UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA CIVIL

ANÁLISE TRIDIMENSIONAL DE EDIFÍCIOS POR ELEMENTOS FINITOS UTILIZANDO PROGRAMAÇÃO ORIENTADA A OBJETOS

Leonardo Slhessarenko Filho

Orientador: **Prof. Dr. Francisco A. Menezes**

DISSERTAÇÃO DE MESTRADO

Campinas-SP, Brasil
1997

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA CIVIL

ANÁLISE TRIDIMENSIONAL DE EDIFÍCIOS POR ELEMENTOS FINITOS UTILIZANDO PROGRAMAÇÃO ORIENTADA A OBJETOS

Leonardo Slhessarenko Filho
Orientador:**Prof. Dr. Francisco A. Menezes**

Dissertação de mestrado apresentada à Faculdade de Engenharia Civil como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Civil.
Área de Concentração: Estruturas

Campinas-SP, Brasil
1997

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

SL29a

Slhessarenko Filho, Leonardo
Análise tridimensional de edificios por elementos finitos utilizando a programação orientada à objetos. / Leonardo Slhessarenko Filho.--Campinas, SP: [s.n.], 1997.

Orientador: Francisco Antonio Menezes
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Civil.

1. Teoria das estruturas. 2. Método dos elementos finitos 3. Programação orientada a objetos. 4. Placas e cascas elásticas. 5. Barras (Engenharia). I. Menezes, Francisco Antonio. II. Universidade Estadual de Campinas. Faculdade de Engenharia Civil. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA CIVIL

ANÁLISE TRIDIMENSIONAL DE EDIFÍCIOS POR ELEMENTOS FINITOS UTILIZANDO PROGRAMAÇÃO ORIENTADA A OBJETOS

Leonardo Slhessarenko Filho.

Dissertação de Mestrado defendida e aprovada, em 15 de dezembro de 1997, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Francisco Antonio Menezes, presidente
FEC - UNICAMP

Prof. Dr. Aloísio Ernesto Assan
FEC – UNICAMP

Prof. Dr. Renato Pavanello
FEM - UNICAMP

ANÁLISE TRIDIMENSIONAL DE EDIFÍCIOS POR ELEMENTOS FINITOS UTILIZANDO PROGRAMAÇÃO ORIENTADA A OBJETOS

Leonardo Slhessarenko Filho
RA 956349

Faculdade de Engenharia Civil
UNICAMP

Aos meus pais, Leonardo e Serys, como prova do meu amor e símbolo da minha eterna gratidão.

Agradecimentos

Ao meu pai Leonardo Slhessarenko e à minha mãe Serys, por todo o apoio, carinho, amor e correções durante a infância e adolescência, enfim por fazerem parte da minha vida.

Aos meus avós João e Olinda, pelo exemplo de vontade de viver frente às dificuldades da idade.

Aos sobrinhos João Paulo e Marina que são para mim motivo de muita alegria e felicidade.

A minha irmã Natasha e ao seu esposo Roberto, que tanto ajudaram deram conselhos e tornaram as tardes de domingo mais alegres.

A minha irmã Larissa e ao seu esposo Paulo, pelo carinho e exemplo de serenidade e alegria.

Ao meu irmão Alexandre e sua esposa Amanda pelo carinho e atenção recebido.

A minha querida Lara pela sua compreensão, pelo seu amor, carinho e estímulos capazes de ultrapassar o obstáculo da distância durante todo o período de minha ausência.

A dona Nize e Luiz Pereira pelo apoio, crença e estímulo recebido.

Aos meus tios, tias, primos e primas pelo incentivo e compreensão.

Aos amigos Lenildo (UnB), Milton (UnB) e Antônio Teixeira (Unicamp) pela amizade e companheirismo.

Ao professor Francisco Antonio Menezes pela orientação constante e dedicada, pela paciência e amizade demonstrada, sem dúvida imprescindíveis para a concretização deste trabalho.

Aos professores Philippe Devloo e Aloísio Assan pela orientação, apoio e presteza durante todo o curso.

A todos do CENAPAD-SP, por toda a colaboração e amparo e acima de tudo pela nossa grande amizade.

Aos professores, colegas e funcionários do Mestrado em Estruturas da FEC-Unicamp, pelo apoio e amizade durante todo o curso.

E sobretudo ao meu Deus, por ter me protegido e iluminado todos os dias e em todos os momentos.

Índice

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho trata da análise de edifícios através de uma técnica discreta, baseada no método dos elementos finitos via processo dos deslocamentos, onde a estrutura é idealizada como uma associação tridimensional de vigas, lajes e pilares. Elaborou-se a partir de um ambiente computacional denominado PZ, sob o paradigma da orientação a objetos, uma série de rotinas, "classes" e "métodos" que permitem o cálculo de deformações e esforços na análise tridimensional de estruturas. O ambiente de computação científica PZ, escrito em linguagem de programação C++, é uma ferramenta computacional de múltiplos propósitos, preparado para a simulação, via método dos elementos finitos, de problemas matemáticos e de engenharia onde o fenômeno a ser analisado pode ser representado através de um conjunto de equações diferenciais parciais. Nesse ambiente, usando o conceito de hierarquia de classes, todo o código pôde ser reutilizado. Assim, aproveitando a funcionalidade do código existente, para análise do tipo elasticidade tridimensional para relações tensão/deformação, desenvolveram-se as formulações variacionais que expressam as deformações em elementos de barra e placa e implementou-se estas em classes de materiais específicos para o cálculo de "vigas ", "pilares " e de "placas ", derivadas de uma classe abstrata existente no PZ, chamada TMaterial. O elemento de barra foi modelado levando em conta a Teoria de Timoshenko e o elemento de placa foi modelado incorporando a Teoria de Reissner-Mindlin, onde associou-se também à formulação o efeito de membrana. Dada a generalidade das formulações, tanto o elemento de barra quanto o elemento de placa podem assumir geometrias arbitrárias (parabólicas, cilíndricas). O elemento de barra foi concebido com seis graus de liberdade por nó para permitir o perfeito acoplamento com o elemento de placa, também implementado com seis variáveis independentes por nó: três translações e três rotações. Foi elaborado um pré-processador baseado na entrada de dados do programa Ansys. Para a visualização dos resultados foram implementado no ambiente PZ alguns "métodos " onde se introduziu o conceito de "elemento gráfico ", através do qual se preparam arquivos de saída de resultados adequado tal que possam ser interpretados por programas de visualização gráfica. Através de implementações computacionais apresentam-se exemplos numéricos cujos resultados foram comparados com o software Ansys, versão 5.2, mostrando a validade das formulações desenvolvidas.

Abstract

This thesis is concerned with the analysis of buildings using a discrete technique based on the finite element method, via process of displacement, where the structure is idealized as a three dimensional association of beams, slabs and columns.

The slabs are discretized with plate finite elements of four nodes. The columns and beams are modelled by unidimensional finite elements submitted to flexure.

Based on the paradigm of the orientation to objects and using the computational environment PZ, a series of routines were created: "classes" and "methods", which permitted the evaluation of deformations and strains in the three dimensional analysis of structures. The scientific computational environment PZ, written in the C++ programming language, is a multiple purpose computational tool prepared for the simulation, via the finite element method, of mathematical and engineering problems, where the phenomenon to be analysed can be represented by a system of partial differential equations. In such environment, using the concept of class hierarchy, variational formulations that express the strains in bar and plate elements by the generation of classes derived from "materials" were implemented.

The classes named TTimoshenko and TPlateReissner derived from an abstract class named TMaterial existing in the PZ environment were implemented.

The bar and plate elements were modelled taking into account the Theory of Timoshenko and the Theory of Reissner-Mindlin, respectively, with the membrane effect taken into consideration in the formulation.membrane effect was also associated in the formulation.

Considering the generality of the formulation, both the bar and plate elements can assume arbitrary geometric shapes.

The bar element was conceived with six nodal degrees of freedom in order to allow the coupling with the plate element, which also has six nodal degrees of freedom: three translations and three rotations.

A pre-processor based on input files created by the programme Ansys was prepared. To the visualization of the results some "methods" allow the the preparation of files containing the answers of the analysed structure that are suitable to be interpreted by softwares of graphic visualization.

The examples presented to confirm the validation of the present computational implementation where always compared with the results obtained with similar simulations realized by the software Ansys.

Capítulo 1

Introdução

A maioria dos fenômenos da natureza, seja nos ramos da biologia, geologia ou da engenharia podem ser expressos de maneira aproximada com o auxílio das leis da física e da matemática em termos algébricos, diferenciais ou através de equações integrais. Determinar a distribuição de tensões em um recipiente pressurizado sujeito à cargas mecânicas, térmicas e/ou aerodinâmicas, encontrar a taxa de concentração de poluentes nas águas ou na atmosfera, simular o clima na tentativa de entender e prognosticar o mecanismo de formação dos tornados e tormentas, são alguns exemplos dos muitos problemas práticos com os quais nos deparamos a todo instante e que podem ser representados através de equações diferenciais.

Conforme Timoshenko[33], o estudo de fenômenos físicos voltados para a engenharia estrutural teve seu início no final do século XVII, porém sofreu seu maior desenvolvimento em meados do século XX com a introdução de novas técnicas de cálculo e surgimento dos computadores digitais.

Engenheiros e cientistas que se dedicam ao estudo de fenômenos físicos estão empenhados em duas principais tarefas:

- formulação matemática dos processos físicos;
- análise numérica do modelo matemático.

Segundo Przermieniecki[24], os métodos de análise estrutural podem ser classificados em analíticos e numéricos. Apesar de apresentarem soluções fechadas, de grande utilidade, os métodos analíticos são muito limitados quando se deseja resolver problemas maiores e mais complexos.

No intuito de se buscar respostas para tais problemas recorre-se, na maioria das vezes, a formulações matemáticas que são representadas pelas equações diferenciais que expressam esses processos físicos. Com o advento dos computadores houve um grande avanço dos métodos numéricos, e modelos matemáticos puderam ser analisados com maior rapidez e confiabilidade. Cálculos que antes eram impossíveis de se realizar, devido a limitações da tecnologia dos computadores, tornaram-se realidade; teorias que envolviam sólidos tridimensionais sofreram grandes evoluções e aos poucos foram sendo desmistificados.

O uso do computador na engenharia, seja para apoio na administração, planejamento, controle ou análise de projetos, entre outros, possibilitou aos engenheiros, projetistas e pessoas envolvidas nos projetos avaliar e manipular as informações mais rapidamente, planejá-las, estimar seus impactos e com isso viabilizar a tomada de decisões com aceitáveis margens de erro.

Para o estudo desses projetos há a necessidade de se construir modelos a fim de permitir a simulação física do comportamento do sistema. Projetos estruturais exigem um certo grau de complexidade para sua completa elaboração e análise. Esses projetos apoiam-se na mecânica dos sólidos, que é a ciência que permite descrever o comportamento desses sistemas físicos. Assim, os problemas que podem ser representados por um número finito de componentes são ditos discretos, podendo ser solucionados por computador; os problemas definidos, usando-se elementos matemáticos infinitesimais, chamam-se contínuos e apenas podem ser resolvidos por intervenções matemáticas que conduzem às equações diferenciais, as quais nem sempre apresentam solução viável.

Para resolver problemas contínuos utilizam-se vários métodos de discretização, os quais envolvem aproximações. Entre esses métodos numéricos aproximados os que apresentaram maior evolução foram os métodos matriciais, com maior destaque para o método dos elementos finitos baseados em deslocamentos, cuja implantação se deve a Zienkiewicz[38], Szilard[32] e Argyris[2] no período de 1955 a 1960. Esta tem sido ao longo dos anos uma das técnicas mais utilizadas para resolver problemas na engenharia estrutural, mecânica dos fluidos e transferência de calor, podendo ser aplicado também a qualquer outro problema que se traduza na solução de um conjunto de equações diferenciais. Seu sucesso é devido ao fato de possuir um procedimento relativamente simples e disciplinado para sua solução. Inicialmente faz-se a formulação do problema; em seguida a discretização do domínio do problema a ser analisado, e por fim a solução efetiva do sistema de equações resultante,

permitindo a obtenção dos resultados numa etapa de pós-processamento.

A principal característica do método dos elementos finitos é que os passos são sempre os mesmos qualquer que seja o problema considerado, o que proporciona ao método grande aceitação no trato dos diversos tipos de análises. Deste modo, o método dos elementos finitos pode ser visto como uma poderosa ferramenta numérica na obtenção de soluções aproximadas para modelos contínuos. Sendo suas bases matemáticas bem postas, este método é atualmente o mais amplamente utilizado em problemas de análise estrutural e da mecânica do meio contínuo em geral.

Aliado a esta ferramenta matemática encontra-se a programação orientada a objetos, em que a principal característica é o reaproveitamento de códigos já desenvolvidos. Enquanto hoje cada vez mais *softwares* utilizam a filosofia de programação orientada a objetos, o desenvolvimento dominante na computação científica ainda permanece baseada na programação estruturada e seqüencial. Isto se deve a vários fatores, tais como:

1. mudança de linguagem de programação exige um investimento que pode levar anos;

2. a programação orientada a objetos pode ser modular, mas desenvolver esses módulos de forma coerente e bem pensada leva muito tempo. Esse conceito novo de programação, sem dúvida, é mais complexo do que a programação estruturada. Com isso, gasta-se mais tempo para ser bem utilizada e tornar-se efetivamente mais produtiva do que a programação estruturada e seqüencial;

3. muitos cientistas têm por cultura usar a programação convencional meramente para verificar os algoritmos que desenvolveram sobre suas bases teóricas, inviabilizando o reaproveitamento do código.

Do ponto de vista do desenvolvimento de pesquisa em grupo envolvendo *software*, é de fundamental importância que o código fonte a ser gerado por um programador seja facilmente entendido e reutilizado por outros em novas aplicações, o que justifica o uso da programação orientada a objetos nesse trabalho.

1.1 Escopo do trabalho

Neste trabalho são apresentadas as teorias para barras e placas de seção transversal indeformável, geometricamente lineares, formuladas com base nos fundamentos da Mecânica dos Sólidos Deformáveis, com vistas ao seu emprego na modelagem de edifícios tridimensionais através do método dos elementos finitos. Muitos aspectos da teoria linear para barras e placas, principalmente por serem derivados das simplificações impostas na teoria de sólidos tridimensionais, serão abordadas no transcorrer do trabalho pois são fundamentais para esclarecer as formulações utilizadas.

As variáveis cinemáticas do problema do edifício, modelado pela associação de placas, pilares e vigas são os deslocamentos de cada nó: 3 de translação e 3 de rotação.

Apresenta-se ainda no trabalho uma extensão dos problemas da viga e placa/casca convencionais para eixos com qualquer desenvolvimento geométrico no espaço tridimensional.

Embora o método dos elementos finitos esteja bastante evoluído no que diz respeito aos meios contínuos (chapas, placas, cascas, estruturas maciças em geral), tem sido relativamente pequena a atenção dedicada ao estudo de sistemas que envolvam elementos espaciais de barras e placas curvas. Ressalta-se que este não é o propósito deste trabalho, tendo essas extensões apenas contribuído para o enriquecimento e generalização das formulações aqui apresentadas. No cálculo das edificações esses elementos serão acoplados, possibilitando, com isso, uma aproximação mais confiável do comportamento estrutural de edifícios.

Partindo de um ambiente de programação totalmente orientado a objetos, denominado **PZ**[1] (Devloo[12]), baseado na linguagem C++, foram desenvolvidos alguns *métodos* para a solução de problemas de elasticidade elástico linear, e duas *classes* de materiais para o tratamento do problema de barras e placas no espaço. Esses *métodos* e *classes* foram incorporados ao ambiente **PZ** que, segundo Devloo[11], oferece ao usuário diversas facilidades para implementação e uma considerável estrutura de *classes* com possibilidades de expansão.

No capítulo 2 são descritos os conceitos necessários para a utilização da filosofia da linguagem orientada a objetos, em que termos reservados a essa

[1] O ambiente PZ encontra-se disponível no Departamento de Estruturas da Faculdade de Engenharia Civil da Unicamp

metodologia são definidos.

No desenvolvimento do presente trabalho foi elaborado um programa computacional com o auxílio da plataforma de programação **PZ,** detalhada no capítulo 3, em que aspectos técnicos desse ambiente, com base no suporte teórico adquirido no capítulo anterior, poderão ser melhor elucidados.

Devido à flexibilidade de uma linguagem orientada a objetos, como é o caso de C++ e ainda a portabilidade do **PZ**, segundo Devloo[12], pode ser utilizada tanto em arquiteturas computacionais do tipo PC como em estações de trabalho baseadas em sistemas UNIX, disponibilizando com isso seu uso em plataformas distintas de computadores.

No capítulo 4 apresentam-se as convenções, hipóteses e conceitos da teoria da elasticidade e princípio dos trabalhos virtuais. São apresentadas inicialmente duas teorias para barras tridimensionais geometricamente lineares, considerando duas hipóteses cinemáticas aqui denominadas por *Teoria de Bernoulli-Euler* e *Teoria de Timoshenko.* Após o tratamento das respectivas formulações variacionais faz-se a análise destas formulações com aplicação a elementos finitos. Encerra-se o capítulo com desenvolvimento da teoria de barras de *Timoshenko* estendida para a análise de vigas curvas e posterior apresentação dos jacobianos unidimensionais para transformação de coordenadas cartesianas e cilíndricas em coordenadas adimensionais do elemento mestre.

O capítulo 5 é reservado para as formulações variacionais dos elementos de placa e posteriormente para aproximação de cascas. As formulações são detalhadas partindo-se do *Princípio dos Trabalhos Virtuais*, no qual busca-se a mínima energia para a formulação de placa de *Kirchoff* com três graus de liberdade por nó, e na seqüência utiliza-se o mesmo princípio para encontrar a formulação que exprime a mínima energia para o problema da placa com as considerações de *Reissner-Mindlin* para seis graus de liberdade por nó.

Faz-se a discretização das formulações a elementos finitos para as teorias de *Kirchoff e Reissner-Mindlin* após a apresentação das respectivas formulações variacionais. Finaliza-se o capítulo com o desenvolvimento da *Teoria de Placas de Reissner-Mindlin* estendida para a análise de cascas curvas e posterior apresentação do jacobiano bidimensional para transformação de coordenadas cartesianas.

No capítulo 6 é apresentado o acoplamento entre os elementos finitos de barra e de placa, permitindo a análise de edifícios. Mostra-se como obter a matriz de rigidez global da estrutura a ser analisada, bem como o tratamento dado ao problema da imposição das condições de contorno e formulação

do vetor de ações nodais equivalentes. Mostra-se também como calcular os esforços solicitantes.

No capítulo 7 são mostrados alguns exemplos de associação de elementos de barras e placas e alguns exemplos de esforços em edifícios. Os resultados dos esforços nos exemplos foram comparados com cálculo similar efetuado através do programa *Ansys*[2][1], versão 5.2. Os exemplos apresentados têm a finalidade única de validar a formulação proposta.

Finalmente, no capítulo 8, são apresentadas as conclusões finais do trabalho, enfocando os assuntos mencionados anteriormente e possíveis projeções ou sugestões de continuidade que o presente trabalho possa suscitar.

A contribuição deste trabalho consiste principalmente no tratamento de elementos de vigas e de placas com a incorporação da deformação por cortante e associação dos dados dos eixos locais dos elementos como parâmetros adicionais dos mesmos. Com isso, a formulação das matrizes de rigidez e vetores de carga dos elementos já foram calculados em termos dos eixos globais, evitando-se o que classicamente é feito em situações semelhantes, ou seja: o cálculo de matrizes de rigidez e vetores de carga dos elementos no sistema local, formulação de matrizes de rotação, e finalmente a obtenção de matrizes de rigidez e vetores de carga expressos nas direções das coordenadas globais a partir das matrizes de rotação e das matrizes de rigidez e vetores de cargas expressos nas coordenadas locais.

Ressalta-se também o enriquecimento do ambiente PZ que passou a incorporar a possibilidade de análise de estruturas de barras, placas e cascas.

1.2 Tarefas principais

As principais tarefas no desenvolvimento desta dissertação foram:

a) implementar inicialmente duas teorias para barras tridimensionais geometricamente lineares, com o objetivo de elucidar os principais conceitos envolvidos nas diferentes hipóteses cinemáticas para as suas formulações;

b) em seguida, implementar duas teorias para placas tridimensionais geometricamente lineares, com o objetivo de esclarecer os conceitos que

[2]No programa de elementos finitos *Ansys* foram usados os elementos *BEAM4* e *SHELL63* existentes em sua biblioteca interna de elementos.

norteiam as diferentes hipóteses cinemáticas utilizadas nas duas teorias de placas envolvidas neste trabalho;

c) utilizar a programação orientada a objetos e reutilização de códigos de elementos finitos para a solução de problemas da engenharia estrutural, através do acoplamento de elementos de barra com elementos de placas;

d) elaborar um programa para cuidar da entrada de dados semelhante à entrada de dados do programa *Ansys*, versão 5.2;

e) propor e analisar vários exemplos numéricos visando validar a implementação computacional proposta;

f) interagir com o software de análise estrutural *Ansys* release 5.2 com vistas a facilitar o pré-processamento dos dados e aferição de resultados;

g) calcular o sistema de equações através de algum método direto existente na classe de matrizes do ambiente PZ;

h) calcular as deformações a partir dos deslocamentos;

i) obter os esforços solicitantes nos elementos estruturais a partir das deformações encontradas;

g) visualizar gráficamente os resultados.

Observa-se que nesse trabalho, o autor não se preocupou com análise de tensões.

A saída de resultados pode ser visualizada graficamente através de dois pós-processadores: o programa DataExplorer da IBM para estações de trabalho e o programa *MView* desenvolvido na PUC/RJ para ambiente Windows.

Os exemplos ilustrados neste trabalho foram concebidos com o uso dos recursos computacionais do CENAPAD-SP[3], em estações de trabalho do tipo RISC/6000 e, em alguns casos, utilizando nós do IBM-SP existente nesse ambiente.

[3]Centro Nacional de Processamento de Alto Desempenho em São Paulo (FINEP/MCT/UNICAMP)

Capítulo 2

Programação orientada a objetos

No desenvolvimento de técnicas computacionais para a obtenção da solução de problemas de análise de estruturas, as dificuldades mais comuns referem-se à elaboração de entrada de dados, processamento e visualização dos resultados. A facilidade de interação entre o usuário e a máquina durante o processo de expansão e implementação de novas potencialidades aos programas resultantes é um outro desafio que vem recebendo cada vez mais importância, e é sem dúvida a chave do sucesso para expandir códigos cada vez mais complexos e robustos.

Os avanços tecnológicos obtidos nos últimos anos, evidenciados pelo aumento da capacidade de memória e rapidez de execução dos cálculos, bem como a redução dos preços dos computadores, trouxe um desenvolvimento bastante acentuado no que se refere às linguagens de programação. Essa mudança na forma de se manipular dados para computadores teve seu marco diferencial dado pela criação de linguagens orientadas a objetos, pautadas por um conceito diferente de programação unindo dados e sub-rotinas.

As linguagens tradicionais interpretam os dados como entidades passivas, as quais permitem que de qualquer parte do código manipule-se tais dados através de "*procedures*" e "*functions*", Scholz[29], facilitando a ocorrência de erros, dificultando a manutenção dos programas e a utilização de trechos do código no desenvolvimento do programa por terceiros ou em outro projeto que possa dar continuidade com reutilização do código.

Na programação estruturada, dados e subrotinas são mantidos separados, unindo-se apenas quando aqueles são passados para estas nas suas chamadas

no corpo do código; as linhas do código são projetadas em torno das funções. Assim, estas são mais importantes nesse tipo de programação que naquela orientada a objetos, os quais são o núcleo em torno do qual as mensagens são trocadas entre os códigos que compõem um programa.

A orientação a objetos utiliza-se do mesmo conceito de estruturação dos dados dentro de um único tipo e o amplia com a inclusão de sub-rotinas, mantendo-os (dados e sub-rotinas) unidos de forma a modificar esse conceito, o qual convencionou-se chamar de *classe*. As funções-membro de uma *classe* também conhecidas como métodos, definem o que pode ou não ser feito com os dados da classe. Dessa forma, os dados ficam encapsulados de modo que apenas os métodos do objeto consigam acessá-los, ou seja, um objeto opera sobre seus próprios dados através de seus próprios métodos, de modo que funções externas à classe do objeto não consigam manipular suas variáveis.

Uma vez que em um sistema, objetos armazenam tipos específicos de informação e operações específicas, o primeiro passo na criação de um programa baseado em objetos é identificar os que fazem parte do sistema, as informações principais e as operações realizadas nos mesmos.

A programação orientada a objetos é uma técnica relativamente nova para a concepção e implantação de sistemas de software, e centraliza-se nos principais conceitos: tipos de dados abstratos e classes, hierarquias de tipos (sub-classes), herança e polimorfismo. Acredita-se que o ponto marcante da programação orientada para objetos seja a possibilidade de se reutilizar código, ou seja, um objeto pode usar as implementações de um outro objeto e estendê-las para alcançar um objetivo específico.

Embora o ambiente PZ apresente a possibilidade de reutilização de código, talvez por falta de decumentação no caso do presente trabalho, a implementação do código foi bastante árdua.

Quando se trabalha com orientação a objetos, todos eles são organizados em classes. São definidas classes básicas e estas são estendidas de forma a criar novas classes, o que torna o software orientado para objeto muito mais fácil de ser documentado e entendido. O principal objetivo da programação orientada a objetos é aumentar a produtividade do programador através de uma maior reutilização do código, além de diminuir a complexidade e o custo da manutenção do mesmo. Para um bom entendimento do trabalho pode-se adquirir maiores conhecimentos sobre programação orientada a objetos em Wiener & Pinson[37].

2.1 Principais elementos da programação orientada a objetos

2.1.1 Classe

Segundo Pappas & Murray[22], *classe* é um novo tipo de dado, definido pelo usuário, como os que existem pré-definidos em compiladores de diversas linguagens de programação. Uma variável de uma classe é chamada objeto e conterá campos de dados e funções. Dessa forma, tira-se vantagem da estruturação de dados em uma variável e a propriedade de limitar o acesso a esses dados específicos a funções próprias. Assim, as classes definem as propriedades e os atributos que descrevem as ações de um objeto, servindo como um padrão para a criação de objetos, que também são chamados de instâncias da classe.

Os dados e funções de uma classe estão localizados geralmente em uma área de acesso proibido para funções que sejam declaradas externamente à classe, denominada área *private* (privada) ou ainda *protected* (protegida), conforme o nível de acesso que se deseja empregar à classe. Um dado (variável) ou uma função podem ainda ser *public* (público), isto é, podem ser acessados de qualquer parte do programa, porém sempre associados a um objeto da classe à qual eles pertencem.

Definir uma classe não cria nenhum objeto, do mesmo modo que a existência do tipo *int* não cria nenhuma variável. A declaração de uma classe começa com a definição de uma estrutura de dados e funções na área *protected*, *private* ou ainda *public*. Entre as funções, geralmente, encontram-se duas especiais: a função construtora chamada automaticamente sempre que um objeto do tipo da classe for declarado, é responsável por inicializar as variáveis do objeto (instância), e a função destrutora, chamada sempre que um objeto alcança o final do escopo em que foi declarado ou, ainda, quando a manutenção do objeto em memória alocado dinamicamente não for mais necessária. Costuma-se implementar as funções (métodos) em arquivo separado da sua declaração, podendo-se implementá-lo no mesmo arquivo. Para o caso de desenvolver o código das funções em arquivos separados daqueles que contêm as declarações, associam-se os arquivos através de *includes* nos arquivos das declarações também chamados arquivos de cabeçalho.

Em C++ pode-se ainda implementar o código das funções na mesma linha da sua declaração; nesse caso, segundo Weiskamp[36], ao encontrar

uma implementação desse tipo, o compilador insere o corpo da função na seqüência de sua declaração, o que torna desnecessária a chamada de função pelo programa. Com isso, não ocorre o desperdício de tempo com a chamada da função - conhecido como "*overhead*", diminuindo o tempo total gasto para executar um programa.

2.1.2 Objetos

A programação orientada por objetos é baseada na escrita de programas em termos de objetos (coisas) que compõem o sistema. Como visto, um objeto representa uma entidade do mundo real que pode armazenar dados (variáveis-membros) e possui um conjunto específico de operações (funções-membro) que são realizadas nele, ou ainda, é um conjunto de dados e procedimentos para trabalhar com esses dados.

2.1.3 Mensagens

Objetos só são úteis quando associados a outros objetos para alcançar um resultado. A troca de informações entre objetos é feita através das mensagens. A mesma mensagem pode ser enviada a objetos diferentes que devido a seus métodos próprios para tratamento dessa mensagem poderá responder de forma diferente, determinando um comportamento denominado polimorfismo.

Uma mensagem possui basicamente três componentes:

- objeto para receber a mensagem;
- nome da operação (método) a ser executada pelo objeto;
- parâmetro (argumento).

2.1.4 Métodos

Como mencionado anteriormente, ao declarar uma classe pode-se definir dados e funções como *private*, *protected* ou *public*. A forma mais segura de se manipular os dados de uma classe é através de suas funções (métodos). Para que um método possa estar disponível em qualquer ponto do programa é necessário que ele seja definido como *public*. Os métodos *private* da classe,

por outro lado, somente podem ser acessados utilizando-se os próprios métodos da classe. Definindo-se métodos privados da classe, o programa poderá controlar os valores atribuídos aos próprios métodos da classe e a forma como eles são utilizados. Existem ainda métodos com acesso *protected* que podem ser acessados pela classe a que pertence e também pelas classes derivadas da mesma.

2.2 Principais características da programação orientada a objetos

2.2.1 Encapsulamento

O encapsulamento refere-se à maneira como cada objeto é definido. Tipicamente, essa definição é parte de uma classe C++ e inclui uma descrição da estrutura interna do objeto, ou seja uma descrição de como ele se relaciona com outros objetos e a forma de proteção que isola os detalhes funcionais do objeto em relação ao exterior da classe. Assim, a classe satisfaz aos requisitos objeto-orientados para o encapsulamento, pois segundo Watson & Chan[35]: "*Não me interessa saber como é feito, somente interessa-me que seja feito...*". Geralmente a encapsulação de dados possui basicamente três finalidades:

- proteger os dados da exposição excessiva;
- ocultar detalhes do armazenamento de dados;
- facilitar a reutilização do código em outro projeto.

2.2.2 Herança

A programação orientada a objetos oferece uma maneira de relacionar classes umas com as outras por meio de hierarquias. Pode-se dividir classes em sub-classes, mantendo-se o princípio de que cada sub-classe herda as características da classe da qual foi derivada. Uma classe origem é chamada classe-base e as classes que compartilham as características de uma classe-base e têm outras características adicionais são chamadas classes derivadas. Uma classe-básica representa os elementos comuns a um grupo de classes derivadas. Uma classe que é derivada de uma classe-básica pode, por sua

vez, ser classe-básica de outra classe. O uso de classes derivadas aumenta a eficiência da programação pelo fato de não necessitar que se criem códigos repetitivos.

A uma biblioteca de funções não se pode adicionar outras implementações a não ser que ela seja reescrita ou que se possua seu código-fonte para alterá-la e recompilá-la. O uso de uma biblioteca de classes oferece uma grande vantagem sobre o uso de uma biblioteca de funções: naquela o programador pode criar classes derivadas a partir de classes-básicas da biblioteca. Assim, sem alterar a classe-base, é possível adicionar a ela características diferentes que a tornarão capaz de executar o que se determina. A facilidade com que classes existentes podem ser reutilizadas sem serem alteradas é um dos maiores benefícios oferecidos por linguagens orientadas a objetos. Com o uso da herança pode-se incluir dados e métodos em uma classe derivada sem ter que alterar a classe original. Dessa forma, um objeto pode herdar dados e métodos de uma classe, evitando a repetição de código para tratar dois objetos de classes diferentes, porém semelhantes.

2.2.3 Polimorfismo

O sentido da palavra polimorfismo[1] provém do uso de um único nome para definir várias formas distintas. Em C++ dá-se o nome de polimorfismo à criação de uma família de funções que compartilham do mesmo nome; mas cada uma tem um código independente. O resultado da ação de cada uma das funções da família é o mesmo. A maneira de atingir esse resultado é distinta.

A linguagem C++ implementa polimorfismo utilizando funções virtuais. Uma função *virtual* é uma função-membro de uma classe que é projetada para trabalhar virtualmente com quaisquer membros da classe-básica ou derivada (de tipo ainda desconhecido). Pode-se pensar numa função *virtual* como sendo uma "função variável", uma função-membro da classe que é projetada para ser substituída mais tarde por uma função-membro de uma classe derivada e ainda desconhecida. Muitas vezes tais funções não fazem nada na classe básica; estão ali somente para que possam ser referenciadas na classe básica e implementadas posteriormente pelas classes derivadas.

[1] *Poly* significa muitos. Já *morfic* significa formas. Quando se combinam os dois termos obtêm-se muitas formas, ou seja, um objeto polimórfico é aquele capaz de possuir duas ou mais formas.

Com o uso das características acima descritas aplicadas a um ambiente orientado a objetos, pode-se dizer que esse paradigma é bastante eficaz na criação de programas que serão facilmente modificados caso haja necessidade, visto que são compostos por rotinas de simples reutilização, servindo como plataforma de apoio para novos códigos.

Os problemas de elementos finitos aplicados ao estudo de estruturas recaem em procedimentos comuns aos diversos elementos que podem ser herdados sem modificação pelos diversos objetos que incorporam o problema. Esses comportamentos semelhantes, porém com algumas diferenças, podem herdar o mesmo método para obter seu resultado; no entanto, possuem diferentes implementações de cálculo. Assim, agiliza-se o trabalho dos programadores, que além do tempo e da confiabilidade de não incluir erros em código estável, ganham maior flexibilidade quando definem uma nova classe.

Um dos maiores desafios encontrados por quem trabalha com programação orientada a objetos é o nível de abstração dado às classes básicas, visto que estas devem estar aptas a acolherem classes derivadas com os mais diversos propósitos.

Capítulo 3

Ambiente PZ

Para solucionar problemas de mecânica do contínuo utilizam-se vários métodos de discretização os quais envolvem aproximações. Dentre esses métodos de aproximação destaca-se o método dos elementos finitos, que consiste em transformar as equações diferenciais em equações algébricas, utilizando uma transformação simples para as variáveis incógnitas. Esse método é amplamente aplicado aos problemas de mecânica dos fluidos, mecânica dos sólidos, podendo também ser aplicado a qualquer outro problema que se traduza na solução de um conjunto de equações diferenciais. Dada a sua grande aplicabilidade nas últimas décadas inúmeros softwares CAE foram desenvolvidos com base no método dos elementos finitos, sendo que a maior parte deles foram concebidos em linguagens estruturadas como Pascal, C e Fortran. Dessa maneira, um programa que poderia ser muito útil não é suficientemente explorado pois, dentre outros fatores, não é receptivo a extensões a outros problemas.

Com o uso de linguagens de programação seqüencial, o desenvolvimento de um trabalho que inclua a elaboração de um programa de computador pode causar uma série de contratempos, pois o simples fato de surgirem novas especificações no transcorrer do projeto causam enormes transtornos, pois tais modificações exigem uma readaptação a partir do início do programa incluindo muitas funções e rotinas já testadas, que acarreta um dispêndio de tempo por parte do programador incorrendo na possibilidade de introduzir novos erros em códigos antes estáveis.

No trato de elementos finitos, o código criado pela metodologia de programação seqüencial e estruturada dificilmente pode ser reutilizado para continuidade de programas que almejam a solução de um problema, sem contar

com o aumento da complexidade do código a ponto de dificultar a manutenção deste. Ao contrário da programação seqüencial e estruturada, o paradigma de programação orientada a objetos tem com uma de suas vantagens a flexibilidade de reutilização do código, o que tem impulsionado o uso dessa nova metodologia, principalmente no desenvolvimento de grandes projetos de software para engenharia.

Segundo Oden[19], um programa de elementos finitos padrão segue geralmente uma seqüência de execução de procedimentos que podem ser expressa nos seguintes passos:

1. divisão do domínio em sub-domínios de elementos finitos;
2. formulação variacional do problema em questão;
3. construção do modelo, definição dos coeficientes da equação diferencial, efetuando-se as contribuições de cada elemento do domínio;
4. aplicação das condições de contorno: deslocamentos e esforços no caso da análise estrutural;
5. solução do sistema linear de equações algébricas para determinar, no caso desse trabalho, a solução aproximada de deslocamentos;
6. pós-processamento dos resultados para análise de esforços na estrutura em estudo;

3.1 Ambiente PZ

Com o aumento do poder computacional evidenciado nos últimos anos, o método dos elementos finitos tem ganhado a cada dia mais adeptos. A evolução do método é notória, se considerarmos o número de artigos sobre o assunto. Segundo Cook[10], na década de 60 foram cerca de 200 artigos, 7000 na década de 70 e quase 20000 artigos em meados da década de 80. Avanços na capacidade de armazenamento de informações em memória e o aumento da velocidade dos processadores dos computadores pessoais contribuiram consideravelmente para o sucesso do método, tornando possível que problemas mais robustos, nunca antes respondidos pudessem ser analisados e melhor entendidos via método dos elementos finitos.

A plataforma PZ é um ambiente de programação científico, desenvolvido em linguagem C++, destinado à resolução de problemas de engenharia que possam ser formulados como um sistema de equações diferenciais parciais, como é o caso do Método dos Elementos Finitos.

Segundo Devloo[12], as classes do ambiente PZ definem tipos inerentes de problemas de elementos finitos tais como malha, elemento, nó e material. Tais classes implementam métodos para o tratamento dos dados como, por exemplo: aplicação das condições de contorno, integração numérica, montagem dos coeficientes do sistema de equações, transformação de coordenadas, criação do material, geração de malhas, solução do sistema, etc. Com o ambiente PZ pode-se tirar vantagem da reutilização de seu código e suas classes podem ser estendidas de modo a aumentar a abrangência de um programa.

É preciso salientar que diferente do que é feito classicamente, no ambiente PZ faz-se uma separação entre o que é malha geométrica e malha computacional. Através dessa estratégia, trata-se a aproximação da geometria independente da aproximação das grandezas físicas do problema.

O ambiente PZ, como na maioria dos ambientes de programação orientado a objetos, é organizado em um conjunto de classes que o usuário utiliza para o desenvolvimento de seus programas. Ao usuário compete a montagem de um código principal responsável pela criação dos objetos numa dada sequência lógica, que a princípio define a malha geométrica e a malha computacional, definição do modelo de material a ser resolvido (equação diferencial) expresso em termos de coeficientes na matriz de rigidez, aplicação das condições de contorno e por fim, o processo com as classes de análise e posterior definição da saída de resultados. Segundo Devloo[12], o ambiente PZ foi idealizado e desenvolvido, com base no paradigma de programação orientada a objetos, para que classes básicas abstraíssem o método dos elementos finitos de tal forma que pudessem ser divididos em cinco seções distintas:

1. classes para a definição da geometria do problema;
2. classes para a definição do espaço de interpolação;
3. classes para a definição dos materiais (equação diferencial) do problema e das condições de contorno;
4. classes para organização, controle, montagem da matriz e definição do tipo de armazenamento da matriz de rigidez com posterior solução do sistema;
5. classes para pós-processamento e visualização dos resultados.

Nesse ambiente deve-se obedecer a uma dada sequência de criação dos objetos, de forma a extrair os recursos disponíveis adequadamente. A Figura 3.1 ilustra a ordem de execução do algoritmo em que o ambiente PZ foi elaborado.

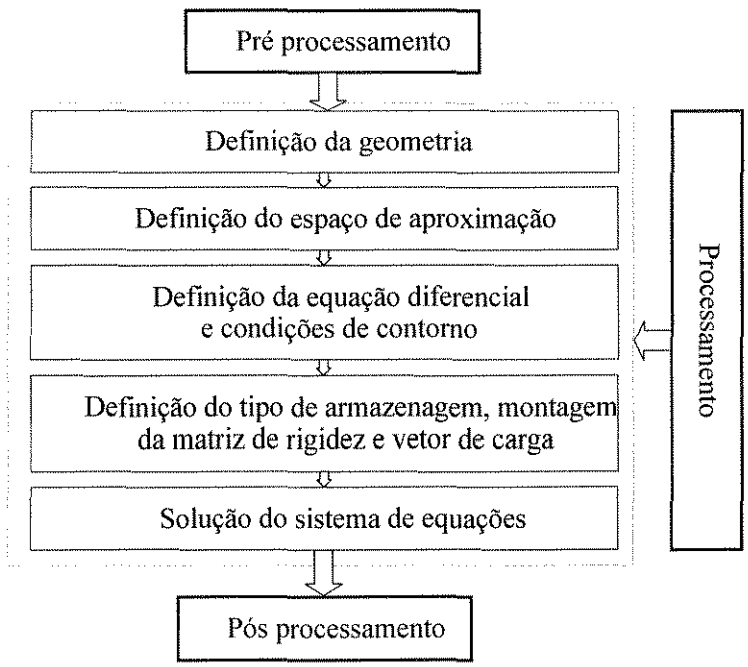

Figura 3.1: Algoritmo de criação dos objetos no ambiente PZ.

O PZ foi concebido para disponibilizar diversas análises. É possível discretizar um mesmo domínio por elementos retangulares e triangulares. Esse ambiente permite que sejam usados diferentes materiais e graus para os polinômios de interpolação para um mesmo domínio, o que permite uma variedade de análises muito abrangente, sem a necessidade de serem desenvolvidas versões particulares para cada tipo de modelo.

Para os problemas unidimensionais pode-se optar por uma discretização por elementos geométricos lineares e quadráticos, com um número livre de graus de liberdade por nó. No caso de malhas bidimensionais, para discretização do domínio, existe implementado no PZ elementos triangulares e retangulares que podem ser representados geometricamente em sua forma

linear e quadrática, também com um número livre de graus de liberdade por nó. No presente trabalho utilizaram-se seis graus de liberdade tanto para o elemento unidimensional de barra quanto para o elemento bidimensional de placa.

3.1.1 Pré-processamento

O ambiente PZ possui classes especialmente projetadas para o tratamento do pré processamento de dados, no qual se insere dados como: numeração dos nós, coordenadas dos nós, conectividade dos elementos, tipo de materiais e as condições de contorno naturais e essenciais existentes. Para o pré-processamento encontra-se implementado no ambiente a classe *TModulef* para leitura de arquivos com descrições de um modelo gerado pelo programa Modulef[30]. Durante o desenvolvimento da presente dissertação foram desenvolvidas a classe *TGengrid* para a criação de malhas retangulares e a classe *TAnsys2pz*. A Figura 3.2 ilustra a hierarquia de classes do pré-processamento.

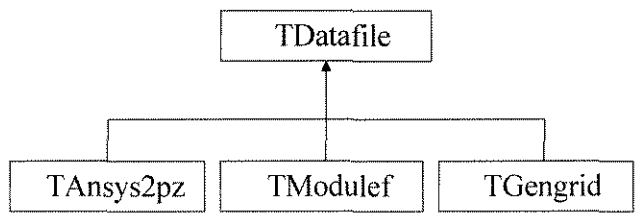

Figura 3.2: Hierarquia das classes que definem o pré processamento.

Com vistas a aumentar a interatividade com o usuário, a classe *TAnsys2pz* converte arquivos de dados gerados no software *Ansys release* 5.2 para o formato de leitura do PZ. Inicialmente, monta-se a geometria da estrutura, a conectividade dos elementos e as respectivas condições de contorno do problema no *Ansys,* e a seguir processam-se tais dados no ambiente PZ.

Da forma como foram implementadas, as classes do pré-processamento lêem os arquivos de relatórios gerados no Ansys, e através de polimorfismo (funções *virtual*) aplicado aos métodos da classe de leitura de materiais, o PZ reconhece automaticamente as informações adequadas desses materiais uni e bidimensionais, que são inseridos no programa.

3.1.2 Pós-processamento

A estrutura de classes responsáveis pelo pós-processamento conta com diversos métodos que geram formatos para visualização em softwares como: *MView* com a classe *TMVGrafGrid*, *View3D* que utiliza a classe *TV3DGrafGrid* e ainda o *Data Explorer* através da classe *TDXGrafGrid*. Essas classes são derivadas da classe *TGrafGrid*, uma classe básica abstrata que implementa métodos e incorpora as necessidades básicas para uma classe que objetiva o pós-processamento. A Figura 3.3 define a hierarquia das classes utilizadas para o pós-processamento.

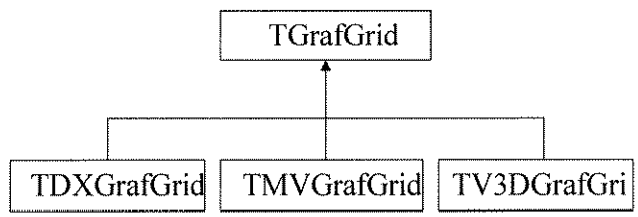

Figura 3.3: Hierarquia das classes que definem o pós processamento.

As classes do ambiente PZ responsáveis pela geração de arquivos de entrada para programas de visualização científica estão preparadas para gerar tais arquivos, tanto para dados escalares quanto para dados vetoriais. Com isso, é possível visualizar a solução com o uso de gradientes e outras ferramentas associadas aos resultados, o que possibilita uma visualização rica e elaborada.

3.1.3 Classes que definem a aproximação da geometria

O domínio de uma equação diferencial pode ser discretizado por elementos finitos, no qual cada elemento define um espaço parametrizado uni, bi e tridimensional. O mapeamento do domínio pode ser definido por funções de interpolação lagrangeanas lineares e quadráticas. Para representar a aproximação da geometria desse domínio por elementos finitos, o ambiente PZ utiliza-se das seguintes classes:

- TGeoGrid : malha geométrica;
- TGeoNod : nó geométrico;

- TGeoEl : elemento geométrico;
- TCosys : sistema de coordenadas.

Tal parametrização é realizada entre um mapeamento da configuração do domínio (deformada) e a configuração de um elemento mestre, como mostra a Figura 3.4.

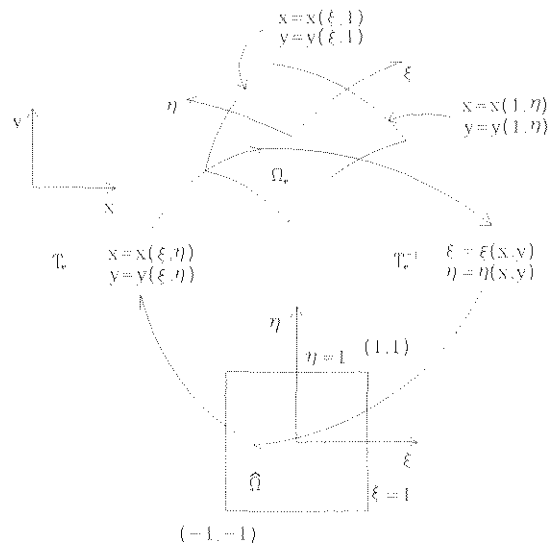

Figura 3.4: Representação do mapeamento do elemento mestre para o elemento deformado.

Para tratamento das condições de contorno foi desenvolvida uma estrutura chamada *TGeoNodBc* aplicada a nós geométricos. A principal classe da aproximação geométrica é a classe *TGeoGrid* que possui listas de ponteiros que apontam para objetos do tipo *TGeoEl*, *TGeoNod* e *TCosys*, acima citados.

Classe TGeoGrid

A classe *TGeoGrid* é responsável pelo controle dos nós e elementos que definem a geometria do domínio da equação diferencial. A classe *TGeoGrid* disponibiliza ao usuário acesso a:

- lista de nós geométricos;
- lista de elementos geométricos;
- lista de nós no contorno do domínio;
- lista de elementos no contorno do domínio;
- métodos para identificação da vizinhança entre elementos.

Para garantir performance e melhorar o controle de acesso aos dados, estes foram armazenados em árvores binárias do tipo *TVoidPtrMap* da GNU C++ Library[16], incorporando-se, assim, os métodos pertencentes a essa biblioteca ao ambiente PZ.

Principais variáveis-membro

- char fChecked

  Verdadeiro se todos os nós estão definidos.

- char fName[64]

  Armazena o nome da malha geométrica.

- int fNDim

  Armazena a dimensão do problema em estudo.

- static TGeoGrid *gCurrent

  Cópia única na memória, determina qual é a malha geométrica corrente para problemas com mais de uma malha.

- TVoidPtrMap fNodeMap

  Lista de ponteiros para nós do tipo TGeoGrid.

- TVoidPtrMap fElementMap

Lista de ponteiros para elementos do tipo TGeoGrid.

- TVoidPtrMap fNodBndCondMap
  Lista de ponteiros para nós com condições de contorno.

- TVoidPtrMap fElBndCondMap
  Lista de ponteiros para elementos com condição de contorno.

Principais funções-membro

- int NumNodes()
  Retorna o número de nós da malha.

- int NumElem()
  Retorna o número de elementos da malha.

- TGeoGrid(int nd = 3)
  Construtor da classe, possui como parâmetro a dimensão da topologia da malha; nesse caso o default é três dimensões.

- TGeoGrid(TGeoGrid &gr)
  Construtor de cópia, no qual constrói-se o objeto corrente com todos os dados de gr.

- TVoidPtrMap &NodeMap()
  Retorna uma referência para uma lista de nós geométricos (TGeoNod).

- TVoidPtrMap &ElementMap()
  Retorna uma referência para uma lista de elementos geométricos (TGeoEl).

- TVoidPtrMap &NodeBcMap()
  Retorna uma referência para uma lista de nós geométricos com condição de contorno.

- TVoidPtrMap &ElementBcMap()

Retorna uma referência para uma lista de elementos geométricos com condição de contorno.

- TGeoNod *FindNode(long nid)

  Retorna um ponteiro para um objeto do tipo TGeoNod, este objeto representa o nó geométrico cujo Id seja igual a nid. Retorna NULL se o nó não for encontrado.

- TGeoEl *FindElem(long elid)

  Retorna um ponteiro para um objeto do tipo TGeoEl, o qual objeto representa o elemento geométrico cujo Id seja igual a elid. Retorna NULL se o elemento não for encontrado.

- TCosys *FindCosys(long cosysid)

  Retorna um ponteiro para o objeto do tipo TGeoEl, o qual sistema representa o sistema de coordenadas cujo Id seja igual a cosysid. Retorna NULL se o sistema de coordenadas não for encontrado.

- void BuildConnectivity()

  Constrói e verifica a conectividade dos elementos na malha, responsável por inicializar a informação de conectividade dentro da malha.

- void GetBoundaryElements(int NodFrom,int NodTo,VoidPtrVec &ElementVec, TIntVec &sides)

  Retorna no vetor de ponteiros ElementVec, ponteiros para os elementos no contorno do domínio, localizados entre os nós NodFrom e NodTo contados no sentido anti-horário e a variável sides é preenchida com o número do lado do elemento que encontra-se no contorno.

Classe TGeoEl

A classe *TGeoEl* trata-se de uma classe abstrata que possui métodos para manipulação dos elementos geométricos. Define o mapeamento entre o elemento deformado e o elemento mestre. Seus métodos são responsáveis pelo cálculo do Jacobiano, identificação do vizinho do elemento, identificação do número de referência da condição de contorno e também do material, divisão do elemento em sub-elementos e criação do elemento computacional a partir do elemento geométrico.

As classes *TGeoEl1d*, *TGeoElQ2d* e *TGeoElT2d* implementam os elementos geométricos unidimensional, quadrilátero bidimensional e triangular bidimensional, respectivamente. Esses elementos são implementados na sua forma linear e quadrática viabilizando a discretização de domínios em geometrias diversas.

Os elementos no ambiente PZ são derivados da classe básica *TGeoEl* e podem ser ilustrados conforme Figura 3.5.

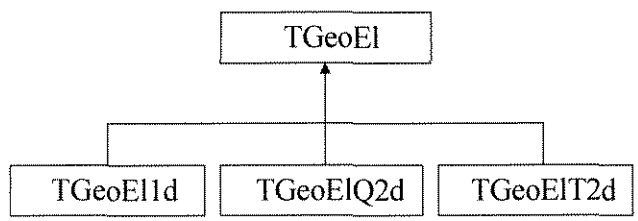

Figura 3.5: Hierarquias das classes que definem os elementos geométricos do PZ

Principais variáveis-membro

- long fId
  Número de identificação do elemento.
- int fNumNodes
  Número de nós no elemento.
- long fMatIndex
  Número de identificação do material.
- int fNumSides
  Número de arestas do elemento.

- TGeoGrid *fGrid

  Ponteiro para a malha a que o objeto corrente pertence.

- TCompEl *fReference

  Ponteiro para um elemento computacional criado segundo o elemento geométrico corrente.

- LongVec fSide

  Vetor de identificação do lado que os elementos vizinhos se conectam no elemento corrente.

- VoidPtrVec fNodep

  Vetor de ponteiros que apontam para os nós que compõe um elemento.

- VoidPtrVec fConnect

  Vetor de ponteiros que apontam para elementos vizinhos conectados ao elemento corrente.

Principais funções-membro

- TGeoEl(long Id)

  Construtor que possui como argumento o número de identificação do elemento(Id). As variáveis são inicializadas com zero e os ponteiros apontam para NULL.

- TGeoEl(TGeoEl &el)

  Construtor de cópia. Cria-se o elemento com as mesmas informações de el.

- TGeoGrid *Grid()

  Método que retorna um ponteiro para a malha que o elemento pertence.

- void SetGrid(TGeoGrid *gr)

Mostra ao elemento corrente que ele pertence à malha apontada por gr.

- long Id()

  Retorna o Id do elemento.

- int NumberOfNodes()

  Retorna o número de nós do elemento.

- long MaterialNumber()

  Retorna o número de identificação do material do elemento corrente.

- TCompEl *Reference()

  Retorna um ponteiro para o elemento computacional que o elemento corrente está apontando.

- virtual TCompEl *CreateCompEl()

  Cria um elemento computacional baseado no elemento geométrico corrente.

- short NumSides()

  Retorna o número de lados do elemento.

- virtual TGeoNod *SideNode(int side, int node) = 0

  Retorna um ponteiro ao nó node do lado side do elemento.

- TGeoEl *Neighbour(short is)

  Retorna um ponteiro para o elemento vizinho que está conectado no lado is do elemento corrente.

- Bc(short side)

  Retorna o número de identificação da condição de contorno do lado side.

- void Shape(double x, int n, TFMatrix &phi, TFMatrix &dphi)

Calcula n funções de forma no ponto x. A seguir o valor da função é retornado em *phi* e de sua derivada em *dphi*. As funções utilizadas no mapeamento da geometria pelo ambiente PZ são funções lagrangeanas lineares e quadráticas unidimensionais, como pode ser exemplificado em (3.1) e (3.2).

Para $n = 2$:

$$phi = \begin{bmatrix} \frac{1-\xi}{2} \\ \frac{1+\xi}{2} \end{bmatrix} \qquad dphi = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} \tag{3.1}$$

Para $n = 3$:

$$phi = \begin{bmatrix} \frac{\xi(\xi-1)}{2} \\ 1-\xi^2 \\ \frac{\xi(\xi+1)}{2} \end{bmatrix} \qquad dphi = \begin{bmatrix} \xi - \frac{1}{2} \\ -2\xi \\ \xi + \frac{1}{2} \end{bmatrix} \tag{3.2}$$

- virtual void Jacobian(DoubleAVec &fl, TFMatrix &jacobian, TFMatrix &axes)

  Retorna em jacobian o cálculo do jacobiano no ponto definido por fl, representado no sistema global de coordenadas por axes.

- virtual void X(DoubleAVec &coord, DoubleAVec &result)

  Disponibiliza as coordenadas do ponto coord no elemento mestre e retorna este com o valor correspondente das coordenadas globais (domínio deformado) em result.

Classe TGeoNod

A classe *TGeoNod* armazena as coordenadas de um nó no espaço tridimensional e é responsável pela representação do nó geométrico da malha. Também armazena o sistema de coordenadas utilizado para representar o ponto, dado em suas coordenadas globais.

Principais variáveis-membro

- char fDefined

  Indica se a malha foi verificada ou não.

- long fId

  Identificação do nó.

- double fCoord[3]

  Vetor que armazena as coordenadas dos nós, limitado a três dimensões.

- TCosys *fSys

  Retorna um ponteiro para sistema de coordenadas corrente.

- TDofNod *fDofNod

  Retorna um ponteiro para um objeto do tipo TDofNod, responsável pela gerência dos graus de liberdade do nó.

Principais funções-membro

- TGeoNod(long id, int d=3, double *xp=NULL, TCosys *ref=NULL)

  Construtor da classe que possui como argumentos o número de identificação do nó, o número que define a dimensão, o ponteiro xp que representa as coordenadas segundo o sistema global e ainda o ponteiro ref indicando o sistema de coordenadas adotado para o nó.

- TGeoNod(TGeoNod &gn)

  Construtor de cópia. Cria o objeto corrente com os dados de gn.

- void SetCoord(double *x, int d=3)

  Atualiza as coordenadas para os valores em x.

- void SetReference(TDofNod *dofn)

  Indica ao nó corrente que o nó computacional referente é dofn, dispõe informações dos graus de liberdade de um nó geométrico.

- void SetCosys(TCosys *cos)

Troca o sistema de coordenadas para o apontado por cos.

- TDofNod *Reference()
  Retorna um ponteiro para o nó computacional.

Classe TCosys

A classe *TCosys* define os métodos e trata dados referentes ao sistema de coordenadas. É uma classe básica e abstrata de onde são derivadas *TCartsys*, *TCylinsys*, *TEsfersys* que tratam coordenadas cartesianas, cilíndricas e esféricas, respectivamente. Dessa forma, disponibilizam-se diferentes tipos de sistemas de coordenadas para os sistemas locais e globais, como ilustra a Figura 3.6.

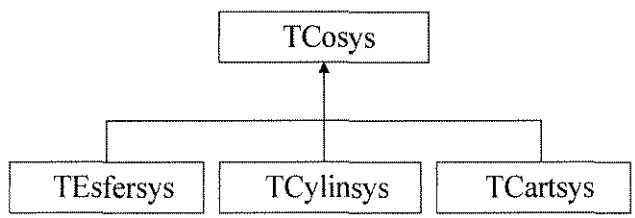

Figura 3.6: Hierarquia das classes que definem os sistemas de coordenadas.

Principais variáveis-membro

- int fNumber
  Identificação do sistema de coordenadas.

- float fOrigin[3]
  Vetor que armazena a origem do sistema de coordenadas, expresso em sistema de coordenadas cartesianas.

- TCosys *fReference
  Ponteiro para o sistema de coordenadas.

- float fTr[3][3]

  Armazena os vetores unitários expressos em coordenadas cartesianas. Relativos ao sistema de coordenadas de referência.

Principais funções-membro

- TCosys(int num, TCosys *ref=NULL, float *org=NULL)

  Construtor da classe, onde num é a identificação do sistema, ref é a referência do sistema e org é a origem do sistema.

- virtual int Type()

  Retorna o tipo do sistema de coordenadas corrente.

- void SetReference(TCosys *co, float *org=NULL)

  Atualiza o sistema de referência para co e a origem para org. Este último expresso no sistema de coordenadas corrente.

- void SetAxes(FloatAVec &x, FloatAVec &z)

  Define o eixo y normal a x e z.

- void ToReference(float point[3])

  Altera point do sistema corrente para o sistema de referência.

- void FromReference(float point[3])

  Altera point do sistema de referência para o sistema corrente.

- void ToGlobal(float point[3])

  Altera point do sistema corrente para o sistema global.

- void FromGlobal(float point[3])

  Altera point do sistema global para o sistema corrente.

Classe TGeoNodBc

A classe *TGeoNodBc* é responsável por definir a condição de contorno aplicada a um nó geométrico. Este objeto é declarado como uma *struct*, assim torna-se acessível de qualquer parte do código. Em sua estrutura possui declarado um ponteiro para *TGeoNod*, para aplicação de condições de contorno em nós geométricos.

Principais variáveis

- TGeoNod *fNod

  Nó geométrico da condição de contorno.

- int fId

  Número de identificação da condição de contorno.

Principais funções

- TGeoNodBc(TGeoNod *nodin, int numberin)

  Atribui a fNod o endereço de nodin e a fId o valor de numberin.

3.1.4 Classes que definem o espaço de aproximação

A complexidade do modelo matemático que representa o comportamento de muitos problemas de engenharia levou ao desenvolvimento de métodos aproximados para sua solução. Dentre esses métodos destacam-se *Rayleigh-Ritz* e *Galerkin*, que mais tarde deram origem ao método dos elementos finitos. A distinção entre o método dos elementos finitos e esses métodos aproximados está na forma sistemática com que o método dos elementos finitos define as funções de interpolação. No ambiente PZ o espaço de funções de aproximação é definido com base em polinômios ortogonais, o que caracteriza a efetiva dissociação das funções de interpolação com as funções de mapeamento da geometria. Assim, além do conceito de malha geométrica, o conceito de malha computacional composta por elementos computacionais e nós computacionais é introduzido para o cálculo de elementos finitos. A cada elemento computacional associa-se um elemento geométrico, este último

usado para calcular o mapeamento entre o elemento deformado e o elemento mestre.

O elemento computacional calcula as funções de forma e suas derivadas nos pontos de integração. Dessa forma, o elemento computacional utiliza informações, como ilustra a Figura 3.7, do elemento geométrico associado para calcular a matriz de rigidez do elemento.

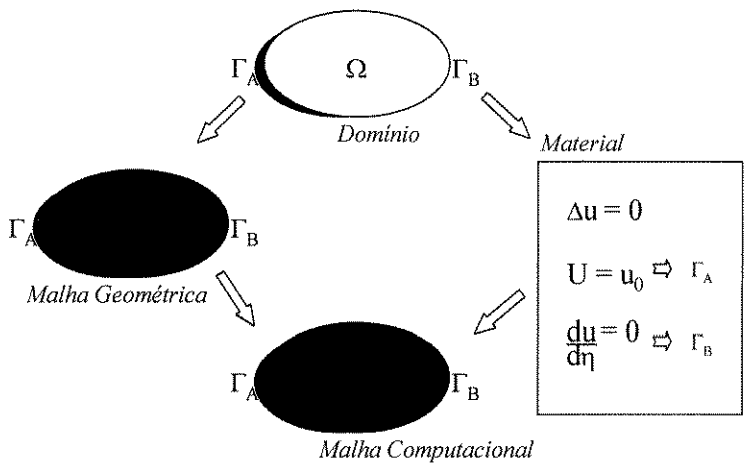

Figura 3.7: Caracterização de um problema de elementos finitos no ambiente PZ.

Classe TCompGrid

A classe *TCompGrid* é responsável pelo controle dos nós e elementos computacionais, materiais e condições de contorno. São implementadas nessa classe outros métodos para:

- calcular a largura da banda do sistema de equações;
- calcular o número de equações do modelo;
- renumeração dos nós segundo o algoritmo de *Cuthill-Mckee*;
- montar o sistema de equações globais e vetor de carga;

- carregar um vetor de solução nos graus de liberdade. A malha computacional está associada a uma única malha geométrica, contudo uma malha geométrica pode relacionar-se com diversas malhas computacionais; assim, a adaptatividade do tipo h pode ser implementada utilizando-se uma malha geométrica e várias malhas computacionais.

Principais variáveis-membro

- static TCompGrid *gCurrent

  Para problemas com mais do que uma malha computacional; define qual malha é a corrente.

- TGeoGrid *fReference

  Retorna um ponteiro para a malha geométrica a qual a malha computacional se refere.

- TVoidPtrMap fNodeMap

  Lista de ponteiros para nós.

- TVoidPtrMap fElementMap

  Lista de ponteiros para elementos.

- TVoidPtrMap fMaterialMap

  Lista de ponteiros para materiais.

- TVoidPtrMap fBndCondMap

  Lista de ponteiros para as condições de contorno.

- TVoidPtrMap fNodBndCondMap

  Lista de ponteiros para nós com condições de contorno associadas.

- TVoidPtrMap fElBndCondMap

  Lista de ponteiros para elementos com condições de contorno associados.

Principais funções-membro

- TCompGrid(TGeoGrid *gr)

  Construtor da classe e possui como argumento uma malha geométrica, utilizada como base para a construção da malha computacional.

- TCompGrid(TCompGrid &gr)

  Construtor de cópia, onde as informações corrente são inicializados com os dados de gr.

- int NumNode()

  Retorna o número de nós da malha.

- int NumElem()

  Retorna o número de elementos da malha.

- int NumMat()

  Retorna o número de materiais da malha.

- int NumBc()

  Retorna a quantidade de condições de contorno na malha.

- int NumElBc()

  Retorna o número de elementos com condições de contorno associados a eles.

- int NumNodBc()

  Retorna o número de nós com condições de contorno associados a eles.

- TVoidPtrMap &NodeMap()

  Retorna uma referência para os nós geométricos.

- TVoidPtrMap &ElementMap()

  Retorna uma referência para uma lista de elementos geométricos.

- TVoidPtrMap &NodeBcMap()

Retorna uma referência para uma lista de nós geométricos que possuem condição de contorno associados.

- TVoidPtrMap &ElementBcMap()

  Retorna uma referência para uma lista de elementos geométricos que possuem condição de contorno associados.

- TVoidPtrMap &MaterialMap()

  Retorna uma referência para uma lista materiais.

- TVoidPtrMap &BndCondMap()

  Retorna uma referência para uma lista de condições de contorno.

- TDofNod *FindNode(long nid)

  Retorna um ponteiro para o nó computacional, cujo número de identificação é igual a nid. Retorna NULL caso o nó não seja encontrado.

- TMaterial *FindMaterial(long matid)

  Retorna um ponteiro para o material, cujo número de identificação é igual a matid.

- TCompEl *FindElement(long elid)

  Retorna um ponteiro para o elemento computacional, cujo número de identificação é igual a elid.

- void ComputeNodeSequence(long ElementId)

  Renumera os nós segundo o algoritmo de Cuthill-Mckee, iniciando pelo elemento cuja identificação é dada por ElementId.

- long NumEquations()

  Retorna o número de equações do sistema global.

- long BandWidth()

  Retorna a largura da banda do sistema gerado.

- void Assemble(TBlock &block, TMatrix &rhs)

Monta a matriz de rigidez na matriz referenciada por block e o vetor de carga em rhs.

Classe TCompEl

A classe básica e abstrata *TCompEl* define o comportamento que o elemento computacional deve ter para enquadrar-se no ambiente PZ. As classes *TCompEl1d* e *TCompEl2d*, caracterizadas por tratarem dos elementos computacionais unidimensional e bidimensional, são derivadas da classe *TCompEl*.

São as classes *TCompElT2d* e *TCompElQ2d*, que implementam o comportamento para elementos computacionais bidimensionais triangulares e retangulares, respectivamente. Por fim, a classe *TElBiot2d* é derivada da classe *TCompElQ2d* destinada a implementar o problema de *Biot* para ambiente.

A classe *TCompEl* foi projetada com o propósito de ser uma classe abstrata, onde alguns de seus métodos são elaborados nas classes derivadas, dado o nível de abstração explorado. Assim, a classe engloba muitas tarefas dentre as quais destacam-se:

- definição da ordem de interpolação;
- definição da regra de integração para o elemento;
- cálculo de funções de forma;
- aplicação das condições de contorno na matriz de rigidez;
- cálculo da matriz de rigidez do elemento;
- cálculo do erro dada pela aproximação.

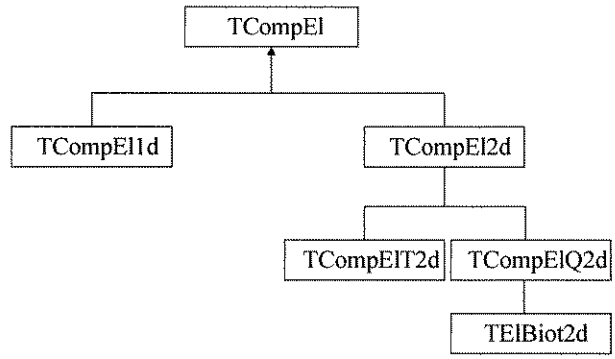

Figura 3.8: Hierarquia das classes que definem os elementos computacionais no ambiente PZ.

Na Figura 3.8 observa-se a associação entre as classes que caracterizam os elementos computacionais.

Principais variáveis-membro

- long fId

  Número de identificação do elemento computacional.

- TCompGrid *fCGrid

  Ponteiro para a malha computacional à qual o objeto corrente pertence.

- TGeoEl *fReference

  Elemento geométrico com referência para o elemento computacional corrente.

Principais funções-membro

- TCompEl(int dim)

  Contrutor da classe, onde o elemento computacional é criado com os dados sem valores iniciais e dimensão dim.

- TCompEl(TCompEl &el)

  Construtor de cópia, o elemento corrente é criado com os dados de el.

- TCompGrid *Grid()

  Retorna um ponteiro para a malha computacional à qual o elemento pertence.

- long Id()

  Retorna o número de identificação do elemento.

- TGeoEl *Reference()

  Retorna um ponteiro para um elemento geométrico referenciado pelo elemento computacional.

- virtual void long MaterialNumber()

  Retorna o índice do material do objeto corrente.

- virtual TDofNod *SideNode(short is)

  Retorna um ponteiro para o nó situado no lado is.

- TCompEl *Connect(short iside)

  Retorna um ponteiro para o elemento computacional vizinho ao lado side do elemento corrente.

- virtual void SetInterpolationOrder(ShortAVec &ord)

  Retorna a ordem do polinômio de interpolação utilizado para o cálculo da aproximação.

- void Chebyshev(double x, int num, TFMatrix &phi, TFMatrix &dphi, long *id)

  Calcula num funções de forma unidimensionais no ponto x e retornando em phi e dphi os valores das funções de forma e de sua derivada, respectivamente.

- virtual void Shape(double x, int num, TMatrix &phi, TMatrix &dphi)

Para o ponto x, retorna o valor de n funções de forma e suas derivadas em phi e dphi, respectivamente. Essas são as funções de forma utilizadas na aproximação numérica do problema. No ambiente PZ faz-se uso das funções hierarquicas de Chebyshev. Nos capítulos que seguem, estas funções serão identificadas por N_i.

- virtual void ApplyBc(TElementMatrix &ek, TElementMatrix &ef, TBndCond &bc, int lado)

  Aplicação das condições de contorno na matriz de rigidez ek e vetor de cargas ef, sendo tais condições de contorno dados por bc.

- virtual void CalcStiff(TElementMatrix &ek, TElement &ef)

  Calcula a matriz de rigidez e vetor de cargas local do elemento em ek e ef, respectivamente.

- virtual void Solution(DoubleAVec &qsi, int var, DoubleAVec &sol, int &numvar)

  Retorna o valor da solução do sistema no vetor sol.

Classe TDofNod

A classe *TDofNod* é responsável pelo gerenciamento dos dados relacionados com o nó computacional. Consiste de métodos para identificar o número da equação associada ao grau de liberdade, o número de elementos conectados ao nó, a solução do sistema, etc.

Principais variáveis-membro

- long fNodId

  Variável que define o número de identificação do nó.

- TGeoNod *fReference

  Ponteiro para um nó geométrico do tipo TGeoNod.

- DoubleAVec fVar

  Vetor que armazena as variáveis dos graus de liberdade.

- int fNumElCon

  Variável que define o número de elementos conectados ao nó computacional corrente.

- static long gNodeCounter

  Contador estático com o objetivo de criar um número de identificação único para o nó.

Principais funções-membro

- TDofNod(int ndof, TGeoNod *ref=NULL)

  Construtor da classe, onde ndof é o número de graus de liberdade do nó e ref é dado pelo nó geométrico associado.

- int NDof()

  Retorna o número de graus de liberdaded do nó corrente.

- void SetNDof(short newsize)

  Altera o número de graus de liberdade associados ao nó.

- void SetVal(int i, double x)

  Atualiza a variável i com o valor x.

- TGeoNod *Reference()

  Retorna um ponteiro para o nó geométrico que o objeto corrente está associado.

- int NumElCon()

  Retorna o número de elementos conectados ao nó.

Classe TDofNodBc

A classe *TDofNodBc* é responsável pelo controle e a aplicação das condições de contorno nos graus de liberdade dos nós computacionais. Essas condições de contorno podem ser essenciais e naturais, também conhecidas por *Dirichlet* e *Newmman*. Existe ainda a condição mista onde associa-se condições naturais e essenciais aos graus de liberdade de um nó. Definido como uma *struct*, suas variáveis estão disponíveis de qualquer parte do programa e para qualquer classe que necessite tais dados.

Principais variáveis

- TDofNod *fNod

  Ponteiro para um nó que possui condição de contorno associada.

- TBndCond *fBc

  Ponteiro para a condição de contorno.

Principais funções

- TDofNodBc(TDofNod *nod, TBndCond *bc)

  Construtor da classe. Inicializa um objeto com fNod = nod e fBc = bc.

3.1.5 Classes que definem os materiais e condições de contorno

A equação diferencial que expressa os fenômenos físicos que ocorrem com os materiais, é transformada no que se convenciona a chamar de formulação variacional do problema. Essas formulações são expressas por integrais de funções cujos argumentos são valores das funções de forma e de suas derivadas, que representam a solução aproximada das mencionadas equações diferenciais. A classe básica e abstrata que implementa essas características é a classe *TMaterial*. Nas classes derivadas de *TMaterial* são manipulados os coeficientes dos materiais específicos que representam o comportamento

de um dado material. Assim, com o auxílio da representação matricial, resolver estas formulações para tais coeficientes corresponde a solucionar um problema de análise computacional em engenharia.

Classe TMaterial

A classe básica *TMaterial* tem por função montar e calcular a contribuição de um elemento na matriz de rigidez e no vetor de cargas. Presupondo que em um problema esteja envolvido materiais de dimensões distintas, foram implementados métodos que verificam se o objeto pertence a uma classe derivada apropriada. Dessa maneira, *TMat1dLin* e *TMat2dLin* são classes derivadas de *TMaterial*, encarregadas de trabalhar com materiais de uma e duas dimensões, respectivamente.

Para analisar o problema do material de barra elástica linear pela hipótese de *Timoshenko*, implementou-se no presente trabalho a classe *TTimoshenko*, derivada de *TMat1dlin*.

Para a análise do problema do material de placa com a hipótese de *Reissner-Mindlin*, implementou-se a classe *TPlateReissner*, derivada de *TMat2dLin*.

Ressalta-se que a classe *TMat2dLin* serve ainda como classe básica para outras classes de materiais como: *TMatBiot*, *TPMat*.

A Figura 3.9 ilustra a distribuição das classes responsáveis pelo tratamento dos dados e métodos referentes aos materiais no ambiente PZ.

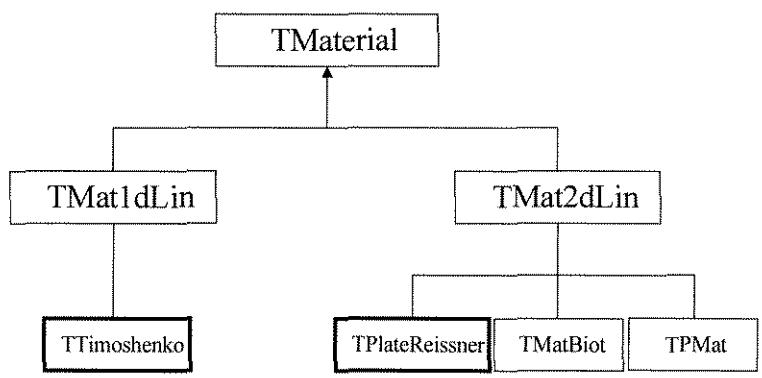

Figura 3.9: Hierarquia das classes que definem o tratamento dos materiais no ambiente PZ.

Principais variáveis-membro

- long fId

  Variável que armazena o número de identificação do material.

Principais funções-membro

- TMaterial(long id)

  Construtor da classe, onde o argumento id é o valor de inicialização de fId.

- TMaterial(TMaterial nn)

  Construtor de cópia. Cria o objeto corrente com os dados de nn.

- virtual char *Name()

  Retorna o nome da classe derivada da qual Name() está sendo referenciada.

- virtual TBndCond *CreateBc(long id, int typ, TFMatrix val1, TFMatrix val2)

  Cria um objeto da classe TBndCond com o número de identificação igual a id, com a condição de contorno typ e os valores das condições de contorno expressos em val1 (Dirichlet) e val2 (Neumman), respectivamente.

- virtual short DerivedFrom(char *Name)

  Retorna 1 caso a classe seja derivada ou igual a classe armazenada em Name.

- long Id()

  Retorna o número de identificação do material.

- virtual short NumberOfFluxes()

  Retorna o número de informações que se deseja para um dado pós processamento. Por exemplo, no problema de viga deseja-se analisar momentos fletores e esforços cortantes, logo o número de fluxos é dois.

- virtual short NumVariables()

  Retorna o número de variáveis de estado. Para o presente estudo, no caso de vigas tridimensionais considera-se três deslocamentos e três rotações.

- virtual void SetForcingFunction(void (*fp)(DoubleAVec loc, DoubleAVec result))

  Define um vetor de carga em forma de função.

- virtual int VariableIndex(char *name)

  Retorna o índice da variável com nome name.

- virtual void Print(ostream out=cout)

  Imprime os dados do material na saída definida por out.

- virtual void Solution(TFMatrix Sol, TFMatrix DSol, int var, DoubleAVec Solout, int numvar)

  Retorna a solução associada a var, definido os valores de estado e suas derivadas em Sol e DSol.

Classe TBndCond

A classe *TBndCond* apenas armazena os dados utilizados por objetos do tipo *TMaterial* para posterior apliacação dos mesmos. Para viabilizar a compatibilidade com o objeto *TMaterial*, a classe *TBndCond* torna acessível apenas objetos do tipo *TMaterial*. Isto é possível pois a classe *TBndCond* declara as classes de materiais como amigas ("*friend*") da classe *TBndCond*, o que possibilita o acesso das classes de materiais aos dados e métodos privados de *TBndCond*. Com isso, as declarações abaixo tornam as variáveis-membro de *TBndCond* acessíveis às classes do tipo *TMaterial*.

- friend class TMat1dLin
- friend class TMat2dLin

- friend class TMatBiot
- friend class TPMat
- friend class TElasticity
- friend class TTimoshenko
- friend class TPlateReissner

Principais variáveis-membro

- long fNumber
  Número de identificação da condição de contorno.
- int fType
  Tipo da condição de contorno (Dirichlet, Neumman e Mista).
- TFMatrix fBcVal1
  Matriz que armazena os valores da condição de contorno essencial (Dirichlet) a ser aplicada.
- TFMatrix fBcVal2
  Matriz que armazena os valores da condição de contorno natural (Neumman) a ser aplicada.
- TMaterial *fMatPtr
  Ponteiro para o objeto TMaterial que criará a condição de contorno.

Principais funções-membro

- TBndCond(TBndCond bc)
  Construtor de cópia. Cria o objeto corrente com os dados de bc.
- TBndCond(long number, int type, TFMatrix val1, TFMatrix val2)
  Construtor da classe, onde os argumentos inicializam fNumber, fType, fBcVal1 e fBcVal2, respectivamente.
- long Type()
  Retorna o tipo da condição de contorno.
- long Number()
  Retorna o número de identificação da condição de contorno.
- TMaterial *MatPtr()
  Retorna um ponteiro para o material que criará a condição de contorno.

3.1.6 Classes que definem a montagem e solução do sistema

Classe TAnalysis

A implementação de uma classe que viabilize a troca de informações e o controle sobre os diversas objetos do ambiente é responsabilidade da classe *TAnalysis*. Uma vez criados estes objetos no ambiente, a classe *TAnalysis* utiliza-se destes e invoca os métodos necessários, controlando o programa até a completa solução do sistema.

Principais variáveis-membro

- TGeoGrid *fGeoGrid

  Ponteiro para a malha geométrica.

- TCompGrid *fCompGrid

  Ponteiro para a malha computacional.

- TMatrix *fStriffness

  Ponteiro para a matriz de rigidez, que armazena a contribuição da rigidez de cada elemento envolvido. Como TMatrix é uma classe básica pode-se referenciar qualquer classe derivada de TMatrix.

- TFMatrix *fRhs

  Armazena o vetor de cargas do problema.

- TFMatrix *fSolution

  Armazena a matriz de rigidez do problema.

Principais funções-membro

- TAnalysis(TCompGrid *calcgrid)

Contrutor da classe, onde a análise é criada com os dados de calcgrid.

- void SetMatrix(TMatrix *stiff)
  Define a matrix stiff como matriz de rigidez.

- void Assemble()
  Montagem da matriz de rigidez e vetor de carga.

- void Solve()
  Resolve o sistema de equações.

- virtual void Run(VoidPtrVec scalnames, VoidPtrVec vecnames, char *plotfile, ostream out=cout)
  Executa de forma ordenada cada método necessário, começando pela montagem da matriz de rigidez, aplicação das condições de contorno, solução do sistema e por fim criação de um arquivo de resultados gerado pelos métodos de pós processamento. O argumento *scalnames* armazena nomes das variáveis escalares e *vecnames* das variáveis vetoriais e as informações pertinentes à análise são gravadas em um arquivo de saída.

  No caso desta dissertação utiliza-se a armazenagem da matriz de rigidez em banda e a solução do sistema pode ser realizada por qualquer método direto disponível nas classes de matrizes associadas ao ambiente PZ, dentre estes destacam-se os métodos de Cholesky, LU e LDLt.

Exemplo de código para acesso ao ambiente PZ

O ambiente PZ possui suas características e padrões que devem ser respeitadas. O procedimento de acesso aos métodos do sistema é de fundamental importância para o sucesso da análise de elementos finitos que se proponha fazer. Entender os passos trilhados pelo ambiente até a solução do sistema é uma tarefa não muito aprazível, muito embora necessário. A seguir lista-se o referido procedimento que possibilita uma análise particular para um dado problema utilizando o ambiente PZ.

```
void ComputElement(TCompGrid &c, TAnalysis * &an);
int main() {
    TGeoGrid geo;
    TCompGrid comp(&geo);
    TCompGrid::gCurrent = &comp;
    TGeoGrid::gCurrent = &geo;
    TAnsys2pz arq1d(''lixo'');
    arq1d.Read(comp);
    ofstream out(''saida.dat'');
    geo.BuildConnectivity();
    geo.Print(out);
    TAnalysis *an;
    ComputElement(comp,an);
    comp.Print(out);
    TSkylMatrix *stiff = new TSkylMatrix(an->NumEquations());
    an->SetMatrix(stiff);
    an->Solver().SetDirect(ECholesky);
    int dimension = 1;
    VoidPtrVec scalnames(0),vecnames(3); vecnames[0] = ''Displacement'';
    vecnames[1] = ''Moment'';
    vecnames[2] = ''Shear'';
    an->DefineGrafGrid(dimension,scalnames,vecnames,''predio1d.dx'');
    dimension = 2; vecnames.resize(1);
    scalnames.resize(2);
    scalnames[0] = ''MomentKsi'';
    scalnames[1] = ''MomentEta'';
    an->DefineGrafGrid(dimension,scalnames,vecnames,''predio2d.dx'');
    an->Run(out);
    int resolution = 3;
    an->PostProcess(resolution);
    return 0;
}
void ComputElement(TCompGrid &c,TAnalysis *&an) {
    TCompEl1d::gOrder = 4;
    TCompEl2d::gOrderx = 4;
    TCompEl2d::gOrdery = 4;
    TGeoGrid &g = *c.ReferenceGrid();
    Pix i = g.ElementMap().first();
```

```
    TGeoEl *gel;
    TCompEl *cel;
    while(i) {
        gel = (TGeoEl *) g.ElementMap().contents(i);
        g.ElementMap().next(i);
        cel = gel->CreateCompEl();
        long id = cel->Id();
        c.ElementMap()[id] = cel;
    }
    c.CreateDofNodBc();
    an = new TAnalysis(&c);
}
```

Capítulo 4

Elemento de barra

4.1 Teoria da elasticidade para sólidos tridimensionais

Neste capítulo é apresentado um breve estudo de teorias para barras retas no espaço tridimensional formuladas a partir de duas hipóteses cinemáticas sob linearidade geométrica.

No decorrer deste capítulo algumas definições são apresentadas de forma repetitiva. Isto foi adotado para que as duas teorias para barras apresentadas possam ser estudadas independentemente, sem a necessidade de se recorrer a definições encontradas em outros sub-capítulos.

Muitas estruturas têm características geométricas, mecânicas ou de cargas que não permitem a utilização de cálculos simplificados; nesses casos é essencial considerar a estrutura como um sólido tridimensional e fazer uso, para a sua análise, da teoria geral da elasticidade em três dimensões. Alguns elementos que compõem as estruturas têm uma geometria particular que permite algumas simplificações em sua análise: é o caso das barras, onde o comprimento do eixo é muito maior que as dimensões da seção transversal. Neste trabalho, como as barras poderão estar posicionadas em qualquer lugar no espaço tridimensional e serão tratadas como um elemento, sob o qual serão feitas algumas simplificações nos estados de solicitações.

4.1.1 Deslocamentos

Seja um sólido tridimensional como o mostrado na Figura 4.1. O movimento de um ponto no espaço pode ser definido pelas três componentes do vetor de deslocamentos de translação.

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} \tag{4.1}$$

Os valores $\bar{u}_x$, $\bar{u}_y$ e $\bar{u}_z$ de uma partícula para um corpo deformado são as componentes dos deslocamentos do ponto segundo os eixos cartesianos x, y e z, respectivamente.

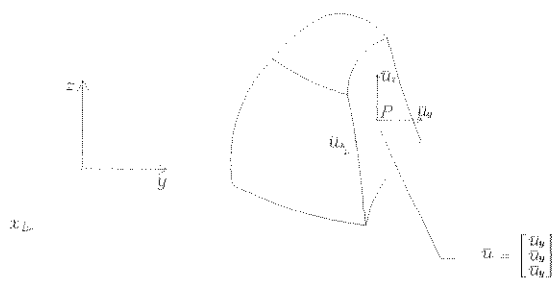

Figura 4.1: Sólido tridimensional. Vetor dos deslocamentos de translação.

4.1.2 Deformações

Pela teoria clássica da elasticidade tridimensional a deformação em um ponto é definida por nove componentes: ε_x, ε_y, ε_z, $\gamma_{xz} = \gamma_{zx}$, $\gamma_{xy} = \gamma_{yx}$ e $\gamma_{yz} = \gamma_{zy}$, o tensor da deformação é reduzido a um vetor de seis componentes, conforme (4.2).

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} \tag{4.2}$$

Considere o corpo elástico com dimensões dx, dy e dz, como mostra a Figura 4.2. O incremento $\triangle dx$, $\triangle dy$ e $\triangle dz$ nas direções x, y e z são originados das deformações normais ε_x, ε_y e ε_z existentes no corpo.

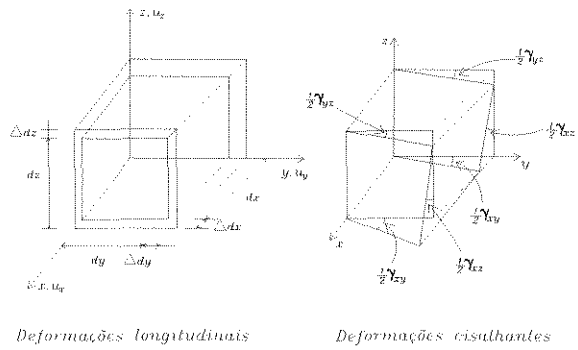

Figura 4.2: Deformações normais e cisalhantes em um corpo.

As deformações cisalhantes apresentadas na Figura 4.2 são responsáveis pela distorção das faces do elemento e são definidas por γ_{yz}, γ_{xz} e γ_{xy}.

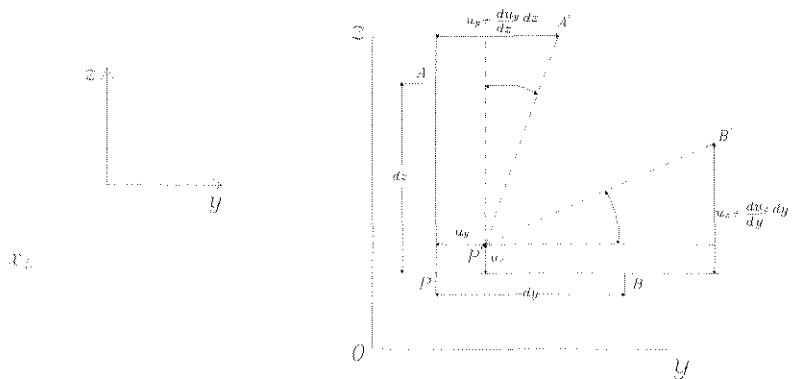

Figura 4.3: Relação entre as componentes de deformação e deslocamento.

A Figura 4.3 ilustra a relação entre as componentes de deformação e

deslocamento, assim de (4.2) o tensor das componentes de deformações pode ser reescrito por:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_z}{\partial z} \\ \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \end{bmatrix} \tag{4.3}$$

4.1.3 Tensões

O tensor das tensões em um ponto é definido por nove componentes de tensão. De modo análogo ao caso das deformações, este tensor pode ser reduzido a seis componentes, pois: $\tau_{xy} = \tau_{yx}$, $\tau_{xz} = \tau_{zx}$ e $\tau_{yz} = \tau_{zy}$ e expressos pelo vetor σ como em (4.4).

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix} \tag{4.4}$$

Em (4.4) os valores de σ_x, σ_y e σ_z expressam as tensões normais e τ_{yz}, τ_{xz} e τ_{xy} são as tensões tangenciais aplicadas ao corpo.

4.1.4 Relação tensão-deformação

A relação entre as seis deformações e as seis tensões é expressa no caso mais geral da elasticidade anisotrópica por uma matriz constitutiva de tamanho 6 × 6, simétrica, e com 21 coeficientes independentes.

Um caso mais simplificado é o do material ortotrópico, onde nove constantes elásticas independentes intervêm no comportamento do material, uma vez que, dada a simetria da matriz constitutiva, tem-se que:

$$\begin{aligned} E_x\nu_{yx} &= E_y\nu_{xy} \\ E_y\nu_{zy} &= E_z\nu_{yz} \\ E_z\nu_{xz} &= E_x\nu_{zx} \end{aligned} \tag{4.5}$$

Dessa forma, as equações constitutivas da teoria da elasticidade, segundo Chen & Han[9], para um material homogêneo e isotrópico, podem ser simplificadas por:

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2G\varepsilon_{ij} \tag{4.6}$$

onde σ_{ij} são as componentes do tensor das tensões, δ_{ij} o delta de *Kronecker*, λ a constante de *Lamé* definida por $\lambda = \frac{\nu E}{(1-\nu^2)}$.

Pode-se ainda escrever a relação tensão-deformação (4.6) por:

$$\sigma = D\varepsilon \tag{4.7}$$

Neste trabalho considera-se o material elástico isotrópico linear, onde o módulo de *Young* no plano de isotropia é representado por E. O coeficiente de *Poisson* ν representa uma redução na deformação transversal no plano de isotropia devida à tensão de tração no mesmo plano.

A matriz D apresentada em (4.7) é denominada matriz constitutiva elástica do material, e com as hipóteses aqui assumidas é definida por:

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \tag{4.8}$$

4.2 Princípio dos trabalhos virtuais

O *Princípio dos Trabalhos Virtuais* tem se mostrado uma poderosa técnica para a solução de diversos problemas freqüentes na engenharia e principalmente na mecânica dos sólidos. A seguir é apresentado de forma genérica tal princípio, o qual possibilita maior compreensão no desenvolvimento

e na obtenção das equações necessárias para subseqüentes considerações na expressão da *Energia Potencial Total* dos elementos de barra e de placa apresentados nesse trabalho.

Seja um corpo deformável Ω, como ilustra a Figura 4.4, cujo contorno é dado por Γ, submetido a um conjunto de forças de corpo F_i no domínio de Ω, forças T_i no contorno Γ_A e restrições aos deslocamentos no contorno Γ_B.

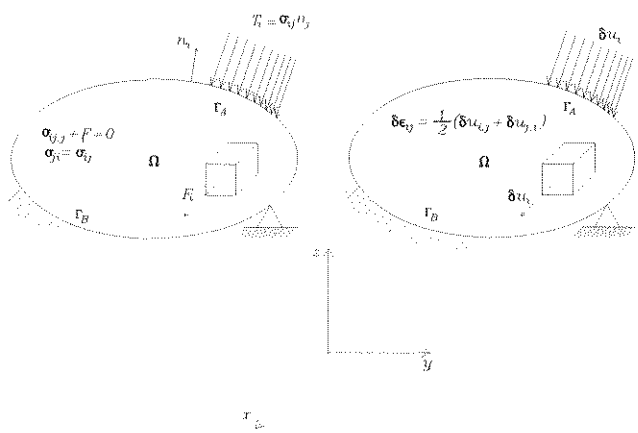

Figura 4.4: Corpo elástico em equilíbrio.

As equações de equilíbrio no corpo em uma dada direção são expressas por:

$$\begin{aligned} \sigma_{ij,j} + F_i &= 0 \text{ em } \Omega \\ \sigma_{ji} &= \sigma_{ij} \end{aligned} \tag{4.9}$$

onde i e j variam no intervalo de 1 a 3, respectivamente associados às direções x, y e z.

Para que o corpo esteja em equilíbrio é necessário que (4.9) seja satisfeita quando sujeita às condições de contorno abaixo descritas, onde h_i são as componentes dos deslocamentos prescritos no contorno Γ_B e n_i são as componentes do vetor normal à superfície Γ.

$$\begin{aligned} u_i &= h_i \text{ em } \Gamma_B \\ \sigma_{ij} n_j &= T_i \text{ em } \Gamma_A \end{aligned} \tag{4.10}$$

4.2.1 Equação do princípio dos trabalhos virtuais

O termo virtual é usado para designar o trabalho realizado por forças externas (*verdadeiras*) com uma variação de deslocamentos (*deslocamentos imaginários, virtuais*). Em (4.11) as quantidades T_i e F_i são forças de superfície externa e de corpo, respectivamente.

Admitindo um estado de forças em equilíbrio e um estado independente de deslocamentos compatíveis e aplicando-se o Princípio dos Trabalhos Virtuais entre eles, tem-se:

$$\int_V \sigma_{ij}\delta\varepsilon_{ij}dV = \int_A T_i\delta u_i dA + \int_V F_i\delta u_i dV \tag{4.11}$$

Similarmente, as deformações $\delta\varepsilon_{ij}$ representam um conjunto de deformações compatíveis com o deslocamento real ou imaginário δu_i, para os pontos de aplicação das forças externas T_i e F_i.

$$\sigma_{ij}n_j = T_i \tag{4.12}$$

Ao substituir (4.12) no lado direito de (4.11), tem-se o trabalho externo das forças para uma variação de deslocamentos.

$$W_{ext} = \int_A \sigma_{ij}n_j\delta u_i dA + \int_V F_i\delta u_i dV \tag{4.13}$$

A primeira integral pode ser transformada em integral de volume; assim, pela aplicação do *Teorema da Divergência*, obtém-se:

$$\begin{aligned} W_{ext} &= \int_V (\sigma_{ij}\delta u_i)_{,j}dV + \int_V F_i\delta u_i dV \\ &= \int_V (\sigma_{ij,j}\delta u_i + \sigma_{ij}\delta u_{i,j})dV + \int_V F_i\delta u_i dV \end{aligned} \tag{4.14}$$

Pode-se escrever ainda (4.14) na forma:

$$W_{ext} = \int_V \left[(\sigma_{ij,j} + F_i)\,\delta u_i + \sigma_{ij}\delta u_{i,j}\right] dV \tag{4.15}$$

O primeiro termo entre parênteses é anulado pela condição de equilíbrio, a qual satisfaz as equações de equilíbrio mencionadas. Dessa maneira, o

trabalho externo W_{ext} é escrito por:

$$W_{ext} = \int_V \sigma_{ij} \delta u_{i,j} dV \tag{4.16}$$

Considerando o trabalho interno virtual W_{int}, dado pelo lado esquerdo de (4.11), tem-se:

$$\begin{aligned} W_{int} &= \int_V \sigma_{ij} \delta \varepsilon_{ij} dV = \int_V \frac{1}{2} \sigma_{ij} (\delta u_{i,j} + \delta u_{j,i}) dV \\ &= \int_V (\frac{1}{2} \sigma_{ij} \delta u_{i,j} + \frac{1}{2} \sigma_{ij} \delta u_{j,i}) dV \end{aligned} \tag{4.17}$$

Por fim, pela simetria de σ_{ij}, chega-se à expressão inicial do trabalho virtual interno W_{int}.

$$W_{int} = \int_V \sigma_{ij} \delta u_{i,j} dV \tag{4.18}$$

Assim fica estabelecido o *Princípio dos Trabalhos Virtuais*, o qual demostra que $W_{int} = W_{ext}$.

4.3 Propriedades geométricas da seção transversal e sistema de coordenadas para elementos de barra

Nesta seção são apresentadas algumas propriedades geométricas de uma seção transversal qualquer, bem como o sistema de coordenadas de um elemento de barra no espaço tridimensional.

4.3.1 Sistema de coordenadas genérico e convenção de sinais

Define-se um sistema de coordenadas onde os pontos da seção transversal de uma barra são representados conforme ilustra a Figura 4.5.

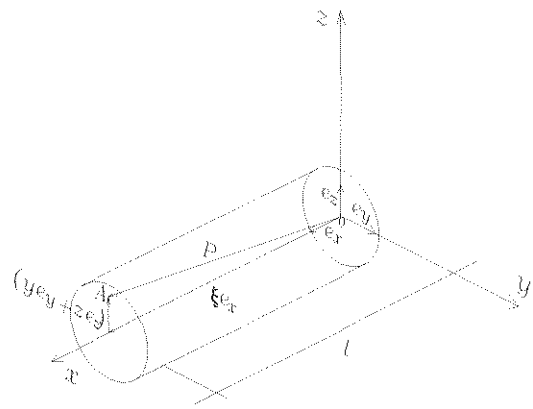

Figura 4.5: Sistema de coordenadas de uma barra no espaço.

Dessa maneira, as coordenadas de um ponto material qualquer numa seção transversal de uma barra podem ser expressas vetorialmente por:

$$\vec{\mathbf{x}} = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z \tag{4.19}$$

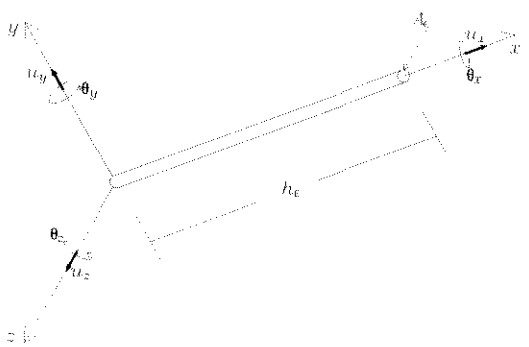

Figura 4.6: Conveção de sinais adotada para elemento de barra com seis graus de liberdade por nó.

Adota-se para convenção de sinais das rotações a regra da mão-direita

aplicada sobre os eixos cartesianos positivos. Os deslocamentos no elemento de barra são representados nesse capítulo por três translações: u_x, u_y e u_z nas direções dos eixos cartesianos x, y e z e três rotações θ_x, θ_y e θ_z em torno dos eixos x, y e z; portanto, têm-se seis graus de liberdade por nó.

Essa convenção de sinais, ilustrada na Figura 4.6, é conveniente para as aspirações de se estender a teoria para aplicações no espaço tridimensional e acoplamento com elemento de placa.

4.3.2 Propriedades geométricas

As propriedades geométricas são implementadas a partir de uma seção transversal indeformável. Inicialmente são estudadas as propriedades como área, momentos estáticos e momentos de inércia para uma dada seção de uma barra.

Esses conceitos e propriedades são apresentados aqui apenas para que se compreenda de forma consistente todas as deduções envolvendo a posição do eixo da barra frente as hipóteses cinemáticas adotadas.

A área da seção transversal é dada por:

$$A = \int_A dA = \int_A dydz \tag{4.20}$$

As coordenadas do centro de gravidade da seção transversal da barra são:

$$\begin{aligned} \bar{y} &= \frac{1}{A}\int_A ydA \\ \bar{z} &= \frac{1}{A}\int_A zdA \end{aligned} \tag{4.21}$$

Os momentos estáticos são relativos aos dois eixos que definem a seção transversal e são expressos por:

$$\begin{aligned} S_y &= \int_A zdA \\ S_z &= -\int_A ydA \end{aligned} \tag{4.22}$$

A partir de (4.21) e (4.22), obtêm-se as expressões:

$$\begin{aligned} \bar{y} &= -\frac{1}{A} S_z \\ \bar{z} &= \frac{1}{A} S_y \end{aligned} \tag{4.23}$$

Assim, os momentos estáticos podem ser reescritos como:

$$\begin{aligned} S_y &= \bar{z} A \\ S_z &= -\bar{y} A \end{aligned} \tag{4.24}$$

Outras propriedades geométricas importantes são os momentos de inércia da seção transversal, definidos por:

$$\begin{aligned} J_{yy} &= \int_A z^2 dA \\ J_{zz} &= \int_A y^2 dA \\ J_{yz} &= -\int_A yz dA \end{aligned} \tag{4.25}$$

Para os eixos centrais a posição dos pontos da seção transversal são:

$$\begin{aligned} y^{CG} &= y - \bar{y} \\ z^{CG} &= z - \bar{z} \end{aligned} \tag{4.26}$$

Os momentos estáticos definidos para estes eixos centrais resultam em:

$$\begin{aligned} S_y^{CG} &= \int_A z^{CG} dA = \int_A (z - \bar{z})\, dA = S_y - S_y = 0 \\ S_z^{CG} &= -\int_A y^{CG} dA = -\int_A (y - \bar{y})\, dA = -S_z + S_z = 0 \end{aligned} \tag{4.27}$$

Os momentos de inércia para os eixos centrais são expressos pelas seguintes componentes:

$$\begin{aligned} J_{yy}^{CG} &= J_{yy} - A\bar{z}^2 \\ J_{zz}^{CG} &= J_{zz} - A\bar{y}^2 \\ J_{yz}^{CG} &= J_{yz} + A\bar{y}\bar{z} \end{aligned} \tag{4.28}$$

Logo, de (4.28), resulta que para eixos quaisquer, segundo a regra de Steiner, os momentos de inércia são representados por:

$$\begin{aligned} J_{yy} &= J_{yy}^{CG} + A\bar{z}^2 \\ J_{zz} &= J_{zz}^{CG} + A\bar{y}^2 \\ J_{yz} &= J_{yz}^{CG} - A\bar{y}\bar{z} \end{aligned} \tag{4.29}$$

A expressão do momento polar para eixos centrais é:

$$J_o^{CG} = J_o - A\bar{y}\bar{y} - A\bar{z}\bar{z} = J_o - A\left(\bar{y}^2 + \bar{z}^2\right) \tag{4.30}$$

Para o caso de eixos com coordenadas quaisquer, o momento polar de inércia é expresso por:

$$J_o = J_o^{CG} + A\bar{y}\bar{y} + A\bar{z}\bar{z} = J_o^{CG} + A\left(\bar{y}^2 + \bar{z}^2\right) \tag{4.31}$$

4.4 Introdução à teoria de vigas

A seguir são apresentadas duas teorias para análise de vigas para elementos finitos unidimensionais: a *Teoria de Viga de Euler-Bernoulli* e a *Teoria de Viga de Timoshenko*. Ambas possuem algumas considerações em comum, diferenciando-se no trato das funções de forma onde a teoria de Euler-Bernoulli necessita de continuidade C^1e a teoria de Timoshenko não exige mais do que funções de forma com continuidade C^0.

Na verdade, a diferença entre as teorias reside no fato da teoria de Euler-Bernoulli não considerar a deformação por cisalhamento, enquanto a teoria de Timoshenko faz uso da deformação por cortante para o cálculo dos deslocamentos.

4.4.1 Teoria de Euler-Bernoulli

Considere a viga, ilustrada na Figura 4.7, de comprimento l, seção transversal de área A e momentos de inércia a flexão J_{yy} e J_{zz}, sobre a qual atuam uma série de ações e momentos externos.

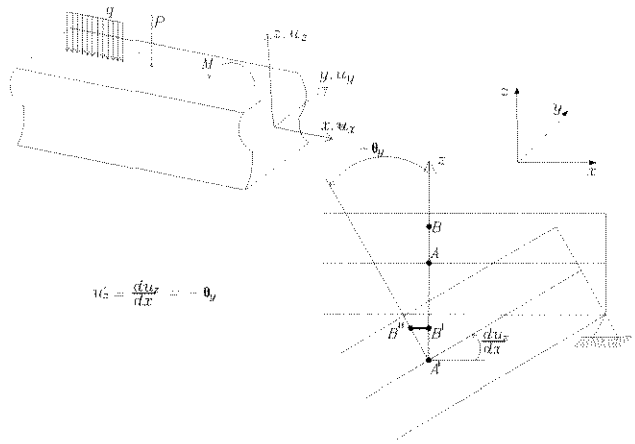

Figura 4.7: Viga convencional de Euler-Bernoulli.

Hipótese cinemática

Na *Teoria Clássica de Barras* também conhecida como *Teoria Clássica de Barras*, mostrada na Figura 4.7, admite-se que as seções transversais normais ao eixo da viga antes da deformação, permanecem planas e ortogonais a esta após a deformação.

O vetor deslocamento $\bar{u}$ pode ser genericamente expresso por:

$$\bar{u} = u + \theta \times a \tag{4.32}$$

Onde:
u: é o deslocamento dos pontos do eixo;
θ: é a rotação da seção transversal;
a: posição dos pontos na seção transversal, definido por:

$$a = y\vec{e}_y + z\vec{e}_z \tag{4.33}$$

A Figura 4.8 ilustra a cinemática dos deslocamentos das fibras em uma seção transversal genérica de um elemento de barra.

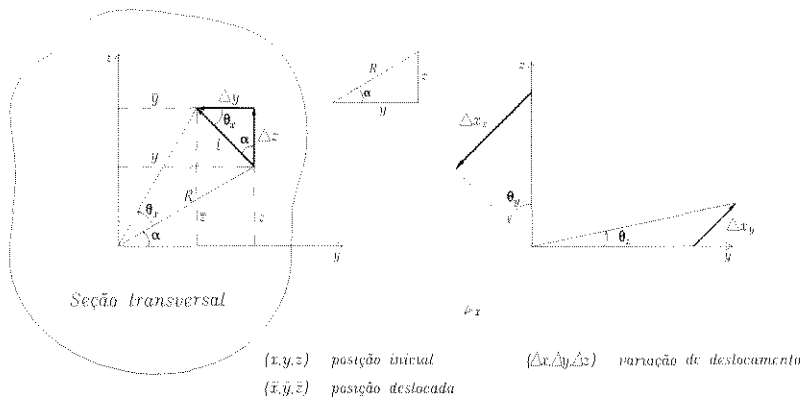

Figura 4.8: Cinemática de deslocamentos de um ponto da seção transversal genérica de uma barra.

Conforme a Figura 4.8, onde R é o vetor posição, tem-se que:

$$y = R\cos\alpha \qquad \text{e} \qquad z = R\sin\alpha \tag{4.34}$$

Por simplificações geométricas considera-se $\ell = R.\theta_x$. Com isso, as variações de deslocamentos na fibra podem ser escritas na forma:

$$\begin{aligned} \Delta x &= \Delta x_z + \Delta x_y = z\theta_y - y\theta_z \\ \Delta y &= \ell\sin\alpha = R\sin\alpha\theta_x = -z\theta_x \\ \Delta z &= \ell\cos\alpha = R\cos\alpha\theta_x = y\theta_x \end{aligned} \tag{4.35}$$

Os sinais negativos que aparecem em (4.35) representam variações de deslocamentos com sentido contrário ao convencionado.

O deslocamento total de um ponto em uma seção transversal pode ser representado, conforme Figura 4.8, por:

$$\begin{aligned} \bar{u}_x &= u_x + \Delta x \\ \bar{u}_y &= u_y + \Delta y \\ \bar{u}_z &= u_z + \Delta z \end{aligned} \tag{4.36}$$

Da hipótese mencionada e das considerações expressas em (4.36), chega-se à expressão que representa a cinemática de deslocamentos de um ponto na seção transversal de um elemento.

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} = \begin{bmatrix} u_x + z\theta_y - y\theta_z \\ u_y - z\theta_x \\ u_z + y\theta_x \end{bmatrix} \tag{4.37}$$

Pela hipótese da teoria de *Euler-Bernoulli*, tem-se que:

$$u'_y = \theta_z \qquad \text{e} \qquad u'_z = -\theta_y \tag{4.38}$$

A associação entre deslocamentos e rotações descrita em (4.38) dá-se o nome de vínculo de *Euler-Bernoulli*. A partir da expressão anterior tem-se que os deslocamentos podem ser reescritos por:

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} = \begin{bmatrix} u_x - zu'_z - yu'_y \\ u_y - z\theta_x \\ u_z + y\theta_x \end{bmatrix} \tag{4.39}$$

Deformações

A matriz jacobiana de deslocamentos, partindo de (4.39), é dada pela seguinte expressão matricial:

$$\jmath = \frac{\partial(\bar{u}_x, \bar{u}_y, \bar{u}_z)}{\partial(x, y, z)} = \begin{bmatrix} \left(u'_x - zu''_z - yu''_y\right) & -u'_y & -u'_z \\ \left(u'_y - z\theta'_x\right) & 0 & -\theta_x \\ \left(u'_z + y\theta'_x\right) & \theta_x & 0 \end{bmatrix} \tag{4.40}$$

Onde, dada a função genérica w, sua derivada em relação a x é representada por $w' = \frac{\partial w}{\partial x}$.

Admitindo a hipótese de pequenos deslocamentos e deformações, o tensor das deformações E sob linearidade geométrica é definido por:

$$\mathbf{E} = \frac{1}{2}\left(\jmath + \jmath^T\right) \tag{4.41}$$

Substituindo (4.40) em (4.41), resulta:

$$\mathbf{E} = \begin{bmatrix} \left(u'_x - zu''_z - yu''_y\right) & \left(-\frac{1}{2}z\theta'_x\right) & \left(\frac{1}{2}y\theta'_x\right) \\ \left(-\frac{1}{2}z\theta'_x\right) & 0 & 0 \\ \left(\frac{1}{2}y\theta'_x\right) & 0 & 0 \end{bmatrix} \tag{4.42}$$

Admite-se que $\varepsilon_x = E_{xx}$, $\gamma_{xy} = 2E_{xy}$ e $\gamma_{xz} = 2E_{xz}$. Aplicando-se essas expressões em (4.42) resulta no vetor das deformações, expresso a seguir que, em termos de componentes, elas são escritas por

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} u'_x - yu''_y - zu''_z \\ 0 \\ 0 \\ 0 \\ y\theta'_x \\ -z\theta'_x \end{bmatrix} \tag{4.43}$$

onde os valores não nulos γ_{xz} e γ_{xy} são as distorções e ε_x é a deformação axial da barra.

Tensões

O tensor das tensões pode ser escrito através de suas componentes não nulas na forma:

$$\vec{T} = \sigma_x \vec{e}_x + \tau_{xy}\vec{e}_y + \tau_{xz}\vec{e}_z \tag{4.44}$$

onde σ_x é a tensão normal ao plano formado pelos vetores $\vec{e}_y$ e $\vec{e}_z$, τ_{xy} e τ_{xz} são as tensões cisalhantes atuantes no plano normal ao vetor $\vec{e}_x$.

As componentes das tensões são representadas por:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix} \tag{4.45}$$

Esforços solicitantes

Os esforços solicitantes em um elemento de barra são forças cortantes V_{xy} e V_{xz}, força normal N_x, momentos fletores M_y e M_z e momento torçor M_x, definidos por:

$$\begin{aligned} N_x &= \int_A \sigma_x dA \\ V_{xy} &= \int_A \tau_{xy} dA \\ V_{xz} &= \int_A \tau_{xz} dA \\ M_x &= \int_A (y\tau_{xz} - z\tau_{xy})\, dA \\ M_y &= \int_A z\sigma_x dA \\ M_z &= \int_A y\sigma_x dA \end{aligned} \tag{4.46}$$

Relação tensão-deformação

A equação constitutiva da elasticidade é formalmente dada por:

$$\sigma = D\epsilon \tag{4.47}$$

No sistema local da barra, a matriz D é escrita por

$$\mathbf{D} = \begin{bmatrix} EA & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & GJ_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & EJ_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & EJ_{zz} \end{bmatrix} \tag{4.48}$$

onde A, E, J_o, J_{yy} e J_{zz} são a área da seção transversal da barra, módulo de elasticidade longitudinal, momento polar de inércia e momentos de inércia em relação aos eixos principais y e z respectivamente.

Esforços e deformações generalizadas

Na *Teoria Clássica de Barras*, o vetor dos esforços internos ou tensões generalizadas $\hat{\sigma}$ é representado por:

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} N_x \\ V_{xy} \\ V_{xz} \\ M_x \\ M_y \\ M_z \end{bmatrix} = \int_A \begin{bmatrix} \sigma_x \\ \tau_{xy} \\ \tau_{xz} \\ (y\tau_{xz} - z\tau_{xy}) \\ z\sigma_x \\ y\sigma_x \end{bmatrix} dA \tag{4.49}$$

Para a formulação das equações constitutivas utilizam-se as seguintes relações elásticas lineares:

$$\begin{aligned} \sigma_x &= E\varepsilon_x \\ \sigma_y &= E\varepsilon_y \\ \sigma_y &= E\varepsilon_z \\ \tau_{yz} &= G\gamma_{yz} \\ \tau_{xz} &= G\gamma_{xz} \\ \tau_{xy} &= G\gamma_{xy} \end{aligned} \tag{4.50}$$

Substituindo (4.50) em (4.49) e lembrando que as forças cortantes não produzem trabalho, obtêm-se os esforços solicitantes em um elemento de barra. Ressalta-se também que a excentricidade não está sendo considerada nessa formulação.

$$\begin{aligned} N_x &= \int_A E\varepsilon_x dA = \int_A E\left(u'_x - zu''_z - yu''_y\right) dA = EAu'_x \\ V_{xy} &= 0 \\ V_{xz} &= 0 \\ M_x &= \int_A G\left(y\gamma_{xz} - z\gamma_{xy}\right) dA = \int_A \left[G\left(y^2 + z^2\right)\theta'_x\right] dA = GJ_0\theta'_x \\ M_y &= \int_A zE\varepsilon_x dA = \int_A zE\left(u'_x - zu''_z - yu''_y\right) dA = -EJ_{yy}u''_z \\ M_z &= \int_A yE\varepsilon_x dA = \int_A yE\left(u'_x - zu''_z - yu''_y\right) dA = -EJ_{zz}u''_y \end{aligned} \tag{4.51}$$

Assim, o vetor das tensões generalizadas pode ser reescrito por

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} EAu'_x \\ 0 \\ 0 \\ GJ_0\theta'_x \\ -EJ_{yy}u''_z \\ -EJ_{zz}u''_y \end{bmatrix} = D\hat{\varepsilon} \tag{4.52}$$

onde o vetor das deformações generalizadas $\hat{\varepsilon}$, em (4.52), é definido por:

$$\hat{\boldsymbol{\varepsilon}} = \begin{bmatrix} u'_x \\ 0 \\ 0 \\ \theta'_x \\ -u''_z \\ u''_y \end{bmatrix} \tag{4.53}$$

Ações

No presente trabalho as ações em um elemento de barra podem ser forças e momentos concentrados nos nós ou distribuídos ao longo do eixo da barra. As forças e momentos pontuais são aplicadas nos nós nas direções x, y e z globais. As forças e momentos distribuídos são aplicados ao longo do eixo da barra nas direções x, y e z local em relação a cada lado do elemento.

A convenção de sinais é a mesma utilizada nas formulações, expressa pela regra da mão direita.

As forças de corpo f_x, f_y e f_z apresentadas na formulação a seguir para o elemento de barra, expressam forças por unidade de volume.

$$\underbrace{\begin{aligned} p_x &= \textstyle\int_A f_x dA \\ p_y &= \textstyle\int_A f_y dA \\ p_z &= \textstyle\int_A f_z dA \\ m_x &= \textstyle\int_A (y f_z - z f_y) f_x dA \\ m_y &= \textstyle\int_A z f_x dA \\ m_z &= \textstyle\int_A y f_x dA \end{aligned}}_{\textit{Forças e momentos distribuídos / comprimento}} \quad \text{e} \quad \underbrace{\begin{aligned} F_1 &\ \text{(força na direção x)} \\ F_2 &\ \text{(força na direção y)} \\ F_3 &\ \text{(força na direção z)} \\ M_1 &\text{(momento em torno x)} \\ M_2 &\text{(momento em torno y)} \\ M_3 &\text{(momento em torno z)} \end{aligned}}_{\textit{Forças e momentos concentrados}} \tag{4.54}$$

Formulação variacional

O trabalho virtual dos esforços internos de uma barra é igual ao trabalho virtual das ações e, sob condições de contorno essenciais, pode-se demonstrar sua equivalência com as equações diferenciais de equilíbrio.

Na formulação que segue para facilitar a notação, eventualmente serão usados os índices 1, 2, 3 em correspondencia a x, y, z.

$$\int_\Omega \delta\varepsilon_{ij}\sigma_{ij} d\Omega - \int_\Omega \delta\bar{u}_i F_i d\Omega - \int_\Gamma \delta\bar{u}_i T_i d\Gamma = 0 \tag{4.55}$$

O domínio pode ser representado pela soma dos sub-domínios e ao considerar $dA = dydz$ pode-se escrever:

$$\int_\Omega \ldots d\Omega = \sum_{e=1}^{n_{el}} \int_{\Omega_e} \ldots d\Omega_e = \sum_{e=1}^{n_{el}} \int_0^{h_e} \int_{A_e} \ldots dA dx_e \tag{4.56}$$

Conforme Reddy[25], aplicam-se as deformações existentes e ainda com as considerações estabelecidas em (4.56) tem-se que:

$$\begin{aligned} 0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_0^{h_e} \int_{A_e} \left(\delta\varepsilon_x \sigma_x + \delta\gamma_{xy}\tau_{xy} + \delta\gamma_{xz}\tau_{xz} \right) dA dx_e \right. \\ & - \int_0^{h_e} \int_{A_e} \left(\delta\bar{u}_x f_x + \delta\bar{u}_y f_y + \delta\bar{u}_z f_z \right) dA dx_e \\ & \left. - \left[\sum_{k=1}^{3} \delta u_k^T F_k + \delta\theta_x M_x + \sum_{k=2}^{3} \delta u_k'^T M_k \right] \right\} \end{aligned} \tag{4.57}$$

Substituindo-se (4.43) em (4.57) obtém-se:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \Bigg\{ \int_0^{h_e} \int_{A_e} \left[\left(\delta u_x' - z\delta u_z'' - y\delta u_y'' \right) \sigma_x + \left(-z\delta\theta_x' \right) \tau_{xy} \right. \\
& \left. + \left(y\delta\theta_x' \right) \tau_{xz} \right] dA dx_e - \int_0^{h_e} \int_{A_e} \left[\left(\delta u_x - z\delta u_z' - y\delta u_y' \right) f_x \right. \\
& \left. + \left(\delta u_y - z\delta\theta_x \right) f_y + \left(\delta u_z + y\delta\theta_x \right) f_z \right] dA dx_e \\
& - \left[\sum_{k=1}^{3} \delta u_k^T F_k + \delta\theta_x M_x + \sum_{k=2}^{3} \delta u_k'^T M_k \right] \Bigg\}
\end{aligned}
\tag{4.58}
$$

Reordenando (4.58) é possível escrevê-la em função de deslocamentos virtuais associados aos correspondentes esforços:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \Bigg\{ \int_0^{h_e} \Bigg[\delta u_x' \underbrace{\int_{A_e} \sigma_x dA}_{N_x} - \delta u_z'' \underbrace{\int_{A_e} z\sigma_x dA}_{M_y} - \delta u_y'' \underbrace{\int_{A_e} y\sigma_x dA}_{M_z} \\
& + \delta\theta_x' \underbrace{\int_{A_e} \left(y\tau_{xz} - z\tau_{xy} \right) dA}_{M_x} \Bigg] dx_e - \int_0^{h_e} \Bigg[\delta u_x \underbrace{\int_{A_e} f_x dA}_{p_x} \\
& + \delta u_y \underbrace{\int_{A_e} f_y dA}_{p_y} \delta u_z + \underbrace{\int_{A_e} f_z dA}_{p_z} + \delta\theta_x \underbrace{\int_{A_e} \left(y f_z - z f_y \right) dA}_{m_x} \\
& + \delta\theta_y \underbrace{\int_{A_e} z f_x dA}_{m_y} - \delta\theta_z \underbrace{\int_{A_e} y f_x dA}_{m_z} \Bigg] dx_e \\
& - \left[\sum_{k=1}^{3} \delta u_k^T F_k + \delta\theta_x M_x + \sum_{k=2}^{3} \delta u_k'^T M_k \right] \Bigg\}
\end{aligned}
\tag{4.59}
$$

Ao substituir (4.51) em (4.59) tem-se que:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_0^{h_e} \left[\delta u_x' E A u_x' + \delta \theta_x' G J_0 \theta_x' + \delta u_z'' E J_{yy} u_z'' + \delta u_y'' E J_{zz} u_y'' \right] dx_e \right. \\
& - \int_0^{h_e} \left[\delta u_x p_x + \delta u_y p_y + \delta u_z p_z + \delta \theta_x m_x + \delta \theta_y m_y - \delta \theta_z m_z \right] dx_e \\
& \left. - \left[\sum_{k=1}^{3} \delta u_k^T F_k + \delta \theta_x M_x + \sum_{k=2}^{3} \delta u_k'^T M_k \right] \right\} \qquad (4.60)
\end{aligned}
$$

O sistema é simétrico, bilinear e pode ser escrito na forma.

$$
a(\bar{u}, u) = (\bar{u}, f) + (\bar{u}, t)_\Gamma \qquad (4.61)
$$

Onde:

$$
\begin{aligned}
a(\bar{u}, u) \;&=\; \sum_{e=1}^{n_{el}} \int_0^{h_e} \left[\delta u_x' E A u_x' + \delta \theta_x' G J_0 \theta_x' + \delta u_z'' E J_{yy} u_z'' + \delta u_y'' E J_{zz} u_y'' \right] dx_e \\
(\bar{u}, f) \;&=\; \sum_{e=1}^{n_{el}} \int_0^{h_e} \left[\delta u_x p_x + \delta u_y p_y + \delta u_z p_z + \delta \theta_x m_x + \delta \theta_y m_y - \delta \theta_z m_z \right] dx_e \\
(\bar{u}, t)_\Gamma \;&=\; \sum_{e=1}^{n_{el}} \left[\sum_{k=1}^{3} \delta u_k^T F_k + \delta \theta_x M_x + \sum_{k=2}^{3} \delta u_k'^T M_k \right] \qquad (4.62)
\end{aligned}
$$

Discretização utilizando o método dos elementos finitos

Para a obtenção da equação de elementos finitos de barra de Euler-Bernoulli, necessita-se escrever os deslocamentos u em termos de deslocamentos nodais u_p, mediante a definição de uma base de funções de interpolação, da seguinte forma

$$
u = N u_p \qquad (4.63)
$$

onde N é a matriz de funções de interpolação, utilizada para encontrar a solução numérica aproximada das grandezas físicas do problema.

A matriz dos coeficientes de interpolação N_i que aparece nas expressões abaixo, tem seu índice i dado pela ordem do polinômio de interpolação utilizado na aproximação.

O vetor de deformações generalizadas $\hat{\varepsilon}$, como visto em (4.53) pode ser escrito na forma matricial por:

$$\hat{\boldsymbol{\varepsilon}} = \begin{bmatrix} u'_x \\ 0 \\ 0 \\ \theta'_x \\ -u''_z \\ u''_y \end{bmatrix} = \sum_{i=1}^{n_{el}} \begin{bmatrix} N'_i u_{x_i} \\ 0 \\ 0 \\ N'_i \theta_{x_i} \\ -N''_i u_{z_i} \\ N''_i u_{y_i} \end{bmatrix} = \sum_{i=1}^{n_{el}} B_i \begin{bmatrix} u_{x_i} \\ u_{y_i} \\ u_{z_i} \\ \theta_{x_i} \\ \left(\frac{\partial u_z}{\partial x}\right)_i \\ \left(\frac{\partial u_y}{\partial x}\right)_i \end{bmatrix} = \sum_{i=1}^{n_{el}} B_i d_i \quad (4.64)$$

Para a implementação computacional faz-se necessário escrever a matriz B que representa o operador diferencial aplicado às funções de interpolação da matriz de deformação generalizada.

A partir de (4.64) define-se a matriz B por:

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{\partial^2 N_i}{\partial x^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial^2 N_i}{\partial x^2} \end{bmatrix} \quad (4.65)$$

Por fim, substituindo-se (4.65) e (4.48) em (4.60) obtém-se a formulação variacional do problema expressa em termos das matrizes B e D.

$$\begin{aligned} 0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_0^{h_e} \left(B^T D B\right) dx_e \right. \\ & - \int_0^{h_e} \left[N_i p_x + N_i p_y + N_i p_z + N_i m_x + N_i m_y - N_i m_z\right] dx_e \\ & \left. - \left[\sum_{k=1}^{3} \delta N_k^T F_k + \delta N_i M_x + \sum_{k=2}^{3} \delta N_k'^T M_k \right] \right\} \end{aligned} \quad (4.66)$$

Como os esforços cisalhantes não são considerados na *Teoria Clássica de Barras*, a matriz de rigidez e vetor de cargas para um elemento de barra, a partir das definições apresentadas, são dados por:

$$\begin{aligned} K^e &= \int_{h_e} \left(B^T D B\right) dx_e \\ f^e &= \begin{cases} \int_0^{h_e} N_a p_i dx_e \\ (-1)^{i+1} \int_0^{h_e} N_a m_i dx_e \end{cases} \end{aligned} \tag{4.67}$$

Ao mapear para o sistema de coordenadas padrão definido por ξ, o diferencial de comprimento do elemento dx é:

$$dx_e = J^e d\xi \tag{4.68}$$

O jacobiano de transformação do elemento de barra é dado por $J^e = \frac{dx_e}{d\xi}$.

Ressalta-se que o jacobiano unidimensional para coordenadas cartesianas é apresentado e detalhado no final deste capítulo abrangendo, assim, as duas teorias de barras apresentadas.

Com isso, (4.66) pode ser reescrita como segue:

$$K_i^e = \int_\xi \left(B^T D B\right) J^e d\xi \tag{4.69}$$

$$f_i^e = \int_\xi \left[N_i p_x + N_i p_y + N_i p_z + N_i m_x + N_i m_y - N_i m_z\right] J^e d\xi \tag{4.70}$$

Avaliando-se numericamente (4.69) e (4.70), utilizando a regra de quadratura simétrica de Gauss para os elementos de barra, tem-se a matriz de rigidez e vetor de carga para cada elemento finito dados por:

$$K_i^e \approx \sum_{k=1}^{n} \left[\left(B^T D B\right) J^e\right]_{\xi=\xi_k} W_k \tag{4.71}$$

$$f_i^e \approx \sum_{k=1}^{n} \left[\left(N_i p_x + N_i p_y + N_i p_z + N_i m_x + N_i m_y - N_i m_z\right) J^e\right]_{\xi=\xi_k} W_k \tag{4.72}$$

Onde W_k representa o peso do k-ésimo ponto de integração e n o número total de pontos de integração em função da ordem do polinômio do elemento.

Observações

1. A hipótese de *Euler-Bernoulli* traz inconvenientes para o método dos elementos finitos, pois u'_y e u'_z passam a ser graus de liberdade da barra exigindo funções de interpolação mais complexas para estes componentes de deslocamentos no sistema local. Dada a associação entre a variável deslocamento e a variável rotação caracterizada na *Teoria de Euler-Bernoulli*, exige-se para sua solução uma família de polinômios interpoladores de continuidade C^1.

2. Em (4.60) os termos de variações que aparecem são usados como função teste.

4.4.2 Teoria de Timoshenko

Diferentemente da hipótese de *Euler-Bernoulli*, na *Teoria de Timoshenko* as rotações são tratadas independentemente dos deslocamentos, devendo ser consideradas as distorções por força cortante.

Hipótese cinemática

As seções planas normais ao eixo da viga antes da deformação permanecem planas, mas não necessariamente normais ao eixo após a deformação.

Neste caso, o vetor deslocamento $\bar{u}$ pode ser expresso por:

$$\bar{\mathbf{u}} = u + \theta \times a \tag{4.73}$$

Onde:

u: é o deslocamento do eixo;

θ: é a rotação da seção transversal;

a: posição do ponto na seção transversal e o símbolo $\times$ representa o produto vetorial.

$$a = y\vec{e}_y + z\vec{e}_z \tag{4.74}$$

Análogo ao estudo da *Teoria Clássica para Barras*, o vetor deslocamentos pode ser expresso algebricamente por suas componentes, conforme ilustra a Figura 4.8 por:

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} = \begin{bmatrix} u_x + z\theta_y - y\theta_z \\ u_y - z\theta_x \\ u_z + y\theta_x \end{bmatrix} \tag{4.75}$$

Deformações

A matriz jacobiana de deslocamentos, a partir de (4.75), é dada pela seguinte expressão matricial:

$$\jmath = \frac{\partial(\bar{u}_x, \bar{u}_y, \bar{u}_z)}{\partial(x, y, z)} = \begin{bmatrix} (u'_x + z\theta'_y - y\theta'_z) & -\theta_z & \theta_y \\ (u'_y - z\theta'_x) & 0 & -\theta_x \\ (u'_z + y\theta'_x) & \theta_x & 0 \end{bmatrix} \tag{4.76}$$

Onde, dada a função genérica w, sua derivada em relação a x é representada por $w' = \frac{\partial w}{\partial x}$.

Admitindo a hipótese dos pequenos deslocamentos e deformações, o tensor das deformações E sob linearidade geométrica é definido por:

$$\mathbf{E} = \frac{1}{2}\left(\jmath + \jmath^T\right) \tag{4.77}$$

Substituindo (4.76) em (4.77) resulta em:

$$\mathbf{E} = \begin{bmatrix} (u'_x + z\theta'_y - y\theta'_z) & \frac{1}{2}(u'_y - z\theta'_x - \theta_z) & \frac{1}{2}(u'_z + y\theta'_x + \theta_y) \\ \frac{1}{2}(u'_y - z\theta'_x - \theta_z) & 0 & 0 \\ \frac{1}{2}(u'_z + y\theta'_x + \theta_y) & 0 & 0 \end{bmatrix} \tag{4.78}$$

Como $\varepsilon_x = E_{xx}$, $\gamma_{xy} = 2E_{xy}$ e $\gamma_{xz} = 2E_{xz}$, ao substituir essas expressões em (4.78) resulta o vetor das deformações escrito em termos de componentes

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} u'_x + z\theta'_y - y\theta'_z \\ 0 \\ 0 \\ 0 \\ u'_z + y\theta'_x + \theta_y \\ u'_y - z\theta'_x - \theta_z \end{bmatrix} \tag{4.79}$$

onde os valores não nulos γ_{xz} e γ_{xy} são as componentes das distorções e ε_x é a deformação axial da barra.

Na Figura 4.9, a hipótese de *Timoshenko* propõe uma rotação média para a seção, de forma que, para efeitos práticos, possa continuar considerando-se a seção plana.

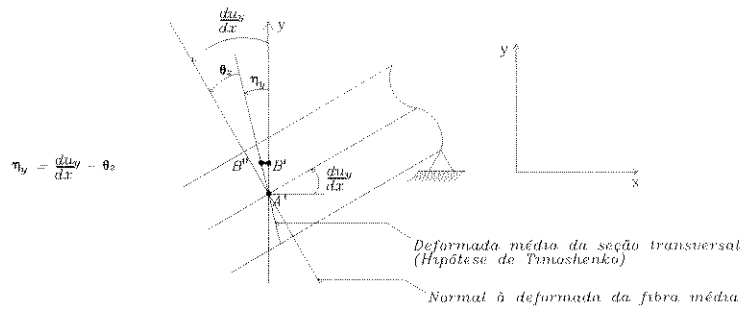

Figura 4.9: Flexão em viga de Timoshenko.

Na *Teoria de Vigas de Timoshenko* as distorções existem e são contabilizadas como ilustra a Figura 4.9.

Tensões

O tensor das tensões pode ser escrito genericamente por suas parcelas não nulas na forma:

$$\vec{T} = \sigma_x \vec{e}_x + \tau_{xy} \vec{e}_y + \tau_{xz} \vec{e}_z \tag{4.80}$$

onde σ_x é a tensão normal ao plano formado pelos vetores $\vec{e}_y$ e $\vec{e}_z$, τ_{xy} e τ_{xz} são as tensões atuantes no plano normal ao vetor $\vec{e}_x$.

As componentes das tensões são dadas por:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix} \tag{4.81}$$

Esforços solicitantes

Análogo à *Teoria Clássica de Barras*, os esforços solicitantes em um elemento de barra são representados por forças cortantes V_{xy} e V_{xz}, força normal N_x, momentos fletores M_y e M_z e momento torçor M_x, como segue:

$$\begin{aligned} N_x &= \int_A \sigma_x dA \\ V_{xy} &= \int_A \tau_{xy} dA \\ V_{xz} &= \int_A \tau_{xz} dA \\ M_x &= \int_A \left(y\tau_{xz} - z\tau_{xy}\right) dA \\ M_y &= \int_A z\sigma_x dA \\ M_z &= \int_A y\sigma_x dA \end{aligned} \tag{4.82}$$

Relação tensão-deformação

A equação constitutiva da elasticidade é formalmente dada por:

$$\sigma = D\varepsilon$$

Na seção transversal de uma viga observa-se que a distribuição de tensões normais σ_x é linear, essa distribuição pode ser considerada "exata" pela hipótese adotada na teoria de viga. Por outro lado, a tensão cisalhante aqui é considerada constante, a qual entra em contradição com a distribuição parabólica da teoria de viga. Para contornar esse problema, considera-se uma distribuição constante de tensão cisalhante, mas modificada por um coeficiente α de maneira que o trabalho de deformação da tensão tangencial constante coincida com seu valor "exato" dado pela teoria de viga.

Este coeficiente[1] conhecido como coeficiente de distorção transversal é obtido a partir de um procedimento energético, conforme Oñate[21].

[1] Em vigas de seção retangular, inércia constante e material homogêneo seu valor é $\alpha = \frac{5}{6}$.

A matriz D é a matriz constitutiva[2] do material expressa localmente por:

$$\mathbf{D} = \begin{bmatrix} EA & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha GA & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha GA & 0 & 0 & 0 \\ 0 & 0 & 0 & GJ_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & EJ_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & EJ_{zz} \end{bmatrix} \tag{4.83}$$

A expressão matricial (4.83) pode ser dividida em:

$$\begin{aligned} D_a &= [EA] \\ D_c &= \begin{bmatrix} \alpha GA & 0 \\ 0 & \alpha GA \end{bmatrix} \\ D_t &= \begin{bmatrix} GJ_0 \end{bmatrix} \\ D_f &= \begin{bmatrix} EJ_{yy} & 0 \\ 0 & EJ_{zz} \end{bmatrix} \end{aligned} \tag{4.84}$$

Esforços e deformações generalizadas

Na *Teoria de Barras de Timoshenko*, o vetor dos esforços internos ou tensões generalizadas $\hat{\sigma}$ é representado por:

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} N_x \\ V_{xy} \\ V_{xz} \\ M_x \\ M_y \\ M_z \end{bmatrix} = \int_A \begin{bmatrix} \sigma_x \\ \tau_{xy} \\ \tau_{xz} \\ (y\tau_{xz} - z\tau_{xy}) \\ z\sigma_x \\ y\sigma_x \end{bmatrix} dA \tag{4.85}$$

Para a formulação das equações constitutivas consideram-se as seguintes relações elásticas lineares:

$$\begin{aligned} \sigma_x &= E\varepsilon_x \\ \sigma_y &= E\varepsilon_y \end{aligned}$$

[2] As parcelas D_a, D_c, D_t e D_f da matriz constitutiva em (4.84) reportam a contribuição do material em relação à rigidez axial, cisalhamento, torção e flexão, respectivamente.

$$
\begin{aligned}
\sigma_z &= E\varepsilon_z \\
\tau_{yz} &= G\gamma_{yz} \\
\tau_{xz} &= \alpha G\gamma_{xz} \\
\tau_{xy} &= \alpha G\gamma_{xy}
\end{aligned} \tag{4.86}
$$

Para obter os esforços internos da estrutura substitui-se (4.86) em (4.82) e integra-se na seção transversal da barra. Assume-se que os eixos locais coincidem com os eixos principais de inércia do elemento, portanto $\int_{A_e} ydA = \int_{A_e} zdA = \int_{A_e} yzdA = 0$.

$$
\begin{aligned}
N_x &= \int_A E\varepsilon_x dA = \int_A E\left(u'_x + z\theta'_y - y\theta'_z\right) dA = EAu'_x \\
V_{xy} &= \alpha \int_A G\gamma_{xy} dA = \int_A G\left(u'_y - z\theta'_x - \theta_z\right) dA = \alpha GA\left(u'_y - \theta_z\right) \\
V_{xz} &= \alpha \int_A G\gamma_{xz} dA = \int_A G\left(u'_z + y\theta'_x + \theta_y\right) dA = \alpha GA\left(u'_z + \theta_y\right) \\
M_x &= \int_A G\left[\left(u'_z + y\theta'_x + \theta_y\right) y - \left(u'_y - z\theta'_x - \theta_z\right) z\right] dA = GJ_0\theta'_x \\
M_y &= \int_A Ez\varepsilon_x dA = \int_A Ez\left(u'_x + z\theta'_y - y\theta'_z\right) dA = EJ_{yy}\theta'_y \\
M_z &= \int_A Ey\varepsilon_x dA = \int_A Ey\left(u'_x + z\theta'_y - y\theta'_z\right) dA = -EJ_{zz}\theta'_z
\end{aligned} \tag{4.87}
$$

Com o exposto, o vetor das tensões generalizadas pode ser reescrito por

$$
\hat{\boldsymbol{\sigma}} = \begin{bmatrix} EAu'_x \\ \alpha GA\left(u'_y - \theta_z\right) \\ \alpha GA\left(u'_z + \theta_y\right) \\ GJ_0\theta'_x \\ EJ_{yy}\theta'_y \\ -EJ_{zz}\theta'_z \end{bmatrix} = D\hat{\varepsilon} \tag{4.88}
$$

onde o vetor das deformações generalizadas $\hat{\varepsilon}$ em (4.88) fica definido por:

$$\hat{\varepsilon} = \begin{bmatrix} u'_x \\ (u'_y - \theta_z) \\ (u'_z + \theta_y) \\ \theta'_x \\ \theta'_y \\ \theta'_z \end{bmatrix} \tag{4.89}$$

Ações

No presente trabalho as ações em um elemento de barra podem ser representadas por forças e momentos concentrados nos nós ou distribuídos ao longo do eixo da barra. As forças e momentos pontuais são aplicadas nos nós nas direções x, y e z globais. As forças e momentos distribuídas são aplicadas ao longo do eixo da barra nas direções x, y e z local em relação a cada lado do elemento.

A convenção de sinais é a mesma utilizada nas formulações, expressa pela regra da mão direita.

As forças de corpo f_x, f_y e f_z apresentadas na formulação a seguir para o elemento de barra expressam forças por unidade de volume.

$$\underbrace{\begin{aligned} p_x &= \textstyle\int_A f_x dA \\ p_y &= \textstyle\int_A f_y dA \\ p_z &= \textstyle\int_A f_z dA \\ m_x &= \textstyle\int_A (y f_z - z f_y) f_x dA \\ m_y &= \textstyle\int_A z f_x dA \\ m_z &= \textstyle\int_A y f_x dA \end{aligned}}_{\textit{Forças e momentos distribuídos / comprimento}} \quad \text{e} \quad \underbrace{\begin{aligned} &F_1 \text{ (força na direção x)} \\ &F_2 \text{ (força na direção y)} \\ &F_3 \text{ (força na direção z)} \\ &M_1\text{(momento em torno x)} \\ &M_2\text{(momento em torno y)} \\ &M_3\text{(momento em torno z)} \end{aligned}}_{\textit{Forças e momentos concentrados}} \tag{4.90}$$

Formulação variacional

Como apresentado na *Teoria Clássica*, partindo do *Princípio dos Trabalhos Virtuais*, o trabalho virtual dos esforços internos de uma barra é igual ao

trabalho virtual dos esforços externos e sob condições de contorno essenciais pode-se demonstrar sua equivalência com as equações diferenciais de equilíbrio. Na formulação que segue para facilitar a notação, eventualmente serão usados os índices 1, 2, 3 em correspondencia a x, y, z.

$$\int_{\Omega} \delta\varepsilon_{ij}\sigma_{ij}d\Omega - \int_{\Omega} \delta\bar{u}_i F_i d\Omega - \int_{\Gamma} \delta u_i T_i d\Gamma = 0 \tag{4.91}$$

O domínio pode ser representado pela soma dos sub-domínios e ao considerar $dA = dydz$, escreve-se:

$$\int_{\Omega} ... \, d\Omega = \sum_{e=1}^{n_{el}} \int_{\Omega_e} ... \, d\Omega_e = \sum_{e=1}^{n_{el}} \int_0^{h_e} \int_{A_e} ... \, dA dx_e \tag{4.92}$$

Aplica-se as deformações e com as considerações estabelecidas em (4.92) tem-se que:

$$\begin{aligned} 0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_0^{h_e} \int_{A_e} \left(\delta\varepsilon_x\sigma_x + \delta\gamma_{xy}\tau_{xy} + \delta\gamma_{xz}\tau_{xz} \right) dA dx_e \right. \\ & - \int_0^{h_e} \int_{A_e} \left(\delta\bar{u}_x f_x + \delta\bar{u}_y f_y + \delta\bar{u}_z f_z \right) dA dx_e \\ & \left. - \sum_{k=1}^{3} \left(\delta u_k^T F_k + \delta\theta_k^T M_k \right) \right\} \end{aligned} \tag{4.93}$$

Ao aplicar em (4.93) a cinemática de deslocamentos expressa em (4.79), tem-se que:

$$\begin{aligned} 0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_0^{h_e} \int_{A_e} \left[\left(\delta u_x' + z\delta\theta_y' - y\delta\theta_z' \right) \sigma_x + \left(\delta u_y' - z\delta\theta_x' - \delta\theta_z \right) \tau_{xy} \right. \right. \\ & \left. + \left(\delta u_z' + y\delta\theta_x' + \delta\theta_y \right) \tau_{xz} \right] dA dx_e - \int_0^{h_e} \int_{A_e} \left[\left(\delta u_x + z\delta\theta_y - y\delta\theta_z \right) f_x \right. \\ & \left. + \left(\delta u_y - z\delta\theta_x \right) f_y + \left(\delta u_z + y\delta\theta_x \right) f_z \right] dA dx_e \\ & \left. - \sum_{k=1}^{3} \left(\delta u_k^T F_k + \delta\theta_k^T M_k \right) \right\} \end{aligned} \tag{4.94}$$

Reordenando (4.94) é possível escrevê-la em função de deslocamentos virtuais associados aos correspondentes esforços apresentados em (4.82).

$$
\begin{aligned}
0 \;=\; \sum_{e=1}^{n_{el}} \Bigg\{ \int_0^{h_e} \Bigg[& \delta u_x' \underbrace{\int_{A_e} \sigma_x dA}_{N_x} + \underbrace{\left(\delta u_y' - \delta\theta_z\right)}_{\gamma_{xy}} \underbrace{\int_{A_e} \tau_{xy} dA}_{V_{xy}} \\
& + \underbrace{\left(\delta u_z' + \delta\theta_y\right)}_{\gamma_{xz}} \underbrace{\int_{A_e} \tau_{xz} dA}_{V_{xz}} + \delta\theta_x' \underbrace{\int_{A_e} \left(y\tau_{xz} - z\tau_{xy}\right) dA}_{M_x} \\
& + \delta\theta_y' \underbrace{\int_{A_e} z\sigma_x dA}_{M_y} - \delta\theta_z' \underbrace{\int_{A_e} y\sigma_x dA}_{M_z} \Bigg] dx_e - \int_0^{h_e} \Bigg[\delta u_x \underbrace{\int_{A_e} f_x \, dA}_{p_x} \\
& + \delta u_y \underbrace{\int_{A_e} f_y dA}_{p_y} + \delta u_z \underbrace{\int_{A_e} f_z dA}_{p_z} + \delta\theta_x \underbrace{\int_{A_e} \left(yf_z - zf_y\right) dA}_{m_x} \\
& + \delta\theta_y \underbrace{\int_{A_e} z f_x dA}_{m_y} - \delta\theta_z \underbrace{\int_{A_e} y f_x dA}_{m_z} \Bigg] dx_e - \sum_{k=1}^{3} \left(\delta u_k^T F_k + \delta\theta_k^T M_k\right) \Bigg\}
\end{aligned}
\tag{4.95}
$$

Ao substituir (4.87) em (4.95) tem-se:

$$
\begin{aligned}
0 \;=\; \sum_{e=1}^{n_{el}} \Bigg\{ \int_0^{h_e} & \left[\delta u_x' EA u_x' + \left(\delta u_y' - \delta\theta_z\right) \alpha GA \left(u_y' - \theta_z\right)\right. \\
& + \left(\delta u_z' + \delta\theta_y\right) \alpha GA \left(u_z' + \theta_y\right) + \delta\theta_x' GJ_0\theta_x' \\
& \left. + \delta\theta_y' EJ_{yy}\theta_y' + \delta\theta_z' EJ_{zz}\theta_z'\right] dx_e - \int_0^{h_e} \left[\delta u_x p_x + \delta u_y p_y + \delta u_z p_z\right. \\
& \left. + \delta\theta_x m_x + \delta\theta_y m_y - \delta\theta_z m_z\right] dx_e - \sum_{k=1}^{3} \left(\delta u_k^T F_k + \delta\theta_k^T M_k\right) \Bigg\}
\end{aligned}
\tag{4.96}
$$

O sistema é simétrico, bilinear e pode ser escrito na forma.

$$a(\bar{u}, u) = (\bar{u}, f) + (\bar{u}, t)_\Gamma \tag{4.97}$$

Onde:

$$\begin{aligned}
a(\bar{u}, u) &= \sum_{e=1}^{n_{el}} \left\{ \int_0^{h_e} \left[\delta u'_x E A u'_x + \delta\gamma_{xy} \alpha G A \gamma_{xy} + \delta\gamma_{xz} \alpha G A \gamma_{xz} \right.\right. \\
&\quad \left.\left. + \delta\theta'_x G J_0 \theta'_x + \delta\theta'_y E J_{yy} \theta'_y + \delta\theta'_z E J_{zz} \theta'_z \right] dx_e \right\} \\
(\bar{u}, f) &= \sum_{e=1}^{n_{el}} \int_0^{h_e} \left[\delta u_x p_x + \delta u_y p_y + \delta u_z p_z + \delta\theta_x m_x + \delta\theta_y m_y - \delta\theta_z m_z \right] dx_e \\
(\bar{u}, t)_\Gamma &= \sum_{e=1}^{n_{el}} \sum_{k=1}^{3} \left(\delta u_k^T F_k + \delta\theta_k^T M_k \right)
\end{aligned} \tag{4.98}$$

Discretização utilizando o método dos elementos finitos

Análogo à equação de elementos finitos pela *Teoria Clássica*, os deslocamentos u necessitam ser escritos em termos de deslocamentos nodais u_p. Pela definição de uma base de funções de interpolação tem-se que

$$u = N u_p \tag{4.99}$$

onde N é a matriz de funções de interpolação, utilizada para encontrar a solução aproximada das grandezas físicas do problema.

A matriz dos coeficientes de interpolação N_i que aparece nas expressões abaixo, tem seu índice i dado pela ordem do polinômio de interpolação utilizado na aproximação.

O vetor das deformações generalizadas $\hat{\varepsilon}$ em (4.89), pode ser reescrito como segue:

$$\hat{\boldsymbol{\varepsilon}} = \begin{bmatrix} u'_x \\ u'_y - \theta_z \\ u'_z + \theta_y \\ \theta'_x \\ \theta'_y \\ \theta'_z \end{bmatrix} = \sum_{i=1}^{n_{el}} \begin{bmatrix} N'_i u_{x_i} \\ N'_i u_{y_i} - N_i \theta_{z_i} \\ N'_i u_{z_i} + N_i \theta_{y_i} \\ N'_i \theta_{x_i} \\ N'_i \theta_{y_i} \\ N'_i \theta_{z_i} \end{bmatrix} = \sum_{i=1}^{n_{el}} B_i \begin{bmatrix} u_{x_i} \\ u_{y_i} \\ u_{z_i} \\ \theta_{x_i} \\ \theta_{y_i} \\ \theta_{z_i} \end{bmatrix} = \sum_{i=1}^{n_{el}} B_i d_i \tag{4.100}$$

Para implementar computacionalmente faz-se necessário escrever a formulação variacional em função da matriz B, que, segundo Oñate[21], representa o operador diferencial aplicado às funções de interpolação da matriz de deformação generalizada.

A partir (4.100) define-se a matriz de operadores[3] B nada por:

$$B_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial x} & 0 & 0 & 0 & -N_i \\ 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & N_i & 0 \\ 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial N_i}{\partial x} \end{bmatrix} \tag{4.101}$$

Onde esta pode ser dividida em:

$$\begin{aligned} B_a &= \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ B_c &= \begin{bmatrix} 0 & \frac{\partial N_i}{\partial x} & 0 & 0 & 0 & -N_i \\ 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & N_i & 0 \end{bmatrix} \\ B_t &= \begin{bmatrix} 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & 0 \end{bmatrix} \\ B_f &= \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial N_i}{\partial x} \end{bmatrix} \end{aligned} \tag{4.102}$$

[3] As submatrizes B_a, B_c, B_t e B_f em (4.102) representam os operadores diferenciais aplicados às funções de forma com relação à rigidez axial, ao cisalhamento, a torção e à flexão, respectivamente.

Conforme Hughes[15], a matriz de rigidez do elemento pode ser calculada atráves da soma das parcelas de rigidez axial, cisalhante, torção e flexão. Com isso a matriz de rigidez e vetor de cargas podem ser escritos na forma:

$$
\begin{aligned}
K^e &= k_a^e + k_c^e + k_t^e + k_f^e \\
k_a^e &= \int_0^{h_e} B_a^T D_a B_a dx_e \qquad \textit{(rigidez axial)} \\
k_c^e &= \int_0^{h_e} B_c^T D_c B_c dx_e \qquad \textit{(rigidez a cisalhamento)} \\
k_t^e &= \int_0^{h_e} B_t^T D_t B_t dx_e \qquad \textit{(rigidez a torção)} \\
k_f^e &= \int_0^{h_e} B_f^T D_f B_f dx_e \qquad \textit{(rigidez a flexão)} \\
f^e &= \begin{cases} \int_0^{h_e} N_a p_i dx_e & p = 6a - 6 + i \\ (-1)^{i+1} \int_0^{h_e} N_a m_i dx_e & p = 6a - 3 + i \end{cases}
\end{aligned}
\tag{4.103}
$$

Substituindo-se (4.103) em (4.96) tem-se a formulação variacional do problema dada em função das matrizes B e D, com isso:

$$
\begin{aligned}
0 = \sum_{e=1}^{n_{el}} \Bigg\{ & \int_0^{h_e} \left[B_a^T D_a B_a + B_c^T D_c B_c + B_t^T D_t B_t + B_f^T D_f B_f \right] dx_e \\
& - \int_0^{h_e} \left[N_i p_x + N_i p_y + N_i p_z + N_i m_x + N_i m_y - N_i m_z \right] dx_e \\
& - \sum_{k=1}^{3} \left(N_k^T F_k + N_k^T M_k \right) \Bigg\}
\end{aligned}
\tag{4.104}
$$

Análogo ao que foi apresentado para a *Teoria de Euler-Bernoulli*, no mapeamento para o sistema de coordenadas padrão definido por ξ, o diferencial de comprimento do elemento dx_e é:

$$
dx_e = J^e d\xi \tag{4.105}
$$

O jacobiano de transformação é expresso por $J^e = \frac{dx_e}{d\xi}$.

Observa-se que o jacobiano unidimensional para coordenadas cartesianas utilizado no presente trabalho é apresentado e detalhado no final deste capítulo.

Em (4.104), as parcelas da matriz de rigidez e vetor de cargas podem ser reescritas por:

$$K_i^e = \int_\xi \left(B_a^T D_a B_a + B_c^T D_c B_c + B_t^T D_t B_t + B_f^T D_f B_f\right) J^e d\xi \qquad (4.106)$$

$$f_i^e = \int_\xi \left[N_i p_x + N_i p_y + N_i p_z + N_i m_x + N_i m_y - N_i m_z\right] J^e d\xi \qquad (4.107)$$

Avaliando-se numericamente (4.106) e (4.107), utilizando a regra de quadratura simétrica de Gauss para elementos de barra, tem-se para cada elemento finito a matriz de rigidez e vetor de cargas dados por:

$$K_i^e \approx \sum_{k=1}^{n} \left[\left(B_a^T D_a B_a + B_c^T D_c B_c + B_t^T D_t B_t + B_f^T D_f B_f\right) J^e\right]_{\xi=\xi_k} W_k \qquad (4.108)$$

$$f_i^e \approx \sum_{k=1}^{n} \left[\left(N_i p_x + N_i p_y + N_i p_z + N_i m_x + N_i m_y - N_i m_z\right) J^e\right]_{\xi=\xi_k} W_k \qquad (4.109)$$

Onde W_k representa o peso do k-ésimo ponto de integração e n o número total de pontos de integração em função da ordem do polinômio do elemento.

Observações

1. as condições cinemáticas dessa formulação não incluem empenamento (seções planas permanecem planas). À medida que a relação comprimento/largura e comprimento/altura diminui, esta formulação de barras não é bem aplicada;

2. o elemento deve ser aplicável sem restrições a qualquer relação entre altura e comprimento da barra;

3. o valor da rigidez à torção é superestimado. Isto devido ao fato da presente formulação não prever o empenamento.

4.4.3 Extensão da formulação de Timoshenko para vigas curvas

Supõe-se que uma viga curva é caracterizada pelo parâmetro ξ. Assim as coordenadas do eixo são $(x(\xi), y(\xi), z(\xi))$. Baseado nisto, um vetor unitário na direção do eixo da viga é dado por:

$$\vec{v}_1 = \frac{\left(\frac{\partial x(\xi)}{\partial \xi}, \frac{\partial y(\xi)}{\partial \xi}, \frac{\partial z(\xi)}{\partial \xi}\right)}{\sqrt{\left(\frac{\partial x(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial y(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial z(\xi)}{\partial \xi}\right)^2}} \tag{4.110}$$

Conforme Devloo[12], associado ao vetor axial $\vec{v}_1(\xi)$ como ilustra a Figura 4.10, definem-se os eixos $\vec{v}_2$ e $\vec{v}_3$ que caracterizam os eixos locais principais do elemento de barra.

Com isto, qualquer ponto da seção é localizado pelas coordenadas:

$$\begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix}^T \begin{bmatrix} \vec{u}_x \\ \vec{u}_y \\ \vec{u}_z \end{bmatrix} \quad \text{e} \quad \begin{bmatrix} \vec{\theta}_1 \\ \vec{\theta}_2 \\ \vec{\theta}_3 \end{bmatrix} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix}^T \begin{bmatrix} \vec{\theta}_x \\ \vec{\theta}_y \\ \vec{\theta}_z \end{bmatrix} \tag{4.111}$$

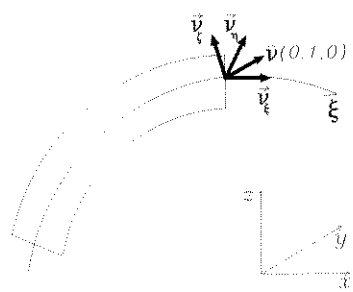

Figura 4.10: Sistema de eixos associado ao ponto de integração para o elemento de barra, onde ($\vec{v}_\xi = \vec{v}_1$, $\vec{v}_\eta = \vec{v}_2$, $\vec{v}_\zeta = \vec{v}_3$).

Assim, as expressões das formulações apresentadas para elementos de barra podem ser reescritas por suas coordenadas locais:

$$\varepsilon = \left[\begin{array}{cccccc} \varepsilon_1 & \varepsilon_2 & \varepsilon_3 & \gamma_{23} & \gamma_{13} & \gamma_{12} \end{array} \right]^T \tag{4.112}$$

Considerações geométricas

A formulação variacional que utiliza a *Teoria de Timoshenko*, objeto alvo para o tratamento de elementos de barra apresentados no presente trabalho, procura atender às seguintes condições:

1. a geometria do elemento deve permitir que se discretize com suficiente precisão uma curva, tão complexa quanto se deseje;

2. com vistas à aplicação ao cálculo de elementos espaciais de barras, deve-se fazer uma escolha de funções paramétricas de deslocamentos generalizados tais que sejam satisfeitas automaticamente as condições de continuidade nas interfaces;

3. o elemento deve ser aplicável sem restrições a qualquer relação entre altura da seção e comprimento da barra. Deve-se, por consequência, levar em conta as deformações devidas tanto aos esforços normais quanto aos esforços cortantes. São admitidas as hipóteses clássicas da teoria de vigas no regime elástico linear para pequenos deslocamentos e pequenas deformações, substituindo-se as hipóteses das seções planas e normais ao eixo do elemento pelas hipóteses de Timoshenko.

O elemento é definido por correspondência entre as coordenadas globais x, y, z e a coordenada curvilínea ξ. A coordenada ξ do elemento mestre do programa varia de -1 a $+1$. Na seção transversal, para um dado ponto, define-se o vetor $\vec{v}_1(\xi)$ na direção axial do elemento, conforme (4.110) e os vetores $\vec{v}_2$ e $\vec{v}_3$ nas direções dos eixos principais de inércia.

No programa utilizado para a aproximação de elementos finitos existe uma forte separação entre o mapeamento geométrico e as funções de interpolação das variáveis do problema. O usuário pode definir funções distintas para serem utilizadas para a interpolação da geometria e para a aproximação computacional do cálculo.

Para a validação da aproximação dos elementos de barras retas e curvas no espaço foram utilizadas as seguintes aproximações geométricas:

- elemento reto, numa posição qualquer do espaço é discretizado linearmente, onde um ponto genérico é descrito pelas coordenadas cartesianas x, y e z;
- elemento de eixo curvo, discretizado de forma quadrática, onde um ponto genérico é descrito pelas coordenadas cartesianas x, y e z;
- elemento de eixo curvo, numa posição qualquer do espaço, onde um ponto genérico é descrito através de coordenadas cilíndricas r, θ e z.

É necessário conhecer os jacobianos unidimensionais para as transformações entre os elementos deformados descritos em coordenadas cartesianas ou coordenadas cilíndricas e o elemento mestre descrito em coordenada adimensional ξ.

Jacobiano unidimensional para coordenadas cartesianas

Segundo Becker[7], o jacobiano para coordenadas cartesianas unidimensionais é definido por:

$$\begin{aligned} x &= \sum_i x_i N_i(\xi) & \qquad \frac{\partial x}{\partial \xi} &= \sum_i x_i N_i'(\xi) \\ y &= \sum_i y_i N_i(\xi) & \text{e} \qquad \frac{\partial y}{\partial \xi} &= \sum_i y_i N_i'(\xi) \\ z &= \sum_i z_i N_i(\xi) & \qquad \frac{\partial z}{\partial \xi} &= \sum_i z_i N_i'(\xi) \end{aligned} \tag{4.113}$$

Assim, pode ser reescrito por:

$$J^e = \sqrt{\left(\frac{\partial x(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial y(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial z(\xi)}{\partial \xi}\right)^2} \tag{4.114}$$

Com o vetor $\vec{v}_1(\xi)$ descrito em (4.110) e os vetores $\vec{v}_2$ e $\vec{v}_3$, pode-se calcular diretamente as contribuições dos coeficientes de rigidez locais na matriz de rigidez global sem a necessidade de mudança do sistema de coordenadas. O vetor $\vec{v}_1(\xi)$ pode ser reescrito na forma:

$$\vec{v}_1 = \frac{1}{J^e}\left(\frac{\partial x(\xi)}{\partial \xi}, \frac{\partial y(\xi)}{\partial \xi}, \frac{\partial z(\xi)}{\partial \xi}\right) \tag{4.115}$$

Os vetores $\vec{v}_2$ e $\vec{v}_3$ são expressos por:

$$\vec{v}_2 = \frac{\tilde{v} - (\tilde{v}.\vec{v}_1)\,\vec{v}_1}{\|\tilde{v} - (\tilde{v}.\vec{v}_1)\,\vec{v}_1\|} \qquad \text{e} \qquad \vec{v}_3 = \vec{v}_1 \times \vec{v}_2 \tag{4.116}$$

Onde $\tilde{v}$ é um vetor auxiliar de referência, obtido externamente ao programa, não necessariamente perpendicular a $\vec{v}_1$.

No presente trabalho foi implementado apenas o caso particular do vetor $\tilde{v}$ paralelo ao eixo global y.

O vetor $\vec{v}_2$ é originado da combinação linear entre os vetores $\vec{v}_1$ e $\tilde{v}$, após a imposição de duas condições:

1. produto escalar entre $\vec{v}_1$ e $\tilde{v}$ é igual a zero ($\vec{v}_1.\tilde{v} = 0$);
2. $\|\vec{v}_2\| = 1$.

Assim, torna-se possível calcular em cada ponto de integração os deslocamentos e rotações em função da orientação dos eixos no ponto.

Jacobiano unidimensional para coordenadas cilíndricas

Para realizar o cálculo das deformações e a análise de esforços em elementos curvos no espaço com maior precisão, foi desenvolvido um elemento finito unidimensional, descrito em coordenadas cilíndricas, que pudesse mais precisamente descrever a geometria do problema. O jacobiano da transformação de coordenadas cilíndricas para coordenadas unidimensionais ξ para o elemento mestre pode ser obtido como segue:

$$\begin{aligned} r(\xi) &= \sum_i r_i N_i(\xi) & & \frac{\partial r}{\partial \xi} = \sum_i r_i N_i'(\xi) \\ \theta(\xi) &= \sum_i \theta_i N_i(\xi) & \text{e} \quad & \frac{\partial \theta}{\partial \xi} = \sum_i \theta_i N_i'(\xi) \\ z(\xi) &= \sum_i z_i N_i(\xi) & & \frac{\partial z}{\partial \xi} = \sum_i z_i N_i'(\xi) \end{aligned} \tag{4.117}$$

Então

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \\ \frac{\partial z}{\partial \xi} \end{bmatrix} = \begin{bmatrix} \cos\theta & -r\,sen\theta & 0 \\ sen\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial r}{\partial \xi} \\ \frac{\partial \theta}{\partial \xi} \\ \frac{\partial z}{\partial \xi} \end{bmatrix} \tag{4.118}$$

onde r, θ e z representam as coordenadas em relação ao sistema de coordenadas cilíndrico, tomado em cada ponto de integração.

Capítulo 5

Elemento de placa

5.1 Introdução à teoria de placas

Define-se placa como um caso particular de um sólido tridimensional no qual uma das dimensões é muito menor do que as outras duas e o carrregamento é aplicado na direção da menor dimensão, gerando efeitos de flexão e de cisalhamento.

Estruturas do tipo placa encontram uma grande variedade de aplicaç ões na prática da engenharia tais como: lajes de edifícios, cascos de navios, estruturas aeroespaciais, muros de contenção, caixas d'água, reatores nucleares, fundações.

Neste capítulo são apresentadas teorias para placas formuladas a partir de duas hipóteses cinemáticas sob linearidade geométrica. São estudadas as formulações que decorrem dessas duas diferentes hipóteses cinemáticas: *Teoria de Kirchoff* e *Teoria de Reissner-Mindlin.*

No decorrer deste capítulo algumas definições são apresentadas de forma repetitiva. Essa estratégia foi adotada para que as duas teorias para placas apresentadas possam ser estudadas independentemente, sem a necessidade de se recorrer a definições encontradas em capítulos anteriores.

Na *Teoria de Kirchoff* para placas finas, as deformações transversais são nulas devido à não imposição da rotação das normais à superfície.

Um novo modelo foi proposto por *Reissner*[26] em 1945 e por *Mindlin*[18] em 1952, no qual consideram a influência da deformação transversal, possibilitando a utilização da teoria para diferentes espessuras de placas.

Na *Teoria de Kirchoff* existe a dificuldade grande de se encontrarem

funções de forma que satisfaçam os requisitos de continuidade de flechas e rotações nos elementos. A *Teoria de Reissner-Mindlin* faz menos restritiva a hipótese de ortogonalidade da normal à superfície deformada, o que introduz o efeito de deformação por cisalhamento transversal, permitindo assim a análise de placas espessas.

Os elementos de placa baseados na *Teoria de Reissner-Mindlin* aceitam funções de forma de classe C^0 o que elimina efeitos de não-conformidade que aparecem em elementos de *Kirchoff*. Em contrapartida, o preço que se paga pela utilização de elementos de *Reissner-Mindlin* é que podem aparecer dificuldades de ordem numérica quando de sua utilização em placas muito finas, obtendo com isso soluções muito rígidas devidas à influência excessiva dos termos de força cortante, o que é conhecido na literatura como efeito de bloqueio ou *shear-locking*.

O problema pode ser eliminado com técnicas de integração reduzida ou através de uma modificação da energia de deformação cisalhante, de modo a eliminar termos espúrios do funcional. Um outro artifício para contornar o problema é a utilização de funções de interpolação de ordem maior ou igual a 4, sem qualquer alteração do funcional da energia de deformação. Neste trabalho optou-se por este caminho.

Maiores detalhes das teorias de placas e de elementos finitos desenvolvidos para placas podem ser encontrados em Hinton & Huang[14], Bathe & Dvorkin[6], Zienkiewicz & Taylor[39] e Oñate, Zienkiewicz, Suarez & Taylor[20].

Considere-se o domínio da placa denotado da seguinte maneira (Hughes[15]).

$$\Omega = \left\{ (x,y,z) \in R^3 |\ z \in \left[-\frac{t}{2}, +\frac{t}{2} \right], (x,y) \in S \subset R^2 \right\} \qquad (5.1)$$

Onde:

S: projeção do domínio Ω sobre a superfície de referência;

t: espessura da placa.

Geometricamente, placas são iguais aos planos elásticos, contudo as placas estão sujeitas somente ao carregamento transversal (carga perpendicular ao plano da placa), não havendo cargas atuando no plano da placa.

Na essência, as distintas teorias de placas diferenciam-se nas hipóteses sobre a rotação das normais em relação ao plano médio. Dessa forma, a *Teoria de Kirchoff* também conhecida por *Teoria Clássica de Placas Delgadas* estabelece que as normais se mantêm retas e ortogonais à superfície média e os elementos necessitam de continuidade C^1, por existir derivada segunda da

flecha na expressão dos trabalhos virtuais. Por outro lado, teorias como a de *Reissner-Mindlin*, não exigem ortogonalidade com a deformada da superfície média da placa após a deformação.

5.1.1 Convenção de sinais

Adota-se para convenção de sinais das rotações a regra da mão-direita, aplicada sobre os eixos cartesianos positivos. Os deslocamentos no elemento de placa são representados nesse capítulo por três translações u_x, u_y e u_z nas direções dos eixos cartesianos x, y e z e três rotações θ_x, θ_y e θ_z em torno dos eixos x, y e z; portanto admitindo seis graus de liberdade por nó. É importante salientar que essa convenção de sinais é necessária para as aspirações de se estender a teoria para aplicações no espaço tridimensional. Entretanto, como ilustra a Figura 5.1, essa adoção de rotações acompanhando os eixos cartesianos traz uma pequena complicação pelo fato de que a rotação positiva em torno do eixo cartesiano x produz tração nas fibras superiores da placa e rotação positiva em torno do eixo y produz tração positiva nas fibras inferiores da placa.

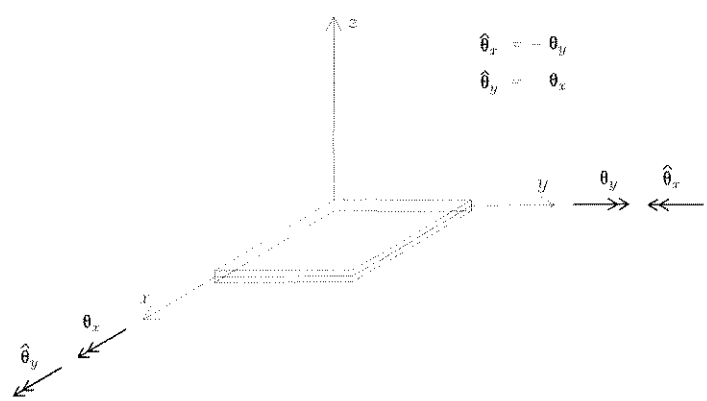

Figura 5.1: Convenção de sinais adotada para as rotações no elemento de placa.

5.1.2 Teoria de Kirchoff para placas

À superfície plana equidistante das superfícies superior e inferior da placa denomina-se plano médio ou ainda superfície média. Por outro lado, define-

se "estado de placa" ao estado em que somente atuam as ações normais ao plano médio da placa e momentos que estão contidos nesse plano.

Hipótese cinemática

Para simplificar o problema, *Kirchoff* criou em 1850 uma teoria estrutural levando esta para um modelo bidimensional baseado nas seguintes hipóteses:

1. Em pontos da superfície média $u_x = u_y = 0$;
2. As deflexões da placa são pequenas, comparada com sua espessura;
3. A tensão normal transversal é desprezível ($\sigma_z = 0$);
4. As normais à superfície de referência indeformada da placa permanecem normais à superfície de referência deformada e não sofrem variação de comprimento, dessa forma as deformações cisalhantes transversais são nulas.

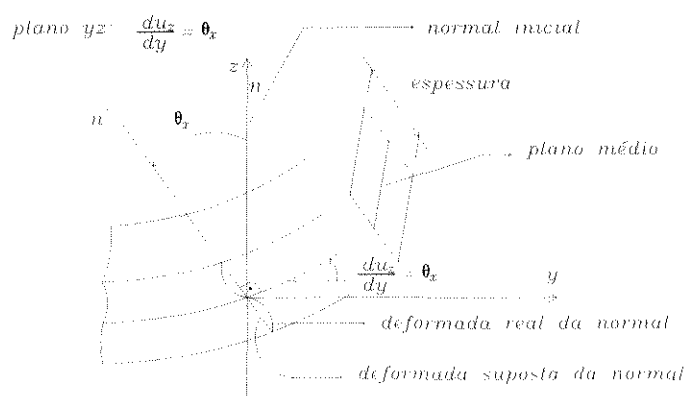

Figura 5.2: Deformação do plano médio de uma placa delgada e rotação da normal.

A Figura 5.2 apresenta o deslocamento de um ponto pertencente ao plano médio da placa; este é dado pela cinemática de deslocamentos do ponto representada por:

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} = \begin{bmatrix} -z\hat{\theta}_x \\ -z\hat{\theta}_y \\ u_z \end{bmatrix} = \begin{bmatrix} z\theta_y \\ -z\theta_x \\ u_z \end{bmatrix} \tag{5.2}$$

Na *Teoria de Kirchoff* considera-se que os giros do plano médio em um ponto coincidem com a derivada do plano médio nesse ponto, com isso e adotando o sistema de convenção para os deslocamentos e rotações ilustrados na Figura 5.1, tem-se que:

$$\frac{\partial u_z}{\partial x} = \hat{\theta}_x = -\theta_y \qquad \text{e} \qquad \frac{\partial u_z}{\partial y} = \hat{\theta}_y = \theta_x \tag{5.3}$$

Ao substituir (5.3) em (5.2) obtém-se:

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} = \begin{bmatrix} -z\frac{\partial u_z}{\partial x} \\ -z\frac{\partial u_z}{\partial y} \\ u_z \end{bmatrix} \tag{5.4}$$

Deformações

A matriz jacobiana de deslocamentos, partindo de (5.4), é dada pela seguinte expressão matricial:

$$J = \frac{\partial\left(\bar{u}_x, \bar{u}_y, \bar{u}_z\right)}{\partial\left(x, y, z\right)} = \begin{bmatrix} -z\frac{\partial^2 u_z}{\partial x^2} & -z\frac{\partial^2 u_z}{\partial x \partial y} & -\frac{\partial u_z}{\partial x} \\ -z\frac{\partial^2 u_z}{\partial x \partial y} & -z\frac{\partial^2 u_z}{\partial y^2} & -\frac{\partial u_z}{\partial y} \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & \frac{\partial u_z}{\partial z} \end{bmatrix} \tag{5.5}$$

Pela teoria dos pequenos deslocamentos e deformações, o tensor das deformações E sob linearidade geométrica é definido por:

$$\mathbf{E} = \frac{1}{2}\left(J + J^T\right) \tag{5.6}$$

Substituindo (5.5) em (5.6) obtém-se:

$$\mathbf{E} = \begin{bmatrix} -z\frac{\partial^2 u_z}{\partial x^2} & -z\frac{\partial^2 u_z}{\partial x \partial y} & 0 \\ -z\frac{\partial^2 u_z}{\partial x \partial y} & -z\frac{\partial^2 u_z}{\partial y^2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.7}$$

As componentes não nulas do tensor das deformações E podem ser escritas num vetor ε de dimensão 6×1 por:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} E_{xx} \\ E_{yy} \\ E_{zz} \\ 2E_{yz} \\ 2E_{xz} \\ 2E_{xy} \end{bmatrix} \tag{5.8}$$

Reescrevendo este vetor em função das derivadas parciais dos deslocamentos tem-se:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} -z\frac{\partial^2 u_z}{\partial x^2} \\ -z\frac{\partial^2 u_z}{\partial y^2} \\ 0 \\ 0 \\ 0 \\ -2z\frac{\partial^2 u_z}{\partial x \partial y} \end{bmatrix} \tag{5.9}$$

Onde γ_{yz}, γ_{xz} e γ_{xy} são as componentes de distorções e ε_x e ε_y são as deformações nas direções x e y existentes na placa.

Observa-se que para a *Teoria de Clássica de Placas Delgadas*, as componentes de distorção γ_{yz} e γ_{xz} são nulas.

Tensões

Considera-se na presente teoria de placas o estado plano de tensões. O tensor das tensões pode ser escrito genericamente na forma:

$$\vec{T} = \sigma_x \vec{e}_x + \sigma_y \vec{e}_y + \tau_{xy} \vec{e}_y \tag{5.10}$$

Onde τ_{xy} é a tensão atuante no plano normal ao vetor $\vec{e}_x$ e as tensões σ_x e σ_y atuam normais ao plano formado pelos vetores $\vec{e}_y \vec{e}_z$ e $\vec{e}_x \vec{e}_z$, respectivamente. Observa-se que para essa teoria a tensão $\sigma_z = 0$ (despresível). As tensões podem ser escritas de forma vetorial por:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix} \tag{5.11}$$

Esforços solicitantes

Os esforços solicitantes em um elemento de placa são forças cortantes V_{xz} e V_{yz}, momentos fletores M_x, M_y e momento volvente M_{xy}, definidos por:

$$\begin{aligned} V_{xz} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{xz} dz \\ V_{yz} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{yz} dz \\ M_x &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_x dz \\ M_y &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_y dz \\ M_{xy} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\tau_{xy} dz \end{aligned} \tag{5.12}$$

Relação tensão-deformação

Para a formulação das equações constitutivas considerou-se que o material é elástico, linear e isotrópico. A equação constitutiva da elasticidade é formalmente representada por

$$\sigma = D\varepsilon \tag{5.13}$$

e a matriz constitutiva do material é definida por:

$$\mathbf{D} = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 & 0 & 0 \\ \nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{5.14}$$

Esforços e deformações generalizadas

Na *Teoria Clássica de Placas*, o vetor dos esforços internos ou tensões generalizadas $\hat{\sigma}$ é definido por:

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} V_{xz} \\ V_{yz} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \begin{bmatrix} \tau_{xz} \\ \tau_{yz} \\ z\sigma_x \\ z\sigma_y \\ z\tau_{xy} \end{bmatrix} dz \tag{5.15}$$

Pelas equações constitutivas a relação tensão-deformação pode ser escrita por:

$$\begin{aligned} \sigma_x &= \frac{E}{(1-\nu^2)} (\varepsilon_x + \nu\varepsilon_y) \\ \sigma_y &= \frac{E}{(1-\nu^2)} (\nu\varepsilon_x + \varepsilon_y) \\ \sigma_z &= 0 \\ \tau_{yz} &= 0 \\ \tau_{xz} &= 0 \\ \tau_{xy} &= G\gamma_{xy} \end{aligned} \tag{5.16}$$

Substitui-se (5.16) em (5.12) e, observando que as forças cortantes não produzem trabalho, obtêm-se os esforços solicitantes ou tensões generalizadas $\hat{\sigma}$ em um elemento de placa:

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} V_{xz} \\ V_{yz} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \begin{bmatrix} 0 \\ 0 \\ z\left[\frac{E}{(1-\nu^2)}\left(\varepsilon_x + \nu\varepsilon_y\right)\right] \\ z\left[\frac{E}{(1-\nu^2)}\left(\nu\varepsilon_x + \varepsilon_y\right)\right] \\ z\left(G\gamma_{xy}\right) \end{bmatrix} dz \tag{5.17}$$

Estes esforços podem serem escritos na forma:

$$\begin{aligned} M_x &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\frac{E}{(1-\nu^2)}\left(\varepsilon_x + \nu\varepsilon_y\right) dz = -\frac{Et^3}{12\left(1-\nu^2\right)}\left[\left(\frac{\partial^2 u_z}{\partial x^2}\right) + \nu\left(\frac{\partial^2 u_z}{\partial y^2}\right)\right] \\ M_y &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\frac{E}{(1-\nu^2)}\left(\nu\varepsilon_x + \varepsilon_y\right) dz = -\frac{Et^3}{12\left(1-\nu^2\right)}\left[\nu\left(\frac{\partial^2 u_z}{\partial x^2}\right) + \left(\frac{\partial^2 u_z}{\partial y^2}\right)\right] \\ M_{xy} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} zG\gamma_{xy}dz = -G\frac{t^3}{12}\left(2\frac{\partial^2 u_z}{\partial x \partial y}\right) \end{aligned} \tag{5.18}$$

O vetor das tensões generalizadas fica expresso por

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} 0 \\ 0 \\ -\frac{Et^3}{12(1-\nu^2)}\left[\left(\frac{\partial^2 u_z}{\partial x^2}\right) + \nu\left(\frac{\partial^2 u_z}{\partial y^2}\right)\right] \\ -\frac{Et^3}{12(1-\nu^2)}\left[\nu\left(\frac{\partial^2 u_z}{\partial x^2}\right) + \left(\frac{\partial^2 u_z}{\partial y^2}\right)\right] \\ -G\frac{t^3}{12}\left(2\frac{\partial^2 u_z}{\partial x \partial y}\right) \end{bmatrix} = \hat{D}\hat{\varepsilon} \tag{5.19}$$

onde $\hat{D} = \frac{t^3}{12}D$. O vetor das deformações generalizadas na placa $\hat{\varepsilon}$, em (5.19) é definido por:

$$\hat{\varepsilon} = \begin{bmatrix} 0 \\ 0 \\ -\left(\frac{\partial^2 u_z}{\partial x^2}\right) \\ -\left(\frac{\partial^2 u_z}{\partial y^2}\right) \\ -\left(2\frac{\partial^2 u_z}{\partial x \partial y}\right) \end{bmatrix} \tag{5.20}$$

Ações

No presente trabalho as ações em um elemento de placa podem ser representadas por forças e momentos pontuais e/ou distribuídos.

As forças pontuais são aplicadas na direção z global. Os momentos pontuais são aplicados nas direções x e y global. As forças distribuídas são aplicadas na superfície da placa e na direção do eixo local z, e os momentos distribuídos são aplicados na superfície da placa na direção dos eixos x e y local.

$$\underbrace{\begin{aligned} p_x &= 0 \\ p_y &= 0 \\ p_z &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} f_z dz \\ m_x &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_x dz \\ m_y &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_y dz \\ m_z &= 0 \end{aligned}}_{\textit{Forças e momentos distribuídos / área}} \quad \text{e} \quad \underbrace{\begin{aligned} F_1 &= 0 \quad \text{(força na direção x)} \\ F_2 &= 0 \quad \text{(força na direção y)} \\ F_3 &\quad \text{(força na direção z)} \\ M_1 &\quad \text{(momento em torno x)} \\ M_2 &\quad \text{(momento em torno y)} \\ M_3 &= 0 \quad \text{(momento em torno z)} \end{aligned}}_{\textit{Forças e momentos concentrados}} \tag{5.21}$$

A convenção de sinais é a mesma utilizada nas formulações, expressa pela regra da mão direita.

As forças de corpo f_x, f_y e f_z, apresentadas na formulação, exprimem forças por unidade de volume.

Formulação variacional

O trabalho virtual dos esforços internos de uma barra é igual ao trabalho virtual das ações e sob condições de contorno essenciais, pode-se demonstrar sua equivalência com as equações diferenciais que expressam o equilíbrio. Na formulação que segue, para facilitar a notação, eventualmente serão usados os índices 1, 2 e 3 em correspondencia a x, y e z. Adota-se também que $u_{3,x} = \frac{\partial u_3}{\partial x}$ e $u_{3,y} = \frac{\partial u_3}{\partial y}$.

A partir do *Princípio dos Trabalhos Virtuais* tem-se:

$$\int_{\Omega} \delta\varepsilon_{ij}\sigma_{ij}d\Omega - \int_{\Omega} \delta\bar{u}_i F_i d\Omega - \int_{\Gamma} \delta u_i T_i d\Gamma = 0 \tag{5.22}$$

O domínio pode ser representado pela soma dos sub-domínios e ao considerar $dA = dxdy$ pode-se escrever:

$$\int_{\Omega} \ldots d\Omega = \sum_{e=1}^{n_{el}} \int_{\Omega_e} \ldots d\Omega_e = \sum_{e=1}^{n_{el}} \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \ldots dzdA_e \tag{5.23}$$

Aplica-se às deformações existentes e ainda com as considerações estabelecidas em (5.23) tem-se que:

$$\begin{aligned} 0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left[\delta\varepsilon_x\sigma_x + \delta\varepsilon_y\sigma_y + \delta\gamma_{xy}\tau_{xy}\right] dzdA_e \right. \\ & - \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left(\delta\bar{u}_x f_x + \delta\bar{u}_y f_y + \delta\bar{u}_z f_z\right) dzdA_e \\ & \left. - \int_{\Gamma_e} \left(\delta u_3 F_3 + \delta u_{3,x} M_y + \delta u_{3,y} M_x\right) d\Gamma_e \right\} \end{aligned} \tag{5.24}$$

Substituindo (5.9) e (5.4) em (5.24) obtém-se:

$$\begin{aligned} 0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left[\left(-z\frac{\partial^2\delta u_z}{\partial x^2}\right)\sigma_x + \left(-z\frac{\partial^2\delta u_z}{\partial y^2}\right)\sigma_y \right. \right. \\ & \left. + \left(-2z\frac{\partial^2\delta u_z}{\partial x\partial y}\right)\tau_{xy} \right] dzdA_e - \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left[\left(-z\frac{\partial\delta u_z}{\partial x}\right) f_x \right. \\ & \left. \left. + \left(-z\frac{\partial\delta u_z}{\partial y}\right) f_y + \delta u_z f_z \right] dzdA_e \right. \end{aligned}$$

$$- \int_{\Gamma_e} \left(\delta u_3 F_3 + \delta u_{3,x} M_y + \delta u_{3,y} M_x\right) d\Gamma_e \Bigg\} \tag{5.25}$$

Reordenando (5.25) é possível escrevê-la em função de deslocamentos virtuais associados aos correspondentes esforços:

$$\begin{aligned}
0 \;=\; \sum_{e=1}^{n_{el}} \Bigg\{ \int_{A_e} \Bigg[& - \left(\frac{\partial^2 \delta u_z}{\partial x^2}\right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_x dz}_{M_x} - \left(\frac{\partial^2 \delta u_z}{\partial y^2}\right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_y dz}_{M_y} \\
& - \left(\frac{2\partial^2 \delta u_z}{\partial x \partial y}\right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z\tau_{xy} dz}_{M_{xy}} \Bigg] dA_e - \int_{A_e} \Bigg[- \left(\frac{\partial \delta u_z}{\partial x}\right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_x dz}_{m_x} \\
& - \left(\frac{\partial \delta u_z}{\partial y}\right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_y dz}_{m_y} + \delta u_z \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} f_z dz}_{p_z} \, dA_e \\
& - \int_{\Gamma_e} \left(\delta u_3 F_3 + \delta u_{3,x} M_y + \delta u_{3,y} M_x\right) d\Gamma_e \Bigg\}
\end{aligned} \tag{5.26}$$

Aplica-se (5.18) em (5.26) obtendo:

$$\begin{aligned}
0 \;=\; \sum_{e=1}^{n_{el}} \Bigg\{ \int_{A_e} \Bigg[& \left(\frac{\partial^2 \delta u_z}{\partial x^2}\right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\left(\frac{\partial^2 u_z}{\partial x^2}\right) + \nu \left(\frac{\partial^2 u_z}{\partial y^2}\right) \right) \\
& + \left(\frac{\partial^2 \delta u_z}{\partial y^2}\right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\nu \left(\frac{\partial^2 u_z}{\partial x^2}\right) + \left(\frac{\partial^2 u_z}{\partial y^2}\right) \right) \\
& + \left(2\frac{\partial^2 \delta u_z}{\partial x \partial y}\right) G \frac{t^3}{12} \left(2\frac{\partial^2 u_z}{\partial x \partial y}\right) \Bigg] dA_e - \int_{A_e} \Bigg[- \left(\frac{\partial \delta u_z}{\partial x}\right) m_x \\
& - \left(\frac{\partial \delta u_z}{\partial y}\right) m_y + \delta u_z p_z \Bigg] dA_e \\
& - \int_{\Gamma_e} \left(\delta u_3 F_3 + \delta u_{3,x} M_y + \delta u_{3,y} M_x\right) d\Gamma_e \Bigg\}
\end{aligned} \tag{5.27}$$

Reescrevendo (5.27) tem-se:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \left[\left(\frac{\partial^2 \delta u_z}{\partial x^2} \right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial x^2} \right) \right. \right. \\
& + \left(\frac{\partial^2 \delta u_z}{\partial x^2} \right) \frac{\nu Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial y^2} \right) \\
& + \left(\frac{\partial^2 \delta u_z}{\partial y^2} \right) \frac{\nu Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial x^2} \right) \\
& + \left(\frac{\partial^2 \delta u_z}{\partial y^2} \right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial y^2} \right) \\
& \left. + \left(2\frac{\partial^2 \delta u_z}{\partial x \partial y} \right) \frac{Gt^3}{12} \left(2\frac{\partial^2 u_z}{\partial x \partial y} \right) \right] dA_e \\
& - \int_{A_e} \left[- \left(\frac{\partial \delta u_z}{\partial x} \right) m_x - \left(\frac{\partial \delta u_z}{\partial y} \right) m_y + \delta u_z p_z \right] dA_e \\
& \left. - \int_{\Gamma_e} \left(\delta u_3 F_3 + \delta u_{3,x} M_y + \delta u_{3,y} M_x \right) d\Gamma_e \right\}
\end{aligned}
\tag{5.28}
$$

O sistema simétrico e bilinear é representado por:

$$
a\left(\delta u, u\right) = \left(\delta u, f\right) + \left(\delta u, T\right)_{\Gamma} \tag{5.29}
$$

Onde cada termo é dado por:

$$
\begin{aligned}
a\left(\delta u, u\right) \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \left[\left(\frac{\partial^2 \delta u_z}{\partial x^2} \right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial x^2} \right) \right. \right. \\
& + \left(\frac{\partial^2 \delta u_z}{\partial x^2} \right) \frac{\nu Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial y^2} \right) \\
& + \left(\frac{\partial^2 \delta u_z}{\partial y^2} \right) \frac{\nu Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial x^2} \right) \\
& + \left(\frac{\partial^2 \delta u_z}{\partial y^2} \right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial^2 u_z}{\partial y^2} \right) \\
& \left. \left. + \left(2\frac{\partial^2 \delta u_z}{\partial x \partial y} \right) \frac{Gt^3}{12} \left(2\frac{\partial^2 u_z}{\partial x \partial y} \right) \right] dA_e \right\}
\end{aligned}
$$

$$
\begin{aligned}
(\delta u, f) &= \sum_{e=1}^{n_{el}} \int_{A_e} \left[-\left(\frac{\partial \delta u_z}{\partial x}\right) m_x - \left(\frac{\partial \delta u_z}{\partial y}\right) m_y + \delta u_z p_z \right] dA_e \\
(\delta u, T)_\Gamma &= \sum_{e=1}^{n_{el}} \int_{\Gamma_e} \left(\delta u_3 F_3 + \delta u_{3,x} M_y + \delta u_{3,y} M_x\right) d\Gamma_e
\end{aligned} \tag{5.30}
$$

Discretização utilizando o método dos elementos finitos

Para obter a formulação de elementos finitos de placa pela *Teoria de Kirchoff*, necessita-se escrever os deslocamentos u em termos de deslocamentos nodais u_p mediante a definição de uma base de funções de interpolação como segue:

$$
u = N u_p \tag{5.31}
$$

onde N representa as funções de forma utilizadas para encontrar a solução numérica aproximada do problema.

A matriz dos coeficientes de interpolação N_i que aparece nas expressões abaixo, tem seu índice i dado pela ordem do polinômio de interpolação utilizado na aproximação.

O vetor das deformações generalizadas $\hat{\varepsilon}$ em (5.20) pode ser escrito matricialmente na forma:

$$
\hat{\varepsilon} = \begin{bmatrix} 0 \\ 0 \\ -\left(\frac{\partial^2 u_z}{\partial x^2}\right) \\ -\left(\frac{\partial^2 u_z}{\partial y^2}\right) \\ -2\left(\frac{\partial^2 u_z}{\partial x \partial y}\right) \end{bmatrix} = \sum_{i=1}^{n_{el}} \begin{bmatrix} 0 \\ 0 \\ -\left(\frac{\partial^2 N_i}{\partial x^2} u_{z_i}\right) \\ -\left(\frac{\partial^2 N_i}{\partial y^2} u_{z_i}\right) \\ -2\left(\frac{\partial^2 N_i}{\partial x \partial y} u_{z_i}\right) \end{bmatrix} = \sum_{i=1}^{n_{el}} B_i \begin{bmatrix} u_{x_i} \\ u_{y_i} \\ u_{z_i} \\ \left(\frac{\partial u_z}{\partial x}\right)_i \\ \left(\frac{\partial u_z}{\partial y}\right)_i \end{bmatrix} \tag{5.32}
$$

Para a implementação computacional é necessário escrever a matriz B que representa o operador diferencial aplicado às funções de interpolação da matriz de deformação generalizada.

Assim, por (5.32) tem-se que a matriz B é definida por:

$$B_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{\partial^2 N_i}{\partial x^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\partial^2 N_i}{\partial y^2} & 0 \\ 0 & 0 & 0 & 0 & -2\left(\frac{\partial^2 N_i}{\partial x \partial y}\right) \end{bmatrix} \tag{5.33}$$

Com isso, a matriz de rigidez e o vetor de cargas do elemento são obtidos a partir da substituição de (5.14) e (5.33) em (5.30) dando origem à expressão:

$$\begin{aligned} K^e &= \int_{A_e} \left(B^T \hat{D} B\right) dA_e \\ f^e &= \begin{cases} \int_{A_e} N_a p_z dA_e \\ -\int_{A_e} N_a m_x dA_e - \int_{\Gamma_e} N_a M_1 d\Gamma_e \\ -\int_{A_e} N_a m_y dA_e - \int_{\Gamma_e} N_a M_2 d\Gamma_e \end{cases} \end{aligned} \tag{5.34}$$

Substitui-se (5.34) em (5.28) e obtém-se a formulação variacional do problema expressa em função das matrizes B e D, como segue:

$$\begin{aligned} 0 = \sum_{e=1}^{n_{el}} \Bigg\{ & \left[\int_{A_e} \left(B^T \hat{D} B\right)\right] dA_e \\ & - \int_{A_e} \left[-N_i m_x - N_i m_y + N_i p_z\right] dA_e \\ & - \int_{\Gamma_e} \left(\delta N_i F_3 + \delta N_{i,x} M_y + \delta N_{i,y} M_x\right) d\Gamma_e \Bigg\} \end{aligned} \tag{5.35}$$

Em (5.35) as parcelas da matriz de rigidez e vetor de cargas são separadas e expressas por:

$$K^e = \int_{A_e} \left(B^T \hat{D} B\right) dA_e \tag{5.36}$$

$$f^e = \int_{A_e} \left(-N_i m_x - N_i m_y + N_i p_z\right) dA_e \tag{5.37}$$

No presente trabalho o sexto grau de liberdade é considerado colocando-se o valor unitário na diagonal da matriz de rigidez.

Ao mapear para o sistema de coordenadas padrão definido por ξ e η, o diferencial de superfície do elemento é dado por dA_e e pode ser escrito por

$$dA_e = \det J^e d\xi d\eta \tag{5.38}$$

onde J^e é a matriz jacobiana dada por:

$$J^e = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} x_i & \frac{\partial N_i^e}{\partial \xi} y_i \\ \frac{\partial N_i^e}{\partial \eta} x_i & \frac{\partial N_i^e}{\partial \eta} y_i \end{bmatrix} \tag{5.39}$$

As coordenadas nodais do elemento em relação ao sistema global de coordenadas são representadas por x_i e y_i.

A inversa da matriz jacobiana J^e é dada por:

$$J^{e^{-1}} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{\det J^e} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \tag{5.40}$$

As derivadas cartesianas das funções de interpolação podem ser obtidas pela regra da derivação em cadeia:

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} = J^{e^{-1}} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} \tag{5.41}$$

Dessa maneira, (5.36) e (5.37) ficam:

$$K^e = \int_{\xi=-1}^{\xi=1} \int_{\eta=-1}^{\eta=1} \left(B^T \hat{D} B \right) \det J^e d\xi d\eta \tag{5.42}$$

$$f^e = \int_{\xi=-1}^{\xi=1} \int_{\eta=-1}^{\eta=1} (-N_i m_x - N_i m_y + N_i p_z) \det J^e d\xi d\eta \tag{5.43}$$

Notar que os coeficientes da matriz B em (5.42) deverão estar expressos através de derivadas no sistema ξ, η. Para isso, basta substituir (5.41) em (5.33).

Numericamente as equações (5.42) e (5.43) podem ser escritas utilizando a regra de quadratura de Gauss para retângulos, conforme Dunavant[13]. Assim, tem-se para cada elemento finito.

$$K_i^e \approx \sum_{k=1}^{n} \left[\left(B^T \hat{D} B \right) \det J^e \right]_{\substack{\xi=\xi_k \\ \eta=\eta_k}} W_k \tag{5.44}$$

$$f_i^e \approx \sum_{k=1}^{n} \left[(-N_i m_x - N_i m_y + N_i p_z) \det J^e \right]_{\substack{\xi=\xi_k \\ \eta=\eta_k}} W_k \tag{5.45}$$

W_k representa o peso do k-ésimo ponto de integração e n expressa o número total de pontos de integração em função da ordem polinomial do elemento.

5.1.3 Teoria de Reissner-Mindlin para placas

A *Teoria de Reissner-Mindlin* é obtida da *Teoria de Kirchoff* pela modificação da hipótese de ortogonalidade da normal durante a deformação da placa.

Hipótese cinemática

As hipóteses da *Teoria de Kirchoff* ficam mantidas e aplica-se uma modificação somente na hipótese em que as fibras normais à superfície média de referência da placa permanecem retas. Nessa teoria as fibras não mais necessariamente permanecem perpendiculares à superfície de referência deformada.

Adota-se o sistema de convenção de sinais para os deslocamentos e rotações ilustrados na Figura 5.1, caracterizado pela regra da mão direita. O deslocamento de um ponto pertencente ao plano médio da placa é expresso pela cinemática de deslocamentos por:

$$\bar{\mathbf{u}} = \begin{bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{bmatrix} = \begin{bmatrix} -z\hat{\theta}_x \\ -z\hat{\theta}_y \\ u_z \end{bmatrix} = \begin{bmatrix} z\theta_y \\ -z\theta_x \\ u_z \end{bmatrix} \tag{5.46}$$

A Figura 5.3 considera a *Teoria de Reissner-Mindlin*. Dessa forma tem-se um meio termo entre entre a posição da fibra no estado inicial e a fibra considerando a *Teoria de Kirchoff*. Observa-se na Figura 5.3 que as rotações $\hat{\theta}_x$ e $\hat{\theta}_y$ representam a regra simplificadora de convenção de sinais, anteriormente discutida e não adotada no trabalho.

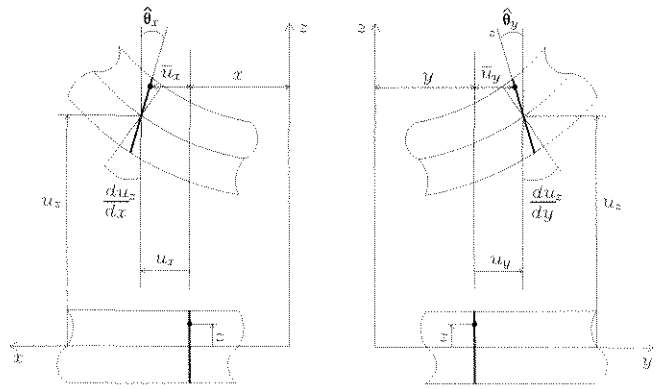

Figura 5.3: Deformação de uma placa de Reissner-Mindlin.

Deformações

A matriz jacobiana de deslocamentos, partindo de (5.46) é dada pela seguinte expressão matricial:

$$\jmath = \frac{\partial\left(\bar{u}_x, \bar{u}_y, \bar{u}_z\right)}{\partial\left(x, y, z\right)} = \begin{bmatrix} z\frac{\partial\theta_y}{\partial x} & z\frac{\partial\theta_y}{\partial y} & \theta_y \\ -z\frac{\partial\theta_x}{\partial x} & -z\frac{\partial\theta_x}{\partial y} & -\theta_x \\ \frac{\partial u_z}{\partial x} & \frac{\partial u_z}{\partial y} & 0 \end{bmatrix} \tag{5.47}$$

Pela hipótese dos pequenos deslocamentos e deformações, o tensor das deformações E sob linearidade geométrica é definido por:

$$\mathbf{E} = \frac{1}{2}\left(\jmath + \jmath^T\right) \tag{5.48}$$

Substituindo (5.47) em (5.48) obtém-se:

$$\mathbf{E} = \begin{bmatrix} z\left(\frac{\partial\theta_y}{\partial x}\right) & -z\left(\frac{\partial\theta_x}{\partial x} - \frac{\partial\theta_y}{\partial y}\right) & \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\ -z\left(\frac{\partial\theta_x}{\partial x} - \frac{\partial\theta_y}{\partial y}\right) & -z\left(\frac{\partial\theta_x}{\partial y}\right) & \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\ \left(\frac{\partial u_z}{\partial x} + \theta_y\right) & \left(\frac{\partial u_z}{\partial y} - \theta_x\right) & 0 \end{bmatrix} \tag{5.49}$$

As componentes não nulas do tensor das deformações E podem ser escritas num vetor de dimensão 6×1 por:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} E_{xx} \\ E_{yy} \\ E_{zz} \\ 2E_{yz} \\ 2E_{xz} \\ 2E_{xy} \end{bmatrix} \tag{5.50}$$

Reescrevendo (5.50) em função das derivadas parciais dos deslocamentos tem–se:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} z\frac{\partial \theta_y}{\partial x} \\ -z\frac{\partial \theta_x}{\partial y} \\ 0 \\ \frac{\partial u_z}{\partial y} - \theta_x \\ \frac{\partial u_z}{\partial x} + \theta_y \\ -z\left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \end{bmatrix} \tag{5.51}$$

onde γ_{yz}, γ_{xz} e γ_{xy} são as componentes de distorções e ε_x e ε_y são as deformações nas direções x e y existentes na placa.

Observa-se que para a *Teoria de Reissner-Mindlin* as componentes γ_{yz} e γ_{xz} existem e contribuem para a distorção nos planos yz e xz, caracterizando, com isso, a existência de deformações cisalhantes para este elemento da placa.

Tensões

Considera-se na presente teoria de placas o estado plano de tensões. Com isso, o tensor das tensões pode ser escrito na forma:

$$\vec{T} = \sigma_x \vec{e}_x + \sigma_y \vec{e}_y + \tau_{yz}\vec{e}_z + \tau_{xz}\vec{e}_z + \tau_{xy}\vec{e}_y \tag{5.52}$$

onde τ_{yz}, τ_{xz} e τ_{xy} são as tensões atuantes nos planos normais aos vetores $\vec{e}_y$, $\vec{e}_x$ e $\vec{e}_x$ respectivamente. As tensões σ_x e σ_y atuam normais ao plano formado pelos vetores $\vec{e}_y\vec{e}_z$ e $\vec{e}_x\vec{e}_z$, respectivamente.

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix} \tag{5.53}$$

Esforços solicitantes

Os esforços solicitantes em um elemento de placa são forças cortantes V_{xz} e V_{yz}, momentos fletores M_x, M_y e momento volvente M_{xy}, definidos por:

$$\begin{aligned} V_{xz} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{xz} dz \\ V_{yz} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{yz} dz \\ M_x &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_x dz \\ M_y &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_y dz \\ M_{xy} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\tau_{xy} dz \end{aligned} \tag{5.54}$$

Relação tensão-deformação

Análogo ao apresentado para a placa de *Kirchoff*, a equação constitutiva da elasticidade é formalmente representada por:

$$\sigma = D\varepsilon$$

Na seção transversal de uma placa observa-se que a distribuição de tensões cisalhantes não é constante na espessura. Geralmente esta distribuição possui a forma parabólica com valores nulos nas faces superior e inferior da placa. Para contornar este problema, aplica-se nas tensões cisalhantes transversais um coeficiente de modo que o trabalho de deformação das mesmas coincida

com o trabalho realizado pelas tensões transversais "exatas". Assim, pode-se afirmar que o trabalho de deformação global da placa coincide com o exato, ao passo que localmente as tensões tangenciais não possuem a distribuição correta.

Estes coeficientes conhecidos como coeficientes de distorção transversal são obtidos a partir de um procedimento energético. Em placas de espessura constante e material homogêneo seu valor é $\alpha = \frac{5}{6}$(Zienkiewicz & Taylor[39]), o qual será considerado a partir da expressão que segue.

Com isso, matriz constitutiva[1] D considerando o material elástico, linear e isotrópico, é definida por:

$$D = \frac{E}{(1-\nu^2)} \begin{bmatrix} \alpha\frac{(1-\nu)}{2} & 0 & 0 & 0 & 0 \\ 0 & \alpha\frac{(1-\nu)}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & \nu & 0 \\ 0 & 0 & \nu & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{5.55}$$

onde D, pode ser redefinida em duas parcelas, a seguir:

$$D_c = \frac{E}{(1-\nu^2)} \begin{bmatrix} \alpha\frac{(1-\nu)}{2} & 0 \\ 0 & \alpha\frac{(1-\nu)}{2} \end{bmatrix} \quad \text{e} \quad D_f = \frac{E}{(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \nu \end{bmatrix} \tag{5.56}$$

Esforços e deformações generalizadas

O vetor dos esforços internos ou tensões generalizadas $\hat{\sigma}$ para a teoria de placas é definido por:

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} V_{xz} \\ V_{yz} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \begin{bmatrix} \tau_{xz} \\ \tau_{yz} \\ z\sigma_x \\ z\sigma_y \\ z\tau_{xy} \end{bmatrix} dz \tag{5.57}$$

Das equações constitutivas tem-se a relação tensão-deformação em placas expressa como segue:

[1] As parcelas D_c e D_f da matriz constitutiva em (5.56) reportam a contribuição do material em relação à rigidez de cisalhamento e flexão, respectivamente.

$$
\begin{aligned}
\sigma_x &= \frac{E}{(1-\nu^2)}\left(\varepsilon_x + \nu\varepsilon_y\right) \\
\sigma_y &= \frac{E}{(1-\nu^2)}\left(\nu\varepsilon_x + \varepsilon_y\right) \\
\tau_{yz} &= \alpha G\gamma_{yz} \\
\tau_{xz} &= \alpha G\gamma_{xz} \\
\tau_{xy} &= G\gamma_{xy}
\end{aligned} \tag{5.58}
$$

Substitui-se (5.58) em (5.54) e observando que para essa teoria as forças cortantes produzem trabalho, obtêm-se os esforços generalizados em um elemento de placa.

$$
\hat{\boldsymbol{\sigma}} = \begin{bmatrix} V_{xz} \\ V_{yz} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \begin{bmatrix} \alpha\int_{-\frac{t}{2}}^{+\frac{t}{2}} \left(G\gamma_{xz}\right) dz \\ \alpha\int_{-\frac{t}{2}}^{+\frac{t}{2}} \left(G\gamma_{yz}\right) dz \\ \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\left[\frac{E}{(1-\nu^2)}\left(\varepsilon_x + \nu\varepsilon_y\right)\right] dz \\ \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\left[\frac{E}{(1-\nu^2)}\left(\nu\varepsilon_x + \varepsilon_y\right)\right] dz \\ \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\left(G\gamma_{xy}\right) dz \end{bmatrix} \tag{5.59}
$$

Assim, os esforços solicitantes em (5.59) podem ser escritos por:

$$
\begin{aligned}
V_{xz} &= \alpha\int_{-\frac{t}{2}}^{+\frac{t}{2}} G\left(\frac{\partial u_z}{\partial x} + \theta_y\right) dz = \alpha t G\left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\
V_{yz} &= \alpha\int_{-\frac{t}{2}}^{+\frac{t}{2}} G\left(\frac{\partial u_z}{\partial y} - \theta_x\right) dz = \alpha t G\left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\
M_x &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\frac{E}{(1-\nu^2)}\left(\varepsilon_x + \nu\varepsilon_y\right) dz = \frac{Et^3}{12\left(1-\nu^2\right)}\left[\left(\frac{\partial\theta_y}{\partial x}\right) - \nu\left(\frac{\partial\theta_x}{\partial y}\right)\right] \\
M_y &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z\frac{E}{(1-\nu^2)}\left(\nu\varepsilon_x + \varepsilon_y\right) dz = \frac{Et^3}{12\left(1-\nu^2\right)}\left[\nu\left(\frac{\partial\theta_y}{\partial x}\right) - \left(\frac{\partial\theta_x}{\partial y}\right)\right] \\
M_{xy} &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} zG\gamma_{xy} dz = -G\frac{t^3}{12}\left(\frac{\partial\theta_x}{\partial x} - \frac{\partial\theta_y}{\partial y}\right)
\end{aligned} \tag{5.60}
$$

O vetor das tensões generalizadas é reescrito por:

$$\hat{\boldsymbol{\sigma}} = \begin{bmatrix} \alpha t G \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\ \alpha t G \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\ \frac{Et^3}{12(1-\nu^2)} \left[\left(\frac{\partial \theta_y}{\partial x}\right) - \nu \left(\frac{\partial \theta_x}{\partial y}\right)\right] \\ \frac{Et^3}{12(1-\nu^2)} \left[\nu \left(\frac{\partial \theta_y}{\partial x}\right) - \left(\frac{\partial \theta_x}{\partial y}\right)\right] \\ -G\frac{t^3}{12} \left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \end{bmatrix} = \hat{D}\hat{\varepsilon} \tag{5.61}$$

Como mostrado anteriormente, a matriz $\hat{D}$ pode ser dividida em duas parcelas

$$\hat{D} = \begin{bmatrix} \hat{D}_c & \vdots & \\ \cdots & \vdots & \cdots \\ & \vdots & \hat{D}_f \end{bmatrix} \tag{5.62}$$

onde:

$$\hat{D}_c = tD_c \qquad \text{e} \qquad \hat{D}_f = \frac{t^3}{12} D_f \tag{5.63}$$

O vetor das deformações generalizadas $\hat{\varepsilon}$, em (5.61), é expresso por:

$$\hat{\boldsymbol{\varepsilon}} = \begin{bmatrix} \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\ \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\ -\frac{\partial \theta_y}{\partial x} \\ \frac{\partial \theta_x}{\partial y} \\ \left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \end{bmatrix} \tag{5.64}$$

Ações

Neste trabalho as ações em um elemento de placa podem ser representados por forças e momentos pontuais ou distribuídos aplicados nos nós e na superfície do elemento respectivamente.

As forças pontuais são aplicadas na direção z global. Os momentos pontuais são aplicados nas direções x e y global. As forças distribuídas são aplicadas na superfície da placa e na direção do eixo local z e os momentos distribuídos são aplicados na superfície da placa na direção dos eixos x e y local.

$$\underbrace{\begin{aligned} p_x &= 0 \\ p_y &= 0 \\ p_z &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} f_z dz \\ m_x &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_x dz \\ m_y &= \int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_y dz \\ m_z &= 0 \end{aligned}}_{\textit{Forças e momentos distribuídos / área}} \quad \text{e} \quad \underbrace{\begin{aligned} F_1 &= 0 \quad \text{(força na direção x)} \\ F_2 &= 0 \quad \text{(força na direção y)} \\ F_3 &\quad \text{(força na direção z)} \\ M_1 &\quad \text{(momento em torno x)} \\ M_2 &\quad \text{(momento em torno y)} \\ M_3 &= 0 \quad \text{(momento em torno z)} \end{aligned}}_{\textit{Forças e momentos concentrados}} \tag{5.65}$$

A convenção de sinais é expressa pela regra da mão direita. As forças de corpo f_x, f_y e f_z apresentadas na formulação expressam forças por unidade de volume.

Formulação variacional

O trabalho virtual dos esforços internos de uma barra é igual ao trabalho virtual dos esforços externos e sob condições de contorno essenciais pode-se demonstrar sua equivalência com as equações diferenciais que expressam o equilíbrio. Na formulação que segue para facilitar a notação, eventualmente serão usados os índices 1, 2 e 3 em correspondencia a x, y e z.

A partir do *Princípio dos Trabalhos Virtuais* tem-se:

$$\int_{\Omega} \delta\varepsilon_{ij}\sigma_{ij} d\Omega - \int_{\Omega} \delta\bar{u}_i F_i d\Omega - \int_{\Gamma} \delta u_i T_i d\Gamma = 0 \tag{5.66}$$

O domínio pode ser representado pela soma dos sub-domínios e ao considerar $dA = dxdy$ pode-se escrever:

$$\int_{\Omega} \ldots d\Omega = \sum_{e=1}^{n_{el}} \int_{\Omega_e} \ldots d\Omega_e = \sum_{e=1}^{n_{el}} \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \ldots dz dA_e \tag{5.67}$$

Aplica-se as deformações existentes e dadas as considerações estabelecidas em (5.67) tem-se:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left[\delta\varepsilon_x \sigma_x + \delta\varepsilon_y \sigma_y + \delta\gamma_{yz}\tau_{yz} + \delta\gamma_{xz}\tau_{xz} + \delta\gamma_{xy}\tau_{xy} \right] dz dA_e \right. \\
& - \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left(\delta\bar{u}_x f_x + \delta\bar{u}_y f_y + \delta\bar{u}_z f_z \right) dz dA_e \\
& \left. - \int_{\Gamma_e} \left(\delta u_3 F_3 + \sum_{k=1}^{2} \delta\theta_k^T M_k \right) d\Gamma_e \right\} \qquad (5.68)
\end{aligned}
$$

Substituindo (5.51) em (5.68) obtém-se:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left[z \left(\frac{\partial \delta\theta_y}{\partial x} \right) \sigma_x - z \left(\frac{\partial \delta\theta_x}{\partial y} \right) \sigma_y + \left(\frac{\partial \delta u_z}{\partial y} - \delta\theta_x \right) \tau_{yz} \right. \right. \\
& \left. + \left(\frac{\partial \delta u_z}{\partial x} + \delta\theta_y \right) \tau_{xz} - z \left(\frac{\partial \delta\theta_x}{\partial x} - \frac{\partial \delta\theta_y}{\partial y} \right) \tau_{xy} \right] dz dA_e \\
& - \int_{A_e} \int_{-\frac{t}{2}}^{+\frac{t}{2}} \left[(z\delta\theta_y) f_x + (-z\delta\theta_x) f_y + \delta u_z f_z \right] dz dA_e \\
& \left. - \int_{\Gamma_e} \left(\delta u_3 F_3 + \sum_{k=1}^{2} \delta\theta_k^T M_k \right) d\Gamma_e \right\} \qquad (5.69)
\end{aligned}
$$

Reordenando (5.69) é possível escrevê-la em função de deslocamentos virtuais associados aos correspondentes esforços:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \int_{A_e} \left[\left(\frac{\partial \delta\theta_y}{\partial x} \right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_x dz}_{M_x} - \left(\frac{\partial \delta\theta_x}{\partial y} \right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z\sigma_y dz}_{M_y} \right. \right. \\
& \left. \left. + \left(\frac{\partial \delta u_z}{\partial y} - \delta\theta_x \right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{yz} dz}_{V_{yz}} + \left(\frac{\partial \delta u_z}{\partial x} + \delta\theta_y \right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{xz} dz}_{V_{xz}} \right.
\end{aligned}
$$

$$
\begin{aligned}
&- z\left(\frac{\partial \delta\theta_x}{\partial x} - \frac{\partial \delta\theta_y}{\partial y}\right) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} \tau_{xy} dz}_{M_{xy}} \Bigg] dA_e - \int_{A_e} \Bigg[(\delta\theta_y) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_x dz}_{m_x} \\
&- (\delta\theta_x) \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} z f_y dz}_{m_y} + \delta u_z \underbrace{\int_{-\frac{t}{2}}^{+\frac{t}{2}} f_z dz}_{p_z} \Bigg] dA_e \\
&- \int_{\Gamma_e} \left(\delta u_3 F_3 + \sum_{k=1}^{2} \delta\theta_k^T M_k \right) d\Gamma_e \Bigg\}
\end{aligned} \tag{5.70}
$$

Aplica-se (5.60) em (5.70) obtém-se:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \Bigg\{ \int_{A_e} \Bigg[\left(\frac{\partial \delta\theta_y}{\partial x}\right) \frac{Et^3}{12\left(1-\nu^2\right)} \left[\left(\frac{\partial \theta_y}{\partial x}\right) - \nu \left(\frac{\partial \theta_x}{\partial y}\right) \right] \\
& - \left(\frac{\partial \delta\theta_x}{\partial y}\right) \frac{Et^3}{12\left(1-\nu^2\right)} \left[\nu \left(\frac{\partial \theta_y}{\partial x}\right) - \left(\frac{\partial \theta_x}{\partial y}\right) \right] \\
& + \left(\frac{\partial \delta u_z}{\partial y} - \delta\theta_x\right) tG \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\
& + \left(\frac{\partial \delta u_z}{\partial x} + \delta\theta_y\right) tG \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\
& + z\left(\frac{\partial \delta\theta_x}{\partial x} - \frac{\partial \delta\theta_y}{\partial y}\right) G\frac{t^3}{12} \left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \Bigg] dA_e \\
& - \int_{A_e} \left[(\delta\theta_y) m_x - (\delta\theta_x) m_y + \delta u_z p_z \right] dA_e \\
& - \int_{\Gamma_e} \left(\delta u_3 F_3 + \sum_{k=1}^{2} \delta\theta_k^T M_k \right) d\Gamma_e \Bigg\}
\end{aligned} \tag{5.71}
$$

Reescrevendo (5.71) tem-se a formulação variacional do problema dado por:

$$
0 \;=\; \sum_{e=1}^{n_{el}} \Bigg\{ \int_{A_e} \Bigg[\left(\frac{\partial \delta\theta_y}{\partial x}\right) \frac{Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_y}{\partial x}\right) - \left(\frac{\partial \delta\theta_y}{\partial x}\right) \frac{\nu Et^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_x}{\partial y}\right)
$$

$$
\begin{aligned}
&- \left(\frac{\partial \delta\theta_x}{\partial y}\right) \frac{\nu E t^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_y}{\partial x}\right) + \left(\frac{\partial \delta\theta_x}{\partial y}\right) \frac{E t^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_x}{\partial y}\right) \\
&+ \left(\frac{\partial \delta u_z}{\partial y} - \delta\theta_x\right) tG \left(\frac{\partial u_z}{\partial y} - \theta_x\right) + \left(\frac{\partial \delta u_z}{\partial x} + \delta\theta_y\right) tG \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\
&+ \left(\frac{\partial \delta\theta_x}{\partial x} - \frac{\partial \delta\theta_y}{\partial y}\right) G \frac{t^3}{12} \left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right)\Bigg] dA_e \\
&- \int_{A_e} \left[\left(\delta\theta_y\right) m_x - \left(\delta\theta_x\right) m_y + \delta u_z p_z\right] dA_e \\
&- \int_{\Gamma_e} \left(\delta u_3 F_3 + \sum_{k=1}^{2} \delta\theta_k^T M_k\right) d\Gamma_e \Bigg\}
\end{aligned} \tag{5.72}
$$

O sistema é simétrico, bilinear e pode ser escrito por:

$$
a\left(\delta u, u\right) = \left(\delta u, f\right) + \left(\delta u, T\right)_{\Gamma} \tag{5.73}
$$

Cada termo é representado por:

$$
\begin{aligned}
a\left(\delta u, u\right) &= \sum_{e=1}^{n_{el}} \Bigg\{ \int_{A_e} \Bigg[\left(\frac{\partial \delta\theta_y}{\partial x}\right) \frac{E t^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_y}{\partial x}\right) \\
&\quad - \left(\frac{\partial \delta\theta_y}{\partial x}\right) \frac{\nu E t^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_x}{\partial y}\right) \\
&\quad - \left(\frac{\partial \delta\theta_x}{\partial y}\right) \frac{\nu E t^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_y}{\partial x}\right) \\
&\quad + \left(\frac{\partial \delta\theta_x}{\partial y}\right) \frac{E t^3}{12\left(1-\nu^2\right)} \left(\frac{\partial \theta_x}{\partial y}\right) \\
&\quad + \left(\frac{\partial \delta u_z}{\partial y} - \delta\theta_x\right) tG \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\
&\quad + \left(\frac{\partial \delta u_z}{\partial x} + \delta\theta_y\right) tG \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\
&\quad + \left(\frac{\partial \delta\theta_x}{\partial x} - \frac{\partial \delta\theta_y}{\partial y}\right) G \frac{t^3}{12} \left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right)\Bigg] dA_e \Bigg\} \\
\left(\delta u, f\right) &= \sum_{e=1}^{n_{el}} \int_{A_e} \left[\left(\delta\theta_y\right) m_x - \left(\delta\theta_x\right) m_y + \delta u_z p_z\right] dA_e
\end{aligned}
$$

$$(\delta u, T)_\Gamma = \sum_{e=1}^{n_{el}} \int_{\Gamma_e} \left(\delta u_3 F_3 + \sum_{k=1}^{2} \delta\theta_k^T M_k \right) d\Gamma_e \tag{5.74}$$

Discretização utilizando o método dos elementos finitos

Para obter a formulação de elementos finitos de placa pela *Teoria de Reissner-Mindlin*, necessita-se escrever os deslocamentos u em termos de deslocamentos nodais u_p, mediante a definição de uma base de funções de interpolação como segue:

$$u = N u_p \tag{5.75}$$

onde N é a matriz de funções de interpolação utilizada para encontrar a solução numérica aproximada do problema.

Conforme Oñate[21], a matriz dos coeficientes de interpolação N_i, que aparece nas expressões abaixo, tem seu índice i dado pela ordem do polinômio de interpolação utilizado na aproximação.

O vetor das deformações generalizadas $\hat{\varepsilon}$ pode ser escrito matricialmente na forma:

$$\hat{\varepsilon} = \begin{bmatrix} \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\ \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \\ -\frac{\partial \theta_y}{\partial x} \\ \frac{\partial \theta_x}{\partial y} \\ \left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \end{bmatrix} = \sum_{i=1}^{n_{el}} \begin{bmatrix} \left(\frac{\partial N_i}{\partial x} u_{z_i} + N_i \theta_{y_i}\right) \\ \left(\frac{\partial N_i}{\partial y} u_{z_i} - N_i \theta_{x_i}\right) \\ -\left(\frac{\partial N_i}{\partial x}\right) \theta_{y_i} \\ \left(\frac{\partial N_i}{\partial y}\right) \theta_{x_i} \\ \left(\frac{\partial N_i}{\partial x} \theta_{x_i} - \frac{\partial N_i}{\partial y} \theta_{y_i}\right) \end{bmatrix} = \sum_{i=1}^{n_{el}} B_i \begin{bmatrix} u_{x_i} \\ u_{y_i} \\ u_{z_i} \\ \theta_{x_i} \\ \theta_{y_i} \end{bmatrix} \tag{5.76}$$

Para a implementação computacional é necessário escrever a matriz B, que representa o operador diferencial aplicado às funções de interpolação da matriz de deformação generalizada.

Assim, por (5.76), tem-se que a matriz constitutiva[2] B é definida por:

[2] As submatrizes B_c e B_f em (5.78) representam os operadores diferenciais aplicados às funções de forma com relação a rigidez de cisalhamento e flexão, respectivamente.

$$B_i = \begin{bmatrix} 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & N_i \\ 0 & 0 & \frac{\partial N_i}{\partial y} & -N_i & 0 \\ 0 & 0 & 0 & 0 & -\frac{\partial N_i}{\partial x} \\ 0 & 0 & 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & -\frac{\partial N_i}{\partial y} \end{bmatrix} \tag{5.77}$$

A expressão (5.77) pode ser dividida em:

$$\begin{aligned} B_c &= \begin{bmatrix} 0 & 0 & \frac{\partial N_i}{\partial x} & 0 & N_i \\ 0 & 0 & \frac{\partial N_i}{\partial y} & -N_i & 0 \end{bmatrix} \\ B_f &= \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{\partial N_i}{\partial x} \\ 0 & 0 & 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & 0 & \frac{\partial N_i}{\partial x} & -\frac{\partial N_i}{\partial y} \end{bmatrix} \end{aligned} \tag{5.78}$$

A matriz de rigidez do elemento é calculada através da soma das parcelas de rigidez cisalhante e de flexão, como segue:

$$\begin{aligned} K^e &= k_c^e + k_f^e \\ k_c^e &= \int_{A_e} B_c^T \hat{D}_c B_c dA_e \qquad \textit{(rigidez cortante)} \\ k_f^e &= \int_{A_e} B_f^T \hat{D}_f B_f dA_e \qquad \textit{(rigidez a flexão)} \\ f^e &= \begin{cases} \int_{A_e} N_a p_z dA_e \\ -\int_{A_e} N_a m_x dA_e - \int_{\Gamma_e} N_a M_1 d\Gamma_e \\ -\int_{A_e} N_a m_y dA_e - \int_{\Gamma_e} N_a M_2 d\Gamma_e \end{cases} \end{aligned} \tag{5.79}$$

Comparando (5.79) com (5.72) obtém-se a formulação variacional do problema expressa em função das matrizes B e D e funções de forma associadas:

$$
\begin{aligned}
0 \;=\; & \sum_{e=1}^{n_{el}} \left\{ \left[\int_{A_e} B_c^T \hat{D}_c B_c + B_f^T \hat{D}_f B_f \right] dA_e \right. \\
& - \int_{A_e} \left[-N_i m_x - N_i m_y + N_i p_z \right] dA_e \\
& \left. - \int_{\Gamma_e} \left(N_i F_3 + \sum_{k=1}^{2} N_i^T M_k \right) d\Gamma_e \right\}
\end{aligned} \tag{5.80}
$$

Em (5.80), as parcelas da matriz de rigidez e vetor de cargas podem ser separadas e expressas por:

$$
K_i^e \;=\; \int_{A_e} \left(B_c^T \hat{D}_c B_c + B_f^T \hat{D}_f B_f \right) dA_e \tag{5.81}
$$

$$
f_i^e \;=\; \int_{A_e} \left(-N_i m_x - N_i m_y + N_i p_z \right) dA_e \tag{5.82}
$$

Ao mapear para o sistema de coordenadas padrão, definido por ξ e η, o diferencial de superfície do elemento é dado por dA_e e pode ser escrito por

$$
dA_e = \det J^e d\xi d\eta \tag{5.83}
$$

onde J^e é a matriz jacobiana dada por:

$$
J^e = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} x_i & \frac{\partial N_i^e}{\partial \xi} y_i \\ \frac{\partial N_i^e}{\partial \eta} x_i & \frac{\partial N_i^e}{\partial \eta} y_i \end{bmatrix} \tag{5.84}
$$

Onde x_i e y_i representam as coordenadas nodais do elemento em relação ao sistema global de coordenadas.

A inversa da matriz jacobiana J^e é dada por:

$$
J^{e^{-1}} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{\det J^e} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \tag{5.85}
$$

As derivadas cartesianas das funções de interpolação são obtidas pela regra da derivação em cadeia.

$$\begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} = J^{e^{-1}} \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{bmatrix} \tag{5.86}$$

Dessa maneira, (5.81) e (5.82) ficam expressas por:

$$K_i^e = \int_{\xi=-1}^{\xi=1} \int_{\eta=-1}^{\eta=1} \left(B_c^T \hat{D}_c B_c + B_f^T \hat{D}_f B_f \right) \det J^e d\xi d\eta \tag{5.87}$$

$$f_i^e = \int_{\xi=-1}^{\xi=1} \int_{\eta=-1}^{\eta=1} (-N_i m_x - N_i m_y + N_i p_z) \det J^e d\xi d\eta \tag{5.88}$$

Numericamente as equações (5.87) e (5.88) são escritas utilizando a regra de quadratura de Gauss para retângulos, conforme Dunavant[13]. Assim, tem-se para cada elemento finito.

$$K^e \approx \sum_{k=1}^{n} \left[\left(B_c^T \hat{D}_c B_c + B_f^T \hat{D}_f B_f \right) \det J^e \right]_{\substack{\xi=\xi_k \\ \eta=\eta_k}} W_k \tag{5.89}$$

$$f_i^e \approx \sum_{k=1}^{n} \left[(-N_i m_x - N_i m_y + N_i p_z) \det J^e \right]_{\substack{\xi=\xi_k \\ \eta=\eta_k}} W_k \tag{5.90}$$

W_k representa o peso do k-ésimo ponto de integração e n expressa o número total de pontos de integração em função da ordem polinomial do elemento.

5.1.4 Extensão da teoria placa de Reissner-Mindlin para cascas curvas

Em face da grande evolução conseguida em estudos dos meios contínuos, tem sido dada relativamente pouca ênfase aos sistemas que associam elementos espaciais de cascas curvos.

Sugere-se aqui resolver o problema de cascas curvas com qualquer desenvolvimento geométrico no espaço tridimensional através do Método dos Elementos Finitos pela associação de elementos de placa curvos com seis graus de liberdade por nó, três translações e três rotações.

Na formulação foram tomados os cuidados para que o elemento de placa pudesse tomar qualquer posição no espaço. Assim, na modelagem do problema via Método dos Elementos Finitos, associou-se em cada elemento no espaço um sistema de eixos cartesianos (ξ, η, ζ) e o equilíbrio de cada elemento foi calculado em relação a estes eixos. Foram considerados dois tipos de elementos: retangular linear de 4 nós e retangular quadrático de 9 nós. Esses elementos foram utilizados para simular alguns exemplos particulares de cascas onde a formulação quadrática foi testada.

Os deslocamentos a considerar no problema da placa são duas translações e duas rotações segundo eixos cartesianos contidos no plano da placa e mais a translação perpendicular a esse plano. Nos problemas de casca os deslocamentos de translação no plano do elemento contabilizam o comportamento de membrana na placa.

Deformações e tensões para cascas no plano xy

Conforme Timoshenko[34], o problema da placa elástica pode ser estudado considerando-se cinco componentes de deformação: três devidas à flexão e duas devidas ao cisalhamento. Na formulação para cascas, o efeito de membrana também contribui. Utiliza-se a notação (u_x, u_y, u_z) para deslocamento e $(\theta_x, \theta_y, \theta_z)$ para rotação. Supõe-se inicialmente que o elemento esteja localizado no plano xy; o vetor das deformações totais é dado por:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_x \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \underbrace{\begin{bmatrix} z\frac{\partial \theta_y}{\partial x} \\ -z\frac{\partial \theta_x}{\partial y} \\ -z\left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \\ \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\ \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \end{bmatrix}}_{placa} + \underbrace{\begin{bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \\ 0 \\ 0 \end{bmatrix}}_{membrana} \tag{5.91}$$

A primeira parcela refere-se à contribuição de flexão e cisalhamento transversal e a segunda parcela é a contribuição do efeito de membrana. Em (5.91), as três primeiras componentes do vetor representam deformações por flexão (efeito de placa + efeito de membrana) e as duas últimas parcelas representam deformações por cisalhamento transversal.

$$\boldsymbol{\varepsilon} = \underbrace{\begin{bmatrix} z\hat{\varepsilon}_f \\ \cdots \\ \hat{\varepsilon}_c \end{bmatrix}}_{placa} + \underbrace{\begin{bmatrix} \hat{\varepsilon}_m \\ \cdots \\ 0 \end{bmatrix}}_{membrana} \tag{5.92}$$

O tensor das deformações pode ser desmembrado em três partes distintas, expostas a seguir:

$$\begin{aligned} \hat{\varepsilon}_m &= \begin{bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \end{bmatrix} \\ \hat{\varepsilon}_f &= \begin{bmatrix} \frac{\partial \theta_y}{\partial x} \\ -\frac{\partial \theta_x}{\partial y} \\ -\left(\frac{\partial \theta_x}{\partial x} - \frac{\partial \theta_y}{\partial y}\right) \end{bmatrix} \\ \hat{\varepsilon}_c &= \begin{bmatrix} \left(\frac{\partial u_z}{\partial x} + \theta_y\right) \\ \left(\frac{\partial u_z}{\partial y} - \theta_x\right) \end{bmatrix} \end{aligned} \tag{5.93}$$

O vetor dos esforços locais em um ponto do plano médio é definido por:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_m \\ \cdots \\ \sigma_f \\ \cdots \\ \sigma_c \end{bmatrix} = \begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ \cdots \\ M_x \\ M_y \\ M_{xy} \\ \cdots \\ V_{xz} \\ V_{yz} \end{bmatrix} = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \\ \cdots \\ z\sigma_x \\ z\sigma_y \\ z\tau_{xy} \\ \cdots \\ \tau_{xz} \\ \tau_{yz} \end{bmatrix} dz = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \begin{bmatrix} \hat{\sigma}_f \\ \cdots \\ z\hat{\sigma}_f \\ \cdots \\ \hat{\sigma}_c \end{bmatrix} dz \tag{5.94}$$

Supõe-se um elemento curvo caracterizado pelos parâmetros ξ e η. Assim, as coordenadas de um ponto do elemento são dadas por $(x(\xi), y(\xi), z(\xi))$.

Baseado nisso, um vetor unitário na direção do eixo ξ do elemento deformado é dado por

$$\vec{v}_1 = \frac{\left(\frac{\partial x(\xi)}{\partial \xi}, \frac{\partial y(\xi)}{\partial \xi}, \frac{\partial z(\xi)}{\partial \xi}\right)}{\sqrt{\left(\frac{\partial x(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial y(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial z(\xi)}{\partial \xi}\right)^2}} \tag{5.95}$$

e um vetor intermediário unitário $\tilde{v}$ na direção do eixo η do elemento deformado, é definido por:

$$\tilde{v} = \frac{\left(\frac{\partial x(\eta)}{\partial \eta}, \frac{\partial y(\eta)}{\partial \eta}, \frac{\partial z(\eta)}{\partial \eta}\right)}{\sqrt{\left(\frac{\partial x(\eta)}{\partial \eta}\right)^2 + \left(\frac{\partial y(\eta)}{\partial \eta}\right)^2 + \left(\frac{\partial z(\eta)}{\partial \eta}\right)^2}} \tag{5.96}$$

Associado ao vetor $\vec{v}_1$ e ao vetor intermediário, conforme ilustra a Figura 5.4, definem-se os vetores $\vec{v}_2$ e $\vec{v}_3$.

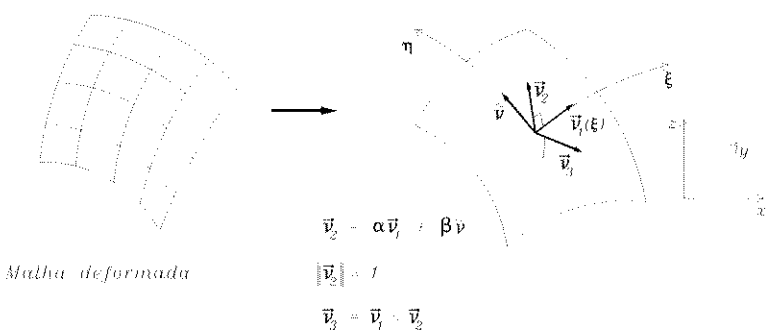

Figura 5.4: Sistema de eixos associado ao ponto de integração do elemento de casca.

Com isso, qualquer ponto da seção é localizado pelas coordenadas:

$$\begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix}^T \begin{bmatrix} \vec{u}_x \\ \vec{u}_y \\ \vec{u}_z \end{bmatrix}$$

$$\begin{bmatrix} \vec{\theta}_1 \\ \vec{\theta}_2 \\ \vec{\theta}_3 \end{bmatrix} = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \vec{v}_3 \end{bmatrix}^T \begin{bmatrix} \vec{\theta}_x \\ \vec{\theta}_y \\ \vec{\theta}_z \end{bmatrix} \tag{5.97}$$

Assim as expressões que representam o vetor das deformações generalizadas podem ser reescrito por:

$$\begin{aligned} \hat{\varepsilon}_m &= \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ \frac{\partial u_2}{\partial y} \\ \left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right) \end{bmatrix} \\ \hat{\varepsilon}_f &= \begin{bmatrix} \frac{\partial \theta_2}{\partial x} \\ -\frac{\partial \theta_1}{\partial y} \\ -\left(\frac{\partial \theta_1}{\partial x} - \frac{\partial \theta_2}{\partial y}\right) \end{bmatrix} \qquad \hat{\varepsilon}_c = \begin{bmatrix} \left(\frac{\partial u_3}{\partial x} + \theta_2\right) \\ \left(\frac{\partial u_3}{\partial y} - \theta_1\right) \end{bmatrix} \end{aligned} \tag{5.98}$$

Jacobiano bidimensional para coordenadas cartesianas

Partindo do exposto para o caso unidimensional , considera-se o Jacobiano em duas dimensões calculado como:

$$\begin{aligned} J_1^e &= \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\right)^2} \\ J_2^e &= \sqrt{\left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2 + \left(\frac{\partial z}{\partial \eta}\right)^2} \end{aligned} \tag{5.99}$$

Para o caso bidimensional o Jacobiano é obtido por um procedimento bastante simples, dependendo de um cálculo prévio dos vetores $\vec{v}_1$, $\vec{v}_2$ e $\vec{v}_3$ para sua determinação.

Adota-se a princípio o vetor $\vec{v}_1$ na direção ξ do elemento e considera-se um segundo vetor intermediário $\tilde{v}$ na direção η do elemento, como ilustra a Figura 5.4. Logo, esses vetores unitários são expressos por:

$$\vec{v}_1 = \frac{1}{J_\xi}\left(\frac{\partial x}{\partial \xi}, \frac{\partial y}{\partial \xi}, \frac{\partial z}{\partial \xi}\right)$$

$$\tilde{v} = \frac{1}{J_\eta}\left(\frac{\partial x}{\partial \eta}, \frac{\partial y}{\partial \eta}, \frac{\partial z}{\partial \eta}\right) \tag{5.100}$$

Como $\vec{v}_1$ e $\tilde{v}$ são linearmente independentes, então todos os vetores da forma $\alpha\vec{v}_1 + \beta\tilde{v}$ podem ser representados sobre um mesmo plano.

$$\vec{v}_2 = \alpha\vec{v}_1 + \beta\tilde{v} \tag{5.101}$$

Conforme Devloo[12], o vetor $\vec{v}_2$ é dado pela combinação linear entre $\vec{v}_1$ e $\tilde{v}$. Ao impor a condição de que o produto escalar entre o vetor $\vec{v}_1$ e $\vec{v}_2$ seja igual a zero e ainda, que o módulo de $\|\vec{v}_2\| = 1$, torna-se possível determinar o vetor $\vec{v}_2$.

Ao multiplicar ambos os lados da combinação linear por $\vec{v}_1$, segue que:

$$\begin{aligned}
\vec{v}_1 \cdot \vec{v}_2 &= \alpha\vec{v}_1 \cdot \vec{v}_1 + \beta\tilde{v} \cdot \vec{v}_1 \\
0 &= \alpha + \beta\left(\tilde{v} \cdot \vec{v}_1\right) \\
\vec{v}_2 &= \beta\left[\tilde{v} - \left(\tilde{v} \cdot \vec{v}_1\right)\vec{v}_1\right] \\
\vec{v}_2 &= \frac{\tilde{v} - \left(\tilde{v} \cdot \vec{v}_1\right)\vec{v}_1}{\left\|\tilde{v} - \left(\tilde{v} \cdot \vec{v}_1\right)\vec{v}_1\right\|}
\end{aligned} \tag{5.102}$$

O vetor $\vec{v}_3$ é dado por

$$\vec{v}_3 = \vec{v}_1 \times \vec{v}_2 \tag{5.103}$$

e o Jacobiano é definido como

$$J = \begin{bmatrix} \frac{\partial \vec{v}_1}{\partial \xi} & \frac{\partial \vec{v}_1}{\partial \eta} \\ \frac{\partial \vec{v}_2}{\partial \xi} & \frac{\partial \vec{v}_2}{\partial \eta} \end{bmatrix} \tag{5.104}$$

onde:

$$\begin{aligned}
\frac{\partial \vec{v}_1}{\partial \xi} &= \frac{\partial \vec{v}_1}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \vec{v}_1}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial \vec{v}_1}{\partial z}\frac{\partial z}{\partial \xi} = \vec{v}_1 \cdot \vec{v}_1 J_1^e \\
\frac{\partial \vec{v}_1}{\partial \eta} &= \frac{\partial \vec{v}_1}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \vec{v}_1}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial \vec{v}_1}{\partial z}\frac{\partial z}{\partial \eta} = \vec{v}_1 \cdot \tilde{v} J_2^e \\
\frac{\partial \vec{v}_2}{\partial \xi} &= \frac{\partial \vec{v}_2}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \vec{v}_2}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial \vec{v}_2}{\partial z}\frac{\partial z}{\partial \xi} = \vec{v}_2 \cdot \vec{v}_1 J_1^e = 0 \\
\frac{\partial \vec{v}_2}{\partial \eta} &= \frac{\partial \vec{v}_2}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \vec{v}_2}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial \vec{v}_2}{\partial z}\frac{\partial z}{\partial \eta} = \vec{v}_2 \cdot \tilde{v} J_2^e
\end{aligned} \tag{5.105}$$

Finalmente o Jacobiano bidimensional é expresso como segue:

$$J^e = \begin{bmatrix} J_1^e & \vec{v}_1 \cdot \tilde{v} J_2^e \\ 0 & \vec{v}_2 \cdot \tilde{v} J_2^e \end{bmatrix} \tag{5.106}$$

Assim torna-se possível calcular em cada ponto de integração os deslocamentos e rotações em função da orientação dos eixos no ponto.

Capítulo 6

Acoplamento entre elementos

6.1 Associação de elementos de barra com elementos de placa

O objetivo deste capítulo é apresentar como foi feito o acoplamento das matrizes de rigidez dos elementos de barras e de placas, bem como o modo com que foi montado o vetor de cargas nodais, por contribuição das ações nodais equivalentes dos mesmos elementos, em programa elaborado no ambiente computacional PZ, com a finalidade de análise tridimensional de edifícios.

A técnica utilizada é semelhante a que foi empregada por Buzar[8] que analisou o problema de placas a partir de um elemento finito retangular de 16 nós com aproximação lagrangeana e por Santos da Silva[28] que associou os elementos de barra e placa para posterior análise elástica.

No ambiente computacional PZ, foi feita uma implementação computacional da associação de elementos finitos unidimensionais de barras baseados na *Teoria de Timoshenko* com elementos finitos de placas formulados com a *Teoria de Reissner-Mindlin*. O edifício é idealizado como a associação no espaço tridimensional de pilares, lajes e vigas.

Os pilares e vigas são discretizados como elementos finitos unidimensionais de barras com dois nós de extremidades. As lajes são discretizadas como elementos de placa com 4 nós de extremidade.

Do modo como implementado, o programa permite que sejam feitas análises de elementos curvos de barra e de casca.

Na linguagem C++ é importante observar que os índices dos coeficientes de matrizes e vetores possui numeração seqüencial que é iniciada no número

zero. Dessa forma a primeira componente de um vetor $\vec{v}$ é $v[0]$. Assim, para percorrer todas as componentes de um vetor de seis componentes, por exemplo, utiliza-se um contador inteiro que varia da posição 0 até a posição 5.

Como os coeficientes de rigidez são forças, no acoplamento basta somar vetorialmente as componentes de direções correspondentes obtendo, então, a matriz de rigidez global por contribuição das matrizes de rigidez dos elementos. Como as matrizes de rigidez dos elementos estão expressos em coordenadas locais, cujas direções coincidem com as direções das coordenadas globais, basta fazer a correspondência de índices e realizar a contribuição dos coeficientes de um elemento nos coeficientes globais a que estão conectados.

A mesma estratégia é usada para a obtenção das ações nodais.

As ações atuantes num elemento são transformadas em ações nodais equivalentes nos nós do elemento e por contribuição são superpostos nas ações nodais nas coordenadas globais dos nós onde o elemento é conectado.

6.1.1 Barra

A contribuição da rigidez de um elemento de barra espacial é escrito como:

$$K_B^n u_B^n = f_B^n \tag{6.1}$$

sendo K_B^n a matriz de rigidez do elemento que representa a contribuição da rigidez do elemento de barra para o nó n da estrutura; u_B^n e f_B^n são o vetor de deslocamentos e o vetor de cargas, relativos ao nó n.

Para o nó de barra, a contribuição deste nó na matriz de rigidez da estrutura é traduzida pelas expressões:

$$
\begin{aligned}
k^{b_n}_{(6i+idf,6j+jdf)} &= EA_x \left(N_i' N_j'\right) v_{(0,idf)} v_{(0,jdf)} w \\
k^{b_n}_{(6i+idf,6j+jdf)} &= GA_y \left(N_i' N_j'\right) v_{(1,idf)} v_{(1,jdf)} w \\
k^{b_n}_{(6i+idf,6j+jdf)} &= GA_z \left(N_i' N_j'\right) v_{(2,idf)} v_{(2,jdf)} w \\
k^{b_n}_{(6i+idf,6j+jdf+3)} &= -GA_y \left(N_i' N_j\right) v_{(1,idf)} v_{(2,jdf)} w \\
k^{b_n}_{(6i+idf,6j+jdf+3)} &= GA_z \left(N_i' N_j\right) v_{(2,idf)} v_{(1,jdf)} w \\
k^{b_n}_{(6i+idf+3,6j+jdf)} &= GA_z \left(N_i N_j'\right) v_{(1,idf)} v_{(2,jdf)} w \\
k^{b_n}_{(6i+idf+3,6j+jdf)} &= -GA_y \left(N_i N_j'\right) v_{(2,idf)} v_{(1,jdf)} w \\
k^{b_n}_{(6i+idf+3,6j+jdf+3)} &= GJ_0 \left(N_i' N_j'\right) v_{(0,idf)} v_{(0,jdf)} w \\
k^{b_n}_{(6i+idf+3,6j+jdf+3)} &= \left[GA_z \left(N_i N_j\right) + EJ_{yy} \left(N_i' N_j'\right)\right] v_{(1,idf)} v_{(1,jdf)} w \\
k^{b_n}_{(6i+idf+3,6j+jdf+3)} &= \left[GA_y \left(N_i N_j\right) + EJ_{zz} \left(N_i' N_j'\right)\right] v_{(2,idf)} v_{(2,jdf)} w
\end{aligned}
\tag{6.2}
$$

A nomenclatura apresentada em (6.2) é:

- i, j : grau do polinômio de interpolação $(1, ..., n_p)$, onde n_p representa a ordem do polinômio de aproximação;
- idf, jdf : graus de liberdade $(1, 2, 3)$;
- v : vetor de direção;
- w : peso no ponto de integração;
- n : nó n;
- N : funções de forma;
- A_x, A_y e A_z : áreas das seções transversais da barra;
- E : módulo de elasticidade longitudinal da barra;
- G : módulo de elasticidade transversal da barra;
- J_{yy}, J_{zz} e J_0 : momentos de inércia em y, z e momento polar, respectivamente.

O vetor de cargas é representado pela associação da função de forma aos carregamentos distribuidos, como segue:

$$\begin{aligned} f^{b_n}_{(6i,0)} &= N_i p_x \\ f^{b_n}_{(6i+1,0)} &= N_i p_y \\ f^{b_n}_{(6i+2,0)} &= N_i p_z \\ f^{b_n}_{(6i+3,0)} &= N_i m_x \\ f^{b_n}_{(6i+4,0)} &= N_i m_y \\ f^{b_n}_{(6i+5,0)} &= N_i m_z \end{aligned} \tag{6.3}$$

Os valores p_x, p_y, p_z, m_x, m_y e m_z representam forças e momentos distribuídos por unidade de comprimento.

6.1.2 Placa

A rigidez do elemento de placa espacial é escrito por:

$$K^n_P u^n_P = f^n_P \tag{6.4}$$

Onde K^n_P é a matriz de rigidez do elemento que representa a contribuição da rigidez do elemento de placa para o nó n da estrutura; u^n_P e f^n_P são o vetor de deslocamentos e o vetor de cargas, relativos ao nó n.

Para o nó de placa, a contribuição na matriz de rigidez na estrutura é traduzida pelas expressões:

$$
\begin{aligned}
k^{p_n}_{(6i+idf,6j+jdf)} &= h\left[\tfrac{E}{(1-\nu^2)}N_{i,x}N_{j,x} + GN_{i,y}N_{j,y}\right] v_{(0,idf)}v_{(0,jdf)}w \\
k^{p_n}_{(6i+idf,6j+jdf)} &= h\left[\tfrac{\nu E}{(1-\nu^2)}N_{i,x}N_{j,y} + GN_{i,y}N_{j,x}\right] v_{(0,idf)}v_{(1,jdf)}w \\
k^{p_n}_{(6i+idf,6j+jdf)} &= h\left[\tfrac{\nu E}{(1-\nu^2)}N_{i,y}N_{j,x} + GN_{i,x}N_{j,y}\right] v_{(1,idf)}v_{(0,jdf)}w \\
k^{p_n}_{(6i+idf,6j+jdf)} &= h\left[GN_{i,x}N_{j,x} + \tfrac{E}{(1-\nu^2)}N_{i,y}N_{j,y}\right] v_{(1,idf)}v_{(1,jdf)}w \\
k^{p_n}_{(6i+idf,6j+jdf)} &= \alpha Gh\left[N_{i,x}N_{j,x} + N_{i,y}N_{j,y}\right] v_{(2,idf)}v_{(2,jdf)}w \\
k^{p_n}_{(6i+idf,6j+jdf+3)} &= -\alpha Gh\left[N_{i,y}N_{j}\right] v_{(2,idf)}v_{(0,jdf)}w \\
k^{p_n}_{(6i+idf,6j+jdf+3)} &= \alpha Gh\left[N_{i,x}N_{j}\right] v_{(2,idf)}v_{(1,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf)} &= -\alpha Gh\left[N_{i}N_{j,y}\right] v_{(0,idf)}v_{(2,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf)} &= \alpha Gh\left[N_{i}N_{j,x}\right] v_{(1,idf)}v_{(2,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf+3)} &= \tfrac{Eh^3}{12(1-\nu^2)}\left[N_{i,y}N_{j,y} + \tfrac{(1-\nu)}{2}N_{i,x}N_{j,x} + \tfrac{6\alpha(1-\nu)}{h^2}N_iN_j\right] v_{(1,idf)}v_{(2,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf+3)} &= -\tfrac{Eh^3}{12(1-\nu^2)}\left[\nu N_{i,y}N_{j,x} + \tfrac{(1-\nu)}{2}N_{i,x}N_{j,y}\right] v_{(0,idf)}v_{(1,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf+3)} &= -\tfrac{Eh^3}{12(1-\nu^2)}\left[\nu N_{i,x}N_{j,y} + \tfrac{(1-\nu)}{2}N_{i,y}N_{j,x}\right] v_{(1,idf)}v_{(0,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf+3)} &= \tfrac{Eh^3}{12(1-\nu^2)}\left[\tfrac{(1-\nu)}{2}N_{i,y}N_{j,y} + N_{i,x}N_{j,x} + \tfrac{6\alpha(1-\nu)}{h^2}N_iN_j\right] v_{(1,idf)}v_{(1,jdf)}w \\
k^{p_n}_{(6i+idf+3,6j+jdf+3)} &= 1
\end{aligned}
\tag{6.5}
$$

Observa-se que os termos $N_{i,x}$ e $N_{i,y}$ apresentados em (6.5) denotam: $\frac{\partial N_i}{\partial x}$ e $\frac{\partial N_i}{\partial y}$, respectivamente.

O vetor de cargas é representado pela associação da função de forma aos carregamentos distribuídos. Ressalta-se, no caso de placas, a existência somente de carregamentos normais ao plano da placa e momentos distribuídos nas direções x e y.

$$
\begin{aligned}
f^{p_n}_{(6i,0)} &= 0 \\
f^{p_n}_{(6i+1,0)} &= 0 \\
f^{p_n}_{(6i+2,0)} &= N_i p_z \\
f^{p_n}_{(6i+3,0)} &= N_i m_x \\
f^{p_n}_{(6i+4,0)} &= N_i m_y \\
f^{p_n}_{(6i+5,0)} &= 0
\end{aligned}
\tag{6.6}
$$

Os valores p_z, m_x e m_y correspondem a forças e momentos distribuídos por unidade de área.

6.2 Sistema de equações

Uma vez computada a contribuição de todos os elementos na matriz de rigidez e no vetor de cargas da estrutura, faz-se a contribuição das ações nodais diretamente aplicadas nos nós no vetor de cargas da estrutura.

Assim que os dados dos nós da malha são introduzidos no ambiente PZ, estes passam por um procedimento de renumeração dos nós denominado *Cutil-Mackee* e a matriz de rigidez, após as contribuições de rigidez do elementos do problema, é então armazenada em banda para posterior processamento.

Chega-se então a um sistema linear de equações da seguinte forma:

$$[K]\,[D] = [A]$$

onde:

$[K]$ de ordem $n \times n$ é a matriz de rigidez global da estrutura, sem as condições de contorno;

$[A]$ vetor de ordem $n \times 1$ das ações nodais;

$[D]$ vetor de ordem $n \times 1$ das incógnitas de deslocamentos.

6.3 Condições de contorno

6.3.1 Barra

Para a fixação da estrutura, foram implementadas condições de contorno naturais (*Newmman*) e essenciais (*Dirichlet*).

As condições de contorno essenciais são dadas pela imposição de um valor alto, por exemplo 10^{12}, na diagonal da matriz de rigidez e no vetor de cargas, para os graus de liberdade em que a condição de contorno está sendo aplicada. No vetor de cargas associam-se também coeficientes de força. Dessa maneira, para o nó n, a imposição dessa condição de contorno afetará a matriz de rigidez e vetor de cargas:

$$\begin{aligned}
f^{p_n}_{(6i,0)} &= 10^{12} F_1 N_i w \\
f^{p_n}_{(6i+1,0)} &= 10^{12} F_2 N_i w \\
f^{p_n}_{(6i+2,0)} &= 10^{12} F_3 N_i w \\
f^{p_n}_{(6i+3,0)} &= 10^{12} M_1 N_i w \\
f^{p_n}_{(6i+4,0)} &= 10^{12} M_2 N_i w \\
f^{p_n}_{(6i+5,0)} &= 10^{12} M_3 N_i w
\end{aligned}
\quad \text{e} \quad
\begin{aligned}
k^{b_n}_{(6i,6j)} &= 10^{12} N_i N_j w \\
k^{b_n}_{(6i+1,6j+1)} &= 10^{12} N_i N_j w \\
k^{b_n}_{(6i+2,6j+2)} &= 10^{12} N_i N_j w \\
k^{b_n}_{(6i+3,6j+3)} &= 10^{12} N_i N_j w \\
k^{b_n}_{(6i+4,6j+4)} &= 10^{12} N_i N_j w \\
k^{b_n}_{(6i+5,6j+5)} &= 10^{12} N_i N_j w
\end{aligned}
\tag{6.7}$$

As condições de contorno naturais são dadas pela contribuição de cargas (*forças e momentos*) pontuais, no vetor de cargas, nos graus de liberdade referentes à aplicação dessas condição de contorno. Portanto, dado um ponto n, sujeito aos carregamentos nodais existentes, o vetor de cargas fica expresso por:

$$\begin{aligned}
f^{b_n}_{(6i,0)} &= F_1 N_i w \\
f^{b_n}_{(6i+1,0)} &= F_2 N_i w \\
f^{b_n}_{(6i+2,0)} &= F_3 N_i w \\
f^{b_n}_{(6i+3,0)} &= M_1 N_i w \\
f^{b_n}_{(6i+4,0)} &= M_2 N_i w \\
f^{b_n}_{(6i+5,0)} &= M_3 N_i w
\end{aligned}
\tag{6.8}$$

6.3.2 Placa

Para a fixação da estrutura, considere inicialmente uma placa na qual coincide o sistema local com o sistema global. Foram implementadas condições de contorno naturais (*Newmman*) e essenciais (*Dirichlet*). As condições de contorno essenciais são dadas pela imposição de um valor alto, por

exemplo 10^{12}, na diagonal da matriz de rigidez para os graus de liberdade em que a condição de contorno está sendo aplicada. Dessa maneira, para o nó n a imposição dessa condição de contorno afetará a matriz de rigidez e vetor de cargas, como segue:

$$\begin{aligned}
f^{p_n}_{(6i,0)} &= 0 \\
f^{p_n}_{(6i+1,0)} &= 0 \\
f^{p_n}_{(6i+2,0)} &= 10^{12} F_3 N_i w \\
f^{p_n}_{(6i+3,0)} &= 10^{12} M_1 N_i w \\
f^{p_n}_{(6i+4,0)} &= 10^{12} M_2 N_i w \\
f^{p_n}_{(6i+5,0)} &= 0
\end{aligned}
\quad \text{e} \quad
\begin{aligned}
k^{p_n}_{(6i,6j)} &= 10^{12} N_i N_j w \\
k^{p_n}_{(6i+1,6j+1)} &= 10^{12} N_i N_j w \\
k^{p_n}_{(6i+2,6j+2)} &= 10^{12} N_i N_j w \\
k^{p_n}_{(6i+3,6j+3)} &= 10^{12} N_i N_j w \\
k^{p_n}_{(6i+4,6j+4)} &= 10^{12} N_i N_j w \\
k^{p_n}_{(6i+5,6j+5)} &= 10^{12} N_i N_j w
\end{aligned}
\tag{6.9}$$

As condições de contorno naturais são dadas pela contribuição de cargas (*forças e momentos*) pontuais, no vetor de cargas, nos graus de liberdade referente a aplicação dessas condição de contorno. Portanto, dado um ponto n, sujeito aos carregamentos nodais existentes, o vetor de cargas é definido por:

$$\begin{aligned}
f^{p_n}_{(6i,0)} &= 0 \\
f^{p_n}_{(6i+1,0)} &= 0 \\
f^{p_n}_{(6i+2,0)} &= F_3 N_i w \\
f^{p_n}_{(6i+3,0)} &= M_1 N_i w \\
f^{p_n}_{(6i+4,0)} &= M_2 N_i w \\
f^{p_n}_{(6i+5,0)} &= 0
\end{aligned}
\tag{6.10}$$

Os vetores de eixos expressos em (6.5) representam o sistema de eixos associado ao ponto de integração do elemento de placa. Com isso, a transformação relativa à contribuição na rigidez do nó é calculado automaticamente em termos de contribuição no sistema global.

6.4 Obtenção das incógnitas

A matriz de rigidez $[K]$, já impostas as condições de contorno, é montada no PZ como uma matriz simétrica em banda, isto é, apenas a parte triangular superior da diagonal da largura da banda é armazenada.

A solução do sistema pode ser realizada por diversos métodos diretos, visto que o ambiente PZ está associado com a biblioteca *TMatrix* (Santana & Devloo[27]), voltada para o tratamento (armazenagem, escalonamento, solução) de matrizes. Dentre estes métodos diretos destacam-se: LDL^T, LU, LL^T.

O sistema $[K]\,[D] = [A]$ foi resolvido por um método direto, no caso o método de *Cholesky* obtendo-se assim os deslocamentos segundo as coordenadas globais.

6.5 Pós-processamento

A solução do sistema de equações fornece como resultados os deslocamentos nodais: três translações segundo os eixos globais x, y e z e três rotações por nó em torno dos eixos globais x, y e z.

Dentro do ambiente PZ foram implementadas rotinas para, a partir dos deslocamentos nodais, efetuar a etapa de pós-processamento calculando os esforços solicitantes em qualquer seção de elementos de barra e em qualquer seção de elementos de placas.

No trabalho optou-se apenas pela determinação de esforços solicitantes ao invés de cálculo de tensões, uma vez que o dimensionamento, via de regra, é feito a partir dos valores dos esforços solicitantes.

Para o cálculo dos esforços nos elementos foi utilizada a formulação indicada nos capítulos 4 e 5.

Para as barras foram implementadas rotinas que fornecem, no eixo da barra, nas seções início e final do elemento, os seguintes esforços solicitantes:

- Força normal F_x
- Força cortante V_{xy}
- Força cortante V_{xz}
- Momento fletor M_y

- Momento fletor M_z
- Momento torçor M_x

Para as placas foram implementadas rotinas que fornecem, no plano médio da placa, ao longo das faces dos elementos, os seguintes esforços solicitantes:

- Força cortante no plano xz V_{xz}
- Força cortante no plano yz V_{yz}
- Momento de flexão M_x
- Momento de flexão M_y
- Momento volvente M_{xy}

Foram implementadas no ambiente PZ rotinas de pós-processamento que preparam e escrevem arquivos contendo os resultados de deslocamentos e/ou de esforços solicitantes num padrão de saída formatado para servirem como arquivo de leitura para programas gráficos de visualização de resultados. As rotinas implementadas permitem preparar arquivos de dados para o programa de visualização Mview3D, desenvolvido na PUC do Rio de Janeiro e para o programa DX da IBM. No capítulo de exemplos alguns resultados foram preparados para serem mostrados através do programa DX.

Capítulo 7

Exemplos

O objetivo deste capítulo é apresentar alguns exemplos que foram resolvidos com a aplicação das formulações expostas nos capítulos anteriores, com a utilização das rotinas elaboradas no ambiente computacional PZ.

Os exemplos aqui mostrados são fictícios. As dimensões e características elásticas das estruturas, bem como os valores das cargas não representam fisicamente uma situação real de projeto de engenharia.

Os resultados obtidos com as rotinas elaboradas nessa dissertação foram comparados com simulações equivalentes realizadas com o software *Ansys*, versão 5.2. Os valores encontrados com o programa elaborado foram sempre muito próximos daqueles obtidos com o software *Ansys*.

Na apresentação dos exemplos serão feitos comentários quanto à ordem dos polinômios de interpolação e discretização das geometrias dos domínios e para a aproximação numérica dos problemas.

No exemplo 7.6 será apresentado as dimensões da matriz de rigidez global gerada pelo programa e a quantidade de memória RAM necessária para resolver tal sistema.

7.1 Exemplo 01 - Pórtico plano

Trata-se de um pórtico plano, contido no plano xz, com barras de seção constante $20cm \times 40cm$. A estrutura é submetida a uma ação uniformemente distribuída de valor $q = 1tf/m$ e a ações concentradas de valores $P = 5tf$, conforme indicado na Figura 7.1.

Os pilares e vigas têm $3m$ de comprimento e cada um deles foi discretizado

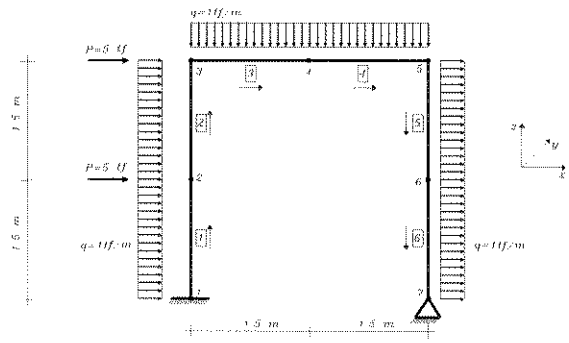

Figura 7.1: Pórtico Plano.

em 2 elementos unidimensionais de barra de comprimento $1,5m$. Na Figura 7.1 está ilustrado a numeração dos nós e dos elementos. Adotou-se módulo de elasticidade longitudinal $E = 2100000\ tf/m^2$. O apoio esquerdo é engastado e o apoio direito é articulado.

Ressalta-se que para a aproximação numérica do problema utilizaram-se polinômios de quarta ordem.

O valores de deslocamentos encontrados estão mostrados na Tabela 7.1, onde também estão indicados os valores correspondentes calculados com o programa *Ansys*.

Tabela 7.1: Deslocamentos/Rotações

	u_x		u_z		θ_y	
Nó	Tese	Ansys	Tese	Ansys	Tese	Ansys
1	0	0	0	0	0	0
2	0.00558	0.00543	0.00002	0.00002	-0.00550	-0.00561
3	0.01232	0.01228	0.00005	0.00005	-0.00263	-0.00267
4	0.012315	0.012307	-0.00084	-0.00081	0.000747	0.000751
5	0.01230	0.01218	-0.000113	-0.000108	-0.000696	-0.000703
6	0.008259	0.008295	-0.000056	-0.000049	-0.004466	-0.004501
7	0	0	0	0	-0.006058	-0.006097

Pode-se observar uma ligeira diferença de resultados entre os deslocamentos calculados pelo programa PZ e pelo programa *Ansys*. Isto se deve ao fato de que as funções de forma não sejam as mesmas e pela própria diferença entre as hipóteses das formulações adotadas (no *Ansys* não foi adotada a hipótese de Timoshenko).

A componente horizontal do deslocamento de translação do pórtico analisado está representado graficamente na Figura 7.2.

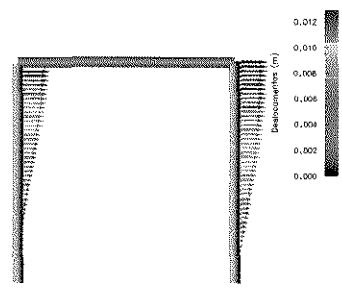

Figura 7.2: Deslocamento u_x no pórtico plano.

Para o pórtico ilustrado na Figura 7.1, o diagrama de distribuição do momento fletor no plano xz está representado graficamente na Figura 7.3.

A convenção utilizada foi representar o momento do lado das fibras tracionadas. Utilizou-se o programa de visualização *DX*. Ao lado do diagrama é apresentada uma barra de cores associada a valores dos esforços.

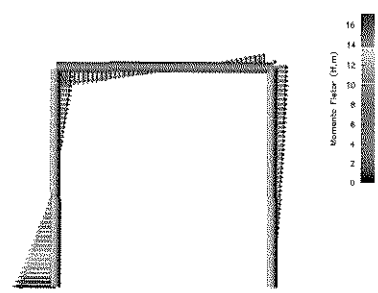

Figura 7.3: Representação do momento fletor M_y no pórtico.

A força cortante ao longo das barras está expressa graficamente na Figura 7.4. A barra de cor representa a mudança de intensidade da força cortante ao longo das barras. O diagrama de força cortante tem sinal. Adota-se a convenção de que força cortante positiva é a que percorre a seção transversal no sentido horário.

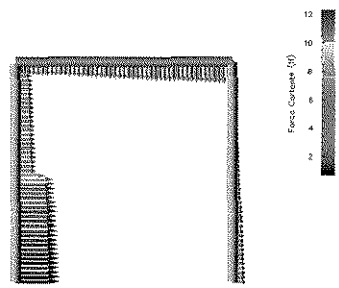

Figura 7.4: Força cortante solicitante no pórtico.

Os esforços normais para o carregamento indicado são constantes ao longo das barras como pode ser observado na Figura 7.5. Os valores positivos são forças normais de tração.

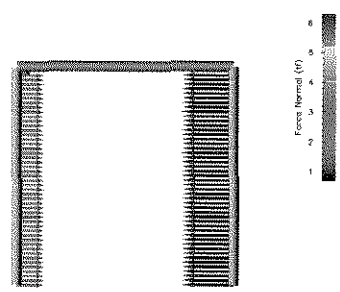

Figura 7.5: Representação do esforço normal no pórtico.

No diagrama essas forças estão representadas pelo sentido das setas coincidindo com o sistema do eixo global positivo, caso contrário são negativas.

Os valores dos esforços solicitantes calculados nas extremidades de cada elemento estão mostrados na Tabela 7.2. Os momentos fletores estão indicados na convenção de Grinter. Forças normais positivas são de tração e forças cortantes positivas são aquelas que percorrem a seção no sentido horário.

Tabela 7.2: Esforços solicitantes de forças normais, cortantes e momentos fletores

		Força Normal (tf)		*Força Cortante (tf)*		*Momento Fletor (tf.m)*	
Barra	Nó	Tese	Ansys	Tese	Ansys	Tese	Ansys
1-2	1	3.31	3.33	12.33	12.34	17.09	17.06
1-2	2	3.31	3.33	10.83	10.84	0.28	0.29
2-3	2	3.31	3.33	5.83	5.83	0.28	0.29
2-3	3	3.31	3.33	4.33	4.34	7.90	7.91
3-4	3	-0.65	-0.67	-3.30	-3.30	7.90	7.91
3-4	4	-0.65	-0.67	-4.80	-4.79	1.82	1.83
4-5	4	-0.65	-0.67	-4.80	-4.79	1.82	1.83
4-5	5	-0.65	-0.67	-6.30	-6.31	-6.51	-6.51
5-6	5	-6.30	-6.32	0.67	0.67	-6.51	-6.51
5-6	6	-6.30	-6.32	2.17	2.18	-4.38	-4.39
6-7	6	-6.30	-6.32	2.17	2.17	-4.38	-4.39
6-7	7	-6.30	-6.32	3.67	3.67	0	0

7.2 Exemplo 02 - Pórtico espacial

Neste exemplo foi feita uma análise de um pórtico espacial, onde atuam carragementos concentrados em alguns nós. A Figura 7.6 mostra as cargas nodais e condições de contorno ao qual a estrutura está submetida, e ilustra a numeração dos nós e elementos. As vigas e pilares do pórtico possuem seção constante $15cm \times 15cm$ e têm $3m$ de comprimento. Cada viga e pilar foi discretizado em 2 elementos unidimensionais de barra. Foram adotadas as seguintes características elásticas: módulo de elasticidade longitudinal $E =$

$2100000tf/m^2$ e módulo de elasticidade transversal $G = 807692.31tf/m^2$ (coeficiente de Poisson $\nu = 0,3$).

Para a aproximação numérica do problema utilizaram-se polinômios de quarta ordem.

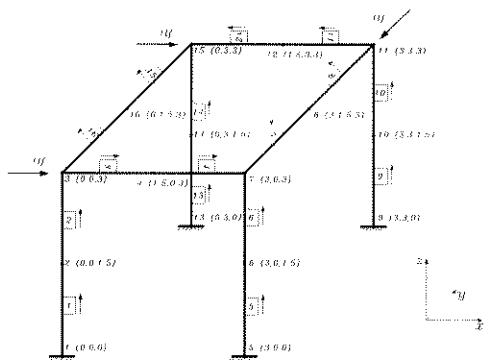

Figura 7.6: Pórtico espacial.

Os resultados de deslocamentos de translação obtidos na análise da estrutura para os nós 7 e 12 são apresentados na Tabela 7.3.

Tabela 7.3: Deslocamentos

	u_x		u_y		u_z	
Nó	Tese	Ansys	Tese	Ansys	Tese	Ansys
7	0.01573	0.01561	-0.01401	-0.01391	-0.00004	-0.00004
12	0.02084	0.02069	-0.00915	-0.00908	0.000009	0.000009

As rotações nestes mesmos nós são mostrados na Tabela 7.4.

Nas Tabelas 7.5 e 7.6 estão apresentados os valores das reações de apoio obtidas para o pórtico espacial do exemplo 02.

Tabela 7.4: Rotações

	θ_x		θ_y		θ_z	
Nó	Tese	Ansys	Tese	Ansys	Tese	Ansys
7	0.00267	0.00265	0.00322	0.00320	-0.00232	-0.00230
12	0.00182	0.00182	-0.00198	-0.00201	-0.00371	-0.00369

Tabela 7.5: Forças internas

	R_x		R_y		R_z	
Nó	Tese	Ansys	Tese	Ansys	Tese	Ansys
1	0.426	0.426	-0.108	-0.108	0.261	0.261
5	0.426	0.425	-0.390	-0.390	-0.688	-0.689
9	0.573	0.573	-0.391	-0.391	-0.166	-0.166
13	0.573	0.574	-0.108	-0.108	0.594	0.595

Tabela 7.6: Momentos fletores

	Rm_x		Rm_y		Rm_z	
Nó	Tese	Ansys	Tese	Ansys	Tese	Ansys
1	0.192	0.192	0.735	0.734	-0.052	-0.052
5	0.665	0.664	0.734	0.733	-0.052	-0.052
9	0.666	0.666	0.981	0.980	-0.052	-0.052
13	0.192	0.192	0.981	0.981	-0.052	-0.052

7.3 Exemplo 03 - Viga curva de eixo parabólico

Trata-se de uma viga curva, de seção constante, contida no plano xy, submetida a uma ação uniformemente distribuída de valor $p = 1kN/m$. A viga curva tem vão de $5m$ e a flecha no meio do vão vale $3,125m$. Seu eixo é parabólico e as ordenadas são dadas pela equação abaixo:

$$y(x) = 2.5x - 0.5x^2$$

A viga curva foi discretizada em 1 e 4 elementos. A Figura 7.7 ilustra a intensidade do momento fletor M_z nas seções da viga curva, calculado através do programa, para uma discretização em quatro elementos quadráticos. As cores estão associadas com aquelas da barra de cores indicadas ao lado.

Os resultados dos valores exatos do momento fletor foram comparados com os valores obtidos com o programa para uma discretização utilizando-se interpolação quadrática lagrangeana da geometria e polinômios de sexto grau para a aproximação dos deslocamentos.

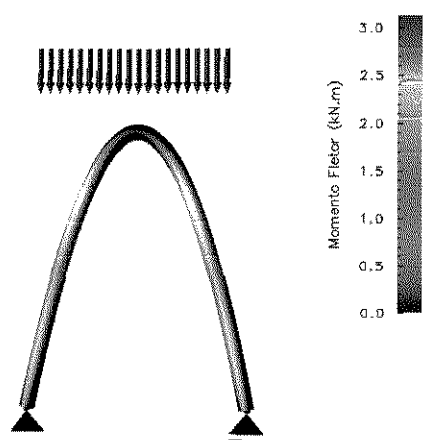

Figura 7.7: Representação gráfica da distribuição da intensidade do momento fletor ao longo da geometria do arco parabólico.

Na Figura 7.8 ilustra-se o êrro porcentual entre a solução exata e as soluções aproximadas obtidas numericamente no ambiente PZ, para as discretizações e interpolações indicadas.

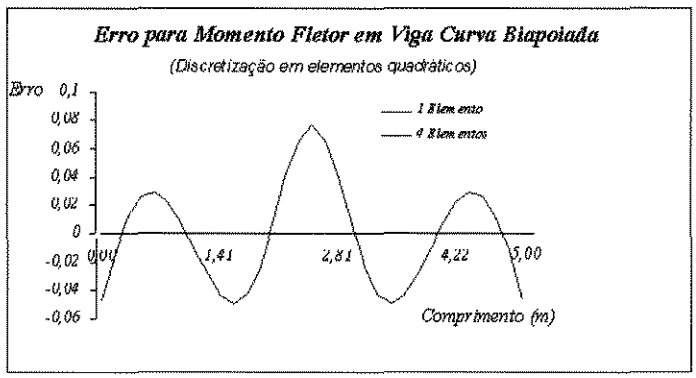

Figura 7.8: Êrro do momento fletor para 1 e 4 elementos quadráticos ao longo do arco.

7.4 Exemplo 04 - Anel comprimido

Neste exemplo foi analisado um anel sujeito a carregamento distribuído de pressão $P = 1kN/m$ aplicado no plano do anel, conforme a Figura 7.9. Os resultados exatos de esforços nesta estrutura são momento fletor e força cortante nulos e força normal de compressão constante de valor numérico igual ao valor do raio.

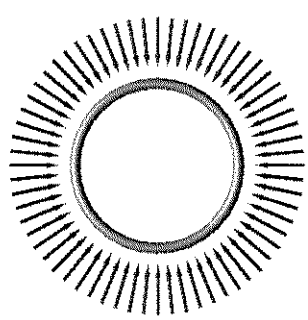

Figura 7.9: Anel sob carregamento de pressão uniforme $P = 1kN/m$.

Fazendo várias discretizações do anel com elementos em coordenadas cilíndricas, os valores obtidos apresentaram resultados com erros bastante pequenos em relação à solução exata, já a partir de dois elementos. Com elementos quadráticos a convergência para a solução exata foi mais lenta.

Na Figura 7.10 é mostrado o valor aproximado do momento fletor obtido utilizando-se 8 elementos quadráticos ao longo do eixo de um anel de raio $5m$, seção constante, submetido a uma ação de pressão uniforme $P = 1kN/m$.

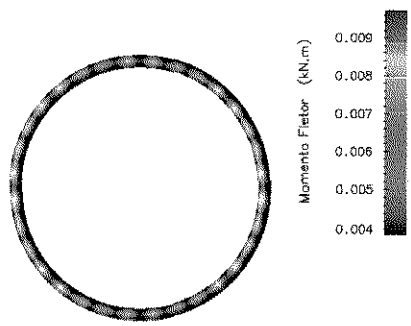

Figura 7.10: Representação da distribuição do momento fletor ao longo do anel, para 8 elementos quadráticos.

Na Figura 7.11 são mostrados os valores aproximados obtidos para o esforço normal utilizando 8 elementos ao longo do eixo do anel.

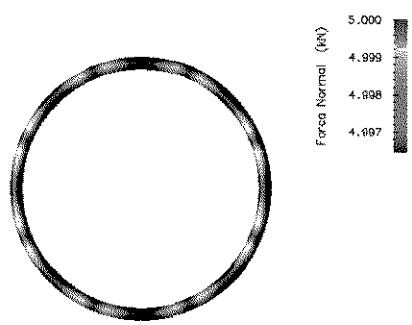

Figura 7.11: Representação da distribuição do esforço normal ao longo do anel, para 8 elementos quadráticos.

7.5 Exemplo 05 - Mola comprimida

Neste exemplo foi analisada uma mola espiral, conforme a Figura 7.12, com dois braços rígidos acoplados nas extremidades, submetida à ação de 2 forças concentradas de valores $P = 1kN$ aplicadas nas extremidades livres das hastes rígidas. As espiras têm raio de $5m$. O valor exato do momento torçor ao longo das espiras é $M_t = 5kN.m$.

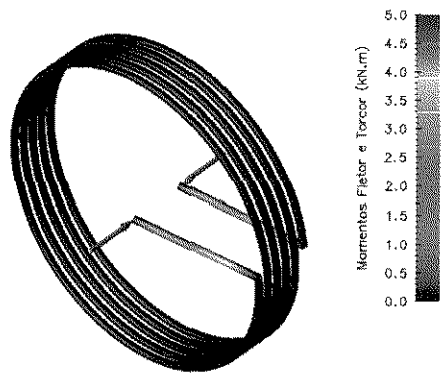

Figura 7.12: Mola com carga pontual aplicada nas extremidades das hastes.

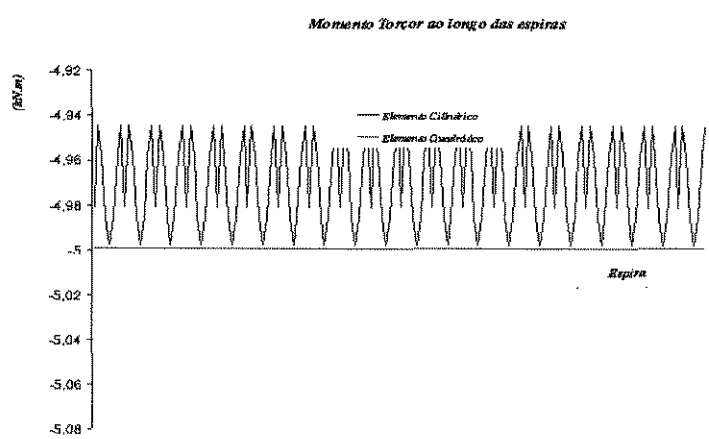

Figura 7.13: Momento torçor ao longo de uma espira retificada da mola.

O objetivo foi fazer uma análise comparativa entre resultados obtidos com discretização utilizando elementos quadráticos e discretização com elementos cilíndricos.

Com apenas um elemento cilíndrico foi possível descrever a geometria de uma espira. Para a malha computacional, entretanto, foi necessário o uso de polinômios interpoladores de alta ordem.

Na Figura 7.13 ilustra-se a diferença de soluções para o momento torçor ao longo das espiras, para discretizações da parte curva em 20 elementos geométricos cilíndricos e 20 elementos quadráticos.

7.6 Exemplo 06 - Placa engastada nos quatro lados

Trata-se de uma placa quadrada engastada nos quatro lados, contida no plano xy. A placa foi discretizada com elementos retangulares de 4 nós, com aproximação linear lagrangeana para a geometria. Foram adotados 8 elementos em cada direção. O carregamento adotado é uma carga uniformemente distribuída $q = 1tf/m$.

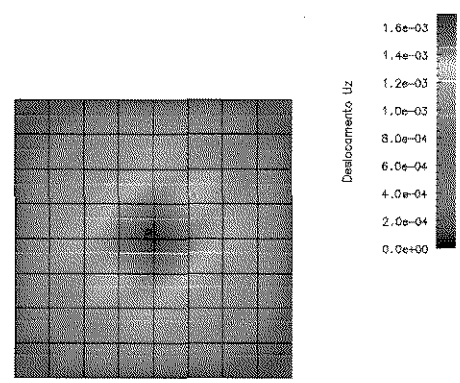

Figura 7.14: Placa quadrada engastada nos quatro lados, sujeito a carregamento distribuido $q = 1tf/m$.

A matriz de rigidez do elemento, utilizando polinômios de aproximação

de ordem quatro, possui dimensão 28 × 28. Dessa forma, a matriz de rigidez global possui dimensão 175 × 175. Como a matriz é simétrica, armazena-se somente sua triangular superior ou inferior, sendo assim, essa matriz é composta por 15.400 elementos do tipo *float* que ocupam $120,32Kb$ de memória RAM.

Para a solução aceitável das grandezas físicas adotou-se polinômios interpoladores de quinta ordem. Na Figura 7.14 mostra-se a flecha (deslocamento em z) ao longo da superfície. A placa possui $4m \times 4m$ e espessura constante de $10cm$. O módulo de elasticidade longitudinal adotado foi $E = 2100000tf/m^2$. O coeficiente de Poisson foi tomado igual a $\nu = 0.30$.

O valor teórico da flecha máxima que ocorre no meio da placa é dado por (7.1):

$$f_{central} = 0.000126\frac{qL^4}{D} \tag{7.1}$$

onde D está associado às constantes elásticas e geométricas da placa conforme (7.2):

$$D = \frac{Eh^3}{12\left(1-\nu^2\right)} \tag{7.2}$$

Para o exemplo estudado o valor exato da flecha máxima $0.0016773m$. O valor encontrado pelo programa da dissertação foi $f_{central} = 0.0016955m$.

7.7 Exemplo 07 - Casca com uma curvatura

Neste exemplo foi analisada uma casca parabólica engastada num único lado e com os demais lados com todos os deslocamentos livres.

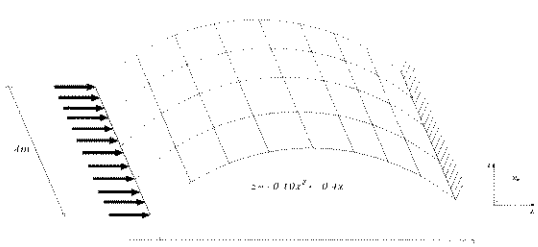

Figura 7.15: Casca parabólica com uma curvatura.

O carregamento considerado é uma ação uniformemente distribuída aplicada ao longo do lado oposto ao lado engastado, conforme mostrado na Figura 7.15. A casca analisada tem dimensões $4m \times 4m$, com espessura de $1cm$.

O módulo de elasticidade longitudinal adotado foi $E = 2100000tf/m^2$ e o coeficiente de Poisson adotado foi $\nu = 0.30$.

A formulação de placa com teoria de Reissner-Mindlin é apropriada para o tratamento de placas semi-espessas. No caso de sua utilização para a análise de placas delgadas pode aparecer um problema numérico conhecido com o nome *"shear-locking"*. No exemplo em discussão adotou-se uma espessura de apenas $1cm$ para tentar verificar o aparecimento do fenômeno.

Observou-se que para os resultados de esforços de flexão se mostrassem em equilíbrio foi necessária a utilização de polinômios de interpolação de quarta ordem. Para os esforços de membrana, entretanto, foi necessário aumentar a ordem de interpolação polinomial para no mínimo cinco.

Utilizaram-se polinômios de interpolação quadráticos lagrangeanos para discretização da geometria, com elementos retangulares com 9 nós.

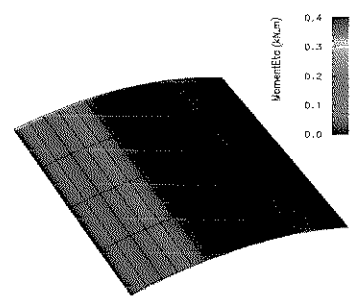

Figura 7.16: Casca parabólica.

7.8 Exemplo 08 - Edifício tridimensional

Neste exemplo foi analisado um edifício de 4 andares formado pela associação de elementos de barras e elementos de placas. Cada pavimento tipo é formado pela associação de 16 lajes com vigas de borda. O edifício possui 26 prumadas de pilares. As vigas e pilares foram modelados como elementos de

barra e as lajes como elementos de placa, conforme a formulação apresentada nos capítulos anteriores.

O carregamento na estrutura é constituído apenas por ações verticais uniformemente distribuídas aplicadas em todas as lajes, de valor $q = 1\,tf/m^2$. As vigas e pilares possuem dimensões padronizadas de $15cm \times 15cm$ e $3m$ de comprimento. Os elementos de placa possuem uma espessura de $7cm$. O módulo de elasticidade longitudinal foi adotado igual para toda a estrutura, valendo $E = 2100000\ tf/m^2$. O coeficiente de Poisson para as lajes foi adotado igual a $\nu = 0.30$.

A discretização da geometria dos elementos de barra foi efetuada por polinômios lagrangeanos lineares, o mesmo acontecendo com os elementos retangulares de placa.

Na Figura 7.17 está mostrado um esboço da estrutura deformada.

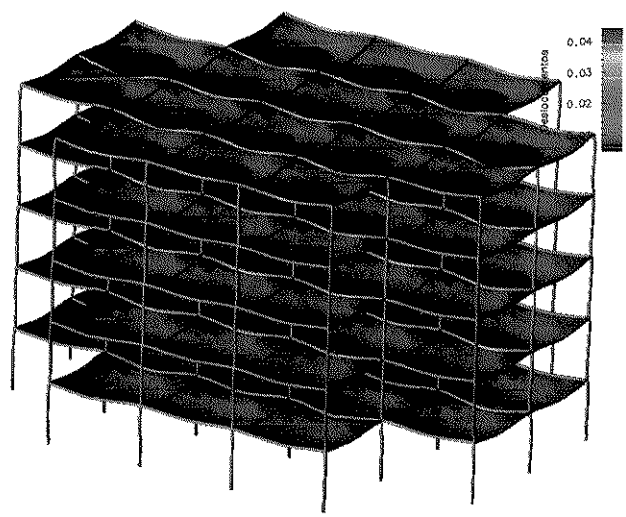

Figura 7.17: Edifício de quatro andares.

O problema foi analisado com o programa PZ e também foi simulado com o programa *Ansys*. Os resultados foram muito próximos, apesar do fato que foram usados elementos de placa de *Reissner-Mindlin* no programa PZ e elementos de placa de *Kirchoff* no programa *Ansys*. Na Figura 7.18 ilustra-se em detalhe a deformada dos elementos de barra e placa associados. Para efeito de comparação os deslocamentos verticais do ponto médio das lajes centrais do primeiro piso são mostrados na Tabela 7.7 e as reações de

apoio calculadas na base do pilar localizado na prumada frontal esquerda são mostradas na Tabela 7.8.

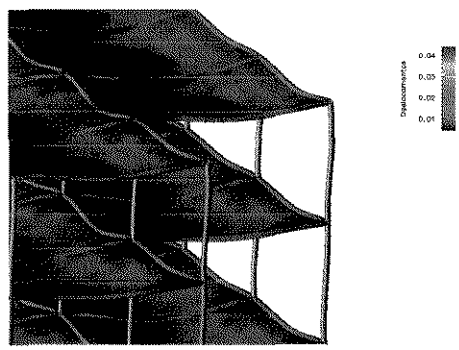

Figura 7.18: Detalhe da deformada associada dos elementos de viga e placa.

Tabela 7.7: Flechas nos pontos médios das lajes centrais do piso 1

Laje	*Flecha (m)*	
	Tese	Ansys
1	-0.03283	-0.03275
2	-0.03283	-0.03275
3	-0.03168	-0.03155
4	-0.03168	-0.03155
5	-0.03664	-0.03659
6	-0.03664	-0.03659

Tabela 7.8: Reações de Apoio na base do pilar, prumada frontal esquerda

Esforços	*x*		*y*		*z*	
	Tese	Ansys	Tese	Ansys	Tese	Ansys
Força	0.54099	0.54078	0.52903	0.52996	28.09330	28.08523
Momento	-0.53087	-0.53070	0.54167	0.54111	0.00000	0.00000

Capítulo 8

Conclusões

O objetivo deste capítulo é apresentar algumas conclusões e alguns comentários sobre os resultados obtidos na presente dissertação, bem como indicar algumas sugestões para continuidade do trabalho.

Tratou-se da implementação de um programa de computador para calcular esforços em edifícios simulados pelo acoplamento de elementos de barras (vigas e pilares) com elementos de placas. Foi utilizado uma plataforma de desenvolvimento de programas de elementos finitos denominada PZ, escrita em linguagem de programação orientada à objetos C++, a partir da qual foram implementadas as rotinas necessárias para o tratamento do problema do cálculo de esforços no edifício.

As duas grandes metas da dissertação foram as seguintes:

- a primeira foi o esforço realizado no desenvolvimento de formulações variacionais de elementos finitos de barras e placas. Buscou-se sistematizar os passos da aplicação do Princípio dos Trabalhos Virtuais. Com isso obteve-se a formulação variacional do problema, sendo que as variações dos deslocamentos foram utilizadas como funções peso nos cálculos de elementos finitos. Deu-se ênfase em resolvê-las através do Princípio dos Trabalhos Virtuais, por esta ser uma técnica bastante empregada e conhecida em Engenharia.

  Foram incorporados à formulação de placas os esforços de membrana, o que permitiu uma extensão da formulação para o tratamento de cascas.

- a segunda meta referiu-se a implementação computacional dessas formulações na plataforma PZ. Duas classes foram a base dessa implemen-

tação: a classes *TTimoshenko* e a classe *TPlateReissner*, ambas derivadas da classe abstrata *TMaterial* uma vez que o ambiente PZ trata as equações diferenciais de um problema como materiais. As classes trataram respectivamente do problema de elementos finitos de barras com as hipóteses de *Timoshenko* e elementos de placas com hipóteses de *Reissner-Mindlin*. Na implementação cuidou-se também de incorporar a formulação dos esforços de membrana, permitindo uma análise aproximada de cascas.

Além dessas classes inúmeras outras implementações foram necessárias. Algumas delas foram:

- desenvolvimento e implementação computacional do cálculo do jacobiano bidimensional na classe *TGeoElQ2d*;
- Criação da classe *TAnsys2pz*, para pré-processamento, viabilizando o uso da interface gráfica do programa *Ansys* para a criação de arquivos de dados, os quais são lidos pela classe *TAnsys2pz* no ambiente PZ;
- cálculos de pós-processamento, com a determinação de esforços solicitantes nos elementos de barras e de placas;
- criação de arquivos de resultados nos padrões interpretados pelos softwares DX e MView3D de visualização.

Como o ambiente computacional PZ é totalmente orientado a objetos, foi possível utilizar as suas classes básicas e criar classes derivadas para implementação das formulações propostas. Esse pressuposto, em tese, deveria dar uma grande produtividade, a qual, na realidade, não foi de todo alcançada. Ocorreram algumas dificuldades de implementação do código pela inexperiência do autor no uso do PZ. A extensão e complexidade do código PZ e a pouca documentação ainda existente sobre tal ambiente foram fatores que contribuiram para atraso na implementação proposta.

Os exemplos mostrados no capítulo 7 simularam algumas situações e algumas modelagens onde o programa computacional escrito foi utilizado. As comparações de resultados foram feitas para tentar dar credibilidade ao código que foi aqui implementado, dando ensejo para que futuras ampliações sejam encorajadas.

Pode-se apresentar algumas considerações e conclusões. É importante ressaltar que as conclusões apresentadas abaixo foram baseadas apenas nas

observações das respostas encontradas para os inúmeros exemplos analisados na fase de testes. As mais importantes são:

- o elemento unidimensional de barra, com 6 deslocamentos por nó utilizado para a modelagem de vigas e pilares, ou barras com qualquer posicionamento no espaço tridimensional, mostrou resultados consistentes com os conseguidos na análise de pórticos planos e de pórticos tridimensionais;

- este elemento de barra, formulado com a hipótese da teoria de viga de *Timoshenko*, mostrou ser aplicável sem restrições a qualquer relação entre altura da seção e comprimento da barra. Exemplos estudados, mas não mostrados no capítulo de exemplos, confirmam esta conclusão;

- ressalta-se também que pela não consideração do empenamento, o valor da constante de torção J_0 na formulação de barra é superestimado para a rigidez a torção;

- na formulação de elementos de barra foi associado um sistema de eixos aos pontos de integração do elemento quadrático e cilíndrico. Deste modo torna-se possível calcular os deslocamentos e rotações em função da orientação dos eixos naquele ponto. Para a análise dos elementos curvos isto representa uma generalização da geometria da estrutura; dessa forma, análises de estruturas mais complexas podem ser feitas usando essa técnica. Embora não seja possível garantir o ineditismo desse procedimento, o mesmo não foi encontrado na literatura;

- o elemento de barra quadrático mostrou também muita flexibilidade no trato de geometrias curvas, como exemplificado através da mola, anel e arco. Para se obter resultados satisfatórios no elemento de barra curva, foi necessário o uso de funções de interpolação de alta ordem;

- a formulação de placa de *Reissener-Mindlin* também demonstrou ser consistente em face dos resultados alcançados. Embora adequada para placas espessas, a formulação foi usada para placa delgada sem apresentar o fenômeno de travamento desde que usados polinômios de interpolação computacional de ordem maior que quatro;

- a formulação do elemento quadrático de placa de nove nós possibilitou estender a implementação para aproximações via elementos finitos não

planos, em função das variáveis de deslocamento e rotação no espaço cartesiano;

- a extensão da formulação de placa, com a consideração do efeito de membrana à formulação de placas de *Reissner-Mindlin*, demonstrou ser adequada. A formulação resulta do cálculo do equilíbrio no plano local da casca em cada ponto de sua superfície. Isto permite a formulação da matriz de rigidez do elemento diretamente no seu plano sem aplicação de eventuais rotações posteriores;

  A formulação proposta foi testada sobre cascas de geometria parabólica e observou-se que para elementos de casca como a configuração exemplificada, o fenômeno de travamento apenas é superado para ordem de interpolação polinomial igual ou superior a quatro;

- o acoplamento entre o elemento de barra e o elemento de placa ocorreu naturalmente, dada a preocupação inicial com relação à convenção de sinais adotada. Dessa forma, os elementos de barra e placa são lidos pela classe *TAnsys2pz* e inseridos no ambiente PZ, onde recebem o tratamento de elementos unidimensional e bidimensional, para posterior contribuição na matriz de rigidez e vetor de cargas da estrutura;

- os resultados obtidos na análise elástica estática do edifício tridimensional, comparados com os obtidos através do programa Ansys, mostraram-se confiáveis, tanto os deslocamentos, quanto os esforços;

- para os testes realizados a memória utilizada não foi um fator limitante. Os equipamentos computacionais do CENAPAD-Unicamp, colocados à disposição do autor permitiram que a matriz em banda da estrutura completa fosse montada inteira na memória RAM;

- não foram tomados maiores cuidados com a máxima eficiência das rotinas e não foi buscada, à exaustão, a diminuição do tempo de execução;

- foram feitos poucos testes quanto à convergência de soluções.

8.1 Sugestões para trabalhos futuros

Apresentam-se a seguir algumas sugestões e considerações para trabalhos futuros:

- criação de classes derivadas da classe básica *TMaterial*, que permitam avaliar os valores do módulo de elasticidade e do módulo tangente através de processos incrementais, obtendo-se a matriz de rigidez do objeto *TTimoshenko* ou derivado, nas quais os efeitos da não-linearidade física ou geométrica estejam presentes.

- incorporação da contribuição do efeito da alvenaria, comumente tratado apenas como vedação. Estas considerações podem não ser tão simples para implementação, devido a grande diferença existente entre as tensões de ruptura do material de vedação e dos elementos estruturais.

- modelagem dos pilares paredes e do núcleo de elevador de edifícios tridimensionais utilizando elemento de chapa. Embora não utilizado, as rotinas já implementadas pelo autor no ambiente PZ, para o tratamento de placa *TPlateReissner* incorpora também o efeito de membrana, e pode tratar adequadamente o efeito de chapa.

- a partir da formulação exposta e implementada para elementos de barra, será muito útil a consideração de excentricidades para acoplamento de vigas fora do plano médio das placas.

- utilizar a classe *TSuperEl* do ambiente PZ para análise das lajes dos edifícios considerando o superelemento proposto por Santana[27], com condensação estática da matriz de rigidez das lajes.

- desenvolver classes que permitam analisar o efeito de conexões semi-rígidas. Considerando o método aplicado por Soares[31], pode-se levar em consideração os efeitos provocados por conexões semi-rígidas implementando-se uma classe derivada da classe *TTimoshenko* que viabilize tais considerações na análise de pórticos planos e espaciais de barra.

- estender o pré-processamento representado pela classe *TAnsys2pz*, implementada nessa dissertação, para outros elementos da biblioteca de elementos do software *Ansys*.

Bibliografia

[1] ANSYS, Inc. ANSYS 5.3 Complete User's Manual Set. 1996.

[2] ARGYRIS, J. H. *An excursion into large rotations.* Computer Methods in Applied Mechanics and Engineering, 32, 85-155, 1982.

[3] ASSAN, A. E. *Método dos elementos finitos - primeiros passos.* Universidade Estadual de Campinas, Departamento de Construção Civil - Faculdade de Engenharia Civil, Mai. 1993, p.59.

[4] ASSAN, A. E. *Análise não-linear de arcos de concreto armado.* XV Congresso Ibero Latino-Americano sobre Métodos Computacionais para Engenharia, Belo Horizonte, MG, Novembro, 1994.

[5] BAREŠ, R. *Tablas para el calculo de placas y vigas paredes.* Editorial Gustavo Gili, Barcelona, 1970.

[6] BATHE, K. J. & DVORKIN, E. *A four-node plate bending element based on Mindlin/Reissner plate theory and a mixed interpolation.* International Journal for Numerical Methods in Engineering, 21, 367-383, 1985.

[7] BECKER, E. B., CAREY, G. F., ODEN, J. T. *Finite elements - An Introduction.* Prentice-Hall International, Inc. Englewood Cliffs, New Jersey, 1981.

[8] BUZAR, M. A. R. *Análise de placas com o método dos elementos finitos e de contorno na modelagem de um edifício.* Dissertação de Mestrado, Departamento de Engenharia Civil, UnB, Brasília, DF, 1996, 110p.

[9] CHEN, W. F. & HAN, D. J. *Plasticity for structural engineers.* Springer-Verlag, New York, 1988.

[10] COOK, R. D., MALKUS, D. S., PLESHA, M. E. *Concepts and applications of finite element analysis.* New York: John Wiley & Sons, 1989. 630 p.

[11] DEVLOO, P. R. B. *On the development of a finite element program based on the object oriented programming philosophy.* In: OONSKI'93 - Annual Object-Oriented Numerics Conference, 1st, 1993, SunRiver. Proceedings of the First Annual Object-Oriented Numerics Conference, Oregon: SIAM.

[12] DEVLOO, P. R. B. PZ: *An object oriented environment for scientific programming.* Computer Methods in Applied Mechanics and Engeneering. vol 150: 133-153, 1997.

[13] DUNAVANT, D. A. *High degree efficient symmetrical Gaussian quadrature rules for the triangle.* International Journal for Numerical Methods in Engineering, vol. 21, p.1129-1148.

[14] HINTON, E. & HUANG, H. C. *A family of quadrilateral Mindlin plate elements with substitute shear strain fields.* Computers & Structures, 23, 409-431, 1986.

[15] HUGHES, T. J. R. *The finite element method - linear static and dynamic finite element analysis.* Prentice-Hall International, Inc. Englewood Cliffs, New Jersey, 1987.

[16] LEA, D. et al. *User's guide to the GNU C++ library.* Cambridge: Free Software Foundation, 1992. 124 p.

[17] MENEZES, F. A. *Cálculo estático de vigas curvas constituídas de barras cirulares horizontais com seções abertas e paredes delgadas.* Tese de Doutorado, EESC-USP, Maio, 1990.

[18] MINDLIN, R. D. *Influence of rotatory inertia and shear in flexural motions of isotropic elastic plates.* Journal of Applied Mechanics, 18, 31-38, 1951.

[19] ODEN, J. T., REDDY, J. N. *An introduction to the mathematical theory of finite elements.* New York: John Wiley & Sons, 1976. 429 p.

[20] OÑATE, E., ZIENKIEWICZ, O. C., SUAREZ, B. & TAYLOR, R. L. *A general metodology for deriving shear constrained Reissner-Mindlin plate elements.* International Journal for Numerical Methods in Engineering, 33, 345-367, 1992.

[21] OÑATE, E. *Calculo de estructuras por el metodo de elementos finitos - analisis elástico lineal.* Centro Internacional de Métodos Numéricos en Ingenieria, Barcelona, 1992.

[22] PAPPAS, C. H. & MURRAY, W. H. *Turbo C++ completo e total.* São Paulo, Makron Books do Brasil Editora Ltda. 1991, p.771.

[23] PIMENTA, P. M. *Fundamentos de teorias das estruturas.* Notas de aula, Departamento de Engenharia de Estruturas e Fundações da Escola Politécnica da Universidade de São Paulo, 1993.

[24] PRZERMIENIECKI, J. S. *Theory of matrix structural analysis.* Dover Publications, Inc. New York, 1985.

[25] REDDY, J. N. *Energy and variational methods in applied mechanics.* John Wiley & Sons. Co., New York, 1984.

[26] REISSNER, E. *The effect of transverse shear deformation on the bending of elastic plates.* Journal of Applied Mechanics, 12, 69-76, 1945.

[27] SANTANA, M. L. M. & DEVLOO, P. R. B. *Object- oriented matrix classes.* In: Joint Conference of Italian Group of Computational Mechanics and Ibero-Latin American Association of Computational Methods in Engineering, 1, 1996, Pádua, Itália. p. 325-328.

[28] SANTOS DA SILVA, L. *Análise elástica e elastoplástica de placas submetidas a excitações estáticas e dinâmicas.* Dissertação de Mestrado, Departamento de Engenharia Civil, UnB, Brasília, DF, 1997, 104p.

[29] SCHOLZ, S. P. *Elements of an object-oriented FEM++ program in C++*, Computers & Structures. vol 43 (3): 517-529, Mai. 1992.

[30] SIMULOG.MODULEF. *Users guide*, no 1.180 p.1995. (modulef@simulog.fr)

[31] SOARES FILHO, M. *Análise elástica e elastoplástica de pórticos planos submetidos a excitações dinâmicas com consideração de conexões semi-rígidas.* Dissertação de Mestrado, Departamento de Engenharia Civil, UnB, Brasília, DF, 1997, 114p.

[32] SZILARD, R. *Theory of plates and shells.* McGraw-Hill, New York, 1974.

[33] TIMOSHENKO, S. S. P. *History of strength of materials.* McGraw-Hill, New York, 1953.

[34] TIMOSHENKO, S. & WOINOWSKY-KRIEGER, S. *Theory of plates and shells.* McGraw-Hill, New York, 1959.

[35] WATSON, A. S. & CHAN, S. H. *A prolog-based object oriented engeneering DBMS.* Computers & Structures. vol 40 (1): 11-21, Jun. 1991.

[36] WEISKAMP, K. & HEINY, L. & FLAMING, B. *Programação orientada para objeto com turbo C++.* São Paulo, Makron Books do Brasil Editora Ltda. 1993, p.474.

[37] WIENER, R. S. & PINSON, L. J. *An introduction to object oriented programming and C++.* Addison Wesley, 1988.

[38] ZIENKIEWICZ, O. C. *The finite element method.* Third edition. McGraw-Hill, New York, 1979.

[39] ZIENKIEWICZ, O. C. & TAYLOR, R. L. *The finite element method - solid and fluid mechanics, dynamics and non-linearity.* vol. 2, Fourth Edition, McGraw-Hill International Editions, Singapore, 1991.