

A Python application to generate digital signals

Tatian-Cristian Mălin, Dorian Nedelcu*, Gilbert-Rainer Gillich

We introduce in this paper an application developed in the Python programming language that can be used to generate digital signals with known frequencies and amplitudes. These digital signals, since have known parameters, can be used to create benchmarks for test and numerical simulation.

Keywords: *Python application, signal generation, sinusoid, noise, damping, phase shift*

1. Introduction

The typical sensor for measuring the dynamic response of structures is the accelerometer. It generates a digital signal that is represented by a sequence of discrete values [1]-[3]. When interpreting earthquake response signals measured with accelerometers, estimation of velocities and displacements is often required [4].

To study the behavior of structures during earthquakes, we need digital signals that are presenting different earthquake movements. A list of databases that can be accessed on the Web that contain earthquake accelerograms in digital form are presented in [5], the most relevant being [6]-[8]. Due to modern techniques and methods of processing and converting analog data into digital data, the original recordings from bulletins and seismograms can be digitized and re-analyzed [9]. However, since most diagrams represent acceleration signals, we need to find the antiderivatives, namely the velocity and displacement. In prior research we developed an algorithm and implemented it in an application written in Python language, which calculates the antiderivatives. To test the accuracy of this application we need using digital signals with known parameters (frequency, amplitude, phase, damping coefficient, existence of noise). In this paper, we introduce the application that generates digital signals with known parameters and exemplify outcomes for different settings of the parameters.

2. The generated signal

The aim of this section is to introduce the theoretical aspects regarding the signals we can generate with an application we developed in Python. In the proposed application we can generate signals with up to three harmonic components S_i ($i=1\dots3$), which have the amplitudes a , b and c , the frequencies f_i and the phase Ph_i . Thus, the harmonic components of the signal can be written as follows:

- the first sinusoidal component is:

$$S_1 = a \cdot \sin(2 \cdot \pi \cdot f_1 \cdot t + \pi \cdot Ph_1) \quad (1)$$

- the second sinusoidal component is:

$$S_2 = b \cdot \sin(2 \cdot \pi \cdot f_2 \cdot t + \pi \cdot Ph_2) \quad (2)$$

- the third sinusoidal component is:

$$S_3 = c \cdot \sin(2 \cdot \pi \cdot f_3 \cdot t + \pi \cdot Ph_3) \quad (3)$$

In relations (1) to (4), t represents the time, i.e. the length of the signal.

We can also add both noise W and damping D to the signal. The damping is generated by involving the term:

$$D = e^{Damp \cdot t} \quad (4)$$

where $Damp$ is the damping coefficient. Note that, the damping coefficient can get associated positive values in the case of increasing the amplitude of the signal, or negative values in the case we intend to decrease the amplitudes.

The effect of noise can be expressed as:

$$W = p(x) \cdot Noise / \max[p(x)] \quad (5)$$

where $p(x)$ is the probability density for the Gaussian distribution and $Noise$ is a randomly generated value for each discrete time moment.

Finally, the most complex form of the signal is:

$$S = D \cdot (S_1 + S_2 + S_3 + W) \quad (6)$$

The signal is used to test application that derivate or integrate signals, for which the signal parameters should be known.

3. The *SignalGeneration* application

The *SignalGeneration* application was developed based on the Python programming language and is defined by four classes: "SignalGeneration", "Table_Grid", "Plot" and "PlotNotebook" and four public functions: "ExtragTextMemory", "ExtragImageMemory", "IsNumeric" and "Put_Clipboard". The main window represents a notebook control, which manage one chart window with the named tab: "Signal Generated". The "Plot" and "PlotNotebook" classes create the main window where the notebook with the chart windows will be created.

The public function "ExtragTextMemory" extract the Excel template from database to be saved as Excel file into "RESULTS" folder created by application. The template contains chart that will be updated at the end of the transfer. The public function "ExtragImageMemory", extracts the icons from database to memory to be used as icons when creating the toolbar.

The public function "Put_Clipboard" copies a string into Windows Clipboard and is called by "OnExcel" function from "SignalGeneration" class. When the results are exported to Excel file, these are memorized in strings copied to Windows Clipboard and pasted into Excel. The reason of this operation is a significative short time required for transfer comparing with export values cell by cell.

The "Table_Grid" show the numerical results on screen into a grid control.

The application uses a SQLite database to memorize the toolbar icons as image format and the "Excel template.xls" file, where the numerical and charts results will be exported. These files were loaded as Binaly Large Objects (BLOB) in "Config.db" application's file. Features of the toolbar are shown in figures 1 and 2.



Figure 1. The toolbar of the *SignalGeneration* application – input data

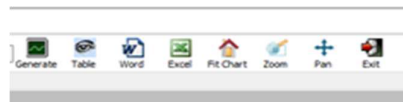











Figure 2. The toolbar of the *SignalGeneration* application – processing buttons

The application's toolbar is located at the top-left of the main window and includes text and button controls marked, with the functions described in table 1.

Table 1. Functions of the *SignalGeneration* toolbar

	Open – Load signal form CSV saved previous file.
N <input type="text" value="5000"/>	Text control to input 'Number of samples' N variable.
FR <input type="text" value="1000"/>	Text control to input 'Sampling frequency' FR variable.
Delta T <input type="text" value="0.001"/>	Text control to show 'Time interval' Delta T variable (Read Only – computed by application).
a / b / c <input type="text" value="1"/> <input type="text" value="0"/> <input type="text" value="0"/>	Text controls to input 'First/Second/ Third Amplitude' values of the A1/ A2/A3 variables.
f1 / f2 / f3 <input type="text" value="5"/> <input type="text" value="15"/> <input type="text" value="20"/>	Text controls to input 'First/Second/Third Target Frequency' values of the f1/f2/f3 variables.
Ph1/Ph2/Ph3 <input type="text" value="0"/> <input type="text" value="0"/> <input type="text" value="0"/>	Text controls to input 'First/Second/ Third Phase' coefficients of the Ph1/ Ph2/Ph3 variables.
x Pi Noise <input type="text" value="0"/>	Text control to input 'White Noise' Noise variable.
Damp <input type="text" value="0"/>	Text control to input 'Damping coefficient' Damp variable.
	Generate – Calculate the signal based on equation (6) and create the chart of the signal.
	Data table – Shows the table of the signal values, calling "Table_Grid " class.
	Word – Save application's graphical results to Word file.
	Excel – Save application's results (graphical and numerical) to Excel file.
	Fit Chart – Returns to initial view in the chart windows.
	Zoom - Enlarges selected area in chart windows.

	Pan - zoom in/out with the right mouse button pressed.
	Exit – Quit the application, calling "OnClose" function from "SignalGeneration" class.

The *SignalGeneration* application use the dependencies presented in table 2.

Table 2. Dependencies used by the *SignalGeneration* program

Python(x,y)	A free scientific and engineering development software for numerical computations, data analysis and data visualization based on Python programming language.	https://python-xy.github.io/
Matplotlib	A Python 2D plotting library which produces publication quality charts.	https://matplotlib.org/
wxPython	The cross-platform Graphical User Interface toolkit for the Python language.	https://wxpython.org/
SQLite	A C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.	https://www.sqlite.org/index.html
numpy	The fundamental package for scientific computing with Python.	https://numpy.org/

4. Examples of generated signals

The signals, which represent measured accelerations in mm/s^2 are generated with a number of samples $N=6000$ by a sampling frequency $FR=1000$ Hz. In table 3 are presented different settings of the parameters used to generate signals with the *SignalGeneration* application.

Table 3. Parameter settings for the generated signals

Figure	a	b	c	f ₁	f ₂	f ₃	Ph ₁	Ph ₂	Ph ₃	Damp	Noise
3	1	0	0	1	0	0	0	0	0	0	0
4	1	0	0	1	0	0	1	0	0	0	0.5
5	1	0	0	1	0	0	1	0	0	0.5	0

6	1	1	0	1	15	0	0	0	0	-0.5	0.5
7	1	1	1	1	5	10	0	0	0	0	0
8	1	1	1	1	5	10	0	0	0	-0.5	0
9	1	1	1	1	5	10	1	1	1	-0.5	0.5

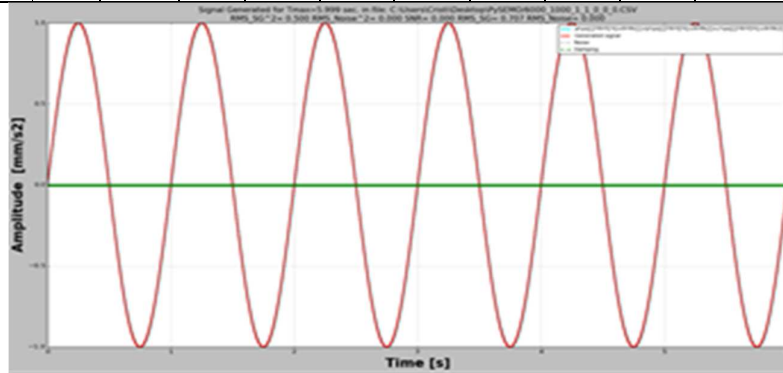


Figure 3. The signal with one harmonic component

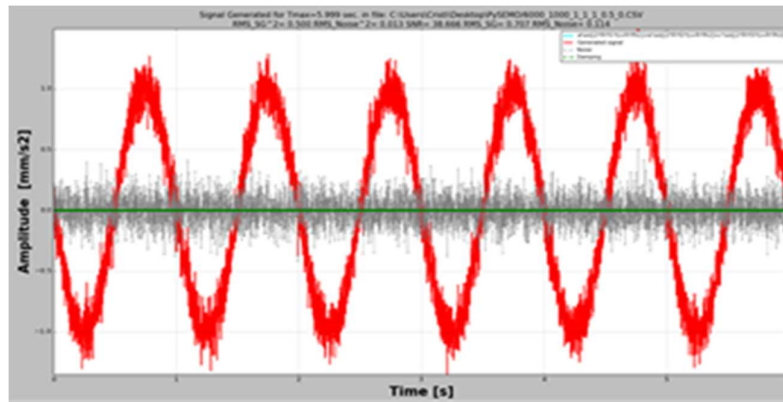
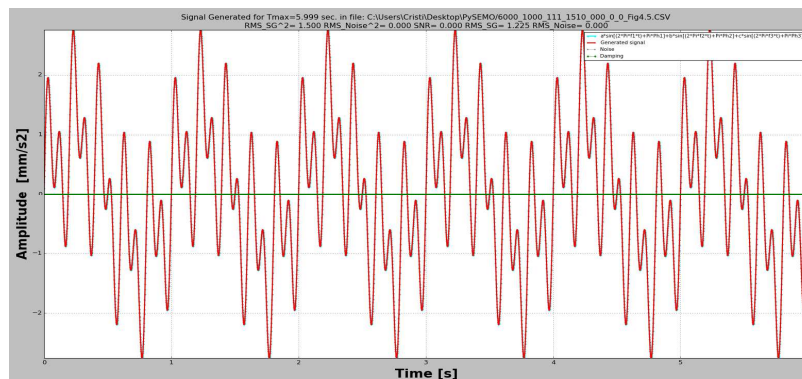
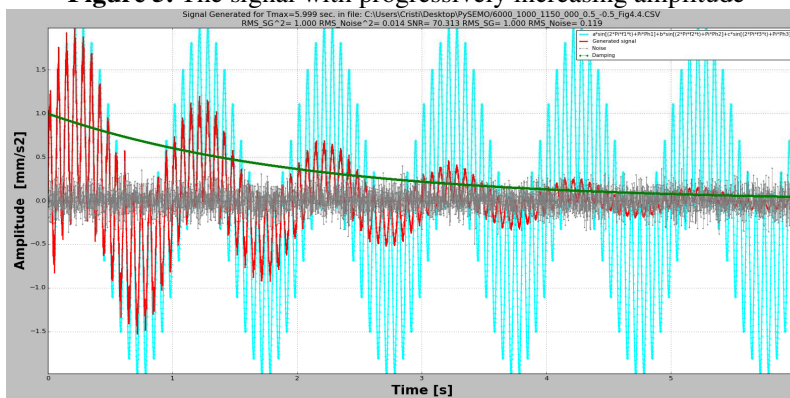
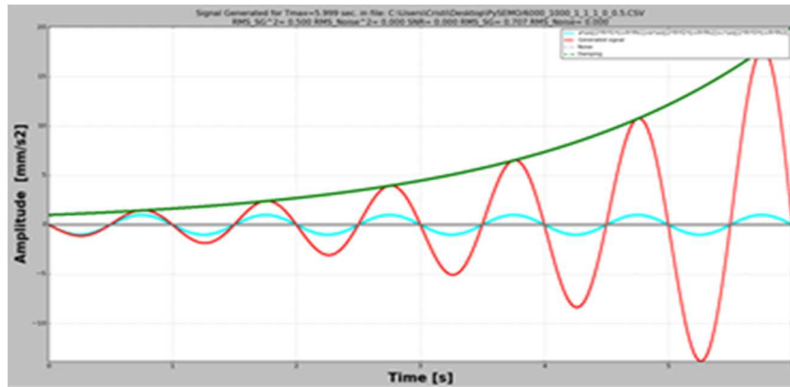


Figure 4. The signal with one harmonic component polluted with noise



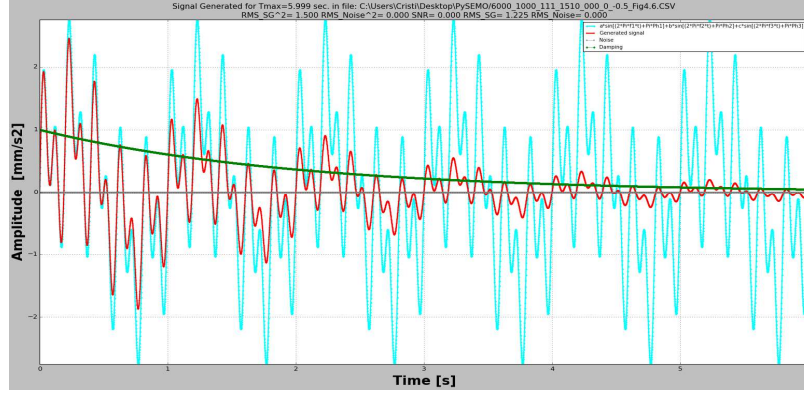


Figure 8. Damped signal with three components

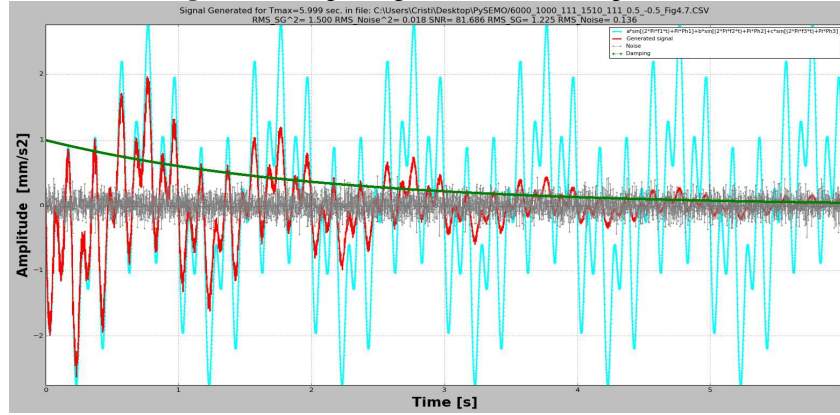


Figure 9. Damped signal with three components polluted with noise

In figures 3 to 9, we present the generated signals with the parameter settings from table 3. The different signals are represented in these figures with different colors (*green* – damping, *gray* – noise, *cyan* – signal with one to three components in the absence of damping and noise) and with *red* is represented the resulted signal.

5. Conclusion

In this paper, we present an application developed in the Python programming language that generates digital signals with known parameters (frequency, amplitude, phase, damping coefficient, existence of noise) and exemplify outcomes for different settings of the parameters. These digital signals, since have known parameters, can be used to create benchmarks for test and numerical simulation.

For our future research we need digital signals, with known parameters, to calculate the velocity and displacement from accelerograms and to use them as input for dynamic simulations, made for base-isolated structures.

References

- [1] Dueck R.K., *Digital Design with CPLD Applications and VHDL*, Cengage Learning, 2nd edition, 2011.
- [2] Proakis J.G., Manolakis D.G., *Digital Signal Processing*, Pearson Prentice Hall, 2007.
- [3] Grahame S., *Analogue and Digital Communication Techniques*, Newnes, 1999.
- [4] Nedelcu D., Malin T.C., Gillich G.R., Barbinta C.I., Iancu V., *Displacement and velocity estimation of the earthquake response signals measured with accelerometers*, The 9th International Conference on Advanced Concepts in Mechanical Engineering - ACME 2020, Jun. 4-5, 2020, Iasi, Romania.
- [5] Malin T.C., Gillich G.R., Nedelcu D., Iancu V., *Earthquake registrations database*, *Analele Universitatii „Eftimie Murgu“ Resita, Fascicula de Inginerie* 26(1), 2019.
- [6] <https://www.ngdc.noaa.gov/hazard/earthqk.shtml> last accessed 09.08.2020.
- [7] <https://earthquake.usgs.gov/earthquakes/search/> last accessed 09.08.2020.
- [8] <https://www.strongmotion.org> last accessed 09.08.2020.
- [9] Malin T.C., Gillich G.R., Nedelcu D., Iancu V., *Digitization of earthquake signals stored as images*, 43rd International Conference on Mechanics of Solids, 2019.

Addresses:

- PhD Student Tatian-Cristian Mălin, Babeş-Bolyai University, Faculty of Engineering, Piața Traian Vuia, nr. 1-4, 320085, Reșița, cristian.malin@student.uem.ro
- Prof. Dr. Eng. Dorian Nedelcu, Babeş-Bolyai University, Faculty of Engineering, Piața Traian Vuia, nr. 1-4, 320085, Reșița, d.nedelcu@uem.ro
(* corresponding author)
- Prof. Dr. Eng. Gilbert-Rainer Gillich, Babeş-Bolyai University, Faculty of Engineering, Piața Traian Vuia, nr. 1-4, 320085, Reșița, gr.gillich@uem.ro