen(2)+z*gen(3);
vector v4=z*gen(1)+(y+1)*gen(3);
vector v5=x*gen(1)+z*gen(2);
module M=v1,v2,v3,v4,v5;
syz(M);
```

Compute the syzygy of the five vectors

Find three syzygy generators (not two)

```
{3}std:1(4)s(3)s(2)ss2(3)s(2)sss3s(3)s(2)4-ss5s
(S:12)-------------
product criterion:0 chain criterion:2
_[1]=xy*gen(2)-y2*gen(1)+y2*gen(4)-yz*gen(2)+x*gen(2)-x*gen(4)-y*gen(1)+z*gen(1)
_[2]=xy2*gen(1)-xy2*gen(4)-xy2*gen(5)+y3*gen(1)-y3*gen(4)-y3*gen(5)+y2z*gen(2)+y2z*gen(5)-xz2*gen(2)+yz2*gen(1)-yz2*gen(4)+z3*ge
n(2)+xy*gen(4)+y2*gen(4)+xz*gen(3)-yz*gen(1)-yz*gen(2)+yz*gen(3)-z2*gen(3)-x*gen(1)+x*gen(5)-y*gen(1)+y*gen(5)+z*gen(1)-z*gen(5)
_[3]=y4*gen(1)-y4*gen(4)-y4*gen(5)+y3z*gen(2)+y3*gen(1)-y3*gen(5)-y2z*gen(1)+y2z*gen(3)+y2z*gen(5)-yz2*gen(4)+z3*gen(2)-y2*gen(1
)+y2*gen(4)+y2*gen(5)-yz*gen(2)+yz*gen(3)-z2*gen(3)-y*gen(1)+y*gen(5)+z*gen(1)-z*gen(5)
```

Here "gen(i)" means the i-th original vector

Lift

A new vector

```
option(redSB);
option(redTail);
option(prot);

ring r=0,(x,y,z),(dp,c);


vector v1=x*gen(1) +y*gen(3);
vector v2=y*gen(1)+gen(3);
vector v3=(y^2-1)*gen(2)+z*gen(3);
vector v4=z*gen(1)+(y+1)*gen(3);
vector v5=x*gen(1)+z*gen(2);
module M=v1,v2,v3,v4,v5;

vector v=z^2*gen(2)+z^2*gen(3)-z*gen(3);



lift(M,v);
```

Try to express the new vector as the linear combination of these vector with polynomial coefficients.

```
_[1,1]=y2-z-1
_[2,1]=yz-z
_[3,1]=z
_[4,1]=-y2+y
_[5,1]=-y2+z+1
```

five polynomial coefficients.

Vielen Dank
und
Auf Wiedersehen