Subquadratic GCD

Niels Möller

October 30, 2008

Outline

Background

History

- 300 BC (or even earlier): Euclid's algorithm.
- 1938: Lehmer's algorithm.
- 1961: Binary GCD described by Stein.
- 1994, 1995: Sorensson, Weber.
- 1970, 1971: Knuth and Schönhage, subquadratic computation of continued fractions.
- ca 1987: Schönhage's "controlled Euclidean descent", unpublished.
- 2004: Stéhle and Zimmermann, recursive binary GCD.
- 2005–2008: Möller. Left-to-right algorithm. Simpler and slightly faster than earlier algorithms.

Comparison of GCD algorithms (before current project)

Algorithm	Time (ms)	# lines	
mpn_gcd	1440	304	GMP-4.1.4 (Weber)
mpn_rgcd	87	1967	"Classical" Schönhage GCD
mpn_bgcd	93	1348	Rec. bin. (Stehlé/Zimmermann)
mpn_sgcd	100	760	1987 alg. (Schönhage/Weilert)
mpn_ngcd	85	733	New algorithm for GMP-5

Comparison of GCD algorithms (before current project)

Algorithm	Time (ms)	# lines	
mpn_gcd	1440	304	GMP-4.1.4 (Weber)
mpn_rgcd	87	1967	"Classical" Schönhage GCD
mpn_bgcd	93	1348	Rec. bin. (Stehlé/Zimmermann)
mpn_sgcd	100	760	1987 alg. (Schönhage/Weilert)
mpn_ngcd	85	733	New algorithm for GMP-5

- Benchmarked on 32-bit AMD, with inputs of 48 000 digits.
- Cross-over around 7 700 digits.
- Today: 82 ms for the same machine and input size.

Questions

Q Where does the complexity come from?

A Accurate computation of the quotient sequence.

Q How to avoid that?

A Stop bothering about quotients.

What is HGCD?

Definition (Reduction)

$$\begin{pmatrix} A \\ B \end{pmatrix} = M \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

- Positive integers A, B, α, and β.
- Matrix M, non-negative integer elements.
- $\det M = 1$.

What is HGCD?

Definition (Reduction)

$$\begin{pmatrix} A \\ B \end{pmatrix} = M \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

- Positive integers A, B, α, and β.
- Matrix M, non-negative integer elements.
- $\det M = 1$.

Fact

For any reduction, $\mathrm{GCD}(A, B) = \mathrm{GCD}(\alpha, \beta)$

What is HGCD?

Definition (Reduction)

$$\begin{pmatrix} A \\ B \end{pmatrix} = M \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

- Positive integers A, B, α, and β.
- Matrix M, non-negative integer elements.
- $\det M = 1$.

Fact

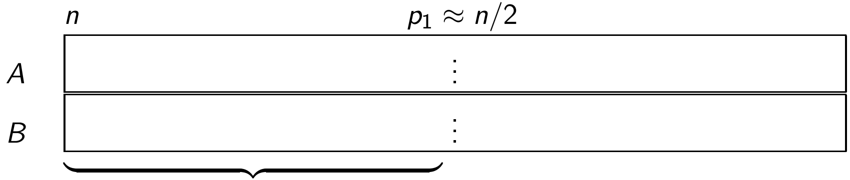
For any reduction, $\mathrm{GCD}(A, B) = \mathrm{GCD}(\alpha, \beta)$

Definition (HGCD, “half gcd”)

Input: A, B, of size n

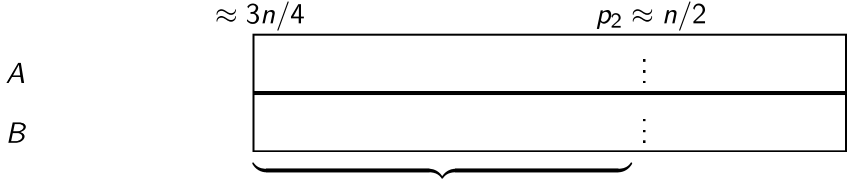
Output: M, with size of α, β and M elements $\approx n/2$

Main idea of subquadratic HGCD



$$M_1 \leftarrow \mathrm{HGCD}(\lfloor 2^{-p_1} A \rfloor, \lfloor 2^{-p_1} B \rfloor)$$

$$\begin{pmatrix} A \\ B \end{pmatrix} \leftarrow M_1^{-1} \begin{pmatrix} A \\ B \end{pmatrix}$$



$$M_2 \leftarrow \mathrm{HGCD}(\lfloor 2^{-p_2} A \rfloor, \lfloor 2^{-p_2} B \rfloor)$$

$$M \leftarrow M_1 \cdot M_2$$

Asymptotic running time

$\textsc{gcd}(A, B)$

while $\#(A, B) > \textsc{gcd-threshold}$
    do
        $n \leftarrow \#(A, B),\ p \leftarrow \lfloor 2n/3 \rfloor$
        $M \leftarrow \textsc{hgcd}(\lfloor 2^{-p} A \rfloor, \lfloor 2^{-p} B \rfloor)$
        $(A; B) \leftarrow M^{-1}(A; B)$
return $\textsc{gcd-base}(A, B)$

Running times for operations on n-bit numbers

$$\begin{aligned} \text{Multiplication:} \quad & M(n) = O(n \log n \log\log n) \\ \textsc{hgcd}: \quad & H(n) = O(M(n) \log n) \\ \textsc{gcd}: \quad & G(n) \approx 2H(n) \end{aligned}$$

Quotient based HGCD

Definition (Quotient sequence)

For any positive integers a, b, the quotient sequence q_j and remainder sequence r_j are defined by

$$r_0 = a \qquad\qquad r_1 = b$$

$$q_j = \lfloor r_{j-1}/r_j \rfloor \qquad\qquad r_{j+1} = r_{j-1} - q_j r_j$$

Definition (Quotient sequence)

For any positive integers a, b, the quotient sequence q_j and remainder sequence r_j are defined by

$$r_0 = a \qquad\qquad r_1 = b$$
$$q_j = \lfloor r_{j-1}/r_j \rfloor \qquad\qquad r_{j+1} = r_{j-1} - q_j r_j$$

Fact

$$\begin{pmatrix} a \\ b \end{pmatrix} = M \begin{pmatrix} r_j \\ r_{j+1} \end{pmatrix}$$

with

$$M = \begin{pmatrix} q_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} q_2 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} q_j & 1 \\ 1 & 0 \end{pmatrix}$$

Theorem (Jebelean's criterion)

Let $a > b > 0$, with remainders r_j and r_{j+1}, and

$$\begin{pmatrix} a \\ b \end{pmatrix} = \underbrace{\begin{pmatrix} u & u' \\ v & v' \end{pmatrix}}_{=M} \begin{pmatrix} r_j \\ r_{j+1} \end{pmatrix}$$

Let $p > 0$ be arbitrary, $0 \leq A', B' < 2^p$, and define

$$\begin{pmatrix} A \\ B \end{pmatrix} = 2^p \begin{pmatrix} a \\ b \end{pmatrix} + \begin{pmatrix} A' \\ B' \end{pmatrix}$$

$$\begin{pmatrix} R_j \\ R_{j+1} \end{pmatrix} = 2^p \begin{pmatrix} r_j \\ r_{j+1} \end{pmatrix} + M^{-1} \begin{pmatrix} A' \\ B' \end{pmatrix}$$

For even j, the following two statements are equivalent:

(i) $r_{j+1} \geq v$ *and* $r_j - r_{j+1} \geq u + u'$

(ii) *For any p and any A', B', the jth remainders of A and B are R_j and R_{j+1}. The quotient sequences are the same.*

Quotient based HGCD

A generalization of Lehmer's algorithm

Define HGCD(a, b) to return an M satisfying Jebelean's criterion.

Example (Recursive computation)

$$\begin{aligned}
(a; b) &= (858\,824; 528\,747) \\
M_1 &= (13, 8; 8, 5) && \text{No difficulties} \\
(c; d) &= M_1^{-1}(a; b) = 16\,(4009; 194) + (0; 15) \\
M_2 &= \textsc{hgcd}(4009, 194) = (21, 20; 1, 1) \\
M_2^{-1}(4009; 194) &= (129; 65) && \text{Satisfies Jebelean} \\
M &= M_1 \cdot M_2 = (281, 268; 173, 165) \\
M^{-1}(a; b) &= (1764; 1355)
\end{aligned}$$

Backup step

Example (Continued)

$$\begin{aligned} (a; b) &= (858\,824; 528\,747) \\ M &= M_1 \cdot M_2 = (281, 268; 173, 165) \\ M^{-1}(a; b) &= (1764; 1355) \qquad \text{Violates Jebelean} \\ 1764 - 1355 &\not\geq 281 + 268 \end{aligned}$$

M corresponds to quotients $1, 1, 1, 1, 1, 1, 20, 1$.
E.g., $(A; B) = 8\,(a; b) + (1; 7)$ has quotient sequence starting with $1, 1, 1, 1, 1, 1, 20, 2$.

Backup step

Example (Continued)

$$\begin{aligned}(a;b) &= (858\,824; 528\,747)\\ M &= M_1 \cdot M_2 = (281, 268; 173, 165)\\ M^{-1}(a;b) &= (1764; 1355) \qquad \text{Violates Jebelean}\\ 1764 - 1355 &\not\geq 281 + 268\end{aligned}$$

M corresponds to quotients $1, 1, 1, 1, 1, 1, 20, 1$.
E.g., $(A; B) = 8\,(a; b) + (1; 7)$ has quotient sequence starting with $1, 1, 1, 1, 1, 1, 20, 2$.

Conclusion

- The quotients are correct for $(a; b)$, but not robust enough.
- Must drop final quotient before returning $\textsc{hgcd}(a, b)$.

Robust HGCD

A robustness condition

Definition (Robust reduction)

A reduction M of $(A; B)$ is robust iff

$$M^{-1}\left\{\begin{pmatrix} A \\ B \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}\right\} > 0$$

for all "small" $(x; y)$. More precisely, for all $(x; y) \in S$, where

$$S = \{(x; y) \in \mathbb{R}^2, |x| < 2, |y| < 2, |x - y| < 2\}$$

A robustness condition

Definition (Robust reduction)

A reduction M of $(A; B)$ is robust iff

$$M^{-1}\left\{\begin{pmatrix} A \\ B \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}\right\} > 0$$

for all "small" $(x; y)$. More precisely, for all $(x; y) \in S$, where

$$S = \{(x; y) \in \mathbb{R}^2, |x| < 2, |y| < 2, |x - y| < 2\}$$

Theorem

The reduction

$$\begin{pmatrix} A \\ B \end{pmatrix} = \underbrace{\begin{pmatrix} u & u' \\ v & v' \end{pmatrix}}_{=M} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

is robust iff $\alpha \geq 2\max(u', v')$ *and* $\beta \geq 2\max(u, v)$

Strong robustness

Definition (Strong robustess)

Let $n = \#(A, B)$ denote the bitsize of the larger of A and B. If $\# \min(\alpha, \beta) > \lfloor n/2 \rfloor + 1$, then M is strongly robust.

Lemma

If a reduction M is strongly robust, then it is robust.

Strong robustness

Definition (Strong robustess)

Let $n = \#(A, B)$ denote the bitsize of the larger of A and B. If $\# \min(\alpha, \beta) > \lfloor n/2 \rfloor + 1$, then M is strongly robust.

Lemma

If a reduction M is strongly robust, then it is robust.

Theorem (Schönhage-Weilert reduction)

For arbitrary $A, B > 0$, let $n = \#(A, B)$ and $s = \lfloor n/2 \rfloor + 1$. Assume $\# min(A, B) > s$. There exists a unique strongly robust M such that $\# \min(\alpha, \beta) > s$ and $\#|\alpha - \beta| \leq s$.

New simpler HGCD

HGCD(A, B)

$n \leftarrow \#(A, B)$
$s \leftarrow \lfloor n/2 \rfloor + 1$
Split: $p_1 \leftarrow \lfloor n/2 \rfloor$, $A = 2^{p_1} a + A'$, $B = 2^{p_1} b + B'$
$(\alpha, \beta, M_1) \leftarrow$ HGCD(a, b)
$(A; B) \leftarrow 2^{p_1}(\alpha; \beta) + M_1^{-1}(A'; B')$ $\quad \triangleright\ \#|A - B| \approx 3n/4$
One subtraction and one division step on $(A; B)$. Update M_1.
Split: $p_2 \leftarrow 2s - \#(A, B) + 1$, $A = 2^{p_2} a + A'$, $B = 2^{p_2} b + B'$
$(\alpha, \beta, M_2) \leftarrow$ HGCD(a, b)
$(A; B) \leftarrow 2^{p_2}(\alpha; \beta) + M_2^{-1}(A'; B')$
$M \leftarrow M_1 \cdot M_2$
while $\#|A - B| > s$ $\quad \triangleright$ At most four times
    One division step on $(A; B)$. Update M.
return (A, B, M)

FFT-related optimizations

Matrix multiplication

$$M_1 \cdot M_2 \quad 2 \times 2 \text{ matrices}$$

Assume FFT and sizes such that the transforms dominates the computation time.

	FFT	IFFT	Saving
Naive	16	8	0%
Schönhage-Strassen	14	7	12%
Invariance	8	4	50%

Recently implemented. 15% speedup of GCD for for large inputs.

Matrix-vector multiplication

- If α, β are returned: M of size $n/4$, A', B' of size $n/2$.

$$M^{-1} \cdot \begin{pmatrix} A \\ B \end{pmatrix} = 2^p \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + M^{-1} \cdot \begin{pmatrix} A' \\ B' \end{pmatrix}$$

	#Mults.	Prod. size	
Naive	4	$3n/4$	Wins in FFT range
Block	8	$n/2$	Can use invariance
S.-S.	7	$n/2$	Wins in Karatsuba range

Matrix-vector multiplication

- If α, β are returned: M of size $n/4$, A', B' of size $n/2$.

$$M^{-1} \cdot \begin{pmatrix} A \\ B \end{pmatrix} = 2^p \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + M^{-1} \cdot \begin{pmatrix} A' \\ B' \end{pmatrix}$$

	#Mults.	Prod. size	
Naive	4	$3n/4$	Wins in FFT range
Block	8	$n/2$	Can use invariance
S.-S.	7	$n/2$	Wins in Karatsuba range

- If only matrix is returned: M of size $n/4$, A, B of size n.

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = M^{-1} \cdot \begin{pmatrix} A \\ B \end{pmatrix}$$

α, β are of size $3n/4$ (cancellation!). Compute mod$(2^k \pm 1)$, with transform size $\approx 3n/4$.

Matrix-vector multiplication

- If α, β are returned: M of size $n/4$, A', B' of size $n/2$.

$$M^{-1} \cdot \begin{pmatrix} A \\ B \end{pmatrix} = 2^p \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + M^{-1} \cdot \begin{pmatrix} A' \\ B' \end{pmatrix}$$

	#Mults.	Prod. size	
Naive	4	$3n/4$	Wins in FFT range
Block	8	$n/2$	Can use invariance
S.-S.	7	$n/2$	Wins in Karatsuba range

- If only matrix is returned: M of size $n/4$, A, B of size n.

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = M^{-1} \cdot \begin{pmatrix} A \\ B \end{pmatrix}$$

  α, β are of size $3n/4$ (cancellation!). Compute $\bmod (2^k \pm 1)$, with transform size $\approx 3n/4$.

- Same transform size, $3n/4$, no matter if reduced numbers are available or not!

FFT multiplication

b	Bit-size for polynomialization
$\mathbb{Z}_m$	Ring for polynomial coefficients
$n = 2^k$	Transform size
ℓ	Length of product polynimial (degree + 1)

For "small-prime" FFT, m is the product if a small number of limb-sized primes.

$c \leftarrow u \cdot v$

1. Split inputs, $u = p_u(2^b) = u_0 + \cdots + u_{\ell_u - 1} 2^{b(\ell_u - 1)}$, $v = p_v(2^b)$
2. Evaluate $p_u(\omega_j) \bmod m$ and $p_v(\omega_j) \bmod m$ for ℓ distinct ω_j
3. Compute $p_c(\omega_j) = p_u(\omega_j) p_v(\omega_j) \bmod m$.
4. Find c_j, so that $p_c(x) = c_0 + c_1 x + \cdots + c_{\ell-1} x^{\ell-1}$
5. Evaluate $c = p_c(2^b)$

Correctness

Fact

If the coefficients of $p_u(x)p_v(x)$, over $\mathbb{Z}$, belong to $[0, m)$, then

$$c = uv \bmod (2^{nb} - 1)$$

Can be extended to other bilinear operations

- $ab + cd$.
- Strassen-multiplication of matrices.

For correctness, the coefficients of the resulting polynomial, over $\mathbb{Z}$, must be uniquely determined modulo m.

FFT interface

Parameters Takes bit size L, a bound for the smaller factor S, and a growth parameter G, and limit parameter M. Outputs a polynomial base b, transform size $n = 2^k$, product length $\ell = \lceil L/b \rceil$, small factor length $\ell_s = \lceil S/b \rceil$, and modulo m, such that

$$nb > L \qquad\qquad 2^{2b}\ell_s G \leq m$$

Transform Takes an integer u and computes the first ℓ elements of the transform.

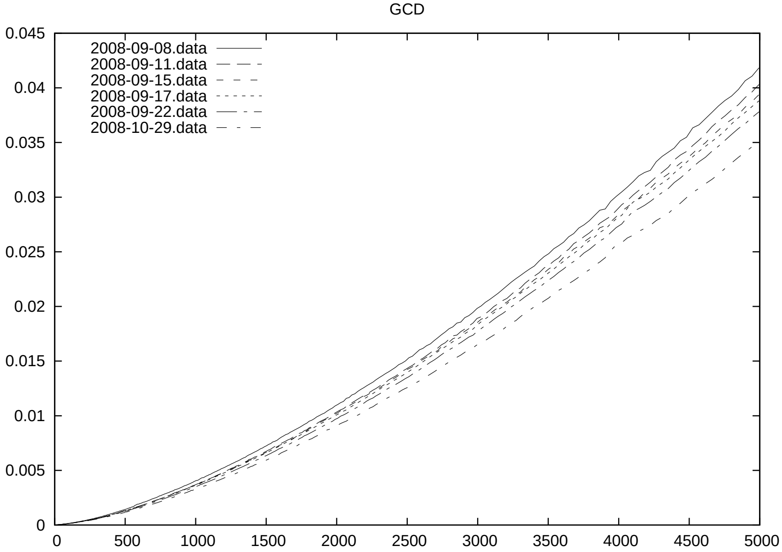
Inverse Takes a the first ℓ elements of a transform, computes ℓ polynomial coefficients u_j under the assumption that the last $n - \ell$ coefficients are zero, and returns the corresponding number. If $M < G$, coefficients may be negative.

Multiplication Multiplies two transforms. One of them should correspond to a polynomial of length at most ℓ_s.

Add, sub Add or subtract two transforms.

Scalar mul Multiply a transform by a small constant.

Results



GCD
2008-09-08.data
2008-09-11.data
2008-09-15.data
2008-09-17.data
2008-09-22.data
2008-10-29.data
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Corresponding changes

2008-09-08	Old quotient-based HGCD.
2008-09-11	New HGCD code.
2008-09-15	Use Strassen multiplication.
2008-09-17	Changed p i GCD outerloop from $n/2$ to $3n/2$.
2008-09-22	New assembler loop for $uA - vB$.
2008-10-29	FFT invariance

Performance for large numbers

- Use more FFT invariance, currently used only for $M_1 \cdot M_2$.
- Try a HGCD function returning only the matrix M, not the reduced numbers. Can use FFT wrap-around.
- Investigate the choice of p in the GCD and GCDEXT outer-loops. $p = 2n/3$ seems to work fine for GCD, but optimal splitting is much harder for GCDEXT.
- Further optimizations of the FFT transformations. Currently, assembler loops only for x64_64, and only the forward transform has been optimized seriously.

Performance for medium size numbers

Linear work O(n) calls to HGCD 2. Current code is full of branches and not optimized for current processors.

Quadratic work In base case.

- Combine mpn_mul_1 and mpn_submul_1 in a single loop computing $va - ub$. Tried on x86_64, with a modest speedup.
- On processors where mpn_mul_2 and mpn_submul_2 are efficient, implement HGCD4, as two calls to HGCD2. Then apply an M with two-limb elements to the bignums.