Solving equations to high precision

D. J. Bernstein
University of Illinois at Chicago

NSF DMS–9970409

Typical equation

$h = f^2$ in $\mathbf{R}[[t]]^*$.

e.g. $h \approx 1 + 4.2t + 5.6t^2 + \cdots$

$f \approx 1 + 2.1t + 0.6t^2 + \cdots$

Given $h \bmod t^{10000}$ and $f(0)$,
compute $f \bmod t^{10000}$.
How many operations in $\mathbf{R}$?

Analogous algorithms for

$\mathbf{F}_p[[t]]$ or $\mathbf{F}_p((t))$,

$\mathbf{Z}_p$ or $\mathbf{Q}_p$,

$\mathbf{R}$,

etc.

Beware carries in $\mathbf{Q}_p$,

roundoff error in $\mathbf{R}$.

Complexity measures

1. Nonlinear operations in **R**.
No cost for additions
or scalar multiplications.

2. Arithmetic operations in **R**.

3. Straight-line L1-cache
UltraSPARC cycles.

UltraSPARC stores real numbers in **memory locations** $m_1, m_2, \ldots$ and **registers** $r_1, r_2, \ldots, r_{32}$.

Input, constants, output in memory locations.

Registers initially 0.

In one cycle, can add, and multiply, and load or store.

Add: Write $r_j \pm r_k$ to r_i.
Multiply: Write $r_j r_k$ to r_i.
Load: Write m_j to r_i.
Store: Write r_i to m_j.

Cannot change a register
twice in one cycle.

Cannot use result of load
until cycle $+ 2$.

Cannot use result of arithmetic
until cycle $+ 3$,
except for store.

Fast Fourier transform

To multiply in $\mathbf{C}[t]/(t^{64}-1)$:

$\mathbf{C}[t]/(t^{64}-1)$
$\to \mathbf{C}[t]/(t^{32}-1) \times \mathbf{C}[t]/(t^{32}+1)$

Continue to $\mathbf{C} \times \mathbf{C} \times \cdots \times \mathbf{C}$.

(Gauss)

≤ 10 operations in $\mathbf{R}$ for
$at^k + bt^{n+k}$
$\mapsto (a + b\zeta)t^k, (a - b\zeta)t^k$
under $\mathbf{C}[t]/(t^{2n} - \zeta^2)$
$\to \mathbf{C}[t]/(t^n - \zeta) \times \mathbf{C}[t]/(t^n + \zeta)$.

Arithmetic $5n \lg n - 10n + 16$
for $\mathbf{C}[t]/(t^n - 1) \to \mathbf{C}^n$
when $n = 2^e$, $e \geq 3$.
Same for scaled inverse.

Twisted FFT

$\mathbf{C}[t]/(t^{64}-1)$
$\to \mathbf{C}[t]/(t^{32}-1) \times \mathbf{C}[t]/(t^{32}+1)$
$\to \mathbf{C}[t]/(t^{32}-1) \times \mathbf{C}[u]/(u^{32}-1)$

by $t \mapsto \zeta u$ where $\zeta^{32} = -1$.

Arithmetic $5n \lg n - 10n + 16$
for $\mathbf{C}[t]/(t^n - 1) \to \mathbf{C}^n$
when $n = 2^e$, $e \geq 3$.

Split-radix FFT

$$\mathbf{C}[t]/(t^{64}-1)$$
$$\to \mathbf{C}[t]/(t^{32}-1) \times \mathbf{C}[t]/(t^{32}+1)$$
$$\to \mathbf{C}[t]/(t^{32}-1) \times$$
$$\mathbf{C}[t]/(t^{16}-i) \times \mathbf{C}[t]/(t^{16}+i)$$
$$\to \mathbf{C}[t]/(t^{32}-1) \times$$
$$\mathbf{C}[u]/(u^{16}-1) \times \mathbf{C}[v]/(v^{16}-1)$$

by $t \mapsto \zeta u$, $t \mapsto \zeta^3 v$
where $\zeta^{16} = i$.

Arithmetic $4n \lg n - 6n + 8$
for $\mathbf{C}[t]/(t^n - 1) \to \mathbf{C}^n$
when $n = 2^e$, $e \geq 3$.
(Yavne, Duhamel, Hollmann, Martens, Stasinski, Vetterli, Nussbaumer)

Reduce loads by
using ζ^{-1} instead of ζ^3.
(new)

UltraSPARC cycles,
$\mathbf{C}[t]/(t^{256} - 1) \to \mathbf{C}^{256}$:

13600 (Swarztrauber FFTPACK)
9300 (Frigo-Johnson FFTW)

6261 (new djbfft)

Also new bounds for Pentium, Pentium II, Alpha, etc.

Real FFT

$\mathbf{R}[t]/(t^{64} - 1)$
$\to \mathbf{R}[t]/(t^{32} - 1) \times \mathbf{C}[t]/(t^{16} - i)$
(Gauss, Bergland)

Arithmetic $(4 + o(1))n \lg n$
for $\mathbf{R}[t]/(t^{2n} - 1) \leftrightarrow \mathbf{R}^2 \times \mathbf{C}^{n-1}$
when $n = 2^e$.
(Yavne et al.)

Arithmetic $(12 + o(1))n \lg n$
to multiply $p, q \in \mathbf{R}[t]/(t^{2n} - 1)$
when $n = 2^e$:
$pq = \text{FFT}^{-1}(\text{FFT}(p)\text{FFT}(q))$
where
$\text{FFT} : \mathbf{R}[t]/(t^{2n} - 1) \to \mathbf{R}^2 \times \mathbf{C}^{n-1}$.

$(8 + o(1))n \lg n$ for p^2.

$(8 + o(1))n \lg n$ for $p^8 - 5p^3$.

Given $p, q \in \mathbf{R}[t]$, $\deg pq < 2n$,
can compute pq
as $pq \bmod t^{2n} - 1$.

In particular,
can multiply in $\mathbf{R}[t]/t^n$
with arithmetic $(12 + o(1))n \lg n$,
when $n = 2^e$.

What if n is not a power of 2?

Still $(12 + o(1))n \lg n$.

For, e.g., $n = 10000$:

$pq \bmod (t^{16384} + 1)(t^{4096} - 1)$.

(Crandall, Fagin)

Or $pq \bmod t^{20480} - 1$.

(Gauss, Good)

Newton's method

More computations in $\mathbf{R}[t]/t^n$:

$(36 + o(1))n \lg n$ for reciprocal,
$(48 + o(1))n \lg n$ for quotient,
$(66 + o(1))n \lg n$ for square root,
$(88 + o(1))n \lg n$ for exp.

(Cook, Sieveking, Kung, Brent; assuming real split-radix FFT)

Improvements:

$(18 + o(1))n \lg n$ for reciprocal,
$(26 + o(1))n \lg n$ for quotient,
$(22 + o(1))n \lg n$ for square root,
$(34 + o(1))n \lg n$ for exp.

(new; partial results by Schönhage, A. Karp, Markstein, Brent, Harley, Zimmermann)

If $f, g \in \mathbf{R}[[t]]$, $gf = 1$,
$g = g_0 + g_1 t^n + \cdots$,
each g_j a polynomial
of degree $< n$:

$1 - g_0 f$ has form $u_1 t^n + \cdots$.
Then $g_1 = g_0 u_1 \bmod t^n$.

Given $f = f_0 + f_1 t^n + \cdots$,
and given $g \bmod t^n$,
can find $g \bmod t^{2n}$
using 7 FFTs:

$\text{FFT}(g_0)$,
$\text{FFT}(f_0)$, $g_0 f_0 = \text{FFT}^{-1}(\ldots)$,
$\text{FFT}(f_1)$, $g_0 f_1 = \text{FFT}^{-1}(\ldots)$,
$\text{FFT}(u_1)$, $g_0 u_1 = \text{FFT}^{-1}(\ldots)$.

Higher-order iterations allow even more FFT reuse.

$u_1 t^n + \cdots = 1 - g_0 f,$

$g_1 t^n + \cdots = g_0 u_1 t^n,$

$u_2 t^{2n} + u_3 t^{3n} + \cdots$
$= 1 - (g_0 + g_1 t^n) f,$

$g_2 t^{2n} + g_3 t^{3n} + \cdots$
$= (g_0 + g_1 t^n)(u_2 t^{2n} + u_3 t^{3n}),$

etc.

Multiplying $g_0 + g_1 t^n$ by $f_0 + f_1 t^n + f_2 t^{2n} + f_3 t^{3n}$, given $\mathrm{FFT}(g_0)$ et al.:

$f_0 g_0$: irrelevant
$f_1 g_0 + f_0 g_1$: one FFT^{-1}
$f_2 g_0 + f_1 g_1$: one FFT^{-1}
$f_3 g_0 + f_2 g_1$: one FFT^{-1}
$f_3 g_1$: irrelevant

Given f and g_0,
can compute $g_1, g_2, \ldots, g_{2^k-1}$
using $2^k \cdot 4.5 + k - 3$ FFTs.

Choose k as a slowly
increasing function of n
to minimize complexity.

Quotient: $h/f = hg$.

Have FFTs for half of g
as part of computing g.
So use 2.5 more FFTs.

Karp-Markstein: Don't compute g.
Multiply h by $1 + u_1 t^n$, then g_0.
Uses only 2 more FFTs.
U.S. Patent 5,341,321.

Square root:

If $f^2 = h$ and $gf = 1$ then

$f \approx f_0 + (1/2)g_0(f_0^2 - h)$,

$g \approx g_0 + g_0(1 - g_0 f)$.

Given h, f_0, g_0,

can compute $f_1, f_2, \ldots, f_{2^k-1}$

using $2^k \cdot 5.5 + k - 7$ FFTs.

Define $D(\sum a_j t^j) = \sum j a_j t^j$,
$I(\sum a_j t^j) = \sum_{j \geq 1} (a_j / j) t^j$.

$\log f = I(D(f)/f)$ if $f(0) = 1$.
About as fast as quotient.

(More work for
high-precision log on **R**.
Use AGM or binary splitting.)

Exponential:

If $f = \exp h$ and $gf = 1$ then

$f \approx f_0 + f_0(h + I(y))$,

$g \approx g_0 + g_0(1 - g_0 f)$

where

$y = (g_0 f_0 - 1)D(h) - g_0 D(f_0)$.

Given h, f_0, g_0,

can compute $f_1, f_2, \ldots, f_{2^k - 1}$

using $2^k \cdot 8.5 + k - 8$ FFTs.