Auto-correlation & Cross-correlation: Visualization with Matlab and Python

Tiep M. H.

Notations: $\mathbb{R}^{m\times n}$ denotes the real field that includes all real-valued matrices of size $m \times n$; $\mathbb{C}^{m\times n}$ denotes the complex field that includes all complex-valued matrices of size $m \times n$; The operation $\text{diag}\left([z_1, \dots, z_K]\right)$ diagonalizes a row vector $[z_1, \dots, z_K]$ into a diagonal matrix; Bold lowercase letters and bold uppercase letters denote vectors and matrices, respectively; $\mathbf{I}_n$ denotes the identity matrix of size $n \times n$; The upperscripts $(\cdot)^\top$, $(\cdot)^*$, and $(\cdot)^\dagger$ represent the transpose, conjugate, and Hermitian operators, respectively; $\mathfrak{R}\{\cdot\}$ denotes the real part of a complex-valued matrix; $\mathfrak{I}\{\cdot\}$ denotes the imaginary part of a complex-valued matrix.

I. Definitions

A. Auto-Correlation of Two Continuous Functions

We use auto-correlation to measure the **self-similarity** of a function $x(t)$. There are two types of definitions as follows:

- If the waveform of $x(t)$ is infinite, the auto-correlation function of $x(t)$ can be defined as

$$C_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} \overline{x(t)}\; x(t-\tau)dt, \tag{1}$$

  where $\overline{x(t)}$ is the conjugate value of $x(t)$. Suppose $x(t)$ is a periodic waveform with the period T. In this case, (1) is replaced by

$$C_{xx}(\tau) = \frac{1}{T} \int_{t_0-T/2}^{t_0+T/2} \overline{x(t)}\; x(t-\tau)dt = \frac{1}{T} \int_{t_0}^{t_0+T} \overline{x(t)}\; x(t-\tau)dt, \tag{2}$$

  where t_0 is arbitrary.

- If the waveform of $x(t)$ is finite, i.e. $x(t) = 0$ for $t < t_1$ or $t_2 > t$, then the auto-correlation function of $x(t)$ can be defined as

$$C_{xx}(\tau) = \int_{t_1}^{t_2} \overline{x(t)}\; x(t-\tau)dt. \tag{3}$$

B. Auto-Correlation of Two Discrete Functions

In the case of discrete functions, we can simplify (3)–(2) into the following definitions:

- For an infinite sequence:

$$C_{xx}(k) = \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} \overline{x(m)}\; x(m-k) \tag{4}$$

- For a finite sequence:

$$C_{xx}(k) = \sum_{m=M_{\text{lower}}}^{M_{\text{upper}}} \overline{x(m)}\; x(m-k), \tag{5}$$

where M_{lower} and M_{upper} are drawn from the following table:

	M_{lower}	M_{upper}	Note
In Matlab	$\max(1, 1+k)$	$\min(L_x, L_y+k)$	Each Matlab array starts with the index 1.
In Python	$\max(0, k)$	$\min(L_x-1, L_y+k-1)$	Each Python array starts with the index 0.
	L_x is the length of the sequence $\{x(m)\}$		

C. Cross-Correlation of Two Discrete Functions

In the case of discrete functions, we can generalize (4)–(5) to the following definitions:

- For two infinite sequences:

$$C_{xy}(k) = \lim_{M\to\infty} \frac{1}{2M+1} \sum_{m=-M}^{M} \overline{x(m)}\; y(m-k) \tag{6}$$

- For two finite sequences:

$$C_{xy}(k) = \sum_{m=M_{\text{lower}}}^{M_{\text{upper}}} \overline{x(m)}\; y(m-k), \tag{7}$$

where M_{lower} and M_{upper} are drawn from the following table:

	M_{lower}	M_{upper}	Note
In Matlab	$\max(1, 1+k)$	$\min(L_x, L_y+k)$	Each Matlab array starts with the index 1.
In Python	$\max(0, k)$	$\min(L_x-1, L_y+k-1)$	Each Python array starts with the index 0.
	L_x is the length of the sequence $\{x(m)\}$		
	L_y is the length of the sequence $\{y(m)\}$		

II. An example of cross-correlation

Let us consider two following sequences:

t	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
$x(t)$	1.2	0.7	-0.6	1.4	-1.1	0.3	1.8	-0.2	1.2				
$y(t)$				0.5	2.2	0.7	1	-0.3	1.4	1.5	0.1	-0.8	-1.1

The below figure depicts $x(t)$ and $y(t)$ versus t.

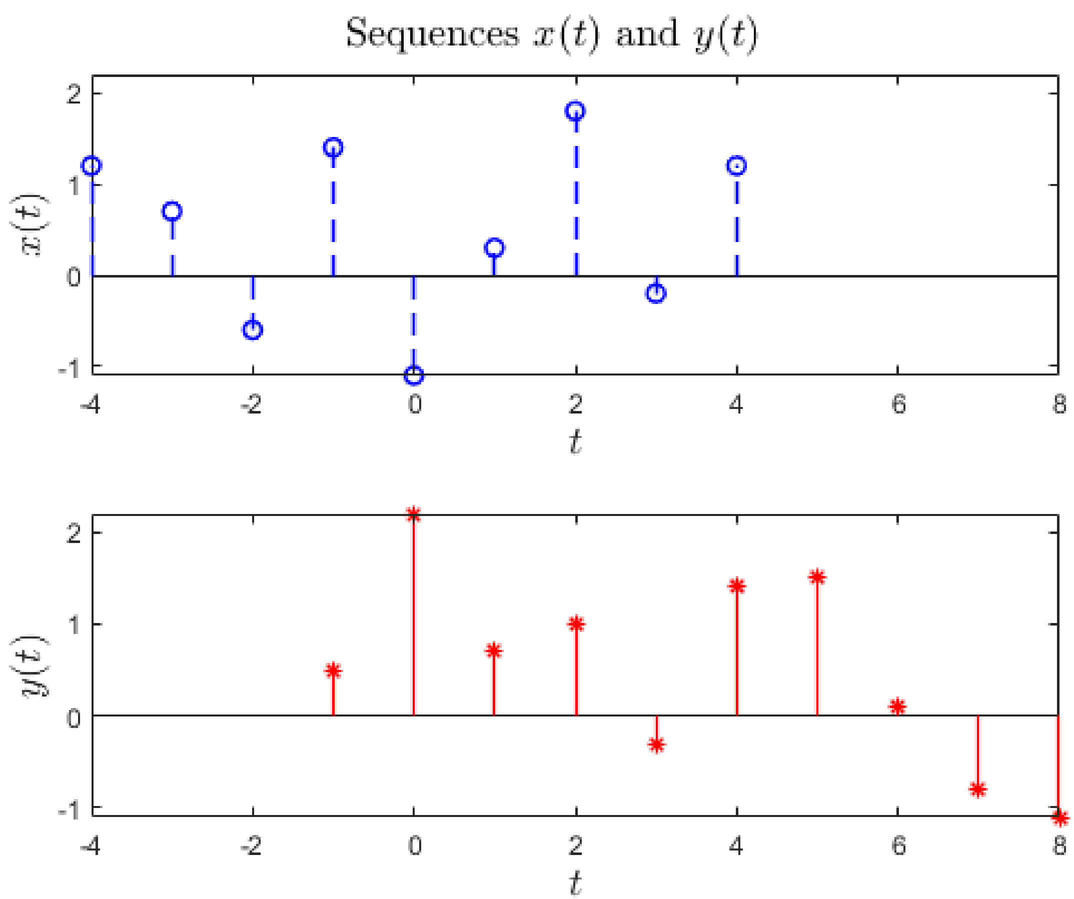

Fig. 1. $x(t)$ and $y(t)$ versus t. As for $x(t)$, there are $L_x = 9$ non-zero values. As for $y(t)$, there are $L_y = 10$ non-zero values.

In order to program the computation of cross-correlation, we first need to rewrite the functions $x(t)$ and $y(t)$ as the arrays $x(m)$ and $y(m)$ so that the first element of $x(m)$ is aligned with the first element of $y(m)$. From a programming perspective, if m is the index of an array, then the first element of $x(m)$ is put the index $m = 1$ (in Matlab) or $m = 0$ (in Python). Also, the first element of $y(m)$ is put at the index $m = 1$ (in Matlab) or $m = 0$ (in Python). This means that we align $x(m)$ and $y(m)$ so that their first elements have the same first index of an array. The following figure depicts the arrangement of the arrays $x(m)$ and $y(m)$ in Matlab.

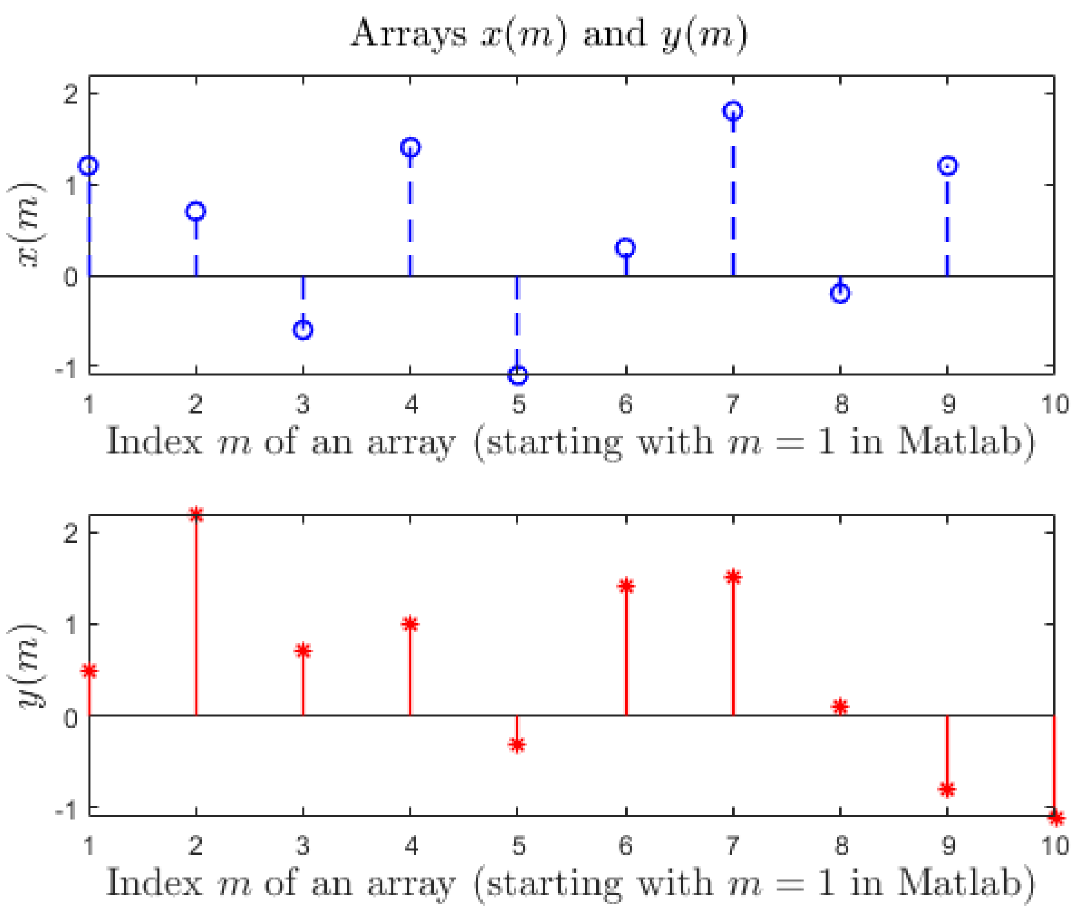

Fig. 2. $x(m)$ and $y(m)$ versus the index m. Every Matlab array starts with the index $m = 1$.

Now, we want to compute the cross-correlation values at different lags. Suppose that we are given a lag k, what we need to do is to keep $x(m)$ and then shift $y(m)$ by $|k|$ positions. The following figure depicts 2 scenarios, where $y(m)$ is shifted by 10 positions to the left and by 9 positions to the right.

It is noticeable that if $|k|$ is sufficiently large, the product $\overline{x(m)}\ y(m-k)$ becomes 0. We do not want to perform an **infinite** number of operations for which we can easily know their results. For example, the equation (7) can be expressed as

$$\begin{aligned} C_{xy}(k) &= \sum_{m=-\infty}^{+\infty} \overline{x(m)}\ y(m-k) \\ &= \underbrace{\ldots+0+0+\ldots+0}_{\text{redundant operations}} + \sum_{m=M_{\text{lower}}}^{M_{\text{upper}}} \underbrace{\overline{x(m)}\ y(m-k)}_{\text{non-zero}} + \underbrace{0+\ldots+0+0+\ldots}_{\text{redundant operations}} \end{aligned} \tag{8}$$

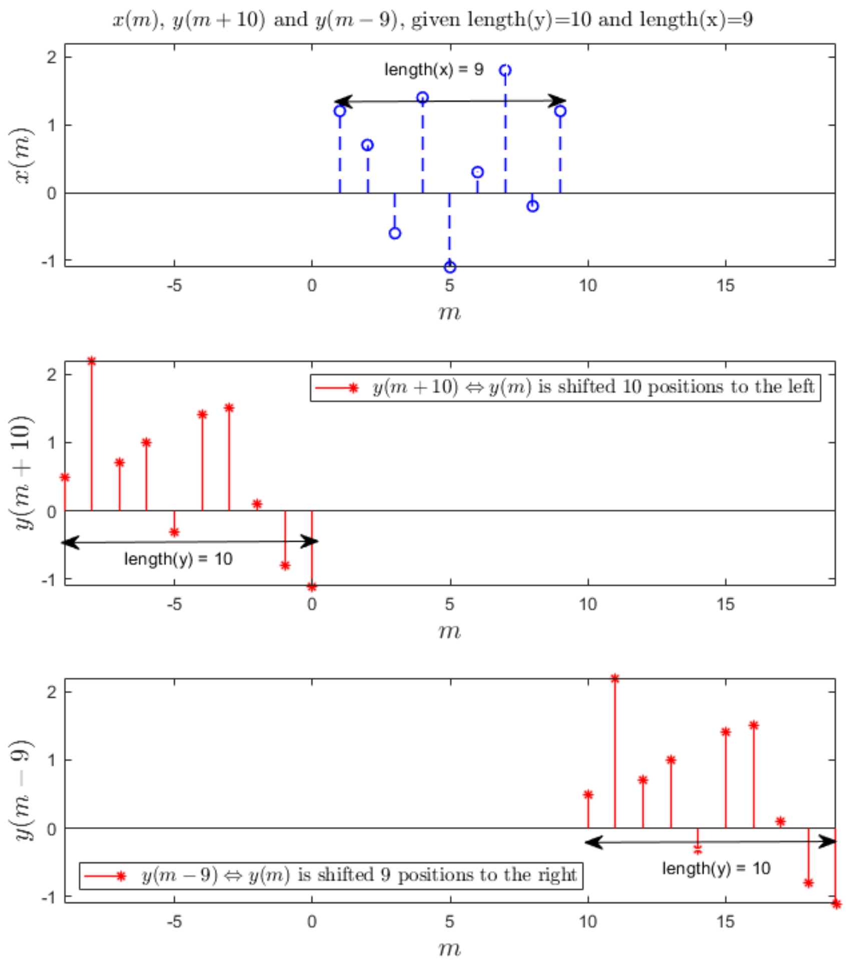

Fig. 3. We shift $y(m)$ by 10 positions to the left and 9 positions to the right. The resultant array is $y(m-k)$, where $k=-10<0$ if we shift $y(m)$ to the left and $k=+9>0$ if we shift $y(m)$ to the right.

To save the computational resources (memory, FLOPS, etc), we only need to compute the sum $\sum_{m=M_{\text{lower}}}^{M_{\text{upper}}} \overline{x(m)}\ y(m-k)$. Thus, the most important thing is to find a suitable range of m. This means that we will find M_{lower} and M_{upper}.

Recall that m indicates the index of an array. Thus, M_{lower} and M_{upper} are the minimum index and the maximum index, respectively. Since the first index of an array in Matlab is different that in Python, we will find M_{lower} and M_{upper} specifically for each programming language.

Let us consider Matlab for which each array starts with the index $m = 1$ (see Figure II). Shifting $y(m)$ to the left until the last element of $y(m)$ just passes the first element of $x(m)$ by only 1 position (see the 2nd sub-figure in Figure II), we can see that M_{lower} should be chosen to be $M_{\text{lower}} = \max(1, 1+k)$. This is because $\overline{x(m)}\ y(m-k)$ starts to become **NON-ZERO** from the position $m \geq \max(1, 1+k)$. Similarly, shifting to the right until the first element of $y(m)$ just passes the last element of $x(m)$ by only 1 position (see the 3rd sub-figure in Figure II), we can choose $M_{\text{upper}} = \min(L_x, L_y + k)$. This is because, $\overline{x(m)}\ y(m-k)$ remains non-zero when $m \leq \min(L_x, L_y + k)$.

Note that we can also find M_{lower} and M_{upper} in Python, based on shifting $y(m)$ to the left/right side to see how far $y(m)$ should go.

III. VISUALIZATION WITH MATLAB

```
%% This example demonstrates how to build a cross-corr. function without using Matlab's xcorr function
clear all; clc;

x   = [1.2, 0.7, -0.6, 1.4, -1.1, 0.3, 1.8, -0.2, 1.2];
y   = [0.5, 2.2, 0.7, 1, -0.3, 1.4, 1.5, 0.1, -0.8, -1.1];

%% Using my function
[corrs, lags] = corr_ALL_lags(x, y);

%% Using the Matlab built-in function
[corr_builtIn, lag_builtIn] = xcorr(x, y);

%% Compare the results
figure()
sgtitle("Compare my function to Matlab's xcorr", 'fontsize', 15)
plot(lag_builtIn, corr_builtIn, 'rx', 'LineWidth', 1.4, 'MarkerSize', 16)
hold on
plot(lags, corrs, '--bo', 'LineWidth', 1.4)
grid on
xlabel('Lag', 'fontsize', 12)
ylabel('Cross--correlation', 'fontsize', 12)
legend("Matlab's built-in function xcorr", "My function", ...
      'fontsize', 10, 'Location', 'southeast')

%% Local functions
function corr_GIVEN_lag_k = corr_GIVEN_a_lag(x,y,lag_k)
%Given the k-th lag, we calculate the cross-correlation
%
```

```
%                        c(lag_k) = sum_m  x(m) * y(m-lag_k)
%
% If lag_k > 0, then y shifts to the right.
% If lag_k < 0, then y shifts to the left.
% NOTE: the index of the 1st element in a Matlab array is 1.
% In another language, such as Python, the index of the 1st element in an array is 0.
% It is important to find the range of m in the loop ''for m=m_lower_bound:m_upper_bound''
    x_len = length(x);
    y_len = length(y);
    m_lower_bound = max(1, 1 + lag_k);
    m_upper_bound = min(x_len, y_len + lag_k);
    corr_GIVEN_lag_k = 0;
    for m=m_lower_bound:m_upper_bound
        corr_GIVEN_lag_k = corr_GIVEN_lag_k + x(m)' * y(m - lag_k);
    end
end

function [corr_at_all_lag_positions, lags] = corr_ALL_lags(x,y)
    x_len = length(x);
    y_len = length(y);
    corrs = zeros(1, x_len + y_len + 1);
    % Let y_first be the 1st element in y(m).
    % Let y_last be the last element in y(m).
    %% if we shift y(m) to the right, y_first passes x(m) after x_len steps
    lag_upper_bound = x_len;
    %% if we shift y(m) to the left, y_last passes x(m) after y_len steps
```

```
    lag_lower_bound = - y_len;
    %% The range of lag should be -y_len <= lag <= x_len
    lags = lag_lower_bound:lag_upper_bound;
    %% calculate the value of cross-corr. for each specific lag
    for i=1:length(lags)
        lag_k = lags(i);
        corrs(i) = corr_GIVEN_a_lag(x,y,lag_k);
    end
    corr_at_all_lag_positions = corrs;
end
```

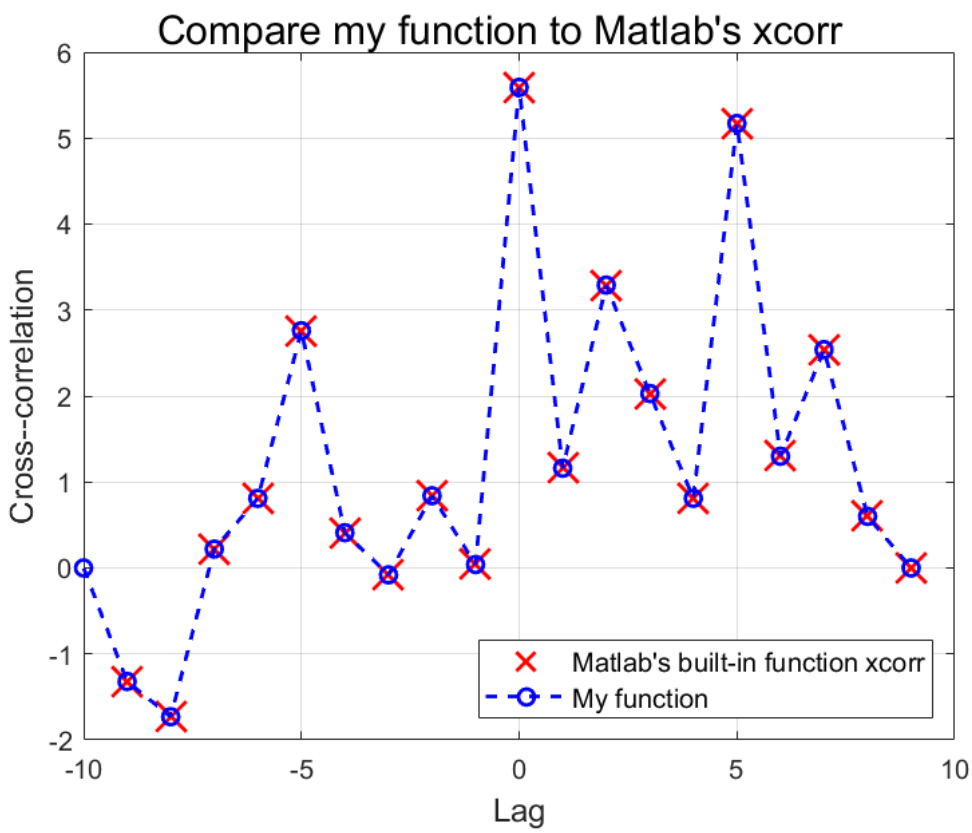

Fig. 4. Visualization with Matlab.

IV. Visualization with Python

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator


def corr_GIVEN_a_lag(x, y, lag_k):
    x_len = len(x)
    y_len = len(y)
    m_lower_bound = max(0, lag_k)
    m_upper_bound = min(x_len - 1, y_len + lag_k - 1)
    corr_GIVEN_lag_k = 0
    for m in range(m_lower_bound, m_upper_bound+1):
        corr_GIVEN_lag_k = corr_GIVEN_lag_k + x[m].conj() *
         ↪  y[m-lag_k]
    return corr_GIVEN_lag_k


def corr_ALL_lags(x, y):
    x_len = len(x)
    y_len = len(y)
    corrs = np.zeros(x_len + y_len + 1)
    lag_upper_bound = x_len
    lag_lower_bound = - y_len
    lags = [lag for lag in range(lag_lower_bound,
     ↪  lag_upper_bound+1)]
    for k in range(len(lags)):
        lag_k = lags[k]
        corrs[k] = corr_GIVEN_a_lag(x, y, lag_k)
    corr_at_all_lag_positions = corrs
    return corr_at_all_lag_positions, lags


""" Define 2 sequences and find their cross-corr. values """
x = np.array([1.2, 0.7, -0.6, 1.4, -1.1, 0.3, 1.8, -0.2, 1.2])
y = np.array([0.5, 2.2, 0.7, 1, -0.3, 1.4, 1.5, 0.1, -0.8,
 ↪  -1.1])

[corrs, lags] = corr_ALL_lags(x, y)

""" Compare the results """
ax = plt.figure().gca()
plt.plot(lags, corrs, '--bo',
         markerfacecolor='none', label='My function')
plt.xlim((-10, 10))
plt.ylim((-2, 6))
plt.xlabel('Lag', fontsize=12)
```

```
plt.ylabel('Cross--correlation', fontsize=12)
plt.legend(loc='best')
plt.grid()
ax.xaxis.set_major_locator(MaxNLocator(integer=True))
```

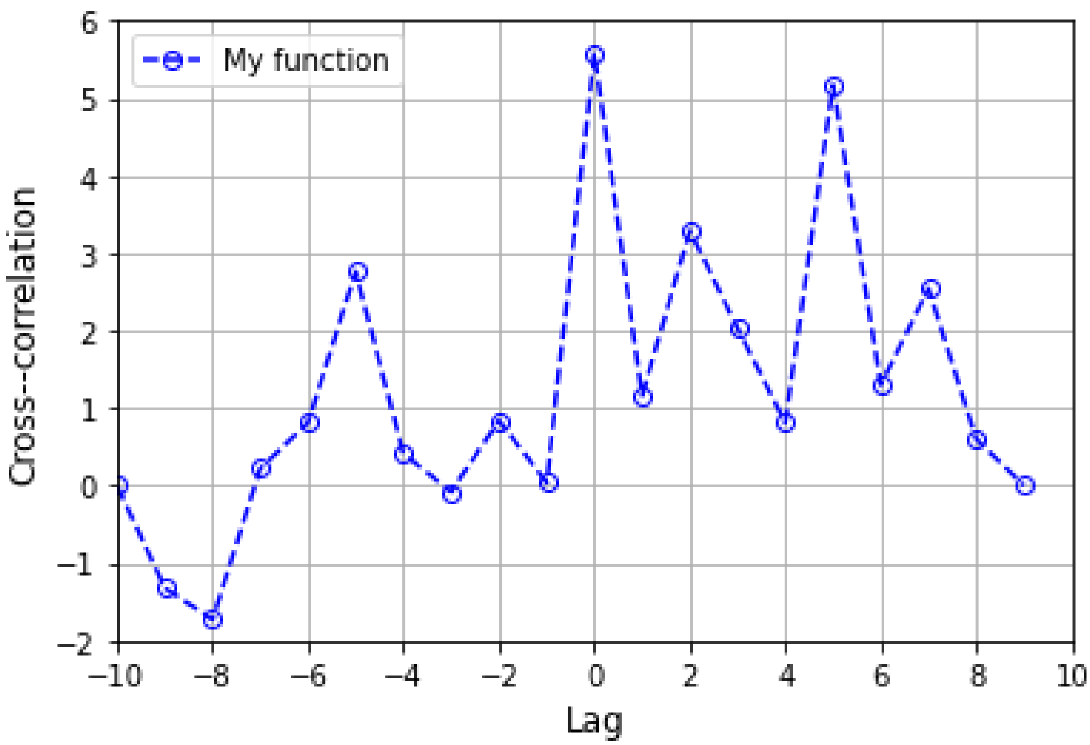

Fig. 5. Visualization with Python.

V. References

[1] https://www.ocean.washington.edu/courses/ess522/lectures/08_xcorr.pdf

[2] http://eceweb1.rutgers.edu/~gajic/solmanual/slides/chapter9_CORR.pdf