INTERFACE TECNOLÓGICA

MÉTODO DE RUNGE-KUTTA E REDES NEURAIS PARA APROXIMAÇÃO NUMÉRICA DE EQUAÇÕES DIFERENCIAIS ORDINÁRIAS

RUNGE-KUTTA METHOD AND NEURAL NETWORKS FOR NUMERICAL APPROXIMATION OF ORDINARY DIFFERENTIAL EQUATIONS

Gilberto Sussumu Hida – gilberto.hida@fatec.sp.gov.br
Faculdade de Tecnologia de São Caetano do Sul (Fatec) – São Caetano do Sul – SP – Brasil

Clayton Suguio Hida – clayton.hida@fatec.sp.gov.br
Faculdade de Tecnologia de Praia Grande (Fatec) – Praia Grande – SP – Brasil

DOI: 10.31510/infa.v21i1.1965
Data de submissão: 15/04/2024
Data do aceite: 10/03/2024
Data da publicação: 20/06/2024

RESUMO

Neste trabalho, apresentamos uma abordagem que combina métodos numéricos tradicionais de resolução de equações diferenciais com técnicas de machine learning, visando obter aproximações mais eficientes das soluções de equações diferenciais ordinárias (EDOs). Em particular, propomos a integração do método de Runge-Kutta para resolver problemas de valores iniciais com redes neurais. O método de Runge-Kutta é inicialmente empregado para gerar aproximações da solução em uma vizinhança do ponto inicial. Em seguida, treinamos uma rede neural utilizando esses pontos como dados de entrada. Os resultados obtidos demonstram que o modelo resultante possui um desempenho comparável ao da rede neural treinada exclusivamente com os pontos da solução real. Na comparação com o método de Runge-Kutta em todo intervalo, usando a mesma quantidade de pontos, o modelo apresentou um desempenho melhor. Entretanto, observamos que a arquitetura da rede neural implementada no processo de modelagem não foi capaz de superar o desempenho do método de Runge-Kutta em todo o intervalo, quando usamos o mesmo comprimento de passo para a obtenção dos pontos. Este resultado sugere a necessidade de investigações mais aprofundadas na concepção da arquitetura da rede neural para melhorar sua capacidade de generalização em problemas de EDOs.

Palavras-chave: Aprendizado de máquina. Redes neurais. Problema de valor inicial. Aproximações.

ABSTRACT

In this work, we present an approach that combines traditional numerical methods for solving differential equations with machine learning techniques, aiming to obtain more efficient approximations of solutions to ordinary differential equations (ODEs). In particular, we propose integrating the Runge-Kutta method to solve initial value problems with neural networks. The Runge-Kutta method is initially employed to generate approximations of the solution in a neighborhood of the initial point. Subsequently, we train a neural network using these points as

input data. The results obtained demonstrate that the resulting model performs comparably to the neural network trained solely with points from the real solution. In the comparison with the Runge-Kutta method over the entire interval, using the same number of points, the model performed better. However, we observed that the neural network architecture implemented in the modeling process was not able to outperform the Runge-Kutta method over the entire interval, when we use the same step length for obtaining the points. This result suggests the need for further investigations in the design of the neural network architecture to improve its generalization capability in ODE problems.

Keywords: Machine learning. Neural networks. Initial value problem. Approximations.

1 INTRODUÇÃO

As equações diferenciais ordinárias nos fornecem um importante meio de modelar fenômenos da natureza. Sobre certas condições simples de se verificar, a teoria nos fornece a existência e até mesmo a unicidade de soluções para nossos problemas. O problema geralmente vem do fato de que esses resultados gerais sobre equações diferenciais ordinárias, nos garantem a existência das soluções, mas não nos fornecem um meio analítico para encontrar essas soluções. “Infelizmente, existem muitos problemas importantes em Engenharia e ciência, especialmente problemas não lineares, nos quais esses métodos ou não se aplicam, ou seu uso é muito complicado”. (BOYCE, 2020).

Surge daí a necessidade de realizarmos aproximações da solução. Um meio de encontrar uma aproximação é através de métodos numéricos já consagrados, como o Método de Euler e o Método de Runge-Kutta. Por outro lado, temos presenciado o avanço da Inteligência artificial em várias vertentes do conhecimento, como por exemplo no processo de diagnósticos (SANTOS, 2019), em diversos ramos da engenharia (MALIK, 2018) e até mesmo em matemática.

Neste artigo, propomos um método que combina métodos numéricos clássicos e o uso de redes neurais. Em (HIDA e HIDA, 2023), os autores utilizaram o método de Euler para obtenção de pontos da solução de um problema de valor inicial. Depois treinaram uma rede neural com esses pontos para obter uma aproximação da solução em um intervalo estendido. Os resultados mostraram que a combinação do método de Euler e de redes neurais teve um desempenho melhor do que apenas o modelo de Euler. Como extensão dos resultados obtidos

em (HIDA e HIDA, 2023), vamos analisar se a mesma combinação de métodos contínua fornecendo resultados melhores, quando consideramos os métodos de Runge-Kutta.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, faremos uma revisão literária sobre equações diferenciais e métodos numéricos que formaram a base para o desenvolvimento deste artigo. Vamos iniciar com as equações diferenciais:

2.1 Equações diferenciais ordinárias.

Uma equação diferencial ordinária é uma igualdade que envolve uma função desconhecida $y = y(x)$

e suas derivadas. Mais formalmente, uma equação diferencial ordinária é uma equação da forma $F\left(x,\ y,\ y',\ y'',..y^{(n)}\right) = 0$

. (SOTOMAYOR, 1979).

Podemos pensar em qualquer equação diferencial como sendo equivalente a uma equação diferencial da forma $y = f(x,\ y)$

, que é denominada de equação diferencial de primeira ordem. Para isso, basta que aumentemos a dimensão do nosso problema. Adicionalmente, se fixamos um par de valores $(x_0,\ y_0)$

e adicionamos a condição de que nossa solução deve satisfazer $y(x_0) = x_0$

, temos o que é chamado de Problema de Valor Inicial (p.v.i) (VIANA e ESPINAR, 2021).

Sob algumas condições sobre a função $f(x, y)$

, podemos sempre garantir a existência e unicidade de soluções do p.v.i, ou seja, podemos garantir a existência e unicidade de uma função $y = y(x)$

que é derivável, com $y(x_0) = y_0$

e tal que $y'(x) = f\left(x,\ y(x)\right)$

para todo x

em um dado intervalo. Esse é o conteúdo do Teorema de Existência e Unicidade para equações diferenciais ordinárias. Recomendamos o leitor ao Teorema 2 do Capítulo 1 de (SOTOMAYOR, 1979).

2.2 Método de Runge-Kutta

Os métodos de Runge-Kutta são métodos numéricos para aproximar soluções de p.v.i, desenvolvidas pelos matemáticos C. Runge e M. W. Kuta por volta de 1900 (BUTCHER, 2016). Trata-se de um método numérico que generaliza os métodos de Euler, que foram os métodos escolhidos em (HIDA e HIDA, 2023). Vamos utilizar nesse artigo a versão clássica do método de Runge-Kutta, também conhecido como método de quarta ordem ou de quatro estágios (BOYCE, 2020). O método é formulado da seguinte forma: Dado um p.v.i $y' = f(x,y),\ y(x_0) = y_0$

e um parâmetro h

, denominado de passo, o método consiste em primeiro obter uma sequência de pontos $x_{n+1} = x_n + h$

(pontos do domínio) e indutivamente, obter uma sequência de pontos $y_{n+1} = \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

, onde

$$k_1 = f(x_k,\ y_k)$$

$$k_2 = f\left(x_k + \frac{h}{2},\ y_k + \frac{hk_1}{2}\right)$$

$$k_3 = f\left(x_k + \frac{h}{2},\ y_k + \frac{hk_2}{2}\right)$$

- e

$$k_4 = f(x_k + h,\ y_k + hk_3)$$

- .

3 PROCEDIMENTOS METODOLÓGICOS

Neste artigo, vamos implementar um modelo de aproximação de solução, que combina o método de Runge-Kutta e uma rede neural. Como situação de teste, vamos fixar o p.v.i

$$y' = -50x,\ y(0) = 1$$

, que tem como solução a função exponencial $y(x) = e^{-50x}$

. Para as simulações, vamos utilizar a linguagem Python 3.1. Nosso modelo, denominado de RK + Neural é construído da seguinte forma: No intervalo $[0,2]$

, obtemos 200 pontos que aproximam a solução pelo Método de Runge-Kutta. Depois treinamos uma rede com os 200 pontos como uma aproximação da função solução no intervalo estendido $[0,10]$

. A implementação do método de Runge-Kutta é mostrada na Figura 1:

Figura 1 - Implementação do método de Runge-Kutta

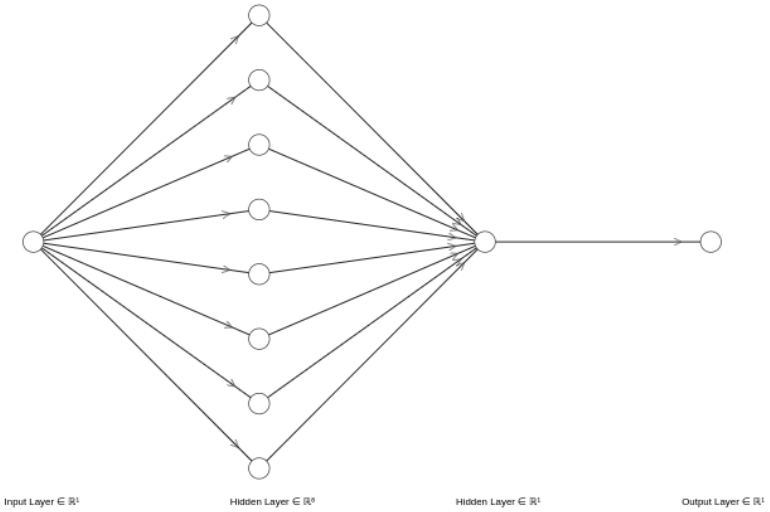
```
def Runge_Kutta(funcao, a,b,y0,n_points, h=0):
    if h==0:
            h = (b-a)/n_points
    else:
        n_points = int((b-a)/h)

    points = np.linspace(a,b, n_points)
    y_values = []
    y_values.append(y0)
    y_temp = y0
    for i in np.arange(1, n_points):
        k1 = funcao(points[i], y_temp)
        k2 = funcao(points[i] + h/2, y_temp + h*k1/2)
        k3 = funcao(points[i] + h/2, y_temp + h*k2/2)
        k4 = funcao(points[i] + h/2, y_temp + h*k3)
        yn = y_temp + (h/6)*(k1 + 2*k2 + 2*k3 + k4)
        y_values.append(yn)
        y_temp = yn
    return points, y_values
```

Fonte: autores (2024)

Já para a rede neural, utilizamos a mesma arquitetura implementada em (HIDA e HIDA, 2023) como pode ser observada na Figura 2 e Figura 3:

INTERFACE TECNOLÓGICA

Figura 2 – Arquitetura da rede neural



Fonte: (HIDA e HIDA, 2023)

Figura 3 – Classe Neural Network

```
# Rede Neural
class Neural_Network:
    def __init__(self, nodes, num_epochs, learning_rate):
        self.nodes = nodes
        self.num_epochs = num_epochs
        self.learning_rate = learning_rate

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def sigmoid_derivative(self,x):
        return self.sigmoid(x) * (1 - self.sigmoid(x))

    def fit_nn(self, X, Y):
        pass
    # Predição
    def predict(self, X):
        pass
```

Fonte: (HIDA e HIDA, 2023)

Para efeito de comparação, também consideramos um modelo denominado de Neural que é um modelo de rede neural treinada com alguns pontos da solução real do nosso p.v.i e um modelo denominado de RK All, que é a aplicação do método de Runge-Kutta no intervalo $[0,10]$

.

4 RESULTADOS E DISCUSSÃO

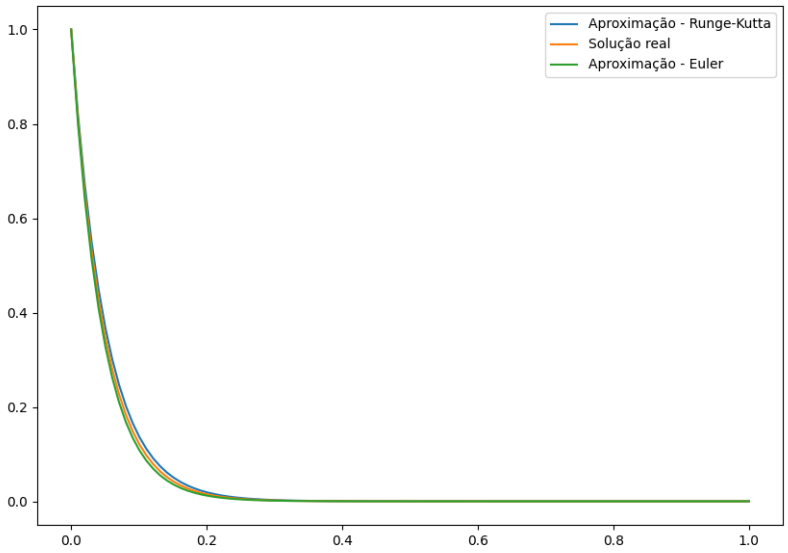
Os métodos de Runge-Kutta apresentam melhores resultados do que o Método de Euler, principalmente devido aos erros de truncamento. Para os métodos de Euler, o erro de truncamento local de $O(h^2)$

, enquanto o método de Runge-Kutta (de quarta ordem) apresenta erro da ordem $O(h^4)$

. A Figura 4 mostra os dois métodos aplicados para o p.v.i $y' = -20y,\ y(0) = 1$

.

Figura 4 – Euler e Runge-Kutta.



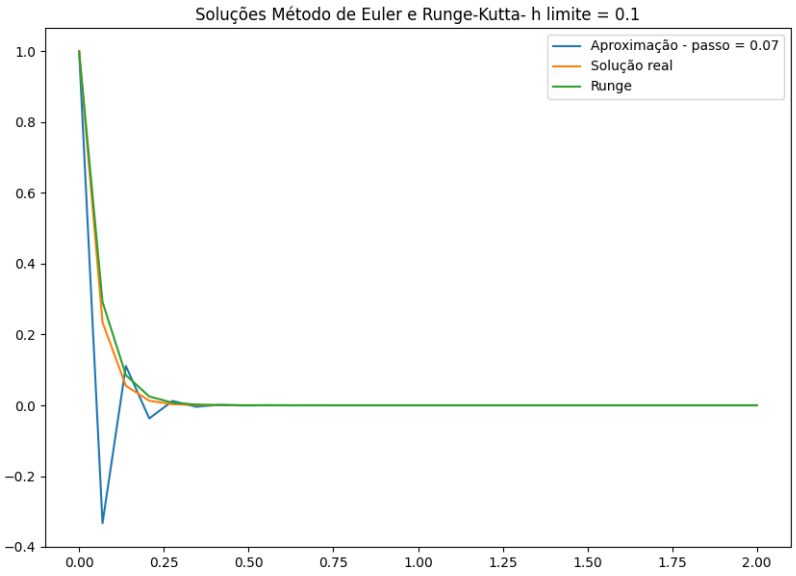
Fonte: autores (2024)

Os métodos de Runge-Kutta apresentam melhores desempenhos em relação ao método de Euler em questões relacionadas a acurácia e estabilidade (BURDEN, 2015). Em particular, a Figura 5 mostra um comparativo entre o método de Runge-Kutta e o método de Euler para uma particular escolha de passo h

.

Figura 5 – Euler e Runge-Kutta - Estabilidade

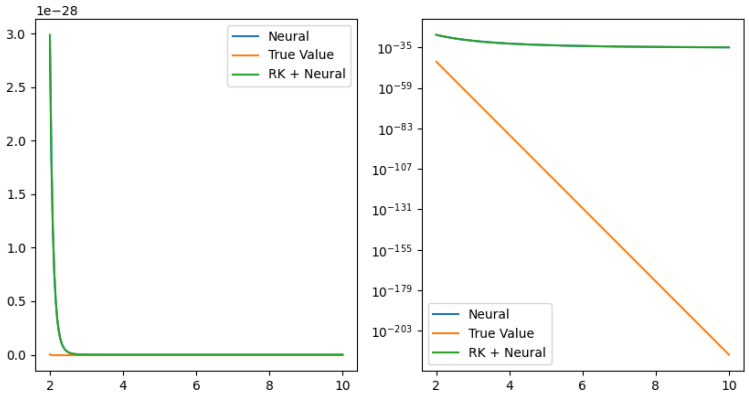
INTERFACE TECNOLÓGICA



Fonte: autores (2024)

A Figura 6 apresenta um comparativo entre os modelos RK+Neural, Neural e a solução real.

Figura 6 – RK+Neural e solução real



Fonte: autores (2024)

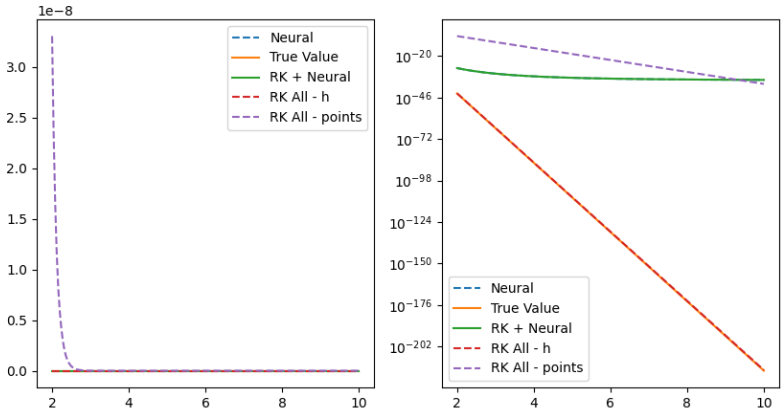
Observamos que os modelos RK+Neural e Neural apresentam comportamento bem semelhantes. Para efeito de comparação, realizamos simulações do método de Runge-Kutta para todo o intervalo $[0,10]$

. Neste caso, temos dois parâmetros para comparação. Primeiro, simulamos o método de Runge-Kutta com o mesmo parâmetro h dos testes anteriores, ou seja, $h = 0.01$

. Chamamos esse modelo de RK All – h. No outro caso, escolhemos por manter o número de pontos obtidos pelo método de Runge-Kutta em 200 pontos e denominamos esse modelo de RK All – points.

Finalmente, a Figura 7 mostra um comparativo entre os quatro modelos: RK+Neural, Neural, RK All –h e RK All – points.

Figura 7 – Runge-Kutta em todo intervalo



Fonte: autores (2024)

Podemos observar que o modelo RK All – h possui uma melhor performance, quando comparado aos outros modelos. Isso também pode ser confirmado pela Figura 8 que mostra os erros (erro médio quadrático) em ordens de magnitude de cada modelo em relação a solução real:

Figura 8 – Tabela de erros

Modelo	Error 10^x
RK+Neural	-58
Neural	-58
RK All - h	-91
RK All - points	-17
Euler+Neural	-57

Fonte: autores (2024)

Na Figura 8 também incluímos o modelo Euler + Neural de (Hida e Hida, 2023). Observamos que o modelo RK+Neural apresenta um desempenho melhor que o modelo obtido em (Hida e Hida, 2023), sendo um resultado esperado, dado que estamos usando um método numérico melhor para a obtenção dos pontos iniciais. A Figura 9 mostra um comparativo entre os modelos RK+Neural e Euler + Neural para diferentes simulações dos parâmetros k, ou seja, para as famílias de p.v.i $y' = ky,\ y(0) = 1$

.

Figura 9 – RK+Neural e Euler+Neural.

k	RK+Neural	Euler+Neural
-10	5.086588e-18	6.042084e-19
-20	2.667452e-35	1.115447e-37
-30	6.510969e-52	1.050338e-54
-40	2.650961e-56	6.351239e-58
-50	7.476554e-58	1.991312e-57
-60	5.361232e-58	6.448621e-55
-70	2.168132e-57	2.680988e-51
-80	2.108140e-56	3.485035e-47
-90	2.955762e-55	5.458769e-43
-100	4.346761e-54	5.663956e-39

Fonte: autores (2024)

Observamos que o modelo Euler + Neural (HIDA e HIDA, 2023) apresenta um melhor desempenho para valores de k menores que 50, mas que para valores maiores que 50, o modelo RK+Neural tem um desempenho melhor.

5 CONCLUSÃO

Neste artigo, que é uma extensão do artigo (Hida e Hida, 2023), propomos o uso de métodos numéricos clássicos para solução numérica de equações diferenciais e o uso de redes neurais. Em (Hida & Hida, 2023), observamos que o uso do Método de Euler e redes neurais mostrou um desempenho melhor que o uso isolado do método de Euler para obtenção de aproximações de problemas de valores iniciais para pontos distantes do ponto inicial. Neste artigo, analisamos se o mesmo resultado valia para o método de Runge-Kutta e redes neurais. Como observamos pela Figura 7 e 8, o treino de uma rede neural com pontos obtidos do método de Runge-Kutta no intervalo $[0,2]$

como uma aproximação da solução do p.v.i $y' = -50y,\ y(0) = 1$

no intervalo $[0,10]$

foi capaz de superar o método de Runge-Kutta aplicado a todo o intervalo $[0,10]$

se considerarmos a mesma quantidade de pontos, mas apresentou desempenho inferior ao método de Runge-Kutta com o mesmo parâmetro h (observando que neste caso o método é calculado em um número de pontos superior ao do modelo RK + Neural). Concluímos também,

baseado na Figura 9, que o modelo RK+Neural apresenta um desempenho melhor que o modelo desenvolvido em (HIDA e HIDA, 2023). Essa conclusão está dentro do esperado, uma vez que os métodos de Runge-Kutta usados para a obtenção dos pontos iniciais tem um desempenho melhor que os métodos de Euler. Como direções futuras, podemos melhorar a estrutura da rede neural apresentada na Figura 2, bem como treinar uma rede neural para obter uma solução do p.v.i, utilizando uma função de perda adaptada (MALL, 2016) (CHEN et al, 2018).

REFERÊNCIAS

BOYCE, W. E. **Equações diferenciais elementares e problemas de valores de contorno.** Grupo GEN, 2020.

BURDEN, R. L.; FAIRES, D. J.; BURDEN, A. M. **Numerical analysis.** Cengage learning, 2015.

BUTCHER, J. C. **Numerical methods for ordinary differential equations.** John Wiley & Sons, 2016.

CHEN, R.T.Q. et al. **Neural ordinary differential equations.** Advances in neural information processing systems, v.31, 2018.

HIDA, C.S.; HIDA, G.S. **Método de Euler e redes neurais para aproximação numérica de equações diferenciais ordinárias.** Interface Tecnológica, v.20, n.12, Dez.2023.

MALIK, H. et al. **Applications of artificial intelligence techniques in engineering.** Sigma 2018, v.2, 2019.

MALL, S.; CHAKRAVERTY, S. **Application of Legendre Neural network for solving ordinary differential equations.** Aplied Soft Computing, v43, June.2016.

SANTOS, M.K. et al. **Inteligência artificial, aprendizado de máquina, diagnóstico auxiliado por computador e radiômica: avanços da imagem rumo à medicina de precisão.** Radiologia brasileira, v.52, Dez. 2019.

SOTOMAYOR, J. **Lições de equações diferenciais ordinárias.** IMPA, 1979.

VIANA, M.; ESPINAR, J. **Equações Diferenciais: Uma abordagem de Sistemas Dinâmicos,** IMPA. 2021.