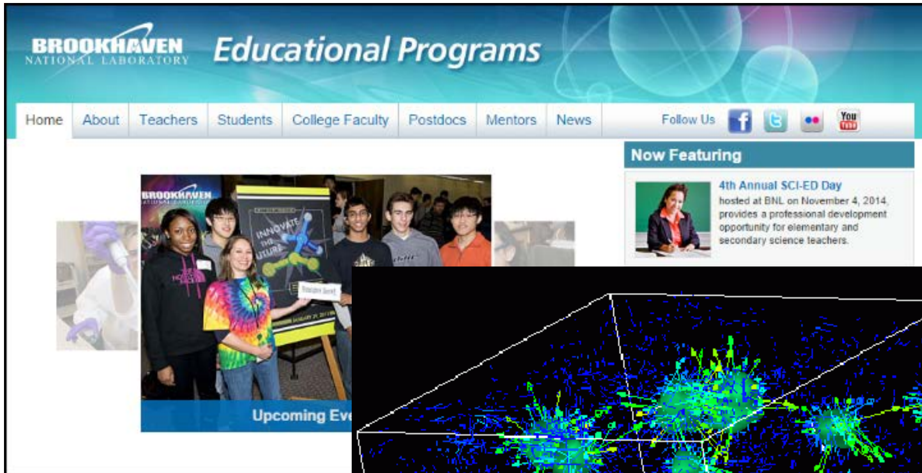
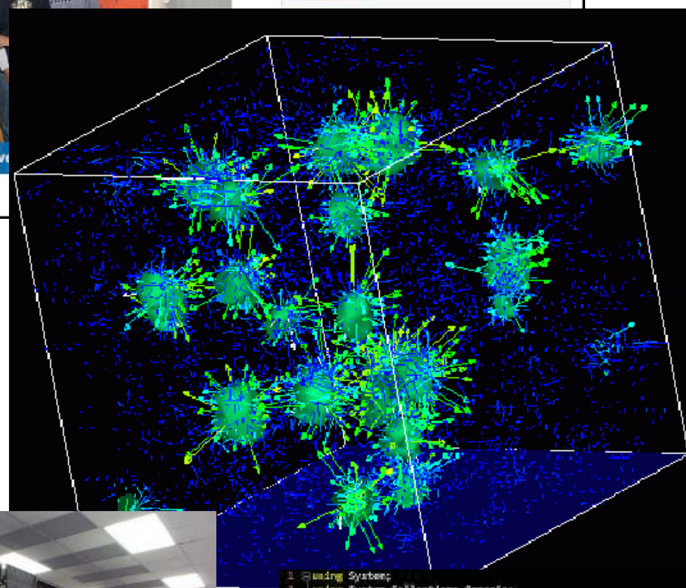
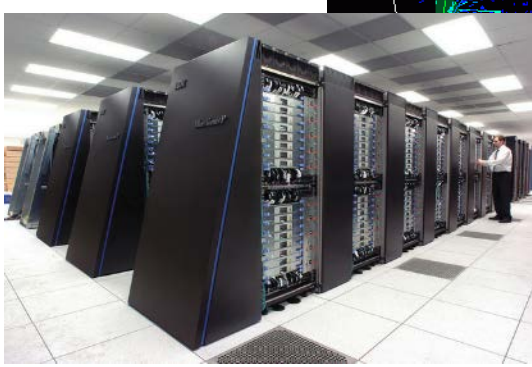






Survey of Scientific Computing

(SciComp 301)

Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

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	Ββ	Γγ	Δδ	Εε	Ζζ
ALPHA	BETA	GAMMA	DELTA	EPSILON	ZETA
Ηη	Θθ	Ιι	Κκ	Λλ	Μμ
ETA	THETA	IOTA	KAPPA	LAMBDA	MU
Νν	Ξξ	Οο	Ππ	Ρρ	Σσ
NU	XI	OMICRON	PI	RHO	SIGMA
Ττ	Υυ	Φφ	Χχ	Ψψ	Ωω
TAU	UPSILON	PHI	CHI	PSI	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

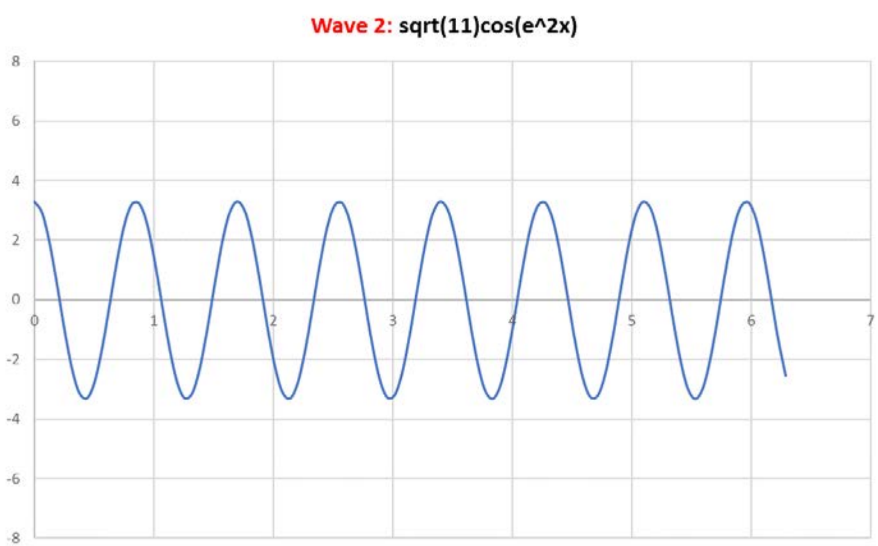
- Electrical
- Magnetic
- Acoustic
- Heat Flow
- Vibrational
- Torsional
- Nuclear / Quantum
- Gravitational
- Oceanic / Tidal
- Orbital Precession
- Springs
- Pendulums
- Image Reconstruction
- Stock Market
- Economics
- Astronomical
- Fluid Dynamics
- Earthquakes
- AC / DC
- AM / FM
- Speech
- Heartbeats

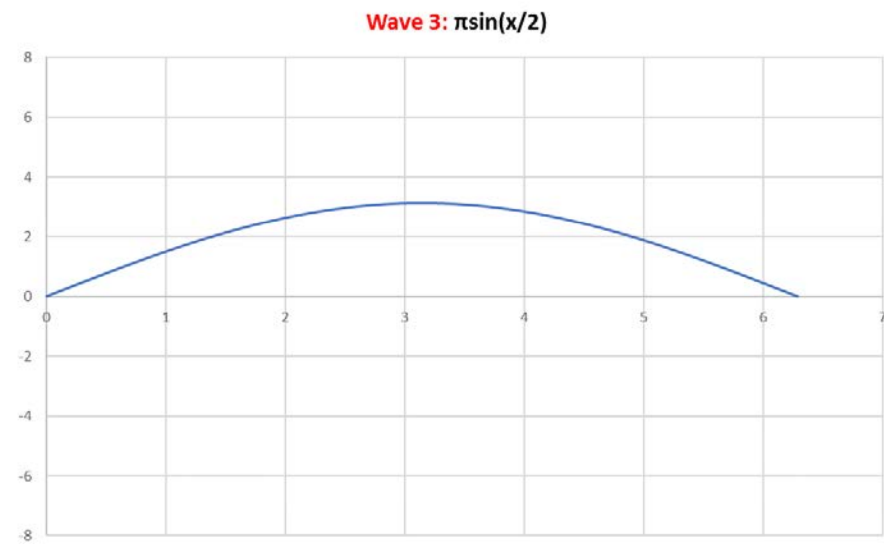
It is really important that you develop a keen understanding of the mathematics of waves!

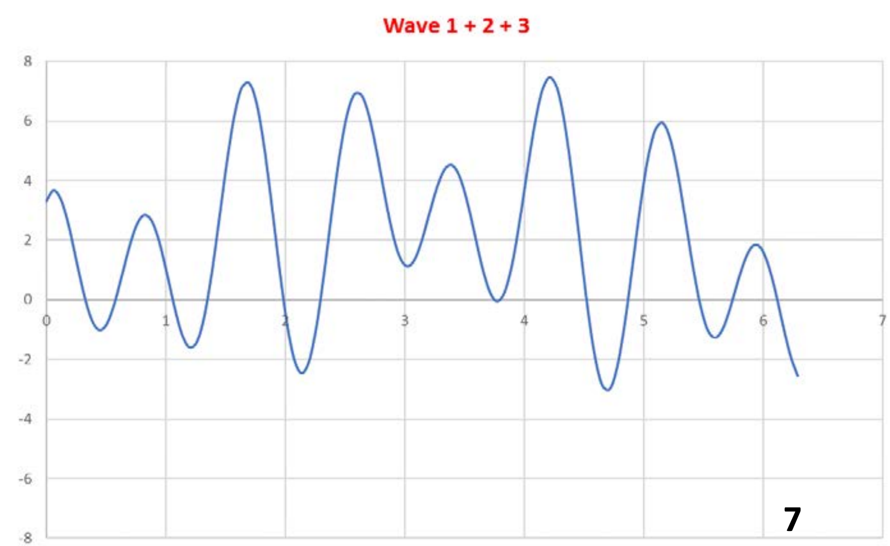
Superposition ⇒ Perceived Complexity

- 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 ⇒ Perceived Complexity



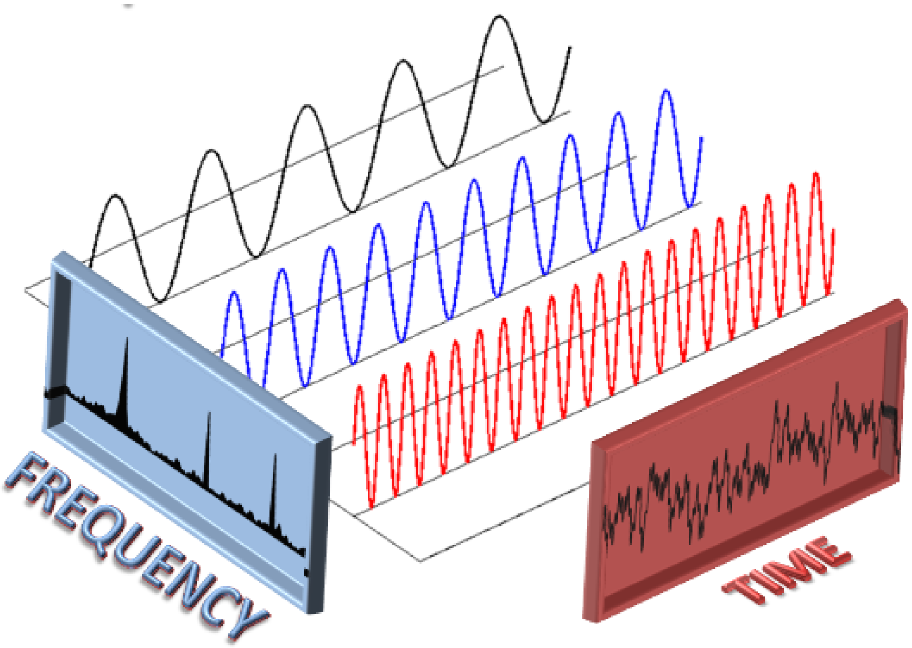




Jean-Baptiste Fourier
(1768-1830)

both r's are silent
"Foo-yeah"

Fourier Transform
Continuous and Discrete

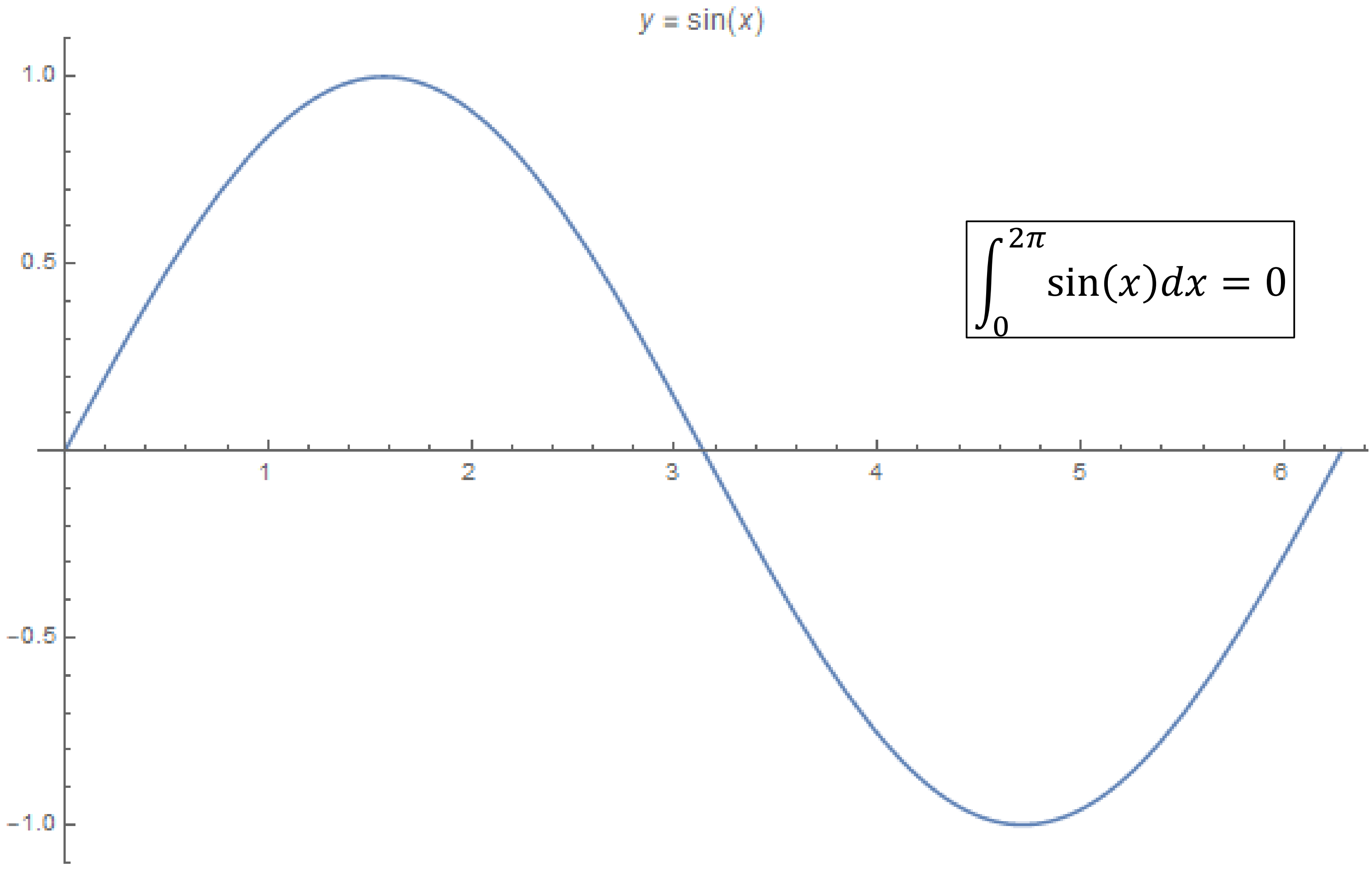


Fourier Analysis

- 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 <u>reconstructing</u> 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 <u>time</u> ***versus*** a sum of amplitudes over integer <u>frequencies</u>
- 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

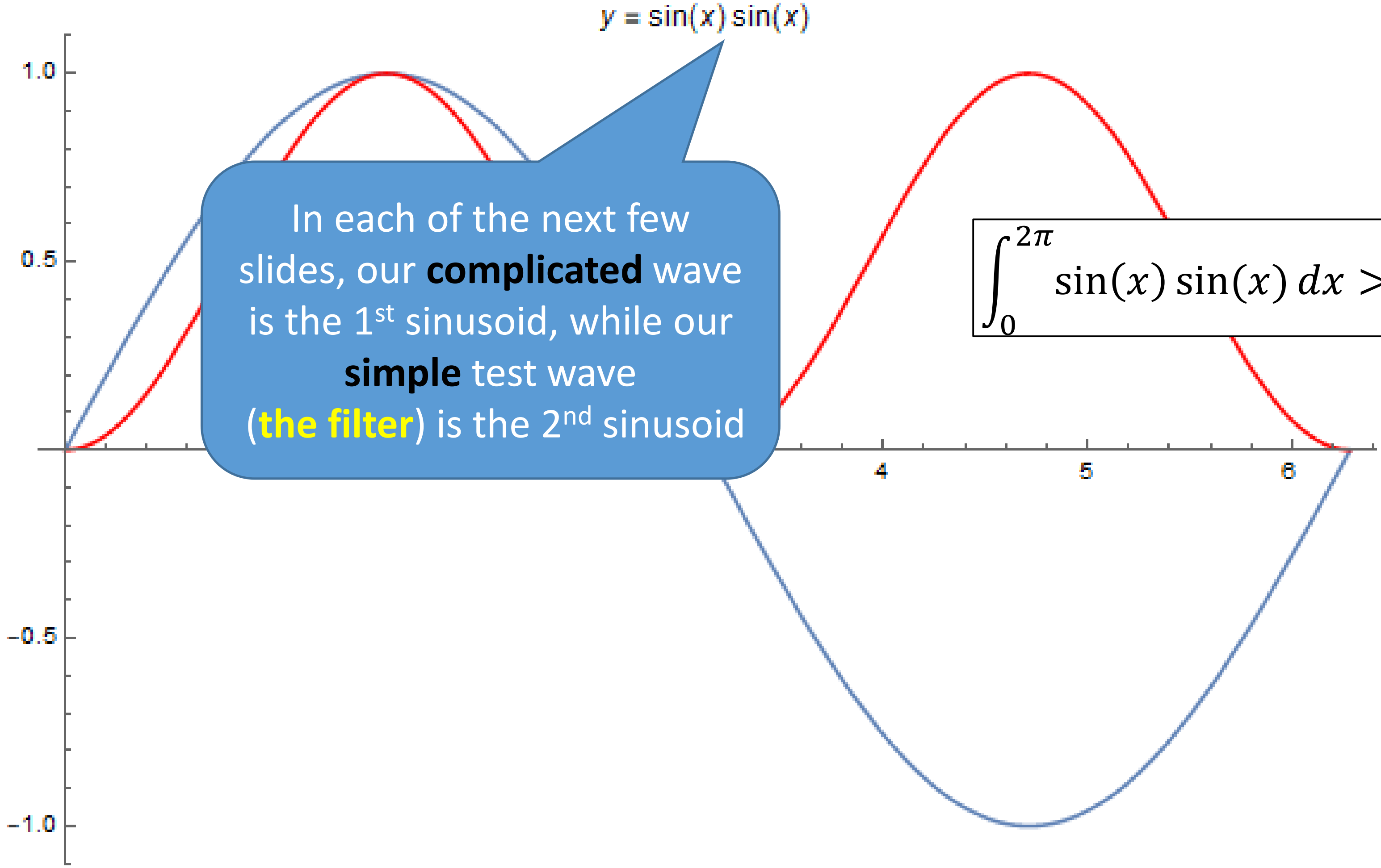
Fourier Analysis - Intuition

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



$$\int_0^{2\pi} \sin(x)dx = 0$$

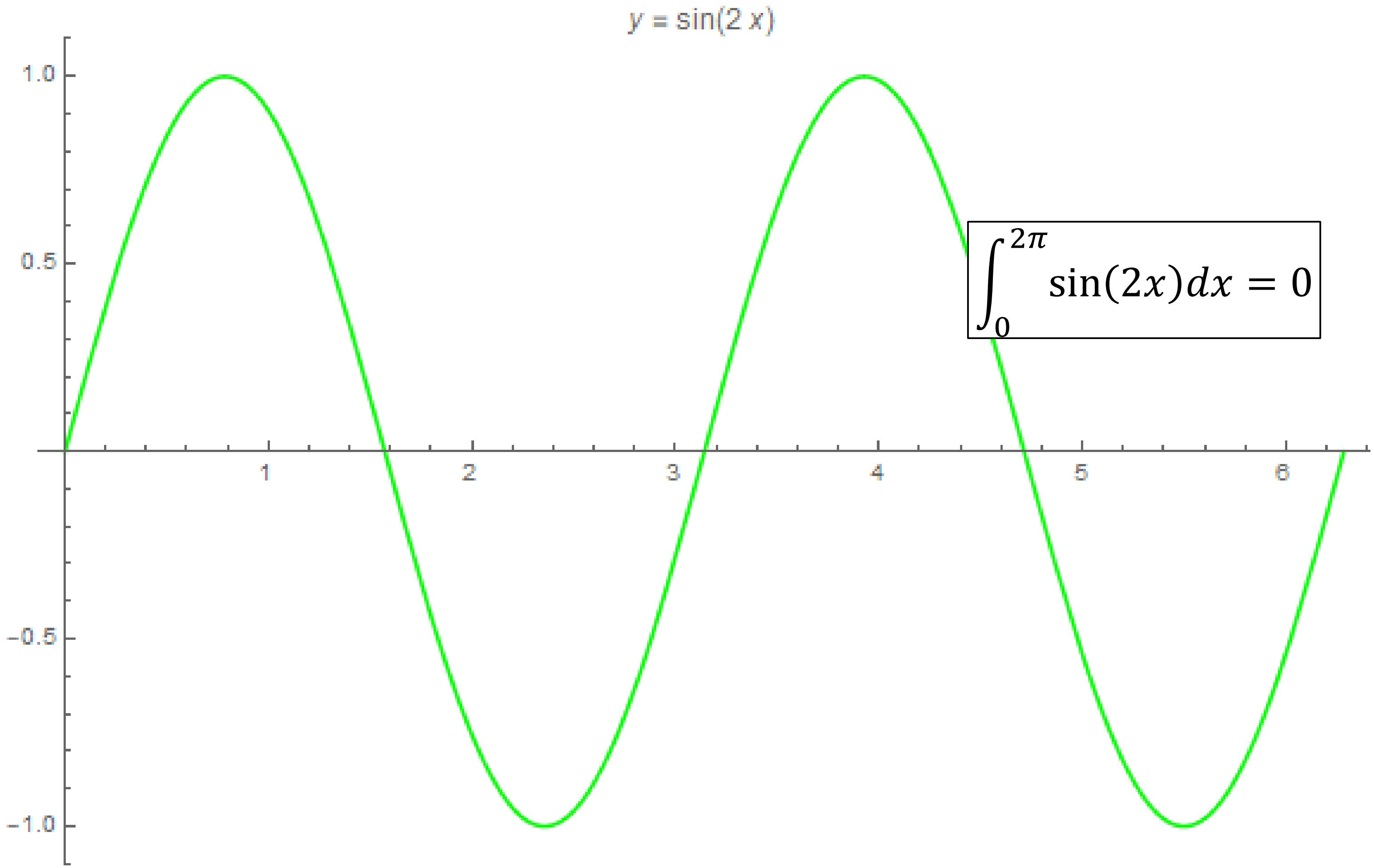
Fourier Analysis - Intuition



In each of the next few slides, our **complicated** wave is the 1st sinusoid, while our **simple** test wave (**the filter**) is the 2nd sinusoid

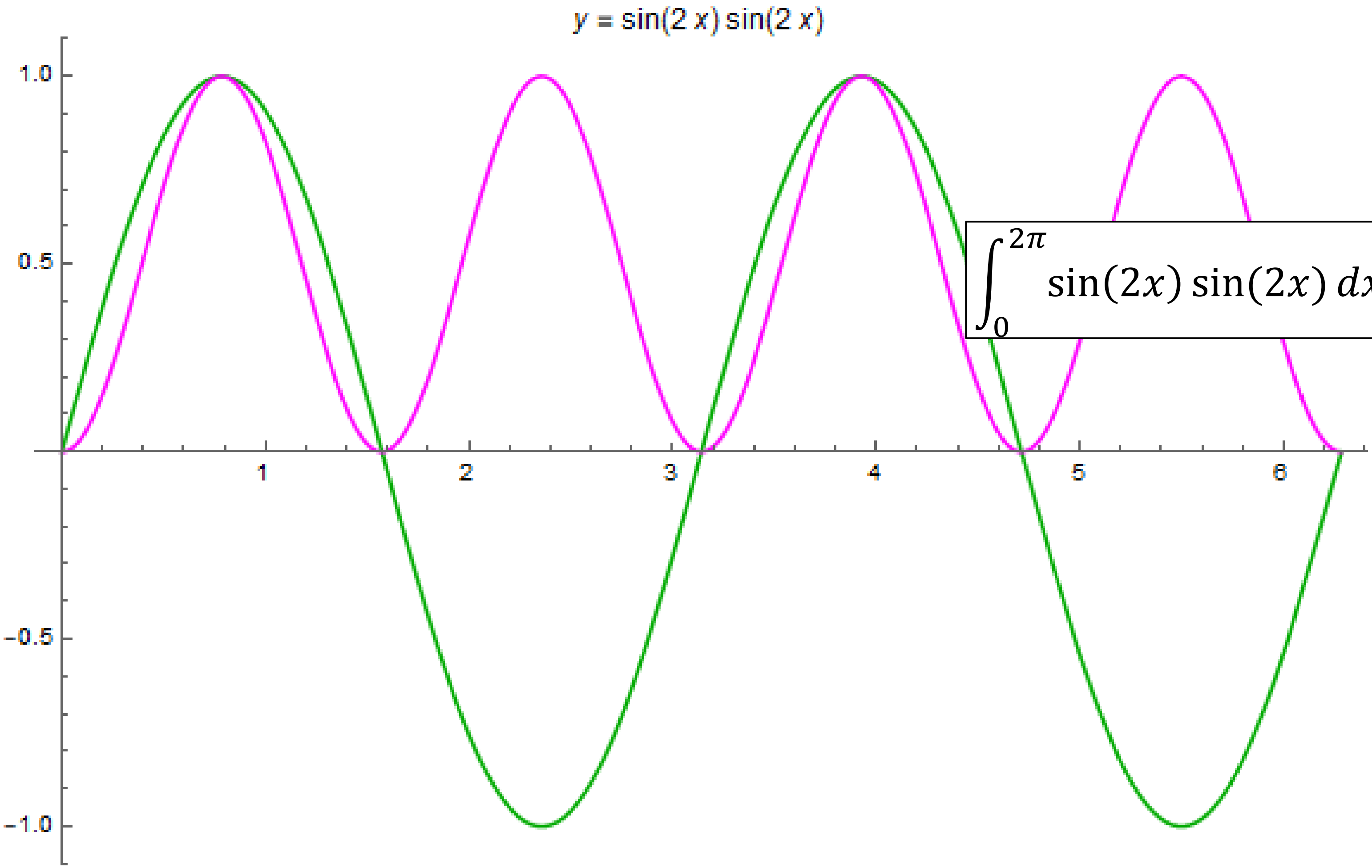
$$\int_0^{2\pi} \sin(x)\sin(x)\,dx > \mathbf{0}$$

Fourier Analysis - Intuition



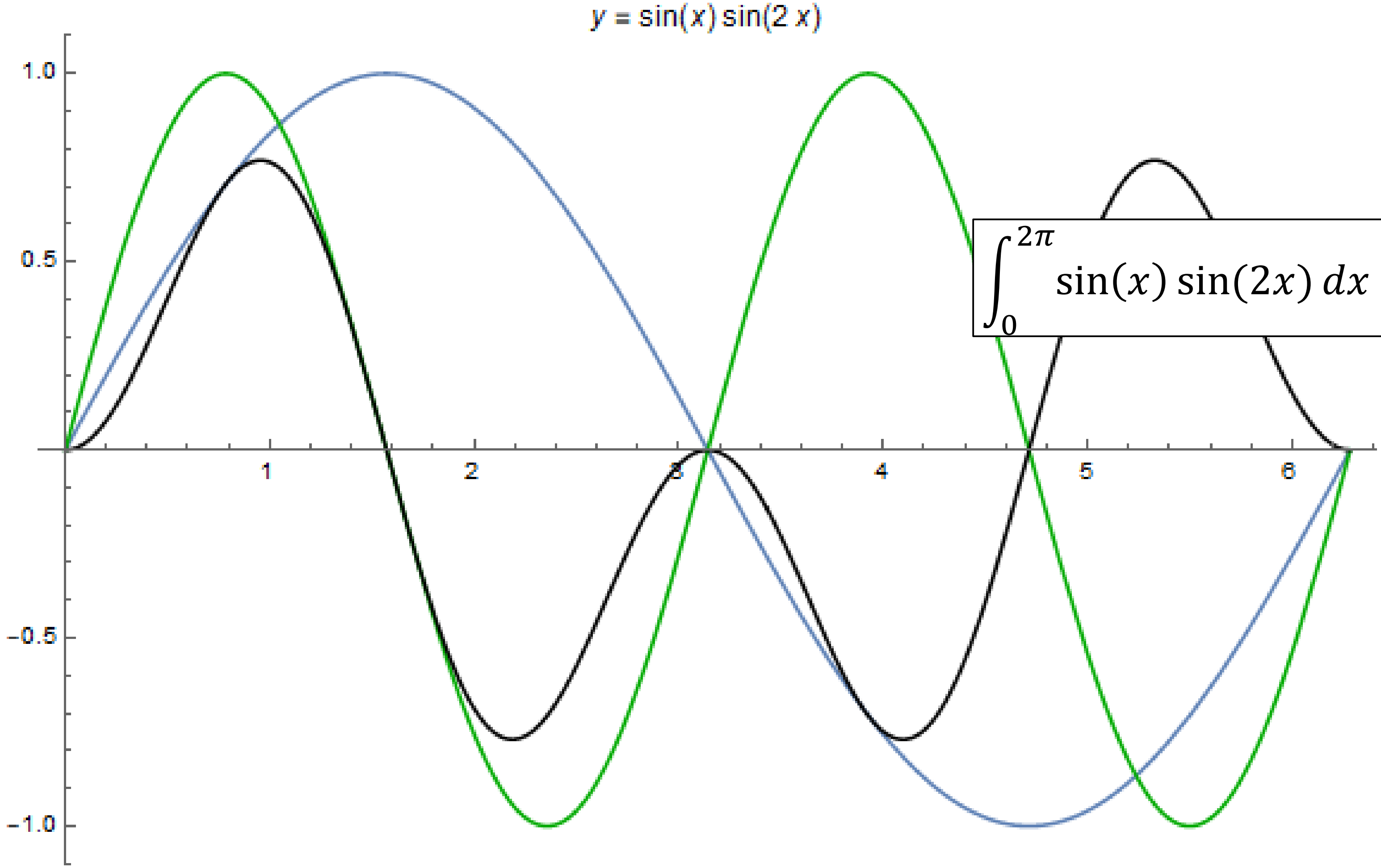
$$\int_0^{2\pi} \sin(2x)dx = 0$$

Fourier Analysis - Intuition



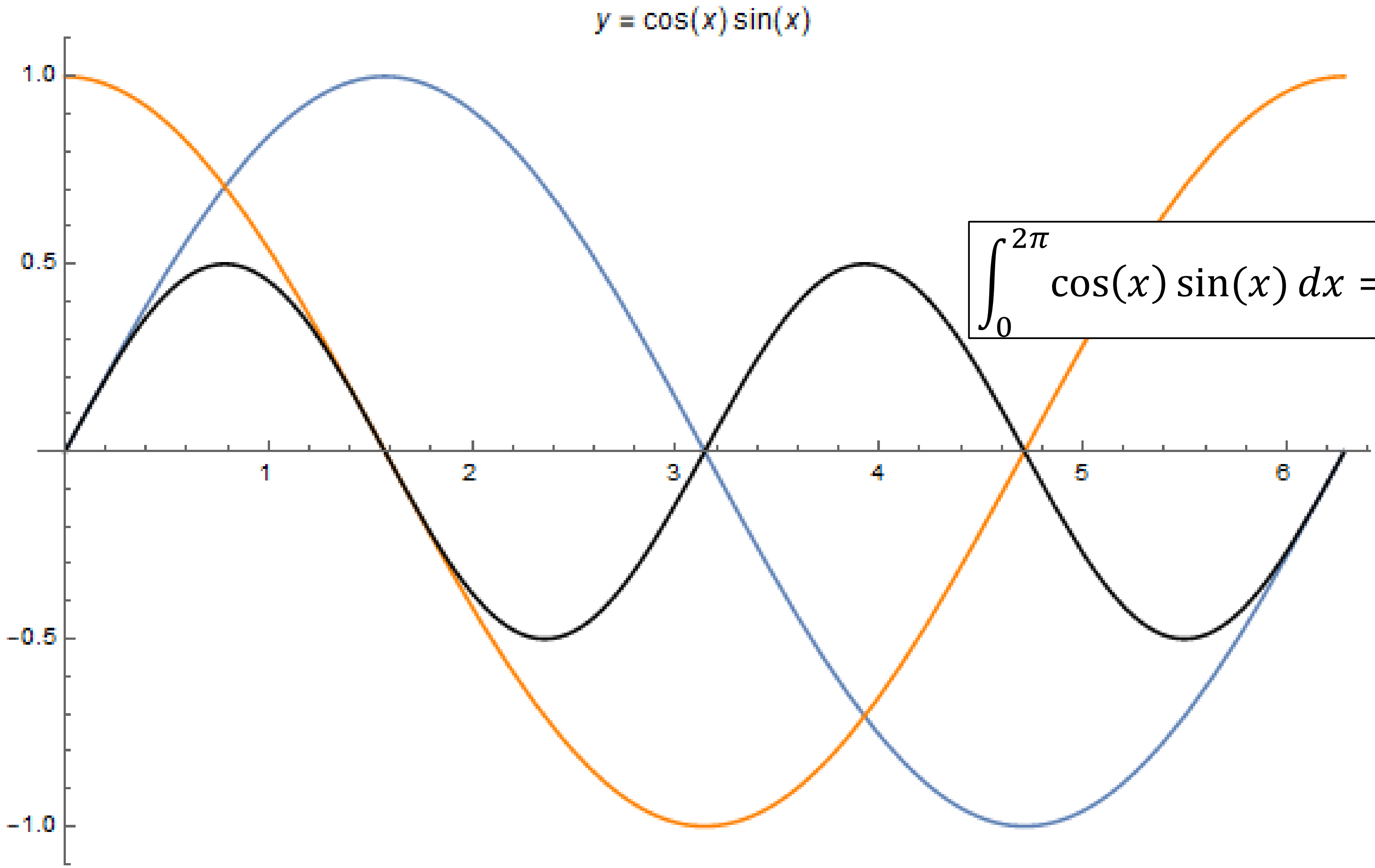
$$\int_0^{2\pi} \sin(2x)\sin(2x)\,dx > \mathbf{0}$$

Fourier Analysis - Intuition



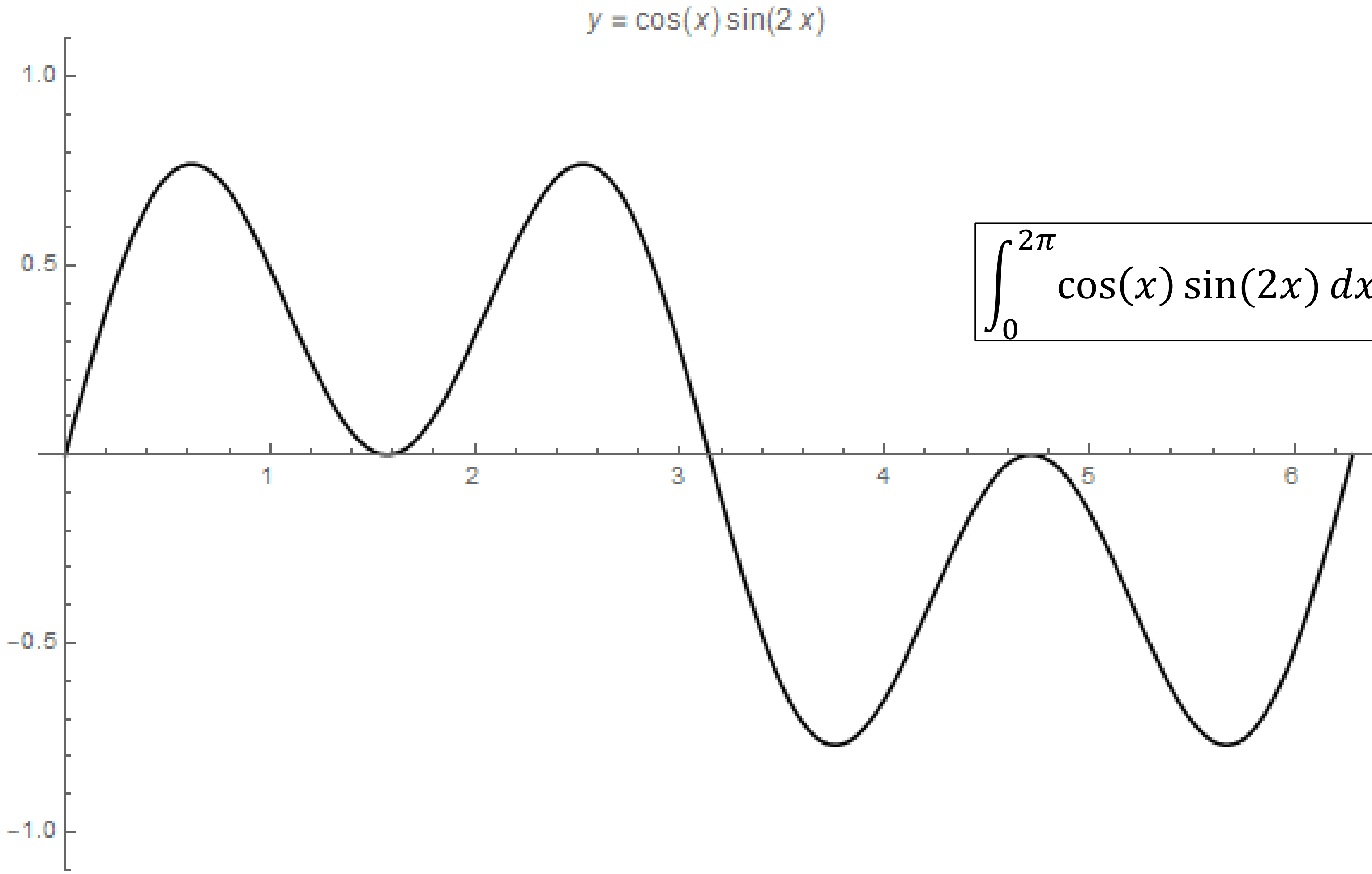
$$\int_0^{2\pi} \sin(x)\sin(2x)\,dx = 0$$

Fourier Analysis - Intuition



$$\int_0^{2\pi} \cos(x)\sin(x)\,dx = 0$$

Fourier Analysis - Intuition



$$\boxed{\int_0^{2\pi} \cos(x)\sin(2x)\,dx = 0}$$

Fourier Analysis - Intuition

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

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

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

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 an **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)

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

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

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

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

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

Euler's Formula

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

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

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

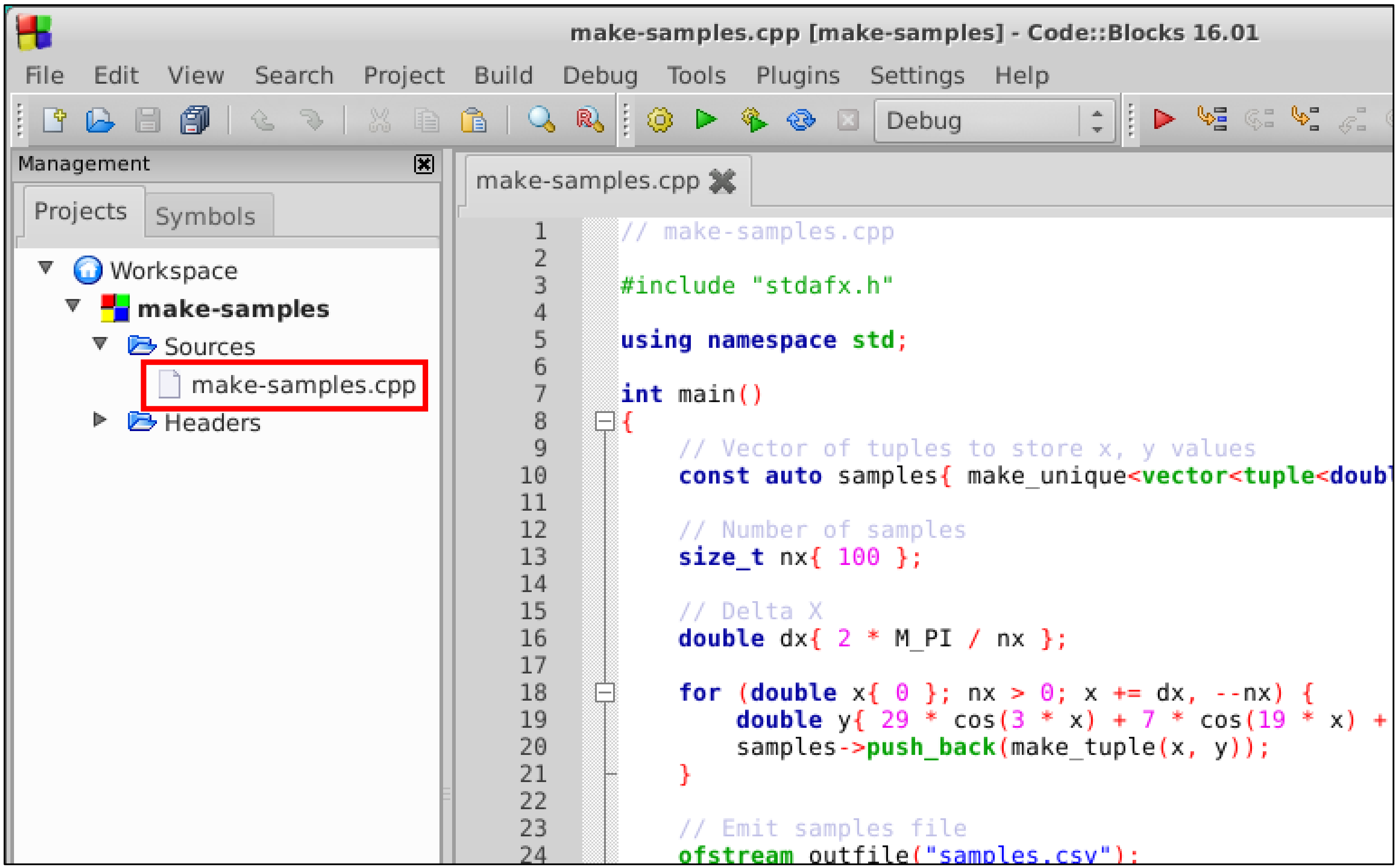
DFT (complex)

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

Inverse DFT (complex)

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

Open Lab 1 - Make Samples for Waveform

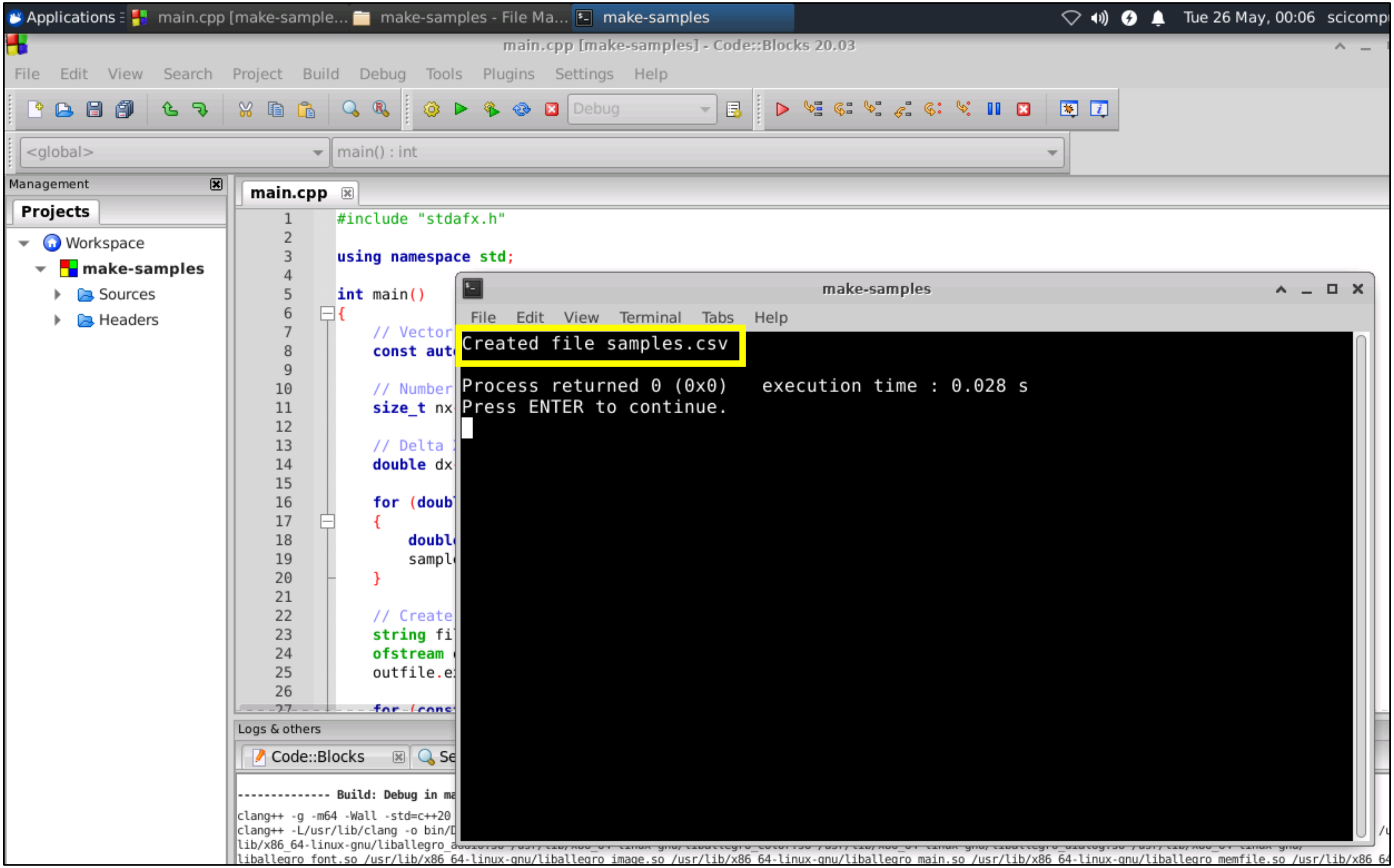


View Lab 1 - Make Samples for Waveform

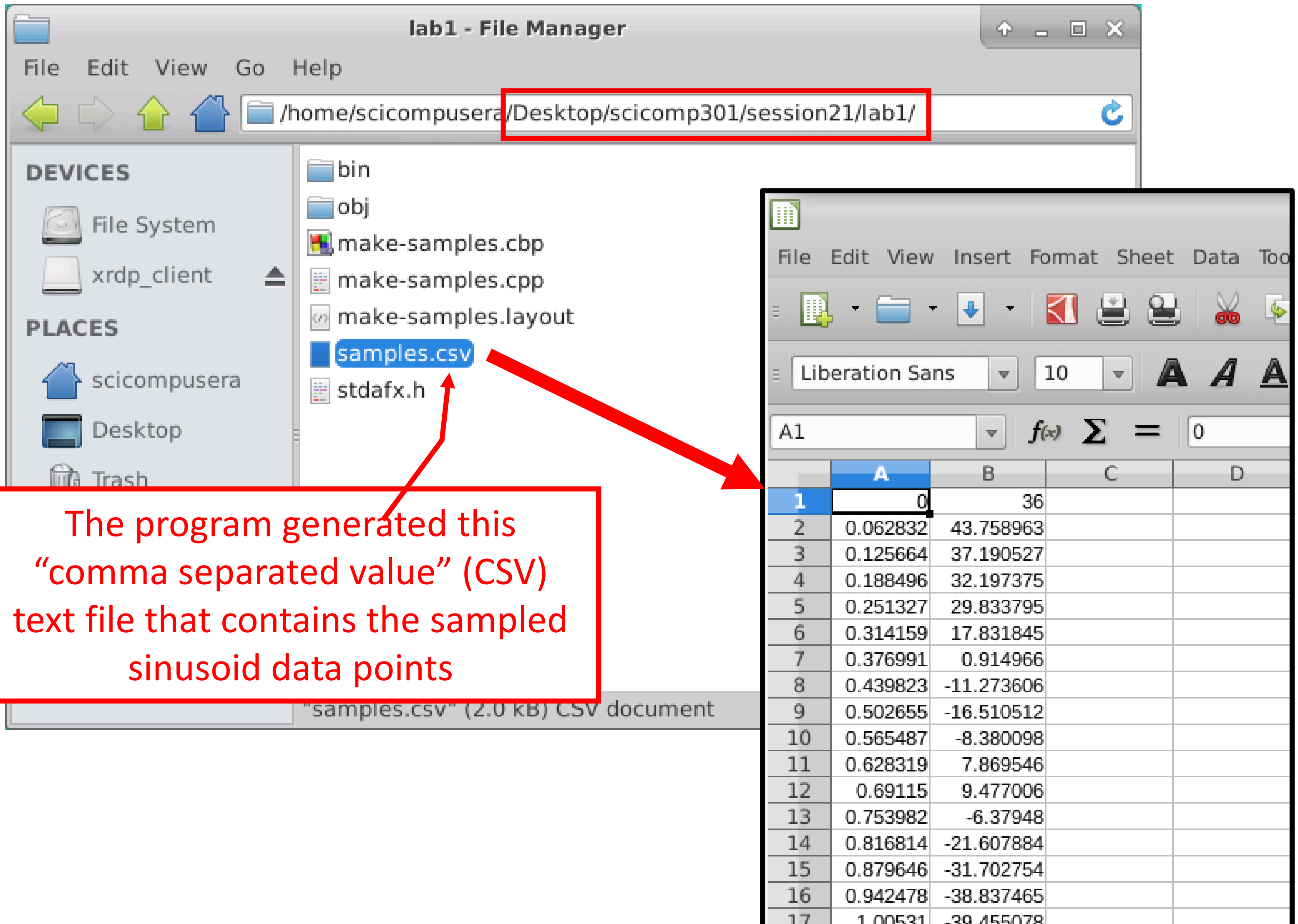
```
int main()
{
    // Vector of tuples to store x, y values
    const auto samples{make_unique<vector<tuple<double, double>>>()};

    // 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

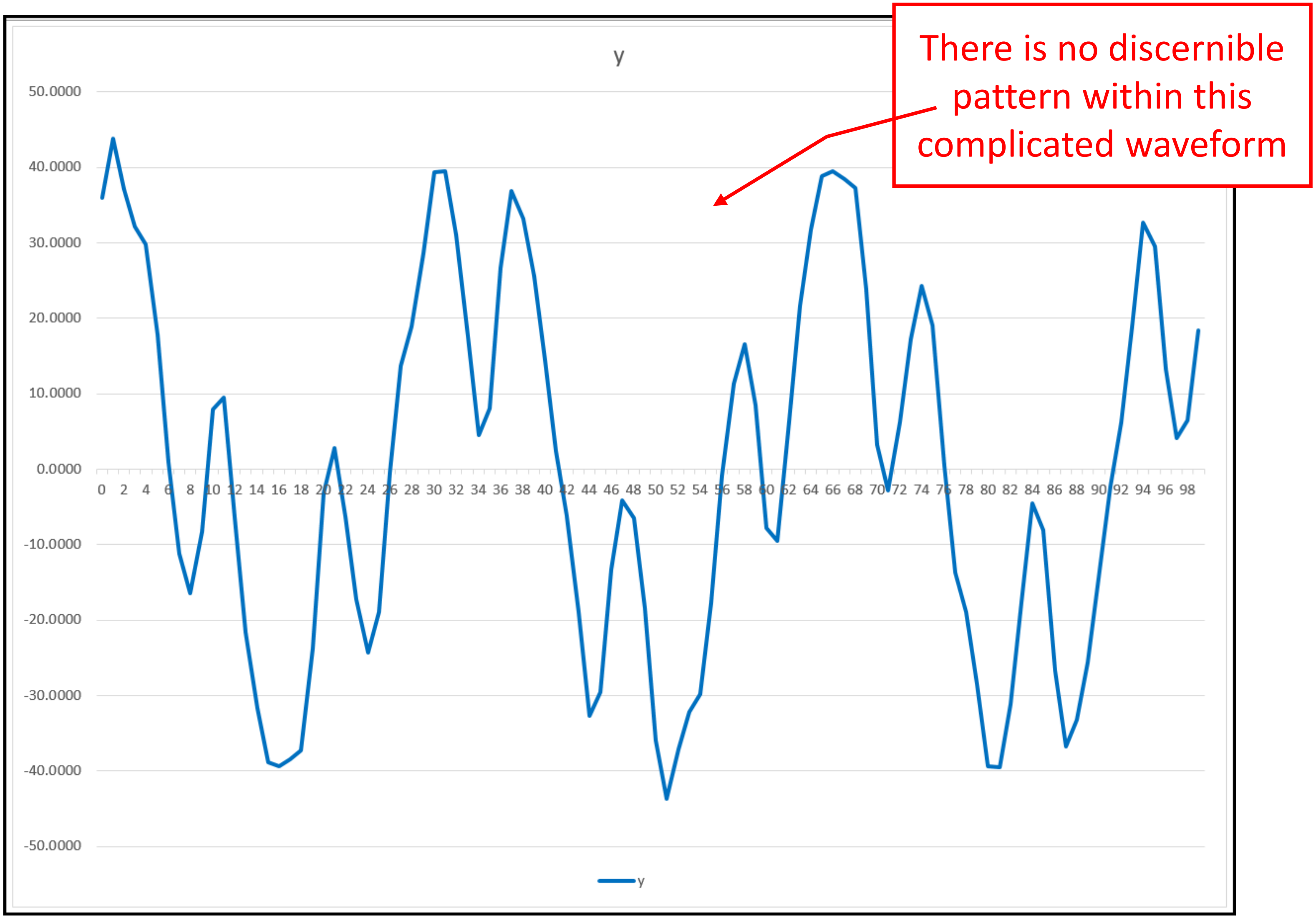


Check Lab 1 - Make Samples for Waveform



The program generated this "comma separated value" (CSV) text file that contains the sampled sinusoid data points

100 Samples of a Complicated Wave



Lab 2 – Discrete Fourier Transform

ROOT
Data Analysis Framework

Visual Studio Code

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;    // Frequence 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(fsin);
    }
}
```

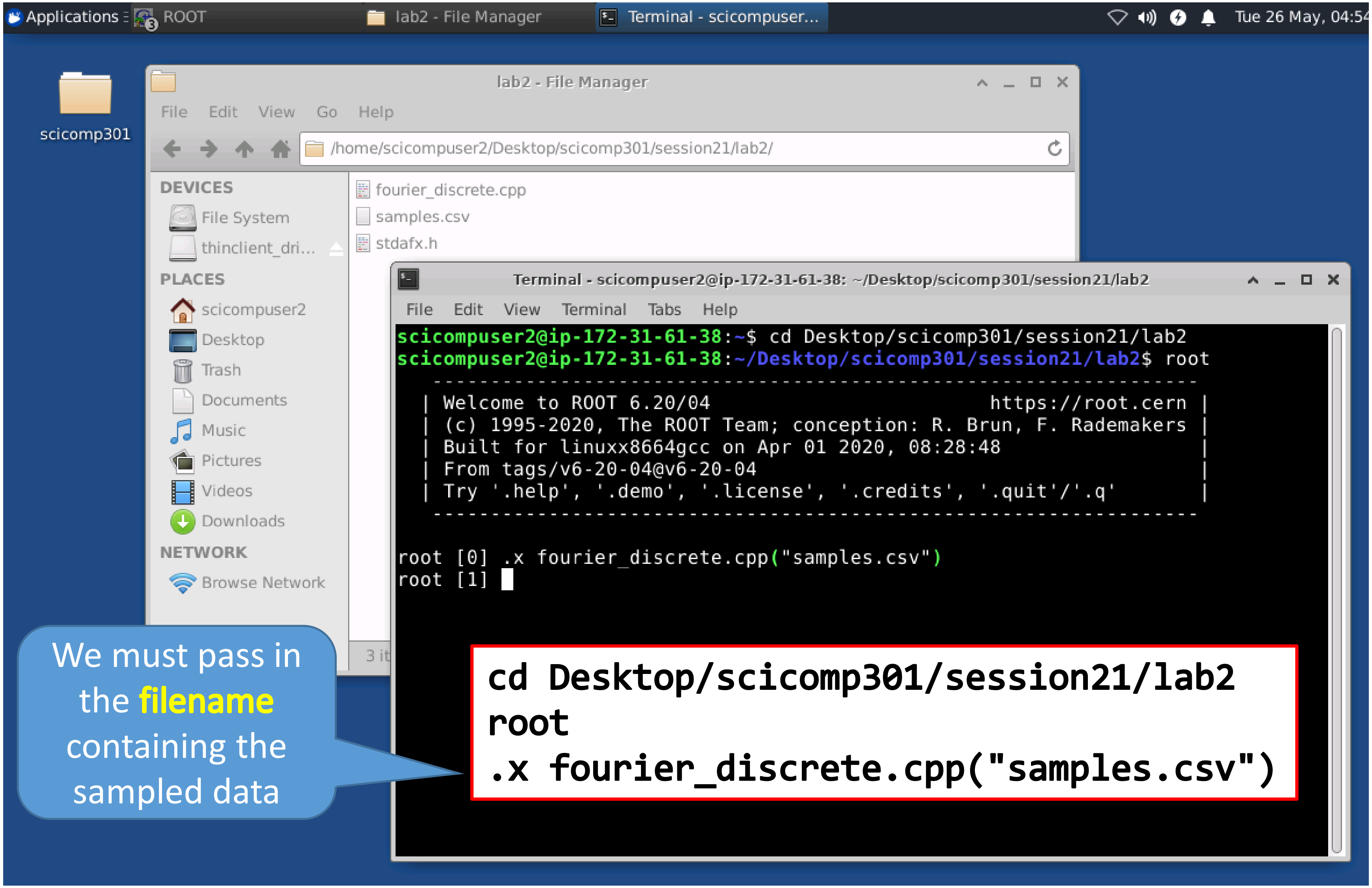
The amplitude of each simple wave is just the **mean value** of the product of every sample of the complicated wave multiplied by that simple wave

View Lab 2 – Discrete Fourier Transform

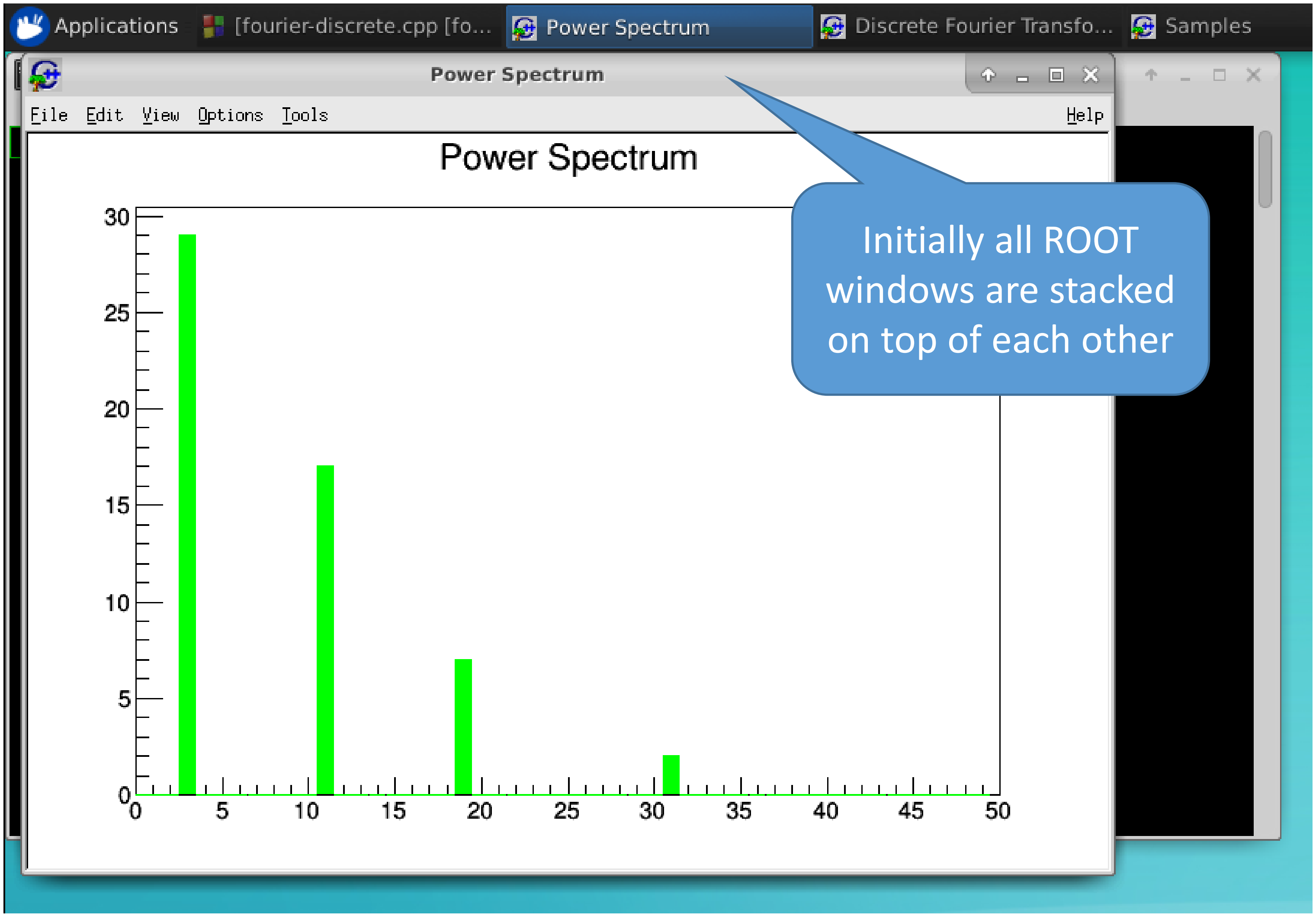
```
void CalcIDFT()
{
    size_t sample_count{yAct.size()};
    size_t term_count{fCos.size()};

    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

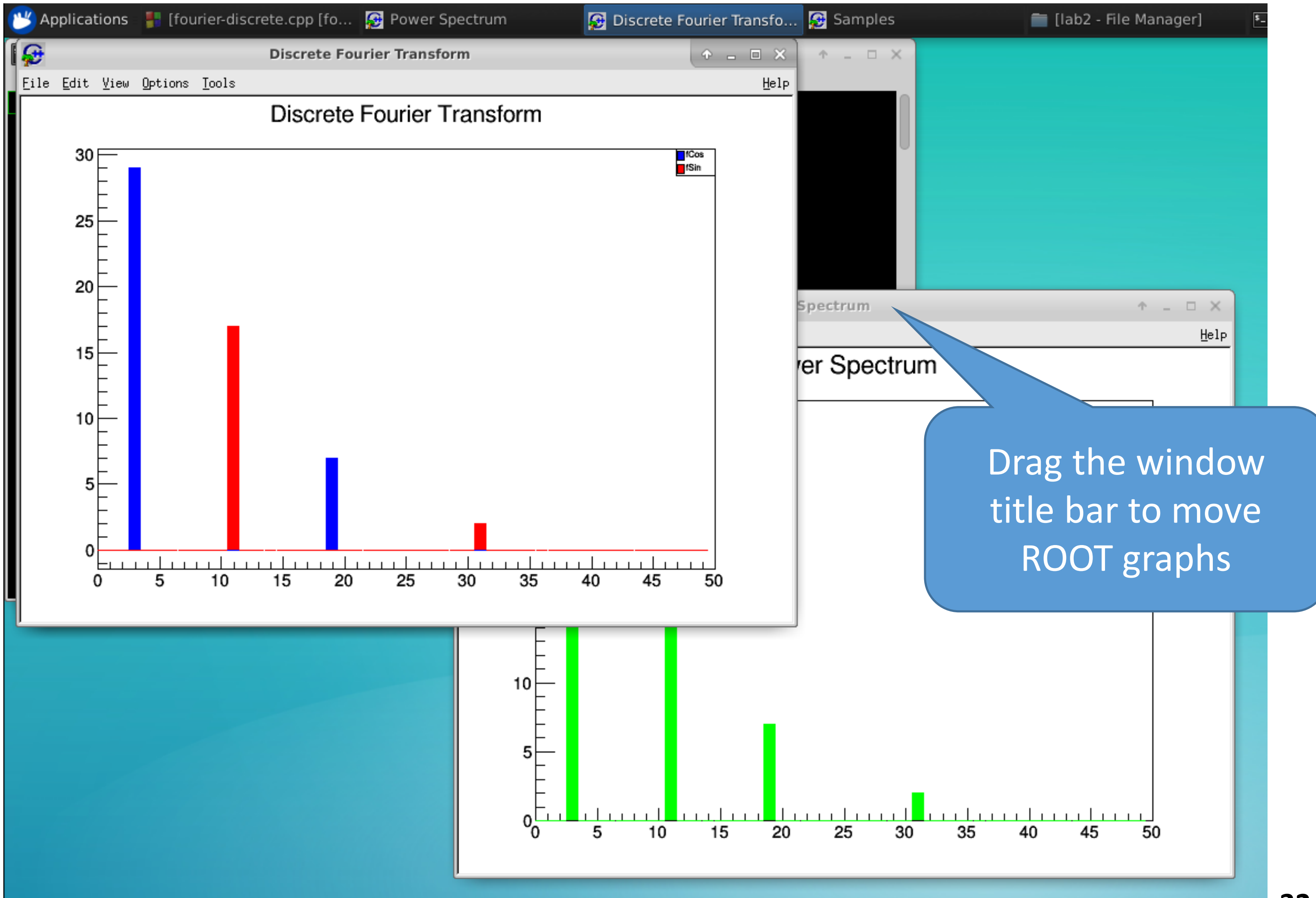
Run Lab 2 – Discrete Fourier Transform

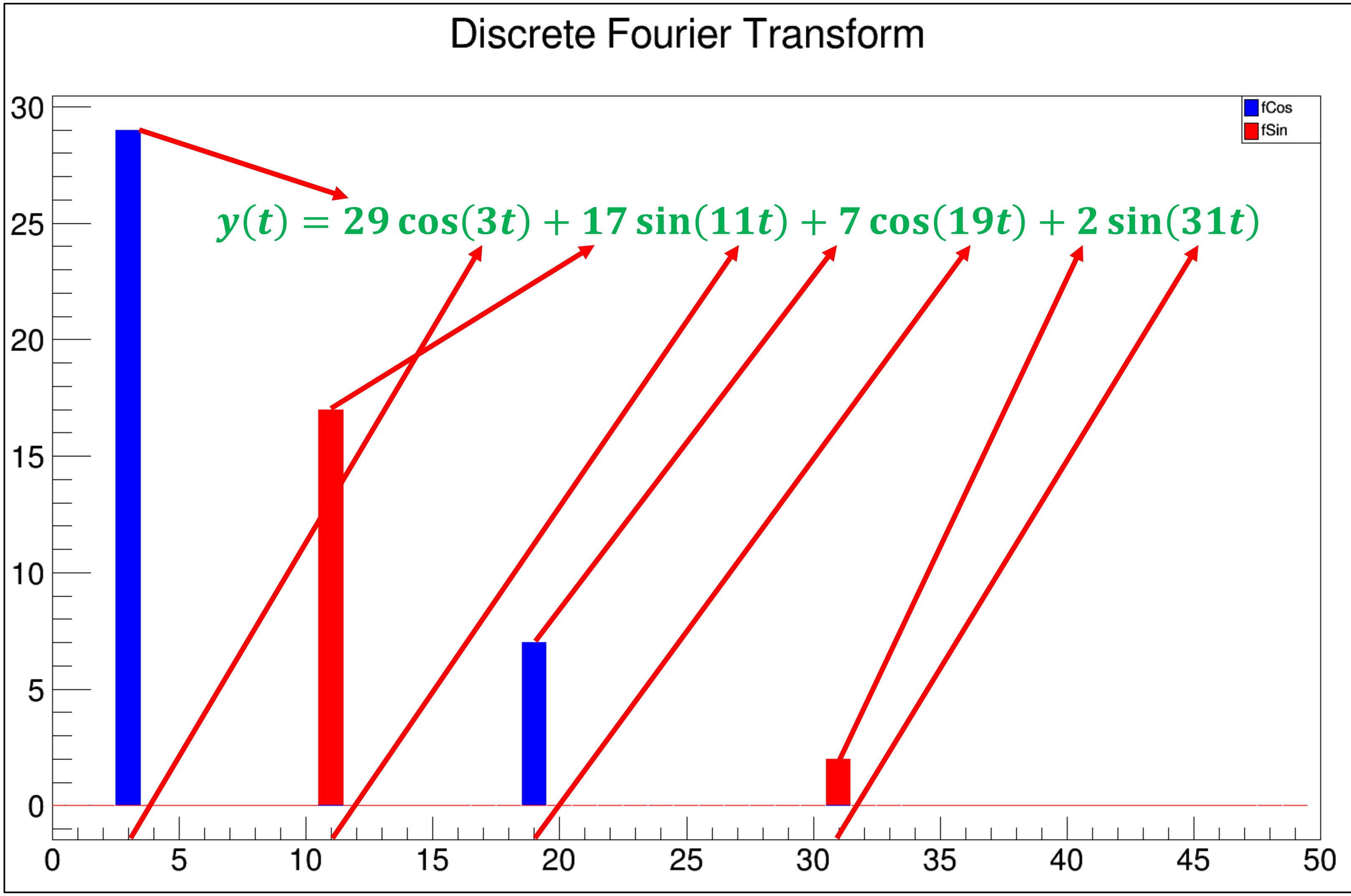


Run Lab 2 – Discrete Fourier Transform

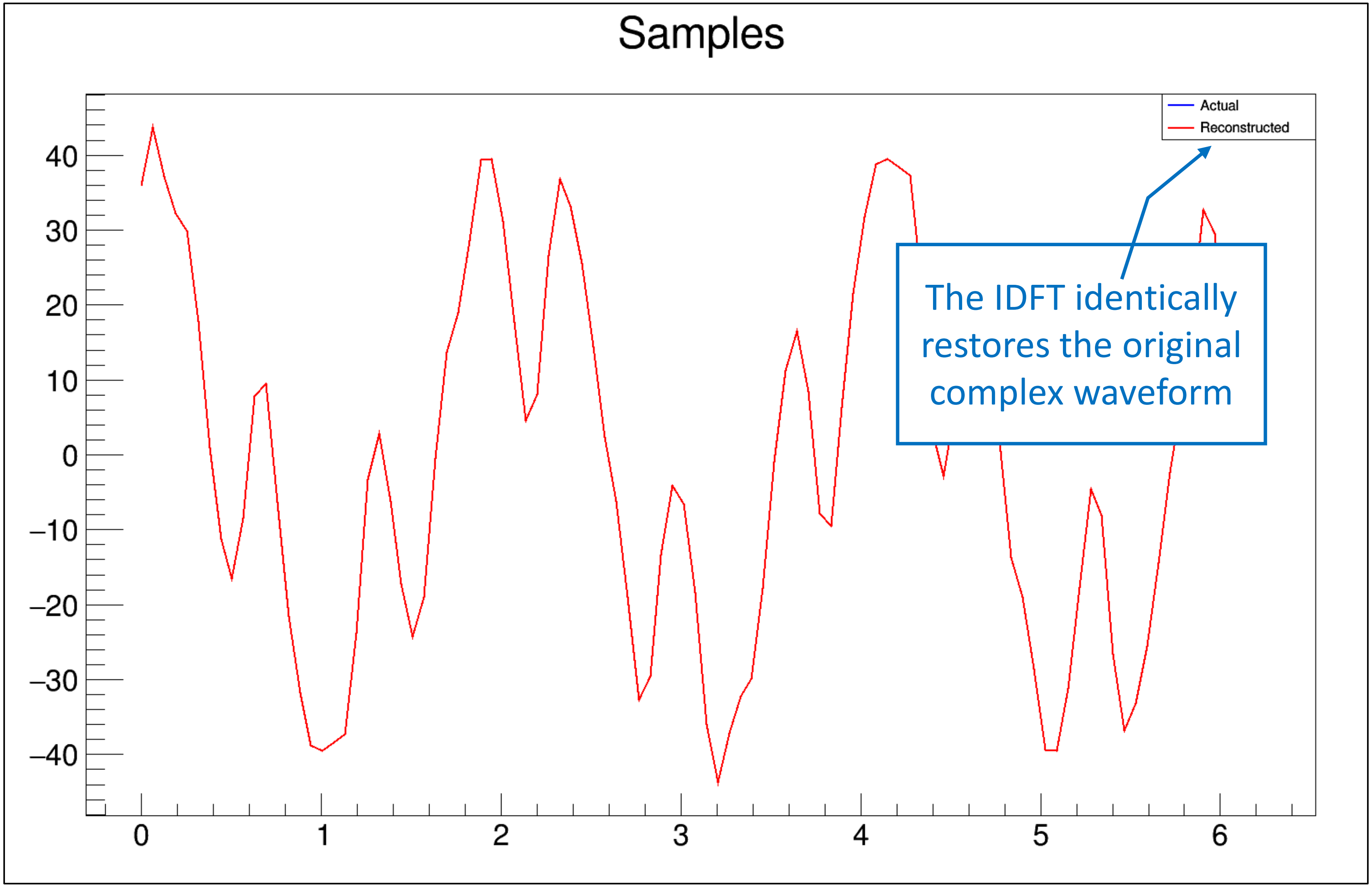


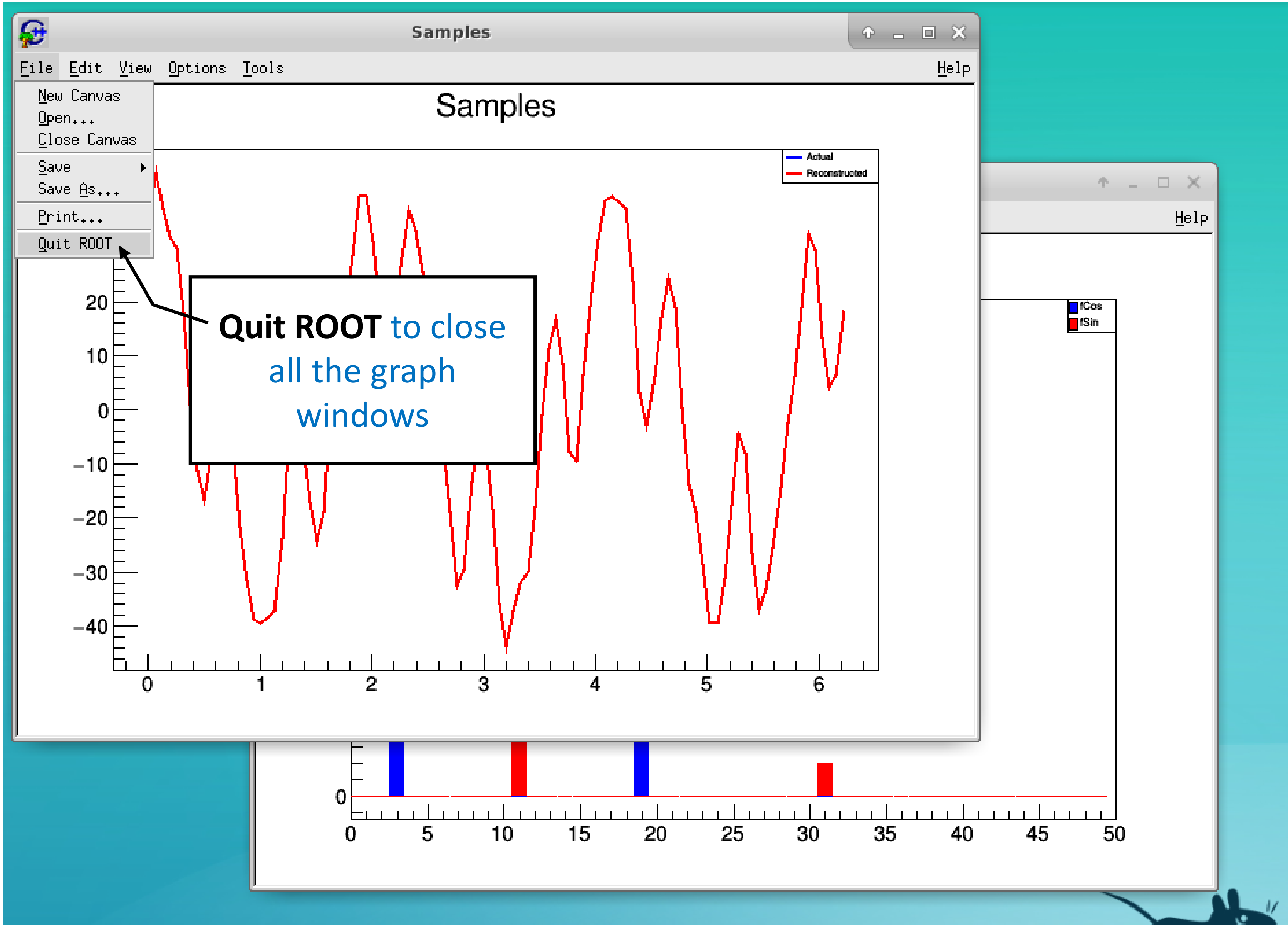
Check Lab 2 – Discrete Fourier Transform




Discrete Fourier Transform
fCos
fSin
$y(t) = 29\cos(3t) + 17\sin(11t) + 7\cos(19t) + 2\sin(31t)$
30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
40
45
50

Check Lab 2 – Discrete Fourier Transform

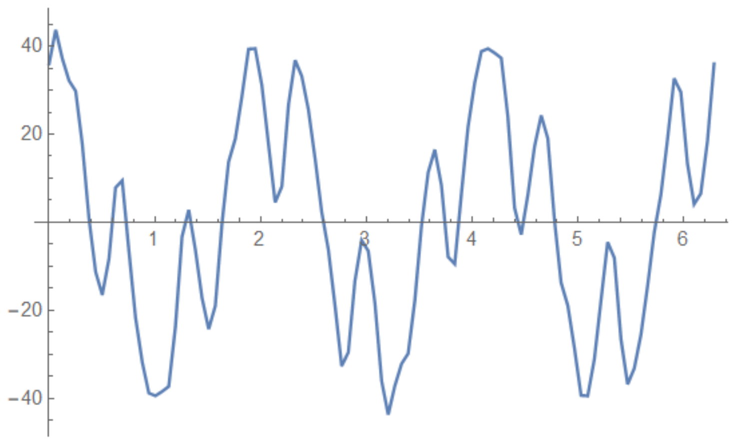




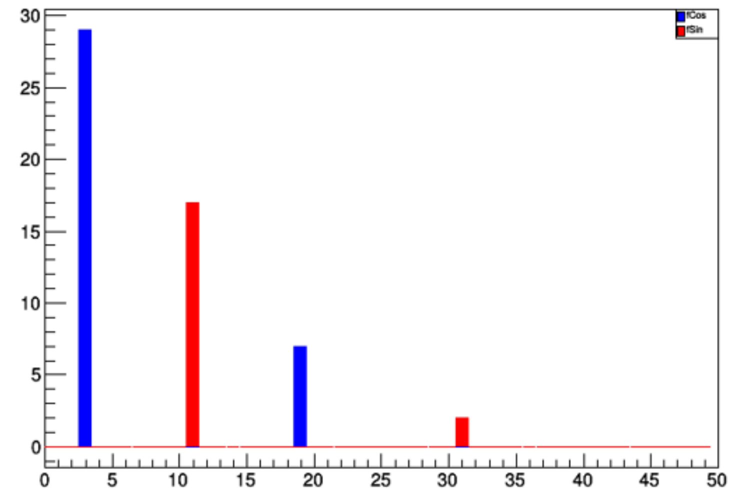
Quit ROOT to close all the graph windows

Information Density

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



Time domain



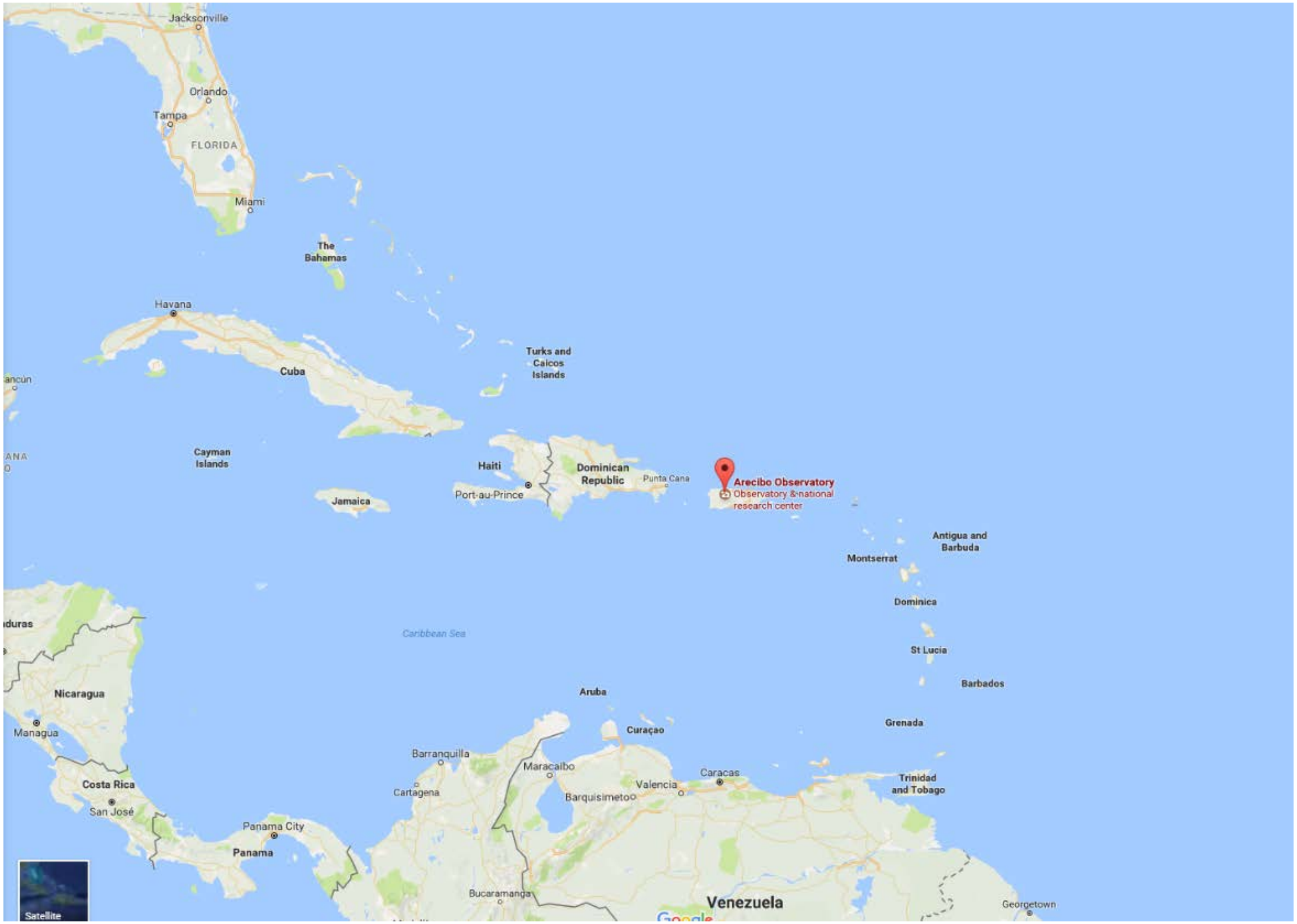
Frequency domain

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

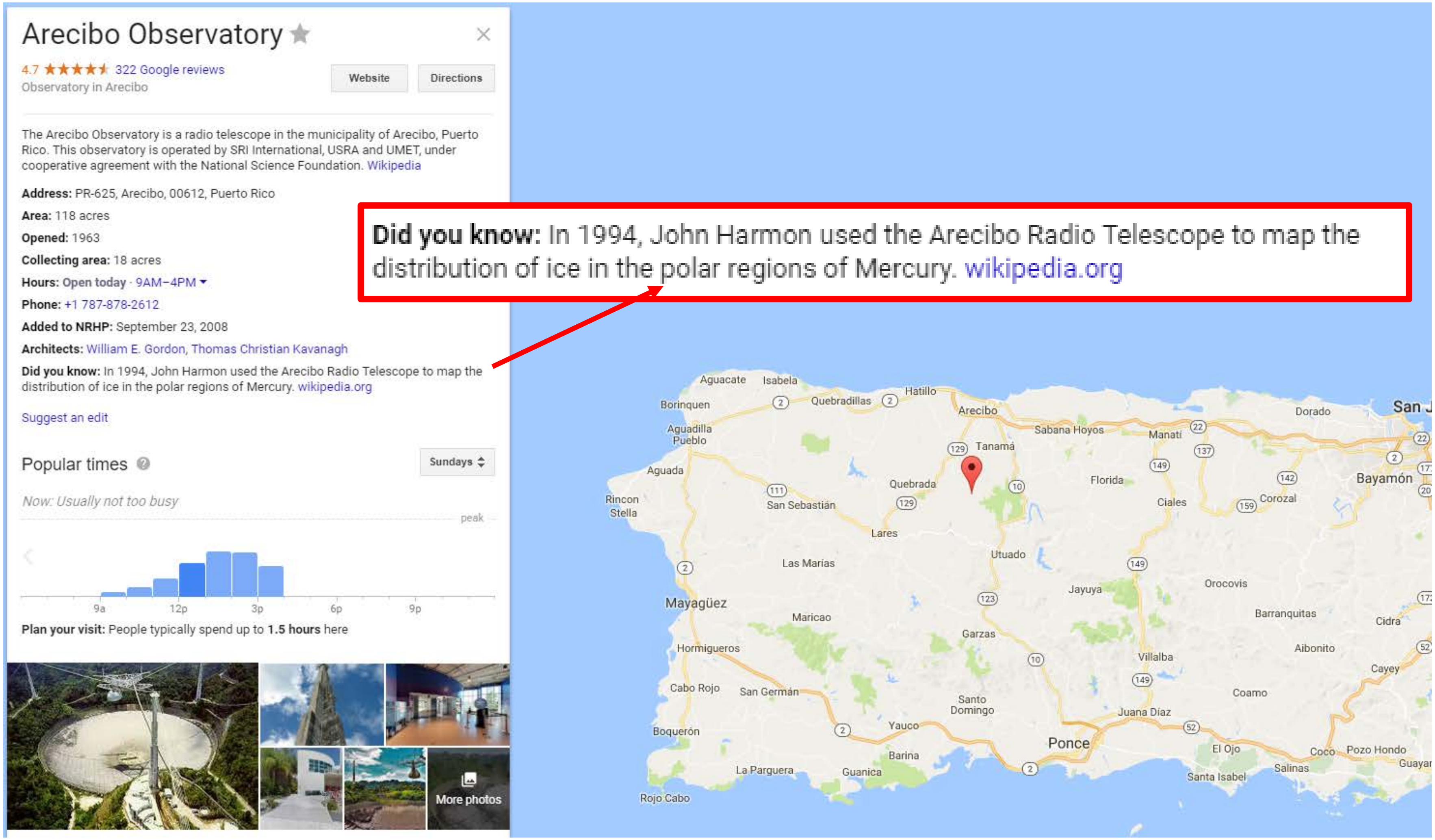
Analysis of Space Signals

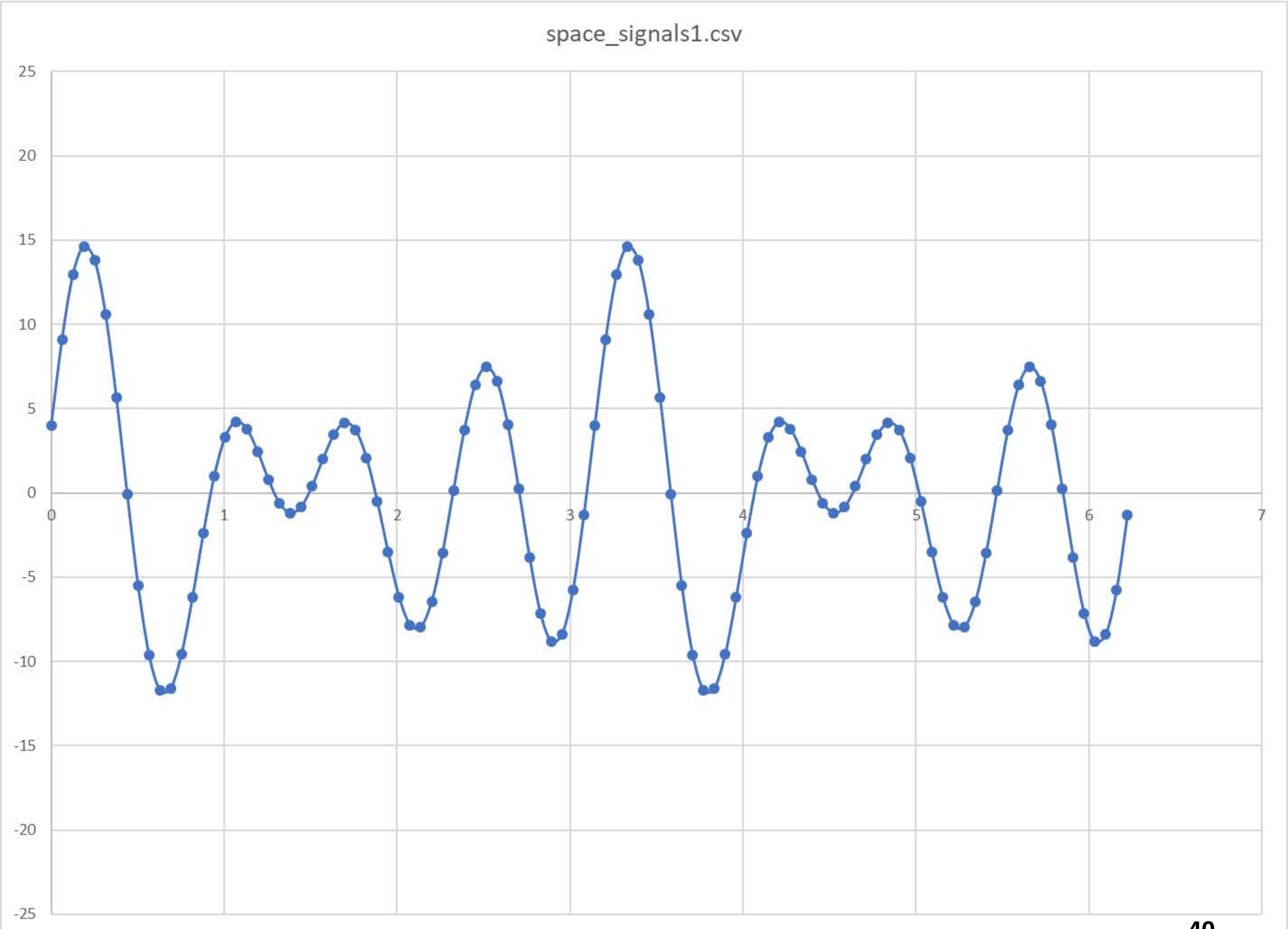
- The **Arecibo Radio Observatory** has detected three candidate signals originating from deep space
- Your task is to perform **Fourier Analysis** on each signal to determine which one **is more likely** to have been a broadcast from an extraterrestrial intelligence

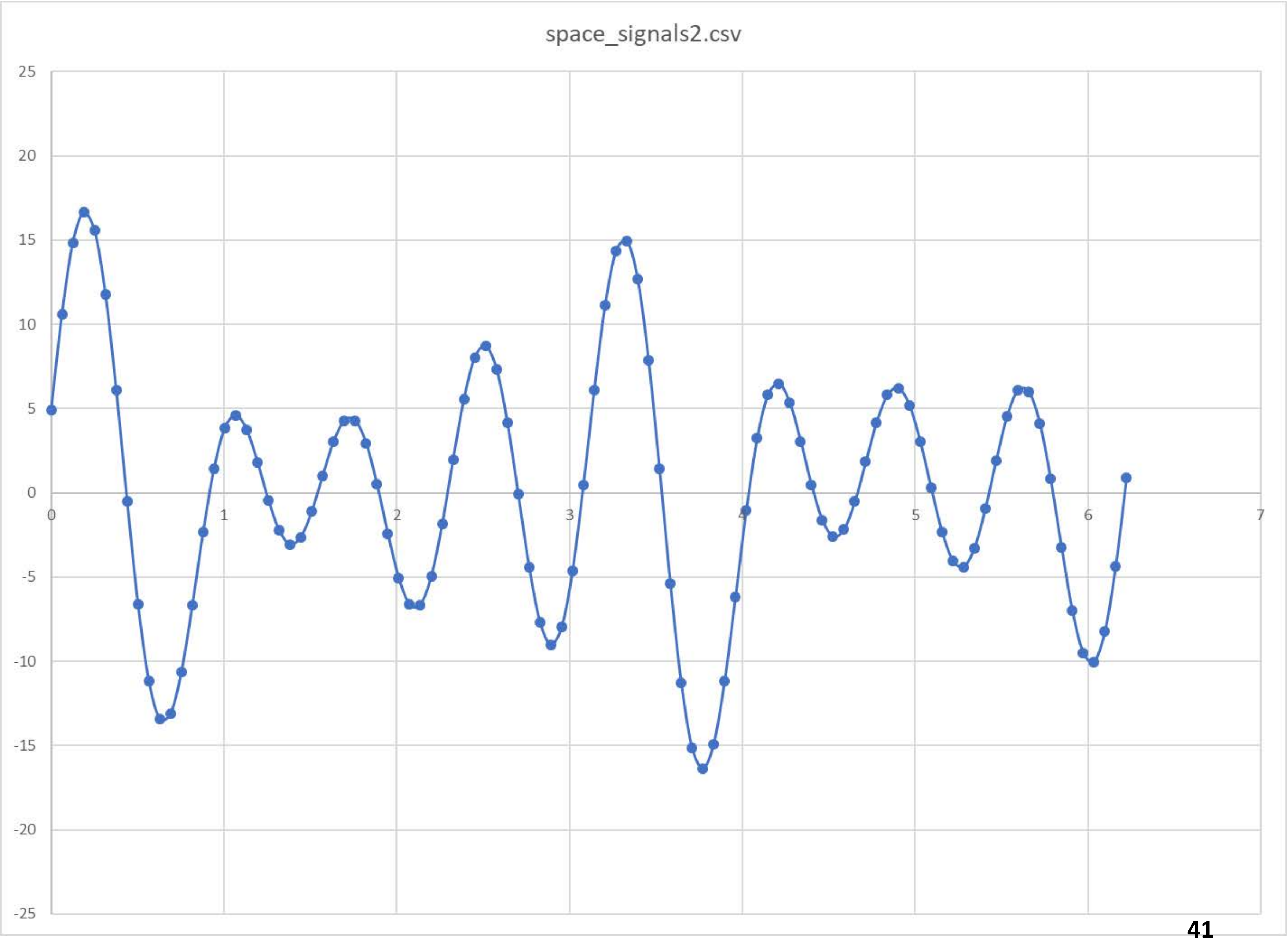
Analysis of Space Signals



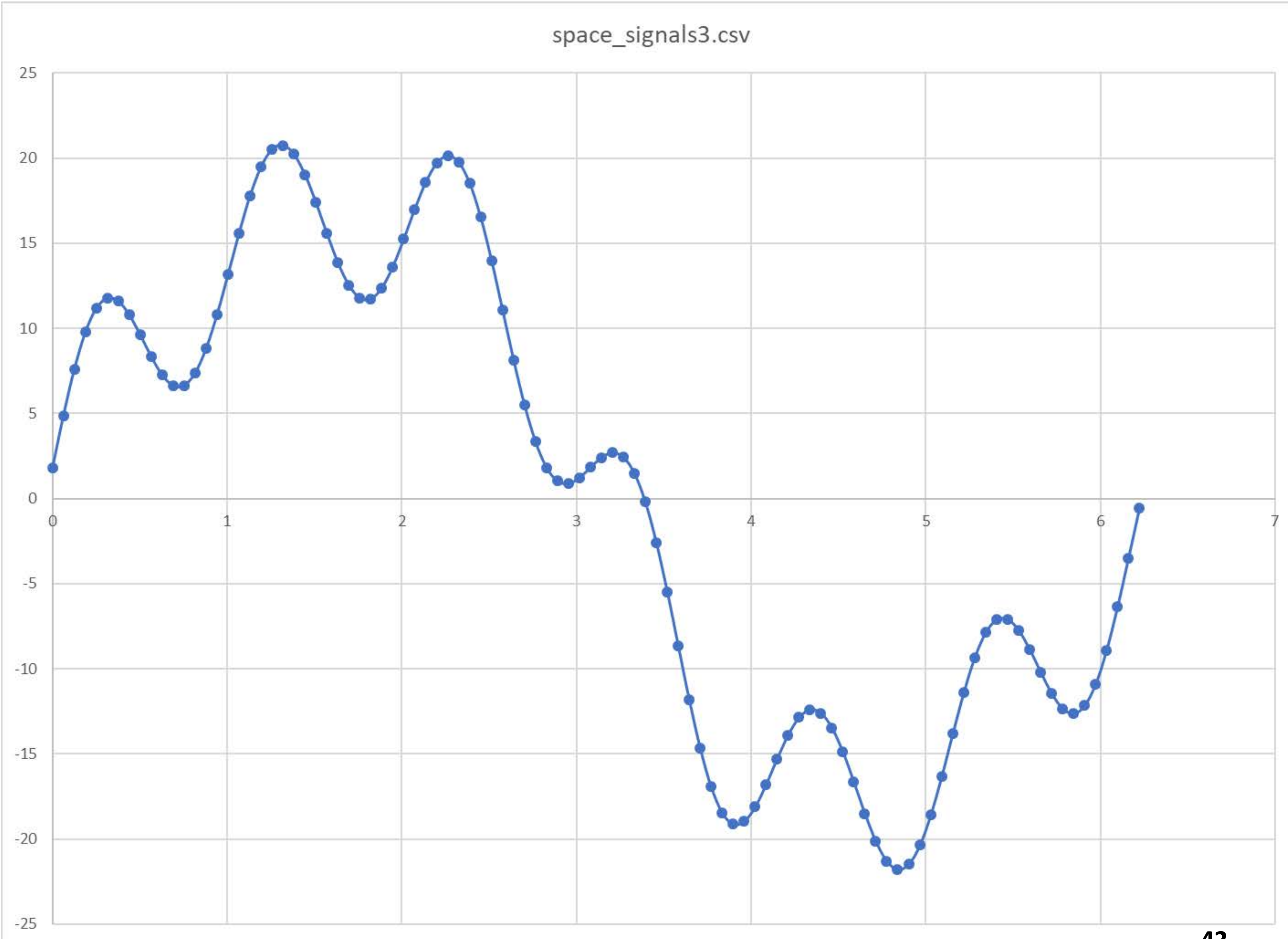
Analysis of Space Signals



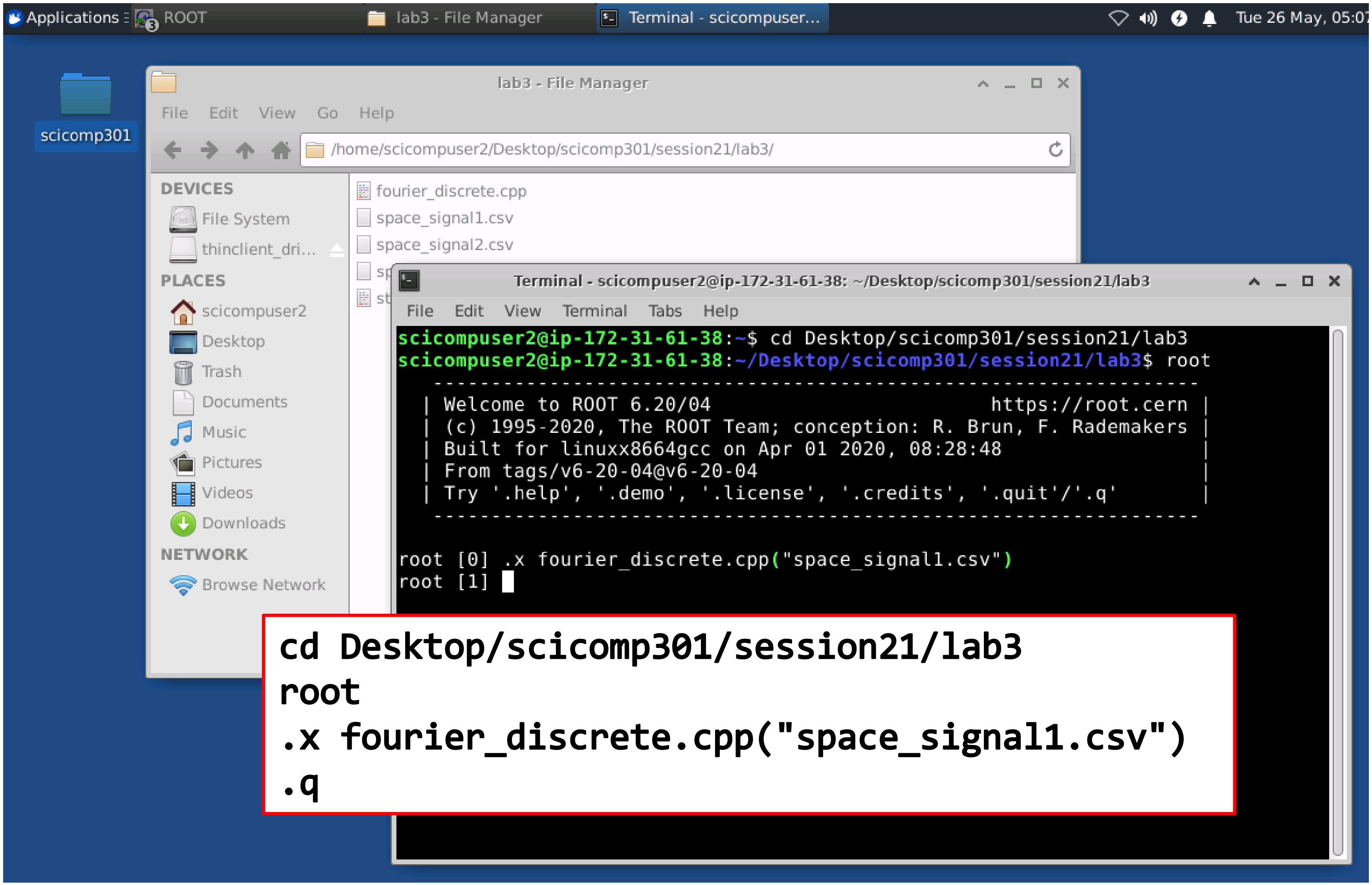

space_signals1.csv
25
20
15
10
5
0
-5
-10
-15
-20
-25
0
1
2
3
4
5
6
7



space_signals2.csv
25
20
15
10
5
0
-5
-10
-15
-20
-25
0
1
2
3
4
5
6
7

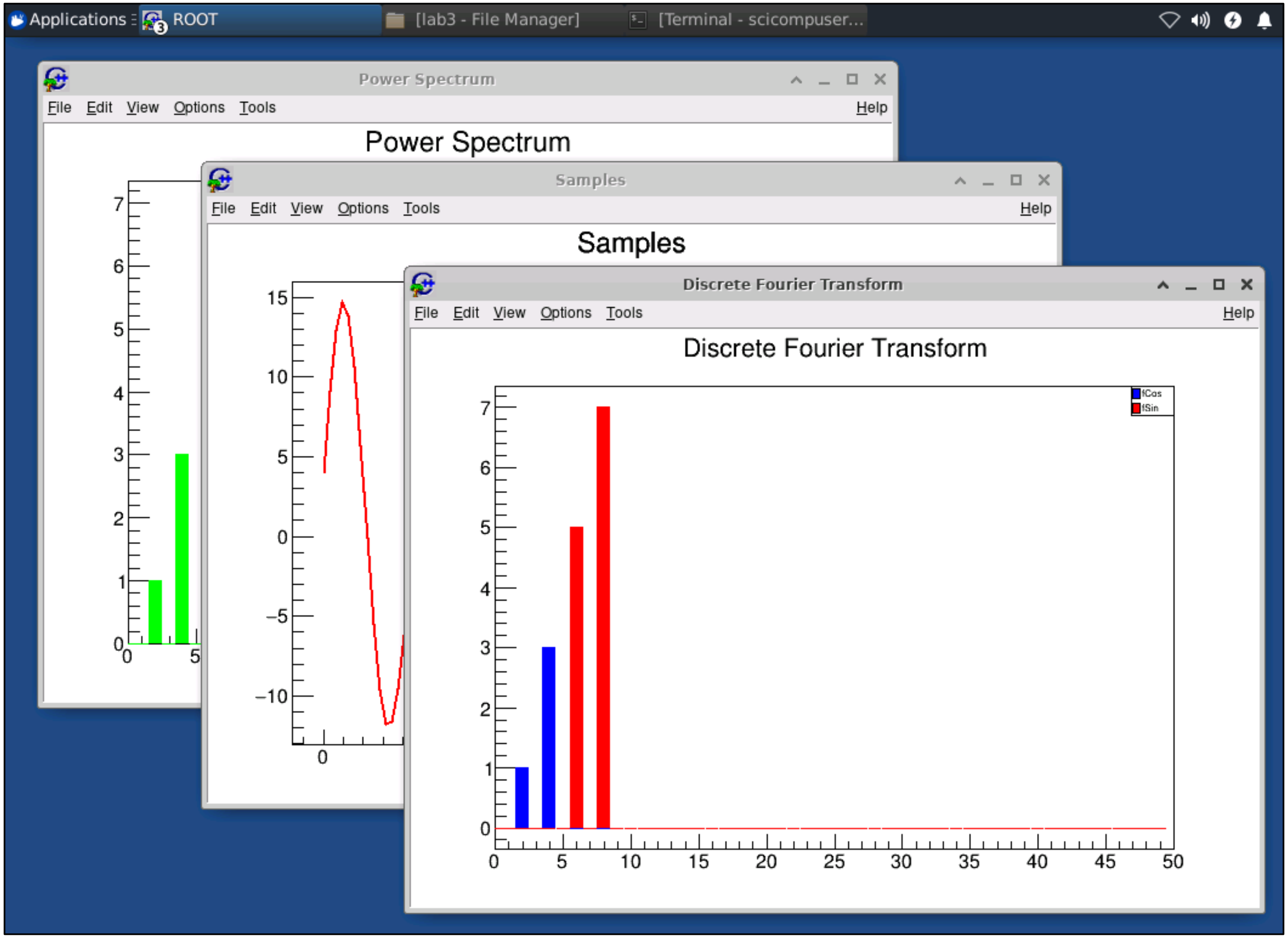

space_signals3.csv
25
20
15
10
5
0
-5
-10
-15
-20
-25
0
1
2
3
4
5
6
7

Run Lab 3 – Space Signals Analysis




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

Check Lab 3 – Space Signals Analysis



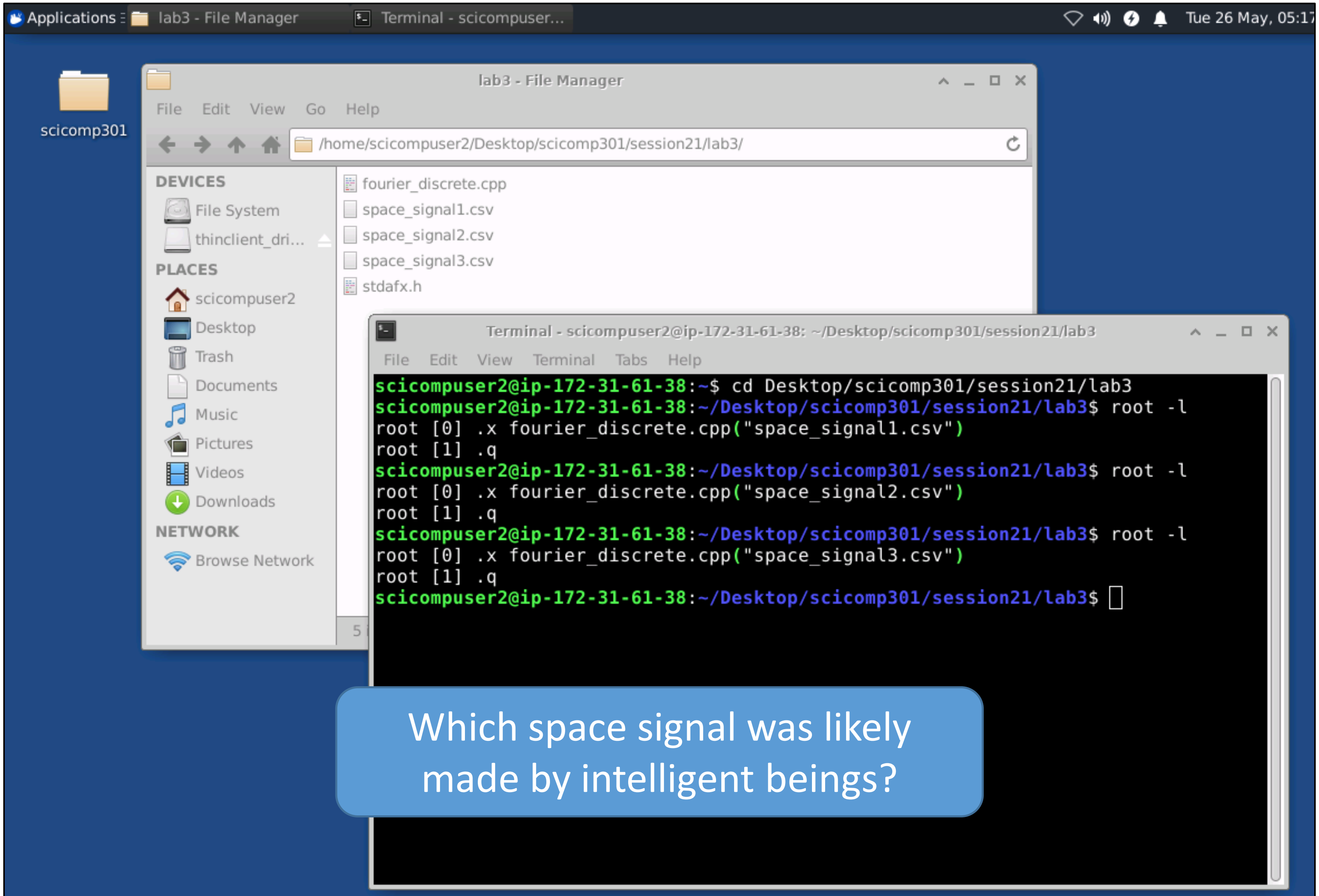
Run Lab 3 – Space Signals Analysis

The lowercase L option suppresses the CERN ROOT Welcome banner

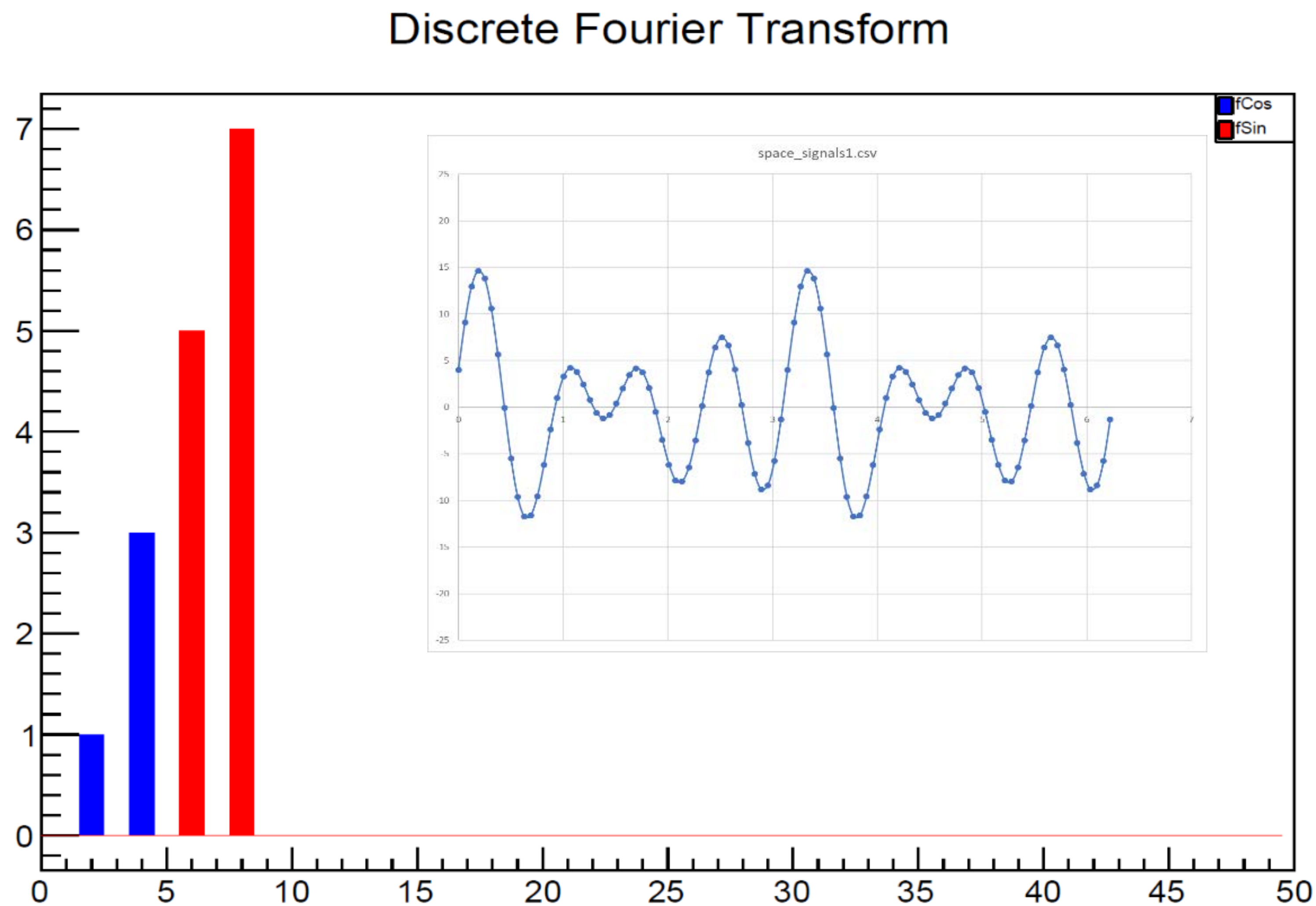
You must run ROOT three times (once for each data file)

```
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
```

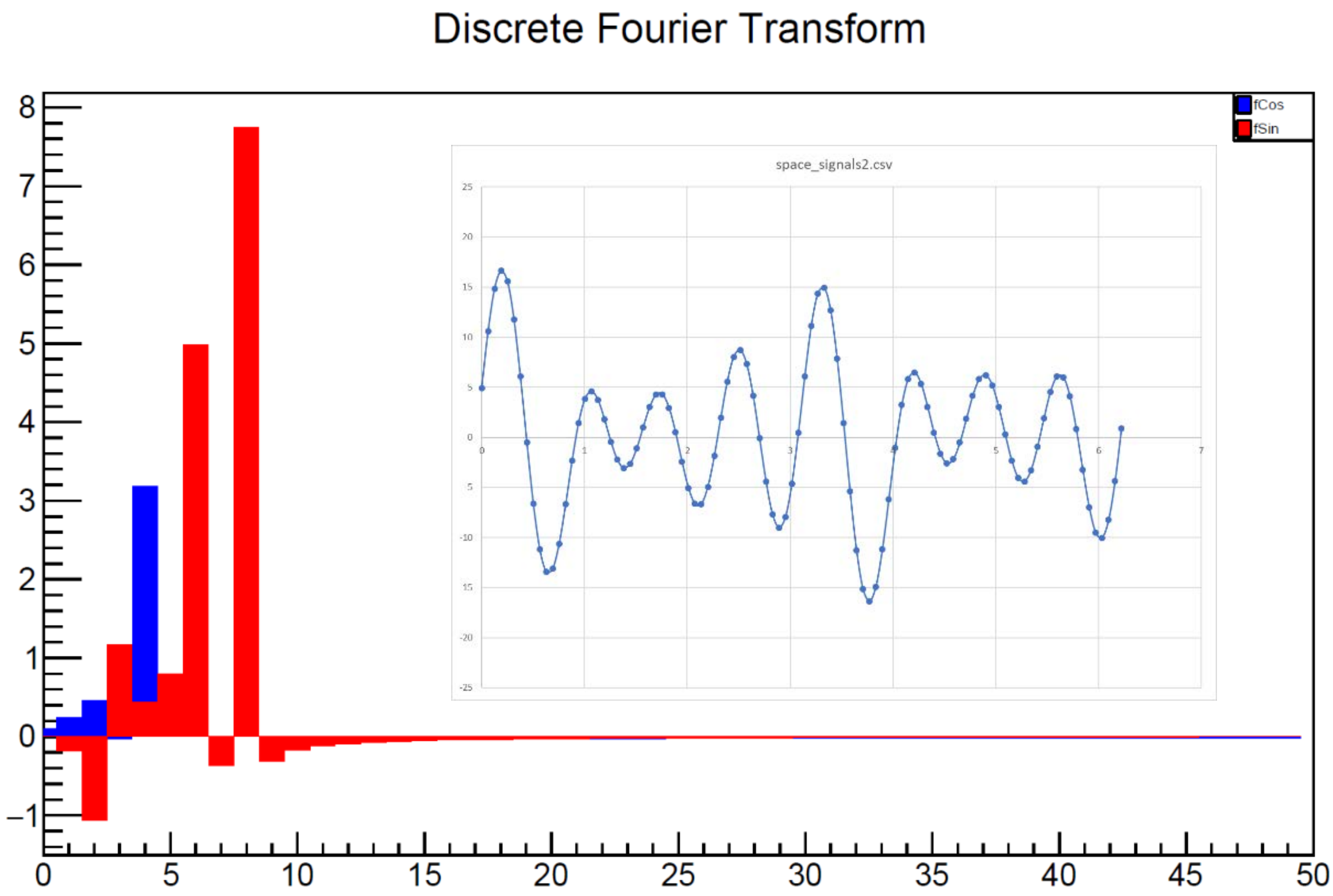
Run Lab 3 – Space Signals Analysis



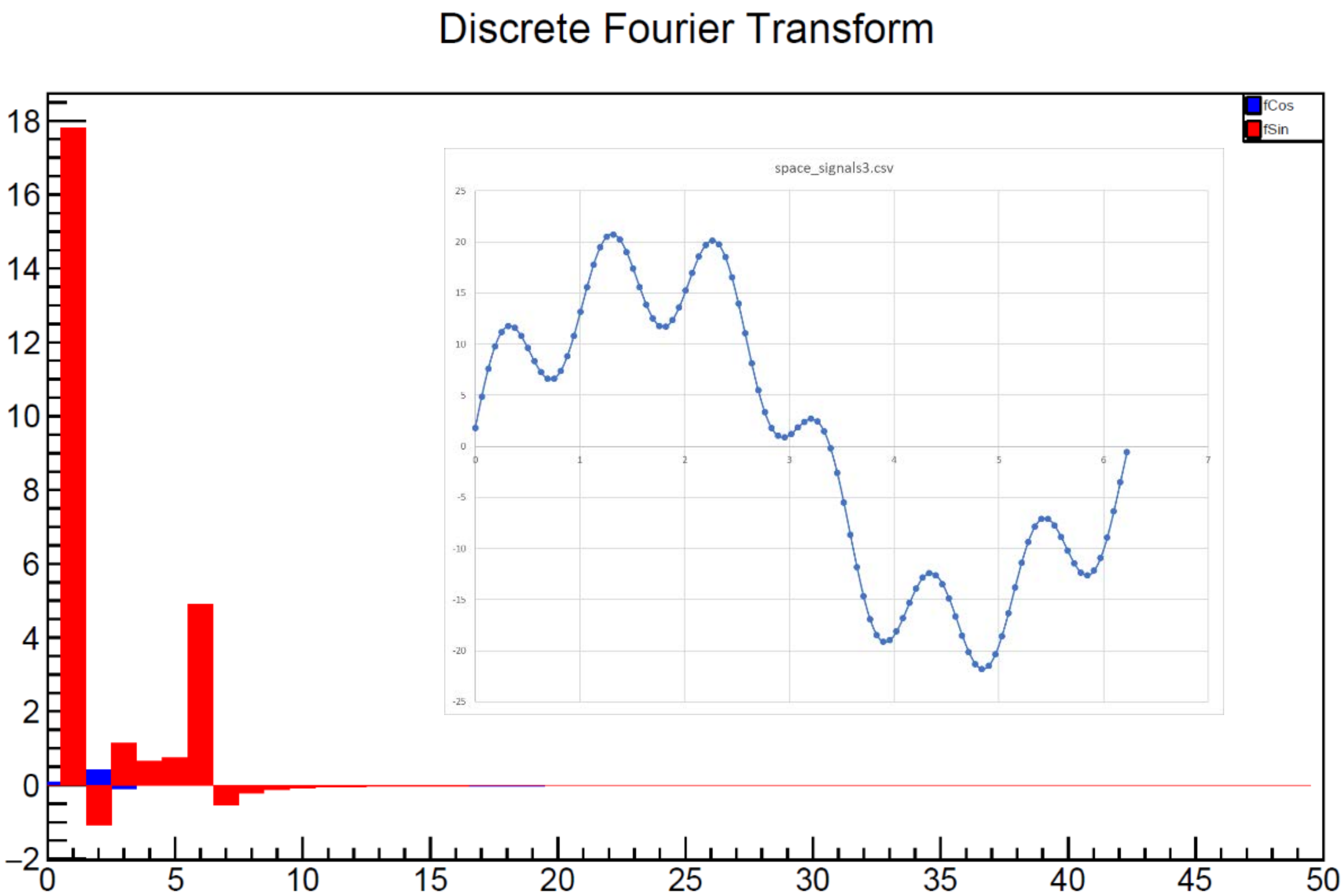
space_signal1



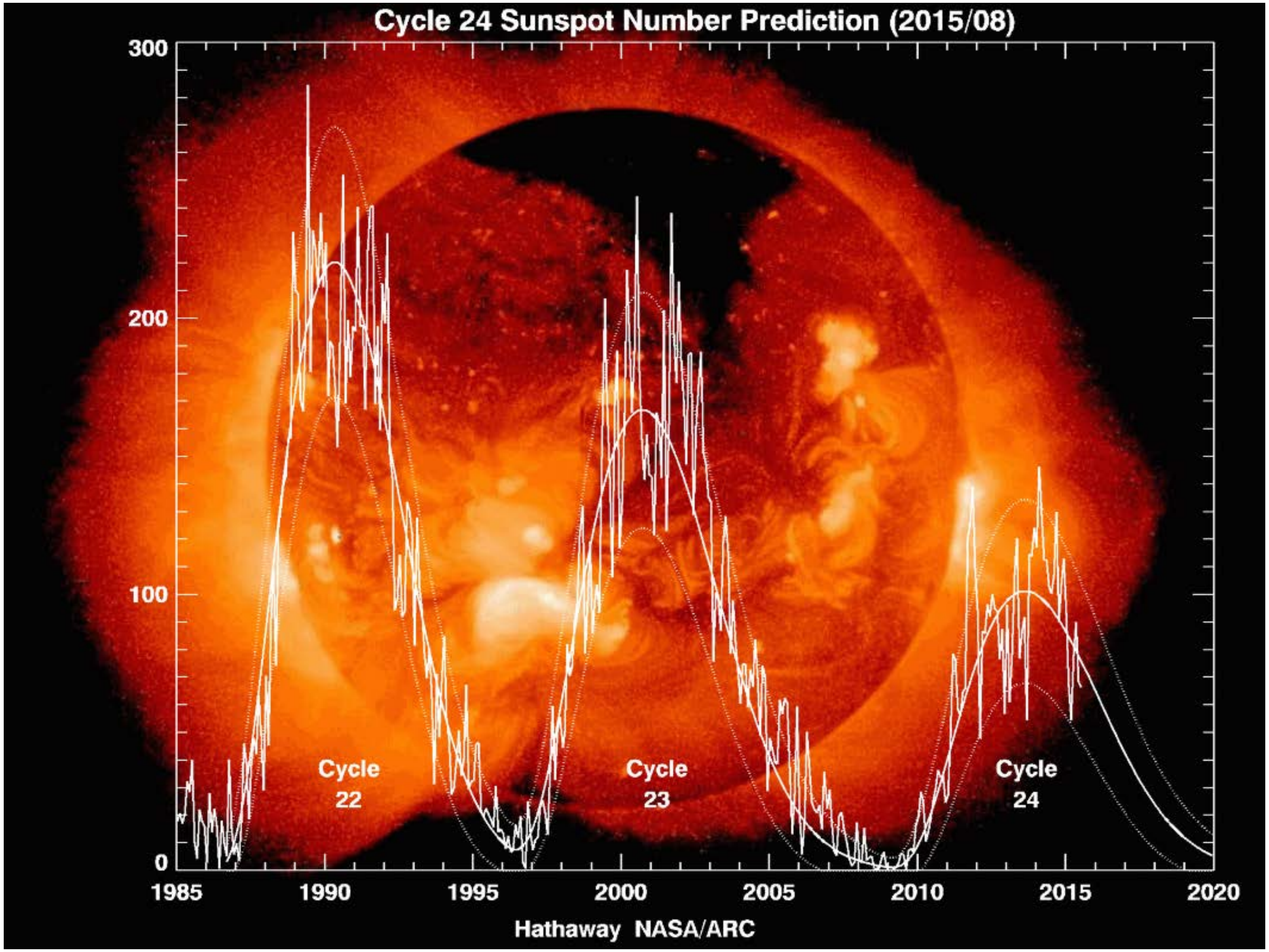
space_signal2



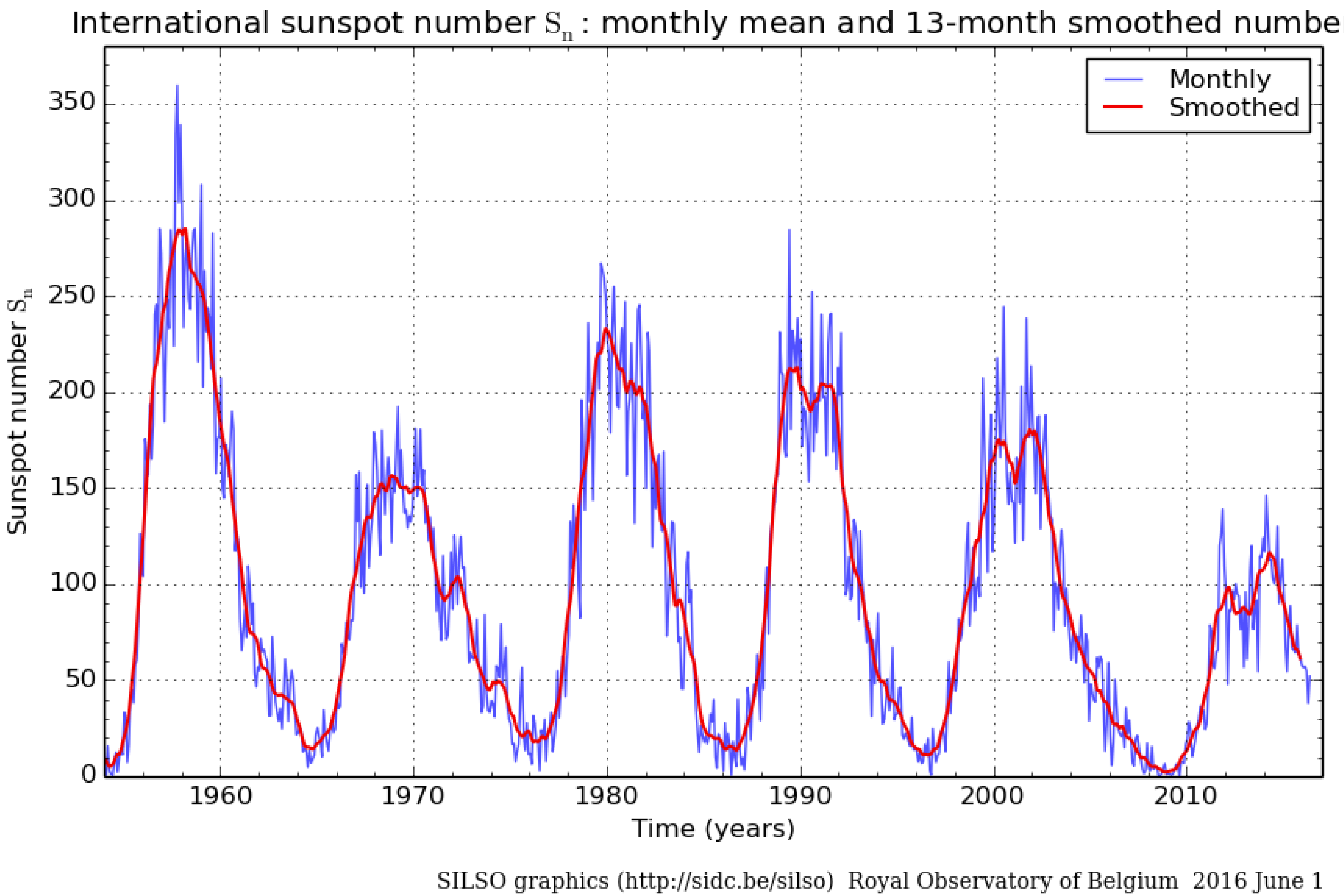
space_signal3



Sunspot Activity



Sunspot Activity



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						
6	1705	96.7						
7	1706	48.3						
8	1707	33.3						
9	1708	16.7						
10	1709	13.3						
11	1710	5						
12	1711	0						
13	1712	0						
14	1713	3.3						
15	1714	18.3						
16	1715	45						
17	1716	78.3						
18	1717	105						

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.

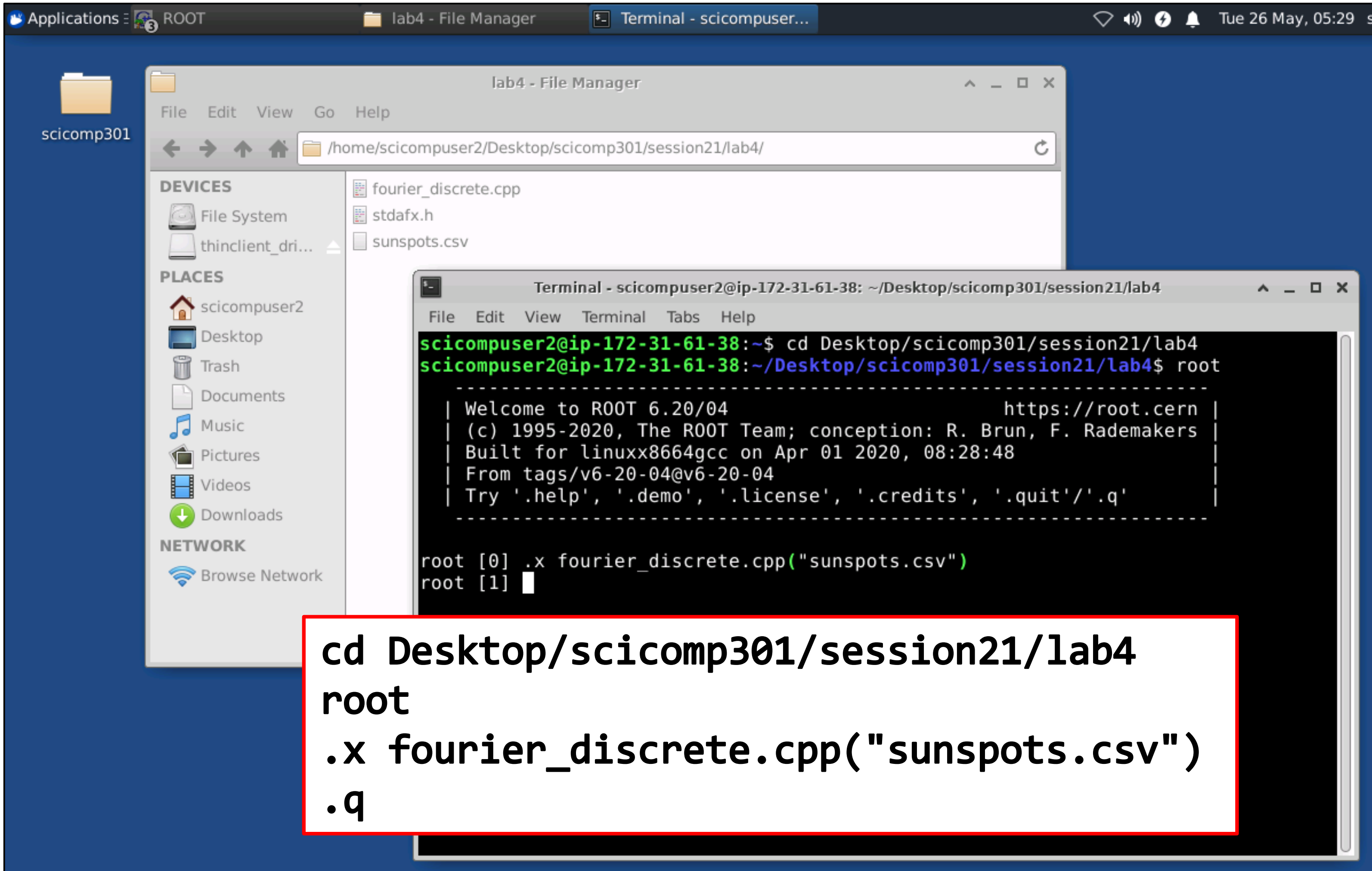
History [edit]

The idea of computing sunspot numbers was originated by Rudolf Wolf in 1848[1] 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.[2]

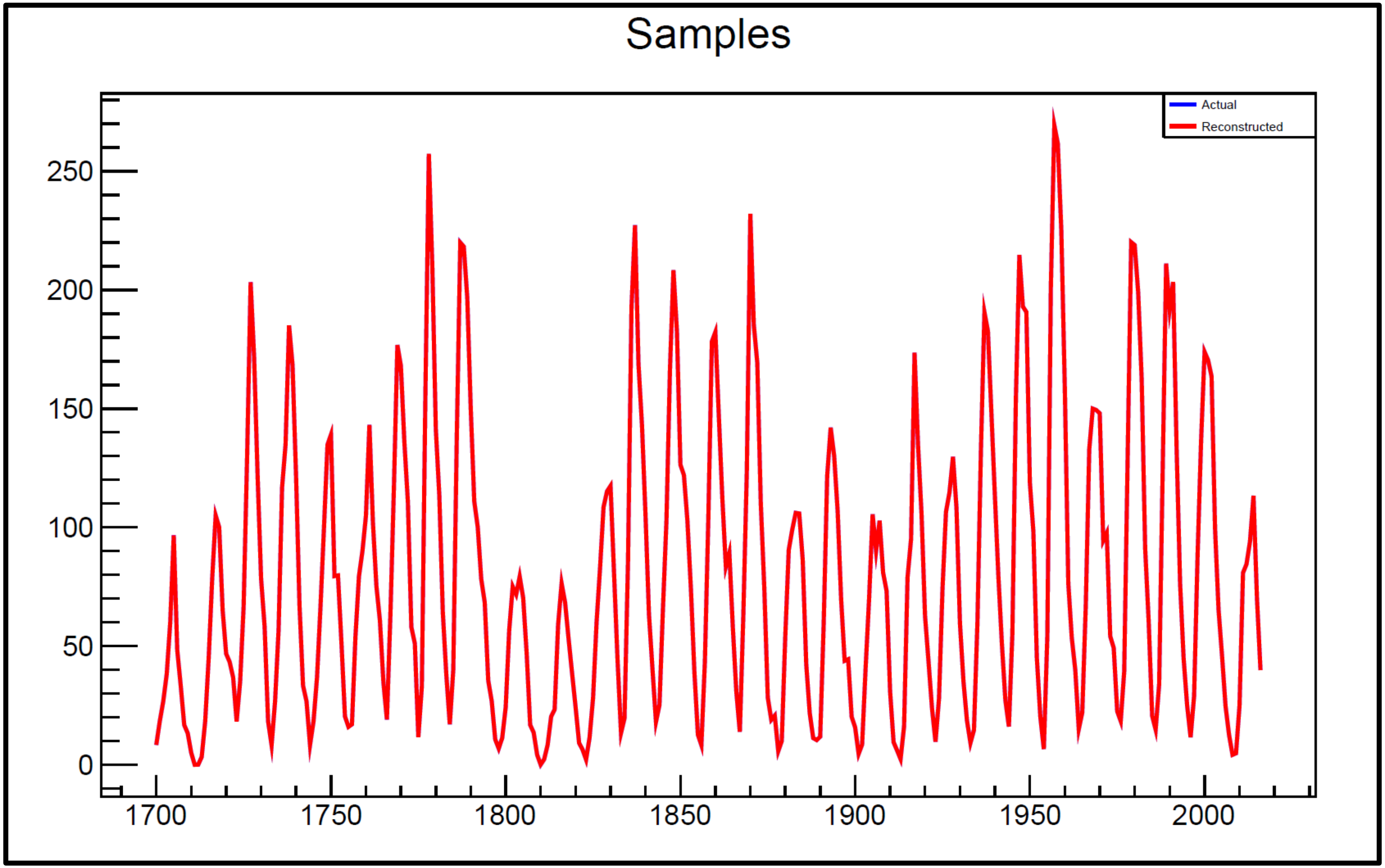
A human has counted sunspots *every day* from **1700 – 2016**!

Run Lab 4 – Sunspot Activity

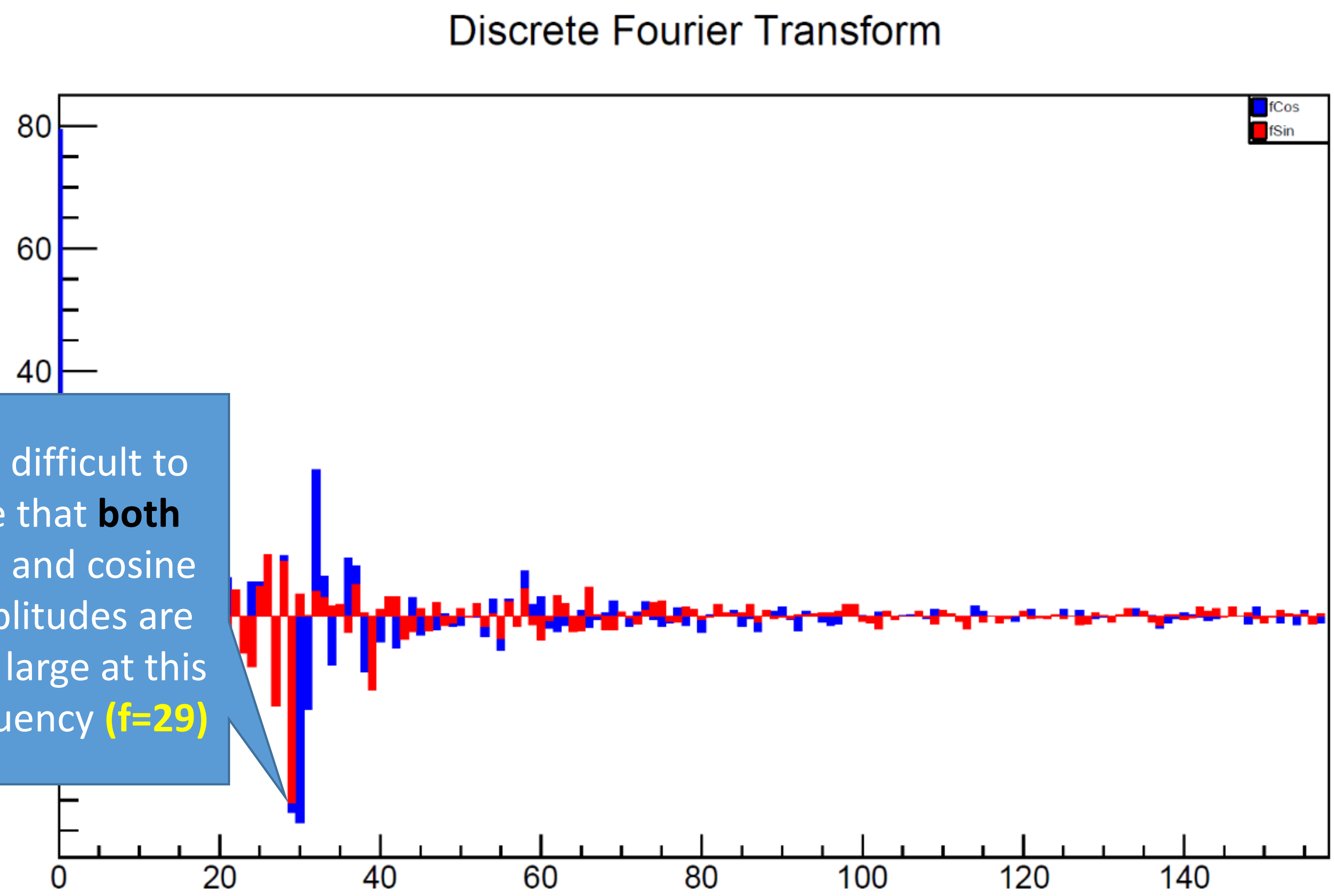



```
cd Desktop/scicomp301/session21/lab4
root
.x fourier_discrete.cpp("sunspots.csv")
.q
```

Check Lab 4 – Sunspot Activity



Check Lab 4 – Sunspot Activity

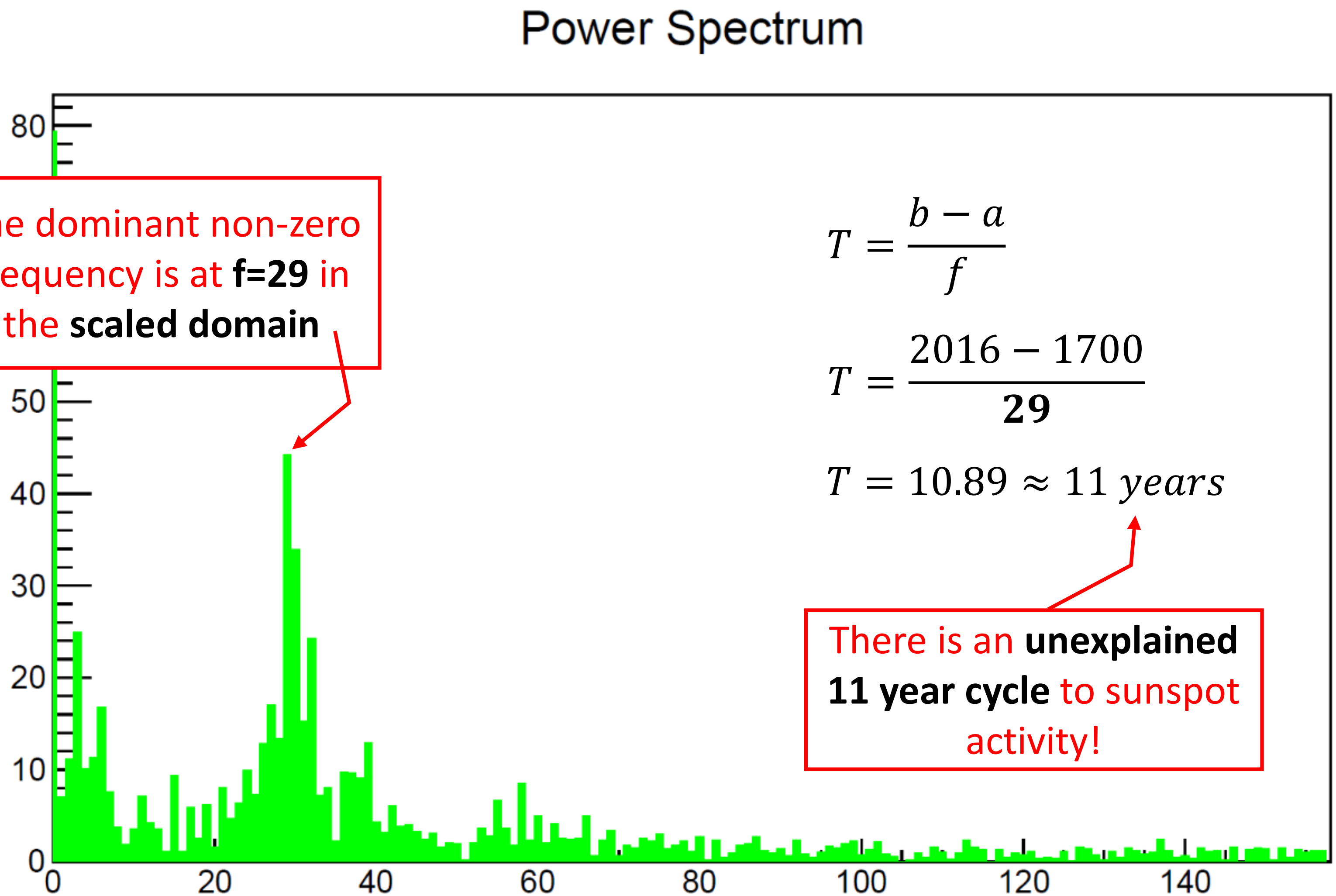


It is difficult to see that **both** sine and cosine amplitudes are very large at this frequency **(f=29)**

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



The dominant non-zero frequency is at **f=29** in the **scaled domain**

$$T = \frac{b-a}{f}$$

$$T = \frac{2016-1700}{\mathbf{29}}$$

$$T = 10.89 \approx 11\ years$$

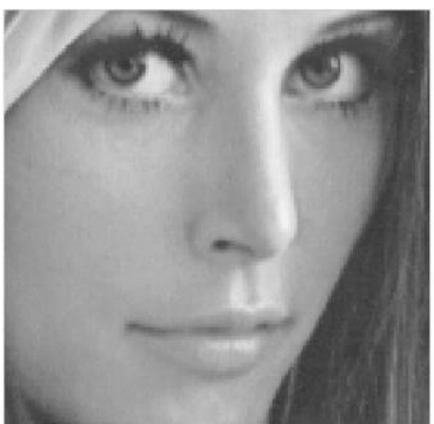
There is an **unexplained 11 year cycle** to sunspot activity!

Every JPEG Image uses Fourier Transforms

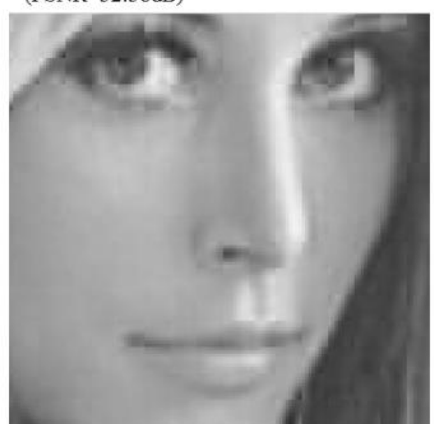
(a) Original Lena image

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

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



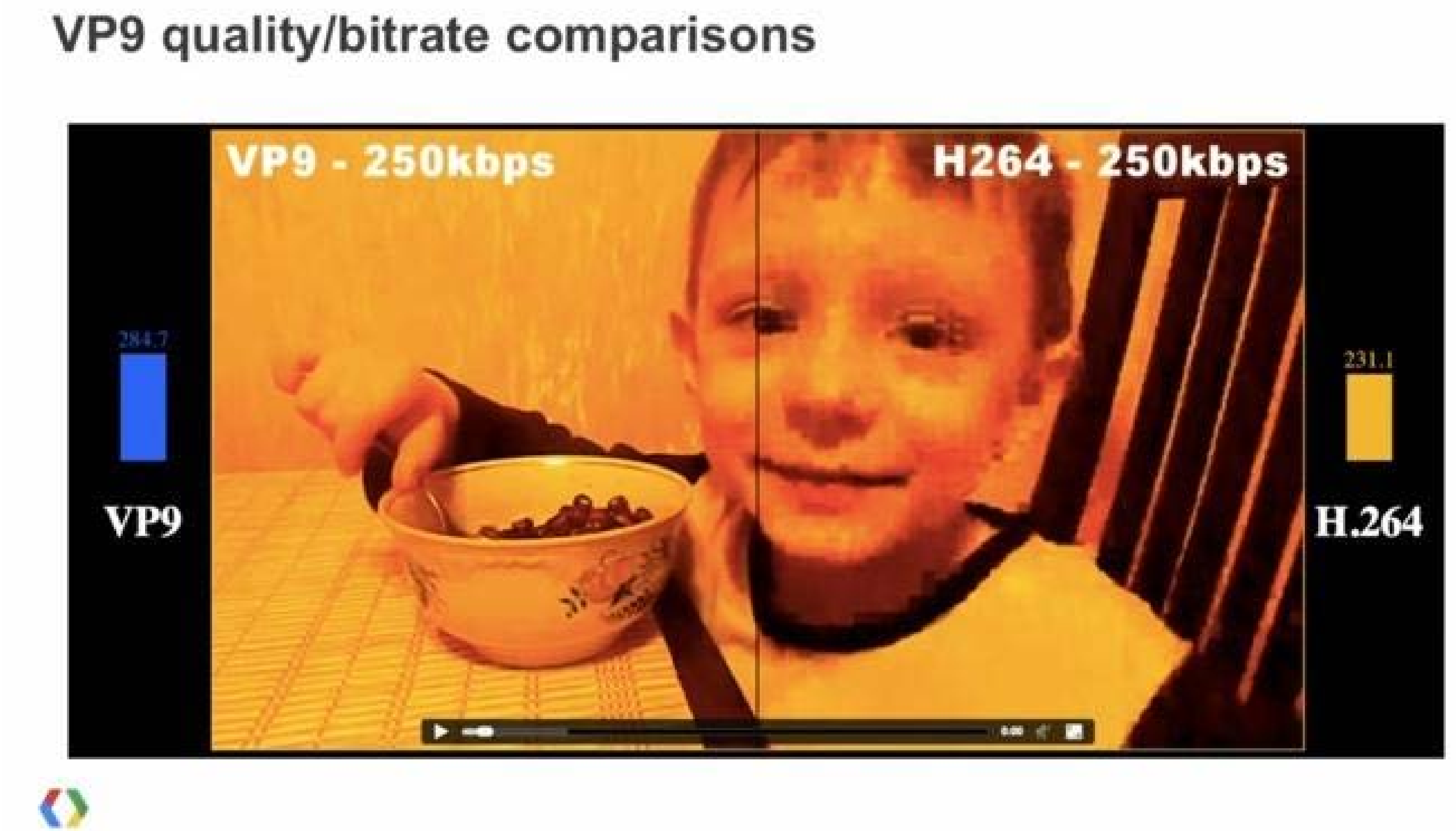
(b) Zoomed original Lena image



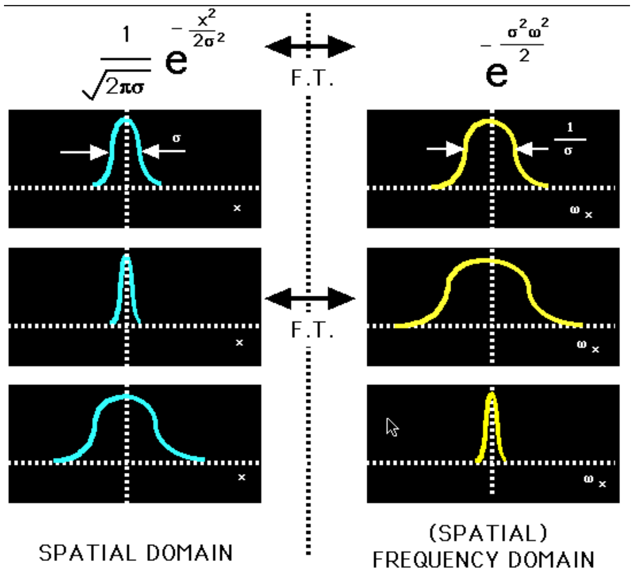
(d) Zoomed DCT-II compressed Lena image

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

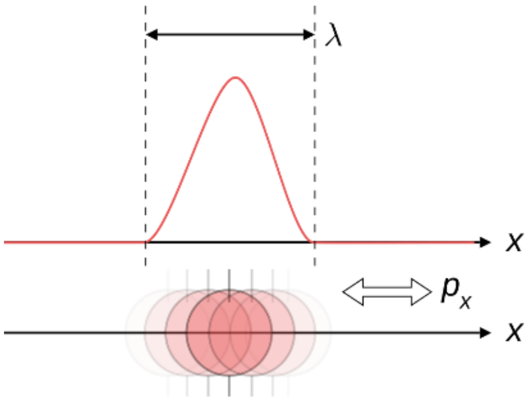
All Streaming Video uses Fourier Transforms

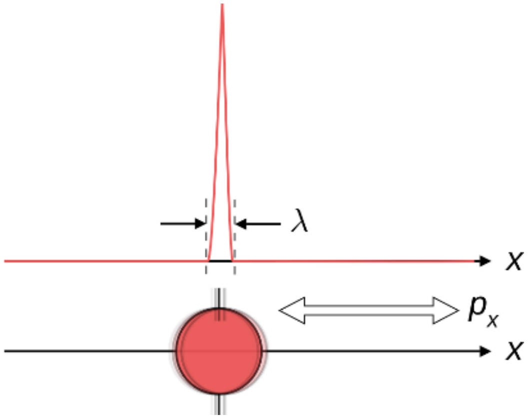


Heisenberg Uncertainty Principle



$$\boxed{\Delta x \cdot \Delta p \geq h/2}$$





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