Computing the general discriminant formula of degree 17

Kinji Kimura(Kyoto University)

The world record of π is 10,000,000,000,050 decimal places computed by Shigeru Kondo(Japanese).

Our aim

Developing
superfast **Linear Algebra PROGramas in Computer Algebra**(LAPROGCA)

Now, some programs are available:
Linear solver, Characteristic polynomial, Minimal polynomial, Determinant and Resultant

Download site

URL:

http://www-is.amp.i.kyoto-u.ac.jp/kkimur/index-e.html

Strategy

Deterministic algorithm or Las Vegas algorithm, **Parallel**, Chinese remainder theorem and Interpolation

We adopt the following theorem in order to achieve deterministic algorithms.

A bound of coefficients for determinants with polynomial entries

The Goldstein and Graham bound

$$\|\det(A)\|_2 \leq \min\left\{ \prod_{i=1}^{N} \sqrt{\sum_{j=1}^{N} \|a_{i,j}\|_1^2}, \prod_{j=1}^{N} \sqrt{\sum_{i=1}^{N} \|a_{i,j}\|_1^2} \right\},$$

for a matrix A

Details

Linear solver:dense, finite field($\mathbb{Z}/p\mathbb{Z}$)

Characteristic polynomial and Minimal polynomial:dense or sparse, integer($\mathbb{Z}$)

Determinant:polynomial entries

Resultant:polynomial coefficients

Computing environment 1

CPU:Intel Core i7 980X 3.33GHz

1CPU(6Core)

Mem:24Gbyte, OS:Fedora 13, CC:gcc 4.4.5

LIB:GotoBLAS2-1.13

Evaluation of performance 1

LU decomposition:dense, random,
30000x30000

GotoBLAS2-1.13 (DGETRF, double precision):
242.388sec

Our Program($\mathbb{Z}/p\mathbb{Z}$,$p = 273967$):
248.52sec

Evaluation of performance 2

Characteristic polynomial:2000x2000, dense, integer, random, $a(i,j) \in [1,10]$

LinBox 1.1.6(examples/charpoly.C):
3125.377sec

Our Program:
483.810sec

Sparse linear solver in LAPROGCA

In the near future, LAPROGCA will supports a sparse linear solver. By virtue of a **nested dissection ordering**, a problem to solve sparse linear equations can be transformed into a lot of **LU decomposition for small dense matrices** and matrix multiplication.

Today's main topic

Parallel computation of determinants of matrices with polynomial entries and resultants with polynomial coefficients

Benchmark problem

A definition of discriminant

Using the Sylvester matrix, we define the resultant as follows,

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_0,$$

$$g(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_0,$$

$$\text{Sylvester}(f(x), g(x)) =$$

$$\begin{pmatrix}
a_m & a_{m-1} & \cdots & a_0 & & & \\
 & a_m & a_{m-1} & \cdots & a_0 & & \\
 & & \ddots & \ddots & \ddots & \ddots & \\
 & & & a_m & a_{m-1} & \cdots & a_0 \\
b_n & b_{n-1} & \cdots & \cdots & b_0 & & \\
 & b_n & b_{n-1} & \cdots & \cdots & b_0 & \\
 & & \ddots & \ddots & \ddots & \ddots & \\
 & & & b_n & b_{n-1} & \cdots & \cdots & b_0
\end{pmatrix},$$

$$\text{resultant}(f(x), g(x)) = \det(\text{Sylvester}(f(x), g(x))).$$

Then, we can define the discriminant,

$$\text{discriminant}(f(x)) = (-1)^{\frac{1}{2}m(m-1)}\frac{1}{a_m}\text{resultant}(f(x), f'(x)),$$

where $f(x)$ is a polynomial of degree m. We try to compute the general discriminant formula for $f(x)$,

$$f(x) = a_{17}x^{17} + a_{16}x^{16} + \cdots + a_0.$$

of degree 17 by using a **modified multivariate Newton interpolation**. After some discussion, without loss of generality, we can assume $a_{17} = 1$ and $a_{16} = 1$.

Computing environment 2

We use a **super computer(Subsystem C)** made by Cray Inc.

Specifications	Machine	Appro 2548X
	Number of Nodes	16
	Theoretical Peak Performance	10.6 teraflops
	Total Memory Capacity	24 TB
	Network Topology	Fat tree
	Bisection Bandwidth	217GB/s (Full-bisection)
Node Specifications	Processor (Core)	4 (4x8 = 32)
	Theoretical Peak Performance	665.6 gigaflops
	Memory	DDR3-1066 1.5TB
	Interconnect	InfiniBand FDR x 2
Processor Specifications	Processor	Intel Xeon E5
	Architecture	x86-64
	Clock	2.6 GHz
	Number of Cores	8
	Theoretical Peak Performance	166.4 gigaflops

Algorithm

We use CRT and a **modified multivariate Newton interpolation**. Our approach is not probabilistic but deterministic. The algorithm can be also used for the purpose of computing determinants with polynomials and resultants in general case.

Result

We show the number of terms in each degree.

degree	the number of terms	polynomials
2	2	$a_1^2 - 4a_2a_0$
3	5	$a_2^2a_1^2 - 4a_3a_1^3 - 4a_2^3a_0 - 27a_3^2a_0^2 + 18a_3a_2a_1a_0$
4	16	too long
5	59	too long
6	246	too long
7	1103	too long
8	5247	too long
9	26059	too long
10	133881	too long
11	706799	too long
12	3815311	too long
13	20979619	too long
14	117178725	too long
15	663316190	too long
16	3798697446	too long
17	**21976689397**	too long

A modified multivariate Newton interpolation

Target:

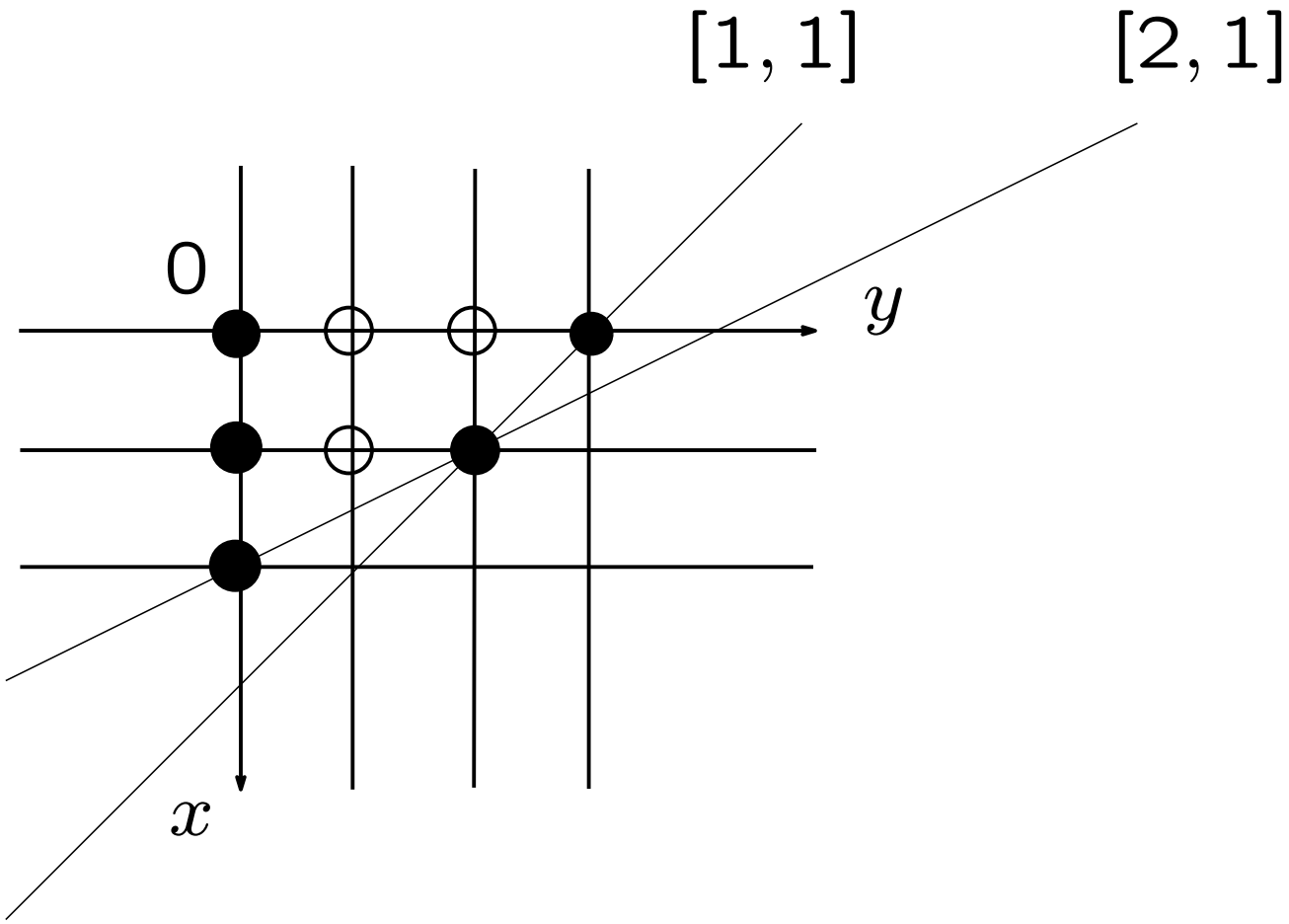
$$8 - 2xy^2 + 3x - 7y^3 + 4x^2$$

The number of evaluation points:

$$(2+1) \times (3+1) = 12$$

This calculation is based on misunderstanding for a multivariate Newton interpolation!

Weighted total degree bounds



We need 8 evaluation points!

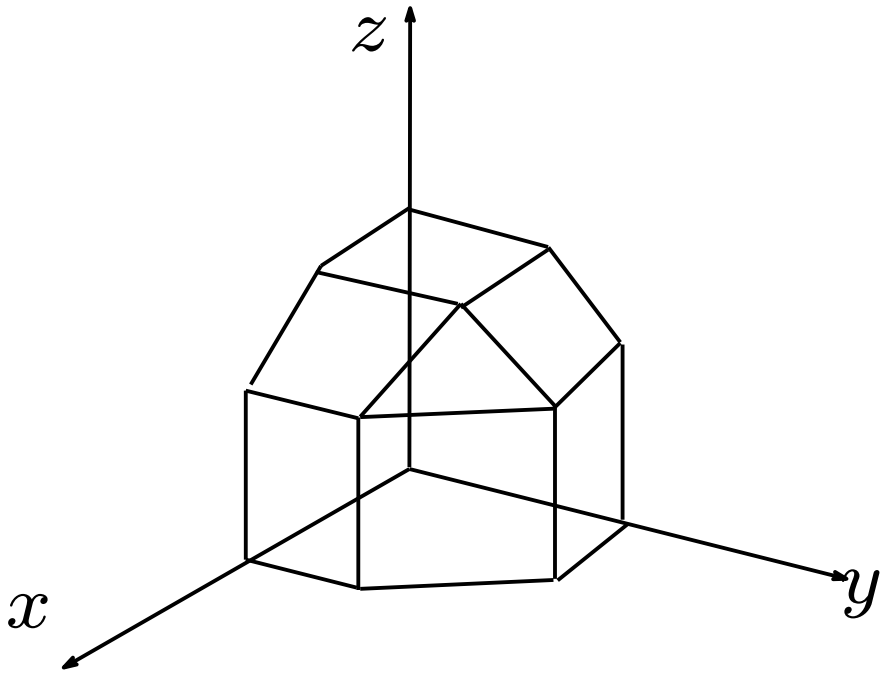
In the case of 3 dim.

$$A = \begin{vmatrix} x+y+z & xy \\ 2 & xyz \end{vmatrix}$$

weight(x, y, z)=[1,1,1]

variables	parameters	(partial) total degree
x	y, z	2
y	x, z	2
z	x, y	2
x, y	z	3
y, z	x	3
z, x	y	3
x, y, z		4

From the table, we can assume the expanded form of A.



We use another weight.

weight(x, y, z)=[1,2,3]

variables	parameters	(partial) total degree
x	y, z	2
y	x, z	4
z	x, y	6
x, y	z	5
y, z	x	8
z, x	y	7
x, y, z		9

It is difficult to figure a Newton polytope.

Weights for computing the general discriminant formula of degree 17

$$f(x) = x^{17} + x^{16} + a_{15}x^{15} + a_{14}x^{14} + a_{13}x^{13}$$
$$+a_{12}x^{12} + a_{11}x^{11} + a_{10}x^{10} + a_9x^9 + a_8x^8 + a_7x^7$$
$$+a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0.$$

We use two types of weights.

	a_{15}	a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
weight_1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
weight_2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	17

How to get nearly optimal weights

cf. Kinji Kimura and Masayuki Noro, Automatic weight generator for the Buchberger algorithm, In Proceedings of International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS2006), (2006/7/24), 33-44.

We can use **same ideas** in this paper to get nearly optimal weights for computing determinants and resultants.

How to get weighted total degree bounds for determinants

linear assignment problem:LAP

Say you have three workers: Jim, Steve and Alan. You need to have one of them clean the bathroom, another sweep the floors & the third wash the windows. What's the best (minimum-cost) way to assign the jobs? First we need a matrix of the costs of the workers doing the jobs.

	Clean bathroom	Sweep floors	Wash windows
Jim	\$1	\$2	\$3
Steve	\$3	\$3	\$3
Alan	\$3	\$3	\$2

If we change "minimum-cost" to "maximum-cost", that is the (weighted) total degree bounds for determinants.

Sylvester determinant vs. Bézout determinant

In order to get the bounds of degrees and coefficients for computing resultants, we need the determinant representation of the resultants.

We have two elections:Sylvester determinant vs. Bézout determinant.

If you want more tighter bounds, Bézout determinant is more suitable.

The detail of the modified multivariate Newton interpolation

From an input,

$$B = \begin{vmatrix} \alpha + \beta & 2 \\ 3 & \alpha\beta \end{vmatrix},$$

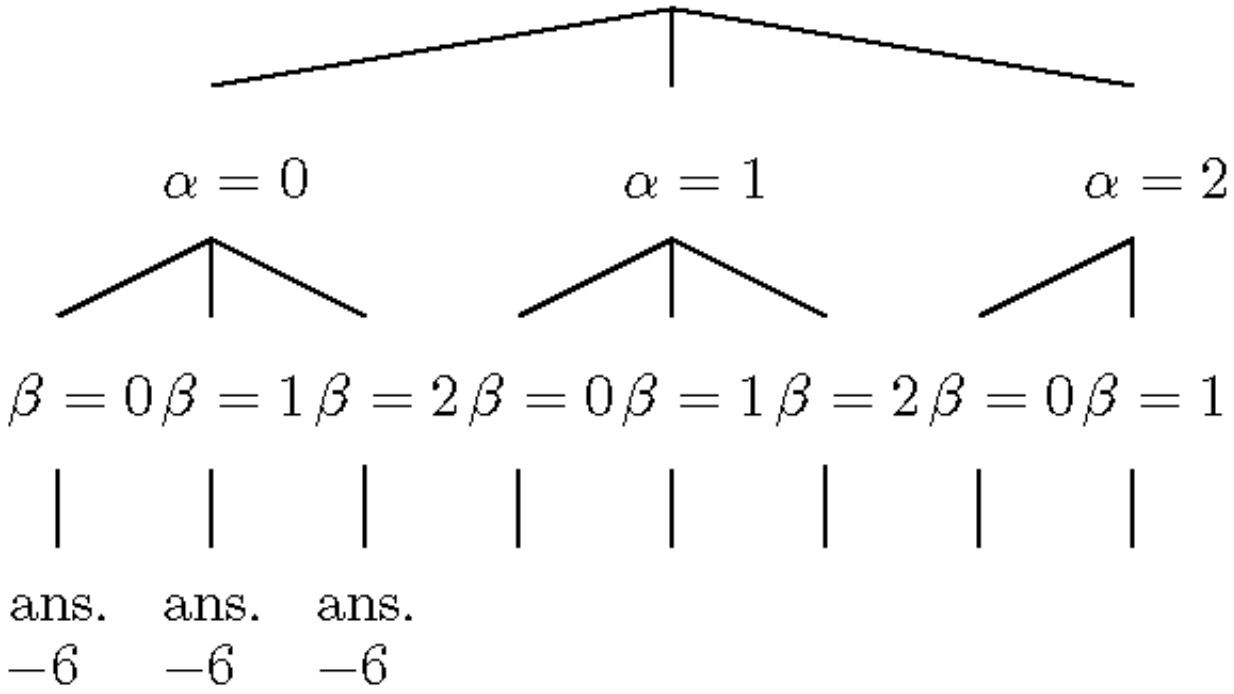
we can assume the expanded form of B,

$$(c_0 + c_1\beta + c_2\beta(\beta - 1)) + \alpha(c_3 + c_4\beta + c_5\beta(\beta - 1))$$
$$+\alpha(\alpha - 1)(c_6 + c_7\beta + c_8\beta(\beta - 1)),$$
$$c_8 = 0,$$

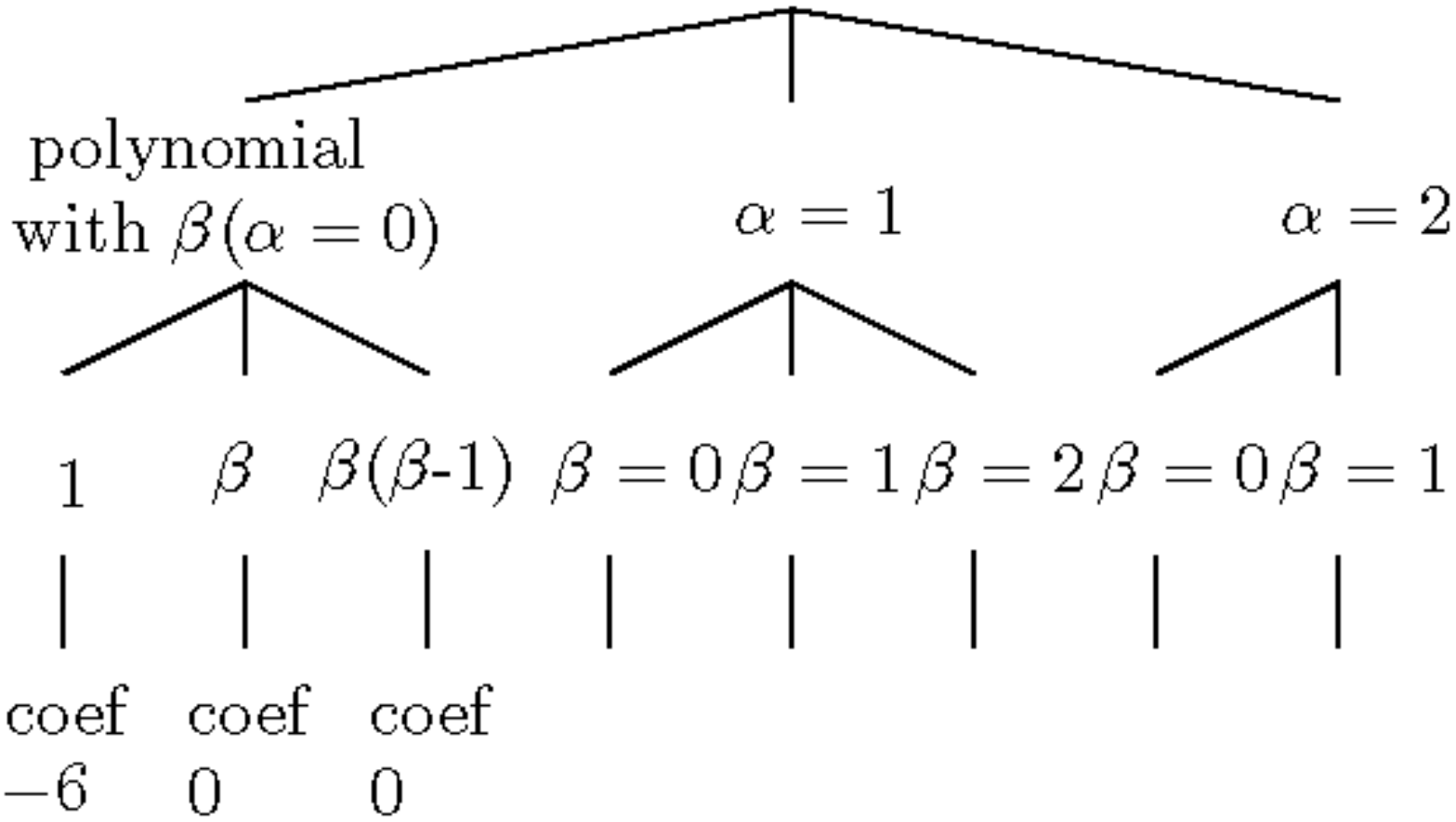
by using weighted total degree bounds. $c_0, \cdots, c_6$ and c_7 are unknown numbers.

In this slide, to make clearly understandable, we compute in $\mathbb{Q}$. However, in computers, we use $\mathbb{Z}/p\mathbb{Z}$ and Chinese remainder theorem.

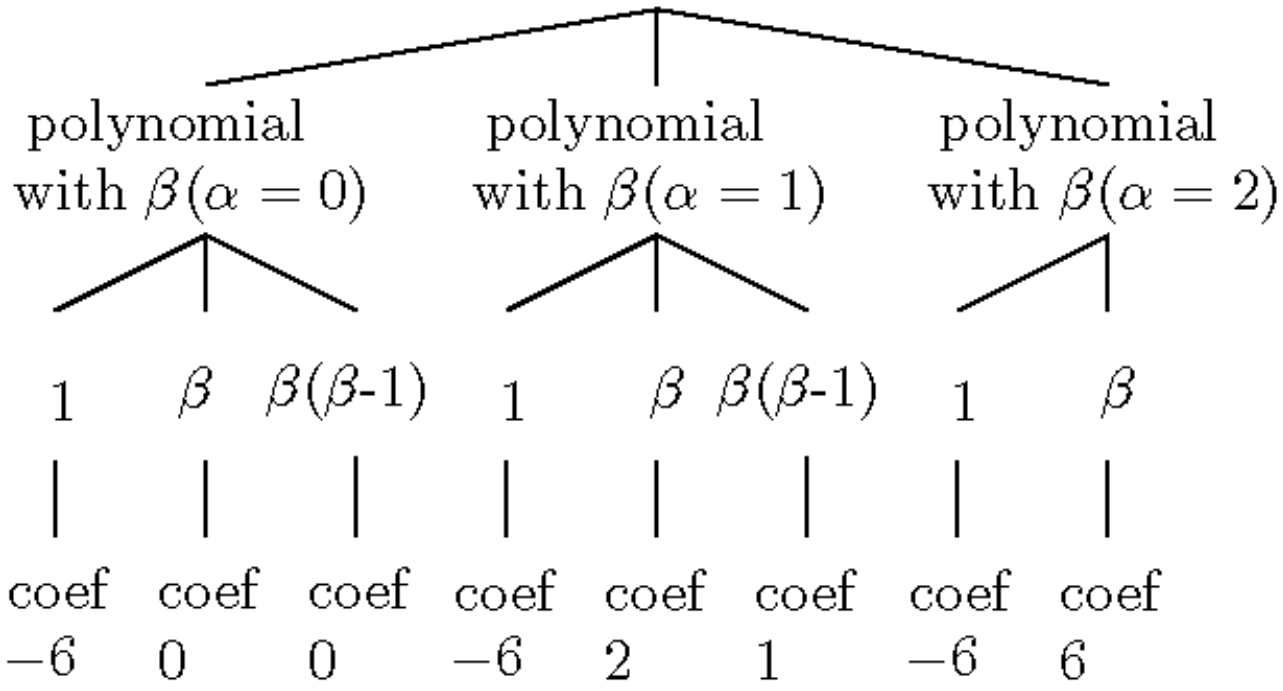
Firstly, we compute the values of the determinant at the points of $\alpha = 0$, $\beta = 0, 1, 2$,



Next, we interpolate at the points of $\alpha = 0$,



Similarly, we compute at the points of $\alpha = 1$ and $\alpha = 2$.

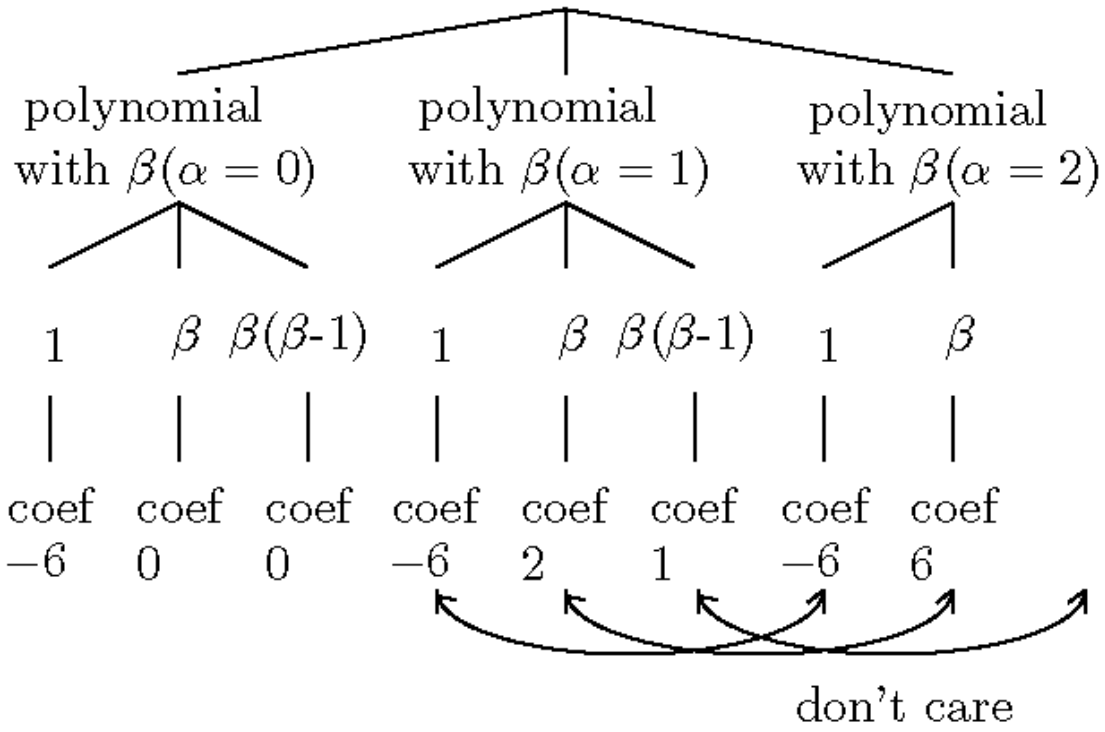


We don't compute the polynomial with β at the points of $\alpha = 2$ exactly. Because Newton interpolation is a sequentially interpolation, if the polynomial with β at the points of $\alpha = 2$ is not exact, finally, we can get exact solutions for the expanded form of B.

We regard polynomials as the data and interpolate.

We subtract the polynomial($\alpha = 1$)

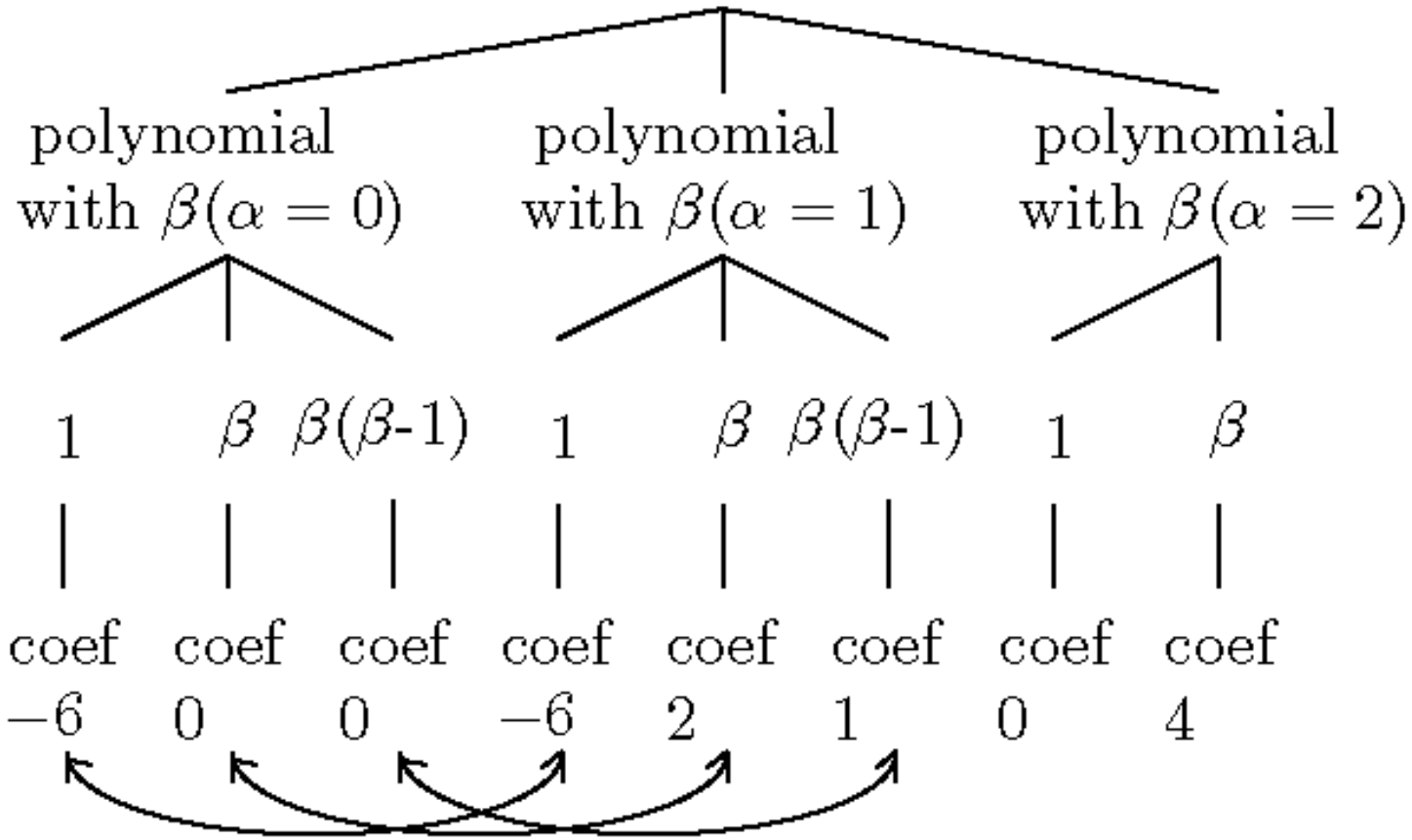
from the polynomial($\alpha = 2$) and store the result to the place of the polynomial($\alpha = 2$).



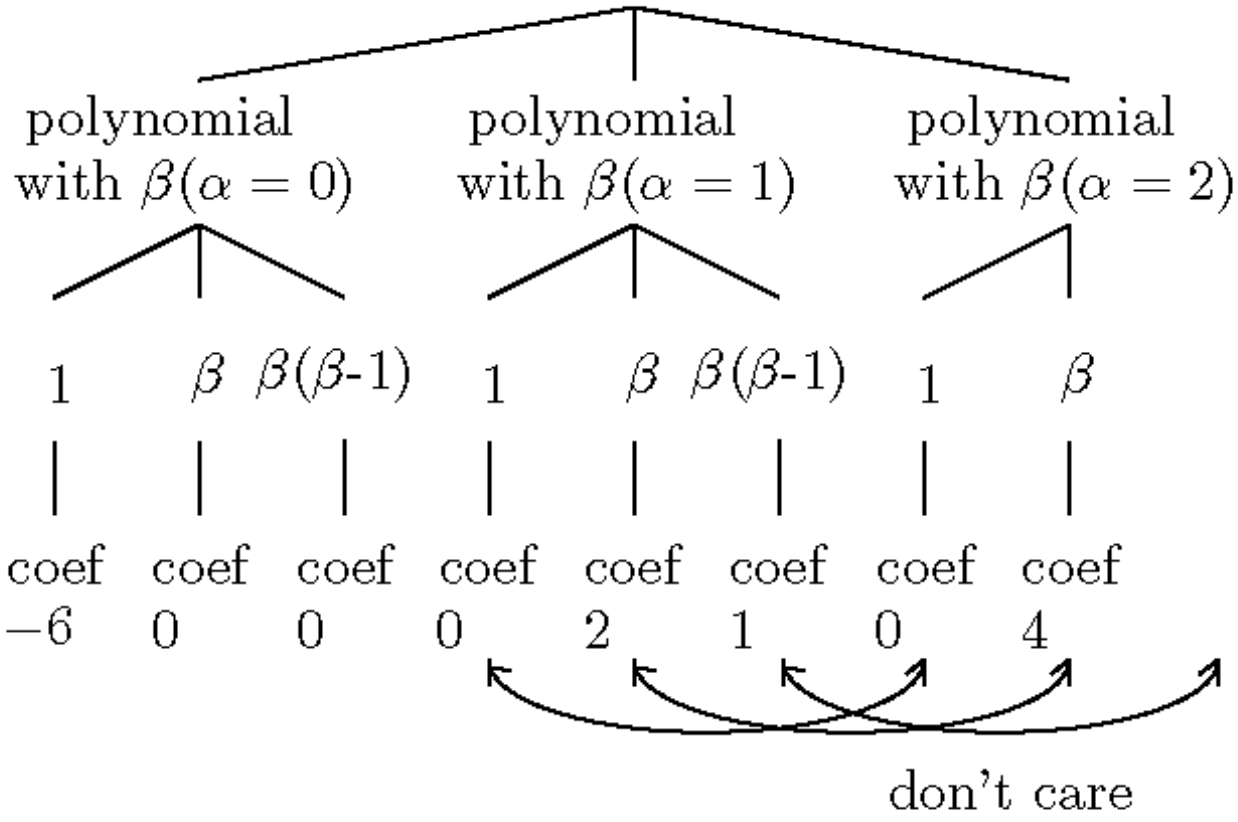
We don't care the coefficient of $\beta(\beta - 1)$

We subtract the polynomial($\alpha = 0$)

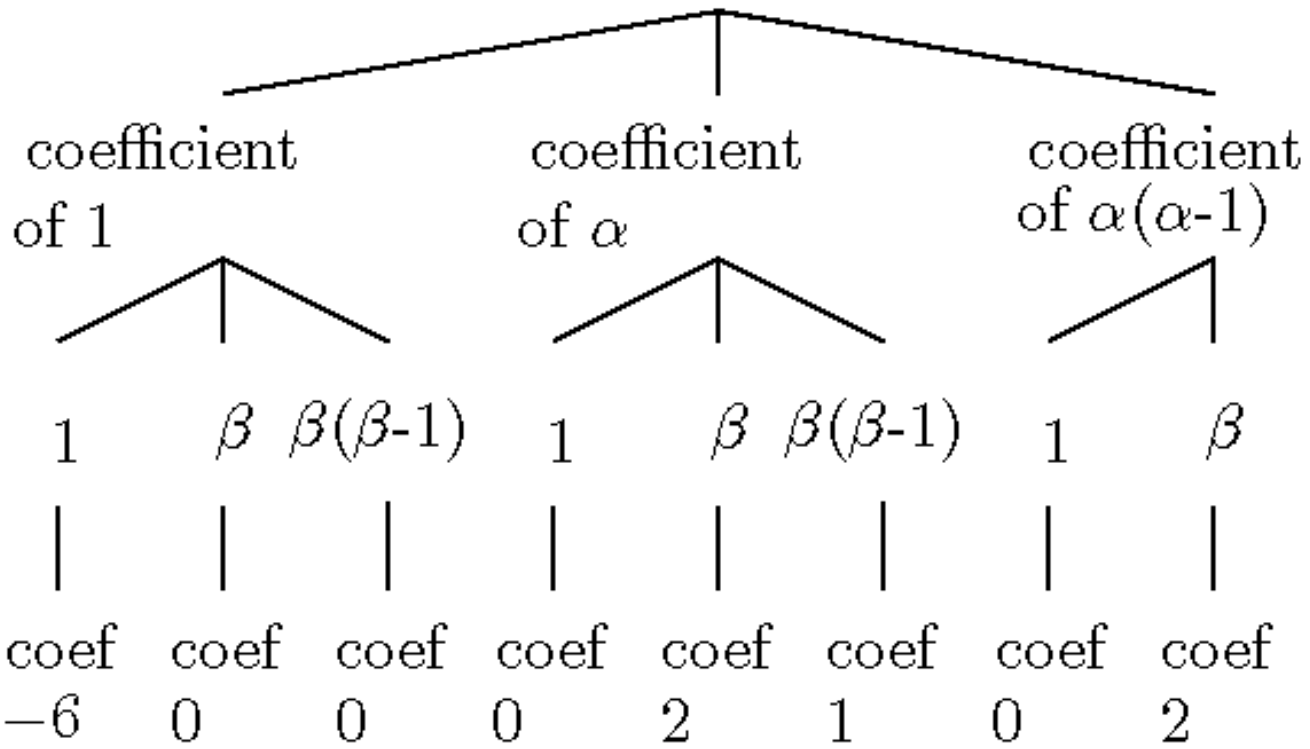
from the polynomial($\alpha = 1$) and store the result to the place of the polynomial($\alpha = 1$).



We subtract the polynomial($\alpha = 1$)
from the polynomial($\alpha = 2$) and store the result to the place of the polynomial($\alpha = 2$).



We don't care the coefficient of $\beta(\beta - 1)$



$$B = (-6 + 0\beta + 0\beta(\beta - 1)) + \alpha(0 + 2\beta + 1\beta(\beta - 1))$$
$$+\alpha(\alpha - 1)(0 + 2\beta + 0\beta(\beta - 1)) \qquad (1)$$

Finally, we expand eq. (1).

Computation time for computing the discriminant for $f(x)$ of degree 17

Divided stage:

Small problem 1 generated by interpolation:871m13.504s

Small problem 2 generated by interpolation:852m10.139s

Small problem 3 generated by interpolation:851m32.733s

Small problem 4 generated by interpolation:877m35.759s

Small problem 5 generated by interpolation:869m11.776s

Small problem 6 generated by interpolation:856m06.433s

Small problem 7 generated by interpolation:842m22.586s

Small problem 8 generated by interpolation:873m54.573s

Small problem 9 generated by interpolation:848m30.900s

Small problem 10 generated by interpolation:777m07.483s

Small problem 11 generated by interpolation:863m36.559s

Small problem 12 generated by interpolation:850m15.090s

Small problem 13 generated by interpolation:849m59.693s

Small problem 14 generated by interpolation:869m23.785s

Small problem 15 generated by interpolation:851m27.902s

Small problem 16 generated by interpolation:818m04.673s

Small problem 17 generated by interpolation:880m35.452s

Small problem 18 generated by interpolation:791m25.893s

These problems can be solved in parallel computing.

We use five nodes to solve these problems.

Conquer stage:

We interpolate the results.

It takes 612m15.268s.

Comparison with another algorithm

Our algorithm can be also used for the purpose of computing determinants with polynomials and resultants in general case. Conversely, there is the **specialized algorithm** for computing general discriminant formulas. The name is the Cayley method.

k	Cayley method Singular-3-1-6	Interpolation Serial	Interpolation Parallel
12	17.197s	2m41.004s	8.912s
13	1m55.036s	18m42.406s	56.499s
14	15m26.978s	177m38.849s	7m46.184s
15	95m09.931s	1265m33.327s	52m55.039s
16	613m47.301s 765424MB	???	372m10.574s 622448MB

The Cayley method is implemented in Singular.

It is not easy to parallelize the Cayley method.

The Cayley method **cannot** be adapted to compute determinants with polynomials and resultants in general case.

General-purpose algorithms for computing determinants with general elements

1.minor expansion

2.Laplace expansion

3.fraction-free Gauss elimination

4.Berkowitz method, Fadeev method

5.**GaussEDF**(That is used in computer algebra system Fermat)

General-purpose algorithms for computing resultants in general case

Bézout matrix representation + (1. or ... 5.) or Subresultant sequence

Comparison with minor expansion

TRIP 1.2.26 is a famous computer algebra system which has fast polynomial multiplication.

k	TRIP 1.2.26 minor exp. Serial	TRIP 1.2.26 minor exp. Parallel	Interpolation Serial	Interpolation Parallel
12	2m59.687s	2m54.455s	2m41.004s	8.912s
13	25m35.621s	20m26.959s	18m42.406s	56.499s
14	237m14.666s	141m35.767s	177m38.849s	7m46.184s
15	out of mem.	out of mem.	1265m33.327s	52m55.039s
16	out of mem.	out of mem.	too long	372m10.574s

CPU:E5-4650L, 4 CPU, 32 Core, mem:1440Gbyte

Practical problem

```
E6(a)=a^27
+12*p2*a^25
+60*p2^2*a^23
-48*p1*a^22
+(168*p2^3+96*q2)*a^21
-336*p2*p1*a^20+(294*p2^4+528*q2*p2+480*p0)*a^19
+(-1008*p2^2*p1-1344*q1)*a^18
+(144*p1^2+336*p2^5+1152*q2*p2^2+2304*p0*p2)*a^17
+((-1680*p2^3-768*q2)*p1-5568*q1*p2)*a^16
+(608*p2*p1^2+252*p2^6+1200*q2*p2^3+4768*p0*p2^2+17280*q0-1248*q2^2)*a^15
+((-1680*p2^4-2688*q2*p2+2304*p0)*p1-8832*q1*p2^2)*a^14
+(976*p2^2*p1^2+3264*q1*p1+120*p2^7+480*q2*p2^4+5696*p0*p2^3+
(43776*q0-4800*q2^2)*p2+12288*q2*p0)*a^13
+(832*p1^3+(-1008*p2^5-3072*q2*p2^2+5888*p0*p2)*p1-6528*q1*p2^3
+10752*q2*q1)*a^12
+((704*p2^3+4224*q2)*p1^2+2688*q1*p2*p1+33*p2^8-144*q2*p2^5+4384*p0*p2^4
+(41472*q0-6720*q2^2)*p2^2+34560*q2*p0*p2-34560*p0^2)*a^11
```

```
+(2560*p2*p1^3+(-336*p2^6-768*q2*p2^3+3584*p0*p2^2+64512*q0+8448*q2^2)*p1
-2112*q1*p2^4+23040*q2*q1*p2-70656*p0*q1)*a^10
+((176*p2^4+8960*q2*p2-18944*p0)*p1^2-5504*q1*p2^2*p1+4*p2^9-192*q2*p2^6
+2176*p0*p2^5+(22528*q0-3840*q2^2)*p2^3+32768*q2*p0*p2^2-39936*p0^2*p2
+110592*q2*q0-40704*q1^2+5120*q2^3)*a^9
+(2688*p2^2*p1^3+4608*q1*p1^2+(-48*p2^7+768*q2*p2^4-1536*p0*p2^3
+(82944*q0+16128*q2^2)*p2-73728*q2*p0)*p1-192*q1*p2^5+13824*q2*q1*p2^2
-64512*p0*q1*p2)*a^8
+(-2560*p1^4+(-32*p2^5+5376*q2*p2^2-16384*p0*p2)*p1^2+(-6144*q1*p2^3
-15360*q2*q1)*p1-48*q2*p2^7+608*p0*p2^6+(9600*q0-480*q2^2)*p2^4
+10752*q2*p0*p2^3-20992*p0^2*p2^2+(156672*q2*q0-38400*q1^2+9984*q2^3)*p2
-165888*p0*q0-56832*q2^2*p0)*a^7
+((1024*p2^3-10240*q2)*p1^3+10240*q1*p2*p1^2+(384*q2*p2^5-1792*p0*p2^4
+(21504*q0+6912*q2^2)*p2^2-57344*q2*p0*p2+49152*p0^2)*p1+1536*q2*q1*p2^3
-19456*p0*q1*p2^2-110592*q1*q0-21504*q2^2*q1)*a^6
+(-1536*p2*p1^4+(-16*p2^6+768*q2*p2^3-4608*p0*p2^2+27648*q0-19200*q2^2)*p1^2
+(-1344*q1*p2^4+10752*q2*q1*p2-9216*p0*q1)*p1+64*p0*p2^7
+(2304*q0+192*q2^2)*p2^5-3072*p0^2*p2^3+(55296*q2*q0-12288*q1^2
+4608*q2^3)*p2^2+(-110592*p0*q0-46080*q2^2*p0)*p2+73728*q2*p0^2)*a^5
```

```
+((64*p2^4-4096*q2*p2+8192*p0)*p1^3-512*q1*p2^2*p1^2+(-256*p0*p2^5+
(3072*q0-768*q2^2)*p2^3-8192*q2*p0*p2^2+16384*p0^2*p2+73728*q2*q0
-39936*q1^2-18432*q2^3)*p1-1024*p0*q1*p2^3+(-36864*q1*q0-3072*q2^2*q1)*p2
+24576*q2*p0*q1)*a^4
+(256*p2^2*p1^4+15360*q1*p1^3+(128*q2*p2^4-1024*p0*p2^3+(-6144*q0
-2560*q2^2)*p2+8192*q2*p0)*p1^2+(-128*q1*p2^5+2048*q2*q1*p2^2
-14336*p0*q1*p2)*p1+256*q0*p2^6-256*q2*p0*p2^5+256*p0^2*p2^4+(9216*q2*q0
-2560*q1^2-256*q2^3)*p2^3+(-18432*p0*q0-7680*q2^2*p0)*p2^2
+24576*q2*p0^2*p2-110592*q0^2+55296*q2^2*q0-30720*q2*q1^2-16384*p0^3
-6912*q2^4)*a^3
+(-1024*p1^5+4096*p0*p2*p1^3+24576*q2*q1*p1^2-12288*q1^2*p2*p1)*a^2
+(-2048*q2*p1^4+2048*q1*p2*p1^3+((-3072*q0-256*q2^2)*p2^2+4096*q2*p0*p2
-4096*p0^2)*p1^2+(512*q2*q1*p2^3-1024*p0*q1*p2^2-36864*q1*q0
+9216*q2^2*q1)*p1-256*q1^2*p2^4-6144*q2*q1^2*p2+12288*p0*q1^2)*a
+(4096*q0-1024*q2^2)*p1^3+(2048*q2*q1*p2-4096*p0*q1)*p1^2
-1024*q1^2*p2^2*p1-4096*q1^3
```

We compute the discriminant of $E6(a)$.

Of course, we cannot use the Cayley method.

What is $E6(a)$?

The polynomial equation of degree 27 with 6 parameters which Galois group is Weyl group $W(E6)$.

cf. T.Shioda, Construction of elliptic curves with high rank via the invariants of the Weyl groups, J.Math.Soc.Japan 43(1991), 673-719

Evaluation of performance 3

$E6_k(a) = E6(a) \bmod a^{k+1}$

k	TRIP 1.2.26 minor exp. Serial	TRIP 1.2.26 minor exp. Parallel	Interpolation Serial	Interpolation Parallel
7	1m59.969s	21.477s	6m28.043s	19.864s
8	7m38.860s	48.304s	16m45.179s	45.613s
9	30m28.221s	2m27.779s	41m32.633s	1m53.171s
10	94m33.848s	7m04.401s	82m55.905s	3m40.043s
11	297m54.106s	21m15.628s	165m27.601s	8m38.818s
12	785m36.664s	60m45.546s	284m07.984s	13m55.036s

CPU:E5-4650L, 4 CPU, 32 Core, mem:1440Gbyte

Intel C++ Compiler:13.1.2

Computing the discriminant of $E6(a)$

We use our interpolation.

Computing environment 1
CPU:Intel Core i7 980X 1CPU(6Core)
Mem:24G
Serial computation: 10913m45.857s
Parallel computation: 1773m28.272s
Speed Up: 6.15 **"super-linear"**

The number of terms in the discriminant of $E6(a)$:
27329463(file size:2.5G)

Remark 1

From an input,

$$A(x,y,z) = \begin{vmatrix} x+y+z^{131}+z^{43}+z^{7} & xy \\ 2yz^{50}+xy & xyz^{100} \end{vmatrix}$$

we regard x and y as dense variables and z as a sparse variable by using the results of $A(x, y_0, z_0), A(x_0, y, z_0)$ and $A(x_0, y_0, z)$. In this case, we apply our interpolation algorithm only for the dense variables x and y. At the each evaluation point, we compute a polynomial with z as evaluation values by using general-purpose algorithms.

If all variables are sparse variables, our algorithms are useless.

Remark 2

We implemented some algorithms for computing determinants and resultants in TRIP.

1.Minor expansion

2.fraction-free Gauss elimination

3.Berkowitz method

4.Generating Bézout determinants

These programs except 2 are also fast.

In another web page(not LAPROGCA), you can download those programs.

Future work

We will extend our program LAPROGCA.

1.Linear solver:sparse matrix by using a nested dissection ordering
2.Linear solver:polynomial coefficients by using multivariate Padé approximation
3.Characteristic polynomial and Minimal polynomial: polynomial entries
4.Characteristic polynomial and Minimal polynomial: for generalized eigenvalue problems

Thank you for your attention!