UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Desenvolvimento de um Programa Base em Elementos de Contorno para Aplicações a Análises Mecânicas Tridimensionais em Bioengenharia

Autor: **Pedro Yoshito Noritomi**
Orientador: **Paulo Sollero**

51/2000

ESTE EXEMPLAR CORRESPONDE À REDAÇÃO FINAL DA TESE DEFENDIDA POR PEDRO YOSHITO NORITOMI E APROVADA PELA COMISSÃO JULGADORA EM 22/02/2000

ORIENTADOR

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

Desenvolvimento de um Programa Base em Elementos de Contorno para Aplicações a Análises Mecânicas Tridimensionais em Bioengenharia

Autor: **Pedro Yoshito Noritomi**
Orientador: **Paulo Sollero**

Curso: Engenharia Mecânica
Área de Concentração: Mecânica Computacional

Dissertação de mestrado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Mestre em Engenharia Mecânica.

Campinas, 2000
S.P . - Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

N777d

Noritomi, Pedro Yoshito
Desenvolvimento de um programa base em elementos de contorno para aplicações a análises mecânicas tridimensionais em bioengenharia / Pedro Yoshito Noritomi.--Campinas, SP: [s.n.], 2000.

Orientador: Paulo Sollero.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Elementos de contorno. 2. Fêmur. 3. Ánalise elástica (Engenharia). I. Sollero, Paulo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânico. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

DISSERTAÇÃO DE MESTRADO

Desenvolvimento de um Programa Base em Elementos de Contorno para Aplicações a Análises Mecânicas Tridimensionais em Bioengenharia

Autor: **Pedro Yoshito Noritomi**
Orientador: **Paulo Sollero**

Prof. Dr. Paulo Sollero, Orientador
Universidade Estadual de Campinas

Prof. Dr. Euclides de Mesquita Neto
Universidade Estadual de Campinas

Prof. Dr. William Dias Belangero
Universidade Estadual de Campinas

Campinas, 22 de fevereiro de 2000

Dedicatória:

Dedico este trabalho a meus pais, minha irmã e minhas tias.

Agradecimentos:

Agradeço ao CNPq pelo apoio financeiro e ao Departamento de Mecânica Computacional da Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas pelo acolhimento e auxílio técnico e científico.

Agradeço a todos os colegas e amigos, cuja ajuda foi e continua sendo de fundamental importância para a construção de minha vida. Em especial, agradecimentos particulares ao Frederico, Éder e William pelas horas de conversa descontraída, fundamental para o bom desenvolvimento das idéias contidas no trabalho. Ao pesquisador Piotr Fedelinski pelas dicas de teste para o código E-Con3D. Ao meu pai, por ter corrigido a versão final da tese e a minha mãe e irmã pelo apoio logístico de me manterem em Campinas.

"Depois de procelosa tempestade,
Noturna sombra e sibilante vento,
Traz a manhã serena claridade,
Esperança de porto e salvamento;
Aparta o sol a negra escuridade,
Removendo o temor do pensamento (...)"

Luíz Vaz de Camões, Os Lusíadas, Canto IV, trecho da primeira estrofe

SUMÁRIO

1. Introdução 1
   1.1. Objetivo 1
   1.2. Considerações iniciais 1
2. Aspectos teóricos 4
   2.1. Sobre a Bioengenharia 4
      2.1.1 Abordagem Experimental 7
      2.1.2 Abordagem Matemática 13
   2.2. O material ósseo 15
   2.3. O método de cálculo 23
   2.4. Implementação numérica computacional 31
   2.5. Cálculo dos tensores de tensão nodal 41
   2.6. A integração singular 44
   2.7. Métodos de otimização utilizados 50
3. Desenvolvimento computacional 52
   3.1. O programa E-Con3D 52
   3.2. Funções principais 60
      3.2.1 A função INTEGAUS.m 60
      3.2.2 A função BCS.m 61
      3.2.3 As funções FRMTRX1.m e FRMTRX2.m 64
      3.2.4 A função OUTPUT.m 75
   3.3 Funções auxiliares 77
      3.3.1 Função COORD.m 78
      3.3.2 Função SHAP8.m 78
      3.3.3 Função LINEAR.m 79

3.3.4 Função JACOBI.m 81
3.3.5 Função TRIANG.m 81
3.3.6 Função KERNEL.m 82
3.3.7 Função INTGR.m 83
3.3.8 Função ITSS.m 84
3.3.9 Função FMATR.m 86
3.3.10 Função TENSNODAL.m 87
3.4 Função de pós-processamento 89
3.5 Programas para conversão de Ansys® para E-Con3D 90
4. Análise de resultados 92
4.1. Exemplo 1 – cubo atuando como barra sob tensão de tração 98
4.2. Exemplo 2 – placa com furo central tracionada uniaxialmente 102
4.3. Exemplo 3 – anel com carga interna em estado plano de deformação 103
4.4. Exemplo 4 – modelo de um fêmur sob carga uniaxial 106
5. Conclusões 111
6. Propostas para trabalhos futuros 113
Referências bibliográficas 114
Anexo 1 – Estrutura dos arquivos de entrada para E-Con3D 118
Anexo 2 – Códigos de tradução de Ansys para E-Con3D 119
Anexo 3 – Listagem do programa E-Con3D 120

Resumo

NORITOMI, Pedro Yoshito, Desenvolvimento de um Programa Base em Elementos de Contorno para Aplicações a Análises Mecânicas Tridimensionais em Bioengenharia, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. . Dissertação (Mestrado)

Este trabalho apresenta o desenvolvimento de um programa base, em elementos de contorno, aplicado à análise tridimensional de tensões, para utilização como ferramenta de investigação do comportamento estrutural de ossos. A análise de tensões em estruturas ósseas é uma aplicação de grande interesse dentro da biomecânica, principalmente na área ligada à ortopedia. Esse tipo de aplicação exige que o programa possibilite a modelagem de geometrias complexas, típicas de estruturas ósseas, além de dispor de uma via aberta para a implementação de novas formulações, mais adequadas aos fenômenos naturais de materiais ósseos. Para realizar a análise tridimensional de tensões foi desenvolvido o programa E-Con3D, que utiliza elementos de contorno do tipo contínuo, quadrilateral de oito nós, quadrático e isoparamétrico. A solução fundamental implementada é a isotrópica linear homogênea. Embora o modelo de material isotrópico seja simples em comparação com o comportamento observado para o material ósseo, este consiste num primeiro passo para o desenvolvimento futuro de modelos mais adequados. Além disso, a utilização do modelo isotrópico é suficiente para o estabelecimento da funcionalidade do programa frente a aplicações envolvendo geometrias reais de estruturas ósseas. O programa foi implementado em linguagem MatLab. O programa MatLab é apresentado como uma ferramenta poderosa no desenvolvimento de programas para análise numérica, pois contém funções pré-programadas que possibilitam a implementação do método numérico em uma programação de alto nível assim como a visualização e análise dos resultados obtidos, como são apresentadas no trabalho.

Palavras-chave

- elementos de contorno, osso do fêmur, análise de tensão tridimensional

Abstract

This work presents a development of a boundary elements program applied to the tridimensional stress analysis and its use in the investigation of the structural behavior of a femur bone. The stress analysis in bone structures is an application of great interest in biomechanics, mainly at the area of orthopedic applications. This kind of application requires a program which enables the modeling of complex geometries, typical of bone structures, and presents an expansion capability for implementation of new formulations more fitting to the natural phenomena that occurs in the bone material. Considering the tridimensional stress analysis a program called E-Con3D was developed. This program uses continuous, quadrilateral, eight nodes, and quadratic isoparametric boundary elements. The fundamental solution implemented is the isotropic linear homogeneous. Although the material isotropic model is quite simple when compared with the behavior observed on the bone material, this model consists in a first step to the future development of more fitting formulations. Beyond this, the use of an isotropic model is sufficient to the functionality verification of the program when applied to real geometries of bone structures. The program was implemented in MatLab® language. The MatLab® program is presented, in this work, as a powerful tool for the development of numerical analysis programs, once contains pre-programmed functions that makes possible the numerical method implementation using a high level programming language. The same may be used to the results visualization, making possible an efficient results interpretation.

Key Words

- boundary elements, femur bone, three dimensional stress analysis

LISTA DE FIGURAS

1	Espécime do tipo compacto montado em dispositivo para ensaio de tração	11
2	Perfis de fratura de corpos de prova compactos	13
3	Representação do fêmur	17
4	Gráfico contendo variação do módulo de elasticidade do fêmur	19
5	Elemento quadrilateral de oito nós	34
6	Construção da matriz *G*	37
7	Construção da matriz *H*	38
8	Elementos triangularizados para integração singular	47
9	Fluxograma de funcionamento do programa E-con3D	54
10	Modelo para o primeiro problema	93
11	Modelo para o segundo problema	94
12	Modelo para o terceiro problema	95
13	Modelo para o quarto problema	98
14	Mapa de deslocamentos dos nós na coordenada global X	99
15	Mapa de tensões sigma XX	100
16	Mapa de deslocamentos em X obtido no Ansys®	101
17	Mapa de deslocamentos em X obtido no Beasy®	101
18	Mapa de tensões nodais sigma YY	103
19	Mapa de deslocamentos radiais	104
20	Mapa de tensões sigma XX na superfície	105
21	Mapa de tensão sigma ZZ	107
22	Mapa de deslocamentos em z	108
23	Mapa de tensões principais máximas	109
24	Mapa de tensão de von Mises	110

LISTA DE TABELAS

1 Alguns valores de K_{Ic} 12

2 Valores para exemplo Kane 104

CAPÍTULO 1

INTRODUÇÃO

1.1 Objetivo

O objetivo deste trabalho de mestrado é a construção de uma ferramenta básica, capaz de suportar o desenvolvimento de métodos numéricos aplicados à bioengenharia no ramo da ortopedia, principalmente no que diz respeito a novos modelos de materiais, novos métodos de cálculo e estudos aplicados a fenômenos relacionados ao comportamento, *in vivo* ou não, de ossos. Sendo assim, é necessário que sejam destacados alguns aspectos, os quais serão relevantes para nortear e justificar alguns dos caminhos seguidos, o que será feito durante a introdução.

1.2 Considerações iniciais

Quando se pensa em bioengenharia aplicada a ossos está se pensando no estudo do comportamento mecânico de estruturas presentes no corpo constituídas desse material.

Isso significa que é necessário descrever o material ósseo do ponto de vista da engenharia, a fim de que seja possível utilizá-lo como se fosse um material de engenharia em simulações numéricas computacionais. Ou seja, é preciso encaixar o material ósseo em um modelo matemático, o que envolve muitas simplificações e exige testes a fim de que se possa comprovar a validade do modelo desenvolvido.

Além disso, na bioengenharia aplicada a ossos, existem fenômenos típicos do osso vivo, os quais não podem ser estudados apenas com os métodos de cálculo comumente aplicados em análises numéricas de engenharia. Fenômenos como o remodelamento ósseo, que envolve o crescimento ou reabsorção de material segundo condições definidas, precisam ser implementados como novas técnicas de cálculo nos métodos computacionais.

Finalmente, a última característica contemplada e que serviu de parâmetro para o desenvolvimento do trabalho, refere-se à complexidade das geometrias envolvidas nos estudos de bioengenharia. Diferentemente das estruturas de engenharia, para as quais a geometria não costuma ser um problema, uma vez que é concebida por um engenheiro, as estruturas ósseas são desenhadas de acordo com sua função, obedecendo a critérios naturais. Isso faz com que as geometrias envolvidas sejam, via de regra, muito complexas e de difícil modelagem. Como a modelagem geométrica constitui etapa primordial para a realização de qualquer análise numérica computacional, é necessário que o programa disponha de uma ferramenta capaz de descrever as geometrias envolvidas.

Dessa forma, a ferramenta computacional básica, cujo desenvolvimento será descrito neste trabalho, deve atender a três características fundamentais:

- flexibilidade para a implementação de novos tipos de material, a fim de que se possa aproximar melhor o comportamento do material ósseo;

- possibilidade de ampliação das técnicas de cálculo do núcleo do programa, para que seja possível avaliar fenômenos como o remodelamento;

- capacidade de descrever modelos de geometrias complexas, tais como as presentes em ossos reais.

Para atender às características de flexibilidade e possibilidade de ampliação do núcleo de cálculo do programa optou-se por implementar um código próprio, de arquitetura aberta. Isso torna possível que usuários posteriores sejam capazes de alterar a estrutura básica do programa, tornando-o mais adequado para a realização de testes para novas implementações ou para o desenvolvimento de novas tecnologias.

Para atender à característica de capacidade de modelagem de geometrias complexas, tridimensionais, típicas de estruturas ósseas, optou-se por desenvolver uma técnica de comunicação com programas comerciais que possibilitasse o uso dos bancos de dados gerados por esses aplicativos. Assim, o que se pretende é utilizar a capacidade de modelagem de um aplicativo comercial para descrever a geometria complexa e utilizar os dados gerados no programa desenvolvido nesse trabalho. Para isso, é necessário o desenvolvimento de uma técnica para migração da base de dados do aplicativo comercial para o programa desenvolvido.

Essas necessidades, aliadas à característica acadêmica do trabalho, levaram à escolha do método dos elementos de contorno aplicado à elasticidade tridimensional como método de solução numérica a ser implementado no núcleo do programa.

CAPÍTULO 2

ASPECTOS TEÓRICOS

2.1 A bioengenharia

Tradicionalmente, a imagem que se tem de medicina está relacionada a um médico e um paciente, uma doença e um remédio para curá-la. No entanto, com os avanços da ciência e da tecnologia, principalmente os ocorridos neste século, tem havido uma progressiva alteração na perspectiva desse relacionamento, dando a paciente e médico a possibilidade de evitar a doença ao invés de apenas tratá-la.

Além da conduta profilática, novas tecnologias e descobertas no ramo da biologia e farmacêutica possibilitaram o desenvolvimento de remédios mais eficientes, fornecendo um novo arsenal químico para o combate a doenças que antes eram incuráveis, ou de alto risco para a vida do paciente.

Tais avanços tiveram como conseqüência primeira o aumento da expectativa de vida das pessoas, mas não somente isso. Juntamente com o aumento da idade veio o aumento na qualidade de vida, o que significou um envelhecimento mais saudável e ativo. Além disso, o estilo de vida moderno, no qual o trabalho ocupa a maior parte da vida, vem contribuindo para que uma grande quantidade de pessoas em idade avançada, livres de

obrigações de trabalho, se interessem em desenvolver atividades de lazer diversas. Isso tem sido facilitado, uma vez que diversas doenças anteriormente incapacitantes e típicas de idades avançadas têm sido evitadas graças aos novos remédios e tratamentos da medicina (*Workshop de Biomaterias* (1999)).

No entanto, essa nova conduta dos pacientes mais idosos despertou nos médicos um outro tipo de preocupação. Se agora as pessoas idosas, em grande parte, estavam livres de problemas de saúde provocados por doenças, quais seriam os efeitos dessa super-atividade sobre as estruturas de sustentação do corpo, desgastadas e envelhecidas, ou seja, quais as conseqüências sobre os sistemas ósseos do corpo.

Sintomas dessa alteração no modo de vida das pessoas puderam ser vistos na ortopedia, ramo da ciência médica que cuida da preservação ou restauração funcional do esqueleto humano, através do aumento dos casos de fratura de ossos longos em pessoas de idade avançada (*Lotz el al.* (1992), *Gharpuray et al.* (1990)). Tipicamente, as causas desse tipo de fratura eram quedas ou esforços originados de atividade física exagerada e sem orientação adequada, associados a uma condição de deficiência de cálcio nos ossos, osteoporose (*Etchebere* (1998)).

O aumento da incidência desse tipo de problema despertou o interesse da comunidade médica em estudar que tipos de esforços são suportados pelos ossos e quais as condições que levam a um risco maior de ocorrência de uma fratura. Tais interesses podem ser traduzidos como uma necessidade de realizar algum tipo de análise mecânica na estrutura óssea dos seres humanos, ou seja, algo semelhante a se fazer uma análise mecânica numa estrutura de engenharia. Nascia assim um ramo da bioengenharia concentrado na aplicação de ferramentas de análise estrutural de engenharia à solução de problemas de medicina ortopédica.

Note que a bioengenharia não está restrita apenas a esse tipo de aplicação, que envolve o uso de ferramentas de análise estrutural para o estudo de ossos e problemas de ortopedia. Ela é muito mais que isso. Diz respeito também ao estudo e desenvolvimento

de novos materiais para aplicações em medicina, ao desenvolvimento de novos equipamentos para auxílio em tratamentos e procedimentos médicos, ao desenvolvimento de novas técnicas cirúrgicas com a utilização de dispositivos protéticos ou sintéticos; enfim, toda uma gama de novas atividade que envolvem o estudo e o desenvolvimento de soluções para problemas biológicos através do uso de técnicas ou dispositivos que sejam projetados de acordo com os procedimentos da engenharia (*Workshop de Biomaterias* (1999)).

No entanto, como o foco deste trabalho é o estudo e desenvolvimento de um programa base adequado ao estudo estrutural de ossos, será dado destaque ao ramo da bioengenharia relacionado com o desenvolvimento de técnicas para adaptação de ferramentas de engenharia ao uso em análises de estruturas ósseas, daqui para frente chamado genericamente de bioengenharia.

Sob esse ponto de vista, é possível evidenciar duas abordagens principais para a análise estrutural dos ossos presentes no esqueleto humano. Uma que pode ser chamada de abordagem experimental, cuja fundamentação está na realização de ensaios mecânicos, segundo normas definidas para materiais de engenharia, em ossos. Isso tem por finalidade tentar caracterizar o osso como um material de engenharia ou adaptar normas existentes a esse novo material. Outra pode ser chamada de abordagem matemática, que visa elaborar uma metodologia de cálculo que possa ser considerado suficiente para representar o material ósseo em uma aplicação específica. Essa abordagem possibilita o uso de um código numérico computacional para estudar situações de interesse sem a necessidade de um procedimento experimental, geralmente mais custoso e demorado.

Apesar de aparentemente desconectadas e das grandes diferenças existentes entre a abordagem experimental e a matemática, ambas estão intrinsecamente ligadas. Na verdade, para que a abordagem matemática possa desenvolver modelos mais adequado e representativos para as aplicações a que se destinam, ela necessita de dados fornecidos pelos experimentos realizados a partir de uma abordagem experimental, seja para

completar e particularizar o modelo matemático a um tipo específico de material, ou para confirmar a validade dos resultados obtidos em relação ao fenômeno real que se deseja representar.

Por outro lado, para que a abordagem experimental possa desenvolver novos procedimentos mais adaptados às novas características e particularidades do material ósseo, ela necessita da velocidade e versatilidade das análises numéricas feitas a partir da abordagem matemática. A grande vantagem das análises realizadas com base na abordagem matemática é que são capazes de realizar um número maior de simulações em tempo e a custos menores que ensaios experimentais, facilitando na determinação de informações essenciais que os ensaios devam buscar a fim de caracterizar o material.

2.1.1 **Abordagem experimental**

A primeira aplicação básica e natural de um método experimental a materiais ósseos é na determinação de propriedades desse material. Quando se fala em propriedades, do ponto de vista da engenharia, está se buscando características intrínsecas ao material, tais como seu módulo de elasticidade, coeficiente de Poisson, módulo de cisalhamento, etc.

No caso da maioria dos materiais, utilizados em engenharia, é necessário apenas um valor de módulo de elasticidade e um coeficiente de Poisson, além de outros dados eventuais como densidade, temperatura, etc.

Em casos mais complicados, para novos materiais, pode ser necessário o uso de um conjunto de valores de módulo de elasticidade e coeficientes de Poisson para diversas orientações do material. Mesmo assim, as orientações das propriedades de interesse são, na maioria das vezes, conhecidas ou mesmo controladas a fim de que se ajustem àquelas supostas durante o projeto.

O osso é um desses casos em que existe um conjunto de valores para as propriedades típicas como o módulo de elasticidade e coeficiente de Poisson. No entanto, a diferença é que, para o osso, tanto o valor das propriedades quanto a direção na qual estas apresentam validade podem ser consideradas anisotrópicas, gerando a necessidade de técnicas capazes de determinar ambos os valores da propriedade, ou seja, seu valor e direção de atuação.

No caso do osso, a medicina oferece um subsídio para o desenvolvimento de técnicas para obtenção das propriedades de interesse. As teorias mais recentes dão conta de que o osso é um material que se altera de maneira funcional. Isso significa que tanto as propriedades quanto direções nas quais se mantêm razoavelmente constantes são determinadas por fatores relacionados com as solicitações mecânicas às quais o osso estiver submetido durante o histórico de sua vida.

Isso faz com que as propriedades apresentem grande variação em seu valor absoluto para um pequeno raio de dispersão. Além disso, as linhas de ação de cada valor de propriedade não são uniformemente orientadas, mas sim dispersadas de acordo com a solicitação pontual a que foi submetida a estrutura óssea.

Toda essa teoria permite concluir que um primeiro procedimento para obtenção de propriedades gerais de um osso exige que se possa fazer isso de forma pontual, ou, pelo menos, de maneira a que regiões significativas sejam coletadas para representar o comportamento total do osso. No entanto, tentativas de realizar esse tipo de procedimento têm se mostrado muito custosas e, mesmo assim, ainda existe a variação individual nas características dos ossos, o que dificulta a generalização das propriedades obtidas a partir de uma amostra para todos os casos. Sob esse ponto de vista, o que tem sido feito é tentar definir um conjunto de valores aplicados a determinadas regiões de ossos a fim de reproduzir o comportamento desejado em situações específicas de interesse.

Na verdade, as tentativas são no sentido de tentar desenvolver diversos modelos de osso que sejam capazes de reproduzir, com uma aproximação razoável, uma ampla gama de características encontradas em apenas um tipo de material.

Essa técnica visa fornecer subsídios para os métodos não experimentais, a fim de que possam realizar simulações mais eficientes e próximas da realidade do mundo experimental.

Natali (1989) descreve o tensor de flexibilidade, $[S_{ijrs}]$, proposto para um modelo matemático ortotrópico aplicado a material ósseo

$$[S_{ijrs}]=\begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & -\frac{\nu_{31}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{32}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{13}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{31}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \qquad (1)$$

sendo E_i os módulos de Young, ν_{ij} os coeficientes de Poisson e G_{ij} os módulos de cisalhamento. Todos determinados para cada direção i ou combinação de direções ij do material.

Como pode ser notado, o tensor é bastante genérico. Sua caracterização depende da inserção das propriedades do material. Segundo *Natali* (1989), a determinação dos parâmetros necessários para a caracterização dessa matriz através de testes mecânicos é

problemática. Isso se deve a algumas características dos ensaios a serem realizados com o material ósseo.

Um problema, relacionado com características dos testes, refere-se à obtenção do dos módulos de Young e coeficientes de Poisson. Como pode ser visto na matriz (1), há necessidade de três módulos de Young distintos e quatro coeficientes de Poisson. No entanto, para cada teste com um espécime de material, apenas um valor de módulo de Young e dois coeficientes de Poisson podem ser determinados. Isso não seria um problema, visto que bastaria realizar o teste mais duas vezes e obter todos os valores necessários, não fosse uma característica intrínseca do material ósseo que é a extrema variação de suas propriedades de uma região para outra.

Na prática, seria como efetuar duas análises de materiais distintos e considerar seus resultados como complementares na determinação de propriedades de um mesmo material, mas em direções distintas.

Uma alternativa, proposta pelo autor, é a utilização de testes de ultra-som para determinação dos módulos de Young e coeficientes de Poisson a partir de somente um espécime, no entanto ainda existem controvérsias sobre a validade da aplicação desse tipo de procedimento a materiais não homogêneos.

Outras propriedades de interesse são aquelas relacionadas com a mecânica da fratura de materiais ósseos. Basicamente, o enfoque que tem sido dado é restrito à obtenção de valores de K_{Ic} válidos para o material ósseo sob determinadas condições. Quando se fala em K_{Ic} está se admitindo um comportamento elástico linear do material à fratura, ou seja, a hipótese básica é a de que o material não sofre deformação plástica na ponta da trinca, permitindo, entre outros fatores, que o raio da ponta da trinca permaneça o mesmo, praticamente nulo, durante a propagação. O problema com essa hipótese está no fato de o material ósseo não se comportar como material elástico. Na verdade, existem apenas aproximações para descrever o comportamento desse material, uma das mais aceitas atualmente é a de que o comportamento pode ser entendido como viscoelástico (*Juricic et*

al. (1972), *Knauss* (1970), *Lee* (1996), *Mueller* (1971), *Williams* (1972)) com características de amortecimento importantes.

De qualquer maneira, a abordagem mais utilizada tem sido a de realizar ensaios baseados nas normas ASTM para materiais metálicos, tais como a ASTM E399, a partir de corpos de prova padronizados do tipo CT confeccionados com material ósseo. A figura 1 mostra uma montagem utilizada para esse fim.

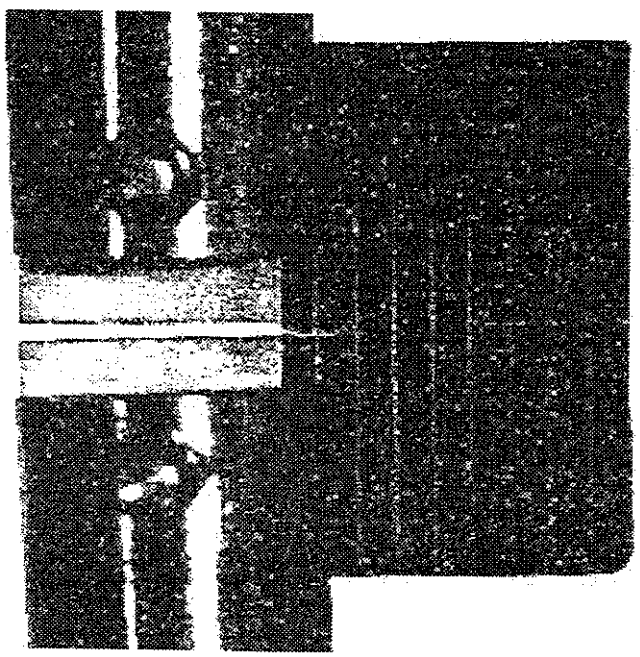


Figura 1 - Espécime do tipo compacto montado em dispositivo para ensaio de tração. (*Bonfield et al.* (1978))

Esse tipo de abordagem tem produzido resultados diversos.

Wright (1977) realizou experimentos com amostras de osso bovino, proveniente da região média diafisária de um fêmur de animal adulto, transformadas em corpos de prova compactos segundo norma ASTM E399-70T, dispondo as fibras em orientação longitudinal. Admitindo a validade da mecânica da fratura elástica linear para o osso em

questão, o objetivo do ensaio realizado foi obter o parâmetro de intensidade de tensão K_{Ic} e sua variação com parâmetros como a densidade do material.

Esse procedimento gerou resultados como os apresentados na Tabela 1.

Tabela 1. Alguns valores de K_{Ic} de acordo com a espessura do corpo de prova e a densidade do material. (*Wright et al.* (1977))

Espessura (cm)	Densidade (g/cm^3)	K_{Ic} ($MNm^{-3/2}$)	G_c ($N\text{-}m/m^2$)
0,190	1,956	3,219	936,1
0,190	2,016	4,356	1862,2
0,312	1,949	3,131	1061,7
0,312	2,012	4,019	1422,2

Como pode ser visto, os valores apresentam comportamento condizente com a teoria da mecânica da fratura elástica linear, uma vez que os valores de K_{Ic} mostram diminuição com o aumento da espessura do corpo de prova. Por outro lado, o aumento na densidade para uma mesma espessura levou a um aumento nos valores de K_{Ic}. Esse tipo de resultado pode ser bastante atraente, visto que abre um caminho para relacionar perda de cálcio dos ossos, identificada através da redução da densidade do material, com uma maior fragilidade, ao que se chama, em medicina, de osteoporose (*Etchebehere* (1998)).

Por outro lado, *Melvin* (1993) faz uma crítica a vários trabalhos sobre determinação de parâmetros da mecânica da fratura em ossos, entre eles o trabalho de *Wright* (1977). Uma das críticas de Melvin refere-se ao caminho seguido pela trinca que pode ter diversas orientações, de acordo com a disposição das linhas de maior resistência do material, formadas por feixes tubulares de colágeno reforçados por material mineral, em relação ao eixo de solicitação do ensaio. Isso pode ser visto na figura 2.

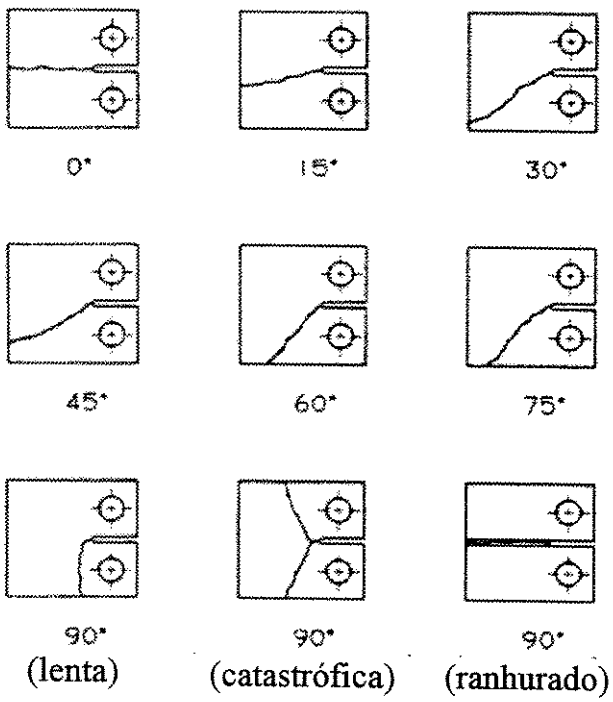


Figura 2 – Perfis de fratura de corpos de prova compactos, testados com a orientação inicial da trinca variando desde longitudinal até transversal em relação ao eixo microestrutural do osso cortical. (Melvin (1993))

2.1.2 **Abordagem matemática**

Mais recentemente, a abordagem matemática por métodos numéricos, conjugada com o uso de recursos computacionais, tem sido aplicada a diversas situações em materiais ósseos. Atuando de forma complementar à abordagem experimental, a abordagem numérica procura fazer uso de teorias e modelos matemáticos para descrever o comportamento do material ósseo em determinadas situações, tais como solicitações dinâmicas para propagação de trincas ou remodelamento, ou solicitações estáticas para verificação de resistência a esforços promovidos por próteses implantadas e afins.

Geralmente, as teorias e os modelos matemáticos são bastante genéricos, sendo aplicados a uma gama de materiais diferentes. Para promover a particularização do modelo ao material ósseo são utilizados os valores das propriedades levantados nos ensaios e testes elaborados pela abordagem experimental.

Do ponto de vista computacional dois métodos têm sido utilizados para a geração de modelos numéricos: o método dos elementos finitos e o método dos elementos de contorno.

O mais popular e conhecido no meio da engenharia é o método dos elementos finitos, que tem encontrado aplicações na área médica em simulações do comportamento mecânico de partes ósseas quando submetidas a esforços determinados ou em interação com implantes ortopédicos. Um exemplo desse tipo de trabalho foi desenvolvido por *Lotz et al.* (1991), no qual se realizou uma análise por elementos finitos da geometria da cabeça do fêmur quando submetida a esforços do tipo impacto provocado por quedas. Numa primeira etapa, foi proposto um modelo linear para o material ósseo, admitindo propriedades médias entre aquelas verificadas no material ósseo verdadeiro. Numa segunda etapa foi proposto um comportamento não linear para o material, com a finalidade de tentar representar as diferenças de comportamento entre o osso cortical e o trabecular. Os resultados mostraram uma boa aproximação do modelo linear no que diz respeito ao comportamento de resistência do osso como um todo. No entanto, para características mais específicas como a deformação das superfícies externas, que pode ser medida no osso real a partir de extensômetros, os resultados foram pobres. Quanto ao modelo não linear, mostrou precisão tanto na determinação da carga de escoamento quanto na de fratura em relação aos valores obtidos com as amostras por métodos experimentais. No entanto, o custo computacional desse tipo de análise mostrou-se muito maior que aquele exibido pela análise linear.

Outro método cuja utilização tem crescido nos últimos anos, principalmente devido ao advento de processadores e computadores mais rápidos e baratos, é o dos elementos de

contorno. Segundo esse método o modelo é descrito apenas por seu contorno externo. Essa característica tem levado os pesquisadores a desenvolverem modelos numéricos, nesse método, mais voltados a analisar os fenômenos característicos do contorno.

Um desses fenômenos é o do remodelamento, que consiste no crescimento do osso sob um determinado tipo de exigência, seja ela advinda da utilização natural, ou da presença de um implante, ou de um dispositivo de síntese para recuperação.

Sadegh et al. (1993) propôs um método numérico baseado em equações integrais de contorno, aplicando o método dos elementos de contorno, para predizer o comportamento do material ósseo quanto ao crescimento sob determinadas condições. Os resultados mostraram que o modelo foi adequado para prever a forma final do osso maduro, após finalizado o processo de crescimento, no entanto, qualquer comparação com estágios intermediários não foi válida.

Wolfe (1993), desenvolveu um trabalho de análise de tensão para implantes endósseos utilizando o método dos elementos de contorno. Nesse trabalho, foi realizada uma comparação com o método dos elementos finitos, concluindo que ambos oferecem resultados numéricos semelhantes, mas com uma maior facilidade para o método dos elementos de contorno, devida ao fato de necessitar da discretização apenas do contorno, o que também facilita alterações posteriores na geometria.

2.2 **O material ósseo**

Quando se fala em material ósseo está se falando de uma conjugação muito específica de materiais que, juntos, atribuem características bastante especiais ao osso. Por exemplo, um osso longo como o fêmur, figura 3, localizado entre os potentes músculos da coxa e responsável por grande parte da amplitude do movimento de andar,

possui uma junção articular que pode ser considerada uniaxial com o tíbia, no joelho, e uma junção articular do tipo esférica com a bacia. Isso permite que o fêmur desempenhe suas funções principais como parte do mecanismo de locomoção do corpo e absorção de impactos ao solo. No entanto, essas funções não dependem apenas da geometria externa desse osso ou de seu inter-relacionamento com os ossos vizinhos, depende também de suas características singulares como material.

A estrutura básica de um grande osso desse tipo é uma camada externa, chamada de osso cortical, bastante compacta e de alta rigidez. Internamente a essa camada pode existir um vazio preenchido por fluido ou uma estrutura em forma de esponja, constituída de minúsculas trabéculas, que são como pequenas treliças, também preenchida por fluido, formando o que se chama de osso trabecular. Então, o fêmur é constituído de apenas um material, mas com propriedades que o tornam capaz de resistir aos esforços de acionamento dos músculos, da sustentação do peso durante o desequilíbrio do passo e aos impactos envolvidos no processo.

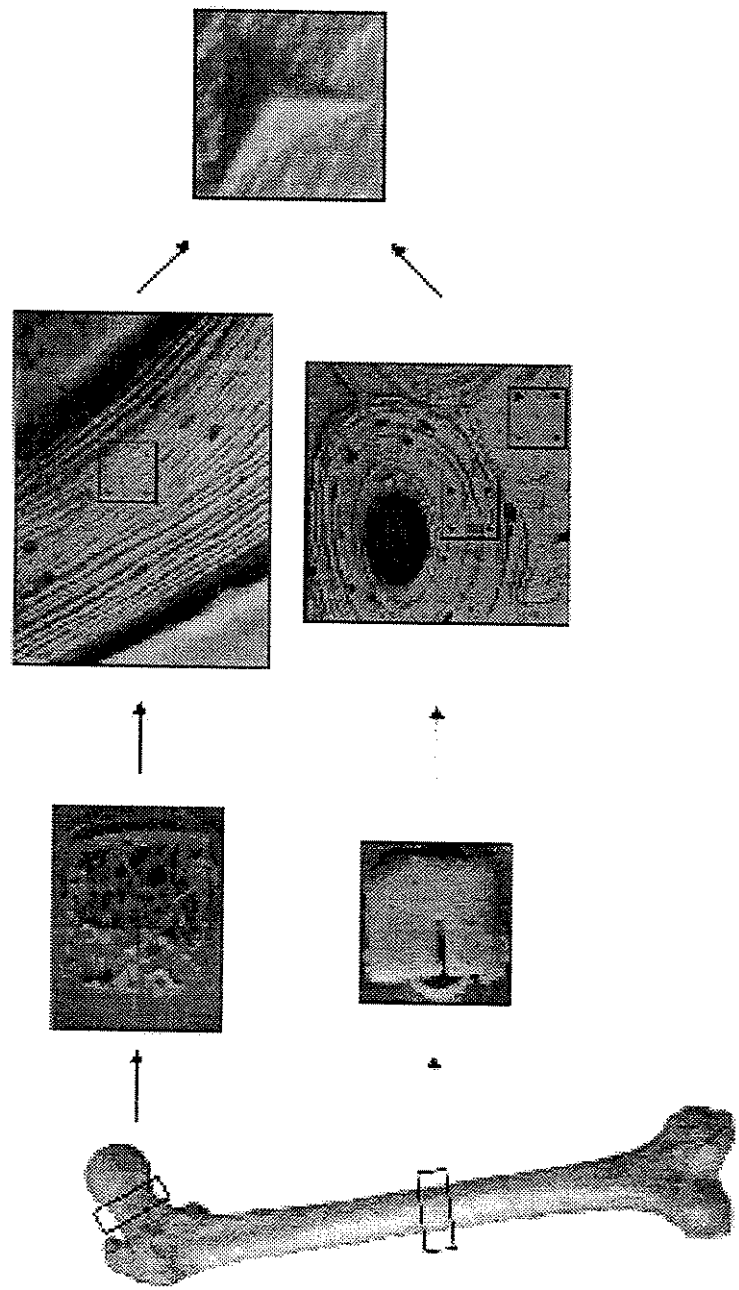


Figura 3. Representação do fêmur (*Zysset et al.* (1999))

Dessa forma, apenas esse osso, que pode ser considerado simples, precisa de duas características. Amortecimento, a fim de absorver os impactos, impedindo que sejam transmitidos diretamente para tecidos mais moles. Resistência mecânica, pois o ato de andar envolve o desequilíbrio do centro de gravidade do corpo, gerando esforços nas

estruturas responsáveis por evitar a queda e recolocar o centro de gravidade em sua posição de equilíbrio. Essas duas características distintas, são obtidas através da combinação das estruturas de osso cortical e trabecular.

O arranjo dessas estruturas no osso é feito de acordo com a solicitação a que está sujeita a região. Assim, o osso cortical, de alta resistência mecânica, está presente em toda a parte externa, mas em maior quantidade nas regiões de maior solicitação mecânica, tais como na região central do osso, formando grossas paredes tubulares. Já o osso trabecular, com alta capacidade de absorção de impactos, está presente nas regiões de proximidade com as articulações, pontos nos quais haverá transmissão de impacto. Isso significa que o fêmur apresenta duas partes distais, compostas por finas camadas de osso cortical preenchidas por osso trabecular, com a finalidade de absorver impactos e uma região proximal, composta por uma camada espessa de osso cortical e um vazio interno, adequada à resistência mecânica de forma semelhante a um tubo de paredes grossas.

Essa é uma maneira bastante simplificada de entender o osso. Na verdade, apesar da aparente homogeneidade de partes compactas, como é o osso cortical, o material apresenta heterogeneidades importantes, tais como vazios e a existência de partículas diversas, muitas delas de origem orgânica, as quais interferem de maneira fundamental no comportamento do material.

Sob o ponto de vista da engenharia, o material ósseo pode ser visto como análogo a um material compósito. No osso, a matriz flexível é composta por fibras de colágeno, as quais são reforçadas com sais de cálcio para se obter a resistência mecânica. A questão é que o material ósseo não é tão simples quanto o compósito utilizado numa aplicação de engenharia. Enquanto o compósito apresenta propriedades, como o módulo de elasticidade, bem definidas e razoavelmente distribuídas por todo o material, o osso mostra variação pontual significativa em suas propriedades, além de não apresentar distribuição uniforme por todo o material. Essa variação de propriedades já foi demonstrada por *Lotz et al.* (1991), em seus estudos sobre fratura em ossos usando

modelos de elementos finitos. Também *Natali* (1989), em artigo sobre as propriedades biomecânicas do osso, descreveu as dificuldades em se obter tais valores como devidas à heterogeneidade e anisotropia do material, sendo que os resultados experimentais mostraram forte dependência da localização anatômica do espécime analisado.

Características biológicas individuais, tais como raça, sexo, idade, nível de atividade durante a vida, entre outras também são apontadas por *Natali* (1989) como fatores causadores das variações de propriedades entre ossos de pessoas diferentes. Isso pode ser exemplificado através do gráfico da figura 4.

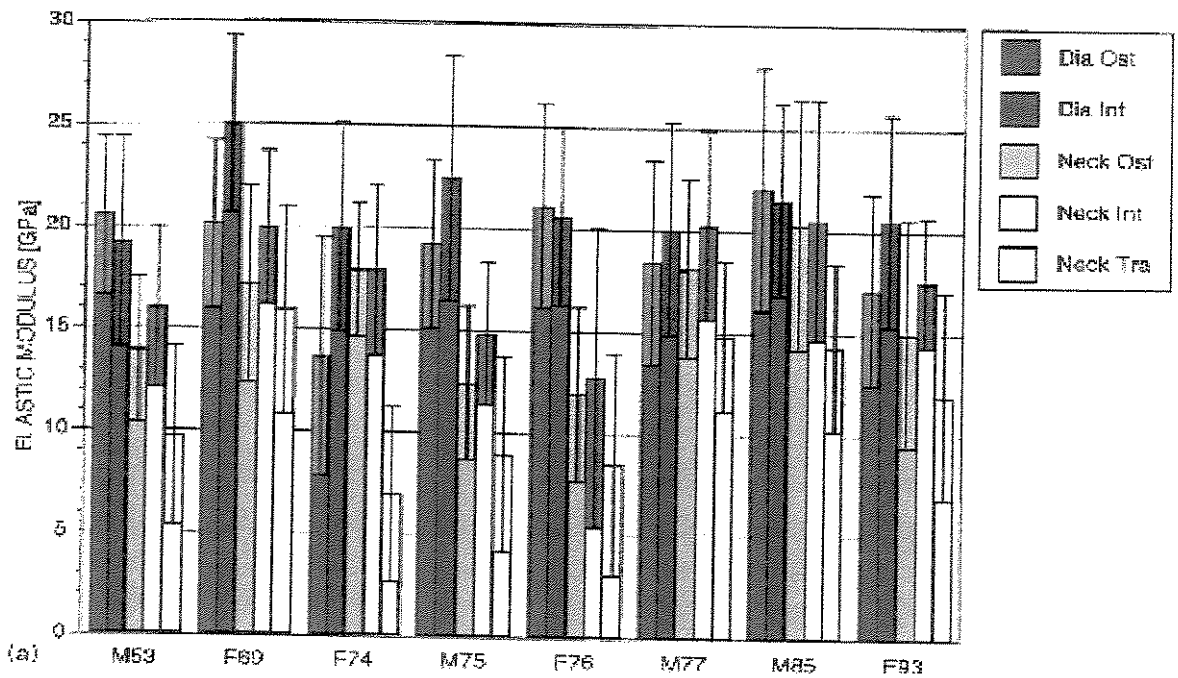


Figura 4. Gráfico contendo variação do módulo de elasticidade de acordo com a região óssea, para pessoas diferentes, do fêmur (*Zysset et al.*(1999)).

No gráfico da figura 4 o eixo das ordenadas mostra valores de módulo de elasticidade, enquanto as abscissas são divididas de acordo com as pessoas, indicando sexo e idade (M53 é masculino de 53 anos). Já as colunas agrupadas, indicando valores

médios dos módulos de elasticidade obtidos, são referentes à região no osso de onde foi retirada a amostra para teste.

Além disso, existem duas características dos ossos que não são comuns aos materiais de utilização típica em engenharia, que são a capacidade de se regenerarem e a possibilidade de variação na estrutura e nas características mecânicas de acordo com as solicitações às quais estão submetidos.

De maneira geral, a teoria mais aceita no meio médico para a variação das características mecânicas do osso é de que existe um mecanismo de deposição seletiva de cálcio sobre as fibras de colágeno. Esse mecanismo é baseado num efeito piezelétrico verificado nas moléculas de colágeno.

O colágeno é uma proteína natural, presente nas cartilagens do corpo humano, e tem como principal característica a alta flexibilidade. Como todas as proteínas, o colágeno é constituído de moléculas longas, formando cadeias que se interligam para formar fibras. Teorias propõe que as moléculas de colágeno, devido à forma geométrica peculiar de suas ligações, possuem um efeito piezelétrico (*Köberle* (1974)), gerando um potencial elétrico quando sofrem deformação. Segundo estudos, esse potencial elétrico funcionaria como um sinal para que células responsáveis pela deposição de sais de cálcio reforçassem as fibras de colágeno que sofrem ciclos de deformação mais freqüentes, aumentando a rigidez das mesmas, adaptando o osso para esse tipo de esforço. Além disso, efeitos hormonais e melhoria na circulação sangüínea, ocasionados pela atividade física, também são fundamentais para os processos de formação do osso.

Esse tipo de teoria pode explicar formações ósseas detectadas através de técnicas de radiografia, nas quais aparecem linhas mais escuras, correspondentes às fibras de colágeno com maior reforço de cálcio, alinhadas com as linhas de tensão obtidas através de cálculos supondo esforços como caminhar, correr ou saltar.

O osso é um material vivo, um tecido que, como quase todos os outros tecidos vivos, é capaz de se alterar e regenerar. Isso significa que uma estrutura de material ósseo vivo é capaz de alterar sua forma, além de alterar suas propriedades, ao que se dá o nome de remodelamento ósseo, além de ter a capacidade de se regenerar.

De forma simplificada e sob um ponto de vista de engenharia, o remodelamento ósseo pode ser visto como uma técnica de otimização estrutural. Nessa técnica existem determinadas condições que levam à formação de material ósseo e outras que levam ao desaparecimento, chamado de reabsorção. Dessa forma, um campo de tensões mais intenso pode funcionar como estímulo para a formação de material ósseo na região, a fim de que esse campo de tensões sofra uma redistribuição, melhorando as solicitações sobre o material como um todo. Esse mesmo fenômeno pode levar à reabsorção de material ósseo em outra região, caso o perfil de tensões torne-se pouco expressivo ali. Por outro lado, solicitações muito intensas, superiores à capacidade do material e à velocidade de alteração possível para a geometria, podem levar a uma reabsorção por morte do tecido, indicativo de que foram superados os limites de resposta do material ósseo vivo.

Quanto à regeneração, do ponto de vista da engenharia pode ser vista com uma atitude de manutenção preventiva, ou seja, antes que ocorra uma falha catastrófica ou que seja necessário substituir uma estrutura para que o todo continue funcionando, as células monitoram todas as partes e reparam pequenos defeitos, tais como trincas e fissuras. Apesar disso, falhas catastróficas, caracterizadas por fraturas da estrutura óssea, podem ocorrer. Nesses casos, desde que sejam dadas condições mínimas de estabilidade e alinhamento para as partes ósseas afetadas, a regeneração óssea será capaz de reparar a fratura, restabelecendo a integridade estrutural da parte afetada.

Portanto, numa primeira análise, simplificada e bastante voltada para a engenharia, pode ser notado que o material ósseo apresenta uma série de características que o tornam muito diferente de materiais tipicamente tratados. Em primeiro lugar, apesar da semelhança com um material compósito, que já não é dos mais simples da engenharia,

apresenta características de viscoelasticidade localizada e variação no módulo de elasticidade ao longo do material, graças ao mecanismo de reforço seletivo. Sua geometria pode sofrer alterações de acordo com o perfil de tensões, graças ao remodelamento, para que possa suportar melhor os esforços a que está mais comumente submetida. Enfim, tem a capacidade de se regenerar, eliminando trincas e mesmo reparando fraturas totais da estrutura.

A título meramente comparativo é possível confrontar as características vistas para o material ósseo com as mesmas de um material de uso típico em engenharia como, por exemplo, um material compósito feito de matriz epóxi reforçada com fibras de carbono em camadas, cada uma colocada ortogonalmente em relação à outra.

Para esse material, pode se considerar comportamento macroscópico elástico, com variação do módulo de elasticidade em apenas duas direções, correspondentes às direções nas quais existe o reforço de fibras de carbono. No material ósseo, a variação de módulo de elasticidade pode ocorrer em diversas direções e depois que o material já está em uso, o que não pode ocorrer com o compósito citado.

O material compósito precisa ser modelado de acordo com esforços conhecidos ou previstos em projeto, para a estrutura da qual faz parte. Alterações nos esforços atuantes na estrutura são aceitáveis dentro de parâmetros contemplados no projeto como coeficientes de segurança. No material ósseo, a estrutura sofre remodelamento contínuo para que apresente sempre na geometria ótima em vista dos esforços a que está submetida.

Finalmente, um material compósito, uma vez pronto e conformado de acordo com sua aplicação, não tem a capacidade de se auto reparar, ficando sujeito ao desgaste e degradação causados pelo tempo e uso. No caso do material ósseo, existe a regeneração, característica de uma material vivo, que permite que, dentro de certos limites, o material seja recuperado de seus defeitos e permaneça funcional.

Assim, a heterogeneidade presente no material ósseo representa enorme dificuldade para a obtenção de um modelo matemático único, capaz de descrever, com razoável aproximação, os fenômenos observados em estruturas ósseas reais. A obtenção de um modelo matemático é necessária se o objetivo é utilizar um método de simulação computacional, como o método dos elementos de contorno ou o método dos elementos finitos.

2.3 **O método de cálculo**

No caso de aplicações de engenharia como uma análise de tensões, o material utilizado e seu comportamento são bastante importantes, uma vez que ajudam na determinação do tipo de metodologia de cálculo que pode ser adotada e a complexidade da mesmo. Outro fator que influi nessa escolha é a complexidade da geometria envolvida.

Para geometrias simples, tais como barras ou vigas, submetidas a condições de contorno bem definidas e bem colocadas, utilizando um material de comportamento simples, ou seja, que possa ser considerado matematicamente homogêneo, linear e isotrópico, uma abordagem matemática do tipo analítica pode ser suficiente para obter os resultados desejados.

No entanto, no caso da bioengenharia aplicada a ossos, a geometria envolvida é bastante complexa, muitas vezes sem simetrias, como pode ser visto na figura 3, e o material dificilmente se adapta a um modelo matemático que pode ser considerado de comportamento simples. Na verdade, o material ósseo poderia ser melhor representado por um modelo anisotrópico e viscoelástico, dependendo da região.

Assim, fica claro que não é conveniente utilizar uma abordagem matemática do tipo analítica para obter resultados interessantes em bioengenharia, pois envolveria uma

quantidade muito grande de simplificações, o que tornaria a análise muito distante do comportamento observado na realidade e que se deseja reproduzir. Assim, para obter resultados sem fazer tantas simplificações, é necessário utilizar um método de cálculo capaz de suportar modelos matemáticos aplicados a geometrias complexas, o que significa, de modo geral, utilizar uma técnica de cálculo numérico.

Atualmente, dois métodos de cálculo numérico se destacam para aplicações computacionais em mecânica. Um fundamentado em aproximações numéricas feitas no domínio, chamado de método dos elementos finitos. Outro com base em aproximações numéricas nas fronteiras do domínio, chamado de método dos elementos de contorno. Ambos os métodos apóiam-se em equações integrais que descrevem o problema de elasticidade. Portanto, tanto em um caso quanto no outro é necessário transformar as equações diferenciais de Navier para elasticidade em equações integrais.

No caso do método dos elementos finitos a integral utilizada é de domínio, o que significa que depende, para uma avaliação numérica, de um processo de aproximação que ocorre no domínio, geralmente feito através da divisão da geometria complexa do domínio real em domínios menores, chamados de elementos finitos, cuja geometria, apesar de poder ser variável, obedece sempre a uma função conhecida, que permite ao método fazer as estimativas necessárias para a efetivação do cálculo.

Um tipo de integral de domínio, semelhante àquela utilizada pelo método dos elementos finitos, pode ser trabalhada matematicamente até se transformar numa integral equivalente de contorno. Isto é, uma integral que pode ser avaliada apenas no contorno, mas que apresenta uma correlação com o domínio de maneira a possibilitar a obtenção de resultados em qualquer ponto do domínio, uma vez resolvida no contorno. Esse procedimento descreve o método dos elementos de contorno. Segundo essa técnica, para avaliar numericamente a integral de contorno, é necessário que sejam realizadas aproximações no contorno, o que é feito, geralmente, a partir da divisão do mesmo em pequenas superfícies chamadas elementos de contorno, para os quais a geometria torna-se

conhecida ao variar de acordo com funções determinadas, o que possibilita que o método realize os cálculos necessários para o prosseguimento da análise. Para o desenvolvimento desse trabalho foi escolhido o método dos elementos de contorno.

Um motivo para a escolha desse método foi sua característica de necessitar da discretização, que é a colocação de uma malha de elementos para realização das aproximações numéricas, apenas no contorno. No caso de um problema tridimensional isso significa que a malha de elementos de contorno assume um caráter de superfície tridimensional, ao invés de volume, reduzindo o tamanho do problema em uma dimensão. Isso pode ser interessante do ponto de vista de tamanho do problema, uma vez que as matrizes envolvidas podem ficar menores.

Outro motivo que levou à escolha do método dos elementos de contorno foi o fato de que é capaz de calcular o resultado exclusivamente no contorno, fato que pode tornar as análises mais rápidas, uma vez que, no caso de problemas de bioengenharia aplicados a ossos, muitos dos fenômenos estudados ocorrem no contorno ou dependem de grandezas que estão presentes exclusivamente no contorno, como é o caso do remodelamento, que ocorre alterando o contorno externo do osso em função das tensões que estão presentes nesse mesmo contorno.

Além desses motivos podem ser listados:

- maior perspectiva de pesquisa em comparação com o método dos elementos finitos, dada a recente evolução do método, que vem sendo impulsionada pelos avanços na área de informática;

- interesse em pesquisa interna, principalmente no sentido de verificar a eficiência de algoritmos como *RISP* e técnicas como a reutilização das matrizes H e G para casos em ocorre uma reavaliação do modelo sem alterações na geometria;

- contribuição para a formação de uma base acadêmica através da disponibilização de um código em elementos de contorno capaz de resolver

problemas tridimensionais voltado para a pesquisa, ou seja, dispondo de um código aberto, que permita alterações e adições de novas implementações em desenvolvimento.

Dessa forma, pensando numa formulação por elementos de contorno, é possível transformar o problema diferencial, representado pela equação de Navier, para uma formulação em termos de equações integrais, o que é mostrado por *Kane* (1994). Para isso, inicialmente é preciso definir matematicamente o tipo de problema a ser resolvido. Isso significa saber qual tipo de equação ou equações matemáticas estão envolvidas.

Para determinar o equacionamento do problema é necessário observá-lo como uma aplicação de engenharia, ou seja, classificar a bioengenharia aplicada a ossos numa das categorias da engenharia.

Como se trata de um análise de esforços, que envolve análise de tensões em ossos, uma teoria adequada para início de trabalho é a da elasticidade, que trata das relações entre tensão, deformação, tração e deslocamentos no regime elástico do material.

Dentro da elasticidade existem relações matemáticas que regem o comportamento dos materiais quando submetidos a solicitações externas, relacionando os esforços aplicados ao corpo em análise com suas respostas em tensão ou deslocamentos. No caso de um material isotrópico, homogêneo e linear, essas relações podem ser dadas pelos três grupos de equações seguintes:

$$e_{ij} = \frac{1}{2} \cdot \left(u_{i,j} + u_{j,i} \right) \tag{2}$$

$$\sigma_{ij} = \lambda \cdot \delta_{ij} \cdot e_{kk} + 2 \cdot \mu \cdot e_{ij} \tag{3}$$

$$\sigma_{ji,j} + f_i = 0 \tag{4}$$

A partir dessas relações entre deformação (e_{ij}) e deslocamento (u_i), equação 2, tensão (σ_{ij}) e deformação, equação 3, e de uma equação de equilíbrio entre as tensões e as forças externas (f_i), equação 4, para materiais isotrópicos é possível obter um sistema de equações diferenciais de quinze incógnitas a quinze equações, como mostrado por *Kane* (1994). Manipulando essas equações e reorganizando-as, é possível obter as equações de equilíbrio chamadas de equações de Navier para elasticidade:

$$\mu \cdot u_{i,jj} + (\mu + \lambda) \cdot u_{j,ji} + f_i = 0 \qquad (5)$$

É necessário saber que as equações de Navier para elasticidade são três equações diferenciais parciais de segunda ordem acopladas em relação aos deslocamentos u_i. Isso significa que a solução para um deslocamento depende e afeta o resultado para os outros. Existem técnicas matemáticas para desacoplar essas três equações de Navier, uma delas utiliza um vetor de expressões envolvendo derivadas segundas de um vetor chamado de vetor de Galerkin, representado por g, para substituir os componentes u_i da equação de Navier. A expressão envolvendo o vetor de Galerkin fica dada por:

$$2 \cdot \mu \cdot u_i = c \cdot g_{i,jj} - g_{j,ji} \qquad (6)$$

Substituindo λ e μ, as constantes de Lamé, por seus respectivos valores de acordo com a teoria da elasticidade e utilizando a equação 6, é possível reescrever a equação 5 em função do vetor de Galerkin, de uma constante c e de uma propriedade do material ν

$$\frac{c}{2} \cdot g_{i,kkjj} + g_{k,kjji} \cdot \left\{ -\frac{1}{2} + \frac{c}{2 \cdot (1 - 2 \cdot \nu)} - \frac{1}{2 \cdot (1 - 2 \cdot \nu)} \right\} + f_i = 0 \qquad (7)$$

nessa equação, pode se fazer uso do fato de que, para o vetor de Galerkin, as derivações $g_{k,kijj}$, $g_{j,kkji}$ e $g_{k,kjji}$ são iguais. Assim, é possível determinar a constante c como aquela que faz com que o valor entre chaves se anule.

$$-\frac{1}{2} + \frac{c}{2 \cdot (1 - 2 \cdot \nu)} - \frac{1}{2 \cdot (1 - 2 \cdot \nu)} = 0$$

$$c = 2 \cdot (1 - \nu) \tag{8}$$

Dessa forma, a equação de equilíbrio, equação 5, descrita em termos do vetor de Galerkin fica dada por:

$$(1 - \nu) \cdot g_{i,kkjj} + f_i = 0 \tag{9}$$

O conjunto de três equações descrito na equação 9 forma um sistema de equações diferenciais parciais de quarta ordem desacopladas, o que torna possível sua solução aplicada a um problema de elasticidade. Uma vez resolvido esse sistema é possível obter a solução para deslocamentos através do uso da equação 6 substituído o valor da constante c previamente calculado, o que resulta na equação seguinte:

$$2 \cdot \mu \cdot u_i = 2 \cdot (1 - \nu) \cdot g_{i,jj} - g_{j,ji} \tag{10}$$

O mesmo tratamento pode ser utilizado para obter resultados referentes à tensão, bastando aplicar a equação 10 na relação entre deformação e deslocamento e usar o resultado desse procedimento para transformar a relação entre tensão e deformação numa relação entre tensão e o vetor de Galerkin. O desenvolvimento desse tratamento matemático pode ser visto com mais detalhe em *Kane* (1994). Sua forma final é apresentada à seguir:

$$\sigma_{ij} = (1 - \nu) \cdot \left[g_{i,kkj} + g_{j,kki} \right] - g_{k,kij} + \nu \cdot \delta_{ij} \cdot g_{l,lkk} \tag{11}$$

A equação 11 pode ser aplicada para resolver o problema de Kelvin em elasticidade tridimensional, que se resume à resposta para o caso de uma carga concentrada pontual aplicada em um domínio infinito elástico tridimensional, homogêneo e isotrópico. A solução desse problema, descrita em detalhe por *Kane* (1994), resulta em três soluções, chamadas de soluções fundamentais, uma para deslocamento, uma para tensão e outra para força de superfície, que são mostradas à seguir:

$$u_{ik} = \frac{P}{16\mu\pi(1-\nu)}\rho^{-1}\left\{(3-4\nu)\delta_{ik} + \rho_{,i}\rho_{,k} \right\} \quad (12)$$

$$\sigma_{ijk} = \frac{P\cdot(1-2\nu)}{8\pi(1-\nu)}\rho^{-2}\left\{\delta_{ij}\rho_{,k} - \delta_{jk}\rho_{,i} - \delta_{ki}\rho_{,j} - \frac{3}{1-2\nu}\rho_{,i}\rho_{,j}\rho_{,k}\right\} \quad (13)$$

$$t_{jk} = \frac{-P}{8\pi(1-\nu)}\rho^{-2}\left\{\rho_{,j}n_j\left[(1-2\nu)\delta_{ki} + 3\rho_{,i}\rho_{,k}\right] + (1-2\nu)\left[\rho_{,i}n_k - \rho_{,k}n_i\right]\right\} \quad (14)$$

nessas equações os índices k representam o ponto de colocação da carga pontual em relação ao qual se está avaliando a solução, uma vez que esse tipo de resposta pode ser obtida para várias localizações do ponto de carregamento em relação ao ponto de observação. O símbolo P representa o carregamento concentrado aplicado no ponto de carregamento e ρ representa a distância entre esse ponto e o ponto de observação.

Essas soluções serão úteis após o desenvolvimento da equação integral de contorno que rege o problema de elasticidade tridimensional. Para desenvolver essa equação é necessário partir da equação de equilíbrio entre tensões internas e forças externas aplicadas, equação 4. Todo o procedimento de conversão dessa equação em uma equação integral de contorno foi descrito em detalhe por *Kane* (1994). De maneira simplificada, o que se faz é aplicar o princípio dos trabalhos virtuais à equação 4, o que transforma a equação inicial numa equação integral que mistura integrais de contorno com integrais de domínio. Em seguida, aplica-se o teorema da reciprocidade, compatibilizando as variáveis na equação ao substituir tensão e deformação por força de superfície, deslocamento e forças externas.

Até agora o que se tem é uma equação integral que mistura integrais de contorno e domínio e que relaciona duas situações de equilíbrio em dois sistemas diferentes:

$$\int_{\Gamma} t_i^{(1)} u_i^{(2)} d\Gamma + \int_{\Omega} f_i^{(1)} u_i^{(2)} d\Omega = \int_{\Gamma} t_i^{(2)} u_i^{(1)} d\Gamma + \int_{\Omega} f_i^{(2)} u_i^{(1)} d\Omega \quad (15)$$

sendo que os números sobrescritos entre parênteses denotam os dois sistemas, os índices i as três coordenadas, Γ representa o contorno e Ω representa o domínio.

Um dos sistemas presentes na equação integral é o sistema no qual se deseja resolver o problema, assim, resta um sistema para o qual é necessário dispor de soluções válidas para o problema de elasticidade. Ora, como as soluções fundamentais foram determinadas a partir da mesma equação de equilíbrio e são soluções para o problema de elasticidade, elas serão as utilizadas como soluções do segundo sistema.

Assim, a equação integral resultante fica:

$$\int_{\Gamma} t_i \, u_{ik} \, d\Gamma + \int_{\Omega} f_i \, u_{ik} \, d\Omega = \int_{\Gamma} t_{ik} u_i \, d\Gamma + \int_{\Omega} \delta_{ik}(x-d) \cdot u_i^{(1)} d\Omega \qquad (16)$$

Trocando a função delta de Dirac por um valor c_{ik}, que varia de acordo com a distância entre o ponto d de aplicação da carga pontual e o contorno, sendo zero para o caso em que d fica fora do contorno, um valor entre zero e um quando está sobre o contorno e um quando está no domínio.

Note-se ainda que a equação integral apresenta uma integral de domínio, relativa às forças externas e a solução fundamental de deslocamento. Embora seja plenamente calculável, não foram abordados métodos de tratamento para forças de corpo, que são representadas por essa integral de domínio. Portanto, para esse trabalho, serão consideradas inexpressivas as forças de corpo, o que faz com que essa integral possa ser desprezada, degenerando a equação 16 para seu formato final de equação integral de contorno para o problema de tensão, chamada de Identidade de Somigliana:

$$\int_{\Gamma} t_i \, u_{ik} \, d\Gamma = \int_{\Gamma} t_{ik} u_i \, d\Gamma + c_{ik} \cdot u_i(d) \qquad (17)$$

A partir dessa equação integral de contorno, avaliada numericamente por um código de integração numérica, é possível encontrar as respostas em força de superfície e

deslocamento para um problema de equilíbrio estático dadas determinadas condições de contorno aplicadas a um sólido considerado homogêneo, linear, isotrópico e contínuo.

2.4 **Implementação numérica computacional**

Uma vez terminado o equacionamento matemático do problema genérico, o que compreendeu o uso de todos os modelos matemáticos desenvolvidos para o problema, é necessário criar uma ferramenta de cálculo que seja capaz de avaliar numericamente o desenvolvimento matemático a fim de obter resultados particulares para cada situação que se deseje verificar.

Avaliar numericamente um equacionamento matemático significa desenvolver um algoritmo capaz de manipular dados provenientes de um banco de dados de maneira a conseguir correlacioná-los com as informações necessárias para completar as equações matemáticas de modo adequado, a fim de que seja possível isolar as respostas dessas equações e torná-las disponíveis na forma de resultados.

No caso da equação integral de contorno o foco principal, que deve ser contemplado por qualquer algoritmo de solução, é a integração, uma vez que a solução dessas equações envolve a integração de funções bastante complexas.

Devido a essa complexidade nas funções a serem integradas, não é possível obter uma integração analítica, daí o fato de o desenvolvimento matemático chegar até a equação integral de contorno e não até a sua solução na forma de uma equação integrada. Para isso, é necessário utilizar um algoritmo de cálculo numérico, o qual estima aproximadamente o valor numérico da integral. A desvantagem desse método é que oferece um resultado particular, enquanto que um integração analítica ofereceria um

resultado válido para qualquer situação. No entanto, é um procedimento necessário para se obter um resultado em aplicações reais.

Para realizar uma integração numérica é necessário que sejam feitas aproximações, a fim de que seja possível estimar os valores das variáveis envolvidas na integral, de modo que tudo possa ser transformado numa soma de valores. Para tanto, é necessário que seja avaliado o integrando, para se saber quais as variáveis necessárias para realizar seu cálculo numérico.

Além disso, é necessário escolher uma técnica de integração numérica. Segundo comparações mostradas por *Kane* (1994), dentre as várias técnicas de integração numérica existentes, a que tem se mostrado mais eficiente para problemas de equações integrais de contorno é a integração de Gauss. Segundo esse método a integral é avaliada em pontos determinados, para os quais são calculados valores ponderadores chamados pesos de Gauss, os quais afetam o resultado numérico da função representada pelo integrando original. Assim, o que se obtém é um resultado parcial equivalente ao valor da integral para a área de influência do ponto de integração de Gauss, onde é avaliado o peso utilizado. Somando os resultados para todos os pontos de Gauss é obtido o valor numérico da integral dentro dos limites dos pontos de Gauss.

Esse procedimento consegue oferecer resultados exatos para integrandos que sejam polinomiais, dependendo apenas do número de pontos de Gauss para que isso ocorra. No caso da equação integral de contorno, o integrando não é uma função polinomial, portanto, o método de integração apresentará um erro e seus resultados constituirão uma aproximação do resultado real. O erro desse procedimento de cálculo pode ser avaliado para o caso de utilização a funções não polinomiais,

$$E = \frac{f^{(2N+2)}(\alpha)}{(2N+2)!\cdot K_{N+1}^2} \tag{18}$$

sendo α um ponto qualquer no domínio do intervalo de integração e K uma constante numérica.

Dessa forma, para que seja possível calcular a integral por meio da técnica de integração de Gauss, é necessário, em primeiro lugar, escolher a quantidade de pontos de integração que será utilizado. Uma vez determinado isso, é necessário que sejam calculadas as coordenadas dos pontos de Gauss e seus respectivos pesos, considerando uma função polinomial como integrando. Isso não consiste em problema, pois são procedimentos independentes da equação integral de contorno. Além disso, é preciso avaliar numericamente o integrando da equação integral de contorno.

Como mostrado na equação 17 as integrais de contorno que constituem a equação integral de contorno para o problema de elasticidade envolvem integrandos contendo as soluções fundamentais de força de superfície e deslocamento, além dos graus de liberdade em força de superfície ou deslocamento que se deseja determinar.

Para estimar o valor numérico desses integrandos é necessário obter dados suficientes para avaliar numericamente as soluções fundamentais, ou seja, as equações mostradas em (12) e (14). Isso significa obter valores como a distância entre ponto de aplicação do carregamento concentrado e ponto de observação, derivadas representado a variação das distâncias etc.

Para que seja possível obter esses dados é necessário estabelecer os pontos que servirão de referência para avaliação dessas informação. Isso significa que é preciso discretizar a superfície que representa o contorno, onde a integral será avaliada, para criar matematicamente os pontos a fim de que possam ser obtidas distâncias e todas as outras informações necessárias.

Para discretizar a superfície de contorno são utilizados elementos de contorno, que são pequenas superfícies padrão que se adaptam para interpolar a superfície real do modelo geométrico que se deseja analisar. Esses elementos podem ter diversas formas

geométricas e conter uma quantidade variável de pontos de referência sobre os quais serão interpolados os dados necessários para a análise numérica.

O elemento utilizado para esse trabalho tem forma quadrilateral, o que significa que tem quatro lados que podem se distorcer livremente até que se encontre a forma que melhor se adapte à geometria do modelo que está sendo discretizado. Apresenta oito pontos de referência, chamados de nós, colocados um em cada vértice e um no centro de cada aresta do quadrilátero, o que significa que o elemento é contínuo, ou seja, compartilha seus nós com o elemento vizinho. Isso permite que sejam utilizadas funções de forma quadráticas para interpolação de valores através dos nós do elemento.

Funções de forma são funções que determinam como as grandezas variam ao longo do elemento. No caso da geometria, se são utilizadas funções de forma quadráticas ao longo das arestas do elemento, significa que qualquer ponto da geometria do elemento pode ser encontrado a partir da utilização dessas funções de forma e dos valores nodais do elemento. O processo é semelhante a encontrar um ponto numa posição qualquer de uma parábola definida por sua equação e três pontos.

No caso do elemento quadrilateral de oito nós, contínuo, utilizando funções de forma quadráticas, implementado nesse trabalho, tanto a geometria quanto as demais grandezas físicas, tais como o comportamento das forças de superfície ou dos deslocamentos, são representadas pelas mesmas funções de forma, o que significa que o elemento é do tipo isoparamétrico.

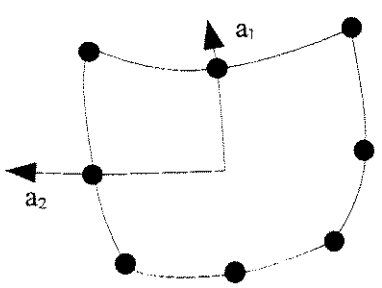


Figura 5. Elemento quadrilateral de oito nós.

Para esse tipo de elemento as funções de forma são dadas por *Kane* (1994), como função de coordenadas intrínsecas a_1 e a_2, que são determinadas a partir de um sistema de coordenadas fixado ao elemento, como mostrado na figura 5. Essas funções de forma são apresentadas à seguir:

$$
\begin{aligned}
h^{(1)} &= \frac{1}{4}(1-a_1)(1-a_2)(-a_1-a_2-1) & \qquad h^{(5)} &= \frac{1}{2}(1-a_1^2)(1-a_2) \\
h^{(2)} &= \frac{1}{4}(1+a_1)(1-a_2)(+a_1-a_2-1) & \qquad h^{(6)} &= \frac{1}{2}(1-a_2^2)(1+a_1) \\
h^{(3)} &= \frac{1}{4}(1+a_1)(1+a_2)(+a_1+a_2-1) & \qquad h^{(7)} &= \frac{1}{2}(1-a_1^2)(1+a_2) \\
h^{(4)} &= \frac{1}{4}(1-a_1)(1+a_2)(-a_1+a_2-1) & \qquad h^{(8)} &= \frac{1}{2}(1-a_2^2)(1-a_1)
\end{aligned}
\tag{19}
$$

A partir do uso dessas funções de forma é possível obter a localização de todas as grandezas sobre o elemento, bem como sua variação. No entanto, pode se notar que as funções de forma são válidas num sistema de coordenadas preso ao elemento, e são sempre as mesmas para representar todos os elementos que possam ser definidos na malha do contorno. Para que isso seja possível é necessário que os elementos de contorno sejam transformados para elementos equivalentes, num sistema de coordenadas parametrizado, o que significa mapear as coordenadas globais dos elementos, definidas segundo o modelo, para coordenadas paramétricas de um elemento genérico. Isso é feito com o uso de uma ferramenta de transformação chamada Jacobiano.

O Jacobiano define uma transformação matemática de uma geometria em um determinado sistema para outra geometria equivalente em outro sistema. No caso dessa aplicação em elementos de contorno, o Jacobiano fará a transformação de uma geometria quadrilateral irregular em um sistema global, utilizado para descrever todos os elementos do problema, para uma geometria quadrilateral regular em um sistema parametrizado, cujas coordenadas sempre são dadas entre –1 e 1.

Isso tudo, aplicado a cada elemento de contorno, possibilita que seja calculado o valor numérico do integrando composto pela solução fundamental, funções de forma e o

Jacobiano. Com isso, para cada elemento, a equação integral pode ser avaliada a menos de uma variável, que consiste no grau de liberdade da equação, que pode ser um deslocamento ou uma força de superfície.

$$\left\{\int_{-1}^{1} t_{ik} h^{(n)} J da\right\}^{(E)} \cdot \left(u_i^{(n)}\right)^{(E)} = H^{(E)} \left(u_i^{(n)}\right)^{(E)} \tag{20}$$

$$\left\{\int_{-1}^{1} u_{ik} h^{(n)} J da\right\}^{(E)} \cdot \left(t_i^{(n)}\right)^{(E)} = G^{(E)} \left(t_i^{(n)}\right)^{(E)} \tag{21}$$

Para cada elemento, em cada um de seus nós, a integração que tem como resultado uma matriz de ordem três. Essa matriz expressa a resposta da equação integral de contorno em dada uma das três direções do problema tridimensional para cada uma das três direções possíveis de aplicação da carga concentrada da solução fundamental.

Realizando a integração para todos os pontos fonte, para os oito nós de todos os elementos existem dois tipos de matrizes que são geradas simultaneamente. Uma que é fruto das integrações para a solução fundamental de deslocamento, a chamada matriz G, outra é construída a partir das integrações para a solução fundamental de força superfície, chamada matriz H. Em forma matricial, a equação integral de contorno, equação 17, pode ser escrita como:

$$\begin{bmatrix} H & \end{bmatrix} \left\{ u \right\} = \begin{bmatrix} & G \ldots & \end{bmatrix} \begin{Bmatrix} t \\ \ldots \end{Bmatrix} \tag{22}$$

Cada uma dessas matrizes é construída a partir das diversas integrações que são feitas para cada elemento em relação a cada ponto fonte considerado para as soluções

fundamentais. Assim, cada coluna representa um resultado de uma integração para um determinado nó de um elemento em uma dada direção. Cada linha representa o conjunto de resultados de integrações por todos os nós do problema em relação a um determinado ponto fonte com o carregamento concentrado colocado numa dada direção. Isso faz com que cada matriz tenha, pelo menos, tantas colunas quantas forem as direções diferentes que cada nó pode ter, ou seja, colunas na mesma quantidade que o número de graus de liberdade do sistema. Como o número de graus de liberdade deve ser acompanhado de igual quantidade de equações independentes para que o sistema tenha solução, é necessário criar linhas nas matrizes de maneira a que se igualem ao número de graus de liberdade, o que se consegue colocando mais pontos fonte.

Isso faz com que a técnica de construção das matrizes tenha que avaliar integrais não somente para um ponto fonte ao longo de todos os elementos, mas sim uma quantidade de pontos fonte igual ao número de nós do problema, o que gerou a prática de integrar colocando os pontos fonte sobre cada nó da malha de contorno de uma vez para gerar todas as linhas das matrizes.

No caso da matriz G, a construção é simples, devendo obedecer à seqüência de se colocar os resultados de acordo com o par (ponto fonte / grau de liberdade, elemento / nó local / grau de liberdade). Essa construção se explica pois a matriz G multiplica os graus de liberdade de força de superfície, que não precisam apresentar compatibilidade quando estão sendo avaliados num nó que é fronteira e pertence a dois elementos. Assim, para cada conjunto de oito nós de cada elemento existirão oito valores de força de superfície diferentes em cada direção e, portanto, vinte e quatro colunas correspondentes na matriz G. Essa maneira de construir pode ser melhor vista na figura 6.

		ELEMENTO 1															...
		nó 1			nó 2			nó 3			...			nó 8			
		X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z	
Pto. Fonte 1	X																
	Y																
	Z																

Figura 6. Construção da matriz G.

Isso resulta numa matriz não quadrada, cujo número de linhas é igual ao número de graus de liberdade e o número de colunas é igual ao produto entre o número de elementos, a quantidade de nós que tem cada elemento e a quantidade de graus de liberdade que cada nó tem.

Para montar a matriz H o procedimento é um pouco mais complexo, uma vez que essa matriz multiplica um vetor cujos componentes são deslocamentos e para os quais existem condições de continuidade que devem ser observadas.

O deslocamento relacionado a um nó para uma dada direção é único, pois o mesmo ponto não pode estar se movendo numa mesma direção com dois valores diferentes. Isso significa que, quando um nó está na fronteira entre dois elementos e pertence aos dois, seu deslocamento precisa ser compatível. O reflexo dessa condição sobre a construção da matriz H é grande, pois nela, para cada nó, independente do elemento em que se localize, existe apenas um valor. Assim, os valores obtidos para cada integração precisam ser somados se pertencerem ao mesmo nó, mesmo que tenham sido calculados em passos referentes a elementos diferentes.

Essa forma de construir a matriz H faz com que ela seja uma matriz quadrada de ordem igual ao número de graus de liberdade do problema. De maneira esquemática, a construção pode ser visualizada na figura 7.

		nó 1			nó 2			nó 3			...		
		X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
Pto. Fonte 1	X												
	Y												
	Z												

Figura 7. Construção da matriz H.

Uma vez construídas as matrizes H e G todo o processo de integração das equações integrais de contorno está concluído. Agora, é necessário proceder à substituição das condições de contorno, o que vai tornar o sistema de equações representado pelas matrizes H e G solúvel.

Aplicar as condições de contorno ao problema significa determinar quais são os graus de liberdade conhecido, substituí-los por seus respectivos valores e proceder aos cálculos matriciais a fim de gerar um vetor com os valores conhecidos. Nesse processo também são agrupados os graus de liberdade desconhecidos, cuja determinação será a solução do problema. Uma vez agrupados, os graus de liberdade desconhecidos permitem separar as colunas das matrizes H e G pelas quais são multiplicados, fazendo surgir uma matriz quadrada chamada de matriz dos coeficientes, pois são os coeficientes que multiplicam cada um dos graus de liberdade desconhecidos em todas as equações do sistema.

O agrupamento dos graus de liberdade desconhecidos e conhecidos leva a um procedimento conhecido como permutação de colunas entre as matrizes H e G. Segundo esse processo, as colunas correspondentes aos graus de liberdade conhecidos, sejam elas de H ou G, são separadas das colunas correspondentes aos graus de liberdade desconhecidos, dando origem a duas matrizes.

A matriz correspondente aos coeficientes dos graus de liberdade conhecidos pode ser imediatamente multiplicada pelo vetor dos valores dos respectivos graus de liberdade, gerando um vetor de termos numéricos. No caso da matriz referente aos coeficientes dos graus de liberdade desconhecidos surge um problema.

Quando o grau de liberdade desconhecido é um deslocamento, sua permutação é única, o que significa que existe apenas uma coluna para ser trocada para aquele determinado nó naquele determinado grau de liberdade. Já se o grau de liberdade desconhecido é uma força de superfície, a permutação pode não ser única. Isso ocorre pelo mesmo motivo que justifica a construção da matriz G. Para um mesmo nó, numa mesma direção, podem existir duas ou mais forças de superfície diferentes atuando, uma em relação a cada elemento que compartilha o nó, assim, pode existir mais que uma coluna em G para um mesmo grau de liberdade de força de superfície desconhecido.

Uma solução para esse problema é realizar a soma simples das colunas quando existir mais que uma. Isso é uma saída possível quando os elementos encontram-se alinhados, o que significa que o desalinhamento entre o vetor normal de um em relação ao outro não é grande. Nesses casos, a contribuição de força de superfície de cada elemento é somada como uma soma algébrica, mas representa uma soma vetorial. Essa foi a saída adotada nesse trabalho.

Dessa forma, ao aplicar as condições de contorno e agrupar os graus de liberdade conhecidos separadamente dos desconhecidos o resultado é a construção de um sistema de equações na forma matricial

$$[A]\cdot\{x\}=\{b\} \tag{23}$$

sendo $[A]$ a matriz dos coeficientes, $\{x\}$ o vetor das incógnitas e $\{b\}$ o vetor das constantes determinadas através da multiplicação das condições de contorno por seus respectivos componentes das matrizes de integração.

Resolver esse sistema passa pela aplicação de um dos algoritmos de cálculo numérico para a solução de sistemas matriciais. Pode ser aplicado, por exemplo, o método da eliminação de Gauss, ou mesmo a inversão direta da matriz $[A]$ para que seja multiplicada pelo vetor $\{b\}$.

Uma vez resolvido o sistema dado pela equação 23, estão determinadas as chamadas respostas primárias, que representam os graus de liberdade, que eram desconhecidos antes da solução do sistema, unidos às condições de contorno prescritas, o que significa que são conhecidos todos os valores de força de superfície e deslocamento para todos os graus de liberdade do problema. No entanto, na maioria das vezes não se está interessado apenas em deslocamentos ou forças de superfície sobre os nós, então, são necessários resultados na forma de tensões nodais ou deformações, o que se chama de respostas secundárias.

Como pode se notar as respostas secundárias não fazem parte da solução direta do problema, mas são produto de manipulação das soluções principais. Assim, para que seja

possível obter qualquer uma das respostas secundárias é necessário obter uma resposta básica, o tensor de tensões em cada um dos nós do problema. A partir desse tensor é possível obter outras tensões desejadas, deformações etc.

O procedimento necessário para realizar o cálculo dos tensores de tensão em cada nó do problema a partir das respostas primárias é mostrado por *Kane* (1994) e é chamado de *surface stress component recovery*.

2.5 **Cálculo dos tensores de tensão nodal**

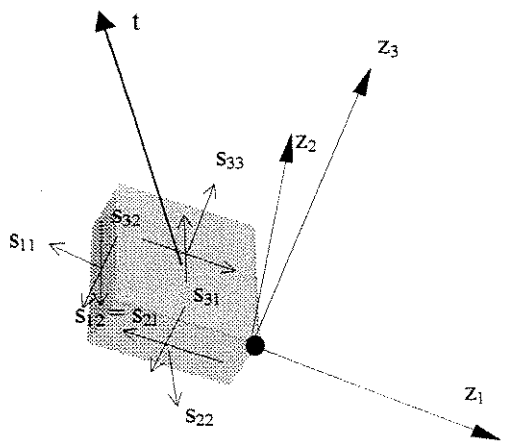
Os resultados primários de força de superfície e deslocamento nodais podem ser utilizados como informações iniciais para o obtenção de um tensor de tensões nodal completo. Para que isso seja possível primeiramente é necessário que se defina um sistema de coordenadas local para cada nó, que tenha um de seus eixos normal e outro tangencial à superfície do elemento no nó.

Analisando o problema para um nó é necessário que se estabeleça um sistema de coordenadas local z_i com a origem sobre o nó, direção z_1 tangente à superfície do elemento de contorno e crescendo no mesmo sentido de crescimento da coordenada intrínseca a_1, direção z_3 normal à superfície do elemento, apontando para fora do domínio e z_2 perpendicular a z_1 e z_3.

Nesse sistema de coordenadas z_i o tensor de tensões pode ser relacionado diretamente com as forças de superfícies nodais, desde que as mesmas sejam transformadas do sistema de coordenadas global para o local. Segundo essa transformação, as forças de superfície nodais passam a atuar na superfície do tensor definido para o sistema de coordenadas local. Isso significa que os componentes de

cisalhamento e normal na superfície em que estão localizadas as forças de superfície nodais transformadas já estão determinados.

$$s_{ij} = \lambda \delta_{ij} e_{kk} + 2\mu e_{ij} \tag{24}$$



$$\begin{aligned} s_{3j} &= p_j \\ s_{11} &= (2\mu + \lambda) e_{11} + \lambda (e_{22} + e_{33}) \\ s_{22} &= (2\mu + \lambda) e_{22} + \lambda (e_{33} + e_{11}) \\ s_{12} &= 2\mu \cdot e_{12} \end{aligned} \tag{25}$$

Sendo s_{ij} o tensor de tensões, p_j as trações e e_{ij} as deformações no sistema de coordenadas local.

A partir da equação 24 particularizada para o caso de s_{33} é possível substituir e_{33}, que não está disponível, das equações mostradas em (25), levando a expressões que possibilitam calcular os valores de s_{11} e s_{22} em função de e_{11}, e_{22} e p_3, que corresponde à força de superfície nodal transformada para o sistema local na coordenada z_3. Dessa forma, as expressões ficam dadas por:

$$
\begin{aligned}
s_{11} &= \frac{1}{2\mu+\lambda}\{4\mu(\mu+\lambda)e_{11} + 2\mu\lambda e_{22} + \lambda p_3\} \\
s_{11} &= \frac{1}{2\mu+\lambda}\{4\mu(\mu+\lambda)e_{22} + 2\mu\lambda e_{11} + \lambda p_3\} \\
s_{12} &= 2\mu \cdot e_{12}
\end{aligned}
\tag{26}
$$

Usando das relações entre deformação e deslocamento, mostradas na equação 2, é possível substituir todas as deformações presentes nas equações em (26) por expressões em função de derivadas do deslocamento. Considerando que esses deslocamentos são valores nodais, é necessário tratá-los como tal, o que significa utilizar das funções de forma. Isso significa que as derivadas do deslocamento que serão utilizadas para o cálculo das deformação devem ser calculadas como somatória das derivadas das funções de forma multiplicadas pelos valores nodais de deslocamento.

$$
\begin{aligned}
s_{11} &= \frac{1}{2\mu+\lambda}\left\{4\mu(\mu+\lambda)\frac{\partial v_1}{\partial z_1} + 2\mu\lambda\frac{\partial v_2}{\partial z_2} + \lambda p_3\right\} \\
s_{11} &= \frac{1}{2\mu+\lambda}\left\{4\mu(\mu+\lambda)\frac{\partial v_2}{\partial z_2} + 2\mu\lambda\frac{\partial v_1}{\partial z_1} + \lambda p_3\right\} \\
s_{12} &= 2\mu\left\{\frac{\partial v_1}{\partial z_2} + \frac{\partial v_2}{\partial z_1}\right\}
\end{aligned}
\tag{27}
$$

Nas equações 27, v_i são os deslocamentos nodais transformados para o sistema de coordenadas local.

As derivadas do deslocamento em relação às coordenadas do sistema local não podem ser avaliadas diretamente, uma vez que a única maneira de calcular esse tipo de derivada é utilizar as funções de forma que são função das coordenadas intrínsecas. Assim, o que se faz é utilizar a regra da cadeia para transformar essa derivada em duas, uma do deslocamento em relação à coordenada intrínseca e outra da coordenada intrínseca em relação à coordenada local.

A derivada do deslocamento em relação à coordenada intrínseca pode ser feita utilizando as derivadas das funções de forma multiplicadas pelo valor nodal somadas para os oito nós do elemento.

Para a derivada da coordenada intrínseca em relação à coordenada local, é necessário lembrar do Jacobiano, cuja definição corresponde exatamente ao inverso dessa derivada. Portanto, essa derivada pode ser substituída pela inversa da matriz do Jacobiano que transforma a geometria do sistema de coordenadas local para o intrínseco.

Dessa forma, uma vez executados os cálculos, terão sido determinados os seis componentes do tensor de tensões no sistema local, o que já é o suficiente, graças à simetria desse tensor. Para completar esse procedimento, basta transformar de volta o tensor de tensões do sistema local para o global.

Isso conclui o procedimento de cálculo para um nó do problema, o mesmo deve ser repetido para todos os demais, armazenando um tensor de tensões de ordem três para cada um dos nós do problema.

2.6 **A integração singular**

Um dos aspectos particulares da integração quando o integrando envolve uma solução fundamental é o aspecto singular que pode existir. No caso das soluções fundamentais necessárias para a determinação da equação integral de contorno, sempre existe uma condição para a qual a solução fundamental apresenta um comportamento singular, visto que representa a solução para um problema num meio infinito sob ação de um carregamento concentrado.

No caso das soluções fundamentais para o problema da elasticidade existe uma carga concentrada que é aplicada num determinado ponto de um domínio infinito.

Quando anexada ao integrando da equação integral de contorno, a solução fundamental, seja de força de superfície ou deslocamento, passa a representar um problema para a integração, uma vez que consiste numa função cujo comportamento é singular em um ponto. No ponto onde a carga concentrada é aplicada, o comportamento do integrando é determinado pela solução fundamental, que tende ao infinito, o que torna complicada sua avaliação numérica.

Esse tipo de problema ocorre apenas para os casos em que a integral precisa ser avaliada num ponto que é coincidente com o ponto de aplicação da carga concentrada. Isso significa que se está integrando sobre um elemento para o qual o nó fonte será um de seus nós.

Nessas situações é preciso aplicar métodos de cálculo especiais a fim de que seja garantida a solução da integral, uma vez que a integração analítica de uma função de comportamento singular pode resultar em uma singularidade que não tem validade como resposta.

Existem métodos para contornar o problema da singularidade presente no integrando devida as soluções fundamentais, dentre eles alguns são tratamentos matemáticos que podem ser aplicados à solução fundamental a fim de que sua singularidade seja eliminada. Técnicas matemáticas como avaliar a integral no sentido do valor principal de Cauchy ou promover transformações de coordenadas para sistemas que possibilitem a integração são correntes.

Para esse trabalho duas técnicas serão utilizadas, uma para avaliar a integral cuja singularidade é devida à solução fundamental de forças de superfície e outra para avaliar a integral cuja singularidade se deve à solução fundamental de deslocamentos. A singularidade devida à solução fundamental de deslocamento pode ser considerada fraca, enquanto a singularidade da solução fundamental de força de superfície é classificada como forte. Isso significa que é necessário tratá-las de maneiras distintas.

A técnica utilizada para tratar a singularidade fraca é aquela baseada num artifício geométrico válido para problemas tridimensionais, utilizando elementos de contorno do tipo quadrilateral de oito nós, que é descrita por *Domingues* (1993), conhecida por triangularização. Segundo essa técnica, a integração da equação integral de contorno é avaliada sempre da mesma forma, utilizando o mesmo algoritmo de integração e as mesmas equações, mas recebe diferentes Jacobianos de acordo com sua localização. Já a técnica utilizada para tratar a singularidade forte envolve a chamada hipótese de movimento de corpo rígido.

Ao realizar a integração da equação de contorno sobre um elemento, o método de cálculo analisa se o ponto fonte pertence ao elemento em questão. Caso a resposta seja negativa, significa que não haverá singularidade e que é possível utilizar o método de integração convencional. No entanto, se a resposta for afirmativa significa que ocorrerá um problema de singularidade, então, se a singularidade for fraca, o elemento é triangularizado em relação ao nó sobre o qual se localiza o ponto fonte para a determinação da integral. Caso ocorra uma singularidade forte, as integrais sobre o elemento simplesmente não são calculadas.

Isso significa que a matriz G, determinada a partir das integrações das equações integrais de contorno cuja solução fundamental apresenta uma singularidade fraca, será completamente determinada. Já a matriz H, determinada a partir das integrações de equações integrais de contorno cuja solução fundamental apresenta singularidade forte, terá todos os seus componentes calculados a menos daqueles presentes numa banda ao longo da diagonal principal. Isso se deve ao fato de que os componentes presentes na banda da diagonal principal dependem de uma integração envolvendo a singularidade forte. Para eliminar a necessidade dessa integração, que é matematicamente mais complicada, esses componentes são calculados posteriormente por uma consideração de movimento de corpo rígido.

A triangularização do elemento consiste em transformar um elemento quadrilateral de oito nós em dois ou três elementos triangulares de três nós de maneira que um dos vértices de um dos triângulos corresponda ao nó de localização do ponto fonte. Isso é feito com a intenção de que o Jacobiano nesse nó seja nulo, anulando a integral que vai para o infinito de maneira matematicamente consistente.

Segundo o método, ao se realizar a triangularização, os elementos gerados tornam-se geometricamente triangulares, no entanto, matematicamente são tratados como elementos quadrilaterais de quatro nós, sendo necessário trocar as funções de forma para funções de forma lineares e adaptar o cálculo do Jacobiano. O artifício geométrico que resolve a integral singular está exatamente nesse aparente paradoxo de se ter um elemento geometricamente triangular, mas matematicamente quadrilateral, o que é possível somente caso dois dos vértices do quadrilátero se fundam em um. Quando isso ocorre e os dois vértices se fundem naquele que é o ponto fonte da integral, seu Jacobiano vai para zero na proporção do raio, anulando o efeito de singularidade do integrando que vai para o infinito na proporção do inverso do raio.

Caso o ponto fonte esteja localizado num nó que é um dos vértices do elemento quadrilateral de oito nós, a triangularização ocorre dividindo o elemento em dois triângulos de acordo com a diagonal do elemento quadrilateral. Caso o ponto fonte se localize no meio de uma aresta, a triangularização requer o uso de três triângulos, um deles tendo um dos vértices como o nó do ponto fonte. Dessa forma, a triangularização pode se utilizar de elementos semelhantes aos mostrados na figura 8.

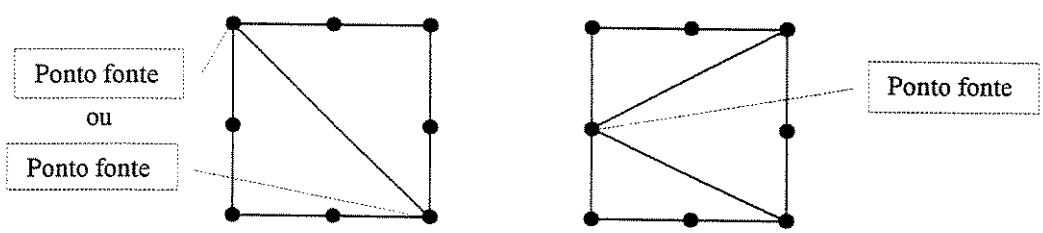


Figura 8. Elementos triangularizados para integração singular.

Esse procedimento é utilizado para possibilitar o cálculo das integrais que compõe a matriz G. No caso da matriz H, que envolve integrais com singularidade forte, os componentes da matriz que dependem de integração na qual ocorre o problema da singularidade não são avaliados, permanecendo indeterminados até que se termine o cálculo da matriz H completa. Uma vez disponível a matriz H completa, a menos dos termos involvendo a singularidade, é feita a chamada hipótese de movimento de corpo rígido.

Basicamente, a hipótese de movimento de corpo rígido consiste em supor uma condição de contorno específica, segundo a qual todos os nós do contorno apresentam um deslocamento unitário em uma mesma direção global. Com isso todas as forças de superfícies tornam-se nulas. Isso gera um vetor de deslocamentos unitário e um vetor de forças de superfície nulo para serem utilizados na equação matricial $[H].\{u\} = [G].\{t\}$.

É possível notar que, com um vetor de forças de superfície, $\{t\}$, nulo a equação se degenera para $[H].\{u\} = \{0\}$. Como o vetor de deslocamentos, {u}, é unitário, resta concluir que $[H] = \{0\}$.

Cada linha da matriz H pode ser vista como uma equação linear independente das demais, que se relaciona com um resultado equivalente no vetor do outro lado que, no caso da hipótese de movimento de corpo rígido, se reduz a um vetor nulo. Assim, é possível dizer que cada linha da matriz H deve ser identicamente nula. Essa conclusão é particularmente útil para essa matriz, pois permite calcular os termos incógnitos associados com as integrais singulares para soluções fundamentais de força de superfície.

Para cada linha da matriz H, no caso de problemas tridimensionais, existem três termos desconhecidos, relacionados com a singularidade que ocorre em cada um dos três graus de liberdade de cada nó do problema. Para obter cada um dos valores é necessário aplicar a hipótese de movimento de corpo rígido para dada um dos graus de liberdade, agrupando as colunas de maneira a que cada equação formada relacione apenas um mesmo grau de liberdade.

Lembrando da disposição dos termos na matriz H, como mostrado na figura 7, os três primeiros termos deconhecidos na primeira linha da matriz ficam nas três primeiras colunas, sendo o da primeira coluna relacionado ao grau de liberdade x, o da segunda ao grau de liberdade y e o da terceira ao z. Dessa forma, para determinar o termo desconhecido relacionado ao grau de liberdade x é necessário aplicar a consideração de movimento de corpo rígido apenas com os valores da matriz relacionados com o grau de liberdade x, o mesmo acontecendo para y e z.

Para agrupar os termos referentes a cada grau de liberdade é preciso lembrar que cada bloco de três colunas corresponde a um nó, portanto, a cada três colunas ocorre a repetição dos graus de liberdade. Assim, a consideração de movimento de corpo rígido deve ser aplicada para todos os nós em um grau de liberdade específico.

Uma vez isolados os termos a serem utilizado para cada grau de liberdade o procedimento fica idêntico àquele aplicado para a solução de uma equação a uma incógnita. Basta somar todos os termos conhecidos e inverter o sinal do resultado para obter o termo procurado.

Repetindo todo o procedimento para cada uma das linhas da matriz H será obtida toda a banda da diagonal principal correspondente aos valores da integração quando o integrando apresenta comportamento singular sem a necessidade de desenvolver o processo de integração.

2.7 Métodos de otimização utilizados

Existem diversos métodos de otimização que podem ser implementados em códigos de análise numérica pelo método dos elementos de contorno que agilizam a obtenção das matrizes necessárias ao método, as matrizes H e G, o que costuma ser a parte mais demorada e custosa da análise.

Para agilizar a obtenção das matrizes H e G foi utilizado, nesse trabalho, um algoritmo descrito por *Kane* (1994) sob a sigla *RISP*, que significa *Reusable Intrinsec Sample Points*, o qual se baseia numa seqüência capaz de garantir que valores calculados para a integração em um elemento possam ser imediatamente utilizados nas integrações posteriores dos outros elementos.

Além desse algoritmo de otimização, foi implementada a reutilização das matrizes principais do método dos elementos de contorno, as matrizes H e G, para os casos em que é necessário realizar uma segunda análise para uma mesma geometria, mas utilizando condições de contorno diferentes. Quando isso acontece, devido a particularidades do método de construção das matrizes H e G, é possível utilizar as mesmas matrizes H e G já calculadas, durante análise anterior, para realizar novas análises.

O processo de formação das matrizes H e G depende, fundamentalmente, das integrações das soluções fundamentais e da colocação dos resultados dessas operações em posições adequadas das matrizes globais. Tanto a matriz H quanto a G são montadas tendo como base a disposição geométrica dos nós do problema, ou seja, o posicionamento e a conectividade de cada um deles. Uma vez montadas, as matrizes H e G passam a representar o comportamento genérico da geometria do problema, que será particularizado de acordo com as condições de contorno aplicadas.

Devido à complexidade das integrações e do procedimento necessário para realizá-las, o tempo necessário para que se conclua a construção das matrizes H e G é,

notadamente, o maior de todo o processo de análise de elementos de contorno, o que significa que, se for possível evitar o cálculo nessa etapa, a economia de tempo será expressiva.

Assim, o algoritmo trabalha salvando as matrizes H e G quando da primeira análise, reutilizando essas mesmas matrizes para análises posteriores com diferentes condições de contorno.

CAPÍTULO 3

DESENVOLVIMENTO COMPUTACIONAL

3.1 O programa E-Con3D

Para efetivar a implementação computacional do método de análise de problemas de elasticidade a partir de equações integrais de contorno, foi construído o código de programa chamado de E-Con3D para elementos de contorno em aplicações tridimensionais. Esse código foi desenvolvido em linguagem MatLab® 5.2 a partir de códigos de programas de elementos de contorno tradicionais, envolvendo implementação de algoritmos de otimização, tais como o *RISP* e a técnica de reutilização das matrizes H e G, assim como o uso de programas comerciais para a construção de bases de dados para os modelos mais complexos.

A escolha da linguagem MatLab® para programação do código deveu-se, principalmente, a conhecimento anterior, consistindo na ferramenta mais indicada para o desenvolvimento de uma técnica e para o teste de novas implementações computacionais. Certamente do ponto de vista de velocidade de execução, o código em MatLab® não representa a melhor solução, uma vez que se trata de uma linguagem interpretada, necessitando de ambiente de execução específico do MatLab®. Por outro lado, uma vez

que está dentro do ambiente de um aplicativo de matemática, o código interpretado pode fazer uso de diversas funções pré-programadas em linguagem compilada, facilitando a programação, pois torna mais intuitiva a tradução de equações da matemática algébrica para formulações numéricas.

Além disso, outro motivo que incentivou o uso do ambiente MatLab® foi a disponibilidade de funções gráficas já implementadas e de fácil uso. Isso permitiu a construção de ferramentas de visualização gráfica que, embora bastante rudimentares, serviram para a representação das soluções em pós-processamento, ou para a verificação da geometria e da malha em pré-processamento.

A descrição do modo de aplicação de todas as funções do MatLab® utilizadas durante a implementação do código E-Con3D, assim como sua fundamentação teórica e limitações estão presentes nos manuais do aplicativo, fornecidos juntamente com o pacote de instalação, ou na forma de material eletrônico em formato de arquivo *.PDF* disponível nos módulos de ajuda, conforme citado em referência.

De maneira genérica, o programa desenvolvido pode ser caracterizado pelo fluxograma da figura 9.

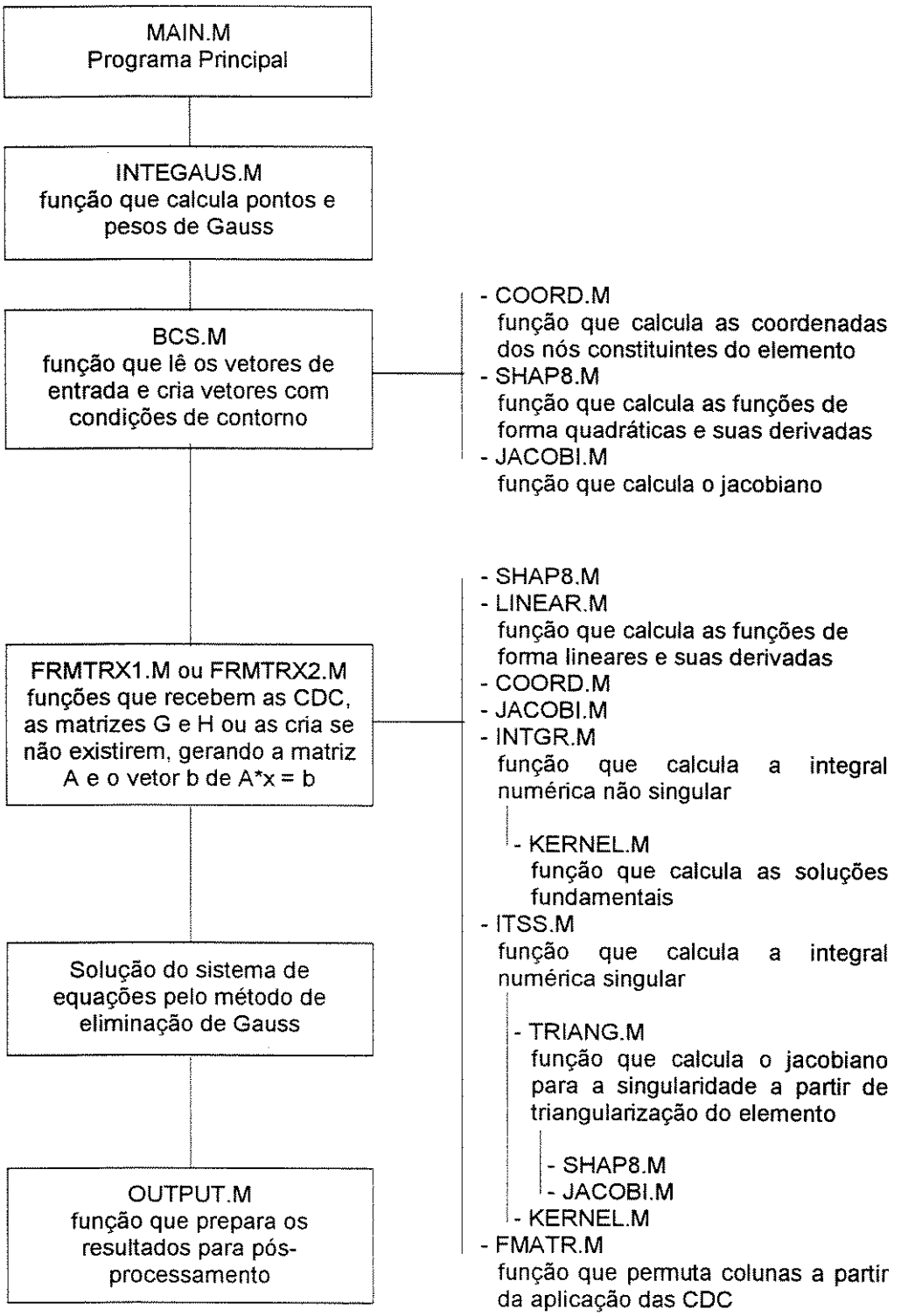


Figura 9. Fluxograma de funcionamento do programa E-con3D.

A partir desse fluxograma é possível identificar as funções desenvolvidas para o programa e sua interação com o fluxo principal de dados. De acordo com o esquema, as funções dentro de caixas representam o fluxo principal de dados, enquanto aquelas sem caixas representam funções auxiliares, responsáveis por partes importantes do processamento da análise, mas que sempre retornam dados para as funções do fluxo principal.

Todas as funções do fluxo principal de dados são chamadas em seqüência pelo programa *MAIN.m*, que também é responsável pela leitura dos dados armazenados em arquivos padrão do tipo texto e pela conversão e separação desses dados em estruturas adequadas ao funcionamento das outras funções do programa. Isso significa que o programa principal receberá dados no formato padrão texto e os converterá em vetores e matrizes de acordo com o necessário para o prosseguimento do programa.

Os arquivos padrão, em formato texto, podem ser escritos manualmente ou extraídos da base de dados do programa comercial Ansys®, através do uso de uma função programada em linguagem própria desse pacote. Basicamente, a estrutura de dados é composta por cinco arquivos texto: *CoordNos.txt*, *NOEL.txt*, *CDCDesl.txt*, *CDCTens.txt*, *Props.txt*. Respectivamente, esses arquivos contêm as coordenadas de todos os nós no sistema global; a matriz de conectividade entre elementos e nós; as condições de contorno de deslocamento e os elementos aos quais se aplicam; as condições de contorno de tensão e os elementos aos quais se aplicam; as propriedades do material juntamente com a máxima dimensão do problema e o número de pontos de Gauss para a integração. A estrutura básica de cada um dos arquivos segue em anexo 1.

Note que as inserções de texto explicativas utilizadas nos modelos mostrados no anexo 1 não devem aparecer nos arquivos de uso. Nesses, apenas os valores numéricos devem estar dispostos na formatação indicada, a fim de que o interpretador implementado no programa *MAIN.m* seja capaz de identificar os valores sem problemas.

Os códigos para conversão de modelos feitos no Ansys® em bases de dados no formato texto, de acordo com os modelos mostrados, constam do anexo 2 desse trabalho. Apenas a título de esclarecimento, todo o código dos dois programas utilizados para a geração das bases de dados em arquivo texto foi gerado utilizando comandos do ambiente interpretativo do Ansys®, os quais constam dos manuais do usuário.

O processo de leitura de dados, assim como a seqüência de chamada das funções do fluxo principal de dados do programa *MAIN.m* podem ser melhor visualizados em sua listagem, que segue em anexo 3.

Uma vez lidos, os dados de entrada são transformados nas variáveis escalares:

- *MAXL*: define a máxima dimensão da geometria do problema;
- *E*: módulo de Young;
- *NU*: coeficiente de Poisson;
- *IGAUS*: quantidade de pontos de Gauss para a integração;
- *NNP*: quantidade de nós do problema;
- *NEL*: quantidade de elementos de contorno da malha;
- *NBCU*: quantidade de condições de contorno de deslocamento prescrito;
- *NBCP*: quantidade de condições de contorno de tensão.

Além das variáveis escalares, são definidas as variáveis matriciais:

- *CoordNos*: matriz das coordenadas globais de todos os nós do problema por linhas;

- *X, Y, Z*: vetores das coordenadas globais correspondentes (x, y ou z) de todos os nós do problema por linhas;

- *NOEL*: matriz de conectividade dos nós, colocados nas colunas, com os elementos, colocados nas linhas;

- *MNEL*: vetor contendo os números dos elementos sujeitos à condição de contorno de deslocamento prescrito;

- *MCOND*: vetor contendo as direções nas quais cada elemento está sujeito à condição de contorno de deslocamento prescrito, por linhas, sendo 1, 2, 3 correspondentes às direções x, y, z;

- *UPRES*: matriz contendo os valores de deslocamento prescritos para cada nó de cada elemento, sendo os elementos colocados nas linhas e os nós nas colunas;

- *CondContTens*: matriz contendo as condições de contorno de tensão prescritas para cada elemento, cuja estrutura será descrita posteriormente ao se tratar da função *BCS.m*.

Seguindo a ordem de surgimento das funções, dada pelo programa principal, a primeira função após o bloco de leitura de dados é aquela responsável pelo cálculo das coordenadas e dos pesos para os pontos de Gauss. A função *INTEGAUS.m* calcula os pontos e pesos de Gauss na quantidade de acordo com o solicitado para a análise. Os valores calculados ficam armazenados nos vetores *XG* e *CG*, sendo o primeiro para as coordenadas dos pontos e o segundo para os pesos. O algoritmo de cálculo para uma quantidade qualquer de pontos de Gauss já havia sido implementado no programa anteriormente, tendo sua validade sido testada através de comparações com valores tabelados por *Kane* (1994), portanto, não serão feitos maiores comentários a respeito. A listagem dessa função encontra-se em anexo 3.

A próxima função a ser chamada é a *BCS.m*, responsável por efetuar um segundo tratamento nos dados de entrada, que agora estão na forma de vetores, correspondentes às condições de contorno de tensão e deslocamento. Através dessa função, matrizes contendo as informações sobre condições de contorno de tensão sobre elementos são transformadas para matrizes contendo informações equivalentes de condições de contorno de força de superfície nodal para cada elemento. Além disso, essa função avalia a validade das condições de contorno de deslocamento, interrompendo o programa em caso de erro. Sua listagem também pode ser vista no anexo 3.

Uma vez prontos os vetores e as matrizes contendo as condições de contorno do problema, os dados são passados para a próxima fase, que consiste na formação da matriz e do vetor que constituirão o sistema de equações a ser resolvido para obter as respostas primárias do problema. Isso é feito usando a função chamada *FRMTRX1.m* ou a *FRMTRX2.m*, ambas listadas no anexo 3.

Ambas as funções são capazes de manipular as matrizes H e G a partir das condições de contorno para arranjá-las na matriz A e vetor BT do sistema matricial a ser resolvido. No entanto, apenas a *FRMTRX1.m* é capaz de criar as matrizes H e G. A função *FRMTRX2.m* é aplicada a casos em que já existem as matrizes H e G formadas durante uma análise anterior e uma nova análise, alterando apenas as condições de contorno, é necessária. Assim, a função *FRMTRX2.m* lê as matrizes H e G geradas durante análise anterior pela função *FRMTRX1.m* e realiza o processo de manipulação dessas matrizes a partir daí, o que permite uma expressiva economia de tempo de análise.

Uma vez terminada a execução de qualquer das duas funções, estão geradas a matriz A e o vetor BT característicos de um sistema de equações matricial. Esse sistema pode ser resolvido através de métodos numéricos, tais como a eliminação de Gauss. Esse método poderia ter sido implementado na forma de uma função, no entanto, optou-se por fazer uso do operador específico do MatLab® para esse tipo aplicação, o *back slash*, que se mostrou mais rápido. Segundo os manuais do aplicativo, esse operador utiliza da

eliminação de Gauss quando a operação é realizada entre uma matriz quadrada cheia e um vetor de mesma ordem que a matriz, mas de uma maneira otimizada e compilada.

Resolvido o sistema, um vetor B é formado contendo todas as respostas primárias para o problema, o que significa respostas para deslocamento e força de superfície nos nós misturadas em um mesmo vetor. Para que se possa pós-processar esses dados de uma maneira mais organizada é preciso separar os deslocamentos das forças de superfície e acrescentar as condições de contorno para completar todas as informações sobre os graus de liberdade do sistema. Isso é feito através da função *OUTPUT.m*, listada em anexo 3.

Essa função trata apenas de organizar os dados presentes no único vetor de resposta em dois vetores, um de deslocamentos nodais e outro de forças de superfície nodais. Além disso, acrescenta aos vetores de resposta as condições de contorno, completando o conjunto de informações em todos os graus de liberdade para todos os nós do problema.

Com isso ficam determinadas as repostas primárias para o problema. Para obter respostas secundárias em tensão é utilizada a função *TENSNODAL.m*, que recebe as matrizes contendo as respostas primárias e, através de uma abordagem canônica, realiza a recuperação das tensões a partir das forças de superfície e dos deslocamentos nodais, como descrito durante o capítulo 2 sobre os aspectos teóricos.

Finalmente, após obter todos os resultados previstos, o programa principal gera um arquivo de resultados contendo as matrizes *Tracao* e *Desloc* das respostas primárias, a matriz *SigNodal* das respostas secundárias em tensão e uma variável *TempoProcesso* que dá o tempo total gasto para realizar a análise. Esse arquivo de resultados recebe um nome escolhido pelo usuário com extensão *.mat*, o que permite que seja lido diretamente pelo ambiente do MatLab®.

3.2 **Funções principais**

As funções chamadas de principais são aquelas que fazem parte do fluxo principal de dados do programa E-Con3D, que são:

- *INTEGAUS.m*;
- *BCS.m*;
- *FRMTRX1.m* ou *FRMTRX2.m*;
- *OUTPUT.m*;

Como é possível notar, não existe uma função para resolver sistemas de equações matriciais, pois optou-se por utilizar o operador disponível no aplicativo MatLab®, o que reduz a função a uma operação que ocorre entre as *FRMTRX1.m* ou *FRMTRX1.m* e a *OUTPUT.m*. As demais funções são encadeadas de acordo com a seqüência dos itens.

3.2.1 **A função *INTEGAUS.m***

A primeira função é responsável por calcular os pontos e pesos de Gauss para a integração numérica que será feita no decorrer do programa. Para isso, essa função precisa receber do usuário a quantidade de pontos de Gauss que ele deseja utilizar no processo de integração.

Vale lembrar que a precisão da integração numérica pelo método de Gauss depende da quantidade de pontos que se utiliza. Na verdade, quanto maior a quantidade de pontos, maior o grau do polinômio que o método consegue integrar de maneira exata. Como a

equação que precisa ser integrada, para que se resolva a equação integral de contorno, é mais complexa que um polinômio, é necessário que se utilize uma quantidade maior de pontos de Gauss. Tipicamente, segundo indicado por *Kane* (1994), para o caso tridimensional são usados, no máximo, oito pontos de Gauss para cada linha do elemento.

No caso, o usuário pode escolher livremente, lembrando-se que, além de maior precisão, uma quantidade maior de pontos de Gauss também implica num tempo maior durante o processo de integração. Assim, a quantidade pode ser diminuída para seis pontos ao invés de oito, com uma redução no tempo de processamento para quase metade em relação àquele que utiliza oito pontos de Gauss.

O único dado necessário e suficiente para que a função *INTEGAUS.m* realize seu trabalho é a quantidade de pontos de Gauss requerida. Uma vez inserida essa informação a função trabalha com um algoritmo automático de geração de pontos de Gauss e retorna dois vetores ao fluxo principal do programa: *XG* contendo as coordenadas dos pontos de Gauss e *CG* contendo os pesos correspondentes a cada ponto.

3.2.2 **A função *BCS.m***

A próxima função na seqüência de chamada do programa principal é a *BCS.m*, responsável por verificar e reagrupar as condições de contorno lidas a partir dos arquivos de entrada.

Apesar de certa similaridade com o procedimento de leitura desenvolvido pelo programa principal, a função *BCS.m* realiza um trabalho distinto. O procedimento de leitura do programa principal tem por finalidade a tradução dos dados do formato texto para variáveis matriciais. Já a função *BCS.m* trabalha com essas variáveis distribuindo

seus conteúdos em outras variáveis de maneira a facilitar o uso para as demais funções do programa.

Os dados de entrada necessários para que *BCS.m* funcione são:

- quantidade de condições de contorno de deslocamento;
- quantidade de condições de contorno de tração nula prescrita;
- quantidade de pontos de Gauss;
- matriz de conectividade;
- coordenadas x, y e z de todos os nós do problema;
- matriz contendo as condições de contorno de tensão sobre os elementos.

A estrutura da matriz das condições de contorno de tensão sobre os elementos é idêntica àquela que é mostrada nos arquivos de entrada, sendo que cada linha corresponde a uma condição de tensão definida por um tensor. A primeira coluna corresponde a quantidade de elementos sujeitos à condição de contorno. A segunda coluna contém um valor numérico caso o tensor corresponda a uma pressão constante e normal ao elemento. As seis colunas a partir da terceira correspondem aos valores do tensor de tensão. As colunas a partir da sétima contêm os números dos elementos que estão sujeitos à condição de contorno.

A primeira ação determinada para essa função é verificar a validade das condições de contorno de deslocamento prescritas, o que significa certificar-se de que existem condições de deslocamento prescritas. Para isso o código se limita a verificar se a variável que armazena a quantidade de condições de contorno de deslocamento prescrito tem tamanho maior que zero. Isso basta para esse estágio do programa, pois a inexistência de condições de contorno de deslocamento levaria a problemas durante a análise, tais como movimentos de corpo rígido, impossibilitando uma análise estática.

Após essa verificação, a função passa a tratar as condições de contorno de tensão prescritas para cada elemento, a fim de transformá-las em condições de contorno de força de superfície nodal. Isso precisa ser feito pois as equações integrais de contorno, como visto no capítulo 2 de aspectos teóricos, ficaram descritas em termos de deslocamentos e forças de superfície. Como já se dispõe dos deslocamentos diretamente a partir das condições de contorno, basta trabalhar para transformar as tensões prescritas em forças de superfície.

Para transformar o estado de tensão prescrito sobre o elemento em forças de superfície colocadas sobre seus nós a função se utiliza da relação de Cauchy entre força de superfície e tensão, mostrada na equação 26.

$$t_i = \sigma_{ij} n_j \tag{26}$$

Da equação de Cauchy a função dispõe do tensor de tensões, σ_{ij}, dado pelos valores na matriz das condições de contorno de tensão e da normal, n_j, que pode ser obtida juntamente com o Jacobiano através da função *JACOBI.m*, que será descrita posteriormente. Dessa forma, a força de superfície, t_i, fica completamente determinada para cada nó do elemento.

Após terminar a transformação das tensões para forças de superfície em todos os elementos prescritos, a função gera um conjunto de variáveis que serão utilizadas pelas demais funções do programa que são:

- *NELM*: vetor contendo os números dos elementos de contorno sujeitos à condição de contorno de força de superfície nodal;
- *JT*: vetor contendo os números globais dos nós que compõe os elementos sujeitos às condições de contorno de força de superfície;
- *jt_local*: vetor contendo os números locais dos nós que compõe os elementos sujeitos às condições de contorno de força de superfície;

- *ICOND*: vetor contendo as direções nas quais atuam as condições de contorno de força de superfície em cada nó dos elementos determinados;

- *TPRES*: vetor contendo o valor de força de superfície que atua em cada nó sob condição de contorno em uma direção específica;

- *NBCT*: quantidade de condições de contorno de força de superfície aplicadas.

Apesar de mais complicado, optou-se por aplicar condições de contorno de tensão sobre os elementos por permitir maior facilidade de descrição das cargas, visto que a maioria delas consiste, na realidade, em pressões aplicadas. Assim, a aplicação de tensão sobre o elemento permite a definição de uma condição de pressão constante normal ao elemento apenas definindo um valor fixo para a diagonal principal do tensor de tensão. Apenas essa informação já é suficiente para que a função calcule todos os vetores de força de superfície sobre os nós de modo a que fiquem sempre normais à superfície do elemento.

Terminado esse processo, finaliza-se a função *BCS.m*, que devolve ao fluxo principal de dados do programa as variáveis geradas para as condições de contorno de força de superfície nodal.

3.2.3 As funções *FRMTRX1.m* e *FRMTRX2.m*

Uma vez pronto todo o conjunto de variáveis contendo as condições de contorno, a geometria do problema, a malha de elementos de contorno e os valores constantes de propriedades aplicados ao problema, é necessário resolver o problema. Para que isso seja efetuado é preciso aplicar, adequadamente, cada uma das variáveis do problema à

equação integral de contorno que descreve o problema de elasticidade isotrópico e estático pelo método dos elementos de contorno. Isso é o que faz a função *FRMTRX1.m*.

Como é possível notar, existem duas funções que podem ser chamadas para formar as matrizes do sistema. Isso ocorre pois, tanto uma quanto a outra é capaz de formar a matriz A e o vetor BT, que são a expressão matricial de todas as equações integrais de contorno que constituem o problema. No entanto, apenas uma das duas é capaz de resolver as equações integrais e formar as duas matrizes principais do método dos elementos de contorno, que são as matrizes H, contendo as integrais para as soluções fundamentais de força de superfície e G, contendo as integrais para as soluções fundamentais de deslocamento.

A função *FRMTRX2.m* apenas lê as matrizes H e G formadas pela *FRMTRX1.m* em alguma análise anterior. A partir das matrizes lidas a função é capaz de manipular as colunas de ambas de acordo com as condições de contorno até formar a matriz A e o vetor BT. O processo de manipulação das colunas de H e G a partir das condições de contorno é chamado de permutação de colunas e tem por objetivo separar as incógnitas dos valores conhecidos.

Como visto no capítulo 2, a equação integral de contorno pode ser escrita de forma matricial, agrupando as integrais que multiplicam os deslocamentos em uma matriz H e as integrais que multiplicam as forças de superfície em uma matriz G. É possível afirmar que as matrizes H e G serão calculadas em sua totalidade através das integrações numéricas das soluções fundamentais. Já os vetores contendo os deslocamentos e forças de superfície, que multiplicam essas matrizes, não estão totalmente determinados. Na verdade, apenas alguns deslocamentos e algumas forças de superfície são conhecidas como condições de contorno.

Num problema bem proposto, a quantidade de condições de contorno de deslocamento e força de superfície, somadas, representam número suficiente para resolver o sistema de equações e obter os demais valores que completam os vetores de

deslocamento e força de superfície. Para que isso possa ser concretizado, basta isolar numa matriz os coeficientes das variáveis que permanecem como incógnitas e em outra aqueles cujas variáveis já estão determinadas como condições de contorno. Isso dá origem a uma matriz que é chamada, tradicionalmente, de A e um vetor BT, fruto da operação de multiplicação entre a matriz dos coeficientes das variáveis conhecidas pelo vetor das respectivas condições de contorno.

Para isolar os coeficientes das variáveis desconhecidas a função *FRMTRX2.m* trabalha em duas etapas. Primeira, lê as condições de contorno de força de superfície e, fazendo uso da função auxiliar *FMATR.m*, gera o vetor BT, chamado de segundo membro, apenas com as contribuições das forças de superfície. Além desse vetor, a função devolve a matriz AO, contendo as colunas da matriz G que correspondem aos coeficientes das variáveis desconhecidas. O processo pelo qual isso é feito será explicado quando a função responsável for comentada.

Após a primeira etapa, a função lê as condições de contorno de deslocamento prescrito e, caso o deslocamento seja diferente de zero, acrescenta sua contribuição ao vetor BT, realizando o produto da integral da solução fundamental em H pelo valor de deslocamento correspondente e somando o resultado ao vetor BT na posição adequada. Caso o deslocamento seja nulo, apenas um zero é colocado no vetor BT na posição adequada. Juntamente com isso, as colunas da matriz G, correspondentes aos valores desconhecidos de força de superfície, são colocadas em substituição na matriz H.

O processo segundo o qual essas permutações ocorrem é bastante evidente e depende apenas das condições de contorno. Para compreender mais facilmente o processo é necessário lembrar um pouco da estrutura envolvida na equação matricial que representa as equações integrais de contorno. Basicamente, existe uma matriz H multiplicando um vetor de deslocamentos nodais, sendo que cada linha do vetor corresponde a um deslocamento de um nó em uma determinada direção, e uma matriz G multiplicando um

vetor de forças de superfície, no qual cada linha corresponde a uma tração em uma dada direção.

Por se tratar de um problema tridimensional, cada nó poderá apresentar três graus de liberdade. No caso de deslocamentos, esses três graus de liberdade significam três deslocamentos possíveis, um em cada direção. Porém, quando se fala em deslocamento, existe uma condição de compatibilidade que precisa ser obedecida, segundo a qual um nó só pode apresentar um valor de deslocamento em cada grau de liberdade. Assim, mesmo que um nó esteja na fronteira entre dois elementos, portanto pertencente a ambos, apresentará somente um valor de deslocamento.

Para o caso de força de superfície nodal, também podem existir três para cada nó, uma em cada direção ou grau de liberdade, no entanto, não existe a condição de compatibilidade. Dessa forma, cada um dos oito nós de cada elemento terá seus próprios vetores de força de superfície. Isso significa que nós compartilhados entre dois ou mais elementos terão dois ou mais vetores de força de superfície em cada grau de liberdade, correspondentes à contribuição de força de superfície de cada elemento.

Isso faz com que existam diferenças fundamentais entre as matrizes H e G. Devido à condição de compatibilidade de deslocamentos, a matriz H apresenta igual número de linhas e colunas, correspondentes à quantidade de graus de liberdade do problema. Já a matriz G não é quadrada, apresentando número de linhas correspondente à quantidade de graus de liberdade, mas número de colunas correspondente à quantidade de elementos multiplicada pelo número de nós por elemento multiplicado pela quantidade de graus de liberdade por nó.

Dessa forma, nas matrizes H e G, cada coluna corresponde ao coeficiente multiplicador de um grau de liberdade, seja de deslocamento ou força de superfície, e cada linha corresponde a uma equação diferente até perfazer o número necessário à solução do sistema.

Considerando que as condições de contorno são valores determinados para alguns dos graus de liberdade de deslocamento e força de superfície, substituí-los nas posições correspondentes dos vetores de deslocamento e força de superfície é como substituir valores conhecidos das variáveis em todas as equações dispostas nas linhas da matriz. Da mesma forma, para isolar esses valores conhecidos, basta mover toda a coluna correspondente ao componente do vetor que é conhecido e realizar a multiplicação matricial para gerar um componente do vetor *BT*.

De modo semelhante, os componentes desconhecidos dos vetores de deslocamento e força de superfície correspondem às incógnitas do problema e os valores que multiplicam nas matrizes *H* e *G* são seus coeficientes dispostos em colunas. Assim, para agrupar os coeficientes dos termos desconhecidos, basta separar as colunas da matriz que correspondem ao componente a ser determinado no vetor de deslocamento ou força de superfície.

O procedimento é válido literalmente para o caso de isolar as colunas da matriz *H*, pois nela existe apenas uma coluna para cada grau de liberdade do problema. No entanto, quando são permutadas as colunas da matriz *G* surge um problema, pois nela pode existir mais de uma coluna para o mesmo grau de liberdade de um nó, referente a dois ou mais elementos que o compartilham. Para resolver esse problema é necessário somar as colunas de *G* que são relativas a um mesmo nó global. Embora esse procedimento seja correto apenas para os casos em que os elementos envolvidos tenham normais praticamente coincidentes, foi adotado de forma genérica, necessitando de revisão posterior.

Após separadas as colunas de *H* e *G* correspondentes às incógnitas do sistema elas são reagrupadas em uma matriz *A*. Da mesma forma, as colunas correspondentes aos valores determinados pelas condições de contorno são separadas e calculadas, dando origem a um vetor chamado de *BT*. Isso encerra a função *FRMTRX2.m*, que retorna a matriz *A* e o vetor *BT* ao fluxo principal de dados do programa.

A função *FRMTRX2.m* é utilizada todas as vezes que se deseja refazer uma análise previamente concluída apenas alterando as condições de contorno, partindo da mesma geometria. Caso se esteja realizando a análise pela primeira vez ou tenha ocorrido alguma alteração de geometria ou propriedades do material, é necessário utilizar a função *FRMTRX1.m*.

Em sua parte final, *FRMTRX1.m* e *FRMTRX2.m* são idênticas, realizando a permutação das colunas de H e G de acordo com as condições de contorno até gerar a matriz A e o vetor BT. No entanto, *FRMTRX1.m* dispõe das ferramentas necessárias para realizar integrações numéricas e formar as matrizes H e G do método dos elementos de contorno.

Inicialmente, a função se encarrega de calcular todas as funções de forma e suas derivadas que serão necessárias durante o processo de integração. Isso significa um conjunto de oito funções de forma, correspondentes a cada um dos oito nós do elemento, para cada um dos pontos de integração de Gauss. No caso de serem utilizados oito pontos de Gauss, ao se integrar pela área do elemento, esses pontos são colocados em cada aresta do quadrilátero, sendo que um ponto de integração corresponde ao cruzamento entre dois pontos de duas arestas. Isso dá origem a sessenta e quatro pontos de integração sobre a área, exigindo sessenta e quatro funções de forma, uma para cada ponto.

Esse procedimento pode ser visto no código da função, no anexo 3, através da chamada da função auxiliar *SHAP8.m* para cada par de pontos de Gauss até compor os sessenta e quatro pontos de integração. As funções de forma e suas derivadas em relação a cada uma das duas coordenadas intrínsecas são armazenadas, respectivamente, nas matrizes *SF*, *DG* e *DH*.

Juntamente com a chamada de *SHAP8.m* é feita a chamada para uma função de nome *LINEAR.m*, que também retorna matrizes contendo os valores de funções de forma e derivadas delas em relação às coordenadas intrínsecas. A diferença entre as funções de forma determinadas por *SHAP8.m* e *LINEAR.m* é que a primeira função trabalha com

funções de forma quadráticas, enquanto a segunda trabalha com funções de forma lineares. As funções de forma lineares são armazenadas nas matrizes *SSF*, sendo suas derivadas colocadas em *SDG* e *SDH*.

As funções de forma quadráticas são utilizadas normalmente no programa para avaliar o comportamento das variáveis nodais ao longo do elemento, o que é necessário para o processo de integração. Para que possam ser utilizadas, as funções de forma quadráticas necessitam de três nós por aresta do elemento, o que é fornecido pelo elemento quadrilateral de oito nós. No entanto, em um momento da integração isso não é possível.

Durante o processo de integração singular, o elemento quadrilateral de oito nós é transformado num elemento quadrilateral de quatro nós, para que dois de seus nós possam ser colapsados num só sobre o ponto fonte, levando o Jacobiano para zero e cancelando a singularidade. Isso significa que as funções de forma quadráticas não podem ser utilizadas, visto que o elemento dispõe apenas de quatro nós, ou seja, dois por aresta. Assim, funções de forma lineares tornam-se necessárias, bem como suas derivadas, a fim de possibilitar a integração singular.

Uma vez calculadas as funções de forma necessárias, a função passa ao processo de montagem das matrizes H e G, o que envolve a integração das soluções fundamentais como parte da solução das equações integrais de contorno. Para isso é necessário estabelecer em que elemento se está trabalhando e qual é o ponto fonte que se considera para as soluções fundamentais.

O elemento é necessário pois define os nós que serão utilizados, suas coordenadas, a geometria, o que permite calcular entre outras coisas, o Jacobiano. Já o ponto fonte nada mais é que um dos nós do problema e sua determinação é necessária pois é utilizado para resolver a solução fundamental, definindo se a integral será singular ou não.

É possível realizar a montagem das matrizes partindo de um elemento fixo, calculando os componentes variando os pontos fonte, ou mantendo o ponto fonte fixo e calcular variando os elementos. Tanto um quanto outro levam ao mesmo resultado. Tradicionalmente, nos livros texto que descrevem o método de formação das matrizes H e G, como *Domingues* (1993), a técnica abordada é a da construção fixando o ponto fonte e calculando os componentes das matrizes para cada elemento. Nas matrizes, esse procedimento significa montá-las por linhas, uma vez que cada linha corresponde a um grau de liberdade de um ponto fonte e cada coluna a um grau de liberdade de um nó do elemento.

No entanto, apesar da montagem tradicional ocorrer por linhas, *Kane* (1994) comenta que esse método não é o mais eficiente, uma vez que, ao fixar o ponto fonte e variar os elementos está se obrigando o programa a recalcular as coordenadas dos nós do elemento, o que altera os Jacobianos e os vetores normais. Já se for escolhido um elemento fixo e se variarem os pontos fonte, as coordenadas dos nós do elemento são sempre as mesmas, o que significa que os Jacobianos e as normais não precisam ser recalculados. Ainda segundo *Kane* (1994), esse processo de reutilização pode se tornar mais eficiente caso se utilizem quantidades de pontos de Gauss adequadas à distância que o elemento se encontra do ponto fonte, quanto mais próximo, maior a quantidade de pontos de Gauss. A esse algoritmo dá-se o nome de *RISP* (*Reusable Intrisic Sample Point*).

Nesse trabalho foi implementada apenas a parte de reutilização do algoritmo *RISP*, fazendo uso de um número fixo de pontos de Gauss para todas as integrações. Isso possibilitou uma maior simplicidade no código e foi suficiente para demostrar os ganhos em velocidade que o método é capaz de oferecer, como será comentado no capítulo de resultados.

Portanto, para esse trabalho será utilizada a técnica de montagem das matrizes H e G por colunas. Dessa forma, assim que se fixa o elemento para o processo de integração já é

possível obter as coordenadas de seus nós, assim como calcular o Jacobiano e os vetores normais para todos os nós. Além disso, no código implementado, é feito o cálculo de uma variável substituta que envolve o Jacobiano e os pesos de Gauss para a integração, substituindo esses dois valores por uma única variável chamada *FACT*.

Após determinar as variáveis que serão reutilizadas para todos os pontos fonte, o programa entra no processo de integração fixando o primeiro ponto fonte. O passo seguinte é verificar se o ponto fonte pertence ao elemento em integração. Caso não pertença é acionada a função de integração convencional, caso contrário é chamada a função de integração singular, pois para um dos nós do elemento a solução fundamental representará uma singularidade.

No caso de uma integração convencional, a função chamada é a *INTGR.m*, que devolve para *FRMTRX1.m* seis vetores contendo as linhas das matrizes H e G do elemento calculadas para o ponto fonte em questão. Dessa forma, os vetores *A1*, *A2* e *A3* armazenam as três linhas da matriz elementar H, enquanto *B1*, *B2* e *B3* armazenam as três linhas da matriz elementar G. Caso a integração seja singular a função chamada é *ITSS.m*, que devolve os mesmos vetores para *FRMTRX1.m*.

Tanto a matriz elementar G quanto H são de tamanho 3 x 24, sendo que as três linhas representam as três direções em que a carga concentrada pode ser aplicada no ponto fonte e as vinte e quatro colunas representam os três graus de liberdade de cada um dos oito nós (3 x 8 = 24) que constituem o elemento e nos quais a solução fundamental deve ser avaliada.

Uma vez determinadas, as matrizes elementares precisam ser colocadas adequadamente nas matrizes globais H e G. Para que isso seja feito é necessário obedecer a algumas regras. No caso da matriz H elementar, cada uma de suas oito colunas correspondentes aos nós locais do elemento precisam ser colocadas nas colunas correspondentes aos nós globais da matriz H global, o que é feito com o auxílio da matriz de conectividade. Já a matriz G elementar pode ser colocada diretamente na matriz G

global, respeitando apenas os posicionamentos de linha e coluna dados pelo ponto fonte que se utiliza e o elemento de integração.

Tanto para H quanto para G globais as linhas são determinadas pelo ponto fonte em que se está, sendo que três linhas são necessárias para cada ponto fonte. Já quanto às colunas, na matriz H global correspondem diretamente aos nós globais do problema, sendo necessárias três colunas para cada nó. Na matriz G global, as colunas correspondem aos elementos, sendo necessárias vinte e quatro para cada elemento, uma vez que cada elemento apresenta oito nós e cada nó três graus de liberdade.

Assim, para colocar a matriz H elementar nas posições adequadas da matriz H global, basta verificar duas coisas. Qual o ponto fonte da integração, o que determinará a posição das três linhas da matriz H global que receberão as três linhas da matriz H elementar. Quais os números globais dos nós que compõe o elemento, correspondendo a coluna do nó local com sua coluna global.

No caso da matriz G elementar a colocação na matriz G global é similar. Para obter as linhas da matriz global basta verificar o ponto fonte do processo de integração. Para obter as colunas da matriz global, basta saber qual é o elemento sobre o qual se está integrando, pois as colunas de G global correspondem a um elemento para cada bloco de vinte e quatro. Encontradas a primeira linha e primeira coluna na matriz G global, basta transferir seqüencialmente a matriz G elementar a partir dessa posição.

Colocadas as matrizes elementares nas matrizes globais a função passa ao próximo ponto fonte, mantendo o mesmo elemento e repete todo o procedimento. Ao término de todos os pontos fonte a função troca para o próximo elemento e recomeça tudo a partir do primeiro ponto fonte. Terminado o último elemento estarão concluídas as matrizes H e G globais, armazenadas nas matrizes *h_global* e *g_global* respectivamente.

O próximo passo da função é calcular a diagonal principal da matriz H global, que não é calculada durante o processo de integração por envolver uma singularidade do tipo

$1/r^2$, para a qual uma abordagem indireta é menos custosa e mais indicada. A abordagem indireta é baseada na hipótese de movimento de corpo rígido.

A hipótese consiste em supor que as condições de contorno do problema sejam aplicadas em todos os nós na forma de deslocamentos que causem movimento de corpo rígido do modelo em uma determinada direção. Nesse caso, as forças de superfície sobre os nós são nulas, anulando a matriz G global. Então, como o valor dos deslocamentos é diferente de zero, para que a igualdade do equilíbrio estático da equação 20, $[H] \times \{u\} = [G] \times \{t\}$, seja mantida, a matriz H precisa ser tal que, multiplicada pelo vetor u resulte zero.

Para facilitar os cálculos, pode-se considerar o vetor u de deslocamentos unitários, então, cada linha da matriz H global deverá apresentar a soma igual a zero. Isso significa que, se há apenas um valor desconhecido, ele será igual ao inverso da soma dos demais, para que a soma toda se anule. Esse é o princípio usado para determinar a diagonal principal da matriz H.

No caso tridimensional existem três posições em cada linha que ficam indeterminadas como diagonal principal de H, pois toda uma matriz de ordem três deixa de ser calculada quando se está na diagonal principal. Assim, é necessário repetir a hipótese de movimento de corpo rígido para mais duas direções, o que conduz à regra de que o inverso da soma dos valores colocados na linha para um determinado grau de liberdade corresponde ao valor da diagonal principal desse grau de liberdade. Exemplificando, se quiser determinar o valor da diagonal principal do elemento da matriz H global que está na linha 1, coluna do grau liberdade x do nó 1, é preciso somar todos os valores nessa linha de todos os demais nós no grau de liberdade x. O inverso desse valor será o valor procurado. Esse procedimento deve ser aplicado em toda a matriz H, determinando completamente sua diagonal principal.

Ao fim dessa etapa, as matrizes H e G globais estão completamente determinadas. Como pode ser notado, não foram utilizadas condições de contorno para constituir essas

matrizes, portanto, elas apenas refletem o problema com sua geometria, independente das condições de contorno. Por esse motivo, nesse ponto, a função salva as matrizes *h_global* e *g_global* em um arquivo padrão chamado *HeG.mat*, o qual pode ser lido pela função *FRMTRX2.m* para reutilização futura, poupando todo o procedimento de cálculo das integrais.

Após essa etapa a função *FRMTRX1.m* transcorre da mesma forma que a *FRMTRX2.m* até a formação da matriz A e do vetor BT que compõe a equação matricial a ser resolvida.

3.2.4 A função *OUTPUT.m*

Uma vez formada a matriz A e o vetor BT caracterizando o sistema matricial a ser resolvido são aplicadas técnicas numéricas de solução de sistemas de equações em arranjo matricial, tais como a eliminação de Gauss. No caso do programa E-Con3D, a solução do problema é feita através de uma função interna do ambiente MatLab®, que consiste num operador específico chamado *back slash*, o qual realiza a verificação do tipo de matriz que caracteriza o sistema de equação e se encarrega de escolher o método mais adequado.

Terminado o processo de solução numérica ficam disponíveis, no fluxo principal do programa, as solução chamadas de primárias do problema no contorno. Essas soluções ficam armazenadas no vetor B. Assim, esse vetor contém soluções de deslocamentos e forças de superfície nodais.

Para que seja possível obter resultados mais interessantes, tais como as tensões nodais, é necessário separar os resultados de deslocamento e forças de superfície, assim como organizá-los em conjunto com as condições de contorno, a fim de que componham

vetores de resposta para todos os nós do problema. Esse trabalho de classificação dos resultados primários é uma das tarefas da função *OUTPUT.m*.

Dispondo das condições de contorno aplicadas inicialmente ao programa e das respostas obtidas com a solução do problema por elementos de contorno, a função *OUTPUT.m* trabalha separando os resultados de deslocamento daqueles de força de superfície. Nesse processo, são geradas duas matrizes, uma delas contendo os valores de deslocamento para cada um dos nós do problema em cada uma das três direções possíveis, chamada de *Desloc*, e uma outra contendo os valores de forças de superfície aplicadas aos nós em cada elemento chamada de *Tracao*.

A matriz *Desloc* recebe os deslocamentos nodais calculados como resposta direta para os nós que não se encontravam restritos, completando os demais nós de acordo com as condições de contorno de deslocamento impostas.

A matriz *Tracao* recebe as forças de superfície calculadas para cada um dos nós de acordo com o elemento a que pertencem. Assim, diferentemente dos deslocamentos, as forças de superfícies permanecem vinculadas aos elementos para os quais foram calculadas. Isso é necessário pois podem existir dois ou mais valores de força de superfície para cada nó numa mesma direção, uma vez que não há necessidade de continuidade como no caso de deslocamento. Então, os resultados primários de forças de superfície são armazenados de acordo com os elementos em relação aos quais foram calculados, a fim de que seja possível calcular as respostas secundárias como a tensão nodal.

A partir dos vetores de resposta de deslocamento, *Desloc*, e forças de superfície, *Tracao*, a função realiza a chamada de uma função auxiliar chamada *TensNodal.m*, responsável por obter as respostas secundárias na forma de tensores de tensão em cada um dos nós do problema.

Finalizado o processo de classificação das respostas primárias e cálculo das respostas secundárias para o problema proposto, são devolvidas ao fluxo principal de dados do programa as matrizes *Desloc*, *Tracao* e *SigNodal*, contendo os deslocamentos nodais, as respostas de forças de superfície por nó de cada elemento e os tensores de tensão para cada nó do problema.

3.3 **Funções auxiliares**

As funções auxiliares são aquelas responsáveis por tratar ou fornecer dados necessários ao fluxo interno de informação das funções principais. Dessa forma, seu funcionamento depende sempre da estrutura interna de dados das funções principais, ou seja, recebem dados que estão dentro da função principal e retornam dados exclusivamente para aquela função que solicitou.

Diferentemente das funções principais, que são chamadas uma única vez ao longo do programa principal, as funções auxiliares podem ser chamadas diversas vezes, de acordo com o tipo de dados que uma função principal necessite. Nessa estrutura, as funções auxiliares são:

- *COORD.m*;
- *SHAP8.m*;
- *LINEAR.m*;
- *JACOBI.m*;
- *TRIANG.m*;
- *KERNEL.m*;

- *INTGR.m*;
- *ITSS.m*;
- *FMATR.m*;
- *TENSNODAL.m*.

3.3.1 Função *COORD.m*

Essa função é utilizada para fornecer as coordenadas dos oito nós relacionados a um determinado elemento. Para isso a função precisa receber o número do elemento para o qual se deseja identificar os nós, a matriz de conectividade *NOEL* e os vetores contendo as coordenadas de todos os nós do problema, que são *X*, *Y* e *Z*. A partir dessas informações a função se encarrega de identificar os nós que fazem parte do elemento requerido e informar as coordenadas desses nós em uma matriz chamada *CORD*, que retorna à função de partida, contendo as coordenadas de cada nó do elemento em seqüência por linhas.

3.3.2 Função *SHAP8.m*

A função *SHAP8.m* é chamada quando se necessita obter as funções de forma quadráticas e suas derivadas em relação às coordenadas intrínsecas. Para isso, são necessárias informações como as coordenadas intrínsecas em relação às quais se deseja avaliar as funções de forma e suas derivadas, o número de pontos de Gauss preestabelecido e uma variável de controle.

Uma vez calculadas as funções de forma e derivadas, os valores ficam armazenados em uma matriz chamada *DE*, que dispõe as derivadas em relação à primeira coordenada intrínseca na primeira linha, as derivadas em relação à segunda coordenada intrínseca na segunda linha e as funções de forma na terceira linha. Cada uma das colunas da matriz corresponde diretamente a um nó do elemento em sua numeração local, que vai de um até oito.

Para o caso em que é necessário calcular diversos conjuntos de função de forma, como para a integração, quando se necessita de um conjunto de oito funções de forma e suas derivadas para cada um dos pontos de Gauss, é utilizada a variável de controle *KY*. Quando o valor dessa variável de controle é igual a zero e o número de pontos de Gauss é especificado, para cada ciclo de cálculo envolvendo a função *SHAP8.m* os valores das derivadas e das funções de forma são repassados da matriz *DE* para matrizes *DG*, *DH* e *SF*. Isso é necessário pois a matriz *DE* é apagada a cada vez que a função é chamada, a fim de permitir o cálculo correto das derivadas e funções de forma, no entanto, é necessário armazenar os valores calculados para uso posterior.

O armazenamento ocorre nas matrizes *DG*, para as derivadas das funções de forma em relação à primeira coordenada intrínseca, *DH*, para as derivadas das funções de forma em relação à segunda coordenada intrínseca e *SF* para as funções de forma. Essas matrizes são atualizáveis e a função dispõe de um algoritmo de colocação de maneira que os valores relativos a cada ponto de Gauss ficam alocados adequadamente.

3.3.3 **Função *LINEAR.m***

Essa função é semelhante à *SHAP8.m*, mas sua aplicação ocorre exclusivamente durante o processo de integração singular, quando é necessário triangularizar o elemento quadrilateral, mas continuar tratando-o como se fosse quadrilateral. A função *LINEAR.m*

é responsável por calcular funções de forma lineares, ao invés das quadráticas originais, para que seja possível tratar os elementos após a triangularização.

Como visto, o processo de integração singular envolve a triangularização do elemento quadrilateral em dois ou três triângulos, mas que continuam sendo tratados como elementos quadrilaterais. Ocorre que, durante o processo de triangularização, um ou mais nós localizados nas arestas do elemento se transformam em nós de vértices dos triângulos. Isso faz com que os novos triângulos apresentem algumas arestas sem nós intermediários, o que impede o uso de funções de forma quadráticas nessas arestas. Assim, para que se possa tratar os novos elementos geometricamente triangulares como quadrilaterais e para que seja possível colapsar dois nós em um vértice, anulando o Jacobiano, é necessário utilizar funções de forma lineares, para as quais dois nós extremos são suficientes.

As funções de forma lineares são calculadas de maneira semelhante às quadráticas, mas utilizando equações lineares, e são armazenadas em matrizes distintas, sendo *SSF* para as funções de forma, *SDG* para as derivadas das funções de forma em relação à primeira coordenada intrínseca e *SDH* para as derivadas das funções de forma em relação à segunda coordenada intrínseca.

Analogamente às matrizes que armazenam as funções de forma quadráticas, o programa dispõe de um algoritmo para colocação dos valores de forma adequada, identificando os pontos de Gauss e os quatro nós com os quais as funções de forma e suas derivadas se relacionam.

3.3.4 **Função *JACOBI.m***

Essa função é responsável por calcular o Jacobiano e o vetor unitário normal para o elemento em cada ponto de integração de Gauss. Para isso necessita das funções de forma e suas derivadas em relação aos pontos de Gauss, das coordenadas dos nós do elemento e de uma variável de controle.

Da mesma forma que a função *SHAP8.m*, a *JACOBI.m* pode ser utilizada para calcular apenas um Jacobiano e um vetor unitário normal por vez, relativo apenas a um ponto definido. Para isso, precisa apenas da matriz *DE*, contendo as derivadas e funções de forma para os oito nós relativas ao ponto definido, das coordenadas dos nós do elemento e da variável de controle *KY* diferente de zero.

Quando a variável de controle é igual a zero a função usará as derivadas e funções de forma armazenadas nas matrizes *DG*, *DH* e *SF*, que são calculadas para todos os pontos de Gauss, criando um vetor de Jacobianos e uma matriz com os vetores unitários normais relacionados. Essas matrizes são devolvidas à função de partida sob os nomes *JACOB* e *UN*.

3.3.5 **Função *TRIANG.m***

Função responsável pela triangularização e determinação dos novos Jacobianos e vetores unitários normais para os elementos quadrilaterais de quatro nós. Sua utilização se restringe ao processo de integração singular e envolve o uso das funções *SHAP8.m* e *JACOBI.m*.

De acordo com o nó ponto fonte em integração no elemento, a função responsável pela integração singular determina se o elemento deverá ser triangularizado em dois ou três triângulos. A partir dessa informação, mais a quantidade de pontos de Gauss, as coordenadas dos nós do elemento, o número do nó ponto fonte e das derivadas e funções de forma lineares e quadráticas, a função *TRIANG.m* é chamada.

Uma vez dentro da função *TRIANG.m* um algoritmo de verificação se encarrega de renumerar os nós de modo a que correspondam a nós de elementos quadrilaterais de quatro nós, mas dentro do sistema de coordenadas intrínseco dos elementos quadrilaterais de oito nós. Após essa etapa é calculada uma matriz Jacobiana que dará origem a um multiplicador chamado *DETER*. Esse multiplicador será aplicado ao Jacobiano convencional, calculado para os quatro nós do elemento quadrilateral, o que permitirá anular o Jacobiano quando ocorrer o colapso de dois vértices do quadrilátero em uma mesma coordenada correspondente ao nó ponto fonte.

Como respostas importantes para o fluxo de dados da função de integração singular, a função *TRIANG.m* retorna os Jacobianos num vetor chamado *JACO* e os vetores unitários normais numa matriz chamada *UN*.

3.3.6 **Função *KERNEL.m***

Essa função é responsável por calcular as soluções fundamentais necessárias para completar o integrando da equação integral de contorno, sejam elas de deslocamento ou de força de superfície, a partir de equações analíticas previamente deduzidas. Dessa forma, a função *KERNEL.m* apenas armazena as equações relacionadas com as soluções fundamentais para o problema da elasticidade isotrópica e as avalia adequadamente.

Para calcular as soluções fundamentais a função necessita das coordenadas dos nós do elemento, das propriedades do material, dos vetores unitários normais e das funções de forma. Uma vez chamada de forma correta, a função retorna as soluções fundamentais em duas matrizes, *UT* para as soluções fundamentais de deslocamento e *TT* para as soluções fundamentais de força de superfície.

Note-se que o tratamento para a singularidade foi feito anteriormente, pela função de integração que utiliza a função *KERNEL.m*. Assim, nenhum trabalho adicional é necessário para evitar problemas de cálculo, mas apenas uma verificação é feita para certificar a efetividade do tratamento aplicado à singularidade. Caso o tratamento não tenha sido eficiente, o raio calculado entre o ponto fonte e o ponto de integração será muito próximo a zero, o que faz com que a função interrompa o processo de cálculo e emita um aviso de erro.

3.3.7 **Função *INTGR.m***

Essa função é responsável pelo processo de integração quando não existe problema de singularidade no núcleo do integrando, ou seja, quando a solução fundamental não vai para infinito. Isso significa que o nó ponto fonte da integração não pertence ao elemento em integração, verificação que é realizada pela função que chama as funções de integração.

Para efetuar a chamada correta da função *INTGR.m* é necessário passar diversas variáveis como argumentos da função. Um conjunto dessas variáveis será utilizado para realizar a chamada da função *KERNEL.m* que, como visto anteriormente, necessita de argumentos para o cálculo das soluções fundamentais. Os demais argumentos passados são utilizados diretamente dentro da função *INTGR.m*.

Os argumentos utilizados dentro da função *INTGR.m* são a quantidade de pontos de Gauss, uma variável matricial chamada *FACT*, que aglutina o Jacobiano e os pesos de Gauss e as funções de forma para todos os pontos de Gauss. A quantidade de pontos de Gauss é utilizada para abrir as matrizes que servirão para o processo de integração, já a matriz *FACT* e as funções de forma são operadas para formar uma terceira variável chamada *SFACT*, que será utilizada para o cálculo final do integrando.

Uma vez preparadas todas as matrizes e calculadas as soluções fundamentais necessárias através da chamada da função *KERNEL.m*, a função de integração passa ao método de integração Gauss, que consiste em avaliar o integrando em cada um dos pontos de Gauss e somá-los. Esse procedimento é reduzido a operações matriciais pré-programadas no núcleo do programa MatLab®. Assim, a operação toda consiste em avaliar cada integrando, armazenando a matriz resultante em uma pilha, formando uma matriz tridimensional. Posteriormente, quando os integrandos para todos os pontos de Gauss estão calculados, é realizada uma soma ao longo da terceira dimensão da matriz, condensando todas as matrizes armazenadas em uma só que é o resultado da integração. Esse procedimento é feito tanto para a obtenção da matriz H quanto para a obtenção da matriz G do elemento.

Terminado o processo de integração a função devolve seis vetores, sendo que os vetores *A1*, *A2* e *A3* armazenam a matriz H do elemento por linhas e os vetores *B1*, *B2* e *B3* armazenam a matriz G do elemento por linhas.

3.3.8 Função *ITSS.m*

Essa função é responsável pela integração quando as condições para a manifestação de singularidade estão envolvidas no processo. Isso significa que o nó ponto fonte é um dos nós do elemento em integração, o que faz com que, em um determinado momento da

integração, a solução fundamental torne-se singular. O problema é que, quando isso ocorre, o método numérico de integração falha, pois envolve a avaliação numérica de um integrando que vai para infinito. Assim, é necessária uma técnica para contornar essa limitação.

No caso da função *ITSS.m* a técnica envolve a triangularização do elemento em integração com a colocação do nó ponto fonte em um dos vértices de um dos triângulos gerados. Essa transformação, aliada ao fato de se tratar o elemento geometricamente triangular como intrinsecamente quadrilateral de quatro nós com dois dos nós de vértice colapsados na mesma coordenada do nó ponto fonte, permite fazer com que o Jacobiano desse elemento vá para zero quando a solução fundamental vai para infinito. Esse comportamento é suficiente para cancelar a singularidade, permitindo a integração através do método de Gauss convencional, mas utilizando o novo conjunto de Jacobianos e funções de forma definidos para os elementos quadrilaterais de quatro nós formados pelo processo de triangularização.

O processo de triangularização tem início dentro da função *ITSS.m*, que escolhe, de acordo com a posição do nó ponto fonte dentro do elemento, quantos elementos triangulares serão gerados. Assim, se o nó ponto fonte está localizado num dos vértices do elemento quadrilateral de oito nós, a função determina a geração de apenas dois triângulos, dividindo o quadrilátero pela diagonal que contém o nó ponto fonte. Já se o nó ponto fonte se localiza numa das arestas do elemento quadrilateral de oito nós, a função determina a geração de três triângulos a partir do nó ponto fonte e dos dois vértices opostos a ele.

Uma vez definida a quantidade de triângulos é chamada a função *TRIANG.m*, que calcula o novo Jacobiano, os novos vetores unitários normais e as novas funções de forma e suas derivadas. Essa variáveis retornam à função *ITSS.m* com os nomes de *JACO*, *UN* e *DE*, respectivamente.

A partir desse ponto o funcionamento é idêntico ao da função *INTGR.m*, pois o processo de integração numérica é o mesmo.

3.3.9 Função *FMATR.m*

Essa função é utilizada pelas funções encarregadas de formar a matriz A e o vetor BT para o sistema de equações matricial. Sua tarefa é manipular a matriz global G, a partir das condições de contorno, de maneira a isolar a parte conhecida de G, que se relaciona com as condições de contorno de força de superfície, da parte desconhecida, que se relaciona com um valores desconhecidos de força de superfície.

A partir das condições de contorno de deslocamento prescrito são determinadas as colunas da matriz G global que estão relacionadas com incógnitas do problema, o que significa que colunas completas da matriz G podem ser passadas diretamente para dentro da matriz A, iniciando a formação do sistema matricial. Por outro lado, as condições de contorno de força de superfície determinam colunas da matriz G global que podem ser operadas para formar o vetor BT dos termos conhecidos.

Assim, as colunas da matriz G global que podem ser operadas com as condições de contorno passam por essa operação e são armazenadas num vetor BT. Já as colunas que se relacionam com incógnitas do problema são armazenadas dentro de uma matriz temporária AO, que será posteriormente unida com as colunas provenientes da matriz H global para formar a matriz A do sistema matricial.

Note-se que existem muitos nós de elementos que não possuem condição de contorno definida, seja de deslocamento ou de força de superfície. Para esses nós é assumida uma condição de superfície livre, o que significa que as forças de superfície são

nulas. Isso faz com que diversas colunas da matriz G global resultem em valores nulos no vetor BT.

Uma vez terminado o processo de separação são devolvidos o vetor BT e a matriz AO, para que o processo de permutação de colunas continue dentro da função responsável pela formação total da matriz A e do vetor BT.

3.3.10 **Função *TENSNODAL.m***

Essa função foi colocada dentro do fluxo principal de dados do programa, logo após a função *OUTPUT.m*, no entanto, está sendo descrita na seção de funções auxiliares, pois apenas trata os resultados finais do problema. Isso significa que sua presença não é fundamental para que o problema seja resolvido, já que sua função é a de transformar as respostas primárias do problema em respostas mais interessantes do ponto de vista da engenharia.

As respostas primárias, oferecidas pelo método como solução para o problema, são valores de deslocamentos e forças de superfície nodais. Na prática pouco se pode fazer apenas com esses valores. Dados mais interessantes e que possibilitam estudos através de teorias de falha são obtidos através de tensões nodais, o que significa que é necessário obter um tensor de tensões para cada um dos nós do problema como passo inicial para obter qualquer valor comparativo. O processo de obtenção do tensor de tensões para cada nó foi descrito no capítulo 2 e é o mesmo implementado na função *TENSNODAL.m*.

Para realizar os cálculos a função precisa receber as matrizes contendo as soluções primárias de força de superfície e deslocamento, as quantidades de elementos e nós do problema, a matriz de conectividade, os vetores de coordenadas de todos os nós do problema e as propriedades do material.

Seguindo o modelo de cálculo proposto no capítulo 2 para cada elemento, a função inicia obtendo as coordenadas dos oito nós do elemento. Para cada nó do elemento são obtidos os vetores unitários normais, utilizando as funções *SHAP8.m* e *JACOBI.m*. Em seguida é feito o cálculo do vetor tangente ao contorno do elemento no nó, de modo que seja positivo no sentido crescente da coordenada intrínseca associada ao elemento. Finalmente, é calculado um vetor perpendicular aos vetores unitário normal e tangente. Isso permite gerar um sistema de coordenadas local sobre o nó em avaliação.

Uma vez estabelecido o sistema de coordenadas local é definida uma matriz de transformação do sistema de coordenadas global para o novo sistema local. Normalmente, essa matriz é definida em termos dos cossenos diretores que transformam os vetores de um sistema para o outro. No entanto, como os vetores utilizados para definir o sistema local são unitários, basta colocá-los adequadamente dentro de uma formação matricial para determinar a matriz de transformação. Nesse caso, para transformar os vetores do sistema global para o local, basta dispor os versores que definem o sistema local em linhas numa matriz, sendo a primeira linha correspondente ao versor tangente, a terceira ao versor normal e a segunda ao versor perpendicular aos dois anteriores.

Essa matriz de transformação permite realizar as transformações de todos os vetores para o sistema de coordenadas local, o que é necessário para que seja possível aplicar a técnica de recuperação de tensões descrita no capítulo 2. Após aplicada essa técnica é necessário transformar o tensor obtido no sistema local para o sistema global, a fim de que sua utilização seja viabilizada. Para isso, é necessário algum cuidado, uma vez que se trata da transformação de um tensor, não mais de vetores. Assim, para que a transformação ocorra corretamente, é necessário empregar a matriz de transformação definida para vetores duas vezes, uma antes do tensor, transposta, outra após o tensor, sem transposição.

Note-se que, assim como as forças de superfície, pode existir mais que um tensor por nó, uma vez que eles também se relacionam com o elemento e um nó pode ser

compartilhado entre dois ou mais elementos. Dessa forma, o que está sendo feito na função *TENSNODAL.m* é somar os tensores que se relacionem a um mesmo nó, de maneira que se possa calcular a média de todos os tensores somados no final do processo.

Terminado o procedimento de cálculo para todos os elementos do problema, a função devolve ao fluxo principal de dados do programa uma matriz chamada *SigNodal*, que contém os tensores para todos os nós do elemento armazenados por linhas. Cada linha da matriz corresponde a um nó e cada conjunto de três colunas corresponde às três colunas do tensor por linha. Assim, em um mesma linha encontram-se σ_{11}, σ_{12}, σ_{13}, σ_{21}, σ_{22}, σ_{23}, σ_{31}, σ_{32}, σ_{33}.

3.4 **Função de pós-processamento**

A função responsável pelo pós-processamento das respostas do problema trabalha, basicamente, com a visualização sólida do modelo e de um mapa de cores associado a valores que é colocado sobre a superfície do sólido. De maneira geral, qualquer resultado pode ser tratado pela função de pós-processamento, desde que a estrutura de dados de entrada, necessária para a chamada da função, seja atendida.

De maneira simplificada, a função de pós-processamento, chamada *SOLIDPOS.m*, trabalha reconstruindo o modelo geométrico a partir da malha de elementos de contorno. Isso significa que o pós-processador desenha cada um dos elementos de contorno a partir dos nós da geometria e da conectividade dos elementos. Para gerar o mapa de cores a função utiliza uma função interna do programa MatLab®, chamada *FILL3*, que gera polígonos tridimensionais a partir das coordenadas de seus vértices e os pinta individualmente de acordo com valores determinados para cada vértice do polígono. A

função associa automaticamente cada valor em cada vértice a uma cor numa escala de cores e interpola as cores na área interna do polígono de acordo com funções quadráticas.

O modo de se utilizar a função de pós-processamento depende de uma sintaxe que é oferecida juntamente com a função. É necessário entrar com o nome do problema analisado, o caminho para o diretório que contém os arquivos de dados do problema, a solução que se deseja mostrar, a direção dessa solução, a posição girada no plano *XY* da qual se deseja ver o sólido tridimensional e a elevação em relação a *z*, assim como o fator de escala para multiplicar os deslocamentos, dando origem à visualização do sólido deformado.

3.5 Programas para conversão de Ansys® para E-Con3D

Foram desenvolvidos dois programa para conversão de partes da base de dados criada em Ansys® para bases de dados utilizadas pelo programa E-Con3D. Ambos os códigos foram gerados a partir de funções encontradas nos manuais do usuário do programa Ansys® para execução direta interpretada dentro do ambiente de modelagem *prep7*, que é parte integrante do ambiente de análise do aplicativo.

Os dois códigos gerados encontram-se em anexo, sendo que a estrutura de ambos é, basicamente, idêntica. O programa *OUTELEM.inp* se encarrega de obter a matriz de conectividade dos elementos da malha gerada e exportá-la para um arquivo texto chamado *NOEL.txt*. O programa *OUTNO.inp* se encarrega de obter as coordenadas de todos os nós da malha gerada e exportá-las para um arquivo texto chamado *COORDNOS.txt*.

Tanto o código para leitura da matriz de conectividade quanto o de leitura das coordenadas dos nós utilizam a mesma estrutura de funcionamento. Em primeiro lugar é

criada uma matriz que servirá de variável temporária para o armazenamento das informações requeridas. Em segundo lugar é utilizado o comando *vget* para obter as informações. No caso da matriz de conectividade é solicitado que *vget* obtenha os valores da variável *elem* em suas oito posições referentes aos números dos nós componentes do elemento. No caso das coordenadas dos nós é solicitado que *vget* obtenha da variável *node* os valores de localização referentes às coordenadas x, y e z.

Posteriormente são utilizados comandos do módulo *output* do aplicativo Ansys®, a fim de que sejam escritos os valores separados nas variáveis temporárias em arquivos. São utilizados os comandos *vwrite* para escrever os valores contidos na variável temporária com uma formatação padrão de linguagem C, utilizando tabuladores como marcadores de colunas; e *list* para gerar o arquivo texto final.

Na estrutura de programação do código interpretado, o * precede um comando, a *!* marca comentários e a / precede o nome do módulo que será utilizado.

CAPÍTULO 4

ANÁLISE DE RESULTADOS

Nesse capítulo serão expostos resultados para problemas analisados utilizando o programa E-Con3D. Esses resultados abrangem a solução de quatro modelos propostos, sendo três deles destinados à verificação do funcionamento adequado do programa e um quarto para demonstrar as capacidades de aplicação em bioengenharia. Todos os modelos tiveram suas malhas desenvolvidas em Ansys®, utilizando o elemento finito de casca tipo *SHELL93*, que apresenta geometria quadrilateral com oito nós, idêntica àquela do elemento de contorno implementado no programa E-Con3D.

O primeiro problema trata de uma geometria cúbica, com arestas unitárias e malha de elementos de contorno mínima, o que significa apenas um elemento de contorno por face do cubo. As condições de contorno escolhidas foram de restrição de deslocamentos em x da face correspondente ao plano YZ em $x = 0$, restrição de deslocamento em y da face correspondente ao plano XZ em $y = 0$, restrição de deslocamento em z da face correspondente ao plano XY em $z = 0$ e aplicação de uma tensão de tração constante e uniforme de dez unidades sobre o elemento da face oposta ao plano YZ em $x = 0$. O modelo gerado bem como as condições de contorno aplicadas podem ser vistos na figura 10.

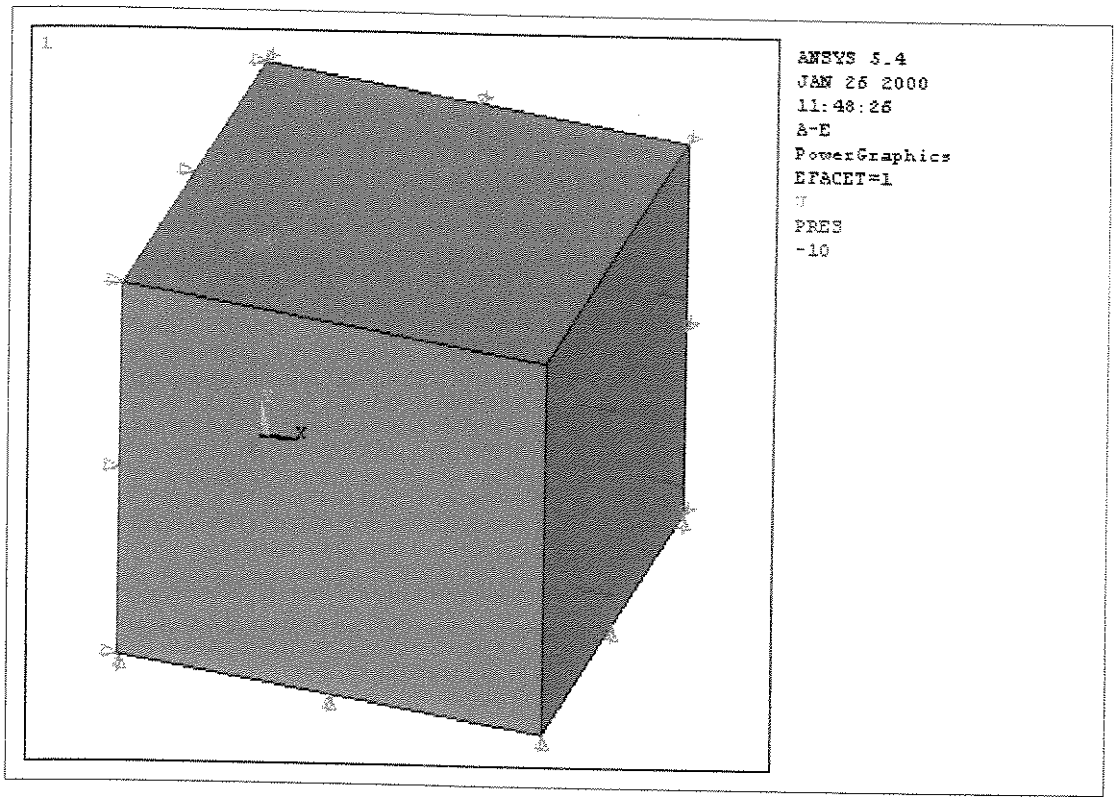


Figura 10. Modelo para o primeiro problema

Para esse problema o material isotrópico recebeu módulo de Young no valor de 1000 e coeficiente de Poisson de 0,3. As unidades são adequadas ao problema e todas compatíveis, de modo que não interfiram na verificação das respostas.

O segundo problema consiste numa placa espessa com um furo no centro submetida a tensão de tração uniaxial ao longo de um dos eixos transversais ao furo. Esse problema foi modelado utilizando simetria de um quarto da placa com restrição total em z da face correspondente ao plano XY para $z = 0$ e tensão de tração aplicada no valor de 1000. O modelo gerado assim como as condições de contorno aplicadas podem ser vistos na figura 11.

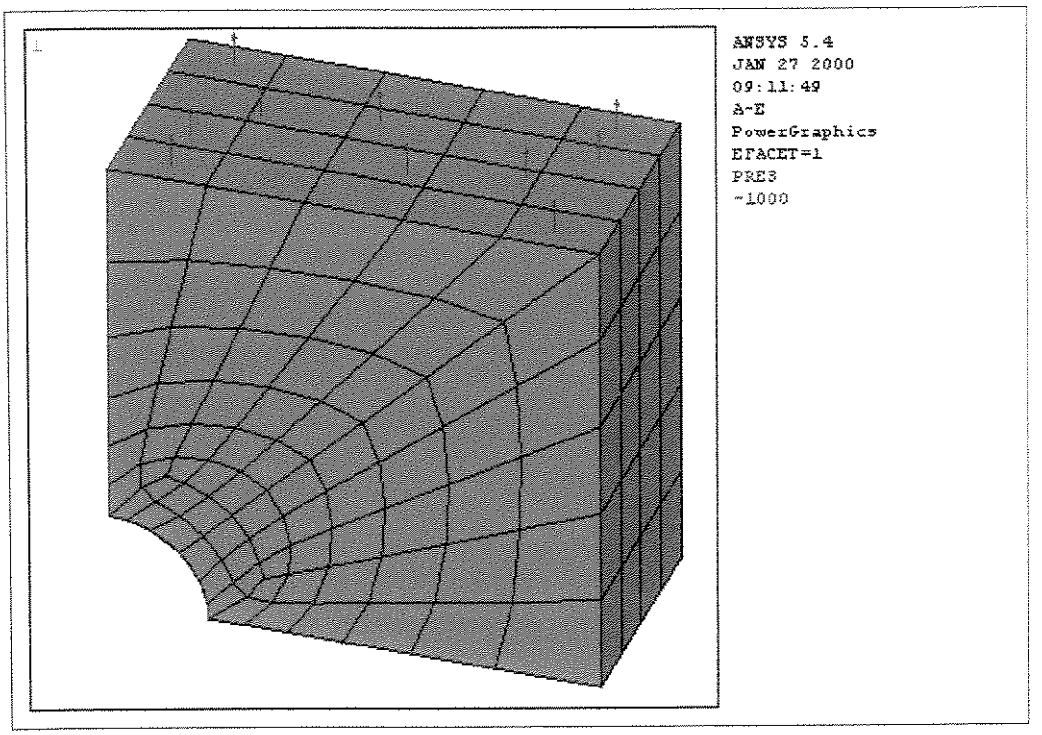


Figura 11. Modelo para o segundo problema

O modelo de material utilizado também foi isotrópico, com módulo de Young no valor de 10^{11} e coeficiente de Poisson de 0,3. Todas as unidades podem ser consideradas adequadas ao problema.

O terceiro problema é proposto por *Kane* (1994) e apresenta a geometria de um quarto de anel de paredes espessas carregado internamente com pressão constante e uniforme 50 unidades. A malha de discretização da geometria seguiu aquela que foi utilizada pelo autor para a solução do problema. As condições de contorno de deslocamento impostas seguiram as propostas de simetria e estado plano de deformação, levando ao modelo que pode ser visto na figura 12.

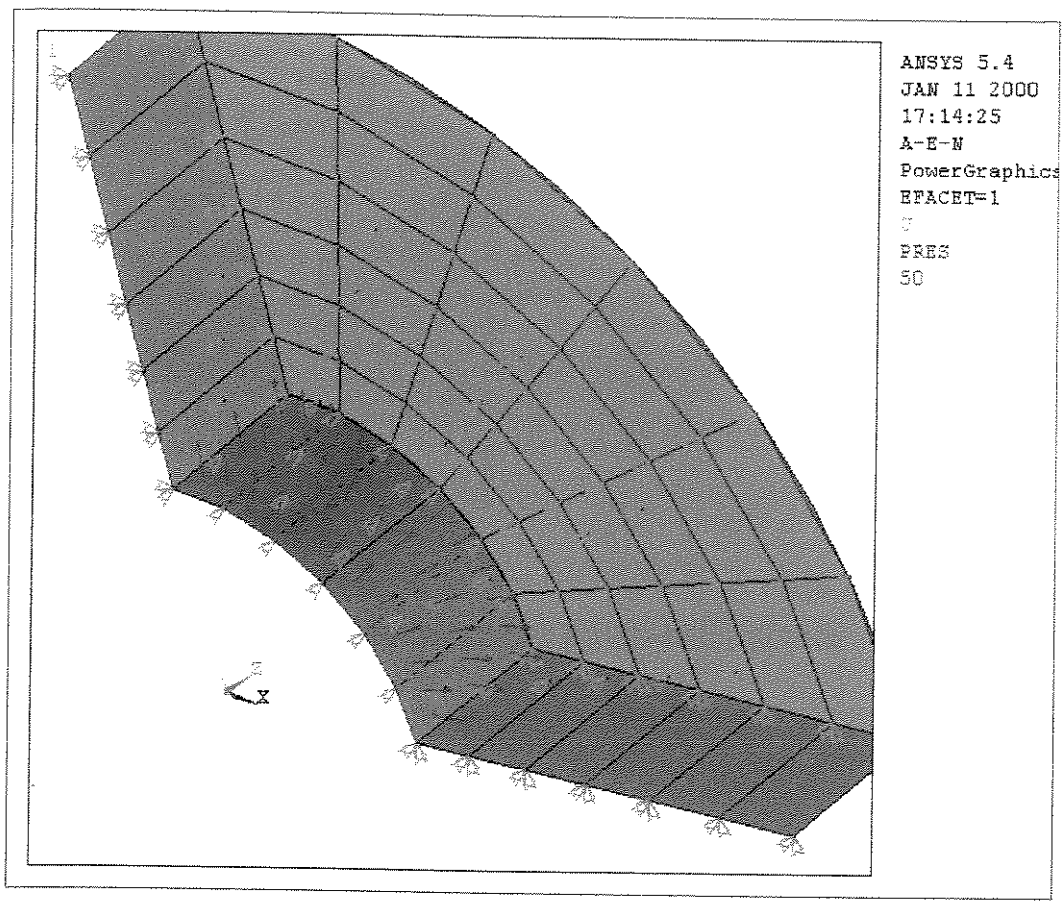


Figura 12. Modelo para o terceiro problema.

Nesse problema, foi utilizado o módulo de Young proposto de 2600 e coeficiente de Poisson de 0,3. Novamente, as unidades são compatíveis de modo que não interfiram na verificação das respostas.

Para o problema proposto por *Kane* (1994) existem respostas que foram obtidas pelo próprio autor, servindo de parâmetro para comparação dos resultados numéricos oferecidos pelo programa E-Con3D.

Finalmente, o quarto problema proposto tem por finalidade demonstrar as capacidades de aplicação do programa a problemas de bioengenharia envolvendo análises mecânicas em ossos. Trata-se de um modelo geométrico de um fêmur, conseguido junto ao *Laboratorio di Tecnologia Medica* do *Intituti Ortopedici Rizzoli*, que é oferecido como

parte de uma proposta de padronização dos modelos ósseos utilizados em pesquisas ligadas à bioengenharia.

O modelo foi entregue pelo laboratório mencionado em formato Ansys®, com todas as áreas e volumes modelados geometricamente. A partir desse modelo foi desenvolvida uma geometria simplificada e aplicada uma malha de elementos finitos do tipo casca de oito nós (*SHELL 93*). Esse elemento foi utilizado pois permite definir uma malha de elementos finitos apenas sobre as áreas do modelo, de forma equivalente a uma malha de elementos de contorno. Além disso, sua construção nodal é adequada ao elemento implementado no programa E-Con3D, ou seja, quadrilateral de oito nós.

Uma vez desenvolvido o modelo discretizado, utilizando o gerador de malha automático do Ansys®, foi feita a tradução da base de dados para arquivos texto no formato adequado para utilização pelo programa E-Con3D. Essa tradução foi feita utilizando duas funções programadas em linguagem interpretada do Ansys®, a partir de comandos disponíveis no módulo *prep7*, responsável pelo pré-processamento. As listagens dessas funções podem ser encontradas no anexo 2.

Uma das funções é responsável por exportar as coordenadas dos nós, chamada *OUTNO.inp*, gerando um arquivo chamado *CoordNos.txt*. A outra função realiza a exportação dos elementos e sua conectividade, *OUTELEM.inp*, gerando o arquivo *NOEL.txt*.

Para o modelo do fêmur foi utilizada uma condição de contorno de restrição de deslocamento nas três direções na base plana inferior e uma condição de contorno de pressão constante e uniforme aplicada sobre dois elementos localizados no topo da geometria semi-esférica da cabeça do fêmur. Essa condição de contorno de pressão procurou simular o efeito de um carregamento de um peso de 80 kg atuando diretamente sobre o osso, como se uma pessoa desse peso estivesse apoiada sobre uma perna apenas.

Como o modelo de material implementado no programa é isotrópico elástico, foi necessário utilizar simplificações e adotar propriedades equivalentes. *Lotz et al.* (1991) propõe uma equação para estimar o módulo de Young em material ósseo de acordo com a densidade mineral equivalente (*QCT*), medida por tomografia computadorizada.

$$E = 0{,}7 \cdot (QCT)^{1{,}2}$$

Segundo essa equação, se a densidade mineral equivalente for dada em g/100cm^3, o módulo de elasticidade (E) será dado em MPa.

Para essa aplicação, uma densidade mineral equivalente média foi determinada a partir de dados fornecidos por *Lotz et al.* (1991) em seu artigo, levando a um valor de módulo de elasticidade de 276 MPa. O coeficiente de Poisson utilizado foi 0,35.

O modelo desenvolvido para essa análise pode ser visto na figura 13.

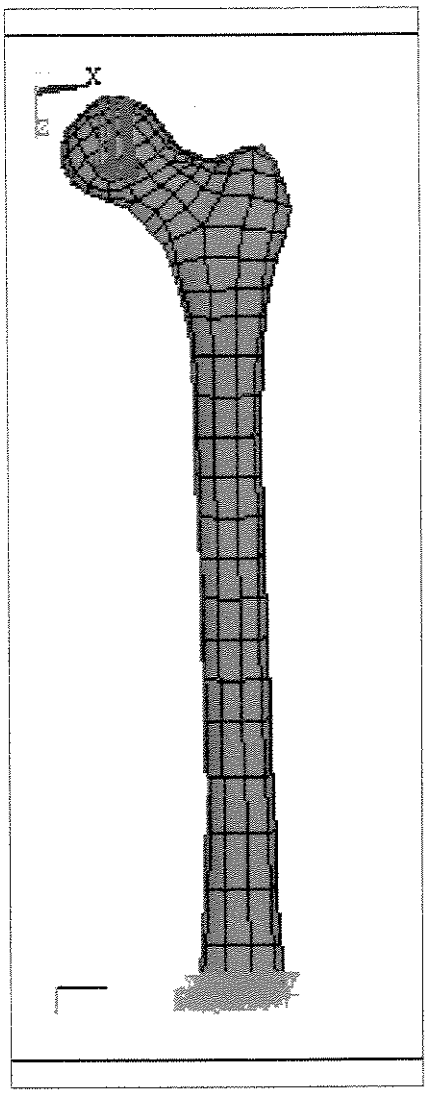


Figura 13. Modelo para o quarto problema.

4.1 **Exemplo 1 – cubo atuando como barra sob tensão de tração**

Uma vez realizada a análise numérica com o programa E-Con3D, um dos resultados que pode ser mostrado é o mapa de deslocamentos dos nós da geometria, mostrando a distensão do cubo ao longo do eixo de solicitação. Esse deslocamento pode ser calculado analiticamente, a partir da teoria de resistência dos materiais, considerando o modelo

como uma barra engastada em uma extremidade e tracionada na oposta. O resultado foi calculado para as condições de contorno aplicadas utilizando a equação 27.

$$u = \frac{E}{A} \cdot P \tag{27}$$

sendo u o deslocamento, E o módulo de Young, A a área da seção transversal da barra e P a carga constante uniforme aplicada. O resultado desse cálculo é um deslocamento de 0,01. A figura 14 mostra o resultado obtido com o E-Con3D.

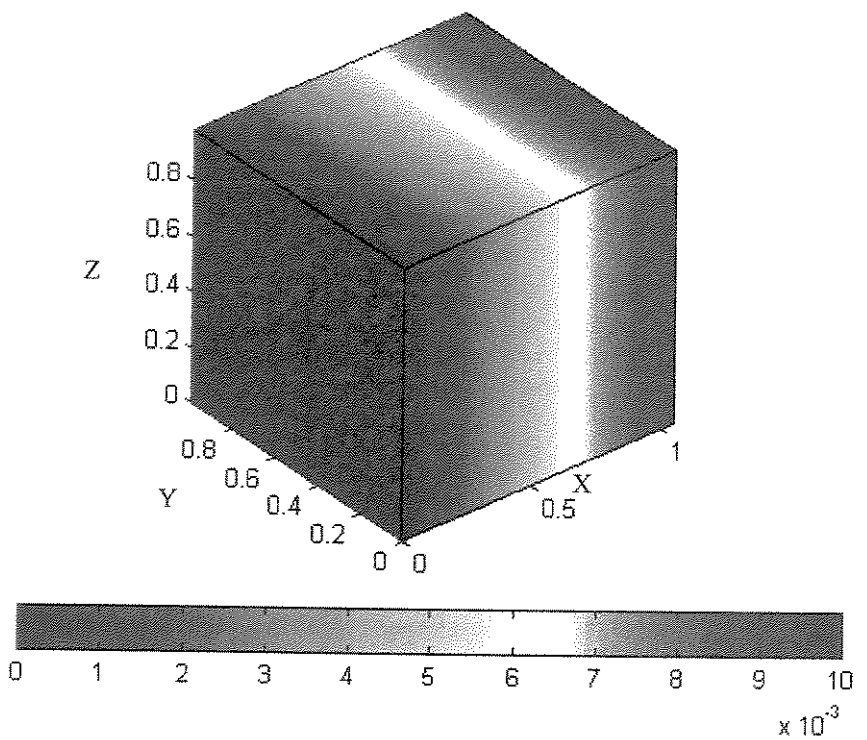


Figura 14. Mapa de deslocamentos dos nós na coordenada global X.

Além disso, é possível obter o mapa das tensões na superfície na direção global X, também chamada de tensão direta sigma XX, como mostra a figura 15.

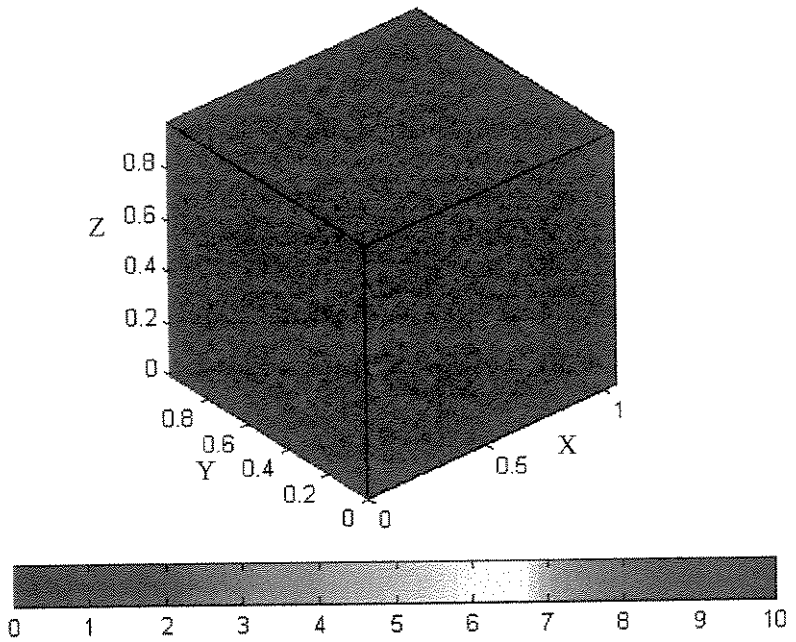


Figura 15. Mapa de tensões sigma XX.

O mapa de deslocamentos pode ser comparado com similares obtidos a partir de análises feitas com programas comerciais, tais como Ansys® e Beasy®. No caso do Ansys® o mapa de cores pode ser visto na figura 16, enquanto o mesmo mapa do Beasy® pode ser visto na figura 17.

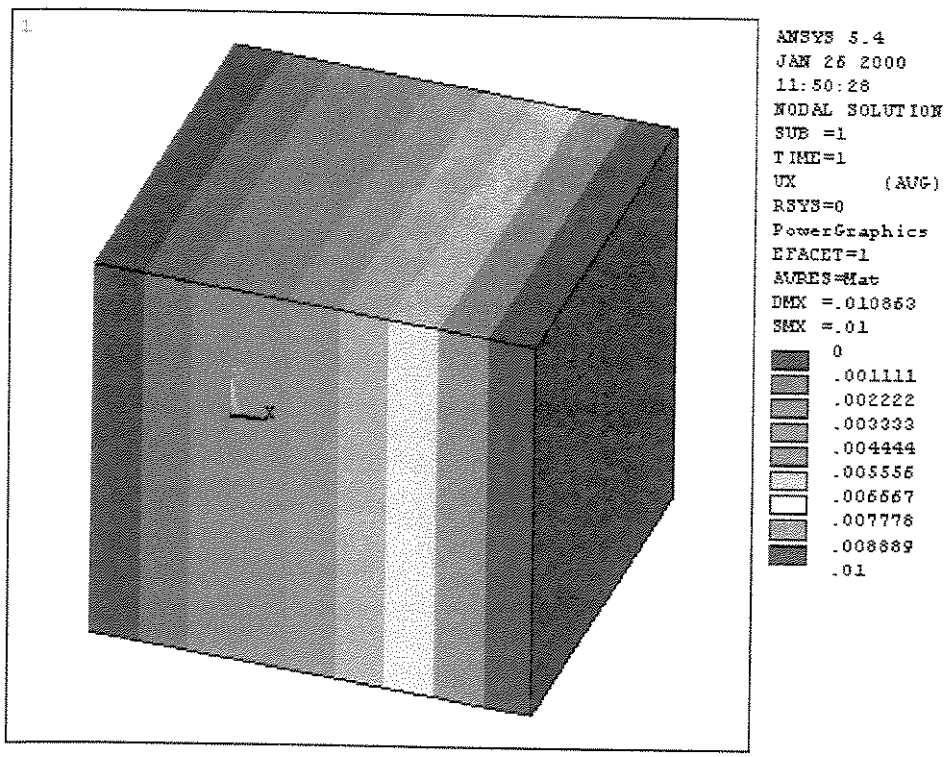


Figura 16. Mapa de deslocamentos em X obtido no Ansys®.

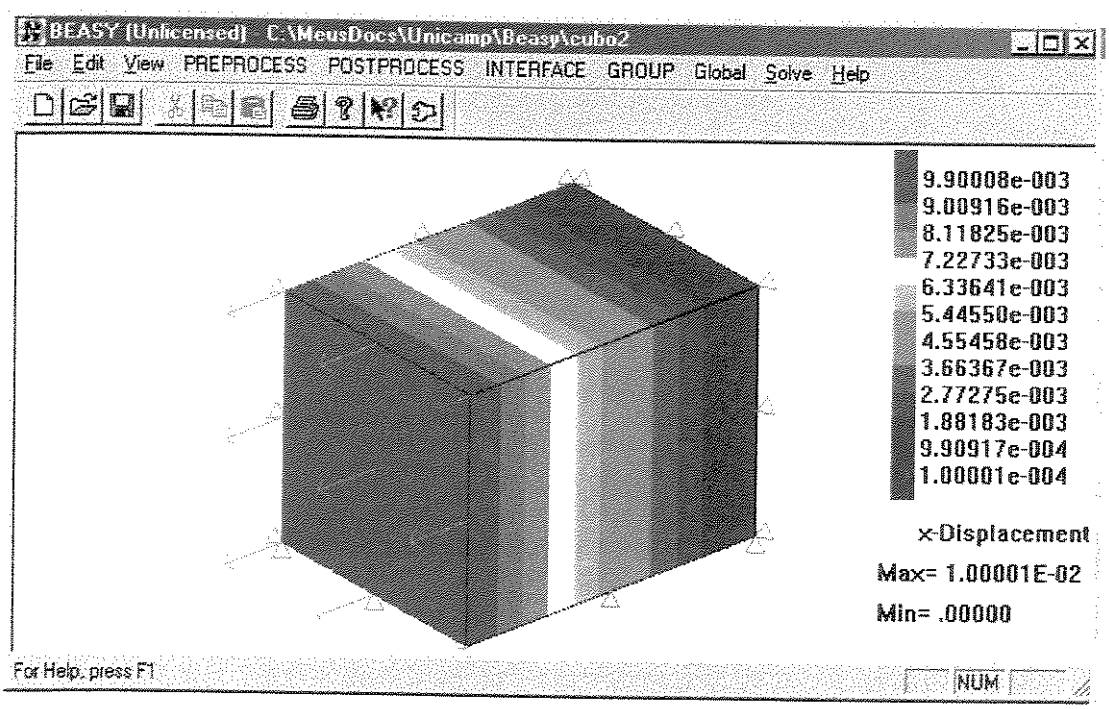


Figura 17. Mapa de deslocamentos em X obtido no Beasy®.

Como se pode notar, os mapas de cores mostram distribuições bastante equivalentes e as escalas de valores mostram resultados muito semelhantes, o que certifica o correto funcionamento do programa E-Con3D.

4.2 Exemplo 2 – placa com furo central tracionada uniaxialmente

O resultado de interesse nesse tipo de problema é o relacionado à concentração de tensão nas bordas do furo colocado no centro da placa. Segundo *Hertzberg* (1996), Inglis formulou analiticamente a concentração de tensão para um furo elíptico em uma placa infinita. No caso de se ter os dois eixos da elipse iguais, o que significa um furo circular, a equação de Inglis mostra uma concentração de tensão mínima de três vezes a tensão de tração uniaxial aplicada nas bordas da placa.

A tensão de tração aplicada para o modelo vale 1000, portanto, o valor de tensão concentrado na borda do furo diretamente tracionada é esperado em torno de 3000. O mapa de cores que mostra a tensão procurada é o de tensão nodal YY, que pode ser visto na figura 18.

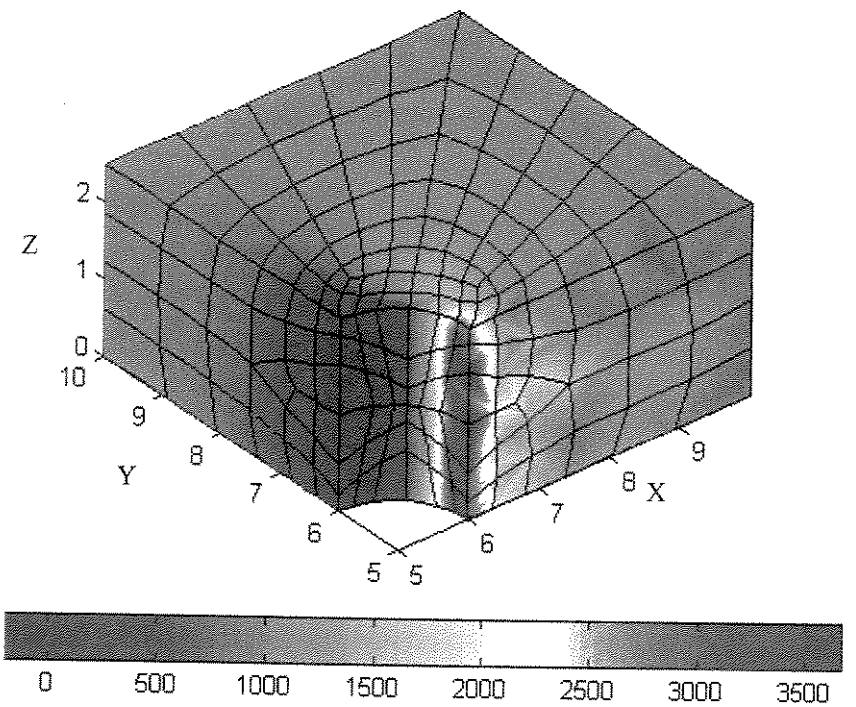


Figura 18. Mapa de tensões nodais sigma YY.

No mapa da figura 18 é possível localizar a concentração de tensão e observar seu valor aproximado de três vezes o valor da tensão de tração aplicada.

4.3 **Exemplo 3 – anel com carga interna em estado plano de deformação**

Os resultados da análise do problema proposto por *Kane* (1994) podem ser agrupados em dois tipos, a fim de possibilitar comparação com os resultados mostrados pelo autor. Um primeiro tipo de resultado refere-se ao deslocamento de nós localizados na superfície do arco interno do anel. Já o segundo resultado refere-se à tensão radial nesses nós, bem como as tensões normais aos nós cuja restrição de deslocamento garante a manutenção do estado plano de deformação.

O resultado para deslocamentos pode ser visto na figura 19.

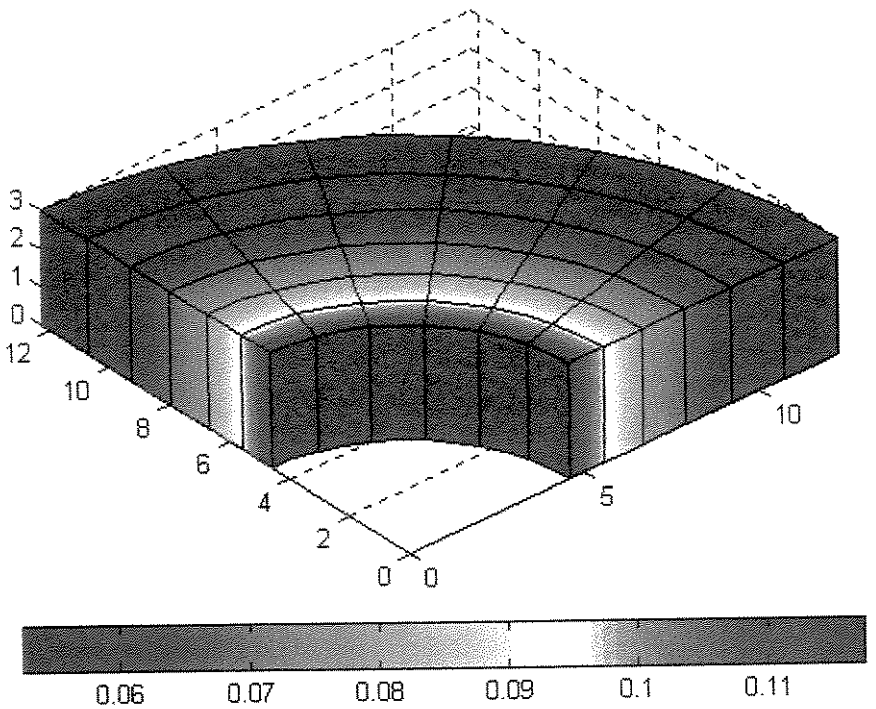


Figura 19. Mapa de deslocamentos radiais

Os resultados mostrados por *Kane* (1994) em comparação com aqueles obtidos com o E-Con3D podem ser vistos na tabela 2.

Tabela 2. Valores para exemplo Kane:

			Kane			E-Con3D			
R	theta	z	ur	sigr	sigz	ur	sigr	sigz	erro % ur
4	0	0	0,1175	-50	3,75	0,1175	-49,38	3,47	0,000
4	15	0	0,1175	-50	3,75	0,1174	-48,68	4,26	0,085
4	30	0	0,1175	-50	3,75	0,1174	-48,84	3,99	0,085
4	45	0	0,1175	-50	3,75	0,1175	-48,84	3,97	0,000
4	45	0,5	0,1175	-50	3,75	0,1175	-50,00	3,75	0,000
4	45	1	0,1175	-50	3,75	0,1175	-49,11	3,36	0,000

Observando os valores numéricos de deslocamento dados pelo programa E-Con3D para a coordenada $R = 4$, *theta* $= 0$ e $z = 0$, por exemplo, encontra-se $ur = 0{,}1175$, valor idêntico àquele mostrado como resposta na tabela 2.

Um mapa de cores de tensão sigma XX na superfície do modelo também pode ser utilizado para verificar a concentração de tensão nas bordas do furo. Esse mapa pode ser visto na figura 20.

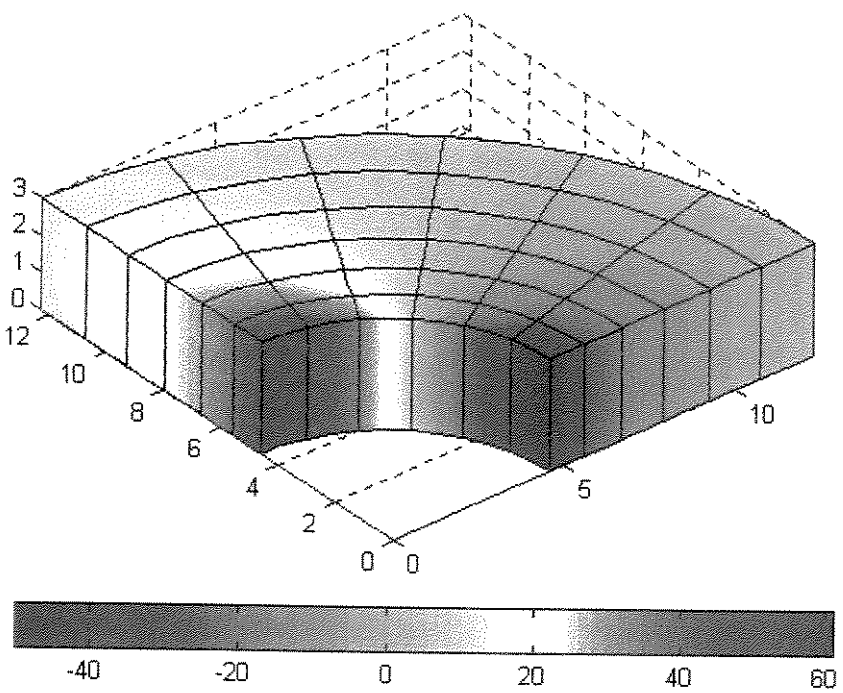


Figura 20. Mapa de tensões sigma XX na superfície.

No mapa mostrado na figura 20 é possível notar tensão de tração nas regiões vermelhas e tensão de compressão nas azuis, qualitativamente condizente com o esperado, uma vez que o mapa retrata apenas as tensões na direção x.

4.4 Exemplo 4 – modelo de um fêmur sob carga uniaxial

A análise de um modelo de fêmur com o carregamento proposto é meramente ilustrativa, permitindo evidenciar as capacidade do programa E-Con3D em analisar geometrias complexas através da interação com um programa comercial como o Ansys®. Apenas para que se possa citar alguns números de referência, o modelo avaliado apresenta 1259 nós dispostos em 419 elementos de contorno. Para cada elemento é necessário descrever os nós que o compõe e para cada nó é necessário entrar com suas três coordenadas globais.

Uma vez colocado o modelo no pré-processador do Ansys®, o gerador de malhas desse aplicativo pode gerar a malha de elementos de contorno e exportá-la para o programa E-Con3D. Após realizar a análise, um dos resultados que pode ser visto é o de tensão na direção global *z*, que é mostrado na figura 21.

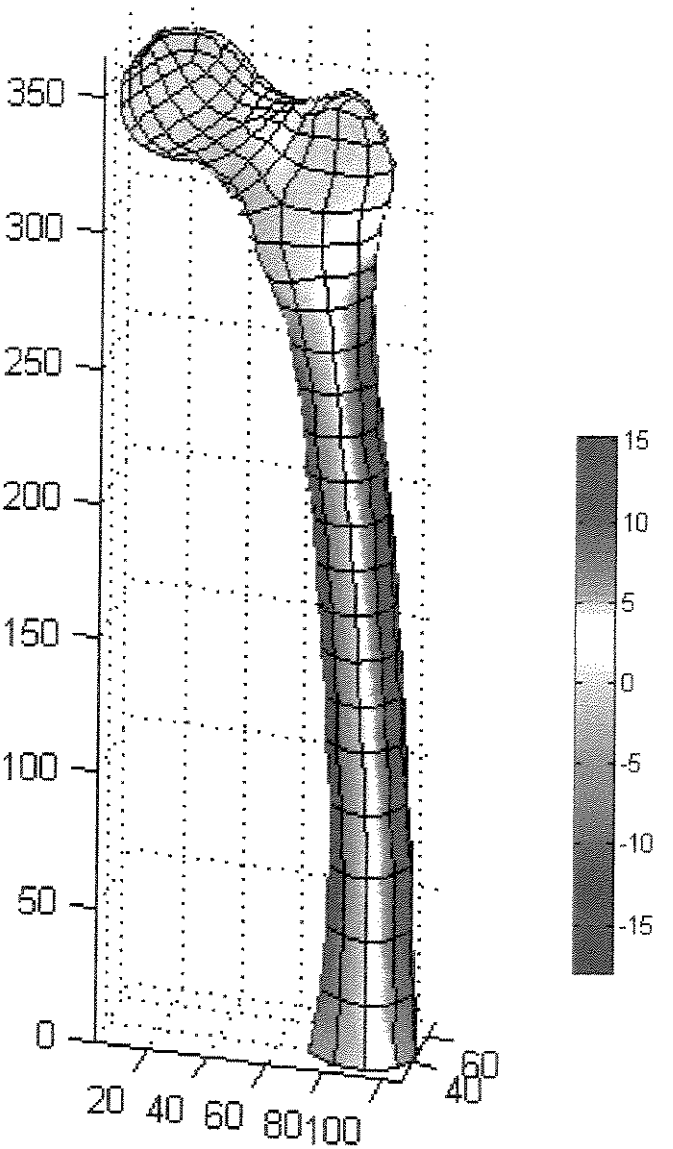


Figura 21. Mapa de tensão sigma ZZ.

É possível notar que o mapa de cores está sendo mostrado sobre a geometria deformada do fêmur. Para evidenciar os deslocamentos ao longo da coordenada *z* pode se pedir um mapa de cores de deslocamentos nessa coordenada, o que pode ser visto na figura 22.

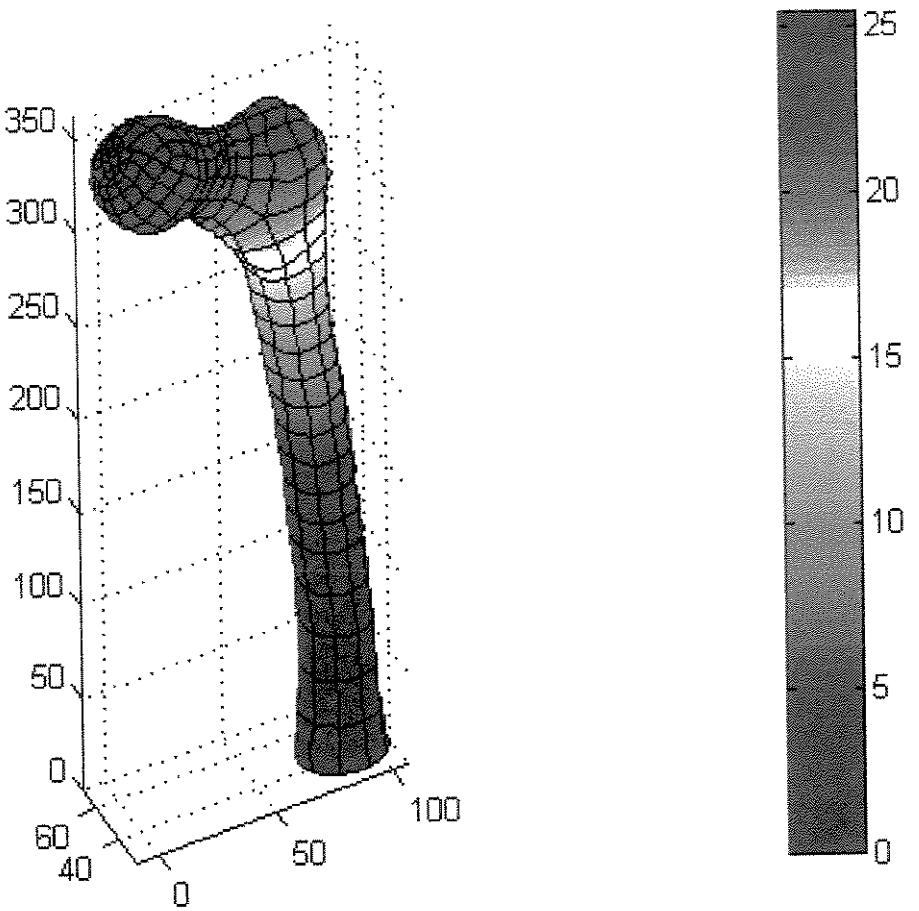


Figura 22. Mapa de deslocamentos em z

Além desses resultados o programa também é capaz de mostrar as tensões principais máximas, usadas comumente em engenharia como critério para definir a direção do crescimento de trincas. O mapa de cores de tensões principais máximas pode ser visto na figura 23.

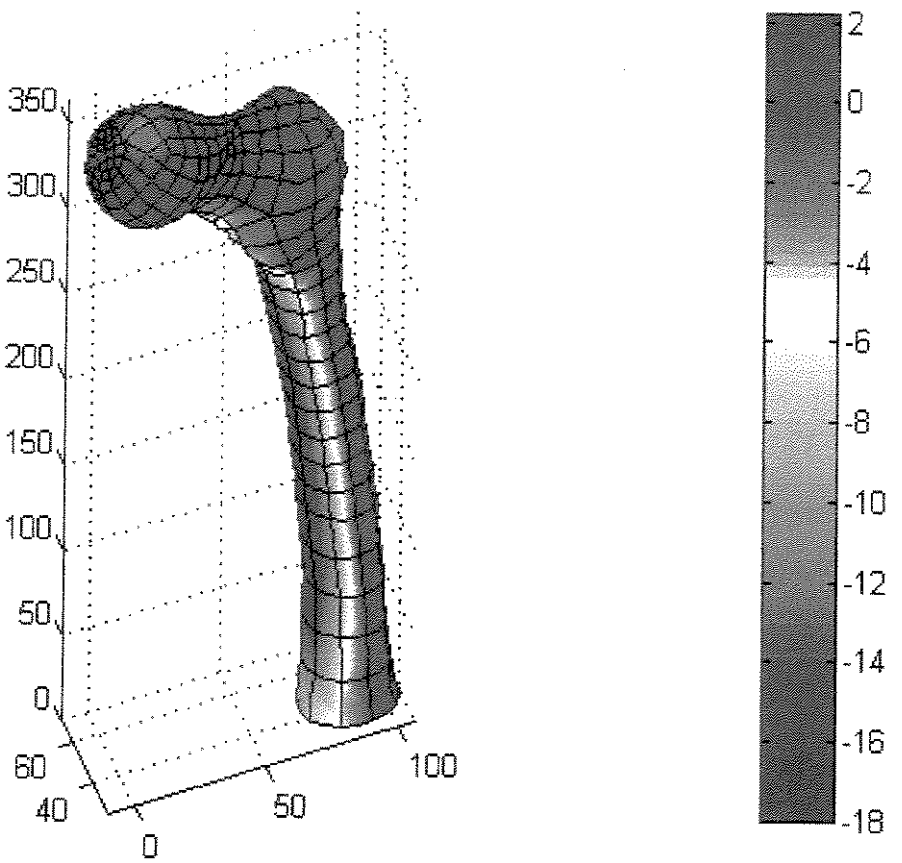


Figura 23. Mapa de tensões principais máximas

Finalmente, um último resultado que pode ser mostrado é o de tensão equivalente de von Mises na figura 24.

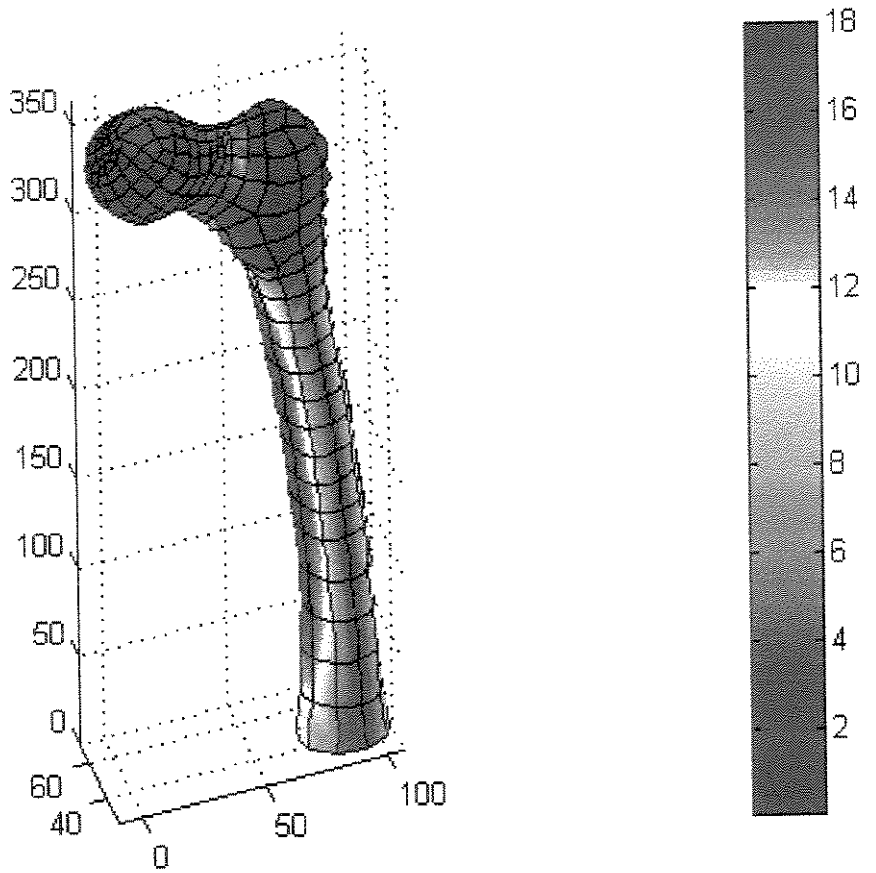


Figura 24. Mapa de tensão de von Mises

Para essa análise do fêmur ficou bastante evidente a vantagem das técnicas de otimização implementadas no código E-Con3D. Em comparação com a maneira tradicional de construção das matrizes *H* e *G* globais, a técnica *RISP* oferece uma redução no tempo de processamento de aproximadamente três quartos. Para a análise do fêmur, um teste feito com uma versão inicial do programa E-Con3D, antes da implementação da técnica *RISP*, levou um tempo de aproximadamente quatro vezes àquele quando realizado com a versão final, que tem a técnica *RISP* implementada. Além disso, a técnica de reutilização das matrizes *H* e *G* permite uma redução ainda maior no tempo de processo.

CAPÍTULO 5

CONCLUSÕES

De acordo com os resultados mostrados no capítulo 4 pode-se concluir que o programa E-Con3D funciona adequadamente para resolver problemas de elasticidade em três dimensões utilizando modelo de material isotrópico linear e homogêneo. Além disso, ficou demonstrada a capacidade de expansão do programa através da capacidade de implementação de novos tipos de elementos, novas soluções fundamentais e novas metodologias de cálculo, mais adequadas para resolver problemas específicos de bioengenharia, tais como o remodelamento ósseo.

Os exemplos 1 e 2, no capítulo 4, mostraram a validade das formulações utilizadas pelo programa ao reproduzir resultados obtidos em programas de uso comercial, Ansys® e Beasy®, e na literatura. Além disso, foi possível notar a facilidade na visualização dos resultados, introduzida pelas funções de pós-processamento, como uma das características desejáveis do programa E-Con3D.

No exemplo 3 foi possível verificar novamente a validade das soluções calculadas pelo programa E-Con3D, mas com dados da literatura oferecidos por *Kane* (1994).

Finalmente, no exemplo 4 foi possível observar a capacidade de interação com o programa comercial Ansys® para geração das bases de dados para o programa E-Con3D,

tornando-o capacitado a lidar com geometrias bastante complexas. Embora o modelo geométrico do fêmur tenha sido fornecido por terceiros já no formato Ansys®, todo o desenvolvimento da malha de elementos de contorno foi realizado como parte desse trabalho, assim como as funções de exportação do Ansys® para o E-Con3D.

Dessa forma, os resultados deixam evidente que o programa E-Con3D funciona de maneira correta, no que diz respeito a sua implementação numérica computacional, cumprindo o objetivo de ser um código aberto e funcional. Além disso, a estrutura de funções mostrou-se suficiente para cumprir as demais exigências de pós-processamento, a fim de tornar mais fácil a interpretação dos resultados, bem como a interação com programas comerciais que possibilitassem um pré-processamento mais eficiente e apto a lidar com geometrias complexas.

CAPÍTULO 6

PROPOSTAS PARA TRABALHOS FUTUROS

Uma vez implementado o programa base, funcional, capaz de analisar geometrias complexas aplicadas à bioengenharia, é possível passar a uma nova fase no desenvolvimento do programa E-Con3D. Inicialmente, uma proposta é a de desenvolvimento de novos modelos de materiais, mais adaptados e representativos do comportamento observado para o material ósseo real. Além disso, a implementação de novas técnicas de cálculo, acrescentando ao núcleo do programa a capacidade de lidar com fenômenos típicos de materiais ósseos, tais como o remodelamento, é uma proposta. Uma última proposta discutida refere-se à implementação de um módulo de análise de mecânica da fratura, possibilitando realizar cálculos de fator de intensidade de tensão em geometrias complexas e mesmo propagação de trincas ao longo da estrutura.

Outra linha de propostas refere-se à otimização do programa, tornando-o mais rápido e utilizável. Essa proposta envolve a tradução do código, gerado em linguagem MatLab®, para uma outra linguagem como C^{++}, que ofereça a possibilidade de compilação, aumentando a velocidade de execução do programa e sua independência em relação a aplicativos. Além disso, a adição de novas técnicas de programação otimizada, assim como a implementação completa do algoritmo *RISP* são propostas para trabalhos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

Aliabadi, M. H. & Sollero, P., 1997, Crack Growth Analysis in Homogeneous Orthotropic Laminates, Department of Engineering, University of London, Queen Mary and Westfield College, Mile End, London, Universidade Estadual de Campinas, Brazil, Composites Science and Technology, 58, pp 1697 – 1703.

ASTM Standards E 399-72. Standard Method of Test for Plane-Strain Fracture Toughness of Metallic Material. American Society for Testing and Materials, Philadelphia, PA.

Bonfield, W. & Grynpas, M. D. & Young, R. J., 1978, Crack Velocity and the Fracture of Bone, Department of Materials, Queen Mary College, London, U.K., Journal of Biomechanics, vol. 11, pp. 473 - 479.

Domingues, J., 1993, Boundary Elements in Dynamics, Escuela Superior de Ingenieros Industriales, Universidad de Sevilla, Computational Mechanics, Southampton, Boston.

Etchebehere, E. C. S. C., 1998, Recentes Avanços em Densitometria Óssea, Grupo MN&D – Medicina Nuclear, Diagnóstico e Terapia, serviço de Medicina Nuclear do Departamento de Radiologia do Hospital das Clínicas da Universidade Estadual de Campinas, Jornal do Médico, pp. 7.

Gharpuray, V. M. & Keer, L. M. & Lewis, J. L., 1990, Cracks Emanating from Circular Voids or Elastic Inclusions in PMMA Near a Bone-Implant Interface, Department of Civil Engineering, Technological Institute, Northwestern University, Evanston, IL, Department of Orthopaedic Surgery, University of Minnesota, Minn., MN, Transactions of the ASME, vol. 112, pp. 22 - 28.

Gharpuray, V. M. & Keer, L. M. & Lewis, J. L., 1992, Cracks Emanating from Slipping Inclusions Near a Bone-Implant Interface, Department of Bioengineering, Clemson

University, Clemson, SC, Department of Civil Engineering, Technological Institute, Northwestern University, Evanston, IL, Department of Orthopaedic Surgery, University of Minnesota, Minn., MN, Transactions of the ASME, vol. 114, pp. 178 - 182.

Hertzberg, R. W., 1996, Deformation and Fracture Mechanics of Engineering Materials, John Wiley & Sons, Fourth Edition.

Juricic, D. & Wnuk, M. P., 1972, Computer Simulation of Fracture Spreading in Visco-Elastic Solid, International Journal of Fracture Mechanics, vol. 8, n. 3, pp. 257 – 265.

Kane, J. H., 1994, Boundary Element Analysis in Engineering Continuum Mechanics, Mechanical & Aeronautical Engineering, Clarkson University, Prentice Hall.

Knauss, W. G., 1970, Delayed Failure – The Griffith Problem for Liearly Viscoelastic Materials, California Institute of Technology, Pasadena, California, International Journal of Fracture Mechanics, vol. 6, n. 1, pp. 7 – 20.

Köberle, G., 1974, Estudos Físicos e Biológicos em Tecido Ósseo, Universidade de São Paulo, Ribeirão Preto, São Paulo, Tese para Livre-Docência.

Lee, S. S., 1996, Time-dependent Boundary Element Analysis for na Interface Crack in a Two-dimensinal Unidirectional Viscoelastic Model Composite, Reactor Mechanical Engineering Department, Korea Atomic Energy Research Institute, Taejon, South Korea, International Journal of Fracture, 77, pp. 15 – 28.

Lotz, J. C. Cheal, E. J. & Hayes, W. C., 1991, Fracture Prediction for the Proximal Femur Using Finite Element Model: Part I - Linear Analysis, Orthopaedic Biomechanics Laboratory, Department of Orthopaedic Surgery, Charles A. Dana Research Institute, Beth Israel Hospital and Harvard Medical School, Boston, Journal of Biomechanical Engineering, vol. 113, pp. 353 - 365.

Melvin, J. W., 1993, Fracture Mechanics of Bone, Biomedical Science Department, NAO Research and Development Center, General Motors Corporation, Warren, Journal of Biomechanical Engineering, vol. 115, pp. 549 – 554.

Mueller, H. K., 1971, Stress-Intensity Factor and Crack Opening for a Linearly Viscoelastic Strip with a Slowly Propagating Central Crack, California Institute of Technology, Pasadena, California, International Journal of Fracture Mechanics, vol. 7, n. 2, pp. 129 - 141.

Natali, A. N. & Meroi, E. A., 1989, A review of the biomechanical properties of bone as a material, Università di Padova, Istituto di Scienza e Tecnica delle Costruzioni, Padova, Italy, Journal of Biomed. Eng., vol. 11, pp. 266 - 276.

Sadegh, A. M. Luo, G. M. & Cowin, S. C., 1993, Bone Ingrowth: an Application of the Boundary Element Method to Bone Remodeling at the Implant Interface, Department of Mechanical Engineering, City College of the City University of New York, Journal of Biomechanics, vol. 26, pp. 167 - 182.

Weichen, S. & Sollero, P., 1998, A Fracture Mechanics Study of Bone Based on Adaptive Elasticity Theory, Department of Computational Mechanics, FEM, University of Campinas.

Williams, J. G., 1972, Visco-Elastic and Thermal Effects on Crack Growth in PMMA, Mechanical Engineering Department, Imperial College, London, International Journal of Fracture Mechanics, vol. 8, n. 4, pp. 393 – 401.

Wolfe, L. A., 1993, Stress analysis of endosseous implants using the Boundary Integral Equation (BIE) Method, Hard Tissue Research Unit, Department of Anatomy and Embryology, University College London, UK, Journal of Biomed. Eng., vol. 15, pp. 319 - 323.

Workshop de Biomateriais, I, 1999, Campinas, Resumos dos Trabalhos Apresentados, São Paulo, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas.

Wright, T. M. & Hayes, W. C., 1977, Fracture Mechanics Parameters for Compact Bone – Effects of Density and Specimen Thickness, Department of Materials Science and Engineering, Department of Mechanical Engineering, Stanford University, Stanford, U.S.A., Journal of Biomechanics, vol. 10, pp. 419 – 430.

Zysset, P. K. et al., 1999, Elastic modulus and hardness of cortical and trabecular bone lamellae measured by nanoindentation in the human femur, Orthopaedic Research Laboratories, University of Michigan, U.S.A, Journal of Biomechanics, vol. 32, pp. 1005 – 1012.

Manuais do MatLab® 5.0

Manuais do Ansys® 5.2

ANEXO 1

CoordNos.txt

X	Y	Z	
.000000000	1.000000000	.000000000	nó 1
.000000000	.000000000	.000000000	nó 2
.000000000	.500000000	.000000000	...
1.000000000	1.000000000	.000000000	nó *n*

NOEL.txt

nó 1	nó 2	nó 3	nó 4	nó5	nó 6	nó 7	nó 8	
2.	3.	1.	5.	4.	7.	6.	8.	elemento 1
9.	11.	10.	13.	12.	15.	14.	16.	elemento 2
9.	17.	2.	8.	6.	18.	10.	11.	...
14.	15.	12.	19.	4.	5.	1.	20.	elemento *e*

CDCDesl.txt

		valores de deslocamento por nó							
nó	direção	nó 1	nó 2	nó 3	nó 4	nó 5	nó 6	nó 7	nó 8
1	3	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00
5	1	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00
3	2	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00	.0E+00

CDCTens.txt

quantidade de elementos sujeitos à mesma CDC de tensão	P	PXX	PYY	PZZ	PXY	PXZ	PYZ	números dos elementos sujeitos à CDC
1	0.	10.	0.	0.	0.	0.	0.	6

Props.txt

4	máxima dimensão do modelo
1000.000000	módulo de Young
0.300000	coeficiente de Poisson
8	quantidade de pontos de Gauss

ANEXO 2

Programa **OUTELEM.inp**

```
/prep7
*dim,elem,array,elmiqr(0,14),8
*do,i,1,8
*vget,elem(1,i),elem,1,node,i
*enddo

/output,NOEL,txt
*vwrite,elem(1,1),elem(1,5),elem(1,2),elem(1,6),elem(1,3),elem(1,7),elem(
1,4),elem(1,8)
(f8.0,f8.0,f8.0,f8.0,f8.0,f8.0,f8.0,f8.0)
!  (i6,4i8)   some systems may need (f6.0,4f8.0)
!*vwrite
!('END OF ELEMENTS')
/output
!*list,myfile,txt
finish
! /exit,nosave
```

Programa **OUTNO.inp**

```
/prep7
*dim,nodes,array,ndinqr(0,14),3
*vget,nodes(1,1),node,1,loc,x
*vget,nodes(1,2),node,1,loc,y
*vget,nodes(1,3),node,1,loc,z

/output,CoordNos,txt
*vwrite,nodes(1,1),nodes(1,2),nodes(1,3)
(f16.9,f16.9,f16.9)
!   (i6,3g12.5)  some systems may need (f6.0,3g12.5)
!*vwrite
!('END OF NODES')
/output
*list,myfile,txt
finish
! /exit,nosave
```

ANEXO 3

Listagem do programa **main.m**

```
function main(problema,CaminhoProblema,arqHeG)

%************************************************************
%                        Programa E-CON3D
%************************************************************
% Programa para análise tridimensional elástica de tensões
%         pelo método dos elementos de contorno
%************************************************************
%Sintaxe: main('nome do problema','diretório de localização dos
%          arquivos de dados','caminho e nome do arquivo HeG.mat',
%          0 caso contrário)
%
%Obs: os arquivos de dados CoordNos.txt, NOEL.txt, CDCTens.txt,
%CDCDesl.txt, Props.txt devem estar no mesmo diretório de E-CON3D.

%** INICIALIZAÇÃO DE VARIÁVEIS DE FUNÇÕES POSTERIORES **
NODPT = 0;
NCOND = 0;
%** INICIALIZAÇÃO DO VETOR DE TEMPO **
T0 = clock;
%** CARREGA ARQUIVOS COM DADOS DE ENTRADA (FORMATO .TXT) **
fprintf('\n'); fprintf('Carregando dados para o problema %s\n',problema);
T0Entrada = clock;
fidin = fopen(strcat(CaminhoProblema,'Props.txt'));
fseek(fidin,-1,'eof');
fim_arq = ftell(fidin) + 1;
frewind(fidin);
while ftell(fidin) < fim_arq
  MAXL = str2num(fgets(fidin));
  E = str2num(fgets(fidin));
  NU = str2num(fgets(fidin));
  IGAUS = str2num(fgets(fidin));
end;

%** Entrada de coordenadas dos pontos nodais **
%** Lê o arquivo CoordNos.txt e cria a matriz CoordNos(NNP,3) **
fidin1 = fopen(strcat(CaminhoProblema,'CoordNos.txt'));
nn = 1;
fseek(fidin1,-1,'eof');
fim_arq1 = ftell(fidin1) + 1;
frewind(fidin1);
while ftell(fidin1) < fim_arq1
  CoordNos1 = str2num(fgets(fidin1));
```

```
  CoordNos(nn,:) = CoordNos1;
  nn = nn + 1;
end;
NNP = size(CoordNos,1);          %quantidade de nós do problema
for ii = 1:NNP
  X(ii) = CoordNos(ii,1);
  Y(ii) = CoordNos(ii,2);
  Z(ii) = CoordNos(ii,3);
end;
%Sequência de entrada dos dados:
%X(I), Y(I), Z(I) = vetores contendo as coordenadas dos pontos
%nodais por linhas
%** Entrada da matriz de conectividade nodal dos elementos **
fidin2 = fopen(strcat(CaminhoProblema,'NOEL.txt'));
nn = 1;
fseek(fidin2,-1,'eof');
fim_arq2 = ftell(fidin2) + 1;
frewind(fidin2);
while ftell(fidin2) < fim_arq2
  NOEL1 = str2num(fgets(fidin2));
  NOEL(nn,:) = NOEL1;
  nn = nn + 1;
end;
NEL = size(NOEL,1);              %quantidade de elementos da malha
%Sequência de entrada de dados:
%NOEL(M,N) = matriz com um elemento em cada linha e seus nós de conectividade
%numerados em sentido anti-horário nas colunas

%** Entrada da quantidade dados para condição de contorno
%   de tração nodal nula**
NBPTU = 0;
%** Entrada de condições de contorno de deslocamento **
fidin3 = fopen(strcat(CaminhoProblema,'CDCDesl.txt'));
nn = 1;
fseek(fidin3,-1,'eof');
fim_arq3 = ftell(fidin3) + 1;
frewind(fidin3);
while ftell(fidin3) < fim_arq3
  CondContDesl1 = str2num(fgets(fidin3));
  CondContDesl(nn,:) = CondContDesl1;
  nn = nn + 1;
end;
NBCU = size(CondContDesl,1);  %quantidade de condições de
                              %contorno de deslocamentos
for ii = 1:NBCU
  MNEL(ii) = CondContDesl(ii,1);
  MCOND(ii) = CondContDesl(ii,2);
  for jj = 1:8
    UPRES(ii,jj) = CondContDesl(ii,(jj + 2));
  end;
end;
%Sequência da entrada de dados:
%MNEL = número do elemento sob a condição de contorno
%MCOND = direção na qual atua a condição de contorno no
%       elemento (1-x, 2-y, 3-z)
%no1...no8 = valor dos deslocamentos nodais
```

```
%** Entrada de restrições pontuais nas quais as trações são nulas **
if NBPTU ~= 0
  CondContTracNula = [NODPT NCOND;  %-> primeiro nó de tração nula
                      NODPT NCOND;  %-> segundo nó de tração nula
                      NODPT NCOND]; %-> NBPTU nó de tração nula
  for ii = 1:NBPTU
    NODPT(ii) = CondContTracNula(ii,1);
    NCOND(ii) = CondContTracNula(ii,2);
  end;
end;
% Sequência de entrada de dados:
% NODE = número do nó sob a condição de contorno
% NCOND = direção na qual atua a condição de contorno no nó (1-x, 2-y, 3-z)
%** Entrada de condições de contorno de tensão sobre elementos **
fidin4 = fopen(strcat(CaminhoProblema,'CDCTens.txt'));
nn = 1;
fseek(fidin4,-1,'eof');
fim_arq4 = ftell(fidin4) + 1;
frewind(fidin4);
while ftell(fidin4) < fim_arq4
  temp = str2num(fgets(fidin4));
  for ii = 1:size(temp,2)
    CondContTens(nn,ii) = temp(ii);
  end;
  nn = nn + 1;
end;
NBCP = size(CondContTens,1);  %quantidade de condições de contorno de tensão
%Sequência de entrada de dados:
%JNEL = quantidade de elementos sujeitos a condição de contorno dada na linha
%P = Pressão normal uniforme sobre um elemento
%PXX, PYY, PZZ, PXY, PXZ, PYZ = condição de contorno de estado de tensão
%                               sobre um elemento
%MEL(NE) = número(s) do(s) elemento(s) sob a condição de contorno descrita
TempoEntrada = etime(clock,T0Entrada);
fprintf('Tempo de entrada de dados = %1.3f\n\n',TempoEntrada);
%** CALCULA E ARMAZENA COORDENADAS E PESOS DOS PONTOS DE GAUSS **
fprintf('1. - Calculando ',IGAUS,' pontos de Gauss\n');
T0InteGauss = clock;
[XG,CG] = INTEGAUS(IGAUS);
TempoInteGauss = etime(clock,T0InteGauss);
fprintf('ok - Pontos de Gauss calculados\n');
fprintf('Tempo de cálculo dos pontos de Gauss = %1.3f\n\n',TempoInteGauss);

%** AVALIA AS CONDIÇÕES DE CONTORNO **
fprintf('2. - Avaliando Condições de Contorno\n');
T0Bcs = clock;
[NELM,JT,jt_local,ICOND,TPRES,NBCT] = BCS(NBCU,NBCP,IGAUS,NOEL,X,Y,Z,
CondContTens);
TempoBcs = etime(clock,T0Bcs);
fprintf('ok - Condições de Contorno avaliadas\n');
fprintf('Tempo de avaliação das condições de contorno = %1.3f\n\n',TempoBcs);

%** FORMA A MATRIZ E SEGUNDO MEMBRO **
fprintf('3. - Formando matriz [A] e vetor [b]\n');
T0FrMtrx = clock;
if arqHeG ~= 0
  fprintf('3.1 - Lendo matrizes [H] e [G] pré-formadas\n');
```

```
  [A,BT,NLIBD] = FRMTRX2(arqHeG,NNP,NEL,NBCU,NBPTU,NBCT,IGAUS,MAXL,NU,E,
NOEL,NELM,MNEL,NODPT,UPRES,TPRES,ICOND,MCOND,NCOND,JT,jt_local,CG,XG,X,Y,
Z);
else
  [A,BT,NLIBD] = FRMTRX1(NNP,NEL,NBCU,NBPTU,NBCT,IGAUS,MAXL,NU,E,NOEL,NELM,
MNEL,NODPT,UPRES,TPRES,ICOND,MCOND,NCOND,JT,jt_local,CG,XG,X,Y,Z);
end;
TempoFrMtrx = etime(clock,T0FrMtrx);
fprintf('ok - Matriz [A] e vetor [b] formados\n');
fprintf('Tempo de formação da matriz [A] e vetor [b] =
%1.3f\n\n',TempoFrMtrx);

%** RESOLVE O SISTEMA MATRICIAL ATRAVÉS DA ELIMINAÇÃO DE GAUSS **
fprintf('4. - Resolvendo sistema por eliminação de Gauss da matriz [A]\n');
T0Gauss_eli = clock;
B = A \ BT; %  [B] = Gauss_eli(A,BT);
TempoGauss = etime(clock,T0Gauss_eli);
fprintf('ok - Sistema resolvido\n');
fprintf('Tempo de solução = %1.3f\n\n',TempoGauss);

%** SAÍDA DE RESULTADOS, DESLOCAMENTOS, TRAÇÕES E TENSÕES NODAIS **
fprintf('5. - Preparando vetores e matrizes de resultado\n');
T0OutPut = clock;
[Tracao,Desloc] = OUTPUT(NNP,NBCU,MNEL,NOEL,MCOND,B,E,UPRES,MAXL,NLIBD,NBCT,
jt_local,NELM,ICOND,TPRES);
[SigNodal] = TensNodal(Tracao,Desloc,NEL,NNP,NOEL,X,Y,Z,E,NU);
TempoOutPut = etime(clock,T0OutPut);
fprintf('ok - Vetores e matrizes de resultado prontos\n');
fprintf('Tempo de preparação do resultado = %1.3f\n',TempoOutPut);
TempoProcesso = etime(clock,T0);
fprintf('Tempo total de processamento = %1.3f\n\n',TempoProcesso);
%** GRAVA O AMBIENTE EM ARQUIVO .MAT **
save(problema,'Tracao','Desloc','SigNodal','TempoProcesso')
fprintf('Resultados salvos no arquivo %s.MAT\n',problema);
%** FECHA TODOS OS ARQUIVOS ABERTOS PARA LEITURA **
fclose('all');
```

Listagem da função **INTEGAUS.m**

```
function [PTSGAUS,PESGAUS] = INTEGAUS(IGAUS)
%** Calcula os pontos e pesos de Gauss para a integração **
EPS = 3*10^(-14);
N = IGAUS;
M = (N+1) / 2;
XM = 0;
XL = 1;
for I = 1:M
  ZZ = cos(pi * (I - .25) / (N + .5));
  P1 = 1;
  P2 = 0;
  for J = 1:N
       P3 = P2;
    P2 = P1;
    P1 = ((2 * J - 1) * ZZ * P2 - (J - 1) * P3) / J;
  end;
  PP = N * (ZZ * P1 - P2) / (ZZ^2 - 1);
```

```
 Z1 = ZZ;
 ZZ = Z1 - P1 / PP;
 while abs(ZZ - Z1) > EPS
   P1 = 1;
   P2 = 0;
   for J = 1:N
        P3 = P2;
     P2 = P1;
     P1 = ((2 * J - 1) * ZZ * P2 - (J - 1) * P3) / J;
   end;
   PP = N * (ZZ * P1 - P2) / (ZZ^2 - 1);
   Z1 = ZZ;
   ZZ = Z1 - P1 / PP;
 end;
 PTSGAUS(I) = XM - XL * ZZ;
 PTSGAUS(N + 1 - I) = XM + XL * ZZ;
 PESGAUS(I) = 2 * XL / ((1 - ZZ^2) * PP^2);
 PESGAUS(N + 1 - I) = PESGAUS(I);
end;
```

Listagem da função **BCS.m**

```
function [NELM,JT,jt_local,ICOND,TPRES,NBCT] = BCS(NBCU,NBCP,IGAUS,NOEL,
X,Y,Z,CondContTens)

%** Processa as condições de contorno **

%** inicialização de variáveis utilizadas em funções posteriores **
DG = 0;
DH = 0;
SF = 0;
SDG = 0;
SDH = 0;
SSF = 0;

%** inicialização de vetores com valores prescritos de G e H **
GNODE = [-1, 0, 1, 1, 1, 0,-1,-1];
HNODE = [-1,-1,-1, 0, 1, 1, 1, 0];

%** verifica condições de contorno de deslocamento **
if NBCU <= 0
  display('ERRO - condições de contorno de deslocamento ausentes ou
negativas')
  STOP;
end;

%** condições de contorno de tração **
NBCT = 0;
if NBCP ~= 0

%** tensão uniforme prescrita sobre os elementos **
  for I = 1:NBCP
    JNEL = CondContTens(I,1);
    P = CondContTens(I,2);
    PXX = CondContTens(I,3);
    PYY = CondContTens(I,4);
```

```
    PZZ = CondContTens(I,5);
    PXY = CondContTens(I,6);
    PXZ = CondContTens(I,7);
    PYZ = CondContTens(I,8);
    for NE = 1:JNEL
      MEL(NE) = CondContTens(I,(8 + NE));
    end;

%** converte tensão uniforme para tração nodal equivalente **
    PMAX = max(abs([PXX,PYY,PZZ,PXY,PXZ,PYZ]));
    if PMAX == 0 & P == 0
      display('ERRO NAS CONDIÇÕES DE CONTORNO DE TRAÇÃO - PARADA')
         STOP;
    elseif PMAX == 0
      PXX = P;
      PYY = P;
      PZZ = P;
      PMAX = abs(P);
    end;
    FMAX = 0.001 * PMAX;
    for J = 1:JNEL
      MLM = MEL(J);
      [CORD] = COORD(MLM,NOEL,X,Y,Z);
      for IC = 1:8
        INOD = NOEL(MLM,IC);

%** avaliação das normais unitárias nos nós sobre os quais a tração atua **
        G = GNODE(IC);
        H = HNODE(IC);
        [DE] = SHAP8(0,0,1,G,H,1,DG,DH,SF);
        [UN] = JACOBI(1,1,DE,DG,DH,CORD);

%** trações nodais - TPRES (Ti=SIGMAij*Nj) **
%** direção 1
        NBCT = NBCT + 1;
        NELM(NBCT) = MLM;
        JT(NBCT) = INOD;
        jt_local(NBCT) = IC;
        ICOND(NBCT) = 1;
        TPRES(NBCT) = PXX * UN(1) + PXY * UN(2) + PXZ * UN(3);
%** direção 2
        if abs(TPRES(NBCT)) > FMAX
          NBCT = NBCT + 1;
          NELM(NBCT) = MLM;
          JT(NBCT) = INOD;
          jt_local(NBCT) = IC;
        end;
        ICOND(NBCT) = 2;
        TPRES(NBCT) = PXY * UN(1) + PYY * UN(2) + PYZ * UN(3);
%** direção 3
        if abs(TPRES(NBCT)) > FMAX
          NBCT = NBCT + 1;
          NELM(NBCT) = MLM;
          JT(NBCT) = INOD;
          jt_local(NBCT) = IC;
        end;
        ICOND(NBCT) = 3;
```

```
        TPRES(NBCT) = PXZ * UN(1) + PYZ * UN(2) + PZZ * UN(3);
        if abs(TPRES(NBCT)) <= FMAX
          NBCT = NBCT - 1;
        end;
      end;
    end;
  end;
end;
```

Listagem da função **FRMTRX1.m**

```
function [A,BT,NLIBD] = FRMTRX1(NNP,NEL,NBCU,NBPTU,NBCT,IGAUS,MAXL,NU,E,
NOEL,NELM,MNEL,NODPT,UPRES,TPRES,ICOND,MCOND,NCOND,JT,jt_local,CG,XG,X,Y,Z)

%** Constrói a matriz e o segundo membro pelo método dos elementos de contorno
**

%** inicialização de variáveis utilizadas em funções posteriores **
DG = zeros(1,8,realpow(IGAUS,2));
DH = zeros(1,8,realpow(IGAUS,2));
SF = zeros(1,8,realpow(IGAUS,2));
SSF = zeros(1,4,realpow(IGAUS,2));
SDG = zeros(1,4,realpow(IGAUS,2));
SDH = zeros(1,4,realpow(IGAUS,2));
TRIG = 0;
TRIH = 0;

%** cálculo do número de graus de liberdade do problema **
NLIBD = 3 * NNP;

%** inicializações **
h_global = zeros(NLIBD);

%** armazena valores das funções de forma e suas derivadas **
%** em cada ponto de Gauss em vetores **
for IG = 1:IGAUS
  JG = [1:1:IGAUS]; %  for JG = 1:IGAUS
  G = XG(IG);
  H = XG(JG);
%** usado em conjunto com a função JACOBI (veja função INTGR) **
  [DE,DG,DH,SF] = SHAP8(IG,JG,0,G,H,IGAUS,DG,DH,SF);
%** usado em conjunto com a função TRIANG (veja função ITSS) **
  [SSF,SDG,SDH] = LINEAR(IG,JG,G,H,IGAUS,SSF,SDG,SDH);
%  end;
end;
DG = permute(DG,[3 2 1]);
DH = permute(DH,[3 2 1]);
SSF = permute(SSF,[3 2 1]);
SDG = permute(SDG,[3 2 1]);
SDH = permute(SDH,[3 2 1]);

%** formação da matriz e segundo membro **

%** loop sobre os elementos **
for M = 1:NEL
```

```
%** obtenção das coordenadas dos nós do elemento **
  [CORD] = COORD(M,NOEL,X,Y,Z);

%** cálculo do Jacobiano e fator para o elemento sem singularidade **
  L = [1:1:realpow(IGAUS,2)];
  [UN,JACOB] = JACOBI(L,0,DE,DG,DH,CORD);
  CGMM = CG' * CG;
  bloco = [1:1:IGAUS];
  for i = 1:IGAUS
    CGM(1,1,bloco) = CGMM(i,:);
    bloco = bloco + IGAUS;
  end;
  FACT(1,1,L) = JACOB(1,1,L) .* CGM(1,1,L);

%** limpando variáveis **
  clear CGMM CGM JACOB bloco;

%** loop sobre os nós fonte **
  for NI = 1:NNP

%** integração numérica dos produtos da solução fundamental **
    JCON = 0;
    for NC = 1:8
      if NI ~= NOEL(M,NC)
      else
        ND = NC;
        JCON = 1;
      end;
    end;
    if JCON == 0

%** primeiro argumento da solução fundamental não é um nó do elemento **
      [A1,A2,A3,B1,B2,B3] = INTGR(NI,M,L,NLIBD,IGAUS,NU,E,MAXL,CORD,NOEL,
X,Y,Z,DE,SF,UN,FACT);
    else

%** primeiro argumento da solução fundamental é um nó do elemento **
      [A1,A2,A3,B1,B2,B3,DE,TRIG,TRIH] = ITSS(NI,M,L,ND,NLIBD,IGAUS,NU,E,
MAXL,CORD,CG,NOEL,X,Y,Z,DE,DG,DH,SF,SDG,SDH,SSF,TRIG,TRIH);
    end;

%** colocação das matrizes H e G locais em H e G globais **
    node_global = NOEL(M,:);
    node_local = [1:1:8];
    idx_global = 3 * node_global - 2;
    idx_local = 3 * node_local - 2;
    idx_linha = 3 * NI - 2;
    idx_g = 24 * M - 23;
    h_global(idx_linha,idx_global)     =     h_global(idx_linha,idx_global)     +
A1(1,idx_local);
    h_global(idx_linha,idx_global+1)   =   h_global(idx_linha,idx_global+1)   +
A1(1,idx_local+1);
    h_global(idx_linha,idx_global+2)   =   h_global(idx_linha,idx_global+2)   +
A1(1,idx_local+2);
    h_global(idx_linha+1,idx_global)   =   h_global(idx_linha+1,idx_global)   +
A2(1,idx_local);
```

```
    h_global(idx_linha+1,idx_global+1) = h_global(idx_linha+1,idx_global+1) +
A2(1,idx_local+1);
    h_global(idx_linha+1,idx_global+2) = h_global(idx_linha+1,idx_global+2) +
A2(1,idx_local+2);
    h_global(idx_linha+2,idx_global)   =  h_global(idx_linha+2,idx_global)   +
A3(1,idx_local);
    h_global(idx_linha+2,idx_global+1) = h_global(idx_linha+2,idx_global+1) +
A3(1,idx_local+1);
    h_global(idx_linha+2,idx_global+2) = h_global(idx_linha+2,idx_global+2) +
A3(1,idx_local+2);
    g_global(idx_linha,idx_g:idx_g+23) = B1(1,:);
    g_global(idx_linha+1,idx_g:idx_g+23) = B2(1,:);
    g_global(idx_linha+2,idx_g:idx_g+23) = B3(1,:);
  end;
end;

%** calculando termos da diagonal principal de H por consideração de corpo
rígido **
h_global1 = h_global(:,[1:3:NLIBD-2]);
h_global2 = h_global(:,[2:3:NLIBD-1]);
h_global3 = h_global(:,[3:3:NLIBD]);
diagonal(:,1) = -sum(h_global1,2);
diagonal(:,2) = -sum(h_global2,2);
diagonal(:,3) = -sum(h_global3,2);
for idx_diag = 1:NNP
  idx = 3 * idx_diag - 2;
  h_global(idx:idx+2,idx:idx+2) = diagonal(idx:idx+2,:);
end;

%** salva matrizes H e G globais para reutilização com diferentes CDC **
save('HeG','h_global','g_global');

%** prepara matriz A **
A = h_global;

%** contribuição das trações conhecidas para o segundo membro **
[BT,AO] = FMATR(NLIBD,NEL,NBCU,NBCT,E,NOEL,NELM,MNEL,ICOND,MCOND,JT,
jt_local,TPRES,g_global);

%** limpando posições para colocação de G global **
for n = 1:NBCU
  no = 3 * (NOEL(MNEL(n),:) - 1) + MCOND(n);
  A(:,no) = 0;
end;

%** contribuição dos deslocamentos conhecidos para o segundo membro **
for NI = 1:NNP
  NZ = zeros(NNP,1);
  JX = 3 * NI - 2;
  for N = 1:NBCU
    M = MNEL(N);
    for IC = 1:8
      U = UPRES(N,IC) / MAXL;
      INOD = NOEL(M,IC);
      if NZ(INOD) == MCOND(N)
      else
        NODE = 3 * (INOD - 1) + MCOND(N);
```

```
        if abs(U) >= 1.E-16
          BT(JX) = BT(JX) - h_global(JX,NODE) * U;
          BT(JX+1) = BT(JX+1) - h_global(JX+1,NODE) * U;
          BT(JX+2) = BT(JX+2) - h_global(JX+2,NODE) * U;
        end;
        no_g = 24 * (M - 1) + 3 * (IC - 1) + MCOND(N);
        A(JX,NODE) = A(JX,NODE) + AO(JX,no_g);
        A(JX+1,NODE) = A(JX+1,NODE) + AO(JX+1,no_g);
        A(JX+2,NODE) = A(JX+2,NODE) + AO(JX+2,no_g);
        NZ(INOD) = MCOND(N);
      end;
    end;
  end;
end;

%** modifica matriz para restrições pontuais **
if NBPTU ~= 0
  for N = 1:NBPTU
    NODE = 3 * (NODPT(N) - 1) + NCOND(N);
    if NODPT(N) ~= NI
      A1(NODE) = 0;
      A2(NODE) = 0;
      A3(NODE) = 0;
    else
      for JB = 1:NLIBD
        if NCOND(N) == 1
          A1(JB) = 0;
          A1(NODE) = 1;
          BT(NODE) = 0;
        elseif NCOND(N) == 2
          A2(JB) = 0;
          A2(NODE) = 1;
          BT(NODE) = 0;
        elseif NCOND(N) == 3
          A3(JB) = 0;
          A3(NODE) = 1;
          BT(NODE) = 0;
        end;
      end;
    end;
  end;
end;
```

Listagem da função **FRMTRX2.m**

```
function [A,BT,NLIBD] = FRMTRX2(arqHeG,NNP,NEL,NBCU,NBPTU,NBCT,IGAUS,MAXL,
NU,E,NOEL,NELM,MNEL,NODPT,UPRES,TPRES,ICOND,MCOND,NCOND,JT,jt_local,CG,XG,X,Y,
Z)

%** Constrói a matriz e o segundo membro pelo método dos elementos de contorno
**

%** cálculo do número de graus de liberdade do problema **
NLIBD = 3 * NNP;

%** carrega matrizes H e G globais para reutilização com diferentes CDC **
```

```
load(arqHeG);

%** prepara matriz A **
A = h_global;

%** contribuição das trações conhecidas para o segundo membro **
[BT,AO] = FMATR(NLIBD,NEL,NBCU,NBCT,E,NOEL,NELM,MNEL,ICOND,MCOND,JT,
jt_local,TPRES,g_global);

%** limpando posições para colocação de G global **
for n = 1:NBCU
  no = 3 * (NOEL(MNEL(n),:) - 1) + MCOND(n);
  A(:,no) = 0;
end;

%** contribuição dos deslocamentos conhecidos para o segundo membro **
for NI = 1:NNP
  NZ = zeros(NNP,1);
  JX = 3 * NI - 2;
  for N = 1:NBCU
    M = MNEL(N);
    for IC = 1:8
      U = UPRES(N,IC) / MAXL;
      INOD = NOEL(M,IC);
      if NZ(INOD) == MCOND(N)
      else
        NODE = 3 * (INOD - 1) + MCOND(N);
        if abs(U) >= 1.E-16
          BT(JX) = BT(JX) - h_global(JX,NODE) * U;
          BT(JX+1) = BT(JX+1) - h_global(JX+1,NODE) * U;
          BT(JX+2) = BT(JX+2) - h_global(JX+2,NODE) * U;
        end;
        no_g = 24 * (M - 1) + 3 * (IC - 1) + MCOND(N);
        A(JX,NODE) = A(JX,NODE) + AO(JX,no_g);
        A(JX+1,NODE) = A(JX+1,NODE) + AO(JX+1,no_g);
        A(JX+2,NODE) = A(JX+2,NODE) + AO(JX+2,no_g);
        NZ(INOD) = MCOND(N);
      end;
    end;
  end;
end;

%** modifica matriz para restrições pontuais **
if NBPTU ~= 0
  for N = 1:NBPTU
    NODE = 3 * (NODPT(N) - 1) + NCOND(N);
    if NODPT(N) ~= NI
      A1(NODE) = 0;
      A2(NODE) = 0;
      A3(NODE) = 0;
    else
      for JB = 1:NLIBD
        if NCOND(N) == 1
          A1(JB) = 0;
          A1(NODE) = 1;
          BT(NODE) = 0;
        elseif NCOND(N) == 2
```

```
          A2(JB) = 0;
          A2(NODE) = 1;
          BT(NODE) = 0;
        elseif NCOND(N) == 3
          A3(JB) = 0;
          A3(NODE) = 1;
          BT(NODE) = 0;
        end;
      end;
    end;
  end;
end;
```

Listagem da função **OUTPUT.m**

```
function [Tracao,Desloc] = OUTPUT (NNP,NEL,NBCU,MNEL,NOEL,MCOND,BT,E,UPRES,
MAXL,NLIBD,NBCT,jt_local,NELM,ICOND,TPRES)

%** Retorna os valores nodais para deslocamentos e trações **

%** troca de valores calculados de U e TRAC para nós nos quais as condições de
contorno **
%** de deslocamento estão prescritas mostra as trações calculadas **
NZ = zeros(NNP,1);
Tracao = zeros(8*NEL,1);
for N = 1:NBCU
  M = MNEL(N);
  for I = 1:8
    INOD = NOEL(M,I);
    if NZ(INOD) ~= MCOND(N)
      NODE = 3 * (INOD - 1) + MCOND(N);
      TRACT(NODE) = BT(NODE) * E;
      BT(NODE) = UPRES(N,I) / MAXL;
            Tracao(8*(M-1)+I,MCOND(N)) = TRACT(NODE);
      NZ(INOD) = MCOND(N);
    end;
  end;
end;

%** removendo o fator de escala para valores calculados de deslocamento **
for I = 1:NLIBD
  BT(I) = BT(I) * MAXL;
end;

%** vetores de saída para deslocamento nodal **
for I = 1:NNP
  Desloc(I,1) = BT(3 * I - 2);
  Desloc(I,2) = BT(3 * I - 1);
  Desloc(I,3) = BT(3 * I);
end;
for cdctrac = 1:NBCT
  Tracao((8*(NELM(cdctrac)-1)+jt_local(cdctrac)),ICOND(cdctrac)) =
TPRES(cdctrac);
end;
```

Listagem da função **COORD.m**

```
function [CORD] = COORD(M,NOEL,X,Y,Z)

%** Coloca o número do nó do elemento e suas coordenadas nodais em matrizes **

%** matriz de coordenadas nodais de cada elemento **
for I = 1:8
%ivdep
  INC = NOEL(M,I);
  CORD(I,1) = X(INC);
  CORD(I,2) = Y(INC);
  CORD(I,3) = Z(INC);
end;
```

Listagem da função **SHAP8.m**

```
function [DE,DG,DH,SF] = SHAP8(IG,JG,KY,G,H,IGAUS,DG,DH,SF)

%** Armazena as funções de forma e derivadas em uma matriz **

%** inicialização de variáveis locais **
G1 = 1 + G;
G2 = 1 - G;
H1 = 1 + H;
H2 = 1 - H;
G3 = G1 .* G2;
H3 = H1 .* H2;

%** derivadas das funções de forma em relação a G **
DE(1,1,:) = 0.25 * H2 .* (2 * G + H);
DE(1,2,:) = -G .* H2;
DE(1,3,:) = 0.25 * H2 .* (2 * G - H);
DE(1,4,:) = 0.5 * H3;
DE(1,5,:) = 0.25 * H1 .* (2 * G + H);
DE(1,6,:) = -G .* H1;
DE(1,7,:) = 0.25 * H1 .* (2 * G - H);
DE(1,8,:) = -0.5 * H3;

%** derivadas das funções de forma em relação a H **
DE(2,1,:) = 0.25 * G2 .* (2 * H + G);
DE(2,2,:) = -0.5 * G3;
DE(2,3,:) = 0.25 * G1 .* (2 * H - G);
DE(2,4,:) = -H .* G1;
DE(2,5,:) = 0.25 * G1 .* (2 * H + G);
DE(2,6,:) = 0.5 * G3;
DE(2,7,:) = 0.25 * G2 .* (2 * H - G);
DE(2,8,:) = -H .* G2;

%** funções de forma **
DE(3,1,:) = -0.25 * G2 .* H2 .* (G + H1);
DE(3,2,:) = 0.5 * G3 .* H2;
DE(3,3,:) = 0.25 * G1 .* H2 .* (G - H1);
DE(3,4,:) = 0.5 * G1 .* H3;
DE(3,5,:) = 0.25 * G1 .* H1 .* (H - G2);
```

```
DE(3,6,:) = 0.5 * H1 .* G3;
DE(3,7,:) = 0.25 * G2 .* H1 .* (H - G1);
DE(3,8,:) = 0.5 * G2 .* H3;
if KY == 0
  L = [IG * IGAUS - IGAUS + 1:1:IG * IGAUS];
  DG(:,:,L) = DE(1,:,:);
  DH(:,:,L) = DE(2,:,:);
  SF(:,:,L) = DE(3,:,:);
end;
```

Listagem da função **LINEAR.m**

```
function [SSF,SDG,SDH] = LINEAR(IG,JG,G,H,IGAUS,SSF,SDG,SDH)

%** Calcula funções de forma lineares e suas derivadas **

%** inicialização de variáveis locais **
L = [IG * IGAUS - IGAUS + 1:1:IG * IGAUS];
G1 = 1 + G;
G2 = 1 - G;
H1 = 1 + H;
H2 = 1 - H;

%** funções de forma lineares **
SSF(1,1,L) = 0.25 * G2 .* H2;
SSF(1,2,L) = 0.25 * G1 .* H2;
SSF(1,3,L) = 0.25 * G1 .* H1;
SSF(1,4,L) = 0.25 * G2 .* H1;

%** derivadas das funções de forma em relação a G **
SDG(1,1,L) = -0.25 * H2;
SDG(1,2,L) = 0.25 * H2;
SDG(1,3,L) = 0.25 * H1;
SDG(1,4,L) = -0.25 * H1;

%** derivadas das funções de forma em relação a H **
SDH(1,1,L) = -0.25 * G2;
SDH(1,2,L) = -0.25 * G1;
SDH(1,3,L) = 0.25 * G1;
SDH(1,4,L) = 0.25 * G2;
```

Listagem da função **JACOBI.m**

```
function [UN,JACOB] = JACOBI(L,KY,DE,DG,DH,CORD)

%** Avalia o jacobiano e os componentes do vetor normal unitário **

%** inicializa AJAC - soma dos produtos das derivadas das funções de forma e
%   coordenadas nodais **
AJAC = zeros(2,3);

%** calcula AJAC **
if KY == 0
  DEA = DG;
  DEB = DH;
```

```
else
  DEA = permute(DE(1,:,L),[3 2 1]);
  DEB = permute(DE(2,:,L),[3 2 1]);
end;
AJAC(1,1,L) = DEA(L,:) * CORD(:,1);
AJAC(1,2,L) = DEA(L,:) * CORD(:,2);
AJAC(1,3,L) = DEA(L,:) * CORD(:,3);
AJAC(2,1,L) = DEB(L,:) * CORD(:,1);
AJAC(2,2,L) = DEB(L,:) * CORD(:,2);
AJAC(2,3,L) = DEB(L,:) * CORD(:,3);

%** componentes do vetor normal **
AJAC1(1,1,L) = AJAC(1,2,L) .* AJAC(2,3,L) - AJAC(1,3,L) .* AJAC(2,2,L);
AJAC2(1,1,L) = AJAC(1,3,L) .* AJAC(2,1,L) - AJAC(1,1,L) .* AJAC(2,3,L);
AJAC3(1,1,L) = AJAC(1,1,L) .* AJAC(2,2,L) - AJAC(1,2,L) .* AJAC(2,1,L);

%** cálculo do jacobiano - JACOB **
JACOB(1,1,L) = realsqrt(realpow(AJAC1(1,1,L),2) + realpow(AJAC2(1,1,L),2) +
realpow(AJAC3(1,1,L),2));
if min(JACOB) <= 1E-8
  if KY == 2
    return
  end;
  display('JACOB = 0. - programa interrompido')
  STOP;
end;

%** normalização do vetor normal e obtenção do vetor normal unitário **
UN(1,1,L) = AJAC1(1,1,L) ./ JACOB(1,1,L);
UN(1,2,L) = AJAC2(1,1,L) ./ JACOB(1,1,L);
UN(1,3,L) = AJAC3(1,1,L) ./ JACOB(1,1,L);
```

Listagem da função **TRIANG.m**

```
function [UN,JACO,DE,TRIG,TRIH] = TRIANG(L,ND,KK,IGAUS,CORD,DG,DH,SF,SDG,
SDH,SSF,TRIG,TRIH)

%** Calcula o jacobiano para o caso em que o primeiro argumento **
%** das soluções fundamentais é um nó do elemento **

%** inicialização dos vetores com valores prescritos de G e H
AG = [-1, 0, 1, 1, 1, 0,-1,-1];
AH = [-1,-1,-1, 0, 1, 1, 1, 0];

%** inicialização de G, H e TJAC **
G = 0;
H = 0;
TJAC = zeros(2,2);

%** ND é um nó de canto **
if rem(ND,2) ~= 0 %função MOD do fortran equivale a REM no MatLab
  ND2 = ND + 2;
  if ND2 > 8
    ND2 = ND2 - 8;
  end;
  ND4 = ND2 + 2;
```

```
  if ND4 > 8
    ND4 = ND4 - 8;
  end;
  ND6 = ND4 + 2;
  if ND6 > 8
    ND6 = ND6 - 8;
  end;

%** coordenada nodal local (quadrilateral) do primeiro sub-elemento triangular
**
  if KK == 1
    TRIG(1,1) = AG(ND);
    TRIG(1,2) = AG(ND);
    TRIG(1,3) = AG(ND2);
    TRIG(1,4) = AG(ND4);
    TRIH(1,1) = AH(ND);
    TRIH(1,2) = AH(ND);
    TRIH(1,3) = AH(ND2);
    TRIH(1,4) = AH(ND4);
  else

%** segundo sub-elemento triangular **
    TRIG(2,1) = AG(ND);
    TRIG(2,2) = AG(ND);
    TRIG(2,3) = AG(ND4);
    TRIG(2,4) = AG(ND6);
    TRIH(2,1) = AH(ND);
    TRIH(2,2) = AH(ND);
    TRIH(2,3) = AH(ND4);
    TRIH(2,4) = AH(ND6);
  end;
else

%** ND é um nó no meio da aresta **
  ND1 = ND + 1;
  if ND1 > 8
    ND1 = ND1 - 8;
  end;
  ND3 = ND1 + 2;
  if ND3 > 8
    ND3 = ND3 - 8;
  end;
  ND5 = ND3 + 2;
  if ND5 > 8
    ND5 = ND5 - 8;
  end;
  ND7 = ND5 + 2;
  if ND7 > 8
    ND7 = ND7 - 8;
  end;

%** coordenada nodal local (quadrilateral) do primeiro sub-elemento triangular
**
  if KK == 1
    TRIG(1,1) = AG(ND);
    TRIG(1,2) = AG(ND);
    TRIG(1,3) = AG(ND1);
```

```
    TRIG(1,4) = AG(ND3);
    TRIH(1,1) = AH(ND);
    TRIH(1,2) = AH(ND);
    TRIH(1,3) = AH(ND1);
    TRIH(1,4) = AH(ND3);

%** segundo sub-elemento triangular **
  elseif KK == 2
    TRIG(2,1) = AG(ND);
    TRIG(2,2) = AG(ND);
    TRIG(2,3) = AG(ND3);
    TRIG(2,4) = AG(ND5);
    TRIH(2,1) = AH(ND);
    TRIH(2,2) = AH(ND);
    TRIH(2,3) = AH(ND3);
    TRIH(2,4) = AH(ND5);
  else

 %** terceiro sub-elemento triangular **
    TRIG(3,1) = AG(ND);
    TRIG(3,2) = AG(ND);
    TRIG(3,3) = AG(ND5);
    TRIG(3,4) = AG(ND7);
    TRIH(3,1) = AH(ND);
    TRIH(3,2) = AH(ND);
    TRIH(3,3) = AH(ND5);
    TRIH(3,4) = AH(ND7);
  end;
 end;

 %** coordenada local (quadrilateral) do ponto de integração **
 I = [1:1:4];
 G = SSF(L,I) * TRIG(KK,I)';
 H = SSF(L,I) * TRIH(KK,I)';

 %** matriz jacobiana de transformação de coordenadas local e seu determinante
 **
 TJAC11(L,:) = SDG(L,I) * TRIG(KK,I)';
 TJAC12(L,:) = SDG(L,I) * TRIH(KK,I)';
 TJAC21(L,:) = SDH(L,I) * TRIG(KK,I)';
 TJAC22(L,:) = SDH(L,I) * TRIH(KK,I)';
 DETER(1,1,L) = abs(TJAC11 .* TJAC22 - TJAC12 .* TJAC21);
 if min(DETER) <= 1.E-8
   display('determinante = 0 na função TRIANG - programa interrompido')
   STOP;
 end;

 %** jacobiano para integração numérica **
 [DE] = SHAP8(0,0,1,G,H,IGAUS,DG,DH,SF);
 [UN,JACOB] = JACOBI(L,1,DE,DG,DH,CORD);
 JACO = JACOB .* DETER;
```

Listagem da função **KERNEL.m**

```
function [UT,TT] = KERNEL(NI,L,KY,NU,E,MAXL,CORD,X,Y,Z,UN,DE,SF)
```

```
%** Calcula as soluções fundamentais U e T **

%** inicialização de variável local **
HPI = 25.13274123;

%** coordenadas do ponto P **
XP(1) = X(NI);
XP(2) = Y(NI);
XP(3) = Z(NI);

%** coordenadas do ponto Q **
XQ = zeros(3,1,size(L,3));
if KY == 0
  SHAP = permute(SF,[3 2 1]);
else
  SHAP = permute(DE(3,:,:),[3 2 1]);
end;
XQ(1,:,L) = SHAP * CORD(:,1);
XQ(2,:,L) = SHAP * CORD(:,2);
XQ(3,:,L) = SHAP * CORD(:,3);

%** calcula a distância entre P e Q - RR **
XX(L) = XP(1) - XQ(1,:,L);
YY(L) = XP(2) - XQ(2,:,L);
ZZ(L) = XP(3) - XQ(3,:,L);
SUMSQ(L) = realpow(XX(L),2) + realpow(YY(L),2) + realpow(ZZ(L),2);

%** verificação para singularidade **
if min(SUMSQ) <= 1.0E-20
  display('Primeiro e segundo argumentos coincidentes - programa
interrompido')
  STOP;
end;

RR(L) = realsqrt(SUMSQ(L));

%** cossenos diretores do vetor PQ **
COSPQ(1,1,L) = XX(L) ./ RR(L);
COSPQ(1,2,L) = YY(L) ./ RR(L);
COSPQ(1,3,L) = ZZ(L) ./ RR(L);
UNCO(1,1,L) = UN(1,1,L) .* COSPQ(1,1,L) + UN(1,2,L) .* COSPQ(1,2,L) +
UN(1,3,L) .* COSPQ(1,3,L);

%** produtos de cossenos diretores **
COAA(1,1,L) = COSPQ(1,1,L) .* COSPQ(1,1,L);
COBB(1,1,L) = COSPQ(1,2,L) .* COSPQ(1,2,L);
COCC(1,1,L) = COSPQ(1,3,L) .* COSPQ(1,3,L);
COAB(1,1,L) = COSPQ(1,1,L) .* COSPQ(1,2,L);
COBC(1,1,L) = COSPQ(1,2,L) .* COSPQ(1,3,L);
COCA(1,1,L) = COSPQ(1,3,L) .* COSPQ(1,1,L);

%** fatores constantes para as soluções fundamentais **
EUA = (3 - 4 * NU);
EUB = (1 - 2 * NU);
FAC = HPI * (1 - NU);
FACT1(1,1,L) = (1 + NU) ./ (FAC * E * RR(L));
FACT2(1,1,L) = 1 ./ (FAC * SUMSQ(L));
```

```
FACT4(1,1,L) = EUB * FACT2(1,1,L);
ETA(1,1,L) = FACT4(1,1,L) .* UNCO(1,1,L);
ETB(1,1,L) = 3 * FACT2(1,1,L) .* UNCO(1,1,L);

%** fatores de escala da solução fundamental UT **
FACT1(1,1,L) = FACT1(1,1,L) * E / MAXL;

%** solução fundamental UT **
UT(1,1,L) = FACT1(1,1,L) .* (EUA + COAA(1,1,L));
UT(1,2,L) = FACT1(1,1,L) .* COAB(1,1,L);
UT(1,3,L) = FACT1(1,1,L) .* COCA(1,1,L);
UT(2,2,L) = FACT1(1,1,L) .* (EUA + COBB(1,1,L));
UT(2,3,L) = FACT1(1,1,L) .* COBC(1,1,L);
UT(3,3,L) = FACT1(1,1,L) .* (EUA + COCC(1,1,L));
UT(2,1,L) = UT(1,2,L);
UT(3,1,L) = UT(1,3,L);
UT(3,2,L) = UT(2,3,L);

%** solução fundamental TT **
TT(1,1,L) = ETA(1,1,L) + ETB(1,1,L) .* COAA(1,1,L);
TT(2,2,L) = ETA(1,1,L) + ETB(1,1,L) .* COBB(1,1,L);
TT(3,3,L) = ETA(1,1,L) + ETB(1,1,L) .* COCC(1,1,L);
TAB(1,1,L) = ETB(1,1,L) .* COAB(1,1,L);
TBC(1,1,L) = ETB(1,1,L) .* COBC(1,1,L);
TCA(1,1,L) = ETB(1,1,L) .* COCA(1,1,L);
AAB(1,1,L) = (UN(1,1,L) .* COSPQ(1,2,L) - UN(1,2,L) .* COSPQ(1,1,L)) .*
FACT4(1,1,L);
ABC(1,1,L) = (UN(1,2,L) .* COSPQ(1,3,L) - UN(1,3,L) .* COSPQ(1,2,L)) .*
FACT4(1,1,L);
ACA(1,1,L) = (UN(1,3,L) .* COSPQ(1,1,L) - UN(1,1,L) .* COSPQ(1,3,L)) .*
FACT4(1,1,L);
TT(1,2,L) = TAB(1,1,L) + AAB(1,1,L);
TT(1,3,L) = TCA(1,1,L) - ACA(1,1,L);
TT(2,1,L) = TAB(1,1,L) - AAB(1,1,L);
TT(2,3,L) = TBC(1,1,L) + ABC(1,1,L);
TT(3,1,L) = TCA(1,1,L) + ACA(1,1,L);
TT(3,2,L) = TBC(1,1,L) - ABC(1,1,L);
```

Listagem da função **INTGR.m**

```
function [A1,A2,A3,B1,B2,B3] = INTGR(NI,M,L,NLIBD,IGAUS,NU,E,MAXL,CORD,
NOEL,X,Y,Z,DE,SF,UN,FACT)

%** Realiza a integração numérica dos produtos da solução fundamental para a
construção **
%** da matriz e segundo membro quando o primeiro argumento não é um nó do
elemento **

%** inicializações **
A1 = zeros(1,24,realpow(IGAUS,2));
A2 = zeros(1,24,realpow(IGAUS,2));
A3 = zeros(1,24,realpow(IGAUS,2));
B1 = zeros(1,24,realpow(IGAUS,2));
B2 = zeros(1,24,realpow(IGAUS,2));
B3 = zeros(1,24,realpow(IGAUS,2));
```

```
%** integração numérica dos produtos da solução fundamental **

%** solução fundamental **
[UT,TT] = KERNEL(NI,L,0,NU,E,MAXL,CORD,X,Y,Z,UN,DE,SF);
MODE = 1;

for IC = 1:8
  SFACT(1,1,L) = SF(1,IC,L) .* FACT(1,1,L);

%** integral do produto de TT e funções de forma **
  A1(1,MODE,L) = A1(1,MODE,L) + TT(1,1,L) .* SFACT(1,1,L);
  A1(1,MODE+1,L) = A1(1,MODE+1,L) + TT(1,2,L) .* SFACT(1,1,L);
  A1(1,MODE+2,L) = A1(1,MODE+2,L) + TT(1,3,L) .* SFACT(1,1,L);
  A2(1,MODE,L) = A2(1,MODE,L) + TT(2,1,L) .* SFACT(1,1,L);
  A2(1,MODE+1,L) = A2(1,MODE+1,L) + TT(2,2,L) .* SFACT(1,1,L);
  A2(1,MODE+2,L) = A2(1,MODE+2,L) + TT(2,3,L) .* SFACT(1,1,L);
  A3(1,MODE,L) = A3(1,MODE,L) + TT(3,1,L) .* SFACT(1,1,L);
  A3(1,MODE+1,L) = A3(1,MODE+1,L) + TT(3,2,L) .* SFACT(1,1,L);
  A3(1,MODE+2,L) = A3(1,MODE+2,L) + TT(3,3,L) .* SFACT(1,1,L);
%** integral do produto de UT e funções de forma **
  B1(1,MODE,L) = B1(1,MODE,L) + UT(1,1,L) .* SFACT(1,1,L);
  B1(1,MODE+1,L) = B1(1,MODE+1,L) + UT(1,2,L) .* SFACT(1,1,L);
  B1(1,MODE+2,L) = B1(1,MODE+2,L) + UT(1,3,L) .* SFACT(1,1,L);
  B2(1,MODE,L) = B2(1,MODE,L) + UT(2,1,L) .* SFACT(1,1,L);
  B2(1,MODE+1,L) = B2(1,MODE+1,L) + UT(2,2,L) .* SFACT(1,1,L);
  B2(1,MODE+2,L) = B2(1,MODE+2,L) + UT(2,3,L) .* SFACT(1,1,L);
  B3(1,MODE,L) = B3(1,MODE,L) + UT(3,1,L) .* SFACT(1,1,L);
  B3(1,MODE+1,L) = B3(1,MODE+1,L) + UT(3,2,L) .* SFACT(1,1,L);
  B3(1,MODE+2,L) = B3(1,MODE+2,L) + UT(3,3,L) .* SFACT(1,1,L);
  MODE = MODE + 3;
end;

%** realização da somatória sobre os pontos de Gauss **
A1 = sum(A1,3);
A2 = sum(A2,3);
A3 = sum(A3,3);
B1 = sum(B1,3);
B2 = sum(B2,3);
B3 = sum(B3,3);

%** limpando variáveis **
clear SFACT UT TT;
```

Listagem da função **ITSS.m**

```
function [A1,A2,A3,B1,B2,B3,DE,TRIG,TRIH] = ITSS(NI,M,L,ND,NLIBD,IGAUS,NU,
E,MAXL,CORD,CG,NOEL,X,Y,Z,DE,DG,DH,SF,SDG,SDH,SSF,TRIG,TRIH)

%** Realiza a integração dos produtos da solução fundamental quando o primeiro
argumento **
%** é um nó do elemento **

%** inicialização do vetor com valores de nó do triângulo prescritos **
NTR = zeros(1,8);
NTR = [2, 3, 2, 3, 2, 3, 2, 3];
NTRI = NTR(ND);
```

```
%** inicialização **
A1 = zeros(1,24,realpow(IGAUS,2));
A2 = zeros(1,24,realpow(IGAUS,2));
A3 = zeros(1,24,realpow(IGAUS,2));
B1 = zeros(1,24,realpow(IGAUS,2));
B2 = zeros(1,24,realpow(IGAUS,2));
B3 = zeros(1,24,realpow(IGAUS,2));

%** integração numérica dos produtos da solução fundamental **
for KK = 1:NTRI
  [UN,JACO,DE,TRIG,TRIH] = TRIANG(L,ND,KK,IGAUS,CORD,DG,DH,SF,SDG,SDH,SSF,
TRIG,TRIH);
  [UT,TT] = KERNEL(NI,L,1,NU,E,MAXL,CORD,X,Y,Z,UN,DE,SF);
  CGMM = CG' * CG;
  bloco = [1:1:IGAUS];
  for i = 1:IGAUS
    CGM(1,1,bloco) = CGMM(i,:);
    bloco = bloco + IGAUS;
  end;
  FACT(1,1,L) = JACO(1,1,L) .* CGM(1,1,L);
  MODE = 1;
  for IC = 1:8
    SFACT(1,1,L) = DE(3,IC,L) .* FACT(1,1,L);
    NODE = NOEL(M,IC);
    if NI ~= NODE

%** integral do produto de TT e funções de forma **
      A1(1,MODE,L) = A1(1,MODE,L) + TT(1,1,L) .* SFACT(1,1,L);
      A1(1,MODE+1,L) = A1(1,MODE+1,L) + TT(1,2,L) .* SFACT(1,1,L);
      A1(1,MODE+2,L) = A1(1,MODE+2,L) + TT(1,3,L) .* SFACT(1,1,L);
      A2(1,MODE,L) = A2(1,MODE,L) + TT(2,1,L) .* SFACT(1,1,L);
      A2(1,MODE+1,L) = A2(1,MODE+1,L) + TT(2,2,L) .* SFACT(1,1,L);
      A2(1,MODE+2,L) = A2(1,MODE+2,L) + TT(2,3,L) .* SFACT(1,1,L);
      A3(1,MODE,L) = A3(1,MODE,L) + TT(3,1,L) .* SFACT(1,1,L);
      A3(1,MODE+1,L) = A3(1,MODE+1,L) + TT(3,2,L) .* SFACT(1,1,L);
      A3(1,MODE+2,L) = A3(1,MODE+2,L) + TT(3,3,L) .* SFACT(1,1,L);
    end;

%** integral do produto de UT e funções de forma **
    B1(1,MODE,L) = B1(1,MODE,L) + UT(1,1,L) .* SFACT(1,1,L);
    B1(1,MODE+1,L) = B1(1,MODE+1,L) + UT(1,2,L) .* SFACT(1,1,L);
    B1(1,MODE+2,L) = B1(1,MODE+2,L) + UT(1,3,L) .* SFACT(1,1,L);
    B2(1,MODE,L) = B2(1,MODE,L) + UT(2,1,L) .* SFACT(1,1,L);
    B2(1,MODE+1,L) = B2(1,MODE+1,L) + UT(2,2,L) .* SFACT(1,1,L);
    B2(1,MODE+2,L) = B2(1,MODE+2,L) + UT(2,3,L) .* SFACT(1,1,L);
    B3(1,MODE,L) = B3(1,MODE,L) + UT(3,1,L) .* SFACT(1,1,L);
    B3(1,MODE+1,L) = B3(1,MODE+1,L) + UT(3,2,L) .* SFACT(1,1,L);
    B3(1,MODE+2,L) = B3(1,MODE+2,L) + UT(3,3,L) .* SFACT(1,1,L);
    MODE = MODE + 3;
  end;
end;

%** realização da somatória sobre os pontos de Gauss **
A1 = sum(A1,3);
A2 = sum(A2,3);
A3 = sum(A3,3);
```

```
B1 = sum(B1,3);
B2 = sum(B2,3);
B3 = sum(B3,3);

%** limpando variáveis **
clear UN JACO CGMM CGM bloco FACT SFACT UT TT;
```

Listagem da função **FMATR.m**

```
function [BT,AO] = FMATR(NLIBD,NEL,NBCU,NBCT,E,NOEL,NELM,MNEL,ICOND,MCOND,
JT,jt_local,TPRES,g_global)

%** Aplica as condições de contorno para formar a matriz e o segundo membro **

%** inicialização de variável local
BT = zeros(NLIBD,1);
AO = zeros(NLIBD,24*NEL);

%** condições de contorno de tração prescritas **

%** contribuição ao segundo membro correspondente aos nós **
%** sobre os quais age a tensão uniforme **
if NBCT ~= 0
  TRC = TPRES(1:NBCT) / E;
  NODE = 24 * (NELM(1:NBCT) - 1) + 3 * (jt_local(1:NBCT) - 1) + ICOND(1:NBCT);
  BT = g_global(:,NODE) * TRC';

%** condições de contorno de deslocamento prescritas **
%  nos_desl = NOEL(MNEL,:);
  bloco = [1:1:8];
  nos_desl = bloco;
  for idx = 1:NBCU
    elem(bloco) = MNEL(idx);
    nos(bloco) = nos_desl;
    mcond(bloco) = MCOND(idx);
    bloco = bloco + 8;
  end;
  NODE = 24 * (elem - 1) + 3 * (nos - 1) + mcond;
  AO(:,NODE) = -g_global(:,NODE);
else

%** condições de contorno de deslocamento prescritas **
  bloco = [1:1:8];
  nos_desl = bloco;
  for idx = 1:NBCU
    elem(bloco) = MNEL(idx);
    nos(bloco) = nos_desl;
    mcond(bloco) = MCOND(idx);
    bloco = bloco + 8;
  end;
  NODE = 24 * (elem - 1) + 3 * (nos - 1) + mcond;
  AO(:,NODE) = -g_global(:,NODE);
end;
```

Listagem da função **TENSNODAL.m**

```
function [SigNodal] = TensNodal(Tracao,Desloc,NEL,NNP,NOEL,X,Y,Z,E,NU)

%** função TensNodal calcula resposta secundária de tensão nodal através de
tensores **

%** inicialização de variáveis utilizadas em funções posteriores **
DG = 0;
DH = 0;
SF = 0;
SDG = 0;
SDH = 0;
SSF = 0;
SigNodal = zeros(NNP,10);

%** inicialização de vetores com valores prescritos de G e H **
GNODE = [-1, 0, 1, 1, 1, 0,-1,-1];
HNODE = [-1,-1,-1, 0, 1, 1, 1, 0];

for elem = 1:NEL
  [CORD] = COORD(elem,NOEL,X,Y,Z);
  for no = 1:8
%** vetor normal unitário num nó do elemento **
    G = GNODE(no);
    H = HNODE(no);
    [DE] = SHAP8(0,0,1,G,H,1,DG,DH,SF);
    [UN] = JACOBI(1,1,DE,DG,DH,CORD);
%** vetor tangente ao contorno num nó do elemento **
    tal(1,1) = sum(DE(1,:) * CORD(:,1));
    tal(1,2) = sum(DE(1,:) * CORD(:,2));
    tal(1,3) = sum(DE(1,:) * CORD(:,3));
    tal  =  tal  /  realsqrt(realpow(tal(1,1),2)  +  realpow(tal(1,2),2)  +
realpow(tal(1,3),2));
%** vetor perpendicular a UN e tal **
    b = cross(UN,tal);
%** matriz de transformação de vetores de XYZ para Zi **
    lij(1,:) = tal;
    lij(2,:) = b;
    lij(3,:) = UN;
%** transformação das coordenadas e deslocamentos nodais de XYZ para Zi **
  for ic = 1:8
    CORDZ(ic,:) = permute(lij * permute(CORD(ic,:),[2 1 3]),[2 1 3]);
    DeslocZ(ic,:) = permute(lij * permute(Desloc(NOEL(elem,ic),:),[2 1 3]),[2
1 3]);
  end;
%** transformação das trações nodais de XYZ para Zi **
    TracaoZ(1,:) = permute(lij * permute(Tracao((8*(elem-1)+no),:),[2 1 3]),[2
1 3]);
%** matriz jacobiana **
    J = [(DE(1,:) * CORDZ(:,1)),(DE(1,:) * CORDZ(:,2));
         (DE(2,:) * CORDZ(:,1)),(DE(2,:) * CORDZ(:,2))];
%** derivada do deslocamento em coordenadas locais em relação a G **
    dvda(1,1) = DE(1,:) * DeslocZ(:,1);
    dvda(1,2) = DE(1,:) * DeslocZ(:,2);
    dvda(2,1) = DE(2,:) * DeslocZ(:,1);
    dvda(2,2) = DE(2,:) * DeslocZ(:,2);
%** derivada do deslocamento em coordenadas locais em relação a z **
```

```
    dvdz = inv(J) * dvda;
%** cálculo das tensões em coordenadas locais **
    mi = E / (2 * (1 + NU));
    lambda = NU * E / ((1 + NU) * (1 - 2 * NU));
    s(1,1)    =    1    /    (2*mi+lambda)    *    (4*mi*(mi+lambda)*dvdz(1,1)    +
2*mi*lambda*dvdz(2,2) + lambda*TracaoZ(1,3));
    s(2,2)    =    1    /    (2*mi+lambda)    *    (4*mi*(mi+lambda)*dvdz(2,2)    +
2*mi*lambda*dvdz(1,1) + lambda*TracaoZ(1,3));
    s(1,2) = mi * (dvdz(1,2) + dvdz(2,1));
    s(3,1) = TracaoZ(1,1);
    s(3,2) = TracaoZ(1,2);
    s(3,3) = TracaoZ(1,3);
    s(2,1) = s(1,2);
    s(1,3) = s(3,1);
    s(2,3) = s(3,2);
    sig = lij' * s * lij;
    SigNodal(NOEL(elem,no),1:3) = SigNodal(NOEL(elem,no),1:3) + sig(1,:);
    SigNodal(NOEL(elem,no),4:6) = SigNodal(NOEL(elem,no),4:6) + sig(2,:);
    SigNodal(NOEL(elem,no),7:9) = SigNodal(NOEL(elem,no),7:9) + sig(3,:);
    SigNodal(NOEL(elem,no),10) = SigNodal(NOEL(elem,no),10) + 1;
  end;
%** tranformação do tensor de tensões do sistema Z para XYZ **
end;
for i = 1:NNP
  SigNodal(i,1:9) = SigNodal(i,1:9) / SigNodal(i,10);
end;
```

Listagem da função **SOLIDPOS.m**

```
function solidpos(problema,CaminhoProblema,quantity,dir,giro,elev,fator)
% Função para visualização sólida de posprocessamento
%
% sintaxe: solidpos('arquivo de resultados','caminho do arquivo de dados',
%          quantity,dir,giro,elev,fator)
%
% Desenvolvida por Pedro Yoshito Noritomi
% Revisão: 20/12/1999

fidin = fopen(strcat(CaminhoProblema,'Props.txt'));
fseek(fidin,-1,'eof');
fim_arq = ftell(fidin) + 1;
frewind(fidin);
while ftell(fidin) < fim_arq
  MAXL = str2num(fgets(fidin));
  E = str2num(fgets(fidin));
  NU = str2num(fgets(fidin));
  IGAUS = str2num(fgets(fidin));
end;

%** Entrada de coordenadas dos pontos nodais **
%** Lê o arquivo CoordNos.txt e cria a matriz CoordNos(NNP,3) **
fidin1 = fopen(strcat(CaminhoProblema,'CoordNos.txt'));
nn = 1;
fseek(fidin1,-1,'eof');
fim_arq1 = ftell(fidin1) + 1;
frewind(fidin1);
```

```
while ftell(fidin1) < fim_arq1
  CoordNos1 = str2num(fgets(fidin1));
  CoordNos(nn,:) = CoordNos1;
  nn = nn + 1;
end;
NNP = size(CoordNos,1);          %quantidade de nós do problema

%** fator de multiplicação para fins de visualização da geometria deformada **
%fator = 3; definido por parâmetro de entrada da função

%** leitura do arquivo de resultados **
load(strcat(CaminhoProblema,problema));
CoordNos = CoordNos + Desloc * fator;

for ii = 1:NNP
  X(ii) = CoordNos(ii,1);
  Y(ii) = CoordNos(ii,2);
  Z(ii) = CoordNos(ii,3);
end;

%** Entrada da matriz de conectividade nodal dos elementos **
fidin2 = fopen(strcat(CaminhoProblema,'NOEL.txt'));
nn = 1;
fseek(fidin2,-1,'eof');
fim_arq2 = ftell(fidin2) + 1;
frewind(fidin2);
while ftell(fidin2) < fim_arq2
  NOEL1 = str2num(fgets(fidin2));
  NOEL(nn,:) = NOEL1;
  nn = nn + 1;
end;
NEL = size(NOEL,1);              %quantidade de elementos da malha

figure

%** inicialização de variáveis **
r = 10 * MAXL;

giro = giro * pi / 180;
elev = elev * pi / 180;

x_obs = r * sin(giro);
y_obs = -r * cos(giro);
z_obs = r * sin(elev);

%** determinação da sequência de preenchimento dos elementos **
for i = 1:NEL
  xp_elem = 0;
  yp_elem = 0;
  zp_elem = 0;
  for n = 1:8
    xp_elem = xp_elem + X(NOEL(i,n));
    yp_elem = yp_elem + Y(NOEL(i,n));
    zp_elem = zp_elem + Z(NOEL(i,n));
  end;
  x_elem = xp_elem / 8;
  y_elem = yp_elem / 8;
```

```
  z_elem = zp_elem / 8;
  pos_elem(i) = sqrt((x_elem - x_obs)^2 + (y_elem - y_obs)^2 + (z_elem -
z_obs)^2);
end;
[dist,seq_elem] = sort(pos_elem);
for j = 1:NEL
  e = seq_elem(NEL-j+1);
  for n = 1:8
    x_fill(n) = X(NOEL(e,n));
    y_fill(n) = Y(NOEL(e,n));
    z_fill(n) = Z(NOEL(e,n));
    cor(n) = quantity(NOEL(e,n),dir);
  end;
  fill3(x_fill,y_fill,z_fill,cor);
  hold on;
end;
grid;
xmin = min(X) - 0.05 * MAXL;
xmax = max(X) + 0.05 * MAXL;
ymin = min(Y) - 0.05 * MAXL;
ymax = max(Y) + 0.05 * MAXL;
zmin = min(Z) - 0.05 * MAXL;
zmax = max(Z) + 0.05 * MAXL;
axis([xmin xmax ymin ymax zmin zmax]);
axis equal;
view([x_obs y_obs z_obs]);
colorbar('horizontal');
hold off;
drawnow;
fclose('all');
```