REVISTA DE
ENGENHARIA CIVIL IMED

Uma aplicação *OpenMP* para implementação da estratégia Elemento por Elemento em Análise de Elementos Finitos

An OpenMP Application for programming with Element By Element Storage in Finite Elements Analysis

Guilherme Piva dos Santos(1), Eduardo Costa Couto(2), Aline Ribeiro Paliga(3), Gerson Geraldo Homrich Cavalheiro(4), Jansen Avila(5)

1 Universidade Federal do Rio Grande do Sul (UFGRS), Porto Alegre, RS, Brasil.
E-mail: guilherme.piva.santos@gmail.com | ORCID: https://orcid.org/0000-0003-3817-560X
2 Universidade Federal de Pelotas (UFPEL), Pelotas, RS, Brasil.
E-mail: e.costacouto@gmail.com | ORCID: http://orcid.org/0000-0003-3419-2576
3 Universidade Federal de Pelotas (UFPEL), Pelotas, RS, Brasil.
E-mail: aline.paliga@ufpel.edu.br | ORCID: http://orcid.org/0000-0003-0003-1133
4 Universidade Federal de Pelotas (UFPEL), Pelotas, RS, Brasil.
E-mail: gerson.cavalheiro@inf.ufpel.edu.br | ORCID: http://orcid.org/0000-0002-4314-3429
5 Universidade Federal de Pelotas (UFPEL), Pelotas, RS, Brasil.
E-mail: jrdavila@inf.ufpel.edu.br | ORCID: https://orcid.org/0000-0002-9974-9120

Revista de Engenharia Civil IMED, Passo Fundo, v. 9, n. 2, p. 54-67, julho-dezembro, 2022 - ISSN 2358-6508

[Recebido: agosto 21, 2020; Aceito: julho 20, 2023]

DOI: https://doi.org/10.18256/2358-6508.2022.v9i2.4263

Sistema de Avaliação: *Double Blind Review*

Como citar este artigo / How to cite item: clique aqui!/click here!

Resumo

Este artigo apresenta o algoritmo de um *solver* que soluciona o sistema de equações lineares gerado pelo Método dos Elementos Finitos, e avalia a sua eficiência em um código que analisa o comportamento de um maciço rochoso quando é realizada a escavação de um túnel. O algoritmo foi desenvolvido para uma plataforma com memória compartilhada, pois usa o Método dos Gradientes Conjugados Pré-Condicionados paralelizado com a interface *OpenMP*. Foi usada a estratégia de armazenamento Elemento por Elemento. Realizou-se uma comparação entre o algoritmo desenvolvido e outro que é sequencial e utiliza o Método de Eliminação de Gauss, associado à técnica de armazenagem *skyline*. Verificou-se que o algoritmo desenvolvido foi 84,07% mais rápido.

Palavras-chave: Elemento por Elemento. *OpenMP*. Implementação Paralela.

Abstract

This article aims to present the solver algorithm, which solves the system of linear equations generated by the Finite Element Method and evaluates its efficiency using a code that analyzes the behavior of a rock mass when a tunnel is excavated. The algorithm was developed for a shared memory platform as it uses the Preconditioned Conjugate Gradient Method paralleled with the OpenMP interface. It was used Element by Element Storage Strategy. A comparison was made between the developed algorithm and another that is sequential and uses the Gaussian Elimination Method, associated with the skyline storage technique. It was found that the developed algorithm was 84.07% faster.

Keywords: Element by Element. OpenMP. Parallel Programming.

1 Introdução

Diversos problemas com importância para a Engenharia podem ser modelados por meio de Equações Diferenciais Parciais. Com exceção de alguns casos particulares, não é possível chegar auma solução analítica exata para estes problemas. O Método dos Elementos Finitos (MEF) é, atualmente, o método numérico mais utilizado para obter soluções aproximadas para este tipo de problema (CHANDRUPATLA; BELEGUNDU, 2014).

O MEF gera um sistema de equações lineares cuja resolução consiste na solução aproximada do problema em estudo. A decisão de utilizar o MEF em uma simulação numérica implica em duas escolhas importantes: (i) a técnica a ser utilizada para o armazenamento do sistema de equações e (ii) o método de solução deste sistema. Estas duas escolhas são de fundamental importância, pois o sistema de equações poderá ter dimensões tais que uma técnica imprópria de armazenamento poderá ultrapassar a capacidade de memória do computador. Por outro lado, a escolha do método de solução do sistema de equações tem relação direta com o tempo de processamento.

Por exemplo, os projetos de túneis, quando baseados em uma abordagem numérica, principalmente, quando usam o elemento estrutural tirante passivo, requerem malhas muito refinadas. A abordagem numérica pode ser direta, então, serão usados elementos lineares para discretizar os tirantes e elementos tridimensionais para discretizar o maciço a ser escavado. Ou indireta, discretizando apenas o maciço e levando em consideração os tirantes passivos por meio do método da homogeneização de meios periódicos ou do método incorporado. Qualquer uma das abordagens resultará em uma grande matriz de rigidez, assim, a escolha da estratégia de armazenamento e do método de solução do sistema de equações é fundamental na minimização do custo computacional inerente a um código desenvolvido para lidar com esse problema.

Na solução do sistema de equações gerado pelo MEF, entre os métodos iterativos, destaca-se o Método dos Gradientes Conjugados Pré-Condicionados (MGCP).

Com relação às técnicas associadas aos métodos iterativos de armazenagem da matriz de rigidez, destaca-se a técnica Elemento por Elemento (EPE), que dispensa a montagem da matriz de rigidez global.

Em 1997 foi criado o *Open MultiProcessing (OpenMP*), que é uma interface baseada em *threads* (fluxos de processamento) utilizada para prover paralelismo incremental em máquinas com memória compartilhada (multiprocessadores) que tem sido bastante utilizada, sobretudo, por sua simplicidade (CHAPMAN *et al.*, 2008).

No MGCP o produto matriz-vetor e o produto interno aparecem de forma recorrente. Essas operações são facilmente paralelizáveis e isso estimulou o uso combinado do MGCP e da interface *OpenMP*, que proporciona maior velocidade na obtenção dos resultados acrescida da técnica de armazenagem EPE, que proporciona menor uso da memória.

O presente trabalho apresenta um algoritmo de fácil implementação da combinação descrita.

O objetivo deste trabalho, além de apresentar o algoritmo de um *solver*, é avaliar sua eficiência em um código que analisa os deslocamentos em um maciço rochoso quando é realizada a escavação de um túnel. Com esse fim, foram realizadas comparações de desempenho entre algoritmos sequencial e paralelo do MGCP-EPE e um algoritmo sequencial direto que utiliza o método de eliminação de Gauss associado à técnica de armazenagem *skyline*. O código usado foi implementado em FORTRAN90 e validado em Couto (2009). Para essas comparações foi simulada uma escavação em um maciço rochoso.

2 Material e Métodos

Na engenharia de estruturas o MEF é usado para formular tanto problemas de análise de estruturas reticuladas, como também de estruturas contínuas bi e tridimensionais. O método utiliza conceitos de "discretização" do contínuo e "matriz de interpolação". Este último fornece os deslocamentos em um ponto dentro do elemento, dependendo de seus deslocamentos nodais. O termo "discretização" refere-se a um modelo com um número finito de incógnitas (deslocamentos nodais) para análise do contínuo, em contraste com a análise com um número infinito de variáveis como aquelas feitas pela teoria da elasticidade, que usa funções contínuas, isto é, com infinitas incógnitas como uma solução (VAZ, 2011). O MEF gera sistemas lineares de equações da forma:

$$\underline{\underline{\boldsymbol{K}}}\,\underline{x} = \underline{b} \quad \text{Eq. 1}$$

onde $\underline{\underline{\boldsymbol{K}}}$ é uma matriz simétrica positiva definida de ordem N, $\underline{x}$ é o vetor de incógnitas e $\underline{b}$ é o vetor de valores conhecidos. Numa formulação em deslocamentos pelo MEF para análise estrutural, é a matriz de rigidez global, $\underline{\underline{\boldsymbol{K}}}$ é o vetor de deslocamentos incógnitos e $\underline{x}$ é o vetor de forças global.

O MGC é um dos métodos iterativos mais simples e efetivos usados para resolver sistemas desse tipo (BATHE, 2014).

O algoritmo desenvolvido por Hestenes e Stiefel (1952) passou por modificações para que a convergência ocorra com menor número de iterações. Com esse fim, foi desenvolvida a técnica de pré-condicionamento, cuja ideia central consiste na aplicação do MCG a um sistema transformado de equações obtido pela pré-multiplicação da Equação 1 por uma matriz ($\underline{\underline{\boldsymbol{P}}}$) na forma:

$$\underline{\underline{\boldsymbol{P}}}^{-1}\underline{\underline{\boldsymbol{K}}}\underline{x} = \underline{\underline{\boldsymbol{P}}}^{-1}\underline{b} \quad \text{Eq. 2}$$

onde $\underline{\underline{\boldsymbol{P}}}$ é uma matriz simétrica positivo definida, denominada de matriz de pré-concionamento (FU *et al.*, 2016).

A matriz simétrica positivo fundamentou o Método dos Gradientes Conjugados Pré-Condicionados (MGCP). O pré-condicionador usado nesta pesquisa é o diagonal, também denominado Jacobi, que consiste em uma matriz formada pelos termos diagonais da matriz de rigidez. O algoritmo do MGCP com pré-condicionador diagonal, baseado no algoritmo proposto por Peng *et al.* (2019), consiste nas seguintes etapas:

1 – Inicialização:

a) $k = 0$, $\underline{x}^0 = 0$, $\underline{r}^0 = b$

b) Resolver $\underline{\underline{\boldsymbol{P}}}\underline{z}^0 = \underline{r}^0$

c) $\underline{p}^0 = \underline{z}^0$

2 – Atualização dos vetores estimativa e resíduo:

d) $\alpha_k = \left(\underline{r}^{k,T}.\underline{z}^k\right)/\left(\underline{p}^{k,T}.\underline{\underline{\boldsymbol{K}}}.\underline{p}^k\right)$

e) $\underline{x}^{k+1} = \underline{x}^k + \alpha_{k.}\underline{p}^k$

f) $\underline{r}^{k+1} = \underline{r}^k - \alpha_{k.}\underline{\underline{\boldsymbol{K}}}.\underline{p}^k$

3 – Teste de Convergência:

g) se $\left\|\underline{r}^{k+1}\right\|_2/\left\|\underline{r}^0\right\|_2 \leqslant tolerância$, pare.

4 – Atualização do vetor direção de busca:

h) Resolver $\underline{\underline{\boldsymbol{P}}}\underline{z}^{k+1} = \underline{r}^{k+1}$

i) $\beta_k = \left(\underline{r}^{k+1,T}.\underline{z}^{k+1}\right)/\left(\underline{r}^{k,T}.\underline{z}^k\right)$

j) $\underline{p}^{k+1} = \underline{z}^k + \beta_{k.}\underline{p}^k$

k) $k = k + 1$

l) Retornar à etapa (b).

Nesse algoritmo, duas operações têm relevância destacada sobre as demais: a solução do sistema $\underline{\underline{\boldsymbol{P}}}\underline{z}^0 = \underline{r}^0$ e o produto matriz vetor $\underline{\underline{\boldsymbol{K}}}\underline{p}^k$. Nessas duas operações aparecem matrizes globais, a de rigidez e a pré-condicionadora, que é montada a partir da matriz de rigidez. O caminho escolhido para chegar ao resultado destas operações poderá envolver a montagem das matrizes globais ou não. Isso define o método de armazenagem a ser usado.

O pré-condicionador, ou matriz de pré-condicionamento, usado nesta pesquisa é o de Jacobi, ou diagonal, assim definido:

$$\underline{\underline{\boldsymbol{P}}} = diag\left(\underline{\underline{\boldsymbol{K}}}\right) \quad \text{Eq. 3}$$

isto é, constituído dos termos diagonais de $\underline{\underline{\boldsymbol{K}}}$ (PENG *et al.*, 2019). Esta escolha foi baseada em dois motivos: o baixo consumo de memória deste pré-condicionador,

tendo em vista que os termos não diagonais de $\underline{\underline{\boldsymbol{P}}}$ valem zero; a facilidade de resolver $\underline{\underline{\boldsymbol{P}}}\underline{z}^0 = \underline{r}^0$, já que $\underline{\underline{\boldsymbol{P}}}$, neste caso, é uma matriz diagonal e, como tal, pode facilmente ser invertida.

Neste trabalho não é realizada a montagem da matriz de rigidez, portanto, o método de armazenagem é o elemento por elemento (EPE). Então, o produto $\underline{\underline{\boldsymbol{K}}}\underline{p}^k$ é realizado em nível de elemento (LANG; RÜGER, 2013):

$$\underline{\underline{\boldsymbol{K}}}\underline{p}^k = \sum_{e=1}^{nel} \underline{\underline{\boldsymbol{K}}}^{(e)} \underline{p}^{k(e)} \quad \text{Eq. 4}$$

onde nel é o número de elementos, $\underline{\underline{\boldsymbol{K}}}^{(e)}$ é a matriz de rigidez de um elemento e $\underline{p}^{k(e)}$ é o vetor local obtido de $\underline{p}^k$.

A sub-rotina desenvolvida para resolver o produto $\underline{\underline{\boldsymbol{K}}}\underline{p}^k$ é paralelizada com a interface *OpenMP*. No trecho paralelo, nas *threads*, são gerados os vetores elementares $\underline{kp}^{(e)}$, que são expandidos para dimensões globais. Ao final da execução das *threads*, seria desejável que esses vetores fossem acumulados gerando o vetor global $\underline{Kp}^{(g)}$. No entanto, o *OpenMP* não oferece esse recurso para variáveis vetoriais ou matriciais. Então, os vetores são acumulados em colunas da matriz $\underline{\underline{\boldsymbol{kp}}}^{(e)}$, onde cada coluna corresponde a uma *thread*. Fora do trecho paralelo, as colunas são somadas dando origem ao vetor $\underline{Kp}^{(g)}$, resultado procurado do produto $\underline{\underline{\boldsymbol{K}}}\underline{p}^{(k)}$. O algoritmo da sub-rotina que realiza essas operações é apresentado a seguir:

1 – Inicialização:

a) $\underline{\underline{K}}^{(e)} = 0$, $\underline{p}^{k(e)} = 0$, $\underline{kp}^{(e)} = 0$, $\underline{\underline{kp}}^{(g)} = 0$, $\underline{Kp}^{(g)} = 0$

2 – Iniciar trecho paralelo:

a) Fazer $el = 1, nel$, onde nel é o número de elementos

b) Montar $\underline{\underline{\boldsymbol{K}}}^{(e)}$

c) Montar $\underline{p}^{k(e)}$

d) Aplicar condições de contorno em $\underline{\underline{\boldsymbol{K}}}^{(e)}$

e) Aplicar condições de contorno em $\underline{p}^{k(e)}$

f) Gerar o vetor $\underline{kp}^{(e)}$, produto de $\underline{\underline{\boldsymbol{K}}}^{(e)}$ e $\underline{p}^{k(e)}$ com condições de contorno

g) Passar $\underline{kp}^{(e)}$ de local para global e acumular em uma coluna da matriz global $\underline{\underline{\boldsymbol{kp}}}^{(g)}$ correspondente à thread em uso

h) Fim do fazer.

3 – Fim do trecho paralelo.

4 – Transformar a matriz global $\underline{\underline{\boldsymbol{kp}}}^{(g)}$ no vetor global $\underline{Kp}^{(g)}$ pela soma termos de uma mesma linha.

A metodologia utilizada no desenvolvimento desta pesquisa visa, além de apresentar o algoritmo de um *solver*, avaliar sua eficiência em um código que analisa

os deslocamentos em um maciço rochoso quando é realizada a escavação de um túnel. Para tanto, foram desenvolvidas sub-rotinas baseadas nos algoritmos apresentados no item anterior que resolvem o sistema de equações lineares gerado pelo MEF. Conforme já exposto, essas sub-rotinas usam o MGCP para resolver o sistema de equações armazenado com a técnica EPE.

Para reduzir o custo computacional desse *solver*, foi usada a interface *OpenMP*. Tanto o código original quanto as sub-rotinas desenvolvidas foram implementados em FORTRAN90. O código original foi validado em COUTO (2009).

O equipamento utilizado tem dois processadores Intel(R) Xeon(R) 2,20 GHz, com 20 *threads* cada, e 32 GB de memória RAM.

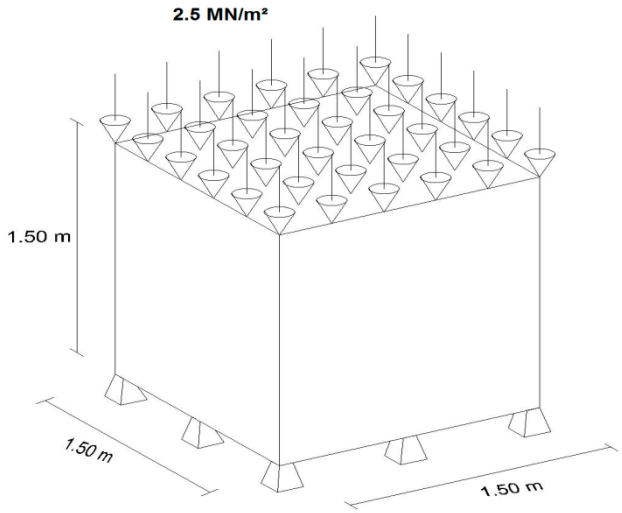
3 Resultados e Discussão

O código baseado no MEF, onde foi inserido o novo *solver*, já foi validado em COUTO (2009) com um *solver* que usa um método direto. Então, primeiro avaliou-se a precisão e a eficiência computacional do *solver* desenvolvido por meio de um problema simples de Mecânica dos Sólidos que possui solução analítica. Depois, fez-se a mesma avaliação por meio da simulação numérica da escavação de 55 cm de um túnel. Em todas as simulações em que foi usado o MGC-PC a tolerância do processo iterativo foi 0,001.

3.1 Cubo Submetido a uma Carga Uniformemente Distribuída

Resolveu-se numericamente o problema de Mecânica dos Sólidos descrito na Figura 1.

Figura 1 - Cubo submetido à ação de carga uniformemente distribuída e com face indeslocável na direção da aplicação da carga



Fonte: Autor.

O cubo representado na Figura 1 foi discretizado com três malhas formadas por elementos hexaédricos de 20 nós e 3 graus de liberdade por nó, porém, com número diferente de elementos. A simplicidade do problema dispensou o teste de independência de malha. Foram utilizadas malhas com diferentes números de elementos para tornar possível a análise do desempenho do algoritmo frente ao tamanho da malha. Na Tabela 1 encontram-se as informações sobre as malhas utilizadas na discretização do problema.

Tabela 1 – Características das malhas usadas no problema

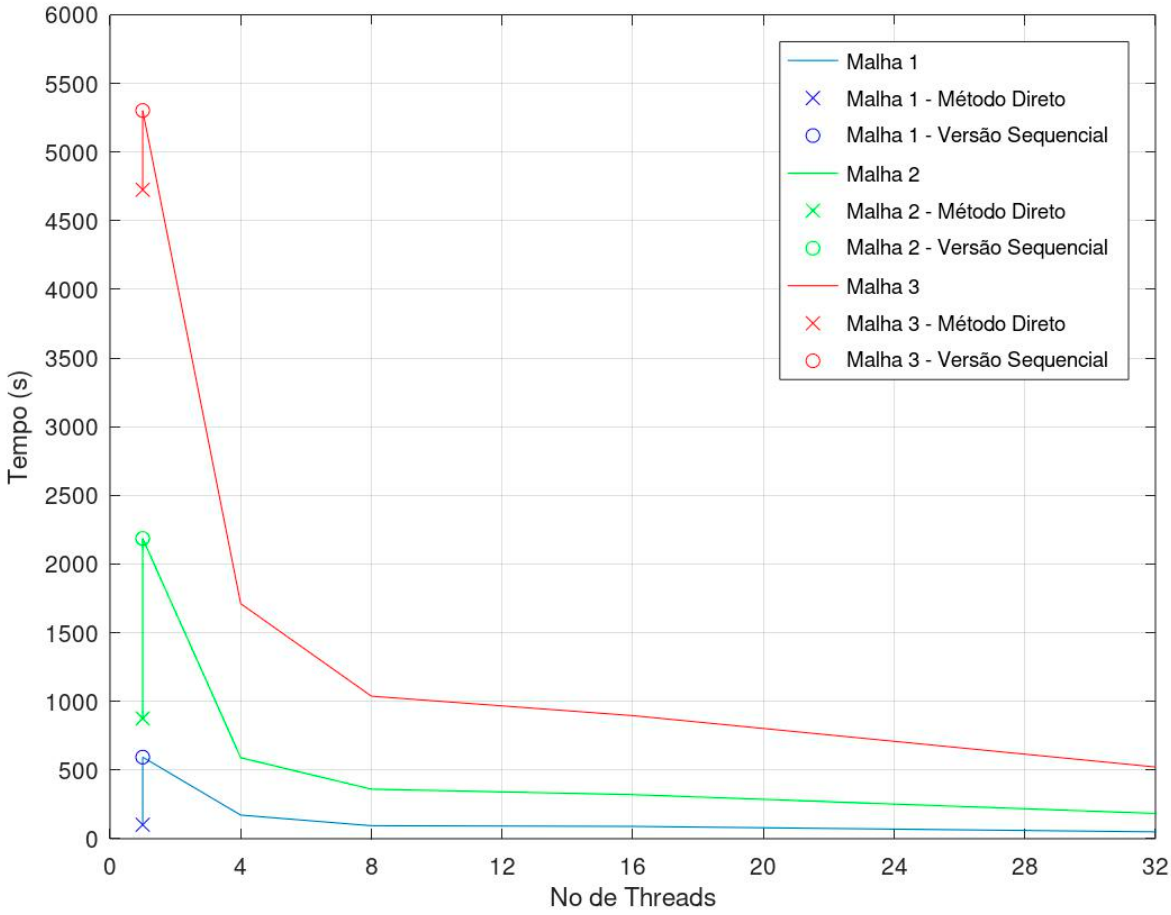
Malha	Número de Elementos	Número de Nós	Número de Equações
1	1.000	4.961	14.883
2	2.744	12.825	38.475
3	5.832	26.353	79.059

Fonte: Autor.

As simulações com as malhas descritas na Tabela 1 ocorreram da seguinte maneira: utilizando apenas uma *thread* do computador, versões do código que utilizam um método direto e a versão sequencial do código que usa o MGCP-EPE; utilizando 8, 16 e 32 *threads* a versão paralela do código que usa o MGCP-EPE. O método direto é o método de eliminação de Gauss associado à técnica de armazenagem *skyline*. Todas as simulações forneceram vetores de deslocamentos nodais que correspondem à solução analítica que é obtida da Lei de Hooke.

Com os resultados das simulações foram geradas três figuras. Na Figura 2 representou-se a curva tempo X número de núcleos. Observa-se que, nas simulações realizadas com apenas um núcleo, o método direto tem um desempenho bastante bom com as malhas menores. Com a malha menor, é 6,73% mais lento do que o paralelo com oito núcleos e, com a malha intermediária, 58,80% mais lento. Com a malha maior o método direto é 78,03% mais lento que o paralelo de oito núcleos. A versão sequencial do MGCP-EPE, com qualquer malha, mostra-se inviável. Com as três malhas verificou-se que o aumento de núcleos melhorou o desempenho da versão paralela do MGCP-EPE, o que era de se esperar. Destaca-se que, quanto maior a malha, mais importante torna-se o número de núcleos.

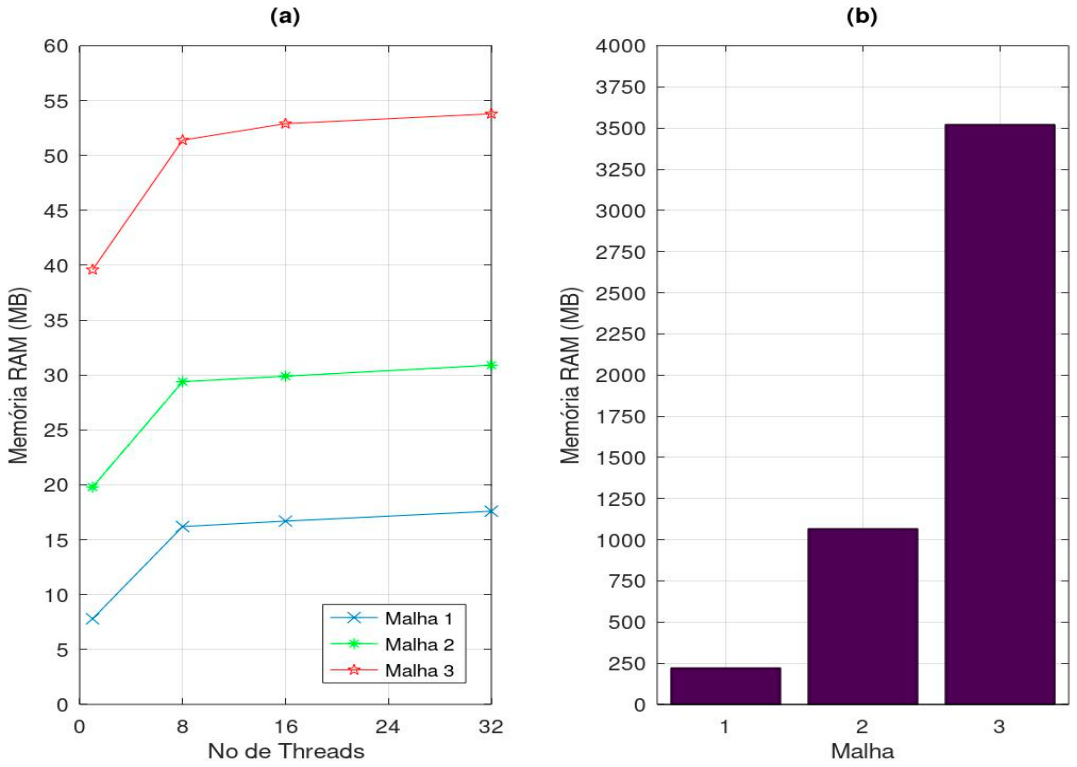
Figura 2 - Gráfico de tempo por número de núcleos para as malhas utilizadas



Fonte: Autor.

Na Figura 3 representaram-se as grandezas memória RAM utilizada X número de núcleos. Com o método direto, Figura 3 (b), que monta a matriz de rigidez global, a memória RAM utilizada com qualquer das malhas é sempre muito maior que a utilizada pelas demais versões do código. Na Figura 3 (a), nas três curvas, verifica-se um trecho inicial com maior inclinação porque, ao passar de sequencial para paralelo, cria-se a variável , além de se manter mais de uma matriz de rigidez de elemento montada. No restante das curvas, a pouca inclinação refere-se à pouca memória consumida pela matriz de rigidez de um elemento.

Figura 3 - Memória RAM x Número de núcleos: (a) versão sequência e paralela; (b) Método Direto

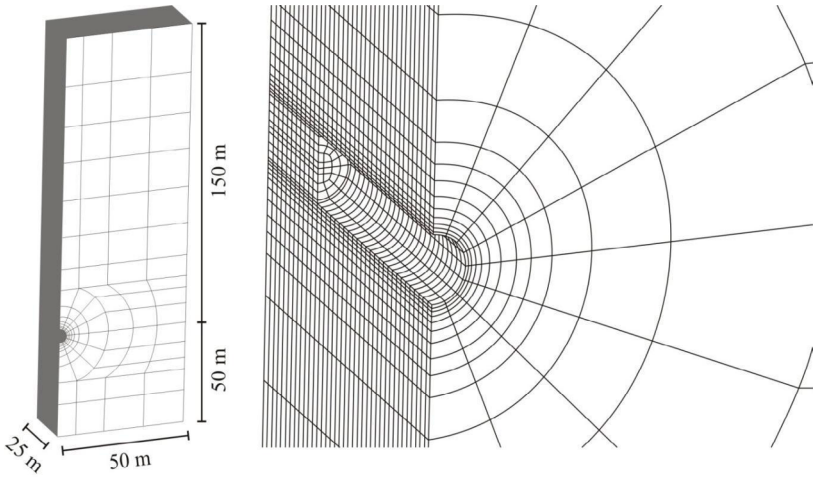


Fonte: Autor.

3.2 Escavação de um Túnel

Para avaliar a eficiência computacional do *solver* desenvolvido em simulações que visam encontrar os deslocamentos de um maciço rochoso na escavação de um túnel, realizou-se uma simulação onde o maciço rochoso está representado por uma malha de 6.808 elementos hexaédricos de 20 nós e 3 graus de liberdade por nó. A malha que representa o maciço ilustra a Figura 4.

Figura 4 - Detalhe da malha de 6.808 elementos



Fonte: adaptado de Couto, 2009.

Nessa simulação o maciço foi considerado elástico e seus parâmetros estão na Tabela 2. Os parâmetros do túnel e do processo de escavação estão na Tabela 3.

Tabela 2 – Parâmetros do maciço

Módulo de Young (E_m)	500 MPa
Coeficiente de Poisson (ν_m)	0,45
Coesão (C_m)	1,2 MPa
Ângulo de atrito (ϕ_m)	4°
Coeficiente de empuxo horizontal (k_0)	0,75
Peso específico (γ_m)	25 kN/m³

Fonte: adaptado de Couto, 2009.

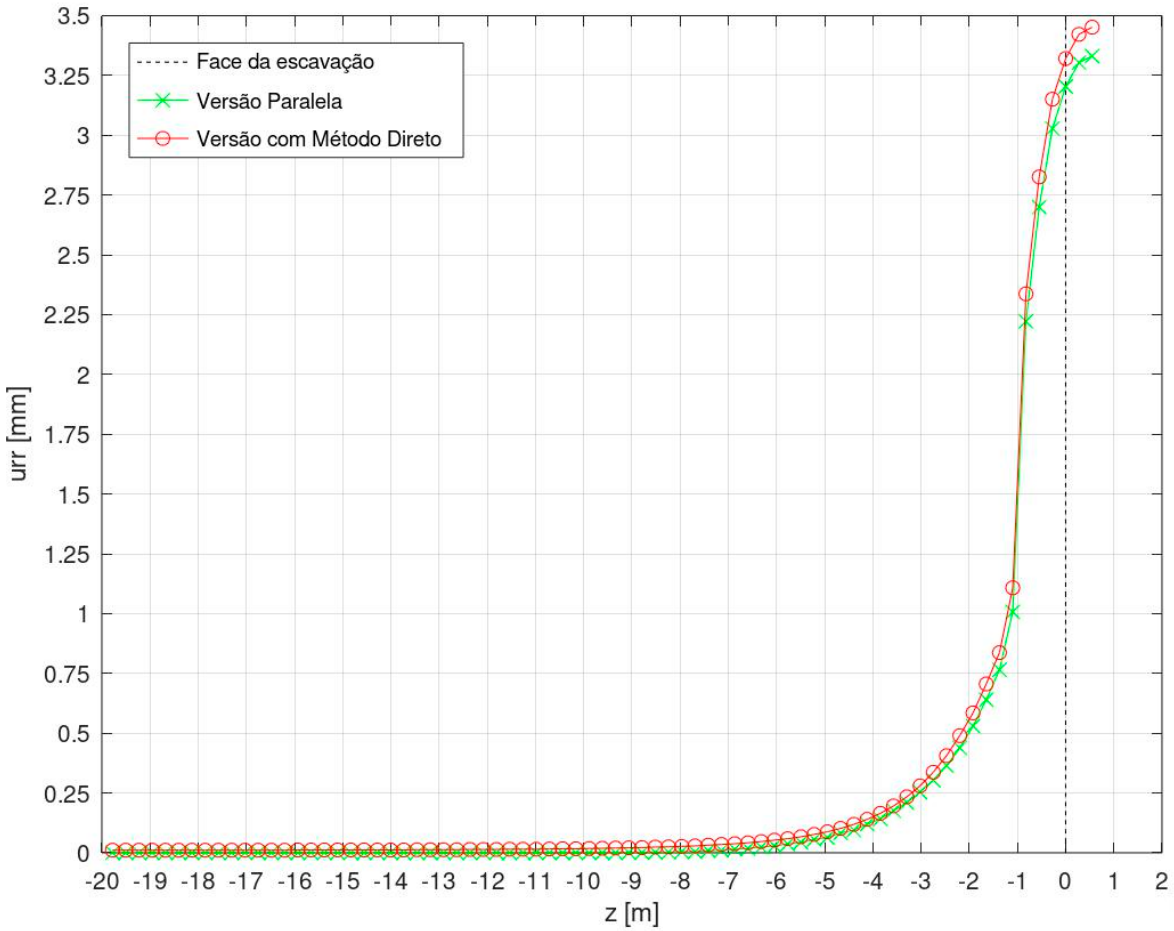
Tabela 3 – Parâmetros do túnel e do processo de escavação

Seção Transversal	**Circular**
Raio (R)	1,65 m
Passo de avanço	1/3
Distância do revestimento à face (d_0)	0

Fonte: adaptado de Couto, 2009.

A Figura 5 compara os deslocamentos radiais no teto de um túnel, correspondentes após um passo de escavação, obtidos com o *solver* original, que usa um método direto, e com o *solver* desenvolvido.

Figura 5 - Deslocamentos radiais no teto do túnel *versus* z



Fonte: Autor.

A boa aproximação das curvas representadas na Figura 5 mostra a eficiência do *solver* desenvolvido. O maior deslocamento radial corresponde à posição z igual a55 cm. A esse deslocamento corresponde a maior diferença entre a solução que utiliza o método direto e a solução paralela que utiliza 32 *threads*. Essa diferença é de 3,90%. É decorrente das tarefas, na solução paralela, serem concorrentes e acontecerem em uma ordem não determinista como ocorre na solução direta. A Tabela 4 compara o tempo consumido e a memória RAM ocupada pelas simulações representadas na Figura 5.

Tabela 4 – Tempo e memória RAM das versões direta e paralela com 32 threads

Solver	Tempo (s)	Memória RAM (MB)
Método Direto	1,695	3,5
Paralelo com 32 threads	270	1,9

Fonte: Autor.

As informações da Tabela 4 mostram que o *solver*paralelo proporciona grande redução no custo computacional, quando comparado com o que usa o método direto. O tempo consumido pela simulação é 84,07% menor e a memória RAM ocupada é 45,71% menor.

4 Conclusões

Este trabalho apresentou o algoritmo de um *solver* baseado no Método dos Gradientes Conjugados Pré-Condicionados associado à técnica de armazenamento Elemento por Elemento e à interface *OpenMP*. O seu código foi escrito em FORTRAN90. O *solver* resultante foi testado em um código pré-existente que analisa os deslocamentos de um maciço rochoso quando é realizada a escavação de um túnel. O *solver* original deste código é baseado no Método de Eliminação de Gauss associado à técnica de armazenagem *skyline*.

Antes de comparar o custo computacional dos dois *solvers* simulando a escavação de um túnel, foram realizadas simulações usando um problema simples de Mecânica dos Sólidos com solução analítica. Nessa etapa verificou-se que, quanto maior a malha, maior é a indicação de uso do *solver* desenvolvido.

Posteriormente foram realizadas as simulações de escavação de um túnel. Confirmando os resultados anteriores, o *solver* desenvolvido apresentou uma significativa redução de tempo e de memória RAM ocupada, quando comparado ao *solver* original que usa um método direto.

Agradecimentos

Os autores agradecem à Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul – FAPERGS pelo financiamento da pesquisa onde este trabalho se insere.

Referências

BATHE, KJ. *Finite Elements Procedures*. 2 Ed., Prentice-Hall, Inc, USA, 2014.

BERNAUD, D., MAGHOUS, S., BUHAN, P., COUTO, E. C. A Numerical Approach for Design of Bolt-Supported Tunnels Regarded as Homogenized Structures. *Elsevier, Tunnelling and Underground Space Technology*, *24*(5), 533-546, 2009.

CHANDRUPATLA, T. R., BELEGUNDU, A. D. *Elementos Finitos*. Tradução Daniel Vieira. 4 Ed., Pearson, São Paulo, 2014.

CHAPMAN, B., JOST, G., VAN DER PAS, R. *Using OpenMP*: Portable Shared Memory Parallel Programming. MIT Press, Massachusetts, 2008.

FU, X. D., SHENG, Q., ZHANG, Y. H., CHEN, J. Investigation of Highly Efficient Algorithms for Solving Equations in the Discontinuous Deformation Analysis Method. *Wiley, International Journal for Numerical and Analytical Methods in Geomechanics*, *40*(4), 469-486, 2016.

HESTENES, M. R., STIEFEL, E. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, *49*(4), 409-436, 1952.

LANG, J., RÜGER, G. Dynamic Distribution of Workload Between CPU and GPU for a Parallel Conjugate Gradient Method in an Adaptive FEM. *Elsevier, Procedia Computer Science*, 18, 299-308, 2013.

PENG, X., CHENG, G., YU, P., ZHANG, Y., GUO, L., WANG, C., CHENG, X., NIU, H. Parallel Computing of Three-Dimensional Discontinuous Deformation Analysis Based on OpenMP. *Elsevier, Computers and Geotechnics*, 106, 304-313, 2019.

VAZ, L. E. *Método dos Elementos Finitos em Análise de Estruturas*. Elsevier, Rio de Janeiro, 2011.