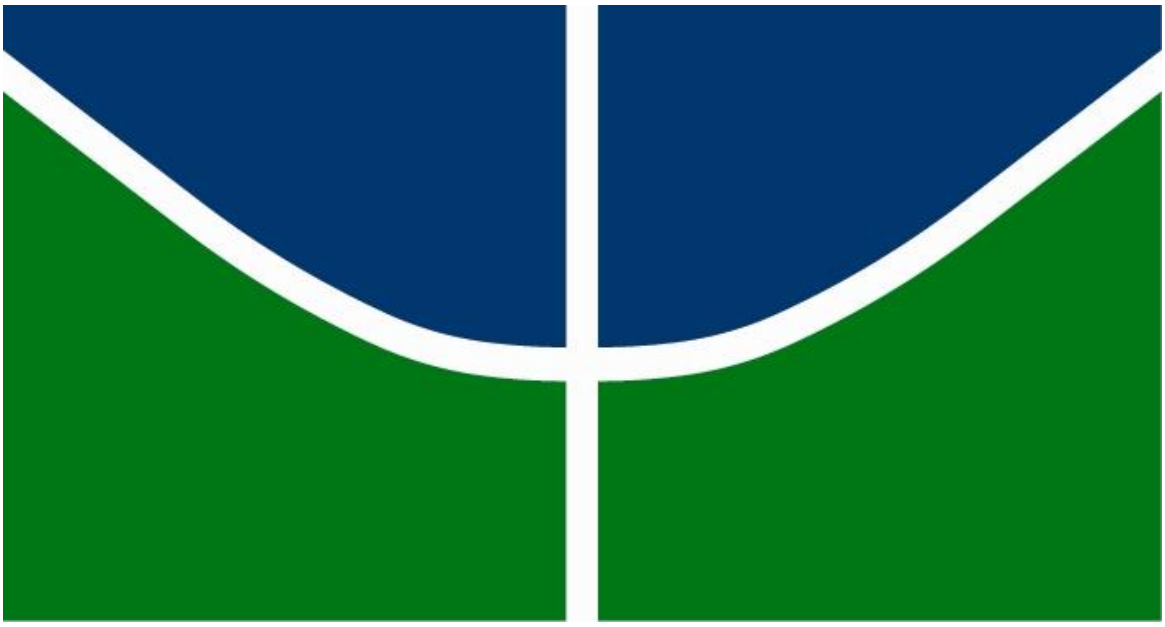


BEMLAB2D: INTERFACE GRÁFICA DE MODELAGEM, VISUALIZAÇÃO E ANÁLISE COM ELEMENTOS DE CONTORNO – UMA APLICAÇÃO EM PROBLEMAS ELASTOSTÁTICOS

ALVARO MARTINS DELGADO NETO

DISSERTAÇÃO DE MESTRADO EM ESTRUTURAS E CONSTRUÇÃO CIVIL
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

BEMLAB2D: INTERFACE GRÁFICA DE MODELAGEM, VISUALIZAÇÃO E ANÁLISE COM ELEMENTOS DE CONTORNO – UMA APLICAÇÃO EM PROBLEMAS ELASTOSTÁTICOS

ALVARO MARTINS DELGADO NETO

ORIENTADOR: GILBERTO GOMES, DSc

DISSERTAÇÃO DE MESTRADO EM ESTRUTURAS E CONSTRUÇÃO CIVIL

PUBLICAÇÃO:
BRASÍLIA/DF: MARÇO – 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL

BEMLAB2D: INTERFACE GRÁFICA DE MODELAGEM, VISUALIZAÇÃO E ANÁLISE COM ELEMENTOS DE CONTORNO – UMA APLICAÇÃO EM PROBLEMAS ELASTOSTÁTICOS

ALVARO MARTINS DELGADO NETO

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA CIVIL E AMBIENTAL DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ESTRUTURAS E CONSTRUÇÃO CIVIL.

APROVADO POR:

Prof. Gilberto Gomes, DSc. (PECC - UnB)
(Orientador)

Prof. Luciano Mendes Bezerra, PhD. (PECC - UnB)
(Examinador interno)

Prof. Márcio Augusto Roma Buzar, DSc. (PPG FAU - UNB)
(Examinador Externo)

BRASÍLIA/DF, 31 DE MARÇO DE 2017.

FICHA CATALOGRÁFICA

DELGADO NETO, ALVARO M.

DD352b – BEMLAB2D: INTERFACE GRÁFICA DE MODELAGEM, VISUALIZAÇÃO E ANÁLISE COM ELEMENTOS DE CONTORNO – UMA APLICAÇÃO EM PROBLEMAS ELASTOSTÁTICOS / Alvaro Martins Delgado Neto; orientador Gilberto Gomes. – – Brasília, 2017.
113 p.

Dissertação (Mestrado – Mestrado em Estruturas e Construção Civil) – – Universidade de Brasília, 2017.

1. Interface Gráfica de usuário
2. Método dos Elementos de Contorno
3. Método dos Elementos de Contorno Dual
4. Elastostática
5. MATLAB
6. Modelos Bidimensionais

I. Gomes, Gilberto, orient
II. Título.

REFERÊNCIA BIBLIOGRÁFICA

DELGADO NETO, ALVARO M. (2017). BEMLAB2D: Interface Gráfica de Modelagem, Visualização e Análise com Elementos de Contorno – Uma Aplicação em Problemas Elastostáticos. Dissertação de Mestrado, Publicação 005A/2017, Departamento de Engenharia Civil e Ambiental, Universidade de Brasília, DF, 113p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Alvaro Martins Delgado Neto
TÍTULO DA DISSERTAÇÃO DE MESTRADO: BEMLAB2D: Interface Gráfica de Modelagem, Visualização e Análise com Elementos de Contorno – Uma Aplicação em Problemas Elastostáticos.
GRAU: Mestre **ANO:** 2017

Alvaro Martins Delgado Neto
SQN 404, bl C, Térreo 02 – Asa Norte

CEP: 70845-030 – Brasília/DF – Brasil

"Eu sei como você se sente neste momento.
Mas eu lhe prometo, apesar do mundo parecer
um local escuro e assustador, haverá luz!"

JAMES GORDAN – STAN LEE

DEDICATÓRIA

Dedico esta dissertação com todo amor aos meus pais, Antonio Martins e Maria Haide, pela força e proteção a mim proporcionadas, através de suas fortes orações, durante o período de desenvolvimento deste trabalho. Tudo que eu conquistei até hoje, eu dedico a vocês, e a vocês entrego toda a minha gratidão por várias vezes pensarem mais em mim do que em vocês próprios.

AGRADECIMENTOS

A minha mãe Haide, meu pai Antonio e meu irmão Paulo Renato, pela compreensão, paciência e apoio dados, diante da minha decisão em vir fazer o mestrado. Amo muito vocês.

Aos amigos no Rio Grande do Norte, que me apoiaram e me incentivaram a minha vinda pra Brasília em um dos momentos mais difíceis da minha vida. Essa decisão eu não tomei sozinho, obrigado a todos.

A pessoa mais importante da minha vida, minha pequena Alana Parente, que durante esses últimos anos mesmo longe sempre me apoiou e, com muita paciência, me esperou com seu "abraço-casa" para me receber de volta. As poucas vezes que nos vimos, o pouco tempo que ficávamos juntos me dava forças para voltar a Brasília e acreditar que eu podia continuar e que e ia terminar essa jornada. "Feliz do homem que tem a amiga, a namorada e a amante na mesma mulher" ... e eu tenho você. Te amo minha princesa.

Aos professores da UFRN, Roberto pelos conselhos e a quem devo toda minha vontade de ser professor também, Fernanda por todo o apoio desde a época de monitoria até hoje como uma grande amiga com seus apoios e Selma pelos conselhos e incentivo dados a minha opção por fazer o mestrado em Brasília, a todos você eu devo uma imensa gratidão meus mestres.

Ao professor Gilberto Gomes, pelo incentivo, orientação, paciência e, principalmente, pela amizade e dedicação demonstradas ao longo desses dois anos.

Ao professor Luciano que com seus conselhos, as vezes, deslocados na aula de matemática me mostrou que não é um pequeno obstáculo que iria atrapalhar meu sonho de me tornar professor.

Aos amigos de mestrado, Guilherme "meu segundo irmão" parceiro pro resto da vida, Nicolas "el perro" que ainda vai me levar pra conhecer o mundo, Jéssica "Uai sô, trem bão" me jogando numa cachoeira eu fico feliz, Fabiano "cabeção cearense" que me ajudou muito quebrando a cabeça comigo nas programações, Eduardo Pains "o mineirinho Dudu" e seu vozeirão nas noites apaixonadas de lua cheia tocando na "Rádio PECC" e Iago

“Coqueta do meu core” por me fazer rir todos os dias desse meninão alegre que vive dentro de você.

À CAPES, pelo apoio financeiro.

A Alexandre, Aldair, Eva e demais funcionários que fazem ou fizeram o meu dia-a-dia no SG12 ser diferente de uma rotina.

A **DEUS**, o nosso criador e como gosto de chama-lo “Meu Amigo confidente tímido”.

RESUMO

O uso de Interface Gráfica de Usuário (GUI) vem assumindo um papel importante como ferramenta visual de suporte aos programas computacionais no campo da engenharia. O emprego de GUIs facilita o processo de criação e manutenção da modelagem geométrica, proporciona maior flexibilidade na construção de malhas, na inserção de condições de contorno e de propriedades físicas – fase de pré-processamento, bem como possibilita interpretar de forma direta e visual os resultados da análise, conhecida como fase de pós-processamento.

Por outro lado, a fase de processamento requer a utilização de técnicas numéricas de análise para representar e solucionar o modelo de engenharia, como o Método dos Elementos Finitos (MEF) e o Método dos Elementos de Contorno (MEC), embora este último tenha se destacado como mais eficiente e flexível, permitindo maior versatilidade aos programas de modelagem geométrica e de malha. O MEC se destaca, dentre os outros métodos numéricos, devido ao seu modelo de discretização, ocorrendo apenas no contorno, o que torna sua implementação simples e versátil.

Neste contexto, este trabalho propõe o desenvolvimento de uma Interface Gráfica de Usuário (GUI) implementada utilizando a linguagem MATLAB, denominada BEMLAB2D. Este software permite a construção de modelos bidimensional com geometrias complexas, inclusões, furos e trincas, bem como editar e visualizar os resultados gerados pelo programa BemCracker2D, construído em C++ para analise via MEC. Como complemento, o BEMLAB2D permite também a geração de malhas de MEF apenas com elementos triangulares e de Métodos Sem Malhas (MESHLESS), ambas geradas a partir da malha de MEC.

Palavras-Chaves: *Interface Gráfica de Usuário, Gerador de Malhas, Método dos Elementos de Contorno, Elastostática, MATLAB, Modelos Bidimensionais.*

ABSTRACT

Using Graphical User Interface (GUI) has assumed an important role as a visual tool support for computer programs in the field of engineering. The use of GUIs facilitates the process of creating and maintaining the geometric modeling provides greater flexibility in building meshes, the insertion of boundary conditions and physical properties - preprocessing step, and enables interpreting directly and visually the results of the analysis, known as the post-processing.

On the other hand, the processing step requires the use of numerical analysis techniques for representing and resolving the engineering model, such as the Finite Element Method (FEM) and Boundary Element Method (BEM), although the latter is highlighted as more efficient and flexible, allowing greater versatility to the geometric and meshes modeling programs. The BEM stands out among the other numerical methods due to its discretization model, occurring only in boundary, which makes a simple and versatile implementation.

In this context, this work proposes the development of a User Graphical Interface (GUI) implemented with MATLAB language, called BEMLAB2D. This software allows the construction of two-dimensional models with complex geometries, inclusions, holes and cracks, as well as edit and visualize the results generated by the program BemCracker2D, built in C ++ for analysis via BEM. As complement, BEMLAB2D also allows the FEM mesh generation with only triangular elements and Meshless Methods (MESHLESS), both generated from the BEM mesh.

Keywords: *Grafical User Interface, Mesh Generator, Boundary Element Method, Elastostatic, MATLAB, Two-dimensional Models.*

SUMÁRIO

1 - INTRODUÇÃO 1

1.1 - GENERALIDADES 1

1.2 - MOTIVAÇÃO E OBJETIVOS 3

1.3 - ORGANIZAÇÃO DA DISSERTAÇÃO 4

2 - REVISÃO BIBLIOGRÁFICA 6

2.1 - GENERALIDADES 6

2.2 - SOBRE PROGRAMAS DE MODELAGEM 7

2.3 - SOBRE GERADOR DE MALHA 8

2.4 - SOBRE O MATLAB COM GUI 9

2.5 - SOBRE O MEC CONVENCIONAL 10

2.5.1 - Formulação Integral de Contorno 12

2.5.2 - Discretização Numérica 14

2.5.3 - Deslocamentos e Tensões em Pontos Internos 15

2.6 - SOBRE O MEC DUAL PARA MODELAGEM DE TRINCA 16

3 - MODELO PROPOSTO PARA O BEMLAB2D 18

3.1 - INTRODUÇÃO 18

3.2 - MODELO VISUAL DO BEMLAB2D 19

3.3 - MÓDULOS DO BEMLAB2D 22

3.3.1 - Módulo I - Geometry 22

3.3.1.1 - Point 22

3.3.1.2 - Lines 23

3.3.1.3 - Arcs 23

3.3.1.4 - Zones 24

3.3.2 - Módulo II - Mesh 25

3.3.2.1 - Malha de MEC 26

3.3.2.2 - Malha de MEF 27

3.3.2.3 - Malha de MESHLESS 28
3.3.3 - Módulo III – Boundary Conditions 28
3.3.3.1 - Displacement 29
3.3.3.2 - Traction 29
3.3.4 - Módulo IV – Elastostatic Analysis 30
3.3.5 - Módulo V – Graphical Results 32
3.4 - MÓDULOS DE ANÁLISE 33
4 - IMPLEMENTAÇÃO COMPUTACIONAL 36
4.1 - AMBIENTE DE DESENVOLVIMENTO 36
4.2 - ALGORITMO E IMPLEMENTAÇÃO 37
4.2.1 - Módulo I - Geometry 37
4.2.1.1 - Point 38
4.2.1.2 - Lines 39
4.2.1.3 - Arcs 40
4.2.1.4 - Zones 42
4.2.2 - Módulo II - Mesh 44
4.2.2.1 - Malha de MEC 48
4.2.2.2 - Malha de MEF 50
4.2.2.3 - Malha de MESHLESS 54
4.2.3 - Módulo III – Boundary Conditions 55
4.2.3.1 - Displacement 56
4.2.3.2 - Traction 57
4.2.4 - Módulo IV – Elastostatic Analysis 58
4.2.5 - Módulo V – Graphical Results 60
5 - EXEMPLOS DE APLICAÇÃO 64
5.1 - PRÉ-PROCESSAMENTO 64
5.1.1 - Exemplo 1 – Cavidade Circular em um Meio Infinito 65

5.1.2 - Exemplo 2 – Fração de um Círculo com um Furo Central 66
5.1.3 - Exemplo 3 – Chapa com Furo Tracionada (Arrancamento) 68
5.1.4 - Exemplo 4 – Chapa Retangular com Trinca Central Inclinada 69
5.1.5 - Exemplo 5 – Chapa Retangular com Três Trincas de Borda 71
5.1.6 - Exemplo 6 – Chapa Cruciforme com Trinca 72
5.1.7 - Exemplo 7 – Chapa Retangular com Furos e Propagação de Trinca 74
5.2 - PÓS-PROCESSAMENTO 75
5.2.1 - Exemplo 1 – Cavidade Circular Sobre Compressão Radial 76
5.2.2 - Exemplo 2 – Fração de um Círculo com um Furo Central Estendido 77
5.2.3 - Exemplo 3 – Chapa com Furo Tracionada (Arrancamento) 78
5.2.4 - Exemplo 4 – Chapa Retangular com Trinca Central Inclinada 78
5.2.5 - Exemplo 5 – Chapa Retangular com Três Trincas de Borda 79
5.2.6 - Exemplo 6 – Chapa Cruciforme com Trinca 80
5.2.7 - Exemplo 7 – Chapa Retangular com Furos e Propagação de Trinca 81
5.3 - GERADOR DE MALHA DE MEF/MESHLESS 83
5.3.1 - Exemplo 1 – Chapa Quadrada com Furos e Trinca de Borda 83
5.3.2 - Exemplo 2 – Modelo de Seção em I 86
6 - CONCLUSÕES E SUGESTÕES 88
6.1 - CONCLUSÕES 88
6.2 - SUGESTÕES PARA TRABALHOS FUTUROS 89
7 - REFERÊNCIAS BIBLIOGRÁFICAS 91

LISTA DE FIGURAS

Figura 1.1 – Modelo de malha gerado pelo BEMLAB2D: a) MEC; b) MEF. 2
Figura 1.2 – Características principais do BEMLAB2D. 4
Figura 2.1 – (a) Elemento constante; (b) elemento linear; (c) elemento quadrático. 12
Figura 2.2 – Região arbitrária composta de dois estados autoequilibrados. 12
Figura 2.3 – Esquema de modelagem com elementos quadráticos (GOMES, 2016). 17
Figura 3.1 – Arquitetura de um sistema de análises de engenharia (GOMES, 2006). 19
Figura 3.2 – Hierarquia de funcionalidade da Interface BEMLAB2D. 21
Figura 3.3 – Ambiente gráfico do programa BEMLAB2d. 21
Figura 3.4 – Módulo I - Geometry. 22
Figura 3.5 – Caixa de diálogo das coordenadas X e Y. 23
Figura 3.6 – Desenhando o segmento reto. 23
Figura 3.7 – Desenhando o segmento curvo. 23
Figura 3.8 – Direcionamento do desenho do arco. 24
Figura 3.9 – Interface gráfica que configura as zonas. 24
Figura 3.10 – Representação das zonas Mestre, Furo e Inclusão. 25
Figura 3.11 – Módulo II – Mesh. 25
Figura 3.12 – (a) Malha MEC, (b) Malha MEF e (c) Malha MESHLESS. 26
Figura 3.13 – Interface gráfica que configura o segmento reto. 26
Figura 3.14 – Caixa de diálogo – Número de elementos no segmento curvo. 27
Figura 3.15 – Caixa de questionamento – Pontos internos. 27
Figura 3.16 – Número de elementos da trinca. 28
Figura 3.17 – Quantidade de eixos conforme o nível de refinamento. 28
Figura 3.18 – Módulo III – Boundary Condition. 28
Figura 3.19 – GUI – Displacement. 29
Figura 3.20 – GUI – Traction. 30
Figura 3.21 – Módulo IV – Boundary Condition. 30
Figura 3.22 – GUI – CrackGrowth bloqueada. 31
Figura 3.23 – GUI – CrackGrowth livre. 31
Figura 3.24 – Título do Problema (a); Tipo do Problema (b). 32
Figura 3.25 – Módulo V – Graphical Results. 32
Figura 3.26 – GUI – GraphicalResults. 33

Figura 3.27 – Diagrama de Classes do programa BemCracker2D (GOMES, 2016). 34
Figura 3.28 – Diagrama de sequência do crescimento da trinca (GOMES, 2016). 35
Figura 4.1 – Ferramenta GUI e suas componentes. (MATLAB, 2015). 36
Figura 4.2 – Representação hierárquica do Módulo I - Geometry. 38
Figura 4.3 – Representação esquemática da função Points. 38
Figura 4.4 – Representação esquemática da função Lines. 39
Figura 4.5 – Representação esquemática da função Arcs. 40
Figura 4.6 – Representação esquemática da função Zonas. 42
Figura 4.7 – Construção da matriz *BZ* e modificação das matrizes *BL* e *BC*. 44
Figura 4.8 – Representação hierárquica do Módulo II – Mesh. 44
Figura 4.9 – Hierarquia do Módulo II da Interface BEMLAB2D. 48
Figura 4.10 – Esquema de modelagem de trinca com elementos de contorno quadráticos descontínuos. 49
Figura 4.11 – Eixos horizontais conforme o nível de refinamento. 51
Figura 4.12 – Malha gerada com os nós da região interna e os vértices do domínio. 52
Figura 4.13 – Resultado Final de uma Malha de Elementos Finitos antes da Suavização. 53
Figura 4.14 – Resultado Final de uma Malha de Elementos Finitos após Suavização. 53
Figura 4.15 – Representação hierárquica do Módulo III – Boundary Conditions. 56
Figura 4.16 – Representação hierárquica do Módulo IV – Elastostatic Analysis. 58
Figura 4.17 – Representação hierárquica do Módulo V – Graphical Results. 63
Figura 5.1 – BEMLAB2D – Módulos I, II, III e IV. 64
Figura 5.2 – Modelo geométrico para o exemplo 1. 65
Figura 5.3 – Malha de MEC para o exemplo 1. 65
Figura 5.4 – Modelo físico-geométrico sobre a malha de MEC do exemplo 1. 66
Figura 5.5 – Modelo geométrico para o exemplo 2. 66
Figura 5.6 – Malha de MEC para o exemplo 2. 67
Figura 5.7 – Modelo físico-geométrico sobre a malha de MEC do exemplo 2. 67
Figura 5.8 – Modelo geométrico para o exemplo 3. 68
Figura 5.9 – Malha de MEC para o exemplo 3. 68
Figura 5.10 – Modelo físico-geométrico sobre a malha de MEC do exemplo 3. 69
Figura 5.11 – Modelo geométrico para o exemplo 4. 69
Figura 5.12 – Malha de MEC com detalhe da trinca para o exemplo 4. 70
Figura 5.13 – Modelo físico-geométrico sobre a malha de MEC do exemplo 4. 70
Figura 5.14 – Modelo geométrico para o exemplo 5. 71

Figura 5.15 – Malha de MEC com detalhe da trinca para o exemplo 5. 71
Figura 5.16 – Modelo físico-geométrico sobre a malha de MEC do exemplo 5..... 72
Figura 5.17 – Modelo geométrico para o exemplo 6. 72
Figura 5.18 – Malha de MEC com detalhe da trinca para o exemplo 6. 73
Figura 5.19 – Modelo físico-geométrico sobre a malha de MEC do exemplo 6..... 73
Figura 5.20 – Modelo geométrico para o exemplo 7. 74
Figura 5.21 – Malha de MEC com detalhe da trinca para o exemplo 7. 74
Figura 5.22 – Modelo físico-geométrico sobre a malha de MEC do exemplo 7..... 75
Figura 5.23 – BEMLAB2D – Módulos V e interface de pós-processamento..... 75
Figura 5.24 – Malha deformada do exemplo 1. 76
Figura 5.25 – Tensões principais do exemplo 1. 76
Figura 5.26 – Malha deformada do exemplo 2..... 77
Figura 5.27 – Tensões principais do exemplo 2. 77
Figura 5.28 – Malha deformada do exemplo 3..... 78
Figura 5.29 – Malha deformada do exemplo 4..... 79
Figura 5.30 – Malha deformada do exemplo 5. 79
Figura 5.31 – Caminho de propagação da trinca do exemplo 6. 80
Figura 5.32 – Fatores de Intensidade de Tensão do exemplo 6..... 80
Figura 5.33 – Resistência residual normalizada e número de ciclos de carga exemplo 6.. 81
Figura 5.34 – Caminho de propagação da trinca do exemplo 7. 81
Figura 5.35 – Fatores de Intensidade de Tensão do exemplo 7..... 82
Figura 5.36 – Resistência residual normalizada e número de ciclos de carga exemplo 7.. 82
Figura 5.37 – BEMLAB2D – Detalhes do Módulos II. 83
Figura 5.38 – Modelo geométrico da chapa quadrada. 84
Figura 5.39 – Malha de MEC para a chapa quadrada 84
Figura 5.40 – Malha de MEF para a chapa quadrada. 85
Figura 5.41 – Malha de MESHLESS para a chapa quadrada..... 85
Figura 5.42 – Modelo geométrico da seção I. 86
Figura 5.43 – Malha de MEC para a seção I 86
Figura 5.44 – Malha de MEF para a seção I. 87
Figura 5.45 – Malha de MESHLESS para a seção I. 87

LISTA DE TABELAS

Tabela 4.1 – Representação das Matrizes Constitutivas Incompletas. 46
Tabela 4.2 – Representação das Matrizes Constitutivas Completas. 46

LISTA DE SÍMBOLOS E ABREVIAÇÕES

GUI	Interface Gráfica de Usuário
MEF	Método dos Elementos Finitos
MEC	Método dos Elementos de Contorno
MESHLESS	Método Sem Malha
BEMLAB2D	Boundary Element Method Laboratory 2D
CAE	Engenharia Auxiliada por Computador
MATLAB	Matrix Laboratory - Linguagem de Programação Utilizada
MECD	Método dos Elementos de Contorno Dual
BP, BL, BC, BZ	Matrizes de informações de Pontos, Linhas, Arcos e Zonas.
m_inc	Matriz de Coordenadas Nodais
m_iel	Matriz de Topologia da Malha
E	Módulo de Elasticidade Longitudinal
ν	Coeficiente de Poisson

1 - INTRODUÇÃO

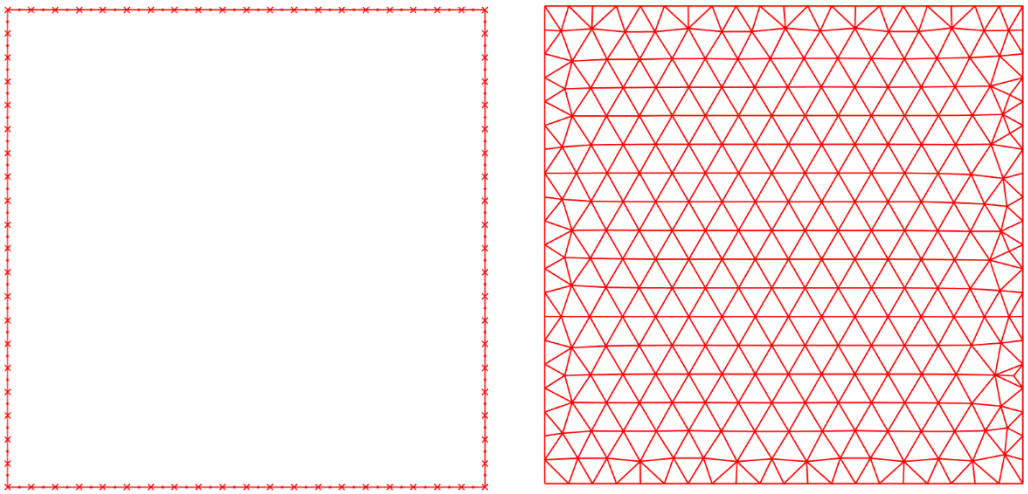
1.1 - GENERALIDADES

Com o desenvolvimento da mecânica computacional e os avanços nos estudos em modelagem e simulações numéricas, impulsionados pelo advento da computação, a utilização de técnicas numéricas em análises de problemas físicos e de engenharia, como o Método dos Elementos Finitos e o Método dos Elementos de Contorno, não representa uma tarefa demasiadamente complexa, uma vez que já estão bem sedimentadas e permitem uma análise eficiente e precisa da maioria dos problemas usuais da engenharia. Por outro lado, pode-se afirmar que os procedimentos mais críticos em uma análise são aqueles relacionados com a forma de representação e manipulação do modelo geométrico genérico, da malha, suas condições de contorno e seus atributos físicos, bem como a alta complexidade no tratamento e gerenciamento dos arquivos de entrada (*neutral file*), resultado da grande quantidade de parâmetros envolvidos na execução da simulação.

Se por um lado o Método dos Elementos Finitos (MEF) (BATHE, 1976) é provavelmente a principal técnica utilizada em análises de engenharia, destacando-se devido a sua grande versatilidade, a sua qualidade de resultados e a sua relativa facilidade de implementação, o Método dos Elementos de Contorno (MEC) (BREBBIA, 1978) é uma alternativa complementar ao MEF, sendo indicado particularmente em casos especiais que requerem melhor interpretação e representação dos dados em problemas com concentração de tensões ou onde o domínio é infinito ou semi-infinito, por exemplo.

Nos sistemas de Engenharia Auxiliados por Computador (*CAE*), em termos gerais, o procedimento de geração dos modelos de MEF consome uma parcela de tempo substancial em relação ao tempo requerido pela análise propriamente dita, enquanto com o MEC (BREBBIA, 1989) a representação da geometria do modelo requer discretização apenas do contorno do mesmo, o que torna a implementação mais fácil e flexível, permitindo maior versatilidade aos programas de modelagem geométrica e de geração de malhas. A Figura 1.1 ilustra a comparação das discretizações da geometria requeridas pelo MEF e pelo MEC em uma aplicação bidimensional modelada com o BEMLAB2D, na qual se observa que as malhas utilizadas pelo MEC permitem a geração e manipulação da geometria, das condições de contorno e da discretização dos modelos sob análise de forma mais simplificada e direta, permitindo-se concluir que a modelagem utilizada pelo MEC pode

facilmente ser implementada em sistemas CAE, pois esta requer apenas a representação do contorno dos modelos.



a) MEC b) MEF

Figura 1.1 – Modelo de malha gerado pelo BEMLAB2D: a) MEC; b) MEF

Apesar dos inúmeros programas desenvolvidos a nível acadêmico no que se diz respeito à modelagem numérica de problemas de engenharia baseada no MEC, a grande parte se vê limitado pelo seu uso devido à falta de uma interface gráfica que facilite a utilização. Comumente, a entrada de dados nesses programas é realizada com arquivos de textos extensos, com um padrão definido de sequência, sendo os resultados igualmente apresentados em arquivos de textos, sem o uso de qualquer artifício de representações gráficas dos resultados.

Neste trabalho é proposto o desenvolvimento de uma interface gráfica, denominada BEMLAB2D, para tratar da questão da modelagem e manipulação de modelos bidimensionais de elementos de contorno, permitindo que informações geométricas, condições de contorno e atributos físicos possam ser gerenciados de forma eficiente e amigável evitando, assim, a tediosa tarefa do uso de arquivos de dados. O programa BEMLAB2D é totalmente implementado no ambiente de *software Matrix Laboratory* (MATLAB) do *MathWorks* sob o sistema operacional Windows e utiliza-se do módulo GUIA (Ambiente Gráfico de Desenvolvimento de Interface de Usuário) que foi usado para criar uma Interface Gráfica de Usuário (GUI) amigável. O uso do BEMLAB2D é dirigido aos alunos e professores do Programa de Pós-Graduação em Estruturas e Construção Civil – PECC, bem como estudantes de outros programas.

1.2 - MOTIVAÇÃO E OBJETIVOS

Devido ao grande avanço tecnológico na área da computação – maior capacidade de armazenamento de dados e velocidade de processamento, cada vez mais o uso desses equipamentos vem sendo solicitado no campo da engenharia. Com esse avanço tecnológico faz também com que surja a necessidade de novos programas computacionais mais eficientes, capazes de modelar estruturas cada vez mais complexas e de reproduzir resultados claros e mais precisos.

Das pesquisas desenvolvidas pelo Programa de Pós-Graduação em Estruturas e Construção Civil da Universidade de Brasília, na qual esta dissertação está inserida, vê-se um crescimento do número de trabalhos ligados à Teoria das Estruturas, principalmente com MEF, fazendo uso de pacotes computacionais fechados (ANSYS, ABAQUS) e interfaces gráficas pouco amigáveis, existindo, portanto, pouco interesse nas atividades relacionadas à implementação computacional e no desenvolvimento de interfaces gráficas de pré e pós-processamento. Além disso, em análise envolvendo o MEC, tem-se sempre que recorrer a programas acadêmicos específicos, geralmente escritos em FORTRAN, que requerem tediosa tarefa de escrever extensos arquivos de dados de entrada.

Procurando dar ênfase a outras técnicas numéricas e buscando suprir uma necessidade acadêmica, este trabalho tem como objetivo principal desenvolver um programa robusto de interface gráfica para pré- e pós-processamentos capaz de representar modelos bidimensionais de elementos de contorno com as seguintes características:

- Desenho do modelo 2D; (1)
- Geração da malha (MEC/MEF) e *MESHLESS*; (2)
- Atributos físicos e condições de contorno para MEC; (3)
- Geração automática de arquivos de dados para analise com MEC, de informações de malha (coordenadas e topologia) para MEF e de coordenadas para métodos sem malha; (4)
- Visualização de resultados diversos extraídos da analise com MEC; (5)
- Interface com o programa BemCracker2D (GOMES, 2016) para análise via MEC. (5)

Essas características podem ser visualizadas na Figura 1.2.

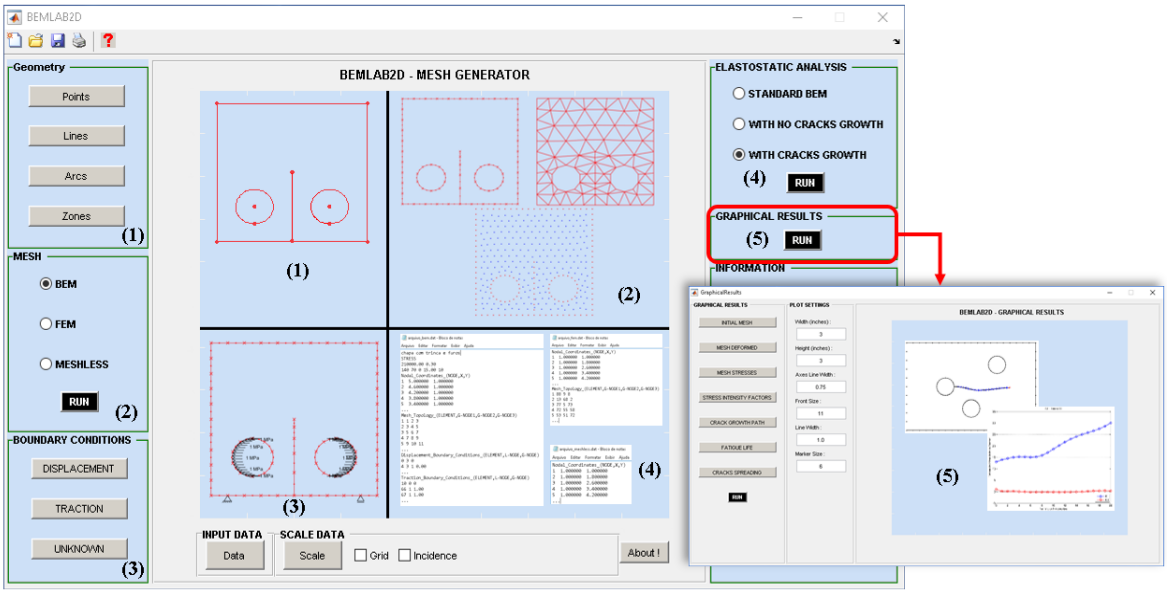


Figura 1.2 – Características principais do BEMLAB2D

Como objetivos secundários, podemos especificar:

- Geração da malha de elementos finitos, a partir da malha de MEC, e armazenamento de suas informações (número de elementos, coordenadas e topologia);
- Geração de modelos através de pontos para análise via métodos sem malha, a partir da malha de MEC, e armazenamento de suas informações (número de pontos e suas coordenadas);
- Geração de arquivos de dados de saída (*fileout*) para a malha de MEF e *Meshless*;
- Adequação e implementação do BEMLAB2D ao programa BemCracker2D (Gomes, 2016) para analise via MEC.

1.3 - ORGANIZAÇÃO DA DISSERTAÇÃO

A construção de uma interface gráfica, considerando-se sua implementação em linguagem MATLAB com função de pré- e pós-processamento gráficos para modelos bidimensionais, e capaz de interagir com um programa de análises pelo Método dos Elementos de Contorno Elastostático, é, inicialmente, apresentada ao longo de cinco capítulos, dos quais esta introdução é o primeiro deles.

No segundo capítulo, será realizada um breve revisão bibliográfica abordados os principais trabalhos sobre programas de modelagem e de geração de malha. Uma breve abordagem sobre o MATLAB e construções de Interfaces Gráficas de Usuários (*GUI*) são apresentadas logo em seguida. Também será apresentado os conceitos envolvidos com a técnica numérica do Método dos Elementos de Contorno (MEC), iniciando-se dos conceitos de modelagem e principais trabalhos, da formulação da equação integral de contorno, da discretização numérica e dos deslocamentos e tensões em pontos internos. Por fim, uma apresentação sobre o MEC Dual para modelagem de trinca.

Os conceitos básicos de modelagem numérica é apresentado no terceiro capítulo o modelo proposto para o BEMLAB2D. Descrevendo cada módulo que o compõe explicando gradativamente sua construção e lógica de modelagem. São apresentados os ícones e suas funcionalidades, assim como as interfaces gráficas auxiliares que são solicitadas pelo BEMLAB2D.

No quarto capítulo, é apresentado a implementação computacional de cada etapa da interface gráfica BEMLAB2D descrita no terceiro capítulo. Os algoritmos, funções e comandos principais que compões esta interface são apresentados e explicados detalhadamente, bem como a construção dos módulos internos da interface e das interações com as interfaces auxiliares e o programa de análise.

No quinto capítulo apresentamos os exemplos numéricos, abordando sete exemplos extraídos da literatura aberta. Serão apresentados os modelos geométricos, as malhas, os modelos físico-geométricos, os arquivos textos gerados e os resultados extraídos do programa de análise plotados na interface auxiliar. Os resultados de pós-processamento tais como deformadas, tensões principais, caminho de propagação de trinca, gráficos de fatores de intensidade de tensão, entre outros.

As conclusões deste trabalho, bem como as sugestões de continuidade do mesmo, finalizam a parte escrita da tese no capítulo seis.

2 - REVISÃO BIBLIOGRÁFICA

2.1 - GENERALIDADES

Os métodos numéricos são ferramentas que apresentam versatilidade e capacidade de resolver problemas, cada vez mais complexos, que surgem no contexto da engenharia. Dentre vários métodos podem ser citados o Método dos Elementos Finitos (MEF) (BATHE e WILSON, 1976) e o Método dos Elementos de Contorno (MEC) (BREBBIA, 1978) que atualmente tangem a grande base de estudos de diversos pesquisadores, porém as dificuldades no emprego desses métodos residem na preparação dos bancos de dados de entrada nos diversos programas de cálculos. Essa etapa desgastante que geralmente consome muito tempo e pode proporcionar diversos erros de montagem desses bancos de dados se deve ao extenso número de informações a serem inseridas nos processadores.

Para auxiliar pesquisas e tornar mais didático o uso da programação em meios acadêmicos, pesquisadores e engenheiros têm desenvolvido diversas interfaces gráficas a fim de conectar processadores de diversos tipos de problemas da engenharia ao visual gráfico de entrada de dados e também de saída de suas respostas. Nesse aspecto, pesquisadores tomam mão dessas ferramentas com o intuito de auxílio para obter resultados mais precisos em suas pesquisas, podendo abordar temas desde a área numérica até a experimental. A versatilidade no uso de uma interface gráfica pode ir desde a criação de modelos bidimensionais com geometrias simples a modelos tridimensionais complexos ou, de estudos de peças isostáticas em estado elástico a modelos de fratura em estruturas elastoplásticas, por exemplo.

Os pré-processadores dessas interfaces são normalmente baseados e construídos de acordo com o que será inserido no processador de dados, dando, assim, maior sensibilidade ao usuário dos métodos numéricos empregados no sistema de entrada. Sendo, por exemplo, o processador baseado no MEF, sua base de dados de entrada será formada por um extenso banco de dados informando as coordenadas nodais e topologias da malha retiradas de uma malha de elementos triangulares ou quadrilaterais que preenchem todo o interior do modelo. No caso de um processador baseado no MEC, seu banco de dados é formado por elementos unidimensionais sobre o contorno do modelo, podendo ser elementos constantes, lineares, quadráticos.

Neste capitulo faremos uma breve revisão bibliográfica acerca dos conceitos básicos envolvidos no desenvolvimento da interface gráfica BEMLAB2D, a saber, aqueles relacionados ao processo de modelagem e a técnica numérica empregada, que é utilizada pelo BEMLAB2D para processar análises diversas via MEC.

2.2 - SOBRE PROGRAMAS DE MODELAGEM

Atualmente, vem ocorrendo um avanço na produção de trabalhos de pesquisa na área de computação gráfica, especificamente, na produção de pré- e pós-processamentos. Porém, a maioria dos trabalhos de modelagem na engenharia, conhecidos da literatura, é desenvolvida para modelos em MEF, e poucos são os trabalhos de desenvolvimento de interfaces gráficas que descrevem bancos de dados para representar modelos em MEC. A seguir, uma revisão sobre programas de pré- e pós-processamentos em MEF e MEC são citados para embasar o estudo deste trabalho.

O desenvolvimento dos métodos numéricos nas áreas de processamento e visualização gráfica permitiu que novos paradigmas de representação surgissem. Bons exemplos de plataformas que manipulam estes problemas encontrados na engenharia podem ser encontrados sem nenhuma dificuldade pela internet; como exemplo de software de pré- e pós-processamentos com MEF tem-se o *FRANC-2D* (WAWRZYNEK, 1986) e o *FRANC-3D* (WAWRZYNEK et al., 1989), pela Universidade de Cornell. Ambos fazem análises de fraturas para problemas 2D e 3D. Outro programa a ser citado é o VISUALANL (RODRIGUES e VARUM, 2005) que de forma interativa e intuitiva define um programa estrutural e possibilita a visualização dos resultados como, por exemplo, modos de vibração, diagramas de esforços, deformadas, evoluções no tempo de esforços, deslocamentos e deformações.

Y-GUI (MAHABADI, et al., 2010) foi desenvolvida para suprir a necessidade de um pré-processamento para o um código robusto de pesquisa bidimensional em MEF chamado de Y2D e desenvolvido por Munjiza em 2004. O código já contava com a combinação das técnicas do Método dos Elementos Finitos-Discretos (MEF/MED), assim o trabalho de Mahabadi foi implementar a interface gráfica de auxílio ao processador e também tratar o código do programa Y2D afim de que permitisse o estudo de materiais heterogêneos.

Dentre diversas áreas, vem se destacando trabalhos neurocientificos com enfase na

visualização e obtenção de informações precisas sobre as regiões cerebrais envolvidas em tarefas específicas através de ressonância magnética funcional. Interfaces gráficas como REST-GCA (ZANG, *et al.*, 2012), BOLDSync (JOSHI, *et al.*, 2014) e o trabalho de Kauweloa, *et al.* em 2016 são grandes propostas de GUIs construida na plataforma MATLAB afim de cobrir experimentos na biomedicina e neurociência.

TOUGH2 (PRUESS, *et al.*, 2012) é um módulo poderoso que simula fratura e fluxos de calor e fluido em meios porosos. Sua entrada de dados foi adaptada em dois trabalhos, iMatTOUGH (TRAN, 2016) é uma interface gráfica de pré e pós-processamento que priorizou a entrada de dados de fluxo e temperaturas dependentes do tempo que também suporta a geração de malha automática e suas conexões. IGMESH (HU, *et al.*, 2016), uma ferramenta gráfica de pré e pós-processamento, é adequada para a construção de malhas tridimensional com base da elevação superior e espessura de cada modelo, além de fornecer funções para atribuição do tipo de rochas, condições de contorno, método de interpolação de elevação e espessura, conversão de resultados de simulação e visualização com um software de plotagem.

Outro trabalho de importância acadêmica, com base no MEC, foi produzido em linguagem C++ e com a possibilidade de construção de modelos bidimensionais em sua interface. O *BEMOO_GI* (GOMES, 2015) foi desenvolvido no contexto da Programação Orientada a Objetos (POO), com funções básicas na forma de GUI que permitem a verificação da validade da estrutura de dados proposta.

2.3 - SOBRE GERADOR DE MALHA

Nos últimos anos, físicos, matemáticos e engenheiros têm realizado inúmeros trabalhos relacionados ao desenvolvimento de técnicas automáticas de geração de malhas como meio de reduzir o esforço e o tempo necessário para elaborar os modelos de elementos finitos. Em paralelo com essas técnicas alguns pesquisadores tomam mão da construção de interfaces gráficas em diversos tipos de linguagem de programação, como MATLAB, C++, PYTHON, entre outros, para auxílio da inclusão de dados pelo usuário.

Como exemplo, tem-se o programa GAMAT2 (GUIMARÃES e FEIJÓO) construído em linguagem C que basicamente particiona em sub-regiões a região bidimensional que se deseja triangulizar, assim tem-se um ganho de tempo computacional na geração da malha. Neste trabalho, o contorno pode ser caracterizado por diversos tipos de segmentos (retos ou

curvos) multiplamente conexos, gerando uma série de pontos sobre os segmentos do contorno que auxiliam a construção dos primeiros triângulos a serem gerados na sub-região, sendo denominada de “front inicial” a lista de dados formada por esses pontos. Posteriormente, o programa gera de forma simples e eficiente um conjunto de nós no interior da região chamados de “nós interiores” que, por sua vez, junto com os nós gerados a partir do “front inicial” conectam-se de maneira a gerar triângulos, o mais equilátero possível, evitando a superposição dos elementos.

Atualmente, pesquisadores têm explorado bastante o uso do algoritmo de Delaunay (DELAUNAY, 1934) para a geração de malha (SAKAMOTO, 2007; SHEWCHUK, 2014; WANG, *et al.,* 2015). Este algoritmo melhora as distribuições dos elementos triangulares maximizando o menor ângulo dos triângulos, ou seja, remove os triângulos finos, melhorando, assim, a qualidade da malha onde, consequentemente, será apresentada melhor precisão dos resultados. No trabalho de Sakamoto foi desenvolvido um gerador de malha de elementos finitos baseado no algoritmo de Delaunay onde uma versão acadêmica desse algoritmo também foi desenvolvida para o Ambiente Mathematica.

O estudo da geração de malha com geometrias complexas e irregulares com furos e concavidades pelo método da triangulação de Delaunay foi estudado por Caires e Sousa (2010). Nesse trabalho foram analisados problemas biomecânicos onde imagens médicas digitais (tomografias) eram tratadas e utilizadas na construção dos modelos geométricos que, a partir das informações do contorno, a malha de finitos era formada. O sistema também integra com o software de análise de elementos finitos, completando a análise estrutural.

Almeida (2014) desenvolveu um software gerador de malhas triangulares para análise com o MEF. A construção do modelo se inicializa com a inserção das coordenadas dos vértices, o modelo também pode ser dividido em subdomínios atribuindo vários níveis de refinamento para cada região discretizada. Em seus algoritmos Almeida usa triângulos equiláteros no domínio do modelo geométrico, nós de contorno sobre as arestas da geometria e otimiza os elementos próximo ao contorno aplicando a suavização Laplaciana com avanço de fronteira de três frentes de geração.

2.4 - SOBRE O MATLAB COM GUI

O MATLAB consiste em um programa computacional que possui a sua própria linguagem

de programação. Seu significado é *Matrix Laboratory* (Laboratório de Matrizes), este é dado devido ao programa ser desenvolvido baseado em cálculos com matrizes. O MATLAB foi criado no final dos anos 70, por Cleve Moler, um dos fundadores da *Mathworks*. Atualmente, o programa é utilizado por profissionais de diversas áreas, sua grande utilidade para a resolução de diversos problemas fez com que sua popularidade aumentasse rapidamente pelo mundo.

A programação no MATLAB pode ser desenvolvida de forma rápida, eficiente e de fácil utilização devido a funções predefinidas armazenada na sua ampla biblioteca. Devido a sua fácil manipulação e suas características citadas anteriormente este programa torna mais fácil o desenvolvimento de programas técnicos em comparação com linguagens como Fortran ou C. O MATLAB é um programa computacional otimizado desenvolvido para resolver problemas clássicos da engenharia e cálculos científicos (CHAPMAN, 2006).

Uma *Graphic User Interface* (GUI) é uma exibição gráfica executada pelo MATLAB de grande importância. Ao contrário das programações por código que realizam rotinas de tarefas, o usuário não precisa entender como as tarefas são executadas. Na GUIs existe uma série de componentes que podem ser inseridos para a o usuário manipular, tais como menus, barras de ferramentas, botões, caixas de diálogos entre outros, que facilitam a execução de qualquer tipo de programa de computação. Também podem ler e escrever arquivos de dados, comunicar-se com outras GUIs e exibir resultados de forma simples e de fácil compreensão.

Devido ao grande avanço ocorrido ao longo dos anos, desde a sua criação, o MATLAB vem se tornando uma ferramenta de grande importância nas mãos de engenheiros e cientistas para estudos e elaborações de projetos. Isso se dá devido à versatilidade que este programa exerce ao resolver os diversos problemas técnicos da engenharia de forma prática. No Programa de Pós Graduação em Estruturas e Construção Civil (PECC) da Universidade de Brasília ele vem sendo uma ferramenta de grande importância para disciplinas que tem base de conhecimento nos métodos numéricos, sua utilização proporciona aos alunos maior familiarização e manuseio com as técnicas de programação.

2.5 - SOBRE O MEC CONVENCIONAL

Nos últimos anos, engenheiros tem se familiarizado muito com as técnicas de análises numéricas, as quais se baseiam na solução aproximada de uma equação ou de um conjunto

de equações que descrevem um problema físico. O primeiro método aproximado amplamente conhecido foi o das diferenças finitas que aproxima as equações governantes do problema usando expansões locais para as variáveis, geralmente usando o truncamento por série de Taylor. A técnica pode ser interpretada como um caso especial dos métodos dos resíduos ponderados mais geral.

No início da década de 60, um grupo de pesquisadores trabalhou nas aplicações das equações integrais para resolver problemas de análise de tensão, potencial e outros problemas da engenharia. Porém, a formulação do problema, a dificuldade de definir a função de Green apropriada e o surgimento paralelo do MEF contribuíram para minimizar a importância deste trabalho. Por meados de 1970 o desenvolvimento do MEF foi de grande importância para o começo das formulações de integrais de contorno. A técnica de aproximação utilizando as equações integrais de contorno foi inserida por (BREBBIA, 1978). Depois desse trabalho outras pesquisas foram estendidas para englobar problemas dinâmicos e que envolvem a não-linearidade (BREBBIA e WALKER, 1980).

Basicamente, a ideia do MEC consiste em transformar a equação diferencial parcial, que representa o problema e suas incógnitas em uma equação integral que relaciona apenas os valores de contorno na busca de sua solução numérica. O cálculo dos valores das tensões ou deslocamentos nos pontos internos é feito de forma direta a partir dos dados encontrados primeiramente no contorno do corpo. Uma vez que todas as aproximações numéricas se dão apenas no contorno, a dimensionalidade do problema é reduzida em um, o que permite trabalhar com um sistema de equações bem menor do que aqueles obtidos com métodos diferenciais como MEF.

No estudo do MEC tem-se que o contorno de um corpo é discretizado em segmentos chamados de elementos que variam em muitos tipos, por exemplo: constante, linear, quadráticos ou de ordem maior. Quando o elemento definido no contorno possui apenas um nó, este é localizado no centro do elemento, ou seja, o elemento é dito como elemento constante e suas incógnitas também são constante sobre o elemento, Figura 2.1a. Para elementos de contorno linear tem-se dois nós localizados na extremidade do elemento, sendo suas incógnitas variando linearmente, Figura 2.1b. Para a modelagem de geometrias curvas o mais ideal é o uso dos elementos quadráticos, os quais possuem três nós e tem suas incógnitas variando sobre uma função de segundo grau ou quadrática, Figura 2.1c.

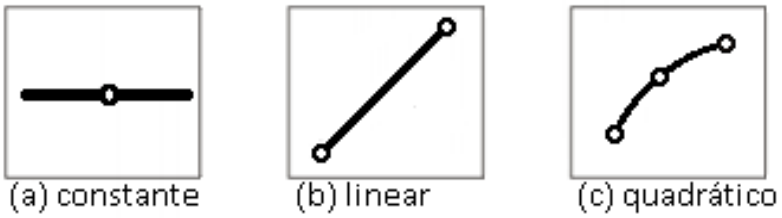


Fig. 2.1 – (a) Elemento constante; (b) elemento linear; (c) elemento quadrático.

2.5.1 - Formulação Integral de Contorno

A formulação direta do Método de Elementos de Contorno (MEC) para elastostática pode ser derivada do teorema do trabalho recíproco de Betti para dois estados autoequilibrados representados por (*ui*, *ti*, *bi*) e (*ui**, *ti**, *bi**); *ui* e *ui** são deslocamentos; *ti* e *ti** são forças de superfície; e *bi* e *bi** são forças de corpo, conforme ilustrado na Figura 2.2.

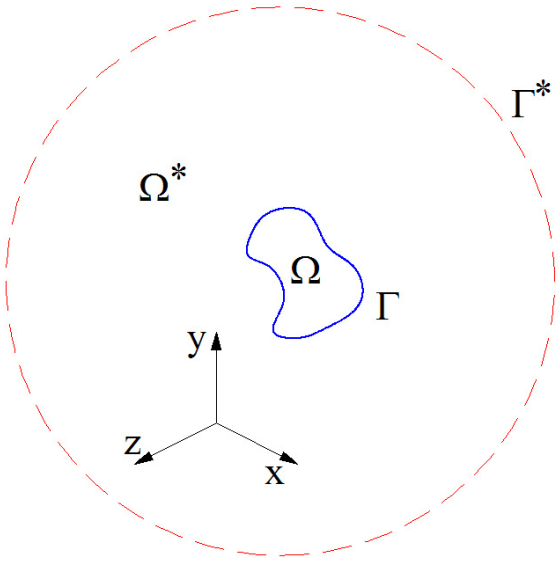


Figura 2.2 – Região arbitrária composta de dois estados autoequilibrados.

A partir das equações de equilíbrio, dadas por,

$$\frac{\partial \sigma_{ij}}{\partial x_i} + b_j = 0 \qquad ou \qquad \sigma_{ij,i} + b_j = 0 \qquad i,j = 1,2 \qquad (2.1)$$

Escreve-se a seguinte relação, em termos de integral, com domínio Ω e contorno Γ,

$$\int_{\Omega} \sigma_{ij,j} u_i^* d\Omega + \int_{\Omega} b_i u_i^* d\Omega = 0 \qquad (2.2)$$

Esta equação, após um processo de derivação da primeira integral e utilização de equações constitutivas, resulta no teorema do trabalho recíproco de Betti, dada por,

$$\int_{\Gamma} t_i u_i^* d\Gamma + \int_{\Omega} b_i u_i^* d\Omega = \int_{\Gamma} t_i^* u_i d\Gamma + \int_{\Omega} b_i^* u_i d\Omega \qquad (2.3)$$

A equação integral de contorno para problemas de elastostática pode ser derivada da equação (2.3) tomando as forças de corpo como sendo forças pontuais num meio infinito, representadas pela função Delta de Dirac da seguinte maneira:

$$b_{i'} = \Delta(\boldsymbol{X}', \boldsymbol{X}) e_i \qquad (2.4)$$

onde o vetor unitário e_i corresponde a um força unitária positiva na direção *i* aplicada no ponto **X'**, denominado ponto fonte. Em problemas bidimensionais, e_i é uma força por unidade de espessura. Os deslocamentos e forças de superfície correspondentes à solução do problema para o caso particular de uma força pontual são escritos como,

$$u_i^* = U_{ij}(\boldsymbol{X}', \boldsymbol{X}) e_j \qquad (2.5)$$

$$t_i^* = T_{ij}(\boldsymbol{X}', \boldsymbol{X}) e_j \qquad (2.6)$$

onde $U_{ij}(\boldsymbol{X}', \boldsymbol{X})$ e $T_{ij}(\boldsymbol{X}', \boldsymbol{X})$ representam, respectivamente, as componentes de deslocamento e forças de superfície na direção *j* no ponto **X** devido a uma força pontual unitária atuando na direção *i* no ponto fonte **X'**. Utilizando essas soluções na equação (2.3) obtém-se,

$$\mathrm{u_i}(\mathbf{X}') = \int_{\Gamma} \mathrm{U_{ij}}(\mathbf{X}', \mathbf{x}) \mathrm{t_j}(\mathbf{x}) \mathrm{d}\Gamma - \int_{\Gamma} \mathrm{T_{ij}}(\mathbf{X}', \mathbf{x}) \mathrm{u_j}(\mathbf{x}) \mathrm{d}\Gamma + \int_{\Omega} \mathrm{U_{ij}}(\mathbf{X}', \mathbf{X}) \mathrm{b_j}(\mathbf{X}) \mathrm{d}\Omega \quad (2.7)$$

onde **X'** e **X** ∈ **Ω** e **x** ∈ **Γ**, e os termos *Uij* e *Tij* são denominados soluções fundamentais. A equação acima é conhecida como identidade de Somigliana para deslocamentos. Ela relaciona o valor dos deslocamentos em um ponto do domínio **Ω** com os valores de forças de superfície e de deslocamentos no contorno **Γ**.

A equação integral de contorno (2.7) é válida para qualquer ponto fonte **X'** localizado

dentro do domínio **Ω** do corpo; porém, para que possa ser utilizada, necessita dos valores dos deslocamentos e forças de superfície nos pontos pertencentes ao contorno, resultando na seguinte equação integral para deslocamentos no contorno:

$$\mathrm{C_{ij}}(\mathbf{x}')\mathrm{u_j}(\mathbf{x}') + \mathrm{VPC}\int_{\Gamma} \mathrm{T_{ij}}(\mathbf{x}',\mathbf{x})\mathrm{u_j}(\mathbf{x})\mathrm{d}\Gamma = \int_{\Gamma} \mathrm{U_{ij}}(\mathbf{x}',\mathbf{x})\mathrm{t_j}(\mathbf{x})\mathrm{d}\Gamma + \int_{\Omega} \mathrm{U_{ij}}(\mathbf{x}',\mathbf{X})\mathrm{b_j}(\mathbf{X})\mathrm{d}\Omega \quad (2.8)$$

2.5.2 - Discretização Numérica

O Método de Elementos de Contorno é um método numérico para solução de equações integrais de contorno partindo de um procedimento de discretização. O primeiro passo nesse procedimento de discretização é dividir, ou aproximar, o contorno **Γ** em *Ne* segmentos ou elementos. Desse modo, na ausência de forças de corpo, para simplificar, a equação (2.8) pode ser reescrita na forma,

$$C_{ij}(\boldsymbol{x}')u_j(\boldsymbol{x}') + \sum_{n=1}^{Ne}\int_{\Gamma_n} T_{ij}(\boldsymbol{x}',\boldsymbol{x})u_j(\boldsymbol{x})d\Gamma_n = \sum_{n=1}^{Ne}\int_{\Gamma_n} U_{ij}(\boldsymbol{x}',\boldsymbol{x})t_j(\boldsymbol{x})d\Gamma_n \qquad (2.9)$$

Considerando os vários tipos de elementos e que a geometria do contorno é dividida em *Ne* elementos retos, os valores das incógnitas e das condições de contorno $u_j(\boldsymbol{x})$ e $t_j(\boldsymbol{x})$ são considerados ao longo do comprimento de cada elemento e iguais aos valores dessas mesmas incógnitas nos pontos centrais (também denominados pontos nodais ou nós funcionais) dos respectivos elementos. Partindo dessas considerações, para elementos constantes, reescreve-se a equação (2.9) como,

$$C_{ij}^c u_j^c + \sum_{n=1}^{Ne} u_j^n \int_{\Gamma_n} T_{ij}^c d\Gamma_n = \sum_{n=1}^{Ne} t_j^n \int_{\Gamma_n} U_{ij}^c d\Gamma_n \qquad (2.10)$$

Onde u_j^n e t_j^n são os valores dos deslocamentos e forças de superfície nos pontos centrais dos elementos de contorno; u_j^c, $C_{ij}^c u_j^c$, T_{ij}^c e U_{ij}^c são, respectivamente, os valores dos deslocamentos, termos livres e soluções fundamentais para o ponto fonte $x' = x_c$. Fazendo os pontos fonte percorrerem, sucessivamente, os pontos centrais dos elementos de contorno, isto é, fazendo *c = 1, 2, ..., Ne*, são geradas *Ne* equações para *Ne* valores de deslocamentos e *Ne* valores de forças de superfície. Aplicando as condições de contorno do problema, ou seja, deslocamentos e forças de superfícies prescritos para os pontos nodais, produz-se um sistema algébrico que, ao ser resolvido, resulta nos valores das incógnitas do

problema.

A equação (2.10), reescrita em forma matricial, produz o sistema algébrico

$$\boldsymbol{H}u = \boldsymbol{G}t \tag{2.11}$$

o qual, quando reordenado através da aplicação das condições de contorno prescritas para o problema, produz o sistema algébrico final

$$\boldsymbol{A}y = \boldsymbol{F} \tag{2.12}$$

onde o vetor **Y** contém todas as incógnitas (deslocamentos ou forças de superfície) do problema; a matriz de coeficientes **A** é formada de colunas de **H** e **G**; o vetor **F** é formado de colunas de **H** e **G** multiplicadas pelas condições de contorno prescritas. Uma vez que esse sistema esteja resolvido, todos os valores das incógnitas de contorno serão conhecidos.

2.5.3 - Deslocamentos e Tensões em Pontos Internos

Os deslocamentos e tensões nos pontos pertencentes ao domínio **Ω** do problema podem, uma vez que os valores das incógnitas de contorno tenham sido obtidos, ser calculados utilizando-se a identidades de Somigliana para deslocamentos, equação (2.7), que para um $\boldsymbol{X}^d \in \boldsymbol{\Omega}$, na ausência de força de corpo, resulta em

$$u^d = \sum_{n=1}^{Ne} \mathbf{G}^{dn}\mathbf{t}^n - \sum_{n=1}^{Ne} \bar{\mathbf{H}}^{dn}\mathbf{u}^n \tag{2.13}$$

A equação (2.13) relaciona os deslocamentos em um ponto qualquer do domínio com os valores dos deslocamentos e forças de superfície de todo o contorno, e pode ser facilmente avaliada dado que as soluções fundamentais não apresentam singularidades. Sendo a identidade de Somigliana para tensões, dada por,

$$\sigma_{ij}(X') = \int_{\Gamma} D_{kij}(X', x)t_k(x)d\Gamma - \int_{\Gamma} S_{kij}(X', x)u_k(x)d\Gamma + \int_{\Omega} D_{kij}(X', X)b_k(X)d\Omega \tag{2.14}$$

Podemos aplicar o mesmo ponto $\boldsymbol{X}^d \in \boldsymbol{\Omega}$ na equação (2.14), na ausência de forças de corpo, e obtermos a seguinte expressão para as tensões internas,

$$\sigma^d = \sum_{n=1}^{Ne} \mathbf{D}^{dn}\mathbf{t}^n - \sum_{n=1}^{Ne} \mathbf{S}^{dn}\mathbf{u}^n \qquad (2.15)$$

Essa equação relaciona as tensões em um ponto qualquer do domínio com os valores dos deslocamentos e forças de superfície de todo o contorno. Como no cálculo dos deslocamentos nos pontos internos, pode ser facilmente avaliada dado que as soluções fundamentais não apresentam singularidades.

2.6 - SOBRE O MEC DUAL PARA MODELAGEM DE TRINCA

O MEC tem se destacado, nas últimas décadas, como uma poderosa ferramenta de análise em problemas de elasticidade, em particular no campo da mecânica da fratura com os trabalhos pioneiros de Ingraffea *et al* (1983), Becker (1986) e Bush (1999), entre outros, que se caracterizam pela aplicação da equação integral de contorno de deslocamento, equação padrão do MEC, diretamente relacionada à Identidade de Somigliana para deslocamentos.

Para problemas envolvendo trincas, a solução não pode ser alcançada com a aplicação direta do MEC padrão, uma vez que as superfícies da trinca são coincidentes, o que levaria a um sistema de equações algébricas singular. Neste caso, os trabalhos de Blandford *et al* (1981) e Portela *et al* (1992) aparecem como os mais gerais. O primeiro usa o método das sub-regiões, onde um contorno artificial se conecta ao contorno da trinca e divide o domínio em duas novas sub-regiões sem trincas, enquanto os outros dois utilizam tanto a equação integral de contorno de deslocamento – MEC padrão, quanto à equação de tração, conhecida como equação hipersingular, para representar o contorno da trinca, sendo aplicadas de forma independentes em cada face da trinca.

Tendo em vista que o BEMLAB2D é uma interface gráfica para modelagem e visualização baseada em modelos bidimensionais de elementos de contorno e, ainda, que possibilita a análise de problemas elastostáticos envolvendo modelos de trincas, por meio do programa BemCracker2D (GOMES, 2016), este trabalho empregará em seu processo de modelagem tanto o MEC padrão, para discretização do contorno usando elementos contínuos, quanto o MEC Dual (PORTELA et al., 1992), para discretização da trinca usando elementos descontínuos. A Figura 2.3 apresenta a estratégia de modelagem aplicada pelo BEMLAB2D em um problema envolvendo trincas, na qual é representada por elementos

quadráticos.

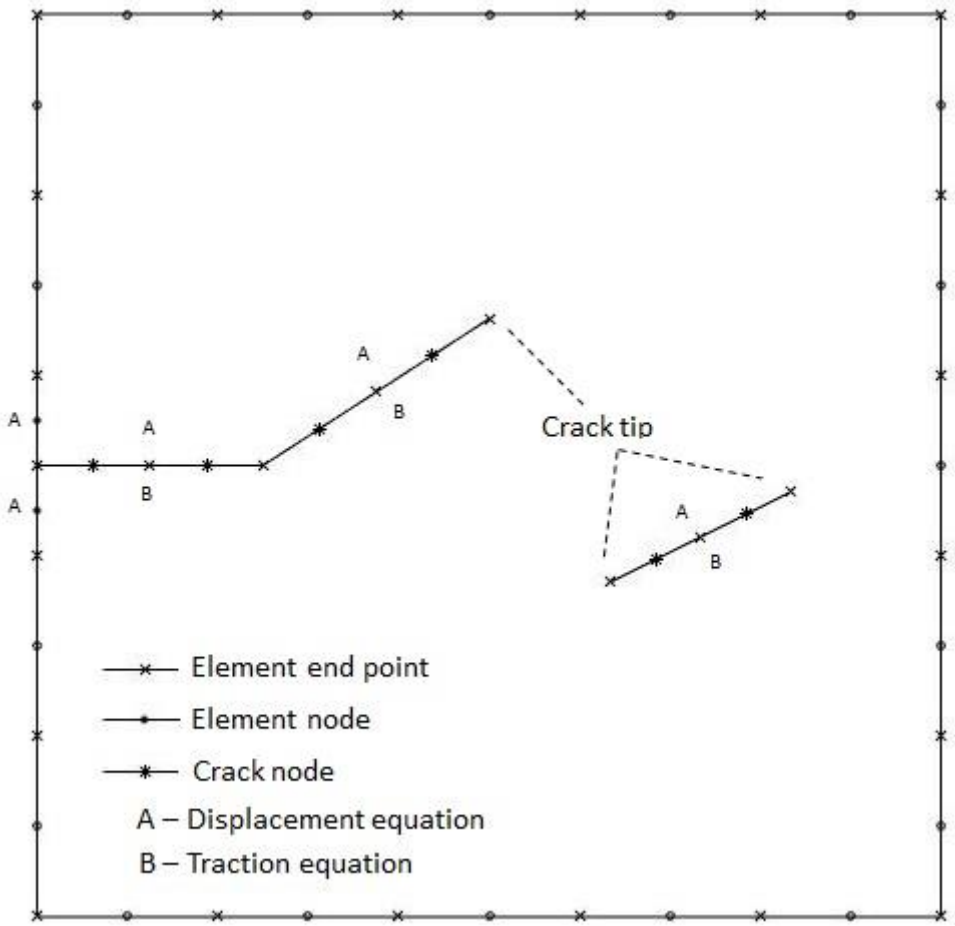


Figura 2.3 – Esquema de modelagem com elementos quadráticos (GOMES, 2016).

3 - MODELO PROPOSTO PARA O BEMLAB2D

Este capítulo abordará uma breve apresentação do programa BEMLAB2D desenvolvido na plataforma MATLAB. O propósito, a funcionalidade entre outras características do programa serão demonstrados junto com as hierarquias de funcionalidade. Uma explicação das etapas de comando visualizada e manuseada por um usuário é abordada nesse capítulo. Também será comentado sobre o módulo de análise adotado.

3.1 - INTRODUÇÃO

A ideia do desenvolvimento do BEMLAB2D surgiu da necessidade de programas de pré- e pós-processamento para suprir o desenvolvimento de forma prática dos dados de entrada e leitores de dados de saída ligando a interface gráfica aos *Solvers* desenvolvidos academicamente que fazem análises a partir do Método dos Elementos de Contorno (MEC).

O desenvolvimento do BEMLAB2D foi tido como grande ferramenta operacional que auxiliará tanto no desenvolvimento de modelos complexos de construir, como no auxílio acadêmico ao entendimento de alunos sobre o MEC. Um dos objetivos do desenvolvimento do BEMLAB2D fora exatamente a necessidade que professores e alunos tinham para construir e visualizar modelos gráficos onde, anteriormente, a base de dados era toda feita a partir de planilhas ou arquivos textos extensos. Com o BEMLAB2D, uma fácil ferramenta de manuseio, alunos e professores poderão exportar arquivos de dados de entrada, além de visualizar e exportar imagens dos modelos e diferentes malhas, além da malha de MEC.

Como dito anteriormente uma das grandes funções do BEMLAB2D é a modelagem completa, possibilitando a visualização e exportação das imagens dos modelos e das malhas, que podem ser de elementos de contorno, elementos finitos e métodos sem malha. Sua programação gera de forma automática arquivos textos completos alimentandos com informações como, coordenadas nodais, topologia de malha, restrições e forças. Além de uma boa qualidade de pré-processamento, o BEMLAB2D também faz a leitura dos arquivos de saída do BemCracker2D (GOMES, 2016) e os imprimem na interface em forma de gráficos e modelos de resultados.

O BEMLAB2D aborda uma construção diferente, porém simples, dos seus modelos,

tornando seu uso de grande eficácia na construção de modelos bidimensionais. Mais detalhes serão apresentados a seguir.

3.2 - MODELO VISUAL DO BEMLAB2D

O BEMLAB2D é um sistema automático completo de análises de engenharia por MEC e composto por três módulos principais: o *pré-processador*, o *processador* e o *pós-processador*. O pré- e pós-processamento formam a interface gráfica entre o sistema e o usuário através de um processo interativo que permite ao usuário controlar as partes, estruturas e aparências dos objetos. O módulo de pré-processamento é responsável pela definição do modelo geométrico do problema, pela associação de atributos físicos à geometria e pela geração da malha de elementos de contorno, elementos finitos e métodos sem malha. As informações da malha e dos atributos são passadas ao processador ou (programa de análises) através de um arquivo de dados gerados na fase de pré-processamento, o qual só faz análise por MEC. O módulo de pós-processamento trata de interpretar os dados fornecidos através de um arquivo de saída gerado pelo processador na análise numérica. A Figura 3.1 ilustra a arquitetura de um sistema completo.

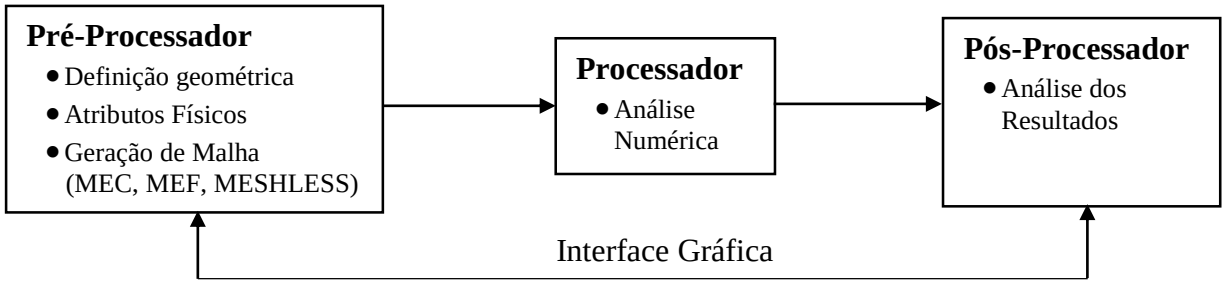


Fig. 3.1 – Arquitetura de um sistema de análises de engenharia (GOMES, 2006)

Este trabalho se concentrará no primeiro e terceiro módulo, isto é, no desenvolvimento de uma interface gráfica para pré- e pós-processamento, cujo desenvolvimento envolveu duas etapas: o projeto visual e a escrita do código. No projeto visual, a aparência do programa foi tratada de forma rápida e direta, utilizando as ferramentas gráficas do MATLAB, como o editor de recursos que permite a construção de objetos como menus e botões de ferramentas para compor a janela principal do programa, sem implementação de nenhum código. Na segunda etapa, o código é escrito usando os métodos, ou *callbacks*, gerados quando da construção dos objetos pelo MATLAB.

Neste aspecto, a interface BEMLAB2D é baseada em ações definidas pelo usuário por

meio das ferramentas botões, mouse e diálogos, cujos principais módulos e características são descritos a seguir.

- GEOMETRY (Módulo I): Este módulo é independente e tem a finalidade de construção do modelo 2D por meio das ferramentas de desenho POINTS, LINES, ARCS e ZONES;
- MESH (Módulo II): Este módulo possibilita a geração de malha, a partir da definição do modelo pelo módulo I, para três diferentes tipos, a saber, Método dos Elementos de Contorno (MEC), Método dos Elementos Finitos (MEF) e Método Sem Malha (MESHLESS), entretanto para os dois últimos tipos, a interface se limita à geração, visualização e armazenamento da geometria da malha;
- BOUNDARY CONDITIONS (Módulo III): Este módulo é específico para análise via elementos de contorno, sendo responsável pela execução das condições do contorno, a saber: DISPLACEMENTS (deslocamentos), TRACTIONS (trações) e UNKNOWN (desconhecidas);
- ELASTOSTATIC ANALYSIS (Módulo IV): Outro módulo específico para análise via elementos de contorno, responsável pela escolha do tipo de análise a ser realizada pelo *Solver*, a saber, STANDARD BEM (MEC padrão), WITH NO CRACKS GROWTH (sem propagação de trinca) e WITH CRACKS GROWTH (com propagação de trinca);
- GRAPHICAL RESULTS (Módulo V): Este módulo é específico para análise via elementos de contorno, sendo responsável pela visualização dos seguintes resultados gráficos: MESH DEFORMED (deformada da malha), MESH STRESSES (malha de tensões), STRESS INTENSITY FACTORS (fatores de intensidade de tensão), CRACK GROWTH PATH (caminho de propagação da trinca), FATIGUE LIFE (vida à fadiga) e CRACKS SPREADING (trincas propagando).

Os conceitos e métodos apresentados no capítulo anterior são a base dos algoritmos utilizados para desenvolvimento do BEMLAB2D. Uma visão geral do funcionamento do programa é representado pelo fluxograma na Figura 3.2:

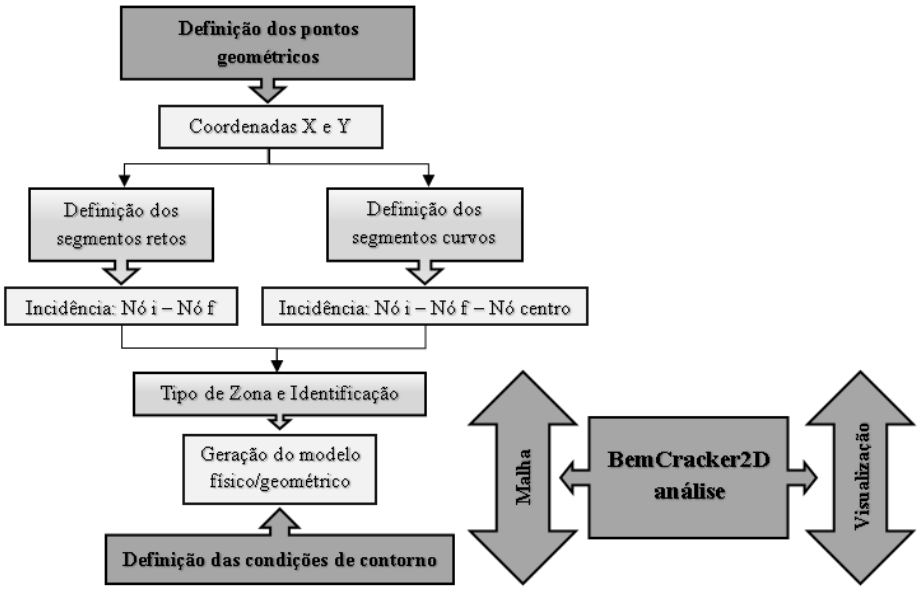


Fig. 3.2 – Hierarquia de funcionalidade da Interface BEMLAB2D

Inicialmente, tem-se uma área de ambiente gráfico com vários botões e menus de comando, onde são inseridos de forma simples os dados de entradas, sendo os mesmos apresentado de forma gráfica numa área de desenho encontrada na própria GUI. Em seguida, implementou-se, no arquivo das linhas de comando do BEMLAB2D, rotinas de comandos que ilustram os modelos físico-geométricos estudados utilizando apenas os dados inseridos inicialmente pelos botões de comando na GUI. Essas informações são organizadas e armazenadas em matrizes para posterior uso no processamento. A implementação computacional e rotinas serão apresentadas de forma mais detalhada no próximo capítulo.

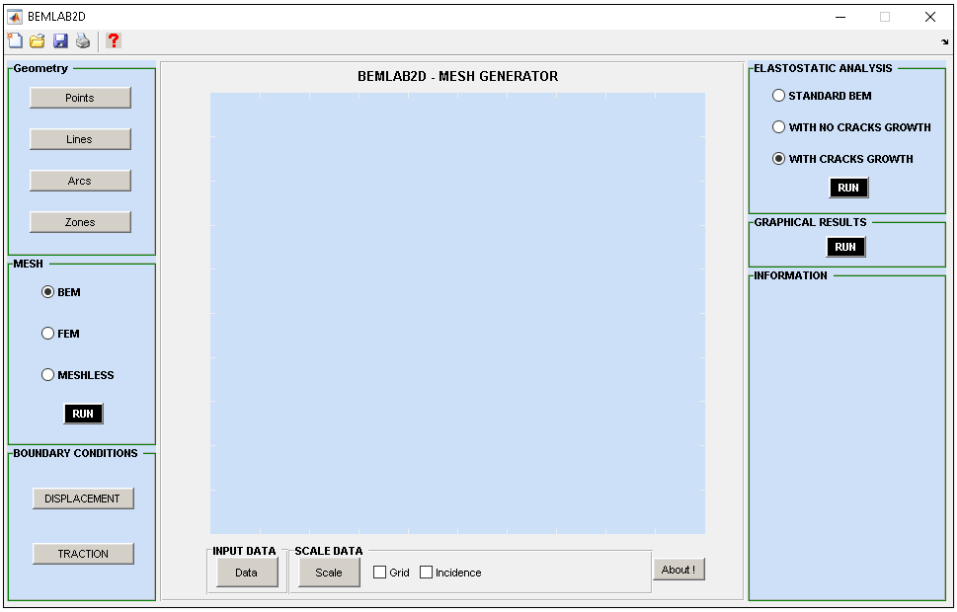


Fig. 3.3 – Ambiente gráfico do programa BEMLAB2D

3.3 - MÓDULOS DO BEMLAB2D

Nos tópicos a seguir serão apresentado cada módulo que compõe o BEMLAB2D. Apresentando de forma sucinta os botões característicos a cada comando e explicando suas funcionalidades. A implementação será discutida no capítulo seguinte.

3.3.1 - Módulo I - Geometry

O módulo I é responsável por gerar os modelos geométricos na área gráfica principal do BEMLAB2D, os ícones desse módulo são todos *PushButton*. Os ícones principais *Points*, *Lines*, *Arcs* e *Zones* tem como prioridade desenhar do modelo geométrico, sendo esses modelos com furos e/ou zonas conexas. Uma apresentação do módulo I pode ser visualizada na Figura 3.4.

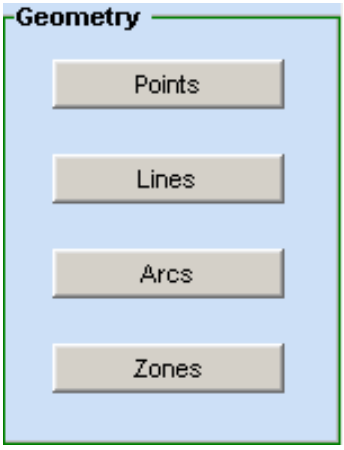


Fig. 3.4 – Módulo I - Geometry

3.3.1.1 - Point

O ícone *Points* derivado do Módulo I que corresponde ao ícone que possibilita a inserção dos pontos geométricos do modelo na interface gráfica. O ícone em questão abre uma caixa de diálogo possibilitando a entrada de dois vetores de coordenadas onde a primeira representa as coordenadas em “X” e a segunda, em “Y”. A Figura 3.5 apresenta a caixa de diálogo descrita anteriormente.

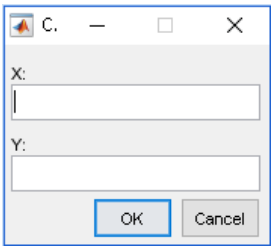


Fig. 3.5 – Caixa de diálogo das coordenadas X e Y

3.3.1.2 - Lines

O ícone *Lines* derivado também do Módulo I, corresponde ao ícone que possibilita inserir segmentos retos. Os segmentos retos são inseridos pelo clique do *mouse* nos pontos que foram inseridos anteriormente pelo ícone *Points*. Para definir um segmento reto o usuário do BEMLAB2D deve clicar no primeiro ponto do segmento e depois no segundo ponto, para desenhar um próximo segmento o procedimento deve ser repetido e finalizado com o botão *Enter* do teclado. A Figura 3.6 ilustra a construção de um segmento reto.

Fig. 3.6 – Desenhando o segmento reto

3.3.1.3 - Arcs

O ícone *Arcs* é muito semelhante ao ícone *Lines*, derivado do Módulo I, possibilita inserir segmentos curvos. Os segmentos curvos também são inseridos pelo clique do *mouse* nos pontos que foram inseridos com o ícone *Points*. Na construção do segmento curvo faz-se necessário clicar em três pontos, estes são o ponto inicial, o ponto final e o centro, bem como a definição do sentido do segmento. A Figura 3.7 ilustra a construção de um segmento curvo e a Figura 3.8 apresenta a caixa lista que define o sentido de desenho do segmento.

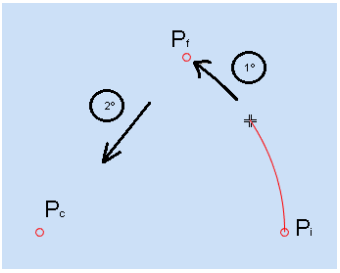


Fig. 3.7 – Desenhando o segmento curvo

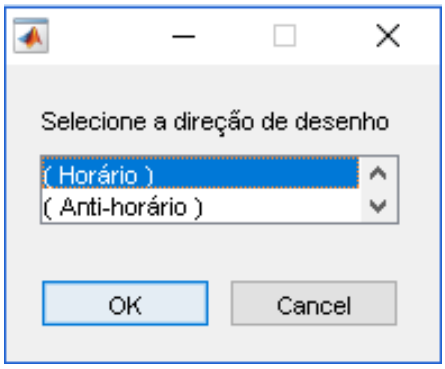


Fig. 3.8 – Direcionamento do desenho do arco

3.3.1.4 - Zones

O ícone *Zones* é na sequência o último ícone derivado do Módulo I, corresponde ao ícone que designa parâmetros sobre as diversas zonas que um objeto ou modelo geométrico possa ter. No BEMLAB2D, quando selecionado o botão *Zones*, uma interface auxiliar chamada Zona se projeta onde são inseridos parâmetros como o módulo de elasticidade e o coeficiente de Poisson para zona Mestre e Inclusões, ver Figura 3.9.

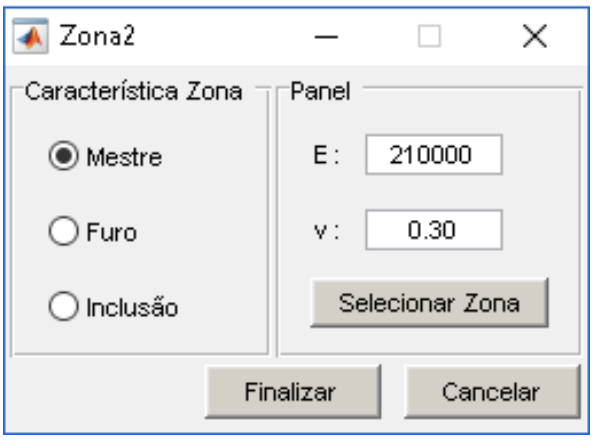


Fig. 3.9 – Interface gráfica que configura as zonas

Após definir os parâmetros e o tipo de zona, ao clicar no botão “Selecionar Zona” o cursor do mouse é habilitado. Os segmentos retos e curvos de uma zona são selecionados com o clique do mouse, assim o usuário pode definir uma zona Mestre, ou uma zona de Inclusão, ou um Furo. A Figura 3.10 ilustra de forma representativa as zonas Mestre, Inclusão e Furo que um modelo pode ter.

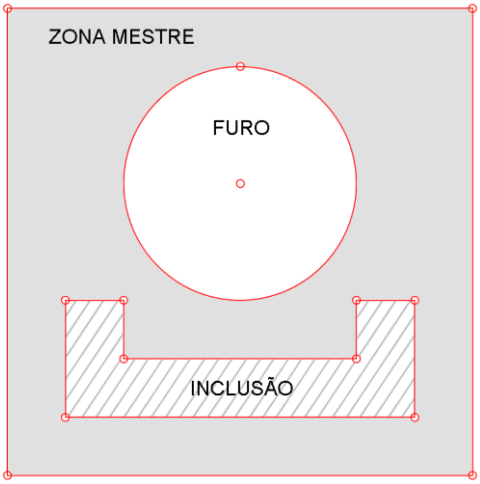


Fig. 3.10 – Representação das zonas Mestre, Furo e Inclusão

3.3.2 - Módulo II - Mesh

O módulo II é responsável por gerar a malha característica do modelo geométrico construído pelo Módulo I, a qual também é representada na área gráfica principal do BEMLAB2D. Na interface o Módulo II está em forma de *PushButton* e *RatioButtons*, sendo de grande importância na construção da malha do Método dos Elementos de Contorno (MEC), a qual é utilizada para a construção do modelo físico-geométrico e para a análise via MEC. O módulo II ainda realiza a construção da malha do Método dos Elementos Finitos (MEF) e do Método Sem Malha (MESHLESS) através da malha de MEC como um complemento ao software BEMLAB2D. Uma apresentação do módulo II pode ser visualizada na Figura 3.11 e os três tipos de malhas são apresentados na Figura 3.12.

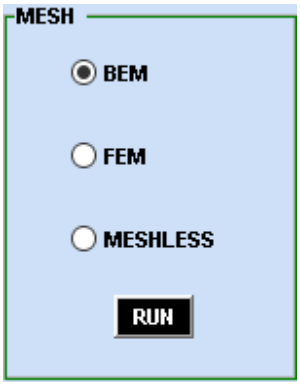


Fig. 3.11 – Módulo II – Mesh

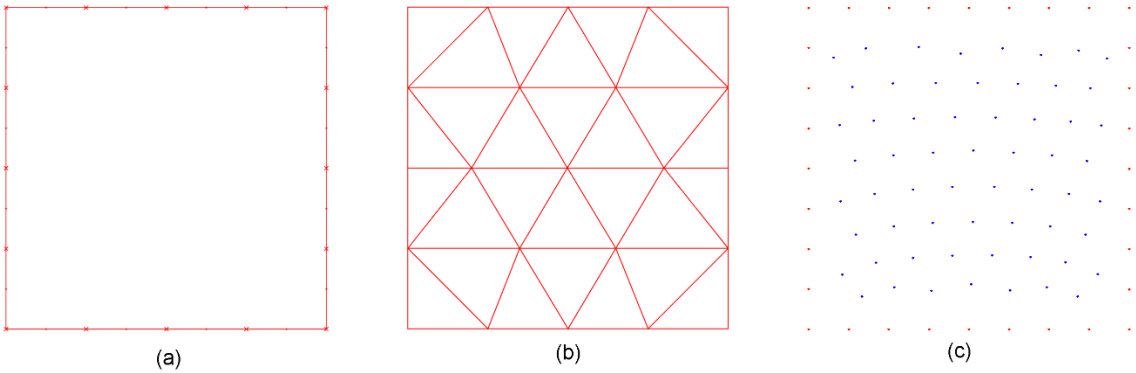


Fig. 3.12 – (a) Malha MEC, (b) Malha MEF e (c) Malha MESHLESS

3.3.2.1 - Malha de MEC

A característica principal do Módulo II é a capacidade de reproduzir as malhas de MEF e MESHLESS a partir das características da malha gerada pela malha de MEC. Sendo assim, as informações pertinentes à malha de MEC sempre serão solicitadas antes de plotar qualquer que seja a malha na área gráfica do BEMLAB2D.

Ao selecionar o botão *Run* com o devido *RatioButtons* selecionado em *BEM*, o cursor do *mouse* se ativa para selecionar um segmento. Sendo um segmento reto selecionado uma interface gráfica auxiliar é acionada de forma a definir características pertinentes ao segmento reto, como continuidade, quantidade de elementos e se é um segmento de trinca, ver Figura 3.13.

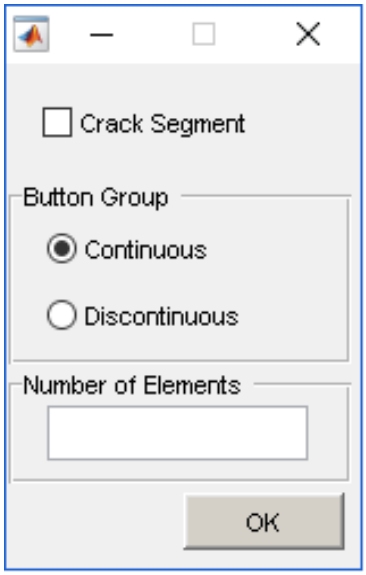


Fig. 3.13 – Interface gráfica que configura o segmento reto

Sendo um segmento curvo selecionado, uma caixa de diálogo é acionada de forma a definir a quantidade de elementos, ver Figura 3.14.

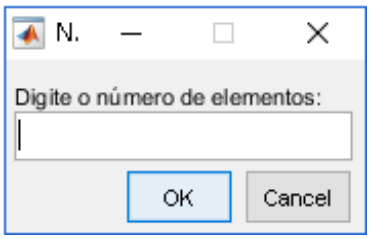


Fig. 3.14 – Caixa de diálogo – Número de elementos no segmento curvo

Ao finalizar a seleção de todos os segmentos, o BEMLAB2D solicita através de uma caixa de questionamento se o usuário definirá pontos internos na sua malha de MEC, caso tenha pontos internos, estes são inseridos por uma caixa de diálogo semelhante a caixa de diálogos aberta no ícone *Points* no módulo I (ver Figura 3.5). A Figura 3.15 ilustra a caixa de questionamento apresentada ao usuário para definir se há ou não pontos internos.

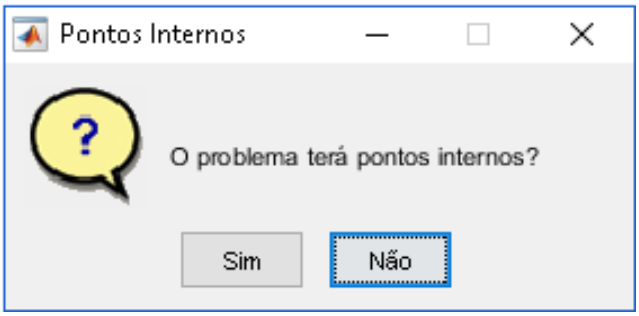


Fig. 3.15 – Caixa de questionamento – Pontos internos

3.3.2.2 - Malha de MEF

Sendo o *RatioButtons* selecionado em *FEM*, a configuração dos elementos do contorno será realizada inicialmente como descrito no subitem anterior, isto é, com aproveitamento dos pontos de contorno da malha de MEC. Apenas haverá mudanças na configuração dos segmentos descontínuos, como nos segmentos de trincas, por exemplo. Nestes segmentos descontínuos serão definidos uma nova quantidade de elementos contínuos, ver Figura 3.16. Além disso, para malha de MEF será configurado a quantidade de eixos de refinamento da malha a partir de uma caixa de diálogo, ver Figura 3.17.

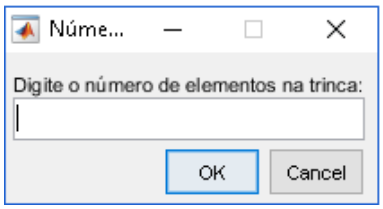


Fig. 3.16 – Número de elementos da trinca

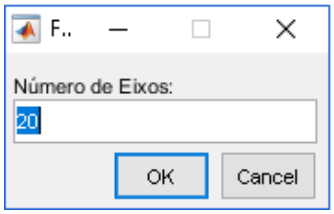


Fig. 3.17 – Quantidade de eixos conforme o nível de refinamento

3.3.2.3 - Malha de MESHLESS

Sendo o *RatioButtons* selecionado em *MESHLESS*, a configuração dos elementos será realizada identicamente descrito no subitem anterior, as mesmas caixas de diálogos da Figura 3.16 e Figura 3.17 serão abertas, a mudança ocorre na implementação que será descrita no próximo capítulo.

3.3.3 - Módulo III – Boundary Conditions

O módulo III é responsável por inserir condições de contorno, como restrições, deslocamentos prescritos e condições de cargas (força ou tensões). Os ícones desse módulo são todos *PushButton* e as informações são todas plotadas na área gráfica principal do BEMLAB2D construindo, assim, o modelo físico-geométrico sobre a malha de MEC. Os ícones principais *DISPLACEMENT* e *TRACTION* tem como prioridade desenhar do modelo físico-geométrico a partir da malha de MEC. Uma apresentação do módulo III pode ser visualizada na Figura 3.18.

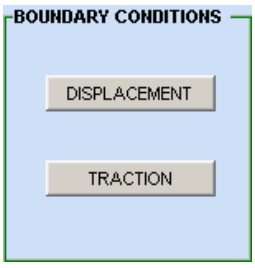


Fig. 3.18 – Módulo III – Boundary Condition

3.3.3.1 - Displacement

O ícone *DISPLACEMENT* encontrado no Módulo III corresponde ao ícone que possibilita a inserção das restrições de deslocamentos como apoios e deslocamentos prescritos do modelo físico-geométrico. Este ícone abre uma interface gráfica auxiliar onde são inseridas as informações pertinentes ao elemento restrito, se sua restrição é em um nó ou em todos e direção da restrição acompanhada do valor do deslocamento prescrito. A interface gráfica auxiliar é apresentada na Figura 3.19 a seguir.

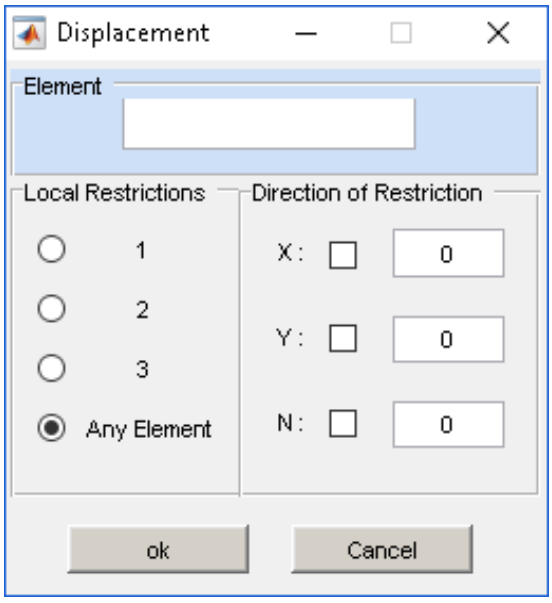


Fig. 3.19 – GUI – Displacement

3.3.3.2 - Traction

O ícone *TRACTION* encontrado no Módulo III corresponde ao ícone que possibilita a inserção das condições de cargas (força e tensão) do modelo físico-geométrico. Este ícone abre uma interface gráfica auxiliar parecida com a do subitem anterior, onde também são inseridas as informações pertinentes ao elemento solicitado, se a solicitação é nodal (força) ou em todo o elemento (tensão) e direção da solicitação acompanhada do valor da mesma. A interface gráfica auxiliar é apresentada na Figura 3.20 a seguir.

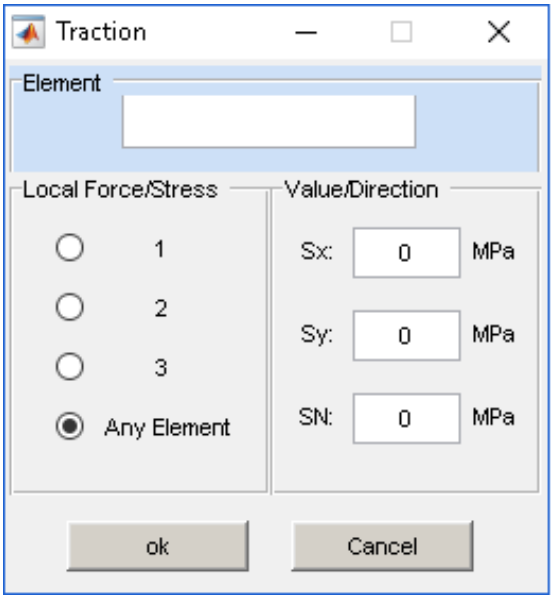


Fig. 3.20 – GUI – Traction

3.3.4 - Módulo IV – Elastostatic Analysis

O módulo IV é responsável por informar qual tipo de análise será realizada pelo BemCracker2D (GOMES, 2016). Os ícones desse módulo são *PushButton* e *RatioButtons*. Ao selecionar o botão *Run*, com o devido *RatioButtons* selecionado, uma interface gráfica auxiliar é acionada, nesta são dadas diversas informações que só podem ser modificadas em determinado tipo de análise. Uma apresentação do módulo IV pode ser visualizada na Figura 3.21.

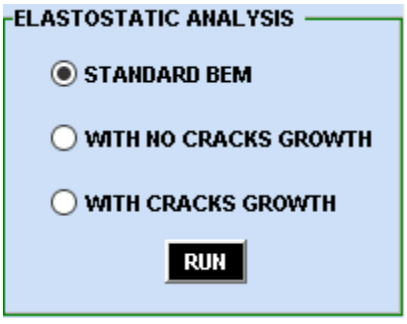


Fig. 3.21 – Módulo IV – Boundary Condition

Ao selecionar o botão *Run* com o devido *RatioButtons* selecionado na opção *STANDARD BEM* ou *WITH NO CRACKS GROWTH*, a interface gráfica auxiliar citada anteriormente é acionada, nesta todas as suas configurações são boqueadas com exceção do número de pontos de Gauss, ver Figura 3.22.

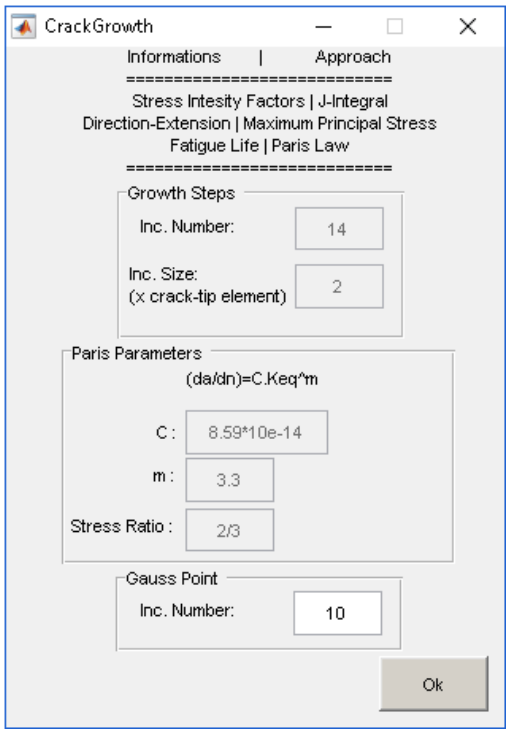


Fig. 3.22 – GUI – CrackGrowth bloqueada

Sendo *RatioButtons* selecionado na opção *WITH CRACKS GROWTH*, a interface gráfica auxiliar citada anteriormente é acionada, neste caso todas as suas configurações são livres para alteração, ver Figura 3.23.

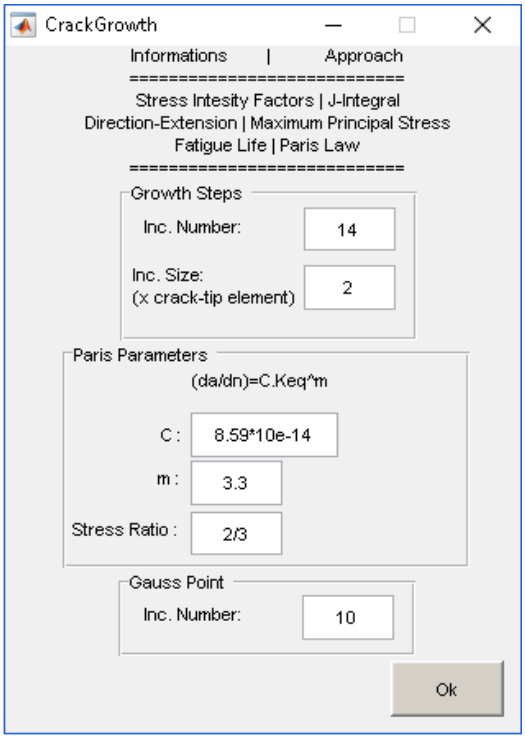


Fig. 3.23 – GUI – CrackGrowth livre

Se não for definido no início da modelagem do programa as informações do “Título do Problema” e o “Tipo do Problema”, neste momento a caixas de diálogos são ativadas de forma automática para que o usuário entre com essas informações. Ver Figura 3.24

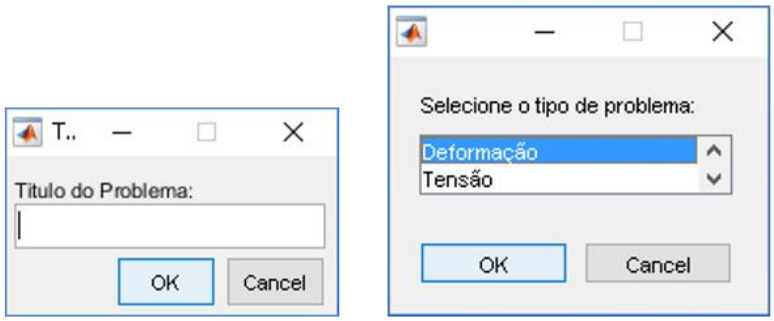


Fig. 3.24 – Título do Problema (a); Tipo do Problema (b)

3.3.5 - Módulo V – Graphical Results

O módulo V é responsável pela leitura e visualização dos arquivos de dados calculados pelo BemCracker2D. O único ícone desse módulo, um *PushButton*, aciona uma interface gráfica auxiliar composta por vários outros *PushButton* que definem a visualização dos resultados gráficos em uma área gráfica na própria interface gráfica auxiliar denominada de GraphicalResults. Uma apresentação do módulo V pode ser visualizada na Figura 3.25.

Fig. 3.25 – Módulo V – Graphical Results

A interface gráfica auxiliar GraphicalResults apresenta resultados gráficos como a malha deformada do modelo (*Mesh Deformed*), malha das tensões (*Mesh Stresses*), gráfico dos fatores de intensidade de tensão (*Stress Intensity Factors*), propagação da trinca (*Cracks Spreading*), vida à fadiga (*Fatigue Life*), entre outros, Figura 3.26.

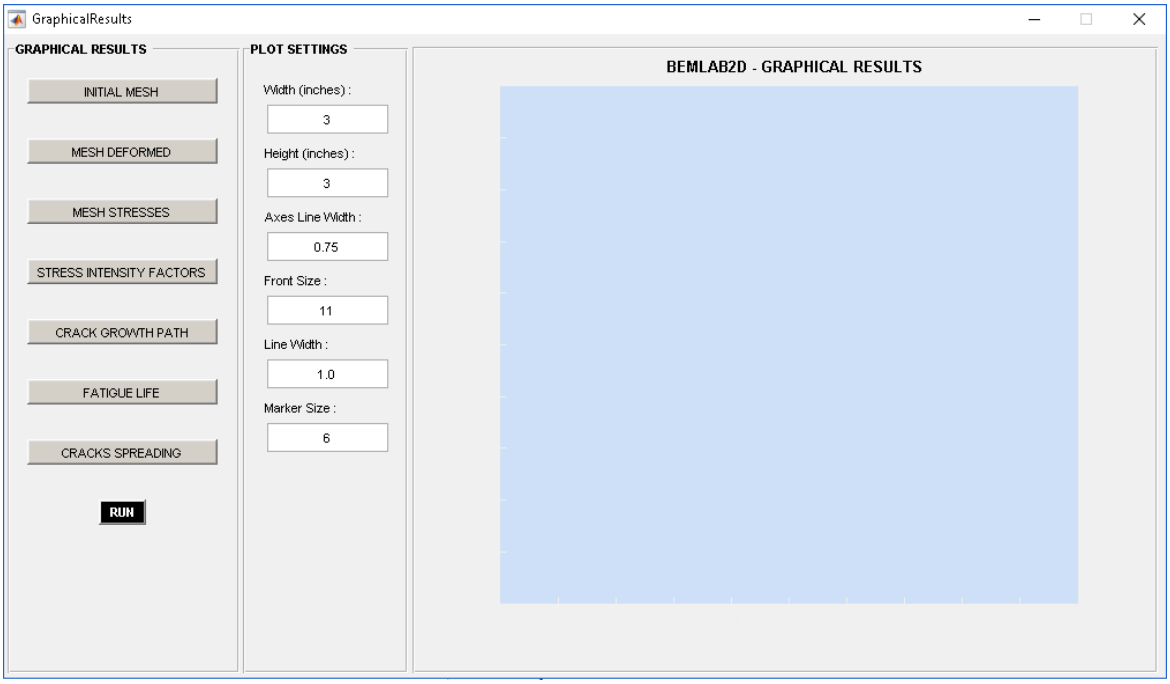


Fig. 3.26 – GUI – GraphicalResults

Na interface ainda existe áreas de edição dos parâmetros de plotagem que exporta em imagem o resultado gráfico selecionado. Essas exibições gráficas são devido ao banco de dados gerado pelo programa que auxilia o BEMLAB2D resolvendo os modelos numericamente através do MEC, o programa BemCracker2D, citado anteriormente, é responsável por toda a análise numérica tida neste trabalho. Este trabalho se resume apenas na elaboração da interface de pré- e pós-processamento de dados no BEMLAB2D, o BemCracker2D é um trabalho em paralelo desenvolvido por Gomes (2016) e que terá uma breve abordagem no próximo subitem.

3.4 - MÓDULOS DE ANÁLISE

Como descrito do subitem anterior, o programa de análise utilizado em conjunto com o BEMLAB2D será o BemCracker2D, este programa faz análises apenas através do MEC, ou seja, o BEMLAB2D tratará apenas modelos e resultados através do MEC também. Contudo o BEMLAB2D também é responsável por gerar malhas de MEF e MESHLESS e construir arquivos de dados, a análise por meio desses dois métodos não foi implementada neste trabalho, assim os arquivos gerados pelo BEMLAB2D servem para suporte de entrada a outros programas de análises que não estão presente neste trabalho.

O BemCracker2D é um programa escrito em linguagem C++ e todo estruturado nos

conceitos da Programação Orientada à Objeto (POO) com objetivo de realizar análises de problemas elastostáticos bidimensional, via MEC.

Como descrito no subitem 3.3.4 o BemCracker2D é solicitado através do BEMLAB2D que no Módulo IV informa qual tipo de processamento será realizado no solver que compõe-se de três módulos de processamento, estes são:

- MEC padrão (módulo I);
- MECD Sem Propagação (módulo II);
- MECD Com Propagação (módulo III)
    - Análise de Tensão com MEC
    - Avaliação de FITs (Integral J)
    - Avaliação da Direção/Correção do Crescimento da Trinca (Critério de Tensão Máxima)
    - Avaliação de Vida à Fadiga (Lei de Paris)

O programa BemCracker2D foi baseado nos trabalhos de Gomes (2000 e 2006) no que diz respeito à modelagem do MEC padrão (equação de deslocamento e uso de elementos quadráticos contínuos) e na estratégia da análise incremental devido a Portela (1993) e Aliabadi (2002). Baseado no diagrama de classes ilustrado na Figura 3.27, onde *BemCrk_BEMSYS* é a classe motora do programa e principal elo de ligação com a interface BEMLAB2D.

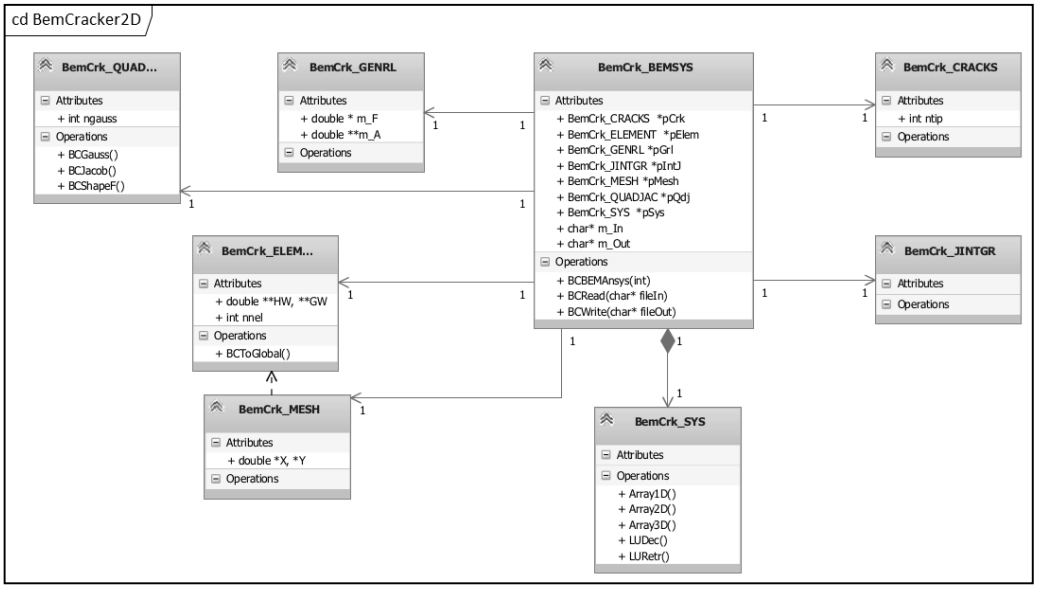


Fig. 3.27 – Diagrama de Classes do programa BemCracker2D (GOMES, 2016)

Como foi dito anteriormente a classe *BemCrk_BEMSYS* é responsável pela comunicação com entre os dois programas, sua finalidade é de fazer a leitura dos dados gerados pelo

BEMLAB2D, analisar, solucionar e imprimir os modelos de trinca. Seus dados de saída ficam disponíveis para leitura e visualização gráfica no pós-processamento do BEMLAB2D.

Outras classes importantes para o funcionamento adequado do BemCracker2D são interligadas a classe descrita anteriormente, todas com finalidade características, por exemplo a classe *BemCrk_GENRL* é responsável pela montagem do sistema de equações $Ax = By = f$, a classe *BemCrk_ELEMENT* desenvolve a montagem do elemento quadrático contínuo e descontínuo (C/D) do modelo gerado no BEMLAB2D, *BemCrk_MESH* é a classe que monta a malha do elemento (C/D) a partir dos elementos montados na classe anterior e a classe *BemCrk_QUADJAC* responsável por gerar os pontos de Gauss, o Jacobiano e as funções de forma (C/D). Ainda nos mesmos modelos de classe se encontra a classe *BemCrk_SYS* que é responsável pela alocação das matrizes e vetores e por resolver o sistema por LU.

As Classes que estão relacionadas com modelos de propagação de trinca são as classes *BemCrk_CRACKS* e *BemCrk_JINTGR*, onde a primeira é responsável pela montagem do incremento de crescimento de trinca e a segunda é responsável por gerar os fatores de intensidade de tensão baseados na Integral. Para melhor esclarecer o entendimento da estratégia da análise incremental do crescimento da trinca tem-se a implementação do diagrama de sequências que pode ser visualizado na Figura 3.28.

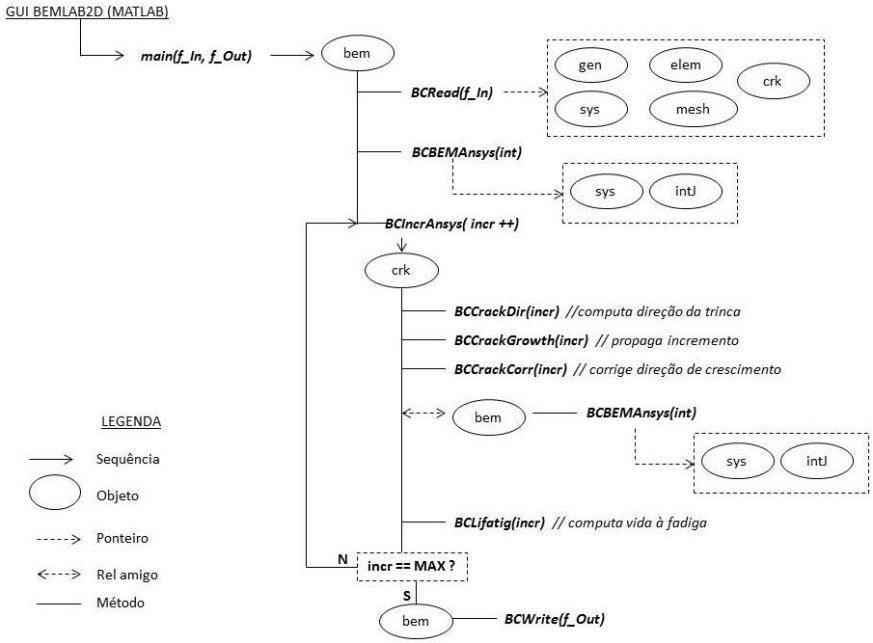


Fig. 3.28 – Diagrama de sequência do crescimento da trinca (GOMES, 2016)

4 - IMPLEMENTAÇÃO COMPUTACIONAL

Este capítulo abordará as técnicas e algoritmos utilizados para a implementação da interface BEMLAB2D desenvolvida neste trabalho, mostrando os vários estágios até sua finalização.

4.1 - AMBIENTE DE DESENVOLVIMENTO

A interface BEMLAB2D foi implementada em ambiente MATLAB, versão R2015a, pois, entre outros motivos, o MATLAB é uma plataforma que se destaca pela simplicidade com que se pode efetuar operações matriciais e plotagem de gráficos, além de possuir uma grande variedade de ferramentas, classes e funções nativas.

Uma dessas ferramentas chama-se GUI (*Graphical User Interface*). Esta é uma janela de exibição gráfica contendo componentes que permitem que o usuário realize tarefas interativas, como ler e gravar arquivos de dados, comunicar com outras interfaces gráficas e exibir dados como tabelas ou gráficos.

Não é necessário criar um *script* ou digitar comandos na linha de comando para realizar tarefas com a GUI. Os componentes da interface gráfica podem incluir menus, barras de ferramentas, vários tipos de botões, caixas de listagem, barra deslizante entre outras, dos quais não precisam de grandes entendimento do usuário para a execução das tarefas de cada uma ou do conjunto, ver Figura 4.1.

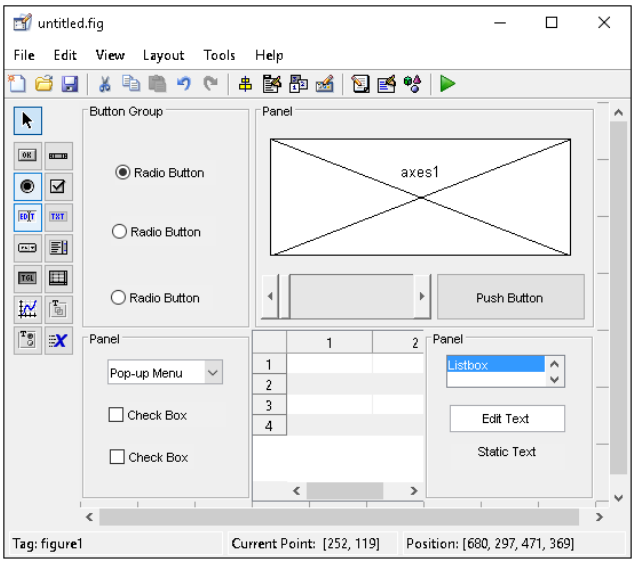


Fig. 4.1 – Ferramenta GUI e suas componentes. (MATLAB, 2015)

4.2 - ALGORITMO E IMPLEMENTAÇÃO

Nos tópicos a seguir serão descritos com mais detalhes os algoritmos e implementações de cada módulo que compõe o BEMLAB2D. Desde o início do desenvolvimento nas linhas de comando, como detalhamento sobre o funcionamento dos comandos e funções bases que operam cada sistema de funções serão descritos para um esclarecimento da construção desse trabalho.

4.2.1 - Módulo I - Geometry

O módulo I é responsável por gerar os modelos geométricos na área gráfica principal do BEMLAB2D, seus algoritmos e comandos fazem a comunicação entre *PushButton* com a *Axes*, principalmente, através das funções *Set* e *Get*, as quais são responsáveis pelo transporte de dados de informação, matrizes ou textos, entre cada função básica dentro do *script* do BEMLAB2D. As matrizes e textos são armazenados em áreas específicas sobre cada componente da interface. Por exemplo o *String*, *UserData*, *Value*, são espaços apropriados para armazenar vetores de coordenadas, ou matrizes topológicas.

A funções *Set* é responsável por armazenar as informações pertinentes de acordo com o espaço adotado:

```
function pushbutton_Callback(hObject, eventdata, handles)
set(handles.pushbutton, 'String' , 'IMPORTANTE')
A = zeros(1,4);
set(handles.pushbutton, 'UserData' , A) % Armazena Dados
```

A funções *Get* chama as informações e armazena em uma variável:

```
function pushbutton_Callback(hObject, eventdata, handles)
matriz = get(handles.pushbutton, 'UserData') % Atribui Dados
num = size(BP);
```

O Módulo I tem como funções principais *Points*, *Lines*, *Arcs* e *Zones* e foram implementados no *Script* do MATLAB, sendo de grande importância na construção de modelos diversos, com furos e zonas conexas, cada função será descrito a seguir. Uma representação hierárquica desses objetos pode ser visualizada na Figura 4.2.

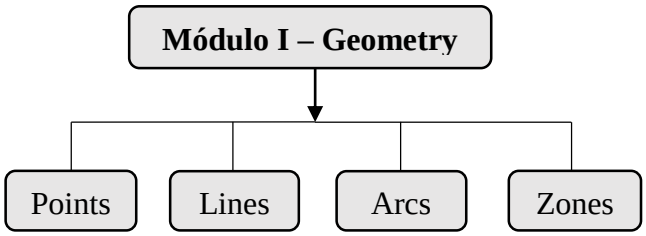


Fig. 4.2 – Representação hierárquica do Módulo I - Geometry

4.2.1.1 - Point

A função *Points* derivada do Módulo I foi implementada para abrir uma caixa de diálogo (Fig.3.5) a partir do comando *inputdlg* que possibilita a entrada de dois vetores de coordenadas. Essa entrada de dados é realizada via texto utilizando o teclado apenas. A Figura 4.3 ilustra a representação das funções de referências.

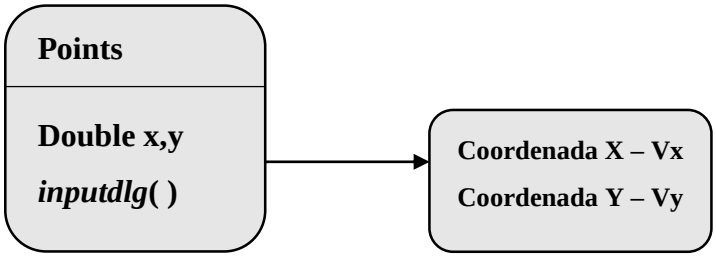


Fig. 4.3 – Representação esquemática da função Points

A função *Points* é um *callback* que manipula informações através das rotinas e comando implementados no *script* principal do BEMLAB2D, nesse *call-back* a matriz *BP* é gerada e armazena na primeira coluna a incidência do ponto, na segunda coluna é armazenada as coordenadas "X" e na terceira coluna armazena as coordenadas "Y". A seguir é apresentado um fragmento da rotina da função *Points*.

```
function point_Callback(hObject, eventdata, handles)
nome = 'Coordenadas dos Pontos Geométricos';
prompt = {'X:','Y:'};
numlinhas = 1;
resp = inputdlg(prompt, nome, numlinhas); % abre a caixa de diálogo
Vx = str2num(char(resp(1))); % vetor de coordenadas X
Vy = str2num(char(resp(2))); % vetor de coordenadas Y
axes(handles.axes1)
n = numel(Vx); % número de elementos do vetor de coordenadas X
plot(Vx,Vy,'ro'); % plota os pontos
for i=1:n
    BP(i,1)=i; BP(i,2)=Vx(i); BP(i,3)=Vy(i); % alocar matriz BP
    set(handles.point,'UserData',BP)
end
```

4.2.1.2 - Lines

Na construção do algoritmo da função *Lines* foram utilizado vários comandos de ação, sendo que os de maiores destaque são os comandos *datacursormode* e *waitforbuttonpress*. O comando *datacursormode* habilita o cursor do *mouse* na função de seleção, *waitforbuttonpress* é um comando de espera que define como "0" o clique do *mouse* e "1" a seleção de qualquer tecla no teclado. Sendo "0" a escolha na seleção o algoritmo permite com que continue a construir segmentos sem a necessidade de selecionar o botão *Lines* na interface gráfica novamente, já a seleção "1" finaliza o ciclo. O algoritmo da função *Lines* ainda toma como auxílio a função *myupdatefcn.m* que detecta a posição do clique do *mouse* e armazena a respectivas coordenadas do ponto para desenho do segmento reto. A Figura 4.4 ilustra a representação das funções de referências.

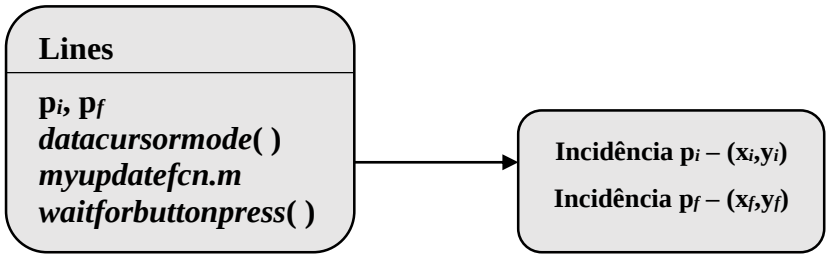


Fig. 4.4 – Representação esquemática da função Lines

No *callback* da função *Lines* é gerada a matriz *BL*, onde é armazena na primeira coluna o número "1" de segmento reto, na segunda coluna é armazenada a incidência do primeiro nó selecionado e na terceira coluna a incidência do segundo nó selecionado. Um fragmento da rotina da função *Points* é apresentado a seguir.

```
function lines_Callback(hObject, eventdata, handles)
datacursormode on; % abre cursor de seleção
BP = get(handles.point,'UserData');
pos = zeros(2,2);
n = numel(BP(:,1));
for i=1:2 % dois pontos a selecionar
    dcm_obj = datacursormode(gcf); % a
    set(dcm_obj,'UpdateFcn',@myupdatefcn);
    w = waitforbuttonpress; % espera a seleção
    if w == 1
        break
    end
    pos(i,:) = get(0,'userdata'); % armazena coordenadas da seleção
    for j=1:n
        difx = BP(j,2) - pos(i,1);
        dify = BP(j,3) - pos(i,2);
        if abs(difx)<1e-4 && abs(dify)<1e-4
            if i==1
```

```
                BL(i,2)=BP(j,1); % aloca segunda coluna
            else
                BL(i,3)=BP(j,1); % aloca terceira coluna
            end
            if BL(i,2)~=0 || BL(i,3)~=0
                BL(i,1) = 1; % aloca primeira coluna
                break
            end
        end
    end
end
plot(pos(:,1), pos(:,2),'r-');
datacursormode off;
```

4.2.1.3 - Arcs

Na construção do algoritmo da função *Arcs* são usados os mesmos comandos que foram utilizados na função *Lines*, *datacursormode*, *waitforbuttonpress* e a função *myupdatefcn.m*. A Figura 4.5 ilustra a representação das funções de referências.

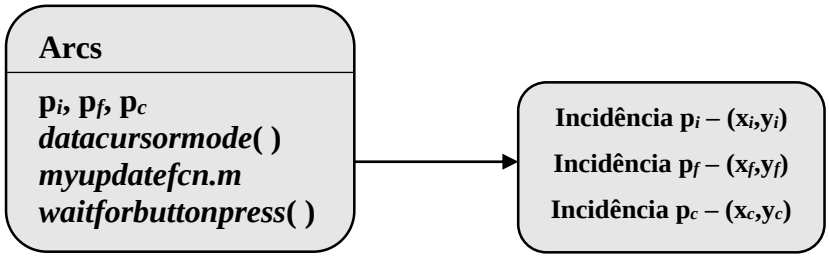


Fig. 4.5 – Representação esquemática da função Arcs

A função *Arcs* é um *callback* que manipula informações da matriz *BC* gerando e armazenando na primeira coluna o número "3" de segmento curvo, na segunda coluna é armazenada a incidência do primeiro nó selecionado e na terceira coluna a incidência do segundo nó selecionado e na quarta coluna a incidência do terceiro nó selecionado (+ incidência = anti-horário; - incidência = horário). A seguir é apresentado um fragmento da rotina da função *Arcs*.

```
function arcs_Callback(hObject, eventdata, handles)
BP = get(handles.point,'UserData');
n = numel(BP(:,1));
datacursormode on; % abre cursor de seleção
for i=1:3 % três pontos a selecionar
    dcm_obj = datacursormode(gcf);
    set(dcm_obj,'UpdateFcn',@myupdatefcn);
    w = waitforbuttonpress; % espera a seleção
    if w == 1
        break
    end
    pos(i,:) = get(0,'userdata'); % armazena coordenadas da seleção
    for j=1:n
```

```
        difx = pos(i,1) - BP(j,2); dify = pos(i,2) - BP(j,3);
        if abs(difx)<1e-4 && abs(dify)<1e-4
            pos(i,1) = BP(j,2); pos(i,2) = BP(j,3);
            if i==1
                BC(k,2) = BP(j,1); % aloca segunda coluna
            end
            if i==2
                BC(k,3) = BP(j,1); % aloca terceira coluna
            end
            if i==3
                % Caixa de dialogo para definir sentido do arco
                str = {'( Horário )','( Anti-horário )'};
                s = listdlg('PromptString','Selecione a direção de
                desenho do arco:','SelectionMode','single',
                'ListSize',[160 31],'ListString',str);
                sent = -1;
                if s==2
                    sent = 1;
                end
                BC(k,1) = 3; % aloca primeira coluna
                BC(k,4) = sent*(BP(j,1)); % aloca quarta coluna
            end
            break
        end
    end
end
%Plotagem da Curva
ic = BC(k,4); irt = sign(ic); = BC(k,2);
n2 = BC(k,3); n3 = irt*ic; ne = 2 nn = ne*2;
for j=1:n %Definindo as Coordenadas dos pontos do Arco
    if BP(j,1)==n1
        x0=BP(j,2); y0=BP(j,3);
    end
    if BP(j,1)==n2
        xL=BP(j,2); yL=BP(j,3);
    end
    if BP(j,1)==n3
        a=BP(j,2); b=BP(j,3);
    end
end
by0 = b - y0; ax0 = a - x0; byL = b - yL; axL = a - xL;
rad1 = sqrt(by0^2 + ax0^2); rad2 = sqrt(byL^2 + axL^2);
if (abs(rad1-rad2)>0.01)
    break;
end
theta0 = atan2(y0-b,x0-a); thetaL = atan2(yL-b,xL-a);
dif = thetaL - theta0;
if (abs(dif)<1e-5)
    dif = 0.;
end
if dif < 0.
    dif = 2*pi + dif;
end
if irt > 0.
    dthet = dif/nn;
else
    dthet = -(2*pi - dif)/nn;
end
dx = zeros(1,100*nn); dy = zeros(1,100*nn);
for j=1:nn
    th_i=theta0+(j-1)*dthet; th_f=theta0+j*dthet;
```

```
    t = linspace(th_i,th_f); dx1 = rad1*cos(t) + a;
    dy1 = rad1*sin(t) + b; jk = (j-1)*100;
    for jj = 1:100
         dx(1,jk+jj) = dx1(1,jj);
         dy(1,jk+jj) = dy1(1,jj);
    end
end
    plot(dx,dy,'r-')
end
datacursormode off;
```

4.2.1.4 - Zones

A função *Zones* é na sequência a última função derivada do Módulo I. A seleção dos segmentos retos e curvos de uma zona são realizados com o clique do mouse, sendo que a habilitação da seleção ocorre através da função *LineSelected.m*, nessa função seu algoritmo trabalha com a alteração da espessura do segmento selecionado para facilitar a visualização do usuário e também armazena as coordenadas do ponto no exato momento do clique com o comando *intersectionpoint*. A coordenada lida na função *Zones* e assim identifica qual segmento armazenando parâmetros na matriz pertinente aos dados de cada zona. A Figura 4.6 ilustra a representação das funções de referências.

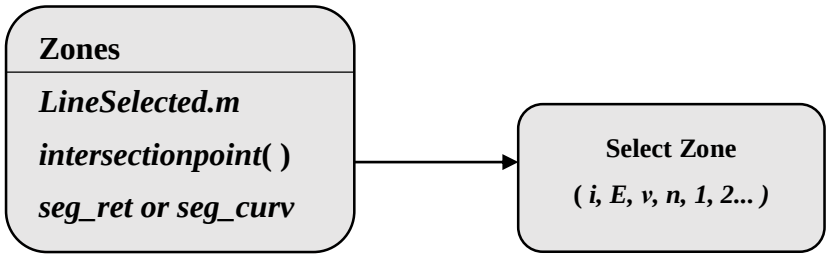


Fig. 4.6 – Representação esquemática da função Zonas

No *callback* da função *Zones* é gerada a matriz *BZ*, onde é armazena na primeira coluna o número "1" identificando a primeira zona como "Mestre", na segunda coluna é armazenada o módulo de elasticidade, na terceira coluna o coeficiente de Poisson, na quarta coluna o número de segmentos e nas colunas seguintes os segmentos dessa zona em uma sequência lógica de construção. Um fragmento da rotina da função *Zones* que é responsável por chamar a interface auxiliar é apresentado a seguir.

```
function zones_Callback(hObject, eventdata, handles)
% Abrindo a Interface Auxiliar Zona
f=Zona;
waitfor(f)
% Chama informações de restrição: zona = [tipo_zona, mod_elast,
coef_poisson]
zona = get(0,'UserData'); % retorna dados gerados na interface Zona
```

```
if zona(1,1)==1 % tipo_zona: Mestre
    % Módulo de Elasticidade.
    young = zona(1,2);
    set(handles.text8, 'String' , young)
    % Coeficiente de Poisson.
    rnu = zona(1,3);
    set(handles.text9, 'String' , rnu)
end
if zona(1,1)~=0
    set(H, 'ButtonDownFcn', {@LineSelected, H}) %Seleciona o segmento
    w = waitforbuttonpress;
    coord_aux(1,:) = get(0,'userdata'); );%retorna coord. do clique
    %Definindo as Coordenadas dos pontos inicial e final da reta
    %...
    %Algoritmo de identificação do segmento.
    %Algoritmo de montagem das matrizes BL, BC e BZ
    %...
end
set(handles.text16,'UserData',BL)
set(handles.text17,'UserData',BC)
set(handles.text18,'UserData',BZ)
set(handles.text14,'UserData',Maux)
```

Dois algoritmos extenso são mencionados nos comentários do fragmento da rotina apresentada acima. O primeiro se refere a um algoritmo que identifica o segmento selecionado a partir a coordenada do clique do *mouse* obtidas através das informações extraídas da função *LineSelected.m*. Sendo identificado o segmento, no caso se é um segmento reto ou curvo, o segundo algoritmo altera as informações da primeira coluna das matrizes *BL* e *BC*. O usuário realizará a seleção dos segmentos de uma zona qualquer em um padrão sequencial de construção, ou seja, o primeiro segmento selecionado será identificado na sua matriz de origem e o número “1” na primeiro coluna a ele, será atribuído. Para o segundo será atribuído o número “2”, o terceiro o número “3” e assim por diante até fechar a zona. Ao mesmo tempo a matriz *BZ* é criada, onde cada linha representa uma zona, no final da seleção da primeira zona a quarta coluna é atribuído o número de segmentos selecionados até o fechamento da zona. Assim é repedido para todas as zonas até finalizar todas as zonas do modelo geométrico. A Figura 4.7 apresenta um modelo e a matriz BZ deste modelo para melhor entendimento.

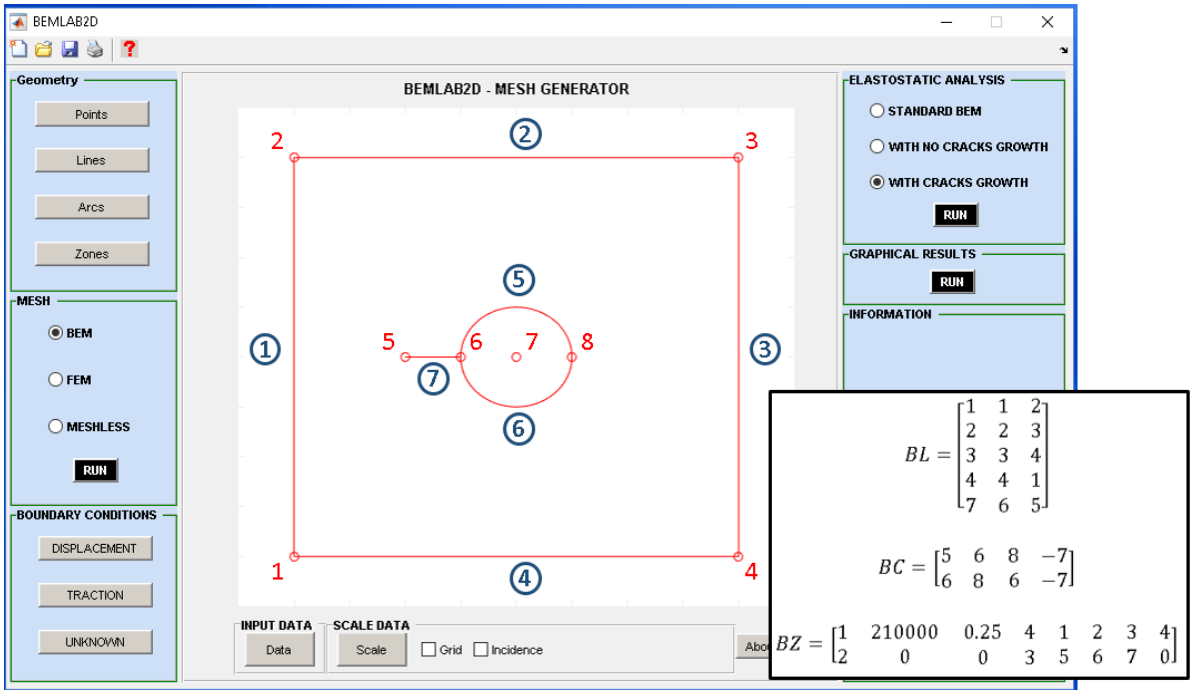


Fig. 4.7 – Construção da matriz *BZ* e modificação das matrizes *BL* e *BC*

4.2.2 - Módulo II - Mesh

O Módulo II é construído em apenas uma função chamada *run_mesh* e foi implementada no *Script* do MATLAB para interagir com o BEMLAB2D. Na interface, o Módulo II está em forma de *PushButton* e *RatioButtons*, sendo de grande importância na construção de três tipos de malha: Malha de Elementos de Contorno (MEC), Malha de Elementos Finitos (MEF) e Métodos Sem Malha (MESHLESS). Uma representação hierárquica desse módulo pode ser visualizada na Figura 4.8.

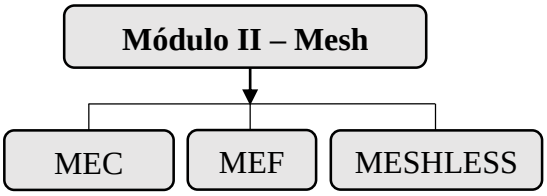


Fig. 4.8 – Representação hierárquica do Módulo II – Mesh

A função *run_mesh* é particionada em três algoritmos bases, onde os dois primeiros são independentes da malha escolhida e o terceiro trabalha de forma distinta de acordo com a opção de malha.

As malhas de MEC, MEF e MESHLESS são construídas a partir do detalhamento do contorno do problema, ou seja, qualquer que seja a malha definida a ser construída na

interface gráfica, o primeiro algoritmo base ativa a função *LineSelected* tornando possível o usuário selecionar cada segmento reto ou curvo e definir a quantidade de elementos que este terá. Ao selecionar o segmento reto uma interface gráfica auxiliar será acionada, *SettingLine* (Fig. 3.13) é uma interface auxiliar na qual o usuário define os parâmetros do segmento reto. Para segmentos curvos uma caixa de diálogo abrirá a partir do comando *inputdlg* para ser informado o número de elementos para cada segmento. Um fragmento do primeiro algoritmo base é apresentado a seguir.

```
function run_mesh_Callback(hObject, eventdata, handles)
BP = get(handles.point, 'UserData');
BL = get(handles.lines, 'UserData');
BC = get(handles.arcs, 'UserData');
BZ = get(handles.zones, 'UserData');
v_bem = (get(handles.radiobutton8,'Value'));
v_fem = (get(handles.radiobutton9,'Value'));
v_meshless = (get(handles.radiobutton10,'Value'));
n = numel(BL(:,1))+numel(BC(:,1));
for i=1:n
    % Abre o cursor de seleção do mouse
    set(H, 'ButtonDownFcn', {@LineSelected, H})
    w = waitforbuttonpress;
    coord_aux(1,:) = get(0,'userdata');
    % Segmento reto selecionado: modificar matrizes BL, BC e BZ
    f=SettingLine; % Chama a Interface Gráfica Auxiliar
    waitfor(f);
    setline = get(0,'UserData');%retorna dados gerados em SettingLine
    tipseg = setline(1,1); % Segmento de trinca ou não
    % Remodelando BZ
    % Remodelando BL e BC
    % ...
    % Segmento curvo selecionado: modificar matrizes BL, BC e BZ
    nome = 'Número de Elementos';
    prompt = {'Digite o número de elementos:'};
    numlinhas = 1;
    resposta = inputdlg(prompt, nome, numlinhas);
    numelem = str2num(char(resposta(1)));
    % Remodelando BC
    BZ1 = BZ(:,2:3);
    BZ = horzcat(BZ(:,1),BZ(:,4:end);
    set(handles.text16,'UserData',BL)
    set(handles.text17,'UserData',BC)
    set(handles.text18,'UserData',BZ)
    set(handles.text19,'UserData',BZ1)
end
```

As matrizes de dados de entrada para gerar qualquer uma das três malhas são geradas nas funções *Points*, *Lines*, *Arcs* e *Zones*, as quais são brevemente representadas a seguir na Tabela 4.1.

Módulo I – Geometry

Matriz de Pontos	Matriz de Segmentos Retos
BP = (*inc, coord_x, coord_y)*	**BL = (*i, inc_i, inc_f)***
Matriz de Segmentos Curvos	Matriz de Zonas
BC = (*i, inc_i, inc_f, inc_c)*	**BZ = (*i, num_seg, seg1, seg2...)***

Tab. 4.1 – Representação das Matrizes Constitutivas Incompletas

Após o primeiro algoritmo base da função *run_mesh* finalizado, os segmentos retos serão acrescidos de algumas colunas. A quantidade depende do segmento que tiver o maior número de elementos, ou seja, será acrescido o número de colunas a direta da matriz *BL* respectivo ao número de elementos do segmento com maior número de elementos mais uma coluna. Se houver segmentos com números de elementos diferentes, as colunas mais à direita dos segmentos com menor número de elementos serão preenchidas com zero. Ficando assim a quarta coluna representando o número de elementos por segmento e as restantes a razão desses elementos sobre o comprimento total do segmento.

O segmentos de trinca em particular provocam alterações mais bruscas na matriz *BL*, ao se definir um segmento de trinca o algoritmo base gera automático um segmento com incidências inicial e final invertida e com características semelhantes ao segmento originalmente selecionado. O algoritmo insere uma linha na matriz *BL* logo abaixo da linha do segmento selecionado, alterando também a matriz *BZ*, gerando uma nova cadeia de construção para a zona que a trinca pertence. A matriz *BC* será acrescida apenas uma coluna à direita, representando o número de elementos do segmento curvo. A tabela acima ficará com as matrizes completas como é mostrada abaixo na Tabela 4.2.

Módulo II – Mesh – (Matrizes Constitutivas Completas)

Matriz de Pontos	Matriz de Segmentos Retos
BP = (*inc, coord_x, coord_y)*	**BL = (*i, inc_i, inc_f, num_elem, raz....)***
Matriz de Segmentos Curvos	Matriz de Zonas
BC = (*i, inc_i, inc_f, inc_c, num_elem)*	**BZ = (*i, num_seg, seg1, seg2...)***

Tab. 4.2 – Representação das Matrizes Constitutivas Completas

O segundo algoritmo base da função *run_mesh* é baseado em um arquivo chamado *pre_file.m* (GOMES, 2016), cuja formatação original em FORTRAN é devido a Portela (1993). Basicamente a estrutura desse algoritmo puxa as informações pertinentes das matrizes constitutivas *BP*, *BL*, *BC* e *BZ* e monta a matriz de coordenadas de todos os pontos do contorno (*m_inc*) e também a matriz topológica da malha de contorno (*m_iel*), como pode ser visto no fragmento do algoritmo a seguir.

```
if BZ(4,1)~=0 %EXISTE BZ
   in = 1;nge = 0;ngn = 0;
   %zona, numero de segmentos, segmentos contorno
   %num_bz: TOTAL DE ZONAS OU REGIOES FECHADAS
   for i=1:numel(BZ(:,1))
       p = 2;
       nseg = BZ(i,2);
       for j=1:nseg
           iseg(j) = BZ(i,p+j);
       end
       is = iseg(1);
       %gerando dados nodais
       ngn = ngn + 1;
       x1(ngn) = dx(is,1); %dx é encontrado pelas matrizes BL e BC
       y1(ngn) = dy(is,1); %dy é encontrado pelas matrizes BL e BC
       for j=1:nseg
           is = iseg(j);
           nn = ne(is)*2+1;
           for k=2:nn
               if (j==nseg && k==nn)
                   break;
               else
                   ngn = ngn + 1;
                   x1(ngn) = dx(is,k);y1(ngn) = dy(is,k);
               end
           end
       end
       %gerando dados dos elementos
       nge0 = nge + 1;
       nge1 = int16(ngn/2);
       for j=nge0:nge1
           iel(1,j) = in;
           iel(2,j) = in+1;
           iel(3,j) = in+2;
           in = in + 2;
       end
       iel(3,nge1) = iel(1,nge0);
       nge = nge1;
    end
end
%nodes of boundary
for i=1:nnod
        xx1(i,1)=x1(i);
        yy1(i,1)=y1(i);
end
n = numel(x1);
m_inc = zeros(n,3);
for i=1:n
    m_inc(i,1) = i;
```

```
    m_inc(i,2) = xx1(i);
    m_inc(i,3) = yy1(i);
end
set(handles.run_mesh,'UserData',m_inc)
%dados dos elementos
for i=1:nnel
    for j=1:3
        m_iel(j,i) = iel(j,i);
    end
end
m_iel=m_iel';
set(handles.radiobutton8,'UserData',m_iel)
```

Outros detalhes do arquivo *pre_file.m* serão discutidos no subitem 4.2.4 mais a frente. Abaixo é apresentada a hierarquia de desenvolvimento da função *run_mesh* até a entrada no terceiro e último algoritmo base, ver Figura 4.9.

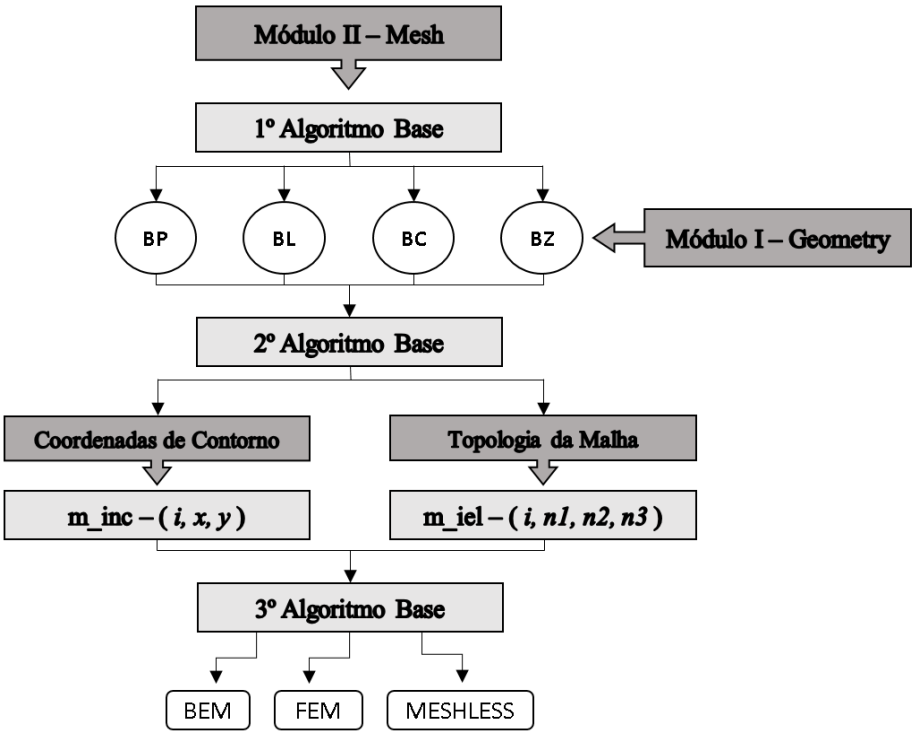


Fig. 4.9 – Hierarquia do Módulo II da Interface BEMLAB2D.

O terceiro algoritmo base será relatado com mais detalhes nos próximos três subitens abaixo, sendo que para cada tipo de malha ainda será apresentado características distintas de implementação no que se diz respeito a geração de malha.

4.2.2.1 - Malha de MEC

Como já foi dito no capítulo 3, a característica principal do Módulo II é a capacidade de reproduzir as malhas de MEF e MESHLESS a partir das características da malha gerada

pela malha do Método dos Elementos de Contorno (MEC). Sendo assim, a malha de MEC será apenas plotada na área gráfica do BEMLAB2D ao final do segundo algoritmo base, não precisando de outros tratamentos como pode-se ver no fragmento do código a seguir.

```
for i=1:numel(m_iel(:,1))
    xi = m_inc(m_iel(i,1),2);
    yi = m_inc(m_iel(i,1),3);
    xf = m_inc(m_iel(i,3),2);
    yf = m_inc(m_iel(i,3),3);
    xm = m_inc(m_iel(i,2),2);
    ym = m_inc(m_iel(i,2),3);
    X = [xi xm xf];
    Y = [yi ym yf];
    hold on
    plot(X,Y,'-r',xi,yi,'xr',xf,yf,'xr',xm,ym,'.r');
    hold off
end
```

A estratégia de modelagem da trinca usada nos algoritmos do Módulo II para a malha de elementos de contorno segue a mesma estratégia aplicada por Portela (1992). Ao desenhar o modelo com as ferramentas do Módulo I a trinca só precisa de um segmento reto, ao se acessar na interface gráfica auxiliar *SettingLine* que abre no processo do primeiro algoritmo base, descrito anteriormente, o segmento definido como trinca constrói um segmento em sentido contrário e de coordenadas coincidente de forma automática, configurada por elementos quadráticos descontínuos, ver Figura 4.10.

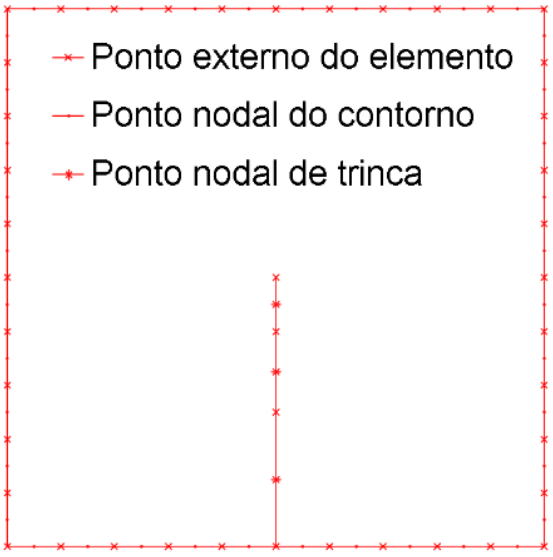


Fig. 4.10 – Esquema de modelagem de trinca com elementos de contorno quadráticos descontínuos.

Ainda no terceiro algoritmo base é aberto uma caixa de diálogo (Fig. 3.5) a partir do comando *inputdlg* que serve para informar, sobre formato de texto, dois vetores de

coordenadas X e Y, que servem para montar a matriz de pontos internos e plotar na área gráfica do BEMLAB2D. Um fragmento do terceiro algoritmo base é apresentado a seguir.

```
% ...
% Pontos Internos ==========================================
if v_bem==1
    resp = questdlg('O problema terá pontos internos?','Pontos
    Internos','Sim','Não','Não');
    switch resp
        case 'Sim'
            nome = 'Coordenadas dos Pontos Geométricos';
            prompt = {'X:','Y:'};
            numlinhas = 1;
            resposta = inputdlg(prompt, nome, numlinhas);
            Vx = str2num(char(resposta(1)));
            Vy = str2num(char(resposta(2)));
            axes(handles.axes1)
            plot (Vx,Vy,'b*');
    end
    for i=1:numel(m_iel(:,1))
        xi = m_inc(m_iel(i,1),2); yi = m_inc(m_iel(i,1),3);
        xf = m_inc(m_iel(i,3),2); yf = m_inc(m_iel(i,3),3);
        xm = m_inc(m_iel(i,2),2); ym = m_inc(m_iel(i,2),3);
        X = [xi xm xf]; Y = [yi ym yf];
        hold on
        plot(X,Y,'-r',xi,yi,'xr',xf,yf,'xr',xm,ym,'.r');
        hold off
    end
end
% ...
```

4.2.2.2 - Malha de MEF

Após a etapa 1 e 2 do algoritmos base realizada, o terceiro algoritmo quando selecionado na opção da malha do Método dos Elementos Finitos (MEF) engloba todo o domínio em um único retângulo. Assim, na geométrica é construído de maneira uniforme disponíveis sobre o eixo horizontal pré-definido. A quantidade desses eixos é informada através de uma caixa de diálogo (ver código a seguir), que é proporcional ao refinamento da malha.

```
if v_fem==1
    nome = 'Fator de refinamento da malha (>=5)';
    prompt = {'Número de Eixos:'};
    numlinhas = 1;
    defaultans = {'20'};
    resposta = inputdlg(prompt, nome, numlinhas, defaultans);
    maxEixos = str2num(char(resposta(1)));%Número de Eixos horizontais
    % máximos
    dY = h / (maxEixos - 1);
    dX = (dY * 2) / sqrt(3);
    nNoEixo = round( b / dX ); %Número de nós no Eixo
    nLinha = nNoEixo * maxEixos;%Número de linhas da matriz de coords
    coords = zeros(nLinha, 2); %Pré alocação da matriz de coords
    nLC = 0; %Controle de linhas matriz coords
end
```

A Figura 4.11 apresenta dois níveis de refinamento.

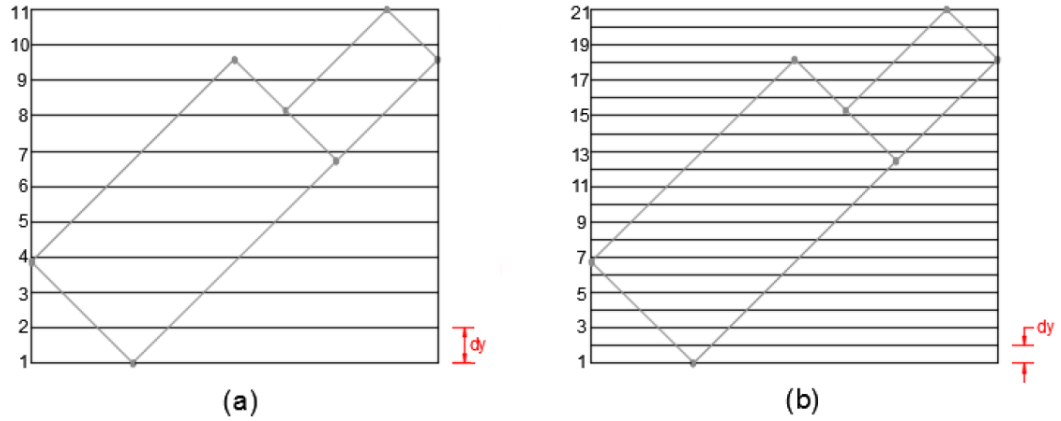


Fig. 4.11 – Eixos horizontais conforme o nível de refinamento

Sendo *dy* encontrado por uma simples expressão algébrica, pode-se criar os eixos verticais de forma que triângulos equiláteros sejam formado entre os eixos. Assumindo os limites do bordo do modelo conhecido, ou seja, conhecido as coordenadas dos pontos de contorno calculados no segundo algoritmo base, pode-se filtrar apenas os nós internos do domínio, como é apresentado no algoritmo a seguir.

```
%Inserção dos nós internos
dy = h / (maxEixos - 1);
dx = (dy * 2) / sqrt(3);
nNoEixo = round( b / dx );
y = yMIN;
for i = 1 : maxEixos;
    y = y + dy;
    if mod(i, 2) == 1 %Linhas impares
        for j = 0 : nNoEixo;
            if j == 0
                x = xMIN;
            else
                if j == nNoEixo
                    x = xMAX;
                else
                    x = x + dx;
                end
            end
            nLC = nLC + 1;
            coords (nLC, 1) = x;
            coords (nLC, 2) = y;
        end
    else %Linhas pares
        for j = 0 : nNoEixo + 1;
            if j == 0
                x = xMIN;
                nLC = nLC + 1;
                coords (nLC, 1) = x;
                coords (nLC, 2) = y;
                x = x + (dx / 2);
            else
```

```
                if j == nNoEixo + 1
                    x = xMAX;
                else
                    x = x + dx;
                end
            end
            nLC = nLC + 1;
            coords (nLC, 1) = x;
            coords (nLC, 2) = y;
        end
    end
end
```

A Figura 4.12 apresenta um modelo representativo de pontos internos no domínio do modelo geométrico.

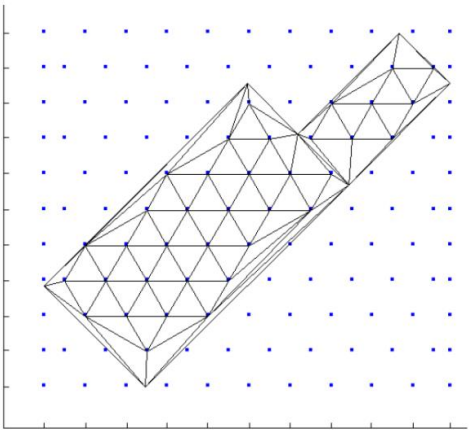


Fig. 4.12 – Malha gerada com os nós da região interna e os vértices do domínio

Com a definição dos nós, o processo de geração de malha se dá a partir do manuseio das funções da classe *DelaunayTriangulation*.

```
%Geração da malha
c_Nodais = [coordsContorno; coords];
nContorno = numel(coordsContorno(:,1));
C = m_iel;
D = c_Nodais;
DT = delaunayTriangulation( D, C );
io = isInterior(DT);
temp = DT.ConnectivityList(io, :);
[coordsNodais] = suavizar(c_Nodais, temp, nContorno);
triplot(temp, coordsNodais(:,1), coordsNodais(:,2))
```

Como pode ser visto na Figura 4.13 os elementos próximo ao bordo não são de tão boa qualidade quanto os elementos do centro.

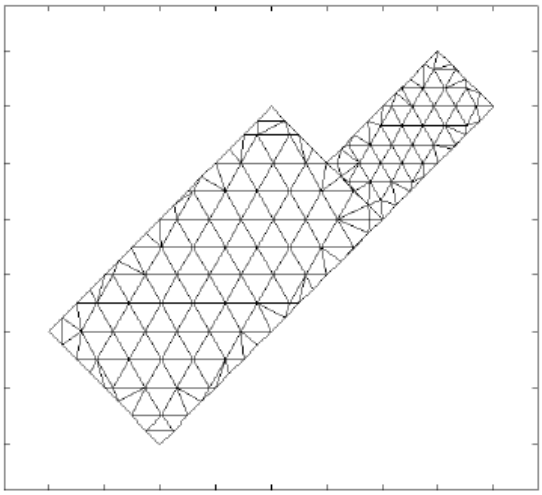


Fig. 4.13 – Resultado Final de uma Malha de Elementos Finitos antes da Suavização

Isso foi resolvido aplicando a suavização Laplaciana com o avanço da fronteira de três frentes de geração. No fragmento apresentado anteriormente a função *suavizar( )* pode ser visualizada, a qual apresenta um resultado considerável de refinamento dos elementos próximo ao contorno, ver Figura 4.14.

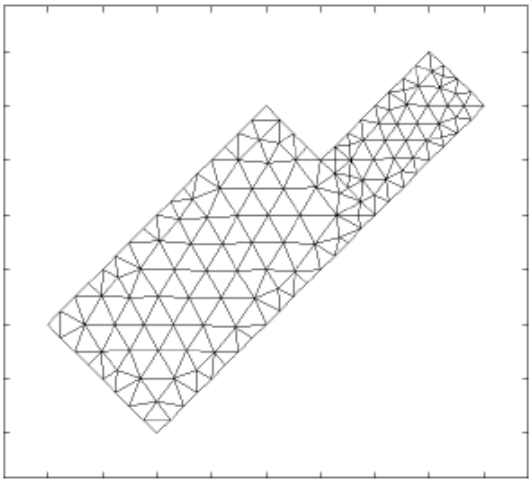


Fig. 4.14 – Resultado Final de uma Malha de Elementos Finitos após Suavização

A etapa 3 do algoritmo base ainda conta com um suporte de geração de arquivos de texto, montando e armazenando as informação pertinentes a malha de MEF. As coordenadas nodais e a topologia da malha são as informações que compões o arquivo texto, um fragmento do algoritmo é apresentado e logo em seguida um exemplo resumido de um arquivo texto de uma malha de MEF.

```
xx_fem = coordsNodais(:,1);
yy_fem = coordsNodais(:,2);
m_ief = conect;
%Montando arquivo modelo de elemento de contorno
fout=fopen('arquivo_fem.dat','w');% open file output
% call output
output_FEM(fout,xx_fem,yy_fem,m_ief);
fclose(fout);
```

arquivo_fem.dat - Bloco de notas

```
Nodal_Coordinates_(NODE,X,Y)
1  1.000000  1.000000
2  1.000000  1.800000
3  1.000000  2.600000
4  1.000000  3.400000
5  1.000000  4.200000
6  1.000000  5.000000
7  1.000000  5.800000
8  1.000000  6.600000
9  1.000000  7.400000
10  1.000000  8.200000
...
Mesh_Topology_(ELEMENT,G-NODE1,G-NODE2,G-NODE3)
1 97 103 96
2 108 16 21
3 58 2 13
4 74 67 75
5 22 60 59
6 64 72 71
7 66 67 74
8 1 13 2
9 60 32 68
10 32 33 68
...
```

4.2.2.3 - Malha de MESHLESS

Na etapa 3 os algoritmos geradores de malha de MEF são compartilhados também para a malha do Método Sem Malha (MESHLESS). Tem-se que os pontos internos gerados são plotados na área gráfica do BEMLAB2D sem o uso da matriz de topologia da malha gerada na etapa 2, assim como pode ser visto na fragmentação do algoritmo a seguir.

```
%Plotar os pontos do contorno
hold on
plot(xx1,yy1,'.r');
hold off
%Plotar os pontos do internos
if maxEixos>=5
    hold on
    plot(pontosinternos(:,1),pontosinternos(:,2),'.b');
    hold off
```

```
end
xx_meshless = coordsNodais(:,1);
yy_meshless = coordsNodais(:,2);
%Montando arquivo modelo de elemento de contorno
fout=fopen('arquivo_meshless.dat','w');% open file output
% call output
output_MESHLESS(fout,xx_meshless,yy_meshless);
fclose(fout);
```

No final do algoritmo anterior tem-se linhas de comando referente ao gerador de arquivos de texto, montando e armazenando as informação pertinentes a malha de MESHLESS, um exemplo simplificado do arquivo texto é apresentado a seguir, onde apenas as coordenadas dos pontos são de importância para este tipo de malha.

arquivo_meshless.dat - Bloco de notas

```
Nodal_Coordinates_(NODE,X,Y)
1   1.000000   1.000000
2   1.000000   1.800000
3   1.000000   2.600000
4   1.000000   3.400000
5   1.000000   4.200000
6   1.000000   5.000000
7   1.000000   5.800000
8   1.000000   6.600000
9   1.000000   7.400000
10   1.000000   8.200000
11   1.000000   9.000000
12   1.800000   9.000000
13   1.800000   1.000000
14   2.048943   6.690983
15   2.048943   7.309017
16   2.412215   7.809017
17   2.412215   6.190983
18   2.600000   1.000000
19   2.600000   9.000000
20   3.000000   6.000000
...
```

4.2.3 - Módulo III – Boundary Conditions

O módulo III é responsável por inserir condições de contorno, as quais são inserida na área gráfica do BEMLAB2D a partir da comunicação entre os *PushButton* e *GUI's* auxiliares com a *Axes*. As funções *Set* e *Get* são responsáveis por toda a comunicação entre as *GUI's* auxiliares e os *PushButton* transportando as informações de condições de contorno adotada.

O módulo III é constituído por duas funções no *script* do BEMLAB2D, elas são *DISPLACEMENT* e *TRACTION*, as duas possuem independência total entre elas quando usadas. Uma representação hierárquica desse módulo pode ser visualizada na Figura 4.15.

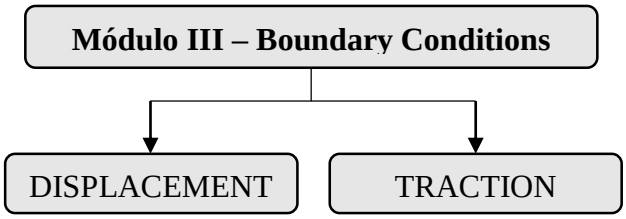


Fig. 4.15 – Representação hierárquica do Módulo III – Boundary Conditions

4.2.3.1 - Displacement

A função *displacement* que se encontra no *script* do BEMLAB2D inicialmente abre a interface gráfica auxiliar *Displacement.fig* (Figura 3.19) onde dentro do *script* da interface auxiliar é armazenada uma matriz com informações sobre restrições de um ou vários elementos, algumas informações das linhas de comando pode ser visto abaixo.

```
dir_x = get(handles.direct_x,'Value');
dir_y = get(handles.direct_y,'Value');
dir_n = get(handles.direct_n,'Value');
rec_x = str2num(get(handles.edit2,'String'));
rec_y = str2num(get(handles.edit4,'String'));
rec_n = str2num(get(handles.edit5,'String'));
n = numel(elem); displac = zeros(n,4);
for i=1:n
    displac(i,1) = elem(i);% Elemento restringido
    displac(i,2) = node;   % Nó restrit [1-nó1,2-nó2,3-nó3,4-todo o elem]
    displac(i,3) = dir_x;  % Restrição na direção X [1-restrito, 0-livre]
    displac(i,4) = dir_y;  % Restrição na direção Y [1-restrito, 0-livre]
    displac(i,5) = rec_x;  % Deslocamento prescrito na direção X
    displac(i,6) = rec_y;  % Deslocamento prescrito na direção Y
    displac(i,7) = dir_n;  % Restrição na direção Normal [1-restrito,
                           % 0-livre]
    displac(i,8) = rec_n;  % Deslocamento prescrito na direção Normal
end
set(0,'UserData',displac)
```

A matriz de informações de restrições (*displac*) é retornada ao algoritmo principal do BEMLAB2D. As informações da matriz lidas pelo algoritmo são comparadas com a matriz de coordenadas nodais (*m_inc*) e matriz topológica (*m_iel*), onde o algoritmo de forma automática gera as imagens geométricas dos apoios (primeiro ou segundo grau), a partir das coordenadas dos nós restritos, na área gráfica do BEMLAB2D.

4.2.3.2 - Traction

A função *traction* foi implementada de forma semelhante a função *displacement*. Ao acessar a função *traction* a interface auxiliar *Traction.fig* (Figura 3.20) se abrirá para que o usuário informe as condições de contorno de força ou tensões, essas são armazenadas na matrizes de informações de solicitação (*tract*) que interagem com o algoritmo principal do BEMLAB2D. Algumas informações das linhas de comando pode ser visto a seguir.

```
dir_x = str2num(get(handles.direct_x,'String'));
dir_y = str2num(get(handles.direct_y,'String'));
dir_n = str2num(get(handles.direct_n,'String'));
n = numel(elem);
k = 0;
if dir_x~=0
    k = k+1;
end
if dir_y~=0
    k = k+1;
end
if dir_n~=0
    k = k+1;
end
tract = zeros(k*n,4);
for i=1:n
    tract(i,1) = elem(i);% Elemento solicitado
    tract(i,2) = node;   % Nó solicitado [1-nó1, 2-nó2, 3-nó3, 4-todo
                         % o elemento]
    if dir_x~=0
        k = k+1;
        tract(k,3) = 1;      % Direção X solicitada
        tract(k,4) = dir_x;  % Valor da solicitação em X
    end
    if dir_y~=0
        k = k+1;
        tract(k,3) = 2;      % Direção Y solicitada
        tract(k,4) = dir_y;  % Valor da solicitação em Y
    end
    if dir_n~=0
        k = k+1;
        tract(k,3) = 0;      % Direção Normal solicitada
        tract(k,4) = dir_n;  % Valor da solicitação Normal
    end
end
set(0,'UserData',tract)
```

Como no subitem acima, as informações das matrizes são usadas para definir as coordenadas dos pontos de incidência das forças ou tensões gerando, assim, as imagens geométricas dos vetores. Quando força, o vetor será encontrado sobre um único nó do elemento, quanto tensão, serão vários vetores distribuído uniformemente sobre o elemento.

4.2.4 - Módulo IV – Elastostatic Analysis

Através das funções *Set* e *Get*, que fazem o armazenamento e busca, respectivamente, dos dados de informação, matrizes ou textos, entre cada função básica dentro do *script* do BEMLAB2D, os algoritmos e comandos do módulo IV realizam a interação entre *PushButton* e *RatioButtons* com a *Axes*.

O Módulo IV é construído em apenas uma função chamada *run_analysis* que foi implementada no *Script* principal do BEMLAB2D. Na interface, o Módulo IV está em forma de *PushButton* e *RatioButtons*, sendo de grande importância na definição dos tipos de análises a serem realizadas pelo BemCracker2D. Sendo essas, análise por MEC padrão (STANDARD BEM), análise sem propagação de trinca (WITH NO CRACKS GROWTH) e análise com propagação de trinca (WITH CRACKS GROWTH). Uma representação hierárquica desses objetos pode ser visualizada na Figura 4.16.

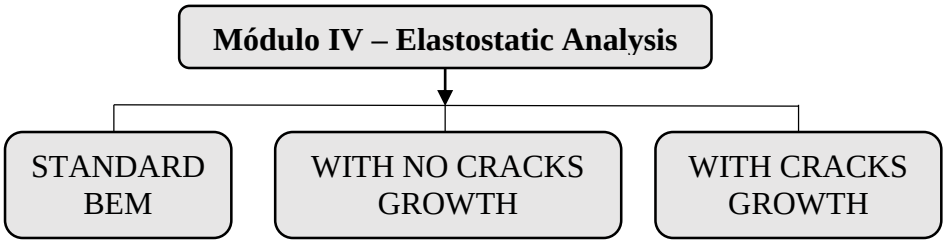


Fig. 4.16 – Representação hierárquica do Módulo IV – Elastostatic Analysis

A função *run_analysis* iniciasse verificando se foi dado inicialmente ao modelo as informações do "Título do Problema" e o "Tipo do Problema" (Fig.3.24), as quais podem ser inseridas a qualquer momento a partir do "*PushButton*" nomeado na área principal do BEMLAB2D por "*Data*". Se não for identificado essas informações, serão abertas caixas de diálogos neste momento para inserir as informações desses dados, como pode ser visto no fragmento da rotina a seguir.

```
function run_analysis_Callback(hObject, eventdata, handles)
%1 - PROBLEMA, PARAMETROS DE CONTROLE E PROPRIEDADES DE MATERIAL:
%TÍTULO DO PROBLEMA
tit_probl = get(handles.text6, 'String');
%CASO DO PROBLEMA
case_probl = get(handles.text7, 'String');
switch tit_probl
    case '0'
        % Título do problema.
        nome = 'Titulo do Problema';
        prompt = {'Titulo do Problema:'};
        numlinhas = 1;
```

```
        resposta = inputdlg(prompt, nome, numlinhas);
        tit_probl = (char(resposta(1)));
        set(handles.text6, 'String' , tit_probl)
        % Tipo de problema: Tensão ou Deformação.
        str = {'Deformação','Tensão'};
        s = listdlg('PromptString','Selecione o tipo de problema:',...
                    'SelectionMode','single','ListSize',[160 31],...
                    'ListString',str);
        switch s
            case 1
                case_probl = 'STRAIN';
            case 2
                case_probl = 'STRESS';
            otherwise
                return
        end
        set(handles.text7, 'String' , case_probl)
end
```

A função *run_analysis* além de ativar a interface auxiliar *CrackGrowth* (Fig. 3.22 e Fig. 3.23), como foi descrito no capítulo 3, possui em seu algoritmo base um adaptação do algoritmo do arquivo *pre_file.m*, citado anteriormente. A principal funcionalidade da função *run_analysis* é organizar as matrizes de dados geradas pelo BEMLAB2D nos módulos II, III e IV. Esses dados são de grande importância para gerar o arquivo de entrada do programa BemCracker2D. Este arquivo é gerado com o nome “arquivo_bem” e tem o formato “.dat”. O arquivo é gerado por uma função externa ao BEMLAB2D, ela é chamada de dentro do script principal pelo comando *output_BEM()*. Um fragmento do algoritmo da função *run_analysis* é apresentado a seguir.

```
%...
%Abre a interface gráfica auxiliar CrackGrowth
f=CrackGrowth;
waitfor(f);
crackgrowth = get(0,'UserData');
if numel(crackgrowth(1,:))==1
    ngaus = crackgrowth(1,1);
else
    ngaus = crackgrowth(1,7);
end
%2 - MATRIZ DE COORDENADAS (m_inc), MATRIZ DE CONECTIVIDADE (m_iel)
m_inc = get(handles.run_mesh,'UserData');
m_iel = get(handles.radiobutton8,'UserData');
%3 - DADOS DE C.C DESLOCAMENTOS, TRAÇOES E PROPAGAÇAO DE TRINCA (CRP):
%A - DADOS DE CONDIÇOES DE CONTORNO DESLOCAMENTOS
displac = get(handles.displacement,'UserData');
%B - DADOS DE CONDIÇOES DE CONTORNO TRAÇOES
tract = get(handles.traction,'UserData');
%C - DADOS DE PROPAGAÇAO DE TRINCA
crackgrowth = crackgrowth';
if numel(crackgrowth(:,1))==1
    v_crp = 0;
else
    v_crp = crackgrowth(1:6)'; %[ncri nadv ppc ppn srat mres]
```

```
end
%Montando arquivo modelo de elemento de contorno
fout=fopen('arquivo_bem.dat','w');% open file output
% call output
output_BEM(fout,tit_probl,case_probl,young,rnu,nnod,nnel,ngaus,num_ipt,…
           rmaxl,xx1,yy1,m_iel,x_int,y_int,v_ccd,ielemd,inoded,dird,…
           disp,v_cct,ielemt,inodet,idirt,trac,v_crp);
fclose(fout)
```

Um modelo do arquivo de dados é apresentado a seguir.

arquivo_bem.dat - Bloco de notas

```
Chapa Quadrada com Propagação de Trinca
STRESS
210000.00 0.30
120 60 0 15.00 10
Nodal_Coordinates_(NODE,X,Y)
1  5.000000  1.000000
2  4.600000  1.000000
3  4.200000  1.000000
4  3.800000  1.000000
5  3.400000  1.000000
...
Mesh_Topology_(ELEMENT,G-NODE1,G-NODE2,G-NODE3)
1 1 2 3
2 3 4 5
3 5 6 7
4 7 8 9
5 9 10 11
...
Displacement_Boundary_Conditions_(ELEMENT,L-NODE,G-NODE)
0 3 0
4 3 1 0.00
...
Traction_Boundary_Conditions_(ELEMENT,L-NODE,G-NODE)
6 0 0
33 1 -1.00
34 1 -1.00
...
Crack_Propagation_(Number_OF_Crack-Extension_Increments)
14
2
8.5900e-13 3.3000 0.6667
4
```

Vale ressaltar que o BEMLAB2D não é responsável pelas análises, a qual é de responsabilidade do BemCracker2D. A principal função do módulo IV é reunir e organizar as informações necessárias de entrada do programa de análise.

4.2.5 - Módulo V – Graphical Results

O módulo V aciona a interface gráfica auxiliar *GraphicalResults* (Fig. 3.26). Toda a implementação do pós-processamento realizada pelo módulo V se encontra no *script* da interface auxiliar, onde cada função inserida representa um tipo de resultado diferente. Este módulo está em fase de construção, portanto no exato momento esse trabalho apenas pode

apresentar graficamente na área gráfica da interface auxiliar a malha inicial, a malha deformada, a malha indeformada e tensões principais, o gráfico do Fator de Intensidade de Tensão, a propagação de trinca e a resistência residual.

O programa responsável por gerar os resultados e visualização das malhas de malha inicial, malha deformada e malha indeformada com tensões principais é o *mecmalha.m*. Este programa lê o arquivo de dados gerado pelo programa BemCracker2D (GOMES, 2016) chamado de *meshI.m*. O programa *mecmalha.m*, como pode ser visualizado no fragmento do algoritmo a seguir, chama em suas linhas de comando, funções que servem para visualizar as malhas devido a seleção realizada com os *ToggleButton* na interface auxiliar.

```
%Programa para geração da Malha e Visualizacao de resultados
meshI %chamando arquivo resultados
%Executando a visualizacao escolhida
switch viwer
    case 0
        % Malha Inicial Padrão MEC
        sit = MalhaIni(nnel, nitp, titl, XY, P_INT, MIEL);
    case 1
        % Malha Deformada
        sit = MalhaDef(titl, nnod, nnel, nitp, XY, P_INT, MIEL, SOLDI,
SOLDC);
    case 2
        % Malha Indeformada e Tensões Principais
        sit = MalhaSIGi(titl, nnel, nitp, XY, P_INT, MIEL, SOLTI, SOLTC);
    case 3
    otherwise
        disp('Opcao Invalida')
end
```

As funções *MalhaIni*, *MalhaDef* e *MalhaSIGi* são as que ligam os *ToggleButton* na interface *GraphicalResults* (Fig. 3.26) com a área gráfica e construindo as malhas para a visualização e interpretação dos resultados gráficos pelo usuário. A função *MalhaIni* é responsável por montar a malha inicial, a função *MalhaDef* gera a malha deformada e *MalhaSIGi* gera a malha indeformada com as tensões principais.

O programa gerador do caminho da trinca é chamado de *MalhaCaTr*, este programa lê os arquivos *camtr.dat* gerado pelo BemCracker2D. Este arquivo é o histórico de propagação da trinca com relação ao números de incremento definidos no BEMLAB2D, através da interface *CrackGrowth* (Fig. 3.23). Uma breve apresentação de um fragmento do programa *MalhaCaTr* é apresentado a seguir.

```
%gerando e plotando nos de contorno
figure('Name','PLOTANDO CAMINHO DA PROPAGAÇAO');
pos = get(gcf, 'Position');
```

```
for i=1:nn(2)
    for j=1:3
        k = IEL(i,j+1);
        Cx(j,1) = XY_C(k,2);
        Cy(j,1) = XY_C(k,3);
    end
    plot(Cx,Cy,'-k.')
    hold on
end
for i=1:nincr
    for ii=nni(2):nni(4)
        for j=1:3
            k = IEL(ii,j+1);
            Cx(j,1) = XY_C(k,2);
            Cy(j,1) = XY_C(k,3);
        end
        plot(Cx,Cy,'-k.')
        hold on
    end
    kk=1;
    while(kk<=nn(3))
        plot(XY_PT(kk,2),XY_PT(kk,3),'*b')
        hold on
        plot(XY_PT(kk,4),XY_PT(kk,5),'*r')
        kk=kk+1;
    end
    pause(3)
end
```

GrafTrin.m é o programa responsável por gerar os resultados e a visualização de gráficos. Este programa lê os arquivos de dados gerados pelo programa BemCracker2D (GOMES, 2016) denominados *sifs.dat* e *rrncc.dat*, os quais devolvem resultados de Fatores de Intensidade de Tensão e Resistência Residual, respectivamente. O programa *GrafTrin.m* pode ser visualizado no fragmento do algoritmo a seguir.

```
%Plotagem de graficos de anaise de trincas
switch opgr %Opção de seleção definida na interface GraphicalResults
    case 1
        figure('Name','PLOTANDO SIF x INCREMETOS');
        fin=fopen('sif1.dat','rt');%abrindo arquivo
        for i=1:nincr
            for j=1:tr
                %ntr=fscanf(fin,'%d',1);%numero da trinca
                auxres=fscanf(fin,'%f',[4,j]);auxres=auxres';
                INC(i,1)=auxres(1,1);
                SIF(i,k)=auxres(1,2);
                SIF(i,k+1)=auxres(1,3);
                k=k+2;
            end
            k=1;
        end
        plot(INC(:,1),SIF(:,1),'-ob',INC(:,1),SIF(:,2),'-or')
    case 2
        figure('Name','PLOTANDO RRES x INCREMETOS');
        fin=fopen('resres.dat','rt');%abrindo arquivo
        for i=1:nincr
            for j=1:tr
```

```
                ntr=fscanf(fin,'%d',1);%numero da trinca
                auxres=fscanf(fin,'%f',[3,j]);auxres=auxres';
                INC(i,1)=auxres(1,1);
                RES(i,k)=auxres(1,2);
                NCL(i,k)=auxres(1,3);
                k=k+1;
            end
            k=1;
        end
        subplot(2,1,1)
        plot(INC(:,1),RES(:,1),'-ob')
        subplot(2,1,2)
        plot(INC(:,1),NCL(:,1),'-ob')
end
```

Os modelos gerados pelas funções citadas acima são construídos na área gráfica da interface *GraphicalResults* (Fig. 3.26) apenas utilizando o comando *plot( )*, alterando poucas características entre cada gráfico. Uma representação hierárquica da funcionalidade detalhada do módulo V pode ser visualizada na Figura 4.17.

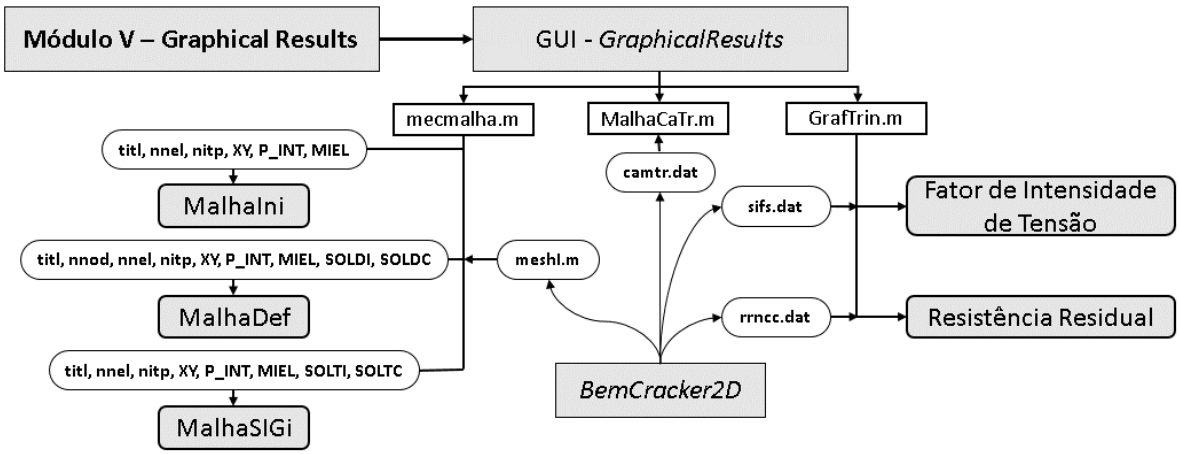


Fig. 4.17 – Representação hierárquica do Módulo V – Graphical Results

5 - EXEMPLOS DE APLICAÇÃO

Neste capítulo serão apresentados alguns exemplos gerados pelo programa BEMLAB2D, com o propósito de demonstrar o seu funcionamento e o desempenho dos módulos que o compões. Os exemplos discutidos neste capítulo referem-se principalmente às características especiais representáveis pelo programa proposto neste trabalho, através de poucos comandos básicos do BEMLAB2D, com o objetivo de conferir e validar a estrutura de dados implementada. Três seções são apresentadas neste capítulo para melhor organização do mesmo, a saber: 1) Pré-processamento, contendo 8 exemplos; 2) Pós-processamento, contendo 7 exemplos; e 3) Gerador de Malhas MEF/MESHLESS, com 2 exemplos.

Para as duas primeiras seções será feita uma aplicação em elasticidade bidimensional de Elementos de Contorno via programa BemCracker2D, responsável pela análise numérica no BEMLAB2D, cujo objetivo é de tão somente ilustrar o funcionamento do pré e pós-processamento através dos resultados já consolidados e atestados pelo BemCracker2D. por fim, a última seção trata apenas da geração de malhas de MEF e MESHLESS.

5.1 - PRÉ-PROCESSAMENTO

Esta seção se refere aos quatro primeiros módulos que compões o BEMLAB2D. Será descrito aqui, em ordem de execução do programa, cada etapa do pré-processamento. A Figura 5.1 a seguir mostra onde se encontram os quatro módulos citados anteriormente na interface gráfica.

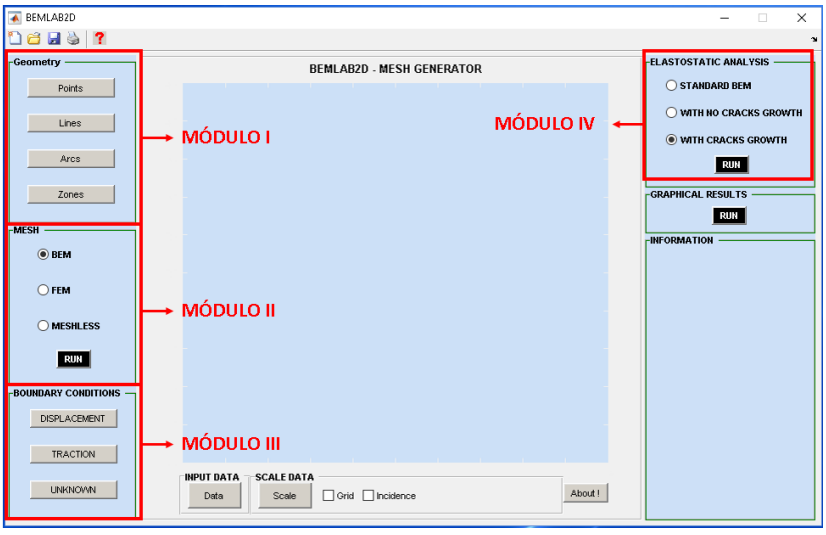


Fig. 5.1 – BEMLAB2D – Módulos I, II, III e IV

5.1.1 - Exemplo 1 – Cavidade Circular em um Meio Infinito

Na figura 5.2, tem-se o modelo geométrico de uma cavidade circular em meio infinito com diâmetro de 200 mm de comprimento. Seu módulo de elasticidade $E = 210\ GPa$ e coeficiente de Poisson $\nu = 0.1$ (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

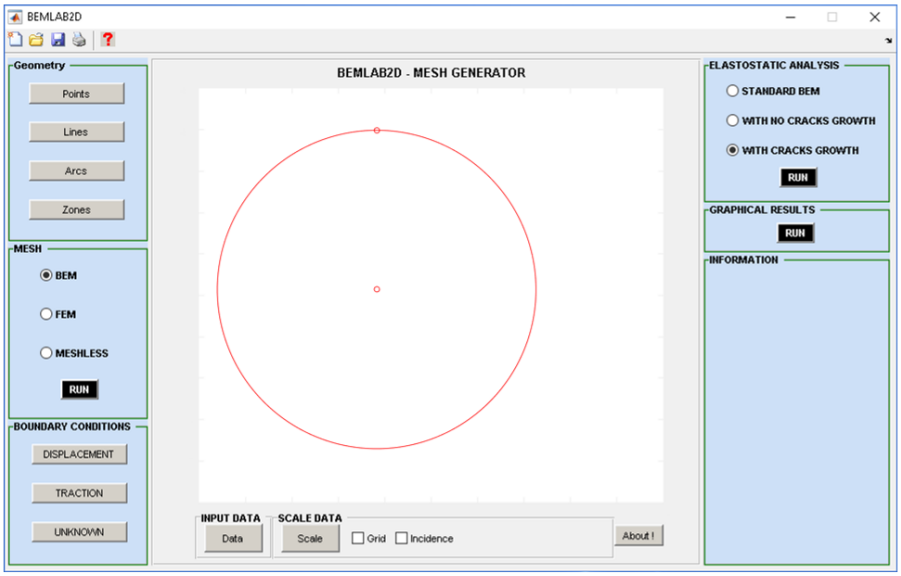


Fig. 5.2 – Modelo geométrico para o exemplo 1

Para este modelo foram usados 12 elementos quadráticos e 2 pontos internos (Módulo II), como pode-se ver na malha de MEC em 5.3.

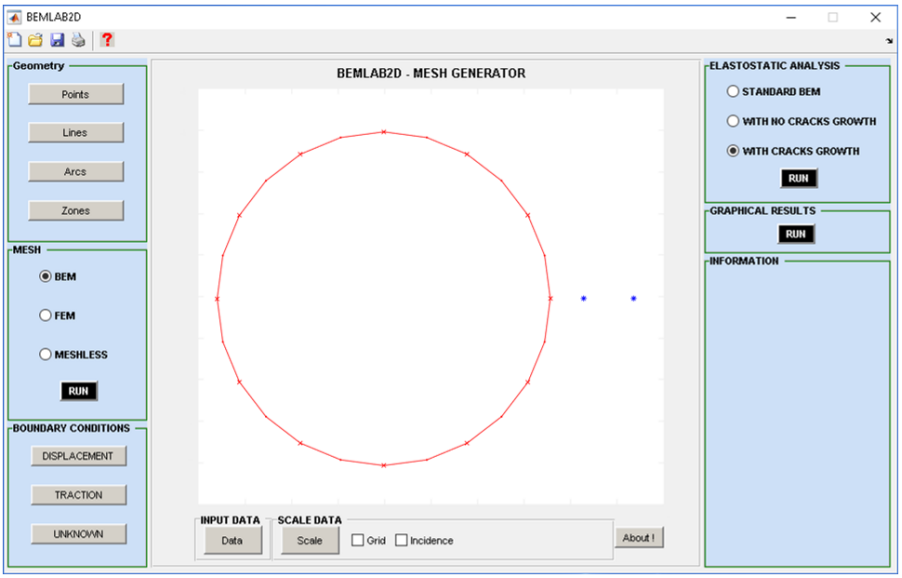


Fig. 5.3 – Malha de MEC para o exemplo 1

A cavidade é submetida a um carregamento distribuído uniformemente de 15 MPa sobre todos os elementos na direção normal e restrita ao deslocamento em apenas um nó de três elementos distintos (Módulo III). Ver Figura 5.4.

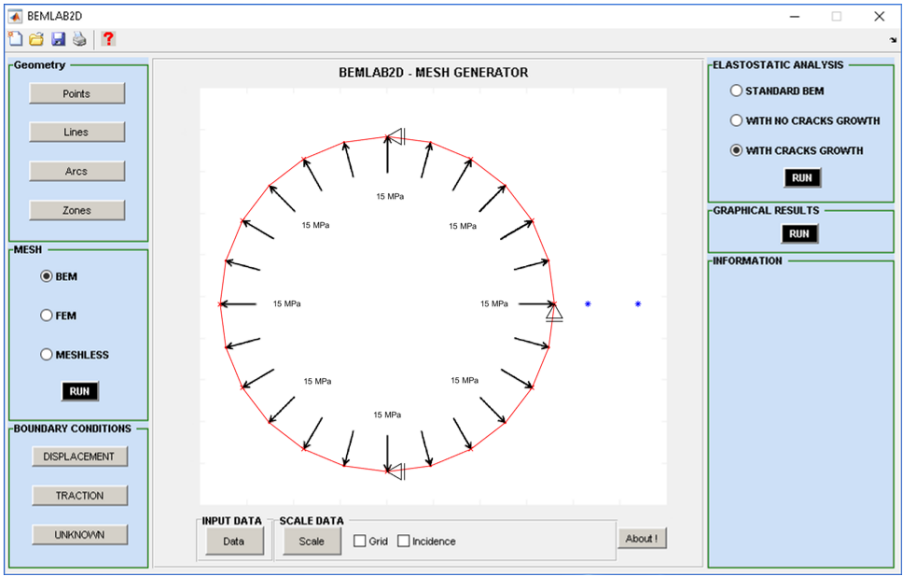


Fig. 5.4 – Modelo físico-geométrico sobre a malha de MEC do exemplo 1

5.1.2 - Exemplo 2 – Fração de um Círculo com um Furo Central

Na figura 5.5, tem-se uma fração de círculo vazado com raio externo de 250 mm de comprimento e raio interno de 100 mm de comprimento. Seu módulo de elasticidade longitudinal $E = 200\ GPa$ e coeficiente de Poisson $\nu = 0.25$ (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

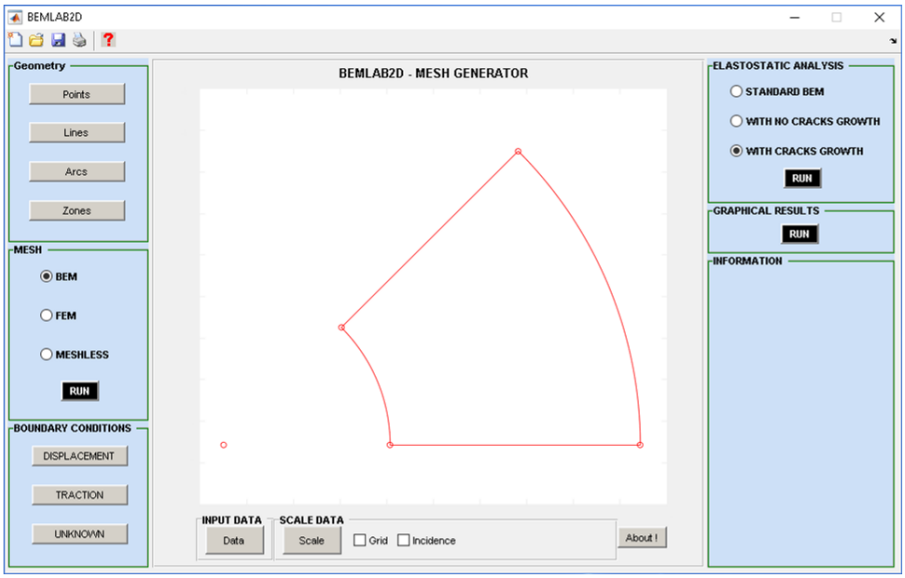


Fig. 5.5 – Modelo geométrico para o exemplo 2

Para este modelo foram usados 10 elementos quadráticos e nenhum pontos internos (Módulo II), como pode-se ver na malha de MEC em 5.6.

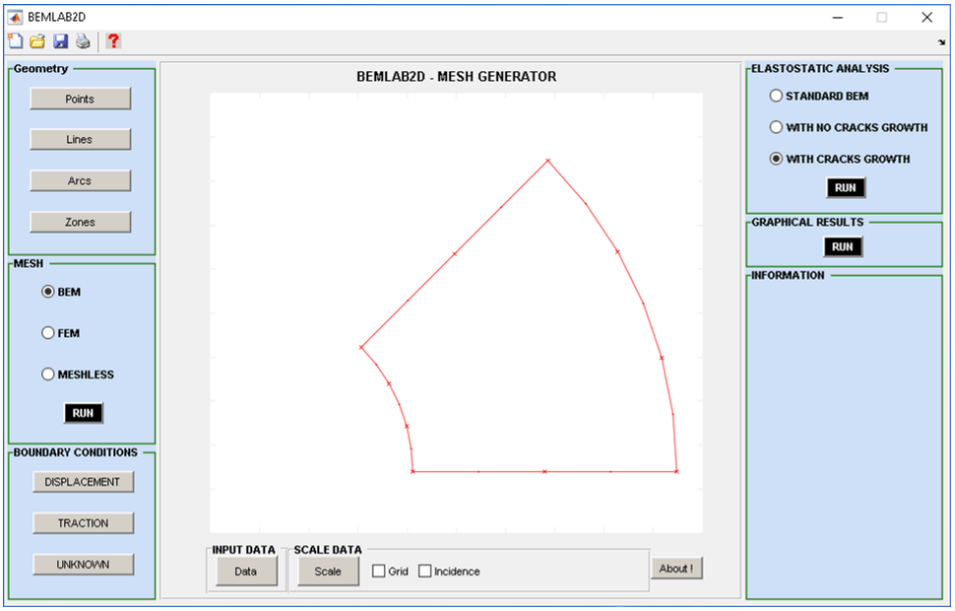


Fig. 5.6 – Malha de MEC para o exemplo 2

A fração de círculo vazado é submetido a um carregamento distribuído uniformemente de 100 MPa na direção normal sobre os elementos do contorno interno do anel e restrita na direção normal as faces retas do modelo (Módulo III). Ver Figura 5.7.

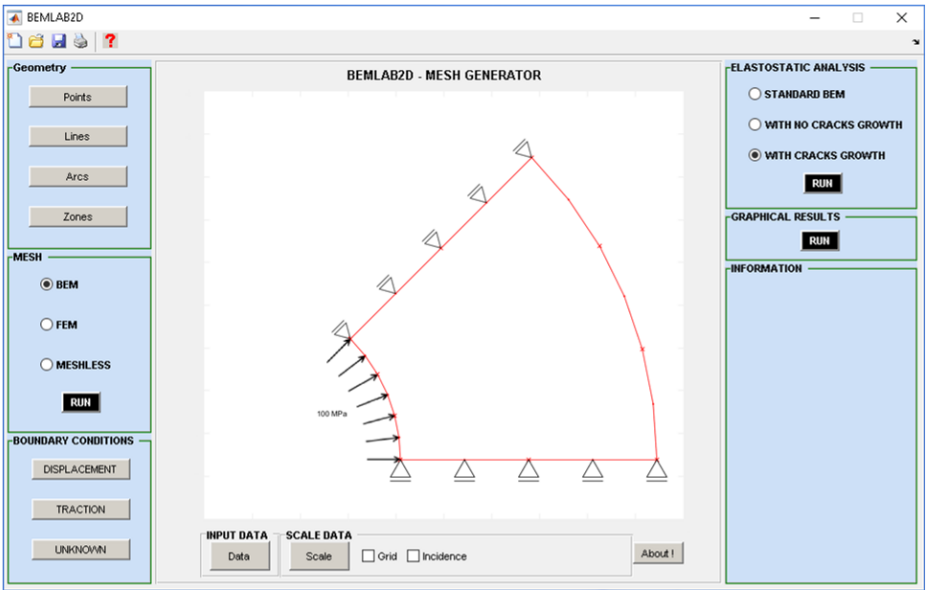


Fig. 5.7 – Modelo físico-geométrico sobre a malha de MEC do exemplo 2

5.1.3 - Exemplo 3 – Chapa com Furo Tracionada (Arrancamento)

Na figura 5.8, tem-se o modelo geométrico de uma chapa alongada com três faces retas ortogonais e uma face curva (100 mm de altura e 50 mm de largura), com um furo próximo a face circular (r = 10 mm). Seu módulo de elasticidade $E = 210\ GPa$ e coeficiente de Poisson ν = 0.3 (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

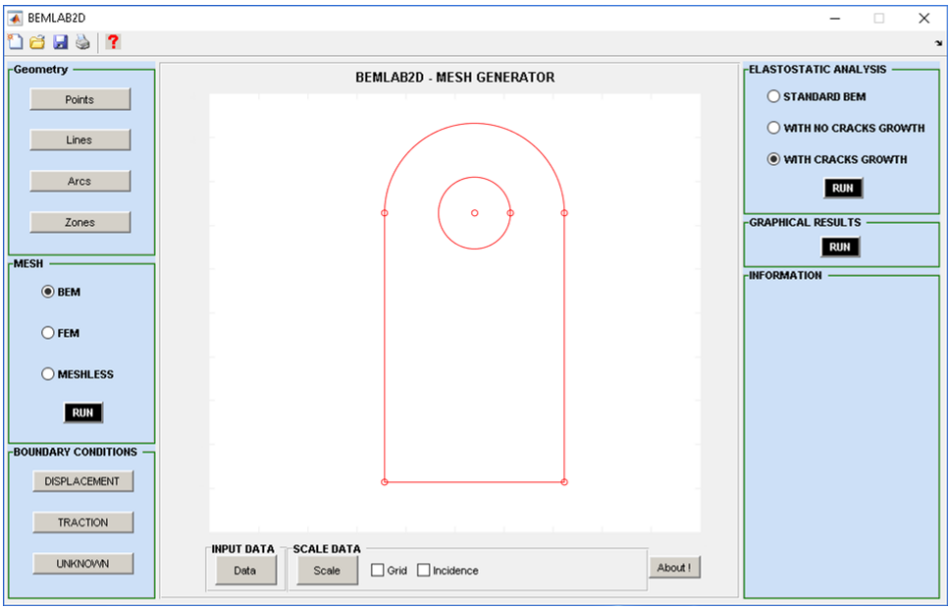


Fig. 5.8 – Modelo geométrico para o exemplo 3

Para este modelo foram usados 74 elementos e 30 pontos internos (Módulo II), como pode-se ver na malha de MEC em 5.9.

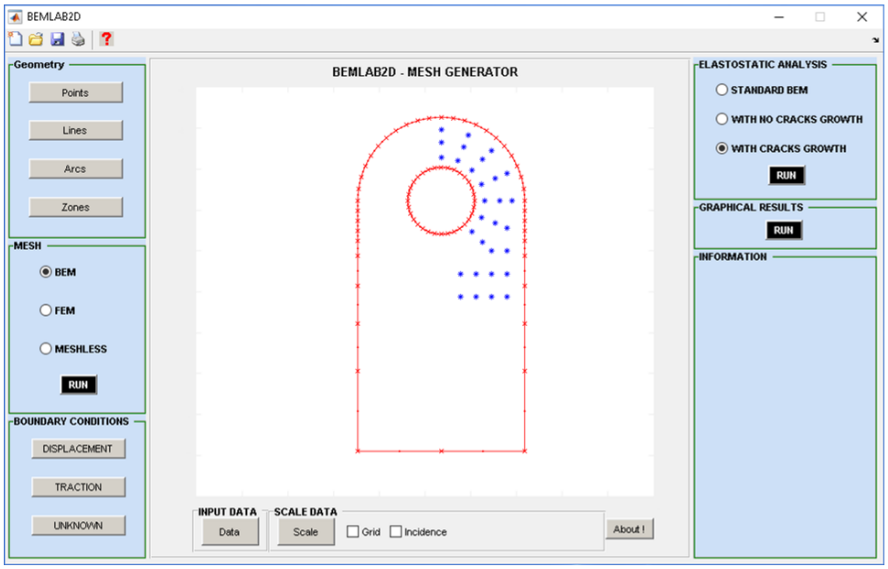


Fig. 5.9 – Malha de MEC para o exemplo 3

A chapa com furo é submetido a um carregamento de arranchamento de 104.53 MPa atuando na direção normal sobre parte dos elementos do furo (Módulo III), como pode ser visualizado na Figura 5.10.

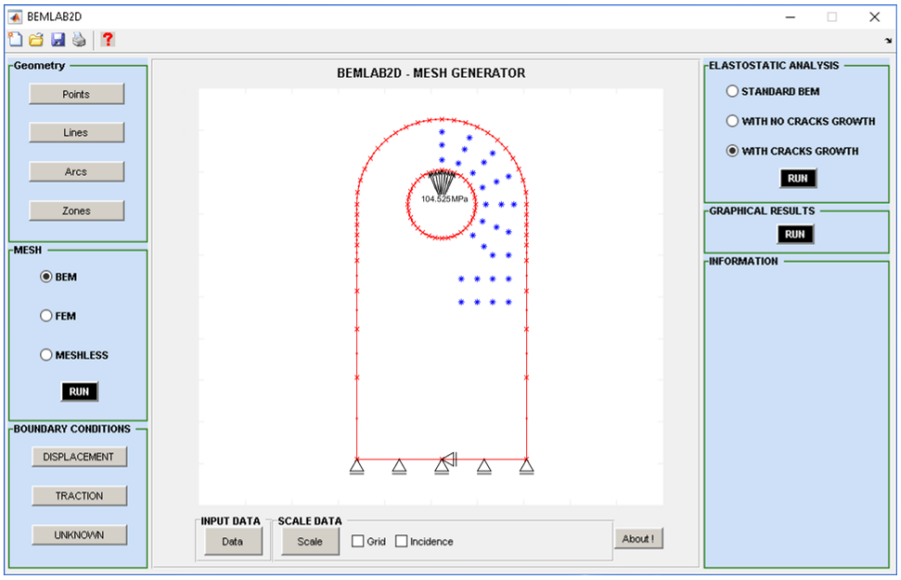


Fig. 5.10 – Modelo físico-geométrico sobre a malha de MEC do exemplo 3

5.1.4 - Exemplo 4 – Chapa Retangular com Trinca Central Inclinada

Na figura 5.11, tem-se o modelo geométrico de uma chapa retangular (40 cm de altura e 20 cm de largura), com uma trinca inclinada de 12 cm de comprimento. Seu módulo de elasticidade $E = 200\ GPa$ e coeficiente de Poisson $\nu = 0.25$ (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

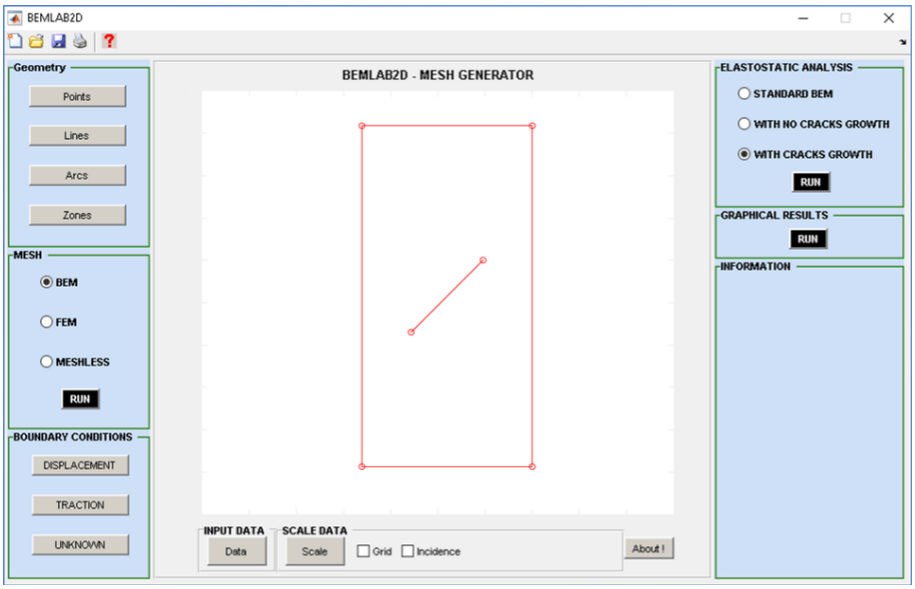


Fig. 5.11 – Modelo geométrico para o exemplo 4

Para este modelo foram usados 46 elementos quadráticos, dos quais 16 são descontínuos e pertencentes à trinca (Módulo II), como pode-se ver na malha de MEC em 5.12.

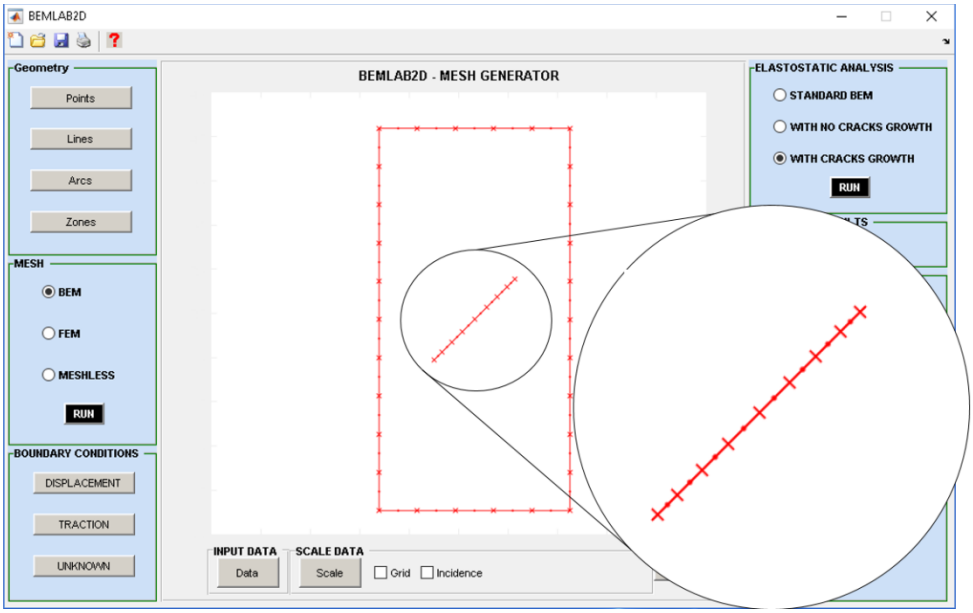


Fig. 5.12 – Malha de MEC com detalhe da trinca para o exemplo 4

A chapa retangular é submetido a um carregamento uniforme de 10 MPa tracionando na direção da altura da peça (Módulo III), como pode ser visualizado na Figura 5.13.

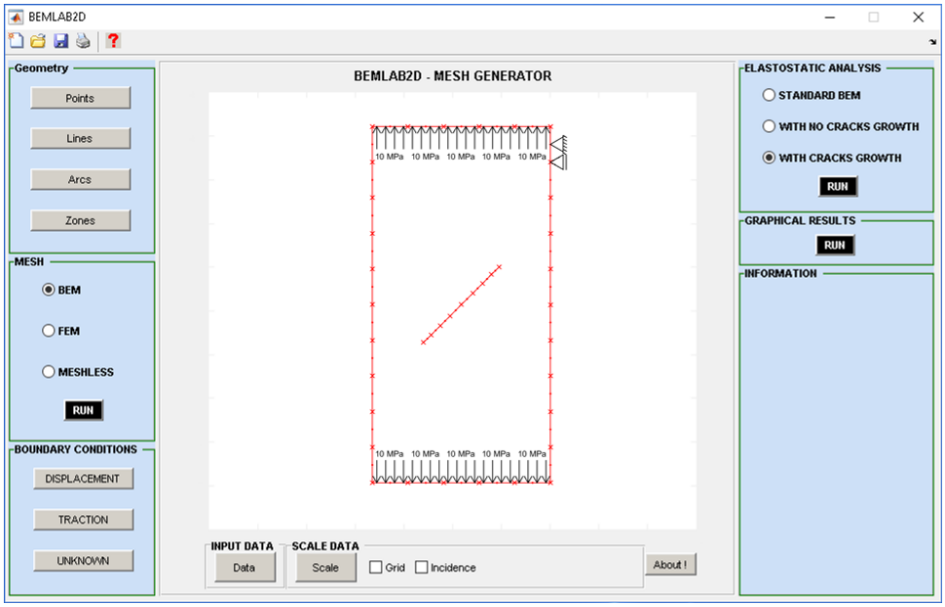


Fig. 5.13 – Modelo físico-geométrico sobre a malha de MEC do exemplo 4

5.1.5 - Exemplo 5 – Chapa Retangular com Três Trincas de Borda

Na figura 5.14, tem-se uma chapa retangular (40 cm de altura e 20 cm de largura), com três trinca de borda onde duas medem 6 cm de comprimento e uma 8 cm de comprimento. Seu módulo de elasticidade $E = 200\ GPa$ e coeficiente de Poisson $\nu = 0.25$ (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

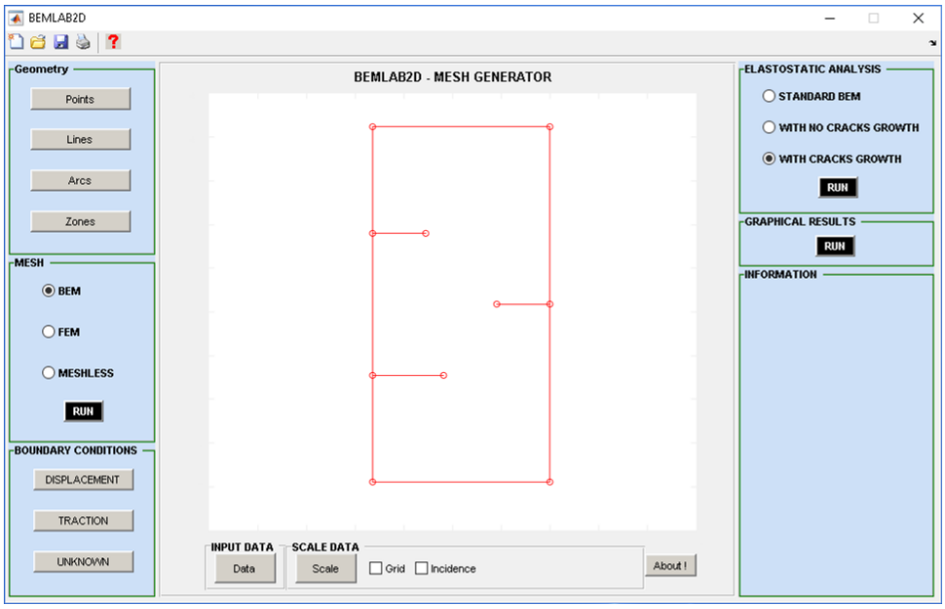


Fig. 5.14 – Modelo geométrico para o exemplo 5

Para este modelo foram usados 54 elementos quadráticos, sendo 8 elementos descontínuos por trinca (Módulo II), como pode-se ver na malha de MEC em 5.15.

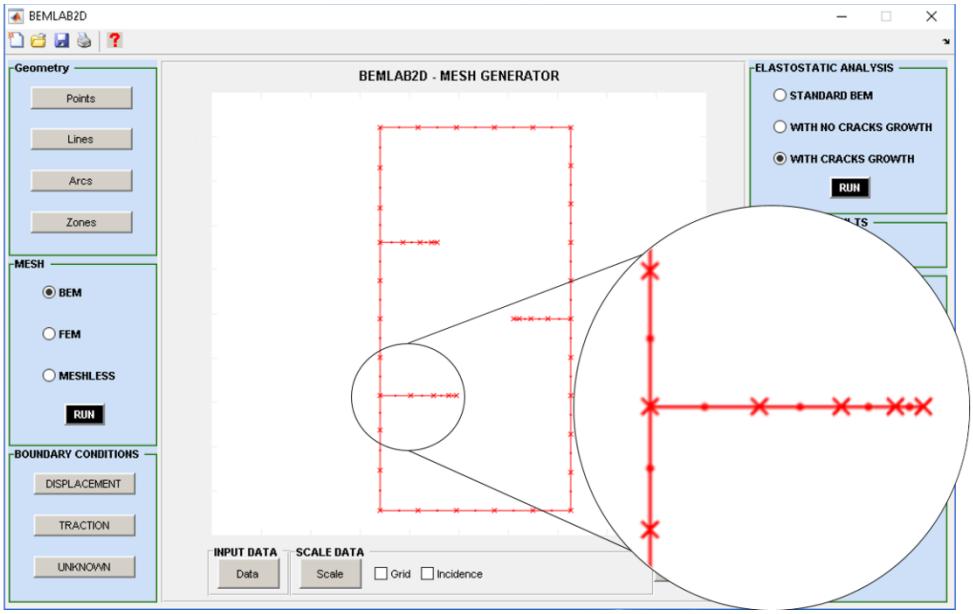


Fig. 5.15 – Malha de MEC com detalhe da trinca para o exemplo 5

A chapa retangular está submetido a um carregamento uniforme distribuído de 10 MPa tracionando na maior direção da peça (Módulo III), como pode ser visualizado na Figura 5.16.

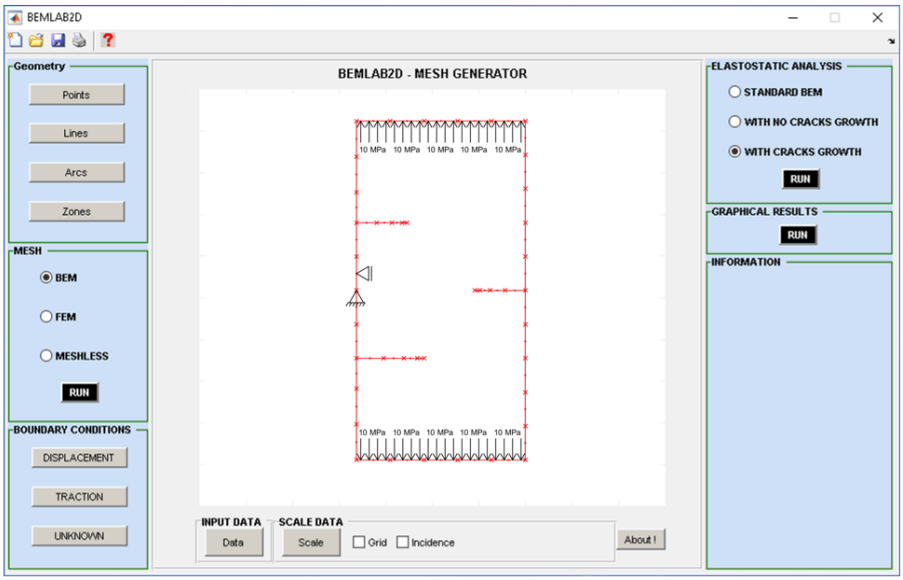


Fig. 5.16 – Modelo físico-geométrico sobre a malha de MEC do exemplo 5

5.1.6 - Exemplo 6 – Chapa Cruciforme com Trinca

Neste exemplo tem-se o modelo geométrico de uma chapa cruciforme, como pode ser visualizada na figura 5.17, com uma trinca medindo 4 mm de comprimento. Seu módulo de elasticidade $E = 200\ GPa$ e coeficiente de Poisson $\nu = 0.25$ (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

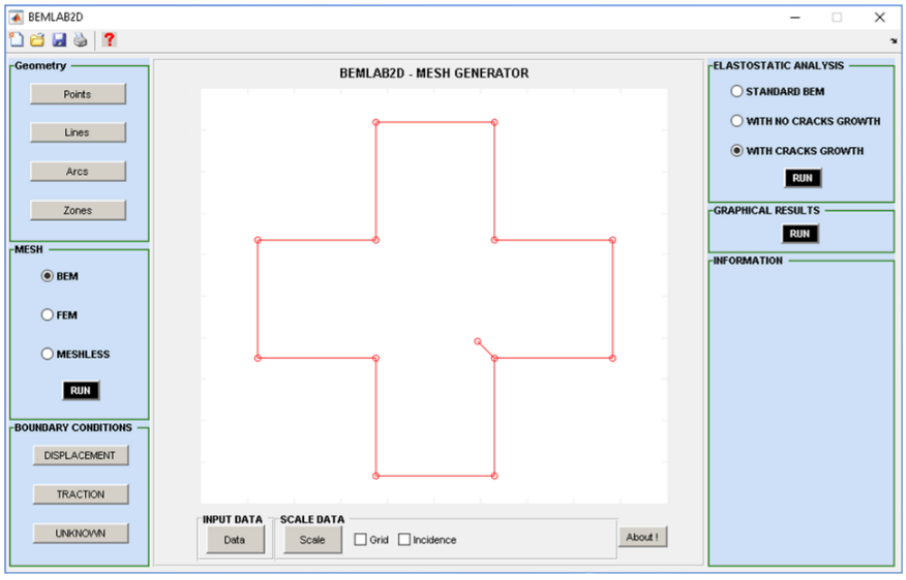


Fig. 5.17 – Modelo geométrico para o exemplo 6

Para este modelo foram usados 56 elementos, sendo 8 elementos descontínuos na trinca (Módulo II), como pode-se ver na malha de MEC em 5.18.

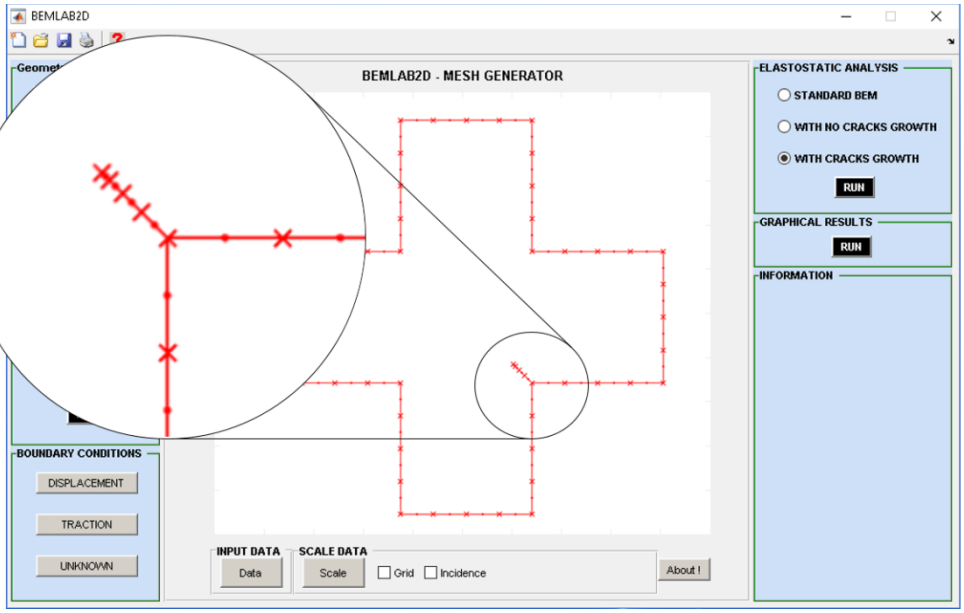


Fig. 5.18 – Malha de MEC com detalhe da trinca para o exemplo 6

A chapa cruciforme está submetido a um carregamento uniforme distribuído de 20 MPa tracionando todas as quatro extremidades da chapa (Módulo III), como pode ser visualizado na Figura 5.19.

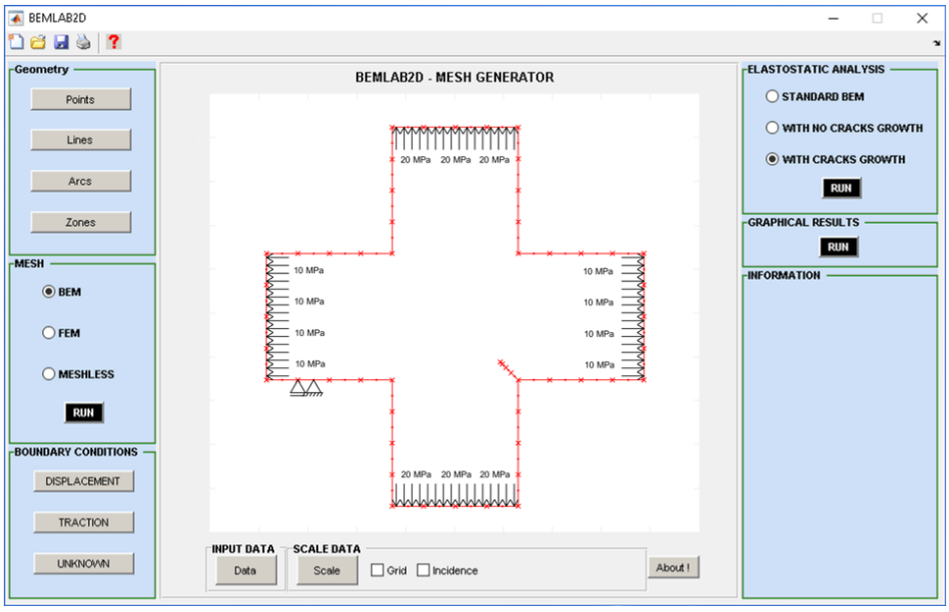


Fig. 5.19 – Modelo físico-geométrico sobre a malha de MEC do exemplo 6

5.1.7 - Exemplo 7 – Chapa Retangular com Furos e Propagação de Trinca

Na figura 5.20, tem-se o modelo geométrico de uma chapa retangular (200 mm de altura e 300 mm de largura), com três furos de fixação não-colineares com diâmetro de 20 mm unidades de comprimento e uma trinca projetada sobre a borda de um dos furos. Seu módulo de elasticidade $E = 200\ GPa$ e coeficiente de Poisson $\nu = 0.25$ (Fig. 3.9), estando o mesmo sujeito a um estado plano de tensão (Módulo I).

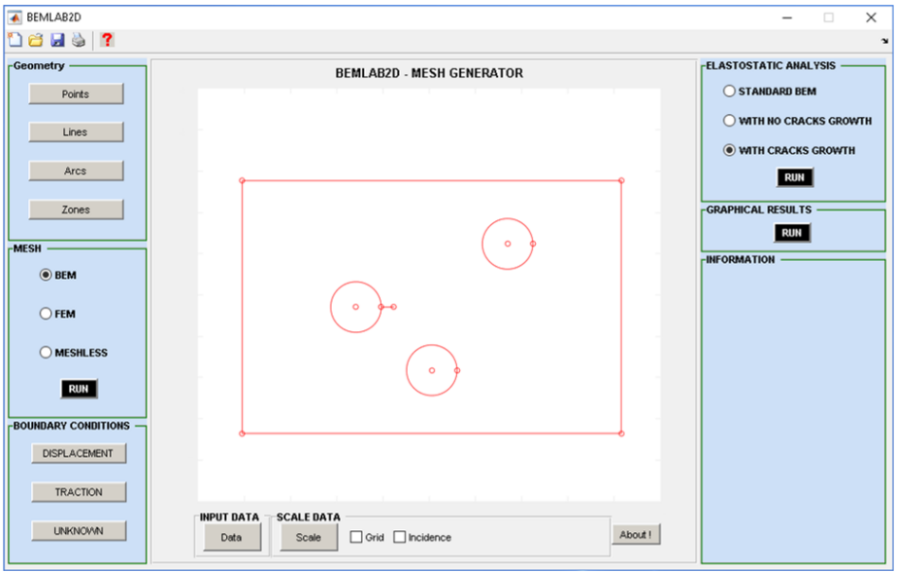


Fig. 5.20 – Modelo geométrico para o exemplo 7

Para este modelo foram usados 72 elementos quadráticos, sendo 8 elementos descontínuos na trinca (Módulo II), como pode-se ver na malha de MEC em 5.21.

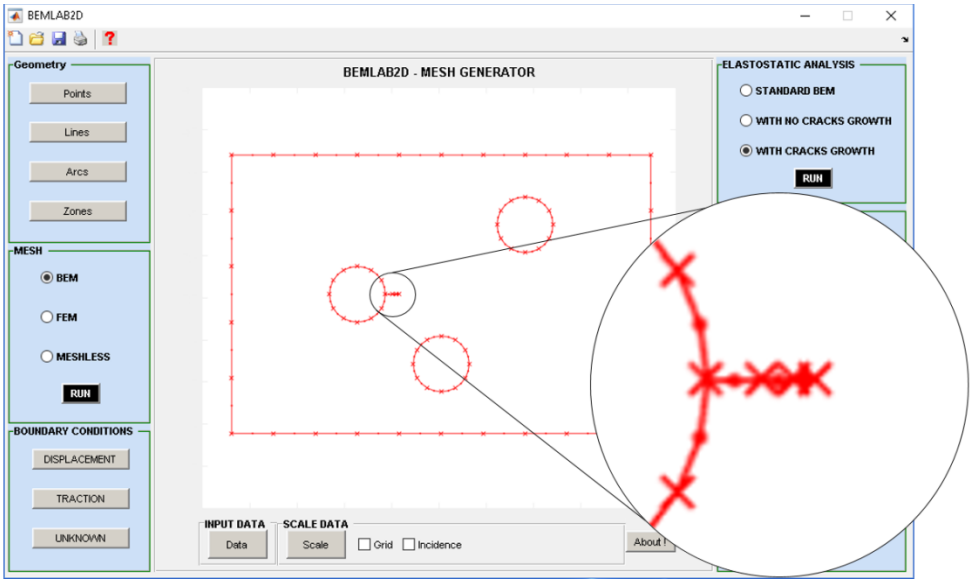


Fig. 5.21 – Malha de MEC com detalhe da trinca para o exemplo 7

A chapa retangular está submetido a um carregamento uniforme distribuído de 10 MPa tracionando a menor direção da peça (Módulo III), como pode ser visualizado na Figura 5.22.

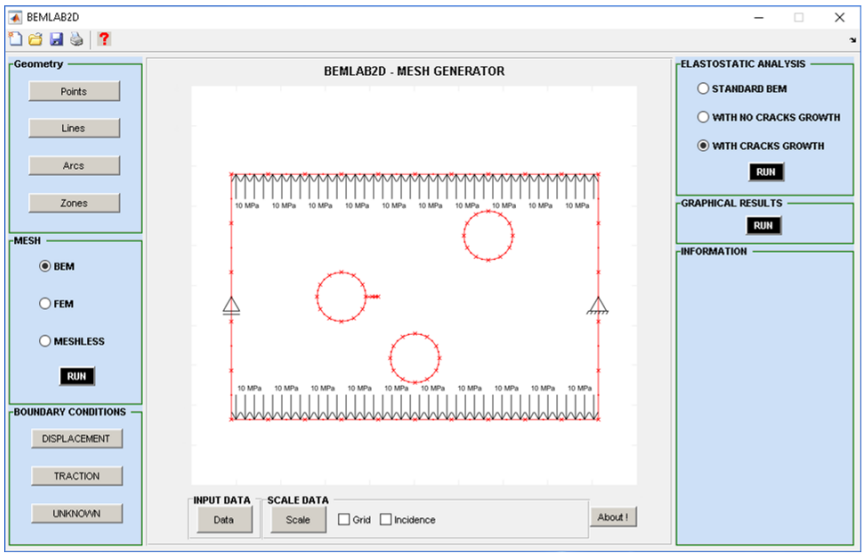


Fig. 5.22 – Modelo físico-geométrico sobre a malha de MEC do exemplo 7

5.2 - PÓS-PROCESSAMENTO

Nesta seção será abordada o último módulo que compõe o BEMLAB2D, onde será apresentado alguns resultados obtidos pelo BemCracker2D e visualizados na área gráfica da interface auxiliar *GraphicalResults*, o pós-processador do BEMLAB2D. A Figura 5.23 a seguir mostra o último módulo citado anteriormente e a interface auxiliar de pós-processamento.

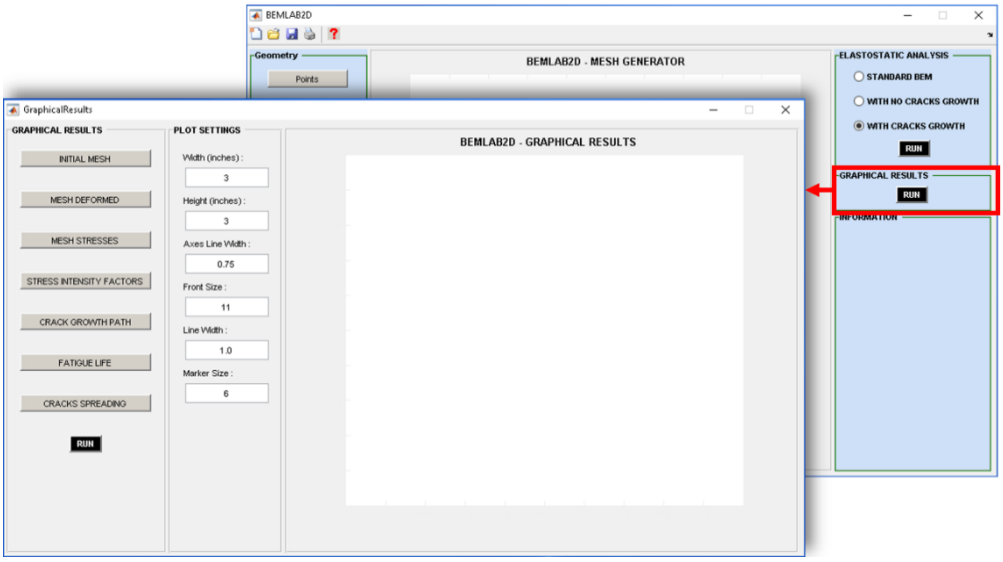


Fig. 5.23 – BEMLAB2D – Módulos V e interface de pós-processamento

5.2.1 - Exemplo 1 – Cavidade Circular Sobre Compressão Radial

Como foi visto no subitem 5.1.1 o modelo estudado é uma cavidade circular sofrendo uma compressão radial em meio infinito, sua análise foi realizada pelo MEC padrão. A Figura 5.24 apresenta a malha deformada em um comparativo com a malha inicial indeformada.

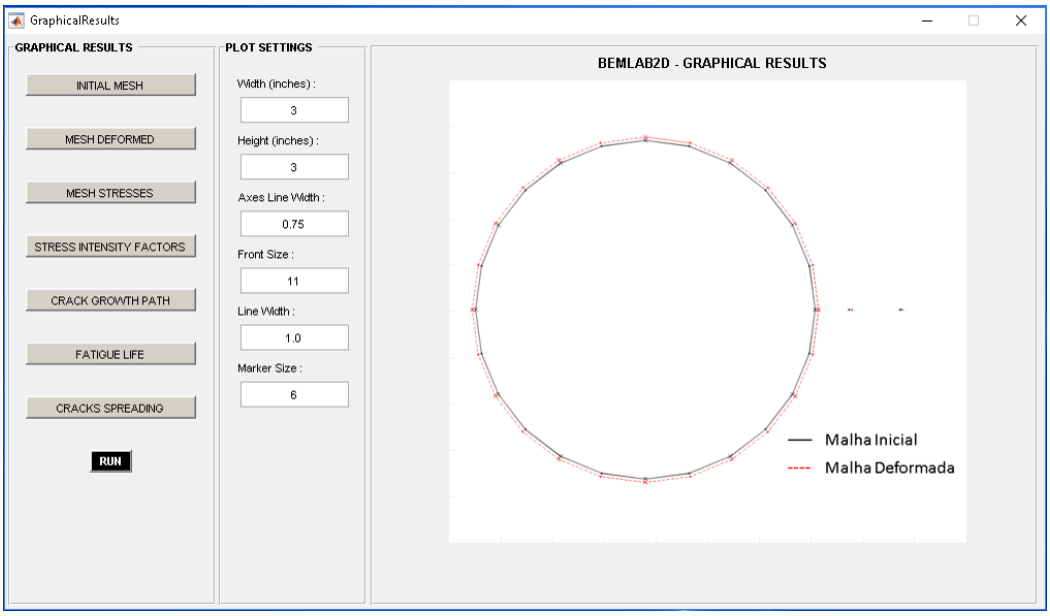


Fig. 5.24 – Malha deformada do exemplo 1

Os resultados das tensões principais obtidos para a cavidade circular sobre compressão radial são apresentados na Figura 5.25, onde S1 e S2 são, respectivamente, tensões principais 1 e 2, podendo ser de tração (T) ou compressão (C).

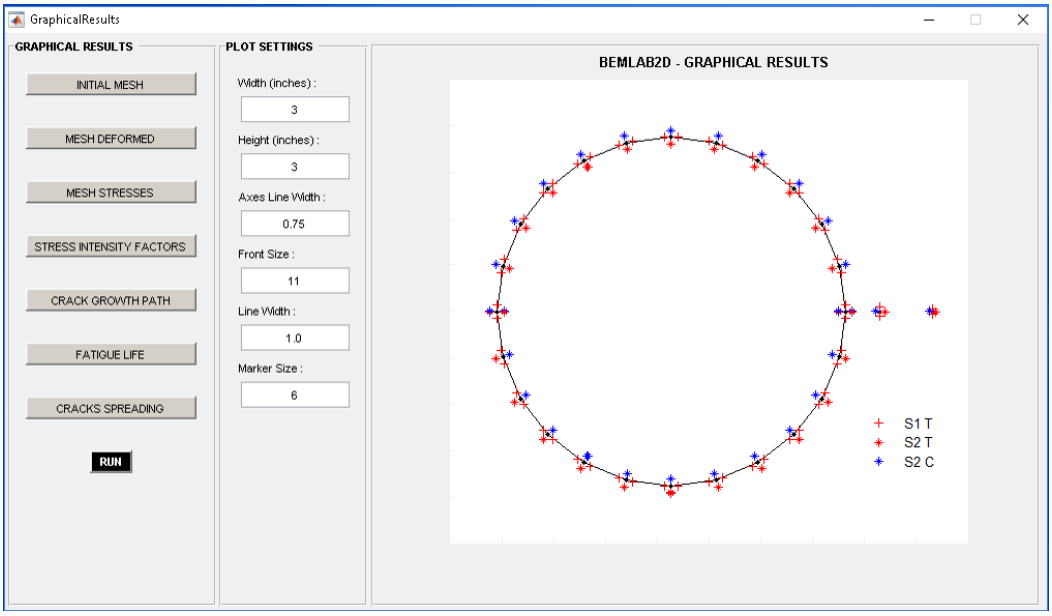


Fig. 5.25 – Tensões principais do exemplo 1

5.2.2 - Exemplo 2 – Fração de um Círculo com um Furo Central Estendido

No subitem 5.1.2 o exemplo abordado é o modelo de uma fração de anel sobre cargas uniformes provocando sua distensão, a análise realizada neste exemplo foi pelo MEC padrão. Um comparativo da malha deformada com a malha inicial indeformada é apresentada na Figura 5.26.

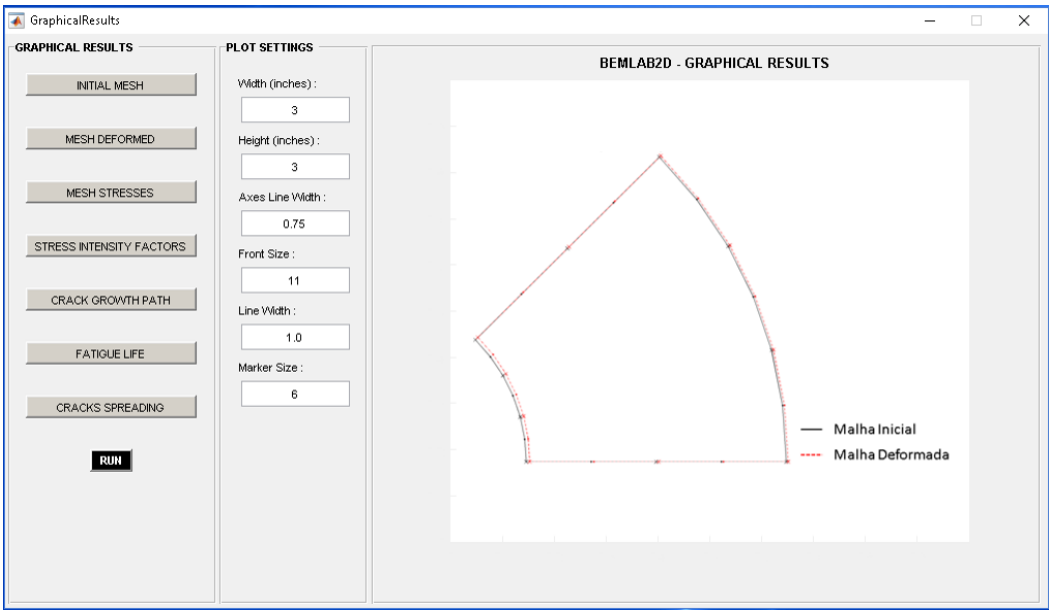


Fig. 5.26 – Malha deformada do exemplo 2

Na Figura 5.27 são apresentados os resultados da tensões principais obtidos para a fração de um círculo com furo central estendido. S1 e S2 são, respectivamente, tensões principais 1 e 2, podendo ser de tração (T) ou compressão (C).

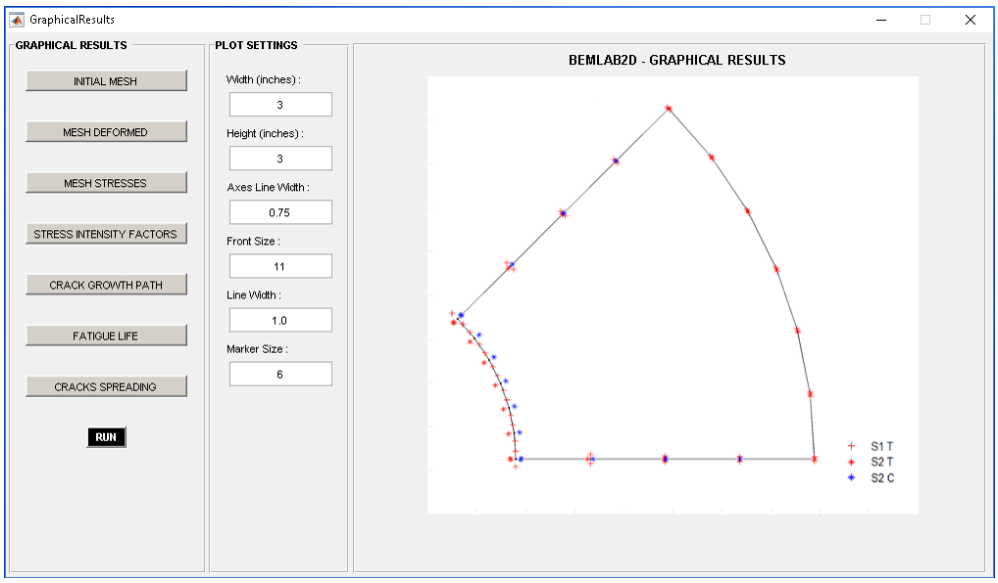


Fig. 5.27 – Tensões principais do exemplo 2

5.2.3 - Exemplo 3 – Chapa com Furo Tracionada (Arrancamento)

Como foi visto no subitem 5.1.3, uma chapa com furo tracionada é abordada neste exemplo, uma análise por MEC padrão é realizada para este modelo. A malha deformada é apresentada juntamente com a malha inicialmente indeformada na Figura 5.28.

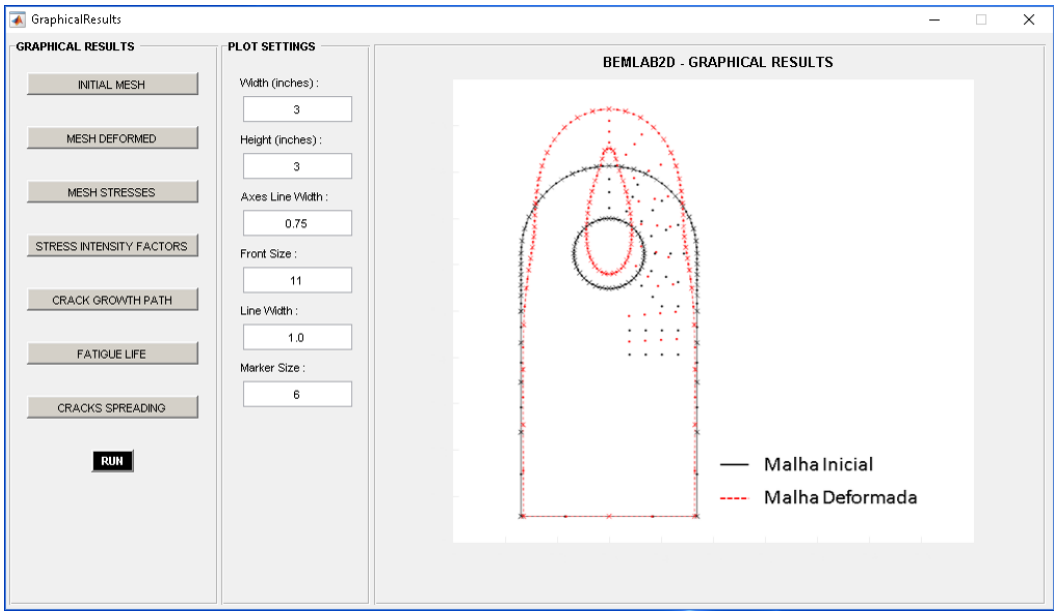


Fig. 5.28 – Malha deformada do exemplo 3

5.2.4 - Exemplo 4 – Chapa Retangular com Trinca Central Inclinada

Como foi visto no subitem 5.1.4 o modelo estudado é uma chapa retangular com trinca inclinada onde o modelo sofre tração na sua maior dimensão, sua análise foi realizada pelo MEC sem a consideração da propagação de trinca. A Figura 5.29 apresenta a malha deformada em um comparativo com a malha inicial indeformada.

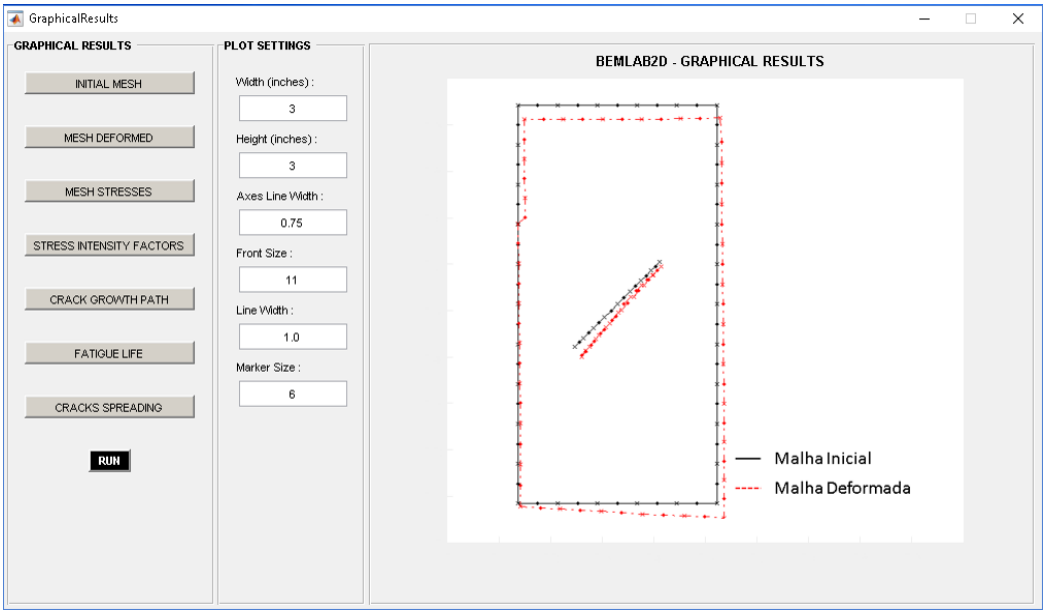


Fig. 5.29 – Malha deformada do exemplo 4

5.2.5 - Exemplo 5 – Chapa Retangular com Três Trincas de Borda

No subitem 5.1.5 o exemplo abordado é o modelo de uma chapa retangular com três trincas de borda, a análise realizada neste exemplo foi pelo MEC sem a consideração da propagação de trinca. Um comparativo da malha deformada com a malha inicial indeformada é apresentada na Figura 5.30.

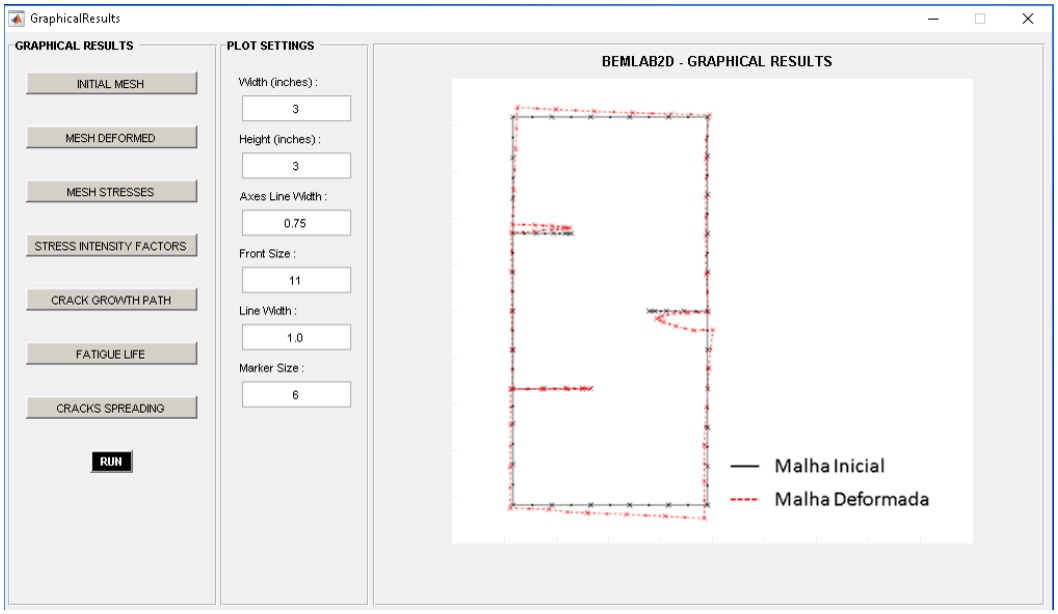


Fig. 5.30 – Malha deformada do exemplo 5

5.2.6 - Exemplo 6 – Chapa Cruciforme com Trinca

Como foi visto no subitem 5.1.6 o modelo estudado é uma chapa cruciforme com trinca propagando do bordo, iniciando-se a partir de um dos vértices internos, sua análise foi realizada pelo MEC Dual considerando a propagação de trinca. A Figura 5.31 apresenta o caminho de propagação da trinca na placa cruciforme.

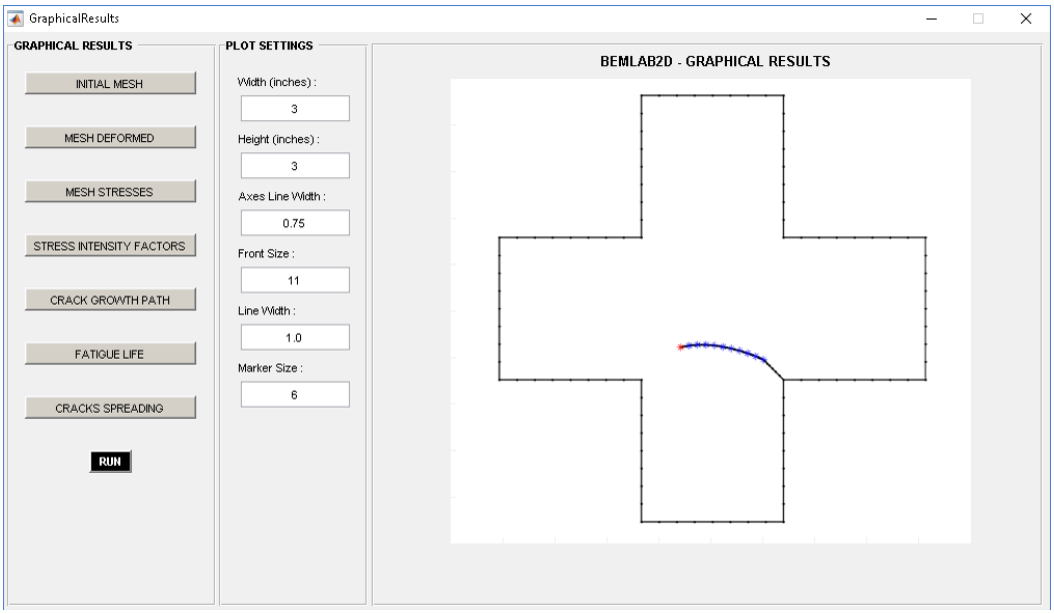


Fig. 5.31 – Caminho de propagação da trinca do exemplo 6

Os gráficos do Fator de Intensidade de Tensão obtidos para a chapa cruciforme com trinca propagando do bordo são apresentados na Figura 5.32.

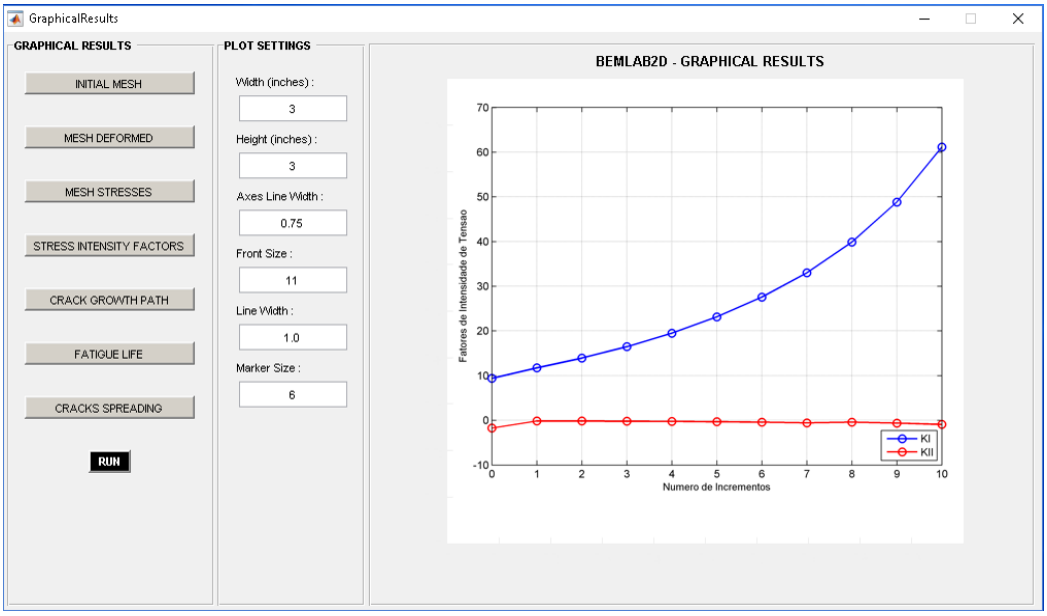


Fig. 5.32 – Fatores de Intensidade de Tensão do exemplo 6

Na Figura 5.33 são apresentados os gráficos da resistência residual normalizada e do número de ciclos de carga sobre cada incremento realizado pelo BemCracker2D.

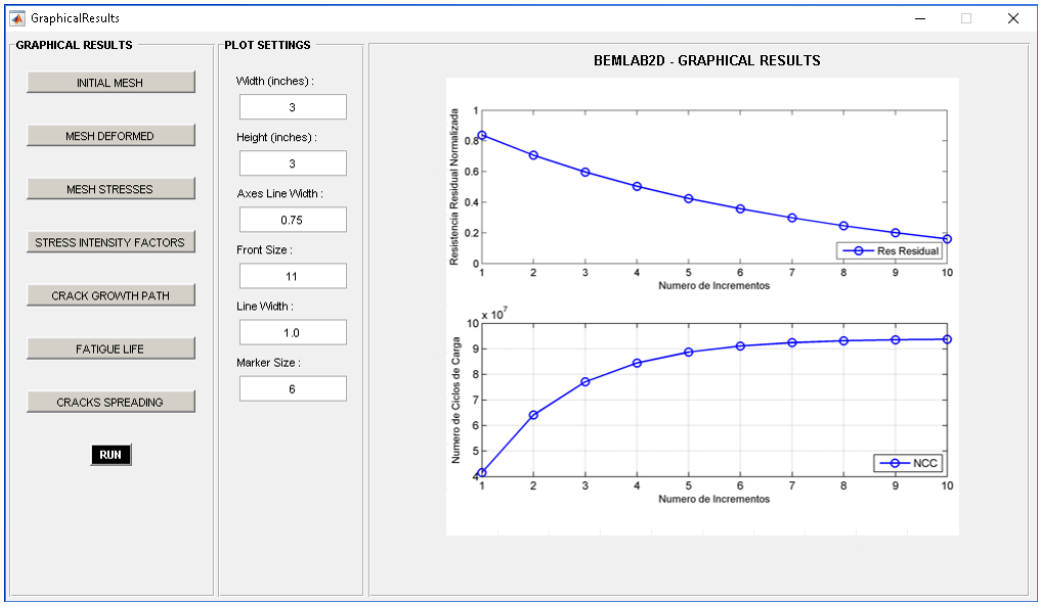


Fig. 5.33 – Resistência residual normalizada e número de ciclos de carga exemplo 6

5.2.7 - Exemplo 7 – Chapa Retangular com Furos e Propagação de Trinca

No subitem 5.1.7 o exemplo abordado é o modelo de uma chapa retangular com três furos de fixação não-colineares e uma trinca projetando sobre a borda de um dos furos, a análise realizada neste exemplo foi pelo MEC Dual consideração da propagação de trinca. O caminho de propagação da trinca na placa é apresentada na Figura 5.34.

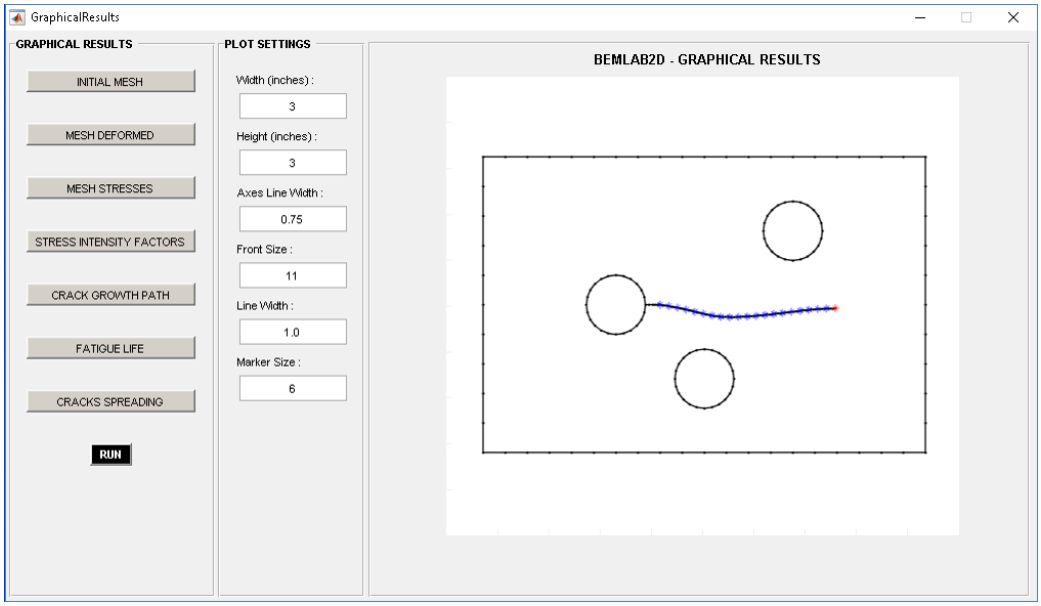


Fig. 5.34 – Caminho de propagação da trinca do exemplo 7

Os gráficos do Fator de Intensidade de Tensão obtidos para a chapa retangular com três furos de fixação não-colineares e uma trinca são apresentados na Figura 5.35.

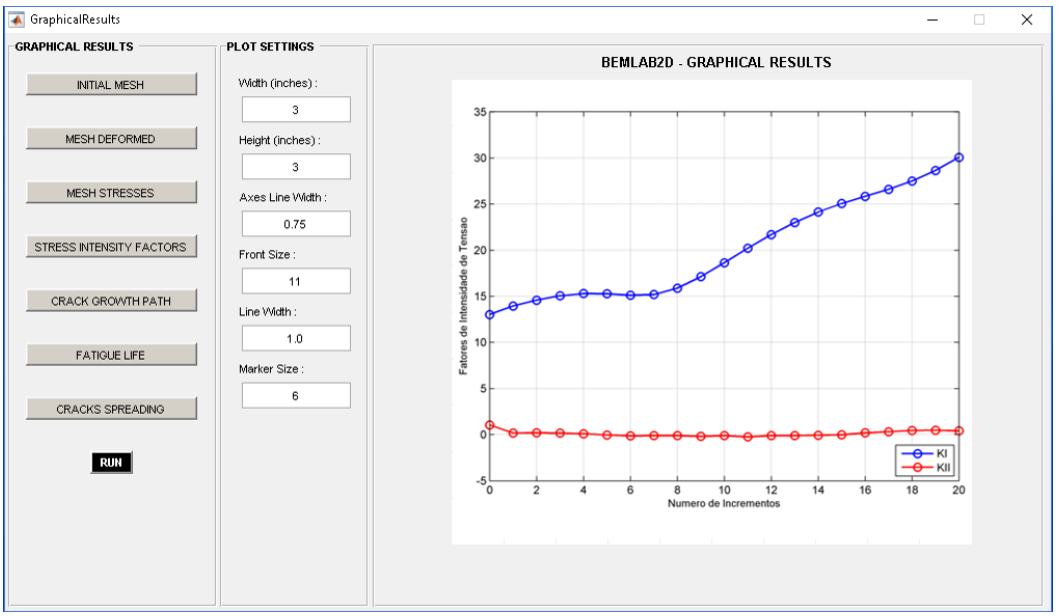


Fig. 5.35 – Fatores de Intensidade de Tensão do exemplo 7

Na Figura 5.36 são apresentados os gráficos da resistência residual normalizada e do número de ciclos de carga sobre cada incremento realizado pelo BemCracker2D para o exemplo em questão.

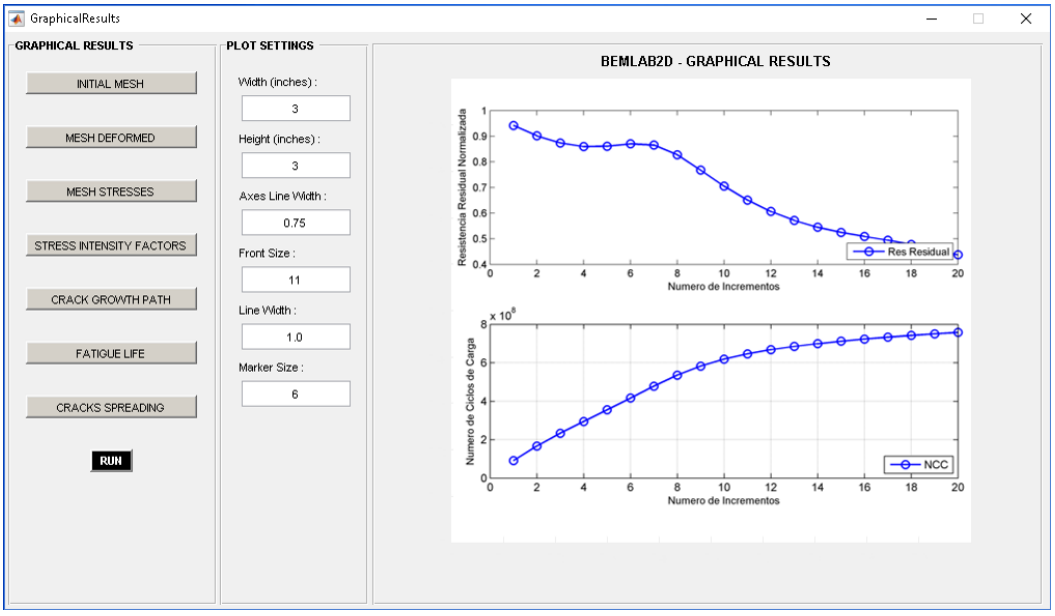


Fig. 5.36 – Resistência residual normalizada e número de ciclos de carga exemplo 7

5.3 - GERADOR DE MALHA DE MEF/MESHLESS

Nos capítulos 3 e 4 foi visto que o Módulo II (Figura 5.37) possui a particularidade de gerar malhas de Método dos Elementos Finitos (MEF) e Método Sem Malha (MESHLESS) a partir da malha do Método dos Elementos de Contorno (MEC). Assim, nesta seção serão apresentados dois exemplos onde as malhas são geradas através do segundo módulo que compõe o BEMLAB2D.

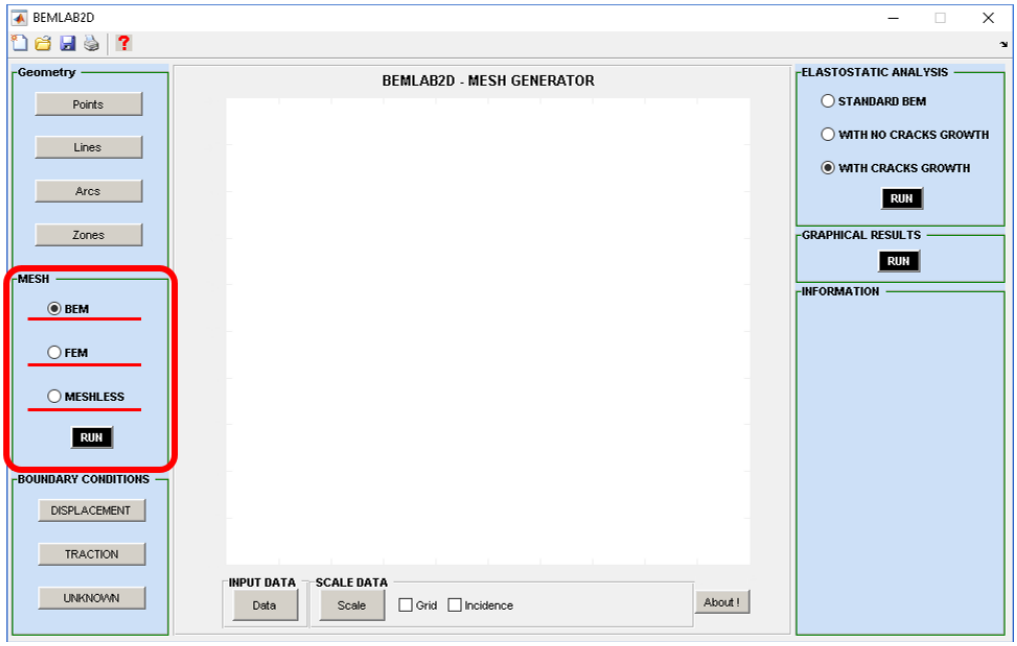


Fig. 5.37 – BEMLAB2D – Detalhes do Módulos II

5.3.1 - Exemplo 1 – Chapa Quadrada com Furos e Trinca de Borda

A chapa quadrada, como pode ser visualizada na figura 5.38, uma trinca se projeta de uma de suas faces com dois furos simétricos com diâmetro de 2 unidades. A trinca projetada mede 4 unidade de comprimento (Módulo I).

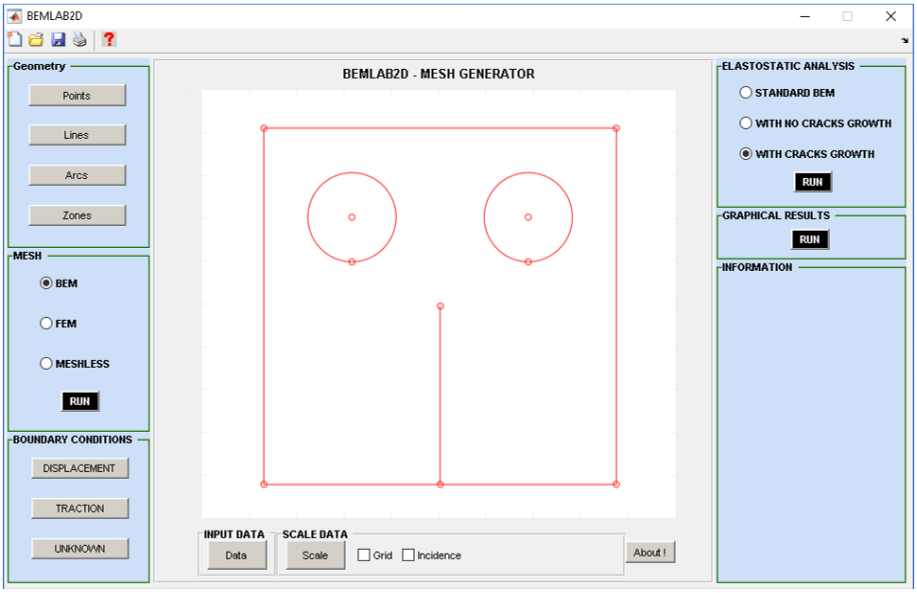


Fig. 5.38 – Modelo geométrico da chapa quadrada

Para este modelo foram usados 140 elementos de contorno e nenhum pontos internos (Módulo II), como pode-se ver na malha de MEC em 5.39.

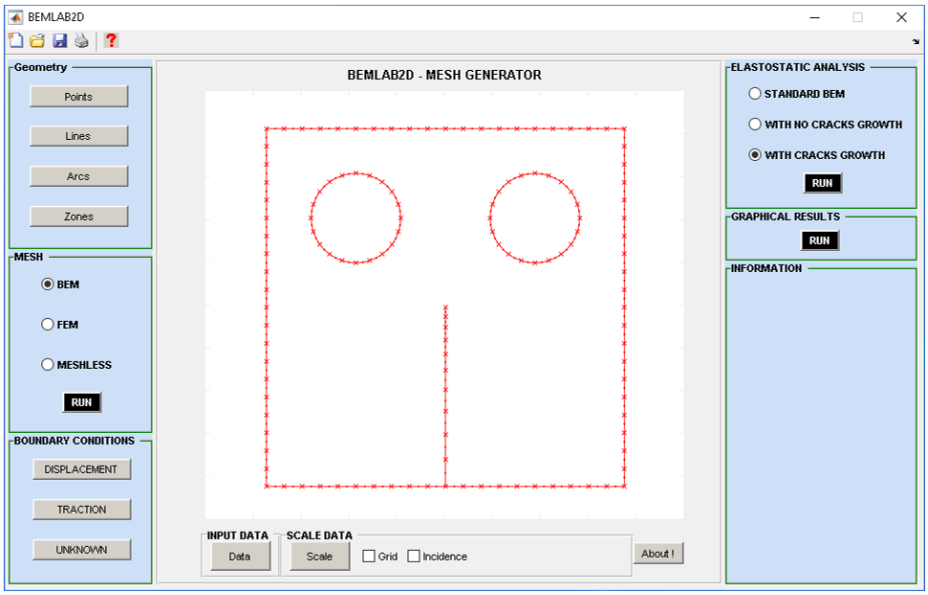


Fig. 5.39 – Malha de MEC para a chapa quadrada

Na construção da malha de MEF para esse exemplo foram utilizados os 140 elementos de contorno da malha de MEC e foi escolhido uma fração de refinamento de malha de 18 eixos horizontais, a trinca foi substituída da forma descontínua, representada na Figura 5.39, para uma trinca de 8 elementos contínuos, Assim, a Figura 5.40 apresenta a malha de MEF.

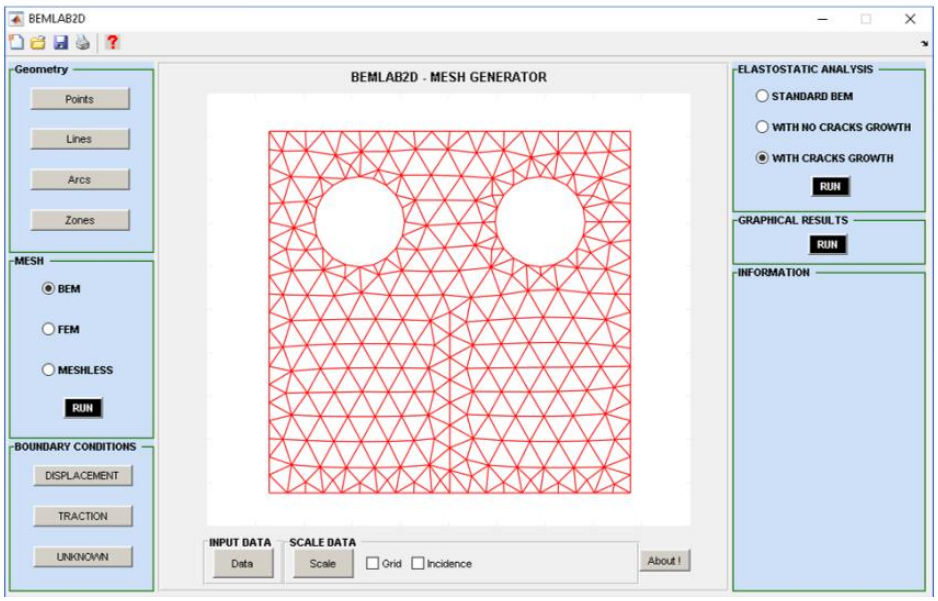


Fig. 5.40 – Malha de MEF para a chapa quadrada

A malha de MESHLESS foi construída a partir dos 140 elementos da malha de MEC, onde os elementos da malha de contorno são composta por elementos quadráticos. Assim três nós do elemento de contorno fica sendo o equivalente a três nós da malha de MESHLESS. A construção dos pontos internos se deu da mesma forma que a de MEF, utilizando uma fração de refinamento de malha de 39 eixos horizontais e a trinca foi substituída por 8 elementos contínuos. A malha de MESHLESS está apresentada na Figura 5.41.

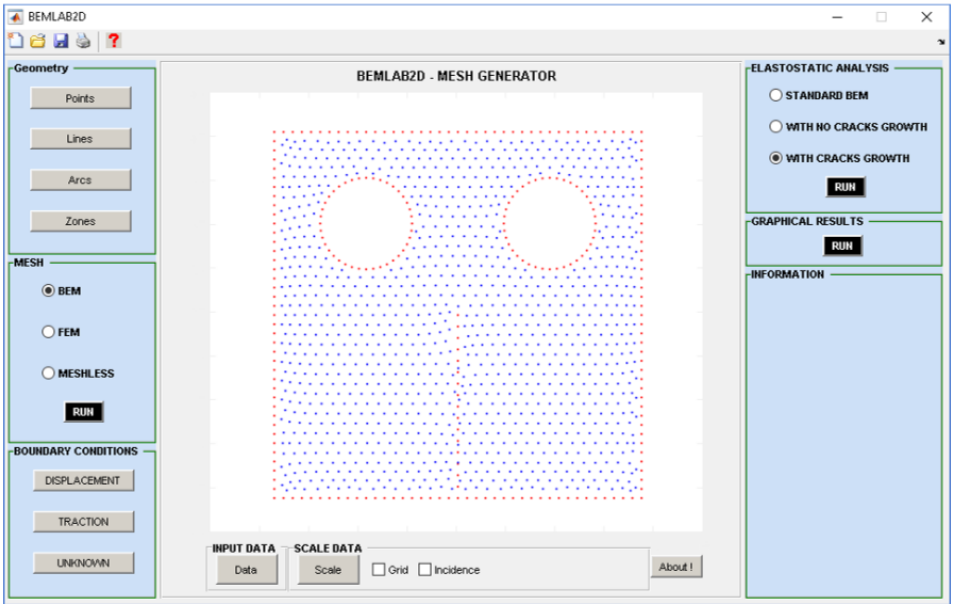


Fig. 5.41 – Malha de MESHLESS para a chapa quadrada

5.3.2 - Exemplo 2 – Modelo de Seção em I

A seção em I, como pode ser visualizada na figura 5.42, tem-se um modelo com suas mesas de tamanho 2x8 unidade de comprimento e alma de 4x2 unidade de comprimento. A seção I foi escolhida para estudar o comportamento dos algoritmos perante entradas vazias nas seções geométricas (Módulo I).

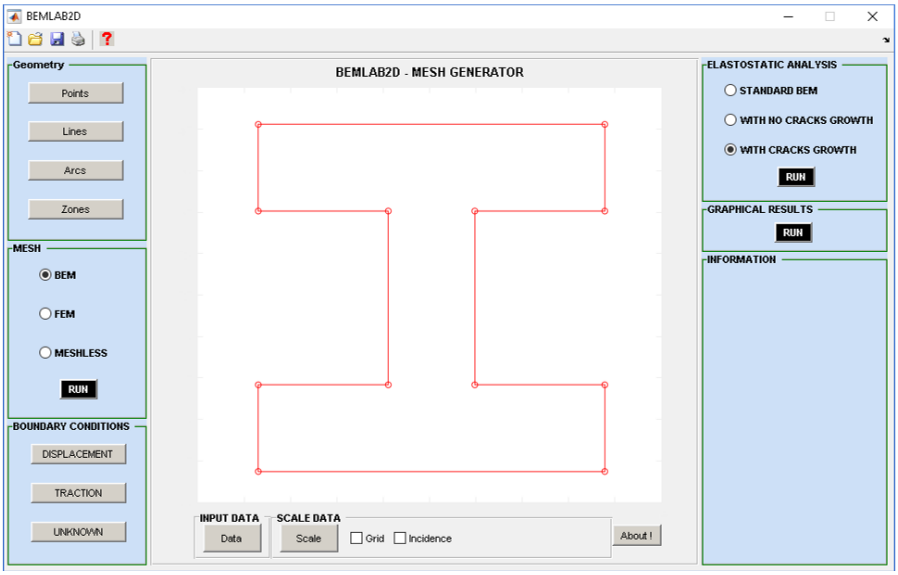


Fig. 5.42 – Modelo geométrico da seção I

Para este modelo foram usados 132 elementos de contorno e nenhum pontos internos (Módulo II), como pode-se ver na malha de MEC em 5.43.

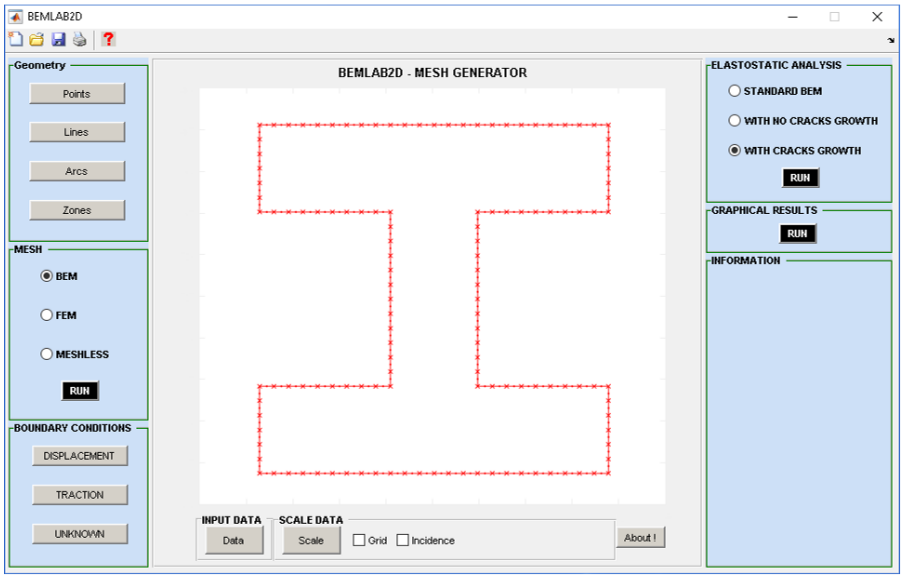


Fig. 5.43 – Malha de MEC para a da seção I

Na construção da malha de MEF para esse exemplo foram utilizados os elementos de contorno da malha de MEC sem alteração. O refinamento escolhido foi de 22 eixos horizontais na construção da malha de MEF representado na Figura 5.44.

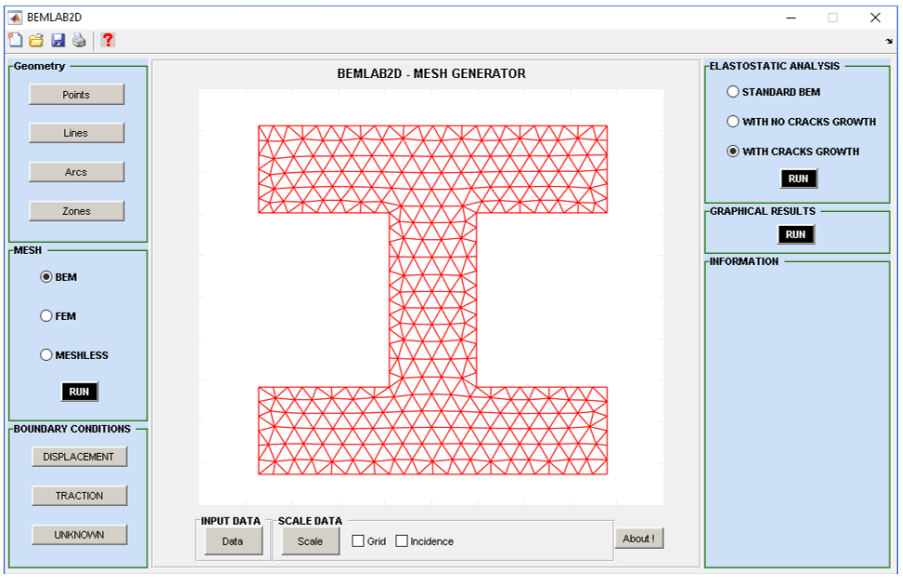


Fig. 5.44 – Malha de MEF para a da seção I

Como no exemplo 1, a malha de MEC neste exemplo é composto por elementos quadráticos, assim pra cada elemento de MEC três pontos são gerados para a malha de MESHLESS. A malha de MESHLESS foi construída a partir dos 132 elementos da malha de MEC. Foi utilizada uma fração de refinamento de malha de 47 eixos horizontais na construção dos pontos internos. A malha de MESHLESS está apresentada na Figura 5.45.

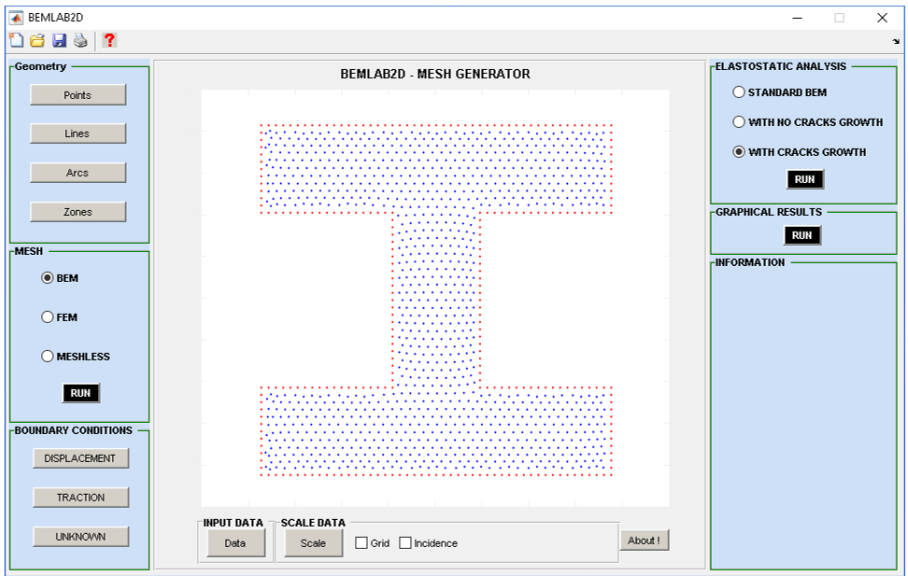


Fig. 5.45 – Malha de MESHLESS para a chapa quadrada

6 - CONCLUSÕES E SUGESTÕES

6.1 - CONCLUSÕES

Interfaces gráficas, de fato, proporcionam um ambiente mais amigável e intuitivo para engenheiros e/ou especialistas modelarem seus experimentos, entretanto, o processo de desenvolvimento de *software* com GUI exige cautela na implementação. Por outro lado, embora a escolha errada dos parâmetros de entrada não seja um fator determinante para o não funcionamento do *software*, pode comprometer a escalabilidade tanto do desenvolvimento quanto da utilização e, neste aspecto, o BEMLAB2D mostrou-se um ambiente de produção bem ágil e seguro, proporcionando maior confiabilidade na execução dos modelos a serem simulados.

O BEMLAB2D agrega em seu fluxo de trabalho diversas etapas de pré-processamento responsáveis pela construção do modelo de forma sequencial e interativa. Estas etapas envolvem desde os objetos de desenho e de atributos físicos até a geração automática de malhas de elementos finitos, elementos de contorno e *meshless*, a partir do acionamento de botões ou cliques do *mouse*, de forma prática e eficiente. E ainda, o modelo fisico-geométrico e malhas, gerados pelo BEMLAB2D nesta etapa, são armazenados em diferentes formatos **(.fig**, **.m**, **.dat**), flexibilizando o manuseio e a edição desses arquivos de dados internamente ou externamente ao seu ambiente. Além disso, esses arquivos podem ser usados como arquivos de entradas para construir malhas de MEF em *softwares* como, ANSYS, ABAQUS, SAP, entre outros.

Com relação ao programa de análise, ou etapa de processamento, utilizou-se um programa chamado BemCracker2D, que faz análise bidimensional de problemas elastostáticos usando o MEC convencional e o MEC Dual para problemas envolvendo trincas, cumprindo, portanto, o segundo objetivo deste trabalho. Para isso, o BEMLAB2D gera um arquivo **.dat**, conforme padrão estabelecido pelo BemCracker2D, que é utilizado na realização da análise. Por sua vez, o programa de análise devolve uma série de arquivos **.out** que são lidos e interpretados pelo BEMLAB2D para alimentar o pós-processador.

O pós-processamento do BEMLAB2D foi desenvolvido especificamente para interagir com o programa BemCracker2D e, por esse motivo, possui diversas *functions* em Matlab que processam graficamente os mais variados tipos de plotagem de resultados, tais como

deformada da malha e caminho de propagação da trinca. Essas *functions* são invocadas pelo simples clicar de seus respectivos botões e plotadas em uma GUI específica do pós-processador, proporcionando versatilidade ao manuseio, edição e visualização dos resultados.

Por fim, ainda com relação à interface gráfica desenvolvida, vale ressaltar que a performance foi alcançada devido à implementação dos algoritmos no ambiente MATLAB, o qual possui rotinas e comandos em sua biblioteca que deram suporte ao desenvolvimento eficiente do BEMLA2D. O programa desenvolvido permite que os modelos geométricos sejam construídos de maneira simples, sequencial e interativa. Além disso, o programa permite também a visualização imediata do modelo, construído a partir dos simples comandos, evitando, assim, a tediosa tarefa de definir a geometria na forma de texto (arquivos de dados).

6.2 - SUGESTÕES PARA TRABALHOS FUTUROS

A interface proposta atende particularmente a construção e edição de modelos e malhas bidimensionais de elementos de contorno. A extensão da estrutura proposta para o campo da modelagem tridimensional de elementos de contorno seria uma das principais sugestões de trabalho futuro, englobando assim uma gama maior de modelos e análises numéricas propostos pela engenharia, além de proporcionar maior robustez e clareza na interpretação física-geométrica destes modelos.

O programa trata apenas de modelos bidimensionais elastostáticos, desenvolvendo entradas de dados apenas do tipo forças/tensões e deslocamentos. Neste aspecto, há necessidade de implementação de entradas e saídas de dados referentes a modelos de problemas de potencial, cargas dinâmicas e cargas móveis, aumentando, assim, o leque de problemas na plataforma BEMLAB2D.

Na construção dos elementos de contorno os dados exportados geram topologias de elementos quadráticos apenas, devido a natureza do programa BemCracker2D. Assim, outra sugestão bastante interessante seria a implementação de diversos tipos de elementos (constantes, linear, cúbicos...) de forma automática e especifica para cada segmento, agregando mais ainda robustez ao BEMLAB2D.

Outra contribuição enriquecedora para a continuação deste trabalho seria a implementação

de mais itens de modelagem e edição, no que diz respeito às etapas do pré-processamento. Um dos incrementos seria melhorias na modelagem, possibilitando mais definições de contorno além de segmentos retos e arcos, trazendo implementações para construção de segmentos elípticos e parabólicos. Também apresenta grandes oportunidades de melhorias nas edições na malha de Método dos Elementos Finitos, adaptando refinamentos em torno de furos, inclusões e trincas de forma automática ou manual, melhorando assim o refinamento da malha em áreas especificas sem precisar refinar todo modelo.

A continuação do desenvolvimento do programa de pós-processamento para a visualização de outros resultados mais completos completaria o conjunto de resultados de análises gerados pela interface gráfica, que são necessários em análises clássicas de engenharia. O estudo de resultados plásticos, iniciação de trincas, envoltórias entre outros agregaria muito ao estudo realizado neste trabalho.

Por fim, tendo em vista que a interface gráfica BEMLAB2D, em seu pré-processamento, independe do programa de análise BemCracker2D, pois gera arquivos independentes para cada tipo de malha e, por outro lado, seu pós-processamento é totalmente dependente do programa de análise, uma outra sugestão seria o desenvolvimento de uma interface de comunicação paralela entre os programas, ou seja, fazer programação paralela entre C++ e MATLAB que possibilitasse, por exemplo, a visualização da propagação de trinca em tempo de execução.

7 - REFERÊNCIAS BIBLIOGRÁFICAS

ALIABADI, M. H. **The Boundary Element Method - Applications in Solids and Structures**. WILEY: [s.n.], v. 2, 2002.

ALMEIDA, M. P. **Software Gerador de Malhas Triangulares para Análise com o Método dos Elementos Finitos**. Universidade Federal do Pará. Belém, p. 67. 2014.

BATHE, K.-J.; WILSON, E. L. **Numerical Methods in Finite Element Analysis**. New Jersey: PRENTICE-HALL, INC, 1976.

BECKER, A. A. The Boundary Integral Equation Method in Axisymmetric Stress Analysis Problems. **Springer-Verlag**, Berlin, 1986.

BLANDFORD, G. E.; INGRAFFEA, A. R.; LIGGETT, J. A. Two-Dimensional Stress Intensity Factor Computations Using the Boundary Element Method. **International Journal for Numerical Methods in Engineering**, v. 17, p. 387-404, 1981.

BREBBIA, C. A. **The Boundary Element Methods for Engineers**. London, New York: Penthec Press, Halstead Press, 1978.

BREBBIA, C. A.; DOMINGUEZ, J. Boundary Elements an Introductory Course. **Computational Machanics Publications**, Southampton, 1989.

BREBBIA, C. A.; WALKER, S. **Boundary Element Techniques in Engineering**. London: Newnes Butterworths, 1980.

BUSH, M. B. Simulation of contact-induced fracture. **Engineering Analysis with Boundary Elements**, v. 23, p. 59-66, 1999.

CAIRES, L. D.; SOUSA, E. A. C. Reconstrução e Geração de Malhas Bidimensionais Não-Estruturadas Utilizando a Triangulação de Delaunay. **VI Congresso Nacional de Engenharia Mecânica**, Campina Grande - PB, agosto 2010. 8.

CHAPMAN, S. J. **"Programação em MATLAB para Engenheiros"**. [S.l.]: Thompson learning, 2006.

DELAUNAY, B. Sur la sphère vide. **Otdelenie Matematicheskikh i Estestvennykh Nauk**, Izvestia Akademii Nauk SSSR, v. 7, p. 793-800, 1934.

GOMES, G. **Estrutura de Dados para Representação de Modelos Bidimensionais de Elementos de Contorno.** Universidade de Brasília. [S.l.], p. 95. 2000. (Publicação 004A/2000).

GOMES, G. **"Aplicação dos Métodos de Elementos de Contorno e Reciprocidade Dual em Problemas de Plasticidade 2D Orientada a Objeto"**. Universidade de Brasília - UnB. Brasília, p. 165. 2006. (E.TD- 002A/06).

GOMES, G. **"BMOO_GI"**. BR512015000283-0, 27 mar. 2015.

GOMES, G.; DELGADO NETO, A. Modelling and 2D cracks view using traction boudary integral equation. **XXXVII Iberian Latin American Congress on Computational Methods in Engineering - CILAMCE**, Brasília, 2016.

GOODIER, J. N.; TIMOSHENKO, S. P. **Theory of Elasticity**. St. Petersburg, Rússia: Mc Graw-Hill, v. I, 1914.

GUIMARÃES, A. C. S.; FEIJÓO, R. A. **GAMAT2 - GERADOR AUTOMÁTICO DE MALHAS TRIANGULARES DE ELEMENTOS FINITOS LINEARES E QUADRÁTICOS**. Laboratório Nacional de Computação Científica. Rio de Janeiro - Brasil: [s.n.]. p. 70-79.

HU, L. et al. IGMESH: A convenient irregular-grid-based pre- and post-processing tool for TOUGH2 simulator. **Computers & Geosciences**, v. 95, p. 11-17, 2016.

INGRAFFEA, A. R.; BLANDFORD, G. E.; LIGGET, J. A. **Automatic Modelling of Mixed-Mode Fatigue and Quasi-Static Crack Propagation Using the Boundary Element Method**. Proc. of Fracture Mechanics: Fourteenth Symposium. ASTM STP 791: ASTM. 1983. p. 407-426.

JOSHI, J.; SAHARAN, S.; MANDAL, P. K. BOLDSync: A MATLAB-based toolbox for synchronized stimulus presentation in functional MRI. **Journal of Neuroscience Methods**, v. 223, p. 123-132, 2014.

KAUWELOA, K. I. et al. A graphical user interface (GUI) toolkit for the calculation of three-dimensional (3D) multiphase biological effective dose (BED) distributions including statistical analyses. **Computer Methods and Programs in Biomedicine**, v. 131, p. 1-12, 2016.

MAHABADI, O. K.; GRASSELLI, G.; MUNJIZA, A. Y-GUI: A graphical user interface and pre-processor for the combined finite-discrete element code, Y2D, incorporating material heterogeneity. **Computers & Geosciences**, v. 36, p. 241-252, 2010.

MUNJIZA, A. **The combined finite-discrete element method**. Chichester. West Sussex, p. 333. 2004.

PORTELA, A.; ALIABADI, M. H.; ROOKE, D. P. The dual boundary element method: Effective implementation for crack problems. **International Journal for Numerical Methods in Engineering**, v. 33, p. 1269-1287, 1992.

PORTELA, A.; ALIABADI, M. H.; ROOKE, D. P. Dual boundary element analysis of cracked plates: singularity subtraction technique. **International Journal of Fracture**, v. 55, p. 17-28, 1993.

PRUESS, L.; OLDENBURG, C.; MORIDIS, G. **TOUGH2 User's Guide, V2.1.** Lawrence Berkeley National Laboratory. California, USA. 2012. (LBNL-43134).

RODRIGUES, H.; VARUM, H. **"Interface Gráfico para Preparação de Dados e Visualização de Resultados de um Programa de Análise Não-Linear de Estruturas"**. Universidade Aveiro. [S.l.]. 2005.

SAKAMOTO, M. M. **Algoritmo de Refinamento de Delaunay a Malha Sequenciais, Adaptativas e com Processamento Paralelo**. Universidade de São Paulo Escola Politécnica. São Paulo, p. 150. 2007.

SHEWCHUK, J. R. Reprint of: Delaunay refinement algorithms for triangular mesh generation. **Computational Geometry: Theory and Applications**, v. 47, p. 741-778, February 2014.

TRAN, A. P.; DAFFLON, B.; HUBBARD, S. iMatTOUGH: An open-source Matlab-

based graphical user interface for pre- and post-processing of TOUGH2 and iTOUGH2 models. **Computers & Geosciences**, v. 89, p. 132-143, 2016.

WANG, B. et al. Fast centroidal Voronoi Delaunay triangulation for unstructured mesh generation. **Journal of Computational and Applied Mathematics**, v. 280, p. 158-173, 2015.

WAWRZYNEK ET AL., P. A. “FRANSYS: A Software System for the Simulation of Crack Propagation in Three-Dimensions”. **Proc. Symposium on Discretization in Structural Mechanics**, Vienna, June 1989. 273-282.

WAWRZYNEK, P. A. **"Interactive Finite Element Analysis of Fracture Processes: in Integrated Approach"**. Cornell University. [S.l.]. 1986.

ZANG, Z. X. et al. Granger causality analysis implementation on MATLAB: A graphic user interface toolkit for fMRI data processing. **Journal of Neuroscience Methods**, v. 203, p. 418-426, 2012.