ESCOLA POLITÉCNICA
DA
UNIVERSIDADE DE SÃO PAULO

DEPARTAMENTO DE ENGENHARIA DE ENERGIA E AUTOMAÇÃO ELÉTRICAS

Desenvolvimento de Ferramenta Computacional para Cálculo de Malhas de Aterramento pelo Método de Elementos Finitos Baseada em Programas de Distribuição Livre

Marco Antonio Mendes Afonso Signori

PROJETO DE FORMATURA / 2007

ESCOLA POLITÉCNICA
DA
UNIVERSIDADE DE SÃO PAULO

DEPARTAMENTO DE ENGENHARIA DE ENERGIA E
AUTOMAÇÃO ELÉTRICAS

Desenvolvimento de Ferramenta Computacional para Cálculo de Malhas de Aterramento pelo Método de Elementos Finitos Baseada em Programas de Distribuição Livre

ALUNO: Marco Antonio Mendes Afonso Signori
ORIENTADORA: Viviane Cristine Silva
COORDENADOR: Lourenço Matakas Júnior

Dedicatória

À minha esposa, Tatiana, por ser um exemplo de profissional, mulher e esposa, e por ter me ensinado a ser um homem melhor a cada dia, e a sorrir e lutar pelos meus objetivos, e por me ensinar o que é amar sem fronteiras e sem medidas.

À Laika, por ser um exemplo de coragem e força de vontade, por ter sobrevivido a tantos desafios e continuar sempre feliz, brincalhona e nunca ter desistido de nada, desde uma simples corrida até mesmo a passar por uma cirurgia cardíaca.

À Clara e ao Anakin, por serem exemplos de um amor incondicional, seja no meu ódio ou na minha alegria, e por terem me feito sorrir nos momentos mais tristes e difíceis que eu passei, e por me ensinar que uma brincadeira pode transformar o dia mais cinzento no dia mais alegre.

Agradecimentos

À minha professora e orientadora, Prof.ª Dr.ª Viviane Cristine Silva, por ter me apoiado na sua forma neste projeto, e por ter acreditado que eu era capaz de desenvolver algo de tal ambição, fornecendo-me o material de apoio necessário para o desenvolvido deste projeto.

Ao Prof. Walter Kaiser, pela sua pressão e apoio para que eu atingisse a graduação, e por ser um exemplo de uma amizade, mesmo que na sua forma peculiar de demonstrá-la.

Ao Departamento de Engenharia Elétrica de Energia e Automação, por ter me proporcionado as maiores experiências da minha vida, e por permitir que eu conseguisse atingir um nível de profissional durante a graduação, além de ter conhecido as novas amizades da minha vida.

Ao Prof. PhD. Christophe Geuzaine, do Departamento de Engenharia Elétrica e Ciência da Computação da University of Liège (Bélgica), por ter respondido pacientemente as minhas dúvidas referentes ao Gmsh e à minha técnica de implementação do mesmo neste trabalho, mesmo que isso demandasse um tempo do qual ele não dispunha, e por ser um exemplo de contribuição à comunidade de código livre.

Ao Prof. Lourenço Matakas Júnior, por ter sido compreensivo com a minha falta de tempo, mas mesmo assim orientando e procurando resolver os conflitos que o trabalho teve, sempre disposto a ouvir e opinar de forma conveniente para o bom andamento do trabalho.

Resumo

SIGNORI, M. A. M. A.; SILVA, V. C. (Orientadora) - *Desenvolvimento de Ferramenta Computacional para Cálculo de Malhas de Aterramento pelo Método de Elementos Finitos Baseada em Programas de Distribuição Livre* – Projeto de Formatura, Departamento de Energia e Automação Elétricas da Escola Politécnica da Universidade de São Paulo – PEA/EPUSP, São Paulo, 2007.

Os sistemas de aterramento na atualidade são de vital importância para os sistemas de transmissão e distribuição de energia elétrica. Uma metodologia que auxilia de forma crucial o projeto de tais sistemas torna-se então algo fundamental. Porém, devido às leis físicas que regem os fenômenos envolvidos, a solução analítica de projetos complexos torna-se inviável.

É por isso que os estudos dos sistemas de aterramento buscam nos métodos numéricos os passos para o projeto de tais sistemas. Dado o estado da arte do Método de Elementos Finitos, este se mostra bastante capacidade para resolver desde problemas simples até problemas bastante complexos. Por isso, a aplicação de tal método para a nossa análise é algo que merece uma especial atenção.

Atualmente, os programas de domínio público (programas de código livre) têm ganhado força no meio acadêmico. Não só por permitirem o intercâmbio de informação, mas também por possuírem uma política intrínseca de difusão de conhecimento. Embora existam pacotes computacionais comerciais voltados para a simulação por Método de Elementos Finitos de sistemas de aterramento, uma aplicação baseada em ferramentas livres é uma solução interessante não só do ponto de vista econômico, mas também acadêmico. Isso porque permite a contribuição de diversas entidades na aplicação, além de ser possível validá-la e verificar possíveis falhas não detectadas anteriormente, mesmo que não seja pelos desenvolvedores originais.

Este trabalho utilizou-se de pacotes computacionais de domínio livre difundidos no meio acadêmico para análise de problemas baseados no Método de Elementos Finitos. Embora os pacotes sejam independentes, aqui estes pacotes irão trabalhar de forma conjunta e organizada. Desta forma, podemos obter um sistema totalmente baseado em ferramentas livres, capaz de auxiliar no projeto de sistemas de aterramento.

Para a validação do modelo e das integrações adotadas, serão estudados dois casos com resultados já conhecidos. Devido à dificuldade de obtenção de um projeto de sistema de aterramento, a comparação irá ser feita em relação a um caso de solução analítica razoavelmente fácil, largamente conhecido na literatura, e em relação a um caso muito estudado e simulado com pacotes computacionais de alta credibilidade na área, sendo seus resultados considerados bons parâmetros.

Assim busca-se então a criação e validação de um pacote computacional de domínio público. A utilização de outros pacotes de outras comunidades permite o aperfeiçoamento do programa somente utilizando atualizações destes pacotes, mostrando então não só a força do Método de Elementos Finitos para a análise de problemas complexos na área de Engenharia Elétrica, mas também a força do uso de programas de domínio público, permitindo a difusão, troca e ampliação de conhecimento entre as comunidades acadêmicas no mundo.

Palavras-chave: Método de Elementos Finitos, simulação computacional, malhas de aterramento, sistemas de aterramento, programas de domínio público, Gmsh, ITL, LMAGLIB.

Abstract

SIGNORI, M. A. M. A.; SILVA, V. C. (Orientadora) - *Desenvolvimento de Ferramenta Computacional para Cálculo de Malhas de Aterramento pelo Método de Elementos Finitos Baseada em Programas de Distribuição Livre* – Projeto de Formatura, Departamento de Energia e Automação Elétricas da Escola Politécnica da Universidade de São Paulo – PEA/EPUSP, São Paulo, 2007.

The grounding systems nowadays are a vital part to the transmission and distribution systems. A methodology that helps in a crucial way the project of such systems becomes a fundamental piece. But, because of the physics laws that rule the phenomena involved, the analytic solution of complex project becomes unviable.

Because of that, the study of grounding systems searches in the numeric methods the steps towards the project of such systems. Given the actual state of the art of the Finite Elements Method, it shows a lot of capacity to solve problems from the very simple ones to the even most complex. So, the application of such method to our analysis deserves a special attention.

Nowadays, the public dominion (open source code) has gained strength in the academic world. Not just because they allow the exchange of information, but also for having an intrinsic politic of knowledge diffusion. Although there are commercial computer packages to the very simulation of grounding systems using the Finite Elements Method, an application based in open source tools is an interesting solution, not only by the economic interest, but also the academic one. Because it allows a contribution of many entities in the application, besides being possible to validate it and verify possible fails not detected before, even if it's not done by the original authors.

This work used open source computer packages to the analysis of problems based in the Finite Elements Method diffused in the academy society. Although the packages are independent from each other, here these packages will work together, in a conjunct and

organized way. In this way, we can obtain a system totally based in open source tools, capable of helping in the project of grounding systems.

To the validation of the model and the integrations used, it will be studied two cases with already known results. Since it's very difficult to obtain a real project of a grounding system, the validation will be done comparing the results with a case of relative easy analytic solution, largely known in the literature, and with a case a lot studied and simulated with computer packages high know and respected in the area, being it's results considered good parameters of comparison.

So, then, it's searched the creation and validation of a computer package of open source code. The use of other packages from other communities allows the improvement of the sofware just by upgrading the used packages, showing not just the strength of the Finite Elements Method to the analysis of complex problems of the Electric Engineering area, but also the strength of the use of open source programs, allowing the diffusion, exchange and extension of knowledge between the academic communities of the world.

Keywords: Finite Elements Method, computer simulation, grounding systems, open source software, Gmsh, ITL, LMAGLIB.

Sumário

1. Introdução

O problema de estudo de sistemas de aterramento é hoje de fundamental importância para o estudo de linhas de transmissão e subestações.

A necessidade de uma boa malha de aterramento provém dos problemas de compatibilidade magnética e proteção de sistemas, não só como meio de descarga de correntes espúrias, mas também para proteção contra o potencial de passo e tensão de toque.

Por isso, os valores determinantes se concentram num valor de segurança pessoal (o potencial de passo e o potencial de toque) e um valor de desempenho (a resistência da malha). A determinação de tais parâmetros é largamente estudada por métodos numéricos [2]. Isso porque o estudo envolvendo equações analíticas do problema se torna inviável, visto as muitas diversidades dos solos das instalações, do número de variáveis envolvidas e da complexidade das geometrias de determinadas malhas. Por isso, um método de cálculo numérico baseado numa aproximação precisa dos resultados analíticos e de forma abrangente se torna necessário. Esse método deve vislumbrar a malha de uma forma plena, sem desperdiçar a precisão dos resultados obtidos.

Existe um método difundido no meio de projetistas de malha que é baseado no método de imagens complexas [1]. Porém, é razoavelmente limitado nas questões de modelagem do solo e da geometria do sistema. Este trabalho visa o modelamento de sistemas de aterramento utilizando um método mais abrangente, capaz de resolver geometrias complexas em solos não uniformes.

O Método de Elementos Finitos [5] é um método de cálculo numérico baseado em discretização de um domínio e solução de sistemas lineares de equações. Em resumo, utiliza-se de algoritmos de discretização de geometrias dentro de um domínio em elementos finitamente pequenos. Em seguida, é feito o equacionamento de cada elemento e a integração ao domínio como forma de resolver o cálculo de grandezas básicas da Engenharia Elétrica. Por último, os dados são integrados para se obter as grandezas de interesse na geometria.

Existem diversas ferramentas no mercado que são capazes de modelar sistemas complexos de problemas eletromagnéticos pelo Método de Elementos Finitos. Um dos programas disponíveis no mercado, o Flux3D [4], é capaz de modelar diversos problemas eletromagnéticos, tanto estáticos quanto dinâmicos.

Entretanto há inconvenientes em se utilizar pacotes comerciais, em geral devido ao custo elevado, mas também devido falta de flexibilidade para quando se depara com problemas não convencionais, ou aplicações para as quais o programa não foi previsto, que exigem modelagens específicas [6].

Atualmente a comunidade de códigos computacionais de domínio livre [7] vem ganhando destaque, não apenas como alternativa aos altos custos praticados pelos proprietários de programas comerciais, mas também pela difusão do conhecimento. Diversas ferramentas de cálculo utilizando o Método de Elementos Finitos têm aparecido na comunidade de código livre. Essas ferramentas podem estar disponíveis seja na forma de código fonte, seja na forma de programas executáveis nas mais diversas plataformas (Windows, Linux, etc). O desafio então, primeiro, é selecionar aquelas mais adequadas ao problema estudado e, segundo, efetuar a integração dessas ferramentas de modo a tornar a sua utilização a mais amigável possível. Normalmente faz-se necessária a implementação de programas que efetuam o gerenciamento e a troca de dados entre as diversas ferramentas.

A meta do projeto é, portanto, a partir de uma formulação teórica, integrar diversas ferramentas baseadas no Método de Elementos Finitos de domínio livre a fim de se obter um aplicativo eficiente para o estudo de sistemas de aterramento. Os aplicativos e bibliotecas escolhidas são: Gmsh, LMAGLIB e ITL. Essas ferramentas serão descritas a seguir no relatório, e foram escolhidas por serem as mais compactas e de fácil aprendizagem.

Desta forma, o trabalho busca uma solução robusta para o problema, de forma a modelar com boa precisão os sistemas de aterramento, dado a capacidade do Método de Elementos Finitos em criar modelos numéricos de aplicações complexas. Com o uso do Método de Elementos Finitos o sistema torna-se bastante flexível para aplicações específicas e complicadas, permitindo uma ferramenta de análise complexa para problemas complicados e não convencionais de sistemas de aterramento.

2. Formulação Matemática

Neste capítulo veremos o processo de equacionamento que torna o Método de Elementos Finitos um método possível e determinado para a solução do problema. A demonstração não será levada a fundo, visto que sua implementação é feita pela biblioteca LMAGLIB, e também que seu aprofundamento foge dos domínios de um trabalho de Graduação. A formulação apresentada é de caráter informativo, e não normativo.

2.1. Equacionamento do Problema

Um sistema de aterramento, submetido a uma solicitação, injeta uma alta densidade de corrente no solo. A sua difusão é dada pela equação de Laplace:

$$\nabla \cdot \sigma \nabla V_0 = 0 \text{ em } \Delta \qquad (2.1)$$

na qual:

$\nabla \rightarrow$ operador gradiente

$\Delta \rightarrow$ domínio aberto estendido ao infinito

$V_0 \rightarrow$ tensão de solicitação (V)

$\sigma \rightarrow$ condutividade do solo (S/m)

Por métodos práticos, verifica-se que o domínio pode ser truncado em superfícies delimitadoras, sobre as quais são aplicadas condições iniciais que, por sua vez, são oriundas do ponto remoto do domínio [11]. Assim sendo, temos que o problema é resumido em encontrar uma distribuição de potencias elétricos que satisfaça as seguintes condições:

$$\begin{aligned} V_0 &= \overline{V} && \text{em } S_1 \\ \frac{\partial V_0}{\partial n} &= 0 && \text{em } S_2 \\ V_0 &= V_d && \text{no ponto de defeito} \end{aligned} \qquad (2.2)$$

nas quais:

$S = S_1 \cup S_2 \rightarrow$ superfície delimitadora de Δ

Vemos também que, por critérios práticos, se definirmos o truncamento do domínio em 6 vezes o tamanho máximo do sistema de aterramento, conseguimos resultados bem precisos com os esperados [11]. Existem técnicas que propõem transformações de coordenadas para problemas tridimensionais com domínio aberto (o que é o nosso caso), mediante as quais o infinito é descrito como um ponto ou uma linha [16]. Mas tais técnicas não serão vistas aqui, pela complexidade e tempo demandados. Uma continuação do projeto poderia estudar tal implementação para melhora no desempenho computacional e da descrição do problema.

Aplicando-se o Método dos Resíduos Ponderados [10], tem-se:

$$\int_{\Delta} W_0 \cdot \left(\nabla \cdot \sigma \nabla V_0\right) d\Delta + \int_{S_1} W_1 \cdot \left(V_0 - \overline{V}\right) dS + \int_{S_2} W_2 \cdot \frac{\partial V_0}{\partial n} dS = 0 \qquad (2.3)$$

Escolhendo-se adequadamente W_0, W_1 e W_2, o problema pode ser escrito (de acordo com as condições de contorno) [11]:

$$\int_{\Delta} W_0 \cdot \left(\nabla \cdot \sigma \nabla V_0\right) d\Delta + \int_{S_2} W_2 \cdot \frac{\partial V}{\partial n} dS = 0 \qquad (2.4)$$

Como no Método de Elementos Finitos o domínio é subdividido em NE subdomínios equivalentes, que são os elementos [10], podemos escrever a equação acima discretizada para cada elemento:

$$\sum_{e=1}^{NE} \left[\int_{\Delta^e} W_0 \cdot \left(\nabla \cdot \sigma \nabla V_0\right) d\Delta + \int_{S_2^e} W_2 \cdot \frac{\partial V_0}{\partial n} dS \right] = 0 \qquad (2.5)$$

Sendo que cada equação deve ser satisfeita em cada elemento, temos, após manipulação matemática:

$$-\int_{\Delta^e} \sigma \nabla W_0 \cdot \nabla V_d d\Delta + \oint_{S^e} \sigma W_0 \cdot \frac{\partial V}{\partial n} dS + \int_{S_2^e} W_2 \cdot \frac{\partial V}{\partial n} dS = 0 \qquad (2.6)$$

Face à arbitrariedade da escolha das funções de ponderação, e impondo ainda $W_2 = -\sigma W_0$, temos finalmente:

$$-\int_{\Delta^e} \sigma \nabla W_0 \cdot \nabla V_d d\Delta + \int_{S_1^e} \sigma W_0 \cdot \frac{\partial V}{\partial n} dS = 0 \qquad (2.7)$$

Demonstra-se que as condições estabelecidas sobre S_2 são automaticamente satisfeitas para este processo [11]. Vale salientar que, para o equacionamento, é importante garantir que a integral em (2.6) exista, o que depende da escolha de W_0 (deve-se garantir que W_0 tenha no mínimo continuidade em C^1).

2.2. Funções de Interpolação

No Método de Elementos Finitos, pode-se escrever o valor da função potencial elétrico de cada elemento como uma interpolação dos seus valores nos vértices do elemento. A escolha criteriosa da função de interpolação é importante para a boa precisão dos resultados. Porém um alto grau na ordem desta causa um maior custo de iteração (ou seja, maior custo computacional).

No caso de sistemas de aterramento, a função bilinear de interpolação mostra resultados favoráveis, tanto em malhas bidimensionais como em malhas tridimensionais [10]. Além disso, a facilidade de adaptá-la ao algoritmo de Delaunay [14] fez com que se adotasse tal função.

Em [11], temos as descrições detalhadas das equações para cada tipo de elemento. Aqui, veremos apenas de forma resumida como a função potencial elétrico será descrita para cada elemento.

Genericamente, podemos escrever, para qualquer elemento:

$$V = \sum_{i=1}^{n} N_i \cdot V_i \qquad (2.8)$$

na qual N_i e n dependem do tipo de elemento considerado.

Para elementos prismáticos, temos:

$$N_i\left(x_j, y_j, z_j\right) = \delta_{ij} \tag{2.9}$$

na qual δ_{ij} é o símbolo de Kronecker, ou seja:

$$\delta_{ij} = \begin{cases} 0 & \text{se } i \neq j \\ 1 & \text{se } i = j \end{cases} \tag{2.10}$$

A partir disso, podemos escrever:

$$\begin{aligned} N_1 &= \alpha_1 \cdot \beta_1 \\ N_2 &= \alpha_2 \cdot \beta_1 \\ N_3 &= \alpha_3 \cdot \beta_1 \\ N_4 &= \alpha_1 \cdot \beta_2 \\ N_5 &= \alpha_2 \cdot \beta_2 \\ N_6 &= \alpha_3 \cdot \beta_2 \end{aligned} \tag{2.11}$$

Os valores de α_i podem ser calculados por [10]:

$$\alpha_i = \frac{1}{2\Delta} \cdot \left(a_i + b_i x + c_i y\right) \qquad i = \{1,2,3\} \tag{2.12}$$

Os coeficientes de (2.12), chamados de coordenadas generalizadas, podem ser calculados por procedimento descrito em [10].

Para os coeficientes β_i, calculamos por [10]:

$$\beta_1 = \frac{Z_2 - z}{Z_1 - Z_2} \quad \text{e} \quad \beta_2 = \frac{z - Z_2}{Z_1 - Z_2} \tag{2.13}$$

Por fim, podemos escrever:

$$V = \sum_{i=1}^{6} N_i \cdot V_i \qquad (2.14)$$

Note que a função N_i é bilinear [10], o que fornece um resultado exato para a função potencial elétrico dentro do elemento.

Para os elementos bidimensionais (ou seja, os elementos de placa), temos:

$$N_i = \alpha_i \qquad i = \{1,2,3\} \qquad (2.15)$$

na qual os coeficientes α_i são calculados da mesma forma que mostrada anteriormente.

Logo, podemos escrever:

$$V = \sum_{i=1}^{3} N_i \cdot V_i \qquad (2.16)$$

Finalmente, para elementos unidimensionais (ou seja, os elementos de condutores e barras), temos:

$$N_i = \beta_i \qquad i = \{1,2\} \qquad (2.17)$$

na qual os coeficientes β_i são calculados da mesma forma que mostrada anteriormente.

Logo, podemos escrever:

$$V = \sum_{i=1}^{2} N_i \cdot V_i \qquad (2.18)$$

Assim, temos como calcular os potenciais elétricos em qualquer posição de qualquer elemento, tendo os potenciais nos seus vértices.

2.3. Geração da Matriz de Potenciais Elétricos

Para a geração da matriz de potenciais elétricos, que fornecerá os valores necessários à obtenção das grandezas de interesse, temos que ver a escolha das funções de ponderação vistas anteriormente. Tal escolha leva-nos a métodos numéricos de resolução de problemas de campo.

Em casos de resolução de problemas de equações diferenciais expressas por operadores auto-adjuntos (como o fenômeno regido pela equação de Laplace), a técnica de Galerkin aplicada ao Método de Elementos Finitos [10] nos leva à obtenção de sistemas de equações algébricas com matrizes simétricas. Isso facilita e melhora o desempenho dos métodos de resolução numérica, bem com a implementação de tal processo de solução. No nosso caso, estaremos utilizando as implementações de resolução numérica de sistemas lineares de coeficientes constantes no tempo formando matrizes simétricas da ITL [9].

Na técnica de Galerkin, a função de ponderação W_0 de (2.7) é escolhida como uma variação arbitrária de V (a variável de estado do problema) [11]. De forma que:

$$W_0 = \delta V = \sum_{j=1}^{n} N_j \cdot \delta V_j \tag{2.19}$$

Assim, verifica-se que, para cada elemento, W_0 e V utilizam-se da mesma função genérica de interpolação. No caso de operadores auto-adjuntos, isso permite a obtenção de matrizes simétricas.

Para S_1 em cada elemento (S_1^e), conforme visto no item 2.1, a condição em S_1 é imposta como V nulo. Disso resulta uma variação sempre nula de V em S_1.

Logo, temos:

$$\sum_{j=1}^{n} \left\{ \sum_{i=1}^{n} \left(\int_{\Delta^e} \sigma \nabla N_j \nabla N_i V_i \, d\Delta \right) \right\} \cdot \delta V_j = 0 \tag{2.20}$$

Como já descrito, sendo δV uma variação arbitrária de V, temos que (2.20) só é satisfeita se:

$$\sum_{i=1}^{n}\left(\int_{\Delta^e}\sigma\nabla N_j\nabla N_i V_i\, d\Delta\right)=0 \qquad j=\{1,2,...,n\} \tag{2.21}$$

Essa representação, para cada elemento do domínio discretizado, é a representação de um sistema linear de equações, do tipo:

$$\left[G^e\right]\cdot\left[V^e\right]=0 \tag{2.22}$$

na qual:

$$G_{ij}^e=\int_{\Delta^e}\nabla Nj\nabla Ni\, d\Delta \qquad (i,j)=\{1,2,...,n\}$$

$$\left[V^e\right]=\left[V_1 \quad V_2 \quad ... \quad V_n\right]^T$$

Segundo processo consagrado [17], podemos expandir esse sistema de equações de um subdomínio (elemento) para todo o domínio. Inserida as condições de contorno e as condições iniciais do problema de sistema de aterramento, obtemos um sistema do tipo:

$$[G]\cdot[V]=[I] \tag{2.23}$$

no qual $[G]$ é a matriz global do problema (e é simétrica, como visto anteriormente), de ordem NxN (sendo N o número de nós do domínio discretizado), e $[I]$ é a matriz de condições iniciais (e de contorno) do problema, descrita em V.

A resolução desse sistema de equações fornece o potencial elétrico em todos os nós do domínio discretizado. Essa informação é crucial para o cálculo das grandezas de interesse (potencial de passo, potencial de toque e resistência de aterramento), o que permitirá a avaliação do sistema de aterramento.

2.4. Cálculo do Potencial de Passo e da Resistência de Aterramento

Utilizando os potenciais elétricos calculados no item anterior, podemos obter as variáveis de interesse para avaliação do sistema de aterramento. Essas variáveis, no caso, são o potencial de passo e a resistência de aterramento. O potencial de toque é calculado pela norma IEEE Std 80 [15] e, portanto, não será apresentado aqui.

No caso do cálculo do potencial de passo, temos que [13]:

$$\vec{E} = -\nabla V \tag{2.24}$$

Sendo os potenciais elétricos conhecidos para cada elemento do domínio discretizado, podemos escrever, para cada elemento, as seguintes equações [11]:

$$\begin{aligned} E_x &= -\frac{\partial V}{\partial x} = -\sum_{i=1}^{n} \frac{\partial N_i}{\partial x} \cdot V_i \\ E_y &= -\frac{\partial V}{\partial y} = -\sum_{i=1}^{n} \frac{\partial N_i}{\partial y} \cdot V_i \\ E_z &= -\frac{\partial V}{\partial z} = -\sum_{i=1}^{n} \frac{\partial N_i}{\partial z} \cdot V_i \end{aligned} \tag{2.25}$$

Como só nos interessa calcular o potencial de passo no plano do solo (ou em algum plano paralelo ao mesmo, que é o plano XY), podemos desconsiderar a coordenada em Z, e calculamos apenas as componentes E_x e E_y em $z=Z_s$, sendo Z_s a cota do solo. Disso resulta que:

$$V_{passo} = \sqrt{E_x^{\,2}(Z_s) + E_y^{\,2}(Z_s)} \tag{2.26}$$

Dado que as faces dos elementos dos problemas são discretizadas pela triangularização de Delaunay, e em vista das funções de interpolação para os elementos bidimensionais, verifica-se que o potencial de passo é constante no interior dos elementos de placa na cota do solo (apenas) [11].

Para o cálculo da resistência de aterramento, vamos considerar primeira a potência dissipada no sistema de aterramento. Temos que [13]:

$$P_d = \sigma E^2 \quad \left[W / m^3\right] \tag{2.27}$$

A equação (2.27) fornece a densidade de potência dissipada por efeito Joule, e sua integração no volume fornece a potência dissipada pelo mesmo. Por isso, para cada elemento do domínio discretizado, podemos escrever [11]:

$$P_d = \int_{V^e} P_d \, dV \tag{2.28}$$

Retomando as equações (2.25), podemos escrever o valor do potencial total como:

$$E^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{\partial N_i}{\partial x}\cdot\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\cdot\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\cdot\frac{\partial N_j}{\partial z}\right)\cdot V_i \cdot V_j \tag{2.29}$$

A partir disso, podemos escrever, na forma matricial, a potencial total dissipada por cada elemento como:

$$P_d^e = \left[V^e\right]^T \cdot \left[G^e\right] \cdot \left[V^e\right] \tag{2.30}$$

na qual as matrizes $\left[G^e\right]$ e $\left[V^e\right]$ são conhecidas para cada elemento, conforme visto no item anterior.

Portanto, a potência total dissipada por efeito Joule do domínio do estudo (discretizado) por ser expressa por:

$$P_d = \sum_{e=1}^{NE} P_d^e \tag{2.31}$$

na qual NE é o número de elementos do domínio discretizado.

Por fim, podemos relacionar a potência dissipada e a resistência de aterramento como [13]:

$$R_{aterramento} = \frac{V_d^2}{P_d} \tag{2.32}$$

na qual V_d é a elevação de potencial no ponto de defeito. Esse potencial é uma condição inicial do problema, forme visto na equação (2.2), e por isso pode ser adotado como valor conhecido.

3. Parametrização de Geometrias Básicas

O uso de geometrias de base para a modelagem do problema é um recurso importante, pois oferece o recurso de se gerar sistemas de aterramento básicos, e então adaptar a geometria ao problema estudado. Aqui, estaremos vendo três geometrias bases de um sistema de aterramento: a malha reticulada, a malha por barras aterradas e a placa aterrada. Cada uma possui suas vantagens e desvantagens, mas essa discussão não faz parte do escopo deste trabalho.

Para a geração das geometrias deve-se tomar o cuidado de se manter as unidades consistentes. A unidade básica deve sempre ser seguida, para todas as outras medidas do projeto. Recomenda-se o uso das unidades do Sistema Internacional.

Embora não seja validado no programa, vale ressaltar que os parâmetros informados para as geometrias abaixo devem ser positivos e não nulos. Dependendo ainda do parâmetro, deve ser um número inteiro apenas (como número de barras, número de lados, etc.).

3.1. Malha Reticulada

A primeira parametrização a ser estudada é a malha reticulada, ou seja, formada por diversos "cubos". É o sistema mais utilizado, em geral em apenas um plano. A idéia da parametrização é deixar o processo o mais maleável possível, então tomemos a malha a mais genérica possível. Um exemplo de malha de aterramento está na Figura 3.1.

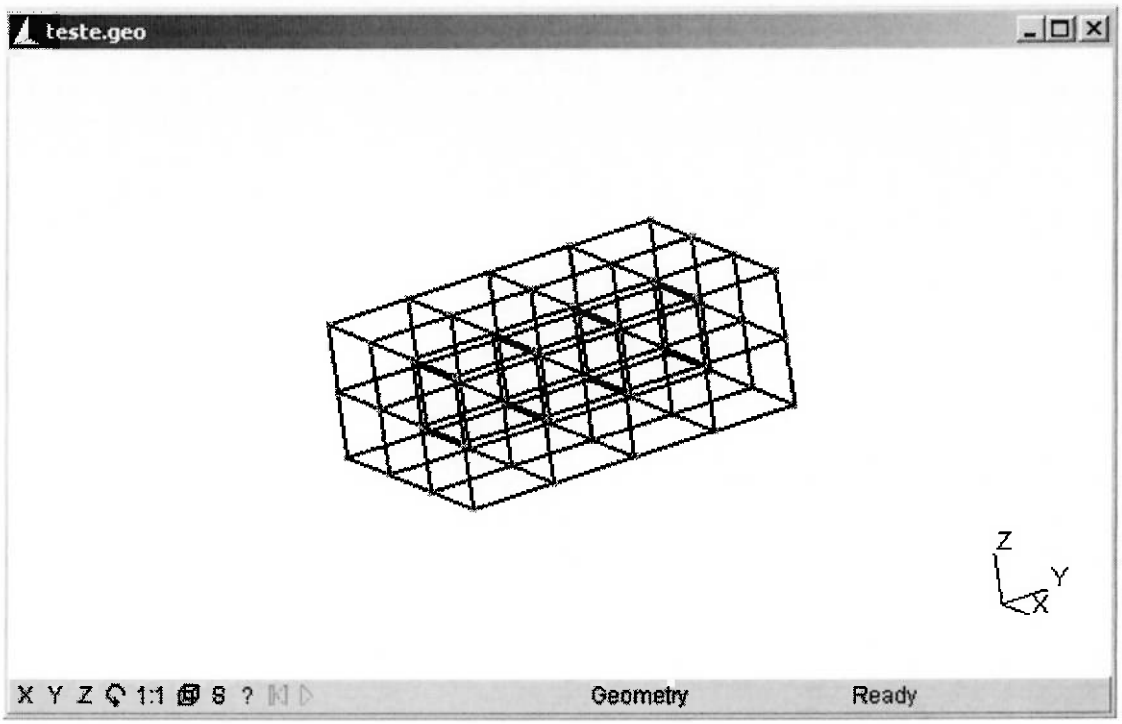


Figura 3.1 – Sistema de aterramento modelado por malha reticulada.

Para gerar tal sistema, os parâmetros necessários serão os valores das variáveis:

n_x → número de blocos no eixo X
n_y → número de blocos no eixo Y
n_z → número de blocos no eixo Z
l_x → tamanho de cada bloco em relação ao eixo X
l_y → tamanho de cada bloco em relação ao eixo Y
l_z → tamanho de cada bloco em relação ao eixo Z

Dados estes valores, podemos dizer primeiramente que a malha terá dimensões de ($n_x.l_x$, $n_y.l_y$, $n_z.l_z$). Para escrever os pontos em cada dimensão, basta tomarmos os valores de 0 até $n_i.l_i$.

Para facilitar o entendimento, e seguir o padrão do Gmsh, os pontos e linhas serão identificados por um código. Este código será um número natural não nulo, e o código do ponto será independente do código da linha.

Os pontos são descritos aqui pelas suas coordenadas (x, y e z), embora no Gmsh exista o parâmetro de refino da malha no ponto. Como esse aspecto não interessa aqui, na geração de geometria, não estaremos abordando o problema

nesta parte. As linhas são descritas pelo ponto de início e de término do segmento de reta que a representa.

Os pontos, para facilitar a automação das outras tarefas, serão criados na seguinte seqüência:

Ponto(1) = {0, 0, 0}

Ponto (2) = {l_x, 0, 0}

Ponto (3) = {$2 \cdot l_x$, 0, 0}

...

Ponto (m) = {$n_x \cdot l_x$, 0, 0}

Ponto (m+1) = {0, l_y, 0}

Ponto (m+2) = {l_x, l_y, 0}

Ponto (m+3) = {$2 \cdot l_x$, l_y, 0}

...

Ponto (n) = {$n_x \cdot l_x$, $n_y \cdot l_y$, 0}

...

Ponto (k) = {$n_x \cdot l_x$, $n_y \cdot l_y$, $n_z \cdot l_z$ }

Depois de gerado os pontos, temos que modelar as barras (que são as linhas). Dado a formação dos pontos, isto se torna uma tarefa razoavelmente fácil.

O processo de desenho é feito para cada camada no eixo Z da geometria. Na primeira etapa do processo, são desenhadas as barras paralelas ao eixo X. Pela formação dos pontos, notamos que são as linhas formadas pelos pontos de códigos n e $n+1$, respectivamente, para o ponto inicial e final. Na segunda etapa do processo, são desenhadas as barras paralelas ao eixo Y. Analogamente, notamos que são as linhas formadas pelos pontos de código n e $n+n_x+1$, respectivamente. Na última etapa, se atingida a última camada em relação ao eixo Z, então são desenhadas as barras no eixo Z. Pela formação dos pontos, são os pontos de código de n e $n+\left((n_x+1)\cdot(n_y+1)\right)$, respectivamente.

O algoritmo representativo de geração de pontos e linhas desta geometria está no apêndice A. No exemplo da Figura 3.1, temos uma malha gerada com o programa para os seguintes valores: n_x=2; n_y=3; n_z=4; l_x=1,2; l_y=1,4; l_z=1,6.

Em geral, os casos reais são representados por uma malha com n_z=0, que representa uma malha reticulada apenas no plano do solo (ou próximo dele). Por isso, o algoritmo do apêndice A já considera este caso como viável, e consegue geral tal modelo.

3.2. Barras Aterradas

A próxima parametrização é a de barras aterradas. Essa geometria consiste em barras aterradas numa certa geometria (vista de cima), e curto-circuitadas na superfície (ou próximo da superfície). Neste caso, serão consideradas, para parametrização, todas as barras como sendo iguais entre si, e a geometria da vista superior como sendo um polígono regular (todos os lados iguais).

Para gerar tal sistema, os parâmetros necessários serão os valores das variáveis:

n → número de lados do polígono de vista superior
l → tamanho do lado do polígono de vista superior
b → comprimento da barra a ser aterrada

Gerar essa geometria é mais simples que a primeira. Se n < 3, então devemos desenhar uma (n = 1) ou duas barras conectadas (n = 2), de comprimento b, separadas por l e conectadas (se n = 2). O algoritmo representativo de geração do sistema para este caso está no apêndice B.

Para o caso de n ≥ 3, basta verificar que, dado que a soma dos ângulos externos de um polígono qualquer é 360°, e sabendo-se que, dado um polígono regular, todos os seus ângulos externos são iguais, podemos calculá-los (denominaremos estes genericamente de α) como:

$$\alpha = \frac{360^\circ}{n} \tag{3.1}$$

A partir do ângulo externo, e adotando um ponto de origem (vamos adotar como o ponto {0,0}), temos que cada ponto do polígono é um incremento de

$l \cdot \cos(\alpha)$ no eixo X e um incremento de $l \cdot \sin(\alpha)$ no eixo Y. Dessa forma, até fechar o polígono, incrementamos n vezes as coordenadas e obtemos todos os pontos do polígono. Fazendo isso nos planos de z = 0 e z = b, obtemos os pontos para a geração das barras do sistema de aterramento. Conectando sempre os pontos gerados nos dois planos, já desenhamos as barras aterradas.

Para desenhar a conexão das barras, basta verificar que o polígono se conecta sempre do ponto i para o ponto i+2 (pois o ponto i+1 está no outro plano, paralelo em relação ao eixo Z ao ponto i). Conectando todos os pontos i a i + 2, sendo i de 1 até o número de pontos gerados menos 2, conectaremos todo o polígono em z = 0 (não nos interessa conectar o polígono para z = b). Deve-se notar que, para o caso de i ser o número de pontos gerados menos 2 (o limite superior das iterações, portanto), a ligação deve ser feita até o ponto 1, pois aí fechamos o polígono.

O algoritmo representativo de geração do sistema para este caso está no apêndice C. Um exemplo de barras aterradas gerada com o programa está na Figura 3.2, com os seguintes parâmetros: n=5; l=8; b=10.

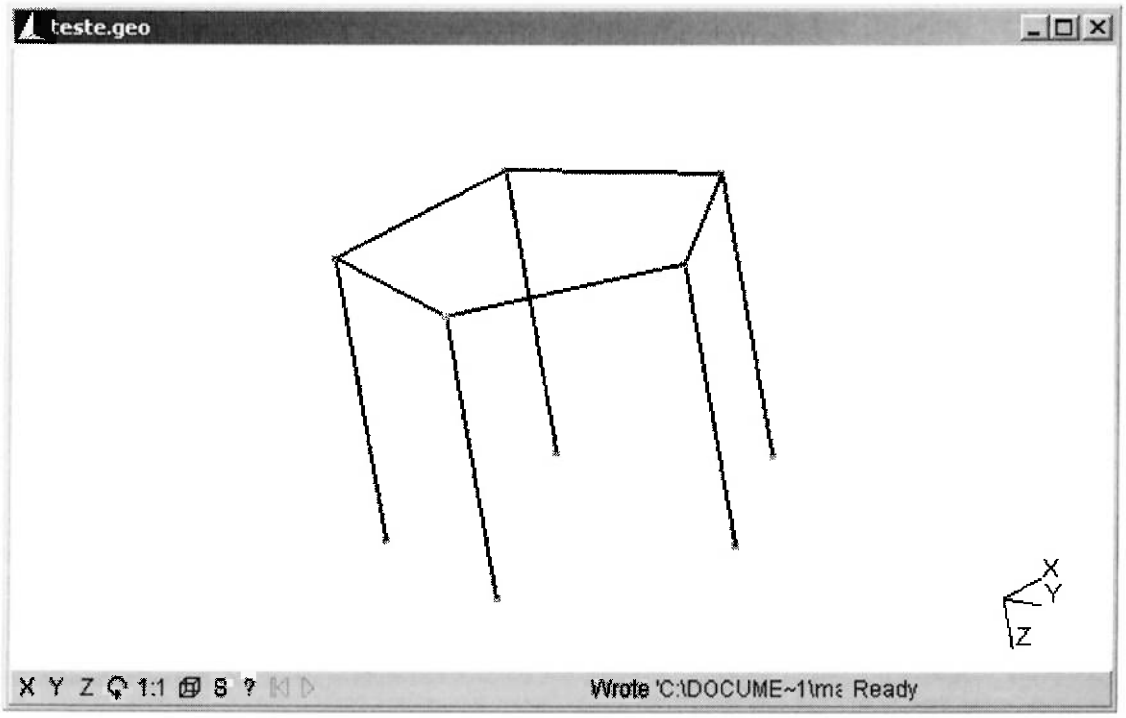


Figura 3.2 – Sistema de aterramento modelado por barras aterradas.

3.3. Placa Aterrada

A última parametrização é a de uma placa aterrada. Provavelmente, em questão de algoritmo, esta é a mais simples de todas. A única diferença é que aqui tratamos de uma geometria de superfície.

Para gerar este sistema, os parâmetros necessários serão os valores das variáveis:

l → largura da placa aterrada
b → comprimento da placa aterrada
p → profundidade em relação ao solo que a placa está aterrada

Basta então descrevermos os pontos, linhas e superfícies de uma forma coordenada e manual. Devemos só tomar verificar se a profundidade de aterramento é zero (placa próxima da superfície) ou maior que zero.

A placa pode ser descrita pelo seguinte conjunto:

Ponto (1) = {0, 0, p}
Ponto (2) = {l, 0, p}
Ponto (3) = {l, b, p}
Ponto (4) = {0, b, p}
Linha (1) = {1, 2}
Linha (2) = {2, 3}
Linha (3) = {3, 4}
Linha (4) = {4, 1}

A superfície então será formada pelo contorno das linhas 1, 2, 3 e 4, nesta ordem de orientação. A orientação é importante para os processos de discretização do domínio.

Caso p seja maior do que zero, devemos considerar uma barra condutora indo do solo até a placa, para efeito de simulação. Para o problema, vamos modelar esta barra entrando na placa pelo ponto 1 descrito acima.

Portanto, além da placa, temos a seguinte barra:

Ponto (5) = {0, 0, 0}

Linha (5) = {1, 5}

Assim sendo, temos o algoritmo que gera o sistema de aterramento de placa aterrado (embora o algoritmo seja apenas um conjunto de linhas, pontos e superfícies parametrizados).

Um exemplo de placa aterrada gerada com o programa está na Figura 3.3, com os seguintes parâmetros: l=10; c=15; p=5.

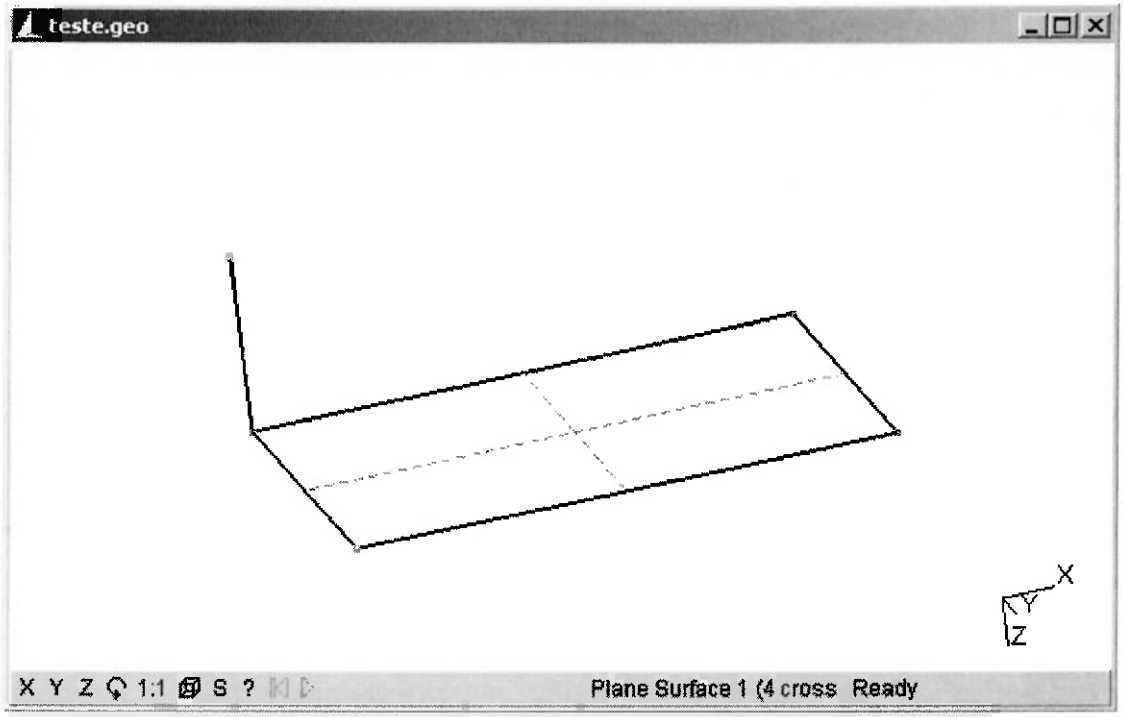


Figura 3.3 – Sistema de aterramento modelado por placa aterrada.

4. Estudo de Ferramentas Utilizadas

Nesta parte serão vistas as ferramentas utilizadas para a implementação matemática do problema de sistemas de aterramento. Todas as ferramentas utilizadas são de código livre[7] e foram estudadas do código-fonte. Com exceção do Gmsh, todas as outras ferramentas foram implementadas dentro do código-fonte do programa, já que são bibliotecas de suporte para desenvolvimento de aplicações.

4.1. Gmsh

O Gmsh é um programa capaz de gerar e visualizar geometrias e malhas de elementos finitos [8]. Ele será usado para a construção de geometrias e geração de malhas de elementos finitos. Embora seja um programa de código livre, neste trabalho será utilizada sua versão executável, e não dentro do programa desenvolvido, por questões de complexidade e de amplitude do estudo.

O Gmsh é escrito em STL C++ [18], e utiliza como classe gráfica o FLTK [21]. Isso permite que a interface gráfica seja totalmente independente da plataforma, portanto o programa pode ter versões tanto para Windows quanto para Linux (embora atualmente o programa esteja com problemas na sua versão para Windows, o que será visto em breve, conforme comentário do autor). Além disso, é possível a parametrização do visualizador e melhor edição de elementos gráficos. Logo, torna-se uma poderosa ferramenta de manipulação gráfica para o Método de Elementos Finitos.

4.2. LMAGLIB

A biblioteca de funções LMAGLIB é uma biblioteca desenvolvida pelo Laboratório de Eletromagnetismo Aplicado do Departamento de Energia e Automação Elétricas da Escola Politécnica da USP (maiores detalhes podem ser encontrados em http://www.lmag.pea.usp.br). É uma biblioteca de suporte escrita em STL C++ [18], que permite a simulação de problemas de Engenharia Elétrica por Método de Elementos Finitos. Ele permite estabelecer as condições de contorno e

condições iniciais, as propriedades físicas dos modelos geométricos (úteis para os cálculos dos coeficientes das funções de interpolação dos elementos do domínio discretizado) e as montagens das matrizes lineares de problemas da área de Engenharia Elétrica.

Neste trabalho, esta biblioteca será utilizada no módulo chamado de equacionamento, ou seja, a parte que compreende a montagem do sistema linear a partir das propriedades físicas e do domínio truncado e discretizado. O processo matemático descrito anteriormente é implementado pela LMAGLIB, embora a solução do sistema linear seja feita pela ITL.

4.3. ITL

A ITL (Iterative Template Library) é uma biblioteca genérica escrita que provê métodos iterativos de solução de sistemas lineares [9]. É uma biblioteca escrita em STL C++ [18] de alto desempenho numérico. Seu formato de dados é conhecido como interface abstrata de matriz-matriz ou matriz-vetor. Isso permite uma melhor utilização de matrizes esparsas e simétricas, o que é bastante comum no Método de Elementos Finitos. Esse formato de entrada e saída de dados é um padrão vastamente conhecido no mundo acadêmico, por isso a biblioteca ITL é uma das principais para solução de sistemas lineares.

Neste trabalho, a ITL será implementada no módulo chamado de resolução, que irá resolver o sistema linear montado a partir das condições físicas e domínio discretizado. Embora um método numérico conhecido pudesse ser implantado, o uso de uma outra biblioteca foi feito porque a parte da solução do sistema linear (para obtenção dos potenciais elétricos) é a parte que consome o maior tempo de processamento, e o uso de uma biblioteca especializada reduz o tempo de implementação e de processamento do problema.

5. Estrutura da Implementação Computacional

Após definidas as ferramentas e os métodos de abordagem, o próximo passo foi a escolha da linguagem a ser usada na implementação computacional do problema. Isto foi razoavelmente fácil, já que, por diversos aspectos, a linguagem STL C++ [18] se mostrou a mais eficaz. Além de todas as bibliotecas de suporte ser escritas primordialmente para esta linguagem, ela é muito difundida por ser totalmente Orientada a Objeto, confiável, de alto desempenho e, desde que especificada todas as suas convenções, pode ser compilada em qualquer ambiente operacional (com o compilador adequado).

O fato de a linguagem ser Orientada a Objeto pode, a princípio, dificultar um pouco a sua implementação. Porém esta técnica permite a escrita de programas mais intuitivos, de fácil compreensão, diagramação e projeto. Utilizando tal técnica, podemos descrever o programa como um diagrama de blocos, e programar cada bloco separadamente, juntando-os numa interface única. Outra vantagem é a independência dos blocos e o reaproveitamento de blocos e código já feito anteriormente, com total clareza e transparência.

Cada bloco, no caso, é chamado de objeto [18]. No caso da programação, não se programa um objeto, e sim uma classe de objeto. A partir de uma classe de objetos, podemos definir, ao longo do programa de interface (que seria a interligação dos programas, ou o diagrama principal), vários objetos da mesma classe. Em outras palavras, em um diagrama de blocos, podemos desenhar vários blocos de um mesmo tipo, cada um com suas definições próprias, mas todos oriundos de certo tipo de bloco (um somador, um integrador, etc.). No diagrama principal, nos preocupamos em utilizar o bloco, e não no seu funcionamento interno. Isso causa enorme transparência e facilidade de entendimento do programa.

A Figura 5.1 mostra o diagrama de blocos principal do programa. Nele podemos notar os passos de solução de qualquer problema de Método de Elementos Finitos, embora esta aplicação tenha um dado objetivo. A partir de agora, iremos nos referir aos blocos como objetos, para estarmos em acordo com a teoria da Programação Orientada a Objetos [18].

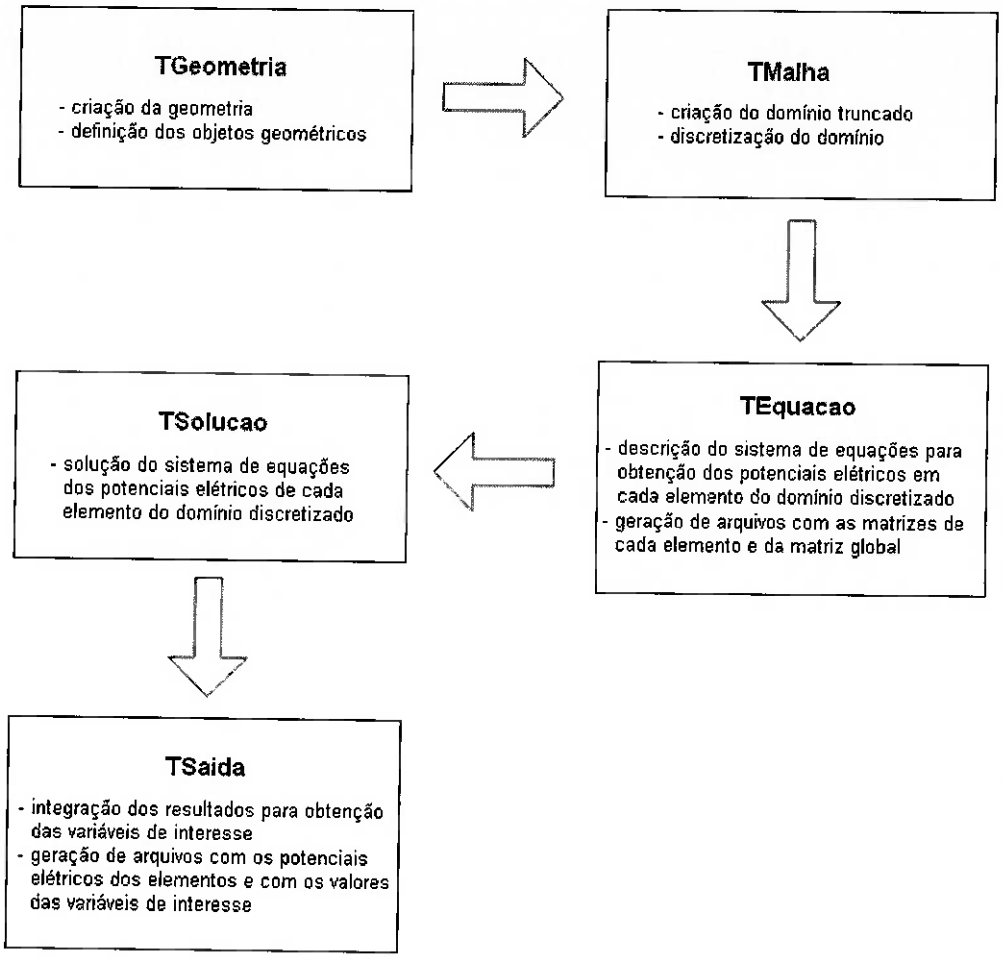


Figura 5.1 – Diagrama principal de objetos da implementação computacional.

Pode-se observar a linearidade do problema. Como cada entrada de um objeto depende da saída do anterior, não podemos aqui criar um paralelismo puro pelos objetos deste diagrama. É importante frisar que na implementação são definidos os operadores de interligação dos objetos (eles não são nativos da linguagem, como os operados de interligação de objetos já nativos).

Os dados de entrada não se referem necessariamente a uma classe de objetos. É responsabilidade da implementação principal de coordenar em quais classes tais dados são inseridos. No caso específico do Método de Elementos Finitos, esses dados entram na parte de modelagem geométrica do problema, geração do sistema linear de equações e de integração das grandezas básicas para obtenção das variáveis de interesse.

A seguir, iremos ver como cada classe de objetos foi feita. A implementação computacional procurou ao máximo ser prática de se entender, adotando normas e recomendações largamente utilizadas em programas acadêmicos, buscando

também uma normalização entre os trabalhos publicados [19]. Por isso, a primeira coisa a ser notada é a definição dos nomes das classes com um prefixo escolhido, e tal prefixo deve ser uniforme em todo o projeto. Escolhemos como "T" o prefixo, por ser uma letra muito utilizada em projetos de STL C++ (o prefixo vem da palavra "type", que significa definir o tipo do objeto criado).

5.1. Classes de Apoio

As classes de apoio são classes que criam objetos simples. Em geral tais objetos são utilizados como vetores ou matrizes em outros objetos. Para facilitar a modelagem das classes principais, as classes de apoio, mesmo que utilizados por apenas uma classe principal, devem ser definidos no escopo global do projeto, para permitir que sejam vistas e que possam ser utilizadas numa etapa posterior, se necessário.

Para facilitar o modelo, as seguintes classes de apoio foram utilizadas:

- **TPonto:** Os pontos são definidas pelas suas coordenadas cartesianas X, Y e Z. Além disso, para efeito de refinamento, os pontos utilizam um quarto parâmetro, que define a menor divisão de discretização do domínio para elementos que possuam este ponto como nó gerador [14].
- **TLinha:** As linhas são definidas pelo ponto de início e pelo ponto de término, sendo pontos objetos da classe TPonto. É importante notar o sentido das linhas para o algoritmo de discretização do Gmsh.
- **TContornoLinhas:** Contornos orientados, definidos por linhas que os formam, para orientação da malha de elementos finitos e para a geração de superfícies. O sentido é definido pelo sentido das linhas, e deve se garantir um contorno fechado para a definição de superfícies.
- **TSuperficie:** As superfícies são definidas exclusivamente pelo seu contorno delimitador, do qual são parte as linhas que formam a superfície. Nem sempre um contorno forma uma superfície, mas qualquer superfície é definida por um contorno (não se define, nesta implementação, uma superfície pelas linhas, numa relação totalmente direta).

- **TContornoSuperfícies:** Contornos orientados também, definidos por superfícies que os formam, para orientação da malha de elementos finitos e para a geração de volumes. O sentido é definido pelo sentido dos contornos de linhas das superfícies, e não necessariamente se garante o fechamento de um volume pelas linhas. Para se definir um volume, no entanto, o contorno de superfícies deve ser fechado (pode ser não orientado, desde que não mude diversas vezes o sentido e piore a geração da malha de elementos finitos).

- **TVolume:** Os volumes são definidos exclusivamente pelo seu contorno delimitador, do qual são parte as superfícies que formam o volume. Nem sempre um contorno forma um volume, mas qualquer volume é definido por um contorno (não se define, nesta implementação, um volume pelas superfícies, numa relação totalmente direta).

- **TNo:** Um nó da discretização do domínio é definido exclusivamente pelas suas coordenadas, em relação ao mesmo eixo da geometria. No caso, os eixos são cartesianos, pois não utilizamos outro sistema de coordenada (não há transformação de coordenadas).

- **TElemento:** Um elemento da discretização do domínio é definido pelos nós que formam o seu vértice. Aqui, não se aplica mais a definição de sentido, pois os elementos são não orientados (embora a ordem de formação dos pontos seja importante para a definição da figura geométrica correspondente ao elemento).

- **TMatrix [9]:** É uma classe que não foi definida nesta implementação, e sim pela ITL. Não existe uma definição nativa para matrizes em STL C++, por isso não se pode fazer operações diretamente com matrizes. Porém, utilizando esta classe de apoio da própria ITL, podemos definir matrizes e realizar operações algébricas com elas diretamente (ou seja, sem nos preocupar em como a operação é feita, apenas nos interessando o resultado). Existem diversas classes de apoio de definição de matrizes na ITL, mas, para efeito dos cálculos internos à implementação, estaremos utilizando a classe que nos traz a definição clássica de matriz, na qual ela é descrita pelos seus elementos nas linhas i e nas colunas j.

5.2. Classe TGeometria

Esta classe cria objetos responsáveis pelo gerenciamento da geometria que modela o sistema de aterramento. Várias classes de apoio (descritas anteriormente) são utilizadas por esta. Aqui, as classes de apoio utilizadas são as de objetos de pontos, linhas, contornos de linhas, superfícies, contornos de superfícies e volumes. Por isso, a classe *TGeometria* é um vetor ordenado de objetos dessas classes de apoio. Analogamente ao modelo real, uma geometria é formada por pontos, linhas, superfícies e volumes. Superfícies são definidas por contornos de linhas, e volumes por contornos de superfícies. Por isso, esta classe é uma imitação da realidade, para ser fiel ao modelo matemático proposto.

Além de ser responsável pela coleção (assim denominaremos) de objetos de modelos geométricos do problema, esta classe possui funções próprias que podem gerar tais objetos. Estas funções são as implementações dos algoritmos descritos no capítulo 3, que servem para gerar geometrias básicas de sistemas de aterramento. Ou seja, a partir de determinados parâmetros, o próprio objeto controlador da geometria pode gerar os pontos, linhas, etc., baseando-se nos algoritmos dos apêndices A, B e C.

Esta classe utiliza o mesmo modelo geométrico do Gmsh. Isso é proposital, para facilitar o intercâmbio dos dados do modelo utilizado neste trabalho com o modelo utilizado pelo Gmsh. Assim sendo, este objeto também possui funções de intercâmbio de dados, entre seu formato e aquele utilizado pelo Gmsh. Embora as definições de geração de malha de elementos finitos sejam definidas já na geometria pelo Gmsh, na implementação essas definições não são de responsabilidade desta classe. As funções de intercâmbio, no entanto, passam essas informações ao Gmsh, mas estas informações são oriundas da classe TMalha (vista a seguir). Logo, a classe TMalha é uma derivação parcial da classe TGeometria [18]. A técnica de uso de classe derivada é largamente utilizada em STL C++, pois permite o controle dos objetos de uma classe por outra (como se a outra fosse virtual), e por isso deve ser empregada sempre que houver dependência fixa entre duas classes [19].

5.3. Classe TMalha

Este classe, conforme mencionado anteriormente, é uma classe derivada da classe TGeometria. Em um problema de Método de Elementos Finitos, a geometria está diretamente ligada à discretização do domínio. Por isso, esta classe é totalmente dependente da anterior. Por isso, um objeto da classe TMalha só pode ser criado se for dada uma geometria (um objeto da classe TGeometria) de referência.

Uma vez obtida a geometria, esta classe irá criar o truncamento do domínio de estudo. Conforme mencionado anteriormente, um truncamento com uma distância seis vezes maior que a maior dimensão da geometria de estudo fornece bons resultados para o método. Não serão vistos métodos para melhorar tal truncamento, como a redução do infinito a um ponto [16].

Definido o truncamento, a classe deverá criar a orientação para geração da malha. Embora existam métodos que permitam o correto refinamento da malha, e tal refinamento deve ser feito cautelosamente para medir os esforços entre boa precisão e tempo de processamento, tais métodos não serão vistos aqui. Seria possível também permitir ao usuário escolher tal refinamento. Mas este trabalho, buscando a automação da solução, tomou como caminho a escolha automática. Para se verificar bons resultados, a malha é refinada para 0,1 da menor dimensão do sistema de aterramento. O refinamento, no caso do gerador da malha (Gmsh) é dado como parâmetro de seus pontos. Conforme mencionado anteriormente, embora tal determinação seja responsabilidade deste objeto, essa propriedade é passada e atualiza no objeto da classe TGeometria de referência, por questões práticas. Assim sendo, tal propriedade é passada aos pontos da geometria.

Por fim, a classe possui funções que solicitam ao Gmsh que gere a malha. Essa solicitação é feita em aberto, ou seja, para um outro aplicativo (no caso, o Gmsh), e não como passagem de variáveis pela memória. A classe então aguarda a geração da malha, e carrega os resultados, quando prontos, para a memória.

Tais resultados são guardados em um vetor ordenado de objetos das classes de nós e de elementos discretos do domínio. É importante ter tais objetos guardados em memória, não só para o equacionamento da malha, mas também para a análise posterior de resultados. O potencial elétrico (a grandeza básica que é obtida no primeiro processo de simulação) é feito para cada elemento, e deve ser integrado

em determinadas formas (conforme visto no capítulo 2) para obtenção dos valores de interesse.

5.4. Classe TEquacao

Esta classe é uma implementação da LMAGLIB. Dado um domínio discretizado, e conhecido determinadas propriedades físicas suas (como condutividade elétrica, tensão de defeito, etc.), esta classe irá aplicar a metodologia do capítulo 2 e obter as matrizes da equação (2.23) do caso. Vale ressaltar que o capítulo 2 não mostra a abordagem completa que foi implementada na LMAGLIB, pois não é o objetivo deste trabalho.

É válido notar que um objeto desta classe só pode ser construído com um objeto da classe TMalha como referência. Uma vez construído tal sistema, esta classe irá guardar as matrizes da equação (2.23) do problema, para eventual uso posterior, e também para validação no modelo. Além disso, a classe armazena as matrizes de cada elemento da equação (2.22), para a obtenção das variáveis de interesse para análise de sistemas de aterramento.

Caso seja desejado conhecer as matrizes do problema, sejam as da equação (2.23) ou de cada elemento da equação (2.22), a classe possui funções implementadas que permitem a saída de tais matrizes. Estas são guardadas nesta classe por objetos da classe de apoio TMatrix, que provém da implementação da ITL utilizada.

5.5. Classe TSolucao

Esta classe é uma implementação de parte da LMAGLIB e da ITL. A função primordial da classe é a solução da equação matricial obtida pela classe TEquacao. Parte da LMAGLIB serve para interpretar os resultados em objetos matriciais para os métodos numéricos utilizados pela ITL. A ITL, por sua vez, obtém os valores do vetor de potenciais elétricos $[V]$ da equação (2.23).

Os métodos numéricos não serão descritos aqui. Porém, para permitir certa melhora da solução, a LMAGLIB possui métodos e procedimentos próprios para passar as matrizes problemas a serem resolvidas para a ITL. Por isso, tais bibliotecas foram escolhidas, para permitir essa passagem de forma eficiente.

Como a classe precisa do sistema a ser resolvido, ela toma como referência um objeto da classe TEquacao. Embora possa resolver qualquer objeto da classe TEquacao, podendo inclusive ser em paralelo, este caso não se aplica ao problema estudado. As funções de solução irão apenas colocar os valores da matriz vetor de potenciais elétricos do objeto da classe TEquacao que foi resolvido. Por isso, o objeto da classe TSolucao não armazena os valores de solução.

5.6. Classe TSaida

A última classe do problema irá realizar o pós-processamento dos dados obtidos. É uma implementação parcial da LMAGLIB para o equacionamento do item 2.4 do capítulo 2. Como é uma biblioteca que trabalha tanto com as matrizes de cada elemento como o sistema global, além dos valores dos potenciais elétricos, esta matriz toma como referência um objeto da classe TSolucao e o objeto da classe TEquacao que foi resolvido. Essa ligação é intrínseca na metodologia descrita, não precisando ser explícita na implementação utilizada.

Vale ressaltar aqui que os resultados finais obtidos são escritos em arquivos simples para efeito de saída. No início do processo da implementação, visou-se utilizar o Gmsh para a visualização de potenciais elétricos em diversos planos. Porém um contato com o Prof. Christophe Geuzaine (responsável pelo projeto do Gmsh) revelou uma falha no uso da FLTK [12] para visualização de variável parametrizada. Até a resolução de tal problema, a implementação não passa esse dado para visualização no Gmsh, apenas para saída ASCII. Mesmo assim, os valores podem ser conferidos e validados, conforme será visto no próximo capítulo. O uso de uma outra versão do Gmsh possibilitaria a visualização, mas o algoritmo de geração de malha está o melhor atual na versão utilizada. Por isso, foi escolhido não ter a visualização para se ter uma malha melhor gerada.

6. Aplicações e Avaliação de Resultados

Uma vez obtidos os resultados do processo de simulação, podemos verificar se estes são satisfatórios. Resultados satisfatórios, para um problema do Método de Elementos Finitos, podem ser considerados como valores com boa precisão para uma malha bem descrita e refinada, próximos dos valores teóricos ou esperados. A validação dos resultados obtidos permite, em primeiro lugar, verificar a sanidade do programa, para corrigir eventuais erros de implementação, programação ou projeto. Em segundo lugar, a validação permite verificar os cuidados que devem ser tomados para a execução da simulação (como refinamento da malha, critérios de convergência, complexidade da geometria, etc.).

Devido a dificuldades de realização em campos confiáveis e de obtenção de dados de projeto de sistemas de aterramento que pudessem ser testados, resolvemos comparar os resultados com dois problemas já consagrados pela literatura. O primeiro problema é a análise da distribuição de potencial de uma haste condutora aterrada verticalmente em um solo homogêneo, cuja solução analítica é facilmente obtida. O segundo problema é a análise de um sistema de aterramento de malha regular reticulada enterrado em solo estratificado em duas camadas, cuja solução é feita e avaliada pelo programa SGA [15].

6.1. Haste Vertical Aterrada Verticalmente

Neste caso, iremos analisar os resultados obtidos da simulação de um sistema de aterramento feito de uma haste condutora aterrada verticalmente, em solo homogêneo. Esses resultados serão comparados com os valores obtidos da solução analítica do problema. Os valores que serão comparados, para efeito de validação, serão a distribuição de potencial elétrico no nível do solo e a resistência de aterramento do sistema.

Para este caso, a resistência de aterramento do sistema pode ser obtida da expressão [13]:

$$R_{aterramento} = \frac{\rho}{2 \cdot \pi \cdot l} \cdot \ln\left(\frac{2 \cdot l}{r}\right) \qquad (6.1)$$

na qual:

$l \rightarrow$ comprimento da haste $[m]$

$r \rightarrow$ raio da haste $[m]$

$\rho \rightarrow$ resistividade do solo homogêneo $[\Omega \cdot m]$

Já a variação do potencial elétrico no plano do solo para a haste aterrada é obtida da expressão [13]:

$$V(y) = \frac{V_0}{2 \cdot \ln\left(\frac{2 \cdot l}{r}\right)} \cdot \ln\left(\frac{l + \sqrt{l^2 + y^2}}{-l + \sqrt{l^2 + y^2}}\right) \qquad (6.2)$$

na qual y é uma direção radial passando pelo centro da haste, sendo obrigatório para a expressão (6.2) que y > r (lembrando que y e r devem ter a mesma unidade).

Para a análise deste problema com os valores oriundos de uma simulação, foi adotado um solo homogêneo de resistividade 300 $\Omega \cdot m$ e uma haste condutora de 2,5 m de comprimento e 2 cm de raio.

A simulação deste problema gerou uma malha não melhorada de 5214861 nós, para 5138724 elementos (contando elementos tridimensionais, bidimensionais e unidimensionais). A malha é considerada excessivamente refinada frente às diversas simetrias e simplificações que o problema apresenta, porém gerou resultados muito precisos do problema (tomando um alto custo computacional).

Na Figura 6.1 temos a curva do potencial elétrico calculado pela expressão (6.2) em comparação com o obtido pela simulação, para um eixo radial passando pelo centro da figura no plano do solo. O eixo obtido foi a reta de equação x = y para z = 0 (plano do solo), sendo que o centro da barra estava passando pelo eixo Z.

Podemos observar que, frente à malha obtida, os valores estão bem próximos do analítico. Isso permite validar a hipótese do Método de Elementos Finitos que diz que a precisão e sanidade do resultado obtido são condicionadas pelo refinamento da malha. Por outro lado, uma malha extremamente refinada aumenta a dimensão da matriz problema, causando um maior tempo e custo computacional para a

resolução do problema. Uma abordagem melhorada seria a obtenção da boa relação entre refinamento da malha e precisão do resultado obtido.

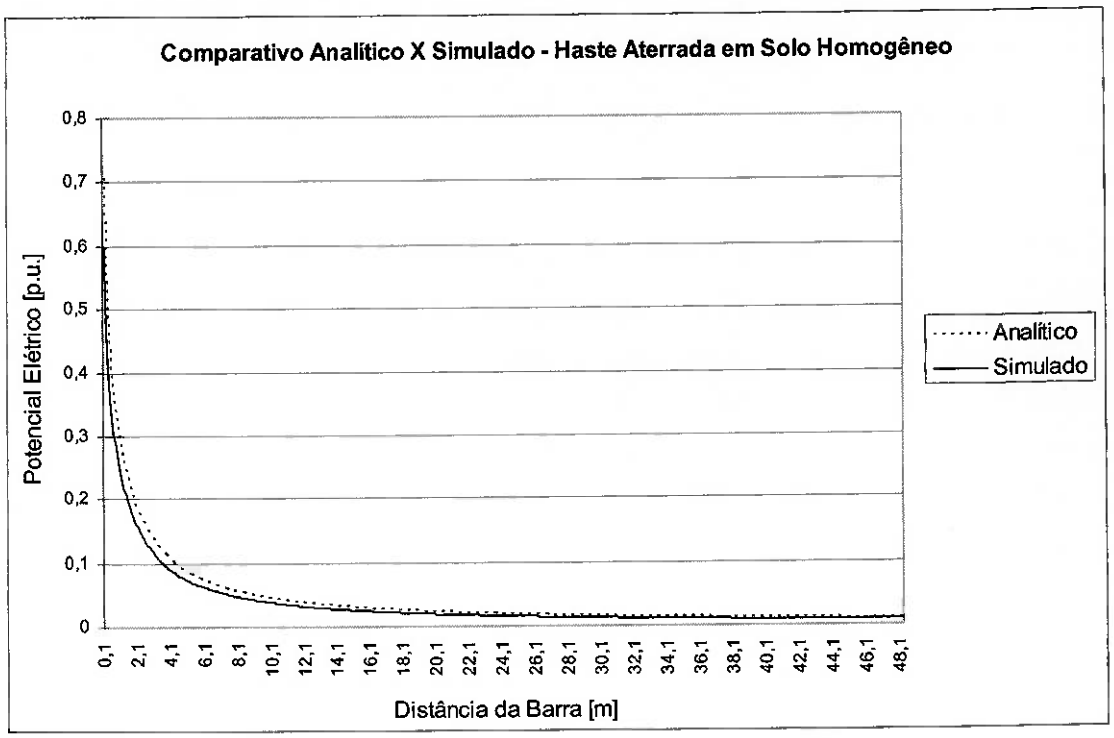


Figura 6.1 – Curva comparativa dos potenciais elétricos (em p.u. da tensão de falta) obtidos pelo cálculo analítico e pela simulação.

Substituindo os valores do problema na equação (6.1), obtemos uma resistência de aterramento de 105,45 Ω. Pela simulação, obtemos o valor de 104,998 Ω. Como a malha é muito refinada, e o problema é de simples solução, o resultado preciso já era esperado, face à conclusão da hipótese do Método de Elementos Finitos mencionada anteriormente.

6.2. Sistema de Aterramento de Malha Regular Reticulada

O IEEE Std 80 [15] apresenta em seu Apêndice A a simulação e os resultados de um sistema de aterramento de malhar regular enterrado em solo estratificado de duas camadas. A simulação deste problema foi feita pelo programa SGA desenvolvido pelo EPRI-PEAC [20] cuja formulação não é divulgada. Porém, sendo um programa consagrado pelas grandes empresas de energia elétrica, seus

resultados possuem alta credibilidade para serem usados como parâmetro de validação.

O anexo A apresenta as características do estudo, bem como os resultados do potencial elétrico ao longo de uma linha. No caso IEEE-80, é dado que a resistência de aterramento é de 1,1 Ω, o potencial de toque é de 50 V [15].

A simulação deste problema gerou uma malha não melhorada de 37685749 nós, para 36756531 elementos (contando elementos tridimensionais, bidimensionais e unidimensionais). Também foi utilizada uma malha extremamente refinada, embora o problema, mesmo consagrado, não tenha uma solução muito fácil. Ao preço de um alto custo computacional, no entanto, os resultados foram muito próximos dos especificados na literatura do caso.

A geometria do problema está na Figura 6.2. Nela foi marcada a direção na qual foram obtidos os potenciais elétricos. Note que, embora o centro da malha não esteja no ponto (0,0,0), ainda sim podemos traçar o gráfico de potencial no eixo indicado começando no ponto central do sistema de aterramento, bastando observar para quais valores de posição desejamos obter o potencial elétrico.

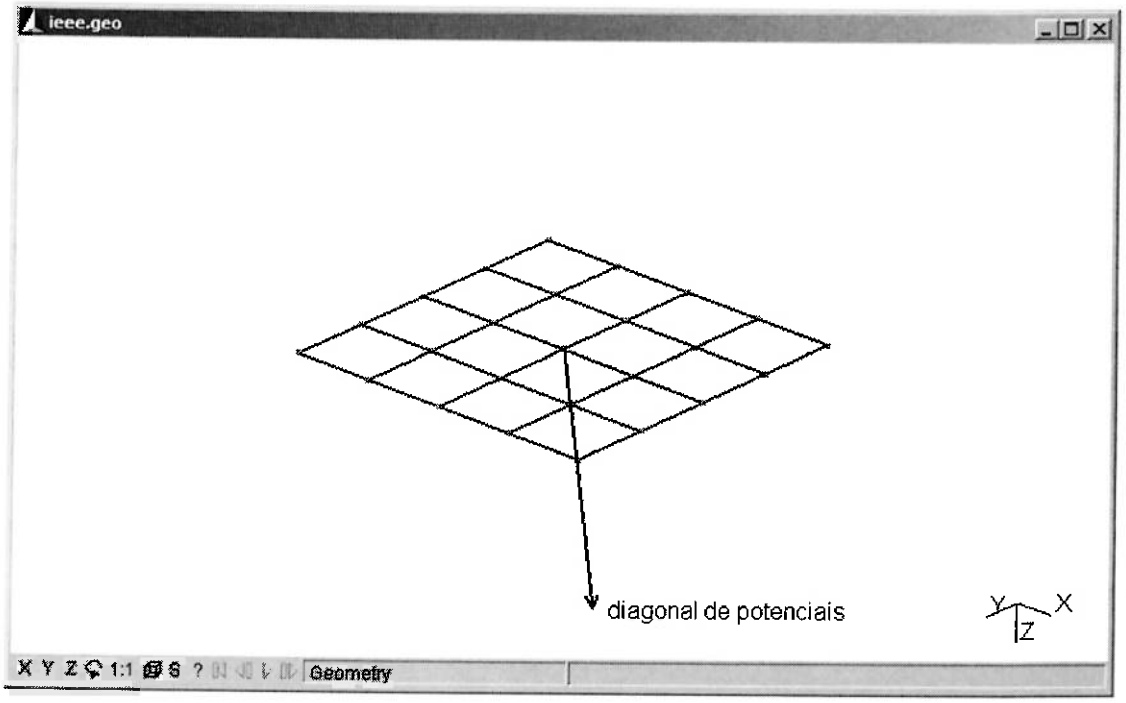


Figura 6.2 – Geometria modelada do caso IEEE Std 80.

Já na Figura 6.3 temos o gráfico do potencial elétrico na diagonal de potenciais da Figura 6.2, bem como a distância relativa ao ponto central do sistema

de aterramento, no eixo de estudo. Esse potencial elétrico é equivalente ao potencial de passo em relação à diagonal estudada. Na norma IEEE Std 80, essa diagonal de potenciais é a que define o potencial de passo de um sistema de aterramento.

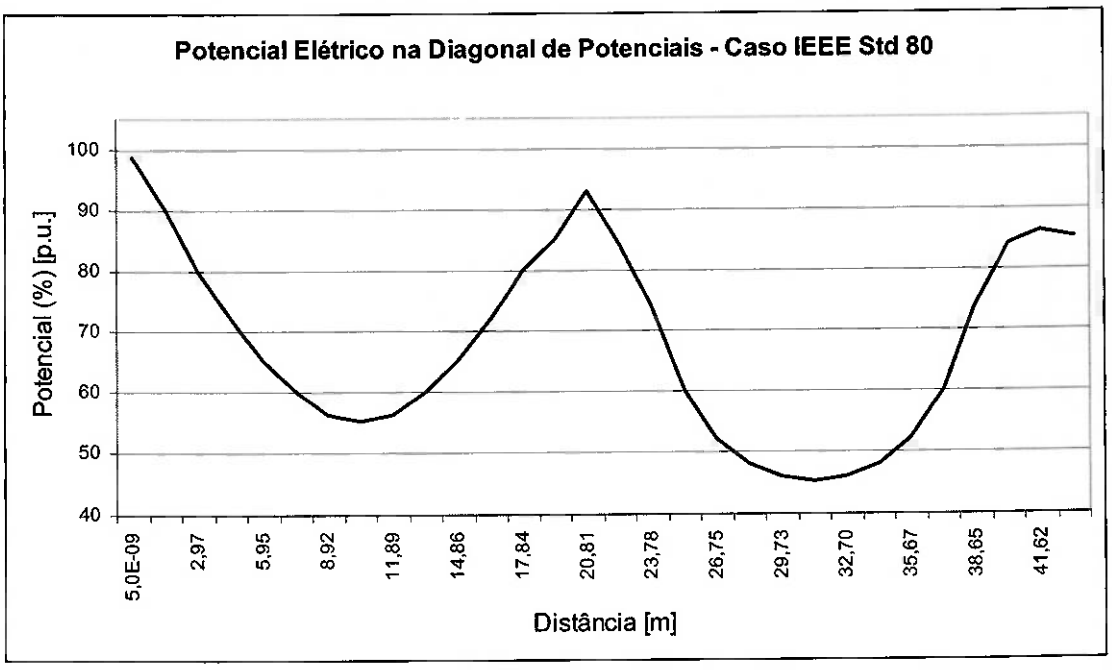


Figura 6.3 – Curva dos potenciais elétricos no plano do solo na diagonal de potenciais do caso IEEE Std 80 obtidos pela simulação do programa.

Note que os valores de distância diferem do apresentado na especificação IEEE Std 80 [15] (gráfico do anexo A). Isso ocorre porque a distância no gráfico é dada em relação a dois eixos passando pelo centro da malha, e em porcentagem relativa ao tamanho máximo em cada direção dessa metade. Isso significa que o gráfico dado na norma possui eixos com valores relativos (porcentagens de um valor nominal), e no caso dos eixos, é do tamanho máximo em cada direção. Isso serve para criar uma curva normalizada, de forma a se definir uma norma. No gráfico apresentado na Figura 6.3, a distância é o valor real, ou seja, a distância do centro do sistema de aterramento na diagonal de estudo.

Podemos observar, frente a isso, que os valores de potencial são relativamente próximos. A curva se comporta da mesma forma que a curva da norma. Os valores de resistência de aterramento também são coerentes. Na simulação obtemos uma resistência de malha de 0,93 Ω. É um valor preciso e coerente com o valor descrito na norma (de 1,1 Ω).

O potencial de toque calculado na norma pelo SGA do EPRI [20] é feito para uma corrente de falta de 2500 A. Essa corrente é considerada como um padrão da norma para o cálculo do potencial de toque. Maiores detalhes do cálculo do potencial de toque estão na norma IEEE Std 80 [15]. O programa mostrou um valor de aproximadamente 50 V para o potencial de toque (que é o considerado do modelo utilizado na norma, o qual está sendo analisado). Adotando o mesmo processo de cálculo, chegamos num potencial de toque de 42 V. Embora a precisão obtida pela simulação seja questionável, ainda sim se encontra numa faixa aceitável. Como é desconhecida a formulação do SGA do EPRI [20], não podemos concluir muito sobre a precisão deste valor. Embora, ainda sim, esteja numa faixa bastante aceitável.

A precisão de resultados ocorre pelo uso de uma malha grande e bem refinada. Porém, isso causou um alto custo computacional, tanto de alocação de memória quanto de processamento desprendido. Por isso, em um projeto utilizando o Método de Elementos Finitos, deve-se observar o compromisso de se obter uma resposta adequada, com uma boa precisão, e uma malha que não necessite de tanta capacidade para ser equacionada e resolvida. Neste projeto não há processo de melhoria da malha, ela é utilizada em um refinamento considerado de alto grau, independente do problema. Em etapas futuras, tal melhoria pode ser estudada e implementada, para melhorar o tempo gasto nas simulações.

7. Conclusão

O trabalho permitiu se verificar uma boa aplicação de ferramentas livres para a simulação por Método de Elementos Finitos de sistemas de aterramento. Embora o custo computacional para aplicações razoavelmente simples tenha deixado claro que existem diversas linhas de melhorias, a coerência dos resultados com os esperados mostrou que a implementação, em seu escopo, conseguiu atingir os objetivos iniciais. Embora exista uma escassez em casos a serem analisados, aqueles estudados já forneceram uma boa fonte de comparação para o estudo.

No entanto, é clara a necessidade de melhorias no desempenho da implementação. Estaremos comentando algumas delas, que foram consideradas as mais críticas. Outras melhorias, menores, também podem ser feitas, mas qualquer implementação computacional de solução de algum problema é um trabalho contínuo, o que demanda um longo tempo até se convergir a um ponto ótimo. Mas as grandes melhorias não se encaixam nos objetivos deste trabalho, e por isso não foram estudadas, aprofundadas ou implementadas.

Uma melhoria importante, que talvez seja a mais crítica no aspecto de implementação computacional, seja o acoplamento do Gmsh, ou de outras ferramentas de geometria e discretização de domínios de estudo, ao código da implementação principal, como uma classe ou um conjunto de classes. O uso de um aplicativo separado do programa traz inconsistências em relação às suas dependências. Mas, acima disso, a passagem de dados se torna lenta, pois não pode ser feita diretamente pela memória (já que o sistema operacional não permite isso). O uso do código-fonte direto do Gmsh na aplicação traria maior velocidade e estabilidade ao seu uso. Isso porque o acoplamento seria direto, como nas outras bibliotecas, e a capacidade de intercâmbio de dados seria muito mais versátil e rápida. Porém, tal tarefa demandaria tempo e esforço que foge do escopo deste trabalho. Por isso, para o nosso caso, optou-se por utilizar o aplicativo já compilado em conjunto com a implementação estudada.

Em relação à discretização do domínio, como já foi dito, não foi aplicado nenhum processo, seja manual ou automático, de estudo da geometria do problema. Foi adotada uma malha refinada sem verificação, que fornece alta precisão sem se preocupar com o custo computacional ou a demanda de memória para o problema. Isto pode causar a inviabilidade de um problema até que simples, dependendo do

caso. Um estudo de melhoria seria verificar a geometria, ou permitir que o usuário o faça, para então se discretizar o domínio de estudo. Desta forma, o usuário poderia verificar onde na geometria deseja uma maior ou pior precisão dos resultados, podendo controlar a demanda e o tempo utilizados na simulação.

Além disso, os critérios de truncamento poderiam ser melhorados, analisando-se metodologias para que o infinito possa ser modelado de outra forma. E também é interessante estudar possíveis simetrias do problema e técnicas de se reduzir o domínio de estudo utilizando-as. Isso permitiria uma redução da malha de elementos finitos, e conseqüentemente do sistema de equações a ser resolvido, melhorando o desempenho da implementação.

Também se deve olhar pelo método numérico de solução do sistema linear do problema, implementado pela ITL. Em geral, as bibliotecas de domínio público de Álgebra Linear possuem um desempenho muito bom, pois são feitas e mantidas por grupos especialistas nos assuntos. No caso específico da ITL, ela é uma biblioteca altamente adaptável, flexível e compreensiva, e de fácil integração com a LMAGLIB, além de ser uma biblioteca altamente especializada em sistemas lineares. O método escolhido da ITL consistiu na abordagem da metodologia do gradiente bi-conjugado estabilizado, embora outras abordagens pudessem ser feitas. Porém, o método mais eficiente para tal implementação deveria considerar o tipo de matriz a ser abordada e o melhor método para cada tipo de matriz-problema, ao invés de utilizar um método genérico considerando apenas uma matriz-problema simétrica, como foi o caso. Como obtemos matrizes esparsas na maioria das vezes com o Método de Elementos Finitos, uma avaliação da matriz antes seria uma melhor abordagem, a fim de se utilizar métodos de melhor desempenho para cada tipo de matriz. No entanto, a análise da matriz-problema foge do escopo aqui utilizado, pois demanda um nível de matemática que não se aplica ao trabalho. Mas poderia ser uma abordagem a ser visitada, a fim de melhorar o desempenho da implementação.

Referências Bibliográficas

[1] FILHO, M. L. P. – *Aplicação do Método de Imagens Complexas ao Cálculo de Malhas de Aterramento em Solos com Estratificação Horizontal* – Tese (Mestrado), Escola Politécnica da Universidade de São Paulo, São Paulo, 1999.

[2] CARDOSO, J. R. – *O Método dos Elementos Finitos Aplicado na Determinação da Resistência de Aterramento* – Anais do Seminário sobre Cálculo de Campo Elétrico com Métodos Numéricos, Santo André, 1985.

[3] CARDOSO, J. R.; RIBEIRO, F. S.; GAMBIRASIO, G. – *O Método dos Elementos Finitos no Modelamento de Sistemas de Aterramento em Solos de Múltiplas Camadas* – Anais do IX SNPTEE, Belo Horizonte, 1987.

[4] CEDRAT Technologies – Cedrat *Flux, One Step Ahead* - *http://www.cedrat.com/software/flux/pdf/flux.pdf*

[5] CARDOSO, J. R. – *Introdução ao Método dos Elementos Finitos para Engenheiros Eletricistas* – EPUSP, São Paulo, 1995.

[6] TURNER, L. R.; ARGONNE NAT. LAB, IL – *3-D Field Computation: The Near-Triumph of Commercial Codes* – Magnetics, IEEE Transactions, Jul 1996, volume 32, issue 4, pp. 2945-2949.

[7] The Free Software Foundation - *http://www.fsf.org/.*

[8] GEUZAINE, C.; REMACLE, J. F. – *Gmsh: A Three-Dimensional Finite Element Mesh Generator with Built-In Pre and Post-Processing Facilities* – *http://www.geuz.org/gmsh/* - Version 2.0.8, May 14, 2006

[9] LUMSDAINE, A.; LEE, L. Q.; SIEK, J. – *ITL: The Iterative Template Library* – *http://osl.iu.edu/research/itl/* - July 27, 1998.

[10] CARDOSO, J. R. – *Problemas de Campos Eletromagnéticos Estáticos e Dinâmicos: Uma Nova Abordagem pelo Método dos Elementos Finitos* – Tese (Doutorado), Escola Politécnica da Universidade de São Paulo, 159 pp., São Paulo, 1985.

[11] CARDOSO, J. R. – *GROUND 3D: Uma Contribuição à Análise dos Sistemas de Aterramento pelo Método de Elementos Finitos* – Tese (Livre-Docência), Escola Politécnica da Universidade de São Paulo, 71 pp., São Paulo, 1993.

[12] BRODSKYN, H. O. – *GROUND 3D Acoplado: Uma Nova Contribuição ao Cálculo da Distribuição de Corrente de Defeito* – Tese (Doutorado), Escola Politécnica da Universidade de são Paulo, 64pp., São Paulo, 1996.

[13] ORSINI, L. Q.; CAMARGO, L. B. – *Eletromagnetismo* – EPUSP, São Paulo, 2003.

[14] CARDOSO, J. R.; LEBENSZTAJN, L.; ANTONIO, J. M. – *Geração Automática de Elementos: Uma Experiência com a Triangularização de Delaunay* – in: Simpósio Franco-Brasileiro sobre Cálculo de Campos Elétricos e Magnéticos, 2, São Paulo, 1989 – Anais, São Paulo, EPUSP, Março 1989.

[15] ANSI/IEEE Std 80, 1986 – *IEEE Guide for Safety in AC Substation Grounding* – Published by IEEE, New York, John Wiley, 1986.

[16] FREEMAN, E. M.; LOWTHER, D. A. – *An Open Boundary Technique for Axisymmetric and 3D Magnetic and Electric Fields Problems* – Magnetics, IEEE Transaction, October 1989, volume 25, issue 5, pp. 4135-37.

[17] NABETA, S. I. – *Solução de Problemas Magnetostáticos com a Utilização do ICCG* – Dissertação (Mestrado), Escola Politécnica da Universidade de São Paulo, 64 pp., São Paulo, 1990.

[18] HORSTMANN, C. S. – *Practical Object-Oriented Development in C++ and Java* – John Wiley & Sons Inc., New York, 1997.

[19] HENRICSON, F. N.; NYQUIST, E. – *Programming in C++, Rules and Recommendations* – rev. C, Ellemtel Telecom. Systems Laboratories, Sweden, 1992.

[20] MELIOPOULOS, A. P. S. – *Substation Grounding Programs* – Electric Power Research Inst. (EPRI), Palo Alto, CA, United States, May, 1992.

[21] SPITZAK, B. – *Fast Light Toolkit (FLTK)* – http://www.fltk.org/, 2006

Apêndice A – Algoritmo de Geração de Malha Reticulada

```
NumPonto ← 1

/* x, y e z são todos números naturais, e serão as identificações de posição em cada eixo. */
Para z indo de 0 até n_z
    {
    Para y indo de 0 até n_y
    {
        Para x indo de 0 até n_x
        {
            Ponto (NumPonto) = {x · l_x, y · l_y, z · l_z}
            NumPonto ← NumPonto + 1
        }
    }
}

/* uma vez gerados os pontos, podemos então gerar as linhas, que representam as barras condutoras */
NumLinha ← 1

/* passar por todas as camadas do plano z */
Para z indo de 0 até n_z
{
    /* desenhar as barras paralelas ao eixo x no plano z */
    NumPonto ← z · ((n_x + 1) · (n_y + 1)) + 1
    Para y indo de 0 até n_y
    {
        Para x indo de 1 até n_x
        {
            Linha (NumLinha) = {NumPonto, NumPonto + 1}
            NumPonto ← NumPonto + 1
            NumLinha ← NumLinha + 1
        }
        NumPonto ← NumPonto + 1
    }

    /* desenhar as barras paralelas ao eixo y no plano z */
    NumPonto ← z · ((n_x + 1) · (n_y + 1)) + 1
    Para y indo de 1 até n_y
    {
        Para x indo de 0 até n_x
        {
            Linha (NumLinha) = {NumPonto, NumPonto + n_x +1}
```

```
            NumPonto ← NumPonto + 1
            NumLinha ← NumLinha + 1
        }
    }

    /* se atingida a última camada de z, desenhar as barras palalelas ao eixo z */
    Se z = n_z então faça
    {
        NumPonto ← z · ((n_x + 1) · (n_y + 1)) + 1

        Faça enquanto NumPonto ≤ (z + 1) · ((n_x + 1) · (n_y + 1))
        {
            Linha (NumLinha) = {NumPonto, NumPonto + ((n_x + 1) · (n_y + 1))}
            NumPonto ← NumPonto + 1
            NumLinha ← NumLinha + 1
        }
    }
}
```

Apêndice B – Geração de Malha de Barras Aterradas (n < 3)

/* analisamos os casos de n = 1 e n = 2 separadamente, mas em ambos os casos os pontos e linhas são descritos manualmente */

```
Se n = 2 então faça
{
    Ponto (1) = {0, 0, 0}
    Ponto (2) = {0, 0, b}
    Linha (1) = {1, 2}

    Ponto (3) = {1, 0, 0}
    Ponto (4) = {1, 0, b}
    Linha (2) = {3, 4}

    Linha (3) = {1, 3}
}

Caso contrário, se n = 1 faça
{
    Ponto (1) = {0, 0, 0}
    Ponto (2) = {0, 0, b}
    Linha (1) = {1, 2}
}
```

Apêndice C – Geração de Malha de Barras Aterradas (n ≥ 3)

/* supondo n ≥ 3, primeiro iremos gerar os pontos do polígono, tanto para o plano de z = 0 quanto para o plano z = b, e iremos unir sempre o ponto no primeiro plano com o segundo, para ser a barra condutora */

```
NumPonto ← 1
NumLinha ← 1
Θ ← 0
x ← 0
y ← 0
```

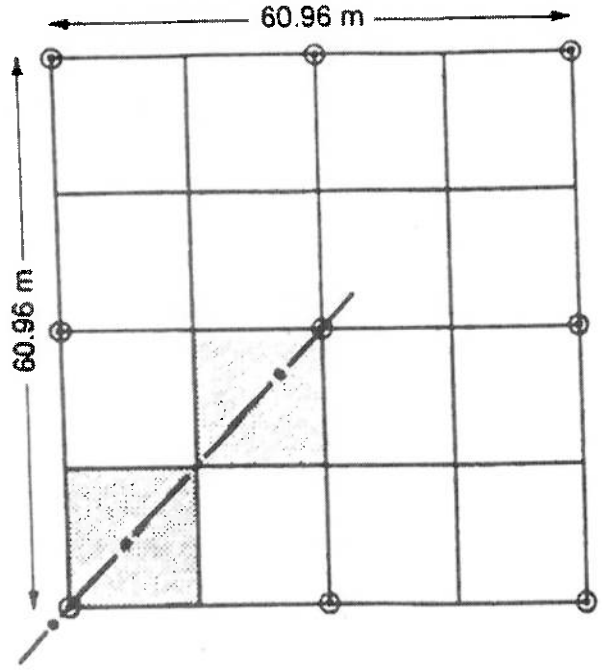
$$\alpha \leftarrow \frac{360^\circ}{n}$$

```
Faça enquanto Θ < 360°
{
    Ponto (NumPonto) = {x, y, 0}
    NumPonto ← NumPonto + 1
    Ponto (NumPonto) = {x, y, b}
    Linha (NumLinha) = {NumPonto, NumPonto - 1}
    NumPonto ← NumPonto + 1
    NumLinha ← NumLinha + 1
    Θ ← Θ + α
    x ← x + 1 · cos(α)
    y ← y + 1 · sin(α)
}
```

/* agora, iremos unir os pontos do polígono em z = 0, conforme o item 3.2 */

```
Limite ← NumPonto
Para NumPonto indo de 1 até Limite com incremento de 2
{
    Se NumPonto = Limite - 2 então faça
    {
        Linha (NumLinha) = {NumPonto, 1}
    }
    Caso contrário faça
    {
        Linha (NumLinha) = {NumPonto, NumPonto + 2}
    }
    NumLinha ← NumLinha + 1
}
```

Anexo A – Caso IEEE Std 80



Legend

- Grid Conductor
- Ground Rod
- Diagonal Voltage Profile Shown in Fig C5

ρ_1 = 300 Ω-m
ρ_2 = 100 Ω-m
H = 4.572 m
D = 15.24 m
d = 0.01 m
h = 0.5 m
Rod length = 9.144 m