# Resolução de Sistema de Equações Lineares de Matrizes Pentadiagonais com Redes Neurais Convolucionais*

Tiago C A Amorim (RA: 100675)[a], Taylon L C Martins (RA: 177379)[b]

[a]*Doutorando no Departamento de Engenharia de Petróleo da Faculdade de Engenharia Mecânica, UNICAMP, Campinas, SP, Brasil*
[b]*Aluno especial, UNICAMP, Campinas, SP, Brasil*

*Keywords:* Rede Neurais Convolucionais, Sistemas de Equações Lineares

## 1. Introdução

## 2. Motivação

O método das diferenças finitas é utilizado para resolver diferentes problemas físicos que podem ser descritos como equações diferenciais. O método se baseia na aproximação das derivadas por diferenças finitas (Exemplos nas equações 2.1). O domínio espaço-temporal é discretizado e a solução das equações diferenciais é aproximada nos nós desta malha [1]. O problema então é transformado em uma série de equações não-lineares, que usualmente são resolvidas por métodos numéricos, como o método de Newton-Raphson [2].

$$\frac{\partial u(x)}{\partial x} \approx \frac{u(x+h) - u(x-h)}{2h} \tag{2.1a}$$

$$\frac{\partial^2 u(x)}{\partial x^2} \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} \tag{2.1b}$$

É usual utilizar aproximações de derivada que utilizam os valores da função do nó e seus vizinhos diretos. Em problemas bidimensionais com malhas regulares (Figura 1) esta escolha leva a sistemas de equações com matrizes pentadiagonais (2.2). Cada equação não-linear tem termos associados a um dos nós (i,j) e seus quatro vizinhos: (i-1,j), (i+1,j), (i,j-1) e (i,j+1). A resolução numérica deste sistema de equações não-lineares usualmente está associada a métodos iterativos, em que novas equações lineares são resolvidas. Desta forma, a resolução do problema original está associada à solução de um significativo número de sistemas de equações lineares com matriz pentadiagonal.

A proposta deste trabalho é avaliar a possibilidade de utilizar uma rede convolucional para resolver este tipo de sistema de equações lineares.

Figura 1: Esquemático de uma malha regular, com destaque para a célula (i,j) e seus vizinhos.

| | | |
|---|---|---|
| | (i,j-1) | |
| (i-1,j) | **(i,j)** | (i+1,j) |
| | (i,j+1) | |

## 3. Trabalhos Correlatos

Não foram encontrados muitos trabalhos com foco na resolução de sistemas de equações lineares com redes neurais. Duas formas distintas de resolução do problema foram propostas. A primeira vertente é resolver o sistema de equações lineares junto com o treinamento da rede ([3]). Uma aplicação interessante desta proposta é o de resolver sistema de grande dimensão, que possivelmente não cabem na memória disponível, e usar a rede para aprender um mapeamento que aproxima a resposta ([4]).

Uma segunda vertente é a de treinar uma rede neural com base em vários exemplos de sistemas de equações a resolver. A rede treinada é utilizada para resolver novos sistemas de equações. Uma proposta focou na solução de sistemas tridiagonais ([5]), utilizando uma série de camadas densas seguidas por conexões residuais. Um outro trabalho ([6]) foca na solução de problemas físicos associados a equações diferenciais. Este trabalho tenta primeiro encontrar uma representação densa dos dados de entrada por meio de uma rede *autoenconder*. Posteriormente a representação densa de cada amostra passa por uma outra rede neural que busca resolver o problema.

A proposta estudada neste projeto segue a segunda vertente. É feita a opção de utilizar camadas convolucionais para que a arquitetura da rede seja agnóstica à discretização do problema (tamanho da malha).

*Projeto final como parte dos requisitos da disciplina IA048: Aprendizado de Máquina.

$$
\begin{bmatrix}
a_{0,1} & a_{1,1} & 0 & \dots & 0 & a_{m,1} & 0 & & \dots & 0 \\
a_{-1,2} & a_{0,2} & a_{1,2} & 0 & \dots & 0 & a_{m,2} & 0 & \dots & 0 \\
0 & a_{-1,3} & a_{0,3} & a_{1,3} & 0 & \dots & 0 & a_{m,3} & \dots & 0 \\
\vdots & & \ddots & \ddots & \ddots & & & & \ddots & \vdots \\
0 & \dots & 0 & a_{-m,n-2} & 0 & \dots & a_{-1,n-2} & a_{0,n-2} & a_{1,n-2} & 0 \\
0 & \dots & & 0 & a_{-m,n-1} & 0 & \dots & a_{-1,n-1} & a_{0,n-1} & a_{1,n-1} \\
0 & \dots & & & 0 & a_{-m,n} & 0 & \dots & a_{-1,n} & a_{0,n}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \\ b_n \end{bmatrix}
\tag{2.2}
$$

## 4. Codificação do Sistema Linear

## 5. Arquitetura da Rede Convolucional

## 6. Resultados

### *6.1. Rede Convolucional* Simples

## 7. Conclusão

## Referências


[1] D. Causon, C. Mingham, Introductory finite difference methods for PDEs, Bookboon, 2010.

[2] R. L. Burden, J. D. Faires, A. M. Burden, Análise numérica, Cengage Learning, 2016.

[3] A. Cichocki, R. Unbehauen, Neural networks for solving systems of linear equations and related problems, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 39 (2) (1992) 124–138.

[4] Y. Gu, M. K. Ng, Deep neural networks for solving large linear systems arising from high-dimensional problems, SIAM Journal on Scientific Computing 45 (5) (2023) A2356–A2381.

[5] Z. Jiang, J. Jiang, Q. Yao, G. Yang, A neural network-based pde solving algorithm with high precision, Scientific Reports 13 (1) (2023) 4479.

[6] K. Kontolati, S. Goswami, G. Em Karniadakis, M. D. Shields, Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems, Nature Communications 15 (1) (2024) 5101.