

The tangent FFT

D. J. Bernstein

University of Illinois at Chicago

Advertisement

“SPEED: Software
Performance Enhancement
for Encryption and Decryption”

A workshop on software speeds
for secret-key cryptography
and public-key cryptography.

Amsterdam, June 11–12, 2007

`http://`

`www.hyperelliptic.org/SPEED`

The convolution problem

How quickly can we multiply polynomials in the ring $\mathbf{R}[x]$?

Answer depends on degrees, representation of polynomials, number of polynomials, etc.

Answer also depends on definition of “quickly.”

Many models of computation; many interesting cost measures.

Assume two inputs $f, g \in \mathbf{R}[x]$,
 $\deg f < m$, $\deg g \leq n - m$,
so $\deg fg < n$. Assume f, g, fg
represented as coeff sequences.

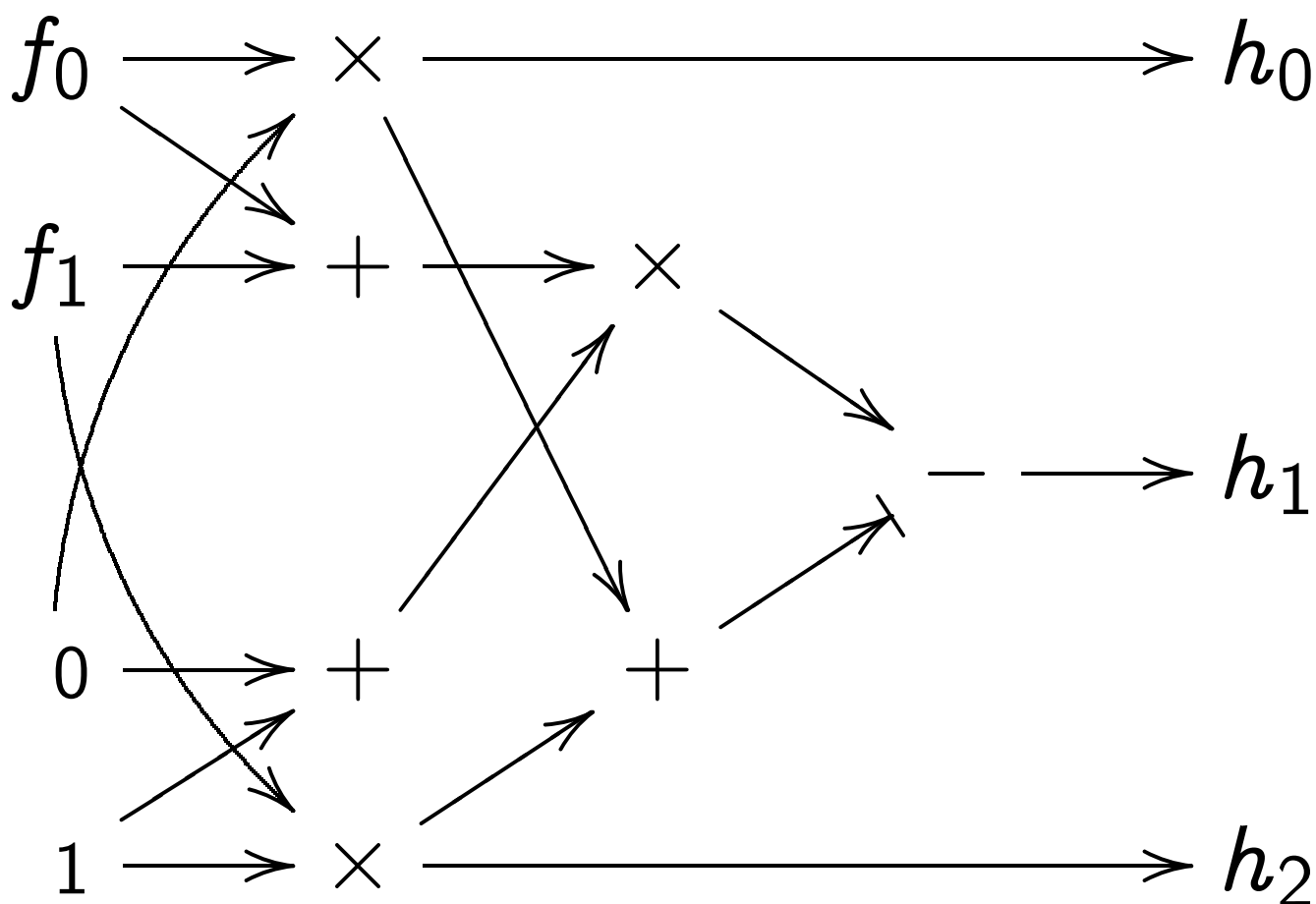
How quickly can we compute the
 n coeffs of fg , given f 's m coeffs
and g 's $n - m + 1$ coeffs?

Inputs $(f_0, f_1, \dots, f_{m-1}) \in \mathbf{R}^m$,
 $(g_0, g_1, \dots, g_{n-m}) \in \mathbf{R}^{n-m+1}$.

Output $(h_0, h_1, \dots, h_{n-1}) \in \mathbf{R}^n$
with $h_0 + h_1x + \dots + h_{n-1}x^{n-1} =$
 $(f_0 + f_1x + \dots)(g_0 + g_1x + \dots)$.

Assume **R**-algebraic algorithms
 (without divs, branches): chains of
 binary **R**-adds $u, v \mapsto u + v$,
 binary **R**-subs $u, v \mapsto u - v$,
 binary **R**-mults $u, v \mapsto uv$,
 starting from inputs and constants.

Example (1963 Karatsuba):



Cost measure for this talk:
total **R**-algebraic complexity.

Cost 1 for binary **R**-add;

cost 1 for binary **R**-sub;

cost 1 for binary **R**-mult;

cost 0 for constant in **R**.

Many real-world computations

use (e.g.) Pentium M's

floating-point operations

to approximate operations in **R**.

Properly scheduled operations

achieve Pentium M cycles

\approx total **R**-algebraic complexity.

Fast Fourier transforms

Define $\zeta_n \in \mathbf{C}$ as $\exp(2\pi i/n)$.

Define $T : \mathbf{C}[x]/(x^n - 1) \hookrightarrow \mathbf{C}^n$
as $f \mapsto f(1), f(\zeta_n), \dots, f(\zeta_n^{n-1})$.

Can very quickly compute T .

First publication of fast algorithm:
1866 Gauss.

Easy to see that Gauss's FFT uses
 $O(n \lg n)$ arithmetic operations
if $n \in \{1, 2, 4, 8, \dots\}$.

Several subsequent reinventions,
ending with 1965 Cooley Tukey.

Inverse map is also very fast.

Multiplication in \mathbf{C}^n is very fast.

1966 Sande, 1966 Stockham:

Can very quickly multiply

in $\mathbf{C}[x]/(x^n - 1)$ or $\mathbf{C}[x]$ or $\mathbf{R}[x]$
by mapping $\mathbf{C}[x]/(x^n - 1)$ to \mathbf{C}^n .

Given $f, \in \mathbf{C}[x]/(x^n - 1)$:

compute f as $T^{-1}(T(f)T(\))$.

Given $f, \in \mathbf{C}[x]$ with $\deg f < n$:

compute f from

its image in $\mathbf{C}[x]/(x^n - 1)$.

R-algebraic complexity $O(n \lg n)$.

A closer look at costs

More precise analysis of Gauss FFT
(and Cooley-Tukey FFT):

$\mathbf{C}[x]/(x^n - 1) \hookrightarrow \mathbf{C}^n$ using
 $(1/2)n \lg n$ binary \mathbf{C} -adds,
 $(1/2)n \lg n$ binary \mathbf{C} -subs,
 $(1/2)n \lg n$ binary \mathbf{C} -mults,
if $n \in \{1, 2, 4, 8, \dots\}$.

$(a, b) \in \mathbf{R}^2$ represents $a + bi \in \mathbf{C}$.

\mathbf{C} -add, \mathbf{C} -sub, \mathbf{C} -mult cost 2, 2, 6:

$$(a, b) + (c, d) = (a + c, b + d),$$

$$(a, b) - (c, d) = (a - c, b - d),$$

$$(a, b)(c, d) = (ac - bd, ad + bc).$$

Total cost $5n \lg n$.

Easily save some time:

eliminate mults by 1;

absorb mults by $-1, i, -i$

into subsequent operations;

simplify mults by $\pm\sqrt{\pm i}$

using, e.g., $(a, b)(1/\sqrt{2}, 1/\sqrt{2}) = ((a - b)/\sqrt{2}, (a + b)/\sqrt{2})$.

Cost $5n \lg n - 10n + 16$

to map $\mathbf{C}[x]/(x^n - 1) \hookrightarrow \mathbf{C}^n$,

if $n \in \{4, 8, 16, 32, \dots\}$.

What about cost of convolution?

$5n \lg n + O(n)$ to compute $T(f)$,

$5n \lg n + O(n)$ to compute $T(g)$,

$O(n)$ to multiply in \mathbf{C}^n ,

similar $5n \lg n + O(n)$ for T^{-1} .

Total cost $15n \lg n + O(n)$

to compute $fg \in \mathbf{C}[x]/(x^n - 1)$

given $f, g \in \mathbf{C}[x]/(x^n - 1)$.

Total cost $(15/2)n \lg n + O(n)$

to compute $fg \in \mathbf{R}[x]/(x^n - 1)$

given $f, g \in \mathbf{R}[x]/(x^n - 1)$: map

$\mathbf{R}[x]/(x^n - 1) \hookrightarrow \mathbf{R}^2 \oplus \mathbf{C}^{n/2-1}$

(Gauss) to save half the time.

1968 R. Yavne: Can do better!

Cost $4n \lg n - 6n + 8$

to map $\mathbf{C}[x]/(x^n - 1) \hookrightarrow \mathbf{C}^n$,

if $n \in \{2, 4, 8, 16, \dots\}$.

1968 R. Yavne: Can do better!

Cost $4n \lg n - 6n + 8$

to map $\mathbf{C}[x]/(x^n - 1) \hookrightarrow \mathbf{C}^n$,

if $n \in \{2, 4, 8, 16, \dots\}$.

2004 James Van Buskirk:

Can do better!

Cost $(34/9)n \lg n + O(n)$.

Expositions of the new algorithm:

Frigo, Johnson,

in *IEEE Trans. Signal Processing*;

Lundy, Van Buskirk,

in *Computing*;

Bernstein, this talk,

expanding an old idea of Fiduccia.

Van Buskirk, comp.arch,
January 2005: “Have you ever
considered changing djbbft to get
better opcounts along the lines of
[home.comcast.net/~kmbtib?](http://home.comcast.net/~kmbtib/)”

Van Buskirk, comp.arch,

January 2005: “Have you ever considered changing djbfft to get better opcounts along the lines of home.comcast.net/~kmbtib/?”

Bernstein, comp.arch: “What do you mean, ‘better opcounts’? The algebraic complexity . . . of a size- 2^k complex DFT has stood at $(3k - 3)2^k + 4$ additions and $(k - 3)2^k + 4$ multiplications since 1968.”

Van Buskirk, comp.arch,
January 2005: “Have you ever
considered changing djbfft to get
better opcounts along the lines of
home.comcast.net/~kmbtib?”

Bernstein, comp.arch: “What do
you mean, ‘better opcounts’? The
algebraic complexity . . . of a size- 2^k
complex DFT has stood at $(3k - 3)2^k + 4$ additions and $(k - 3)2^k + 4$
multiplications since 1968.”

Van Buskirk, comp.arch:
“Oh, you’re so 20th century, Dan.
Read the handwriting on the wall.”

Understanding the FFT

The FFT trick: $\mathbf{C}[x]/(x^n - 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/2} - 1) \oplus \mathbf{C}[x]/(x^{n/2} + 1)$
by unique $\mathbf{C}[x]$ -algebra morphism.
Cost $2n$: $n/2$ \mathbf{C} -adds, $n/2$ \mathbf{C} -subs.

e.g. $n = 4$: $\mathbf{C}[x]/(x^4 - 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^2 - 1) \oplus \mathbf{C}[x]/(x^2 + 1)$
by $g_0 + g_1x + g_2x^2 + g_3x^3 \mapsto$
 $(g_0 + g_2) + (g_1 + g_3)x,$
 $(g_0 - g_2) + (g_1 - g_3)x.$

Representation: $(g_0, g_1, g_2, g_3) \mapsto$
 $((g_0 + g_2), (g_1 + g_3)),$
 $((g_0 - g_2), (g_1 - g_3)).$

Recurse: $\mathbf{C}[x]/(x^{n/2} - 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/4} - 1) \oplus \mathbf{C}[x]/(x^{n/4} + 1);$
 similarly $\mathbf{C}[x]/(x^{n/2} + 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/4} - i) \oplus \mathbf{C}[x]/(x^{n/4} + i);$
 continue to $\mathbf{C}[x]/(x - 1) \oplus \dots$.

General case: $\mathbf{C}[x]/(x^n - \alpha^2) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/2} - \alpha) \oplus \mathbf{C}[x]/(x^{n/2} + \alpha)$
 by $g_0 + \dots + g_{n/2}x^{n/2} + \dots \mapsto$
 $(g_0 + \alpha g_{n/2}) + (g_1 + \alpha \dots)x + \dots,$
 $(g_0 - \alpha g_{n/2}) + (g_1 - \alpha \dots)x + \dots$

Cost $5n$: $n/2$ \mathbf{C} -mults,
 $n/2$ \mathbf{C} -adds, $n/2$ \mathbf{C} -subs.

Recurse, eliminate easy mults.

Cost $5n \lg n - 10n + 16$.

Alternative: the twisted FFT

After previous $\mathbf{C}[x]/(x^n - 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/2} - 1) \oplus \mathbf{C}[x]/(x^{n/2} + 1)$,
apply unique \mathbf{C} -algebra morphism
 $\mathbf{C}[x]/(x^{n/2} + 1) \hookrightarrow \mathbf{C}[y]/(y^{n/2} - 1)$
that maps x to $\zeta_n y$.

$$g_0 + g_1 x + \dots + g_{n/2} x^{n/2} + \dots \mapsto$$
$$(g_0 + g_{n/2}) + (g_1 + g_{n/2+1})x + \dots,$$
$$(g_0 - g_{n/2}) + \zeta_n (g_1 - g_{n/2+1})y + \dots$$

Again cost $5n$: $n/2$ \mathbf{C} -mults,
 $n/2$ \mathbf{C} -adds, $n/2$ \mathbf{C} -subs.

Eliminate easy mults, recurse.

Cost $5n \lg n - 10n + 16$.

The split-radix FFT

FFT and twisted FFT end up with
same number of mults by ζ_n ,
same number of mults by $\zeta_{n/2}$,
same number of mults by $\zeta_{n/4}$,
etc.

Is this necessary? No!

Split-radix FFT: more easy mults.

“Don’t twist until you see
the whites of their i ’s.”

(Can use same idea to speed up
Schönhage-Strassen algorithm
for integer multiplication.)

Cost $2n$: $\mathbf{C}[x]/(x^n - 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/2} - 1) \oplus \mathbf{C}[x]/(x^{n/2} + 1)$.

Cost n : $\mathbf{C}[x]/(x^{n/2} + 1) \hookrightarrow$
 $\mathbf{C}[x]/(x^{n/4} - i) \oplus \mathbf{C}[x]/(x^{n/4} + i)$.

Cost $6(n/4)$: $\mathbf{C}[x]/(x^{n/4} - i) \hookrightarrow$
 $\mathbf{C}[y]/(y^{n/4} - 1)$ by $x \mapsto \zeta_n y$.

Cost $6(n/4)$: $\mathbf{C}[x]/(x^{n/4} + i) \hookrightarrow$
 $\mathbf{C}[y]/(y^{n/4} - 1)$ by $x \mapsto \zeta_n^{-1} y$.

Overall cost $6n$ to split into
 $1/2, 1/4, 1/4$, entropy 1.5 bits.

Eliminate easy mults, recurse.

Cost $4n \lg n - 6n + 8$,

exactly as in 1968 Yavne.

The tangent FFT

Several ways to achieve cost 6 for mult by $e^{i\theta}$.

One approach: Factor $e^{i\theta}$ as $(1 + i \tan \theta) \cos \theta$.

Cost 2 for mult by $\cos \theta$.

Cost 4 for mult by $1 + i \tan \theta$.

For stability and symmetry, use $\max\{|\cos \theta|, |\sin \theta|\}$ instead of $\cos \theta$.

Surprise (Van Buskirk):

Can merge some cost-2 mults!

Rethink basis of $\mathbf{C}[x]/(x^n - 1)$.

Instead of $1, x, \dots, x^{n-1}$ use

$1/s_{n,0}, x/s_{n,1}, \dots, x^{n-1}/s_{n,n-1}$

where $s_{n,k} =$

$$\max\left\{\left|\cos\frac{2\pi k}{n}\right|, \left|\sin\frac{2\pi k}{n}\right|\right\}.$$

$$\max\left\{\left|\cos\frac{2\pi k}{n/4}\right|, \left|\sin\frac{2\pi k}{n/4}\right|\right\}.$$

$$\max\left\{\left|\cos\frac{2\pi k}{n/16}\right|, \left|\sin\frac{2\pi k}{n/16}\right|\right\}.$$

\dots

Now $(g_0, g_1, \dots, g_{n-1})$ represents

$$g_0/s_{n,0} + \dots + g_{n-1}x^{n-1}/s_{n,n-1}.$$

Note that $s_{n,k} = s_{n,k+n/4}$.

Note that $\zeta_n^k(s_{n/4,k}/s_{n,k})$ is

$\pm(1 + i \tan \dots)$ or $\pm(\cot \dots + i)$.

Cost $2n$:

$\mathbf{C}[x]/(x^n - 1)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(x^{n/2} - 1)$, basis $x^k / s_{n,k}$, \oplus

$\mathbf{C}[x]/(x^{n/2} + 1)$, basis $x^k / s_{n,k}$.

Cost n :

$\mathbf{C}[x]/(x^{n/2} - 1)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(x^{n/4} - 1)$, basis $x^k / s_{n,k}$, \oplus

$\mathbf{C}[x]/(x^{n/4} + 1)$, basis $x^k / s_{n,k}$.

Cost n :

$\mathbf{C}[x]/(x^{n/2} + 1)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(x^{n/4} - i)$, basis $x^k / s_{n,k}$, \oplus

$\mathbf{C}[x]/(x^{n/4} + i)$, basis $x^k / s_{n,k}$.

Cost $n/2 - 2$:

$\mathbf{C}[x]/(x^{n/4} - 1)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(x^{n/4} - 1)$, basis $x^k / s_{n/4,k}$.

Cost $n/2 - 2$:

$\mathbf{C}[x]/(x^{n/4} + 1)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(x^{n/4} + 1)$, basis $x^k / s_{n/2,k}$.

Cost $n/2$:

$\mathbf{C}[x]/(x^{n/4} + 1)$, basis $x^k / s_{n/2,k}$,

\hookrightarrow

$\mathbf{C}[x]/(x^{n/8} - i)$, basis $x^k / s_{n/2,k}$,

\oplus

$\mathbf{C}[x]/(x^{n/8} + i)$, basis $x^k / s_{n/2,k}$.

Cost $4(n/4) - 6$:

$\mathbf{C}[x]/(x^{n/4} - i)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(y^{n/4} - 1)$, basis $y^k / s_{n/4,k}$.

Cost $4(n/4) - 6$:

$\mathbf{C}[x]/(x^{n/4} + i)$, basis $x^k / s_{n,k}$, \hookrightarrow

$\mathbf{C}[x]/(y^{n/4} - 1)$, basis $y^k / s_{n/4,k}$.

Cost $4(n/8) - 6$:

$\mathbf{C}[x]/(x^{n/8} - i)$, basis $x^k / s_{n/2,k}$,

\hookrightarrow

$\mathbf{C}[x]/(y^{n/8} - 1)$, basis $y^k / s_{n/8,k}$.

Cost $4(n/8) - 6$:

$\mathbf{C}[x]/(x^{n/8} + i)$, basis $x^k / s_{n/2,k}$,

\hookrightarrow

$\mathbf{C}[x]/(y^{n/8} - 1)$, basis $y^k / s_{n/8,k}$.

Overall cost $8.5n - 28$ to split into $1/4, 1/4, 1/4, 1/8, 1/8,$
entropy $9/4$.

Recurse: $(34/9)n \lg n + O(n)$.

What if input is in $\mathbf{C}[x]/(x^n - 1)$
with usual basis $1, x, \dots, x^{n-1}$?

Could scale immediately,

but faster to scale upon twist.

Cost $(34/9)n \lg n - (124/27)n -$
 $2 \lg n - (2/9)(-1)^{\lg n} \lg n +$
 $(16/27)(-1)^{\lg n} + 8,$

exactly as in 2004 Van Buskirk.

Easily handle $\mathbf{R}[x]/(x^n + 1)$
by mapping to $\mathbf{C}[x]/(x^{n/2} - i)$.

Easily handle $\mathbf{R}[x]/(x^n - 1)$
by mapping to
 $\mathbf{R}[x]/(x^{n/2} - 1) \oplus \mathbf{R}[x]/(x^{n/2} + 1)$.

Cost $(17/9)n \lg n + O(n)$ for
 $\mathbf{R}[x]/(x^n - 1) \hookrightarrow \mathbf{R}^2 \times \mathbf{C}^{n/2-1}$,
so cost $(17/3)n \lg n + O(n)$
to compute $fg \in \mathbf{R}[x]/(x^n - 1)$
given $f, g \in \mathbf{R}[x]/(x^n - 1)$.

Cost $(17/3)n \lg n + O(n)$
for size- n convolution.

Open: Can $17/3$ be improved?