Algorithms for computing cyclotomic polynomials

Andrew Arnold

Centre for Experimental and Constructive Mathematics
Simon Fraser University

Thesis defence

Definition

The nth **cyclotomic polynomial**, $\Phi_n(z)$, *is the monic polynomial whose* $\phi(n)$ *distinct roots are the complex nth primitive roots of unity:*

$$\Phi_n(z) = \prod_{\substack{0 \le j < n \\ \gcd(j,n)=1}} (z - e^{2\pi i j/n}).$$

$\Phi_n(z)$ is an irreducible polynomial over $\mathbb{Q}$ with integer coefficients.

Definition

The nth **inverse cyclotomic polynomial**, $\Psi_n(z)$, *is the monic polynomial whose* $n - \phi(n)$ *roots are the nth nonprimitive roots of unity.*

We let the **order** of $\Phi_n(z)$ ($\Psi_n(z)$) be the number of distinct odd primes dividing n.

$$\Phi_1(z) = z - 1$$
$$\Phi_2(z) = z + 1$$
$$\Phi_3(z) = z^2 + z + 1$$
$$\Phi_4(z) = z^2 + 1$$
$$\Phi_5(z) = z^4 + z^3 + z^2 + z + 1$$
$$\Phi_6(z) = z^2 - z + 1$$
$$\Phi_7(z) = z^6 + z^5 + z^4 + z^3 + z^2 + z + 1$$
$$\Phi_8(z) = z^4 + 1$$
$$\Phi_9(z) = z^6 + z^3 + 1$$
$$\Phi_{10}(z) = z^4 - z^3 + z^2 - z + 1$$
$$\Phi_{11}(z) = z^{10} + z^9 + z^8 + z^7 + z^6 + z^5 + z^4 + z^3 + z^2 + z + 1$$
$$\Phi_{12}(z) = z^4 - z^2 + 1$$

Definition

We denote by $A(n)$ the height of $\Phi_n(z)$, i.e. the magnitude of its largest coefficient.

Example

E.g. As $\Phi_5(z) = 1 + z + z^2 + z^3 + z^4$, $A(5) = 1$.

Theorem (Erdős, 1961)

For every $c > 0$ there exists $n \in \mathbb{N}$ such that $A(n) > n^c$.

Question:

What is the least n such that $A(n) > n$? n^2? n^3? ...

Table: The least n for which $A(n) > n^c$

c	n	$A(n)$
1	*1181895	14102773
2	*43730115	86255063890874931
3	416690995	80103182105128365570406901971
4	*1880394945	64540997036010911566826446181523888971563
5,6	17917712785	8103387856491540894577 28164730920979670285722435974067632498282 7
7	99660932085	6126 72087174078366708962023243952601247252547 3 33815307867896175514937877391553644718537 0

*Monagan

To compute these results we needed to implement fast algorithms for computing $\Phi_n(z)$.

Some identities

1. Let p be a prime not dividing $m > 0$, then $\Phi_{mp}(z) = \Phi_m(z^p)/\Phi_m(z)$.

E.g. If p is prime, then
$\Phi_p(z) = \Phi_1(z^p)/\Phi_1(z) = (z^p - 1)/(z - 1) = z^{p-1} + z^{p-2} + \cdots + z + 1$.

2. Let q be a prime that divides m then $\Phi_{mq}(z) = \Phi_m(z^q)$.

E.g. $\Phi_9(z) = \Phi_3(z^3) = (z^3)^2 + (z^3) + 1 = z^6 + z^3 + 1$.

3. Let m be odd, then $\Phi_{2m}(z) = \Phi_n(-z)$.

E.g. $\Phi_6(z) = \Phi_3(-z) = (-z)^2 + (-z) + 1 = z^2 - z + 1$.

If $\bar{n}$ is the product of the distinct odd prime divisors of n, then
$A(n) = A(\bar{n})$.
We are namely intested in computing $\Phi_n(z)$ for odd, squarefree n.

The Fast Fourier Transform

1. Let p be a prime not dividing $m > 0$, then $\Phi_{mp}(z) = \Phi_m(z^p)/\Phi_m(z)$.

We can compute $\Phi_n(z)$ for $n = p_1 p_2 \cdots p_k$, a squarefree product of k distinct primes, using k exact polynomial divisions.

The Fast Fourier Transform

1. Let p be a prime not dividing $m > 0$, then $\Phi_{mp}(z) = \Phi_m(z^p)/\Phi_m(z)$.

We can compute $\Phi_n(z)$ for $n = p_1 p_2 \cdots p_k$, a squarefree product of k distinct primes, using k exact polynomial divisions.

Input: $n = p_1 p_2 \cdots p_k$, a product of k distinct odd primes
$m \longleftarrow p_1, \quad \Phi_m(z) \longleftarrow \sum_{i=0}^{p-1} z^i$
for $i \longleftarrow 2$ **to** k **do**
  $\Phi_{mp_i}(z) \longleftarrow \frac{\Phi_m(z^{p_i})}{\Phi_m(z)}$
  $m \longleftarrow m \cdot p_i$
end
return $\Phi_n(z)$

Classical polynomial division is quadratic-time. One obvious optimization would be to expedite division via the Fast Fourier Transform (FFT).

The Fast Fourier Transform

Definition

Let $N > 0$ be an integer and ω an N primitive root of unity modulo a prime q. The N-point **discrete Fourier transform** *of a polynomial $f(z)$ over $\mathbb{Z}_q$ is the evaluation of f at the powers of ω.*

$$\mathrm{DFT}(N, \omega, f) = (f(\omega^0), f(\omega^1), \ldots, f(\omega^{N-1})) \tag{1}$$

- If $N = 2^s$ is a power of 2, the FFT can compute $\mathrm{DFT}(N, \omega, f)$ from $f(z)$ in $\mathcal{O}(N \log(N))$ arithmetic operations mod q.
- The inverse FFT can interpolate $f(z)$ from $\mathrm{DFT}(N, \omega, f)$ (provided $N > \deg(f)$), also in $\mathcal{O}(N \log(N))$ operations.

The Fast Fourier Transform

Let $f(z), g(z) \in \mathbb{Z}_q[z]$ be such that $g(z)|f(z)$, and suppose we want to compute $h(z) = f(z)/g(z)$. We can

1. Obtain the N-point DFTs of f and g via the FFT, where N is the least power of 2 greater than $\deg(h)$.
2. Compute $h(\omega^j) = f(\omega^j)/g(\omega^j) \bmod q$ for $0 \leq j < N$
3. Interpolate h from its discrete Fourier transform via the inverse FFT

We need that $h(\omega^j) \neq 0$. This is not a problem when computing $\Phi_m(z^p)/\Phi_m(z)$.

Lemma

Let $m, N > 1$ be coprime and let ω be an Nth primitive root of unity mod q. Then $\Phi_m(\omega^j) \bmod q \neq 0$ for $j \in \mathbb{Z}$.

The cyclotomic Fourier tranform

We can directly compute the DFT of $\Phi_{mp}(z)$ from that of $\Phi_m(z)$:

$$\Phi_{mp}(\omega^j) = \Phi_m(\omega^{jp \bmod N})/\Phi_m(\omega^j)$$

We can compute the DFT of $\Phi_n(z)$, for squarefree $n = p_1 p_2 \cdots p_k$, as follows:

1. Let N be the least power of 2 greater than $\deg(\Phi_n(z)) = \phi(n)$.
2. Compute the N-point DFT of $\Phi_1(z) = z - 1$ (this can be done in linear-time).
3. For $1 \leq i \leq k$, compute the DFT of $\Phi_{p_1 \cdots p_i}(z)$ from that of $\Phi_{p_1 \cdots p_{i-1}}(z)$.

Here we can compute the DFT of $\Phi_n(z)$ in $\mathcal{O}(kN)$ arithmetic operations (as opposed to $\mathcal{O}(N \log N)$ via the FFT).
We call this approach the **cyclotomic Fourier transform (CFT)**.

The cyclotomic Fourier tranform

The CFT requires kN divisions in $\mathbb{Z}_q$.
We make further optimizations to the CFT that allow us to compute the DFT of $\Phi_n(z)$ using $\mathcal{O}(kN)$ multiplications as opposed to kN divisions.

Table: A comparison of times to compute $\Phi_n(z)$ modulo a 64-bit Fourier prime q.

n	FFT	CFT
1181895	0.83	0.42
3949491	3.93	1.42
15069565	42.59	13.52
43730115	94.83	29.69

The sparse power series algorithm

Identity

$$\Phi_n(z) = \prod_{d|n} (1 - z^d)^{\mu(n/d)} \qquad (\text{for} \quad n > 1).$$

E.g. $\Phi_{105}(z) = \dfrac{(1 - z^{105})(1 - z^3)(1 - z^5)(1 - z^7)}{(1 - z^{15})(1 - z^{21})(1 - z^{35})(1 - z)}$

- We call the $(1 - z^d)^{\pm 1}$ comprising $\Phi_n(z)$ the **subterms** of $\Phi_n(z)$.
- In this approach, we compute $\Phi_n(z)$ as a truncated power series.
- We can multiply a power series by $(1 - z^s)$ or $(1 - z^s)^{-1} = 1 + z^s + z^{2s} + \ldots$, truncated to degree D, in $\mathcal{O}(D)$ operations.

The sparse power series algorithm

The Sparse Power Series (SPS) Algorithm

Input: $n = p_1 p_2 \cdots p_k$, a product of k distinct primes

Output: The first half of the coefficients of $\Phi_n(z)$

$D \longleftarrow \phi(n)/2 + 1$, $a(0) \longleftarrow 1$

for $1 \leq i \leq D$ **do** $a(i) \longleftarrow 0$

for $d|n$ **do**

  if $\mu(\frac{n}{d}) = 1$ **then** // multiply by $1 - z^d$

    for $i = D$ **down to** d **by** -1 **do** $a(i) \longleftarrow a(i) - a(i-d)$

  else // divide by $1 - z^d$

    for $i = d$ **to** D **do** $a(i) \longleftarrow a(i) + a(i-d)$

  end

end

return $a(0), a(1), \ldots, a(\phi(n)/2)$

$n = p_1 p_2 \cdots p_k$, a product of k distinct primes, has 2^k divisors. Thus the SPS algorithm requires $\sim 2^k \phi(n)/2$ integer additions and subtractions.

We need only compute the first half of the coefficients of $\Phi_n(z)$, as the coefficients of $\Phi_n(z)$ are **palindromic**.

E.g. $\Phi_{15}(z) =$
$(\mathbf{1})z^8+(\mathbf{-1})z^7+(\mathbf{0})z^6+(\mathbf{1})z^5+(\mathbf{-1})z^4+(\mathbf{1})z^3+(\mathbf{0})z^2+(\mathbf{-1})z+(\mathbf{1})(z^0)$.

In general,

Lemma

If $f(z)$ is a product of cyclotomic polynomials

$$f(z) = \Phi_{n_1}(z)\Phi_{n_2}(z)\cdots\Phi_{n_s}(z),$$

such that n_i is odd for $1 \leq i \leq s$. Then if $\deg(f)$ is even, $f(z)$ has palindromic coefficients, else $f(-z)$ has palindromic coefficients (i.e. the coefficients of $f(z)$ are **anti-palindromic**).

The sparse power series algorithm

The SPS algorithm requires $\sim (2^k)(\phi(n)/2)$ additions and subtractions on our power series coefficients.

Table: A comparison of times to compute $\Phi_n(z)$

n	FFT	CFT	SPS
1181895	0.83	0.42	0.01
3949491	3.93	1.42	0.06
15069565	42.59	13.52	0.40
43730115	94.83	29.69	1.72

Improving the SPS algorithm

- Suppose that we iterate through the divisors of $n = p_1 p_2 \cdots p_k$ in the order $d_1, d_2, d_3, \ldots$.
- The SPS algorithm computes $f_s(z) = \prod_{i=1}^{s} (1 - z^{d_i})^{\mu(n/d_i)}$, for $1 \leq s \leq 2^k$, as an intermediate result.
- If $f_s(z)$ is a polynomial of degree D_s, then either $f(z)$ or $f(-z)$ is palindromic.
- In order to easily obtain all the coefficients of f_s, we need only truncate to degree $D_s/2$ when multiplying by the subterms $(1 - z^d)^{\pm 1}$ comprising f_s (as opposed to $\phi(n)/2$.

Idea to improve the SPS algorithm

Order the divisors of n in a manner which minimizes the degree we need truncate to (we call if the **degree bound**) over the computation of $\Phi_n(z)$.

1st improvement: SPS2

Let $n = mp$ be an odd, squarefree integer with greatest prime divisor p.

$$\Phi_n(z) = \frac{\Phi_m(z^p)}{\Phi_m(z)} = \Psi_m(z)\Phi_m(z^p)\frac{1}{z^m - 1} \tag{2}$$

(Recall $\Psi_m(z) = (z^m - 1)/\Phi_m(z)$).
We compute $\Phi_n(z)$ as follows:

1. Compute $\Psi_m(z)$ as a product of subterms, truncating to degree $m - \phi(m)/2$.
2. Use the palindromic property to get the higher-degree terms of $\Psi_m(z)$.
3. Multiply by the remaining subterms, truncating to degree $\phi(n)/2$.

We call this approach **SPS2**

1st improvement: SPS2

$$\Phi_n(z) = \Psi_m(z)\Phi_m(z^p)\frac{1}{z^m-1}$$

If $\Phi_n(z)$ is of order k, then SPS2 requires less than

$$(2^{k-1}-1)(m-\phi(m)/2)+(2^{k-1}+1)(\phi(n)/2)$$

arithmetic operations on our power series coefficients. This is roughly half that of the original SPS algorithm.

Table: Times to compute $\Phi_n(z)$

n	$k = \text{order}(\Phi_n(z))$	SPS	SPS2
1073741829	3	4.72	3.25
1073741837	4	11.09	6.63
1073741835	5	16.07	7.96
1073742117	6	25.09	13.58
1073746605	7	45.80	25.72
1074800265	8	74.00	38.97

The iterative and recursive SPS algorithms

We can express $\Phi_n(z)$ as a product of inverse cyclotomic polynomials of decreasing index, multiplied by some additional subterms,

Let $n = p_1 p_2 \cdots p_k$ be an odd, squarefree product of k primes. For $1 \leq i \leq k$, let $m_i = p_1 p_2 \cdots p_{i-1}$ and $e_i = p_{i+1} \cdots p_k$. Then

$$\Phi_n(z) = \Psi_{m_k}(z^{e_k}) \cdots \Psi_{m_1}(z^{e_1}) \cdot \left(\prod_{j=1}^{k} (z^{n/p_j} - 1)^{-1} \right) \cdot (z^n - 1) \qquad (3)$$

Example

$$\Phi_{105}(z) = \Psi_{15}(z)\Psi_3(z^7) \cdot (z^{15} - 1)^{-1}(z^{21} - 1)^{-1}(z^{35} - 1)^{-1} \cdot (z^{105} - 1)$$

In the **iterative SPS** or **SPS3** we compute the product (3) from left to right, raising the degree bound as necessary.

The iterative and recursive SPS algorithms

We can, furthermore, break inverse cyclotomic polynomials into products of cyclotomic polynomials.

Again, given $n = p_1 p_2 \cdots p_k$, an odd, squarefree product of k primes, let, for $1 \leq i \leq k$, let $m_i = p_1 p_2 \cdots p_{i-1}$ and $e_i = p_{i+1} \cdots p_k$. Then

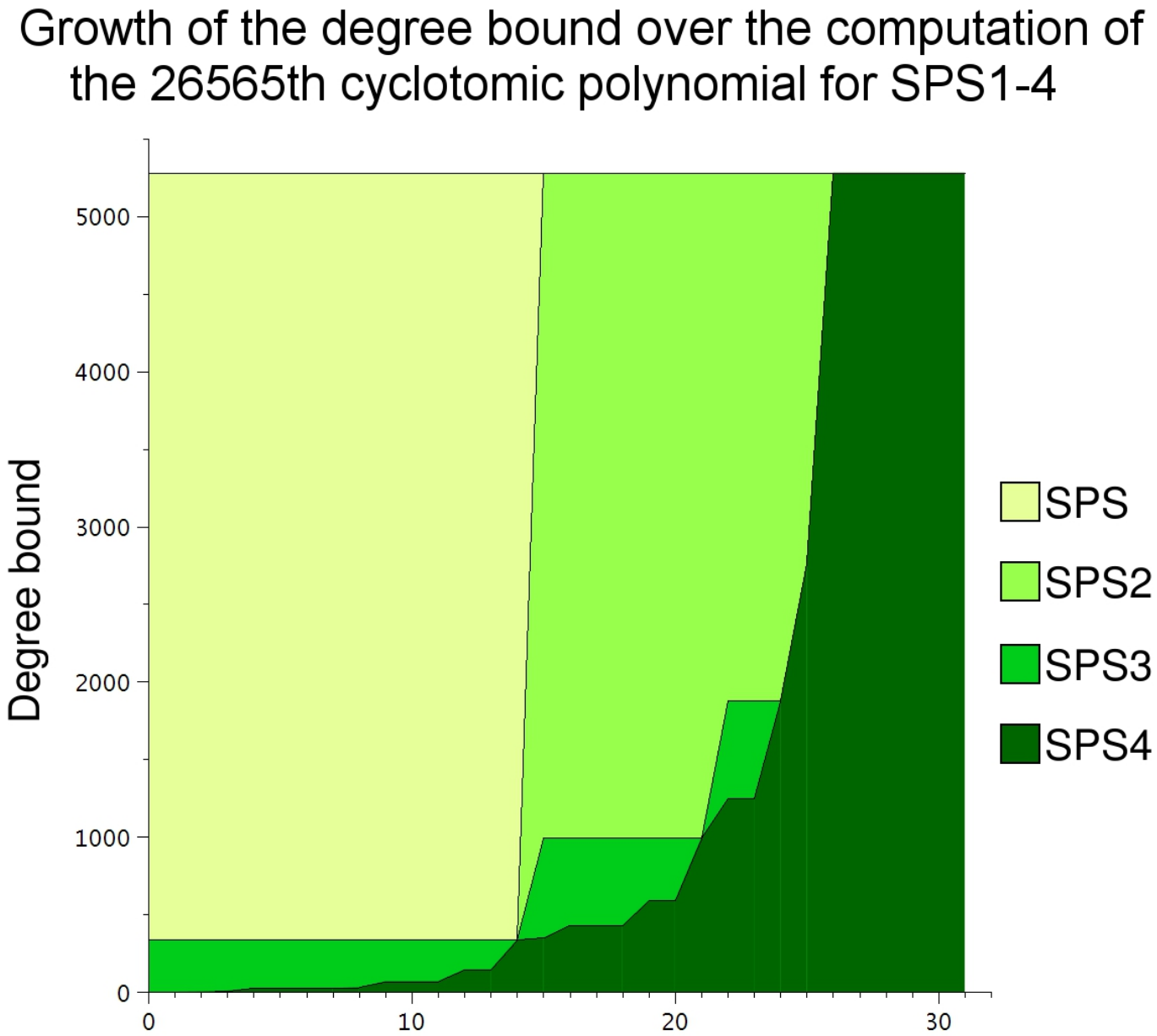
$$\Phi_{m_k}(z^{e_k}) \cdots \Phi_{m_1}(z^{e_1}) \Psi_{m_1}(z^{e_1})$$

Example

$$\Psi_{105}(z) = \Phi_{15}(z)\Phi_5(z^7)\Phi_1(z^{35})$$

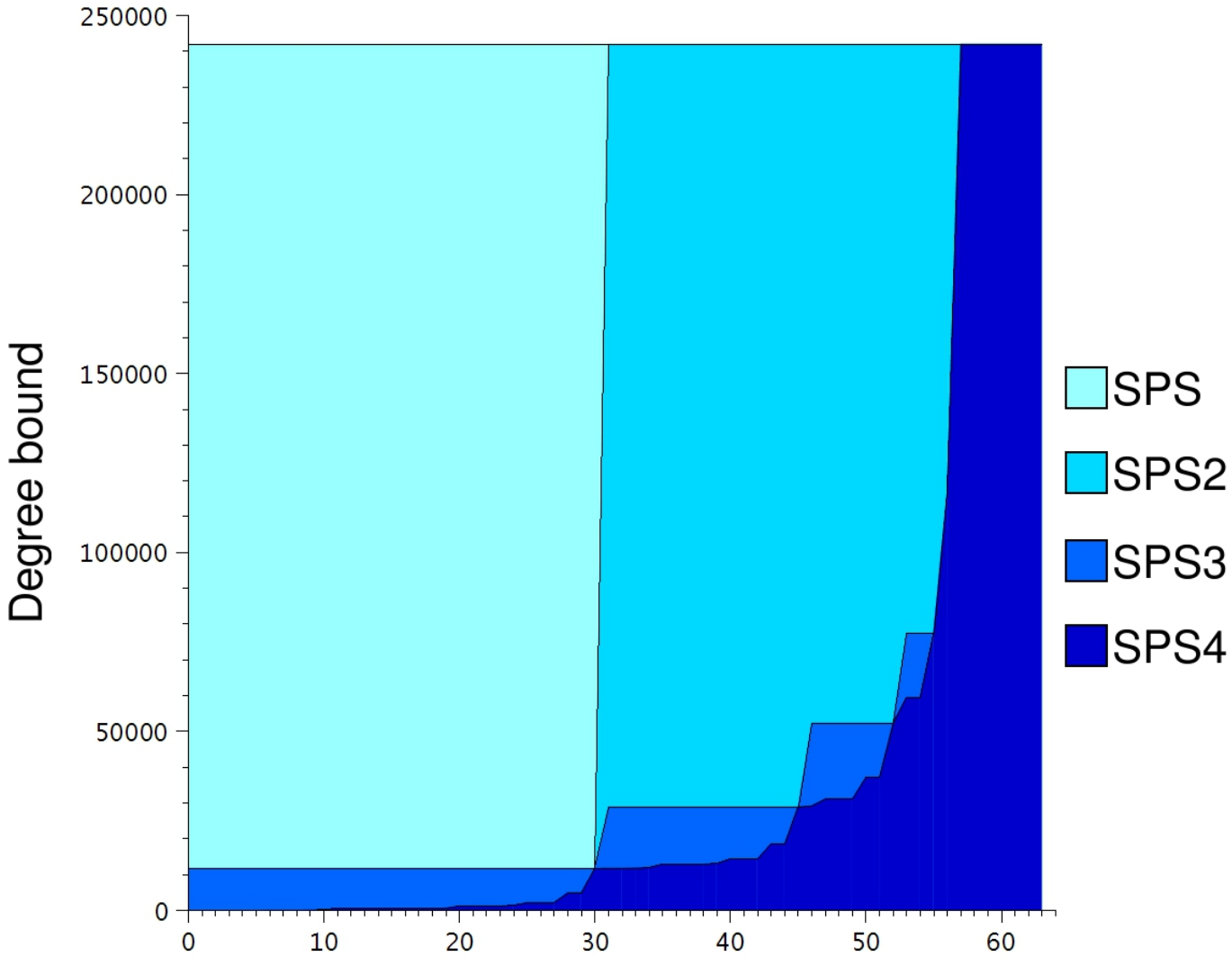
In the **recursive SPS** or **SPS4**, we recursively break cyclotomic polynomials and inverse cyclotomic polynomials into products as much we can.

The iterative and recursive SPS algorithms

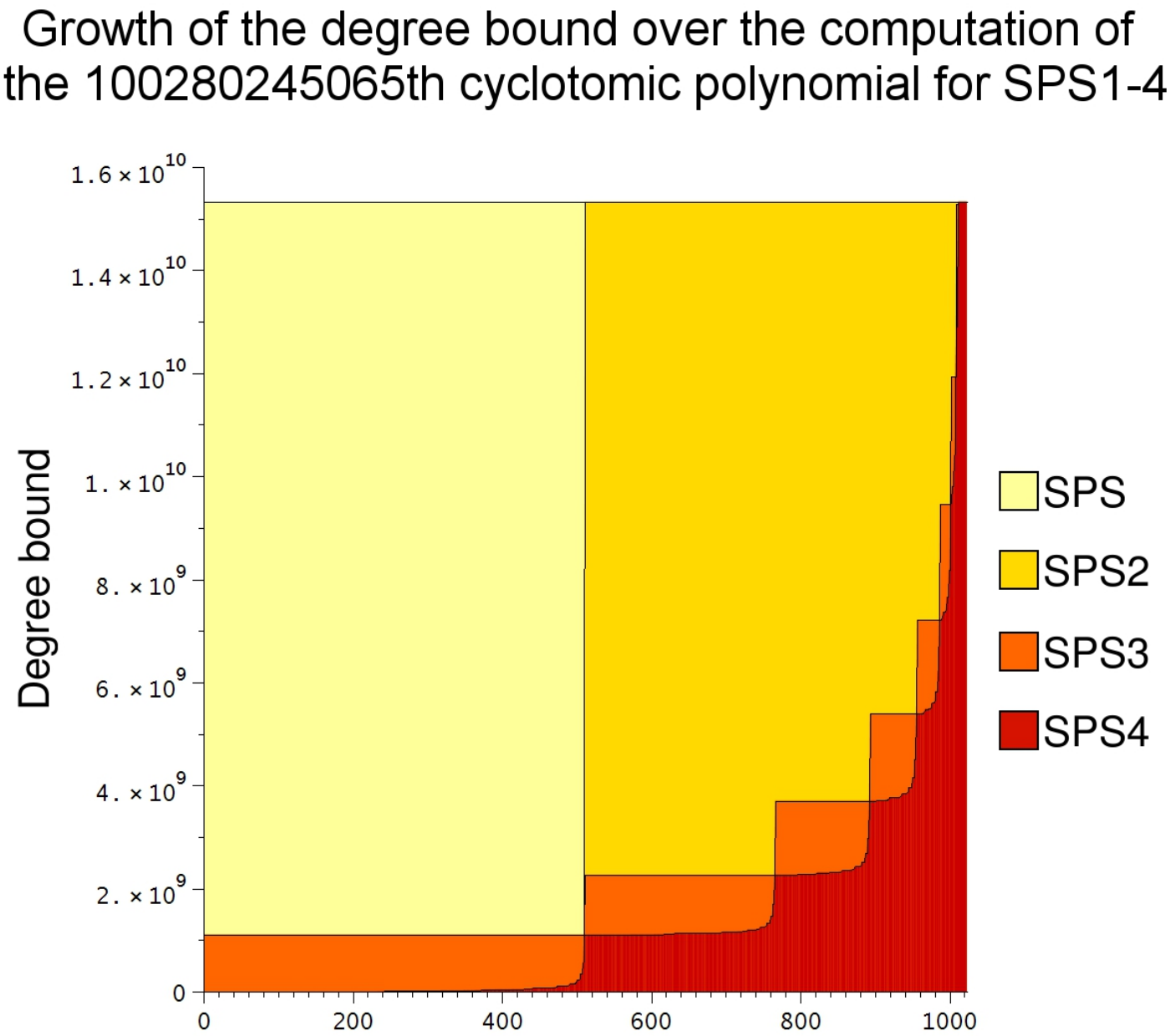


The iterative and recursive SPS algorithms

Growth of the degree bound over the computation of the 1181895th cyclotomic polynomial for SPS1-4.



The iterative and recursive SPS algorithms



The iterative and recursive SPS algorithms

Table: Times to compute $\Phi_n(z)$

n	$k = \text{order}(\Phi_n(z))$	SPS	SPS2	SPS3	SPS4
1073741829	3	4.72	3.25	1.92	2.20
1073741837	4	11.09	6.63	3.87	4.32
1073741835	5	16.07	7.96	2.32	2.69
1073742117	6	25.09	13.58	3.18	3.37
1073746605	7		25.72	4.23	4.27
1074800265	8	74.00	38.97	10.19	6.65

The big prime algorithm

If $\Phi_n(z)$ can fit in main memory, it is easy to compute.

What if we want to compute $A(n)$, but $\Phi_n(z)$ cannot fit in main memory? What if $\Phi_n(z)$ cannot fit on a hard disk?

E.g. Nathan Kaplan asked us to compute $A(n)$ for $n = 2,576,062,979,535 = 3 \cdot 5 \cdot 29 \cdot 2609 \cdot 2269829$. The degree of $\Phi_n(z)$ is $1,326,015,358,976$. Kaplan was looking for $\Phi_n(z)$ of order five such that $A(n) = 1$.

The big prime algorithm

Let $n = mp$, and write $\Phi_m(z^p)\Phi_m(z) = \sum c(i)z^i$ & $\Phi_n(z) = \sum a(i)z^i$. As

$$\Phi_n(z) = \Phi_m(z^p)\Phi_m(z) \cdot \tfrac{1}{z^m-1} = (\sum c(i)z^i)(-1 - z^m - z^{2m} - \dots),$$

We have that

$$a(t) = -\sum_{0 \le s \le t, m|(t-s)} c(s) = a(s) - c(t)$$

- We compute all the values $a(t)$ incrementally.
- We use an array of m integers $[\bar{a}(0), \bar{a}(1), \dots, \bar{a}(m-1)]$, and store $a(t)$ in $\bar{a}(t \bmod m)$.
- When we want to compute $a(t+m)$ we write it to $\bar{a}(t \bmod m)$ and discard the value $a(t)$.

The big prime algorithm

Let $n = mp$, and write $\Phi_m(z^p)\Phi_m(z) = \sum c(i)z^i$ a $\Phi_n(z) = \sum a(i)z^i$.

$$a(t) = -\sum_{0 \le s \le t, m|(t-s)} c(s) = a(s) - c(t)$$

The big prime algorithm to compute $A(n)$

$\bar{a}(0), \bar{a}(1), \ldots, \bar{a}(m-1) \longleftarrow 0, 0, \ldots, 0, \qquad H \longleftarrow 0.$

for $t \leftarrow 0$ **to** $\phi(n)/2$ **do**

  Compute $c(t)$ from $\Phi_m(z)$ and $\Psi_m(z)$

  $\bar{a}(t \bmod N) \longleftarrow \bar{a}(t \bmod N) - c(t)$

  if $|\bar{a}(t \bmod N)| > H$ **then** $H \longleftarrow |\bar{a}(t \bmod N)|$

end

return H

As $\deg(\Phi_m), \deg(\Psi_m) < m$, we can compute all the coefficients $c(t)$ of $\Phi_m(z^p)\Psi_m(z)$, given $\Phi_m(z)$ and $\Psi_m(z)$ in $\mathcal{O}(m^2)$, integer arithmetic operations.

The big prime algorithm

This algorithm was used to look for $\Phi_n(z)$ of order 5 with $A(n) = 1$.
For these cyclotomic polynomials, $\Psi_m(z)$ and $\Phi_m(z)$ are typically sparse, and it saves time to use a sparse representation for $\Psi_m(z)$ and $\Phi_m(z)$.

Table: Time to compute $A(n)$ using the big prime algorithm with 8-bit integers and sparse representations of $\Psi_m(z)$ and $\Phi_m(z)$

n	factorization of n	time
746443728915	$3 \cdot 5 \cdot 31 \cdot 929 \cdot 1727939$	0.27
1147113361785	$3 \cdot 5 \cdot 29 \cdot 1741 \cdot 1514671$	0.51
36654908721735	$3 \cdot 5 \cdot 29 \cdot 6959 \cdot 12108659$	1.90
117714212390685	$3 \cdot 5 \cdot 59 \cdot 3539 \cdot 37584179$	1.97
134926610295958 5	$3 \cdot 5 \cdot 59 \cdot 8849 \cdot 172290029$	4.92

We found many examples of $\Phi_n(z)$ of order 5 and height 2, but none of height 1.

A low-memory SPS algorithm

- The big prime algorithm is not effective for $\Phi_n(z)$ without a large prime factor.
- The SPS algorithms are faster for most cyclotomic polynomials, but slow considerably when we cannot fit $\Phi_n(z)$ in main memory.

Aim

Compute the coefficients of $\Phi_n(z)$ in a manner with the same computational complexity as one of the SPS algorithms, while limiting disk I/O.

A low-memory SPS algorithm

If we compute the coefficients of $\Phi_m(z^p)\Psi_m(z) = \sum c(i)z^i$ (in some order), then we can multiply by $(z^m - 1)^{-1} = -1 - z^m - z^{2m} - \ldots$ using an array of m integers as we did with the big-prime algorithm.
We can break $\Psi_m(z)$ into polynomials $f_j(z)$, $0 \leq j < p$ such that

$$\Psi_m(z) = \sum_{j=0}^{p-1} z^j f_j(z^p),$$

In which case,

$$\Psi_m(z)\Phi_m(z^p) = \sum_{j=0}^{p-1} z^j f_j(z^p)\Phi_m(z^p),$$

thus, to compute the coefficients of $\Psi_m(z)\Phi_m(z^p)$, we can compute $f_j(z)\Phi_m(z)$ for $0 \leq j < p$.

A low-memory SPS algorithm

If $l = \deg(\Psi_m) - j \bmod p$, then $z^j f(z^p) + z^l f(z^p)$ has (anti)palindromic polynomial coefficients. We can leverage this to multiply both f_j and f_l by Φ_m in a manner similar to the recursive SPS algorithm (where we compute only half the coefficients of any intermediate polynomial).

1. Compute $\Psi_m(z)$ via SPS4.
2. Break Ψ_m into polynomials f_j such that $\Psi_m(z) = \sum z^j f_j(z^p)$.
3. For $0 \leq j < p$, if $l \leq j$, multiply both $f_j(z)$ and $f_l(z)$ by $\Phi_m(z)$ and write the resulting coefficients to disk.
4. Multiply $\Psi_m(z)\Phi_m(z^p) = \sum c(i)z^i$ by $(z^m - 1)$ in a manner similar to the big prime algorithm.

We used this method to compute many $\Phi_n(z)$ for many $n > 10^{10}$.

Table: n such that $A(n) > A(m)$ for $m < n$

n	$A(n)$
1	1
105	2
385	3
1365	4
1785	5
2805	6
3135	7
6545	9
10465	14
11305	23
17255	25
20615	27
26565	59
40755	359
106743	397
171717	434

FFT-based, original SPS, low-memory SPS algorithm

Table: n such that $A(n) > A(m)$ for $m < n$

n	$A(n)$
255255	532
279565	1182
327845	31010
707455	35111
886445	44125
983535	59815
1181895	14102773
1752465	14703509
3949491	56938657
8070699	74989473
10163195	1376877780831
13441645	1475674234751
15069565	1666495909761
30489585	2201904353336
37495115	2286541988726
40324935	2699208408726

FFT-based, original SPS, low-memory SPS algorithm

Table: n such that $A(n) > A(m)$ for $m < n$

n	$A(n)$
43730115	86255063889087493 1
169828113	31484567640915734941
185626077	42337944402802720258
416690995	80103182105128365570406901971
437017385	86711753206816303264095919005
712407185	11185937095152669880319825792 5
1250072985	13756580004264445418853130688 6
1311052155	19289231441599758355173100941 0
1880394945	64540997036010911566826446181523888971563
2317696095	67075962666923019823602030663153118803367
7981921311	45433611853877309220963701599924010686327284 1
12436947159	63362031348392041042436427665367419759880499 5
17917712785	81033 87856491540894577281647309209796702857224359740676324982827
22084622735	94928 13291464815330681848221029648678194321867848264652910092651

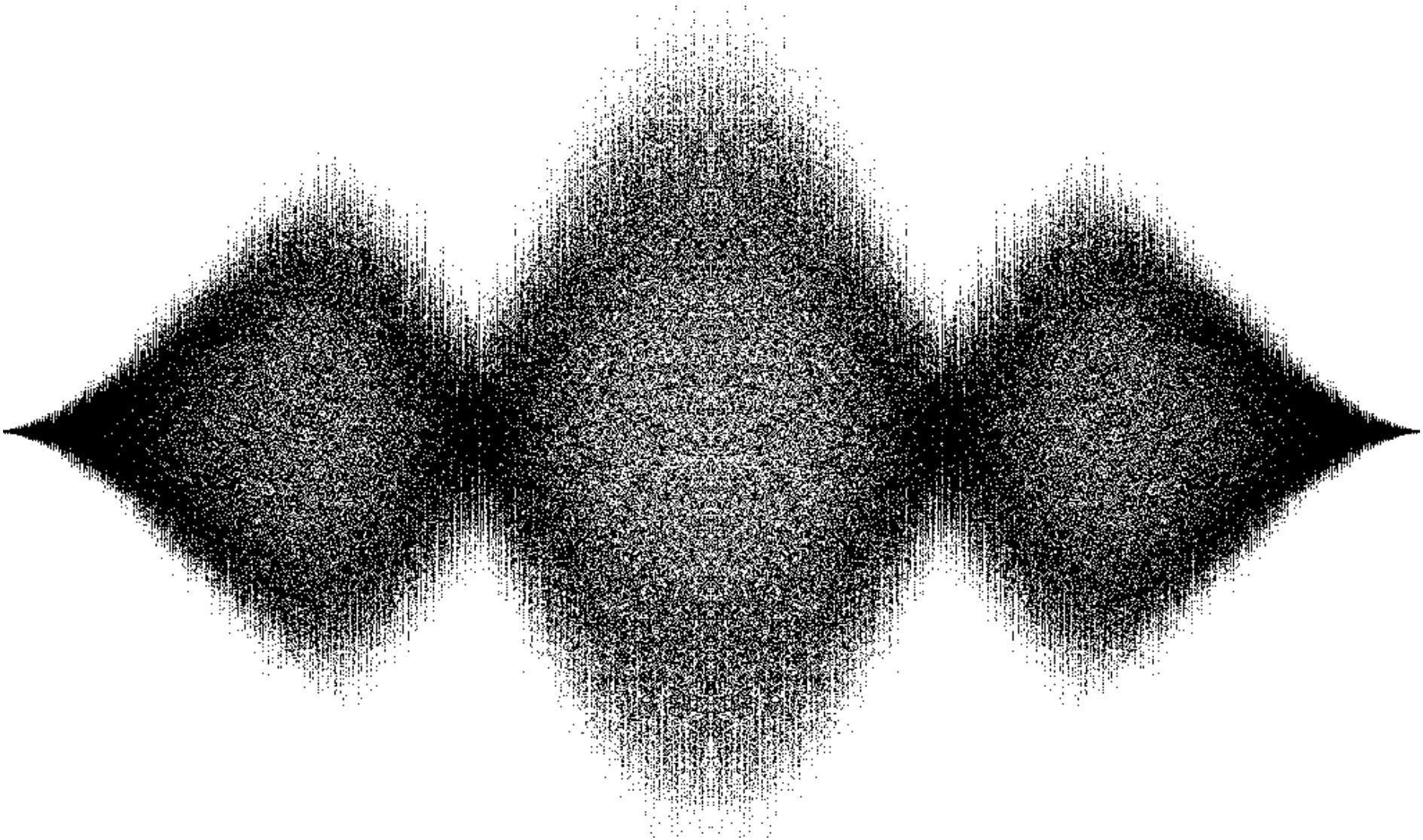
FFT-based, original SPS, low-memory SPS algorithm

Table: n such that $A(n) > A(m)$ for $m < n$

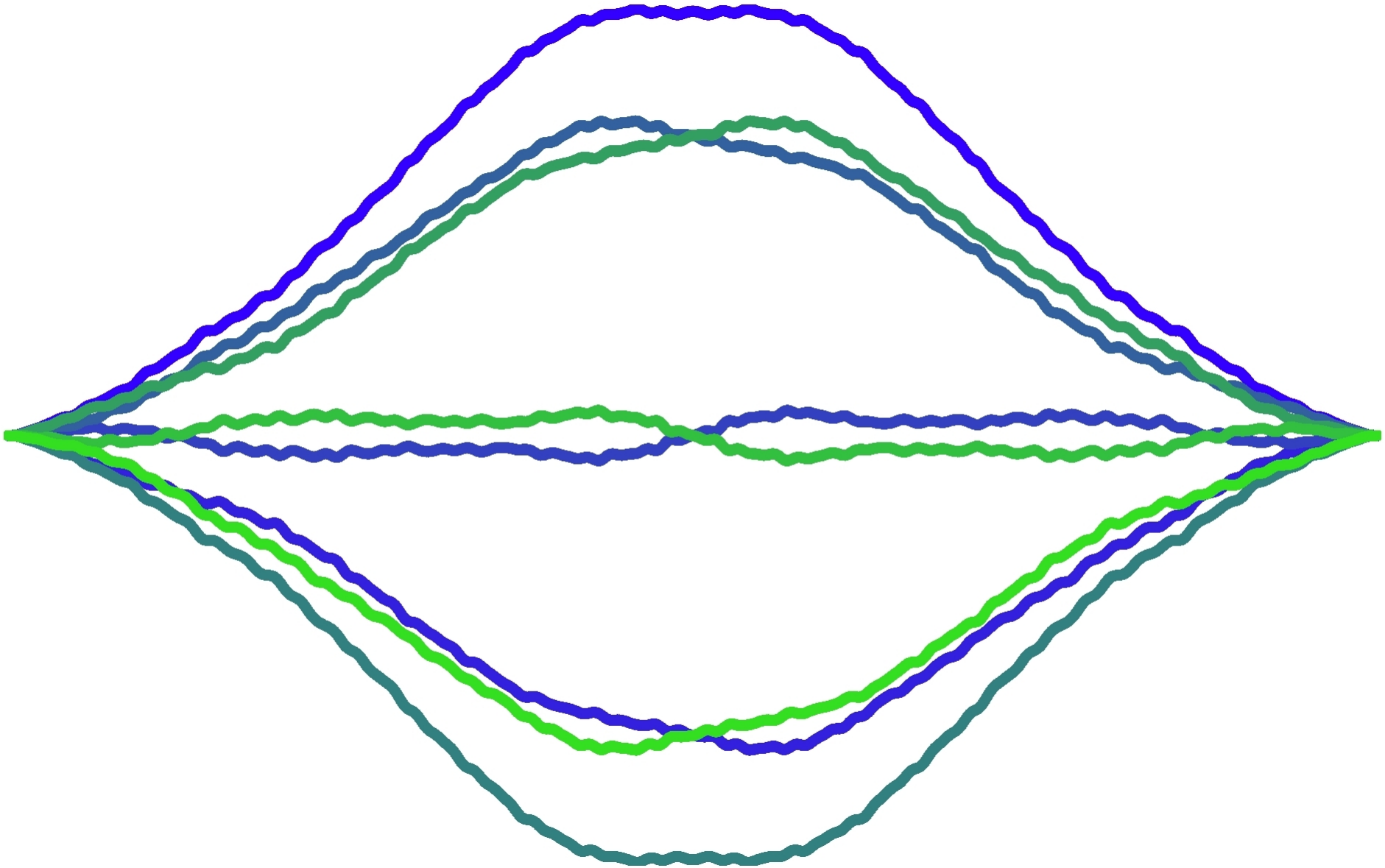
n	$A(n)$
53753138355	185020 439179865837393212415265919539992363783830789874059256100 51
66253868205	186123 630445073227618614172155463629537535125344944705583274334 58
99660932085	612672087174078366708962023243952601247252547333815307867896175514937877391553644718537 0

FFT-based, original SPS, low-memory SPS algorithm

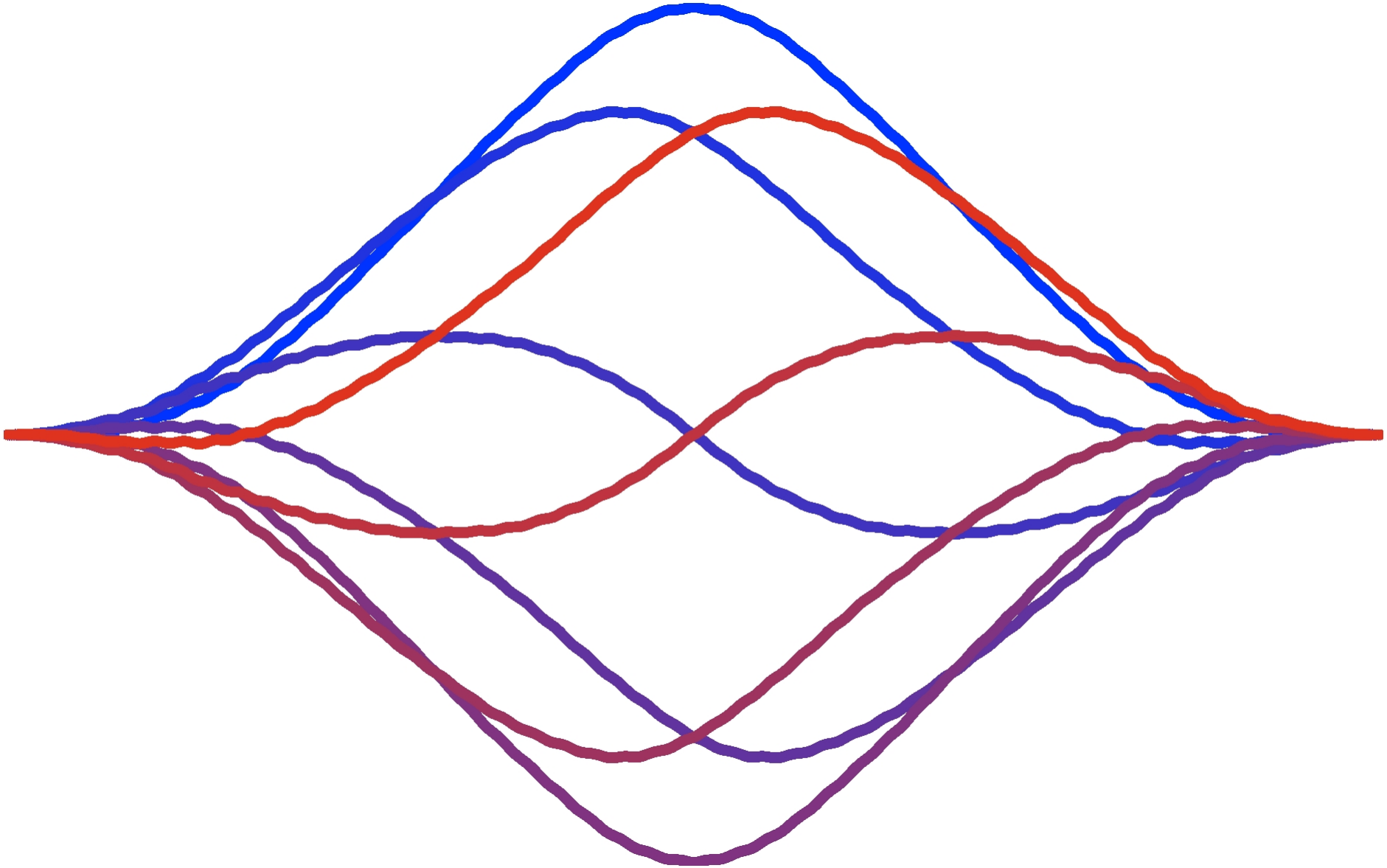
The coefficients of $\Phi_{4849845}(z)$



Plots of $\Phi_{1181895}(z)$



Plots of $\Phi_{43730115}(z)$



The coefficients of $\Phi_{40324935}(z)$