Uma comparação entre métodos baseados em images ou parâmetros escalares para a seleção de precondicionadores

Michael Souza[1], UFC, Fortaleza, CE, Luiz Mariano Carvalho[2], UERJ, Rio de Janeiro, RJ, Douglas Augusto[3], FOC, Rio de Janeiro, RJ , Jairo Panetta[4], ITA, São José dos Campos, SP, Paulo Goldfeld[5], UFRJ, Rio de Janeiro, RJ, José R. P. Rodrigues[6], CENPES/Petrobras, Rio de Janeiro, RJ

Resumo. Em computação de alto desempenho (HPC), a solução eficiente de grandes sistemas lineares esparsos é fundamental, sendo os métodos iterativos a escolha predominante. No entanto, a performance desses métodos está ligada ao precondicionador escolhido. A natureza multifacetada das matrizes esparsas torna difícil a prescrição universal de precondicionadores. Avançando em metodologias anteriores, esta pesquisa apresenta a representação da esparsidade de uma matrizes por meio de imagens RGB. Utilizando uma rede neural convolucional (CNN), a tarefa de seleção do precondicionador se transforma em um problema de classificação multi-classe. Testes extensivos em 126 matrizes da coleção SuiteSparse enfatizam a adequação do modelo CNN, observando um aumento de 32% na acurácia e uma redução de 25% no tempo de execução.

Palavras-chave. computação de alto desempenho (HPC), sistemas lineares esparsos, métodos iterativos, escolha de precondicionadores, rede neural convolucional, classificação multi-classe

1 Introdução

As simulações numéricas em, por exemplo, engenharia de reservatórios exigem a solução de grandes sistemas lineares com matrizes esparsas, ocupando em muitos casos mais de 50% do tempo de computação [16, 17]. Os métodos de Krylov são os solvers lineares preferidos nesse contexto. No entanto, seu sucesso depende da escolha de um precondicionador adequado, o que ainda é uma tarefa desafiadora, geralmente baseada em tentativa e erro ou na experiência do usuário [29, 32].

Um precondicionador é uma matriz ou operador que modifica o sistema original para acelerar a convergência de solvers lineares iterativos. Vários fatores influenciam a eficácia do precondicionador, incluindo, entre outros, as diversas propriedades - estruturais ou matemáticas - da matriz esparsa, a arquitetura computacional, as estruturas de dados empregadas [3, 4]. Neste trabalho, abordaremos apenas o primeiro desses fatores.

Propor uma representação adequada para matrizes esparsas quando da sugestão de precondicionadores é um desafio, principalmente devido aos requisitos de escalabilidade. Para sistemas esparsos grandes, o ideal é que a complexidade computacional seja linear em relação ao número de elementos não nulos. Essa restrição decorre da necessidade de manter a eficiência à medida que o tamanho do sistema aumenta, o que é comum em, por exemplo, simulações de reservatórios ou em dinâmica de fluidos computacional. A dificuldade em obter representações compactas está em

[1]michael@ufc.br
[2]luizmc@ime.uerj.br
[3]daa@fiocruz.br
[4]jairo.panetta@gmail.com
[5]goldfeld@matematica.ufrj.br
[6]jrprodrigues@petrobras.com.br

capturar as propriedades da matriz que mais influenciam o desempenho dos precondicionadores dentro dessa restrição de complexidade linear.

Entre os atributos da matriz que influenciam a escolha dos precondicionadores, podemos destacar a ordem da matriz, seus autovalores e valores singulares, o número de condicionamento, o padrão de esparsidade, a densidade e a dominância diagonal, entre outros. Com a exceção do padrão de esparsidade, esses atributos são numéricos. A esparsidade encapsula atributos topológicos sobre a conexão entre os elementos não nulos da matriz. Embora alguns atributos escalares, como a largura de banda e o número de elementos diferentes de zero, forneçam informações sobre padrões complexos de esparsidade, sua sensibilidade é baixa. Essa característica dificulta a obtenção de uma representação numérica adequada para a esparsidade.

Embora não seja o único fator determinante, o padrão de esparsidade afeta o nível de paralelismo possível tanto na aplicação quanto na construção de precondicionadores como ILU(k) [24, 29]. No método multigrid algébrico (AMG), os operadores grosseiros codificam a esparsidade e os valores numéricos para criar uma aproximação multinível do sistema [35]. Dependendo do padrão de esparsidade, até mesmo métodos diretos podem ser aplicáveis para resolver problemas esparsos [11].

Neste trabalho, exploramos técnicas de aprendizagem profunda (AP) para obter automaticamente representações compactas de matrizes esparsas para selecionar precondicionadores. Ampliando a abordagem de Yamada et al. [38], usamos imagens RGB para codificar espacialmente os padrões de esparsidade, facilitando a análise orientada por dados para aumentar a eficiência dos solvers lineares. Diferentemente do trabalho de Yamada, nossa metodologia incorpora a classificação de vários rótulos para identificar uma gama de precondicionadores adequados para uma determinada matriz esparsa. Para processar as representações baseadas em imagens, empregamos uma rede neural convolucional (CNN) [21].

Nossas contribuições são as seguintes:

Modelo multi-rótulo: Em tarefas de classificação convencionais, uma matriz esparsa é frequentemente mapeada para apenas um precondicionador. Entretanto, em nosso conjunto de dados, aproximadamente 20% das matrizes exibem vários precondicionadores ideais. Com base na pesquisa de Yamada et al. [38], nós não seguimos a abordagem de mapeamento de um para um. Ao adotar uma estrutura com vários rótulos, permitimos atribuir vários precondicionadores a uma única matriz, lidando melhor com casos em que vários precondicionadores têm desempenho semelhante.

Modelos escalares versus modelos baseados em imagens: Comparamos modelos escalares e com alguns baseados em imagens. Nossa pesquisa também introduziu um modelo misto que combina os dois atributos, transformando os atributos baseados em imagem em um formato vetorial (**achatamento**) e, posteriormente, integrando-os à tabela de atributos escalares.

Resultados promissores: Em nossa pesquisa inicial, os modelos baseados em imagens superam os modelos baseados em escalas na seleção do precondicionador. Eles mostram uma probabilidade 32% maior de sugestão de precondicionador ideal e uma chance 26% maior de baixo impacto computacional quando não se escolhe o precondicionador ideal. Esses resultados destacam a eficiência e a eficácia superiores dos modelos baseados em imagens.

Esta contribuição resume um trabalho em desenvolvimento que se encontra disponível em [34]; vários detalhes omitidos aqui, dado o limite do número de páginas, podem ser encontrados neste trabalho. O restante deste artigo está assim estruturado: a Seção 2 analisa a literatura relevante, a Seção 3 descreve nossa metodologia, a Seção 4 discute os resultados empíricos e a Seção 5 conclui o artigo, destacando sua importância e sugerindo futuras direções de pesquisa.

2 Trabalhos relacionados

O potencial da aprendizagem profunda (AP) para discernir padrões complexos e facilitar a tomada de decisões orientada por dados foi reconhecido como uma solução eficaz para vários desafios na computação de alto desempenho aplicada à solução de sistemas lineares [14, 25, 37].

Uma área de aplicação da AP é a seleção automática da estrutura de dados para o armazenamento de matrizes esparsas. Sedaghati et al. [33] usaram algoritmos de árvore de decisão para automatizar a seleção do formato de armazenamento com base nas propriedades da matriz. Nisa et al. [26] aplicaram técnicas de aprendizado de máquina para prever os formatos de armazenamento mais adequados para GPUs. A importância de sincronizar a estrutura de dados de armazenamento com a eficiência computacional também é destacada na pesquisa de Barreda et al. [2] e Cui et al. [8] que exploram melhorias de desempenho em diferentes plataformas de computação.

Outra aplicação de AP notável envolve (**autoajuste**) de solvers lineares. Por exemplo, Peairs e Chen [27] utilizaram uma estratégia de aprendizagem por reforço para determinar os parâmetros de reinício ideais para o GMRES. Bhowmick et al. [5] aplicaram a AP para selecionar os melhores solvers para sistemas lineares esparsos em tempo de execução, adaptando-se aos dados e à arquitetura computacional disponível. Dufrechou et al. [13] empregaram técnicas de AP para prever o solver ideal para um sistema linear específico, concentrando-se em situações em que um número limitado de sistemas triangulares é resolvido para a mesma matriz. Em outra abordagem, Funk et al. [15] apresentaram uma técnica de aprendizagem profunda para identificar o solver iterativo ideal para um determinado sistema linear esparso, obtendo uma acurácia de classificação de 60%.

Uma tendência crescente na AP é o uso de redes neurais para acelerar os operações de álgebra linear. Cui et al. [8] empregaram um sistema de AP para prever a melhor implementação da multiplicação matriz-vetor (SpMV) para uma determinada matriz. Götz e Anzt [19] introduziram uma rede neural convolucional (CNN) para detectar estruturas de blocos em padrões de esparsidade de matriz. Em uma abordagem diferente, Ackmann et al. [1] propuseram o uso de uma rede neural feed-forward como precondicionador. Taghibakhshi et al. [36] introduziram um método inovador usando um agente de aprendizagem por reforço baseado em redes neurais de grafos (GNNs) para construir espaços grosseiros em uma abordagem multigrid.

Embora a AP tenha otimizado substancialmente vários aspectos da álgebra linear computacional, seu potencial na seleção de precondicionadores merece ser mais explorado. Este trabalho é uma contribuição nessa direção.

3 Metodologia

Propomos codificar a esparsidade da matriz e alguns atributos facilmente computáveis como imagens RGB. Essa representação geométrica é uma descrição compacta e adequada da estrutura da matriz subjacente. Para comprovar essa afirmação, realizamos experimentos comparando modelos de AP treinados em atributos de matriz escalar com aqueles treinados em imagens. Em ambos os cenários, o objetivo principal é a seleção ideal de precondicionadores para minimizar o tempo de convergência dos solucionadores lineares.

Para este estudo, empregamos 126 matrizes simétricas, não diagonais e positivo-definidas da coleção de matrizes SuiteSparse usando a biblioteca `ssgetpy` [20, 28].

Empregamos um conjunto de parâmetros escalares para caracterizar as matrizes, ver Tabela 1. Entre esses parâmetros, o número da condicionamento foi calculado usando a função `condest` do MATLAB, enquanto os menores e maiores autovalores foram obtidos por meio da função `eigs` com as opções `smallestabs` e `largestabs`, respectivamente [23]. Embora o número de condicionamento e os autovalores sejam informativos, eles são computacionalmente caros. Em contrapartida, nossa abordagem baseada em imagens oferece uma alternativa igualmente informativa, porém mais

Tabela 1: Propriedades Escalares.

Nome	**Definição**
Densidade	Número de elementos diferentes de zero dividido pelo número total de elementos.
N	Número de linhas.
NNZ	Número de elementos diferentes de zero.
RowNNZ	Número médio de elementos diferentes de zero por linha.
Condest	Estimativa do número de condicionamento da matriz, com a norma 1.
MinEigs	Estimativa do menor autovalor da matriz.
MaxEigs	Estimativa do maior autovalor da matriz.
DDom	Percentagem das linhas dominadas diagonalmente. Se D for a matriz diagonal extraída de A, $B = A - D$, $S = \{i : \lvert D_{ii}\rvert > \sum_j \lvert B_{ij}\rvert\}$, então DDom $= \frac{\lvert S\rvert}{N}$.
DDeg	Razão mínima entre o valor absoluto do elemento diagonal e a soma das entradas não diagonais. DDeg $= \min_i \frac{\lvert D_{ii}\rvert}{\sum_j \lvert B_{ij}\rvert}$.

Tabela 2: Propriedades das Matrizes.

	N	NNZ	Densidade	RowNNZ	MaxEigs	MinEigs	Condest	DDom	DDeg
média	4.1E+04	6.1E+05	1.1E-02	2.5E+01	1.9E+13	9.6E+07	1.2E+17	1.4E-01	2.0E-01
std	1.2E+05	1.2E+06	1.8E-02	3.8E+01	1.3E+14	7.3E+08	9.1E+17	3.1E-01	3.6E-01
mín	1.0E+02	5.9E+02	5.0E-06	2.9E+00	9.8E-13	-3.6E+00	3.6E+00	0.00	7.9E-15
25%	1.1E+03	1.9E+04	3.8E-04	6.8E+00	9.7E+00	1.5E-04	4.5E+03	0.00	8.5E-04
50%	8.6E+03	1.3E+05	3.8E-03	1.6E+01	2.9E+04	2.0E-01	3.9E+06	0.00	4.8E-03
75%	2.9E+04	5.6E+05	1.3E-02	2.6E+01	5.7E+08	2.8E+00	1.5E+09	2.6E-03	2.2E-01
máx	1.0E+06	5.5E+06	9.1E-02	3.6E+02	1.1E+15	5.8E+09	9.4E+18	1.0E+00	1.8E+00

eficiente.

A Tabela 2 apresenta um resumo estatístico de propriedades escalares das matrizes analisadas. Apesar de todas as matrizes serem simétricas e positivo-definidas (SPD), o menor valor próprio estimado é $-3,60$. Essa contradição decorre da acurácia limitada da função `eigs`, que pode não calcular com corretamente o menor autovalor para matrizes específicas. Embora o número de condicionamento de uma matriz SPD seja definido por seus autovalores extremos, os atributos que empregamos são estimativas, o que os torna não redundantes.

Para representar visualmente as propriedade da matriz, criamos um conjunto de dados de imagem com base no método proposto por Yamada et al. [38]. Cada matriz A é particionada em blocos A_{ij} de dimensão $b \approx N/m$, em que N é a dimensão da matriz e m é a resolução da imagem. Esses blocos são representados como pixels p_{ij} em uma imagem de $m \times m$ pixels. Geramos quatro conjuntos de imagens com $m \in \{32, 64, 128, 256\}$.

Cada pixel p_{ij} consiste em três componentes alinhados com os canais RGB: vermelho, verde e azul. O canal vermelho captura a magnitude dos elementos diferentes de zero no bloco da matriz, o azul incorpora as dimensões da matriz e o verde transmite a densidade do bloco. Como resultado, as características distintas de cada matriz são visualmente representadas, com os atributos de cada bloco influenciando a cor do pixel correspondente.

Denotaremos os canais vermelho, verde e azul do pixel p_{ij} como R_{ij}, G_{ij} e B_{ij}, respectivamente. O intervalo dos componentes do pixel é limitado de 0 a 255. Essa restrição e a necessidade de uma

representação geral de matrizes heterogêneas exigem um esquema de codificação cuidadoso. Agora descrevemos a transformação da matriz em uma imagem:

Canal Azul: O canal azul tem o mesmo valor para todos os pixels, dado por

$$B_{ij} = \left\lfloor \frac{N - N_{\min}}{N_{\max} - N_{\min}} \times 255 \right\rfloor,\ i, j = 1, \ldots, m,$$

em que N é a ordem da matriz, $N_{\min}$ e $N_{\max}$ são as ordens mínima e máxima de todas as matrizes no conjunto de dados, respectivamente.

Canal Verde: O valor do canal verde, G_{ij}, é a densidade de elementos diferentes de zero em um determinado bloco de matriz. Ele é definido como:

$$G_{ij} = \left\lfloor \frac{NNZ_{ij}}{b^2} \times 255 \right\rfloor$$

em que NNZ_{ij} representa a contagem de elementos diferentes de zero no bloco A_{ij} e b denota a ordem do bloco.

Canal Vermelho: Para representar a magnitude média dos elementos diferentes de zero nos blocos de uma matriz esparsa, primeiro ajustamos ou "polarizamos" seus elementos diferentes de zero. Isso é feito para garantir que todos os valores sejam maiores que zero. Especificamente, para qualquer elemento diferente de zero a na matriz A, seu valor enviesado $v(a)$ é dado por $v(a) = a - \min(A) + 1$, em que $\min(A)$ é o elemento mínimo de A.

A próxima etapa é calcular a média desses valores enviesados para cada bloco da matriz. Vamos chamar essa média de γ_{ij} para o bloco A_{ij}. Se a diferença entre os valores máximo e mínimo de toda a matriz for igual ou inferior a 255, calcularemos a média dos valores enviesados dentro do bloco. Se a diferença for maior que 255, calculamos o logaritmo de base 2 dos valores enviesados antes de calcular a média. Esse método lida com eficiência com blocos de valores amplamente variáveis, mudando o foco da magnitude para a ordem de magnitude.

A fórmula é a seguinte:

$$\gamma_{ij} = \frac{\sum_{a \in A_{ij}} v(a)}{NNZ_{ij}},$$

quando a diferença entre os valores máximo e mínimo for igual ou inferior a 255, e

$$\gamma_{ij} = \frac{\sum_{a \in A_{ij}} \log_2 v(a)}{NNZ_{ij}},$$

caso contrário.

Por fim, com as médias de bloco em mãos, podemos definir o valor R_{ij} como a normalização da média de bloco com relação ao intervalo geral de médias de bloco, ou seja

$$R_{ij} = \left\lfloor \frac{\gamma_{ij} - \min(\gamma)}{\max(\gamma) - \min(\gamma)} \times 255 \right\rfloor$$

A Figura 1 ilustra a conversão de uma matriz de 20×20 em uma imagem de 5×5 pixels, com $m = 5$. Aqui, cada pixel na imagem corresponde a um bloco de 4×4 da matriz. O conjunto completo de dados de imagem pode ser acessado e baixado do repositório do GitHub [**ImagePrecGitHub**].

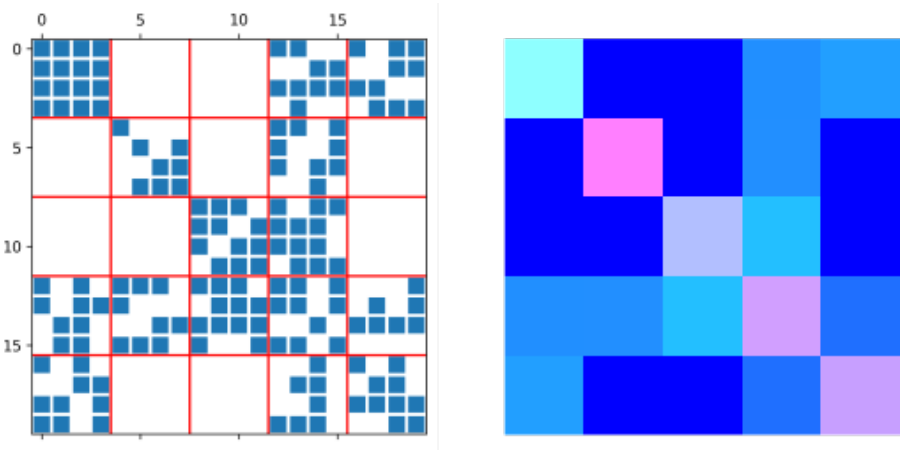

Figura 1: Representação baseada em imagem de uma matriz de 20×20. A matriz é dividida em blocos 5×5 e os valores das propriedades são codificados nos canais de cores RGB
. entradas

A representação da esparsidade como imagens captura naturalmente as informações topológicas, codificando-as em relações geométricas, que são mais difíceis de serem obtidas com representações escalares. Para destacar a importância dessas nuances geométricas nas representações de imagens, geramos um banco de dados misto que combina atributos escalares e imagens. Nessa variação, as imagens RGB de $m \times m$ pixels foram colocadas em vetores com $3m^2$ entradas. Concatenamos os vetores achatados com os atributos escalares, combinando assim as duas representações.

Esse processo gerou quatro conjuntos de atributos escalares estendidos, correspondentes a cada $m \in \{32, 64, 128, 256\}$. Para gerenciar o aumento da dimensionalidade e reter os aspectos mais informativos desses atributos estendidos, usamos a análise de componentes principais (PCA), capturando 99% da variação nos dados [**guyon2006introduction**].

4 Resultados

próprios

5 Conclusões

Equações inseridas no trabalho completo devem ser enumeradas sequencialmente e à direita no texto, por exemplo

$$\frac{\partial u}{\partial t} - \Delta u = f, \quad \text{em } \Omega. \tag{1}$$

Consulte o arquivo `.tex` para mais detalhes sobre o código-fonte gerador da equação (1).

6 Tabelas e Figuras

As(os) autoras(es) podem inserir figuras e tabelas no artigo. Elas devem estar dispostas próximas de suas referências no texto.

6.1 Inserção de Tabelas

A inserção de tabela deve ser feita com o ambiente `table`, sendo enumerada, disposta horizontalmente centralizada, próxima de sua referência no texto, e a legenda imediatamente acima dela. Por exemplo, consulte a Tabela 3.

Tabela 3: Categorias dos trabalhos.

Categoria do trabalho	Número de páginas	Tipo do trabalho
1	2	A, B e C
2	entre 5 e 7	apenas C

6.2 Inserção de Figuras

A inserção de figura deve ser feita com o ambiente `figure`, ela deve estar enumerada, disposta horizontalmente centralizada, próxima de sua referência no texto, e legenda imediatamente abaixo dela. **Quando não própria, deve-se indicar/referências a fonte.** Por exemplo, consulte a Figura 2.

Figura 2: Exemplo de imagem. Fonte: indicar.

7 Sobre as Referências Bibliográficas

As referências bibliográficas devem ser inseridas conforme especificado neste padrão, sendo que serão automaticamente geradas em ordem alfabética pelo sobrenome do primeiro autor. Este *template* fornece suporte para a inserção de referências bibliográficas com o BibLaTeX. Os dados de cada referência do trabalho devem ser adicionados no arquivo `refs.bib` e a indicação da referência no texto deve ser inserida com o comando `\cite`. Seguem alguns exemplos de referências: livro [6], artigos publicados em periódicos [7, 9], capítulo de livro [10], dissertação de mestrado [12], tese de doutorado [22], livro publicado dentro de uma série [18], trabalho publicado em anais de eventos [30], *website* e outros [31]. Por padrão, os nomes de todos os autores da obra citada aparecem na bibliografia. Para obras com mais de três autores, é também possível indicar apenas o nome do primeiro autor, seguido da expressão et al. Para implementar essa alternativa, basta remover "`,maxnames=50`" do comando correspondente do código-fonte. Sempre que disponível forneça o DOI, ISBN ou ISSN, conforme o caso.

8 Considerações Finais

Esta seção é reservada às principais conclusões e considerações finais do trabalho.

Agradecimentos (opcional)

Seção reservada aos agradecimentos dos autores, caso for pertinente. Por exemplo, agradecimento a fomentos. Se os autores optarem pela inclusão de Agradecimentos, a palavra "(opcional)" deve ser removida do título da seção. Esta seção não é numerada e deve ser disposta entre a última seção do corpo do texto e as Referências.

Referências

[1] Jan Ackmann et al. **Machine-Learned Preconditioners for Linear Solvers in Geophysical Fluid Flows**. 2020. arXiv: 2010.02866 [physics.ao-ph].

[2] Maria Barreda et al. "Performance modeling of the sparse matrix–vector product via convolutional neural networks". Em: **The Journal of Supercomputing** 76.11 (2020), pp. 8883–8900. URL: doi.org/10.1007/s11227-020-03186-1.

[3] Nathan Bell e Michael Garland. **Efficient sparse matrix-vector multiplication on CUDA**. Rel. técn. Nvidia Technical Report NVR-2008-004, Nvidia Corporation, 2008.

[4] Michele Benzi. "Preconditioning Techniques for Large Linear Systems: A Survey". Em: **Journal of Computational Physics** 182.2 (2002), pp. 418–477. DOI: 10.1006/jcph.2002.7176.

[5] Sanjukta Bhowmick et al. **Application of Machine Learning in Selecting Sparse Linear Solvers**. 2006.

[6] J. L. Boldrini et al. **Álgebra Linear**. 3a. ed. São Paulo: Harbra, 1986. ISBN: 9788529402024.

[7] L. O. Contiero et al. "Rainbow Erdös–Rothschild Problem for the Fano Plane". Em: **SIAM Journal on Discrete Mathematics** (2021). Aceito. DOI: 10.1137/20M136325X.

[8] Hang Cui et al. "A code selection mechanism using deep learning". Em: **2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)**. IEEE. 2016, pp. 385–392. DOI: 10.1109/MCSoC.2016.46.

[9] J. A. Cuminato e V. Ruas. "Unification of distance inequalities for linear variational problems". Em: **Computational and Applied Mathematics** 34 (2014), pp. 1009–1033. DOI: 10.1007/s40314-014-0163-6.

[10] P. L. Da Silva e I. L. Freire. "On the group analysis of a modified Novikov equation". Em: **Interdisciplinary Topics in Applied Mathematics, Modeling and Computational Science, Springer Proceedings in Mathematics and Statistics**. Ed. por M. Cojocaru et al. Vol. 117. Springer, 2015. Cap. 23, pp. 161–166. DOI: 10.1007/978-3-319-12307-3_23.

[11] Timothy A. Davis, Sivasankaran Rajamanickam e Wissam M. Sid-Lakhdar. "A survey of direct methods for sparse linear systems". Em: **Acta Numerica** (2016), pp. 383–566. DOI: 10.1017/S0962492916000076.

[12] G. L. Diniz. "A mudança no habitat de populações de peixes: de rio a represa - o modelo matemático". Dissertação de mestrado. Unicamp, 1994.

[13] Ernesto Dufrechou, Pablo Ezzatti e Enrique S. Quintana-Ortí. "Automatic selection of sparse triangular linear system solvers on GPUs through machine learning techniques". Em: **2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)**. IEEE. 2019, pp. 41–47. DOI: 10.1109/SBAC-PAD.2019.00020.

[14] Thomas L. Falch e Anne C. Elster. "Machine learning-based auto-tuning for enhanced performance portability of OpenCL applications". Em: **Concurrency and Computation: Practice and Experience** 29.8 (2017). e4029 cpe.4029, e4029. DOI: 10.1002/cpe.4029. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4029.

[15] Yannick Funk, Markus Götz e Hartwig Anzt. "Prediction of optimal solvers for sparse linear systems using deep learning". Em: **Proceedings of the 2022 SIAM Conference on Parallel Processing for Scientific Computing**. SIAM. 2022, pp. 14–24. DOI: 10.1137/1.9781611977141.2.

[16] Vassilis Gaganis e Nikos Varotsis. "Machine Learning Methods to Speed up Compositional Reservoir Simulation". Em: **SPE Europec featured at EAGE Conference and Exhibition**. Jun. de 2012, SPE-154505–MS. DOI: 10.2118/154505-MS. eprint: https://onepetro.org/SPEEURO/proceedings-pdf/12EURO/All-12EURO/SPE-154505-MS/1612365/spe-154505-ms.pdf.

[17] Leonardo Gasparini et al. "Hybrid parallel iterative sparse linear solver framework for reservoir geomechanical and flow simulation". Em: **Journal of Computational Science** 51 (2021), p. 101330. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2021.101330.

[18] L. T. Gomes, L. C. Barros e B. Bede. **Fuzzy differential equation in various approaches**. Springer Briefs in Mathematics. SBMAC - Springer, 2015. ISBN: 978-3-319-22575-3.

[19] Markus Götz e Hartwig Anzt. "Machine learning-aided numerical linear algebra: Convolutional neural networks for the efficient preconditioner generation". Em: **2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (scalA)**. IEEE. 2018, pp. 49–56. DOI: 10.1109/ScalA.2018.00010. URL: https://doi.ieeecomputersociety.org/10.1109/ScalA.2018.00010.

[20] Scott P. Kolodziej et al. "The SuiteSparse Matrix Collection Website Interface". Em: **Journal of Open Source Software** 4.35 (2019), p. 1244. DOI: 10.21105/joss.01244.

[21] Zewen Li et al. "A survey of convolutional neural networks: analysis, applications, and prospects". Em: **IEEE Transactions on neural networks and learning systems** 33.12 (2022), pp. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827.

[22] S. M. Mallet. "Análise Numérica de Elementos Finitos". Tese de doutorado. LNCC/MCTI, 1990.

[23] MathWorks. **MATLAB**. Version 9.14.0.2254940 (R2023a). 2023.

[24] Jan A. Meijerink e Henk A. van der Vorst. "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix". Em: **Mathematics of Computation** 31 (1977), pp. 148–162.

[25] Suejb Memeti et al. "Using meta-heuristics and machine learning for software optimization of parallel computing systems: a systematic literature review". Em: **Computing** 101.8 (2019), pp. 893–936. URL: 10.1007/s00607-018-0614-9.

[26] Israt Nisa et al. "Effective machine learning based format selection and performance modeling for SpMV on GPUs". Em: **2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. IEEE. 2018, pp. 1056–1065. DOI: 10.1109/IPDPSW.2018.00164.

[27] Lisa Peairs e Tzu-Yi Chen. "Using reinforcement learning to vary the m in GMRES (m)". Em: **Procedia Computer Science** 4 (2011), pp. 2257–2266. DOI: 10.1016/j.procs.2011.04.246.

[28] Sudarshan Raghunathan. **SSGETPY: Search and download sparse matrices from the SuiteSparse Matrix Collection**. GitHub repository. 2023. URL: https://github.com/drdarshan/ssgetpy (acesso em 20/10/2023).

[29] Yousef Saad. **Iterative methods for sparse linear systems**. SIAM, 2003.

[30] I. L. D. Santos e G. N. Silva. "Uma classe de problemas de controle ótimo em escalas temporais". Em: **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**. 2013, pp. 010177-1–6. DOI: 10.5540/03.2013.001.01.0177.

[31] SBMAC. **Site oficial do Congresso Nacional de Matemática Aplicada**. Online. Acessado em 08/12/2021, http://www.cnmac.org.br.

[32] Jennifer Scott e Miroslav Tůma. "An Introduction to Sparse Matrices". Em: **Algorithms for Sparse Linear Systems**. Cham: Springer International Publishing, 2023, pp. 1–18. ISBN: 978-3-031-25820-6. DOI: 10.1007/978-3-031-25820-6_1.

[33] Naser Sedaghati et al. "Automatic selection of sparse matrix representation on GPUs". Em: **Proceedings of the 29th ACM on International Conference on Supercomputing**. ICS '15. Association for Computing Machinery, 2015, pp. 99–108. ISBN: 9781450335591. DOI: 10.1145/2751205.2751244.

[34] Michael Souza et al. **A Comparison of Image and Scalar-Based Approaches in Preconditioner Selection**. 2023. arXiv: 2312.15747 [math.NA]. URL: https://arxiv.org/abs/2312.15747.

[35] Klaus Stüben. "A review of algebraic multigrid". Em: **Numerical Analysis: Historical Developments in the 20th Century**. Ed. por C. Brezinski e L. Wuytack. Amsterdam: Elsevier, 2001, pp. 331–359. ISBN: 978-0-444-50617-7. DOI: 10.1016/B978-0-444-50617-7.50015-X. URL: https://www.sciencedirect.com/science/article/pii/B978044450617750015X.

[36] Ali Taghibakhshi et al. "Optimization-Based Algebraic Multigrid Coarsening Using Reinforcement Learning". Em: **Advances in Neural Information Processing Systems**. Ed. por M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 12129–12140. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/6531b32f8d02fece98ff36a64a7c8260-Paper.pdf.

[37] Ozan Tuncer et al. "Diagnosing Performance Variations in HPC Applications Using Machine Learning". Em: **High Performance Computing: 32nd International Conference, ISC High Performance 2017, Frankfurt, Germany, June 18–22, 2017, Proceedings 32**. Ed. por Julian M. Kunkel et al. Springer International Publishing, 2017, pp. 355–373. ISBN: 978-3-319-58667-0.

[38] Kenya Yamada et al. "Preconditioner auto-tuning using deep learning for sparse iterative algorithms". Em: **2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)**. IEEE. 2018, pp. 257–262. DOI: 10.1109/CANDARW.2018.00055.