UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME OZANSKI FAURAUX DE MORAES

ESTRATÉGIAS PARA EVITAR SOLUÇÕES TRIVIAIS INDESEJADAS NA SOLUÇÃO NUMÉRICA VIA REDES NEURAIS DE EQUAÇÕES DIFERENCIAIS PARCIAIS HOMOGÊNEAS

CURITIBA

2024

GUILHERME OZANSKI FAURAUX DE MORAES

ESTRATÉGIAS PARA EVITAR SOLUÇÕES TRIVIAIS INDESEJADAS NA SOLUÇÃO NUMÉRICA VIA REDES NEURAIS DE EQUAÇÕES DIFERENCIAIS PARCIAIS HOMOGÊNEAS

Monografia apresentada como requisito parcial à conclusão do curso de graduação em Matemática Industrial pela Universidade Federal do Paraná.

Orientador: Prof. Thiago de Oliveira Quinelato, DSc.
Coorientador: Prof. Roberto Ribeiro Santos Junior, DSc.

CURITIBA

2024

RESUMO

Neste trabalho são estudadas estratégias para evitar a falha de propagação, limitação que pode ocorrer na utilização das Redes Neurais Informadas pela Física (*Physics-Informed Neural Networks* – PINNs) quando utilizada em Equações Diferenciais Parciais (EDPs) homogêneas. Após uma revisão sobre redes neurais e a formulação original das PINNs, são revisadas duas estratégias com a finalidade de evitar a falha de propagação: os pontos adaptativos e a função *gate*. Também é proposta a formulação de PINNs com métodos numéricos clássicos.

As estratégias mencionadas, assim como a formulação original de PINNs, foram testadas na equação do transporte com velocidade variável, onde a estratégia dos pontos adaptativos se mostrou mais eficiente, sendo a única estratégia capaz de aproximar a solução da EDP com precisão.

Palavras-chaves: Redes Neurais Informadas pela Física. Falha de propagação. Equações diferenciais homogêneas.

ABSTRACT

In this work, strategies to prevent the propagation failure are studied. Propagation failure is a limitation that can occur when using Physics-Informed Neural Networks (PINNs) with homogeneous Partial Differential Equations (PDEs). Following a review of neural networks and the original PINNs formulation, two strategies aimed at preventing the propagation failure are examined: adaptive points and the gate function. Additionally, the formulation of PINNs in conjunction with classical numerical methods is proposed.

The mentioned strategies, along with the original PINNs formulation, were tested on the transport equation with variable velocity. Among these, the adaptive points strategy proved to be the most efficient, being the only approach capable of approximating the PDE solution with precision.

Key-words: Physics-Informed Neural Networks. Propagation failure. Homogeneous differential equations.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O problema de encontrar soluções para equações diferenciais parciais (EDPs) tem grande importância para as mais diversas áreas do conhecimento, por exemplo, dinâmica dos fluidos, geofísica, modelagem do clima, entre tantas outras. Encontrar soluções analíticas para esses problemas é uma tarefa bastante complicada, por vezes até impossível. Por isso, procuramos encontrar soluções numéricas para essas equações.

O estudo de soluções numéricas para equações diferenciais resulta em diversos métodos numéricos. Um dos métodos mais comuns é o método de diferenças finitas, que consiste em aproximar as derivadas da equação diferencial por séries de Taylor truncadas, fornecendo a aproximação em uma malha de pontos no domínio. Esquemas desse tipo enfrentam certas dificuldades e limitações, motivando o estudo de outros métodos numéricos.

Raissi *et al.* (2017) propuseram um método numérico para a aproximação da solução de equações diferenciais via redes neurais, que ficou conhecido como *Physics Informed Neural Networks* (PINNs). Essa nova aplicação de redes neurais mostrou-se promissora, resolvendo equações importantes com precisão, como a Equação de Burgers com viscosidade e a Equação de Schrödinger. Desde então, essa abordagem tem sido aplicada a várias equações, com propostas para aprimorar as aproximações via PINNs (Fraces; Tchelepi, 2021; Diab; Kobaisi, 2021; Jacot *et al.*, 2018).

Um problema recorrente na utilização das PINNs para a solução numérica de EDPs homogêneas é o surgimento de soluções triviais, mesmo quando as condições iniciais e de contorno não são nulas: ocasionalmente, a aproximação construída por uma rede neural pode convergir erroneamente para a solução trivial em uma parte do domínio da solução.

Daw *et al.* (2022) apresentaram duas ideias com o propósito de evitar o surgimento espúrio das soluções triviais durante a aproximação via PINNs. A primeira é baseada na escolha de pontos de colocação adaptativos, inspirada nos algoritmos de malha adaptativa de métodos de elementos finitos. Motivado pelos métodos de diferenças finitas, o segundo algoritmo consiste na utilização de uma função, chamada de função *gate*, com a finalidade de propagar o dado inicial de uma equação diferencial ao longo do domínio. Neste trabalho de conclusão de curso, faremos uma revisão desses algoritmos.

Adicionalmente, será proposta uma outra forma de evitar soluções nulas indesejadas. Essa formulação consiste em unir métodos clássicos para soluções de EDP

(como esquemas de diferenças finitas) com as PINNs, a fim de obter soluções via redes neurais mais precisas e evitar as soluções nulas não desejadas.

Neste trabalho a apresentação dos métodos adicionais em PINNs está dividida em quatro capítulos. No primeiro capítulo, será feita uma revisão sobre redes neurais, conceitos básicos desse tópico e como as redes neurais são utilizadas. No capítulo seguinte, é feita uma descrição do método das PINNs: como definir um problema de otimização com base em uma EDP e posteriormente utilizar redes neurais para resolver esse problema. Nesse capítulo também é aplicado o método original das PINNs para a equação do transporte, expondo a falha de propagação. Seguindo para o terceiro capítulo, é realizada a apresentação das duas abordagens adicionais ao método das PINNs, com o objetivo de resolver o problema da falha de propagação. Continuando no terceiro capítulo, é proposta uma formulação com métodos numéricos clássicos para a solução de EDPs unida com as PINNs. Por fim, no último capítulo são feitas as conclusões e considerações finais sobre o trabalho.

Os códigos desenvolvidos para este trabalho estão disponíveis nos repositórios do GitHub https://github.com/LabFluid/NeuralPDE.jl e https://github.com/LabFluid/SHOPDEPINN.

1.1 OBJETIVOS GERAIS E OBJETIVOS ESPECÍFICOS

O objetivo deste trabalho é estudar algoritmos para evitar a convergência errônea para soluções nulas durante a aproximação de EDPs homogêneas via PINNs. Como objetivos específicos, destacam-se o estudo da escolha adaptativa de pontos de colocação, a utilização de funções do tipo *gate* e a combinação do métodos de diferenças finitas e as PINNs.

1.2 JUSTIFICATIVA

Como mencionado, resolver uma equação diferencial de forma analítica não é uma tarefa trivial. A procura de formas mais simples e rápidas de resolver essas equações levou ao estudo de métodos numéricos para soluções de EDPs.

As PINNs vêm ganhando destaque nessa linha de pesquisa pelo seu bom desempenho na aproximação de soluções de diferentes EDPs. No entanto, as PINNs apresentam problemas na aproximação de solução para uma EDP homogênea, como a convergência indesejada para a solução nula.

Tendo em vista as dificuldade enfrentadas pelas PINNs, este trabalho revisa e propõe técnicas para evitar a aproximação de soluções triviais não desejadas. Essas contribuições têm como objetivo aprimorar o desempenho das PINNs na aproximação de soluções para equações diferenciais homogêneas.

1.3 REFERENCIAL TEÓRICO

Os principais trabalhos utilizados como referência para a escrita deste trabalho de conclusão de curso foram os artigos Raissi *et al.* (2017) e Daw *et al.* (2022). O primeiro trabalho apresenta a formulação básica de PINNs, enquanto o segundo trabalho trata do estudo de algoritmos adicionais no método das PINNs para uma aproximação mais precisa em equações diferenciais homogêneas.

Como ferramentas computacionais foram utilizados os pacotes NeuralPDE (Zubov *et al.*, 2021), Lux (Pal, 2022), Optimization (OPTIMIZATION.JL, 2024), ModelingToolkit (Ma *et al.*, 2021), MKL (MKL.JL., 2024) e LinearAlgebra (LINEARALGEBRA.JL, 2024), além de uma implementação própria, utilizando a linguagem de programação Julia (Bezanson *et al.*, 2017).

2 REDES NEURAIS ARTIFICIAIS

Neste capítulo será feita uma breve descrição sobre as redes neurais que foram utilizadas neste trabalho. Para mais informações sobre redes neurais, consulte Bramburger (2024), Carvalho (2024) e Mochinski (2023).

O primeiro modelo de redes neurais artificiais foi apresentado por Warren McCulloch e Walter Pitts em 1943, na tentativa de descrever o comportamento de neurônios biológicos com um modelo matemático utilizando lógica proposicional. Posteriormente, as redes neurais foram aprimoradas para modelos matemáticos mais completos, com a intenção de aproximar soluções de problemas mais complexos.

Uma rede neural é uma função construída a partir da composição de funções lineares e não-lineares. Neste trabalho, uma rede neural $\hat{u}$ calculada nos pontos $(x,t) \in \mathbb{R}^2$ com d camadas ocultas é escrita como:

$$\hat{u}(x,t) = W_{d+1}\phi_d(W_d \cdots \phi_1(W_1\phi_0(W_0[x,t]^T + b_0) + b_1) \cdots + b_d) + b_{d+1}, \tag{2.1}$$

onde as funções ϕ_i, $i = \{0,1,\ldots,d\}$ são denominadas funções de ativação e devem ser não-lineares. As matrizes e vetores W_j e b_j, $j = \{0,1,\ldots,d,d+1\}$ são chamados de matrizes de pesos e vieses da rede neural, respectivamente.

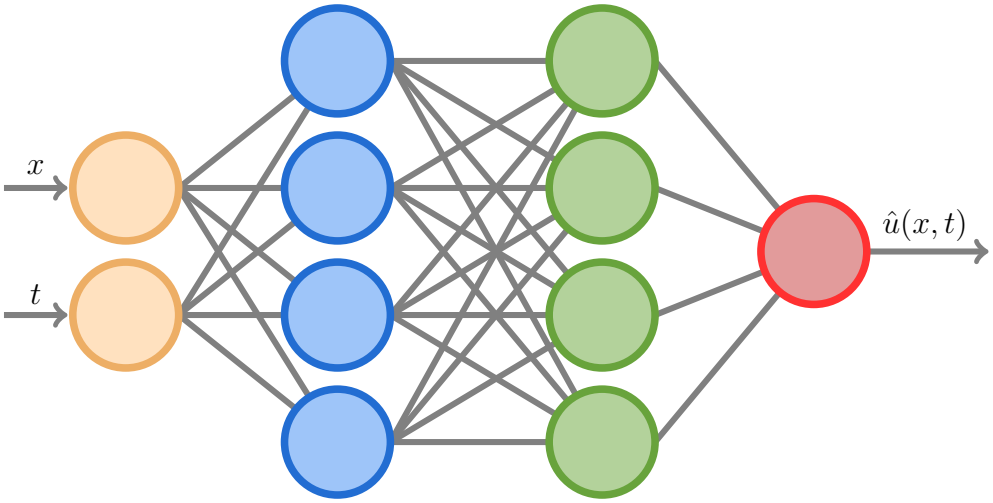

FIGURA 1 – Ilustração de uma rede neural.

A Figura 1 ilustra uma rede neural. Nesse exemplo a rede neural tem quatro camadas, sua primeira camada (camada de entrada) tem dois neurônios, as duas camadas seguintes (camadas ocultas) têm quatro neurônios em cada camada e, por fim, uma última camada (camada de saída) tem um neurônio.

Redes neurais com dois neurônios de entrada e um de saída serão amplamente utilizadas neste trabalho, visto que o objetivo da aplicação das PINNs neste trabalho é

aproximar funções reais de duas variáveis reais.

Pela formulação de uma rede neural apresentada, fica evidente que o valor de uma rede neural depende diretamente dos valores dos pesos e vieses. Esses parâmetros são ajustados com a utilização de um método de otimização, com o intuito de minimizar uma função, denominada função custo ou função de perda. O processo de otimização dessa função é conhecido como treinamento da rede neural, o número de épocas é a quantidade de vezes que todo o conjunto de treinamento é apresentado à rede neural e a arquitetura de uma rede neural é a disposição dos seus neurônios, camadas e funções de ativação.

Neste trabalho, a maioria dos testes numéricos foram feitos com os parâmetros listados na Tabela 1. Quando algum parâmetro diferente for utilizado, isso será devidamente mencionado.

Parâmetro	Valor
Função de Ativação	$\tanh$
Camadas	10
Neurônios por camada	10
Otimizador	Adam
Pesos e vieses iniciais	Glorot uniform
Taxa de aprendizagem	10^{-3}
Épocas	10 000

TABELA 1 – Parâmetros das redes neurais utilizadas neste trabalho.

3 REDES NEURAIS INFORMADAS PELA FÍSICA

Neste capítulo será mostrado como podemos utilizar as redes neurais para resolver um problema modelado por EDPs. O procedimento aqui apresentado é inspirado no trabalho de Raissi *et al.* (2017).

Como já mencionado, para trabalhar com uma rede neural é necessária a definição de uma função custo, onde então é aplicado um método de otimização para determinar os pesos e vieses da rede neural a fim de obter a aproximação, dada por uma rede neural, da solução de um problema, no caso desse capítulo, uma EDP.

Geralmente, a função custo é dada pelo quadrado da diferença entre o valor calculado pela rede neural e o valor correto. No caso das PINNs, usar uma função custo dessa forma seria ineficaz, pois se já tivermos a solução da equação diferencial, não é necessário utilizar um método numérico para encontrar sua aproximação. Portanto, mostraremos como definir uma função custo que possa ser usada para aproximar a solução de uma EDP.

Seja o problema de valor inicial:

Encontrar a função $u : \mathbb{R} \times [0, \infty) \to \mathbb{R}$ que satisfaça a EDP homogênea

$$\begin{cases} \mathcal{D}u = 0, & (x,t) \in \mathbb{R} \times [0,\infty); \\ u(x,0) = f(x), & x \in \mathbb{R}, \end{cases} \tag{3.1}$$

onde $\mathcal{D}$ é um operador diferencial e f é o valor inicial da EDP.

Definimos um domínio computacional para trabalharmos com esse problema de valor inicial. Seja o domínio $\Omega \subset \mathbb{R} \times [0, \infty)$ limitado, dado por $[a, b] \times [0, T]$ com $a < b$ e $0 < T$. Tomando como referência o problema (3.1) e o domínio Ω, podemos definir os problemas de minimização:

Problema 1. Operador diferencial:

$$\min |\mathcal{D}u|, \quad (x,t) \in \Omega. \tag{3.2}$$

Problema 2. Condição inicial:

$$\min |u(x,0) - f(x)|, \quad x \in [a,b]. \tag{3.3}$$

Queremos encontrar uma função que satisfaça as duas condições ao mesmo tempo. Será proposto uma função de perda para treinar uma rede neural que minimize os problemas simultaneamente.

Função de perda 1. Operador diferencial:

$$EQM_{\mathcal{D}} = \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} |\mathcal{D}\hat{u}(x_{\mathcal{D}}^i, t_{\mathcal{D}}^i)|^2. \tag{3.4}$$

Função de perda 2. Condição inicial:

$$EQM_I = \frac{1}{N_I} \sum_{i=1}^{N_I} |\hat{u}(x_I^i, 0) - f(x_I^i)|^2, \tag{3.5}$$

onde $\hat{u}$ é a solução numérica via rede neural e os conjuntos $\Omega_{\mathcal{D}} = \{(x_{\mathcal{D}}^i, t_{\mathcal{D}}^i)\}_{i=1}^{N_{\mathcal{D}}}$ e $\Omega_I = \{x_I^i\}_{i=1}^{N_I}$ são formados por pontos amostrados aleatoriamente, conforme uma distribuição uniforme, nos subconjuntos Ω e $[a, b]$, respectivamente. Os conjuntos $\Omega_{\mathcal{D}}$ e Ω_I são tradicionalmente mantidos fixos durante todo o treinamento da PINN e são nomeados como pontos de colocação no domínio e pontos de colocação iniciais.

Assim, definimos a função de custo da rede neural, EQM, como a soma das funções $EQM_{\mathcal{D}}$ e EQM_I:

$$EQM = EQM_{\mathcal{D}} + EQM_I \tag{3.6}$$

e queremos resolver o problema de otimização:

$$\min_{W,b} EQM, \tag{3.7}$$

onde W, b são os pesos e vieses da rede neural, como descrito em (2.1).

Tendo em vista que a solução da EDP é uma função pelo menos contínua e a rede neural é uma função dada pela expressão (2.1), então sob as hipóteses do Teorema da Aproximação Universal (Hornik *et al.*, 1989) é possível utilizar uma rede neural para resolver uma EDP numericamente.

A seguir aplicaremos essa formulação na equação do transporte. Para isso, alguns parâmetros serão mantidos iguais durante todo o trabalho. Na Tabela 2 são listados os valores desses parâmetros.

Parâmetro	Valor
$N_{\mathcal{D}}$	2 000
N_I	1 000
Intervalo espacial: $[a, b]$	[0, 5]
Tempo máximo: T	6,4

TABELA 2 – Parâmetros das PINNs que aproximam a solução da equação do transporte.

Para comparar a aproximação numérica dada pelos métodos utilizados com a solução analítica do problema, serão utilizados os erros relativos descritos a seguir.

Inicialmente, definiremos uma malha de pontos igualmente espaços $\Omega_M \subset [a,b] \times [0,T]$, onde cada elemento é um ponto (x_p, t_q) escrito por:

$$\begin{aligned} x_p &= a + ph,\ p \in \{0,1,\ldots,P\}, \\ t_q &= qk,\ q \in \{0,1,\ldots,Q\}, \end{aligned} \tag{3.8}$$

onde $h = (b-a)/P$ e $k = T/Q$ definem o espaçamento da malha. Note que $x_0 = a$, $x_P = b$, $t_0 = 0$ e $t_Q = T$.

Definimos a matriz U associada à função $u(x,t)$ solução de uma EDP, com entradas $U[q,p] = u(x_p,t_q),\ \forall (x_p,t_q) \in \Omega_M$. De maneira análoga, a matriz $\hat{U}$ é associada à função $\hat{u}(x,t)$ dada por uma rede neural.

Com isso, definimos o erro relativo da função $\hat{u}(x,t)$ em comparação à função $u(x,t)$ como

$$\frac{\|U - \hat{U}\|_F}{\|U\|_F}, \tag{3.9}$$

onde $\|.\|_F$ é a norma de Frobenius, dada por

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2},\ \forall A \in \mathbb{R}^{m\times n}. \tag{3.10}$$

O erro no instante $t = 0$ será definido de forma similar. Definimos o vetor F associado à função $f(x)$, com entradas dadas por $F[p] = f(x_p),\ \forall x_p,\ p \in \{0,1,\ldots,P\}$. De forma semelhante o vetor $\hat{U}_I$ é associado à função $\hat{u}(x_p,0)$.

Por fim, definimos o erro relativo no instante $t = 0$ da função $\hat{u}(x,0)$ em comparação à função $f(x)$ como

$$\frac{\|F - \hat{U}_I\|_2}{\|F\|_2}, \tag{3.11}$$

onde $\|.\|_2$ é a norma euclidiana, dada por

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2},\ \forall v \in \mathbb{R}^n. \tag{3.12}$$

3.1 UTILIZANDO PINNS NA EQUAÇÃO DO TRANSPORTE

Nesta seção será analisado o desempenho da formulação das PINNs, apresentada anteriormente, na equação do transporte com velocidade variável:

$$\begin{cases} u_t + c(x)u_x = 0, & (x,t) \in \mathbb{R} \times [0,\infty), \\ u(x,0) = f(x), & x \in \mathbb{R}. \end{cases} \tag{3.13}$$

Fisicamente, essa equação modela a dinâmica de uma onda, com perfil inicial dado pela função $f(x) = e^{-100(x-1)^2}$, que se propaga com velocidade variável no espaço $c(x) = 0{,}2 + \sin^2(x-1)$. A solução analítica desse problema é encontrada em Botelho (2023). As Figuras 2 e 3 expõem o comportamento de transporte da equação mencionado anteriormente.

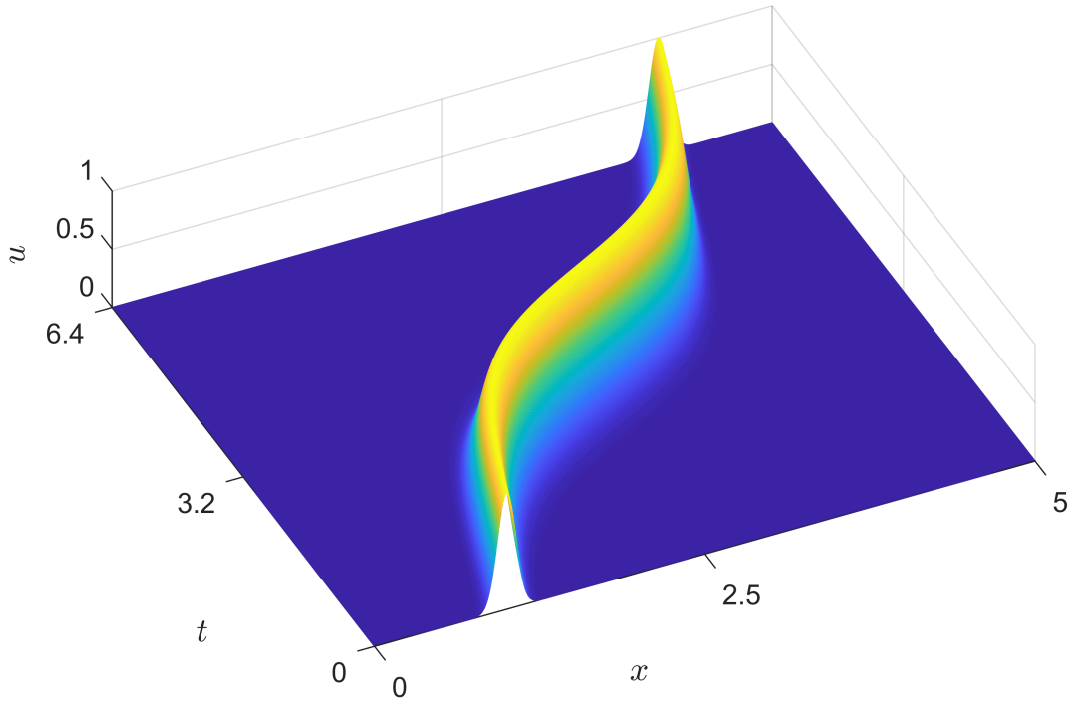

FIGURA 2 – Gráfico da solução analítica da equação (3.13).

Utilizando os parâmetros definidos nas Tabelas 1 e 2, aplicamos o método das PINNs na EDP (3.13). Ao final do treinamento da rede neural a função custo (3.6) obteve o valor de $1{,}1 \cdot 10^{-6}$. A comparação entre o resultado, ilustrado na Figura 5, com a solução exata do problema, na Figura 3, evidencia que a aproximação obtida pela PINN não é satisfatória, apesar de ter uma boa aproximação no instante $t = 0$, com erro relativo de $6{,}3 \cdot 10^{-3}$, como exposto na Figura 4. Esse comportamento é conhecido como falha de propagação.

A falha de propagação ocorre quando a solução numérica via PINN consegue aproximar com precisão o dado inicial da EDP, mas falha em aproximar a solução em outras regiões do domínio. Mesmo que em algumas regiões do domínio a solução numérica represente bem a física do problema, em regiões específicas do domínio a aproximação fornecida pela PINN tende à solução trivial. Isso indica que a rede neural falhou em aprender a solução completa dentro do domínio de interesse.

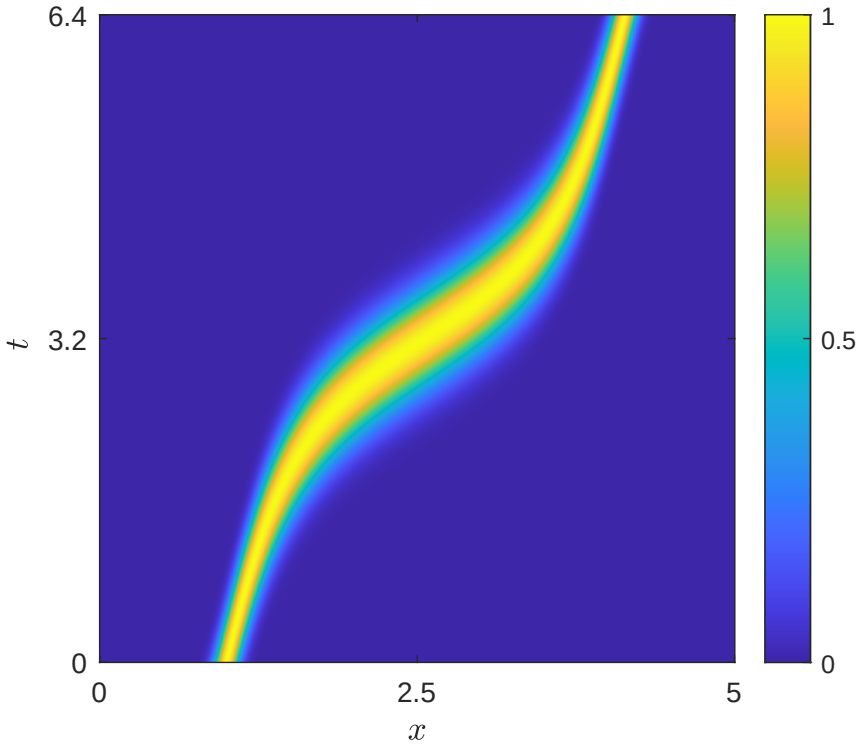

FIGURA 3 – Mapa de calor da solução analítica da equação (3.13).

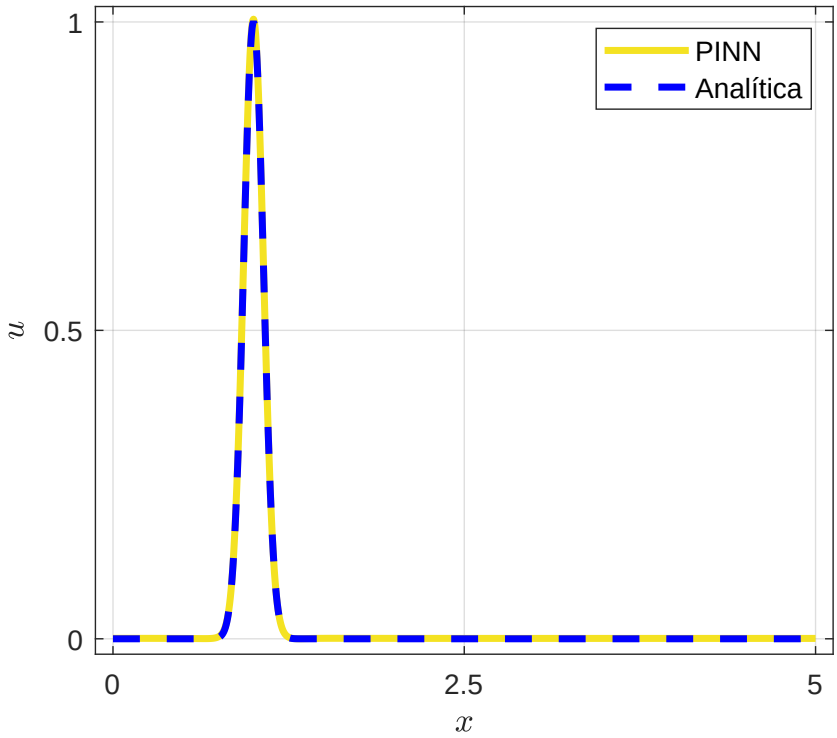

FIGURA 4 – Comparação entre a solução numérica via PINN e a solução analítica da equação (3.13) no instante $t = 0$.

Motivado pelas dificuldades observadas na utilização das PINNs na equação do transporte com velocidade variável, o capítulo seguinte tem como objetivo apresentar

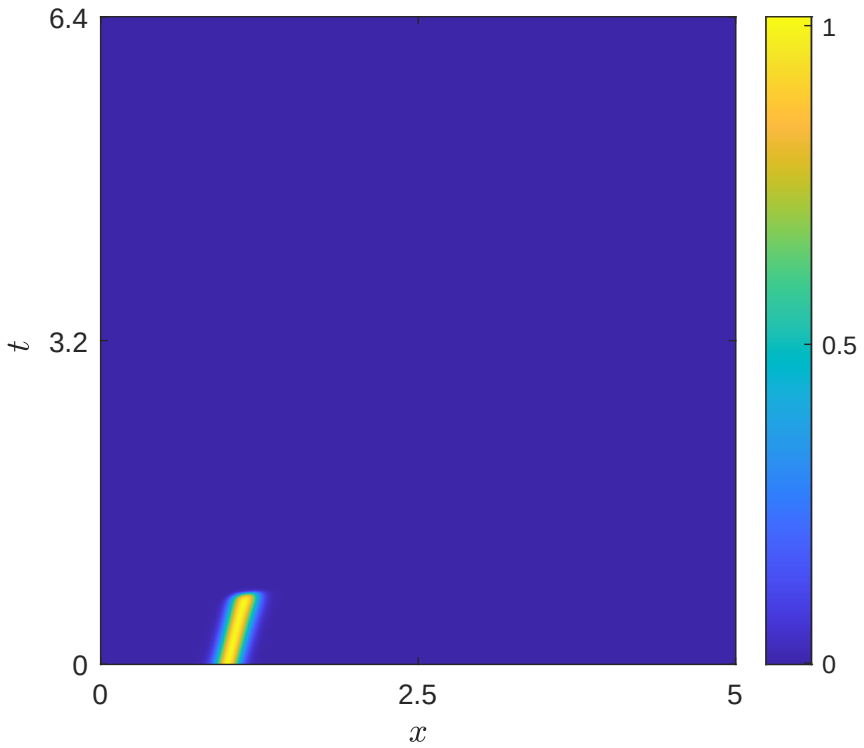

FIGURA 5 – Mapa de calor da solução numérica via PINN da equação (3.13).

e explorar algoritmos adicionais à formulação original das PINNs, visando tanto evitar a convergência para soluções nulas, quanto melhorar a aproximação das PINNs em EDP homogêneas, focando especificamente na equação do transporte apresentada nesta sessão.

4 ESTRATÉGIAS PARA EVITAR A FALHA DE PROPAGAÇÃO

A falha de propagação pode ocorrer quando resolvemos uma equação diferencial homogênea via redes neurais. Como foi exposto na formulação do método das PINNs, o problema de valor inicial (3.1) é separado em dois problemas de otimização: (3.2) e (3.3). Mesmo que posteriormente esses problemas sejam unidos, como é feito em (3.6), uma limitação dessa formulação se mantém: a solução nula é uma solução para o problema (3.2) mas não para o problema (3.7) se $f(x) \neq 0$ para algum x.

Observe que se $u(x,t) = 0$ em uma região do domínio, então $\mathcal{D}u = 0$ para todo ponto no interior dessa região. Isso implica que no processo de minimização do problema (3.7), a aproximação dada por uma rede neural pode ser nula em um subconjunto do domínio Ω, o que não é penalizado pelo processo de minimização.

Em busca de evitar soluções numéricas nulas não desejadas, Daw *et al.* (2022) propõem duas abordagens adicionais às PINNs: uma que chamaremos de pontos adaptativos e outra que será denominada função *gate*. Nas seções a seguir será feita uma descrição desses dois métodos, assim como experimentos numéricos na equação do transporte. Posteriormente, será apresentada uma terceira abordagem à formulação original das PINNs, integrando métodos numéricos clássicos com a estrutura das PINNs.

4.1 PONTOS ADAPTATIVOS

A abordagem com pontos adaptativos busca desconsiderar regiões do domínio onde a função (3.6) está próxima de zero, priorizando regiões do domínio onde a função de perda apresenta valores significativos, isto é, onde a função está alta. Daw *et al.* (2022) destacam que a ideia de utilizar pontos adaptativos está relacionada com as malhas adaptativas de Métodos de Elementos Finitos (Zienkiewicz; Taylor, 2005).

Para isso, vamos avaliar a função de perda da PINN em cada ponto do conjunto $\Omega_{\mathcal{D}}$ e substituir os pontos onde a função $EQM_{\mathcal{D}}$ obteve os menores valores por novos pontos amostrados conforme uma distribuição uniforme. Esse processo tem o propósito de enfatizar regiões onde a função de perda apresenta valores elevados, possibilitando melhorar a aproximação da PINN nessas regiões, assim como evitar regiões onde a rede neural pode ter uma solução constante nula.

O algoritmo de pontos adaptativos é descrito conforme os seguintes passos:

1. **Definições Iniciais**: Defina o conjunto Ω_E, formado por N_E pontos distribuídos aleatoriamente no domínio de uma função de perda $E(x,t)$.

2. **Avaliação da perda**: Calcule os valores da função de perda $E(x,t)$ em cada ponto de Ω_E. Esses valores indicam o desempenho da solução numérica em diferentes regiões do domínio.

3. **Remoção de pontos**: Selecione $N_r < N_E$ pontos que serão removidos do conjunto Ω_E. Os pontos removidos correspondem àqueles com os menores valores de $E(x,t)$, ou seja, pontos em regiões onde a solução já é satisfatória.

4. **Redistribuição de pontos**: Distribua aleatoriamente N_r novos pontos no domínio. Esses pontos são adicionados ao conjunto restante do passo anterior, formando um novo conjunto atualizado Ω_E.

5. **Iteração do algoritmo**: Repita os passos 2, 3 e 4 pelo número de épocas desejado.

Vejamos um exemplo de como os pontos adaptativos se comportam com uma função fixa $E : [-2,2] \times [-2,2] \to \mathbb{R}$:

$$E(x,t) = e^{-(x^2+t^2-1)^2}. \tag{4.1}$$

O gráfico e o mapa de calor dessa função são mostrados nas Figuras **??** e 7, respectivamente. É possível notar que seu maior valor é atingido quando os pontos (x,t) pertencem à circunferência de raio 1 centrada em $(0,0)$.

Seguindo os passos do algoritmo, inicialmente definimos o conjunto Ω_E com $N_E = 500$ pontos escolhidos aleatoriamente, conforme uma distribuição uniforme, como é apresentado na Figura 8 e a cada iteração serão substituídos $N_r = 200$ pontos. Aplicando o algoritmo de pontos adaptativos por cinco iterações, obtém-se a distribuição de pontos da Figura 9. Com mais 45 iterações do algoritmo, fica ainda mais evidente a concentração dos pontos de colocação, como é observado na Figura 10.

Analisando as Figuras 9 e 10, é possível notar que quanto maior o número de iterações, maior é a concentração de pontos na circunferência de raio 1 centrada em $(0,0)$. Comparando esse comportamento com o mapa de calor da função (4.1) na Figura 7, conclui-se que os pontos estão sendo agrupados onde a função E tem maior valor.

Vamos utilizar o algoritmo de pontos adaptativos em conjunto com as PINNs na equação (3.13). Utilizaremos os parâmetros listados nas Tabelas 1 e 2. Serão substituídos 30% dos pontos do conjunto $\Omega_{\mathcal{D}}$, equivalente a 600 pontos, a cada 1000 iterações do método de otimização.

O método das PINNs com pontos adaptativos foi bem-sucedido na aproximação da solução da equação (3.13). O erro relativo no domínio (3.9) é de $1{,}6 \cdot 10^{-2}$, enquanto

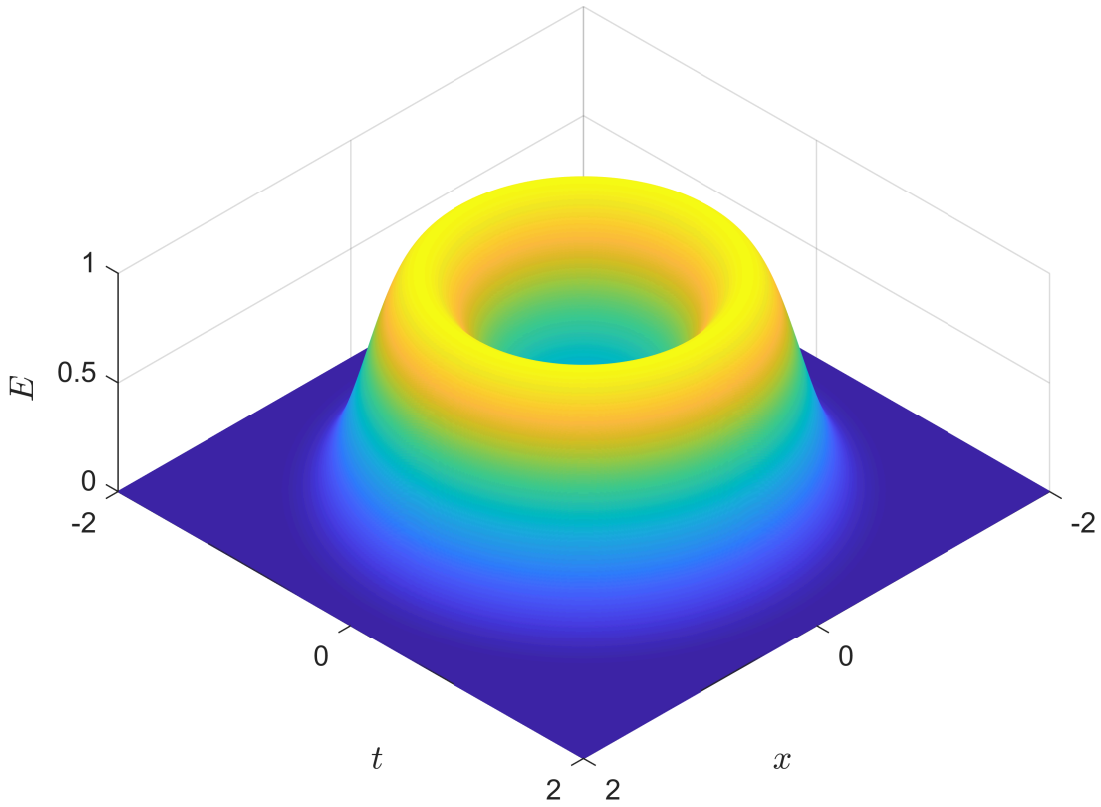

FIGURA 6 – Gráfico da função $E(x,t)$.

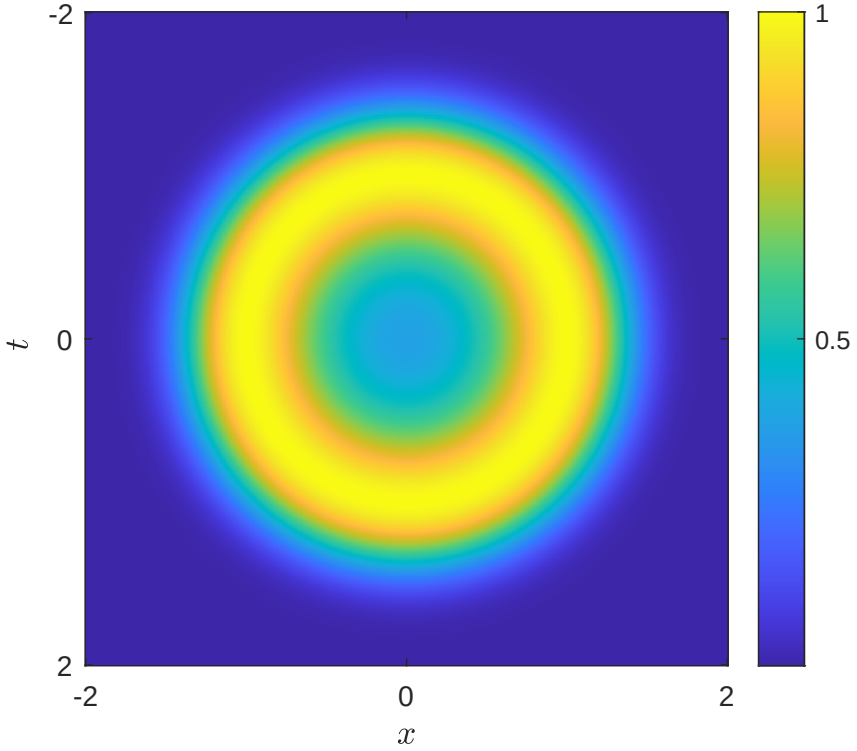

FIGURA 7 – Mapa de calor da função $E(x,t)$.

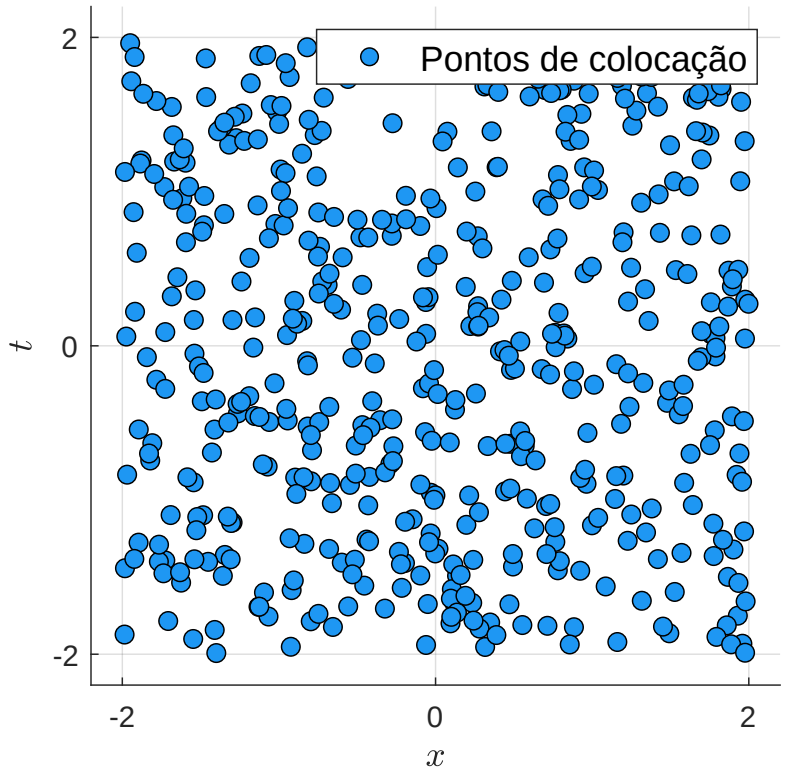

FIGURA 8 – Pontos do conjunto Ω_E distribuídos aleatoriamente.

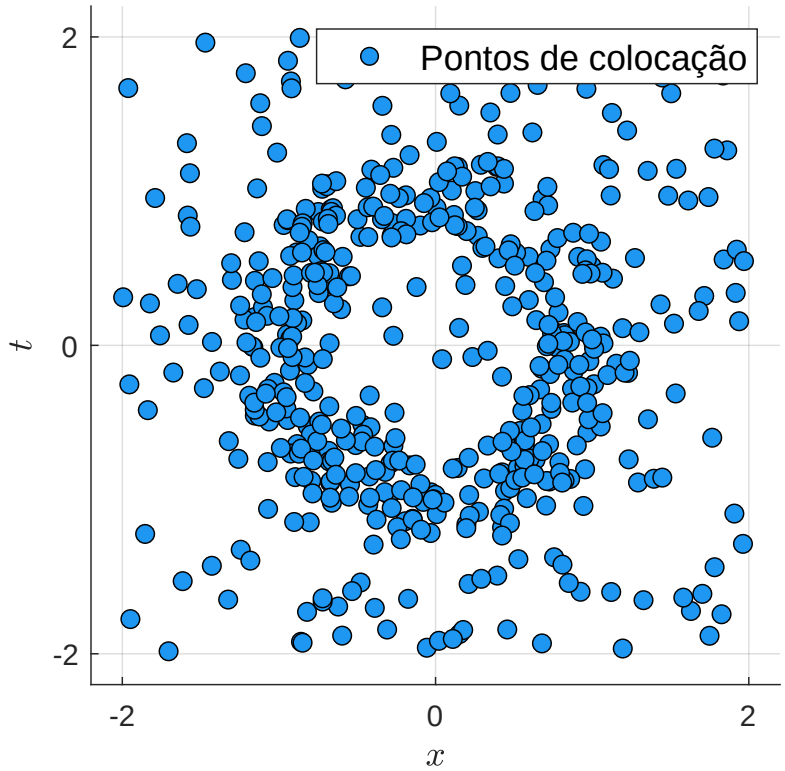

FIGURA 9 – Pontos do conjunto Ω_E após 5 iterações do algoritmo de pontos adaptativos.

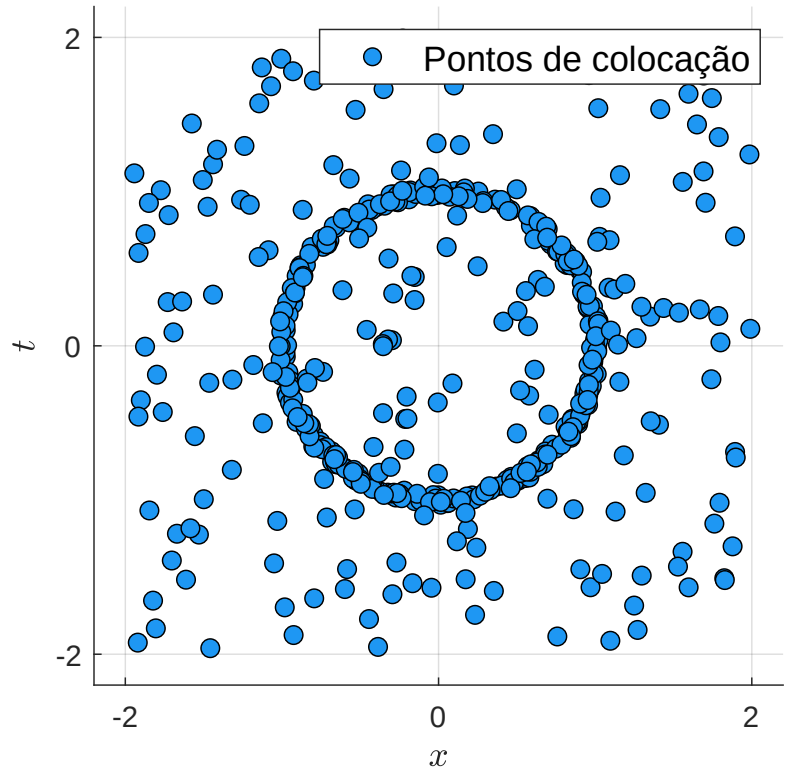

FIGURA 10 – Pontos do conjunto Ω_E após 50 iterações do algoritmo de pontos adaptativos.

o erro relativo no instante inicial (3.11) é $8{,}3 \cdot 10^{-3}$. O resultado é apresentado nas Figuras 12 e 11.

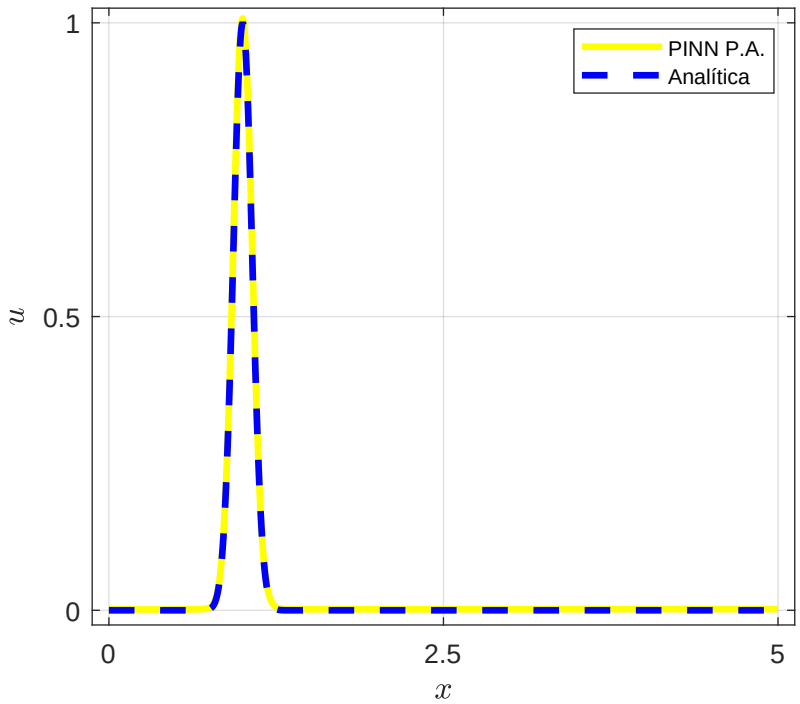

FIGURA 11 – Comparação entre a solução numérica via PINN com pontos adaptativos e a solução analítica da equação (3.13) no instante $t = 0$.

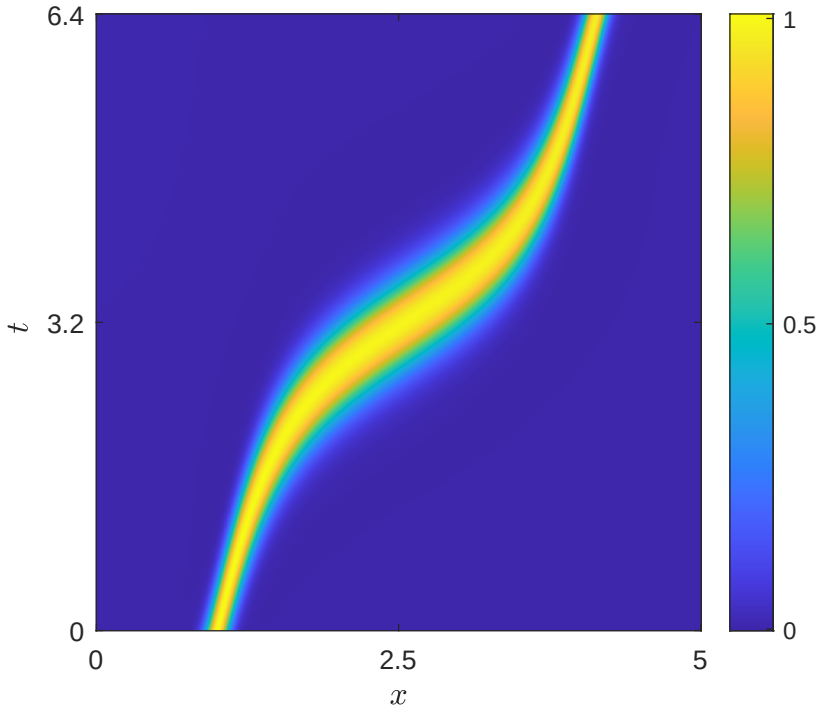

FIGURA 12 – Mapa de calor da solução numérica via PINN com pontos adaptativos da equação (3.13).

Uma pergunta natural a se fazer nesse caso é: qual foi a distribuição de pontos de colocação no final do treinamento da PINNs? Na Figura 13 são ilustrados os pontos do conjunto $\Omega_{\mathcal{D}}$ no início do treinamento, enquanto a Figura 14 exibe a organização dos pontos ao fim das iterações do método de otimização. Observa-se que os pontos tendem a se concentrar na região do plano onde a solução da equação (3.13) tem valores diferentes de zero, justamente a região em que o método original das PINNs teve dificuldade para aproximar a solução da EDP.

4.2 FUNÇÃO *GATE*

Outra ideia para evitar a falha de propagação, sugerida por Daw *et al.* (2022), consiste em alterar a função de perda da rede neural, multiplicando cada termo do somatório da função custo (3.4) por uma função, denominada de função *gate*, que varia entre zero e um. Essa alteração tem a intenção de propagar o dado inicial da EDP ao longo do domínio de interesse, semelhante à forma com que métodos de diferenças finitas constroem as aproximações para problemas de valor inicial.

O procedimento descrito acima tem como objetivo enfatizar regiões do domínio próximas ao instante $t = 0$, no inicio do treinamento, proporcionando que a PINN obtenha uma boa aproximação nessa região inicial. Em seguida, os parâmetros da função *gate* são alterados para priorizar uma maior região do domínio, possibilitando que a PINN não apenas aproxime corretamente os dados iniciais, mas também propague a

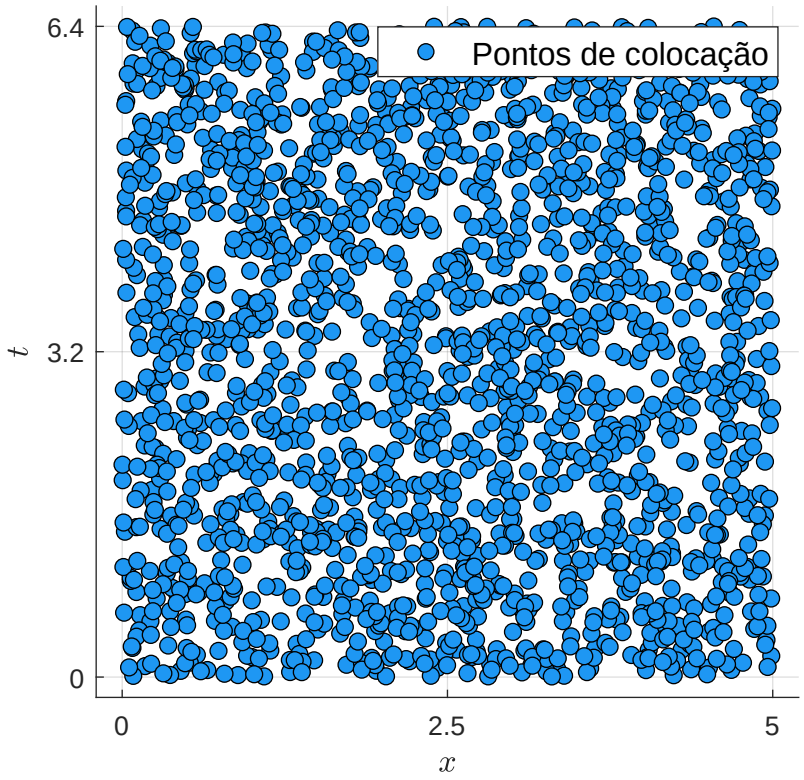

FIGURA 13 – Pontos de colocação no início do treinamento da PINN.

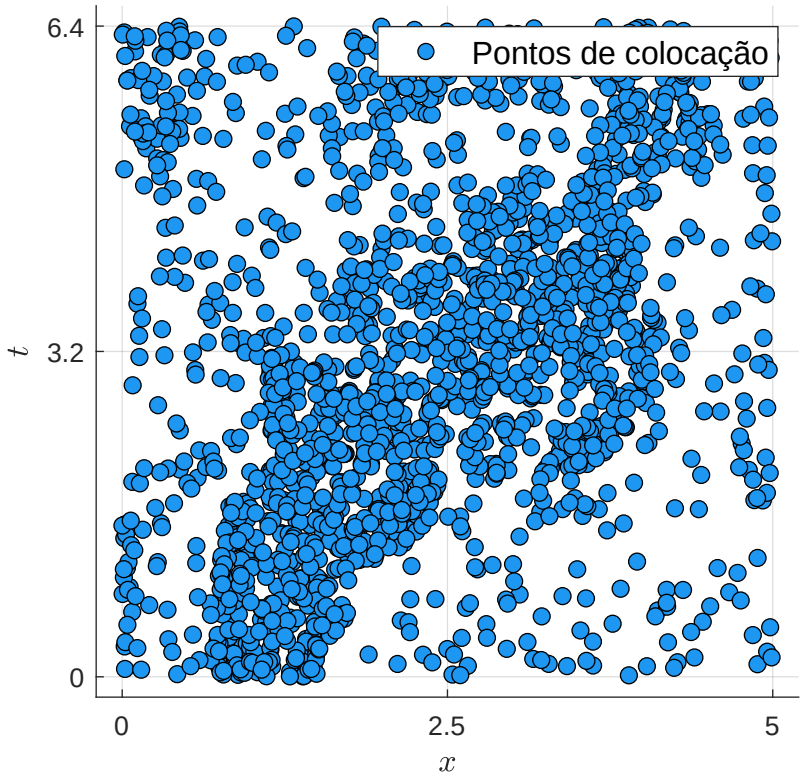

FIGURA 14 – Pontos de colocação no fim do treinamento da PINN.

informação do dado inicial de forma consistente ao longo do domínio.

A função *gate* é uma função parametrizada, escrita como $g : [0, 1] \to (0, 1)$:

$$g(\tau; \alpha, \theta) = \frac{1 - \tanh(\alpha(\tau - \theta))}{2}, \tag{4.2}$$

em que $\tau = t/T$, os parâmetros α e θ são números reais, onde o primeiro controla a inclinação da função *gate* e o segundo é responsável por transladar a função. As Figuras 15 e 16 ilustram como esses parâmetros alteram a função.

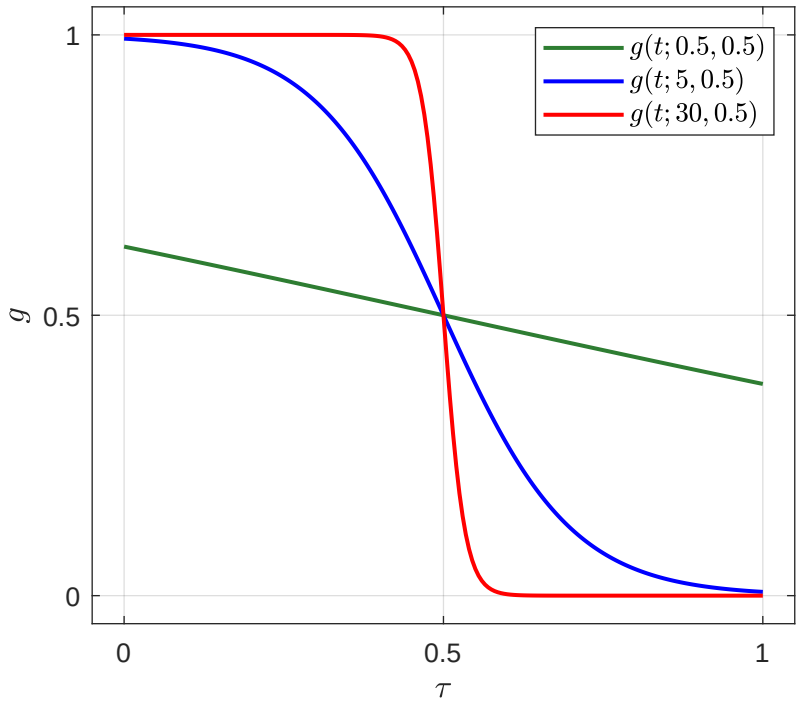

FIGURA 15 – Exemplos de funções *gate* alterando o parâmetro α.

A função *gate* será utilizada para alterar a função de perda das PINNs. Para isso, a expressão $EQM_{\mathcal{D}}$ será substituída na função custo (3.6) por

$$EQM_g = \frac{1}{N_{\mathcal{D}}} \sum_{i=1}^{N_{\mathcal{D}}} |\mathcal{D}\hat{u}(x^i_{\mathcal{D}}, t^i_{\mathcal{D}})|^2 g(\tau^i_{\mathcal{D}}; \alpha, \theta), \tag{4.3}$$

onde $\tau^i_{\mathcal{D}} = t^i_{\mathcal{D}}/T$. Dessa forma, as regiões do domínio onde a função *gate* atinge seu maior valor são priorizadas ao longo do treinamento das PINNs. O parâmetro θ é alterado com o objetivo de aumentar a região onde a função $g(\tau^i_{\mathcal{D}}; \alpha, \theta)$ tem valores próximos de um.

No trabalho de Daw *et al.* (2022), a atualização desse valor é feita com base no valor da função (4.3). Neste trabalho, será feita uma abordagem diferente: o valor de θ depende linearmente da quantidade de épocas utilizadas no treinamento de uma rede neural:

$$\theta(I) = \beta I + \gamma, \tag{4.4}$$

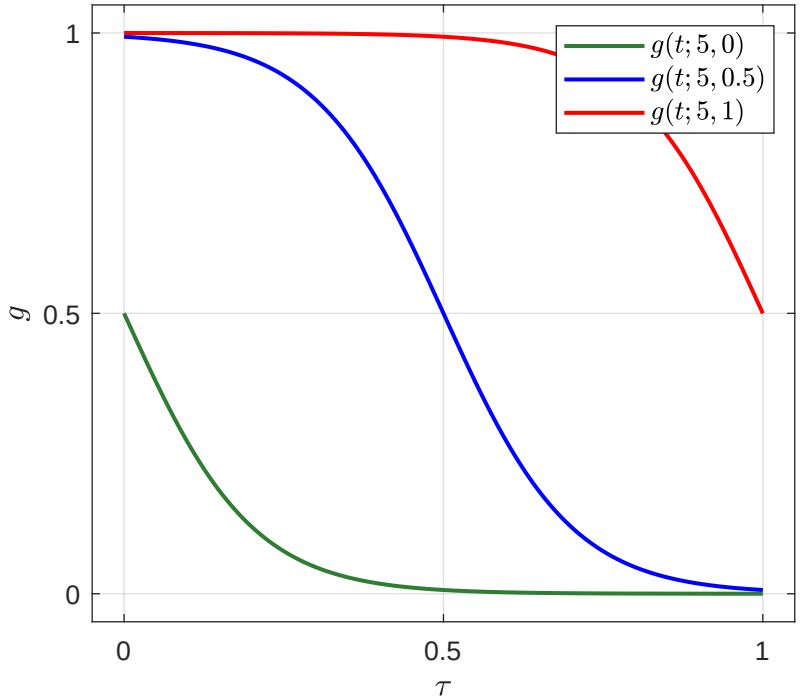

FIGURA 16 – Exemplos de funções *gate* alterando o parâmetro θ.

onde I é a razão entre a época atual e a quantidade total de épocas e β e γ são parâmetros a se determinar.

Vamos utilizar a formulação das PINNs com a função *gate* na equação (3.13). Para as redes neurais, serão usados os dados das Tabelas 1 e 2. Já para a função *gate*, foram testados diversos valores para os parâmetros α, β e γ. Apesar de gerarem resultados diferentes, todos mantiveram uma solução numérica com falha de propagação.

Os resultados ilustrados nas Figuras 18 e 17 foram obtidos com a escolha $\alpha = 5$, $\beta = 1.5$ e $\gamma = 0.25$. É possível observar que mesmo com a mudança na formulação da função de perda das PINNs, a falha de propagação se manteve, ou seja, o dado inicial da EDP está sendo bem aproximado, com erro relativo, definido pela expressão (3.11), de $2{,}9 \cdot 10^{-3}$, porém a solução numérica tende erroneamente à solução trivial em uma região considerável do domínio.

4.3 UNIÃO ENTRE PINNS E MÉTODOS CLÁSSICOS

Nesta seção será proposta uma terceira variação na formulação das PINNs, que consiste em utilizar métodos numéricos clássicos para facilitar seu treinamento, buscando soluções mais precisas e fugindo de soluções triviais indesejadas. Essa abordagem é composta por três etapas principais: inicialmente, uma solução numérica é obtida utilizando um método clássico adequado ao problema em questão; em seguida,

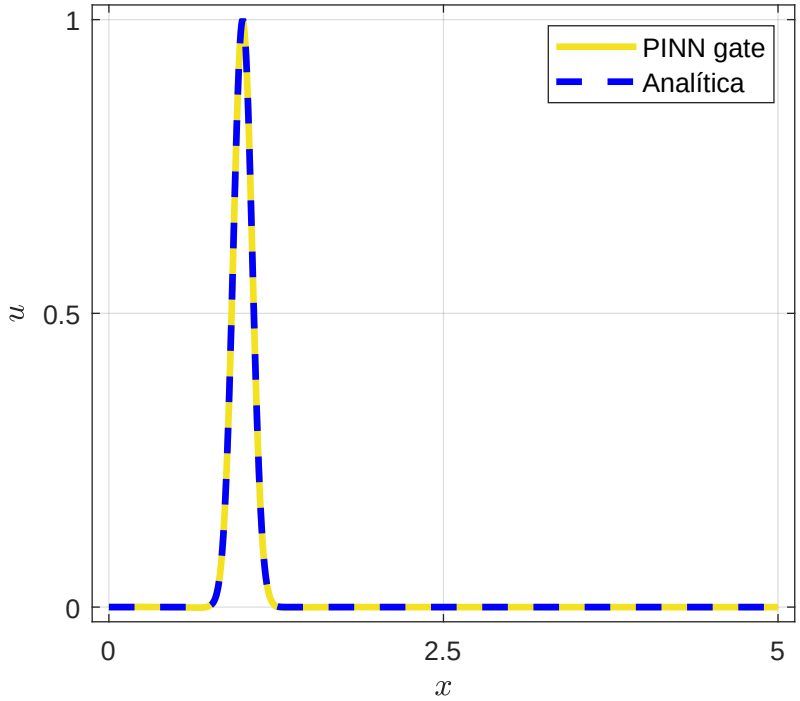

FIGURA 17 – Comparação entre a solução numérica via PINN com função *gate* e a solução analítica da equação (3.13) no instante $t = 0$.

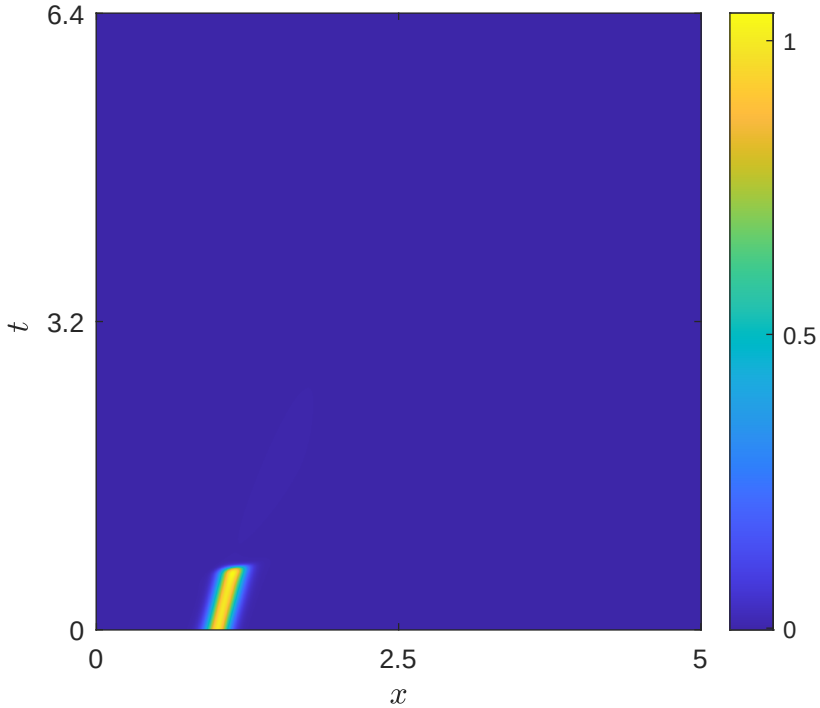

FIGURA 18 – Mapa de calor da solução numérica via PINN com função *gate* da equação (3.13).

uma rede neural é treinada com o objetivo de ajustar os dados gerados por esse método; por fim, a mesma rede neural é refinada utilizando a função de perda tradicional das PINNs.

Com essa formulação, espera-se que a rede neural possa aprimorar as soluções numéricas obtidas por métodos clássicos e também superar algumas de suas limitações. Por exemplo, no método de diferenças finitas, uma das principais limitações é a dependência de uma malha de pontos, o que dificulta a análise da solução em pontos fora da malha, além do cálculo das derivadas da aproximação obtida. A aplicação do método das PINNs com métodos clássicos permite que as soluções numéricas sejam representadas como funções contínuas, eliminando a necessidade de uma malha de pontos. Isso não apenas possibilita que a solução obtida pelas PINNs seja uma boa aproximação para a solução da EDP, mas também supera limitações dos métodos clássicos, como descritos anteriormente.

Espera-se que a rede neural seja capaz se ajustar aos dados fornecidos pelo método numérico clássico, garantindo que o treinamento da rede neural não enfrente problemas de convergência para soluções nulas indesejadas, quando utilizada a função de perda original das PINNs. As etapas dessa formulação serão detalhadas a seguir para uma EDP escrita de forma geral, como o problema de valor inicial (3.1).

A primeira etapa é utilizar um método numérico adequado para a EDP, obtendo uma solução numérica. Definimos o conjunto $C = \{((x_c^i, t_c^i), u_c(x_c^i, t_c^i))\}_{i=0}^{N_c}$ com a aproximação dada pelo método clássico, onde $u_c(x_c^i, t_c^i)$ é a aproximação fornecida pelo método numérico clássico no ponto (x_c^i, t_c^i) e N_c é a quantidade de dados gerados pelo método clássico.

Após o primeiro passo, será feito o treinamento da rede neural com o objetivo de se ajustar aos dados do conjunto C, obtido anteriormente. Para essa finalidade, é definida uma função de perda da forma:

$$EQM_C = \frac{1}{N_c} \sum_{i=0}^{N_c} |\hat{u}(x_c^i, t_c^i) - u_c(x_c^i, t_c^i)|^2. \tag{4.5}$$

Não existe evidência de que a rede neural treinada com essa função de perda seja uma boa aproximação para a solução do problema (3.1), mas espera-se que o treinamento da PINN, quando realizado com a rede neural ajustada aos dados, convirja mais facilmente para a solução do problema em comparação com a formulação clássica das PINNs.

Com o passo anterior concluído, a função de perda será alterada para a função (3.6). Em seguida, a rede neural ajustada ao conjunto C é treinada como uma PINN, da maneira descrita no capítulo 3.

Neste trabalho, será utilizado o método de diferenças finitas para gerar os dados iniciais, mais precisamente o esquema *upwind*. O método de diferenças finitas consistem em aproximar a solução da EDP por séries de Taylor truncadas, construindo uma solução numérica em uma malha de pontos do domínio. Para mais informações

sobre o esquema *upwind* para a equação (3.13) consulte Botelho (2023). Para uma visão geral do método de diferenças finitas, consulte Strikwerda (2004).

Inicialmente, será definida uma discretização do domínio de interesse, dada por $x_j = j\Delta x$, $j \in \{0, 1, \ldots, N_x\}$ e $t_n = n\Delta t$, $n \in \{0, 1, \ldots, N_t\}$, gerando a malha de pontos $M = \{(x_j, t_n)\}_{j,n=0,0}^{N_x,N_t}$.

A função $u(x,t)$, solução da equação (3.13), calculada nos pontos da malha M será escrita como $u_j^n = u(x_j, t_n)$, assim como suas derivadas parciais calculadas nos pontos da malha serão escritas como $[u_x]_j^n = u_x(x_j, t_n)$ e $[u_t]_j^n = u_t(x_j, t_n)$. A seguir, serão calculadas as séries de Taylor truncadas da função u, de forma regressiva no espaço e progressiva no tempo, com o objetivo de aproximar as derivadas parciais da equação do transporte:

$$\begin{aligned} u_{j-1}^n &= u_j^n - \Delta x[u_x]_j^n; \\ u_j^{n+1} &= u_j^n + \Delta t[u_t]_j^n. \end{aligned} \tag{4.6}$$

Reorganizando os termos, obtemos uma aproximação para as derivadas espacial e temporal, respectivamente:

$$\begin{aligned} [u_x]_j^n &= \frac{u_j^n - u_{j-1}^n}{\Delta x}; \\ [u_t]_j^n &= \frac{u_j^{n+1} - u_j^n}{\Delta t}. \end{aligned} \tag{4.7}$$

Escrevendo $c_j = c(x_j)$ e utilizando as aproximações de $[u_x]_j^n$ e $[u_t]_j^n$ na equação (3.13), obtemos o operador de diferenças da equação (3.13):

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c_j \frac{u_j^n - u_{j-1}^n}{\Delta x} = 0. \tag{4.8}$$

Por fim, os termos são reorganizados novamente, a fim de explicitar u_j^{n+1}, obtendo o esquema *upwind*:

$$u_j^{n+1} = u_j^n - \lambda_j(u_j^n - u_{j-1}^n), \tag{4.9}$$

onde $\lambda_j = c_j\Delta t/\Delta x$. É possível construir uma aproximação ao longo de um domínio temporal, iterando a expressão (4.9), iniciando com $u_j^0 = f_j = f(x_j)$.

Além disso, para utilizar o método *upwind*, também é necessário introduzir uma condição de bordo. Quando a função $c(x)$ é estritamente maior que zero a condição de bordo é aplicada na fronteira esquerda do domínio. Neste problema, foi utilizada a condição de bordo nula.

Para atender a condição CFL, o valor de λ_j deve ser limitado superiormente por 1. Sendo assim, os valores Δx e Δt devem ser escolhidos para atender essa condição:

$$\lambda_j \leq 1 \; \forall j \in \{0, 1, \ldots, N_x\} \Rightarrow \max_{j \in \{0,1,\ldots,N_x\}} \lambda_j \leq 1 \Rightarrow \Delta t \leq \frac{\Delta x}{\max_{x \in \mathbb{R}} c(x)} = \frac{\Delta x}{1.2}, \tag{4.10}$$

Dessa maneira, conseguimos obter o valor de Δt para qualquer valor de Δx, garantindo que a condição CFL seja sempre satisfeita.

Vamos aplicar o método das PINNs com métodos clássicos para a EDP (3.13). Inicialmente, usaremos o esquema *upwind* com $\Delta x = 10^{-2}$ e $\Delta t = \Delta x/1.2$ para gerar o conjunto de dados C.

O resultado do esquema *upwind* é ilustrado nas Figuras 20 e 19. É possível observar a dissipação da solução numérica oriunda de erros de aproximação do método numérico.

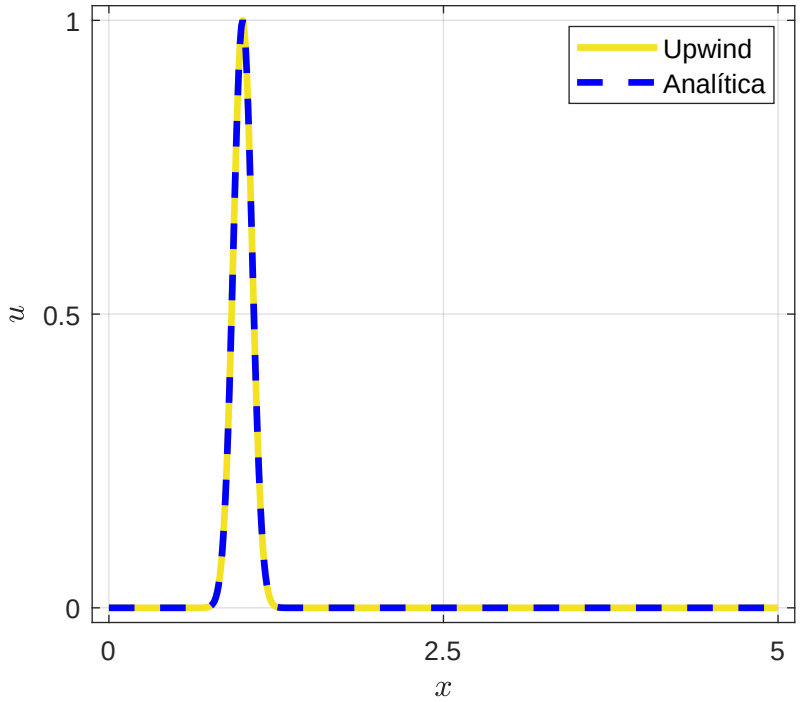

FIGURA 19 – Comparação entre a solução numérica via upwind e a solução analítica da equação (3.13) no instante $t = 0$.

Então, com os dados obtidos pelo *upwind*, passamos para a segunda etapa desta formulação. O treinamento de uma rede neural que se ajuste aos dados, com a função de perda (4.5). Os parâmetros utilizados na rede neural são descritos nas Tabelas 1 e 2, com exceção da quantidade de épocas, que foram utilizadas 1500.

O resultado do treinamento da rede neural com os dados do esquema *upwind* é ilustrado nas Figuras 22 e 21. É possível observar que o ajuste aos dados não está muito preciso, mas de certo modo, o gráfico da solução calculada pela rede neural obteve um formato parecido com o gráfico da solução analítica do problema. Espera-se que essa aproximação, mesmo que com erros, seja suficiente para que a rede neural, quando treinada com a função de perda tradicional de PINNs, convirja mais facilmente para a solução da EDP (3.13).

Prosseguindo com a formulação das PINNs com métodos clássicos, a função de perda da rede neural será substituída pela função custo original das PINNs, dada

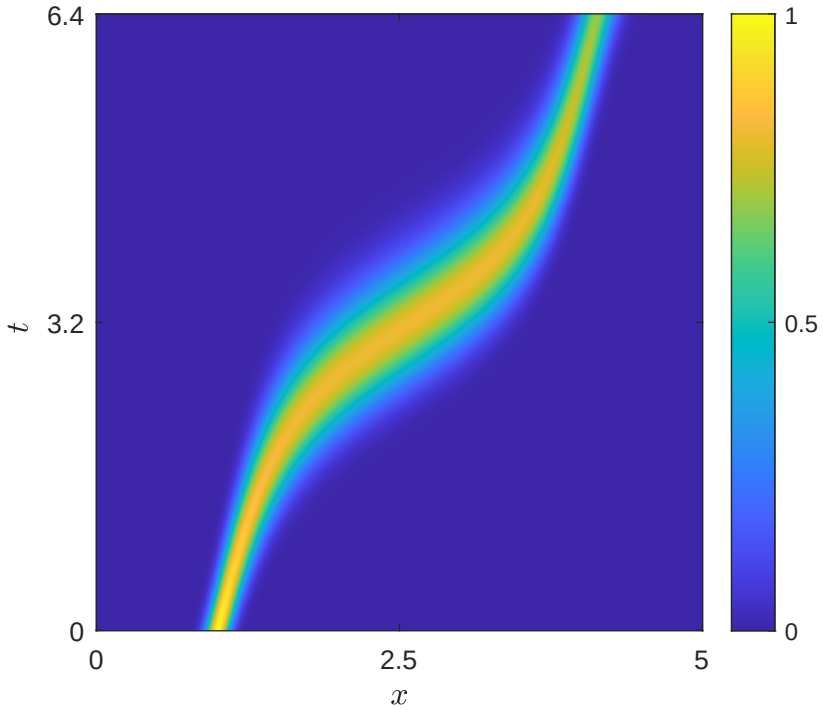

FIGURA 20 – Mapa de calor da solução numérica via upwind da equação (3.13).

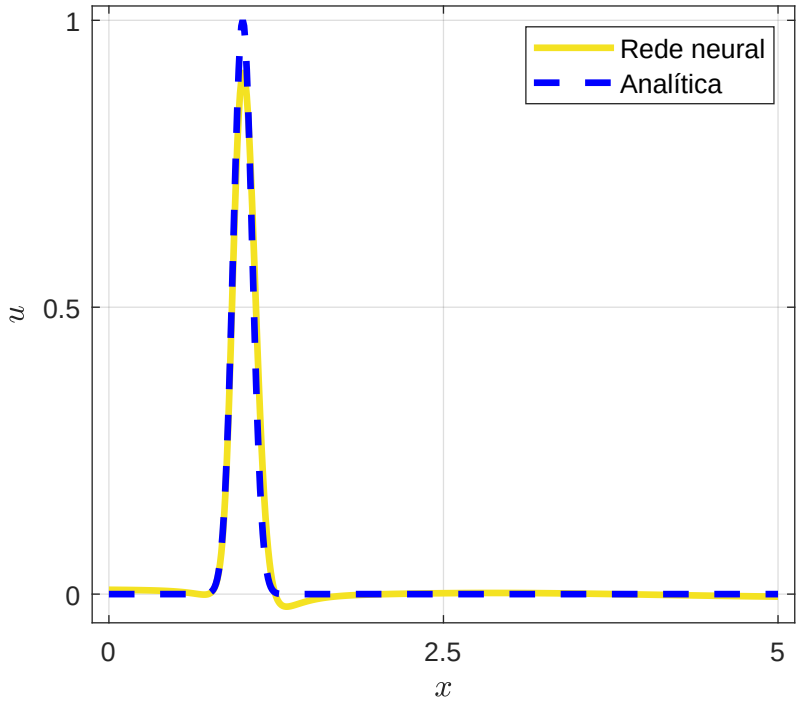

FIGURA 21 – Comparação entre a rede neural ajustada aos dados e a solução analítica da equação (3.13).

na expressão (3.6). Com isso, a rede neural será treinada como uma PINN por 8500 épocas, totalizando 10000 épocas durante toda a formulação.

As Figuras 24 e 23 expõem o resultado final da formulação de PINNs com

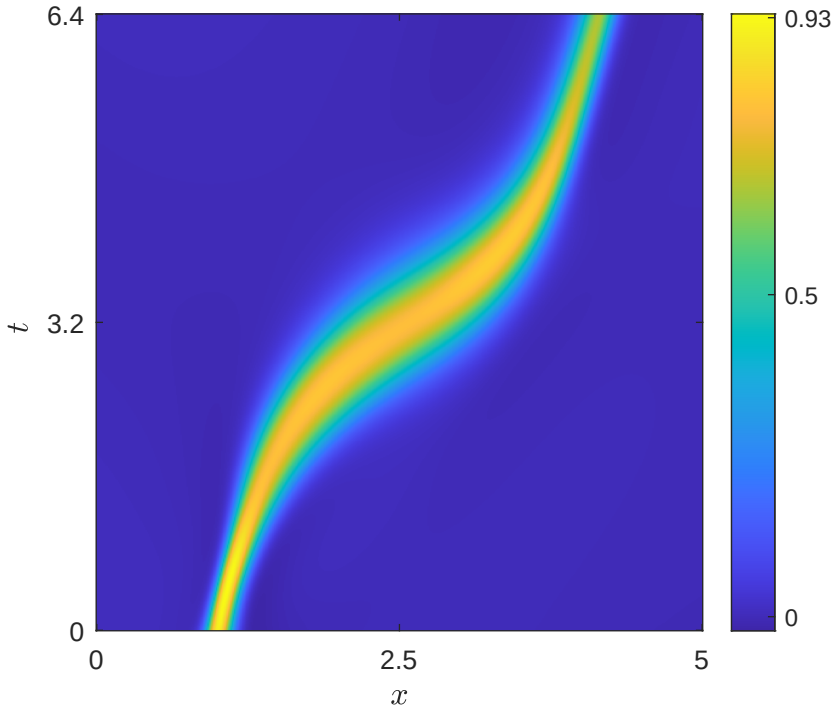

FIGURA 22 – Mapa de calor da rede neural ajustada aos dados.

métodos clássicos. O dado inicial da equação (3.13) não foi bem aproximado, com erro relativo, definido em (3.11), de $9{,}4 \cdot 10^{-2}$, enquanto a aproximação ao em todo o domínio também não foi satisfatória, já que mesmo com o treinamento prévio a falha de propagação se manteve, o que mostra que essa estratégia não foi eficiente na aproximação da solução da EDP (3.13).

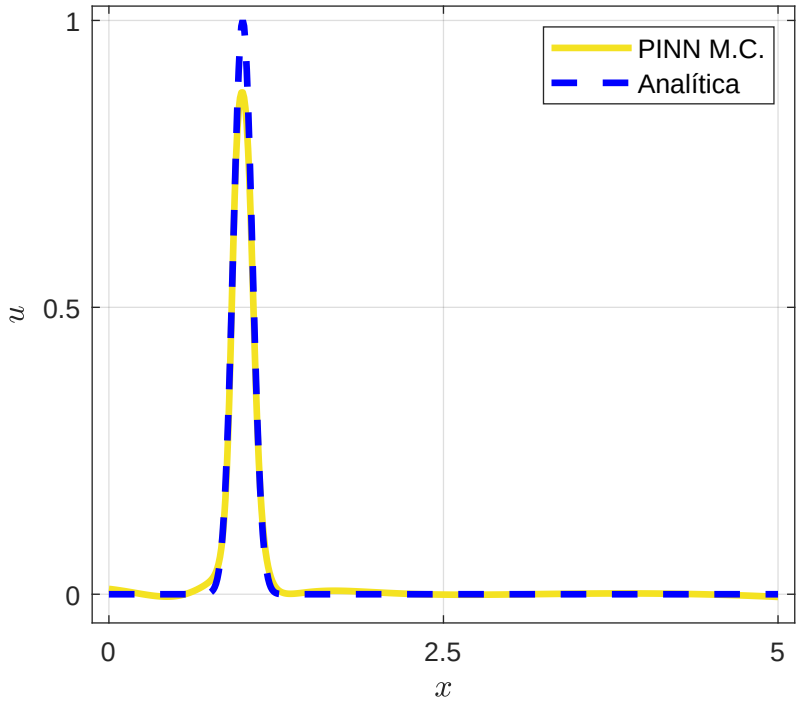

FIGURA 23 – Comparação entre a solução numérica via PINN com métodos clássicos e a solução analítica da equação (3.13) no instante $t = 0$.

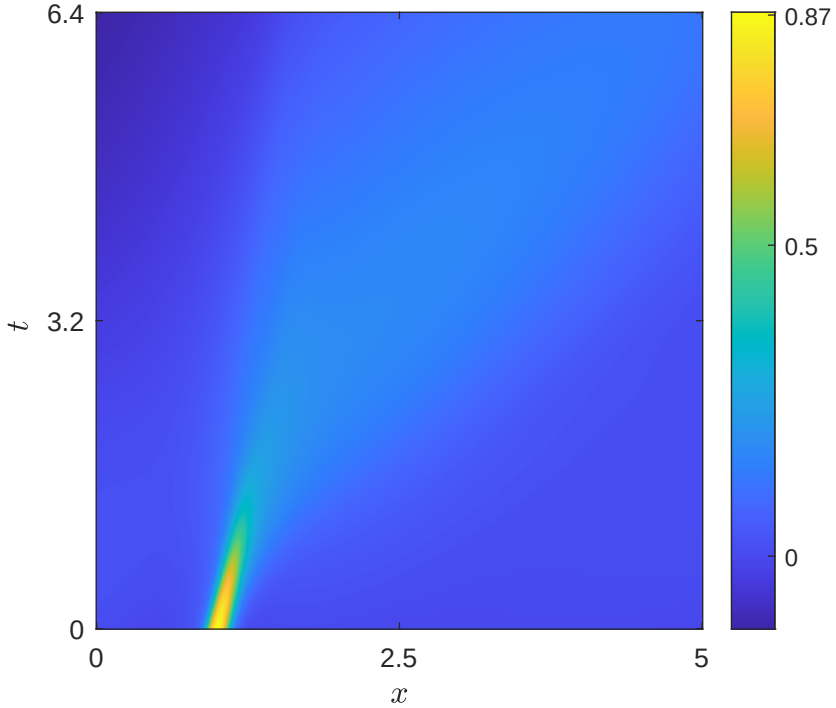

FIGURA 24 – Mapa de calor da solução numérica via PINN com métodos clássicos da equação (3.13).

5 CONCLUSÃO

O objetivo deste trabalho de conclusão de curso foi revisar, propor e analisar a aplicação de estratégias adicionais à formulação clássica de redes neurais informadas pela física na aproximação da solução da equação do transporte com velocidade variável, com o objetivo de evitar soluções triviais não desejadas. As estratégias revisadas neste trabalho foram os pontos adaptativos e a função *gate*, enquanto a formulação proposta foi a união de PINNs com métodos numéricos clássicos.

Para apresentar as PINNs, fez-se necessário revisar a noção de uma rede neural, sua formulação matemática, assim como conceitos básicos, como camadas, neurônios, função de ativação, pesos e vieses, além de uma breve explicação sobre o treinamento de uma rede neural.

Em seguida, foi realizada a descrição do método das PINNs: como definir um problema de otimização com base em uma EDP. A aplicação desse método na equação do transporte expôs a falha de propagação, uma dificuldade enfrentada pelas PINNs na solução de EDPs homogêneas. Neste exemplo o dado inicial da EDP foi aproximado com precisão, com erro relativo de $6{,}3 \cdot 10^{-3}$, porém a solução numérica como um todo foi insatisfatória, tendo em vista a convergência para a solução trivial de forma errônea em uma determinada região do domínio.

Após a apresentação da falha de propagação, foi estudada a estratégia de pontos adaptativos para PINNs. A aplicação desse método na equação do transporte com velocidade variável obteve um resultado satisfatório, com erro relativo de $1{,}6 \cdot 10^{-2}$ em comparação a solução analítica da EDP, enquanto manteve uma aproximação precisa em comparação do dado inicial da EDP, com erro relativo de $8{,}3 \cdot 10^{-3}$.

Adiante, foi revisada a estratégia de PINNs com função *gate*, em que foi proposta uma abordagem diferente para a atualização do parâmetro θ da função *gate*, baseada nas épocas do treinamento da rede neural. No entanto, o resultado obtido com a aplicação dessa metodologia na equação do transporte apresentou limitações e não alcançou o desempenho esperado, uma vez que a falha de propagação permaneceu, mesmo que a aproximação do dado inicial da EDP tenha sido satisfatória, com erro relativo de $2{,}9 \cdot 10^{-3}$. Esse comportamento é provavelmente devido à alta frequência da atualização do parâmetro da função *gate*, além de uma escolha de parâmetros não otimizada.

Por fim, a união de PINNs com métodos clássicos é apresentada como uma terceira alternativa para evitar a falha de propagação. Inicialmente, foi utilizado o esquema *upwind* para criar um conjunto de dados. A seguir, uma rede neural foi

treinada com os dados gerados pelo *upwind*. Finalmente, essa rede neural foi treinada como uma PINN. Essa formulação não se mostrou eficiente na resolução da equação do transporte com velocidade variável, mantendo a falha de propagação e alcançando o pior erro relativo em comparação ao dado inicial, com o valor de $9{,}4 \cdot 10^{-2}$. Esse resultado é provavelmente devido a uma escolha inadequada do método clássico: o esquema *upwind* tem comportamento dissipativo devido aos erros de aproximação, observado nos estudos numéricos realizados. Esse comportamento pode ter levado a PINN a uma falha de propagação que foi mantida durante todo o processo de otimização. Outro possível motivo da ineficiência dessa estratégia é a mudança entre a rede neural que se ajusta aos dados do método clássico e a PINN, que neste trabalho foi feita de forma brusca.

Conclui-se que, dentre os métodos adicionais às PINNs, a estratégia de pontos adaptativos mostrou-se a mais eficiente, sendo a única capaz de resolver com precisão a equação do transporte com velocidade variável. Vale ressaltar que as estratégias revisadas e propostas neste trabalho, assim como as PINNs de forma geral, dependem de parâmetros definidos pelo usuário, os quais influenciam diretamente o desempenho e a precisão das aproximações obtidas. Os resultados apresentados neste trabalho são preliminares, motivando investigações futuras mais abrangentes sobre todos os temas aqui abordados, incluindo a escolha dos parâmetros utilizados nas estratégias e as equações diferenciais envolvidas.

REFERÊNCIAS

BEZANSON, J.; EDELMAN, A.; KARPINSKI, S.; SHAH, V. B. Julia: A fresh approach to numerical computing. **SIAM Review**, SIAM, v. 59, n. 1, p. 65–98, 2017. Acesso em: 09/12/2024. Disponível em: https://epubs.siam.org/doi/10.1137/141000671. Citado 1 vez na página 9.

BOTELHO, J. G. S. **Formulação de Esquemas Numéricos para a Resolução de Modelos de Equações Diferenciais Parciais de Transporte com Velocidade Variável e Fronteira Periódica**. Curitiba: 2023. Trabalho de Conclusão de Curso, Matemática Industiral, Universidade Federal do Paraná. Citado 2 vezes nas páginas 15, 29.

BRAMBURGER, J. J. **Data-Driven Methods for Dynamic Systems**. : SIAM, 2024. Citado 1 vez na página 10.

CARVALHO, R. S. **Análise do Desempenho de Redes Neurais Artificiais em Problemas de Classificação Multiclasse.** Brasília: 2024. Dissertação, Mestrado em Matemática, Universidade de Brasília. Citado 1 vez na página 10.

DAW, A.; BU, J.; WANG, S.; PERDIKARIS, P.; KARPATNE, A. Mitigating Propagation Failures in Physics-Informed Neural Networks using Retain-Resample-Release (R3) Sampling. **arXiv preprint arXiv:2207.02338**, 2022. Citado 6 vezes nas páginas 7, 9, 18, 23, 25.

DIAB, W.; KOBAISI, M. A. PINNs for the Solution of the Hyperbolic Buckley-Leverett Problem with a Non-convex Flux Function. **arXiv preprint arXiv:2112.14826**, 2021. Citado 1 vez na página 7.

FRACES, C. G.; TCHELEPI, H. Physics informed deep learning for flow and transport in porous media. **arXiv preprint arXiv:2112.14826**, 2021. Citado 1 vez na página 7.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. ISSN 0893–6080. DOI: 10.1016/0893-6080(89)90020-8. Citado 1 vez na página 13.

JACOT, A.; GABRIEL, F.; HONGLER, C. Neural tangent kernel: Convergence and generalization in neural networks. **Advances in neural information processing systems**, v. 31, 2018. Citado 1 vez na página 7.

LINEARALGEBRA.JL. **GitHub**. : 2024. Acesso em: 09/12/2024. Disponível em: https://github.com/JuliaLang/julia. Citado 1 vez na página 9.

MA, Y.; GOWDA, S.; ANANTHARAMAN, R.; LAUGHMAN, C.; SHAH, V.; RACKAUCKAS, C. **ModelingToolkit.jl GitHub.** : 2021. Acesso em: 09/12/2024. Disponível em: https://github.com/SciML/ModelingToolkit.jl. Citado 1 vez na página 9.

MKL.JL. **GitHub**. : 2024. Acesso em: 09/12/2024. Disponível em: https://github.com/JuliaLinearAlgebra/MKL.jl. Citado 1 vez na página 9.

MOCHINSKI, M. E. D. N. **Aspectos Matemáticos do Treinamento de Redes Neurais Artificiais**. Curitiba: 2023. Trabalho de Conclusão de Curso, Matemática Industiral, Universidade Federal do Paraná. Citado 1 vez na página 10.

OPTIMIZATION.JL. **GitHub**. : 2024. Acesso em: 09/12/2024. Disponível em: https://github.com/SciML/Optimization.jl. Citado 1 vez na página 9.

PAL, A. **Lux.jl. GitHub**. : 2022. Acesso em: 09/12/2024. Disponível em: https://github.com/LuxDL/Lux.jl. Citado 1 vez na página 9.

RAISSI, M.; PERDIKARIS, P.; KARNIADAKIS, G. E. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. **arXiv preprint arXiv:1711.10561**, 2017. Citado 3 vezes nas páginas 7, 9, 12.

STRIKWERDA, J. C. **Finite difference schemes and partial differential equations**. : SIAM, 2004. Citado 1 vez na página 29.

ZIENKIEWICZ, O. C.; TAYLOR, R. L. **The finite element method set**. : Elsevier, 2005. Citado 1 vez na página 18.

ZUBOV, K.; MCCARTHY, Z.; MA, Y.; CALISTO, F.; PAGLIARINO, V.; AZEGLIO, S.; BOTTERO, L.; LUJÁN, E.; SULZER, V.; BHARAMBE, A.; VINCHHI, N.; BALAKRISHNAN, K.; UPADHYAY, D.; RACKAUCKAS, C. NeuralPDE: Automating Physics-Informed Neural Networks (PINNs) with error approximations. **arXiv preprint arXiv:2107.09443**, 2021. Citado 1 vez na página 9.