Ch. 7: Fourier Transforms

PHYS 3266/6260 – Prof. Wise

Fourier Transform

- A Fourier transform converts a physical-space (or time series) representation of a function into frequency space
  - Equivalent representation of the function, but gives a new window into its behavior.
  - Inverse operation exists

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x k} dx \qquad f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i x k} dx$$

- You can think of F(k) as being the amount of the function f represented by frequency k

Fourier Transform

- For discrete data, the discrete analog of the Fourier transform gives:
  - Amplitude and phase at discrete frequencies (wavenumbers)
  - Allows for an investigation into the periodicity of the discrete data
  - Allows for filtering in frequency space
  - Can simplify the solution of PDEs: derivatives change into multiplications in frequency space

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x k} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i x k} dx$$

Discrete Fourier Transform

- The discrete Fourier transform (DFT) operates on discrete data
  - Usually we have evenly-spaced, real data.
    - For example, a time-series from an experiment
    - Simulation data for a velocity field
- DFT transforms the N spatial/temporal points into N frequency points. Transform = F_k; Inverse f_n.

$$F_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i n k/N} \qquad f_n = \frac{1}{N} \sum_{n=0}^{N-1} F_k e^{2\pi i n k/N}$$

Notation

$$F_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i n k/N} \quad f_n = \frac{1}{N} \sum_{n=0}^{N-1} F_k e^{2\pi i n k/N}$$

- Many different notations are used in various textbooks and papers.
  - Original function: f(x) or f(t)
  - Transformed function: $F(k), \mathcal{F}(k), \hat{f}(n)$
- For the discrete version:
  - Original function: f_n
  - Transformed function: $F_k, \mathcal{F}_k, \hat{f}_k$

Real space vs. frequency space

- What are the significance of the real and imaginary parts?
  - Recall that we are integrating with $e^{-2\pi ikx}$
  - Euler's formula: $e^{ix} = \cos x + i \sin x$
  - Real part represents the cosine terms, symmetric functions
  - Imaginary part represents the sine terms, asymmetric functions
  - Can also think in terms of an amplitude and a phase
  - If f_n is real, then

$$\mathrm{Re}(F_k) = \sum_{n=0}^{N-1} f_n \cos\left(\frac{2\pi nk}{N}\right) \qquad \mathrm{Im}(F_k) = \sum_{n=0}^{N-1} f_n \sin\left(\frac{2\pi nk}{N}\right)$$

DFT Example

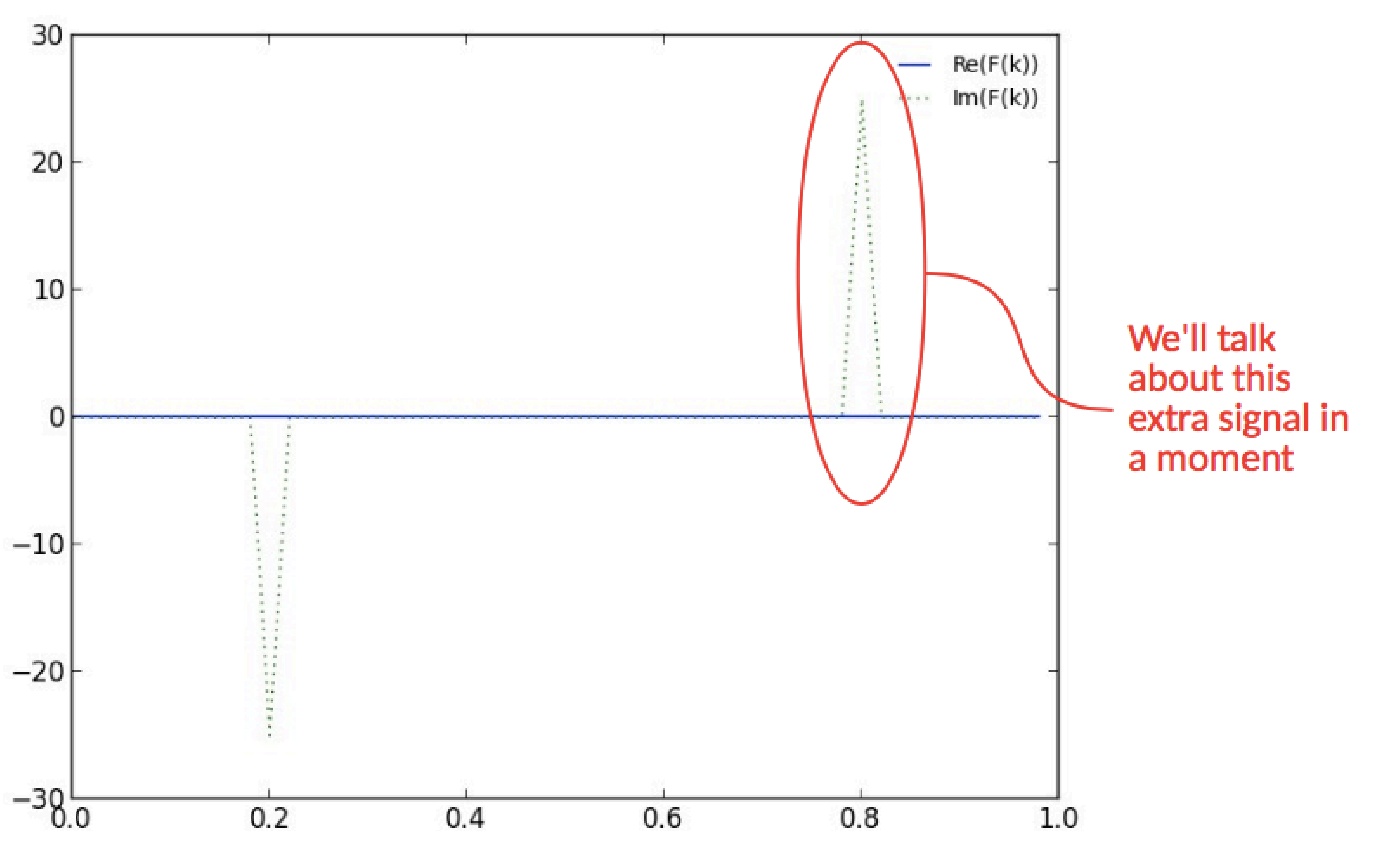
$$F_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i n k/N}$$

- Implementing the discrete Fourier transform is straightforward
  - Double sum: for each wavenumber, we sum over all the spatial points

```python
def dft(f_n):
    N = len(f_n)
    f_k = np.zeros(N, dtype=np.complex128)
    for k in range(N):
        for n in range(N):
            f_k += f_n[n] * np.exp(-2*np.pi*1j*n*k/N)
    return f_k
```

DFT Example

- DFT of $\sin(2\pi f_0 x)$ with $f_0 = 0.2$



Frequencies

- Note that in the DFT, nowhere does the physical coordinate value x_n, enter — instead, we just look at the index n itself
    - This assumes that the data is regular gridded
- In this index space, the smallest wavelength is from one cell to the next, and the smallest frequency is 1/N
    - Note that this means that if we add points to our data, then we open up higher and higher frequencies (in terms of index space)

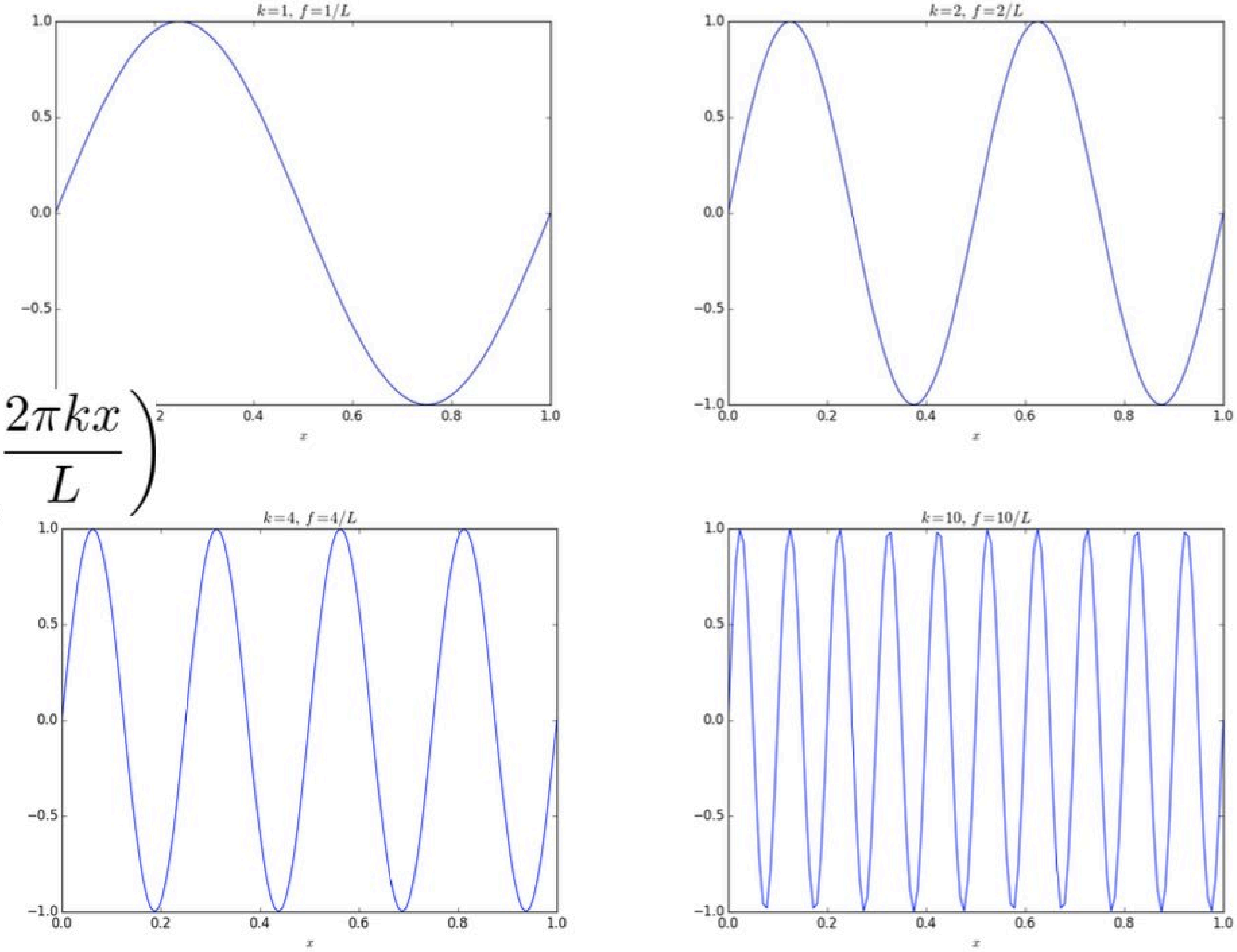
Frequencies

- Clearly there is a physical scale for the frequency

$$\nu_k = \frac{k}{N} \cdot \frac{1}{\Delta x} = \frac{k}{N} \cdot \frac{N}{L}$$

- Lowest frequency: $1/L$
- Highest frequency: $\sim 1/\Delta x$

$$\sin(2\pi f x) = \sin\left(\frac{2\pi k x}{L}\right)$$



Frequencies

- k = 0 is special:

$$\mathrm{Re}(F_0) = \sum_{n=0}^{N-1} f_n \cos\left(\frac{2\pi n 0}{N}\right) = \sum_{n=0}^{N-1} f_n$$

$$\mathrm{Im}(F_0) = \sum_{n=0}^{N-1} f_n \sin\left(\frac{2\pi n 0}{N}\right) = 0$$

- This is sometimes called the **DC offset**

FFT = DFT

- The Fast Fourier Transfer (FFT) is equivalent to the discrete Fourier transform
    - Faster because of special symmetries exploited in performing sums
    - The calculation expense scale as (N log N) instead of (N^2)
- We won't investigate how FFTs work, but there are details in the book

Normalization

- Normalization of the forward and inverse transforms follows from Parseval's theorem:

$$\int_{-\infty}^{\infty} |f(x)|^2 dx = \int_{-\infty}^{\infty} |F(k)|^2 dk$$

  - Discrete form (taking Δx = 1 and Δk = 1/N)

$$\sum_{n=0}^{N-1} |f_n|^2 = \frac{1}{N} \sum_{k=1}^{N-1} |F(k)|^2$$

- The definition of the inverse transformation compatible with this condition is

$$f_n = \frac{1}{N} \sum_{k=0}^{N-1} F_k e^{2\pi i n k/N}$$

Normalization

- To illustrate this, consider the transform of 1
  - Analytically: f(x) = 1 ⟺ F(k) = δ(k)
  - Discrete transform:

$$F_k = \sum_{n=0}^{N-1} 1 \cdot e^{-2\pi i k n/N} = \sum_{n=0}^{N-1} [\cos(2\pi kn/N) + i\sin(2\pi kn/N)]$$

  - For k = 0, cos(0) = 1, sin(0) = 0, so F_0 = N
  - For k > 0, we are essentially doing a discrete sum of the cosine and sine using equally-spaced points, and the sum is always over a multiple of a full wavelength, therefore F_K = 0.

Normalization

- Discrete inverse transform:

$$f_n = \frac{1}{N}\sum_{k=0}^{N-1} F_k e^{2\pi i n k/N} = \frac{1}{N} N e^{2\pi i n 0/N} = 1$$

- When plotting, we want to put the physical scale back in, as well as make the correspondence to the continuous representation
  - Already saw how to put the wave numbers into a physical frequency
  - Rewrite inverse equation

$$f_n = \frac{1}{N}\sum_{k=0}^{N-1} F_k e^{2\pi i n k/N} = \sum_{k=0}^{N-1} \left(\frac{F_k}{N}\right) e^{2\pi i n k/N}$$

Plot this

  - Another interpretation: $F_k/N \Longleftrightarrow F_k\, dk$

Real space vs. frequency space

- Imagine transforming a real-valued function f with N data points
  - The FFT of f will have N/2 complex data points
  - Same amount of information in each case, but each complex point corresponds to 2 real numbers
  - This is a consequence of the analytic Fourier transform satisfying
    - $F(-k) = F^*(k)$ if $f(x)$ is real

Real space vs. Frequency space

- Most FFT routines will return N complex points — half of them are duplicate, that is, they don't add to the information content
    - Often, there are specific implementations optimized for the case where the function is real (e.g. rfft)
- This affects normalization (note k=0 different)
- This is also referred to as aliasing.
- The maximum frequency, $1/(2\Delta x)$, is called the **Nyquist frequency.**

Power spectrum

- The power spectrum is simply defined as:

$$P(k) = |F(k)|^2 = F(k)F^{\star}(k)$$

- It is a single number showing the importance or weight of a given wavenumber

- Also useful to look at the phase at each wavenumber

Python / Numpy's FFT

- numpy.fft: https://docs.scipy.org/doc/numpy/reference/routines.fft.html
- fft/ifft: 1D data
  - By design, the k=0,…,N/2 data is first, followed by the negative frequencies. The latter are not relevant for a real-valued f(x)
  - k-values can be obtained from fftfreq(n)
  - fftshift(x) shifts the k=0 to the center of the spectrum
- rfft/irfft: 1D real-valued functions. Basically the same as fft/ifft but doesn't return negative frequencies
- 2D and N-D routines are analogously defined

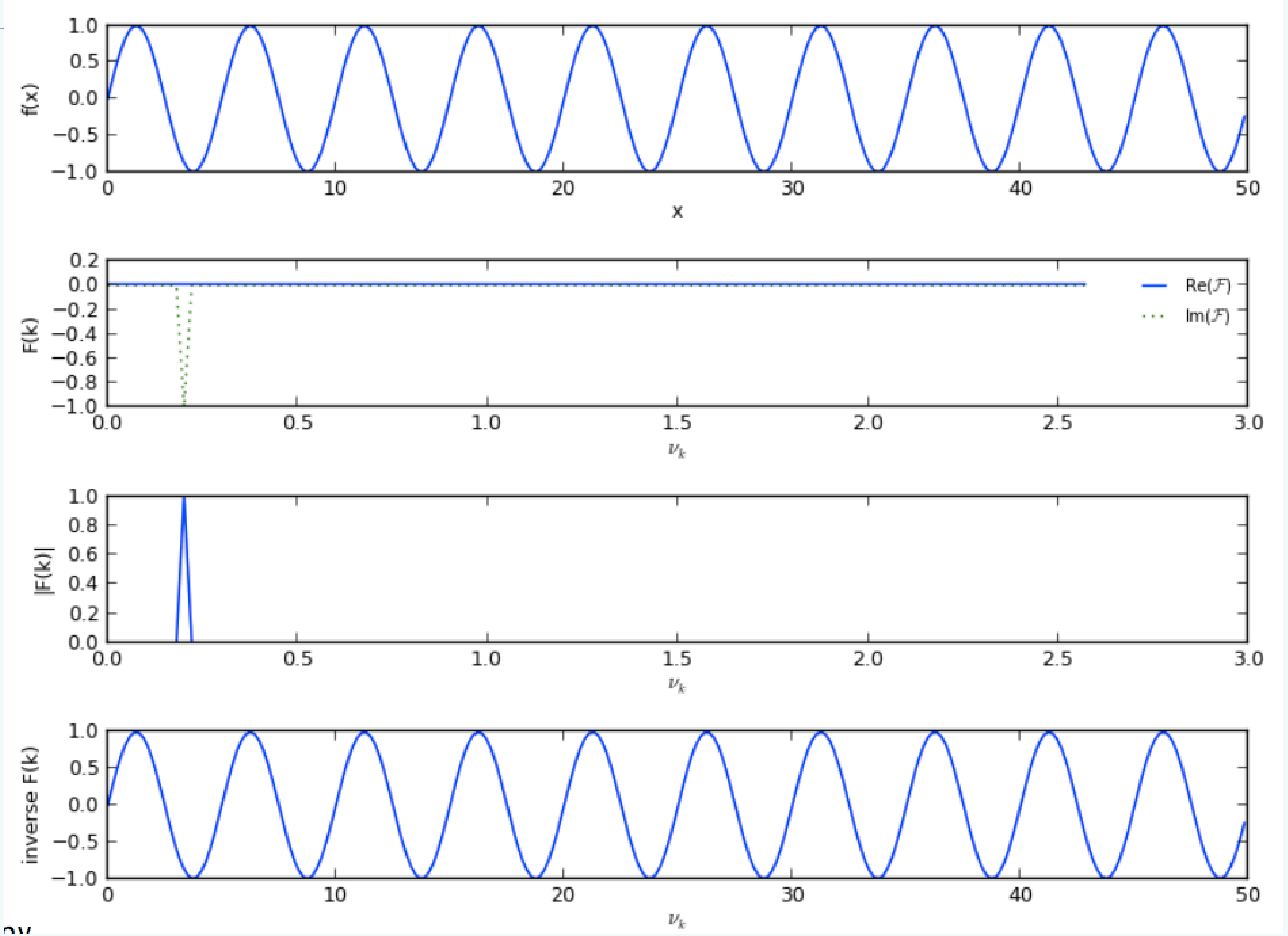
Python's FFT

- It's always a good idea to run some simple tests to make sure the FFT is behaving the way you'd expect
  - $\sin(2\pi f_0 x)$ — should be purely imaginary at a single wavenumber
  - $\cos(2\pi f_0 x)$ — should be purely real at a single wavenumber
  - $\sin(2\pi f_0 x + \pi/4)$ — should have equal magnitude real and imaginary parts at a single wavenumber

Example: Single Frequency Sine Function

- Transform a single mode sine ($f_0 = 0.2$) and inverse transform back — do you get the original result (to roundoff errors)?

- Notice that the since sine is odd, the FFT is only non-zero in the imaginary component.

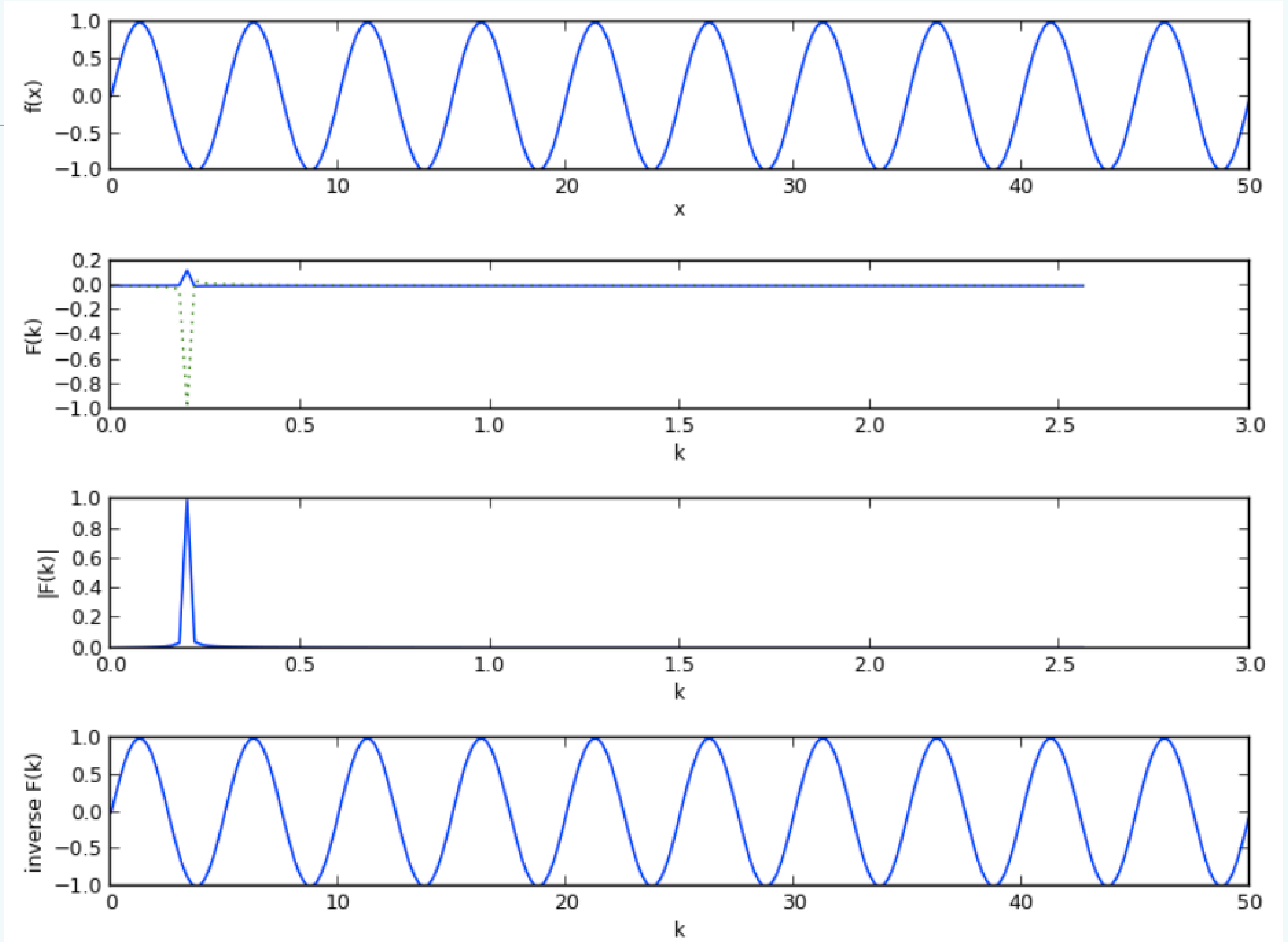


Sample points

- Be careful about how you define the points
  - The FFT considers the data without any spatial coordinates — it just considers the distance in terms of the number of points
  - Using the Numpy linspace() routine puts a point at both the start and end of the interval
    - e.g. np.linspace(0, 1, 5) = [0. 0.25 0.5 0.75 1.]
    - The FFT routine treats the first and last point as distinct
    - If you define sin(2πx) on these data, the first and last points will be equivalent, and the FFT picks up an extra (non-periodic) signal
    - Instead, do np.linspace(0, 1, 5 endpoint=False)

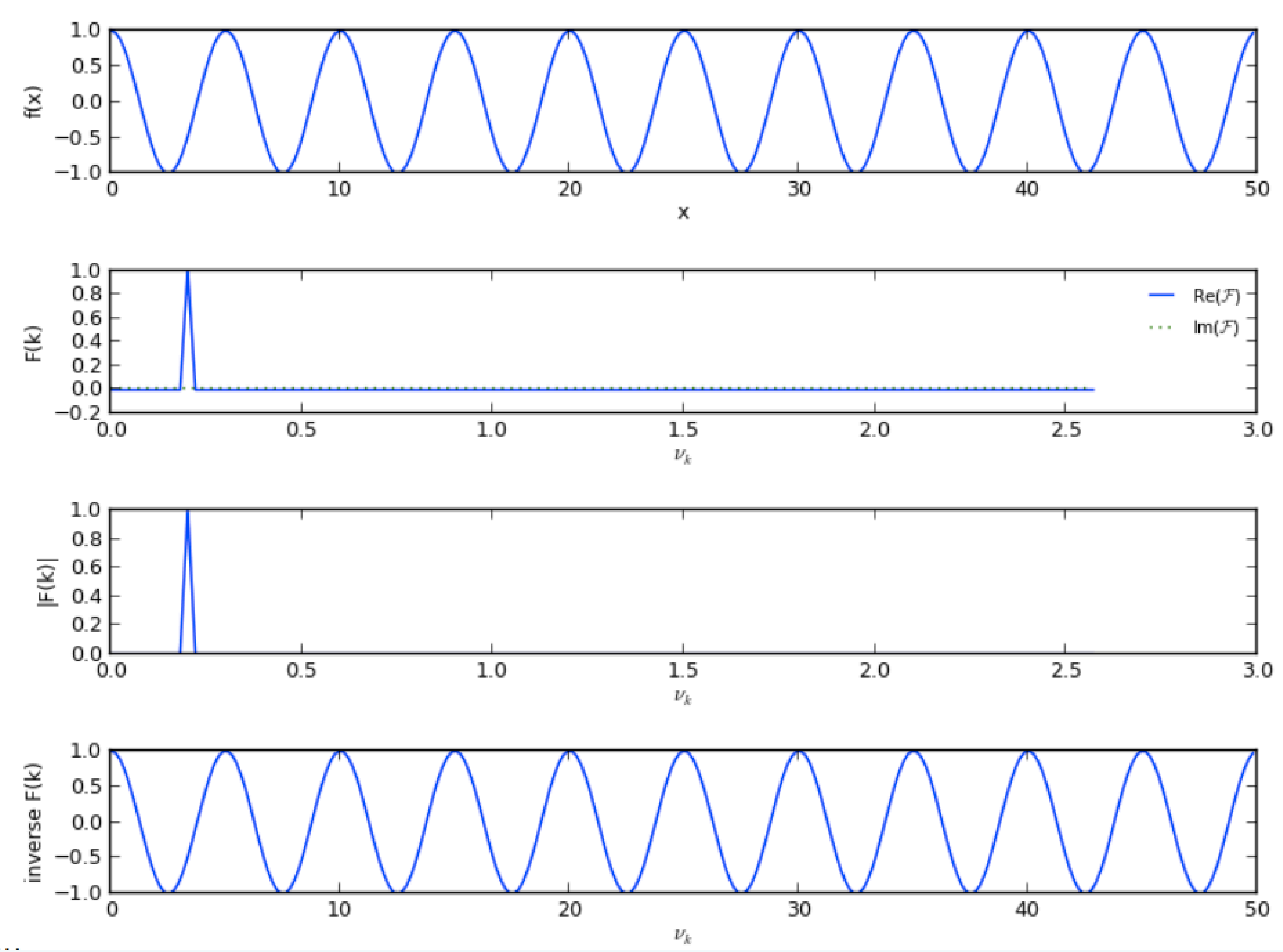
Sample points (periodicity)

- FFT with np.linspace, putting a point at both endpoints.
- Note the blip in the real portion of the FFT.



Example: Single Frequency Cosine

- "Opposite" of the sine
- Notice that the since cosine is even, the FFT is only non-zero in the real component.
- Also note the normalization (+ instead of –)

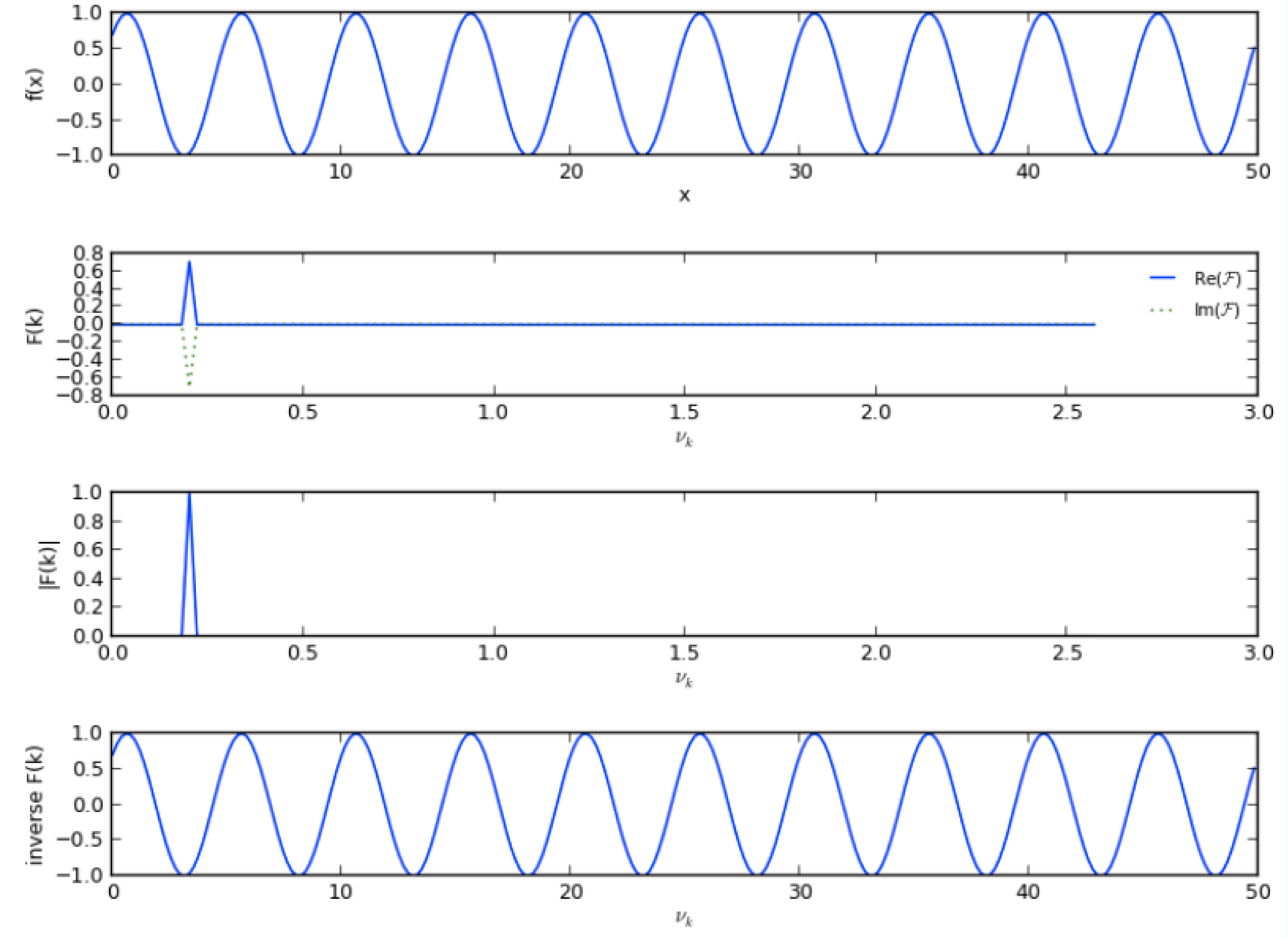


Example: Phase Shift

- Amplitude and phase. Consider: $\sin(2\pi f_0 x + \pi/4)$

- Phase

$$\phi = \tan^{-1}\left[\frac{\mathrm{Im}(F_k)}{\mathrm{Re}(F_k)}\right]$$



Example: Filtering

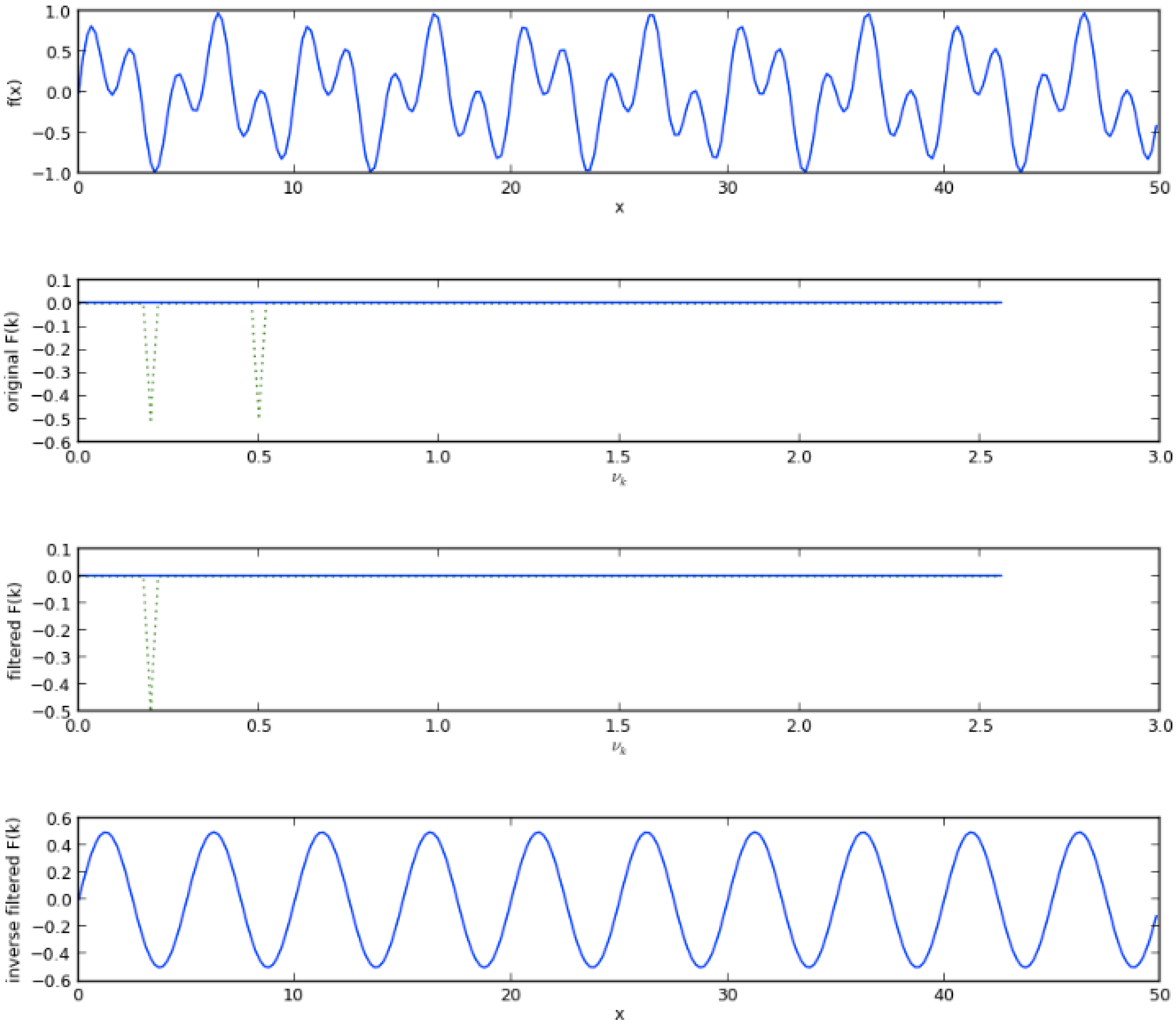
- In frequency-space, we can filter out high or low frequency components.
- Consider:

$$f(x) = \frac{1}{2}\left[\sin(2\pi f_0 x) + \sin(2\pi f_1 x)\right]$$

  - with $f_0 = 0.2$ and $f_1 = 0.5$

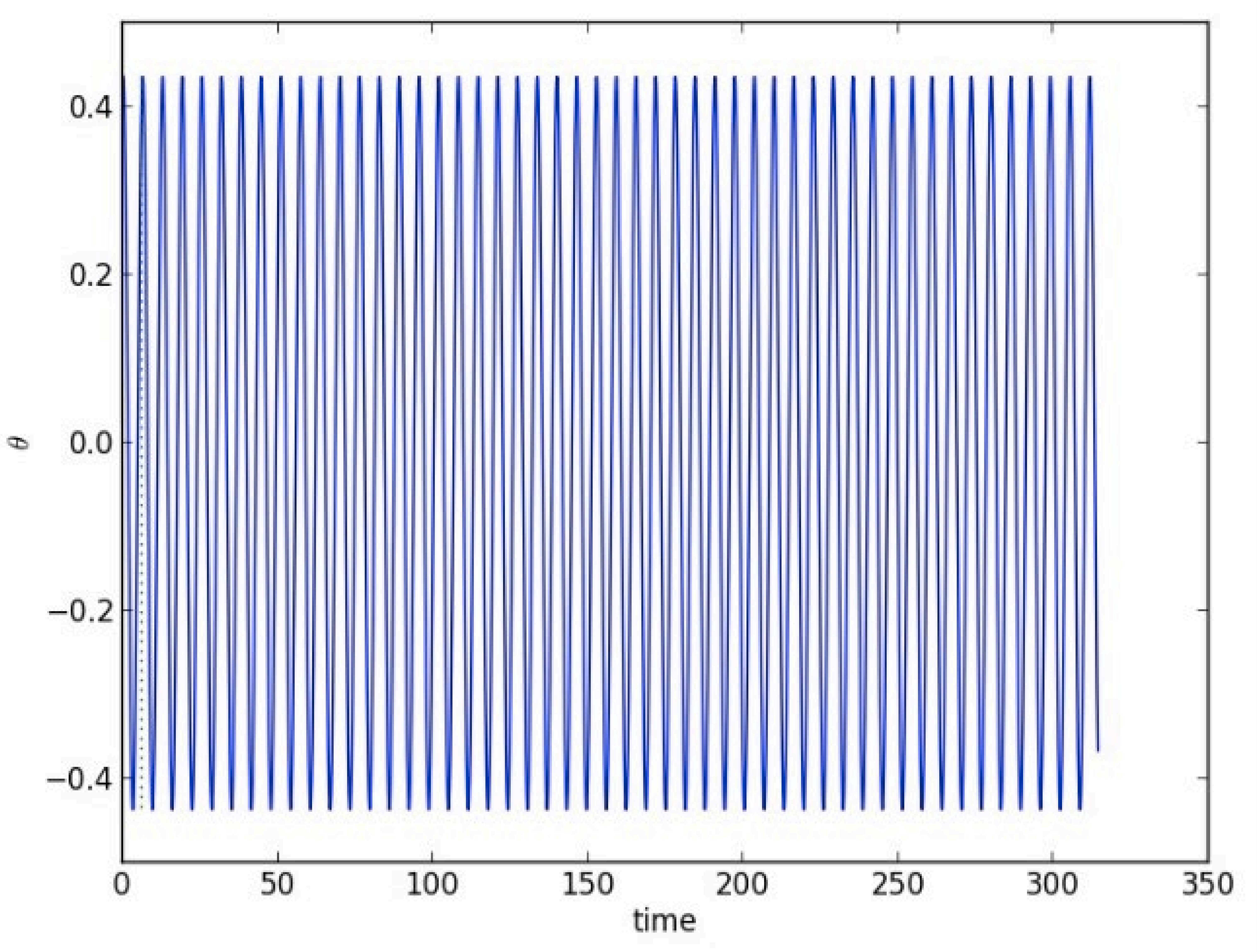
Example: Filtering

- Remove the higher frequency component from F(k)
- Take the inverse FFT to recover the filtered data



Example: Time-series analysis

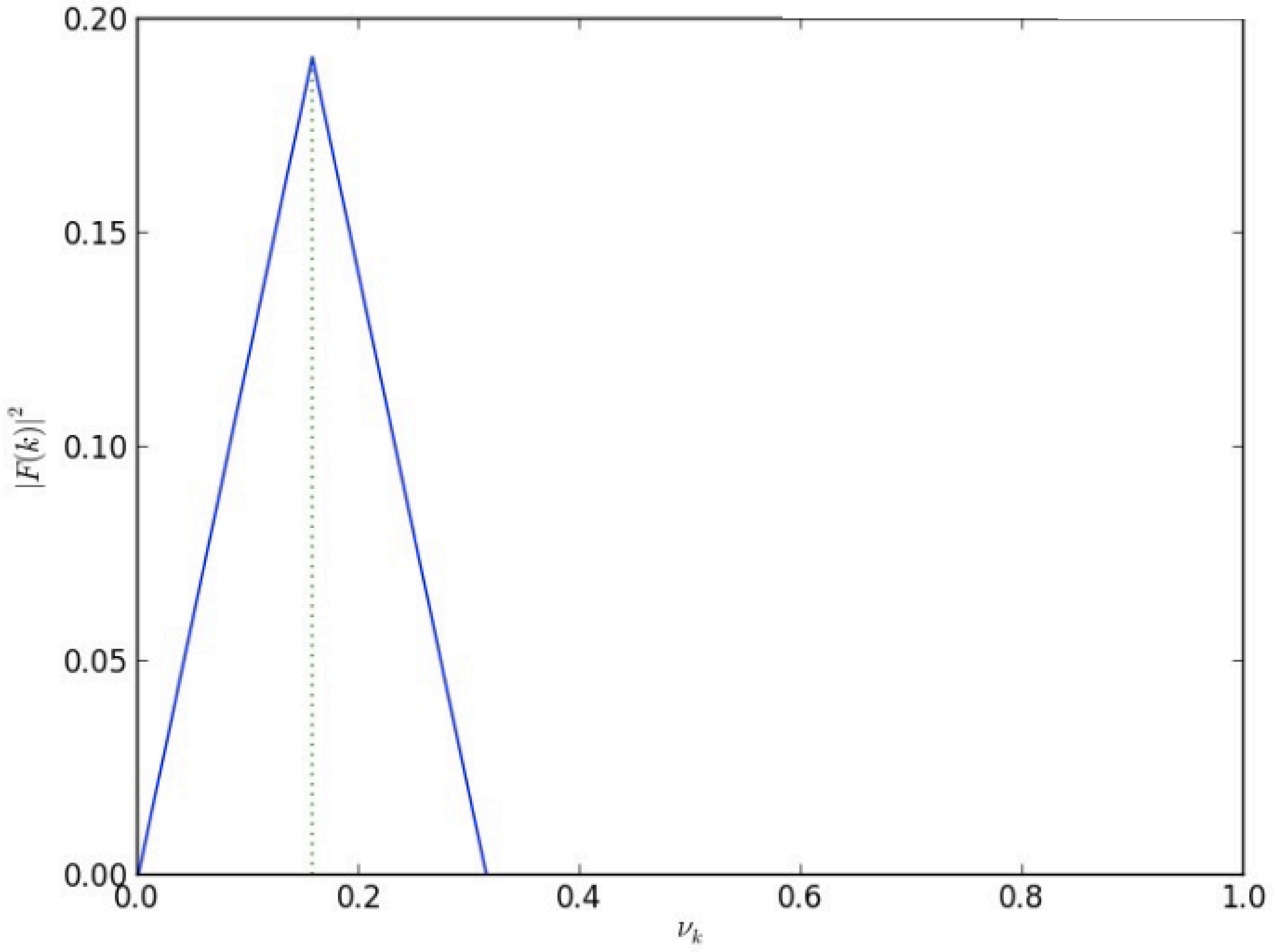
- Consider a simple pendulum
- Finite-amplitude system
- The following solution is a numerical one
- 50 periods are integrated
- Take the initial angle $\theta_0 = 25^\circ$



Example: Time-series analysis

- Let's look at the power spectrum, $|F(k)|^2$, of the first period
- Dotted line is the analytical frequency estimate:

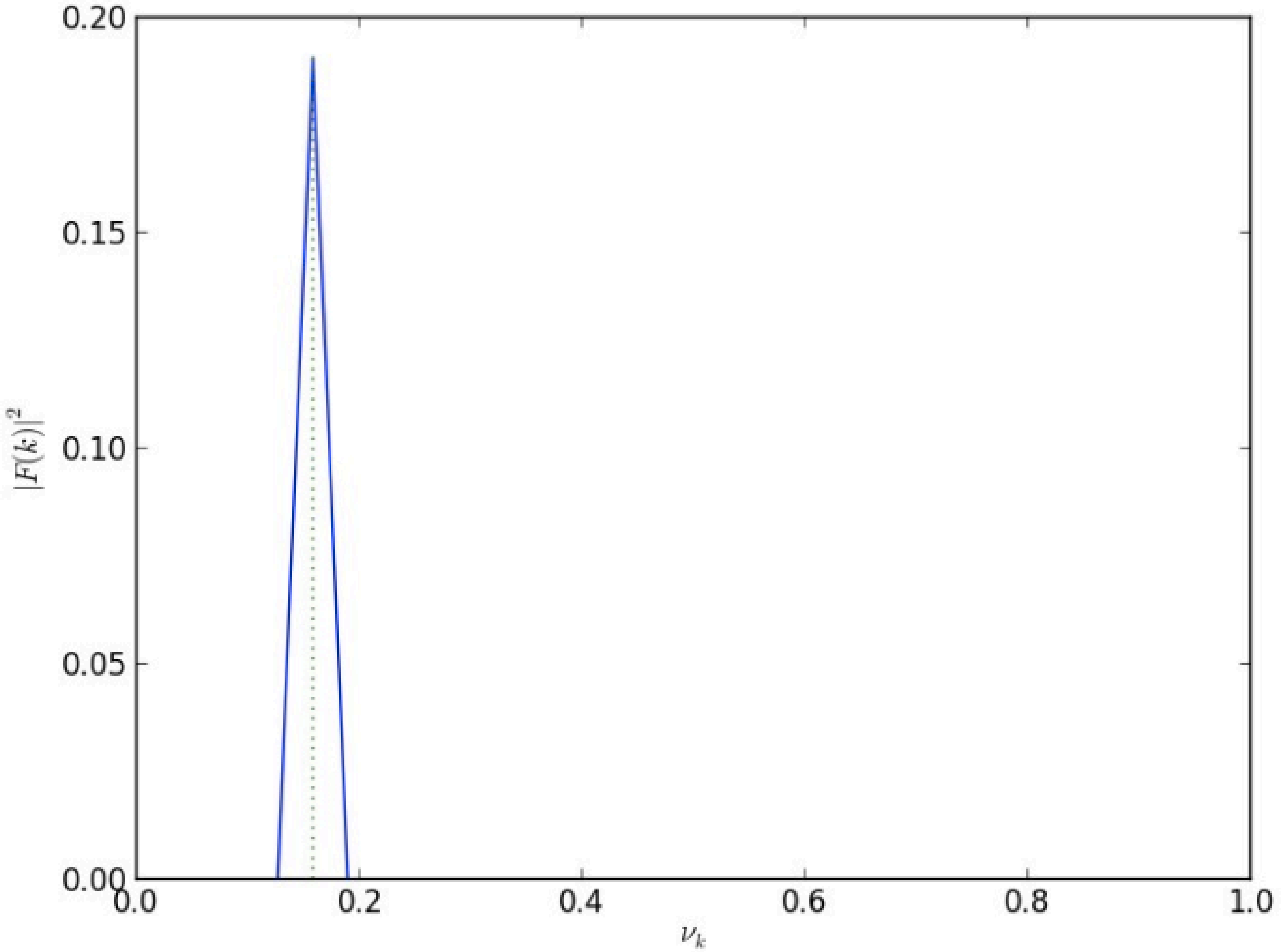
$$f^{-1} = T = 2\pi\sqrt{\frac{L}{g}}\left(1 + \frac{\theta_0}{16}\right)$$



Example: Time-series analysis

- First 5 periods
- Dotted line is the analytical frequency estimate:

$$f^{-1} = T = 2\pi\sqrt{\frac{L}{g}}\left(1 + \frac{\theta_0}{16}\right)$$

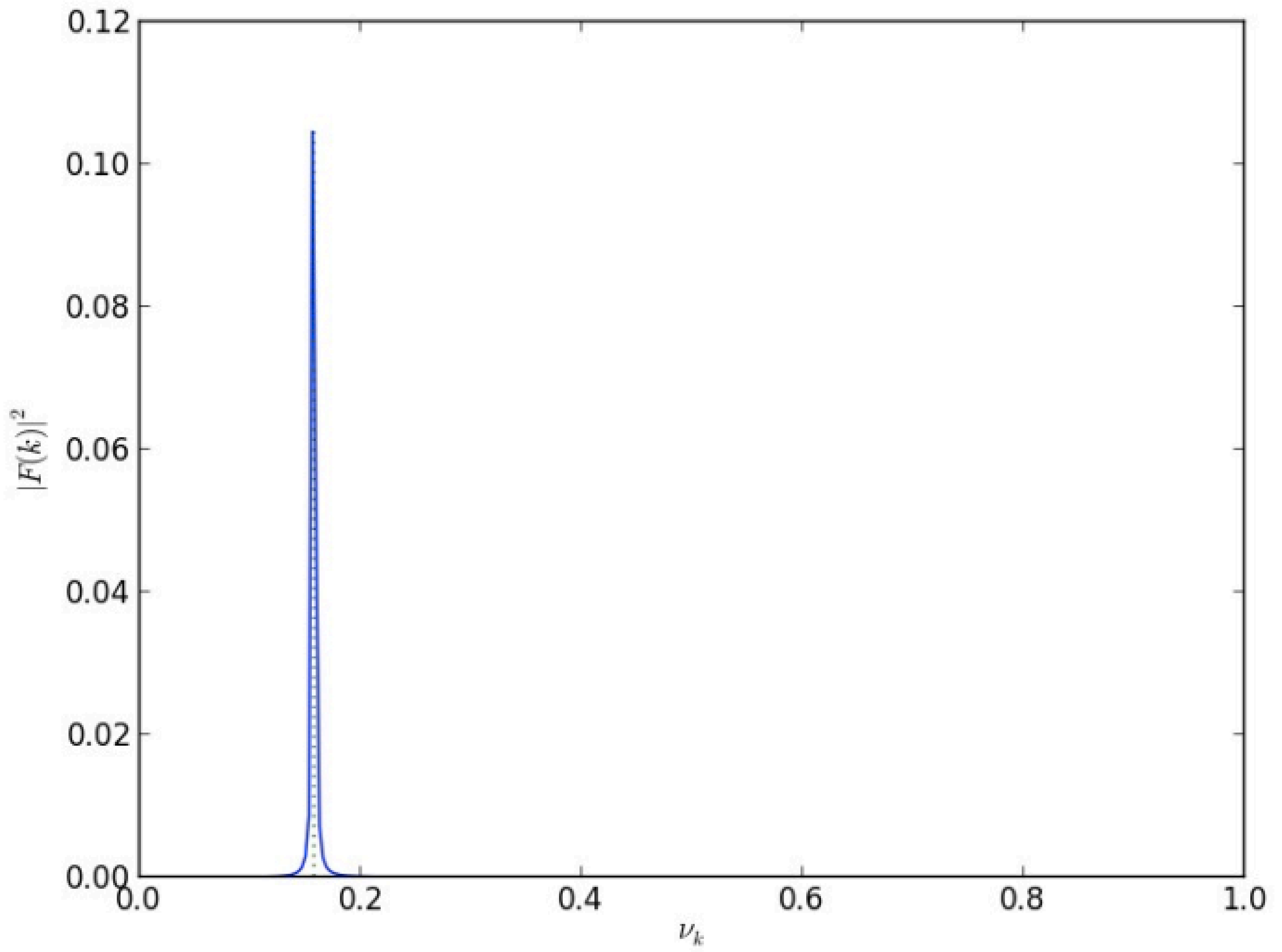


Example: Time-series analysis

- All 50 periods
- Dotted line is the analytical frequency estimate:

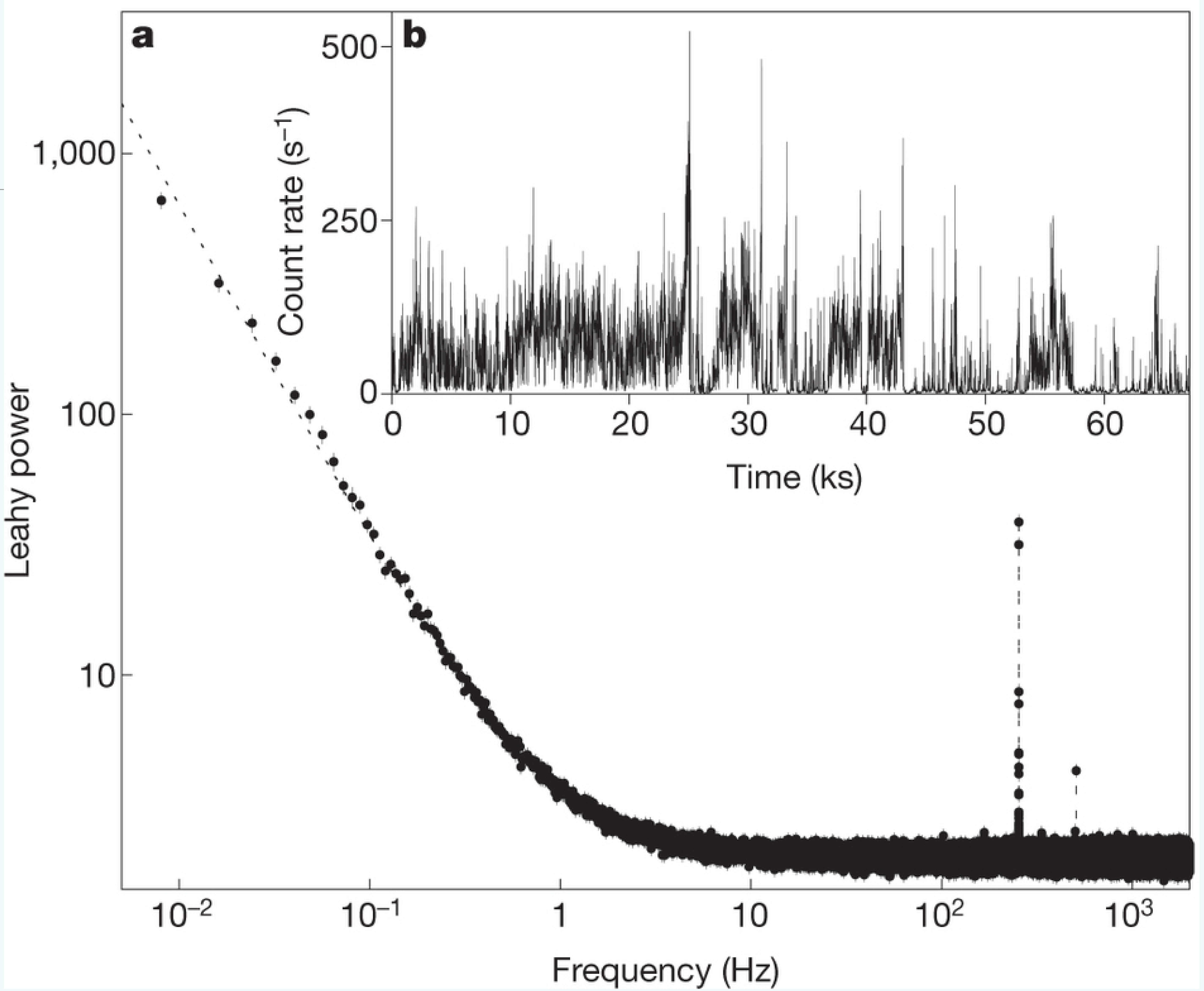
$$f^{-1} = T = 2\pi\sqrt{\frac{L}{g}}\left(1 + \frac{\theta_0}{16}\right)$$

- Smoothing at the base is from numerical error



Example: Time-series analysis

- Astrophysical example from an X-ray emitting pulsar (spinning neutron star) time-series
- [Credit]()



Multi-dimensional FFTs

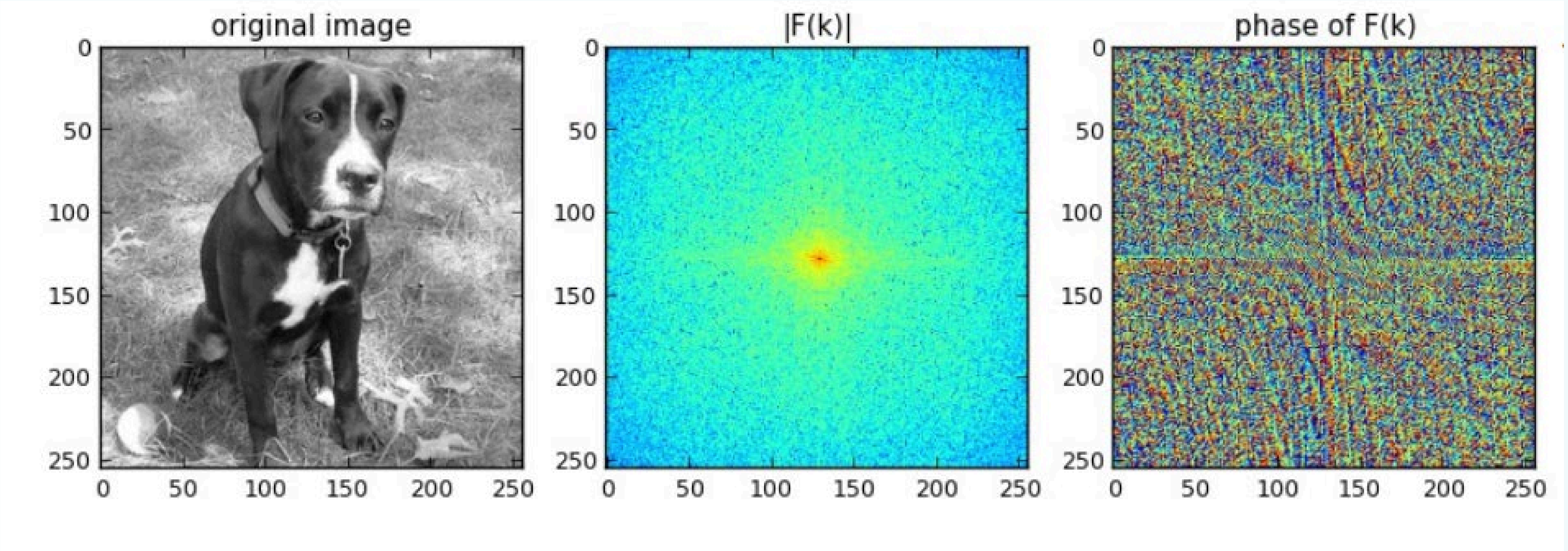
- Multi-dimensional FFTs decompose into a sequence of one-dimensional FFTs

$$
\begin{aligned}
F_{k_x,k_y} &= \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} f_{mn} e^{-2\pi i (k_x m/N_x + k_y n/N_y)} \\
&= \sum_{m=0}^{N_x-1} e^{-2\pi i k_x m/N_x} \boxed{\sum_{n=0}^{N_y-1} f_{mn} e^{-2\pi i k_y n/N_y}}
\end{aligned}
$$

Transform in the y-direction

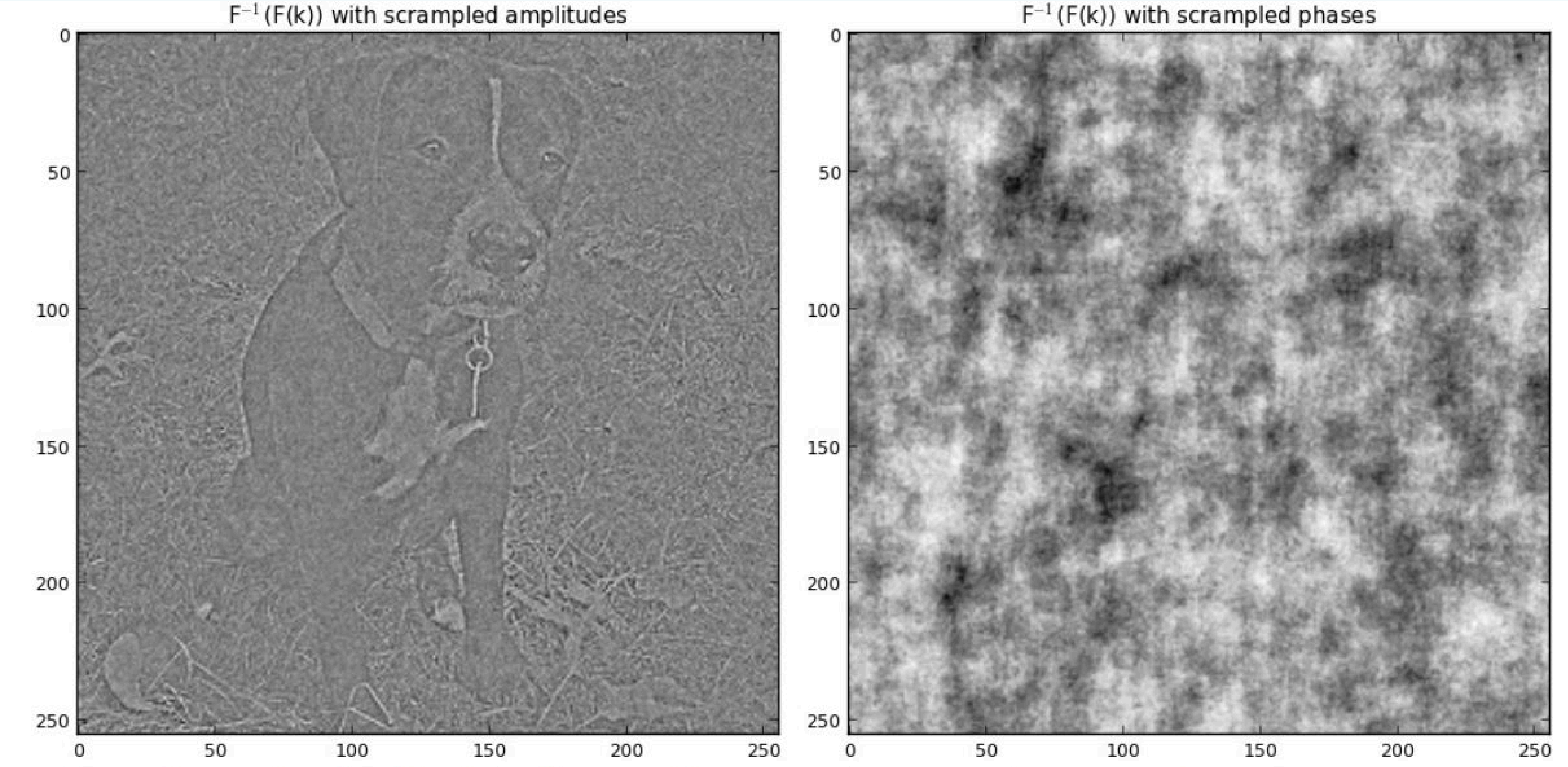
Multi-dimensional FFTs

- To visualize what's going on, we need to look at both the amplitude and the phase
  - Note that only 1 quadrant is significant because our input was real.



Multi-dimensional FFTs

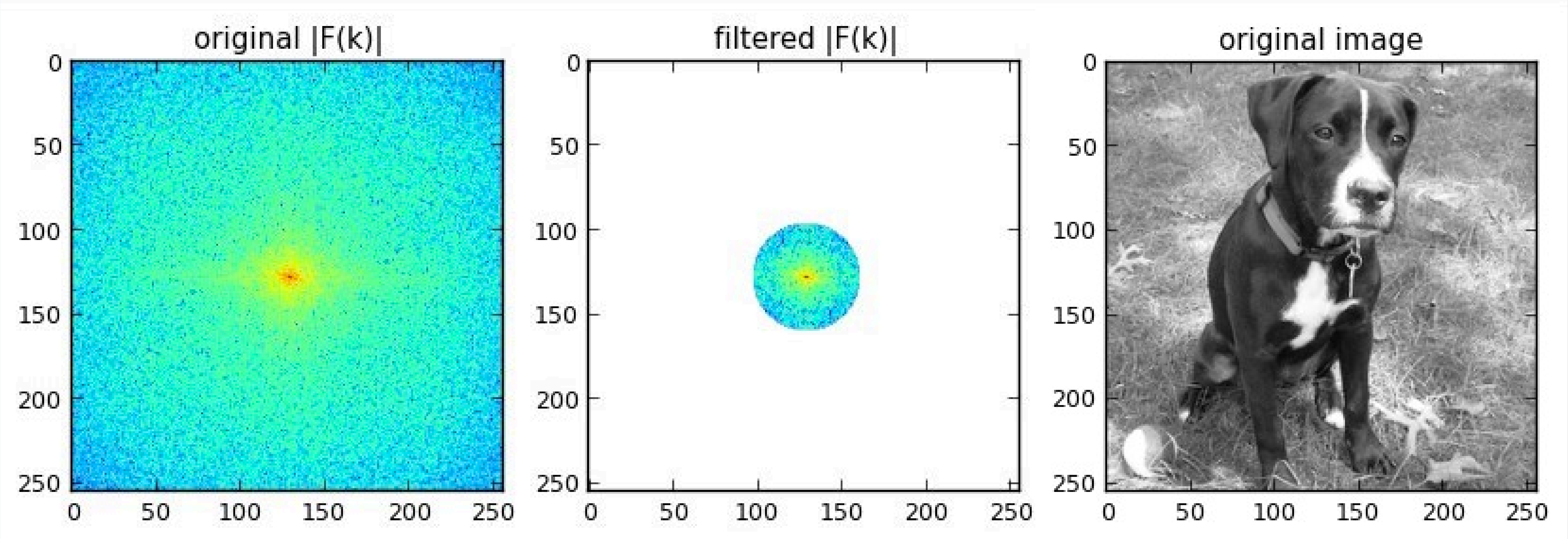
- To understand what the magnitude and phase influence, here we scrape each of them in turn and take inverse of it.



Example based on :http://matlabgeeks.com/tips-tutorials/how-to-do-a-2-d-fourier-transform-in-matlab/

Multi-dimensional FFTs

- In phase space, we can filter out frequency components to do image processing

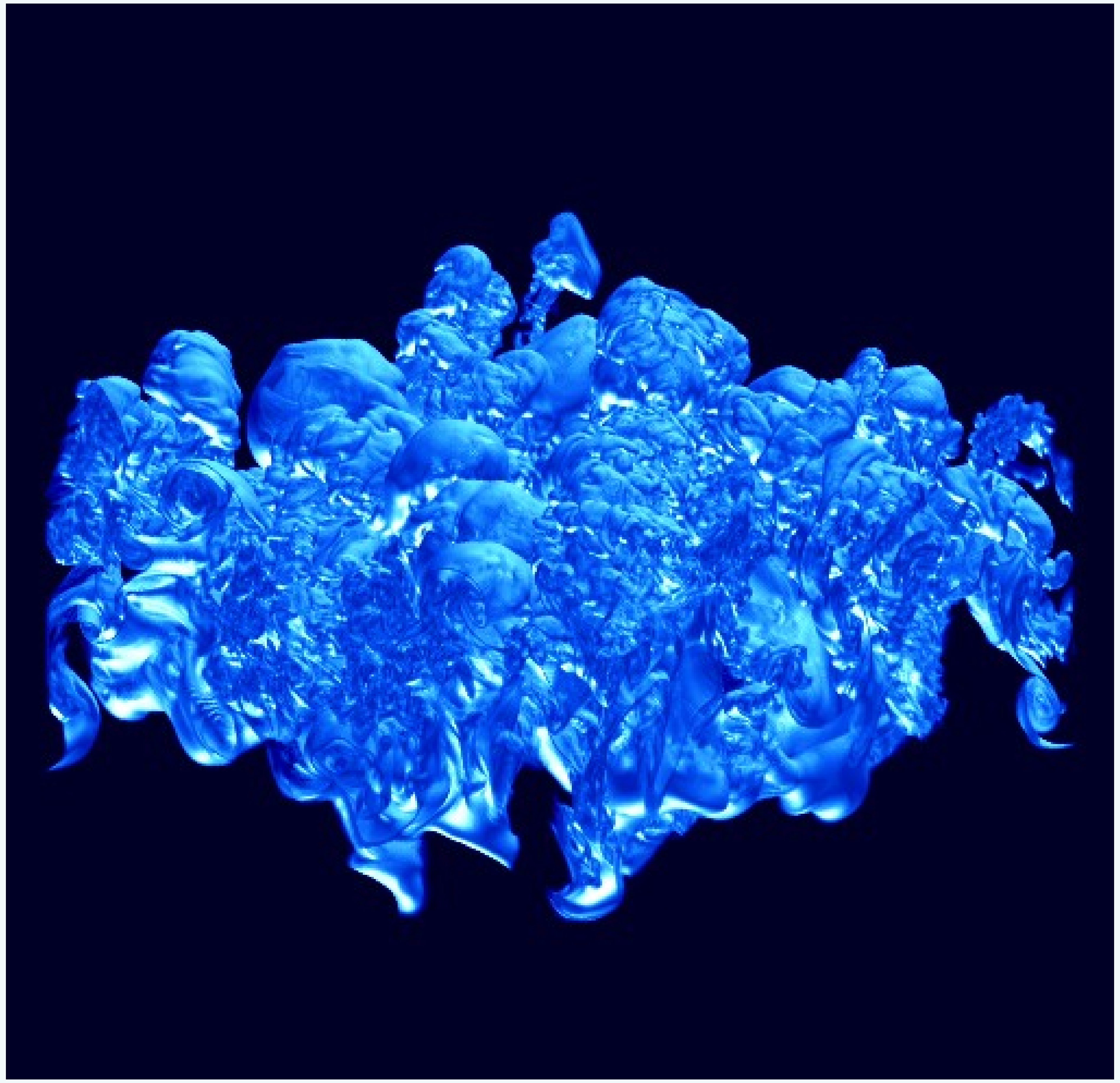


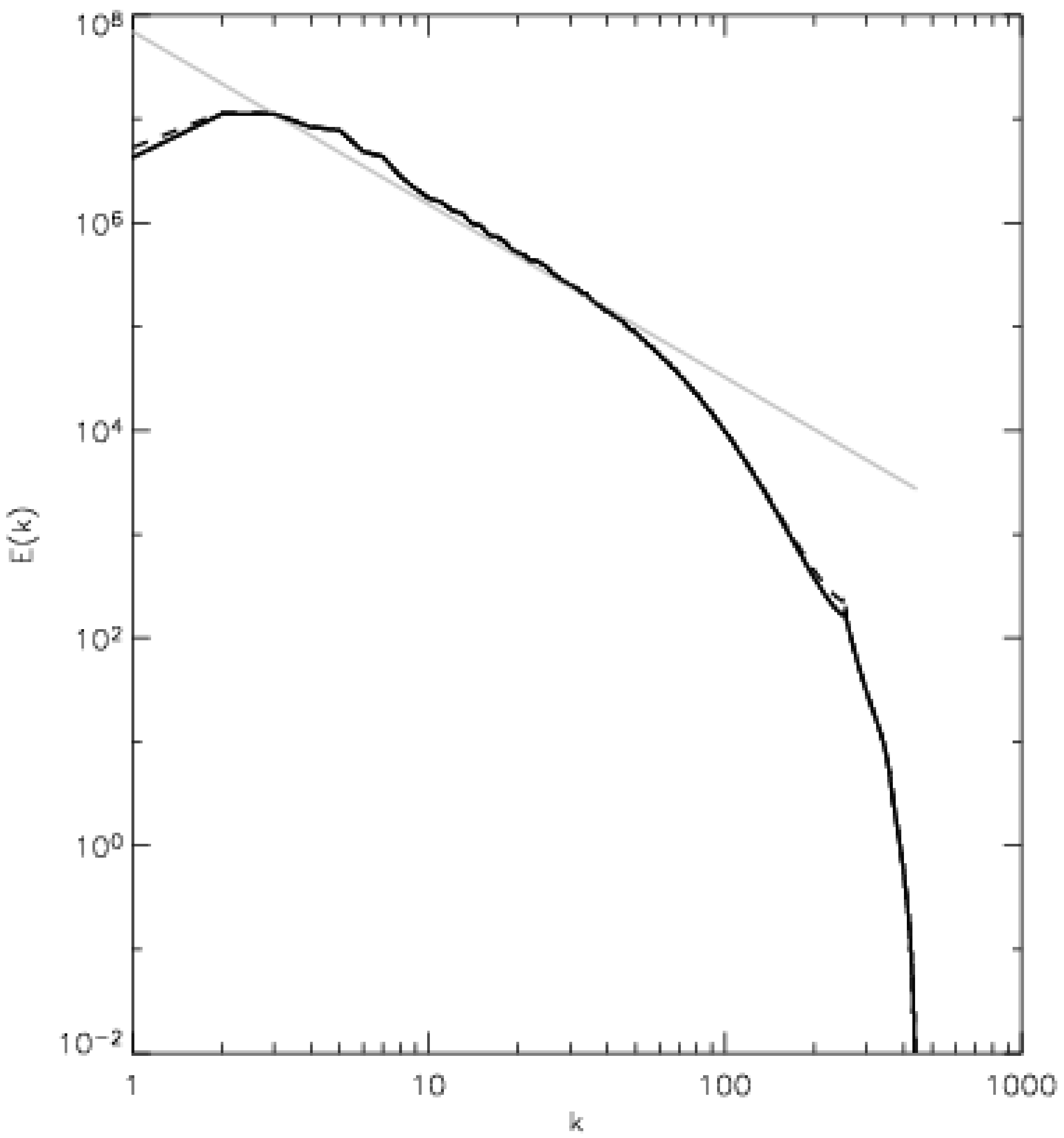
$$E(K) = \int_{K=|k|} [\hat{u}^2(k) + \hat{v}^2(k) + \hat{w}^2(k)]$$

$$k^2 = k_x^2 + k_y^2 + k_z^2$$

Example: Turbulence

- The power spectrum of the velocity field is used to understand the turbulent energy cascade.





What is the FFT really doing?

- Consider our expression for the discrete Fourier transform, but let's group the odd and even terms

$$
\begin{aligned}
F_k &= \sum_{n=0}^{N-1} f_n e^{-2\pi i k n/N} \\
&= \sum_{r=0}^{N/2-1} f_{2r} e^{-2\pi i k(2r)/N} + \sum_{r=0}^{N/2-1} f_{2r+1} e^{-2\pi i k(2r+1)/N}
\end{aligned}
$$

- Now look at the even terms:

$$E_k \equiv \sum_{r=0}^{N/2-1} f_{2r} e^{-2\pi i k(2r)/N} = \boxed{\sum_{r=0}^{N/2-1} f_{2r} e^{-2\pi i k r/(N/2)}}$$

DFT of the N/2 even samples

What is the FFT really doing?

- Now the odd terms:

$$\sum_{r=0}^{N/2-1} f_{2r+1} e^{-2\pi i k(2r+1)/N} = e^{-2\pi i k/N} \sum_{r=0}^{N/2-1} f_{2r+1} e^{-2\pi i k r/(N/2)}$$
$$= e^{-2\pi i k/N} O_k$$

- O_k is the just the DFT of the N/2 odd samples
- Define: $\omega^k \equiv e^{-2\pi i k/N}; \quad F_k = E_k + \omega^k O_k$
- In doing this, we went from FFTs involving N samples to 2 FFTs with N/2 samples → the number of wave numbers is also halved.

FFTs

- Periodicity tells us that $F_{k+N/2} = E_k - \omega^k O_k$
- E_k and O_k are simply periodic with N/2
- We can apply this decomposition recursively
- Let's consider the case where N = 2^m

$$E_k \equiv \sum_{r=0}^{N/2-1} f_{2r} e^{-2\pi i k r/(N/2)}$$

$$O_k \equiv \sum_{r=0}^{N/2-1} f_{2r+1} e^{-2\pi i k r/(N/2)}$$

FFTs

- Consider 8 samples: $f_n = \{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$
  - There are 3 levels that we break this down
- Level 3:
  - This is the final answer. We want F_k for k = 0…7
  - Express this in terms of the FFTs of the even and odd terms

$$F_k = \mathcal{F}_k(f_0, f_2, f_4, f_6) + \omega^k \mathcal{F}_k(f_1, f_3, f_5, f_7)$$

  - This is defined for k=0…3 since that's what the FFTs of each N/2 set of samples is defined with

FFTs $F_k = \mathcal{F}_k(f_0, f_2, f_4, f_6) + \omega^k \mathcal{F}_k(f_1, f_3, f_5, f_7)$

- The other half of the frequencies (k = 4…7):

$$F_{k+N/2} = \mathcal{F}_k(f_0, f_2, f_4, f_6) - \omega^k \mathcal{F}_k(f_1, f_3, f_5, f_7)$$

- So 2 FFTs of 4 samples each gives us the FFT of 8 samples defined over 8 wavenumbers.

- Apply this recursively…

FFTs

- Eventually we get down to N FFTs of 1 sample each $\mathcal{F}_0(f_n) = f_n$
- We get the acceleration over the DFT by applying this recursively
  - Consider $N = 2^m$ samples, decomposing them m times
  - At the lowest level, we will be considering FFTs of a single sample

$$F_0 = f_0 e^{-2\pi i k 0/N} = f_0$$

  - The DFT of a single sample is just the sample itself
- At each of these levels, we have much smaller FFTs to consider, so we save on a lot of work.
- Putting it all together, the computational work scales as O(N log N)

```python
def fft(f_n):
    N = len(f_n)
    if N == 1:
        return f_n
    else:
        # split into even and odd and find the FFTs of each half
        f_even = f_n[0:N:2]
        f_odd = f_n[1:N:2]
        F_even = fft(f_even)
        F_odd = fft(f_odd)
        # combine them.  Each half has N/2 wavenumbers, but due to
        # periodicity, we can compute N wavenumbers
        omega = np.exp(-2*np.pi*1j/N)
        # allocate space for the frequency components
        F_k = np.zeros((N), dtype=np.complex128)
        for k in range(N/2):
            F_k[k] = F_even[k] + omega**k * F_odd[k]
            F_k[N/2 + k] = F_even[k] - omega**k * F_odd[k]
return F_k
```

Beyond data analysis

- So far we've focused on using the FFT for data analysis
- We can also use it directly for solving (some) partial differential equations (Chapter 9).
- Consider the Poisson equation: $\frac{d^2\phi_n}{dx^2} = f_n$
- Express in terms of the FFTs

$$\phi(x) = \int \Phi(k) e^{-2\pi i k x} dk$$

$$f(x) = \int F(k) e^{-2\pi i k x} dk$$

$$\phi(x) = \int \Phi(k) e^{-2\pi i k x} dk$$

$$f(x) = \int F(k) e^{-2\pi i k x} dk$$

Beyond data analysis

- Easy to differentiate:

$$\frac{d^2\phi(x)}{dx^2} = -4\pi^2 k^2 \int \Phi(k) e^{-2\pi i k x} dk$$

- Then: $-4\pi^2 k^2 \Phi(k) = F(k)$
- Easy to solve: $\Phi(k) = -\frac{F(k)}{4\pi^2 k^2}$
- Solve algebraically in Fourier space and then transform back
    - Only works for certain boundary conditions