UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Programa Institucional de Bolsas de Iniciação Científica (PIBIC)

Relatório Parcial

Estudo e implementação de redes neurais hierárquicas de aprendizado profundo para cálculo de interpolações em elementos finitos

Autor: Pedro Azevedo Coelho Carriello Corrêa
Orientador: Dr. Pablo Giovanni Silva Carvalho

São Carlos, SP
Março, 2024

1 Introdução e Motivação

A dinâmica dos fluidos é uma área na qual tendem a surgir sistemas de equações bastante complexas e, por isso, desde o início do uso de computadores para simulações, a Dinâmica dos Fluidos Computacional (CFD) se fez uma das suas grandes vertentes de estudo. Atualmente, com uma grande expansão do uso de Inteligência Artificial em diversos setores, naturalmente são aplicados diversos métodos de aprendizado de máquina na área de CFD. Nesse contexto, um dos métodos que se destacam são as Redes Neurais Artificiais, em especial as de *Multi-Layer Perceptron* (SHARMA et al., 2023).

Nessa pesquisa, são utilizadas redes neurais do tipo *multi-layer perceptron* para aperfeiçoar métodos de soluções numéricas aplicados em equações de dinâmica dos fluidos e futuramente analisar, até certo escopo, o tempo de execução dos algoritmos, a precisão das soluções encontradas e outros parâmetros relevantes para a escolha do uso do método.

1.1 Mudança de direcionamento da pesquisa

No início da pesquisa bibliográfica, o foco era o estudo da implementação de redes neurais para o aprimoramento de solução de sistemas por meio do Método de Diferenças Finitas. Porém, após um estudo inicial desse método (baseado em Langtangen e Linge (2017)) e do Método de Elementos Finitos (a partir de Becker e etc. (1981a)), da formação de um grupo de estudo com professores e alunos de pós-graduação focados na biblioteca `FEniCSx` (que se baseia em elementos finitos), uma decisão foi feita para uma mudança na metodologia do projeto. Por contar com o apoio dos integrantes do grupo de estudos de elementos finitos, e após a constatação de que a mudança de foco no estágio inicial do projeto seria possível, a alteração foi considerada uma decisão sensata pelo aluno e pelo orientador da pesquisa.

Tanto o de diferenças finitas, quanto o de elementos finitos, são métodos de solução numérica extensivamente aplicados e consolidados historicamente no contexto de equações de dinâmica dos fluidos (THOMÉE, 1984) e, no contexto de aprendizado de máquina, as redes neurais artificiais já se provaram como uma ferramenta capaz de aperfeiçoar ambos os métodos, como se pode verificar em Pantidis e Mobasher (2023), Meethal et al. (2023), Le-Duc, Nguyen-Xuan e Lee (2023) para elementos finitos e Tu e Nguyen (2022), Shi et al. (2020) para diferenças finitas. Assim, a mudança da metodologia da pesquisa ainda conserva o uso de métodos já consolidados para soluções numéricas.

2 Objetivos

Inspirado nos artigos Saha et al. (2021) e Zhang et al. (2021b), o projeto visa a formulação de um método que utiliza uma rede neural hierárquica para realimentar a

localização dos nós de elementos finitos. Essa realimentação viria a partir do treinamento da rede e, a partir dela, os elementos finitos ficariam mais refinados nas regiões de maior sensibilidade da solução, ou seja, os nós iriam se concentrar nas regiões que a solução mais flutua, de modo a diminuir seu erro.

3 Metodologia

Os principais modelos de redes neurais e métodos de aprendizado de máquina foram estudados pelo aluno por meio do curso das disciplinas SCC0230 - Inteligência Artificial e SCC0270 - Redes Neurais e Aprendizado Profundo, ministradas, respectivamente, pela Prof. Solange Oliveira Rezende e Prof. Moacir Antonelli Ponti. Já o método de elementos finitos foi estudado principalmente a partir da leitura de Becker e etc. (1981a).

Já a implementação do algoritmo está sendo em `Python`, principalmente através do uso da biblioteca `PyTorch`. Também estão sendo utilizadas outras bibliotecas auxiliares, como a `MatPlotLib` e `NumPy`.

3.1 Redes Neurais *Multi-Layer Perceptron*

Redes neurais de aprendizado profundo são redes que conseguem, em grande parte, superar limitações de modelos lineares *perceptron* incorporando mais camadas. A forma mais natural de fazer isso é alocando camadas conectadas em seguida, de forma que cada camada anterior alimenta a próxima, até gerar a saída do modelo. Essa arquitetura é chamada *multilayer perceptron* (MLP) (ZHANG et al., 2021a).

No geral, redes desse tipo são modelos de aprendizado supervisionado, ou seja, que são treinados a partir de um banco de dados de entradas e suas respectivas saídas esperadas. Porém, nesse projeto foi feita uma modificação em relação ao modelo *perceptron* original, na qual a função de perda (*Loss Function*) utilizada para treinar o modelo não é calculada a partir de um banco de dados, mas sim, recalculando a nova solução em elementos finitos considerando as novas posições dos nós dadas pela rede e retornando o seu erro na equação original. Tal abordagem é mais explicada na Seção 3.4.

3.2 Método de Elementos Finitos

Atualmente, a equação genérica governante no domínio $x_0 < x < x_L$ que o algoritmo em desenvolvimento resolveria numericamente tem a seguinte forma:

$$-k\frac{\mathrm{d}u(x)^2}{\mathrm{d}^2x} + c\frac{\mathrm{d}u(x)}{\mathrm{d}x} + bu(x) = f(x) \tag{1}$$

Onde k, c e b são coeficientes fixos e $f(x)$ uma função de x dados pela equação que se queira resolver. Considerando condição de borda de Dirichlet onde $u(x_0) = u_0$ e $u(x_L) = u_L$, e $v(x)$ uma função de teste definida em todo o intervalo, pode-se analisar a Equação 1 pelo ponto de vista do método de elementos finitos e reescrevê-la na forma variacional:

$$\int_{x_0}^{x_L} (ku'v' + cu'v + buv)\mathrm{d}x = \int_{x_0}^{x_L} (fv)\mathrm{d}x \tag{2}$$

Nesse contexto, v e u para satisfazerem corretamente o enunciado devem pertencer ao subespaço H^1, que são as funções cujas integrais da Equação 3 convergem.

$$\int_{x_0}^{x_L} \left[(v')^2 + v^2\right] \mathrm{d}x < +\infty \tag{3}$$

Dividi-se, arbitrariamente, o domínio do problema em N elementos finitos com h de comprimento. Após isso, são construídas uma função de forma ϕ_i para cada elemento e que geram uma base para o subespaço H^h de H^1: $\{\phi_1, \phi_2, \ldots, \phi_N\}$.

Assim, procura-se uma função $u_h \in H_h$ que pode ser escrita como:

$$u_h(x) = \sum_{j=1}^{N} \alpha_j \phi_j(x) \tag{4}$$

Escrevendo v_h da mesma forma e substituindo na Equação 2, temos:

$$\int_{x_0}^{x_L} (ku_h'v_h' + cu_h'v_h + bu_hv_h)\mathrm{d}x = \int_{x_0}^{x_l} (fv_h)\mathrm{d}x$$

Equivalentemente:

$$\sum_{j=1}^{N} K_{ij}\alpha_j = F_i, \quad i = 1, 2, \ldots N \tag{5}$$

Sendo K comumente chamada de matriz de rigidez e F, de vetor de carga.

$$K_{ij} = \int_{x_0}^{x_L} (k\phi_i'\phi_j' + c\phi_i'\phi_j + b\phi_i\phi_j)\mathrm{d}x \tag{6}$$

$$F_i = \int_{x_0}^{x_L} (f\phi_i)\mathrm{d}x \tag{7}$$

Com $1 \leq i, j \leq N$.

Após resolver a Equação 5 para os coeficientes α_i, utilizamos a Equação 4 para obter a aproximação de Galerkin para o problema. Todo esse processo está descrito em Becker e etc. (1981b).

Atualmente, o código do projeto utiliza esse procedimento para encontrar a primeira iteração da aproximação da solução e, após o treinamento da rede, a ideia era que os nós se deslocassem para posições mais convenientes e as matrizes K e F fossem recalculadas considerando as novas funções de forma dos nós. Porém, estão ocorrendo problemas com a implementação do código com a biblioteca `PyTorch` e os nós ainda não se delocam.

Algo interessante é que há infinitas possibilidades de escolhas para a função utilizada como ϕ_i, o que torna esse método bastante flexível. Até então, o modelo apenas trabalha com uma função de forma linear, mas há trabalho para futuramente adicionar também a opção de aproximações quadráticas.

A função de forma linear é definida de acordo com a Equação 8, considerando h_i o tamanho de cada elemento finito, que agora pode ser variável.

$$\phi_i(x) = \begin{cases} \dfrac{x - x_i}{h_i}, & \text{para } x_{i-1} \leq x \leq x_i \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & \text{para } x_i \leq x \leq x_{i+1} \\ 0, & \text{para } x \leq x_{i-1} \text{e } x \geq x_{i+1} \end{cases} \tag{8}$$

3.3 Redes Neurais de Aprendizado Profundo Hierárquicas (HiDeNN)

3.4 Cálculo da perda para treinamento do modelo

4 Atividades

5 Resultados

6 Próximas Etapas

- consertar treinamento
- utilizar a rede tbm para recalcular os displacements
- readequar o codigo para rodar a partir da biblioteca FEniCSx
- utilizar funcoes de forma quadraticas tbm

Referências

BECKER, E. B.; etc. **Finite Elements: v. 1: An Introduction**. Harlow, England: Longman Higher Education, 1981.

_____. _____. Harlow, England: Longman Higher Education, 1981. 40-59 p.

LANGTANGEN, H. P.; LINGE, S. **Finite difference computing with PDEs: A modern software approach**. 1. ed. Basel, Switzerland: Springer International Publishing, 2017.

LE-DUC, T.; NGUYEN-XUAN, H.; LEE, J. A finite-element-informed neural network for parametric simulation in structural mechanics. **Finite Elem. Anal. Des.**, v. 217, n. 103904, p. 103904, 2023.

MEETHAL, R. E.; KODAKKAL, A.; KHALIL, M.; GHANTASALA, A.; OBST, B.; BLETZINGER, K.-U.; WÜCHNER, R. Finite element method-enhanced neural network for forward and inverse problems. **Adv. Model. Simul. Eng. Sci.**, v. 10, n. 1, 2023.

PANTIDIS, P.; MOBASHER, M. E. Integrated finite element neural network (I-FENN) for non-local continuum damage mechanics. **Comput. Methods Appl. Mech. Eng.**, v. 404, n. 115766, p. 115766, 2023.

SAHA, S.; GAN, Z.; CHENG, L.; GAO, J.; KAFKA, O. L.; XIE, X.; LI, H.; TAJDARI, M.; KIM, H. A.; LIU, W. K. Hierarchical deep learning neural network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering. **Comput. Methods Appl. Mech. Eng.**, v. 373, n. 113452, p. 113452, 2021.

SHARMA, P.; CHUNG, W. T.; AKOUSH, B.; IHME, M. A review of physics-informed machine learning in fluid mechanics. **Energies**, v. 16, n. 5, p. 2343, 2023.

SHI, Z.; GULGEC, N. S.; BERAHAS, A. S.; PAKZAD, S. N.; TAKAC, M. Finite difference neural networks: Fast prediction of partial differential equations. In: **2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)**. [S.l.]: IEEE, 2020. p. 130–135.

THOMÉE, V. The finite difference versus the finite element method for the solution of boundary value problems. **Bull. Aust. Math. Soc.**, v. 29, n. 2, p. 267–288, 1984.

TU, S. N. T.; NGUYEN, T. FinNet: Solving time-independent differential equations with finite difference neural network. 2022.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. Dive into deep learning. p. 170–186, 2021.

ZHANG, L.; CHENG, L.; LI, H.; GAO, J.; YU, C.; DOMEL, R.; YANG, Y.; TANG, S.; LIU, W. K. Hierarchical deep-learning neural networks: finite elements and beyond. **Comput. Mech.**, v. 67, n. 1, p. 207–230, 2021.