Spectral analysis

Fundamental elements

Axel Hutt

preamble

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
….

class signal_class:
        def __init__(self):
                self.num_time = …

        def func1(self,….):
                ….


        def plot(self):
                ……

S = signal_class()
S.func1(..)
…..
S.plot()
```

preamble

- all provided Python-codes run under Python3.9

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
….

class signal_class:
        def __init__(self):
                self.num_time = …

        def func1(self,….):
                ….


        def plot(self):
                ……

S = signal_class()
S.func1(..)
…..
S.plot()
```

preamble

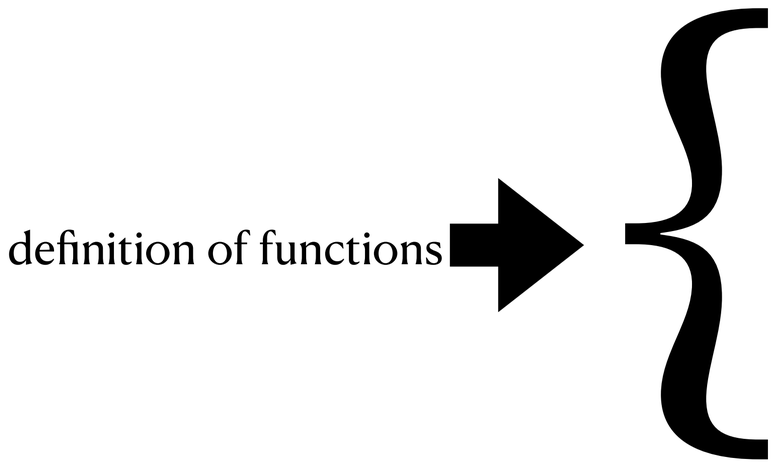
- all provided Python-codes run under Python3.9
- Python classes to enhance modular structure

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
….

class signal_class:
        def __init__(self):
                self.num_time = …

        def func1(self,….):
                ….


        def plot(self):
                ……

S = signal_class()
S.func1(..)
…..
S.plot()
```

preamble

- all provided Python-codes run under Python3.9
- Python classes to enhance modular structure

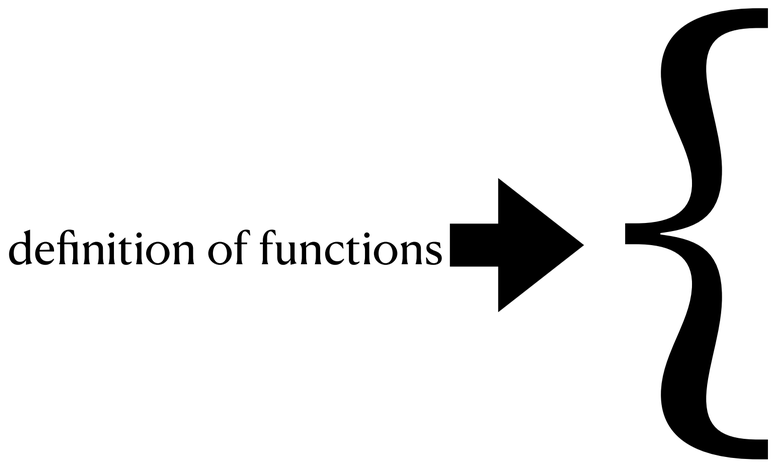
definition of functions



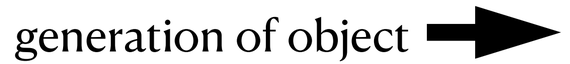
```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
....

class signal_class:
        def __init__(self):
                self.num_time = …

        def func1(self,….):
                ….


        def plot(self):
                ……

S = signal_class()
S.func1(..)
…..
S.plot()
```

preamble

- all provided Python-codes run under Python3.9
- Python classes to enhance modular structure



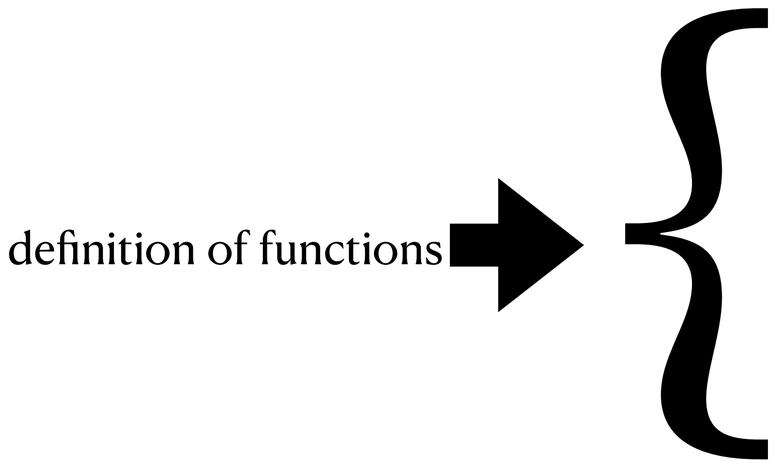

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
....

class signal_class:
        def __init__(self):
                self.num_time = …

        def func1(self,….):
                ….


        def plot(self):
                ……
```




```
S = signal_class()
S.func1(..)
…..
S.plot()
```

preamble

- all provided Python-codes run under Python3.9
- Python classes to enhance modular structure




```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np
....

class signal_class:
        def __init__(self):
                self.num_time = …

        def func1(self,….):
                ….


        def plot(self):
                ……
```

generation of object →

call of functions →

```
S = signal_class()
S.func1(..)
…..
S.plot()
```

data sampling

Fourier analysis

errors in analysis

linear filters

time-frequency analysis

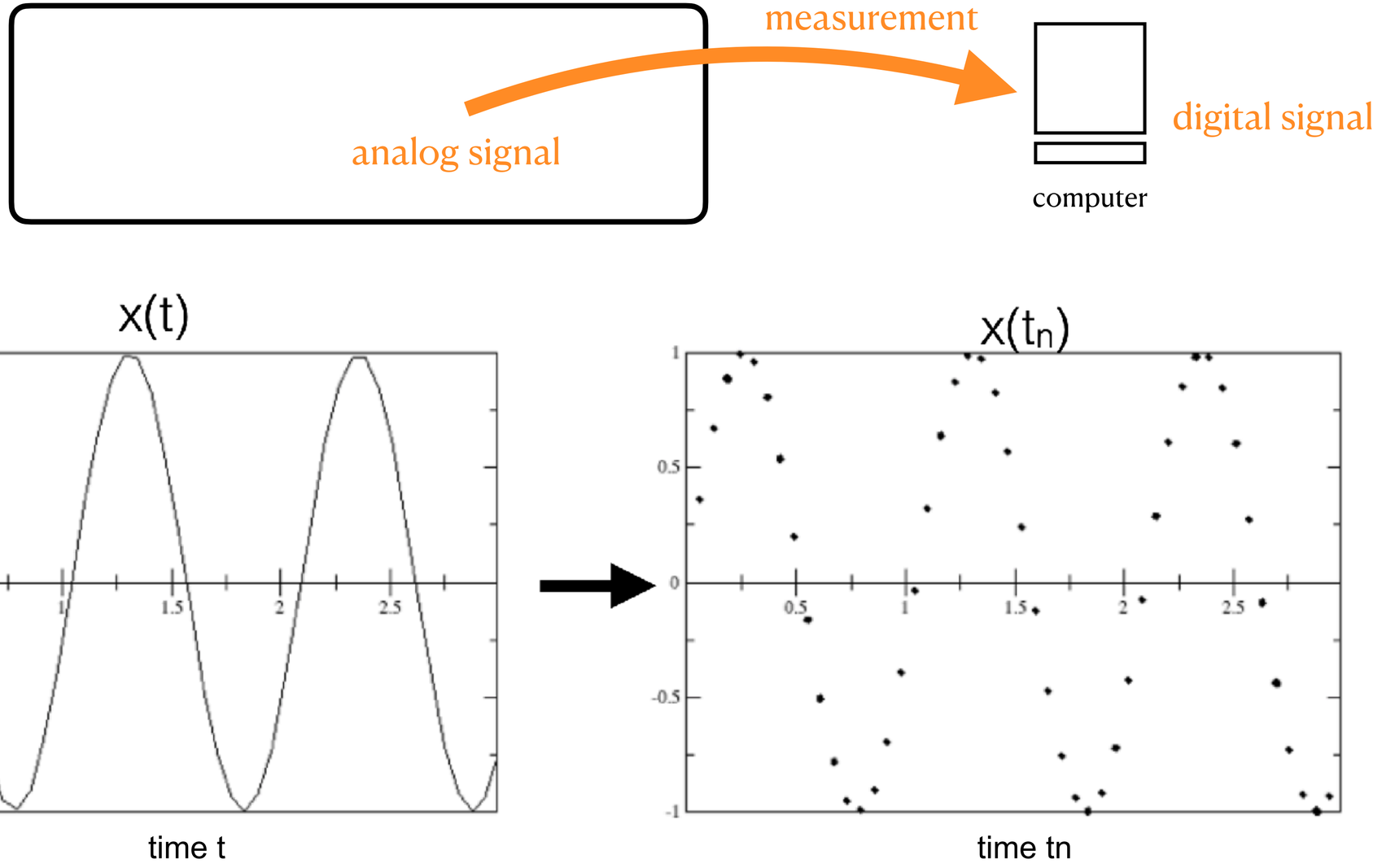
data sampling

Fourier analysis

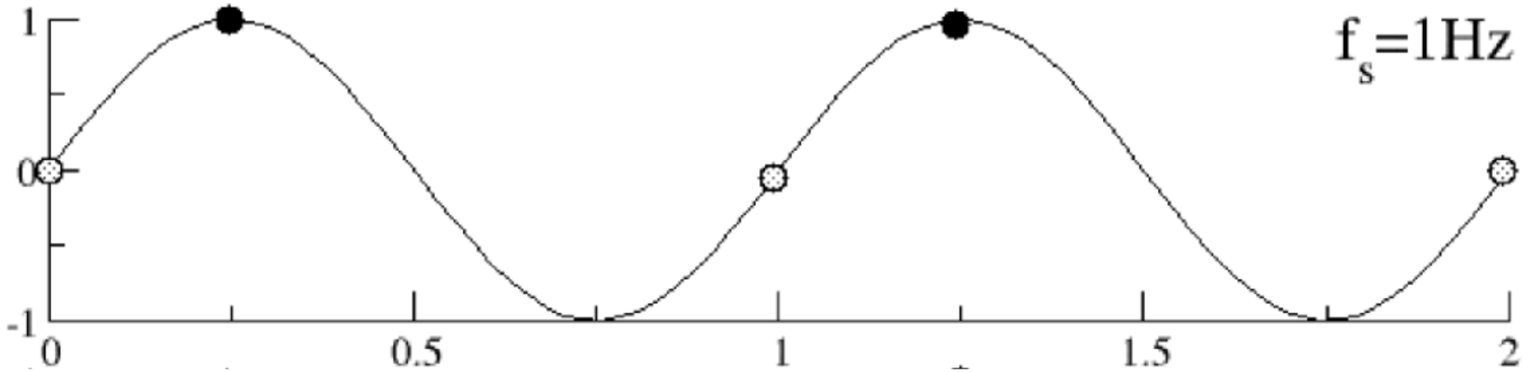
errors in analysis

linear filters

time-frequency analysis

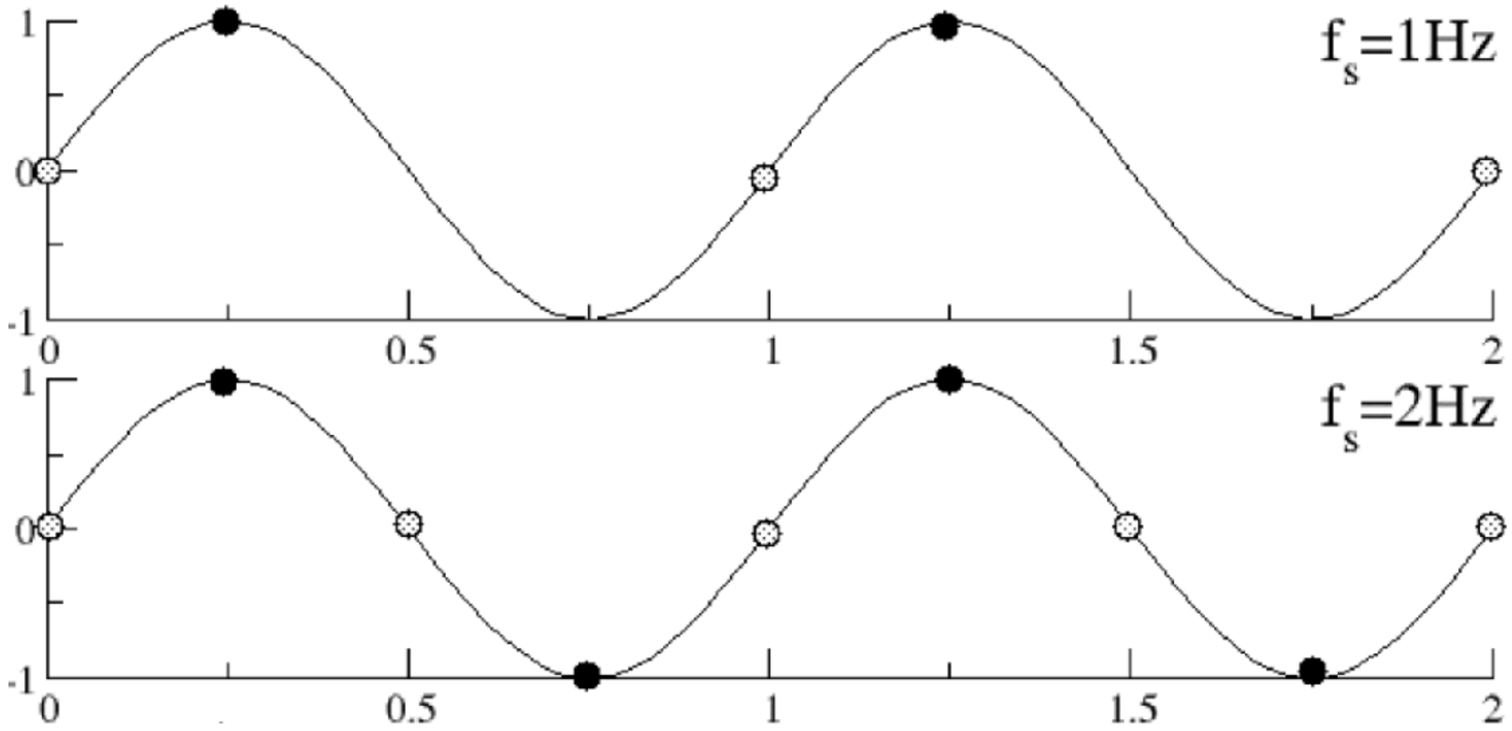

measurement
analog signal
digital signal
computer
x(t)
1
0.5
0
-0.5
-1
0.5
1
1.5
2
2.5
time t
x(tn)
1
0.5
0
-0.5
-1
0.5
1
1.5
2
2.5
time tn

$T_m = 1s$



$T_s = 1s$

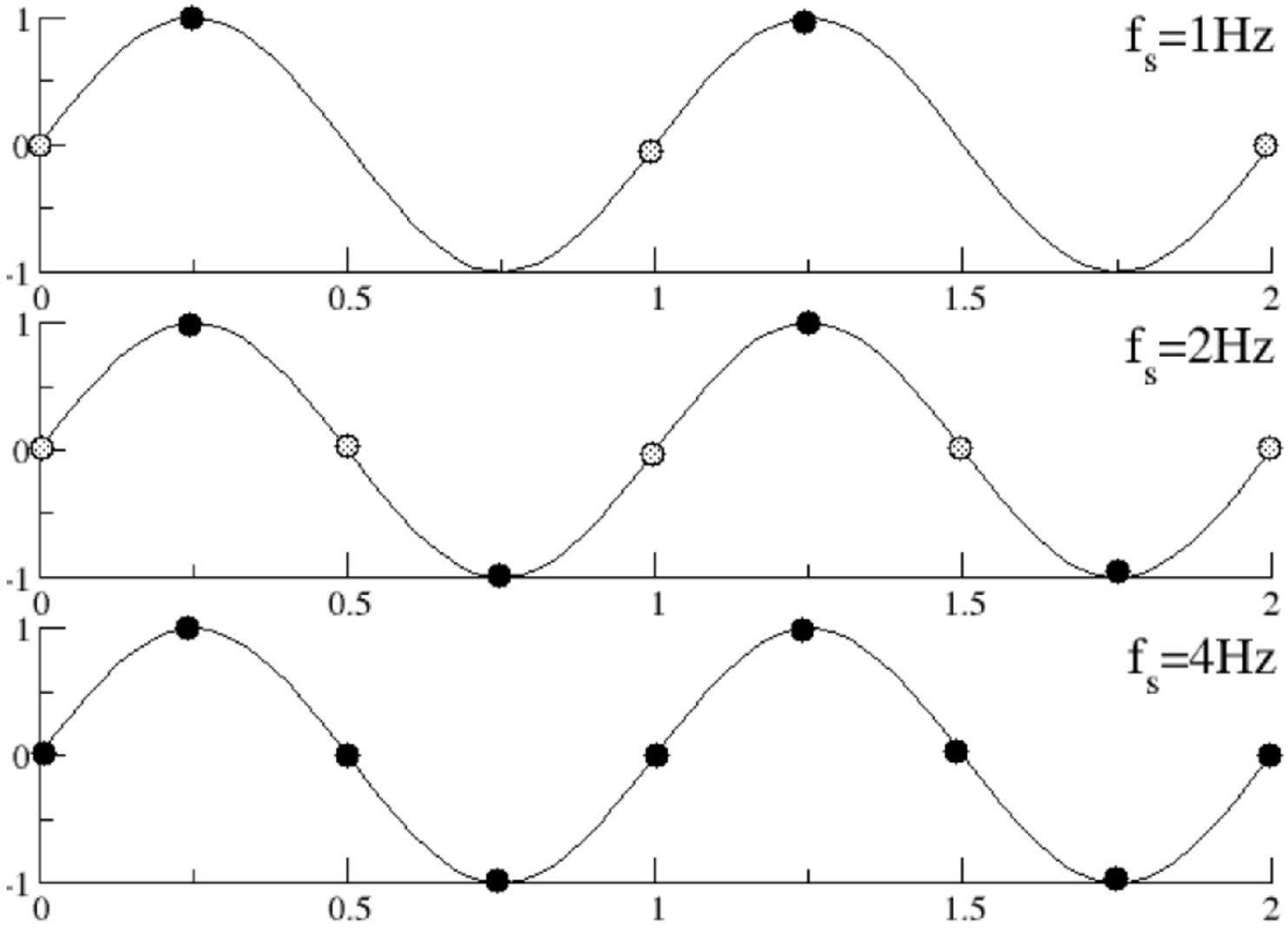
$T_m = 1s$



$T_s = 1s$

$T_s = 0.5s$

$T_m = 1s$

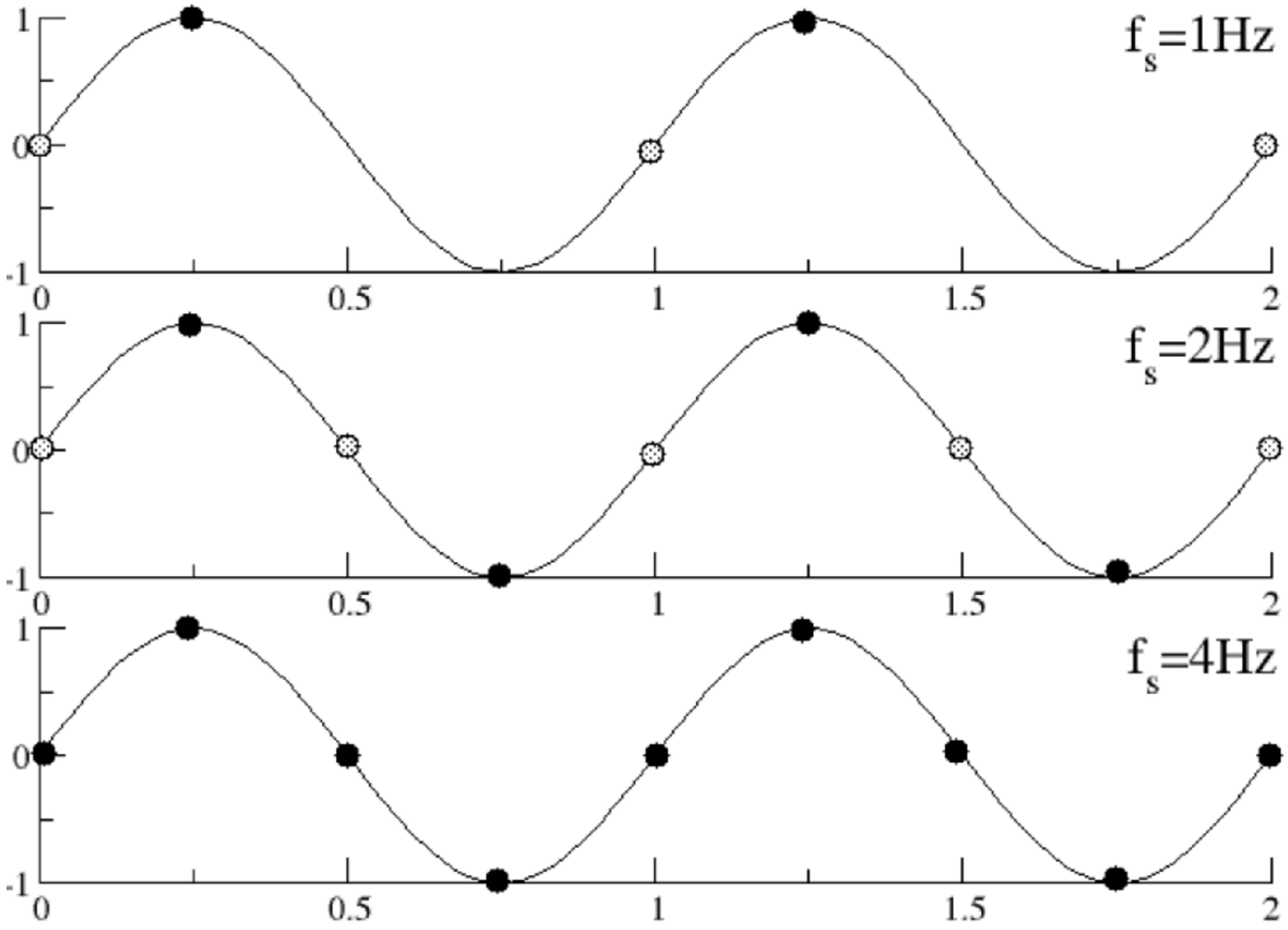


$T_s = 1s$

$T_s = 0.5s$

$T_s = 0.25s$

$T_m = 1s$



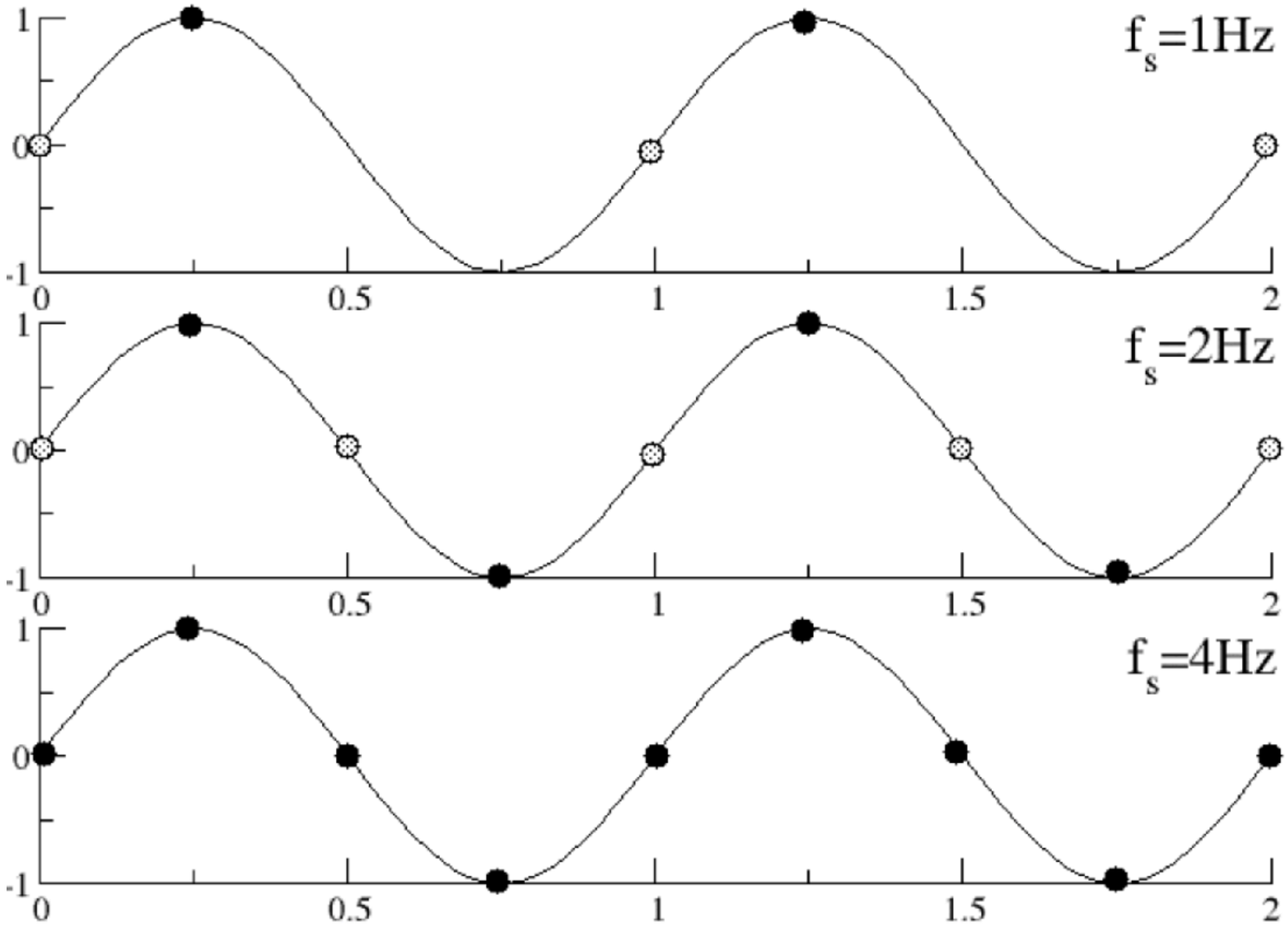
$T_s = 1s$

$T_s = 0.5s$

$T_s = 0.25s$

good sampling: $T_s \leq T_m/2$

$T_m = 1s$



$T_s = 1s$

$T_s = 0.5s$

$T_s = 0.25s$

Nyquist law : $f_s \geq 2f_m$

Sampling theorem

(Shannon/Nyquist/Whitaker/Kotelnikov)

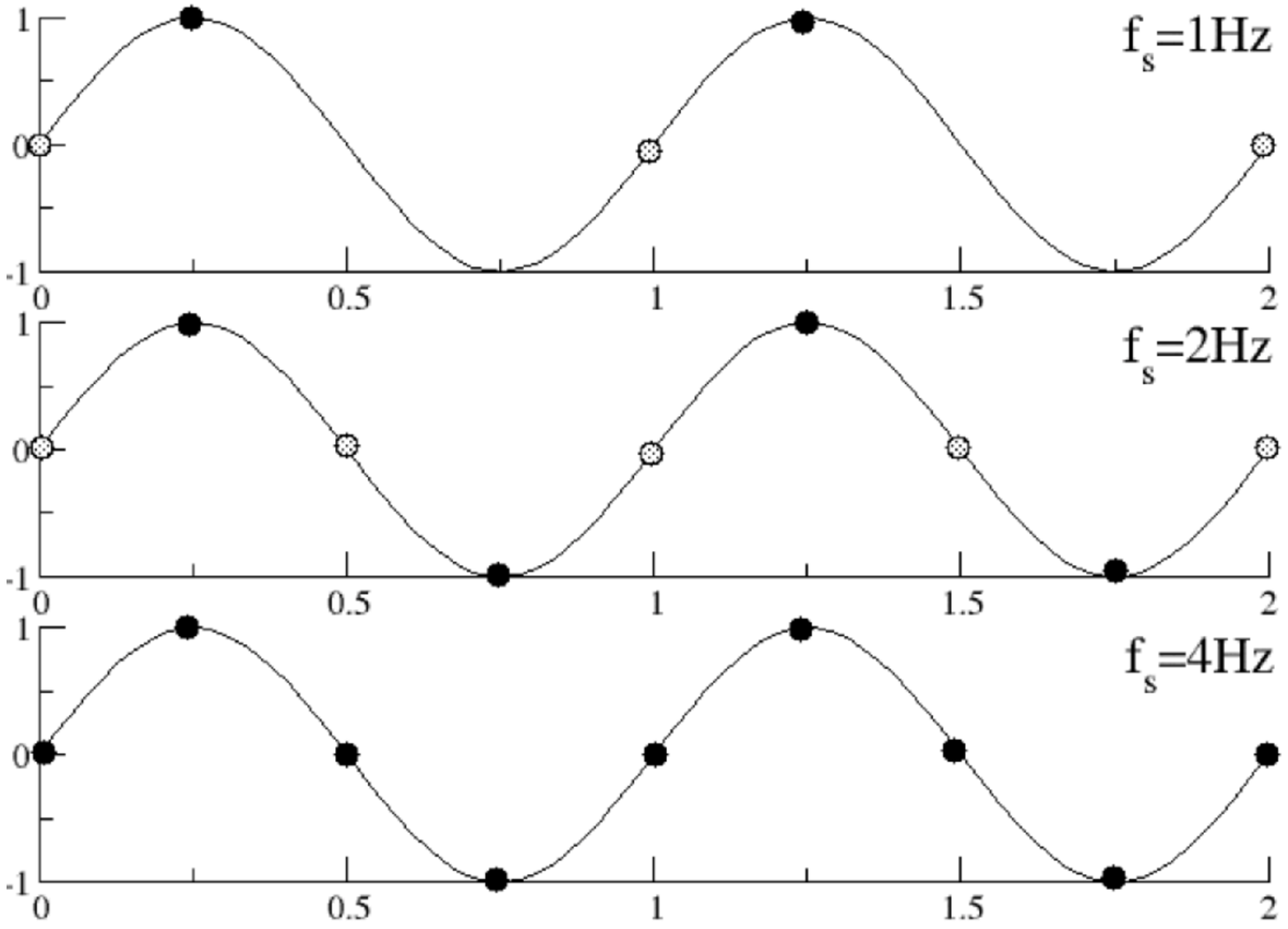
given:

- continuous function s(t)
- s(t) has maximum frequency f_m
- s(t) is sampled with frequency f_s , i.e. $s(t) \rightarrow s(t_n)\ ,\ t_n = n\Delta t = n/f_s$

hypothesis: information loss by sampling continuous function s(t)

objective: find sampling frequency f_s, for which no information loss occurs

solution: $f_s \geq 2f_m$



comment: in practice it is a good idea to choose $f_s \geq 4f_m$

data sampling

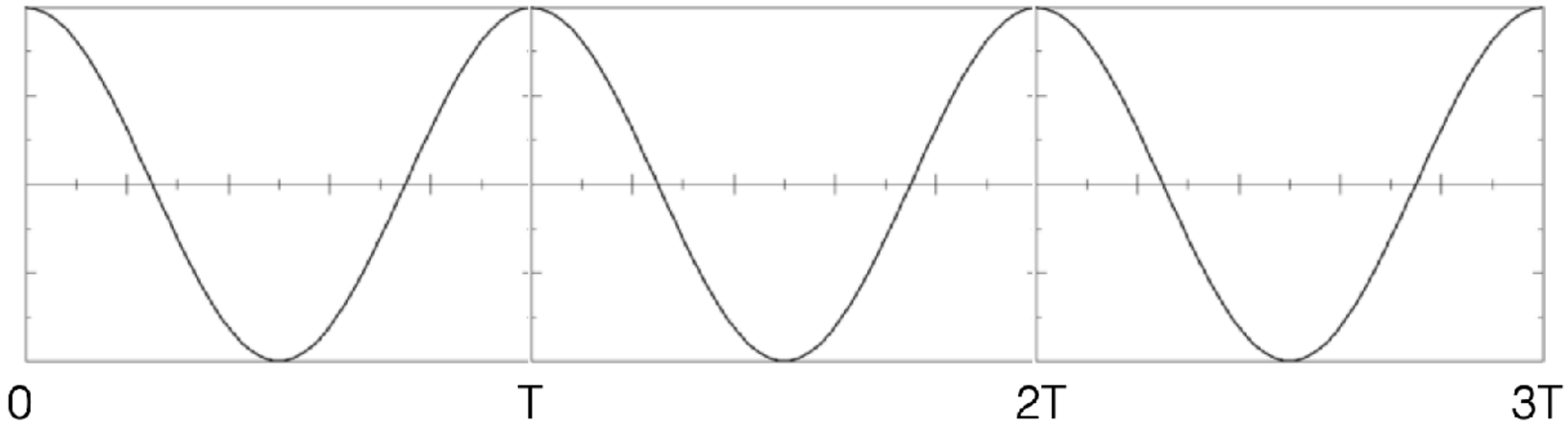
Fourier analysis

errors in analysis

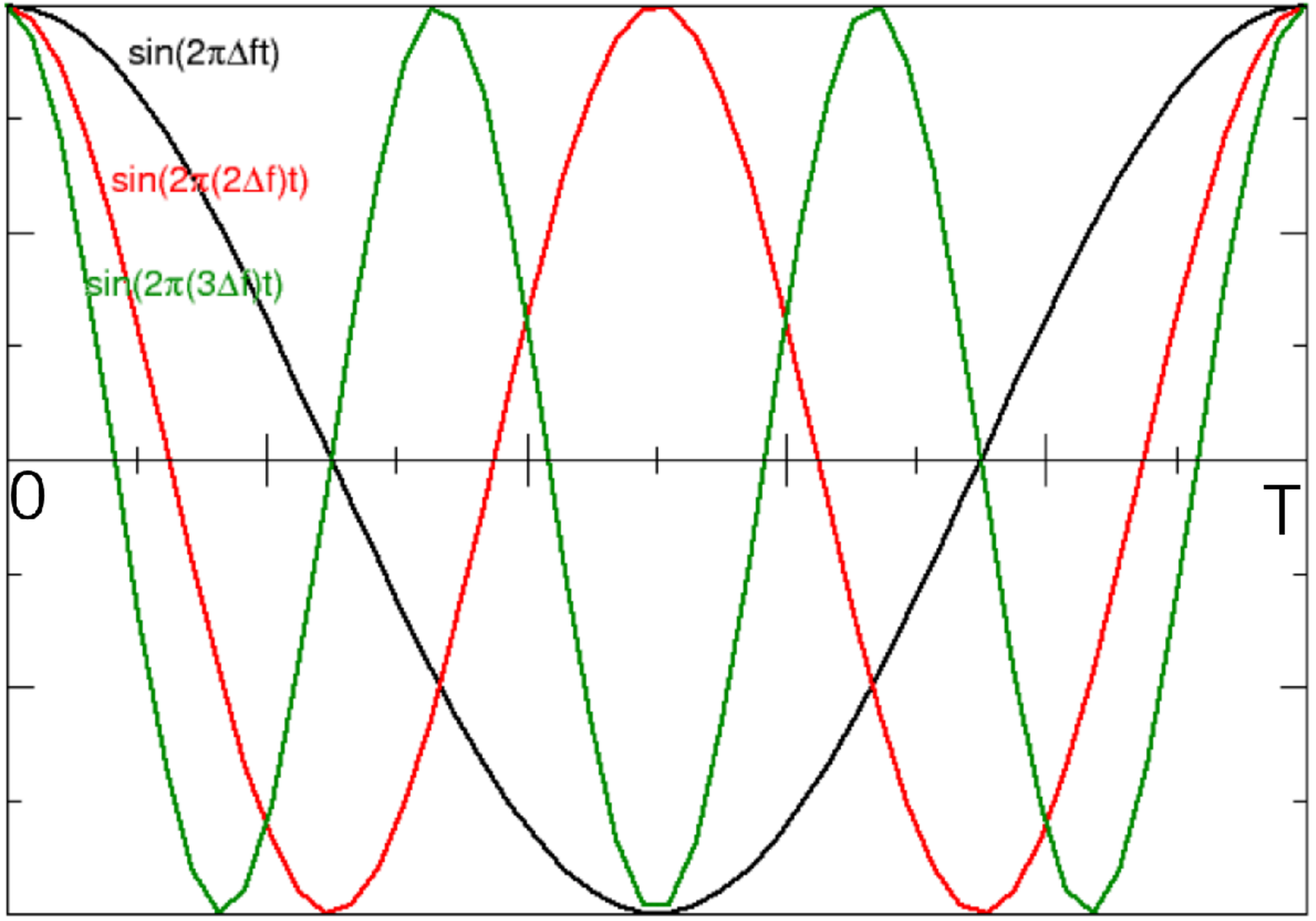
linear filters

time-frequency analysis

all-time assumption: the signal under study is periodic



the signal under study is periodic

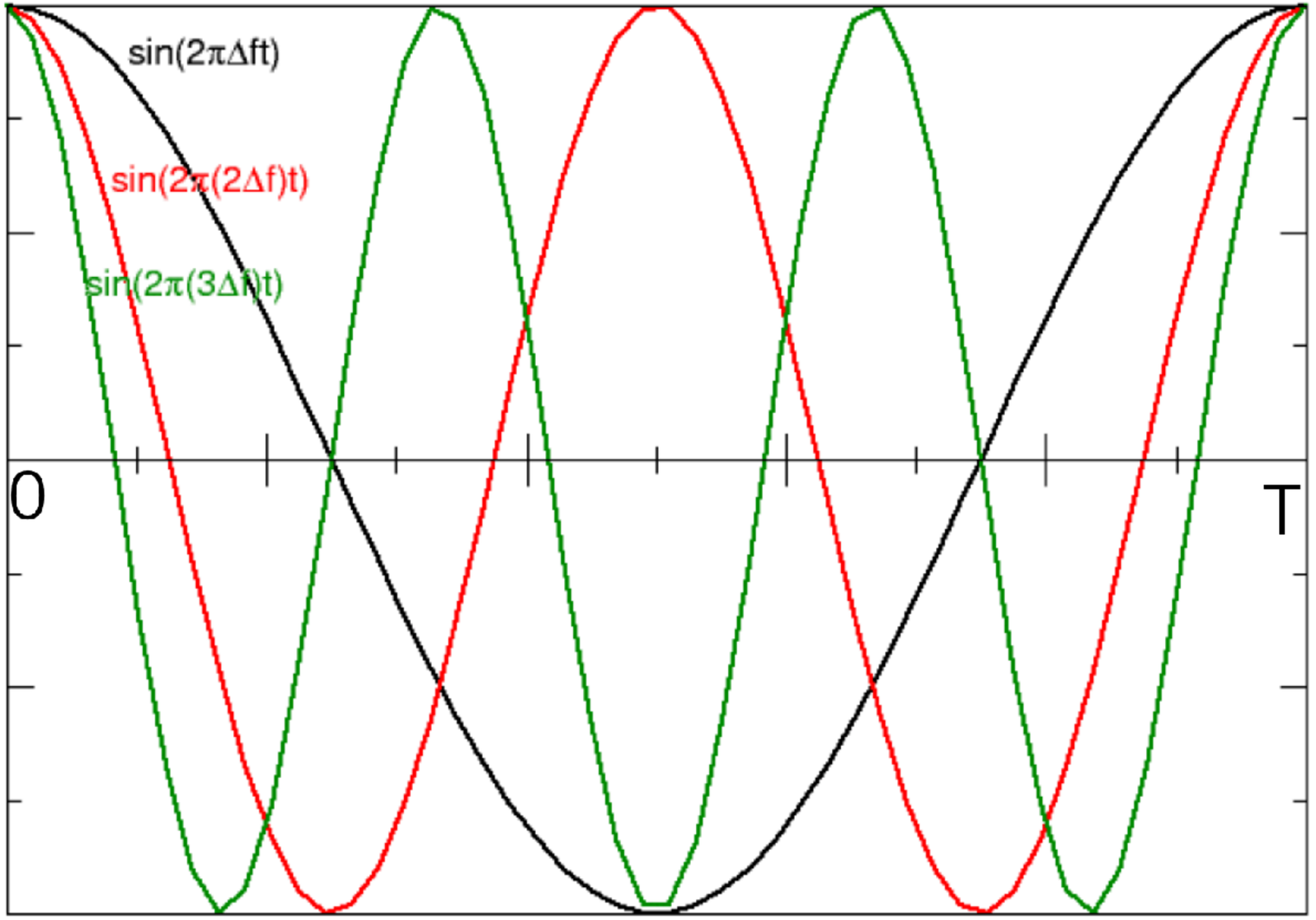


one period in interval

two periods in interval

three periods in interval

the signal under study is periodic



one period in interval

two periods in interval

three periods in interval

smallest frequency: $\Delta f = \frac{1}{T}$

$f_n = n\Delta f$

data sampling

Fourier analysis Fourier analysis spectral power

errors in analysis

linear filters

time-frequency analysis

Fourier series

If a signal s(t) is periodic and continuous in time, then it may be expressed as

Fourier series

If a signal s(t) is periodic and continuous in time, then it may be expressed as

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{N} a_n \cos(2\pi f_n t) + b_n \sin(2\pi f_n t) \qquad N \to \infty$$

$$a_n, \;\; b_n \in \mathbb{R}$$ (a_n, b_n are real)

Fourier series

If a signal s(t) is periodic and continuous in time, then it may be expressed as

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{N} a_n \cos(2\pi f_n t) + b_n \sin(2\pi f_n t) \qquad N \to \infty$$

$a_n, \; b_n \in \mathbb{R}$ (a_n, b_n are real)

Fourier analysis describes a signal as **a sum of periodic sinusoidal functions**

Fourier series

If a signal s(t) is periodic and continuous in time, then it may be expressed as

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{N} a_n \cos(2\pi f_n t) + b_n \sin(2\pi f_n t) \qquad N \to \infty$$

$a_n, \; b_n \in \mathbb{R}$ (a_n, b_n are real)

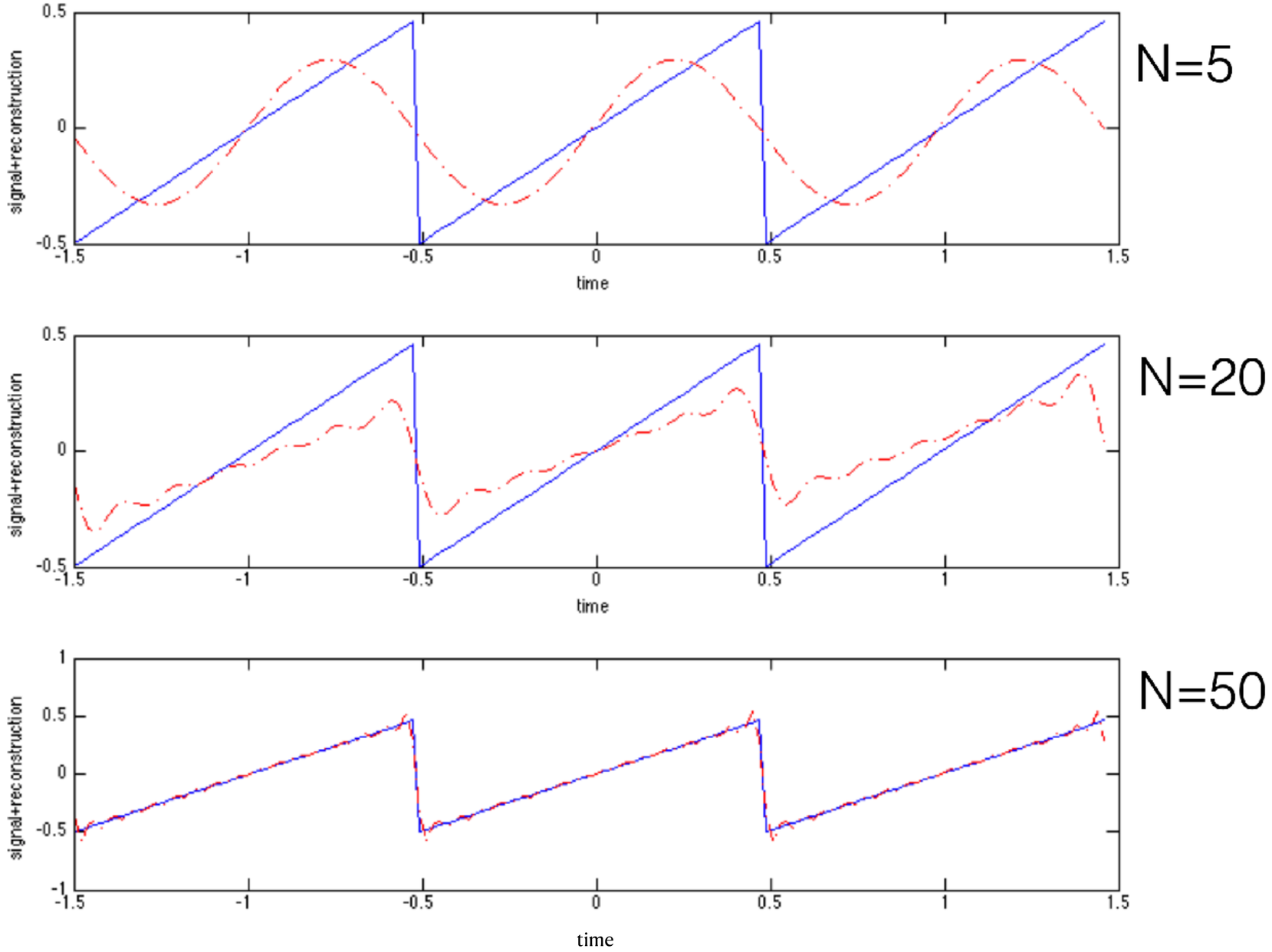
Fourier analysis describes a signal as **a sum of periodic sinusoidal functions**

equivalent formulation: $$s(t) \approx \sum_{n=-N}^{N} c_n e^{i2\pi f_n t} \qquad c_n \in \mathbb{C}$$ (c_n are complex)

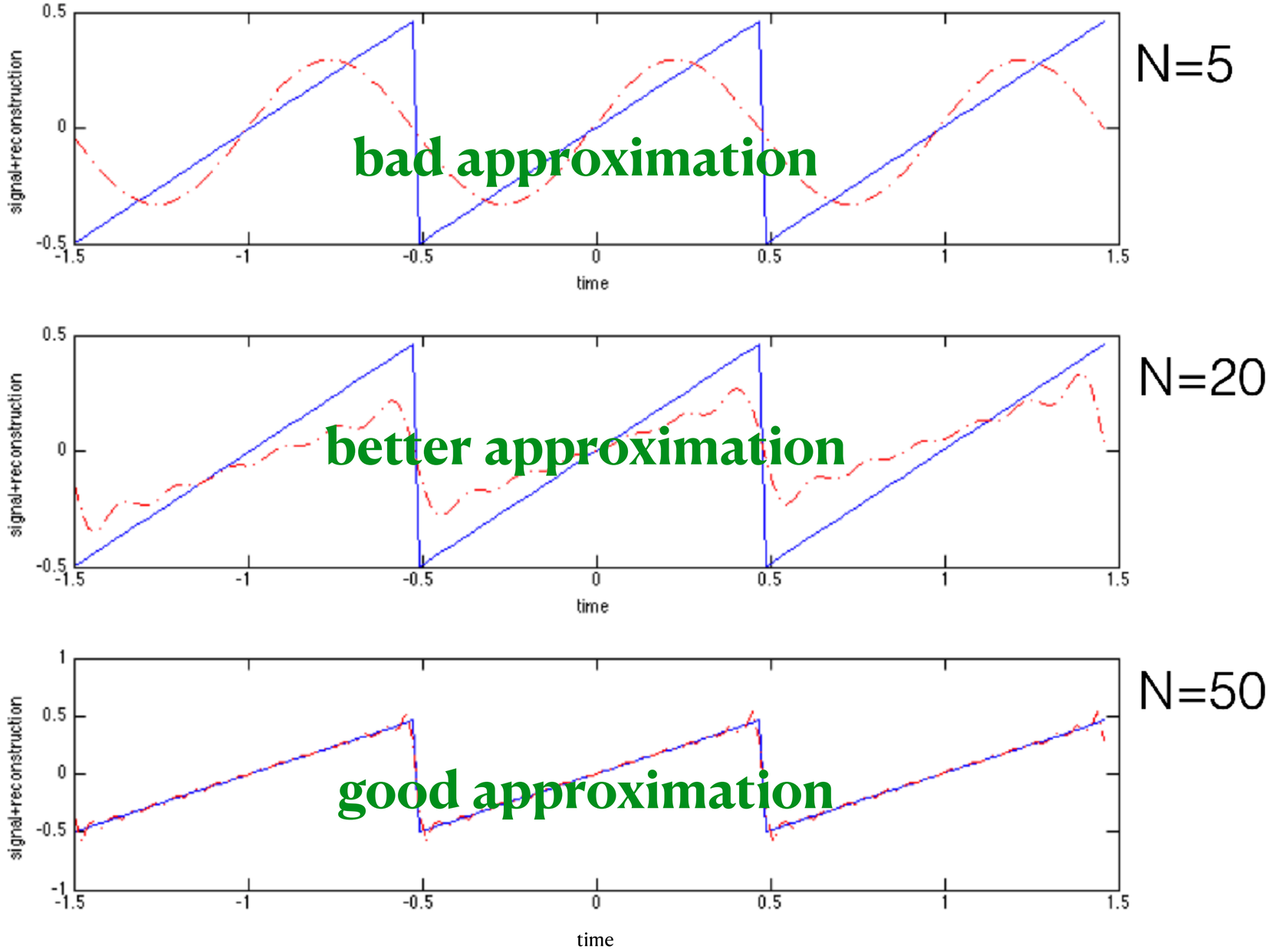
N to be chosen

approximation by different number of Fourier modes

approximation by different number of Fourier modes N



approximation by different number of Fourier modes N



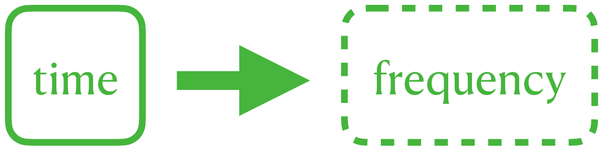
coefficients a_n, b_n or c_n estimates the contribution of Fourier mode n to signal

coefficients a_n, b_n or c_n estimates the contribution of Fourier mode n to signal

If signal is sampled with $t \to t_k = k\Delta t = k/f_s$ with N sample points

coefficients a_n, b_n or c_n estimates the contribution of Fourier mode n to signal

If signal is sampled with $t \to t_k = k\Delta t = k/f_s$ with N sample points

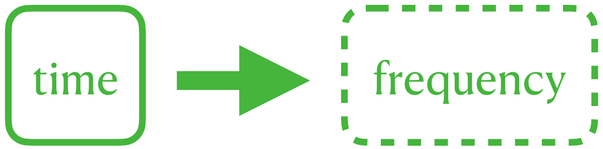


$$c_n = \sum_{k=1}^{N} s(t_k) e^{-i2\pi nk/N}$$

Discrete Fourier Transform (DFT)

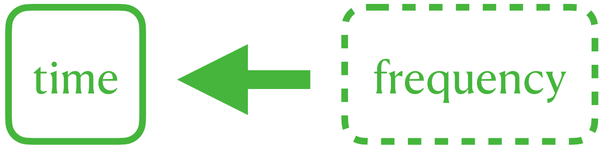
coefficients a_n, b_n or c_n estimates the contribution of Fourier mode n to signal

If signal is sampled with $t \to t_k = k\Delta t = k/f_s$ with N sample points



$$c_n = \sum_{k=1}^{N} s(t_k) e^{-i2\pi nk/N}$$

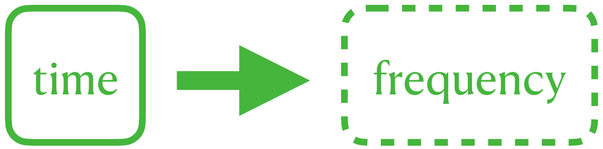
Discrete Fourier Transform (DFT)



$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

coefficients a_n, b_n or c_n estimates the contribution of Fourier mode n to signal

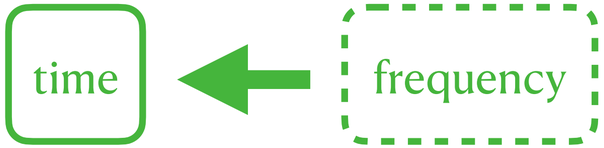
If signal is sampled with $t \to t_k = k\Delta t = k/f_s$ with N sample points



$$c_n = \sum_{k=1}^{N} s(t_k) e^{-i2\pi nk/N}$$

$$f_n = n\Delta f = n/T$$

Discrete Fourier Transform (DFT)



$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

data sampling

Fourier analysis Fourier analysis spectral power

errors in analysis

linear filters

time-frequency analysis

$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

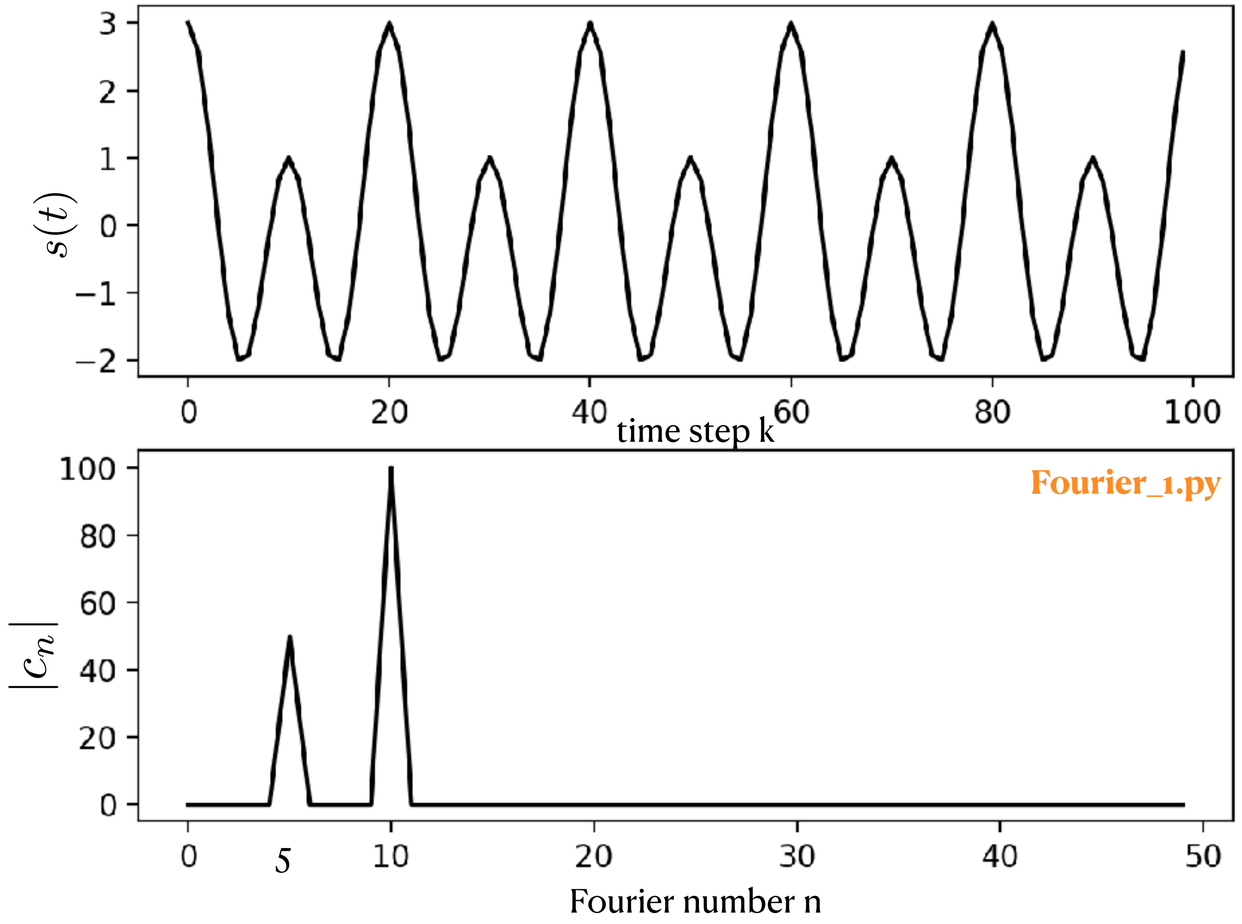
$$c_{-n} = c_n^* \qquad |c_{-n}| = |c_n|$$

$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

$$c_{-n} = c_n^* \qquad |c_{-n}| = |c_n|$$

$$s(t) = \cos(2\pi f_1 t) + 2\cos(2\pi f_2 t)$$

$$f_1 = 0.5Hz \ , \ f_2 = 1.0Hz$$



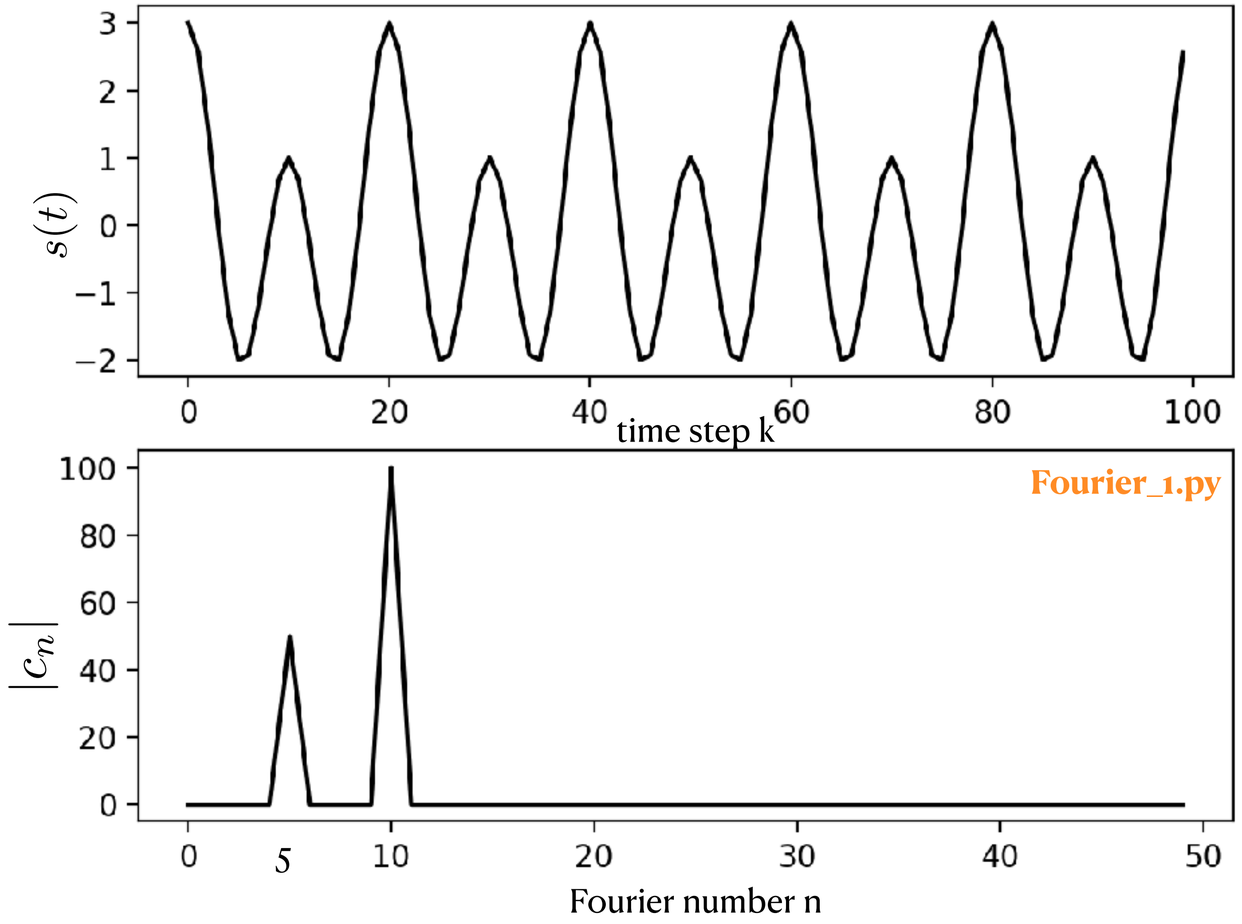
$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

$$c_{-n} = c_n^* \qquad |c_{-n}| = |c_n|$$

$$s(t) = \cos(2\pi f_1 t) + 2\cos(2\pi f_2 t)$$

$$f_1 = 0.5Hz \ , \ f_2 = 1.0Hz$$

- duration T=10s
- f_s=10Hz



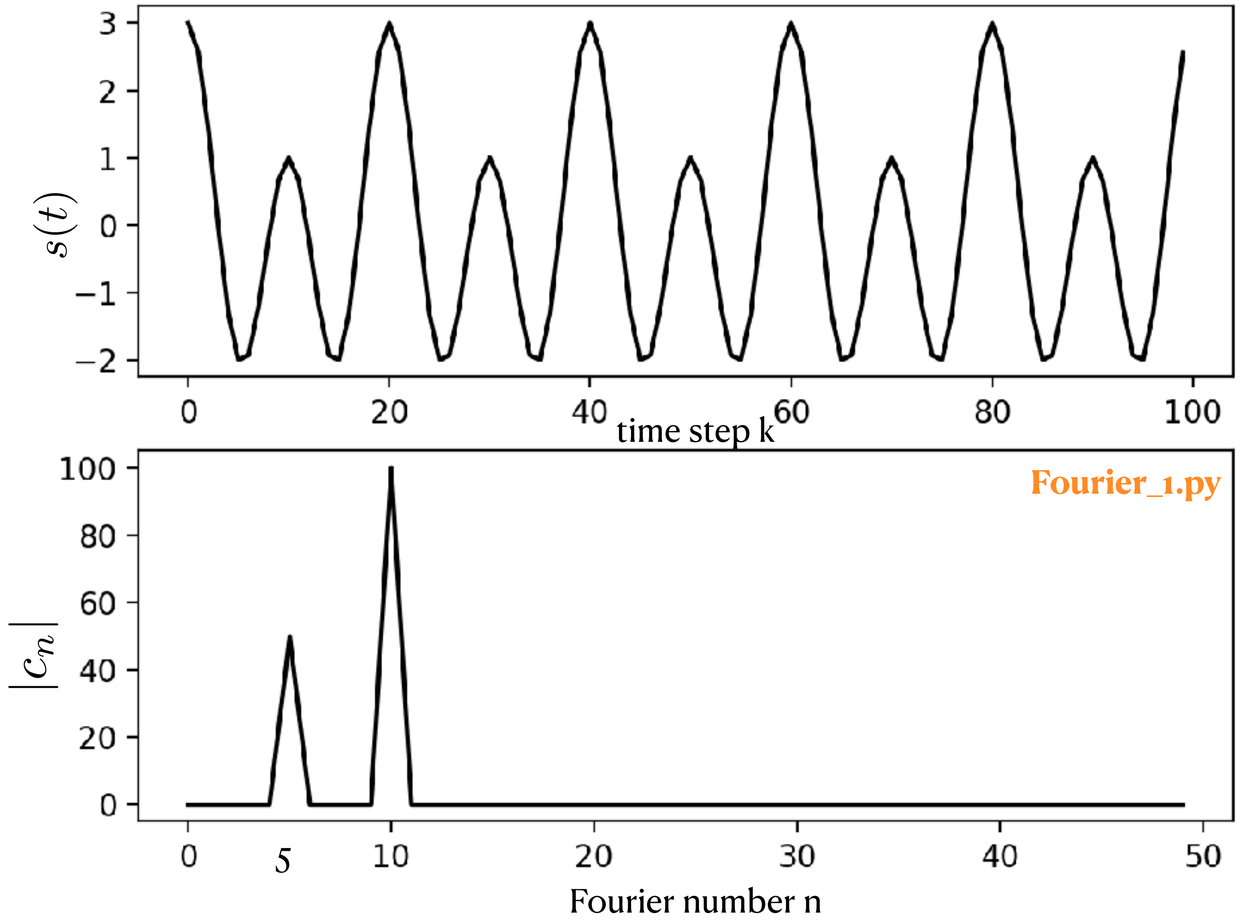
$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

$$c_{-n} = c_n^* \qquad |c_{-n}| = |c_n|$$

$$s(t) = \cos(2\pi f_1 t) + 2\cos(2\pi f_2 t)$$

$$f_1 = 0.5Hz \ , \ f_2 = 1.0Hz$$

- duration T=10s $\Delta f = 0.1Hz$
- f_s=10Hz $\Delta t = 0.1s$



$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

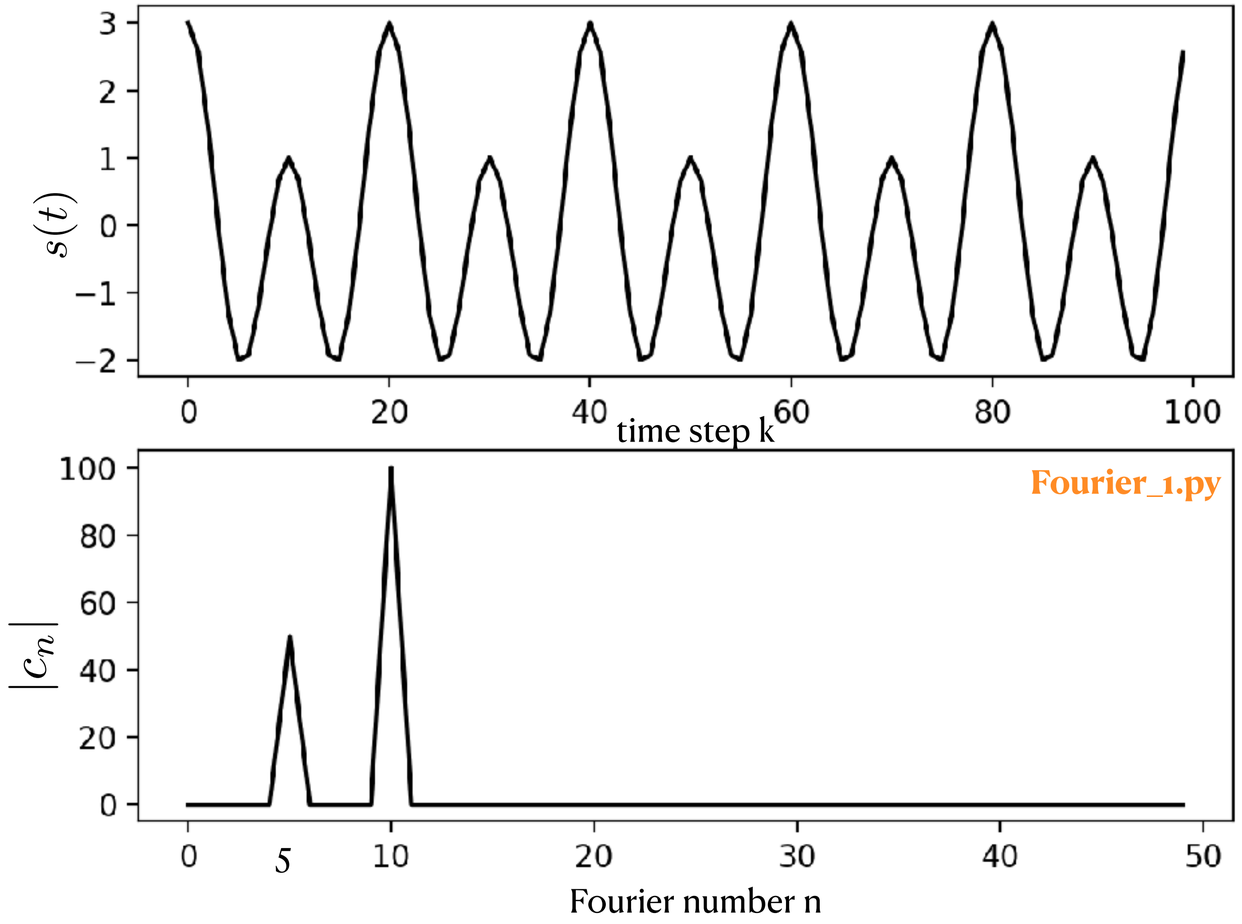
$$c_{-n} = c_n^* \qquad |c_{-n}| = |c_n|$$

$$s(t) = \cos(2\pi f_1 t) + 2\cos(2\pi f_2 t)$$

$$f_1 = 0.5Hz \ , \ f_2 = 1.0Hz$$

- duration T=10s $\Delta f = 0.1Hz$
- f_s=10Hz $\Delta t = 0.1s$

$$N = 100$$



$$s(t_k) = \frac{1}{N} \sum_{n=-N/2}^{N/2} c_n e^{i2\pi nk/N}$$

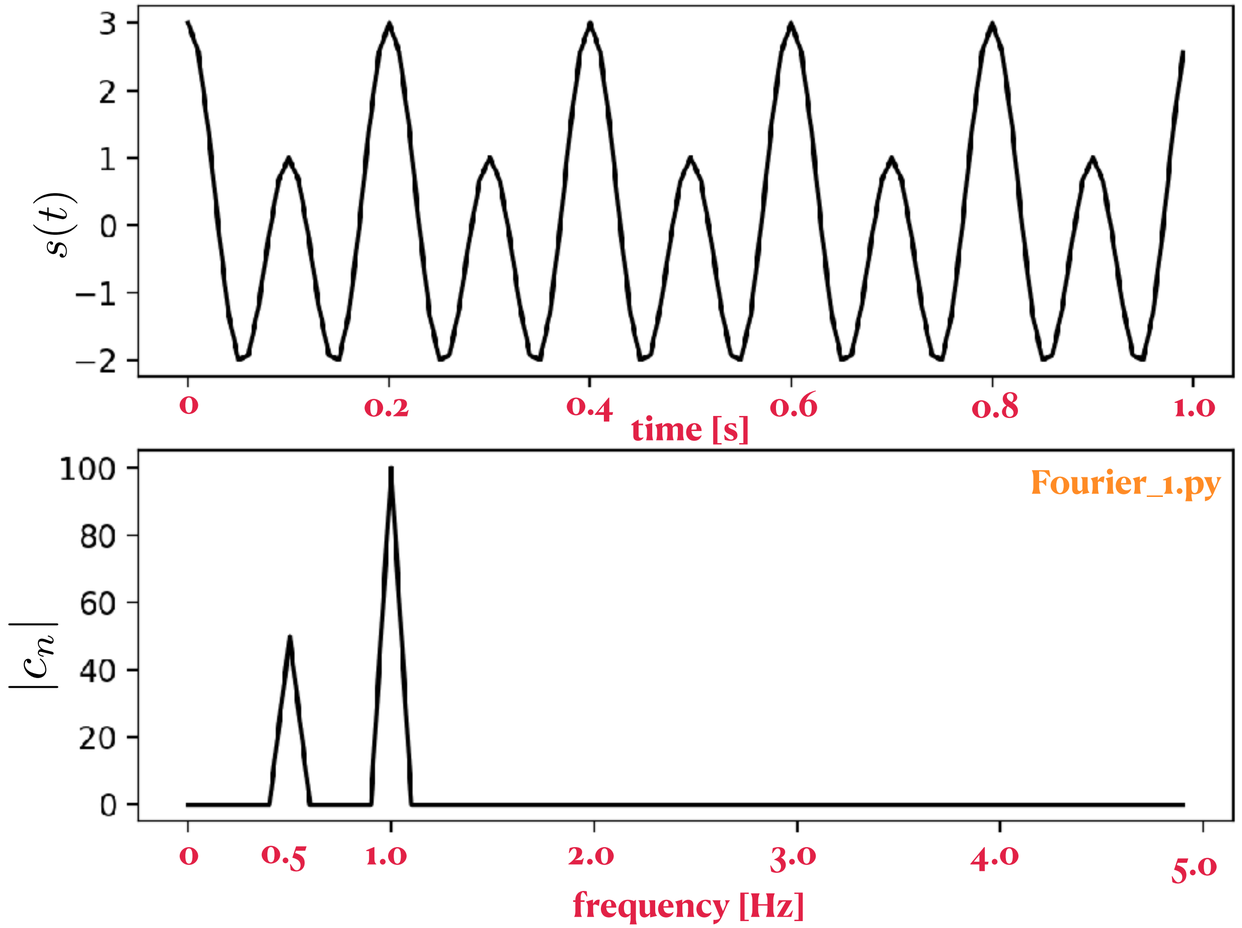
$$c_{-n} = c_n^* \qquad |c_{-n}| = |c_n|$$

$$s(t) = \cos(2\pi f_1 t) + 2\cos(2\pi f_2 t)$$

$$f_1 = 0.5Hz \ , \ f_2 = 1.0Hz$$

- duration T=10s $\quad \Delta f = 0.1Hz$
- f_s=10Hz $\quad \Delta t = 0.1s$

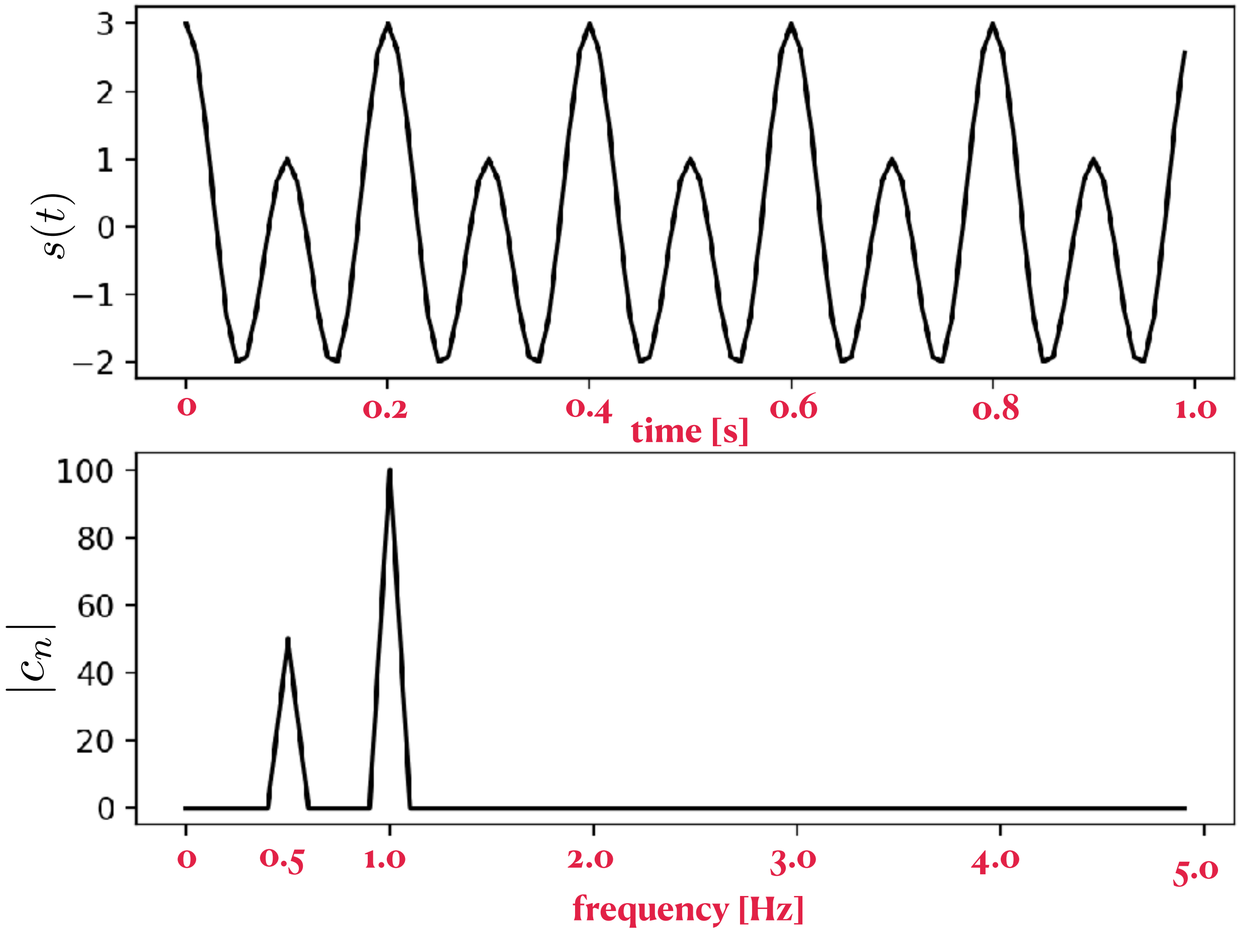
$$N = 100$$



We note:

- for the spectral power it is sufficient to consider

$$0 \leq f \leq f_s/2$$



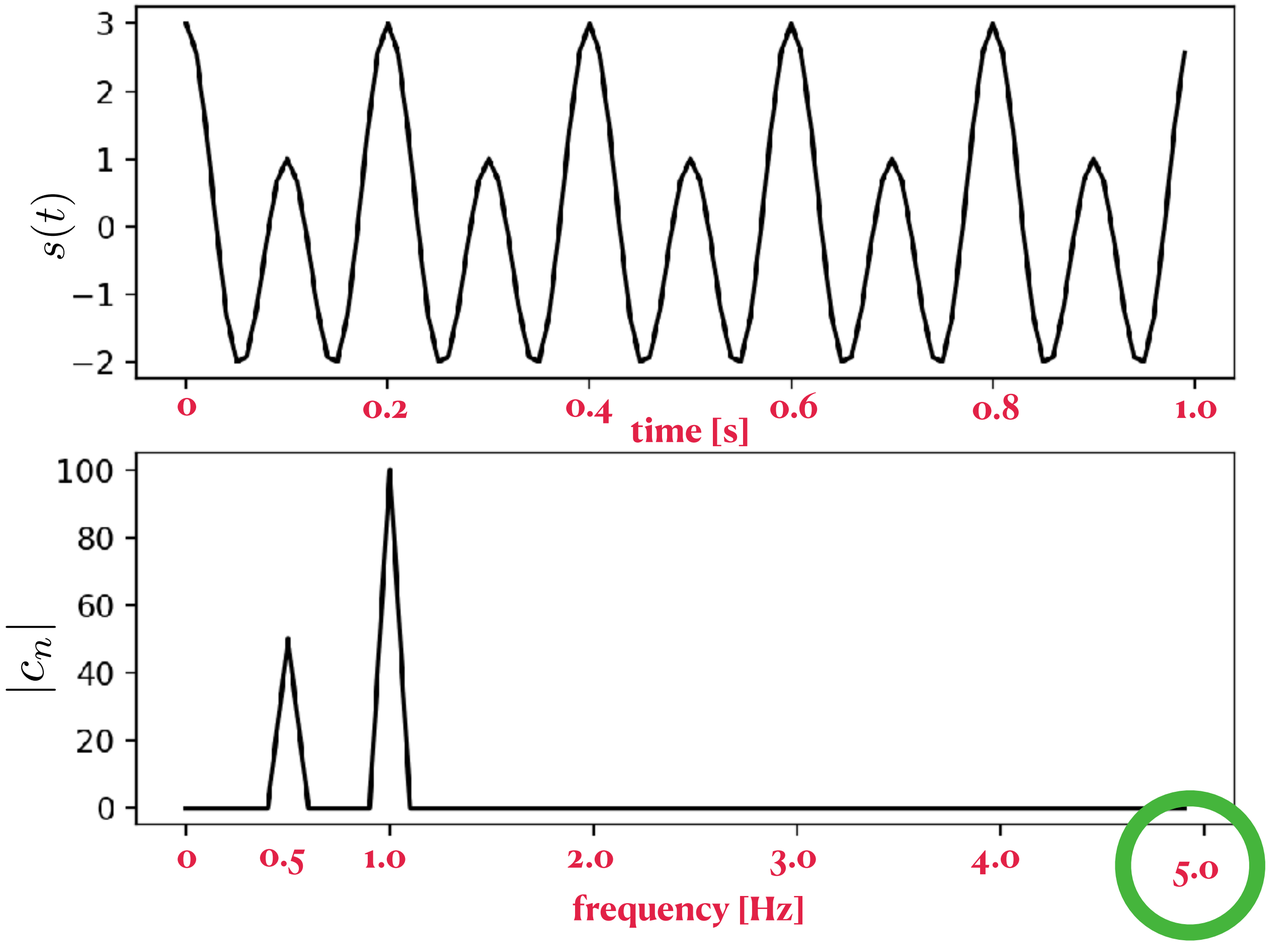
We note:

- for the spectral power it is sufficient to consider

$$0 \leq f \leq f_s/2$$

- maximum frequency of DFT is

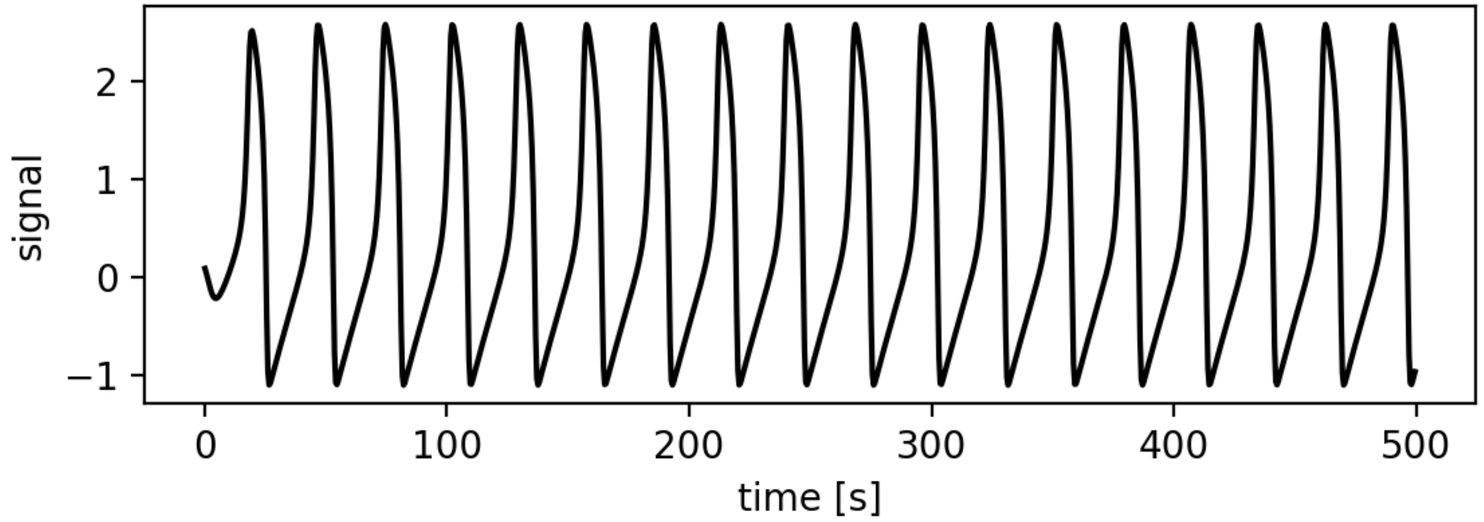
Nyquist frequency $f_s/2$

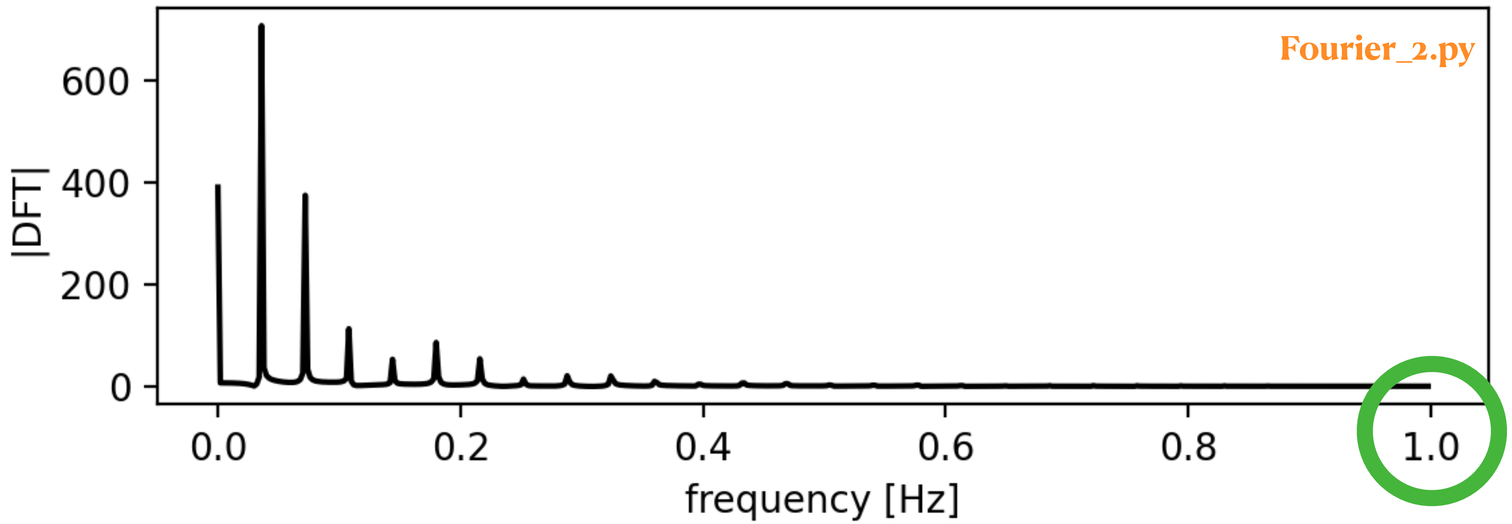


example signal: FitzHugh-Nagumo model activity

- T = 500s $\Delta f = 0.002Hz$
- f_s = 2Hz

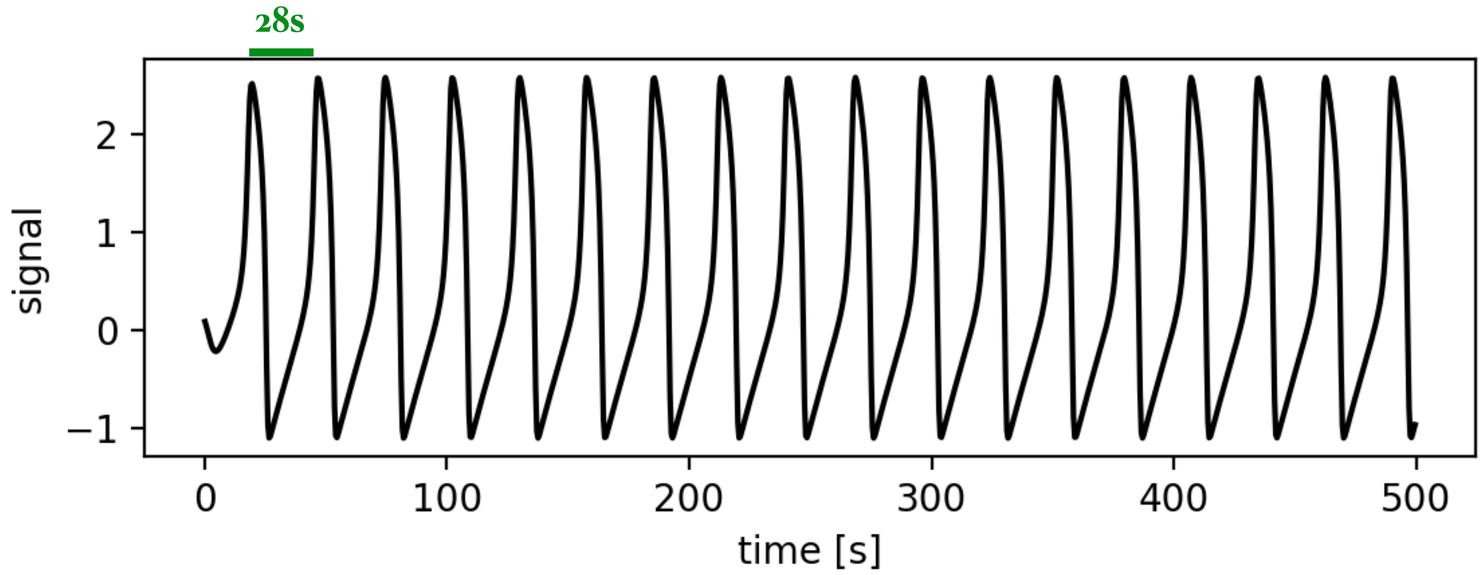
Nyquist frequency is 1Hz
= maximum frequency





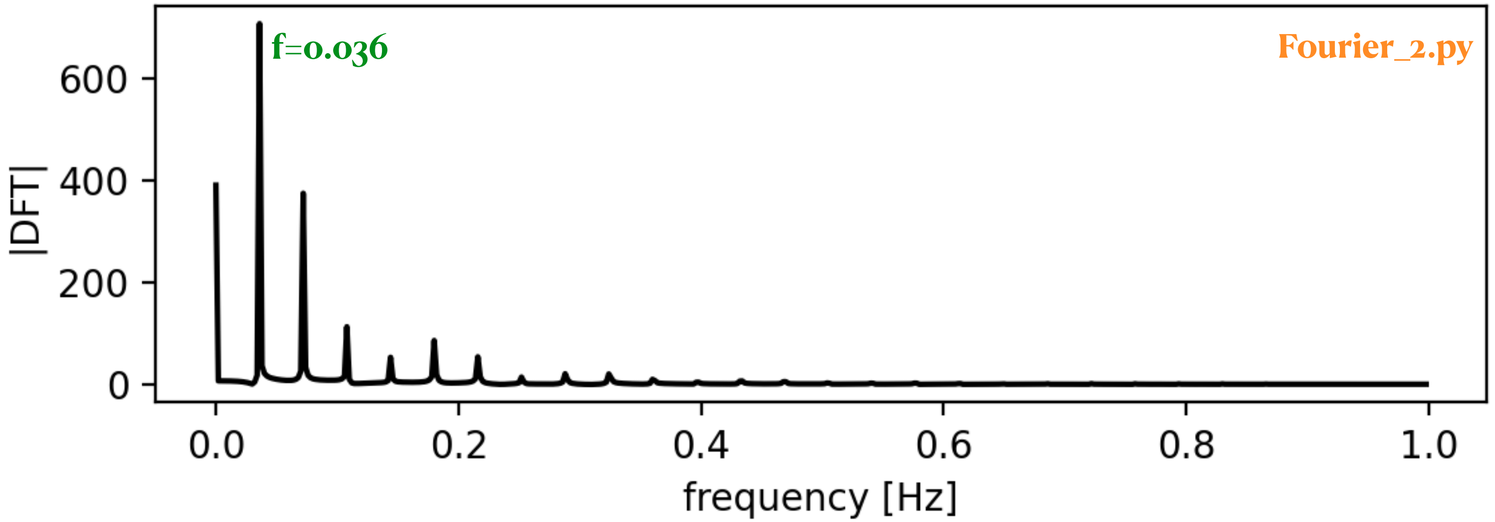
example signal: FitzHugh-Nagumo model activity

- T = 500s $\quad \Delta f = 0.002Hz$
- f_s = 2Hz



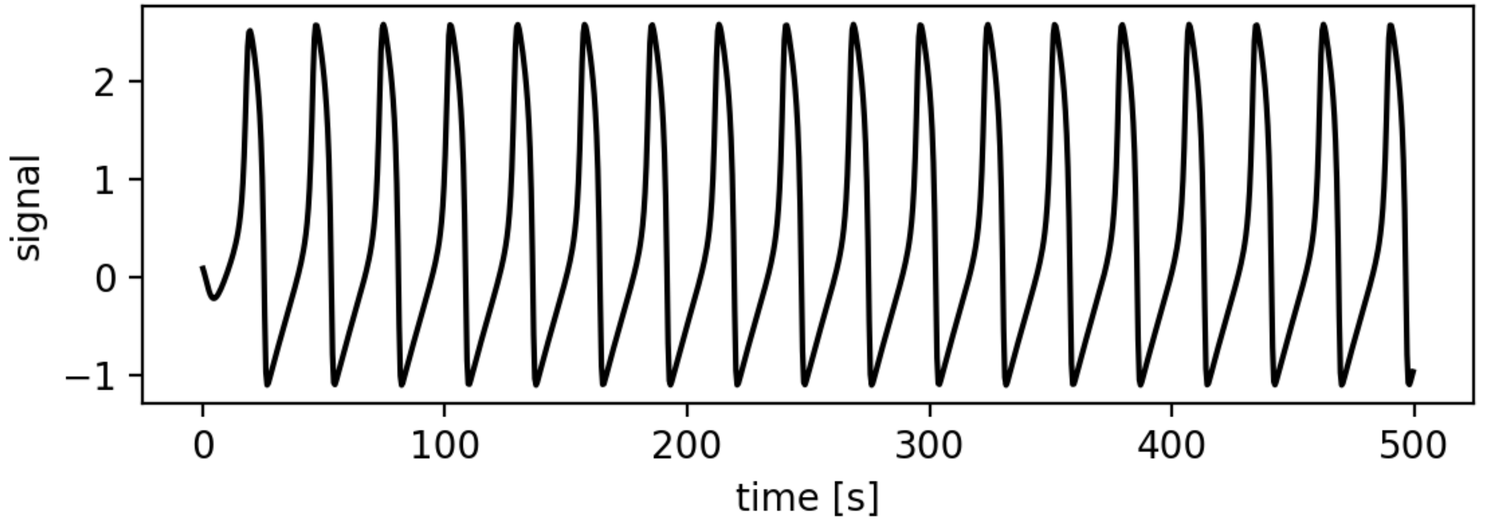
Nyquist frequency is 1Hz
= maximum frequency

principal data frequency
≈ highest power frequency



example signal: FitzHugh-Nagumo model activity

- T = 500s $\Delta f = 0.002Hz$
- f_s = 2Hz



Nyquist frequency is 1Hz
= maximum frequency

principal data frequency
≈ highest power frequency

DFT has multiple power peaks

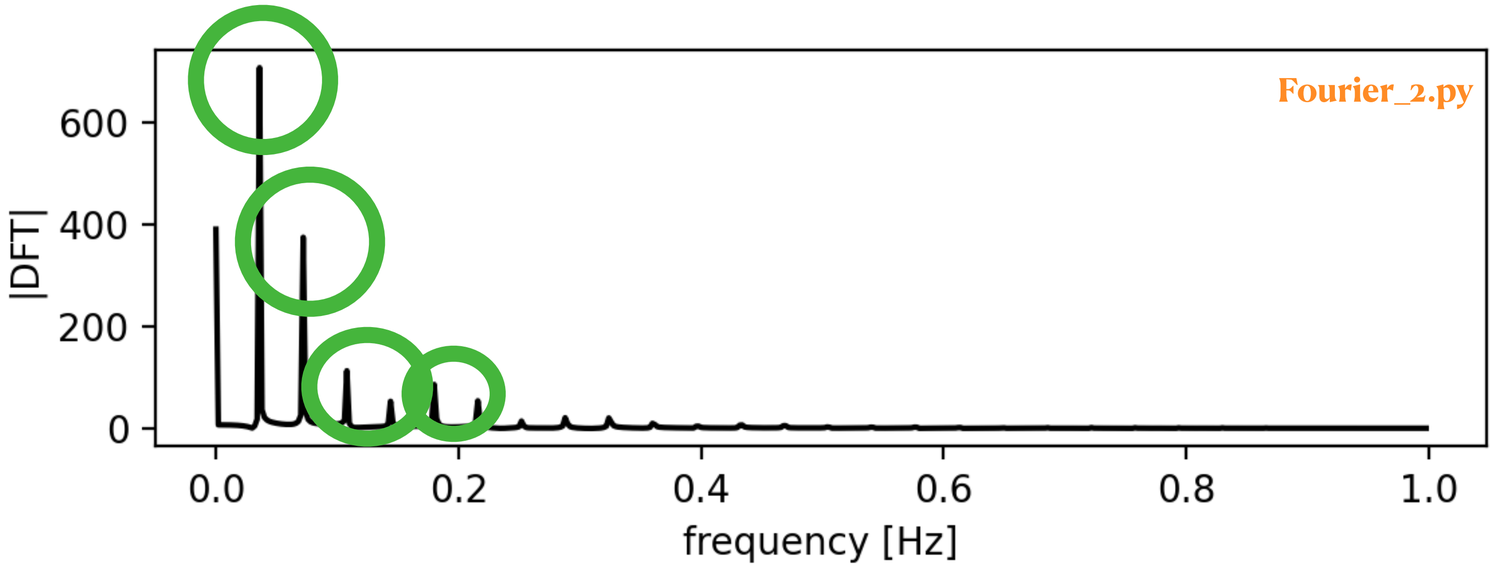
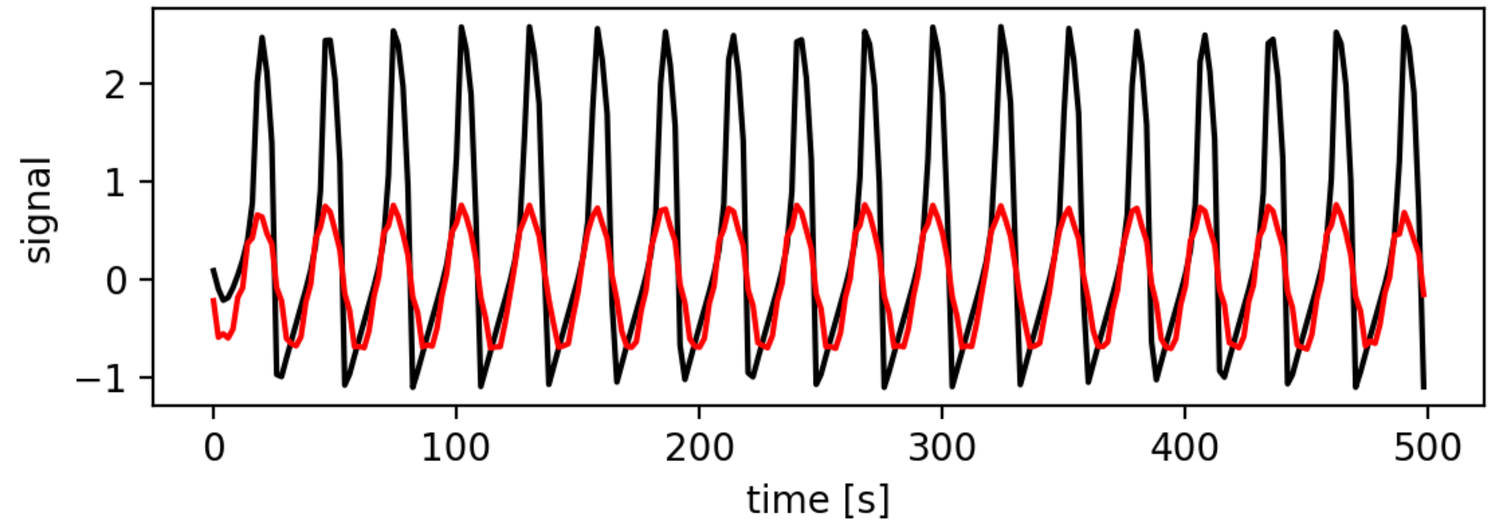
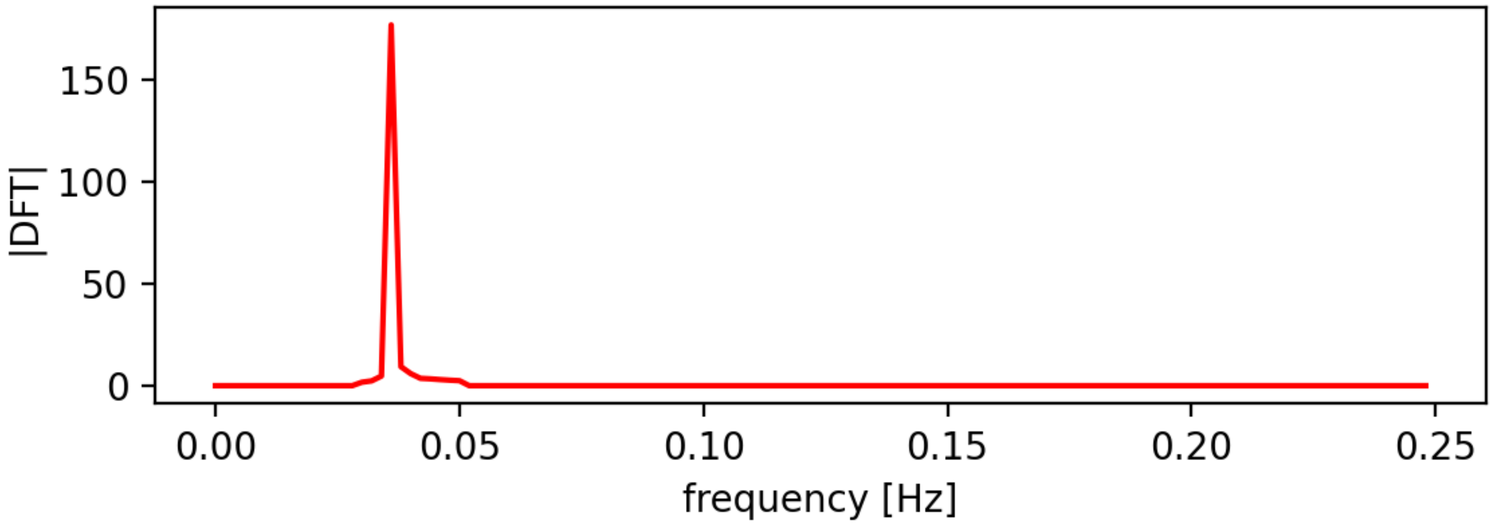
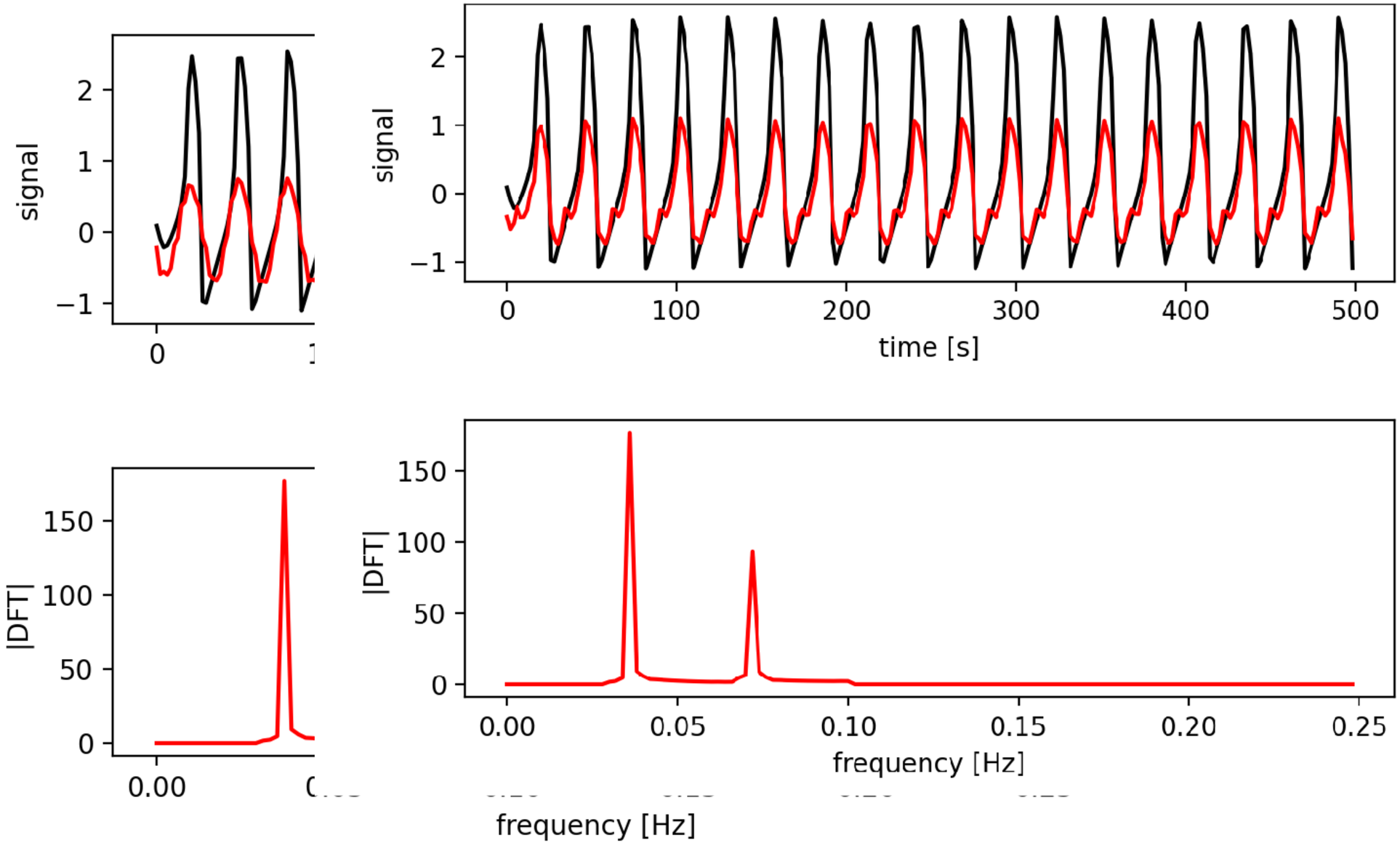
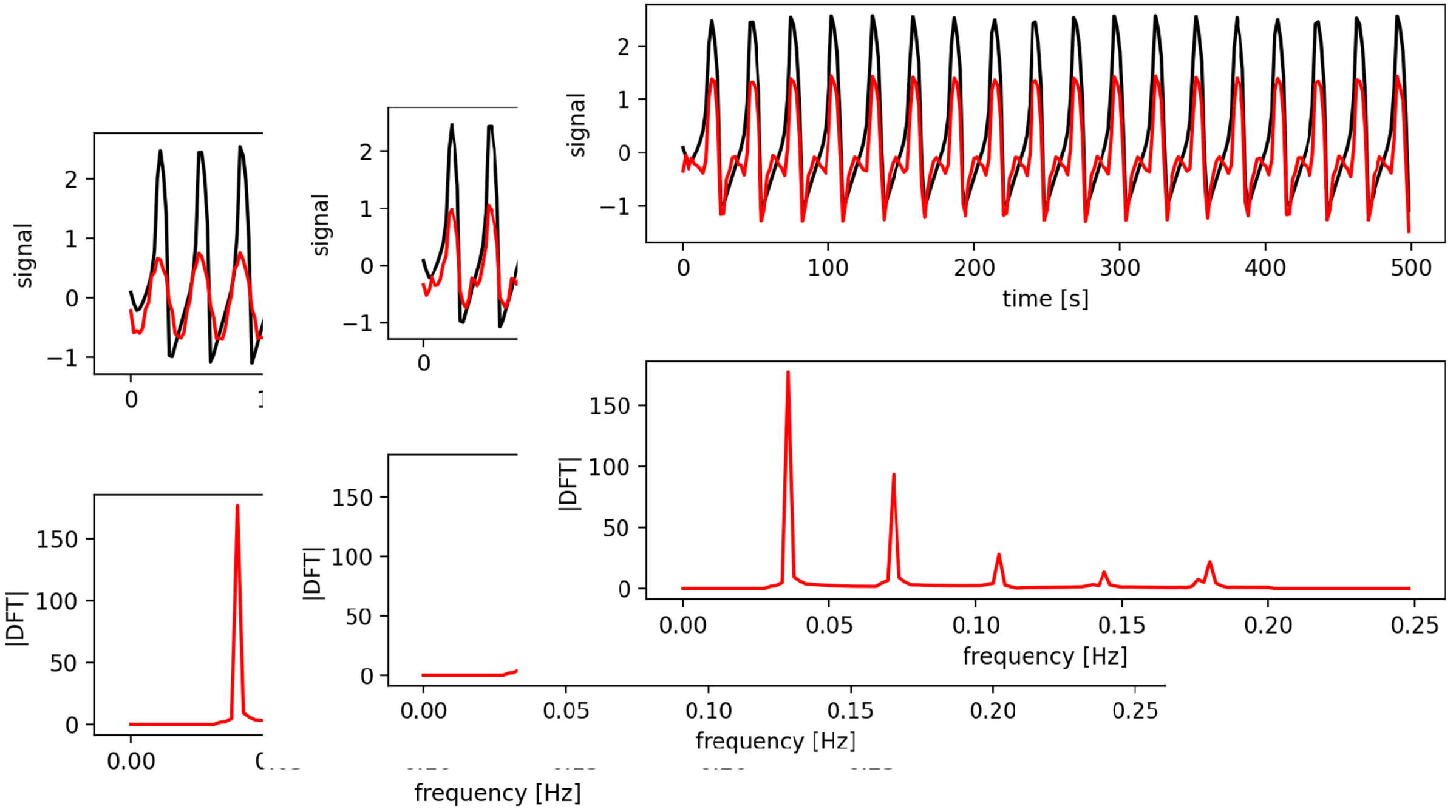


illustration for spectral decomposition (Fourier_2.py)






signal
2
1
0
−1
0
signal
2
1
0
−1
0
100
200
300
400
500
time [s]
|DFT|
150
100
50
0
0.00
|DFT|
150
100
50
0
0.00
0.05
0.10
0.15
0.20
0.25
frequency [Hz]
frequency [Hz]


signal
time [s]
|DFT|
frequency [Hz]

data sampling

Fourier analysis

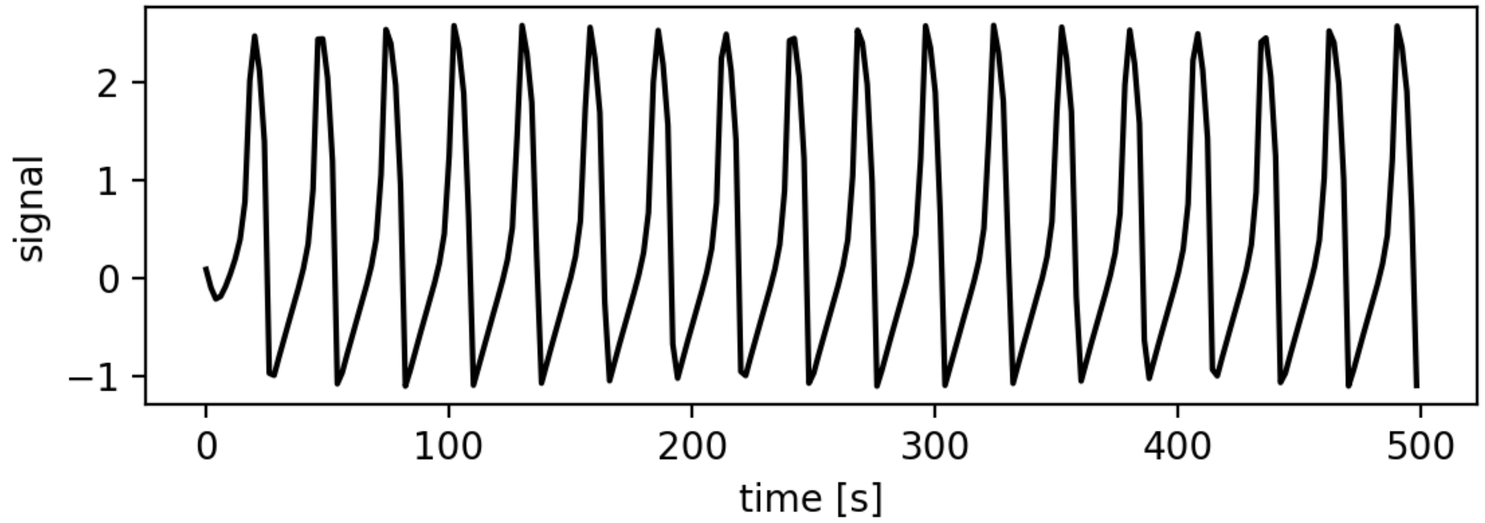
errors in analysis

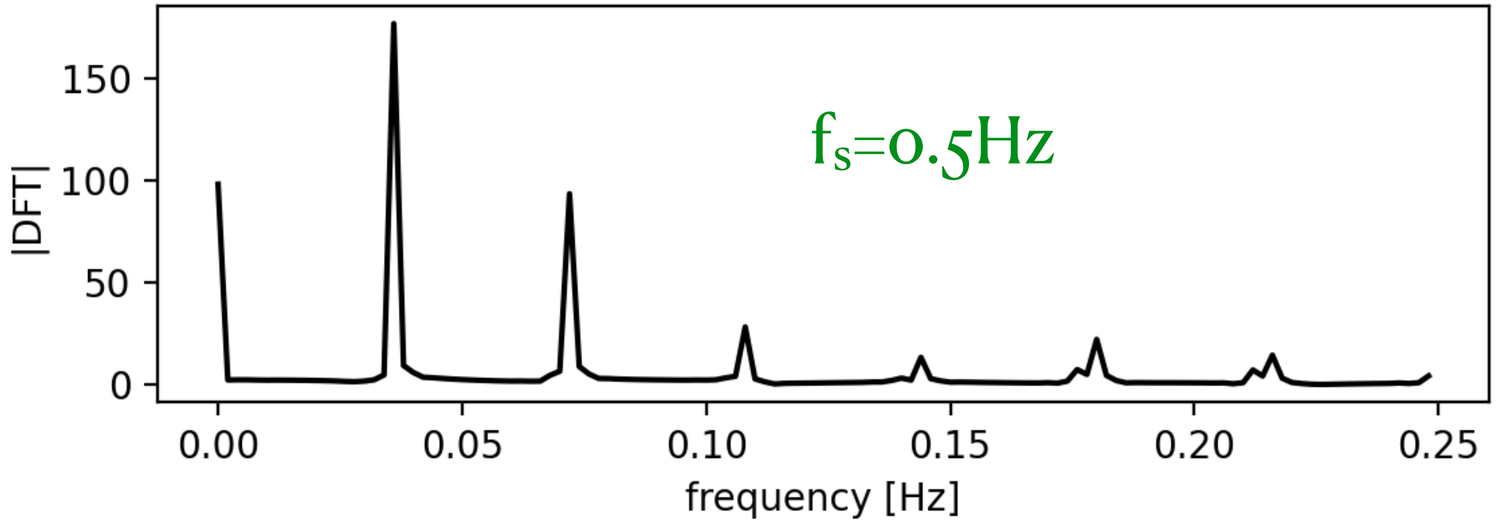
linear filters

time-frequency analysis

Error: subsampling

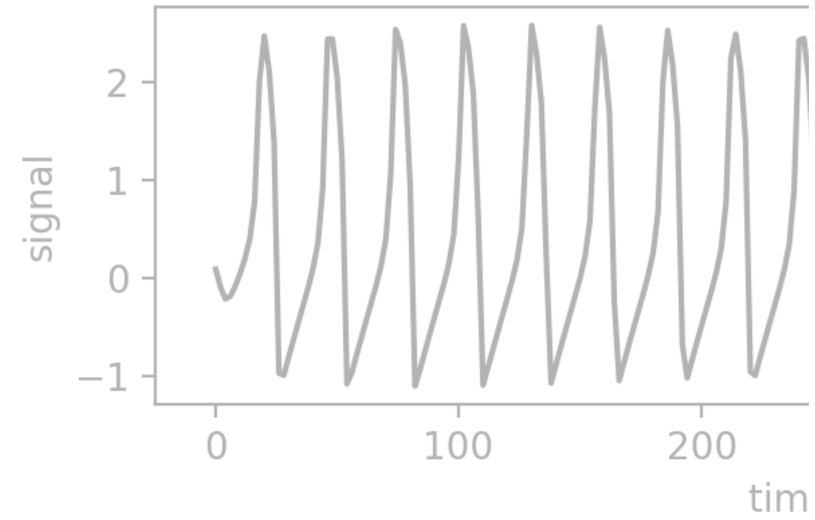
SamplingError_1.py

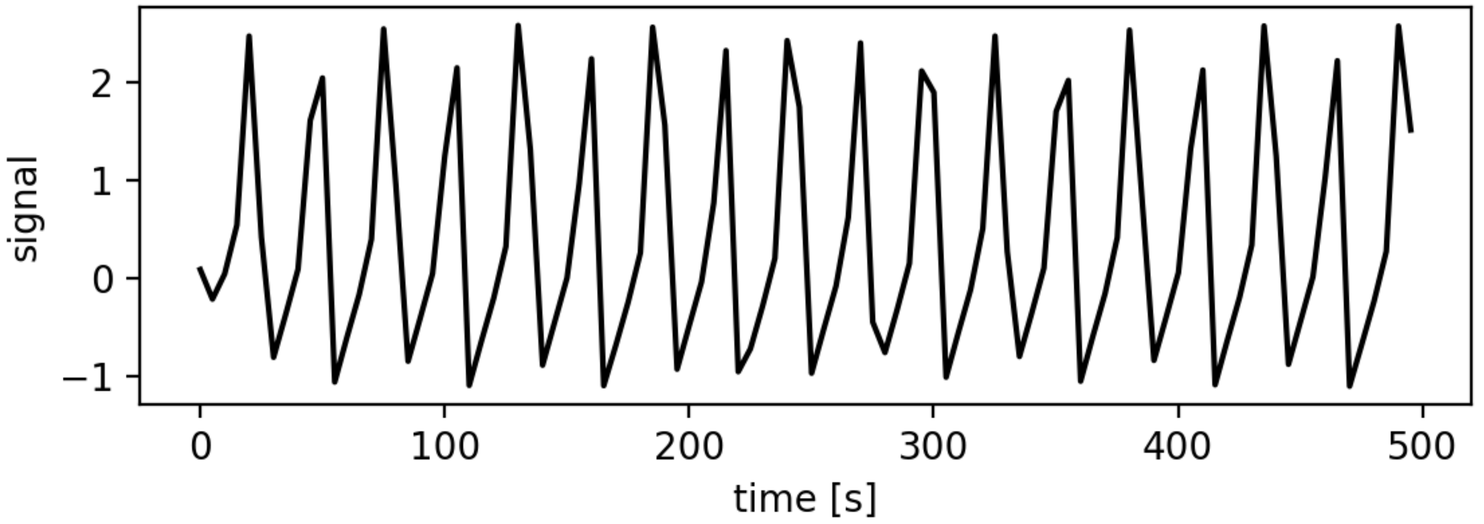


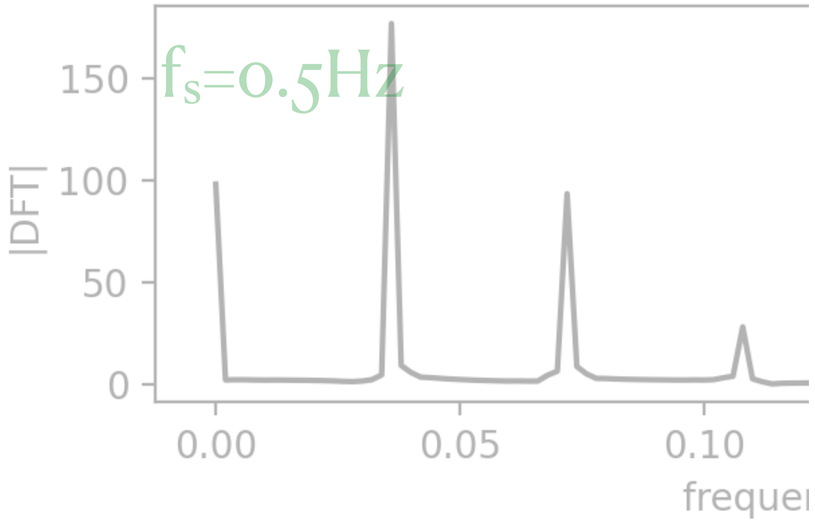


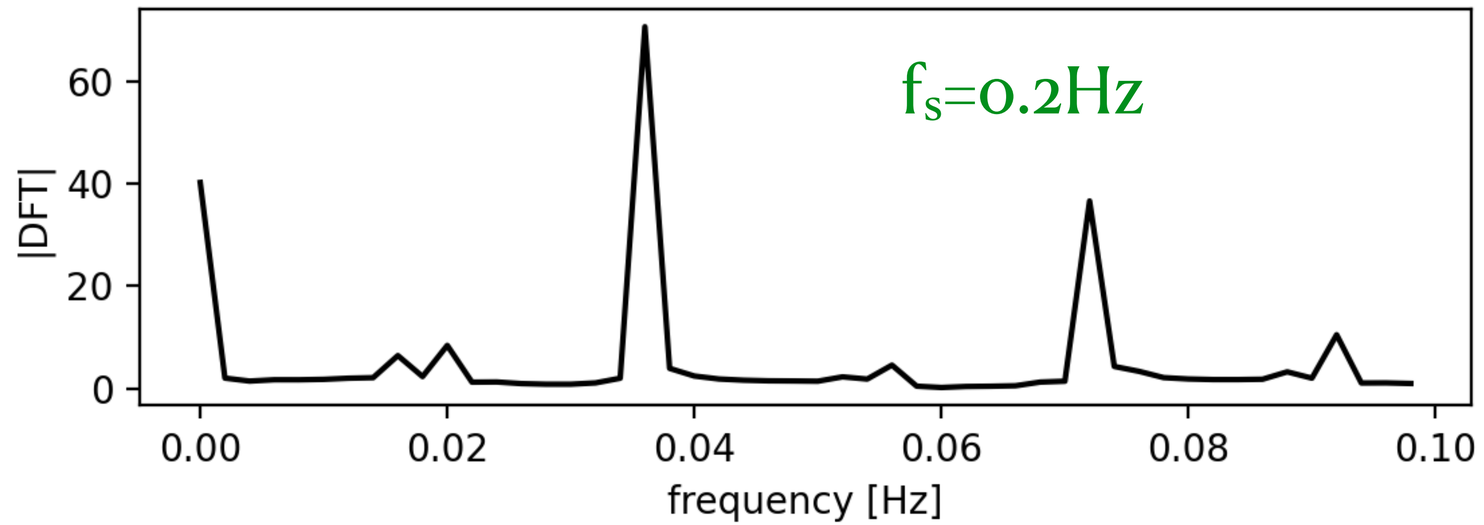
f_s=0.5Hz

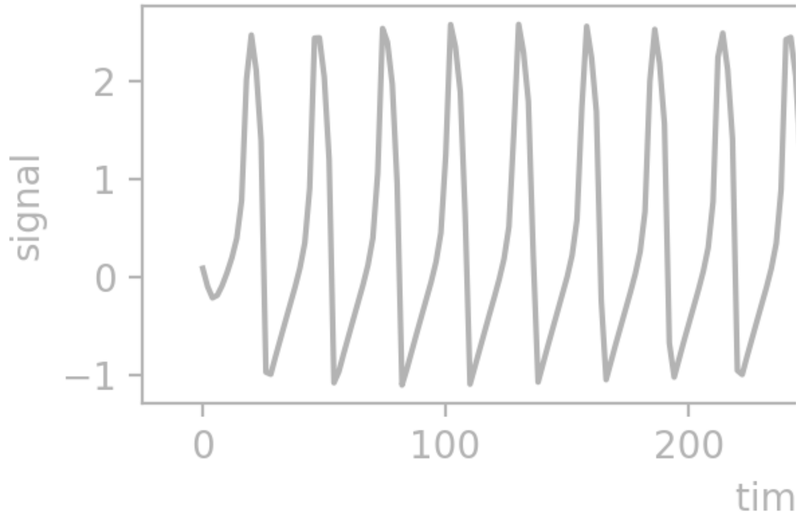
SamplingError_1.py

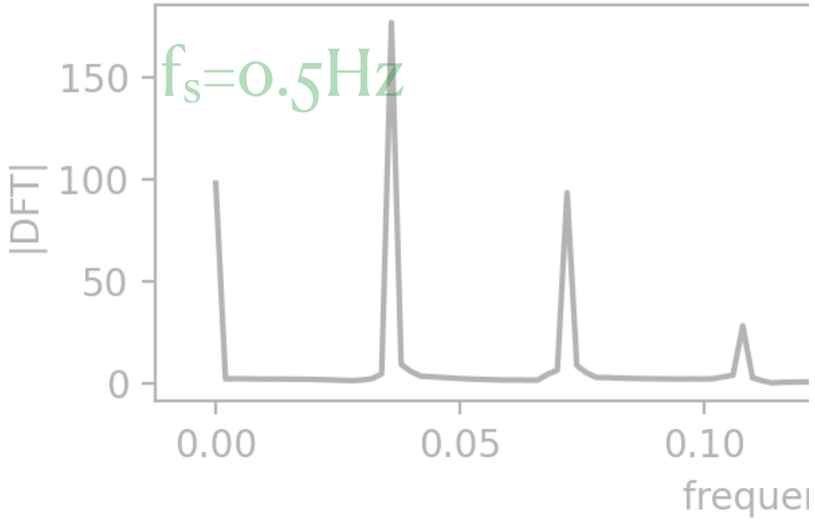




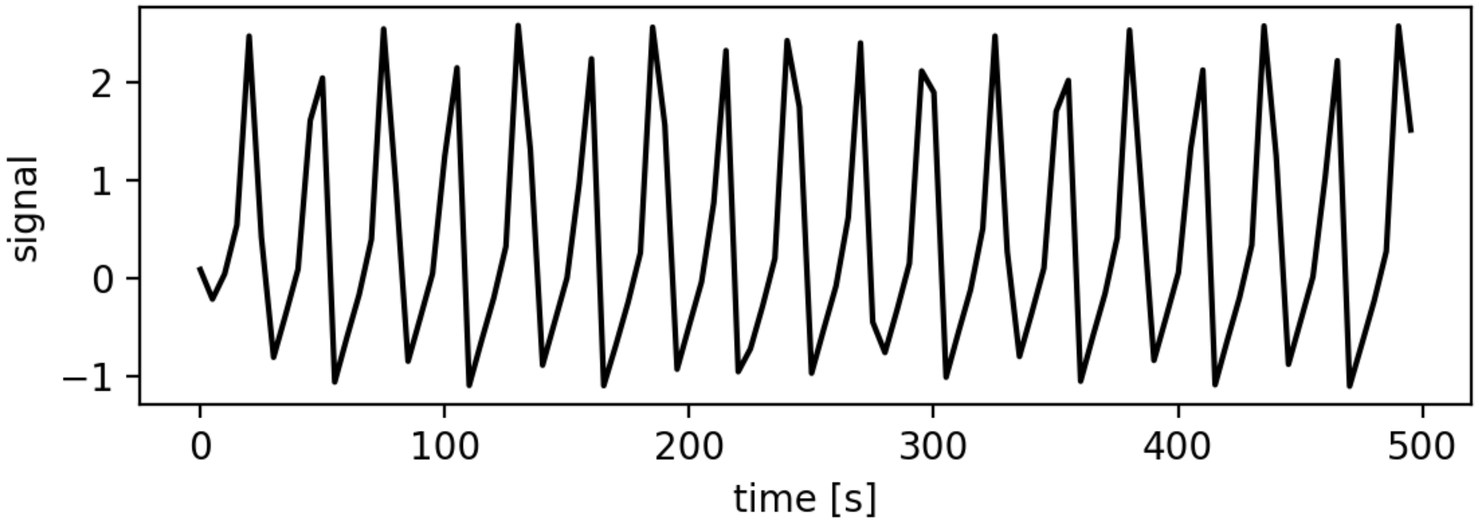


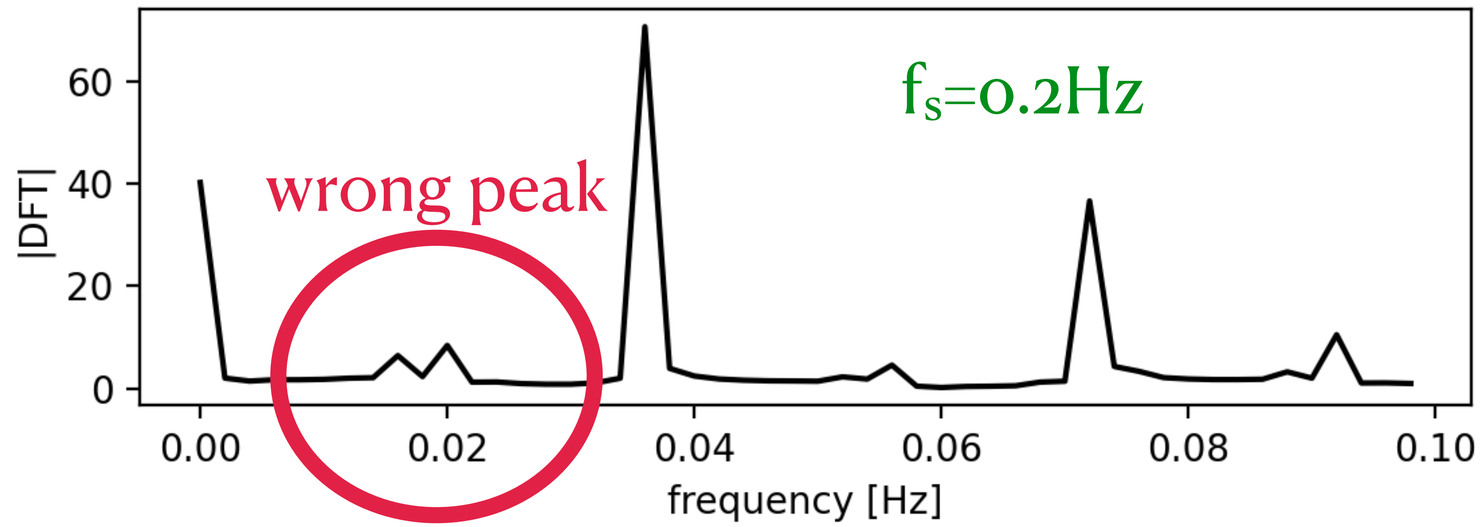






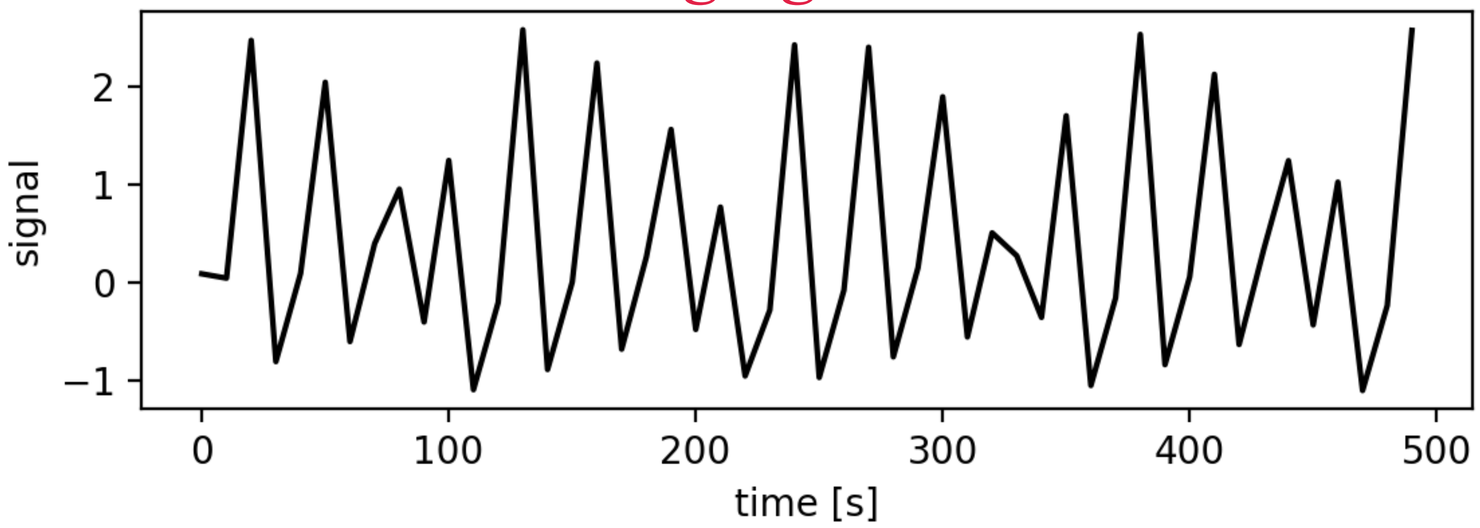
SamplingError_1.py

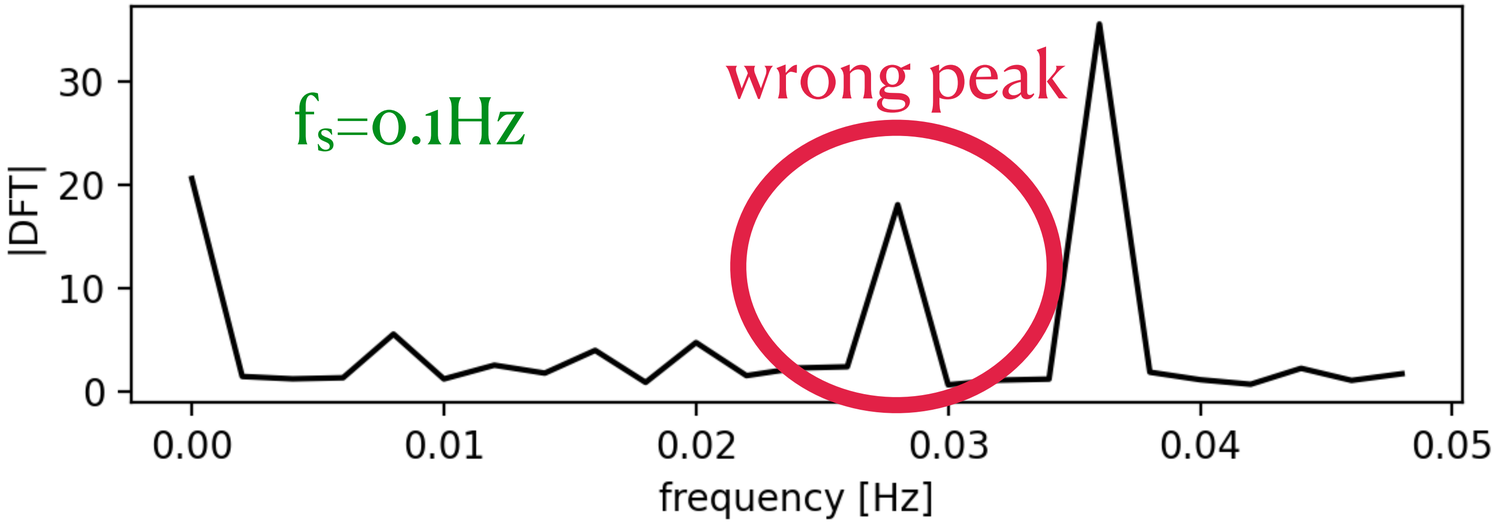


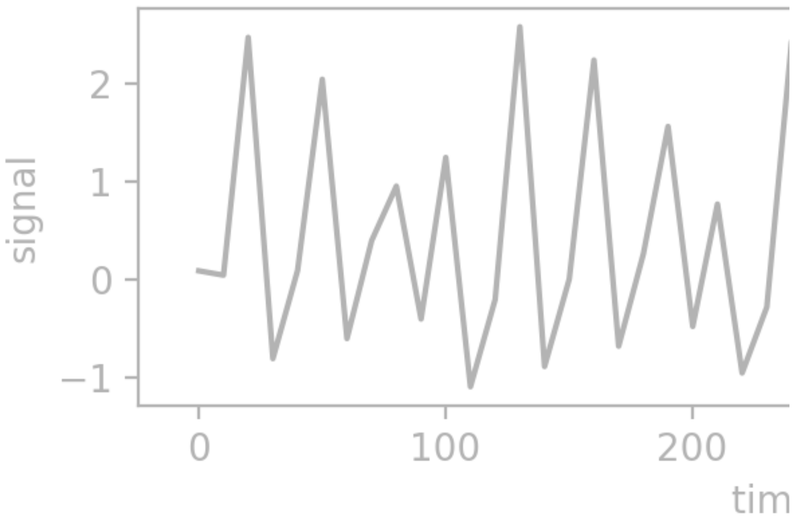


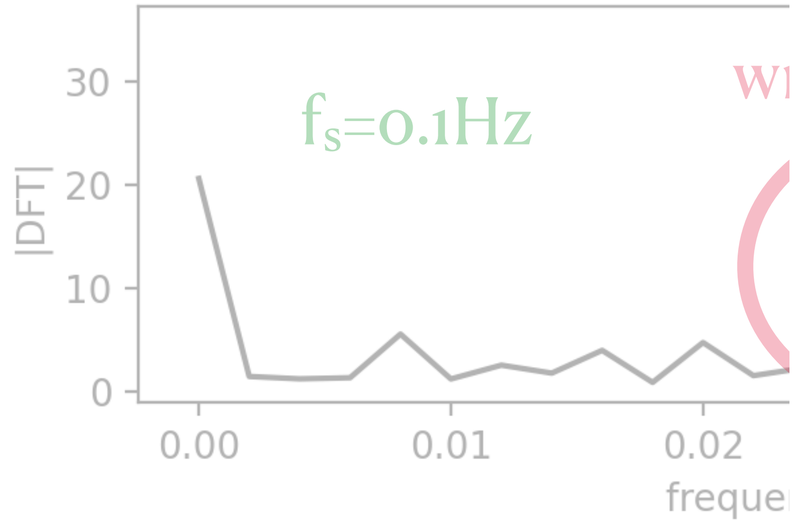
wrong signal

SamplingError_1.py





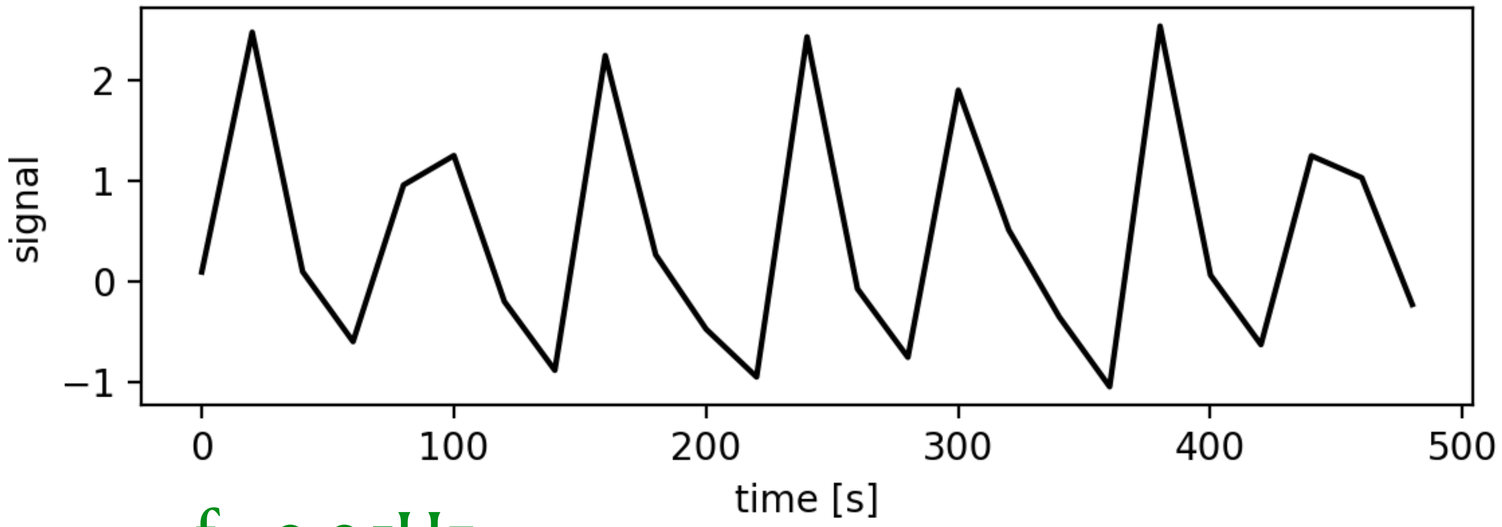




f_s=0.1Hz

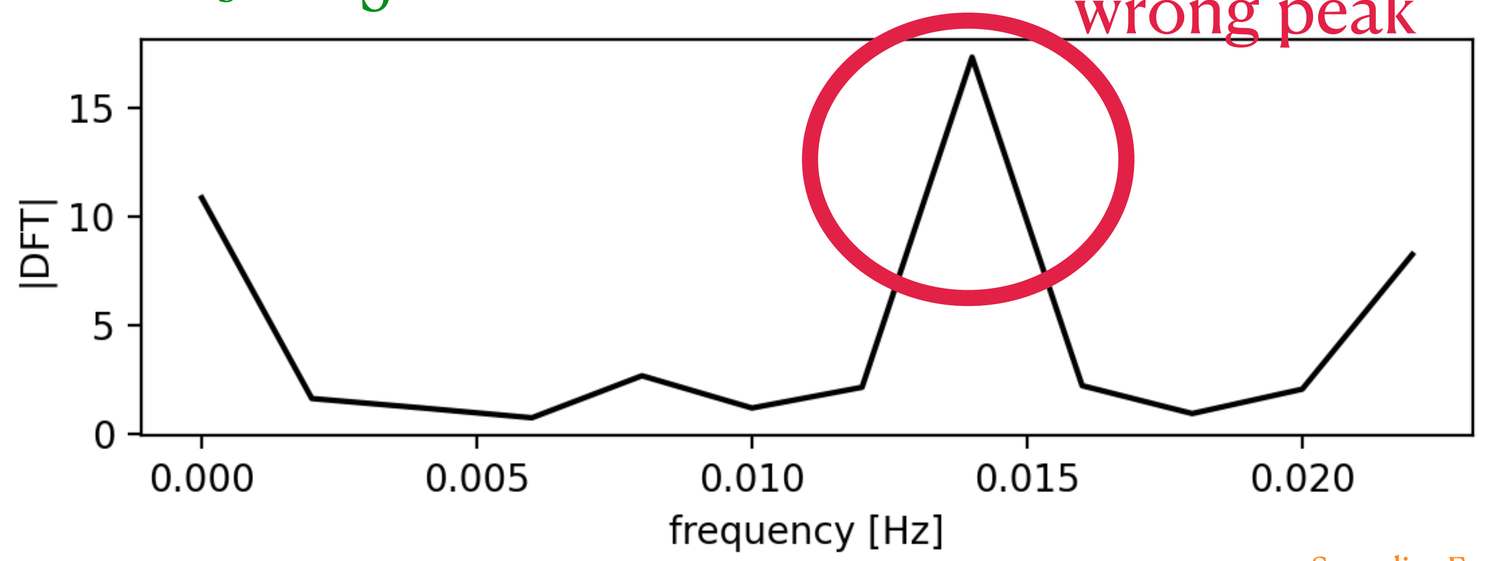
wr

wrong signal



f_s=0.05Hz

wrong peak



SamplingError_1.py

data sampling

Fourier analysis

errors in analysis **aliasing** spectral leakage spectral power

linear filters

time-frequency analysis

- assume: we have a signal s(t) and its Fourier Transform X(f)

- assume: we have a signal s(t) and its Fourier Transform X(f)

- then the Poisson summation formula yields:

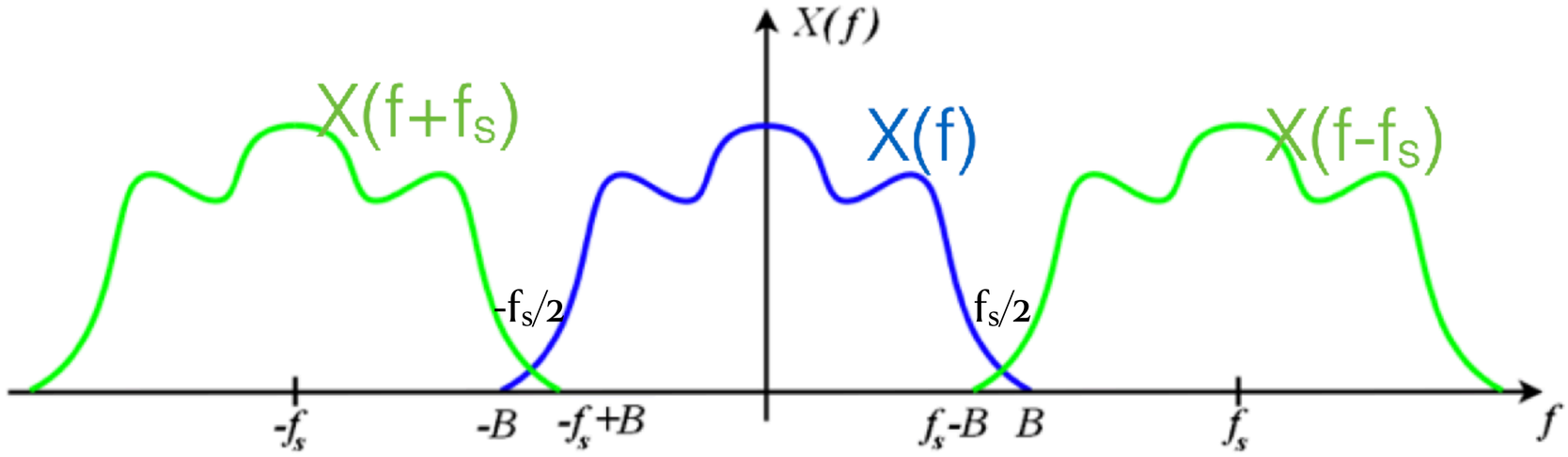
$$\sum_{k=1}^{N} s(t_k) e^{-i2\pi nk/N} = \frac{1}{\Delta t} \sum_{p=-\infty}^{\infty} X(f_n - p f_s)$$

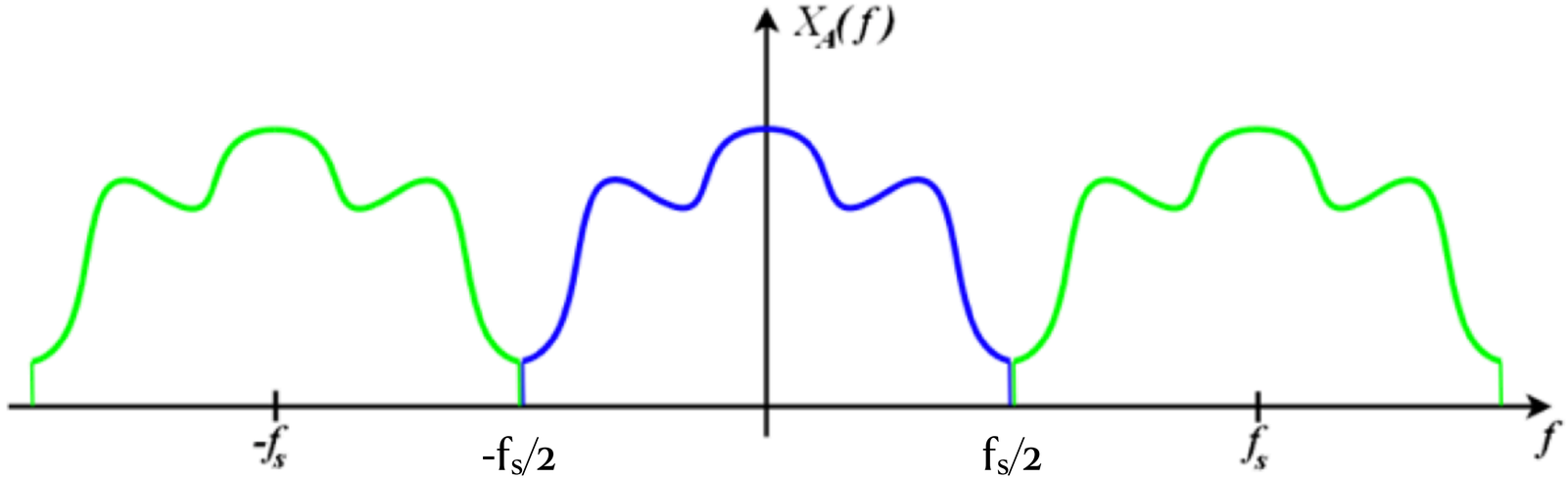
DFT

periodic continuation
of Fourier transform with
period f_s

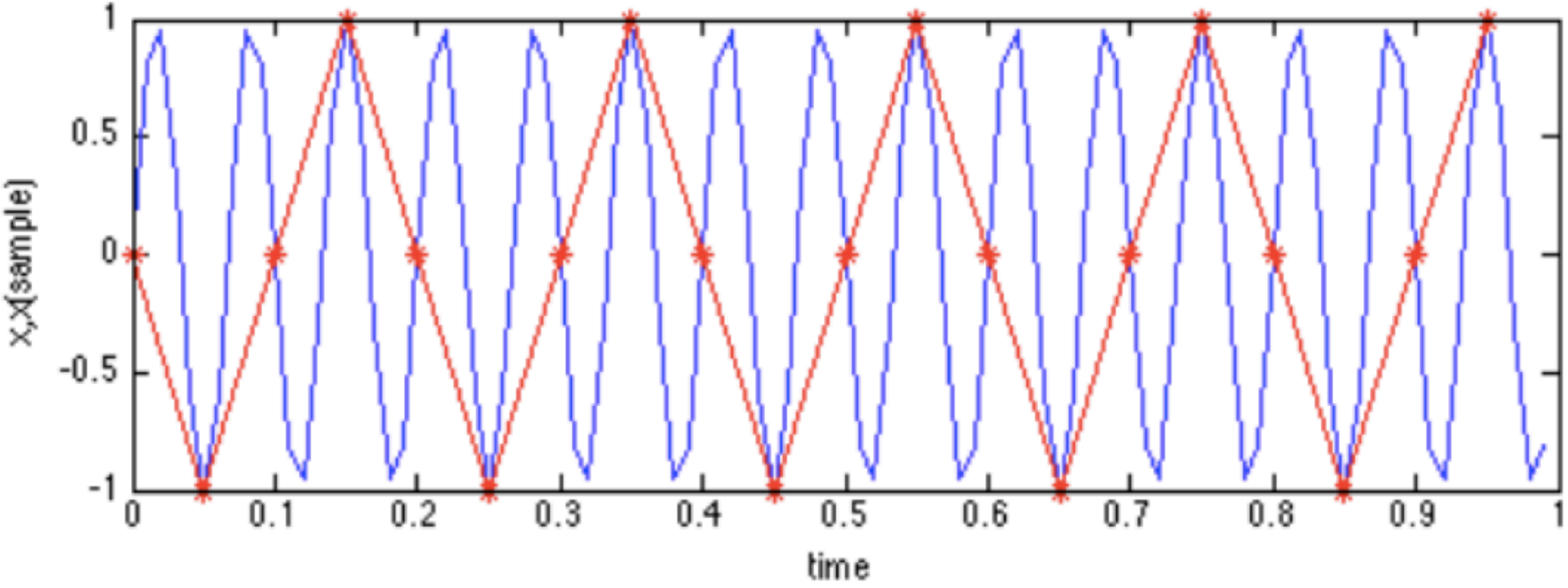
$$\mathrm{DFT} = X_A \sim X(f_n) + X(f_n - f_s) + X(f_n + f_s) + \cdots$$

$$\text{DFT} = X_A \sim X(f_n) + X(f_n - f_s) + X(f_n + f_s) + \cdots$$



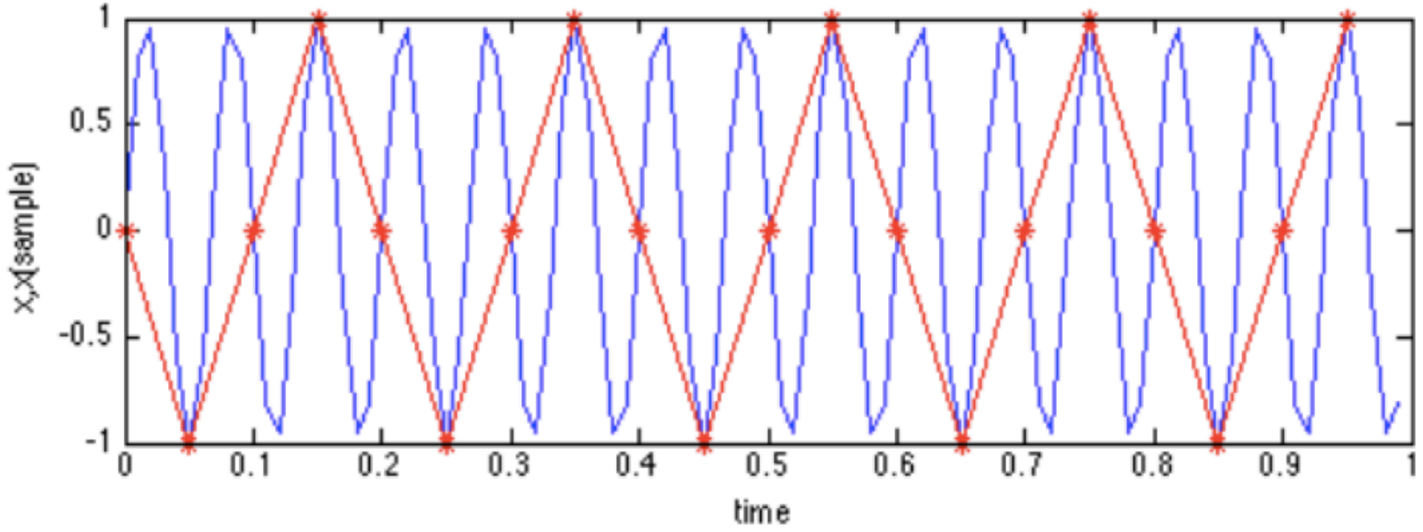


Example: sub-sampled periodic signal



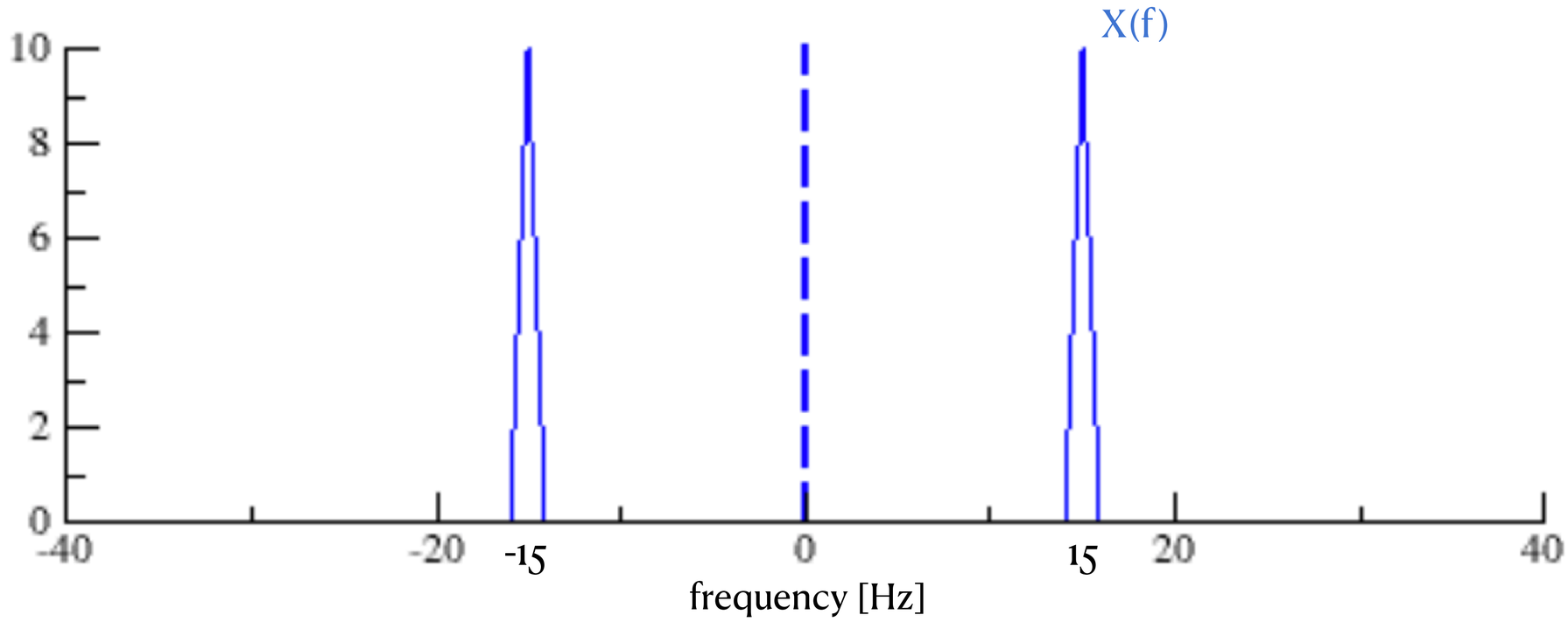
original time series with f_m=15Hz

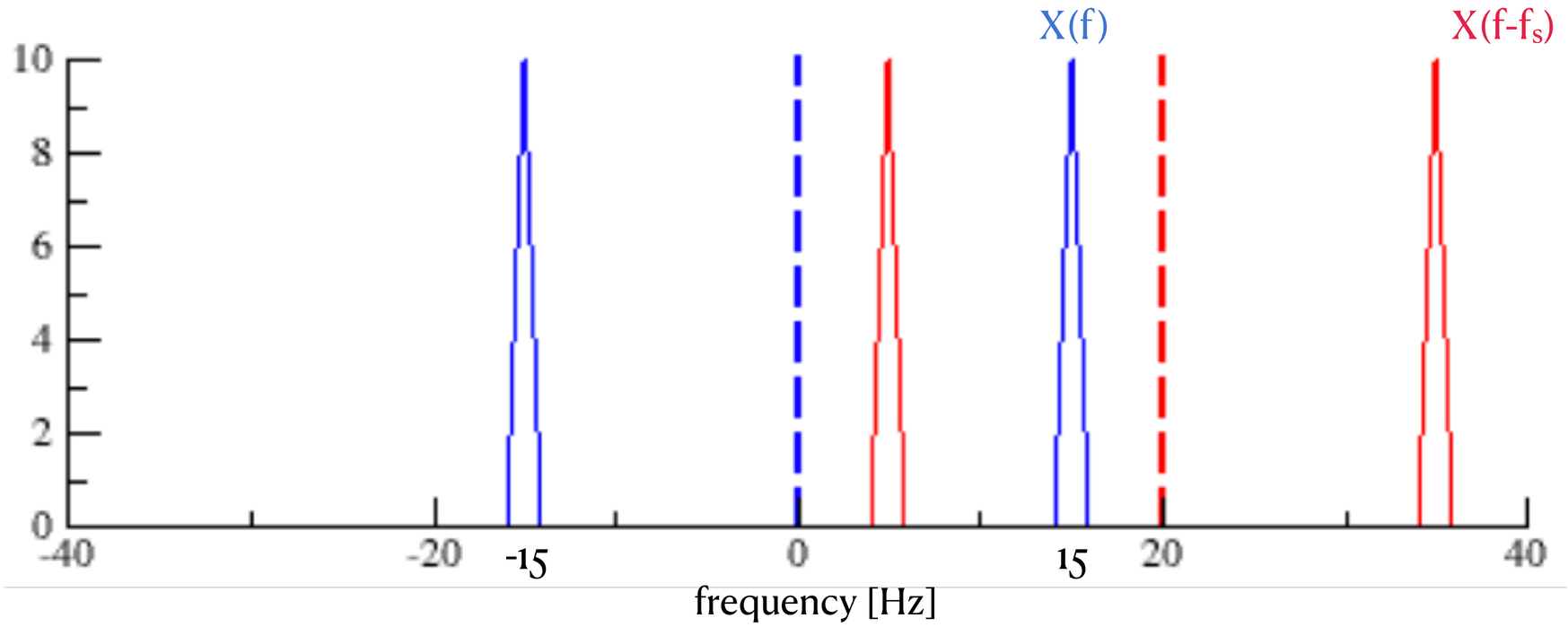
subsampled time series with f_s=20Hz

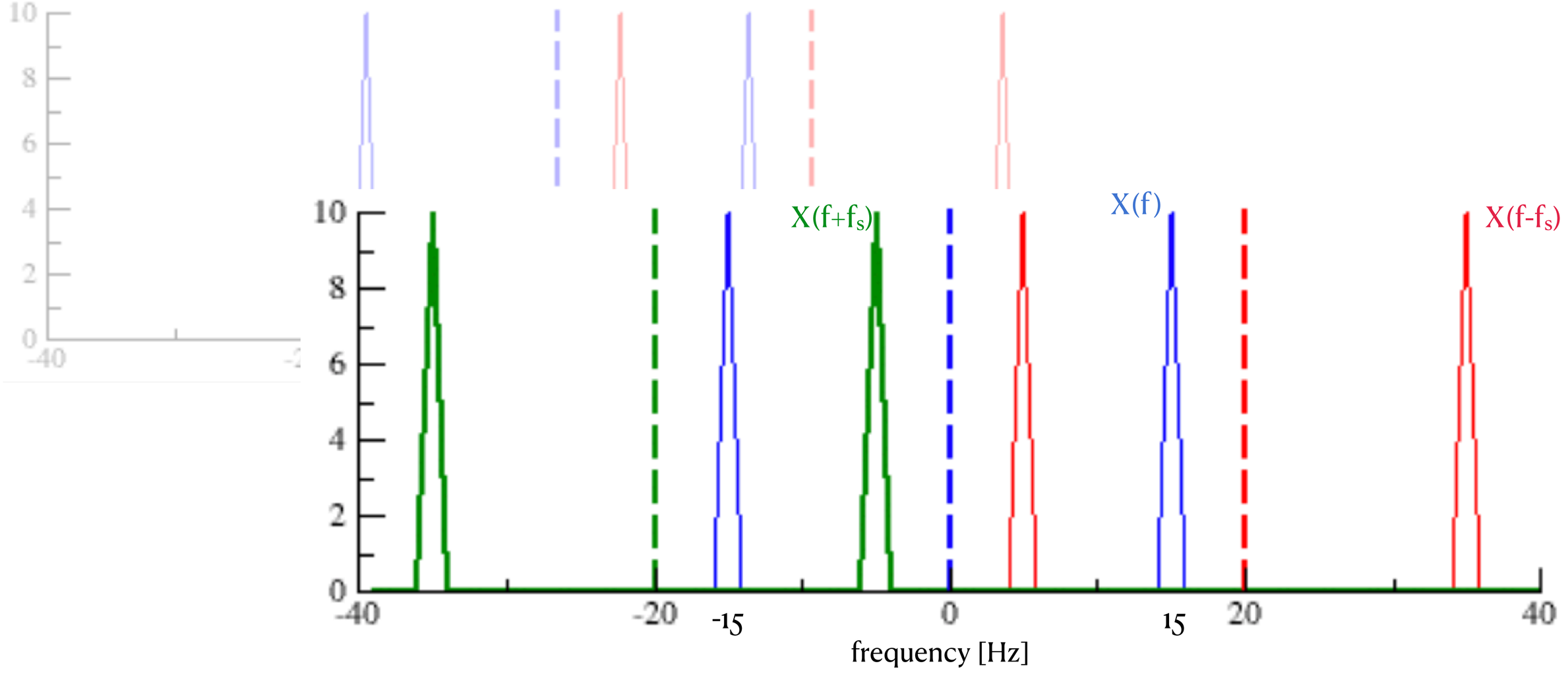


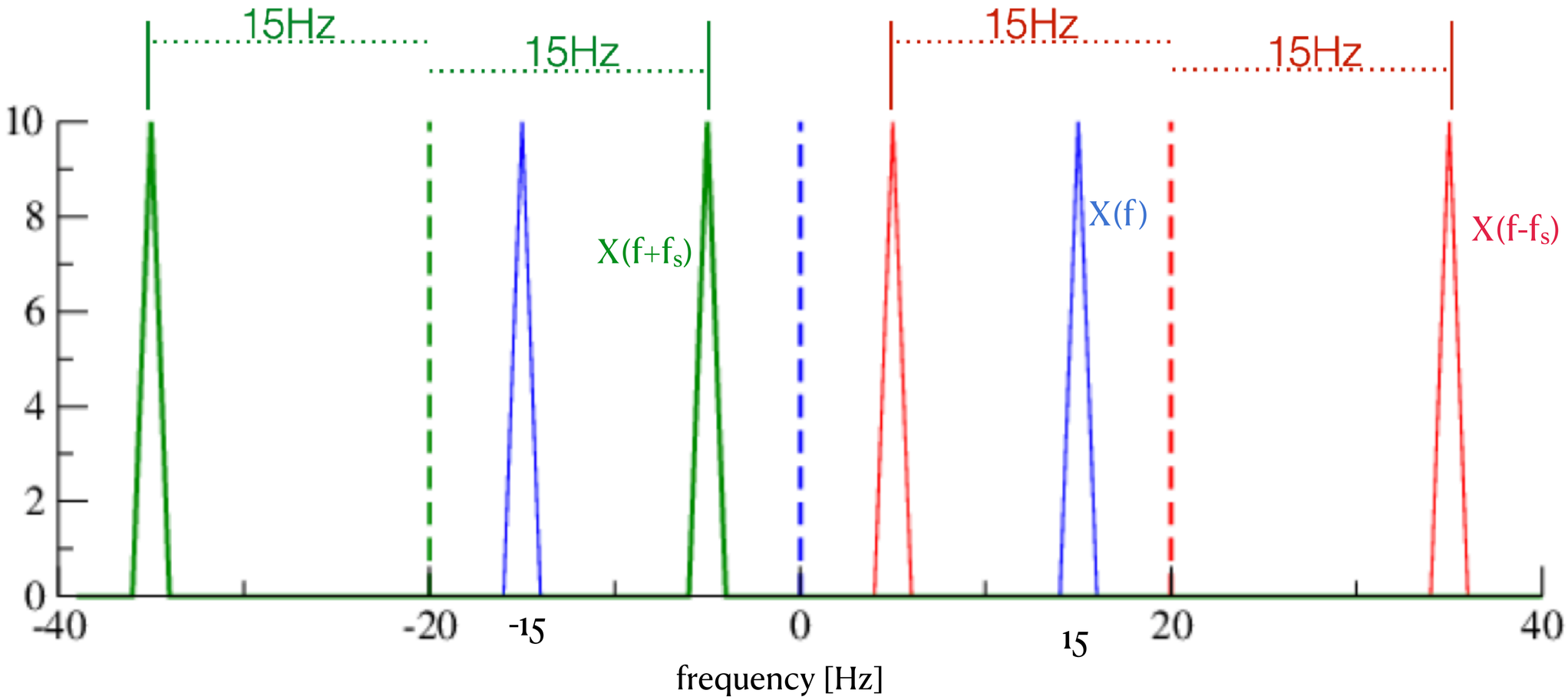
original time series with f_m=15Hz

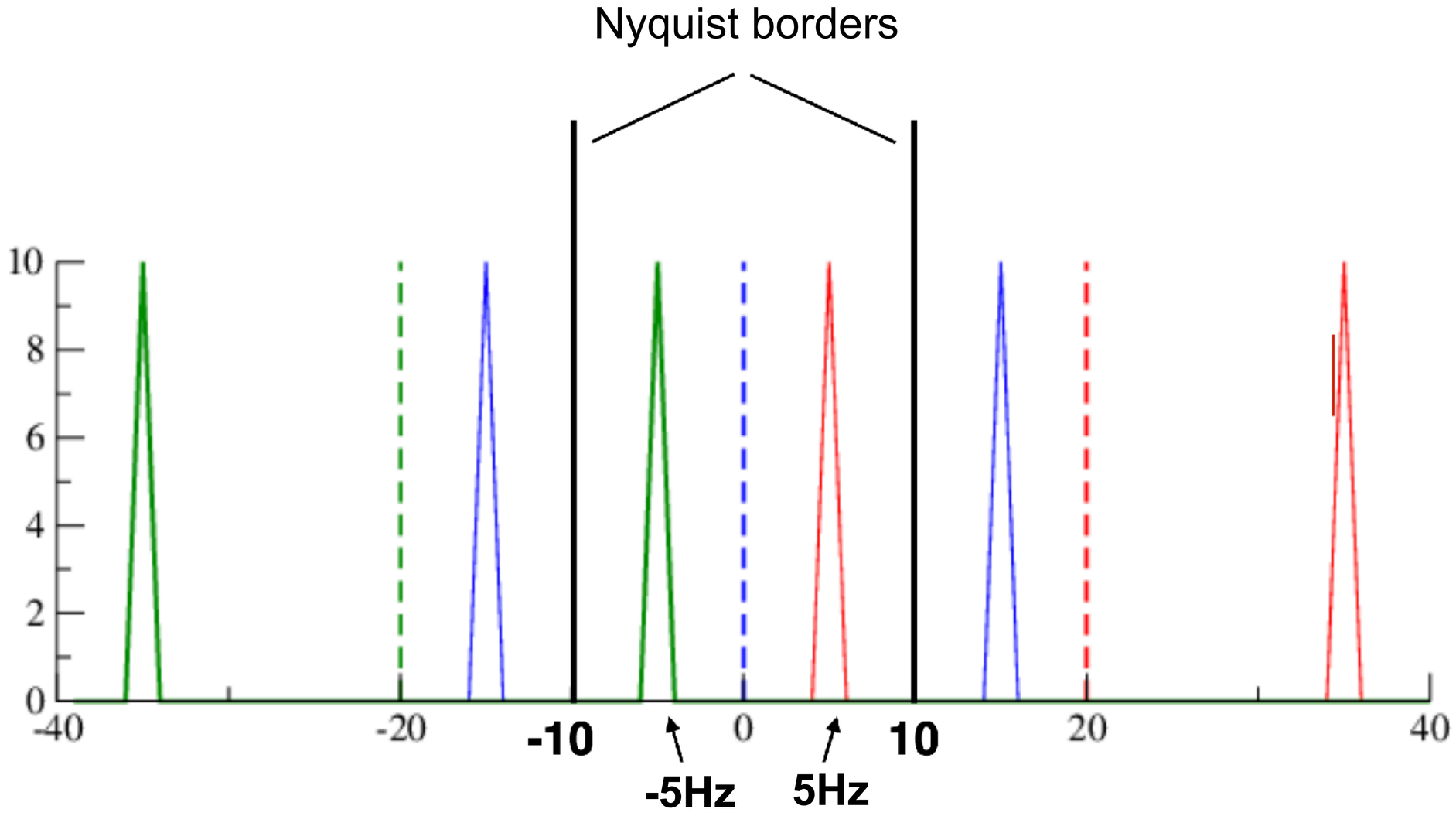
subsampled time series with f_s=20Hz

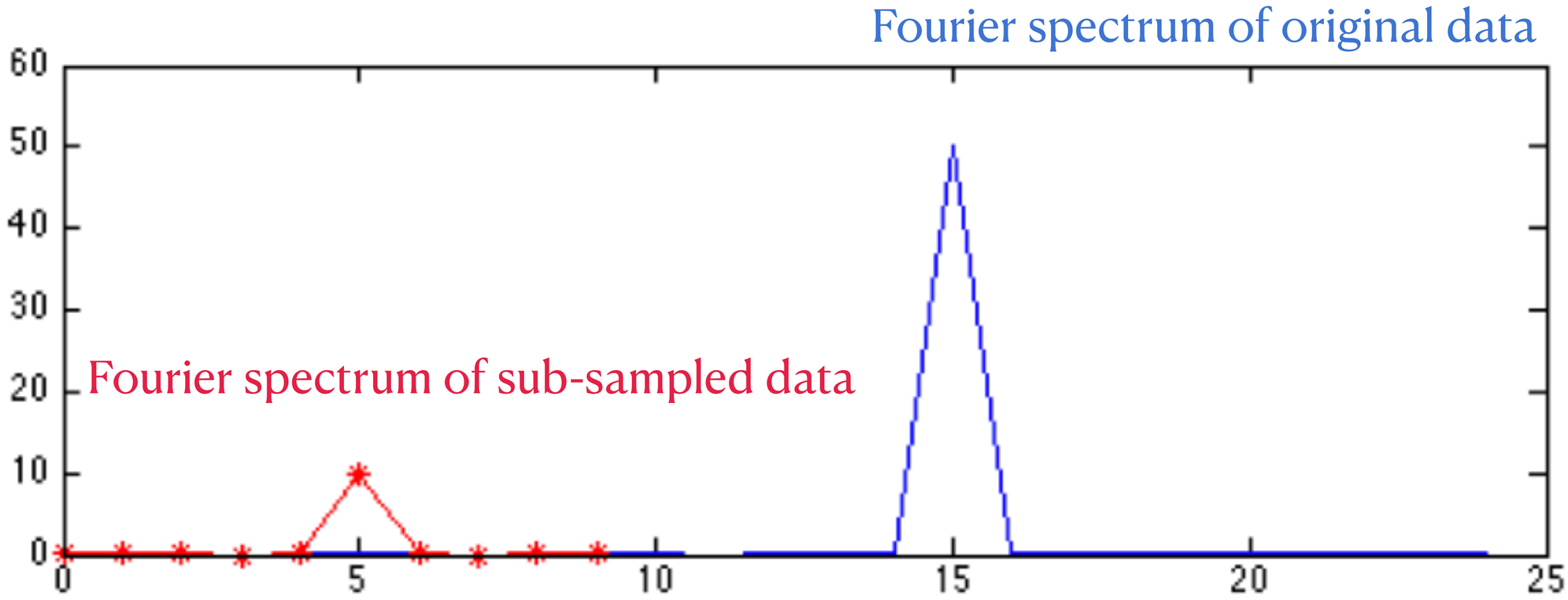



X(f)
X(f-fs)
10
8
6
4
2
0
-40
-20
-15
0
15
20
40
frequency [Hz]

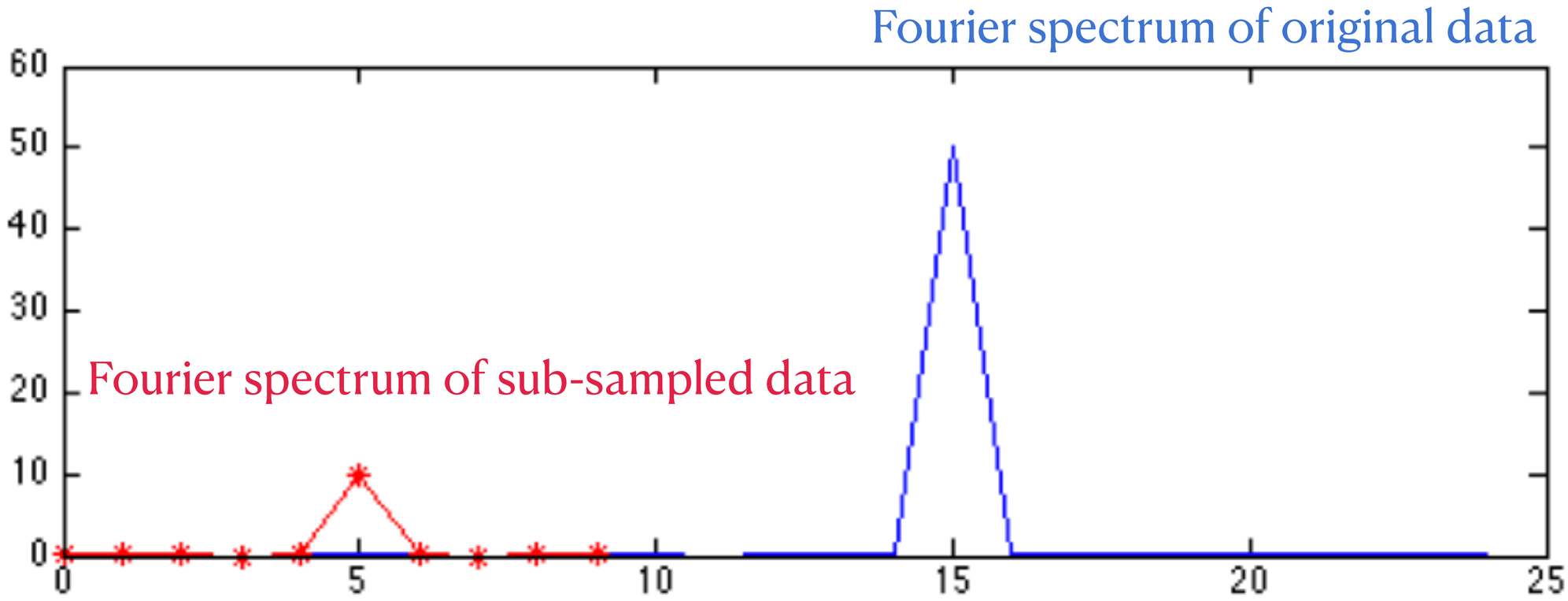

10
8
6
4
2
0
-40
X(f+fs)
X(f)
X(f-fs)
10
8
6
4
2
0
-40
-20
-15
0
15
20
40
frequency [Hz]


15Hz
15Hz
15Hz
15Hz
10
8
6
4
2
0
X(f+fs)
X(f)
X(f-fs)
-40
-20
-15
0
15
20
40
frequency [Hz]


Nyquist borders
10
8
6
4
2
0
-40
-20
-10
0
10
20
40
-5Hz
5Hz

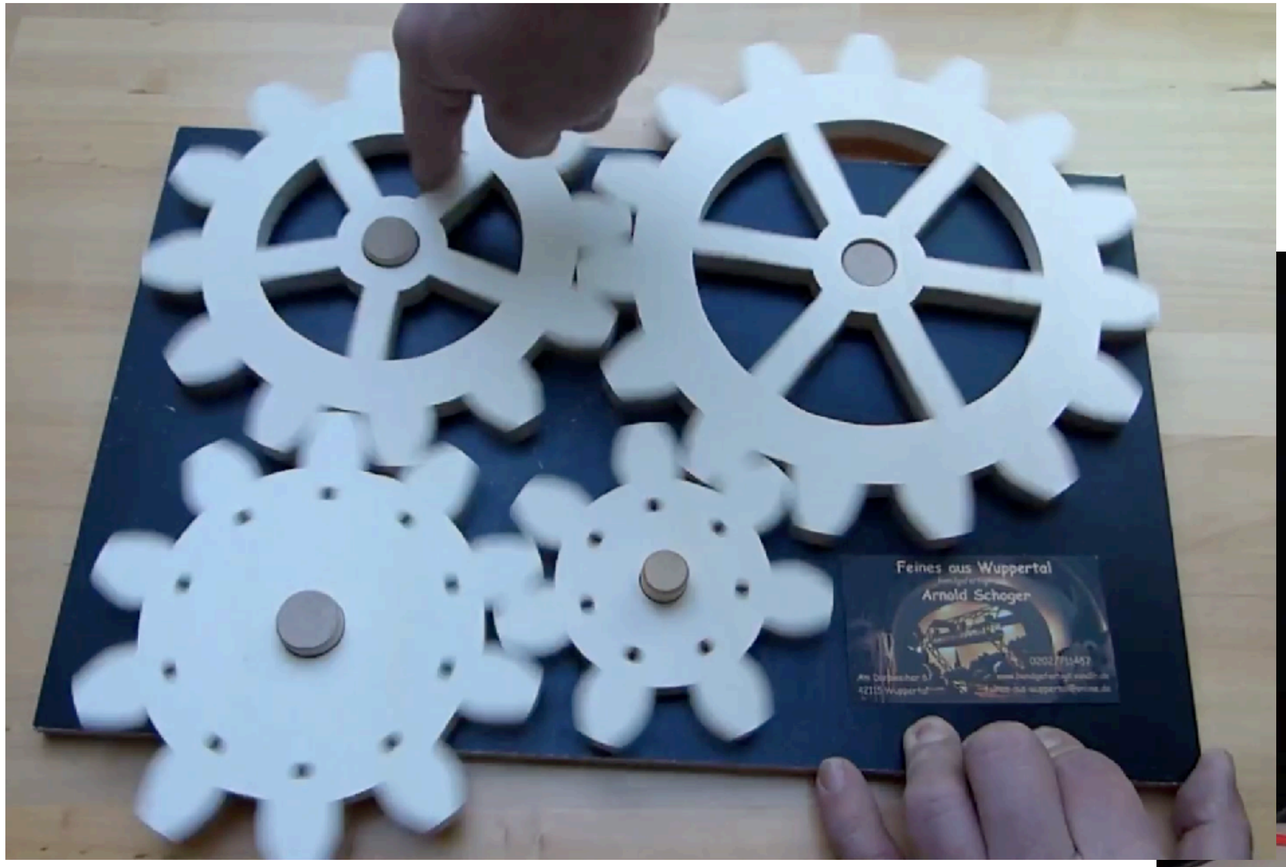

Fourier spectrum of original data
Fourier spectrum of sub-sampled data
60
50
40
30
20
10
0
0
5
10
15
20
25

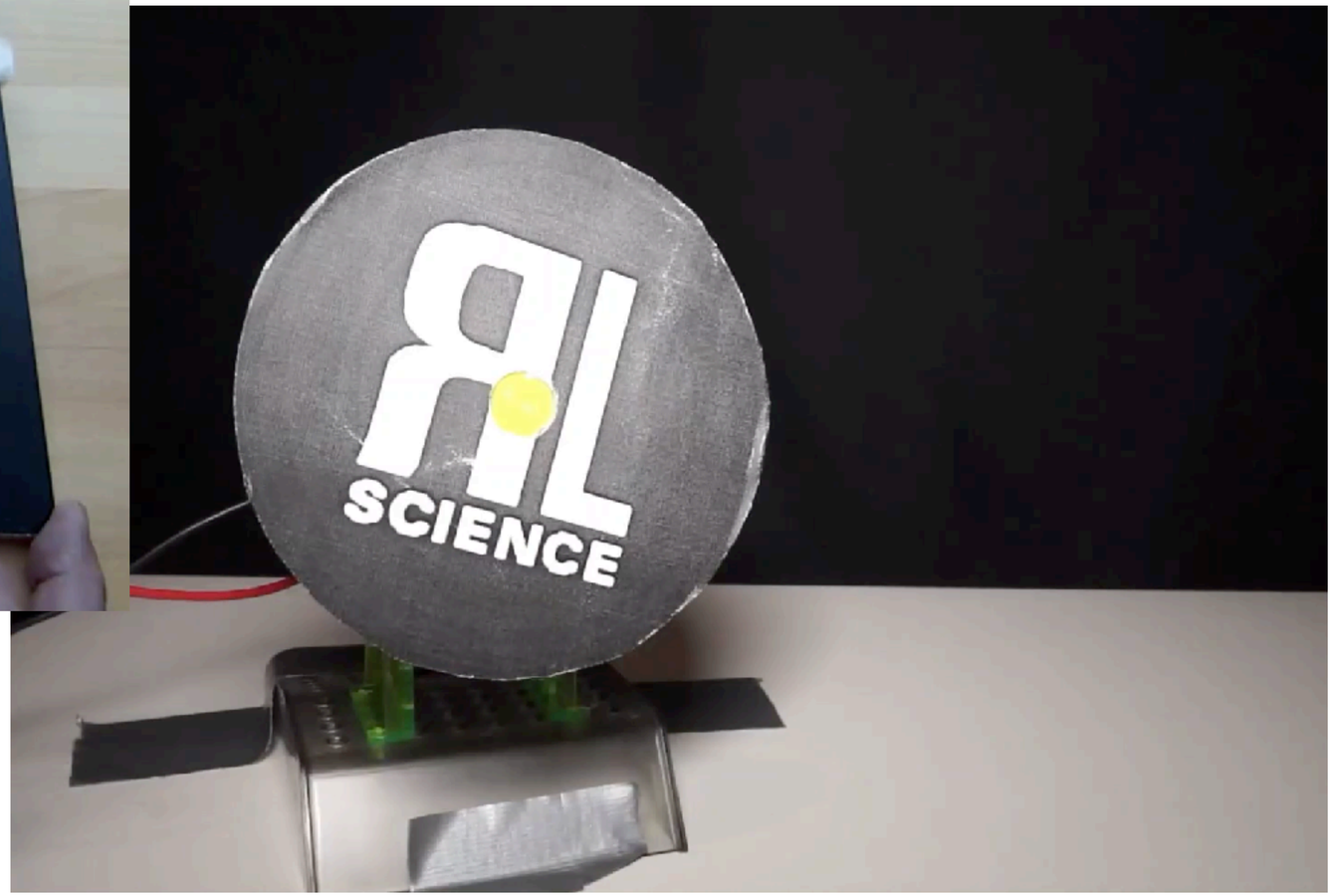
aliasing



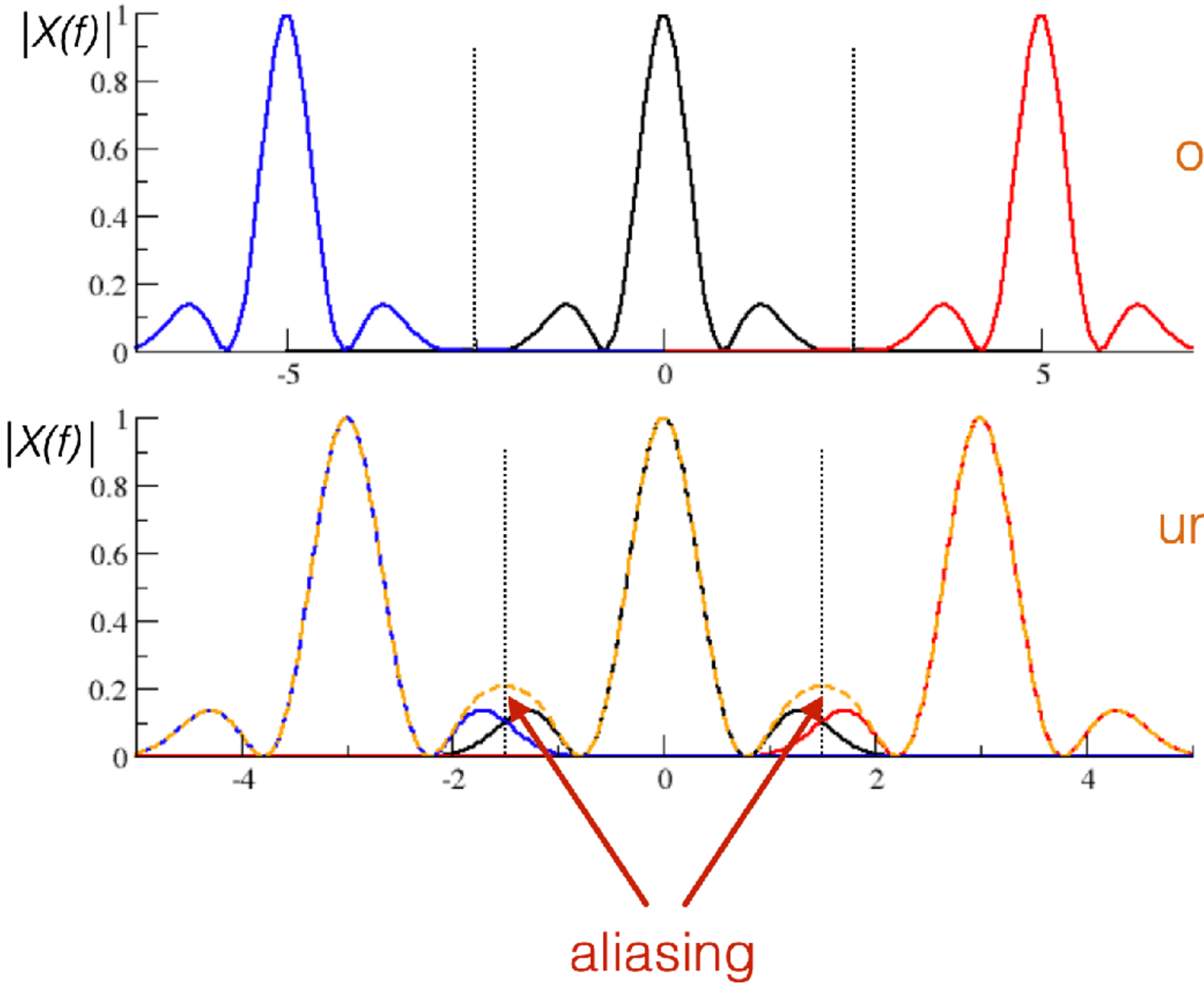
erroneous peak

aliasing = stroboscope effect





summary for aliasing



choose sampling rate that is more than double the maximum frequency

data sampling

Fourier analysis

errors in analysis

aliasing **spectral leakage** spectral power

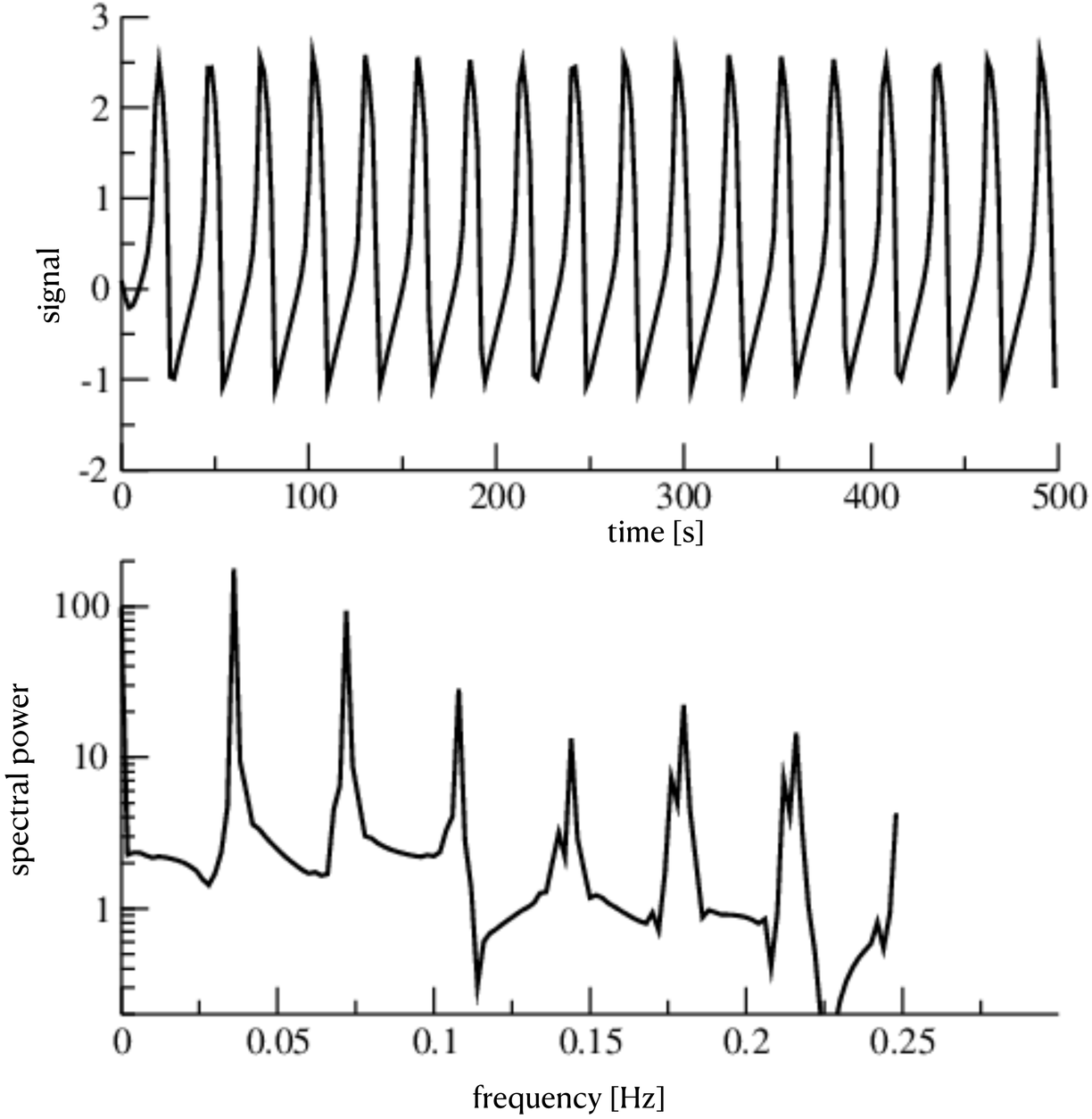
linear filters

time-frequency analysis

Example: FHN oscillator

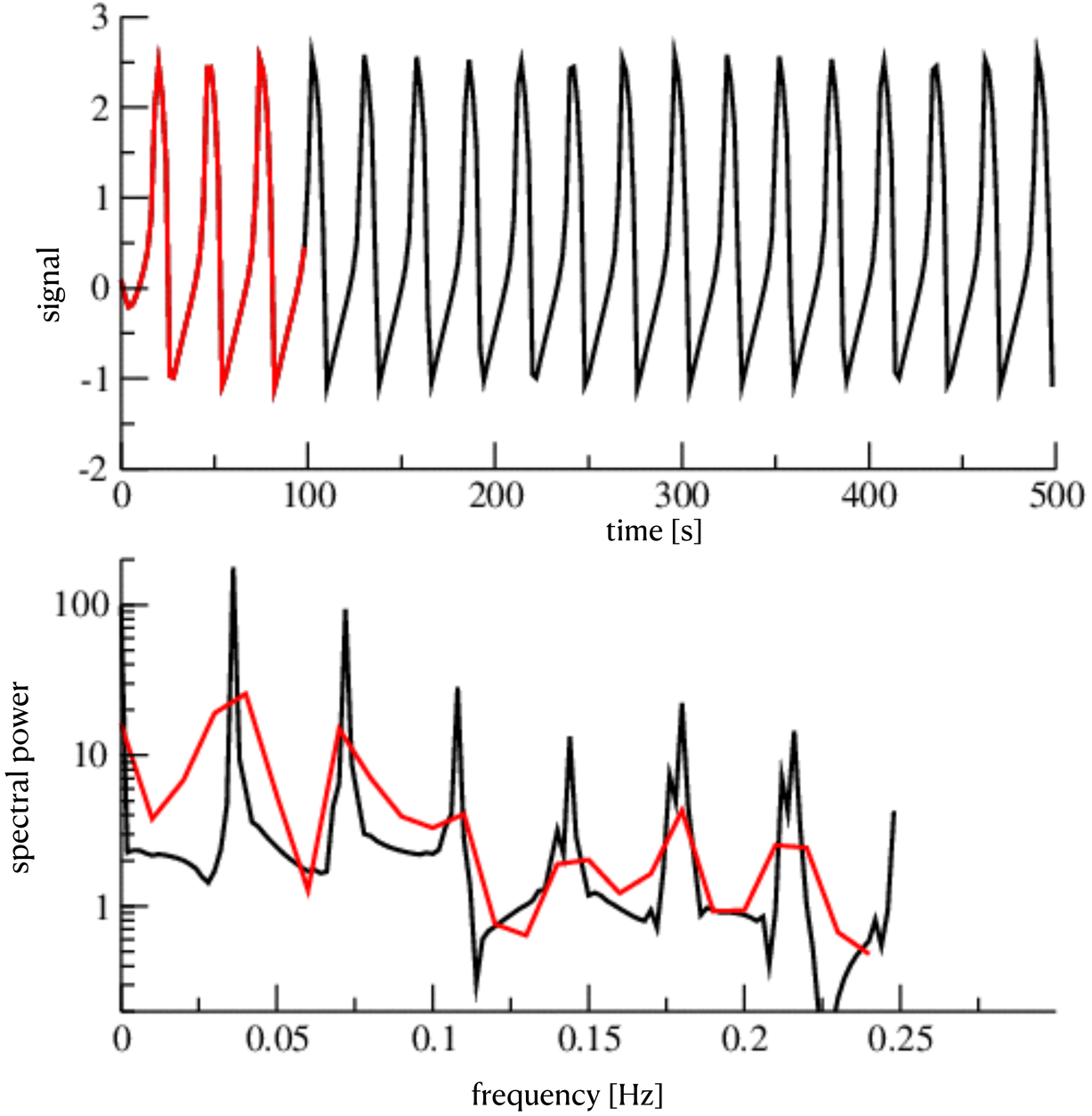
$f_s = 0.5\text{Hz}$

$T = 500\text{s}$

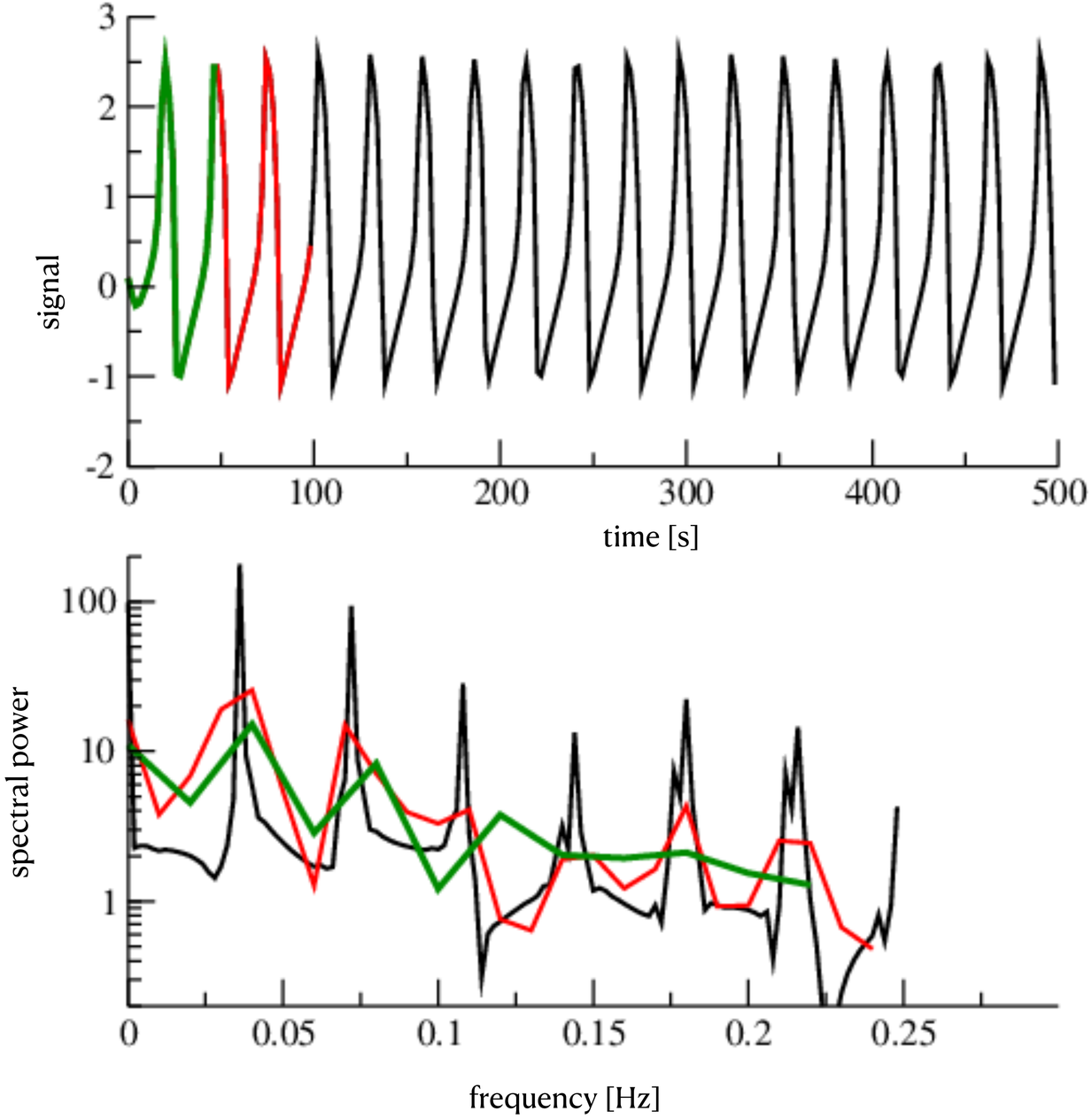


SamplingError_2.py

T= 100s


3
2
1
0
-1
-2
signal
0
100
200
300
400
500
time [s]
100
10
1
spectral power
0
0.05
0.1
0.15
0.2
0.25
frequency [Hz]

T= 50s

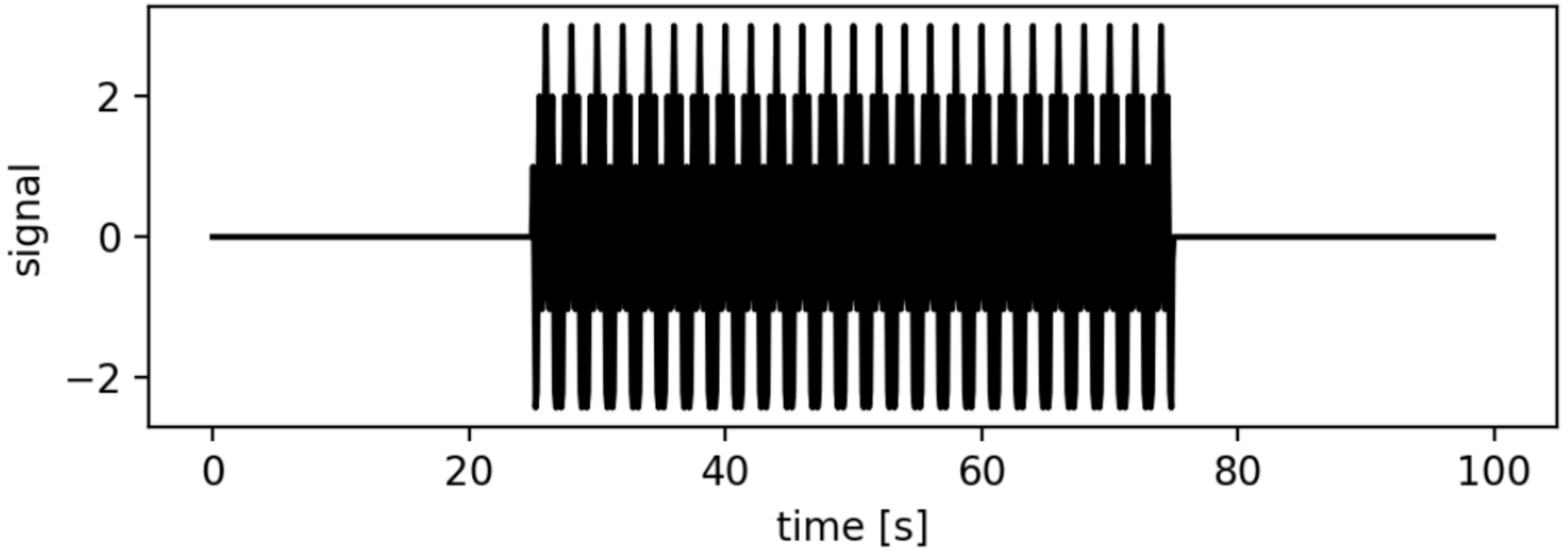


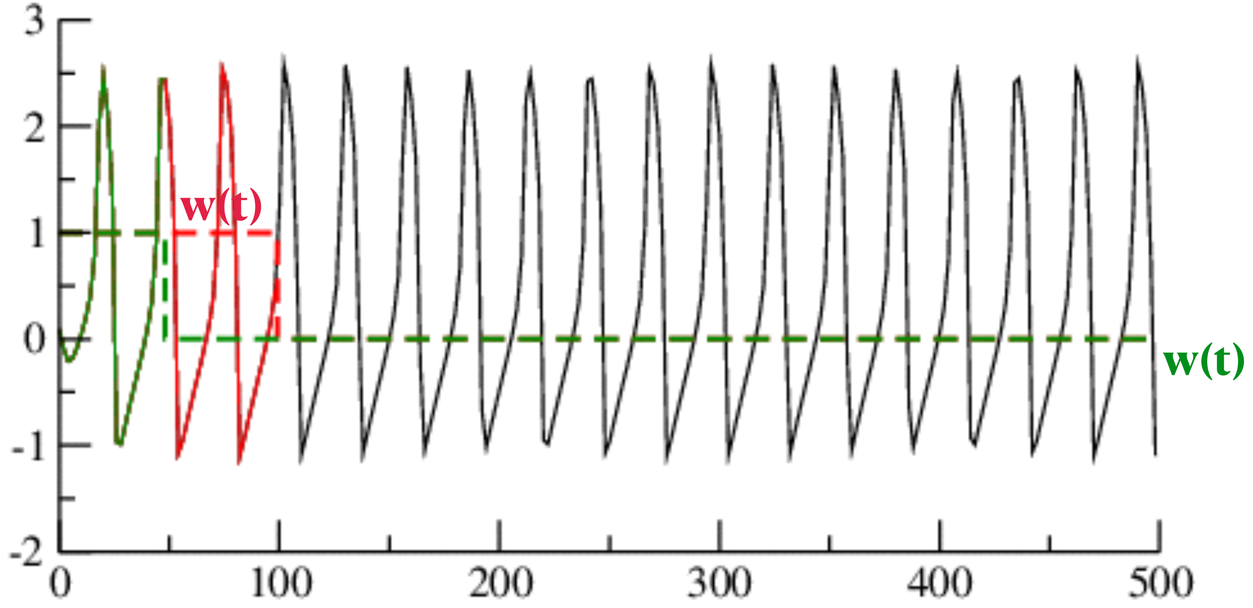
the shorter the time series, the worse the spectral estimate

how to improve the spectral estimate ?

how to improve the spectral estimate ?

idea: prolongation of signal by zero-padding

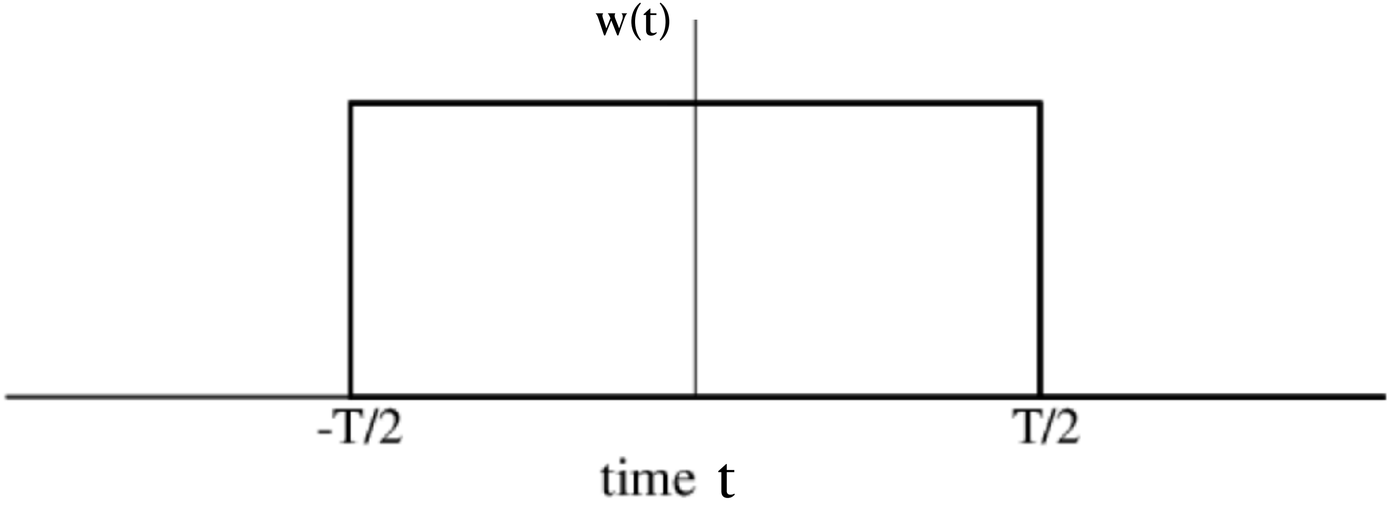




description of short signal:

multiplication by window function

$$s(t_k) \;\rightarrow\; s(t_k)w(t_k)$$



$$\text{DFT} = \sum_{k=1}^{N} s(t_k) w(t_k) e^{-i2\pi nk/N} \sim \sum_{m=-N/2}^{N/2} c_{n-m} d_m$$

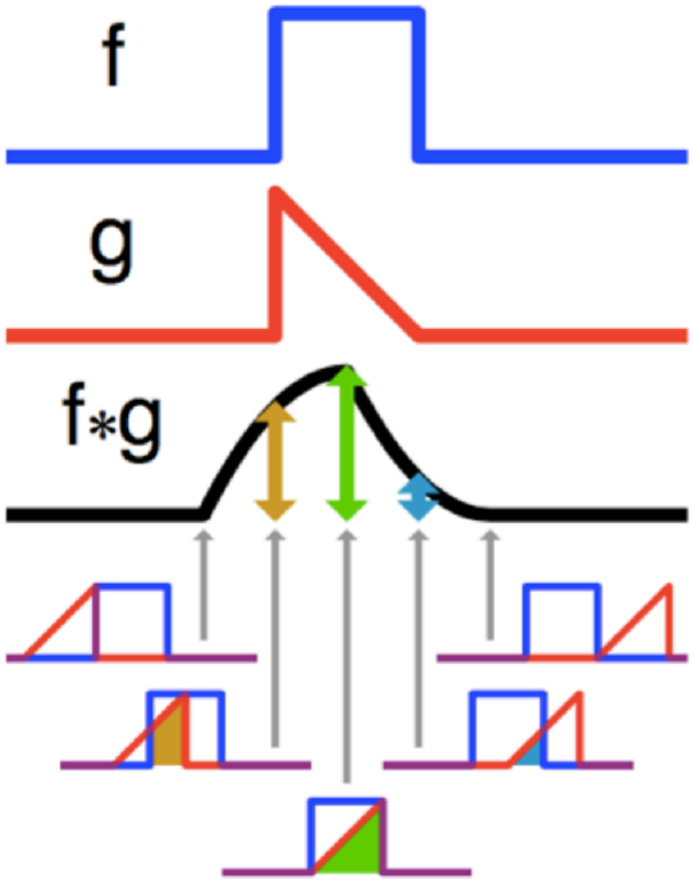
$$d_m = \sum_{k=1}^{N} w(t_k) e^{-i2\pi nk/N}$$

$$\text{DFT} = \sum_{k=1}^{N} s(t_k) w(t_k) e^{-i2\pi nk/N} \sim \sum_{m=-N/2}^{N/2} c_{n-m} d_m$$

$$d_m = \sum_{k=1}^{N} w(t_k) e^{-i2\pi nk/N}$$

$$\text{DFT} = \sum_{k=1}^{N} s(t_k) w(t_k) e^{-i2\pi nk/N} \sim \sum_{m=-N/2}^{N/2} c_{n-m} d_m$$

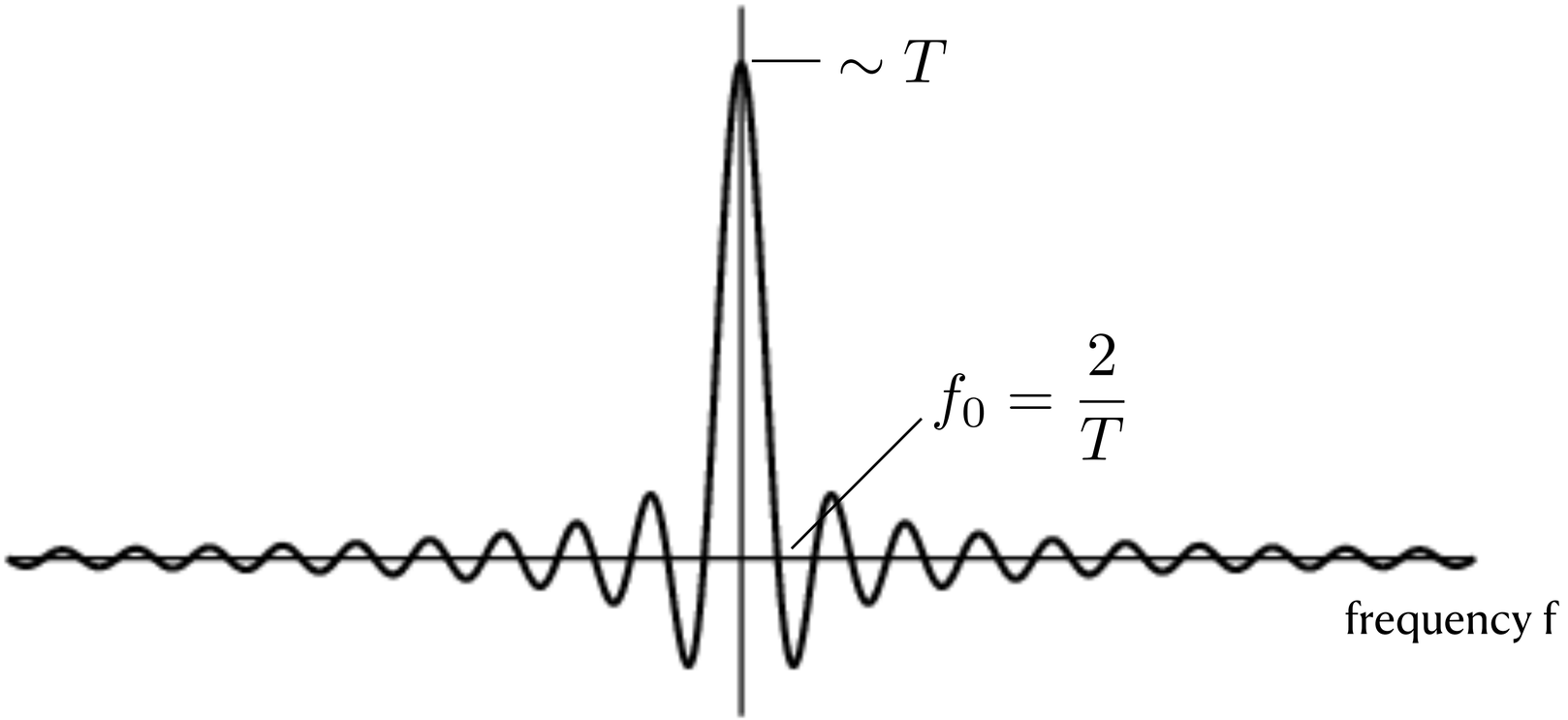
$$d_m = \sum_{k=1}^{N} w(t_k) e^{-i2\pi nk/N}$$



spectrum is convoluted with window Fourier transform

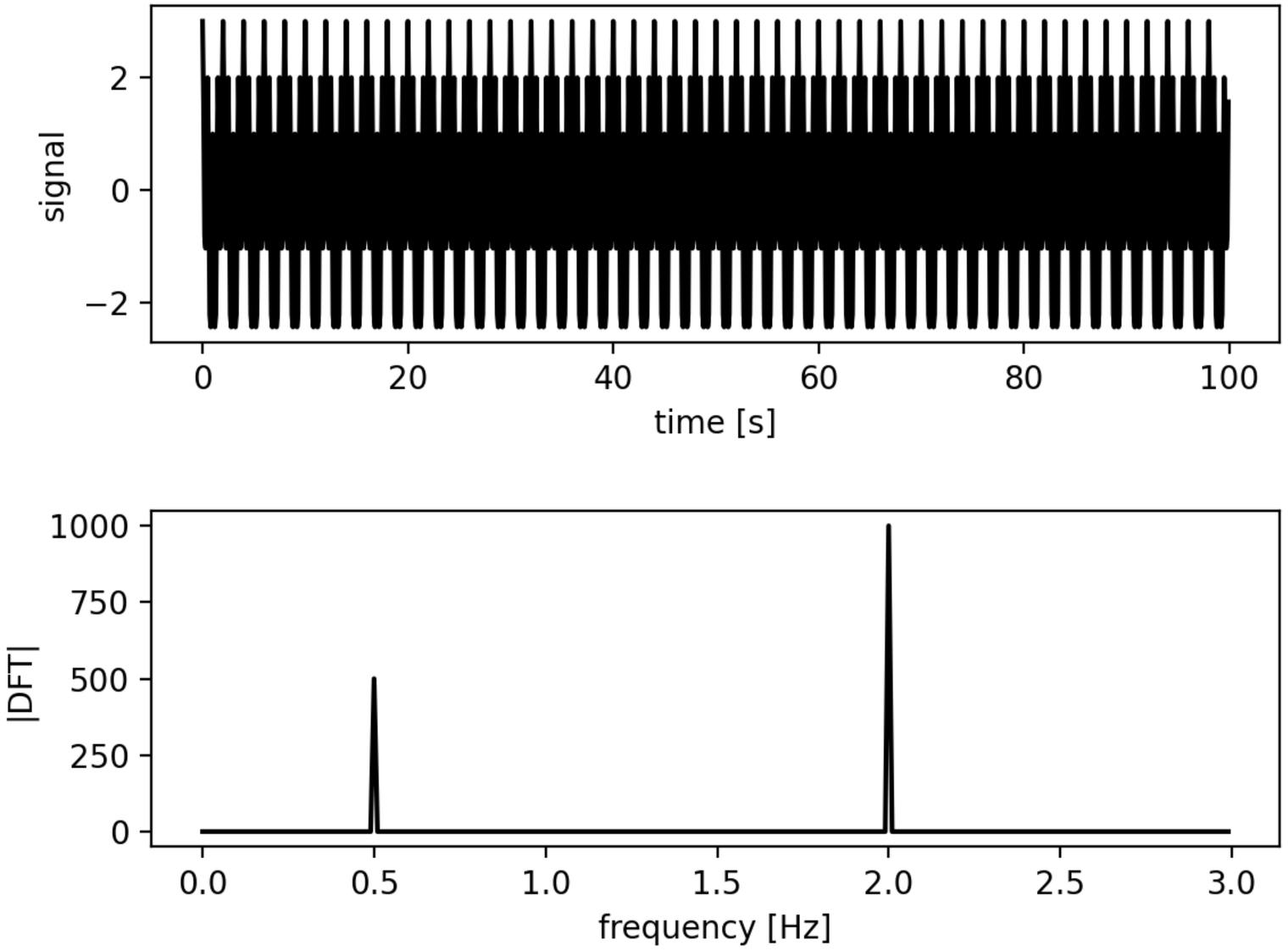
$$d_k \sim \frac{\sin(\pi f_k T)}{f_k}$$

Fourier transform of window function



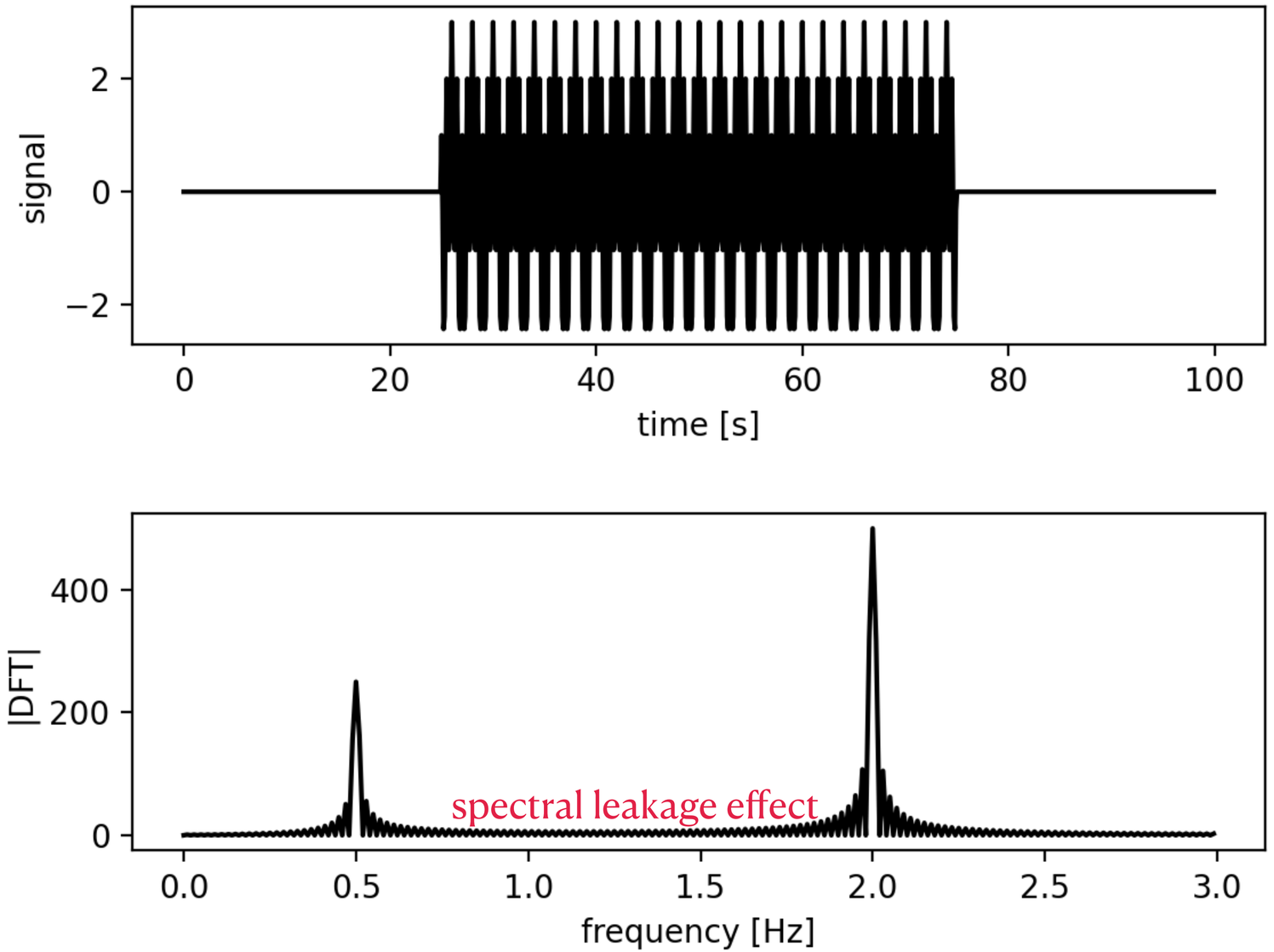
spectral leakage effect

Example: sinusoidal oscillation



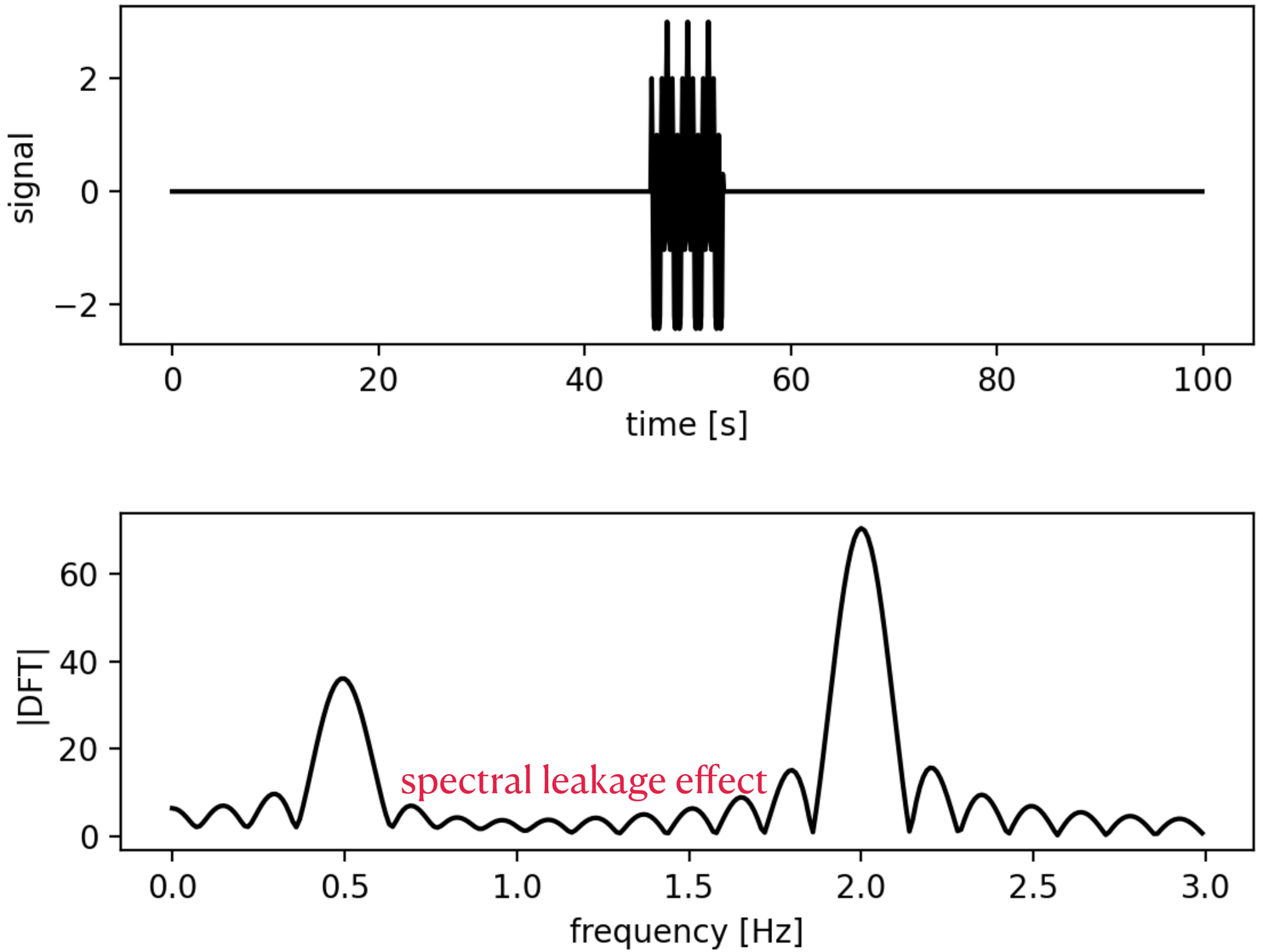
SamplingError_3.py

Example: sinusoidal oscillation



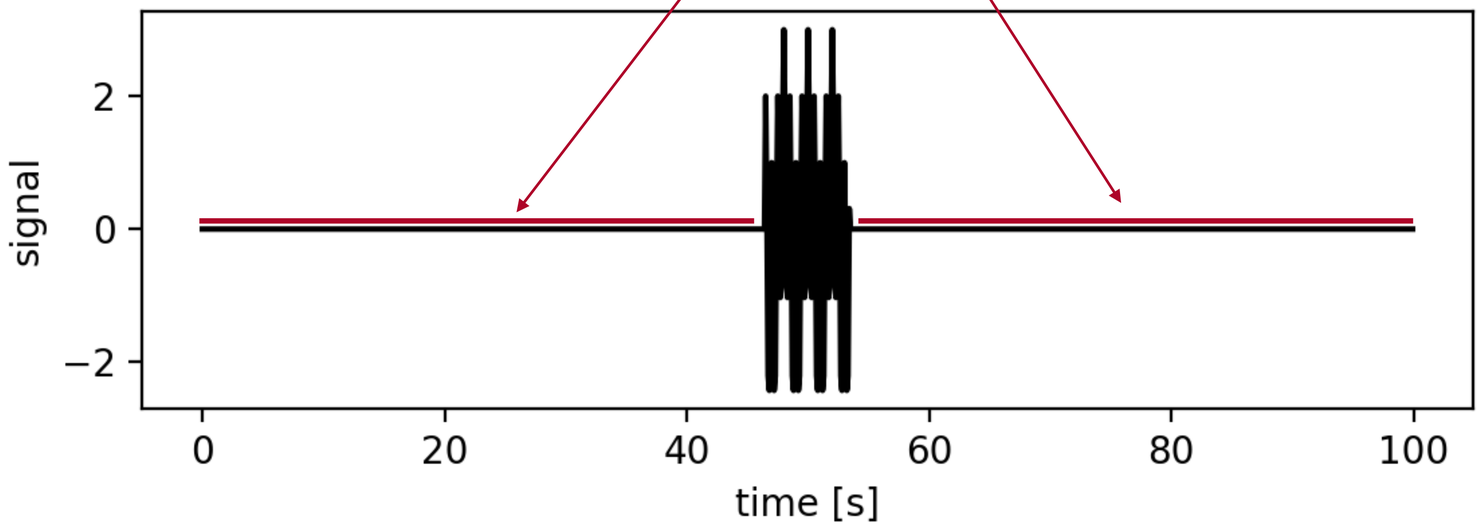
SamplingError_3.py

Example: sinusoidal oscillation

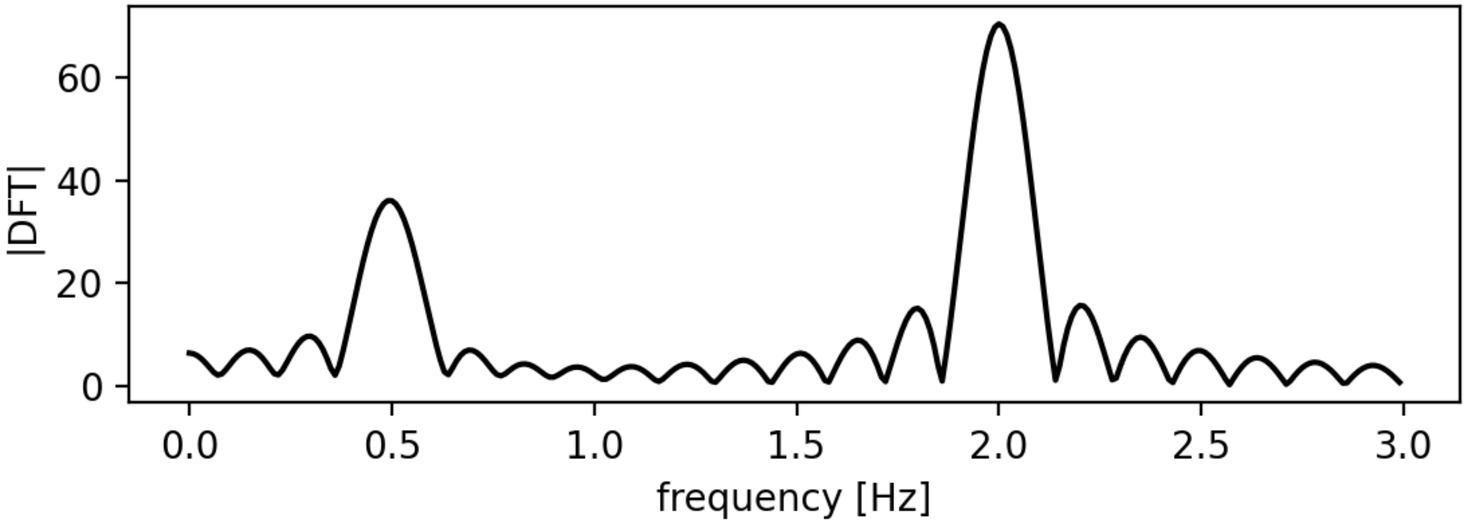


SamplingError_3.py

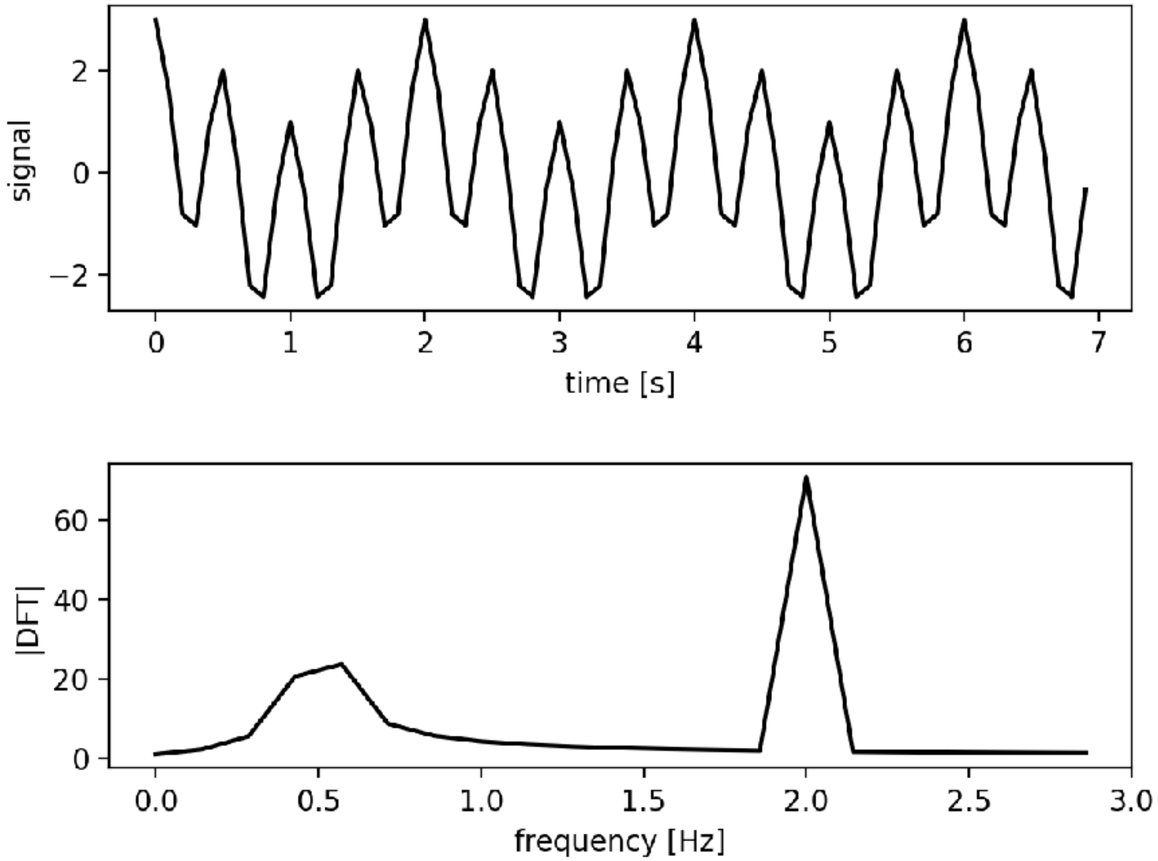
zero padding



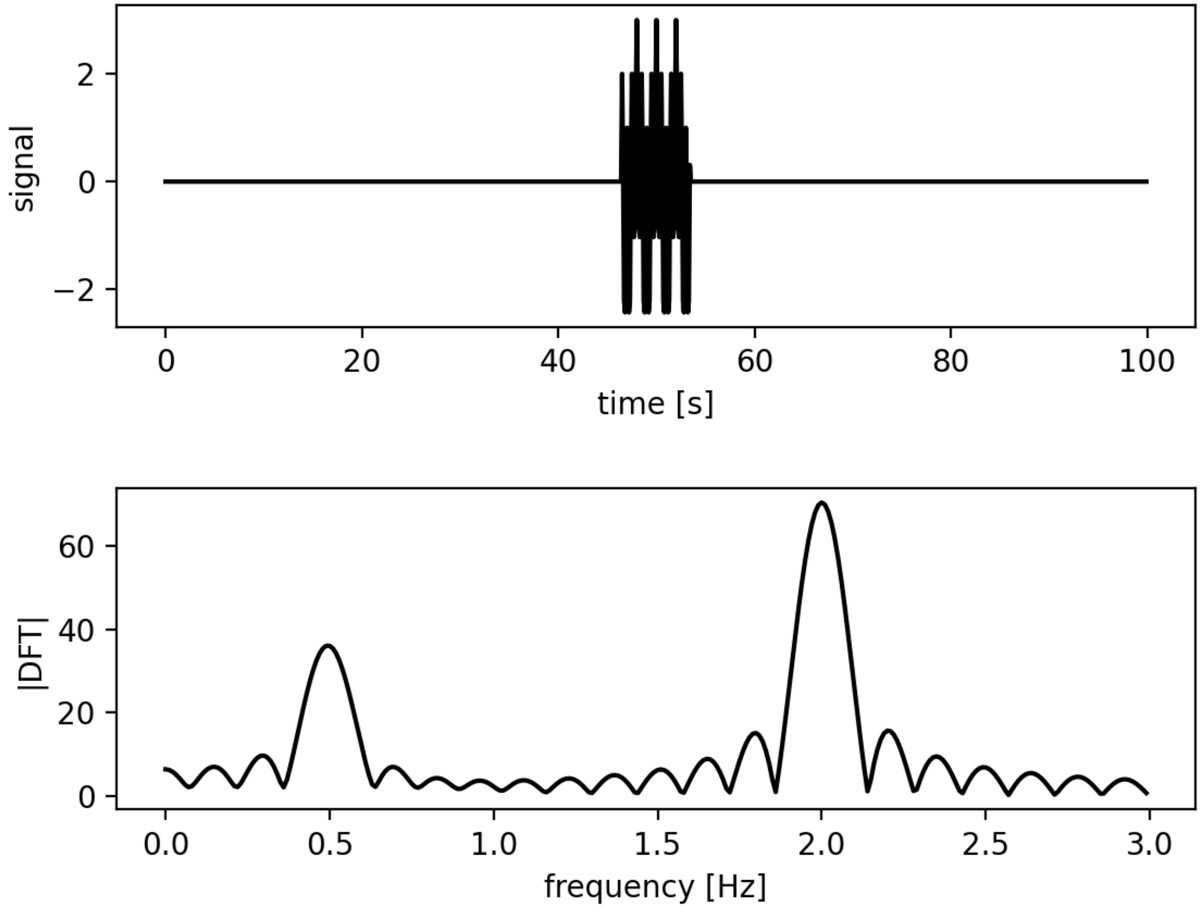
SamplingError_3.py



short signal, no zero padding

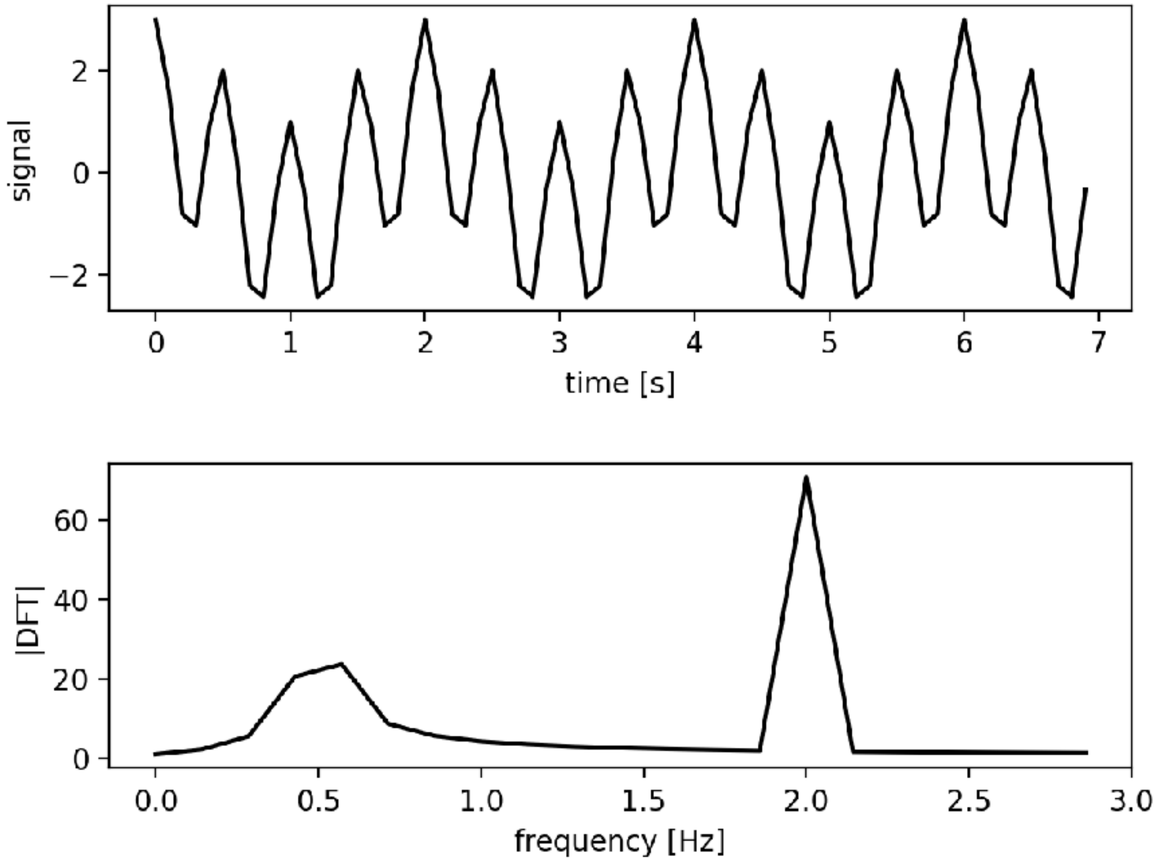


long signal, with zero padding

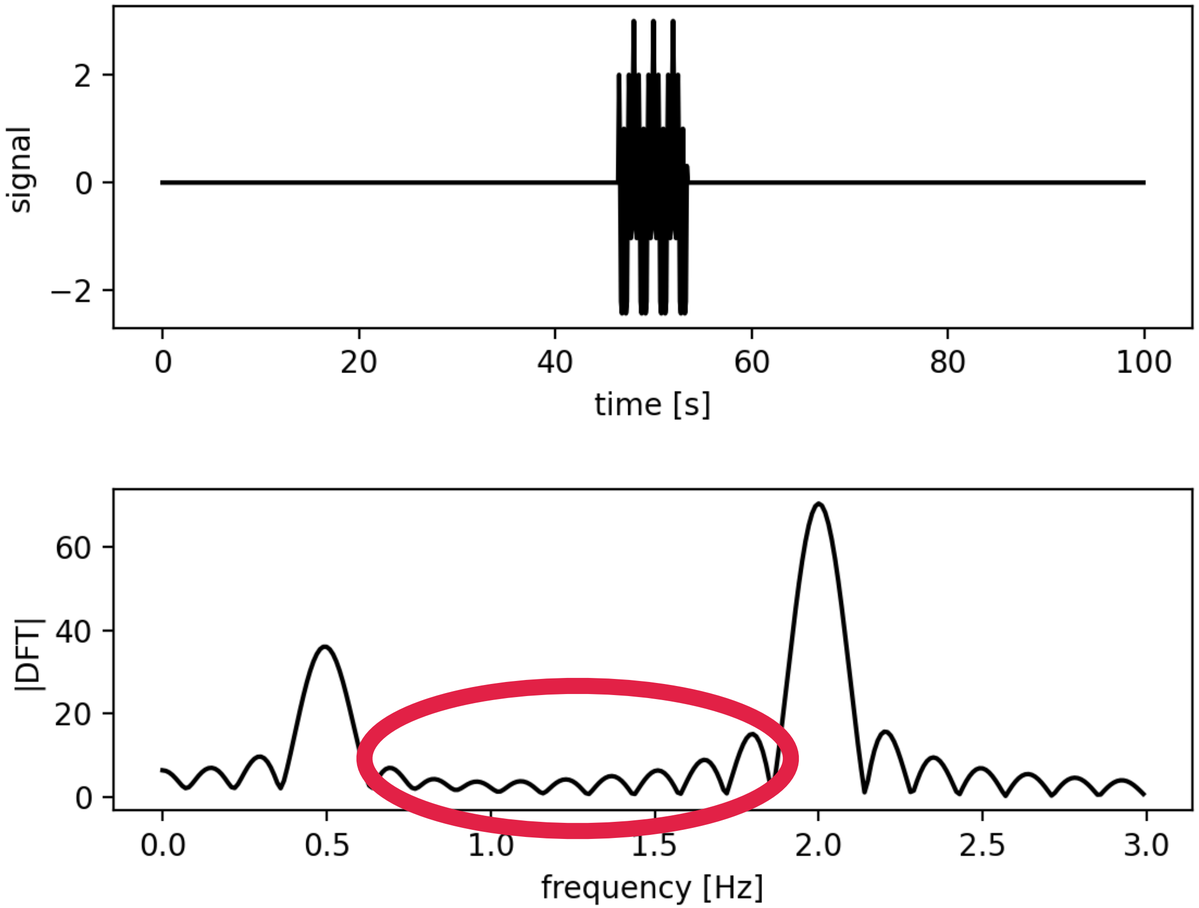


spectral leakage effect by zero-padding

short signal, no zero padding

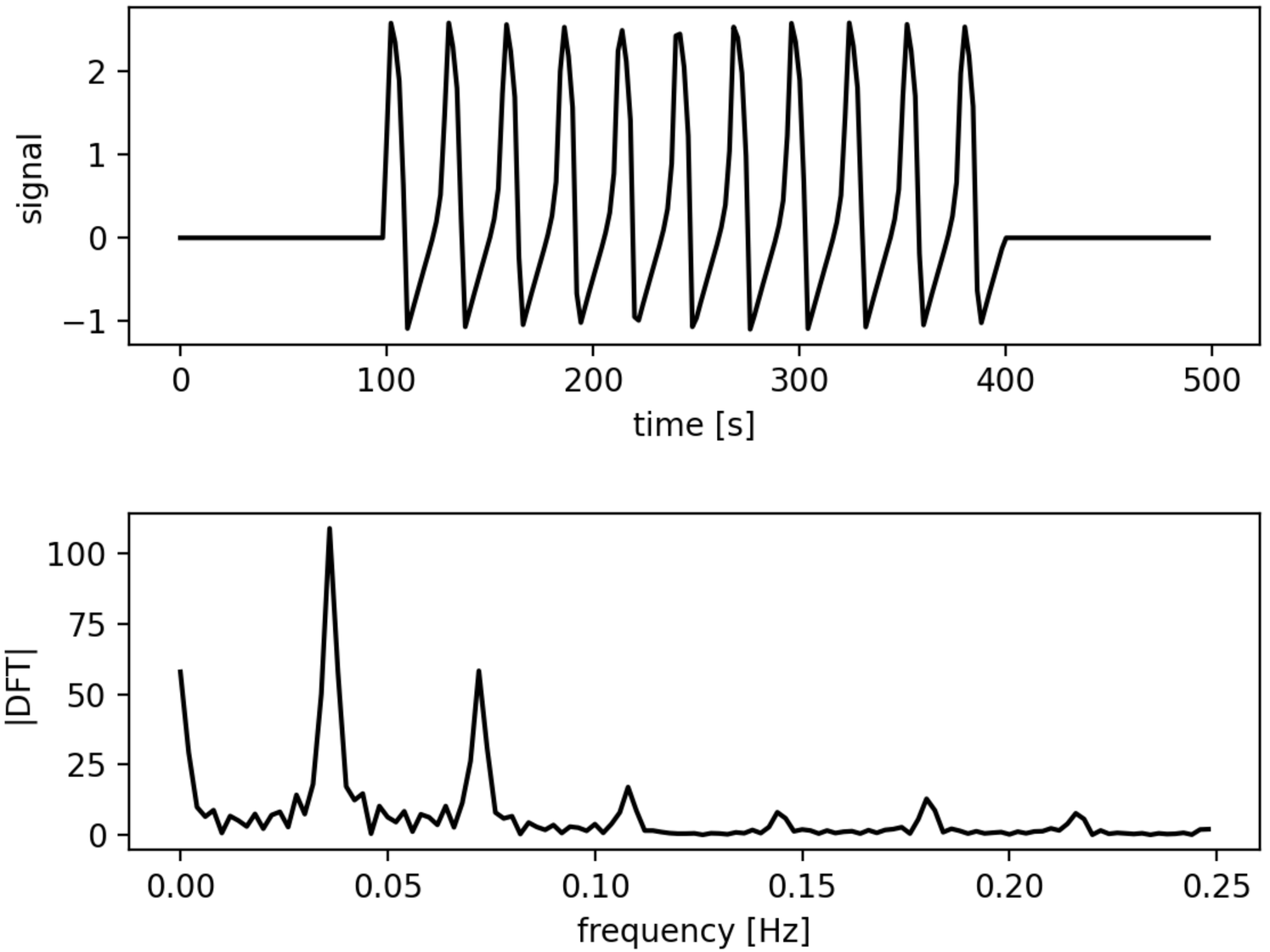


long signal, with zero padding

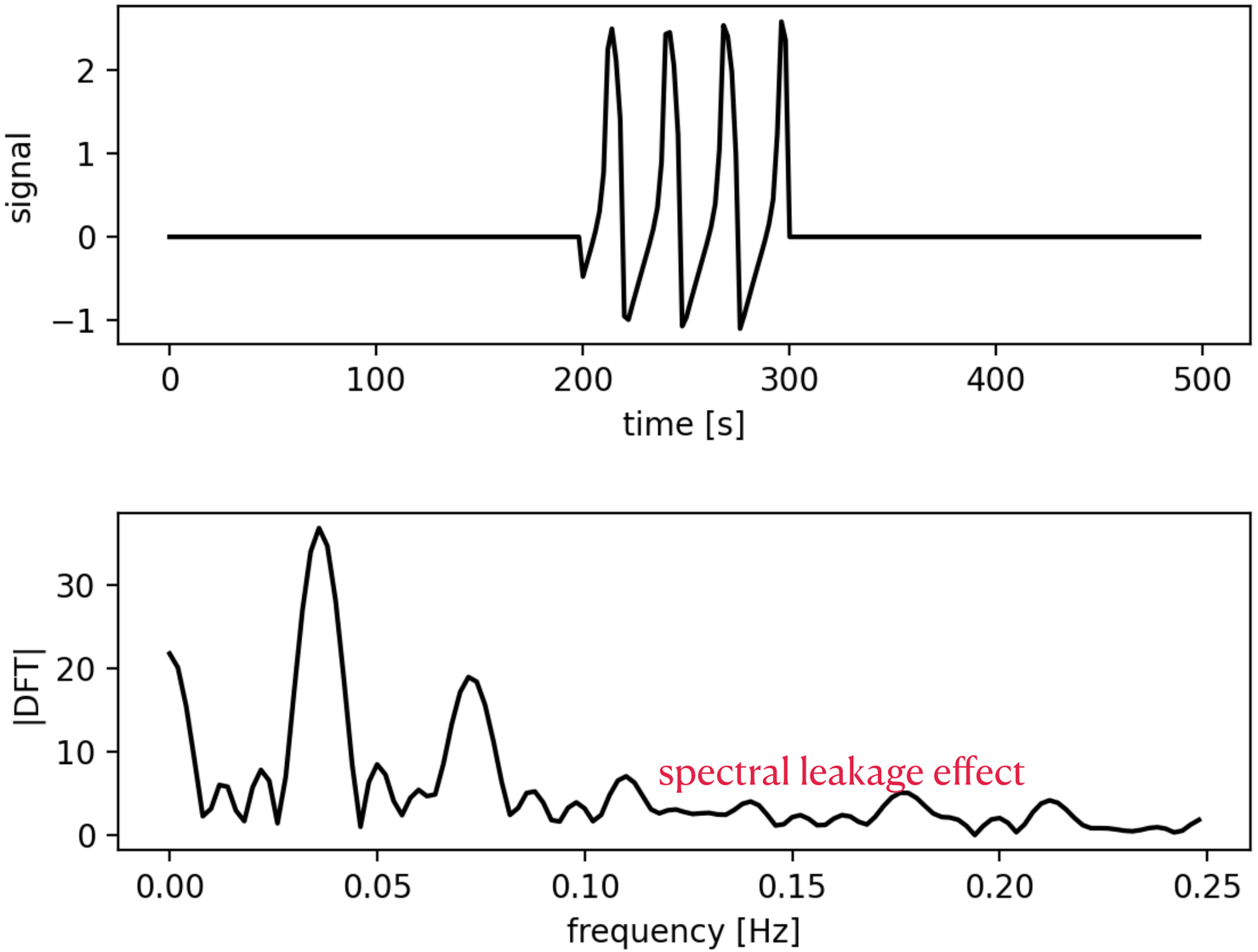


spectral leakage effect by zero-padding

Example: FHN-model

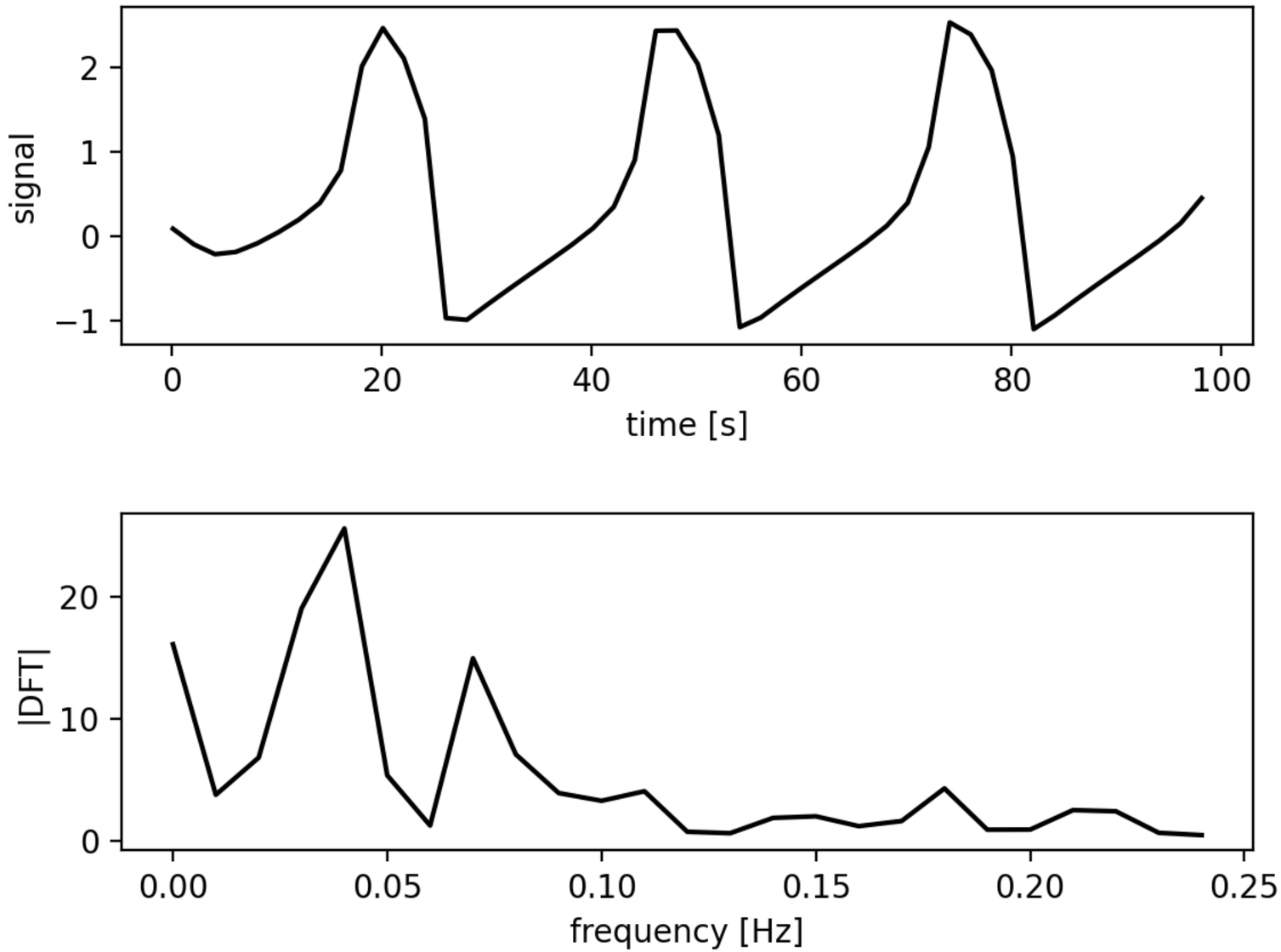


Example: FHN-model



Example: FHN-model

short signal for comparison

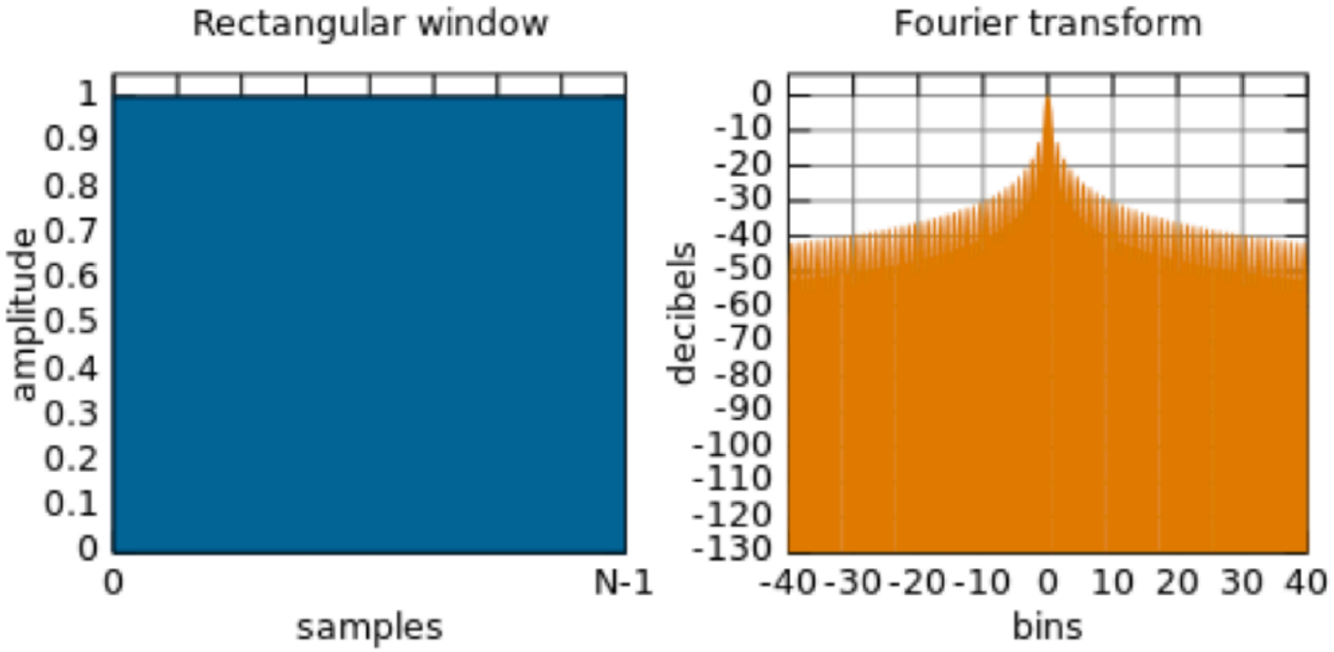


how to reduce spectral leakage:

- choose long time series
- avoid zero-padding

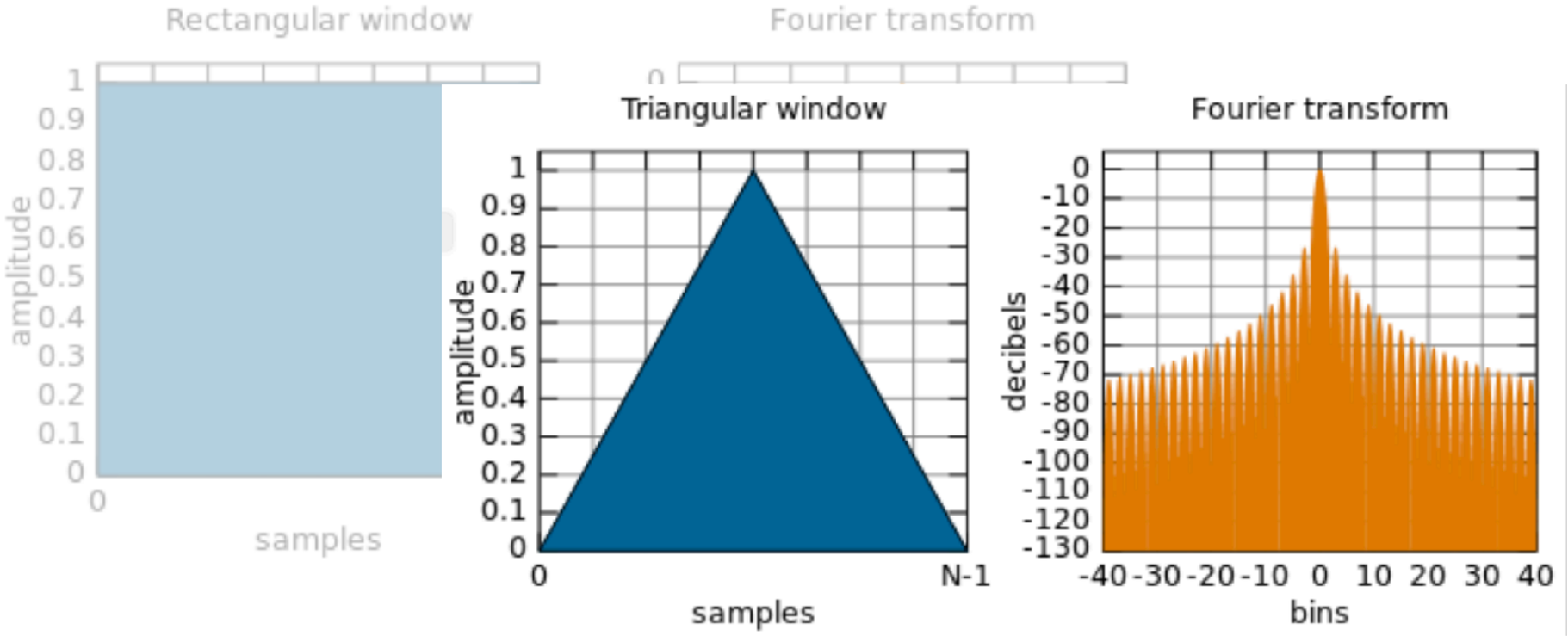
how to reduce spectral leakage:

- choose long time series
- avoid zero-padding
- choose better window function



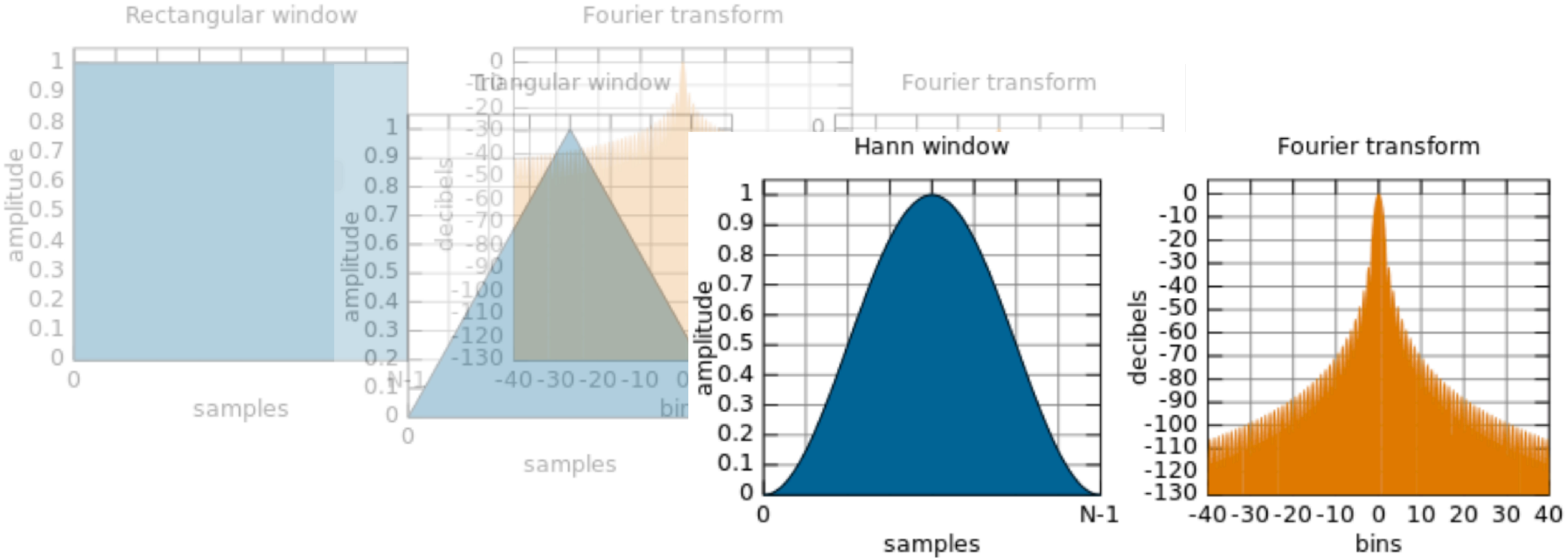
how to reduce spectral leakage:

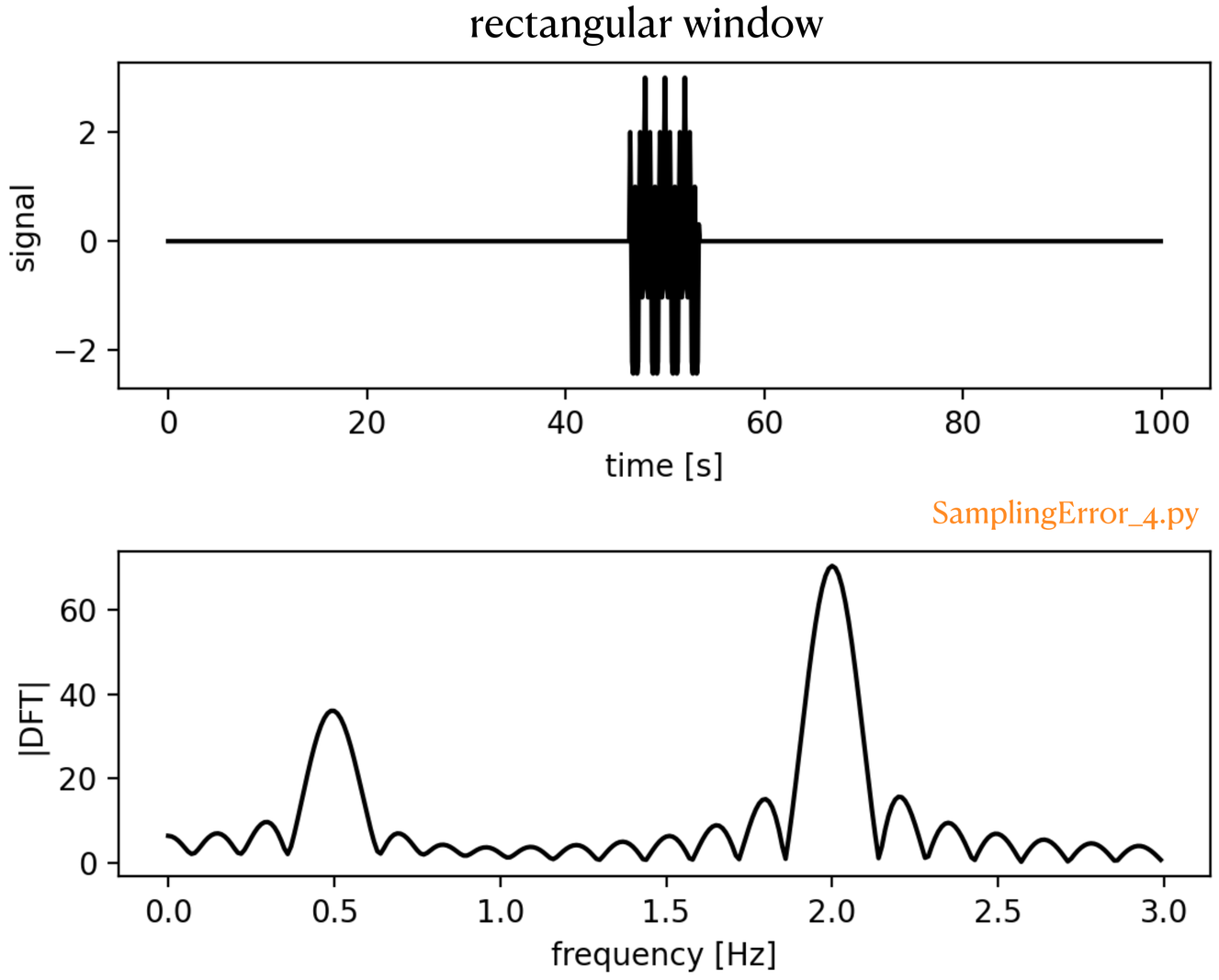
- choose long time series
- avoid zero-padding
- choose better window function

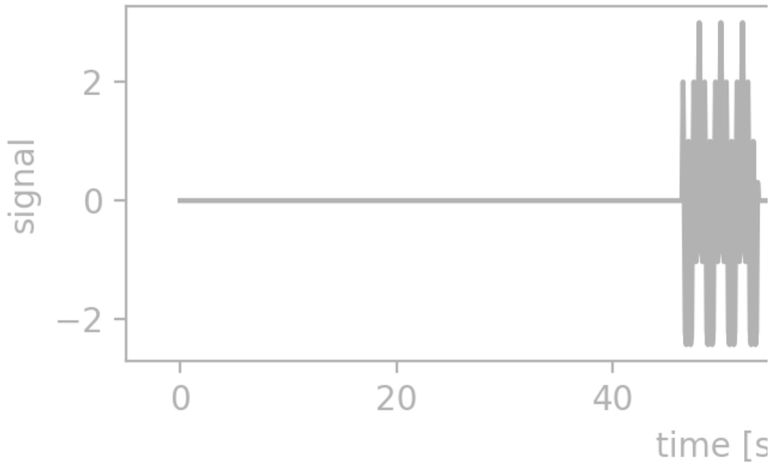


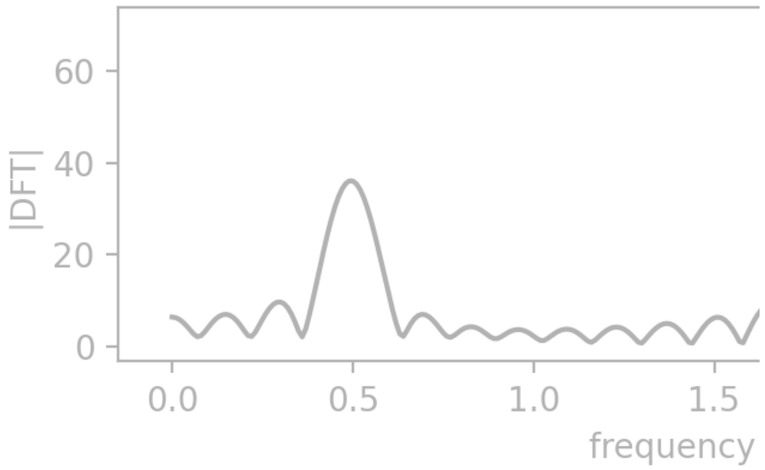
how to reduce spectral leakage:

- choose long time series
- avoid zero-padding
- choose better window function

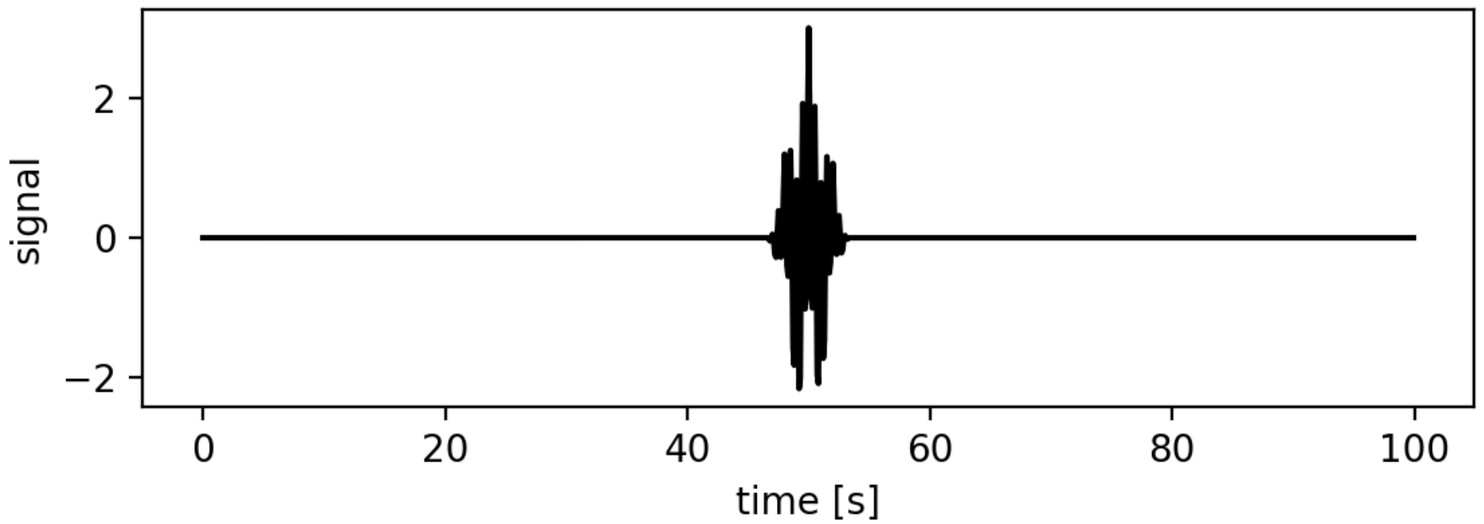



rectangular window
signal
2
0
−2
0
20
40
60
80
100
time [s]
SamplingError_4.py
|DFT|
60
40
20
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
frequency [Hz]

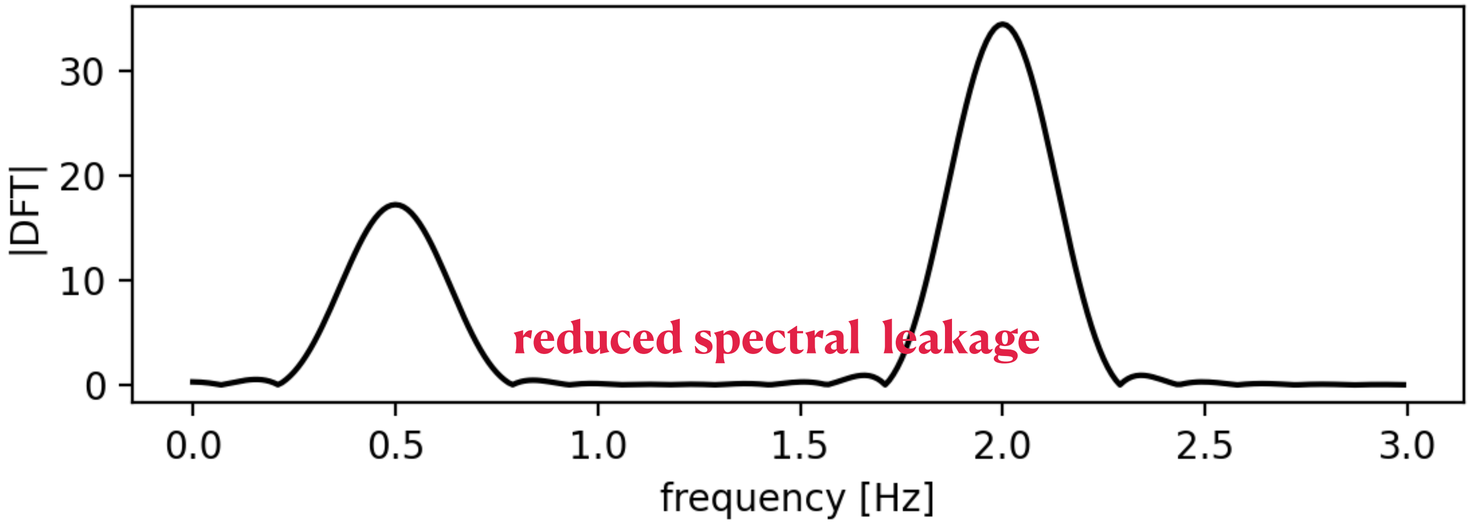




Hann window



SamplingError_4.py



summary:

if your signal is too short, then prolong with zeros (zero-padding)

and apply a specific window function (e.g. Hann window)

data sampling

Fourier analysis

errors in analysis aliasing spectral leakage **spectral power**

linear filters

time-frequency analysis