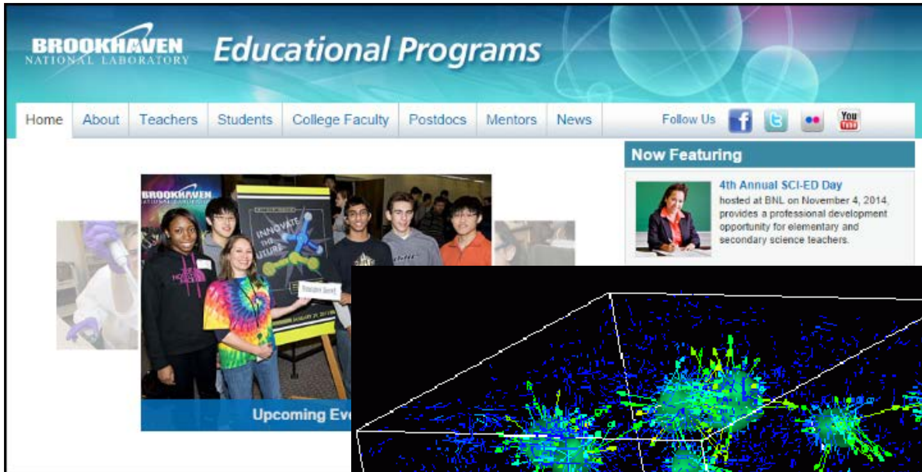
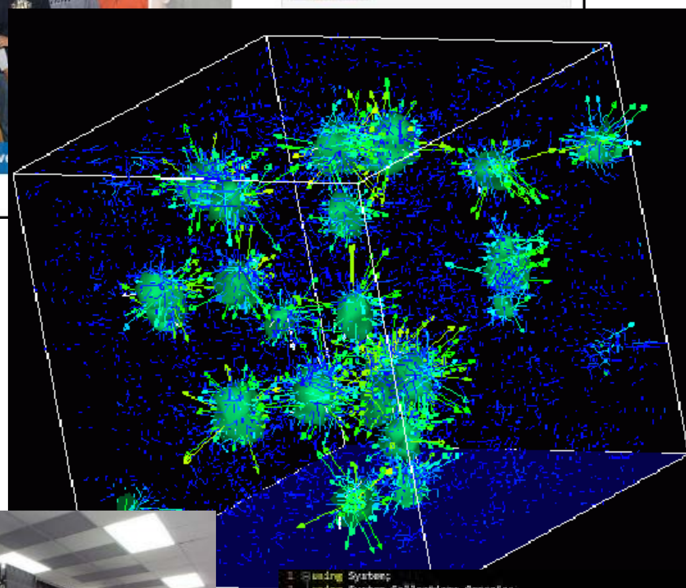
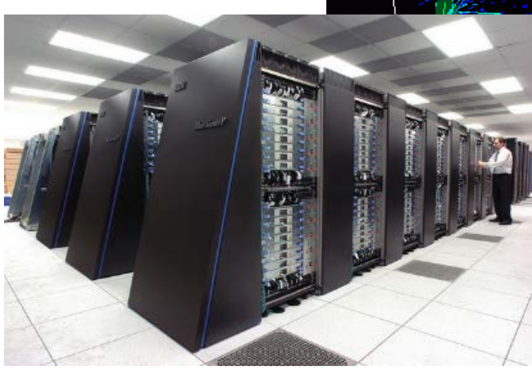






Survey of Scientific Computing

(SciComp 301)

Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

Session 21

Fourier Transform, Signals Analysis

Session Goals

- Learn about **Fourier Analysis**
- Compute the Discrete Fourier Transform (**DFT**)
- Compute the Inverse Discrete Fourier Transform (**IDFT**)
- Perform signals analysis on transmissions from outer space
- Calculate frequency of **unexplained** cycles in sunspot activity

<u>Know</u> 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

<u>Know</u> 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

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

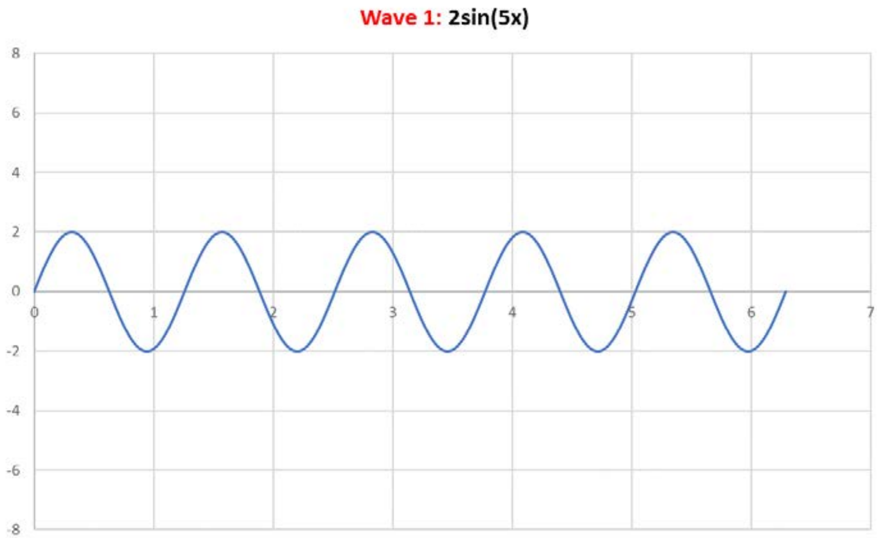
All of Physics *is* Waves

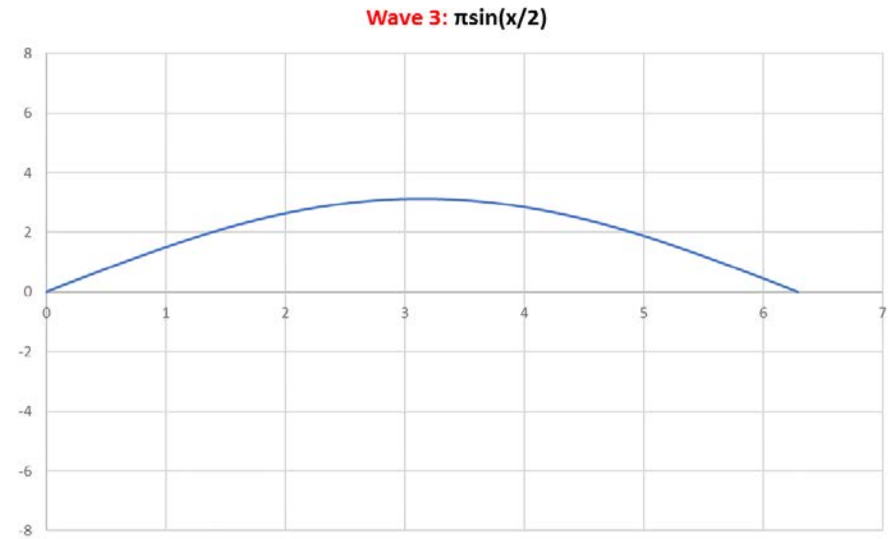
- Electrical
- Magnetic
- Acoustic
- Heat Flow
- Vibrational
- Torsional
- Nuclear / Quantum
- Gravitational
- Oceanic / Tidal
- Orbital Precession
- Springs
- Pendulums
- Image Reconstruction
- Stock Market
- Economics
- Astronomical
- Fluid Dynamics
- Earthquakes
- AC / DC
- AM / FM
- Speech
- Heartbeats

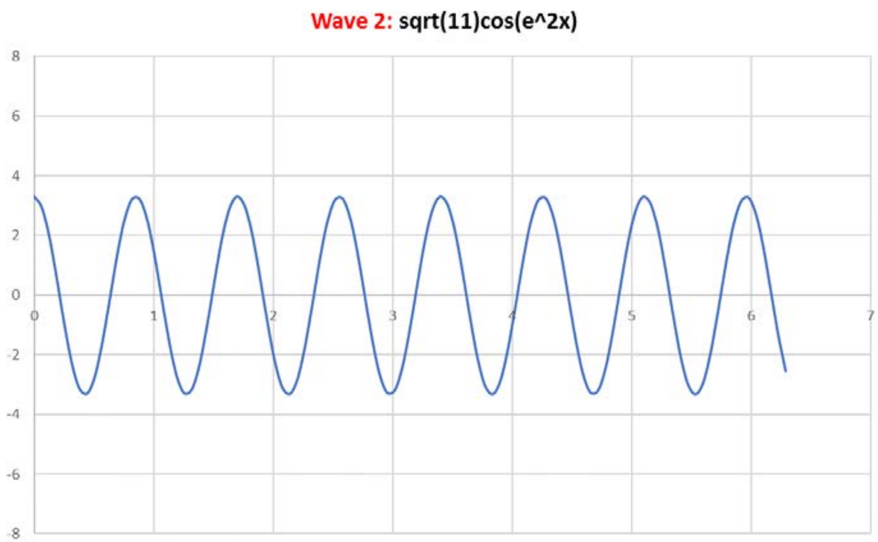
Superposition ⇒ Perceived Complexity

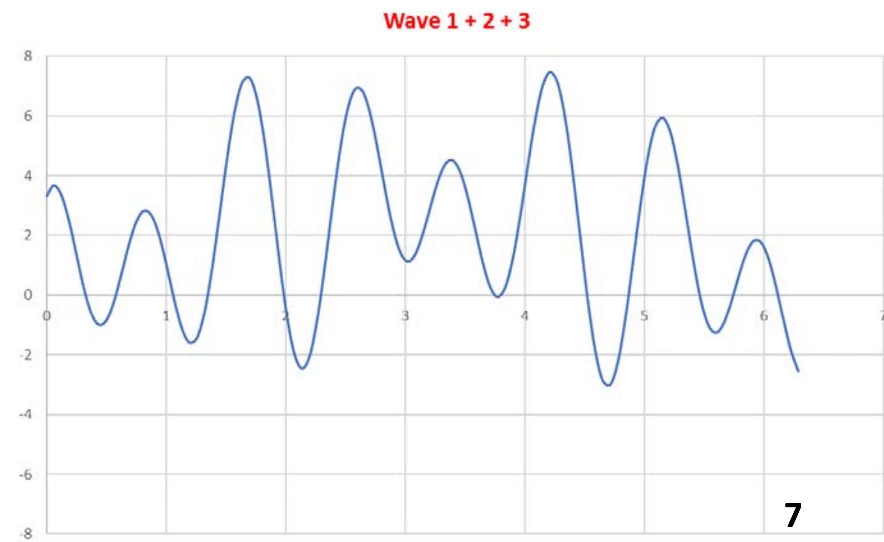
- What appears to be a **complex** waveform might actually be just a series of **simple** waves all added together (linear superposition)
- Underneath the perceived **random** behavior of a function undergoing wild fluctuations might be a system of straightforward waves
- These simple waves may individually 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





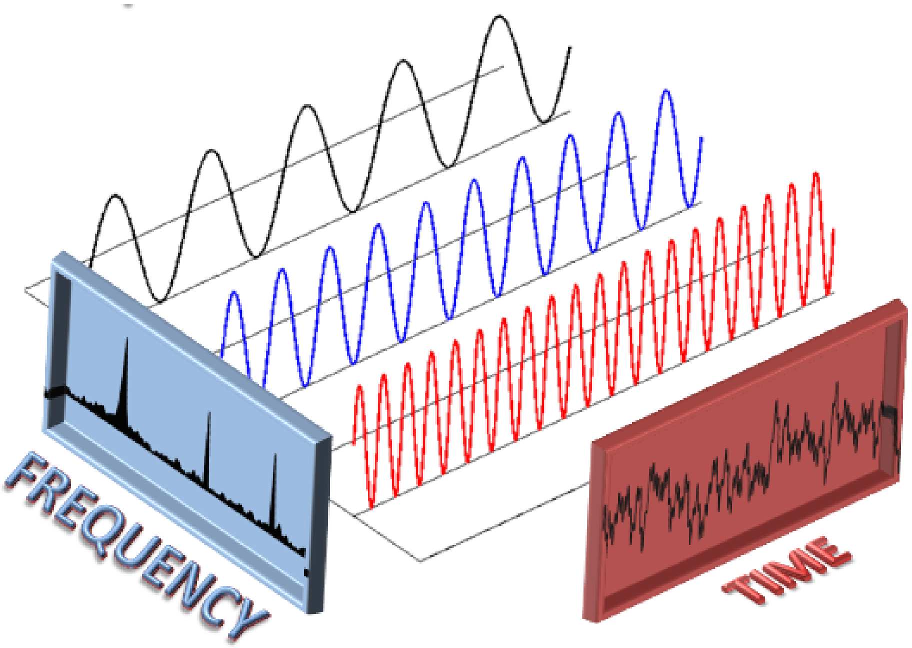




Jean-Baptiste Fourier
(1768-1830)

both r's are silent
"Foo-yeah"

Fourier Transform
Continuous and Discrete

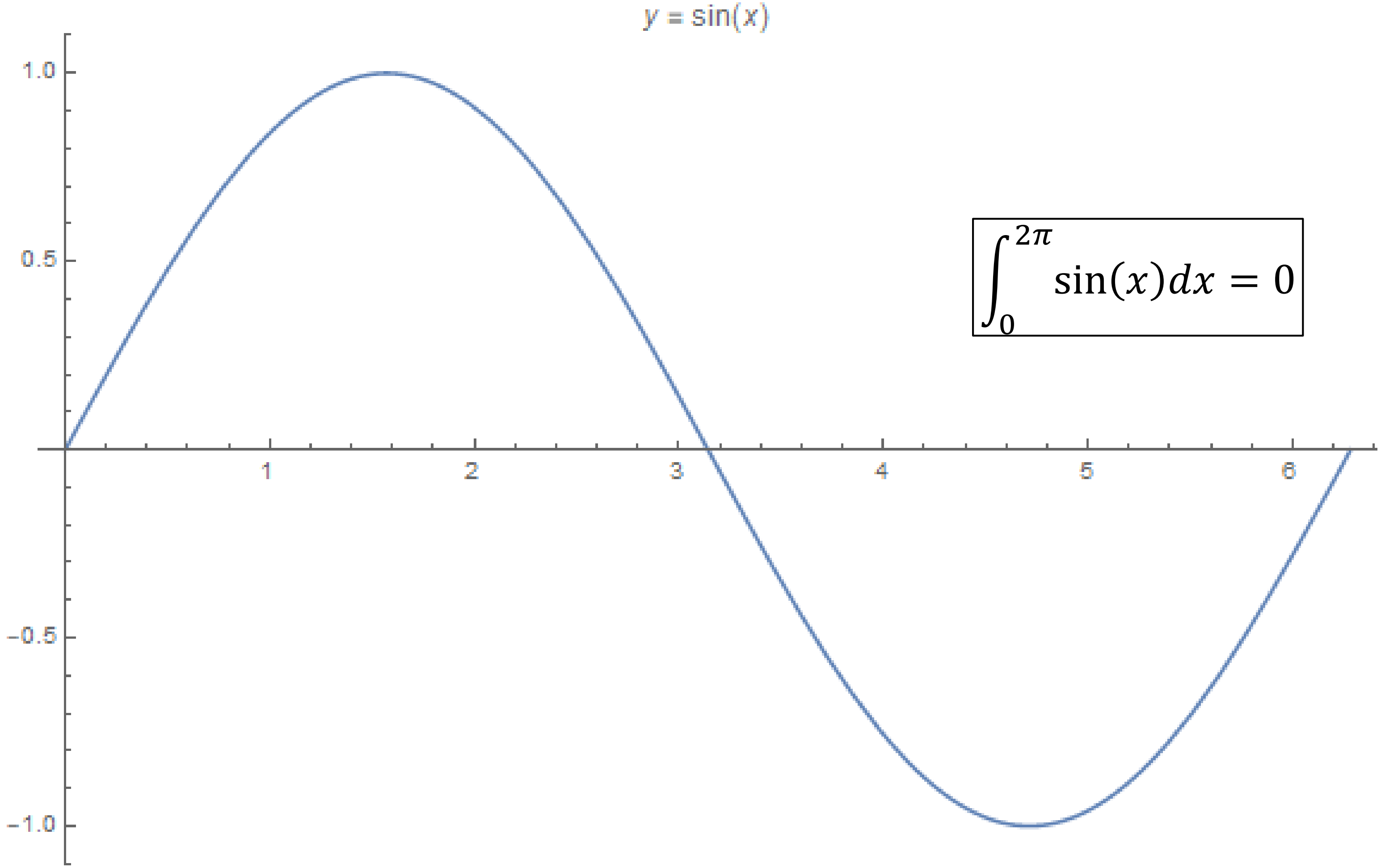


Fourier Analysis

- 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 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

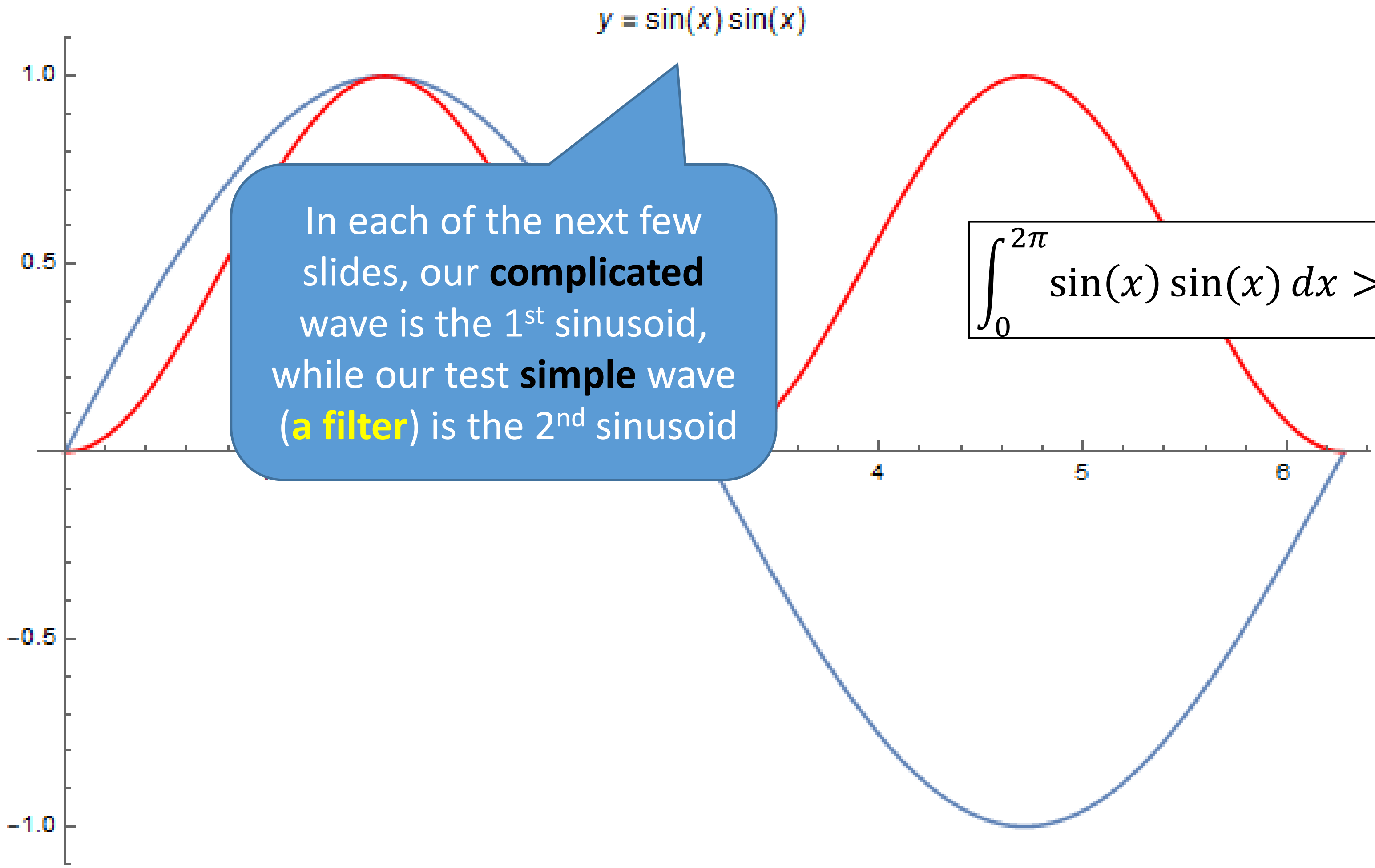
Fourier Analysis - Intuition

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



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

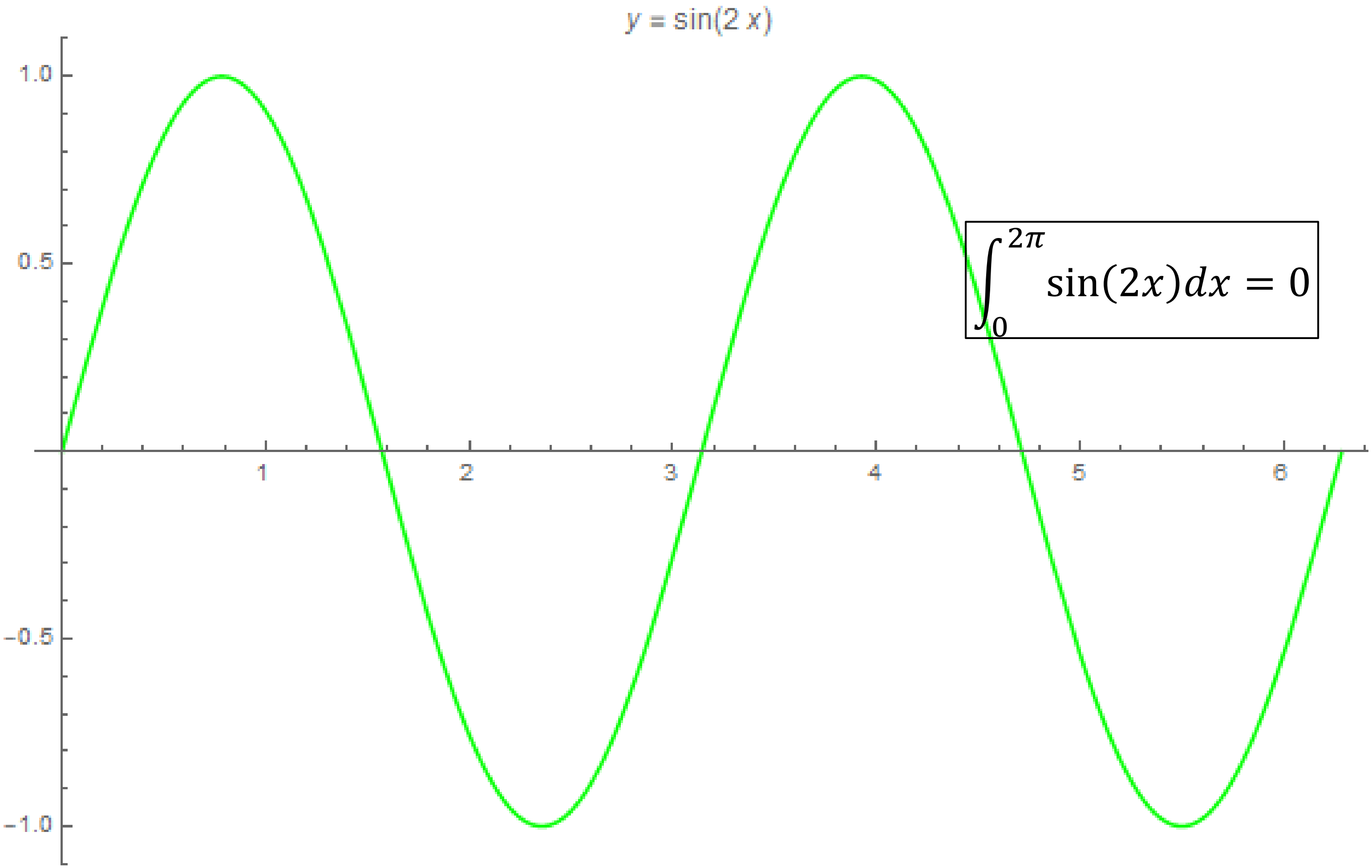
Fourier Analysis - Intuition



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

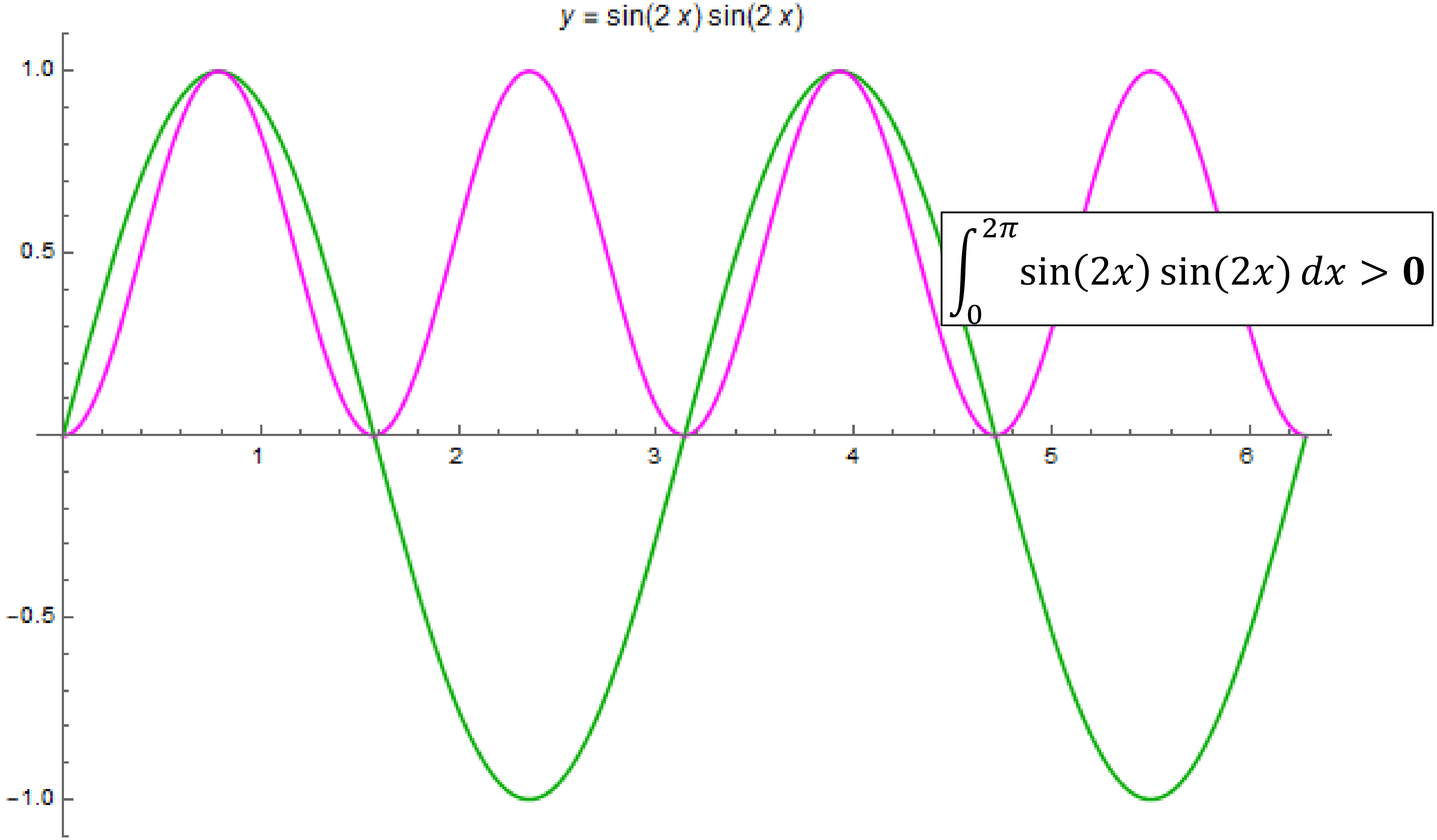
$$\int_0^{2\pi} \sin(x)\sin(x)\,dx > \mathbf{0}$$

Fourier Analysis - Intuition

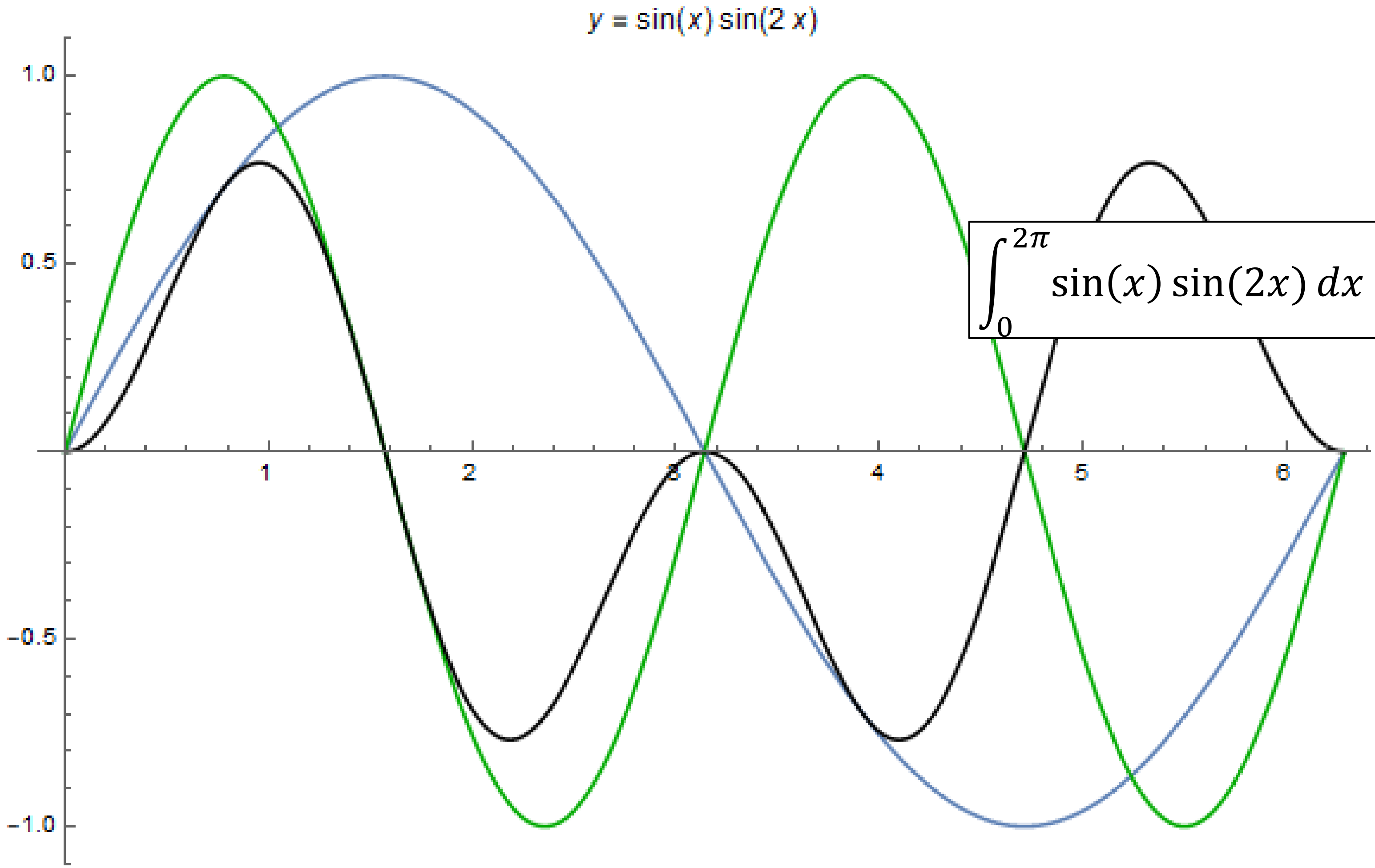


$$\boxed{\int_0^{2\pi} \sin(2x)dx = 0}$$

Fourier Analysis - Intuition

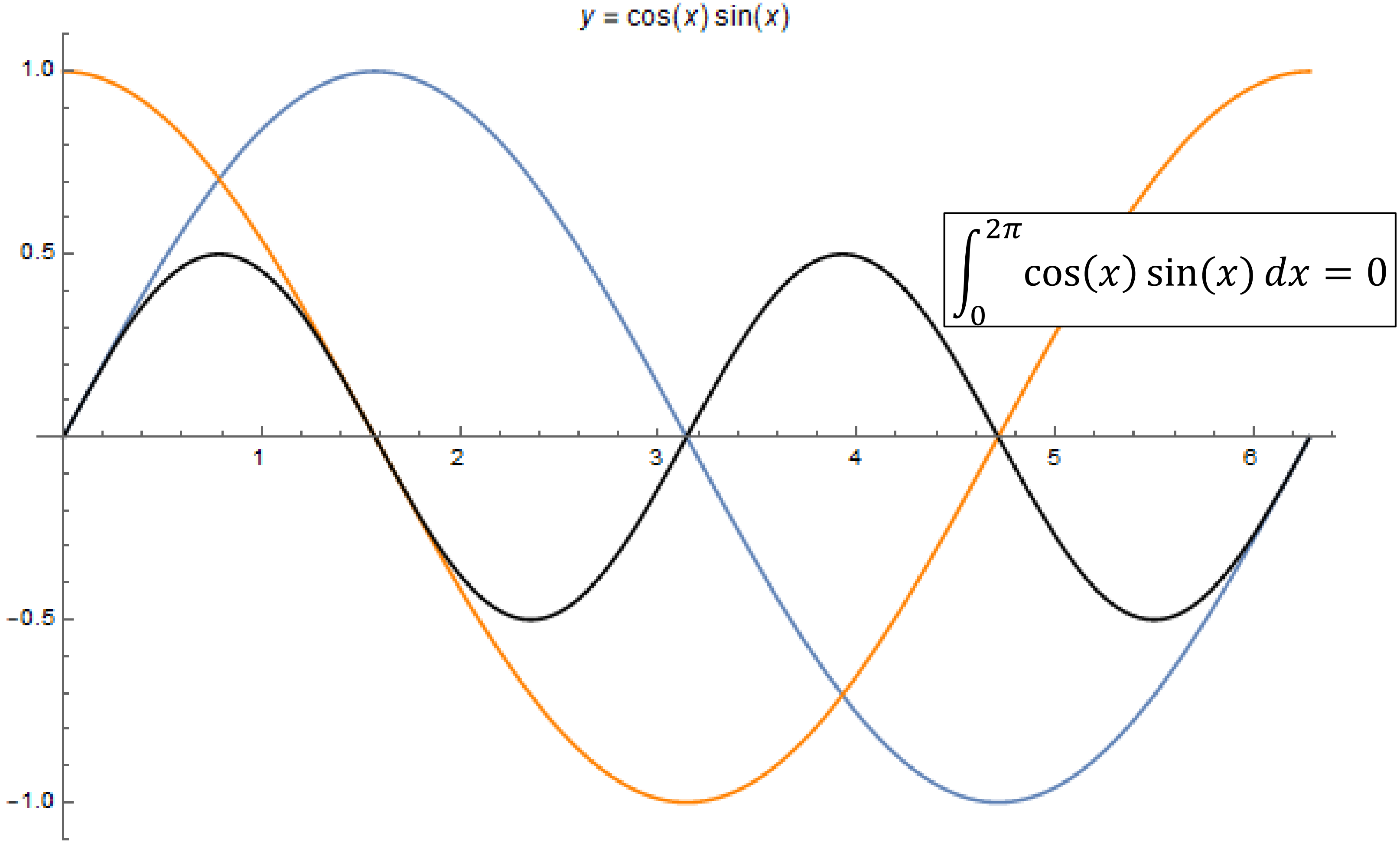


Fourier Analysis - Intuition

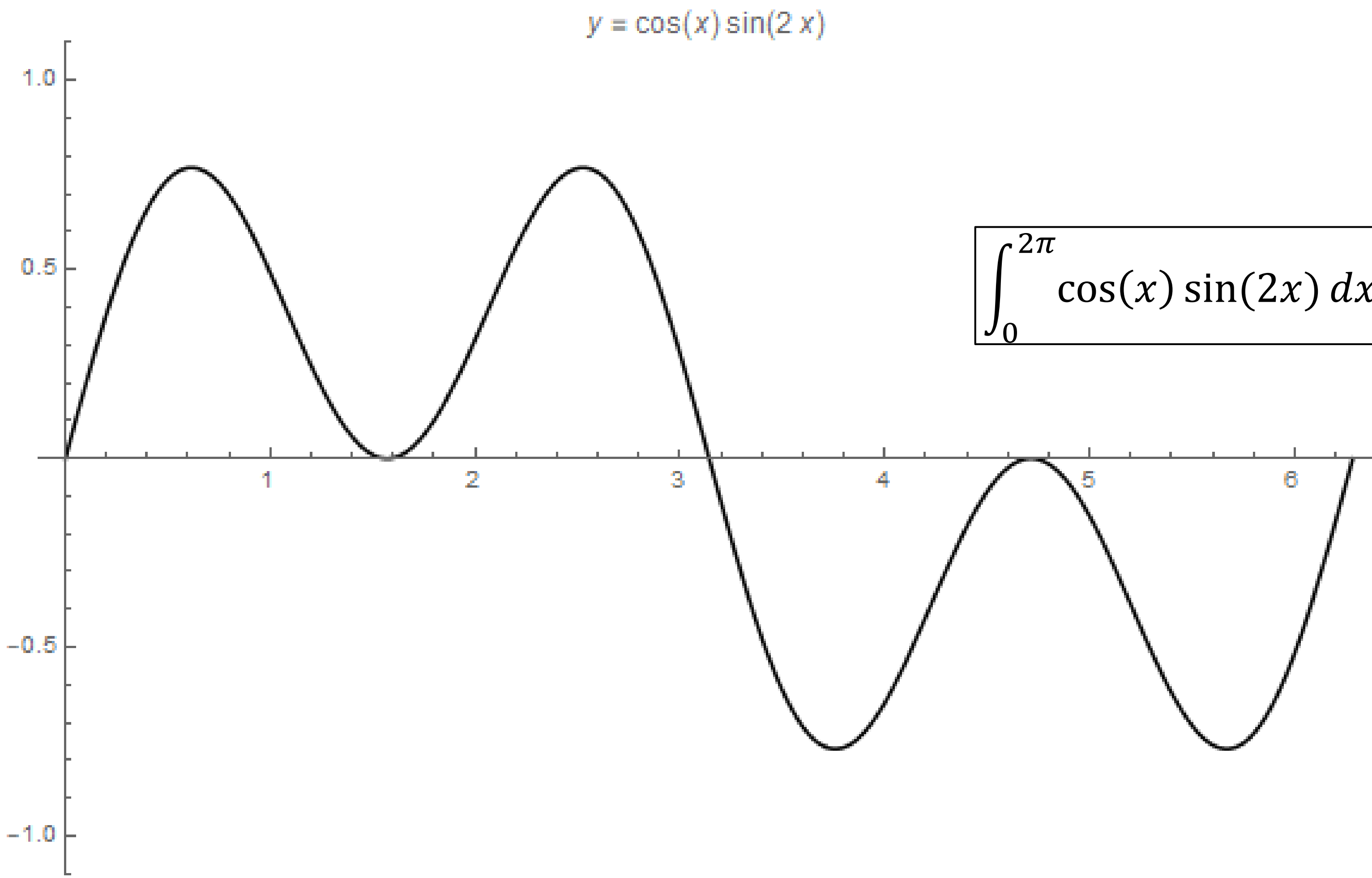


$$\int_0^{2\pi} \sin(x)\sin(2x)\,dx = 0$$

Fourier Analysis - Intuition



Fourier Analysis - Intuition



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

Fourier Analysis - Intuition

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

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

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

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

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

Discrete Fourier Analysis

- Converting a sampled complicated wave into a series of sampled ***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)

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

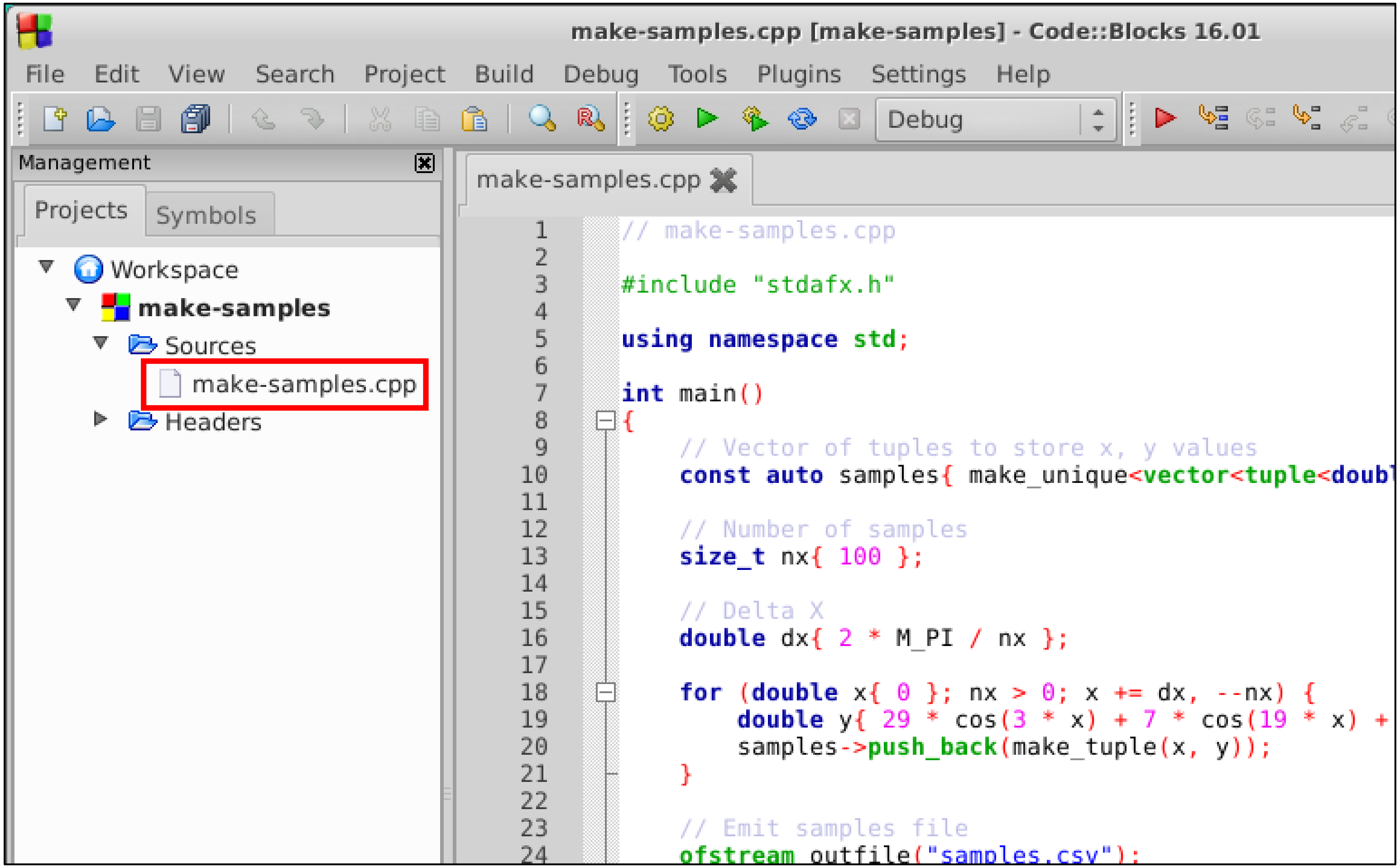
$$A_0 = \frac{\sum_{s=0}^{samples} y(t_s)}{samples}$$

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

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

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

Open Lab 1 - Make Samples for Waveform

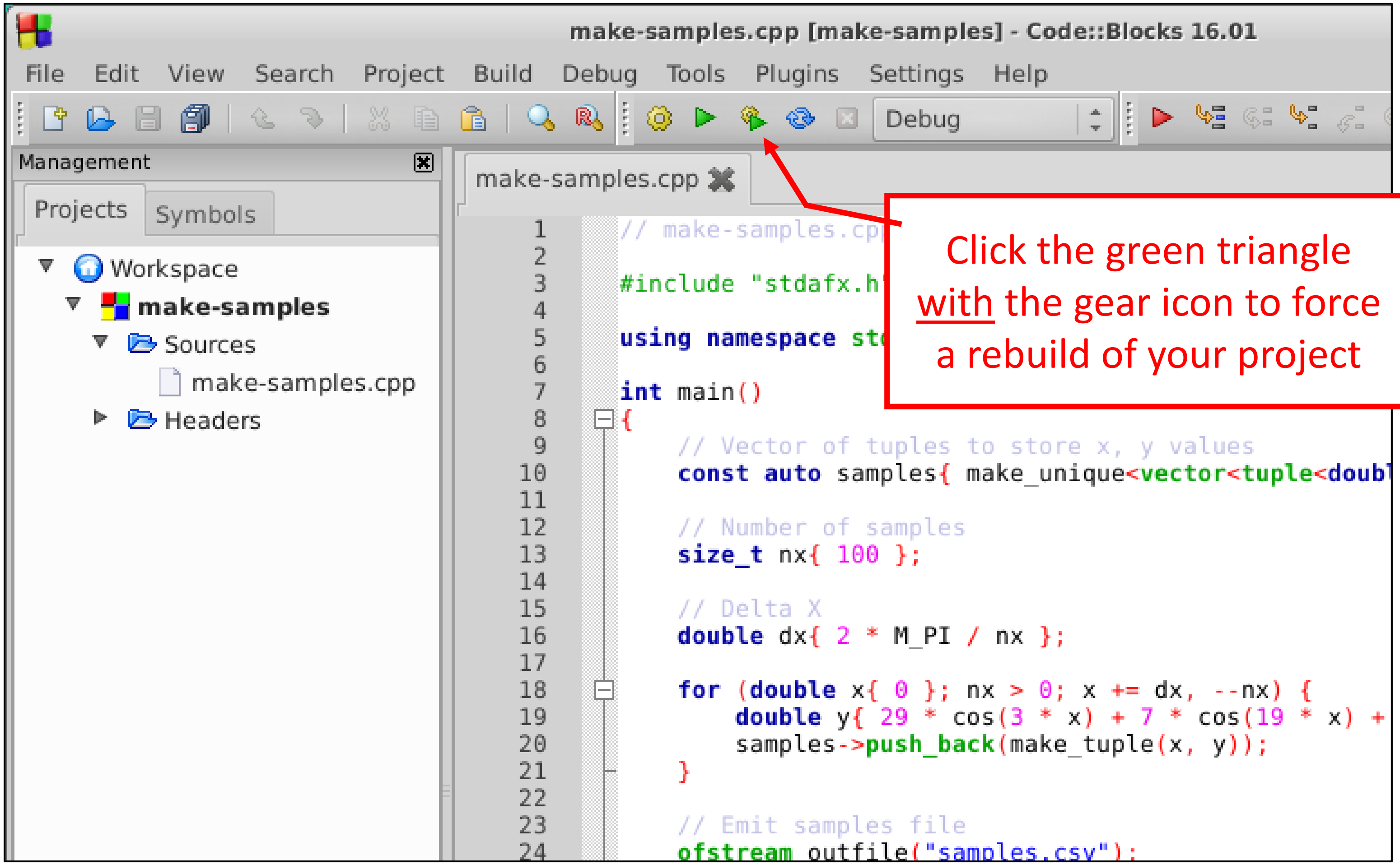


View Lab 1 - Make Samples for Waveform

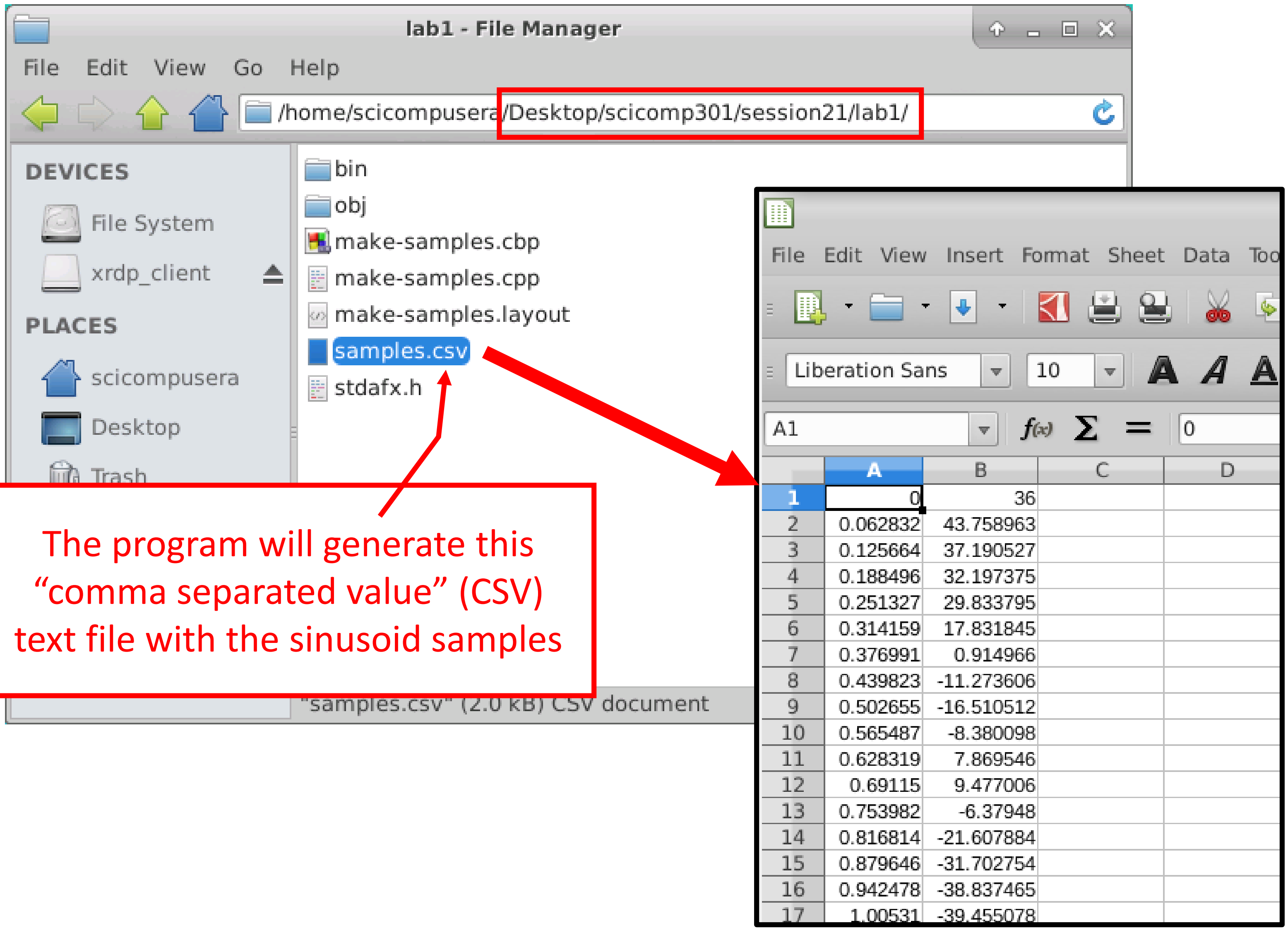
```
int main()
{
    // Vector of tuples to store x, y values
    const auto samples{ make_unique<vector<tuple<double, double>>>() };

    // 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));
    }

    // Emit samples file
    ofstream outfile("samples.csv");
    outfile.exceptions(ofstream::failbit);
    for (auto d : *samples)
    {
        outfile << fixed << get<0>(d) << ", ";
        outfile << fixed << get<1>(d) << endl;
    }
    outfile.flush();
    outfile.close();

    return 0;
}
```

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

Run Lab 1 - Make Samples for Waveform

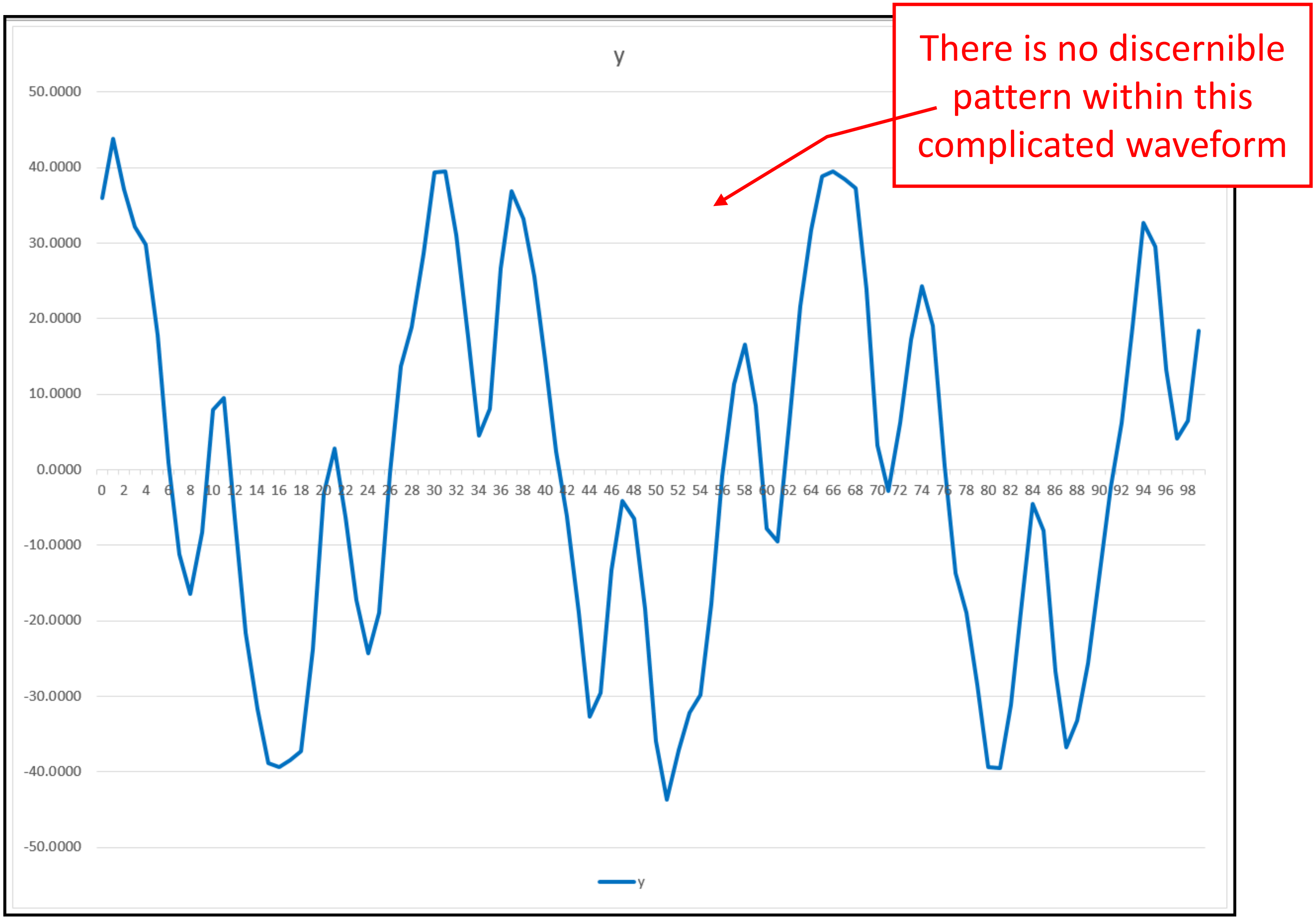


Check Lab 1 - Make Samples for Waveform

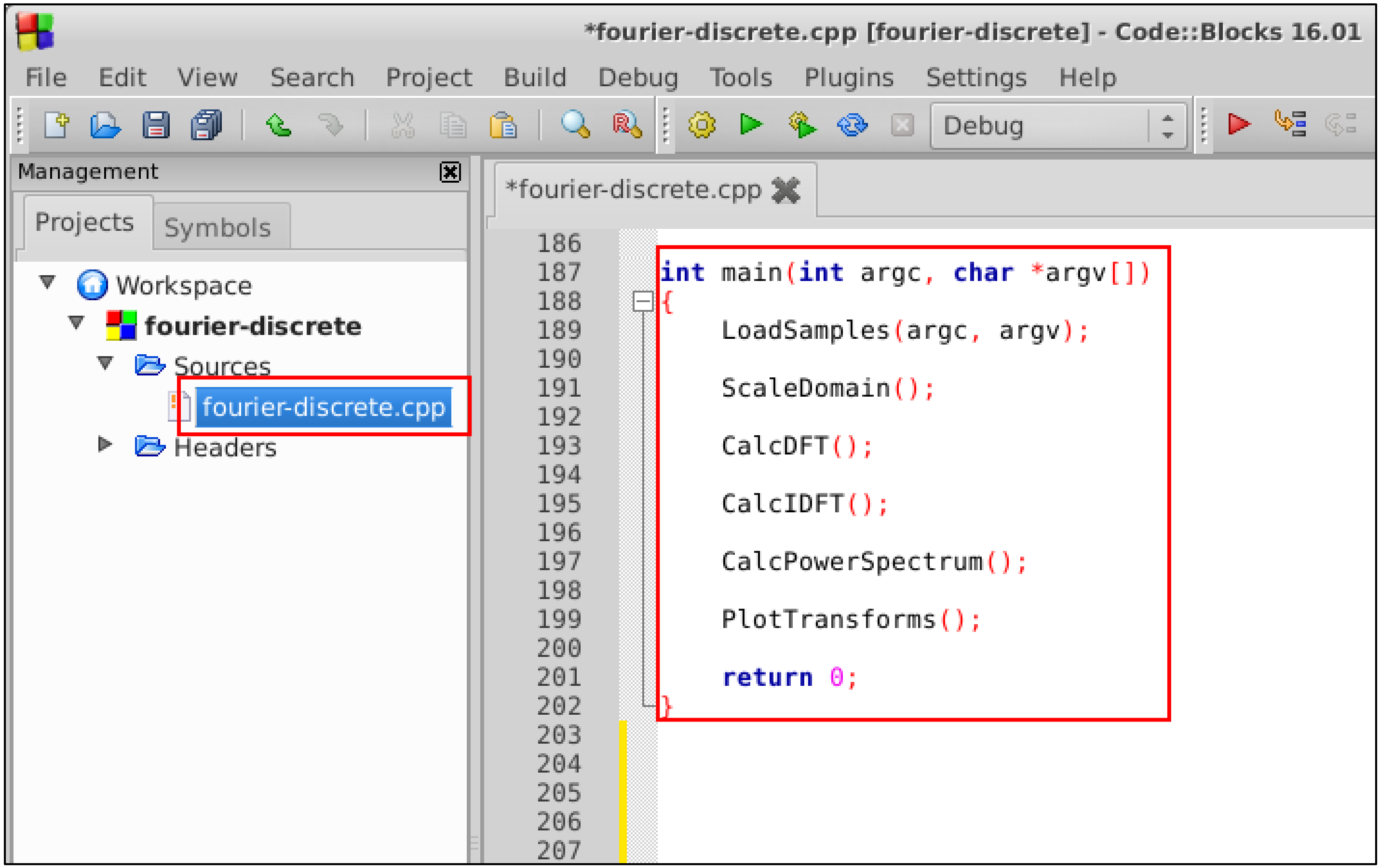


The program will generate this "comma separated value" (CSV) text file with the sinusoid samples

100 Samples of a Complicated Wave



Open Lab 2 – Discrete Fourier Transform



View Lab 2 – Discrete Fourier Transform

```
void LoadSamples(int argc, char *argv[])
{
    if (argc < 2) {
        cout << "Error: missing samples filename";
        exit(-1);
    }

    string filename{ argv[1] };
    ifstream infile(filename);
    if (!infile.is_open()) {
        cout << "Error: Unable to open file "
            "\"" << filename << "\"" << endl;
        exit(-1);
    }

    double x, y;
    string line;
    stringstream ss;

    ss.imbue(locale(locale(), new csv_reader()));
    while (getline(infile, line)) {
        ss.str(line);
        ss >> x >> y;
        ss.clear();
        xAct.push_back(x);
        yAct.push_back(y);
        xOrd.push_back(xAct.size() - 1);
    }

    infile.close();
}
```

View Lab 2 – Discrete Fourier Transform

```
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);
    }
}
```

Multiply every term (integer frequency)…

…by every sample data value…

…to find the contribution of each simple wave

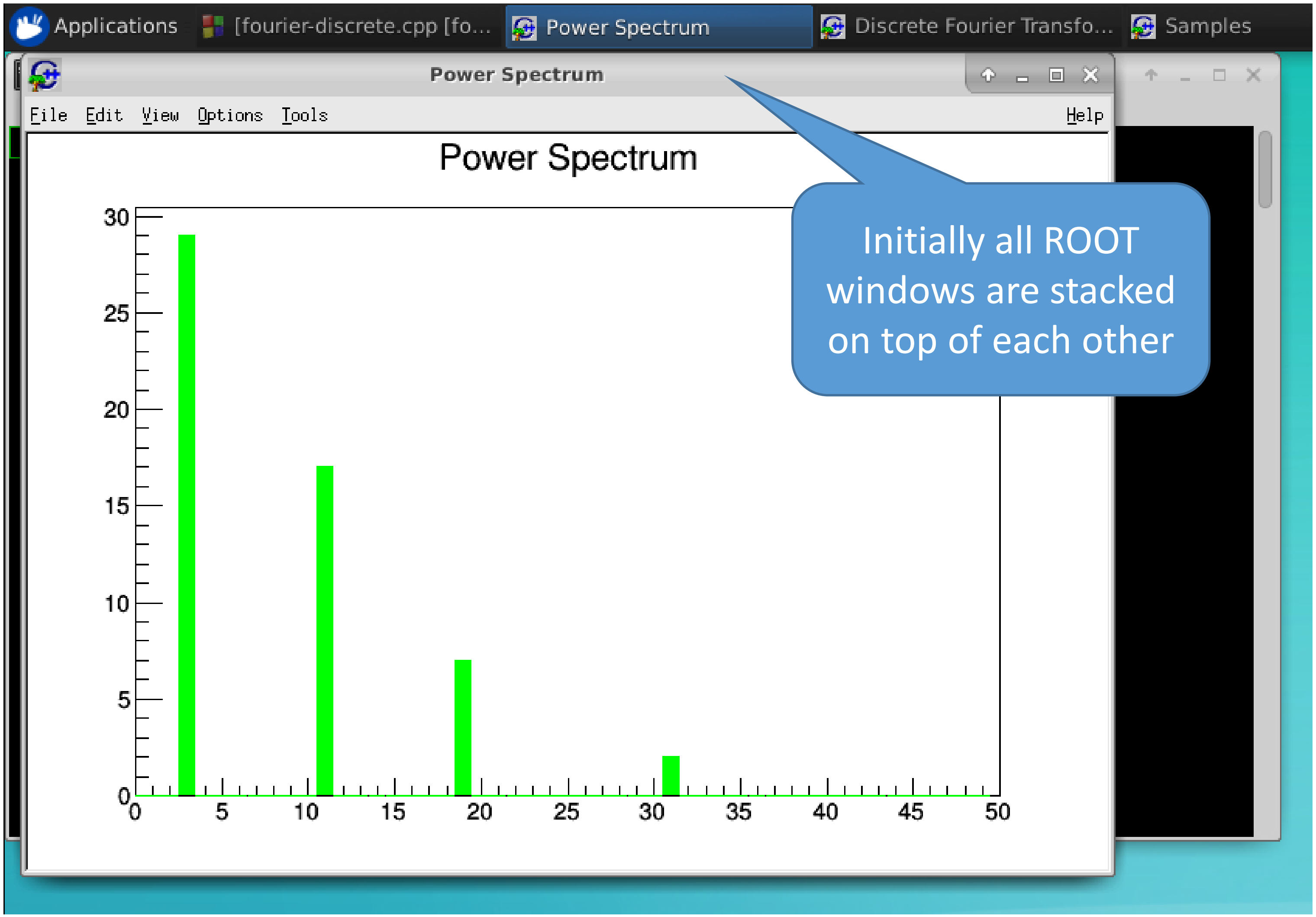
The amplitude of each simple wave is just the **mean value** of the product of every sample of the complicated wave multiplied by that simple wave

View Lab 2 – Discrete Fourier Transform

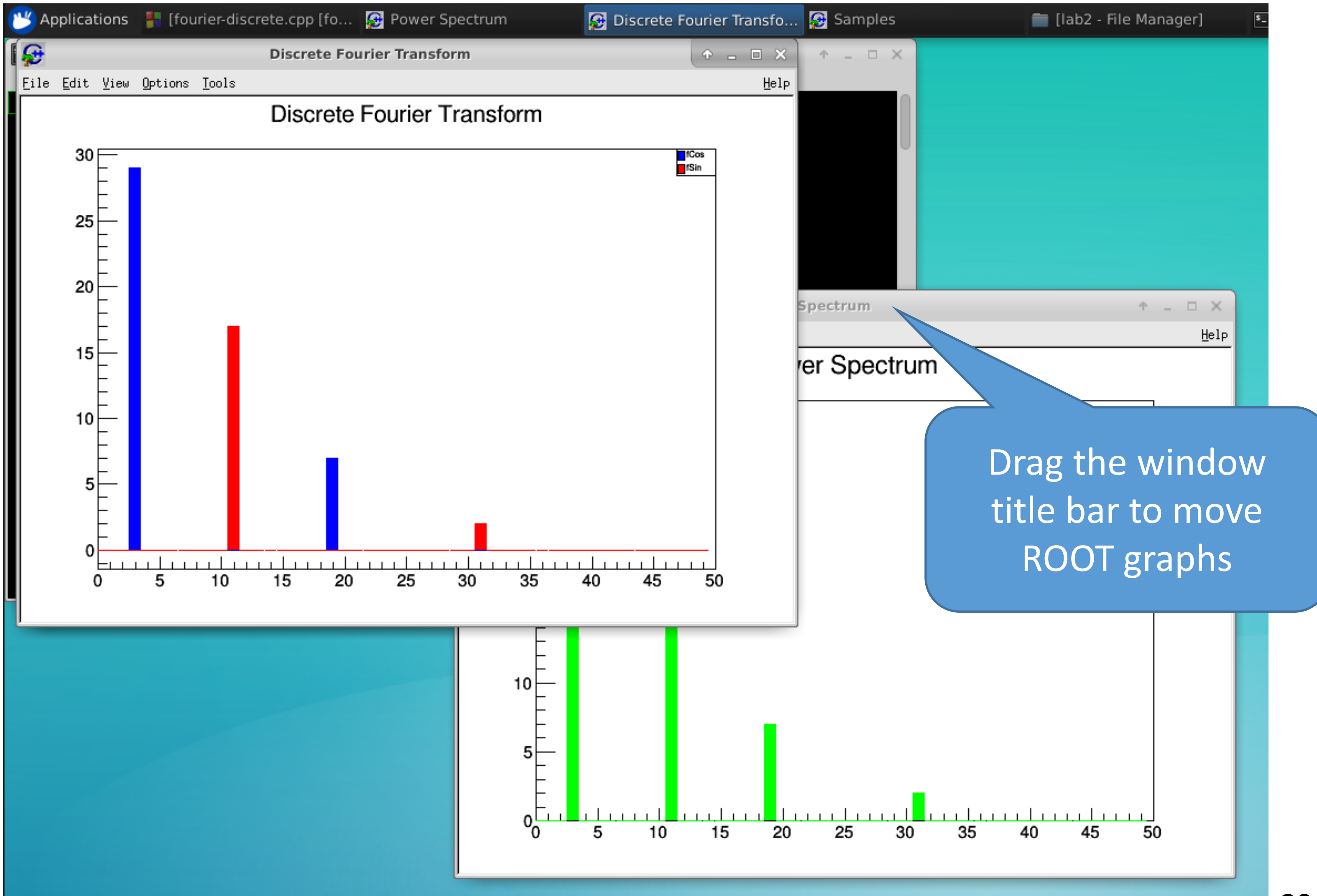
```
void CalcIDFT()
{
    size_t sample_count{ yAct.size() };
    size_t term_count{ fCos.size() };

    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)));
}
```

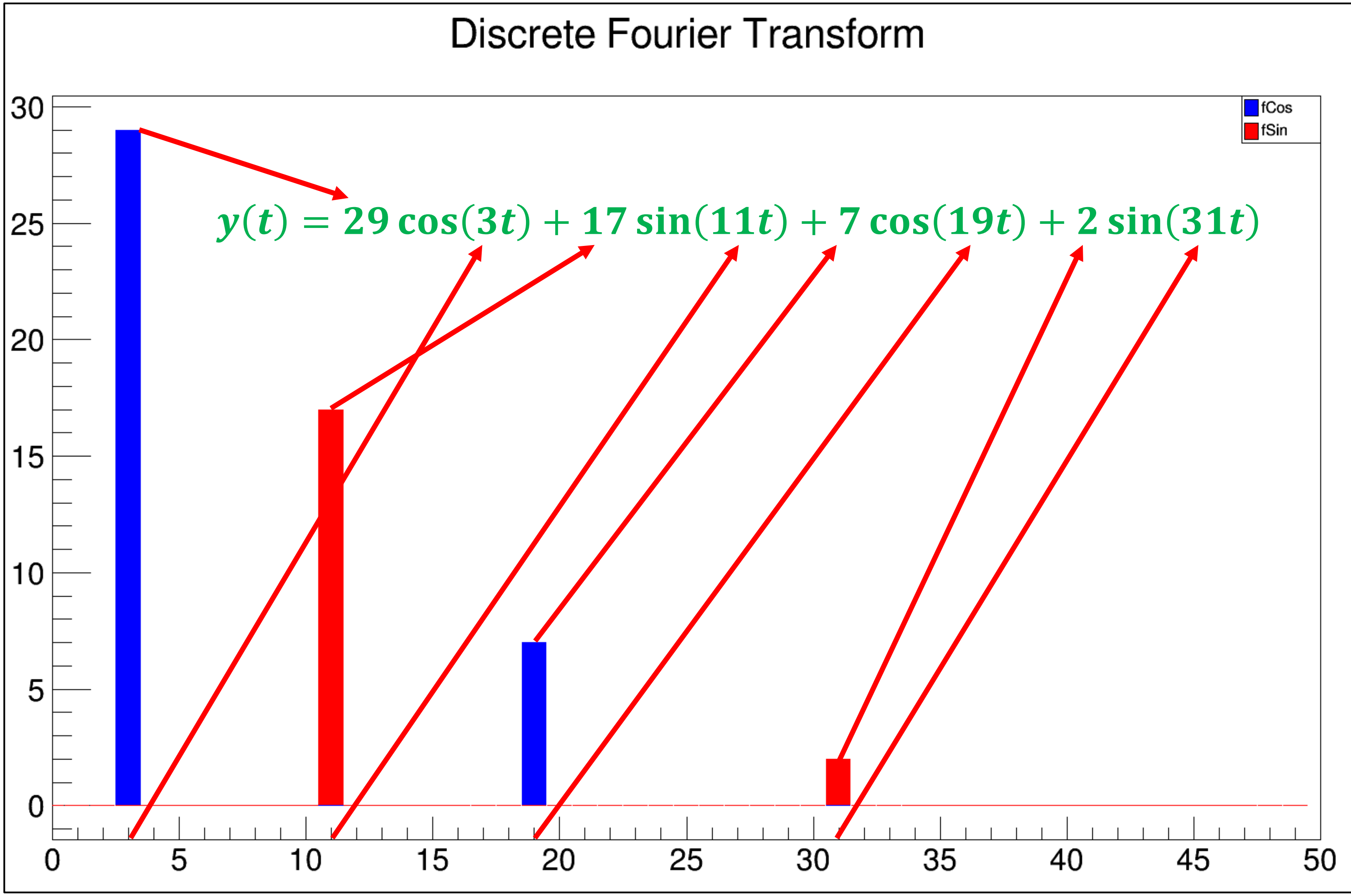
The reconstructed wave is just the **linear sum** of the amplitude of each simple cosine and sine wave

Run Lab 2 – Discrete Fourier Transform

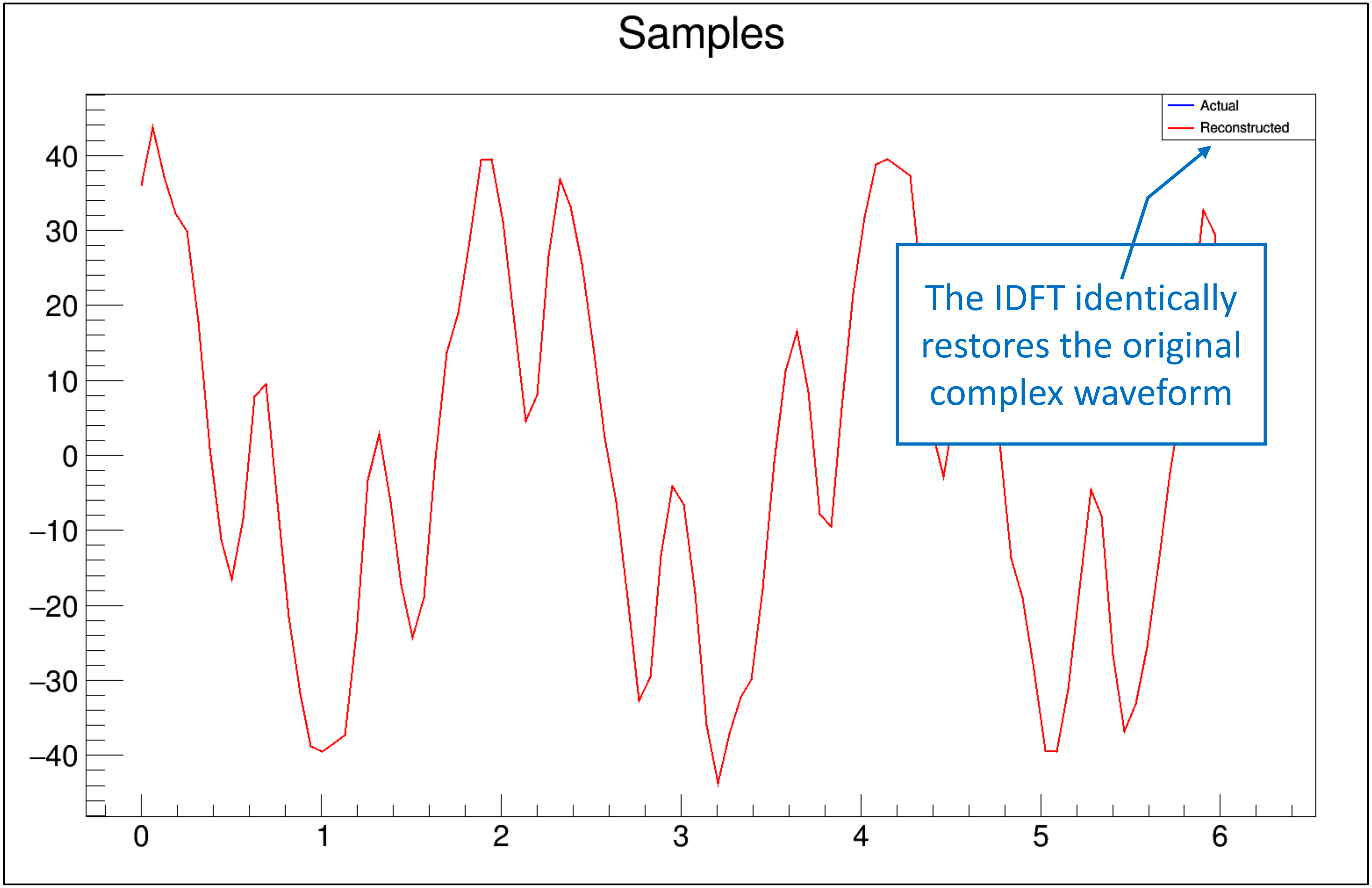


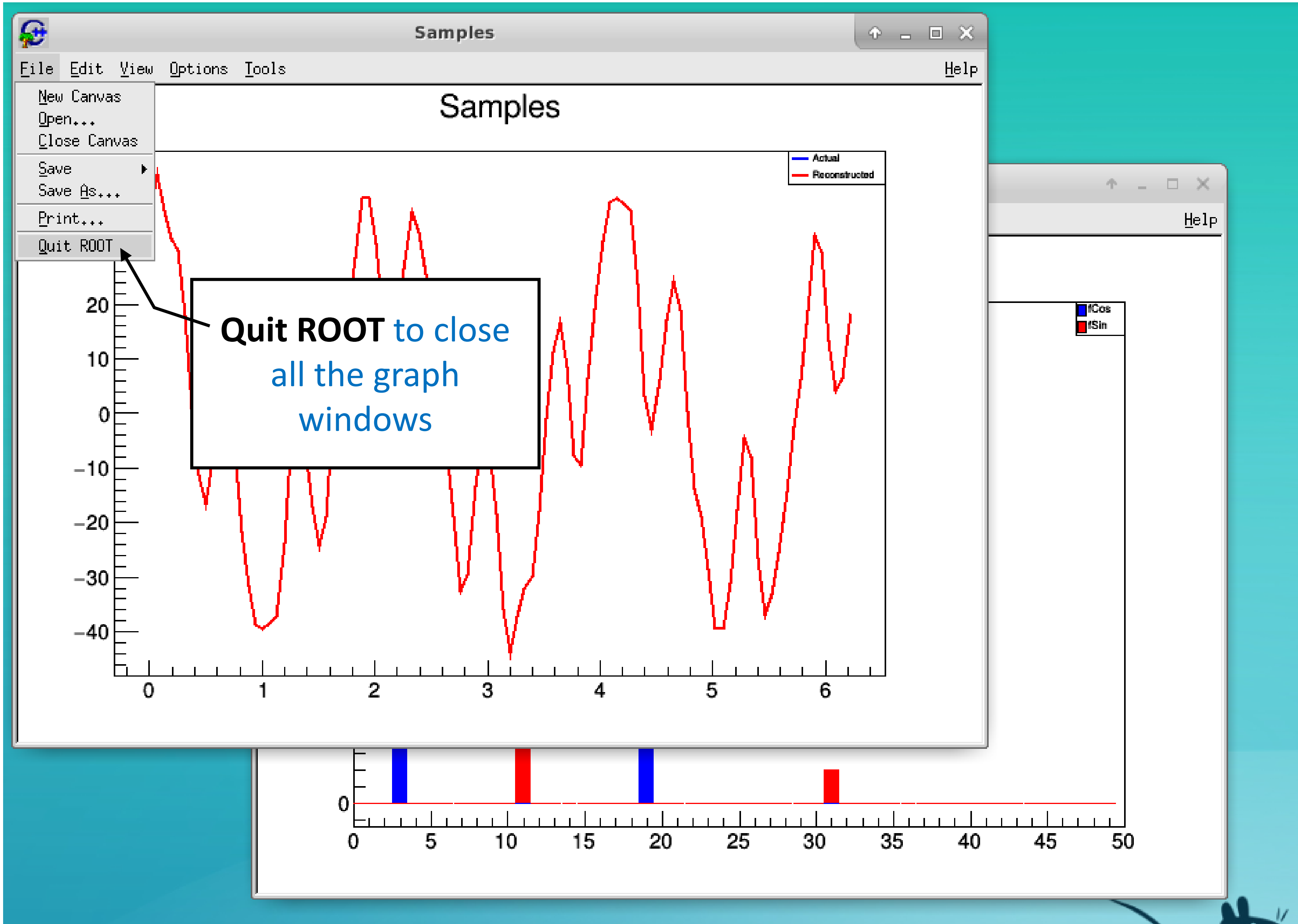
Check Lab 2 – Discrete Fourier Transform




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

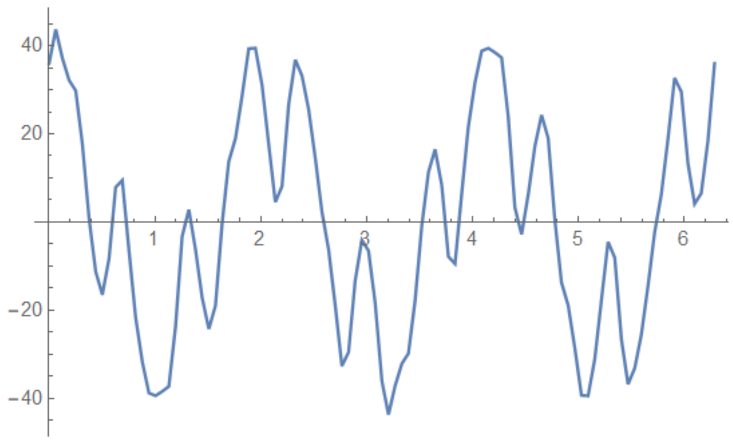
Check Lab 2 – Discrete Fourier Transform



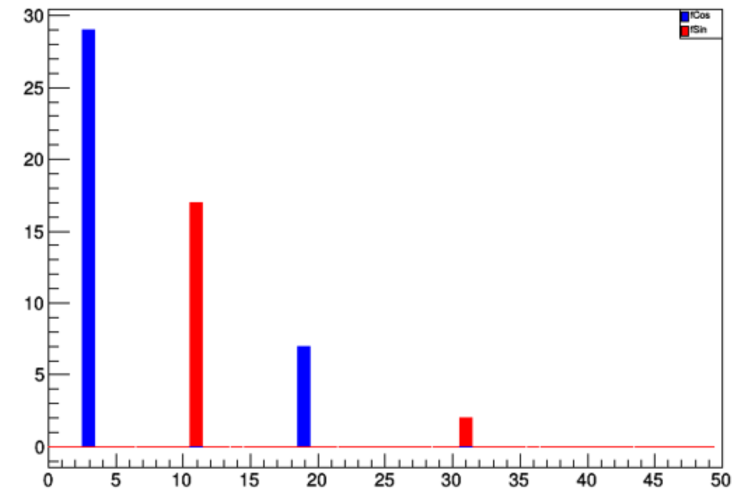

Samples
File Edit View Options Tools Help
New Canvas
Open...
Close Canvas
Save
Save As...
Print...
Quit ROOT
Samples
Actual
Reconstructed
Quit ROOT to close all the graph windows
fCos
fSin
Help

Information Density

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



Time domain



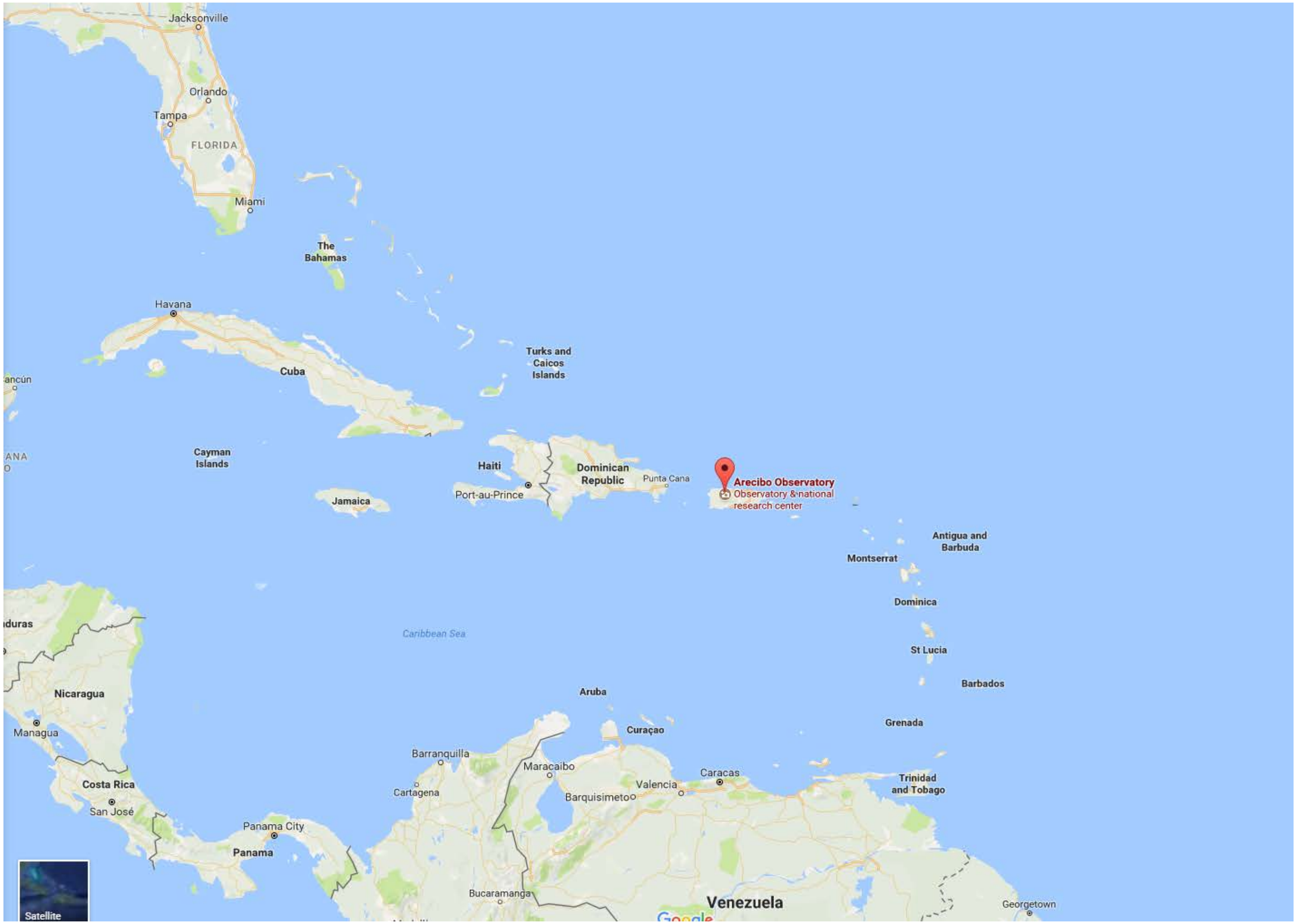
Frequency domain

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!

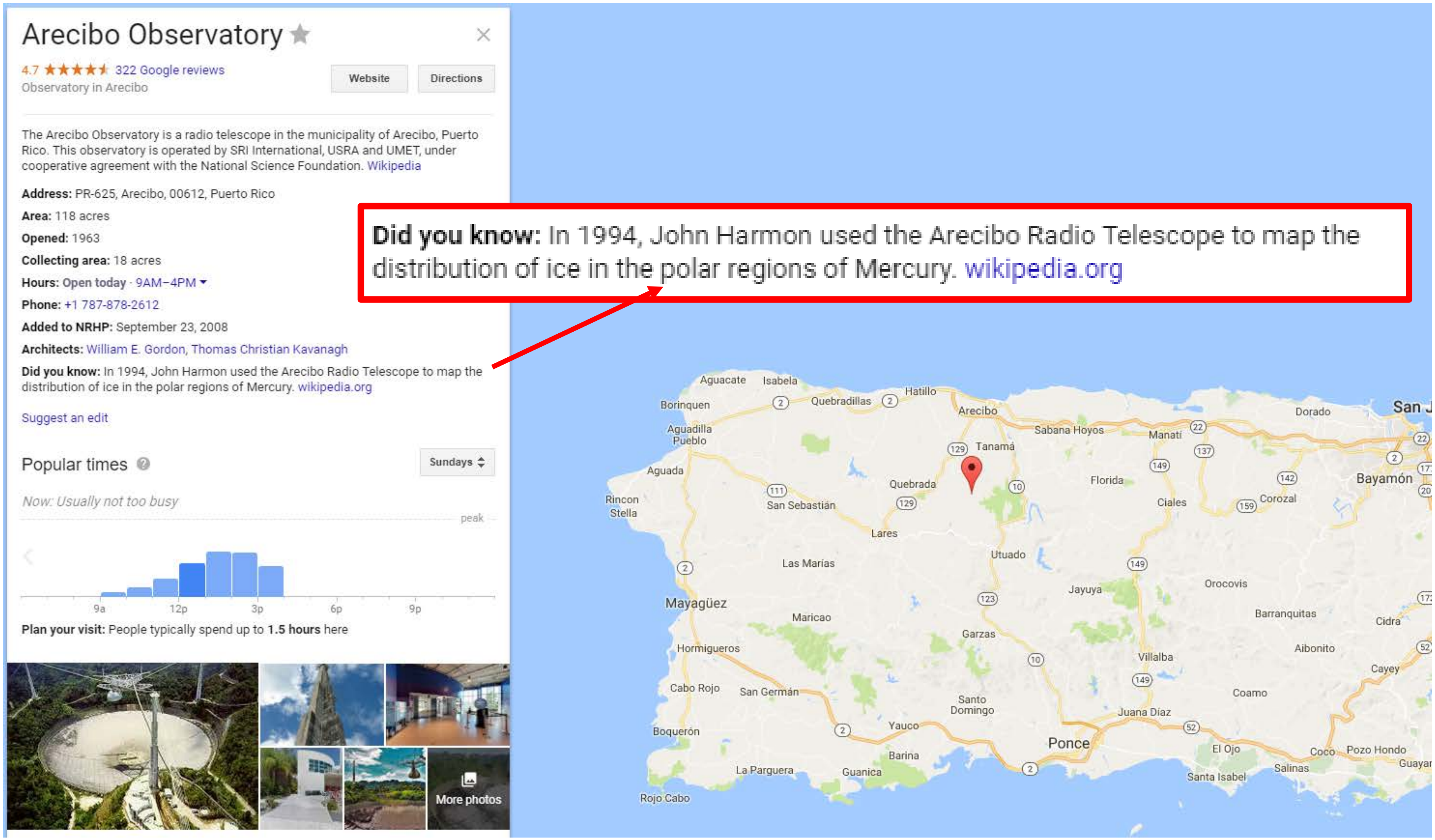
Analysis of Space Signals

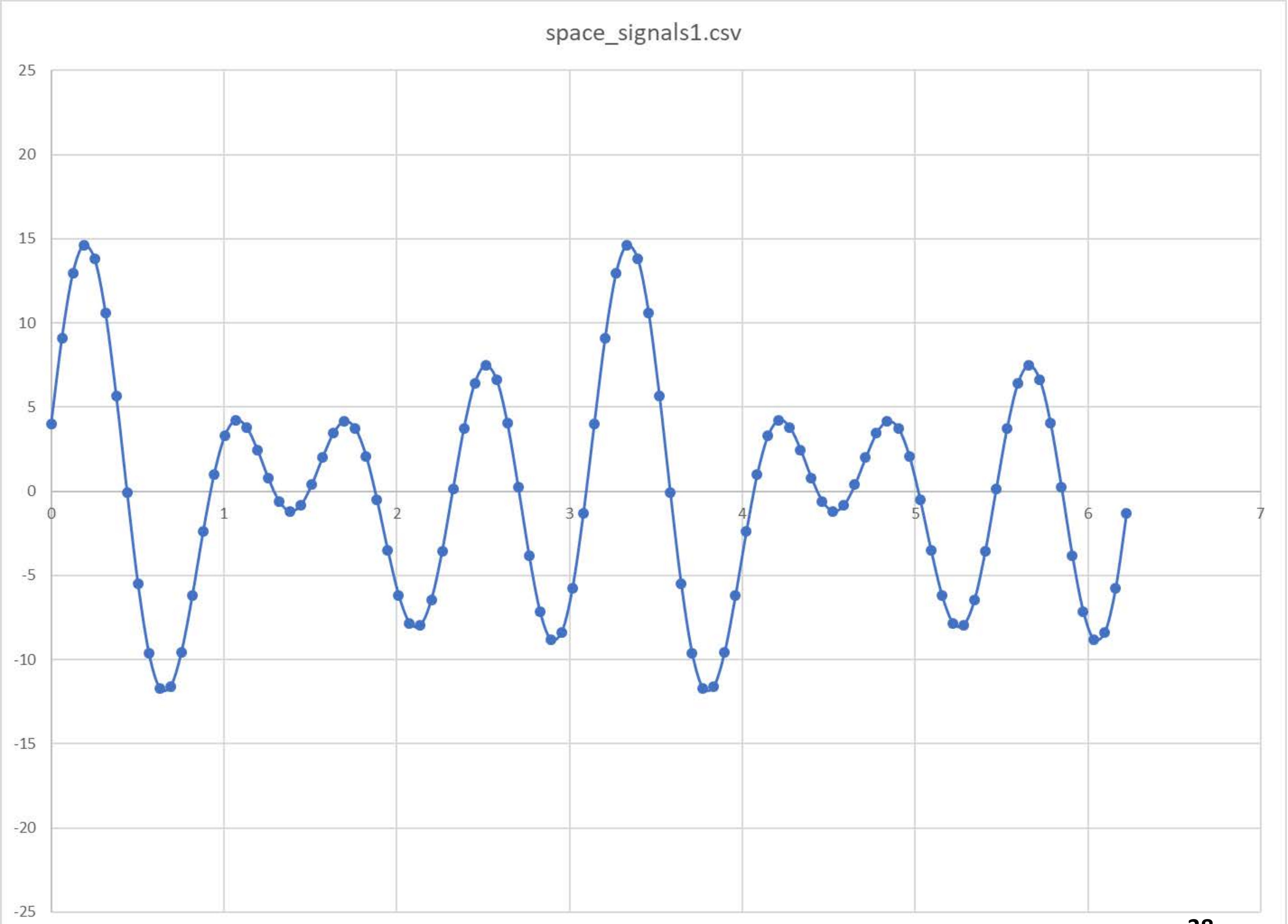
- The **Arecibo Radio Observatory** has detected three candidate signals originating from deep space
- Your task is to perform **Fourier Analysis** on each signal to determine which one **is more likely** to have been a broadcast from an extraterrestrial intelligence

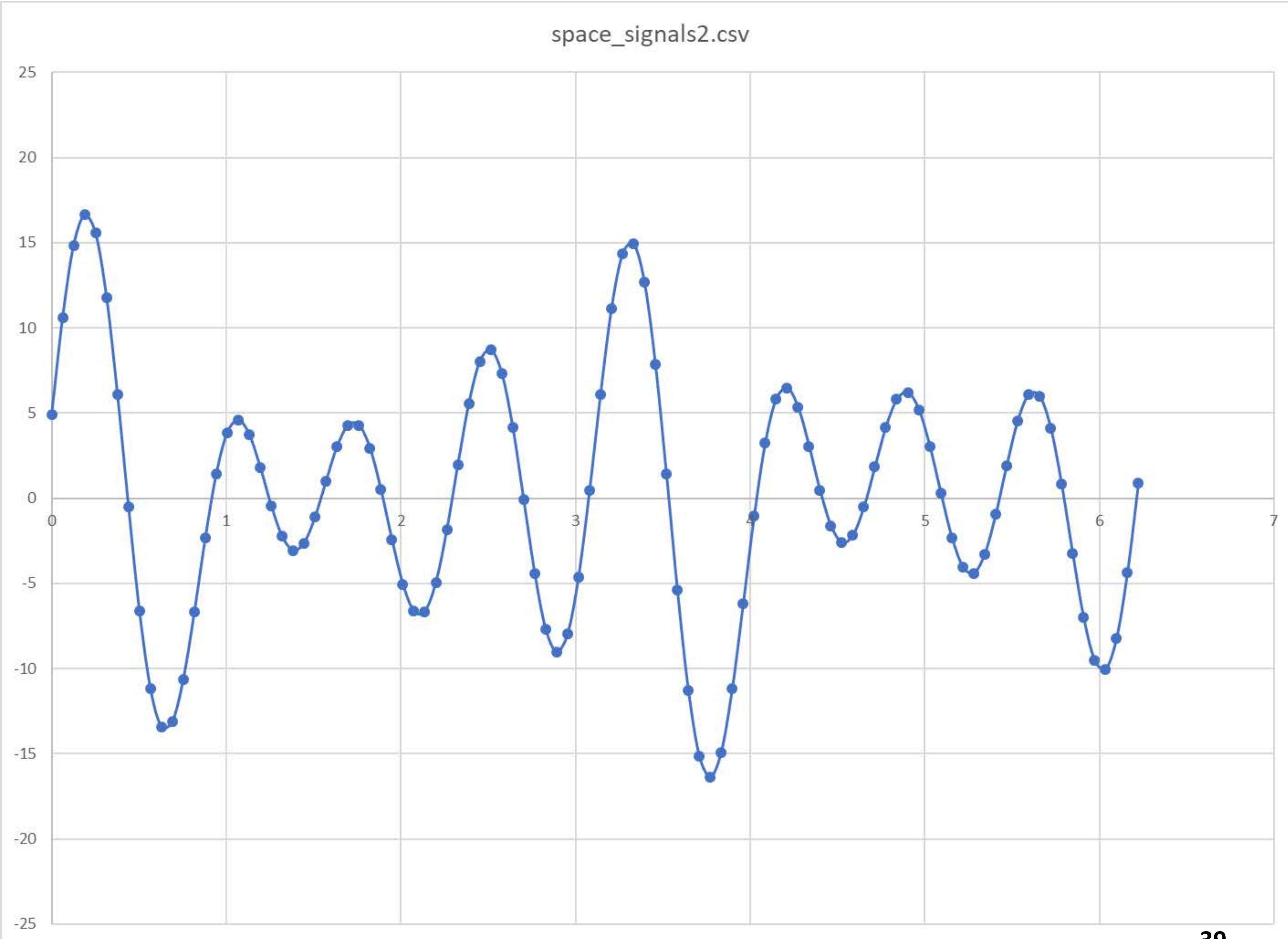
Analysis of Space Signals

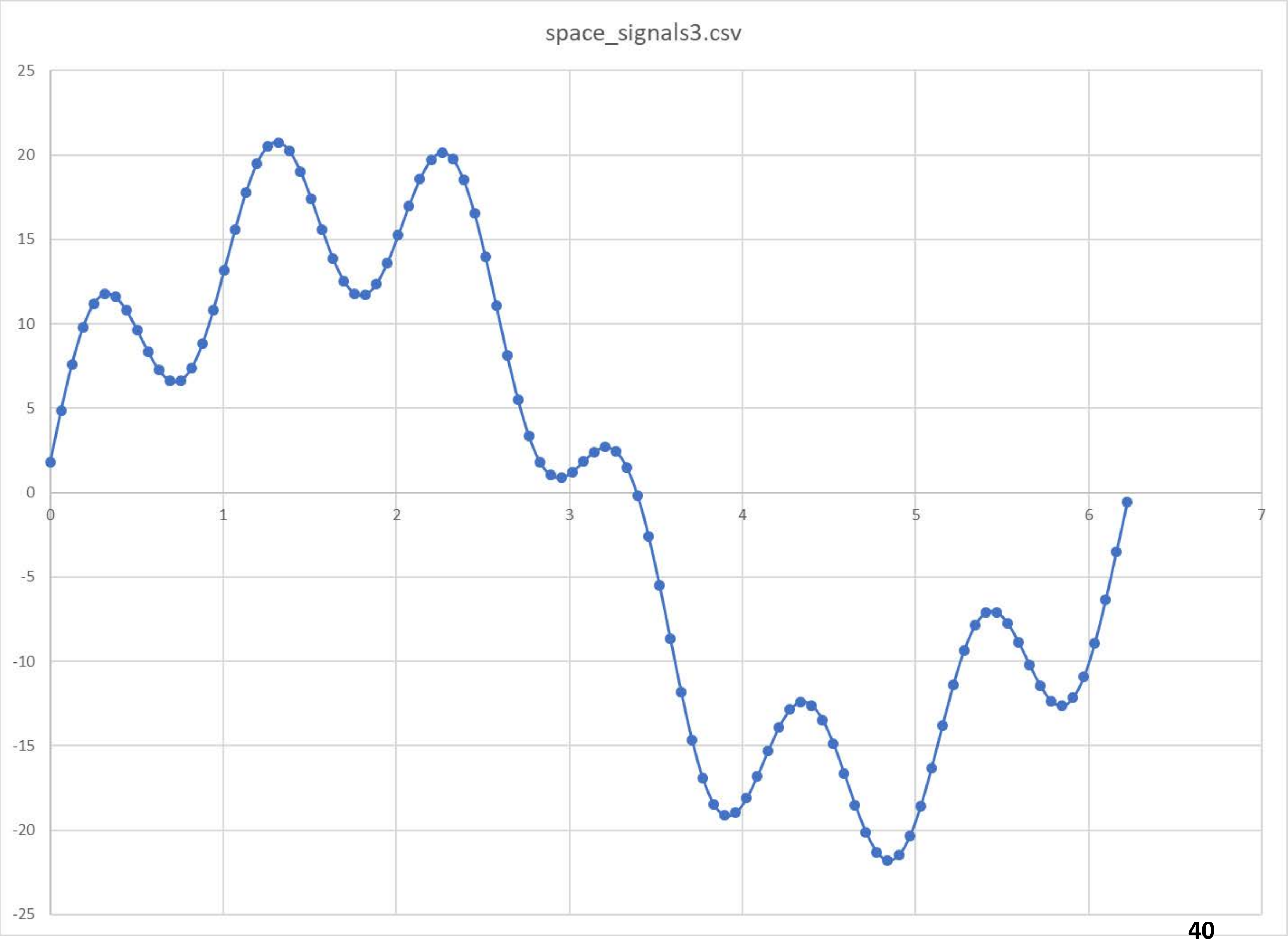


Analysis of Space Signals

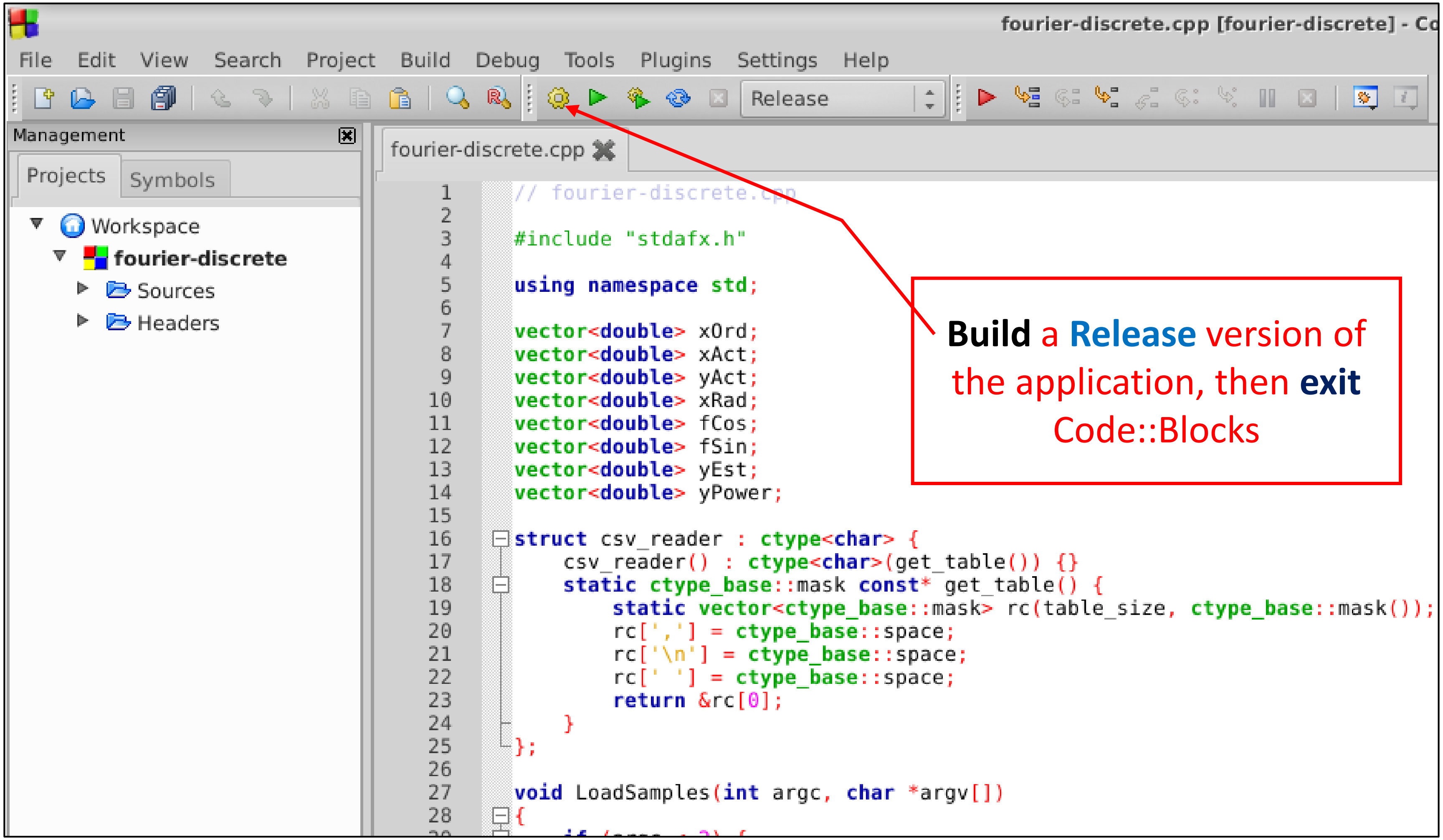



space_signals1.csv
25
20
15
10
5
0
-5
-10
-15
-20
-25
0
1
2
3
4
5
6
7

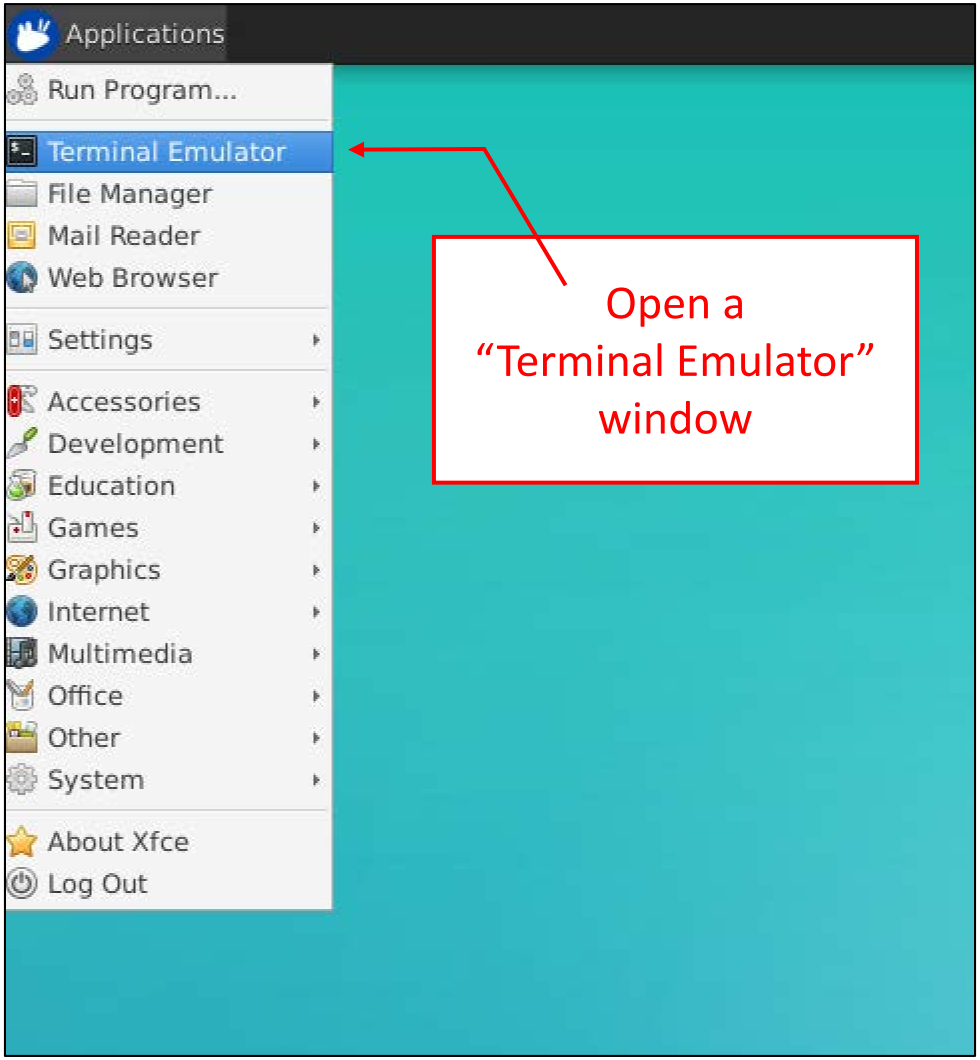

space_signals2.csv
25
20
15
10
5
0
-5
-10
-15
-20
-25
0
1
2
3
4
5
6
7


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

Build Lab 3 – Space Signals



Run Lab 3 – Space Signals



Run Lab 3 – Space Signals

Terminal - scicompusera@ip-172-31-71-74: ~/Desktop/scicomp301/session21/lab3/bin/Release

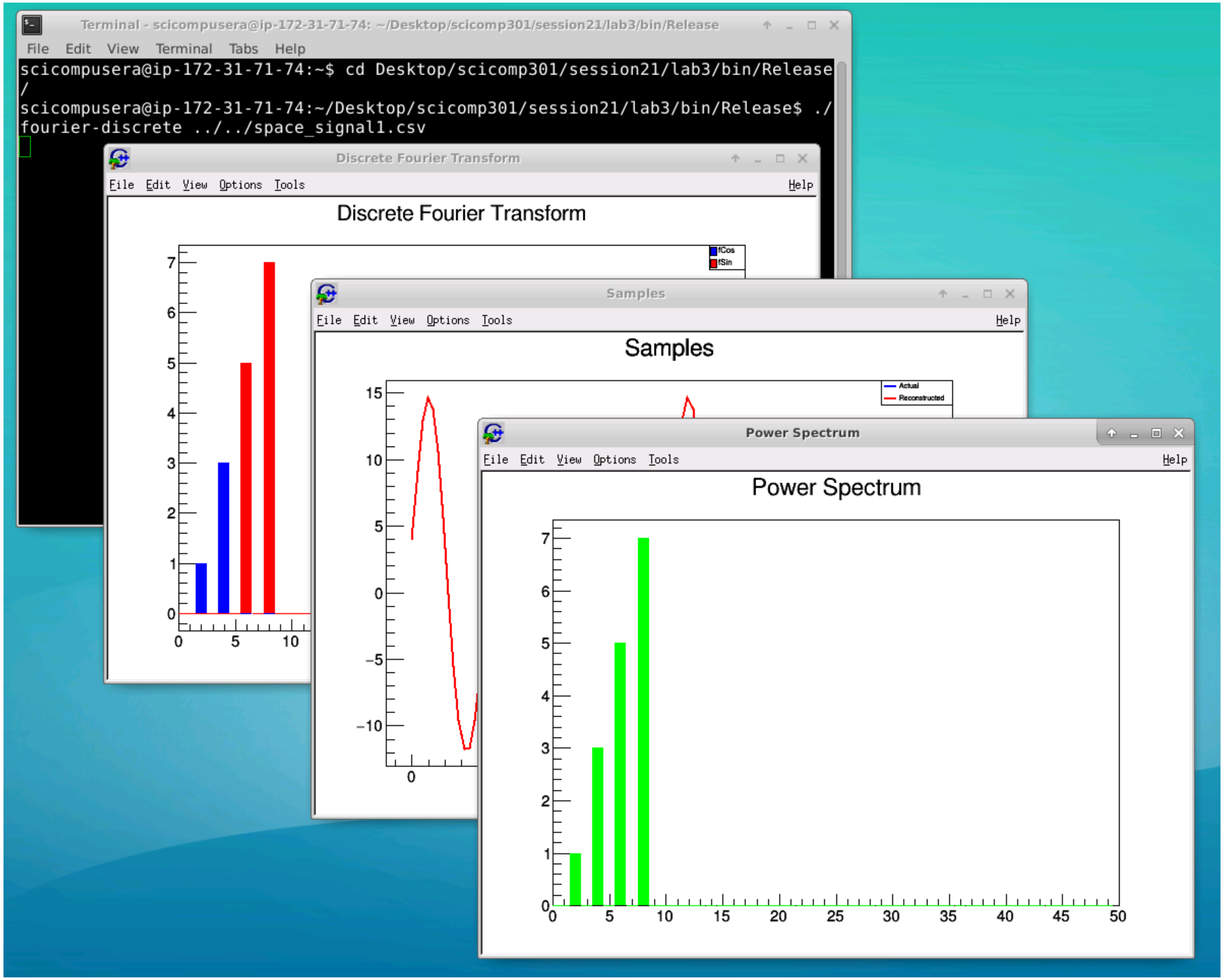
File Edit View Terminal Tabs Help

```
scicompusera@ip-172-31-71-74:~$ cd Desktop/scicomp301/session21/lab3/bin/Release
/
scicompusera@ip-172-31-71-74:~/Desktop/scicomp301/session21/lab3/bin/Release$ ./
fourier-discrete ../../space_signal1.csv
```

Enter these commands, pressing ENTER at the end of each line:

```
cd Desktop/scicomp301/session21/lab3/bin/Release
./fourier-discrete ../../space_signal1.csv
```

Check Lab 3 – Space Signals



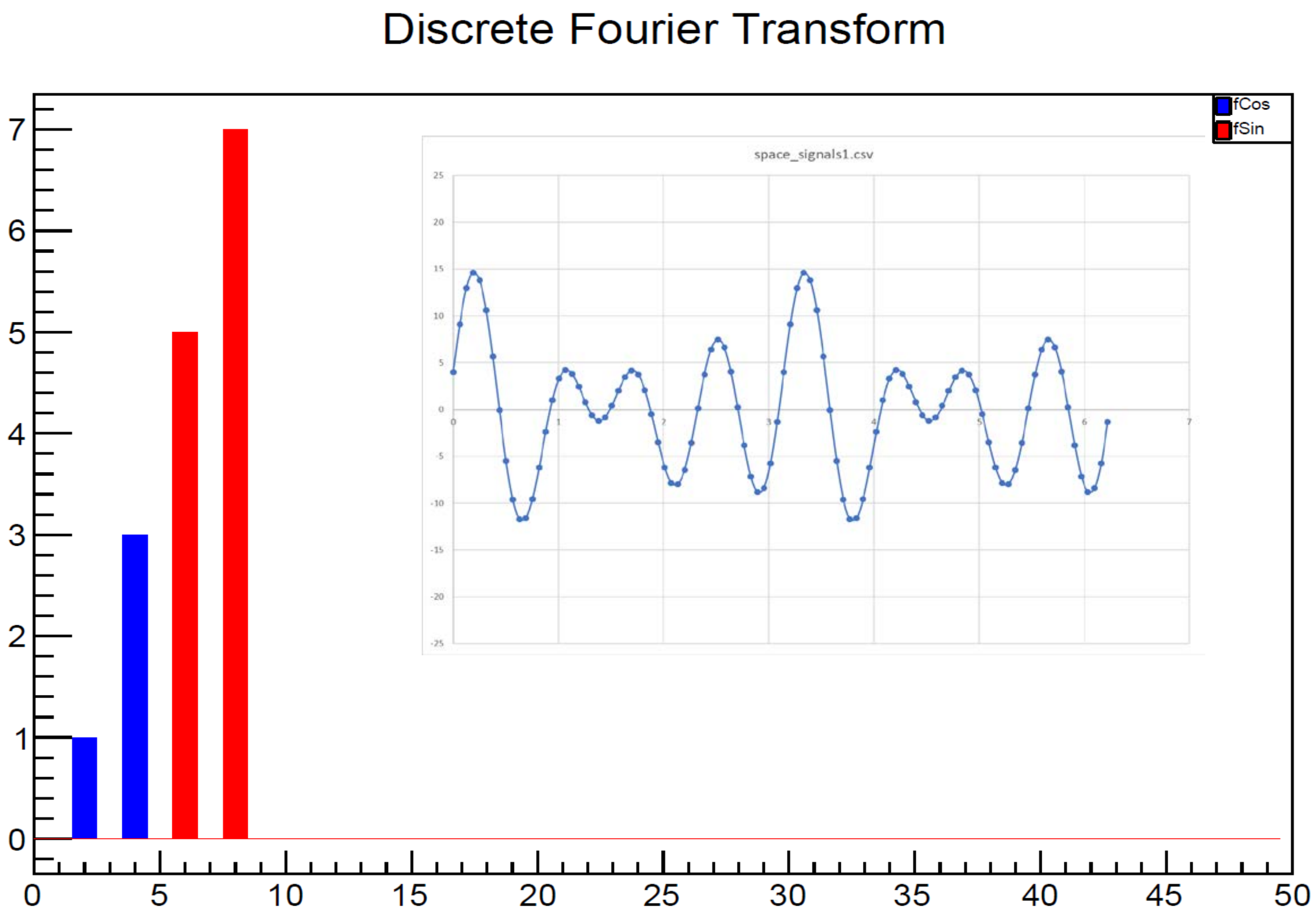
Check Lab 3 – Space Signals

- Remember to select "**File…Quit ROOT**" menu option to close all the windows
- Return to the Terminal Emulator window to analyze the other two space signals CSV files: *(type these commands)*

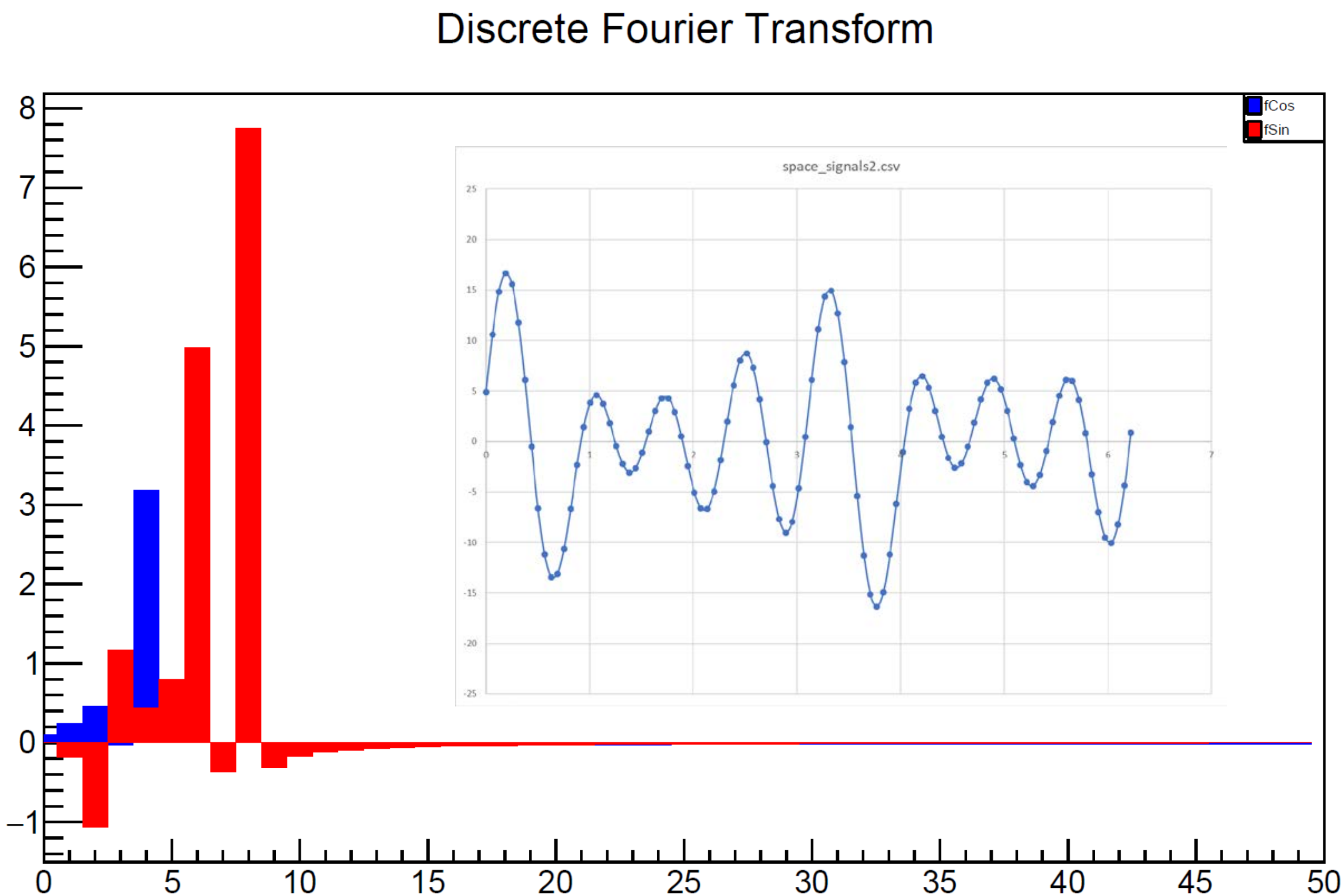
```
./fourier-discrete ../../space_signal2.csv
./fourier-discrete ../../space_signal3.csv
```

- Which space signal was likely made by *intelligent* beings? **Why**?

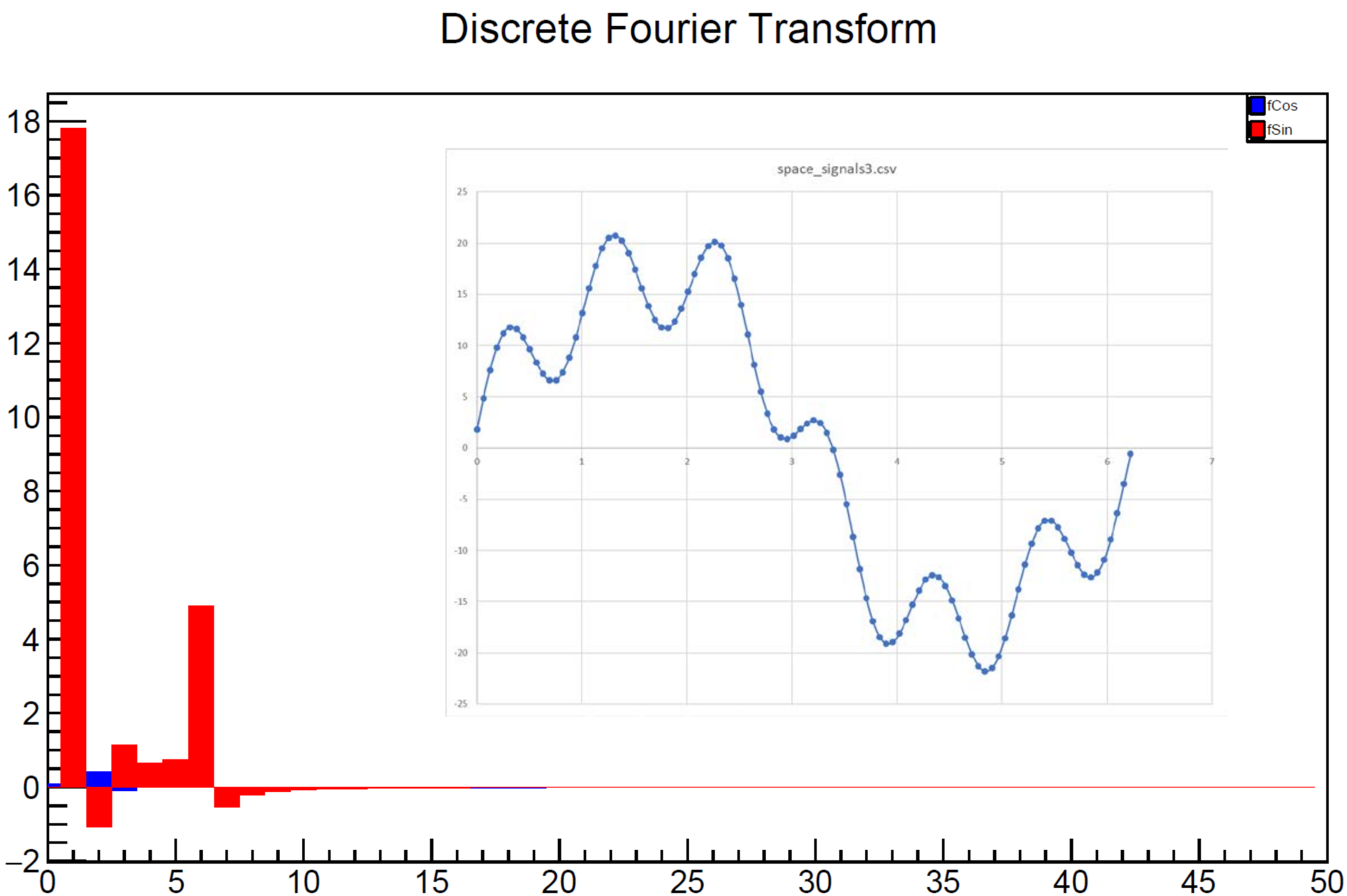
space_signal1



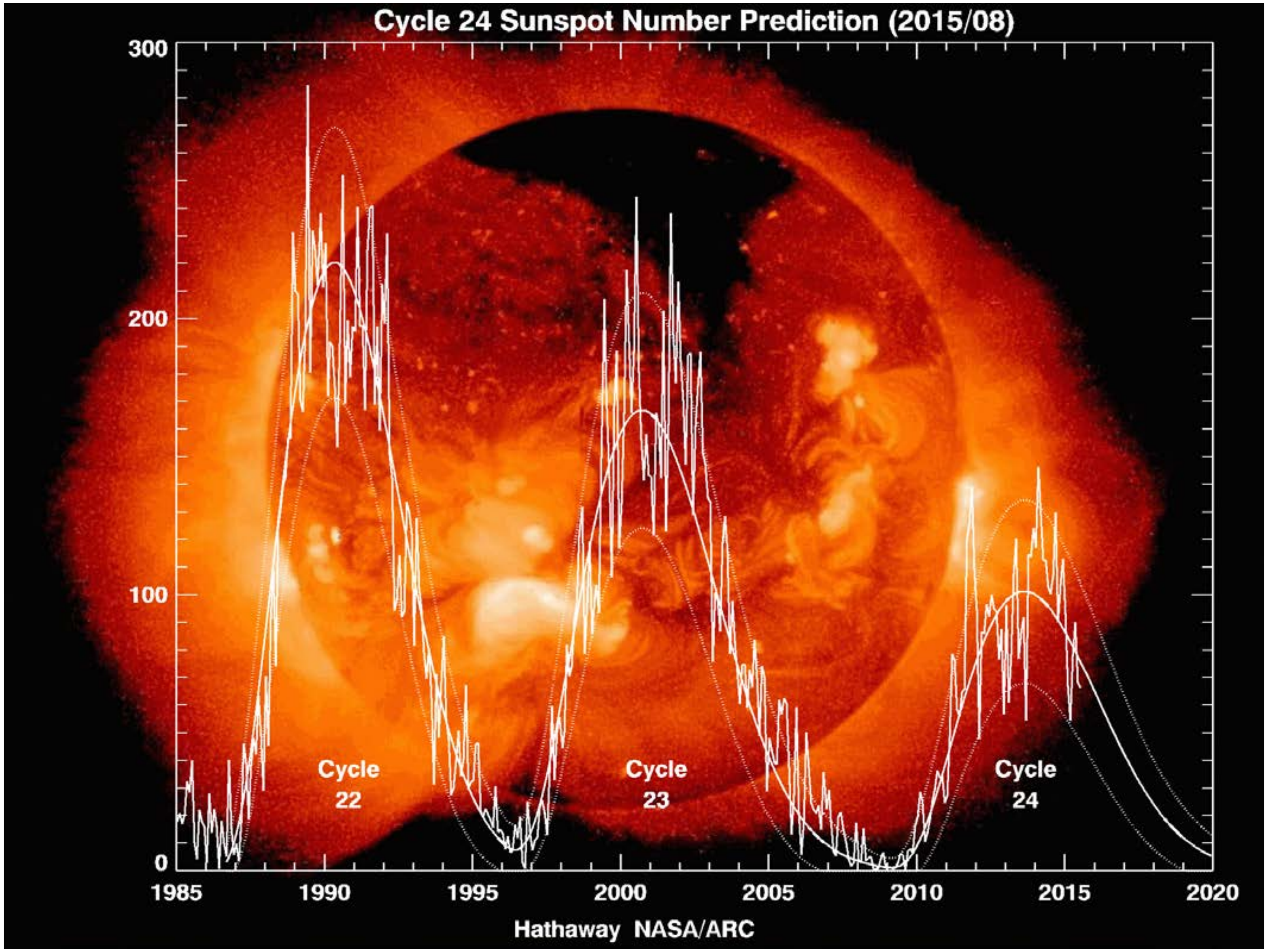
space_signal2



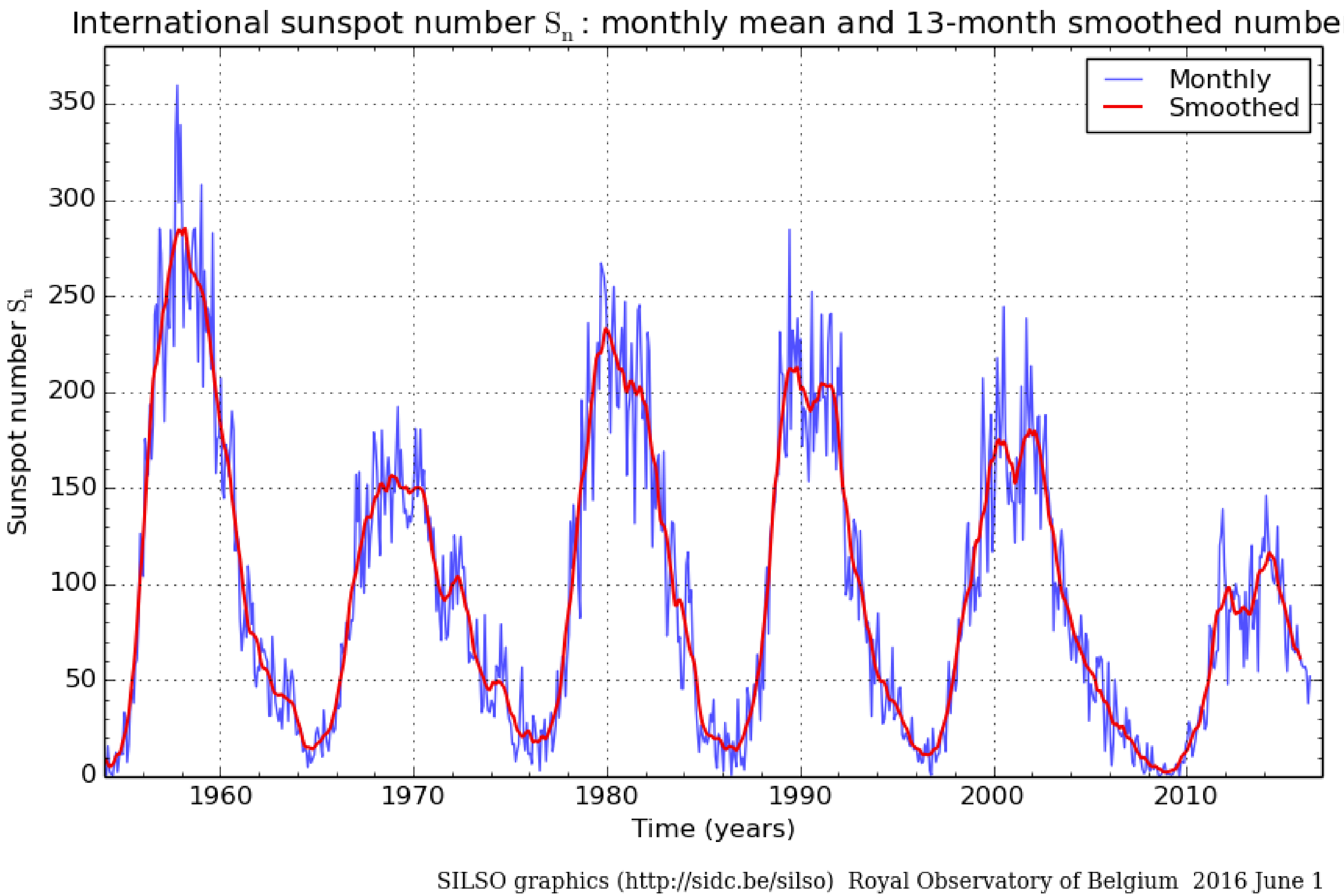
space_signal3



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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
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.

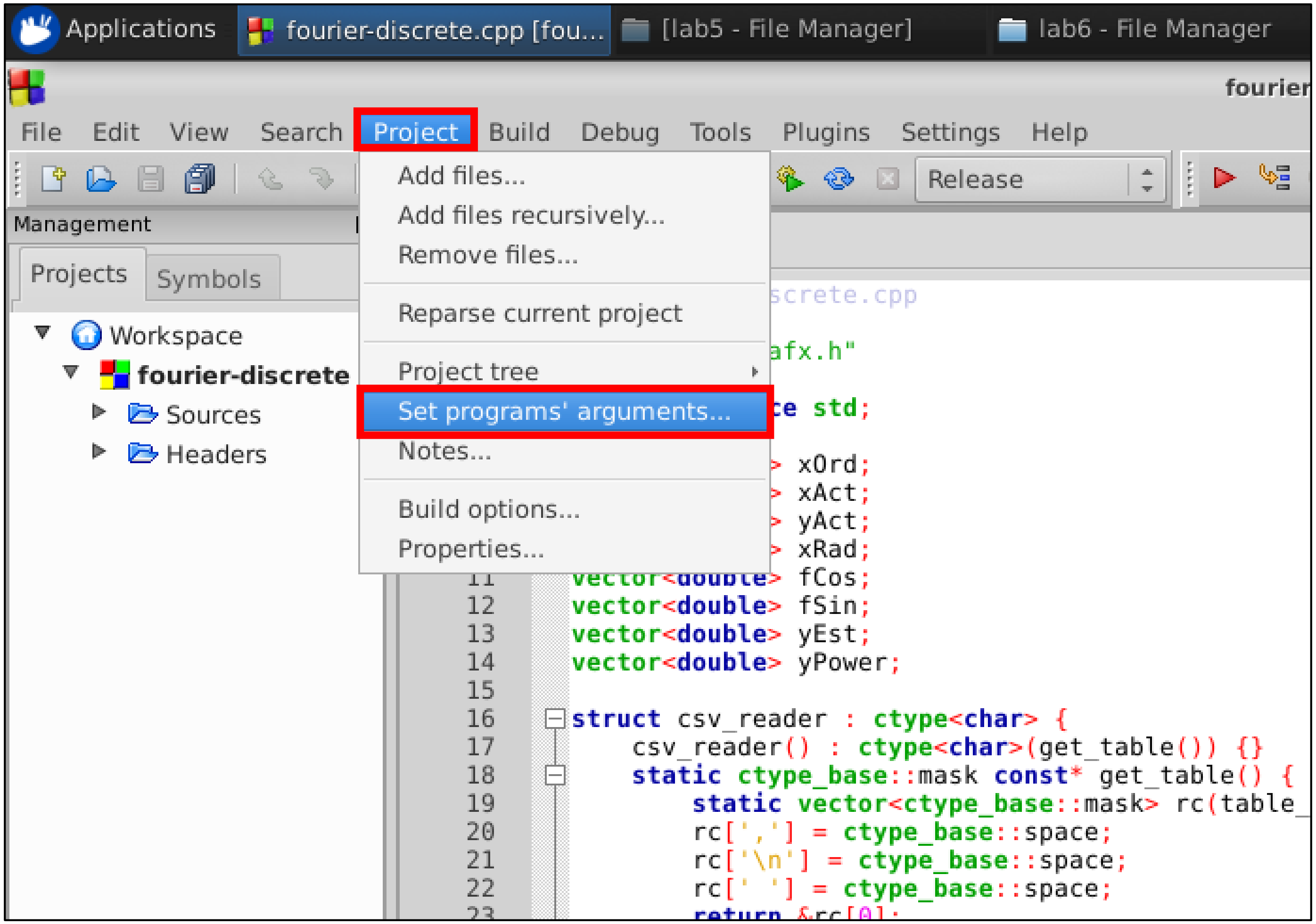
History [edit]

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.

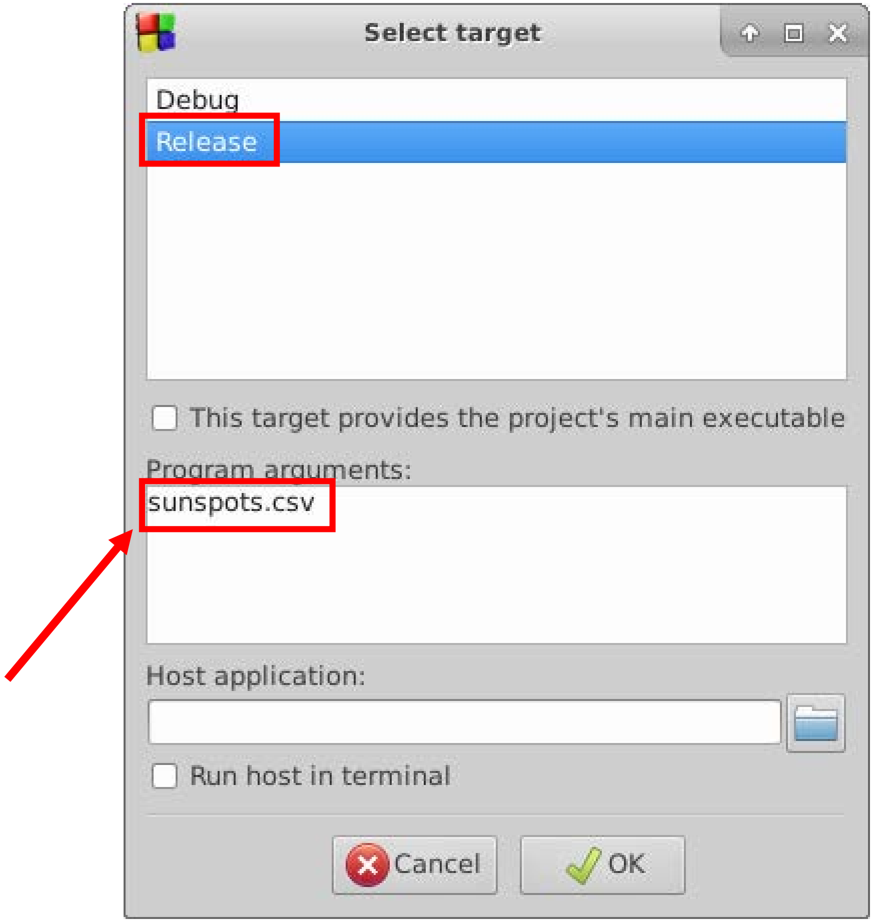
This number has been collected and tabulated by researchers for over 150 years.[2]

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

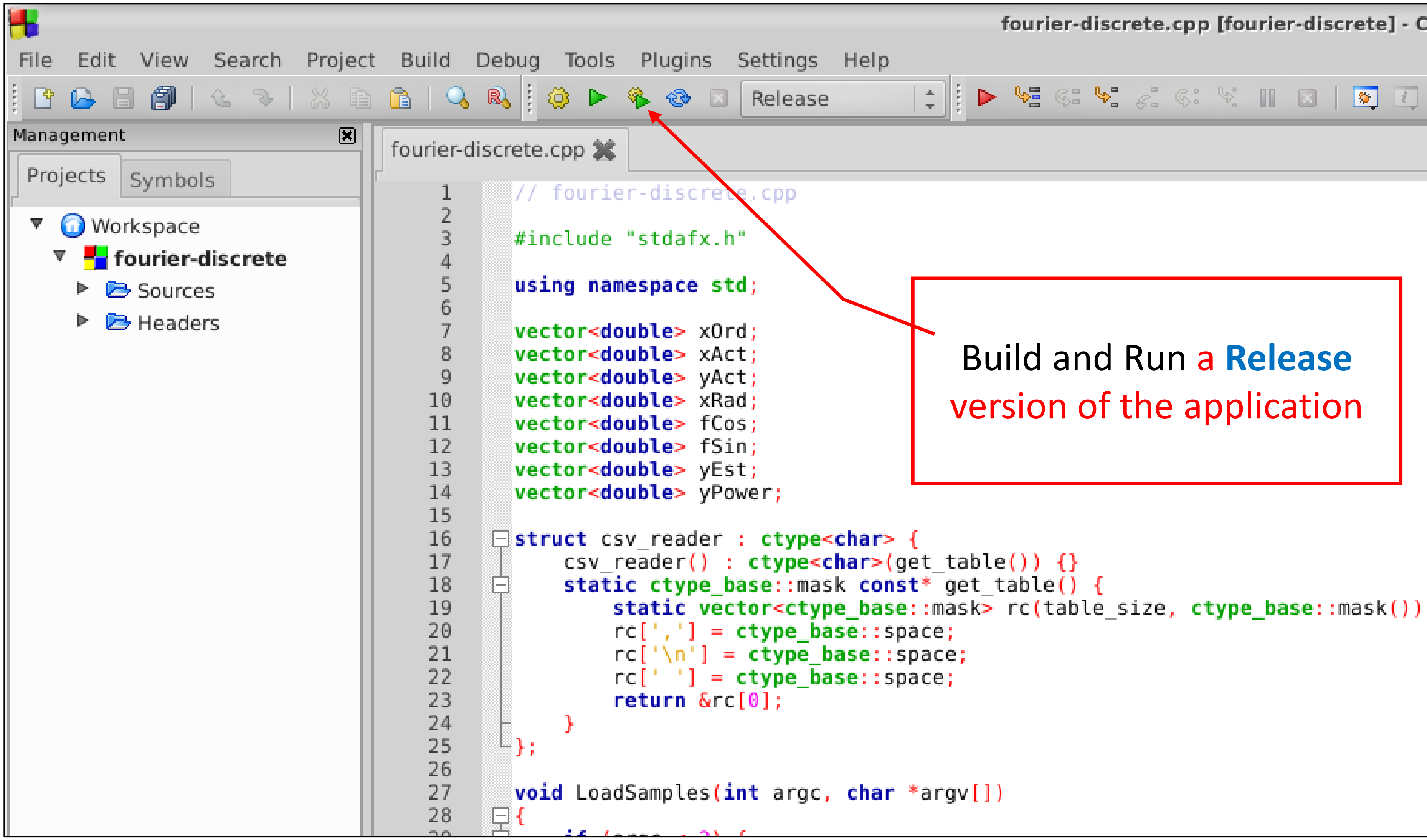
Open Lab 4 – Sunspot Activity



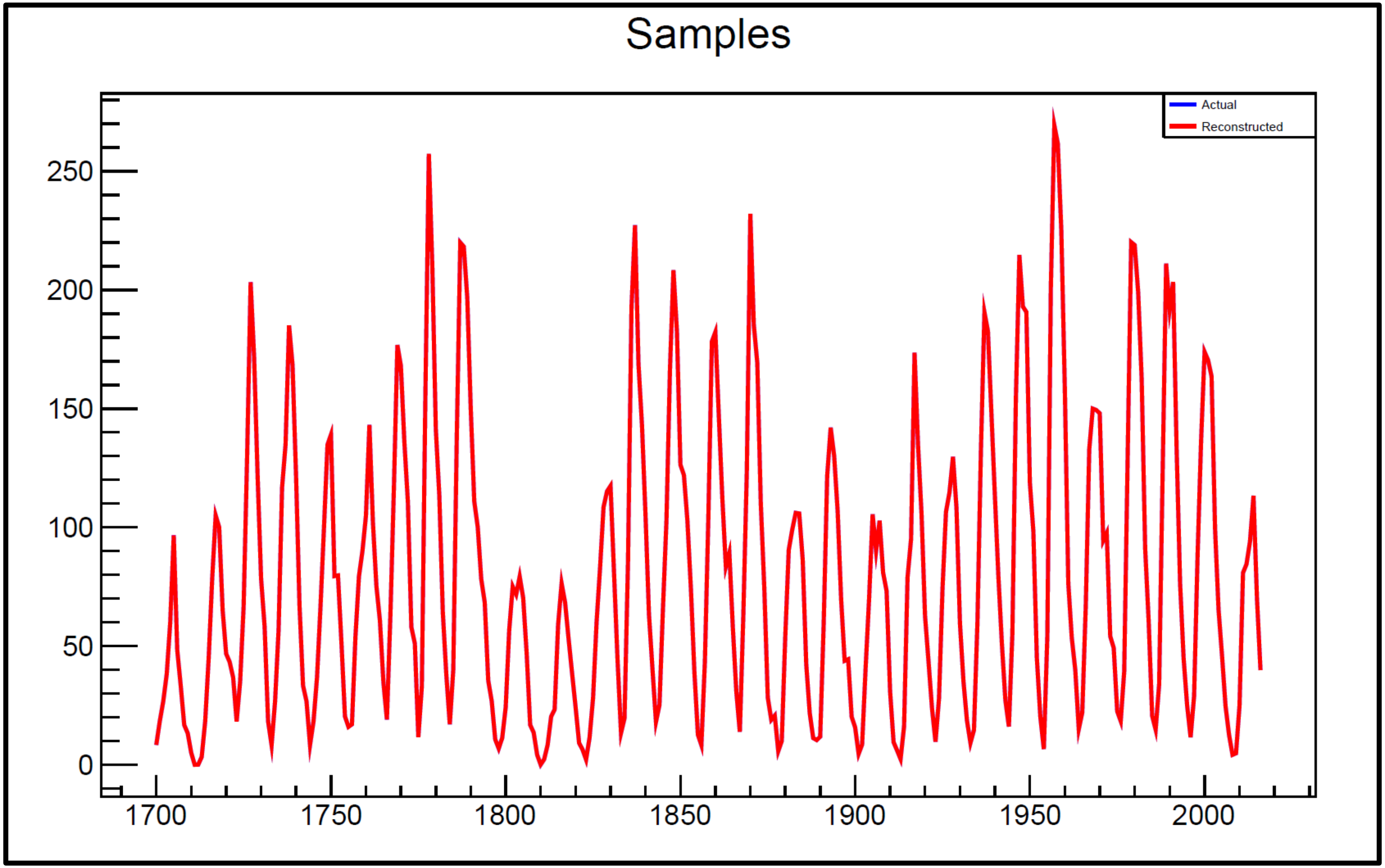
Edit Lab 4 – Sunspot Activity



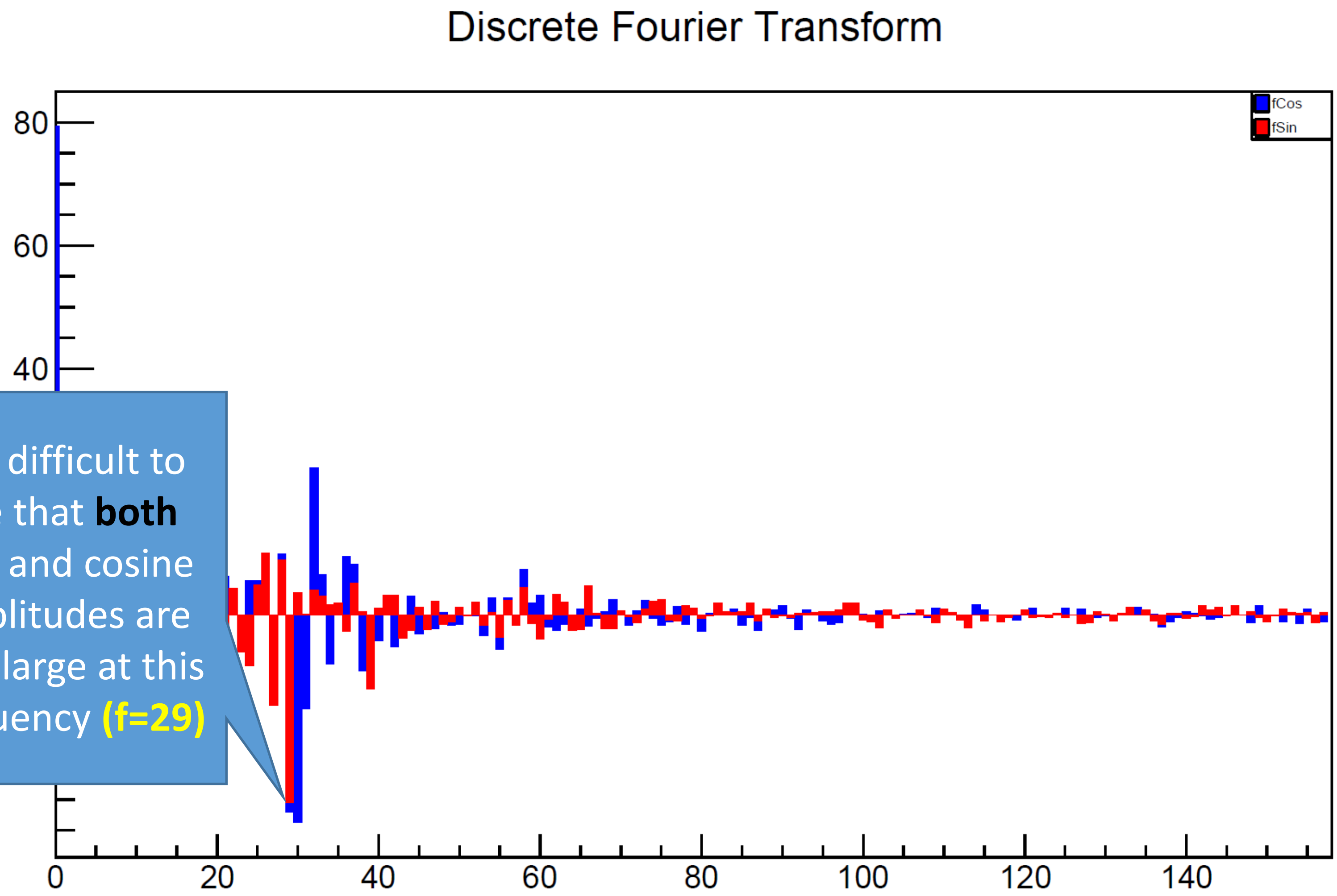
Run Lab 4 – Sunspot Activity



Check Lab 4 – Sunspot Activity



Check Lab 4 – Sunspot Activity

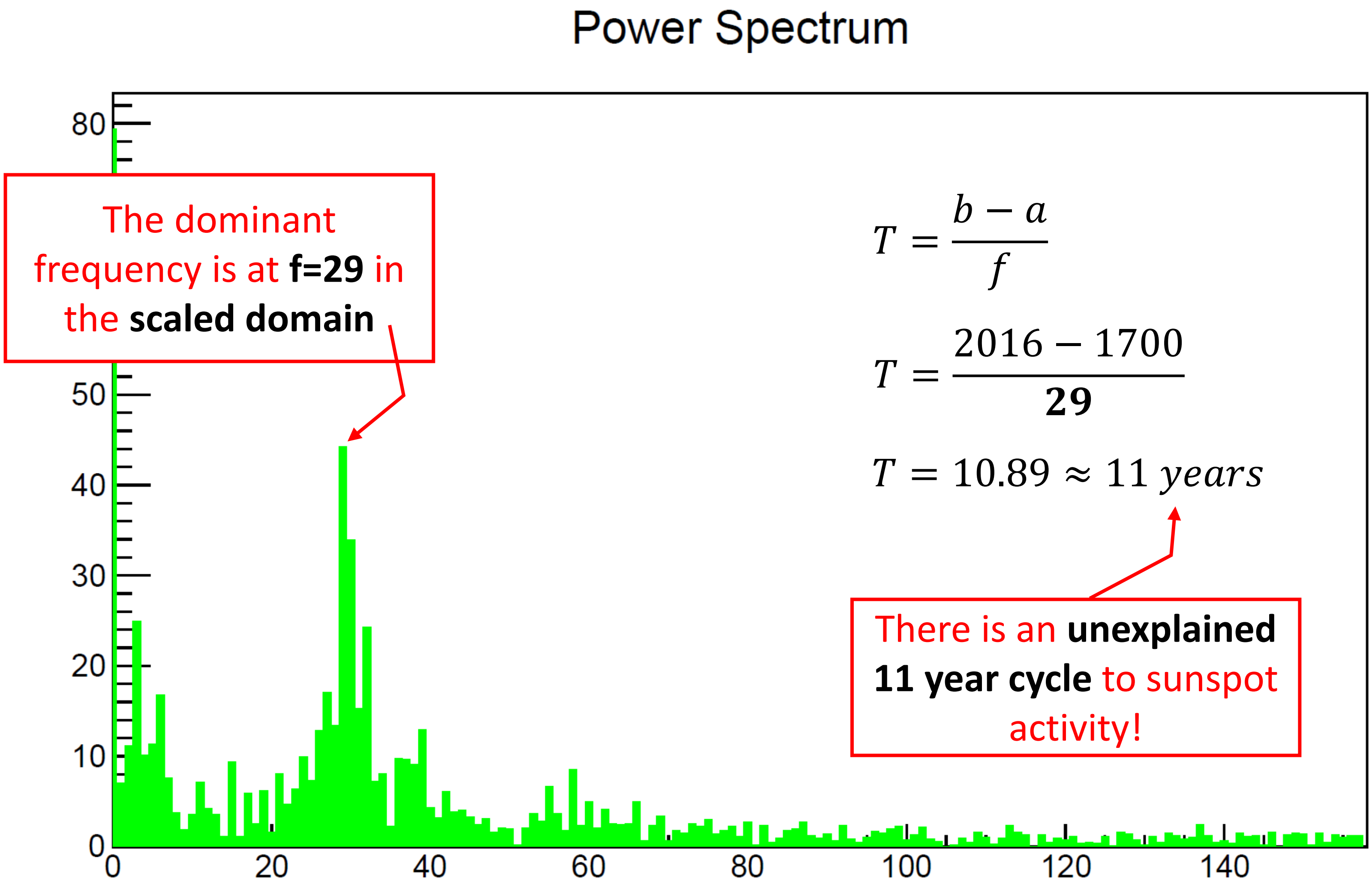


It is difficult to see that **both** sine and cosine amplitudes are very large at this frequency **(f=29)**

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



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

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

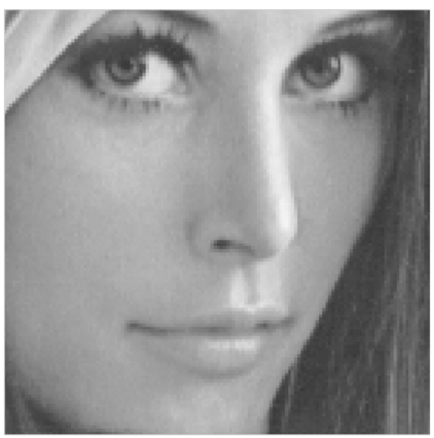
$$T = 10.89 \approx 11 \; years$$

Every JPEG Image uses Fourier Transforms

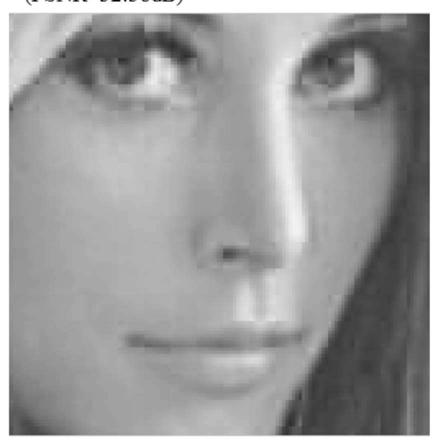
(a) Original Lena image

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

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



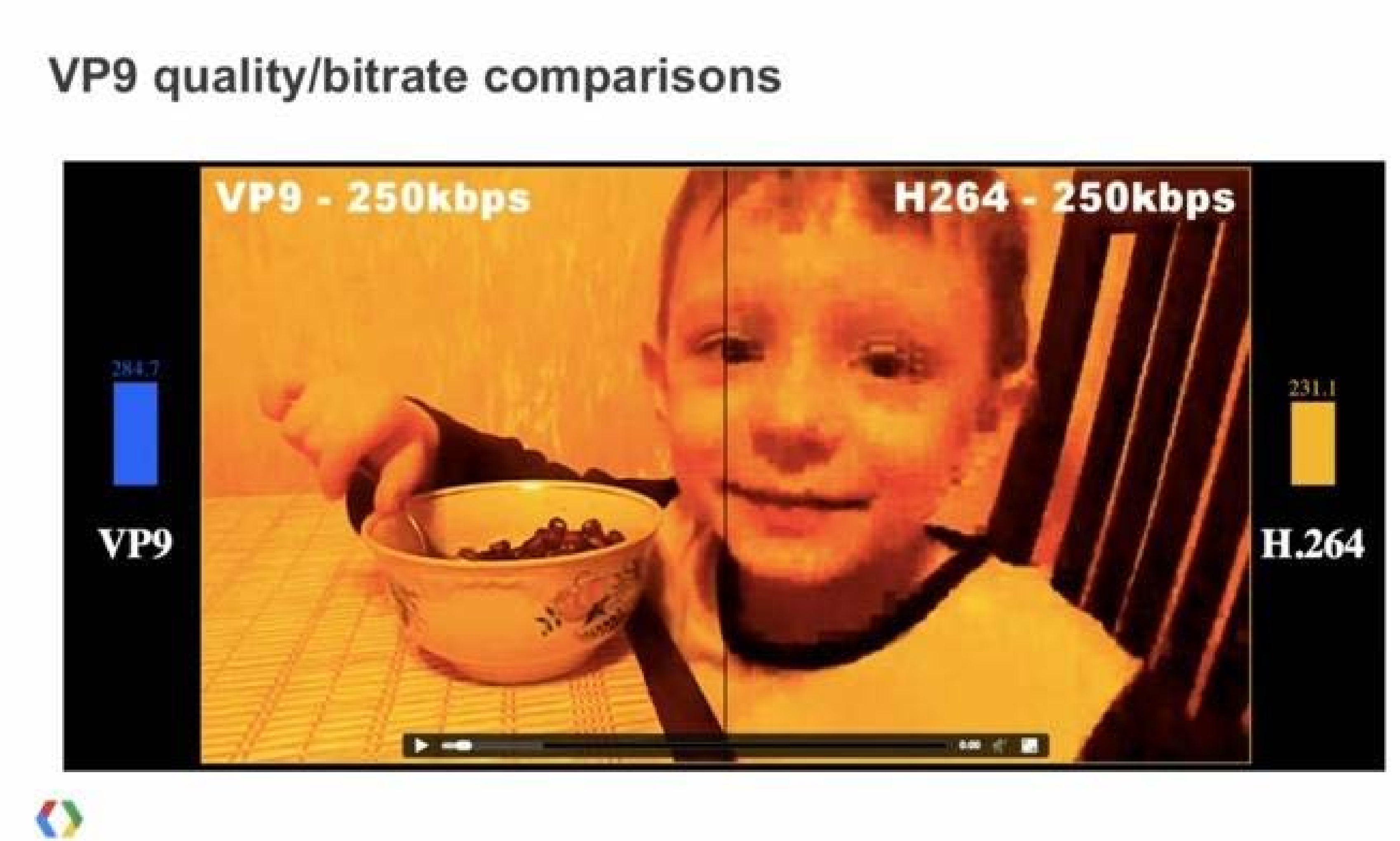
(b) Zoomed original Lena image



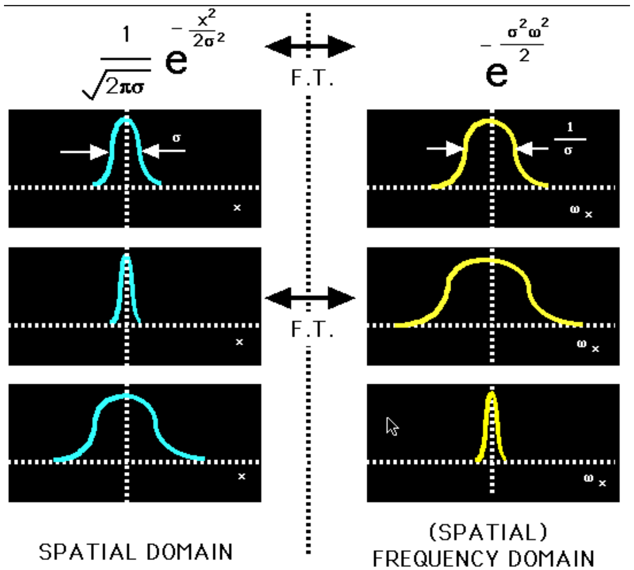
(d) Zoomed DCT-II compressed Lena image

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

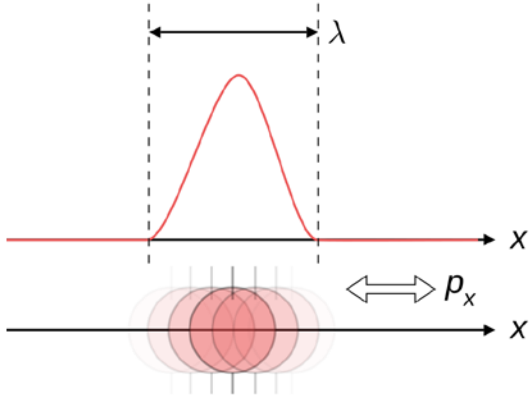
All Streaming Video uses Fourier Transforms

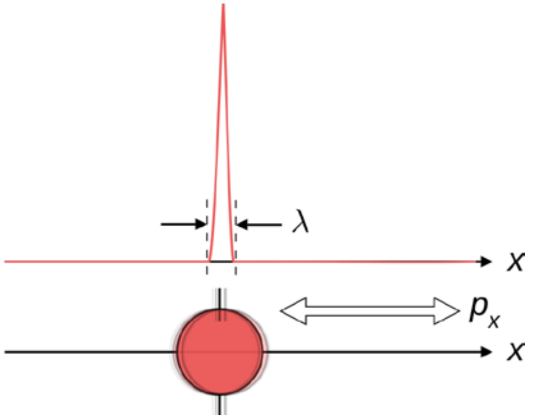


Heisenberg Uncertainty Principle



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





Now you know…

- Fourier Analysis is ***the*** **microscope** of scientific computing
- The Fourier Transform determines the amplitudes and wavelengths of the **constituent simple harmonics** of a complicated waveform
- What appears to be a complicated waveform in the **time domain** might have a simple underlying representation in the **frequency domain**
- Discrete Fourier Transform (DFT) can provide insight in the character of the **physical law** generating the wave