Computational Physics – Lecture 17

(15 April 2020)

Recap

In the previous class, we understood how to compute the DFT using the FFT algorithm.

We also set up a procedure for computing the FT for a function as a DFT of n numbers.

Today's plan

In today's class, we want to go deeper into the idea of computing the FT of a function as a DFT of n numbers.

FT using DFT

The FT of a function $f(x)$ is the infinite integral

$$\tilde{f}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp(-ikx). \tag{1}$$

We can compute $\tilde{f}(k)$ at n discrete points k_q $(q = 0, \ldots, n-1)$ in three steps by

1. Sampling $f(x)$ at n evenly spaced sample points
2. Computing the DFT of the sample
3. Multiplying the DFT by some simple numbers

FT using DFT

We denoted the sample by

$$f(x_i) \quad \text{for } i = 0, \ldots, n-1. \tag{2}$$

where the x_i are evenly spaced sample points

$$x_i = x_{\min} + i\Delta, \tag{3}$$

for $i = 0, \ldots, n-1$, and Δ being the sampling rate.

The maximum value of x_i is $x_{\max} = x_{\min} + (n-1)\Delta$.

The numbers n and Δ are related by

$$n = \frac{x_{\max} - x_{\min}}{\Delta} + 1 \tag{4}$$

FT using DFT

Then we wrote

$$
\begin{aligned}
\tilde{f}(k_q) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp\left(-ik_q x\right), \\
&= \frac{1}{\sqrt{2\pi}} \sum_{x_p = x_{\min}}^{x_{\max}} \Delta \cdot f(x_p) \cdot \exp\left(-ik_q x_p\right), \\
&= \Delta \cdot \frac{1}{\sqrt{2\pi}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ik_q x_p\right), \\
&= \Delta \cdot \frac{1}{\sqrt{2\pi}} \sum_{p=0}^{n-1} f(x_{\min} + p\Delta) \cdot \exp\left(-ik_q \cdot [x_{\min} + p\Delta]\right), \\
&= \Delta \cdot \sqrt{\frac{n}{2\pi}} \cdot \exp\left(-ik_q \cdot x_{\min}\right) \cdot \mathrm{DFT}\left[\{f(x_p)\}\right],
\end{aligned}
$$

where $p = 0, \ldots, n-1$, and $k_q = 2\pi q / n\Delta$, for $q = 0, \ldots, n-1$.

FT using DFT

Finally, we noted that we have ended up computing a sample of the FT.

The frequencies are

$$k_0 = 0, k_1 = \frac{2\pi}{n\Delta}, k_2 = \frac{4\pi}{n\Delta}, \ldots, k_{n-1} = \frac{2\pi(n-1)}{n\Delta}. \tag{5}$$

The quantity

$$f_n = \frac{1}{2\Delta} \tag{6}$$

turns out to be important and is called the Nyquist frequency.

So we can write

$$k_0 = 0, k_1 = \frac{4\pi f_n}{n}, k_2 = \frac{8\pi f_n}{n}, \ldots, k_{n-1} = \frac{4\pi(n-1)f_n}{n}. \tag{7}$$

An example using Numpy

Let us try to compute the Fourier Transform of a Gaussian

$$f(x) = \exp(-x^2). \tag{8}$$

using Numpy.

In this case, the Fourier transform can be obtained analytically and is given by

$$\tilde{f}(k) = \frac{1}{\sqrt{2}} \exp(-k^2/4). \tag{9}$$

So the Fourier transform of a Gaussian is a Gaussian.

An example using Numpy

Notice that our chosen function is real.

So we would expect its Fourier transform to be complex. But in our case the Fourier transform is also real. Why is that?

The reason is that our chosen function is also even, that is $f(x) = f(-x)$.

When a function is real and even, its Fourier transform is real.

An example using Numpy

Let $f(x)$ be real and even. Then consider the complex conjugate of its Fourier transform $\tilde{f}(k)$. This is given by

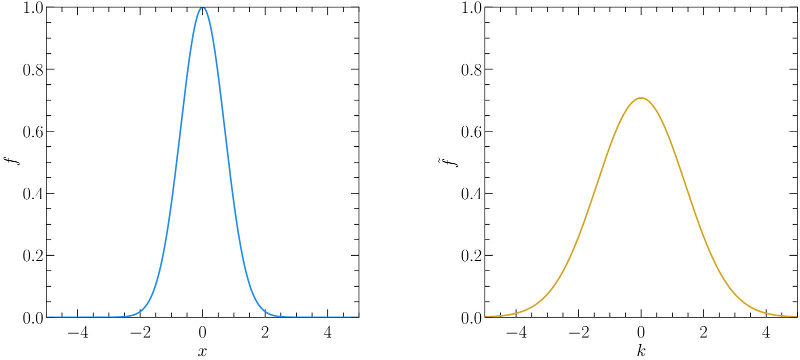
$$\begin{aligned}
\left[\tilde{f}(k)\right]^* &= \left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp(-ikx)\right]^* && (10)\\
&= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp(ikx) && (11)\\
&= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \mathrm{d}x \cdot f(-x) \cdot \exp(ikx) && (12)\\
&= \frac{-1}{\sqrt{2\pi}}\int_{\infty}^{-\infty} \mathrm{d}y \cdot f(y) \cdot \exp(-iky) && (13)\\
&= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} \mathrm{d}y \cdot f(y) \cdot \exp(-iky) && (14)\\
&= \tilde{f}(k). && (15)
\end{aligned}$$

An example using Numpy

Using similar logic, we can also see that $\tilde{f}(k)$ is even. This is the case because

$$\tilde{f}(-k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(x) \cdot \exp\left(ikx\right) \tag{16}$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}x \cdot f(-x) \cdot \exp(ikx) \tag{17}$$
$$= \frac{-1}{\sqrt{2\pi}} \int_{\infty}^{-\infty} \mathrm{d}y \cdot f(y) \cdot \exp(-iky) \tag{18}$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}y \cdot f(y) \cdot \exp(-iky) \tag{19}$$
$$= \tilde{f}(k). \tag{20}$$

An example using Numpy

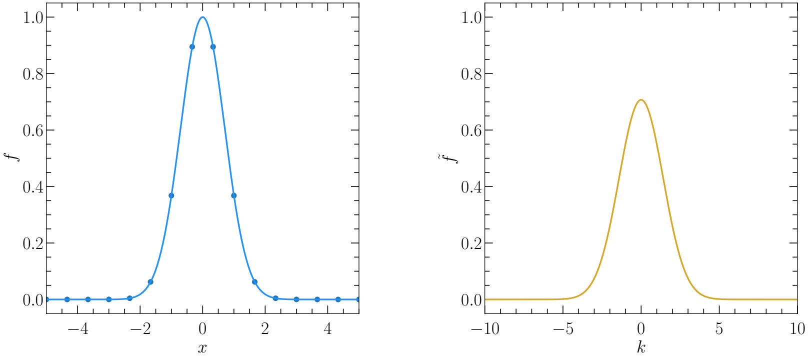


Our chosen function (left) and its analytically calculated Fourier transform (right).

An example using Numpy

```
def f(x):
    return np.exp(-x*x)

xmin = -5.0
xmax = 5.0

numpoints = 16
dx = (xmax-xmin)/(numpoints-1)

sampled_data = np.zeros(numpoints)
xarr = np.zeros(numpoints)

for i in range(numpoints):
    sampled_data[i] = f(xmin+i*dx)
    xarr[i] = xmin+i*dx
```

An example using Numpy



Our chosen function with our sample (left) and its analytically calculated Fourier transform (right).

An example using Numpy

We now use `numpy.fft.fft` to compute the DFT.

```
nft = np.fft.fft(sampled_data, norm='ortho')
```

(The `norm='ortho'` keyword argument asks `numpy.fft.fft` to use unitary transform.)

An example using Numpy

Before we proceed, we should check the DFT convention used by Numpy.

Recall that our DFT is written as

$$\tilde{f}(k_q) = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-ik_q x_p\right), \tag{21}$$

and the inverse DFT as

$$f(x_p) = \frac{1}{\sqrt{n}} \sum_{q=0}^{n-1} \tilde{f}(k_q) \cdot \exp\left(ik_q x_p\right), \tag{22}$$

for $p = 0, 1, \ldots, n-1$ and $q = 0, 1, \ldots, n-1$, and

$$x_p = p\Delta \tag{23}$$

and

$$k_q = 2\pi q/n\Delta. \tag{24}$$

An example using Numpy

This can be written as

$$\tilde{f}(k_q) = \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-2\pi i k_q x_p\right), \tag{25}$$

and

$$f(x_p) = \frac{1}{\sqrt{n}} \sum_{q=0}^{n-1} \tilde{f}(k_q) \cdot \exp\left(2\pi i k_q x_p\right), \tag{26}$$

for $p = 0, 1, \ldots, n-1$ and $q = 0, 1, \ldots, n-1$, and

$$x_p = p\Delta \tag{27}$$

and

$$k_q = q/n\Delta. \tag{28}$$

An example using Numpy

This is the convention followed by Numpy (if `norm='ortho'` keyword argument is given).

So now we can compute

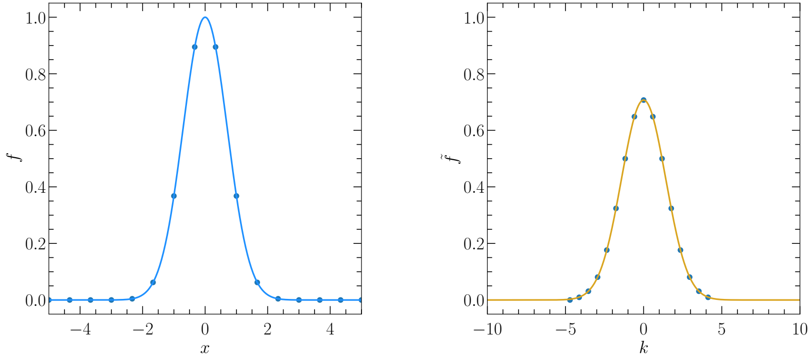
$$\tilde{f}(k) = \Delta \cdot \sqrt{\frac{n}{2\pi}} \cdot \exp\left(-ik_q \cdot x_{\min}\right) \cdot \mathrm{DFT}\left[f\left(x_p\right)\right] \tag{29}$$

An example using Numpy

Now we can write

```
karr = np.fft.fftfreq(numpoints, d=dx)
karr = 2*np.pi*karr
factor = np.exp(-1j * karr * xmin)

aft = dx * np.sqrt(numpoints/(2.0*np.pi)) * factor * nft
```

An example using Numpy



Result for $x_{\min} = -5$, $x_{\max} = 5$, and $n = 16$.

An example using Numpy

Notice that $\tilde{f}(k_q)$ is a periodic function of q with period n.

$$\begin{aligned}
\tilde{f}(k_{q+n}) &= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-2\pi i x_p \cdot \frac{q+n}{n\Delta}\right), \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-2\pi i \cdot \frac{x_p q}{n\Delta}\right) \cdot \exp\left(-2\pi i \cdot \frac{x_p n}{n\Delta}\right), \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-2\pi i \cdot \frac{x_p q}{n\Delta}\right) \cdot \exp\left(-2\pi i \cdot \frac{p\Delta n}{n\Delta}\right), \\
&= \frac{1}{\sqrt{n}} \sum_{p=0}^{n-1} f(x_p) \cdot \exp\left(-2\pi i \cdot \frac{q}{n\Delta} \cdot x_p\right) \cdot 1 \\
&= \tilde{f}(k_q)
\end{aligned} \tag{30}$$

An example using Numpy

So now, instead of

$$k_q = q/n\Delta \quad \text{with} \quad q = 0, 1, \ldots, n-1, \tag{31}$$

we can use

$$k_q = q/n\Delta \quad \text{with} \quad q = 1, \ldots, n, \tag{32}$$

or

$$k_q = q/n\Delta \quad \text{with} \quad q = 2, \ldots, n+1, \tag{33}$$

without any change in our formulas.

An example using Numpy

So now, instead of

$$k_q = q/n\Delta \quad \text{with} \quad q = 0, 1, \ldots, n-1, \tag{34}$$

we can use

$$k_q = q/n\Delta \quad \text{with} \quad q = 1, \ldots, n, \tag{35}$$

or

$$k_q = q/n\Delta \quad \text{with} \quad q = 2, \ldots, n+1, \tag{36}$$

without any change in our formulas.

An example using Numpy

Or, we could use

$$k_q = q/n\Delta \quad \text{with} \quad q = -\frac{n}{2}, \ldots, n - 1 - \frac{n}{2}, \tag{37}$$

that is

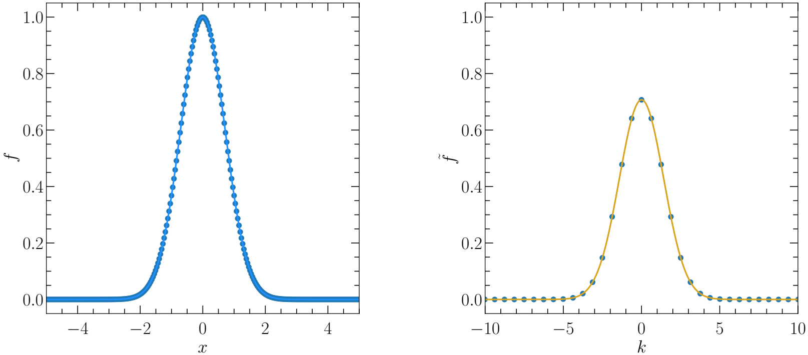
$$k_q = q/n\Delta \quad \text{with} \quad q = -\frac{n}{2}, \ldots, \frac{n}{2} - 1. \tag{38}$$

This is *almost* how Numpy computes the DFT. The actual k values are

$$k_q = 0, \frac{q}{n\Delta} \text{ with } q = 1, \ldots, \frac{n}{2} - 1, \frac{q}{n\Delta} \text{ with } q = -\frac{n}{2}, \ldots, -1.$$

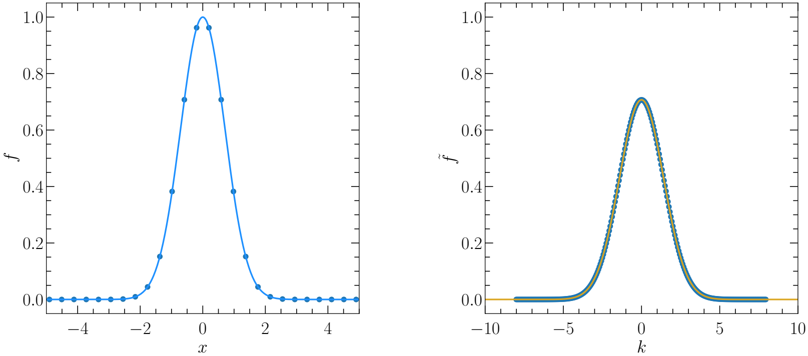
(This is another thing to look for in library documentation.)

An example using Numpy



Result if we take 256 points instead of 16 in the same range $(x_{\text{min}}, x_{\text{max}} = -5, 5)$.

An example using Numpy



Result if we take 256 points instead of 16 in the range $(x_{\min}, x_{\max} = -50, 50)$.

Choice of FT parameters

We choose x_{min} and x_{max} such that if the function is non-zero in a finite range, that range is contained within $(x_{\text{min}}, x_{\text{max}})$.

If the function is non-zero everywhere, then $(x_{\text{max}} - x_{\text{min}})$ represents a limitation of our analysis. This is largest scale that we can study.

In this case, we try to choose x_{min} and x_{max} such that the interval $(x_{\text{min}}, x_{\text{max}})$ contains "most of the function".

Or, if the function is periodic-like, we try to choose x_{min} and x_{max} such that the behaviour of the function within $(x_{\text{min}}, x_{\text{max}})$ is somewhat similar to the behavior of the function outside this interval.

Choice of FT parameters

Recall that the minimum non-zero k at which we can calculate $\tilde{f}(k)$ is

$$k_1 = \frac{2\pi}{n\Delta}. \tag{39}$$

This value is called the *fundamental frequency* or the *fundamental mode*.

Because

$$n = \frac{x_{\max} - x_{\min}}{\Delta} + 1, \tag{40}$$

the larger the value of $(x_{\max} - x_{\min})$, the smaller is the fundamental mode (for fixed Δ).

The smallest k value thus corresponds to the biggest physical scale. Ideally, we want to make $(x_{\max} - x_{\min})$ as large as we can.

Choice of FT parameters

Available computer memory puts a limitation on the maximum n with which we can work.

(This is why we are always looking for bigger computers, e.g., in cosmology.)

Once we have chosen our $x_{\rm min}$ and $x_{\rm max}$ by looking at the function and we have chosen the n by looking at our computer, the Δ is simply

$$\Delta = \frac{x_{\rm max} - x_{\rm min}}{n-1}. \tag{41}$$

The quantity Δ is the smallest physical scale that we are using. Also known as the *resolution*.

For fixed $x_{\rm min}$ and $x_{\rm max}$, the larger the n, the smaller the Δ and the larger the Nyquist frequency

$$f_n = \frac{1}{2\Delta}. \tag{42}$$

Choice of FT parameters

The spatial resolution sets the largest k value in the problem. The largest spatial scale sets the smallest k value in the problem.

Because $x_p = x_{\min} + p\Delta$ and $k_q = 2\pi q/n\Delta$, the resolution in configuration space is

$$\delta x = \Delta \tag{43}$$

and the resolution in Fourier space is

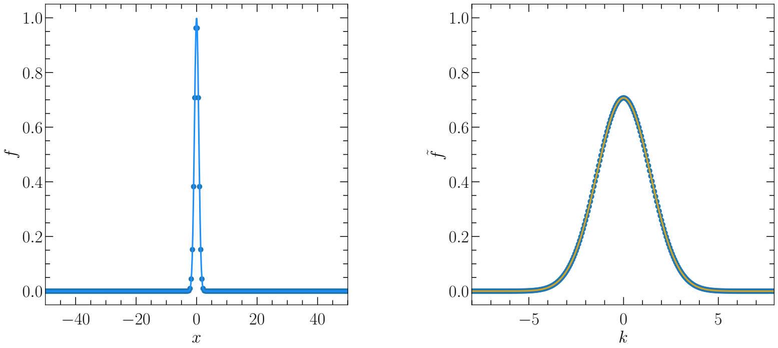
$$\delta k = \frac{2\pi}{n\Delta}. \tag{44}$$

This gives us the uncertainty principle

$$\delta x \cdot \delta k = \frac{2\pi}{n}. \tag{45}$$

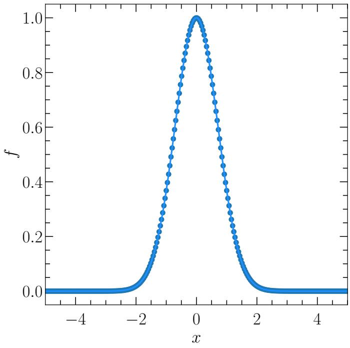
For a fixed n – that is, for a fixed computer – better spatial resolution will give you worse Fourier resolution.

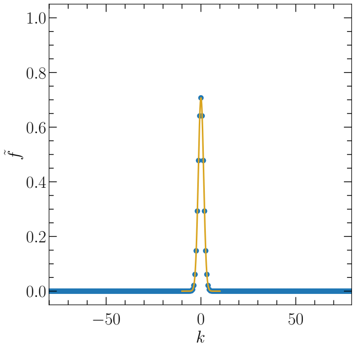
Choice of FT parameters



Here we have $x_{\min} = -50$, $x_{\max} = 50$, with $n = 256$. This gives $\delta x = \Delta = 0.3921$, and $\delta k = 0.06259$. The $k_{\min}$ is -8.011 and $k_{\max}$ is 7.948.

Choice of FT parameters





Here we have $x_{\min} = -5$, $x_{\max} = 5$, with $n = 256$. This gives $\delta x = \Delta = 0.03921$, and $\delta k = 0.6259$. The $k_{\min}$ is -80.11 and $k_{\max}$ is 79.48.

Activity

Consider the (real, even) sinc function

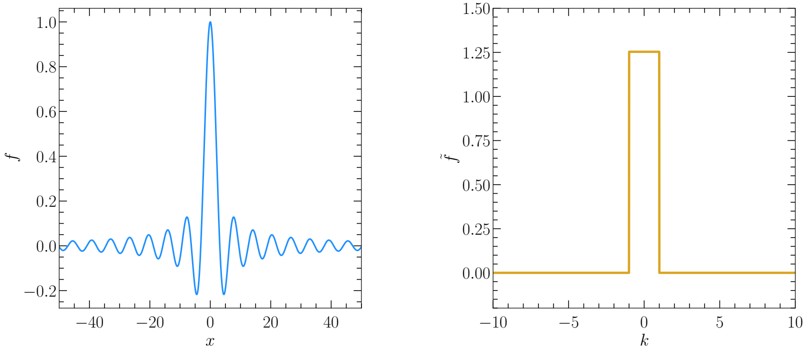
$$f(x) = \begin{cases} \frac{\sin x}{x}, & \text{if } x \neq 0 \\ 1, & \text{otherwise,} \end{cases} \tag{46}$$

for which the Fourier transform is known to be the (real, even) box function

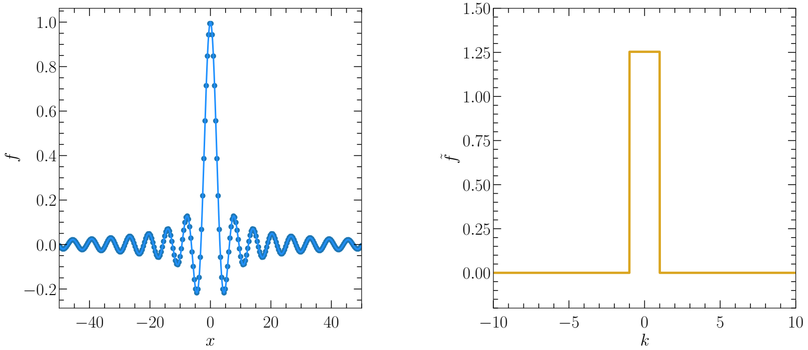
$$f(k) = \begin{cases} \sqrt{\pi/2} & \text{if } -1 < k < 1 \\ 0, & \text{otherwise.} \end{cases} \tag{47}$$

Compute this Fourier transform using `numpy.fft.fft` with $x_{\min} = -50$, $x_{\max} = 50$, and $n = 256$.

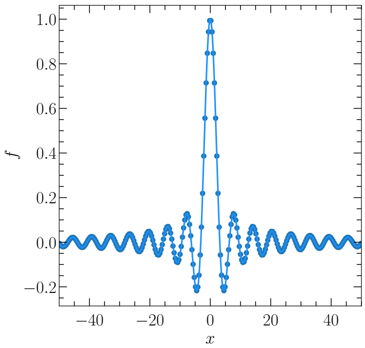
FT of the sinc function

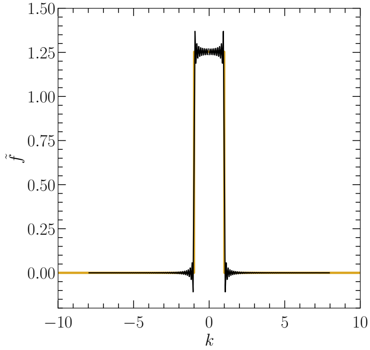


FT of the sinc function

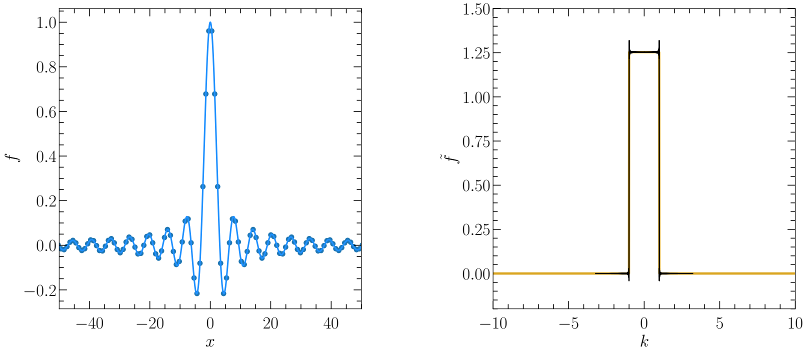


FT of the sinc function





FT of the sinc function



Result for $x_{\min} = -500$, $x_{\max} = 500$, and $n = 1024$.