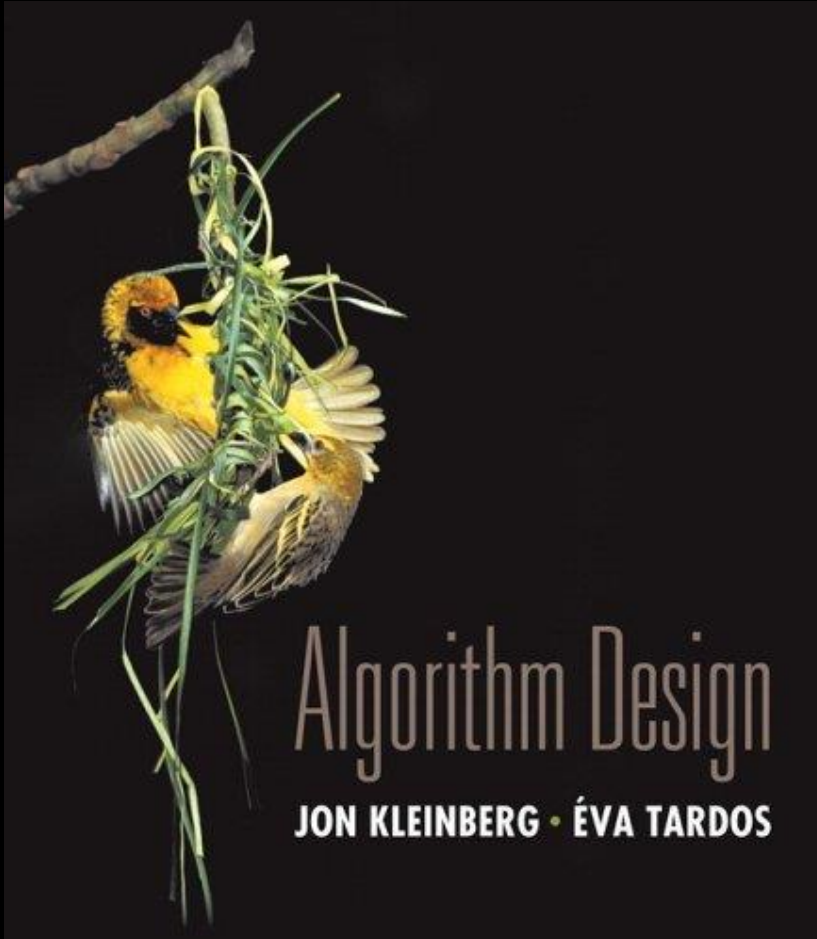


Chapter 5

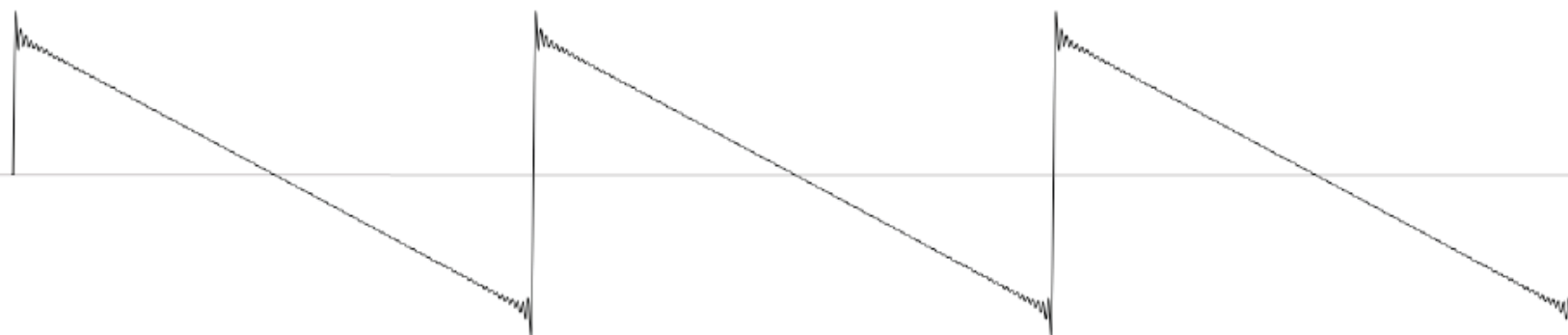
Divide and Conquer



Slides by Kevin Wayne.
Copyright © 2005 Pearson-Addison Wesley.
All rights reserved.
Updated by T. K. Prasad

Fourier analysis

Fourier theorem. [Fourier, Dirichlet, Riemann] Any (sufficiently smooth) periodic function can be expressed as the sum of a series of sinusoids.



$$y(t) = \frac{2}{\pi} \sum_{k=1}^N \frac{\sin kt}{k} \quad N = 100$$

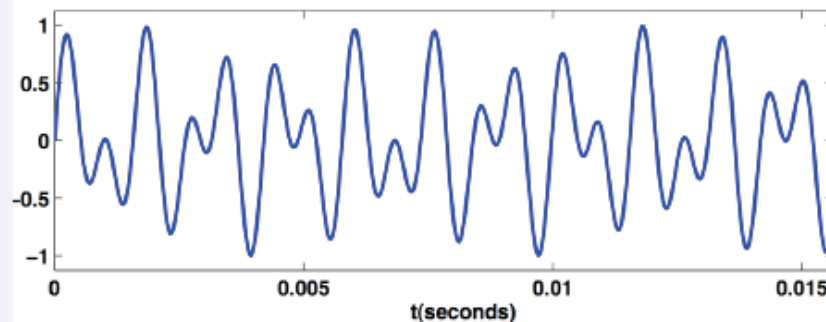
Time domain vs. frequency domain

Signal. [touch tone button 1] $y(t) = \frac{1}{2} \sin(2\pi \cdot 697 t) + \frac{1}{2} \sin(2\pi \cdot 1209 t)$

Time domain.

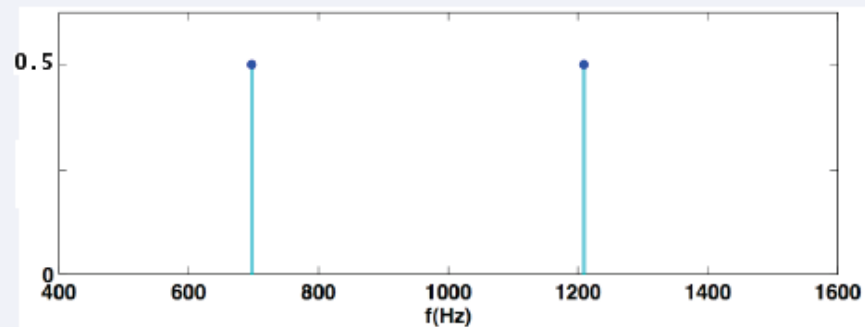


sound
pressure



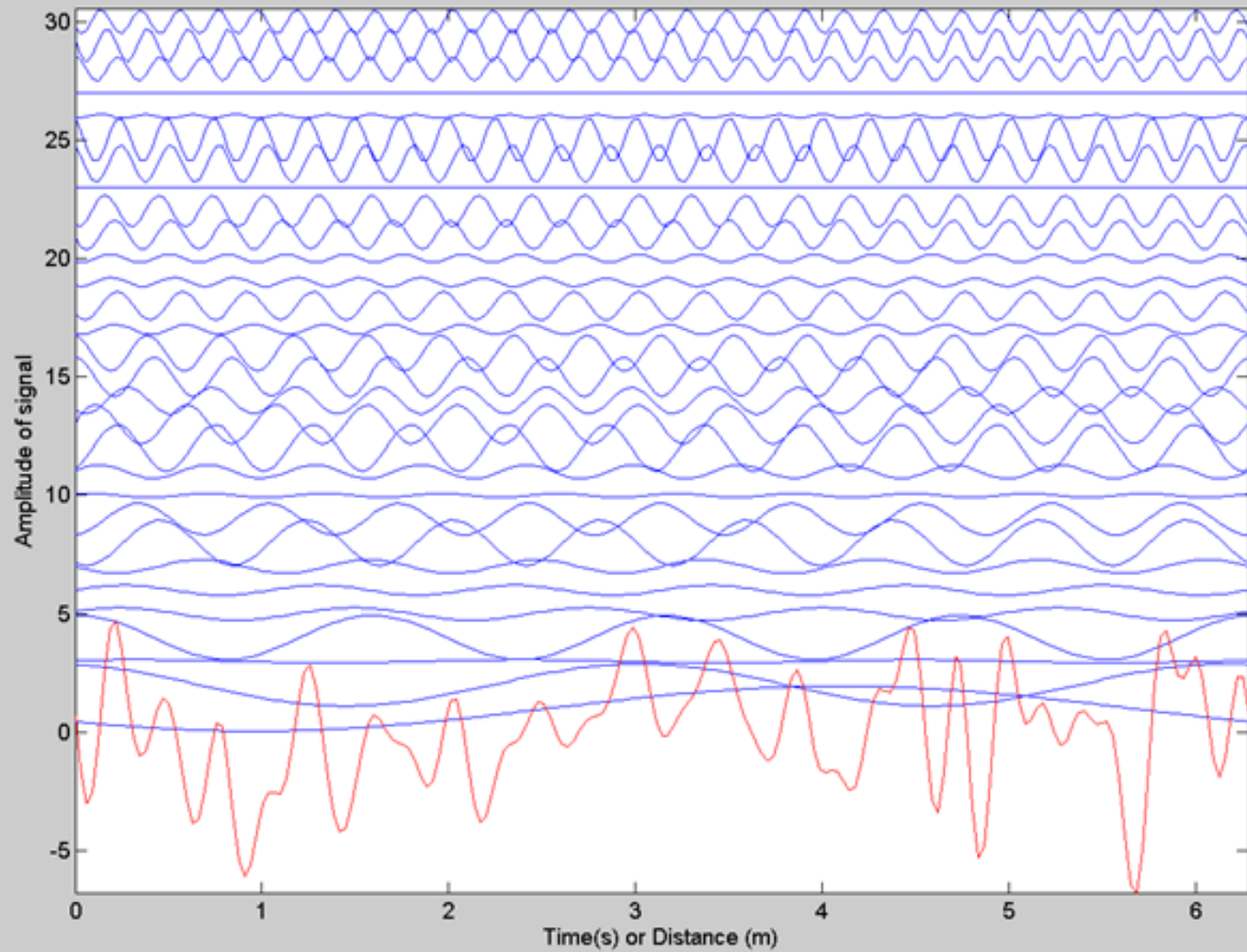
Frequency domain.

amplitude

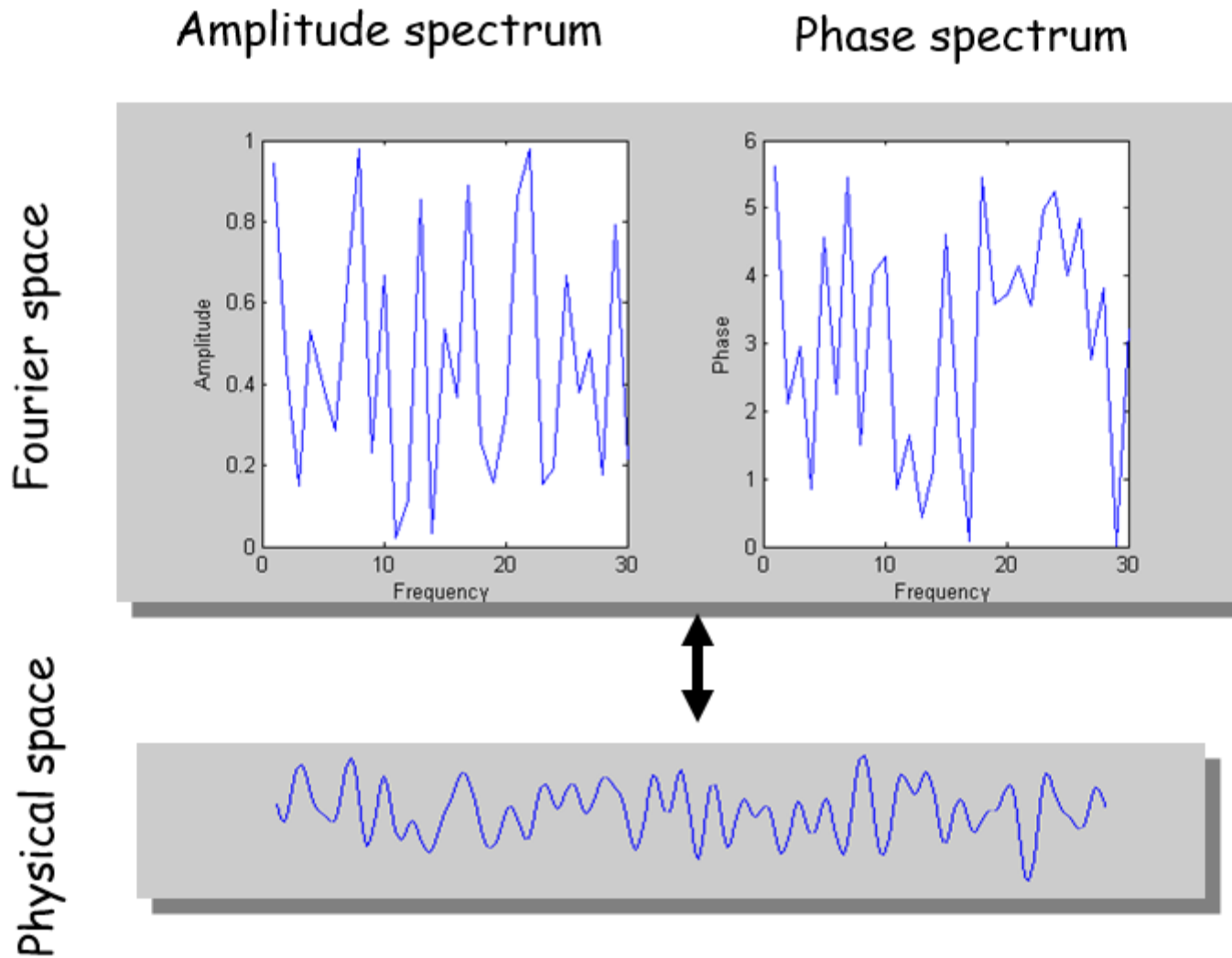


Reference: Cleve Moler, Numerical Computing with MATLAB

Spectral Synthesis : Computational Geophysics and Data Analysis

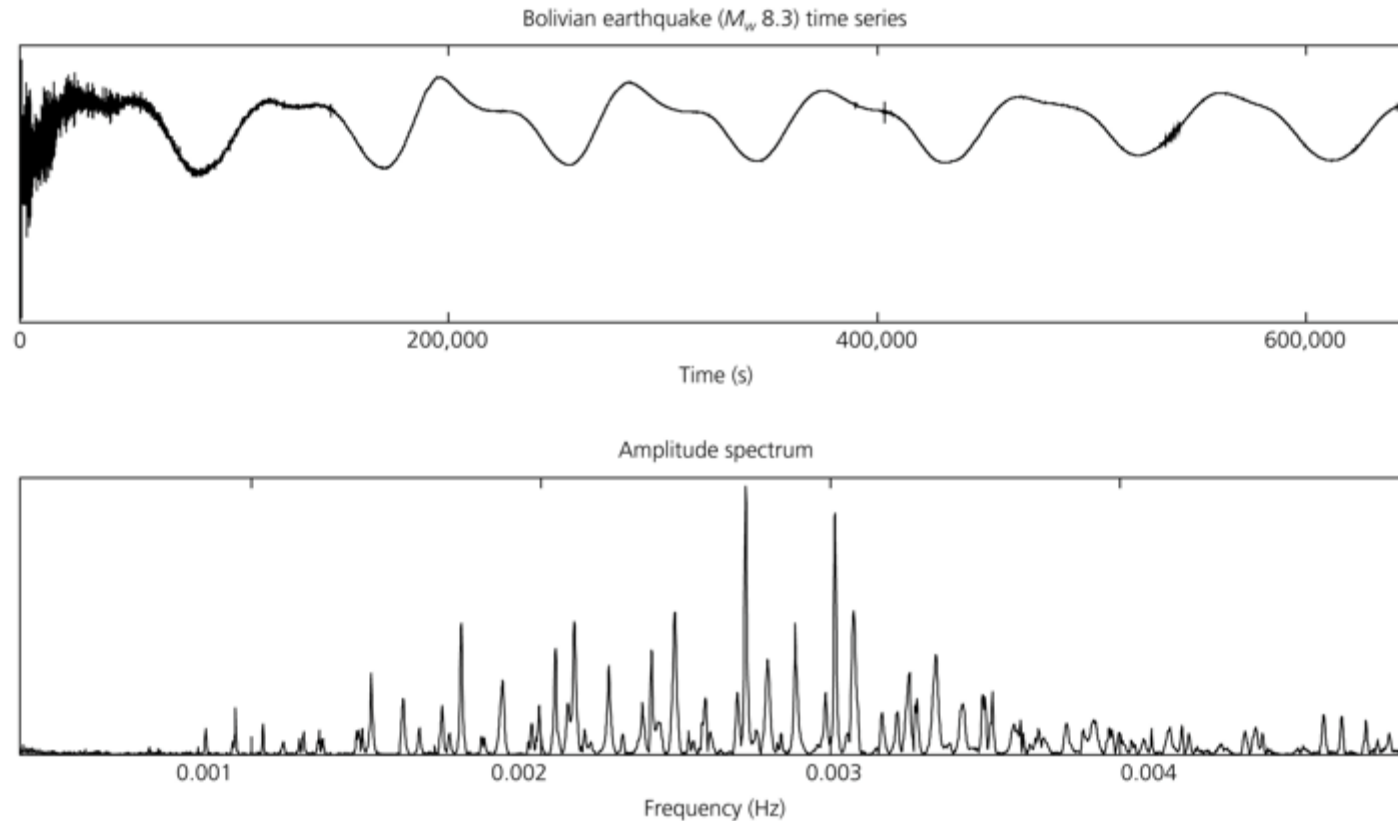


Spectral Synthesis : Computational Geophysics and Data Analysis

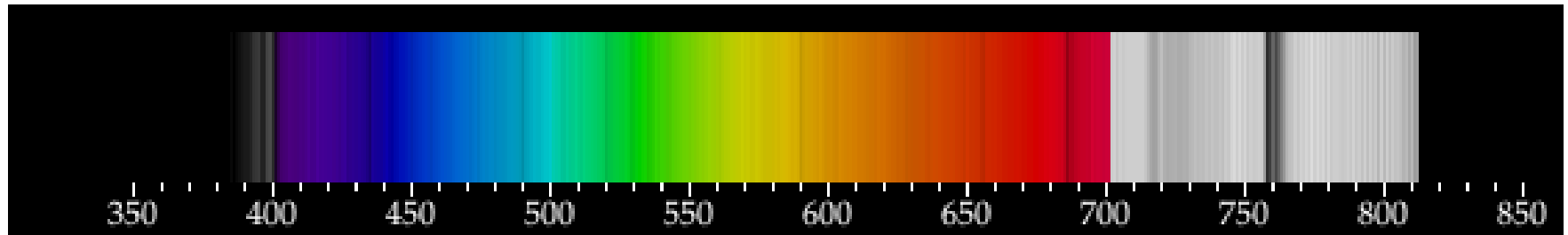


Amplitude Spectrum of an Earthquake

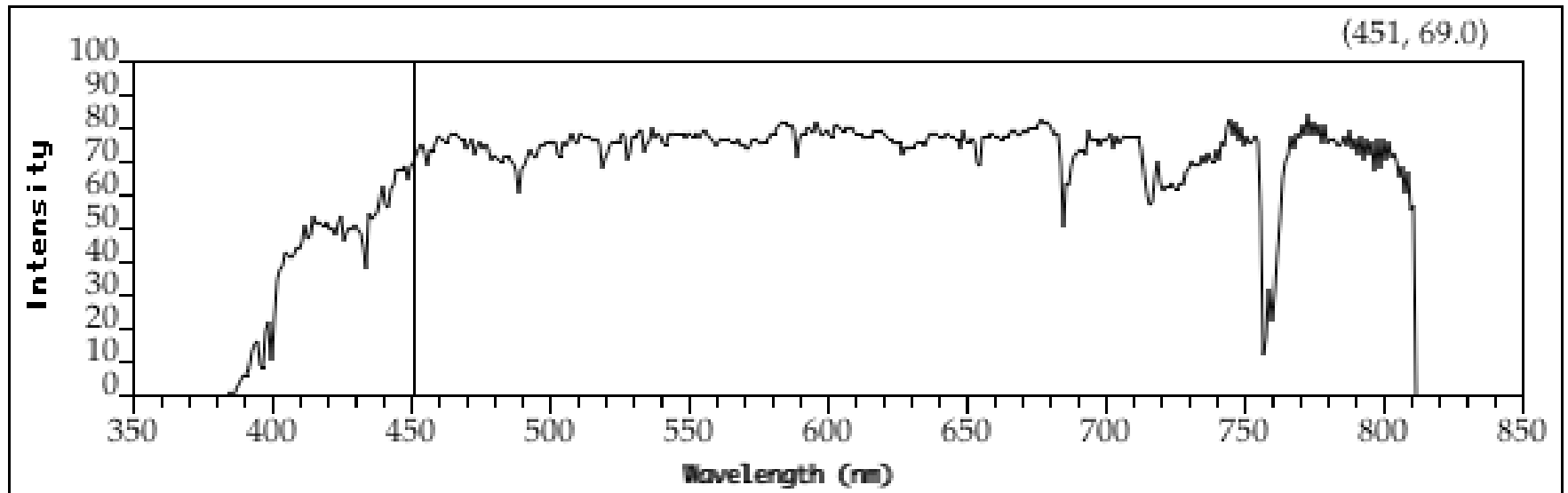
Figure 6.2-4: Amplitude spectra of a vertical-component seismogram from the great 1994 Bolivian earthquake.



Electromagnetic Spectrum of Star Light

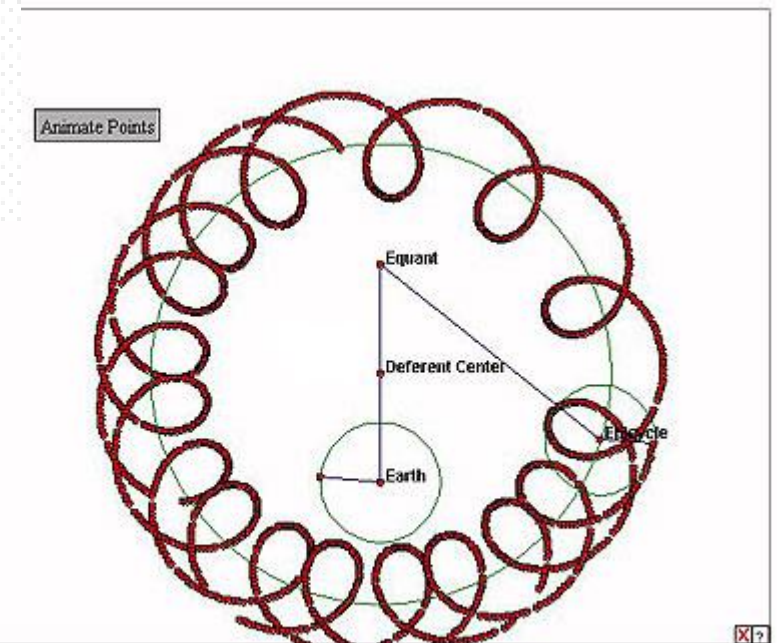
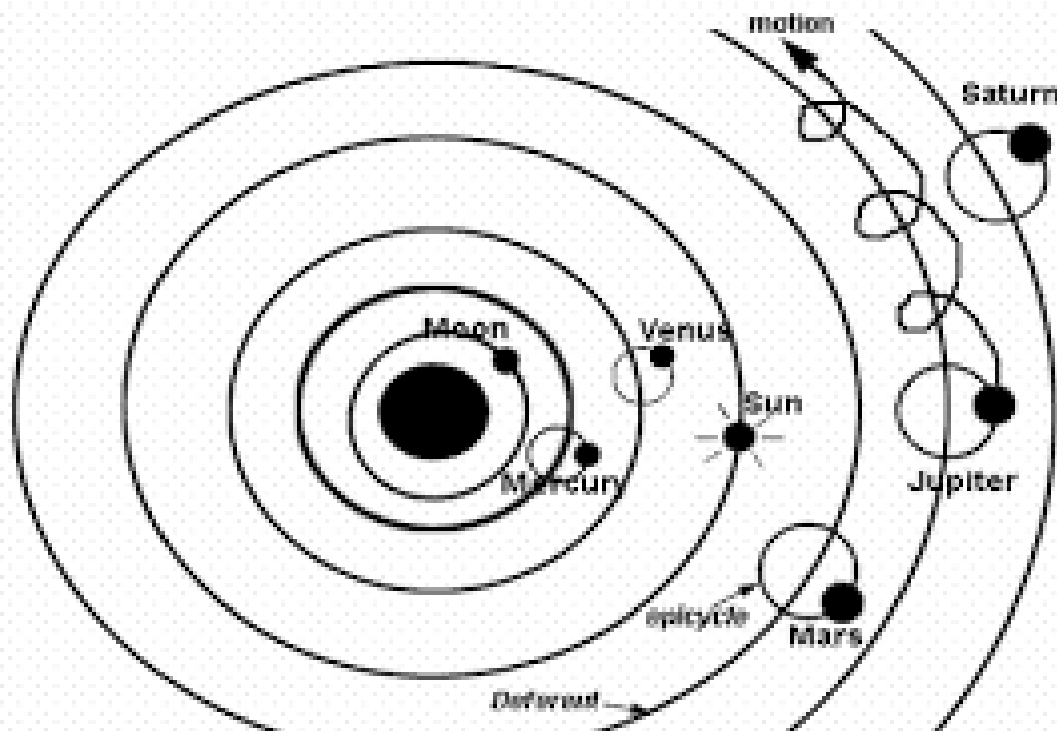


Emission Graph

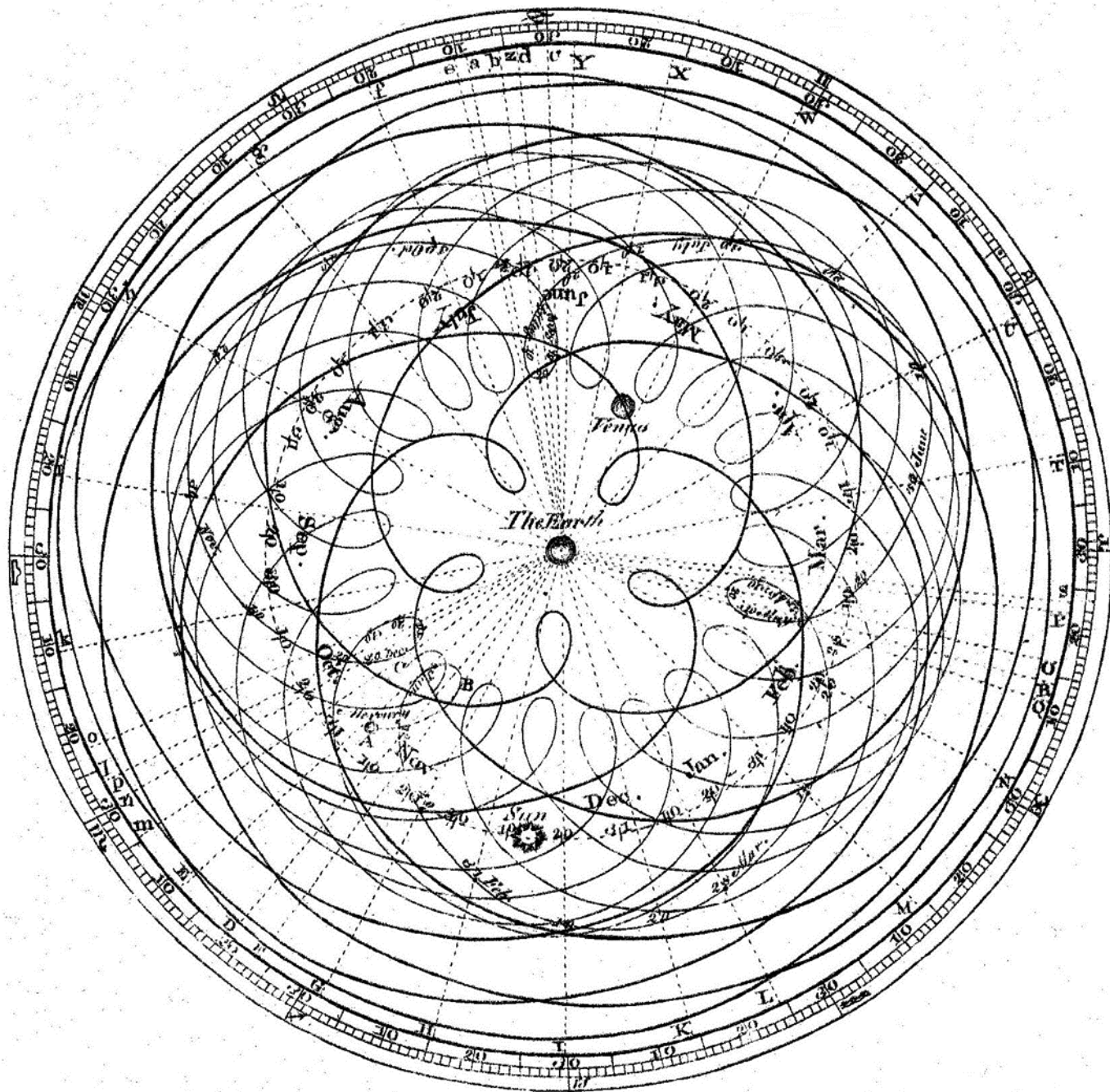


Electromagnetic Spectrum

Approximating planet orbits using epicycles in Geocentric Theory



Ptolemy's Model



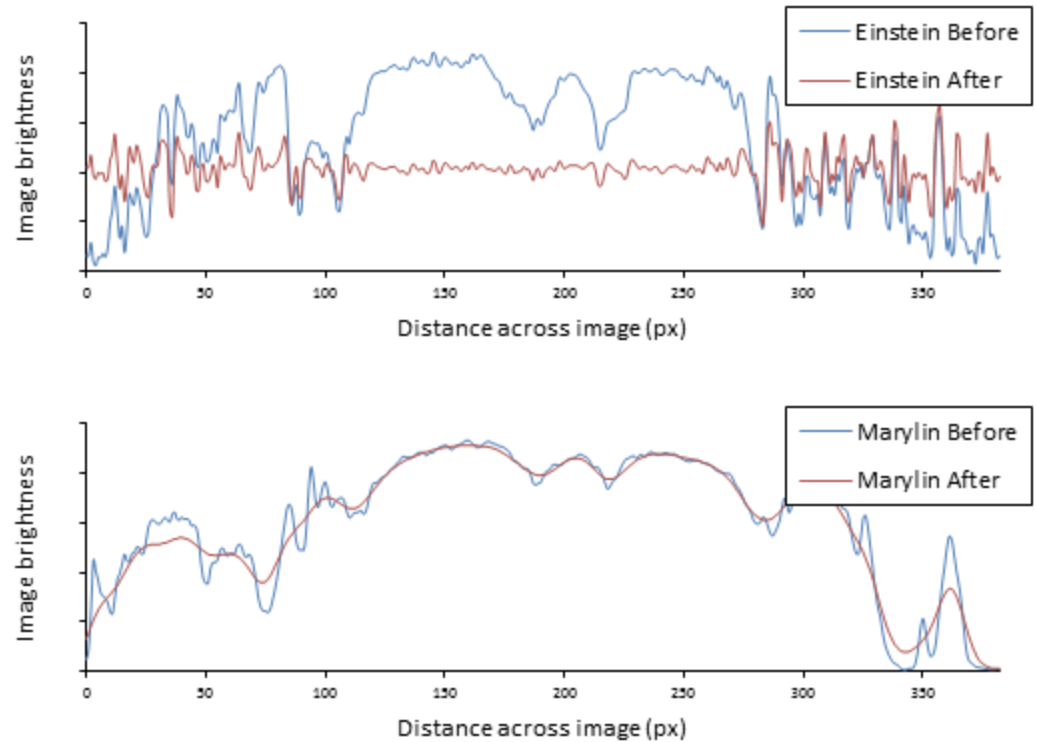
Hybrid Images

Constructing hybrid images is one particularly nice example of manipulating the Fourier spectrum of two images, and then combining them back into a single image.

E.g., The effect of applying low-pass and high-pass filters on Marilyn Monroe image are shown below.



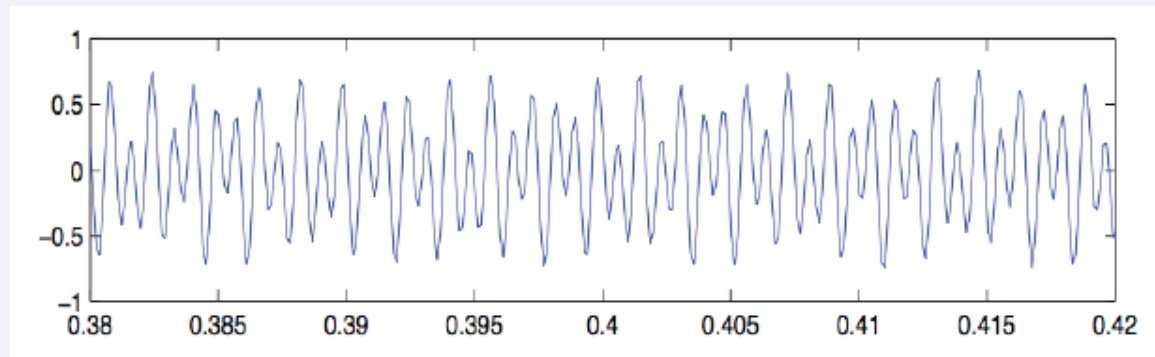
Hybrid Image: Marilyn Monroe + Albert Einstein



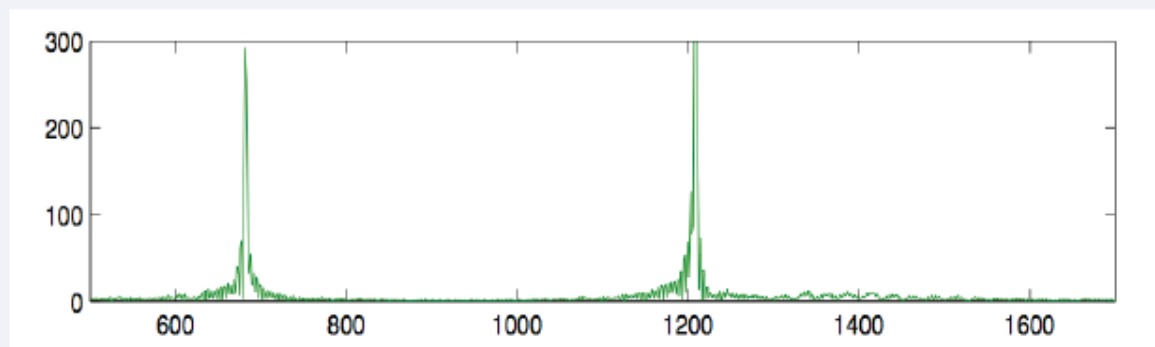
High-pass filter on Einstein's image and low-pass filter on Monroe's image.

Time domain vs. frequency domain : Digitizing analog signal

Signal. [recording, 8192 samples per second]

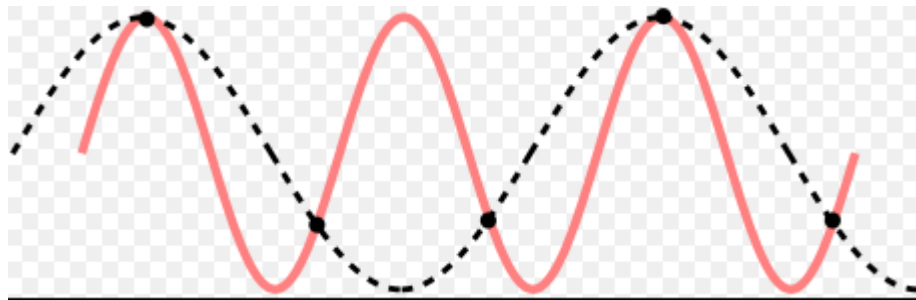


Magnitude of discrete Fourier transform.



Nyquist-Shannon Sampling Theorem

- If a function $x(t)$ contains no frequencies higher than B cps (cycles per second / Hertz), it is completely determined by giving its ordinates at a series of points spaced $< 1/(2B)$ seconds apart.
 - Equivalently, a band-limited analog signal can be “faithfully” reconstructed from its digitized version if it is sampled at Nyquist Rate, which is twice the largest frequency in the analog signal.



Aliasing absent
because orange
not allowed.

Using the 2D FFT for image compression

Image = 200x320 matrix of values

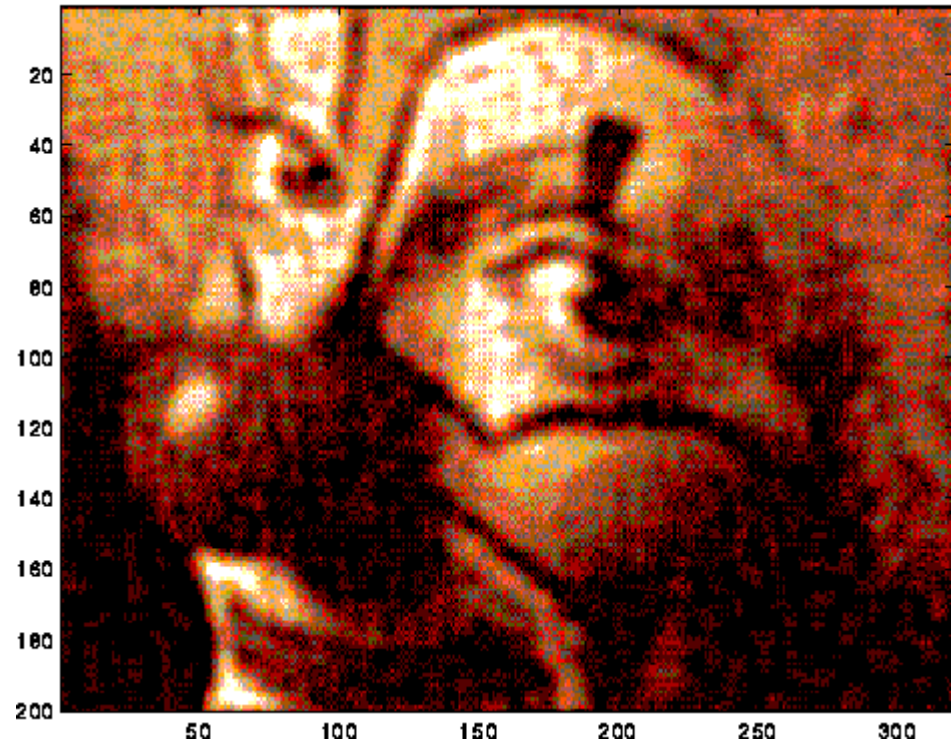
Compress by keeping largest 2.5% of FFT components

Similar idea used by jpeg

Original Image



Keep only largest 2.5% of entries of 2DFFT



Fast Fourier Transform (also: Fourier Series, Fourier Transform)

- **Signal Processing Viewpoint:** Approach to understand time-domain signal in terms of its frequency domain description as a superposition of sinusoids (characterized by their frequency, amplitude and phase)
 - A *periodic* signal of frequency f is made up of sinusoids of frequencies $f, 2f, 3f, \dots$ [harmonics]
- **Computational viewpoint:** An encoding technique for efficient computation, especially w.r.t. trade-offs involved in multiplying and evaluating polynomials

5.6 Convolution and FFT

Fast Fourier Transform: Applications

Applications.

- Optics, acoustics, quantum physics, telecommunications, control systems, signal processing, speech recognition, data compression, image processing.
- DVD, JPEG, MP3, MRI, CAT scan.
- Numerical solutions to Poisson's equation.

The FFT is one of the truly great computational developments of this [20th] century. It has changed the face of science and engineering so much that it is not an exaggeration to say that life as we know it would be very different without the FFT.

-Charles van Loan

Fast Fourier Transform: Brief History

Gauss (1805, 1866). Analyzed periodic motion of asteroid Ceres.

Runge-König (1924). Laid theoretical groundwork.

Danielson-Lanczos (1942). Efficient algorithm.

Cooley-Tukey (1965). Monitoring nuclear tests in Soviet Union and tracking submarines. Rediscovered and popularized FFT.

Importance not fully realized until the advent of digital computers.

Polynomials: Coefficient Representation

Polynomial. [coefficient representation]

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

$$B(x) = b_0 + b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1}$$

Add: $O(n)$ arithmetic operations.

$$A(x) + B(x) = (a_0 + b_0) + (a_1 + b_1)x + \dots + (a_{n-1} + b_{n-1})x^{n-1}$$

Evaluate: $O(n)$ using Horner's method.

$$A(x) = a_0 + (x(a_1 + x(a_2 + \dots + x(a_{n-2} + x(a_{n-1}))) \dots))$$

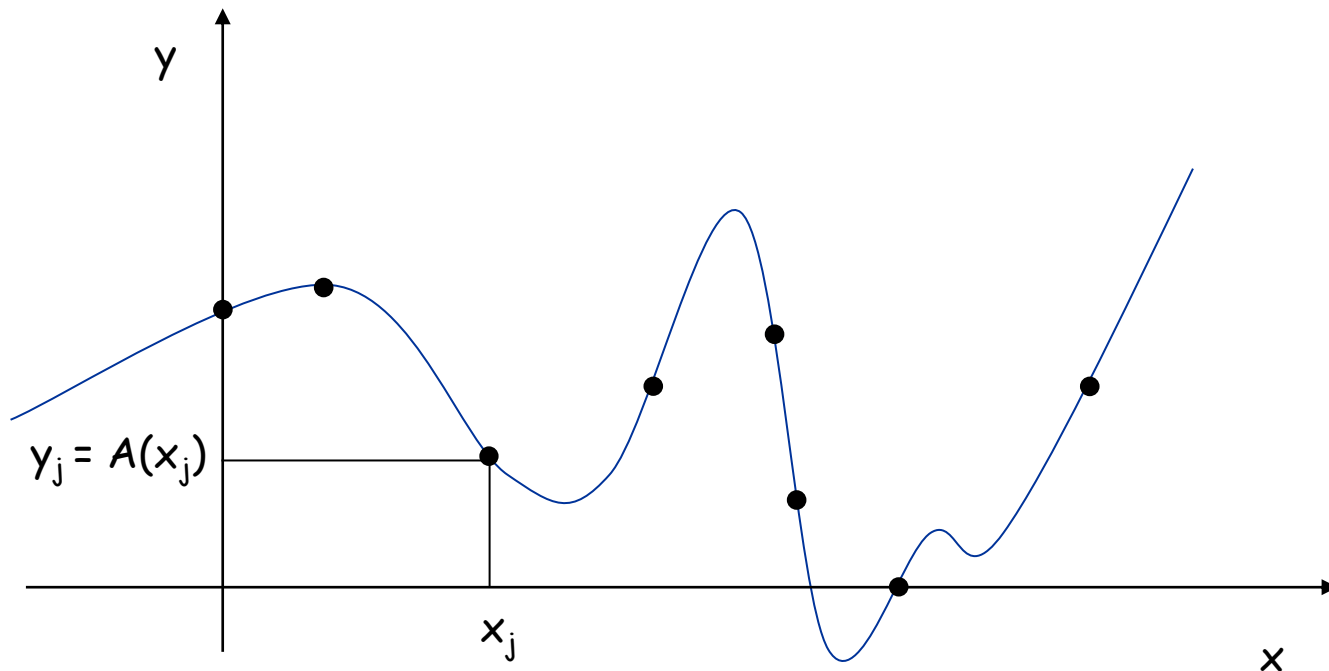
Multiply (convolve): $O(n^2)$ using brute force.

$$A(x) \cdot B(x) = \sum_{i=0}^{2n-2} c_i x^i, \text{ where } c_i = \sum_{j=0}^i a_j b_{i-j}$$

Polynomials: Point-Value Representation

Fundamental theorem of algebra. [Gauss, PhD thesis] A degree n polynomial with complex coefficients has n complex roots.

Corollary. A degree $n-1$ polynomial $A(x)$ is uniquely specified by its evaluation at n distinct values of x .



Polynomial Basics

Factoring polynomial: $x^2 + 2x - 15 = (x - 3)(x + 5)$

Zeros/roots of a polynomial equation/function:

$$(x + 5)(x - 3) = 0$$

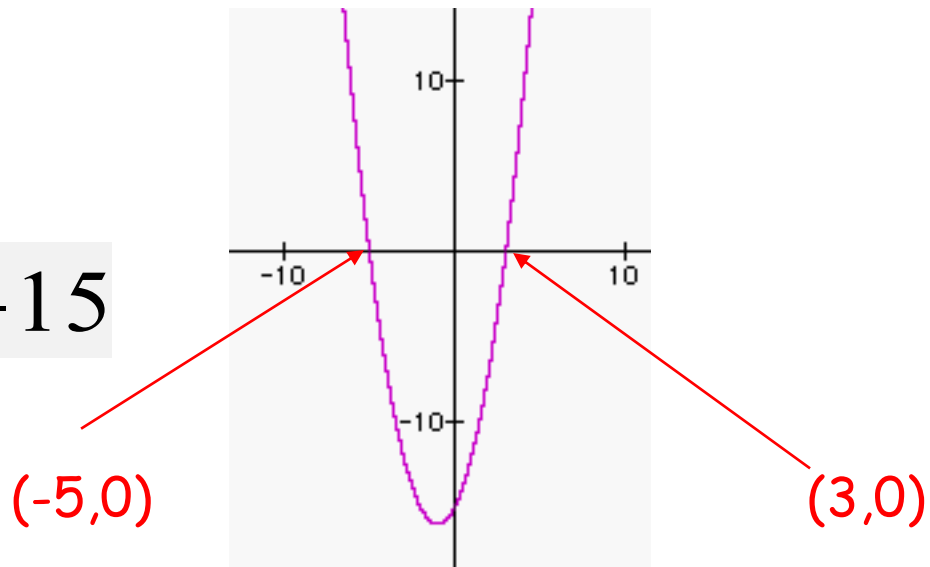
$$(x + 5) = 0 \quad \text{and} \quad (x - 3) = 0$$

$$x = -5$$

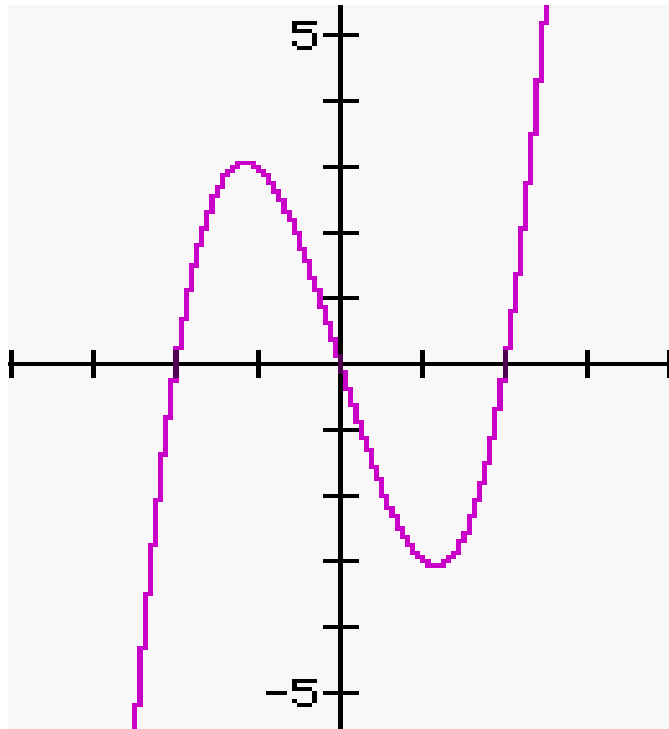
$$x = 3$$

Graphing:

$$y = x^2 + 2x - 15$$

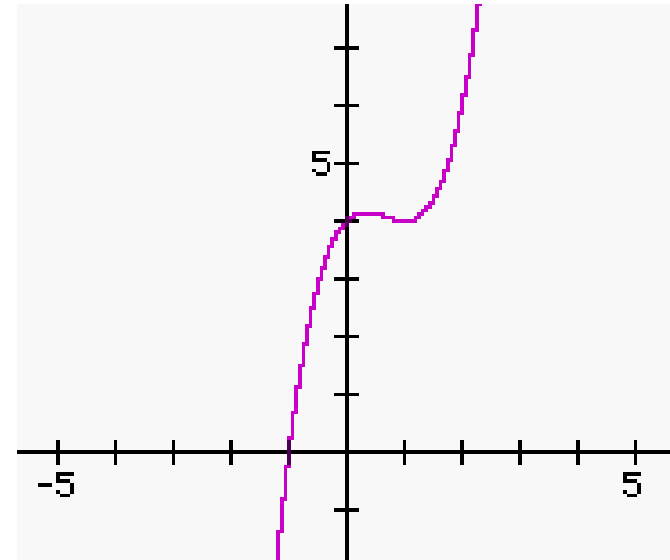


$$y = x^3 - 4x$$



Roots = -2, 0, 2

$$y = x^3 - 2x^2 + x + 4$$



Real Root = -1
(plus two complex roots)

$$f(x) = x^3 - 5x^2 - 7x + 51$$

$$= (x + 3)(x - (4 - i))(x - (4 + i))$$

Roots: -3, $4 - i$, $4 + i$.

Polynomials: Point-Value Representation

Polynomial. [point-value representation]

$$A(x) : (x_0, y_0), \dots, (x_{n-1}, y_{n-1})$$

$$B(x) : (x_0, z_0), \dots, (x_{n-1}, z_{n-1})$$

Add: $O(n)$ arithmetic operations.

$$A(x) + B(x) : (x_0, y_0 + z_0), \dots, (x_{n-1}, y_{n-1} + z_{n-1})$$

Multiply: $O(n)$, but need $2n$ points.

$$A(x) \times B(x) : (x_0, y_0 \times z_0), \dots, (x_{2n-1}, y_{2n-1} \times z_{2n-1})$$

Evaluate: $O(n^2)$ using Lagrange's formula.

$$A(x) = \sum_{k=0}^{n-1} y_k \frac{\tilde{O}(x - x_j)}{\tilde{O}(x_k - x_j)}$$

Lagrange Interpolating Polynomial

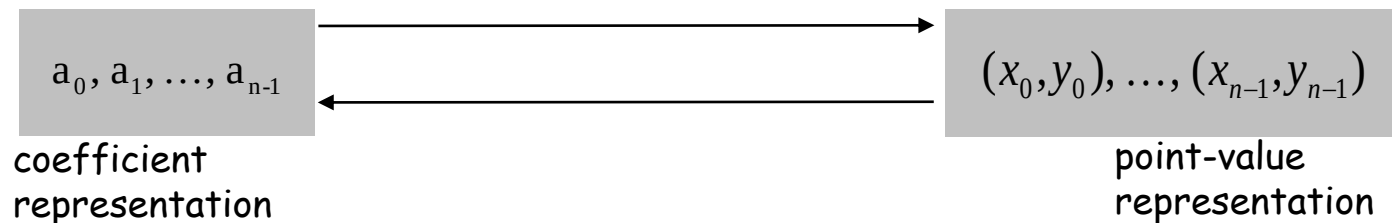
<u>Data Set</u>	<u>Polynomial</u>
$\{(x_1, y_1)\}$	$p(x) = y_1$
$\{(x_1, y_1), (x_2, y_2)\}$	$p(x) = y_1 \frac{(x - x_2)}{(x_1 - x_2)} + y_2 \frac{(x - x_1)}{(x_2 - x_1)}$
$\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$	$p(x) = y_1 \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + y_3 \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}$
$\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$	$p(x) = y_1 \frac{(x - x_2)(x - x_3) \cdots (x - x_m)}{(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_m)} + y_2 \frac{(x - x_1)(x - x_3) \cdots (x - x_m)}{(x_2 - x_1)(x_2 - x_3) \cdots (x_2 - x_m)} +$ $\cdots + y_m \frac{(x - x_1)(x - x_2) \cdots (x - x_{m-1})}{(x_m - x_1)(x_m - x_2) \cdots (x_m - x_{m-1})}$ $p(x) = \sum_{i=1}^m y_i \prod_{\substack{j=1 \\ j \neq i}}^m \frac{(x - x_j)}{(x_i - x_j)}$

Converting Between Two Polynomial Representations

Tradeoff. Fast evaluation or fast multiplication. We want both!

Representation	Multiply	Evaluate
Coefficient	$O(n^2)$	$O(n)$
Point-value	$O(n)$	$O(n^2)$

Idea. Make all ops fast by efficiently converting between two representations.



Converting Between Two Polynomial Representations: Brute Force

Coefficient to point-value. Given a polynomial $a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$, evaluate it at n distinct points $x_0, \dots, x_{n-1}$.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

$O(n^2)$ for matrix-vector multiply

$O(n^3)$ for Gaussian elimination

↑
Vandermonde matrix is invertible iff x_i distinct

Point-value to coefficient. Given n distinct points $x_0, \dots, x_{n-1}$ and values $y_0, \dots, y_{n-1}$, find unique polynomial $a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$ that has given values at given points.

Divide-and-conquer

Decimation in frequency. Break up polynomial into low and high powers.

- $A(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7.$
- $A_{low}(x) = a_0 + a_1x + a_2x^2 + a_3x^3.$
- $A_{high}(x) = a_4 + a_5x + a_6x^2 + a_7x^3.$
- $A(x) = A_{low}(x) + x^4 A_{high}(x).$

Decimation in time. Break up polynomial into even and odd powers.

- $A(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7.$
- $A_{even}(x) = a_0 + a_2x + a_4x^2 + a_6x^3.$
- $A_{odd}(x) = a_1 + a_3x + a_5x^2 + a_7x^3.$
- $A(x) = A_{even}(x^2) + x A_{odd}(x^2).$

Coefficient to Point-Value Representation: Intuition

Coefficient to point-value. Given a polynomial $a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$, evaluate it at n distinct points $x_0, \dots, x_{n-1}$.

Divide. Break up polynomial using even and odd power coefficients.

- $A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7.$
- $A_{\text{even}}(x) = a_0 + a_2 x + a_4 x^2 + a_6 x^3.$
- $A_{\text{odd}}(x) = a_1 + a_3 x + a_5 x^2 + a_7 x^3.$
- $A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2).$
- $A(-x) = A_{\text{even}}(x^2) - x A_{\text{odd}}(x^2).$

Intuition. Choose two points to be ± 1 .

- $A(1) = A_{\text{even}}(1) + 1 A_{\text{odd}}(1).$
- $A(-1) = A_{\text{even}}(1) - 1 A_{\text{odd}}(1).$

Can evaluate polynomial of degree $\leq n$ at 2 points by evaluating two polynomials of degree $\leq \frac{1}{2}n$ at 1 point.

Coefficient to Point-Value Representation: Intuition

Coefficient to point-value. Given a polynomial $a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$, evaluate it at n distinct points $x_0, \dots, x_{n-1}$.

Divide. Break up polynomial into even and odd power coefficients.

- $A(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 + a_7 x^7.$
- $A_{\text{even}}(x) = a_0 + a_2 x + a_4 x^2 + a_6 x^3.$
- $A_{\text{odd}}(x) = a_1 + a_3 x + a_5 x^2 + a_7 x^3.$
- $A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2).$
- $A(-x) = A_{\text{even}}(x^2) - x A_{\text{odd}}(x^2).$

Intuition. Choose four points to be $\pm 1, \pm i$.

- $A(1) = A_{\text{even}}(1) + 1 A_{\text{odd}}(1).$
- $A(-1) = A_{\text{even}}(1) - 1 A_{\text{odd}}(1).$
- $A(i) = A_{\text{even}}(-1) + i A_{\text{odd}}(-1).$
- $A(-i) = A_{\text{even}}(-1) - i A_{\text{odd}}(-1).$

Can evaluate polynomial of degree $\leq n$ at 4 points by evaluating two polynomials of degree $\leq \frac{1}{2}n$ at 2 points.

Discrete Fourier Transform

Coefficient to point-value. Given a polynomial $a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$, evaluate it at n distinct points $x_0, \dots, x_{n-1}$.

Key idea: choose $x_k = \omega^k$ where ω is principal n^{th} root of unity.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

↑
Discrete Fourier transform

↑
Fourier matrix F_n

Roots of Unity

Def. An n^{th} root of unity is a complex number x such that $x^n = 1$.

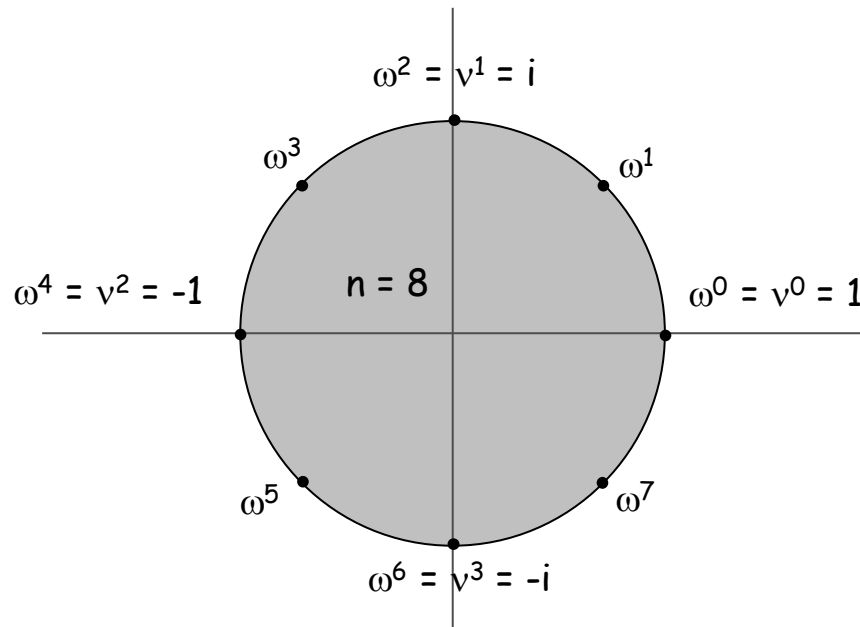
Euler's identity. $e^{ix} = \cos x + i \sin x$.

Fact. The n^{th} roots of unity are: $\omega^0, \omega^1, \dots, \omega^{n-1}$ where $\omega = e^{2\pi i / n}$.

Pf. $(\omega^k)^n = (e^{2\pi i k / n})^n = (e^{\pi i})^{2k} = (-1)^{2k} = 1$.

Fact. The $\frac{1}{2}n^{\text{th}}$ roots of unity are: $v^0, v^1, \dots, v^{n/2-1}$ where $v = e^{4\pi i / n}$.

Fact. $\omega^2 = v$ and $(\omega^2)^k = v^k$.



Fast Fourier Transform (corrected)

Goal. Evaluate a degree $n-1$ polynomial $A(x) = a_0 + \dots + a_{n-1} x^{n-1}$ at all n^{th} roots of unity: $\omega^0, \omega^1, \dots, \omega^{n-1}$.

Divide. Break up polynomial into even and odd power coefficients.

- $A_{\text{even}}(x) = a_0 + a_2x + a_4x^2 + \dots + a_{(n-2)} x^{(n/2-1)}.$
- $A_{\text{odd}}(x) = a_1 + a_3x + a_5x^2 + \dots + a_{(n-1)} x^{(n/2-1)}.$
- $A(x) = A_{\text{even}}(x^2) + x A_{\text{odd}}(x^2)$

Conquer. Evaluate polynomials $A_{\text{even}}(x)$ and $A_{\text{odd}}(x)$ at the $\frac{1}{2}n^{\text{th}}$ roots of unity: $v^0, v^1, \dots, v^{n/2-1}$.
(Recall $\omega^2 = v$)

Combine.

- $A(\omega^k) = A_{\text{even}}(v^k) + \omega^k A_{\text{odd}}(v^k), \quad 0 \leq k < n/2$
- $A(\omega^{k+\frac{1}{2}n}) = A_{\text{even}}(v^k) - \omega^k A_{\text{odd}}(v^k), \quad 0 \leq k < n/2$

$$\omega^{k+\frac{1}{2}n} = -\omega^k$$

$$\omega^{\frac{1}{2}n} = (e^{2\pi i / n})^{\frac{1}{2}n} = \cos \pi + i \sin \pi = -1 + i 0$$

$$\begin{aligned} v^k &= (\omega^2)^k \\ &= (\omega^k)^2 \\ &= (\omega^k)^2 (\omega^n) \\ &= (\omega^{k+\frac{1}{2}n})^2 \end{aligned}$$

FFT Algorithm

FFT ($n, a_0, a_1, a_2, \dots, a_{n-1}$)

if ($n = 1$) return a_0 .

$(e_0, e_1, \dots, e_{n/2-1}) \leftarrow \text{FFT}(n/2, a_0, a_2, a_4, \dots, a_{n-2})$.

$(d_0, d_1, \dots, d_{n/2-1}) \leftarrow \text{FFT}(n/2, a_1, a_3, a_5, \dots, a_{n-1})$.

for $k = 0$ to $n/2 - 1$.

$$\omega^k \leftarrow e^{2\pi i k/n}.$$

$$y_k \leftarrow e_k + \omega^k d_k.$$

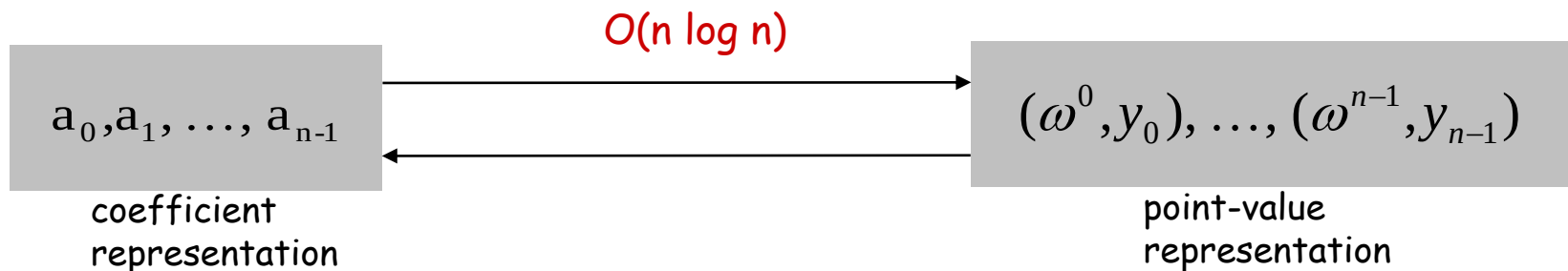
$$y_{k+n/2} \leftarrow e_k - \omega^k d_k.$$

return $(y_0, y_1, y_2, \dots, y_{n-1})$.

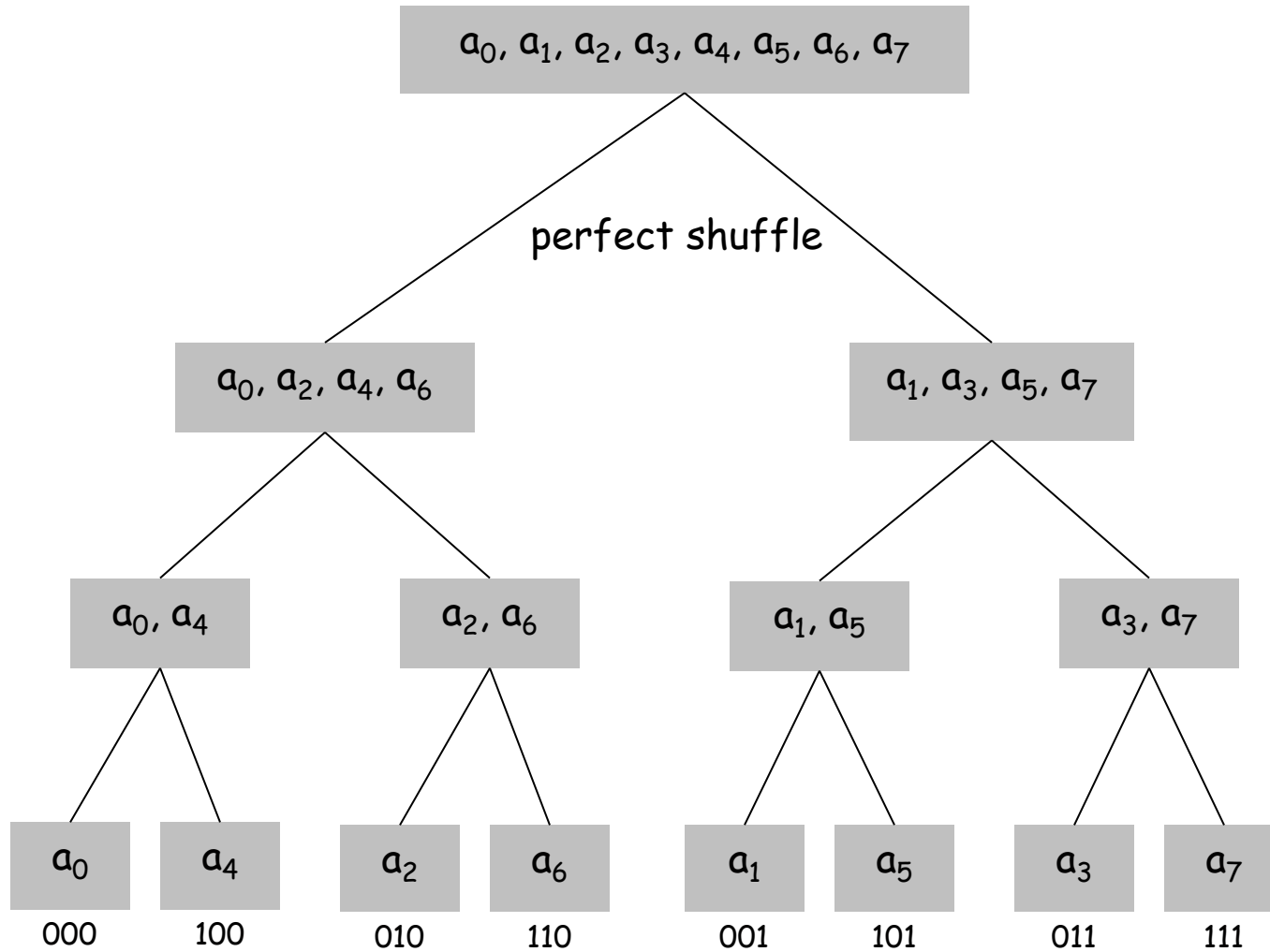
FFT Summary

Theorem. FFT algorithm evaluates a degree $n-1$ polynomial at each of the n^{th} roots of unity in $O(n \log n)$ steps. ↑
assumes n is a power of 2

Running time. $T(n) = 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$.



Recursion Tree



"bit-reversed" order

FFT Example

(coefficient)

$(a_0, a_1, a_2, a_3) =$

$(36, 22, 45, 15)$



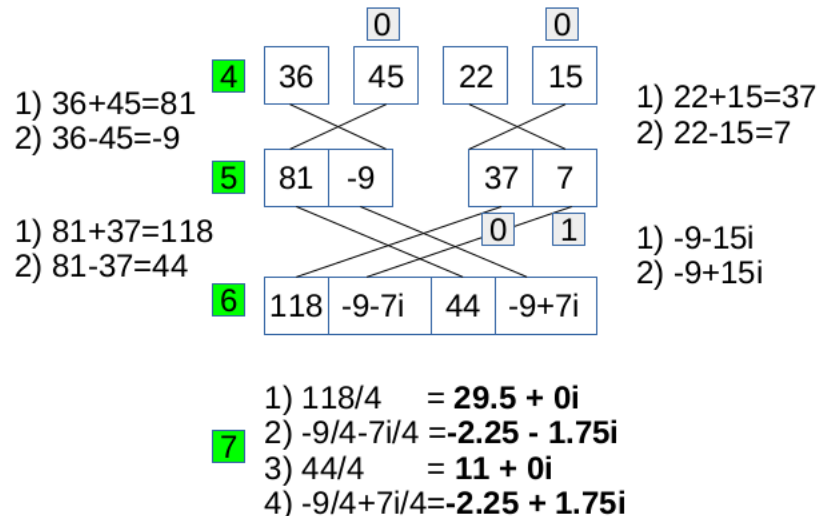
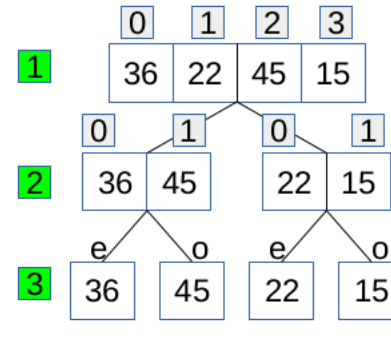
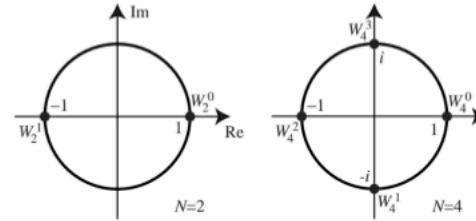
(point-value)

$(p_1, p_i, p_{-1}, p_{-i}) =$

$(118, -9 + 7i,$

$44, -9 - 7i)$

(If this were **Inverse FFT** Example from point-value to coefficients, we need to divide the end result by 4 as shown in the last step.)



Point-Value to Coefficient Representation: Inverse DFT

Goal. Given the values $y_0, \dots, y_{n-1}$ of a degree $n-1$ polynomial at the n points $\omega^0, \omega^1, \dots, \omega^{n-1}$, find unique polynomial $a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$ that has given values at given points.

$$\begin{array}{c}
 \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & w^1 & w^2 & w^3 & \dots & w^{n-1} \\ 1 & w^2 & w^4 & w^6 & \dots & w^{2(n-1)} \\ 1 & w^3 & w^6 & w^9 & \dots & w^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & w^{3(n-1)} & \dots & w^{(n-1)(n-1)} \end{pmatrix}^{-1} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix}
 \end{array}$$

$\uparrow$
 Inverse DFT

$\uparrow$
 Fourier matrix inverse $(F_n)^{-1}$

Inverse FFT

Claim. Inverse of Fourier matrix is given by the following formula.

$$G_n = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ \omega^{-1} & \omega^{-2} & \omega^{-4} & \omega^{-6} & \dots & \omega^{-(n-1)} \\ \omega^{-2} & \omega^{-4} & \omega^{-6} & \omega^{-9} & \dots & \omega^{-3(n-1)} \\ \omega^{-3} & \omega^{-6} & \omega^{-9} & \omega^{-12} & \dots & \omega^{-6(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^{-(n-1)} & \omega^{-2(n-1)} & \omega^{-3(n-1)} & \omega^{-4(n-1)} & \dots & \omega^{-(n-1)(n-1)} \end{pmatrix}$$

Consequence. To compute inverse FFT, apply same algorithm but use $\omega^{-1} = e^{-2\pi i / n}$ as principal n^{th} root of unity (and divide by n).

Inverse FFT: Proof of Correctness

Claim. F_n and G_n are inverses.

Pf.

$$(F_n G_n)_{kk} = \frac{1}{n} \sum_{j=0}^{n-1} w^{kj} w^{-jk} = \frac{1}{n} \sum_{j=0}^{n-1} w^{(k-k)j} = \begin{cases} 1 & \text{if } k = k \\ 0 & \text{otherwise} \end{cases}$$

↑
summation lemma

Summation lemma. Let ω be a principal n^{th} root of unity. Then

$$\sum_{j=0}^{n-1} \omega^{kj} = \begin{cases} n & \text{if } k \equiv 0 \pmod{n} \\ 0 & \text{otherwise} \end{cases}$$

Pf.

- If k is a multiple of n then $\omega^k = 1 \Rightarrow$ **sums to n .**
- Each n^{th} root of unity ω^k is a root of $x^n - 1$.

$$x^n - 1 = (x - 1)(1 + x + x^2 + \dots + x^{n-1}).$$
- if $\omega^k \neq 1$ we have: $1 + \omega^k + \omega^{k(2)} + \dots + \omega^{k(n-1)} = 0 \Rightarrow$ **sums to 0.**

Inverse FFT: Algorithm

Need to divide the final result by n .

Inverse-FFT ($n, a_0, a_1, a_2, \dots, a_{n-1}$)

if ($n = 1$) return a_0 .

$(e_0, e_1, \dots, e_{n/2-1}) \leftarrow \text{Inverse-FFT}(n/2, a_0, a_2, a_4, \dots, a_{n-2})$.

$(d_0, d_1, \dots, d_{n/2-1}) \leftarrow \text{Inverse-FFT}(n/2, a_1, a_3, a_5, \dots, a_{n-1})$.

for $k = 0$ to $n/2 - 1$.

$\omega^k \leftarrow e^{-2\pi i k/n}$.

$y_k \leftarrow (e_k + \omega^k d_k)$.

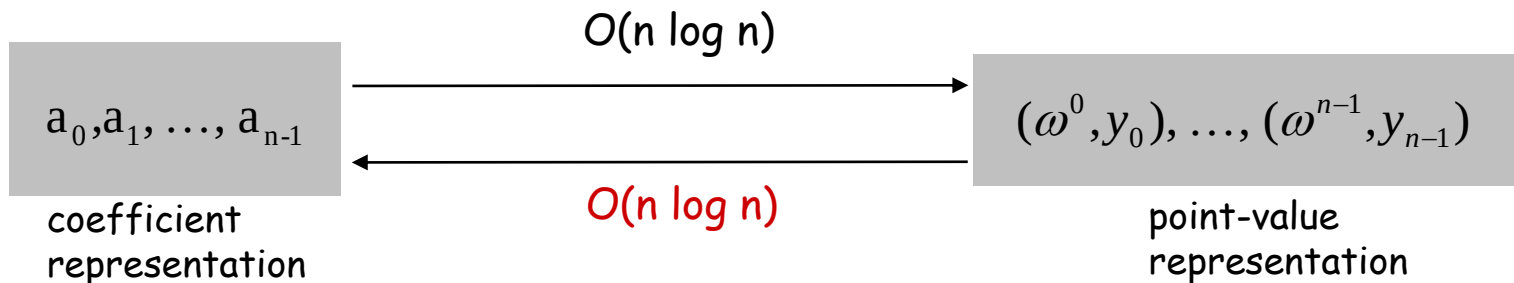
$y_{k+n/2} \leftarrow (e_k - \omega^k d_k)$.

return $(y_0, y_1, y_2, \dots, y_{n-1})$.

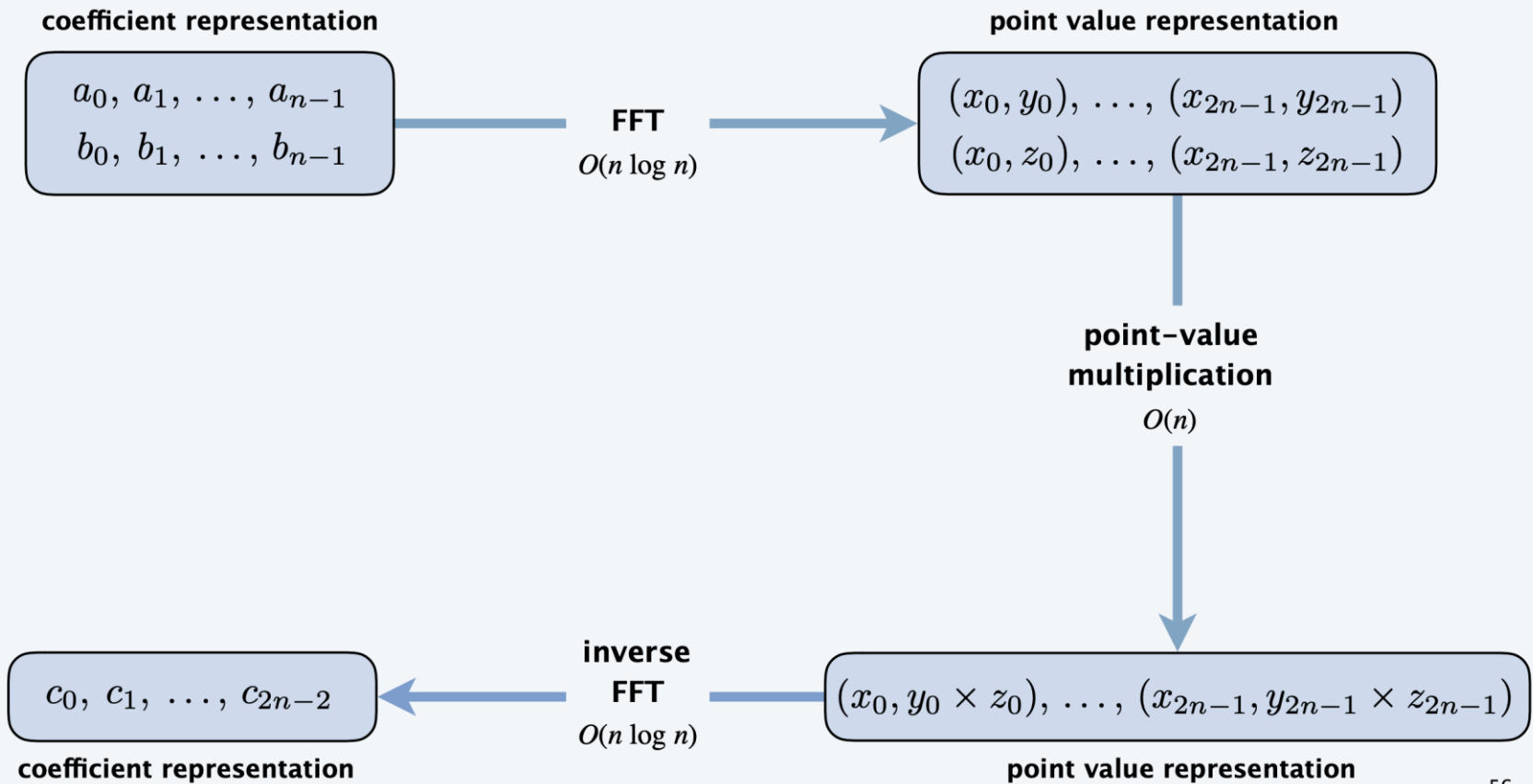
Inverse FFT Summary

Theorem. Inverse FFT algorithm interpolates a degree $n-1$ polynomial given values at each of the n^{th} roots of unity in $O(n \log n)$ steps.

↑
assumes n is a power of 2



Application. Polynomial multiplication (coefficient representation).



FFT in Practice

Fastest Fourier transform in the West. [Frigo and Johnson]

- Optimized C library.
- **Features:** DFT, DCT, real, complex, any size, any dimension.
- Won 1999 Wilkinson Prize for Numerical Software.
- Portable, competitive with vendor-tuned code.

Implementation details.

- Instead of executing predetermined algorithm, it evaluates your hardware and uses a special-purpose compiler to optimize code catered to "shape" of the problem.
- Core algorithm is nonrecursive version of Cooley-Tukey radix 2 FFT.
- $O(n \log n)$, even for prime sizes.

Reference: <http://www.fftw.org>

Integer Multiplication

Integer multiplication. Given two n bit integers $a = a_{n-1} \dots a_1 a_0$ and $b = b_{n-1} \dots b_1 b_0$, compute their product $c = a \times b$.

Convolution algorithm.

- Form two polynomials.
- **Note:** $a = A(2)$, $b = B(2)$.
- Compute $C(x) = A(x) \times B(x)$.
- Evaluate $C(2) = a \times b$.
- **Running time:** $O(n \log n)$ **complex arithmetic steps.**

$$A(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

$$B(x) = b_0 + b_1x + b_2x^2 + \dots + b_{n-1}x^{n-1}$$

Theory. [Schönhage-Strassen 1971] $O(n \log n \log \log n)$ **bit operations.**

Theory. [Fürer 2007] $n \log n 2^{O(\log^* n)}$ bit operations.

Practice. [GNU Multiple Precision Arithmetic Library] GMP proclaims to be "the fastest bignum library on the planet." It uses brute force, Karatsuba, and FFT, depending on the size of n .