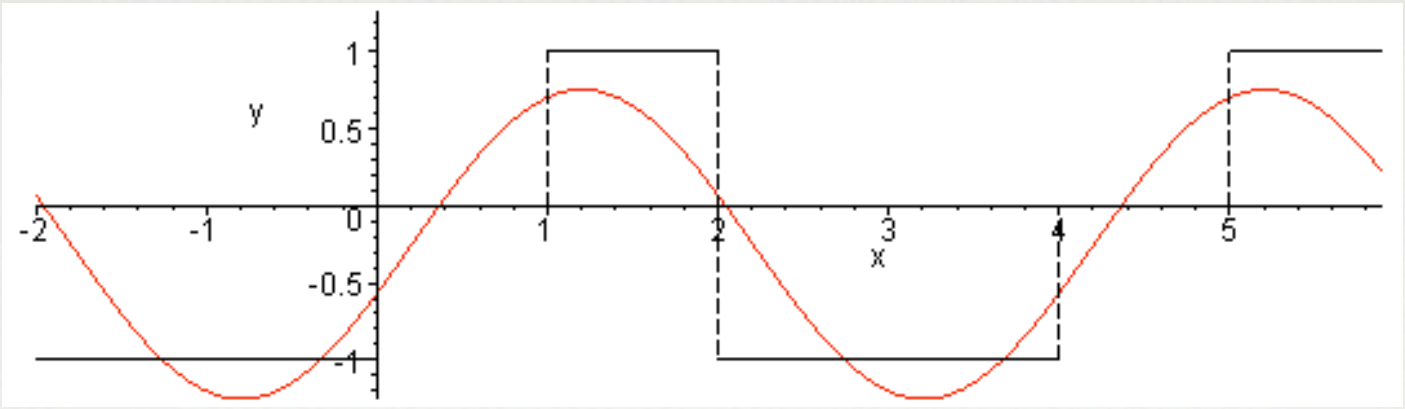
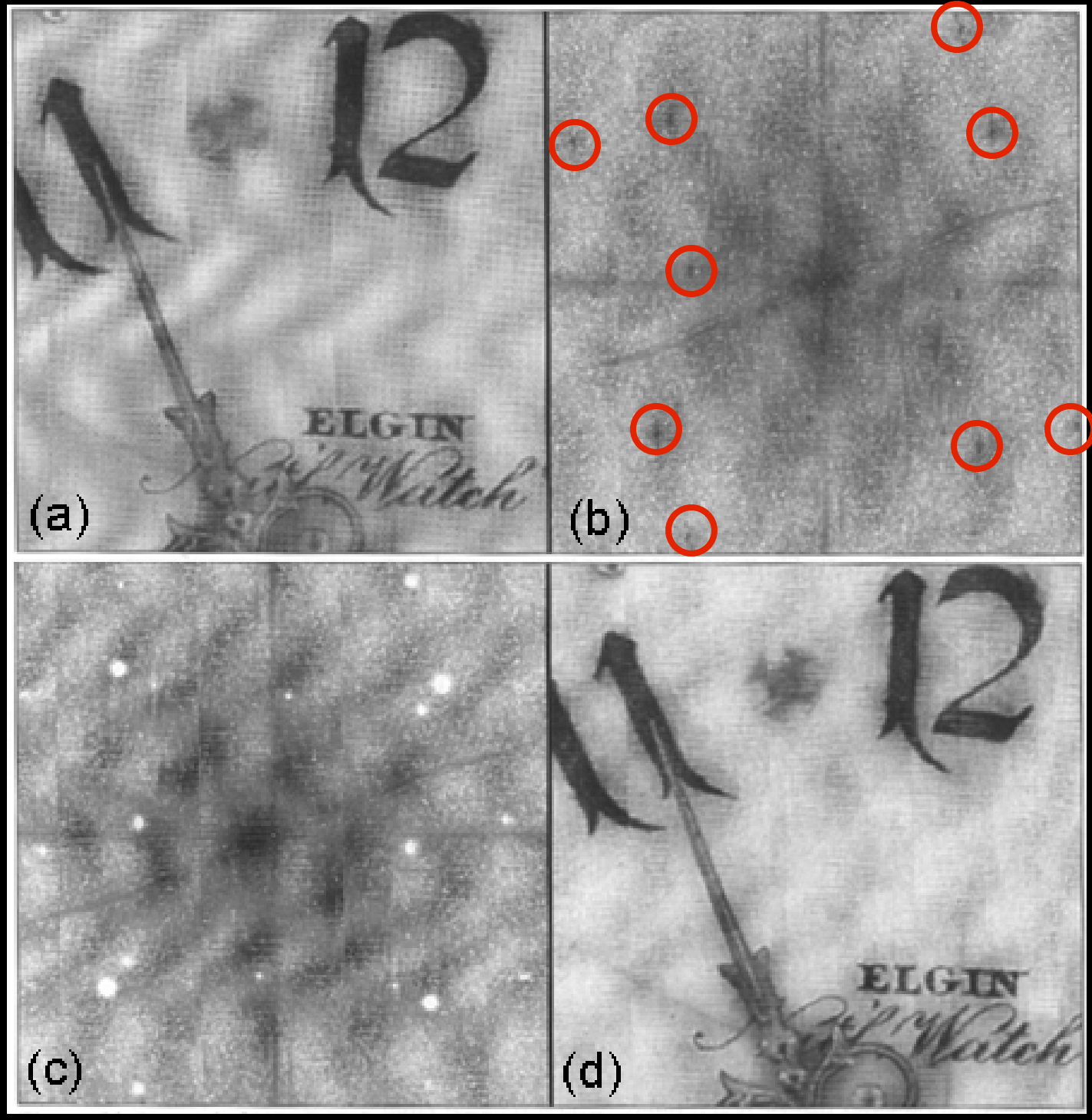
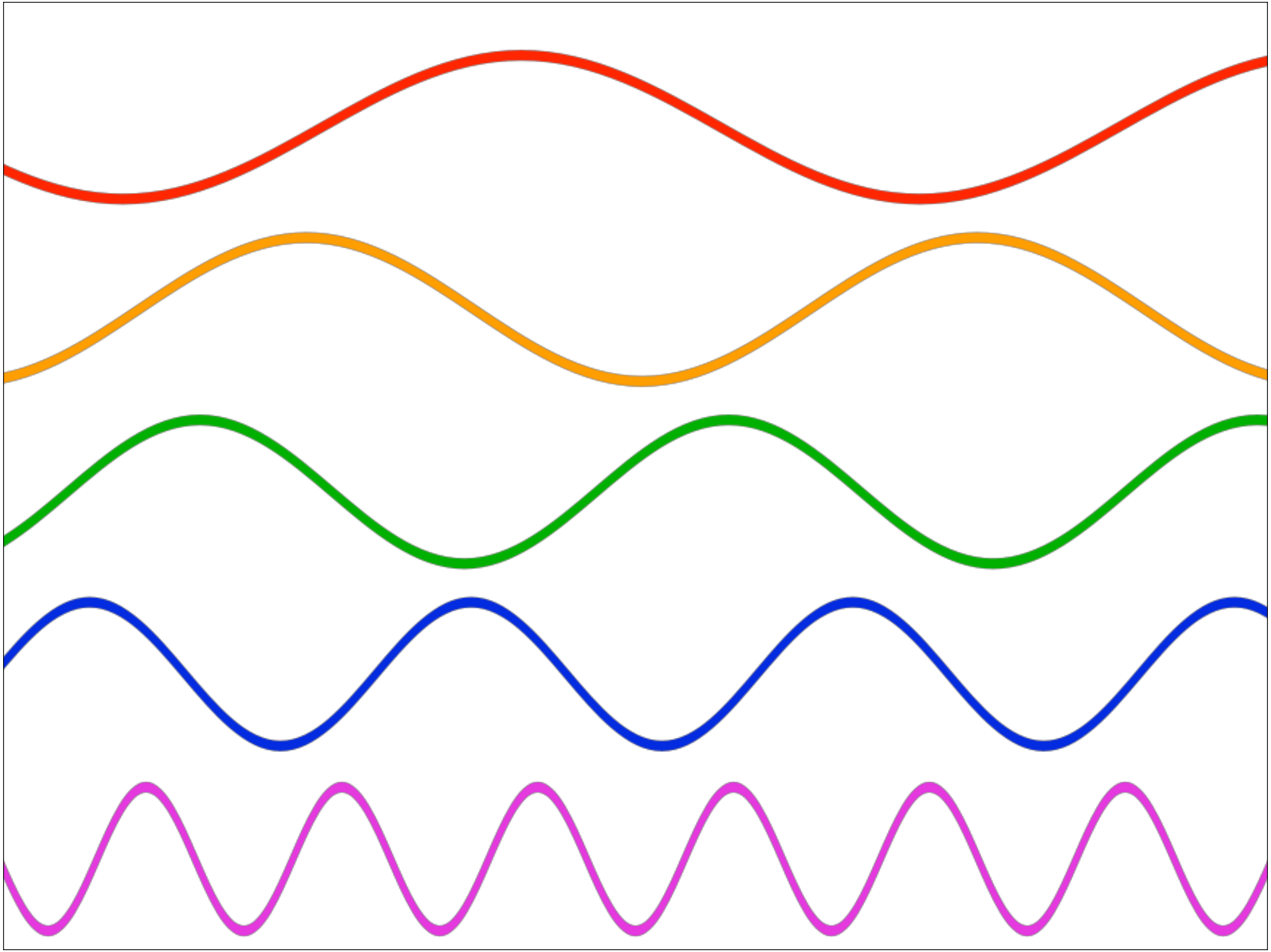


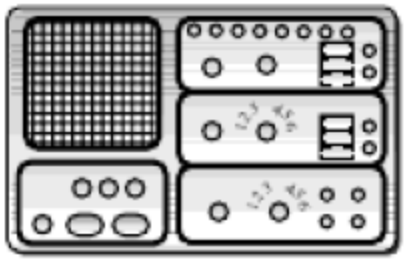
PHY-4810
COMPUTATIONAL PHYSICS

LECTURE 17: FOURIER TRANSFORMS


12
ELGIN
(a)
(b)
(c)
12
ELGIN
(d)

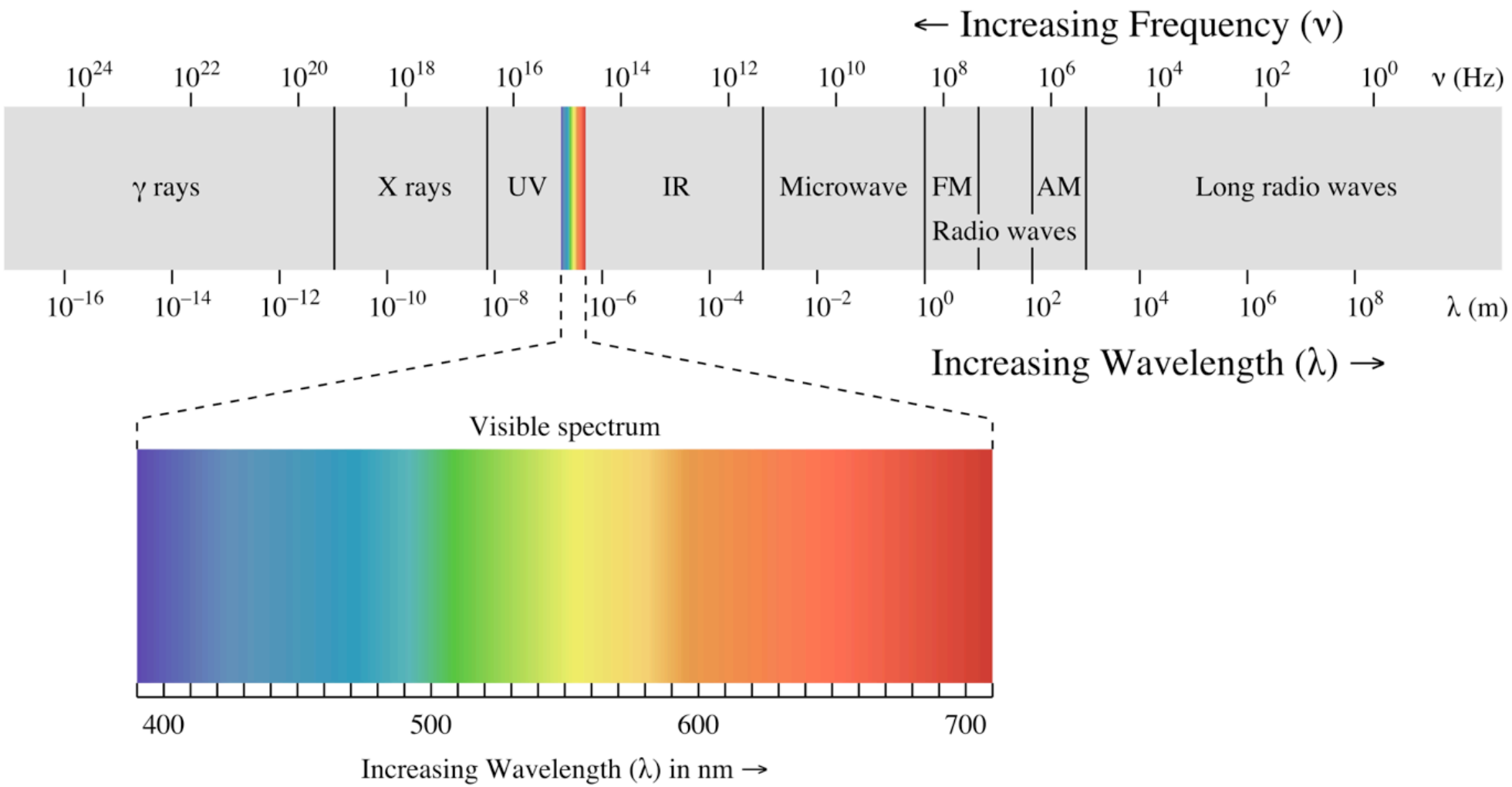


SINE WAVE



VIOLIN

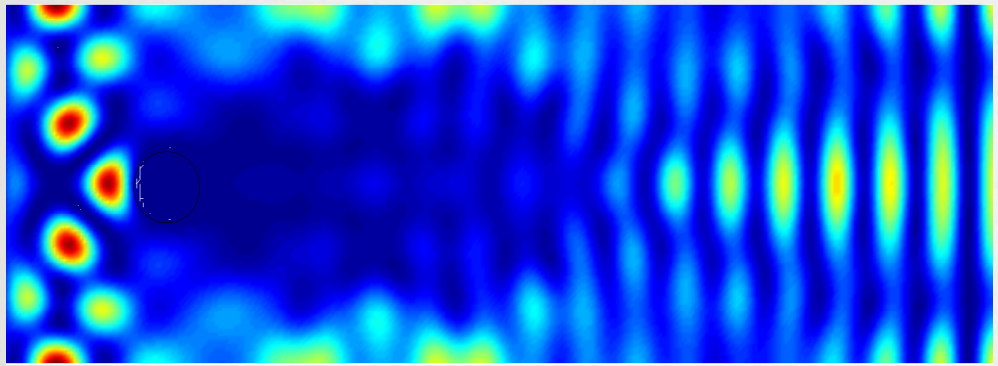
PIANO

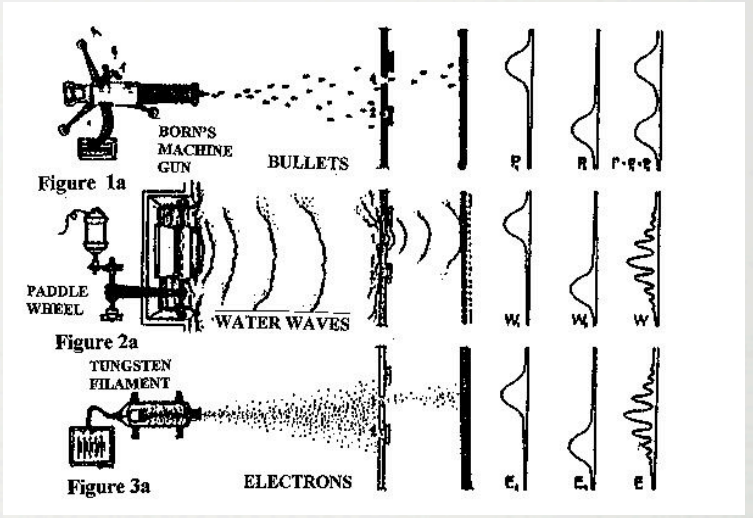

← Increasing Frequency (ν)
10^{24} 10^{22} 10^{20} 10^{18} 10^{16} 10^{14} 10^{12} 10^{10} 10^{8} 10^{6} 10^{4} 10^{2} 10^{0} ν (Hz)
γ rays
X rays
UV
IR
Microwave
FM
AM
Radio waves
Long radio waves
10^{-16} 10^{-14} 10^{-12} 10^{-10} 10^{-8} 10^{-6} 10^{-4} 10^{-2} 10^{0} 10^{2} 10^{4} 10^{6} 10^{8} λ (m)
Increasing Wavelength (λ) →
Visible spectrum
400
500
600
700
Increasing Wavelength (λ) in nm →

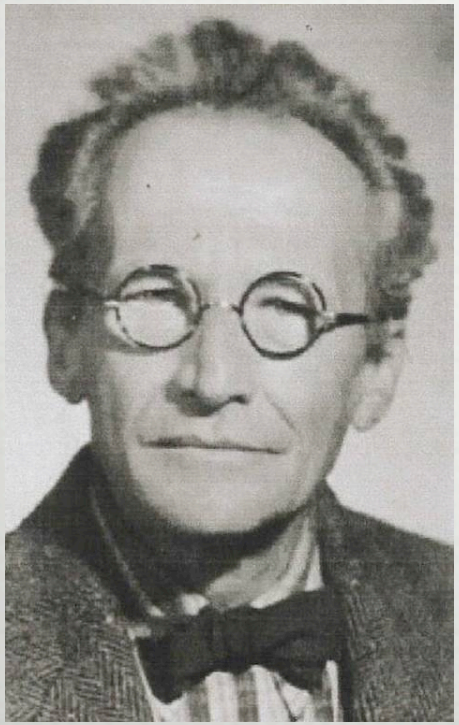
Frequency Range

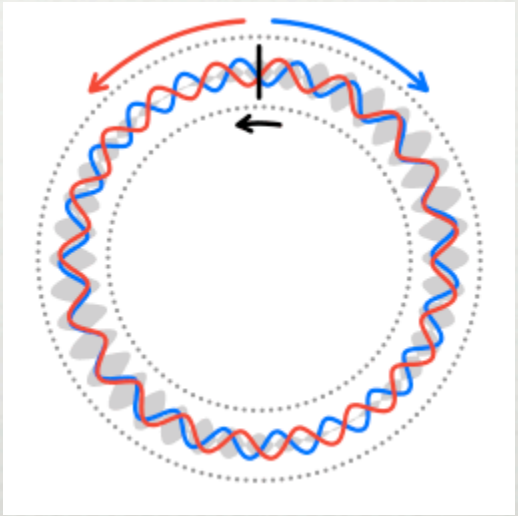
Species	Approximate Range (Hz)
human	64-23,000
dog	67-45,000
cat	45-64,000
cow	23-35,000
horse	55-33,500
sheep	100-30,000
rabbit	360-42,000
rat	200-76,000
mouse	1,000-91,000
gerbil	100-60,000
guinea pig	54-50,000
hedgehog	250-45,000
raccoon	100-40,000
ferret	16-44,000
opossum	500-64,000
chinchilla	90-22,800
bat	2,000-110,000
beluga whale	1,000-123,000
elephant	16-12,000
porpoise	75-150,000
goldfish	20-3,000
catfish	50-4,000
tuna	50-1,100
bullfrog	100-3,000
tree frog	50-4,000
canary	250-8,000
parakeet	200-8,500
cockatiel	250-8,000
owl	200-12,000
chicken	125-2,000

WAVE MECHANICS









GENERAL INTRODUCTION

- Fourier transforms are among the most ubiquitous tools of modern physics.

- One way to describe them is that they allow us to switch back and forth between the description of a physical process as a function of time (or position), and the description of the same process as a function of frequency (or spatial frequency).

$$t \rightleftarrows \omega \qquad x \rightleftarrows k_x$$

SPECTRAL EXPANSIONS

- General definition of spectral expansion: **the representation of a function *h(x)* as a sum, with appropriate coefficients, over a set of *N* basis functions:**

$$h(x) = \sum_{k=0}^{N-1} \tilde{h}_k \phi_k(x)$$

BASIS OF THE EXPANSION

- Note that *N* might be infinite, and that the index *k* might actually represent a continuous variable
(in that case, the symbolic sum should be written as an integral).

SPECTRAL EXPANSIONS

- Different spectral expansions use different bases, chosen to have certain desired properties
- For instance, the Fourier expansion uses functions of definite frequency, the complex exponentials

$$\phi_k(x) = \exp(-2\pi i f_k x)$$

FOURIER SERIES

- Consider a periodic function *h(x)* with *h(x+L) = h(x)*;
- Without losing any information, we may then restrict our considerations to the interval [0, L], and write the Fourier series

$$h(x) = \sum_{k=-\infty}^{\infty} \tilde{h}_k e^{-2\pi i f_k x}, \quad f_k = k/L$$

- Where the Fourier coefficients h_k can be recovered as

$$\tilde{h}_k = \frac{1}{L} \int_0^L h(x) e^{2\pi i f_k x} \, dx$$

SPECIAL FREQUENCIES

- **k=0 : constant offset from zero**
  - ***(i.e. average value of the function)***
- **k=+/- 1: smallest frequency included in the series (1/L)**
  - ***(i.e. one oscillation in the [0,L] interval)***
- **1/L difference between successive frequencies**
  - ***(i.e. frequency resolution)***

$$h(x) = \sum_{k=-\infty}^{\infty} \tilde{h}_k e^{-2\pi i f_k x}, \quad f_k = k/L$$

$$\tilde{h}_k = \frac{1}{L} \int_0^L h(x) e^{2\pi i f_k x} \, dx$$

DISCRETE FOURIER TRANSFORMS

$$h(x) = \sum_{k=-\infty}^{\infty} \tilde{h}_k e^{-2\pi i f_k x}, \quad f_k = k/L$$

$$\tilde{h}_k = \frac{1}{L}\int_0^L h(x) e^{2\pi i f_k x}\, dx$$

- **Problems:**
  - **(1) the function *h(x)* is reconstructed by means of an infinite series**
  - **(2) the coefficients h_k are expressed formally as analytic integrals over a continuous range of *x* values**

$$h_j = h(x_j), \quad \text{with} \quad x_j = jL/N, \quad \text{and} \quad j = 0, \ldots, N-1$$

Special note about j=N
(function is periodic, that point is equivalent to j=0)

DISCRETE FOURIER TRANSFORM AND ITS INVERSE

- **We can truncate the Fourier series at the Nyquist frequency, setting *M = N/2***

$$h_j = \sum_{k=-N/2}^{N/2-1} \tilde{h}_k e^{-2\pi i f_k x_j}, \quad \tilde{h}_k = \frac{1}{N} \sum_{i=0}^{N-1} h_j e^{2\pi i f_k x_j}, \quad \text{with} \quad x_j = jL/N, \; f_k = k/L.$$

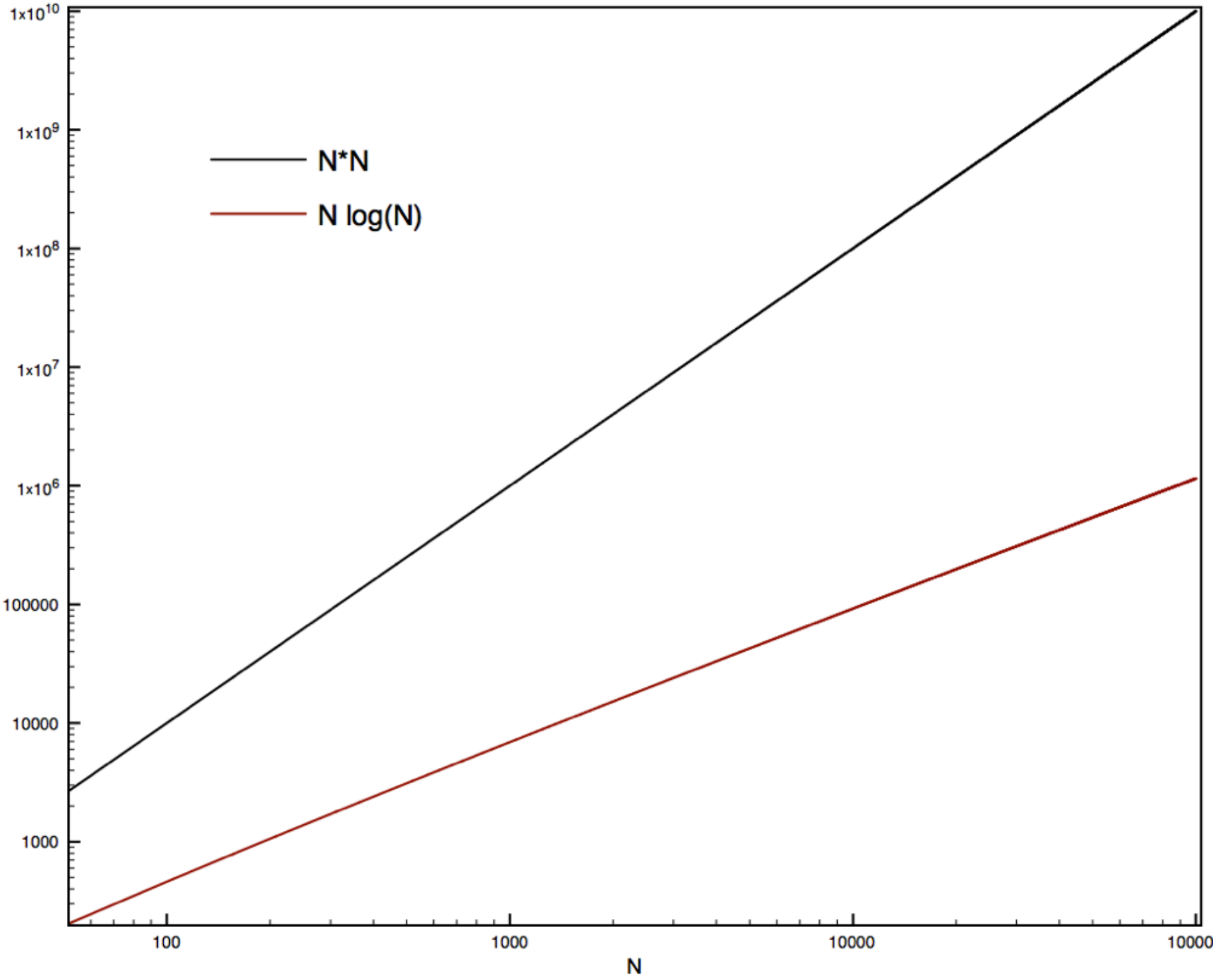
- **These equations define the discrete Fourier transform and its inverse, which map a set of *N* physical-space values h_i into a set of *N* Fourier-space coefficients h_k, and backwards.**

In computational applications, it is often not so important whether we know how to do something, but whether we know how to do it fast.

FAST FOURIER TRANSFORM

$$\tilde{h}_k = \frac{1}{N}\sum_{j=0}^{N-1} h_j e^{2\pi i j k/N} \text{ (transform)}, \quad h_j = \sum_{k=0}^{N-1} \tilde{h}_k e^{-2\pi i j k/N} \text{ (inverse transform)}$$

- The discrete Fourier transform owes its popularity and success to the fact that computer scientists have devised fast **Fourier transform (FFT) algorithms** that can evaluate both transforms,in **O(N log N)** operations instead of the **$O(N^2)$** operations that are (apparently) implicit
- This is a dramatic speedup!


1x10^10
1x10^9
1x10^8
1x10^7
1x10^6
100000
10000
1000
N*N
N log(N)
100
1000
10000
10000
N

FFT: THE DANIELSON-LANCZOS LEMMA

$$\tilde{h}_k = \frac{2}{2N}\sum_{j=0}^{N/2-1} h_{2j} e^{2\pi i j k/(N/2)} + \frac{2e^{2\pi i k/N}}{2N}\sum_{j=0}^{N/2-1} h_{2j+1} e^{2\pi i j k/(N/2)}$$

$$= 2\,\tilde{h}_k^{\rm even} + 2\,e^{2\pi i k/N}\,\tilde{h}_k^{\rm odd},$$

- Separate odd and even factors from the sum
- We end up with 2 FFT of half the size:
  - we replace a N^2 problem by a 2x $(N/2)^2$
  - we can repeat the process and at the end, we have a FFT of just one number!
- We end up with a N log (N) scaling

IN PRACTICE: NUMERICAL RECIPE INTERFACE

`struct WrapVecDoub {`
Object for accessing a `VecDoub` as if it were a complex vector of half the length, with wraparound periodicity.

```
Int j,n=256;
VecDoub dat(2*n);
WrapVecDoub data1(dat), data2(2*n);
for (j=0;j<n;j++) {
    data1[j] = Complex(... , ...);
    data2.real(j) = ... ;
    data2.imag(j) = ... ;
}
four1(data1,1);
four1(data2,1);
for (j=-n/2;j<n/2;j++) {
    ... = data1.real(j);
    ... = data2[j];
}
```

256 complex components, e.g.
A real vector to hold them.
Examples of the two constructors.
Loop over complex components.
Set a complex value directly,
or set real and imag separately.

Invokes `four1(VecDoub&,Int)` through the conversion operator.
Can address negative frequencies directly!
Get real part of component j.
Get component as a complex value.

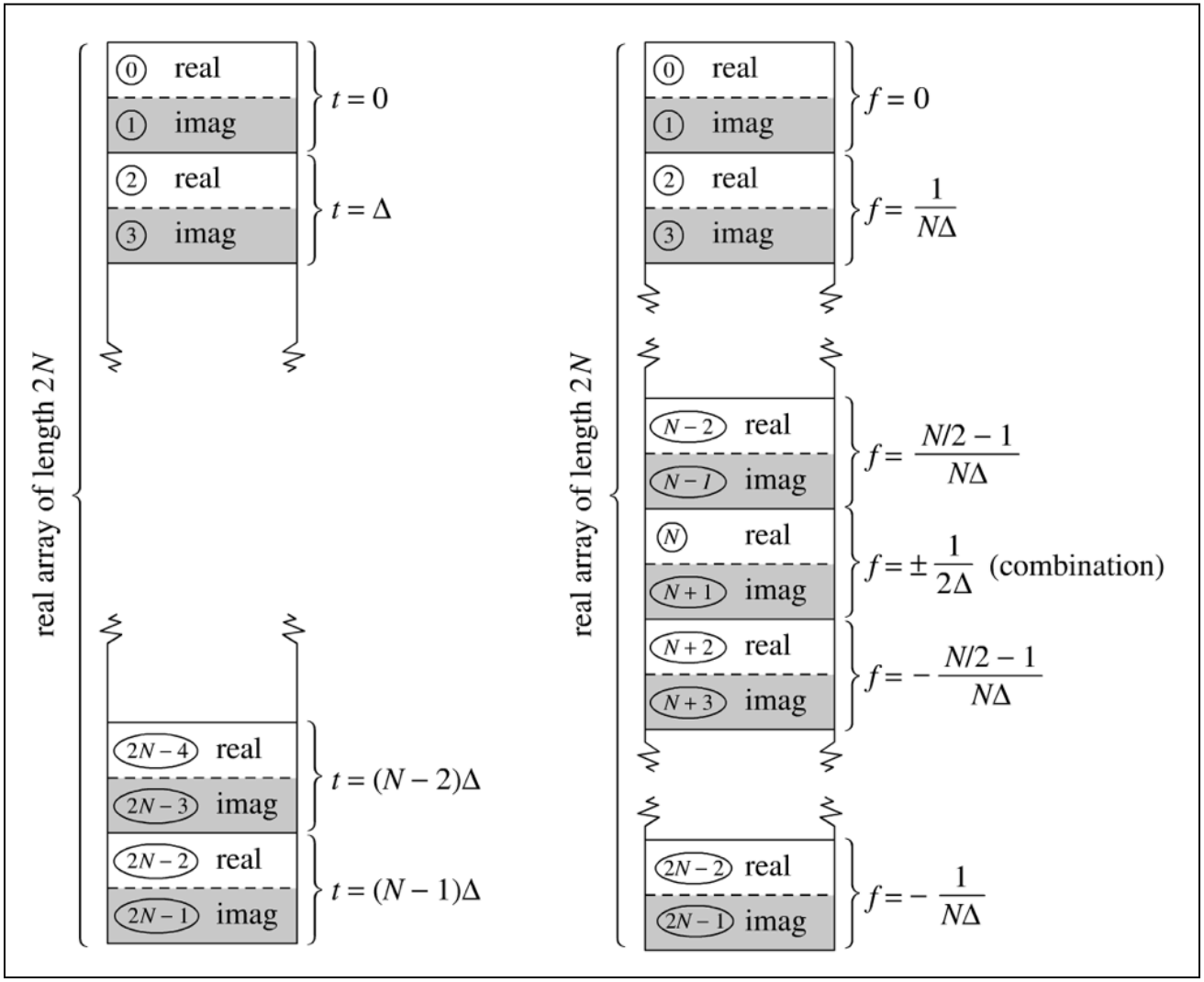


Figure 12.2.2. Input and output arrays for FFT. (a) The input array contains N (a power of 2) complex time samples in a real array of length $2N$, with real and imaginary parts alternating. (b) The output array contains the complex Fourier spectrum at N values of frequency. Real and imaginary parts again alternate. The array starts with zero frequency, works up to the most positive frequency (which is aliased with the most negative frequency). Negative frequencies follow, from the second-most negative up to the frequency just below zero.

EXAMPLE OF C++ CODE USING NUMREC

```
#include "nr3.h"
#include "fourier.h"

int main()
{
    VecDoub mysignal(2*ndata);
    WrapVecDoub signal(mysignal);

    //here we need to define the signal
    for(i=0;i<ndata;i++){
      signal.real(i)=...;
      signal.imag(i)=...;
    }

//Fourier Transform
    four1(signal,-1);

    //The Fourier transform is now in signal.real and signal.imag


//INVERSE FT
    four1(signal,1);

}
```

NOTE ON FFT

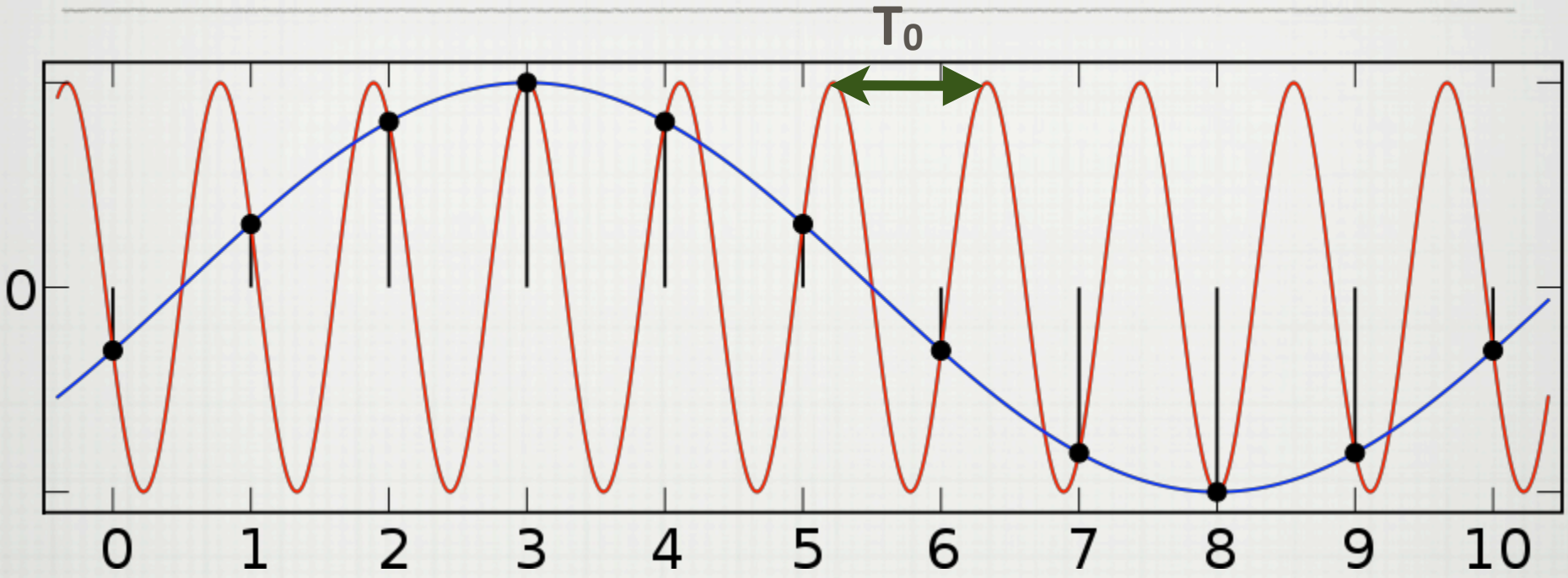
- The best implementation of FFT is fftw
- at least, that's the Fastest FT in the West.
- This can be found at fftw.org

EXAMPLE OF SIMPLE FFTW PROGRAM

```
#include <iostream>
#include <fftw3.h>
#include <math.h>
using namespace std;
int main (int argc, char * const argv[]) {

  int N=32768;

  fftw_complex *in, *out;
  fftw_plan p;
  float x=0, dx=0.5;

  in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
  out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);

  for(int i=0;i<N;i++){
    x=x+dx;
    in[i][0]=sin(x)/(1);
    in[i][1]=0.0;
  }
  p = fftw_plan_dft_1d(N, in, out, FFTW_FORWARD, FFTW_ESTIMATE);
  fftw_execute(p);
  for(int i=0;i<N/2;i++){
    cout << 1.0*i/(N*dx) << " " << sqrt(out[i][0]*out[i][0]+out[i][1]*out[i][1]) << en
dl;
  }
  fftw_destroy_plan(p);
  fftw_free(in); fftw_free(out);

 }
```

PROBLEMS WITH FFT: ALIASING & LEAKAGE

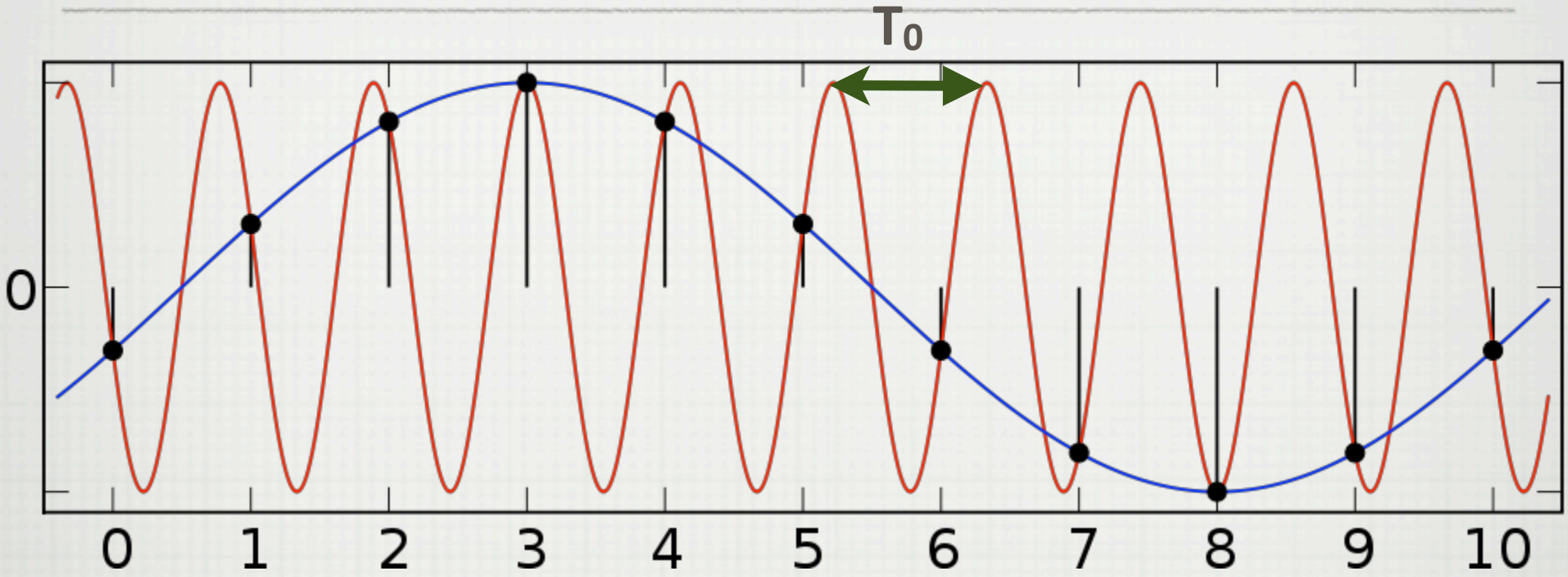
ALIASING



At least two points for each oscillation are needed to sample the original signal (red curve)

In other words, sample rate $\Delta t <= T_0/2$, (T_0=1.111s)
This corresponds to frequency: $2/T_0$ =1.8 hz which is exactly twice that of the highest frequency of the signal.
This is called the Nyquist frequency: you need to sample at a frequency at least twice that of the highest frequency in your signal

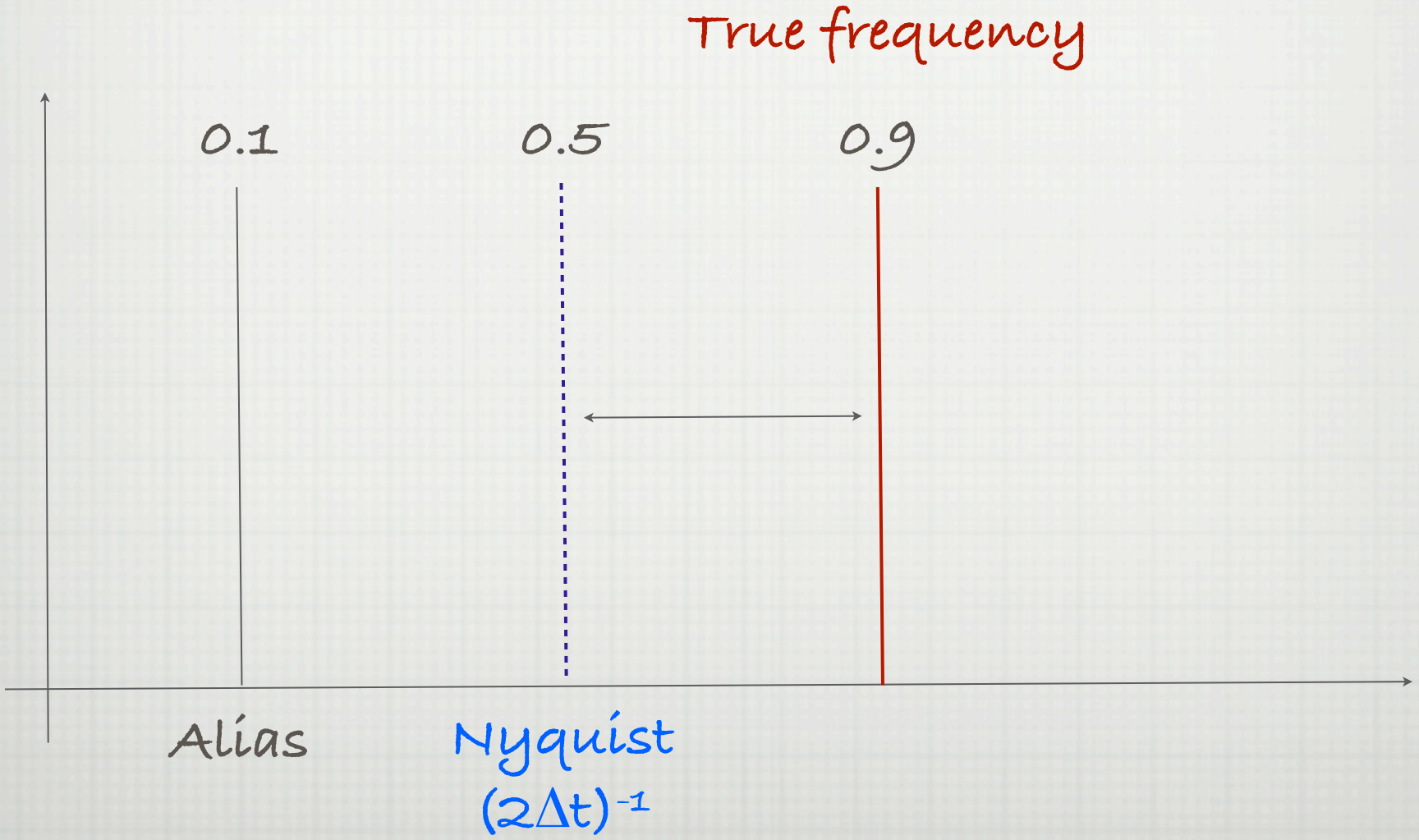
ALIASING



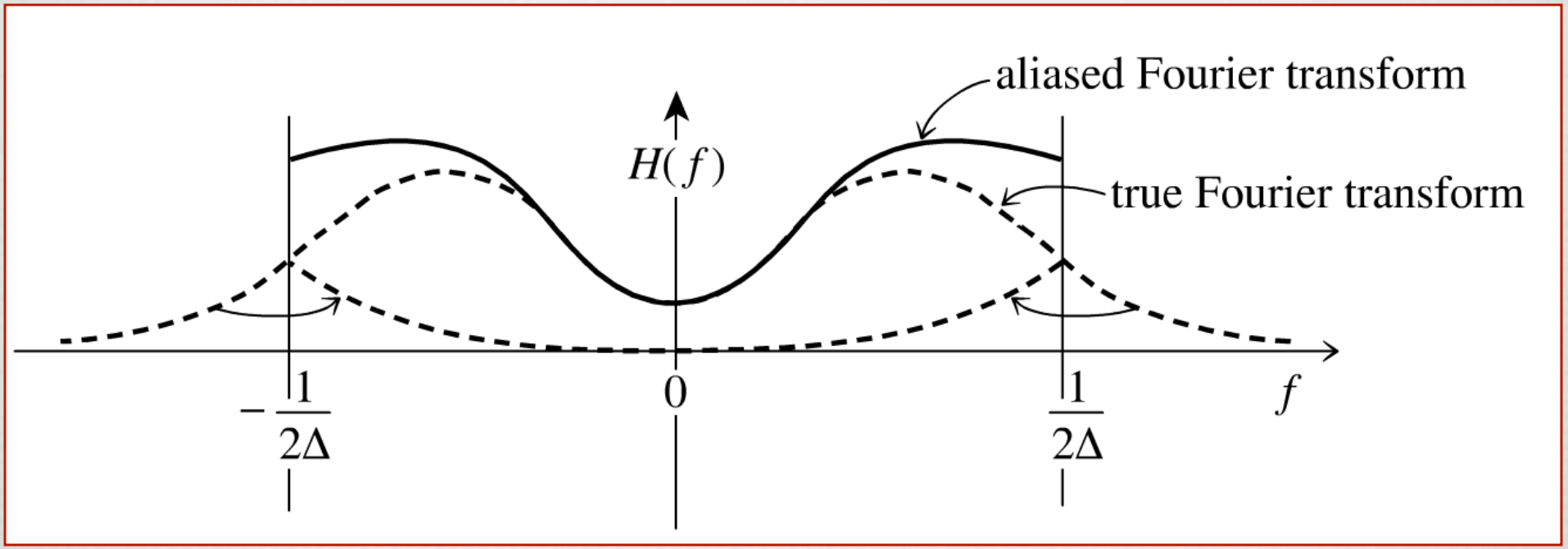
If $\Delta t > T_0/2$ (blue curve), the actual measured frequency is a fraction of what it should be (Δt_{blue}= 1 second and the highest representable frequency is f_{NYQ}=0.5)

But the frequency of the signal is 1/1.111=0.9 Hz
The signal seems to have a frequency of 1/10=0.1Hz, due to aliasing)
This can be seen as a frequency folding

FOLDING (SAMPLING RATE: 1S)



ALIASING IS EQUIVALENT TO FOLDING



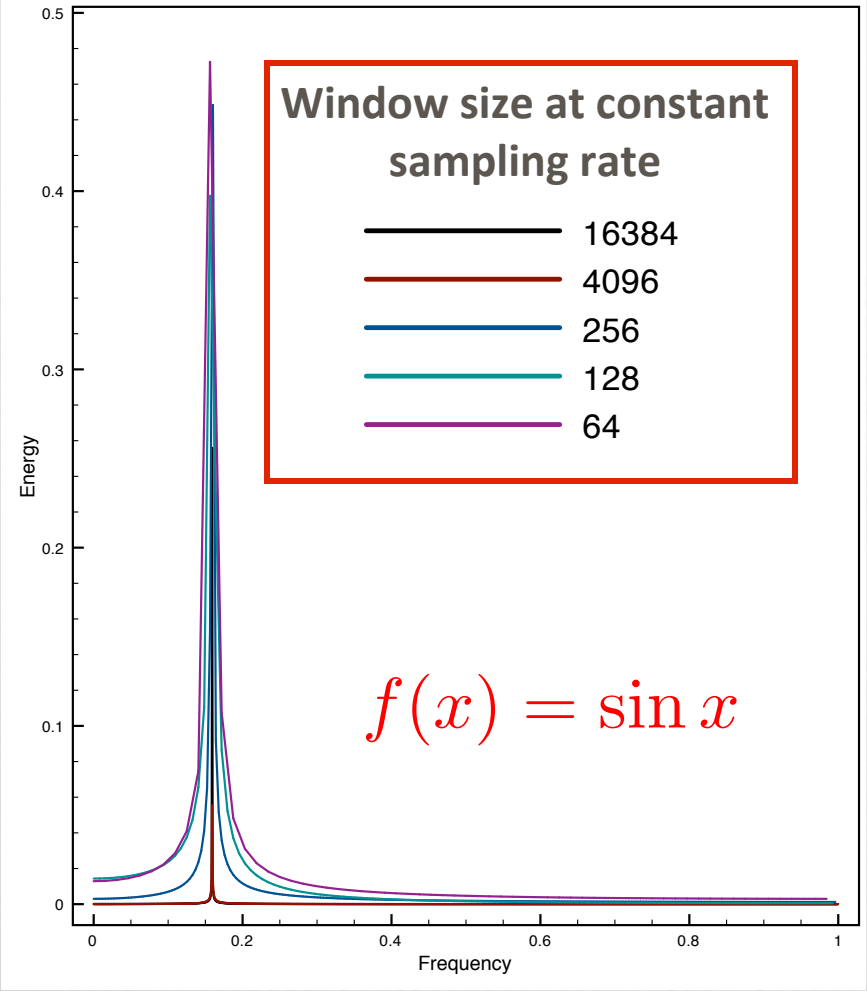
HOW TO AVOID ALIASING?

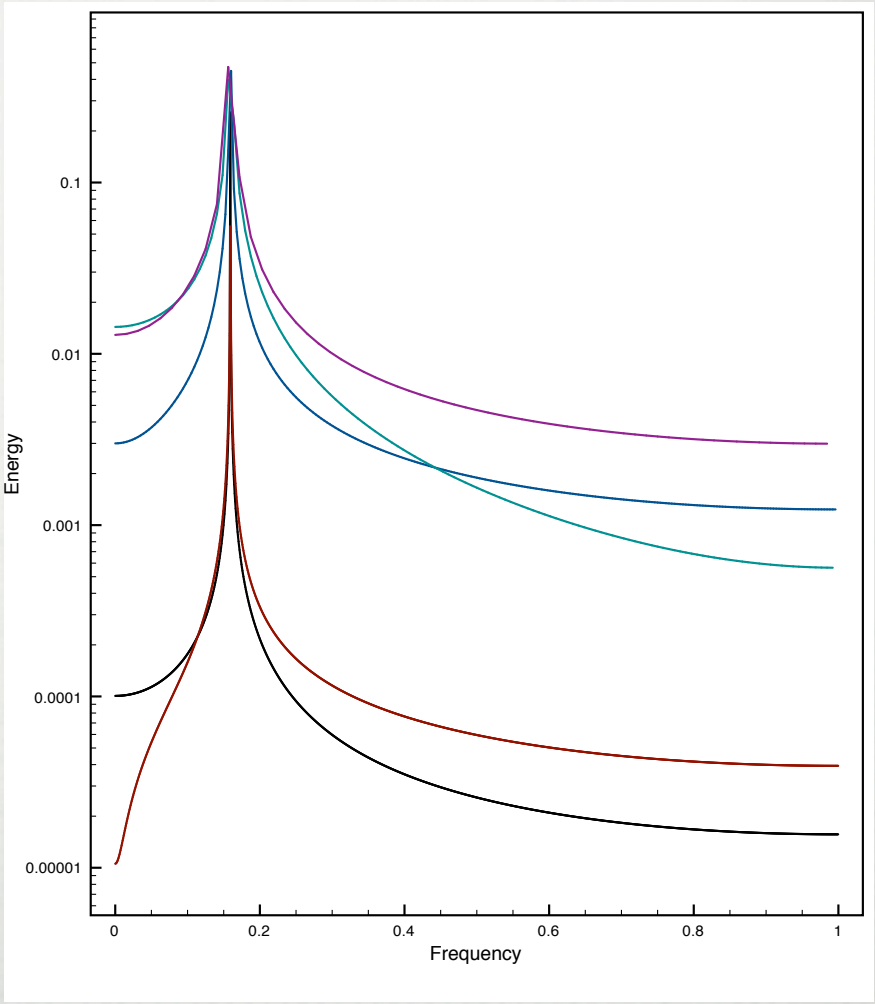
Solution: Increase sample rate; decrease Δt

LEAKAGE

- The Fast Fourier Transform is commonly used because it requires much less processing power than the Fourier Transform.
- The signal must be periodic in the sample window or leakage will occur.
- Leakage is the smearing of energy from the true frequency of the signal into adjacent frequencies.
- Leakage also causes the amplitude representation of the signal to be less than the true amplitude of the signal.

LEAKAGE: EXAMPLE



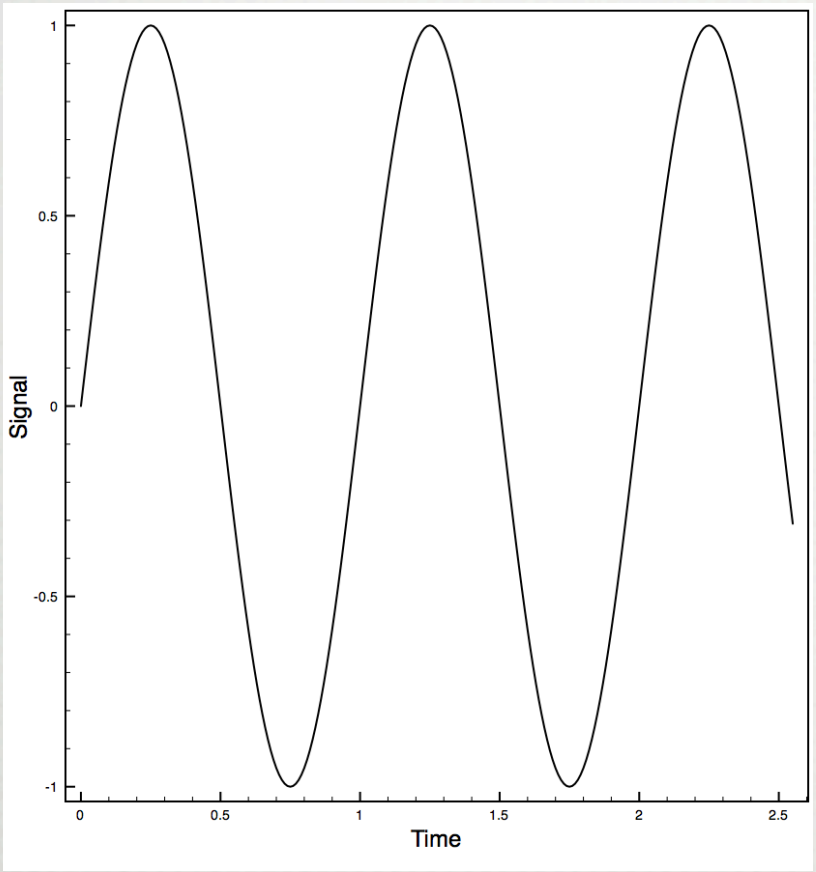


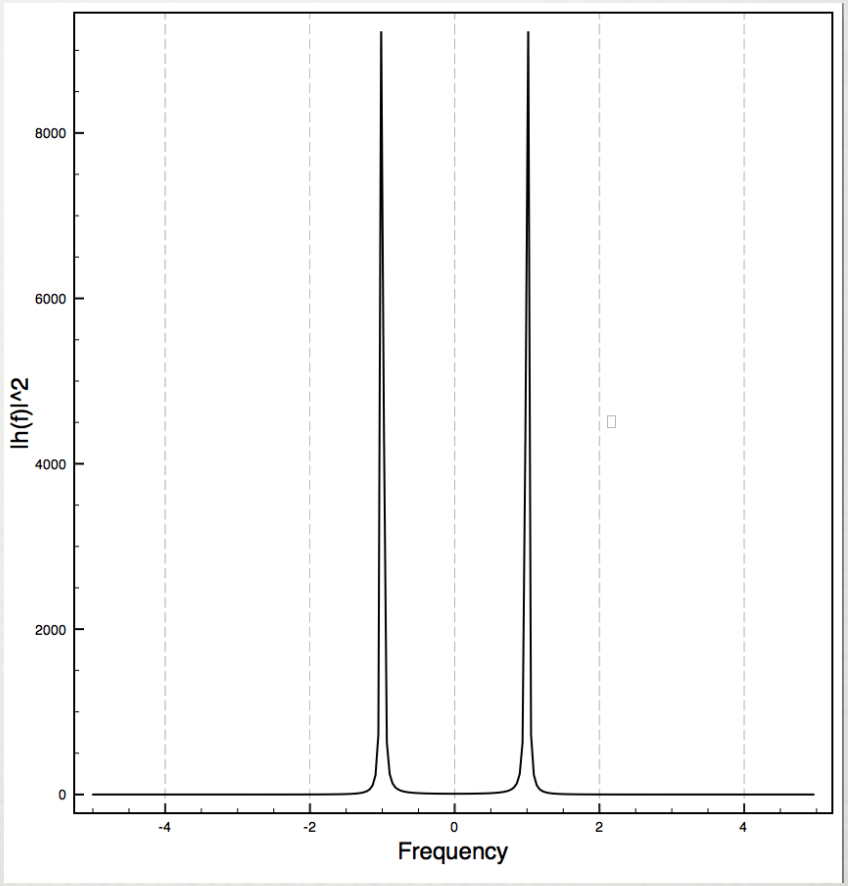
Single frequency at 1/2π = 0.159...

HOW TO AVOID LEAKAGE?

Solution: Increase sample time

EXAMPLES I. SIMPLE SINE





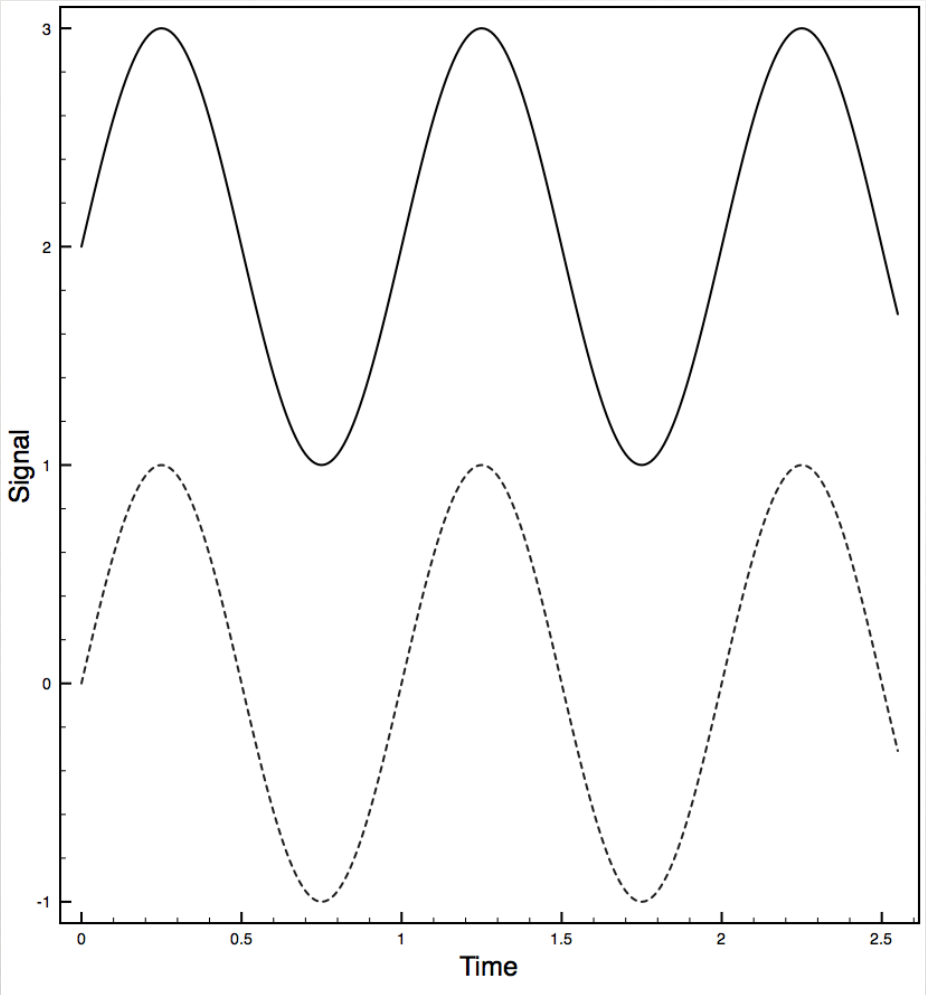
$f(t) = \sin 2\pi t$

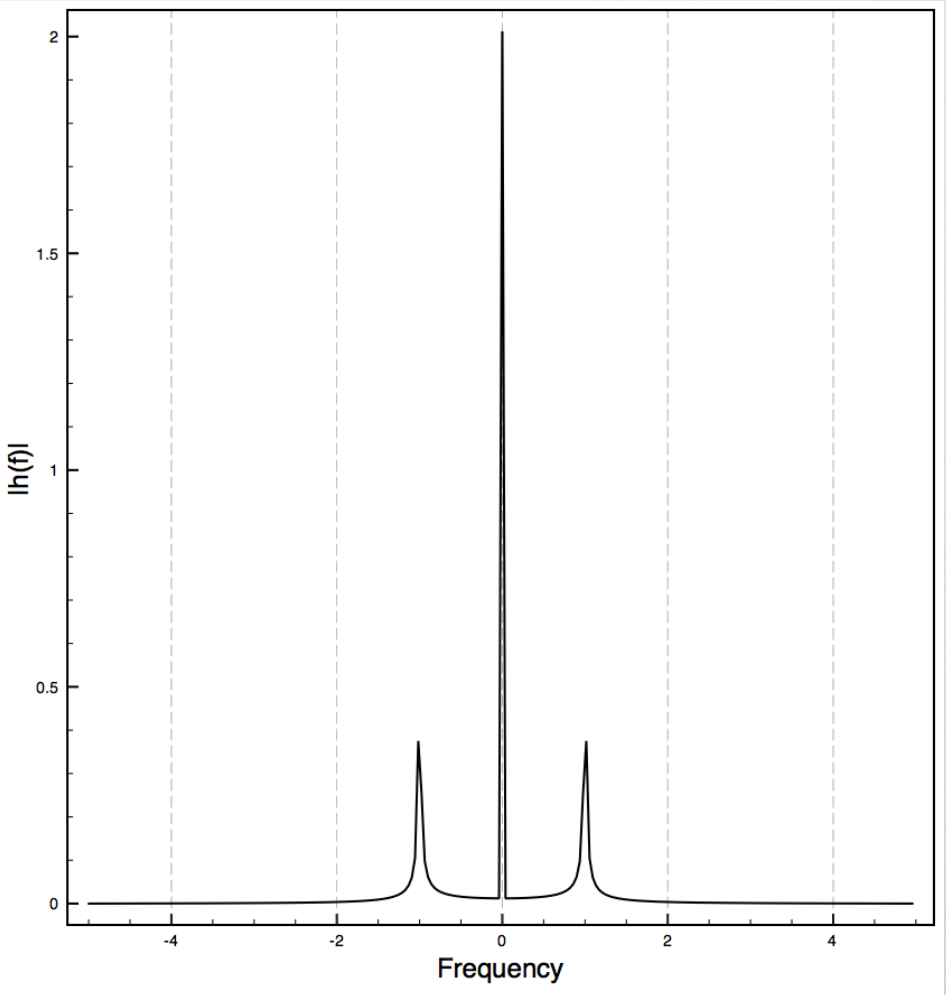
Highest frequency: 1

$\Delta t = 0.1 \rightarrow f_{nyq} = 5$

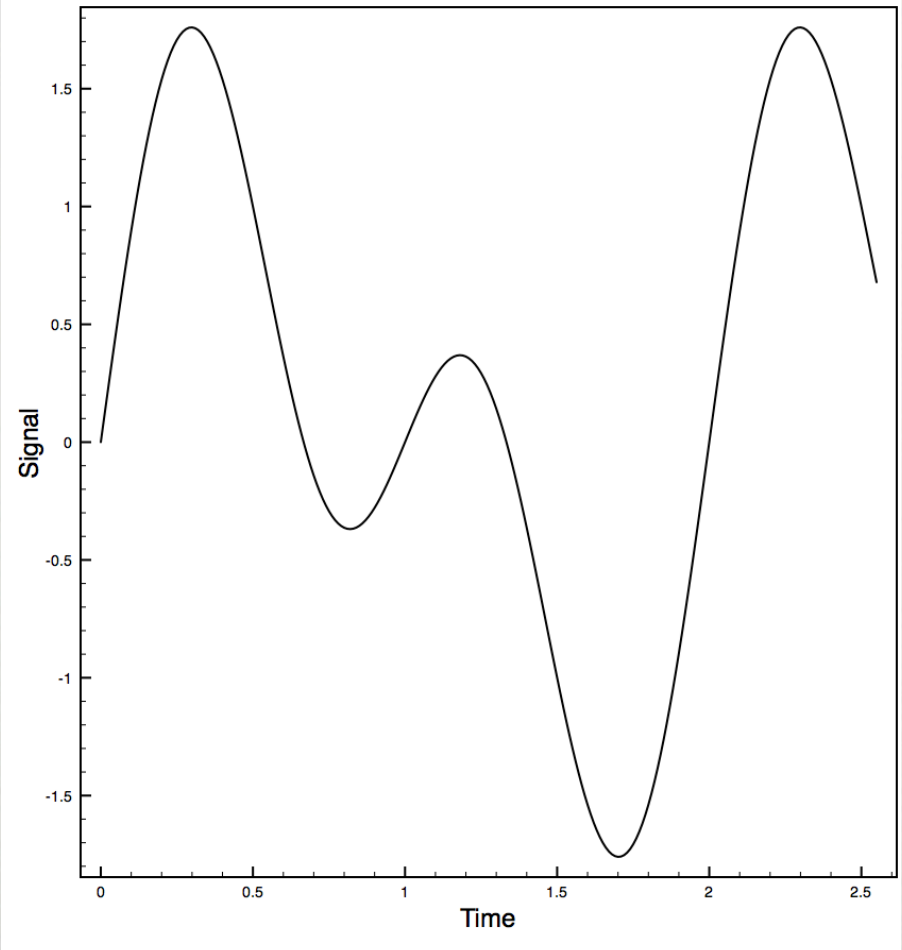
$N = 256$

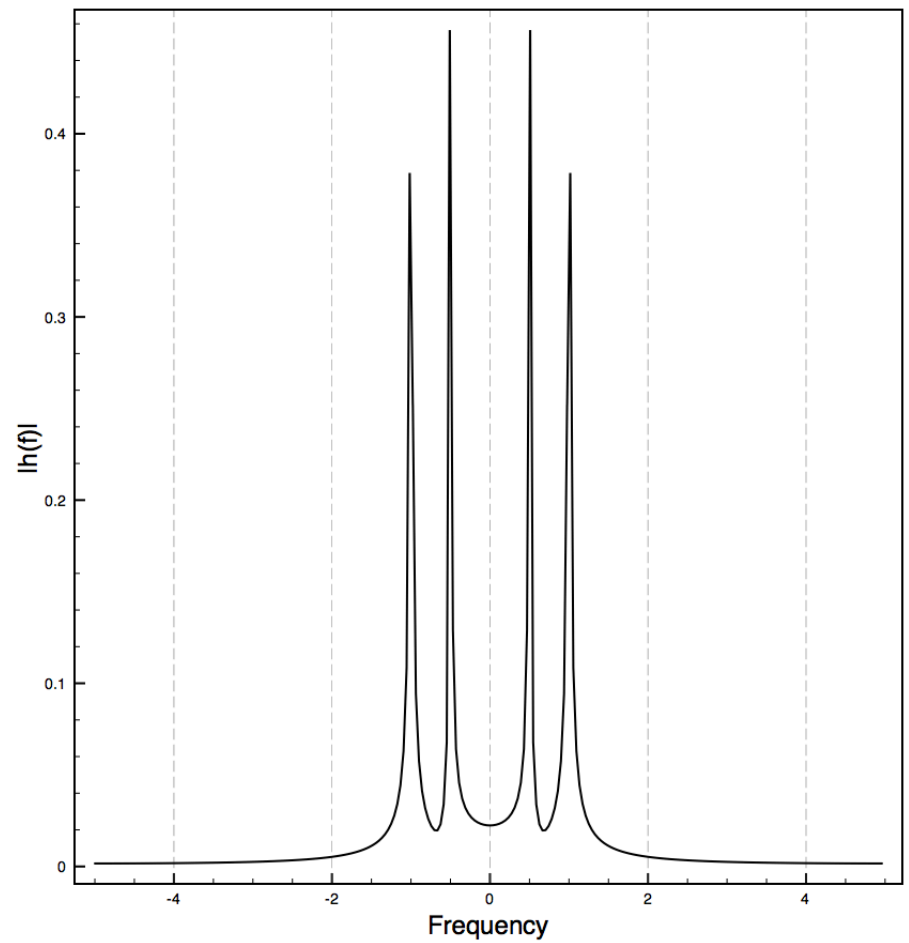
EXAMPLES
1B. SIMPLE SINE + OFFSET





EXAMPLES: 2. TWO SINES

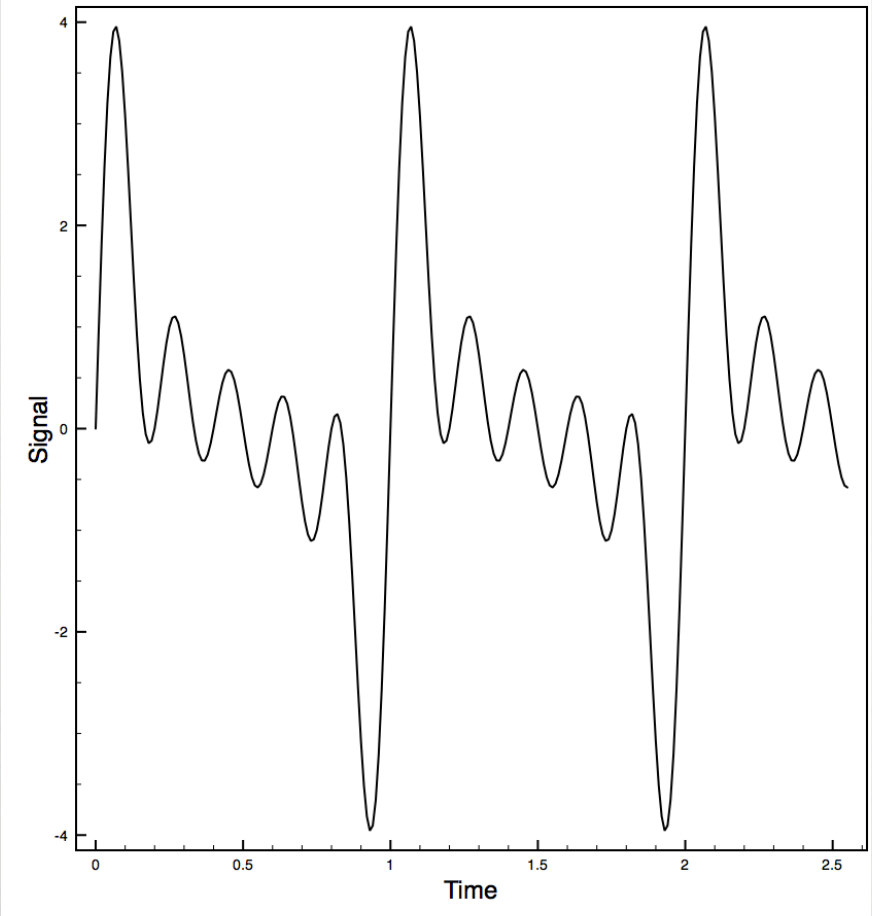


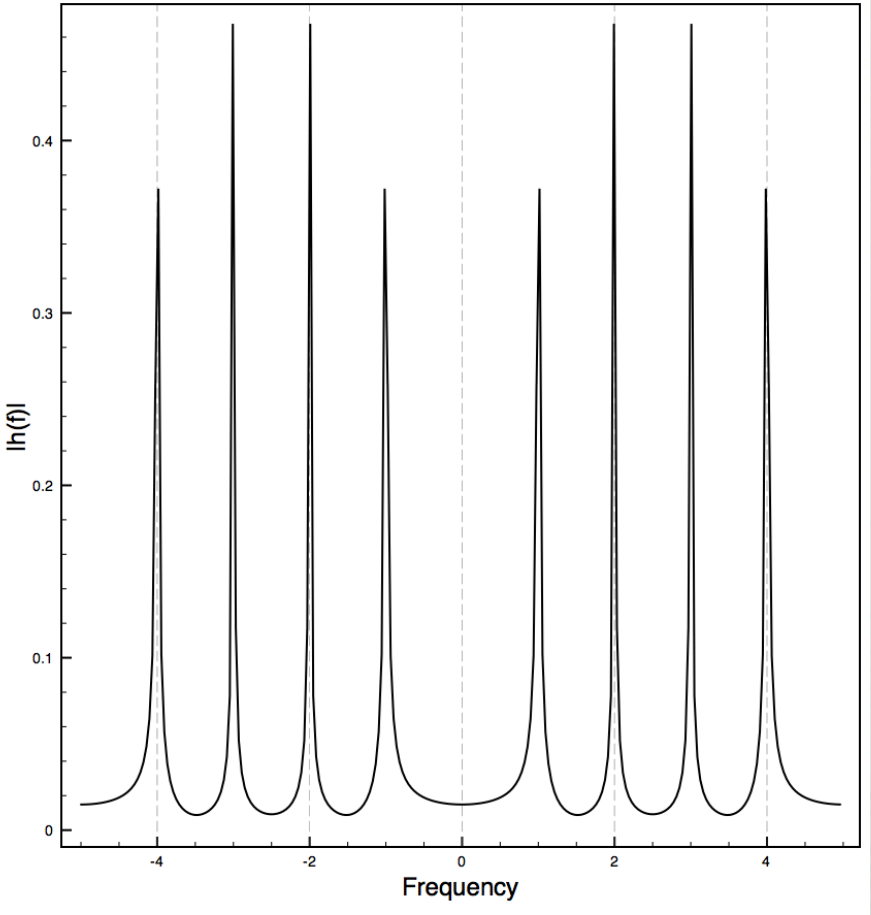


$$f(t) = \sin 2\pi t + \sin \pi t$$

$$\omega_{1,2} = 1, 1/2$$

EXAMPLE 3: 5 SINES

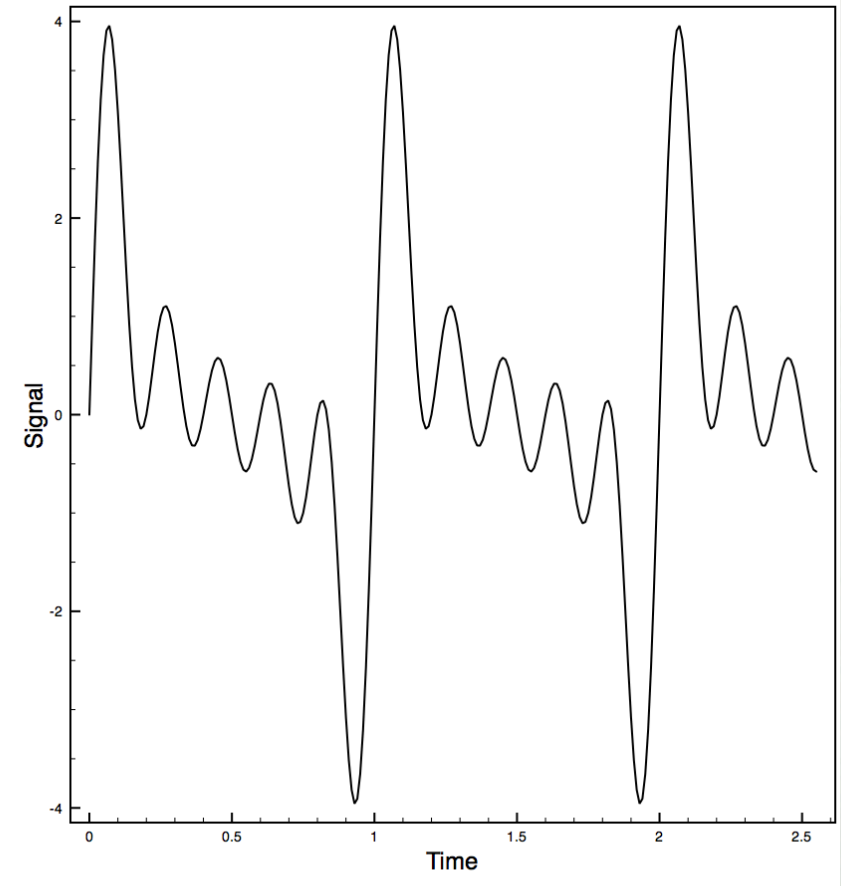


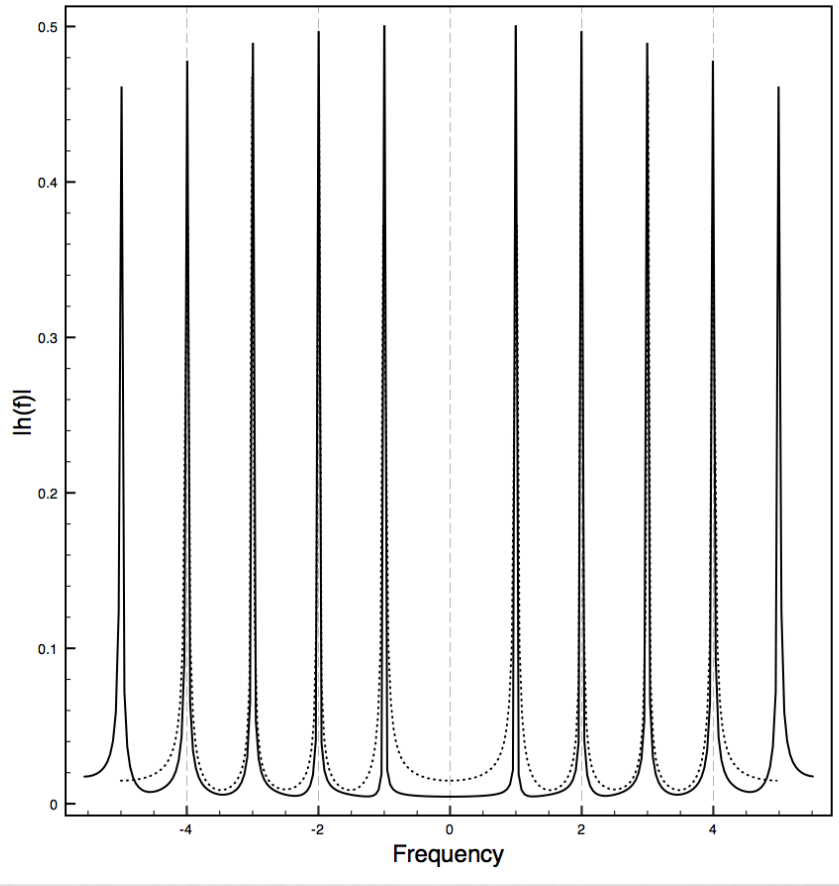


$$f(t) = \sum_{i=1,5} \sin 2j\pi t$$

$$\omega_j = j$$

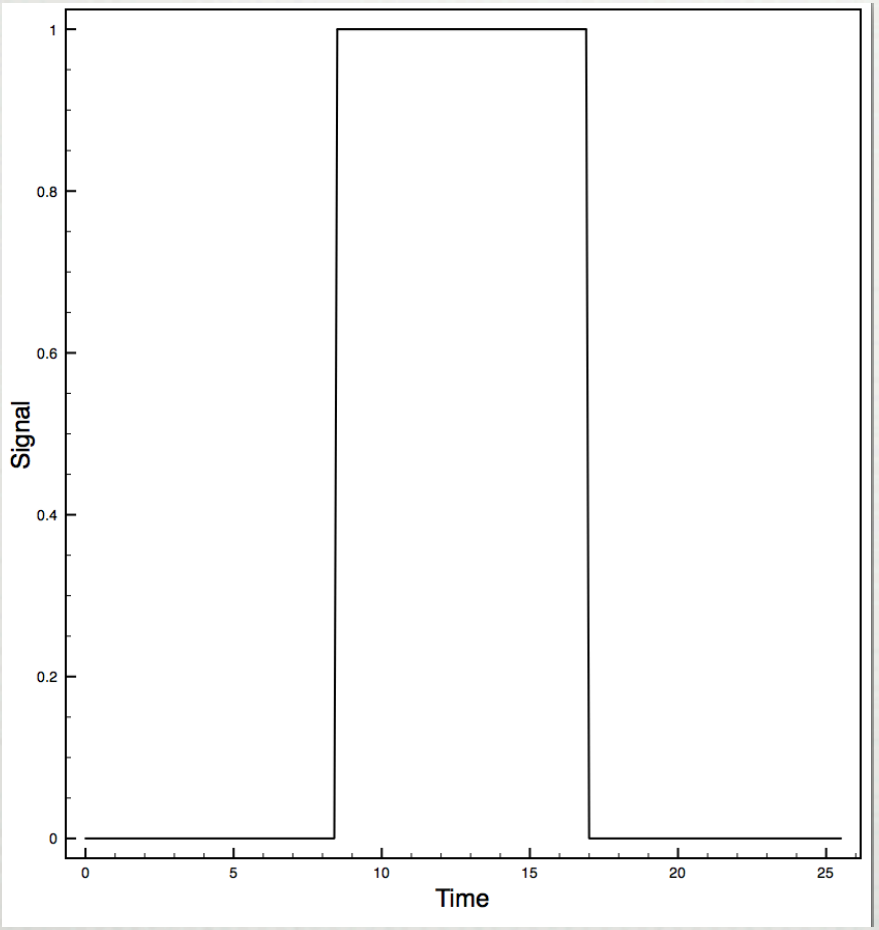
EXAMPLE 3B: 5 SINES; REDUCING ALIASING

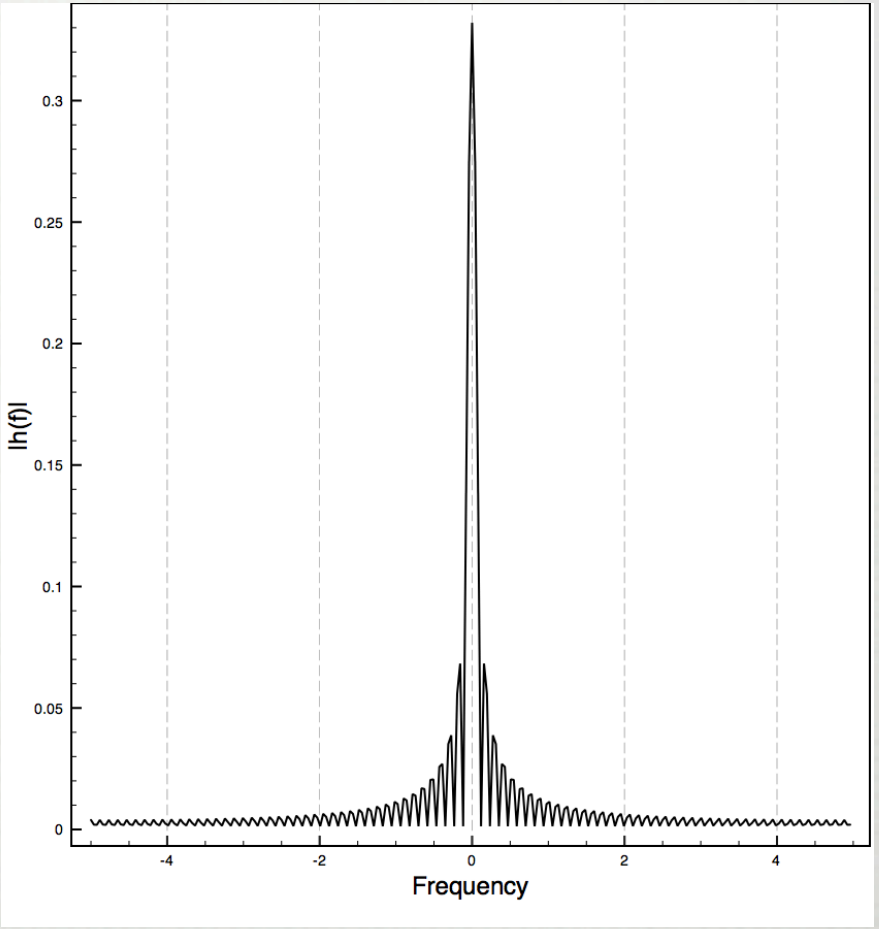




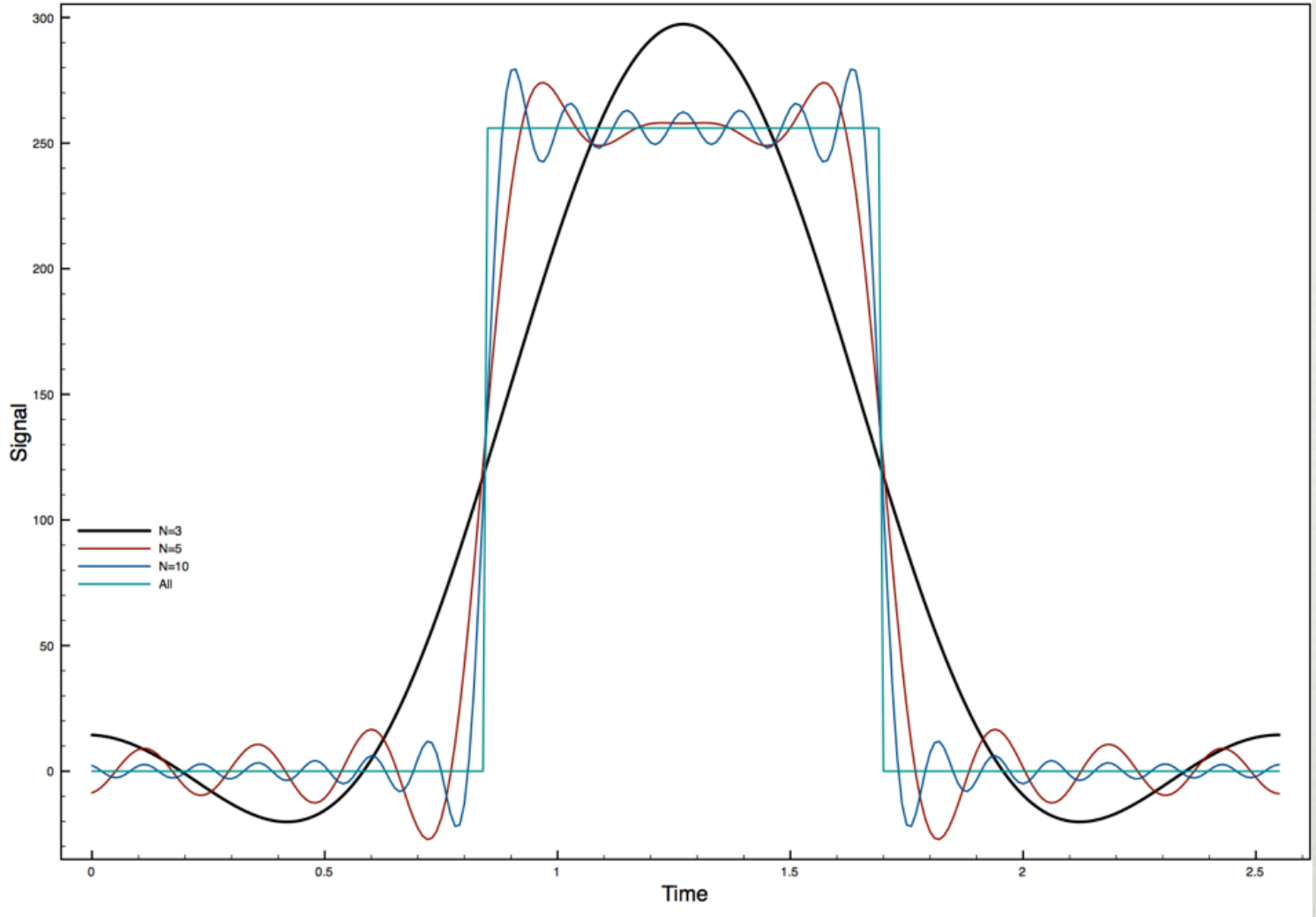
$$\Delta t = 0.09 \Rightarrow \omega_{\mathrm{NYQ}} = (2\Delta t)^{-1} = 5.\bar{5}$$

EXAMPLE 4: STEP FUNCTION



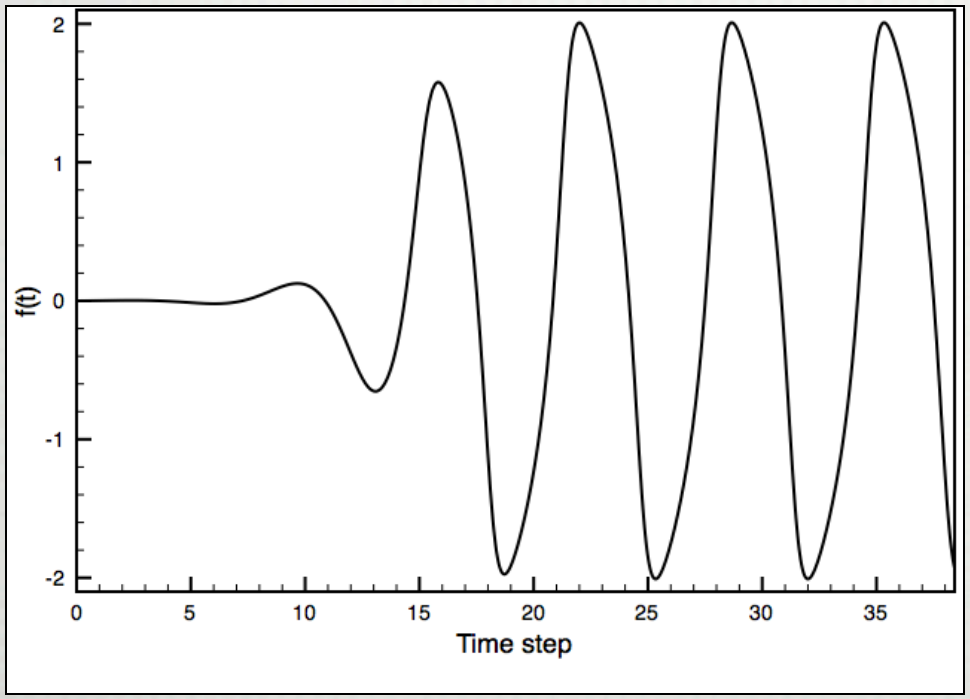


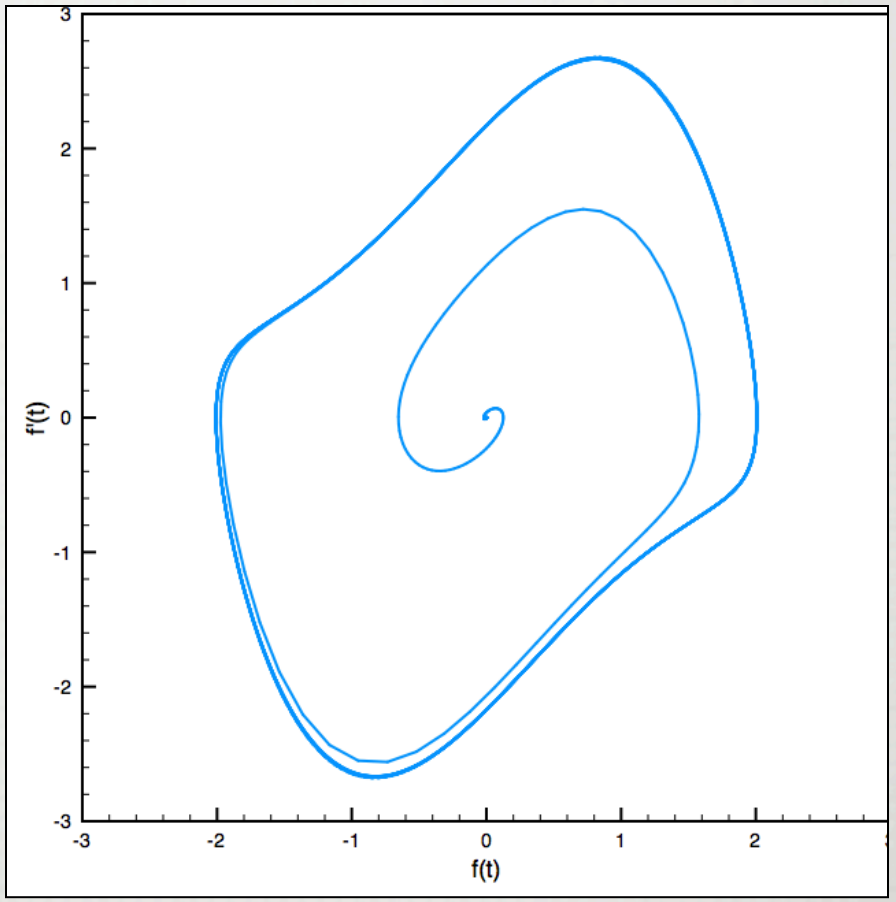
RECONSTRUCTING SIGNAL



VAN DER POL OSCILLATOR (MU=1)

$$y'' - \mu(1 - y^2)y' + y = 0$$

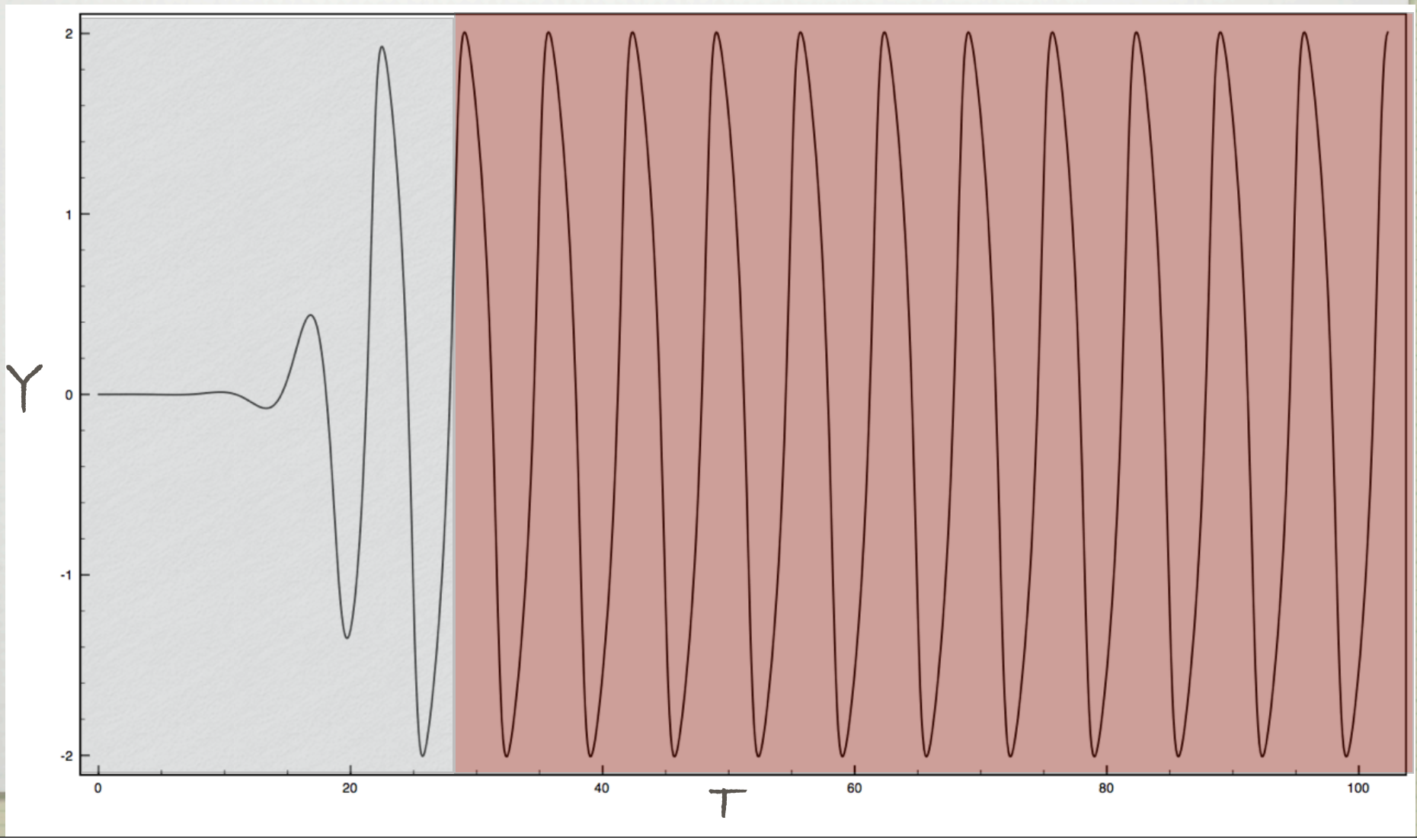




It is an equation describing self-sustaining oscillations in which energy is fed into small oscillations and removed from large oscillations. This equation arises in the study of circuits containing vacuum tubes.

NON-LINEAR OSCILLATOR

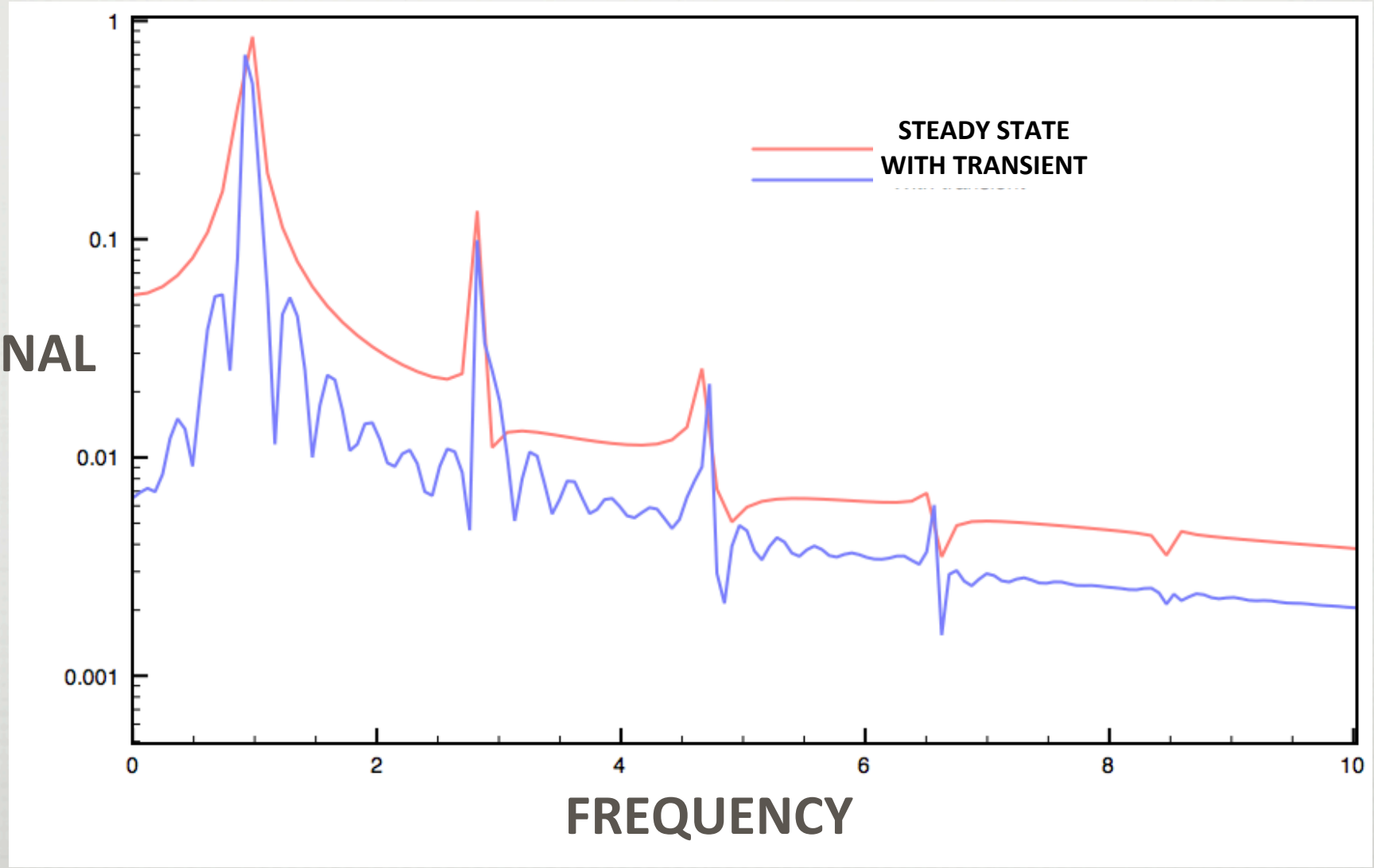
TRANSIENT STEADY STATE



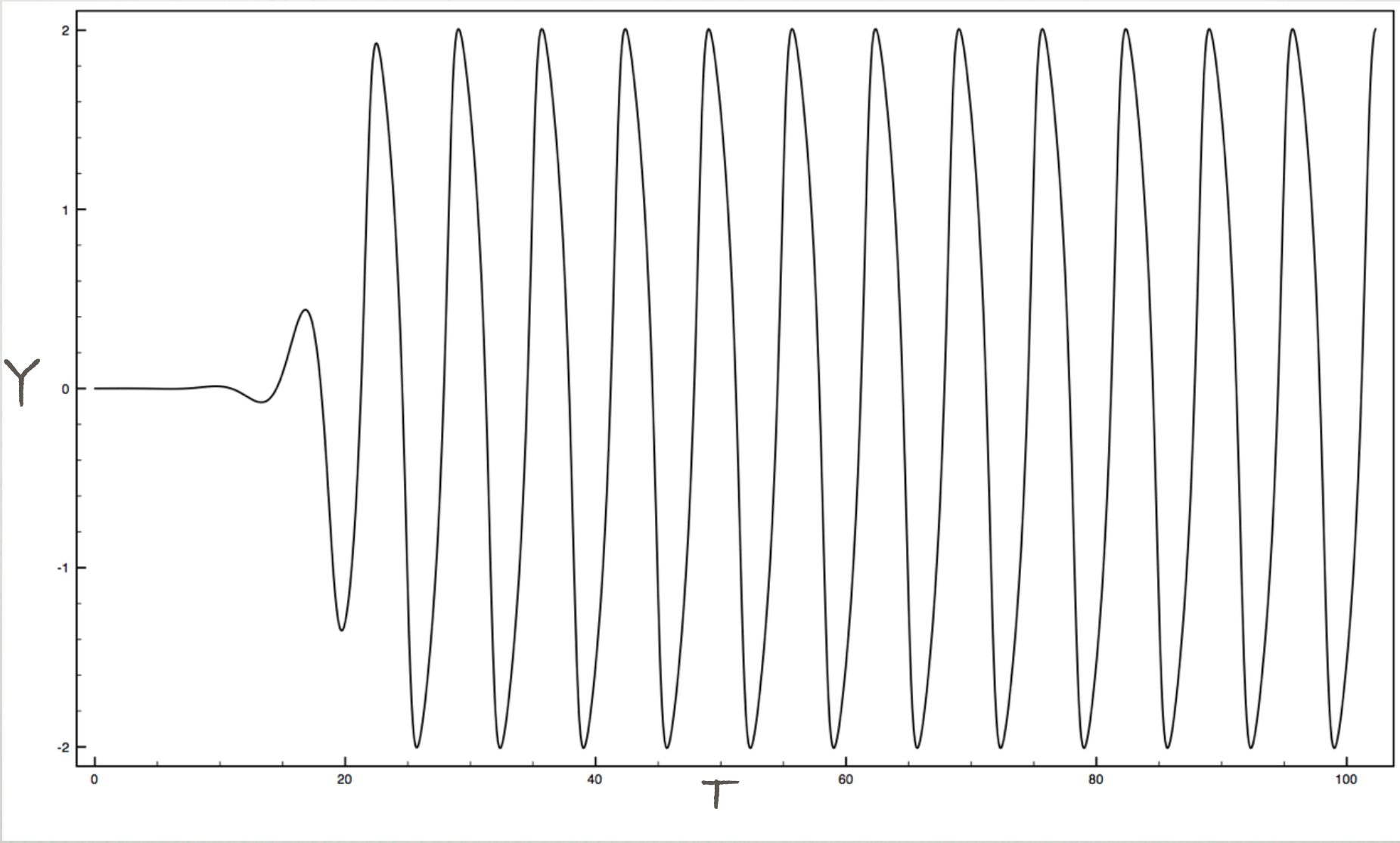
FREQUENCY ANALYSIS

- **Real signal: positive frequency is enough (symmetrical)**
- **Maximum frequency: Nyquist frequency**
- **What is the frequency content of that signal?**

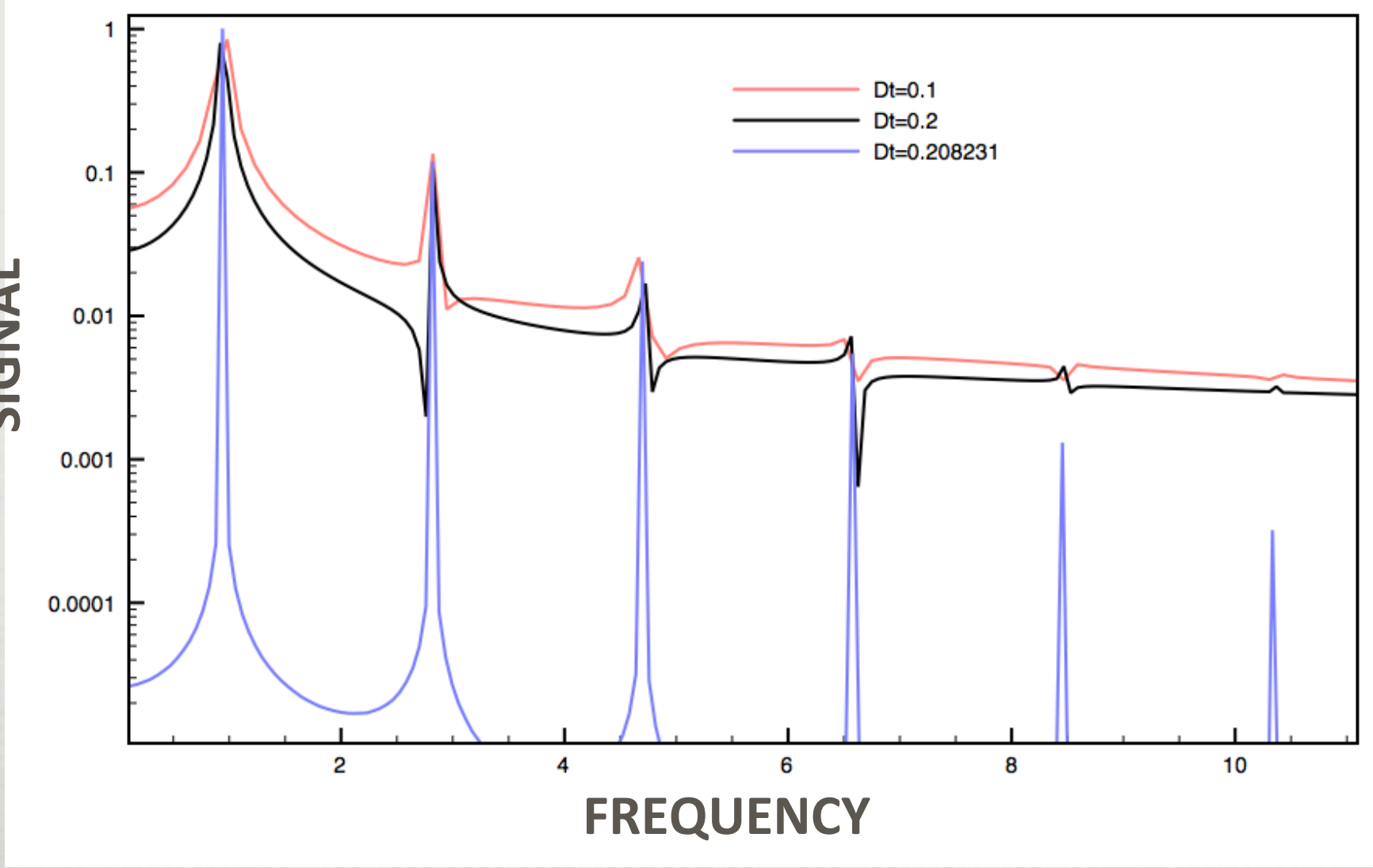
FOURIER ANALYSIS ($\Delta t=0.1, N=1024$)



NON-LINEAR OSCILLATOR



IMPROVEMENT: REDUCING LEAKAGE



YOUR TURN

- ☐ **Modify your RK code and**
- ☐ **Write a FFT procedure to**
- ☐ **Study the anharmonic oscillator and study the frequency content.**