Revista Brasileira de Ensino de Física, vol. 48, e20250447 (2026)
www.scielo.br/rbef
DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

Artigos Gerais

Uma abordagem didática para a resolução numérica da equação de Schrödinger com Python no ensino de mecânica quântica

A didatical approach to the numerical solution of the Schrödinger equation using Python in quantum mechanics education

Camilla Balliana[1], José Arthur Luna Oliveira[1], Natália Pereira dos Santos[1], Bruno R. Carvalho*[1]

[1]Universidade Federal do Rio Grande do Norte, Departamento de Física, 59078-970, Natal, RN, Brasil.

Recebido em 11 de outubro de 2025. Revisado em 02 de março de 2026. Aceito em 29 de março de 2026.

Neste trabalho, exploramos abordagens computacionais voltadas ao ensino de mecânica quântica por meio da resolução numérica da equação de Schrödinger unidimensional. Analisamos dois sistemas de referência: o poço de potencial finito, tratado por análise gráfica e pelo método de diferenças centradas, e o oscilador harmônico, resolvido pelo método de Numerov. Em ambos os casos, utilizamos a linguagem Python para implementar os algoritmos e gerar visualizações que favorecem a interpretação física dos resultados. A proposta busca estimular a aprendizagem ativa, aproximando teoria, programação e experimentação virtual, além de incentivar o desenvolvimento da proficiência computacional dos estudantes. Dessa forma, contribui-se para uma compreensão mais intuitiva, crítica e engajadora da mecânica quântica no contexto da formação em física moderna.
Palavras-chave: Equação de Schrödinger, Programação em Python, Métodos numéricos, Ensino de mecânica quântica.

In this work, we explore computational approaches aimed at teaching quantum mechanics through the numerical solution of the one-dimensional Schrödinger equation. Two benchmark systems are analyzed: the finite potential well, addressed via graphical analysis and the finite-difference method, and the harmonic oscillator, solved using the Numerov method. In both cases, Python is employed to implement the algorithms and generate visualizations that enhance the physical interpretation of the results. The approach is designed to foster active learning by bridging theory, programming, and virtual experimentation, while also promoting students' computational proficiency. In doing so, it contributes to a more intuitive, critical, and engaging understanding of quantum mechanics in the context of modern physics education.
Keywords: Schrödinger equation, Python programming, Numerical methods, Quantum mechanics education.

1. Introdução

Ensinar mecânica quântica impõe desafios pedagógicos: a abstração conceitual (superposição, tunelamento) e a algebraização do conteúdo podem distanciar os estudantes da intuição física. No contexto da graduação em Física, em que predomina a ênfase nas soluções analíticas, torna-se desejável incorporar abordagens complementares que aproximem a formulação matemática da visualização de fenômenos e da experimentação computacional.

Ferramentas como o Python, aliado a bibliotecas amplamente acessíveis para cálculo numérico e visualização científica (por exemplo, `NumPy` e `Matplotlib`), oferecem um ambiente de baixo custo de entrada para simulações reprodutíveis em cadernos Jupyter. Embora a literatura nacional venha se expandindo e demonstrando um crescente interesse na área [1–5], permanece o desafio de produzir materiais em língua portuguesa que articulem, de modo sistemático e integrado, a modelagem computacional ao fortalecimento da intuição física em mecânica quântica. Ressaltamos que este trabalho é fruto de uma iniciativa de aprendizagem ativa desenvolvida por estudantes de graduação. Assim, o artigo apresenta-se como um estudo de caso de como a integração entre teoria e programação pode fortalecer o protagonismo discente no ensino de Física Moderna.

A literatura brasileira tem apresentado abordagens valiosas para a solução numérica da equação de Schrödinger, detalhando métodos como Numerov, técnicas de tentativa (*shooting methods*) e até Monte Carlo [4–7]. Apesar da disponibilidade dessas técnicas, nota-se a importância de materiais que construam explicitamente a ponte entre as soluções de livros-texto e a modelagem computacional. A contribuição deste trabalho reside na estruturação de uma comparação direta e sistemática entre três paradigmas: a solução semi-analítica (equações transcendentais), a discretização por diferenças centradas e o método de Numerov. Ao

*Endereço de correspondência: brunorc@fisica.ufrn.br
Editor-Chefe: Marcello Ferreira

integrar essas abordagens para o poço finito e o oscilador harmônico, oferecemos ao estudante uma análise crítica sobre convergência, precisão e limitações de cada técnica, preenchendo uma lacuna entre a teoria algébrica e a aplicação computacional.

Nesse cenário, a física computacional assume um papel estratégico no ensino, consolidando-se como uma ferramenta cognitiva que permite transitar entre o rigor da teoria e a visualização fenomenológica. Ao atuar como um laboratório virtual, essa abordagem permite que o estudante manipule parâmetros e observe as consequências físicas nas funções de onda e nas densidades de probabilidade. Essa interatividade reduz a sobrecarga cognitiva associada à resolução algébrica exaustiva, deslocando o foco para a análise crítica e para o desenvolvimento da intuição física [1–3].

Neste trabalho, apresentamos uma sequência de atividades computacionais para dois problemas canônicos em uma dimensão (1D): o poço de potencial quadrado finito e o oscilador harmônico. Para o primeiro, combinamos análise gráfica de equações transcendentais com uma discretização por diferenças centradas – abordagem que converte a equação diferencial em um problema matricial de autovalores –; para o segundo, empregamos o método de Numerov, uma técnica de integração numérica de alta precisão otimizada para equações diferenciais de segunda ordem. Em ambos os casos, confrontamos resultados numéricos com as soluções analíticas, discutindo ganhos pedagógicos e limitações. Os códigos, comentados e prontos para execução, são disponibilizados em português, com o objetivo de ampliar o acesso a recursos didáticos e favorecer práticas de ensino ativas e baseadas em investigação. Além disso, com o objetivo de facilitar o acesso e garantir a reprodutibilidade dos resultados, o código-fonte integral utilizado neste trabalho, foi disponibilizado em um repositório público no GitHub [8]. Essa abordagem dialoga com a literatura recente, que aponta a modelagem computacional como ferramenta essencial para o desenvolvimento da intuição física e a consolidação de modelos mentais em mecânica quântica [1–3].

O trabalho está organizado da seguinte forma: na Seção 2, estabelecemos a fundamentação teórica pertinente. A Seção 3 detalha a metodologia numérica, descrevendo a discretização por diferenças centradas e o método de Numerov. Na Seção 4, aplicamos essas técnicas ao problema do poço de potencial finito, confrontando a solução da matriz do Hamiltoniano com a análise gráfica das equações transcendentais. A Seção 5 dedica-se ao oscilador harmônico quântico, focando na implementação computacional via integração numérica. Por fim, na Seção 6, apresentamos as considerações finais e perspectivas para o uso didático do material.

2. Abordagem Teórica

A equação de Schrödinger constitui a base formal da mecânica quântica não relativística, descrevendo a evolução de um estado quântico no espaço e no tempo. Em sua forma geral, dependente do tempo, ela é expressa por [9]:

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t), \tag{1}$$

onde $\Psi(\mathbf{r},t) = \langle \mathbf{r}|\Psi(t)\rangle$ representa a função de onda do sistema, e $\hat{H}$ o operador Hamiltoniano, que contém a informação sobre a energia total do sistema.

Para isso, restringimo-nos ao estudo de estados estacionários, nos quais a distribuição de probabilidade é independente do tempo. Nesse caso, a equação de Schrödinger assume a forma independente do tempo:

$$\hat{H}\psi(\mathbf{r}) = E\psi(\mathbf{r}), \tag{2}$$

em que E é a energia associada ao estado estacionário descrito por $\psi(\mathbf{r})$.

Nosso objetivo é apresentar uma abordagem computacional para sistemas unidimensionais frequentemente discutidos em disciplinas introdutórias de mecânica quântica. Cada sistema possui um Hamiltoniano específico, o que leva a diferentes formas da equação de Schrödinger. Nas seções seguintes, descrevemos a formulação e a resolução desses modelos, destacando a aplicação de métodos numéricos e seu potencial pedagógico.

3. Metodologia

A resolução da equação (2) consiste em determinar os autovalores de energia e as respectivas autofunções que satisfazem a equação sob condições de contorno apropriadas do sistema físico em estudo. Os autovalores discretos correspondem aos níveis de energia quantizados, enquanto as autofunções associadas descrevem os estados estacionários nos quais a partícula se encontra espacialmente confinada.

Para analisar os dois sistemas propostos, adotamos uma abordagem metodológica que combina a solução analítica com o cálculo numérico. Para o poço de potencial finito, restringindo a análise aos estados ligados ($E < V_0$), empregamos uma metodologia dupla: ($i.$) combinando a análise gráfica das equações transcendentais obtidas da solução analítica, e ($ii.$) a solução numérica por meio do método de diferenças finitas centradas. Essa combinação permite uma comparação direta e pedagogicamente instrutiva entre a previsão teórica e o resultado computacional.

No caso do oscilador harmônico, empregamos o método de Numerov, uma técnica numérica reconhecida por sua eficiência na solução de equações diferenciais de segunda ordem sem o termo da primeira derivada [4, 10, 11]. A equação de Schrödinger para esse sistema é reorganizada de forma a atender os requisitos estruturais do algoritmo, que fornece aproximações de alta precisão para as autofunções a partir de estimativas iniciais para os autovalores de energia. Esses valores são, então, refinados iterativamente, de modo a obter soluções numéricas que satisfaçam as condições de contorno físicas do problema.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

Tabela 1: Principais bibliotecas Python utilizadas nas simulações.

Biblioteca	Função principal
`NumPy`	Computação numérica, manipulação de arrays e operações matriciais
`Matplotlib`	Visualização gráfica e plotagem de funções
`SciPy.sparse`	Criação e manipulação de matrizes esparsas
`SciPy.sparse.linalg`	Resolução de problemas de autovalores e autovetores
`warnings`	Controle e supressão de alertas durante a execução

As soluções numéricas foram implementadas em linguagem de programação Python no ambiente Jupyter Notebook (distribuição Anaconda) [12]. A escolha dessa ferramenta justifica-se por seu valor didático em atividades de aprendizagem ativa, permitindo integrar código e texto formatado (Markdown/LaTeX). Essa estrutura facilita a organização do pensamento físico do estudante, permitindo que a descrição do método e a discussão dos resultados ocorram em um ambiente único e interativo. A escolha do Python deve-se à sua flexibilidade, clareza sintática e ao amplo ecossistema de bibliotecas científicas disponíveis. Em particular, utilizamos as bibliotecas listadas na Tabela 1: `NumPy` para computação numérica, `Matplotlib` para visualização gráfica e `SciPy`, notadamente os submódulos `sparse` e `linalg`, para manipulação de matrizes esparsas e resolução de problemas de autovalores. Adicionalmente, a biblioteca `warnings` foi empregada para controle de alertas durante a execução do código. Essa combinação de ferramentas permitiu a implementação eficiente e didática das simulações de problemas quânticos unidimensionais.

Vale ressaltar que a abordagem numérica aqui desenhada foca na determinação de estados ligados, caracterizados por um espectro de energia discreto. Isso permite a visualização da penetração da função de onda nas regiões classicamente proibidas, onde a densidade de probabilidade decai rapidamente. A análise de estados de espalhamento (espectro contínuo), embora passível de tratamento numérico, foge ao escopo desta proposta didática voltada para os sistemas confinados.

Os códigos desenvolvidos são apresentados ao longo do artigo, integrados à discussão dos conceitos físicos e matemáticos. Esta estratégia busca favorecer uma aprendizagem contínua, aproximando a teoria da implementação computacional. Acreditamos que essa abordagem torna o aprendizado mais acessível e intuitivo, facilitando a compreensão dos fenômenos quânticos por meio de recursos visuais e dinâmicos, o que permite sua livre utilização, reprodução e adaptação por educadores e estudantes.

3.1. Solução numérica com o método de diferenças centradas

Embora as soluções analíticas sejam fundamentais para compreensão conceitual dos sistemas quânticos, em muitos casos elas exigem manipulações algébricas complexas ou simplesmente não estão disponíveis. Nesses cenários, os métodos numéricos surgem como ferramentas eficazes para determinar níveis de energia e funções de onda de forma direta. Destacam-se desde a discretização por diferenças finitas [2] até métodos voltados a sistemas complexos, como o de Hartree-Fock [1], permitindo uma visualização concreta dos resultados e a superação de limitações analíticas.

Entre esses métodos, adotamos o método de diferenças centradas, que consiste em discretizar a equação de Schrödinger independente do tempo sobre uma malha de pontos igualmente espaçados por meio da expansão em série de Taylor, aproximando a derivada de segunda ordem ($\psi''(x)$) através dos pontos vizinhos x_{i-1} e x_{i+1}. No Python, isso permite transformar a equação diferencial de Schrödinger em um problema de autovalores matriciais. A eficiência dessa técnica reside na obtenção de uma matriz tridiagonal simétrica. Nessa representação, os elementos da diagonal contêm a energia potencial e as diagonais adjacentes contêm o termo da energia cinética [6, 13]. Do ponto de vista didático, essa tradução é fundamental, pois materializa o conceito abstrato de operador Hamiltoniano, permitindo que o aluno identifique a energia como um autovalor e a função de onda como um autovetor do sistema matricial. A derivada de segunda ordem é aproximada por uma expressão finita envolvendo apenas os pontos vizinhos, de modo que o problema contínuo é convertido em um problema de autovalores matricial. Essa transformação é particularmente eficiente porque leva a uma matriz tridiagonal simétrica, cuja diagonal contém o potencial e cujas sub- e super-diagonais refletem o termo cinético [2, 14].

Do ponto de vista formal, a aproximação para a derivada de segunda ordem é a base do método, e considera os pontos imediatamente vizinhos x_{i-1} e x_{i+1}, igualmente espaçados por Δx:

$$\psi''(x_i) = \frac{\mathrm{d}^2\psi(x_i)}{\mathrm{d}x^2} \simeq \frac{\psi(x_{i+1}) - 2\psi(x_i) + \psi(x_{i-1})}{\Delta x^2}, \quad (3)$$

Substituindo essa expressão na equação de Schrödinger, obtemos uma equação matricial da forma $H\vec{\psi} = E\vec{\psi}$, em que H é a matriz do Hamiltoniano. Essa matriz é construída a partir da soma de duas contribuições: (i) a matriz cinética tridiagonal ($\hat{T}$), que possui coeficientes -2 em sua diagonal principal e 1 nas subdiagonais adjacentes (decorrentes da discretização da derivada segunda), e (ii) a matriz potencial diagonal ($\hat{U}$), que contém os valores discretizados de $V(x)$. Dessa forma, cada elemento da diagonal principal do Hamiltoniano final é o resultado da soma do termo cinético (coeficiente -2) com o valor local do potencial naquela posição da grade.

Multiplicando (i) pelo vetor coluna com os valores da função de onda discretizada, temos:

$$\hat{T}\vec{\psi} = E\vec{\psi} \tag{4}$$

$$\frac{-\hbar^2}{2m}\cdot\frac{1}{\Delta x^2}\begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & 1 & -2 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}\begin{pmatrix} \psi(x_1) \\ \psi(x_2) \\ \vdots \\ \vdots \\ \psi(x_{N-1}) \end{pmatrix} = E\begin{pmatrix} \psi(x_1) \\ \psi(x_2) \\ \vdots \\ \vdots \\ \psi(x_{N-1}) \end{pmatrix}.$$

Para a implementação numérica do poço finito, definimos um domínio de cálculo $[-L, L]$ tal que $L > a$. Diferente do poço infinito, a função de onda no poço finito apresenta uma penetração na região classicamente proibida ($|x| > a$), onde decai rapidamente. Portanto, as condições de contorno numéricas impostas são $\psi(-L) = \psi(L) = 0$. Fisicamente, a escolha de L deve garantir que a extensão espacial do estado ligado esteja contida no intervalo, assegurando que a amplitude da função de onda nas bordas da malha seja desprezível. Este cuidado é crucial para os estados com maior número quântico n, que possuem maior penetração na barreira. Por exemplo, no ponto extremo x_1 a aproximação é:

$$\psi''(x_{N-1}) \simeq \psi(x_{N-2}) - 2\psi(x_{N-1}) + \psi(x_N), \tag{5}$$

com $\psi(x_N) = 0$. O mesmo se aplica ao ponto x_0.

Para o potencial finito, a matriz que representa o operador associado à energia potencial é diagonal:

$$\hat{U} \approx \begin{pmatrix} V(x_1) & 0 & 0 & \dots & 0 \\ 0 & V(x_2) & 0 & \dots & 0 \\ 0 & 0 & V(x_3) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & V(x_{N-1}) \end{pmatrix}$$

Somando os operadores de energia cinética e potencial, obtemos o Hamiltoniano total:

$$\hat{H}\psi(x) = [\hat{T} + \hat{U}]\psi(x) = E\psi(x). \tag{6}$$

Assim, no espaço discreto, a função de onda $\psi(x)$ é representada por um vetor coluna com os valores $\psi(x_i)$, e o Hamiltoniano do sistema assume a forma:

$$\frac{-\hbar^2}{2m}\cdot\frac{1}{\Delta x^2}\begin{pmatrix} V(x_1)-2 & 1 & 0 & \dots & 0 \\ 1 & V(x_2)-2 & 1 & \dots & 0 \\ 0 & 1 & V(x_3)-2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & V(x_{N-1})-2 \end{pmatrix}$$

$$\begin{pmatrix} \psi(x_1) \\ \psi(x_2) \\ \vdots \\ \vdots \\ \psi(x_{N-1}) \end{pmatrix} = E\begin{pmatrix} \psi(x_1) \\ \psi(x_2) \\ \vdots \\ \vdots \\ \psi(x_{N-1}) \end{pmatrix}$$

Nesse método numérico, a malha é composta por N pontos sendo que os valores nas extremidades são fixados em zero pelas condições de contorno. A matriz do Hamiltoniano é construída para atuar sobre os pontos internos da malha ($i = 1, \dots, N-2$), cujos valores da função de onda constituem as incógnitas do problema de autovalor matricial. Diferentemente do poço infinito, onde as fronteiras da malha coincidem com as paredes físicas do poço, no caso do poço finito é necessário definir um domínio computacional significativamente maior que a largura do poço ($L > a$). Isso permite capturar o decaimento exponencial da função de onda na região classicamente proibida (onde $V(x) = V_0$), garantindo que a solução numérica tenda a zero de forma suave antes de atingir as bordas da malha.

Do ponto de vista didático, o método de diferenças centradas é de grande valor. Ele conecta de maneira direta a álgebra matricial à mecânica quântica, reinterpretando a equação de Schrödinger como um problema de autovalores linear, algo já familiar aos estudantes de álgebra linear. Além disso, sua implementação em Python é imediata com o uso de bibliotecas como `NumPy` e `SciPy`, que permitem resolver sistemas de autovalores de forma eficiente e extrair tanto energias quanto autofunções com poucas linhas de código. Finalmente, o método favorece uma visualização pedagógica clara. Ao fazer um gráfico dos autovalores e as funções de onda resultantes, torna-se possível comparar diretamente os resultados numéricos com as previsões gráficas obtidas pela análise transcendental da subseção anterior.

Entre as limitações deste método, destaca-se o erro de truncamento de ordem $O(\Delta x^2)$. Como consequência, a obtenção de alta precisão requer o uso de malhas espaciais mais densas, o que eleva o custo computacional devido ao aumento na dimensão da matriz Hamiltoniana. No entanto, tal fator torna-se praticamente irrelevante em problemas unidimensionais, nos quais os recursos computacionais modernos processam essas matrizes de forma quase instantânea.

Assim, o método de diferenças centradas não apenas fornece uma solução numérica robusta para o poço de potencial finito, mas também reforça a integração entre formalismo teórico, ferramentas computacionais e visualização física, consolidando seu papel como recurso didático no ensino de mecânica quântica.

3.2. Método de Numerov

Para resolver a equação de Schrödinger do oscilador harmônico, utilizamos o método de Numerov [4, 10, 11], uma técnica especializada para equações diferenciais

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

lineares de segunda ordem que não possui o termo da primeira derivada. Este método utiliza uma relação de recorrência de três pontos para propagar a função de onda ao longo da malha. A equação diferencial pode ser escrita na forma:

$$\frac{d^2\psi(x)}{dx^2} + k^2(x)\psi(x) = 0, \tag{7}$$

onde definimos:

$$k^2(x) = \frac{2m}{\hbar^2}\left(E - \frac{1}{2}m\omega^2 x^2\right). \tag{8}$$

Essa forma é adequada para o algoritmo de Numerov, cuja ideia central é calcular a função de onda em um ponto a partir de seus dois vizinhos, usando a relação de recorrência:

$$\psi_{i+1} = \frac{2\left(1 - \frac{5\delta^2}{12}k_i^2\right)\psi_i - \left(1 + \frac{\delta^2}{12}k_{i-1}^2\right)\psi_{i-1}}{1 + \frac{\delta^2}{12}k_{i+1}^2}, \tag{9}$$

Aqui, δ é o espaçamento entre os pontos da malha, e os índices $i-1$, i e $i+1$ se referem a pontos consecutivos do domínio espacial. O método apresenta erro de ordem $O(\delta^6)$, o que garante alta precisão mesmo em malhas relativamente pouco densas.

O procedimento numérico segue as seguintes etapas:

1. Define-se um domínio espacial simétrico $[-L, L]$ e condições iniciais para a função de onda, $\psi(x_{\min}) = 0$ e um pequeno valor arbitrário em $x_{\min} + \delta$.
2. Escolhe-se um valor de teste para a energia E.
3. Propaga-se a função de onda em toda a malha usando a fórmula de Numerov.
4. Verifica-se se a solução obtida é fisicamente admissível, isto é, se permanece finita e se anula adequadamente nas extremidades do domínio.
5. Ajusta-se o valor de E até encontrar os autovalores corretos, normalizando a função de onda correspondente.

O uso do método de Numerov para o oscilador harmônico tem grande valor didático. Em primeiro lugar, este método mostra como um problema que possui solução analítica conhecida pode ser reproduzido com excelente aproximação numérica, validando a técnica antes de aplicá-la a sistemas mais complexos. Conecta conceitos de álgebra numérica com a mecânica quântica, ao formular a equação de Schrödinger como um problema de autovalores estruturado por meio de relações de recorrência. Mostra a importância de condições de contorno e da normalização numérica, elementos centrais na interpretação física. Favorece ainda a visualização gráfica das autofunções e da distribuição de probabilidade, reforçando o entendimento de quantização e do papel do potencial harmônico.

Assim, a análise do oscilador harmônico pelo método de Numerov não apenas fornece resultados quantitativos consistentes, mas também constitui uma ponte entre formalismo matemático, ferramentas computacionais e intuição física, consolidando sua relevância como recurso didático no ensino de mecânica quântica.

Por possuir erro de ordem $O(\Delta x^6)$, ele atinge uma alta precisão mesmo com poucos pontos na malha. No caso do oscilador harmônico, isso permite ao aluno verificar que os níveis de energia são exatamente igualmente espaçados ($E_n = \hbar\omega(n + \frac{1}{2})$), reforçando a confiabilidade desse método numérico. O método de Numerov exige que o aluno faça uma busca pelo autovalor correto, simulando um processo de descoberta. Por consequência, é um método de tiro (*shooting method*), se a energia inicial for ligeiramente incorreta, a função de onda diverge no infinito. Isso exige o uso de técnicas de refinamento como a bisseção [7].

Vale salientar que o método de Numerov é preferível ao método de diferenças centradas na resolução do oscilador harmônico quântico por apresentar maior precisão e estabilidade numérica, uma vez que é um método de ordem superior, enquanto as diferenças centradas são, em geral, de segunda ordem. Além disso, o método de Numerov é naturalmente adaptado à forma da equação de Schrödinger estacionária, permitindo uma implementação direta por meio de uma relação recursiva simples, sem a necessidade de formular e diagonalizar grandes matrizes, o que resulta em uma obtenção mais precisa dos autovalores de energia com menor custo computacional. Ainda assim, optou-se por apresentar ambos os métodos ao longo do trabalho para refletir a rica diversidade de abordagens explorada durante uma atividade de aprendizagem ativa. Nesse contexto, diferentes estudantes implementaram estratégias independentes para problemas específicos, o que permitiu uma análise comparativa de diferentes ferramentas computacionais. Dessa forma, a utilização de mais de um método não visa estabelecer uma superioridade técnica absoluta, mas sim ilustrar a versatilidade das estratégias numéricas e promover a compreensão de suas características, limitações e potencial pedagógico no ensino de física.

4. Poço de Potencial Finito

O poço de potencial finito unidimensional é um problema canônico e instrutivo no ensino de mecânica quântica, pois introduz conceitos fundamentais como estados ligados, tunelamento quântico e simetrias das funções de onda. Ao contrário do poço infinito, este sistema não possui soluções analíticas fechadas para os níveis de energia, o que motiva o uso de métodos numéricos e o torna uma excelente oportunidade pedagógica para integrar aspectos conceituais e computacionais.

Consideramos uma partícula em um potencial atrativo simétrico de profundidade finita, definido por:

$$\begin{cases} V(x) = V_0, & |x| \geq a \\ V(x) = 0, & |x| < a, \end{cases} \tag{10}$$

onde $V_0 > 0$ caracteriza a profundidade do poço e, $2a$ sua largura total, centrado na origem [13]. A equação (2) assume a forma:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + [V(x) - E]\,\psi(x) = 0, \qquad (11)$$

com E a energia e $\psi(x)$ a função de onda da partícula. O caráter não trivial do problema decorre do fato de que apenas certos valores discretos de energia $E < V_0$ satisfazem as condições de contorno impostas pela continuidade de $\psi(x)$ e de sua derivada $\psi'(x)$.

Como o potencial $V(x)$ é definido por partes, resolvemos a equação em três regiões: I) $x < -a$, II) $-a < x < a$, e III) $x > a$. Nas regiões I e III, onde $V(x) = V_0$ é constante, obtemos:

$$\frac{d^2\psi(x)}{dx^2} = \kappa^2\psi(x), \quad \text{com} \quad \kappa = \sqrt{\frac{2m(V_0 - E)}{\hbar^2}}, \qquad (12)$$

cujas soluções fisicamente aceitáveis (decrescentes no infinito) são exponenciais:

$$\psi_I(x) = Ae^{\kappa x}, \quad \psi_{III}(x) = Fe^{-\kappa x}. \qquad (13)$$

Na região II, $V(x) = 0$,

$$\frac{d^2}{dx^2}\psi(x) = -k^2\psi(x), \quad \text{com} \quad k = \sqrt{\frac{2mE}{\hbar^2}}, \qquad (14)$$

de modo que, para $E < V_0$,

$$\psi_{II}(x) = C\cos(kx) + D\sin(kx). \qquad (15)$$

Com a simetria do potencial ($V(x) = V(-x)$), as soluções podem ser escolhidas com paridade definida. Para soluções pares, $D = 0$ e $\psi_{II}(x) = C\cos(kx)$; para soluções ímpares, $C = 0$ e $\psi_{II}(x) = D\sin(kx)$. Essa simetria reduz a complexidade do problema, permitindo resolver apenas para $x > 0$, bastando impor as condições de continuidade de $\psi(x)$ e de sua derivada $\psi'(x)$ em $x = \pm a$, o que leva às equações transcendentais, que relacionam k e κ,

$$\kappa = k\tan(ka), \quad \text{(estados pares)} \qquad (16)$$

$$\kappa = -k\cot(ka), \quad \text{(estados ímpares)} \qquad (17)$$

Estas relações determinam, implicitamente, os valores permitidos de k (e, portanto, de E) e não admitem solução analítica fechada. Nas próximas subseções, exploramos (i) uma análise gráfica dessas equações e (ii) uma solução numérica por diferenças finitas centradas, permitindo comparar, de forma didática, as previsões teóricas com resultados computacionais.

4.1. Análise gráfica das equações transcendentais

Essa abordagem é particularmente instrutiva. Do ponto de vista teórico, revela a dependência direta entre profundidade do poço e número de estados ligados; do ponto de vista computacional, permite implementar em Python um procedimento simples de busca de interseções, reforçando a ponte entre formalismo matemático e visualização gráfica.

Para determinar os níveis de energia permitidos no poço de potencial finito, implementamos esse procedimento numérico em Python a partir da análise gráfica das equações transcendentais. A ideia consiste em representar as equações graficamente e identificar suas raízes a partir das interseções, dentro de uma tolerância numérica apropriada.

As condições de contorno podem ser reescritas em termos das variáveis adimensionais $u = \kappa a$ e $v = ka$, que satisfazem a equação do círculo modificada:

$$u^2 + v^2 = z_0^2, \qquad \text{com} \quad z_0 = a\sqrt{\frac{2mV_0}{\hbar^2}}. \qquad (18)$$

Essa forma evidencia que os estados ligados correspondem às interseções entre a curva do círculo e as curvas das soluções pares e ímpares. O raio z_0 fornece ainda uma estimativa do número máximo de estados ligados para dado poço. O primeiro passo é definir os parâmetros físicos do sistema e a malha de energias a serem testadas:

Código 1: Definição dos parâmetros físicos e da malha de energias para o poço de potencial finito.

```
import numpy as np
import matplotlib.pyplot as plt

# Parâmetros do sistema (unidades arbitrárias)
a = 2.0        # Semi-largura do poço
V0 = 12.0      # Profundidade do poço
hbar = 1.0     # Constante de Planck reduzida
m = 1.0        # Massa da partícula

# Varredura de energias (restrita a E < V0)
E_varredura = np.linspace(0.01, V0 * 0.99,
    10000)
```

Em seguida, calculamos as constantes k e κ, as variáveis adimensionais u e v, bem como a curva do círculo. As equações transcendentais para as soluções pares e ímpares envolvem funções trigonométricas com assíntotas (como $\tan v$), o que pode causar divisões por zero e gerar o aviso `RuntimeWarning: invalid value encountered in true_divide`. Para contornar esse problema, suprimimos temporariamente o aviso e filtramos valores não finitos (`inf`, `NaN`) antes de detectar as interseções:

Código 2: Definição das constantes adimensionais e tratamento das funções transcendentais.

```
# Constantes k e kappa
k = np.sqrt(2 * m * E_varredura / hbar**2)
kappa = np.sqrt(2 * m * (V0 - E_varredura) /
    hbar**2)

# Variáveis adimensionais
u = kappa * a
v = k * a

# Constante (raio do círculo)
z0 = a * np.sqrt(2 * m * V0 / hbar**2)
```

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

```
circulo = np.sqrt(z0**2 - v**2)

u_max = z0 * 1.5  # Define o eixo x 50% maior que o raio do círculo
u_ = np.linspace(0.01, u_max, 10000)

# Suprime o aviso causado pelas assíntotas da tangente
np.seterr(divide='ignore', invalid='ignore')
funcao_par_ = u_ * np.tan (u_)
funcao_impar_ = -u_ / np.tan (u_)
funcao_par = v * np.tan(v)
funcao_impar = -v / np.tan(v)
np.seterr(divide='warn', invalid='warn')
```

Para localizar as interseções de forma robusta, construímos uma função que filtra valores próximos às assíntotas e procura mudanças de sinal entre pares de pontos consecutivos. Isso evita que descontinuidades sejam interpretadas erroneamente como soluções.

Código 3: Função para detecção segura das interseções entre as equações transcendentais e a curva do círculo.

```
def encontrar_intersecoes(v, f1, f2, limite_valor=2*z0):
    mask_finite = np.isfinite(f1) & np.isfinite(f2)
    # Remove pontos muito grandes (próximos às assíntotas)
    mask_finite &= np.abs(f1) < limite_valor
    mask_finite &= np.abs(f2) < limite_valor

    idx = []
    for i in range(len(v) - 1):
        if mask_finite[i] and mask_finite[i+1]:
            if (f1[i] - f2[i]) * (f1[i+1] - f2[i+1]) < 0:
                idx.append(i)
    return np.array(idx)

# Identificação das interseções
idx_par = encontrar_intersecoes(v, circulo, funcao_par)
idx_impar = encontrar_intersecoes(v, circulo, funcao_impar)
```

Com as interseções localizadas, podemos plotar o resultado. O gráfico exibe a curva do círculo, as funções transcendentais pares e ímpares e os pontos de interseção que correspondem aos autovalores de energia.

Código 4: Construção gráfica das equações transcendentais e identificação dos autovalores de energia.

```
fig, ax = plt.subplots(1, 1, figsize=(8, 5))

# Curva de referência (círculo)
ax.plot(v, circulo, '--', color="black", linewidth=1.5)
ax.plot(v * 1.2, circulo * 1.2, '--', color="gray", linewidth=1.5)
ax.plot(v * 1.4, circulo * 1.4, '--', color="gray", linewidth=1.5)

# Funções transcendentes (scatter com transparência)
ax.scatter(u_, funcao_par_, s=1.5, color='#1f77b4', alpha=0.6, label='Soluções pares')
ax.scatter(u_, funcao_impar_, s=1.5, color='#ff7f0e', alpha=0.6, label='Soluções ímpares')

ax.scatter(v, funcao_par, s=1.5, color='#1f77b4', alpha=0.6, label='Soluções pares')
ax.scatter(v, funcao_impar, s=1.5, color='#ff7f0e', alpha=0.6, label='Soluções ímpares')

# Pontos de interseção
ax.plot(v[idx_par], circulo[idx_par], 'o', ms=7, mfc='#aec7e8', mec='black', label='Interseções Pares')
ax.plot(v[idx_impar], circulo[idx_impar], 'o', ms=7, mfc='#ffbb78', mec='black', label='Interseções Ímpares')

# Configurações do gráfico
ax.set_xlim([0.0, u_max])
ax.set_ylim([0.0, z0+5])
ax.set_ylabel(r'$\kappa a = \sqrt{z_0^2 - (ka)^2}$', fontsize=16)
ax.set_xlabel(r'$ka$', fontsize=16)
ax.grid(True, linestyle=':', alpha=0.3)

# Legenda para funções e interseções
handles, labels = plt.gca().get_legend_handles_labels()
unique_labels, unique_handles = [], []
for i, label in enumerate(labels):
    if label not in unique_labels:
        unique_labels.append(label)
        unique_handles.append(handles[i])
ax.legend(unique_handles, unique_labels, loc='lower left', fontsize=8)

plt.tight_layout()
plt.show()
```

A Figura 1 sintetiza a ponte entre o formalismo e a implementação numérica. No plano adimensional (v, u), em que $v = ka$ representa a região propagante dentro do poço e $u = \kappa a$ a região evanescente nas barreiras, o círculo $u^2 + v^2 = z_0^2$ expressa a relação geométrica entre as duas contribuições. As curvas azul e laranja representam, respectivamente, as soluções pares ($u = v \tan v$) e ímpares ($u = -v \cot v$) impostas pelas condições de contorno. Cada ponto de interseção entre uma dessas curvas e o círculo identifica um estado ligado, permitindo obter k e κ, e portanto a energia $E = \hbar k^2/(2m)$ e a paridade do estado (pela curva que intercepta).

Didaticamente, a Figura 1 evidencia de forma clara o mecanismo de quantização: os estados só existem onde as condições de contorno são simultaneamente satisfeitas (nas interseções). Observa-se na Figura 1, por meio das curvas tracejadas em cinza, que o aumento da profundidade V_0 (ou o alargamento do poço pelo parâmetro a) resulta na expansão do raio z_0. Geometricamente, esse crescimento permite que o círculo intercepte ramos adicionais das funções transcendentais, o que fisicamente se traduz no surgimento de novos estados ligados. Tal comportamento demonstra que a capacidade de confinamento do sistema é proporcional à magnitude do potencial e à sua extensão espacial. Destaca-se, ainda, a alternância entre soluções pares e ímpares com o aumento da energia, antecipando a

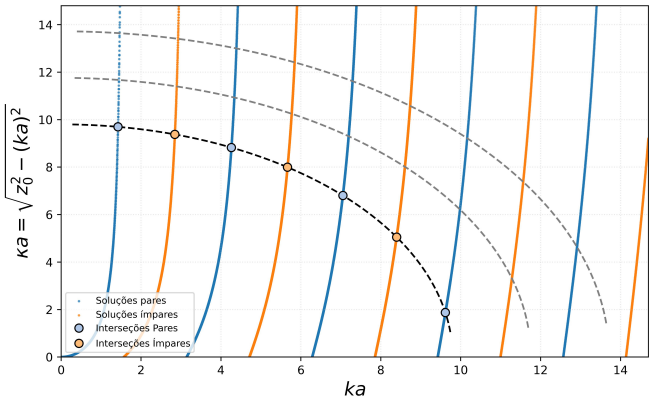

Figura 1: Equações transcendentais e interseções no poço finito. Representação gráfica dos estados ligados em um poço de potencial finito. As curvas azul e laranja correspondem às soluções pares e ímpares, enquanto as tracejadas representam a curva do círculo para diferentes profundidades V_0. Os pontos de interseção destacados indicam os valores de $v = ka$ que satisfazem as condições de contorno, correspondendo aos níveis de energia permitidos. O círculo preto tracejado delimita a região de estados ligados para o potencial padrão adotado. Os círculos em cinza representam sistemas com potenciais mais profundos ou larguras maiores.

contagem de nós das autofunções. Sendo assim, essa experimentação virtual permite que o discente visualize o mecanismo de quantização de forma dinâmica, consolidando a intuição física sobre como a geometria do potencial molda o espectro de energia da partícula antes mesmo do tratamento puramente algébrico.

Do ponto de vista computacional, o gráfico é construído a partir de uma varredura discreta em v, com valores não finitos próximos às assíntotas das funções tangente e cotangente, evitando o aparecimento de falsas raízes. Esse tratamento simples fornece resultados robustos e suficientemente precisos para fins didáticos, ao mesmo tempo em que ilustra como técnicas de programação básicas podem ser aplicadas à resolução de problemas transcendentais. A Figura 1 não apenas confirma as previsões teóricas sobre a relação entre profundidade do poço e número de estados ligados, como também demonstra como uma implementação computacional acessível em Python pode traduzir as condições de contorno em um problema de interseções geométricas, fortalecendo a integração entre teoria, abordagem numérica e interpretação física.

4.2. Implementação computacional do método de diferenças centradas

Para implementar o método, a primeira etapa é a construção da matriz do Hamiltoniano, de acordo com o formalismo de diferenças centradas. Conforme a aproximação da segunda derivada, o operador de energia cinética é representado por uma matriz tridiagonal com os coeficientes `1, -2, 1` nas diagonais, enquanto o potencial é representado por uma matriz diagonal. Para a implementação, utilizamos as bibliotecas `NumPy` para operações matemáticas, `Matplotlib` para visualização, e o módulo `sparse.linalg` do `SciPy`, que oferece solucionadores otimizados para matrizes esparsas.

O código a seguir inicia o processo definindo os parâmetros do sistema e montando a matriz do Hamiltoniano.

Código 5: Construção da matriz do Hamiltoniano via diferenças centradas.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import diags
from scipy.sparse.linalg import eigsh  # solver recomendado para matrizes simétricas
import warnings

# Parâmetros físicos do sistema (adimensionais)
hbar = 1.0
m = 1.0

# Parâmetros da malha espacial
xmin, xmax = -5.0, 5.0
N_pontos = 1002 # N total de pontos, incluindo as bordas
x = np.linspace(xmin, xmax, N_pontos)
dx = x[1] - x[0]

# Parâmetros do potencial
L_poco = 2.0    # largura do poço
V0 = 12.0       # profundidade do poço

# Cria o vetor de potencial apenas para os pontos internos da malha
x_interno = x[1:-1]
V_interno = np.where((x_interno < L_poco) & (x_interno > -L_poco), 0.0, V0)

# Construção do Hamiltoniano (sparse)
dim_H = N_pontos - 2  # dimensão da matriz do Hamiltoniano (N-2 pontos internos)

# Operador de energia cinética no formato matricial
energia_cinetica = -(hbar**2) / (2 * m * dx**2)
diagonal_principal = np.full(dim_H, -2.0)
diagonal_secundaria  = np.full(dim_H - 1, 1.0)
T = diags([diagonal_secundaria, diagonal_principal, diagonal_secundaria], offsets=[-1, 0, 1], shape=(dim_H, dim_H))
T = energia_cinetica * T

# Operador de energia potencial (matriz diagonal)
Vmat = diags([V_interno], [0], shape=(dim_H, dim_H))
# Hamiltoniano total
H = T + Vmat
```

Vale ressaltar que, embora tenhamos optado pela construção explícita da matriz para fins didáticos, a eficiência computacional pode ser significativamente aumentada em problemas de grande escala utilizando-se funções específicas para matrizes tridiagonais, como a `scipy.linalg.eigh_tridiagonal`, que otimizam o uso de memória e o tempo de execução.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

Após a construção do Hamiltoniano, o próximo passo é extrair os autovalores (níveis de energia) e os autovetores (funções de onda). Para isso, utilizamos o solucionador `eigsh`, que é ideal para matrizes esparsas e simétricas. Para garantir a robustez da solução, implementamos um mecanismo de `fallback` que, em caso de falha, recorre ao solucionador denso `numpy.linalg.eigh`.

Uma vez obtidos, os autovalores e autovetores são processados. Primeiro, eles são ordenados em ordem crescente de energia (do estado fundamental para o mais excitado). Em seguida, o código filtra os autovalores para identificar apenas os estados ligados, ou seja, aqueles com energia menor que a profundidade do poço.

Código 6: Extração dos autovalores e autovetores do Hamiltoniano, identificando apenas os estados ligados.

```
# Número de autovalores a pedir (sempre < dim_H)
k_estados = 6
k_estados = max(1, min(k_estados, dim_H - 2))  # segurança

# Tentativa principal: eigsh (simétrico)
try:
    with warnings.catch_warnings():
        warnings.simplefilter("ignore")  # suprimir warnings numéricos momentâneos
        todos_autovalores, autovetores_interno = eigsh(H, k=k_estados, which='SA')  # 'SA' = smallest algebraic
    todos_autovalores = todos_autovalores.real
    autovetores_interno = autovetores_interno.real
    soluc_usado = "eigsh (ARPACK)"
except Exception as e:
    # fallback robusto: solução densa
    H_densa = H.toarray()
    valores, vetores = np.linalg.eigh(H_densa)
    todos_autovalores = valores[:k_estados].real
    autovetores_interno = vetores[:, :k_estados].real
    soluc_usado = "numpy.linalg.eigh (fallback)"

# Ordenamento consistente dos autovalores e vetores
ordem = np.argsort(todos_autovalores)
todos_autovalores = todos_autovalores[ordem]
autovetores_interno = autovetores_interno[:, ordem]

# Selecionar estados ligados E < V0 (com margem numérica)
tol = 1e-8
ligacao_idx = np.where(todos_autovalores < V0 - tol)[0]
E_ligado = todos_autovalores[ligacao_idx]
autovetores_ligado_interno = autovetores_interno[:, ligacao_idx]

num_ligado = len(E_ligado)

if num_ligado == 0:
    raise RuntimeError("Nenhum estado ligado encontrado. Tente aumentar k_states, V0 ou L_poco.")

# Reconstruir vetores no domínio completo (com bordas 0)
autovetores_completo = np.zeros((N_pontos, num_ligado))
autovetores_completo[1:-1, :] = autovetores_ligado_interno
```

Finalmente, as funções de onda são normalizadas e têm seu sinal padronizado. Os autovetores são estendidos para o domínio completo, e a integral do módulo quadrado de cada função de onda é calculada numericamente usando a regra do trapézio (`np.trapz`) por sua simplicidade conceitual e precisão adequada para o problema, garantindo que a normalização final seja unitária. Para fins de consistência visual na apresentação gráfica, adotou-se uma convenção de sinal na qual, se a amplitude de pico for negativa, a função é multiplicada por -1. Vale ressaltar que essa operação não altera o estado físico do sistema, uma vez que a densidade de probabilidade $|\psi|^2$, esta sim com significado físico, é invariante sob uma mudança de fase global.

Código 7: Normalização das autofunções e preparação do potencial para visualização.

```
# Normalização segura usando x
def normalizacao(psi, xvet):
    """ Normaliza um vetor de função de onda usando a regra do trapézio """
    norm = np.sqrt(np.trapz(np.abs(psi)**2, x=xvet))
    if norm == 0:
        return psi
    return psi / norm

for i in range(num_ligado):
    # Normaliza primeiro
    autovetores_completo[:, i] = normalizacao(autovetores_completo[:, i], x)

    #Correção de fase para fixar concavidade
    """Encontra o valor absoluto máximo da função de onda"""
    max_amp = np.max(np.abs(autovetores_completo[:, i]))

    """Encontra o índice correspondente a este valor máximo"""
    max_idx = np.where(np.abs(autovetores_completo[:, i]) == max_amp)[0][0]

    """Se o valor nesse ponto for negativo, inverte a função de onda inteira"""
    if autovetores_completo[max_idx, i] < 0:
        autovetores_completo[:, i] *= -1
```

Por fim, as funções de onda e as densidades de probabilidade são plotadas, incluindo anotações visuais para uma interpretação mais clara dos resultados.

Código 8: Plotagem das funções de onda e densidades de probabilidade normalizadas em comparação com o potencial.

```
# Preparando para plot
V_plot = np.where((x < L_poco) & (x > -L_poco), 0.0, V0)

# Construção gráfica
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,4))
```

```
6
7  # Colormap para diferenciar cada estado
8  cores = plt.cm.viridis(np.linspace(0, 1, len(
       E_ligado)))
9
10 ### Painel (a): Funções de onda ###
11
12 # Desenhando o potencial
13 ax1.plot(x, V_plot, color='tab:gray', linestyle
       ='--', linewidth=1.2, label=r'$V(x)$')
14
15 for i, E in enumerate(E_ligado):
16     """Encontra a amplitude máxima da curva"""
17     max_amp = np.max(np.abs(
       autovetores_completo[:, i]))
18     """Define um fator de escala (ajuste se
       necessário)"""
19     fator_escala = 0.05 * V0 / max_amp
20
21     ax1.plot(x, E + fator_escala *
       autovetores_completo[:, i], color=cores[i],
        linewidth=1.4,
22              label=rf"$n={i}$")
23     ax1.hlines(E, xmin, xmax, linestyle=':',
       color='gray', linewidth=0.8)
24
25 ### Painel (b): Densidades de probabilidade ###
26
27 # Desenhando o potencial
28 ax2.plot(x, V_plot, color='tab:gray', linestyle
       ='--', linewidth=1.2, label=r'$V(x)$')
29
30 for i, E in enumerate(E_ligado):
31     """Encontra a amplitude máxima da curva"""
32     max_amp_prob = np.max(np.abs(
       autovetores_completo[:, i])**2)
33     """Define um fator de escala (ajuste se
       necessário)"""
34     fator_escala_prob = 0.1 * V0 / max_amp_prob
35
36     ax2.plot(x, E + fator_escala_prob * np.abs(
       autovetores_completo[:, i])**2, color=cores
       [i], linewidth=1.4,
37              label=rf"$n={i}$")
38     ax2.hlines(E, xmin, xmax, linestyle=':',
       color='gray', linewidth=0.8)
39
40     ax2.text(xmax + 0.1, E + 0.02 * V0, rf"$n={
       i}$", fontsize=8, color=cores[i])
41
42 ### Ajustes finais ###
43 for ax, label in zip([ax1, ax2], ['(a)', '(b)'
       ]):
44     ax.vlines([-L_poco, L_poco], 0, V0, color='
       gray', linestyle='--', linewidth=1.0)
45     ax.set_xlim(xmin, xmax)
46     ax.set_ylim(min(E_ligado) - 0.05*V0, 1.05*
       V0)
47     ax.grid(True, linestyle=':', alpha=0.4)
48     ax.text(-0.12, 1.05, label, transform=ax.
       transAxes, fontsize=12, va='top', ha='left'
       )
49
50 ax1.set_xlabel(r"distância $(\mathrm{un.\ arb
       .})$", fontsize=12)
51 ax2.set_xlabel(r"distância $(\mathrm{un.\ arb
       .})$", fontsize=12)
52 ax1.set_ylabel(r" $\psi_n(x)$", fontsize=12)
53 ax2.set_ylabel(r"$|\psi_n(x)|^2$", fontsize=12)
54 plt.tight_layout()
55 plt.show()
```

As visualizações obtidas com a solução numérica fornecem uma interpretação direta dos fenômenos quânticos. A Figura 2a, que mostra as funções de onda $\psi_n(x)$, e a Figura 2b, que exibe as densidades de probabilidade correspondentes $|\psi_n(x)|^2$, são a culminação da abordagem computacional. Na Figura 2a, observamos dois comportamentos distintos: (*i.*) um comportamento oscilatório dentro do poço, característico da partícula confinada, e (*ii.*) um decaimento exponencial suave para fora do poço, nas regiões classicamente proibidas. Cada uma dessas curvas representa um estado estacionário do sistema, com seu respectivo nível de energia. A visualização da função de onda estendendo-se para além das barreiras do poço é uma ilustração clara da física quântica, que permite a existência da partícula em locais inacessíveis para a física clássica.

A Figura 2b apresenta a densidade de probabilidade $P(x) = |\psi(x)|^2$. Físicamente, enquanto $P(x)$ representa a densidade em cada ponto x, a probabilidade de encontrar a partícula em um intervalo infinitesimal entre x e $x + dx$ é dada por $P(x)dx$. Essa representação é fundamental para visualizar como o confinamento no poço finito permite uma probabilidade não nula de localização da partícula fora das paredes do poço ($|x| > a$).

A altura da curva em cada ponto x representa a probabilidade de detecção. Observamos que à medida que o número quântico n aumenta, a função de onda se torna mais complexa, com um número crescente de nós (pontos de probabilidade nula). A densidade de probabilidade também decai exponencialmente fora do poço, confirmando visualmente o fenômeno do túnel quântico, onde a partícula tem uma probabilidade finita de ser encontrada nas regiões onde sua energia total é menor que o potencial. Juntas, as Figuras 2a,b servem como uma ferramenta didática robusta para construir uma intuição física sobre conceitos abstratos como quantização, distribuição de probabilidade e tunelamento, fortalecendo a conexão entre a matemática e a interpretação física da mecânica quântica.

Vale ressaltar que cada curva na Figura 2a representa uma solução da equação de Schrödinger associada a um nível de energia quantizado permitido para a partícula. Como o poço de potencial é composto pela: região II (dentro do poço), onde o potencial é mínimo e os estados ligados se localizam, e as regiões I e III (as barreiras), onde o potencial é mais elevado e, do ponto de vista clássico, a partícula não poderia existir. A transição entre essas regiões ilustra como as funções de onda e as probabilidades de presença da partícula variam com o potencial.

Essa convergência demonstra ao estudante que, embora a abordagem matricial seja uma aproximação numérica (ver Tabela 2), sua capacidade de reproduzir resultados analíticos com alta fidelidade a qualifica como uma alternativa poderosa para investigar potenciais para os quais não existem soluções exatas.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

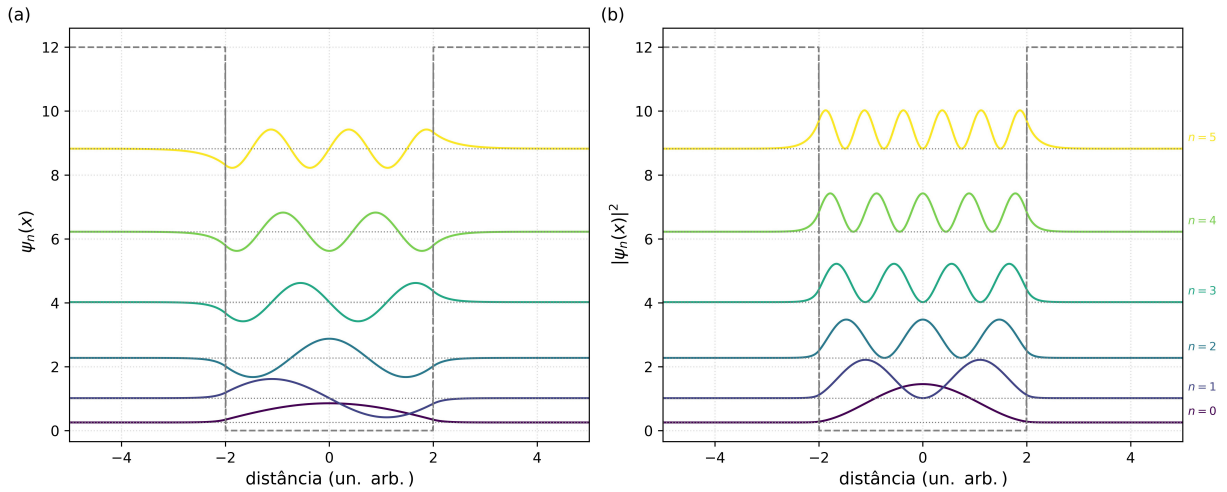

Figura 2: Visualização numérica de estados quânticos em um poço de potencial. (a) Funções de onda normalizadas calculadas numericamente pelo método das diferenças centradas, ilustrando os diferentes estados estacionários no poço de potencial finito. (b) Densidades de probabilidade $|\psi(x)|^2$, que indicam onde a partícula tem maior chance de ser encontrada. A comparação entre (a) e (b) reforça a interpretação física da função de onda e conecta o formalismo matemático com o significado probabilístico da mecânica quântica. Para fins de visualização, as curvas foram deslocadas verticalmente para coincidirem com os seus respectivos níveis de energia E_n.

Tabela 2: Comparação entre os resultados numéricos (E_{num}) e analíticos (E_{anali}) para os cinco primeiros autovalores de energia do poço finito. Os resultados demonstram a alta precisão do método de diferenças centradas.

n	E_{num}	E_{anali}
0	0.2542	0.2534
1	1.0148	1.0119
2	2.2748	2.2703
3	4.0204	4.0130
4	6.2242	6.2139
5	8.8242	8.8101

Para simplificar a implementação computacional, adotamos variáveis adimensionais definindo $\hbar = m = 1$. Para recuperar os valores em unidades do Sistema Internacional (SI), utiliza-se a relação:

$$E_{SI} = E_{num} \cdot \frac{\hbar^2}{mL^2}, \tag{19}$$

onde L é a escala de comprimento utilizada no código (por exemplo, largura do poço) e E_{num} é o valor numérico obtido. Como exemplo prático, para um elétron ($m \approx 9.1 \times 10^{-31}$ kg) em um poço de largura $L = 1$ nm, um autovalor numérico de $E_{num} = 0.25$ corresponde a uma energia física de aproximadamente 19.05 meV.

Código 9: Recuperando as unidades no Sistema Internacional.

```
# Constantes (SI)
hbar = 1.0545718e-34  # Constante de Planck reduzida (J.s)
m_e = 9.10938356e-31  # Massa do elétron (kg)
q_e = 1.60217663e-19  # Carga elementar (para conversão Joule -> eV)

# Parâmetros do sistema físico
L = 1.0e-9            # Largura do poço quântico (1 nm)
E_num = 0.25          # Autovalor adimensional obtido numericamente

# Cálculo da energia no SI (Joules)
fator_conversao = (hbar**2) / (m_e * (L**2))
E_joule = E_num * fator_conversao

# Conversão para milieletrón-volts (meV)
E_mev = (E_joule / q_e) * 1000

print(f"Energia numérica: {E_num}")
print(f"Energia em Joules: {E_joule:.4e} J")
print(f"Energia em meV: {E_mev:.2f} meV")
```

5. Oscilador Harmônico

Entre os sistemas quânticos exatamente solucionáveis, o oscilador harmônico ocupa um papel central, tanto pela sua relevância física quanto pelo valor pedagógico. Este sistema descreve uma partícula sujeita a uma força restauradora proporcional ao deslocamento em relação à posição de equilíbrio, sendo aplicável a uma ampla gama de fenômenos, desde vibrações moleculares até modos normais em sistemas de campos quânticos.

O Hamiltoniano unidimensional desse sistema é dado por:

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2 x^2, \tag{20}$$

e conduz, ao ser inserido na equação de Schrödinger independente do tempo, a

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\psi(x) + \left(\frac{1}{2}m\omega^2 x^2 - E\right)\psi(x) = 0. \tag{21}$$

As soluções analíticas desse problema são bem conhecidas: autofunções expressas em termos de polinômios de Hermite e autovalores igualmente espaçados, $E_n = \hbar\omega\left(n + \frac{1}{2}\right)$. No entanto, aqui optamos por uma abordagem numérica. Essa escolha se justifica por dois motivos principais: (*i.*) ressaltar o caráter didático dos métodos numéricos, mostrando como eles permitem resolver problemas mesmo quando as soluções analíticas não estão disponíveis, e (*ii.*) utilizar o oscilador como sistema de referência para validar a precisão do método aplicado.

5.1. Implementação computacional

A implementação do método de Numerov em Python segue uma sequência lógica que reflete diretamente o algoritmo teórico. O código é estruturado em etapas modulares, cada uma com um propósito bem claro: definir as funções-base, identificar os autovalores por meio de uma varredura de energia com refinamento por bisseção, processar os resultados e, finalmente, visualizá-los.

O valor pedagógico dessa implementação está em demonstrar como, a partir de condições iniciais simples e de um algoritmo recursivo eficiente, é possível recuperar numericamente os mesmos autovalores e autofunções obtidos analiticamente. Isso valida o método e, ao mesmo tempo, introduz o estudante a ferramentas indispensáveis para lidar com sistemas sem solução exata.

A primeira etapa é definir as funções principais do algoritmo. A função `K(x, E)` calcula o termo $k^2(x)$ da equação de Schrödinger em unidades adimensionais ($\hbar = m = \omega = 1$). A função `numerov(E, x)` aplica a fórmula recursiva do método de Numerov para um valor de energia de teste E, propagando a função de onda a partir de condições iniciais simples. Já a função `normalize(psi, x)` garante que a função de onda seja normalizada e que com fase consistente para visualização.

Código 10: Definição das funções principais do método de Numerov. Inclui a função $k^2(x)$, a implementação da recorrência de Numerov para propagação da função de onda e a normalização das autofunções.

```
import numpy as np
import matplotlib.pyplot as plt

### Funções auxiliares ###
def K(x, E):
    """k^2(x) para o oscilador harmônico (OH) adimensional."""
    return 2*E - x**2

def numerov(E, x):
    """Aplica o método de Numerov para o OH adimensional."""
    delta = x[1] - x[0]    # passo da malha espacial
    psi = np.zeros_like(x) # vetor para armazenar a solução

    # Condições iniciais
    psi[0] = 0.0
    psi[1] = delta

    # Recorrência de Numerov
    for i in range(2, len(x)):
        k0 = K(x[i-2], E)
        k1 = K(x[i-1], E)
        k2 = K(x[i], E)
        psi[i] = ((2 - 5*delta**2*k1/6)*psi[i-1] - (1 + delta**2*k0/12)*psi[i-2]) / (1 + delta**2*k2/12)
    return psi

def normalizacao(psi, x):
    """Normaliza a função de onda e garante fase positiva no pico."""
    norm = np.sqrt(np.trapz(np.abs(psi)**2, x))
    if norm > 0:
        psi = psi / norm
    # Ajuste de fase: garante que o maior pico seja positivo
    imax = np.argmax(np.abs(psi))
    if psi[imax] < 0:
        psi = -psi
    return psi
```

A busca pelos autovalores é feita combinando varredura de energia e bisseção. O algoritmo percorre um intervalo de energias e detecta mudanças de sinal em $\psi(x_{\max})$, o que indica a presença de um autovalor. Em seguida, o método da bisseção refina o resultado.

Código 11: Rotina de busca dos autovalores de energia do oscilador harmônico utilizando varredura e refinamento por bisseção. As autofunções correspondentes são normalizadas para análise posterior.

```
### Parâmetros do problema ###

# Domínio espacial: de -L até L
L = 6
N_pontos = 10000
x = np.linspace(-L, L, N_pontos)

E_min, E_max, dE = 0.4, 6.0, 0.01 # Intervalo de energias para busca dos autovalores
n_estados = 5 # Número de estados desejados

### Busca de autovalores ###
energia = []
list_psi = []

E_valores = np.arange(E_min, E_max, dE)
f_prev = numerov(E_valores[0], x)[-1]  # psi no último ponto

for E in E_valores[1:]:
    psi = numerov(E, x)
    f_curr = psi[-1]
    """
    Se há troca de sinal no valor da função de onda no contorno,
    então cruzamos um autovalor verdadeiro (condição de contorno satisfeita)
    """
    if f_prev * f_curr < 0:

        # Refina a energia encontrada usando bisseção
        a, b = E - dE, E
        for _ in range(20): # 20 iterações garantem boa precisão
            c = 0.5*(a + b)
```

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2025-0447

```
            psi_c = numerov(c, x)
            if psi_c[-1] * numerov(a, x)[-1] < 0:
                b = c
            else:
                a = c
        E_star = 0.5*(a + b) # Autovalor refinado
        psi_star = normalizacao(numerov(E_star, x), x) # Autofunção correspondente, normalizada

        energia.append(E_star)
        list_psi.append(psi_star)

        # Interrompe quando já encontramos todos os estados desejados
        if len(energia) >= n_estados:
            break
    f_prev = f_curr

energia = np.array(energia)
array_psi = np.array(list_psi)

### Potencial do oscilador harmônico ###
def V(x):
    return 0.5*x**2
```

A Tabela 3 compara os resultados numéricos com os valores analíticos, confirmando a precisão do método de Numerov:

Tabela 3: Comparação entre os resultados numéricos (E_{num}) e analíticos (E_{anali}) para os cinco primeiros autovalores de energia do oscilador harmônico. Os resultados demonstram a alta precisão do método de Numerov.

n	E_{num} (eV)	E_{anali} (eV)
0	0.5000	0.5000
1	1.5000	1.5000
2	2.5000	2.5000
3	3.5000	3.5000
4	4.5000	4.5000

Da Tabela 3, observa-se que o desvio em relação aos valores analíticos é extremamente reduzido, reforçando a robustez e o alto nível de confiabilidade deste método para potenciais continuamente variáveis.

A etapa final consiste em visualizar as soluções. O primeiro gráfico exibe as funções de onda normalizadas e os níveis de energia, enquanto o segundo mostra as densidades de probabilidade.

Código 12: Visualização numérica dos estados quânticos do oscilador harmônico. O painel (a) mostra as funções de onda normalizadas sobrepostas ao potencial harmônico e aos níveis de energia; o painel (b) exibe as densidades de probabilidade associadas.

```
## Construção gráfica
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

# Colormap para diferenciar cada estado
cores = plt.cm.viridis(np.linspace(0.2, 0.8, len(energia)))

### Painel (a): Funções de onda ###

# Desenhando o potencial
ax1.plot(x, V(x), color='tab:gray', linestyle='--', linewidth=1.2, label=r'$V(x)$')

for i, E_n in enumerate(energia):
    """Define um fator de escala (ajuste se necessário)"""
    escala = 0.35 / np.max(np.abs(array_psi[i]))

    ax1.plot(x, E_n + escala*array_psi[i], color=cores[i], linewidth=1.4,
             label=rf"$n={i}$, $E={E:.4f}$")
    ax1.hlines(E_n, x[0], x[-1], linestyle=':', color='gray', linewidth=0.8)

### Painel (b): Densidades de probabilidade ###

# Desenhando o potencial
ax2.plot(x, V(x), color='gray', linestyle='--', linewidth=1.2, label=r'$V(x)$')

for i, E_n in enumerate(energia):
    """Define um fator de escala (ajuste se necessário)"""
    escala_prob = 0.35 / np.max(abs(array_psi[i]**2))

    ax2.plot(x, E_n + escala_prob*array_psi[i]**2, color=cores[i], linewidth=1.4,
             label=rf"$n={i}$, $E={E:.4f}$")
    ax2.hlines(E_n, x[0], x[-1], linestyle=':', color='gray', linewidth=0.8)

    ax2.text(x[-1] + 0.3, E_n, f"$n={i}$", color=cores[i], fontsize=10, va='center')

### Ajustes finais ###
for ax, label in zip([ax1, ax2], ['(a)', '(b)']):
    ax.set_xlim(-L, L)
    ax.set_ylim(0, max(energia) + 1)
    ax.grid(True, linestyle=':', alpha=0.4)
    ax.text(-0.12, 1.05, label, transform=ax.transAxes, fontsize=12, va='top', ha='left')

ax1.set_xlabel(r"posição $x$ $(\mathrm{un.\ arb.})$", fontsize=12)
ax2.set_xlabel(r"posição $x$ $(\mathrm{un.\ arb.})$", fontsize=12)
ax1.set_ylabel(r"$\psi_n(x)$", fontsize=12)
ax2.set_ylabel(r"$|\psi_n(x)|^2$", fontsize=12)

plt.tight_layout()
plt.show()
```

As Figuras 3a,b ilustram claramente os aspectos teóricos: (*i.*) as funções de onda (Figura 3a) apresentam comportamento oscilatório dentro do potencial e decaimento exponencial fora dele; (*ii.*) as densidades de probabilidade (Figura 3b) mostram como o estado fundamental tem forma aproximadamente gaussiana, enquanto os estados excitados apresentam nós e estrutura crescente de picos.

Uma observação central é que os níveis de energia do oscilador harmônico são igualmente espaçados, em contraste com o poço de potencial finito. Essa diferença, visível tanto nos gráficos quanto na tabela, reforça

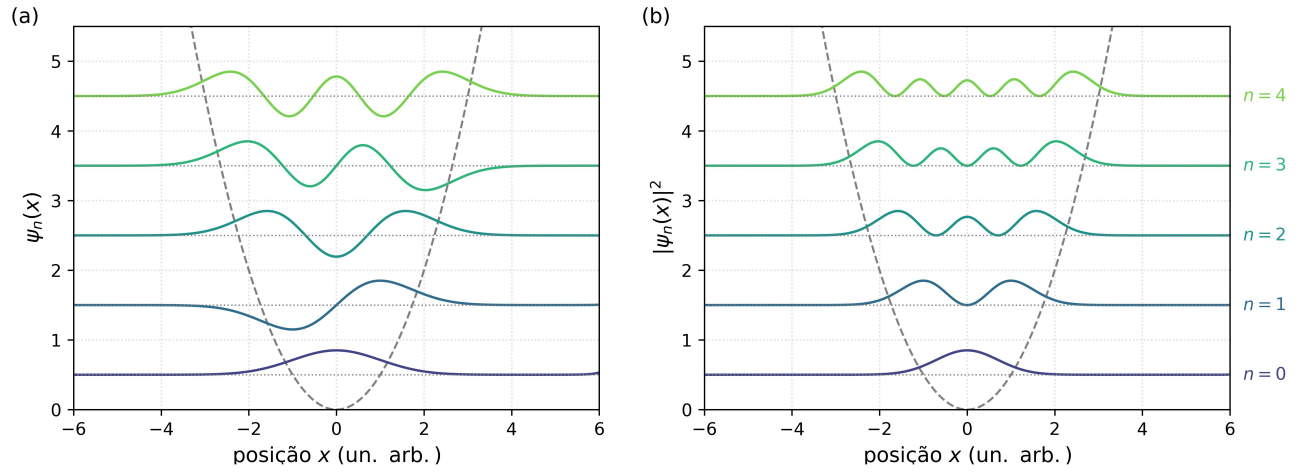

Figura 3: Visualização numérica do oscilador harmônico. (a) Funções de onda $\psi_n(x)$ obtidas pelo método de Numerov, sobrepostas ao potencial harmônico $V(x)$ e aos respectivos níveis de energia E_n. (b) Densidades de probabilidade $|\psi_n(x)|^2$ normalizadas, destacando a distribuição espacial da partícula para diferentes estados quânticos. A comparação entre (a) e (b) reforça a interpretação probabilística da função de onda e evidencia o espaçamento uniforme entre os níveis de energia, característica do oscilador harmônico. Para fins de visualização, as curvas foram deslocadas verticalmente para coincidirem com os seus respectivos níveis de energia E_n.

a intuição do aluno sobre como diferentes potenciais moldam os espectros quânticos.

6. Considerações Finais

A implementação de métodos computacionais em Python, utilizando o ambiente Jupyter Notebook, demonstrou ser uma ferramenta de potencial valor didático para auxiliar o ensino de mecânica quântica, permitindo a visualização de conceitos abstratos e a experimentação virtual. A discretização da equação de Schrödinger e a resolução numérica permitiram obter aproximações confiáveis para os níveis de energia e funções de onda, favorecendo a visualização de conceitos abstratos e contribuindo para a construção de uma intuição física mais sólida. O maior mérito deste trabalho reside no fato de ter sido integralmente proposto, desenvolvido e conduzido por estudantes de graduação, em uma iniciativa de aprendizagem ativa que evidenciou como a integração entre teoria, programação e visualização gráfica pode fortalecer o engajamento com conteúdos complexos da física moderna e estimular maior protagonismo discente. Outro aspecto de destaque é sua dimensão de acessibilidade. Ao disponibilizar os materiais em português e em formato de acesso aberto, este trabalho contribui para a democratização do ensino de física no Brasil, incentivando a adoção de metodologias computacionais inovadoras no ambiente universitário e escolar. Dessa forma, amplia-se o alcance de conteúdos didáticos de qualidade e cria-se um espaço fértil para a formação crítica e criativa. Em síntese, acreditamos que este trabalho não apenas fornece um recurso didático útil, mas também pode inspirar a próxima geração de físicos a explorar de maneira integrada a interface entre a física e a computação, enriquecendo tanto a compreensão conceitual quanto a prática pedagógica em mecânica quântica.

Contribuição dos autores

C.B., J.A.L.O. e N.P.S. contribuíram igualmente para este trabalho. O projeto foi concebido, idealizado e desenvolvido pelos discentes C.B., J.A.L.O. e N.P.S. como parte de uma iniciativa de aprendizagem ativa no curso de graduação em Física, com o objetivo de disponibilizar materiais acessíveis, produzidos em língua portuguesa, para o ensino de Física. C.B. e N.P.S. foram responsáveis pela redação da introdução e pela solução do problema do poço de potencial finito. J.A.L.O. redigiu a seção de abordagem teórica e conduziu a resolução do problema do oscilador harmônico. B.R.C. supervisionou todas as etapas do projeto, além de colaborar na revisão crítica e na edição final do manuscrito. Todos os autores participaram da revisão crítica do trabalho e aprovaram a versão final.

Agradecimentos

C.B., J.A.L.O., N.P.S. e B.R.C. agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Universidade Federal do Rio Grande do Norte pelo apoio institucional.

Referências

[1] H.R. Armando, M. Ferreira, O.L.S. Filho e D.L. Azevedo, Rev. Bras. Ens. Fís. **47**, e20250043 (2025).

[2] G.A. Monerat, L.G.F. Filho, E.V.C. Silva, G. Oliveira-Neto, P.H.A.S. Nogueira e A.R.P. Assunção, Rev. Bras. Ens. Fís. **32**, 1304 (2010).

[3] E.L. Pinheiro, K.M. Barata, O.J.N. Neto, R.S. Martins e M.R.S. Siqueira, Rev. Bras. Ens. Fís. **46**, e20240011 (2024).

[4] F. Caruso, V. Oguri e F. Silveira, Rev. Bras. Ens. Fís. **44**, e20220098 (2022).

[5] R.C.S. Tonhon e L. Madeira, Rev. Bras. Ens. Fís. **46**, e20240300 (2024).

[6] F. Caruso e V. Oguri, Rev. Bras. Ens. Fís. **36**, 2310 (2014).

[7] M.S.M. Sousa e F.A. Nascimento, Rev. Bras. Ens. Fís. **42**, e20190144 (2020).

[8] C. Baliana, J. A. Luna Oliveira, N. Pereira dos Santos e B. R. Carvalho, *Equação de Schrödinger com Python*, disponível em: https://github.com/Josearthur266/Equacao-de-Schrodinger-com-Python, acessado em: 10/01/2026.

[9] D.J. Griffiths, *Introduction to Quantum Mechanics* (Cambridge University Press, Cambridge, 2018).

[10] D. Bennett, *Numerical solution of time-independent 1-D Schrodinger equation*, disponível em: https://www.maths.tcd.ie/~dbennett/js/schro.pdf, acessado em: 05/10/2025.

[11] M. Purevkhuu e V.I. Korobov, Phys. Part. Nuclei Lett. **18**, 153 (2021).

[12] Anaconda, *Anaconda Software Distribution. Web. Versão 2-2.4.0*, disponível em: https://anaconda.com, acessado em: 05/09/2025.

[13] C. Cohen-Tannoudji, B. Diu e F. Laloë, *Quantum Mechanics* (John Wiley & Sons, 2019), v. 1, 2 ed.

[14] M.E.J. Newman, *Computational Physics* (CreateSpace Independent Publishing Platform, North Charleston, 2013).