Algebraic Adventures in Python

Adrian Lomas

1 Implementation of the *FFT* algorithm

1.1 Vandermonde Systems

1.2 Multiplication and the Fast Fourier Transform

The Discrete Fourier Transform of a polynomial of degree strictly less than n (resp. of an n-dimensional vector) is the evaluation at every $\zeta_n^k = \exp(2ik\pi/n)$ for $0 \leq k \leq n-1$ of this polynomial (resp. of the polynomial whose coefficients are the vector's coordinates, in increasing degree order, although this is an arbitrary choice).
More formally, if $P(x) = \sum_{0\leq k\leq n-1} a_k x^k$ (and a_{n-1} is not necessarily non-zero) or if $S = (a_k)_{0\leq k\leq n-1}$ is an n-dimensional vector, then

$$\widehat{P}(j) = \widehat{S}(j) = \sum_{0\leq k\leq n-1} a_k e^{2\pi kj/n} = \sum_{0\leq k\leq n-1} a_k \zeta_n^{kj}$$

And the discrete Fourier transform of P or S is the vector

$$\widehat{P} = \widehat{S} = \left(\widehat{S}(j)\right)_{0\leq j\leq n-1}$$

The Fast Fourier Transform or FFT is an algorithm that evaluates the discrete Fourier transform of a polynomial in $O(n \log n)$ time instead of the usual $O(n^2)$ time one would get by evaluating each $P(\zeta_n^k)$ one after the other.
This is done by taking advantage of the fact that P can be separated in "even" and "odd" polynomials verifying the equation $P(x) = A(x^2) + x \cdot B(x^2)$ and noticing that if n is a power of 2 then $\zeta_n^{k+n/2} = -\zeta_n^k$ so that one only needs to know the values of $A(\zeta_n^{2k})$ and $B(\zeta_n^{2k})$ up to $k = n/2 - 1$ to determine the n values ζ_n^k for k ranging from 0 to $n-1$, which is done by applying the FFT algorithm to A and B for half the coefficients, once more and recursively.

P does not need to be of degree $n-1$ with n a power of 2 necessarily; if it is not the case, one can always pad P with missing 0 coefficients above its degree.
Say $\deg P = d < n = 2^m$, and S is a d-dimensional vector with the coefficients of P as coordinates, we can write

$$S = (a_k)_{0 \leq k \leq d} \longrightarrow S_{\text{new}} = (a_0, \cdots, a_d, 0, \cdots, 0) \quad \text{a vector with } n \text{ coordinates}$$

Naturally, FFT works around the need to evaluate n roots of unity whose rectangular form contains sines and cosines that are almost never rational numbers (at least evaluated at angles $2\pi k/n$ with n a power of 2), this raises the question of the accuracy at which these trigonometric functions need to be evaluated in order for the FFT algorithm to result in an output close enough to the exact discrete Fourier transform, in a metric which we will explicit.
We consider polynomials as interchangeable to vectors with n coordinates, so vector norms are appropriate, in particular we will use $||\cdot||_1$, $||\cdot||_2$ and $||\cdot||_\infty$ norms and see that coefficient-wise accuracy is best evaluated with the $||\cdot||_\infty$ norm, in fact if the $||\cdot||_\infty$ norm of the difference of two vectors is bound by some $\varepsilon > 0$ then every difference of two coordinates is bound by this very ε.

More precisely, the goal of this section is to prove the following theorem:

Theorem 1.1. *Let $A, B \in \mathbf{Z}[X]$ and $C = A \times B$ of degree $\deg C < n = 2^m \in \mathbf{N}$.*
Let $\zeta = \zeta_n + \varepsilon_n$ with $\varepsilon_n = (1/\alpha!) \cdot (2\pi i/n)^\alpha + o(n^{-\alpha})$ (resp. $\zeta_{inv} = \zeta_n^{-1} + \varepsilon'_n$ with $\varepsilon'_n = (1/\alpha!) \cdot (-2\pi i/n)^\alpha + o(n^{-\alpha})$) for some $\alpha \in \mathbf{N}$ and $C_{err}(j) = \frac{1}{n} \sum_{0 \leq k \leq n-1} (\sum_{0 \leq i \leq n-1} c_i \zeta^{ki}) \zeta_{inv}^{-kj}$, if

$$\log \|C\|_\infty + 3 \log n + \log 2 < \alpha \left(\log n + \frac{1}{\alpha} \log(\alpha!) - \log(2\pi) \right)$$

then $\|C - C_{err}\|_\infty < 1/2$ (with C seen as a vector rather than a polynomial) so that rounding the coefficients of C_{err} gives us C exactly.

Alternatively, if A and B are polynomials with integer coefficients, then the algorithm to compute their product C via FFT should give us back the exact integer coefficients of C given that ζ_n is evaluated with a Taylor series at order $\alpha - 1$.

Lemma 1.2. *Suppose $\alpha > 3$. If $0 \leq k \leq n^2$, then we have the following*

1. $\zeta^k = \zeta_n^k + \varepsilon_n k \zeta_n^{k-1} + o(n^{-\alpha+2})$
2. $\zeta_{inv}^k = \zeta_n^{-k} + \varepsilon'_n k \zeta_n^{1-k} + o(n^{-\alpha+2})$
3. $\zeta \times \zeta_{inv} = 1 + \varepsilon_n(\zeta_n^{-1} + (-1)^\alpha \zeta_n) + o(n^{-\alpha})$ *and moreover* $\zeta \times \zeta_{inv} = 1 + o(n^{-\alpha})$ *if α is even*

Proof. Since

$$\begin{aligned}\zeta^k &= (\zeta_n + \varepsilon_n)^k \\ &= \zeta_n^k + \varepsilon_n k\zeta_n^{k-1} + \sum_{2\leq j\leq k}\binom{k}{j}\varepsilon_n^j\zeta_n^{k-j}\end{aligned}$$

Then we just have to prove that each term in the series is $o(n^{-\alpha})$ so we can sum them up to get $o(n^{-\alpha+2})$, since there are at most n^2 terms.
Indeed,

$$\begin{aligned}\left|\binom{k}{j}\varepsilon_n^j\zeta_n^{k-j}\right| &\leq |k^j|\cdot|\varepsilon_n|^j \\ &\leq n^{2j}\cdot o(n^{-j\alpha+j}) = o(n^{-j\alpha+3j})\end{aligned}$$

For $j = 2$ this is $o(n^{-2\alpha+6})$ which is $o(n^{-\alpha+2})$ when $\alpha \geq 4$, and for $j = 3$ this is $o(n^{-3\alpha+6})$ which is $o(n^{-\alpha+2})$ when $\alpha \geq 7/2$, so $\alpha \geq 4$ is enough, we can then take these two terms outside of the sum, and for $j \geq 4$, the term is already $o(n^{-\alpha})$, as we can see from

$$\begin{aligned}-j\alpha + 3j &\leq -\alpha \\ \iff 3(j-1)+3 &\leq \alpha(j-1) \\ \iff 3 + \frac{3}{j-1} &\leq \alpha \\ \iff 4 &\leq \alpha\end{aligned}$$

Since $\sup_{j\geq 4}\left\{3 + \frac{3}{j-1}\right\} = 4$. Hence

$$\begin{aligned}\zeta^k &= \zeta_n^k + \varepsilon_n k\zeta_n^{k-1} + 2\times o(n^{-\alpha+2}) + \sum_{4\leq j\leq k} o(n^{-\alpha}) \\ &= \zeta_n^k + \varepsilon_n k\zeta_n^{k-1} + o(n^{-\alpha+2}) + (k-3)\cdot o(n^{-\alpha}) \\ &= \zeta_n^k + \varepsilon_n k\zeta_n^{k-1} + o(n^{-\alpha+2})\end{aligned}$$

Given that $k \leq n$.
The proof for ζ_{inv}^k uses exactly the same arguments, since $\varepsilon'_n = (-1)^\alpha\varepsilon_n$ the arguments regarding the order do not change.
As for the third point, it suffices to write it out:

$$\begin{aligned}\zeta\times\zeta_{\text{inv}} &= (\zeta_n + \varepsilon_n)\times(\zeta_n^{-1} + \varepsilon'_n) \\ &= \zeta_n\times\zeta_n^{-1} + \varepsilon_n\zeta_n^{-1} + \varepsilon'_n\zeta_n + \varepsilon_n\varepsilon'_n\end{aligned}$$

And, since $\varepsilon'_n = (-1)^\alpha \varepsilon_n$ (from their definition), we have

$$\begin{aligned} \zeta \times \zeta_{\text{inv}} &= 1 + \varepsilon_n(\zeta_n^{-1} + (-1)^\alpha \zeta_n) + o(n^{-\alpha+1}) \cdot o(n^{-\alpha+1}) \\ &= 1 + \varepsilon_n(\zeta_n^{-1} + (-1)^\alpha \zeta_n) + o(n^{-\alpha}) \end{aligned}$$

Since $2 \leq \alpha \implies -2\alpha + 2 \leq -\alpha$ at the very least. □

Now we can begin the proof of the theorem.

Proof. 1.1 Now let us simplify the expression of $C_{\text{err}}(j)$.

$$\begin{aligned} C_{\text{err}}(j) &= \frac{1}{n} \sum_{0 \leq k \leq n-1} \left(\sum_{0 \leq i \leq n-1} c_i \zeta^{ki} \right) \zeta_{\text{inv}}^{kj} \\ &= c_j \cdot \left(\frac{1}{n} \sum_{0 \leq k \leq n-1} \zeta^{jk} \cdot \zeta_{\text{inv}}^{jk} \right) + \frac{1}{n} \sum_{i \neq j} c_i \left(\sum_{0 \leq k \leq n-1} \zeta^{ik} \cdot \zeta_{\text{inv}}^{jk} \right) \end{aligned}$$

And, supposedly, the first term is going to be close to c_j, and the second, to 0. Let us write, more simply

$$C_{\text{err}}(j) = c_j \cdot E_j + \frac{1}{n} \sum_{i \neq j} c_i \cdot \delta_j(i)$$

So our goal now is to compute E_j and each $\delta_j(i)$ with $i \neq j$, for any fixed j.
First of all

$$\begin{aligned} n \cdot E_j &= \sum_{0 \leq k \leq n-1} \zeta^{jk} \cdot \zeta_{\text{inv}}^{jk} \\ &= \sum_{0 \leq k \leq n-1} (\zeta \cdot \zeta_{\text{inv}})^{jk} \\ &= \sum_{0 \leq k \leq n-1} \left(1 + o(n^{-\alpha+1})\right)^{jk} \\ &= \sum_{0 \leq k \leq n-1} (1 + jk \cdot o(n^{-\alpha+1})) \\ &= n + j \cdot o(n^{-\alpha+3}) \\ \implies E_j &= 1 + j \cdot o(n^{-\alpha+2}) \end{aligned}$$

Then, for each $\delta_j(i)$ we have

$$\begin{aligned}
\delta_j(i) &= \sum_{0\le k\le n-1} (\zeta)^{ik} \times (\zeta_{\text{inv}})^{jk} \\
&= \sum_{0\le k\le n-1} \left(\zeta_n^{ik} + (ik)\cdot\varepsilon_n\cdot\zeta_n^{ik-1} + o(n^{-\alpha+2})\right) \times \left(\zeta_n^{-jk} + (jk)\cdot\varepsilon_n\cdot\zeta_n^{1-jk} + o(n^{-\alpha+2})\right) \\
&= \sum_{0\le k\le n-1} (\zeta_n^{(i-j)})^k + (jk)\cdot\varepsilon'_n\cdot(\zeta_n^{i-j})^k\zeta_n + (ik)\cdot\varepsilon_n\cdot(\zeta_n^{i-j})^k\zeta_n^{-1} \\
&\quad +(ik)\cdot\varepsilon_n\cdot\zeta_n^{ik-1}\cdot o(n^{-\alpha+2}) + (jk)\cdot\varepsilon'_n\cdot\zeta_n^{jk-1}\cdot o(n^{-\alpha+2}) \\
&\quad +(ijk^2)\cdot(\zeta_n^{i-j})^k\varepsilon_n\varepsilon'_n + o(n^{-\alpha+2}) + o(n^{-\alpha+2}) + +o(n^{-2\alpha+2})
\end{aligned}$$

Using the upper bound $i,j,k\le n$ and the fact that ε_n and ε'_n are $o(n^{-\alpha+1})$ we can see that most of these terms are $o(n^{-\alpha+2})$ to finally obtain

$$\begin{aligned}
\delta_j(i) &= \sum_{0\le k\le n-1} \left((\zeta_n^{(i-j)})^k + (j\varepsilon'_n\zeta_n + i\varepsilon_n\zeta_n^{-1})\cdot k(\zeta_n^{i-j})^k + o(n^{-\alpha+2})\right) \\
&= 0 + \left(\left(j\varepsilon'_n\zeta_n + i\varepsilon_n\zeta_n^{-1}\right)\sum_{0\le k\le n-1} k(\zeta_n^{i-j})^k\right) + o(n^{-\alpha+3})
\end{aligned}$$

We can finally evaluate the quantity $\|C - C_{\text{err}}\|_\infty$, in fact for any given j, the value of $|C_{\text{err}}(j) - c_j|$ is

$$\begin{aligned}
&\left|c_j\cdot j\cdot o(n^{-\alpha+2}) + \frac{1}{n}\sum_{i\ne j} c_i\left((j\varepsilon'_n\zeta_n + i\varepsilon_n\zeta_n^{-1})\times\left(\sum_{0\le k\le n-1} k(\zeta_n^{i-j})^k\right) + o(n^{-\alpha+3})\right)\right| \\
\le\ & |c_j\cdot j\cdot o(n^{-\alpha+2})| + \frac{1}{n}\sum_{i\ne j}|c_i|\times|j\varepsilon'_n\zeta_n + i\varepsilon_n\zeta_n^{-1}|\times\left(\sum_{0\le k\le n-1}|k(\zeta_n^{i-j})^k|\right) + |c_i|\times o(n^{-\alpha+3}) \\
\le\ & \|C\|_\infty\cdot n\cdot o(n^{-\alpha+2}) + \frac{1}{n}\left(\sum_{i\ne j}|c_i|\cdot 2n\cdot n^{-\alpha}\cdot\frac{(2\pi)^\alpha}{\alpha!}\times\left(\sum_{0\le k\le n-1} k\right)\right) + \|C\|_\infty\cdot o(n^{-\alpha+3}) \\
\le\ & \|C\|_\infty\cdot\frac{(2\pi)^\alpha}{\alpha!}\cdot n^{-\alpha+3} + \|C\|_\infty\cdot o(n^{-\alpha+3})
\end{aligned}$$

And now we can clearly see that this quantity does not depend on j so we only need

$$\begin{aligned}
& \|C\|_\infty\cdot((2\pi)^\alpha/\alpha!)\cdot n^{-\alpha+3} && < \frac{1}{2} \\
\iff\ & \log\|C\|_\infty + \alpha\log(2\pi) - \log(\alpha!) - (\alpha-3)\log n && < -\log 2 \\
\iff\ & \log\|C\|_\infty + 3\log n + \log 2 && < \alpha\log n + \log(\alpha!) - \alpha\log(2\pi)
\end{aligned}$$

Finally, by *rounding* we mean evaluating $\lfloor C_{\text{err}}(j) + \frac{1}{2} \rfloor$ so naturally

$$\begin{aligned} C_{\text{err}}(j) - \frac{1}{2} \leq \lfloor C_{\text{err}}(j) + \frac{1}{2} \rfloor &< C_{\text{err}}(j) + \frac{1}{2} \\ C_{\text{err}}(j) - c_j - \frac{1}{2} \leq \lfloor C_{\text{err}}(j) + \frac{1}{2} \rfloor - c_j &< C_{\text{err}}(j) - c_j + \frac{1}{2} \end{aligned}$$

Suppose $-1/2 < C_{\text{err}}(j) - c_j \leq 0$, then we have

$$-1 < \lfloor C_{\text{err}}(j) + \tfrac{1}{2} \rfloor - c_j < \frac{1}{2}$$

And since $\lfloor C_{\text{err}}(j) + \frac{1}{2} \rfloor - c_j$ is an integer, it must be zero, likewise, if $0 \leq C_{\text{err}}(j) - c_j < 1/2$, then we have

$$-\frac{1}{2} \leq \lfloor C_{\text{err}}(j) + \tfrac{1}{2} \rfloor - c_j < 1$$

And by the same argument, $\lfloor C_{\text{err}}(j) + \frac{1}{2} \rfloor - c_j$ has to be zero, hence the result. □

Proposition 1.3. *The Multiplication-via-FFT algorithm taking inputs $A, B \in \mathbf{Z}[X]$ yields an output $C_{err} \in \mathbf{Z}[X]$ with coefficients*

$$C_{err}(j) = \frac{1}{n} \sum_{0 \leq k \leq n-1} \left(\sum_{0 \leq i \leq n-1} c_i \zeta^{ki} \right) \zeta_{inv}^{-kj}$$

.

Remark: The algorithm is composed of 4 steps, which require computing:

1. ζ then ζ_{inv}
2. $A_{\text{vector}} = (A(\zeta), \cdots, A(\zeta^{n-1}))$ and $B_{\text{vector}} = (B(\zeta), \cdots, B(\zeta^{n-1}))$
3. the product of each evaluation at ζ^k of A and B for each coordinate of a new vector $\widehat{C}_{\text{err}}$, which are going to be exactly $(C(\zeta), \cdots, C(\zeta^{n-1}))$
4. the inverse FFT of $\widehat{C}_{\text{err}}$, *ie.* evaluations of this polynomial at each ζ_{inv}^k

Proof. First of all, if coordinate k of the vector A_{vector} is denoted by $A_{\text{vector}}(k)$, then

$$A_{\text{vector}}(k) = A(\zeta^k) = \sum_{0 \leq i \leq n-1} a_i \zeta^{ki}$$

And the same goes for B, keeping in mind that a_i and b_i up to $n-1$ are not necessarily non-zero in this notation. Then, at coordinate k, the product of the two corresponding coordinates of A_{vector} and B_{vector} are going to be

$$\begin{aligned} A_{\text{vector}}(k) \times B_{\text{vector}}(k) &= A(\zeta^k) \times B(\zeta^k) \\ &= C(\zeta^k) \end{aligned}$$

Then the inverse FFT of $\widehat{C}_{\text{err}}$ is going to be the evaluations of the corresponding polynomial at each ζ_{inv}^j normalized by $1/n$, namely

$$C_{\text{err}}(j) = \frac{1}{n} \sum_{0 \leq k \leq n-1} \left(\sum_{0 \leq i \leq n-1} c_i \zeta^{ki} \right) \zeta_{\text{inv}}^{-kj}$$

□

To conclude by stating the obvious, ζ can be an approximation of ζ_n of order α by just writing the TAYLOR series of the exponential to the order α.

Now if you think that the boundary described in 1.1 does not yield an obvious way for a machine to compute α you would be absolutely right.
Finding the smallest integer α satisfying this inequality roughly requires inverting $\alpha \log \alpha$, also we can see that one of the terms on the left-hand side depends on $\|C\|_\infty$, which we are meant to compute.
This however can easily be fixed by noticing that $\|C\|_\infty \leq (n/2) \times \|A\|_\infty \times \|B\|_\infty$ since each coefficient of $\|C\|_\infty$ is the sum of at most $\min(\deg A, \deg B) + 1$ products of a coefficient of A and a coefficient of B and $n > \deg C$.
Now in order to simplify the right-hand side one can notice, using a series-to-integral comparison, that

$$\begin{aligned} \int_1^\alpha \log(t) \mathrm{d}t &\leq \int_1^{\alpha+1} \log(t) \mathrm{d}t \leq \log(\alpha!) \\ = \alpha \log \alpha - \alpha + 1 &\geq \alpha(\log \alpha - 1) \end{aligned}$$

So that we get

$$\alpha \left(\log \alpha + \log(n/2\pi) - 1 \right) \leq \alpha \left(\log n + \frac{1}{\alpha} \log(\alpha!) - \log(2\pi) \right)$$

So if we let $\beta = \log(n/2\pi) - 1$ and $\gamma = \log\left((n/2) \cdot \|A\|_\infty \cdot \|B\|_\infty\right) + 3 \log n + \log 2$ then finding the smallest α is equivalent to computing the ceiling of the root of the function $f(x) = x(\log x + \beta) - \gamma$.

Given $n > 3$, let us show that $f(2\gamma) > 0$, it suffices to show that $\log 2 + \log\gamma + \beta > 1/2$, because then $f(2\gamma) > \gamma - \gamma = 0$, indeed

$$\begin{aligned}
\log 2 + \log\gamma + \beta &= (\log 2 + \log n) + (\log\gamma) - (\log(2\pi) + 1) \\
&\geq 3\log 2 + (\log(\log 2 + 3\log n + \log 2)) - (\log 7 + 1) \text{ , since } 2\pi < 7 \\
&\geq 3\log 2 + (\log(\log 2 + 7\log 2)) - (\log 7 + 1) \\
&\geq 3\log 2 + \log 7 + \log\log 2 - (\log 7 + 1) \\
&\geq (3\log 2 - 1) + \log\log 2 \\
&\geq 1 - (1/2) = 1/2 \text{ , since } 3\log 2 \geq 2 \text{ and } -\log\log 2 \geq -1/2
\end{aligned}$$

And since $f(e^{-\beta}) = -\gamma < 0$ this shows that a root a of f exists, and since $f'(x) = \log x + \beta + 1$, at $x = e^{-\beta}$, f' is positive and stays positive afterwards, so f is increasing on the interval $I = \left[e^{-\beta}, 2\gamma\right]$ and therefore admits a unique root a.
Now if we choose $x_0 = 2\gamma$ and compute $x_N = x_{N-1} - (f(x_{N-1})/f'(x_{N-1}))$ for $N \geq 1$ using NEWTON's method, the convergence criterion states that we need $K|x_0 - a| < 1$ to have a fast quadratic rate of convergence, where $K = \max_I |f'|/2\min_I |f''|$, namely $K|x_N - a| \leq (K|2\gamma - a|)^{2^N}$.
Indeed,

$$\begin{aligned}
& e^{-\beta} < a < 2\gamma \\
\implies 0 \;<\; & 1 - \frac{3}{2(\gamma/a + 1)} < 1 - \frac{3}{2(\gamma e^{\beta} + 1)}
\end{aligned}$$

Moreover, notice that

$$\begin{aligned}
1 - \frac{3}{2(\gamma/a + 1)} &= \frac{2(\gamma/a) - 1}{2(\gamma/a + 1)} && (1) \\
&= \frac{2\gamma - a}{2(\gamma + a)} && (2)
\end{aligned}$$

And since $f(a) = 0$, then $a(\log a + \beta) = \gamma$, so

$$\begin{aligned}
K &= \frac{\max_{x\in I}\{1/x\}}{2\min_{x\in I}\{\log x + \beta + 1\}} \\
&= \frac{1}{2a(\log a + \beta + 1)} \\
&= \frac{1}{2(\gamma + a)}
\end{aligned}$$

Which means that $K|x_0 - a| = K|2\gamma - a| = |2\gamma - a|/2(\gamma + a)$ which is the same term as (2), before, that means

$$\begin{aligned} K|x_N - a| \leq (K|x_0 - a|)^{2^N} &\leq \left(1 - \frac{3}{2(\gamma e^\beta + 1)}\right)^{2^N} \\ \implies \log(K|x_N - a|) &\leq 2^N \log\left(1 - \frac{3}{2(\gamma e^\beta + 1)}\right) \\ \implies \log K + \log|x_N - a| &\leq \frac{-2^N \times 3}{2(\gamma e^\beta + 1)} \end{aligned}$$

What matters now is finding N so that the difference between the approximation x_N and a is less than an 1 so that we can be wrong by at most one order of α, this is equivalent to their logarithm being less than 0, which happens very quickly and results from the following

$$\begin{aligned} \log 2(\gamma + a) = \log(1/K) &\leq \frac{2^{N-1} \times 3}{\gamma e^\beta + 1} \\ \iff (\gamma e^\beta + 1) \times \log 2(\gamma + a) &\leq 2^{N-1} \times 3 \\ \iff \log(\gamma e^\beta + 1) + \log\log 2(\gamma + a) - \log 3 &\leq (N-1)\log 2 \end{aligned}$$

The biggest terms on the left-hand side are roughly $\log\gamma$ — which mostly depends on $\log\log(n \cdot \|A\|_\infty \cdot \|B\|_\infty)$ which is supposedly very small; if $n \cdot \|A\|_\infty \cdot \|B\|_\infty = e^{e^3}$ which is approximately a billion, this double logarithm takes the value 3 — and $\log(e^\beta) = \beta = O(\log n)$, obviously this means that the method does not take very long to converge to a reasonable answer for α, and iterating the method $O(\log n)$ times still falls way below the expected time complexity of the FFT.

Finally, we have learned that, when writing

$$\begin{aligned} \zeta &= \sum_{0 \leq k \leq \alpha - 1} \frac{(2\pi i/n)^k}{k!} \text{ , and} \\ \zeta_{\text{inv}} &= \sum_{0 \leq k \leq \alpha - 1} \frac{(-2\pi i/n)^k}{k!} \end{aligned}$$

in the FFT algorithm, it is enough to require that α verifies

$$\log\left(n \cdot \|A\|_\infty \cdot \|B\|_\infty\right) + 3\log n + \log 2 < \alpha\left(\log\alpha + \log(n/2\pi) - 1\right)$$

which is easy to approximate using Newton's method, in order to retrieve the exact integer coefficients of the product $C = A \times B$.

2 Finding an appropriate base for $\mathbf{Q}(\zeta_n)$

describe the process in [1] then compare this basis to the "natural" basis and algorithmically too (doing polynomial division $n-d$ times compared to calculating every moduli of powers of ζ_n not in the basis and then reconstructing the linear expression in the basis from that)

3 Abstract arithmetic using appropriate bases

A machine does not understand the concept of doing arithmetic with both numbers and unknown variables whereas a human is able to make heuristic choices on whether or not to separate some variables from others, separate them from written-out numbers like integers for example, and such.
A criterion by Cohn states the following:

Theorem 3.1. (Cohn's irreducibility criterion) *Let $b \in \mathbf{N}$ and $b \geq 2$, $P \in \mathbf{Z}[x]$ of degree n with $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ such that: $\forall k \in [\![0, n]\!], 0 \leq a_k \leq b-1$. If $P(b)$ is a prime number, then P is irreducible in $\mathbf{Z}[x]$.*

Basically, if a prime number is ever written in base b then the corresponding polynomial must be irreducible in $\mathbf{Z}[x]$. In the same vein, not concerning irreducibility but rather identifying polynomials, if some polynomial P is unknown with height $H(P) := \max_k |a_k| \leq b$ for some integer b then P is entirely characterized by the value of $P(2b+1)$, that is:

Theorem 3.2. *Let $P \in \mathbf{Z}[x]$ be a polynomial of degree n with $P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$ and b an integer such that $b \geq H(P) = \max_k |a_k|$.*
Define $Q(x) = \sum_{0 \leq k \leq n} b x^k = b(x^{n+1}-1)/(x-1)$ and $q = P(2b+1) + Q(2b+1)$.
If q is written in base $2b+1$ in the form of $q = \overline{d_n d_{n-1} ... d_1 d_0}_{(2b+1)}$ (where d_k is the k-th digit of q) then the coefficients a_k of P verify the relation:

$$\forall k \in [\![0, n]\!], a_k = d_k - b$$

Likewise, the degree n of P verifies $n = \lfloor \log_{2b+1} |P(2b+1)| + \log_{2b+1}(2) \rfloor$.

Proof. We see from the definition of Q that:

$$q = P(2b+1) + Q(2b+1) = (a_k + b)(2b+1)^n + \cdots + (a_1 + b)(2b+1) + (a_0 + b)$$

Since $0 \leq |a_k| \leq b$ then $-b \leq a_k \leq b$, and then $0 \leq a_k + b \leq 2b$ so the $a_k + b$ can be seen as digits of a number in base $2b+1$, digits of q.

Let us prove the result related to the degree n of P. Let $N = \lfloor \log_{2b+1} |P(2b+1)| + \log_{2b+1}(2) \rfloor$, we have $N = \lfloor n + \log_{2b+1} |a_n + a_{n-1}/(2b+1) + \cdots + a_0/(2b+1)^n| + \log_{2b+1}(2) \rfloor = \lfloor n + \log_{2b+1} \left| \sum_{\mathbf{0} \leq k \leq n} a_{n-k}/(2b+1)^k \right| + \log_{2b+1}(2) \rfloor$. Now, since $|a_n| \geq 1$, we must verify:

$$
\begin{aligned}
|1 - \left| \sum_{\mathbf{1} \leq k \leq n} |a_{n-k}|/(2b+1)^k \right| | \quad &\leq \quad \left| \sum_{0 \leq k \leq n} a_{n-k}/(2b+1)^k \right| \leq \sum_{\mathbf{0} \leq k \leq n} |a_{n-k}|/(2b+1)^k \\
|1 - \left| b \left(\frac{2b+1}{2b} - 1 \right) \right| | \quad &< \quad \left| \sum_{0 \leq k \leq n} a_{n-k}/(2b+1)^k \right| < b \frac{2b+1}{2b} \\
2^{-1} = \left| 1 - \frac{1}{2} \right| \quad &< \quad \left| \sum_{0 \leq k \leq n} a_{n-k}/(2b+1)^k \right| < b + \frac{1}{2} \\
-\log_{2b+1}(2) \quad < \quad \log_{2b+1} &\left| \sum_{0 \leq k \leq n} a_{n-k}/(2b+1)^k \right| < \log_{2b+1} \left(b + \frac{1}{2} \right) \\
0 \quad < \quad \log_{2b+1} &\left| \sum_{0 \leq k \leq n} a_{n-k}/(2b+1)^k \right| + \log_{2b+1}(2) < \log_{2b+1}(2b+1) = 1
\end{aligned}
$$

This proves that $N = \lfloor n + \varepsilon \rfloor$ where $0 < \varepsilon < 1$, since n is an integer, $N = n$. □

Remark: Seeing this theorem as an algorithm taking inputs b and $N = P(2b+1)$ two binary positive integers, we can represent each step as follows:

1. $n = \lfloor \log_{2b+1} |N| + \log_{2b+1}(2) \rfloor$
2. $r = Q(2b+1) = \frac{1}{2}((2b+1)^{n+1} - 1)$
3. $q = r + N$
4. Convert q from base 2 to base $2b+1$ to get each digit d_k from $q = \overline{d_n d_{n-1} ... d_1 d_0}_{(2b+1)}$
5. $\forall k, a_k = d_k - b$

Steps $2, 3$ and 5 have obvious complexities of $O(M(\log b) \log |N| / \log b)$, $O(1)$ and $O(\log |N|)$ where $M(x)$ is the complexity of our multiplication algorithm for an x-digit number.
For step 1, one way to compute $\log_{2b+1} |N|$ and $\log_{2b+1}(2)$ is to set $a = \lfloor \log_2 |N| \rfloor$ and $b = \lfloor \log_2(2b+1) \rfloor$ and hope that $n' = \lfloor \frac{a}{b} + \frac{1}{b} \rfloor$ is accurate.
a and b are just going to be the number of digits of N and $2b+1$, which are usually immediately known, if not for directly counting the number of digits in the binary representation of both of them, which have complexity $O(\log |N|)$ and $O(\log b)$.

If $x = \log_2 |N|$ and $y = \log_2(2b+1)$ then we have:

$$\begin{aligned}
& \forall t \in \mathbf{R}, t-1 < \lfloor t \rfloor \leq t, \text{ hence,} \\
\frac{x-1}{y} &\leq \frac{a}{b} \leq \frac{x}{y-1} \\
\frac{x}{y} &\leq \frac{a}{b} + \frac{1}{b} \leq \frac{x+1}{y-1} \\
\frac{x}{y} - \left(\frac{x}{y} + \frac{1}{y}\right) &\leq \left(\frac{a}{b} + \frac{1}{b}\right) - \left(\frac{x}{y} + \frac{1}{y}\right) \leq \frac{x+1}{y-1} - \left(\frac{x}{y} + \frac{1}{y}\right) \\
-\frac{1}{y} &\leq \left(\frac{a}{b} + \frac{1}{b}\right) - \left(\frac{x}{y} + \frac{1}{y}\right) \leq \frac{x+1}{y(y-1)}
\end{aligned}$$

Simplifying $\left(\frac{a}{b} + \frac{1}{b}\right) - \left(\frac{x}{y} + \frac{1}{y}\right)$ as $A - B$ for clarity purposes we deduce that, since:

$$(A-B) - 1 \leq \lfloor A \rfloor - \lfloor B \rfloor \leq (A-B) + 1$$

Then, we have:

$$-\frac{1}{y} - 1 \leq n' - n \leq \frac{x+1}{y(y-1)} + 1$$

Notice that only the right hand side can get arbitrarily large but the condition on x (and therefore, on N), is very weak; for example, if we want this right hand side to not exceed 2, we only need $|N| \leq (2b+1)^{d-1}$ where d is the number of digits of $2b+1$ in base 2.
Applying this to concrete numbers we can visualize the condition better, say P has coefficients between -100 and 100, then as long as $|P(201)|$ is not larger than 201^7 (a sixteen-digit number), choosing n' as the value for the degree will be accurate within ± 1 (since it cannot get to $+2$ given that the condition is met), meaning that if the first 2 choices during the run of the algorithm of n' are wrong, the 3rd one will be correct.

The final step to evaluate the complexity of is the 4th, a very simple base conversion algorithm can be described as:

1. Set $d_0 = q \mod (2b+1)$, $r_0 = \lfloor q/(2b+1) \rfloor$
2. For each digit d_k afterwards, do $d_k = r_k \mod (2b+1)$, $r_{k+1} = \lfloor r_k/(2b+1) \rfloor$

This takes n steps, considering that division and modular remainder can be approximated to $O(M(\log q)) = O(M(\log |N|))$ complexity (since N has the same

number of digits as q), we do these n times, so in total we get a complexity of $O(M(\log|N|)\log|N|)$.
From this we see that the most demanding steps are 2 and 4, and in total we get a complexity of:

$$O\left(\frac{M(\log b)\log|N|}{\log b} + M(\log|N|)\log|N|\right)$$

We conclude by saying that, given that the conditions of the theorem are met (we know b and N and the height $H(P)$ of P is not greater than b), the complexity of this algorithm which retrieves the coefficients of P will often look like some power of $\log|N|$.

Example 3.3. *Let us use theorem 3.2 in order to compute the* 84*th cyclotomic polynomial:*
We know that the height of cyclotomic polynomials of order less than 104 *are bounded by* 1*, so using the assumptions of theorem 3.2, we can let* $b = 1$ *and* $2b + 1 = 3$*, and try to compute*

$$\Phi_{104}(3) \quad = \quad \prod_{k\wedge 104=1} (3 - \zeta_{104}^k)$$

Expanding the product expression of $\Phi_{84}(3)$ *and since* $\varphi(84) = 24$ *(where* φ *is* Euler*'s totient function), we know that each term other than* 3^{24} *is going to be a sum of at most* 24 *products of at most* 24 *powers of* ζ_{84}*. Let* $n = 84$*, if* ζ_{84} *is accurate to the order* $n^{-\alpha}$ *then* ζ_{84}^{83}*, in the worst case, will be accurate to the order* $n^{-\alpha+1}$*.*
Each product will then be accurate to $n^{-\alpha+2}$ *since* $24 \leq n$*, and the argument is the same for the sum of* 24 *of these products, making each coefficient in front of a power of* 3 *accurate to* $n^{-\alpha+3}$*, this power of* 3 *can be at most* 3^{23} *and since* $23\times\log(3)/\log(84) \leq 6$ *then each of the terms in the expanded product of* $\Phi_{84}(3)$ *will be accurate to* $n^{-\alpha+9}$*, summing* 24 *of these terms will give an integer accurate to* $n^{-\alpha+10}$*.*
Using this information we just have to compute ζ_{84} *as the* Taylor *series of the exponential to the order* 11 *and compute the product directly.*
We obtain $\Phi_{84}(3) \approx 313380653041.0002 - 0.000905i$ *approximately, hence* $\Phi_{84}(3) = 313380653041$ *to which we add the quantity* $Q(3) = \frac{1}{2}(3^{25} - 1)$ *to get*

$$q = \overline{21211101011121110101112 12}_3$$

(which is surprisingly a palindrome) which, using the algorithm, yields the coefficients

$$\Phi_{84}(X) \quad = \quad X^{24} + X^{22} - X^{18} - X^{16} + X^{12} - X^8 - X^6 + X^2 + 1$$

We can see the coefficients from the digits of q *without thinking too much; each* 2 *is going to be a coefficient of* 1*, each* 0 *is going to be a coefficient of* -1 *and each* 1 *is not going to appear in the expression of* Φ_{84}*, since the resulting coefficient will be zero (we can check these are correct using Wolfram Alpha for example).*

References

[1] Wieb Bosma *Canonical Bases for Cyclotomic Fields* AAEEC, Springer-Verlag, 1990

[2] Andrew Arnold, Michael Monagan *A high-performance algorithm for calculating cyclotomic polynomials* International Workshop on Parallel and Symbolic Computation (PASCO), 2010

[3] W. M. Gentleman, G. Sande *Fast Fourier Transforms - For Fun and Profit* AFIPS, 1966

[4] François Schwarzentruber *Diviser pour régner* 2020