UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE FÍSICA

CURSO DE GRADUAÇÃO EM FÍSICA

JOÃO VICTOR DA SILVA TAVARES

SOLUÇÃO DA EQUAÇÃO DE SCHRODINGER PELO MÉTODO VARIACIONAL COM REDES NEURAIS ARTIFICIAIS

FORTALEZA

2022

JOÃO VICTOR DA SILVA TAVARES

SOLUÇÃO DA EQUAÇÃO DE SCHRODINGER PELO MÉTODO VARIACIONAL COM REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Física.

Orientador: Prof. Dr. Jeanlex Soares de Sousa.

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

T23s Tavares, João Victor da Silva.
Solução da equação de Schrodinger pelo método variacional com redes neurais artificiais / João Victor da Silva Tavares. – 2022.
48 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Física, Fortaleza, 2022.
Orientação: Prof. Dr. Jeanlex Soares de Sousa..

1. Equação de Schrodinger. 2. Método Variacional. 3. Redes Neurais Artificiais. 4. Solução Numérica. I. Título.

CDD 530

JOÃO VICTOR DA SILVA TAVARES

SOLUÇÃO DA EQUAÇÃO DE SCHRODINGER PELO MÉTODO VARIACIONAL COM REDES NEURAIS ARTIFICIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Física.

Aprovada em: 15/12/2022

BANCA EXAMINADORA

Prof. Dr. Jeanlex Soares de Sousa. (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Cláudio Lucas Nunes de Oliveira.
Universidade Federal do Ceará (UFC)

Prof. Dr. Jorge Luís Barbosa de Araújo
Universidade de Fortaleza (Unifor)

Dedico este trabalho à meus pais e à minha avó, que infelizmente não pode estar aqui para me ver um físico formado.

AGRADECIMENTOS

Agradeço aos meus pais, minha mãe Aparecida e meu pai César, que desde sempre me deram total apoio e suporte para que eu pudesse ter o melhor ambiente para estudar, não poderia ter pais melhores e espero honrar todo esforço e dedicação deles a mim. Agradeço ao primo de minha mãe, Claudiomar, a quem considero um tio, que sempre deu apoio moral e financeiro para poder comparecer a universidade, sem ele meu caminha seria muito mais difícil. Agradeço aos bons amigos que me sustentaram durante momentos difíceis e que me orientaram nas difíceis decisões que tive que tomar durante esta jornada. Em especial, gostaria de agradecer a minha amiga e colega Ísis do Vale, pois foi por ela ter me indicado para receber a bolsa de iniciação cientifica que este trabalho pode ser realizado. Por fim agradeço ao Professor Dr. Jeanlex Soares de Sousa pela sua orientação e pela oportunidade de trabalhar no projeto que teve como produto final este trabalho.

"Os homens pedem a seus deuses que provem sua existência com milagres; mas a maravilha eterna é o fato de não haver milagres a todo instante".

(POINCARÉ,1995, p.12)

RESUMO

O presente trabalho tem por objetivo solucionar a equação de Schrodinger independente do tempo através do método variacional, que consiste em utilizar o valor esperado do operador hamiltoniano como um funcional das funções de onda do sistema, minimizando este funcional encontramos as auto-funções. Nesse ponto introduzimos as redes neurais artificiais perceptron multicamada, que podem ser usadas para aproximar uma função real de varias variáveis, este resultado é garantido pelo teorema da aproximação universal. Usamos as redes neurais no funcional do valor esperado de H, e minimizamos este funcional em relação aos parâmetros da rede neural, assim obtendo uma aproximação para cada uma das auto-funções do sistema. Empregamos esse método em sistemas quânticos unidimensionais, o oscilador harmônico e o poço quadrado, encontramos as três primeiras auto-funções destes sistemas e também as suas três primeiras auto-energias. Utilizamos a linguagem de programação Python e suas bibliotecas para implementar computacionalmente o método, e realizar a minimização.Os resultados encontrados foram muito positivos e mostram que o método possui eficiência em solucionar sistemas unidimensionais.

Palavras-chave: equação de Schrodinger; método variacional; redes neurais artificiais; solução numérica

ABSTRACT

This work aims to solve the time-independent Schrodinger equation through the variational method, which consists of using the expected value of the Hamiltonian operator as a functional of the system's wave functions, minimizing this functional to find the eigenfunctions. At this point we introduce multilayer perceptron artificial neural networks, which can be used to approximate a real function of several variables, this result is guaranteed by the universal approximation theorem. We use the neural networks in the functional of the expected value of H, and minimize this functional in relation to the neural network parameters, thus obtaining an approximation for each of the eigenfunctions of the system. We employ this method in one-dimensional quantum systems, the harmonic oscillator and the square well, we find the first three eigenfunctions of these systems and also their first three eigenergies. We used the Python programming language and its libraries to computationally implement the method and perform the minimization. The results found were very positive and show that the method is efficient in solving one-dimensional systems.

Keywords: Schrodinger equation; variational method; Artificial neural networks; numerical solution

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\mathscr{H}$	Espaço de Hilbert
$\mathscr{S}$	Subespaço de um Hilbert
H	Operador Hamiltoniano do sistema.
$\mathfrak{F}$	Funcional valor esperado
$\mathscr{F}$	Conjunto de todas as funções de onda.
$\mathscr{N}$	Rede neural de uma única camada oculta.

SUMÁRIO

1 INTRODUÇÃO

A equação de Schrodinger independente do tempo desempenha papel fundamental na mecânica quântica, na caracterização de diversos sistemas físicos importantes, sendo uma equação de autovalor, conhecer a sua solução completa consiste em conhecer suas auto-energias e auto-funções. O estado físico de um sistema pode ser escrito como uma combinação das auto-funções da equação de Schrodinger (COHEN-TANNOUDJI *et al.*, 2020). Assim todo o trabalho em resolver um sistema quântico e obter suas propriedades relevantes consiste em encontrar estas auto-funções e auto-energias, no entanto poucos sistemas podem ser resolvidos analiticamente, mesmo os mais simples necessitam da introdução de funções especiais para representar as auto-funções. Sendo assim, como métodos de solução analítica não são aplicáveis a sistemas quânticos complexos, os métodos aproximativos aliados à métodos numéricos computacionais são a alternativa mais viável do ponto de vista prático, tendo exito em solucionar uma variedade enorme de sistemas.

Entre os principais métodos aproximativos, um de grande importância teórica é o método variacional, que consiste na minimização de um funcional envolvendo o valor esperado do operador hamiltoniano, através dele é possível obter com grande precisão as primeiras auto-energias do sistema. Entretanto este método geralmente exige a suposição a priori da forma funcional da auto-função, geralmente uma função elementar conhecida dependente de algum parâmetro ajustável, essa limitação é parcialmente eliminada com a utilização de redes neurais artificias de camada única na representação da auto-função, com seus pesos sinápticos se tornando os parâmetros que devem ser ajustados na minimização.

O capitulo 2 discute aspectos teóricos envolvendo o método variacional e uma breve introdução às redes neurais artificiais, em especial a sua utilização na representação de funções reais de varias variáveis. O capitulo 3 apresenta um método computacional que une os conceitos teóricos apresentados no capitulo 2 para a resolução de sistemas quânticos complexos. No capitulo 4 o método será usado na resolução de sistemas quânticos simples que possuem solução analítica, o potencial poço quadrado infinito e o oscilador harmônico, para que as soluções aproximadas possam ser comparadas com as soluções analíticas a fim de verificar a eficiência do método apresentado no capitulo anterior. Por fim, o capitulo 5 apresenta uma discussão sobre a viabilidade da aplicação desse método em sistemas mais complexos.

2 FUNDAMENTOS

2.1 O método variacional

Os métodos variacionais desempenham grande papel na formulação teórica das teorias físicas, por exemplo na mecânica analítica com as integrais de ação ou na termodinâmica com a entropia. A obtenção do estado físico do sistema consiste na obtenção de um ponto de extremo relativo a essas funções (CALLEN, 1985). Em geral, a ideia é supor a existência de um **funcional** que cumpra duas condições, a primeira é que o funcional seja função de um certo conjunto que entre os seus elementos se encontre os **estados físicos** do sistema, e a segunda é que esses estados físicos sejam os **pontos de extremo** (máximos ou mínimos) desse funcional.

2.1.1 Funcional para o estado fundamental

Na mecânica quântica existe um funcional que obedece essas condições, porém ele fornece apenas auto-funções do sistema. Para obter este funcional, inicialmente partimos do **espaço de Hilbert** geral $\mathscr{H}$ que representa os vetores de estado físico do sistema, a partir da representação de posição dos aūto-estados obtemos as auto-funções (COHEN-TANNOUDJI *et al.*, 2020). Seja um sistema quântico que possui um operador hamiltoniano dado por H, os auto-estados do sistema são as soluções da seguinte equação de autovalor:

$$H|\psi\rangle = E|\psi\rangle \tag{2.1}$$

Representando os autovetores que solucionam (2.1) por $|n\rangle$, podemos escrever a solução geral $|\psi\rangle$ como a uma combinação linear dos seus auto-estados:

$$|\psi\rangle = \sum_{n=0}^{\infty} c_n|n\rangle \tag{2.2}$$

Usando a forma do estado físico $|\psi\rangle$ dado pela forma (2.2). o valor esperado do hamiltoniano H é calculado e encontramos as seguintes relações:

$$\langle H\rangle = \langle\psi|H|\psi\rangle = \sum_{ij} c_i^* c_j \langle i|H|j\rangle = \sum_{ij} c_i^* c_j E_j \delta_{ij} = \sum_{n=0}^{\infty} |c_n|^2 E_n \tag{2.3}$$

Sabendo que a a energia do estado fundamental do sistema é a menor de todas as auto-energias, temos $E_0 \leq E_n \; \forall n$ e portanto:

$$\langle H\rangle = \sum_{n=0}^{\infty} |c_n|^2 E_n \geq (\sum_{n=0}^{\infty} |c_n|^2) E_0 \tag{2.4}$$

A norma ao quadrado do vetor de estado $|\psi\rangle$ é dada por $\sum_{n=0}^{\infty}|c_n|^2$ que é justamente o termo presente do lado direito da desigualdade (2.4), assim adicionamos a condição $\||\psi\rangle\|^2 = \sum_{n=0}^{\infty}|c_n|^2 = 1$ ou seja que o vetor $|\psi\rangle$ deve ser normalizado. Portanto temos:

$$\langle H\rangle \geq E_0 \tag{2.5}$$

O valor esperado de um determinado operador na mecânica depende unicamente do vetor de estado $|\psi\rangle$ usado no seu calulo, assim podemos encarar o valor esperado como um funcional que depende dos vetores de estado. Assim definimos este funcional:

$$\mathfrak{F}[|\psi\rangle] = \langle\psi|H|\psi\rangle \tag{2.6}$$

e sabendo que vale a condição de que $\||n\rangle\|^2 = 1$ (os auto-estado também são normalizados) as auto-energias podem ser escritas da seguinte forma $E_n = \langle n|H|n\rangle$, e assim temos que $E_0 = \langle 0|H|0\rangle = \mathfrak{F}[|0\rangle]$ e podemos reescrever a desigualdade (2.5):

$$\mathfrak{F}[|\psi\rangle] \geq \mathfrak{F}[|0\rangle] \tag{2.7}$$

Esta desigualdade atesta uma importante propriedade do funcional de valor esperado, no entanto outro resultado sobre este funcional nos permite tirar conclusões ainda mais profundas, este é o chamado **teorema de Ritz** (COHEN-TANNOUDJI *et al.*, 2020).

Teorema 1 (de Ritz) *Seja um funcional $\mathfrak{F} : \mathscr{H} \longrightarrow \mathbb{R}$ definido como o de (2.6), então os auto-vetores do operador hamiltoniano H são pontos extremos do funcional $\mathfrak{F}[|\psi\rangle]$.*

Demonstração: (consultar **APÊNDICE A**)

Concluímos a partir da desigualdade (2.7) e do teorema de Ritz que o funcional $\mathfrak{F}$ possui um mínimo global no estado fundamental do sistema quando os vetores de estado que servem como seus argumentos estão sujeitos a uma condição de normalização. Assim este funcional (o valor esperado do hamiltoniano H) é o funcional que cumpre as duas condições mencionadas no inicio da seção e portanto, para encontrar o estado fundamental do sistema, basta minimizar o funcional definido em (2.6) vinculado a condição de normalização. Aqui reside a maior virtude do método variacional, que é encontrar a energia do estado fundamental de qualquer sistema quântico que possua um espectro de energia discreto.

2.1.2 Extensão para os demais auto-estados

Apesar de grande parte de suas aplicações serem a respeito do estado fundamental, o método variacional não se limita apenas a ele, podendo ser estendido para os demais auto-estados do sistema, e é o que desenvolveremos a seguir. A mecânica quântica exige que os autovetores de um determinado **observável** sejam todos ortogonais entre si, assim formando um conjunto completo e servindo para representar o estado físico do sistema. Sejam os auto-estados do operador hamiltoniano H sendo representados por $\{|0\rangle, |1\rangle, ..., |k\rangle, ...\}$ se tomarmos os vetores $\{|k\rangle, |k+1\rangle, ...\}$ vemos que estes formam um conjunto completo para um novo espaço vetorial $\mathscr{S}$ que é subespaço do espaço de Hilbert original $\mathscr{H}$, assim definimos $\mathscr{S}$ como:

$$\mathscr{S} = \{|\psi\rangle \in \mathscr{H} | \text{ tal que } |\psi\rangle = \sum_{n=k}^{\infty} c_n |n\rangle\}$$

assim a ortogonalidade dos auto-estados implica um importante resultado, se tivermos $|\psi\rangle \in \mathscr{S}$ então para o conjunto de auto-estados restante $\{|0\rangle, |1\rangle, ..., |k-1\rangle\}$ teremos as seguintes condições de ortogonalidade:

$$\langle\psi|0\rangle = 0$$

$$\langle\psi|1\rangle = 0$$

$$...$$

$$\langle\psi|k-1\rangle = 0$$

Agora vamos calcular o valor esperado para vetores que pertencem ao subespaço $\mathscr{S}$ seguindo o mesmo raciocino da subseção anterior, assim sendo $|\psi\rangle \in \mathscr{S}$ temos:

$$\langle H\rangle = \langle\psi|H|\psi\rangle = \sum_{ij=k} c_i^* c_j \langle i|H|j\rangle = \sum_{ij=k} c_i^* c_j E_j \delta_{ij} = \sum_{n=k}^{\infty} |c_n|^2 E_n \tag{2.8}$$

a menor energia para o sistema restrito a esse subespaço é E_k, desta forma teremos $E_K \leq E_n \ \forall n$ e portanto:

$$\langle H\rangle = \sum_{n=k}^{\infty} |c_n|^2 E_n \geq (\sum_{n=k}^{\infty} |c_n|^2) E_k \tag{2.9}$$

Além disso, assim como no caso do estado fundamental, também impomos uma condição de normalização aos vetores $|\psi\rangle$ o que implica que $\||\psi\rangle\|^2 = \sum_{n=k}^{\infty} |c_n|^2 = 1$.

Por fim, com a condição de normalização e a desigualdade (2.9) obtemos uma nova desigualdade muito semelhante a desigualdade (2.5):

$$\langle H \rangle \geq E_k \tag{2.10}$$

O funcional $\mathfrak{F}$ pode ser definido também para esta desigualdade, porém o seu domínio deve estar restrito ao subespaço $\mathscr{S}$, essa condição pode ser cumprida simplesmente impondo as condições de ortogonalidade $\{\langle \psi|0\rangle = 0, \langle \psi|1\rangle = 0, \langle \psi|2\rangle = 0, ..., \langle \psi|k-1\rangle = 0\}$ já que se um certo vetor $|\psi\rangle$ as cumpre, necessariamente ele deve pertencer ao subespaço $\mathscr{S}$, portando chagamos a seguinte desigualdade:

$$\langle H \rangle \geq E_k = \langle k|H|k\rangle \Longrightarrow \mathfrak{F}[|\psi\rangle] \geq \mathfrak{F}[|k\rangle] \text{ se } |\phi\rangle \in \mathscr{S}$$

$$\mathfrak{F}[|\psi\rangle] \geq \mathfrak{F}[|k\rangle] \tag{2.11}$$

A desigualdade (2.11), assim como a desigualdade (2.7), estabelece que o auto-estado $|k\rangle$ é o que minimiza o funcional, e o teorema de Ritz garante que este auto-estado é um ponto de extremo do funcional, portanto concluímos que este ponto é um mínimo global. Assim, para determinar $|k\rangle$ basta minimizar o funcional $\mathfrak{F}$ sujeito as k condições de ortogonalidade $\{\langle \psi|0\rangle = 0, \langle \psi|1\rangle = 0, \langle \psi|2\rangle = 0, ..., \langle \psi|k-1\rangle = 0\}$ e a condição de normalização $\||\psi\rangle\|^2 = 1$.

Por fim, encontramos um procedimento recursivo para encontrar todos os auto-estados do sistema, se desejamos encontrar o auto-estado correspondente ao k-ésimo nível de energia, basta conhecermos os k auto-estados anteriores e usar estes para encontrar as k condições de vinculo, em seguida minimizamos o funcional $\mathfrak{F}$ sujeito a essas condições de vinculo com a condição de normalização.

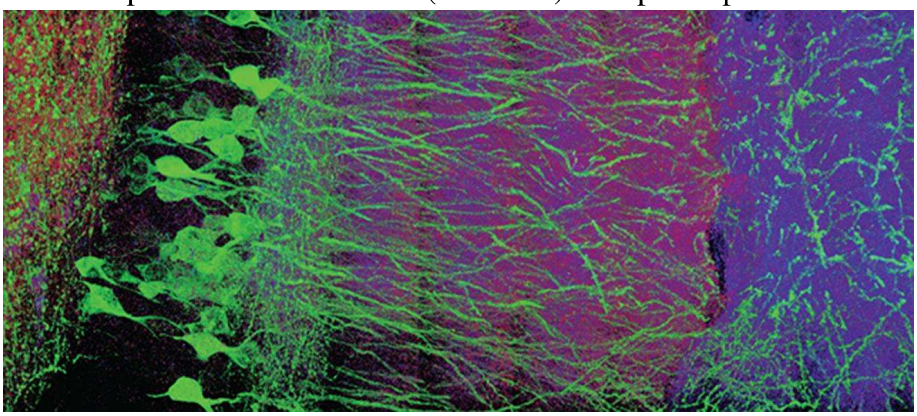
Apesar do caráter geral do procedimento descrito nesta seção, ele possui algumas limitações. A primeira delas e a respeito da natureza do espectro de energia do sistema, que e necessariamente discreto, como foi assumido implicitamente nesta seção. A segunda limitação é sobre a degenerescência dos auto-estados de energia. Se ela ocorre, então para um mesmo valor de energia temos mais de um auto-estado associado, e em vez de um único mínimo global o funcional terá vários. Isso não necessariamente inviabiliza o método mas gera uma serie de dificuldades, o que explica a preferencia de sua aplicação no estado fundamental em vez dos demais auto-estados.

2.2 Redes neurais

2.2.1 Redes neurais biológicas

Durante o curso da evolução das espécies, os seres vivos se tornaram cada vez mais complexos e desenvolveram novas aptidões que lhes permitiram sobreviver às adversidades, muitas vezes complexas, no ambiente em que viviam. Uma dessas aptidões se destaca, se tornando fundamental na sobrevivência dos seres vivos complexos, a capacidade de aprender e de exibir respostas complexas baseadas nesse aprendizado. Tal mecanismo dependeu em grande parte do surgimento das redes neurais biológicas, estruturas que davam ao ser vivo a capacidade de aprender e a reagir a eventos no ambiente de forma complexa e que maximizam suas chances de sobrevivência e reprodução. Existe uma variedade enorme de tipos de redes neurais biológicas na natureza, um em especial é o que compõe o sistema nervoso central dos seres humanos, que é formado por uma variedade de células especializadas, entre elas uma que cumpre papel de destaque é o neurônio.

Figura 1 – Microscopia de uma rede neural (em verde) no hipocampo de um camundongo.

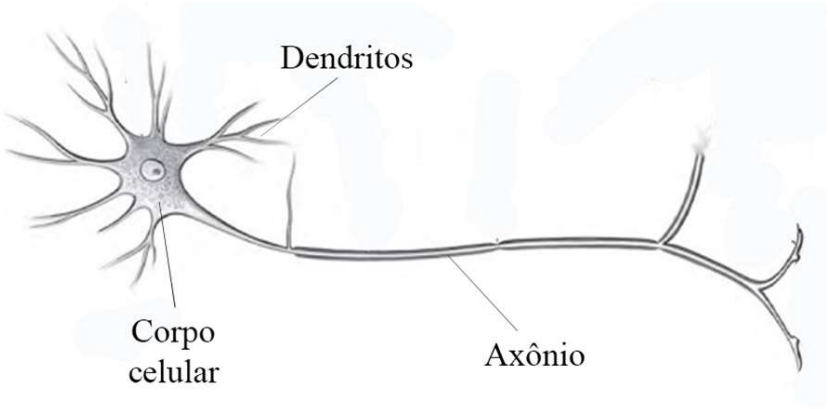


Fonte: Adaptado de (CHEN *et al.*, 2015)

Os neurônios são o principal tipo de celular que compõem o sistema nervoso central humano, eles têm a função de receber e transmitir informações através de substâncias químicas chamadas neurotransmissores. A estrutura básica de um neurônio é composta por três partes principais, o axônio, o corpo celular e os dendritos (HAYKIN, 1998). Os componentes e o funcionamento real de uma rede neural biológica são altamente complexos e por isso a descrição a seguir busca capturar apenas os aspectos gerais mais relevantes do funcionamento desse sistema.

O corpo celular é a parte principal do neurônio, contendo o núcleo e organelas celulares importantes. Os detritos são as estruturas responsáveis por receber os sinais de outros neurônios, o sinal é recebido pelo dendrito podendo ser atenuado ou amplificado por ele, a intensidade com que isso ocorre é uma propriedade do dendrito denominada peso sináptico, cada neurônio geralmente pode receber sinais de vários neurônios através de seus dendritos. Por fim temos o axônio, que recebe os sinais vindos dos vários dendritos do neurônio através do corpo celular, todos os sinais são unidos e enviados pelo axônio até o dendrito de outro neurônio (LUDWIG; MONTGOMERY, 2007).

Figura 2 – Principais estruturas de um neurônio biológico.



Fonte: Adaptado de (NOBACK *et al.*, 2005)

Os neurônios se conectam uns com os outros seguindo a descrição de funcionamento do parágrafo anterior, assim eles formam a rede neural, onde a rede recebe um estímulo ou conjunto de sinais de entrada e processa esses sinais produzindo um novo conjunto de sinais ao final. A maneira como a rede processa esses sinais está diretamente relacionada com os pesos sinápticos dos dendritos presentes nos neurônios da rede, modificando esses pesos, a maneira como a rede lida com o estímulo muda, desta forma, esse processo de modificação dos pesos sinápticos capacita a rede a aprender a lidar adequadamente com um determinado estímulo. A modificação frequente dos pesos sinápticos é o que possibilita o aprendizado e o armazenamento de informações nas redes neurais, e é justamente nesse processo que reside todo o poder das redes neurais biológicas.

2.2.2 Redes neurais artificiais

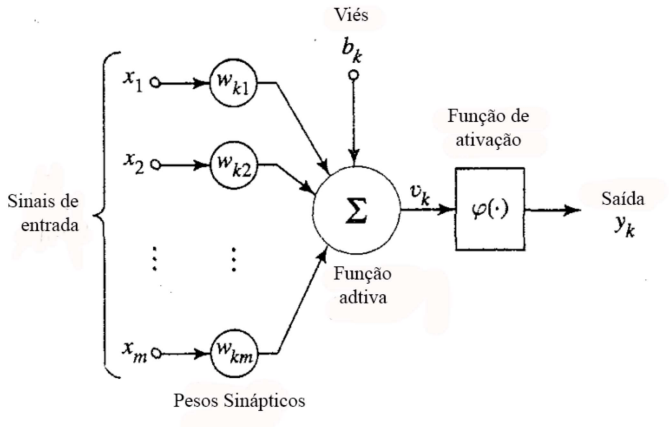
A eficiência das redes neurais biológicas em lidar com problemas complexos motivou vários modelos matemáticos que buscavam replicar o seu funcionamento computacionalmente,

assim possibilitando a sua utilização na resolução de problemas em que algoritmos convencionais se mostram pouco eficientes (LUDWIG; MONTGOMERY, 2007). Vários modelos foram teorizados, porém o limitado poder de computação da época não permitiu grandes aplicações práticas, porém, com o rápido crescimento do poder de computação no início do século a importância e aplicações das redes neurais artificiais se expandiram exponencialmente.

2.2.3 Perceptron

Para simular uma rede neural biológica primeiro é necessário elaborar um modelo matemático para seus componentes principais, os neurônios, assim surge a ideia do neurônio matemático, o **perceptron**. Basicamente, o perceptron busca replicar de forma muito simples o processo de funcionamento dos neurônios biológicos, que foi descrito na seção anterior. Inicialmente temos um conjunto de n números reais x_i $(i \in \{1,...,n\})$, estes são os sinais de entrada, cada um desses números é multiplicado por um outro número w_i $(i \in \{1,...,n\})$ que pertence a um conjunto de números que são chamados de pesos sinápticos, em seguida todos esses produtos são somados e é somado também um número b conhecido como "viés"ou Bias. No fim, o valor obtido dessa combinação linear é submetido a uma função real $\phi(v)$ que é chamada de função de ativação, existem vários tipos de funções de ativação, cada uma sendo mais conveniente dependendo do tipo de aplicação que a rede neural terá. Assim, o perceptron simula o processo que ocorre no neurônio como uma combinação linear dos sinais de entrada, ponderados pelos pesos sinápticos, que assim como nos neurônios biológicos, tem o papel de atenuar ou amplificar o sinal de entrada.

Figura 3 – Modelo de um neurônio quando a função aditiva é uma combinação linear.



Fonte: Adaptado de (HAYKIN, 1998)

A escolha da função de ativação é o primeiro passo na elaboração de uma rede neural, neste trabalho a função de ativação que adotaremos é a função sigmoide $\sigma(x)$, amplamente utilizada em diversas arquiteturas de redes neurais, a sua utilização se deve a um importante teorema que será apresentado mais adiante, o teorema da aproximação universal.

$$\sigma(v) = \frac{1}{1+\exp(-v)}$$

A função sigmoide é uma função continua com valores de sua imagem estando limitados entre 0 e 1, sendo 1 no limite $x \to \infty$ e sendo 0 para $x \to -\infty$, essas propriedades tornam $\sigma(x)$ uma função de probabilidade acumulada. Assim, a atuação da função sigmoide possui uma interpretação, ao receber um determinado valor combinado dos sinais de entrada ela avalia a probabilidade do neurônio estar "ativado", quanto mais positivo for o sinal mais próximo o valor da função estará de 1 e consequentemente maior a sua probabilidade de estar ativado e vice versa. Assim, seja um conjunto de valores de entrada x_i $(i \in \{1,...,n\})$, um conjunto de pesos w_i $(i \in \{1,...,n\})$ e um viés b, a representação matemática de um perceptron é dada pela equação:

$$\sigma(\sum_{i=1}^{n} x_i w_i + b) = y \tag{2.12}$$

2.2.4 Perceptron multicamada

O modelo do perceptron (por simplicidade, chamaremos também os perceptrons de neurônios) permite construir uma rede neural a partir da associação de vários neurônios em camadas, a rede resultante desse processo é chamada de rede perceptron multicamada (de agora em diante sempre que nos referirmos a uma rede neural estaremos nos referindo a uma rede perceptron multicamada). Seja um conjunto de dados de entrada xi $i \in \{1,...,n\}$, consideremos também um conjunto de m perceptrons, como o conjunto de pesos e vieses define unicamente um perceptron então para cada j $j \in \{1,...,m\}$ existe um conjunto de n pesos sinápticos $\{w_{j1}, w_{j2}, ..., w_{jn}\}$ e um viés b_j de tal forma que estes definem o j-ésimo perceptron do conjunto. Assim, as m equações que representam os perceptrons são:

$$\sigma(\sum_{i=1}^{n} x_i w_{ji} + b_j) = y_j \quad \text{para cada} \quad j \in \{1,...,m\} \tag{2.13}$$

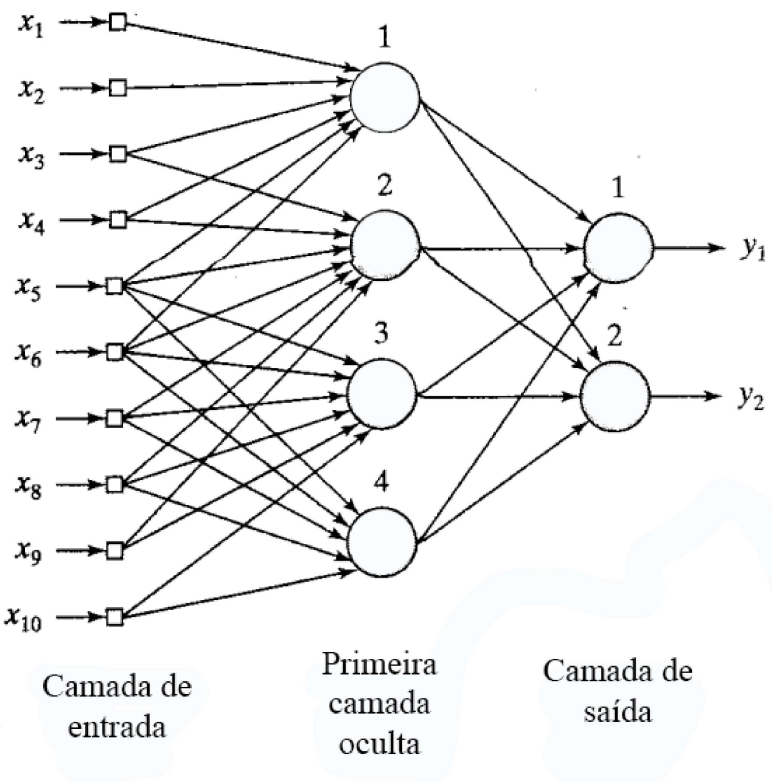
O conjunto dos m sinais de saída $\{y_1,...,y_m\}$ pode ser encarado como um conjunto de sinais de entrada para um outro conjunto de k perceptrons, para cada l, com $l \in \{1,...,k\}$,

existe um perceptron com um conjunto de pesos $\{\alpha_{l1}, \alpha_{l2}, ..., \alpha_{lm}\}$ e um viés γ_l que gera um novo conjunto de sinais de saída z_l:

$$\sigma(\sum_{j=1}^{m} \alpha_{lj}\sigma(\sum_{i=1}^{n} x_i w_{ji} + b_j) + \gamma_j) = z_l \quad \text{para cada} \quad l \in \{1, ..., k\} \tag{2.14}$$

Este processo é repetido sucessivamente um número finito de vezes até que, ao final, um último sinal de saída seja gerado, toda a estrutura que compõe esse processo é a rede neural perceptron multicamada. Cada um dos conjuntos de sinais de entrada envolvidos no processo da rede compõe o que é chamado de **camada**, o conjunto de valores inicias da rede forma a **camada de entrada**, já todos os conjuntos de sinais que são saída de alguma conjunto de perceptrons compõe a**camada oculta**, a exceção a essa regra são os últimos sinais de saída, que compõe a **camada de saída** (LUDWIG; MONTGOMERY, 2007). As camadas ocultas e a de saída possuem perceptrons, a configuração de pesos sinápticos e de vieses de cada um deles determina como a rede lida com os sinais de entrada, assim, se desejamos utilizar a rede para lidar com os sinais de uma determinada maneira, devemos modificar os pesos até que se atinja o objetivo desejado.

Figura 4 – Rede neural perceptron multicamada com apenas uma camada oculta.



Fonte: Adaptado de (HAYKIN, 1998)

2.2.5 Teorema da aproximação universal

Nesta seção desenvolveremos um dos resultados mais poderosos envolvendo a teoria de redes neurais, o **teorema da aproximação universal**. Seja uma rede neural com n sinais de entrada e apenas uma camada oculta com N neurônios, suponhamos também que ela possua apenas um neurônio na ultima camada e que este não tem seu valor submetido a função de ativação sigmoide e nem é acrescido com um valor de viés. Sem ter seu sinal de saída submetido a função sigmoide, ele pode assumir qualquer valor real, assim, esta **arquitetura** de rede pode ser encarada como uma **função real de n variáveis**. Assim, a rede é representada da seguinte forma:

$$\mathscr{N}(\vec{r},\vec{a}) = \sum_{j=1}^{N} \alpha_j \sigma(\sum_{i=1}^{n} x_i w_{ji} + b_j) \tag{2.15}$$

Onde $\vec{r} \in \mathbb{R}^n$ e o vetor $\vec{a} = (\alpha_j, w_{ji}, b_j)$ é composto por todos os valores de pesos e vieses da rede, assim uma vez fixado o numero N de neurônios da camada oculta, temos que a rede fica unicamente determinada se os valores do vetor $\vec{a}$ forem especificados.

Teorema 2 *Seja A um conjunto compacto em $\mathbb{R}^n$ e $\phi : A \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ uma função limitada e diferenciável em A. Para todo $\varepsilon > 0$ existem N e $\vec{a}$ tais que:*

$$|\phi(\vec{r}) - \mathscr{N}(\vec{r},\vec{a})| < \varepsilon$$

Este é o **teorema da aproximação universal** que basicamente afirma que qualquer função real em $\mathbb{R}^n$ pode ser aproximada por uma rede neural. Basta especificar um "erro"= ε entre a função que se deseja aproximar e a rede, e o teorema garante que é sempre possível encontrar parâmetros N e $\vec{a}$ que tornem esse erro menor que ε (HAYKIN, 1998).

Este é um teorema de existência, que apresenta apenas uma condição **suficiente** para que a aproximação possa ser realizada, no caso, que haja no mínimo uma camada oculta. No entanto, redes com mais camadas ocultas também podem ser usadas para aproximar funções (HAYKIN, 1998). A natureza das funções que podem ser aproximadas por uma rede neural deste tipo possibilita a utilização deste teorema em vários métodos numéricos que evolvem a resolução de equações diferenciais e problemas de otimização.

3 METODOLOGIA

No presente trabalho, vamos tratar de problemas envolvendo **sistemas de uma partícula**, embora o método também possa ser utilizado em sistemas de múltiplas partículas. Assim, propomos a utilização de uma rede neural do tipo $\mathscr{N}(\vec{r},\vec{a})$ para representar a função de onda na equação de Schrodinger independente do tempo (2.1). Em seguida utilizamos o método variacional para minimizar o funcional $\mathfrak{F}$ com respeito aos pesos e vieses representados pelo vetor $\vec{a}$ sujeito as condições de ortogonalidade e normalização descritas na seção (2.1.2). O procedimento em detalhes é descrito a seguir.

Seja um sistema quântico de uma partícula de massa m submetida a um potencial $V(\vec{r})$, o funcional $\mathfrak{F}$ é o valor esperado do operador hamiltoniano H, portando pode ser escrito em forma de uma integral que envolve a **representação de posição** de $|\psi\rangle$ e H, assim, temos:

$$\mathfrak{F}[|\psi\rangle] = \langle\psi|H|\psi\rangle = \int \psi(\vec{r})^* H\psi(\vec{r})d\vec{r} = \mathfrak{F}[\psi(\vec{r})] \tag{3.1}$$

Onde:

$$H = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})$$

A representação integral nos mostra que o funcional depende apenas da função de onda $\psi(\vec{r})$, assim, seja $\mathscr{F}$ o conjunto de todas as funções $\psi : \mathbb{R}^n \longrightarrow \mathbb{C}$ que são funções de onda, definimos o funcional $\mathfrak{F} : \mathscr{F} \longrightarrow \mathbb{R}$ através da integral presente em (3.1). Para cada $|\psi\rangle$ existe uma única representação de posição $\psi(\vec{r})$, portanto minimizar o funcional com respeito ao espaço de funções $\mathscr{F}$ é equivalente a minimiza-lo com respeito ao espaço de *Hilbert* $\mathscr{H}$ (e eventualmente, com respeito a seus subespaços $\mathscr{S}$), pois se $|\psi\rangle$ é um ponto de mínimo então $\psi(\vec{r})$ também deverá ser um ponto de mínimo de $\mathfrak{F}$ com respeito ao espaço $\mathscr{F}$ (COHEN-TANNOUDJI *et al.*, 2020).

Sendo $\psi(\vec{r})$ a função que desejamos encontrar, seja também $\psi_n(\vec{r})$ a n-ésima auto-função do operador H, na representação de posição, as relações de ortogonalidade $\{\langle\psi|0\rangle = 0, \langle\psi|1\rangle = 0, \langle\psi|2\rangle = 0, ..., \langle\psi|k-1\rangle = 0\}$ e de normalização $\||\psi\rangle\|^2 = 1$ se tornam integrais da forma:

$$\langle\psi|i\rangle = \int \psi(\vec{r})^* \psi_i(\vec{r})d\vec{r} = 0 \quad \text{para cada} \quad i \in \{0, ..., k-1\} \tag{3.2}$$

$$\||\psi\rangle\|^2 = \int \psi(\vec{r})^* \psi(\vec{r})d\vec{r} = 1 \tag{3.3}$$

Na seção (2.2.5), vimos que uma rede neural do tipo $\mathscr{N}(\vec{r},\vec{a})$ pode ser usada para aproximar qualquer função que atenda as exigências do teorema da aproximação universal. As funções de onda de $\mathscr{F}$ são funções complexas, porém, numa região limitada do seu domínio suas componentes real e imaginaria cumprem essas exigências, uma vez que devem ser continuais, diferenciáveis e também devem ser quadrado-integráveis (COHEN-TANNOUDJI *et al.*, 2020). Assim, as componentes de uma função de onda $\psi(\vec{r})$ podem ser representadas por duas redes neurais da seguinte maneira:

$$\psi(\vec{r}) = \mathscr{N}_1 + i\mathscr{N}_2 = (\mathscr{N}_1, \mathscr{N}_2) \tag{3.4}$$

onde:

$$\mathscr{N}_1 = \mathscr{N}(\vec{r},\vec{a}_1)$$

$$\mathscr{N}_2 = \mathscr{N}(\vec{r},\vec{a}_2)$$

Assim como foi discutido na seção (2.2.5) a rede neural depende unicamente do seu conjunto de pesos e vieses $\vec{a}_1$ e $\vec{a}_2$ para ser caracterizada, portanto para encontrar uma configuração de pesos e vieses que aproxime a função de onda basta minimizar o funcional em relação a esses vetores. Substituindo a relação (3.4) na integral do funcional em (3.1) temos:

$$\int \psi(\vec{r})^* H\psi(\vec{r})d\vec{r} = \int (\mathscr{N}_1 - i\mathscr{N}_2)(H\mathscr{N}_1 + iH\mathscr{N}_2)d\vec{r}$$

$$\Rightarrow \int [(\mathscr{N}_1 H\mathscr{N}_1 + \mathscr{N}_2 H\mathscr{N}_2) - i(\mathscr{N}_1 H\mathscr{N}_2 - \mathscr{N}_2 H\mathscr{N}_1)]d\vec{r}$$

$$\Rightarrow \int (\mathscr{N}_1 H\mathscr{N}_1 + \mathscr{N}_2 H\mathscr{N}_2)d\vec{r} - i\int (\mathscr{N}_1 H\mathscr{N}_2 - \mathscr{N}_2 H\mathscr{N}_1)d\vec{r}$$

A parte imaginaria do funcional deve ser nula, pois a integral deve ser um numero real, portanto temos:

$$\int (\mathscr{N}_1 H\mathscr{N}_2 - \mathscr{N}_2 H\mathscr{N}_1)d\vec{r} = 0$$

A integral que resulta depende unicamente das duas redes neurais, e portanto, depende unicamente dos dois vetores $\vec{a}_1$ e $\vec{a}_2$, definindo $\vec{a} = (\vec{a}_1, \vec{a}_2)$, temos:

$$\mathfrak{F}(\vec{a}) = \int (\mathscr{N}_1 H\mathscr{N}_1 + \mathscr{N}_2 H\mathscr{N}_2)d\vec{r} \tag{3.5}$$

Para realizar o calculo da integral do operador em (3.5) devemos conhecer analiticamente a atuação do operador H sobre a rede neural. O operador atua separadamente em cada componente da função de onda $\psi(\vec{r})$, assim temos:

$$H\psi(\vec{r}) = H\mathcal{N}_1 + iH\mathcal{N}_2$$

Onde:

$$H\mathcal{N}_1 = (-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r}))\mathcal{N}_1 = -\frac{\hbar^2}{2m}\sum_{v=1}^{n}\frac{\partial^2\mathcal{N}_1}{\partial x_v^2} + V(\vec{r})\mathcal{N}_1$$

$$H\mathcal{N}_2 = (-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r}))\mathcal{N}_2 = -\frac{\hbar^2}{2m}\sum_{v=1}^{n}\frac{\partial^2\mathcal{N}_2}{\partial x_v^2} + V(\vec{r})\mathcal{N}_2$$

Escrevendo a rede neural na forma (2.15) e lembrando que $\vec{r} = (x_1, ..., x_n)$, é possível calcular diretamente as suas derivadas parciais de qualquer ordem.

$$\frac{\partial^2\mathcal{N}}{\partial x_v^2} = \frac{\partial^2}{\partial x_v^2}(\sum_{j=1}^{N}\alpha_j\sigma(\sum_{i=1}^{n}x_i w_{ji} + b_j))$$

$$\Rightarrow \sum_{j=1}^{N}\alpha_j\frac{\partial^2}{\partial x_v^2}\sigma(\sum_{i=1}^{n}x_i w_{ji} + b_j)$$

$$\Rightarrow \sum_{j=1}^{N}\alpha_j\frac{\partial}{\partial x_v}(\sigma'(\sum_{i=1}^{n}x_i w_{ji} + b_j)(\sum_{i=1}^{n}\frac{\partial x_i}{\partial x_v}w_{ji}))$$

$$\Rightarrow \sum_{j=1}^{N}\alpha_j\frac{\partial}{\partial x_v}(\sigma'(\sum_{i=1}^{n}x_i w_{ji} + b_j)w_{jv})$$

$$\Rightarrow \sum_{j=1}^{N}\alpha_j(\sigma''(\sum_{i=1}^{n}x_i w_{ji} + b_j)(\sum_{i=1}^{n}\frac{\partial x_i}{\partial x_v}w_{ji})w_{jv})$$

Portanto:

$$\frac{\partial^2\mathcal{N}}{\partial x_v^2} = \sum_{j=1}^{N}\alpha_j\sigma''(\sum_{i=1}^{n}x_i w_{ji} + b_j)w_{jv}^2 \tag{3.6}$$

Assim, podemos substituir essa equação (3.6) na formula da atuação do operador H na rede neural, e também escrevemos a rede $\mathscr{N}$ na sua forma explicita (2.15).

$$H\mathscr{N} = -\frac{\hbar^2}{2m}\sum_{v=1}^{n}(\sum_{j=1}^{N}\alpha_j\sigma''(\sum_{i=1}^{n}x_i w_{ji}+b_j)w_{jv}^2)+V(\vec{r})(\sum_{j=1}^{N}\alpha_j\sigma(\sum_{i=1}^{n}x_i w_{ji}+b_j))$$

$$\Rightarrow \sum_{v=1}^{n}(\sum_{j=1}^{N}\alpha_j(-\frac{\hbar^2}{2m})\sigma''(\sum_{i=1}^{n}x_i w_{ji}+b_j)w_{jv}^2)+(\sum_{j=1}^{N}\alpha_j V(\vec{r})\sigma(\sum_{i=1}^{n}x_i w_{ji}+b_j))$$

Definindo $z_j = \sum_{i=1}^{n} x_i w_{ji} + b_j$ fincamos com:

$$\Rightarrow \sum_{v=1}^{n}(\sum_{j=1}^{N}\alpha_j(-\frac{\hbar^2}{2m})\sigma''(z_j)w_{jv}^2)+(\sum_{j=1}^{N}\alpha_j V(\vec{r})\sigma(z_j))$$

$$\Rightarrow \sum_{j=1}^{N}\alpha_j(-\frac{\hbar^2}{2m})\sigma''(z_j)(\sum_{v=1}^{n}w_{jv}^2)+\sum_{j=1}^{N}\alpha_j V(\vec{r})\sigma(z_j)$$

$$\Rightarrow \sum_{j=1}^{N}\alpha_j\{(-\frac{\hbar^2}{2m})\sigma''(z_j)(\sum_{v=1}^{n}w_{jv}^2)+V(\vec{r})\sigma(z_j)\}$$

$$H\mathscr{N} = \sum_{j=1}^{N}\alpha_j\{V(\vec{r})\sigma(z_j)-\frac{\hbar^2}{2m}\sigma''(z_j)(\sum_{v=1}^{n}w_{jv}^2)\} \tag{3.7}$$

Por fim, poderemos finalmente escrever o funcional de forma explicita, definindo os parâmetros das duas redes que compõe a parte real e imaginaria da função de onda, temos $\vec{a}_1 = (\alpha_j^{(1)}, w_{ji}^{(1)}, b_j^{(1)})$ e $\vec{a}_2 = (\alpha_j^{(2)}, w_{ji}^{(2)}, b_j^{(2)})$, também podemos definir $z_j^{(1)} = \sum_{i=1}^{n} x_i w_{ji}^{(1)} + b_j^{(1)}$ e $z_j^{(2)} = \sum_{i=1}^{n} x_i w_{ji}^{(2)} + b_j^{(2)}$. Assim, o funcional $\mathfrak{F}(\vec{a})$ é dado por:

$$\mathfrak{F}(\vec{a}) = \int(\mathscr{N}_1 H\mathscr{N}_1 + \mathscr{N}_2 H\mathscr{N}_2)d\vec{r}$$

Com:

$$H\mathscr{N}_1 = \sum_{j=1}^{N}\alpha_j^{(1)}\{V(\vec{r})\sigma(z_j^{(1)})-\frac{\hbar^2}{2m}\sigma''(z_j^{(1)})(\sum_{v=1}^{n}(w_{jv}^{(1)})^2)\}$$

$$H\mathscr{N}_2 = \sum_{j=1}^{N}\alpha_j^{(2)}\{V(\vec{r})\sigma(z_j^{(2)})-\frac{\hbar^2}{2m}\sigma''(z_j^{(2)})(\sum_{v=1}^{n}(w_{jv}^{(2)})^2)\}$$

A equação (3.5) mostra que o funcional se torna uma função real de $N(2n+4)$ variáveis (este é o tamanho do vetor $\vec{a}$), onde N é o numero de neurônios da camada oculta e n é o numero de valores de entrada que corresponde dimensão do espaço $\mathbb{R}^n$ onde a função de onda esta definida. As condições de normalização e de ortogonalidade, escritas em forma de integrais em (3.2) e (3.3), também podem se tornar funções de $\vec{a}$ representando as funções de onda $\psi(\vec{r})$ por redes neurais. Seja $\psi(\vec{r}) = \mathcal{N}_1 + i\mathcal{N}_2$ suponhamos que conhecemos as k funções $\psi_i(\vec{r}) = \mathcal{N}_{i1} + i\mathcal{N}_{i2}$ (que podem ser ou não representadas por redes neurais), substituindo ambas em (3.2) temos:

$$\int \psi(\vec{r})^* \psi_i(\vec{r}) d\vec{r} = \int (\mathcal{N}_1 - i\mathcal{N}_2)(\mathcal{N}_{i1} + i\mathcal{N}_{i2}) d\vec{r}$$

$$\Rightarrow \int (\mathcal{N}_1\mathcal{N}_{i1} + \mathcal{N}_2\mathcal{N}_{i2}) d\vec{r} - i\int (\mathcal{N}_1\mathcal{N}_{i2} - \mathcal{N}_2\mathcal{N}_{i1}) d\vec{r} = 0$$

Portanto, para cada $i \in \{0, ..., k-1\}$ temos que, assim como em (3.5), as integrais da parte real e imaginaria dependem do vetor $\vec{a}$, que é o parâmetro da rede, assim definimos as funções $R_i(\vec{a})$ e $I_i(\vec{a})$:

$$R_i(\vec{a}) = \int (\mathcal{N}_1\mathcal{N}_{i1} + \mathcal{N}_2\mathcal{N}_{i2}) d\vec{r} = 0$$

$$I_i(\vec{a}) \int (\mathcal{N}_1\mathcal{N}_{i2} - \mathcal{N}_2\mathcal{N}_{i1}) d\vec{r} = 0$$

Assim, temos:

$$R_i(\vec{a}) = 0 \quad \text{e} \quad I_i(\vec{a}) = 0 \quad \text{para cada} \quad i \in \{0, ..., k-1\} \tag{3.8}$$

A condição de normalização em (3.1) também se torna função de $\vec{a}$ se a rede neural for usada para representar sua função de onda, e portanto, definimos a função $h(\vec{a})$ da seguinte maneira:

$$\|\psi(\vec{r})\|^2 = \int (\mathcal{N}_1^2 + \mathcal{N}_2^2) d\vec{r} = h(\vec{a}) = 1$$

Por fim, encontramos a ultima condição de vinculo do funcional:

$$h(\vec{a}) - 1 = 0 \tag{3.9}$$

A discussão precedente serviu para mostrar que a introdução das redes neurais da forma $\mathscr{N}(\vec{r},\vec{a})$ no problema de minimização do funcional, transforma inteiramente um problema que antes estava definido num espaço de *Hilbert* abstrato para um problema de minimização de uma função de varias variáveis sujeita a vínculos. O funcional se torna uma função $\mathfrak{F}:\mathbb{R}^{N(2n+4)}\longrightarrow\mathbb{R}$ e seus vínculos (3.8) e (3.9) também são transformados em funções de mesma natureza.

Agora partimos para solução do problema, desejamos encontrar as auto-funções do sistema, representadas por uma rede neural, assim iniciamos com o **estado fundamental**, como foi discutido na seção (2.1) basta que o funcional seja minimizado sujeito a condição de normalização da auto-função, assim, para encontrar $\psi_0(\vec{r})$ basta:

$$\begin{cases} \text{Minimizar} & \mathfrak{F}(\vec{a}) = \int(\mathscr{N}_1 H \mathscr{N}_1 + \mathscr{N}_2 H \mathscr{N}_2)d\vec{r} \\ \text{Sujeito a} & h(\vec{a}) - 1 = 0 \end{cases}$$

O resultado obtido é um certo vetor $\vec{a}_0$ que corresponde aos parâmetros da auto-função do estado fundamental. Achar a auto-função do estado fundamental não costuma ser problemático, pois ela não costuma possuir degenerescências, porém a condição de normalização ainda é necessária para que a rede neural satisfaça as condições do teorema de Ritz.

Como também foi mostrado na seção (2.1), para encontrar a k-ésima auto-função devemos conhecer as outras k auto-funções anteriores, por exemplo, se sabemos o estado fundamental $\psi_0(\vec{r})$, para encontrar o estado de nível 1 basta usar a condição de que $\psi_1(\vec{r})$ seja ortogonal a $\psi_0(\vec{r})$, o que é expresso nos vínculos $R_0(\vec{a}) = 0$ e $I_0(\vec{a}) = 0$ definidos em (3.8), e assim minimizar o funcional sujeito a esses vínculos e ao vinculo de normalização. Podemos repetir o processo para o nível 2, nível 3 e assim por diante, acumulando os vínculos (3.8). Assim, a para encontrar a a k-ésima auto-função, devemos:

$$\begin{cases} \text{Minimizar} & \mathfrak{F}(\vec{a}) = \int(\mathscr{N}_1 H \mathscr{N}_1 + \mathscr{N}_2 H \mathscr{N}_2)d\vec{r} \\ \text{Sujeito a} & h(\vec{a}) - 1 = 0 \quad \text{e} \quad \{R_i(\vec{a}) = 0, I_i(\vec{a}) = 0\} \end{cases}$$

$$\text{para cada} \quad i \in \{1,...,k-1\}$$

A seguir, sera descrito de que forma esse método pode ser implementado computacionalmente.

3.1 Implementação computacional

3.1.1 Definindo as principais funções

As equações e o método foram implementados utilizando a linguagem de programação *Python* com o auxilio de varias bibliotecas desta linguagem. Primeiramente iniciamos o programa importando essas bibliotecas, em especial, a biblioteca *Scipy* tem papel fundamental neste programa, uma vez que é dela que vira o método de otimização vinculada que será usado para minimizar o funcional.

Figura 5 – Bibliotecas utilizadas no programa.

```
import numpy as np
import random as rd
import matplotlib.pyplot as plt
import math as m
import scipy.special as sp
import scipy.constants as scp
import scipy.integrate as inte
from scipy.optimize import minimize
```

Fonte: Elaborado pelo autor.

Em seguida, definimos as principais funções que serão usadas no decorrer do programa, todas pensadas para operar com *Arrays* (com exceção apenas a função *Potencial(x)*).

Figura 6 – Definição de funções que serão utilizadas no decorrer do programa.

```
o=4
def sig(x):
  return 1/(1+np.exp(-x))
def d2sig(x):
  return (np.exp(-2*x)-np.exp(-x))/((1+np.exp(-x))**3)
def ale(u):
  a3=[]
  for i in range(u):
    a3.append(rd.uniform(-o,o))
  return a3
def Potencial(x):
  return 0
```

Fonte: Elaborado pelo autor.

A função *sig(x)* é a função sigmoide, que como já foi discutido, tem papel central no teorema da aproximação universal, e por sua vez a sua derivada segunda é representa pela função *d2sig(x)* que tem papel no calculo do operador hamiltoniano sobre a rede neural.

A função *ale(u)* recebe um numero inteiro u e tem o proposito de gerar um *array* de u números aleatórios, esses números são gerados uniformemente de um intervalo simétrico $(-o, o)$ onde a variável o esta definida na primeira linha. Por fim, a função *Potencial(x)* é justamente o potencial da hamiltoniana do sistema quântico que se deseja solucionar. Passamos agora para definição do funcional e o vinculo de normalização.

Figura 7 – Primeiro trecho de código da implementação do funcional.

```
n0=8
#funcional
def g(a,x,n):
  a0=np.array(a)
  z1=a0[2*n:3*n]*x+a0[4*n:5*n]
  z2=a0[3*n:4*n]*x+a0[5*n:6*n]
  R=np.dot(a0[:n].T,sig(z1))
  I=np.dot(a0[n:2*n].T,sig(z2))
  return (R**2)+(I**2)
def h(a,x,n):
  a0=np.array(a)
  z1=a0[2*n:3*n]*x+a0[4*n:5*n]
  z2=a0[3*n:4*n]*x+a0[5*n:6*n]
  L=Potencial(x)
  c=((scp.hbar**2)/2)
  R1=np.dot(a0[:n].T,L*sig(z1)-c*(a0[2*n:3*n]**2)*d2sig(z1))
  I1=np.dot(a0[n:2*n].T,L*sig(z2)-c*(a0[3*n:4*n]**2)*d2sig(z2))
  R2=np.dot(a0[:n].T,sig(z1))
  I2=np.dot(a0[n:2*n].T,sig(z2))
  return R1*R2+I1*I2
```

Fonte: Elaborado pelo autor.

A variável *n0* corresponde ao número de neurônios da camada oculta. Em todas as funções neste bloco de código a variável a é um *array* que contém os pesos e vieses que definem a rede neural, x é a variável de posição da função de onda, e n é o numero de neurônios da camada oculta da rede. A função *g(a,x,n)* é o integrando da integral de normalização em (3.3), as variáveis *z1* e *z2* são o argumento da função sigmoide, R e I são respectivamente as redes neurais da parte real e imaginaria da função de onda. Por fim, a função *h(a,x,n)* é o integrando da integral do funcional $\mathfrak{F}(\vec{a})$, com R_1 e I_1 sendo as formulas (3.7) para atuação da hamiltoniana nas redes neurais, e R_2 e I_2 sendo as redes neurais como na função *g(a,x,n)*. O *array a* contem todos os pesos e vieses das duas redes neurais que formam a função de onda.

Figura 8 – Segundo trecho de código da implementação do funcional.

```
def w(a1,a2,x,n):
  ax=np.array(a1)
  ay=np.array(a2)
  zx1=ax[2*n:3*n]*x+ax[4*n:5*n]
  zx2=ax[3*n:4*n]*x+ax[5*n:6*n]
  Rx=np.dot(ax[:n].T,sig(zx1))
  Ix=np.dot(ax[n:2*n].T,sig(zx2))
  zy1=ay[2*n:3*n]*x+ay[4*n:5*n]
  zy2=ay[3*n:4*n]*x+ay[5*n:6*n]
  Ry=np.dot(ay[:n].T,sig(zy1))
  Iy=np.dot(ay[n:2*n].T,sig(zy2))
  return (Ry*Rx-Ix*Iy,Ry*Ix+Rx*Iy)
def F(a,n):
  return inte.quad(lambda x: h(a,x,n),0,1)[0]
def res3(a):
  return inte.quad(lambda x: g(a,x,n0),0,1)[0]-1
```

Fonte: Elaborado pelo autor.

A função *w(a1,a2,x,n)* foi escrita de forma semelhante a função *h(a,x,n)*, e é o integrando do produto interno entre duas funções de onda representadas por uma rede neural, assim com esta função serão definidos os vínculos (3.8) de ortogonalidade, os vetores *a1* e *a2* são os *arrays* que definem as duas funções de onda, as variáveis *x* e *n* são definidas como nas funções anteriores. Ao fim, a função retorna uma *tupla* com a parte real e imaginaria do integrando.

No final deste bloco de código temos finalmente a definição da função *F(a,n)* que é o funcional $\mathfrak{F}(\vec{a})$, a condição de normalização é definida na função *res3(a)*, ambas são definidas usando a biblioteca *scipy*, o intervalo de integração é ajustado conforme a necessidade do problema.

3.1.2 Sequential Least Squares Programming (SLSQP)

Apos definir as principais funções utilizadas, passamos a minimização do funcional. Neste trabalho, foi utilizado o método *Sequential Least Squares Programming*, que pertence a família de algorítimos *Sequential quadratic programming* que tem o objetivo de solucionar problemas de otimização sujeitos a vínculos não lineares (NOCEDAL; WRIGHT, 2006). A ideia geral deste tipo de método é solucionar o problema de otimização através de uma sequencia de aproximações quadráticas da função que se deseja otimizar e uma aproximação linear para as equações de vinculo, cada um dos subproblemas é bem mais simples de ser resolvido, a cada iteração a solução das aproximações quadráticas e lineares convergem para solução real do problema (NOCEDAL; WRIGHT, 2006).

A biblioteca do *Scipy* dispõe de um modulo que soluciona problemas de otimização, o *scipy.optimize.minimize*, que foi importado com abreviaturas no inicio do programa, este modulo possibilita a utilização de diversos métodos números de minimização e entre eles selecionamos o *Sequential Least Squares Programming (SLSQP)*, assim, escrevemos:

Figura 9 – Implementação do método SLSQP para o estado fundamental.

```
aux=[]
for i in range(6*n0):
  aux.append((-2*o,2*o))
b=tuple(aux)
```

```
r3={'type':'eq','fun': res3}
cons=([r3])
```

```
res=minimize(lambda x: F(x,n0), ale(6*n0),
             method='SLSQP', bounds=b,
             constraints=cons,options={'ftol': 1e-100})
```

Fonte: Elaborado pelo autor.

O método SLSQP do *scipy.optimize.minimize* exige que os *arrays* candidatos a solução sejam procurados em um espaço de busca restrito, assim no primeiro bloco de código definimos uma *tupla* salva da variável *b*, que representa um retângulo multidimensional de *6n0* dimensões (*n0* sendo o números de neurônios na camada oculta), onde as soluções serão buscadas. O retângulo tem lados iguais, definidos a partir da variável *o* que foi definida no bloco de código da **Figura 6**.

Como foi dito anteriormente, a primeira auto-função que iremos buscar é a do estado fundamental, assim necessitamos apenas da condição de normalização como condição de vinculo. As condições de vinculo devem ser definidas em *dicionários* do *Python*, primeiramente usamos a função *res3(a)* para definir a primeira condição de normalização, por padrão o *Scipy* considera que a função de vinculo seja igual a zero, isso justifica a escrita da condição de normalização na forma $h(\vec{a}) - 1 = 0$ como na equação (3.9). Na **Figura 9** isso é feito no segundo bloco de código, onde o dicionario *r3* é definido, estabelecendo as chaves *'type'* como *'eq'*, indicando que se trata de uma equação, e *'fun'* como a função *res3* definida anteriormente, assim esse dicionario é adicionado a um lista, todos os dicionários desta lista são os vínculos do problema, neste bloco de código o vinculo de normalização é guardado na variável *cons*.

Por fim, no ultimo bloco da **Figura 9** escrevemos função *minimize* que tem como

primeiro parâmetro a função que se deseja otimizar. O segundo parâmetro é o *array* aleatório que é o ponto de partida para o algorítimo, este é definido usando a função *ale(u)*. O terceiro *'method'* é o metodo utilizado para minimização, que é o SLSQP. O quarto, *'bounds'* é a região onda a solução sera buscada. Por ultimo, *'constraints'* é a lista de dicionários que representam os vínculos do problema. Ao final da execução, entre as variáveis que o programa retorna esta o *array* $\vec{a}_0$ que produz a rede neural que aproxima o estado fundamental do sistema. O valor que a função *F(a,n)* assume para esse *array* é justamente a **energia do estado fundamental do sistema**. Agora vamos determinar a solução para o primeiro estado excitado, escrevemos mais uma bloco de código:

Figura 10 – Implementação do método SLSQP para o primeiro estado excitado.

```
def res4a(a):
  return inte.quad(lambda x: w(a,res.x,x,n0)[0],0,1)[0]
def res4b(a):
  return inte.quad(lambda x: w(a,res.x,x,n0)[1],0,1)[0]
```

```
r4a={'type':'eq','fun': res4a}
r4b={'type':'eq','fun': res4b}
cons1=([r3,r4a,r4b])
```

```
res1=minimize(lambda x: F(x,n0), ale(6*n0),
              method='SLSQP', bounds=b,
              constraints=cons1,options={'ftol': 1e-100})
```

Fonte: Elaborado pelo autor.

Primeiramente, usando a função *w(a1,a2,x,n)* como integrando, definimos por meio de integrais as funções *res4a(a)* e *res4b(a)* que são as parte real e imaginaria da condição de vinculo $\langle\phi|\psi_0\rangle = 0$ e com o *array* do estado fundamental encontrado no passo anterior, usamos ele na entrada *a2*, assim temos a primeira condição de ortogonalidade. Assim como foi feito para condição de normalização, definimos dois *dicionários r4a* e *r4b* para cada condição de ortogonalidade, e junto do dicionario *r3* adicionamos todos a lista de dicionários na variável *cons1*. Por fim, no ultimo bloco de código realizamos a minimização da mesma forma que foi feita no anteriormente, como a diferença de que agora temos os vínculos de ortogonalidade, com eles ao final da execução é produzido um *array* que forma a rede neural que representa o **primeiro estado excitado.**

Agora chegamos a ultima parte do programa, repetimos aqui o mesmo procedimento adotado no paragrafo anterior. Com a função *w(a1,a2,x,n)* definimos mais duas funções *res5a* e *res5b*, elas são condições do normalização, porem com o *array* encontrado no passo anterior, assim temos a segunda condição de ortogonalidade $\langle\phi|\psi_1\rangle = 0$, por fim definimos também os *dicionários r5a* e *r5b* e adicionamos eles junto dos outros que forma definido anteriormente na variável *cons2*.

Figura 11 – Implementação do método SLSQP para o segundo estado excitado.

```
def res5a(a):
  return inte.quad(lambda x: w(a,res1.x,x,n0)[0],0,1)[0]
def res5b(a):
  return inte.quad(lambda x: w(a,res1.x,x,n0)[1],0,1)[0]
```

```
r5a={'type':'eq','fun': res5a}
r5b={'type':'eq','fun': res5b}
cons2=([r3,r4a,r4b,r5a,r5b])
```

```
res2=minimize(lambda x: F(x,n0), ale(6*n0),
              method='SLSQP', bounds=b,
              constraints=cons2,options={'ftol': 1e-76})
```

Fonte: Elaborado pelo autor.

No ultimo bloco de código, temos a minimização da função sujeita aos vínculos do estado fundamente e do primeiro estado excitado, assim ao final da execução é produzida a rede que representa a auto-função do segundo estado exitado.

O programa encontrou 3 soluções aproximadas para as 3 primeiras auto-funções do sistema quântico considerado. O processo poderia continuar indefinidamente, encontrando tantas auto-funções quanto se desejasse, porém o custo computacional aumentaria, já que as condições de vinculo se acumulariam a cada passo. Também devemos lembrar que as soluções obtidas são aproximações, dependendo da qualidade da solução do estado fundamental a qualidade das soluções das demais auto-funções seria afetada pois as condições de ortogonalidade dependem das soluções anteriores, que são aproximações, portanto é de se esperar que a qualidade das soluções caia a cada passo para se obter uma nova solução. O funcional $\mathfrak{F}(\vec{a})$ aplicado no vetor $\vec{a}$ fornece uma aproximação para a **auto-energia** de uma auto-função quando este vetor $\vec{a}$ é o que produz uma rede neural que aproxima uma auto-função, assim também podemos obter um conjunto de auto-energias para cada solução encontrada.

4 RESULTADOS E DISCUSSÕES

Para que a eficiência do método fosse testada, aplicamos ele em dois sistemas quânticos unidimensionais que possuem solução analítica, o **potencial infinito com poço quadrado** e o **oscilador harmônico**. A escolha de potenciais unidimensionais deve-se principalmente ao fato de que sistemas dessa natureza não possuem estados degenerados, como foi dito na seção (2.1.2) estados degenerados impõe dificuldades ao método apesar de não inviabiliza-lo, esta escolha foi feita por simplicidade (COHEN-TANNOUDJI *et al.*, 2020).

A analise dos resultado nos permite observar uma característica importante nestes dois sistemas, sabemos previamente que as auto-funções do potencial de poço quadrado são limitadas ao um intervalo da rete real e que as auto-funções do oscilador harmônico estão definidas em toda a reta, lembrando que o **teorema da aproximação universal** tem como um dos seus pressupostos que a função a ser aproximada esteja definida num intervalo compacto, o caso do poço quadrado não apresenta problemas, porém o oscilador harmônico sim (COHEN-TANNOUDJI *et al.*, 2020). Para contornar essa dificuldade escolhemos um intervalo na reta que que seja suficientemente grande, como as auto-funções tem que ter seu modulo quadrado integral por integrais indefinidas, devem ir a zero no limite $x \to \infty$ e $x \to -\infty$, assim investigaremos os efeitos dessa aproximação sobre a qualidade das soluções.

Neste trabalho nos limitaremos a analisar dois aspectos básicos referentes a qualidade das soluções. O primeiro é a comparação entre os gráficos das soluções analíticas e as soluções aproximadas, este parâmetro é puramente qualitativo, não tendo grande relevância na obtenção das propriedades do sistema, porém ele nos dá uma noção geral de quão próximo das soluções reais esta a solução aproximada, servindo com um parâmetro para avaliar a eficacia do método. O segundo é referente a comparação entre as auto-energias produzidas pelas aproximações e as produzidas pela solução analítica, comparando diretamente seus valores e o erro percentual envolvido.

4.1 Potencial infinito com poço quadrado

Analisaremos as soluções do potencial com poço quadrado, o seu hamiltoniano tem a forma:

$$H = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)$$

Com o potencial definido por:

$$V(x) = \begin{cases} 0 & \text{para} \quad x \in (0,L) \\ \infty & \text{caso contrario} \end{cases}$$

Pela forma do potencial, sabemos previamente que a partícula não pode ocupar as regiões fora do intervalo aberto $(0,L)$, portanto devemos impor às autofunções $\psi(x)$, que solucionam o problema, as seguintes condições de contorno:

$$\psi(0) = 0 \quad \text{e} \quad \psi(1) = 0 \tag{4.1}$$

Condições do tipo (4.1) não são mencionadas explicitamente no método descrito no capitulo 3, porém esta dificuldade pode ser contornada simplesmente adicionando as condições (4.1) ao conjunto de vínculos ao qual o funcional esta sujeito. No programa, usamos a função *g(a,x,n)* que é justamente o modulo quadrado da função de onda, assim podemos escrever:

Figura 12 – Adição das condições de contorno as condições vinculo do algorítimo.

```
def F(a,n):
  return inte.quad(lambda x: h(a,x,n),0,1)[0]
def res1(a):
  return g(a,0,n0)
def res2(a):
  return g(a,1,n0)
def res3(a):
  return inte.quad(lambda x: g(a,x,n0),0,1)[0]-1
```

```
aux=[]
for i in range(6*n0):
  aux.append((-2*o,2*o))
b=tuple(aux)
```

```
r1={'type':'eq','fun': res1}
r2={'type':'eq','fun': res2}
r3={'type':'eq','fun': res3}
cons=([r1,r2,r3])
```

Fonte: Elaborado pelo autor.

Para representar as condições de contorno (4.1) definimos as funções *res1(a)* e *res2(a)* e adicionamos elas as condições de vinculo assim como fizemos para a condição de normalização.

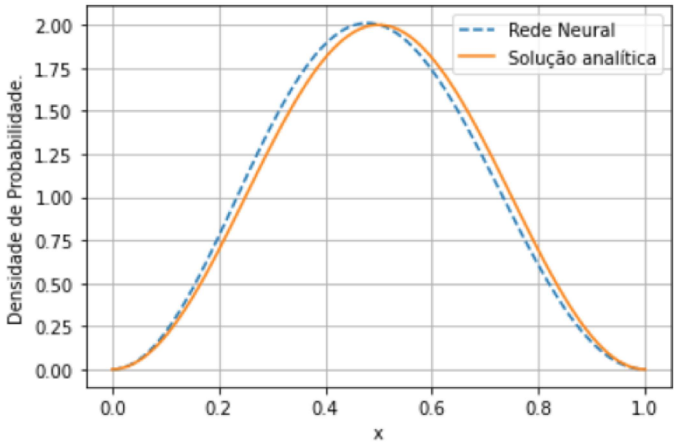
Consideremos os valores $L = m = 1$ para simplificar a solução. Assim, diante de todas essas considerações, sabemos que a solução analítica (já normalizada) é dada por:

$$\psi_n(x) = \sqrt{2}\sin(n\pi x)$$

$$E_n = \frac{n^2\pi^2\hbar^2}{2}$$

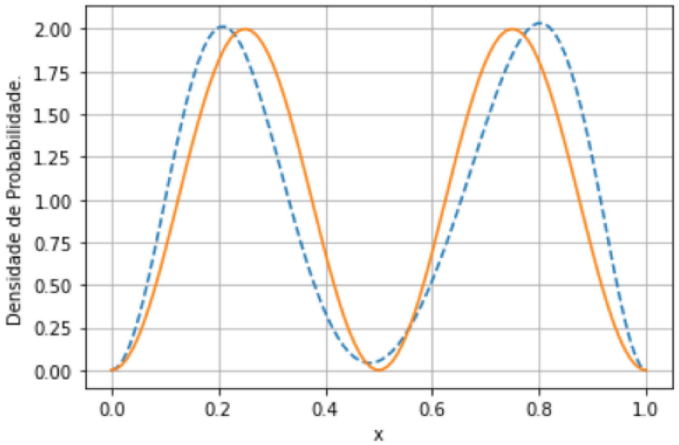
Iniciamos o programa definindo a variável *o=4* (esta define o intervalo para o espaço de busca) e *n0=8* (o numero de na camada oculta). Os gráficos referentes às densidades de probabilidade das auto-funções são listados abaixo.

Figura 13 – Estado fundamental para o potencial de poço quadrado.



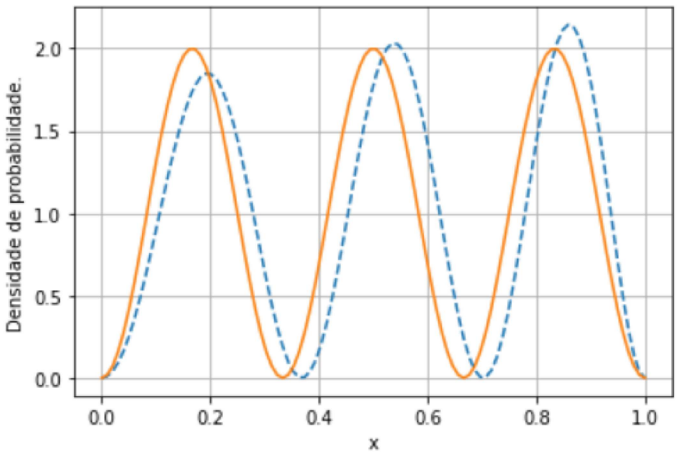
Fonte: Elaborado pelo autor.

Figura 14 – Primeiro estado excitado para o potencial de poço quadrado.



Fonte: Elaborado pelo autor.

Figura 15 – Segundo estado excitado para o potencial de poço quadrado.



Fonte: Elaborado pelo autor.

A natureza probabilística do método de minimização SLSQP causou algumas dificuldades, o programa teve que ser executado varias vezes até que um conjunto aceitável de soluções fosse encontrado, os gráficos exibidos nas figuras acima são a melhor solução encontrada. Isto indica uma dificuldade, já que em uma situação onde não há solução analítica disponível, não existe forma fácil de determinar se o produziu uma boa aproximação. Como veremos mais a frente no caso do oscilador harmônico, essa dificuldade pode ser contornado fazendo modificações na equação de Schrodinger. A seguir, temos a tabela com todos os valores de energias obtidos pelo método comparados com os valores exatos.

Tabela 1 – Resultados obtidos para as auto-energias da solução analítica e aproximada do potencial poço quadrado.

Energias	Solução analítica	Solução aproximada	Erro percentual
E_1	5.487×10^{-68} J	5.526×10^{-68} J	0.689%
E_2	2.194×10^{-67} J	2.451×10^{-67} J	11.654%
E_3	4.938×10^{-67} J	5.073×10^{-67} J	2.728%

Apesar de um erro relativamente alto no estado de energia E_2 podemos afirmar que em geral os valores obtidos se aproximam do valor real obtido analiticamente .Um outro ponto a se observar é que inevitavelmente a qualidade das soluções decai a medida que novas auto-funções são encontradas baseadas nas condições de ortogonalidade que envolvem as auto-funções anteriores, isso fica evidente nos gráficos, onde as soluções do primeiro e segundo estado excitado se mostram piores que a solução do estado fundamental.

4.2 Oscilador Harmônico

Passamos agora a solução do oscilador harmônico, este sistema possui a hamiltoniana dada por:

$$H = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2 x^2 \tag{4.2}$$

Em contraste com o potencial de poço quadrado, o oscilador harmônico possui suas soluções definidas e toda a reta real. Sendo ω a frequência angular natural do sistema, as auto-funções e auto-energias que são soluções analíticas do problema são:

$$\psi_n(x) = \frac{1}{\sqrt{2^n n!}}(\frac{m\omega}{\pi\hbar})^{\frac{1}{4}} H_n(X)\exp(-\frac{X^2}{2})$$

$$E_n = (\frac{1}{2} + n)\hbar\omega$$

$$\text{Onde:} \quad X = \sqrt{\frac{m\omega}{\hbar}}x$$

Agora vamos obter as soluções através do método variacional. Ao se tentar encontrar as soluções diretamente da mesma forma como foi feito para o poço quadrado percebeu-se que o programa sempre produzia valores muito distantes dos valores previstos para as auto-energias, e também gráficos das densidades de probabilidade não respeitavam nem mesmo as condições de normalização. Após varias tentativas de solucionar esta dificuldade chegou-se a conclusão de que o problema estava na constante $\hbar$ presente da equação de Schrodinger, ao elimina-la o programa teve seu desempenho bastante melhorado. Assim, isso sugere que um caminho para a solução pode ser reformular o hamiltoniano (4.2) de forma que a constante $\hbar$ seja eliminada. Seja a equação de Schrodinger dada por:

$$(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + \frac{1}{2}m\omega^2 x^2)\psi_n(x) = E_n\psi_n(x)$$

Seja $X = \sqrt{\frac{m\omega}{\hbar}}x$, façamos a seguinte transformação de coordenadas: substituindo x por X, assim calculamos as derivadas do hamiltoniano no novo sistema de coordenadas:

$$\frac{d}{dX} = \frac{dx}{dX}\frac{d}{dx} = \sqrt{\frac{\hbar}{m\omega}}\frac{d}{dx} \quad \Longrightarrow \quad \frac{d^2}{dX^2} = \frac{\hbar}{m\omega}\frac{d^2}{dx^2}$$

Portanto:

$$(-\frac{\hbar^2}{2m}\frac{m\omega}{\hbar}\frac{d^2}{dX^2}+\frac{1}{2}m\omega^2\frac{\hbar}{m\omega}X^2)\Psi_n(X)=E_n\Psi_n(X)$$

$$(-\frac{\hbar\omega}{2m}\frac{d^2}{dX^2}+\frac{1}{2}\hbar\omega X^2)\Psi_n(X)=E_n\Psi_n(X)$$

$$(-\frac{1}{2m}\frac{d^2}{dX^2}+\frac{1}{2}X^2)\Psi_n(X)=\frac{E_n}{\hbar\omega}\Psi_n(X)$$

Assim definimos $\varepsilon_n=\frac{E_n}{\hbar\omega}$ e chamamos esta de **energia adimensional**, e por fim, temos a equação sem a presença da contante de Planck reduzida $\hbar$, que é:

$$-\frac{1}{2m}\frac{d^2\Psi_n(X)}{dX^2}+\frac{1}{2}X^2\Psi_n(X))=\varepsilon_n\Psi_n(X) \tag{4.3}$$

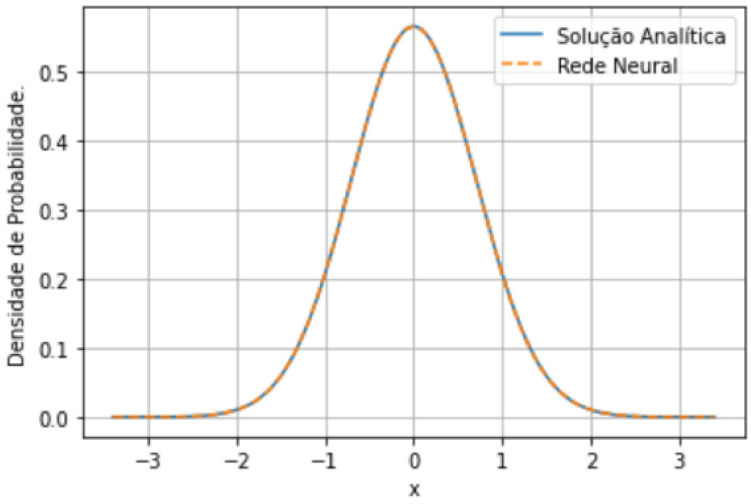
Esta equação é semelhante a equação de Schrodinger original, e portanto, quando encontramos suas soluções basta retornar a variável original com a transformação $X=\sqrt{\frac{m\omega}{\hbar}}x$ e assim as soluções encontradas passam a coincidir com as soluções de (4.2), que é a hamiltoniana da equação original. As soluções analíticas de (4.3) são:

$$\Psi_n(x)=\frac{1}{\pi^{\frac{1}{4}}\sqrt{2^n n!}}H_n(X)\exp(-\frac{X^2}{2})$$

$$\varepsilon_n=(\frac{1}{2}+n)$$

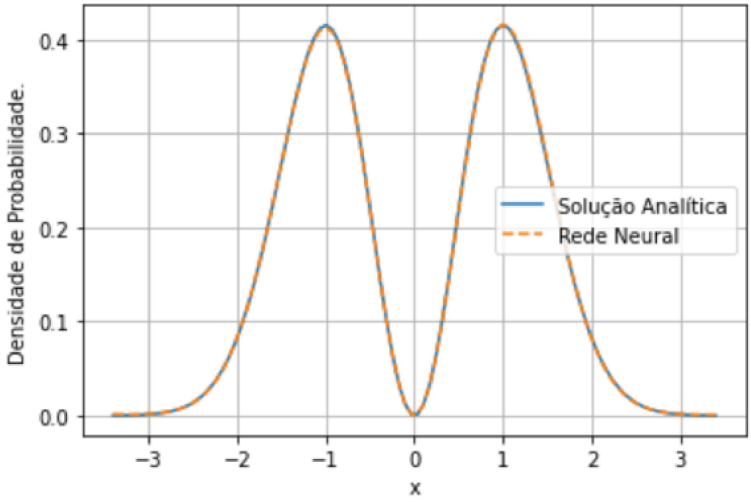
Finalmente vamos solucionar a equação (4.3) com as redes neurais. Para simplificar as soluções, consideremos que seja $\omega=m=1$ da mesma forma que na solução do poço quadrado. Como o potencial do oscilador harmônico esta definido em toda a reta real, para satisfazer o teorema da aproximação universal devemos escolher um intervalo na reta na qual a maior parte da auto-função esta contida. Escolhendo um intervalo $[-3.4,3.4]$ (consideramos que trabalhamos nas unidades da equação (4.3)) e definindo uma rede com 8 neurônios na camada oculta, encontramos as 3 primeiras auto-funções.

Figura 16 – Estado fundamental para o Oscilador harmônico.



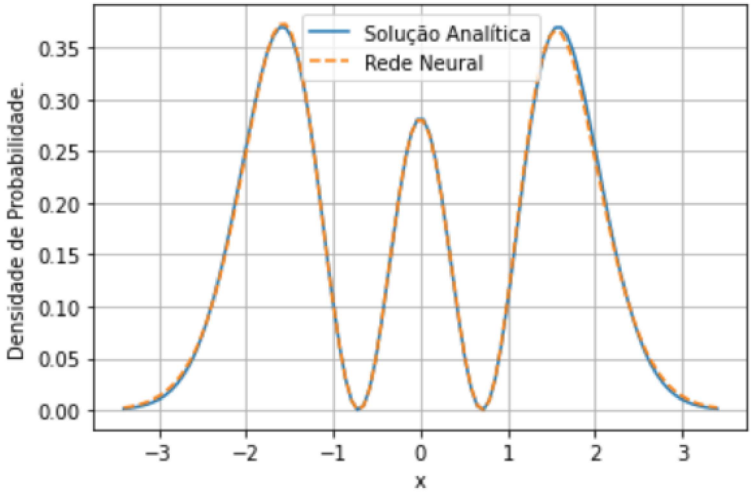
Fonte: Elaborado pelo autor.

Figura 17 – Primeiro estado exitado para o Oscilador harmônico.



Fonte: Elaborado pelo autor.

Figura 18 – Segundo estado exitado para o Oscilador harmônico.



Fonte: Elaborado pelo autor.

Os gráficos das **Figuras 16, 17 e 18** são os da solução da equação de Schrodinger (4.3), como já foi mencionado, para obter as solução da equação de Schrodinger original, basta substituir a transformação $X = \sqrt{\frac{m\omega}{\hbar}}x$ no argumento da rede neural, assim sendo $\Psi_n(X) = \mathscr{N}_{n1}(X) + i\mathscr{N}_{n2}(X)$ uma solução de (4.3), então a solução de (4.2) (considerando $\omega = m = 1$) é simplesmente:

$$\psi_n(x) = \Psi_n(\sqrt{\frac{1}{\hbar}}x) = \mathscr{N}_{n1}(\sqrt{\frac{1}{\hbar}}x) + i\mathscr{N}_{n2}(\sqrt{\frac{1}{\hbar}}x) \tag{4.4}$$

Um ponto importante para se discutir é sobre a normalização desta auto-função, quando fizemos a mudança de escala e voltamos a variável x, a função $\psi_n(X)$ estava normalizada em relação a variável X, portanto devemos normalizar a solução quando mudamos para variável x novamente. Agora para obter as auto-energias lembramos da formula $\varepsilon_n = \frac{E_n}{\hbar\omega} = \frac{E_n}{\hbar}$ assim temos $E_n = \varepsilon_n\hbar$, as soluções encontradas nos permitem obter das energia adimensionais ε_n, assim temos a seguinte tabela com os valores de auto-energias:

Tabela 2 – Resultados obtidos para as auto-energias da solução analítica e aproximada do oscilador harmônico.

Energias	Solução analítica	Solução aproximada	Erro percentual
E_1	5.273×10^{-35} J	5.272×10^{-35} J	0.019%
E_2	1.582×10^{-34} J	1.581×10^{-34} J	0.063%
E_3	2.635×10^{-34} J	2.638×10^{-34} J	0.113%

A mudança de abordagem em remover a constante $\hbar$ da equação de Schrodinger provocou um grande aumento no desempenho do programa, as auto-funções foram encontradas com muito mais facilmente e rapidamente coincidiam com os gráficos das soluções, tudo sem a necessidade de executar o programa varias vezes como foi feito da resolução do potencial de poço quadrado. Também se nota uma grande proximidade entre as auto-energias esperadas e as que foram obtidas numericamente.

A suposição de que as soluções estavam contidas num intervalo compacto da reta possibilitou a utilização do teorema da aproximação universal, pela qualidade das soluções encontradas podemos afirmar que isso não afetou a eficiência do programa em encontrar boas soluções.

5 CONCLUSÕES

Os resultados obtidos permitem concluir que o método variacional aliado às redes neurais artificiais têm alguma eficiência em encontrar as auto-funções e auto-energias de sistemas quânticos unidimensionais. Estes sistemas foram escolhidos por sua simplicidade e ausência de estados degenerados, porém como já foi mencionado, nada impede que sistemas de dimensões maiores possam ser resolvidos desta maneira. Por exemplo, um caminho pode ser encontrar apenas o estado fundamental, que geralmente é não degenerado, e em seguida usar os processo de ortogonalização de Gram-Schmidt para encontrar todas as auto-funções do sistema, pois todas elas devem ser ortogonais entre si independente da sua degenerescência. Em sistemas de maior dimensão é de esperar que as funções de onda sejam mais complexas e por isso as redes neurais também sejam, isso pode aumentar a complexidade do programa e assim tornar mais difícil o seu emprego prático. Os resultados obtidos neste trabalho não permitem tirar conclusões sobre a eficiência do método em sistemas de maior dimensão.

Outro aspecto importante a se destacar é que em sistemas quânticos onde o potencial está definido em toda a reta real, apesar do teorema da aproximação universal só garantir que a rede aproxima funções em conjuntos compactos, as redes neurais podem aproximar as funções de onda em regiões compactas do seu domínio, assim como foi feito no oscilador harmônico. Se soubermos que uma certa região compacta do domínio da função de onda concentra a maior parte dos valores da densidade de probabilidade então podemos aproximar na função nessa região pela rede neural sem perder grande informação, no oscilador harmônico sabíamos que a força era de natureza restauradora e por isso a partícula deveria estar restrita a uma certa região, o que faria a probabilidade de encontrá-la fora desta região ir a zero. Também uma possível aplicação seria em potenciais periódicos com os de redes cristalinas, assim poderíamos resolver o sistema para uma região que corresponde a uma célula unitária da rede, e usar os vetores que definem a rede de Bravais para estender a solução para toda a rede cristalina.

Como ficou evidente na solução do Oscilador harmônico, eliminar a constante de Planck reduzida $\hbar$ torna o programa muito mais eficiente em encontrar as auto-funções, por isso usar unidades de medida em que essa constante seja 1 pode facilitar muito a solução do sistema.

REFERÊNCIAS

BAUMEISTER, J.; LEITÃO, A. **Introdução a Teoria do Controle e Programação Dinâmica**. Rio de Janeiro: IMPA, 2014.

CALLEN, H. B. **Thermodynamics and an Introduction to Thermostatistics**. New York: John Wiley Sons, 1985.

CHEN, F.; TILLBERG, P. W.; BOYDEN, E. S. Expansion microscopy. **Science**, v. 347, p. 543–548, 2015.

COHEN-TANNOUDJI, C.; DIU, B.; LALOE, F. **Quantum Mechanics**: basic concepts, tools, and applications. Weinheim, Germany: Wiley-VCH, 2020. v. 1.

HAYKIN, S. O. **Neural Networks**: a comprehensive foundation. Upper Saddle River, New Jersey: Pearson, 1998.

LUDWIG, O.; MONTGOMERY, E. **Redes Neurais**: fundamentos e aplicações com programas em c. Rio de Janeiro: Ciência Moderna, 2007.

NOBACK, C. R.; STROMINGER, N. L.; DEMAREST, R. J.; RUGGIERO, D. A. **The Human Nervous System**: structure and function. New Jersey: Humana Press, 2005. v. 1.

NOCEDAL, J.; WRIGHT, S. J. **Numerical Optimization**. New York: Springer, 2006. v. 1.

POINCARÉ, H. **O valor da ciência**. Rio de Janeiro: Contraponto, 1995. v. 1.

APÊNDICE A – DEMONSTRAÇÃO DO TEOREMA DE RITZ

Antes de iniciarmos a demonstração enunciaremos uma serie de resultados do **cálculo das variações** (BAUMEISTER; LEITÃO, 2014). Inciamos com a **Variação de Gâteux** que é uma generalização para espaços vetoriais gerais da derivada direcional do calculo de varias variáveis.

Definição A.0.1 *Seja Y um espaço vetorial e $\mathfrak{F} :\to \mathbb{R}$. Dados $y, v \in Y$ definimos a* ***Variação de Gâteaux*** *de $\mathfrak{F}$ em y na direção de v através de*

$$\delta\mathfrak{F}[y : v] = \lim_{\varepsilon\to 0} \frac{\mathfrak{F}[y+\varepsilon v] - \mathfrak{F}[y]}{\varepsilon}$$

quando este limite existe.

Teorema 3 (de Lagrange) *Seja Y um espaço vetorial normado e $\mathfrak{F}$, $\mathfrak{G}$ aplicações de Y em $\mathbb{R}$. Suponha que as variações de Gâteaux $\delta\mathfrak{F}[y : v]$, $\delta\mathfrak{G}[y : v]$ estão bem definidos para $y, v \in Y$ e ainda que cada $y, v \in Y$ tenhamos $\delta\mathfrak{F}[y_n : v] \to \delta\mathfrak{F}[y : v]$, $\delta\mathfrak{G}[y_n : v] \to \delta\mathfrak{G}[y : v]$ sempre que $y_n \to y$ em Y. Se $\overline{y}$ é um mínimo local sujeito de $\mathfrak{F}$ sujeito a restrição $\mathfrak{G}[y] = 0$, então existe um multiplicador $\lambda \in \mathbb{R}$, tal que*

$$\delta\mathfrak{F}[\overline{y} : v] = \lambda\,\delta\mathfrak{G}[\overline{y} : v]$$

Sendo $Y = \mathscr{H}$ o espaço de Hilbert associado ao sistema quântico, sabemos que o funcional $\mathfrak{F}[|\psi\rangle] = \langle\psi|H|\psi\rangle$ satisfaz todas as condições do **teorema de Lagrange**. Seja $\mathfrak{G}[|\psi\rangle] = \langle\psi|\psi\rangle - 1$ pela condição de normalização que foi assumida na seção (2.1.1) devemos ter $\mathfrak{G}[|\psi\rangle] = 0$. Vamos calcular as variações de Gâteaux para ambos os funcionais.

$$\delta\mathfrak{F}[|\psi\rangle : |\phi\rangle] = \lim_{\varepsilon\to 0} \frac{\mathfrak{F}[|\psi\rangle + \varepsilon|\phi\rangle] - \mathfrak{F}[|\psi\rangle]}{\varepsilon}$$

Temos:

$$\mathfrak{F}[|\psi\rangle + \varepsilon|\phi\rangle] = (\langle\psi| + \varepsilon\langle\phi|)H(|\psi\rangle + \varepsilon|\phi\rangle)$$

$$= \langle\psi|H|\psi\rangle + \varepsilon\langle\psi|H|\phi\rangle + \varepsilon\langle\phi|H|\psi\rangle + \varepsilon^2\langle\phi|H|\phi\rangle$$

$$= \langle\psi|H|\psi\rangle + \varepsilon(\langle\psi|H|\phi\rangle + \langle\psi|H|\phi\rangle^*) + \varepsilon^2\langle\phi|H|\phi\rangle$$

$$= \langle\psi|H|\psi\rangle + 2\varepsilon\Re(\langle\psi|H|\phi\rangle) + \varepsilon^2\langle\phi|H|\phi\rangle$$

$$\delta\mathfrak{F}[|\psi\rangle : |\phi\rangle] = \lim_{\varepsilon\to 0}\frac{2\varepsilon\Re(\langle\psi|H|\phi\rangle) + \varepsilon^2\langle\phi|H|\phi\rangle}{\varepsilon} = 2\Re(\langle\psi|H|\phi\rangle)$$

Portanto:

$$\delta\mathfrak{F}[|\psi\rangle : |\phi\rangle] = 2\Re(\langle\psi|H|\phi\rangle) \tag{A.1}$$

Agora para o outro funcional.

$$\delta\mathfrak{G}[|\psi\rangle : |\phi\rangle] = \lim_{\varepsilon\to 0}\frac{\mathfrak{G}[|\psi\rangle + \varepsilon|\phi\rangle] - \mathfrak{G}[|\psi\rangle]}{\varepsilon} = \lim_{\varepsilon\to 0}\frac{(\langle\psi| + \varepsilon\langle\phi|)(|\psi\rangle + \varepsilon|\phi\rangle) - \langle\psi|\psi\rangle}{\varepsilon}$$

$$(\langle\psi| + \varepsilon\langle\phi|)(|\psi\rangle + \varepsilon|\phi\rangle) = \langle\psi|\psi\rangle + \varepsilon\langle\psi|\phi\rangle + \varepsilon\langle\phi|\psi\rangle + \varepsilon^2\langle\phi|\phi\rangle$$

$$= \langle\psi|\psi\rangle + \varepsilon(\langle\psi|\phi\rangle + \langle\psi|\phi\rangle^*) + \varepsilon^2\langle\phi|\phi\rangle$$

$$= \langle\psi|\psi\rangle + 2\varepsilon\Re(\langle\psi|\phi\rangle) + \varepsilon^2\langle\phi|\phi\rangle$$

$$\delta\mathfrak{G}[|\psi\rangle : |\phi\rangle] = \lim_{\varepsilon\to 0}\frac{2\varepsilon\Re(\langle\psi|\phi\rangle) + \varepsilon^2\langle\phi|\phi\rangle}{\varepsilon} = 2\Re(\langle\psi|\phi\rangle)$$

E finalmente, temos:

$$\delta\mathfrak{G}[|\psi\rangle : |\phi\rangle] = 2\Re(\langle\psi|\phi\rangle) \tag{A.2}$$

Consideramos $|\phi\rangle$ um vetor arbitrário no espaço de Hilbert.

Seja $\lambda \in \mathbb{R}$ um numero real, calculamos a diferença entre as equações (A.1) e (A.2).

$$\delta\mathfrak{F}[|\psi\rangle : |\phi\rangle] - \lambda\,\delta\mathfrak{G}[|\psi\rangle : |\phi\rangle] = 2\Re(\langle\psi|H|\phi\rangle) - 2\lambda\Re(\langle\psi|\phi\rangle) \tag{A.3}$$

Agora, consideremos que os vetores $|\psi\rangle$ são os auto-estados $|n\rangle$ do hamiltoniano H.

$$\delta\mathfrak{F}[|n\rangle : |\phi\rangle] - \lambda\,\delta\mathfrak{G}[|n\rangle : |\phi\rangle] = 2\Re(\langle n|H|\phi\rangle) - 2\lambda\Re(\langle n|\phi\rangle)$$

$$= 2E_n\Re(\langle n|\phi\rangle) - 2\lambda\Re(\langle n|\phi\rangle)$$

Escolhemos $\lambda = E_n$, portanto:

$$2E_n\Re(\langle n|\phi\rangle) - 2E_n\Re(\langle n|\phi\rangle) = 0$$

$$\implies \delta\mathfrak{F}[|n\rangle : |\phi\rangle] - \lambda\,\delta\mathfrak{G}[|n\rangle : |\phi\rangle] = 0$$

Por fim, encontramos condição:

$$\delta\mathfrak{F}[|n\rangle : |\phi\rangle] = E_n\delta\mathfrak{G}[|n\rangle : |\phi\rangle] \tag{A.4}$$

Provamos que se os os vetores $|\psi\rangle$ forem auto-estados $|n\rangle$ do operador H então existe um $\lambda = E_n$ que faz a equação (A.4) ser satisfeita, logo, pelo teorema de Lagrange, concluímos que os auto-estados são pontos de extremos do funcional $\mathfrak{F}$ e assim demonstramos o teorema de Ritz. (POINCARé, 1995)