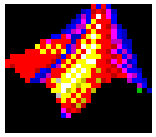
Sines and Cosines

- From the phasor diagram, the difference between sines and cosines is just the point of origin



ptrain1

$$f(t) \Leftrightarrow F(f)$$

$$f(t-T) \Leftrightarrow F(f)e^{-j2\pi fT} \qquad \text{Delay Theorem}$$

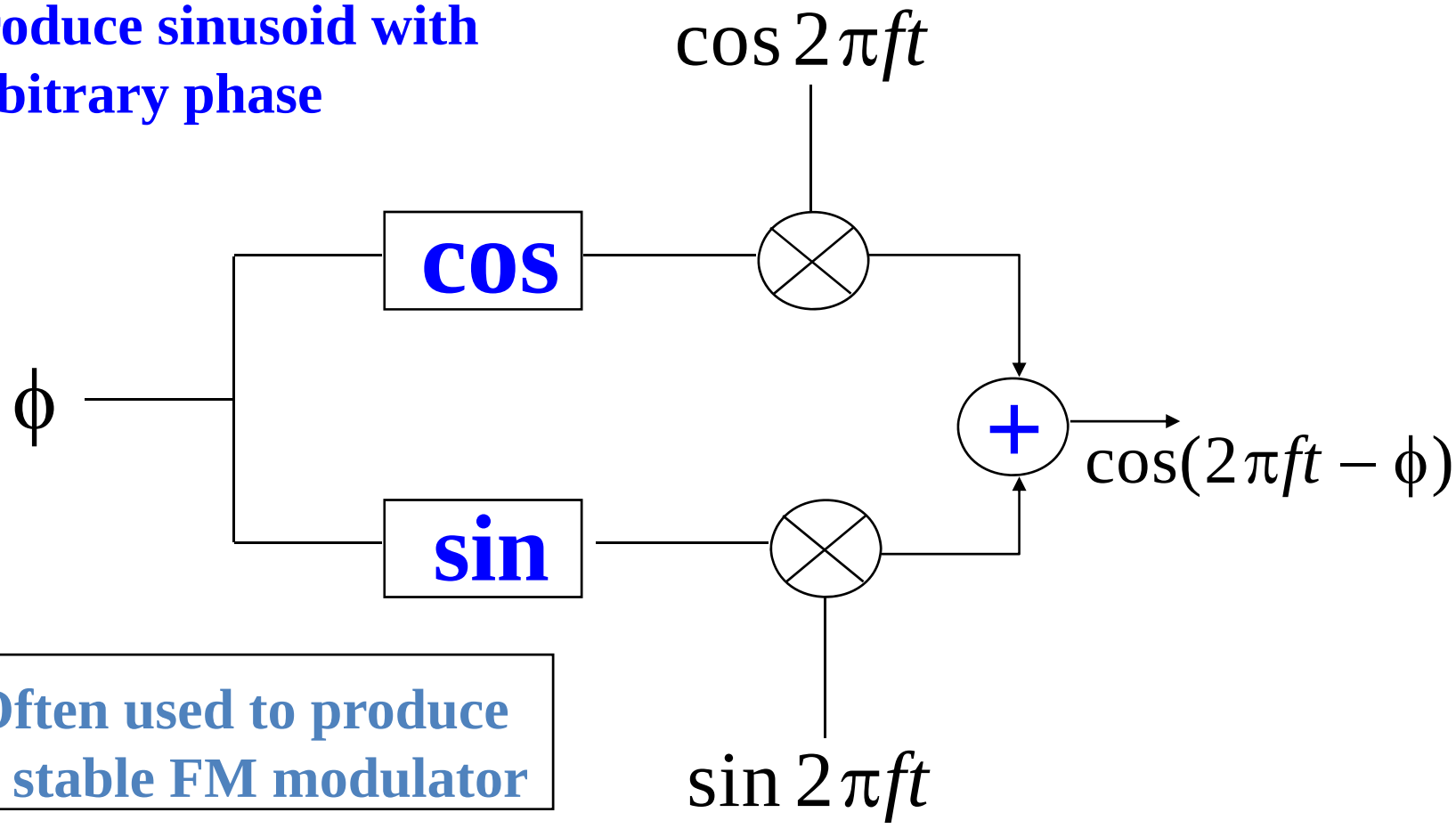
Time Delay $\Leftrightarrow$ Phase Shift

Similarly

$$f(t)e^{-j2\pi f_0 t} \Leftrightarrow F(f-f_0) \qquad \text{Modulation Theorem}$$

Phase Modulation

Produce sinusoid with arbitrary phase

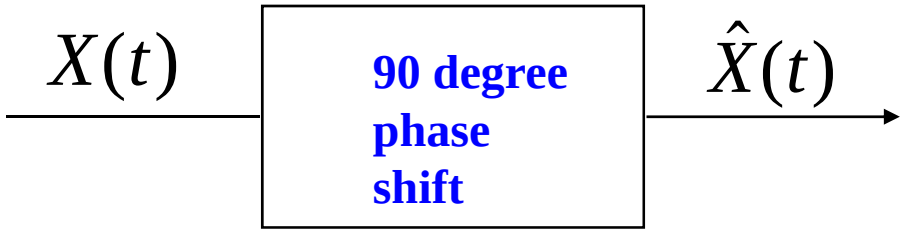


Often used to produce a stable FM modulator

How to produce sine from cosine?

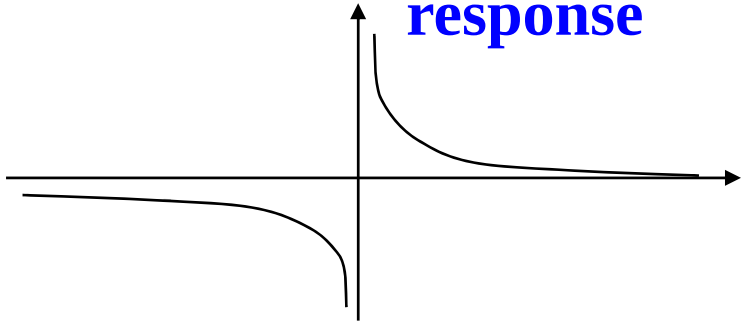
Need all-pass wideband 90 degree phase shifter.

→ Hilbert transformer



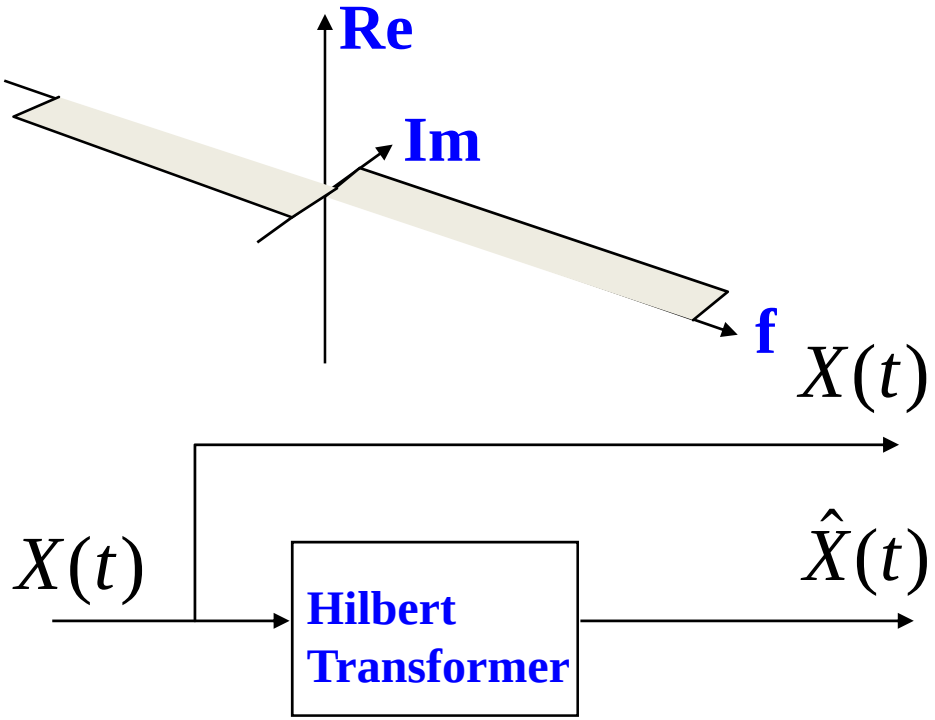
$$\hat{X}(t) = \int_{-\infty}^{+\infty} \frac{X(\tau)}{t-\tau} d\tau = \frac{1}{t} * X(t)$$

filter impulse response



Hilbert Transformer

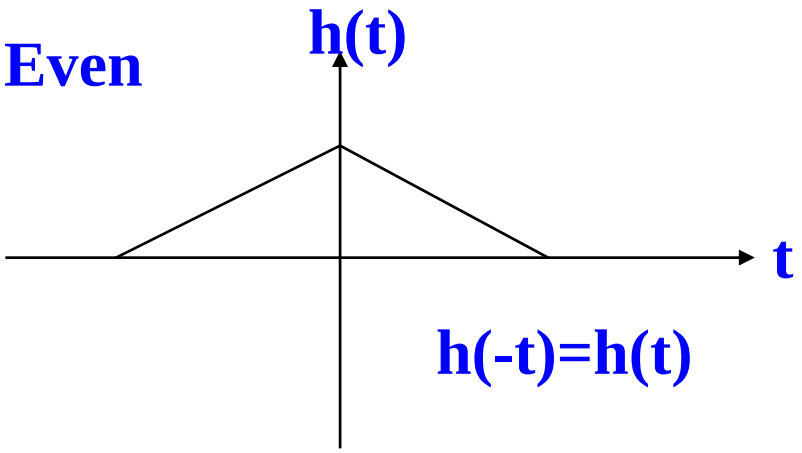
Frequency response: pure imaginary



$X(t) + j\,\hat{X}(t)$ Analytic signal

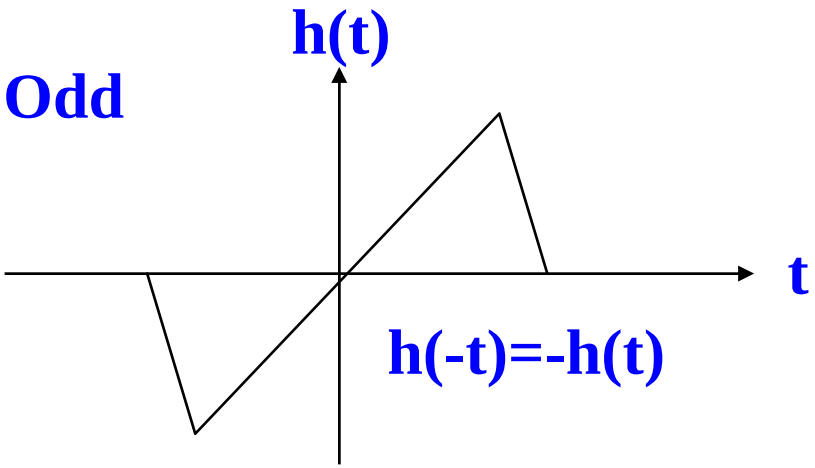
Even and Odd Functions

Even



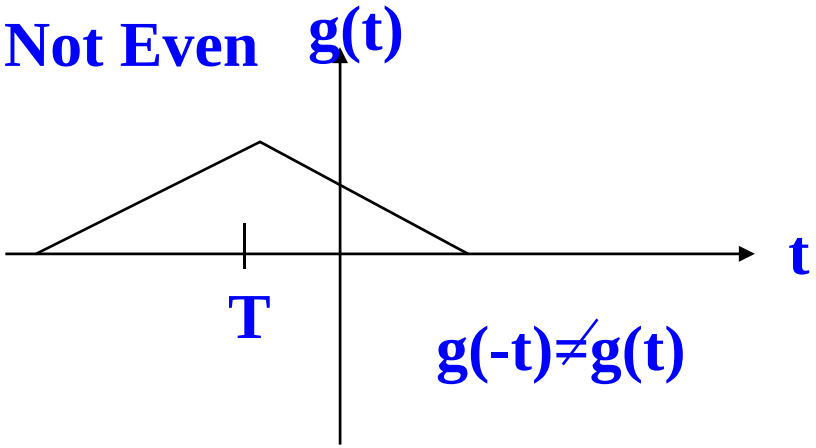
h(-t)=h(t)

Odd



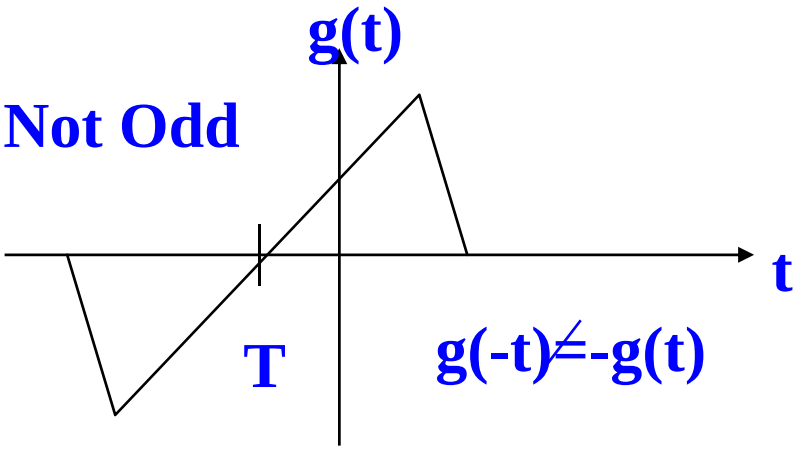
h(-t)=-h(t)

Not Even



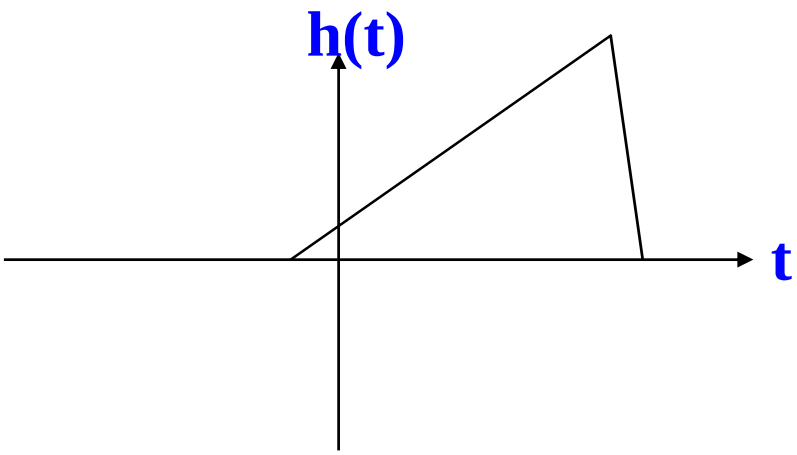
g(-t)≠g(t)

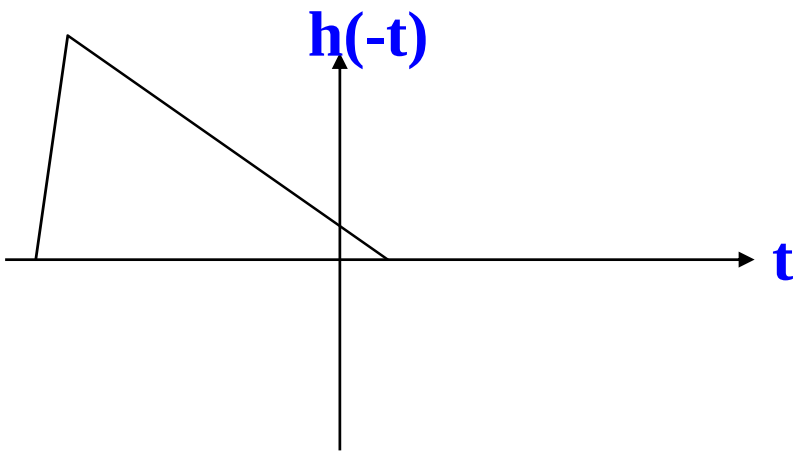
Not Odd

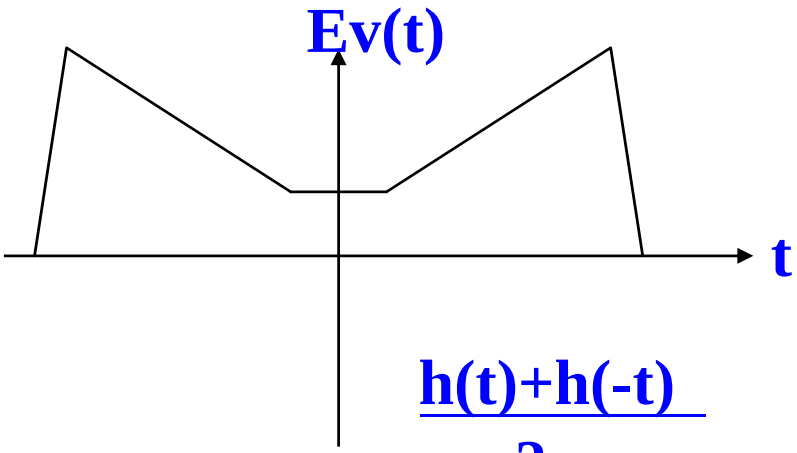


g(-t)≠-g(t)

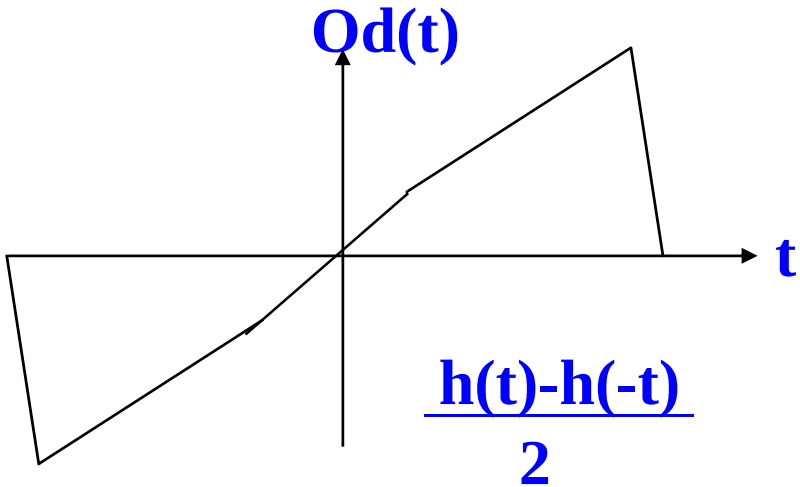
Finding Even and Odd Parts







$$\frac{h(t)+h(-t)}{2}$$



$$\frac{h(t)-h(-t)}{2}$$

Even and Odd Functions

- All functions can be split into even and odd parts in the following manner.

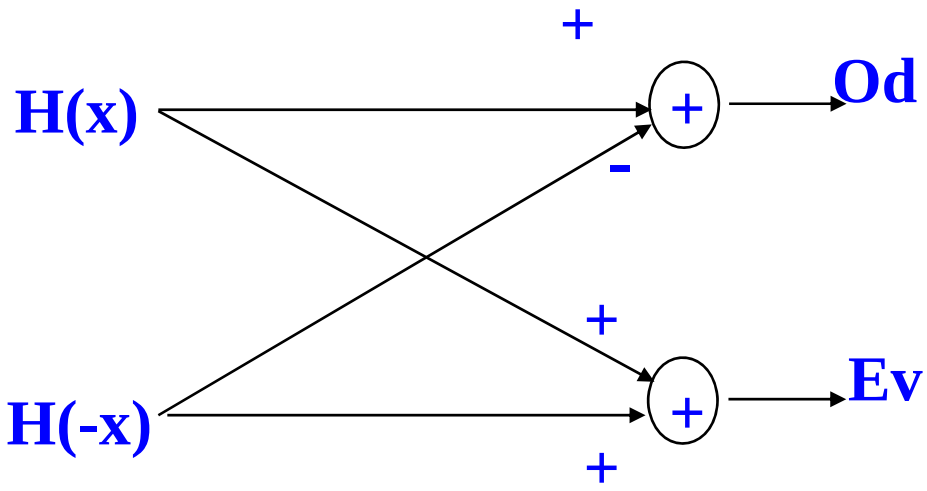
$$Ev(x) = \frac{H(x) + H(-x)}{2}$$

$$Od(x) = \frac{H(x) - H(-x)}{2}$$

Note: for discrete sequences, this decomposition is a 2 point DFT

Butterfly

- A 2-point DFT is often called a Butterfly



Butterfly

This is a real butterfly so you don't get confused

Finding Odd and Even Parts

Example

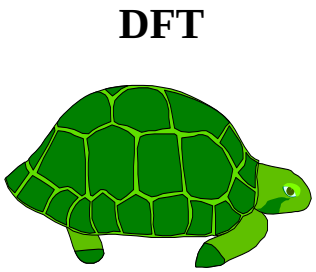
$$h(t) = e^{j\omega t}$$

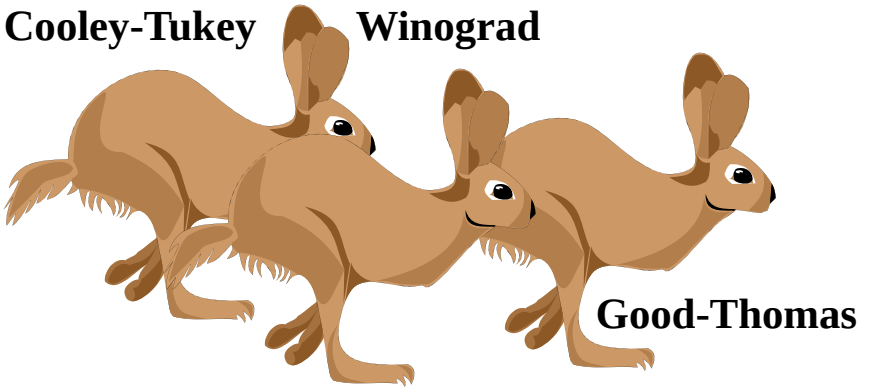
$$Ev(t) = \frac{e^{j\omega t} + e^{-j\omega t}}{2} = \cos(\omega t)$$

$$Od(t) = \frac{e^{j\omega t} - e^{-j\omega t}}{2} = j\sin(\omega t)$$

The Discrete Fourier Transform

Properties and Fast Algorithms





The Discrete Fourier Transform

- Could be called the Discrete-Time Discrete-Frequency FT as opposed to the Discrete-Time Continuous frequency FT.
- Definition of DFT:

$$\text{Transform: } X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}}$$

$$\text{Inverse: } \quad x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi kn}{N}}$$

$$k, n, N \in \mathrm{Z} \quad x, X \in \mathrm{C}$$

The Discrete Fourier Transform

- A sequence of N points will be transformed to another set of N points.

Matlab Notation

Circular right shift

x = [1 0 0 0];
X = fft(x);
X = [1 1 1 1]

$$* e^{-j\frac{2\pi k}{N}\bullet 1}$$

x = [0 1 0 0];
X = fft(x);
X = [1 -j -1 j]

Phase change only
Same magnitude

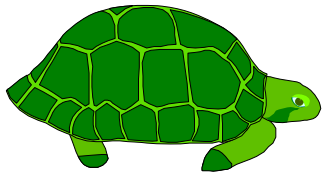
Matrix Form of DFT

Direct implementation of the Definition

$$X = Dx$$

$$\begin{bmatrix} 1 \\ -j \\ -1 \\ j \end{bmatrix} = \begin{matrix} \\ \mathbf{i} \downarrow \end{matrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j\frac{2\pi}{4}} & e^{-j\frac{4\pi}{4}} & e^{-j\frac{6\pi}{4}} \\ 1 & e^{-j\frac{4\pi}{4}} & e^{-j\frac{8\pi}{4}} & e^{-j\frac{12\pi}{4}} \\ 1 & e^{-j\frac{6\pi}{4}} & e^{-j\frac{12\pi}{4}} & e^{-j\frac{18\pi}{4}} \end{bmatrix} \begin{matrix} \mathbf{j} \rightarrow \\ \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Slow!



Requires

N*N complex multiplications

N(N-1) complex additions

$$W_N^{ij} = e^{-j\frac{2\pi}{N}ij}$$

Matrix Form of DFT

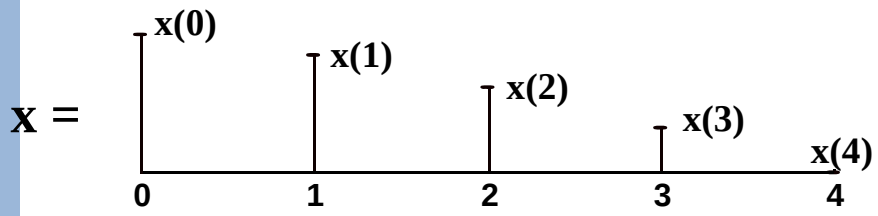
Note: same property applies to rows

$$X = Dx$$

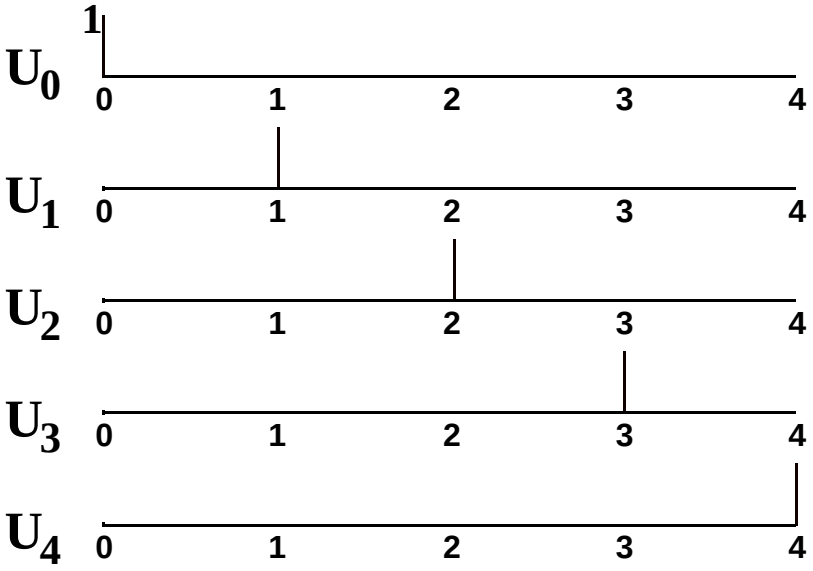
$$\begin{bmatrix} 1 \\ -j \\ -1 \\ j \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j\frac{2\pi}{4}} & e^{-j\frac{4\pi}{4}} & e^{-j\frac{6\pi}{4}} \\ 1 & e^{-j\frac{4\pi}{4}} & e^{-j\frac{8\pi}{4}} & e^{-j\frac{12\pi}{4}} \\ 1 & e^{-j\frac{6\pi}{4}} & e^{-j\frac{12\pi}{4}} & e^{-j\frac{18\pi}{4}} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

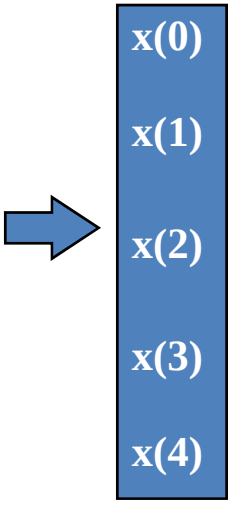
Row 0: constant (all 1's)
Row 1: 1 cycle of complex sinusoid (1 -j -1 j)
Row 2: 2 cycles of complex sinusoid (1 -1 1 -1)
Row 3: 3 cycles of complex sinusoid (1 j -1 -j)

Data Set as a Point in N-Dimensional Space



Find Components by Cross-Multiplying and Summing





Ortho-Normal Basis of Unit Translates: Index indicates location of unit translate

$$\sum_n U_l * U_k^* = 1, k = l$$

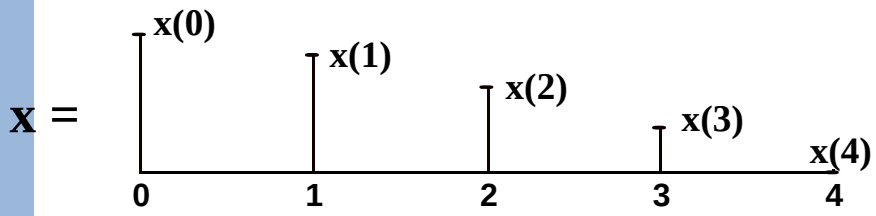
$$\sum_n U_l * U_k^* = 0, k \neq l$$

$$x = x(0)U_0 + x(1)U_1 + \cdots + x(N-1)U_{N-1}$$

Comments

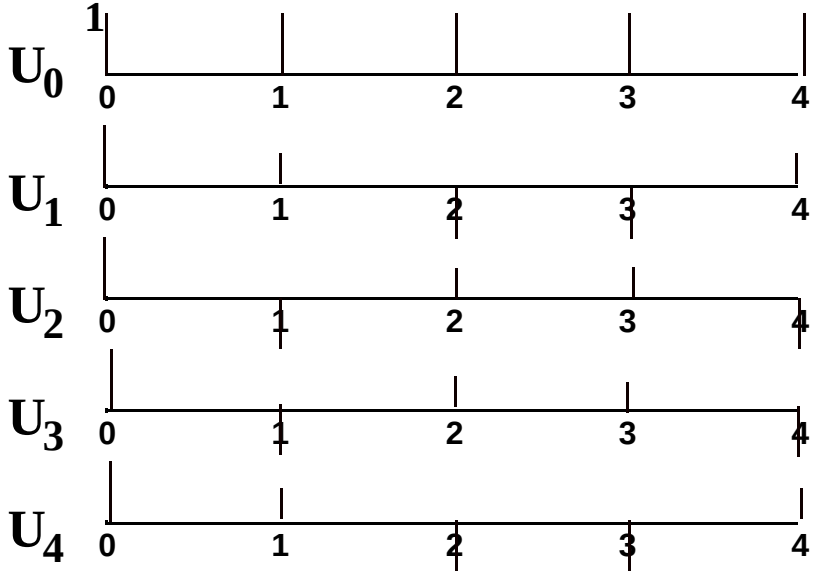
- Finding the coefficients of the unit translates is equivalent to multiplying x by the identity matrix, which is an orthonormal matrix
- In terms of Linear Algebra, multiplication of a vector in N-ary space by an orthonormal matrix is always completely reversible and often corresponds to a rotation or reflection. In this case the rotation is by 0 degrees.
- In other words, the data can be reconstructed exactly from the coefficients of the basis vectors.

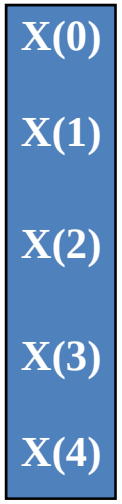
Data Set as a Point in N-Dimensional Space



Real part shown

Find Components by Cross-Multiplying and Summing





Orthogonal Basis of Complex Sinusoids: Index indicates cycles per interval

$$\sum_n U_l * U_k^* = 5, k = l$$

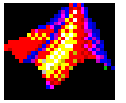
$$\sum_n U_l * U_k^* = 0, k \neq l$$

$$x = X(0)U_0 + X(1)U_1 + \cdots + X(N-1)U_{N-1}$$

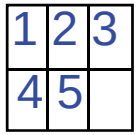
Comments

- **Finding the coefficients of the complex sinusoids is equivalent to multiplying x by an orthonormal matrix.**
- **Once again, this corresponds to a simple rotation or reflection and scaling in N-ary space, and is therefore completely reversible.**
- **In other words, the data can be reconstructed exactly from the coefficients of the basis vectors. That is, the DFT coefficients.**
- **Thus the DFT can be viewed as a simple rotation and scaling of the coordinate system so that one can view the data from a better “angle.”**

DFT Tricks and Tips



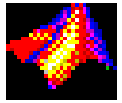
pad.m
wrap.m
zpack.m

- DFT transforms N input points to N output points
- What if we want to transform N points to M points?
  - If M > N use zero padding before DFT
    - 12345 ⇨ 12345000
  - If M < N, wrap up data on circle before DFT
    - 12345 ⇨  ⇨ 573

- What is the effect of zero packing? 123⇨102030
  - The spectrum is replicated for each zero inserted.
  - That is the Nyquist interval is increased and we see the spectral replicas of the discrete-time data explicitly instead of implicitly.

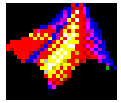
More Tricks and Tips

- Multiplication Free (fast) complex heterodynes
  - Multiplication by $\exp(-j\pi n) = 1, -1, 1, -1, \ldots$.
    - Circularly rotates transform data by n/2 (180 degrees)
    - Equivalent to fftshift() for even DFT lengths
  - Multiplication by $-\exp(-j\pi n) = -1, 1, -1, 1, \ldots$.
    - Circularly rotates transform data by n/2 (180 degrees) and changes phase by 180 degrees (inverts data)
  - These two operations can be used
    - in separation of even and odd samples
    - quadrature mirror filters
    - low pass to high pass transformations



chet1.m

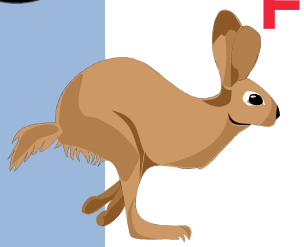
More Tricks and Tips



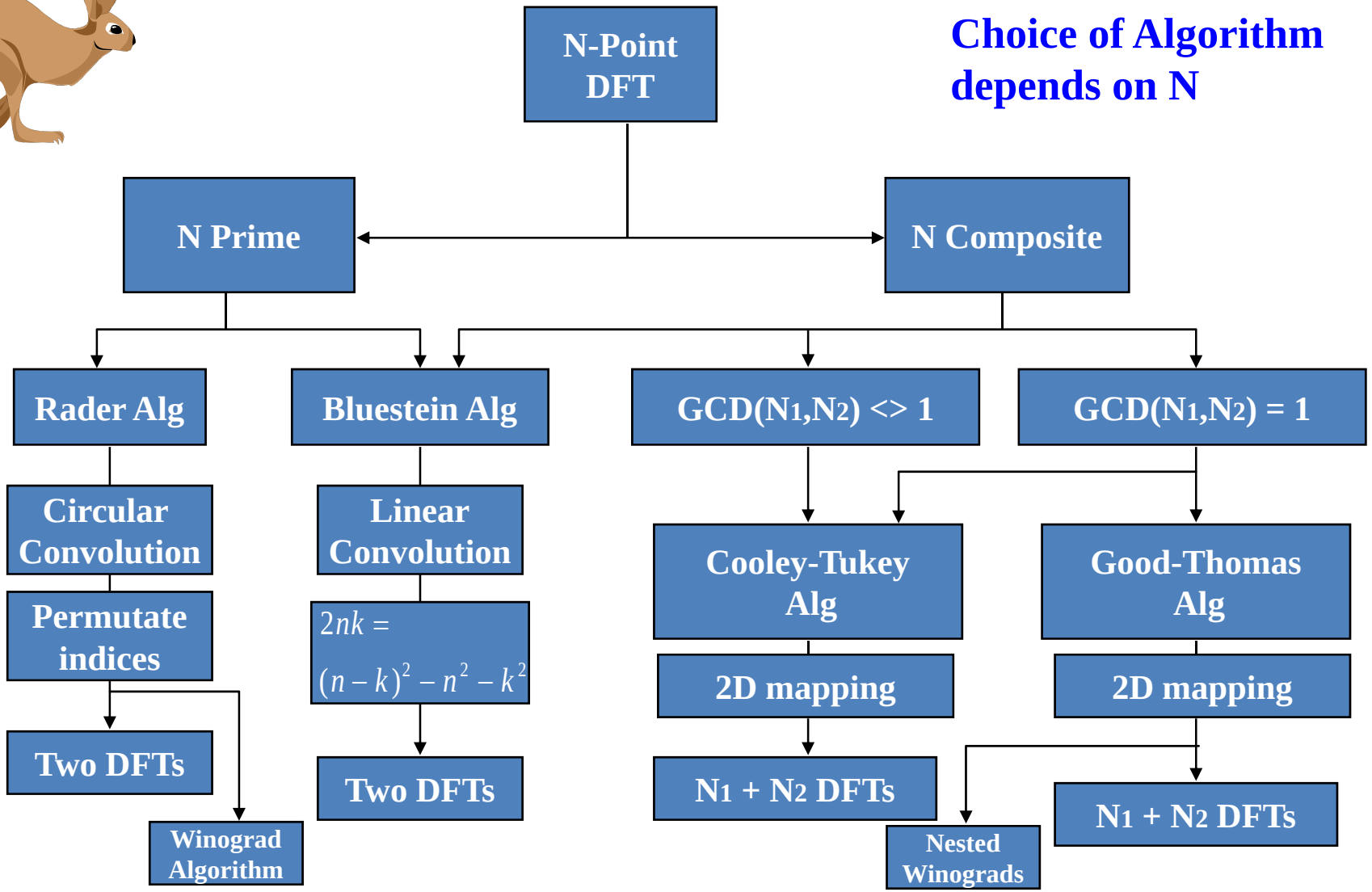
chet2.m

- **Multiplication Free (fast) complex heterodynes**
  - **Multiplication by exp(jπn/2) = 1, j, -1, -j, ...**
    - **Circularly rotates transform data right by n/4 (90 degrees)**
  - **Multiplication by exp(-jπn/2) = 1, -j, -1, j, ...**
    - **Circularly rotates transform data left by n/4 (90 degrees)**

Fast Fourier Transform Tree



Choice of Algorithm depends on N



To be continued...

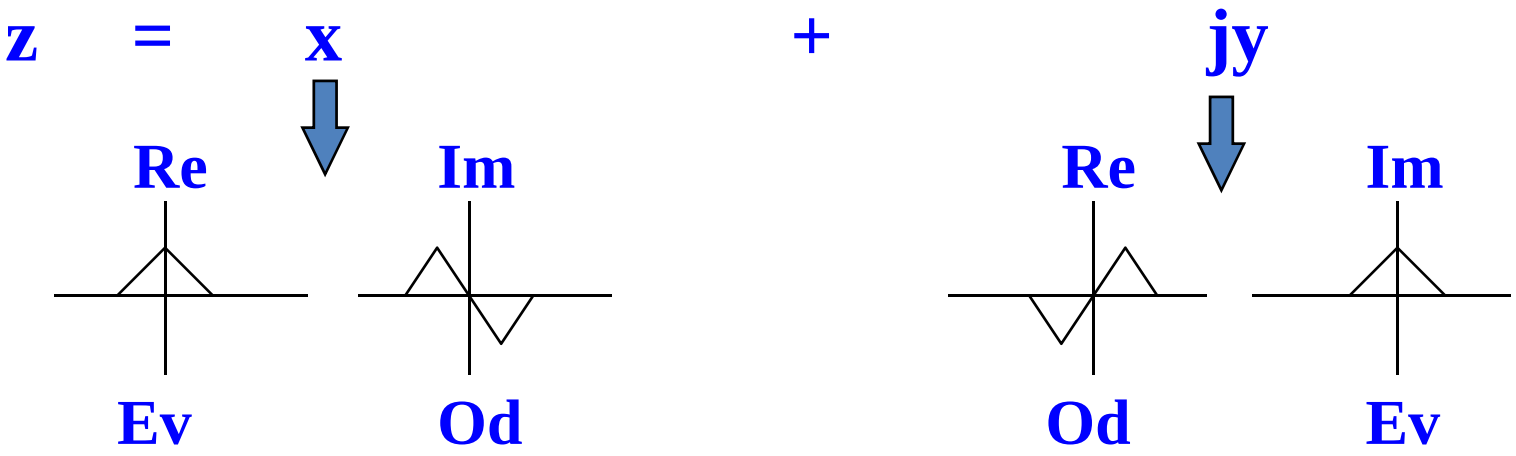
Fourier Symmetries

- Fourier Transform Symmetries
  - Even real data -> Even ; purely real
  - Odd real data -> Odd ; purely imaginary
  - Real data -> Even real; Odd imaginary
  - Imag data -> Odd real; Even imaginary
- **How do we remember all these symmetries?**
  - Ev <-> Ev (same) i.e., Re<->Re
  - Od <-> Od (other) i.e., Im <->Re, Re<->Im

Exploiting Symmetries

- **Double Transform Algorithm**
  - Fourier transform two real N-point sequences using one complex N-point transform.
- Given two real N-point sequences, **x** and **y**
  - Form **z=x+jy**
  - Fourier Transform **z** using FFT to obtain **Z**
  - **X =** Ev(Re(**Z**))+ jOd(Im(**Z**))
  - j**Y=** Od(Re(**Z**))+jEv(Im(**Z**)) or **Y** = Ev(Im(**Z**)) - jOd(Re(**Z**))

Exploiting Symmetries

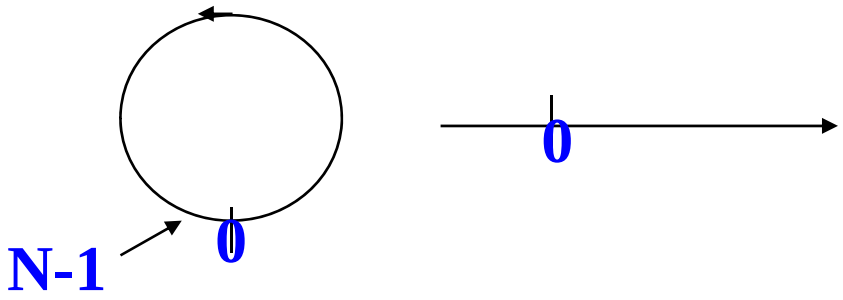


- **X =** Ev(Re(**Z**))+ jOd(Im(**Z**))
- j**Y=** Od(Re(**Z**))+jEv(Im(**Z**)) or **Y** = Ev(Im(**Z**)) - jOd(Re(**Z**))

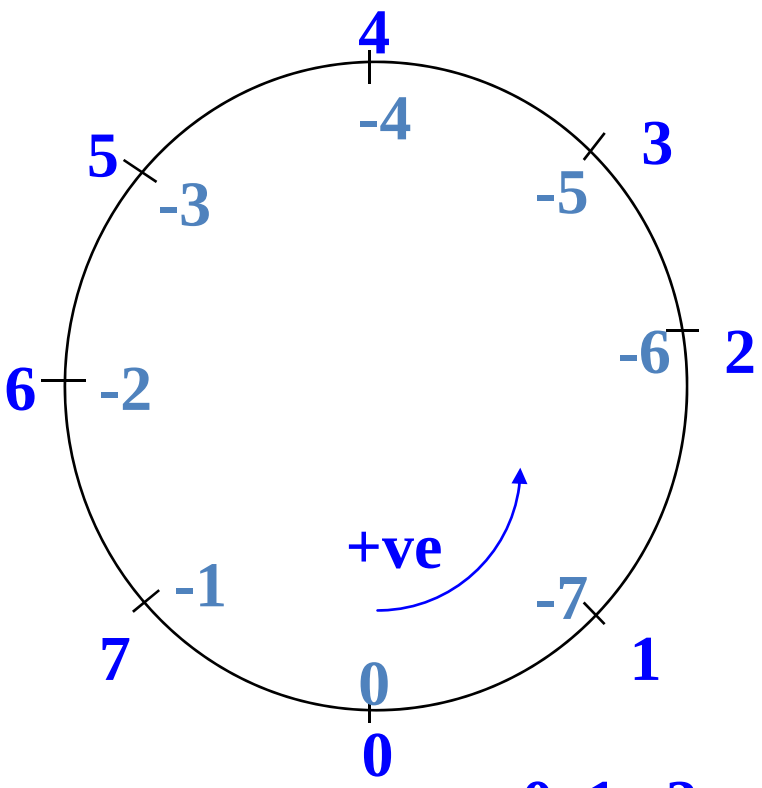
Doubles the efficiency of the FFT routine. With this modification the FFT achieves the same efficiency as the Fast Hartley Transform (Bracewell)

Implementation Issues

- In a real machine, if one set of numbers is larger than the other, numerical precision errors can decrease the SNR in the transform of the smaller data set.
- Solution: Prescale X and Y to have similar maximum amplitudes by left shifting data.
- Even and Odd have a slightly different interpretation in DFT data which is defined on the circle rather than the line.
  - Even -> X(n) = X(N-n)
  - Odd -> X(n) = -X(N-n)

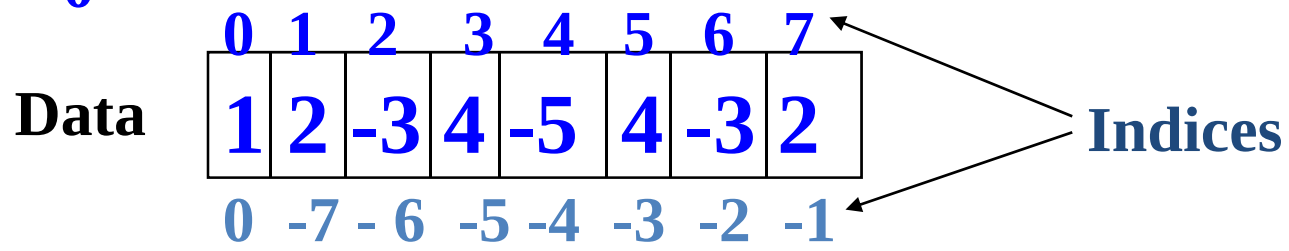


Circular Data



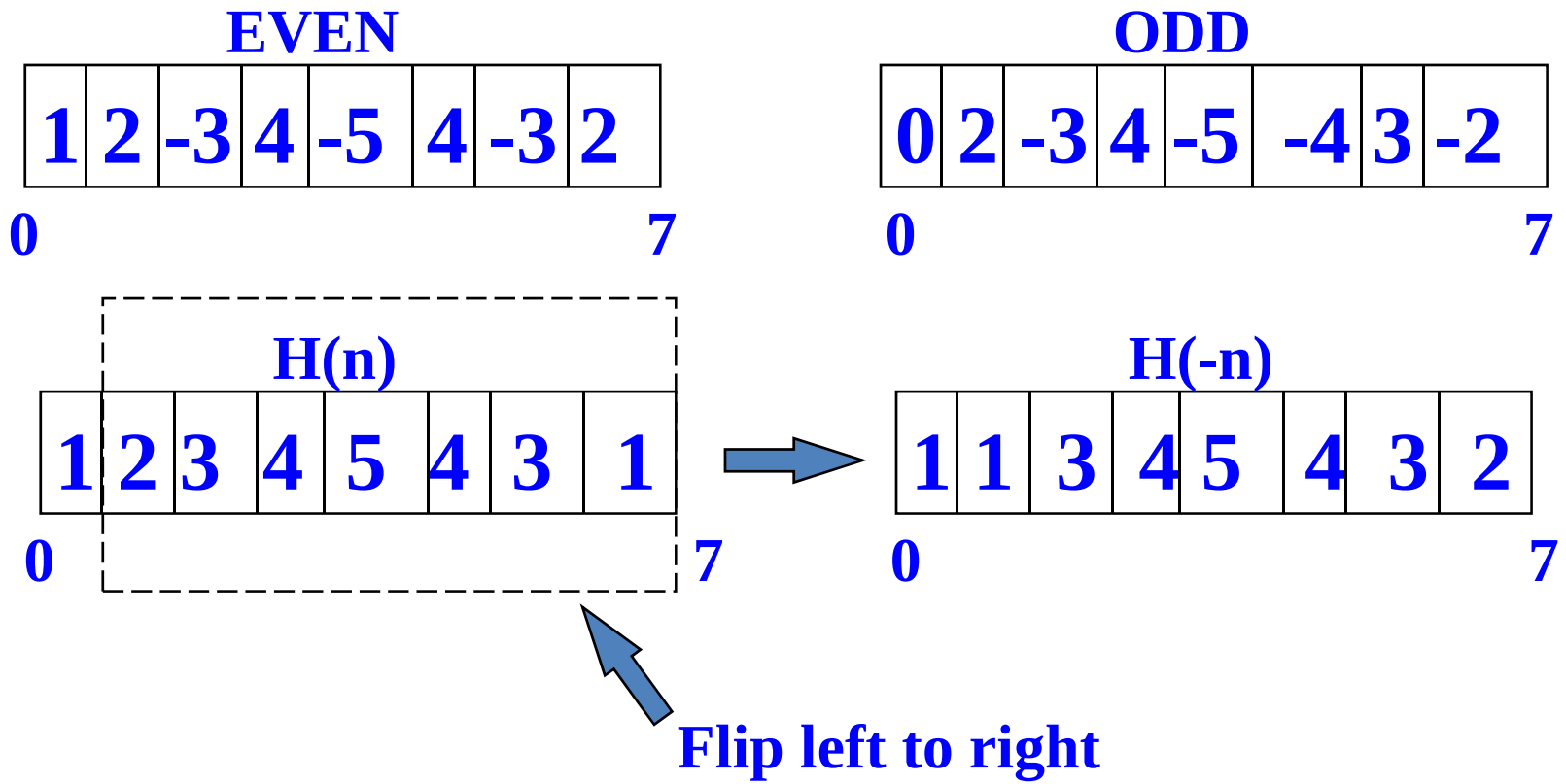
Usually represented as a vector by unwrapping anticlockwise from the origin.

Can be very misleading!

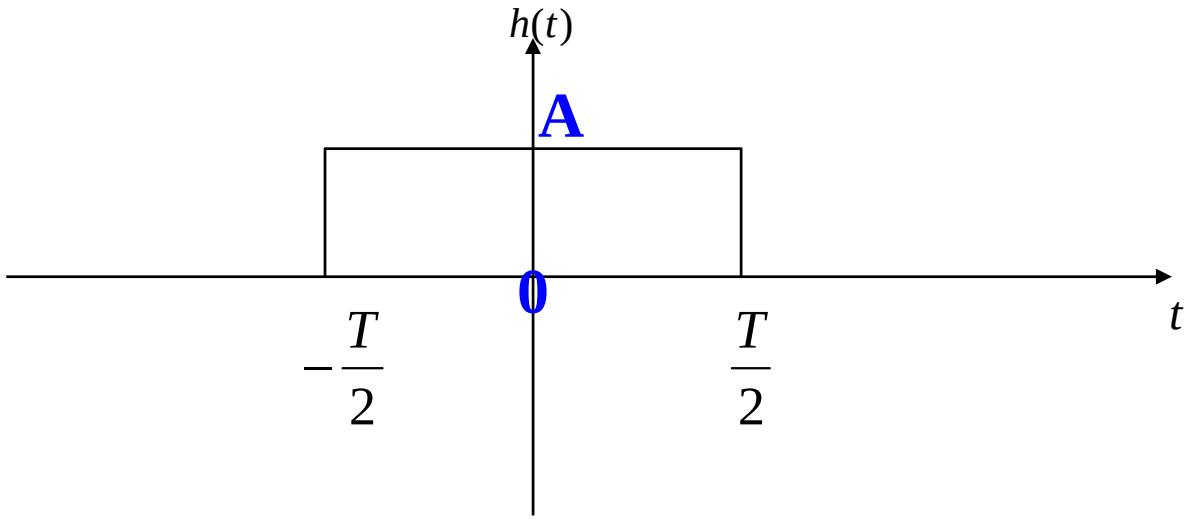


DFT Symmetric Vectors

Examples for N=8



Fourier Transform of Pulse

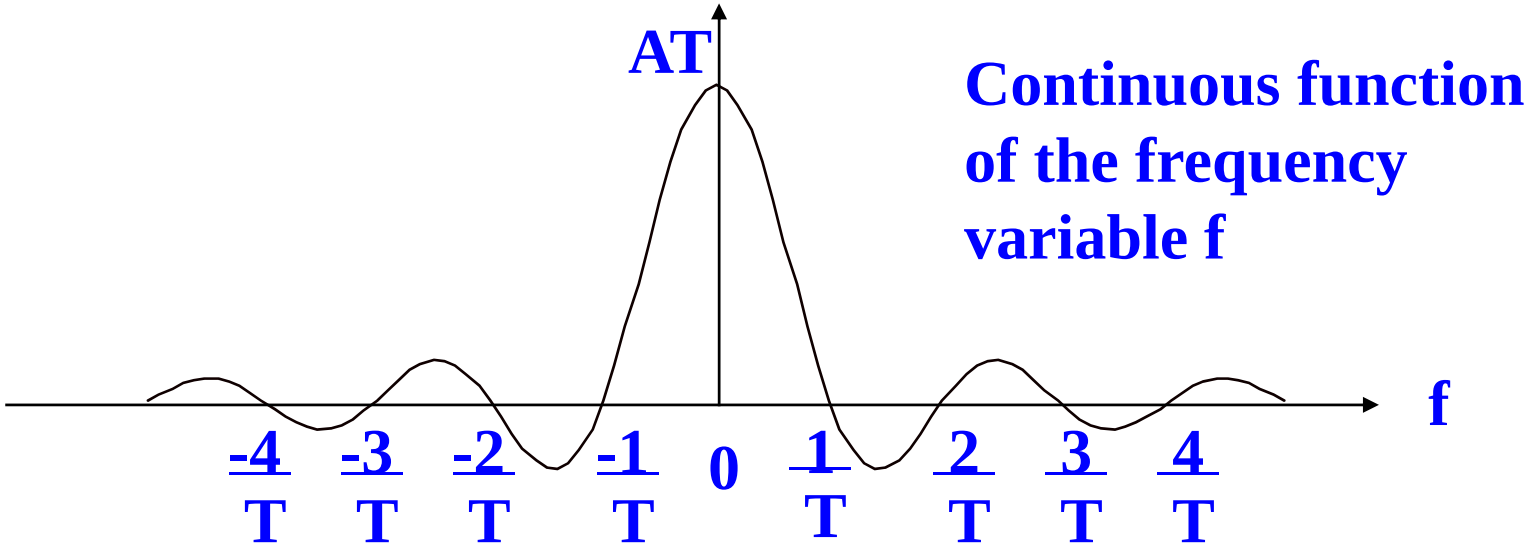


$$H(f) = \int_{-\infty}^{+\infty} h(t) e^{-j2\pi ft} dt$$

Fourier Transform of Pulse

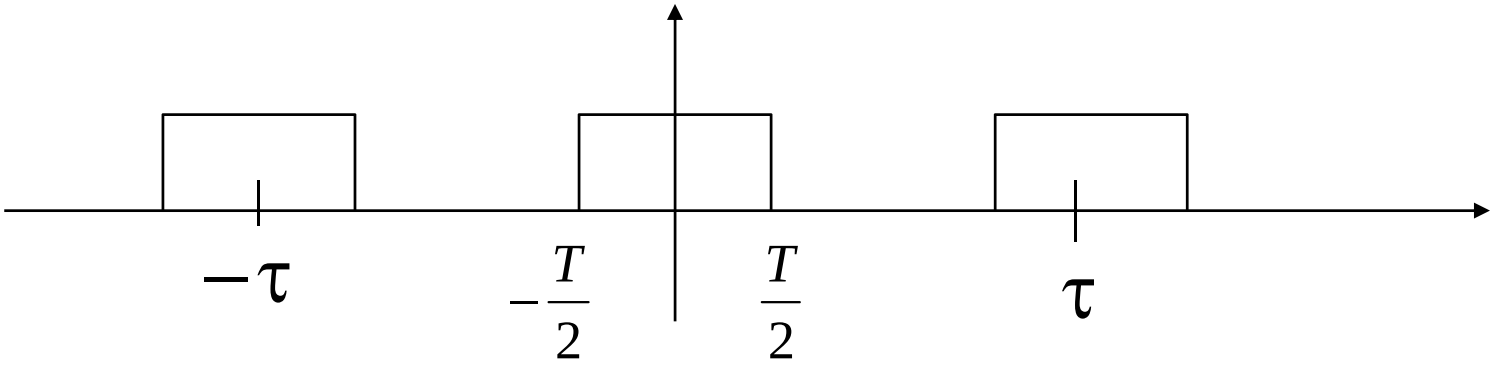
$$H(f) = AT\frac{\sin(2\pi fT/2)}{(2\pi fT/2)}$$

$$= AT\,\mathrm{sinc}(Tf)$$

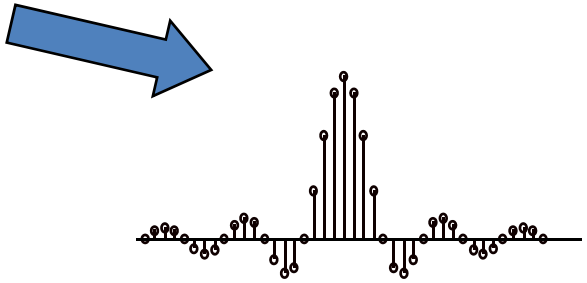


Continuous function of the frequency variable f

Fourier Transform of Pulse Train



Note: Periodization (circular nature) in time yields sampled transform

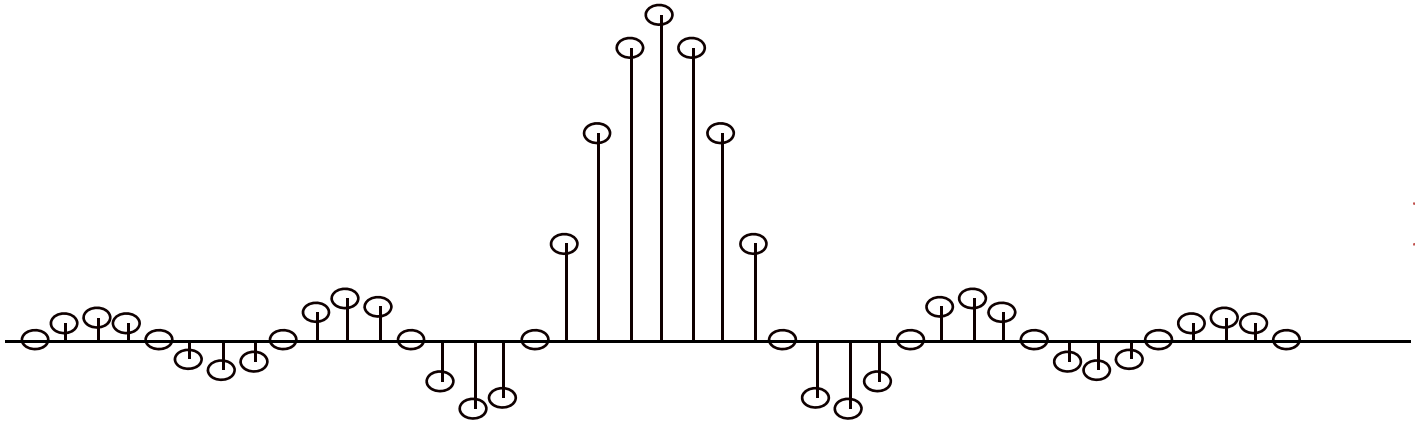


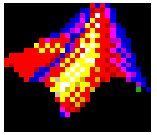
Fourier Transform of Pulse Train

Spacing of delta functions is $1/\tau$

Envelope Function is

$$H(f) = AT\,\mathrm{sinc}(Tf)$$

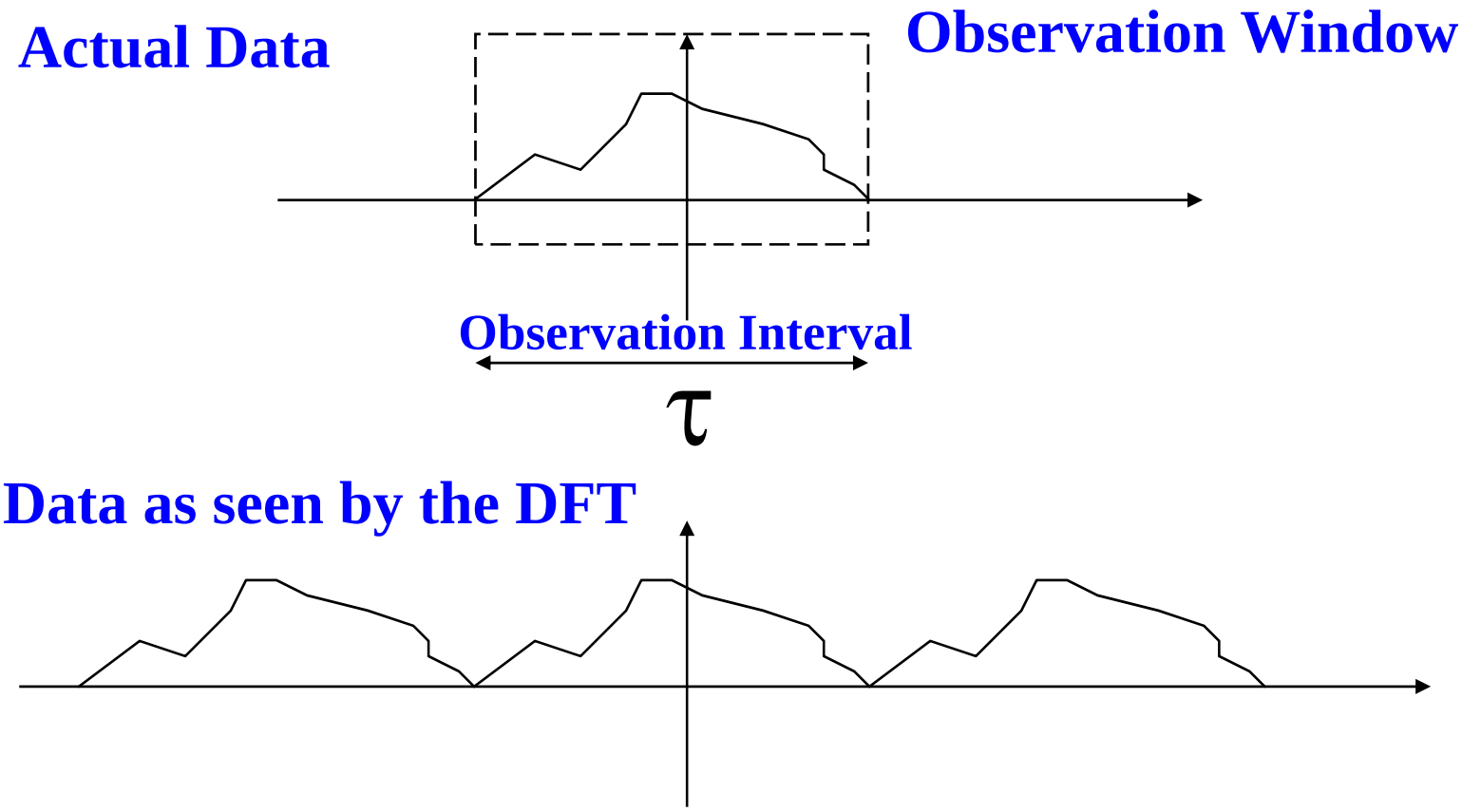




ptrain3

This is exactly the output one obtains by using a DFT (FFT). Thus when we analyse data with the DFT, there is an implicit (usually wrong) assumption that the data is a periodic function.

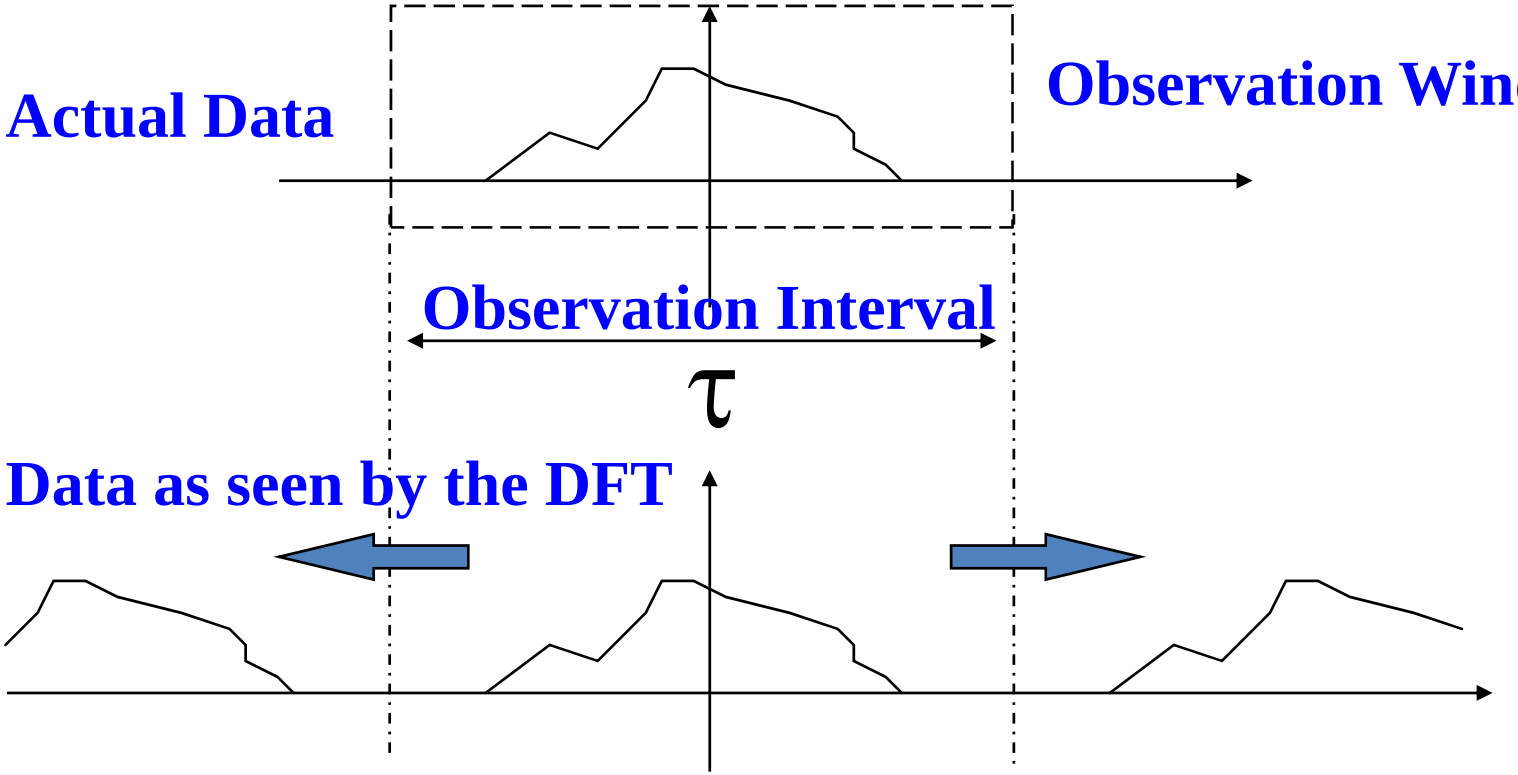
DFT Analysis



Increase Observation Interval

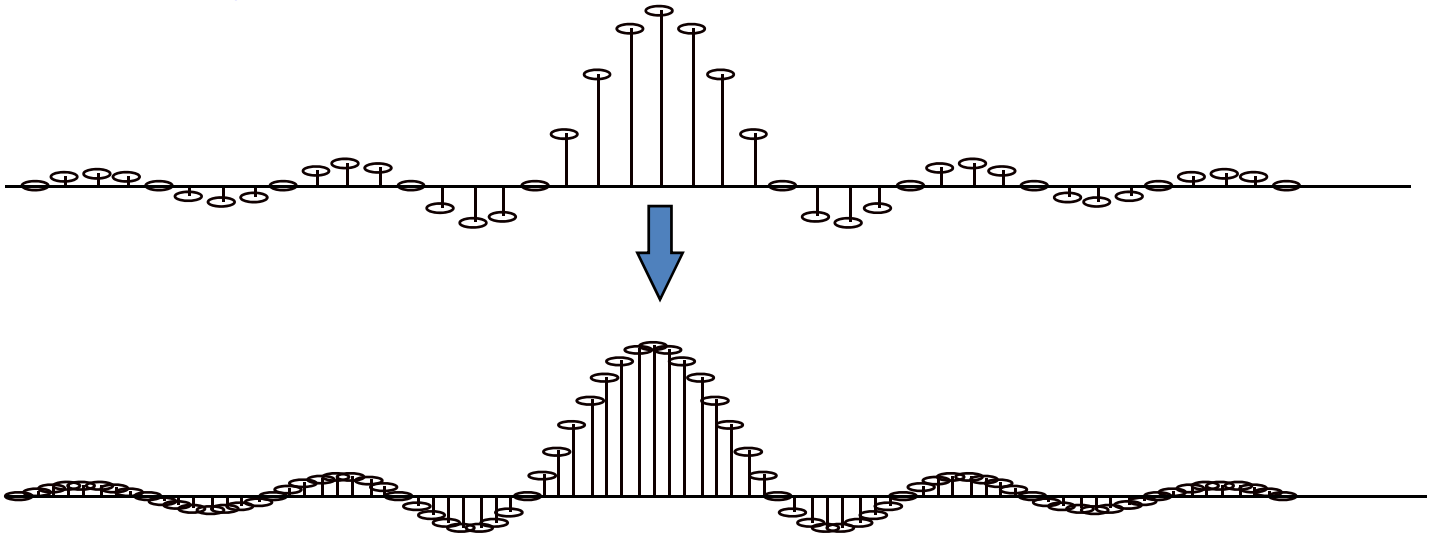
How? Zero pad the data

Works fine if data is really zero outside the observation window as in the case of isolated pulses



Increase Observation Interval

In the frequency domain the zero padding has the effect of reducing the spacing between delta functions, but the envelope (representing the FT of a single pulse) remains the same.



Sometimes refer to this process as sin(x) on x interpolation

Sampling

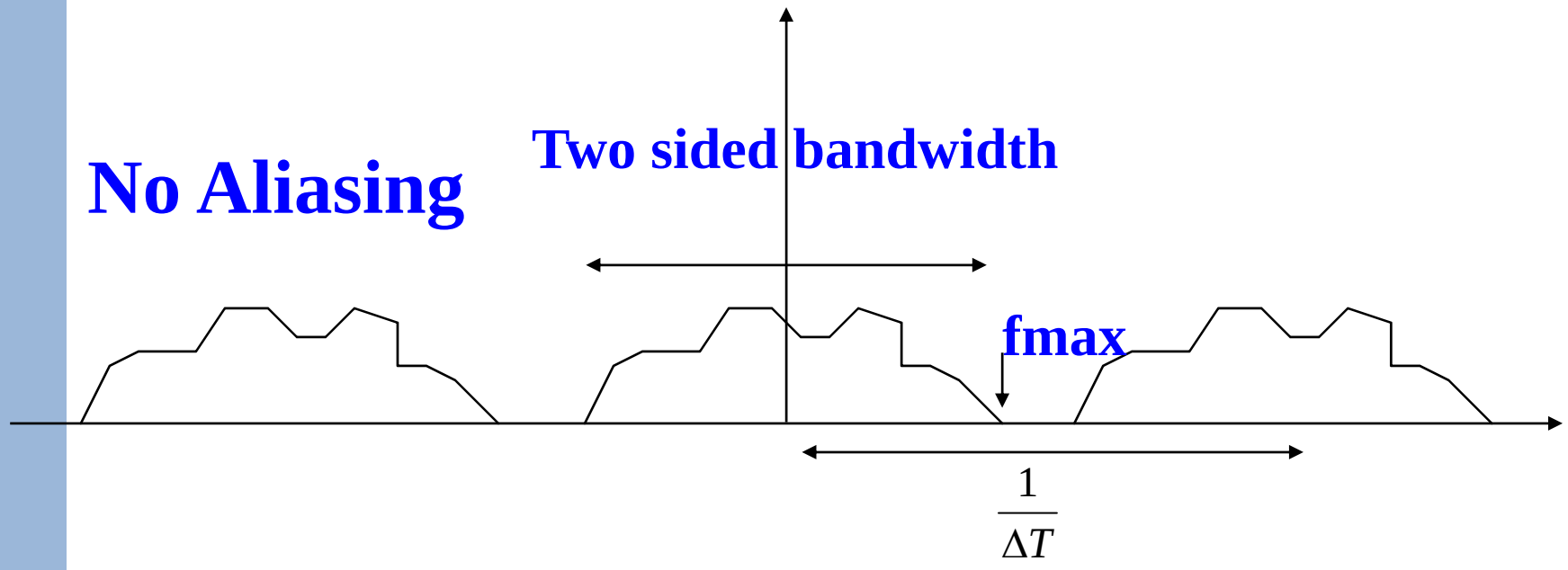
- We are really interested in discrete time (sampled) rather than continuous time signals.
- We have seen that when we transform periodic data (e.g., pulse trains), the frequency transform becomes discretized (sampled).
- The dual result is that when we discretize (sample) data in the time domain, the frequency transform becomes periodized (replicated).
- We must sample the data at a sufficient rate to ensure that the spectral replicas do not overlap to avoid aliasing (if required).

Sampling

Sampling Theorem $\frac{1}{\Delta T} \geq$ Two Sided Bandwidth

Not twice the maximum signal frequency!
Only applies to baseband signals

No Aliasing

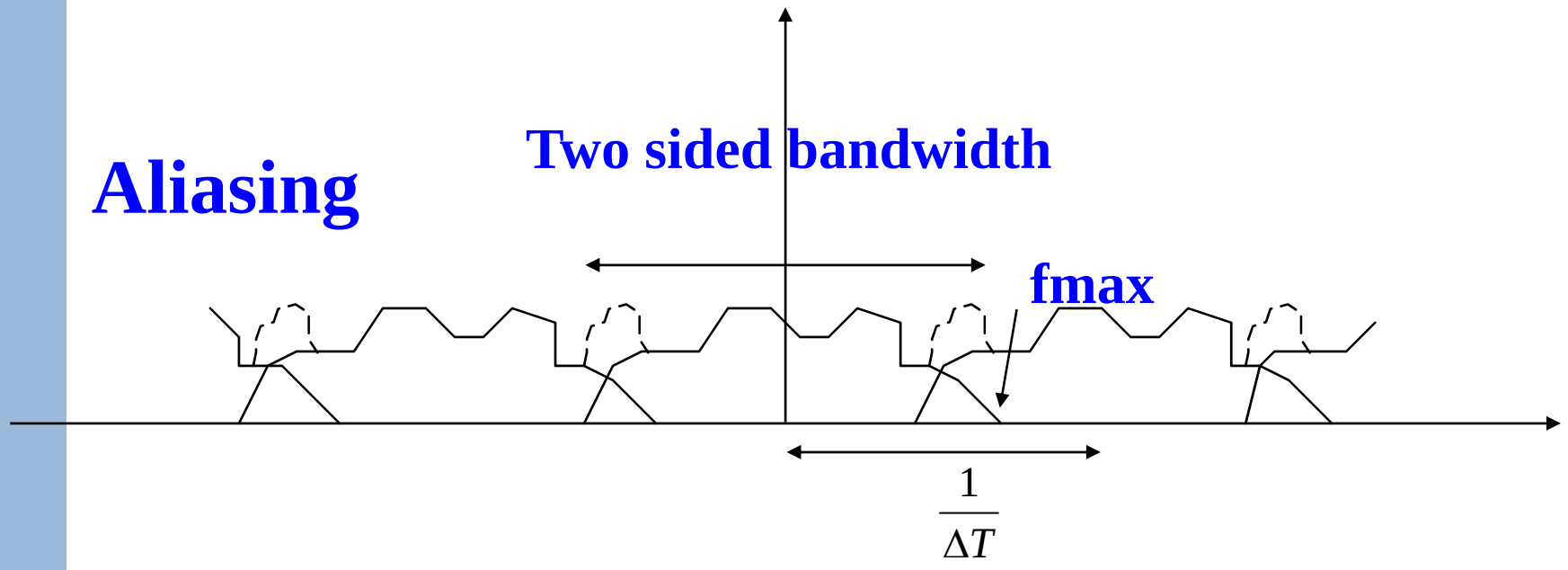


Sampling

Sampling Theorem $\frac{1}{\Delta T} \geq$ Two Sided Bandwidth

Not twice the maximum signal frequency!
Only applies to baseband signals

Aliasing

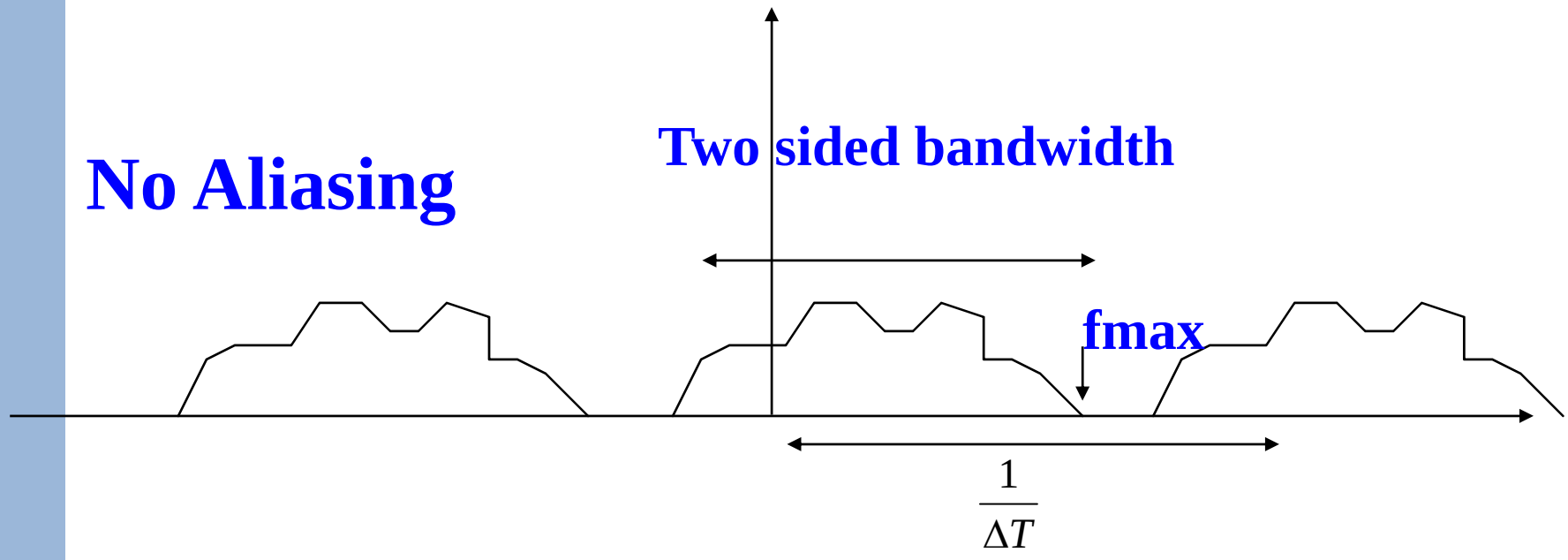


Sampling

Sampling Theorem $\frac{1}{\Delta T} \geq$ Two Sided Bandwidth

Frequency Shifted Signal

No Aliasing



Aliasing

Example of an aliased sinc function