2022-03

A Python-based Pipeline for Processing and Plotting Automatisation of Nanoindentation Data: Plotting with GraphPad Prism software

Đoćoš Miroslav, Brdar Sanja, Vejin Marija, Petrović Bojan, Milić Lazar, Kojić Sanja

21st International Symposium INFOTEH-JAHORINA, 16-18 March 2022

A Python-based Pipeline for Processing and Plotting Automatisation of Nanoindentation Data

Plotting with GraphPad Prism software

Miroslav Đoćoš
Department of Biomedical Engineering
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
djocos.miroslav@uns.ac.rs

Sanja Brdar
University of Novi Sad, Institute BioSense
Novi Sad, Serbia
sanja.brdar@biosense.rs

Marija Vejin
Department of Biomedical Engineering
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
vejinmarija@uns.ac.rs

Bojan Petrović
Department of Dental Medicine
University of Novi Sad, Faculty of Medicine
Novi Sad, Serbia
bojan.petrovic@mf.uns.ac.rs

Lazar Milić
Department of Biomedical Engineering
University of Novi Sad, Faculty of Medicine
Novi Sad, Serbia
lmilic@uns.ac.rs

Sanja Kojić
Department of Power, Electronics and Telecommunication
University of Novi Sad, Faculty of Technical Sciences
Novi Sad, Serbia
sanjakojic@uns.ac.rs

Abstract— **In this paper a Python based pipeline encompassing steps necessary for automatic processing and plotting nanoindentation data is presented. It enables to process and plot big amount of created raw data from Agilent Nanoindenter G200 and thus significantly increases knowledge discovery from such data. The described solution is demonstrated on example of 100 measurements on a polymer sample.**

Keywords-Python; nanoindentation; average; plotting; GraphPad Prism

I. Introduction

The Python programming language [1] has become very popular among researchers from various fields for data analysis [2,3]. Because the user operates with high-level abstractions that hide all low-level mechanisms such as memory management and hardware communications, it has a steep learning curve for beginners. As a result, one can focus on algorithm development. Several Python packages for interacting with and visualizing multi-dimensional data have recently been released. The Python programming language [1] has become very popular among researchers from various fields for data analysis [2,3]. Because the user operates with high-level abstractions that hide all low-level mechanisms and hardware communications, it has a steep learning curve for beginners.

By far the most popular method for determining the mechanical properties of 2D materials is indentation testing. This method is divided into two categories: (1) AFM indentation of free-standing (FS) 2D materials, also known as free-standing indentation (FSI) testing, and (2) instrumented nanoindentation of 2D materials on a substrate. The load-displacement relationship of a 2D material, whether FS or on a substrate, is measured in these tests, and the material's mechanical properties are extracted by fitting the measured load-displacement curve to an analytical model. All existing indentation models are based on the assumption of an isotropic linear elastic solid, and 2D materials subject to indentation are typically regarded as linear and isotropic [4].

The development of a large family of precision force nano testing methods known as "nanoindentation" has influenced progress in mechanical behaviour analysis of various materials at sub-micro and nanoscale levels to a large extent. The application of small forces to the test object and precise measurements of deformation with nanometre resolution under uniaxial compression, tension, bending, shear, and torsion by various methods with simultaneous in situ registration of their microstructural parameters is referred to as force nanomechanical testing. The main research directions developed in the last decade to improve experimental approaches and the results are obtained in amorphous solids and biomaterials using these approaches on single-, micro-, and nanocrystalline materials, composites, films, and coatings [5].

The Agilent Nanoindenter G200 is a machine that performs a large number of individual measurements since it is usually necessary to perform a sufficient number of tests so that the obtained data can be statistically analysed with an adequate power sample size and therefore generates a large amount of output data. Usually, about 100 measurement locations are made per sample, where each location is measured in over 300 points, depending of the nanoindentation protocol. Agilent has its own data processing software – Agilent Analyst but it is old and confusing and does not work on newer versions of Windows. Existing software cannot represent an averaged curve that symbolises the mean values of a certain number of curves. It is not uncommon that for one research it is necessary to perform more than 100 000 measuring points [6,7].

The nanoindentation measurement method for obtaining mechanical material characteristics is important not only for mechanical engineers but also for many other disciplines such as biomedical engineers, material scientists and archaeologists [8-10].

This paper presents a new software solution which can process and plot the data obtained from Nanoindenter. The program can calculate the mean representative curve of the standard nanoindentation curves (load vs displacement into surface curves), which the old Agilent program could not. Moreover, there is an option to choose measurements which should be excluded and that is possible in two ways – manually and automatically.

Everything previously mentioned was important from the aspect of data processing. An equally important component of this program, in addition to data processing, is the possibility of automated data plotting in the desired software, in this case the GraphPad Prism software.

II. The Python Pipeline

The Agilent Nanoindenter is the instrument which generates excel output files with numerous sheets. The first sheet contains Youngs modulus and hardness (calculated from the Load on sample vs Displacement into surface curves [8]), their mean values and standard deviation. Then there are as many following sheets as there are measurement locations, each sheet with the data for one curve - Load on sample vs Displacement into surface. Sometimes there is a need to merge various excel files into one plot.

A. *Main algorithm*

The algorithm in Fig. 1. depicts structure, flow and arbiter of the programme and it has the following sections:

- Choosing files,
- Visual checking of the data,
- Automatic or manual selection of measurement curves,
- Calculation of the average curve,
- Forming of excel output file that corelates with the plotted data and
- Forming the GraphPad Prizm software file with plotted data.

The first step of the algorithm is to select the one or more .xls raw files from the nanoindentation and import them to the Python list. Next, there are two paths, one dedicated to calculate average curve and preview of all measurement curves plus average curve, then to form the excel file of all curves and to create GraphPad Prism file with plotted data. Other is designed to arbiter the choice of measured curves. Further, there are two ways, one to manually select the measurement locations that should be excluded and other to choose the factor k. When users are satisfied with the selection of the location algorithm connects to the section of forming excel and plotting data.

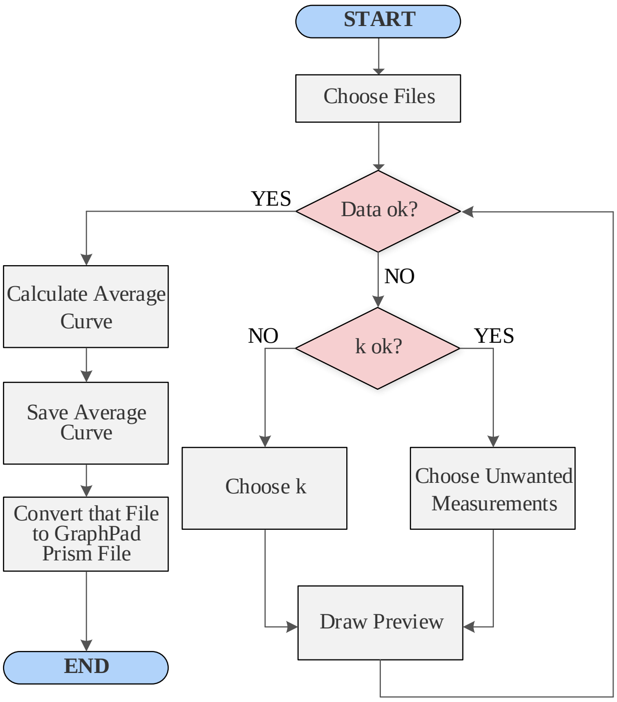


Figure 1. Main algorithm diagram

B. *Data Transformations*

To clarify the data transformation and transitions Fig. 2. is formed. The blue sections represent the excel data files and GraphPad Prism file while grey sections are from Python sections.

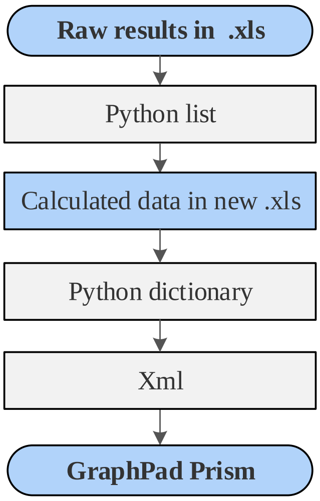
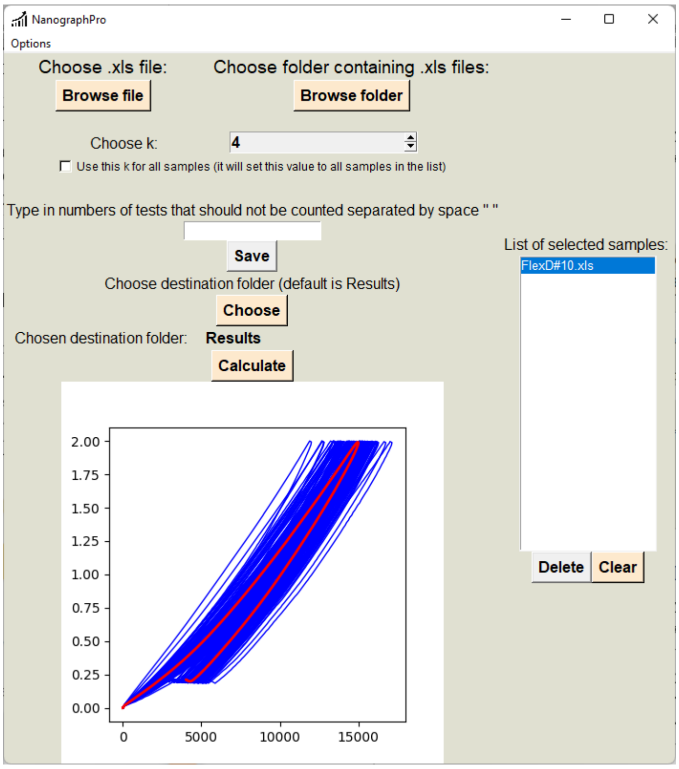


Figure 2. Data transition and transformation

III. Graphic User Interface

The user interface for running the processing pipeline is implemented in Python using *tkinter*. The GUI (Graphic User Interface) window view of the programme is depicted in Fig. 3.



NanographPro

Options

Choose .xls file: Browse file

Choose folder containing .xls files: Browse folder

Choose k: 4

☐ Use this k for all samples (it will set this value to all samples in the list)

Type in numbers of tests that should not be counted separated by space " "

Save

Choose destination folder (default is Results)

Choose

Chosen destination folder: Results

Calculate

List of selected samples:

FlexD#10.xls

Delete Clear

Figure 3. Data transition and transformation

After application is turned on, it requires choosing *.xls* files containing raw data from nanoindenter to process. The data from *.xls* file are collected as object of *ExcelFile* class found in the *pandas* module. The algorithm will calculate an average curve and plot all measurement curves (the blue lines) with the average curve (the red line) on one graph. Thus, user can see results preview. Failed tests marked with "****" in the raw .xls data file are automatically excluded from processing. To calculate the average curve, firstly two empty lists are created. One for the x-axis (average Displacement into surface, *AvD*):

$$AvD_i = \frac{1}{N}\sum_{j=0}^{N-1} D_{ij} \quad i = 0, 1, \dots, P-1 \quad , \qquad (1)$$

and the second for y- axis (average Load on sample, *AvL*):

$$AvL_i = \frac{1}{N}\sum_{j=0}^{N-1} L_{ij} \quad i = 0, 1, \dots, P-1 \quad . \qquad (2)$$

In the previous formulas, P stands for the length of the shortest measurement, and N is the number of measurement locations. The number of elements in the new list is actually the same as the number of the elements in the shortest measurement L. The algorithm is cutting and ignoring data that are on index P and above. The cut-off curves are usually the curves which gathered the data from the material defects and they do not represent material behaviour. If the preview graph is adequate, the user can proceed with converting data to the *.pzfx* files (GraphPad Prism file extension). If not, there are more settings to improve the result. The measurements of one sample are considered to be distributed as in Gaussian distribution. To exclude outliers from the calculation of the average curve, every measurement that has a peak out of range (Average peak displacement) ± k · σ should be excluded. Where σ is a standard deviation of all peaks and k is a parameter that can be determined by the user. For every change of the parameter k, the algorithm is plotting preview graph again, so the user can choose the best k for his/hers needs. When there are some measurements with obviously deformed curve, the user can exclude them by typing their location numbers in the entry.

When all configuration has been done and the user has adequate graph, the converting process can start. The algorithm first makes new *.xls* containing two sheets. In the first sheet the calculated average curve (*AvD* and *AvL*) is placed as two columns. The second sheet contains two columns for every measurement that is not excluded by user, one for displacement into surface and second for load on sample. The data from the new *.xls* files are now loaded with function *read_excel* from *Pandas* module. For easier access, the data are placed in Python dictionary as follows:

$$data = \{\text{"Average"}: [AvD, AvL], \text{"Samples"}: [[disp1, los1], [disp2, los2], \dots]\}$$

To create the new *.pzfx* file, the algorithm is using pre-created template in GraphPad Prism that is parsed as *xml* using Python module *BeautifulSoup*.

Now algorithm replaces existing d values from *.pzfx* template with values from data dictionary. At the end, when all data has been replaced, the algorithm is saving modified *.pzfx* with the new name. The result is two new files, first *.xls* with calculated average curve and not excluded measurements, and second *.pzfx* file that has all measurements written in separate table and one table for the calculated average curve. All measurements are plotted on one graph with lines in shades of grey and the average curve is plotted with the red line. The Fig. 4 illustrates one example of plotted nanoindentation measurements performed on 100 locations on FlexDym polymer with k = 4.

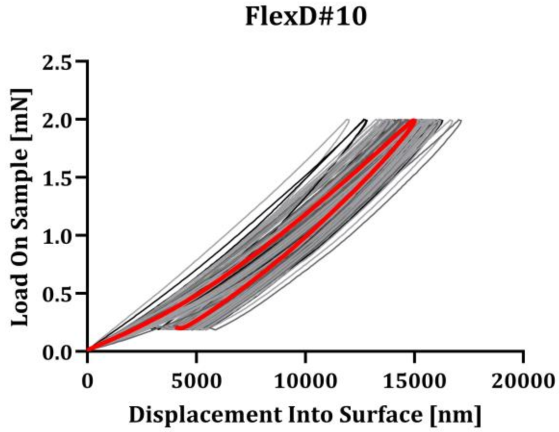


Figure 4. An illustration of plotted nanoindentation measurements

IV. Conclusions

We present a Python programme for optimised processing Nanoindetation raw results in this paper. The NanoSuite lacks additional built-in Excel data processing tools that would allow users to tailor data acquisition to their specific needs. Due to a lack of this type of interface, unwanted and time-consuming processing occurs. The presented programme allows the user to run a programme and get results quickly, precisely, and efficiently with a possibility to perform all needed adjustments.

An additional advantage of this program is reflected in the fact that it can deliver an averaged curve of all desired measurements. This curve is of particular importance when testing materials that are not fully prepared according to the nanoindentation protocol, meaning they are not polished or they are heterogeneous. In these cases, this curve descriptively determines the properties of the tested materials, rather than giving the absolute values of the tested parameters. Moreover, when these averaged curves are compared with each other, it is easier to compare the behaviour of similar materials tested by nanoindentation.

The time required to execute multiple Excel files can be incomparably reduced when automated processing is used instead of manual processing, according to the findings. The same is true when dealing with a single file containing up to 100 tests. Similar methods that export raw results in the form of an Excel file can benefit from this type of optimized processing.

Acknowledgment

This project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 951747.

References

[1] G. Van Rossum, and F.L. Drake, Python Reference Manual, 2006. http://docs.python.org/ref/ref.html

[2] E. Gouillart, J. Nunez-Iglesias, and S. van der Walt, ‘Analyzing microtomography data with Python and the scikit-image library’, *Adv Struct Chem Imaging*, vol. 2, no. 1, p. 18, 2017, doi: 10.1186/s40679-016-0031-0.

[3] R. Shkarin *et al.*, ‘Quanfima: An open source Python package for automated fiber analysis of biomaterials’, *PLOS ONE*, vol. 14, no. 4, p. e0215137, Apr. 2019, doi: 10.1371/journal.pone.0215137.

[4] G. Cao and H. Gao, ‘Mechanical properties characterization of two-dimensional materials via nanoindentation experiments’, *Progress in Materials Science*, vol. 103, pp. 558–595, Jun. 2019, doi: 10.1016/j.pmatsci.2019.03.002.

[5] Yu. I. Golovin, ‘Nanoindentation and Mechanical Properties of Materials at Submicro- and Nanoscale Levels: Recent Results and Achievements’, *Phys. Solid State*, vol. 63, no. 1, pp. 1–41, Jan. 2021, doi: 10.1134/S1063783421010108.

[6] S. Kojic, ‘Design, testing and fabrication of microfluidic devices for biomedical application’, PhD thesis, 2021 available online: http://www.ftn.uns.ac.rs/310901761/doktorska-disertacija.

[7] G. Stojanović, M. Pojić, S. Kojić, A. Mišan, and D. Vasiljević, ‘Mechanical properties of edible biofilm as a substrate for printed electronics’, *Appl. Phys. A*, vol. 125, no. 8, p. 576, Aug. 2019, doi: 10.1007/s00339-019-2881-5.

[8] B. Petrović *et al.*, ‘Surface characterization of the raw and cooked bovine cortical metatarsal bone’, *Acta of Bioengineering and Biomechanics*, vol. 21, no. 1, pp. 13–21, 2019, doi: 10.5277/ABB-01197-2018-02.

[9] B. Petrovic *et al.*, ‘Characterization of glass ionomer cements stored in various solutions’, *Materiali in tehnologije*, vol. 53, pp. 285–293, Apr. 2019, doi: 10.17222/mit.2018.159.

[10] M. R. Radovanović, S. P. Kojić, D. Z. Vasiljević, and G. M. Stojanović, ‘Characterization of LC sensor structures realized by PCB and LTCC technology for determining moisture in building materials’, *Processing and Application of Ceramics*, vol. 12, no. 1, pp. 13–20, 2018, doi: 10.2298/PAC1801013R.

[11] J. Lazarević, M. Drljača, S. Brdar, B. Petrovic, S. Kojic, and G. Stojanovic, ‘Automating of Nanoindentation raw results processing with Python module’, presented at the IcETRAN 2018, Palić, Serbia, Jun. 2018.