In-place Arithmetic for Univariate Polynomials over an Algebraic Number Field

Michael Monagan
Center for Experimental and Constructive Mathematics,
Simon Fraser University, Vancouver, British Columbia.

Joint work with Mahdi Javadi.

GCDs over algebraic number fields.

Let $L = \mathbb{Q}(\alpha_1, \alpha_2, ..., \alpha_k)$.
Let f_1, f_2 be non-zero in $L[x_1, x_2, ..., x_n]$.
Compute $g = gcd(f_1, f_2)$.

Modular GCD Algorithms.

[1979] $k = 0, n \geq 1$: Zippel (CRT and sparse interpolation)

[1987] $k = 1, n = 1$: Langemyr and McCallum (CRT)

[1989] $k = 1, n = 1$: Geddes, Gonnet and Smedley

$$\mathbb{Z}[z]/m(z) \xrightarrow{z=a\in\mathbb{Z}} \mathbb{Z}_{m(a)}$$

[1995] $k = 1, n = 1$: Encarnacion (CRT + ratrecon)

[2002] $k \geq 1, n = 1$: van Hoeij and MM (CRT + ratrecon)

[2004] $k \geq 1, n \geq 1$: van Hoeij and MM (dense)

[2007] $k \geq 1, n \geq 1$: Javadi and MM (sparse)

[2007] $k \geq 1, n = 1$: Moreno Maza and Schost (FFT based)

Outline

- Example of GCD in $L[x]$ in Magma for $L = \mathbb{Q}(\sqrt[3]{2}, \sqrt{1 + \sqrt[3]{2}})$
- Using a recursive dense representation.
- In-place algorithms for $L_p[x]$.
- A benchmark with Magma for arithmetic $L_p[x]$.
- What's in the paper and current work.

Magma example.

```
Magma V2.15-15    Tue Nov 17 2009 12:25:02 on maple    [Seed = 475287942]
Type ? for help.  Type <Ctrl>-D to quit.

> Q := RationalField();
> P1<z1> := PolynomialRing(Q);
> m1 := z1^3-2; L1<z1> := quo<P1|m1>;     // L1 = Q[z1]/(z1^3-2)
> P2<z2> := PolynomialRing(L1);
> m2 := z2^2-1-z1; L2<z2> := quo<P2|m2>;  // L2 = L1[z2]/(z2^2-1-z1)
> IsField(L2);
true
> L<x> := PolynomialRing(L2);
> f1 := (x-z1-z2+2/3)*(x^2+z1*z2*x-1);
> f2 := (x-z1-z2+2/3)*((z2+z1^2+z1+6)*x-1);
> f1;
x^3 + ((z1-1)*z2 - z1 + 2/3)*x^2 + ((-z1^2+2/3*z1)*z2 -
    z1^2 - z1 - 1)*x + z2 + z1 - 2/3
> f2;
(z2 + (z1^2+z1+6))*x^2 + ((-z1^2-2*z1-16/3)*z2 -
    1/3*z1^2 - 19/3*z1)*x + z2 + z1 - 2/3
> Gcd(f1,f2);
x - z2 - z1 + 2/3
```

Magma example continued ...

The monic Euclidean algorithm.

```
> u := LeadingCoefficient(f2); u;
z2 + z1^2 + z1 + 6
> f2 := (1/u)*f2; // make f2 monic
> r1 := f1 mod f2;
> r1;

((-121/12675*z1^2 + 2362/12675*z1 - 964/12675)*z2 - 542/12675*z1^2 -
    226/12675*z1 - 11828/12675)*x + (-5702/38025*z1^2 + 638/2925*z1 +
    34282/38025)*z2 - 7129/38025*z1^2 + 30838/38025*z1 - 16786/38025

> u := LeadingCoefficient(r1); r1 := (1/u)*r1; // make r1 monic
> f2 mod r1;
0
> r1; // = gcd(f1,f2)
x - z2 - z1 + 2/3
```

Magma example continued ...

Using the modular algorithm of Monagan and van Hoeij (2002).

```
> p := 13;
> Zp := FiniteField(p);
> P1<z1> := PolynomialRing(Zp);
> m1 := z1^3-2; L1<z1> := quo<P1|m1>;
> P2<z2> := PolynomialRing(L1);
> m2 := z2^2-1-z1; L2<z2> := quo<P2|m2>;
> L<x> := PolynomialRing(L2);
> f1 := (x-z1-z2+2/3)*(x^2+z1*z2*x-1);
> f2 := (x-z1-z2+2/3)*((z2+z1^2+z1+6)*x-1);
> u := LeadingCoefficient(f2); u;
z2 + z1^2 + z1 + 6
> 1/u;

Runtime error in '/': Argument is not invertible

> IsField(L2);
false
```

Magma example continued ...

```
> p := 17;

> Zp := FiniteField(p);
...
> f2 := (x-z1-z2+2/3)*((z2+z1^2+z1+6)*x-1);
> u := LeadingCoefficient(f2);

> 1/u;
(13*z1^2 + 15*z1 + 14)*z2 + 12*z1^2 + 6*z1 + 13

> f2 := (1/u)*f2; // make f2 monic
> r1 := f1 mod f2;
> u := LeadingCoefficient(r1); r1 := (1/u)*r1; // make r1 monic
> f2 mod r1;
0
> g17 := r1; g17; // = gcd(f1,f2) mod 17
x + 16*z2 + 16*z1 + 12
```

Magma example continued ...

Repeat for additional primes.

```
> p := 19;
...
> g19 := r1; g19; // = gcd(f1,f2) mod 19
x + 18*z2 + 18*z1 + 7
```

Apply Chinese remaindering and rational reconstuction.

```
Maple 13 (X86 64 LINUX)
> g17 := x + 16*z2 + 16*z1 + 12:
> g19 := x + 18*z2 + 18*z1 + 7:
> g := iratrecon( chrem([g17,g19],[17,19]), 17*19 );

                    g := 2/3 + x - z2 - z1
```

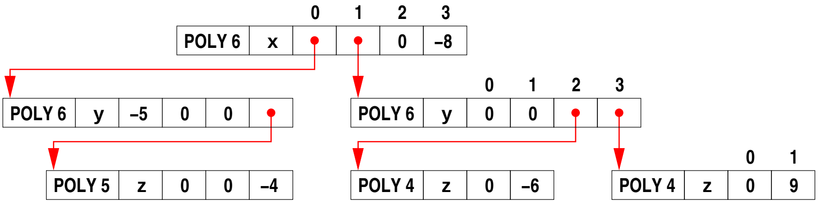
If g divides f_1 and f_2 stop and output g.

If we represent elements of $L = \mathbb{Q}(\alpha_1, \alpha_2, ..., \alpha_k)$ as polynomials in $\mathbb{Q}[z_1][z_2]\cdots[z_k] \mod \langle m_1(z_1), m_2(z_2), \ldots, m_k(z_k)\rangle$,

How do we do arithmetic in $L[x]$ mod p ?
How do we represent elements of $L[x]$ mod p ?

Recursive Dense Representations for Polynomials.

Pari's recursive dense representation for $\mathbb{Z}[z][y][x]$.

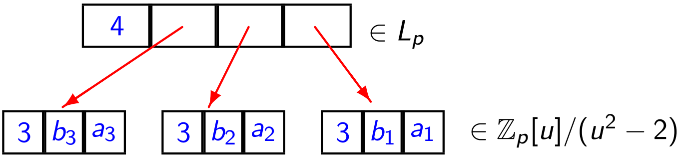


$$(-5y - 4z^2y^3) + (-6zy^2 + 9zy^3)x - 8x^3$$

Let $L = \mathbb{Q}(\sqrt{2}, \sqrt{5+\sqrt{2}})$.

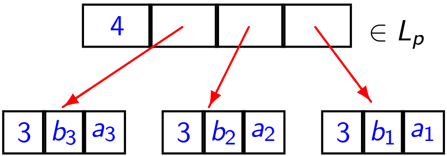
Build $L_p[x] = \mathbb{Z}_p[u]/(u^2-2)[v]/(v^2-5-u)[x]$.

Use arrays of arrays of arrays of machine integers for $\mathbb{Z}_p[u][v][x]$.



An element $(a_1 u + b_1)v^2 + (a_2 u + b_2)v + (a_3 u + b_3) \in L_v$.

Let $L = \mathbb{Q}(\sqrt{2}, \sqrt{5+\sqrt{2}})$.
Build $L_p[x] = \mathbb{Z}_p[u]/(u^2-2)[v]/(v^2-5-u)[x]$.
Use arrays of arrays of arrays of machine integers for $\mathbb{Z}_p[u][v][x]$.



$\in L_p$

$\in \mathbb{Z}_p[u]/(u^2-2)$

An element $(a_1 u + b_1)v^2 + (a_2 u + b_2)v + (a_3 u + b_3) \in L_v$.

Consider $(au+b) \times (cu+d)$ in $L_u = \mathbb{Z}_p[u]/(u^2-2)$.

We multiply $(au+b) \times (cu+d) =$ | 4 | bd | $ad+bc$ | ac |

then reduce mod $u^2 - 2$ to get | 3 | $ad+bc$ | $bd+2ac$ |.

Real work: 6 multiplications in $\mathbb{Z}_p$ and 3 additions in $\mathbb{Z}_p$.

Main idea: reuse an array W of working storage to reduce the number of storage allocations from

$$O(\deg_x(f_1) \times \deg_x(f_2) \times \deg_{z_2}(m_2)^2 \times \ldots \times \deg_{z_k}(m_k)^2) \text{ to } O(1).$$

Our recursive dense representation.

Let $\bar{m}_1 = z_1^3 + 3 = m_1 \bmod p$ and
$\bar{m}_2 = z_2^2 + (5z_1 + 4)z_2 + (7z_1^2 + 3z_1 + 6) = m_1 \bmod p$ and
$f = (3 + 4z_1) + (5 + 6z_1)z_2 + ((7 + 8z_1 + 9z_1^2) + 10z_2)x.$

Representation for $E = [\bar{m}_1 = m_1 \bmod p, \bar{m}_2 = m_2 \bmod p]$.

$E =$

2	2	6	3	7	1	4	5	0	0	1	0	0	3	3	0	0	1
$\bar{m}_2$													$\bar{m}_1$				

Representation for f in $L_p[x]$.

$A =$

1	1	1	3	4	0	1	5	6	0	1	2	7	8	9	0	10	0	0
	$(3 + 4z_1) + (5 + 6z_1)z_2$									$(7 + 8z_1 + 9z_1^2) + 10z_2$								

Our in-place C library

Let $L = \mathbb{Q}(\alpha_1, ..., \alpha_K)$ and $D = \deg(L)$.
Let S_K be the space to represent one element of L mod p.

We have $S_K = 1 + \deg(m_K) \cdot S_{K-1} \implies D < S_K < 2D.$

Our in-place C library

Let $L = \mathbb{Q}(\alpha_1, ..., \alpha_K)$ and $D = \deg(L)$.
Let S_K be the space to represent one element of L mod p.
We have $S_K = 1 + \deg(m_K) \cdot S_{K-1} \implies D < S_K < 2D$.

IP_MUL(K, E, p, a, b, c, W); $|W| < 6S_K$

Multiplies $a \times b$ in $L[x]$ mod p. Answer is written into c.

IP_REM(K, E, p, a, b, W); $|W| < 6S_K$

Divides a by b in $L[x]$ mod p. Quotient and remainder are written into a.

Our in-place C library

Let $L = \mathbb{Q}(\alpha_1, ..., \alpha_K)$ and $D = \deg(L)$.
Let S_K be the space to represent one element of L mod p.
We have $S_K = 1 + \deg(m_K) \cdot S_{K-1} \implies D < S_K < 2D.$

IP_MUL(K, E, p, a, b, c, W); $|W| < 6S_K$

Multiplies $a \times b$ in $L[x]$ mod p. Answer is written into c.

IP_REM(K, E, p, a, b, W); $|W| < 6S_K$

Divides a by b in $L[x]$ mod p. Quotient and remainder are written into a.

IP_INV(K, E, p, a, W); $|W| < 15S_K$

Inverts a in L mod p. Answer is written into a.

IP_GCD(K, E, p, a, b, W); $|W| < 17S_K$

Computes $\gcd(a, b)$ in $L[x]$ mod p.

Answer is written into either a or b (both are destroyed).

Let $L_k = \mathbb{Z}[z_k][z_{k-1}]\ldots[z_1]/\langle m_1, \ldots, m_k, p\rangle)$.
Let $a = \sum_{i=0}^{d_a} a_i x^i$ and $b = \sum_{i=0}^{d_b} b_i x^i$ where $a_i, b_i \in L_k[x]$.
Let $c = a \times b = \sum_{i=0}^{d_a+d_b} c_i x^i$. Compute

$$c_j = \left[\sum_{i=\max(0,j-d_b)}^{\min(j,d_a)} a_i \times b_{j-i} \right] \mod m_k(z_k)$$

where $a_i \times b_{j-i}$ is done in $L_{k-1}[z_k]$.

- Needs working storage for $2 \in O(1)$ elements of $L_{k-1}[z_k]$ of degree $2(\deg_{z_k}(m_k) - 1)$.
- Division in $L_k[x]$ can be similarly implemented.

Benchmark

Here $p = 3037000453$, $L = \mathbb{Q}(\alpha_1, \alpha_2)$, m_1 and m_2 have degrees d_1 and d_2 such that $d = d_1 \times d_2 = 60$. We choose three polynomials a, b, g of degree d_x in x with coefficients chosen from L_p at random.

d_1	d_2	d_x	IP_MUL	MAG_MUL	IP_REM	MAG_REM	IP_GCD	MAG_GCD
2	30	40	0.124	0.050	0.123	0.09	0.384	2.26
3	20	40	0.108	0.054	0.106	0.11	0.340	2.35
4	15	40	0.106	0.056	0.106	0.10	0.327	2.39
6	10	40	0.106	0.121	0.105	0.14	0.328	5.44
10	6	40	0.100	0.093	0.100	0.37	0.303	7.84
15	4	40	0.097	0.055	0.095	0.17	0.283	3.27
20	3	40	0.092	0.046	0.091	0.14	0.267	2.54
30	2	40	0.087	0.038	0.087	0.10	0.242	1.85
2	30	80	0.477	0.115	0.478	0.27	1.449	9.41
3	20	80	0.407	0.127	0.409	0.27	1.304	9.68
4	15	80	0.404	0.132	0.406	0.28	1.253	9.98
6	10	80	0.398	0.253	0.400	0.35	1.234	22.01
10	6	80	0.380	0.197	0.381	0.86	1.151	31.57
15	4	80	0.365	0.127	0.364	0.40	1.081	13.49
20	3	80	0.353	0.109	0.353	0.33	1.030	10.59
30	2	80	0.336	0.086	0.337	0.26	0.932	7.83

Table: Timings in CPU seconds on an AMD Opteron 254 CPU running at 2.8 GHz

Benchmark

Here $p = 3037000453$, $L = \mathbb{Q}(\alpha_1, \alpha_2)$, m_1 and m_2 have degrees d_1 and d_2 such that $d = d_1 \times d_2 = 60$. We choose three polynomials a, b, g of degree d_x in x with coefficients chosen from L_p at random.

d_1	d_2	d_x	IP_MUL	MAG_MUL	IP_REM	MAG_REM	IP_GCD	MAG_GCD
2	30	40	0.124	0.050	0.123	0.09	0.384	2.26
3	20	40	0.108	0.054	0.106	0.11	0.340	2.35
4	15	40	0.106	0.056	0.106	0.10	0.327	2.39
6	10	40	0.106	0.121	0.105	0.14	0.328	5.44
10	6	40	0.100	0.093	0.100	0.37	0.303	7.84
15	4	40	0.097	0.055	0.095	0.17	0.283	3.27
20	3	40	0.092	0.046	0.091	0.14	0.267	2.54
30	2	40	0.087	0.038	0.087	0.10	0.242	1.85
2	30	80	0.477	0.115	0.478	0.27	1.449	9.41
3	20	80	0.407	0.127	0.409	0.27	1.304	9.68
4	15	80	0.404	0.132	0.406	0.28	1.253	9.98
6	10	80	0.398	0.253	0.400	0.35	1.234	22.01
10	6	80	0.380	0.197	0.381	0.86	1.151	31.57
15	4	80	0.365	0.127	0.364	0.40	1.081	13.49
20	3	80	0.353	0.109	0.353	0.33	1.030	10.59
30	2	80	0.336	0.086	0.337	0.26	0.932	7.83

Table: Timings in CPU seconds on an AMD Opteron 254 CPU running at 2.8 GHz

Integer division optimization.

Multiplication in $\mathbb{Z}_p[z]$.

```
M = p*p;
d_c = d_a+d_b;
for( k=0; k<=d_c; k++ ) {
   t = 0;
   for( i=max(0,k-d_b); i <= min(k,d_a); i++ )
   {
      // t = (t+A[i]*B[k-i]) % p;
      if( t<0 ); else t = t-M;
      t = t+A[i]*B[k-i];
   }
   t = t % p;
   if( t<0 ) t = t+p;
   C[k] = t;
}
```

This improved performance of IP_GCD by a factor of 5 to 6.

What's in the paper?

- Pseudo-code for IP_MUL, IP_REM, IP_INV, IP_GCD.
- Formulas for bounds for $|W|$.
- Website link for repository of C code and test problems.

Current and future work.

- We have integrated IP_MUL, IP_GCD into Maple 14.
- Will integrate IP_REM into Maple 15.
- Are experimenting with an FFT based multiplication.