On the Genericity of the Modular Polynomial GCD Algorithm

Erich Kaltofen

www.math.ncsu.edu/~kaltofen

Joint work with: Michael Monagan (Simon Fraser University)

W. S. Brown's 1971 modular GCD algorithm

For $\mathbb{Z}[x_1, \ldots, x_n]$ (Algorithm M)
Compute GCDs modulo several primes p_i (see P below);
Chinese-remainder the coefficients of the GCDs;
Test if enough primes.

For $\mathbb{Z}_p[x_1, \ldots, x_n]$ (Algorithm P)
Compute GCDs for several evaluations $x_n \leftarrow b_i$ (recursively);
Interpolate the coefficients of the GCDs;
Test if enough evaluations.

Details of Brown's algorithm

Select appropriate scalar multiples of modular images

$\longrightarrow$Impose GCD of leading coeffs as leading coeff of GCD, or

$\longrightarrow$Perform rational recovery on coefficients

Eliminate (finitely many) "unlucky" primes/evaluations

$\longrightarrow$Use symmetric remainders in range $-\lfloor p/2 \rfloor, \ldots, \lfloor p/2 \rfloor$

$\longrightarrow$Test first if GCD has not changed (folklore)

and then if it divides input polynomials

Subsequent Work

Moses and Yun ’73, Wang ’80, Kaltofen ’85
  Hensel lifting “EZ”-GCD algorithms and their sparse versions

Caviness and Rothstein ’75
  Modular GCD algorithm over Gaussian integers

Zippel ’79   Modular sparse GCD algorithm

Char, Gonnet, and Geddes ’84
  Map all the way to integer GCDs (heuristic)

Langemyr-McCallum ’89, Encarnación 95, Monagan-Margot ’98
  Algebraic number coefficients

Kaltofen ’85, Kaltofen and Trager ’88, Díaz and Kaltofen ’95
  GCDs of straight-line programs and black boxes

Dobbertin's example

```
z := (A^2*B^2*Z^2+A^2*B^2+B^2*Z^2+Z^2+B^2)/Z:
a := (B^2*C^2*A^2+B^2*C^2+C^2*A^2+A^2+C^2)/A:
b := (C^2*Z*B^2+C^2*Z+Z*B^2+B^2+Z)/B:
c := (Z*A*C^2+Z*A+A*C^2+C^2+A)/C:
P := Z^2*A^3*B^2*C^2*(z^2*b^2*c^2
                              +(a+z+1)^2*(a+c^2+1)):
P := expand(P) mod 2;
```

$P := A^2B^8Z^4C^4 + A^2B^2Z^4C^4 + A^4B^8Z^4C^4 + \ldots$ (158 terms total)

Maple V.4 fails to squarefree decompose P modulo 2.

Our idea

For $\mathbb{Z}_p[x_1][x_2, \ldots, x_n]$, where p is small (Algorithm M$'$)
Compute GCDs modulo several irreducibles $m_i(x_1)$;
Chinese-remainder the coefficients of the GCDs;
Test if enough primes.

Notes

1. Dan Grayson points out: modulo $m_i(x_1)$ is equivalent to $x_1 \leftarrow z \in \mathbb{Z}_p[z]/(m_i(z)) \supset \mathbb{Z}_p$
$\longrightarrow$FoxBox's extended domain black boxes

2. Unassociated irreducibles $m_i(x_1)$ are sufficiently dense
$\longrightarrow$see paper

3. Maple timings establish speed-up better than theory
$\longrightarrow$internal representation of algebraic extension

Conclusions

– Finding “right” generalization
can be difficult, but generic algorithms look simple

– $\mathbb{Z}_p[y]$ trick applies to Zippel/Ben-Or-Tiwari sparse interpolation

– Generic implementation may reveal subtle bugs/problems

* Generalization to a single generic algorithm for sparse+dense needs to be done