



# Survey of Scientific Computing (SciComp 301)

Dave Biersach  
Brookhaven National  
Laboratory  
[dbiersach@bnl.gov](mailto:dbiersach@bnl.gov)



```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9
10 namespace SimpleEvents
11 {
12     public partial class Form1 : Form
13     {
14         Person person = new Person();
15
16         public Form1()
17         {
18             InitializeComponent();
19             person.FirstName = "Christian";
20             person.LastName = "Pano";
21         }
22
23         private void button1_Click(object sender, EventArgs e)
24         {
25             person.MainColor = textBox1.Text;
26         }
27     }
28 }
```

**Session 21**  
Fourier Transform,  
Signals Analysis

# Session Goals

- Perform **Fourier Analysis** using the Discrete Fourier Transform (**DFT**)
- Perform **Fourier Synthesis** using the Inverse Discrete Fourier Transform (**IDFT**)
- Remove high-frequency noise from a signal and investigate deep space signals received at the **Arecibo Radio Observatory**
- Determine the fundamental frequency of **Sun Spot activity**

# Know the Greek Alphabet

Αα

ALPHA

Ββ

BETA

Γγ

GAMMA

Δδ

DELTA

Εε

EPSILON

Ζζ

ZETA

Ηη

ETA

Θθ

THETA

Ιι

IOTA

Κκ

KAPPA

Λλ

LAMBDA

Μμ

MU

Νν

NU

Ξξ

XI

Οο

OMICRON

Ππ

PI

Ρρ

RHO

Σσ

SIGMA

Ττ

TAU

Υυ

UPSILON

Φφ

PHI

Χχ

CHI

Ψψ

PSI

Ωω

OMEGA

# Know the Double Struck Letters

$\mathbb{N}$  = Natural Numbers (1, 2, 3)

$\mathbb{Z}$  = Integers (-3, -2, -1, 0, 1, 2, 3)

$\mathbb{Q}$  = Rational Numbers ( $\mathbb{Z}/\mathbb{Z}$ )

$\mathbb{R}$  = Real Numbers (decimals)

$\mathbb{C}$  = Complex Numbers (Re + Im)

Think Q for  
quotient

$$\mathbb{N} \in \mathbb{Z} \in \mathbb{Q} \in \mathbb{R} \in \mathbb{C}$$

# All of Physics *is* Waves

- Electrical
- Magnetic
- Acoustic
- Heat Flow
- Vibrational
- Torsional
- Nuclear / Quantum
- Gravitational
- Oceanic / Tidal
- Orbital Precession
- Springs
- Pendulums
- Image Reconstruction
- Stock Market
- Economics
- Astronomical
- Fluid Dynamics
- Earthquakes
- AC / DC
- AM / FM
- Speech
- Heartbeats

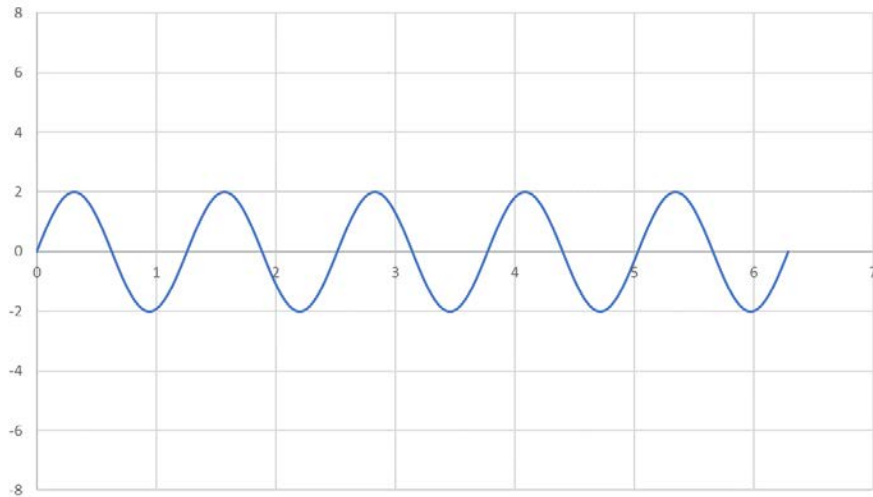
It is really important that you develop a keen understanding of the mathematics of waves!

# Superposition $\Rightarrow$ Perceived Complexity

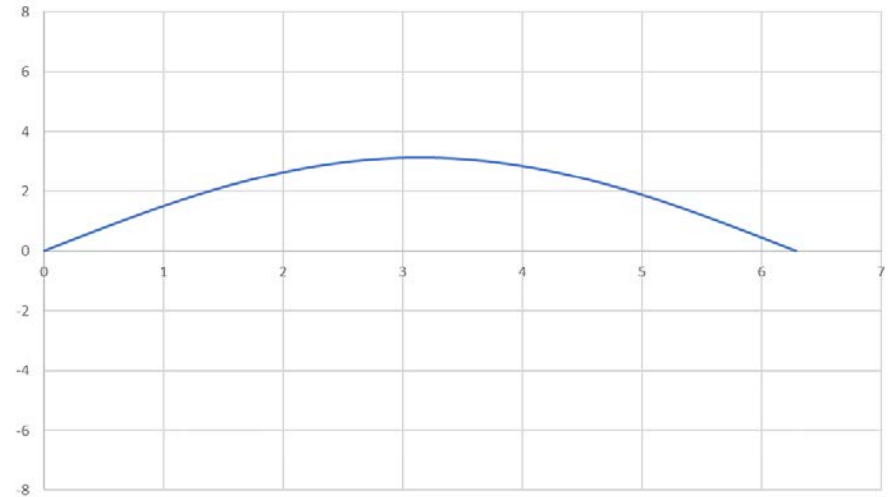
- What appears to be a **complex** waveform might actually be just a series of **simple** waves all added together (linear superposition)
- Underneath the perceived **random** behavior of a function undergoing wild fluctuations might be a system of straightforward waves
- These simple waves may individually convey **some important knowledge** about the true nature of the observed complexity
- The process of determining the underlying simple waveforms for a complex wave is called **Fourier Analysis**

# Superposition $\Rightarrow$ Perceived Complexity

Wave 1:  $2\sin(5x)$



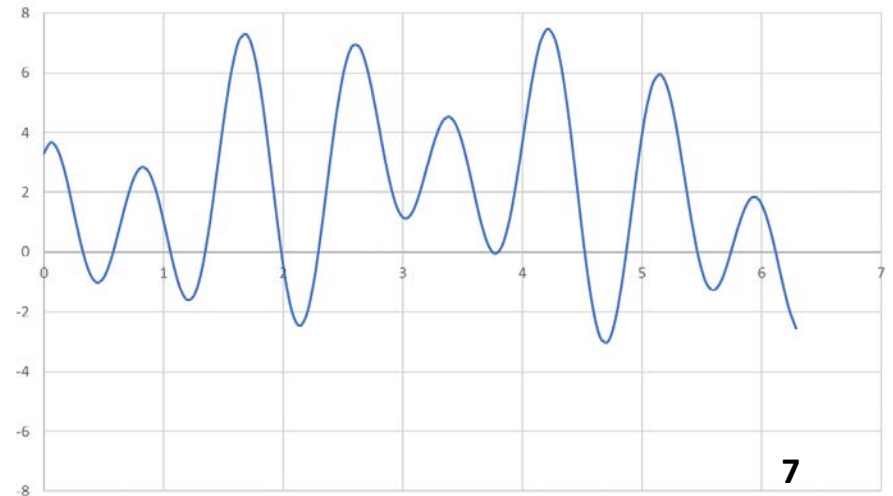
Wave 3:  $\pi\sin(x/2)$



Wave 2:  $\sqrt{11}\cos(e^{2x})$



Wave 1 + 2 + 3

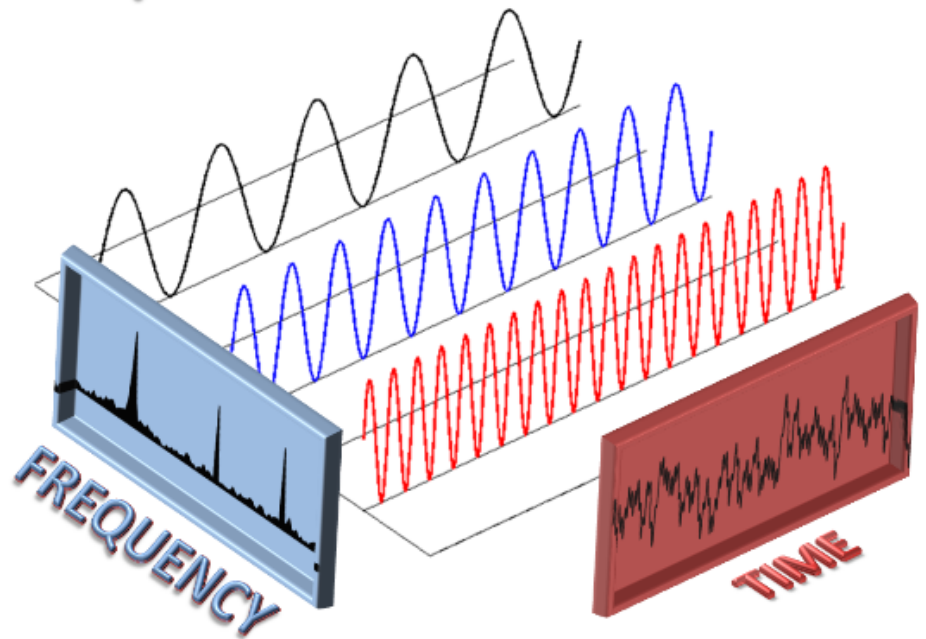




**Jean-Baptiste Fourier  
(1768-1830)**

*both r's are silent  
"Foo-yeah"*

## Fourier Transform Continuous and Discrete



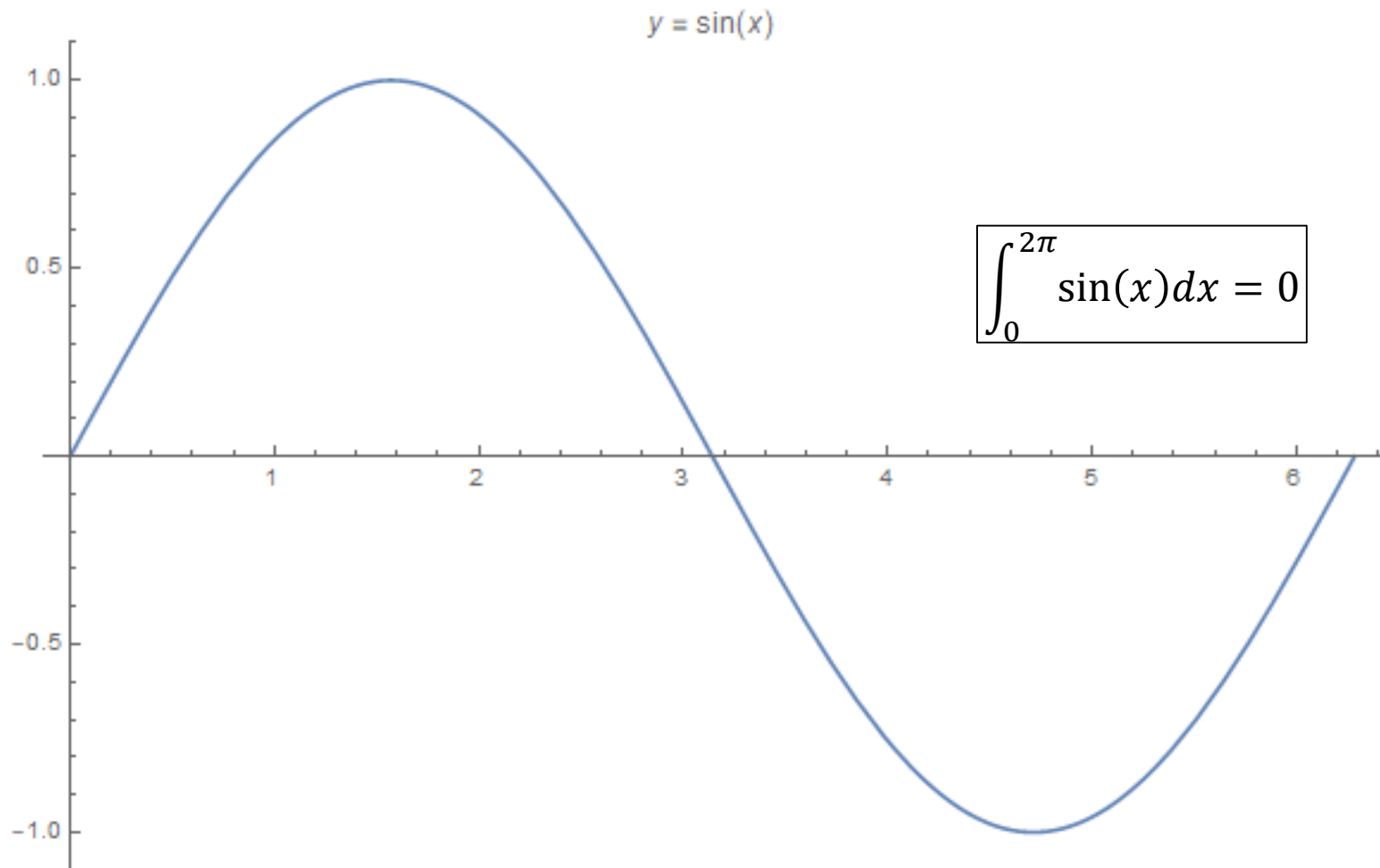


# Fourier Analysis

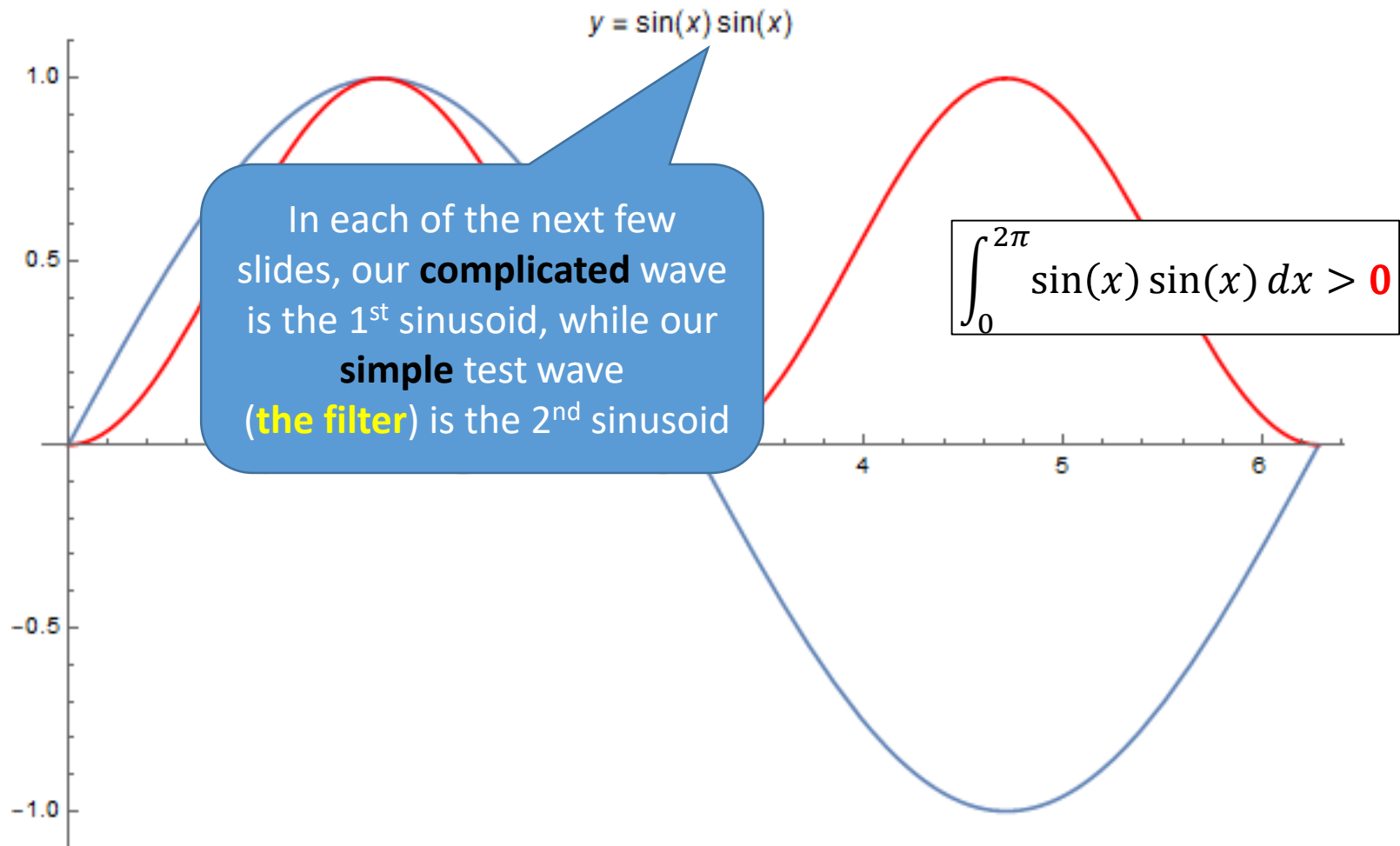
- Converting a complicated wave into a series of simple waves having different amplitudes but *integer* frequencies is called the **Fourier Transform**
- The reverse process of reconstructing the original complicated wave by summing all the contributions of the simple waves is called the **Inverse Fourier Transform**
- It is a **transform** because we are converting back and forth between representing a wave as a sum of amplitudes over time *versus* a sum of amplitudes over integer frequencies
- It is **two sides of the same mirror** – both approaches equally describe the same wave – but the *frequency* view often reveals hidden patterns in the wave

# Fourier Analysis - Intuition

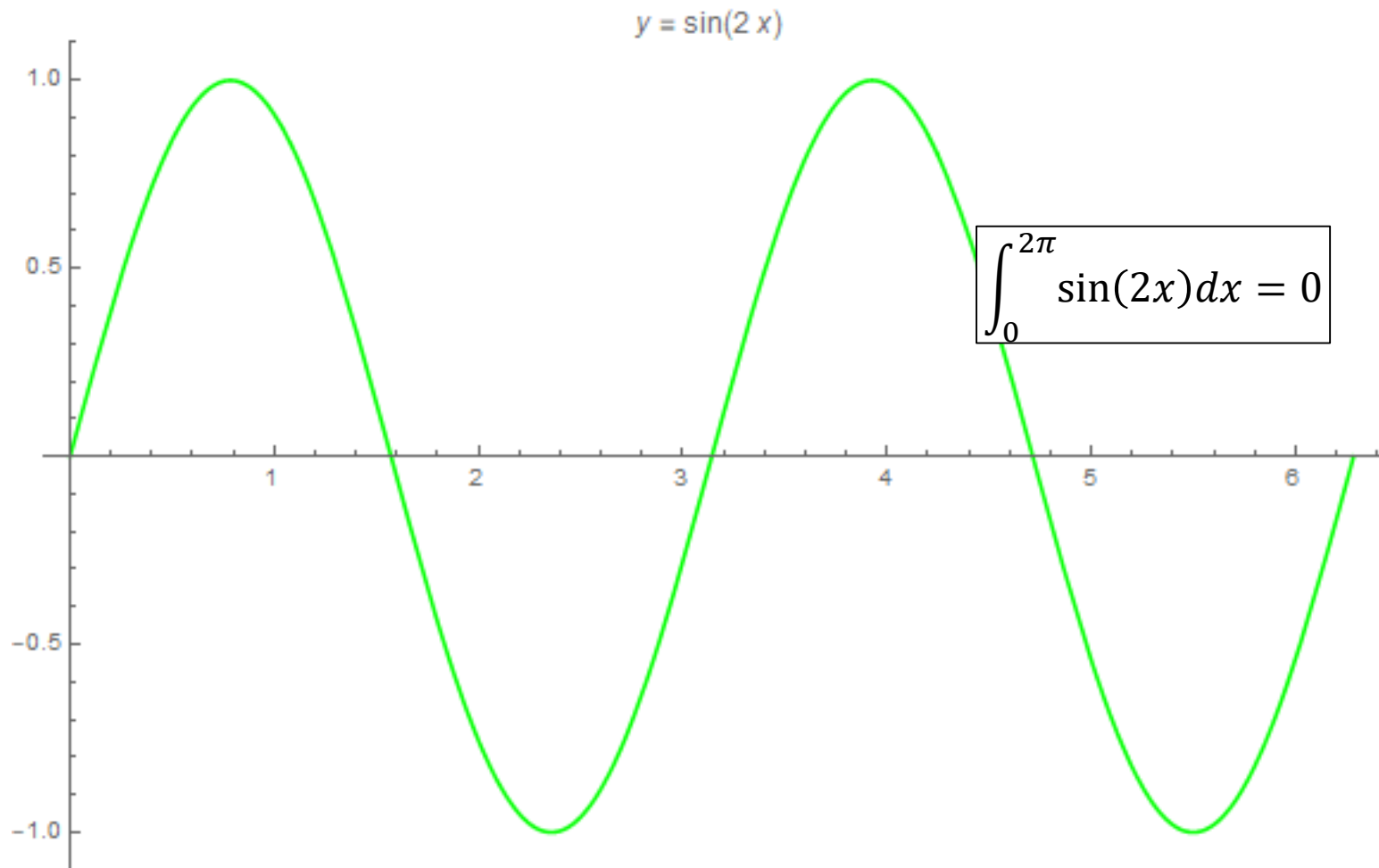
$$0 \leq x < 2\pi$$



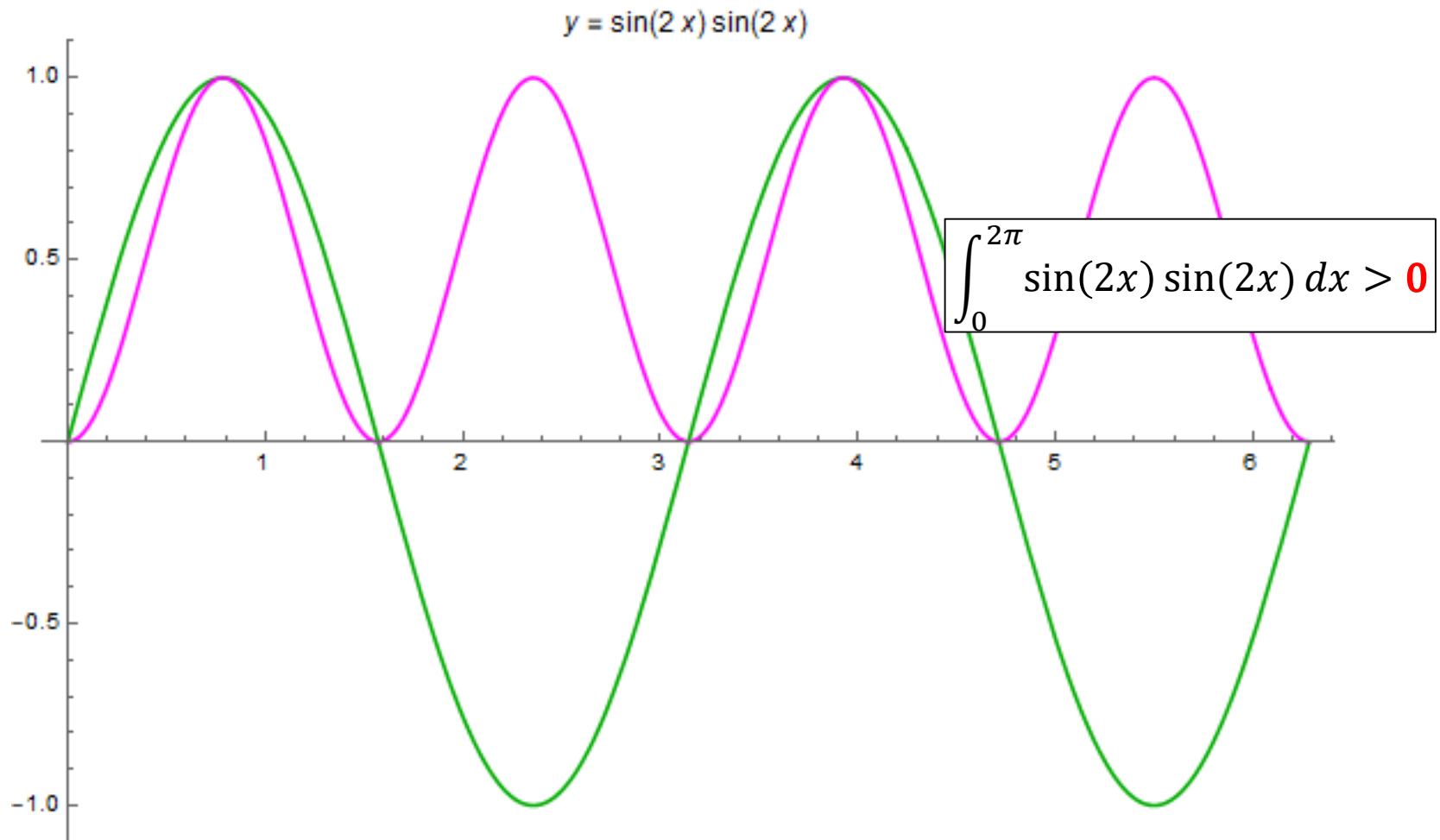
# Fourier Analysis - Intuition



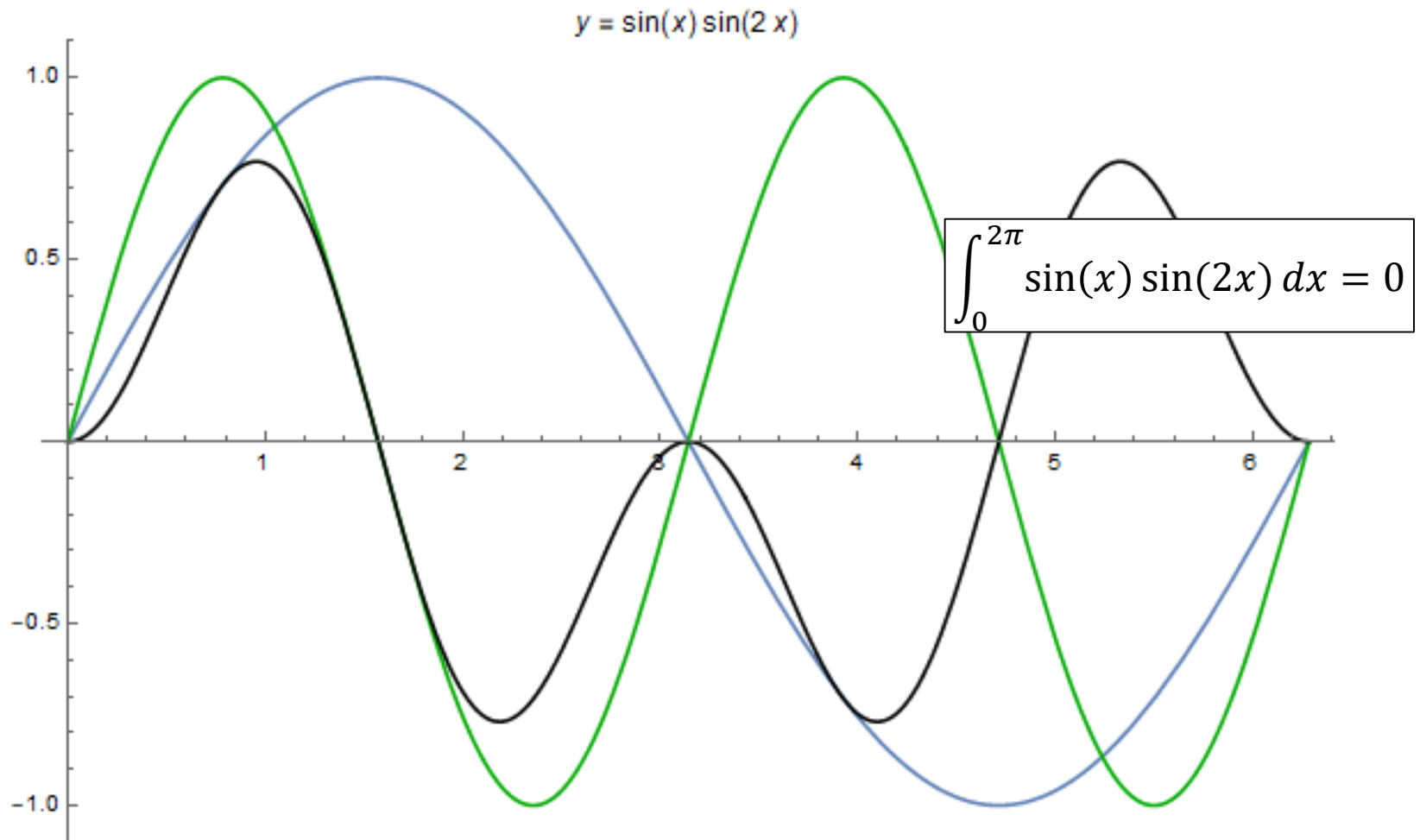
# Fourier Analysis - Intuition



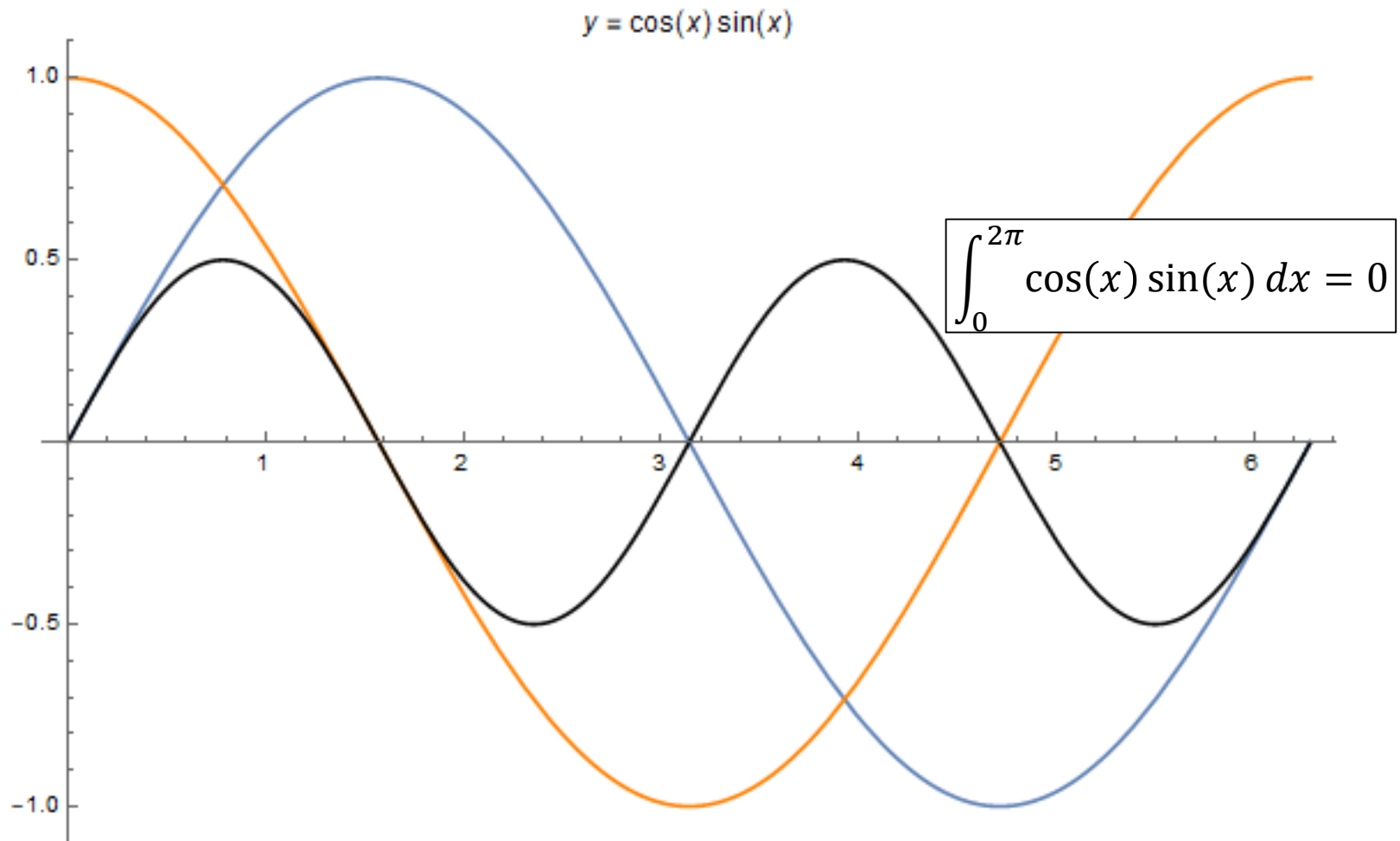
# Fourier Analysis - Intuition



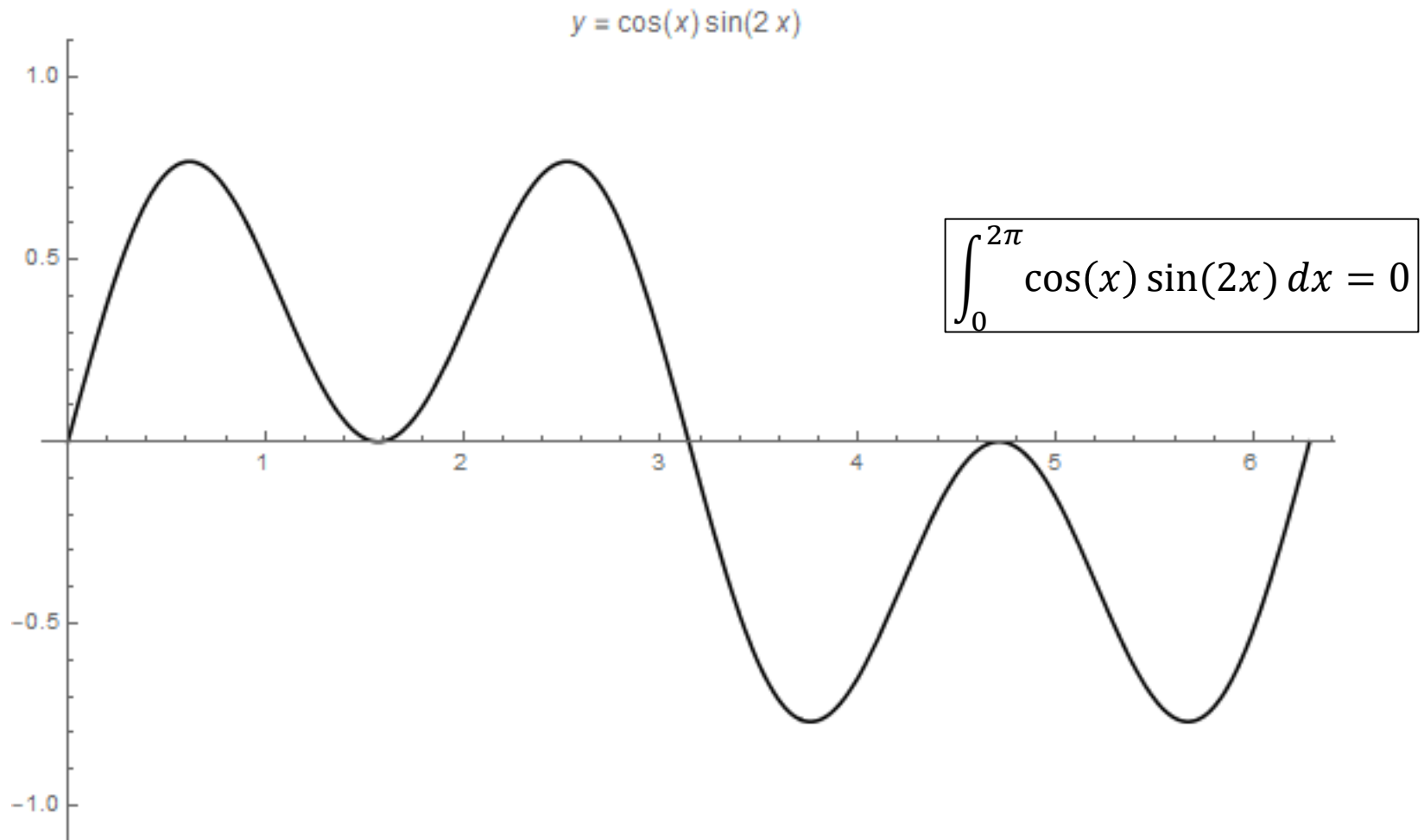
# Fourier Analysis - Intuition



# Fourier Analysis - Intuition



# Fourier Analysis - Intuition





# Fourier Analysis - Intuition

$$\int_0^{2\pi} \sin(kt) \sin(nt) dt \quad \begin{array}{l} > 0 \text{ when } n = k \\ = 0 \text{ when } n \neq k \end{array}$$

$$\int_0^{2\pi} \cos(kt) \cos(nt) dt \quad \begin{array}{l} > 0 \text{ when } n = k \\ = 0 \text{ when } n \neq k \end{array}$$

$$\int_0^{2\pi} \cos(kt) \sin(nt) dt \quad \begin{array}{l} = 0 \text{ when } n = k \\ = 0 \text{ when } n \neq k \end{array}$$

The only time the integral of the product of two sinusoids is **> 0** is when **both waves have matching frequency and phase**

Only the simple waves (each having an increasing integer frequency) that yield a **non-zero integral** when **multiplied against the complicated wave** can be actual components of the original wave

# Discrete Fourier Analysis

- Converting a sampled complicated wave into a series of **simple** waves, each with different amplitudes at specific integer frequencies, is called the **Discrete Fourier Transform (DFT)**
- The reverse process of reconstructing the original complex wave by summing all of the contributing simple wave forms is called the **Inverse Discrete Fourier Transform (IDFT)**
- The **sum** of many *discrete* samples of a wave approximates its *continuous* **integral** if the spacing (time) between samples is sufficiently small – this is analogous to a **Riemann sum**

# Discrete Fourier Transform (DFT)

## DFT (real)

$$y(t_s) \approx \sum_{n=0}^{\text{terms}} [A_n \cos(nt_s) + B_n \sin(nt_s)]$$

$$A_0 = \frac{\sum_{s=0}^{\text{samples}} y(t_s)}{\text{samples}}$$

$$A_n = \frac{2 \sum_{s=0}^{\text{samples}} y(t_s) \cos(nt_s)}{\text{samples}}$$

$$B_n = \frac{2 \sum_{s=0}^{\text{samples}} y(t_s) \sin(nt_s)}{\text{samples}}$$

$n = \text{term number } \{n \in \mathbb{Z}^+\}$

## Euler's Formula

$$e^{j\omega t} = \cos \omega t + j \sin \omega t$$

$$\cos \omega t = \frac{1}{2} e^{j(-\omega)t} + \frac{1}{2} e^{j\omega t}$$

$$\sin \omega t = \frac{1}{2} e^{j(-\omega)t} - \frac{1}{2} e^{j\omega t}$$

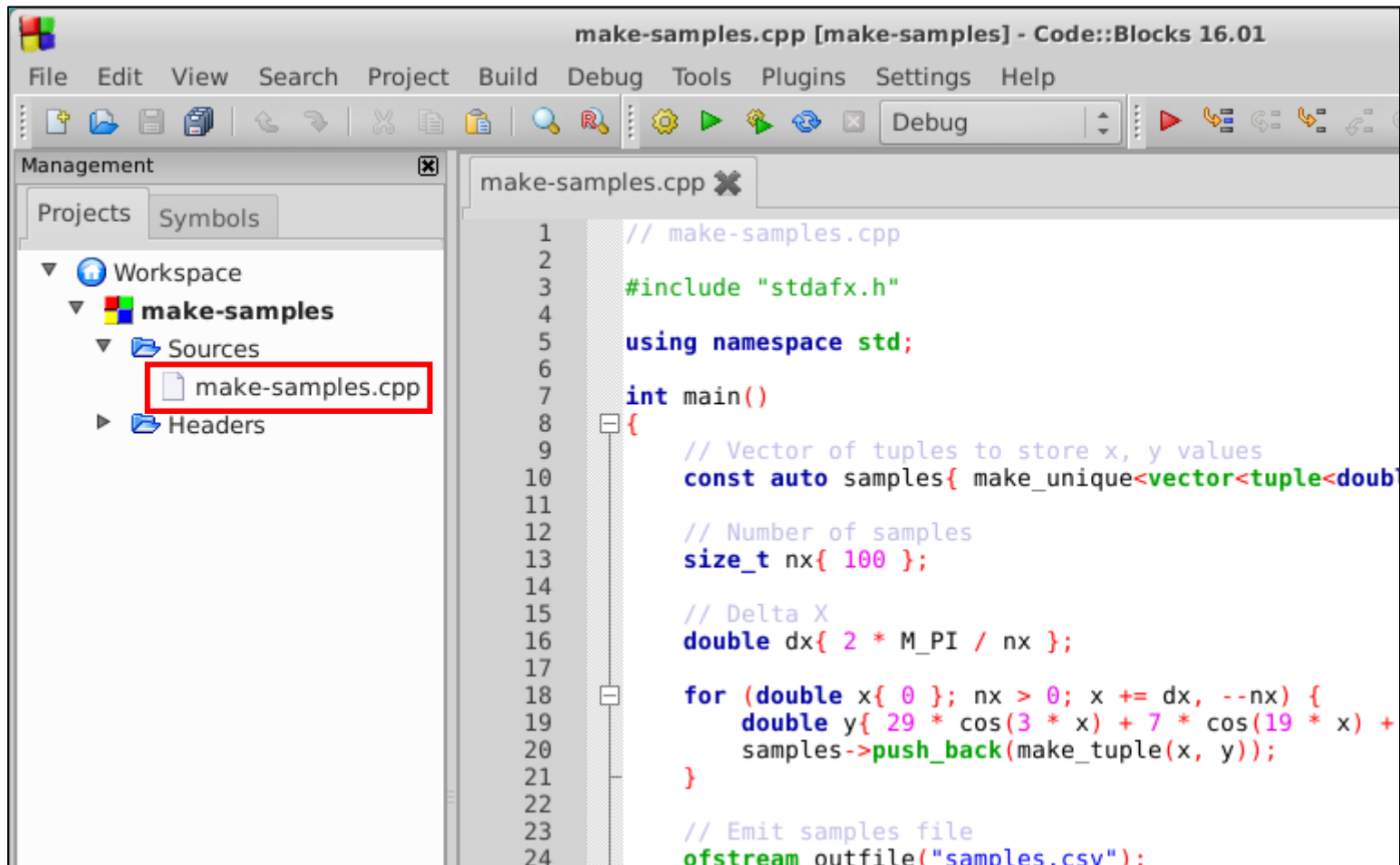
## DFT (complex)

$$X[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

## Inverse DFT (complex)

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}$$

# Open Lab 1 - Make Samples for Waveform



The screenshot shows the Code::Blocks IDE interface. The title bar reads "make-samples.cpp [make-samples] - Code::Blocks 16.01". The menu bar includes File, Edit, View, Search, Project, Build, Debug, Tools, Plugins, Settings, and Help. The toolbar contains various icons for file operations, search, and execution. The left sidebar, labeled "Management", has tabs for "Projects" and "Symbols". Under "Projects", the "Workspace" is expanded, showing a project named "make-samples". Within this project, the "Sources" folder is expanded, and the file "make-samples.cpp" is highlighted with a red rectangle. The main editor window displays the source code for "make-samples.cpp".

```
1 // make-samples.cpp
2
3 #include "stdafx.h"
4
5 using namespace std;
6
7 int main()
8 {
9     // Vector of tuples to store x, y values
10    const auto samples{ make_unique<vector<tuple<double, double>>>() };
11
12    // Number of samples
13    size_t nx{ 100 };
14
15    // Delta X
16    double dx{ 2 * M_PI / nx };
17
18    for (double x{ 0 }; nx > 0; x += dx, --nx) {
19        double y{ 29 * cos(3 * x) + 7 * cos(19 * x) +
20            samples->push_back(make_tuple(x, y));
21    }
22
23    // Emit samples file
24    ofstream outfile("samples.csv");
```

# View Lab 1 - Make Samples for Waveform

```
int main()
{
    // Vector of tuples to store x, y values
    const auto samples{make_unique<vector<tuple<double, double>>>()};

    // Number of samples
    size_t nx{100};

    // Delta X
    double dx{2 * M_PI / nx};

    for (double x{0}; nx > 0; x += dx, --nx)
    {
        double y{29 * cos(3 * x) + 7 * cos(19 * x) + 17 * sin(11 * x) + 2 * sin(31 * x)};
        samples->push_back(make_tuple(x, y));
    }

    // Create samples file
    string filename{"samples.csv"};
    ofstream outfile(filename);
    outfile.exceptions(ofstream::failbit);

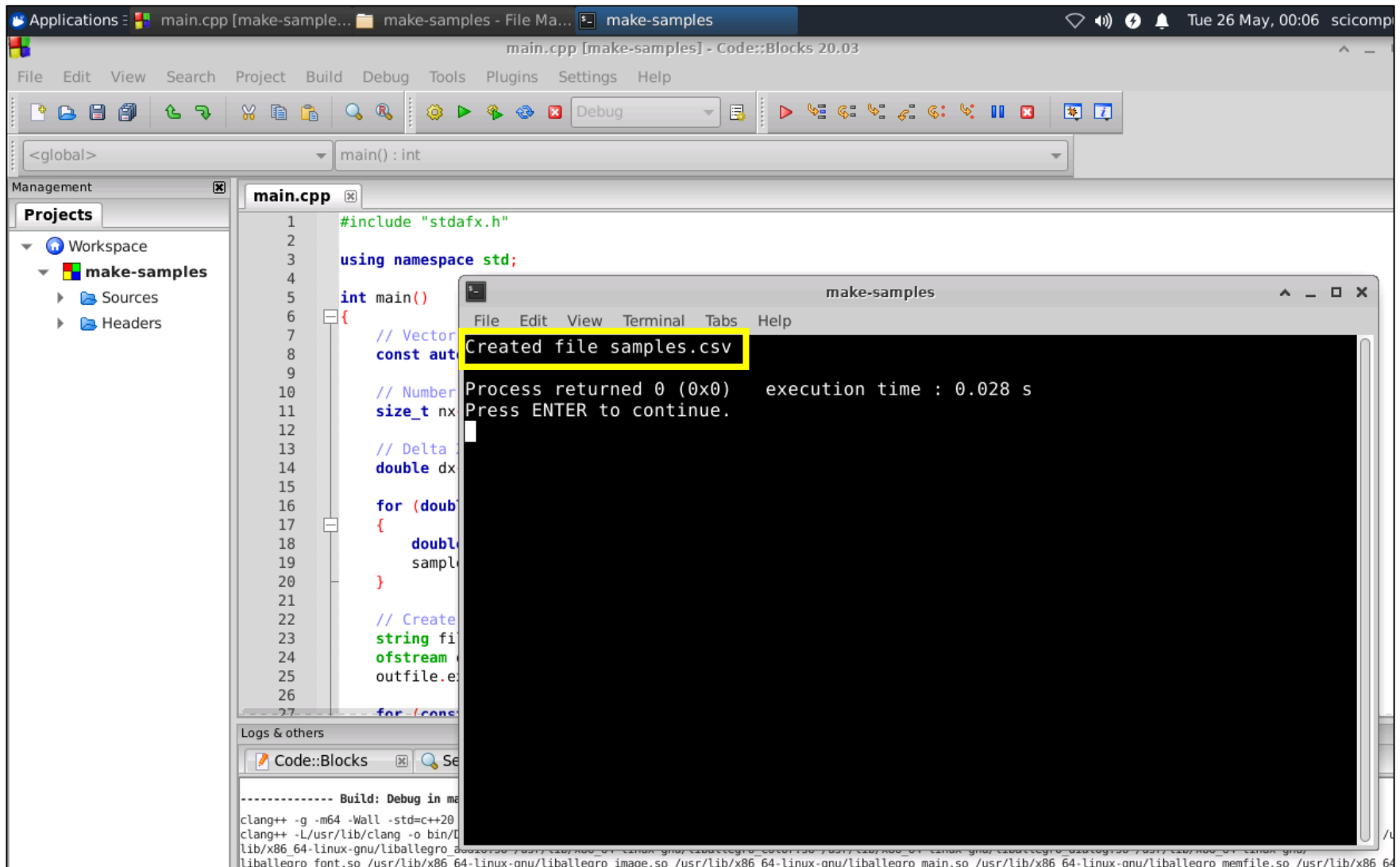
    for (const auto &d : *samples)
    {
        outfile << fixed << get<0>(d) << ", ";
        outfile << fixed << get<1>(d) << endl;
    }
    outfile.flush();
    outfile.close();

    cout << "Created file " << filename << endl;

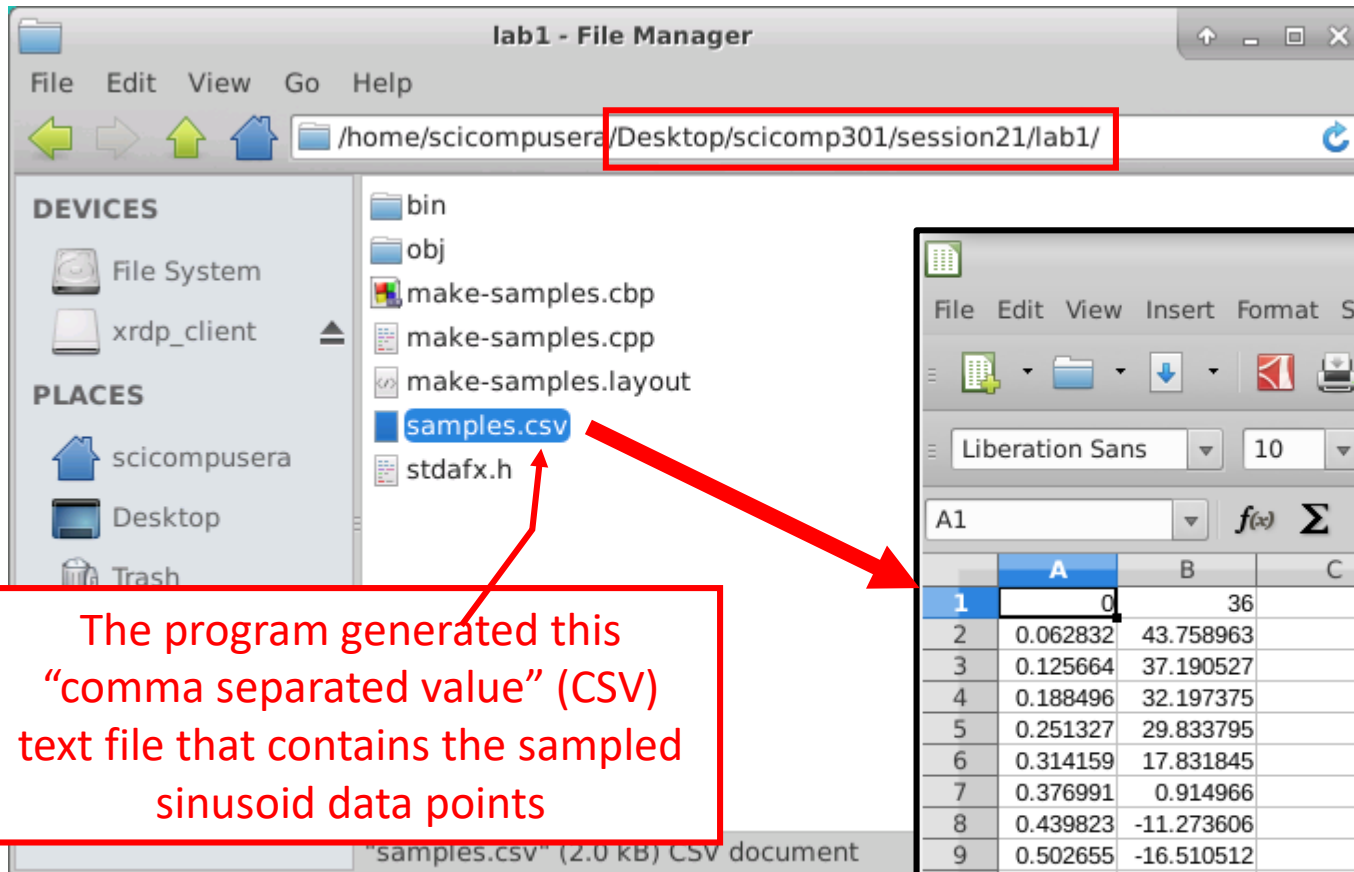
    return 0;
}
```

Pretend we don't know this insider information - we are not normally privy to the underlying formula that generated the complicated wave

# Run Lab 1 - Make Samples for Waveform

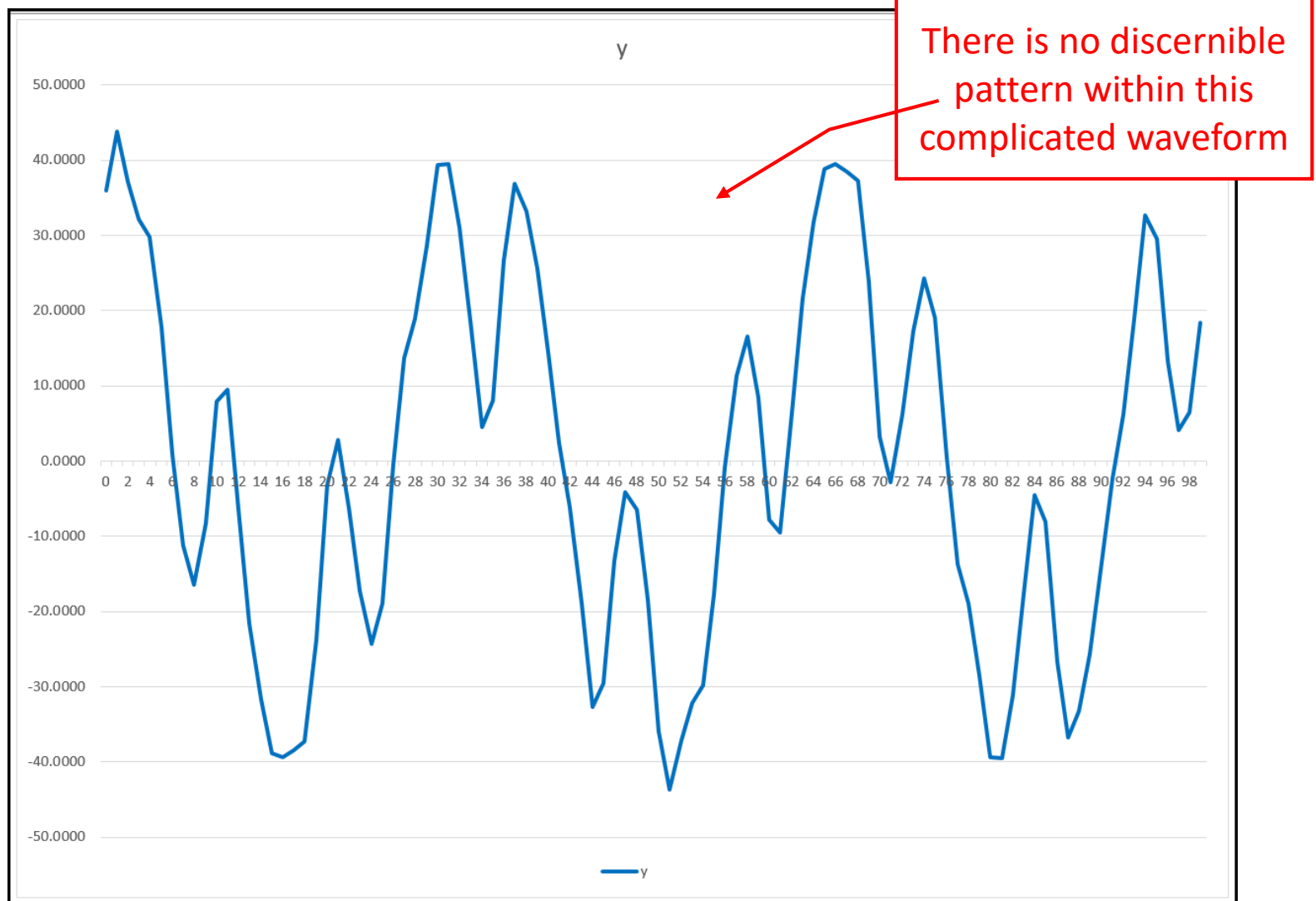


# Check Lab 1 - Make Samples for Waveform



	A	B	C	D
1	0	36		
2	0.062832	43.758963		
3	0.125664	37.190527		
4	0.188496	32.197375		
5	0.251327	29.833795		
6	0.314159	17.831845		
7	0.376991	0.914966		
8	0.439823	-11.273606		
9	0.502655	-16.510512		
10	0.565487	-8.380098		
11	0.628319	7.869546		
12	0.69115	9.477006		
13	0.753982	-6.37948		
14	0.816814	-21.607884		
15	0.879646	-31.702754		
16	0.942478	-38.837465		
17	1.00531	-39.455078		

## 100 Samples of a Complicated Wave





# Lab 2 – Discrete Fourier Transform



# Open Lab 2 – Discrete Fourier Transform

```
vector<double> xOrd;    // Sample # (0-99)
vector<double> xAct;    // Actual X value sampled
vector<double> yAct;    // Actual Y value sampled
vector<double> xRad;    // Scaled X value (radians)
vector<double> fCos;    // Frequency Cosine Amplitude
vector<double> fSin;    // Frequency Sine Amplitude
vector<double> yEst;    // Reconstructed Y value
vector<double> yPower;  // Frequency Power Amplitude
```

```
void fourier_discrete(string filename)
{
    LoadSamples(filename);

    ScaleDomain();

    CalcDFT();

    CalcIDFT();

    CalcPowerSpectrum();

    PlotTransforms();
}
```

## View Lab 2 – Discrete Fourier Transform

```
void LoadSamples(string filename)
{
    ifstream infile(filename);
    if (!infile.is_open())
    {
        cout << "Unable to open " << filename << endl;
        exit(-1);
    }
    string line{};
    const regex comma(",");
    while (infile && getline(infile, line))
    {
        vector<string> row{
            sregex_token_iterator(line.begin(), line.end(), comma, -1),
            sregex_token_iterator()};
        xAct.push_back(stod(row.at(0)));
        yAct.push_back(stod(row.at(1)));
        xOrd.push_back(xAct.size() - 1);
    }
}
```

# View Lab 2 – Discrete Fourier Transform

Multiply every term  
(integer frequency)...

...by every sampled  
data value...

...to find the  
contribution of  
each simple wave

```
void CalcDFT()
{
    const size_t sample_count{yAct.size()};
    const size_t term_count{sample_count / 2};

    for (size_t term{0}; term < term_count; ++term)
    {
        double fcos{0}, fsin{0};
        for (size_t i{0}; i < sample_count; ++i)
        {
            double xs = xRad.at(i);
            double ys = yAct.at(i);
            fcos += cos(term * xs) * ys;
            fsin += sin(term * xs) * ys;
        }
        fcos /= sample_count;
        fsin /= sample_count;
        if (term > 0)
        {
            fcos *= 2;
            fsin *= 2;
        }
        fCos.push_back(fcos);
        fSin.push_back(fsин);
    }
}
```

The amplitude of each simple wave is just the **mean value** of the product of every sample of the complicated wave multiplied by that simple wave

# View Lab 2 – Discrete Fourier Transform

```
void CalcIDFT()
{
    size_t sample_count{yAct.size()};
    size_t term_count{fCos.size()};

    for (size_t i{}; i < sample_count; ++i)
    {
        double xs = xRad.at(i);
        double yt{};
        for (size_t term{}; term < term_count; ++term)
        {
            yt += fCos.at(term) * cos(term * xs);
            yt += fSin.at(term) * sin(term * xs);
        }
        yEst.push_back(yt);
    }
}

void CalcPowerSpectrum()
{
    size_t term_count{fCos.size()};
    for (size_t term{}; term < term_count; ++term)
        yPower.push_back(sqrt(pow(fCos.at(term), 2) + pow(fSin.at(term), 2)));
}
```

The reconstructed wave is just the **linear sum** of the amplitude of each simple cosine and sine wave

# Run Lab 2 – Discrete Fourier Transform

The screenshot shows a Linux desktop environment with a dark blue background. In the top bar, there are icons for Applications, ROOT, and a terminal window titled 'Terminal - scicompuser...'. The date and time 'Tue 26 May, 04:54' are displayed on the right. The main window is a file manager titled 'lab2 - File Manager' showing the directory '/home/scicompuser2/Desktop/scicomp301/session21/lab2/'. It contains three files: 'fourier\_discrete.cpp', 'samples.csv', and 'stdafx.h'. A terminal window titled 'Terminal - scicompuser2@ip-172-31-61-38: ~/Desktop/scicomp301/session21/lab2' is open in the foreground. It shows the user navigating to the directory and running a program. A blue speech bubble points to the terminal with the text: 'We must pass in the filename containing the sampled data'. A red-bordered box highlights the commands entered in the terminal.

lab2 - File Manager

File Edit View Go Help

/home/scicompuser2/Desktop/scicomp301/session21/lab2/

DEVICES

- File System
- thinclient\_dri...

PLACES

- scicompuser2
- Desktop
- Trash
- Documents
- Music
- Pictures
- Videos
- Downloads

NETWORK

- Browse Network

Terminal - scicompuser2@ip-172-31-61-38: ~/Desktop/scicomp301/session21/lab2

File Edit View Terminal Tabs Help

```
scicompuser2@ip-172-31-61-38:~$ cd Desktop/scicomp301/session21/lab2
scicompuser2@ip-172-31-61-38:~/Desktop/scicomp301/session21/lab2$ root

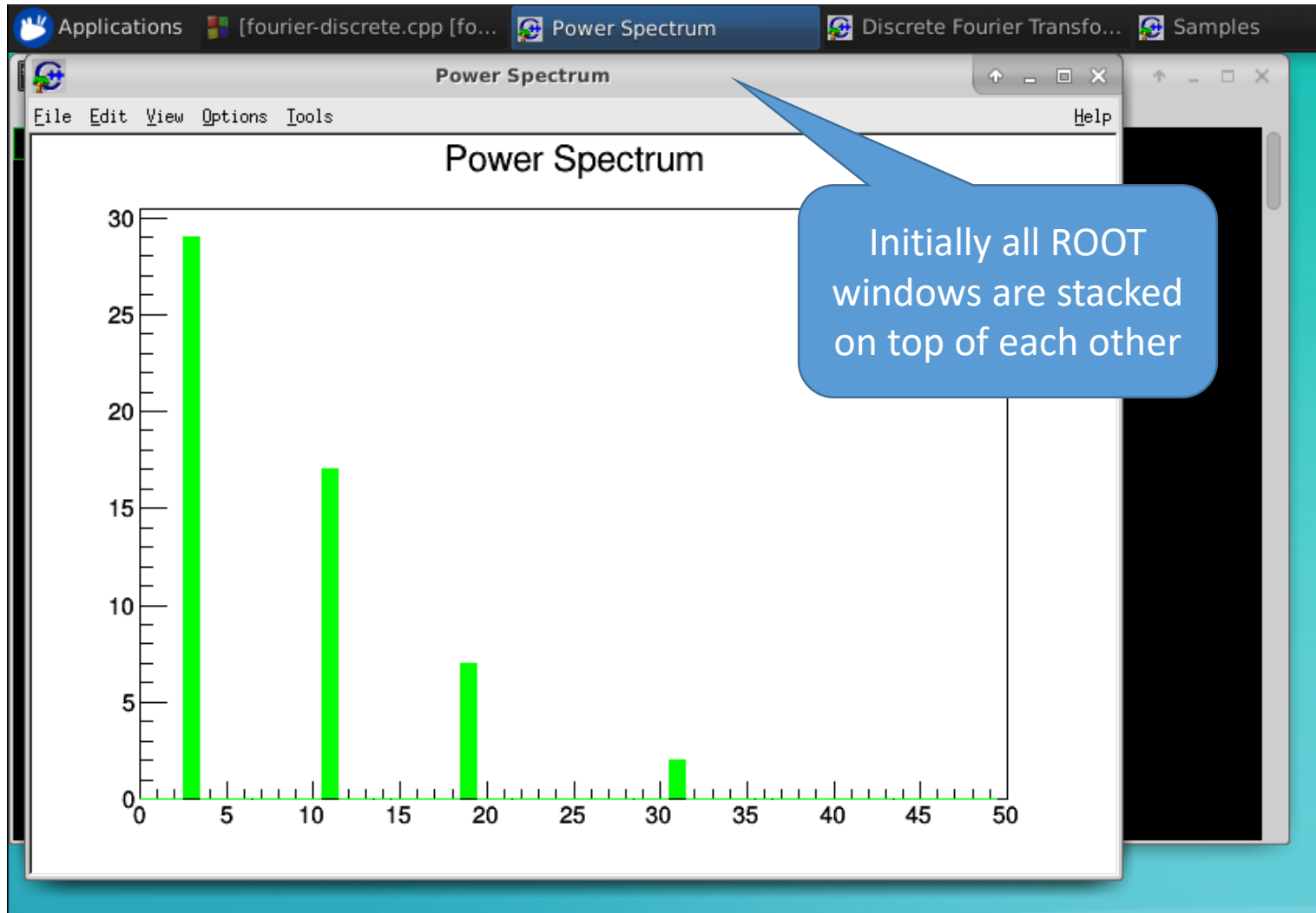
-----
Welcome to ROOT 6.20/04                                     https://root.cern
(c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers
Built for linuxx8664gcc on Apr 01 2020, 08:28:48
From tags/v6-20-04@v6-20-04
Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q'
-----

root [0] .x fourier_discrete.cpp("samples.csv")
root [1]
```

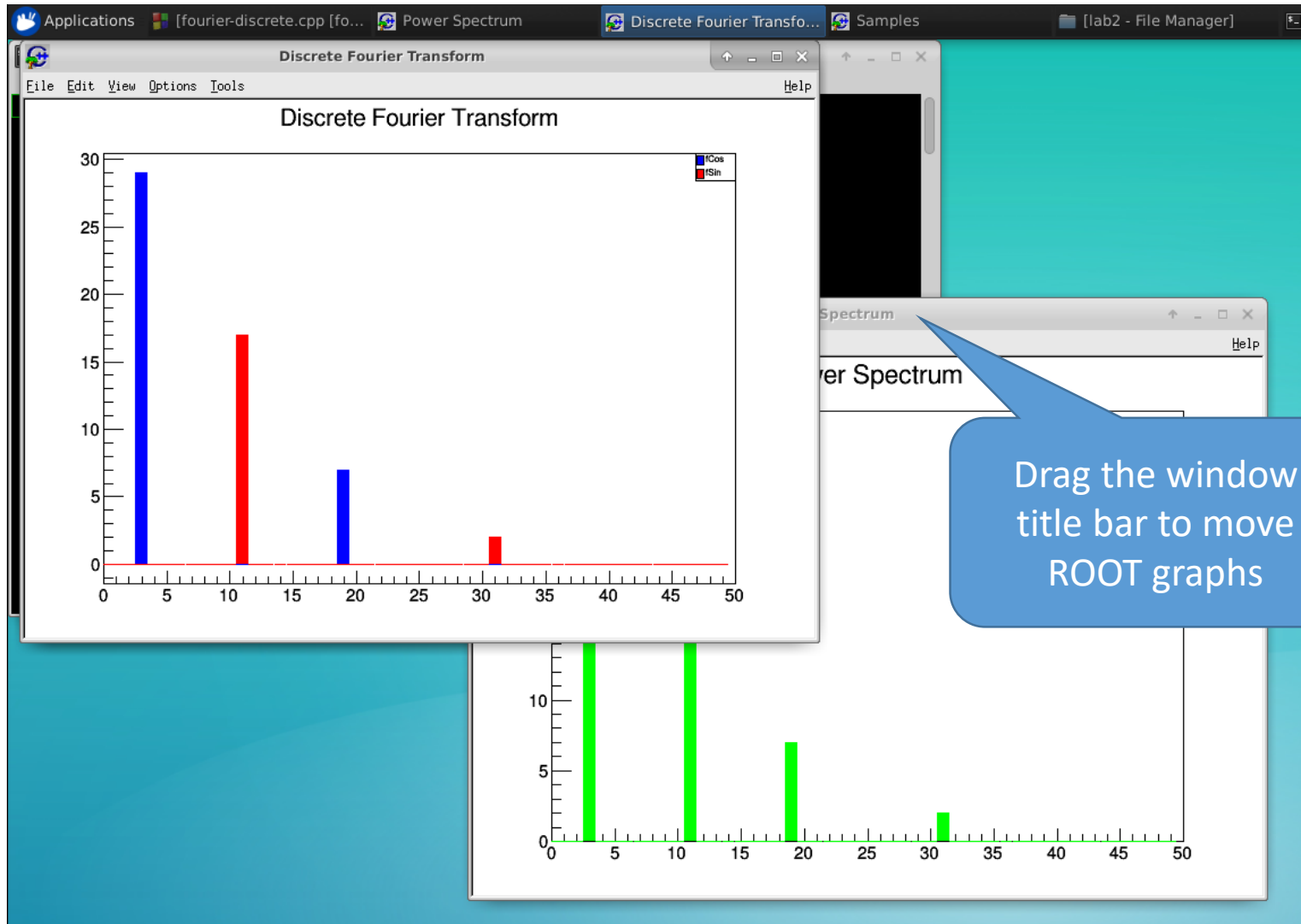
We must pass in the filename containing the sampled data

cd Desktop/scicomp301/session21/lab2  
root  
.x fourier\_discrete.cpp("samples.csv")

# Run Lab 2 – Discrete Fourier Transform

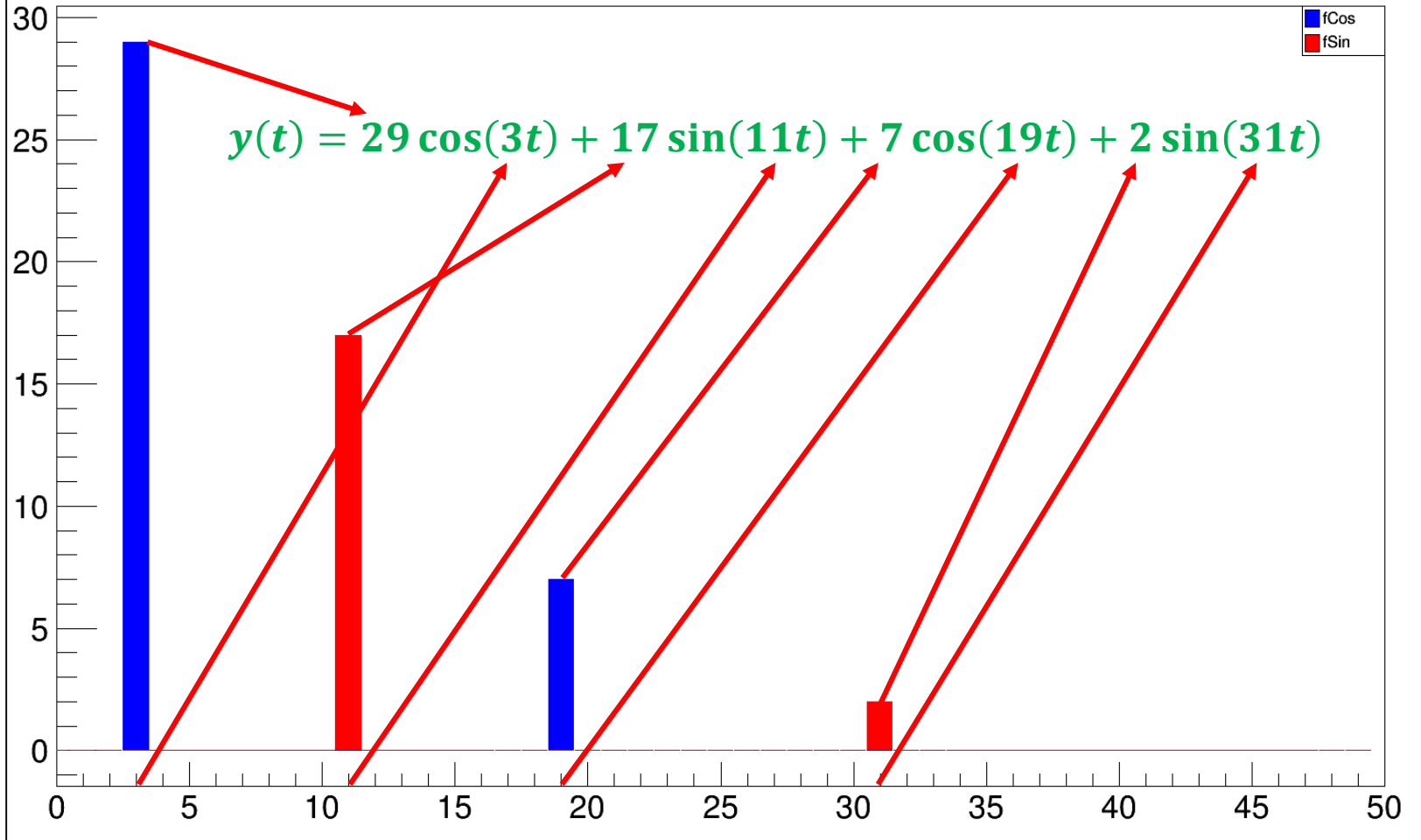


# Check Lab 2 – Discrete Fourier Transform

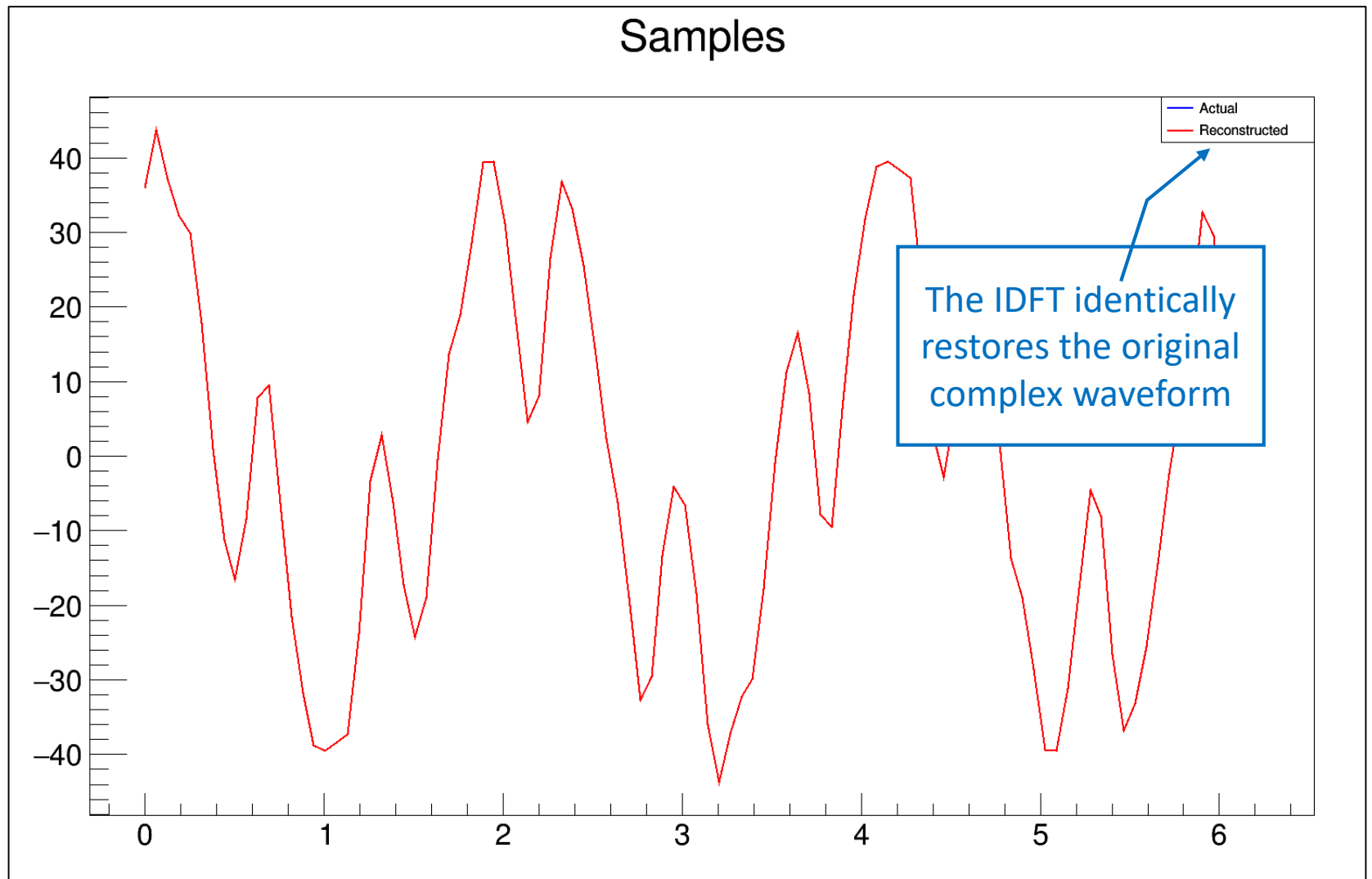


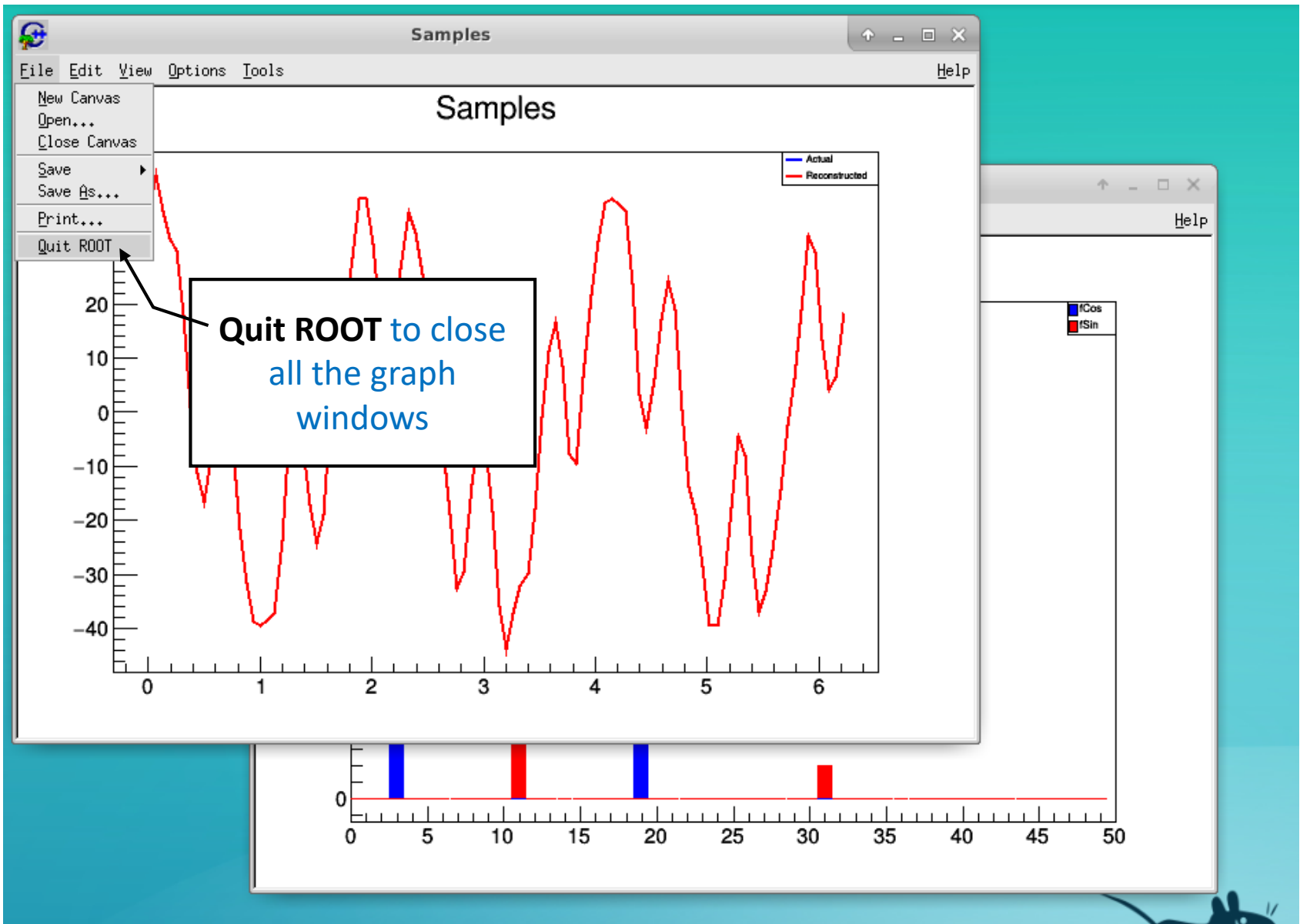


## Discrete Fourier Transform



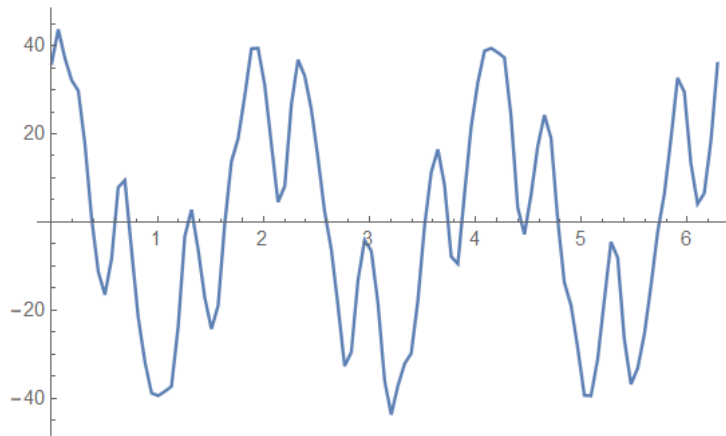
# Check Lab 2 – Discrete Fourier Transform



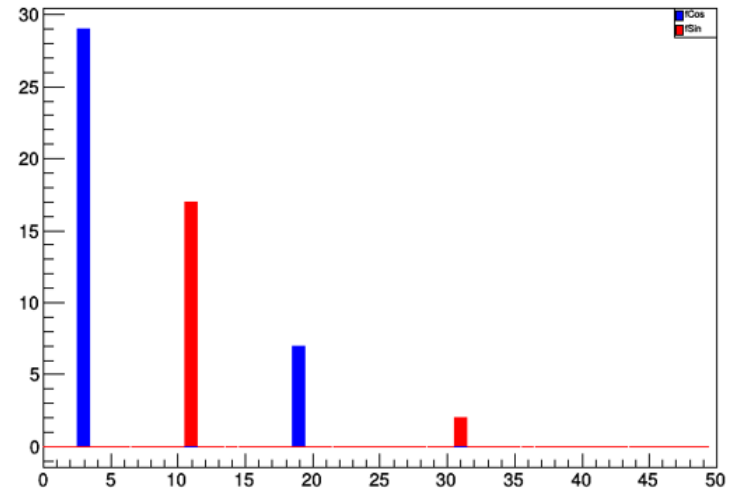


# Information Density

$$y(t) = 29 \cos(3t) + 17 \sin(11t) + 7 \cos(19t) + 2 \sin(31t)$$



Time domain



Frequency domain

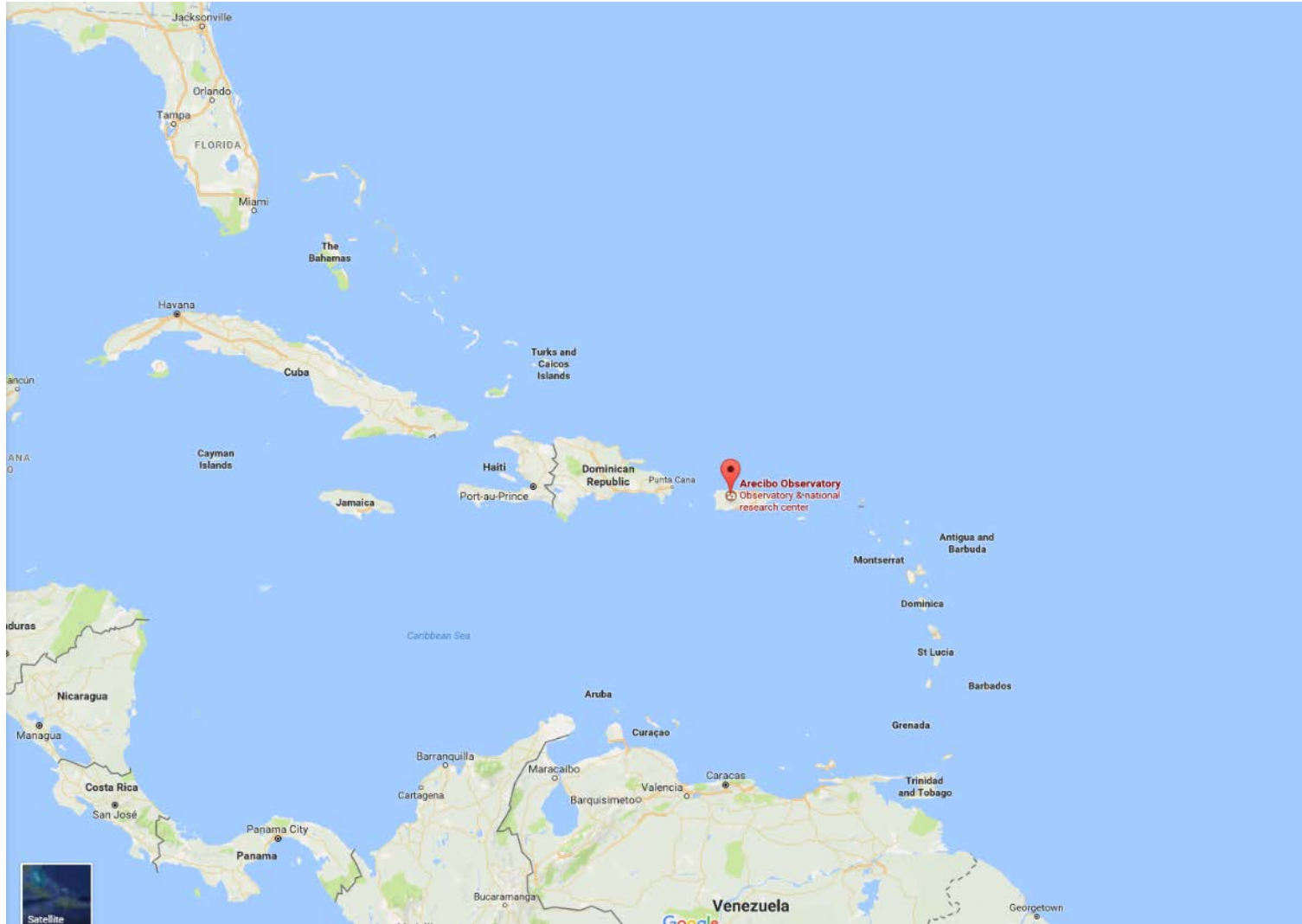
All the complexity of the original waveform in the time domain (200 numbers) is fully captured using just ***eight*** numbers in the frequency domain!

# Analysis of Space Signals



- The **Arecibo Radio Observatory** has detected three candidate signals originating from deep space
- Your task is to perform **Fourier Analysis** on each signal to determine which one **is more likely** to have been a broadcast from an extraterrestrial intelligence

# Analysis of Space Signals



# Analysis of Space Signals

**Arecibo Observatory** ★

4.7 ★★★★★ 322 Google reviews  
Observatory in Arecibo

[Website](#) [Directions](#)

The Arecibo Observatory is a radio telescope in the municipality of Arecibo, Puerto Rico. This observatory is operated by SRI International, USRA and UMET, under cooperative agreement with the National Science Foundation. [Wikipedia](#)

**Address:** PR-625, Arecibo, 00612, Puerto Rico  
**Area:** 118 acres  
**Opened:** 1963  
**Collecting area:** 18 acres  
**Hours:** Open today · 9AM–4PM ▾  
**Phone:** +1 787-878-2612  
**Added to NRHP:** September 23, 2008  
**Architects:** William E. Gordon, Thomas Christian Kavanagh

**Did you know:** In 1994, John Harmon used the Arecibo Radio Telescope to map the distribution of ice in the polar regions of Mercury. [wikipedia.org](#)

[Suggest an edit](#)

**Popular times** ⓘ Sundays ↕

Now: Usually not too busy

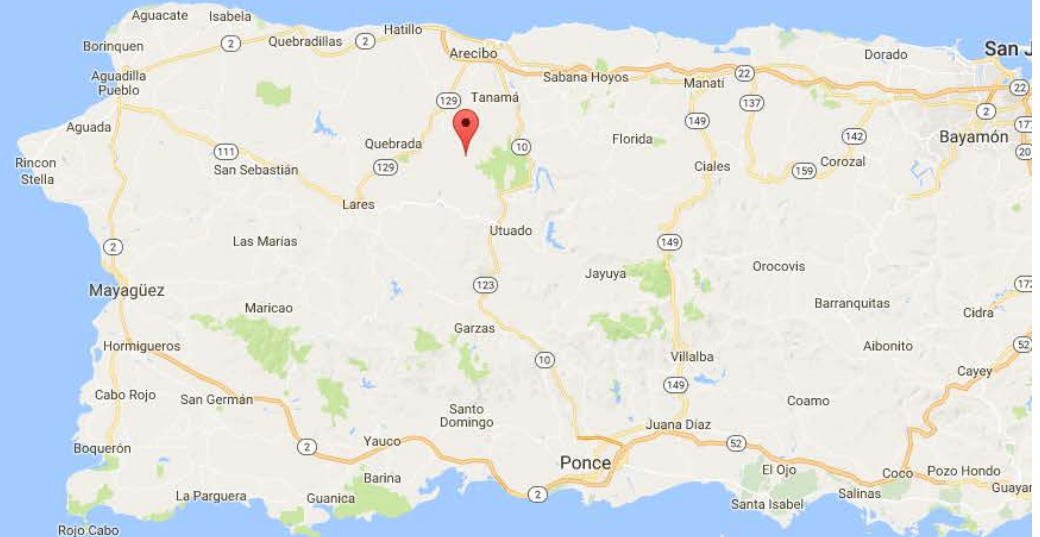


**Plan your visit:** People typically spend up to 1.5 hours here



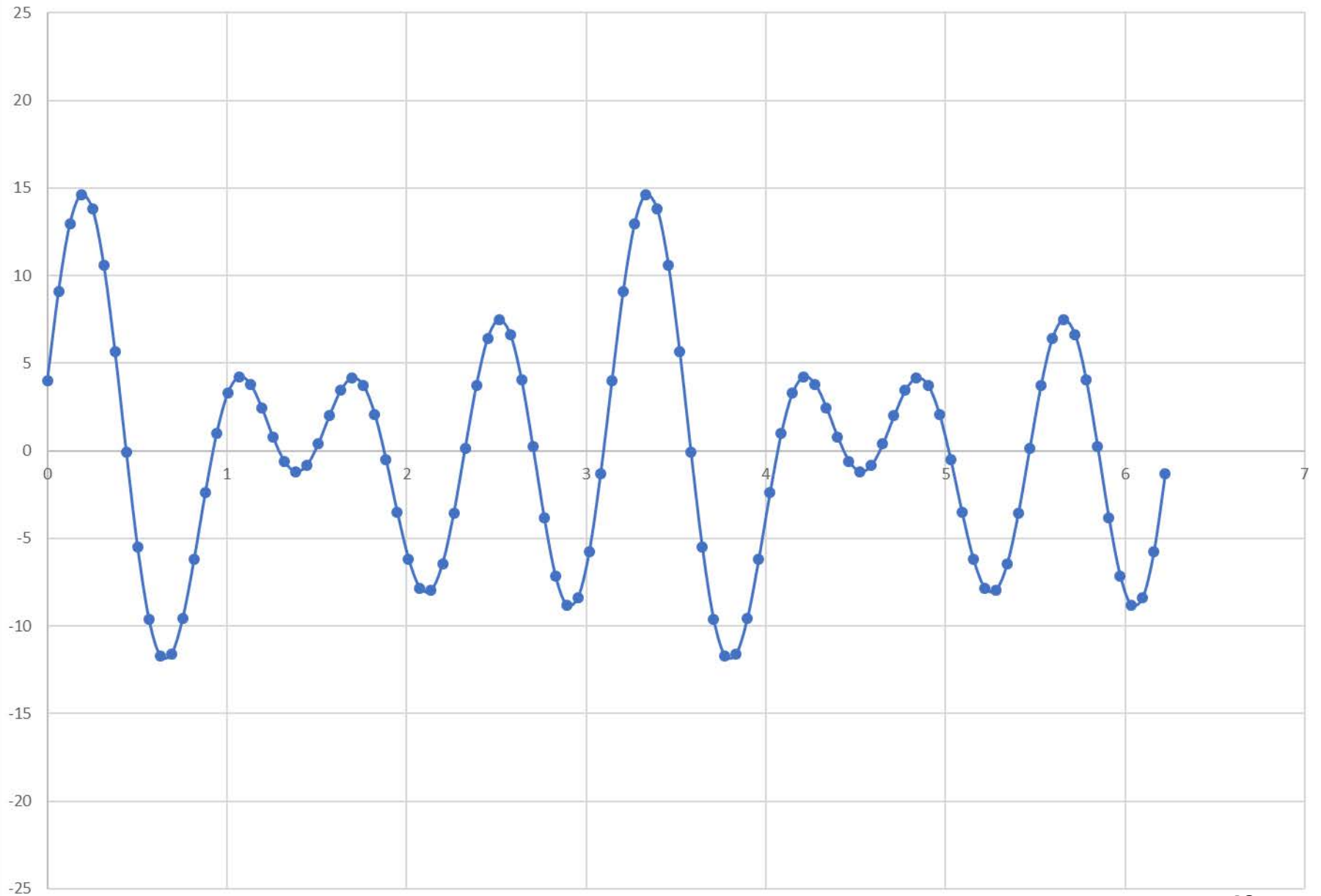
[More photos](#)

**Did you know:** In 1994, John Harmon used the Arecibo Radio Telescope to map the distribution of ice in the polar regions of Mercury. [wikipedia.org](#)



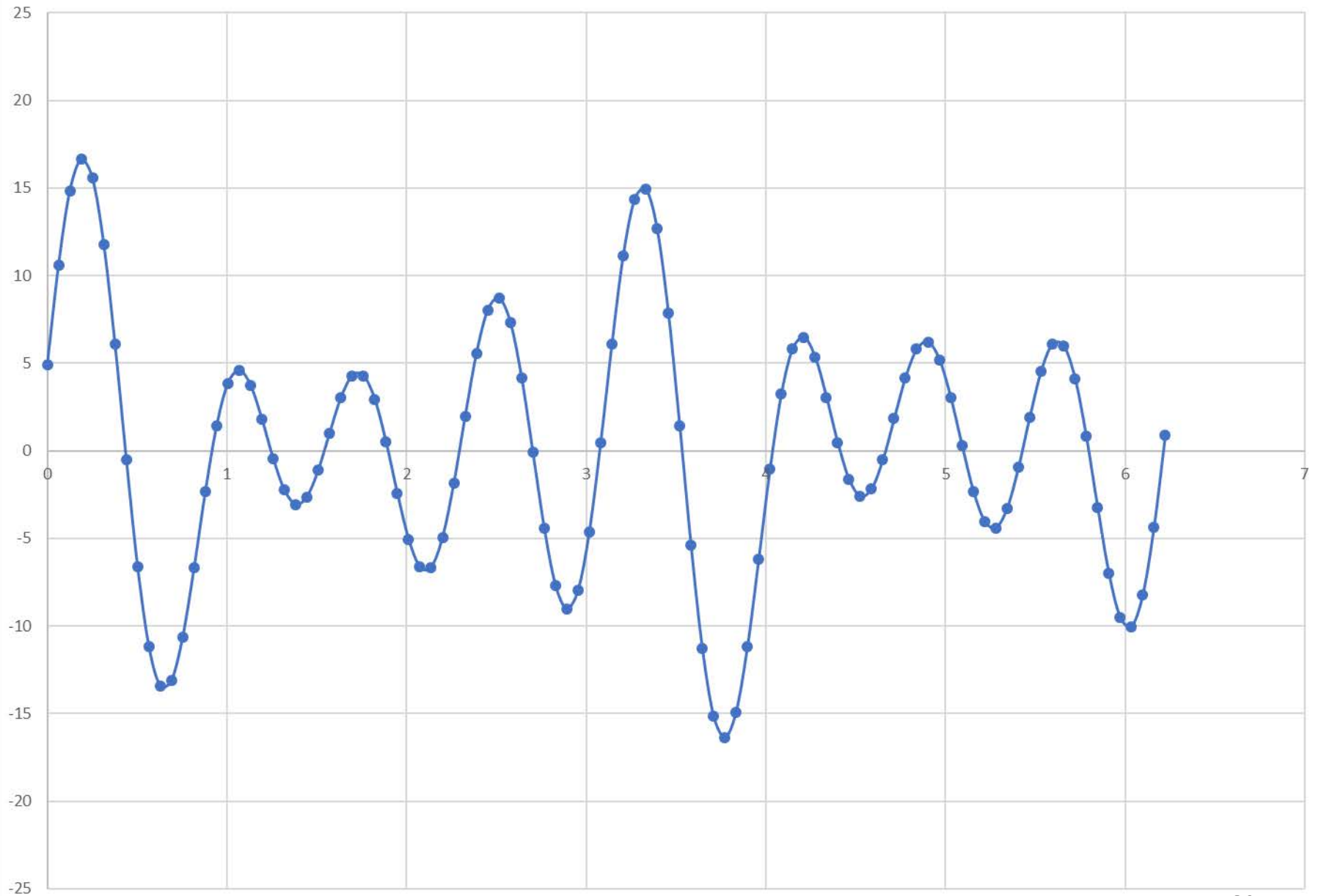


space\_signals1.csv

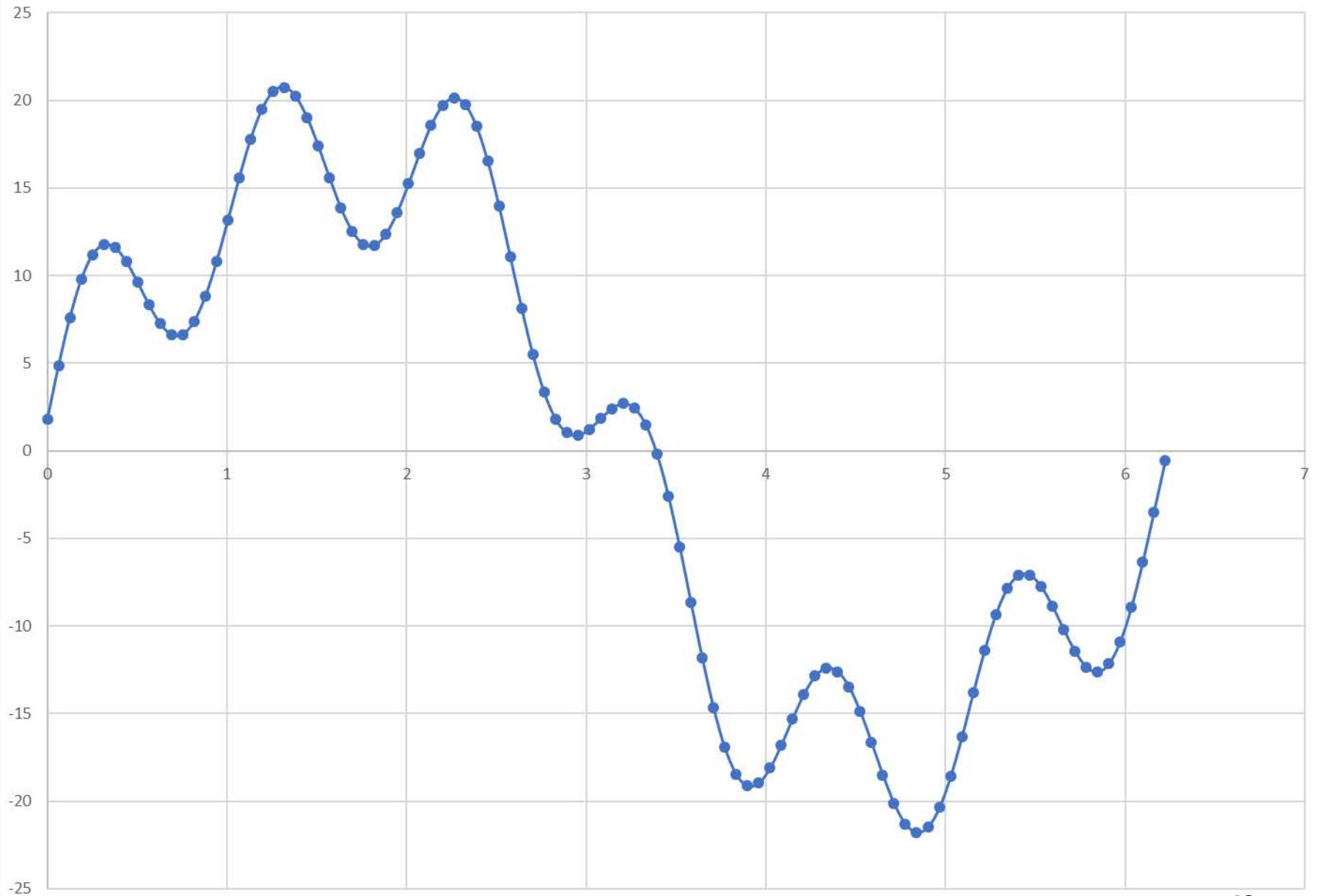




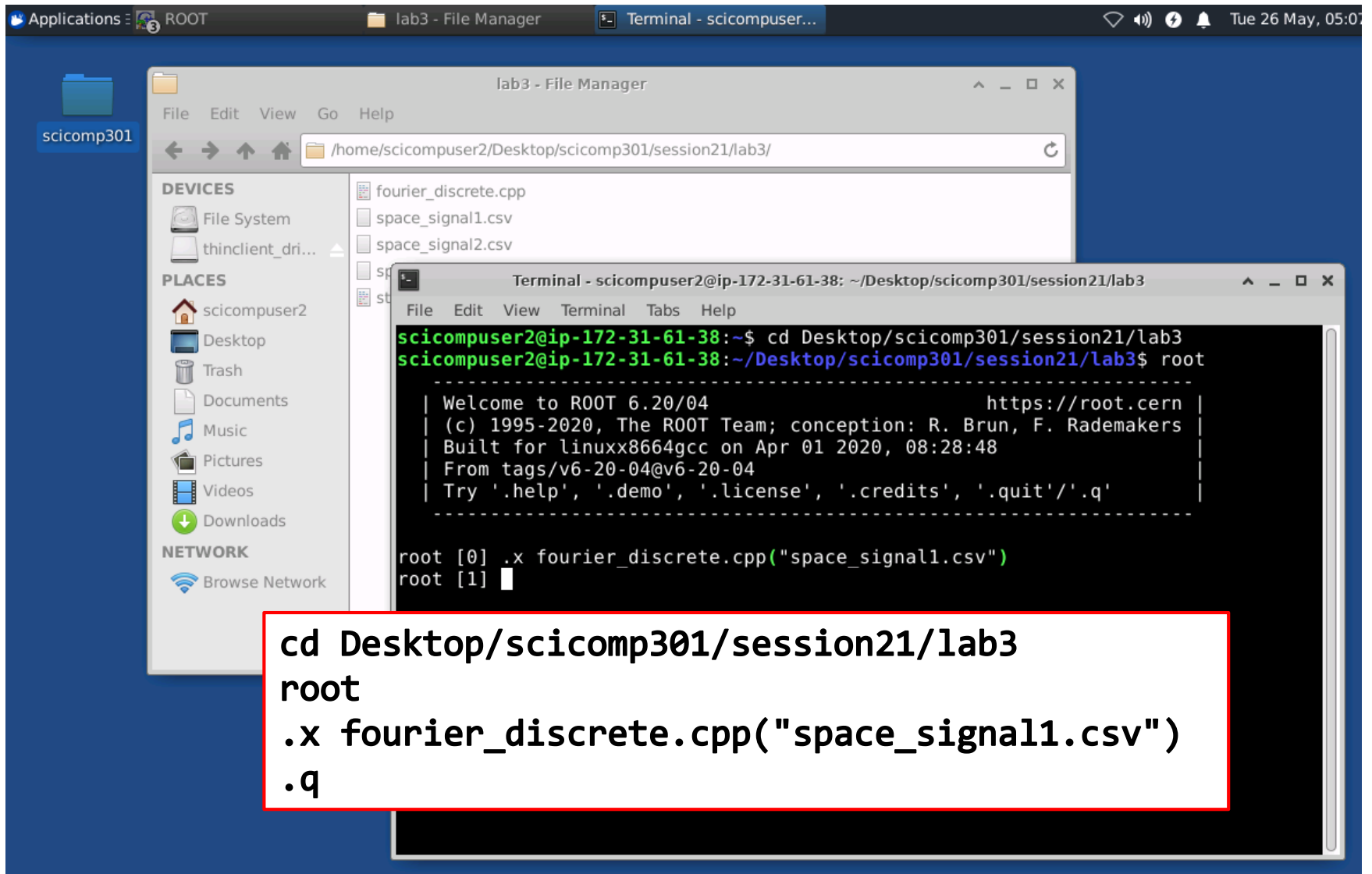
space\_signals2.csv



space\_signals3.csv



# Run Lab 3 – Space Signals Analysis



The screenshot shows a Linux desktop environment with a blue background. The top panel displays the system menu, the current user 'ROOT', and open windows: 'lab3 - File Manager' and 'Terminal - scicompuser...'. The system clock shows 'Tue 26 May, 05:07'. The 'lab3 - File Manager' window is open to the directory '/home/scicompuser2/Desktop/scicomp301/session21/lab3/'. It lists files: 'fourier\_discrete.cpp', 'space\_signal1.csv', and 'space\_signal2.csv'. The 'Terminal - scicompuser2@ip-172-31-61-38: ~/Desktop/scicomp301/session21/lab3' window shows the following commands and output:

```
scicompuser2@ip-172-31-61-38:~$ cd Desktop/scicomp301/session21/lab3
scicompuser2@ip-172-31-61-38:~/Desktop/scicomp301/session21/lab3$ root

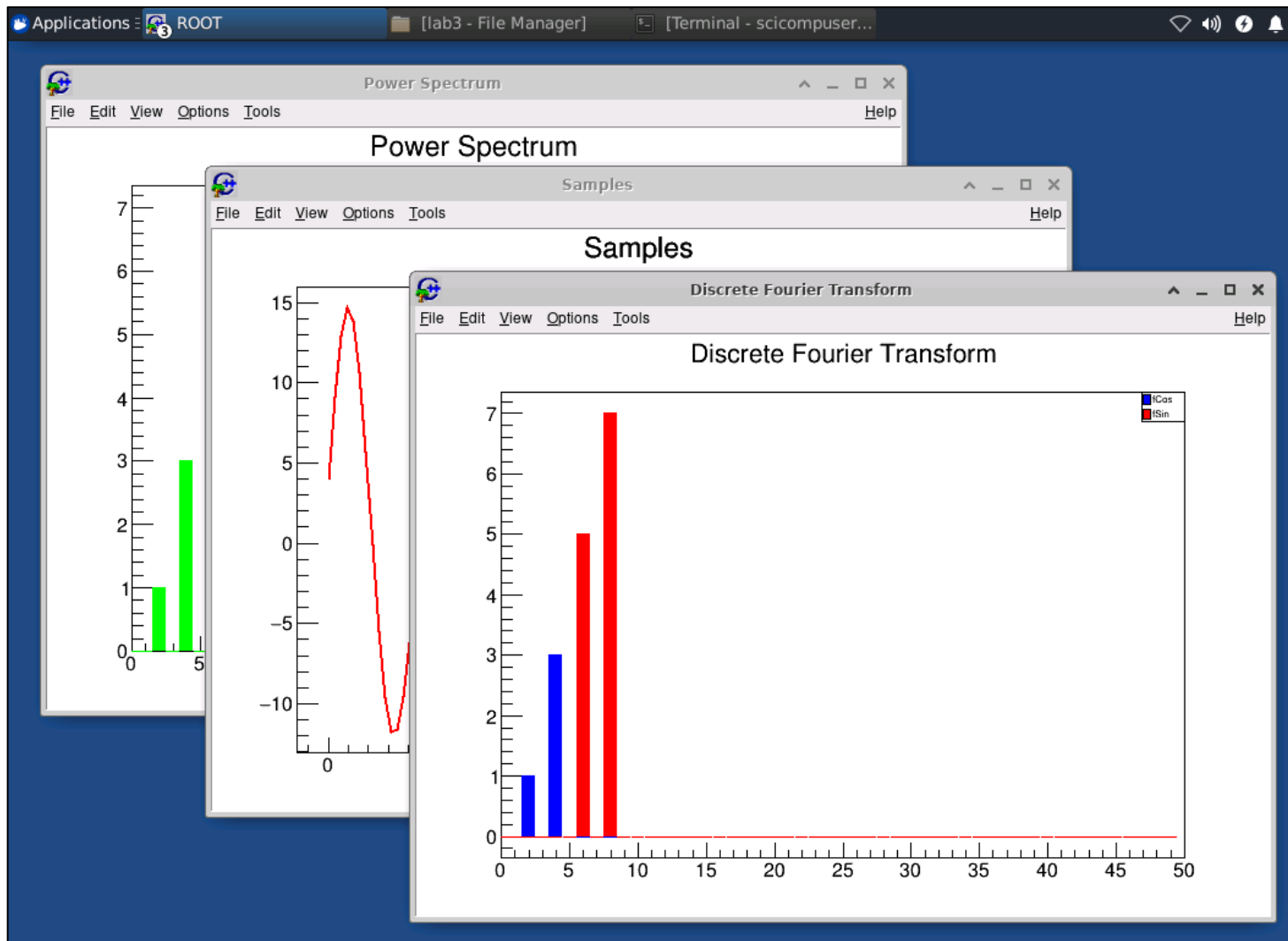
-----
| Welcome to ROOT 6.20/04                                     https://root.cern |
| (c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers |
| Built for linuxx8664gcc on Apr 01 2020, 08:28:48 |
| From tags/v6-20-04@v6-20-04 |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
-----

root [0] .x fourier_discrete.cpp("space_signal1.csv")
root [1]
```

A red-bordered box highlights the commands entered in the terminal:

```
cd Desktop/scicomp301/session21/lab3
root
.x fourier_discrete.cpp("space_signal1.csv")
.q
```

# Check Lab 3 – Space Signals Analysis



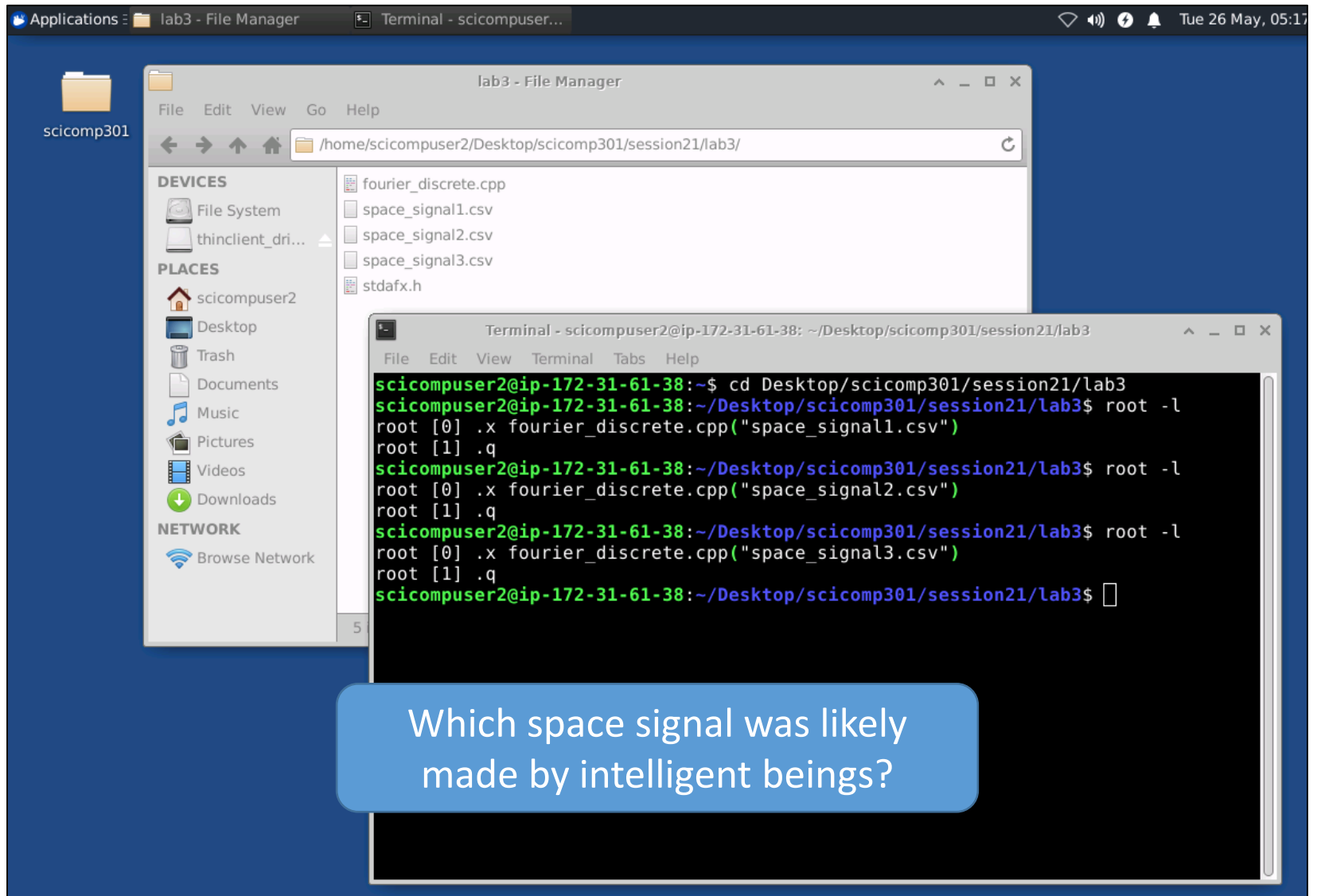
# Run Lab 3 – Space Signals Analysis

The **lowercase L** option suppresses the CERN ROOT Welcome banner

You must run ROOT **three** times (once for each data file)

```
cd Desktop/scicomp301/session21/lab3
root -l
.x fourier_discrete.cpp("space_signal1.csv")
.q
root -l
.x fourier_discrete.cpp("space_signal2.csv")
.q
root -l
.x fourier_discrete.cpp("space_signal3.csv")
.q
```

# Run Lab 3 – Space Signals Analysis



The screenshot shows a Linux desktop environment. In the background, a file manager window titled "lab3 - File Manager" is open, displaying the directory `/home/scicompuser2/Desktop/scicomp301/session21/lab3/`. The file list includes `fourier_discrete.cpp`, `space_signal1.csv`, `space_signal2.csv`, `space_signal3.csv`, and `stdafx.h`. In the foreground, a terminal window titled "Terminal - scicompuser2@ip-172-31-61-38: ~/Desktop/scicomp301/session21/lab3" is open. The terminal shows the following commands and output:

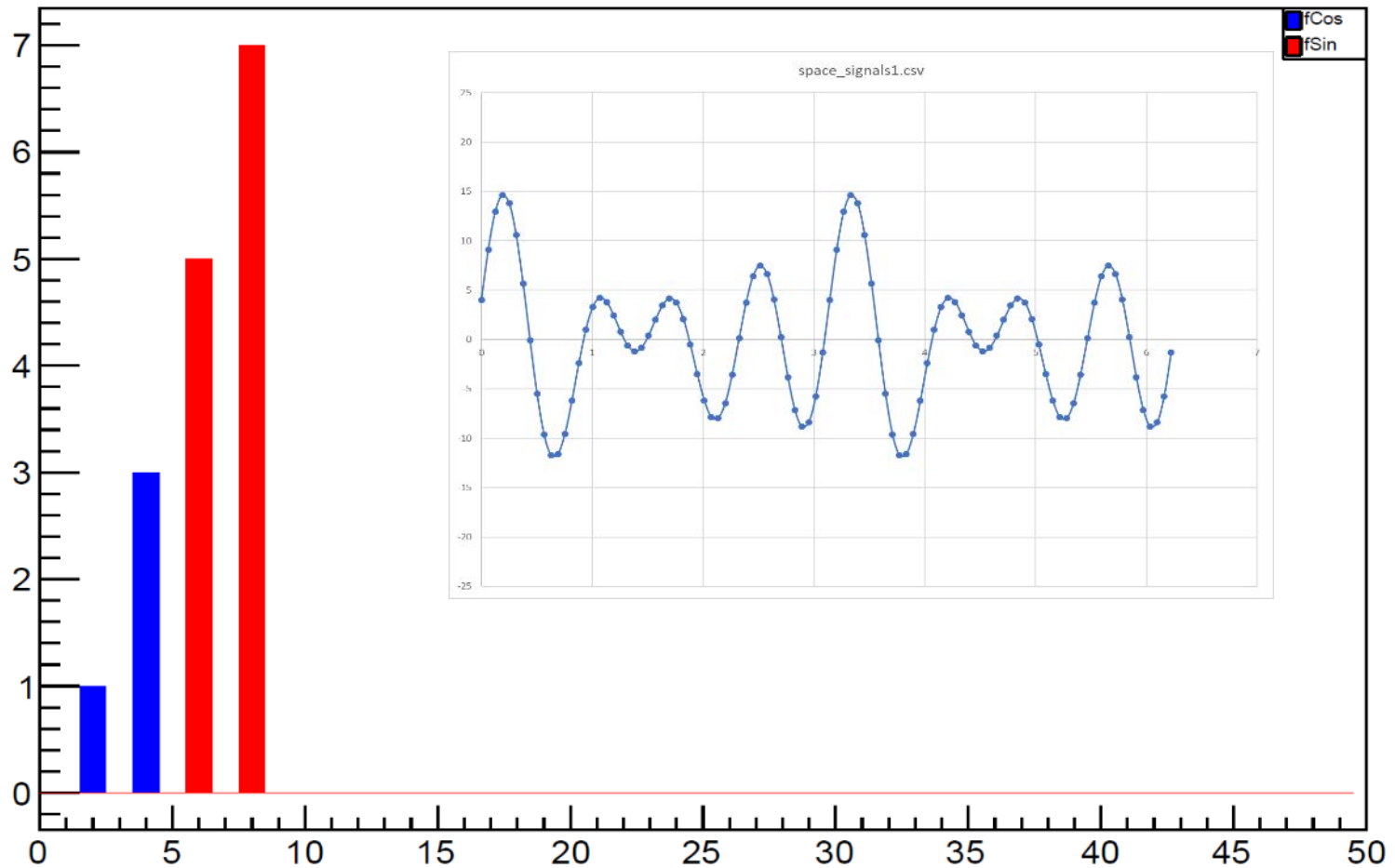
```
scicompuser2@ip-172-31-61-38:~$ cd Desktop/scicomp301/session21/lab3
scicompuser2@ip-172-31-61-38:~/Desktop/scicomp301/session21/lab3$ root -l
root [0] .x fourier_discrete.cpp("space_signal1.csv")
root [1] .q
scicompuser2@ip-172-31-61-38:~/Desktop/scicomp301/session21/lab3$ root -l
root [0] .x fourier_discrete.cpp("space_signal2.csv")
root [1] .q
scicompuser2@ip-172-31-61-38:~/Desktop/scicomp301/session21/lab3$ root -l
root [0] .x fourier_discrete.cpp("space_signal3.csv")
root [1] .q
scicompuser2@ip-172-31-61-38:~/Desktop/scicomp301/session21/lab3$
```

A blue speech bubble is overlaid on the terminal window, containing the text:

Which space signal was likely made by intelligent beings?

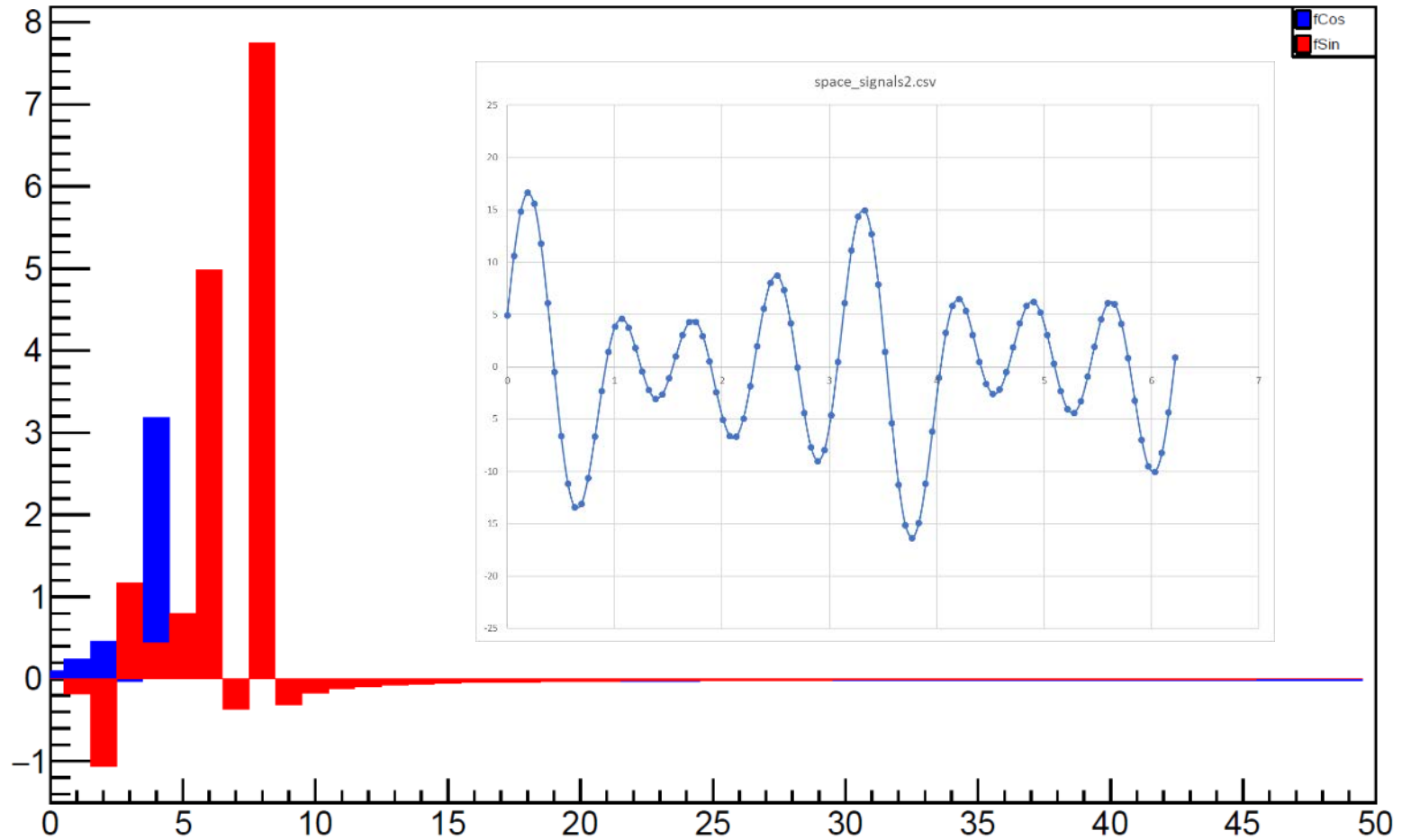
# space\_signal1

## Discrete Fourier Transform



# space\_signal2

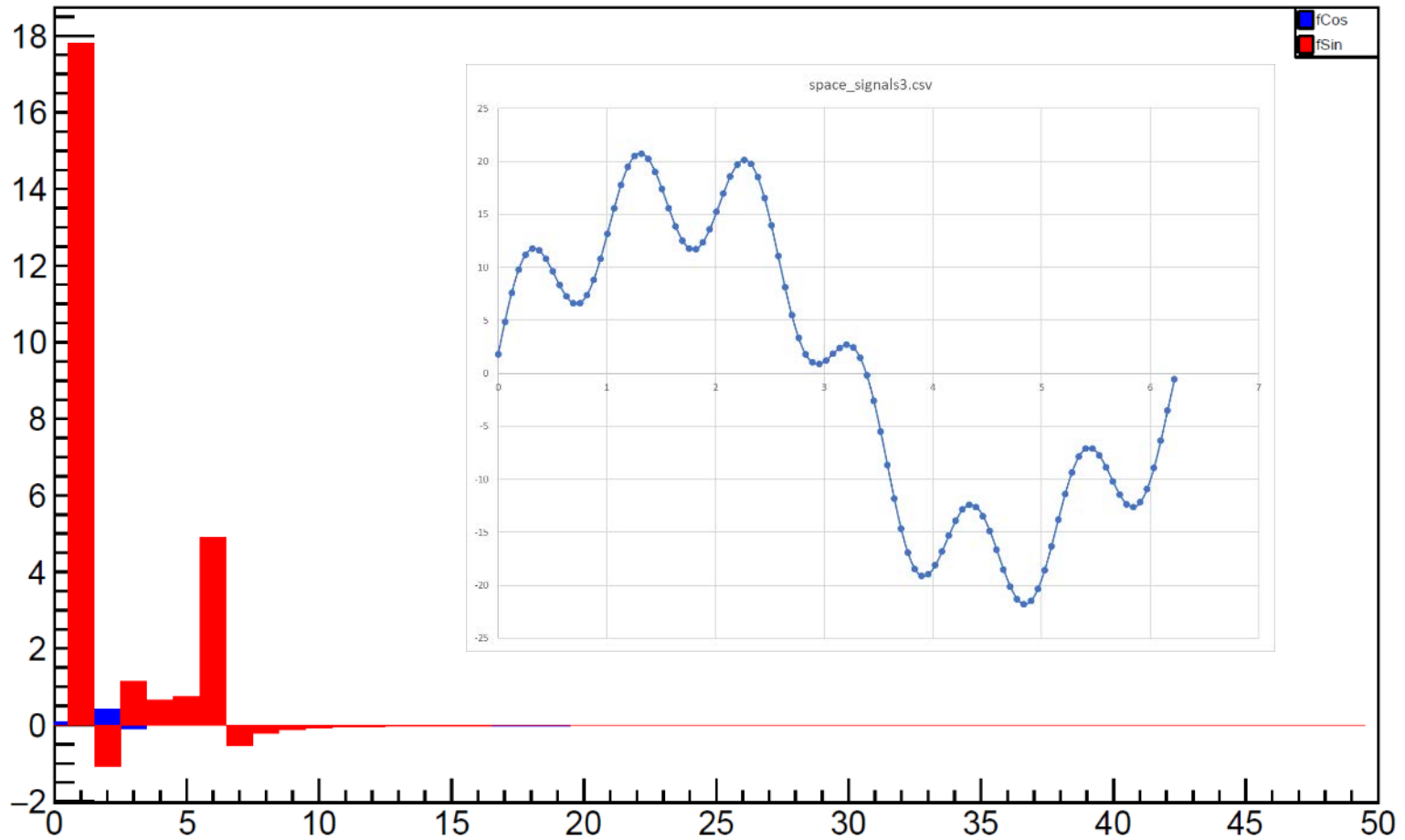
## Discrete Fourier Transform



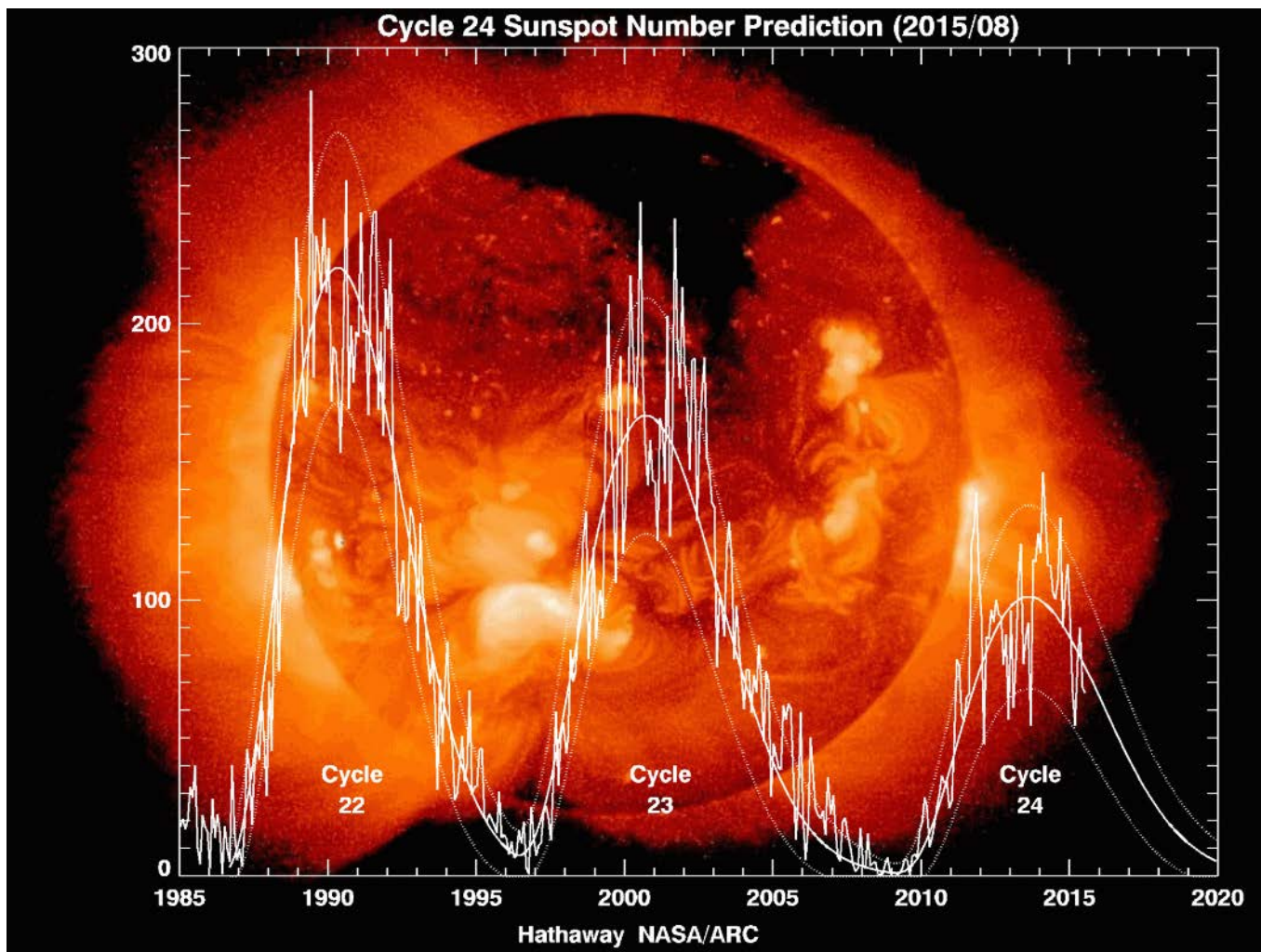


# space\_signal3

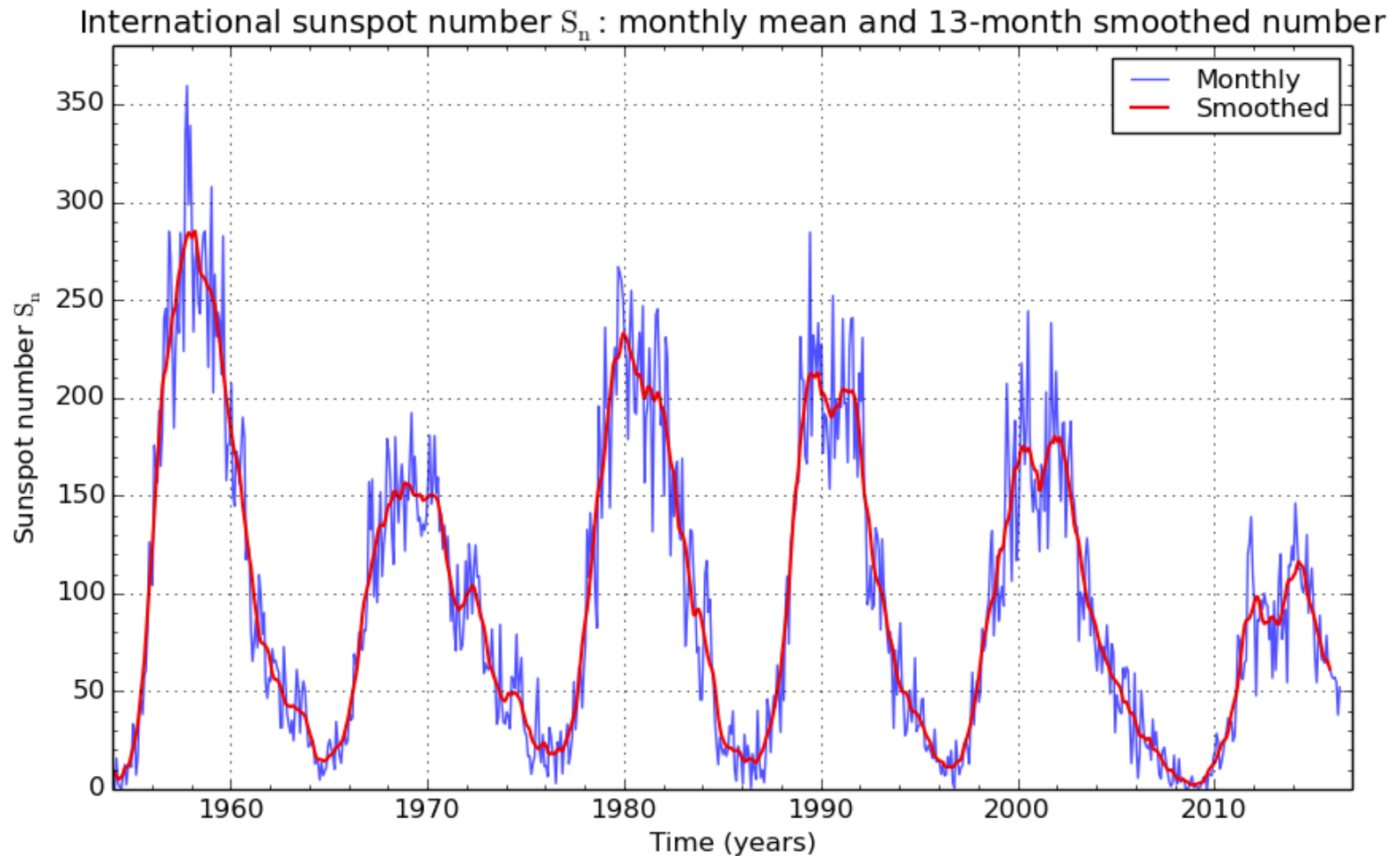
## Discrete Fourier Transform



# Sunspot Activity



# Sunspot Activity



SILSO graphics (<http://sidc.be/silso>) Royal Observatory of Belgium 2016 June 1

# Sunspot Activity

[sunspots.csv](#)

	A	B	C	D	E	F	G	H
1	1700	8.3	1800	24.2	1900	15.7	2000	173.9
2	1701	18.3	1801	56.7	1901	4.6	2001	170.4
3	1702	26.7	1802	75	1902	8.5	2002	163.6
4	1703	38.3	1803	71.8	1903	40.8	2003	99.3
5	1704	60	1804	70.2	1904	70.1	2004	65.2
6	1705	96.7	1805	70.2	1905	70.1	2005	65.2
7	1706	48.3	1806	70.2	1906	70.1	2006	65.2
8	1707	33.3	1807	70.2	1907	70.1	2007	65.2
9	1708	16.7	1808	70.2	1908	70.1	2008	65.2
10	1709	13.3	1809	70.2	1909	70.1	2009	65.2
11	1710	5	1810	70.2	1910	70.1	2010	65.2
12	1711	0	1811	70.2	1911	70.1	2011	65.2
13	1712	0	1812	70.2	1912	70.1	2012	65.2
14	1713	3.3	1813	70.2	1913	70.1	2013	65.2
15	1714	18.3	1814	70.2	1914	70.1	2014	65.2
16	1715	45	1815	70.2	1915	70.1	2015	65.2
17	1716	78.3	1816	70.2	1916	70.1	2016	65.2
18	1717	105	1817	70.2	1917	70.1	2017	65.2

A human has counted  
sunspots *every day* from  
**1700 – 2016!**

## Wolf number

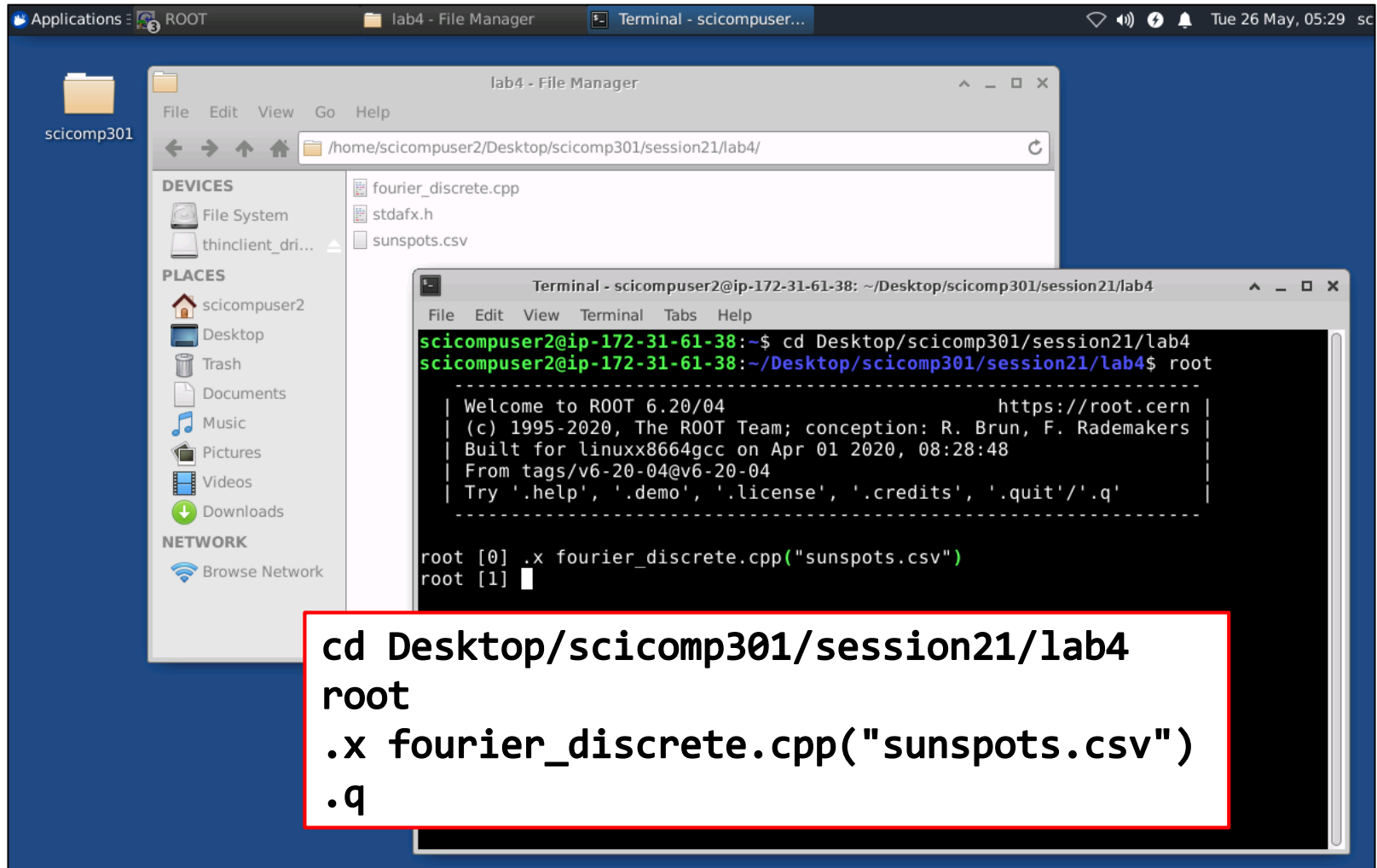
The **Wolf number** (also known as the **International sunspot number**, **relative sunspot number**, or **Zürich number**) is a quantity that measures the number of [sunspots](#) and groups of sunspots present on the surface of the sun.

## History [\[ edit \]](#)

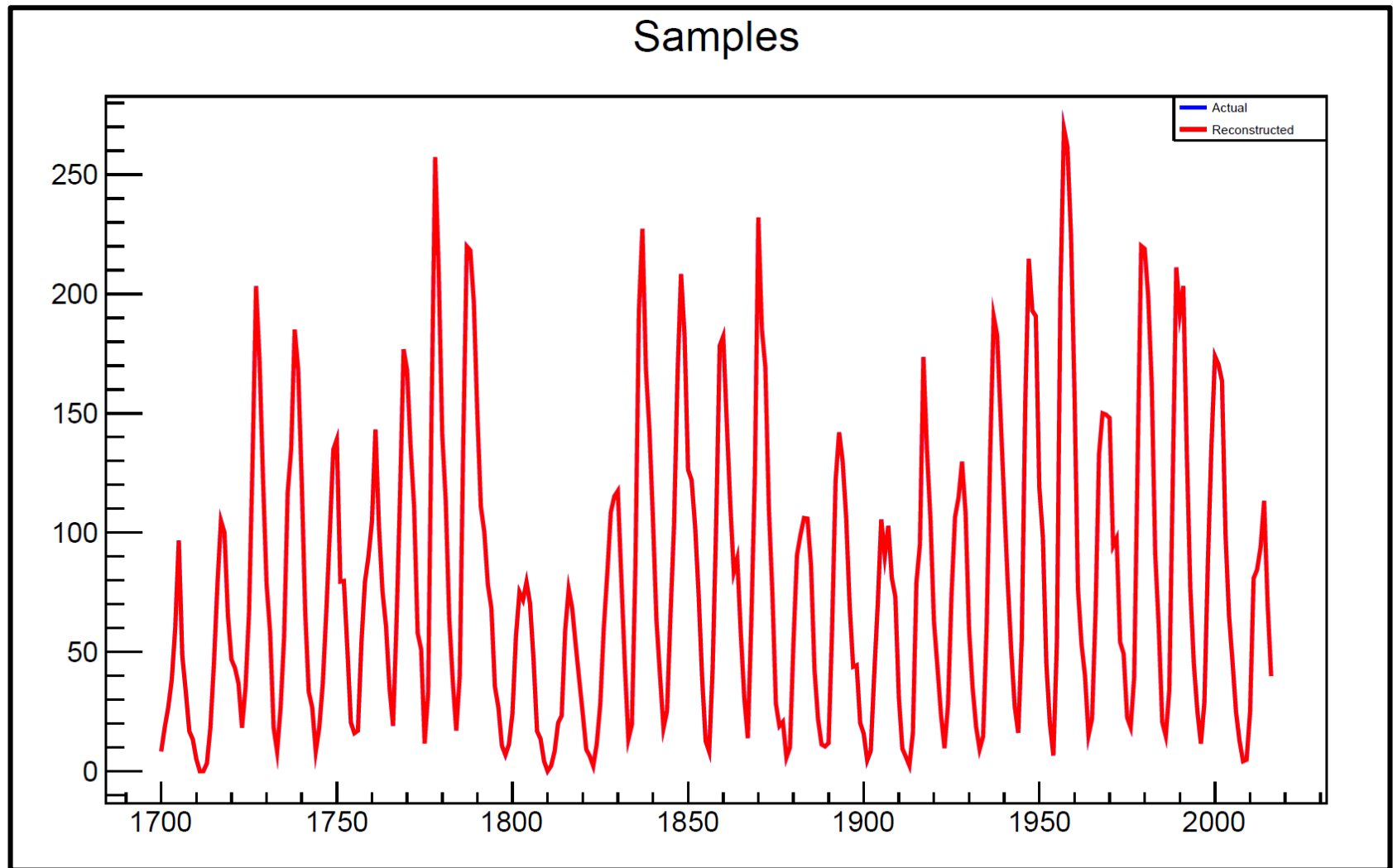
The idea of computing sunspot numbers was originated by [Rudolf Wolf](#) in 1848<sup>[1]</sup> in [Zurich, Switzerland](#) and, thus, the procedure he initiated bears his name (or place). The combination of sunspots and their grouping is used because it compensates for variations in observing small sunspots.

This number has been collected and tabulated by researchers for over 150 years.<sup>[2]</sup>

# Run Lab 4 – Sunspot Activity

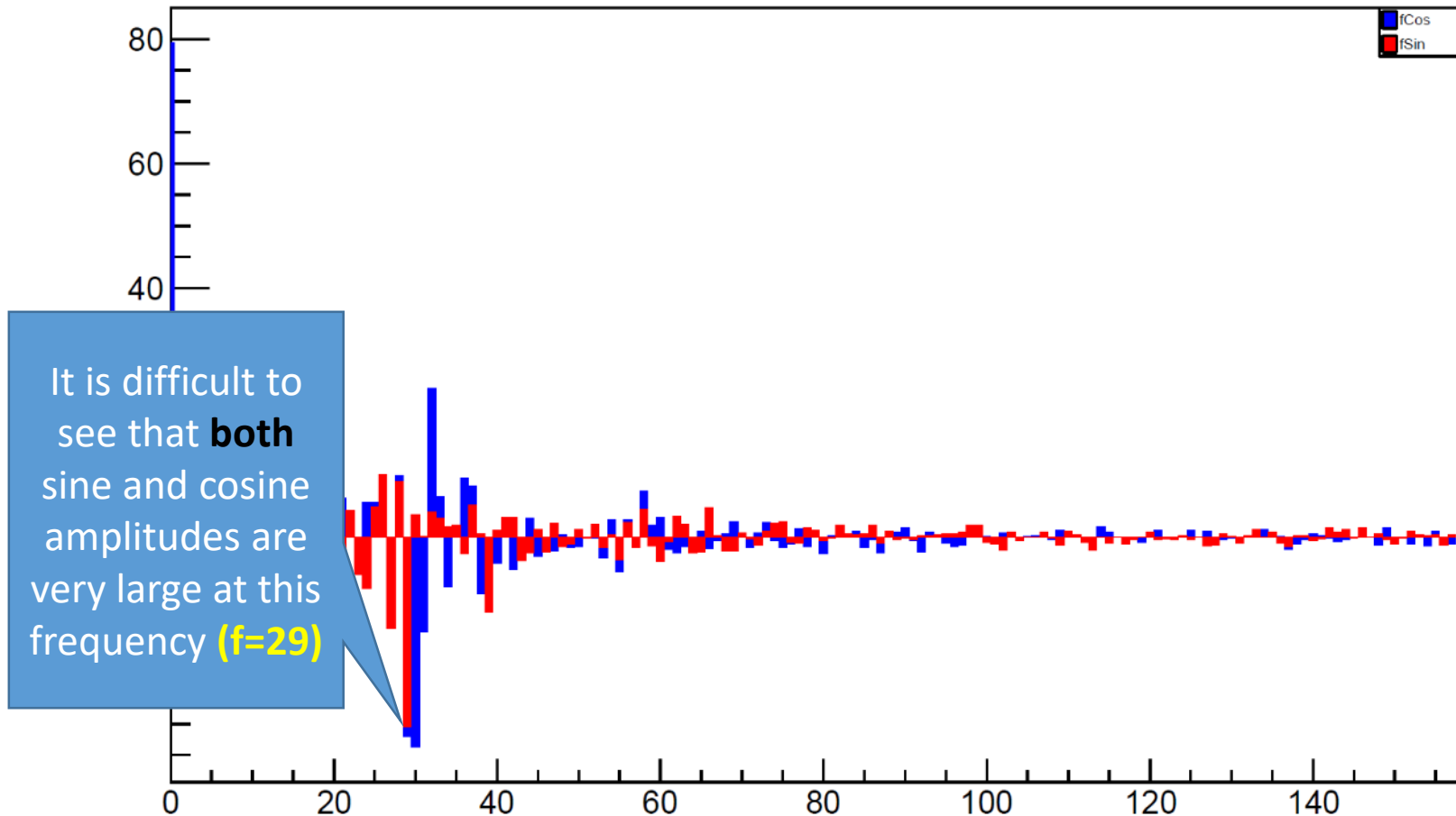


# Check Lab 4 – Sunspot Activity



# Check Lab 4 – Sunspot Activity

## Discrete Fourier Transform



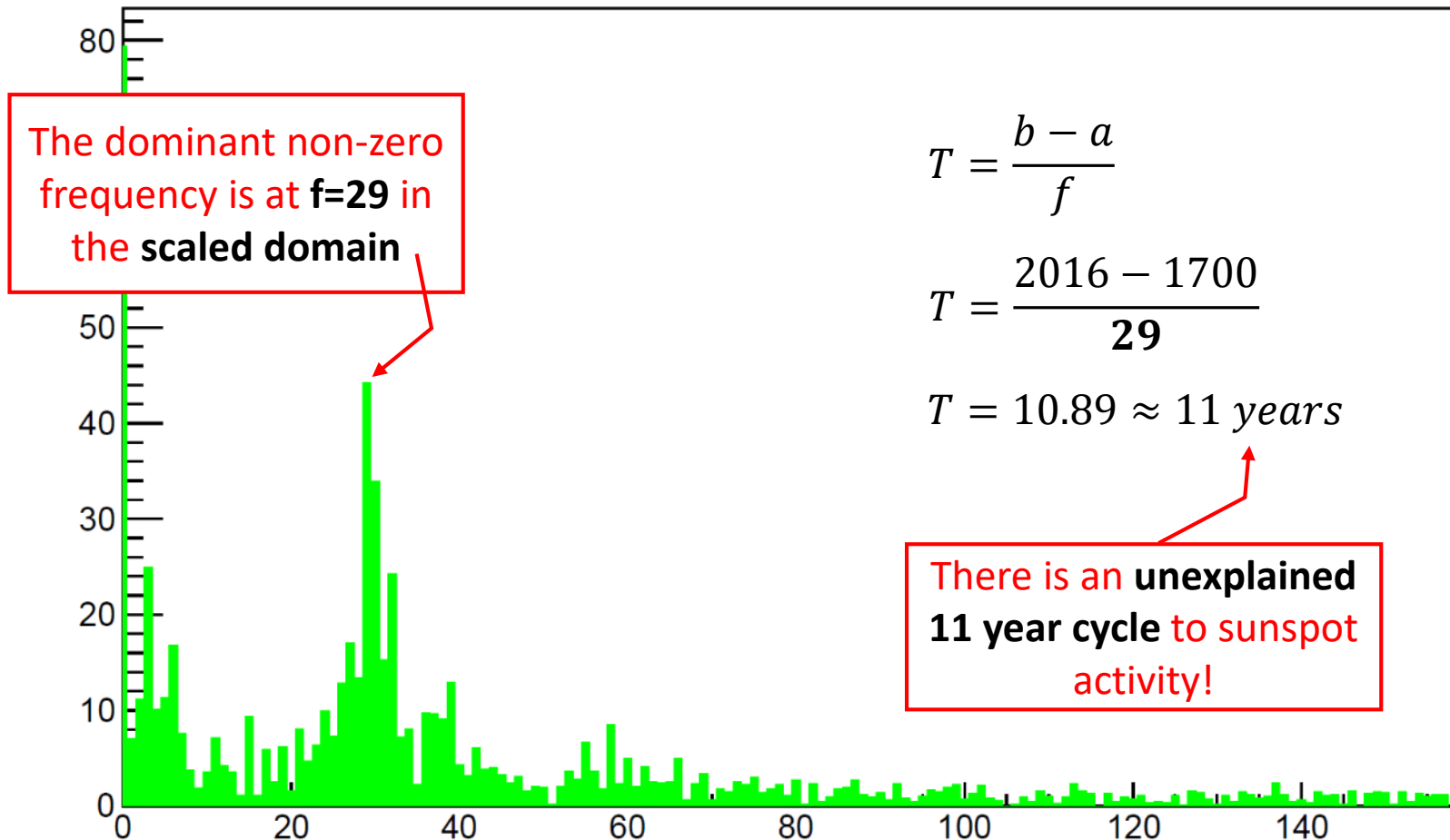
# Power Spectrum

- The green “Power Spectrum” graph is the positive **square root of the sum of the squares** of the cosine and sine amplitudes for each frequency present in the original signal
- High peaks in a power spectrum indicate the significant (**dominant**) frequencies in the original complicated waveform
- Recall the original domain spanned the years **1700 – 2016**
- Can we use the power spectrum graph to determine the **main underlying period** of the recurring sunspot cycles?



# Check Lab 4 – Sunspot Activity

## Power Spectrum



# Every JPEG Image uses Fourier Transforms



(a) Original Lena image



(c) DCT-II compressed Lena image  
(PSNR=32.38dB)



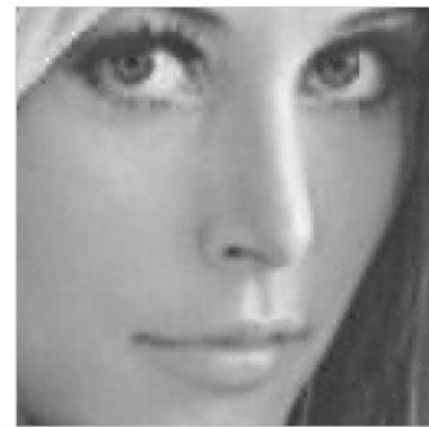
(e) DCT/DST-II compressed Lena image  
(PSNR=35.12dB)



(b) Zoomed original Lena image



(d) Zoomed DCT-II compressed Lena image



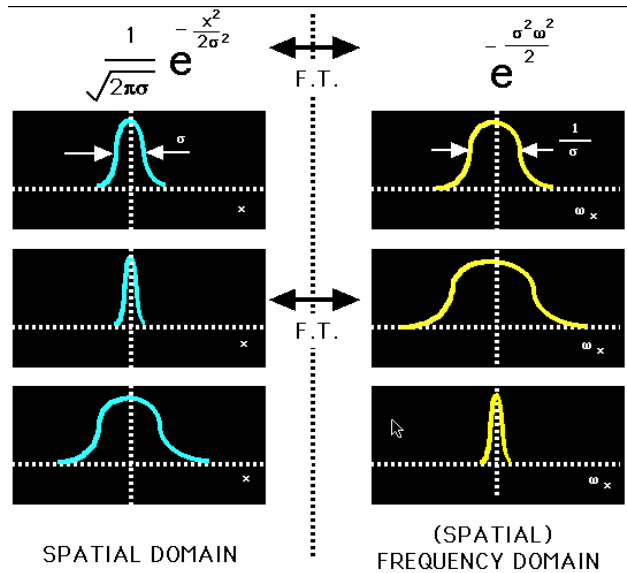
(f) Zoomed DCT/DST-II compressed  
Lena image

# All Streaming Video uses Fourier Transforms

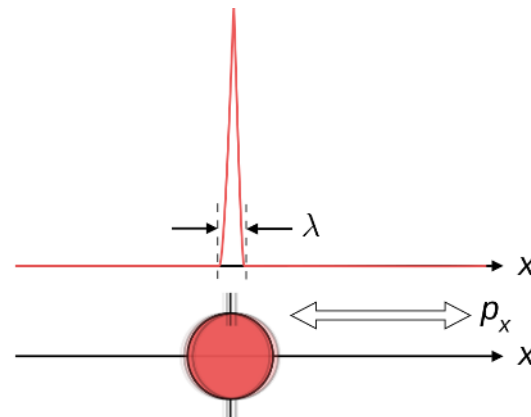
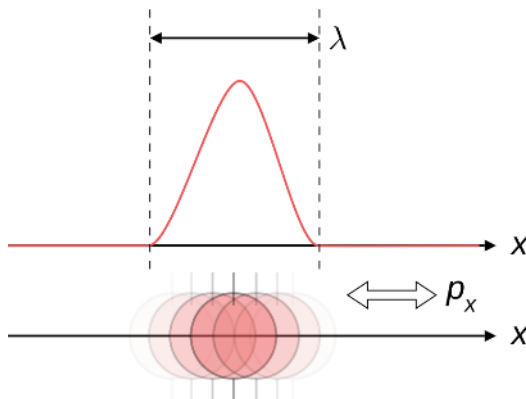
## VP9 quality/bitrate comparisons



# Heisenberg Uncertainty Principle



$$\Delta x \cdot \Delta p \geq h/2$$



# Now you know...

- Fourier Analysis is the **microscope** of scientific computing
  - What appears to be a complicated waveform in the **time domain** might have a simple underlying representation in the **frequency domain**
  - The Fourier Transform determines the amplitudes and wavelengths of the **constituent simple sinusoids** that make up a complicated waveform
  - Discrete Fourier Transform (DFT) can provide insight in the character of the **physical law** generating the observable
- The Fourier Transform is just a special case of the more general **Laplace** Transform that can analyze signals having both sinusoids *and* exponential growth/decay