UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
PROGRAMA DE PÓS GRADUAÇÃO EM MATEMÁTICA APLICADA

MÉTODO ANN-MoC PARA PROBLEMAS DE TRANSPORTE DE PARTÍCULAS NEUTRAS EM GEOMETRIA 1D

por

Augusto Tchantchalam

Dissertação submetida como requisito parcial

para a obtenção do grau de

Mestre em Matemática Aplicada

Prof. Dr. Pedro Henrique de Almeida Konzen

Orientador

Porto Alegre, 7 de junho de 2024.

Augusto Tchantchalam
Método ANN-MoC para Problemas de Transporte de Partículas Neutras em Geometria 1D/ Augusto Tchantchalam. – Porto Alegre, 7 de junho de 2024

Orientador: Prof. Dr. Pedro Henrique de Almeida Konzen

Dissertação de mestrado – Universidade Federal do Rio Grande do Sul
Instituto de Matemática e Estatística
Programa de Pós-Graduação em Matemática Aplicada, 7 de junho de 2024.

1. Redes neurais artificiais. 2. Método das características. 3. Transporte de partículas neutras. 4. Geometria 1D.

Método ANN-MoC para Problemas de Transporte de Partículas Neutras em Geometria 1D

por

Augusto Tchantchalam

Dissertação submetida ao Programa de Pós-Graduação em Matemática Aplicada do Instituto de Matemática e Estatística da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de

Mestre em Matemática Aplicada

Dissertação apresentada e aprovada em 07 de junho de 2024.

Orientador:

Prof. Dr. Pedro Henrique de Almeida Konzen
IME - UFRGS

Banca examinadora:

Prof. Dr. Júlio Lombaldo
IME - UFRGS

Profa. Dra. Camila Becker Picoloto
UFSM

Prof. Dr. Cássio Baissvenger Pazinatto
IFRS

Prof. Dr. Bardo Ernst Josef Bodmann
UFRGS

Prof. Dr. Lucas da Silva Oliveira
Coordenador

Porto Alegre
7 de junho de 2024

Este trabalho é dedicado a minha mãe, senhora Sadjo Djata, que lutou bastante para que eu pudesse chegar onde estou hoje.

Agradecimentos

Primeiramente a Deus, por ter me dado saúde, sabedoria e paciência durante todo o curso de mestrado.

À minha mãe, senhora Sadjo Djata, pelo seu grande esforço me ajudar no meu estudo e aos meus irmãos e familiares que deram também suas contribuições para que eu pudesse traçar toda essa trajetória acadêmica.

Ao governo Brasileiro, pela parceria com o governo da Guiné-Bissau, e em especial à PPGMAp - UFRGS, pela oportunidade de poder realizar o meu sonho de fazer o curso de pós-graduação e de ser mestre em Matemática Aplicada.

Ao Prof. Dr. Pedro Henrique de Almeida Konzen pela excelente orientação.

Aos professores da PPGMAp - UFRGS e especialmente aos professores que tive oportunidade de cursar disciplinas com eles.

Aos professores participantes da banca examinadora: Prof. Dr. Júlio Lombaldo, Prof. Dra. Camila Becker Picoloto e Prof. Dr. Cássio Baissvenger Pazinatto pelo tempo, pelas valiosas colaborações e sugestões.

Aos colegas do grupo de pesquisa N3, pelas reflexões, críticas e sugestões.

À CAPES, pelo apoio financeiro com a manutenção da bolsa de mestrado.

"A fé aperfeiçoa a condição de entendimento da razão e a razão é o argumento aceito por todos os homens para esclarecer os assuntos da fé."
(Santo Tomás de Aquino)

Resumo

O fenômeno de transporte de partículas neutras aparece em muitas importantes aplicações na engenharia e na medicina, tendo como grandes áreas as modelagens do transporte radiativo e o de neutrons. O modelo matemático fundamental baseia-se na equação linear de Boltzmann, uma equação integro-diferencial de primeira ordem. Nesse contexto, apresentamos o método ANN-MoC como operador integral no procedimento de resolver problemas de transporte de partículas neutras em geometria unidimensional (1D). O novo método consiste no acoplamento de uma rede neural artificial (ANN, do inglês, *artificial neural network*) com o método das características (MoC, do inglês, *method of characteristics*). Este fornece uma forma explícita da solução, mas que dependente de estimativas do fluxo escalar de partículas. Seguindo um esquema de iteração de fonte, uma rede neural artificial (ANN) é treinada para fornecer estimativas deste fluxo em qualquer ponto do domínio computacional. Trata-se de uma alternativa à aplicação de técnicas clássicas de interpolação e aproximação de funções. O ANN-MoC é um método conceitual que busca usufruir da flexibilidade, adaptabilidade e boas propriedades de aproximação de funções por ANNs. A apresentação desse novo método é acompanhada de uma série de estudos de casos que buscam analisar suas vantagens e desvantagens. Comparações com soluções manufaturadas e com as clássicas variantes Linear-MoC (interpolação linear) e Quadrática-MoC (interpolação quadrática) são apresentadas e discutidas para três problemas de transporte de partículas neutras em geometria 1D. Em geral, o treinamento (calibração) e a própria aplicação (cálculo) de uma ANN é computacionalmente mais custosa do que a de polinômios lineares ou quadráticos. Isso implica que o método ANN-MoC tenha um custo computacional maior que suas variantes clássicas. O ANN-MoC apresentou resultados com boa precisão gráfica para todos os problemas estudados, embora pareça ser menos preciso que as variantes comparadas. Mesmo que o método proposto possa ser computacionalmente mais custoso ou menos preciso que suas variantes clássicas, ele abre um novo paradigma no que oferece maior flexibilidade e adaptabilidade. Principalmente, o método ANN-MoC permite o acoplamento no treinamento das ANNs de dados (fornecidos *a priori* ou *in loco*), o que pode ser futuramente explorado na resolução de problemas diretos e inversos de transporte de partículas neutras.

Palavras-chave: Redes Neurais Artificiais; Método das Características; Transporte de Partículas Neutras; Geometria 1D.

Abstract

The phenomenon of neutral particle transport appears in many important applications in engineering and medicine, with major areas including the modeling of radiative transport and neutron transport. The fundamental mathematical model is based on the linear Boltzmann equation, a first-order integro-differential equation. In this context, we present the ANN-MoC method as integral operator to solve neutral particle transport problems in one-dimensional (1D) geometry. This novel approach involves coupling an artificial neural network (ANN) with the method of characteristics (MoC). The MoC provides an explicit form of the solution, but it depends on estimates of the scalar flux of particles. Following a source iteration scheme, an ANN is trained to provide estimates of this flux at any point in the computational domain. This approach serves as an alternative to the application of classical techniques for function interpolation and approximation. The ANN-MoC is a conceptual method that seeks to leverage the flexibility, adaptability, and good function approximation properties of ANNs. The presentation of this novel method is accompanied by a series of case studies that aim to analyze its advantages and disadvantages. Comparisons with manufactured solutions and with the classic Linear-MoC (linear interpolation) and Quadratic-MoC (quadratic interpolation) variants are presented and discussed for three neutral particle transport problems in 1D geometry. Generally, the training (calibration) and the actual application (calculation) of an ANN is computationally more expensive than that of linear or quadratic polynomials. This implies that the ANN-MoC method has a higher computational cost than its classical variants. The ANN-MoC showed results with good graphical accuracy for all the studied problems, although it appears to be less accurate than the compared variants. Even if the proposed method may be more computationally costly or less accurate than its classical variants, it opens a new paradigm by offering greater flexibility and adaptability. Importantly, the ANN-MoC method allows for the coupling of data (provided *a priori* or *in situ*) in the training of the ANNs, which can be further explored in the resolution of direct and inverse neutral particle transport problems.

Keywords: Artificial Neural Networks; Method of Characteristics; Neutral Particle Transport; 1D Geometry.

Lista de Figuras

Figura 1 – Neurônio biológico. 25
Figura 2 – Unidade de processamento do tipo perceptron. 31
Figura 3 – Função de ativação Limiar ou Heaviside. 32
Figura 4 – Função de ativação Linear por Partes com $\alpha = 1$. 32
Figura 5 – Função de ativação sigmoide (ou logística), $\alpha = 1$. 33
Figura 6 – Função de ativação tangente hiperbólica, $a, b = 1$. 34
Figura 7 – Função de ativação ReLU. 34
Figura 8 – Função de ativação Softplus. 35
Figura 9 – Rede do tipo perceptron multicamadas. 36
Figura 10 – Estimativa do fluxo em uma malha 1D. 45
Figura 11 – Comparação entre resultados esperados $y = \hat{\Psi}(x)$ e estimados $y = \tilde{\Psi}(x)$ para redes com arquitetura $1 - 200 \times 3 - 1$ e treinamento com $n_s = 40$. Comparações com diferentes funções de ativação. 54
Figura 12 – Comparação entre os valores esperados $\hat{\Psi}(x)$ e os valores estimados $\tilde{\Psi}(x)$ com uma rede $\{1 - 200 \times 3 - 1\}^{40}$, função $y = \text{relu}(v)$ e função $y = \text{softplus}(v)$ como funções de ativação nas camadas ocultas e de saída, respectivamente. 55
Figura 13 – Problema de Transporte 1. Comparação da solução ANN-MoC $y = \tilde{\Psi}(x)$ com a solução esperada $y = \hat{\Psi}(x)$ para $x \in \mathcal{D} = [0, 1]$. Rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \tanh(v)$ e $y = \text{id}(x)$ nas camadas escondidas e camada de saída, $N = 2$. 61
Figura 14 – Problema de Transporte 1. Comparação da solução ANN-MoC $y = \tilde{\Psi}(x)$ com a solução esperada $y = \hat{\Psi}(x)$ para $x \in \mathcal{D} = [0, 1]$. Rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(x)$ nas camadas escondidas e camada de saída, $N = 2$. 65
Figura 15 – Problema de Transporte 1. Comparações da solução ANN-MoC $y = \tilde{\Psi}(x)$ com a solução esperada $y = \hat{\Psi}(x)$ para $x \in \mathcal{D} = [0, 1]$. Rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(x)$ nas camadas escondidas e camada de saída, $N = 2$. . . 67
Figura 16 – Problema de Transporte 1. Comparações da solução ANN-MoC $y = \tilde{\Psi}(x)$ com variantes clássicas Linear-MoC e Quadrática-MoC. ANN-MoC com MLP de arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(x)$ nas camadas escondidas e camada de saída, $N = 2$. . . 68
Figura 17 – Problema de Transporte 2. Aproximação das variantes ANN-MoC, Linear e Quadrática cujas funções de ativação ReLU nas camadas ocultas e Softplus na saída com arquitetura $1 - 200 \times 3 - 1$. 72

Figura 18 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$. 80
Figura 19 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$, $n_s = 50$ e $N = 2, 8$. . . 80
Figura 20 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \mathrm{relu}(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$, $n_s = 10, 25$ e $N = 2, 4, 8$. 81
Figura 21 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \mathrm{relu}(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$, $n_s = 50$ e $N = 2, 8$. . . 81
Figura 22 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \mathrm{relu}(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.5, 0.75$ e $\sigma_t = 1$, $n_s = 100$ e $N = 4$. 82

Lista de Tabelas

Tabela 1 – Testes de treinamento de redes para o estudo de aproximação de função. Arquiteturas de rede $1 - n_n \times n_h - 1$, camadas escondidas com função de ativação $y = \tanh(v)$ e saída com $y = \mathrm{id}(v)$. Os valores tabelados são $\bar{n}_e \backslash \bar{t}_c$, número médio de épocas e tempo de execução médio para atingir o critério de parada $\varepsilon < 10^{-6}$. 51

Tabela 2 – Treinamento da rede usando diferentes funções da ativação na camada escondida. Valores tabelados são $\bar{n} \backslash \bar{t}_c$ número médio de épocas e tempo de execução médio. Critério de parada para o treinamento $\varepsilon < 10^{-6}$. . . 52

Tabela 3 – Testes de treinamento de redes com função de ativação Softplus para camada de saída e função ReLU nas camadas ocultas. Valores tabelados são $\bar{n} \backslash \bar{t}_c$ número médio de épocas e tempo de execução médio. Critério de parada para o treinamento $\varepsilon < 10^{-6}$. 56

Tabela 4 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8$ e $N = 2$. 58

Tabela 5 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 10$ e $N = 2, 8$. 59

Tabela 6 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 25, 50, 100$ e $N = 10, 20, 40$. 60

Tabela 7 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \mathrm{relu}(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8, 10$ e $N = 2, 8$. 62

Tabela 8 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \mathrm{relu}(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 25, 50, 100$ e $N = 10, 20, 40$. 63

Tabela 9 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 25, 50, 100, 200$ e $N = 2, 10, 20$. 64

Tabela 10 – Problema de Transporte 1. Comparações entre os resultados do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. Modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$, $y = \text{softplus}(v)$ nas camadas escondidas e de saída. 66

Tabela 11 – Problema de Transporte 2. Comparações entre os resultados do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. ANN-MoC com modelo de rede MLP de arquitetura $1 - 200 \times 3 - 1$ e funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e da saída, respectivamente. Casos $n_s = 10, 25, 50$ pontos de colocação. 70

Tabela 12 – Problema de Transporte 2. Comparações entre os resultados do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. ANN-MoC com modelo de rede MLP de arquitetura $1 - 200 \times 3 - 1$ e funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e da saída, respectivamente. Casos $n_s = 100, 150, 200$ pontos de colocação. 71

Tabela 13 – Problema de Transporte 3. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \text{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8, 25, 50, 100, 200$ e $N = 2, 4, 8$. 74

Tabela 14 – Problema de Transporte 3. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \text{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8, 25, 50, 100, 200$ e $N = 2, 4, 8$. 75

Tabela 15 – Problema de Transporte 3. Comparação de resultados com as variantes Quadrática-MoC, Linear-MoC e ANN-MoC, fixados $\kappa = 0.9$ e $\sigma_s = 0.1$. Rede de arquitetura $1 - 200 \times 3 - 1$. 77

Tabela 16 – Problema de Transporte 3. Comparação de resultados com as variantes Quadrática-MoC, Linear-MoC e ANN-MoC, fixados $\kappa = 0.5$ e $\sigma_s = 0.5$. Rede de arquitetura $1 - 200 \times 3 - 1$. 78

Tabela 17 – Problema de Transporte 3. Comparação de resultados com as variantes Quadrática-MoC, Linear-MoC e ANN-MoC, fixados $\kappa = 0.1$ e $\sigma_s = 0.9$. Rede de arquitetura $1 - 200 \times 3 - 1$. 79

Sumário

1 INTRODUÇÃO 21
1.1 Modelagem do Transporte de Partículas Neutras 22
1.2 Métodos para Problemas de Transporte 23
1.3 Redes Neurais Artificiais 24
1.4 Estrutura da Dissertação 26

2 REDES NEURAIS ARTIFICIAIS 29
2.1 Perceptron 30
2.2 Funções de Ativação 31
2.3 Perceptron Multicamadas 35
2.4 Treinamento de RNAs 37
2.4.1 Método do Gradiente Descendente 39
2.4.2 Método Adam 40

3 MÉTODO ANN-MOC 43
3.1 MoC Aplicado ao Problema de Transporte 43
3.2 Estimativa do Fluxo Escalar por RNA 46
3.2.1 Algoritmo ANN-MoC 47
3.2.2 Detalhes de Implementação 48

4 RESULTADOS 49
4.1 Estudo de Aproximação de Função Exponencial por RNA . . . 49
4.2 Problema de Transporte 1: Solução Manufaturada Exponencial 57
4.3 Problema de Transporte 2: Solução Manufaturada Dependente da Direção 68
4.4 Problema de Transporte 3: Solução Manufatura Trigonométrica 72

5 CONSIDERAÇÕES FINAIS 83

REFERÊNCIAS 85

1 Introdução

O fenômeno de transporte de partículas neutras aparece em muitas importantes aplicações na engenharia e na medicina. A modelagem fundamental baseia-se na equação linear de Boltzmann, uma equação integro-diferencial de primeira ordem. Por exemplo, a modelagem do transporte de calor por radiação [40] envolve o transporte de fótons e tem aplicações industriais na manufaturada em altas temperaturas [14], na medicina ótica, entre outras [1, 28, 61]. Outro exemplo é o transporte de nêutrons com aplicações na geração de energia nuclear e também na medicina [33, 58]. Em geral, não é possível obter-se soluções analíticas, o que faz com que soluções numéricas aproximadas tenham importante relevância.

Nesse contexto, propomos o método ANN-MoC para a solução de problemas de transporte de partículas neutras. O novo método consiste no acoplamento de uma rede neural artificial (ANN, do inglês, *artificial neural network*) com o método das características (MoC, do inglês, *method of characteristics*) [12]. Este fornece uma forma explícita da solução, mas que dependente do fluxo escalar de partículas. Seguindo um esquema de iteração de fonte [2], uma rede neural artificial (ANN, do inglês, *artificial neural network*) é treinada para fornecer estimativas deste fluxo em qualquer ponto do domínio computacional a partir do ponto conhecido. Trata-se de uma alternativa à aplicação de técnicas clássicas de interpolação e aproximação de funções. O ANN-MoC é um método conceitual que busca usufruir da flexibilidade, adaptabilidade e boas propriedades de aproximação de funções por ANNs.

ANNs são técnicas de aprendizagem profunda (*deep learning*), uma subárea do aprendizado de máquina (*machine learning*) e da inteligência artificial (IA) [15]. A IA está relacionada com o desenvolvimento de paradigmas e algoritmos que buscam permitir que máquinas realizem atividades cognitivas. Uma RNA é uma máquina de processamento maciçamente paralelo e distribuído, constituída pela composição de unidades de processamento simples (neurônios artificiais). O conhecimento é adquiro através de um processo de treinamento (aprendizado), que consiste na calibração dos parâmetros da rede. São estes parâmetros que armazenam o conhecimento [19, 20].

O método ANN-MoC explora a propriedade de que algumas RNAs são aproximadoras universais, em particular redes do tipo perceptron multicamadas (PMC) [19]. Enquanto que, de forma iterativa, soluções discretas do problema de transporte são obtidas pelo MoC, uma RNA é treinada com base nestes dados para fornecer estimativas do fluxo escalar médio em qualquer ponto do domínio computacional. No que o método ANN-MoC é apresentado, também buscamos avaliá-lo para diferentes problemas de transporte de

partículas neutras em geometria unidimensional (1D). Casos testes foram selecionados de forma a realçar as vantagens e desvantagens do método proposto.

1.1 Modelagem do Transporte de Partículas Neutras

A origem da teoria de transporte remonta a mais de um século, com a equação de Boltzmann, que foi inicialmente formulada para o estudo da teoria cinética dos gases de fora do equilíbrio [33, 29]. Nos anos 1930, o estudo do transporte de radiação em atmosferas estelares levou a diversas soluções analíticas para problemas de transporte. No entanto, esses estudos se limitavam a geometrias unidimensionais em meios semi-infinitos. O interesse em resolver problemas de transporte de partículas neutras em diversas configurações geométricas só surgiu com o desenvolvimento dos reatores nucleares em cadeia na década de 1940, quando essas questões se tornaram relevantes para a modelagem de reatores e blindagem contra radiação. Desde os anos 1940, têm sido explorados diversos métodos para abordar problemas transporte [33, 5].

Em geometria unidimensional (1D), o transporte de partículas neutras em um meio participativo com espalhamento isotrópico pode ser modelado como segue [40, 33]

$$\forall \mu = \cos(\theta) \in (-1,1) \setminus \{0\} : \ \mu \cdot \frac{\partial}{\partial x} I(x,\mu) + \sigma_t I = \sigma_s \Psi(x) + q(x,\mu), \ \forall x \in \mathcal{D}, \quad (1.1a)$$

$$\forall \mu > 0 : \ I(a,\mu) = I_a; \ \forall \mu < 0 : \ I(b,\mu) = I_b. \quad (1.1b)$$

onde θ é o ângulo que a direção do movimento das partículas forma com o eixo x. A direção μ, definida como $\mu = \cos(\theta)$, representa o cosseno desse ângulo, variando entre -1 e 1. Por exemplo, em uma aplicação de transporte radiativo, $I(x,\mu)$ (W/sr) é a intensidade de radiação no ponto $x \in \mathcal{D} = [a,b]$ e na direção $-1 < \mu < 1$, $\mu \neq 0$. O coeficiente de absorção total é denotado por $\sigma_t = \kappa + \sigma_s$ (1/m), onde κ (1/m) e σ_s (1/m) são os coeficientes macroscópicos de absorção e de espalhamento, respectivamente. As fontes são denotadas por $q(x,\mu)$ (W/(m $\cdot$ sr)) no domínio e por I_a e I_b (W/sr) nas fronteiras. Aqui, assumi-se fronteiras não reflexivas. O fluxo escalar médio (W/sr)sem anisotropia é denotado por

$$\Psi(x) := \frac{1}{2} \int_{-1}^{1} I(x,\mu)\, d\mu. \quad (1.2)$$

Existem várias maneiras de resolver o problema (1.1) sendo o método das ordenadas discretas (DOM) uma das abordagens mais empregadas [40]. Neste, o problema é aproximado usando um conjunto finito de direções específicas, resultando em um sistema de equações apenas para essas direções discretas. Essas equações (1.1) e (1.2) podem ser tratadas separadamente usando a estratégia de iteração de fonte (IF). O sistema é resolvido iterativamente para estimativas do fluxo escalar médio $\Psi(x) \approx \Psi^{(j)}(x)$ (com j indicando o passo da iteração), até que um critério de parada seja atingido. Cada IF

produz um conjunto de equações diferenciais parciais lineares de primeira ordem, as quais podem ser resolvidas usando o método das características[1] (MdC). Para fazer isso, é necessário calcular uma integral que depende da estimativa de Ψ, que é o foco principal nesta dissertação e é apresentado em detalhe no Capítulo 3.

1.2 Métodos para Problemas de Transporte

Existem vários métodos numéricos para a solução de problemas de transporte de partículas neutras. Por exemplo, o Método de Monte Carlo [56], é amplamente usado tanto para cálculos de transporte de nêutrons quanto de fótons. Tornou-se um dos métodos de referência devido a sua capacidade de incorporar todos os efeitos relevantes do fenômeno de transporte radiativa em uma simulação sem recorrer a aproximações [24]. Sua desvantagem está relacionada à necessidade de recursos computacionais de alto desempenho [58, 39, 24, 27].

Alternativamente, temos os métodos determinísticos em que soluções aproximadas são obtidas a partir do modelo matemático, usualmente um sistema de equações integro-diferenciais, como em (1.1). A vantagem aqui é a existência de vários métodos que vão desde mais precisos a computacionalmente mais eficientes, métodos como Método da Probabilidade de Colisão, Métodos dos Harmônicos Esféricos e Método das Ordinadas Discretas. Entretanto, essas aproximações numéricas têm o desafio de preservar propriedades físicas importantes do transporte de partículas. Por exemplo, as características de como uma superfície reflete ou emite radiação podem mudar dependendo do comprimento de onda da luz. Além disso, a radiação pode ser dispersa em várias direções de maneiras diferentes, dependendo das partículas no meio, e as superfícies podem refletir ou emitir radiação de formas diferentes dependendo do ângulo da luz que incide sobre elas. Finalmente, as propriedades de absorção ou emissão de radiação dos materiais podem mudar dependendo da sua posição ou temperatura no espaço. Esses fatores adicionais tornam os cálculos baseados em modelos matemáticos determinísticos mais desafiadores.

As técnicas comuns que podem ser utilizadas para resolver os modelos matemáticos de transporte normalmente dependem de diversos graus de aproximação. Uma das técnicas mais utilizadas é o método das ordenadas discretas (MOD), em que o fluxo escalar é aproximada por uma quadratura numérica. A dependência nas direções pelos nós da quadratura gera um sistema com isso. A precisão da solução obtida está associada ao número de pares (pesos e nodos) da quadratura [24].

Resultante da aplicação do MOD é um sistema acoplado de equações diferenciais parciais lineares de primeira ordem. Este pode ser resolvido por técnicas clássicas de

[1] Em inglês, method of characteristics (MoC)

discretização, como o método de diferenças finitas (MDF), método de volumes finitos (MVF) e métodos de elementos finitos (MEF).

Usando a técnica de iteração de fonte (IF), cada passo da iteração consiste em um sistema desacoplado de equações diferenciais parciais de primeira ordem. Cada equação modela a intensidade radiativa em um dada direção e pode ser resolvida, também, por métodos de discretização. Alternativamente, o método das características (MdC) [50] pode ser aplicado. Este tem a vantagem de fornecer um forma explícita da solução na direção associada a equação.

Ao leitor que deseja aprofundar-se sobre esses e mais outros métodos, recomendamos os seguintes livros e artigos [12, 32, 13, 50, 23, 39, 27, 58, 40, 33, 24].

1.3 Redes Neurais Artificiais

Modelos de sistemas nervosos biológicos e/ou do próprio ser humano inspiraram estruturas de redes neurais artificiais (RNAs), tendo componentes computacionais ou unidades de processamento, que são denominados de neurônios artificiais. Esses componentes computacionais têm a capacidade de adquirir e manter conhecimento (baseado em informações), interligadas por uma série de interconexões (sinapses artificiais), implementadas por vetores e matrizes de pesos sinápticos [55]. De modo geral, uma RNA funcionava como máquina projetada para modelar o funcionamento do cérebro, isto é, de que modo ele realiza uma tarefa particular ou função de interesse. Para que funcione, utilizam-se componentes eletrônicos para sua implementação ou usa-se de uma simulação em computador digital. Trata-se de uma composição maciça de células computacionais simples denominadas neurônios ou unidades de processamento [19].

No aspecto do desempenho computacional não se espera que esses modelos artificiais tenham o mesmo e nem cheguem perto do desempenho que uma rede neural biológica, por várias razões [62]. Por exemplo, um dos motivos é que não entendemos completamente a forma como é o funcionamento de um neurônio biológico. Muito menos, temos a capacidade de saber as interconexões neurais que existem, também é quase impossível simular o número de neurônios e suas interconexões como existe em uma rede biológica, e suas operações no modo assíncrono natural. Estes são fatores que mostram a complexidade de uma rede neural. A Figura 1 é um exemplo de um neurônio biológico formado por núcleo, dendritos e axônio, estes permitindo a ligação com outros neurônios.

O trabalho pioneiro das RNAs da era moderna começou com McCulloch e Pitts (1943) [19]. McCulloch que foi um psiquiatra e também neuroanatomista, que por aproximadamente 20 anos dedicou-se à forma de como seria a representação de um evento de sistema nervoso. Por outro lado, o seu companheiro de trabalho, Pitts, era um matemático. O resultado mais importante dessa colaboração foi o artigo de 1943, onde descrevem um

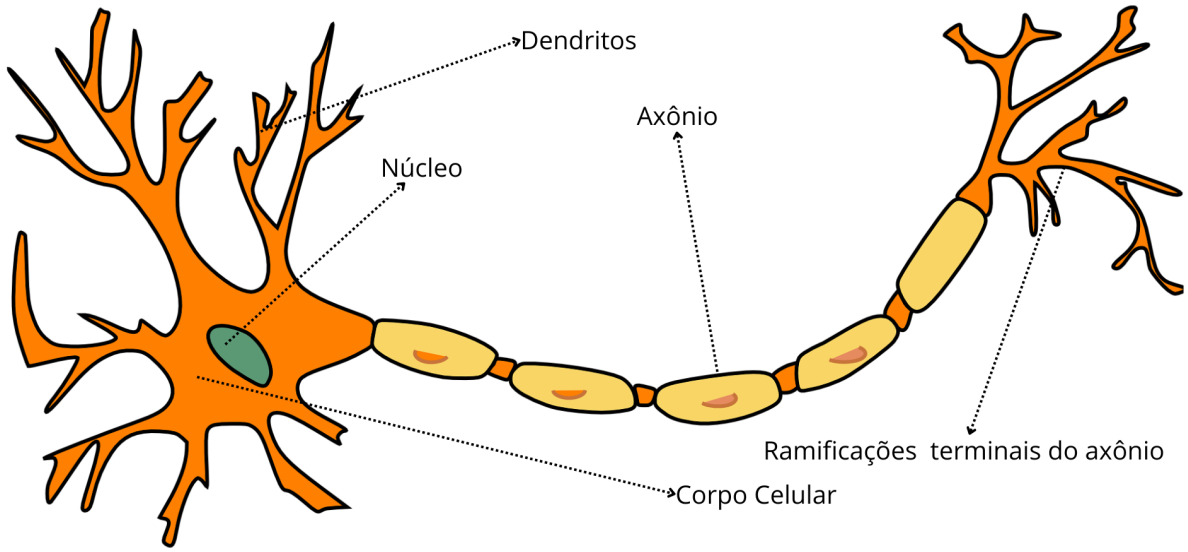


Figura 1 – Neurônio biológico.

cálculo lógico das redes neurais que envolvia os estudos de neurofisiologia e da lógica matemática.

Graças ao trabalho de McCulloch e Pitts, von Neumann pôde usar as chaves de atraso idealizado, que originou o neurônio de McCulloch e Pitts para construção do EDVAC (*Electronic Discrete Variable Automatic Computer*) que foi desenvolvido a partir do ENIAC (*Electronic Numerical Integrator and Computer*) [19], o primeiro computador eletrônico construído na Escola de Engenharia Elétrica Moore da University of Pensylvania.

Anos depois, em 1967, o livro de Minsky intitulado "*Computation: Finite and Infinite Machines*" contribuiu para a extensão dos resultados de 1943 de McCulloch e Pitts, e dessa forma, os colocou no contexto da teoria dos autômatos e da teoria de computação. Mas, antes do lançamento deste livro, 15 anos após a publicação do artigo de McCulloch e Pitts, o pesquisador Rosenblatt publicou seu trabalho tratando sobre *perceptron*, com uma abordagem nova que envolve o problema de reconhecimento de padrões, em 1958. Este trabalho é conhecido também como teorema da convergência do perceptron, onde a primeira demonstração foi delineada por ele mesmo. Em seguida, em 1960, foi introduzido o algoritmo do mínimo quadrado médio (*least mean squares method*, LMS) por Widrow e Hoff, que foi utilizado na formulação do Adaline (elemento linear adaptativo) [19].

Agora, no que tange à inteligência artificial e como disciplina especifica, ela foi implementada na década de 1950 pela primeira vez. Nessa primeira tentativa foi um fracasso devido a falta de uma técnica eficiente de treinamento, o que foi resolvido, em parte, com o surgimento da técnica de retro-propagação. Entretanto, outro fracasso aconteceu, pela segunda vez, por causa da incapacidade de formar suas próprias representações internas, na década de 1980, o que posteriormente foi solucionado graças à consolidação do aprendizado

profundo e pela disponibilidade de computadores mais potentes. Já na década de 1990, o tremendo aumento no poder de computação das máquinas finalmente contribuiu para o sucesso da inteligência artificial, capacitando-a para a resolução de diversos problemas de nível industrial, como reconhecimento de imagem, reconhecimento de fala, processamento de linguagem natural, reconhecimento de padrões, previsão, classificação, carros autônomos, automação robótica e assim por diante [38].

Como definido por Haykin (2001) [19], uma rede neural é uma máquina adaptativa, isto é, *uma rede neural é um processamento maciçamente paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso*. A semelhança com o cérebro se dá em dois aspectos: primeiro, a rede adquire o conhecimento a partir do seu ambiente por meio de um processo de *aprendizagem*; segundo, com conhecimentos já adqueridos, as forças de conexão entre neurônios, chamadas de *pesos sinápticos*, são usados com intuito de armazenar o conhecimento adquerido. Outras definições complementares são dadas em [46, 17, 8]. Por exemplo [46] fornece uma noção mais abrangente, no qual afirma que, (...) *uma rede neural é uma rede de funções em que a sincronização pode ser considerada explicitamente ou não.*

Assim, podemos definir uma rede neural é uma estrutura para processamento de informação que tem a caraterística que unidades de processamento simples se encontram interconectadas de forma multípla com os demais unidades do sistema, que têm a propensão. Portanto o sistema tem caraterísticas coletivas, que refletem propriedades emergentes não presente em cada componente que compõe o sistema como um todo. Uma implementação de uma rede neural roda num computador monoprocessador, portanto não há paralelismo ou distribuição conforme a definição do Haykin (2009).

A definição de redes neurais artificiais mais aceita nas discussões entre cientistas de computação, especialistas em inteligência artificial e matemáticos é de Haykin [19], mas em contrapartida, um artigo publicado em Procedia Computer Science em 2011 [16], sugere que a definição de Haykin é limitada, alegando que *muitas das definições explicam a RNA referindo-se a gráficos em vez de fornecer definições matemáticas bem explicadas; portanto, gráficos ponderados enganosos (como em redes de problema de fluxo de custo mínimo) se encaixam na definição de RNA*. A definição do artigo [16] é mais ampla e detalhada, mas de certa forma, não foge totalmente das definições mais aceitas.

1.4 Estrutura da Dissertação

No Capítulo 2, apresentamos uma breve contextualização de revisão bibliográfica de redes neurais artificiais, em que visa mostrar a forma como é treinada uma rede, sua arquitetura, sua estrutura, isto é, o conjunto de dados e os parâmetros necessários para ela

funcionar, também vemos algumas funções de ativação que serão usadas em nossos testes numéricos. Apresentamos no Capítulo 3, o método ANN-MoC, desenvolvido para resolver desafios relacionados aos problemas de transporte unidimensional de partículas neutras [26]. No Capítulo 4, encontram-se os testes numéricos selecionados de forma a discutir sobre as vantagens e desvantagens do método proposto. No Capítulo 5, encontram-se nossas considerações finais.

2 Redes Neurais Artificiais

No princípio, um modelo de rede neural artificial (RNA) foi um sistema concebido para emular o modo como o cérebro executa uma tarefa ou função específica de interesse. Geralmente, a rede é construída utilizando componentes eletrônicos ou é simulada em software em um computador digital. Para alcançar eficiência, as redes neurais empregam uma extensa rede de unidades de processamento simples, conhecidas como neurônios [20, 45].

John Denker [22], destacou que as redes neurais são uma alternativa secundária para a resolução de uma variedade de tarefas. No entanto, conforme ressalta [49] é necessário realizar ajustes para configurar adequadamente a estrutura da rede e garantir a convergência a um ponto próximo ao ótimo global no espaço de pesos. Desse modo, [22] enfatiza que a opção principal na visão de John Denker seria encontrar e aplicar regras específicas ou algoritmos ideais para cada problema individual, porém esse processo é muitas vezes dispendioso e consome tempo. Assim, o método secundário baseado no aprendizado, por exemplo, fornece uma alternativa prática e viável para resolver problemas de maneira mais eficiente. No entanto, nos trabalhos [20, 11, 59] observa-se que as redes neurais não conseguem resolver sozinhas o problema por completo. Elas requerem uma integração em um sistema de engenharia consistente. Em detalhes, um problema complexo é desmembrado em tarefas mais simples, e as redes neurais são designadas a lidar com um subconjunto dessas tarefas que se alinham com suas habilidades específicas.

Uma RNA é composta por vários nós ou unidades conectados por ligações, cada um com um peso numérico associado. Esses pesos são fundamentais para o armazenamento de informações de longo prazo na rede. O processo de aprendizado geralmente se dá pela calibração dos pesos. Alguns nós estão conectados ao ambiente externo, podendo ser identificados como unidades de entrada ou saída. A modificação dos pesos visa ajustar o comportamento de entrada e saída da rede para que se aproxime ao comportamento dos dados [49]. E, de acordo com [6] as redes neurais empregam uma composição profunda de funções de base (chamadas de funções de ativação), consulte também [41, 21]. O termo profundo significa uma sucessiva composição de funções modificadas.

Ao compor uma rede neural com interconexão complexa, ou seja, não linear de neurônios, a própria rede torna-se não linear. Essa não linearidade é de um tipo especial, estando disseminada por toda a estrutura da rede. Esse aspecto não linear é altamente relevante, especialmente quando o mecanismo físico subjacente à geração do sinal de entrada (como, por exemplo, um sinal de fala) é inerentemente não linear. A operação de um neurônio pode ser linear ou não linear [20].

2.1 Perceptron

Aqui descrevemos as configurações de apenas uma unidade de processamento do tipo perceptron, que no caso é a base para a formação de projeto de RNAs do tipo perceptron de multicamadas (MLP). Um perceptron é uma unidade de processamento de informação (neurônio artificial) que possui a tarefa fundamental para a operação de uma rede neural [19] que, por sua vez, é formada pela composição dessas unidades. A ideia originária de um perceptron tem paralelo com os neurônios biológicos [7].

Matematicamente, um perceptron é descrito pelas seguintes equações

$$u = \sum_{j=1}^{m} w_j x_j \tag{2.1}$$

e

$$y = f(u + b), \tag{2.2}$$

onde x_j é um escalar que representa um sinal de entrada, w_j é um escalar chamado de peso sináptico associado à j-ésima entrada, u é o escalar de saída do combinador linear, b é o *bias* (escalar), $f(\cdot)$ é a função de ativação e y é um escalar que representa o sinal de saída do neurônio. O *bias* b influencia na aplicação de uma transformação afim para saída u do combinador linear, fornecendo a chamada pré-ativação

$$v = u + b. \tag{2.3}$$

Para uma representação ilustrativa do perceptron, consulte a Figura 2. Cabe observar, que originalmente a função de ativação de um perceptron é a função de Heaviside. No entanto, pela relevância histórica, vamos manter essa denominação mesmo que no trabalho permitimos outras funções.

Em suma, a operação de propagação em um perceptron pode ser descrita como segue [55, 57]:

1. Apresentar um conjunto de valores ao neurônio, representando as variáveis de entrada;
2. Multiplicar cada entrada do neurônio pelo seu peso sináptico correspondente;
3. Computar a pré-ativação produzida pela soma ponderada dos sinais de entrada e do *bias*;
4. Aplicar uma função de ativação à pré-ativação.

Ainda de acordo com [55], as funções de ativação podem ser categorizadas em dois grupos fundamentais: funções parcialmente diferenciáveis e funções totalmente diferenciáveis, quando considerados seus domínios de definição completos. Com isso, prosseguimos agora para discutir os tipos de função de ativação, principalmente as de interesse para este trabalho.

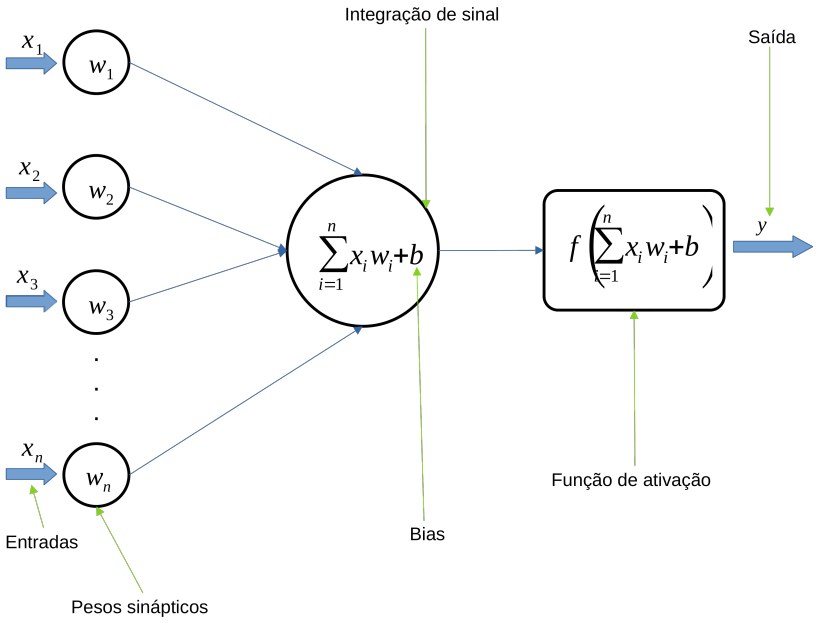


Figura 2 – Unidade de processamento do tipo perceptron.

2.2 Funções de Ativação

Ressaltamos que a função de ativação $f(v)$ tem o papel de definir a saída de um neurônio em termos da pré-ativação v. Apresentamos, aqui, seis tipos de funções de ativação:

i. *Função de Limiar ou Heaviside*. Esta função faz parte de funções parcialmente diferenciáveis (consulte Figura 3).

$$y = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases} \tag{2.4}$$

Esta é a função originalmente utilizada no perceptron [19].

ii. *Função Linear por Partes*. Este tipo de função de ativação pode ser relacionado como uma aproximação de um amplificador não linear.

$$y = \begin{cases} 1, & v \geq 1/\alpha \\ \alpha v, & 1/\alpha > v > -1/\alpha \\ 0, & v < -1/\alpha \end{cases} \tag{2.5}$$

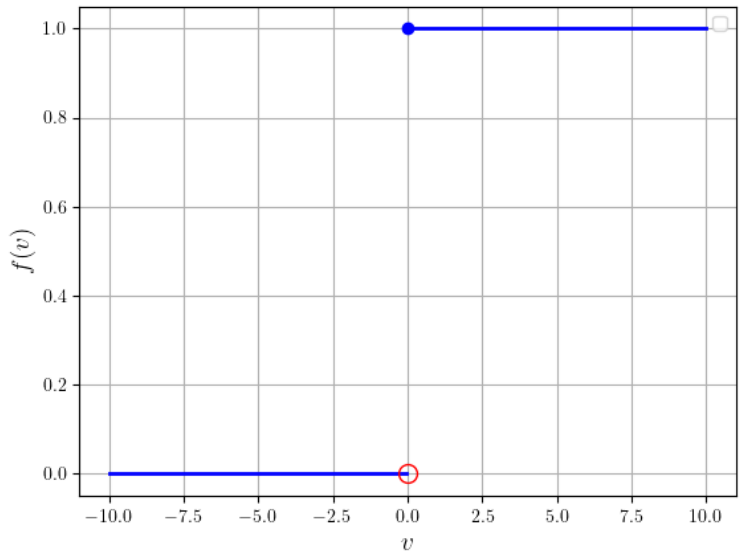


Figura 3 – Função de ativação Limiar ou Heaviside.

onde podemos assumir que o fator de amplificação dentro da região linear de operação é a unidade $\alpha = 1$. Como pode-se notar, caso a região linear de operação é mantida sem entrar em saturação, a saída é o valor do combinador linear. Para $\alpha \to \infty$, a função linear por partes se reduz à função limiar [19].

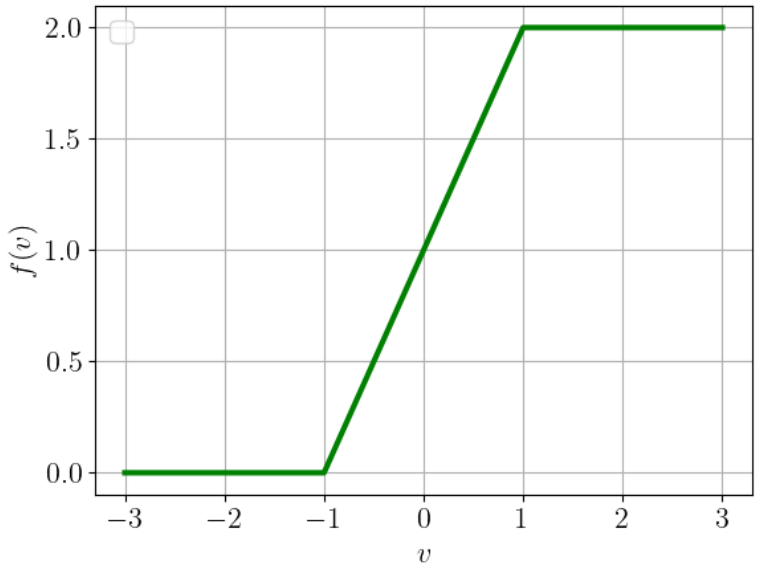


Figura 4 – Função de ativação Linear por Partes com $\alpha = 1$.

iii. *Função Sigmoide ou Logística.* O gráfico de função sigmoide possui um formato conhecido como "S", ela é definida como uma função estritamente crescente que mostra qual é o balanceamento adequado que existe entre o comportamento linear e

não linear (consulte a Figura 5).

$$f(v) = \frac{1}{1 + \exp(-\alpha v)} \tag{2.6}$$

onde α é o parâmetro de inclinação da função sigmoide. Quando a variação de parâmetro de inclinação se aproxima do infinito, no limite, a tendência da função sigmoide é de tornar a função de limiar.

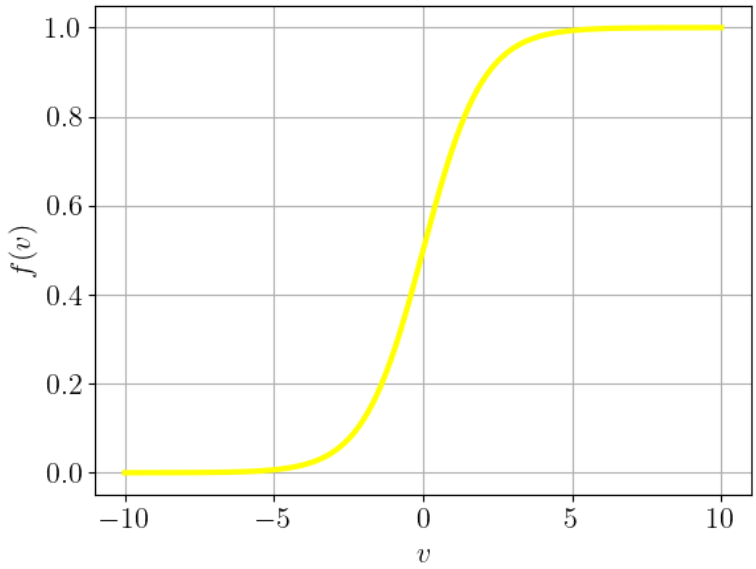


Figura 5 – Função de ativação sigmoide (ou logística), $\alpha = 1$.

iv. *Função tangente hiperbólica.* Também do tipo sigmoide, esta função está limitada entre -1 e 1, conforme a Figura 6.

$$f(v) = \tanh(\beta v) \tag{2.7}$$

v. *Função ReLU.* A função de ativação ReLU (unidade linear retificada, em inglês *rectified linear unit*) foi introduzida por Nair e Hinton em 2010 [42] e é amplamente utilizada em aplicações de aprendizagem profunda. Uma de suas principais características é permitir a ativação seletiva de neurônios [54]. Um neurônio é desativado quando sua pré-ativação é menor ou igual a zero. Consulte a Figura 7.

$$f(x) = max(0, v) = \begin{cases} v, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases} \tag{2.8}$$

vi. *Função softplus.* Proposta por Dugas et al. em 2001 [10], a função softplus é uma variação suave da função ReLU, que apresenta propriedades de suavização e um

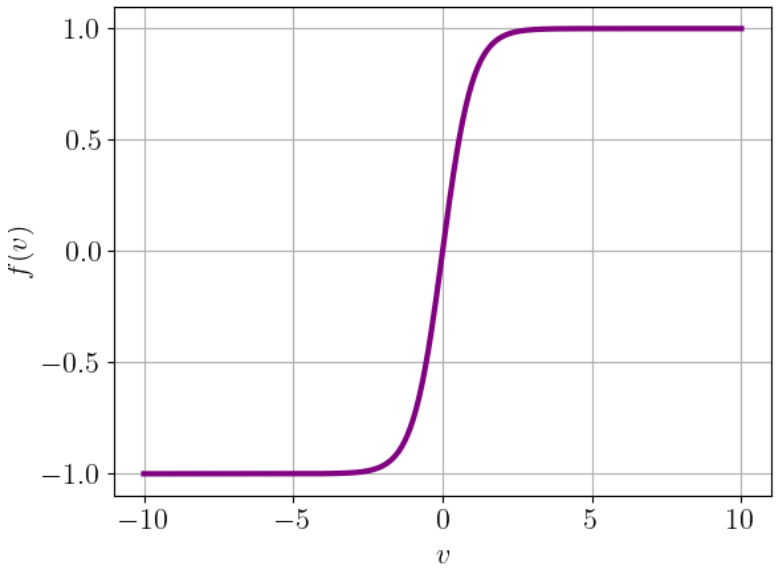


Figura 6 – Função de ativação tangente hiperbólica, $a, b = 1$.

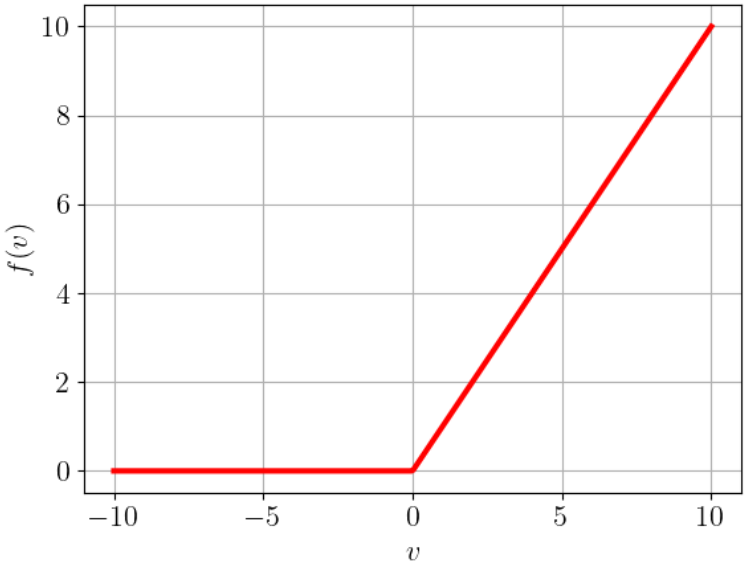


Figura 7 – Função de ativação ReLU.

gradiente diferente de zero, o que ajuda a melhorar a estabilidade e o desempenho de redes neurais profundas [43], consulte a Figura 8).

$$f(x) = (1 + e^{v}). \tag{2.9}$$

Comparativamente à ReLU, a softplus apresenta várias vantagens. Em primeiro lugar, é suave em todo o seu domínio de definição. Essa característica torna a função softplus mais estável, independentemente de ser avaliada a partir de direções positivas ou negativas, ao contrário do ReLU, que possui um gradiente descontínuo em zero.

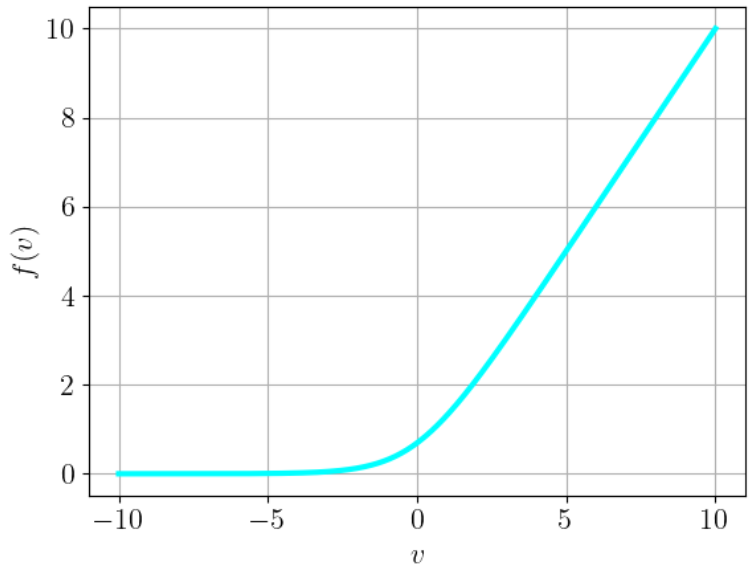


Figura 8 – Função de ativação Softplus.

Além disso, a softplus possui um gradiente não nulo quando a entrada é negativa. Enquanto a ReLU não propaga gradiente para $x < 0$, a função softplus é capaz de propagar gradientes para todas as entradas reais. Ademais, a unidade softplus supera a unidade sigmoide em vários aspectos, sendo um deles a sua derivada, que é uma função sigmoide [64]. Uma desvantagem é que a softplus demanda um custo computacional elevado na aplicação em um ponto, quando comparada a ReLU.

2.3 Perceptron Multicamadas

De acordo com [19] *o perceptron é a forma mais simples de uma rede neural usada para a classificação de padrões ditos linearmente separáveis (isto é, padrões que se encontram em lados opostos de um hiperplano*. Ele consiste basicamente de um único neurônio contendo pesos sinápticos e *bias*, e é limitado, pois foi projetado para realizar classificação de padrões que contém apenas duas classes. A realização de classificação com mais de duas classes depende do alinhamento de mais de um perceptron em camada. Além disso, para problemas não linearmente separáveis, faz-se necessário a inclusão de sucessivas composições de camadas de neurônios, conhecida como uma estrutura profunda.

O perceptron multicamadas (MLP) é uma arquitetura de RNA em que os neurônios (unidades de processamento) são organizadas em camadas conectadas, alguns dos neurônios se conectam com o mundo real para receber suas informações de entradas. E, por outro lado, outros neurônios fornecem ao mundo real as informações de saídas da rede. Todos os outros neurônios estão ocultos, i.e., recebem informação de neurônios, processam e passam a informação para outros neurônios [3]. Quando adicionamos uma ou mais camadas

ocultas, fazemos com que a rede se torne capaz de extrair estatísticas de ordem elevada [19]. Este é o conceito fundamental da aprendizagem profunda, em que simples unidades de processamento são sucessivamente compostas, formando uma estrutura capaz de capturar diversos comportamentos não lineares.

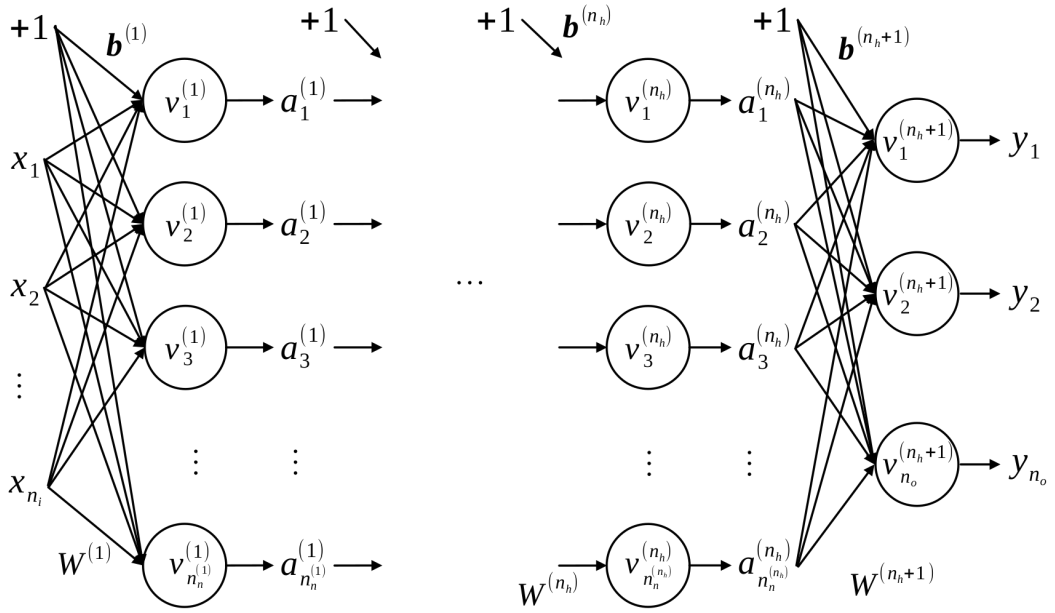


Figura 9 – Rede do tipo perceptron multicamadas.

Em uma MLP, o fluxo de processamento se dá pela propagação dos dados de entrada para a primeira camada escondida. Nela os dados são processados e os resultados da ativação são propagados para a camada seguinte. Por fim, a camada de saída, recebe as ativações da última camada escondida, processa-as e fornece a saída ativada (consulte a Figura 9). Mais precisamente, denotamos uma MLP de n_l camadas por

$$x, b^{(1)} \to n_i \times 1$$
$$W^{(1)} \to n_n^{(1)} \times n_i$$

$$\boldsymbol{y} = \mathcal{N}\left(\boldsymbol{x}; \left(W^{(l)}, \boldsymbol{b}^{(l)}, f^{(l)}\right)_{l=0}^{n_h+1}\right), \tag{2.10}$$

onde $\left(W^{(l)}, \boldsymbol{b}^{(l)}, \boldsymbol{f}^{(l)}\right)$ é a tripa de pesos, *biases* e função de ativação da l-ésima camada da rede, $l = 1, 2, \ldots, n_h + 1$. Uma rede com essa arquitetura tem uma camada de entrada, n_h camadas escondidas e uma camada de saída.

A saída da rede neural é calculada iterativamente, começando com os dados de entrada. Esses dados passam por várias camadas ocultas, onde são transformados gradualmente por operações matemáticas. Finalmente, os dados processados são enviados para a camada de saída, que produz a saída final da rede, i.e.

$$a^{(l)}, b^{(l)} \to n_l \times 1$$

$$W^{(l)} \to n_n^{(l)} \times n_n^{(l-1)}$$

$$\boldsymbol{a}^{(l)} = f^{(l)}\left(W^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \tag{2.11}$$

para $l = 0, 1, \ldots, n_h + 1$, denotando a entrada por $\boldsymbol{x} =: \boldsymbol{a}^{(0)}$ e a saída por $\boldsymbol{y} =: \boldsymbol{a}^{(n_h+1)}$. A pré-ativação da l-ésima camada é dada por

$$\boldsymbol{v}^{(l)} = W^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}. \tag{2.12}$$

2.4 Treinamento de RNAs

O treinamento de RNAs consiste na calibração de seus parâmetros (pesos e *biases*) e é classificada conforme os seguintes tipos. O treinamento supervisionado [36] é um paradigma de aprendizado de máquina que visa adquirir informações sobre a relação entre entradas e saídas de um sistema com base em um conjunto de amostras de treinamento em que cada entrada está associada a uma saída conhecida. Como a saída é considerada o rótulo dos dados de entrada ou a supervisão, essas amostras de treinamento entrada-saída são frequentemente chamadas de dados de treinamento rotulados ou dados supervisionados. Enquanto que, quando um algoritmo emprega tanto dados de treinamento supervisionado quanto não supervisionado, é classificado como um algoritmo de aprendizado semi-supervisionado. Se um algoritmo busca de forma ativa rótulos ou informações de um usuário ou instrutor durante o processo de treinamento, isso é conhecido como aprendizado ativo (ou iterativo supervisionado).

Em contrapartida, o treinamento por reforço é um problema de aprendizado comportamental. Lidar com este treinamento pode ser categorizado em dois modelos [20]:

1. abordagem clássica - o aprendizado é conduzido através de um sistema de recompensas e punições, buscando a aquisição de um comportamento.

2. abordagem moderna - emprega um método matemático conhecido como programação dinâmica, visando decidir uma ação a ser tomada ao considerar possíveis estágios futuros sem experienciá-los realmente, o foco é na elaboração de planos.

Aqui, vamos nos concentrar ao treinamento supervisionado, em que é conhecido um conjunto de treinamento com pares de entrada e saída esperada de dados. O treinamento de RNAs é desafiador, pois recai em um problema de otimização de grande porte. Consiste na calibração dos parâmetros da rede (pesos e *biases*) de forma que ela forneça as saídas esperadas para os dados de entrada correspondentes. A etapa de ajuste dos pesos é conhecida como retro-propagação (em inglês, *backpropagation*) e é baseada na resolução de um problema de minimização [53, 20, 30]. São, basicamente, três etapas: [19]: inicialização, propagação e retro-propagação. A inicialização consiste na escolha de pesos e *biases* iniciais,

usualmente, dados por uma distribuição uniforme randômica e média zero e variância unitária. Propagação é a computação das saídas estimadas pela rede para os dados de treinamento. A retro-propagação é a atualização dos pesos e *biases* de forma a minimizar o erro entre as saídas estimadas e as esperadas.

Dado um conjunto de treinamento $\{\boldsymbol{x}^{(s)}, \boldsymbol{y}^{(s)}\}_{s=1}^{n_s}$, com n_s amostras, o treinamento da rede consiste em resolver o seguinte problema de minimização

$$\min_{(W,\boldsymbol{b})} \left\{ \varepsilon := \frac{1}{n_s} \sum_{s=1}^{n_s} \varepsilon^{(s)} \left(\tilde{\boldsymbol{y}}^{(s)}, \boldsymbol{y}^{(s)} \right) \right\} \tag{2.13}$$

onde ε é uma dada função erro e $\varepsilon^{(s)}$ é uma medida do erro da saída estimada $\tilde{\boldsymbol{y}}^{(s)}$ pela saída esperada $\boldsymbol{y}^{(s)}$.

O problema de minimização pode ser resolvido por um método de declive e, de forma geral, consiste em:

1. $W, \boldsymbol{b}$ aproximações iniciais.
2. Para $e \leftarrow 1, \ldots, n_e$:
   a) $(W, \boldsymbol{b}) \leftarrow (W, \boldsymbol{b}) - l_r \boldsymbol{d} \left(\nabla_{W,\boldsymbol{b}} \varepsilon \right)$

onde, n_e é o número de épocas, l_r é uma dada taxa de aprendizagem e $\boldsymbol{d} = \boldsymbol{d} \left(\nabla_{W,\boldsymbol{b}} \varepsilon \right)$ é o vetor direção, onde

$$\nabla_{W,\boldsymbol{b}} \varepsilon := \left(\frac{\partial \varepsilon}{\partial W}, \frac{\partial \varepsilon}{\partial \boldsymbol{b}} \right) \tag{2.14}$$

$$= \frac{1}{n_s} \sum_{s=1}^{n_s} \left(\frac{\partial \varepsilon^{(s)}}{\partial W}, \frac{\partial \varepsilon^{(s)}}{\partial \boldsymbol{b}} \right) \tag{2.15}$$

O cálculo dos gradientes pode ser feito por retro-propagação. Para os pesos da última camada, temos[1]

$$\frac{\partial \varepsilon^{(s)}}{\partial W^{(n_h)}} = \frac{\partial \varepsilon^{(s)}}{\partial \tilde{\boldsymbol{y}}} \frac{\partial \tilde{\boldsymbol{y}}}{\partial \boldsymbol{v}^{(n_h+1)}} \frac{\partial \boldsymbol{v}^{(n_h+1)}}{\partial W^{(n_h)}}, \tag{2.16}$$

$$= \frac{\partial \varepsilon^{(s)}}{\partial \tilde{\boldsymbol{y}}} f' \left(\boldsymbol{v}^{(n_h+1)} \right) \frac{\partial \boldsymbol{v}^{(n_h+1)}}{\partial \boldsymbol{a}^{(n_h)}} \frac{\partial \boldsymbol{a}^{(n_h)}}{\partial \boldsymbol{v}^{(n_h)}} \frac{\partial \boldsymbol{v}^{(n_h)}}{\partial W^{(n_h)}} \tag{2.17}$$

$$= \frac{\partial \varepsilon^{(s)}}{\partial \tilde{\boldsymbol{y}}} f' \left(\boldsymbol{v}^{(n_h+1)} \right) W^{(n_h+1)} f' \left(\boldsymbol{v}^{(n_h)} \right) \boldsymbol{a}^{(n_h-1)} \tag{2.18}$$

Para os pesos da penúltima camada, temos

$$\frac{\partial \varepsilon^{(s)}}{\partial W^{(n_h)}} = \frac{\partial \varepsilon}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}^{(n_h+1)}} \frac{\partial \boldsymbol{z}^{(n_h+1)}}{\partial W^{(n_h)}}, \tag{2.19}$$

[1] Com um certo abuso de linguagem devido à álgebra matricial envolvida.

$$= \frac{\partial \varepsilon^{(s)}}{\partial \boldsymbol{y}} f'\left(\boldsymbol{z}^{(n_h+1)}\right) \frac{\partial \boldsymbol{z}^{(n_h+1)}}{\partial \boldsymbol{a}^{(n_h)}} \frac{\partial \boldsymbol{a}^{(n_h)}}{\partial \boldsymbol{z}^{(n_h)}} \frac{\partial \boldsymbol{z}^{(n_h)}}{\partial W^{(n_h)}} \tag{2.20}$$

$$= \frac{\partial \varepsilon^{(s)}}{\partial \boldsymbol{y}} f'\left(\boldsymbol{z}^{(n_h+1)}\right) W^{(n_h+1)} f'\left(\boldsymbol{z}^{(n_h)}\right) \boldsymbol{a}^{(n_h-1)} \tag{2.21}$$

e assim, sucessivamente para as demais camadas da rede. Os gradientes em relação aos *biases* podem ser calculados de forma análoga.

Na sequência, discorremos sobre alguns algoritmos de otimização comumente empregados no treinamento de RNAs.

2.4.1 Método do Gradiente Descendente

O método do gradiente descendente (MDG) é empregado para otimização multidimensional com o intuito de atingir o valor mínimo global de uma dada função. É uma técnica popular em vários modelos de aprendizado de máquina, sendo fundamental para aprimorar ou otimizar as previsões do modelo. Uma das implementações do método é conhecida como método do gradiente estocástico (SGD), que tem grande relevância no contexto das RNAs. A otimização compreende-se no cálculo do valor do erro (função da diferença entre os valores estimados e os esperados) e a alteração dos parâmetros da rede para minimizá-lo. A direção para encontrar o mínimo corresponde à direção oposta do gradiente da função de perda [53, 47, 37]. Há uma vasta literatura sobre o desenvolvimento e aplicações do método, consulte, por exemplo [44, 52, 63, 18, 4, 31, 48, 35].

Uma explanação adequada da funcionalidade de algoritmos de gradiente descendente pode ser baseado no exemplo disponível em [18]: suponha que você esteja perdido nas montanhas em meio a uma densa neblina e só possa sentir a inclinação do terreno sob seus pés. Uma estratégia eficaz para chegar ao vale mais rápido é seguir morro abaixo na direção da inclinação mais acentuada. Essa é, basicamente, a proposta do MDG. Avaliamos o gradiente local da função de erro em relação ao vetor de parâmetros $\boldsymbol{\theta} = (W, \boldsymbol{b})$ e avançamos na direção do declive mais acentuado (direção oposta do gradiente da função erro em relação a $\boldsymbol{\theta}$). Ao alcançarmos um ponto com gradiente nulo, considera-se que o mínimo foi atingido. Concretamente, inicia-se $\boldsymbol{\theta}$ com valores randômicos. Em seguida, ocorre um aprimoramento gradual, avançando um passo de cada vez, no intuito de reduzir a função de custo. Dentro do algoritmo do MGD, a questão fundamental reside no valor da taxa de aprendizagem, a qual determina o tamanho dos passos ao ajustar os parâmetros. Se essa taxa for muito pequena, o algoritmo demandará várias iterações para convergir para o mínimo, estendendo o tempo necessário para a otimização [18].

Digamos que o desafio para o algoritmo do gradiente descendente é discernir se atingiu o vale mais baixo ou apenas o ponto mais baixo em um vale superior. O ponto mais baixo no vale mais baixo é chamado de mínimo global, enquanto outros vales têm mínimos locais. Se o algoritmo se depara com um mínimo local, fica preso, indicando uma

limitação do método [44]. Enfim, o gradiente é uma medida local da declividade.

Sob outra perspectiva, e conforme explica [52], a abordagem convencional do gradiente descendente é empregada para minimizar uma função convexa diferenciável $\varepsilon(\boldsymbol{\theta})$. O gradiente de uma função diferenciável consiste no vetor composto por suas derivadas parciais em relação a cada uma de suas variáveis, i.e.

$$\nabla\varepsilon(\boldsymbol{\theta}) := \left(\frac{\partial\varepsilon(\boldsymbol{\theta})}{\partial\theta_1}, \ldots, \frac{\partial\varepsilon(\boldsymbol{\theta})}{\partial\theta_m}\right). \tag{2.22}$$

O MDG é iterativo, e inicializamos com um valor inicial de $\boldsymbol{\theta}$ (ou seja, $\boldsymbol{\theta}^{(1)} = 0$). Em cada iteração subsequente, avançamos na direção oposta ao gradiente no ponto atual. Isso significa que a etapa de atualização é:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - l_r\nabla\varepsilon(\boldsymbol{\theta}^{(t)}), \tag{2.23}$$

onde $l_r > 0$ é um parâmetro[2]. Observamos que uma das desvantagens do MDG é sua lenta convergência próxima do mínimo da função. Para propriedades de convergência do MDG no contexto de RNAs, recomendamos a leitura do livro [52].

2.4.2 Método Adam

O método Adam (MA) é um método eficaz para otimização em situações estocásticas que usa apenas informações básicas dos gradientes, ocupando pouca memória [25]. Ele ajusta automaticamente a taxa de aprendizagem para cada parâmetro com base em estimativas de como esses gradientes variam ao longo do tempo. Ou seja, o Adam vem da ideia de ajustar essas estimativas de forma adaptativa para capturar melhor o comportamento dos gradientes. O Adam oferece várias vantagens, como garantir que as atualizações nos parâmetros não sejam afetadas pela escala dos gradientes, limitar os tamanhos dos passos através de um hiper-parâmetro, não precisar de um objetivo estático, lidar bem com gradientes dispersos e realizar naturalmente um ajuste gradual no tamanho dos passos.

Dada uma função erro ε nos parâmetros $\boldsymbol{\theta}$, o algoritmo pode ser resumido como segue [25]:

1. Definir o valor de l_r (taxa de aprendizagem).
2. Definir os valores de $0 < \beta_1, \beta_2 < 1$, taxas de decaimento exponencial utilizado para calcular as estimativas dos momentos.
4. Definir $\boldsymbol{\theta}^{(0)}$, os parâmetros iniciais.
5. Inicializar os vetores do primeiro e segundo momentos:

[2] Chamado de taxa de aprendizagem, no contexto das RNAs.

- $\boldsymbol{m_1}^{(0)} \leftarrow \mathbf{0}$ (Inicializar vetor do 1º momento)
- $\boldsymbol{m_2}^{(0)} \leftarrow \mathbf{0}$ (Inicializar vetor do 2º momento)
- $t \leftarrow 0$ (contator)

6. Enquanto $\boldsymbol{\theta}^{(t)}$ não convergiu, faça:
   - $t \leftarrow t + 1$
   - $\boldsymbol{g}^{(t)} \leftarrow \nabla_{\boldsymbol{\theta}} \varepsilon(\boldsymbol{\theta}^{(t-1)})$ (Obter gradientes em relação ao objetivo estocástico no passo de tempo t)
   - $\boldsymbol{m_1}^{(t)} \leftarrow \beta_1 \cdot \boldsymbol{m_1}^{(t-1)} + (1 - \beta_1) \cdot \boldsymbol{g}^{(t)}$ (Atualizar estimativa de bias do primeiro momento)
   - $\boldsymbol{m_2}^{(t)} \leftarrow \beta_2 \cdot \boldsymbol{m_2}^{(t-1)} + (1 - \beta_2) \cdot \left(\boldsymbol{g}^{(t)}\right)^2$ (Atualizar estimativa de bias do segundo momento bruto)
   - $\hat{\boldsymbol{m_1}}^{(t)} \leftarrow \frac{\boldsymbol{m_1}^{(t)}}{1-\beta_1^t}$ (Calcular estimativa do primeiro momento corrigida para bias)
   - $\hat{\boldsymbol{m_2}}^{(t)} \leftarrow \frac{\boldsymbol{m_2}^{(t)}}{1-\beta_2^t}$ (Calcular estimativa do segundo momento bruto corrigido para bias)
   - $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - l_r \cdot \frac{\hat{\boldsymbol{m_1}}^{(t)}}{\sqrt{\hat{\boldsymbol{m_2}}^{(t)}} + \epsilon}$ (Atualizar parâmetros)
7. Retornar $\boldsymbol{\theta}^{(t)}$ (Parâmetros resultantes)

Observamos que todas as operações neste algoritmo são elemento-a-elemento. No contexto descrito, estamos lidando com uma série de funções de custo convexas representadas por $\varepsilon^{(1)}(\boldsymbol{\theta}), \varepsilon^{(2)}(\boldsymbol{\theta}), ..., \varepsilon^{(T)}(\boldsymbol{\theta})$, onde $\boldsymbol{\theta}$ representa o parâmetro a ser otimizado. Em cada momento t, nosso objetivo é prever o parâmetro $\boldsymbol{\theta}^{(t)}$ e avaliá-lo em uma função de custo específica, $\varepsilon^{(t)}$, que é desconhecida previamente. Como não conhecemos a natureza exata da sequência de funções de custo com antecedência, avaliamos a eficácia do nosso algoritmo considerando a perda acumulada. Essa perda acumulada é calculada como a soma das diferenças entre nossas previsões anteriores, $\varepsilon^{(t)}(\boldsymbol{\theta}^{(t)})$, e o melhor parâmetro fixo, $\hat{\varepsilon}^{(t)}(\boldsymbol{\theta}^{(t)})$, selecionado a partir de um conjunto viável X, ao longo de todos os passos anteriores. Para informações sobre as propriedades de convergência do método Adam, recomendamos a leitura de [25].

3 Método ANN-MoC

Neste capítulo, apresentamos o método ANN-MoC para resolver problemas de transporte de partículas neutras em geometria unidimensional [26]. Trata-se de uma nova abordagem e consiste na principal contribuição desta dissertação. Sua aplicação é feita a partir da aproximação do método de ordenadas discretas (MOD) e do esquema de iteração de fonte (IF) [40, 9]. A ideia fundamental do método proposto é utilizar o treinamento de uma rede neural artificial (ANN) para calcular a estimativa do fluxo de partícula $\Psi^{(j)}$ durante cada iteração da fonte. Este método combina o uso de uma ANN (ANN, do inglês, *artificial neural network*) com o método das características (MoC, do inglês, *method of characteristics*).

3.1 MoC Aplicado ao Problema de Transporte

O método das características (MoC) [51, 60, 40, 12] fornece uma solução explícita para equações diferenciais parciais de primeira ordem. Isso é realizado ao identificar e usar curvas específicas, chamadas características, ao longo das quais a solução é obtida. Com os dados iniciais provenientes das condições de contorno, a solução pode ser obtida através da integração da equação diferencial resultante ao longo dessas curvas características.

A aplicação do MoC à equação de transporte parte das aproximações pelo MOD e do IF [40]. Assumindo a quadratura gaussiana $\{(\mu_i, \omega_i)\}_{i=1}^{N}$, a formulação MOD do problema (1.1) é dada por

$$1 \leq i \leq N: \ \mu_i \cdot \frac{\partial}{\partial x} I_i(x) + \sigma_t I_i = \sigma_s \Psi(x) + q(x, \mu_i), \ \forall x \in \mathcal{D}, \tag{3.1a}$$

$$I_i(a) = I_a, \ \forall \mu_i > 0; \ I_i(b) = I_b, \ \forall \mu_i < 0, \tag{3.1b}$$

onde $I_i(x) = I(x, \mu_i)$ e o fluxo escalar médio é aproximado pela quadratura para problemas isotrópicos

$$\Psi(x) = \frac{1}{2} \sum_{i=1}^{N} \omega_i I_i(x). \tag{3.2}$$

O problema (3.1) consiste em um sistema de N equações diferenciais parciais de primeira ordem, acopladas pelo fluxo escalar. De forma iterativa, o desacoplamento por ser feito pelo esquema IF

$$1 \leq i \leq N: \ \mu_i \cdot \frac{\partial}{\partial x} I_i^{(j)}(x) + \sigma_t I_i^{(j)} = \sigma_s \Psi^{(j-1)}(x) + q(x, \mu_i), \ \forall x \in \mathcal{D}, \tag{3.3a}$$

$$I_i^{(j)}(a) = I_a, \ \forall \mu_i > 0; \ I_i^{(j)}(b) = I_b, \ \forall \mu_i < 0, \tag{3.3b}$$

onde $I_i^{(j)}(x) = I^{(j)}(x, \mu_i)$, $j = 1, 2, \ldots, L$, com dada aproximação inicial $\Psi^{(0)}(x)$. A j-ésima aproximação do fluxo escalar é dada por

$$\Psi^{(j)}(x) = \frac{1}{2} \sum_{i=1}^{N} \omega_i I_i^{(j)}(x). \tag{3.4}$$

Para propriedades sobre a convergência do esquema IF, recomendamos a leitura de [2].

Em cada IF, o problema (3.3) consiste em um sistema desacoplado de equações diferenciais parciais de primeira ordem com condições de contorno (3.3b). Agora utilizamos o MoC aplicando a mudança de variáveis $x(s) = x_0 + s\mu_i$. Com isso, observamos que

$$\frac{d}{ds} I_i^{(j)}(x(s)) = \frac{\partial I_i^{(j)}}{\partial x} \frac{dx}{ds} = \mu_i \frac{\partial I_i^{(j)}}{\partial x}. \tag{3.5}$$

Dessa forma, para cada $i = 1, ..., N$, a equação (3.3a) pode ser reescrita na forma

$$\frac{d}{ds} I_i^{(j)}(s) + \sigma_t I_i^{(j)}(s) = \sigma_s \Psi^{(j-1)}(s) + q(s, \mu_i), \tag{3.6}$$

onde $I_i^{(j)}(s) = I_i^{(j)}(x(s))$, e análogo para o outro termo.

Agora, (3.6) é uma equação diferencial ordinária linear e de primeira ordem em s na direção μ_i. Pelo método dos fatores integrantes, buscamos por uma função $\nu = \nu(s)$ não nula tal que

$$\nu \frac{d}{ds} I_i^{(j)}(s) + \nu \sigma_t I_i^{(j)}(s) = \nu S(s), \tag{3.7}$$

onde $S_i^{(j)}(s) = \sigma_s \Psi^{(j-1)}(s) + q(s, \mu_i)$ é o termo fonte modificado. Ainda, assumimos

$$\nu \frac{d}{ds} I_i^{(j)}(s) + \nu \sigma_t I_i^{(j)}(s) = \frac{d}{ds} \left(\nu I_i^{(j)}(s) \right) \tag{3.8}$$

$$= \frac{d}{ds}(\nu) I_i^{(j)}(s) + \nu \frac{d}{ds}(I_i^{(j)}(s)). \tag{3.9}$$

Agora, voltando a (3.7), temos que

$$I_i^{(j)}(s) \frac{d}{ds} \nu = \sigma_t \nu I_i^{(j)}(s)$$

$$\frac{d}{ds} \nu = \sigma_t \nu$$

$$\frac{1}{\nu} \frac{d}{ds} \nu = \sigma_t. \tag{3.10}$$

Integrando a Eq. (3.10) no intervalo $[0, s]$ em ambos os lados, segue que

$$\int_0^s \frac{1}{\nu} \frac{d}{ds'} \nu ds' = \int_0^s \sigma_t ds'. \tag{3.11}$$

Assumindo σ_t constante, temos

$$\ln |\nu(s)| = \sigma_t s \tag{3.12}$$

$$\nu(s) = e^{\sigma_t s}. \tag{3.13}$$

Agora, multiplicamos ambos os lados da Eq. (3.6) pelo fator integrante (neste caso o fator integrante $e^{\sigma_t s}$ é obtido), temos

$$e^{\sigma_t s}\frac{d}{ds}I_i^{(j)}(s) + \sigma_t e^{\sigma_t s} I_i^{(j)}(s) = e^{\sigma_t s}\sigma_s \Psi^{(j-1)}(s) + e^{\sigma_t s} q(s, \mu_i) \tag{3.14}$$

$$\frac{d}{ds}\left(e^{\sigma_t s} I_i^{(j)}(s)\right) = e^{\sigma_t s}\sigma_s \Psi^{(j-1)}(s) + e^{\sigma_t s} q(s, \mu_i) \tag{3.15}$$

$$\int_0^s \frac{d}{ds'}\left(e^{\sigma_t s'} I_i^{(j)}(s')\right) ds' = \int_0^s e^{\sigma_t s'}\sigma_s \Psi^{(j-1)}(s')\, ds' + \int_0^s e^{\sigma_t s'} q(s', \mu_i)\, ds' \tag{3.16}$$

aplicando o teorema fundamental de cálculo, obtemos

$$e^{\sigma_t s} I_i^{(j)}(s)\Big|_0^s = \int_0^s e^{\sigma_t s'}\sigma_s \Psi^{(j-1)}(s')\, ds' + \int_0^s e^{\sigma_t s'} q(s', \mu_i)\, ds' \tag{3.17}$$

$$e^{\sigma_t s} I_i^{(j)}(s) - e^{\sigma_t \cdot 0} I_i^{(j)}(0) = \int_0^s e^{\sigma_t s'}\sigma_s \Psi^{(j-1)}(s')\, ds' + \int_0^s e^{\sigma_t s'} q(s', \mu_i)\, ds' \tag{3.18}$$

$$e^{\sigma_t s} I_i^{(j)}(s) = I_i^{(j)}(0) + \int_0^s e^{\sigma_t s'}\sigma_s \Psi^{(j-1)}(s')\, ds' + \int_0^s e^{\sigma_t s'} q(s', \mu_i)\, ds' \tag{3.19}$$

$$I_i^{(j)}(s) = e^{-\sigma_t s} I_i^{(j)}(0) + \frac{\int_0^s e^{\sigma_t s'}\left[\sigma_s \Psi^{(j-1)}(s')\, ds' + q(s', \mu_i)\right]\, ds'}{e^{\sigma_t s}} \tag{3.20}$$

$$I_i^{(j)}(s) = e^{-\sigma_t s} I_i^{(j)}(0) + e^{-\sigma_t s}\int_0^s \left[\sigma_s \Psi^{(j-1)}(s') + q(s', \mu_i)\right] e^{\sigma_t s'}\, ds' \tag{3.21}$$

ou seja,

$$I_i^{(j)}(s) = e^{-\sigma_t s} I_i^{(j)}(0) + e^{-\sigma_t s}\int_0^s \left[\sigma_s \Psi^{(j-1)}(s') + q(s', \mu_i)\right] e^{\sigma_t s'}\, ds' \tag{3.22}$$

A determinação do valor de $\Psi^{(j)}(s)$ através da integração é desafiadora, pois frequentemente exige a avaliação dessa função em diversos pontos dentro do intervalo $s \in (0, s)$. A extensão desse intervalo pode ser considerável, especialmente levando em conta a direção μ_i [34]. A abordagem proposta pelo método ANN-MoC visa contornar essa dificuldade ao treinar uma RNA para estimar $\Psi^{(j)}$ em cada iteração da fonte.

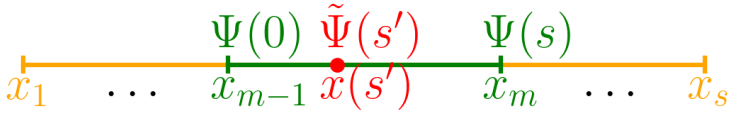


Figura 10 – Estimativa do fluxo em uma malha 1D.

Na prática, o MoC é aplicado sobre uma malha computacional $x_m = a + mh_x$, $h_x = (b-a)/n_x$, $m = 0, 1, 2, \ldots, n_x$. Sem perda de generalidade, assumindo $\mu > 0$, (3.22) fornece $I_i^{(j)}(x_m, \mu_i) = I_i^{(j)}(s_{m_i})$, $s = (x_m - x_{m-1})/\mu_i$, conhecido o valor de $I_i^{(j)}(x_{m-1}, \mu_i) = I_{i,m-1}^{(j)}(0)$ e os valores do fluxo escalar $\Psi^{(j-1)}$ nos nodos x_{m-1} e x_m. Notamos que a solução

MoC (3.22), depende da integral de $\Psi^{(j-1)}$ no intervalo $[x_{m-1}, x_m]$, o que, em geral, demanda estimativas do fluxo em pontos internos desse intervalo, consulte a Figura (10) (simplificamos a notação, dispensando o uso do índice superior (j)). Usualmente, variantes do MoC empregam técnicas clássicas de interpolação ou de aproximação de funções para o cálculo destas estimativas. O método ANN-MoC é uma alternativa, em que propomos a utilização de uma ANN para essas computações, ou seja, para implementar a operação de integração.

3.2 Estimativa do Fluxo Escalar por RNA

A proposta do método ANN-MoC, consiste em treinar uma RNA para fornecer as estimativas do fluxo escalar médio $\Psi(x)$ em qualquer ponto do domínio computacional. Assim, assumindo uma ANN do tipo perceptron multicamada, o valor estimado $\tilde{\Psi}(x) \approx \Psi(x)$ é dado por

$$\tilde{\Psi}(x) = \mathcal{N}\left(x; \left\{\left(W^{(l)}, \boldsymbol{b}^{(l)}, \boldsymbol{f}^{(l)}\right)\right\}_{l=1}^{n_h+1}\right), \tag{3.23}$$

onde $\left(W^{(l)}, \boldsymbol{b}^{(l)}, \boldsymbol{f}^{(l)}\right)$ denota a tripla de pesos $W^{(l)} = \left[w_{i,j}^{(l)}\right]_{i,j=1}^{n^{(l-1)},n^{(l)}}$, de *bias* $\boldsymbol{b}^{(l)} = \left(b_i^{(l)}\right)_{i=1}^{n^{(l)}}$ e de função de ativação $\boldsymbol{f}^{(l)} : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ na l-ésima camada da rede. O número de neurônios (unidades de processamento) em cada camada é denotado por $n^{(l)}$, $l = 1, 2, \ldots, n_h + 1$, onde n_h é o número de camadas escondidas da rede. A propagação no perceptron multicamada é calculado de forma iterativa pelos acoplamentos

$$\boldsymbol{a}^{(l)} = f^{(l)}(W^{(l)}\boldsymbol{a}^{(l-1)} + \boldsymbol{b}^{(l)}), \tag{3.24}$$

onde $\boldsymbol{a}^{(0)} = x$ e $\tilde{\Psi}(x) = \boldsymbol{a}^{(n_l)}$.

Com uma arquitetura predefinida (quantidade de camadas n_l, quantidade de unidades $n^{(l)}$ por camada e as funções de ativação), o treinamento da RNA se dá pela resolução do seguinte problema de otimização

$$\min_{\{(W^{(l)}, \boldsymbol{b}^{(l)})\}_{t=1}^{n_l}} \frac{1}{n_s} \sum_{m=1}^{n_s} \left(\tilde{\Psi}_m - \Psi_m\right)^2 \tag{3.25}$$

para um conjunto de treinamento dado $\{x_m, \Psi(x_m)\}_{m=1}^{n_s}$, onde n_s é o número de amostras. Ou seja, a RNA é treinada para minimizar a função erro médio quadrático entre os valores esperados $\Psi_m = \Psi(x_m)$ e os estimados $\tilde{\Psi}_m$ (computados nas iteração de fonte anterior).

A cada iteração de fonte, treina-se uma RNA com base nos valores discretos conhecidos de $\Psi^{(j-1)}$. Com isso, estimativas

$$\tilde{\Psi}^{(j-1)}(x) = \mathcal{N}(x) \tag{3.26}$$

são conhecidas em qualquer ponto do domínio. O cálculo de $\Psi^{(j)}$ é feito apenas nos nodos computacionais, assim como o das novas intensidades $I_i^{(j)}$ por (3.22) e da quadratura

$$\Psi^{(j)}(x_k) = \frac{1}{2}\sum_{i=1}^{N} \omega_i I_i^{(j)}(x_k). \tag{3.27}$$

O processo segue, então, para a próxima iteração de fonte, com um retreinamento da rede com base no novo $\Psi^{(j)}$ conhecido nos nodos computacionais.

3.2.1 Algoritmo ANN-MoC

O algoritmo ANN-MoC segue as seguintes etapas:

i. Configure a estrutura da rede neural artificial $\mathcal{N}(x)$ atribuindo pesos e um *biases* iniciais.

ii. Defina o valor de n_s e o conjunto inicial de amostras de pontos de colocação $\{x_m\}_{m=1}^{n_s}$.

iii. Defina a aproximação inicial $\Psi^{(0)}$ nos nodos computacionais.

iv. Treine a $\mathcal{N}$ utilizando o conjunto de dados

$$\{x_m, \Psi^{(0)}(x_m)\}_{m=1}^{n_s}. \tag{3.28}$$

v. Estabeleça a quadratura $\{\mu_i, w_i\}_{i=1}^{N}$.

vi. Para $j = 1, ..., L$:

vi.a) Para $i = 1, ..., N$, para $m = 1, ..., n_s$:

* Se $\mu_i > 0$, então $s = (x_m - a)/\mu_i$

$$I_i^{(j)}(x_m) = I_a e^{-\sigma_t s} + e^{-\sigma_t s}\int_0^s \left[\mathcal{N}(s') + q(s', \mu_i)\right] e^{\sigma_t s'} ds' \tag{3.29}$$

* Se $\mu_i < 0$, então $s = (x_m - b)/\mu_i$

$$I_i^{(j)}(x_m) = I_b e^{-\sigma_t s} + e^{-\sigma_t s}\int_0^s \left[\mathcal{N}(s') + q(s', \mu_i)\right] e^{-\sigma_t s'} ds' \tag{3.30}$$

vi.b) Compute $\Psi^{(j)} = \frac{1}{2}\sum w_i I_i^{(j)}$ nos nodos computacionais.

vi.c) Treine novamente $\mathcal{N}$ com o novo conjunto de treinamento

$$\{x_m, \Psi^{(j)}(x_m)\}_{m=1}^{n_s}. \tag{3.31}$$

vi.d) Verifique um critério de parada fornecido.

vi.e) Reinicie o conjunto de pontos randômicos $\{x_m\}_{m=1}^{n_s} \subset \mathcal{D}$.

Observamos que não há necessidade de que o treinamento no passo vi.c) seja muito preciso, dada a própria imprecisão da aproximação $\Psi^{(j)}$. O custo de treinamento é relativamente alto, e o impacto de uma menor precisão acaba por implicar em uma maior quantidade de iterações de fonte até a convergência para uma tolerância desejada. Em geral, uma época de treinamento com o esquema *leave-one-out* é suficiente para o método ANN-MoC seja convergente.

3.2.2 Detalhes de Implementação

Nossa implementação computacional do método ANN-MoC foi realizada em linguagem de programação Python. Os modelos computacionais de RNAs foram construídos com a ajuda da biblioteca de aprendizagem de máquina PyTorch. No treinamento, o algoritmo de otimização Adam utilizado foi o disponível nesta mesma biblioteca.

Outros aspectos da implementação foram realizados com o suporte de arranjos da biblioteca NumPy. As integrais em (3.29) e (3.30) foram computadas com o método QUADPACK disponível na SciPy. A tolerância de erro empregada foi de 1.49×10^{-8} (relativo e absoluto).

O algoritmo ANN-MoC, foi implementado em uma versão paralela com paradigma MPI (*message passing interface*), disponível no pacote *MPI for Python*. A paralelização tem efeito apenas no laço das direções $i = 1, ..., N$, passo vi.a) do algoritmo, consulte a Subseção 3.2.1.

4 Resultados

Neste capítulo, exploramos o método ANN-MoC em aplicações selecionadas de forma a estudarmos seus vantagens e desvantagens. Nosso objetivo é realizar uma série de testes criteriosos para avaliar a eficácia deste método em comparação com outras abordagens conhecidas, em especial, com alternativas que usam de interpolação para as estimativas do fluxo escalar médio.

Primeiramente, apresentamos um rápido estudo sobre a aproximação de funções exponenciais por redes neurais artificiais (ANNs). A escolha se deve pelo fato de que o transporte de partículas neutras comumente apresenta um decaimento exponencial das intensidades de partículas. Ainda, o estudo serve de baliza para verificar o método de escolha da arquitetura das ANNs que serão empregadas nos problemas de transporte selecionados. Em seguida, aplicamos o método ANN-MoC a um problema com solução manufaturada idêntica à aproximada no estudo preliminar. Outro problema com solução manufatura é, então, estudado. Neste, assume-se uma solução dependente das direções. Por fim, consideramos um terceiro estudo de caso, em que a solução não tem decaimento exponencial.

4.1 Estudo de Aproximação de Função Exponencial por RNA

Neste estudo de caso, buscamos estudar a aproximação da função

$$\hat{\Psi}(x) = e^{-\alpha\sigma_t x} \tag{4.1}$$

por RNAs do tipo perceptron multicamadas. Assumimos $\alpha = 1$ e $\sigma_t = 1$ no domínio $x \in \mathcal{D} = [0, 1]$.

O objetivo, aqui, é de validar nossa implementação computacional da MLP e do algoritmo de treinamento. A escolha da função (4.1) está relacionada ao problema de transporte que será investigado na próxima seção, na qual (4.1) é a solução manufaturada do problema. Com isso, esperamos que os resultados obtidos aqui sirvam como balizadores para essa aplicação. Ainda, observamos que decaimentos exponenciais também são da natureza do fenômeno de transporte de partículas neutras.

Em todos os testes, trabalhamos com MLPs de arquitetura $1 - n_n \times n_h - 1$ (uma entrada, n_h camadas escondidas cada uma com n_n neurônios, uma saída)

$$\tilde{\Psi}(x) = \mathcal{N}(x), \tag{4.2}$$

onde $\tilde{\Psi}(x)$ é a estimativa do valor esperado $\hat{\Psi}(x)$. As funções de ativação são fixadas e o treinamento consiste em resolver o problema de minimizar a função erro (erro médio

quadrático)

$$\varepsilon = \frac{1}{n_s} \sum_{s=1}^{n_s} \left| \tilde{\Psi}(x_s) - \hat{\Psi}(x_s) \right|^2, \tag{4.3}$$

no espaço dos parâmetros da rede (pesos e *biases*). Em todos os testes, o otimizador escolhido foi o método Adam com taxa de aprendizagem $l_r = 0.001$ e momentos $\beta_1 = 0.9$ e $\beta_2 = 0.999$. Como critério de parada, fixamos $\mathcal{L} < 10^{-6}$ e número máximo de épocas 50000.

Começamos os testes avaliando diferentes arquiteturas (número de camadas escondidas n_h e número de neurônios por camada n_n) com função tangente hiperbólica $y = \tanh(v)$ e função identidade $y = \mathrm{id}(v)$ como funções de ativação nas camadas escondidas e na camada de saída, respectivamente. Na Tabela 1, temos o número médio de épocas $\bar{n}_e$ e o tempo computacional médio $\bar{t}_c$ (s) demandado para o treinamento de redes com diferentes arquiteturas. Devido à estocasticidade do algoritmo de treinamento, cada teste foi realizado três vezes e a média dos resultados são apresentados. A análise detalhada dos resultados desta tabela revela padrões significativos no desempenho das diferentes arquiteturas de redes neurais conforme variamos o tamanho do conjunto de dados. O objetivo é identificar configurações que apresentem uma rápida convergência, indicada pelo número reduzido de épocas necessárias para atingir a condição de parada e o menor custo computacional. O custo computacional foi medido pelo tempo de execução. Todos os testes foram feitos em um mesmo computador Intel Core i7-3612QM com memória 5.7GB DDR3.

Da Tabela 1, observamos que redes com arquiteturas pequenas necessitam de muitas épocas de treinamento para alcançar o critério de parada. Ou seja, embora essas tenham um baixo custo computacional de propagação, acabam demandando um alto custo para o treinamento. Para $n_s = 10$, o melhor resultado foi obtido com $n_n = 200$ e $n_h = 4$ em que o treinamento ocorreu em um número médio de épocas $\bar{n}_e < 1000$ e em um tempo computacional médio $\bar{t}_e = 5.25$ segundos. Nesta mesma tabela assim como nas Tabelas 2 e 3, precisamente nas posições 100×3, 200×3 e 400×4 observa-se um padrão nas regiões com os menores valores quase na mesma posição $n_n \times n_h$, ou seja, a tendência é que a necessidade de n_n com n_s crescente o que é intuitivo.

O aumento no número de amostras $n_s = 20, 40$ melhorou a estabilidade dos resultados, no sentido de que redes com $n_h = 3, 4$ e $n_n = 100, 200$ fornecem resultados similares. Neste sentido, o maior número de amostra melhora a previsibilidade do treinamento da ANN.

Em resumo, as configurações com 100 neurônios parecem desempenhar um papel crucial na eficiência do treinamento, enquanto a adição de camadas adicionais em algumas configurações contribui para uma convergência mais eficiente. Estes resultados sugerem que, para o problema em questão, arquiteturas com 100 neurônios e 3 ou 4 camadas podem representar escolhas eficazes, proporcionando uma convergência rápida durante o

Tabela 1 – Testes de treinamento de redes para o estudo de aproximação de função. Arquiteturas de rede $1 - n_n \times n_h - 1$, camadas escondidas com função de ativação $y = \tanh(v)$ e saída com $y = \mathrm{id}(v)$. Os valores tabelados são $\bar{n}_e \backslash \bar{t}_c$, número médio de épocas e tempo de execução médio para atingir o critério de parada $\varepsilon < 10^{-6}$.

$n_n \backslash n_h$	1	2	3	4	5
			$n_s = 10$		
5	25231\39.97	10156\21.93	7969\21.47	8745\28.90	3795\13.93
10	8329\12.56	6523\13.84	8036\21.50	9183\30.55	6144\23.02
20	14508\22.98	4663\10.48	3887\11.44	4746\16.67	2850\11.88
40	7031\11.62	3067\8.09	3463\11.48	2363\9.19	2673\11.35
50	7674\14.37	4111\10.76	2005\6.66	2207\8.78	2483\11.92
100	10061\21.78	3748\10.49	1491\5.31	4883\23.72	5350\29.53
200	8801\17.48	3164\10.23	1123\5.53	845\5.25	3426\24.67
400	10594\19.86	2419\10.83	3784\27.73	3227\31.26	1909\23.58
$n_n \backslash n_h$	1	2	3	4	5
			$n_s = 20$		
5	14821\29.11	7564\19.46	10175\28.50	6039\20.22	7353\29.14
10	12959\21.31	6758\16.11	5831\17.99	7925\29.88	4021\16.72
20	8773\15.49	5828\15.65	3637\9.73	3702\12.02	3133\11.89
40	9491\15.02	4186\8.92	2376\6.67	2030\6.99	2533\10.12
50	11245\17.60	3263\6.89	2493\6.73	1957\6.71	2743\10.99
100	7560\11.88	3045\7.07	2453\8.53	2173\8.22	1581\7.09
200	6964\11.46	2561\7.04	1462\5.72	2791\14.24	2590\15.73
400	6999\11.97	2306\9.29	1589\10.32	827\7.10	2038\23.58
$n_n \backslash n_h$	1	2	3	4	5
			$n_s = 40$		
5	18882\28.68	7781\15.72	11151\28.82	9309\29.32	8306\29.86
10	11981\18.36	8867\18.73	7714\19.99	3644\11.23	3862\14.17
20	7594\11.49	6419\14.79	4466\14.39	4265\16.98	2875\11.48
40	7842\13.33	2205\5.42	2216\6.92	2287\8.86	1846\8.33
50	6120\10.78	2792\8.20	3757\14.16	2289\10.72	2382\12.99
100	8978\19.45	3177\10.08	1172\4.81	1231\5.21	5076\26.87
200	5744\10.14	1948\5.20	1671\7.67	1237\7.29	1716\12.79
400	6573\13.13	2510\12.44	1056\7.83	2165\20.54	1905\21.38

treinamento. No entanto, uma análise mais aprofundada, considerando a estabilidade do treinamento e o desempenho em conjuntos de dados de validação, é recomendada para uma seleção final da arquitetura mais adequada. Feito isso, é importante estudar o desempenho de outros tipos de funções de ativação, lembrando que todos os testes realizados até aqui possuem aplicação de função tangente hiperbólica $y = \tanh(v)$ como função de ativação com critério de parada $\varepsilon < 10^{-6}$. Por isso fazemos a questão de avaliar também as funções de ativação ReLU $y = \text{relu}(v)$, sigmoide $y = \text{sigmoid}(v)$ e softplus $y = \text{softplus}(v)$.

Na Tabela 2, apresentamos resultados para a comparação do treinamento de redes com diferentes arquiteturas e escolhas da função de ativação em camadas escondidas. Em todos os casos, a função identidade $y = \text{id}(v)$ é aplicada na camada de saída da rede. A notação $\{n_n \times n_h\}^{n_s}$ é usada para indicar uma rede de arquitetura $1 - n_n \times n_h - 1$ com n_s amostras em cada época de treinamento. Novamente, os resultados apresentados são o número de épocas médio e o tempo de execução médio de três tentativas de treinamento com critério de parada $\varepsilon < 10^{-6}$.

Tabela 2 – Treinamento da rede usando diferentes funções da ativação na camada escondida. Valores tabelados são $\bar{n}\backslash\bar{t}_c$ número médio de épocas e tempo de execução médio. Critério de parada para o treinamento $\varepsilon < 10^{-6}$.

$\{n_n \times n_h\}^{n_s}$	$\bar{n}_e/\bar{t}_c$ tanh	relu	sigmoid	softplus
$\{40 \times 2\}^{10}$	3067\8.09	7683\17.46	9522\24.71	4598\14.13
$\{50 \times 3\}^{10}$	2005\6.66	2274\7.00	12591\48.45	2518\10.74
$\{50 \times 4\}^{10}$	2207\8.78	1011\3.79	10358\47.81	1269\6.67
$\{100 \times 3\}^{10}$	1491\5.31	948\3.08	14961\64.82	1915\9.95
$\{200 \times 3\}^{10}$	1123\5.53	252\1.04	8728\50.22	1266\10.10
$\{200 \times 4\}^{10}$	845\5.25	294\1.53	7804\47.32	1807\17.95
$\{50 \times 4\}^{20}$	1957\6.71	1903\7.18	5434\27.91	1668\8.84
$\{100 \times 3\}^{20}$	2453\8.53	519\1.66	11757\52.52	1565\8.83
$\{100 \times 4\}^{20}$	2173\8.22	567\2.33	4067\22.74	2529\17.95
$\{100 \times 5\}^{20}$	1581\7.09	381\1.98	2919\19.34	3076\27.26
$\{200 \times 3\}^{20}$	1462\5.72	343\1.49	10042\60.60	1761\14.43
$\{400 \times 4\}^{20}$	827\7.10	159\1.37	7170\95.11	9940\173.13
$\{40 \times 2\}^{40}$	2205\5.42	6839\16.85	7727\21.64	2931\8.84
$\{40 \times 3\}^{40}$	2216\6.92	2968\9.03	10327\38.45	1993\6.95
$\{100 \times 3\}^{40}$	1172\4.81	1112\3.89	8909\41.26	1591\8.27
$\{100 \times 4\}^{40}$	1231\5.21	726\3.07	3293\19.25	2382\15.92
$\{200 \times 3\}^{40}$	1671\7.67	239\0.98	7832\50.29	1175\9.55
$\{200 \times 4\}^{40}$	1237\7.29	586\3.41	2971\24.31	678\6.99

Ao analisarmos os resultados apresentados na Tabela 2 que compara o desempenho de diferentes funções de ativação em redes neurais, podemos extrair informações para determinar qual função é mais eficaz para o treinamento da rede em nosso cenário. A métrica utilizada para avaliação é o número de épocas necessárias para atender à condição de parada, juntamente com o tempo computacional correspondente, expresso como $\bar{n}_e\backslash\bar{t}_c$.

A função de ativação tangente hiperbólica $y = \tanh(v)$ demonstra consistente bom desempenho em diversas configurações, o que pode ser atribuído às suas propriedades favoráveis durante o treinamento de redes neurais. A $y = \tanh(v)$ é uma função centrada em zero, produzindo saídas no intervalo $[-1, 1]$, e é caracterizada pela sua suavidade. Esta suavidade contribui para a estabilidade do treinamento, evitando extremos nos gradientes e reduzindo a propensão a problemas como gradientes que desaparecem ou explodem.

A função ReLU $y = \text{relu}(v)$ destaca-se por seu desempenho competitivo, especialmente em arquiteturas maiores (consulte o caso $\{40 \times 4\}^{20}$). ReLU é computacionalmente eficiente, uma vez que ativa apenas os neurônios com saída positiva. Isso contribui para um treinamento mais rápido em algumas configurações, resultando em menor número de épocas necessárias para atingir a convergência. Entretanto, é importante notar que a eficiência do ReLU pode variar muito a depender da arquitetura da rede. Em algumas situações, a ativação abrupta de ReLU pode levar a problemas como neurônios inativos, onde esses neurônios permanecem sem ativação e não contribuem para o aprendizado. Apesar disso, ReLU continua sendo uma escolha popular devido à sua simplicidade e eficácia.

Em contrapartida, a função de ativação sigmoide $y = \text{sigmoid}(v)$ mostrou-se menos eficiente em muitos casos, exigindo um número significativo de épocas para atingir a convergência. Embora da mesma classe da função $y = \tanh(v)$, observa-se que a positividade da função sigmoide restringiu o desempenho do treinamento. Mesmo que a função objetivo seja também não nula, um balanço entre ativações negativas e positivas parece ser importante para o aprendizado de uma função exponencial. A ReLU também é não negativa, mas desativa neurônios com pré-ativações negativas.

Por outro lado, redes com a função softplus $y = \text{softplus}(v)$ (também uma função não-negativa) apresenta um desempenho similar a redes com a função $y = \tanh(v)$, embora com um custo computacional ligeiramente maior. Softplus é uma função suave e não saturante, o que significa que evita problemas associados à saturação de gradientes.

Do ponto de vista gráfico, todas as redes treinadas apresentam resultados satisfatórios. Por exemplo, consulte a Figura 11, em que mostramos uma comparação, com diferentes funções de ativação, entre resultados esperados $y = \hat{\Psi}(x)$ e estimados $y = \tilde{\Psi}(x)$ para redes com arquitetura $1 - 200 \times 3 - 1$ e treinamento com $n_s = 40$. Isso reforça que a escolha pode ser baseada nos indicadores de número de épocas e tempo computacional.

Voltando a uma análise abrangente dos resultados mostrados na Tabela 2, observamos que as funções $y = \text{relu}(v)$ e $y = \tanh(v)$ são as que mostraram melhores resultados, porém, a função de ativação ReLU destaca-se como uma escolha geralmente mais eficiente, apresentando uma boa relação entre o número de épocas e o tempo computacional em diversas arquiteturas. Especialmente, destacamos a arquitetura $\{200 \times 3\}^{40}$, onde obtemos uma métrica de 239 épocas em relação ao tempo computacional de 0.98 segundo. A

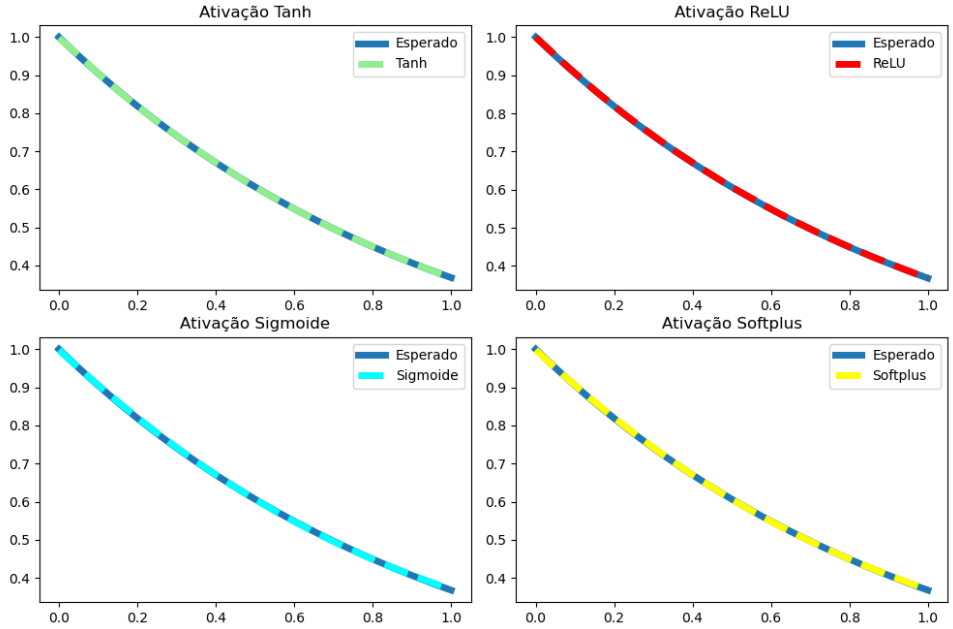


Figura 11 – Comparação entre resultados esperados $y = \hat{\Psi}(x)$ e estimados $y = \tilde{\Psi}(x)$ para redes com arquitetura $1-200\times3-1$ e treinamento com $n_s = 40$. Comparações com diferentes funções de ativação.

eficiência da ReLu tem relação com os seguintes motivos:

a) ReLU é conhecida por sua eficiência computacional, uma vez que ativa apenas os neurônios com saída positiva. Isso resulta em cálculos mais rápidos durante o treinamento da rede. No contexto da arquitetura $\{200 \times 3\}^{40}$, a métrica de 239 épocas em apenas 0.98 segundo sugere uma convergência rápida em comparação com outras funções de ativação.

b) A ativação linear para valores positivos em ReLU ajuda a mitigar o problema de gradientes muito altos ou saturados. Isso é crucial, especialmente em arquiteturas mais profundas, onde o controle sobre os gradientes é essencial para o treinamento estável.

c) A arquitetura $\{200 \times 3\}^{40}$ pode beneficiar-se da não-centralização de ReLU, uma vez que a ativação não é limitada a um intervalo específico como ocorre com $y = \tanh(v)$. Isso pode ser vantajoso para representar uma variedade mais ampla de padrões nos dados.

d) ReLU tende a se sair bem em conjuntos de dados esparsos, o que pode ser benéfico dependendo das características do problema em questão.

e) A ReLU demanda um baixo custo computacional para seu cálculo.

No entanto, é importante mencionar que a escolha da ReLU não é uma regra geral e pode depender das características específicas do problema, da natureza dos dados e

da arquitetura da rede. Experimentação e validação em conjuntos de dados de teste são essenciais para confirmar a escolha da função de ativação mais adequada ao contexto específico.

Ainda nesta etapa da análise, vamos investigar o impacto da função de ativação na camada de saída durante o treinamento de redes neurais. Nos experimentos anteriores, destacou-se o desempenho eficaz da função ReLU nas camadas ocultas, proporcionando convergência rápida e eficiente. Agora, direcionaremos nossa atenção para a camada de saída, onde anteriormente não aplicamos nenhuma função de ativação. Optaremos por utilizar a função Softplus na camada de saída, uma escolha motivada por sua similaridade com a ReLU e por ser uma função não-negativa, assim como o fluxo escalar de partículas.

A Tabela 3, apresenta os resultados de desempenho do treinamento de MLPs de arquitetura $1 - n_n \times n_h - 1$, com função $y = \text{relu}(v)$ e função $y = \text{softplus}(v)$ como funções de ativação em camadas ocultas e camada de saída, respectivamente. Observamos que redes pequenas e pouco profundas (poucas camadas escondidas) não tiveram êxito em aprender a função objetivo $\hat{\Psi}$. Em contrapartida, redes maiores e mais profundas tiveram um ótimo desempenho. Especialmente, a rede 200×3 com $n_s = 40$ obteve um desempenho notável. Consulte, também, a Figura 12 para uma comparação gráfica entre os valores esperados $\hat{\Psi}(x)$ e os valor estimados $\tilde{\Psi}(x)$ com esta arquitetura de rede.

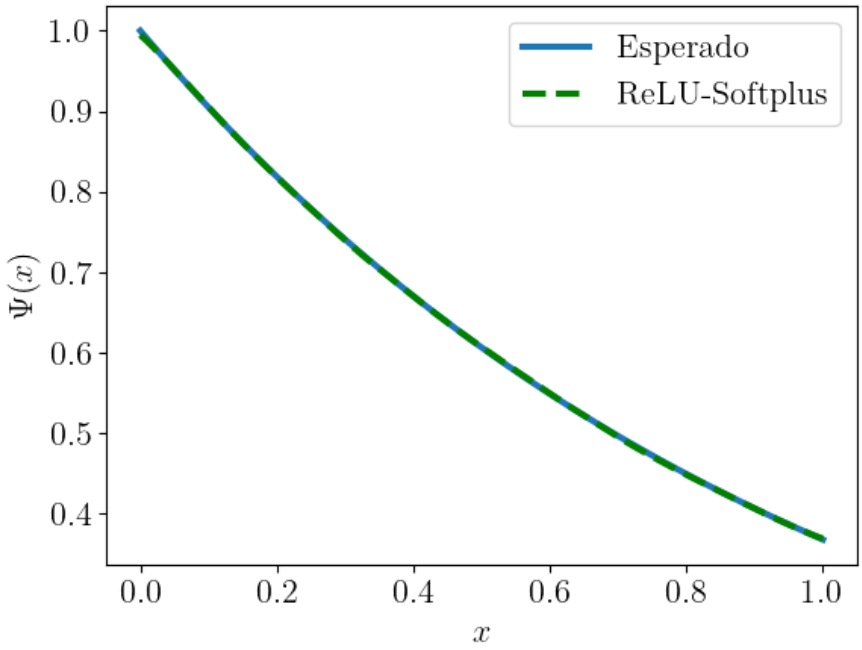


Figura 12 – Comparação entre os valores esperados $\hat{\Psi}(x)$ e os valores estimados $\tilde{\Psi}(x)$ com uma rede $\{1 - 200 \times 3 - 1\}^{40}$, função $y = \text{relu}(v)$ e função $y = \text{softplus}(v)$ como funções de ativação nas camadas ocultas e de saída, respectivamente.

Em resumo dos resultados obtidos nesta seção, observamos que:

Tabela 3 – Testes de treinamento de redes com função de ativação Softplus para camada de saída e função ReLU nas camadas ocultas. Valores tabelados são $\bar{n}\backslash\bar{t}_c$ número médio de épocas e tempo de execução médio. Critério de parada para o treinamento $\varepsilon < 10^{-6}$.

$n_n \backslash n_h$	1	2	3	4	5
			$n_s = 10$		
5	-	-	-	-	-
10	-	23793\55.08	4482\11.90	3388\11.06	2755\10.14
20	4890\8.18	2105\5.07	1740\5.24	870\3.1	1394\5.74
40	4664\7.93	1468\3.65	770\2.42	544\1.99	420\1.76
50	2456\4.03	652\1.57	434\1.28	454\1.71	434\1.91
100	706\1.16	393\1.00	190\0.65	166\0.68	134\0.65
200	789\1.41	202\0.56	119\0.50	105\0.55	141\0.91
400	241\0.45	107\0.53	110\0.71	137\1.39	129\1.89
$n_n \backslash n_h$	1	2	3	4	5
			$n_s = 20$		
5	-	-	-	-	-
10	-	-	17121\47.03	18945\65.66	7277\26.97
20	10464\17.08	4095\9.72	1561\4.44	534\2.19	515\1.96
40	2355\3.48	1250\2.82	521\1.47	541\1.94	289\1.12
50	2943\5.10	735\2.15	401\1.32	524\2.20	513\2.55
100	1068\1.90	280\0.83	222\0.86	142\0.65	124\0.73
200	629\1.32	150\0.50	103\0.55	110\0.71	111\0.85
400	191\0.41	125\0.65	83\0.69	87\0.98	116\1.62
$n_n \backslash n_h$	1	2	3	4	5
			$n_s = 40$		
5	-	-	-	-	-
10	-	36711\93.24	3581\10.31	5229\18.64	7007\29.31
20	6091\10.34	4265\10.74	1130\3.45	951\3.48	915\4.18
40	2449\4.13	814\2.00	452\1.38	284\1.10	277\1.27
50	1053\1.72	649\1.61	390\1.32	381\1.55	382\1.69
100	700\1.15	253\0.68	131\0.44	116\0.51	121\0.72
200	266\0.61	106\0.40	94\0.40	124\0.72	101\0.77
400	258\0.49	85\0.40	85\0.60	96\1.00	98\1.71

- Redes com função de ativação $y = \tanh(v)$ nas camadas ocultas e $y = \mathrm{id}(v)$ na saída apresentaram resultados estáveis e consistentes para todas as arquiteturas testadas.

- Redes com função de ativação $y = \mathrm{relu}(v)$ nas camadas ocultas e $y = \mathrm{id}(v)$ na saída apresentaram um desempenho melhor para redes com pelo menos $n_n = 100, 200$ e com profundidade $n_h = 3, 4$. Redes pequenas e pouco profundas não tiveram bons resultados.

- Redes com função de ativação $y = \text{relu}(v)$ nas camadas ocultas e $y = \text{softplus}(v)$ na saída apresentaram o melhor desempenho para redes com pelo menos $n_n = 100, 200$ e com profundidade $n_h = 3, 4$. Redes pequenas e pouco profundas não lograram apreender a função objetivo.

4.2 Problema de Transporte 1: Solução Manufaturada Exponencial

Nesta seção, apresentamos nossa primeira aplicação do método ANN-MoC em um problema de transporte de partículas modelado por (1.1). Seguindo uma abordagem de solução manufatura, assumimos que o fluxo angular de partículas é isotrópico dado pela seguinte função exponencial

$$\hat{I}(x, \mu) = e^{-\alpha\sigma_t x}. \tag{4.4}$$

com $\alpha = 5$ e $\sigma_t = 1$. Substituindo $\hat{I}$ por I em (1.1), obtemos a fonte de partículas

$$\mu \cdot \frac{\partial}{\partial x}(e^{-\alpha\sigma_t x}) + \sigma_t e^{-\alpha\sigma_t x} = \sigma_s \Psi(x) + q(x, \mu) \tag{4.5}$$

$$-\mu \cdot \alpha\sigma_t e^{-\alpha\sigma_t x} + \sigma_t e^{-\alpha\sigma_t x} = \sigma_s \Psi(x) + q(x, \mu) \tag{4.6}$$

$$(\sigma_t - \mu\alpha\sigma_t) e^{-\alpha\sigma_t x} = \sigma_s \Psi(x) + q(x, \mu) \tag{4.7}$$

$$q(x, \mu) = (\sigma_t - \mu\alpha\sigma_t) e^{-\alpha\sigma_t x} - \sigma_s \Psi(x). \tag{4.8}$$

O fluxo médio esperado também é imediatamente calculado

$$\hat{\Psi}(x) = \frac{1}{2} \int_{-1}^{1} (e^{-\alpha\sigma_t x})\, d\mu \tag{4.9}$$

$$= \frac{1}{2} e^{-\alpha\sigma_t x} \int_{-1}^{1} d\mu \tag{4.10}$$

$$= \frac{1}{2} e^{-\alpha\sigma_t x} (\mu) \Big|_{-1}^{1} \tag{4.11}$$

$$= \frac{1}{2} e^{-\alpha\sigma_t x} (1 - (-1)) \tag{4.12}$$

$$= e^{-\alpha\sigma_t x}. \tag{4.13}$$

Portanto, substituindo (4.13) em (4.8), obtemos

$$q(x, \mu) = (\sigma_t - \mu\alpha\sigma_t) e^{-\alpha\sigma_t x} - \frac{\sigma_s}{2} (2e^{-\alpha\sigma_t x}) \tag{4.14}$$

$$= \left[\sigma_t (1 - \mu\alpha) - \sigma_s\right] e^{-\alpha\sigma_t x}. \tag{4.15}$$

No que segue, apresentamos resultados assumindo $\sigma_s = 0.25$.

O método ANN-MoC requer a escolha *a priori* da arquitetura da RNA. Durante os testes numéricos, usamos MLPs com diferentes funções de ativação, a começar com tangente

hiperbólica como função de ativação na camada escondida. Arquiteturas $1-n_n \times n_h-1$ foram empregadas, isto é, uma entrada, n_h camadas escondidas, cada uma com n_n neurônios, e uma saída. Observamos que, em geral, a primeira iteração de fonte demanda o treinamento mais custoso da rede neural. Em iterações de fonte subsequentes, o treinamento da rede pode ser inicializado com a técnica *warm initialization*, em que seus parâmetros são mantidos do treinamento anterior. Como critério de parada para as iterações de fonte, usamos $\|\tilde{\Psi}^{(l)} - \tilde{\Psi}^{(l-1)}\|_2 < 10^{-6}$ e número máximo de iterações 100.

Tabela 4 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8$ e $N = 2$.

$n_n \times n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 8,\ N = 2$						
5×1	6.3447e-01	3.9860e-01	2.3604e-01	-6.3155e-03	77	10.74
5×2	6.3828e-01	4.3819e-01	2.9495e-01	3.6063e-03	66	9.22
5×3	6.9086e-01	4.4564e-01	2.6717e-01	-6.7191e-02	39	6.46
5×4	7.7138e-01	4.0115e-01	1.9609e-01	-4.8786e-02	54	10.11
10×1	7.2785e-01	4.2762e-01	2.3007e-01	-9.7207e-02	58	8.09
10×2	6.9978e-01	4.1770e-01	2.3997e-01	-6.6594e-02	45	6.18
10×3	6.5351e-01	4.0685e-01	2.3809e-01	-7.8503e-02	28	4.69
10×4	6.7853e-01	4.0392e-01	2.2001e-01	-8.0790e-02	24	4.53
25×1	6.8904e-01	4.2258e-01	2.4448e-01	-8.9170e-02	22	3.21
25×2	6.9478e-01	4.1730e-01	2.2962e-01	-1.1570e-01	26	3.66
25×3	6.6536e-01	3.7842e-01	1.9196e-01	-1.3235e-01	19	3.15
25×4	8.7982e-01	5.8977e-01	3.8763e-01	3.2426e-02	20	3.83
50×1	6.8548e-01	4.3917e-01	2.6275e-01	-8.9324e-02	24	3.39
50×2	7.1202e-01	4.3024e-01	2.5272e-01	-5.9832e-02	22	3.20
50×3	8.3406e-01	4.4661e-01	2.2211e-01	-1.1706e-01	24	4.10
50×4	6.8875e-01	3.9792e-01	2.0686e-01	-1.1639e-01	13	2.57
100×3	8.1749e-01	5.1127e-01	3.1744e-01	-2.8875e-04	8	1.69
100×4	9.4908e-01	2.1698e-01	7.4824e-02	2.2822e-02	37	7.27
200×3	9.3124e-01	2.8125e-01	7.6319e-02	1.1940e-03	30	5.54
200×4	9.8449e-01	2.2011e-01	8.5351e-02	2.3553e-02	41	9.22
Esperado	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	-x-	-x-

A Tabela 4 apresenta os resultados do fluxo escalar médio estimado $\tilde{\Psi}(x)$ e esperado $\hat{\Psi}(x)$ para pontos selecionados $x = 0.0, 0.3, 0.5, 0.9$, o número de iterações de fonte L e o tempo computacional demandados. Aqui, usou-se apneas $n_s = 8$ amostras de pontos de colocação em cada iteração. Tendo em vista que, neste estudo de caso, I não dependente das direções, a quadratura guassiana com $N = 2$ é suficiente. Observamos que redes MLPs pequenas ou pouco profundas produziram resultados imprecisos, ao mesmo tempo em que

demandaram um alto número de iteração de fonte L até a convergência.

Tabela 5 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 10$ e $N = 2, 8$.

$n_n \times n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 10,\ N = 2$						
25×2	6.7775e-01	4.3035e-01	2.4853e-01	-1.1224e-01	29	5.10
25×4	7.4211e-01	5.1112e-01	3.3597e-01	-2.1180e-02	13	3.24
50×3	9.4434e-01	2.0999e-01	6.4149e-02	9.8721e-03	66	13.82
50×4	9.9506e-01	2.1822e-01	7.8465e-02	7.7194e-03	67	15.99
100×1	6.7320e-01	4.2521e-01	2.6210e-01	-3.2646e-02	16	2.76
100×3	9.7467e-01	2.1372e-01	6.8226e-02	7.9722e-03	68	14.38
100×4	9.8985e-01	2.1790e-01	7.9623e-02	1.2435e-02	43	10.56
200×2	6.1764e-01	3.3387e-01	1.6838e-01	-9.0469e-02	15	2.62
200×4	9.9213e-01	2.1772e-01	8.2458e-02	1.2273e-02	48	13.70
$n_s = 10,\ N = 8$						
25×1	6.7693e-01	4.2074e-01	2.3893e-01	-1.2186e-01	33	19.53
25×3	8.0783e-01	5.5753e-01	3.7552e-01	1.7071e-02	14	10.29
50×2	6.8643e-01	4.6077e-01	2.9232e-01	-5.9049e-02	15	9.38
50×4	9.8929e-01	2.1645e-01	6.1800e-02	1.3473e-02	39	31.82
100×2	7.2936e-01	4.2734e-01	2.3156e-01	-1.2371e-01	15	9.45
100×3	9.9499e-01	2.2389e-01	8.1693e-02	1.1526e-02	82	58.56
100×4	9.9471e-01	2.2232e-01	8.3804e-02	9.3938e-03	71	58.67
200×1	8.0903e-01	5.0551e-01	3.1621e-01	-1.4107e-02	23	13.92
200×2	6.5287e-01	3.7166e-01	1.9598e-01	-1.0410e-01	18	11.27
200×3	9.4899e-01	2.1917e-01	6.5247e-02	1.1207e-02	43	36.79
200×4	9.9271e-01	2.2106e-01	8.5959e-02	1.8490e-02	46	43.88
exata	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	-x-	-x-

A Tabela 5 apresenta resultados análogos aos anteriores, mas agora com $n_s = 10$ amostras de pontos de colocação a cada iteração de fonte do método ANN-MoC. Observa-se um ganho substancial na precisão dos resultados obtidos. Porém, nota-se que em ambas as tabelas, especificamente em $x = 0.9$, temos valores negativos, o que pode ser devido a função $\tanh(v)$, utilizado nas camadas ocultas, mapeia valores de entrada para um intervalo entre -1 e 1. Isso pode causar um efeito de comprimir os valores, especialmente próximo das extremidades do intervalo de entrada. Lembrando que, a função $\mathrm{id}(v)$ (função identidade) na camada de saída não realiza nenhuma transformação não linear, o que implica que qualquer erro acumulado nas camadas anteriores será refletido diretamente na saída. Por outro lado, como esperado, o aumento no número de pares da quadratura para

$N = 8$ não forneceu alteração substancial na precisão dos resultados, ao mesmo tempo que implica em um custo computacional maior.

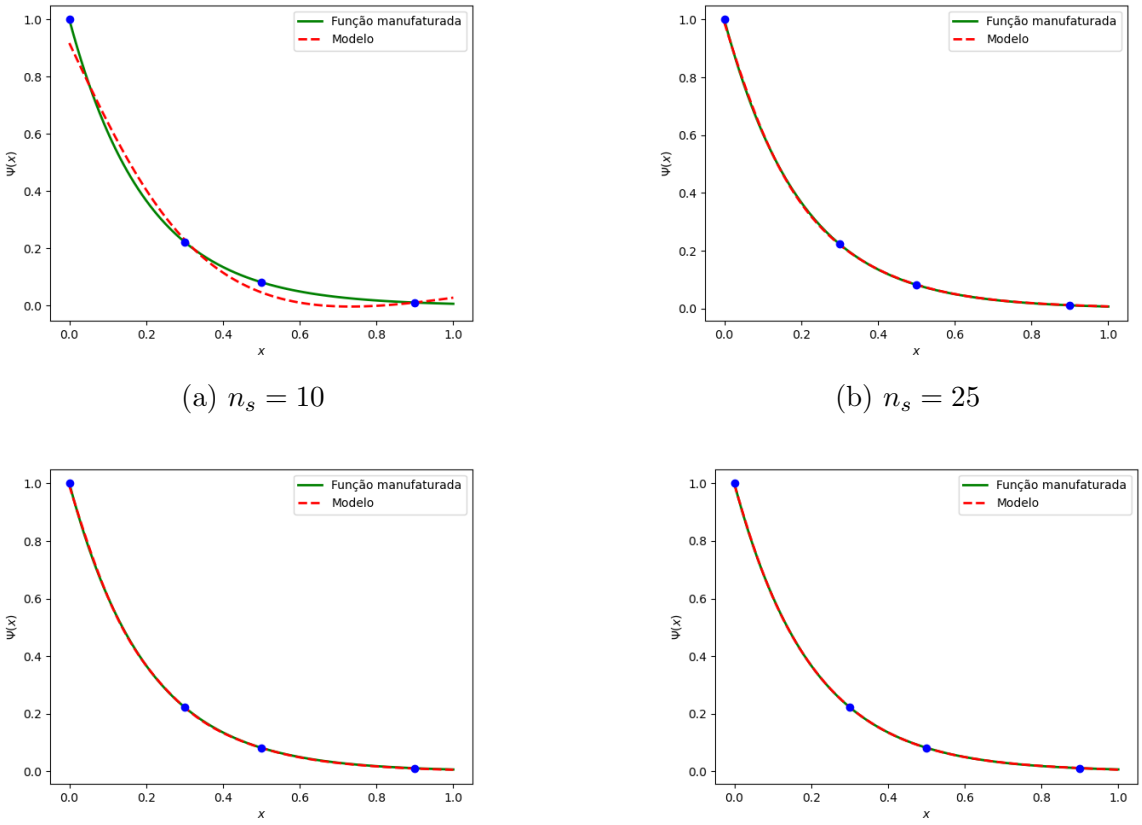
Na Tabela 6, apresentamos mais resultados comparativos para $n_s = 25, 50, 100$ e $N = 10, 20, 40$. Observamos que o aumento no número de amostras e de pares da quadratura acabam por demandar um custo computacional substancialmente mais alto, mas com pouco ganho na precisão dos resultados.

Tabela 6 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 25, 50, 100$ e $N = 10, 20, 40$.

$n_n \times n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 25,\ N = 10$						
25×2	6.4878e-01	3.8567e-01	2.1168e-01	-1.1561e-01	12	21.01
25×4	9.9016e-01	2.2090e-01	8.5235e-02	8.9458e-03	59	145.94
50×3	9.9032e-01	2.2362e-01	8.2647e-02	1.0453e-02	58	128.11
100×4	9.9674e-01	2.2561e-01	8.7615e-02	1.5852e-02	33	86.10
200×3	9.9178e-01	2.2252e-01	8.3980e-02	1.1606e-02	56	142.21
$n_s = 25,\ N = 20$						
100×3	9.9229e-01	2.2122e-01	8.6278e-02	1.1625e-02	49	220.84
100×4	9.9159e-01	2.1888e-01	8.1364e-02	9.4726e-03	28	136.63
200×2	9.0391e-01	2.8596e-01	7.6147e-02	3.2459e-02	37	130.00
200×4	9.9431e-01	2.2546e-01	8.8032e-02	1.3735e-02	25	132.11
$n_s = 50,\ N = 20$						
25×2	9.7835e-01	2.2066e-01	7.6354e-02	1.3154e-02	76	579.37
25×3	9.9579e-01	2.2097e-01	7.9481e-02	8.6654e-03	81	734.87
50×2	9.8059e-01	2.1440e-01	7.7437e-02	1.0426e-02	100	788.26
100×3	9.9499e-01	2.2197e-01	7.9753e-02	9.5238e-03	78	712.60
200×2	9.8881e-01	2.2177e-01	8.5283e-02	1.5268e-02	93	742.47
200×3	9.9301e-01	2.2193e-01	7.9724e-02	9.2171e-03	67	691.88
$n_s = 100,\ N = 40$						
50×4	9.9655e-01	2.2434e-01	8.3720e-02	1.2223e-02	44	1887.84
100×2	9.9024e-01	2.2191e-01	8.4209e-02	1.1208e-02	63	1993.58
100×3	9.9836e-01	2.2636e-01	8.5995e-02	1.5152e-02	26	1008.91
200×2	9.8972e-01	2.2241e-01	8.4786e-02	1.1679e-02	89	2928.38
200×3	9.9604e-01	2.2620e-01	8.3439e-02	1.3463e-02	23	865.51
exata	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	-x-	-x-

Dos resultados apresentados, concluímos que MLPs com arquitetura $1 - 200 \times 3 - 1$ e $N = 2$ são escolhas adequadas para o método ANN-MoC. A Figura 13 apresenta uma comparação gráfica entre soluções ANN-MoC $y = \tilde{\Psi}(x)$ (linha tracejada vermelha) e a

solução esperada $y = \hat{\Psi}(x)$ (linha contínua verde), $x \in \mathcal{D} = [0, 1]$. Observa-se que os resultados do modelo MLP obtido do método ANN-MoC apresentam boa precisão gráfica quando usado $n_s = 25$ ou maior.



(a) $n_s = 10$ (b) $n_s = 25$

(c) $n_s = 50$ (d) $n_s = 100$

Figura 13 – Problema de Transporte 1. Comparação da solução ANN-MoC $y = \tilde{\Psi}(x)$ com a solução esperada $y = \hat{\Psi}(x)$ para $x \in \mathcal{D} = [0, 1]$. Rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(x)$ nas camadas escondidas e camada de saída, $N = 2$.

Na sequência apresentados testes numéricos do método ANN-MoC empregando MLPs com funções de ativação $y = \mathrm{relu}(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e de saída, respectivamente. Lembramos que no estudo da aproximação da solução manufaturada realizado na Seção 4.1, indicam que esta seria uma escolha adequada (menor tempo computacional) para redes com arquiteturas profundas. Na Tabela 7, apresentados resultados comparativos para diferentes arquiteturas de rede em pontos de colocação selecionados. Em comparação com os resultados anteriores (função tangente hiperbólica nas camadas escondidas), observa-se que os testes com ReLU demandaram maior tempo computacional e forneceram soluções com precisões similares. O maior custo computacional se deve pela maior demanda no número de iterações de fonte. Isso, por sua vez, é explicado pelo fato da rede MLP com ReLU na inicialização $\tilde{\Psi}(x) \equiv 0$ conter parâmetros (pesos e *biases*) muito diferentes daqueles necessários para a aproximação de $\hat{\Psi}(x)$. Também nota-se valores

negativos no ponto $x = 0.9$ em algumas arquitetura, isto é, resultado muito longe do esperado. Mesmo com a função ReLU, a camada de saída pode produzir valores negativos devido à combinação linear dos pesos e biases aprendidos durante o treinamento. Refinar a malha e aumentar a complexidade da rede neural ajuda a melhorar a precisão das previsões.

Tabela 7 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8, 10$ e $N = 2, 8$.

$n_n \times n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 8,\ N = 2$						
25×2	8.2366e-01	3.4284e-01	1.3097e-01	1.3458e-02	55	24.04
25×4	9.3561e-01	2.0019e-01	8.6280e-02	-1.6099e-02	48	38.89
50×3	9.7034e-01	2.2029e-01	7.5892e-02	1.3000e-02	52	33.59
50×4	9.6780e-01	2.1774e-01	8.9740e-02	2.6927e-02	54	36.43
100×1	7.2154e-01	3.8227e-01	1.9920e-01	-8.2373e-02	12	6.23
100×3	9.6526e-01	2.3173e-01	6.4885e-02	-7.0206e-03	28	24.42
100×4	9.9065e-01	2.3093e-01	9.2129e-02	8.0500e-03	20	15.68
200×2	9.7169e-01	2.2485e-01	8.5420e-02	1.5378e-02	62	43.56
200×4	9.9914e-01	2.1978e-01	8.0874e-02	8.7146e-03	33	37.91
$n_s = 10,\ N = 8$						
25×1	5.9972e-01	4.2889e-01	2.8557e-01	1.1331e-02	15	17.81
25×3	9.5416e-01	2.2187e-01	8.2583e-02	1.1605e-02	64	136.50
50×2	9.0574e-01	2.5497e-01	8.2398e-02	1.4718e-02	67	145.43
50×4	9.8633e-01	2.1904e-01	7.7712e-02	7.9945e-03	37	171.12
100×2	7.7254e-01	3.0531e-01	8.7722e-02	2.0928e-03	26	64.50
100×3	1.0221e+00	2.4547e-01	8.2864e-02	-1.4372e-02	17	47.09
100×4	9.9330e-01	2.2031e-01	7.9965e-02	1.1380e-02	27	109.49
200×1	9.0943e-01	2.1577e-01	5.6592e-02	-8.1652e-03	28	69.79
200×2	9.3948e-01	2.4624e-01	7.1719e-02	8.7545e-03	36	95.46
200×3	9.7809e-01	2.1893e-01	8.0548e-02	1.2208e-02	27	100.29
200×4	9.9849e-01	2.3439e-01	8.3900e-02	1.0825e-02	24	98.33
exata	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	-x-	-x-

Mais alguns resultados com $n_s = 25, 50$ e 100 são apresentados na Tabela 8. Como anteriormente, o aumento no número de amostras de pontos de colocação não forneceu uma substancial melhora na precisão dos resultados.

Na Figura 14, temos comparações gráficas entre as soluções estimadas ANN-MoC $y = \tilde{\Psi}(x)$ e a solução esperada $y = \hat{\Psi}(x)$, $x \in \mathcal{D} = [0, 1]$. O modelo de rede MLP é $1 - 200 \times 3 - 1$ com funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e de saída, respectivamente. Resultados de testes numéricos com $n_s = 10, 25, 50$ e 100 são

Tabela 8 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 25, 50, 100$ e $N = 10, 20, 40$.

$n_n \times n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 25,\ N = 10$						
25×2	9.4530e-01	2.1960e-01	8.2877e-02	1.1564e-02	50	156.96
25×4	9.9545e-01	2.2220e-01	8.1578e-02	1.1748e-02	44	233.45
50×3	9.8104e-01	2.2222e-01	7.7915e-02	1.1826e-02	34	139.03
100×4	9.9991e-01	2.2355e-01	8.2050e-02	1.1259e-02	35	194.50
200×3	9.9489e-01	2.2288e-01	8.2866e-02	1.1630e-02	26	143.90
$n_s = 25,\ N = 20$						
100×3	9.8427e-01	2.2571e-01	8.4100e-02	1.1231e-02	27	259.12
100×4	9.9812e-01	2.2330e-01	8.2185e-02	1.1103e-02	31	347.59
200×2	9.7583e-01	2.2426e-01	8.0564e-02	1.1010e-02	40	287.05
200×4	9.9806e-01	2.2346e-01	8.1561e-02	1.1515e-02	32	369.66
$n_s = 50,\ N = 20$						
25×2	9.7546e-01	2.2508e-01	8.2371e-02	1.1565e-02	59	585.13
25×3	9.9133e-01	2.2360e-01	7.8884e-02	1.2713e-02	47	575.52
50×2	9.7077e-01	2.2694e-01	8.5091e-02	9.8544e-03	42	414.37
100×3	9.9671e-01	2.2319e-01	8.2164e-02	1.1081e-02	35	464.72
200×3	9.9778e-01	2.2298e-01	8.2363e-02	1.1263e-02	25	355.89
200×4	9.9792e-01	2.2319e-01	8.2052e-02	1.1127e-02	18	316.47
exata	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	-x-	-x-

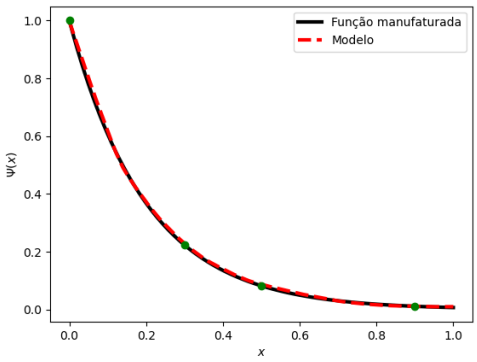
apresentados. Em comparação com a Figura 13 ($y = \tanh(v)$ nas camadas escondidas), nota-se, aqui, um melhor resultado no caso $n_s = 10$. Nos demais, $n_s = 25, 50$ e 100, resultados com boa precisão gráfica também foram obtidos.

Tendo em vista que o fluxo escalar médio de partículas é não negativo, testamos modelos MLPs com funções de ativação $y = relu(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e de saída, respectivamente. A Tabela 9 apresenta resultados comparativos para diferentes arquiteturas de rede, em formato similar às tabelas anteriores. Aqui, observamos um ganho na precisão dos resultados, quando comparados com a solução esperada $\hat{\Psi}$. O tempo computacional de execução é similar ao do caso anterior, $y = \text{id}(x)$ na camada de saída. Novamente, destacamos modelos de rede com arquitetura $1 - 200 \times 3 - 1$.

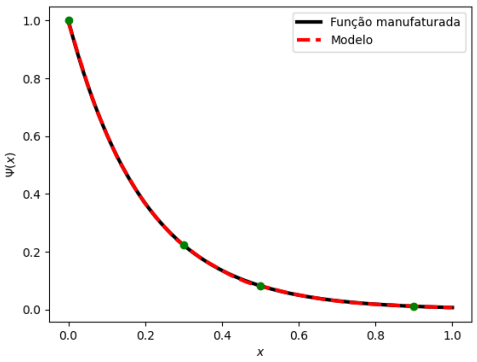
A Figura 15 apresenta comparações gráficas entre a solução esperada e a solução ANN-MoC como arquitetura de rede $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e de saída, respectivamente. Com $N = 2$, resultados do método com $n_s = 10, 25, 50$ e 100 são mostrados. Como nos testes anteriores,

Tabela 9 – Problema de Transporte 1. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 25, 50, 100, 200$ e $N = 2, 10, 20$.

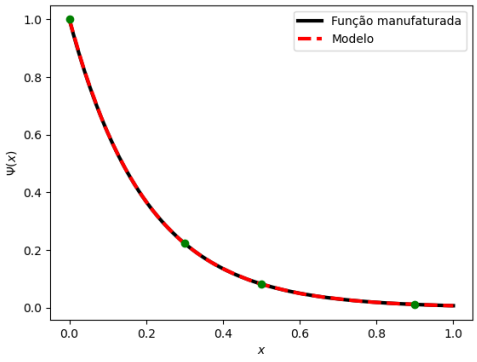
n_n/n_h	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 8,\ N = 2$						
25×2	8.8566e-01	3.8975e-01	2.0677e-01	6.9212e-02	64	18.82
25×4	9.9145e-01	2.1120e-01	6.3938e-02	1.2814e-02	43	24.83
50×3	9.7941e-01	2.1095e-01	5.9398e-02	8.7928e-03	64	25.49
50×4	9.7457e-01	2.0129e-01	7.8293e-02	1.2945e-02	33	17.75
100×3	9.8246e-01	2.1026e-01	6.9471e-02	7.3374e-03	40	21.85
$n_s = 25,\ N = 10$						
25×2	9.6663e-01	2.2579e-01	8.0928e-02	2.2186e-02	71	161.35
25×4	9.9117e-01	2.1794e-01	7.9823e-02	1.8649e-02	29	89.70
50×3	9.9827e-01	2.2450e-01	7.7378e-02	1.6685e-02	55	199.53
100×4	1.0001e+00	2.2308e-01	8.1927e-02	1.0594e-02	43	179.56
200×3	9.9357e-01	2.2346e-01	8.1939e-02	1.0746e-02	21	84.85
$n_s = 25,\ N = 20$						
100×3	8.1102e-01	2.1844e-01	8.0208e-02	1.1847e-02	42	285.96
100×4	9.9898e-01	2.2170e-01	8.1521e-02	1.2127e-02	39	365.79
200×3	9.9406e-01	2.2257e-01	8.2003e-02	1.1409e-02	34	247.91
200×4	9.9970e-01	2.2374e-01	8.2334e-02	1.0983e-02	29	255.86
$n_s = 50,\ N = 20$						
25×2	9.6608e-01	2.1381e-01	7.4681e-02	1.5267e-02	59	503.55
25×3	9.7009e-01	2.1861e-01	7.1497e-02	1.4319e-02	33	343.68
50×2	9.6910e-01	2.1602e-01	7.3322e-02	2.1434e-02	57	516.11
100×3	9.9825e-01	2.2280e-01	8.1926e-02	1.1308e-02	48	529.7
200×3	9.9906e-01	2.2314e-01	8.2183e-02	1.1342e-02	22	248.89
200×4	9.9895e-01	2.2291e-01	8.2417e-02	1.1107e-02	23	308.48
$n_s = 100,\ N = 10$						
25×3	9.9374e-01	2.2628e-01	7.8704e-02	1.1922e-02	45	391.68
25×4	9.9870e-01	2.2306e-01	8.1394e-02	1.1272e-02	37	384.48
50×3	9.9605e-01	2.2370e-01	8.2445e-02	1.1126e-02	55	565.12
100×4	9.9968e-01	2.2215e-01	8.2140e-02	1.1382e-02	52	644.78
200×3	9.9866e-01	2.2301e-01	8.1708e-02	1.1182e-02	24	274.09
$n_s = 200,\ N = 10$						
100×3	9.9879e-01	2.2344e-01	8.2199e-02	1.1105e-02	41	847.76
100×4	9.9840e-01	2.2295e-01	8.1820e-02	1.0981e-02	29	597.23
200×2	9.9946e-01	2.2309e-01	8.2019e-02	1.1133e-02	60	912.49
200×3	9.9920e-01	2.2303e-01	8.2080e-02	1.1080e-02	34	752.29
200×4	9.9914e-01	2.2299e-01	8.2031e-02	1.1109e-02	30	751.32
exata	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	-x-	-x-



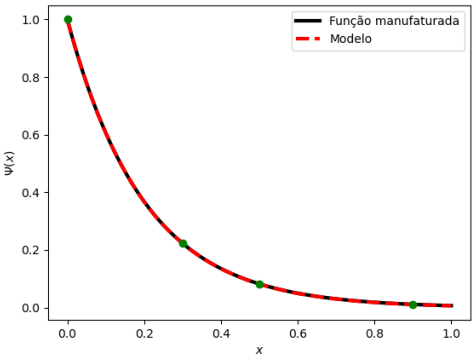
(a) $n_s = 10$



(b) $n_s = 25$



(c) $n_s = 50$



(d) $n_s = 100$

Figura 14 – Problema de Transporte 1. Comparação da solução ANN-MoC $y = \tilde{\Psi}(x)$ com a solução esperada $y = \hat{\Psi}(x)$ para $x \in \mathcal{D} = [0, 1]$. Rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(x)$ nas camadas escondidas e camada de saída, $N = 2$.

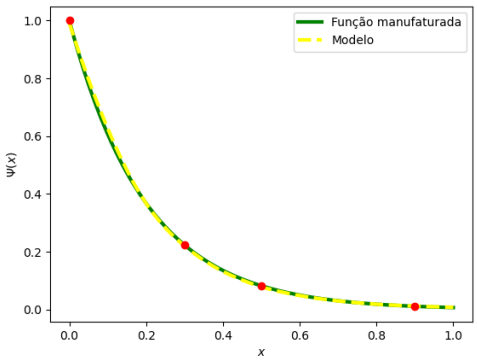
treinamentos com $n_s \geq$ forneceram resultados com boa precisão gráfica.

O método ANN-MoC, em resumo, consiste em usar uma RNA para estimar o fluxo de partículas médio em qualquer ponto do domínio. Alternativas clássicas, consistem em empregar um método de interpolação baseado nos pontos de colocação em uma malha predeterminada. Comparamos os resultados entre estas diferentes abordagem na Tabela 10. Nela, "ANN" denota os resultados com o método ANN-MoC, "Linear" para MoC com interpolação linear (Linear-MoC) e "Quadrática" para MoC com interpolação quadrática (Quadrática-MoC). Ainda, nestes casos (Linear-MoC e Quadrática-MoC) uma malha uniforme com n_s pontos de colocação é empregada. Lembramos que no caso ANN-MoC as amostras de pontos de colocação são randomizadas a cada iteração de fonte. Na Tabela 10, o modelo de rede empregado foi $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$, $y = \text{softplus}(v)$ nas camadas escondidas e de saída, respectivamente.

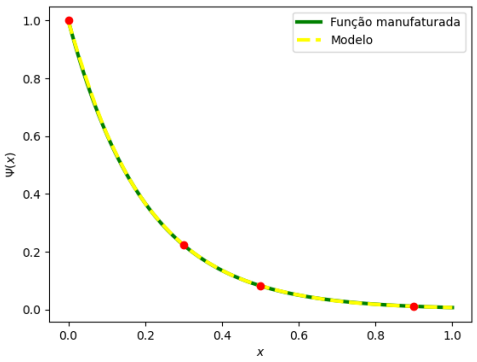
Os resultados da Tabela 10 mostram que o método ANN-MoC demanda um custo computacional significativamente maior. A precisão dos resultados, também é melhor nas

Tabela 10 – Problema de Transporte 1. Comparações entre os resultados do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. Modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$, $y = \text{softplus}(v)$ nas camadas escondidas e de saída.

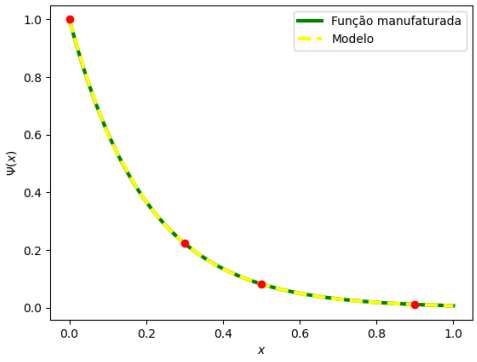
n_s	Tipo	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$	L	Tempo (s)
			$N = 10$				
	ANN	9.8882e-01	2.1870e-01	8.0975e-02	1.2319e-02	34	61.87
10	Linear	1.0039e+00	2.3875e-01	9.0478e-02	1.5218e-02	13	27.49
	Quadrática	9.9963e-01	2.2141e-01	8.1183e-02	1.0809e-02	10	6.28
	ANN	9.9317e-01	2.2267e-01	8.0868e-02	1.1348e-02	44	134.77
25	Linear	1.0022e+00	2.2606e-01	8.4075e-02	1.1922e-02	14	44.69
	Quadrática	1.0000e+00	2.2313e-01	8.2089e-02	1.1111e-02	20	27.41
	ANN	9.9785e-01	2.2301e-01	8.2063e-02	1.1858e-02	30	150.48
40	Linear	1.0009e+00	2.2535e-01	8.2838e-02	1.1343e-02	12	46.87
	Quadrática	1.0000e+00	2.2313e-01	8.2083e-02	1.1109e-02	22	55.75
	ANN	9.9908e-01	2.2352e-01	8.1954e-02	1.1134e-02	34	182.19
50	Linear	1.0005e+00	2.2378e-01	8.2560e-02	1.1317e-02	13	57.30
	Quadrática	1.0000e+00	2.2313e-01	8.2082e-02	1.1109e-02	20	57.28
	ANN	9.9831e-01	2.2316e-01	8.2076e-02	1.1116e-02	49	410.11
80	Linear	1.0003e+00	2.2344e-01	8.2274e-02	1.1188e-02	11	74.63
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	21	110.47
			$N = 20$				
	ANN	9.9862e-01	2.2282e-01	8.2122e-02	1.1150e-02	34	578.91
80	Linear	1.0002e+00	2.2335e-01	8.2264e-02	1.1188e-02	12	150.46
	Quadrática	1.0000e+00	2.2313e-01	8.2086e-02	1.1109e-02	21	194.16
	ANN	1.0000e+00	2.2318e-01	8.2108e-02	1.1057e-02	47	944.85
100	Linear	1.0001e+00	2.2351e-01	8.2174e-02	1.1150e-02	12	187.97
	Quadrática	1.0000e+00	2.2314e-01	8.2085e-02	1.1109e-02	20	263.98
	ANN	9.9933e-01	2.2316e-01	8.2032e-02	1.1113e-02	47	1385.46
125	Linear	1.0001e+00	2.2327e-01	8.2145e-02	1.1137e-02	20	354.76
	Quadrática	1.0000e+00	2.2313e-01	8.2080e-02	1.1109e-02	17	282.38
			$N = 40$				
	ANN	9.9876e-01	2.2322e-01	8.2113e-02	1.1112e-02	25	1783.54
150	Linear	1.0001e+00	2.2332e-01	8.2170e-02	1.1135e-02	13	591.74
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	23	859.77
exata	-x-	1.0000e+0	2.2313e-1	8.2085e-2	1.1109e-2	-x-	-x-



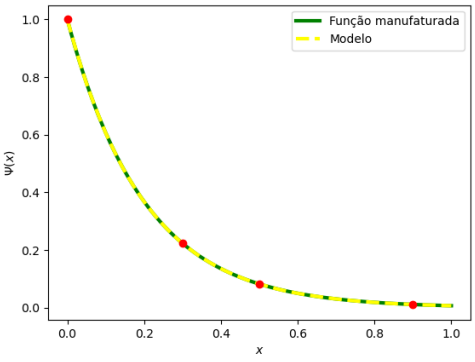
(a) $n_s = 10$



(b) $n_s = 25$



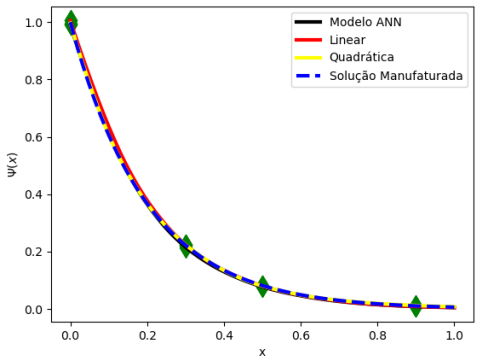
(c) $n_s = 50$



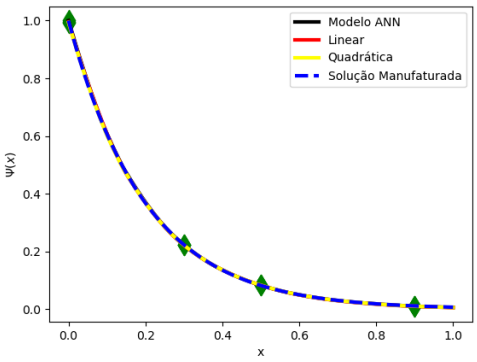
(d) $n_s = 100$

Figura 15 – Problema de Transporte 1. Comparações da solução ANN-MoC $y = \tilde{\Psi}(x)$ com a solução esperada $y = \hat{\Psi}(x)$ para $x \in \mathcal{D} = [0, 1]$. Rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(x)$ nas camadas escondidas e camada de saída, $N = 2$.

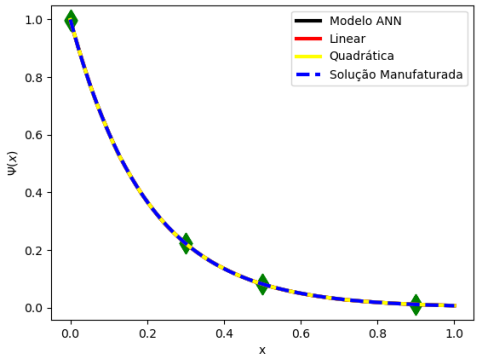
variantes clássicas Linear-MoC e Quadrática-MoC. Por outro lado, interpolação linear tem a desvantagem de ser de baixa ordem, enquanto que a quadrática reproduz em todos os casos melhor resultado esperado, em geral. Com isso, a escolha entre elas pode ser problema-dependente. Por outro lado, o método ANN-MoC tem a vantagem de permitir uma maior flexibilidade, conforme a escolha da função de ativação assumida na saída da rede. Embora menos preciso, o método proposto apresenta uma boa precisão gráfica, como já mostrado nas figuras anteriores e na Figura 16. Esta apresentada uma comparação gráfica entre as diferentes variantes aplicadas.



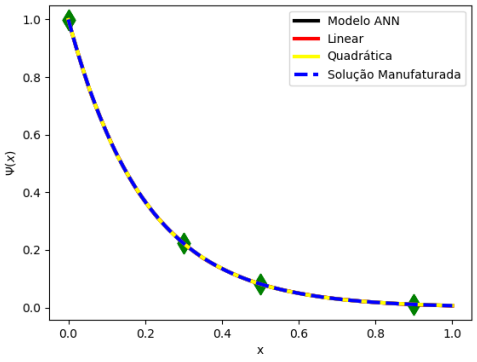
(a) $n_s = 10$



(b) $n_s = 25$



(c) $n_s = 50$



(d) $n_s = 100$

Figura 16 – Problema de Transporte 1. Comparações da solução ANN-MoC $y = \tilde{\Psi}(x)$ com variantes clássicas Linear-MoC e Quadrática-MoC. ANN-MoC com MLP de arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(x)$ nas camadas escondidas e camada de saída, $N = 2$.

4.3 Problema de Transporte 2: Solução Manufaturada Dependente da Direção

Vamos, agora, direcionar nosso estudo para uma problema de transporte de partículas com solução manufaturada

$$\hat{I}(x, \mu) = e^{-\alpha\sigma_t x}(1 + \mu), \tag{4.16}$$

onde $-1 < \mu < 1$, $\mu \neq 0$, é a direção da partícula, $x \in \mathcal{D} = [0, 1]$. Assumimos $\alpha = 5$ e $\sigma_t = 1$. Com isso, temos que o fluxo escalar médio esperado é

$$\hat{\Psi}(x) = \frac{1}{2}\int_{-1}^{1} \hat{I}(x, \mu')\, d\mu' \tag{4.17}$$

$$= e^{-\alpha\sigma_t x}. \tag{4.18}$$

Substituindo (4.16) e (4.17) em (1.1), calculamos a fonte de partículas associada,

como segue

$$\mu \cdot \frac{\partial}{\partial x}(e^{-\alpha\sigma_t x}(1+\mu)) + \sigma_t e^{-\alpha\sigma_t x}(1+\mu) = \sigma_s \Psi(x) + q(x,\mu) \tag{4.19}$$

$$\frac{\partial}{\partial x}(e^{-\alpha\sigma_t x}(1+\mu)) = -\alpha\sigma_t e^{-\alpha\sigma_t x}(1+\mu) + e^{-\alpha\sigma_t x}\frac{\partial}{\partial x}(1+\mu) \tag{4.20}$$

$$= -\alpha\sigma_t e^{-\alpha\sigma_t x}(1+\mu) + e^{-\alpha\sigma_t x}\frac{d\mu}{dx} \tag{4.21}$$

$$\mu\left(-\alpha\sigma_t e^{-\alpha\sigma_t x}(1+\mu) + e^{-\alpha\sigma_t x}\frac{d\mu}{dx}\right) + \sigma_t e^{-\alpha\sigma_t x}(1+\mu) = \frac{\sigma_s}{2}\Psi(x) + q(x,\mu) \tag{4.22}$$

$$\left(-\alpha\mu\sigma_t + \sigma_t + \mu\frac{d\mu}{dx}\right)e^{-\alpha\sigma_t x}(1+\mu) = \sigma_s\Psi(x) + q(x,\mu) \tag{4.23}$$

$$q(x,\mu) = e^{-\alpha\sigma_t x}(1+\mu)\left(-\alpha\mu\sigma_t + \sigma_t + \mu\frac{d\mu}{dx}\right) - \sigma_s\Psi(x) \tag{4.24}$$

$$q(x,\mu) = e^{-\alpha\sigma_t x}(1+\mu)\left(-\alpha\mu\sigma_t + \sigma_t\right) - \sigma_s\frac{1}{2}\int_{-1}^{1} e^{-\alpha\sigma_t x}(1+\mu)\,d\mu \tag{4.25}$$

$$= e^{-\alpha\sigma_t x}(1+\mu)\left(-\alpha\mu\sigma_t + \sigma_t\right) - \sigma_s\frac{1}{2}e^{-\alpha\sigma_t x}\cdot 2 \tag{4.26}$$

$$= e^{-\alpha\sigma_t x}(1+\mu)\left(-\alpha\mu\sigma_t + \sigma_t\right) - \sigma_s e^{-\alpha\sigma_t x} \tag{4.27}$$

Portanto, a fonte $q(x,\mu)$ é

$$q(x,\mu) = [(1-\alpha\mu)(1+\mu)\sigma_t - \sigma_s]e^{-\alpha\sigma_t x}. \tag{4.28}$$

Notamos que, aqui, o fluxo escalar médio (4.17) é o mesmo dos estudos realizados nas seções anteriores e, deles, esperamos que uma rede MLP de arquitetura $1-200\times 3-1$ seja suficiente para aproximar a solução. A dependência de grau 1 dos fluxos angulares em relação a direção nos indica que $N=2$ seja suficiente para o cálculo exato do fluxo escalar médio pela quadratura gaussiana. As Tabelas 11 e 12 apresentam uma série de resultados comparativos entre o método ANN-MoC e as variantes clássicas Linear-MoC e Quadrática-MoC. Várias configurações com $n_s = 10, 25, 50, 100, 150, 200$ e $N = 2, 4, 8, 16, 32, 64$. No ANN-MoC, o modelo de rede MLP é fixado com arquitetura $1-200\times 3-1$ e funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e da saída, respectivamente.

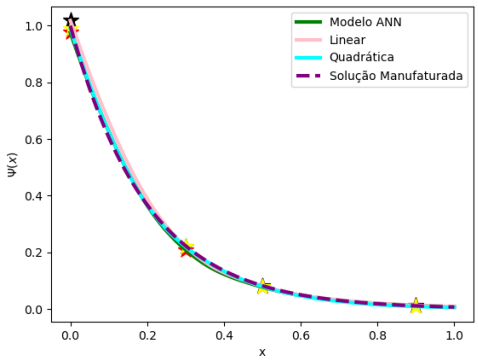
Os resultados apresentados nas Tabelas 11 e 12 permitem conclusões similares às obtidas no estudo de caso anterior. Ou seja, o método ANN-MoC, embora forneça boa precisão gráfica (consulte a Figura 17), é menos preciso e demanda um tempo computacional maior que as variantes clássicas Linear-MoC e Quadrática-MoC.

Tabela 11 – Problema de Transporte 2. Comparações entre os resultados do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. ANN-MoC com modelo de rede MLP de arquitetura $1 - 200 \times 3 - 1$ e funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e da saída, respectivamente. Casos $n_s = 10, 25, 50$ pontos de colocação.

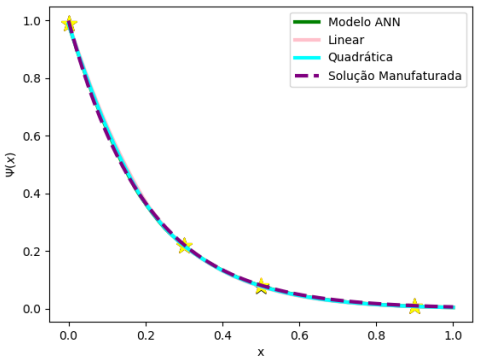
n_s	Tipo	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$	L	Tempo (s)
			$N = 2$				
	ANN	9.8300e-01	2.1041e-01	7.5189e-02	1.9411e-02	45	19.99
10	Linear	1.0043e+00	2.3381e-01	9.5105e-02	1.6457e-02	11	6.30
	Quadrática	9.9985e-01	2.2279e-01	8.2901e-02	1.0963e-02	18	2.75
			$N = 4$				
	ANN	9.9822e-01	2.2044e-01	7.3932e-02	9.8266e-03	62	54.31
10	Linear	1.0086e+00	3.8941e-01	2.1818e-01	1.7443e-02	10	11.65
	Quadrática	1.0009e+00	2.2625e-01	8.1844e-02	1.0892e-02	12	3.35
			$N = 8$				
	ANN	9.6736e-01	2.0711e-01	7.3386e-02	1.3365e-02	43	81.08
10	Linear	1.0062e+00	4.4509e-01	8.8537e-02	1.4867e-02	17	36.98
	Quadrática	1.0016e+00	2.2132e-01	8.2344e-02	1.1178e-02	15	8.44
			$N = 2$				
	ANN	9.8950e-01	2.2191e-01	8.0526e-02	1.2536e-02	53	34.78
25	Linear	1.0015e+00	2.2627e-01	8.3434e-02	1.1942e-02	11	7.69
	Quadrática	1.0000e+00	2.2304e-01	8.2093e-02	1.1111e-02	26	7.72
			$N = 8$				
	ANN	9.7683e-01	2.1883e-01	7.6186e-02	1.3443e-02	32	82.58
25	Linear	1.0031e+00	2.7303e-01	1.2843e-01	1.1990e-02	11	34.26
	Quadrática	1.0000e+00	2.2372e-01	8.2039e-02	1.1107e-02	14	17.69
			$N = 8$				
	ANN	9.9966e-01	2.2222e-01	8.1353e-02	1.1102e-02	65	258.34
50	Linear	1.0004e+00	2.2363e-01	8.3621e-02	1.1254e-02	12	48.26
	Quadrática	1.0000e+00	2.2315e-01	8.2094e-02	1.1110e-02	19	47.67
			$N = 16$				
	ANN	9.9050e-01	2.2600e-01	7.9582e-02	1.2348e-02	97	834.22
50	Linear	1.0004e+00	2.2360e-01	8.2514e-02	1.1246e-02	11	82.08
	Quadrática	1.0000e+00	2.2315e-01	8.2089e-02	1.1111e-02	17	84.96
			$N = 32$				
	ANN	9.9475e-01	2.2294e-01	8.1571e-02	1.2209e-02	68	1088.07
50	Linear	1.0003e+00	2.2429e-01	8.3540e-02	1.1269e-02	10	146.49
	Quadrática	1.0000e+00	2.2312e-01	8.2083e-02	1.1111e-02	14	128.39
exata	-x-	1.0000e+0	2.2313e-1	8.2085e-2	1.1109e-2	-x-	-x-

Tabela 12 – Problema de Transporte 2. Comparações entre os resultados do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. ANN-MoC com modelo de rede MLP de arquitetura $1 - 200 \times 3 - 1$ e funções de ativação $y = \text{relu}(v)$ e $y = \text{softplus}(v)$ nas camadas escondidas e da saída, respectivamente. Casos $n_s = 100, 150, 200$ pontos de colocação.

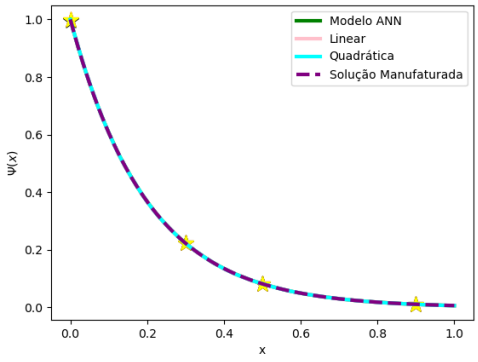
n_s	Tipo	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$	L	Tempo (s)
			$N = 2$				
	ANN	.9610e-01	2.2265e-01	8.2109e-02	1.0904e-02	73	160.96
100	Linear	1.0001e+00	2.2327e-01	8.2176e-02	1.1172e-02	11	17.86
	Quadrática	1.0000e+00	2.2313e-01	8.2084e-02	1.1109e-02	21	26.23
			$N = 4$				
	ANN	9.9709e-01	2.2339e-01	8.1837e-02	1.1289e-02	60	246.77
100	Linear	1.0001e+00	2.2323e-01	8.2278e-02	1.1148e-02	12	36.31
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	22	57.66
			$N = 8$				
	ANN	9.9786e-01	2.2272e-01	8.2600e-02	1.0892e-02	37	311.24
100	Linear	1.0001e+00	2.2325e-01	8.2393e-02	1.1156e-02	12	81.27
	Quadrática	1.0000e+00	2.2313e-01	8.2097e-02	1.1109e-02	19	98.26
			$N = 8$				
	ANN	9.9628e-01	2.2341e-01	8.2138e-02	1.1144e-02	50	572.58
150	Linear	1.0001e+00	2.2325e-01	8.2258e-02	1.1127e-02	15	130.68
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	22	162.92
			$N = 32$				
	ANN	9.9836e-01	2.2383e-01	8.0657e-02	1.1184e-02	56	2698.45
150	Linear	1.0000e+00	2.2320e-01	8.2125e-02	1.1129e-02	14	515.51
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	22	670.74
			$N = 2$				
	ANN	9.8130e-01	2.2502e-01	8.2140e-02	1.1275e-02	23	104.89
200	Linear	1.0000e+00	2.2317e-01	8.2148e-02	1.1123e-02	18	55.30
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	19	51.71
			$N = 16$				
	ANN	9.9923e-01	2.2356e-01	8.2109e-02	1.1696e-02	24	783.56
200	Linear	1.0000e+00	2.2323e-01	8.2137e-02	1.1126e-02	16	400.50
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	17	365.49
			$N = 32$				
	ANN	9.9794e-01	2.2293e-01	8.1894e-02	1.1055e-02	55	3353.35
200	Linear	1.0000e+00	2.2320e-01	8.2147e-02	1.1126e-02	13	611.21
	Quadrática	1.0000e+00	2.2313e-01	8.2085e-02	1.1109e-02	18	733.57
exata	-x-	1.0000e+0	2.2313e-1	8.2085e-2	1.1109e-2	-x-	-x-



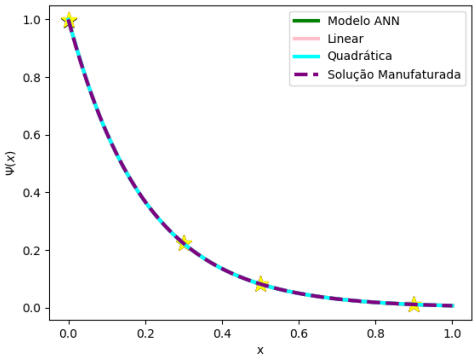
(a) $n_s = 10$, $N = 2$



(b) $n_s = 10$, $N = 4$



(c) $n_s = 50$, $N = 2$



(d) $n_s = 50$, $N = 4$

Figura 17 – Problema de Transporte 2. Aproximação das variantes ANN-MoC, Linear e Quadrática cujas funções de ativação ReLU nas camadas ocultas e Softplus na saída com arquitetura $1 - 200 \times 3 - 1$.

4.4 Problema de Transporte 3: Solução Manufatura Trigonométrica

Apresentamos um último estudo de caso de aplicação do método ANN-MoC para um problema de transporte com solução manufaturada trigonométrica

$$\hat{I}(x, \mu) = 1 + \operatorname{sen}(2\kappa\pi x) \cos\left(\frac{\pi}{2}\mu\right). \tag{4.29}$$

O fluxo escalar médio esperado é, então

$$\hat{\Psi}(x) = \frac{1}{2}\int_{-1}^{1} \hat{I}(x, \mu')\, d\mu' \tag{4.30}$$

$$= \frac{1}{2}\int_{-1}^{1} \left(1 + \operatorname{sen}(2\kappa\pi x) \cos\left(\frac{\pi}{2}\mu'\right)\right) d\mu' \tag{4.31}$$

$$= \frac{1}{2}\int_{-1}^{1} 1\, d\mu' + \frac{1}{2}\int_{-1}^{1} \operatorname{sen}(2\kappa\pi x) \cos\left(\frac{\pi}{2}\mu'\right) d\mu' \tag{4.32}$$

$$= \frac{1}{2}\left[\mu\right]_{-1}^{1} + \frac{1}{2}\operatorname{sen}(2\kappa\pi x)\int_{-1}^{1} \cos\left(\frac{\pi}{2}\mu'\right) d\mu' \tag{4.33}$$

$$= \frac{1}{2}(1-(-1)) + \frac{1}{2}\operatorname{sen}(2\kappa\pi x)\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos(u)\frac{2}{\pi}\,du \tag{4.34}$$

$$= 1 + \frac{1}{2}\operatorname{sen}(2\kappa\pi x)\left[\frac{2}{\pi}\operatorname{sen}(u)\right]_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \tag{4.35}$$

$$= 1 + \frac{1}{2}\cdot\frac{2}{\pi}\operatorname{sen}(2\kappa\pi x)\left(\operatorname{sen}\left(\frac{\pi}{2}\right) - \operatorname{sen}\left(-\frac{\pi}{2}\right)\right) \tag{4.36}$$

$$= 1 + \frac{1}{\pi}\operatorname{sen}(2\kappa\pi x)\cdot 2 \tag{4.37}$$

$$= 1 + \frac{2}{\pi}\operatorname{sen}(2\kappa\pi x) \tag{4.38}$$

A fonte de partículas associada é obtida substituindo (4.29) e (4.38) no problema (1.1), como segue

$$\mu\frac{\partial}{\partial x}\left(1+\operatorname{sen}(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right)\right)+\sigma_t\left(1+\operatorname{sen}(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right)\right) = \sigma_s\Psi(x)+q(x,\mu) \tag{4.39}$$

$$2\mu\kappa\pi\cos(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right) + \sigma_t\left(1+\operatorname{sen}(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right)\right) = \sigma_s\Psi(x) + q(x,\mu) \tag{4.40}$$

$$q(x,\mu) = 2\mu\kappa\pi\cos(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right) + \sigma_t\left(1+\operatorname{sen}(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right)\right) - \sigma_s\Psi(x) \tag{4.41}$$

Agora, vamos determinar $\Psi(x)$. Segue Portanto,

$$\begin{aligned} q(x,\mu) = {} & 2\mu\kappa\pi\cos(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right) + \sigma_t\left(1+\operatorname{sen}(2\kappa\pi x)\cos\left(\frac{\pi}{2}\mu\right)\right) \\ & - \sigma_s\left(1+\frac{2}{\pi}\operatorname{sen}(2\kappa\pi x)\right). \end{aligned} \tag{4.42}$$

Neste estudo de caso, o método ANN-MoC com modelo de rede MLP e função de ativação $y = \text{softplus}(v)$ na camada de saída não obteve resultados convergentes, isto é, não atende o critério de parada definido. Desta forma, no que segue, modelos MLP com função de ativação $y = \text{id}(x)$ são apresentados. Tendo em vista que $\hat{\Psi}$ é outra classe de função do que nos estudos anteriores, aqui, voltamos a investigar arquiteturas com funções de ativação $y = \tanh(v)$ e $y = \text{relu}(v)$ nas camadas escondidas.

Na Tabela 13 apresentamos resultados de testes do método ANN-MoC com modelo de rede MLP com funções de ativação $y = \tanh(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Mesmo utilizando um número reduzido de pares na quadratura gaussiana (com $N = 4$), observamos que os resultados obtidos pelo modelo ANN-MoC foram consistentes em relação aos resultados esperados. Novamente, observamos que redes com arquitetura $1 - 200 \times 3 - 1$ forneceram resultados adequados.

Agora, na Tabela 14, voltamos a testar redes com funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e de saída, respectivamente. Observamos que as escolhas $n_s = 200$ e $N = 4$ apresentaram bons resultados em todas as arquiteturas de rede testadas. Em comparação com os resultados com $y = \tanh(v)$ (tabela anterior), uma pequena

Tabela 13 – Problema de Transporte 3. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8, 25, 50, 100, 200$ e $N = 2, 4, 8$.

$n_n \backslash n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 8,\ N = 2$						
25×2	1.3192e+00	1.1067e+00	9.4816e-01	6.1565e-01	47	5.39
25×4	1.2481e+00	1.0374e+00	8.5942e-01	4.7018e-01	18	2.80
50×3	1.4369e+00	1.2225e+00	1.0443e+00	6.2946e-01	16	2.37
50×4	1.3527e+00	1.1285e+00	9.4737e-01	5.5284e-01	10	1.66
100×3	1.3938e+00	1.2538e+00	1.0004e+00	3.0171e-01	29	4.84
$n_s = 25,\ N = 2$						
25×2	1.5521e+00	1.2519e+00	1.0217e+00	5.2465e-01	26	9.30
25×4	1.5078e+00	1.3121e+00	1.0970e+00	5.2183e-01	14	6.86
50×3	1.0246e+00	1.6054e+00	1.2372e+00	4.5115e-01	35	15.19
100×4	9.9916e-01	1.6106e+00	1.1906e+00	4.3003e-01	51	26.44
200×3	9.9885e-01	1.6116e+00	1.1886e+00	4.2760e-01	86	40.69
$n_s = 25,\ N = 4$						
50×3	1.0214e+00	1.6364e+00	1.1890e+00	4.7178e-01	69	59.41
100×4	9.9821e-01	1.6318e+00	1.1967e+00	4.1806e-01	53	53.96
200×3	1.0018e+00	1.6294e+00	1.2023e+00	4.2968e-01	49	46.19
200×4	1.0090e+00	1.6321e+00	1.1989e+00	4.1386e-01	48	54.01
$n_s = 50,\ N = 8$						
25×3	1.0120e+00	1.6372e+00	1.1947e+00	4.7753e-01	51	169.73
50×2	1.5485e+00	1.2640e+00	1.0221e+00	4.8761e-01	13	37.93
100×3	1.0058e+00	1.6285e+00	1.1977e+00	4.0562e-01	45	154.42
200×3	1.0021e+00	1.6318e+00	1.1967e+00	4.0793e-01	63	245.12
$n_s = 100,\ N = 2$						
25×3	9.9745e-01	1.6090e+00	1.1883e+00	4.2776e-01	90	169.57
50×3	1.0018e+00	1.6106e+00	1.1892e+00	4.2779e-01	85	154.26
100×4	1.0007e+00	1.6105e+00	1.1900e+00	4.2685e-01	40	89.55
200×3	1.0001e+00	1.6116e+00	1.1912e+00	4.2630e-01	42	88.00
$n_s = 200,\ N = 4$						
50×3	9.9901e-01	1.6295e+00	1.1944e+00	4.0523e-01	61	425.36
100×2	9.9368e-01	1.6298e+00	1.1946e+00	4.0718e-01	100	608.92
100×3	1.0021e+00	1.6310e+00	1.1962e+00	4.0771e-01	44	323.49
200×3	1.0035e+00	1.6307e+00	1.1961e+00	4.0843e-01	55	445.16
exata	1.0000e+00	1.6316e+00	1.1967e+00	4.0809e-01	-x-	-x-

Tabela 14 – Problema de Transporte 3. Estimativas ANN-MoC $\tilde{\Psi}(x)$ para diferentes arquiteturas MLPs com funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas ocultas e na camada de saída, respectivamente. Comparação com valores estimados $\hat{\Psi}(x)$ para $x = 0.0, 0.3, 0.5, 0.9$, $n_s = 8, 25, 50, 100, 200$ e $N = 2, 4, 8$.

$n_n \backslash n_h$	Soluções				L	Tempo (s)
	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$		
$n_s = 8,\ N = 2$						
25×2	6.8422e-01	7.3770e-01	7.6512e-01	7.9874e-01	6	0.84
50×3	1.4154e+00	1.2514e+00	1.1068e+00	6.4614e-01	27	5.25
50×4	1.0394e+00	8.8998e-01	7.8563e-01	6.0391e-01	14	2.79
100×3	1.3259e+00	1.2167e+00	1.0546e+00	5.3157e-01	21	4.72
$n_s = 25,\ N = 2$						
25×2	1.3718e+00	1.4374e+00	1.1300e+00	4.7827e-01	41	16.32
50×3	1.0089e+00	1.6191e+00	1.1714e+00	4.6943e-01	73	43.31
100×4	1.0026e+00	1.6099e+00	1.1900e+00	4.2858e-01	58	46.08
200×3	1.0027e+00	1.6109e+00	1.1874e+00	4.1793e-01	45	30.25
$n_s = 25,\ N = 4$						
50×3	1.0266e+00	1.6214e+00	1.1738e+00	4.6884e-01	62	68.29
200×3	1.0005e+00	1.6324e+00	1.1935e+00	4.1393e-01	42	59.32
200×4	1.0115e+00	1.6379e+00	1.1952e+00	4.0511e-01	20	33.75
$n_s = 50,\ N = 8$						
25×2	1.1126e+00	1.5947e+00	1.1286e+00	3.5517e-01	30	89.34
25×3	1.0179e+00	1.6317e+00	1.1867e+00	3.9234e-01	55	216.47
50×2	1.0320e+00	1.6872e+00	1.1489e+00	4.4142e-01	59	181.50
100×3	1.0065e+00	1.6313e+00	1.1947e+00	4.0961e-01	27	114.83
200×3	1.0004e+00	1.6307e+00	1.1929e+00	4.0784e-01	46	195.49
$n_s = 100,\ N = 2$						
25×3	1.0045e+00	1.6079e+00	1.1885e+00	4.2934e-01	77	155.52
25×4	1.0066e+00	1.6072e+00	1.1879e+00	4.2124e-01	36	83.97
50×3	1.0059e+00	1.6110e+00	1.1888e+00	4.2165e-01	53	105.09
100×4	1.0036e+00	1.6080e+00	1.1904e+00	4.2721e-01	33	81.35
200×3	1.0022e+00	1.6138e+00	1.1903e+00	4.2729e-01	52	125.53
$n_s = 200,\ N = 4$						
50×3	1.0044e+00	1.6367e+00	1.1952e+00	4.0819e-01	27	189.87
50×4	1.0092e+00	1.6261e+00	1.1899e+00	4.0499e-01	36	318.37
100×2	1.0050e+00	1.6396e+00	1.1922e+00	4.0521e-01	34	214.23
100×3	1.0024e+00	1.6317e+00	1.1950e+00	4.0759e-01	33	255.57
200×3	1.0177e+00	1.6481e+00	1.2123e+00	4.1955e-01	40	349.90
200×4	1.0013e+00	1.6316e+00	1.1968e+00	4.0834e-01	20	215.40
exata	1.0000e+00	1.6316e+00	1.1967e+00	4.0809e-01	-x-	-x-

vantagem pode ser observada, em especial ao número de iterações de fonte requeridas para a convergência do método.

Passamos, então, a comparação do método ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC para o problema de transporte 3. Com base nos estudos anteriores, os resultados apresentados a partir daqui foram gerados com arquiteturas de rede $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Lembrando que $\sigma_t = \kappa + \sigma_s$, comparamos os desempenhos das variantes para os casos $\sigma_s = 0.1, 0.5, 0.9$, sempre com $\sigma_t = 1$. Consulte as Tabelas 15, 16 e 17. Observamos que, conforme o valor de σ_s aumenta é esperado uma demanda maior de iterações de fonte, o que de fato ocorre nas variantes Linear-MoC e Quadrática-MoC. Notavelmente, esta característica não é diretamente observável nos resultados com o método ANN-MoC. Este fato aparentemente inesperado pode ser explicado pela estocasticidade do método proposto.

Por fim, passamos a uma análise gráfica das soluções estimadas pelos métodos ANN-MoC, Linear-MoC e Quadrática-MoC em comparação com a solução esperada. No que segue, no método ANN-MoC um modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$ com função de ativação $y = \text{id}(v)$ na camada de saída. Primeiramente, analisamos o problema de transporte com $\sigma_s = 0.1$, $\sigma_t = 1$. Na Figura 18 temos os resultados com $n_s = 10, 25$ e $N = 2, 4, 8$ e modelos de redes com $y = \tanh(v)$ nas camadas escondidas. Observa-se que, similarmente aos casos anteriores, a solução ANN-MoC tem precisão gráfica comparável às variantes clássicas para $n_s = 25, 50$, consulte também a Figura 19.

Nas Figuras 20 e 21, apresentamos as comparações gráficas entre as variantes do método MoC testadas. Aqui, o método ANN-MoC é avaliado com um modelo de rede MLP $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Em comparação aos testes anteriores, pode-se observar um pequeno ganho em precisão gráfica quando comparado a redes com $y = \tanh(v)$ nas camadas escondidas.

Uma última comparação gráfica é apresentada na Figura 22, em que analisamos resultados com todas as variantes do método MoC para $\kappa = 0.5, 0.75$, $\sigma_t = 1$. Para o ANN-MoC empregamos um modelo de rede $1-200\times3-1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Os resultados foram obtidos com $n_s = 100$ e $N = 4$. Observamos que o método ANN-MoC novamente apresentou boa precisão gráfica.

Tabela 15 – Problema de Transporte 3. Comparação de resultados com as variantes Quadrática-MoC, Linear-MoC e ANN-MoC, fixados $\kappa = 0.9$ e $\sigma_s = 0.1$. Rede de arquitetura $1 - 200 \times 3 - 1$.

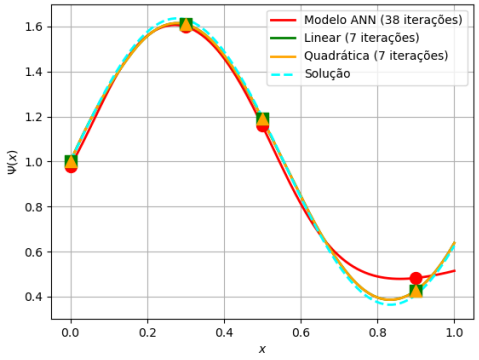
n_s	Tipo	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$	L	Tempo (s)
			$N = 2$				
	ANN	9.9786e-01	1.5973e+00	1.1808e+00	4.5613e-01	43	24.13
25	Linear	9.9970e-01	1.6083e+00	1.1903e+00	4.3121e-01	7	2.27
	Quadrática	9.9974e-01	1.6110e+00	1.1903e+00	4.2736e-01	7	1.75
			$N = 4$				
	ANN	9.9992e-01	1.6303e+00	1.1935e+00	4.1267e-01	62	63.29
25	Linear	9.9995e-01	1.6287e+00	1.1967e+00	4.1211e-01	7	4.12
	Quadrática	1.0000e+00	1.6316e+00	1.1967e+00	4.0811e-01	7	3.65
			$N = 8$				
	ANN	9.9752e-01	1.6315e+00	1.1991e+00	4.1296e-01	65	132.88
25	Linear	9.9995e-01	1.6287e+00	1.1967e+00	4.1212e-01	7	8.66
	Quadrática	1.0000e+00	1.6316e+00	1.1967e+00	4.0811e-01	7	7.34
			$N = 2$				
	ANN	1.0008e+00	1.6115e+00	1.1911e+00	4.2747e-01	58	69.00
50	Linear	9.9973e-01	1.6102e+00	1.1900e+00	4.2768e-01	7	4.38
	Quadrática	9.9974e-01	1.6110e+00	1.1903e+00	4.2734e-01	7	3.77
			$N = 4$				
	ANN	1.0072e+00	1.6364e+00	1.2016e+00	4.1266e-01	69	148.63
50	Linear	9.9999e-01	1.6307e+00	1.1964e+00	4.0845e-01	7	8.37
	Quadrática	1.0000e+00	1.6316e+00	1.1967e+00	4.0809e-01	7	7.15
			$N = 4$				
	ANN	1.0017e+00	1.6318e+00	1.1967e+00	4.0673e-01	53	240.93
100	Linear	1.0000e+00	1.6314e+00	1.1966e+00	4.0818e-01	7	18.30
	Quadrática	1.0000e+00	1.6316e+00	1.1967e+00	4.0809e-01	7	15.26
			$N = 2$				
	ANN	9.9621e-01	1.6034e+00	1.1790e+00	4.1696e-01	29	150.39
200	Linear	9.9974e-01	1.6110e+00	1.1903e+00	4.2736e-01	7	17.53
	Quadrática	9.9974e-01	1.6110e+00	1.1903e+00	4.2734e-01	7	14.44
			$N = 4$				
	ANN	1.0011e+00	1.6311e+00	1.1964e+00	4.0615e-01	44	343.59
200	Linear	1.0000e+00	1.6315e+00	1.1967e+00	4.0811e-01	7	35.42
	Quadrática	1.0000e+00	1.6316e+00	1.1967e+00	4.0809e-01	8	33.12
exata	-x-	1.0000e+00	1.6316e+00	1.1967e+00	4.0809e-01	-x-	-x-

Tabela 16 – Problema de Transporte 3. Comparação de resultados com as variantes Quadrática-MoC, Linear-MoC e ANN-MoC, fixados $\kappa = 0.5$ e $\sigma_s = 0.5$. Rede de arquitetura $1 - 200 \times 3 - 1$.

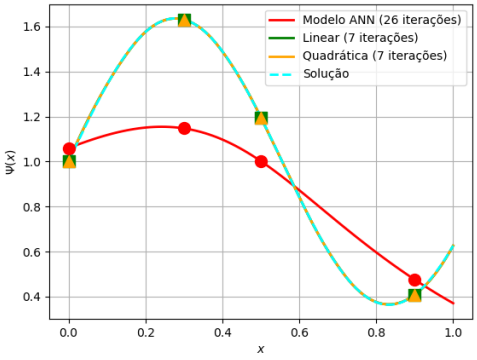
n_s	Tipo	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$	L	Tempo (s)
			$N = 2$				
	ANN	9.9847e-01	1.4947e+00	1.6096e+00	1.1840e+00	49	25.16
25	Linear	9.9632e-01	1.4924e+00	1.6103e+00	1.1857e+00	13	4.16
	Quadrática	9.9647e-01	1.4933e+00	1.6105e+00	1.1863e+00	13	3.88
			$N = 4$				
	ANN	1.0042e+00	1.5194e+00	1.6361e+00	1.1956e+00	56	53.06
25	Linear	9.9984e-01	1.5140e+00	1.6363e+00	1.1961e+00	14	8.57
	Quadrática	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	14	7.32
			$N = 8$				
	ANN	1.0014e+00	1.5160e+00	1.6363e+00	1.1962e+00	48	90.69
25	Linear	9.9984e-01	1.5140e+00	1.6363e+00	1.1961e+00	15	17.25
	Quadrática	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	15	15.23
			$N = 2$				
	ANN	9.9681e-01	1.4933e+00	1.6106e+00	1.1861e+00	77	82.32
50	Linear	9.9647e-01	1.4933e+00	1.6105e+00	1.1863e+00	14	7.22
	Quadrática	9.9644e-01	1.4930e+00	1.6102e+00	1.1862e+00	13	7.69
			$N = 4$				
	ANN	1.0024e+00	1.5159e+00	1.6360e+00	1.1954e+00	48	97.26
50	Linear	9.9996e-01	1.5147e+00	1.6362e+00	1.1966e+00	14	16.46
	Quadrática	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	14	14.23
			$N = 4$				
	ANN	1.0007e+00	1.5156e+00	1.6375e+00	1.1973e+00	51	186.51
100	Linear	9.9999e-01	1.5150e+00	1.6365e+00	1.1967e+00	15	35.65
	Quadrática	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	15	30.61
			$N = 2$				
	ANN	9.9387e-01	1.4898e+00	1.6071e+00	1.1833e+00	31	133.32
200	Linear	9.9647e-01	1.4933e+00	1.6105e+00	1.1863e+00	14	33.97
	Quadrática	9.9647e-01	1.4933e+00	1.6105e+00	1.1863e+00	16	32.77
			$N = 4$				
	ANN	9.9999e-01	1.5137e+00	1.6360e+00	1.1957e+00	30	244.10
200	Linear	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	16	76.89
	Quadrática	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	16	66.60
exata	-x-	1.0000e+00	1.5150e+00	1.6366e+00	1.1967e+00	-x-	-x-

Tabela 17 – Problema de Transporte 3. Comparação de resultados com as variantes Quadrática-MoC, Linear-MoC e ANN-MoC, fixados $\kappa = 0.1$ e $\sigma_s = 0.9$. Rede de arquitetura $1 - 200 \times 3 - 1$.

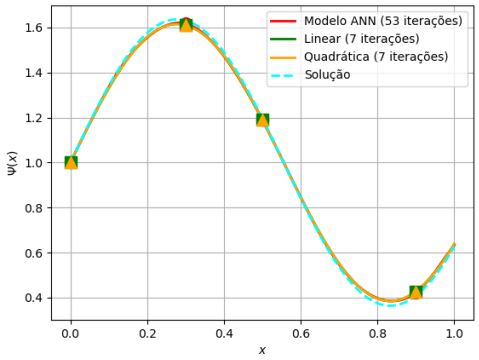
n_s	Tipo	$\Psi(0.0)$	$\Psi(0.3)$	$\Psi(0.5)$	$\Psi(0.9)$	L	Tempo (s)
			$N = 2$				
	ANN	9.9653e-01	1.1099e+00	1.1840e+00	1.3246e+00	54	30.35
25	Linear	9.9636e-01	1.1100e+00	1.1842e+00	1.3243e+00	22	6.61
	Quadrática	9.9636e-01	1.1100e+00	1.1842e+00	1.3244e+00	23	6.14
			$N = 4$				
	ANN	9.9950e-01	1.1193e+00	1.1965e+00	1.3413e+00	37	35.36
25	Linear	9.9999e-01	1.1193e+00	1.1967e+00	1.3411e+00	26	14.20
	Quadrática	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	25	12.00
			$N = 8$				
	ANN	9.9981e-01	1.1193e+00	1.1965e+00	1.3413e+00	44	80.83
25	Linear	9.9999e-01	1.1193e+00	1.1967e+00	1.3411e+00	27	29.64
	Quadrática	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	27	24.85
			$N = 2$				
	ANN	9.9571e-01	1.1098e+00	1.1836e+00	1.3247e+00	33	34.68
50	Linear	9.9636e-01	1.1100e+00	1.1842e+00	1.3244e+00	23	13.68
	Quadrática	9.9636e-01	1.1100e+00	1.1842e+00	1.3244e+00	23	11.79
			$N = 4$				
	ANN	1.0002e+00	1.1189e+00	1.1964e+00	1.3412e+00	19	36.79
50	Linear	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	26	30.63
	Quadrática	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	26	26.16
			$N = 4$				
	ANN	9.9937e-01	1.1190e+00	1.1967e+00	1.3414e+00	51	199.21
100	Linear	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	26	62.65
	Quadrática	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	26	54.45
			$N = 2$				
	ANN	9.9581e-01	1.1101e+00	1.1840e+00	1.3246e+00	31	136.81
200	Linear	9.9636e-01	1.1100e+00	1.1842e+00	1.3244e+00	23	55.74
	Quadrática	9.9636e-01	1.1100e+00	1.1842e+00	1.3244e+00	24	53.31
			$N = 4$				
	ANN	9.9922e-01	1.1191e+00	1.1966e+00	1.3413e+00	40	342.52
200	Linear	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	26	125.94
	Quadrática	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	26	106.76
exata	-x-	1.0000e+00	1.1193e+00	1.1967e+00	1.3411e+00	-x-	-x-



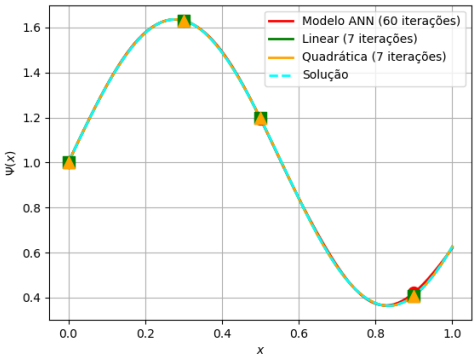
(a) $n_s = 10$, $N = 2$



(b) $n_s = 10$, $N = 8$

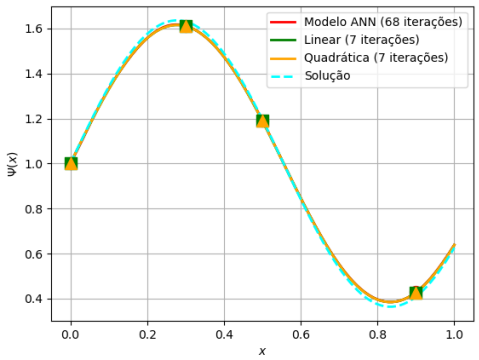


(c) $n_s = 25$, $N = 2$

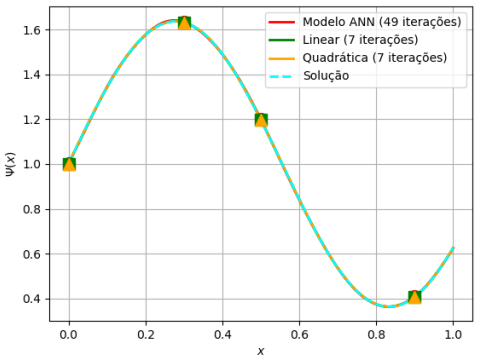


(d) $n_s = 25$, $N = 4$

Figura 18 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$.

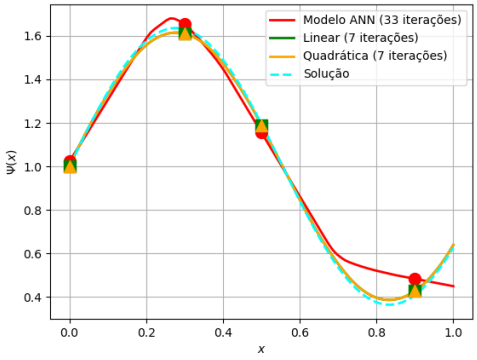


(a) $n_s = 50$, $N = 2$

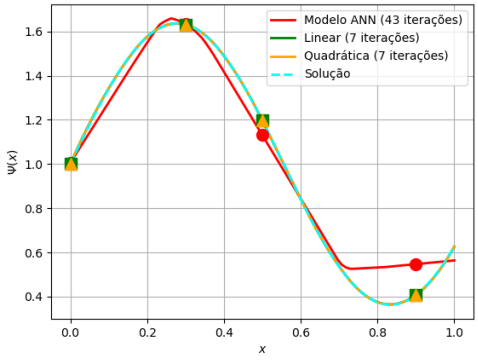


(b) $n_s = 50$, $N = 8$

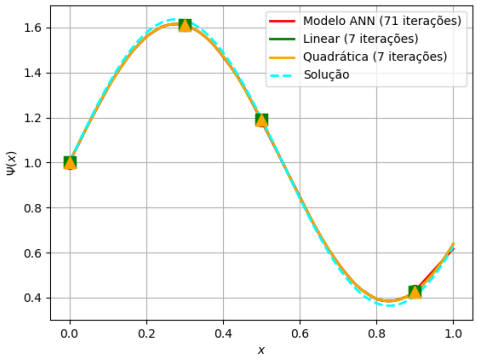
Figura 19 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \tanh(v)$ e $y = \mathrm{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$, $n_s = 50$ e $N = 2, 8$.



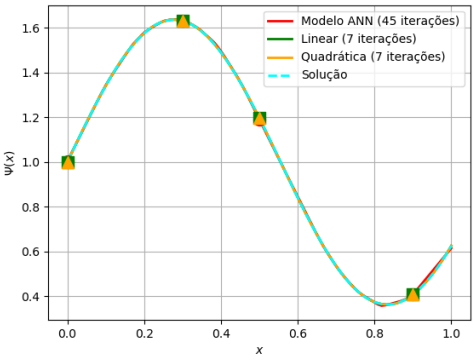
(a) $n_s = 10,\ N = 2$



(b) $n_s = 10,\ N = 8$

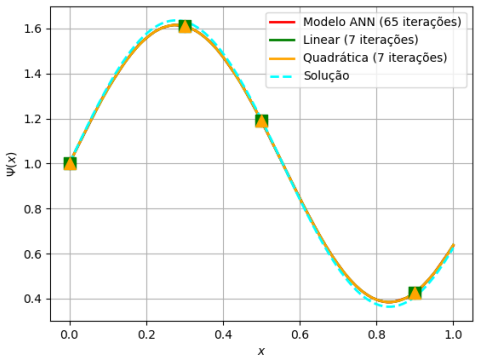


(c) $n_s = 25,\ N = 2$

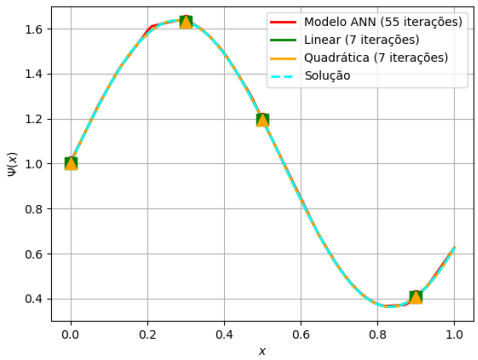


(d) $n_s = 25,\ N = 4$

Figura 20 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$, $n_s = 10, 25$ e $N = 2, 4, 8$.

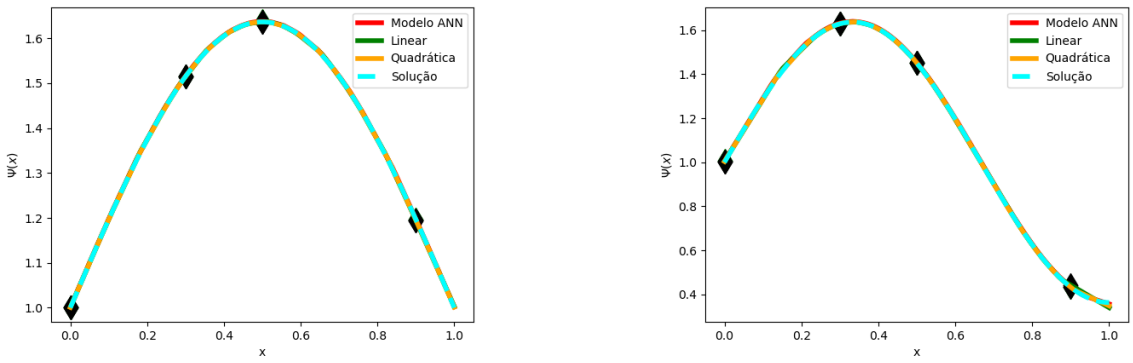


(a) $n_s = 50,\ N = 2$



(b) $n_s = 50,\ N = 8$

Figura 21 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.1$ e $\sigma_t = 1$, $n_s = 50$ e $N = 2, 8$.



(a) $\kappa = 0.50$ (b) $\kappa = 0.25$

Figura 22 – Problema de Transporte 3. Comparação gráfica das soluções ANN-MoC, Linear-MoC e Quadrática-MoC com a solução esperada. ANN-MoC com modelo de rede MLP com arquitetura $1 - 200 \times 3 - 1$, funções de ativação $y = \text{relu}(v)$ e $y = \text{id}(v)$ nas camadas escondidas e da saída, respectivamente. Parâmetros $\sigma_s = 0.5, 0.75$ e $\sigma_t = 1$, $n_s = 100$ e $N = 4$.

5 Considerações Finais

Apresentamos o método ANN-MoC para a solução de problemas de transporte de partículas neutras em geometria 1D. O novo método consiste no acoplamento de uma rede neural artificial com o método das características. Este fornece uma forma explícita da solução, mas que dependente do fluxo escalar de partículas. Seguindo um esquema de iteração de fonte, uma rede neural artificial (ANN) é treinada para fornecer estimativas deste fluxo em qualquer ponto do domínio computacional. Trata-se de uma alternativa à aplicação de técnicas clássicas de interpolação e aproximação de funções. O ANN-MoC é um método conceitual que busca usufruir da flexibilidade, adaptabilidade e boas propriedades de aproximação de funções por ANNs.

A análise do método proposto foi realizada por sua aplicação em três problemas de transporte com soluções manufaturadas distintas. No primeiro, assumimos que os fluxos angulares de partículas têm um decaimento exponencial. Para este problema, investigamos a aplicabilidade do método ANN-MoC e o estudamos com diferentes arquiteturas de ANNs. No segundo problema, os fluxos angulares de partículas são dependentes da direção e, também, têm um decaimento exponencial. O foco da análise, aqui, foi na comparação do ANN-MoC com as variantes clássicas Linear-MoC e Quadrática-MoC. Por fim, estudamos a aplicação em um problema de transporte com fluxos de partículas descritos por uma combinação de funções trigonométricas. Voltamos a estudar os impactos na eficiência do ANN-MoC com diferentes escolhas de arquiteturas de rede. Bem como, fizemos comparações com as variantes clássicas do método das características. Desses estudos, podemos concluir que o método proposto é, em geral, menos preciso e computacionalmente mais custoso, quando comparado com as suas variantes clássicas. Entretanto, o ANN-MoC forneceu resultados com boa precisão gráfica para todos os problemas estudados.

O método ANN-MoC oferece um novo paradigma para a resolução de problemas de transporte de partículas neutras. Uma de suas vantagens reside na flexibilidade e adaptabilidade das ANNs na aproximação de funções diversas. Mais precisamente, o método pode ser adaptado a diferentes problemas pela escolha da arquitetura da RNA a ser empregada e, em especial, de suas funções de ativação. O método também abre novas oportunidades, em especial, permite o acoplamento de dados (*a priori* ou *in loco*) na resolução de problemas diretos e inversos. Ainda, sua otimização pode ser possível através do aperfeiçoamento das técnicas de treinamento, na implementação das ANNs em GPU (*graphics processing unit*) e com a aplicação de métodos de aceleração nas iterações de fonte. Por fim, trabalhos futuros podem almejar sua aplicação para problemas anisotrópicos e em geometria 2D.

Referências

[1] G.S. Abdoulaev and A.H. Hielscher. Three-dimensional optical tomography with the equation of radiative transfer. *Journal of Electronic Imaging*, 12:594–601, 2003.

[2] M.L. Adams. Fast iterative methods for discrete-ordinates particle transport calculations. *Progress in Nuclear Energy*, 40:3–159, 2002.

[3] D. Anderson and G. McNeill. *Artificial Neural Networks Technology*. Rome Laboratory, New York., 1992.

[4] M. Biehl. Supervised learning - an introduction: Lectures given at the 30th canary islands winter school of astrophysics, 2019.

[5] R. B. Bird, W. E. Stewart, and Edwin N. Lightfoot. *Transport phenomena*. John Wiley & Sons, 2007.

[6] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] S. Chakraverty and S.K. Jeswal. *Applied Artificial Neural Network Methods for Engineers and Scientists: Solving Algebraic Equations*. World Scientific, 2021.

[8] S. Chakraverty and J.S. Kumar. *Applied Artificial Neural Network Methods for Engineers and Scientists: Solving Algebraic Equations*. National Institute of Technology Rourkela, India, 2021.

[9] J. L. Consalvi, F. Nmira, and W. Kong. On the modeling of the filtered radiative transfer equation in large eddy simulations of lab-scale sooting turbulent diffusion flames. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 2018.

[10] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. Incorporating second-order functional knowledge for better option pricing. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[11] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.

[12] L.C. Evans. *Partial Differential Equations*. AMS, 2. edition, 2010.

[13] S.J. Farlow. *Partial Differential Equations for Scientists and Engineers*. New York, USA, 1982.

[14] M. Frank, M. Seaïd, A. Klar, R. Pinnau, G. Thömmes, and J. Janicka. A comparison of approximate models for radiation in gas turbines. *Progress in Computational Fluid Dynamics, An International Journal*, 4:191–197, 2004.

[15] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[16] E. Guresen and G. Kayakutlu. Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science, Elsevier Ltd*, 3:426–433, 2011.

[17] K. Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc.325 Chestnut St. Suite 800 Philadelphia, PAUnited States, 1997.

[18] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2nd edition, 2019.

[19] S. Haykin. *Redes Neurais: Princípios e prática*. Porto Alegre, RS, 2. edition, 2001.

[20] S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, 3. edition, 2009.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. CRC Press, 1. edition, 1991.

[23] J. E. Hoogenboom. Adjoint monte carlo photon transport in continuous energy mode with discrete photons from annihilation. *Proceedings of the PHYSOR 2000*, 2000.

[24] J. R. Howell. The monte carlo method in radiative heat transfer. *Journal of Heat Transfer*, 120:547–560, Aug 1, 1998.

[25] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[26] P.H.A. Konzen. Ann-moc method for solving unidimensional neutral particle transport problems. In *Apresentado em XLII CNMAC 2023*, Bonito, Brasil. Disponível em: arXiv:2307.03114, 2023.

[27] J.R. Lamarsh and A.J. Baratta. *Introduction to Nuclear Engineering*. Prentice Hall, New Jersey, 3. edition, 2001.

[28] E.W. Larsen, G. Thömmes, A. Klar, M. Seaïd, and T. Götz. Simplified P_N approximations to the equations of radiative heat transfer and applications. *Journal of Computational Physics*, 183:652–675, 2002.

[29] L. G. Leal. *Advanced Transport Phenomena: Fluid Mechanics and Convective Transport Processes*. Cambridge University Press, 2007.

[30] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[32] R.J. Leveque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[33] E. E. Lewis and W. F. Miller. *Computational Methods of Neutron Transport*. Wiley, New Mexico, 1. edition, 1984.

[34] J. Lim, Y. Sivathanu, J. Ji, and J. Gore. Estimating scalars from spectral radiation measurements in a homogeneous hot gas layer. *Combustion and Flame*, 2004.

[35] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön. *Supervised Machine Learning: Lecture notes for the Statistical Machine Learning course*. Department of Information Technology, Uppsala University, 2019.

[36] Q. Liu and Y. Wu. Supervised learning. *In: Seel, N.M. (eds) Encyclopedia of the Sciences of Learning*, 2012.

[37] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[38] I. Livshin. *Artificial Neural Networks with Java: Tools for Building Neural Network Applications*. Chicago, IL, USA, 2019.

[39] I. Lux and L. Koblinger. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC Press, 1991.

[40] M.F. Modest. *Radiative Heat Transfer*. Academic Press, 3. edition, 2013.

[41] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[42] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. *ICML 2010*, pages 807–814, 2010.

[43] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *2nd International Conference on Computational Sciences and Technology - Jamshoro, Jamshoro, Pakistan*, pages 124–133, 2021.

[44] J. Patterson and A. Gibson. *Deep Learning: A practitioner's approach*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2017.

[45] K. L. Priddy and P. E. Keller. *Artificial Neural Networks: An Introduction*. SPIE PRESS, 2005.

[46] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag Berlin Heidelberg, 1996.

[47] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[48] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[49] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, 2010.

[50] S. A. Sarra. The method of characteristics with applications to conservation laws. *Journal of Online mathematics and its Applications*, 3:1–16, 2003.

[51] F. Scheben. *Iterative Methods for Criticality Computations in Neutron Transport Theory*. PhD thesis, University of Bath, 2011.

[52] S. Shalev-Shwartz and S. Ben-David. Understanding machine learning: From theory to algorithms. *Cambridge University Press*, 2014.

[53] R. Shanmugamani. *Deep Learning for Computer Vision: Expert techiques to train advanced neural networks using TensorFlow and keras*. Packt, Birmingham, UK, 2018.

[54] S. Sharma, S. Sharma, and A. Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 4:310–316, 2020.

[55] I.N. Silva, D.H. Spatti, R.A. Flauzino, L.H.B. Liboni, and A.S.F. Reis. *Artificial Neural Networks: A Practical Course*. Springer International Publishing Switzerland, 2017.

[56] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, and ... Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[57] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, and ... Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[58] W.M. Stacey. *Nuclear Reactor Physics*. Wiley-VCH, 2. edition, 2007.

[59] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[60] A. Vögler, S. Shelyag, M. Schüssler, F. Cattaneo, T. Emonet, and T. Linde. Simulations of magneto-convection in the solar photosphere: Equations, methods, and results of the muram code. *EDP sciences*, 2004.

[61] L.V. Wang and H. Wu. *Biomedical Optics: Principles and Imaging*. Wiley, New Jersey, 1. edition, 2007.

[62] B. Yegnanarayana. *Artificial Neural Networks*. Prentice-Hall of India Private Limited, 2005.

[63] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *CoRR*, abs/2106.11342, 2021.

[64] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li. Improving deep neural networks using softplus units. *International Joint Conference on Neural Networks (IJCNN)*, pages 1–4, 2015.