Fast Fourier Transformation 1

John Augustine (IIT Madras)

Krishna Palem (RICE University)

Source:
These slides are based on source material from *Algorithms* by *Dasgupta et al* (Published by McGraw-Hill Education.)

Acknowledgement: We thank Parishkrati and Ashutosh Ingole (both at IIT Madras) for collaboration that resulted in these slides.

Topics in this Lecture

- Fourier Spectra and Transforms
  - Spectral Analysis of Boolean Functions
  - Spectral Analysis of (Discrete) Periodic Functions
  - Building the Analogy between them
- A Brief Introduction to the Fast Fourier Transform Algorithm
- Application 1: Exploiting Sparsity in Signals
- Application 2: Fast multiplication of polynomials
- The Road ahead (for the FFT presentation in next lecture)

Fourier Spectra and Transforms

A perspective

Spectral Analysis of Boolean Functions A Recap

$$f: \{0,1\}^n \to \mathbb{R}$$

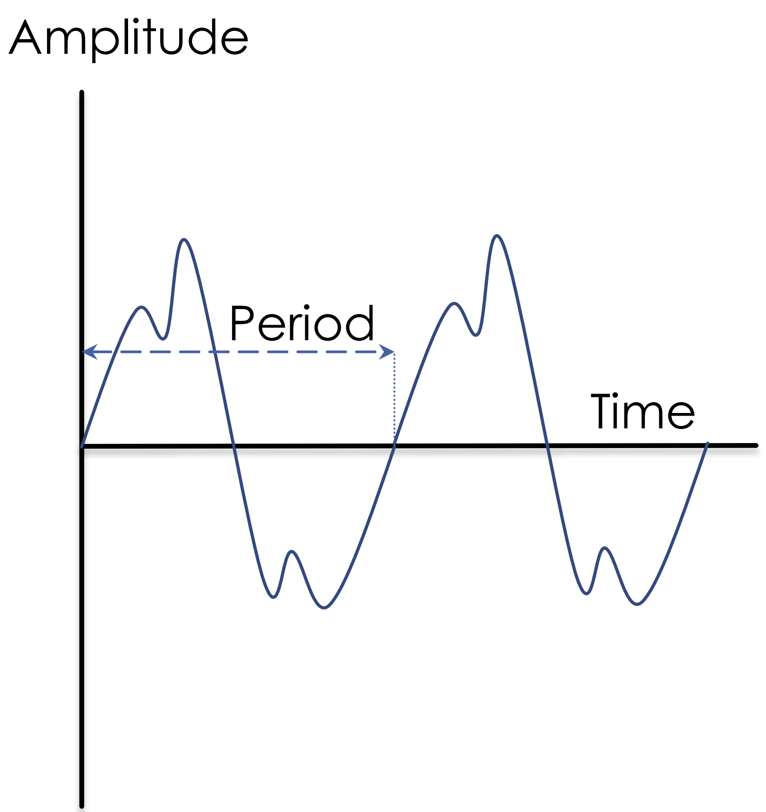
- A vector space of 2^n dimensions.
- Standard Basis: 2^n Boolean functions. Each basis function outputs 1 only for one particular input string.
- Fourier Basis: 2^n parity functions.

Spectral Analysis of Discrete Periodic Functions – an Overview

1. Periodic functions with some examples
2. Fourier expansion of periodic functions
3. Discrete sampling of continuous periodic functions
4. Analogy Between Boolean Functions and Discrete Periodic Functions

Signals as Periodic Functions

- A periodic function is a function that repeats it's values in regular intervals or periods.
- Example: Trigonometric functions, which repeats over the intervals of 2π radians.
- Prominent real world application: radio wave signals, audio/video signals, etc.



Fourier Expansion of Periodic Functions

Fourier expansion of a periodic function $s(x)$ (with period 2π) is an infinite sum of $sines$ and $cosines$:

$$s(x) = \frac{a_0}{2} + \sum_{j=1}^{\infty} A_j \cos(jx) + \sum_{j=1}^{\infty} B_j \sin(jx),$$

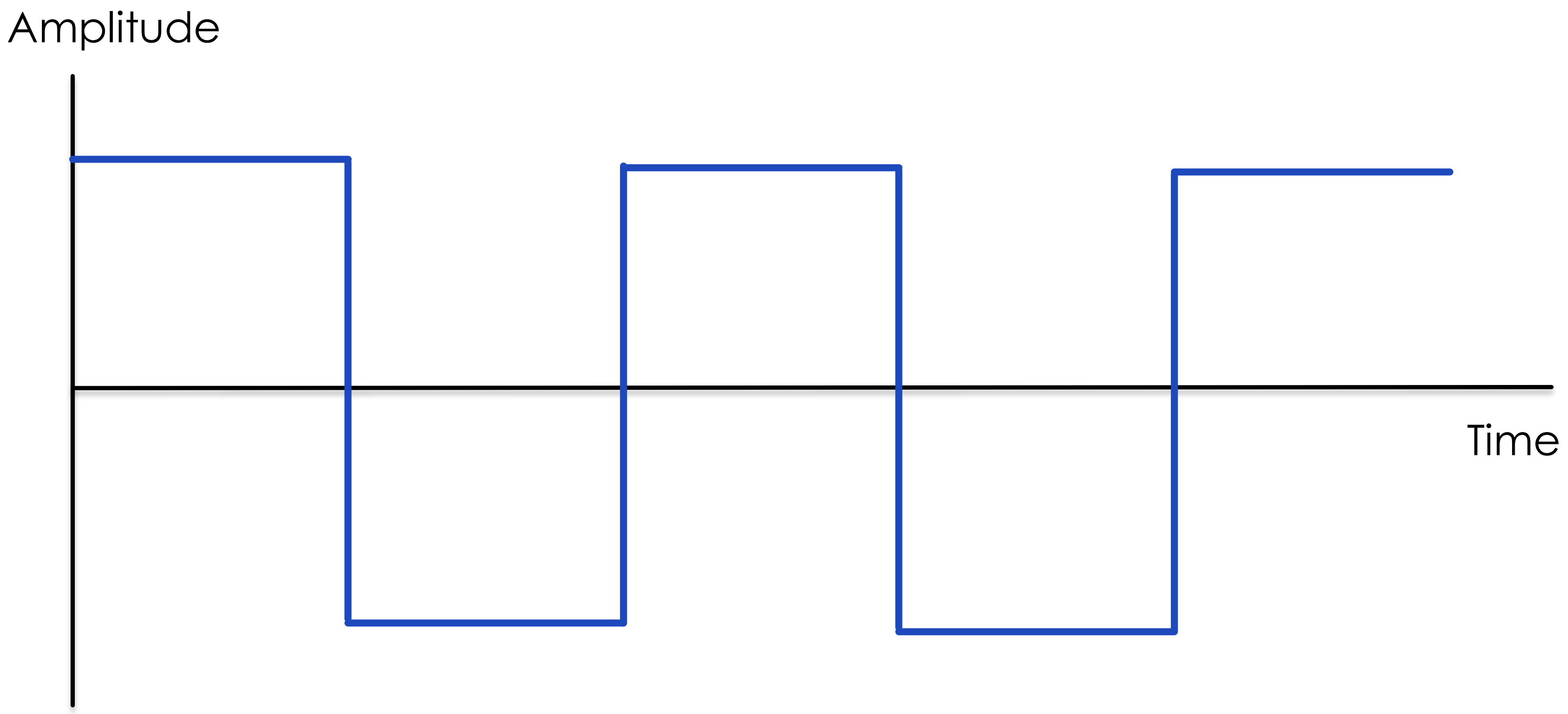
Where $A_j = a_j \sin(\phi_j)$ and $B_j = a_j \cos(\phi_j)$ for suitable ϕ_j values, which can be interpreted as a phase shifts to get an equivalent form:

Fourier Expansion of Periodic Functions

$$s(x) = \frac{a_0}{2} + \sum_{j=1}^{\infty} a_j \sin(jx + \phi_j)$$

Let us now see an example that illustrates the significance of the first few terms.

The Square Wave: A Simple Example



Approximation With a Single Sine wave

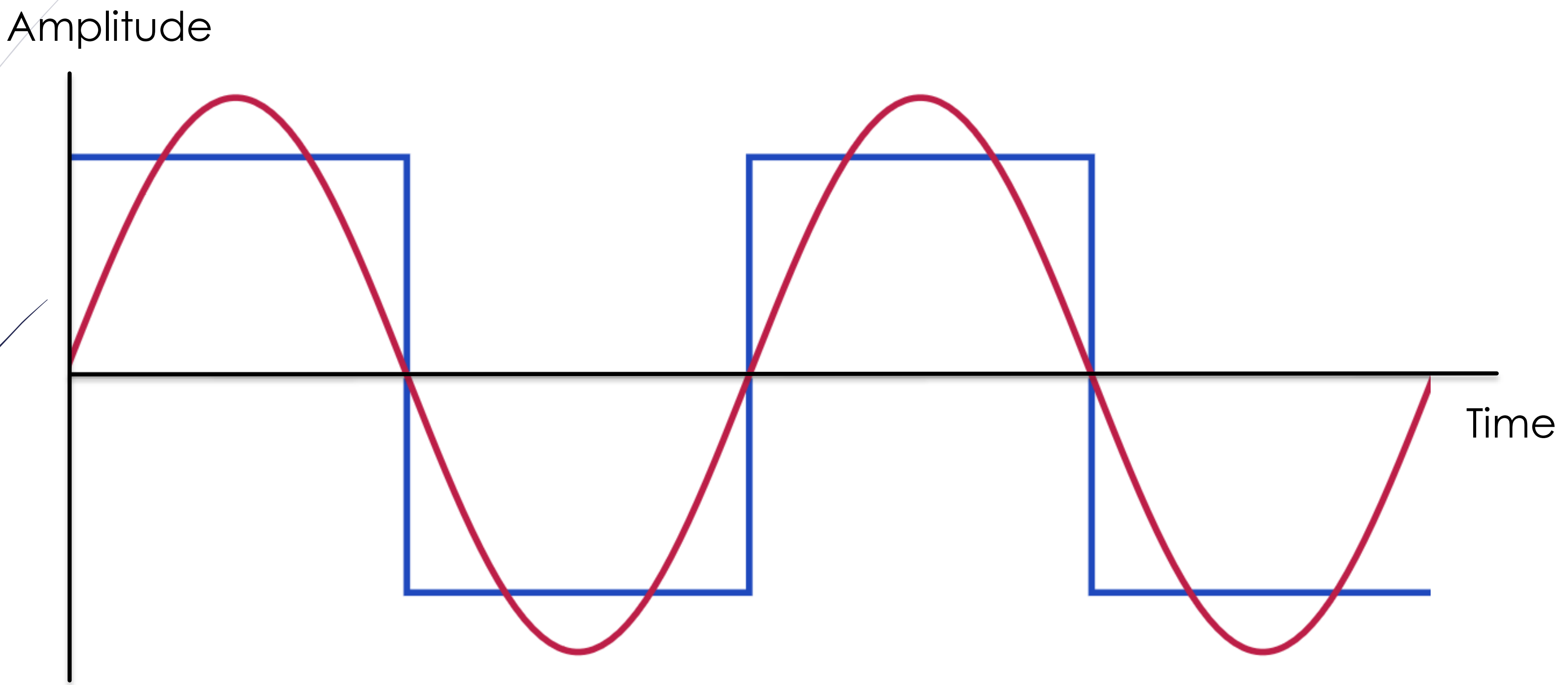


Image and example are courtesy of Wikipedia

Source: https://en.wikipedia.org/wiki/File:Fourier_Series.svg

Including the Second Sine wave

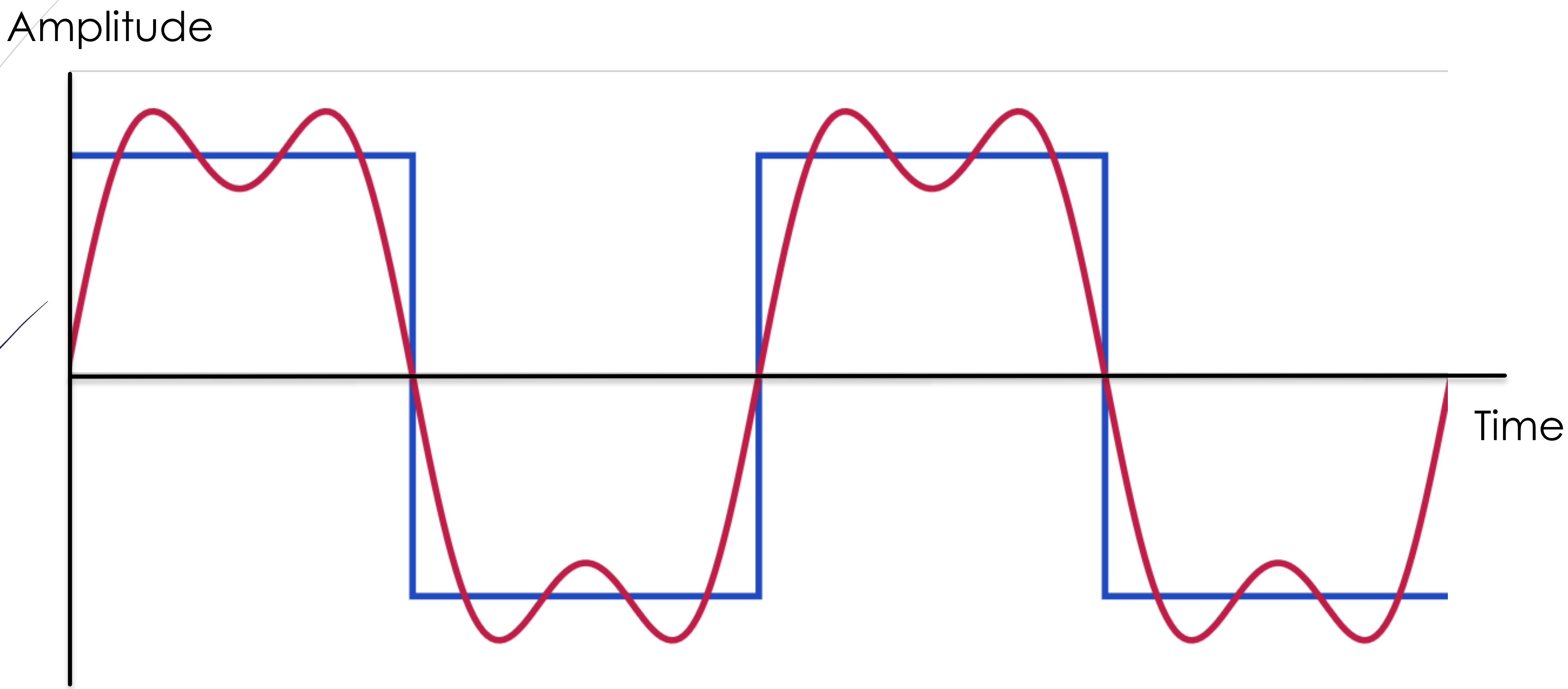


Image and example are courtesy of Wikipedia

Source: https://en.wikipedia.org/wiki/File:Fourier_Series.svg

Including a Few More Sine Waves

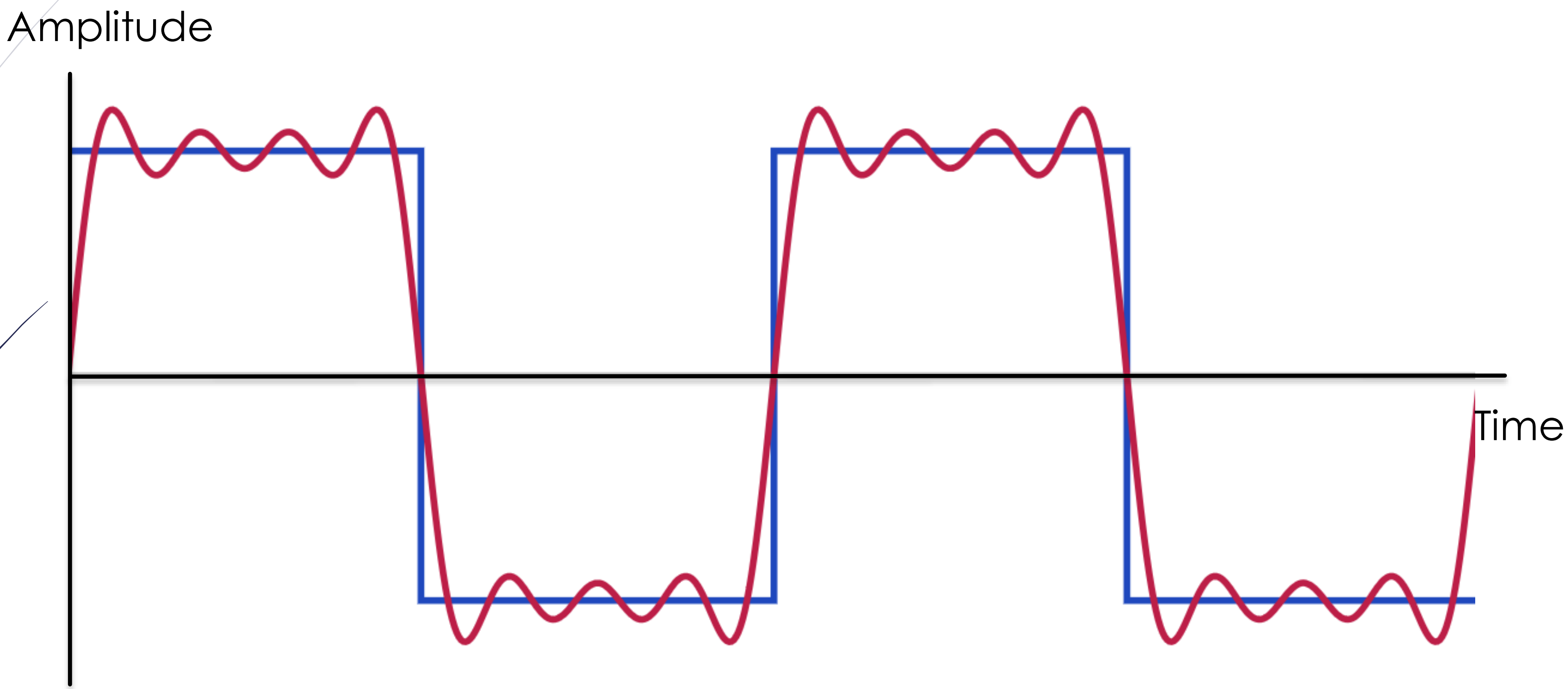


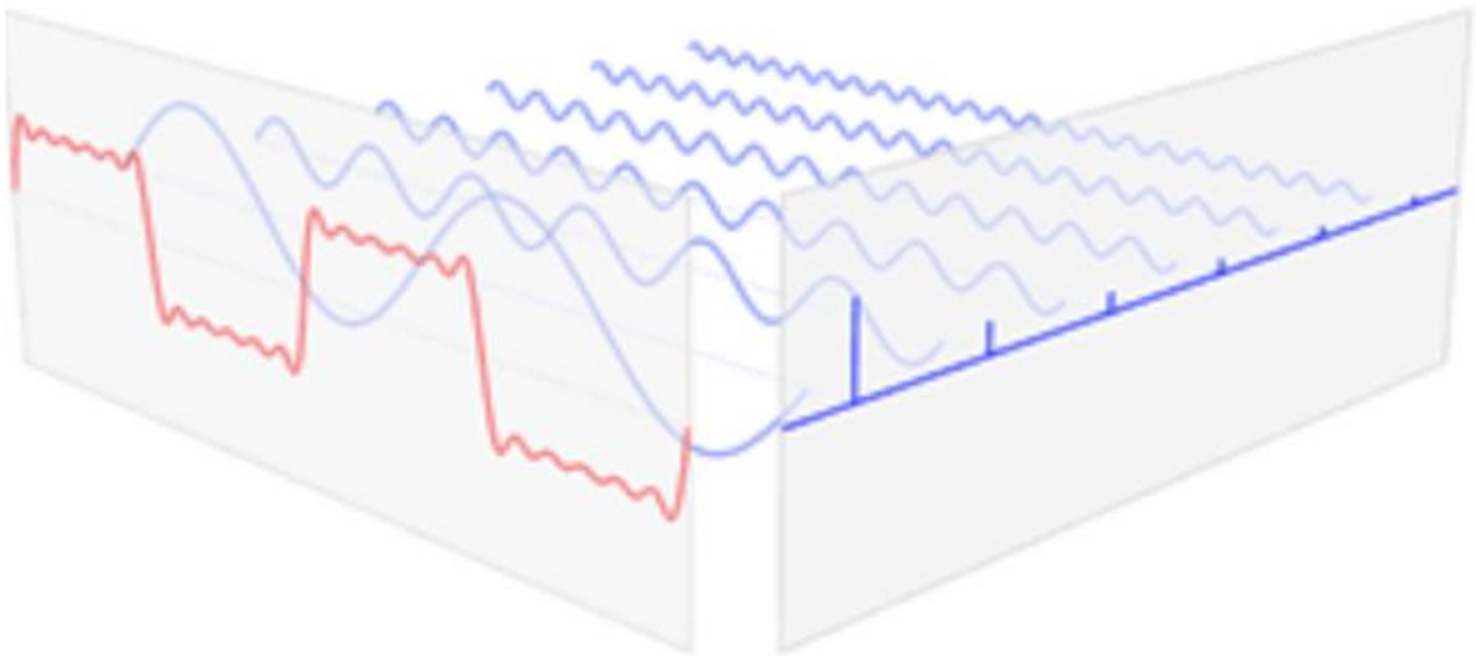
Image and example are courtesy of Wikipedia

Source: https://en.wikipedia.org/wiki/File:Fourier_Series.svg

Extracting the Fourier (Frequency) Spectrum

Source: https://upload.wikimedia.org/wikipedia/commons/2/2b/Fourier_series_and_transform.gif

Extracting the Fourier (Frequency) Spectrum



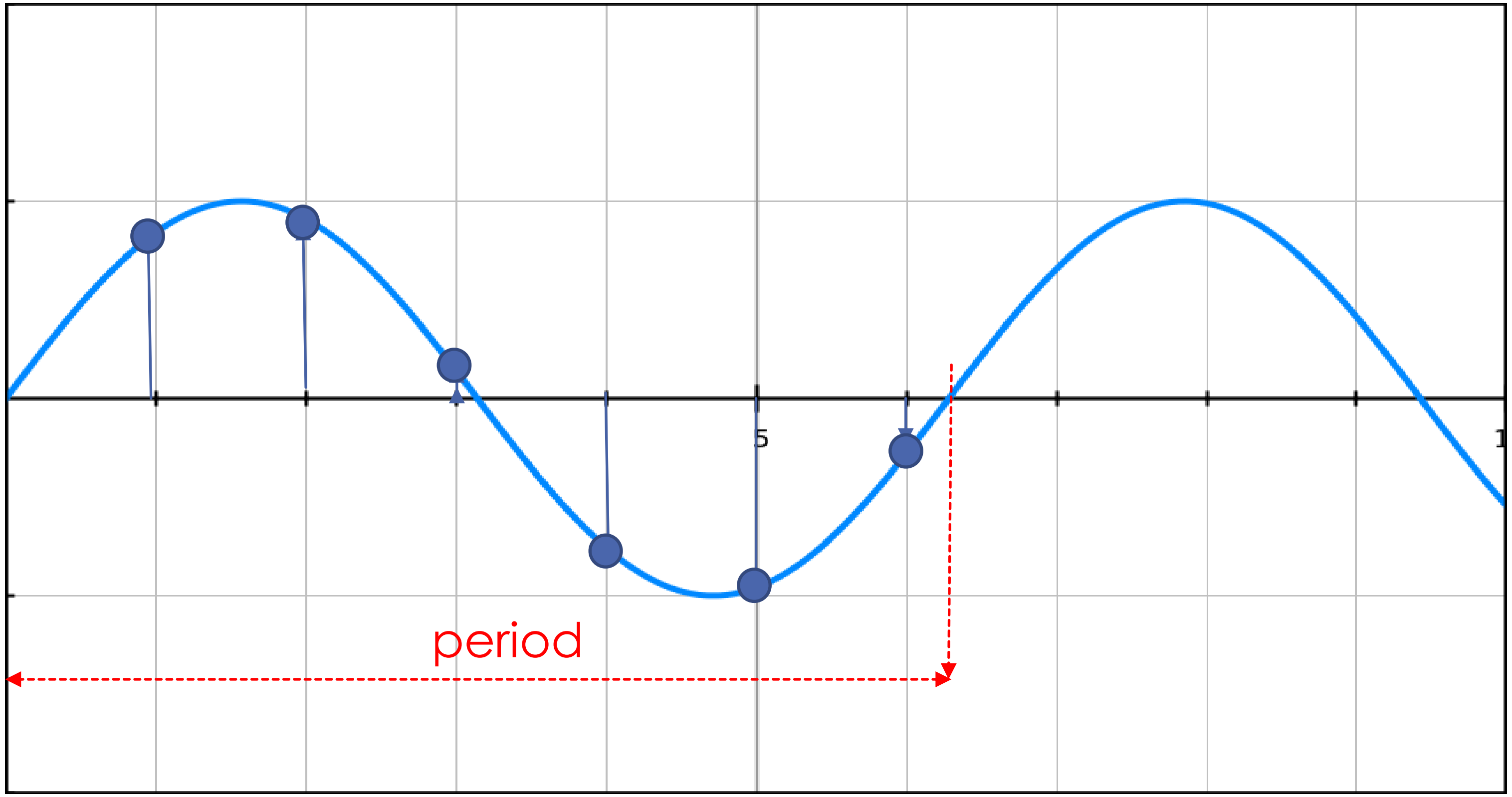
Source: https://upload.wikimedia.org/wikipedia/commons/2/2b/Fourier_series_and_transform.gif

Extracting the Fourier (Frequency) Spectrum

Source: https://upload.wikimedia.org/wikipedia/commons/2/2b/Fourier_series_and_transform.gif

Discrete Sampling

- The continuous time signals can be approximated by sampling.

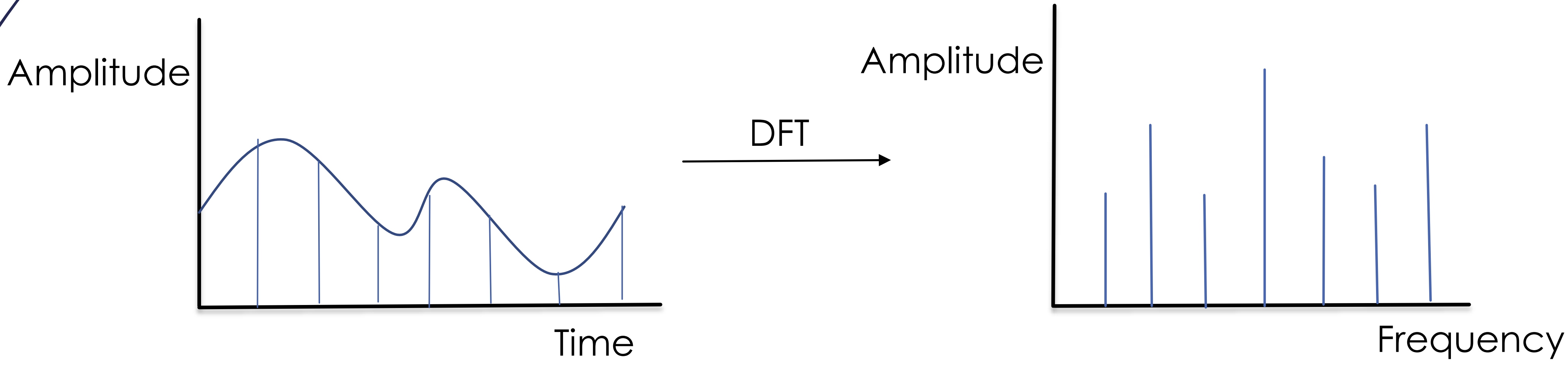


Discrete Sampling

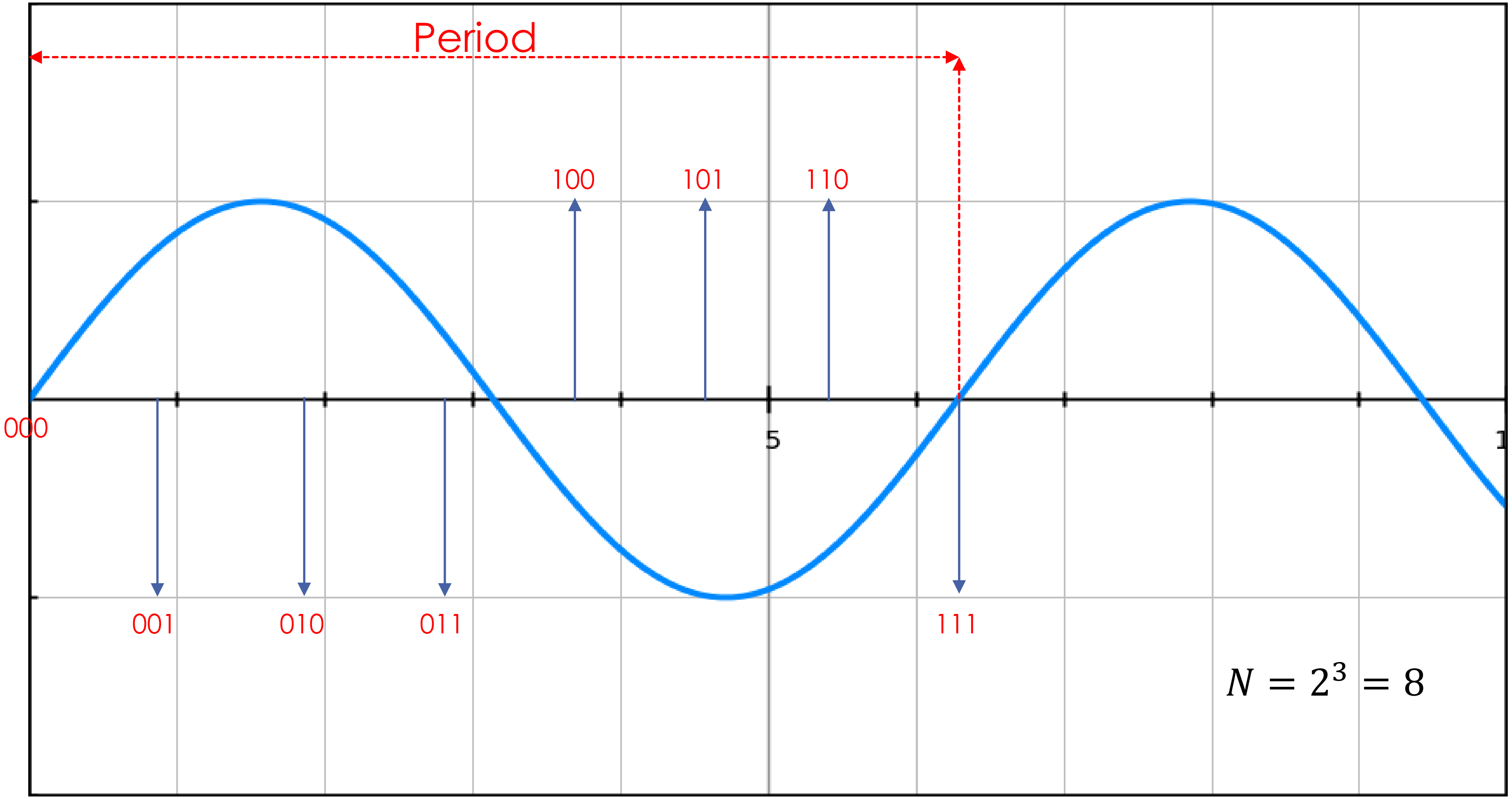
- Sampling theorems like the Nyquist-Shannon Theorem permit a discrete sequence of samples to capture all the information from a continuous time signal whenever it has a finite frequency spectrum.
- The sampled values of continuous signal are now represented by a vector of length N, the number of sampled values in a period.

Discrete Sampling

- The *Fourier transform* of a periodic function of time requires an infinite number of sinusoidal waves of different frequencies.
- *Discrete Fourier transform* (DFT) of a periodic function sampled at discrete points only requires a finite number.



Towards an Analogy Between Boolean Functions and Periodic Functions



Analogy Between Boolean Functions and Discrete Periodic Functions

Boolean Functions	Discrete Periodic Functions
Vector Space of dimension 2^n, where n is the number of bits in input vector.	Vector Space of dimension N, where N is the number of samples values in a period.
Period = 2^n	Period = N
Parity basis.	Sine/Cosine functions.
Spectrum: Vector in the parity basis	Spectrum: coefficients in Fourier expansion
Standard basis is same for both type of functions.	
Most of the functions in practice are sparse in the Fourier basis.	

Introduction to the Fast Fourier Transform Algorithm

Extracting the Fourier Spectrum through Transforms

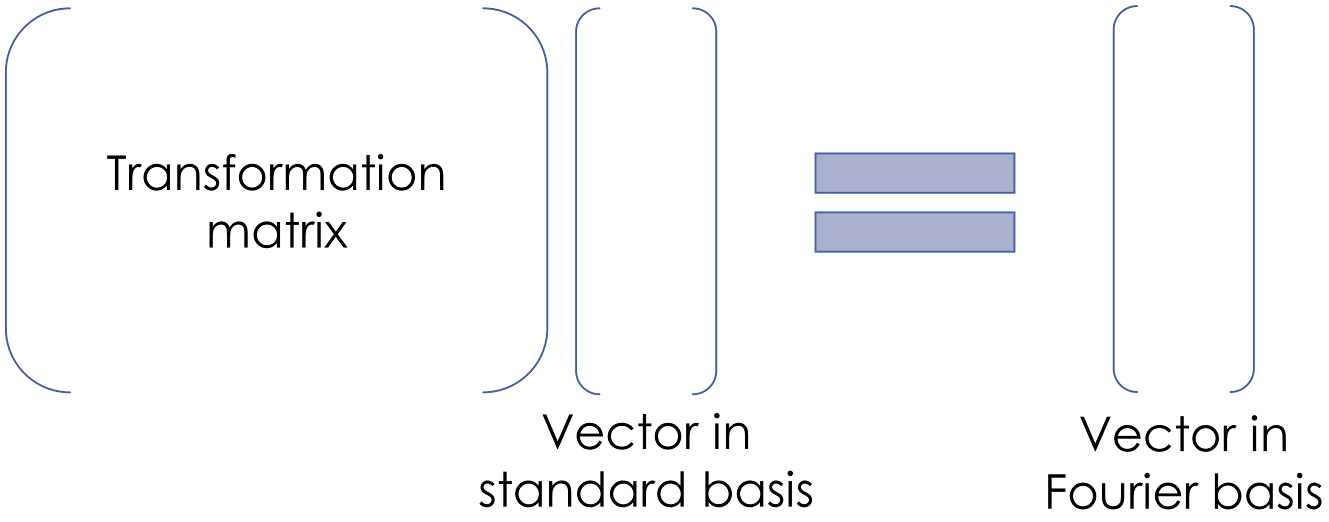
Recall the following:

1. A Fourier spectrum of a vector is its representation in an alternate "Fourier basis."
2. Such a change of basis can be obtained via a suitable transformation matrix multiplication.

A Fourier transform is therefore any algorithm to transform a vector from its standard basis to its Fourier basis. [matrix multiplication = brute force]

A Naïve Discrete Fourier Transform

Being a change of basis, DFT is a matrix multiplication.



But this requires $O(\text{dimension}^2)$ time

Can We Do Better?

- Some nice properties of DFTs:
  1. The terms in each row are in geometric progression.
  2. The entries can be complex.
- We will exploit them to devise an $O(d \log d)$ time *Fast Fourier Transform* (FFT) algorithm – here d is the dimension.
- Divide-and-Conquer paradigm
- The algorithm is more widely applicable as long as the required properties hold.
- In fact, our exposition will be on polynomials!

Three Vector Spaces and their Fourier Bases

Scope of the FFT Algorithm

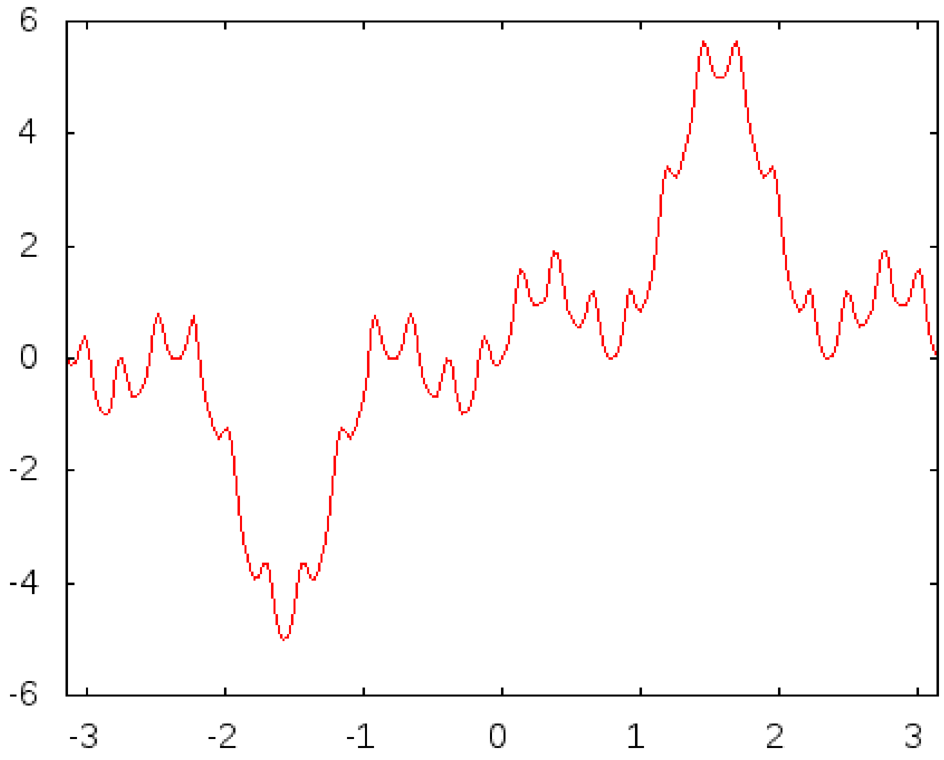
Our Primary Focus

- Boolean functions: Its Fourier basis – we studied earlier – is the set of parity functions.
- (Discrete) periodic functions: Wide application in digital signal processing
  - Fourier basis consists of sines and cosines of different frequencies, hence often called the frequency domain.
- Polynomials: Fundamental and wide applicability with a carefully chosen Fourier basis.

Why do we need the Fourier Basis?

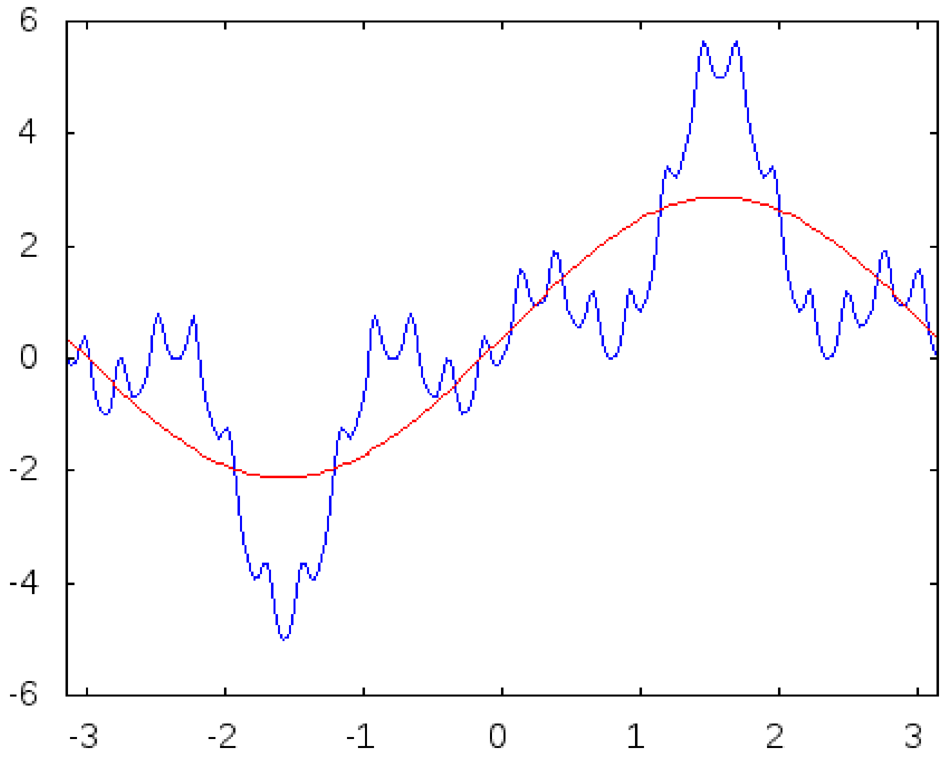
- Vectors in some real world contexts are often "dense" in the standard basis (i.e, require many non-zero terms), but "sparse" in the Fourier basis.
- Example: signals are often sparse in the Fourier basis.
- Let us now walk through one such example (courtesy of Wikipedia.)

An Example of Sparsity in Signals



Source: https://upload.wikimedia.org/wikipedia/commons/0/0b/Fourier_series_for_trig_poly_high_degree.gif

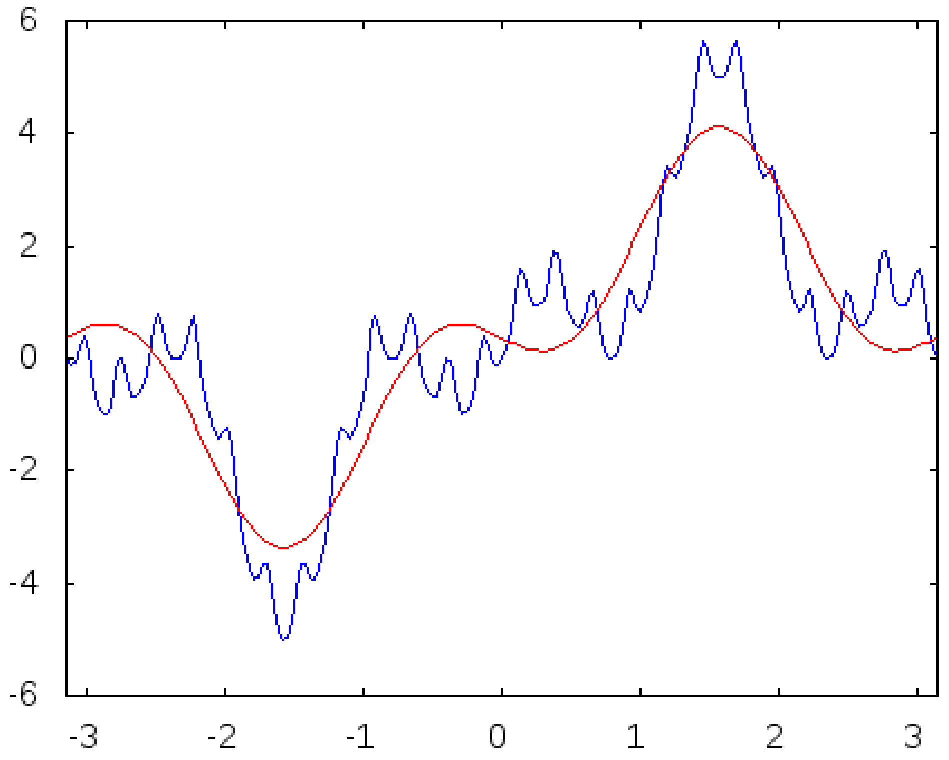
An Example of Sparsity in Signals



With one sine wave

Source: https://upload.wikimedia.org/wikipedia/commons/0/0b/Fourier_series_for_trig_poly_high_degree.gif

An Example of Sparsity in Signals

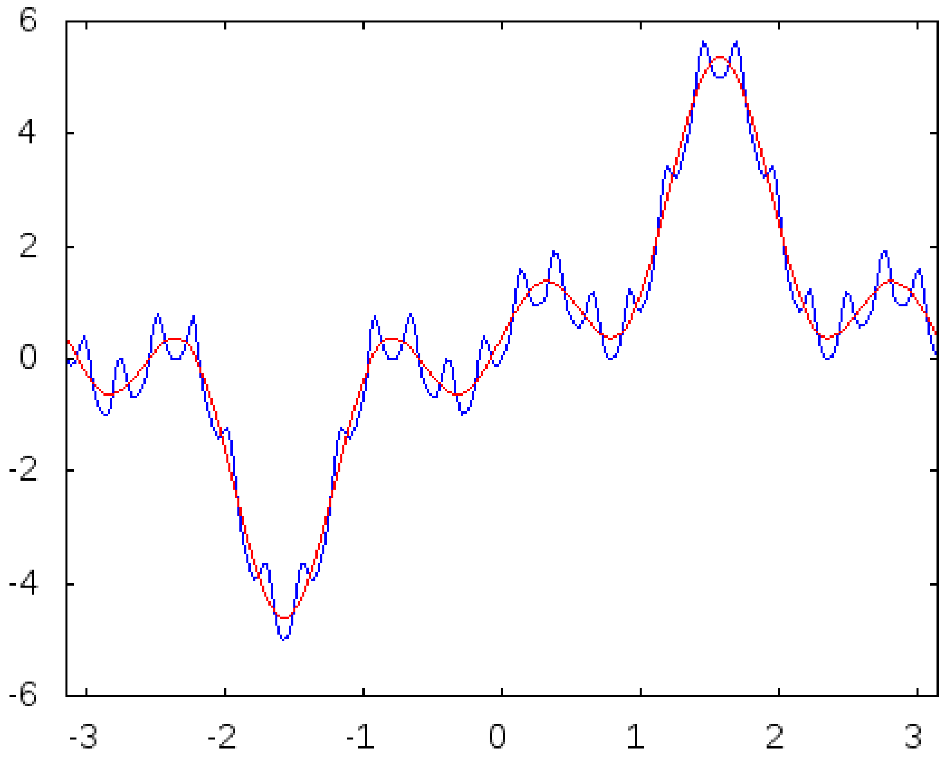


With four sine waves

Source: https://upload.wikimedia.org/wikipedia/commons/0/0b/Fourier_series_for_trig_poly_high_degree.gif

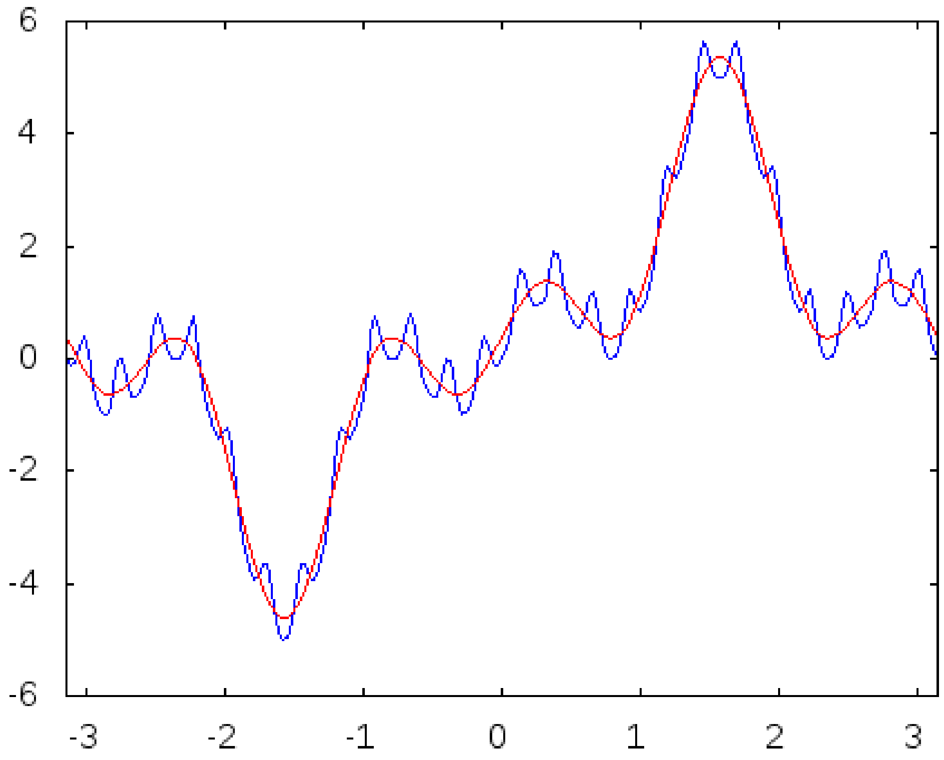
An Example of Sparsity in Signals

With five sine waves



Source: https://upload.wikimedia.org/wikipedia/commons/0/0b/Fourier_series_for_trig_poly_high_degree.gif

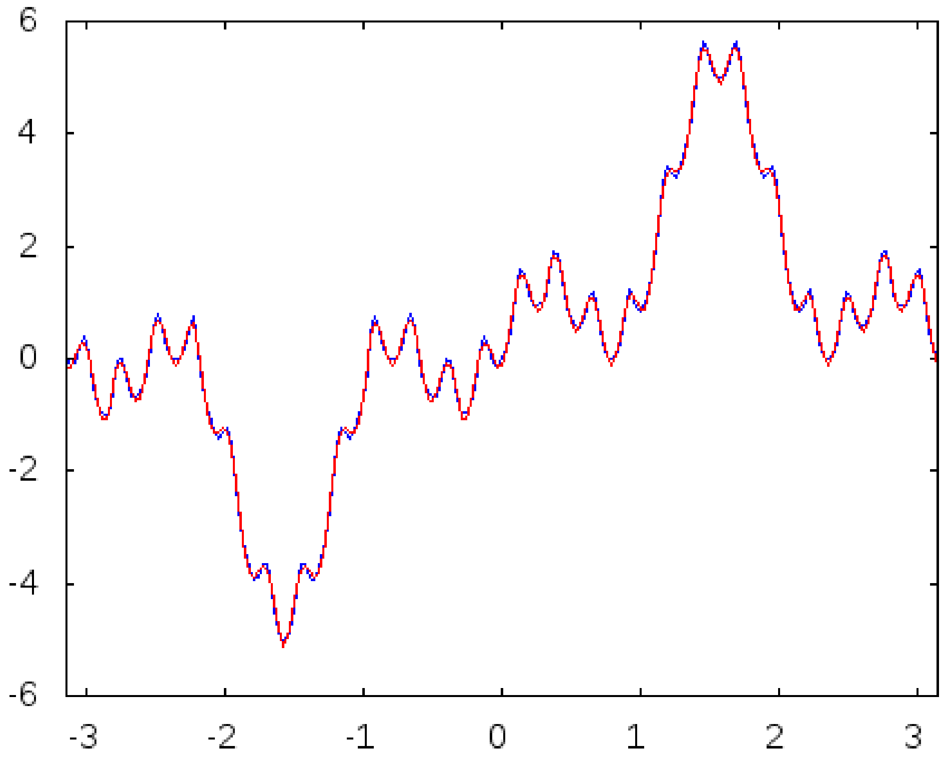
An Example of Sparsity in Signals



With five sine waves

Source: https://upload.wikimedia.org/wikipedia/commons/0/0b/Fourier_series_for_trig_poly_high_degree.gif

An Example of Sparsity in Signals



With 25 sine waves

Source: https://upload.wikimedia.org/wikipedia/commons/0/0b/Fourier_series_for_trig_poly_high_degree.gif

Applications of Fourier Transforms

- The alternative basis may make the vectors more convenient for some operations
- Example: The Fourier transform of polynomials lead to faster polynomial multiplication.
- We will see this in detail, but first a sketch.
- Recall: naïvely multiplying two polynomials of degree d will require $O(d^2)$ time. But…

Applications of Fourier Transforms

- If the polynomials have been evaluated at several points ($d + 1$ is sufficient as we will see)
- Then, obtaining the product polynomial is simply pointwise multiplication. Only $O(d)$ time!
- Evaluating a polynomial for a carefully chosen set of points (in the complex domain) is, as we shall show, a Fourier transform.

The Road Ahead in a Nutshell

- We will first show that polynomials of degree at most d can be viewed as vectors in a vector space of dimension $d + 1$.
- Evaluating a polynomial in any distinct $d + 1$ points can be viewed – will show – as a change of basis. This basis is called the Lagrange basis.
- We will present the Fast Fourier Transform algorithm as a transformation from the standard basis to a particular Lagrange basis.
- We will then illustrate how to multiply two polynomials in $O(d)$ time. Among other things, we will have to show that returning to the standard basis is also by way of the FFT algorithm.
- We will then briefly discuss how the algorithm can be applied to signals and periodic functions.

Fast Fourier Transformation 2

John Augustine (IIT Madras)

Krishna Palem (RICE University)

Source:
These slides are based on source material from *Algorithms* by *Dasgupta et al* (Published by McGraw-Hill Education.)

Acknowledgement: We thank Parishkrati and Ashutosh Ingole (both at IIT Madras) for collaboration that resulted in these slides.

Vector space of Polynomials

Polynomial Vector Space

- Consider the polynomial:
$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m.$$
- We will usually store it in the computer as a list of coefficients:
$$a = (a_0, a_1, \ldots, a_m).$$
  This representation is also called *coefficient representation*.
- Now consider the set $\boldsymbol{P_d}$ of all such polynomials of degree $m \leq d$. (For reasons that will become clear in due course, we will assume d is a power of 2.)

Exercise : Show that $\boldsymbol{P_d}$ is a vector space of dimension $d + 1$.

This will require you to check if all eight properties of vector space hold. As a hint, we will now present the standard basis.

Monomial Basis or Standard Basis

- The standard basis or the monomial (single variable) basis is given by:

$$\boldsymbol{b}(\boldsymbol{x}) = [\,1, \quad x, \quad x^2, \dots, x^{d-1}, \quad x^d]$$

- Notice that (as required) $p(x)$ can be written as a linear combination of the basis functions in $\boldsymbol{b}(\boldsymbol{x})$. In other words,

$$p(x) = b(x)a^T.$$

Polynomial Interpolation

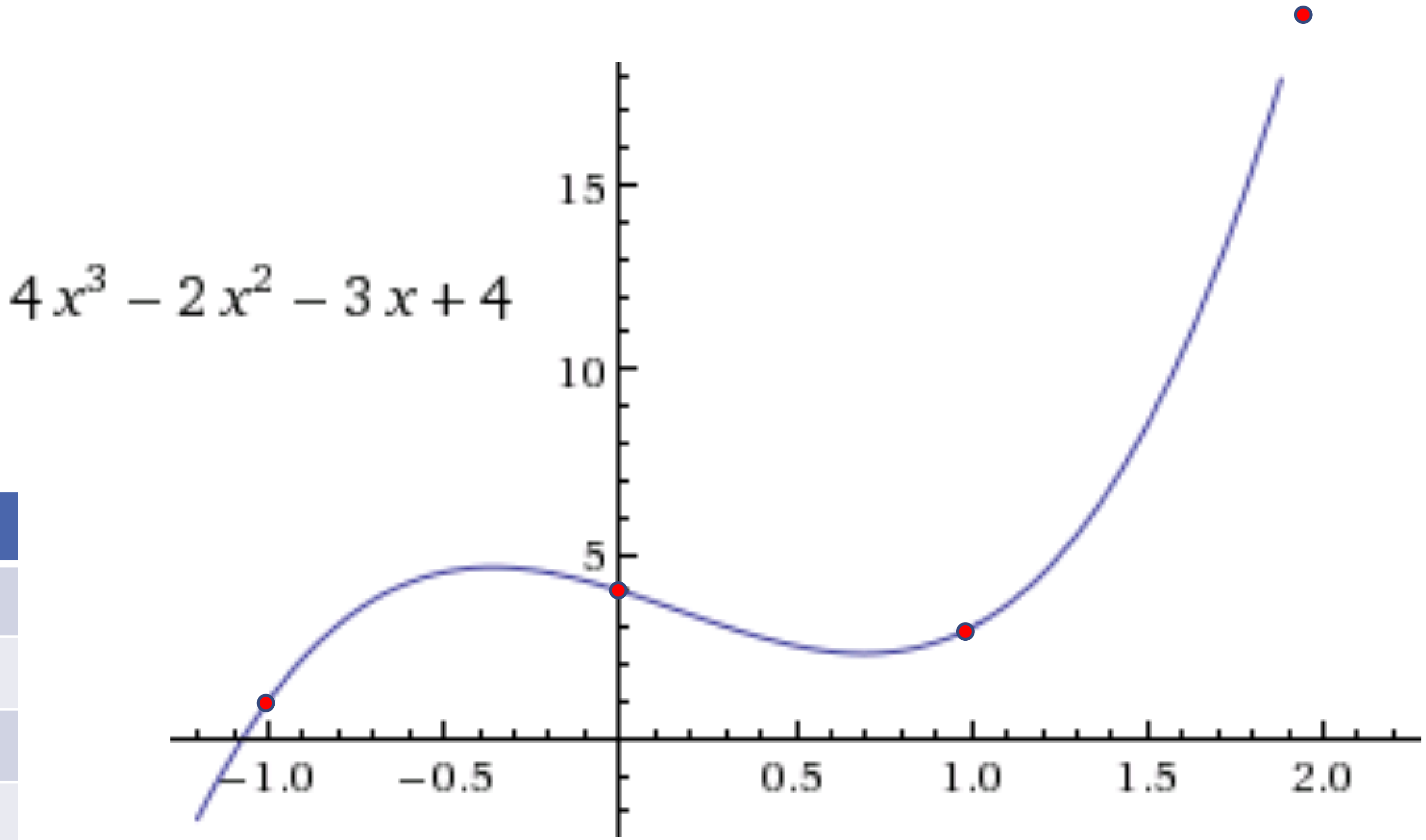
- In the problem of polynomial interpolation, we are given points $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ and required to compute a polynomial $p(x)$ such that

$$p(x_i) = y_i \text{ for } i = 0,1,2, \ldots, d \, .$$

- The points x_i are called *interpolation points*.
- The polynomial $p(x)$ that satisfies these condition is called the *interpolating polynomial*.
  - Does such an interpolating polynomial always exist?
  - Could there be many such polynomials?

An Example

x	y
-1	1
0	4
1	3
2	22



Uniqueness of the Interpolating Polynomial

Theorem : Given $d+1$ distinct points $x_0, x_1, x_2, \ldots, x_d$ and arbitrary values $y_0, y_1, y_2, \ldots, y_d$, there is a unique polynomial $p(x)$ of degree at most d such that

$$p(x_i) = y_i, \text{ for } i = 0,1,\ldots,d.$$

Proof : For contradiction, assume there are two distinct polynomials $p(x)$ and $q(x)$ of degree at most d, such that for all i,

$$p(x_i) = q(x_i) = y_i$$

Consider the polynomial $r(x) = p(x) - q(x)$.

The polynomial $r(x)$ has degree at most d, since it is the difference of two polynomials of degree at most d.

Uniqueness of the Interpolating Polynomial

For $i = 0,1,\ldots,d$

$$r(x_i) = p(x_i) - q(x_i) = 0$$

So $r(x)$ is a polynomial of degree at most d, with $d+1$ roots.

Fundamental Theorem of Algebra : Any nonzero polynomial with degree at most d has at most d roots.

This implies that $r(x)$ must be a zero polynomial.

Hence $p(x) = q(x)$. ■

An Alternate Basis

- Given the Unique Interpolating Polynomial Theorem, we know that a polynomial of degree d can be uniquely represented by its values at $d + 1$ distinct interpolation points.
- Recall: the dimension of polynomial vector space is $d + 1$.
- Can this connection lead to an alternate basis?
- We indeed can, and it's called the Lagrange basis.
- We will now try to understand this basis and its application in fast polynomial interpolation and multiplication.

Lagrange Basis

- The Lagrange basis is defined for a particular list of points $(x_0, x_1, \dots, x_d)$. For each x_i, the corresponding Lagrange polynomial is denoted $l_i(x)$ and has the property that:

$$l_i(x) = \begin{cases} 0 & \text{if } x = x_j \text{ for } j \neq i \\ 1 & \text{if } x = x_i \end{cases}$$

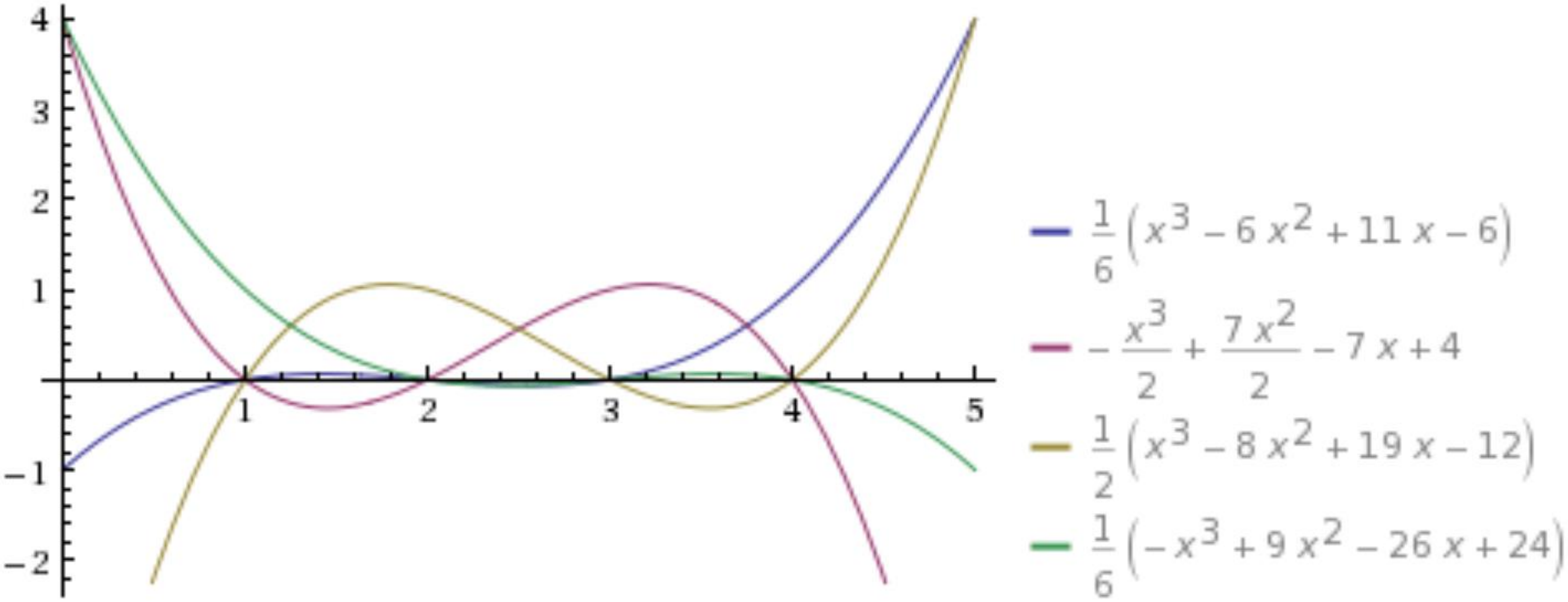
- Notice that the Unique Interpolating Polynomial Theorem assures us that there is exactly one polynomial for which the above property holds.

- The Lagrange basis is the list $\left(l_0(x), l_1(x), \dots, l_d(x)\right)$ of Lagrange polynomials. (We will shortly see how to compute them.)

Exercise: Prove that the Lagrange basis is an orthonormal basis for the vector space of all polynomials of degree at most d.

Hint: Lagrange basis representation of polynomials.

Lagrange Basis Functions for $x = \{1, 2, 3, 4\}$



Lagrange Basis

- Given a set of $d+1$ data points $(x_i, y_i): i = 0,1,\ldots,d$, the unique interpolating polynomial can be represented as

$$\boldsymbol{p}(\boldsymbol{x}) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_d l_d(x) = \sum_{i=0}^{d} y_i l_i(x)$$

where $y = (y_0, y_1, \ldots, y_d)$ are the coordinates of $p(x)$ in the Lagrange basis.

Lagrange Basis

- How do we compute each $l_i(x)$?

$$l_i(x) = \frac{(x-x_0)}{(x_i-x_0)} \cdots \frac{(x-x_{i-1})}{(x_i-x_{i-1})} \frac{(x-x_{i+1})}{(x_i-x_{i+1})} \cdots \frac{(x-x_d)}{(x_i-x_d)}.$$

Notice the missing term

- $l_i(x_i) = \frac{x_i-x_0}{x_i-x_0} \cdots \cdot \frac{x_i-x_j}{x_i-x_j} \cdots \cdot \frac{x_i-x_d}{x_i-x_d} = 1$
- $l_{i\neq j}(x_j) = \frac{x_i-x_0}{x_i-x_0} \cdots \cdot \frac{x_j-x_j}{x_i-x_j} \cdots \cdot \frac{x_i-x_d}{x_i-x_d} = 0$

Monomial basis to Lagrange Basis

- Consider the polynomial of degree d :

$$p(x) = a_0 + a_1 x_1 + \ldots + a_d x^d$$

- To convert p from the monomial to the Lagrange basis, we need to evaluate p at $d+1$ distinct points.

$$\underbrace{\begin{bmatrix} 1 & x_0 & \cdots & x_0^{d-1} & x_0^d \\ 1 & x_1 & \cdots & x_1^{d-1} & x_1^d \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & x_d & \cdots & x_d^{d-1} & x_d^d \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{bmatrix}}_{y}$$

- The coefficients of Lagrange basis are determined by $Va = y$.
- The terms in each row of matrix V are in geometric progression. Such a matrix is called *Vandermonde matrix* and is known to be invertible --- a useful property that will come in handy very soon.

Point Value Representation

- The representation of a polynomial, $p(x)$ of degree d, by $d+1$ point-value pairs $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ such that all x_i are distinct, is called *point value representation* of $\boldsymbol{p(x)}$.
- Notice that the point value representation of polynomial $\boldsymbol{p(x)}$ is equivalent to the coordinates of $\boldsymbol{p(x)}$ in the Lagrange basis defined by $(x_0, x_1, \ldots, x_d)$.

Polynomial Multiplication

Polynomial Multiplication

- The product of two degree m polynomials is a polynomial of degree $2m$.
- Consider two polynomials $A(x)$ and $B(x)$, where
  - $A(x) = a_0 + a_1 x + \ldots + a_m x^m$
  - $B(x) = b_0 + b_1 x + \ldots + b_m x^m$
- Let $C(x) = A(x)B(x)$.
- The coefficients of $C(x)$ can be represented as

$$c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

- Computing c_k from this formula takes $\Theta(k)$ steps, and finding all $2m + 1$ coefficients would therefore require $\Theta(m^2)$ time.
- Can we multiply faster than this?

Point Value Representation

- We can specify a degree m polynomial $A(x)$ by either one of the following :
  - Its coefficients $a_{0,} a_1, \ldots, a_m$ [Monomial Basis]
  - The values $A(x_0), A(x_1), \ldots., A(x_m)$ [Lagrange Basis]
- The product $C(x)$ has degree $2m$, so it is completely determined by any $2m + 1$ points.
- The value of $C(x)$ at any given point z is $A(z) \times B(z)$.
- Thus when A and B are given in point value representation, the point value representation of C can be computed in time that is linear in m.

Polynomial Multiplication

- Typically, A and B are given input in coefficient representation.
- To translate input from coefficient to value representation, we need to evaluate the polynomial at chosen points.
- After multiplying in the value representation we need to translate back to coefficient representation.

$$\begin{array}{c} \text{Coefficient representation} \\ a_0, a_1, \ldots., a_m \end{array} \underset{\text{Interpolation}}{\overset{\text{Evaluation}}{\rightleftarrows}} \begin{array}{c} \text{Value representation} \\ A(x_0), A(x_1), \ldots., A(x_m) \end{array}$$

Polynomial Multiplication

- **Input** : Coefficients of polynomial, $A(x)$ and $B(x)$, of degree m.
- **Output** : Their product $C = AB$

1) **Selection** : Pick some points $x_0, x_1, \ldots, x_{d-1}$, where $d \geq 2m + 1$.
2) **Evaluation** : Compute $A(x_0), A(x_1), \ldots, A(x_{d-1})$ and $B(x_0), B(x_1), \ldots, B(x_{d-1})$
3) **Multiplication** : Compute $C(x_k) = A(x_k)B(x_k)$ for all $k = 0,1,2, \ldots, d - 1$
4) **Interpolation:** Recover $C(x) = c_0 + c_1 x + .. + c_{2m} x^{2m}$

An Example

- **Input**: Consider $(3x^2 + 2x - 1) \times (5x - 2)$
- **Selection**: Lagrange basis for $x = \{0,1,2,3\}$
- **Evaluation**: The vector representation of the two polynomials in the Lagrange basis are:
  - $(-1, 4, 15, 32)$, and
  - $(-2,3,8, 13)$.
- **Multiplication**: Product in Lagrange basis: $(2, 12, 120, 416)$
- **Interpolation**: $15x^3 + 4x^2 - 9x + 2$ (verify!)

Polynomial Multiplication

- Multiplication step in algorithm takes linear time.
- Polynomial evaluation at d distinct points can be represented by $y = Va$.
- Similarly, interpolation step can be represented by $a = V^{-1}y$.
- Evaluation and interpolation steps in algorithm take $O(d^2)$ time.
- We need to do better. How?

Picking the d points for interpolation: A first (incorrect) attempt

- We can choose any set of points at which we evaluate $A(x)$. In other words, we can choose the right Lagrange basis amenable to efficient computation.
- Polynomial $A(x)$ can be represented as

$$A(x) = A_e(x^2) + xA_o(x^2)$$

$A_e(\cdot)$: polynomial with the even numbered coefficients.

$A_o(\cdot)$: polynomial with the odd numbered coefficients.

- Suppose we pick positive-negative pairs, that is

$$\pm x_0, \pm x_1, \ldots, \pm x_{\frac{d}{2}-1}.$$

- Then, notice that $A_e(x_i^2) = A_e((-x_i)^2)$. Similarly, $A_o(x_i^2) = A_o((-x_i)^2)$.

Picking the d points for interpolation: A first (incorrect) attempt

- Given paired points $\pm x_i$, the calculation needed for $A(x_i)$ can be recycled toward computing $A(-x_i)$.

$$A(x_i) = A_e\left(x_i^2\right) + x_i A_o(x_i^2)$$
$$A(-x_i) = A_e\left(x_i^2\right) - x_i A_o(x_i^2)$$

- So evaluating $A(x)$ at d paired points $\pm x_1, \pm x_2, \dots, \pm x_{\frac{d}{2}-1}$ reduces to evaluating $A_e(x)$ and $A_o(x)$ at just $d/2$ points, $x_0^2, x_1^2, \dots, x_{\frac{d}{2}-1}^2$.
- The degree of $A_e(x^2)$ and $A_o(x^2)$ is half of the degree of $A(x)$.

Picking the d points for interpolation: A first (incorrect) attempt

- The original problem of size d is in this way recast as two sub-problems of size $d/2$, followed by some linear time arithmetic.
- If we could continue recursively, we could get a procedure with running time

$$T(d) = 2T\left(\frac{d}{2}\right) + O(d) \in O(d \log d).$$

- The plus-minus trick only works at the top level of the recursion because $A_e(x^2)$ cannot be split further owing to the fact that x^2 is always positive.

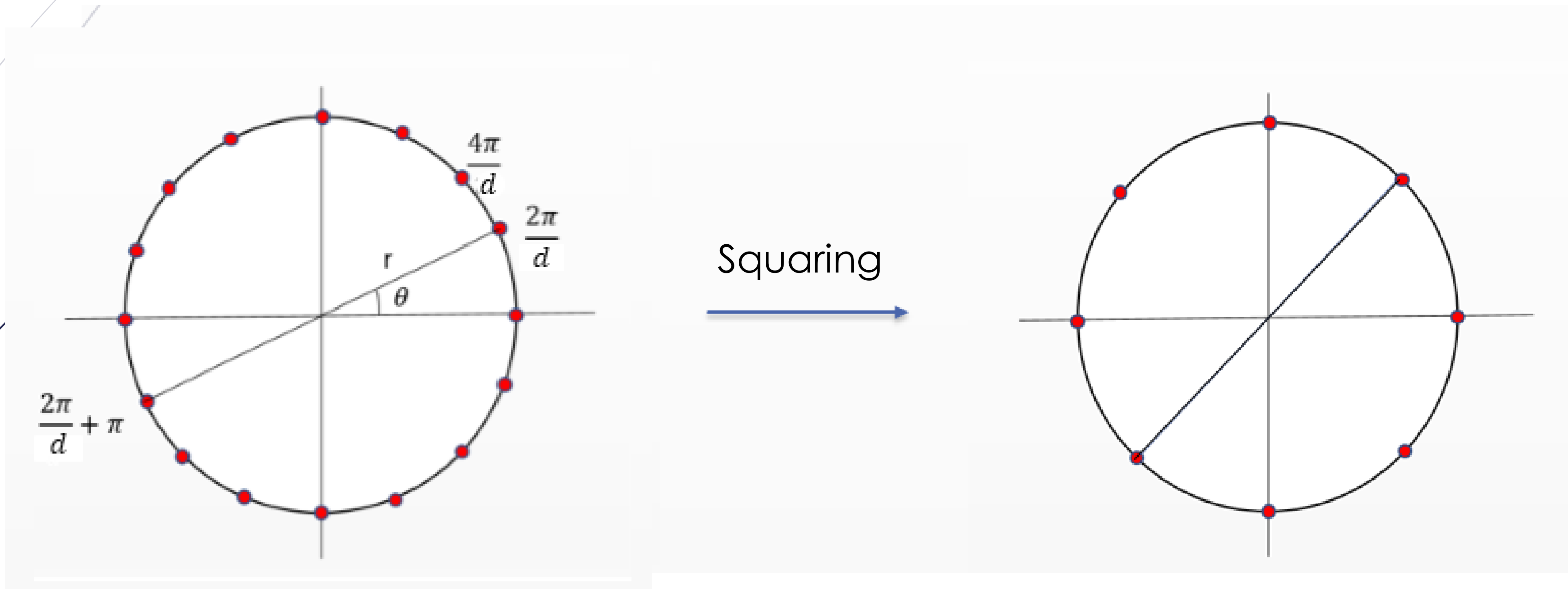
Required Invariant: Plus-Minus Paired After Squaring

$x_0 \quad -x_0 \qquad x_1 \quad -x_1 \qquad \cdots \qquad x_{\{\frac{d}{2}-1\}} \quad -x_{\{\frac{d}{2}-1\}}$

$$\underbrace{x_0^2 \qquad x_1^2 \qquad \cdots \qquad x^2_{\{\frac{n}{2}-1\}}}_{\boxed{\text{No Longer Plus-Minus Paired}}}$$

Unless… we use complex numbers

Properties of d^{th} complex roots of unity



16^{th} complex roots of unity.
Plus-Minus Paired.

8^{th} complex roots of unity.
Plus-Minus Paired.

Complex Numbers: A Quick Recap

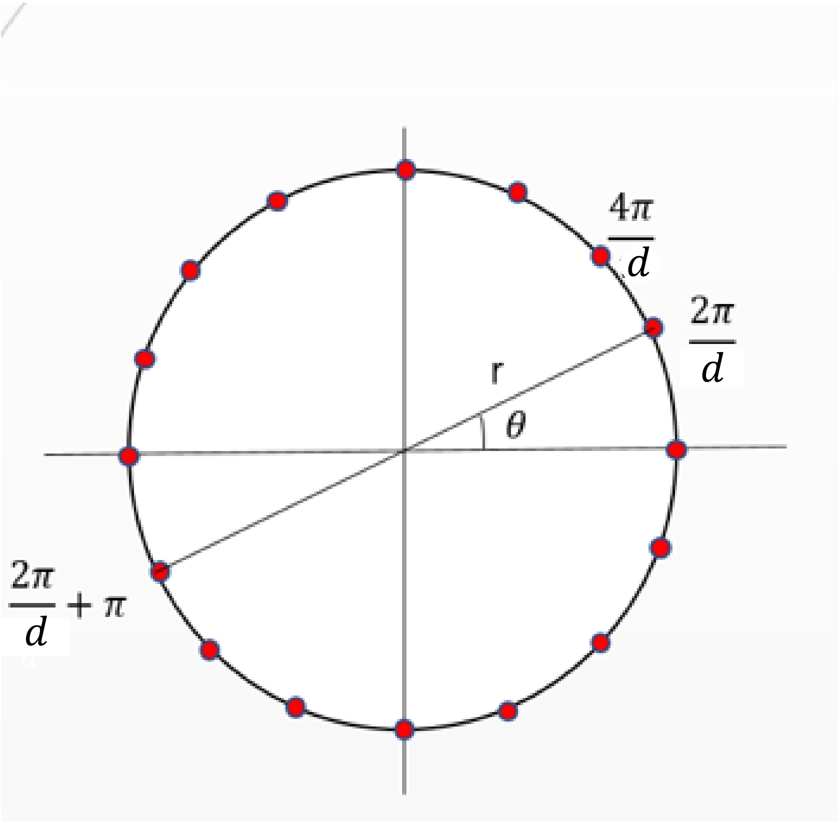
The Complex d^{th} Roots of Unity

- The complex d^{th} roots of unity are the d complex solutions to the equation $z^d = 1$.
- The d^{th} roots of unity are the complex numbers $1, \omega, \omega^2 \ \ldots, \omega^{d-1}$.
- Here $\omega = e^{2\pi i/d}$, the principal d^{th} root of unity and $i = \sqrt{-1}$ is the imaginary unit.
- The d^{th} roots of unity can also be represented as $e^{2\pi ik/d}$ for $k = 0,1,\ldots.,d-1$, where using Euler's formula, we get

$$e^{2\pi ik/d} = \cos\left(\frac{2\pi k}{d}\right) + i\sin\left(\frac{2\pi k}{d}\right)$$

The Complex d^{th} Roots of Unity

- In complex plane $z = a + bi$ is plotted at position (a, b).
- We can rewrite it as $z = r(\cos\theta + i\sin\theta) = re^{i\theta}$.
- Here length $r = \sqrt{a^2 + b^2}$ and angle $\theta \in [0,2\pi)$.
- Solution to the equation $z^d = 1$ are $z = (1, \theta)$, for θ a multiple of $\frac{2\pi}{d}$. (shown here for $d = 16$).



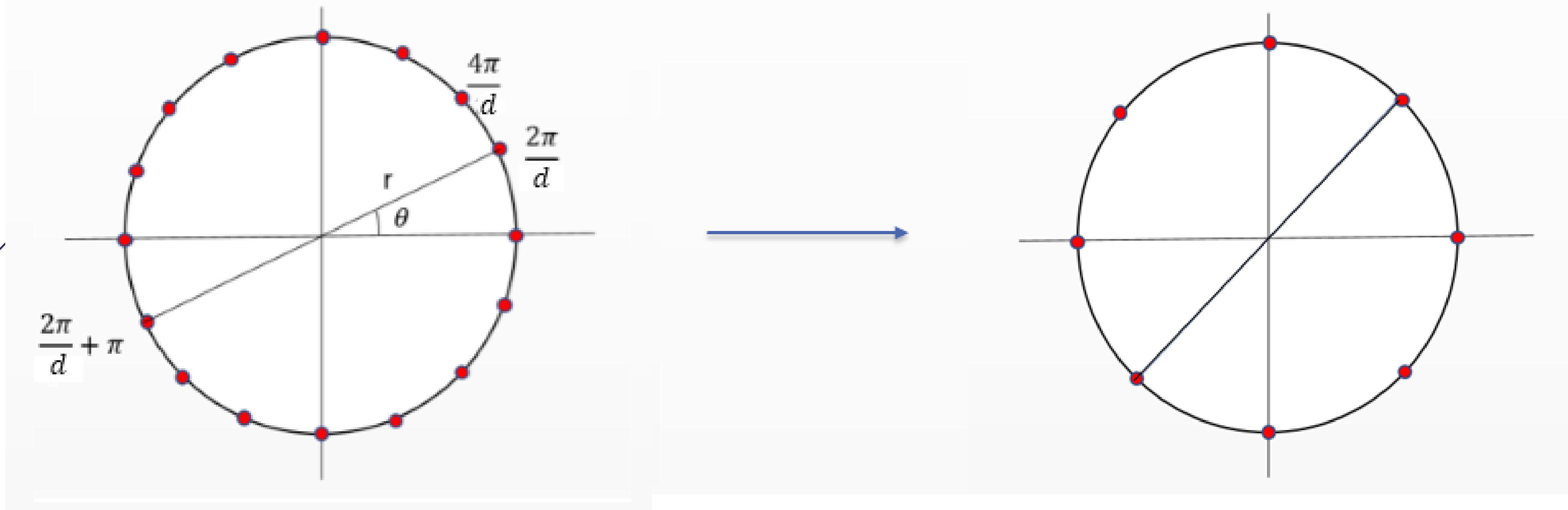
Properties of d^{th} complex roots of unity

- **Property 1 :** If d is even, d^{th} roots of unity are plus-minus paired, i.e.,

$$\omega^{\frac{d}{2}+j} = -\omega^{j}.$$

- **Property 2 :** If $d > 0$ is even, then the square of the complex d^{th} roots of unity are the $d/2$ complex $\left(\frac{d}{2}\right)^{th}$ roots of unity.

Properties of d^{th} complex roots of unity



Squaring 16^{th} complex roots of unity.

Back to Polynomial Evaluation

Evaluation

- We can take advantage of the special properties of the complex roots of unity to perform evaluation in time $O(d \log d)$.
- We evaluate at $1, \omega, \omega^2, \ldots, \omega^{d-1}$.
- Represent polynomial as

$$A(x) = A_e(x^2) + xA_o(x^2)$$

- The problem of evaluating $A(x)$ reduces to evaluating two $d/2$ degree polynomials A_e and A_o at the points $(\omega^0)^2, (\omega^1)^2, \ldots, \left(\omega^{d-1}\right)^2$.
- The points $(\omega^0)^2, (\omega^1)^2, \ldots, \left(\omega^{d-1}\right)^2$ do not contain d distinct values but $d/2$ complex $\left(\frac{d}{2}\right)^{\text{th}}$ roots of unity.

How does this work ?

- For $k = 0, 1, \dots, \frac{d}{2} - 1$
  - $A(\omega^k) = A_e(\omega^{2k}) + \omega^k\, A_o(\omega^{2k})$
  - $A\left(\omega^{k+\frac{d}{2}}\right) = A_e(\omega^{2k+d}) + \omega^{k+\frac{d}{2}}\, A_o(\omega^{2k+d})$

    $= A_e(\omega^{2k}) - \omega^k A_o(\omega^{2k})$

    Since $\omega^{2k+d} = \omega^{2k}$ and $-\omega^k = \omega^{k+\frac{d}{2}}$

The recursive algorithm to evaluate a polynomials at points $1, \omega, \omega^2, \dots, \omega^{d-1}$ is called FFT *algorithm*.

FFT Algorithm (Polynomial Evaluation)

Function $FFT(A, \omega)$

- Input : Coefficient representation of a polynomial $A(x)$ of degree at most $d - 1$, where d is power of 2 and ω is a d^{th} root of unity.
- Output : Value representation $A(\omega^0), \ldots, A(\omega^{d-1})$

1) if $\omega = 1$: return $A(1)$
2) express $A(x)$ in the form $A_e(x^2) + xA_o(x^2)$.
3) call $FFT(A_e, \omega^2)$ to evaluate A_e at even powers of ω
4) call $FFT(A_o, \omega^2)$ to evaluate A_o at even powers of ω
5) for $j = 0$ to $d - 1$
   Compute $A(\omega^j) = A_e(\omega^{2j}) + \omega^j A_o(\omega^{2j})$
6) return $A(\omega^0), \ldots, A(\omega^{d-1})$.

Matrix Representation

- Let us now move towards polynomial interpolation.
- We can represent evaluation step in matrix vector multiplication form

$$\begin{bmatrix} A(\omega^0) \\ A(\omega^1) \\ \vdots \\ A(\omega^{d-1}) \end{bmatrix} = \begin{bmatrix} 1 & \omega^0 & \cdots & \omega^0 \\ 1 & \omega^1 & \cdots & \omega^{d-1} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \omega^{d-1} & \cdots & \omega^{(d-1)(d-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{bmatrix}$$

- Call the middle matrix M or $M_d(\omega)$, where d denotes that size of M.
- Notice that M is a Vandermonde matrix.

Interpolation

- The matrix M is invertible, since it's a Vandermonde Matrix.
- Recall the relationship $Ma = y \Longrightarrow a = M^{-1}y$.
- We multiply coefficient vector with M, to get the value representation.
- To get the coefficients we need to multiply the value vector y with M^{-1}.
- Exercise: Show that the columns of M are orthogonal and then show that $M_d(\omega)^{-1} = \frac{1}{d} M_d(\omega^{-1})$. Here, $M_d(\omega)$ refers to a $d \times d$ version of matrix M.

Interpolation and the FFT

- We can modify the FFT algorithm to perform interpolation in time $O(d \log d\,)$.
- $\langle values \rangle = FFT(\langle coefficients \rangle, \omega)$
- $\langle coefficients \rangle = \frac{1}{d} FFT(\langle values \rangle, \omega^{-1})$

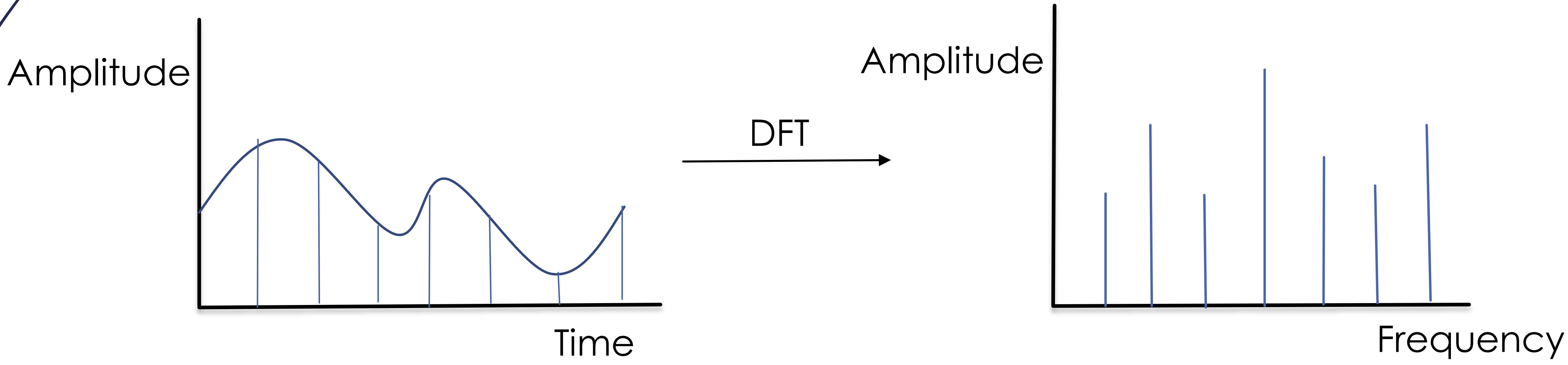
Fourier Spectra, Transforms, Algorithm Redux

- Fourier Spectrum
  - We view it as an alternate Fourier basis.
  - Fourier Transform: changing to the Fourier basis.
- Sparsity and big-data.
- Polynomial Multiplication.
- Fast Fourier Transform: efficient ($O(d \log d)$ time) divide-and-conquer algorithm.

FFT Applied to Discrete Periodic Functions

Discrete Sampling

- The *Fourier transform* of a periodic function of time requires an infinite number of sinusoidal waves of different frequencies.
- *Discrete Fourier transform* (DFT) of a periodic function sampled at discrete points only requires a finite number.



Discrete Fourier Transformation

- Recall that a function can be represented as a vector.
- Given the value of a function at d uniformly sampled data points in time domain by a vector

$$a = (a_0, a_1, a_2, \dots, a_{d-1})$$

- Discrete Fourier Transformation of vector a is given by vector $c = (c_0, c_1, c_2, \dots, c_{d-1})$, where

$$c_k = \Sigma_{j=0}^{d-1} a_j e^{-i\frac{2\pi}{d}jk}$$

- Let $\omega = e^{-i\frac{2\pi}{d}}$, which is a d^{th} root of unity.

- The equation can be rewritten as

$$c_k = \Sigma_{j=0}^{d-1} a_j \omega^{jk}$$

DFT

- DFT can be written as a matrix times a vector as $c = Ma$. which, for $\omega = e^{-\frac{i2\pi}{d}}$ is

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{d-1} \end{bmatrix} = \begin{bmatrix} \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \dots & \omega^{d-1} \\ \vdots & \vdots & \vdots & \vdots \\ \omega^0 & \omega^{d-1} & \dots & \omega^{(d-1)(d-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{d-1} \end{bmatrix}$$

- Here a is the original input signal, and c is the DFT of the signal.