Lecture 3. Basic models

Alexander Zhigalov / Dept. of CS, University of Helsinki and Dept. of NBE, Aalto University

Outline / overview

- **Section 1.** Periodic time series
- **Section 2.** Non-periodic time series
- **Section 3.** Random time series
- **Section 4.** Autocorrelation
- **Section 5.** Power spectrum

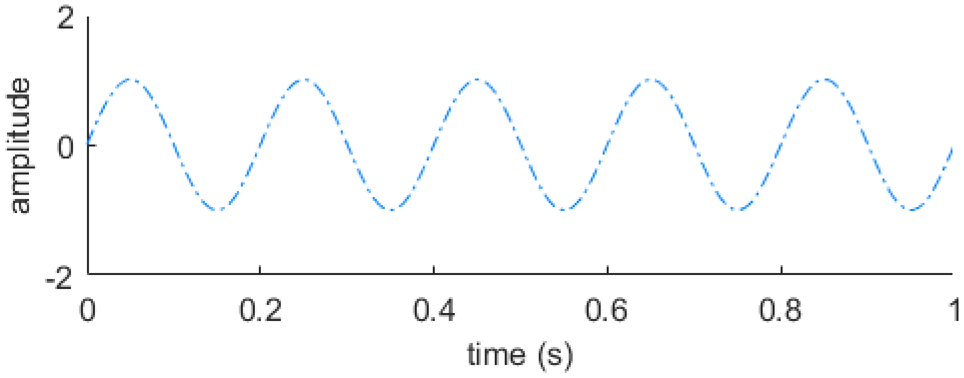
Section 1. Periodic time series

Periodic functions and their parameters (1/4)

Parameters of periodic functions

A – amplitude of signal
f – frequency of signal
phi – phase of signal

```
fs = 1000  # Hz
T = 1      # seconds
N = T * fs # duration or length

# init
t = np.linspace(0, T, N)

# function
X = A * np.sin(2 * np.pi * t * f + phi)
```




```
A = 1
f = 5
phi = 0
```

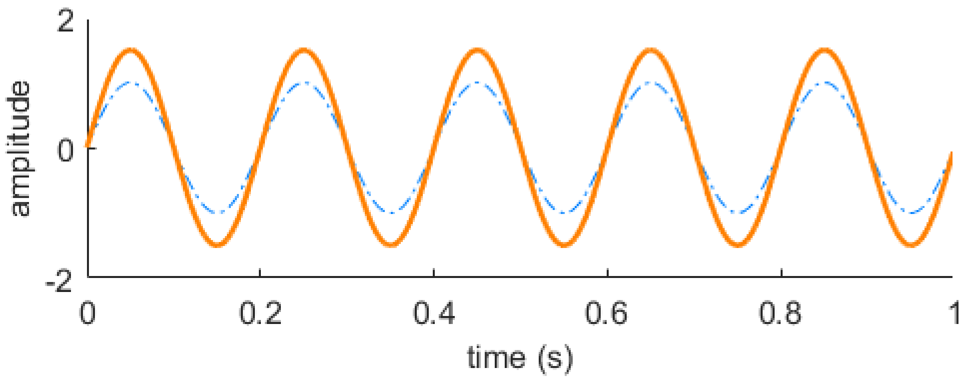
See, “L03_periodic_signal.py”

Periodic functions and their parameters (2/4)

Parameters of periodic functions

A – amplitude of signal
f – frequency of signal
phi – phase of signal

```
fs = 1000   # Hz
T = 1       # seconds
N = T * fs  # duration or length

# init
t = np.linspace(0, T, N)

# function
X = A * np.sin(2 * np.pi * t * f + phi)
```




```
A = 1.5
f = 5
phi = 0
```

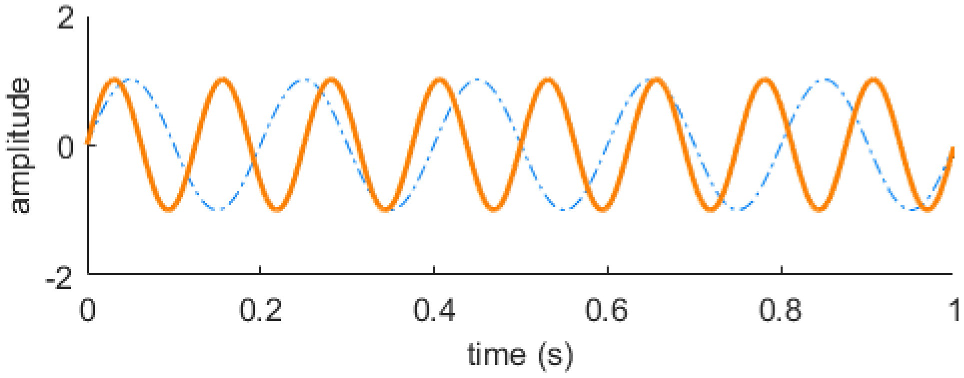
See, “L03_periodic_signal.py”

Periodic functions and their parameters (3/4)

Parameters of periodic functions

A – amplitude of signal
f – frequency of signal
phi – phase of signal

```
fs = 1000  # Hz
T = 1      # seconds
N = T * fs # duration or length

# init
t = np.linspace(0, T, N)

# function
X = A * np.sin(2 * np.pi * t * f + phi)
```




```
A   = 1
f   = 8
phi = 0
```

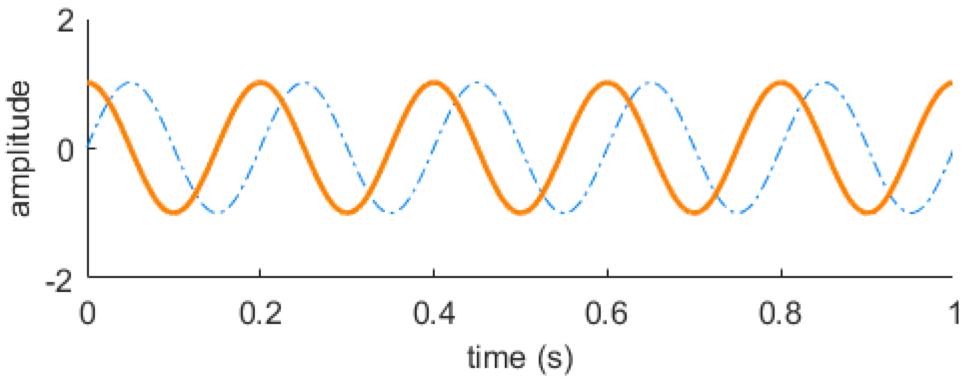
See, “L03_periodic_signal.py”

Periodic functions and their parameters (4/4)

Parameters of periodic functions

A – amplitude of signal
f – frequency of signal
phi – phase of signal

```
fs = 1000   # Hz
T = 1       # seconds
N = T * fs  # duration or length

# init
t = np.linspace(0, T, N)

# function
X = A * np.sin(2 * np.pi * t * f + phi)
```

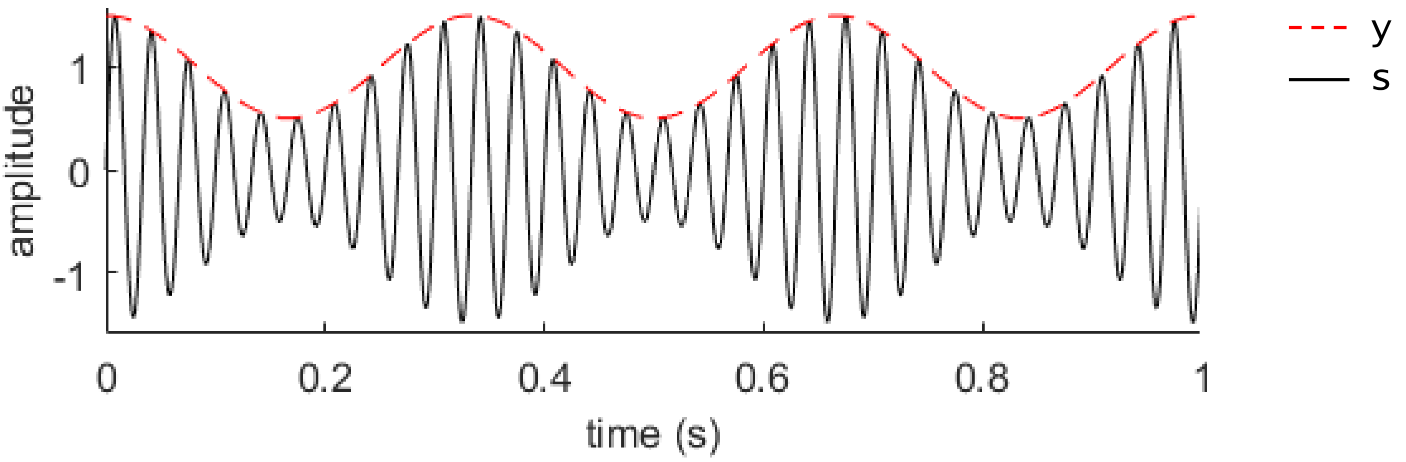



```
A = 1
f = 8
phi = pi/2
```

See, "L03_periodic_signal.py"

Amplitude modulation

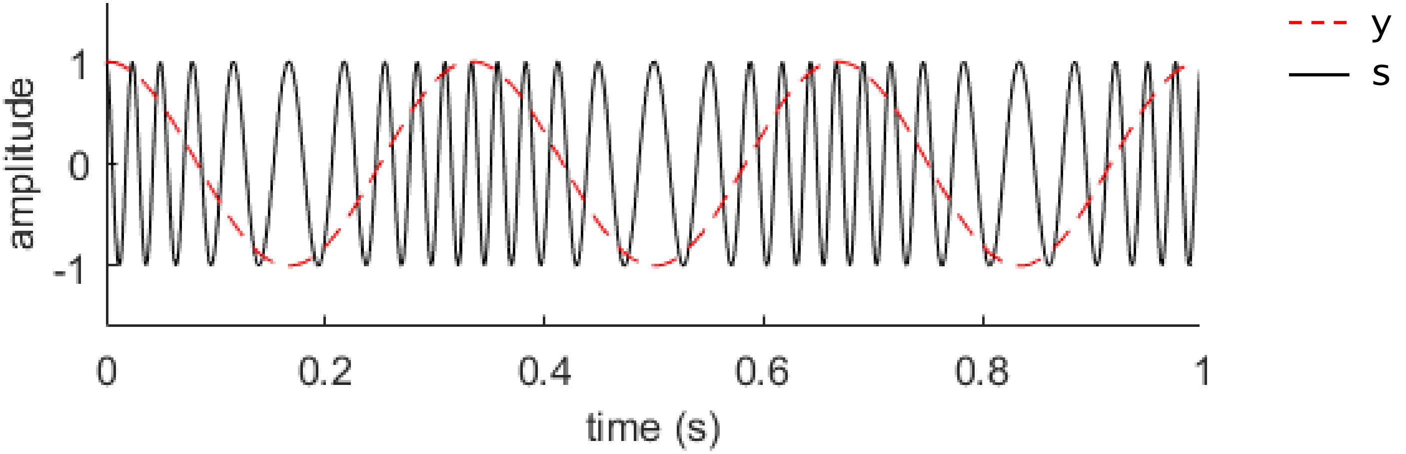
```
# parameters
fs = 1000   # sampling rate, in Hz
N  = 1000   # duration, in samples
T  = N / fs # duration, in seconds

f0 = 3      # signal frequency, in Hz
fc = 30     # carrier frequency, in Hz
AM = 0.5    # modulation factor

# time variable
t = np.linspace(0, T, N)

# amplitude envelope (message)
y = 1 + AM * np.cos(2 * np.pi * f0 * t)

# amplitude modulated signal
s = np.sin(2 * np.pi * fc * t) *
    (1 + AM * np.cos(2 * np.pi * f0 * t)
```



See, “L03_AM.py”

Frequency modulation

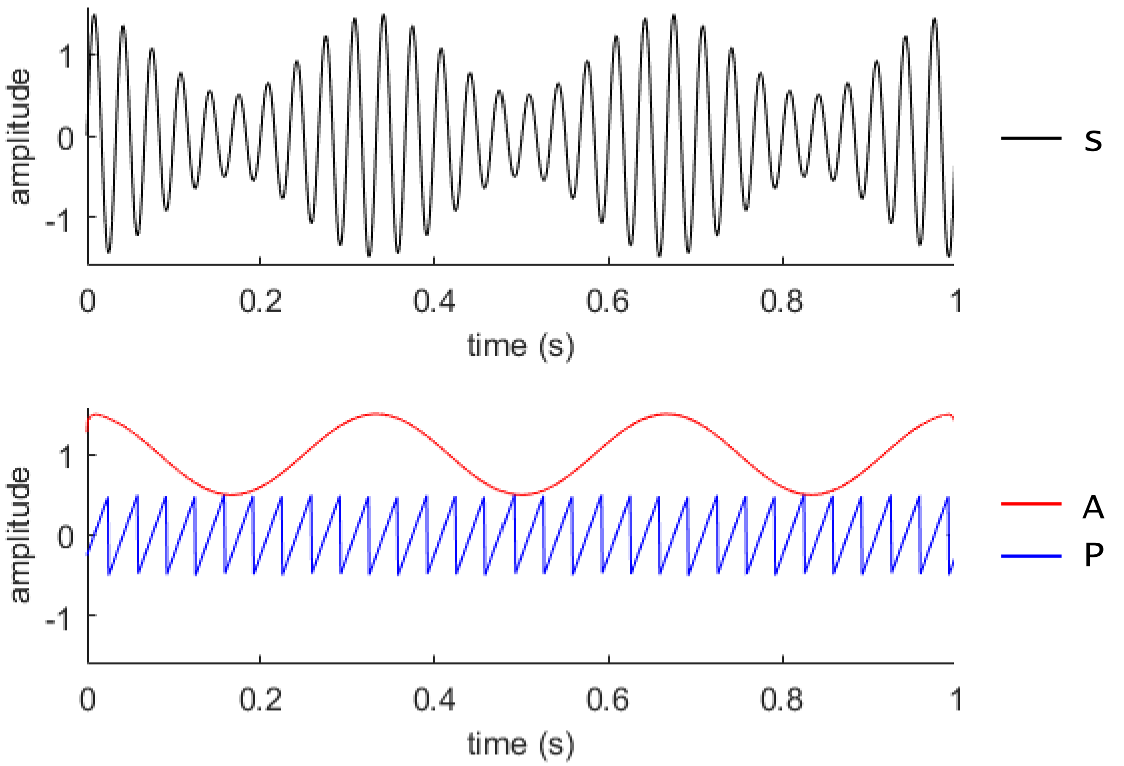
```python
# parameters
fs = 1000    # sampling rate, in Hz
N  = 1000    # duration, in samples
T  = N / fs  # duration, in seconds

f0 = 3       # signal frequency, in Hz
fc = 30      # carrier frequency, in Hz
FM = 5       # modulation factor

# time variable
t = np.linspace(0, T, N)

# message
y = np.sin(2.0 * np.pi * f0 * t)

# frequency modulated signal
s = np.cos(2.0 * np.pi * fc * t +
    FM * np.cos(2.0 * np.pi * f0 * t))
```



See, "L03_FM.py"

How to detect amplitude and phase of a signal?

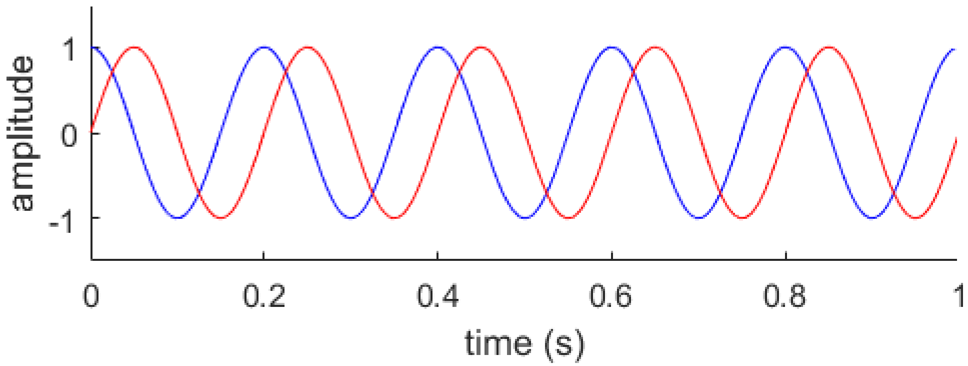
```python
import numpy as np
from scipy import signal

# s is an amplitude-modulated signal

# detect amplitude
A = np.abs(signal.hilbert(s))

# detect phase
P = np.angle(signal.hilbert(s))
```

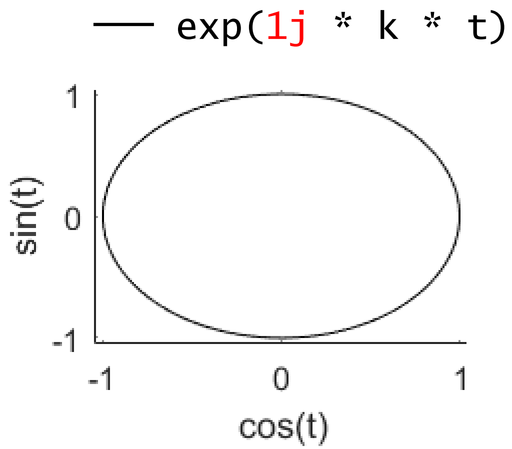


See, “L03_amplitude_and_phase.py”

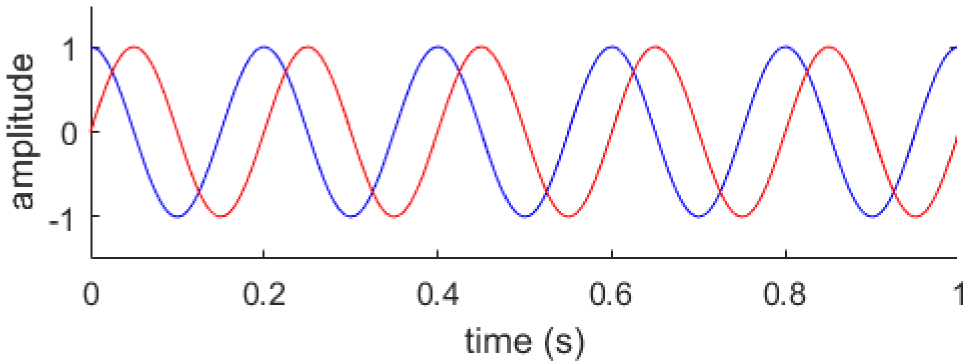
Complex-valued signal

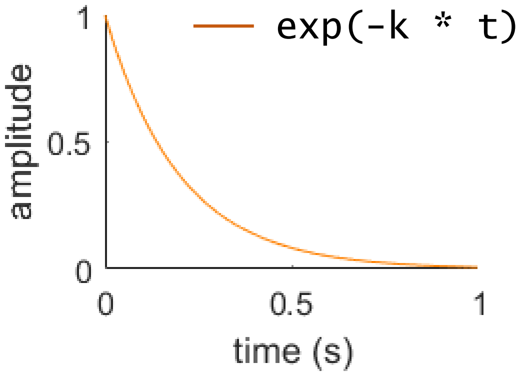
```
x1 = np.sin(2 * np.pi * f * t)

x2 = np.cos(2 * np.pi * f * t)
```






```
x3 = np.imag(
     np.exp(1j * 2 * np.pi * f * t))

x4 = np.real(
     np.exp(1j * 2 * np.pi * f * t))
```



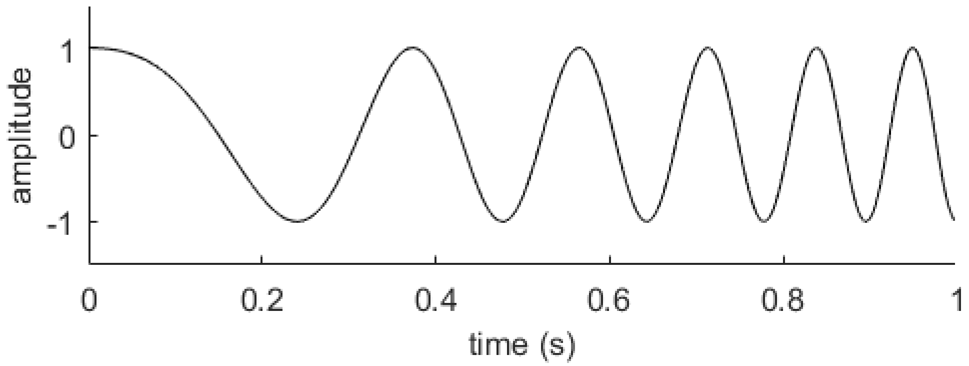


See, “L03_complex_value.py”

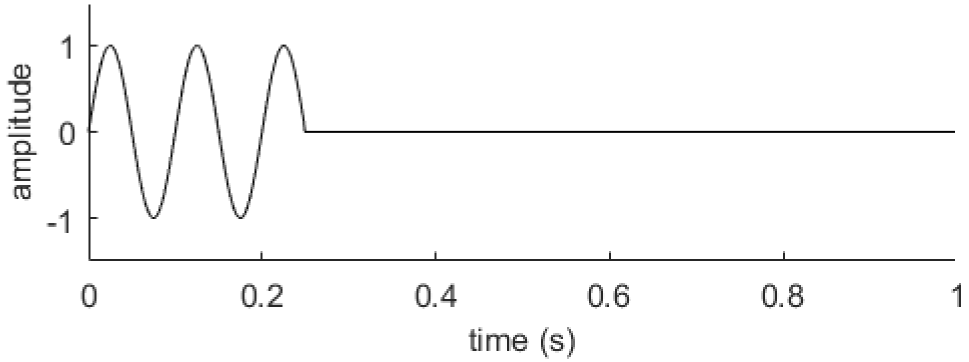
Section 2. Non-periodic time series

Non-periodic signals / non-stationary signals

```
# chirp signal
f0 = 1
f1 = 10
t1 = T
y = signal.chirp(t, f0, t1, f1)
```




```
# non-periodic signal
f0 = 10
u = np.sin(2 * np.pi * f0 * t)
u[int(N/4):] = 0
```



See, "L03_non_periodic_signal.py"

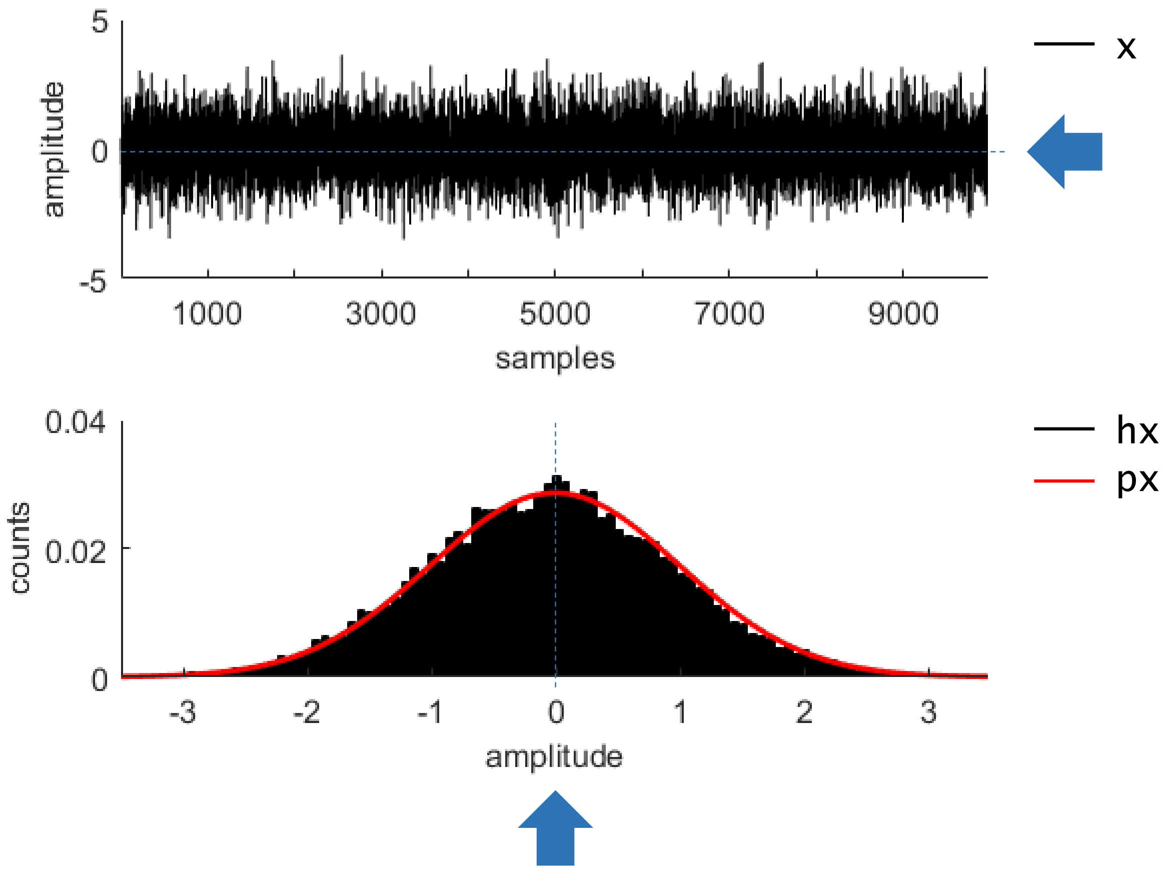
Section 3. Random time series

Random data with normal distribution (white noise, Gaussian noise)

- White or Gaussian noise is the same as random data with normal distribution.
- “White” originates from the fact that white color appears as a mixture of all other colors, and each color has certain frequency band. For instance, red light has frequency around $440 * 10^{12}$ Hz.
- Many theories of signal processing assumes that the noise presented in a system is Gaussian (or white) noise.

Random data with normal PDF and its parameters (mean, standard deviation)

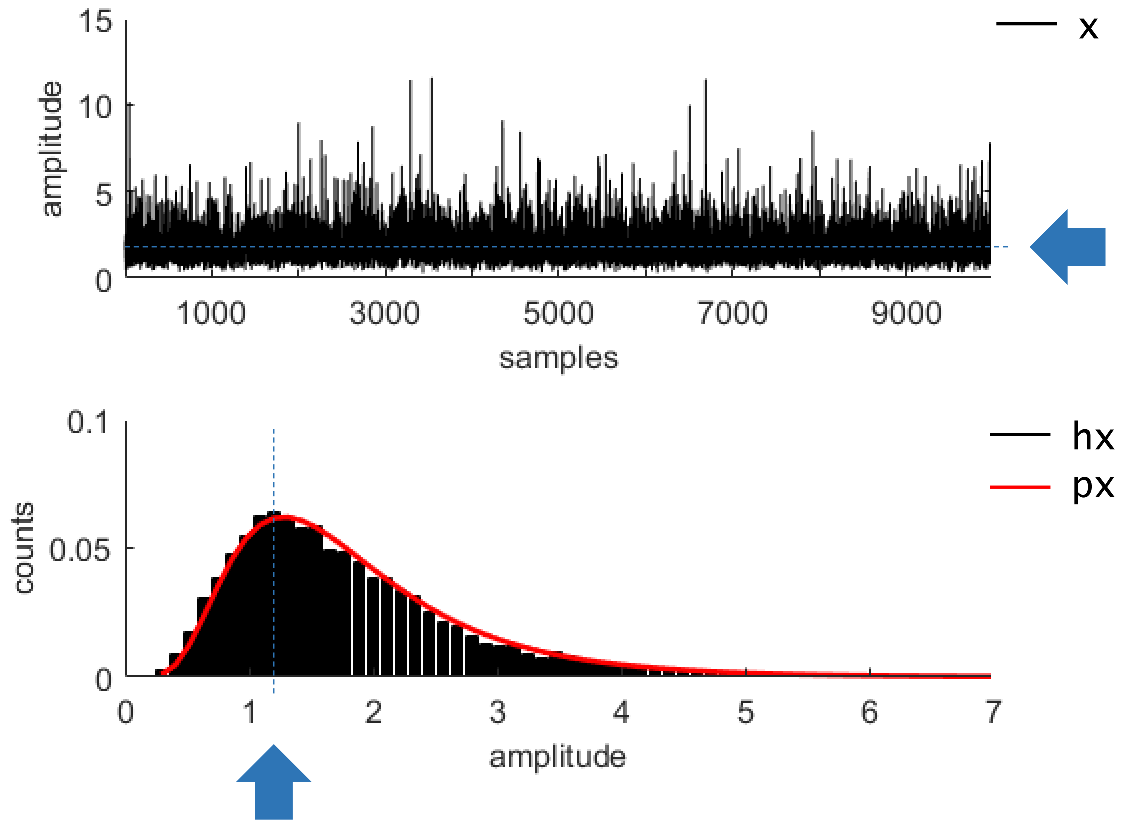
```
import numpy as np
from scipy import signal

# generate gaussian noise
N = 10000
x = np.random.randn(N) # mu = 0, std = 1

# histogram
bx = np.linspace(xmin, xmax, 100)
hx, bx = np.histogram(x, bx)

# pdf
mu, std = norm.fit(x)
px = norm.pdf(bx, mu, std)
```



See, “L03_noise.py”

Random data with lognormal PDF and its parameters (mean, standard deviation)

```
import numpy as np
from scipy import signal

# generate log gaussian noise
N = 10000
x = np.random.lognormal(0.5, 0.5, size=N)

# histogram
bx = np.linspace(xmin, xmax, 100)
hx, bx = np.histogram(x, bx)

# pdf
p0, p1, p2 = lognorm.fit(x)
px = lognorm.pdf(bx, p0, p1, p2)
```

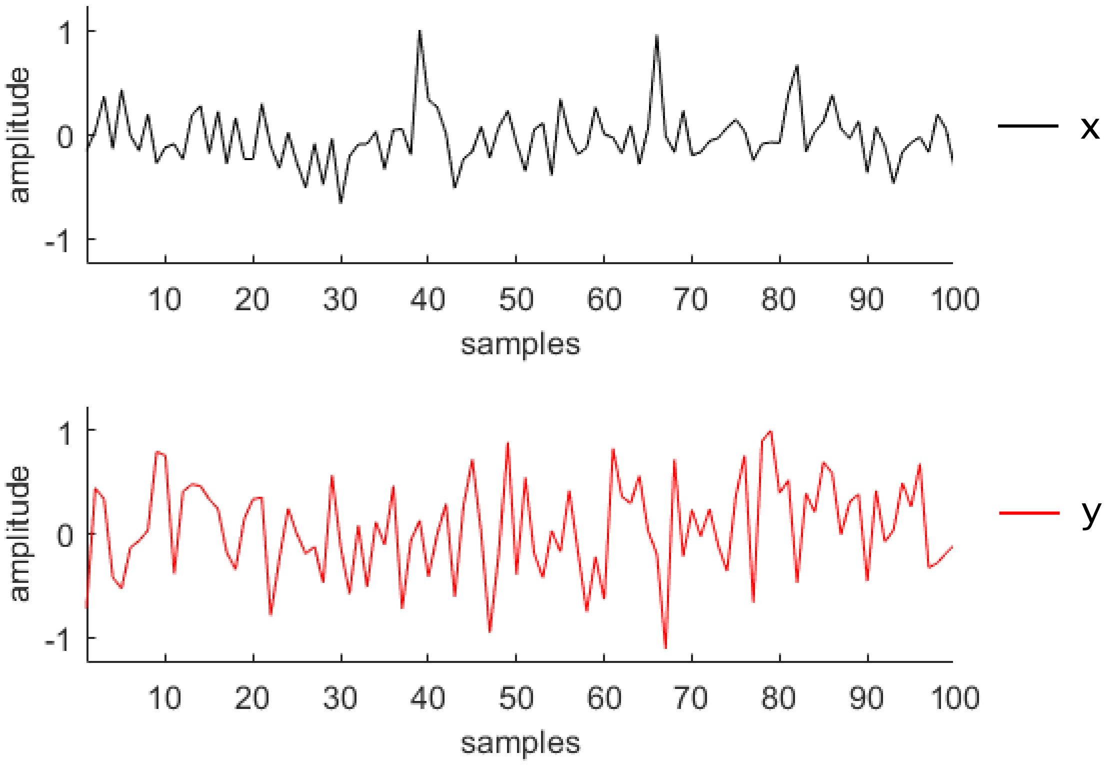


See, “L03_noise.py”

Section 4. Autocorrelation

What is correlation? (1/2)

```
# mean value
x_mu = 0.0
for i in range(0, N):
  x_mu += x[i] / N

# standard deviation
x_sd = 0.0
for i in range(0, N):
  x_sd += ((x[i] - x_mu) ** 2) / N
x_sd = np.sqrt(x_sd)

# correlation coefficient (r)
r = 0.0
for i in range(0, N):
  r += ((x[i] - x_mu) * (y[i] - y_mu)) /
       (x_sd * y_sd) / N

# in case of zero mean and unit variance
r = 0.0
for i in range(0, N):
  r += (x[i] * y[i]) / N # mean of product
```

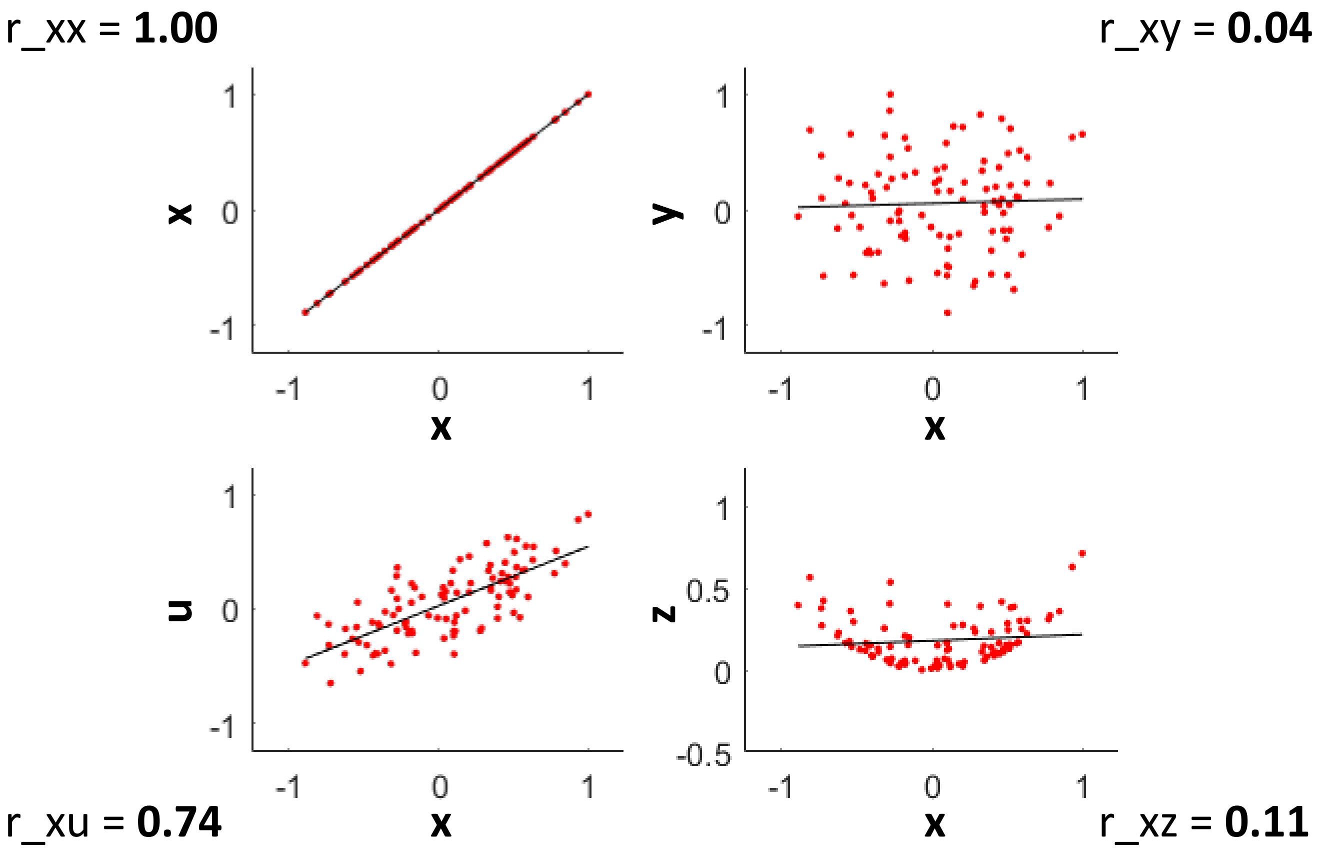



```
# correlation coefficient using numpy
r_np = np.corrcoef(x, y)[0, 1]
```

See, “L03_correlation.py”

What is correlation? (2/2)

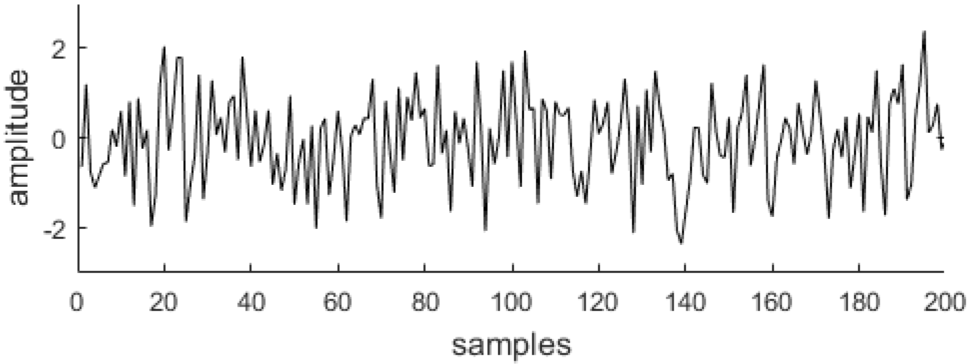
```
# random variables
N = 100
x = np.random.randn(N)
y = np.random.randn(N)

# dependencies between variables
u = (x + y) / 2             # linear
z = (x ** 2 + y ** 2) / 4 # non-linear

# correlation
r_xx = np.corrcoef(x, x)[0, 1]
r_xy = np.corrcoef(x, y)[0, 1]
r_xu = np.corrcoef(x, u)[0, 1]
r_xz = np.corrcoef(x, z)[0, 1]
```

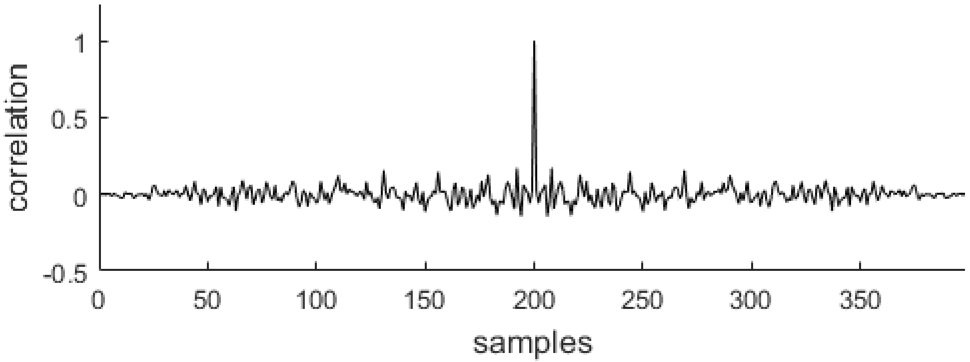


See, “L03_dependencies.py”

What is autocorrelation? (1/2)

```
import numpy as np
from scipy import signal

# generate gaussian noise
N = 100
x = np.random.randn(N)

# compute ACF
rx = signal.correlate(x, x)
rx = rx / np.max(rx)
```

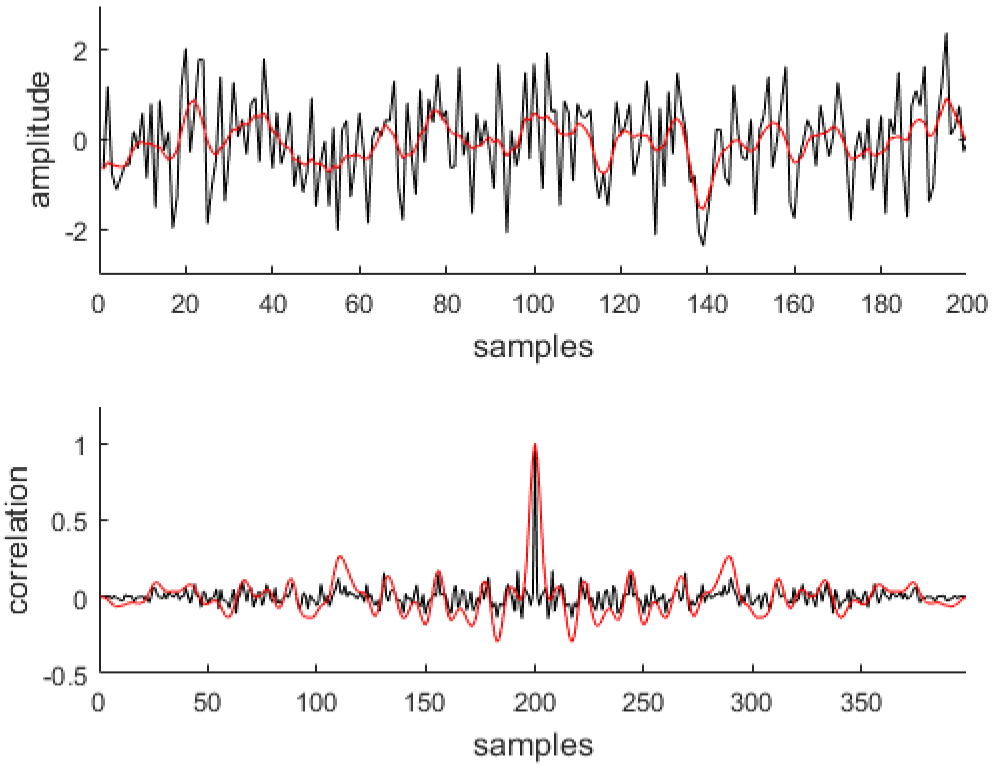




See, “L03_acf.py”

What is autocorrelation? (2/2)

```
import numpy as np
from scipy import signal

# generate gaussian noise
N = 100
x = np.random.randn(N)

# compute ACF
rx = signal.correlate(x, x)
rx = rx / np.max(rx)

# smooth signal by 4 neighboring points
y = signal.filtfilt(np.ones(4) / 4, 1, x)

# compute ACF
ry = signal.correlate(y, y)
ry = ry / np.max(ry)
```

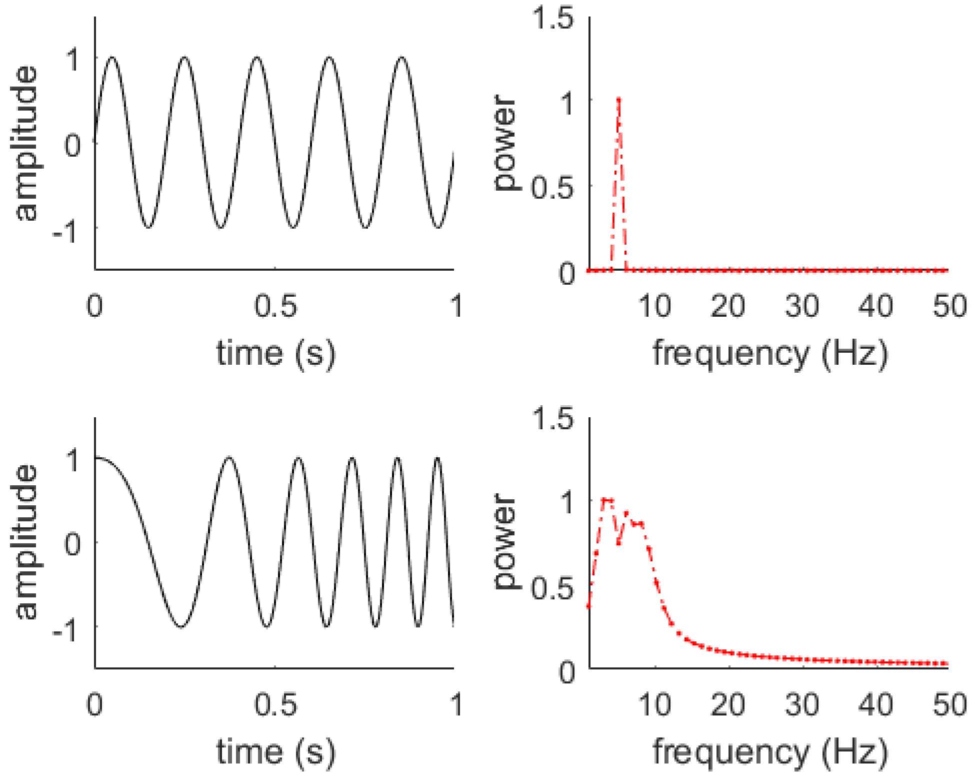


See, “L03_acf.py”

Section 5. Power spectrum

Power spectrum (1/2)

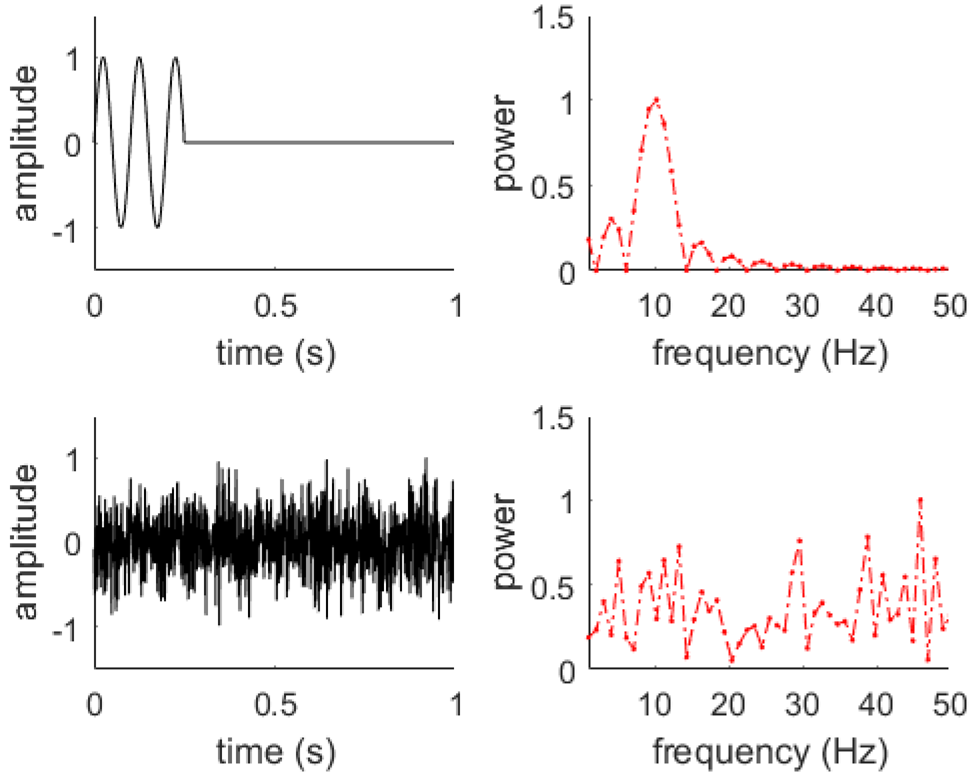
```
from scipy.fftpack import fft

# sin signal
f0 = 5
x1 = np.sin(2 * np.pi * f0 * t)

# power spectrum
y1 = np.abs(fft(x1))


# chirp signal
f0 = 1
f1 = 10
t1 = T
x2 = signal.chirp(t, f0, t1, f1)
```



See, “L03_power_spectrum.py”

Power spectrum (2/2)

```
# non-periodic signal
f0 = 10
x3 = np.sin(2 * np.pi * f0 * t)
x3[int(N/4):] = 0
```




```
# random noise
x4 = np.random.randn(N)
```

See, “L03_power_spectrum.py”

Literature

- **Python programming language**

- http://www.scipy-lectures.org/, see “materials/L02_ScipyLectures.pdf”