Speeding up polynomial GCD, a crucial operation in Maple

Michael Monagan, Simon Fraser University
Email: mmonagan@sfu.ca

Maple Conference
November 2-5, 2021 • Free Virtual Event

NSERC
CRSNG

SFU

Problem: given $A, B \in \mathbb{Z}[x_0, x_1, \ldots, x_n]$ compute $G = \gcd(A, B)$.
In Maple

```
> simplify(A/B);
```

computes G and outputs $(A \div G)/(B \div G)$.

I'm interested in **sparse** A, B and G.
Let $d_i = \deg(G, x_i)$. We say G is sparse if $\#G \ll \prod_{i=0}^{n}(d_i + 1)$.
Example

$$G = x_0^4 + x_1^4 x_2 + x_2^4 x_3 + x_3^4 x_4 + x_4^4 x_5 + x_5^4 x_0 + 1$$

Here $\#G = 7$ and $(4+1)^6 = 15625$.
Goal: an algorithm whose cost is polynomial in $n, d_i, \#A, \#B, \#G$.

Approach: We compute $\gcd(A, B)$ modulo primes $p_1, p_2, \ldots$ then apply the CRT.
How do we compute $G = \gcd(A, B) \mod p$ where $p = p_1$?

Since 2005 Maple uses Zippel's GCD Algorithm

The implementation is detailed in de Kleine, Monagan, Wittkopf [3]

Input $A, B \in \mathbb{Z}_p[x_0, x_1, \ldots, x_n]$.

1 For $j = 1,2,3,\ldots$ do

Pick $\bar{\alpha}_j \in \mathbb{Z}_p^n$ at random.

Compute $g_j = \gcd(A(x_0, \alpha_{j1}, \ldots, \alpha_{jn}), B(x_0, \alpha_{j1}, \ldots, \alpha_{jn}))$.

3 Interpolate $G = \gcd(A, B) \mod p$ from images g_j and points $\bar{\alpha}_j$.

Let $d_i = \deg(G, x_i)$, $D = d_1 + d_2 + \cdots + d_n$ and $t = \#G$.
Zippel 1979 [5]: needs $O(Dt)$ images g_j and does $O(D)$ linear solves.

Monagan/Hu 2016 [2]: needs $O(t)$ images and one linear solve.
Uses a Kronecker substitution and Ben-Or/Tiwari sparse interpolation.

NB: In step 2 we skip $j = 0$ as $\gcd(A(x_0, 1, \ldots, 1), B(x_0, 1, \ldots, 1))$ which could be an unlucky evaluation.

Timing comparison with Maple's gcd

I've implemented the Monagan/Hu algorithm in Maple + C code for main subroutines.

$$A = G\bar{A} \quad B = G\bar{B} \quad n = 7 \quad d_i = 30 \quad \text{coeffs=rand}(2^{100})$$

$\#G$	$\#\bar{A}, \#\bar{B}$	$\#A, \#B$	Maple	MGCD1	t	MGCD2	t
10^1	10^4	10^5	21.59 s	0.661 s	5	0.395 s	3
10^2	10^3	10^5	47.74 s	0.731 s	18	0.707 s	19
10^3	10^2	10^5	295.4 s	3.852 s	201	1.262 s	22
10^4	10^1	10^5	11084. s	45.00 s	2112	1.450 s	2
10^1	10^5	10^6	331.1 s	7.32 s	5	5.35 s	3
10^2	10^4	10^6	2413. s	10.95 s	19	6.90 s	24
10^3	10^3	10^6	31952. s	30.49 s	198	25.56 s	197
10^4	10^2	10^6	NA	238.2 s	2063	13.15 s	23
10^5	10^1	10^6	NA	3511. s	21037	10.47 s	3

MGCD2: interpolate smallest of $G, \bar{A}, \bar{B}$

Ben-Or/Tiwari sparse polynomial interpolation [1]

Let $C(x_1,\ldots,x_n) = \sum_{i=1}^{t} a_i M_i(x_1,\ldots,x_n)$ where $a_i \in \mathbb{Z}$.
Context: $C = coeff(G, x_0^k)$ that we are interpolating and let $T \geq t$ be given.

1 Compute $b_j = C(2^j, 3^j, 5^j, \ldots, p_n^j)$ for $1 \leq j \leq 2T$.

2 Let $m_i = M_i(2, 3, 5, \ldots, p_n)$ and $\Lambda(z) = \prod_{i=1}^{t}(z - m_i)$.
Compute t and $\Lambda(z)$ from b_j using the Berlekamp-Massey algorithm [4].

3 Factor $\Lambda(z)$ to get the integer roots m_i.

4 Factor the integers m_i using trial division by $2, 3, \ldots, p_n$ to get M_i.
E.g. if $m_i = 45000 = 2^3 3^2 5^4$ then $M_i = {x_1}^3 {x_2}^2 {x_3}^4$.

5 Solve the shifted $t \times t$ Vandermonde linear system below for the coefficients a_i.

$$V\,a = b \;\text{ where }\; V_{ij} = m_i^j \;\text{ for } 1 \leq i \leq t, 1 \leq j \leq t.$$

Problem: The $b_j = C(2^j, 3^j, \ldots, p_n^j)$ are very large integers!
Solution? Do steps 1,2,3,5 modulo a prime $p > m_i \implies p > 2^{d_1} 3^{d_2} 5^{d_5} \ldots p_n^{d_n}$.
E.g. for $n = 8, d_i = 10$ we require $p > 7.4 \times 10^{69}$.

Modified Ben-Or/Tiwari sparse polynomial interpolation [2]

Use an invertible Kronecker substitution $\mathbf{Kr} : \mathbb{Z}_p[x_1, \ldots, x_n] \to \mathbb{Z}_p[y]$.

For $r_i > d_i = \deg(C, x_i)$ let $\mathbf{Kr}(C) = C(y, y^{r_1}, \ldots, y^{r_1 r_2 \ldots r_{n-1}}) = \sum_{i=1}^{t} a_i y^{e_i}$.

1. Pick a random generator α in $\mathbb{Z}_p$.
   Compute $b_j = \mathbf{Kr}(C)(\alpha^j) \mod p$ for $1 \leq j \leq 2T$.
2. Let $m_i = \alpha^{e_i}$ and $\Lambda(z) = \prod_{i=1}^{t}(z - m_i)$.
   Compute t and $\Lambda(z)$ from b_j using the Berlekamp-Massey algorithm.
3. Factor $\Lambda(z)$ to get the roots $m_i \in \mathbb{Z}_p$ using Cantor-Zassenhaus.
4. Solve $\alpha^{e_i} = m_i$ in $\mathbb{Z}_p$ for e_i – a discrete logarithm. Require $p > e_i$ and p to be "smooth".
5. Solve the $t \times t$ shifted Vandermonde system below for the coefficients a_i.

$$V\,a = b \;\text{ where }\; V_{ij} = m_i^j \;\text{ for } 1 \leq i \leq t, 1 \leq j \leq t.$$

Prime size: for $n = 8, d_i = 10$ this only requires $p > 11^8 = 214,358,881$.
Our code uses $p = 4,601,552,919,265,804,289 = 61 \times 67 \times 2^{50} + 1 = 2^{61.99}$.

Implementation Notes

We've coded the following routines in C to speed up the implementation.
The complexities on the right are arithmetic operations in $\mathbb{Z}_p$.

1 Evaluation $Kr(A)(x_0, \alpha^j)$ and $Kr(B)(x_0, \alpha^j)$ .... $O(D + st)$
2 Euclidean algorithm in $\mathbb{Z}_p[x]$ .... $O(d_0^2 t)$
3 Berlekamp-Massey algorithm to get $\Lambda(z)$ .... $O(t^2)$
4 Root finding in $\mathbb{Z}_p[x]$ (Cantor-Zassenhaus) .... $O(t^2 \log p)$
5 Discrete logarithms (Pohlig Hellman + Shanks) .... $O(t \log p)$
6 Zippel's $O(t^2)$ Vandermonde solver from [6] .... $O(t^2)$

Key: $D = \sum_{i=1}^{n} d_i$ where $d_i = \deg(G, x_i)$, $s = \#A + \#B$, $t = \min(\#G, \#\bar{A}, \#\bar{B})$.

The C code supports primes $p < 2^{63}$.
The **bottleneck** is usually step 1 since usually $s \gg t$.

Scaling the images and interpolating $G, \bar{A}, \bar{B}$.

The image $g_j = \gcd(Kr(A)(x, \alpha^j), Kr(B)(x, \alpha^j)) \in \mathbb{Z}_p[x]$ is unique up to a scalar.
We need g_j to be an image of a polynomial to use polynomial interpolation.

Let $\mathrm{LC}(G)$ denote the leading coefficient of G in x_0.
To interpolate G from g_j we need $\mathrm{LC}(g_j) = Kr(\mathrm{LC}(G))(\alpha^j)$.
Problem: We don't know $\mathrm{LC}(G)$.
Solution: We do know $\mathrm{LC}(A)$ and $\mathrm{LC}(A) = \mathrm{LC}(G) \times \mathrm{LC}(\bar{A})$, a multiple of $\mathrm{LC}(G)$.

1. Pick a random generator α in $\mathbb{Z}_p$.
2. For $j = 1, 2, 3, \ldots$ do
   $a_j \leftarrow Kr(A)(x_0, \alpha^j) \mod p$; $b_j \leftarrow Kr(B)(x_0, \alpha^j) \mod p$.
   $g_j \leftarrow monic(\gcd(a_j, b_j) \mod p)$.
   $\bar{a}_j \leftarrow a_j/g_j$; $\bar{b}_j \leftarrow b_j/g_j$; $g_j \leftarrow LC(a_j) \times g_j$.
3. Interpolate H, I, or J using g_j, $\bar{a}_j$ and $\bar{b}_j$ respectively, to obtain the smaller of $H = \mathrm{LC}(\bar{A})G$, $I = \mathrm{LC}(G)\bar{A}$, and $J = \mathrm{LC}(G)\bar{B}$, then remove the content.

References

- Michael Ben-Or and Prasoon Tiwari.
  A deterministic algorithm for sparse multivariate polynomial interpolation.
  In *Proc. of STOC '20*, pp. 301–309, ACM, 1988.
- Jiaxiong Hu and Michael Monagan. A Fast Parallel Sparse Polynomial GCD Algorithm.
  J. Symb. Cmpt. **105**:(1) 28–63, Springer, July 2021.
- J. de Kleine, M. Monagan, A. Wittkopf. Algorithms for the Non-monic case of the Sparse Modular GCD Algorithm. *Proc. ISSAC '2005*, pp. 124–131, ACM, 2005.
- J. L. Massey. Shift-register synthesis and BCH decoding.
  IEEE Trans. on Information Theory, 15:122–127, 1969.
- Richard Zippel. Probabilistic algorithms for sparse polynomials.
  In *Proc. of EUROSAM '79*, pp. 216–226, Springer, 1979.
- Richard Zippel. Interpolating Polynomials from their Values.
  J. Symb Cmpt. **9**:375–403, Springer, 1990.