Integer Multiplication and FFT

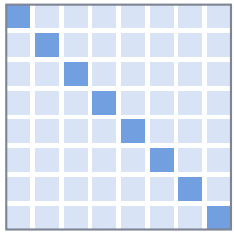
From divide-and-conquer to Fourier analysis

Lecture 2

How expensive is one very large product?

Suppose A and B each have $n \approx 10^4$ decimal digits.

- Grade-school multiplication forms every digit pair.
- That is $n^2 \approx 10^8$ single-digit products.
- Can structure replace most of this work?



$n \times n$ digit pairs

Guiding question

How close can multiplication get to the $\Omega(n)$ time needed merely to read the input?

Today: exploit structure twice

1. **Karatsuba:** split numbers and replace four half-size products by three.
2. **Polynomial encoding:** view digit multiplication as coefficient convolution.
3. **FFT:** change from coefficients to values at roots of unity.
4. **Fourier viewpoint:** understand why convolution becomes pointwise multiplication.

Learning outcome

By the end, you should be able to derive the FFT recurrence, prove the inverse formula, and justify the multiplication algorithm and its running time.

Part 1

Karatsuba multiplication

Grade-school multiplication is quadratic

Write the base-β expansions

$$A = \sum_{i=0}^{n-1} a_i \beta^i, \qquad B = \sum_{j=0}^{n-1} b_j \beta^j.$$

Then

$$AB = \sum_{k=0}^{2n-2} \left(\sum_{i+j=k} a_i b_j \right) \beta^k.$$

- There are n^2 products $a_i b_j$.
- Adding partial products and propagating carries also costs $O(n^2)$ digit operations.

Baseline

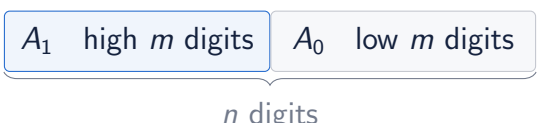
$T_{\text{school}}(n) = \Theta(n^2)$ for fixed base β.

Split each input into low and high halves

Assume first that n is even; leading zeros can pad to a power of two. Let $m = n/2$:

$$A = A_0 + \beta^m A_1, \qquad B = B_0 + \beta^m B_1,$$

where A_0, A_1, B_0, B_1 have at most m base-β digits.



$$AB = A_0 B_0 + \beta^m (A_0 B_1 + A_1 B_0) + \beta^{2m} A_1 B_1.$$

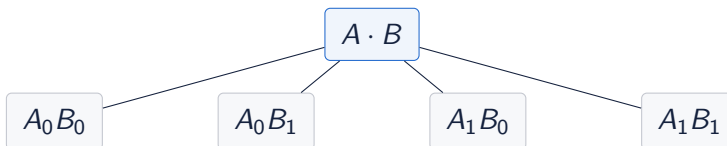
A direct split still uses four products

The expansion asks for

$$A_0B_0, \quad A_0B_1, \quad A_1B_0, \quad A_1B_1.$$

Thus

$$T(n) = 4T(n/2) + O(n) = \Theta(n^2).$$



Splitting alone does not help; we must reduce the number of recursive products.

Karatsuba recovers the middle term from one product

Compute only three recursive products:

$$
\begin{aligned}
P_0 &= A_0 B_0, \\
P_2 &= A_1 B_1, \\
P_1 &= (A_0 + A_1)(B_0 + B_1) - P_0 - P_2.
\end{aligned}
$$

Since

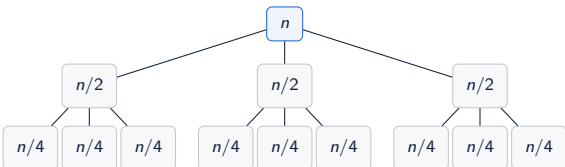
$$(A_0 + A_1)(B_0 + B_1) = P_0 + A_0 B_1 + A_1 B_0 + P_2,$$

we have $P_1 = A_0 B_1 + A_1 B_0$. Therefore

Recombination

$$AB = P_0 + \beta^m P_1 + \beta^{2m} P_2.$$

The recursion tree has $3^{\log_2 n}$ leaves



At level ℓ:

$$3^\ell \text{ nodes}, \qquad \text{size } n/2^\ell.$$

Nonrecursive work:

$$O\left(3^\ell \frac{n}{2^\ell}\right) = O\left(n(3/2)^\ell\right).$$

It grows geometrically, so the bottom levels dominate.

Solving the recurrence gives $O(n^{1.585})$

Karatsuba satisfies

$$T(n) = 3T(n/2) + O(n).$$

The depth is $L = \log_2 n$. Summing the work over internal levels,

$$\sum_{\ell=0}^{L-1} O\big(n(3/2)^{\ell}\big) = O\big(n(3/2)^{L}\big)$$
$$= O\big(n\, n^{\log_2(3/2)}\big) = O\big(n^{\log_2 3}\big).$$

The $3^L = n^{\log_2 3}$ leaves have the same order, hence

$$\boxed{T(n) = \Theta\big(n^{\log_2 3}\big) \approx \Theta(n^{1.585})}.$$

What counts as one operation?

We will use a word-RAM-style model:

- the inputs contain n base-β digits, with fixed β;
- arithmetic on $O(\log n)$-bit words costs $O(1)$;
- longer values are stored as arrays of such words.

Why $O(\log n)$ bits appear later

A convolution coefficient sums at most n bounded digit products, so its magnitude is $O(n)$ and its binary length is $O(\log n)$.

Model caveat

A bit-level Turing-machine analysis must also charge for word arithmetic and numerical precision. We will state explicitly when the model matters.

Part 2

The polynomial bridge

Polynomial coefficients multiply by convolution

Let

$$A(x) = \sum_{j=0}^{n-1} a_j x^j, \qquad B(x) = \sum_{j=0}^{n-1} b_j x^j.$$

Their product $C(x) = A(x)B(x)$ has degree at most $2n - 2$ and coefficients

$$c_k = \sum_{i+j=k} a_i b_j.$$

An integer is a digit polynomial evaluated at the base

For base β digits $a_j, b_j \in \{0, \ldots, \beta - 1\}$, define

$$A(x) = \sum_{j=0}^{n-1} a_j x^j, \qquad B(x) = \sum_{j=0}^{n-1} b_j x^j.$$

The encoded integers are

$$\widetilde{A} = A(\beta), \qquad \widetilde{B} = B(\beta).$$

Therefore

$$\widetilde{A}\widetilde{B} = (A \cdot B)(\beta) = \sum_{k=0}^{2n-2} c_k \beta^k.$$

The c_k need not be valid digits, but

$$0 \le c_k \le n(\beta - 1)^2,$$

so every coefficient uses only $O(\log n)$ bits for fixed β.

Carries convert raw coefficients back to digits

Example: $99 \cdot 99$. The coefficient vectors are listed low degree first:

$$(9, 9) * (9, 9) = (81, 162, 81).$$

Propagate carries in base 10 from left to right:

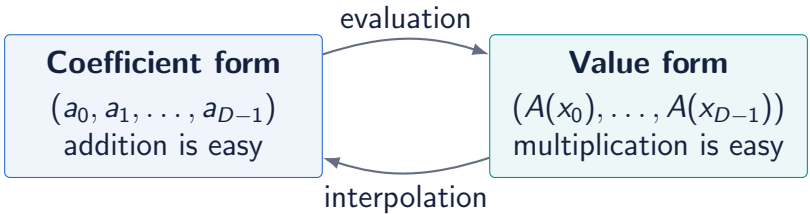
position	value + carry	output digit	next carry
0	81	1	8
1	$162 + 8 = 170$	0	17
2	$81 + 17 = 98$	8	9
3	9	9	0

The normalized digits are $(1, 0, 8, 9)$, hence 9801.

Cost

One pass over $O(n)$ coefficients: $O(n)$ word operations.

A polynomial has two useful representations



If D is larger than the product degree, then D values uniquely determine the product polynomial.

The strategy

Find evaluation points for which both arrows can be computed in $O(D \log D)$ time.

Interpolation lemma: values determine a polynomial

Lemma

Let $x_0, \ldots, x_d$ be **distinct** scalars and let $y_0, \ldots, y_d$ be arbitrary. There is a unique polynomial p of degree at most d such that

$$p(x_i) = y_i \qquad (0 \le i \le d).$$

Two parts need proof:

1. **Existence:** construct a polynomial with the required values.
2. **Uniqueness:** show that two such polynomials must agree everywhere.

Existence: build one selector polynomial per point

Define the Lagrange basis polynomials

$$L_i(x) = \prod_{\substack{0 \le j \le d \\ j \neq i}} \frac{x - x_j}{x_i - x_j}.$$

The denominators are nonzero because the x_i are distinct. Moreover,

$$L_i(x_j) = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases}$$

Therefore

$$p(x) = \sum_{i=0}^{d} y_i L_i(x)$$

has degree at most d and satisfies $p(x_j) = y_j$ for every j.

Uniqueness: too many roots force the zero polynomial

Suppose p and q both have degree at most d and agree at all $d+1$ points. Let

$$r(x) = p(x) - q(x).$$

Then $\deg r \leq d$, while

$$r(x_0) = r(x_1) = \cdots = r(x_d) = 0.$$

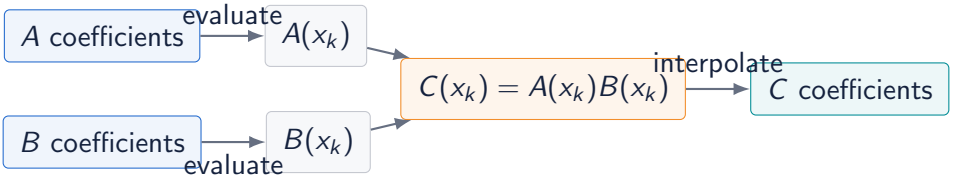
A nonzero degree-d polynomial has at most d distinct roots. Hence r must be the zero polynomial, so $p = q$.

Consequence

For any D distinct points, coefficient form and value form are equivalent for polynomials of degree $< D$.

Multiplication in value form is pointwise

Choose $D \geq 2n - 1$ distinct points $x_0, \ldots, x_{D-1}$.



Pointwise multiplication uses only $D = O(n)$ scalar products.

Bottleneck

At arbitrary points, evaluation and interpolation each cost $\Theta(n^2)$ by straightforward methods.

Part 3

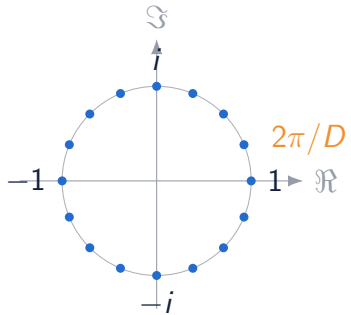
Fast Fourier transform

Choose the Dth roots of unity

Let D be a power of two and

$$\zeta_D = e^{2\pi i/D}.$$

The points $1, \zeta_D, \ldots, \zeta_D^{D-1}$ are distinct and satisfy $\zeta_D^D = 1$.



We evaluate at ζ_D^{-k}; this is the same set of roots, only in reverse order.

The DFT is simultaneous evaluation at those roots

Zero-pad the coefficient vector to length D. Define the (unnormalized) discrete Fourier transform

Forward DFT

$$\mathcal{F}_D(a)_k = \sum_{j=0}^{D-1} a_j \zeta_D^{-kj}, \qquad 0 \le k < D.$$

If $A(x) = \sum_{j=0}^{D-1} a_j x^j$, then

$$\mathcal{F}_D(a)_k = A(\zeta_D^{-k}).$$

Naively, each of D outputs is a D-term sum: $\Theta(D^2)$ work. The FFT computes exactly the same transform in $\Theta(D \log D)$.

Split the DFT sum into even and odd coefficients

Assume D is even. For each output index k,

$$\begin{aligned}\mathcal{F}_D(a)_k &= \sum_{m=0}^{D/2-1} a_{2m}\zeta_D^{-k(2m)} + \sum_{m=0}^{D/2-1} a_{2m+1}\zeta_D^{-k(2m+1)} \\ &= \sum_{m=0}^{D/2-1} a_{2m}(\zeta_D^2)^{-km} + \zeta_D^{-k}\sum_{m=0}^{D/2-1} a_{2m+1}(\zeta_D^2)^{-km}.\end{aligned}$$

Let

$$E = \mathcal{F}_{D/2}(a_0, a_2, \ldots, a_{D-2}), \qquad O = \mathcal{F}_{D/2}(a_1, a_3, \ldots, a_{D-1}),$$

using the primitive $(D/2)$th root ζ_D^2.

Two half-size transforms produce all D outputs

Because the half-size transforms are periodic in their index, for $0 \le k < D/2$,

$$E_{k+D/2} = E_k, \qquad O_{k+D/2} = O_k.$$

Also $\zeta_D^{D/2} = -1$. Therefore

$$\mathcal{F}_D(a)_k = E_k + \zeta_D^{-k} O_k,$$
$$\mathcal{F}_D(a)_{k+D/2} = E_k - \zeta_D^{-k} O_k.$$

Key reuse

One pair (E_k, O_k) gives two outputs: a sum and a difference.

Recursive FFT algorithm

FFT$(a_0, \ldots, a_{D-1})$

1. If $D = 1$, return (a_0).
2. Recursively compute $E = \mathbf{FFT}(a_0, a_2, \ldots)$.
3. Recursively compute $O = \mathbf{FFT}(a_1, a_3, \ldots)$.
4. For $0 \leq k < D/2$, set $t = \zeta_D^{-k} O_k$ and output

$$F_k = E_k + t, \qquad F_{k+D/2} = E_k - t.$$

The FFT recurrence solves to $\Theta(D \log D)$

Two transforms of half the length plus a linear combine step give

$$T(D) = 2T(D/2) + \Theta(D).$$

At every recursion level, the total amount of data processed is D:

Therefore

$$\boxed{T(D) = \Theta(D \log D)}.$$

Checkpoint: why must D be a power of two here?

Radix-2 FFT repeatedly halves the transform length until it reaches 1.

For this algorithm

Choose

$$D = 2^{\lceil \log_2(2n-1) \rceil}.$$

Then $2n - 1 \leq D < 4n - 2$, so $D = \Theta(n)$.

Powers of two are not mathematically required for Fourier transforms; they are required only by this particular radix-2 recursion. Other FFT factorizations handle other composite lengths.

Part 4

Inverse transform and multiplication

Root-of-unity sums behave like Kronecker deltas

Orthogonality lemma

For every integer m,

$$\sum_{j=0}^{D-1} \zeta_D^{jm} = \begin{cases} D, & m \equiv 0 \pmod{D}, \\ 0, & m \not\equiv 0 \pmod{D}. \end{cases}$$

This identity is the algebraic engine behind

- inverse DFT;
- orthogonality of characters;
- cancellation of nonmatching frequencies.

Proof of the orthogonality lemma

If $m \equiv 0 \pmod{D}$, every summand is 1, so the sum is D.
Otherwise let $\alpha = \zeta_D^m$. Then $\alpha \neq 1$ but

$$\alpha^D = (\zeta_D^D)^m = 1.$$

The finite geometric-series identity gives

$$\sum_{j=0}^{D-1} \alpha^j = \frac{1-\alpha^D}{1-\alpha} = \frac{1-1}{1-\alpha} = 0.$$

Inverse DFT: reverse the sign and divide by D

If

$$F_k = \sum_{r=0}^{D-1} a_r \zeta_D^{-kr},$$

then

Inverse DFT

$$a_j = \frac{1}{D} \sum_{k=0}^{D-1} F_k \zeta_D^{-kj}.$$

The factor $1/D$ normalizes the surviving term in the root-of-unity sum; the sign reversal undoes the forward phases.

Proof that the inverse recovers every coefficient

Proof.

$$\frac{1}{D}\sum_{k=0}^{D-1} F_k \zeta_D^{kj} = \frac{1}{D}\sum_{k=0}^{D-1}\sum_{r=0}^{D-1} a_r \zeta_D^{-kr}\zeta_D^{kj}$$
$$= \sum_{r=0}^{D-1} a_r \left(\frac{1}{D}\sum_{k=0}^{D-1} \zeta_D^{k(j-r)}\right) = a_j.$$

□

The parenthesized sum equals 1 when $r = j$ and 0 otherwise, by the orthogonality lemma.

The same FFT machinery computes the inverse

Compare the formulas:

$$F_k = \sum_j a_j \zeta_D^{-kj},$$
$$a_j = \frac{1}{D} \sum_k F_k \zeta_D^{kj}.$$

For complex inputs, an equivalent identity is

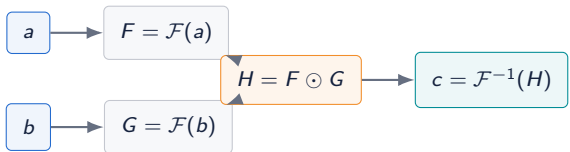
$$\mathcal{F}_D^{-1}(F) = \frac{1}{D} \overline{\mathcal{F}_D(\overline{F})}.$$

Therefore interpolation at roots of unity also costs $\Theta(D \log D)$.

FFT polynomial multiplication: the complete algorithm

Given A, B of degree $< n$:

1. Choose a power of two $D \geq 2n - 1$.
2. Zero-pad both coefficient vectors to length D.
3. Compute $F = \mathcal{F}_D(a)$ and $G = \mathcal{F}_D(b)$.
4. Multiply pointwise: $H_k = F_k G_k$ for $0 \leq k < D$.
5. Compute $c = \mathcal{F}_D^{-1}(H)$.



Zero-padding prevents circular wrap-around

A length-D inverse DFT recovers *cyclic* convolution, where indices are taken modulo D:

$$c_k^{\text{cyc}} = \sum_{i+j \equiv k \pmod{D}} a_i b_j.$$

But $a_i = b_j = 0$ for indices $\geq n$, so every nonzero pair satisfies

$$0 \leq i + j \leq 2n - 2 < D.$$

No term crosses the boundary and wraps back to a smaller index. Hence

$$c_k^{\text{cyc}} = \sum_{i+j=k} a_i b_j,$$

exactly the ordinary polynomial-product coefficient.

The total arithmetic cost is $\Theta(n \log n)$

With $D = \Theta(n)$:

operation	cost
forward FFT of A	$\Theta(D \log D)$
forward FFT of B	$\Theta(D \log D)$
D pointwise products	$\Theta(D)$
inverse FFT	$\Theta(D \log D)$
carry propagation	$\Theta(D)$
total	$\boxed{\Theta(n \log n)}$

This is an arithmetic-operation count; exact integer multiplication also requires a representation that controls rounding or uses exact modular arithmetic.

How can a complex-valued FFT return exact integers?

The true coefficients c_k are integers. A numerical FFT computes approximations $\widetilde{c}_k$.

Exact recovery criterion

If $|\widetilde{c}_k - c_k| < \frac{1}{2}$ for every k, rounding to the nearest integer recovers all coefficients exactly.

In the stated word-RAM model:

- intermediate magnitudes and coefficient bounds are polynomial in n;
- $O(\log n)$ bits of controlled precision, with guard bits, suffice for the required error bound in a careful implementation.

Practical libraries split inputs into smaller blocks, manage error bounds, and often verify results. Number-theoretic transforms avoid floating-point error by working modulo suitable primes.

Karatsuba and FFT remove different bottlenecks

method	central idea	running time
grade school	form every digit pair	$\Theta(n^2)$
Karatsuba	three products per split	$\Theta(n^{\log_2 3})$
FFT	values make convolution pointwise	$\Theta(n \log n)$[1]

Constants matter: grade school wins for small inputs, Karatsuba for medium sizes, and FFT-based methods for very large inputs.

[1] Arithmetic/word-operation model described earlier; bit-level precision costs require separate accounting.

Part 5

Fourier analysis on a cyclic group

Functions on $\mathbb{Z}_D$ form a D-dimensional vector space

Let $G = \mathbb{Z}_D$. A function $f : G \to \mathbb{C}$ is a vector

$$(f(0), f(1), \ldots, f(D-1)) \in \mathbb{C}^D.$$

Equip this space with the normalized inner product

$$\langle f, g \rangle = \frac{1}{D} \sum_{x \in \mathbb{Z}_D} f(x)\overline{g(x)}.$$

The standard basis isolates positions. Fourier analysis uses a different basis that isolates frequencies.

Change of viewpoint

The FFT is not only fast evaluation: it is a fast change of basis in a structured vector space.

Characters are the Fourier basis

For $k \in \mathbb{Z}_D$, define the character

$$\chi_k(x) = \zeta_D^{kx}.$$

Characters turn addition into multiplication:

$$\chi_k(x+y) = \chi_k(x)\chi_k(y).$$

Their inner products are

$$\begin{aligned}\langle \chi_k, \chi_{k'} \rangle &= \frac{1}{D} \sum_{x=0}^{D-1} \zeta_D^{kx} \overline{\zeta_D^{k'x}} \\ &= \frac{1}{D} \sum_{x=0}^{D-1} \zeta_D^{(k-k')x} = \begin{cases} 1, & k = k', \\ 0, & k \neq k'. \end{cases}\end{aligned}$$

There are D orthonormal characters in a D-dimensional space, so they form a basis.

Fourier coefficients are coordinates in that basis

Define the normalized Fourier coefficient

$$\widehat{f}(k) = \langle f, \chi_k \rangle = \frac{1}{D} \sum_{x=0}^{D-1} f(x) \zeta_D^{-kx}.$$

Since the characters form an orthonormal basis,

Fourier expansion

$$f(x) = \sum_{k=0}^{D-1} \widehat{f}(k) \chi_k(x) = \sum_{k=0}^{D-1} \widehat{f}(k) \zeta_D^{kx}.$$

This is exactly the inverse-DFT formula, expressed as basis expansion.

Two normalizations differ by a factor of D

object	definition	inversion
engineering DFT	$\mathcal{F}f(k) = \sum_x f(x)\zeta_D^{-kx}$	$f(x) = \frac{1}{D}\sum_k \mathcal{F}f(k)\zeta_D^{kx}$
basis coefficient	$\widehat{f}(k) = \frac{1}{D}\sum_x f(x)\zeta_D^{-kx}$	$f(x) = \sum_k \widehat{f}(k)\zeta_D^{kx}$

Thus

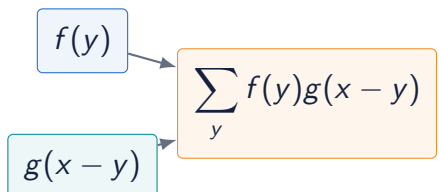
$$\mathcal{F}f(k) = D\,\widehat{f}(k).$$

Both conventions are correct. Mixing them silently creates missing factors of D.

Convolution combines all translated overlaps

For $f, g : \mathbb{Z}_D \to \mathbb{C}$, define cyclic convolution

$$(f * g)(x) = \sum_{y \in \mathbb{Z}_D} f(y) g(x - y).$$



one value for each shift x

The coefficient formula for polynomial multiplication is the non-cyclic version of the same operation.

Fourier turns convolution into multiplication

Using the unnormalized DFT $\mathcal{F}$,

Convolution theorem

$$\mathcal{F}(f * g)(k) = \mathcal{F}f(k)\,\mathcal{F}g(k).$$

Translation in the input becomes a phase in frequency; summing all translated overlaps leaves an ordinary product at each frequency.
With normalized basis coefficients, the same statement is

$$\widehat{f * g}(k) = D\widehat{f}(k)\widehat{g}(k).$$

Proof of the convolution theorem

Proof.

$$\begin{aligned}\mathcal{F}(f * g)(k) &= \sum_x \sum_y f(y) g(x-y) \zeta_D^{-kx} \\ &\overset{z=x-y}{=} \sum_y \sum_z f(y) g(z) \zeta_D^{-k(z+y)} \\ &= \left(\sum_y f(y) \zeta_D^{-ky} \right) \left(\sum_z g(z) \zeta_D^{-kz} \right).\end{aligned}$$

□

The substitution $z = x - y$ is valid because subtraction and addition are modulo D, so $(x, y) \leftrightarrow (z, y)$ is a bijection on $\mathbb{Z}_D^2$.

Zero-padding connects the theorem to polynomials

Let a, b be length-D vectors obtained by zero-padding polynomial coefficients. Then

$$\mathcal{F}^{-1}\big(\mathcal{F}(a) \odot \mathcal{F}(b)\big) = a * b.$$

If $D \geq 2n - 1$, cyclic and ordinary convolution agree on these padded vectors. Therefore the FFT multiplication algorithm is precisely

$$\boxed{\text{convolution} = \mathcal{F}^{-1}\big(\text{pointwise product of transforms}\big)}.$$

The "coefficient-to-value" story and the "convolution theorem" story are the same algebra viewed from two angles.

Independent sums are convolutions of distributions

Let X, Y be independent random variables taking values in $\mathbb{Z}_D$, with

$$p_X(x) = \Pr[X = x], \qquad p_Y(x) = \Pr[Y = x].$$

Then

$$\begin{aligned}\Pr[X + Y = x] &= \sum_y \Pr[X = y] \Pr[Y = x - y] \\ &= (p_X * p_Y)(x).\end{aligned}$$

The unnormalized DFT of a probability mass function is

$$\mathcal{F}p_X(k) = \sum_x p_X(x)\zeta_D^{-kx} = \mathbb{E}\left[\zeta_D^{-kX}\right].$$

Independence becomes multiplication:

$$\mathcal{F}p_{X+Y}(k) = \mathcal{F}p_X(k)\mathcal{F}p_Y(k).$$

Repeated convolution explains random-walk smoothing

If $S_t = X_1 + \cdots + X_t$ for independent, identically distributed steps with law p, then

$$p_{S_t} = \underbrace{p * p * \cdots * p}_{t \text{ times}}, \qquad \mathcal{F}p_{S_t}(k) = \big(\mathcal{F}p(k)\big)^t.$$

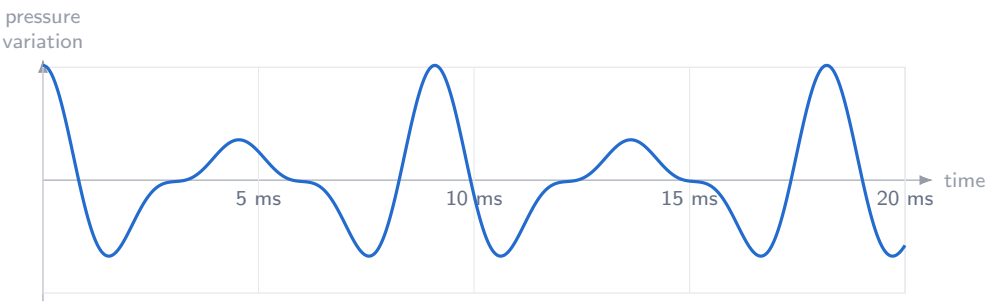
- The constant mode satisfies $\mathcal{F}p(0) = 1$.
- Modes with $|\mathcal{F}p(k)| < 1$ decay geometrically in t.
- The distribution becomes smoother as high-frequency variation disappears.

This spectral picture extends from finite random walks to diffusion and the heat equation.

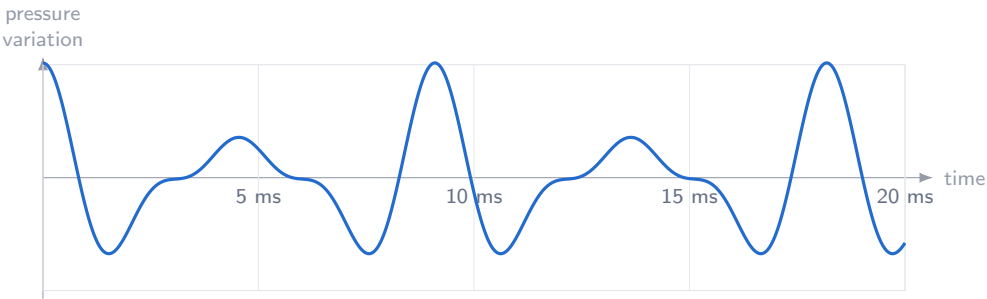
A microphone records a mixture, not separate notes

A microphone samples variations in air pressure over time. In an idealized example, three notes produce

$$p(t) = 0.85\cos(2\pi \cdot 220t) + 0.60\cos(2\pi \cdot 330t) + 0.40\cos(2\pi \cdot 440t).$$



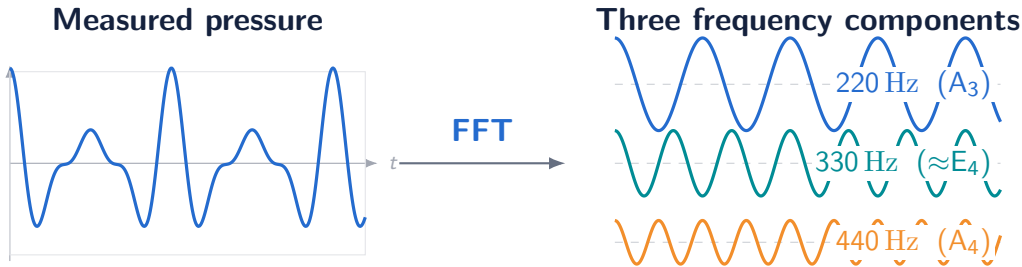
Can we recover the three contributing signals?



What is hidden in this graph?

The time-domain waveform shows the total pressure variation, but the three contributing notes are difficult to read off directly.

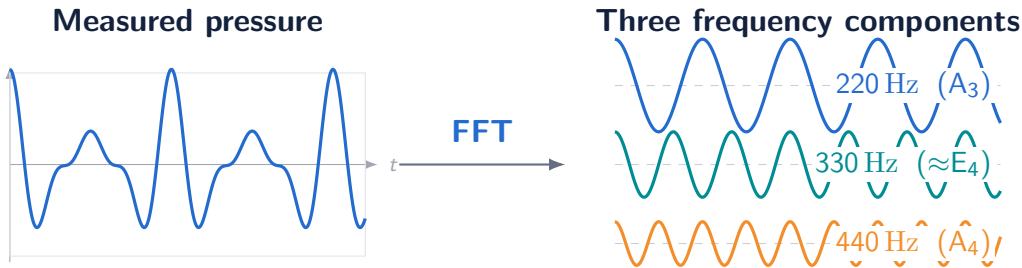
The FFT separates the mixture into pure tones



Real part of a character is just a wave

Note that $\chi_k(j) = \zeta_D^{kj} = \cos(jk\omega_0) + i\sin(jk\omega_0)$ where $\omega_0 = 2\pi/D$. This is just a wave with frequency $k\omega_0$. FFT lets us decompose observed f as a linear combination $f(x) = \sum_k \hat{f}(k)\cos(x \cdot k\omega_0)$ of waves with various frequencies.

The FFT separates the mixture into pure tones



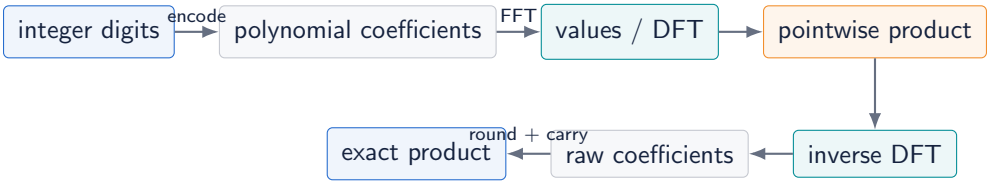
Idealized audio-processing interpretation

Peaks in the Fourier transform reveal the frequencies—and therefore the notes—that make up the sound. Real instruments also produce harmonics, transients, and noise, so practical pitch detection requires additional care.

Part 6

Synthesis

The whole reduction in one picture



Central takeaway

Multiplication becomes fast after a change of representation turns a global convolution into independent scalar products.

What to remember

1. Karatsuba: 3 half-size products give $\Theta(n^{\log_2 3})$ time.
2. Digit multiplication is polynomial coefficient convolution plus carries.
3. Roots of unity enable a radix-2 divide-and-conquer evaluation algorithm.
4. FFT and inverse FFT each cost $\Theta(n \log n)$.
5. Orthogonality proves inversion; zero-padding prevents cyclic wrap-around.
6. Fourier characters diagonalize convolution, linking multiplication to probability, random walks, and diffusion.