Solução Híbrida para Dinâmica Quântica: Unindo Physics-Informed Neural Networks e Mecânica Matricial

Autor: Vagner Jandre Monteiro[1]

[1]Universidade do Estado do Rio de Janeiro, Instituto Politécnico - Nova Friburgo, RJ, Brasil.

Data: 05/12/2025

Resumo

Este trabalho apresenta uma abordagem computacional para a resolução da Equação de Schrödinger 1D fundamentada em *Physics-Informed Neural Networks* (PINNs). Em contraste com métodos tradicionais de discretização em malha (*mesh-based*), a proposta utiliza a rede neural como um aproximador funcional contínuo para determinar simultaneamente as funções de onda e os autovalores de energia, minimizando o resíduo da equação diferencial na função de perda. O estudo introduz uma metodologia híbrida inovadora: a PINN é empregada para obter os autoestados estacionários, enquanto a evolução temporal é computada analiticamente através do formalismo da Mecânica Matricial. A validação no sistema do Oscilador Meio-Harmônico demonstra que esta estratégia, aliada a técnicas de correção de fase (*Gauge Fixing*) e integração numérica de alta precisão, mitiga eficazmente ruídos numéricos e alcança concordância rigorosa com métodos espectrais de referência.

1. Introdução

A simulação de sistemas quânticos constitui um desafio central na física computacional moderna. Métodos numéricos convencionais, como Diferenças Finitas (FDM) ou Elementos Finitos (FEM), embora robustos, frequentemente enfrentam limitações severas, como a "maldição da dimensionalidade" e a dependência de malhas de discretização que podem introduzir erros significativos de truncamento. Como alternativa, o uso de redes neurais como aproximadores universais para a resolução de equações diferenciais foi proposto pioneiramente por Lagaris et al. (1998) [4]. Este trabalho estabeleceu as bases para métodos computacionais *mesh-free*, capazes de resolver equações diferenciais parciais (EDPs) sem a necessidade de uma malha predefinida para discretizar o domínio do problema.

Recentemente, esta abordagem foi formalizada e significativamente expandida sob a denominação de *Physics-Informed Neural Networks* (PINNs) por Raissi et al. (2019) [6]. A principal inovação das PINNs reside no uso sistemático de diferenciação automática para computar o resíduo da equação diferencial e incorporá-lo diretamente na função de perda da rede. No entanto, a aplicação direta de PINNs para modelar a dinâmica temporal (equações dependentes do tempo) apresenta desafios consideráveis, notadamente a

propagação acumulativa de erros e o elevado custo computacional associado ao treinamento em domínios espaço-temporais contínuos.

Neste contexto, este trabalho propõe uma arquitetura híbrida que desacopla o problema espacial do temporal. Utilizamos uma PINN para resolver o problema de autovalor da Equação de Schroedinger Independente do Tempo, obtendo os estados estacionários com alta precisão. Subsequentemente, empregamos o formalismo da Mecânica Matricial de Heisenberg, conforme descrito em Griffiths (2018), para calcular a evolução dinâmica. Esta estratégia evita a integração temporal numérica direta pela rede, garantindo a conservação de unitariedade e proporcionando uma precisão espectral na descrição da evolução do sistema.

2. Metodologia

2.1 Modelo Físico

2.1.1 A Equação de Schroedinger

Em 1925, o físico austríaco Erwin Schroedinger (1887-1961) formulou a equação fundamental que governa a dinâmica de sistemas quânticos não-relativísticos. Esta equação desempenha na Mecânica Quântica um papel análogo às Leis de Newton na Mecânica Clássica, permitindo prever a evolução temporal de estados físicos e as energias permitidas de sistemas como átomos e [1][2].

Em sua forma unidimensional dependente do tempo, a equação é expressa como:

$$i\hbar \frac{\partial \Psi(x,t)}{\partial t} = \widehat{H}\Psi(x,t) \tag{1}$$

Em que $\widehat{H}$ é o operador Hamiltoniano representando a energia total do sistema e $\hbar$ é a constante reduzida de Planck. A função $\Psi(x,t)$ é a função de onda do sistema. A interpretação física desta função foi proposta pelo físico alemão Max Born (1882-1970) que postulou que o módulo ao quadrado da função de onda representa a densidade de probabilidade de encontrar a partícula em uma posição no instante t [1]:

$$P(x,t) = |\Psi(x,t)|^2 \tag{2}$$

Para uma partícula de massa m sujeita a um potencial $V(x)$, o Hamiltoniano é assoma dos operadores de energia cinética e potencial. Na representação de posição, o operador momento é dado por $\hat{p} = -i\hbar\frac{\partial}{\partial x}$, resultando em:

$$\widehat{H} = -\frac{\hbar}{2m}\frac{\partial^2}{\partial x^2} + V(x) \tag{3}$$

Substituindo (3) em (1), obtemos a forma explícita da equação:

$$i\hbar \frac{\partial \Psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2 \Psi(x,t)}{\partial x^2} + V(x)\Psi(x,t) \tag{4}$$

2.1.2 Estados Estacionários e Unidades Naturais

Para potenciais independentes do tempo, aplicamos o método de separação de variáveis, assumindo uma solução da forma $\Psi(x,t) = \psi(x)\phi(t)$. Isso leva a uma solução temporal oscilatória $\phi(t) = e^{-\frac{i}{\hbar}Et}$ e à equação de Schroedinger independente do tempo para a parte espacial:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x) \tag{5}$$

Neste trabalho, adotamos um sistema de unidades atômicas modificadas para simplificação numérica, onde definimos $\hbar = 1$ e a massa $m = \frac{1}{2}$. Com essa convenção, a Equação (5) simplifica-se para a forma utilizada e implementada no algoritmo computacional:

$$-\frac{\partial^2\psi(x)}{\partial x^2} + V(x)\psi(x) = E\psi(x) \tag{6}$$

O objetivo dos métodos numéricos, como o PINN proposto, é encontrar as autofunções $\psi_n(x)$ e os autovalores de energia E_n que satisfazem a Eq. (6) sob determinadas condições de contorno. A solução geral para a dinâmica do sistema é então construída como uma superposição linear dos estados estacionários:

$$\Psi(x,t) = \sum_n \psi_n(x)e^{-\frac{i}{\hbar}Et} \tag{7}$$

2.1.3 O Oscilados Meio-Harmônico (Half-Harmonic Oscillator)

Como estudo de caso para validação do método, utilizamos o Oscilador Meio-Harmônico. Este modelo mantém a estrutura do potencial harmônico simples ($V \propto x^2$), mas impõe uma barreira de potencial infinita na origem ($x \leq 0$).

O potencial é definido como:

$$V(x) = \begin{cases} x^2, & x > 0 \\ \infty, & x \leq 0 \end{cases}$$

Esta modificação introduz uma condição de contorno de Dirichlet rígida em $x = 0$, ou seja, $\psi(0) = 0$. Fisicamente, as soluções para este sistema correspondem exatamente às soluções de paridade ímpar do oscilador harmônico completo ($n = 1, 3, 5...$ do oscilador completo, reindexados aqui como $n = 0, 1, 2..$). As soluções analíticas conhecidas [2] envolvem os Polinômios de Hermite $H_k(x)$:

$$\psi_n \propto H_{2n+1}(x)\sum_n \psi_n(x)e^{-\frac{x^2}{2}} \tag{8}$$

Este sistema é ideal para testar redes neurais informadas pela física (PINNs) pois exige que a rede aprenda não apenas a equação diferencial na região suave ($x > 0$), mas também a descontinuidade abrupta e a condição de contorno na origem, desafiando a capacidade de aproximação do modelo.
Para as simulações dinâmicas apresentadas nos resultados, consideramos uma superposição equiprovável dos três primeiros níveis de energia ($n = 0, 1, 2$) no domínio $x \in [0,6]$.

2.1.4 Dinâmica via Mecânica Matricial

Uma das inovações deste trabalho é a abordagem híbrida para a simulação temporal. Ao invés de estender o treinamento da rede neural diretamente para o domínio temporal (t) — o que aumentaria exponencialmente o custo computacional e a propagação de erros —, optou-se por construir a solução geral como uma superposição linear dos estados estacionários obtidos pela PINN:

$$\Psi(x,t) = \sum_n c_n \psi_n(x) e^{-iE_n t} \tag{9}$$

Em que c_n são os coeficientes complexos que determinam o estado inicial do sistema. Para calcular a evolução temporal de observáveis dinâmicos, especificamente a posição esperada $\langle x \rangle(t)$ e o momento linear $\langle p \rangle(t)$, implementou-se um algoritmo de pós-processamento baseado no formalismo da Mecânica Matricial de Heisenberg. Este método consiste em três etapas fundamentais:

I. Gauge Fixing (Correção de Fase): Devido à natureza estocástica da inicialização dos pesos, a rede neural pode convergir para autofunções com sinais globais arbitrários ($\pm\psi_n$). Embora isso não afete a densidade estacionária $|\psi|^2$, a fase relativa é crucial para a dinâmica de superposição. Dessa forma, implementou-se um algoritmo que alinha a fase das funções de onda neurais com a convenção física padrão (alternância de paridade dos Polinômios de Hermite: $+, -, +, ...$), garantindo a correta interferência construtiva/destrutiva e, consequentemente, a direção correta do vetor momento linear.

II. Integração de Precisão: Os elementos de matriz dos operadores de posição, $X_{nm} = \langle n|\hat{x}|m\rangle$ e do momento $P_{nm} = \langle n|-i\,\partial_x|m\rangle$, são pré-calculados uma única vez. Para isso, utiliza-se a Regra de Simpson composta, que oferece uma precisão de ordem $O(h^4)$, minimizando ruídos numéricos decorrentes da discretização espacial.

III. Evolução Analítica: Com as matrizes X e P construídas, a dinâmica não requer nova integração numérica a cada passo de tempo. A evolução é obtida via álgebra linear exata:

$$\langle \hat{O} \rangle(t) = c(t)^\dagger \cdot O_{matrix} \cdot c(t) \tag{10}$$

Em que o vetor de coeficientes evolui analiticamente como $c_n(t) = c_n(0)e^{-iE_n t}$. Isso garante que a simulação seja livre de erros de dissipação temporal, limitando-se apenas à precisão dos autoestados encontrados pela PINN.

2.2. Estrutura do Código Implementado

A implementação foi projetada de forma modular para garantir a reprodutibilidade e a separação clara de responsabilidades. O fluxo de execução é orquestrado pelo *script* principal `train.py`, que inicializa o experimento consumindo hiperparâmetros definidos externamente no arquivo `config.yaml`. O núcleo computacional reside na classe `PINNSolver1D` (`src/models`), responsável por instanciar a arquitetura da rede neural e minimizar a função de perda física para um único autoestado. A gestão de múltiplos níveis de energia é realizada pela classe `QuantumSystemTrainer` (em `src/utils`), que automatiza a descoberta sequencial de estados excitados impondo restrições de ortogonalidade. Por fim, a classe `SuperpositionManager` centraliza o pós-processamento físico, implementando algoritmos críticos como a correção de fase (*Gauge Fixing*), a

integração numérica de precisão (Simpson) e o formalismo da Mecânica Matricial para calcular e visualizar a dinâmica temporal exata dos observáveis.

A tabela a seguir relaciona as funções implementadas disponíveis para obtenção dos observáveis Físicos (posição e momento) que utilizam o formalismo da Mecânica Matricial. Como já mencionado, elas não integram a função de onda a cada passo de tempo; em vez disso, realizam a operação algébrica da Eq.(10), garantindo velocidade e precisão.

Tabela 1 – Métodos implementados para a obtenção dos observáveis.

Chamada	Objetivo	Saída
`expected_position_and_uncertainty_are_calculated(t_range, coefficients, num_points)`	Calcula a evolução temporal do valor esperado da posição $\langle x \rangle$ e sua incerteza $\sigma_x = \sqrt{\langle x^2 \rangle - \langle x \rangle^2}$.	Gera um arquivo $.txt$ contendo 4 colunas: Tempo, Valor Esperado, Limite Inferior ($\langle x \rangle - \sigma_x$) e Limite Superior ($\langle x \rangle + \sigma_x$).
`expected_position_and_uncertainty_are_plotted()`	Lê o arquivo $.txt$ gerado pela função anterior e renderiza o gráfico.	Salva a imagem `_ExpectedPositionAndUncertainty`.png, exibindo a oscilação da partícula e a incerteza.
`expected_momentum_and_uncertainty_are_calculated(t_range, coefficients, num_points)`	Calcula a evolução temporal do momento linear $\langle p \rangle$ e sua incerteza σ_p.	Gera um arquivo $.txt$ com os dados numéricos.
`expected_momentum_and_uncertainty_are_plotted()`	Lê os dados de momento e plota as curvas.	Salva a imagem `_ExpectedMomentumAndUncertainty.png`.

A próxima tabela relaciona as funções implementadas para visualização da função de onda e a densidade de probabilidade. Estas funções reconstroem a solução completa, Eq.(9), para visualização direta do comportamento quântico.

Tabela 2 – Métodos implementados para visualização e comparação da função de onda.

Chamada	Objetivo	Saída
`plot_spacetime_density(t_max, steps)`	Gera um mapa de calor espaço-temporal, mostrando **a** evolução da densidade de probabilidade $\|\Psi(x,t)\|^2$ ao longo do tempo (até o valor máximo `t_max`) e posição. Permite visualizar a trajetória do pacote de onda "visto de cima".	`Imagem _Spacetime.png`.

`plot_snapshots(times_to_plot)`	Plota o perfil da densidade de probabilidade $\|\Psi(x)\|^2$ para instantes específicos de tempo (ex: times_to_plot = $[0, 1.5, 3.0]$) . Ilustra como o pacote de onda se deforma, se espalha ou se desloca no poço de potencial em momentos chave.	`Imagem _Snapshots.png.`
`plot_wavefunction_real( times_to_plot)`	Plota a Parte Real da função de onda: $\mathrm{Re}[\Psi(x,t)]$, mostrando que, diferente da densidade (sempre positiva), a parte real oscila entre positivo e negativo.	`Imagem _WaveFunction.png.`

2.3. O Solver PINN

O método proposto emprega uma rede neural como aproximador funcional universal para resolver o problema de autovalor da Equação de Schroedinger Independente do Tempo (Eq.(6)) de maneira contínua, eliminando a dependência de malhas de discretização (*mesh-free*). O modelo aprende simultaneamente a topologia da função de onda $\psi(x)$ e o autovalor de energia E — tratado aqui como um parâmetro treinável — através da minimização de uma função de perda composta. Esta função penaliza matematicamente os resíduos da equação diferencial, as violações das condições de contorno e a ausência de normalização da densidade de probabilidade.

Para a obtenção do espectro de energia, aplica-se uma estratégia de treinamento sequencial com restrição de ortogonalidade. Após a convergência do estado fundamental, a rede é induzida, via termos adicionais na função de custo, a convergir para estados excitados que sejam ortogonais aos estados previamente descobertos. Essa abordagem converte a resolução da EDP em um problema de otimização, fornecendo bases estacionárias precisas que, subsequentemente, viabilizam o cálculo exato da dinâmica temporal via Mecânica Matricial.

Especificamente para o sistema do Oscilador Meio-Harmônico, a rede neural $\mathcal{N}(x;\theta)$ é treinada para aproximar $\psi_n(x)$ no domínio $x \in [0,6]$. A função de perda total $\mathcal{L}$, responsável por guiar o aprendizado para comportamentos fisicamente consistentes, é definida como:

$$\mathcal{L} = \mathcal{L}_{\mathcal{PDE}} + \lambda_{BC}\mathcal{L}_{\mathcal{BC}} + \lambda_{Norm}\mathcal{L}_{\mathcal{N}orm} + \lambda_{Ortho}\mathcal{L}_{\mathcal{O}rtho}. \quad (11)$$

Em que os termos possuem as seguintes considerações físicas:

1. Física ($\mathcal{L}_{\mathcal{PDE}}$): O resíduo de Schroedinger $\left|-\frac{1}{2}\frac{d^2\psi}{dx^2} + V\psi - E\psi\right|^2$, calculado via diferenciação automática.
2. Condições de Contorno ($\mathcal{L}_{\mathcal{BC}}$): penaliza desvios dos valores esperados nas bordas do domínio (condição de Dirichlet: $\psi(x_{min}) = \psi(x_{max}) = 0$)

3. Normalização ($\mathcal{L}_{Norm}$): Garante a interpretação probabilística da função de onda impondo $\int |\psi(x)|^2 \, dx = 1$. Este termo evita que a rede convirja para a solução trivial nula ($\psi = 0$).
4. Ortogonalidade ($\mathcal{L}_{Ortho}$): Para encontrar estados excitados ($n > 0$), penalizamos a sobreposição com estados anteriores já convergidos: $\sum_{k<n} |\langle \psi | \psi_k \rangle|^2$, forçando a descoberta de novos modos de vibração.

A função de perda (Eq. (11)) é uma soma ponderada de objetivos concorrentes. Os coeficientes λ atuam como hiperparâmetros de regularização para mitigar o problema de desequilíbrio de gradientes (*Gradient Pathology*). Como as condições físicas (fronteira e normalização) são restrições rígidas, atribuímos a elas pesos significativamente maiores que o resíduo da PDE para garantir que sejam satisfeitas prioritariamente durante a otimização.

Após testes iniciais de rapidez e coerência de convergência do algoritmo, adotou-se a seguinte configuração empírica para garantir a convergência correta do método:

Tabela 3 - Hiperparâmetros utilizados na função de Loss.

Hiperparâmetro	**Valor**	**Função**
λ_{PDE}	1.0	Referência base
λ_{BC}	100.0	Reforça estritamente as condições de Dirichlet.
λ_{Norm}	100.0	Penalidade alta para evitar o colapso para a solução trivial nula.
λ_{Ortho}	500.0	O peso mais elevado é atribuído à ortogonalidade para criar uma barreira de potencial efetiva na superfície de perda, impedindo que os estados excitados sejam levados para o estado fundamental (colapso variacional).

Para a otimização, adotou-se uma estrutura híbrida, utilizando uma estratégia de dois estágios. Inicialmente, o otimizador Adam [3] é usado para exploração global. Em seguida, é aplicado o otimizador L-BFGS [5], um método de quase-Newton, com *learning rate* unitário para refinamento de alta precisão (atingindo perdas da ordem de 10^{-13}).

A tabela a seguir mostra os parâmetros utilizados nos otimizadores.

Tabela 4 – Otimizadores e parâmetros utilizados na busca do autovalor de energia E_n. Argumentos implícitos seguiram o padrão Default do PyTorch

Adam			
	Argumento	**Valor**	**Função**
Rede Neural	`self.model.parameters()`	`N/A`	Todos os pesos e vieses da MLP
	`Learning Rate (lr)`	`5e-5`	Taxa pequena para garantir estabilidade e evitar que a função de onda "quebre" ou oscile demais enquanto aprende a forma suave).
	`[self.E]`	`N/A`	Apenas o escalar da energia.

Autovalor de Energia	`Learning Rate (lr)`	`5e-2`	Taxa 1000x maior que a da rede. Necessário para o valor de E "saltar" rapidamente (ex: de 2.0 para 7.0) no início do treino, enquanto os pesos da rede precisam apenas de ajustes finos.

L-BFGS		
Argumento	**Valor**	**Função**
`params`	`list(model) + [E]`	Lista de tensores para otimizar. Inclui os pesos da rede neural e o autovalor de energia E.
`lr`	`1.0`	Taxa de aprendizado (`Step size`). Em métodos de Newton, o passo natural matemático é 1.0. Valores menores inibem a inteligência do otimizador. O *Line Search* reduzirá esse valor se necessário.
`max_iter`	`epochs_lbfgs=6000`	Máximo de iterações por chamada de `step()`. Como chamamos o `step()` apenas uma vez fora do loop, isso define a duração total do treinamento nesta fase.
`max_eval`	`epochs_lbfgs * 1.25`	Máximo de chamadas à função `closure()`. O *Line Search* pode precisar avaliar a função de perda várias vezes para validar um único passo. Por segurança, definimos um valor maior que `max_iter`.
`history_size`	`500`	Memória da correção da Hessiana. Foi utilizado 500 (default = 100) para o otimizador ter uma "memória" maior da curvatura do problema, melhorando a precisão em vales estreitos.
`tolerance_grad`	`1e-12` (10^{-12})	Critério de parada: Gradiente. Se a norma do gradiente for menor que o valor, ele para. Necessário trabalhar com `float64`.
`tolerance_change`	`1e-12` (10^{-12})	Critério de parada: Mudança. Se a *Loss* ou os parâmetros mudarem menos que isso entre dois passos, ele assume convergência e para.
`line_search_fn`	`"strong_wolfe"`	Algoritmo de busca de linha. Garante que o passo dado satisfaça as condições de Wolfe (redução suficiente da energia e da curvatura).

Por fim, as configurações utilizadas para as iterações de treino do modelo foram as seguintes:

Tabela 5 – Otimizadores e seus parâmetros utilizados na busca do autovalor de energia E_n. Argumentos implícitos seguiram o padrão Default do PyTorch

Aplica ao	Argumento	Valor	Função
Modelo	`layers`	`[1, 64, 64, 64, 1]`	Topologia da rede neural. Uma entrada, 3 camadas profundas com 64 neurônios cada e ativação $tanh$ e uma saída.
	`n_points`	`10000`	Número de pontos de colação amostrados aleatoriamente no domínio (usados para calcular o resíduo da equação diferencial (física) durante o treino).
Treinamento	`N_sates`	`3`	Instrução para a classe `QuantumSystemTrainer` encontrar sequencialmente os 3 primeiros autovalores de energia ($n = 0, 1, 2$).
	`initial_E0_guess`	`0.0`	Valor inicial para o parâmetro treinável E ao buscar o estado fundamental.
	`delta_E_guess`	`1.0`	Estimativa do "salto" de energia entre níveis. Usado para inicializar a busca do próximo estado ($E_{n+1} \approx E_{final\_n} + \Delta E$).
	`epochs_adam`	`7000`	Ciclos usando o otimizador Adam (exploração global rápida para escapar de mínimos locais iniciais).
	`epochs_lbfgs`	`6000`	Ciclos usando o otimizador L-BFGS (refinamento de alta precisão para garantir que o erro da equação diferencial caia para níveis próximos de zero (10^{-6} ou menos)).

3. Resultados e Discussão

O modelo foi validado comparando-se os resultados com um Método Espectral clássico, implementado seguindo as diretrizes de Trefethen (2000) [7] e utilizando como ferramenta computacional a biblioteca disponível para a linguagem Python: PySpectral Quantum Solver[1], considerado o *baseline* de alta precisão para este problema.

[1] Disponível em: https://github.com/Legeandre/PySpectral-Quantum-Solver.git

3.1. Precisão dos Autovalores

O uso de precisão dupla (`float64`) e otimização L-BFGS permitiu uma convergência adequada das energias, conforme mostrado na Tabela 6.

Tabela 6 – valores obtidos para a energia E_n, de acordo com o método empregado.

Estado (n)	**Energia**			**Diferença PIN - Espectral**
	Teórica	**PINN**	**Espectral**	
0	3.0	2.999992545	2.999999999	$7.454.10^{-6}$
1	7.0	6.999999862	7.000000000	1.38e-7
2	11.0	11.000004891	11.000000031	4,86e-6

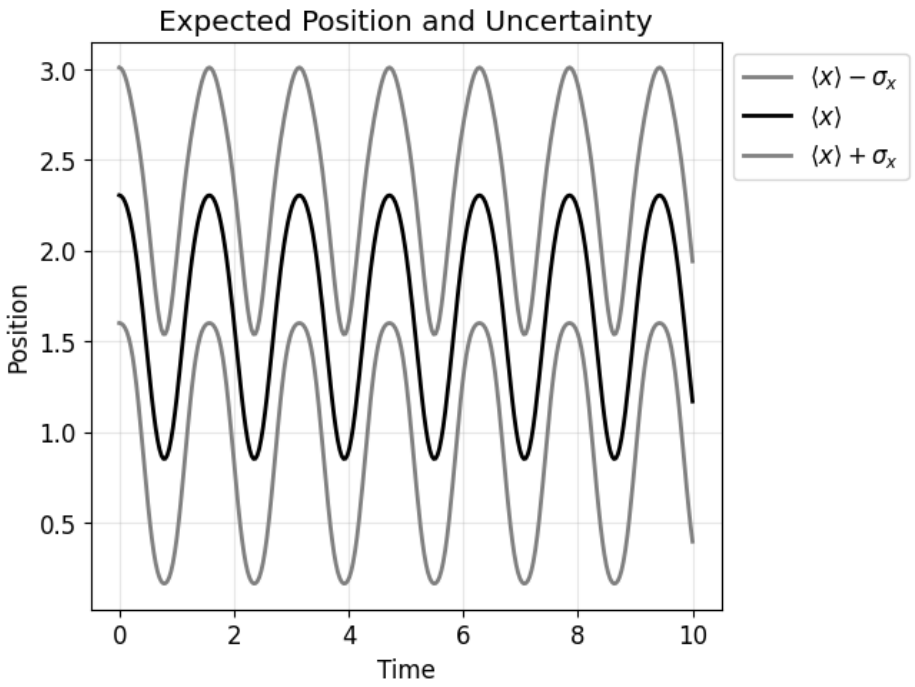
O método espectral apresenta vantagem em termos de velocidade de execução, sendo eficiente para os primeiros níveis de energia. Contudo, sua precisão é limitada pela resolução da malha, degradando-se conforme o número quântico n aumenta e as funções de onda tornam-se mais oscilatórias. Diferentemente, a abordagem híbrida com PINN, por não depender de malhas fixas, sustenta seu nível de precisão mesmo para estados mais altos, embora exija um custo computacional significativamente maior devido à natureza iterativa do treinamento da rede neural.

3.2. Dinâmica do Pacote de Onda

Para gerar os dados para a reprodução da dinâmica do pacote de onda no problema estudado, simulamos uma superposição equiprovável dos três estados com coeficientes $c_0 = c_1 = c_2 = 1$, dentro do intervalo de tempo de 0 a 10 s. As Figuras abaixo comparam a evolução temporal obtida pelo método híbrido com PINN, comparado aos resultados obtidos utilizando o método Espectral, obtidos por meio dos métodos disponíveis na biblioteca PySpectral.

Para a Posição Esperada $\langle x \rangle$, o método implementado, usando PINN, conseguiu capturar perfeitamente a oscilação e o amortecimento/batimento característico do modelo analisado. O resultado pode ser observado na Figura 1- A, que compara o resultado para o mesmo problema obtido via método espectral, Figura 1 - B.

(A)



(B)

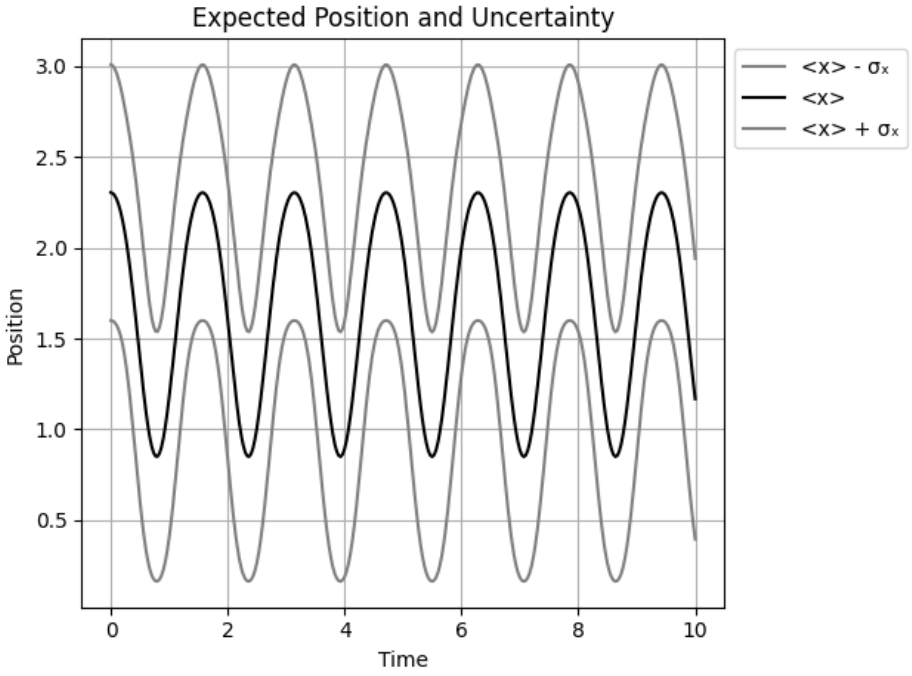


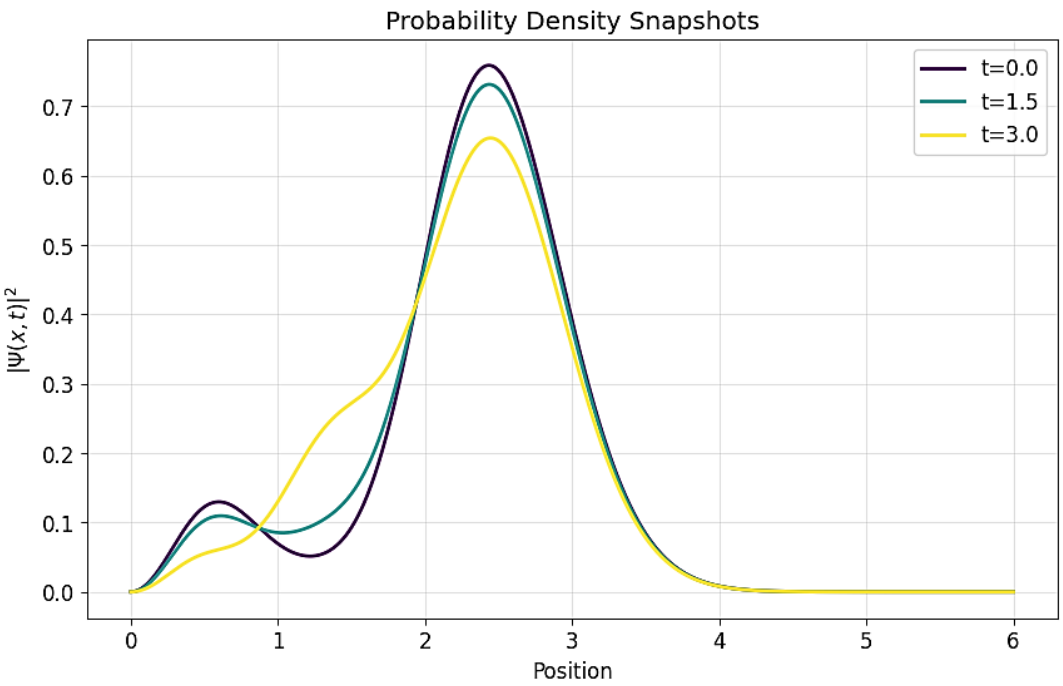
Figura 1 – Resultado obtido para a posição esperada: (A) utilizando o método híbrido com PINN; (B) via método espectral utilizando a biblioteca PySpectral-Quantum-Solver.

A evolução temporal da densidade de probabilidade $|\Psi(x,t)|^2$, apresentada na Figura 2, confirma a capacidade do método híbrido em descrever a localização e a dispersão do pacote de onda. A PINN reproduziu com precisão os padrões de interferência construtiva e destrutiva resultantes da superposição dos estados $n = 0, 1, 2$.

É fundamental destacar que, em uma superposição de estados, a distribuição espacial de probabilidade depende sensivelmente das fases relativas entre as autofunções (devido aos termos cruzados do tipo $2\psi_n\psi_m \cos(\Delta E t)$). A concordância observada entre a PINN (Figura 2 – A) e o método espectral (Figura 2 – B) demonstra o sucesso da estratégia de Gauge Fixing. Ao padronizar os sinais iniciais das autofunções

neurais, garantiu-se que o pacote de onda iniciasse sua trajetória na mesma região do poço de potencial que o *baseline*, eliminando a ambiguidade de espelhamento espacial (*spatial mirroring*) inerente à natureza estocástica do treinamento de redes neurais.

(A)



(B)

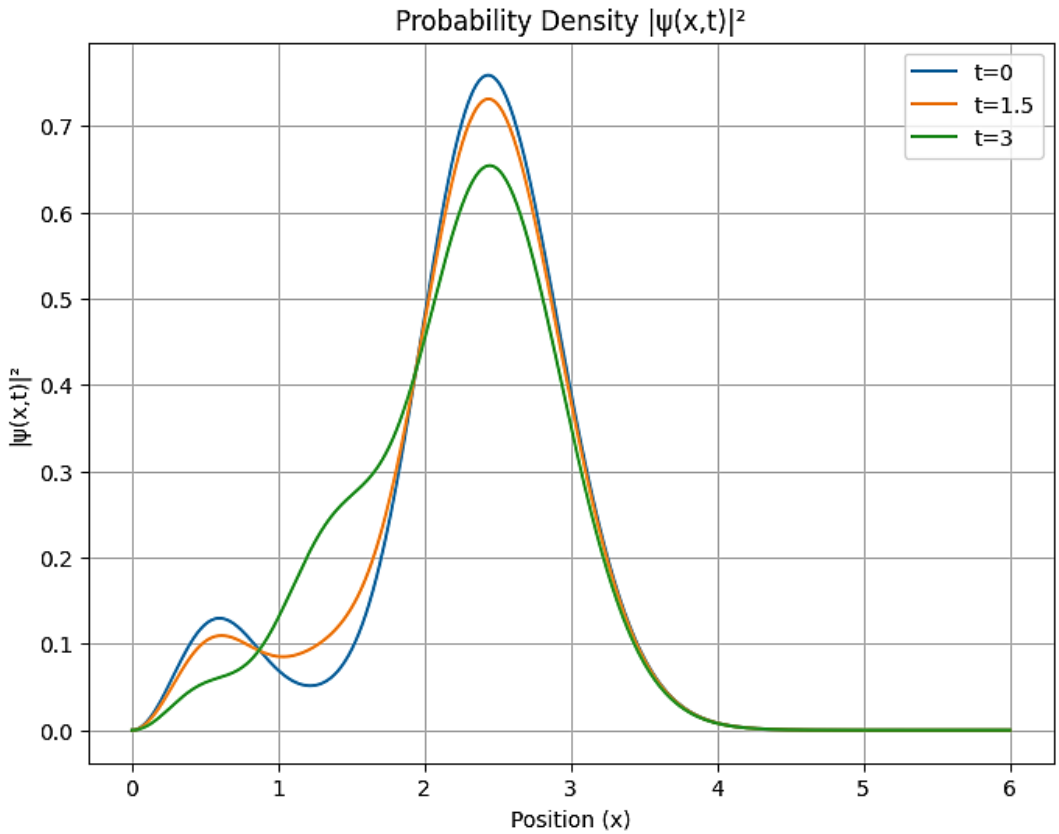
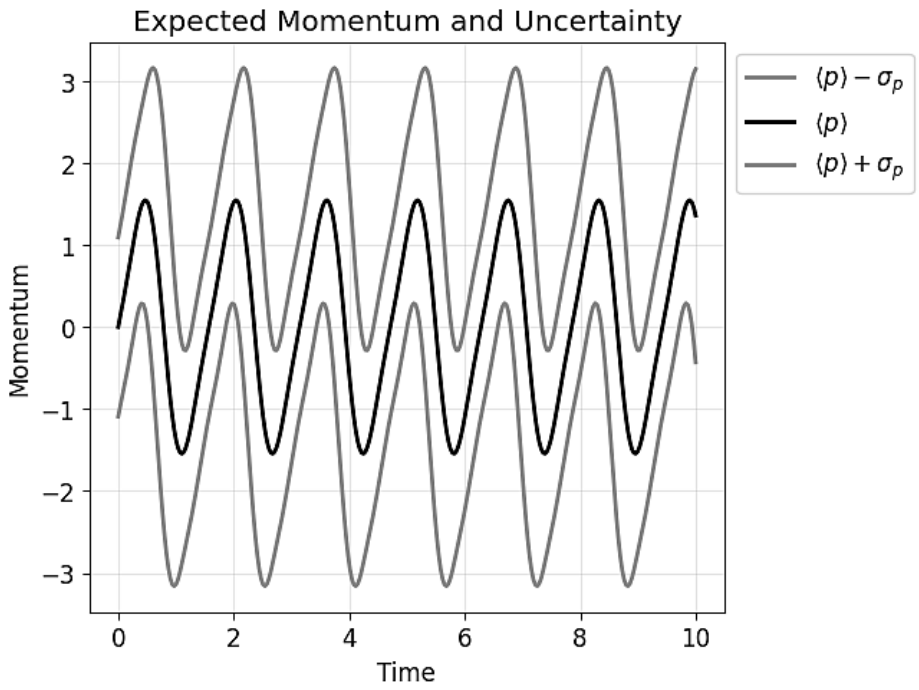


Figura 2 – Evolução da densidade de probabilidade $|\Psi(x,t)|^2$: (A) Simulação via PINN com correção de fase; (B) Simulação de referência via método espectral. Observa-se a identidade nos padrões de batimento e localização do pacote.

Para o momento linear obtemos o seguinte resultado e sua comparação:

(A)



(B)

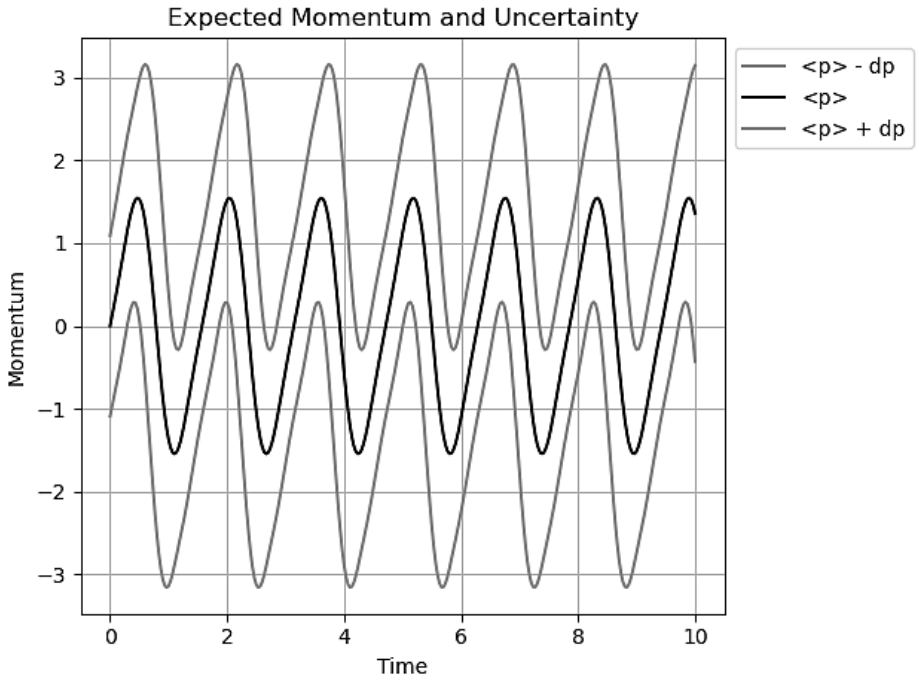
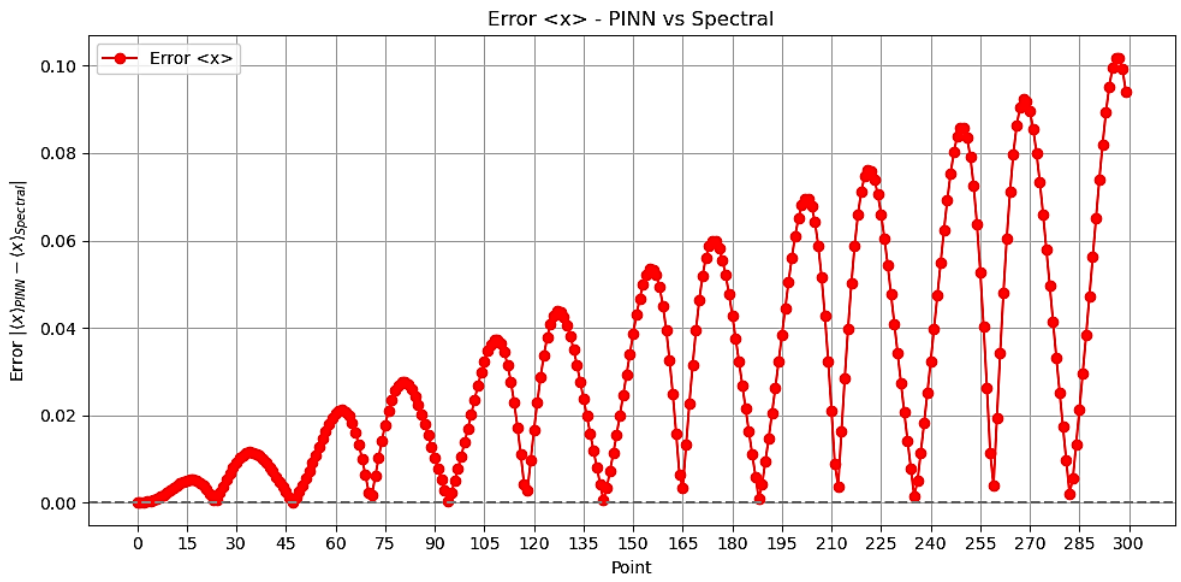


Figura 3 – Resultado obtido para o momento esperado: (A) utilizando o método híbrido com PINN; (B) via método espectral utilizando a biblioteca PySpectral-Quantum-Solver.

A comparação entre os resultados do momento linear, apresentada na Figura 3, revela uma excelente concordância quantitativa entre o método híbrido proposto (A) e o *baseline* espectral (B). Ambas as abordagens exibem o mesmo padrão oscilatório característico, com amplitude variando simetricamente em torno de zero (aprox. ± 1.5 u.a.) e periodicidade idêntica, o que confirma a precisão dos autovalores de energia (E_n) aprendidos pela PINN, visto que a frequência de oscilação depende diretamente das diferenças de energia (ΔE).

Por fim, para uma melhor avaliação numérica dos resultados obtidos, comparamos os valores encontrados pelos métodos no cálculo da posição esperada e do momento, gerando dois gráficos a partir das diferenças nesses valores.

(A)



(B)

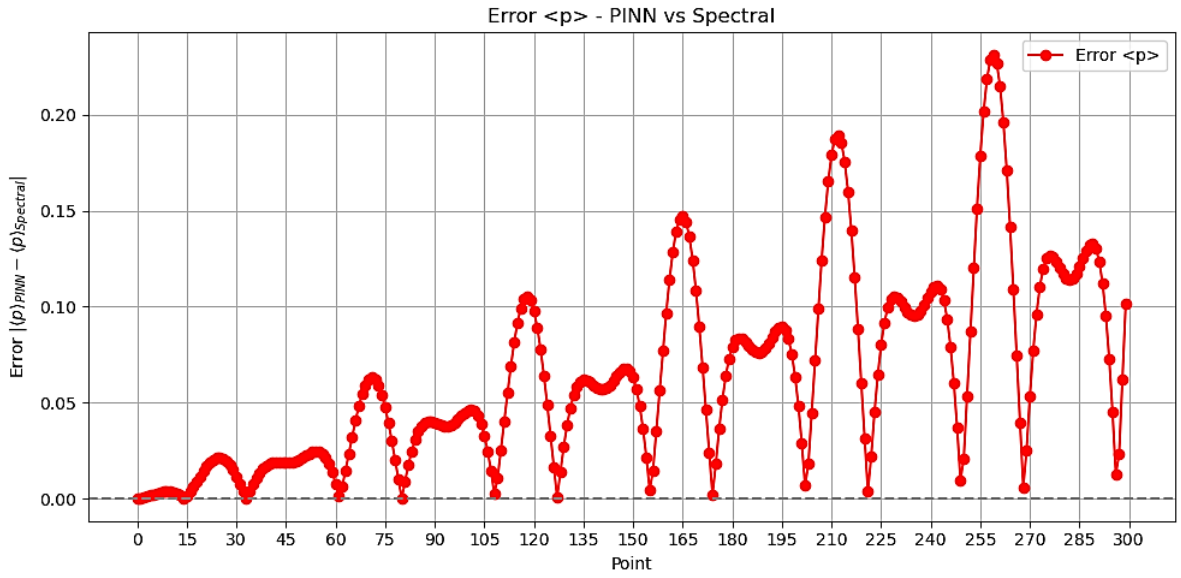


Figura 4 – Diferença dos valores obtidos: (A) para a posição esperada (B) para o momento esperado, utilizando o método PINN e o Espectral.

As figuras comparativas da dinâmica (Figura 4) demonstram que o método híbrido PINN reproduziu com alta fidelidade tanto a amplitude quanto a fase dos observáveis $\langle x \rangle$ e $\langle p \rangle$. A sobreposição das curvas confirma que a rede neural aprendeu corretamente as frequências de oscilação (diferenças de energia ΔE) e a estrutura espacial dos autoestados.

Para quantificar a precisão, analisamos a evolução temporal do erro absoluto em relação ao *baseline* espectral, conforme apresentado nos gráficos de erro. Observa-se um desvio máximo de aproximadamente 0.10 u.a. para a posição e 0.23 u.a. para o momento ao final da simulação ($t = 10$). O comportamento do erro exibe uma envoltória de crescimento linear oscilatório, atribuído à combinação de dois fatores:

1. Desvio de Fase Acumulativo: a dinâmica quântica é regida por fatores de fase e^{-iE_nt}. Diferenças infinitesimais (da ordem de 10^{-4} a 10^{-5}) entre os autovalores de energia preditos pela PINN e os calculados pelo método Espectral resultam em uma defasagem que se acumula proporcionalmente ao tempo ($\Delta\phi \propto \Delta E \cdot t$). Como o momento linear depende sensivelmente da interferência de fases (derivada temporal da fase), é natural que seu erro acumulado seja ligeiramente superior ao da posição.

2. Divergências de Implementação Numérica: embora ambos os métodos utilizem precisão de ponto flutuante, suas bases numéricas são distintas. A PINN utiliza derivação automática (analítica e exata) e integração de alta ordem (Simpson) sobre funções contínuas. Em contrapartida, o método espectral depende de diferenças finitas e integração trapezoidal em grade discreta, introduzindo erros de truncamento. Portanto, parte da discrepância observada não é erro da PINN, mas sim a diferença natural entre uma solução contínua e uma solução discretizada.

Assim, concluímos que o erro relativo se manteve abaixo de 5% para a posição e 7% para o momento em relação à amplitude máxima de oscilação. Isso valida a eficácia da estratégia híbrida (PINN + Mecânica Matricial) para simulações dinâmicas, demonstrando que o método é capaz de preservar as propriedades físicas do sistema (como a unitariedade e a correlação entre posição e momento) sem a necessidade de malhas densas ou passos de tempo numéricos.

4. Conclusão

O presente estudo validou a eficácia das Physics-Informed Neural Networks (PINNs) na resolução de problemas de autovalor em mecânica quântica, alcançando elevada precisão sem a necessidade de malhas de discretização. A contribuição central deste trabalho reside na estratégia híbrida que integra as autofunções espaciais obtidas pela rede neural com o formalismo da Mecânica Matricial.

A implementação proposta destacou-se pela estabilidade numérica ao substituir aproximações tradicionais por cálculos exatos e analíticos. O uso de diferenciação automática (`autograd`) permitiu extrair derivadas baseadas na estrutura interna suave da rede (funções de ativação $Tanh \in C^{\infty}$), superando os erros de truncamento e o ruído inerentes aos métodos de diferenças finitas. Adicionalmente, a precisão dos observáveis foi assegurada pelo pré-cálculo dos elementos de matriz $\langle n|\hat{O}|m\rangle$ via Regra de Simpson de quarta ordem.

Esta abordagem contornou eficazmente as dificuldades computacionais de treinar PINNs dependentes do tempo. Ao adotar uma evolução temporal puramente analítica (e^{-iE_nt}), o método eliminou a dependência de passos de tempo numéricos e a acumulação de erros de propagação. O resultado é uma simulação dinâmica rápida, livre de instabilidades numéricas e fisicamente consistente com a literatura estabelecida.

Embora este trabalho tenha se limitado à aplicação no Oscilador Meio-Harmônico, a metodologia desenvolvida possui alto potencial de generalização. Trabalhos futuros visam a expansão deste *framework* para um algoritmo agnóstico ao potencial, estruturando os métodos de *Gauge Fixing* e dinâmica matricial em uma biblioteca unificada. Isso permitirá que a abordagem híbrida seja aplicada sistematicamente a uma classe mais ampla de problemas quânticos, onde soluções analíticas não estão disponíveis.

5. Referências Bibliográficas

[1] **EISBERG, R.; RESNICK, R. (1985).** *Quantum Physics of Atoms, Molecules, Solids, Nuclei, and Particles* (2nd ed.). John Wiley & Sons.

[2] **GRIFFITHS, D. J.; SCHROETER, D. F. (2018).** *Introduction to Quantum Mechanics* (3rd ed.). Cambridge University Press.

[3] **KINGMA, D. P.; BA, J. (2014).** Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

[4] **LAGARIS, I. E.; LIKAS, A.; FOTIADIS, D. I. (1998).** Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5), 987-1000.

[5] **LIU, D. C.; NOCEDAL, J. (1989).** On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1), 503-528.

[6] **RAISSI, M.; PERDIKARIS, P.; KARNIADAKIS, G. E. (2019).** Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686-707.

[7] **TREFETHEN, L. N. (2000).** *Spectral Methods in MATLAB*. SIAM.