ISSN 2286-4822
www.euacademic.org

EUROPEAN ACADEMIC RESEARCH
Vol. XI, Issue 4/ July 2023

Impact Factor: 3.4546 (UIF)
DRJI Value: 5.9 (B+)

Scientific Programming for the Solution of Problems in Oscillating Physical Systems

ANTHONY JAYRO NASCIMENTO RIBEIRO
Federal Institute of Pará – IFPA
anthonyjayro8@gmail.com
JAIME LUIZ CARDOSO DA CRUZ FILHO
Federal Institute of Pará – IFPA
jaime.filho@ifpa.edu.br
WILSON LUNA MACHADO ALENCAR
Federal Institute of Pará – IFPA
wilson.alencar@ifpa.edu.br

Abstract:
Problems related to the nature of oscillations are diverse and are present in all areas of human life, understanding their simplest cases is necessary for the development of more complex systems. Therefore, the main objective of this project, in addition to scientific initiation, is to understand the simplest cases of oscillatory systems. The methodology for this teaching is the use of programming to build models that accurately describe the cases of simple and damped harmonic oscillatory motions. The programming language chosen was Python because it is easy and intuitive, in order not to present an obstacle to learning. As a numerical analysis method, the fourth-order Runge Kutta method was used to obtain greater precision in the results.

Keywords: Physics; Oscillations; Programming, Python

1. INTRODUCTION

Oscillatory movement is understood as any periodic movement that, due to the action of a restoring force, has its direction of movement reversed at a given moment. Understanding this phenomenon is fundamental for engineering and basic sciences, as this event is intrinsically linked to our daily lives, whether in the buildings we visitor in the particles that make us up.

However, the analysis of oscillatory movements becomes quite complex when considering that other forces action the system, such as friction or any other force that forces the system to oscillate. To solve this problem, we use scientific programming.

Scientific programming is an increasingly important tool in physics, allowing scientists and researchers to model, simulate and analyze complex systems more quickly and accurately. Programming is today a fundamental skill for anyone who wants to excel in the area of theoretical or computational physics (Ginsparg, 2011).

With the increasing availability of data and advances in computing technology, scientific programming is critical to exploring new physical phenomena and developing technologies that impact our daily lives. Programming is now an indispensable tool for data analysis and simulation of complex physical systems" (Landau and Binder, 2014).

In this article we use the Python language that has become an increasingly important tool in the field of physics. The language stands out for being easy to learn, having a large community of developers and a vast library of specific modules for the scientific area. According to Fangohr (2018), Python is now a dominant language in computational physics and an important resource for solving complex problems in the area. With the help of libraries like Num Py, Math, and Mat plot lib, you can perform complex numerical calculations and data analysis faster and more efficiently.

Scientific programming with Python is also useful in creating visualizations and graphs that help in understanding and interpreting results. Python has a variety of data visualization libraries, allowing the creation of graphs and images that can help understand and communicate results more efficiently" (VanderPlas, 2016).

We apply the fourth-order Runge - Kutta numerical solution method to develop a script capable of solving problems related to simple, damped and forced harmonic motions, which were the oscillatory motions chosen for this study.

2. METHODOLOGY

For the development of this work, we performed the following steps:

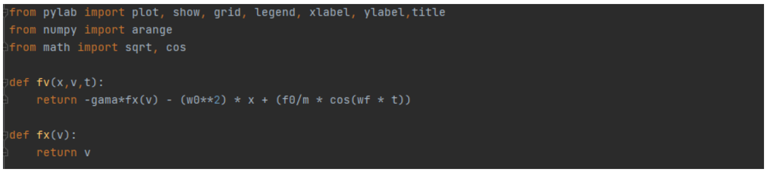
- **Study of oscillating physical systems:** In this step, a bibliographic review on oscillating physical systems was carried out, with the aim of understanding the fundamental concepts and the equations that describe them.
- **Development of the mathematical model:** Based on the concepts learned in the previous step, a mathematical model was developed for the chosen oscillating physical system. The model was described by the differential equation, the so-called equation of motion, which can be seen below:

$$m\frac{d^2x(t)}{dt^2} + b\frac{dx(t)}{dt} + k\,x(t) = F_0\cos(\omega t)$$

- **Implementation in Python:** Using the Numpy, Math and Matplotlib libraries, the mathematical model was implemented in Python. The numerical method of Runge-Kutta was used of fourth order to solve the differential equations, and at the end the graphics were plotted showing the particularities.

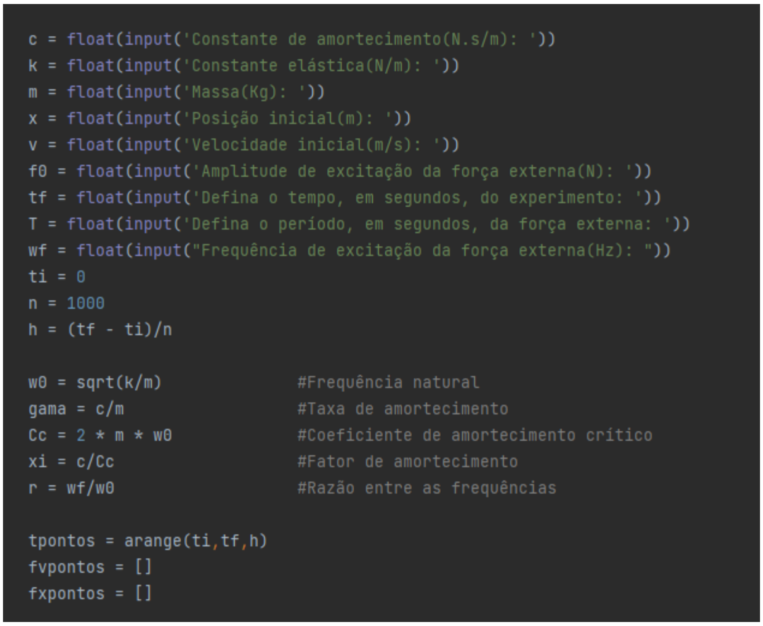
Figure 1 shows the import of the necessary libraries for the development of the script along with the equations to be used. Figure 2 presents the declaration of the adopted constants, while figure 3 presents the graphics formatting. Finally, figure 4 shows the creation of the graphics in the script.

Figure 1: Importing libraries and developing equation



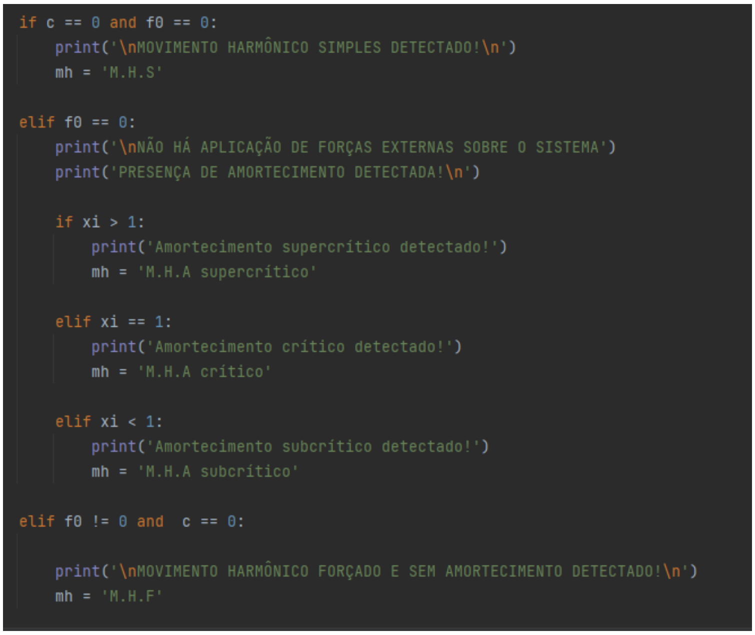
```
from pylab import plot, show, grid, legend, xlabel, ylabel,title
from numpy import arange
from math import sqrt, cos

def fv(x,v,t):
    return -gama*fx(v) - (w0**2) * x + (f0/m * cos(wf * t))

def fx(v):
    return v
```

Figure 2: Declaration of constants



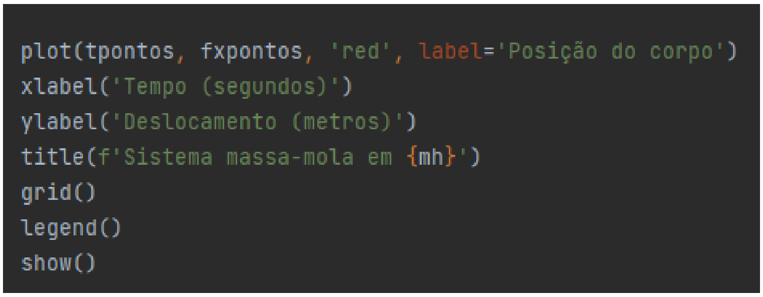
```
c = float(input('Constante de amortecimento(N.s/m): '))
k = float(input('Constante elástica(N/m): '))
m = float(input('Massa(Kg): '))
x = float(input('Posição inicial(m): '))
v = float(input('Velocidade inicial(m/s): '))
f0 = float(input('Amplitude de excitação da força externa(N): '))
tf = float(input('Defina o tempo, em segundos, do experimento: '))
T = float(input('Defina o período, em segundos, da força externa: '))
wf = float(input("Frequência de excitação da força externa(Hz): "))
ti = 0
n = 1000
h = (tf - ti)/n

w0 = sqrt(k/m)              #Frequência natural
gama = c/m                  #Taxa de amortecimento
Cc = 2 * m * w0             #Coeficiente de amortecimento crítico
xi = c/Cc                   #Fator de amortecimento
r = wf/w0                   #Razão entre as frequências

tpontos = arange(ti,tf,h)
fvpontos = []
fxpontos = []
```

Figure 3: Help with chart formatting



```
if c == 0 and f0 == 0:
    print('\nMOVIMENTO HARMÔNICO SIMPLES DETECTADO!\n')
    mh = 'M.H.S'

elif f0 == 0:
    print('\nNÃO HÁ APLICAÇÃO DE FORÇAS EXTERNAS SOBRE O SISTEMA')
    print('PRESENÇA DE AMORTECIMENTO DETECTADA!\n')

    if xi > 1:
        print('Amortecimento supercrítico detectado!')
        mh = 'M.H.A supercrítico'

    elif xi == 1:
        print('Amortecimento crítico detectado!')
        mh = 'M.H.A crítico'

    elif xi < 1:
        print('Amortecimento subcrítico detectado!')
        mh = 'M.H.A subcrítico'

elif f0 != 0 and  c == 0:

    print('\nMOVIMENTO HARMÔNICO FORÇADO E SEM AMORTECIMENTO DETECTADO!\n')
    mh = 'M.H.F'
```

Figure 4: Creating the chart



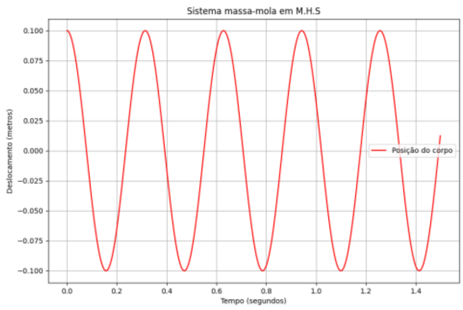
```
plot(tpontos, fxpontos, 'red', label='Posição do corpo')
xlabel('Tempo (segundos)')
ylabel('Deslocamento (metros)')
title(f'Sistema massa-mola em {mh}')
grid()
legend()
show()
```

3. RESULTS AND DISCUSSION

After the development of the script, we performed the tests indicating some parameters to evaluate them. Some graphs were generated to describe the position of a body connected to a spring (springmass system), with the absence or presence of dissipative forces and external forces in the system, over time.

In Graph 1, there are no dissipative forces or external forces acting on the system, characterizing simple harmonic motion, where the body will oscillate perpetually.

Graph 1: Spring mass system executing simple harmonic motion.

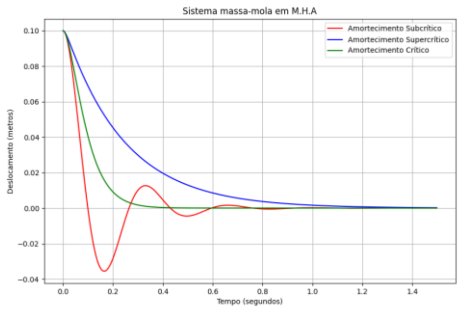


For Graph 1, the parameters used were the following:

Damping constant: 0 N.s/m
Elastic constant: 800 N/m
Mass: 2 Kg
Starting position: 0.1 meters
Starting speed: 0 m/s
Amplitude/Intensity of external force: 0 N
Time, in seconds, of the experiment: 1.5

In Graph 2, the system has a dissipative force, called damping, which, depending on its intensity, can be of three types: Sub-critical, Super-critical and Critical. Sub-critical damping is the lowest intensity, systems with this type of damping are able to oscillate, but the amplitude of oscillations decreases over time. In turn, the Super-critical damping is the most intense, systems subjected to this type are not able to oscillate and return to the equilibrium position after the initial disturbance. Finally, we have the critical damping, it has medium intensity, the systems subject to this type do not oscillate and return to the resting position very quickly.

Graph 2: Spring mass system executing damped harmonic motion

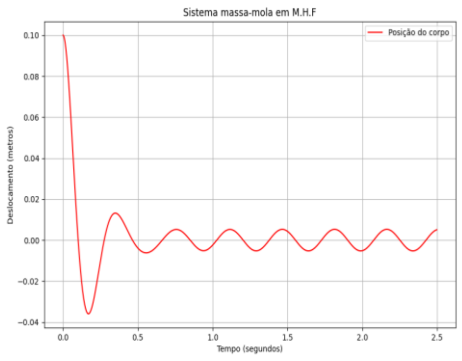


For Graph 2, the parameters used were the following:

Damping constant(subc): 25 N.s/m
Damping constant (supc): 200 N.s/m
Damping constant(c): 80 N.s/m
Elastic constant: 800 N/m
Mass: 2 kg
Starting position: 0.1m
Starting speed: 0 m/s
Amplitude/Intensity of external force: 0 N
Experiment time: 1.5 s

Graph 3 has a system subject to dissipative forces and external forces. The external force returns energy to the system, while forcing it to oscillate with its own frequency.

Graph 3: Spring mass system executing forced harmonic motion.



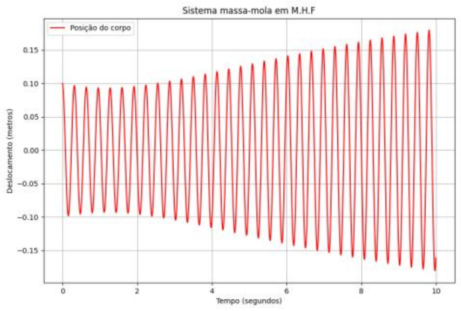
For Graph 3, the parameters used were the following:

Damping constant: 25 N.s/m
Elastic constant: 800 N/m
Mass: 2 kg

Starting position: 0.1 meters
Starting speed: 0 m/s
Amplitude/Intensity of external force: 2.5 N
External power frequency: 18 Hz
Experiment time: 2.5 s

Finally, we have graph 4, where the phenomenon of resonance occurs, which is characterized by the equality between the frequency of the external force and the natural frequency of the system, resulting in a perpetual increase in the amplitude of oscillations. There are no dissipative forces in the system.

Graph 4: Spring Mass System in Resonance



For Graph 4, the parameters used were the following:

Damping constant: 0.5 N.s/m
Elastic constant: 800 N/m
Mass: 2 kg
Starting position: 0.1 meters
Starting speed: 0 m/s
Amplitude/Intensity of external force: 2.5 N
External power frequency: 20 Hz
Experiment time: 10 s

4. FINAL CONSIDERATIONS

In this work, it was possible to observe and understand the importance of studying oscillations and their impact on human life, especially in engineering. In addition, the importance of computational tools was proven, as the simulations generated from the scripts created were fundamental allies in reducing expenses and, mainly, in the dissemination of knowledge.

The development of the script enabled the accurate and efficient simulation of the studied movements, with low cost of material, in addition to the inclusion of high school students, bringing them closer to the scientific environment and allowing contact with higher education content.

BIBLIOGRAPHICAL REFERENCES

1. Ginsparg, P. The global village and the ivory tower: reflections on the past, present and future of ar Xiv. *Journal of Physics: Conference Series*, 331(1), 01200, 2011.
2. Landau, D. P., & Binder, K. A guide to Monte Carlo simulations in statistical physics. Cambridge University Press, 2014.
3. Fangohr, H. Python in computational science. In Computational Science and Its Applications--ICCSA 2018. Springer, pp. 3-16, 2018.
4. VanderPlas, J. Python data science handbook: Essential tools for working with data. O'Reilly Media, Inc, 2016.