UDC 57.087

PROCESSING OF AIR FLOW MEASUREMENT RESULTS USING THE PYTHON SCIPY LIBRARY

Yukhymenko Y. A., Stelmakh N. V.
National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, Ukraine
E-mail: yuhim01@gmail.com, stelmakhnv@ukr.net

Annotation. After studying the air flow in the ventilation channel of the ventilator (artificial lung ventilation) using the Matlab system, files with the extension.mat were created with the results of measuring the air flow according to various indicators, such as the deviation of the indicators from the set values, the integral of the air flow, acoustic noise, and so on further. Also, .m files were created, which build graphs based on these data for further analysis.

Introdution

Air flow control in ventilators is a very important stage in their design, since these devices belong to the field of critical medicine and their breakdown or errors during operation are unacceptable [1]. That is why a detailed study of the accuracy of the manufactured devices is carried out at the stages of equipment design and testing [2]. After studying the flow of air in ventilator units using the Matlab numerical analysis application program package and saving large data sets with the results of the flow study, there is a need for visualization processing of these data. During the study, the air flow was gradually increased from 0 l/min to 100 l/min, and the readings of ventilator air sensors were also recorded. The flow was supplied at a known interval with a step of 2 l/min per second. Also, when the air flow increased, the amount of the flow itself and the acoustic noise were measured. Also, the Matlab system was used to calculate the integral of the absolute error of the flow measurement, the integral of the power of the flow noise, the maximum absolute error (in total 44 .mat files with data).

The Matlab numerical analysis software package is a convenient tool and has many instruments for data analysis, which is why it is usually used to calculate intermediate or final results of scientific research. But its huge disadvantage was and remains the fact that the programs developed on it are not standalone. This means that in order to open the developed .m file and plot graphs based on already measured values, you need to have Matlab installed in the system. This in turn can be problematic, since Matlab has relatively high system requirements (minimum 4 gigabytes of RAM, as well as about 30-50 gigabytes of free memory space, depending on the required additional modules, which makes these programs simply not suitable for use in weak systems). In addition, the Matlab system is paid, which also limits the possibilities of its application. Therefore, the search and application of other software tools and tools for solving the tasks of processing large arrays of information is relevant.

Materials and methods

The closest alternative to Matlab in the field of data processing is the Python programming language. It is free, compatible with the most popular systems (Windows 7, 10 and 11, Mac OS, Linux). Also, there are many libraries (or modules) for data processing in Python. The Pandas module is used for forming matrices and convenient work with them, the NumPy module has convenient methods for working with numerical arrays, the Keras module is used for training artificial intelligence from the received data, and so on. All Python modules are open-source, meaning they are free and open to modification and distribution.

The Python language itself can be installed on any system easily and quickly, and does not take up much disk space. Modules are installed by Python's built-in pip package manager. The scipy module, which was experimentally used for reading .mat files, is installed by a simple command "pip install scipy" [3].

Also, programs (or scripts) written in Python can be run anywhere by Python itself (if it is installed), or can be compiled for any system to run independently. Along with open-source, this removes any restrictions that were present when using Matlab [4].

As described above, the scipy library and its loadmat method, was used to experimentally open .mat files with air flow calculations. This method accepts a path to a .mat file, reads it, and returns its contents. For example, to read the measurement results of the first experiment of the first muffler model, we will use the commands:

```
mat = scipy.io.loadmat('FlowSensor_data/FlowSensor_002_00_M_1.mat')
print(mat)
```

print will display the contents of the file FlowSensor_002_00_M_1.mat on the screen. In the terminal of the Visual Studio Code editor, which was used for writing the code, the contents of the file will be received, which is presented in Fig. 1.

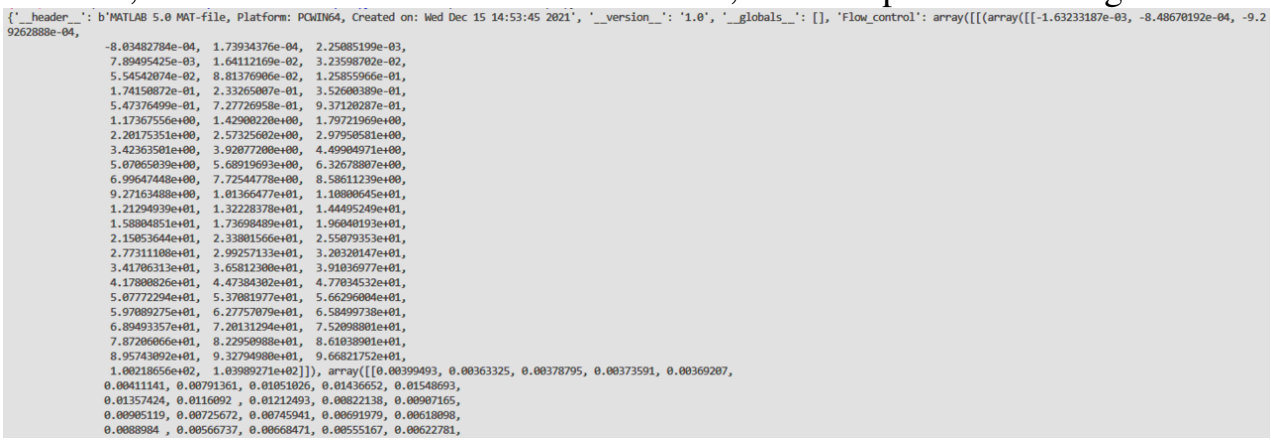

Fig.1 Read array of data from .mat file

As can be seen in fig. 1 - the data is presented in a "raw" format, therefore it is necessary to process it. But also analyzing the result shows that the data in the .mat file is stored in the "key-value" format. And Python read them as a dictionary data

type, that is, a dictionary [5]. It has the same key-value format. To check this, the data value by key can be received. As can be seen in fig. 1 there is one of the keys 'Flow_control', i.e. control values of air flow. By using the key as follows:

```
mat = scipy.io.loadmat('FlowSensor_data/FlowSensor_002_00_M_1.mat')
print(mat['Flow_control'])
```

An array of values specifically for 'Flow_control' will be acquired. The result from the Visual Studio Code terminal is presented in Fig. 2.

```
[[(array([[-1.63233187e-03, -8.48670192e-04, -9.29262888e-04,
          -8.03482784e-04,  1.73934376e-04,  2.25085199e-03,
           7.89495425e-03,  1.64112169e-02,  3.23598702e-02,
           5.54542074e-02,  8.81376906e-02,  1.25855966e-01,
           1.74150872e-01,  2.33265007e-01,  3.52600389e-01,
           5.47376499e-01,  7.27726958e-01,  9.37120287e-01,
           1.17367556e+00,  1.42900220e+00,  1.79721969e+00,
           2.20175351e+00,  2.57325602e+00,  2.97950581e+00,
           3.42363501e+00,  3.92077200e+00,  4.49904971e+00,
           5.07065039e+00,  5.68919693e+00,  6.32678807e+00,
           6.99647448e+00,  7.72544778e+00,  8.58611239e+00,
           9.27163488e+00,  1.01366477e+01,  1.10800645e+01,
           1.21294939e+01,  1.32228378e+01,  1.44495249e+01,
           1.58804851e+01,  1.73698489e+01,  1.96040193e+01,
           2.15053644e+01,  2.33801566e+01,  2.55079353e+01,
           2.77311108e+01,  2.99257133e+01,  3.20320147e+01,
           3.41706313e+01,  3.65812300e+01,  3.91036977e+01,
           4.17800826e+01,  4.47384302e+01,  4.77034532e+01,
           5.07772294e+01,  5.37081977e+01,  5.66296004e+01,
           5.97089275e+01,  6.27757079e+01,  6.58499738e+01,
           6.89493357e+01,  7.20131294e+01,  7.52098801e+01,
           7.87206066e+01,  8.22950988e+01,  8.61038901e+01,
           8.95743092e+01,  9.32794980e+01,  9.66821752e+01,
           1.00218656e+02,  1.03989271e+02]]), array([[0.00399493, 0.00363325, 0.00378795, 0.00373591, 0.00369207,
         0.00411141, 0.00791361, 0.01051026, 0.01436652, 0.01548693,
         0.01357424, 0.0116092 , 0.01212493, 0.00822138, 0.00907165,
         0.00905119, 0.00725672, 0.00745941, 0.00691979, 0.00618098,
         0.0088984 , 0.00566737, 0.00668471, 0.00555167, 0.00622781,
         0.00859522, 0.00793358, 0.00931153, 0.01244971, 0.01517052,
         0.02176529, 0.02212638, 0.03008469, 0.03295992, 0.04475616,
         0.05782879, 0.06871501, 0.07599125, 0.09381443, 0.11201928,
         0.10596933, 0.12838428, 0.11020445, 0.13813307, 0.12927348,
         0.16575108, 0.15762093, 0.18535756, 0.17024072, 0.24306386,
         0.22072285, 0.28546528, 0.32636352, 0.34316474, 0.36414709,
         0.36406743, 0.37861597, 0.39610824, 0.42363956, 0.4249108 ,
         0.46924757, 0.48312998, 0.48739994, 0.52226522, 0.55892692,
```

Fig. 2 Data array obtained by key 'Flow_control'

That is, it is clear that data can be obtained by key for any parameter that is necessary [6].

Conclusions

As a result of the material proposed and considered in the work, it can be concluded that replacing the Matlab application program package for numerical analysis with the high-level Python programming language and its additional libraries as a tool for processing the results of air flow research in ventilators is possible without conducting new measurements, but using the results from .mat files. This gives a more flexible to use tool for processing and visualizing data, but at the same time it is just as functional. In the future, it is possible to use artificial intelligence via Keras module to process the measured data, since the volumes of data arrays and the number of experiments are very large (in total 44 .mat files with data).

Keywords: Python, Matlab, SciPy, Data Science.

References

[1] V. V. Boyko, V. M. Lisovyi, *Urgent Surgery: A Textbook for 6th-year students of medical faculties of medical universities*, Ministry of Health of Ukraine, Kharkiv National Medical University, NTMT, 514 p 2019. (in Ukrainian).

[2] E. Diaz, A. H. Rodriguez, J. Rello, "Ventilator-associated pneumonia: issues related to the artificial airway", *Respiratory Care*, vol. 50, no. 7, pp. 900-909, 2005.

[3] R. Vallat, "Pingouin: statistics in Python", *J. Open Source Softw*., vol. 3, no. 31, p. 1026, 2018.

[4] H. R. Paur et al., "In-vitro cell exposure studies for the assessment of nanoparticle toxicity in the lung – A dialog between aerosol science and biology," *J. Aerosol Sci*., vol. 42, no. 10, pp. 668-692, 2011.

[5] Y. A. Yukhymenko, N. V. Stelmakh. "Simulation of air flow in vent channels of medical equipment", in *XVII All-Ukrainian Scientific-Practical Conference of Students, Postgraduates, and Young Scientists "Efficiency and Automation of Engineering Solutions in Instrumentation",* December 07-08, 2021, pp. 330-333.

[6] I. V. Mastenko, N. V. Stelmakh, "Generative design of a frame type construction", *KPI Science News,* 2, pp. 81–89, 2021. DOI: 10.20535/kpisn.2021.2.236954

УДК 612.171.1+ 004.852

ПРО СТРУКТУРИЗАЦІЮ БАЗ МЕДИКО-БІОЛОГІЧНИХ ДАНИХ

Шуляк О. П.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна
E-mail: shulyak.alex.47@gmail.com , amnevec-ee22@lll.kpi.ua

Структуризація баз даних у роботі розглядається як один із перших кроків на шляху здійснення глибинного системного аналізу інформації, накопиченої у них, яка є узагальненням практичного досвіду роботи з пацієнтами з тієї чи іншої конкретної проблематики з їх станом здоров'я.

Перехід до структурованих баз даних пов'язаний з бажанням і зрозумілою потребою зберігати у аналізі даних пацієнтів у нерозривному вигляді ті конкретні реальні сполучення значень параметрів і показників, які мали місце у практиці роботи з пацієнтами, і разом розкривають зміст кожного їх клінічного випадку. Структурування баз даних здійснюється без втрати співвідношень та зв'язків у значеннях показників у складі симптомокомплексів пацієнтів.

Для уточнення умов вирішення такого питання визначено структурні особливості записів дев'яти баз даних, що відрізняються різною спеціалізацією щодо захворювань пацієнтів, різним цільовим призначенням таких баз даних, різним набором типів клінічних випадків, кількістю та конкретним змістом параметрів та характеристик здоров'я організму людини.

Характерні особливості восьми з них наведені у таблиці 1.

У процесі структурування всі записи бази даних, обраної для аналізу, розподіляються по категоріальним групам. Критерієм відбору записів у кожну таку групу є обраний дослідником конкретний набір значень параметрів, які