SIMULAÇÃO NUMÉRICA EM PYTHON DO PÊNDULO COM ARRASTO AERODINÂMICO PARA O ENSINO DE FÍSICA

RAFAEL OLIVEIRA ALVES[1]; PATRICK RITTER [2]; CARLOS ALBERTO VAZ DE MORAIS JUNIOR[3]

[1] *Universidade Federal de Pelotas – rafael_alvescnt@hotmail.com*
[2] *Universidade Federal de Pelotas – patrickritter23@yahoo.com*
[3] *Universidade Federal de Pelotas – cavmjunior@ufpel.edu.br*

1. INTRODUÇÃO

No ensino de Física, tradicionalmente baseado em formatos expositivos e demonstrações experimentais presenciais, persistem dificuldades relacionadas à motivação dos estudantes e à visualização de fenômenos dinâmicos complexos; conceitos de mecânica, em particular os associados ao movimento de pêndulos, podem permanecer abstratos quando apresentados apenas por meio de fórmulas e gráficos estáticos. Assim, torna-se fundamental explorar metodologias que promovam a construção ativa do conhecimento e ampliem a compreensão conceitual dos discentes. O presente projeto tem como objetivo tornar o ensino de Física mais eficaz por meio de uma simulação em Python do pêndulo simples considerando o arrasto aerodinâmico. A proposta consiste em desenvolver um ambiente computacional interativo que permita aos alunos manipular parâmetros do sistema (comprimento do fio, coeficiente de arrasto, amplitude inicial) e observar, em tempo real, os efeitos dessas variáveis sobre o movimento oscilatório.

O algoritmo implementado resolve o sistema por integração numérica utilizando o método de Runge-Kutta de quarta ordem. As rotinas principais são nomeadas no código e a implementação pré-calcula vetores temporais e energéticos para posterior visualização. A arquitetura do programa foi construída sobre bibliotecas consolidadas – `NumPy` para álgebra vetorial e `matplotlib` para plotagem, animação (`FuncAnimation`) e widgets (`Slider`, `Button`, `CheckButtons`) – permitindo ajuste em tempo real dos parâmetros $k, \gamma, Q, \omega, x_0$, controle da animação e geração de gráficos de espaço de fases, energia, posição e velocidade.

A utilização de simulações computacionais no ensino de Ciências tem sido amplamente investigada na literatura. MILLER; JUNGER (2010) demonstram que ambientes virtuais interativos podem aumentar o engajamento e promover a experimentação segura de fenômenos físicos. LEE et al. (2011) ressaltam que simulações facilitam a compreensão de sistemas dinâmicos ao permitir múltiplas repetições e visualizações detalhadas dos resultados. Esses estudos indicam que a integração de ferramentas digitais, como a plataforma Python, pode fortalecer a aprendizagem de forma significante e apoiar diferentes estilos cognitivos dos estudantes.

2. ATIVIDADES REALIZADAS

As atividades do projeto foram organizadas de forma sequencial, contemplando desde a modelagem teórica até a validação computacional do algoritmo. Inicialmente, foram definidas as equações diferenciais do pêndulo amortecido e forçado,

$$\dot{x} = v, \quad \dot{v} = -k\sin(x) - \gamma v + Q\sin(\Omega t),$$

e estabelecidos os parâmetros físicos (k, γ, Q, Ω, l) e numéricos (Δt, t_{max}) que seriam utilizados. Em seguida, estruturou-se a arquitetura do código em Python, de modo a separar as rotinas de integração, simulação e visualização, facilitando a manutenção e futuras expansões.

Na etapa de implementação, utilizou-se o método de Runge–Kutta de 4ª ordem (RK4), escolhido pela precisão e estabilidade em sistemas não-lineares. A função `runge_kutta_4` foi responsável pela integração temporal, acoplada à função `derivadas`, que calcula as taxas de variação do sistema. A rotina `simular` foi implementada para gerar os vetores temporais de posição, velocidade e energia, enquanto a função `calcular_energia` permitiu o acompanhamento da dissipação e do efeito do forçamento. Os parâmetros iniciais foram definidos e testados, garantindo que o algoritmo pudesse reproduzir regimes amortecidos, quase-resonantes e forçados.

A terceira atividade consistiu na construção da interface interativa utilizando as bibliotecas `NumPy` e `matplotlib`. O ambiente gráfico foi estruturado com múltiplos painéis: animação do pêndulo, espaço de fases, evolução temporal de posição/velocidade, energia total e força externa aplicada. Foram implementados controles interativos como `Slider` (para k, γ, Q, Ω, x_0), `Button` (Play/Reset) e `CheckButtons` (exibição de energia e trajetória), todos atualizados dinamicamente pelas funções `atualizar_frame` e `update_params`. Essa organização garantiu uma experiência de exploração em tempo real, com clareza visual e flexibilidade para manipulação dos parâmetros.

Por fim, foi realizada a documentação do código-fonte, incluindo instruções para execução, dependências necessárias (`numpy`, `matplotlib`) e exemplos de simulação. Também foi incorporada a função para salvar a tela da aplicação em execução, permitindo a inclusão da imagem ilustrativa no relatório. A Figura 1 mostra a interface interativa em funcionamento, evidenciando os gráficos gerados e os controles disponíveis. Ressalta-se que, como continuidade do projeto, está prevista a disponibilização do código em repositório público (GitHub) e a realização de oficinas pedagógicas com pré-teste e pós-teste, de modo a avaliar a eficácia da ferramenta no processo de ensino-aprendizagem em Física.

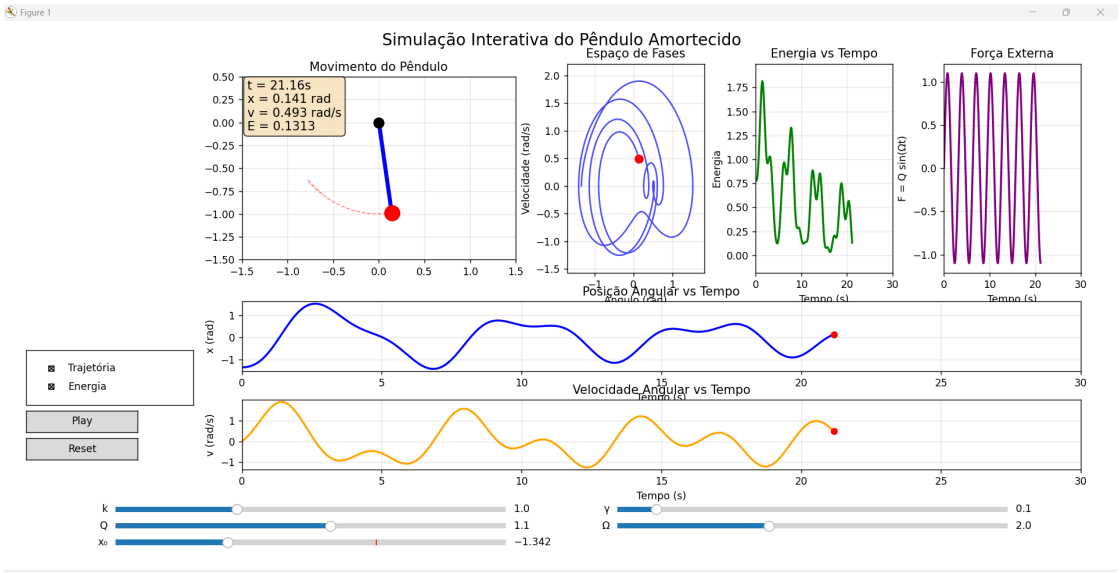

Figura 1: Interface da simulação interativa do pêndulo amortecido e forçado, desenvolvida em Python. O ambiente permite manipulação de parâmetros físicos e visualização em tempo real de espaço de fases, energia, posição e velocidade.

CONSIDERAÇÕES FINAIS

Do ponto de vista computacional, o uso das bibliotecas `NumPy` e `matplotlib` possibilitou uma implementação clara e eficiente, com recursos de interação e animação capazes de enriquecer tanto o estudo individual quanto atividades em grupo. A organização modular do código, o detalhamento dos parâmetros e a documentação produzida favorecem a reprodutibilidade e a adaptação da ferramenta a outros contextos de ensino e pesquisa.

Embora a aplicação direta em turmas ainda não tenha sido realizada, o projeto delineia com clareza esse passo como etapa futura. O repositório público a ser disponibilizado no *GitHub* permitirá a ampla difusão da ferramenta, oferecendo a docentes e discentes a oportunidade de utilizá-la, adaptá-la e contribuir com melhorias. A realização de pré-testes e pós-testes, acompanhados de análise estatística, constituirá a estratégia metodológica para avaliar a eficácia pedagógica da simulação no ensino de Mecânica.

O produto imediato e verificável deste projeto é o algoritmo e sua interface de visualização — desenvolvidos, testados e documentados durante a disciplina — que serão disponibilizados em repositório público (GitHub) para download, inspeção e contributo comunitário. A aplicação em sala de aula (oficinas, atividades práticas e avaliações por pré-teste/pós-teste) constitui a etapa subsequente prevista para implementação futura.

Conclui-se, portanto, que este trabalho cumpriu seu objetivo central de desenvolver e validar um algoritmo interativo para visualização do pêndulo, estabelecendo uma base sólida para futuras investigações de caráter educacional. A integração entre rigor numérico, clareza computacional e potencial pedagógico confere ao projeto relevância tanto para a formação em Física quanto para a inovação em metodologias de ensino.

3. REFERÊNCIAS

1. MILLER, J.T.; JÜNGER, B. Interactive Virtual Physics Laboratories: Enhancing Student Engagement and Understanding. *Journal of Science Education and Technology*, Amsterdam, v. 19, n. 3, p. 243–252, 2010.

2. LEE, S.H.; KIM, E.J.; PARK, J.H. Computational Simulations in Physics Education: Effects on Conceptual Learning. *Physics Education Research Journal*, Amsterdam, v. 12, n. 2, p. 115–124, 2011.

3. RUNGE, C.; KUTTA, M. Über die numerische Auflösung gewöhnlicher Differentialgleichungen. *Mathematische Annalen*, Berlin, v. 46, n. 2, p. 167–178, 1901.

4. NUMPY COMMUNITY. NumPy User Guide. [online] Disponível em: <https://numpy.org/doc/>. Acesso em: 10 jul. 2025.

5. MATPLOTLIB DEVELOPMENT TEAM. Matplotlib: Visualization with Python. [online] Disponível em: <https://matplotlib.org/>. Acesso em: 10 jul. 2025.