Demonstrative irregular fatigue cycle counting by rainflow method implementation using a Python program

Raul CORMOS[1,2], Catalin Andrei NEAGOE*[,2,3], Miruna CIOLCA[2], Anton HADAR[2,4,5]

*Corresponding author
[1]INCAS – National Institute for Aerospace Research "Elie Carafoli", Iuliu Maniu Blvd. 220, 061126, Bucharest, Romania
*[,2]National University of Science and Technology POLITEHNICA Bucharest, Splaiul Independentei 313, 060042, Bucharest, Romania, catalin.neagoe@upb.ro
[3]Institute of Solid Mechanics of the Romanian Academy, Constantin Mille 15, 030167, Bucharest, Romania
[4]Technical Science Academy of Romania, Dacia Blvd. 26, 030167, Bucharest, Romania
[5]Academy of Romanian Scientists, Ilfov Street 3, 050045, Bucharest, Romania

DOI: 10.13111/2066-8201.2024.16.3.2

Received: 03 July 2024/ Accepted: 05 August 2024/ Published: September 2024

Abstract: *The main aim of this article is to present a fatigue cycle counting program for a given application. The application is an example from a given case presented in the literature, with the analytical solution discussed herein. Cycle counting for irregular fatigue spectra is important to evaluate for various given load cases, in different industries. Such evaluations are difficult to make without dedicated computer programs. Several specific methods are available in the technical literature to evaluate irregular fatigue spectrums. For this study, one of the most well-known and used methods was chosen, the rainflow-counting algorithm. This method extracts the number of cycles from an irregular fatigue recording which, in turn, is used in the Palmgren-Miner rule to evaluate the fatigue damage for the examined application, considering that the Wohler curve is given. The developed program is written in the Python programming language. The theoretical background is presented for the application and the use of it with the program algorithm.*

Key Words: *cycle counting, rainflow method, fatigue, Python*

1. INTRODUCTION

Fatigue degradation is an important cause of failure in the exploitation of modern metallic structures, where about 90% of mechanical failures are due to fatigue phenomena [1]. Fatigue loading is manifested in mechanical structures by alternated maximum and minim principal stress of various amplitude.

In the majority of the examined cases, fatigue loading spectra are registered using experimental devices. After the stress-time spectrum has been registered, the load spectrum is made. Usually, for real life applications, these spectra are highly irregular. In some cases, they are approximated with easy-to-read and register cycles.

The main purpose of this article is to present a demonstrative fatigue cycle counting program written in Python programming language for a simple irregular cycle, using the rainflow method. This program will load the fatigue cycle and count the number of cycles; however, no fatigue damage evaluation is performed. All command lines are presented and explained. Even though in most cases the primary working program in engineering is Excel together with the VBA application programming language, here, a demonstrative use of Python is performed. Fatigue cycle counting can be used for fatigue damage calculation using the classic Palmgren-Miner rule. The Palmgren-Miner rule has been the most commonly employed for fatigue damage evaluation [2-3].

Python is an object-oriented language with two compilator versions. In this article, the IDLE (Python 3.8 32-bit) version has been used, which is free of charge.

Irregular fatigue cycles are very common in industrial applications. Usually, when ultimate load cases are not considered for conservative fatigue evaluation, the fatigue spectra are being registered during the structure's normal exploitation procedure. Having the fatigue loading spectra, the cycle counting can be performed. One of the most known methods for fatigue cycle counting is the rainflow method [4-8], adopted herein.

2. RAINFLOW-COUNTING METHOD

2.1 Algorithm steps

Many methods were proposed for irregular cycle counting. The most widely accepted is the rainflow method. It was developed in Japan in the 1960s and since then it has been used and further enhanced in other methods. To demonstrate its use, a proposed example has been adapted from [9]. Consider the following irregular fatigue cycle:

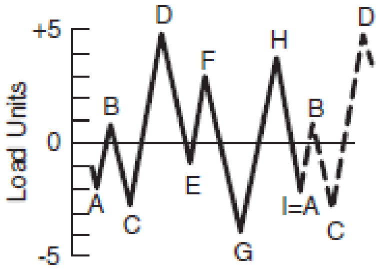


Fig. 1 – Irregular fatigue cycle [9]

The first step of the counting method is to rearrange the spectrum starting with the highest or lowest value:

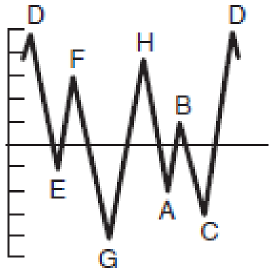


Fig. 2 – Fatigue spectra arrangement starting with the highest value [9]

A cycle is considered counted if a point projection value on an additional point is lower, as an example here are points E on FG and point A on AB. Thus, there are cycles EFE' and ABA', where the projection point has been recorded. After this, the corresponding cycles are eliminated and the process is repeated. In the meantime, the cycles are being recorded.

2.2 Palmgren-Miner rule

Considering a structural component from an isotropic metallic material, we can study its fatigue damage in a relatively simple way. Knowing the fatigue spectra and the allowable number of cycles for the given loading, the fatigue damage can be calculated using the Palmgren-Miner rule [9-10]:

$$\sum \frac{N_j}{N_{fj}} \leq 1 \tag{1}$$

where:

N_j – number of cycles for a given load;

N_{fj} – number of cycles to fail for a given load case, from Wohler's curve.

3. APPLICATION AND PROGRAM DESCRIPTION

3.1 Introduction

To demonstrate the application of cycle counting, a demonstrative application is considered with 9 registered points. After the analytical solution of the proposed problem has been completed, a Python program developed by the authors is presented to solve the given application.

Consider the points represented in the irregular fatigue cycle below:

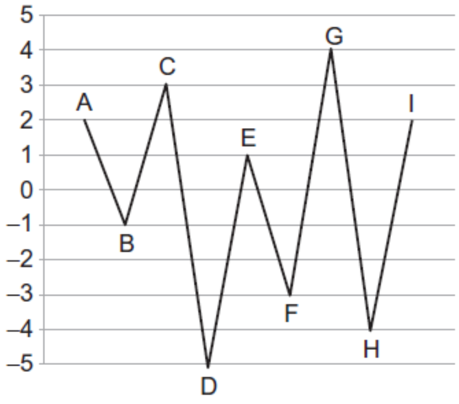


Fig. 3 – Rainflow application example [11]

Table 1 – Rainflow example table

Point	Stress amplitude - Sa [MPa]	Time [s]
A	2	1
B	-1	2
C	3	3
D	-5	4
E	1	5

F	-3	6
G	4	7
H	-4	8

The first step is to organize the cycle starting from the lowest or the highest valley:

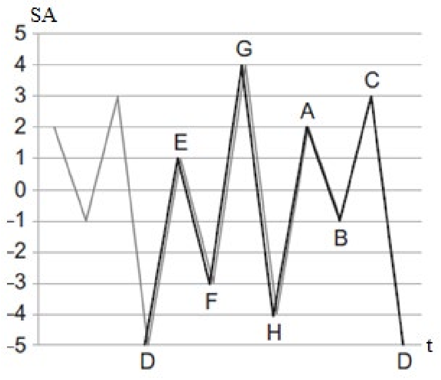


Fig. 4 – Rainflow cycle reorganization [11]

From the first cycle counting it can be observed that the first cycles that are numbered are on EF and AB:

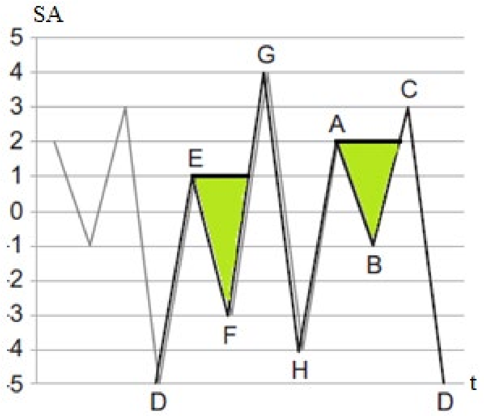


Fig. 5 – Rainflow first cycles identification [11]

Eliminating EF and AB, the cycle becomes:

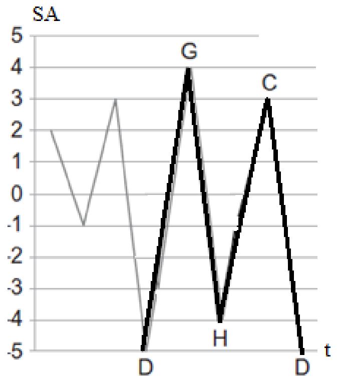


Fig. 6 – Fatigue cycle after elimination of EF and AB cycles [11].

It can be seen that the next portion to be identified is cycle HC, as highlighted in the figure below:

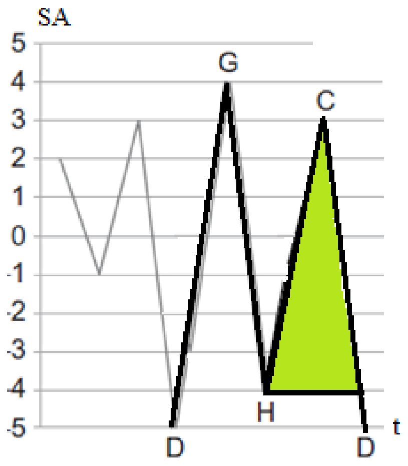


Fig. 7 – Fatigue cycle after HC cycle [11].

Eliminating the HC cycle, only the final cycle DGD will remain. Thus, after these counting steps, we can determine that there are 4 cycles in the given data.

3.2 Program overall description

The considered Python program was written for the given 9 points and for the presented solved application. The program consists of three main parts, as illustrated below:

- data import;
- computation using the rainflow method;
- result export and presentation.

The developed program is used for demonstrative purposes for a simple rainflow fatigue counting example.

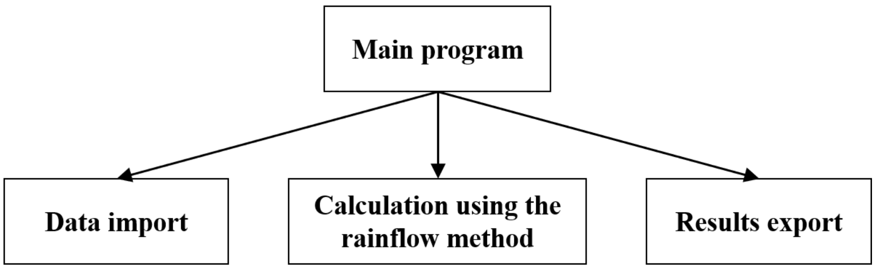


Fig. 8 – Python program short top-down design

3.3 Data import

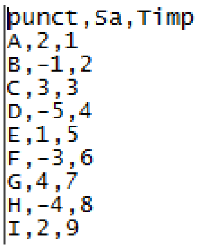
The first component of the program is detailed in this section. The purpose of this part is to import a data sample:

fisier=open("Cicluri_Young_Csv.csv","r") – reads the input data
linie_fisier=fisier.readline() – first line reading
punct=[]
amplitudine=[]
timp=[]
$\Big\}$ Lists definitions for point amplitude and time

```
counter_i=0 – counter for row lists
while linie_fisier!="":
   p,a,t=linie_fisier.split(",")
   if counter_i!=0:
      punct.append(p)
      amplitudine.append(float(a))
      timp.append(float(t))
   counter_i+=1
   linie_fisier=fisier.readline()
```

-reading of all lines
-extraction of subscripts
-completion of lists

The input file format is presented in figure below, where Sa is the stress amplitude:



```
punct,Sa,Timp
A,2,1
B,-1,2
C,3,3
D,-5,4
E,1,5
F,-3,6
G,4,7
H,-4,8
I,2,9
```

Fig. 9 – Input file format

3.4 Computation using the rainflow method

```
sigma_amplitudine=amplitudine[:] – creation of amplitude list
id_ciclu=0 – id cycles corresponding to the top
id_ciclu_aditional=0 – id additional cycle
lista_id_ciclu=[] – list of additional cycle
for i in range(1,len(sigma_amplitudine)-1):
   if i<len(sigma_amplitudine) and sigma_amplitudine[i]<sigma_amplitudine[i+1]:
# and (abs(sigma_amplitudine[i]-
sigma_amplitudine[i+1])<=abs(sigma_amplitudine[i+1]-sigma_amplitudine[i+2])): - for
loop used to evaluate conditions for rainflow
      numar_cicluri+=1
      id_cilcu=i
      lista_id_ciclu.append(i)
  - Identification of first cycle
      if sigma_amplitudine[i+1]>sigma_amplitudine[i]:
         id_ciclu_aditional=i-1
      else:
         id_ciclu_aditional=i+1
      lista_id_ciclu.append(id_ciclu_aditional)
```

The next part will reset and identify the rest of the remaining cycles.

```
for i in range(0,len(lista_id_ciclu)):
   punct[lista_id_ciclu[i]]="Eliminare"
   sigma_amplitudine[lista_id_ciclu[i]]="Eliminare"
sigma_amplitudine_temp=[]
punct_temp=[]
for i in range(0,len(sigma_amplitudine)):
   if sigma_amplitudine[i]!="Eliminare":
       sigma_amplitudine_temp.append(sigma_amplitudine[i])
   if punct[i]!="Eliminare":
       punct_temp.append(punct[i])
```

3.5 Results export

For the results presentation, the following code lines were written within the Python program:

```
punct=punct_temp
sigma_amplitudine=sigma_amplitudine_temp
print(punct)
print(sigma_amplitudine)
print(numar_cicluri)
fisier.close()
```

4. RESULTS ANALYSIS

The program's sequence presentation is displayed in Fig. 10, with the necessary steps.

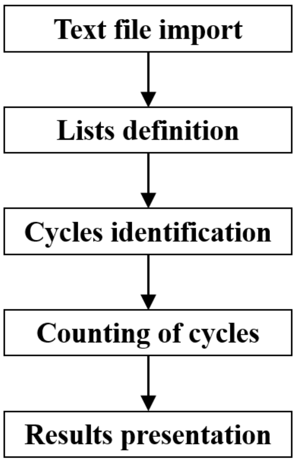
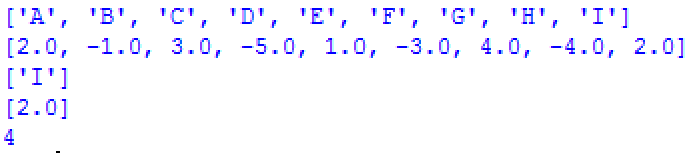


Fig. 10 – Program sequence presentation

After running the developed program, the following results are obtained.



```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
[2.0, -1.0, 3.0, -5.0, 1.0, -3.0, 4.0, -4.0, 2.0]
['I']
[2.0]
4
```

Fig. 11 – Results printing

The four lines are:
- points used, the I point is repetition at end of A point;
- amplitude of each point;
- last point;
- last point amplitude;
- number of counted cycles.

The estimated number of cycles obtained from the given data example matches the real number of registered cycles, as seen from the previous figures. Therefore, the proposed Python program can accurately count the number of cycles from an irregular fatigue spectrum.

5. CONCLUSIONS

After writing the program and analyzing the obtained data, the following conclusions can be stated:

1. The main purpose of this article was to create a small demonstrative software program for irregular fatigue cycle counting for a specific application.
2. The numerical computation of the number of cycles that was performed is in accordance with the observed data.
3. The capabilities of the program can be enhanced to eliminate low-amplitude loadings and to be used for longer, extended cycles.
4. Knowing the Wohler fatigue curve for a given material and application, the algorithm can be further developed to include fatigue damage assessment using the Palmgren-Miner rule.

ACKNOWLEDGEMENTS

This work was supported by the POCU/993/6/13-153178 grant, co-financed by the European Social Fund through the Sectorial Operational Programme Human Capital 2014-2020.

REFERENCES

[1] N. S. Gokhale, S. S. Deshpande, S. V. Bedekar, A. N. Thite, *Practical finite element analysis*, *Finite to Infinite*, 2008.

[2] L. Gharani, *Unlock Excel VBA and Excel Macros*, Udemy.

[3] J. Keyser, *How to Program: Computer Science Concepts and Python Exercises*, The Great Courses, 2016.

[4] R. Stelzer, B. Carlton, S. Mazzoni, *Comparison of cycle counting methods for potential liquefaction and structural fatigue assessment*, 17th World Conference on Earthquake Engineering (17WCEE), Sendai, Japan, September 13-18, 2020.

[5] J. M. Twomey, D. Y. Chen, M. D. Osterman, M. G. Pecht, Development of a cycle counting algorithm with temporal parameters, *Microelectronics Reliability*, vol. **109**, 113652, ISSN 0026-2714, 2020.

[6] G. Liu, D. Wang, Z. Hu, *Application of the rain-flow counting method in fatigue*, 2nd International Conference on Electronics, Network and Computer Engineering (ICENCE 2016), Yinchuan, China, August 13-14, Atlantis Press, 2016.

[7] S. H. Baek, S. S. Cho, W. S. Joo, Fatigue life prediction based on the rainflow cycle counting method for the end beam of a freight car bogie, *International Journal of Automotive Technology*, vol. **9**, pp. 95-101, ISSN 1229-9138, 2008.

[8] G. K. Tayi, Optimal inventory cycle counting, *Omega*, vol. **13**(6), pp. 535-539, ISSN 0305-0483, 1985.

[9] N. E. Dowling, *Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue, Fourth Edition*, Pearson, 2013.

[10] R. I. Stephens, A. Fatemi, R. R. Stephens, H. O. Fuchs, *Metal Fatigue in Engineering, Second Edition*, John Wiley & Sons, 2001.

[11] Y. L. Lee, M. E. Barkey, H.T. Kang, *Metal Fatigue Analysis Handbook: Practical Problem-Solving Techniques for Computer-Aided Engineering,* Elsevier, 2012.