THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
N°. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

# AUTOMATED PROCESSING OF MECHANICAL TEST DATA FOR VARIOUS COMPOSITE MATERIALS USING MATLAB

**Vasile BRIA, Marius BODOR***
"Dunarea de Jos" University of Galati, Romania
e-mail: marius.bodor@ugal.ro

## ABSTRACT

*The behaviour of composite materials during the mechanical testing process might exhibit, in some situations, very different patterns compared to those of conventional materials. This is why the eventual automation of data processing might require additional steps towards obtaining realistic results from mechanical testing. The present work addresses this issue by proposing an algorithm written using MATLAB software and applying it in processing data from mechanical testing of selected composite materials with various compositions and behaviours.*

KEYWORDS: MATLAB; data collection; mechanical characterizations

## 1. Introduction

The new materials industry, like many other fields, is constantly changing, and this leads to the design of new materials with enhanced mechanical properties. The most relevant examples are composite materials, regardless of their type. By definition, a composite material is an assembly consisting of at least two components of different natures and behaviours, provided that they do not interact chemically, so that the material formed has superior characteristics [1].

Recently, such materials have been used in almost all fields, from medicine, sports, to the aerospace field. The use of a composite material requires detailed knowledge of the mechanical behavior.

The mechanical characterization of a material can be done according to a set of standard tests. For the characterization of a material, two types of tests are performed: static and dynamic. Dynamic tests are aimed at describing fatigue behaviour, and static tests are aimed at describing the mechanical behaviour of materials under tension, compression and bending [2].

The branch of science that deals with the characterization of materials, of any kind, is mechanics of materials. For each of the three static tests, the description of a material requires the analysis of the stress/strain characteristic curve, and following the analysis of the characteristic curve, different parameters can be calculated that describe the behaviour of the tested material [3].

In general, test equipment only measures force and displacement, and machine software can generate both graphs and specific parameter values [2]. Information technology plays an important role in the process of reading, analysing and characterizing materials, because it can do this automatically, in a very short time, avoiding the propagation of errors. However, after obtaining the data, in most cases an evaluation of the results is needed.

Automating the evaluation of the obtained data leads to more accurate results and minimizes errors. An optimal environment for such data processing can be the MATLAB language.

MATLAB (from Matrix Laboratory) is a development environment for numerical computation and statistical analysis, it allows manipulation of matrices, visualization of functions, implementation of algorithms, creation of interfaces and can interact with other applications [4].

The focus of the present work is on mechanical static tests performed on composite materials, more precisely on data collection, their processing and finally, obtaining correct data for material characterization. The processing and final data acquisition were performed using an algorithm especially created for this purpose in MATLAB and its functionality is explained in detail in this article.

## 2. Materials and methods

For material testing, the Instron 8852 mechanical testing machine was used, equipped with special modules for each test. The Instron 8850 Series System is a servo-hydraulic, biaxial dynamic testing

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
N°. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

system that provides axial and torsional loads on the specimen. Featuring a precision-aligned, high-rigidity two-column frame, the 8850 series meets the challenging demands of a diverse range of static and dynamic biaxial testing requirements. The Instron 8852 system has an axial load capacity of ±100 kN and a torque capacity of ±1000 Nm with an actuator axial travel of 150 mm and a rotational travel of 90° [5].

The pressure required to operate the mechanical testing machine is provided by the Instron 3520 series hydraulic pump, with a noise level of between 58-63 dB. The pump provides a flow rate ranging from 13 litres/min to 249 litres/min and a nominal operating pressure between 207 bar and 280 bar. The pump is equipped with a programmable logic controller (PLC), which provides system control, monitors oil level and temperature, filter status and engine temperature. Cooling is achieved by using an air-cooling unit, which transfers heat directly to the ambient air [6].

Console software is the main user interface for the 8800MT. Running on a PC, it allows all controller functions to be viewed and configured including control-loop optimization, setting of operational limits, and running of simple cyclic tests. Console provides the foundation for running more demanding tests in application software such as WaveMatrix and Bluehill 3 [7].

For complex data processing, in order to obtain results that reflect as faithfully as possible, the behavior of the tested materials, the Matlab programming language, developed by MathWorks, was used. This tool, through its libraries, allows advanced numerical processing, the implementation of complex algorithms, graphical representation of functions and data, making it the right environment for effective processing and analysis of the data obtained when testing the mechanical characteristics of materials. The MATLAB version used is: 8.4.0.150421 (R2014b). For tensile, compression and bending testing of composite materials, the test methods given by ASTM standards, corresponding to each type of test, were used [8-10].

The software dedicated to the primary processing of the data acquired from the mechanical testing machine, which is used to calculate the characteristic parameters of each test, identifies the critical points on the stress/strain curve, with the conventional tensile test characteristic curve as a benchmark (Fig. 1).

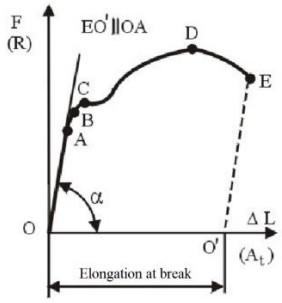


***Fig. 1.*** *The conventional tensile test characteristic curve: OA – quasi-linear zone; OB – elastic zone; C – the horizontal portion starting from point C, called the yield point; D – maximum force; E – breaking point of the specimen*

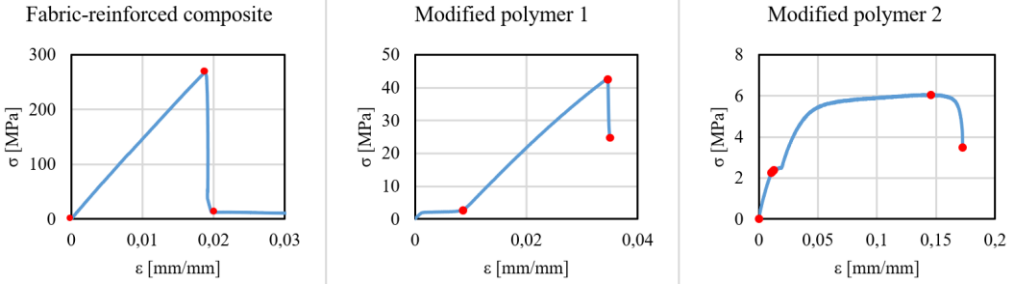


***Fig. 2.*** *Examples of different shapes of curves obtained on the same mechanical testing machine for composite materials, depending on the material class*

Starting from this shape of the conventional characteristic curve, it is easy to identify the technical elasticity limit (segment OA), the area in which the modulus of elasticity is calculated. The technical proportionality limit is the point on the curve corresponding to the maximum normal stress at which the slope of the tangent to the characteristic curve differs by 10% from the slope of the tangent at the origin [11]. In the case of composite materials, depending on the material class, the curves obtained on the same mechanical testing machine may have different or even completely different shapes, as presented in the examples in Figure 2.

In Figure 2, the curve on the left was obtained from a fabric-reinforced composite material, the preparation method being presented in Table 1. The

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
N°. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

curve in the middle represents the result of the tensile test on Epiphen E 4020 epoxy resin modified with aspartic acid (0.1 mol phr), which was maintained for half an hour after moulding under the influence of electromagnetic radiation (blue-UV). The curve on the right represents the result of the tensile test on SG1452 epoxy resin, modified with (poly)methylene methacrylate (PMMA), also maintained for half an hour after moulding under the influence of electromagnetic radiation (blue-UV).

The shapes of the curves differ, both from the shape of the conventional characteristic curve and from each other. While for the first and last curves, the area of proportionality between force and elongation (OA) is exactly the first part of the curve, for the second curve this area is preceded by a portion of the curve that represents the repositioning of the jaws of the mechanical testing machine. Starting from the figure above and from the way the technical proportionality limit is identified, in the case of the curve (in the middle) the Young's modulus will be calculated based on the initial area of the curve, so an incorrect result will be obtained.

***Table 1.*** *Materials and architecture used to obtain the fabric reinforced composite material*

| Layer | Fabric type | Specific density [g/m$^2$] | Orientation [°] |
|---|---|---|---|
| 1-4 | Mixt (aramid carbon) | 188 | 0/90/0/90 |
| 5-8 | Mixt (aramid carbon) | 68 | 45/-45/45/-45 |
| 9-10 | Carbon | 240 | 0/0 |
| 11-12 | Glass | 108 | 90/90 |
| 13-14 | Glass | 163 | 45/-45 |
| 15-18 | Glass | 280 | 0/0/0/0 |
| 19-20 | Glass | 163 | 45/-45 |
| 21-22 | Glass | 108 | 90/90 |
| 23-28 | Carbon | 160 | 0/0/45/-45/45/-45 |
| 29-30 | Carbon | 240 | 0/0 |

## 3. Results and discussions

Previous experience has shown that results from mechanical tests of composite materials can produce different curve shapes; therefore, the primary processing performed by the software of the mechanical testing machine (Bluehill 3) is not sufficient to obtain realistic results. Therefore, a code was written in the MATLAB programming language, which allows the processing of data obtained from the Instron 8852 universal mechanical testing machine.

The data provided by the mechanical testing machine are obtained in comma-separated values (CSV) format, one file for each tested specimen, similar to the example in Figure 3. According to the method defined in the Bluehill 3 application, in this file, by columns, the data are saved in columns as follows: Time (s), Elongation (mm), Load (N), Specific tensile elongation (mm), Tensile strain (mm/mm) and Tensile stress (MPa).

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Time | Elongation | Load | Specific tensile elongation | Tensile strain (Elongation) | Tensile stress |
| 2 | (s) | (mm) | (N) | (mm) | (mm/mm) | (MPa) |
| 3 | 0 | 0,01603 | -0,32846 | -0,00132 | -0,00003 | -0,01141 |
| 4 | 0,1 | 0,02247 | 3,38137 | 0,00512 | 0,0001 | 0,11748 |
| 5 | 0,2 | 0,03034 | 6,94066 | 0,01299 | 0,00026 | 0,24115 |
| 6 | 0,3 | 0,03918 | 10,32148 | 0,02182 | 0,00044 | 0,35861 |
| 7 | 0,4 | 0,0476 | 13,5313 | 0,03025 | 0,0006 | 0,47013 |
| 8 | 0,5 | 0,05558 | 16,34303 | 0,03823 | 0,00076 | 0,56782 |
| 9 | 0,6 | 0,06394 | 18,58927 | 0,04659 | 0,00093 | 0,64586 |
| 10 | 0,7 | 0,07252 | 20,48202 | 0,05517 | 0,0011 | 0,71163 |
| 11 | 0,8 | 0,08084 | 23,31186 | 0,06348 | 0,00127 | 0,80995 |
| 12 | 0,9 | 0,08896 | 26,14528 | 0,07161 | 0,00143 | 0,90839 |
| 13 | 1 | 0,09722 | 28,43645 | 0,07987 | 0,0016 | 0,98799 |
| 14 | 1,1 | 0,10555 | 31,19174 | 0,0882 | 0,00176 | 1,08372 |
| 15 | 1,2 | 0,11388 | 33,64328 | 0,09653 | 0,00193 | 1,1689 |
| 16 | 1,3 | 0,12226 | 36,59449 | 0,10491 | 0,0021 | 1,27144 |
| 17 | 1,4 | 0,13043 | 38,86968 | 0,11308 | 0,00226 | 1,35049 |
| 18 | 1,5 | 0,13891 | 41,52585 | 0,12156 | 0,00243 | 1,44277 |
| 19 | 1,6 | 0,14756 | 44,0269 | 0,1302 | 0,0026 | 1,52967 |
| 20 | 1,7 | 0,15551 | 46,30633 | 0,13815 | 0,00276 | 1,60886 |
| 21 | 1,8 | 0,16375 | 48,2326 | 0,1464 | 0,00293 | 1,67579 |

***Fig. 3.*** *Representative format of mechanical test data automatically produced by the specialized software (Bluehill 3)*

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
No. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

For *n* tested materials, each with *m* samples, there will be *n*m* CSV files. These files are grouped into *n* folders, each material in its own folder, named after the material.

The algorithm created in MATLAB (Annex 1) for processing these data goes through several stages, as follows:

1. reads all CSV files from the existing folders;
2. identifies and extracts the columns with strain and stress data;
3. saves the extracted data in an Excel file in the "Original data" sheet (optionally, a graphical representation of the curves for each material can be generated manually using the saved data – Fig. 4);

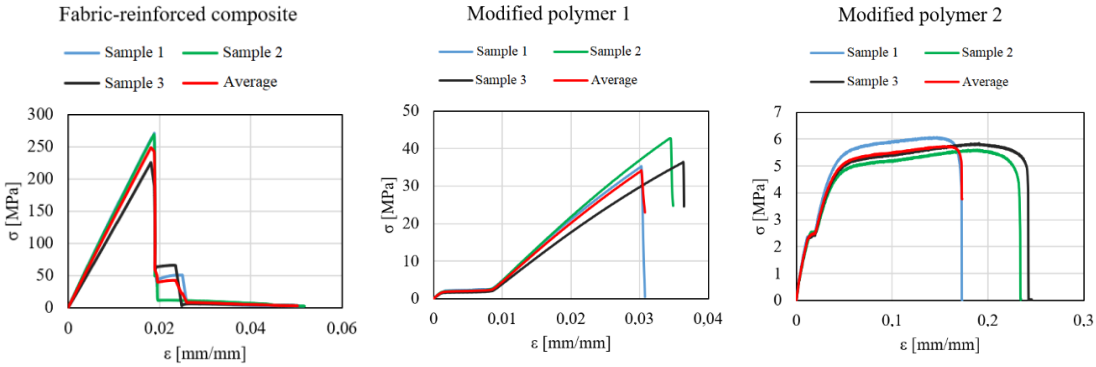


***Fig. 4.*** *Graphics obtained immediately after mechanical testing*

4. using the moving average method, the algorithm calculates the values of the trend curve;
5. the length of the data series recorded following the performed tests is adjusted (depending on the duration of the test, the data series for different tests may vary);
6. saves the new values in the same Excel file, in the "ALL processed curves" sheet;
7. automatically generates a graphic for each material, with redrawn curves (Fig. 5);
8. allows manual selection on the graph of the area on which the modulus of elasticity is to be calculated (Fig. 6);

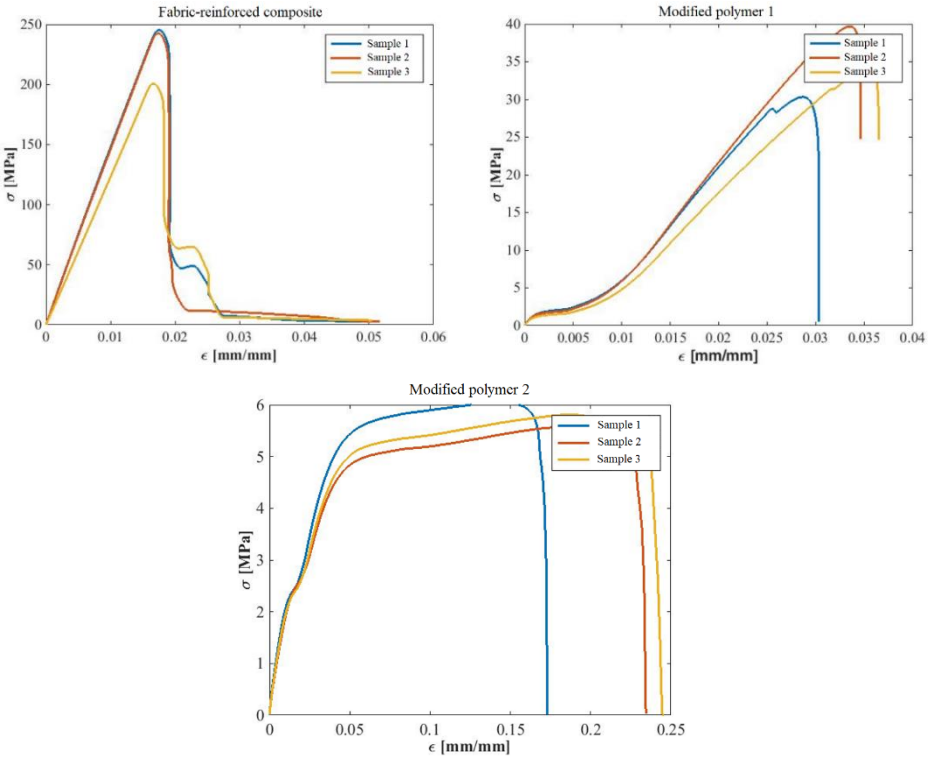


***Fig. 5.*** *Graphics obtained automatically by the algorithm, after adjusting the lengths of all vectors to the minimum length identified*

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
Nº. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

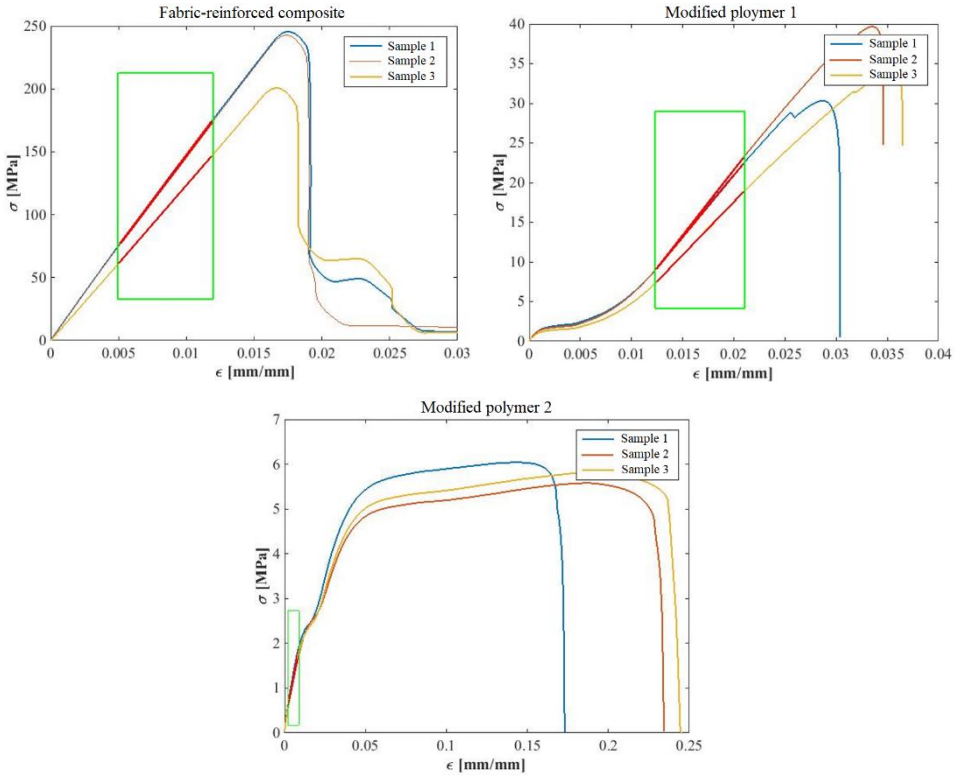


***Fig. 6.*** *Manual selection, on the graph, of the area on which the modulus of elasticity is to be calculated*

9. draws the tangent to each curve on the graph, within the selected area;

10. saves the values of the slopes of all curves in the Excel file, in the sheet "ALL Slopes" (Table 2);

11. identifies the values of the slopes that deviate by more than 5% from the average and eliminates the slopes of the respective curves;

12. calculates the average of the remaining slopes and their standard deviation;

13. saves the remaining slopes in the Excel file, in the sheet "GOOD and AVERAGE Slopes" (Table 3);

14. allows manual selection of the portion of the curves that includes the area of interest;

15. saves the new curves in the Excel file, in the sheet "Final Curves" (automatically generates the final shapes of the curves for all materials – Fig. 7);

16. calculates the average curve, for each material;

17. saves in the Excel file, in the sheet "AVERAGE Curves", the data representing the average curves of the materials.

***Table 2.*** *Values of the slopes of all curves, automatically calculated by the algorithm*

| | **$SE_1$ [Mpa]** | **$SE_2$ [Mpa]** | **$SE_3$ [Mpa]** |
|---|---|---|---|
| **Fabric-reinforced composite** | 14510 | 14382 | 12398 |
| **Modified polymer 1** | 1563 | 1653 | 1339 |
| **Modified polymer 2** | 205 | 180 | 192 |

***Table 3.*** *Values of the slopes of all curves, automatically calculated by the algorithm for the remaining slopes, after eliminating the ones with a deviation greater than 5%*

| | **$SE_1$ [Mpa]** | **$SE_2$ [Mpa]** | **$SE_3$ [Mpa]** | **Average [MPa]** | **St. dev.** |
|---|---|---|---|---|---|
| **Fabric-reinforced composite** | 14510 | 14382 | - | 14446 | 64 |
| **Modified polymer 1** | 1563 | 1653 | - | 1608 | 45 |
| **Modified polymer 2** | 205 | 180 | 192 | 192 | 10.20 |

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
Nº. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

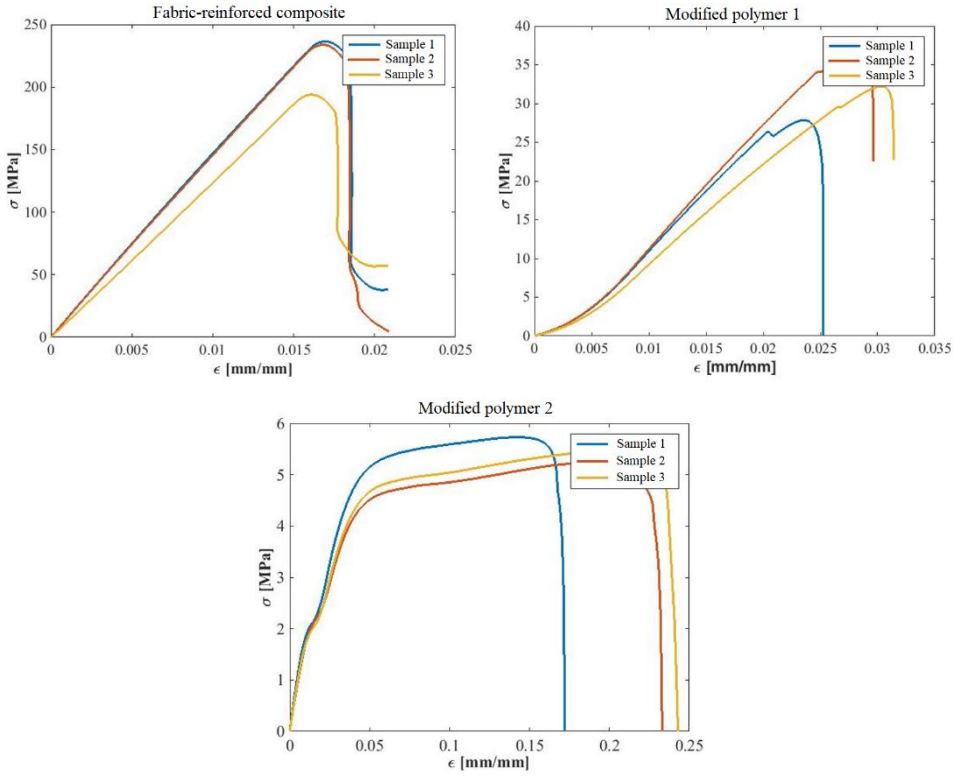


***Fig. 7.*** *Final shapes of the curves automatically represented by the algorithm for each mechanically tested material*

These final results are visibly much closer to a conventional curve, the average values of the monitored parameters being calculated on the average curve - a curve that best reproduces the shape of the closest curve, after the curves with a slope deviating more than 5% from the average have been eliminated (comparison: original vs. processed with MATLAB).

The algorithm proposed in this work calculates the slope and the longitudinal modulus of elasticity (Young's modulus) over the manually selected interval. Also, provides both the elastic modulus values for all curves, as well as the relevant values and their average, as seen in Table 3.

These extra steps and the proposed algorithm provide the foundation to rapidly and elegantly avoid some possible errors that can interfere with the mechanical testing results, especially when analysing composite materials.

## 4. Conclusions

The multitude of composite materials obtained through an increasing number of combinations of components can create situations that are difficult to manage in relation to results from the necessary testing of these materials. The automation of data processing is nothing new in the area of materials analysis and its utilisation in the present work comes with an optimisation tailored to the particularities of the aforementioned composite materials. Thus, this paper has demonstrated that the addition of some steps in an algorithm used for data processing of mechanical testing results can ensure greater reliability through elimination of possible errors that can appear during the initial processing of data. The demonstration was carried out using selected composite materials that presented significant differences both among themselves and compared to conventional materials.

Although the whole process still needs an operator to select an area on the obtained curves to be used in the calculation process, the much higher precision of the final results makes this process viable.

## References

**[1]. Carey J. P.**, *Introduction to braided composites*, *Handbook of Advances in Braided Composite Materials: Theory, Production, Testing and Applications*, p. 1-21, doi: 10.1016/B978-0-08-100369-5.00001-5, Jan. 2017.

**[2].** ***, *Mechanical Testing Methods | Anton Paar Wiki*, [Online]. Available: https://wiki.anton-paar.com/en/mechanical-testing-methods/, Accessed: Apr. 09, 2025.

**[3].** ***, *Mechanical Testing of Composites*, [Online]. Available: https://www.addcomposites.com/post/mechanical-testing-of-composites, Accessed: Apr. 09, 2025.

**[4].** ***, *MATLAB*, [Online]. Available: https://www.mathworks.com/products/matlab.html, Accessed: Apr. 09, 2025.
**[5].** ***, *8850 SERIES AXIAL-TORSION SYSTEM ±100 kN, ±1000 Nm / ±250 kN, ±2000 Nm*, 2024, [Online]. Available: www.instron.com, Accessed: Apr. 09, 2025.
**[6].** ***, *Hydraulic Power Unit 3621 Series (207 Bar)*, 2024, [Online]. Available: www.instron.com, Accessed: Apr. 09, 2025.
**[7].** ***, *8800MT CONTROLLER Intelligence and safety built in*, 2023, [Online]. Available: www.instron.com, Accessed: Apr. 09, 2025.
**[8].** ***, *Designation: D3039/D3039M − 08 Standard Test Method for Tensile Properties of Polymer Matrix Composite Materials 1*, doi: 10.1520/D3039_D3039M-08, 2008.
**[9].** ***, *Standard Test Method for Flexural Properties of Polymer Matrix Composite Materials 1*, doi: 10.1520/D7264_D7264M-21, 2008.
**[10].** ***, *Standard Practice Unnotched Compression Testing of Polymer Matrix Composite Laminates 1*, doi: 10.1520/D8066_D8066M, 2023.
**[11].** **Mocanu Costel Iulian**, *Rezistența materialelor*, 2nd ed. Galati: Zigotto, 2005.

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
No. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

# Annex 1

```
clear all;
clc;

selectedFolder = uigetdir('', 'Select a folder');
if isempty(selectedFolder)
    disp('Selection canceled.');
else
    subdirectories = dir(selectedFolder);
    subdirectories =
subdirectories(~ismember({subdirectories.name}, {'.', '..'}) &
[subdirectories.isdir]);
    if isempty(subdirectories)
        disp('Warning: No subdirectories found.');
    else
    end
lungimiVectoriEpsilon = [];
end
        for i = 1:length(subdirectories)
            currentName = subdirectories(i).name;
            currentSubdirectory = fullfile(selectedFolder,
currentName);
            NumeSubdirector = strtok(subdirectories(i).name, '.');
            csvFiles = dir(fullfile(currentSubdirectory, '*.csv'));
            if ~isempty(csvFiles)
                for j = 1:length(csvFiles)
                    currentCSVFile = fullfile(currentSubdirectory,
csvFiles(j).name);
                    tableData = readtable(currentCSVFile, 'Delimiter', ';');
                    epsilonColumnIndex =
findColumnByPartialName(tableData.Properties.VariableNames,
'Deform');
                    sigmaColumnIndex =
findColumnByPartialName(tableData.Properties.VariableNames,
'Stress');
                    if ~isempty(epsilonColumnIndex) &&
~isempty(sigmaColumnIndex)
                        epsilonColumn = str2double(strrep(tableData{:,
epsilonColumnIndex}, ',', '.'));
                        sigmaColumn = str2double(strrep(tableData{:,
sigmaColumnIndex}, ',', '.'));
                        epsilonColumn =
epsilonColumn(~isnan(epsilonColumn));
                        sigmaColumn =
sigmaColumn(~isnan(sigmaColumn));
                        epsilonColumn = abs(epsilonColumn);
                        sigmaColumn = abs(sigmaColumn);
                        lungimiVectoriEpsilon = [lungimiVectoriEpsilon;
length(epsilonColumn)];
                        grupuriCaractere = strsplit(NumeSubdirector, ' ');
                        ultimulGrup = grupuriCaractere{end};
                        epsilonColumnName =
matlab.lang.makeValidName(['Epsilon_' ultimulGrup '_'
num2str(j)]);
                        sigmaColumnName =
matlab.lang.makeValidName(['Sigma_' ultimulGrup '_'
num2str(j)]);
                        epsilonColumnNames{i}{j} =
epsilonColumnName;
                        sigmaColumnNames{i}{j} = sigmaColumnName;
                        validNameEpsilon =
matlab.lang.makeValidName(['Epsilon_' ultimulGrup '_'
num2str(j)]);
                        validNameSigma =
matlab.lang.makeValidName(['Sigma_' ultimulGrup '_'
num2str(j)]);
                        assignin('base', validNameEpsilon, epsilonColumn);
                        assignin('base', validNameSigma, sigmaColumn);
                        disp(['Storing the epsilon vector for'
validNameEpsilon ' and the sigma vector for ' validNameSigma ' in
workspace.']);
                    else
                        disp(['Worning: The columns for Deformation or
Stress were not found in the file header.']);
                    end
                end
            end
            nameBeforeDot = strtok(currentName, '.');
            disp(['The sigma and epsilon vectors for the directory '
nameBeforeDot ' were saved in the workspace.']);
        end

dataCell = cell(length(epsilonColumnNames) * 2, 2);
idx = 1;
for i = 1:length(epsilonColumnNames)
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        dataCell{idx, 1} = epsilonVarName;
        dataCell{idx, 2} = epsilonData;
        dataCell{idx+1, 1} = sigmaVarName;
        dataCell{idx+1, 2} = sigmaData;
        idx = idx + 2;
    end
end

allData = cell2table(dataCell, 'VariableNames', {'Nume',
'Termenul'});
[~, folderName, ~] = fileparts(selectedFolder);
excelFileName = fullfile(selectedFolder, [folderName '.xlsx']);
writetable(allData, excelFileName, 'Sheet', 'Date originale',
'WriteVariableNames', true);
disp(['Datele au fost salvate in fisierul Excel: ' excelFileName]);

windowSize = 60;
for i = 1:length(subdirectories)
    currentName = subdirectories(i).name;
    currentSubdirectory = fullfile(selectedFolder, currentName);
    figure('Name', currentName);
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        epsilonSmoothed = smooth(epsilonData, windowSize);
        sigmaSmoothed = smooth(sigmaData, windowSize);
        assignin('base', sigmaVarName, sigmaSmoothed);
        plot(epsilonSmoothed, sigmaSmoothed, 'DisplayName',
['Sample ' num2str(j)]);
        hold on;
    end
    title(['Curves ' strtok(currentName, '.')]);
    xlabel('\epsilon [mm/mm]');
    ylabel('\sigma [MPa]');
    legend('show');
saveas(gcf, fullfile(selectedFolder, ['Curves ' currentName '.jpg']));
    close all
end

minLength = min(lungimiVectoriEpsilon);
for i = 1:length(epsilonColumnNames)
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
```

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
No. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

```
        fEpsilon = @(x) interp1((1:length(epsilonData))', epsilonData,
x, 'linear', 'extrap');
        fSigma = @(x) interp1((1:length(sigmaData))', sigmaData, x,
'linear', 'extrap');
        epsilonAdjusted = fEpsilon(linspace(1, length(epsilonData),
minLength));
        sigmaAdjusted = fSigma(linspace(1, length(sigmaData),
minLength));
        assignin('base', epsilonVarName, epsilonAdjusted);
        assignin('base', sigmaVarName, sigmaAdjusted);
    end
end

for i = 1:length(subdirectories)
    currentName = subdirectories(i).name;
    NumeCurent = strtok(currentName, '.');
    grupuriCaractere = strsplit(NumeCurent, ' ');
    ultimulGrup = grupuriCaractere{end};
    figure('Name', ultimulGrup);
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        plot(epsilonData, sigmaData, 'DisplayName', ['Sample '
num2str(j)]);
        hold on;
    end
    title(['Curves ' strtok(currentName, '.')]);
    xlabel('\epsilon [mm/mm]');
    ylabel('\sigma [MPa]');
    legend('show');
    disp(['Select border for SLOPES, on the graph '
strtok(currentName, '.') '.']);
    [xSelected, ySelected] = ginput(2);
    hold on;
    plot(xSelected([1 2 2 1 1]), ySelected([1 1 2 2 1]), 'g-',
'LineWidth', 1);
    intervalX = sort(xSelected);
    intervalY = sort(ySelected);
    validVarNameX = matlab.lang.makeValidName(['Interval_X_'
strtok(currentName, '.')]);
    validVarNameY = matlab.lang.makeValidName(['Interval_Y_'
strtok(currentName, '.')]);
    assignin('base', validVarNameX, intervalX);
    assignin('base', validVarNameY, intervalY);
     NumeSubdirector = strtok(subdirectories(i).name, '.');
     grupuriCaractere = strsplit(NumeSubdirector, ' ');
     ultimulGrup = grupuriCaractere{end};
    numePanteVector = ['Slopes_'
matlab.lang.makeValidName(ultimulGrup)];
    eval([numePanteVector ' = []']);
    Pante = [];
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        idxInterval = epsilonData >= intervalX(1) & epsilonData <=
intervalX(2);
        epsilonInterval = epsilonData(idxInterval);
        sigmaInterval = sigmaData(idxInterval);
        [~, idx1] = min((epsilonInterval - intervalX(1)).^2 +
(sigmaInterval - intervalY(1)).^2);
        [~, idx2] = min((epsilonInterval - intervalX(2)).^2 +
(sigmaInterval - intervalY(2)).^2);
        closestX = epsilonInterval([idx1, idx2]);
        closestY = sigmaInterval([idx1, idx2]);
        p = polyfit(closestX, closestY, 1);
        dreaptaTendinta = polyval(p, epsilonInterval);
        panta = p(1);
        disp(panta)
        eval([numePanteVector ' = [ ' numePanteVector ' slope]']);
        validCurrentName =
matlab.lang.makeValidName(currentName);
        fieldName = matlab.lang.makeValidName(['Sample_'
num2str(j)]);
        rezultate.(validCurrentName).(fieldName).dreaptaTendinta =
dreaptaTendinta;
        rezultate.(validCurrentName).(fieldName).panta = panta;
        plot(epsilonInterval, dreaptaTendinta, 'r', 'LineWidth', 1,
'DisplayName', ['Right Samples ' num2str(j)]);
    end
saveas(gcf, fullfile(selectedFolder, ['Slopes ' ultimulGrup '.jpg']));
end

disp('SLOPE results have been calculated and displayed..');
for i = 1:length(subdirectories)
    currentName = subdirectories(i).name;
    currentSubdirectory = fullfile(selectedFolder, currentName);
    figure('Name', currentName);
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        plot(epsilonData, sigmaData, 'LineWidth', 2, 'DisplayName',
['Epruveta ' num2str(j)]);
        hold on;
    end
    title(['Curves ' strtok(currentName, '.')]);
    xlabel('\epsilon [mm/mm]');
    ylabel('\sigma [MPa]');
    legend('show');
    disp(['Select the CURVE AREA you want to represent, on the
graph ' strtok(currentName, '.') '.']);
    [xSelected, ySelected] = ginput(2);
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        idxChenar = epsilonData >= min(xSelected) & epsilonData
<= max(xSelected) & sigmaData >= min(ySelected) & sigmaData
<= max(ySelected);
        epsilonDataNou = epsilonData(idxChenar);
        sigmaDataNou = sigmaData(idxChenar);
        epsilonDataNou = epsilonDataNou - epsilonDataNou(1);
        sigmaDataNou = sigmaDataNou - sigmaDataNou(1);
        assignin('base', epsilonVarName, epsilonDataNou);
        assignin('base', sigmaVarName, sigmaDataNou);
        plot(epsilonDataNou, sigmaDataNou, 'r', 'LineWidth', 2,
'DisplayName', ['Sample ' num2str(j)]);
    end
end

noiLungimiVectoriEpsilon = [];
for i = 1:length(epsilonColumnNames)
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        noiLungimiVectoriEpsilon = [noiLungimiVectoriEpsilon;
length(epsilonData)];
    end
end

nouaMinLength = min(noiLungimiVectoriEpsilon);
for i = 1:length(epsilonColumnNames)
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
```

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
No. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

```
        sigmaData = evalin('base', sigmaVarName);
        fEpsilon = @(x) interp1((1:length(epsilonData))', epsilonData,
x, 'linear', 'extrap');
        fSigma = @(x) interp1((1:length(sigmaData))', sigmaData, x,
'linear', 'extrap');
        epsilonAdjusted = fEpsilon(linspace(1, length(epsilonData),
nouaMinLength));
        sigmaAdjusted = fSigma(linspace(1, length(sigmaData),
nouaMinLength));
        assignin('base', epsilonVarName, epsilonAdjusted);
        assignin('base', sigmaVarName, sigmaAdjusted);
    end
end

datePrelucrateCell = cell(length(epsilonColumnNames) * 2, 2);
idx = 1;
for i = 1:length(epsilonColumnNames)
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        epsilonData = evalin('base', epsilonVarName);
        sigmaData = evalin('base', sigmaVarName);
        datePrelucrateCell{idx, 1} = epsilonVarName;
        datePrelucrateCell{idx, 2} = epsilonData;
        datePrelucrateCell{idx+1, 1} = sigmaVarName;
        datePrelucrateCell{idx+1, 2} = sigmaData;
        idx = idx + 2;
    end
end
toateDatelePrelucrate = cell2table(datePrelucrateCell,
'VariableNames', {'Name', 'Term'});
writetable(toateDatelePrelucrate, excelFileName, 'Sheet', 'Curves
processed ALL ', 'WriteVariableNames', true, 'Range', 'A1');
disp(['The processed data (Epsilon and Sigma) were saved in the
Excel file: ' excelFileName]);
panteCell = cell(length(subdirectories), 2);
for i = 1:length(subdirectories)
    currentName = subdirectories(i).name;
      NumeSubdirector = strtok(subdirectories(i).name, '.');
      grupuriCaractere = strsplit(NumeSubdirector, ' ');
      ultimulGrup = grupuriCaractere{end};
    numePanteVector = ['Slopes_'
matlab.lang.makeValidName(ultimulGrup)];
    valoriPante = evalin('base', numePanteVector);
    panteCell{i, 1} = numePanteVector;
    panteCell{i, 2} = valoriPante;
end
tabelPante = cell2table(panteCell, 'VariableNames', {'Name',
'Slope'});
writetable(tabelPante, excelFileName, 'Sheet', 'ALL slopes',
'WriteVariableNames', true, 'Range', 'A1');
disp([' The slope values were saved in the Excel file.: '
excelFileName]);
for i = 1:length(subdirectories)
    NumeSubdirector = strtok(subdirectories(i).name, '.');
    grupuriCaractere = strsplit(NumeSubdirector, ' ');
    ultimulGrup = grupuriCaractere{end};
    numePanteVector = ['Slopes_'
matlab.lang.makeValidName(ultimulGrup)];
    numeEpsilonVector = ['Epsilon_'
matlab.lang.makeValidName(ultimulGrup)];
    numeSigmaVector = ['Sigma_'
matlab.lang.makeValidName(ultimulGrup)];
    valoriPante = evalin('base', numePanteVector);
    PantaMedie=mean(valoriPante);
    limitaDiferentaRelativa = (PantaMedie*5)/100;
        i = 1;
        pantaEliminata = 0;
        while i < length(valoriPante)
            difRelative = abs(mean(valoriPante) - valoriPante(i));
            if difRelative > limitaDiferentaRelativa
                valoriPante(i) = [];
                pantaEliminata = i;
            else
                i = i + 1;
            end
        end
    assignin('base', numePanteVector, valoriPante);
    if pantaEliminata > 0
        numeEpsilonEliminat = ['Epsilon_'
matlab.lang.makeValidName(ultimulGrup) '_'
num2str(pantaEliminata)];
        numeSigmaEliminat = ['Sigma_'
matlab.lang.makeValidName(ultimulGrup) '_'
num2str(pantaEliminata)];
        evalin('base', ['clear ' numeEpsilonEliminat ' '
numeSigmaEliminat]);
    end
end
    assignin('base', numePanteVector, valoriPante);
mediiPanteCell = cell(length(subdirectories), 3);
for i = 1:length(subdirectories)
    currentName = strsplit(strtok(subdirectories(i).name, '.'));
    ultimulGrup = currentName{end};
    numePanteVector = ['Slopes_'
matlab.lang.makeValidName(ultimulGrup)];
    valoriPante = evalin('base', numePanteVector);
    mediaPantelor = mean(valoriPante);
     mediiPanteCell{i, 1} = ultimulGrup;
    mediiPanteCell{i, 2} = valoriPante;
    mediiPanteCell{i, 3} = mediaPantelor;
end
tabelMediiPante = cell2table(mediiPanteCell, 'VariableNames',
{'Subdirectory', 'Slope', 'Slopes_average'});
writetable(tabelMediiPante, excelFileName, 'Sheet', 'GOOD and
AVERAGE slopes', 'WriteVariableNames', true, 'Range', 'A1');
disp(['SIGNIFICANT slopes and their means were saved in the
Excel file: ' excelFileName]);
datePrelucrateCell = cell(length(epsilonColumnNames) * 2, 2);
idx = 1;
for i = 1:length(epsilonColumnNames)
    for j = 1:length(epsilonColumnNames{i})
        epsilonVarName = epsilonColumnNames{i}{j};
        sigmaVarName = sigmaColumnNames{i}{j};
        try
            epsilonData = evalin('base', epsilonVarName);
            sigmaData = evalin('base', sigmaVarName);
            datePrelucrateCell{idx, 1} = epsilonVarName;
            datePrelucrateCell{idx, 2} = epsilonData;
            datePrelucrateCell{idx+1, 1} = sigmaVarName;
            datePrelucrateCell{idx+1, 2} = sigmaData;
            idx = idx + 2;
        catch
            disp(['Vector ' epsilonVarName ' or ' sigmaVarName ' was
not found in the workspace. Moving on to the next vector.']);
        end
    end
end
toateDatelePrelucrate = cell2table(datePrelucrateCell,
'VariableNames', {'Name', 'Term'});
writetable(toateDatelePrelucrate, excelFileName, 'Sheet', 'Curves
FINAL', 'WriteVariableNames', true, 'Range', 'A1');
disp(['FINAL curves (Epsilon and Sigma) were saved in the Excel
file: ' excelFileName]);

epsilonMediu = zeros(length(subdirectories),
length(epsilonAdjusted));
sigmaMediu = zeros(length(subdirectories),
length(epsilonAdjusted));
for i = 1:length(subdirectories)
    NumeSubdirector = strtok(subdirectories(i).name, '.');
    grupuriCaractere = strsplit(NumeSubdirector, ' ');
```

THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE IX. METALLURGY AND MATERIALS SCIENCE
No. 1 - 2025, ISSN 2668-4748; e-ISSN 2668-4756
Article DOI: https://doi.org/10.35219/mms.2025.1.05

```
    ultimulGrup = grupuriCaractere{end};
    numarVectori = 0;
    for j = 1:sigmaColumnIndex
        numeEpsilonVector = ['Epsilon_'
matlab.lang.makeValidName(ultimulGrup) '_' num2str(j)];
        numeSigmaVector = ['Sigma_'
matlab.lang.makeValidName(ultimulGrup) '_' num2str(j)];
        if evalin('base', ['exist('" numeEpsilonVector '", "var")']) &&
evalin('base', ['exist('" numeSigmaVector '", "var")'])
            epsilonVector = evalin('base', numeEpsilonVector);
            sigmaVector = evalin('base', numeSigmaVector);
            epsilonMediu(i, :) = epsilonMediu(i, :) + epsilonVector;
            sigmaMediu(i, :) = sigmaMediu(i, :) + sigmaVector;
            numarVectori = numarVectori + 1;
        else
            disp(['Vector ' numeEpsilonVector ' or ' numeSigmaVector '
was not found in the workspace. Moving on to the next vector.']);
        end
    end
    if numarVectori > 0
        epsilonMediu(i, :) = epsilonMediu(i, :) / numarVectori;
        sigmaMediu(i, :) = sigmaMediu(i, :) / numarVectori;
        epsilonSubdirectorCell{i} = epsilonMediu(i, :);
        sigmaSubdirectorCell{i} = sigmaMediu(i, :);
        numeMedieEpsilon = ['Epsilon_AVERAGE_'
matlab.lang.makeValidName(ultimulGrup)];
        numeMedieSigma = ['Sigma_AVERAGE_'
matlab.lang.makeValidName(ultimulGrup)];
        assignin('base', numeMedieEpsilon, epsilonMediu(i, :));
        assignin('base', numeMedieSigma, sigmaMediu(i, :));
    else
        disp(['There are no vectors for the subdirectory '
NumeSubdirector '. Moving on to the next subdirectory.']);
    end
end

toateMediileCell = cell(length(subdirectories), 2);
idx = 1;
for i = 1:length(subdirectories)
    NumeSubdirector = strtok(subdirectories(i).name, '.');
    grupuriCaractere = strsplit(NumeSubdirector, ' ');
    ultimulGrup = grupuriCaractere{end};
    numeMedieEpsilon = ['Epsilon_AVERAGE_'
matlab.lang.makeValidName(ultimulGrup)];
    numeMedieSigma = ['Sigma_AVERAGE_'
matlab.lang.makeValidName(ultimulGrup)];
    try
        epsilonMediu = evalin('base', numeMedieEpsilon);
        sigmaMediu = evalin('base', numeMedieSigma);
        toateMediileCell{idx, 1} = numeMedieEpsilon;
        toateMediileCell{idx, 2} = epsilonMediu;
        toateMediileCell{idx+1, 1} = numeMedieSigma;
        toateMediileCell{idx+1, 2} = sigmaMediu;
        idx = idx + 2;
    catch
        disp(['Averages for the subdirectory ' NumeSubdirector ' were
not found in the workspace. Moving to the next subdirectory.']);
    end
end

toateMediile = cell2table(toateMediileCell, 'VariableNames',
{'Name', 'Average'});
writetable(toateMediile, excelFileName, 'Sheet', 'AVERAGE
Curves', 'WriteVariableNames', true, 'Range', 'A1');
disp(['Epsilon and Sigma AVERAGES were saved in the Excel
file: ' excelFileName]);
```