Subquadratic GCD

Niels Möller

October 30, 2008
Outline

Background
  Algorithm comparison
  The half-gcd (HGCD) operation
  Subquadratic HGCD

Quotient based HGCD
  Jebelean’s criterion
  Why backup steps?

Robust HGCD
  Difference-based HGCD

FFT-related optimizations

FFT interface

Optimizations
Background
300 BC (or even earlier): Euclid’s algorithm.
1938: Lehmer’s algorithm.
1961: Binary $\text{gcd}$ described by Stein.
2004: Stéhle and Zimmermann, recursive binary $\text{gcd}$.
Comparison of GCD algorithms (before current project)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (ms)</th>
<th># lines</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpn_gcd</td>
<td>1440</td>
<td>304</td>
<td>GMP-4.1.4 (Weber)</td>
</tr>
<tr>
<td>mpn_rgcd</td>
<td>87</td>
<td>1967</td>
<td>“Classical” Schönhage GCD</td>
</tr>
<tr>
<td>mpn_bgcd</td>
<td>93</td>
<td>1348</td>
<td>Rec. bin. (Stehlé/Zimmermann)</td>
</tr>
<tr>
<td>mpn_sgcd</td>
<td>100</td>
<td>760</td>
<td>1987 alg. (Schönhage/Weilert)</td>
</tr>
<tr>
<td>mpn_ngcd</td>
<td>85</td>
<td>733</td>
<td>New algorithm for GMP-5</td>
</tr>
</tbody>
</table>
Comparison of GCD algorithms (before current project)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (ms)</th>
<th># lines</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpn_gcd</td>
<td>1440</td>
<td>304</td>
<td>GMP-4.1.4 (Weber)</td>
</tr>
<tr>
<td>mpn_rgcd</td>
<td>87</td>
<td>1967</td>
<td>“Classical” Schönhage GCD</td>
</tr>
<tr>
<td>mpn_bgcd</td>
<td>93</td>
<td>1348</td>
<td>Rec. bin. (Stehlé/Zimmermann)</td>
</tr>
<tr>
<td>mpn_sgcd</td>
<td>100</td>
<td>760</td>
<td>1987 alg. (Schönhage/Weilert)</td>
</tr>
<tr>
<td>mpn_ngcd</td>
<td>85</td>
<td>733</td>
<td>New algorithm for GMP-5</td>
</tr>
</tbody>
</table>

- 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 \( A, B, \alpha, \) and \( \beta. \)
- 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 $A$, $B$, $\alpha$, and $\beta$.
- Matrix $M$, non-negative integer elements.
- $\det M = 1$.

**Fact**

*For any reduction, $\gcd(A, B) = \gcd(\alpha, \beta)$*
What is HGCD?

Definition (Reduction)

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

- Positive integers \( A, B, \alpha, \) and \( \beta. \)
- Matrix \( M, \) non-negative integer elements.
- \( \det M = 1. \)

Fact

For any reduction, \( \gcd(A, B) = \gcd(\alpha, \beta) \)

Definition (HGCD, “half gcd”)

Input: \( A, B, \) of size \( n \)
Output: \( M, \) with size of \( \alpha, \beta \) and \( M \) elements \( \approx n/2 \)
Main idea of subquadratic HGCD

\[
\begin{align*}
M_1 &\leftarrow \text{HGCD}(\lfloor 2^{-p_1} A \rfloor, \lfloor 2^{-p_1} B \rfloor) \\
\begin{pmatrix} A \\ B \end{pmatrix} &\leftarrow M_1^{-1} \begin{pmatrix} A \\ B \end{pmatrix} \\
m &\approx 3n/4 \\
M_2 &\leftarrow \text{HGCD}(\lfloor 2^{-p_2} A \rfloor, \lfloor 2^{-p_2} B \rfloor) \\
M &\leftarrow M_1 \cdot M_2
\end{align*}
\]
Asymptotic running time

\[ \text{GCD}(A, B) \]

1. while \( \#(A, B) > \text{GCD-THRESHOLD} \) do
2. \hspace{1em} n ← \#(A, B), \ p ← \lfloor 2n/3 \rfloor
3. \hspace{1em} M ← \text{HGCD}(\lfloor 2^{-p} A \rfloor, \lfloor 2^{-p} B \rfloor)
4. \hspace{1em} (A; B) ← M^{-1}(A; B)
5. return \text{GCD-BASE}(A, B)

Running times for operations on \( n \)-bit numbers

- **Multiplication:** \( M(n) = O(n \log n \log \log n) \)
- **HGCD:** \( H(n) = O(M(n) \log n) \)
- **GCD:** \( G(n) ≈ 2H(n) \)
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

\[
\begin{align*}
    r_0 &= a \\
    r_1 &= b \\
    q_j &= \left\lfloor \frac{r_{j-1}}{r_j} \right\rfloor \\
    r_{j+1} &= r_{j-1} - q_j r_j
\end{align*}
\]
**Definition (Quotient sequence)**

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

\[
\begin{align*}
    r_0 &= a \\
    q_j &= \lfloor r_{j-1}/r_j \rfloor \\
    r_1 &= b \\
    r_{j+1} &= r_{j-1} - q_j r_j
\end{align*}
\]

**Fact**

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

with

\[
M = \begin{pmatrix} q_1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} q_2 & 1 \\ 1 & 0 \end{pmatrix} \ldots \begin{pmatrix} q_j & 1 \\ 1 & 0 \end{pmatrix}
\]
Theorem (Jebelean’s criterion)

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

\[
\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} u & u' \\ v & v' \end{pmatrix} \begin{pmatrix} r_j \\ r_{j+1} \end{pmatrix} = M
\]

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 \( j \)th remainders of \( A \) and \( B \) are \( R_j \) and \( R_{j+1} \). The quotient sequences are the same.
Quotient based HCGD

A generalization of Lehmer’s algorithm

Define $\text{HCGD}(a, b)$ to return an $M$ satisfying Jebelean’s criterion.

Example (Recursive computation)

$$(a; b) = (858824; 528747)$$

$$M_1 = (13, 8; 8, 5) \quad \text{No difficulties}$$

$$(c; d) = M_1^{-1}(a; b) = 16 (4009; 194) + (0; 15)$$

$$M_2 = \text{HCGD}(4009, 194) = (21, 20; 1, 1)$$

$$M_2^{-1}(4009; 194) = (129; 65) \quad \text{Satisfies Jebelean}$$

$$M = M_1 \cdot M_2 = (281, 268; 173, 165)$$

$$M^{-1}(a; b) = (1764; 1355)$$
Example (Continued)

\[(a; b) = (858\,824; 528\,747)\]
\[M = M_1 \cdot M_2 = (281, 268; 173, 165)\]
\[M^{-1}(a; b) = (1764; 1355)\]

Violates Jebelean

\[1764 - 1355 \not\geq 281 + 268\]

\[M\] 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)

\[(a; b) = (858\,824; 528\,747)\]

\[M = M_1 \cdot M_2 = (281, 268; 173, 165)\]

\[M^{-1}(a; b) = (1764; 1355)\] Violates Jebelean

\[1764 - 1355 \not\geq 281 + 268\]

\(M\) 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 \(\text{HGCD}(a, b)\).
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

$$M^{-1}\left\{\left(\begin{array}{c}A \\ B\end{array}\right) + \left(\begin{array}{c}x \\ y\end{array}\right)\right\} > 0$$

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*

$$\left(\begin{array}{c}A \\ B\end{array}\right) = \left(\begin{array}{cc}u & u' \\ v & v'\end{array}\right) \left(\begin{array}{c}\alpha \\ \beta\end{array}\right) = M$$

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

Definition (Strong robustness)

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 robustness)**

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

\[ \text{HGCD}(A, B) \]

1. \( n \leftarrow \#(A, B) \)
2. \( s \leftarrow \lfloor n/2 \rfloor + 1 \)
3. Split: \( p_1 \leftarrow \lfloor n/2 \rfloor, A = 2^{p_1} a + A', B = 2^{p_1} b + B' \)
4. \( (\alpha, \beta, M_1) \leftarrow \text{HGCD}(a, b) \)
5. \( (A; B) \leftarrow 2^{p_1}(\alpha; \beta) + M_1^{-1}(A'; B') \quad \triangleright \# |A - B| \approx 3n/4 \)
6. One subtraction and one division step on \((A; B)\). Update \(M_1\).
7. Split: \( p_2 \leftarrow 2s - \#(A, B) + 1, A = 2^{p_2} a + A', B = 2^{p_2} b + B' \)
8. \( (\alpha, \beta, M_2) \leftarrow \text{HGCD}(a, b) \)
9. \( (A; B) \leftarrow 2^{p_2}(\alpha; \beta) + M_2^{-1}(A'; B') \)
10. \( M \leftarrow M_1 \cdot M_2 \)
11. While \( \# |A - B| > s \quad \triangleright \text{At most four times} \)
12. One division step on \((A; B)\). Update \(M\).
13. 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>FFT</th>
<th>IFFT</th>
<th>Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>16</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>Schönhage-Strassen</td>
<td>14</td>
<td>7</td>
<td>12%</td>
</tr>
<tr>
<td>Invariance</td>
<td>8</td>
<td>4</td>
<td>50%</td>
</tr>
</tbody>
</table>

Recently implemented. 15% speedup of GCD for for large inputs.
Matrix-vector multiplication

- If $\alpha, \beta$ are returned: $M$ of size $n/4$, $A', B'$ of size $n/2$.

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

<table>
<thead>
<tr>
<th></th>
<th>#Mults.</th>
<th>Prod. size</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>4</td>
<td>$3n/4$</td>
<td>Wins in FFT range</td>
</tr>
<tr>
<td>Block</td>
<td>8</td>
<td>$n/2$</td>
<td>Can use invariance</td>
</tr>
<tr>
<td>S.-S.</td>
<td>7</td>
<td>$n/2$</td>
<td>Wins in Karatsuba range</td>
</tr>
</tbody>
</table>
Matrix-vector multiplication

- If $\alpha, \beta$ are returned: $M$ of size $n/4$, $A', B'$ of size $n/2$.

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

<table>
<thead>
<tr>
<th>#Mults.</th>
<th>Prod. size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>4</td>
</tr>
<tr>
<td>Block</td>
<td>8</td>
</tr>
<tr>
<td>S.-S.</td>
<td>7</td>
</tr>
</tbody>
</table>

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

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

$\alpha, \beta$ are of size $3n/4$ (cancellation!). Compute $\text{mod}(2^k \pm 1)$, with transform size $\approx 3n/4$. 
Matrix-vector multiplication

- If $\alpha, \beta$ are returned: $M$ of size $n/4$, $A', B'$ of size $n/2$.

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

<table>
<thead>
<tr>
<th>#Mults.</th>
<th>Prod. size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>4</td>
</tr>
<tr>
<td>Block</td>
<td>8</td>
</tr>
<tr>
<td>S.-S.</td>
<td>7</td>
</tr>
</tbody>
</table>

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

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

$\alpha, \beta$ are of size $3n/4$ (cancellation!). Compute $\text{mod}(2^k \pm 1)$, with transform size $\approx 3n/4$.

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>Bit-size for polynomialization</td>
</tr>
<tr>
<td>$\mathbb{Z}_m$</td>
<td>Ring for polynomial coefficients</td>
</tr>
<tr>
<td>$n = 2^k$</td>
<td>Transform size</td>
</tr>
<tr>
<td>$\ell$</td>
<td>Length of product polynomial (degree + 1)</td>
</tr>
</tbody>
</table>

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 $p_u(\omega_j) \mod m$ and $p_v(\omega_j) \mod m$ for $\ell$ distinct $\omega_j$
3. Compute $p_c(\omega_j) = p_u(\omega_j)p_v(\omega_j) \mod m$.
4. Find $c_j$, so that $p_c(x) = c_0 + c_1x + \cdots + c_{\ell-1}x^{\ell-1}$
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

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

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

\[ nb > L \quad 2^{2b}\ell_s G \leq m \]

Transform Takes an integer $u$ and computes the first $\ell$ elements of the transform.

Inverse Takes the first $\ell$ elements of a transform, computes $\ell$ 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 $\ell_s$.

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

Graph showing data points for GCD from different dates.
Corresponding changes

2008-09-08  Old quotient-based \texttt{HGCD}.
2008-09-11  New \texttt{HGCD} code.
2008-09-15  Use Strassen multiplication.
2008-09-17  Changed $p$ i $\text{gcd}$ 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 $\text{HGCD}_2$. Current code is full of branches and not optimized for current processors.

**Quadratic work** In base case.

- Combine $\text{mpn\_mul\_1}$ and $\text{mpn\_submul\_1}$ in a single loop computing $va - ub$. Tried on x86_64, with a modest speedup.
- On processors where $\text{mpn\_mul\_2}$ and $\text{mpn\_submul\_2}$ are efficient, implement $\text{HGCD}_4$, as two calls to $\text{HGCD}_2$. Then apply an $M$ with two-limb elements to the bignums.