

Thiago de Sousa Goveia

**Avaliação quantitativa do método do gradiente
conjugado precondicionado para a solução paralela e
concorrente de um modelo de elementos finitos**

Proposta de trabalho apresentada como
requisito das disciplinas trabalho de conclusão
de curso I e metodologia de pesquisa.

Centro Federal de Educação Tecnológica de Minas Gerais

Campus Timóteo

Graduação em Engenharia de Computação

Orientador: Márcio Matias

Timóteo

2017

Sumário

1	INTRODUÇÃO	4
1.1	Problema de Pesquisa	5
1.2	Justificativa	5
1.3	Objetivos	5
1.3.1	Objetivos Gerais	5
1.3.2	Objetivos Específicos	6
1.4	Resultados Esperados	6
2	REVISÃO BIBLIOGRÁFICA	7
2.1	Estado da arte	7
2.2	Fundamentação teórica	9
2.2.1	Problema de Valor de Contorno	9
2.2.2	Métodos numéricos para PVCs	10
2.2.3	Método dos Elementos Finitos	10
2.2.4	Método dos Resíduos Ponderados	11
2.2.5	Método do Gradiente Conjuguado	15
3	DESENVOLVIMENTO	17
3.1	Materiais e métodos	17
3.1.1	Especificação da geometria	17
3.1.2	Obtenção dos valores de referência	18
3.1.3	Implementação do FEM	19
3.1.4	Implementação do EBE-FEM	21
3.1.5	Coloração da malha	23
3.1.6	Paralelização do algoritmo	24
3.2	Resultados	28
3.2.1	Resultados preliminares	28
4	CONCLUSÃO	32
	REFERÊNCIAS	33
	APÊNDICES	36
	APÊNDICE A – SISTEMAS DE ELEMENTOS DISCRETOS	37
A.1	Modelo de Elementos Discretos	37

A.2	Montagem do Sistema Global	40
A.3	Atribuição das condições de Contorno	41
A.4	Exemplo: Associação de resistores	41
APÊNDICE B – MÉTODO DOS ELEMENTOS FINITOS EM 2 DIMENSÕES		44

1 Introdução

Devido à popularização da computação de alto desempenho (HPC), trabalhos recentes têm retomado problemas tradicionais a fim de adequá-los aos novos paradigmas e arquiteturas de computação. De acordo com Kiss et al. (2012) a conformidade entre o problema a ser resolvido e a estrutura do ambiente de execução é capaz de ampliar a performance e reduzir a energia dispendida no processamento. Devido ao aumento da demanda por recursos computacionais, dispositivos que possibilitam a execução paralela acabaram se estabelecendo no mercado da informática. Por meio das arquiteturas *manycore* e *multicore*, é possível se executar paralela ou concorrentemente tanto tarefas corriqueiras como a exibição de vídeos e jogos até cálculos complexos da ciência e da engenharia. Podem ser citados como processadores *multicore* as linhas Core e Xeon da Intel, Opteron e Ryzen da AMD e a linha Power da IBM. Os dispositivos *manycore* por sua vez, têm como principais representantes as unidades de processamento gráfico (GPU) da qual fazem parte as placas GeForce, Quadro e Tesla da NVIDIA e Radeon e FirePro da AMD.

O método dos elementos finitos (FEM) é um método numérico para a resolução de problemas de valor de contorno (PVC) modelados por equações diferenciais e também de problemas associados à minimização de um funcional de energia (SZABO; BABUŠKA, 2009). O algoritmo clássico do FEM foi concebido em sua forma sequencial e consiste principalmente na solução de um sistema linear esparsa. A tarefa de se resolver tal sistema é geralmente custosa em termos de memória quando adotados métodos diretos como a eliminação gaussiana e computacionalmente custosa quando adotados métodos iterativos como o método de Jacobi. O método dos gradientes conjugados (CG) e suas variantes pertencem à família dos métodos exatos/iterativos do subespaço de Krylov (ANZT et al., 2016) e tem sido adotados na literatura para a solução paralela de elementos finitos. Alguns trabalhos correlatos que utilizam a família CG são apresentados por Yao et al. (2015), Ahamed e Magoulès (2016) e Iwashita et al. (2017).

A fim de adequar o FEM às arquiteturas modernas será adotado neste trabalho a abordagem elemento a elemento (EbE-FEM) proposta por Hughes, Levit e Winget (1983). Esta técnica baseia-se no fato de que a matriz do sistema de elementos finitos é caracterizada como uma função parcialmente separável (DAYDE; L'EXCELLENT; GOULD, 1995), resultado da soma das matrizes elementares. Assim sendo, as operações da solução do sistema de elementos finitos podem ser realizadas em nível elementar, sem a necessidade de se montar a matriz global de coeficientes. Adicionalmente tem-se a vantagem de que elementos não adjacentes podem ser calculados simultaneamente a partir dos métodos da família CG (WATHEN, 1989). Esta última característica torna a técnica EbE-FEM propícia para a implementação paralela nas arquiteturas modernas.

1.1 Problema de Pesquisa

Neste trabalho é feita uma análise quantitativa do desempenho do algoritmo do EbE-FEM nas linguagens concorrentes C++, Erlang e Scala e nas linguagens de computação paralela CUDA e Harlan. O problema *benchmark* a ser resolvido refere-se à equação de Laplace originada do cálculo da distribuição de potencial e do campo elétrico de um capacitor de placas paralelas (BOYLESTAD, 2011, Exemplo 10.3). A escolha deste problema se deve à simplicidade de sua modelagem e ao seu comportamento já explorado em livros de eletromagnetismo e circuitos elétricos.

1.2 Justificativa

A justificativa deste trabalho se baseia na contínua transformação dos paradigmas de programação e arquiteturas de hardware. Como coloca Guo et al. (2014), com aumento de núcleos de processamento, ocorre a redução da razão memória por núcleo, o que impõe uma forte demanda para que os algoritmos utilizem eficientemente todos os níveis de paralelismo disponíveis enquanto minimizam a movimentação de dados. Tal evolução não se limita ao cenários dos *clusters* e *grids* mas alcança inclusive os dispositivos móveis, que atualmente já possuem oito núcleos. Pensando em um futuro próximo, com os avanços da computação ubíqua que introduz temas como internet das coisas, dispositivos "usáveis", realidade aumentada e realidade virtual, a necessidade de se aproveitar ao máximo todo o poder de processamento disponível se torna ainda mais evidente, uma vez que nessas tecnologias há alta demanda de processamento e/ou pouco espaço físico para comportar um processador adequado. A justificativa para a escolha das linguagens C++, Scala e Erlang se deve à ausência de trabalhos acadêmicos relacionando tais tecnologias e seu desempenho. C/C++ é uma linguagem abrangente, robusta e atual, presente por trás de grande parte das aplicações *desktop* e *mobile*. Scala e Erlang (e/ou Elixir) são linguagens naturalmente concorrentes e que possuem crescente *market share*.

1.3 Objetivos

1.3.1 Objetivos Gerais

- Demonstrar o processo de mudança do paradigma sequencial para uma solução paralela do método do elementos finitos;
- Avaliar quantitativamente, segundo as métricas propostas (tempo de execução, uso de memória, e *speedup*) o desempenho do processo de solução do EbE-FEM.

- Apresentar à comunidade resultados acadêmicos experimentais do desempenho das linguagens C++, Scala, Erlang, CUDA e Harlan.

1.3.2 Objetivos Específicos

- Investigar a viabilidade da técnica EbE-FEM como alternativa dos *solvers* iterativos e do solver do MATLAB[®] ;
- Desenvolver um material acessível e de fácil compreensão de introdução ao FEM para o nível da graduação;
- Investigar o quanto otimizada é a convergência do EbE a partir do emprego de diferentes preconditionadores;
- Relacionar as métricas e conceitos de estatísticas necessários para a avaliação adequada de performance;
- Comparar conforme as métricas o desempenho das linguagens nativamente concorrentes e paralelas na resolução do problema *benchmark* deste trabalho.

1.4 Resultados Esperados

Espera-se com este trabalho iniciar no CEFET-MG campus Timóteo uma nova linha de pesquisa a ser continuada nos trabalhos futuros, voltada para a análise numérica de problemas de valor de contorno da física aplicada. De forma similar, espera-se o incentivo e a adoção por parte da universidade de paradigmas e linguagens emergentes, a fim de diversificar o currículo dos graduandos.

2 Revisão bibliográfica

2.1 Estado da arte

Como coloca Kiss et al. (2012), o processamento paralelo de um problema modelado pelo FEM pode ser feito a partir da decomposição do domínio do problema. Esta decomposição pode ser feita por meio do particionamento da malha ou por meio da decomposição apropriada da matriz de coeficientes. Os trabalhos de Boehmer et al. (2011) e Ahamed e Magoulès (2016) apresentam casos com a solução paralela do FEM por meio do particionamento da malha. No primeiro caso o processamento é executado comparativamente nas arquiteturas de memória compartilhada e distribuída com o uso da API OpenMP e da biblioteca MPI respectivamente. No segundo caso, a comparação é feita entre as linguagens CUDA e OpenCL que são executadas em uma arquitetura *manycore*.

A técnica EbE é uma forma de decomposição da matriz de coeficientes e será tratada neste trabalho. Por meio desta, as operações são realizadas na matriz de cada elemento, sem que seja necessária a montagem do sistema global. Esta estrutura de dados foi proposta originalmente por Hughes, Levit e Winget (1983) como uma fatoração especial para a matriz de coeficientes, de forma a melhorar sua condição a acelerar a convergência (CAREY et al., 1988). Devido ao seu desempenho, precisão, economia de memória e à possibilidade de processamento paralelo (LEVIT, 1987; HU; QUIGLEY; CHAN, 2008; KISS et al., 2012) a adoção da abordagem EbE tem sido recorrente à medida em que surgem novas tecnologias de HPC.

Carey et al. (1988) apresenta uma implementação EbE do gradiente biconjugado (BiCG) para solucionar um sistema de elementos finitos (FE). Segundo ele, o surgimento de novas arquiteturas tais como os processadores vetoriais, paralelos e estações de trabalho microprocessadas foram responsáveis por se repensar o algoritmo original do FEM. A execução vetorial foi realizada no computador CRAY-XMP e foi 8 vezes mais rápida em relação ao processamento sequencial e apresentou *speed-up* variando de 4.25 à 6.5. A execução paralela foi feita no ALLIANT-FX/8, um mini supercomputador com 8 CPUs. Foi adotado um esquema de ordenação de nós a fim de se evitar condições de corrida no acesso às variáveis globais. Com a utilização de todas as unidades de processamento, o *speed-up* foi em torno de 7 em relação ao uso de uma única CPU.

O trabalho de Dayde, L'Excellent e Gould (1995) faz uma análise comparativa entre 5 precondicionadores de nível elementar para o algoritmo do gradiente conjugado (CG), a saber: *Element matrix factorization* (EMF), *finite element preconditioner* (FEP), *one-pass element-by-element preconditioner* (EbE), *two-pass element-by-element preconditioner*

(EbE2) e Gauss-Seidel *element-by-element preconditioner* (GS-EBE). Como os autores colocam, uma vez que se têm problemas de larga escala mas parcialmente separáveis, torna-se necessário explorar diferentes alternativas a fim de se aproveitar as vantagens oferecidas pela estrutura do problema. A abordagem EbE se mostrou a melhor opção entre seus concorrentes. EMF e FEP requerem a montagem parcial do sistema, o que agrupa maior custo de processamento. EBE2 e GS-EBE não apresentaram boas aproximações para elementos com pouca vizinhança. A fim de se realizar a paralelização, é sugerida a coloração da malha de forma que elementos vizinhos sejam processados sequencialmente.

Uma implementação EbE-CG em FPGA (Field Programmable Gate Array) é proposta por [Hu, Quigley e Chan \(2008\)](#). Neste trabalho as operações sobre as matrizes elementares foram realizadas por meio da configuração de um circuito lógico no chip 4VLX160 da Xilinx. O processamento sequencial foi feito em um PC 2.01 GHz Athlon 64 com as devidas otimizações na compilação. Graças à implementação diretamente em hardware foi alcançado um *speed-up* máximo igual a 40.

O surgimento da linguagem CUDA em 2006 e a popularização das GPGPU possibilitaram que a técnica de EbE pudesse ser revisitada e aplicada nas arquiteturas modernas. [Kiss et al. \(2012\)](#) soluciona o um modelo de elementos finitos por meio do BiCG com precondicionador de Jacobi. De acordo com este trabalho, a abordagem EbE é adequada para o processamento em GPU, cuja arquitetura embora seja massivamente paralela, possui um gargalo, que é a capacidade limitada de memória. O caráter localizado do EbE faz com que o trânsito de dados seja mínimo, o que reduz o consumo de energia e maximiza o potencial do dispositivo. Para a realização dos testes foram utilizados o processador quad-core Xeon X3440 da Intel e a placa GTX 590 da NVIDIA, contendo 2 GPUs. A execução com aceleração em GPU consumiu 20 vezes menos memória e foi 10 vezes mais rápida que a execução unicamente em CPU.

O trabalho de [Wu et al. \(2015\)](#) também apresenta o método BiCG elemento a elemento com precondicionador de Jacobi implementado em GPU. Foi utilizado o processador Xeon E5-2696v2 da Intel e a placa Tesla K20c da NVIDIA. Nos dois refinamentos de malha testados foi alcançado um *speed-up* de 4.63, e como coloca o autor, o resultado obtido se torna ainda mais efetivo à medida em que o número de elementos aumenta. No trabalho mais recente dos mesmos autores ([YAN et al., 2017](#)) é apresentado o uso do precondicionador de Gauss-Seidel para a abordagem EbE, o qual apresentou melhores resultados para o número de iterações e tempo de execução em relação ao precondicionamento de Jacobi, mantendo a mesma precisão.

Outros trabalhos importantes para esta monografia são os de [Xu, Yin e Mao \(2005\)](#), [Yao et al. \(2015\)](#) e [Chou e Chen \(2016\)](#). O primeiro apresenta a implementação do EbE-CG e a técnica de atribuição das condições de contorno que também será empregada no presente trabalho. O segundo apresenta a comparação da performance da implementação

Condição	Tipo
$u(x_i) = \alpha$	Dirichlet
$u(x_i) = 0$	Dirichlet Homogênea
$u'(x_i) = \beta$	Neumann
$u'(x_i) = 0$	Neumann Homogênea

Tabela 1 – Exemplos de condições de contorno

do BiCG estabilizado (BiCGStab) em CPU e GPU, utilizando-se diferentes formas de armazenamento de matriz esparsa e diferentes ferramentas de resolução de sistema linear em GPU, a saber: CUSPARSE, CUSP e CULA Sparse. O último trabalho apresenta a comparação entre diferentes tecnologias de processamento paralelo (Cuda C, Cuda Fortran, MPI e OpenMP) na resolução de dois problemas *benchmark*

2.2 Fundamentação teórica

2.2.1 Problema de Valor de Contorno

Um problema de valor inicial PVI, pode ser definido como uma equação diferencial (ou sistema de equações) na qual são dadas as condições iniciais do fenômeno. Tais condições são impostas sobre a variável dependente e suas derivadas em um único instante de tempo (inicial) t_0 . Um problema de valor de contorno PVC por sua vez, apresenta tais condições impostas em pontos distintos, como por exemplo em x_i e x_f . Geralmente os PVI são dados em função do tempo enquanto os PVC são dados em função do espaço (BOYCE; DIPRIMA, 2010).

Como pode ser visto na tabela 1. as condições estabelecidas sobre a variável dependente, são chamadas de *condições de Dirichlet ou essenciais*, enquanto as que são estabelecidas sobre as derivadas da variável dependente são conhecidas como *condições de Neumann ou naturais*. Além destas duas, existem as restrições específicas do fenômeno modelado, como por exemplo, condições de radiação ou de impedância para problemas do eletromagnetismo (JIN, 2002).

A solução analítica de um PVC pode ser obtida por meio da integração direta ou a partir da aplicação de técnicas como a separação de variáveis, expansão em séries ou pela transformada de Laplace. No entanto, a maioria dos problemas da engenharia e da ciência não são lineares e apresentam geometria ou condições de contorno complexas. Estas características fazem com que a resolução analítica de tais problemas seja impraticável, sendo necessário recorrer a métodos numéricos para se obter uma solução aproximada (BOYCE; DIPRIMA, 2010; POWERS, 2006).

2.2.2 Métodos numéricos para PVCs

O método dos elementos finitos (FEM) é uma alternativa numérica para a solução de PVCs. Neste método, o domínio do problema é visto como uma coleção de subdomínios, chamados de elementos finitos, sobre os quais, a equação que modela o problema é aproximada por um método variacional ou de resíduos ponderados [Reddy \(2006\)](#). Essas diferentes vertentes do método surgiram graças aos esforços independentes de matemáticos, cientistas e engenheiros ([ZIENKIEWICZ, 2005](#)).

O método das diferenças finitas (FDM) assim como o FEM é uma abordagem numérica para a aproximação de PVCs. Este método consiste na discretização do domínio do problema por meio de uma grade de pontos e na aproximação de cada derivada da equação por um quociente-diferença adequado ([FAIRES, 2008](#)). Embora este método seja útil em muitos casos, se torna difícil aplicá-lo em problemas com geometria irregular ou com condições de contorno não usuais. Um exemplo pode ser visto na figura 1.

Diferentemente do FDM, como coloca [Huebner et al. \(2001\)](#), o FEM divide o domínio não em pontos, mas em subdomínios, sobre os quais as equações são aproximadas por polinômios definidos por partes. O FEM também é capaz de representar mais fielmente o contorno (ou a borda) do problema. Desta forma, ele se apresenta como uma técnica mais poderosa e versátil para a modelagem de fenômenos com geometria complexa e meios não homogêneos ([SADIQU, 2001](#)).

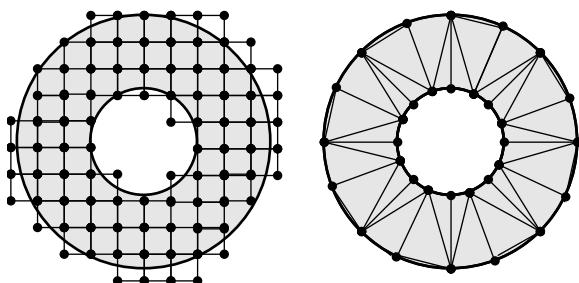


Figura 1 – Método das diferenças Finitas e método dos Elementos Finitos

2.2.3 Método dos Elementos Finitos

O FEM surgiu originalmente como uma técnica de análise de deslocamentos e elasticidade de estruturas mecânicas, mas em seguida foi estendido para solucionar problemas de outros campos da física e da engenharia ([JIN, 2002; DESAI, 1972; ZIENKIEWICZ, 2005](#)).

As primeiras formulações do FEM são conhecidas como *abordagem direta* ou *formulação física*, que embora forneça a interpretação intuitiva do método, é útil apenas para a resolução de problemas relativamente simples ([HUEBNER et al., 2001; DESAI, 1972; ZIENKIEWICZ, 2005](#)). O uso do princípio do trabalho virtual, para a determinação

de forças na abordagem direta, levou à generalização do FEM, por meio da estratégia de minimização do funcional de energia. Esta técnica mais genérica ficou conhecida como *formulação variacional* (DESAI, 1972; ZIENKIEWICZ, 2005; JIN, 2002) e tem como principal representante o método de Rayleigh-Ritz. Uma terceira abordagem, conhecida como *Método dos Resíduos Ponderados* ou *formulação generalizada* (ZIENKIEWICZ, 2005; HUEBNER et al., 2001) é tradicionalmente utilizada e é ainda mais genérica que o princípio variacional, pois resolve diretamente as equações diferenciais do modelo, sem necessitar da existência de um funcional de energia (DESAI, 1972), ou ainda, sem exigir que o sistema seja positivo-definido. Neste trabalho será adotado o método dos resíduos ponderados, mais especificamente, o método de Galerkin.

2.2.4 Método dos Resíduos Ponderados

De acordo com Jin (2002), a aplicação do FEM pode ser feita a partir de 4 passos básicos: discretização do domínio, seleção das funções de interpolação, formulação do sistema de equações e solução do sistema de equações. Os três primeiros passos são descritos nos tópicos a seguir e o último na subseção [seguinte](#).

Discretização do Domínio

A discretização do domínio consiste na transformação do contínuo Ω em uma malha de elementos finitos (discretos). Cada elemento Ω_e dessa malha representa um subdomínio de Ω . Nesta etapa são definidas a forma, a quantidade e o tamanho dos elementos, de maneira que a representação em malha seja a mais próxima possível do objeto em análise (DESAI, 1972).

Conforme pode ser visto na figura 2, cada elemento é identificado na malha a partir de um número que lhe é atribuído. Os vértices (ou nós) de cada elemento também são numerados. Cada nó possui dois valores vinculados a ele, um atuando como identificador global (numeração do nó na malha) e o outro como identificador local (numeração dentro de um dado elemento). A numeração local é geralmente feita no sentido anti-horário, a fim de se obter um valor positivo no cálculo da área ou volume por meio do determinante (SADIKU, 2001; JIN, 2002).

Um fator que deve ser levado em consideração é o balanceamento entre o refinamento da malha e o esforço computacional necessário (DESAI, 1972). Como o valor da solução é aproximado para cada elemento, o excesso de elementos pode causar a propagação do erro de aproximação, levando a resultados indesejáveis.

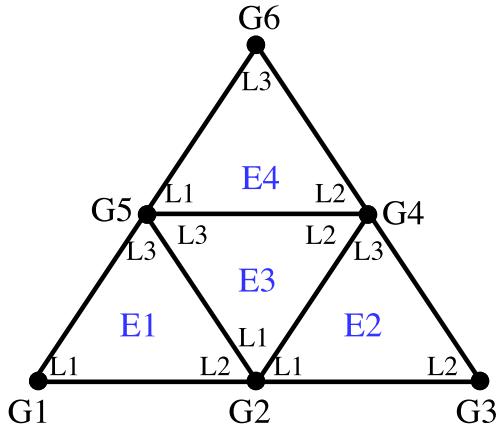


Figura 2 – Identificadores de elementos e nós

Seleção das funções de interpolação

O passo seguinte é a escolha das funções de interpolação (ou função de base ou de forma) (HUEBNER et al., 2001) que fornece uma aproximação da equação original dentro de cada subdomínio (JIN, 2002). A função interpoladora escolhida geralmente é um polinômio, e isso ocorre por dois motivos a priori (DESAI, 1972):

- Facilidade de manipulação matemática, principalmente derivação e integração;
- Aproximação satisfatória quando truncado em uma ordem qualquer.

Na prática são escolhidos polinômios de primeira ou segunda ordem, mas ordens superiores podem ser adotadas para reduzir o erro de aproximação, sobretudo em bordas com curvaturas, no entanto, ocorre também o aumento da carga computacional (JIN, 2002).

Seja a função ϕ desconhecida, da qual se deseja obter uma aproximação. A nível elementar, a função $\tilde{\phi}^e$ aproxima ϕ dentro do domínio do elemento e . Considerando um subdomínio bidimensional triangular a aproximação linear para o elemento e pode ser feita como na equação 2.1. De acordo com esta aproximação, o valor em um dado nó i pode ser obtido pela equação 2.2.

$$\tilde{\phi}^e = a^e + b^e x^e + c^e y^e \quad (2.1)$$

$$\tilde{\phi}_i^e = a^e + b^e x_i^e + c^e y_i^e \quad i = 1, 2, 3 \quad (2.2)$$

Após a resolução do sistema 2.2, a fim de se obter os valores dos coeficientes a , b e c , uma função de interpolação N é gerada. Conforme mostra a equação 2.3, os valores ϕ^e

em um ponto (x, y) de e podem ser aproximados a partir dos valores nodais ϕ_j^e .

$$\tilde{\phi}^e = \sum_{j=1}^n N_j^e(x, y) \phi_j^e = \{N^e\}^T \{\phi^e\} \quad (2.3)$$

As funções N_i^e obtidas interpolam os valores no interior do domínio e preservam os valores nodais $\tilde{\phi}_1^e$, $\tilde{\phi}_2^e$ e $\tilde{\phi}_3^e$. Para que isso ocorra, essas funções devem se comportar como o delta de Kronecker, como mostra a equação 2.4 a seguir.

$$N_i^e(x_j^e, y_j^e) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (2.4)$$

As funções de interpolação mais comuns pertencem às famílias de funções Lagrange e Serendipity (ZIENKIEWICZ, 2005; VOLAKIS; CHATTERJEE; KEMPEL, 1998).

Formulação do sistema de equações

Na abordagem variacional destaca-se o método de Ritz, ou Rayleigh-Ritz, o qual tem por objetivo, minimizar o funcional variacional, ou funcional de energia, do problema aproximado (VOLAKIS; CHATTERJEE; KEMPEL, 1998). Embora tal abordagem tenha sido historicamente utilizada e possua fundamentação física e matemática, sua adoção, em muitos casos, é mais complicada em relação ao Método dos Resíduos Ponderados, pois demanda a formulação variacional do problema. Desta forma, se um problema é dado por um modelo diferencial, é necessário se obter a partir deste modelo a forma variacional equivalente, para só então se aplicar o método para a obtenção do sistema de equações. No eletromagnetismo, a formulação variacional das equações de Maxwell não é bem estabelecida (JIN, 2002).

Para problemas que apresentam explicitamente condições de Dirichlet no contorno e cujo operador \mathcal{L} é linear e auto-adjunto, é possível se obter imediatamente o funcional de energia correspondente (ZIENKIEWICZ, 2005), no entanto, como mostra (VOLAKIS; CHATTERJEE; KEMPEL, 1998, p. 29), para este tipo de operador a mesma integral da formulação variacional é obtida pelo método de Galerkin.

Seja a equação 2.5, uma equação de um PVC na qual \mathcal{L} é um operador diferencial, f é uma função de excitação conhecida e ϕ é a solução procurada.

$$\mathcal{L}\phi = f \quad (2.5)$$

Se substituirmos a solução exata pela sua aproximação apresentada na equação 2.3, um resíduo r surge, como pode ser visto na equação 2.6, em decorrência dos erros de

aproximação.

$$r = \mathcal{L}\tilde{\phi} - f \neq 0 \quad (2.6)$$

Embora o resíduo em cada elemento seja diferente de zero em sua maioria, deseja-se que o resíduo total seja igual a zero. Para que isso seja possível, uma função de ponderação w_i é introduzida para cada subdomínio e . Desta forma na média dos M subdomínios de Ω a condição residual é atendida (VOLAKIS; CHATTERJEE; KEMPEL, 1998). As equações 2.7 e 2.8 correspondem ao sistema de resíduos ponderados. A ideia chave dessa abordagem é transformar o problema de valor de contorno em um sistema linear que forneça o valor aproximado.

$$R_i^e = \int_{\Omega} w_i^e r \, d\Omega = 0 \quad i = 1, 2, 3. \quad (2.7)$$

$$R_i^e = \int_{\Omega} w_i^e [\mathcal{L}\tilde{\phi} - f] \, d\Omega = 0 \quad i = 1, 2, 3. \quad (2.8)$$

A escolha particular de $w_i^e = N_i^e$ configura o método de Galerkin, o qual, para um operador \mathcal{L} auto adjunto a matriz do sistema é simétrica e equivalente à matriz gerada pelo método de Ritz. Após a substituição da equação 2.3 em 2.8 obtém-se a equação 2.9, da qual são extraídas as matrizes elementares, como mostra a equação 2.10.

$$R_i^e = \int_{\Omega} N_i^e \mathcal{L} \left(\sum_{j=1}^n N_j^e \phi_j^e \right) - \int_{\Omega} N_i^e f \, d\Omega = 0 \quad i = 1, 2, 3. \quad (2.9)$$

$$\sum_{j=1}^n \int_{\Omega} N_i^e \mathcal{L} N_j^e \phi_j^e = 0 \Rightarrow [K^e] \{\phi^e\} = \{b^e\} \quad i = 1, 2, 3. \quad (2.10)$$

É importante colocar que o parâmetro w_i deve ser um conjunto de funções integráveis e que respeitem as condições de contorno, ou seja, w_i deve ser igual a zero nos nós de contorno do problema. (REDDY, 2006). Alguns casos especiais do método dos resíduos ponderados são obtidos a partir da escolha de w_i :

Método de Petrov-Galerkin		$w_i = \psi_i \neq \phi_i$
Método de Galerkin		$w_i = \phi_i$
Método dos Mínimos quadrados	$w_i = \frac{d}{dx} \left(a(x) \frac{d\phi_i}{dx} \right)$	
Método da colocação		$\delta(x - x_i)$

(2.11)

Uma vez que as matrizes elementares foram obtidas é realizado o mapeamento dos identificadores locais para os identificadores globais. Este processo é conhecido como

montagem da matriz global e é detalhado no apêndice A. O sistema global é esparsa e da ordem do número de nós da malha. As contribuições elementares (ou os coeficientes das matrizes de elementos) são agregados na matriz e as condições de contorno são aplicadas.

Neste trabalho são abordadas apenas as condições de contorno de Dirichlet. Essas condições, também conhecidas como essenciais, correspondem aos valores pre-estabelecidos da variável dependente. Dessa forma a definição do contorno consiste na substituição das incógnitas dos nós de contorno por seus respectivos valores. Assim, se um sistema de ordem 1000 possui 200 valores pré-fixados ele pode ser reduzido à ordem 800. Um exemplo bidimensional com os passos descritos nessa subseção é apresentado no apêndice B.

2.2.5 Método do Gradiente Conjugado

O método do gradiente conjugado (CG) foi proposto por Hestenes e Stiefel em 1952 como um método direto para a solução de sistemas n por n positivos definidos (FAIRES, 2008). Embora sua concepção seja de um método exato que converge após n iterações, sua aplicação tem sido feita como um método iterativo para se aproximar a solução de sistemas esparsos (BARRETT et al., 1995). O método trabalha na busca da direção de maior decrescimento do funcional de energia, ou seja, na direção contrária do vetor gradiente para problemas de minimização. Inicialmente é atribuído um “chute” para a solução, e à este chute é vinculado um certo resíduo, proporcional à distância entre o valor do chute e o valor da solução (mínimo do funcional). A cada iteração é calculada a correção da direção, a qual é tomada de tal forma que seja ortogonal às direções anteriores, impedindo que o método repita uma direção já percorrida. As direções ortogonais são obtidas por meio de sucessivas multiplicações da matriz do sistema pelo vetor direção. Dessa forma a aproximação da solução em cada iteração é encontrada em um subespaço de Krylov, como mostra a equação 2.12. O pseudocódigo 1 descreve as iterações do CG.

$$K^m = \text{span}(d, Ad, A^2d, A^3d, \dots, A^{m-1}d) \quad (2.12)$$

Algorithm 1 Algoritmo do Gradiente Conjugado

```
1:  $d_0 = r_0 = b - Ax_0$ 
2: for  $j = 1, 2, \dots$  do
3:    $\alpha = \frac{r^i T r^i}{d^i T A d^i}$ 
4:    $x^{i+1} = x^i + \alpha d^i$ 
5:    $r^{i+1} = r^i - \alpha d^i$ 
6:    $\beta = \frac{r^{i+1} T r^{i+1}}{r^i T r^i}$ 
7:    $d^{i+1} = d^i - \beta d^i$ 
8:   if  $\frac{\|r^{i+1}\|}{\|b\|}$  then
9:     break
10:    end if
11: end for
```

3 Desenvolvimento

O desenvolvimento deste trabalho está dividido nas etapas de *validação do problema e aplicação das estratégias*. Na primeira são apresentados os recursos utilizados para modelar o problema e para definir as possíveis estratégias de solução. As decisões tomadas na segunda etapa terão como base os resultados preliminares obtidos na primeira.

3.1 Materiais e métodos

O ambiente de desenvolvimento MATLAB® foi adotado para a realização da etapa de validação do problema. A motivação dessa escolha se deu pela facilidade da manipulação de matrizes oferecida pelo ambiente e pela existência de ferramentas dedicadas à especificação de PVCs e à programação paralela.

3.1.1 Especificação da geometria

O problema *benchmark* utilizado neste trabalho é o exemplo 10.3 do livro Análise de Circuitos de Boylestad (2011) e é ilustrado na figura 3(a). O capacitor é composto de duas placas quadradas paralelas com lado de 2 polegadas sendo a distância entre elas igual a $\frac{1}{32}$ polegadas. O problema informa que a diferença de potencial entre as placas é de 48V, dessa forma foi assumido que uma das placas possui 48V e que a outra está aterrada.

A fim de se realizar uma análise mais ampla do problema a distribuição do campo elétrico é calculada entre as placas mas também no espaço ao redor. Será considerada uma região quadrada de 16cm de lado como mostra o esquema na figura 3(b). Também foi considerado que a espessura de cada placa é a metade da largura da região entre elas, ou seja, $\frac{1}{64}$ polegadas.

A fim de se obter um parâmetro da corretude dos resultados obtidos, foi feita a simulação do problema na *partial differential equation toolbox* do MATLAB®. Por meio dessa ferramenta é possível programaticamente e via interface gráfica definir a equação diferencial que rege o problema, a geometria, as condições de contorno, a malha inicial e seus refinamentos bem como a solução e exibição dos resultados do PVC. As figuras 3(c) e 3(d) mostram a geometria e a malha inicial modeladas com o auxílio da interface gráfica da *pdetool*.

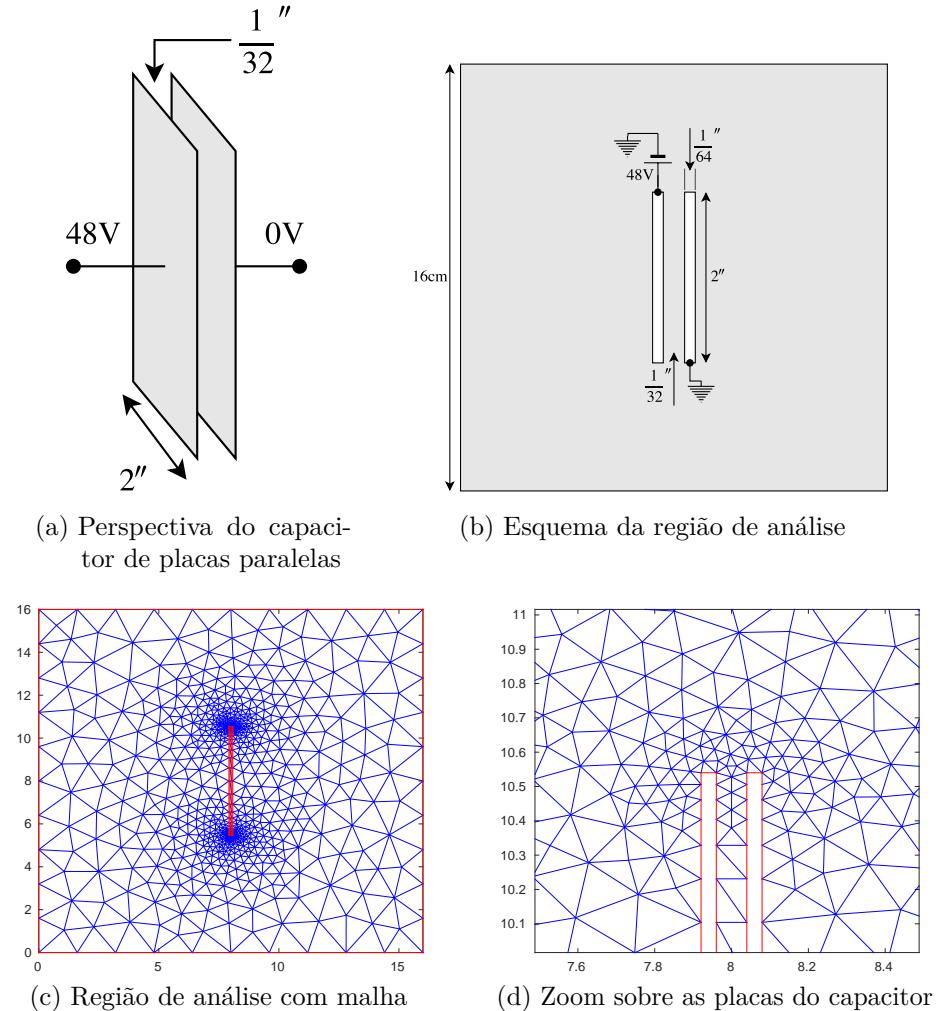


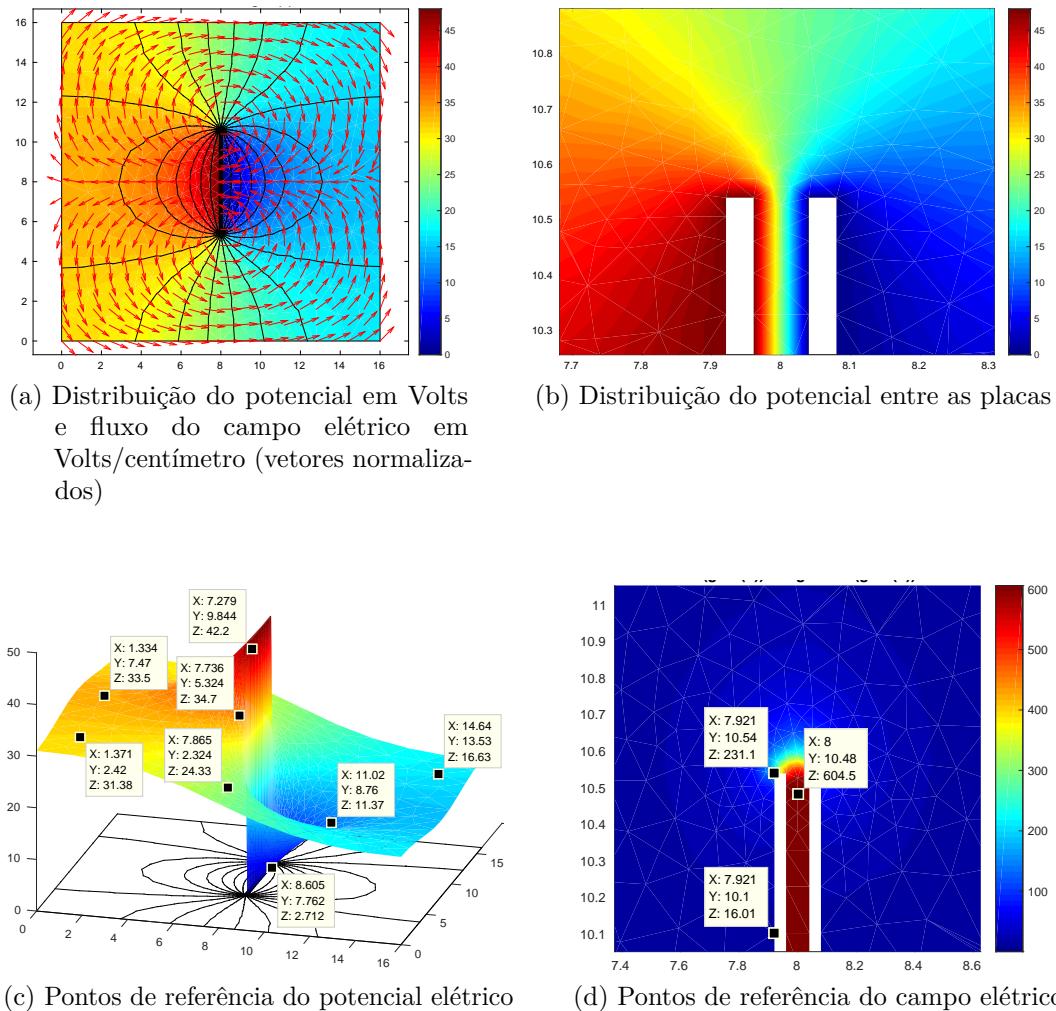
Figura 3 – Geometria do capacitor de placas paralelas

3.1.2 Obtenção dos valores de referência

Definidas a geometria e a malha, foram feitas as atribuições das condições de contorno de Dirichlet nas 4 bordas da seção de cada placa. À placa da esquerda foram atribuídos 48V e 0V para a placa da direita. Para se definir a EDP do problema é necessário atribuir valores aos coeficientes da fórmula geral de EDPs elípticas, mostrada na equação 3.1.

$$-\nabla \cdot (c \nabla (u)) + au = f \quad (3.1)$$

Na equação de Laplace, que modela a distribuição do potencial elétrico, apenas o coeficiente c é diferente de zero. A constante c corresponde à permissividade no vácuo na equação de Poisson, mas como a densidade volumétrica de carga representada por f e o auto valor a são iguais a zero na equação de Laplace, c passa a valer 1. Estabelecidos os valores das constantes a solução do problema pode ser obtida, como mostram as figuras 4(a) e 4(b).

Figura 4 – Solução do PVC com a *pdetool*

De posse da solução da *pdetool*, foram tomados 8 pontos de referência da distribuição do potencial e 3 pontos de referência do campo elétrico. A localização e os valores destes pontos são apresentados na figuras 4(c) e 4(d).

3.1.3 Implementação do FEM

Após a simulação e obtenção dos valores de referência foi realizada a implementação do método dos elementos finitos em duas dimensões, conforme o apresentado na seção B do referencial teórico. A fim de não tirar proveito das funções do MATLAB[®] de forma a manter a generalidade da solução implementada e a facilidade de transcrevê-la para outras linguagens, foram utilizadas nos algoritmos deste trabalho apenas as funções da *PDE toolbox* relativas à geometria, geração da malha e exibição dos resultados. No pseudocódigo 2 as palavras destacadas em itálico são variáveis, as funções destacadas em negrito fazem parte do MATLAB[®] e as funções descritas nas linhas de 9 a 12 são detalhadas no exemplo do apêndice B. A tabela 2 contém a descrição das funções utilizadas.

Algorithm 2 Pseudocódigo do FEM

```

1: load(dadosGeometria);
2: load(coordReferencia);
3: load(dadosContorno);
4:  $g = \text{decsg}(\text{dadosGeometria})$ ;
5:  $m = \text{createpde}(1)$ ;
6: geometryFromEdges( $m, g$ );
7: generateMesh( $m$ , 'Hmax', valHmax);
8: ( $nos, tri$ ) = meshToPet( $m.Mesh$ );
9:  $C = \text{geraMatElementares}(nos, tri)$ ;
10:  $G = \text{geraMatGlobal}(C, tri)$ ;
11: ( $A, b$ ) = atribuiContorno( $G, dadosContorno$ );
12:  $sol = \text{resolveSistLinear}(A, b)$ ;
13: pdeplot( $m, \dots$ );

```

A função *resolveSistLinear* é o ponto central deste trabalho, uma vez que a análise é feita sobre o desempenho das implementações durante solução do sistema de equações originado do FEM. Foram utilizados 2 *solvers* a fim de se verificar a performance e a gestão de memória. O primeiro deles é o *solver* “\” do MATLAB® e o segundo é a implementação do método do gradiente conjugado proposta por [Barrett et al. \(1995\)](#). O CG como explicitado no referencial teórico, é um método iterativo não estacionário do subespaço de Krylov. Como coloca [Wathen \(1989\)](#), o uso de métodos iterativos se tornou competitivo à medida em que os sistemas de equações se tornavam maiores. A técnica EbE desenvolvida neste trabalho foi proposta originalmente por [Hughes, Levit e Winget \(1983\)](#) como sendo um precondicionador para o CG de forma a possibilitar que as computações com matrizes sejam realizadas no nível elementar ([KISS et al., 2012](#)).

A motivação da escolha do *solver* do MATLAB® (“\” ou *mldivide*) está no fato de ele utilizar diferentes algoritmos dependendo da estrutura da matriz de coeficientes, tais como simetria e esparsidade . Especificamente para o problema deste trabalho, cuja matriz é Hermitiana (auto-adjunta) e positiva definida, o *solver* adota a fatoração de Cholesky ou a LDL.

A motivação para a escolha do CG implementado por [Barrett et al. \(1995\)](#) está no fato de que seu material é citado em diferentes trabalhos da revisão bibliográfica, como em [Kiss et al. \(2012\)](#), [Dolwightayakul, Chantrapornchai e Chumchob \(2012\)](#), [Yang, Liu e Chen \(2016\)](#). Além de um completo *survey* de métodos numéricos para a solução de sistemas lineares, os autores disponibilizam as implementações na linguagem do MATLAB® e em C++. O pseudocódigo 3 refere-se ao CG precondicionado com a matriz M .

Função	Descrição
load	Carrega para o <i>workspace</i> as variáveis salvas nos arquivos <i>.mat</i> .
decsg	Cria a geometria do problema por meio da associação de regiões primitivas. O espaço de análise do capacitor de placas paralelas, por exemplo é composto por 2 retângulos e 1 quadrado.
createpde	Instancia um modelo de PDE contendo n equações.
geometryFromEdges	Vincula a geometria originada da função <i>decsg</i> ao modelo de PDE gerado por <i>createpde</i> .
generateMesh	Cria uma malha sobre a geometria do modelo de PDE. Parâmetros adicionais podem ser incluídos para modificar a qualidade malha ou a ordem dos elementos (ver 2.2.4). O parâmetro <i>Hmax</i> determina o tamanho máximo das arestas dos elementos.
meshToPet	Obtém as matrizes de pontos, arestas e triângulos, as quais serão utilizadas na geração das matrizes elementares e da matriz global.
geraMatElementares	À partir dos dados da triangulação da malha, gera a matriz de cada elemento por meio das funções de aproximação e interpolação.
geraMatGlobal	Realiza a agregação ou o mapeamento das matrizes elementares no sistema global. A matriz resultante é esparsa e sua ordem corresponde ao número de nós da malha.
atribuiContorno	Atribui os m valores de contorno pré-estabelecidos e com isso realiza a redução dos sistemas em m ordens, fazendo com que deixe de ser homogêneo.
resolveSistLinear	Consiste na aplicação de métodos numéricos para a solução eficiente de sistemas lineares esparsos.
pdePlot	Função utilizada para a exibição da malha e dos resultados. Conforme os parâmetros adicionais, são plotados gráficos tridimensionais, campos vetoriais e linhas de campo.

Tabela 2 – Descrição das funções utilizadas no algoritmo do FEM

3.1.4 Implementação do EBE-FEM

A implementação do EBE-FEM consiste basicamente em antecipar a solução do sistema de equações, passando da montagem das matrizes elementares diretamente para a atribuição das condições de contorno e solução do sistema, sem que seja necessária a montagem da matriz de coeficientes global. Para realizar esta mudança de rota na solução do problema é necessário modificar tanto o algoritmo quanto a estrutura de dados

Algorithm 3 Pseudocódigo do CG

```

1:  $x_0 = \{0\}$ 
2:  $r_0 = b - Ax_0$ 
3: for  $i = 1, 2, \dots$  do
4:    $z_{i-1} = \text{resolve}(M, r_{i-1})$ 
5:    $\rho_{i-1} = r_{i-1}^T z_{i-1}$ 
6:   if  $i = 1$  then
7:      $p_1 = z_0$ 
8:   else
9:      $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ 
10:     $p_i = z_{i-1} + \beta_{i-1} p_{i-1}$ 
11:   end if
12:    $q_i = Ap_i$ 
13:    $\alpha_i = \rho_{i-1} / p_i^T q_i$ 
14:    $x_i = x_{i-1} + \alpha_i p_i$ 
15:    $r_i = r_{i-1} - \alpha_i q_i$ 
16:   if converge( $r, tol$ ) then
17:     break
18:   end if
19: end for

```

utilizados.

A adaptação do bloco referente à atribuição das condições de contorno (linha 11 do algoritmo 2) é baseada na abordagem proposta por Xu, Yin e Mao (2005) a qual consiste na criação de vetores *rhs* (*right hand side*) elementares. Ao invés de se atribuir os valores de contorno às células do vetor *rhs* global que correspondem à cada nó da malha, a atribuição é feita no nível elementar. Por exemplo, se um nó compartilhado por 2 elementos possui valor de contorno igual a 10, a atribuição é feita como uma média no vetor *rhs* de cada um dos 2 elementos. Assim sendo, cada vetor recebe o valor de contorno igual a 5. A figura 5 contém o esquema que ilustra este exemplo. Yan et al. (2017) e Wu et al. (2015) propõe uma abordagem parecida baseada não no número de conexões do nó, mas na média ponderada do valor de contorno original pelos coeficientes de cada elemento que contém o nó.

De posse das matrizes e vetores elementares com as condições de contorno atribuídas, é feita a adaptação no algoritmo do CG a fim de que os cálculos envolvendo matrizes sejam feitos elemento a elemento. As operações entre vetores ou vetores e escalares são mantidas no nível global, visto que não requerem tanto esforço computacional como as operações matriciais. As adaptações consistem basicamente em percorrer os N elementos, realizando os cálculos pertinentes e em seguida, armazenar nos vetores globais os resultados gerados para cada nó do elemento. As linhas 4 e 12 do algoritmo 3 possuem operações matriciais e são modificadas conforme apresentado nos pseudocódigos 4 e 5. Além disso, o resíduo inicial da linha 2 do 3 é igual a b , dado que o chute x_0 vale zero para todos os nós.

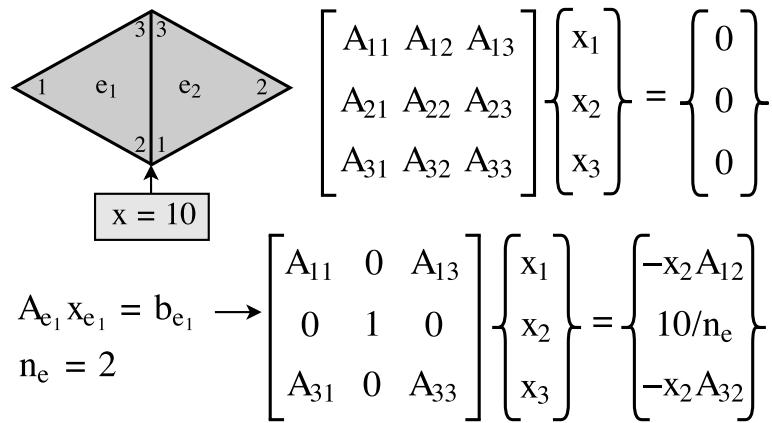


Figura 5 – Esquema da aplicação das condições de contorno de um elemento

Como os valores de b foram atribuídos pelas condições de contorno em nível elementar, é necessário realizar a montagem do vetor rhs global para determinar o resíduo inicial. Este processo de montagem é apresentado no pseudocódigo 6.

Algorithm 4 Adaptação da declaração $z_{i-1} = \text{resolve}(M, r_{i-1})$

```

1:  $z = \{0\}$ 
2: for  $j = 1, 2, \dots, |tri|$  do
3:    $map = tri(j)$ 
4:    $M_e = precond(A_e(j))$ 
5:    $z(map) = z(map) + M_e r(map)$ 
6: end for

```

Algorithm 5 Adaptação da declaração $q_i = A p_i$

```

1:  $q = \{0\}$ 
2: for  $j = 1, 2, \dots, |tri|$  do
3:    $map = tri(j)$ 
4:    $q(map) = q(map) + A_e(j)q(map)$ 
5: end for

```

Algorithm 6 Adaptação da declaração $r_0 = b - Ax_0$

```

1:  $b = \{0\}$ 
2: for  $j = 1, 2, \dots, |tri|$  do
3:    $map = tri(j)$ 
4:    $b(map) = b(map) + b_e(j)$ 
5: end for
6:  $r_0 = b$ 

```

3.1.5 Coloração da malha

Proposta por Wathen (1989) e implementada por Kiss et al. (2012) a coloração da malha possibilita que os trechos do CG descritos nos algoritmos 4, 5 e 6 sejam paralelizados.

A coloração assim como o processo de exclusão mútua são estratégias adotadas para evitar condições de corrida. Condição de corrida é o estado inconsistente dos dados causado por transações não atômicas. Desta forma, se por exemplo, dois processos ou *threads* alteram ao mesmo tempo um dado em memória compartilhada, não se pode prever o valor final do dado, mas sabe-se que será inconsistente. A coloração deve acontecer de tal forma que elementos que compartilhem um mesmo nó possuam cores distintas. Uma característica desejável num algoritmo otimizado é que haja uma distribuição equalizada na quantidade de elementos de cada cor, ao invés de uma quantidade mínima de cores. Por exemplo, numa malha de 1000 elementos é preferível se utilizar 20 cores com 50 elementos de cada cor, ao invés de 6 cores com uma distribuição desequilibrada de elementos, como por exemplo $\{500\ 50\ 200\ 150\ 10\ 90\}$. Contudo, como o objetivo da etapa de *validação do problema* não é obter resultados ótimos, foi adotada uma heurística gulosa sem balanceamento de cores, mas que apenas prioriza a aplicação das cores menos utilizadas. A figura 6 mostra um zoom na coloração da malha no topo entre as duas placas.

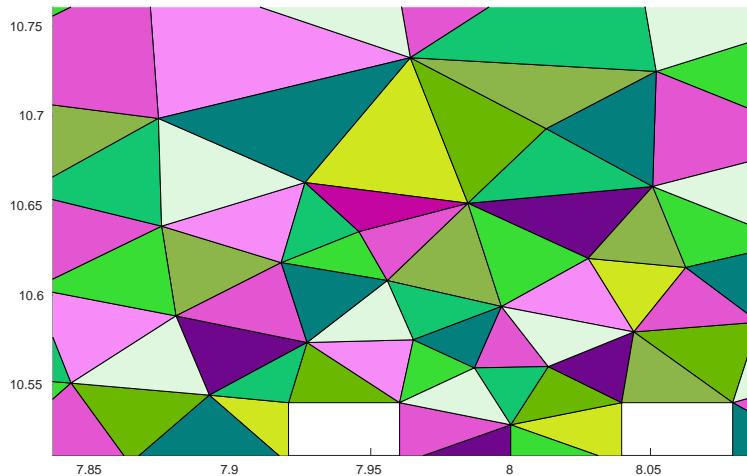


Figura 6 – Exemplo de coloração da malha

Uma vez realizada a coloração, o processamento pode ocorrer de tal forma que elementos da mesma cor seja executados simultaneamente e elementos de cores distintas sequencialmente. Como elementos de uma determinada cor não possuem vértices em comum, não há a possibilidade de acontecer condição de corrida durante o acesso às variáveis globais z e r do algoritmo 7 e q e d dos algoritmos 8 e 9 respectivamente. A estrutura de dados *cores* contém os dados da localização do elemento na malha.

3.1.6 Paralelização do algoritmo

A fim de manter a conformidade com o ambiente de desenvolvimento optou-se por utilizar a *parallel Computing toolbox* do MATLAB[®]. Esta ferramenta possibilita a implementação para plataformas *multicore*, GPUs, *clusters*, *grids* e *cloud*. De acordo com a documentação do MATLAB[®], funções de álgebra linear como *fft*, *mldivide*, *eig*,

Algorithm 7 Aplicação da coloração no algoritmo 4

```

1:  $z = \{0\}$ 
2: for  $c = 1,2,\dots|\text{cores}|$  do
3:    $\text{triCor} = \text{cores}(c)$ 
4:   for  $j = 1,2,\dots|\text{triCor}|$  do
5:      $\text{idG} = \text{triCor}(j)$ 
6:      $\text{map} = \text{tri}(\text{idG})$ 
7:      $M_e = \text{precond}(A_e(\text{idG}))$ 
8:      $z(\text{map}) = z(\text{map}) + M_e(\text{map})$ 
9:   end for
10: end for

```

Algorithm 8 Aplicação da coloração no algoritmo 5

```

1:  $q = \{0\}$ 
2: for  $c = 1,2,\dots|\text{cores}|$  do
3:    $\text{triCor} = \text{cores}(c)$ 
4:   for  $j = 1,2,\dots|\text{triCor}|$  do
5:      $\text{idG} = \text{triCor}(j)$ 
6:      $\text{map} = \text{tri}(\text{idG})$ 
7:      $q(\text{map}) = q(\text{map}) + A_e(\text{idG})q(\text{map})$ 
8:   end for
9: end for

```

Algorithm 9 Aplicação da coloração no algoritmo 6

```

1:  $b = \{0\}$ 
2: for  $c = 1,2,\dots|\text{cores}|$  do
3:    $\text{triCor} = \text{cores}(c)$ 
4:   for  $j = 1,2,\dots|\text{triCor}|$  do
5:      $\text{idG} = \text{triCor}(j)$ 
6:      $\text{map} = \text{tri}(\text{idG})$ 
7:      $b(\text{map}) = b(\text{map}) + b_e(\text{idG})$ 
8:   end for
9: end for
10:  $r_0 = b$ 

```

svd, e *sort* são *multithread* desde a versão R2008a. No entanto, é possível melhorar a performance de funções customizadas por meio da *parallel toolbox*.

A fim de garantir a robustez do código paralelizado, o MATLAB® disponibiliza a programação em múltiplos *workers* ao invés de *threads*. Cada *worker* é um processo que trabalha em função do processo principal (*client*) que o originou. Desta forma, as tarefas são divididas entre os *workers* e coordenadas pelo *client*, num modelo mestre-escravo. A fim de evitar condições de corrida e o *overhead* de comunicação entre os processos, a ferramenta trabalha por meio de distribuição e troca de mensagens (MPI) ao invés do compartilhamento de memória. Assim sendo, os dados são copiados para cada *worker* e ao final do processamento, agregados novamente no *client*. A tabela 3 contém a descrição das funções utilizadas da *parallel toolbox*.

Para que a paralelização do código ocorra por meio das funções *parfor* e *spmd* é necessário que a estrutura de dados da coloração seja modificada, a fim de que possa ser “fatiada” e distribuída entre os *workers*. Como pode ser visto nos algoritmos de 7 a 9, as variáveis globais são indexadas pela variável *map*, que contém a lista dos identificadores de cada nó do elemento, como por exemplo {25 12 96}. Como estes índices não são contíguos, ocorre um erro durante a compilação do *parfor* e o *overhead* de comunicação no *spmd*. Esta restrição do MATLAB® demanda uma nova adaptação da estrutura de dados e impede que a atribuição às variáveis globais seja feita de forma paralela.

A nova estrutura de dados gerada durante a coloração já retorna o vetor *b* (ou *rhs*) completo, dessa forma, não há a necessidade do processo de montagem à partir dos vetores elementares, como descrito nos algoritmos 6 e 9. Além dessa melhoria, o objeto retornado da coloração possui para cada cor, as matrizes elementares e o vetor *rhs* referente aos elementos da cor. Em resumo, a nova abordagem retorna os próprios objetos que serão utilizados na paralelização ao invés do seu “endereço” na malha. Os pseudocódigos 10 e 11 mostram a aplicação das funções *parfor* e *spmd* no algoritmo 7. A mesma adaptação é feita para o algoritmo 8.

Algorithm 10 Aplicação do *parfor* no algoritmo 7

```

1: for c = 1,2...|cores| do
2:   matElCor = cores(c).matEl
3:   rhsCor = cores(c).rhs
4:   R = reshape(r(rhsCor), 3, [ ])
5:   PARFOR
6:   for j = 1,2...|cores(c)| do
7:     Me = precond(matElCor(j))
8:     Z = MeR(j)
9:   end for
10:  END PARFOR
11:  z(rhsCor) = z(rhsCor) + reshape(Z, 1, [ ])
12: end for

```

Função	Descrição
gcp	<i>Get current parallel pool</i> retorna as informações sobre os <i>workers</i> ativos.
parpool	Cria uma nova <i>parallel pool</i> , ou seja, um conjunto de <i>workers</i> e os demais processos necessários para o funcionamento da <i>toolbox</i> .
parfor	Executa as iterações de um <i>loop</i> em <i>workers</i> . Para que isso seja possível as iterações devem ser independentes entre si e as estruturas de dados (coleções) utilizadas devem ter a característica de serem “fatiadas”. Por exemplo, uma matriz bidimensional pode ser fatiada de 2 formas, pelas linhas ou pelas colunas.
spmd	A função <i>single program, multiple data</i> executando o mesmo código em diferentes <i>workers</i> . Diferente do <i>parfor</i> , os dados não precisam necessariamente estar particionados, visto que o <i>spmd</i> permite a comunicação entre os <i>workers</i> .
codistributed	Cria uma coleção que é distribuída entre os <i>workers</i> e não copiada integralmente.
getLocalPart	Obtém os dados de uma determinada partição de uma coleção distribuída.
codistributor	Função que realiza a distribuição de uma coleção. Diferente da distribuição limitada que ocorre no <i>parfor</i> , essa função possibilita que a distribuição seja feita de diferentes formas em uma coleção.
gather	Função que realiza a agregação de uma coleção distribuída.
drange	Executa um <i>loop</i> entre as partições de uma coleção. Este bloco de código faz com que os <i>workers</i> trabalhem simultaneamente em cada distribuição da coleção, mas impede a comunicação entre eles.

Tabela 3 – Descrição das funções utilizadas na paralelização do EbE-FEM

Enquanto a paralelização por *parfor* é feita elemento a elemento, a função *spmd* distribui o processamento em q partes, sendo q a quantidade de *workers*. Cada *worker* executa sequencialmente a parcela do problema que lhe é atribuída. Conforme a recomendação da documentação, para se obter melhor desempenho, a quantidade de *workers* deve ser a mesma de núcleos físicos do processador. Ao se utilizar o bloco *for – drange* na linha 13 ao invés do *for* convencional, o comportamento do *spdm* passa a ser equivalente ao do *parfor*.

Algorithm 11 Aplicação do *spmd* no algoritmo.
[7](#)

```

1: for c = 1,2...|cores| do
2:   matElCor = cores(c).matEl
3:   rhsCor = cores(c).rhs
4:   R = reshape(r(rhsCor), 3, [ ])
5:   SPMD
6:   codMat = codistributor(formaDistMat)
7:   MatDist = codistributed(matElCor, codMat)
8:   codR = codistributor(formaDistR)
9:   RDist = codistributed(R, codR)
10:  MatPart = getLocalPart(MatDist)
11:  RPart = getLocalPart(RDist)
12:  for j = 1,2...qtdWorkers do
13:    for k = 1,2...|cores(c)| do
14:      Me = precond(MatPart(j, k))
15:      Z = MeRPart(j, k)
16:    end for
17:  end for
18:  END SPMD
19:  z(rhsCor) = z(rhsCor) + reshape(Z, 1, [ ])
20: end for

```

3.2 Resultados

Os resultados deste trabalho estão divididos nas subseções *resultados preliminares* e *resultados aplicados*. Na primeira subseção são apresentados os resultados da modelagem com o MATLAB® e nas subseções seguintes os resultados da aplicação do algoritmo desenvolvido nas linguagens C++, Scala, Erlang, CUDA e Harlan.

3.2.1 Resultados preliminares

O objetivo desta primeira etapa de atividades é validar o processo de paralelização do EbE-FEM na tentativa de antecipar as eventuais dificuldades que possam surgir na implementação do algoritmo nas linguagens concorrentes e paralelas e nas arquiteturas multi e *manycore*. As execuções foram realizadas em um notebook Dell Inspiron N4050 com processador i5-2450M, 2.50GHz e memória SO-DIMM DDR3 de 4 GiB.

Overview da triangulação

No início do desenvolvimento foram selecionados pontos de referência a fim de se estimar a precisão das aproximações, com base nos resultados da pdetool (figura 4). A *pdetool* fornece a opção de refinamento da malha, que consiste na subdivisão dos triângulos já existentes, no entanto, a opção de refinamento programático disponibilizada pela toolbox é dada pelo parâmetro *Hmax* da função *generateMesh* (algoritmo 2). Enquanto o

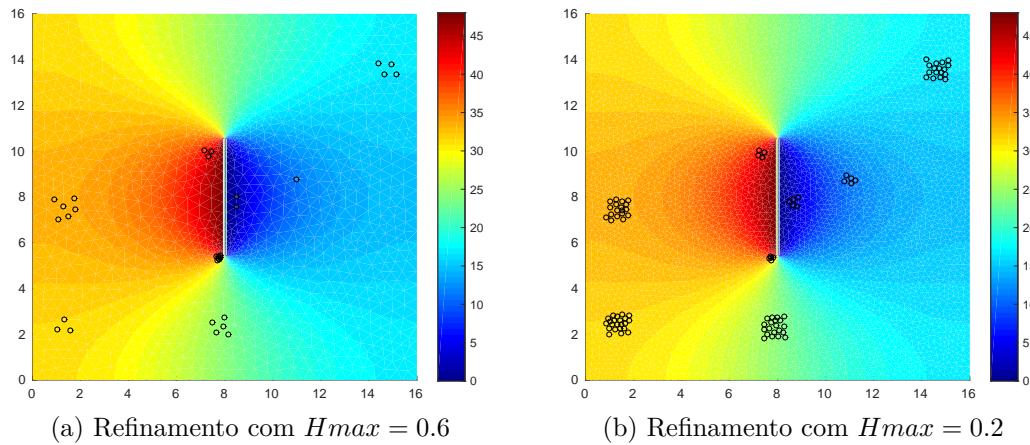


Figura 7 – Vizinhança dos pontos de referência

Hmax	Nós	Elementos
0.6	1700	3190
0.5	2196	4162
0.4	3154	6046
0.3	5287	10244
0.2	11439	22416
0.1	44692	88522
0.05	176971	352148

Tabela 4 – Mapeamento dos identificadores locais do circuito da figura 9

refinamento da *pdetool* mantém os pontos da malha original, a alteração do $Hmax$ gera uma nova malha, fazendo com que os pontos de referência se percam. Embora essa abordagem possibilite o maior controle da quantidade de elementos, ela impossibilita a verificação da precisão do valor de um ponto após o refinamento. A figura 7 destaca os nós da vizinhança de cada ponto de referência. Como não se pôde observar a precisão foi observado o valor médio próximo aos pontos de referência. A tabela 4 apresenta a quantidade de elementos e nós gerados para cada valor de $Hmax$.

Overview das implementações

Conforme apresentado no capítulo de materiais e métodos, foram desenvolvidas duas versões do algoritmo do EbE-FEM, sendo a primeira adequada para sistemas de memória compartilhada e *multithread* (algoritmos 7 a 9) e a segunda para sistemas de memória distribuída (10 e 11). A primeira versão do algoritmo proposta por Kiss et al. (2012) apresenta coloração e é paralelizável, no entanto, devido às restrições dos blocos *parfor* e *spmd* da *parallel computing toolbox* do MATLAB® foi possível testá-la apenas sequencialmente. A segunda implementação utiliza uma estrutura de dados distribuída e

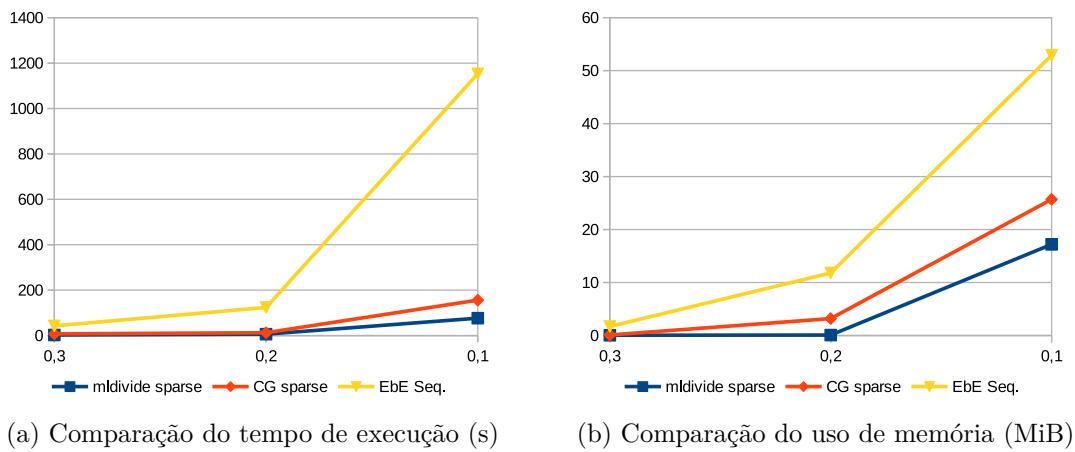


Figura 8 – Comparação do EbE com soluções sequenciais eficientes

desacoplada para atender aos critérios de “fatiamento” e independência dos blocos *parfor* e *spmd*.

A tabela 5 exibe os resultados das execuções com o uso de *solvers* sequenciais. As linhas indicadas com *x* mostram as limitações do processo de montagem da matriz global devido ao uso excessivo de memória. A mensagem de erro retornada pelo MATLAB® é *Requested 44692x44692 (14.9GB) array exceeds maximum array size preference*. A adoção do mecanismo de matrizes esparsas do MATLAB® no entanto possibilitou que o uso de memória e o percentual de CPU fossem reduzidos. Os *solvers* de matrizes esparsas e o EbE apresentaram um uso equilibrado da CPU em relação ao *mldivide* e *CG* com matrizes *full* que aumentaram o percentual do uso da CPU à medida que o número de elementos da malha aumentou. A figura 8 exibe a comparação do tempo de processamento e uso de memória dos *solvers* mais competitivos para os refinamentos 0.3, 0.2 e 0.1.

Como pode ser visto na tabela 6 o desempenho das implementações paralelas do EbE com coloração foram muito piores em relação a tempo de execução que a implementação sequencial. Como pode ser visto na última coluna, uma iteração do CG com *spmd* demorou cerca de 66 vezes mais que a versão sequencial e a solução completa do problema 48 vezes mais. Após dezenas de adaptações no algoritmo e nas estruturas de dados a fim de melhorar estes números foi verificado por meio da leitura mais aprofundada da documentação da *partool* que a existência dos comandos *parfor* e *spmd* dentro de *loops* não é recomendada visto que a cada iteração do *loop* externo ocorre troca de mensagens entre o processo *client* e os *workers*. Além disso, a documentação informa que iterações que apresentam operações simples e baixo tempo de execução não devem ser paralelizadas pela *toolbox* devido ao custo adicional de memória e comunicação para se criar e manter os *workers*.

Solver	Hmax	Tempo (s)	Memo (MiB)	%CPU
mldivide	0.6	0.390717	22.1	8
	0.5	0.524704	37	24
	0.4	0.911115	81.8	28
	0.3	2.572255	215.2	29
	0.2	21.337269	995	50
	0.1	x	x	x
mldivide Sparse	0.3	2.447401	0.05	25
	0.2	6.158577	0.08	26
	0.1	76.558448	17.2	29
CG	0.6	0.541897	32.9	19
	0.5	0.789027	36.2	16
	0.4	1.56392	71.3	27
	0.3	4.562658	210.7	51
	0.2	27.876141	1006.3	55
	0.1	x	x	x
CG Sparse	0.3	5.114425	0.1	29
	0.2	6.30303	3.2	31
	0.1	79.396158	25.7	33
	0.05	1379.017507	172.2	35
EBE Seq	0.6	6.082238	0.05	25
	0.5	8.84496	0.1	26
	0.4	16.518907	0.3	25
	0.3	35.125817	1.7	27
	0.2	111.13326	11.8	29
	0.1	996.990077	52.9	29

Tabela 5 – Tempo de Execução, consumo de memória e uso da CPU dos solvers sequenciais

	Sequencial	ParFor	SPMD	ParFor/seq.	SPMD/seq.
Tempo iter. (s)	0,002388	0,052530	0,120366	28,148739	65,683311
% CPU	28	74	42	2,642857	1,5
Tempo Total (s)	4,53238	84,250123	219,3455	18,58849	48,395230

Tabela 6 – Comparação das implementações paralelas para $Hmax = 1$

4 Conclusão

O EbE-FEM se mostrou uma estratégia competitiva para o processamento de sistemas esparsos em relação aos *solvers* otimizados *mldivide* e *CG*, sobretudo na demanda de memória. A paralelização adequada deste algoritmo por meio do uso de *threads* tende a ser promissora na melhoria do tempo de processamento apresentado na figura 8. Os desafios enfrentados na paralelização do algoritmo por meio dos blocos *parfor* e *spmd* possibilitaram a criação de duas versões do EbE, sendo uma de memória compartilhada e outra de memória distribuída, que facilitarão na adequação do código para as linguagens e arquiteturas que serão utilizadas na segunda etapa do trabalho.

Referências

- AHAMED, A.-K. C.; MAGOULÈS, F. Conjugate gradient method with graphics processing unit acceleration: CUDA vs OpenCL. *Advances in Engineering Software*, v. 47, n. 1, p. 164–169, 2016. ISSN 0965-9978. Disponível em: <<http://dx.doi.org/10.1016/j.advengsoft.2011.12.013>>. Citado 2 vezes nas páginas 4 e 7.
- ANZT, H. et al. Preconditioned Krylov solvers on GPUs. *Parallel Computing*, Elsevier B.V., v. 0, p. 1–13, 2016. ISSN 01678191. Disponível em: <<http://dx.doi.org/10.1016/j.parco.2017.05.006>>. Citado na página 4.
- BARRETT, R. et al. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. *Mathematics of Computation*, v. 64, n. 211, p. 1349, 1995. ISSN 00255718. Disponível em: <<http://www.jstor.org/stable/2153507?origin=crossref>>. Citado 2 vezes nas páginas 15 e 20.
- BOEHMER, S. et al. Numerical simulation of electrical machines by means of a hybrid parallelisation using MPI and OpenMP for finite-element method. *IET Science, Measurement & Technology*, v. 6, n. 5, p. 339, 2011. ISSN 17518822. Disponível em: <<http://digital-library.theiet.org/content/journals/10.1049/iet-smt.2011.0126>>. Citado na página 7.
- BOYCE, W. E.; DIPRIMA, R. C. *Equações Diferenciais Elementares e Problemas de Valores de Contorno*. [S.l.]: LTC, 2010. ISBN 978521617563. Citado na página 9.
- BOYLESTAD, R. L. *Análise de Circuitos*. [S.l.]: Pearson, 2011. ISBN 978-85-64574-20-5. Citado 2 vezes nas páginas 5 e 17.
- CAREY, G. F. et al. Element-by-element vector and parallel computations. *Communications in Applied Numerical Methods*, v. 4, n. 3, p. 299–307, 1988. ISSN 0748-8025. Disponível em: <<http://doi.wiley.com/10.1002/cnm.1630040303>>. Citado na página 7.
- CHOU, C.-y.; CHEN, K.-t. Performance Evaluations of Different Parallel Programming Paradigms for Pennes Bioheat Equations and Navier-Stokes Equations. p. 503–508, 2016. Citado na página 8.
- DAYDE, M. J.; L'EXCELLENT, J.-Y.; GOULD, N. I. M. On the Use of Element-by-Element Preconditioners to Solve Large Scale Partially Separable Optimization. *Computing*, 1995. Citado 2 vezes nas páginas 4 e 7.
- DESAI, J. A. C. S. *Introduction to the Finite Element Method: A Numerical Method for Engineering Analysis*. [S.l.]: Van Nostrand Reinhold Company, 1972. Citado 5 vezes nas páginas 10, 11, 12, 37 e 38.
- DOCUMENTATION: mldivide. <<https://www.mathworks.com/help/matlab/ref/mldivide.html>>. Acesso em: 03/08/2017. Nenhuma citação no texto.
- DOLWITHAYAKUL, B.; CHANTRAPORNCHAI, C.; CHUMCHOB, N. An efficient asynchronous approach for Gauss-Seidel iterative solver for FDM/FEM equations on

- multi-core processors. *JCSSE 2012 - 9th International Joint Conference on Computer Science and Software Engineering*, p. 357–361, 2012. Citado na página 20.
- ERLANG Market Share. <<https://w3techs.com/technologies/details/pl-erlang/all/all>>. Acesso em: 19/07/2017. Nenhuma citação no texto.
- FAIRES, R. L. B. J. D. *Análise Numérica*. [S.l.]: CENGAGE Learning, 2008. ISBN 978-85-221-0601-1. Citado 3 vezes nas páginas 10, 15 e 39.
- GUO, X. et al. Developing a scalable hybrid MPI/OpenMP unstructured finite element model. *Computers and Fluids*, Elsevier Ltd, v. 110, p. 227–234, 2014. ISSN 00457930. Disponível em: <<http://dx.doi.org/10.1016/j.compfluid.2014.09.007>>. Citado na página 5.
- HU, J.; QUIGLEY, S. F.; CHAN, A. An element-by-element preconditioned conjugate gradient solver of 3d tetrahedral finite elements on an FPGA coprocessor. *Proceedings - 2008 International Conference on Field Programmable Logic and Applications, FPL*, p. 575–578, 2008. Citado 2 vezes nas páginas 7 e 8.
- HUEBNER, K. H. et al. *The Finite Element Method for Engineers*. [S.l.]: John Wiley & sons, 2001. ISBN 978-0-471-37078-9. Citado 3 vezes nas páginas 10, 11 e 12.
- HUGHES, T.; LEVIT, I.; WINGET, J. Element-by-element implicit algorithms for heat conduction. *Journal of Engineering Mechanics*, v. 109, n. 2, p. 576–585, 1983. ISSN 07339399. Citado 3 vezes nas páginas 4, 7 e 20.
- IWASHITA, T. et al. Software framework for parallel BEM analyses with H-matrices. *IEEE CEFC 2016 - 17th Biennial Conference on Electromagnetic Field Computation*, Elsevier B.V., v. 108, n. June, p. 2200–2209, 2017. ISSN 18770509. Disponível em: <<http://dx.doi.org/10.1016/j.procs.2017.05.263>>. Citado na página 4.
- JIN, J. *The Finite Element Method in Electromagnetics*. [S.l.]: John Wiley & sons, 2002. ISBN 0471438189. Citado 5 vezes nas páginas 9, 10, 11, 12 e 13.
- KISS, I. et al. High locality and increased intra-node parallelism for solving finite element models on GPUs by novel element-by-element implementation. In: *2012 IEEE Conference on High Performance Extreme Computing, HPEC 2012*. [S.l.: s.n.], 2012. ISBN 9781467315760. Citado 6 vezes nas páginas 4, 7, 8, 20, 23 e 29.
- LEVIT, I. Element by element solvers of order n. v. 27, n. 3, p. 357–360, 1987. Citado na página 7.
- POWERS, D. L. *Boundary Value Problems and Partial Differential Equations*. [S.l.]: Elsevier, 2006. ISBN 139780125637381. Citado na página 9.
- PROCESSADOR Power9 da IBM. <<http://www.ibmsystemsmag.com/power/businessstrategy/competitiveadvantage/POWER9-Plans/>>. Acesso em: 19/07/2017. Nenhuma citação no texto.
- PROCESSADORES Intel. <<https://www.intel.com/content/www/us/en/products/processors.html>>. Acesso em: 19/07/2017. Nenhuma citação no texto.
- PRODUTOS AMD. <<http://www.amd.com/pt-br/products>>. Acesso em: 19/07/2017. Nenhuma citação no texto.

- PRODUTOS NVIDIA. <<http://www.nvidia.com.br/page/products.html>>. Acesso em: 19/07/2017. Nenhuma citação no texto.
- REDDY, J. N. *An Introduction to the Finite Element Method*. [S.l.]: McGraw Hill, 2006. ISBN 0071244735. Citado 2 vezes nas páginas 10 e 14.
- RUN MATLAB on multicore and multiprocessor machines. <<https://www.mathworks.com/discovery/matlab-multicore.html>>. Acesso em: 03/08/2017. Nenhuma citação no texto.
- SADIKU, M. N. O. *Numerical Techniques in Electromagnetics*. [S.l.]: CRC Press, 2001. ISBN 0-8493-1395-3. Citado 3 vezes nas páginas 10, 11 e 44.
- SCALA Market Share. <<https://w3techs.com/technologies/details/pl-scala/all/all>>. Acesso em: 19/07/2017. Nenhuma citação no texto.
- SZABO, B.; BABUŠKA, I. *An Introduction to Finite Element Analysis - Ch4 Generalized Formulations*. [S.l.: s.n.], 2009. 109–144 p. ISBN 9780470977286. Citado na página 4.
- VOLAKIS, J. L.; CHATTERJEE, A.; KEMPEL, L. C. *Finite Element Method for Electromagnetics*. [S.l.]: John Wiley & sons, 1998. ISBN 0780334256. Citado 2 vezes nas páginas 13 e 14.
- WATHEN, A. J. An analysis of some element-by-element techniques. *Computer Methods in Applied Mechanics and Engineering*, v. 74, n. 3, p. 271–287, 1989. ISSN 00457825. Citado 3 vezes nas páginas 4, 20 e 23.
- WU, D. et al. GPU Acceleration of EBE Method for 3-D Linear Steady Eddy Current Field. 2015. Citado 2 vezes nas páginas 8 e 22.
- XU, J.; YIN, W. Y.; MAO, J. Capacitance extraction of high-density 3D interconnects using finite element method. *Asia-Pacific Microwave Conference Proceedings, APMC*, v. 2, p. 3–5, 2005. Citado 2 vezes nas páginas 8 e 22.
- YAN, X. et al. Research on Preconditioned Conjugate Gradient Method Based on EBE-FEM and the Application in Electromagnetic Field Analysis. v. 53, n. 6, 2017. Disponível em: <http://www.ieee.org/publications{_}standards/publications/rights/index.h>. Citado 2 vezes nas páginas 8 e 22.
- YANG, B.; LIU, H.; CHEN, Z. Preconditioned GMRES solver on multiple-GPU architecture. *Computers and Mathematics with Applications*, Elsevier Ltd, v. 72, n. 4, p. 1076–1095, 2016. ISSN 08981221. Disponível em: <<http://dx.doi.org/10.1016/j.camwa.2016.06.027>>. Citado na página 20.
- YAO, L. et al. Parallel implementation and performance comparison of BiCGStab for massive sparse linear system of equations on GPU libraries. *Proceedings - 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing, 2015 IEEE 12th International Conference on Advanced and Trusted Computing, 2015 IEEE 15th International Conference on Scalable Computing and Communications, 20*, p. 603–608, 2015. Citado 2 vezes nas páginas 4 e 8.
- ZIENKIEWICZ, R. L. T. J. Z. C. *The Finite Element Method its Basis & Fundamentals*. [S.l.]: Elsevier, 2005. ISBN 0750663200. Citado 7 vezes nas páginas 10, 11, 13, 37, 39, 40 e 41.

Apêndices

APÊNDICE A – Sistemas de elementos discretos

(DESAI, 1972, p. 68) introduz o conceito de Método de Elementos Discretos (MED) como sendo uma etapa intermediária da formulação física MEF. Na representação em elementos discretos, cada elemento é unidimensional, de tal forma que o sistema completo é representado por uma estrutura aramada, conforme mostra a figura 9.

De forma similar, (ZIENKIEWICZ, 2005, p. 2) apresenta o Sistema Discreto Padrão (SDP) como uma forma unificada de analisar problemas de natureza discreta.

Em ambos os casos, os sistemas originais não se tratam de um contínuo, propriamente dito, mas de um sistema composto por partes. Cada parte por si só é um componente físico funcional, seja ela uma viga, um tubo ou um resistor, por exemplo. Entender o sistema de elementos discretos é importante para compreensão do funcionamento do MEF (ZIENKIEWICZ, 2005, p. 6), cujos elementos (quadrados, triângulos ou tetraedros) não são funcionais ou não possuem um significado físico fora do sistema.

A.1 Modelo de Elementos Discretos

Considere a malha bidimensional apresentada na figura 9. Esta malha pode representar por exemplo, a abstração de uma estrutura mecânica (pórtico), de um sistema hidráulico ou de circuito elétrico, como o apresentado na figura 10.

Na figura 9, os elementos são identificados por números (sem parentesis) de 1 a 6. A identificação de cada nó, ou junção, é feita entre parêntesis. Na figura 9(a), em negrito, cada nó é relacionado à um identificador global, enquanto na figura 9(b), os nós possuem identificadores locais, relativos não ao conjunto total de nós, mas apenas aos nós de um dado elemento. Estes identificadores são necessários para a análise de um único elemento e em seguida para a montagem do sistema como um todo.

Fazendo a analogia com um pórtico, as forças que atuam sobre cada nó, podem ser definidas a partir dos deslocamento destes nós (ZIENKIEWICZ, 2005, p. 3). Esta relação é dada pela lei de Hooke simples, mostrada na equação A.1, na qual as forças Q_1 e Q_2 sobre os nós de um elemento são dadas calculadas em função dos deslocamentos u_1 e u_2 . A constante A é a área do elemento, E é o módulo de Young (Módulo de Elasticidade) e l

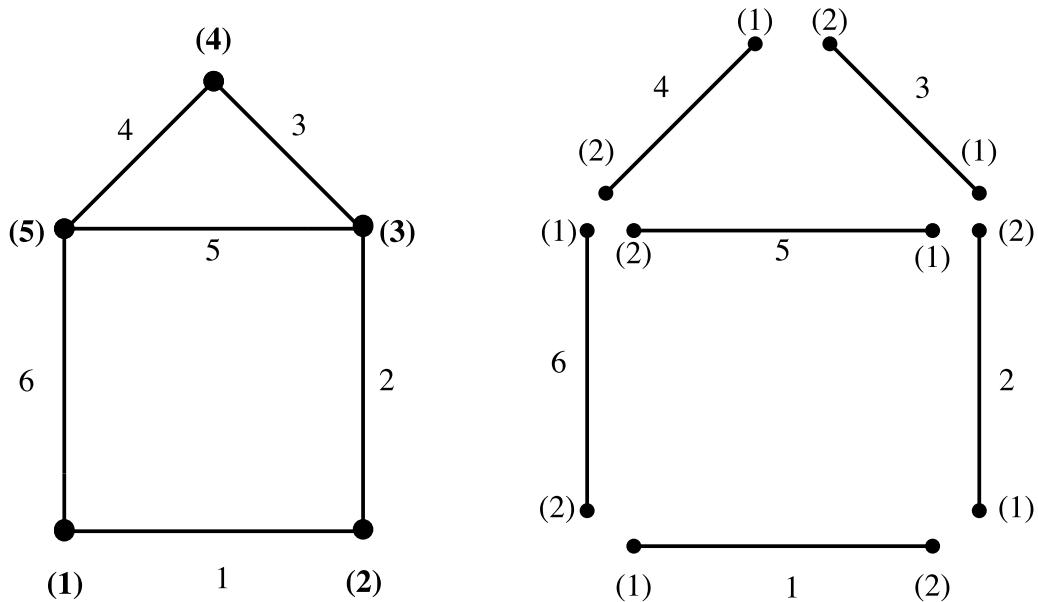


Figura 9 – identificação global e local

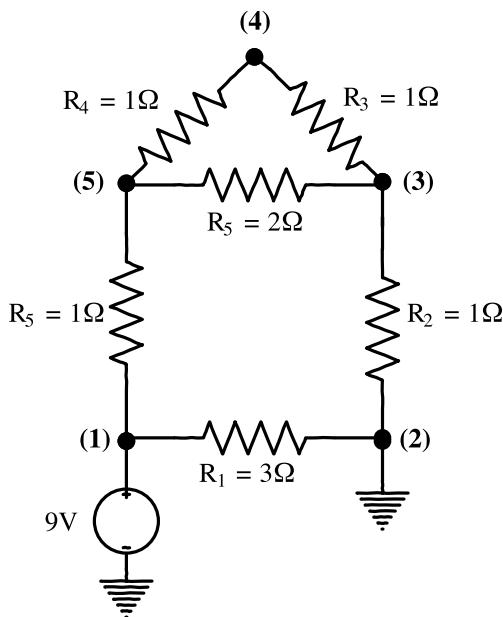


Figura 10 – Circuito puramente resistivo, um possível objeto de representação da malha da figura 9

é o cumprimento do elemento (DESAI, 1972, p. 13).

$$\begin{Bmatrix} Q_1 \\ Q_2 \end{Bmatrix} = \frac{AE}{l} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \quad (\text{A.1})$$

No circuito resistivo da figura 10, a relação para um elemento é dada pela Lei de Ohm, na qual as correntes em cada nó, podem ser definidas à partir das tensões nos terminais do elemento. Na equação A.2, I_a e V_a são respectivamente a corrente e a tensão

no nó a de um elemento, enquanto R é a resistência do mesmo (ZIENKIEWICZ, 2005, p. 7).

$$\begin{Bmatrix} I_1 \\ I_2 \end{Bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} V_1 \\ V_2 \end{Bmatrix} \quad (\text{A.2})$$

O padrão encontrado nas equações A.1 e A.2 pode ser generalizado para outros campos da engenharia, ciência e até mesmo administração e economia, desde que possam ser modelados por sistema de equações lineares (FAIRES, 2008, p. 331).

A questão a ser respondida é: Como analisar o efeito global de uma ação realizada em um conjunto limitado de nós? Por exemplo, como obter o potencial elétrico nos nós 3, 4 e 5, da figura 10, a partir da aplicação das tensões 9V e 0V nos nós 1 e 2 respectivamente? Como compilar os resultados locais para o cálculo de grandezas secundárias, como a corrente ou potência elétrica em cada elemento, por exemplo? Estas questões serão respondidas nas duas subseções seguintes.

sectionEquações do Elemento Discreto

Se um fenômeno que atua sobre um dado elemento e é modelado por um sistema de equações lineares, pode ser representado em sua forma matricial como:

$$\{q^e\} = [K^e]\{u^e\} + \{f^e\} \quad (\text{A.3})$$

O vetor q é conhecido ou parcialmente conhecido. Ele é geralmente uma grandeza secundária, calculada a partir da distribuição da grandeza primária u desconhecida. A matriz K é a matriz de coeficientes e determina como é feita a distribuição da grandeza u . No exemplo de estruturas, K é a matriz de rigidez, q é o conjunto de forças que atuam nos nós do elemento e u é o deslocamento destes nós causados por estas forças. Para circuitos elétricos, K relaciona as resistências do sistema, q é a corrente no elemento, originada pela diferença de potencial u em seus nós. O vetor f é o estado inicial do elemento. Caso este estado seja de equilíbrio (sem torção inicial ou corrente residual, por exemplo), f é igual a zero (ZIENKIEWICZ, 2005, p. 7).

As componentes de cada vetor da equação A.3 são relativas aos nós do elemento e a ordem da matriz K é igual ao número de nós deste elemento. O número de nós de um elemento também é denominado **grau de liberdade** (ZIENKIEWICZ, 2005, p. 5). A equação A.4 mostra a representação de um elemento com grau de liberdade m . Se vista

como uma matriz de adjacência, K mostra como cada nó se relaciona com os demais.

$$q^e = \begin{Bmatrix} q_1^e \\ q_2^e \\ \vdots \\ q_m^e \end{Bmatrix} \quad u^e = \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{Bmatrix} \quad K^e = \begin{bmatrix} K_{11}^e & K_{12}^e & \dots & K_{1m}^e \\ K_{11}^e & \ddots & & \vdots \\ \vdots & \vdots & & \vdots \\ K_{m1}^e & \dots & \dots & K_{mm}^e \end{bmatrix} \quad (\text{A.4})$$

Como pode ser visto na figura 9(b), cada elemento compartilha nós com seus vizinhos, de forma que, o resultado obtido no nó de um elemento deve ser compatível com os resultados obtidos nos outros elementos, para o mesmo nó. Esta condição de compatibilidade e equilíbrio será vista na subseção seguinte.

A.2 Montagem do Sistema Global

Para que uma ação local em um nó seja corretamente transmitida para os nós vizinhos, e dessa forma tenha seus efeitos refletidos em todo o sistema, é necessário relacionar todos os elementos e nós em um único sistema de equações. A equação A.5 mostra um sistema global genérico. Sendo n o número total de nós, q , u e f são vetores coluna de comprimento n e K é uma matriz quadrada de ordem n .

$$\{q\} = [K]\{u\} + \{f\} \quad (\text{A.5})$$

A montagem do sistema A.5 consiste no mapeamento dos sistemas elementares para um único sistema global e na aplicação das condições de equilíbrio pertinentes ao problema (ZIENKIEWICZ, 2005, p. 5).

No mapeamento, a matriz global de coeficientes pode ser vista como uma matriz de adjacência de todos os nós. Desta forma, ações locais sobre um elemento podem ser propagadas para os elementos vizinhos. Enquanto nas equações elementares os nós são referenciados localmente, como na figura 9(b), no sistema global cada nó possui uma única referência, conforme a figura 9(a). As influências de diferentes elementos sobre um nó são somadas na matriz global, de forma que se tenha uma influência resultante.

O próximo passo é a aplicação da condição de equilíbrio para cada nó, conforme a lei que modela o problema. Para problemas de mecânica, por exemplo, a condição de equilíbrio dada pela 3^a lei de Newton diz que o somatório das forças em um ponto é igual a zero. Similarmente na análise de circuitos, a lei de Kirchoff diz que a soma das correntes que entram e saem de um nó também é igual a zero. A equação A.6 mostra a aplicação da

condição de equilíbrio no nó a .

$$\sum_{e=1}^m q_a^e = q_a^1 + q_a^2 + \dots = 0 \quad (\text{A.6})$$

Considerando que o estado inicial f do corpo é nulo, o sistema [A.5](#) que possui m nós e n elementos, pode ser representado como

$$q = \sum_{b=1}^n \sum_{e=1}^m K_{ab}^e u_b = 0 \quad (\text{A.7})$$

A.3 Atribuição das condições de Contorno

A atribuição dos valores de contorno consiste na imposição dos valores preestabelecidos da variável u . No exemplo [10](#), as condições de contorno são os valores da tensão u impostos nos nós 1 e 2, como mostra a equação a seguir

$$u = \begin{Bmatrix} 9V \\ 0V \\ u_3 \\ u_4 \\ u_5 \end{Bmatrix} \quad (\text{A.8})$$

Com isso, a inserção de valores de contorno no sistema promove a redução do número de equações de equilíbrio do sistema apresentado em [A.7](#), por meio da eliminação das linhas cujo valor de u já foi especificado ([ZIENKIEWICZ, 2005](#), p. 5).

A.4 Exemplo: Associação de resistores

A partir dos passos apresentados nas subseções acima, deve-se determinar os valores das tensões nos nós 3, 4 e 5 à partir da diferença de potencial entre os nós 1 e 2.

O primeiro passo para a resolução do problema é a especificação das equações de cada elemento. Como é apresentado na equação [A.2](#), as correntes em cada nó de um elemento são dadas pelas diferenças de potencial destes nós. Com isso, para cada resistor,

Identificação Local	Identificação Global
e_1^1, e_2^6	1
e_2^1, e_1^2	2
e_2^2, e_1^3, e_1^5	3
e_2^3, e_1^4	4
e_2^4, e_2^5, e_1^6	5

Tabela 7 – Mapeamento dos identificadores locais do circuito da figura 9

tem-se:

$$\begin{aligned}
 e = 1 : & \left\{ \begin{array}{l} I_1^1 = \frac{1}{3}(V_1^1 - V_2^1) \\ I_2^1 = \frac{1}{3}(-V_1^1 + V_2^1) \end{array} \right. & e = 2 : & \left\{ \begin{array}{l} I_1^2 = V_1^2 - V_2^2 \\ I_2^2 = -V_1^2 + V_2^2 \end{array} \right. \\
 e = 3 : & \left\{ \begin{array}{l} I_1^3 = V_1^3 - V_2^3 \\ I_2^3 = -V_1^3 + V_2^3 \end{array} \right. & e = 4 : & \left\{ \begin{array}{l} I_1^4 = V_1^4 - V_2^4 \\ I_2^4 = -V_1^4 + V_2^4 \end{array} \right. \\
 e = 5 : & \left\{ \begin{array}{l} I_1^5 = \frac{1}{2}(V_1^5 - V_2^5) \\ I_2^5 = \frac{1}{2}(-V_1^5 + V_2^5) \end{array} \right. & e = 6 : & \left\{ \begin{array}{l} I_1^6 = V_1^6 - V_2^6 \\ I_2^6 = -V_1^6 + V_2^6 \end{array} \right.
 \end{aligned} \tag{A.9}$$

Uma vez definidas as equações de cada elemento é feita a montagem do sistema global. Conforme dito anteriormente, a etapa de montagem consiste no mapeamento da numeração local dos nós para um sistema de numeração global. Analisando a figura 9(b), é possível notar que os nós 1 e 2 do elemento 2, por exemplo, são os nós 2 e 3 do sistema global apresentado na figura 9(a). A tabela 7 relaciona os identificadores locais e seus equivalentes no sistema global.

A equação A.10 a seguir, apresenta o mapeamento do elemento 1 e em seguida, na equação A.11, é incluído o mapeamento do elemento 2.

$$\left\{ \begin{array}{l} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \right\} = \left[\begin{array}{ccccc} \frac{1}{3} & \frac{-1}{3} & 0 & 0 & 0 \\ \frac{-1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \left\{ \begin{array}{l} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \right\} \tag{A.10}$$

$$\left\{ \begin{array}{l} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \right\} = \left[\begin{array}{ccccc} \frac{1}{3} & \frac{-1}{3} & 0 & 0 & 0 \\ \frac{-1}{3} & \frac{1}{3} + 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \left\{ \begin{array}{l} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \right\} \quad (\text{A.11})$$

Todas as contribuições em um dado nó, devem ser consideradas. Por este motivo, quando houverem influências de diferentes elementos sobre um nó, as mesmas devem ser somadas no sistema global. A equação A.12 apresenta o sistema global final, com as contribuições de todos os 6 elementos.

$$\left\{ \begin{array}{l} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{array} \right\} = \left[\begin{array}{ccccc} \frac{1}{3} & \frac{-1}{3} & 0 & 0 & 0 \\ \frac{-1}{3} & \frac{4}{3} & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{array} \right] \left\{ \begin{array}{l} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \right\} \quad (\text{A.12})$$

APÊNDICE B – Método dos elementos finitos em 2 dimensões

Neste capítulo será apresentada a solução de um modelo de elementos finitos. O exemplo desenvolvido foi adaptado do exemplo 6.1 de Sadiku (2001). Deseja-se saber o valor do potencial elétrico nos pontos a e b da placa submetida a uma diferença de potencial de $10V$, como mostra a figura 11a.

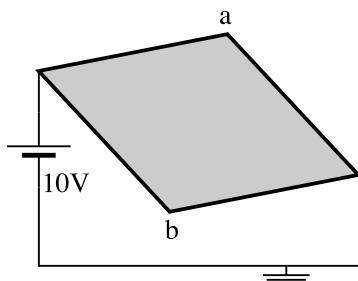
O primeiro passo consiste em realizar a triangulação da malha. Uma possibilidade de malha é apresentada na figura 11[b]. A estrutura de dados *tri* contém a informação sobre quais vértices formam cada elemento enquanto a estrutura *nos* possui as coordenadas de todos os vértices da geometria do problema.

O passo seguinte é a definição da função de interpolação. A aproximação da distribuição do potencial num dado elemento pode ser feita por uma função linear, como mostra a equação B.1. Desta forma, os valores de V em cada nó podem ser obtidos pelo sistema de equações B.2

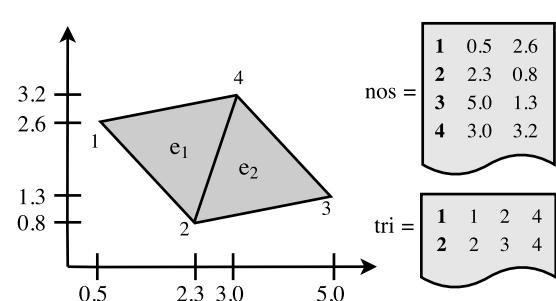
$$\tilde{V}^e = a^e + b^e x + c^e y \quad (\text{B.1})$$

$$\begin{Bmatrix} \tilde{V}_1^e \\ \tilde{V}_2^e \\ \tilde{V}_3^e \end{Bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} \quad (\text{B.2})$$

Deseja-se então resolver o problema encontrando os valores dos coeficientes a , b e c que fazem a aproximação de V . Dada a simplicidade o sistema ele pode ser resolvido invertendo-se a matriz por meio do cálculo dos determinantes e cofatores. Os valores da



(a) Geometria do problema



(b) Triangulação do problema

Figura 11 – Exemplo do FEM em 2 dimensões

transposta da matriz de cofatores são dados na equação B.3 e o determinante na equação B.4. A equação B.5 contém a substituição na equação B.1 dos valores dos coeficientes calculados.

$$cof(A^e)^T = \begin{bmatrix} (x_2y_3 - x_3y_2) & (x_3y_1 - x_1y_3) & (x_1y_2 - x_2y_1) \\ (y_2 - y_3) & (y_3 - y_1) & (y_1 - y_2) \\ (x_3 - x_2) & (x - 1 - x_3) & (x_2 - x_1) \end{bmatrix} \quad (\text{B.3})$$

$$\det(A^e) = (x_1y_2 - x_2y_1) + (x_3y_1 - x_1y_3) + (x_2y_3 - x_3y_2) \quad (\text{B.4})$$

$$\tilde{V}^e = \begin{Bmatrix} 1 & x & y \end{Bmatrix} \begin{Bmatrix} a \\ b \\ b \end{Bmatrix} = \begin{Bmatrix} 1 & x & y \end{Bmatrix} \frac{cof(A^e)^T}{\det(A^e)} \begin{Bmatrix} \tilde{V}_1^e \\ \tilde{V}_2^e \\ \tilde{V}_3^e \end{Bmatrix} = \sum_{j=1}^3 N_j^e(x, y) \tilde{V}_j^e \quad (\text{B.5})$$

Obtidas as funções de interpolação N^e , a próxima etapa consiste na montagem do sistema de equações do problema, começando pela definição das matrizes elementares. A distribuição do potencial é modelada pela equação de Laplace $\nabla^2 V = 0$. Substituindo esta equação em 2.8 e adotando-se o método de Galerkin, tem-se a integral B.6.

$$\iint N_i^e \nabla^2 V^e dx dy = R_i^e \quad i = 1, 2, 3. \quad (\text{B.6})$$

Aplicando a integração por partes fazendo $u = N$ e $dv = \nabla^2 V$, tem-se como resultado da relação $\int u dv = uv - \int v du$ a equação B.7.

$$\iint \nabla V^e \nabla N_i^e dx dy = R_i^e \quad i = 1, 2, 3. \quad (\text{B.7})$$

Substituindo V^e por sua aproximação obtida em (B.5) obtém-se a fórmula da matriz elementar nas equações B.8 e B.9.

$$\iint \sum_{j=1}^3 \nabla N_j^e \tilde{V}_j^e \nabla N_i^e dx dy = \sum_{j=1}^3 \iint (\nabla N_i^e \nabla N_j^e) \tilde{V}_j^e dx dy = R_i^e \quad i = 1, 2, 3. \quad (\text{B.8})$$

$$K_{ij}^e = \iint (\nabla N_i^e \nabla N_j^e) dx dy \quad (\text{B.9})$$

De forma a agilizar os cálculos das matrizes elementares define-se na equação B.10 a partir dos elementos das equações B.3 e B.4 as variáveis P , Q e A . P e Q são obtidas a

partir das simplificações nos cálculos e A é a área do triângulo. A equação B.11 mostra como os coeficientes podem ser calculados após a simplificação

$$\begin{aligned} P_1 &= (y_2 - y_3) & P_2 &= (y_3 - y_1) & P_3 &= (y_1 - y_2) \\ Q_1 &= (x_3 - x_2) & Q_2 &= (x_1 - x_3) & Q_3 &= (x_2 - x_1) \\ A &= \frac{1}{2}(P_2Q_3 - P_3Q_2) \end{aligned} \quad (\text{B.10})$$

$$K_{ij}^e = \frac{1}{4A}(P_iP_j + Q_iQ_j) \quad (\text{B.11})$$

As matrizes dos elementos e a matriz global são apresentadas nas equações B.12 e B.13. Após a definição das matrizes elementares é realizado o mapeamento de seus coeficientes para a matriz global. Enquanto a matriz elementar é da ordem do número de nós do elemento, a matriz global de coeficientes é da ordem do número de nós da malha. O processo de mapeamento é detalhado no apêndice A.

$$K^1 = \begin{pmatrix} 0,560 & -0,285 & -0,274 \\ -0,285 & 0,592 & -0,3066 \\ -0,274 & -0,306 & 0,580 \end{pmatrix} \quad K^2 = \begin{pmatrix} 0,620 & -0,257 & -0,362 \\ -0,257 & 0,509 & -0,252 \\ -0,362 & -0,252 & 0,615 \end{pmatrix} \quad (\text{B.12})$$

$$K = \begin{pmatrix} 0,560 & -0,285 & 0 & -0,274 \\ -0,285 & 1,213 & -0,257 & -0,669 \\ 0 & -0,257 & 0,509 & -0,252 \\ -0,274 & -0,669 & -0,252 & 1,195 \end{pmatrix} \quad (\text{B.13})$$

A atribuição dos valores de contorno consiste na redução da ordem na matriz B.13 por meio da eliminação dos graus de liberdade (nós) com valores pré estabelecidos (contorno). As equações B.14 e B.15 exibem o sistema reduzido e a solução.

$$\begin{pmatrix} 0,560 & -0,285 & 0 & -0,274 \\ -0,285 & 1,213 & -0,257 & -0,669 \\ 0 & -0,257 & 0,509 & -0,252 \\ -0,274 & -0,669 & -0,252 & 1,195 \end{pmatrix} \begin{Bmatrix} 10 \\ V^2 \\ 0 \\ V^4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (\text{B.14})$$

$$\begin{pmatrix} -0.285 & -0.257 \\ -0.274 & -0.252 \end{pmatrix} \begin{Bmatrix} V^2 \\ V^4 \end{Bmatrix} = \begin{Bmatrix} 2.858 \\ 2.741 \end{Bmatrix} \Rightarrow \begin{Bmatrix} V^2 \\ V^4 \end{Bmatrix} = \begin{Bmatrix} 5.241 \\ 5.227 \end{Bmatrix} \quad (\text{B.15})$$