UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Civil, Arquitetura e Urbanismo

FÁBIO HENRIQUE CAMPOS MAUAD

INTEGRAÇÃO DE MÉTODOS NUMÉRICOS AO AUTOCAD

CAMPINAS

2003

FÁBIO HENRIQUE CAMPOS MAUAD

INTEGRAÇÃO DE MÉTODOS NUMÉRICOS AO AUTOCAD

Dissertação de Mestrado apresentada a Faculdade de Engenharia Civil, Arquitetura e Urbanismo da Unicamp, para obtenção do título de Mestre em Engenharia Civil, na área de Estruturas.

Orientador: Prof. Dr. Philippe Remy Bernard Devloo

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA PELO ALUNO FÁBIO HENRIQUE CAMPOS MAUAD E ORIENTADO PELO PROF. DR. PHILIPPE REMY BERNAD DEVLOO

_____

CAMPINAS

2003

FICHA CATALOGRÁFICA ELABORADA PELA

BIBLIOTECA DA ÁREA DE ENGENHARIA – BAE – UNICAMP

Mauad, Fábio Henrique Campos
M44i Integração de métodos numéricos ao AutoCAD / Fábio Henrique Campos Mauad--Campinas, SP: [s.n.], 2003.

Orientador: Philippe Remy Bernard Devloo.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Civil.

1. AutoCad (Programa de computador). 2. Programação orientada a objetos (Computação). 3. Projeto Estrutural. 4. Método dos elementos finitos. I. Devloo, Philippe Remy Bernard. II Universidade Estadual de Campinas. Faculdade de Engenharia Civil. III Título

UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Civil, Arquitetura e Urbanismo

INTEGRAÇÃO DE MÉTODOS NUMÉRICOS AO AUTOCAD

Fábio Henrique Campos Mauad

Dissertação de Mestrado aprovada pela Banca Examinadora, constituída por:

Prof. Dr. Philippe Remy Bernard Devloo

Presidente e Orientador / DES /FEC / UNICAMP

Prof. Dr. Marco Lúcio Bittencourt

DPM / FEM / UNICAMP

Prof. Dr. José Luiz Antunes de Oliveira e Sousa

DES /FEC / UNICAMP

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmica do aluno

Campinas, 28 de fevereiro de 2003

À minha esposa e companheira Juçara.

Agradecimentos

Ao Departamento de Estruturas da Faculdade de Engenharia Civil - UNICAMP pela infraestrutura.

À FINEP, CNPQ e PETROBRAS pelo financiamento da infraestrutura computacional.

À PETROBRAS pela bolsa de estudos.

Ao orientador e amigo Prof. DI. Philippe R. B. Devloo pela dedicação, apoio e pela disponibilidade.

Aos amigos e professores do departamento de estruturas, em especial o Prof. DI. Francisco Antônio Menezes, pelo compartilhamento dos conhecimentos e interesses no desenvolvimento do trabalho.

Ao amigo e professor Flávio de Oliveira Costa pelas constantes exposições e troca de informações, que servem de motivação na busca de uma Engenharia melhor em nosso país.

Aos amigos do LabMec Edmar Cesar Rylo, Cedric Bravo, Gustavo Camargo Longhin e Renato Gomes Damas, pelo apoio, companheirismo e pelas valiosas informações passadas, imprescindíveis ao desenvolvimento do meu trabalho.

Ao amigo e companheiro Erick Slis, com quem muito aprendi, não somente sobre o nosso trabalho, mas também sobre respeito, seriedade e companheirismo.

À meus Pais e Irmãs, pelo carinho, pela atenção e pela união que nos torna próximos, apesar da distância

Resumo

A utilização de ferramentas gráficas computacionais para a concepção e *desenvolvimento de um modelo estrutural, torna esta tarefa mais eficiente* e *permite ainda a rápida verificação de diferentes arranjos estruturais, buscando a solução mais adequada para* o *problema* em *estudo.*

O programa AutoCAD é o software mais utilizado para o desenvolvimento da representação gráfica de projetos de Engenharia e Arquitetura. A disponibilização de ferramentas que permitam aos usuários criar elementos gráficos com significado estrutural, mostra-se muito eficiente, na medida em que elimina erros de interpretação de desenho e de concepção do modelo estrutural, permitindo uma visualização rápida do trabalho executado e eliminando o retrabalho. Em outras palavras, permite que o lançamento de estrutura seja feito via AutoCad, sobre uma representação gráfica existente.

Neste trabalho procurou-se desenvolver estas ferramentas que, uma vez residentes no AutoCAD, permitirão a compatibilização das suas funções gráficas, com os módulos de análise estrutural do Ambiente de Elementos Finitos PZ[6], através da biblioteca ObjectARX Isto significa interpretar os dados das entidades estruturais implementadas, fornecendo as informações necessárias para os módulos do PZ e manipular os resultados de maneira a apresentá-los aos usuários de forma adequada.

Abstract

The design and development of a structural model based on a graphical envirionment improve efficiency and allow prompt verifications of different structural geometries, aiding the achievement of the most-adequate solution.

AutoCAD(R) is the most-used software for the development of graphical representations of Architectural and Engineering projects. The possibility of extending it, allowing the users to create graphical entities with structural properties, reduces the risks of interpretation mestakes while exchanging informations among design and strutural models. In order words, this extension allows the strutural design to be developed over an architectural representation through AutoCAD(R).

In this work, the author developed an extension tool linking AutoCAD's graphical tools with the analysis routines from the Finite Element Environment PZ through the ObjectARX libraries. This tool, once linked and embedded in AutoCAD(R), enables the strutural design using the AutoCAD(R) interface, the evaluation of it using PZ and the graphical post processing in AutoCAD(R).

Sumário

1 Introdução

O desenvolvimento do um projeto estrutural é uma tarefa que envolve três etapas que são o pré-processamento, o processamento e o dimensionamento e detalhamento. No pré-processamento, partindo-se de um desenho de arquitetura faz-se a determinação de um modelo estrutural, composto de barras e placas, determinam-se os carregamentos atuantes nos elementos e as condições de vinculação dos nós da estrutura. Na etapa de processamento, realiza-se a resolução do problema para a determinação dos esforços de momento e cortante nos elementos estruturais.

Na fase de dimensionamento e detalhamento, utilizam-se os resultados obtidos do cálculo para dimensionar os elementos. Nessa fase, caso se conclua que é necessário realizar alguma alteração ou correção no modelo de cálculo, deve-se retomar ao modelo inicial e refazer todo o processo novamente.

Para realizar essas tarefas existem diversos *softwares.* Com utilização desses é possível fazer esse trabalho de forma eficiente e rápida. Algumas dessas ferramentas computacionais são SAP2000, Nastran, Ansys, CypeCAD, Eberic, AltoQI e TQS.

Os programas NASTRAN e ANSYS são consagrados no mercado mundial. São apropriados para análises sofisticadas, para problemas de difícil equacionamento e nos quais é necessária uma confiabilidade elevada nos resultados. Permitem a realização de muitos tipos de problema e análises. Utilizam como método de resolução o método dos elementos finitos. Permitem a criação de muitos tipos de elementos e permitem uma modelagem tridimensional. Permitem a utilização de diferentes tipos de materiais. Permitem ao usuário observar os resultados em tela ou em arquivo texto. Apresentam um ambiente gráfico de trabalho com limitados recursos para entrada de dados e análise de resultados. Não fazem o dimensionamento dos elementos.

O programa SAP2000 apresenta as mesmas características dos anteriores além de realizar o dimensionamento de elementos de barra. Contém normas técnicas Americanas e Européias para dimensionamento. Contém um banco de perfis metálicos.

Nenhum desses três permite a confecção de desenhos dentro do ambiente de trabalho. Não apresentam uma integração com um ambiente de desenho, apenas permitem exportar arquivos no formato DXF.

Os programas CypeCAD, Eberic, AltoQI e TQS são específicos para problemas de engenharia civil.

O programa CypeCAD é subdividido em módulos, sendo cada um deles voltado para uma parte da estrutura. Existe um módulo para dimensionamento de fundação direta por sapatas, para estruturas de concreto armado e para estruturas metálicas, além de módulos complementares de gerenciamento. Apresenta um ambiente gráfico com limitados recursos para entrada de dados e análise de resultados. Realiza dimensionamento de elementos de concreto armado e metálicas. Gera desenhos com detalhamento dos elementos e ligações entre os elementos. Os desenhos gerados são bons, mas apresentam falhas. Por isto torna-se necessária a utilização de outros programas para finalizar esses desenhos.

Os programas Eberic, AltoQI e TQS também apresentam módulos para cálculo e dimensionamento de estruturas de concreto armado e metálica, mas não dimensionam fundações. Também apresentam deficiência com relação às ferramentas gráficas para entrada de dados e para confecção dos desenhos finais.

Pelo fato de existirem poucos programas que realizam todo o desenvolvimento do projeto de forma adequada, um projeto estrutural normalmente é feito utilizando-se mais de uma ferramenta. Com isto é preciso transportar as informações de um ambiente para o outro, o que pode introduzir erros. Além disso, caso seja observado algum erro em uma das etapas, ou seja, necessária alguma adequação nas etapas de pré-processamento ou de processamento, o trabalho realizado é todo perdido, pois não existe interligação entre as etapas.

Com a utilização de um único ambiente de trabalho onde se possa realizar todas as tarefas mencionadas, pode-se evitar o surgimento de muitos problemas. Além disto, simplifica-se a realização das alterações e pode-se diminuir o tempo gasto para desenvolver o projeto [10].

O objetivo desse trabalho é desenvolver uma ferramenta que permita aos profissionais das áreas de engenharia o desenvolvimento de todo o trabalho em um único ambiente gráfico, e a confecção de desenhos dentro dos padrões adequados.

O programa AutoCAD é um *software* consagrado mundialmente para o desenvolvimento de desenhos de projetos de Engenharia e Arquitetura, devido ao fato de possuir um conjunto completo de ferramentas e recursos que facilitam a execução dos trabalhos e lhes confere qualidade e organização. O AutoCAD permite a incorporação de novas funcionalidades escritas na linguagem *C++*. Dessa forma, é possível desenvolver aplicativos ou funções que permitam uma maior integração entre as diferentes etapas no desenvolvimento completo de um projeto [10].

A contribuição que este trabalho espera oferecer é incorporar ao programa AutoCAD um conjunto de rotinas para análise estrutural que possuam as funcionalidades gráficas do programa. Para isso, pretende-se duas ferramentas básicas para o desenvolvimento do trabalho que são o *Ambiente* PZ [6] o *ObjectARX* [10]**.**

Para o desenvolvimento dessas ferramentas, a *AutoDesk*[1] disponibiliza o ambienta orientado a objetos *ObjectARX* Desenvolvido na linguagem *C++,* esse ambiente apresenta um conjunto de bibliotecas de classes derivadas das classes do *AutoCAD* e por meio delas os usuários podem desenvolver novos comandos e entidades que são tratadas pelo programa como entidades nativas [13].

Para a realização de análise estrutural, utilizou-se o ambiente *PZ,* um ambiente para computação científica, estruturado para o desenvolvimento de aproximações de soluções para problemas de valor de contorno pelo método dos elementos finitos uma descrição do ambiente PZ é apresentada no capítulo 2.

Desenvolveu-se um *software* que trabalha dentro do programa *AutoCAD* e que contém um conjunto de ferramentas que, uma vez residentes no *AutoCAD,* permite a compatibilização das suas funções gráficas, com os módulos de análise estrutural do Ambiente de Elementos Finitos PZ, através da biblioteca ObjectARX. Isto significa interpretar os dados das entidades estruturais implementadas, fornecendo as informações necessárias para os módulos do PZ e manipular os resultados de maneira a apresentá-los aos usuários.

Nesse quadro, o usuário irá desenvolver um projeto estrutural completo dentro de um único ambiente de trabalho, evitando o surgimento de todos os problemas já relatados.

Uma descrição da biblioteca ObjecARX e da estruturação de um desenho do programa AutoCAD é apresentada no capítulo 3.

No capítulo 4 foi apresentada a teoria de *Timoshenko* para vigas e a aplicação do método dos elementos finitos para um elemento de barra [8]**.** A formulação foi desenvolvida pelo autor como aplicação da metodologia documentada no trabalho de *Menezes* e *Devloo [14],* baseada nos conceitos apresentados no apêndice C deste trabalho.

No capítulo 5 tem-se uma extensão da formulação apresentada no capítulo 4 para o caso de vigas curvas. Foi apresentada a estratégia de determinação do *Jacobiano* para cada ponto de integração do elemento.

No capítulo 6 apresentou-se a formulação utilizada para a implementação do elemento de placa. Para esse elemento utilizou-se a teoria de *Reissner* e *Midilin* [12] e a metodologia apresentada no trabalho de *Menezes&Devloo [14]*. Trabalhou-se com um

elemento em uma posição qualquer do espaço, em relação a um sistema global de coordenadas. Também foi descrita a implementação computacional do elemento de placa.

No capítulo 7 foi apresentada a descrição da implementação computacional e a metodologia utilizada para o desenvolvimento do trabalho. Foram descritas as classes implementadas e os elementos desenvolvidos. Além disto, foram descritas, de forma detalhada, as características dos elementos que constituem um projeto desenvolvido para o programa *AutoCAD.*

Finalmente, no capítulo 8 foram apresentados alguns exemplos de aplicação das ferramentas desenvolvidas.

Com o objetivo de fornecer ao leitor uma fonte de consulta para o desenvolvimento de estudos semelhantes, nos apêndices A, B, C e D, são apresentados os conceitos básicos utilizados para o desenvolvimento deste trabalho.

No apêndice A é apresentada uma introdução sobre o método dos elementos finitos com os seus conceitos fundamentais. Para motivar o desenvolvimento, aplicou-se o método a um problema unidimensional. Esta introdução baseia-se no livro de Oden, Becker e Cary[1].

Em seguida, no apêndice B é apresentada uma introdução sobre a linguagem de programação *C++,* que foi utilizada para desenvolver os códigos desse trabalho, com um breve histórico, o seu surgimento e desenvolvimento. Esta introdução baseia-se no livro de B. Straustrup [17]. Ainda no apêndice B, são apresentados os conceitos básicos sobre a técnica de Programação Orientada a Objetos e as suas principais características, com as definições de Classes, Objetos, Mensagens e Métodos.

Finalmente, tem-se no apêndice C uma descrição sobre os principais conceitos utilizados neste trabalho. Apresenta as hipóteses cinemáticas para o deslocamento de um ponto material. Em seguida pode-se observar o equacionamento da relação entre a tensão e a deformação um corpo constituído de material elástico linear. Esse estudo foi apresentado para os casos de tensão unidimensional, bidimensional e tridimensional, para materiais ortótropos e isotrópicos. Finalmente tem-se uma descrição do princípio dos trabalhos virtuais que encerra o capítulo [4].

1.1 Objetivos

Esse projeto tem o objetivo de desenvolver um software de Análise Estrutural, para estruturas planas ou espaciais, baseado na metodologia de orientação para objetos, utilizando o

Método dos Elementos Finitos e que tem como plataforma gráfica o software AutoCad 2000. Para isso, pretende-se criar um conjunto independente de comandos e entidades, derivadas das classes primitivas do AutoCAD.

Pretende-se tornar disponíveis as seguintes funcionalidades:

- Geração automática dos elementos de barra e elementos de placa;
- Determinação das características dos materiais constituintes da estrutura;
- Determinação das condições de vinculação e deslocabilidade dos nós da estrutura;
- Montagem da malha de Elementos Finitos;
- Realização do cálculo do sistema;
- A visualização e análise dos resultados obtidos por meio de gráficos e diagramas.

2 Ambiente PZ

A plataforma *PZ* é um ambiente de programação científica, desenvolvido na linguagem *C/C++,* destinado à resolução de problemas de engenharia que podem ser formulados como um sistema de equações diferenciais parciais. Especificamente, o ambiente *PZ* foi concebido para aplicações que utilizem o Método dos Elementos Finitos.

Segundo Devloo [6], as classes do ambiente *PZ* definem elementos inerentes de problemas de elementos finito s como malha, elemento, nó e material. Tais classes implementam métodos para o tratamento dos dados como, por exemplo: 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.

Diferentemente do que se apresenta usualmente, no ambiente *PZ* existe uma separação entre os elementos geométricos e os elementos computacionais[1]. Dessa forma, para cada problema há duas malhas, uma geométrica e uma computacional. A um conjunto de elementos e nós, denomina-se malha. Em uma malha geométrica há os elementos geométricos e nós geométricos. Já uma malha computacional contém elementos e nós computacionais.

O ambiente *PZ,* como na maioria dos ambientes de programação orientada 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 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] A atribuição de elementos geométricos e elementos computacionais será documentada a seguir.

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 2.1, ilustra a ordem de execução do algoritmo em que o ambiente PZ foi elaborado.

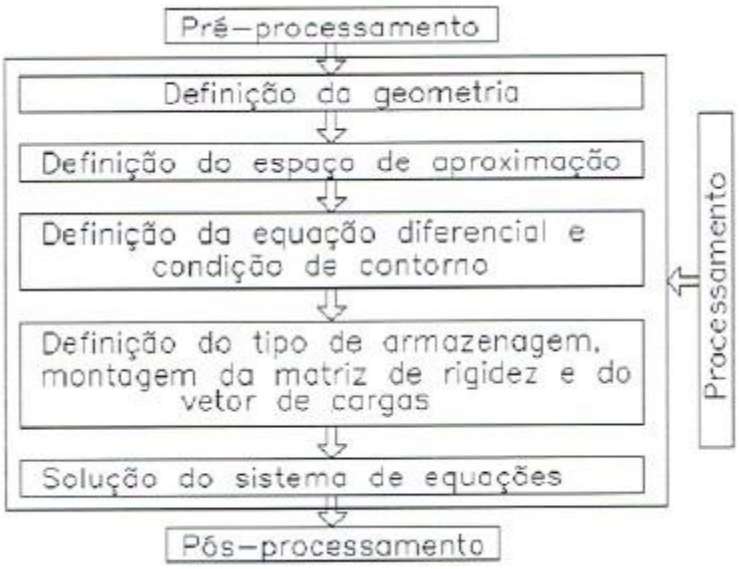

Figura 2.1: Exemplo de fluxograma de execução utilizando as classes do Ambiente PZ.

O PZ foi concebido para disponibilizar diversas análises. É possível discretizar um mesmo domínio por elementos retangulares e triangulares, dentre outros. 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, com um número livre de graus de liberdade por nó. No caso de malhas bidimensional, 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 utilizou-se seis graus de liberdade tanto para o elemento unidimensional de barra quanto para o elemento bidimensional de placa.

No ambiente PZ, encontram-se classes responsáveis pela geração de arquivos de entrada para programas de visualização científica, 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.

2.1 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:

- TPZGeoMesh: malha geométrica;
- TPZGeoNod: nó geométrico;
- TPZGeoEI: elemento geométrico;
- TPZCosys: sistema de coordenadas.

A definição desses parâmetros é realizada entre um mapeamento da configuração do domínio (deformada), e a configuração de um elemento mestre.

2.1.1 Classe TPZGeoMesh

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

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

2.1.2 Classe TPZGeoEI

A classe *TPZGeoEl* é 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, divisão do elemento em sub elementos e criação do elemento computacional a partir do elemento geométrico.

As classes *TPZGeoElPoint, TPZGeoEl1d, TPZGeoElQ2d* e *TPZGeoElT2d* implementam os elementos geométricos pontual, 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.

2.1.3 Classe TPZGeoNode

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

2.1.4 Classe TPZCosys

A classe *TPZCosys* define os métodos e trata dados referentes ao sistema de coordenadas. É uma classe básica e abstrata de onde são derivadas funções que tratam coordenadas cartesianas, cilíndricas e esféricas. Dessa forma, disponibilizam-se diferentes tipos de sistemas de coordenadas para os sistemas locais e globais.

2.1.5 Classe TPZGeoEIBC

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

Principais variáveis

- TPZGeoEI *fElement: Ponteiro para acessar um elemento geométrico.

- TPZGeoEI *fBCElement: Ponteiro para o elemento que define a geometria do contorno.

- int fSide: Número de identificação do lado em que está aplicada uma condição de contorno.

- int fld: Número de identificação da condição de contorno do elemento.

Principais funções

Essa classe possui três construtores e uma função print, para escrever para arquivo.

- TPZGeoElBC( )

- TPZGeoElBC(TPZGeoEI *el, int side, int id, TPZGeoMesh &mesh)

- TPZGeoElBC(TPZGeoEISide &elside, int id, TPZGeoMesh &mesh)

- void Print (ostream &out = cout)

2.2 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 o de *Rayleigh- Ritz* e o de *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, esse 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 do elemento geométrico associado para calcular a matriz de rigidez do elemento.

2.2.1 Classe TPZCompMesh

A classe *TPZCompMesh* é responsável pelo controle dos nós e elementos computaciopais, 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;
- método para 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.

2.2.2 Classe TPZCompEI

A classe *TPZCompEl* implementa o elemento computacional. Essa classe tem 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 dado pela aproximação.

2.2.3 Classe TPZCompElld

A classe *TPZCompElld* trata o elemento computacional unidimensional e é derivada da classe *TPZlnterpolatedElement,* que por sua vez é derivada da classe *TPZCompEl.*

2.2.4 Classe TPZCompEIT2d e Classe TPZComEIQ2d

As classes *TPZCompElT2d* e. *TPZCompElQ2d* implementam o comportamento de elementos computacionais bidimensionais triangulares e retangulares, respectivamente. São derivadas da classe *TPZlnterpolatedElement,* que por sua vez é derivada da classe *TPZCompEl.*

2.3 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 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 aproximadas das mencionadas equações diferenciais. A classe básica e abstrata que implementa essas características é a classe *TPZMaterial.* Nas classes derivadas de *TPZMaterial,* são manipulados os coeficientes dos materiais específicos que representam o comportamento de um dado material. Assim, com o auxílio da representação matricial, pode-se resolver essas formulações para tais coeficientes corresponde a solucionar um problema de análise computacional em engenharia.

2.3.1 Classe TPZMaterial

A classe básica *TPZMaterial* tem por função montar e calcular a contribuição de um elemento na matriz de rigidez e no vetor de cargas. Pressupondo que em um problema estejam envolvidos materiais de dimensões distintas, foram implementados métodos que verificam se o objeto pertence a uma classe derivada apropriada.

Para esse trabalho, visando analisar o problema do material de barra elástica linear pela hipótese de *Timoshenko,* utilizou-se a classe *TTimoshenko,* derivada de *TPZMaterial.*

Para a análise do problema do material de placa com a hipótese de *Reissner-Mindlin,* foi utilizada a classe *TPZMatPlaca2,* que é derivada da classe *TPZMaterial.*

2.4 Classes que definem a montagem e solução do sistema

Classe TPZAnalysis

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

3 ObjectARX

3.1 Introdução

As atuais versões do *software* AutoCAD são elaboradas em linguagem orientada a objetos. Desde a versão 14, a AutoDesk fornece uma possibilidade de estender as funcionalidades desse *software* no mesmo nível em que seus programadores elaboram as extensões. A tecnologia denominada ObjectARX é composta dos seguintes itens:

- Arquitetura aberta;
- Bibliotecas;
- Arquivos fonte;
- Documentação;

O *ObjectARX (AutoCAD Object Oriented Runtime Extension)* consiste em uma abertura da arquitetura do AutoCAD [11, Kramer]. As bibliotecas de ligação dinâmica *(DLL)* utilizadas pelo AutoCAD são também disponibilizadas para os programadores que se interessam em desenvolver aplicativos em *C++,* orientados a objetos. Para isso, as funcionalidades dessas DLL's são também forneci das pela *AutoDesk* sob a forma de bibliotecas estáticas e arquivos de cabeçalho contendo os protótipos de classes e funções.

Os aplicativos que se utilizam da estrutura interna de classes do *AutoCAD* podem ser compilados externamente ao *AutoCAD,* gerando bibliotecas dinâmicas de extensão. Às bibliotecas de conexão direta com o programa são atribuídas as extensões *.ARX* e *.DBX,* ao invés de *DLL.* É importante frisar que esses três tipos de bibliotecas não diferem quanto à forma de ligação, apenas na funcionalidade [13, McAley].

3.2 Orientação a Objetos na estrutura do ObjectARX

O *software AutoCAD* é organizado em uma hierarquia de classes, em que as classes se subdividem entre aquelas que implementam representação gráfica (entidades), aquelas

destinadas a reger o relacionamento entre as entidades (reatores, notificadores, transatores, dentre outros), as que fornecem a comunicação entre esses objetos e os usuários e com o AutoCAD. Dentro dessa organização é necessário definir o que é um desenho, o que é uma entidade e o que é um desenho visto como um banco de dados.

3.3 Desenho

Um desenho é uma forma de representar uma ideia ou uma concepção de algo. Uma das formas de desenho é a representação técnica que se utiliza em projetos de Engenharia e Arquitetura. A função de um desenho é passar informações de forma visual. O programa AutoCAD é uma ferramenta que permite criar e desenvolver representações bidimensionais de modelagens bidimensionais ou tridimensionais. Com o AutoCAD pode-se estender o conceito de desenho, pois não existem limites para construção, além de permitir acoplar aos elementos criados muito mais do que sua própria representação gráfica.

Nesse trabalho, quando se refere a um desenho, refere-se a desenho assistido por computador, construído utilizando-se o *software AutoCAD,* sendo o desenho estruturado como um banco de dados, composto de entidades que possuem representação gráfica.

Em um desenho do *AutoCAD,* pode-se encontrar dois espaços primários, onde residem os objetos que são o *PaperSpace* e o *ModelSpace.*

Tipicamente, os elementos geométricos são colocados em um espaço de coordenadas tridimensionais denominado *M odelSpace.* Utiliza-se esse espaço para "criar" um desenho ou representação gráfica.

O *PaperSpace* é um espaço destinado aos elementos finais, necessários para impressão ou plotagem de um desenho, tais como vistas das entidades, textos, linhas de cota e demais elementos necessários à formatação final do trabalho.

3.4 Entidade

Na hierarquia de classes do AutoCAD, apenas um grupo de classes tem representação gráfica. Nesse grupo, todas as classes são derivadas da classe abstrata *AcDbEntity,* que define todos os seus pré-requisitos. Para diferenciação com os demais objetos das demais classes de gerência do AutoCAD, os objetos dessas classes recebem o nome de entidades.

3.5 Desenho como um Banco de Dados

Definiu-se que uma entidade é um objeto com representação gráfica. Um desenho pode ser definido como uma coleção de entidades criadas e ordenadas segundo regras pré-definidas. Em um desenho do AutoCAD, essa estrutura é fornecida pela utilização de um banco de dados orientado a objetos, subdividido em tabelas, conforme ilustra a figura 3.1. Todos os objetos armazenados no banco de dados de um desenho possuem no nome o prefixo *AcDb.* O significado dessa sigla será apresentado mais adiante.

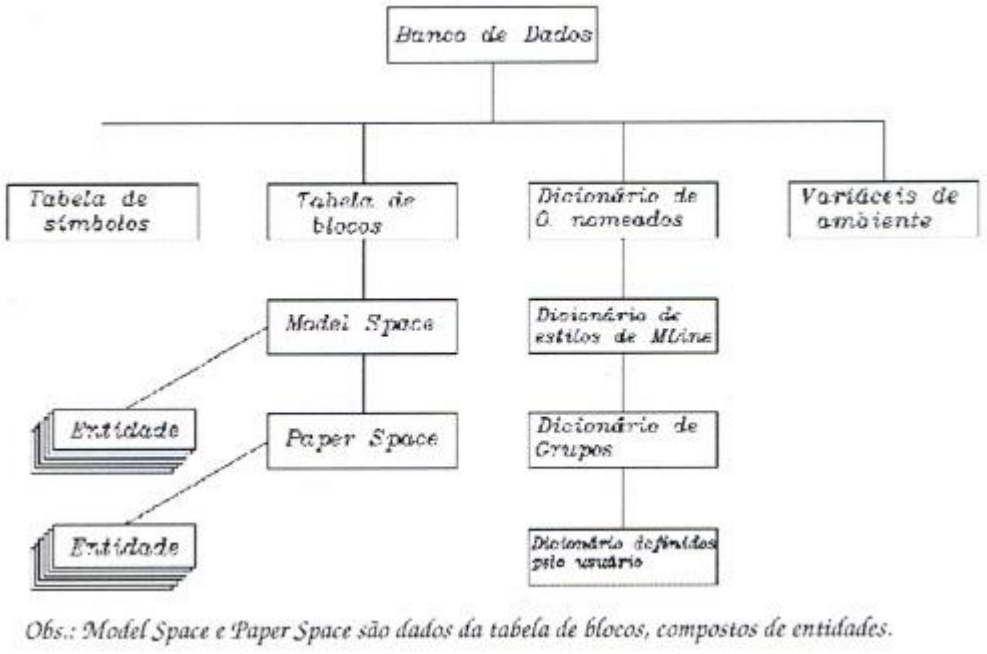

Figura 3.1: Estrutura do banco de dados de um desenho no AutoCAD.

Vale ressaltar que nesse banco de dados, além de entidades, existem objetos auxiliares que implementam as formatações gerais de um desenho, (textos, tipos de linhas, dentre outras) e a intercomunicação entre objetos (reatores, notificadores, dentre outros).

A seguir, apresenta-se os elementos que compõe o banco de dados de um desenho.

3.5.1 Tabelas de símbolos

Um desenho no AutoCAD contém nove tabelas de símbolos, onde são armazenados os objetos criados. Essas tabelas são:

AcDc ViewTable

Nessa tabela de símbolos são armazenados os objetos do tipo *AcDb ViewTableRecord,* que representam as vistas de um desenho armazenadas no banco de dados.

AcDc ViewportTable

Uma *Viewport* é uma janela de visualização que pode ser criada em um desenho. Nessa tabela de símbolos são armazenados objetos do tipo *AcDb ViewportTableRecord,* que representam a configuração de *viewports* em um desenho.

AcDbLinetypeTable

Essa tabela contém objetos do tipo *AcDbLinetype TableRecord,* que descrevem os tipos de linhas existentes em um desenho.

AcD bLayerTable

Essa tabela contém objetos do tipo *AcDbLayerTableRecord,* que descrevem as *Layers* ou camadas existentes em um desenho.

AcDbTextStyleTable

Essa tabela contém objetos do tipo *AcDbTextStyleTableRecord,* que implementam os estilos de texto disponíveis em um desenho.

AcDbUCSTable

Essa tabela contém objetos do tipo *AcDb UCSTableRecord,* que representam os sistemas de coordenadas utilizados pelo usuário em um desenho. As letras *UCS* significam User Coordinate System (Sistema de coordenadas do usuário).

AcDbRegAppTable

Essa tabela contém objetos do tipo *AcDbRegApp TableRecord,* que representam os nomes de aplicativos registrados para dados estendidos de entidades dos objetos residentes no banco de dados do desenho.

AcD bDimStyleTable

Essa tabela contém objetos do tipo *AcDbDimStyleTableRecord,* que representam os estilos de dimensionamento existentes em um desenho.

AcDbBlockTable

Essa tabela contém objetos do tipo *AcDbBlockTableRecord* que contém as definições de todas as entidades visíveis em um desenho.

Em todas essas tabelas são encontrados objetos do tipo apropriado. Podem-se adicionar entradas a essas tabelas, mas não é possível criar novas tabelas de símbolos.

3.5.2 Tabela de Objetos

Tabelas de objetos são estruturas que contém um tipo único de objeto. Os objetos armazenados em uma tabela podem ser constituídos de um conjunto de objetos diferentes, como é o caso da tabela de blocos.

Criar novas entradas de tabelas é simplesmente uma forma de adicionar novas tabelas de registros de um objeto tabela particular que se deseja expandir. Para tanto, em primeiro lugar é necessário criar um objeto tabela de registros, a partir da tabela que se deseja expandir. Em seguida, são atribuídas as características dessa nova tabela, tais como nome e tipo de camada, entre outras. Finalmente, usamos o método de adição de tabelas, para adicionar nova tabela de registros.

3.5.3 Tabela de blocos

Essa tabela contém os nomes de conjuntos de entidades que são armazenados como bloco de entidade e/ou objetos. O termo blocos se refere a um grupo de entidades.

Apresentam dois conjuntos ou blocos básicos que são chamados *Paper Space* e *Model Spac.*

Na figura 3.2, podemos observar a hierarquia da tabela de blocos.

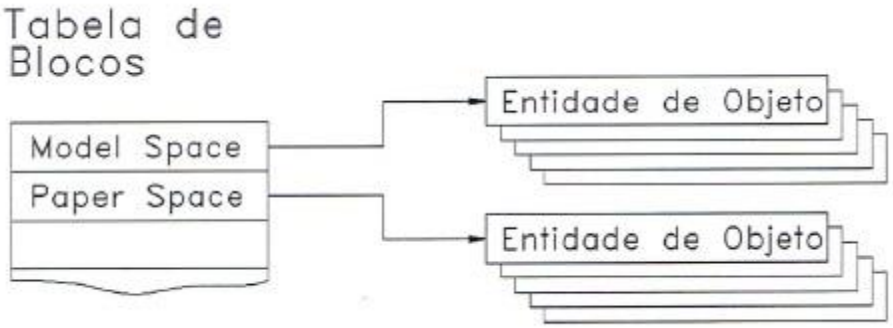

Figura 3.2: Hierarquia da tabela de blocos.

As relações básicas entre os blocos de entidades podem ser observadas na figura 3.3.

3.5.4 Dicionário de Objetos Nomeados

Quando um desenho é criado, além das nove tabelas descritas é criado um *Dicionário de Objetos Nomeados.* Assim como as tabelas de símbolos, esse dicionário contém objetos armazenados no banco de dados.

Enquanto nas tabelas de símbolos são encontrados objetos do tipo específico de cada tabela, no dicionário de objetos são encontrados objetos de todos os tipos, nativos ou não do *AutoCAD.* No estado inicial de um desenho, o dicionário de objetos contém alguns dados que podem ser outros dicionários ou estilos de entidades (por exemplo, estilos de plotagem).

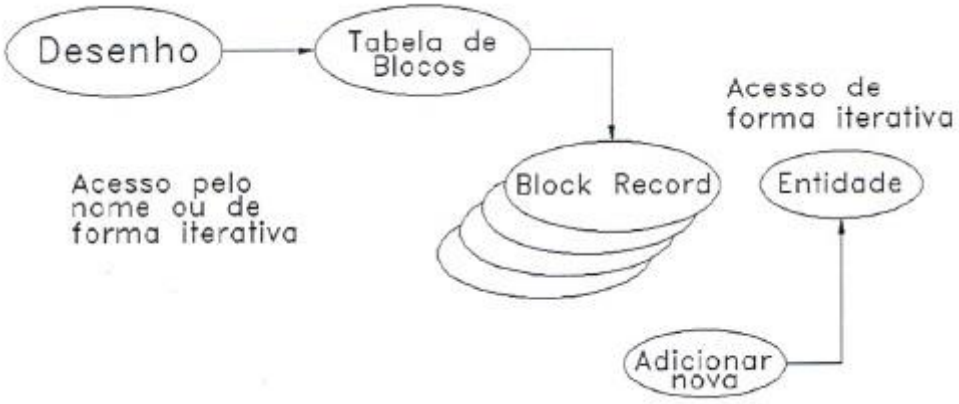

Figura 3.3: Forma de acesso aos blocos de entidades.

3.5.5 Handle de entidades, Id's de objetos e múltiplos bancos de dados

Todos os objetos contidos no banco de dados de um desenho possuem um *AcDb handle,* que é único nesse banco de dados. Além disso, quando um desenho é aberto, é iniciada uma nova sessão de trabalho e nesse momento é associado um número a cada entidade do tipo *AcDb,* denominado *ObjectI* d.

O AutoCAD 2000 permite que sejam manipulados mais de um desenho ao mesmo tempo; portanto, pode-se ter uma sessão de trabalho com mais de um banco de dados aberto ao mesmo tempo. Um *Id* é único em uma sessão de trabalho e um *Handle* é único em um banco de dados. A combinação de um *Id* e um *handle* cria um identificador único na sessão de trabalho, ainda que se tenha mais de um desenho aberto em uma sessão de trabalho. Ao fechar um desenho ou encerrar uma sessão de trabalho, esse *Id* é descartado, mas o seu *handle* é mantido.

Com o *Id* de um objeto, pode-se obter um ponteiro para uma entidade que contém esse *Id.* Com isso, pode-se manipular essa entidade. *Id's* de elementos do banco de dados do AutoCAD são do tipo *AcDbObjectID.* Um *Id* não é necessariamente o mesmo em sessões diferentes de trabalho.

3.5.6 Biblioteca de ObjectARX

Uma biblioteca de classes é um conjunto de itens, classes e funções que são agrupados segundo suas características e funcionalidades. No caso da biblioteca *ObjectARX,* o desenvolvimento se deu na linguagem de programação *C++,* seguindo a filosofia de orientação para objetos.

A seguir é apresentada uma descrição da biblioteca ObjectARX.

3.6 Módulos da biblioteca ObjectARX

Como já dito, o *ObjectARX* contém um conjunto de bibliotecas, para programação em *C++*. Essas bibliotecas estão separadas por funcionalidade. Seus nomes possuem quatro letras. As duas primeiras são AC (do *AutoCAD),* seguidas por outras duas letras, que fazem referência a uma biblioteca particular. Por exemplo, AcDb é utilizado para fazer referência a *AutoCAD DataBase Library.*

Todos os objetos contidos em uma biblioteca têm nome.s que iniciam com as quatro letras que dão nome à classe, facilitando o reconhecimento da classe da qual fazem parte e a qual biblioteca essa classe pertence.

As bibliotecas oferecidas com o *ObjectARX,* são as seguintes:

- AcRx: Essa biblioteca contém rotinas de sistema para executar links de ARX, em arquivos DLL. Contém também ferramentas básicas.

- AcEd: Essa biblioteca contém funções para adicionar novos comandos e para eventos de notificação.

- AcDb: Apresenta as definições de todos os tipos de objetos geométricos do banco de dados e das tabelas do *AutoCad.*

- AcGi: Biblioteca com interfaces de renderização de objetos desenhados.

- AcGe: Biblioteca de definições de geometria de objetos, tais como, pontos, além de ferramentas para manipulação de dados de geometria.

- ADS: Biblioteca com utilitários para seleção e manipulação de entidades em um ambiente interativo, para *AutoCAD R14.*

Esses utilitários estão agora integrados às bibliotecas do *ObjectARX 2000,* listadas acima.

3.7. Conectando aplicativos ao AutoCAD

O processo de *link* entre o *AutoCAD* e um aplicativo desenvolvido com o *ObjectARX* requer alguns passos. Boa parte do trabalho de edição é realizado internamente pelo *AutoCAD,* quando é carregado um módulo, ou aplicativo *ObjectARX*

Em um típico programa em *C++,* a execução se inicia com uma rotina chamada "main". Em uma *DLL ObjectARX,* a execução se inicia com a rotina *"acrxEntryPoint".* Em todos os programas deve existir essa função para que o AutoCAD possa iniciar o aplicativo. **.**

Sempre que um programa *ObjectARX* é carregado, é realizada uma operação de *"ObjectARX Load"* e o *AutoCAD* pode iniciar a comunicação entre as sub-rotinas de *"entry point"* do módulo carregado. Depois de carregado, um programa *ObjectARX* é acionado quando um novo desenho é iniciado ou quando o usuário utiliza alguma de suas funções.

Uma das mais importantes atividades que acontecem quando iniciamos a primeira execução da sub-rotina *"entry point"* é que ponteiros podem ser criados entre os aplicativos customizados e o *AutoCAD.* É importante salientar que a função *"entry point"* trata duas mensagens. A primeira é para iniciar um novo comando. A segunda mensagem é para indicar que um desenho está sendo carregado. Isso significa que o *AutoCAD* está totalmente operacional e os comandos podem ser utilizados.

3.7.1 ACED: Biblioteca de edição de entidades do AutoCAD

Essa biblioteca, chamada AcEd, é a mais utilizada por um aplicativo *ObjectARX* Ela contém rotinas que auxiliam os programadores a declarar novos comandos para *AutoCAD.* No *ObjectARX,* comandos podem ser adicionados ao *AutoCAD,* colocando-os em uma pilha de comandos do programa.

A pilha de comandos é um objeto definido na biblioteca AcEd. Ela contém os comandos disponíveis para serem utilizados pelos usuários. Por meio da função "*add Command",* um novo comando é adicionado a essa pilha, no espaço de memória alocado para

armazenar os comandos. Esse local é o endereço da sub-rotina escrita para criar o novo comando, que é definido no momento em que a sub-rotina é carregada. A linguagem de programação *C++* permite criar funções simples para obter esse endereço, o qual é um parâmetro necessário para o funcionamento correto do comando.

Tendo uma pilha de comandos, o *AutoCAD* aciona eficientemente o programa *ObjectARX Os* comandos que são encontrados na pilha de comandos, são considerados comandos nativos.

Pelo fato de se encontrarem residentes na memória, em um nível muito próximo ao *A utoCAD,* os eventos gerados com o acionamento de um aplicativo *ObjectARX* são interpretados diretamente pelo programa, como se fosse um evento dele próprio. Em outras palavras, um aplicativo é acionado mediante um evento interno do *AutoCAD,* e não somente porque um comando foi acionado. Com isto percebe-se que esses aplicativos têm a habilidade de monitorar eventos e manipular entidades dentro do programa. Essa é a forma mais indicada de se desenvolver customizações CAD/CAM/CAE.

3.7.2 ACED: Editor de eventos

A biblioteca AcEd também contém objetos que possibilitam o desenvolvimento de aplicativos com a funcionalidade de atuar no editor de desenhos do *AutoCAD.* As classes dessa biblioteca contêm definições de funções que podem ser chamadas se um determinado evento acontecer no nível do editor.

Em muitos casos, eventos do editor resultam em aplicativos que carregam ou armazenam informações em um arquivo. Essas funções se enquadram na categoria conhecida como reatores, ou funções que reagem a uma função, retomando algum tipo de informação. Em outras palavras, elas funcionam em reação ou resposta a algum evento. Por outro lado, esses aplicativos não têm a condição de cancelar uma atividade ou simplesmente enviar uma notificação, caso um determinado evento ocorra.

Essa rotina de notificação, está tipicamente disponível quando uma atividade é iniciada ou finalizada no editor de desenhos. Também existem notificações específicas se acontecerem erros e quando uma atividade é cancelada.

3.7.3 AcDb: Biblioteca do Banco de Dados de Objetos do AutoCAD

Alguns aplicativos podem editar um desenho corrente e, para isso, deve ser estabelecida uma conexão entre a área de trabalho e o banco de dados de objetos do AutoCAD. Esse acesso é feito por meio da biblioteca AcDb, que contém definições de objetos para todas as entidades do AutoCAD e para determinar alterações nas tabelas dos desenhos.

Um desenho é um objeto que contém muitos *outros* objetos, como pode ser visto na Figura 3.4.

Esses objetos podem ser tabelas, definições ou listas de entidades que formam o desenho. Para um aplicativo acessar o banco de dados de um desenho, deve acessar um objeto de desenho antes.

O ObjectARX nos permite trabalhar com mais de um desenho aberto na área de trabalho ao mesmo tempo. Apenas o desenho corrente pode ser acessado em uma seção de edição e isto é feito por meio do acesso à lista de objetos desse desenho. Em um ambiente que permite trabalhar com múltiplos documentos, o operador precisa tornar um documento ativo para poder realizar alguma operação nele.

Figura 3.4: Objetos de um desenho no AutoCAD.

Desenhos que não estejam abertos na área de trabalho podem ser inseridos em um aplicativo ObjectARX para serem manipulados. Esses não poderão ser vistos caso não sejam

inseridos em um outro desenho corrente, ou caso não sejam abertos, mas as edições poderão ser feitas. Um exemplo de utilização de múltiplos documentos, pode ser visto com a utilização dos comandos *INSERT* e *XREF,* pelos quais um outro arquivo *D WG* é aberto e lido dentro de um desenho corrente.

Para trabalhar com um desenho diferente do desenho corrente, existe uma função chamada "*readDwgFile",* que pode ser utilizada para abrir um desenho como sendo um objeto. Uma vez aberto, o programa pode manipular os objetos contidos no segundo desenho com as mesmas técnicas utilizadas para manipular os objetos do desenho principal. Isso é uma característica da utilização da programação orientada a objetos para o AutoCAD.

Usualmente, as funções básicas do AutoCAD trabalham no desenho corrente e manipulam diretamente as suas entidades. O acesso a entidades individuais é obtido por meio de ponteiros. Uma forma de obter esse tipo de ponteiro é por meio de seleção de objetos em tela. Esses ponteiros são também encontrados em várias tabelas de objetos como na tabela de blocos.

3.8. Outras bibliotecas encontradas no ObjectARX

Existem ainda duas outras bibliotecas que contém classes utilizadas por outras bibliotecas para diferentes operações e manipulações. São elas:

ACGE: Definição de geometria de objetos

Essa biblioteca utiliza as classes de definição de geometria para elementos constituintes de outros objetos. Esses incluem pontos, vetores e matrizes. Para objetos existentes, os membros da biblioteca de geometria de objetos contêm métodos para manipulação de dados desses objetos. Existem funções para realizar operações básicas com vetores, tais como produto vetorial, além de muitas outras. Muitas das manipulações matemáticas que se deseja realizar em um ambiente computacional gráfico podem ser encontradas nessa biblioteca. É possível reconhecer dois grupos básicos nessa biblioteca. Um é um conjunto de ferramentas para objetos bidimensionais e o outro é para objetos tri dimensionais. Para a maior parte, existem as mesmas funcionalidades para os dois grupos.

ACGI: Biblioteca de interfaces gráficas

As entidades do AutoCAD são desenhadas em tela utilizando-se as funções encontradas nessa biblioteca. Ela contém classes que descrevem como desenhar todos os elementos básicos, tais como linhas, arcos e textos. É importante lembrar que os objetos não são adicionados ao banco de dados do desenho por meio dessa biblioteca. Em vez disto, a biblioteca de banco de dados utiliza as suas funções para renderizar os objetos descritos. A biblioteca AcGi permite acessar os processos requisitados por eventos como salvar um slide, ou representações gráficas temporárias, tais como pontos de captura (Object Snap). Elementos gráficos criados com a classe AcGi podem ser direcionados para todas as vistas existentes ao mesmo tempo.

Quando é criado um objeto customizado, a biblioteca de interface gráfica é utilizada para realizar a geração do novo objeto. Suponha-se que o novo objeto seja composto de muitas faces. Serão necessárias muitas chamadas a funções da biblioteca AcGi para renderizar as faces cheias em duas ou três dimensões.

4 Teoria de Timoshenko para vigas

4.1 Hipóteses

Para determinação da formulação de viga utilizada nesse trabalho, foram adotadas as seguintes hipóteses principais [8]:

1. O modelo utilizado foi o modelo de viga descrito por *Timoshenko,* no qual as seções planas, normais ao eixo da viga antes de uma deformação, permanecem planas após uma deformação, mas não necessariamente normais ao eixo.

2. A lei constitutiva adotada foi o regime elástico linear isotrópico, descrita no item C.5.

3. Admitiu-se ainda o estado tri axial de tensões com as seguintes condições: $\sigma_x \neq 0$, $\sigma_y = 0$, $\sigma_z = 0$, $\tau_{xy} \neq 0$, $\tau_{xz} \neq 0$.

4. A viga pode estar em uma posição qualquer do espaço.

4.2 Definição da cinemática

No estudo cinemático, tomemos o ponto P, na figura 4.1. Após uma translação, o ponto *P* passa para uma posição *pl,* e tem-se o vetor deslocamento *u,* expresso por

$$u(x, y, z) = u(x) + \theta_x \cdot a$$

Em que:

$u \rightarrow$ Deslocamento no eixo;

$\theta_x \rightarrow$ Rotação da seção transversal;

$a \rightarrow$ Vetor que vai do eixo de referência ao ponto P.

Todos se referem ao eixo da viga.

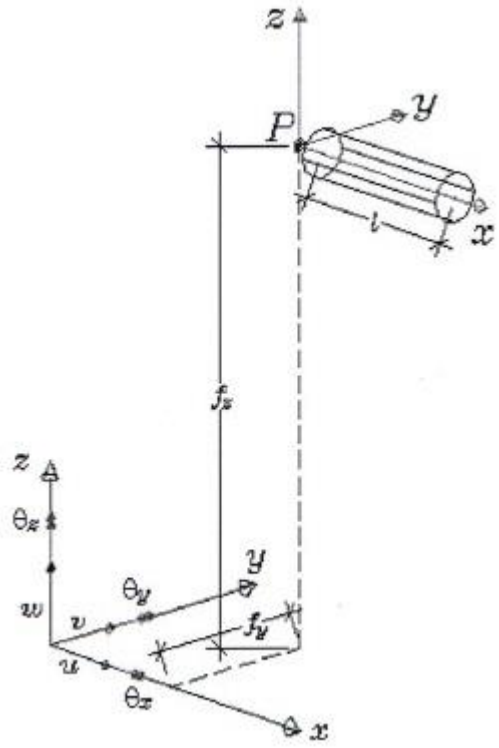

Figura 4.1: Elemento de barra em uma posição qualquer do espaço.

4.3 Deslocamentos

Os deslocamentos em um elemento de barra, como descrito na figura 4.1, são expressos por um vetor u, em função dos deslocamentos na direção dos eixos coordenados x, y e z. Seguindo a definição em C.2, o deslocamento é expresso algebricamente por

$$u(x,y,x) = \begin{bmatrix} u(x) - \theta_z(x)(f_y + y) + \theta_y(x)(f_z + z) \\ v(x) - \theta_x(x)(f_z + z) \\ w(x) + \theta_x(x)(f_y + y) \end{bmatrix}$$

4.4 Deformações

As deformações do ponto P foram determinadas em função dos seus deslocamentos, em relação ao sistema global de referência. O tensor de deformação $E = \varepsilon$ é expresso na forma da equação (118) . O jacobiano do deslocamento é dado por

$$\nabla u(x,y,x) = \begin{bmatrix} u(x) - \theta_z(x)(f_y + y) + \theta_y(x)(f_z + z) & -\theta_z(x) & \theta_y(x) \\ v(x) - \theta_x(x)(f_z + z) & 0 & -\theta_x(x) \\ w(x) + \theta_x(x)(f_y + y) & \theta_x(x) & 0 \end{bmatrix}$$

Substituindo a expressão do Jacobiano na expressão de ε, teremos

$$\varepsilon = \begin{bmatrix} u'(x) + \theta'_y(x)(f_z+z) - \theta'_z(x)(f_y+y) & \frac{1}{2}(v'(x) - \theta'_x(x)(f_z+z) - \theta_z(x)) & \frac{1}{2}(w'(x) + \theta'_x(x)(f_y+y) + \theta_y(x)) \\ \frac{1}{2}(v'(x) - \theta'_x(x)(f_z+z) - \theta_z(x)) & 0 & 0 \\ \frac{1}{2}(w'(x) + \theta'_x(x)(f_y+y) + \theta_y(x)) & 0 & 0 \end{bmatrix}$$

Pela simetria da matriz, podemos escrevê-la como um vetor de deformações, em função das suas componentes,

$$\varepsilon = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} u'(x) + \theta'_y(x)(f_z+z) - \theta'_z(x)(f_y+y) \\ 0 \\ 0 \\ 0 \\ (w'(x) + \theta'_x(x)(f_y+y) + \theta_y(x)) \\ (v'(x) - \theta'_x(x)(f_z+z) - \theta_z(x)) \end{bmatrix}$$

em que ε_x representa a deformação axial da barra e γ_{xz} e γ_{xy} representam as componentes da distorção na face do elemento.

É importante ressaltar que no tensor ε os valores das tensões ε_y, ε_z e ε_{yz} fornecidos pelo cálculo do jacobiano são nulos, mas pelas considerações expostas no item *C.5,* sabe-se que são observados efeitos de deformação indireta nas direções *y* e *z* , muito embora sejam nulas as tensões nestas direções.

4.5 Tensões

Para o estado uniaxial de tensões e seguindo o desenvolvimento C.5.3, tem-se que as componentes do tensor de tensões são

$$\sigma = \begin{bmatrix} \sigma_x \\ 0 \\ 0 \\ 0 \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix}$$

4.6 Esforços solicitantes

Pela hipótese da teoria clássica de barras, os esforços em um elemento são:

- Força Normal N_x:

$$N_x = \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} \sigma_x \, dydz = bhE \, (f_z\theta'_y(x) - \, (f_y\theta'_z(x) + \, u'(x))$$

- Forças cortantes V_{xy} e V_{xz}:

$$V_{xy} = k \cdot \left(\int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} \tau_{xy} \, dydz \right) = bhGk \left(v'(x) - \, \theta_z \, (x) - \, f_z\theta'_x(x) \right)$$

$$V_{xz} = k \cdot \left(\int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} \tau_{xz} \, dydz \right) = bhGk \, \left(w'(x) + \, \theta_y \, (x) + \, f_y\theta'_x(x) \right)$$

- Momentos fletores M_x, M_y e M_z:

$$M_x = k \cdot \left(\int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} \left[\left((f_y + y)\tau_{xz} - \, (f_z + z) \, \tau_{xy} \right) \right] \, dydz \right)$$

$$\mathrm{M_x} = \mathrm{bhGk} \cdot \left(\mathrm{f_y\theta_y(x) + f_z\theta_z(x) + f_y^2 \, \theta'_x(x) + f_z^2 \, \theta'_z(x) - f_z v'(x) + f_y w'(x)} \right) + \mathrm{GJ\theta'_X(x)}$$

em que *b* é a largura da seção, *h* é a altura da seção, *G* é o módulo de elasticidades transversal, *E* é o módulo de elasticidade longitudinal, *k* é o coeficiente de distorção transversal

e *J* é o momento polar de inércia da seção. O fator *k,* que aparece multiplicando os esforços *Vxy* , *Vxz* e *Mx,* é um fator de correção da distribuição dessas tensões ao longo da seção do elemento de barra. Tendo em vista essa distribuição não linear ao longo da seção do elemento, faz-se necessária a introdução desse fator.

4.7 Ações

O elemento de barra implementado pode trabalhar com cargas concentradas nos nós, pontuais ou de momento, e cargas distribuídas ao longo de seu eixo maior. As forças pontuais são aplicadas aos nós da barra, em relação ao eixo global do sistema ou em relação ao eixo local da barra. As forças e momentos são aplicados ao longo do eixo da barra, podendo ser aplicadas em relação ao eixo global do sistema ou em relação ao eixo local da barra.

A convenção de sinais é idêntica à utilizada nas formulações, expressa pela "regra da mão direita".

4.8 Aplicação do PTV

Como exposto no item C.6, ao se aplicar o método para uma barra, busca-se uma condição de equilíbrio na qual o trabalho externo das forças é igual ao trabalho interno das forças. Assim, o trabalho das forças internas será dado pela combinação dos esforços externos com os deslocamentos compatíveis na direção das forças, ou seja, pelo produto entre os esforços aplicados na superfície da barra e as variações dos deslocamentos no plano de referência.

Para a aplicação do PTV, utilizou-se a estratégia apresentada por Menezes & Devloo [14]. Com isso, a partir da expressão do PTV, determinou-se um conjunto de quatro matrizes, denominadas “matrizes *de contribuição"*. Com essa técnica, podem ser obtidos os coeficientes da matriz de rigidez e da contribuição da carga nodal equivalente em cada nó para o vetor de carga do sistema.

Demonstra-se que o *PTV* pode ser escrito com base nas expressões integrais dos esforços, obtendo-se uma expressão equivalente à expressão global do *PTV* A utilização desse recurso tem por objetivo permitir a utilização do fator de correção *k* para os esforços V_{xy} , V_{xz} e M_x. Da equação (C.32), tem-se que o trabalho interno, expresso em função das variações dos deslocamentos, será dado pela expressão

$$w_{int} = \int_A \left[N_x \frac{\partial \delta u(x)}{\partial x} + V_{xy}\left(\frac{\partial \delta v(x)}{\partial x} - \delta v_z(x)\right) + V_{xz}\left(\frac{\partial \delta w(x)}{\partial x} - \delta v_y(x)\right) + M_x \frac{\partial \delta \theta_x(x)}{\partial x} + M_y \frac{\partial \delta \theta_y(x)}{\partial x} + M_z \frac{\partial \delta \theta_z(x)}{\partial x} \right] \quad (4.1)$$

Com as variáveis δu, δv, δw, $\delta\theta_x$, $\delta\theta_y$, $\delta\theta_z$, representam-se as variações dos deslocamentos fundamentais.

Seguindo o trabalho de Menezes & Devloo[14], os deslocamentos fundamentais foram agrupados em um vetor denominado *{u},* no qual

$$\{u\}^T = \{u(x)v(x)w(x)\ \theta_x\ \theta_y\ \theta_z\} \quad (4.2)$$

Por conveniência, definiu-se um vetor *{δu},* cujas componentes são as variações dos deslocamentos fundamentais, que são

$$\{\delta u\}^T = \{\delta u(x)\ \delta v(x)\ \delta w(x)\ \delta\theta_x\ \delta\theta_y\ \delta\theta_z\} \quad (4.3)$$

As derivadas parciais envolvidas na equação do *PTV* são em relação à variável independente *x.* Utilizando-se o operador linear $\frac{\partial}{\partial x}$ essas derivadas podem ser agrupadas em dois novos vetores 6 x 1. Dessa forma, o integrando da equação que dá o trabalho virtual dos esforços internos associado a um ponto pode ser agrupado matricialmente a partir de quatro vetores que são:

- Vetor das variações dos deslocamentos $\{\delta u\}$
- Vetor das derivadas parciais das variações dos deslocamentos em relação a x: $\frac{\partial\{\delta u\}}{\partial x}$
- Vetor de deslocamentos $\{u\}$
- Vetor das derivadas parciais dos deslocamentos em relação a x : $\frac{\partial\{u\}}{\partial x}$

Utilizando-se o programa Mathemática, foi possível resolver analiticamente a expressão (C.34) e arranjar as parcelas em 4 grupamentos matriciais, conforme indicado a seguir.

$$\mathrm{w_{int}} = \int_A \left[\left(\frac{\partial\{\delta u\}}{\partial x}\right)^{\mathrm{T}} \mathrm{K_{xx}} \left(\frac{\partial\{u\}}{\partial x}\right)^{\mathrm{T}} + \{\delta u\}^{\mathrm{T}} \mathrm{B_{0x}} \left(\frac{\partial\{u\}}{\partial x}\right) + \left(\frac{\partial\{\delta u\}}{\partial x}\right)^{\mathrm{T}} \mathrm{B_{x0}} \{u\} + \{\delta u\}^{\mathrm{T}} \mathrm{B_{00}} \{u\}\right] dx \quad (4.4)$$

4.8.1 Matriz de rotação

Para o caso de um elemento em uma posição qualquer do espaço, introduziu-se uma matriz de rotação *R.* Um vetor *V* pode ser representado por suas componentes *(Vx, Vy, Vz)* no sistema global de referência ou por componentes *(Vnb Vn2, Vn3)* em relação a um sistema local de referência. Pode-se definir uma matriz *R,* denominada matriz de rotação, de ordem 3 x 3, que permite escrever as componentes *(Vnb Vn2, Vn3)* em função das componentes *(Vx, Vy, Yz).* Os coeficientes dessa matriz são os cossenos diretores dos ângulos entre os vetores *aI, a2* e *a3* e os vetores unitários que representam o sistema global de referência.

$$R = \begin{bmatrix} a_{11} & a_{21} & a_{31} & 0 & 0 & 0 \\ a_{12} & a_{22} & a_{32} & 0 & 0 & 0 \\ a_{13} & a_{23} & a_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{11} & a_{21} & a_{31} \\ 0 & 0 & 0 & a_{12} & a_{22} & a_{32} \\ 0 & 0 & 0 & a_{13} & a_{23} & a_{33} \end{bmatrix}$$

Nesse sistema os coeficientes a_{ij} da matriz, são as componentes $\mathrm{a_1}$ ($\mathrm{a_{11}}$, $\mathrm{a_{12}}$, $\mathrm{a_{13}}$), $\mathrm{a_2}$ ($\mathrm{a_{21}}$, $\mathrm{a_{22}}$, $\mathrm{a_{23}}$), $\mathrm{a_1}$ ($\mathrm{a_{31}}$, $\mathrm{a_{32}}$, $\mathrm{a_{33}}$) de um vetor *a* que localiza um elemento no espaço.

Seguindo o desenvolvimento para a aplicação do *PTV,* acrescentando a matriz *R,* tem-se

$$\mathrm{w_{int}} = \int_A \left[\left(\frac{\partial\{\delta u\}}{\partial x}\right)^{\mathrm{T}} (R^T \cdot \mathrm{K_{xx}} \cdot R) \left(\frac{\partial\{u\}}{\partial x}\right)^{\mathrm{T}} + \{\delta u\}^{\mathrm{T}} (R^T \cdot \mathrm{B_{0x}} \cdot R) \left(\frac{\partial\{u\}}{\partial x}\right) + \left(\frac{\partial\{\delta u\}}{\partial x}\right)^{\mathrm{T}} (R^T \cdot \mathrm{B_{0x}} \cdot R)\{u\} + \{\delta u\}^{\mathrm{T}} (R^T \cdot \mathrm{B_{00}} \cdot R)\{u\}\right] dx \quad (4.5)$$

4.9 Aplicação do método dos elementos finitos

Para aplicar o método dos elementos finitos, nesse trabalho (oi utilizada a proposta encontrada no trabalho de *Menezes & Devloo*[14]. As formulações foram implementadas em um programa de elementos finitos denominado *PZ,* que apresenta aos usuários um espaço de interpolação contínuo e um procedimento de integração sobre os elementos. Para implementar um sistema de equações diferenciais, o usuário do ambiente precisa criar um *método"* que - dado os valores das funções de forma e das suas derivadas, a localização do ponto de integração e as direções dos eixos Jacobianos - calcula as contribuições para a matriz de rigidez do elemento e para o vetor de carga. O restante do programa já está em funcionamento propiciando ao usuário a oportunidade de escolher o tipo de solução de sistema mais conveniente.

A matriz de rigidez e o vetor de ações nodais são montados por contribuições dos pontos de integração. Os deslocamentos componentes dos vetores {u} e {δu} e as suas derivadas parciais são discretizadas por meio de funções de forma construídas segundo o método dos elementos finitos. São geradas quatro equações para cada função de interpolação.

Para o cálculo da contribuição de um dado ponto de integração para a matriz de rigidez, adicionou-se no programa de elementos finitos uma função que implementa a expressão (4.1). O número de variáveis associadas a um nó é seis. Enumerando as funções de forma φ com índices i e *j* e as variáveis do sistema de equações diferenciais com α e β, a contribuição para a matriz de rigidez será

$$K_{(6i+\alpha,6j+\beta)} = w \cdot \left[(K_{xx})_{\alpha\beta} \frac{\partial \varphi_i}{\partial \mathrm{a}_1} \frac{\partial \varphi_i}{\partial \mathrm{a}_1} + (B_{0x})_{\alpha\beta\varphi i} \frac{\partial \varphi_i}{\partial \mathrm{a}_1} \varphi_j + (B_{00})_{\alpha\beta\partial\varphi i\partial\varphi j} \right] \quad (4.6)$$

em que $\alpha = 0 \rightarrow 6$ e $\beta = 0 \rightarrow 6$. Os índices i e *j* variam de acordo com o número de funções de forma e ω é a função *"peso de integração"*

As quatro matrizes que estão na expressão do trabalho interno, que são K_{xx}, B_{0x}, B_{x0} e B_{00}, são denominadas matrizes de contribuição. Essas são de ordem 6x6 e sua utilização simplifica a determinação da contribuição de cada elemento na matriz de rigidez global do sistema, na aplicação do método dos elementos finitos.

As matrizes *Kxx, Box, Bxo* e , *Boo,* são calculadas como:

$$K_{xx}$$

$$= \begin{bmatrix} bEh & 0 & 0 & 0 & bEf_z h & -bEf_y h \\ 0 & bGhk & 0 & -bf_z Ghk & 0 & 0 \\ 0 & 0 & bGhk & bf_y Ghk & 0 & 0 \\ 0 & -bf_z Ghk & bf_y Ghk & J_0 + bGhk\left(J + f_y^2 + f_z^2\right) & 0 & 0 \\ bEf_z h & 0 & 0 & 0 & \frac{1}{12} bEh\left(12 f_Z^2 + h^2\right) & -bEf_y f_z h \\ -bEf_y h & 0 & 0 & 0 & -bEf_y f_z h & Eh\left(\frac{b^3}{12} + bf_y^2\right) \end{bmatrix}$$

$$B_{0x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & bGhk & bf_y Ghk & 0 & 0 \\ 0 & -bGhk & 0 & bf_z Ghk & 0 & 0 \end{bmatrix}$$

$$B_{x0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -bGhk \\ 0 & 0 & 0 & 0 & bGhk & 0 \\ 0 & 0 & 0 & 0 & bf_y Ghk & bf_z Ghk \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_{00} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & bGhk & 0 \\ 0 & 0 & 0 & 0 & 0 & bGhk \end{bmatrix}$$

A contribuição do vetor de carga envolve somente valores das funções de forma. Considerando-se que as forças por unidade de área são (f_x, f_y, f_z), as contribuições para o vetor de carga serão:

$$EF_{\lambda i} += \omega \cdot f_x$$

$$EF_{\lambda i+1} += \omega \cdot f_y$$

$$EF_{\lambda i+2} += \omega \cdot f_z$$

5 Extensão da formulação de Timoshenko para vigas curvas

Na caracterização de uma viga curva, tome-se um ponto qualquer no eixo de uma viga, de coordenadas $P(\xi) = (x(\xi), y(\xi), z(\xi))$, em que ξ é uma coordenada curvilínea. Com essa parametrização, pode-se calcular o *jacobiano* para cada ponto de integração e, dessa forma, pode-se determinar a matriz de rotação em cada um desses pontos.

Tem-se um vetor unitário na direção do eixo da viga, dado pela expressão:

$$v_1(\xi) = \frac{\left(\frac{\partial x(\xi)}{\partial \xi}, \frac{\partial y(\xi)}{\partial \xi}, \frac{\partial z(\xi)}{\partial \xi}\right)}{\sqrt{\left(\frac{\partial x(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial y(\xi)}{\partial \xi}\right)^2 + \left(\frac{\partial z(\xi)}{\partial \xi}\right)^2}} \qquad (5.1)$$

na qual o denominador da expressão é o *jacobiano,* que irá variar a cada ponto.

Conforme o trabalho de Devloo [6], associado ao vetor axial $\overrightarrow{v_1}(\xi)$, como ilustra a figura 5.1, podem ser definidos os eixos $\overrightarrow{v_2}(\xi)$ e $\overrightarrow{v_3}(\xi)$ que caracterizam os eixos locais principais do elemento de barra curva.

Com isso, podemos localizar qualquer ponto ao longo do eixo da viga pelas coordenadas:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

e

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}$$

nas quais u_1, u_2, u_3 são os deslocamentos nos eixos locais, e u_x, u_y, u_z são os deslocamentos nos eixos globais. As expressões das formulações apresentadas para elementos de barra podem ser reescritas por suas coordenadas locais

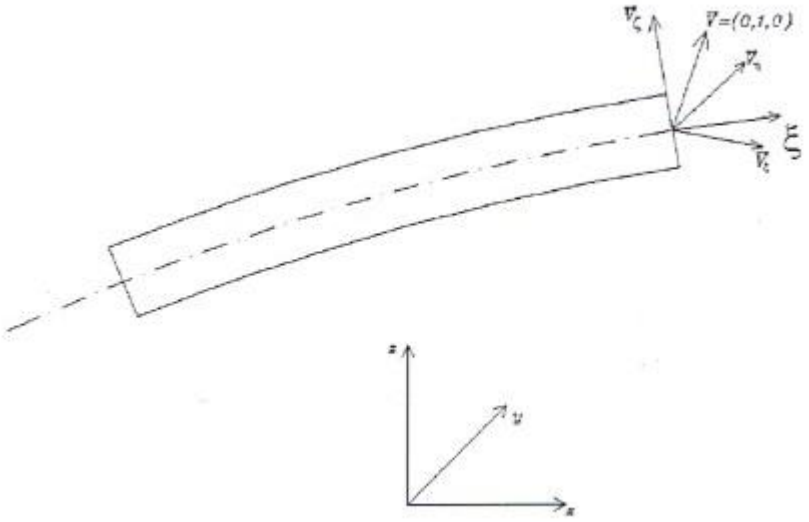

Figura 5.1: Sistema de eixos associado ao ponto de integração para um elemento de barra, onde $\overrightarrow{v_\xi} = \overrightarrow{v_1}, \overrightarrow{v_n} = \overrightarrow{v_2}, \overrightarrow{v_\varsigma} = \overrightarrow{v_3}$ ..

$$\varepsilon = [\varepsilon_1\ \varepsilon_2\ \varepsilon_3\ \gamma_{23}\ \gamma_{13}\ \gamma_{12}]^T$$

O elemento é definido por correspondência entre as coordenadas globais *x, y, z* e a coordenada curvilínea ξ. A coordenada ξ do elemento mestre varia de -1 a + 1. Na seção transversal, para um dado ponto, define-se o vetor $\overrightarrow{V_1}(\xi)$ na direção axial do elemento, conforme (5.1), e os vetores $\overrightarrow{V_2}(\xi)$ e $\overrightarrow{V_3}(\xi)$ nas direções dos eixos principais de inércia.

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, em que 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, em que um ponto genérico é descrito através de coordenadas cilíndricas *r,* θ e *z.*

Para isso, foi necessário utilizar os jacobianos unidimensionais para as transformações entre os elementos deformadas descritos em coordenadas cartesianas, ou coordenadas cilíndricas, e o elemento mestre, descrito em coordenada adimensional ξ.

5.1 Jacobiano unidimensional para coordenadas cartesianas

Conforme descrito por *Becker,* o jacobiano para coordenadas cartesianas unidimensionais é definido por

$$x = \sum x_i N_i(\xi)$$

$$y = \sum y_i N_i(\xi)$$

$$z = \sum z_i N_i(\xi)$$

e

$$\frac{\partial x}{\partial \xi} = \sum x_i N'_i(\xi)$$

$$\frac{\partial y}{\partial \xi} = \sum y_i N'_i(\xi)$$

$$\frac{\partial z}{\partial \xi} = \sum z_i N'_i(\xi)$$

Pode-se reescrever a expressão do Jacobiano por:

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

Com o vetor $\overrightarrow{v_1}(\xi)$ descrito no item (5.1) os vetores $\overrightarrow{v_2}(\xi)$ e $\overrightarrow{v_3}(\xi)$, podem ser calculados diretamente as contribuições dos coeficientes de rigidez locais na matriz de rigidez global sem a necessidade de mudança do sistema de coordenadas.

5.2 Jacobiano unidimensional para coordenadas cilíndricas

Para realizar o cálculo das deformações e a análise de esforços em elementos curvos no espaço com maior precisão, utilizou-se um elemento finito unidimensional, descrito em coordenadas cilíndricas, pois melhor define a geometria do problema. O *jacobiano* da transformação de coordenadas cilíndricas para coordenadas unidimensionais ξ para elemento mestre pode ser obtido de acordo com:

$$r(\xi) = \sum r_i N_i\,(\xi)$$

$$\theta(\xi) = \sum \theta_i N_i\,(\xi)$$

$$z(\xi) = \sum z_i N_i\,(\xi)$$

e

$$\frac{\partial r}{\partial \xi} = \sum r_i N'_i\,(\xi)$$

$$\frac{\partial \theta}{\partial \xi} = \sum \theta_i N'_i\,(\xi)$$

$$\frac{\partial z}{\partial \xi} = \sum z_i N'_i\,(\xi)$$

Então

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

em que r, θ e z representam as coordenadas em relação ao sistema de coordenadas cilíndricas, tomando em cada ponto de integração.

6 Elemento de placa

6.1 Introdução

Entende-se como placa um elemento estrutural plano, no qual a espessura é muito menor que as outras duas dimensões. Esse elemento ou placa pode ser fino, ou espesso, e essa classificação se baseia na relação entre a espessura do elemento e sua menor dimensão.

Nesse trabalho, foi utilizada a teoria de *Reissner e Mindlin* para placas espessas. Além disso, utilizou-se também o desenvolvimento matemático apresentado no trabalho de *Menezes e Devloo* [14].

Na figura 6.1, pode-se observar uma representação de um elemento de placa.

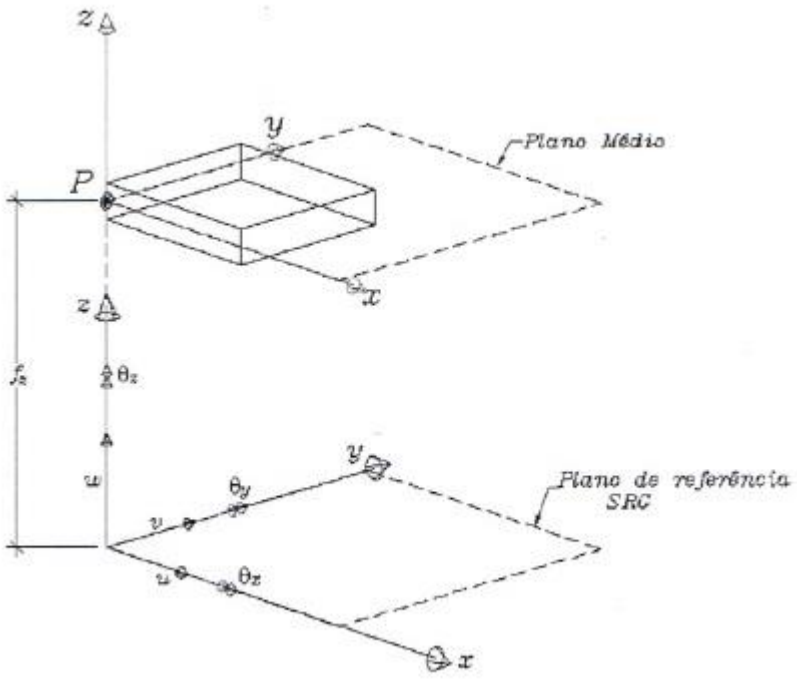

Figura 6.1: Sistema de coordenadas para um elemento de barra

6.2 Hipóteses

Nesse trabalho foram consideradas as seguintes hipóteses de *Reissner-Mindlin* que são:

1. O domínio Ω é dado por:

$$\Omega = \left\{(x,y,z) \in \Re^3 \mid z \in \left[-\frac{t}{2}, \quad \frac{t}{2}\right], (x,y) \in \mathrm{A} \subset \Re^2\right\}$$

onde *t* é a espessura da placa e *A* é a área da placa. O contorno de *A* é definido por *s*.

2. $\sigma_{33} = 0$.

3. $u_\alpha(x, y, z) = -z\,\theta_\alpha\,(x, y)$

4. $u_3(x, y, z) = w\,(x, y)$

A lei constitutiva admitida foi a de regime elástico linear.

O estudo do elemento finito de placa foi tratado como um problema de elasticidade plana, no qual foram aplicadas as simplificações do estado plano de tensões. Assim, a tensão normal ao plano da placa foi admitida como nula;

Os parâmetros elásticos para uma placa são: módulos de elasticidade longitudinal E_1e E_2, coeficientes de Poisson v_1e v_2 e módulos de elasticidade transversalG_{12}, G_{13} e G_{23}, todos em relação aos eixose_1, e_2 e e_3. Devido às condições de ortotropia, segundo *Lekhnistskii* deve ser imposta a condição:

$$E_1\,v_2 = E_2\,v_1$$

6.3 Hipóteses cinemáticas

Considere-se um elemento da' placa de espessura h_a e que tenha uma distância f_a do seu plano médio a um plano de referência adotado, paralelo a esse elemento de placa. Para um ponto P qualquer, pertencente à essa placa, adota-se um sistema de referência *xyz,* local, no plano médio da placa, com eixo *z* vertical e apontando para cima. Esse sistema de referência é chamado de SRL, do qual o plano médio da camada dista f . Adotamos ainda um segundo sistema de referência, denominado SRG, com eixos x_r y_r z_re com x_r e y_r coincidentes com o plano de referência adotado e paralelos a *x* e *y*. Os eixos *z* e z_r são alinhados e possuem o mesmo sentido.

Conforme pode-se observar na figura 6.2, a cinemática do deslocamento do ponto P é expressa pelo vetar *u* por em que

$$u = \begin{bmatrix} u(x_r, y_r) \\ v(x_r, y_r) \\ w(x_r, y_r) \end{bmatrix} \qquad (6.1)$$

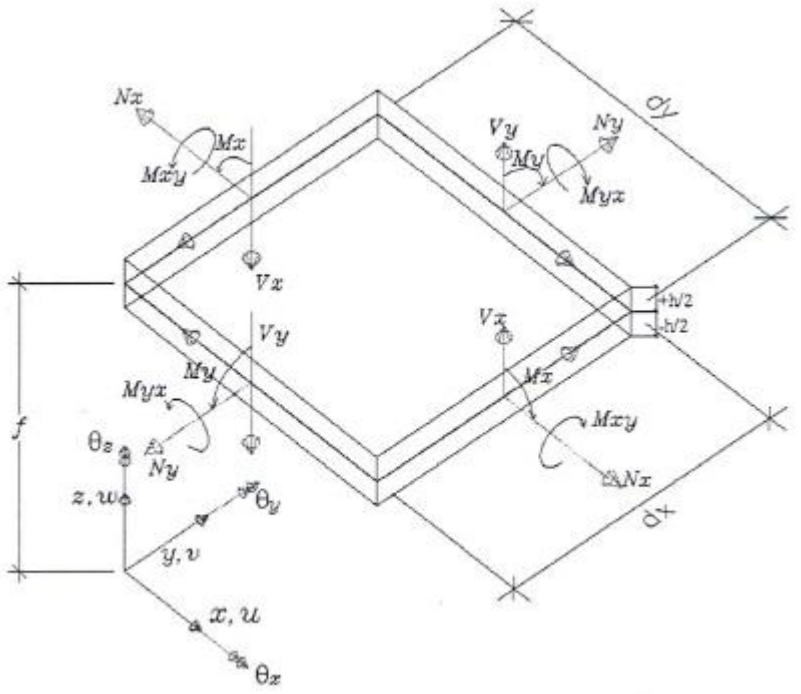

Figura 6.2: Sistema de coordenadas para um elemento de placa.

em que as componentes da translação, são

$$u\,(x,y,z) = \begin{bmatrix} u(x,y,z) = u(x_r,y_r) + \theta_y(z+f) \\ v(x,y,z) = v(x_r,y_r) - \theta_y(z+f) \\ w(x,y,z) = w(x_r,y_r) \end{bmatrix} \qquad (6.2)$$

Essas deformações *u* podem ser expressas em um vetor, definido por:

$$\{u\}^T = \{u\ v\ w\ \theta_x\ \theta_y\ \theta_z\} \qquad (6.3)$$

6.4 Deformações

As deformações, no ponto *P,* são calculadas em função dos deslocamentos de translação, expressos no sistema local. Como a formulação utilizada possibilita o posicionamento da uma placa em qualquer lugar do espaço, foi incorporada uma rotação θ_z, denominada absoluta. Essa rotação engloba o ponto *P* e se dá em torno do eixo perpendicular ao plano da placa. Com essa abordagem, pode-se utilizar essa formulação na aproximação de problemas de casca.

A essa rotação associou-se uma deformação fictícia é $\varepsilon\theta_z$, calculada no plano de referência. Assim, pode-se obter uma medida da deformação pela diferença entre θ_z e o

movimento de corpo rígido correspondente à rotação infinitesimal do plano de referência em torno do eixo *Z*. Essa diferença é dada pela expressão.

$$\theta_z(x_r, y_r) - \frac{1}{2}\left(\frac{\partial v\,(x, y, -f)}{\partial x} - \frac{\partial u\,(x, y, -f)}{\partial y}\right) \tag{6.4}$$

Com isso, podemos escrever as componentes do tensor de deformações em um vetor de dimensão 6x1, dado por:

$$\varepsilon = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{\theta z} \\ \varepsilon_{yz} \\ \varepsilon_{xz} \\ \varepsilon_{xy} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial u(x,y,z)}{\partial x} \\ \dfrac{\partial v(x,y,z)}{\partial y} \\ \theta_z(x_r, y_r) - \dfrac{1}{2}\left(\dfrac{\partial v(x,y,-f)}{\partial x} - \dfrac{\partial u(x,y,-f)}{\partial y}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial w(x,y,z)}{\partial x} - \dfrac{\partial u(x,y,z)}{\partial z}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial w(x,y,z)}{\partial y} - \dfrac{\partial v(x,y,z)}{\partial z}\right) \\ \dfrac{1}{2}\left(\dfrac{\partial u(x,y,z)}{\partial y} - \dfrac{\partial v(x,y,z)}{\partial x}\right) \end{bmatrix} \tag{6.5}$$

6.5 Tensões

A placa está submetida a um estado plano de tensões, garantido pela tensão normal nula na direção do eixo *z.* As fibras do elemento de placa estão alinhadas com os eixos *x* e *y* e *z,* e os valores das tensões em um ponto podem ser encontrados de acordo com o item C.5.2.

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{bmatrix} = \begin{bmatrix} \dfrac{E_x(\varepsilon_x + v_y\varepsilon_y)}{1 - v_x v_y} \\ \dfrac{E_y(\varepsilon_y + v_x\varepsilon_x)}{1 - v_x v_y} \\ 0 \\ G_{xy}\gamma_{xz} \\ G_{yz}\gamma_{yz} \\ G_{xy}\gamma_{xy} \end{bmatrix} \tag{6.6}$$

As constantes E_xe E_y são denominadas Módulo de Elasticidade Longitudinal. Os coeficientes v_x e v_y são denominados coeficientes de *Poisson.* E as constantes G_{xy}, G_{xz} e G_{xy} são denominadas Módulos dê Elasticidade Transversal. Essas constantes caracterizam as propriedades elásticas do material.

6.6 Esforços Solicitantes

Os esforços solicitantes por unidade de comprimento, cujos sentidos estão indicados na figura 6.1, atuantes nas faces de um elemento infinitesimal de placa, paralelo ao plano horizontal, são calculados como indicado nas seguintes equações:

1. Esforços de membrana por unidade de comprimento, que são as resultantes das integrações das tensões normais σ_x, σ_y e τ_{xy} ao longo da espessura *h* da placa. São distribuídos uniformemente ao longo da sua meia altura. N_xe N_y são normais às faces do elemento e N_{xy} tem direção tangente às faces.

$$N_x = \int_{-\frac{h}{2}}^{+\frac{h}{2}} \sigma_x dz \ , \qquad N_y = \int_{-\frac{h}{2}}^{+\frac{h}{2}} \sigma_y dz \ , \qquad N_{xy} = \int_{-\frac{h}{2}}^{+\frac{h}{2}} \tau_{xy} dz \tag{6.7}$$

2. Esforços associados à flexão por unidade de comprimento, que podem ser forças de cisalhamento por unidade de comprimento e momentos por unidades de comprimento.

- Forças cortantes V_x e V_y são as resultantes da integração das tensões de cisalhamento τ_{xz} e τ_{yz} ao longo da espessura h da placa. São equivalentes a forças cortantes por unidade de comprimento, distribuídas uniformemente ao longo da meia altura dos lados.

$$V_y = \int_{-\frac{h}{2}}^{+\frac{h}{2}} k\, \tau_{yz}\, dz \, , \; V_x = \int_{-\frac{h}{2}}^{+\frac{h}{2}} k\, \tau_{xz}\, dz \tag{6.8}$$

A constante k, que aparece na equação (6.8), é um coeficiente com a finalidade de corrigir a diferença entre a distribuição de tensões proposta e a distribuição da teoria da elasticidade. Nesse trabalho, adotou-se $k = \frac{5}{6}$ [14].

- Momentos por unidade de comprimento são as resultantes das integrais das tensões normais σ_x e σ_y e das tensões de cisalhamento τ_{xz} e τ_{yz} ao longo da espessura do elemento de placa, multiplicados pela distância do ponto de integração ao plano de referência. São denominados, respectivamente, momentos fletores M_x e M_y e momento volvente M_{xy}. Um ponto de coordenadas locais *(x,y,z)* está a uma distância *(f + z)* do plano de referência.

$$M_x = \int_{-\frac{h}{2}}^{+\frac{h}{2}} (z+f)\sigma_x dz \ , M_y = \int_{-\frac{h}{2}}^{+\frac{h}{2}} (z+f)\sigma_y dz \ , M_{xy} = \int_{-\frac{h}{2}}^{+\frac{h}{2}} (z+f)\tau_{xy} dz \quad (6.9)$$

Além dos esforços de momento, para possibilitar a representação do problema de cascas, acrescentou-se, ainda, um momento fictício $M\theta_z = Sh\varepsilon\theta_z$, que é associado à deformação fictícia correspondente a $\varepsilon\theta_z$. O valor da constante *S* é definido em $10^{-6}E_x$ [14].

6.7 Aplicação do Princípio dos Trabalhos Virtuais

Como descrito em (C.6), pelo Princípio dos Trabalhos Virtuais busca-se um equilíbrio entre o trabalho interno do corpo e o trabalho das forças externas ao corpo. Essa relação pode ser expressa por

$$W_{int} = W_{ext}$$

O trabalho interno pode ser determinado pelos esforços solicitantes e pelas variações dos deslocamentos essenciais, em relação ao plano de referência. Inclui-se nessa parcela a influência da base elástica, que é dada pelo produto entre a rigidez do solo (k_{mola}) e o deslocamento vertical *w* na direção *z* e suas variações. Dessa forma, tem-se a contribuição da base elástica diretamente na matriz de rigidez de cada elemento. A equação que representa o trabalho virtual interno é dada por

$$W_{int} = \int_A \left[N_x \frac{\partial \delta u\,(x,y,-f)}{\partial z} + N_y \frac{\partial \delta u\,(x,y,-f)}{\partial y} + N_{xy} \left(\frac{\partial \delta u\,(x,y,-f)}{\partial y} + \frac{\partial \delta v\,(x,y,-f)}{\partial x} \right) \right.$$
$$+ V_x \left(\frac{\partial \delta u\,(x,y,z)}{\partial z} + \frac{\partial \delta w\,(x,y,z)}{\partial y} \right) + V_y \left(\frac{\partial \delta u\,(x,y,z)}{\partial z} + \frac{\partial \delta w\,(x,y,z)}{\partial y} \right)$$
$$+ M_x \frac{\partial \delta \theta_y\,(x_r,y_r)}{\partial x} - M_y \frac{\partial \delta \theta_x\,(x_r,y_r)}{\partial x}$$
$$+ M_{xy} \left(\frac{\partial \delta \theta_y\,(x_r,y_r)}{\partial y} - \frac{\partial \delta x\,(x_r,y_r)}{\partial x} \right)$$
$$+ M_{\theta z} \left(\delta \theta_z\,(x_r,y_r) - \frac{1}{2} \left(\frac{\partial \delta v\,(x,y,-f)}{\partial x} - \frac{\partial \delta u\,(x,y,-f)}{\partial y} \right) \right) + k_{mola}$$
$$\left. \cdot w \quad \cdot \delta w \right] dxdy \qquad (6.10)$$

Os deslocamentos *u, v, w* são os indicados na equação (6.2). Utiliza-se as componentes da rotação absoluta da placa em torno dos eixos de referência x_r , y_r , z_r .

Os deslocamentos fundamentais usados em (6.10) foram agrupados em um vetor $\{u\}$, dado pela expressão (6.3). Por conveniência, adotou-se também um vetor $\{\delta u\}$, cujas componentes são as variações dos deslocamentos fundamentais, expresso por:

$$\{\delta u\}^T = \{\delta u\ \delta v\ \delta w\ \delta \theta_x\ \delta \theta_y\ \delta \theta_z\} \qquad (6.11)$$

As variáveis parciais envolvidas na equação do PTV são, em relação às variáveis independentes *x* e *y,* associadas aos eixos locais. Utilizam-se os operadores lineares $\frac{\partial}{\partial x}$ e $\frac{\partial}{\partial x}$, essas derivadas podem ser agrupadas em quatro novos vetores de ordem 6 x 1. Com isso, o integrando da equação (6.10) pode ser agrupado matricialmente à partir dos seguintes vetores

- Vetor dos deslocamentos $\{u\}$
- Vetor das variações dos deslocamentos $\{\delta u\}$
- Vetor das derivadas parciais dos deslocamentos em relação a $x \rightarrow \frac{\partial \{u\}}{\partial x}$

- Vetor das derivadas parciais dos deslocamentos em relação a $y \rightarrow \frac{\partial\{u\}}{\partial y}$

- Vetor das derivadas parciais das variações dos deslocamentos em relação a

$x \rightarrow \frac{\partial\{\delta u\}}{\partial x}$

- Vetor das derivadas parciais das variações dos deslocamentos em relação a

$y \rightarrow \frac{\partial\{\delta u\}}{\partial y}$

Com essa separação, e utilizando-se o programa *Mathematica,* pode-se resolver analiticamente o integrando da expressão (6.10) e agrupá-lo em nove matrizes, denominadas "Matrizes de Contribuição".

Aplicando essa simplificação, temos a expressão:

$$W_{int} = \int_A \left[\left(\frac{\partial\{\delta u\}^T}{\partial x}\right) K_{xx} \frac{\partial\{u\}}{\partial x} + \left(\frac{\partial\{\delta u\}^T}{\partial y}\right) K_{yy} \frac{\partial\{u\}}{\partial y} + \left(\frac{\partial\{\delta u\}^T}{\partial x}\right) K_{xy} \frac{\partial\{u\}}{\partial y} + \left(\frac{\partial\{\delta u\}^T}{\partial y}\right) K_{xy} \frac{\partial\{u\}}{\partial x} + \{\delta u\}^T B_{ox} \frac{\partial\{u\}}{\partial x} + \{\delta u\}^T B_{oy} \frac{\partial\{u\}}{\partial y} + \left(\frac{\partial\{\delta u\}^T}{\partial x}\right)^T B_{xo}\{u\} + \left(\frac{\partial\{\delta u\}^T}{\partial y}\right)^T K_{yo}\{u\} + \{\delta u\}^T B_{oo}\{u\} + k_{mola} \cdot w \cdot \delta w \right] dxdy \qquad (6.12)$$

6.8 Formulação para uma placa em uma posição qualquer no espaço

Nesse tópico, o objetivo é equacionar o problema de um elemento de placa com um desenvolvimento geométrico qualquer no espaço. Seguindo-se o desenvolvimento descrito no trabalho de Menezes & Dveloo[14], tomou-se um elemento de placa "a", em uma posição qualquer do espaço e como superfície de referência paralela ao plano do elemento. O plano

médio da camada tem excentricidade "f' em relação ao plano de referência, conforme ilustrado na figura6.3.

No plano médio dessa camada "a", definiu-se um par de vetores ε_1e ε_2, orientados nas direções de ortotropia do material da camada, sendo a direção ε_1 denominada "Direção das Fibras". Um terceiro eixo ε_3, normal à superfície da camada "a", é definido pelo produto vetorial entre ε_1 e ε_2. Temos ainda três vetores n_1, n_2 e n_3, definidos na superfície de referência, com $n_1 e\ n_2$ orientados paralelos a ε_1e ε_2, respectivamente, e n_3 como o produto vetorial entre n_1 e n_2 perpendicular ao plano de referência.

Três sistemas de coordenadas são definidos a partir da definição desses vetores. O primeiro é um sistema de referência global definido por $X_r Y_r Z_r$, com eixo Z_r orientado verticalmente para cima e aqui denominado SRG. O segundo sistema, denominado SRN é definido na superfície de referência por $n_1 n_2 n_3$, com n_3 orientado paralelamente e com o mesmo sentido do vetor ε_3. O terceiro sistema de coordenadas é denominado SRE é definido por $\varepsilon_1\ \varepsilon_2 \varepsilon_3$, no plano médio da camada "a".

Existem ainda três vetores α_1, α_2 e α_3, definidos no elemento finito de placa. Em cada ponto de integração do elemento é definido um vetor α_1 na direção do lado do elemento finito que contém esse ponto. A direção desse vetor α_1, é definida pela "lista de incidência" do elemento. Perpendicular a α_1 temos o vetor α_2, que tem orientação destrógira em relação a α_1. Os vetores α_1 e α_2 formam uma superfície paralela à superfície de referência, e o produto vetorial entre eles define o vetor α_3, que tem a mesma direção de n_3.

A representação, em relação ao sistema SRG, dos vetores $n_1, n_2\ e\ n_3$ e α_1, $\alpha_2\ e\ \alpha_3$ é dada por

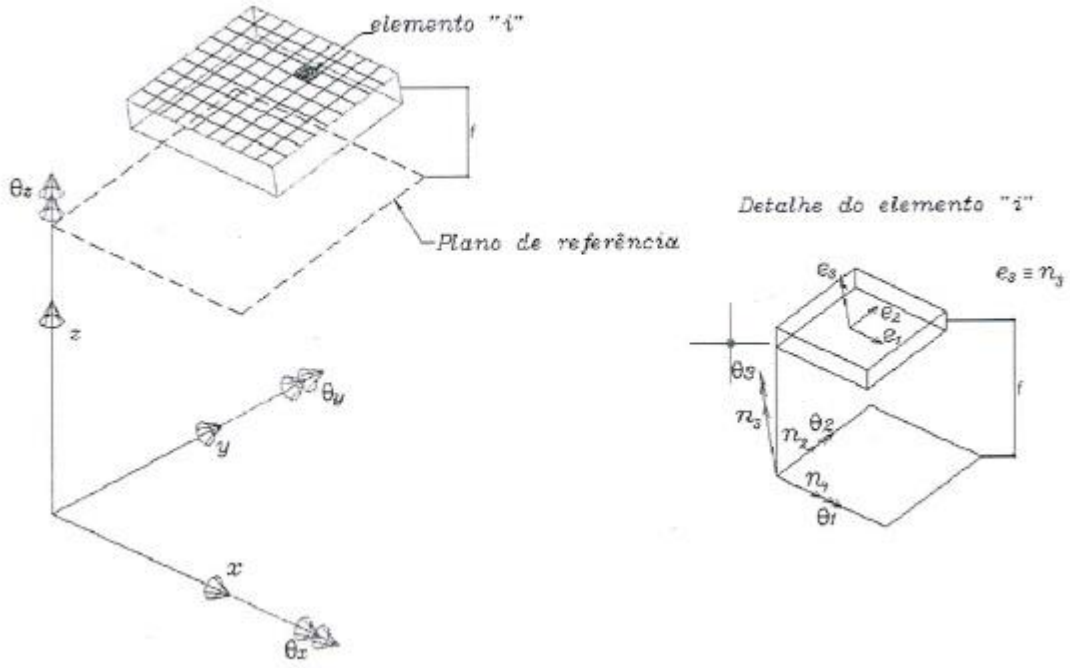

Figura 6.3: Sistema de coordenadas para um elemento de barra.

$$\begin{array}{ll} \alpha_1 = (a_{00}, a_{01}, a_{02}) & n_1 = (n_{00}, n_{01}, n_{02}) \\ \alpha_2 = (a_{10}, a_{11}, a_{12}) & n_2 = (n_{10}, n_{11}, n_{12}) \\ \alpha_3 = (a_{20}, a_{21}, a_{22}) & n_3 = (n_{20}, n_{21}, n_{22}) \end{array} \qquad (6.13)$$

6.8.1 Matriz de rotação

Um vetor t pode ser representado por componentes (t_x, t_y, t_z) em relação ao sistema de coordenadas globais *SRG* ou por componentes (t_{n1}, t_{n2}, t_{n3}) em relação ao sistema de coordenadas *SRN*. É possível definir uma matriz de rotação $[r]$ de ordem 3 x 3 que permite escrever as componentes (t_{n1}, t_{n2}, t_{n3}) em função das componentes (t_x, t_y, t_z). Os coeficientes dessa matriz de rotação são os cossenos diretores dos ângulos entre os vetores $n_1, n_2 \; e \; n_3$ e os versores que representam o *SRG*.

$$[r] = \begin{bmatrix} n_{00} & n_{01} & n_{02} \\ n_{10} & n_{11} & n_{12} \\ n_{20} & n_{21} & n_{22} \end{bmatrix} \begin{pmatrix} t_{n1} \\ t_{n2} \\ t_{n3} \end{pmatrix} = [r] \begin{pmatrix} t_{xr} \\ t_{yr} \\ t_{zr} \end{pmatrix} \quad \begin{pmatrix} t_{xr} \\ t_{yr} \\ t_{zr} \end{pmatrix} = [r]^T \begin{pmatrix} t_{n1} \\ t_{n2} \\ t_{n3} \end{pmatrix} \qquad (6.14)$$

6.8.2 Definição cinemática

Diferentemente do desenvolvimento apresentado para uma placa paralela à superfície de referência, para uma placa em uma posição qualquer em relação à superfície de referência, deve-se levar em consideração a existência de dois sistemas de coordenadas, devendo, portanto, o deslocamento *u* de um ponto *P* ser expresso nesses dois sistemas.

A translação do ponto O para o ponto O' pode ser representada no *SRN* por

$$[u_n] = \begin{bmatrix} u_n(n_1, n_2) \\ v_n(n_1, n_2) \\ w_n(n_1, n_2) \end{bmatrix} \qquad (6.15)$$

E pode ser representada no *SRG* por

$$[u] = \begin{bmatrix} u(x_r, y_r, z_r) \\ v(x_r, y_r, z_r) \\ w(x_r, y_r, z_r) \end{bmatrix} \qquad (6.16)$$

Da mesma forma, as componentes da rotação da placa podem ser expressas nos sistemas SRN e SRG, respectivamente, por

$$\begin{bmatrix} \theta_1 = \theta_1\,(n_1, n_2) \\ \theta_2 = \theta_2\,(n_1, n_2) \\ \theta_3 = \theta_3\,(n_1, n_2) \end{bmatrix} e \begin{bmatrix} \theta_x = \theta_x\,(x_r, y_r, z_r) \\ \theta_y = \theta_y\,(x_r, y_r, z_r) \\ \theta_z = \theta_z\,(x_r, y_r, z_r) \end{bmatrix} \qquad (6.17)$$

Com essas componentes, podem ser definidas as variáveis fundamentais do deslocamento, que são:

- A translação de OO'.
- As componentes da rotação do elemento de placa.

A representação dessas variáveis, na forma vetorial, nos sistemas *SRN* e *SRG,* é

$$\{u_n\}^T = \{u_n\ v_n\ w_n\ \theta_1\ \theta_2\ \theta_3\}$$

$$\{u\}^T = \{u\ v\ w\ \theta_x\ \theta_y\ \theta_z\} \qquad (6.18)$$

Cada um desses vetores é composto por três componentes de translação e três componentes de rotação, obtidas nos respectivos sistemas de referência. Pode-se também obter uma relação entre esses vetores, por meio da matriz de rotação *[r]*. Essa relação é dada por

$$\{u_n\}_{6x1} = \begin{bmatrix} [r]_{3x3} & 0 \\ 0 & [r]_{3x3} \end{bmatrix} \{u\}_{6x1}\ e\ \{u\}_{6x1} = \begin{bmatrix} [r]^T_{3x3} & 0 \\ 0 & [r]^T_{3x3} \end{bmatrix} \{u_n\}_{6x1} \qquad (6.19)$$

A translação de um ponto *P,* definido no plano médio da camada "a", pode ser descrita à partir da translação de OO' e da rotação absoluta do ponto *P* do elemento finito, em relação ao sistema SRE, por

$$\begin{gathered} u_e\ (e_1, e_2, e_3) = u_n\ (n_1, n_2) + \theta_2\ (e_3 + f) \\ v_e\ (e_1, e_2, e_3) = v_n\ (n_1, n_2) - \theta_1\ (e_3 + f) \\ w_e\ (e_1, e_2, e_3) = w_n\ (n_1, n_2) \end{gathered} \qquad (6.20)$$

6.8.3 Tensões

Nesse caso, temos as fibras orientadas na direção do eixo 1, e com isso, as direções de ortotropia são na direção dos eixos e_1 e e_2. Com essa orientação, temos os parâmetros elásticos E_1, E_2, v_1, v_2, G_{12} e G_{23}, relacionados aos eixos e_1 e e_2. A formulação descrita segue a hipótese do estado plano de tensões C.5.2, com tensão normal nula na direção do eixo *z*.

A tensão σ é representada pelo vetor

$$\sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{13} \\ \tau_{23} \\ \tau_{33} \end{bmatrix} = \begin{bmatrix} \dfrac{E_1(\varepsilon_1 + v_2\varepsilon_2)}{1 - v_1 v_2} \\ \dfrac{E_2(\varepsilon_2 + v_1\varepsilon_1)}{1 - v_1 v_2} \\ 0 \\ G_{13}\gamma_{13} \\ G_{23}\gamma_{23} \\ G_{12}\gamma_{12} \end{bmatrix} \qquad (6.21)$$

6.8.4 Deformações

As deformações em um ponto *P* são calculadas em função dos deslocamentos de translação do ponto *P,* no sistema *SRE,* e da rotação absoluta da camada *a* de referência. Já as componentes da rotação devem ser expressas no sistema *SRN.* Considerando-se também a rotação absoluta *B3* em torno do eixo perpendicular ao plano do elemento, estende-se a formulação à análise de cascas. A essa rotação, fez-se corresponder uma deformação fictícia θ_3, calculada no plano de referência de abscissa "-f" em relação ao eixo local e_3. Admite-se que essa deformação fictícia associa uma medida de deformação à diferença entre θ_3 e o movimento de corpo rígido correspondente à rotação infinitesimal do plano de referência em torno do eixo *z,* dada por:

$$\left(\frac{\partial v_n(n_1,n_2)}{\partial n_1}-\frac{\partial u_n(n_1,n_2)}{\partial n_2}\right)=\left(\frac{\partial v_e(e_1,e_2,-f)}{\partial e_1}-\frac{\partial u_e(e_1,e_2,-f)}{\partial e_2}\right) \qquad (6.22)$$

Assim, as deformações na camada "a", em um ponto *P,* são dadas por

$$\varepsilon=\begin{bmatrix}\varepsilon_1\\ \varepsilon_2\\ \varepsilon_{03}\\ \varepsilon_{13}\\ \varepsilon_{23}\\ \varepsilon_{12}\end{bmatrix}=\begin{bmatrix}\dfrac{\partial u_e(e_1,e_2,e_3)}{\partial e_1}\\ \dfrac{\partial v_e(e_1,e_2,e_3)}{\partial e_2}\\ \theta_3(n_1,n_2)-\dfrac{1}{2}\left(\dfrac{\partial v_e(e_1,e_2,-f)}{\partial e_1}-\dfrac{\partial u_e(e_1,e_2,-f)}{\partial e_2}\right)\\ \dfrac{1}{2}\left(\dfrac{\partial u_e(e_1,e_2,e_3)}{\partial e_3}+\dfrac{\partial w_e(e_1,e_2,e_3)}{\partial e_1}\right)\\ \dfrac{1}{2}\left(\dfrac{\partial v_e(e_1,e_2,e_3)}{\partial e_3}+\dfrac{\partial w_e(e_1,e_2,e_3)}{\partial e_2}\right)\\ \dfrac{1}{2}\left(\dfrac{\partial u_e(e_1,e_2,e_3)}{\partial e_2}+\dfrac{\partial v(e_1,e_2,e_3)}{\partial e_1}\right)\end{bmatrix} \qquad (6.23)$$

6.8.5 Esforços solicitantes em um elemento de placa

Os esforços solicitantes são determinados em função do sistema de coordenadas *SRE* de eixos $e_1e_2e_3$. Seguem a definição apresentada para o mesmo tópico do caso de placa

paralela ao sistema de referência, com as referências particulares ao sistema de referência utilizado. Assim, temos as seguintes expressões:

1. Esforços de membrana por unidade de comprimento, que são as resultantes das integrações das tensões normais $\sigma_1, \sigma_2\ e\ \tau_{12}$ ao longo da espessura *h* da placa. São distribuídos uniformemente ao longo da sua meia altura.

$$N_1 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} \sigma_1 de_3\ , N_2 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} \sigma_2 de_3\ , N_3 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} \tau_{12} de_3 \qquad (6.24)$$

2. Esforços associados à flexão por unidade de comprimento, que podem ser forças de cisalhamento por unidade de comprimento e momentos por unidades de comprimento.

3. Forças cortantes V_1e V_2, que são as resultantes da integração das tensões de cisalhamento τ_{13} e τ_{23} ao longo da espessura *h* da placa. São equivalentes a forças cortantes por unidade de comprimento, distribuídas uniformemente ao longo da meia altura dos lados.

$$V_1 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} k\tau_{13} de_3\ , V_2 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} k\tau_{23} de_3 \qquad (6.25)$$

Nesse trabalho, adotou-se $k = \frac{5}{6}$.

4. Momentos por unidade de comprimento são as resultantes das integrais das tensões normais σ_1 e σ_2e da tensão de cisalhamento τ_{12}, ao longo da espessura do elemento de placa, multiplicadas pela distância do ponto de integração ao plano de referência. São denominados, respectivamente, momentos fletores M_1 e M_2 e momento volvente M_{12}.

Os momentos também são esforços por unidade de comprimento ao longo dos lados.

$$M_1 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} (e_3 + f)\sigma_1 de_3\ , M_2 = \int_{-\frac{h}{2}}^{+\frac{h}{2}} (e_3 + f)\sigma_2 de_3\ , M_{12} = \int_{-\frac{h}{2}}^{+\frac{h}{2}} (e_3 + f)\tau_{12} de_3 \quad (6.26)$$

Além dos esforços de momento, para possibilitar a representação do problema de cascas, acrescentou-se, ainda, um momento fictício $M\theta_3 = Sh\varepsilon\theta_3$, que é associado à deformação fictícia correspondente a$\varepsilon\theta_3$, correspondente à rotação absoluta θ_3 na placa. O valor da constante S é definido em $10^{-6}E_1$.

6.8.6 Aplicação do Princípio dos Trabalhos Virtuais

Como apresentado no item *C.6,* pelo Princípio dos Trabalhos Virtuais, busca-se um equilíbrio entre o trabalho interno do corpo e o trabalho das forças externas ao corpo, expresso por:

$$W_{int} = W_{ext}$$

Como no item 6.7, também se considera aqui o efeito de uma base elástica, representado pelo produto entre uma rigidez K e o deslocamento verticalW_e, realizado na direçãoe_3. Com essa metodologia, consegue-se determinar a contribuição de uma base elástica, diretamente na rigidez de cada elemento.

Os esforços foram calculados no sistema de referência *SRE* da camada "a", sendo transferidos para a superfície de referência. Assim, os deslocamentos correspondentes devem ser os calculados nessa superfície de referência.

A equação que representa o trabalho virtual interno é

$$W_{int} = \int_A \left[N_1 \frac{\partial \delta u_e(e_1, e_2, -f)}{\partial n_1} + N_2 \frac{\partial \delta v_e(e_1, e_2, -f)}{\partial n_2} \right.$$
$$+ N_{12} \left(\frac{\partial \delta u_e(e_1, e_2, -f)}{\partial n_2} + \frac{\partial \delta v_e(e_1, e_2, -f)}{\partial n_1} \right)$$
$$+ V_1 \left(\frac{\partial \delta u_e(e_1, e_2, e_3)}{\partial n_3} + \frac{\partial \delta w_e(e_1, e_2, e_3)}{\partial n_1} \right)$$
$$+ V_2 \left(\frac{\partial \delta u_e(e_1, e_2, e_3)}{\partial n_3} + \frac{\partial \delta w_e(e_1, e_2, e_3)}{\partial n_2} \right) + M_1 \frac{\partial \delta \theta_2(n_1, n_2)}{\partial n_1}$$
$$- M_2 \frac{\partial \delta \theta_1(n_1, n_2)}{\partial n_2} + M_{12} \left(\frac{\partial \delta \theta_2(n_1, n_2)}{\partial n_2} - \frac{\partial \delta \theta_1(n_1, n_2)}{\partial n_1} \right)$$
$$+ M_{\theta 3} \left(\delta \theta_3 (n_1, n_2) - \frac{1}{2} \left(\frac{\partial \delta v_e(e_1, e_2, -f)}{\partial n_1} - \frac{\partial \delta u_e(e_1, e_2, -f)}{\partial n_2} \right) \right)$$
$$\left. + K_{mola} w_e \delta w_e \right] dn_1 dn_2 \qquad (6.27)$$

Na equação (6.27) os deslocamentos e suas variações estão expressos em função das direções dos eixos n_1, n_2 e n_3 e as derivadas de deslocamentos e variações estão expressas em relação às direções dos eixos n_1 e n_2. Essas grandezas podem ser colocadas em seis vetares, como apresentado no item 6.7.:

- Vetor das derivadas parciais dos deslocamentos em relação à direção

$n_1 \rightarrow \frac{\partial \{u_n\}}{\partial n_1}$

- Vetor das derivadas parciais dos deslocamentos em relação à direção

$n_2 \rightarrow \frac{\partial \{u_n\}}{\partial n_2}$

- Vetor das derivadas parciais das variações dos deslocamentos em relação à direção

$n_1 \rightarrow \frac{\partial \{\partial u_n\}}{\partial n_1}$

- Vetor das derivadas parciais das variações dos deslocamentos em relação à direção

$n_2 \rightarrow \frac{\partial \{\partial u_n\}}{\partial n_2}$

- Vetor de deslocamentos $\{u_n\}$

- Vetor de variação dos deslocamentos $\{\delta u_n\}$

Na implementação computacional, os deslocamentos foram escritos em coordenadas globais (sistema SRG) e as derivadas desses deslocamentos calculadas nas direções α_1e α_2, associadas ao lado do elemento que contém o nó do elemento finito.

Utilizando-se a matriz de rotação [R] definida no item 6.8.1, pode-se obter a troca de sistemas de referência, passando-se de $\{u_n\}$ para $\{u\}$ e de $\{\delta u_n\}$ para $\{\delta u\}$.

Aplicando essas considerações, pode-se simplificar a expressão do trabalho interno por:

$$W_{int} = \int_A \left[\left(\frac{\partial\{\delta u\}^T}{\partial\alpha_1}\right) K_{11} \frac{\partial\{u\}}{\partial\alpha_1} + \left(\frac{\partial\{\delta u\}^T}{\partial\alpha_2}\right) K_{22} \frac{\partial\{u\}}{\partial\alpha_2} + \left(\frac{\partial\{\delta u\}^T}{\partial\alpha_1}\right) K_{12} \frac{\partial\{u\}}{\partial\alpha_2} + \left(\frac{\partial\{\delta u\}^T}{\partial\alpha_2}\right) K_{21} \frac{\partial\{u\}}{\partial\alpha_1} + \{\delta u\}^T B_{o1} \frac{\partial\{u\}}{\partial a_1} + \{\delta u\}^T B_{o2} \frac{\partial\{u\}}{\partial a_2} + \left(\frac{\partial\{\delta u\}^T}{\partial\alpha_1}\right)^T B_{1o}\{u\} + \left(\frac{\partial\{\delta u\}^T}{\partial\alpha_2}\right)^T K_{2o}\{u\} + \{\delta u\}^T B_{000}\{u\} + K_{mola}\, w_e \delta w_e \right] da_1 da_2 \qquad (6.28)$$

Para a implementação computacional dessa formulação seguiu-se o mesmo caminho apresentado para o caso da placa paralela ao plano de referência. Foram utilizadas nove matrizes K_{ij} (i = 1,2 *e* *j* = 1,2) e B_{0j} (j = 1,2), B_{j0} (*j* = 1,2) e B_{000}, calculadas a partir das nove matrizes de contribuição utilizadas no caso em que a placa estava posicionada na horizontal. As matrizes são:

$$K_{11} = [r]^T \left(K_{xx} \frac{\partial\alpha_1}{\partial n_1}\frac{\partial\alpha_1}{\partial n_1} + K_{yy} \frac{\partial\alpha_1}{\partial n_2}\frac{\partial\alpha_1}{\partial n_2} + K_{xy} \frac{\partial\alpha_1}{\partial n_1}\frac{\partial\alpha_1}{\partial n_2} + K_{yx} \frac{\partial\alpha_1}{\partial n_2}\frac{\partial\alpha_1}{\partial n_1} \right) [r]$$

$$K_{22} = [r]^T \left(K_{xx} \frac{\partial\alpha_2}{\partial n_1}\frac{\partial\alpha_2}{\partial n_1} + K_{yy} \frac{\partial\alpha_2}{\partial n_2}\frac{\partial\alpha_2}{\partial n_2} + K_{xy} \frac{\partial\alpha_2}{\partial n_1}\frac{\partial\alpha_2}{\partial n_2} + K_{yx} \frac{\partial\alpha_2}{\partial n_2}\frac{\partial\alpha_2}{\partial n_1} \right) [r]$$

$$K_{12} = [r]^T \left(K_{xx} \frac{\partial\alpha_1}{\partial n_1}\frac{\partial\alpha_2}{\partial n_1} + K_{yy} \frac{\partial\alpha_1}{\partial n_2}\frac{\partial\alpha_2}{\partial n_2} + K_{xy} \frac{\partial\alpha_1}{\partial n_1}\frac{\partial\alpha_2}{\partial n_2} + K_{yx} \frac{\partial\alpha_1}{\partial n_2}\frac{\partial\alpha_2}{\partial n_1} \right) [r]$$

$$K_{21} = [r]^T \left(K_{xx} \frac{\partial \alpha_2}{\partial n_1} \frac{\partial \alpha_1}{\partial n_1} + K_{yy} \frac{\partial \alpha_2}{\partial n_2} \frac{\partial \alpha_1}{\partial n_2} + K_{xy} \frac{\partial \alpha_2}{\partial n_1} \frac{\partial \alpha_1}{\partial n_2} + K_{yx} \frac{\partial \alpha_2}{\partial n_2} \frac{\partial \alpha_1}{\partial n_1} \right) [r] \quad (6.29)$$

$$B_{01} = [r]^T \left(B_{0x} \frac{\partial \alpha_1}{\partial n_1} + B_{0y} \frac{\partial \alpha_1}{\partial n_2} \right) [r]$$

$$B_{10} = [r]^T \left(B_{x0} \frac{\partial \alpha_1}{\partial n_1} + B_{y0} \frac{\partial \alpha_1}{\partial n_2} \right) [r]$$

$$B_{02} = [r]^T \left(B_{0x} \frac{\partial \alpha_2}{\partial n_1} + B_{0y} \frac{\partial \alpha_2}{\partial n_2} \right) [r]$$

$$B_{20} = [r]^T \left(B_{x0} \frac{\partial \alpha_2}{\partial n_1} + B_{y0} \frac{\partial \alpha_2}{\partial n_2} \right) [r]$$

$$B_{000} = [r]^T (B_{00}) [r]$$

6.8.7 Implementação computacional

As formulações foram implementadas no ambiente de elementos finitos *pz*. Nesse ambiente, o usuário encontra um espaço de interpolação contínuo e um procedimento de integração sobre os elementos. Para implementar um sistema de equações diferenciais, o usuário deve criar um "processo" que - dados os valores das funções de forma e das suas derivadas, a localização do ponto de integração e as direções dos eixos jacobianos - calcula as contribuições para a matriz de rigidez do elemento e para o vetor de carga. O restante da implementação já está disponível no ambiente. Ainda é permitido ao usuário escolher o tipo de solução de sistema mais conveniente.

A matriz de rigidez e o vetor de ações são montados pela contribuição dos pontos de integração. As componentes do vetor de deslocamentos, do vetor de variação dos deslocamentos e as suas derivadas parciais de direções α_1 e α_2 são discretizadas por meio das funções de forma geradas segundo o método dos elementos finitos. No contexto do sistema de

equações diferenciais de placa (ou casca) em estudo, são geradas seis equações para cada função de interpolação.

Para o cálculo da contribuição de um dado ponto de integração para a matriz de rigidez, foram implementados no ambiente *PZ* os seguintes passos:

- Cálculo da direção das fibras em função da localização do ponto de integração;
- Cálculo do produto entre os eixos do elemento e as direções das fibras;
- Cálculo das matrizes de contribuição;
- Cálculo da contribuição para a matriz de rigidez.

O número de variáveis associadas a um nó é 6. Enumerando as funções de forma φ com índices i e *j* e as variáveis de equações diferenciais com α e β, a contribuição para a matriz de rigidez será:

$$K_{(6_i+\alpha,6_j+\beta)} += \omega \cdot \left[(K_{11})_{\alpha\beta} \frac{\partial\varphi_i}{\partial a_1}\frac{\partial\varphi_j}{\partial a_1} + (K_{21})_{\alpha\beta} \frac{\partial\varphi_i}{\partial a_2}\frac{\partial\varphi_j}{\partial a_1} + (K_{12})_{\alpha\beta} \frac{\partial\varphi_i}{\partial a_2}\frac{\partial\varphi_j}{\partial a_2} + (K_{22})_{\alpha\beta} \frac{\partial\varphi_i}{\partial a_2}\frac{\partial\varphi_j}{\partial a_1} + (B_{01})_{\alpha\beta\varphi i} \frac{\partial\varphi_i}{\partial a_1} + (B_{02})_{\alpha\beta\varphi i} \frac{\partial\varphi_i}{\partial a_2} + (B_{10})_{\alpha\beta\varphi j} \frac{\partial\varphi_i}{\partial a_1} + (B_{20})_{\alpha\beta\varphi j} \frac{\partial\varphi_i}{\partial a_2} + (B_{000})_{\alpha\beta}\partial\varphi_i\partial\varphi_j \right] \quad (6.30)$$

em que $\alpha = 0 \rightarrow 6$ e $\beta = 0 \rightarrow 6$. Os índices i e *j* variam de acordo com o número de funções de forma e ω é a função *"peso de integração"*

- A contribuição para o vetor de carga envolve apenas os valores das funções de forma. Considerando que as forças por unidade de área são (f_x, f_y, f_z), as contribuições para o vetor de carga serão:

$$\begin{aligned} Ef_{\lambda i} &+= \omega \cdot f_x \\ Ef_{(\lambda i+1)} &+= \omega \cdot f_y \\ Ef_{(\lambda i+2)} &+= \omega \cdot f_z \end{aligned} \quad (6.31)$$

7 Metodologia

A ideia principal do estudo é desenvolver funções que automatizem a criação de elementos estruturais, a atribuição das características dos materiais, a aplicação dos carregamentos e das ações e a determinação das vinculações nos apoios e restrições de deslocamento e giro nos nós da estrutura. Além disso, pretende-se retirar do AutoCAD as informações geométricas de cada elemento, necessárias para a montagem das malhas, geométrica e computacional, para, em seguida, executar o processo de cálculo e a posterior visualização dos resultados.

7.1 Separação ObjectARX/ObjectDBX

A sigla ARX é uma alusão a "AutoCAD Runtime Extension", e os arquivos compilados segundo essa estrutura recebem a extensão ".arx". Nesse arquivo, ou projeto ARX, é delineada a interface com o usuário. Em outras palavras, nesse arquivo ou projeto, se dá a disponibilização dos comandos do AutoCAD.

Já o projeto DBX é bem mais complexo, pois nele estão definidos os métodos e funções que definem o comportamento e criação das novas entidades. A sigla DBX refere-se a "DataBase Extension", no que tange à ampliação da quantidade de entidades disponíveis no acervo do AutoCAD. Na medida em que se criam novas entidades, a complexidade e extensão do código aumentam acentuadamente. Esse é um dos motivos de se separar a interface com o usuário da extensão do banco de dados. Embora seja uma separação conceitual, ela é necessária para que se entenda que não se deve confundir a interação das entidades com o AutoCAD com a interação das entidades com o usuário.

Seguindo essa estrutura, dividiu-se o projeto em três partes, que são:

1. Um módulo ARX para agrupar os comandos de geração das entidades geométricas;

2. Um módulo DBX para agrupar os objetos/entidades geométricas;

3. Um módulo ARX para agrupar os comandos para realização da análise estrutural.

Essa divisão foi necessária, pois cada módulo apresenta uma estrutura diferente. Além disto, visou agrupar procedimentos semelhantes em grupos distintos. De fato, podemos tratá-los como diferentes pacotes, na medida em que são bibliotecas dinâmicas que podem ser importadas pelo AutoCAD separadamente.

7.2 Projeto DBX Geométrico

No projeto de extensão".dbx" foram introduzidas as novas classes para criação dos elementos geométricos.

Como esse programa focaliza em estruturas reticuladas, iniciou-se pelo desenvolvimento de entidades básicas que são: o nó e o elemento de barra.

Implementaram-se as classes TPZnode, TPZbeam, TPZPlate, TPZReactor e TPZGemprop a partir da classe AcDbEntity. Por ser a classe que origina as entidades gráficas do AutoCAD, a sua especialização garante aos objetos das classes derivadas as mesmas características fundamentais dos objetos residentes no programa AutoCAD.

7.2.1 Classe TPZnode

O nó, como elemento de ligação entre os elementos de barra, não tem função estrutural, mas é imprescindível para definir a continuidade das malhas e as condições de contorno. Cabe ainda ao nó guardar as referências para as barras que a ele se conectam, bem como gerenciar essa informação para armazenamento em disco.

Além de possuir essas características internas, um nó possui ainda representação gráfica, sendo identificado por um círculo desenhado na tela, cuja dimensão e cores podem ser configuradas pelo usuário.

A uma entidade do tipo nó pode-se associar uma carga concentrada. Dado que a carga pontual é composta de uma direção, representada por um vetor unitário, e o valor da ação, pode-se definir um carregamento e uns direção qualquer.

As condições de contorno da malha (ou seja, a forma como a malha estrutural se vincula ao meio) também são definidas no nó. Segundo esse, são quatro os tipos de vinculação possíveis:

- Nó totalmente móvel;
- Apoio fixo segundo a direção de um vetor, e móvel no plano ao qual o vetar é normal;
- Apoio móvel segundo a direção de um vetor, e fixo no plano ao qual o vetor é normal;
- Apoio fixo.

Além dessas quatro possibilidades de vinculação, o nó ainda guarda as condições de impedimento de giro, que podem ser:

- Giro liberado em todas as direções;
- Giro impedido segundo um eixo dado pela direção de um vetor e livre nas outras duas direções ortonormais;
- Giro livre segundo um eixo dado pela direção de um vetor e impedido nas outras duas direções ortonormais;

- Giro totalmente impedido, ou seja, engaste.

Os Principais métodos da classe TPZnode são:

Construtor simples, indispensável ao AutoCAD

- *TPZnode()*

Construtor já especificando a coordenada do nó

- TPZnode(AcGePoint3d Point);

Destrutor simples, indispensável ao AutoCAD

- virtual ~TPZnodeO;

Funções virtuais do AutoCAD reimplementadas

Método virtual imprime em tela um conjunto de características de um objeto do tipo TPZnode

- *virtual void list() const;*

Método virtual utilizado pelo AutoCAD para excluir uma entidade

- *virtual Acad::ErrorStatus subErase(Adesk::Boolean erasing);*

Método que escreve em disco as características de um objeto do tipo TPZnode

- *virtual Acad::ErrorStatus dwgOutFields(AcDbDwgFiler* pFiler) const;*

Método que escreve lê do disco as características de um objeto do tipo TPZnode

- *virtual Acad::ErrorStatus dwglnFields(AcDbDwgFiler* pFiler);*

Método que representa em tela um objeto do tipo TPZnode

- *virtual Adesk::Boolean worldDraw(AcGiWorldDraw* mode);*

Método que separa as diferentes entidades que formam um objeto TPZnode

- *virtual Acad::ErrorStatus explode(AcDb VoidPtrArray& entitySet) const;*

Método que aplica uma transformação a um objeto TPZnode

- *virtual Acad::ErrorStatus transformBy(const AcGeMatrix3dξ xform);*

Método que atribui a um objeto TPZnode a funcionalidade move do método Stretch do AutoCAD

- *virtual Acad::ErrorStatus moveStretchPointsAt(const AcDblntArrayξ índices, const Ac*Ge *Vector3dξ offset);*

Método que captura um objeto TPZnode e disponibiliza para 'esse objeto as funcionalidades do método Stretch do AutoCAD

- *virtual Acad:: ErrorStatus getStretchPoints (AcGePoint3dArray ξ stretchPoints) const;*

Método que atribui a um *grip* de um objeto TPZnode a funcionalidade move do método Stretch do AutoCAD

- *virtual Acad::ErrorStatus moveGripPointsAt(const AcDblntArray ξ indices, const Ac*Ge *Vector3dξ offset);*

Método que captura um grip de um objeto TPZnode e disponibiliza para esse objeto as funcionalidades do método Stretch do AutoCAD

- *virtual Acad::ErrorStatus getGripPoints(AcGePoint3dArray ξ gripPoints, AcDblntArrayξ osnapModes, AcDblntArrayξ geomlds) const;*

Método que atribui a um grip de um objeto TPZnode a funcionalidade move do método Osnap do AutoCAD

- *virtual Acad::ErrorStatus getOsnapPoints(AcDb::OsnapMode osnapMode, int gsSelectionMark, const AcGePoint3dξ pickPoint, const AcGePoint3dξ lastPoint, const Ac-GeMatrix3dξ viewXform, AcGePoint3dArraξ snapPoints, AcDbIntArrayξ geomlds) const;*

Métodos e variáveis públicas da classe TPZnode:

Método que retoma o índice do material associado a um nó

- *int MaterialIndex();*

Método que atribui a um nó o índice de um material

- *void SetMaterialIndex(int mat);*

Método que informa se um nó tem uma condição de contorno ou não. Retoma *True* se tem condição de contorno

- *bool HasBoundaryCondition();*

Função de acesso externo, utilizado por um TPZnode ou TPZbeam para adicionar de um Id de um TPZbeam a um nó

- *void addElementId(const AcDbObjectId newld);*

Função de acesso externo, utilizado por um TPZnode ou TPZbeam para remover um Id de um TPZbeam de um nó

- *Adesk::Boolean removeElementId(const AcDbObjectId remld);*

Método que retoma duas listas de Ids, a partir da variável fElementId interna da classe TPZnode, que são:

1. ValidIdList: lista dos Ids de TPZbeams válidos

2. LostIdList: lista dos Ids de TPZbeams que constam como apagados

- *void rclldList(AcDbObjectIdArrayξ ValidldList, AcDbObjectIdArrayξ LostIdL1*

Método que retoma as coordenadas de um TPZnode

- *AcGePoint3d getNodePoint() const;*

Método que inicializa a posição do TPZnode

- *void setNodePoint(AcGePoint3d Point);*

Método que adiciona um reator a um nó

- *Acad::ErrorStatus addMeAReactor();*

Método que atribui restrições de deslocamento a um nó

- *void setConstraints(const int FixedDisplacements, const AcGeVector3d DisplacementDir);*

Método que retoma as restrições de deslocamento de um nó

- *void getConstraints(int ξ FixedDisplacements, AcGeVector3d ξ DisplacementDir) const;*

Método que atribui restrições de deslocamento e giro a um nó

- *void setRotConstr(const int FixedRotations, const AcGe Vector3d RotationAxis);*

Método que retoma as restrições de giro de um nó

- *void getRotConstr(int &FixedRotations,* AcGe*Vector3d &RotationAxis)const;*

Método que adiciona uma carga pontual a um nó

- *void setLoad(const AcGeVector3d LoadVector);*

Método que retoma o valor de uma carga aplicada em um nó

- *void getLoad(* AcGe*Vector3d & LoadVector)const;*

Método que exporta as propriedades de um nó. Essas propriedades são:

1. Restrições de deslocamento;
2. Restrições de giro;
3. Vetor que indica a direção de um carregamento aplicado em um nó

- *void exportProperties(AcGeMatrix2d & constr, AcGeMatrix2d & rotConstr,* AcGe Vec*tor3d &LoadVector);*

Método que retoma uma variável que indica a existência de um carregamento em um nó. Caso exista um carregamento esse método retoma o valor TRUE.

- *Adesk::Boolean HaveLoad();*

Métodos e variáveis privadas da classe TPZnode

Variável que armazena as coordenadas de um TPZnode

- *AcGePoint3d fNodePoint;*

Variável que armazena uma lista dos Ids dos elementos do tipo TPZbeam de um TPZnode

- *AcDbObjectIdArray fElementIds;*

Inteiro que indica o tipo de restrição de deslocamento

- *int fFixedDisplacements;*

Vetor que contem a direção de um deslocamento impedido

- AcGe*Vector3d fDisplacementDir;*

Inteiro que indica o tipo de restrição de deslocamento

- *int fFixedRotations;*

Vetor que contem uma direção de giros impedidos

- *AcGeVector3d fRotationAxis;*

Vetor que contém uma carga aplicada em um nó

- *AcGeVector3d fLoadVector;*

Variável que armazena o índice de um material associado a um TPZnode

- *int fMaterialIndex;*

Operadores da classe TPZnode

- *void *operator new[](unsigned nSize)* { *return 0;* }

- *void operator delete[](void *p)* {};

- *void *operator new[](unsigned nSize, const char *file, int tine)* { *return 0;* }

7.2.2 Classe TPZbeam

Para a criação de um objeto da classe *TPZbeam* são necessários dois objetos da classe TPZnode. Embora um objeto da classe TPZnode possa ter muitas referências de objetos do tipo TPZbeam, um objeto do tipo TPZbeam pode ter apenas referência a dois objetos do tipo TPZnode.

A representação gráfica da barra é a mais complexa do programa. A barra, no momento da criação, é uma linha que une os dois nós. Assim que as características físicas são atribuídas a ela, a barra tem a cor de sua representação alterada e surge no meio dessa um contorno representando a sua seção transversal. A adoção dessa estratégia dispensa a necessidade da representação completa da viga. Como se não bastasse, o desenho da seção transversal auxilia no entendimento da malha tridimensional da estrutura.

A barra também admite carregamento, mas nesse caso distribuído de modo trapezoidal, triangular ou uniforme. A orientação do carregamento pode se dar segundo a direção de um vetor global ou mesmo segundo uma das três direções principais da barra. Esse último caso permite que a orientação do carregamento seja alterada juntamente com a alteração da orientação da barra.

Pelo fato da barra depender da existência de dois nós, as ferramentas de edição da barra não precisaram ser criadas. Assim, a única forma de se alterar a orientação da barra é

movimentar algum de seus nós. Isso torna a relação de dependência entre nó e barra mais evidente, e o conjunto, mais robusto.

Os principais métodos e variáveis da classe TPZbeam são:

Construtor simples, indispensável ao AutoCAD

- *TPZbeam();*

Construtor já especificando os Id's dos nós da barra

- *TPZbeam(const AcDbObjectId FirstNodeId, const AcDbObjectId SecondNodeId);*

Destrutor simples, indispensável ao AutoCAD

- *virtual ~TPZbeam();*

Funções virtuais do AutoCAD reimplementadas

Método virtual imprime em tela um conjunto de características de um objeto do tipo TPZnode

- *virtual void list() const;*

Método virtual utilizado pelo AutoCAD para excluir uma entidade

- *virtual Acad::ErrorStatus subErase(Adesk::Boolean erasing);*

Método que aplica uma tranformação a um objeto TPZnode

- *virtual Acad::ErrorStatus transformBy(const AcGeMatrix3dξ xform);*

Método que escreve em disco as características de um objeto do tipo TPZnode

- *virtual Acad::ErrorStatus dwgOutFields(AcDbDwgFiler* pFiler) const;*

Método que escreve lê do disco as características de um objeto do tipo TPZnode

- *virtual Acad::ErrorStatus dwglnFields(AcDbDwgFiler* pFiler);*

Método que representa em tela um objeto do tipo TPZnode

- *virtual Adesk::Boolean worldDraw(AcGiWorldDraw* mode);*

Método que separa as diferentes entidades que formam um objeto TPZnode

- *virtual Acad::ErrorStatus explode(AcDb VoidPtrArrayξ entitySet) const;*

Métodos e variáveis públicas da classe TPZbeam:

Variáveis que armazenam, respectivamente, os resultados dos momentos e cortantes em uma barra e as posições nas quais é feito o pós processamento em uma barra

- *ads- real fMomento[3][20], fCortante[3][20], faxes[5}[9];*

Variáveis que armazenam os Id's dos nós de uma barra

- *AcDbObjectId fFirstNodeId, fSecondNodeId;*

Método que atribui a uma barra os valores do módulo de elasticidade E e o coeficiente Ni do material.

- *void setbeampar(const ads _real E,const ads _real Ni);*

Método que retoma os valores do módulo de elasticidade E e o coeficiente Ni do material de uma barra.

- *void getbeampar(ads _real E,ads _real Ni)const;*

Método que atribui a uma barra as características geométricas H, B e vetor que indica a orientação da seção transversal

- *void setProperties(const ads_real Height, const ads_real Width, const AcGe Vector3d MainSectionDir, const ads_real E, const ads_real Mi);*

Método que retoma as características geométricas H, B e vetor que indica a orientação da seção transversal de uma barra

- *void getProperties(ads_real ξ Height, ads_real ξ Width, AcGe Vector3d ξMainSectionDir, ads_real ξE, ads_real ξMi)const;*

Método que retoma o vetor que indica a orientação da seção transversal de uma barra

- *void getMainSectionDir(AcGe Vector3d ξMainSectionDir)const;*

Método que atribui a uma barra um carregamento distribuído e a orientação vetorial do carregamento

- *void setLoad(const ads_real Load[2], const AcGe Vector3d LoadDir, const Adesk::Boolean LocalCS);*

Método que retoma um carregamento distribuídos e a orientação vetorial do carregamento de uma barra

- *void getLoad(ads_real Load[2],AcGe Vector3d ξ LoadDir,Adesk::Boolean ξLocalCS)const;*

Método que retoma um carregamento distribuído e a orientação vetorial global do carregamento de uma barra

- *void getGlobalLoad(ads_real Load[2],AcGe Vector3d &LoadDir)const;*

Método que atribui a uma barra os valores de momento e força cortante resultantes do cálculo

- *void setMQ(const ads_real M[3][20],const ads_real Q[3][20],const ads_real axes[5][9]);*

Método que retorna os valores de momento e força cortante resultantes do cálculo, associados a uma barra

- *void getInMQ(ads_real M[3][20], ads_real Q[3][20], ads_ real axes[5][9])const;*

Método que retorna uma variável que indica a existência de propriedades geométricas em uma barra. Caso existam propriedades geométricas, esse método retoma o valor TRUE.

- *Adesk::Boolean amIFullFilled();*

Método que retorna uma variável que indica a existência de um carregamento distribuído em uma barra. Caso exista um carregamento, esse método retoma o valor TRUE.

- *Adesk::Boolean HaveLoad();*

Método que retorna uma variável que indica a existência de valores de momento associados a uma barra. Caso exista, esse método retoma o valor TRUE.

- *Adesk::Boolean HaveMoment();*

Método que retorna o vetor que indica a orientação de uma barra

- *Acad::ErrorStatus getBeamAxis(AcGe Vector3d &BeamAxis)const;*

Método que retorna o Id do primeiro nó de uma barra

- *AcDbObjectId getFirstNodeId() const;*

Método que retorna o Id do primeiro nó de uma barra

- *AcDbObjectId getSecondNodeId() const;*

Métodos e variáveis privadas da classe TPZnode

Variável que armazenam, respectivamente, a altura e a largura de uma seção, um carregamento distribuído de uma barra e os valores do módulo de elasticidade E e o coeficiente Ni de uma barra

- *Ads_real fWidth, fHeight, fLoad[2], fE, fMi;*

Variável que armazena um vetor que indica a orientação da seção transversal de uma barra

- *AcGe Vector3d fMainSectionDir;*

Variável que armazena um vetor que indica a direção global de um carregamento distribuído

- *AcGe Vector3d fLoadDir;*

Variável que armazena um vetor que indica a direção local de um carregamento distribuído

- *Adesk::Boolean fLocalCS;*

Método que associa o Id do primeiro nó de uma barra

- *void setFirstNodeId(const AcDbObjectId FirstNodeId);*

Método que associa o Id do segundo nó de uma barra

- *void setSecondNodeId(const AcDbObjectId SecondNodeId);*

Método que retoma as coordenadas de um ponto pertencente a uma barra

- *Acad::ErrorStatus getNodePoint(const AcDbObjectId NodeId, AcGePoint3d ξ Point) const;*

Operadores da classe TPZbeam

- *void *operator new[](unsigned nSize)* { *return 0;* }
- *void operator delete[] (void *p)* {};

- *void *operator new[](unsigned nSize, const char *file, int fine)* { *return 0;* }

7.2.3 Classe TPZplate

Para a criação de um objeto da classe *TPZplate* são necessários quatro objetos da classe TPZnode. Embora um objeto da classe TPZnode possa ter muitas referências de objetos do tipo TPZbeam, um objeto do tipo TPZplate pode ter apenas referência a quatro objetos do tipo TPZnode.

A representação gráfica da placa compreende a representação do seu contorno e dos quatro nós dos vértices da placa. Para a placa não existe a representação da seção transversal, tal como existe para a barra.

- *virtual Acad::ErrorStatus explode(AcDb VoidPtrArrayξ entitySet) const;*

Método que aplica uma transformação a um objeto TPZplate

- *virtual Acad::ErrorStatus transformBy(const AcGeMatrix3dξ xform);*

Métodos e variáveis públicas da classe TPZplate:

Método que atribui o Id de um nó ao primeiro vértice â um objeto do tipo TPZplate

- *void setFirstNodeId(const AcDbObjectId FirstNodeId);*

Método que atribui o Id de um nó ao segundo vértice d um objeto do tipo TPZplate

- *void setSecondNodeId(const AcDbObjectId SecondNodeId);*

Método que atribui o Id de um nó ao terceiro vértice de um objeto do tipo TPZplate

- *void setThirdNodeId(const AcDbObjectId ThirdNodeId);*

Método que atribui o Id de um nó ao quarto vértice de um objeto do tipo TPZplate

- *void setFourNodeId(const AcDbObjectId FourNodeId);*

Variáveis que armazenam os Id's dos nós dos vértices de um objeto do tipo TPZplate

- *AcDbObjectId obid, fFirstNodeId, fSecondNodeId, fThirdNodeId, fFourNodeId;*

Método que retoma o Id do nó do primeiro vértice de um objeto do tipo TPZplate

- *AcDbObjectId getFirstNodeId() const;*

Método que retorna o Id do nó do segundo vértice de um objeto do tipo TPZplate

- *AcDbObjectId getSecondNodeId() const;*

Método que retoma o Id do nó do terceiro vértice de um objeto do tipo TPZplate

- *AcDbObjectId getThirdNodeld() const;*

Método que retoma o Id do nó do quarto vértice de um objeto do tipo TPZplate

- *AcDbObjectId getFourNodeld() const;*

Método que retoma uma variável que indica a existência de um carregamento em um TPZplate. Caso exista um carregamento esse método retoma o valor TRUE.

- *Adesk::Boolean amIFullFilled();*

Métodos e variáveis privadas da classe TPZnode

Variáveis que armazenam a espessura largura de uma placa e um vetor que armazena os carregamentos aplicados em uma placa.

- *Ads_real fWidth, fHeight, fLoad[2];*

Variável que armazena um vetor que indica a direção de um carregamento aplicado em uma placa.

- *AcGe Vector3d fLoadDir;*

Variável que armazena um vetor que indica a direção da seção de uma placa.

- *AcGe Vector3d fMainSectionDir;*

Variável que armazena um vetor que indica a direção do eixo local de uma placa.

- *Adesk::Boolean fLocalCS;*

Operadores da classe TPZplate

- *void *operator new[](unsigned nSize)* { *return 0;* }
- *void operator delete[] (void *p)* {};
- *void *operator new[](unsigned nSize, const char *file, int line)* { *return O;* }

7.2.4 Classe TPZreactor

Introduziu-se a estrutura de reatores para garantir que as alterações aplicadas ao nó sejam integralmente aplicadas à barra. Isto se justifica pela dependência da barra em relação ao nó. Isto faz com que a barra tenha que respeitar as alterações que o AutoCAD e o usuário impõem ao nó.

No AutoCAD o reator é uma entidade que é registrada não no banco de dados de entidades de desenho, mas no *Named Object Dictionary,* ou seja, no dicionário de entidades nomeadas. Isso ocorre porque o reator não é selecionado via interface gráfica com o mouse, mas pelo seu nome. Na verdade, o reator nem mesmo apresenta representação gráfica. Por não apresentar comportamento de entidade de desenho e não ter representação gráfica, a classe TPZreactor é derivada da classe AcDbObject, e não da AcDbEntity. No momento do registro do reator, é necessário informar a qual entidade o reator estará associado.

Os principais métodos e variáveis da classe TPZreactor são:

Construtor simples, indispensável ao AutoCAD

- *TPZreactor();*

Destrutor simples, indispensável ao AutoCAD

- *virtual ~TPZreactor();*

Construtor da classe TPZreactor já especificando o id do nó ao qual o reator será associado.

- *TPZreactor(AcDbObjectId Ownerld);*

Funções virtuais do AutoCAD reimplementadas

Método que escreve em disco as características de um objeto do tipo TPZreactor virtual

- *Acad::ErrorStatus dwgOutFields(AcDbDwgFiler* pFiler) const;*

Método que escreve lê do disco as características de um objeto do tipo TPZreactor

- *virtual Acad::ErrorStatus dwglnFields (AcDbDwgFiler * pFiler);*

Métodos e variáveis públicas da classe TPZreactor:

Método que altera a posição de um TPZreactor, quando esse é movido pelo usuário

- *virtual void modijied(const AcDbObject* dbObj);*

Método que altera a lista de Id's válidos de objetos associados a um reator

- *Acad::ErrorStatus rclOwnerIdList(AcDbObjectIdArray ξ ValidIdList, AcDbObjectIdArrayξ LostI dList) const;*

Métodos e variáveis privadas da classe TPZreactor

Variável que armazena o Id de um objeto associado a um reator

- *AcDbObjectId fOwnerId;*

Método que adiciona o Id de um objeto na lista de Id's válidos associados a um reator

- *void setReactorOwnerId(AcDbObjectId OwnerId);*

Método que retoma o Id de um objeto associado a um reator

- *AcDbObjectId getReactorOwnerId() const;*

Operadores da classe TPZreactor

- *void *operator new[](unsigned nSize)* { *return 0;* }

- *void operator delete[] (void *p)* {};

- *void *operator new[](unsigned nSize, const char *file, int tine)* { *return 0;* }

7.2.5 Classe TPZgenProp

A classe TPZgenProp foi a última a ser definida, e, como o nome sugere, guarda as informações sobre as propriedades gerais do projeto. Na verdade, sua criação decorre da necessidade de se armazenar dados gerais em disco que fossem organizados e gerenciados por um dispositivo alheio às classes TPZnode e TPZbeam. Assim, centralizam-se as operações de armazenamento e recuperação de dados e se otimiza as funções das classes que precisam dessas informações.

Os principais métodos e variáveis da classe TPZreactor são:

Construtor simples, indispensável ao AutoCAD

- *TPZgenProp();*

Destrutor simples, indispensável ao AutoCAD

- *virtual ~TPZgenProp();*

Funções virtuais do AutoCAD reimplementadas

Método que escreve em disco as características de um objeto do tipo TPZgenProp

- *virtual Acad::ErrorStatus dwgOutFields (AcDbDwgFiler * pFiler) const;*

Método que escreve lê do disco as características de um objeto do tipo TPZgenProp

- *virtual Acad::ErrorStatus dwglnFields(AcDbDwgFiler* pFiler);*

Métodos e variáveis públicas da classe TPZgenProp:

Método que atribui a uma barra *os* parâmetros do material

- *void setMatpar(const ad_real E,const ads_real* Ni);

Método que retoma *os* parâmetros do material de uma barra

- *void getMatpar(ads_real E,ads_real Ni)const;*

Os métodos e variáveis que vem a seguir permitem a manipulação das características de representação e geometria dos objetos criados, bem como modificar as responsáveis pela representação em tela destas características e o sistema de unidades utilizado pelo usuário

- *void setNodeColors(const Adesk::Ulnt16 NodeColor, const Adesk::Ulnt16 NodeLoadColor)* ;
- *void getNodeColors(Adesk::Ulnt16 ξ NodeColor, Adesk::Ulnt16 ξNodeLoadColor)const;*
- *void setNodeGeometry(const ads_real NodeRadius, const AcGe Vector3d NodeNormal);*
- *void getNodeGeometry(ads_real ξ NodeRadius, AcGe Vector3d ξNodeNormal)const;*
- *void setBeamColors(const Adesk::Ulnt16 BeamColor, const Adesk::Ulnt16 FullFilled BeamColor, const Adesk::Ulnt16 BeamLoadColor);*
- *void getBeamColors(Adesk::Ulnt16 ξ BeamColor,Adesk::Ulnt16 ξFullFilledBeamColar, Adesk::Ulnt16 ξBeamLoadColor)const;*

- *void setPLateColors(const Adesk::Ulnt16 PlateColor, const Adesk::Ulnt16 FullFilled PlateColor, const Adesk::Ulnt16 PlateLoadColor);*

- *void getPlateColors(Adesk:: Ulnt16 ξ PlateColor, Adesk:: Ulnt16 f3FullFilledPLateColar, Adesk:: Ulnt16 ξPlateLoadColor)const;*

- *void setCurUnits(const char *LengthUnit, const char *ForceUnit);*

- *void getCurUnits(char *LengthUnit,char *ForceUnit)const;*

- *ads_real userLengthlntoMeter(const ads_real Source)const;*

- *ads_real meterlntoUserLength(const ads_real Source)const;*

- *ads_real userForcelntokgf(const ads_real Source)const;*

- *ads_real kgflntoUserForce(const ads_real Source)const;*

- *ads_real getLoadScale()const;*

- *void setLoadScale(const ads- real LoadScale);*

- *void setDrawableltems(const Adesk::Boolean DrawLoad, const Adesk::Boolean DrawGeometry, const*

- *Adesk::Boolean DrawDmomento, const Adesk::Boolean DrawDcortante);*

- *void getDrawableltems(Adesk::Boolean ξ DrawLoad, Adesk::Boolean ξDrawGeometry, Adesk::B oolean ξ DrawDmomento, Adesk:: Boolean ξ DrawDcortante) const;*

- *void setDrawLoad(const Adesk::Boolean DrawLoad);*

- *void getDrawLoad(Adesk::Boolean ξDrawLoad)const;*

- *void setDrawGeometry(const Adesk::Boolean DrawGeometry);*

- *void getDrawGeometry(Adesk::Boolean ξDrawGeometry)const;*

- *void setDrawDmomento(const Adesk::Boolean DrawDmomento);*

- *void getDrawDmomento(Adesk::Boolean ξDrawDmomento)const;*

- *void setDrawDcortante(const Adesk::Boolean DrawDcortante);*

- *void getDrawDcortante(Adesk::Boolean ξDrawDcortante)const;*

- *void getfDrawDcortante (Adesk.::Boolean ξDrawDcortante) const;*

- *void getShowDMomento(Adesk.::Boolean ξDrawDmomento).const;*
- *void getShowDCortante (Adesk::Boolean ξ DrawDcortante) const;*
- *Adesk::Boolean ShowDMomento();*
- *Adesk::Boolean ShowDCortante();*
- *Adesk::UInt16 fNodeColor;*
- *Adesk::UInt16 fNodeLoadColor;*
- *Ads_real fNodeRadius;*
- *Ads_real fE,fNi;*
- *AcGe Vector3d fNodeNormal;*
- *Adesk::UInt16 fBeamColor;*
- *Adesk::UInt16 fPlateColor;*
- *Adesk::UInt16 fFullFilledBeamColor;*
- *Adesk::UInt16 fFullFilledPlateColor;*
- *Adesk::UInt16 fBeamLoadColor;*
- *Adesk::UInt16 fPlateLoadColor;*
- *char fmatname[50];*
- *char fLengthUnit[3];*
- *char fForceUnit[4];*
- *ads_real fLoadScale;*
- *Adesk::Boolean fDrawLoad;*
- *Adesk::Boolean fDrawGeometry;*
- *Adesk::Boolean fDrawDmomento;*
- *Adesk::Boolean fDrawDcortante;*
- *Operadores da classe TPZgenProp*
- *void *operator new[] (unsigned nSize) { return 0; }*

- *void operator delete[] (void *p) {};*
- *void *operator new[](unsigned nSize, const char *file, int tine) { return 0; }*

7.3 Projeto ARX Geométrico

O projeto ARX é mais simples, não define classes novas e é procedural, mas tem a função de solicitar do usuário os dados necessários para a criação e edição de entidades, bem como a verificação da validade desses dados. Nele estão as instruções para o registro dessas novas funções no AutoCAD, o registro das novas classes, o registro do arquivo de extensão ".dbx", o registro das entidades criadas no banco de dados, a associação dos reatores aos nós e a remoção desses elementos da memória quando o AutoCAD é encerrado.

Nesse projeto foram colocados à disposição dos usuários os seguintes comandos:

- *void tpzplate():*

Esse comando permite ao usuário criar uma placa;

- *void tpznode():*

Esse comando permite ao usuário criar um nó;

- *void tpzbeam():*

Esse comando permite ao usuário criar um elemento de barra;

- *void tpzbeamproperties():*

Esse comando permite ao usuário atribuir características geométricas a uma barra;

- *void tpznodedisplacements():*

Esse comando permite ao usuário aplicar restrições de deslocamento a um nó;

- *void tpznoderotations ():*

Esse comando permite ao usuário definir as características básicas de trabalho, tais como unidades e forma de representação das entidades.

- *void tpznodeload():*

Esse comando permite ao usuário aplicar uma carga pontual em um nó;

- *void tpzbeamload():*

Esse comando permite ao usuário aplicar uma carga distribuída trapezoidal a uma barra;

- *void tpzgenprop():*

Esse comando permite ao usuário definir as características básicas de trabalho, tais como unidades e forma de representação das entidades;

- *void tpzbeammat():*

Esse comando permite ao usuário definir as propriedades materiais de uma barra;

- *Acad::ErrorStatus addReactor(const AcDbObjectId entityId, AcDbEntity *pEnt);*

Método que associa um reator a uma entidade do tipo TPZnode

- *Acad::ErrorStatus addBeamToNode(const AcDbObjectId Beamld, const AcDbObjectIdNodeld);*

Método que associa um Id de um nó a uma barra

- *Acad::ErrorStatus addPlateToNode(const AcDbObjectId Plateld, const AcDbObjectIdNodeId);*

Método que associa o Id de um nó a uma placa

- *int getXYZorAnyVector(char* FirstString, char* SecondString, AcGeVector3d ξ Dir);*

Esse método armazena um vetor que representa uma direção, com relação aos eixos globais X, Y e Z.

- *int getXYZLocalorAny Vector(char* FirstString, char* SecondString, AcGe Vector3dξ Dir, Adesk::Boolean ξ LocalCS)*

Esse método armazena um vetor que representa uma direção, com relação aos eixos locais 1, 2 e 3.

- *int getColor(const char* String, Adesk::UInt16 ξ Color)*

Esse método recebe do usuário uma *string* que representa uma cor.

Além destes métodos para novos comandos, existem alguns métodos que são indispensáveis para o funcionamento do programa. Com estes métodos o programa AutoCAD consegue interpretar os novos comandos e compreender as ações do usuário.

Os principais métodos auxiliares são:

- *Acad::ErrorStatus getOpenedEntity(AcDbEntity *(ξ pOpen), AcRxClass* pType, AcDb::OpenModemode);*

Método que coloca uma entidade em ponto de trabalho. Em muitos momentos, para que o AutoCAD possa atuar em uma entidade é preciso que ela esteja aberta para leitura e escrita. Esse método recebe uma entidade e a retoma aberta para leitura.

7.4 Projeto ARX de análise

Esse projeto foi criado para permitir a troca de informações entre as entidades incorporadas ao AutoCAD e as funções do ambiente PZ. A principal classe desse projeto é a classe TPZanalysis. Nessa classe, temos as seguintes funções:

- *void tpzanalisys():*

Esse método que aciona o comando *Analisys.* Com esse comando o usuário dispara o processo de montagem, resolução e impressão dos resultados do problema.

- *Acad::ErrorEstatus pre_processor_node(ads_name nodeSelection, TPZGeoMesh ξ gmesh, TPZCompMesh ξ compmesh):*

Método que faz o pré-processamento dos nós. Esse método recebe um conjunto seleção de objetos do tipo TPZnode. Em seguida é criado um objeto geométrico para cada um dos elementos do conjunto seleção com as respectivas características geométricas, condições de vinculação e carregamentos e os insere em uma malha geométrica.

- *Acad::ErrorEstatus pre_processor_beam(ads_name beamSelection, TPZGeoMesh ξ gmesh, TPZCompMesh ξ compmesh):*

Método que faz o pré-processamento das barras. Esse método recebe um conjunto seleção de objetos do tipo TPZbeam. Em seguida é criado um objeto geométrico unidimensional para cada um dos elementos do conjunto seleção com as respectivas características geométricas e estes são inseridos em uma malha geométrica. Também é criado um material para cada barra e esse é inserido em uma malha geométrica.

- *Acad::ErrorEstatus pre_processo_plate(ads_name plateSelection, TPZGeoMesh ξ gmesh, TPZCompMesh ξ compmesh):*

Método que faz o pré-processamento das placas. Esse método recebe um conjunto seleção de objetos do tipo TPZplate. Em seguida é criado um objeto geométrico para cada um dos elementos do conjunto seleção com as respectivas características geométricas e estes são inseridos em uma malha geométrica. Também é criado um material para cada placa e esse é inserido em uma malha geométrica.

- *Acad::ErrorEstatus processor(TPZGeoMesh ξ gmesh, TPZCompMesh ξ compmesh):*

Esse método realiza o processamento do sistema. Recebe como argumento uma malha computacional e uma malha geométrica, monta a conectividade entre os elementos e resolve o sistema resultante.

- *Acad::ErrorStatus post_processor(long bemsize, long platesize, TPZGeoMesh ξ gmesh, TPZComplv. ξ compmesh):*

Método que realiza o pós-processamento do problema e associa aos elementos os respectivos resultados.

- *int findnodeId (int Id, TPZGeoMesh ξ gmesh);*

Método que verifica a unicidade de um Id na malha geométrica.

Em todas essas etapas descritas são utilizados objetos e funções dos projetos .ARX, .DBX e do ambiente PZ.

8 Exemplos

Nos exemplos apresentados neste capítulo procurou-se mostrar as possibilidades de representação geométrica possíveis. Os resultados dos cálculos se mostraram coerentes em comparação com os resultados dos mesmos exemplos executados em programas de cálculo comerciais existentes no mercado. Não se procurou mostrar a acurasidade nos cálculos, pois os códigos utilizados para realização destes cálculos já foram validados e os resultados apresentados em outros trabalhos citados nesta tese.

Na figura 8.1, podemos observar uma viga contínua com um dos tramos em balanço e outros dois nós engastados. Foram aplicados carregamentos concentrados e distribuídos. Na figura 8.2, são apresentados os diagramas de momento resultantes dos cálculos.

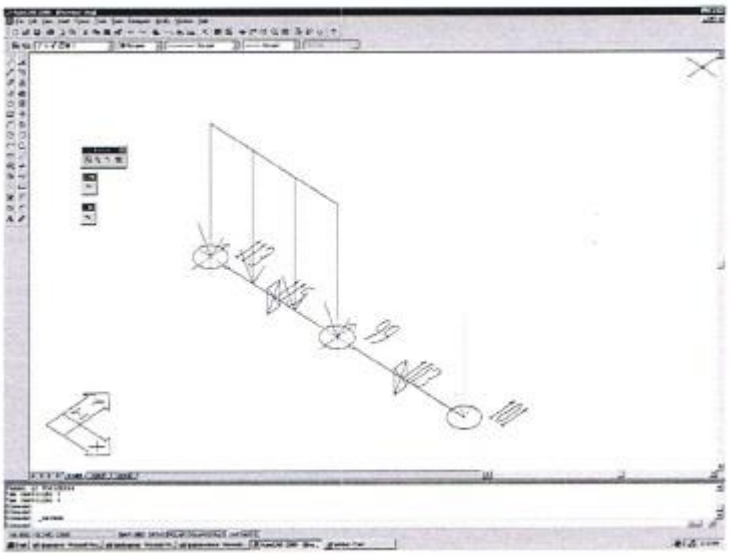

Figura 8.1: Exemplo de uma viga contínua com carregamentos aplicados

Na figura 8.3, podemos observar um pórtico espacial com dois apoios engastados. Foram aplicados carregamentos concentrados fora do plano do pórtico em um dos nós e distribuídos em uma barra. Na figura 8.4, são apresentados os diagramas de momento resultantes dos cálculos.

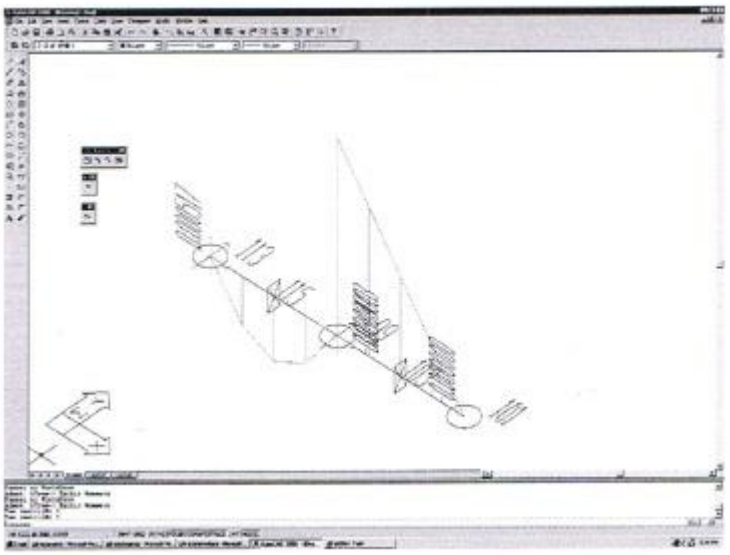

Figura 8.2: Exemplo de uma viga contínua com o diagrama de momento resultante do cálculo.

Na figura 8.3, pode-se observar uma placa com dois nós engastados e dois nós livres. Foram aplicados carregamentos concentrados nos dois nós livres. Ainda não existe representação para os diagramas de momento e cortante para uma placa.

Na figura 8.6, podemos observar um pórtico espacial com dois níveis e com apoios engastados na base. Foram aplicados carregamentos concentrados fora do plano do pórtico em uma das laterais. Na figura 8.7, são apresentados os diagramas de momento resultantes dos cálculos.

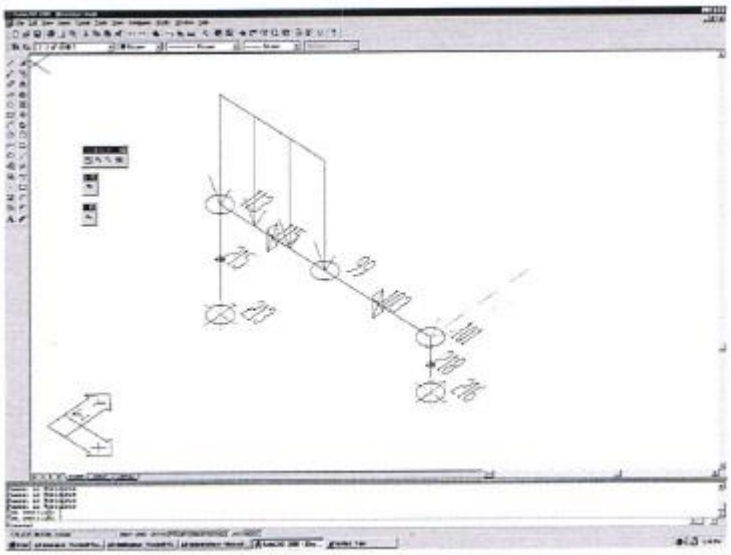

Figura 8.3: Exemplo de um pórtico espacial com carregamentos aplicados

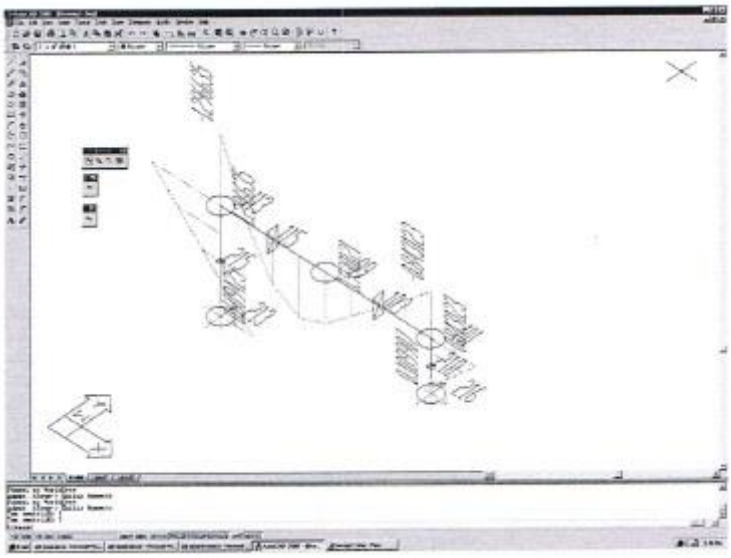

Figura 8.4: Exemplo de pórtico espacial com a representação dos diagramas de momento nas barras

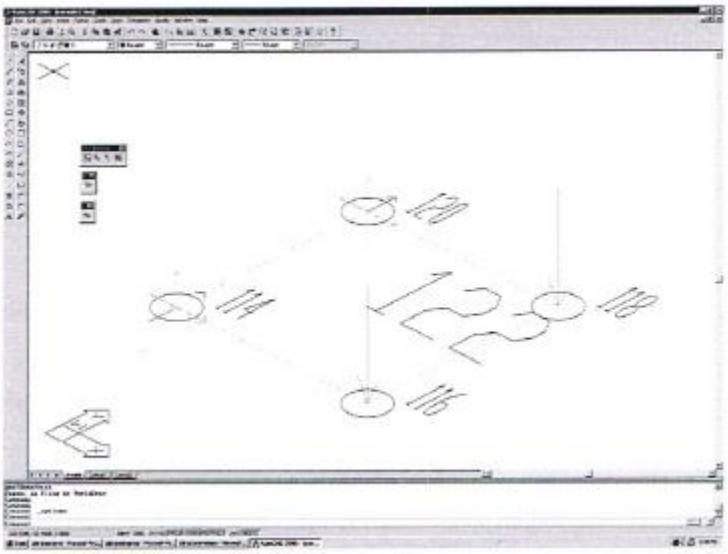

Figura 8.5: Exemplo de uma placa com dois nós engastados e cargas aplicadas nos dois outros nós

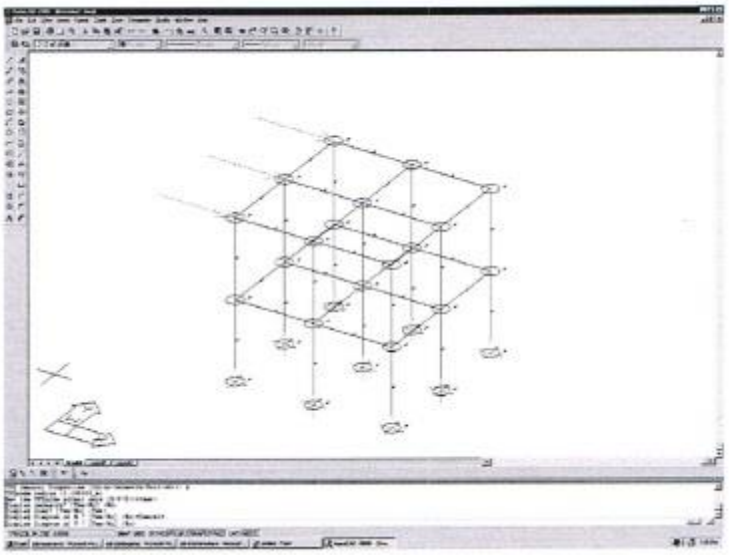

Figura 8.6: Exemplo de um pórtico de vários níveis, constituído de barras e placas, com carregamentos aplicados horizontais

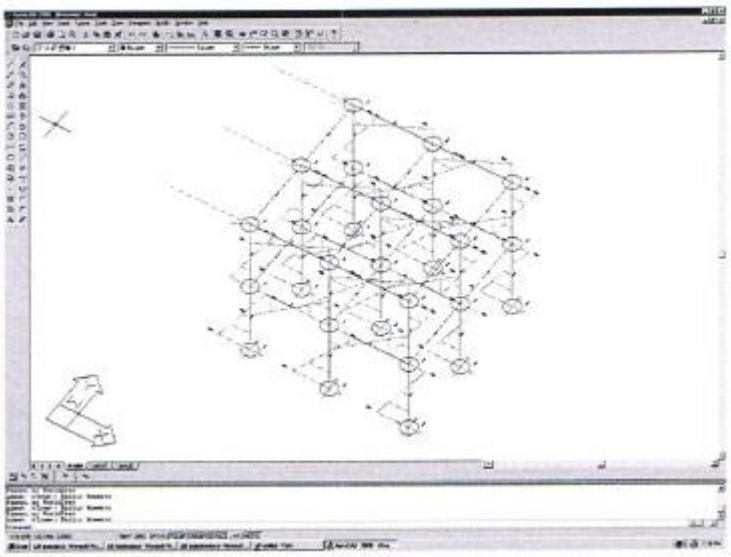

Figura 8.7: Representação dos diagramas de momento resultantes do cálculo nas barras

9 Considerações finais e possibilidades de desenvolvimentos futuros

A relevância desse trabalho consiste na união das funcionalidades do ambiente de Elementos Finitos PZ e a biblioteca ObjectARX para a construção de uma nova ferramenta que trabalha dentro do AutoCAD e que possibilita o cálculo de esforços em estruturas, espaciais ou planas, compostas de barras e placas.

É importante frisar que não há bibliografia específica que apresente esse tipo de união, o que demonstra a importância desse estudo, não apenas para Engenheiros Civis, mas também para outros profissionais do ramo de Engenharia.

Neste trabalho, construiu-se um ambiente de cálculo estrutural dentro de uma plataforma gráfica comercial. Esse acoplamento AutoCAD /ferramenta de cálculo é pioneira para problemas de Engenharia Civil e se utilizou a ferramenta mais avançada de customização para AutoCAD, denominada ObjectARX. A criação de novas entidades no AutoCAD permite a modelagem da estrutura da mesma praticidade com que um usuário de AutoCAD cria um desenho, e o emprego do ambiente de cálculo PZ possibilita o cálculo estrutural do sistema modelado em segundo plano, encapsulado pela plataforma gráfica. A biblioteca gerada pode ser vista tanto como uma extensão do AutoCAD para inclusão de um módulo de cálculo, quanto uma abertura do PZ, simplificando e sistematizando a modelagem e cálculo de estruturas formadas por barras, placas e cascas.

A capacitação para a utilização da biblioteca ARX foi um processo demorado, principalmente pela existência de lacunas na documentação. Alguns dos métodos disponibilizados pela AutoDesk não se comportaram tal qual a documentação informa. Principalmente, os métodos necessários para construir cópias de entidades não puderam ser utilizados. Outros métodos conseguiram ser utilizados apesar de apresentarem um comportamento diferente do indicado na documentação. Destaca-se ainda a aprendizagem da linguagem de programação *C++,* que não é comum a engenheiros, mas de grande valia para a vida profissional.

Como resultado do trabalho desenvolvido até aqui, obteve-se um conjunto de novas entidades que apresentam forma e comportamento controlados pelo usuário. Essas características podem ser estendidas ou adaptadas para diferentes tipos de elementos de acordo com as necessidades dos usuários ou com um tipo particular de problema que se deseje resolver.

Referências Bibliográficas

[1] Eriek B. Beeker, Graham F. Carey, and J. Tinsley Oden. *FINITE ELEMENTS, an introducion.* Prentiee-Hall, 1981.

[2] Graham Birtwistle. *SIMULA begin.* Auerbaeh, 1973.

[3] Jorge L. D. Calle. *Introdução a Linguagem C++, Programação Orientada para Objetos.* CENAPAD - UNICAMP, Novembro 1999.

[4] Saleeb Chefio *Constitutive Equations for Engeneering Materials, Second edition.* Elsevier Seienee B.V., 1994.

[5] OJ Dahl and K Nygaard. Simula - an algol-based simulation language. In *Communications of the ACM,* 1515 Broadway, New York, NY 10036, 1966. Assoe. Computing Maehinery

[6] Philippe Remy Bernard Devloo. pz: An objeet oriented environment for scientifie pro
gramming. *Comput. Methods Appl. Mech. Engrg.,* 1997.

[7] José Davi Furlan. *Modelagem de Objetos Através da UML - The Unified Modeling*
Language. Makron Books do Brasil Ltda., 1998.

[8] Adele Goldberg. *Smalltalk-BO.. The Interactive Programming Environmen.* Addison
Wesley Publishing, ,1983.

[9] Thomas J. R. Hughes. *FINITE ELEMENTS, Linear Static and Dynamic Finite Element*
Analysis. Prentiee-Hall, 1987.

[10] Ted Kaehler and Dave Patterson. *ATaste of Smalltalk.* WW Norton & Co, May 1986.

[11] Bill Kramer. *ObjectATX Primer.* Autodesk Press, 2000.

[12] Glen Krasner. *Smalltalk-BO.. Bits of History, Words 01 Advice.* Addison-Wesley Pu
blishing, 1983.

[13] S. G. Lekhnitskii. *Anisotropic Plates, Second edition.* Gordon and Breaeh Sciense Publishers, 1968.

[14] Charles MeAley. *Programming AutoCAD 2000 Using ObjectARX* Autodesk Press, 2000.

[15] F. A. Menezes and P. R. B. Devloo. Análise comparativa entre duas formulacões para

placas multi-camadas. In *XXIX jornadas Sudamericanas de Ingenieria EstructuraZ, in*
CDRoom, Punia deZ Este, Uruguay, 2000.

[16] J. Tinsley Oden and Graham F. Carey. *FINITE ELEMENTS, MathematicaZ Aspects.*
Prentice-Hall, 1983.

[17] Robert Sebesta. *Concepts of Programming Languages.* Addison- Wesley Publishing, CA., 1996.

[18] Bjarne Stroustru. *The C++ Programming Languagem, Third Edition.* Addison-Wesley Publishing Group, 1997.

Apêndice A

Método dos Elementos Finitos e problemas unidimensionais

A.1 Introdução

O Método dos Elementos Finitos é uma técnica para construção de soluções aproximadas para Problemas de Valor de Contorno[16]. Nesse tópico apresenta-se uma introdução ao método, voltada para problemas unidimensionais. Inicialmente é feita uma introdução às leis de conservação, em seguida é definido o método de *Galerkin* e finalmente apresenta-se o Método dos Elementos Finitos. Mostra-se também a definição das funções de forma, a formulação variacional do problema, sua discretização, o cálculo das matrizes de elementos e sua montagem para formar a matriz de rigidez global *K* e o vetar de força global *F*. Em seguida, discute-se a aplicação das condições de contorno no sistema de equações lineares algébricas resultantes.

No desenvolvimento do trabalho, mostra-se como o Método dos Elementos Finitos pode ser utilizado para resolver problemas que têm origem em leis de conservação.

A.2 Leis de Conservação

A Mecânica dos Meios Contínuos é regida por leis de conservação. Isto por si só justifica o estudo de métodos para resolvê-las. Dentre eles, um dos que têm recebido dessa que é o de elementos finitos.

Uma lei de conservação unidimensional, em um intervalo *[a, b],* em regime permanente, é uma relação do tipo:

$$\sigma(a) - \sigma(b) = \int_a^b f(x)dx \qquad (A.1)$$

entre o fluxo $\sigma(x)$ de uma variável de estado u(x) e a sua geração por unidade de tempo e comprimento $f(x)$, chamado de termo de fonte. Na lei de conservação da massa, por exemplo, $f(x)$ é a fonte de massa por unidade de tempo e comprimento e $\sigma(x) = \rho(x)U(x)$, em que ρ é a densidade e U a velocidade do meio. Essa lei se aplica em qualquer região [a, b] do meio. Em geral o que se deseja calcular é a variável de estado U (x) que satisfaz a lei de conservação (A.I).

Se $f(x)$ for contínua em uma vizinhança (x_1, x_2) de x_0, então o teorema do valor médio para integrais garante a existência de ξ em (x_0, x_2) tal que

$$\int_{x_0}^{x_2} f(x)dx = f(\xi)(x_2 - x_0) \qquad (A.2)$$

Isto permite escrever a lei de conservação no intervalo $[x_0, x_2]$ na forma

$$\sigma(x_2) - \sigma(x_0) = f(\xi)(x_2 - x_0)$$

Dividindo por $(x_2 - x_0)$ e tomando o limite quando $x_2 \rightarrow x_0^+$ se obtém

$$\lim_{x_2 \rightarrow x_0^+} = \frac{\sigma(x_2) - \sigma(x_0)}{(x_2 - x_0)} = \lim_{x_2 \rightarrow x_0^+} f(\xi) = f(x_0^+) \qquad (A.3)$$

Como *f* é contínua em x_0 e ξ está entre x_0 e x_2, o lado direito da equação (A.3) tende a $f(x_0)$e o lado esquerdo tende a $\sigma'(x_0^+)$, que é a derivada à direita de σ em x_0. Portanto a equação (A.3) se reduz a

$$\sigma'(x_0^+) = f(x_0) \qquad (A.4)$$

De modo análogo, deduz-se que

$$\sigma'(x_0^-) = f(x_0) \qquad (A.5)$$

Essas duas últimas equações garantem que o- é derivável em Xo e que

$$\sigma'(x_0) = f(x_0) \qquad (A.6)$$

Se *f* apresentar uma descontinuidade de salto em x_0, mas for contínua à esquerda e à direita desse ponto, em alguma vizinhança (x_1, x_2) de x_0, o raciocínio feito anteriormente continua aplicável. Repetindo-o obtém-se

$$\sigma'(x_0^+) = f(x_0^+) \qquad (A.7)$$

$$\sigma'(x_0^+) = f(x_0^+) \qquad (A.8)$$

em que

$$f(x_0^+) = \lim_{x_2 \rightarrow x_0^+} + f(\xi) \qquad (A.9)$$

e

$$f(x_0^-) = \lim_{x_2 \rightarrow x_0^+} - f(\xi) \qquad (A.10)$$

Não são necessariamente iguais a $f(x_0)$, uma vez que *f* é descontínua em (x_0). Subtraindo a equação (A.7) da equação (A.8) obtém-se

$$[[\sigma']](x_0) = [[f]](x_0) \qquad (A.11)$$

em que, para uma função g qualquer,

$$\left[[g]\right](x) = g(x^{+}) - g(x^{-}) \qquad (A.12)$$

é o salto de *g* no ponto *x.*

Isto significa que' nas descontinuidades de salto de $f(x)$, a derivada de $\sigma(x)$ sofre uma descontinuidade de salto, sendo (A.ll) a relação entre elas.

Considere-se agora uma fonte concentrada em um ponto, do tipo

$$f(x) = \hat{f}\,\delta\,(x - x_0) \qquad (A.13)$$

em que $\hat{f}$ é uma constante e $\delta\,(x - x_0)$, o *delta de Dirac.* A lei de conservação aplicada a um intervalo $[x_1, x_2]$ que contém x_0 nos leva a

$$\sigma(x_2) - \sigma(x_1) = \int_{x_1}^{x_2} f(x)\,dx = \int_{x_1}^{x_2} \hat{f}\delta\,(x - x_0)dx = \hat{f}$$

Fazendo $x_2 \rightarrow x_0^{+}$ e $x_1 \rightarrow x_0^{-}$, obtém-se

$$\sigma\,(x_2^{+}) - \sigma\,(x_2^{-}) = \hat{f}$$

ou

$$\left[[\sigma(x_0)]\right] = \hat{f} \qquad (A.14)$$

Isso mostra que σ terá uma descontinuidade de salto em x_0 e esse salto será igual a $\hat{f}$.

Nos materiais ditos lineares há uma relação linear entre o fluxo $\sigma\,(x)$ e a variável de estado $U(x)$. Essa relação, chamada de equação constitutiva, tem a forma

$$\sigma(x) = -k(x)\frac{dU}{dx}\,(x) \qquad (A.15)$$

em que *k(x)* é uma função conhecida e que depende do meio. Com essa equação constitutiva, a lei de conservação (A.I) se reduz a

$$\frac{d}{dx}\left[-k(x)\frac{dU}{dx}\,(x)\right] = f(x) \qquad (A.16)$$

nos pontos de continuidade de *f,*

$$\left[[\sigma']\right](x) = \left[[f]\right](x) \qquad (A.17)$$

nos pontos em que f tiver uma descontinuidade de salto e

$$[[\sigma]](x_0) = \hat{f} \qquad (A.18)$$

quando f tiver uma concentração do tipo $\hat{f}\delta\ (x - x_0)$.

Em determinados problemas, a fonte é do tipo $f(x) - b(x)U(x)$, quando então escreve-se a equação (A.16) na forma

$$\frac{d}{dx}\left[-k(x)\frac{dU}{dx}(x)\right] + b(x)U(x) = f(x) \qquad (A.19)$$

Nos problemas físicos, se a região de interesse se restringir ao intervalo *[a, b],* então fornecem-se condições de contorno. Tais condições podem ser de *Dirichlet*

$$U(a) = U_a\ , U(b) = U_b \qquad (A.20)$$

de Newmann

$$U'^{(a)} = U'_a\ , U'^{(b)} = U'_b \qquad (A.21)$$

ou mistas

$$k(a)\ U'(a) = P_a[U(a) - U_a] \qquad (A.22)$$

$$-k(b)\ U'(b) = P_b[U(b) - U_b] \qquad (A.23)$$

em que U_a, U_b, U'_a, U'_b, P_a, P_b são constantes dadas. As condições (A.20) são chamadas essenciais, as condições (A.21) são chamadas de condições naturais e as condições (A.22) e (A.23) são chamadas de condições mistas. A razão dessa nomenclatura surgirá no decorrer desse trabalho.

Conclui-se que uma formulação equivalente da lei de conservação é dada pela equação diferencial ordinária (A.19) nos pontos em que *f* for contínua, e pelas relações de salto (A.17) e (A.18) nos pontos em que *f* possuir descontinuidade de salto ou for uma fonte concentrada.

A.3 Formulação variacional

Nesse tópico, desenvolve-se uma formulação variacional, equivalente à lei de conservação, que constitui a base do método de elementos finitos. Para tanto, tome-se como região de interesse o intervalo *[a, b]* e que a parcela $f(x)$ da fonte seja do tipo

$$\bar{f}(x) + \hat{f}\delta(x - x_2)$$

com $\bar{f}(x)$ contínua, exceto em x_1 onde existe uma descontinuidade de salto e em x_2 onde existe uma condição de *Dirac,* e $\hat{f}$ uma constante dada. Considerou-se ainda $b(x)$ e $c(x)$ contínuas e $k(x)$ também contínua, exceto no ponto x_3, onde $K(x)$ tem uma descontinuidade de salto. Fisicamente isso significa, por exemplo, que em x_3 soldamos materiais com propriedades diferentes, como se pode observar na figura A.1.

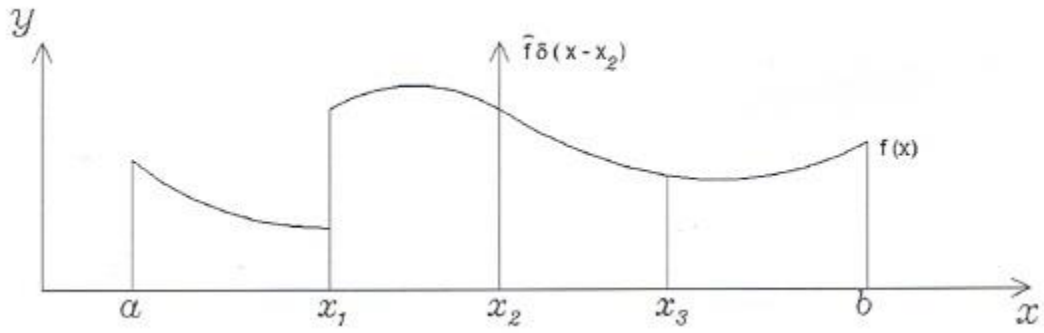

Figura A.l: Gráfico de *f(x)* no intervalo *[a, b].*

O ponto x_3 é aquele no qual k(x) tem um salto.

Nos intervalos (x_0, x_1), (x_1, x_2), (x_2, x_3), e (x_3, x_4), em que $x_0 = a$ e $x_4 = b$, a lei de conservação é equivalente à equação diferencial (A.19). Nos pontos x_1, x_2 e x_3 ela nos fornece o salto experimentado por $\sigma(x)$. Para esse problema, a lei de conservação, juntamente com as condições de fronteira, é equivalente ao problema de valor de contorno definido pela equação diferencial

$$\frac{d}{dx}\left[-k(x)\frac{dU}{dx}(x)\right] + b(x)U(x) = f(x) \qquad (A.24)$$

válida para x em $E_i = (x_{i-1}, x_i)$, $i = 1,2,3$, pelas condições de salto $\left[k(x)\frac{dU}{dx}(x)\right] = 0$ em

$$x = x_1 \, , x = x_3 \; e \; x = x_2 \qquad (A.25)$$

e pelas condições de fronteira

$$\alpha_0 \frac{dU}{dx}(a) + \beta_0 U(a) = \gamma_0 \qquad (A.26)$$

e

$$\alpha_1 \frac{dU}{dx}(b) + \beta_1 U(b) = \gamma_1 \qquad (A.27)$$

com α_0, β_0, γ_0, α_1, β_1, γ_1, constantes conhecidas. Se forem condições de fronteira do tipo (A.22) e (A.23), então $\alpha_0 = k\,(a)$, $\beta_0 = -P_a$, $\gamma_0 = -P_a U_a$, aI = k(b), $\alpha_1 = k\,(b)$, $\beta_1 = -P_b$, $\gamma_1 = P_b U_b$.

Multiplicando-se a equação (A.24) por uma função $V(x)$, pertencente a um espaço *H* de funções, que será especificado posteriormente e integrando o resultado em cada intervalo E_i obtém-se

$$-\int_{E_i}[k(x)U'(x)]'V(x)dx+\int_{E_i}b(x)U(x)V(x)dx=\int_{E_i}\bar{f}(x)V(x)dx\,,i=1,2,3$$

A função $V(x)$ é chamada de função tese e *H,* de espaço das funções tese. Integrando por partes, tem-se

$$-\int_{E_i}[k(x)U'(x)V(x)dx+b(x)U(x)V(x)]\ \ dx=$$

$$=\int_{E_i}\bar{f}(x)V(x)dx+k(x)U'(x)V(x)\Big|_{x_{i-1}^{+}}^{x_i^{-}}\qquad(A.28)$$

em que, para qualquer função *g(x),*

$$g(x)\Big|_{x_{i-1}^{+}}^{x_i^{-}}=g(x_i^{-})-g(x_{i-1}^{+})$$

Somando a equação (A.28) em todos os intervalos E_i, $i=1,2,3$, obtém-se

$$\int_a^b[kU'V'+bUV]dx=\int_a^b\bar{f}(x)V(x)dx+\sum_{i=1}^{3}\big[[kU'V]\big](x_i)+k(a)U'(a)V(a)-k(b)U'(b)V(b)\qquad(A.29)$$

Se as funções *V* forem contínuas, então

$$\big[[kU'V]\big](x_i)=V\,(x_i)\,\big[[kU']\big](x_i)\qquad(A.30)$$

Usando as condições (??) e (A.25), vê-se que a equação (A.29) se reduz a

$$\int_a^b[kU'V'+cU'V+bUV]dx$$
$$=\int_a^b\bar{f}Vdx+k(a)U'(a)V(a)-k(b)U'(b)v(b)+\hat{f}(x_2)\,(A.31)$$

em que *V* é uma função qualquer de *H.* É preciso caracterizar o espaço *H* das funções tese *V* e o espaço das funções admissíveis *U,* no qual o problema descrito faz sentido. Observe que na equação (A.31) aparecem as integrais de *U,* de *V* e suas derivadas primeiras que, portanto, devem ser integráveis. Se, eventualmente, tomarmos $V=U,$ teremos na equação (A.31),

$$\int_a^b U^2dx\ \ e\ \int_a^b U'^2dx$$

Desse modo, a equação acima fará sentido se *U* e *V,* bem como suas derivadas primeiras, forem de quadrado integrável, isto é, estiverem em L_2. Tal espaço de funções é denotado por H^1.

Quando as condições de contorno forem naturais, então

$$U'(a) = \frac{\gamma_0}{\alpha_0} - \frac{\beta_0}{\alpha_0} U(a)$$

$$U'(b) = \frac{\gamma_1}{\alpha_1} - \frac{\beta_1}{\alpha_1} U(b)$$

Com essas equações, pode-se escrever a equação (A.31) na forma

$$\int_a^b [kU'V' + cU'V + bUV]dx$$
$$= \int_a^b \bar{f}Vdx + k(a)\left[\frac{\gamma_0}{\alpha_0} - \frac{\beta_0}{\alpha_0} U(a)\right] V(a) - k(b)\left[\frac{\gamma_1}{\alpha_1} - \frac{\beta_1}{\alpha_1} U(b)\right] v(b)$$
$$+ \hat{f}\, V(x_2) \qquad (A.32)$$

Se *U* for uma solução de classe C^2 em E_i da equação (A.32), então ela satisfará o problema de valor de contorno definido em (A.24), (A.25), (A.26), (A.27). Reciprocamente, se *U* for solução das equações (A.24), (A.25), (A.26), (A.27), então ela será solução da equação (A.32) para toda função tese de *V,* no espaço das funções admissíveis H^1. Note-se que a equação (A.32), por não conter derivadas da segunda ordem, tem sentido em uma classe mais ampla de funções do que a equação (A.24), que é uma equação diferencial ordinária de segunda ordem. Desse modo, a equação (A.32) pode ter soluções que não satisfazem ao problema de valor no contorno original. Observe que na lei de conservação também não aparece a derivada segunda da variável de estado *U(x).* Nesse sentido, a formulação da lei de conservação dada por (A.32) é mais natural que o problema de valor no contorno definido pelas equações (A.24), (A.25), (A.26), (A.27). A essa formulação daremos o nome de formulação variacional do problema de valor no contorno.

A formulação variacional do problema de valor no contorno especificado no início dessa seção, consiste em determinar $U(x)$ em H^1, satisfazendo a equação (A.32), para toda $V(x)$ em H^1. A função $U(x)$, solução do problema variacional, é chamada de solução fraca do problema de valor no contorno definido em (A.24), (A.25), (A.26), (A.27). Pode-se provar que o problema variacional acima possui uma única solução.

Se as condições de contorno forem essenciais (de *Dirichlet),* então

$$U(a) = U_a \; e \; U(b) = U_b \qquad (A.33)$$

Neste caso utiliza-se eliminar os termos $U'(a)$ e $U'(b)$ na equação (A.31). Para contornar isto, tomamos o espaço das funções admissíveis $V(x)$ formado pelos elementos de H^1 que se anulam em *a* e *b,* isto é, $V(a) = V(b) = 0$. Tal espaço é denotado por H_0^1. Com essa escolha, nossa formulação variacional consistirá na determinação de *U (x)* em H^1, que satisfaz a equação (A.33) e à equação:

$$\int_a^b [kU'V' + cU'V + bUV]dx = \int_a^b \bar{f}Vdx + \hat{f}V(x_2) \qquad (A.34)$$

$$U'(a) = \frac{\gamma_0}{\alpha_0} - \frac{\beta_0}{\alpha_0} U(a)$$

$$U'(b) = \frac{\gamma_1}{\alpha_1} - \frac{\beta_1}{\alpha_1} U(b)$$

para toda função $V\ (x)$ em H_0^1. Observe-se que a equação (A.34) é idêntica à equação (A.32), com $V(a) = V(b) = 0$.

Observe-se que as condições naturais entram diretamente na equação variacional e as condições essenciais influenciam na escolha dos espaços de funções tese e de solução.

A.3.1 O Método de Galerkin

O método de *Galerkin* consiste em procurar uma aproximação $U_h(x)$ do problema variacional (A.32) em um subespaço vetorial H^h de H^1, de dimensão *N* finita. Escolhido esse subespaço e tomando-se nele uma base Φ_1, Φ_2, Φ_3, ..., Φ_N, pode-se decompor U_h numa combinação linear desses elementos e escrever

$$U_h(x) = \sum_{j=i}^{N} U_j \Phi_j(x) \qquad (A.35)$$

Aos elementos da base dá-se o nome de funções de forma. A aproximação deverá ser tal que a formulação variacional seja satisfeita para todo $V_h \in H^h$. Para que isso ocorra, basta que a equação (A.32) seja satisfeita para todo Φ_i, $i = 1,2,3, ..., N$ da base de H^h. Se as condições de contorno forem de *Dirichlet,* então H^h será um subespaço de H_0^1 e (A.35) será uma solução aproximada se satisfizer a equação (A.34) para toda função V^h de H^h, ou equivalente a toda função Φ_i da base.

Substituindo-se a equação (A.35) em (A.32) e tomando-se $V = \Phi_i$ i = **1,2,3,** ..., *N,* obtém-se o seguinte sistema de equações:

$$\left\{\sum_a^N \int_a^b \left(k\Phi_j'\Phi_i' + c\Phi_j'\Phi_i + b\Phi_j\Phi_i\right)dx + k(a)\frac{\beta_\theta}{\alpha_\theta}\Phi_j(a)\Phi_i(a)\ k(b)\frac{\beta_\theta}{\alpha_\theta}\Phi_j(b)\Phi_i(b)\right\} U_j$$
$$= \int_a^b \bar{f}\,\Phi_i dx + k(a)\frac{\gamma_1}{\alpha_1}\Phi_i(a) - k(b)\frac{\gamma_1}{\alpha_1}\Phi_i(b) + \hat{f}\Phi_i(x_2), i = 1,2,3, \dots, N$$

Reorganizando os termos dessas equações, chega-se a um sistema linear

$$\sum_{}^{N} k_{ij}U_j = f_i\,, i = 1,2,3,\dots,N \qquad (A.36)$$

nas incógnitas U_j, em que.

$$K_{ij} = \int_a^b \left(k\Phi_j'\Phi_i' + c\Phi_j'\Phi_i + b\Phi_j\Phi_i\right)dx + k(a)\frac{\beta_\theta}{\alpha_\theta}\Phi_j(a)\Phi_i(a) - k(b)\frac{\beta_\theta}{\alpha_\theta}\Phi_j(b)\Phi_i(b) \qquad (A.37)$$

e

$$F_i = \int_a^b \bar{f}\,\Phi_i dx + k(a)\frac{\gamma_1}{\alpha_1}\Phi_i(a) - k(b)\frac{\gamma_1}{\alpha_1}\Phi_i(b) + \hat{f}\Phi_i(x_2) \qquad (A.38)$$

A solução de (A.36) fornece a aproximação U_h procurada. A matriz $K = (K_{ij})_{n\times n}$ é chamada de matriz de rigidez e $F = (F_i)_{n\times 1}$ é chamada de vetor de forças.

Resta agora mostrar como se constrói um espaço H_h e como escolher uma base Φ_i, $i = \mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, N$ desse espaço. Essa escolha deve simplificar a montagem do sistema (A.36) e levar a uma boa aproximação da solução exata.

A.4 O Método dos Elementos Finitos

O Método dos Elementos Finitos é uma técnica que sistematiza a escolha do espaço H^h e da base Φ_i, $i = 1,2,3,\dots,N$,.

Num dos procedimentos possíveis, toma-se uma partição $P = \{x_k;\ k = 1,2,3,\dots,N, a = x_1 < x_2 < \dots < x_n = b\}$ do intervalo *[a, b]*. Os pontos x_k , $k = 1,2,3,\dots,N$, são chamados de nós. Os intervalos $[x_{i-1}, x_i]$, $i = 1,2,3,\dots,N$, juntamente com seus nós x_{i-1} e x_i , são chamados de elementos. Em seguida, define-se

$$\Phi_i\colon [a,b] \to R$$

por

$$\Phi_i(x) = \begin{cases} \dfrac{x - x_i - 1}{h_i} & x_{i-1} \le x \le x_i \\ \dfrac{x + x_i - 1}{h_i} & x_i \le x \le x_{i+1} \\ 0 & x < x_{i-1}\ e\ x > x_{i+1} \end{cases}$$

em que $h_i = x_i - x_{i-1}$, para cada $i = 1,2,3,\dots,N$. Subentende-se que quando $i = 0$ a função não está definida em $[x_{i-1}, x_i]$. Vale uma observação análoga quando $i = N$.

Essas funções são polinômios lineares por partes, sendo não nulas apenas no intervalo $[x_{i-1}, x_{i+1}]$. Satisfazem as propriedades

$$\Phi_i(x_j) = 0 \; se \; i \neq j$$

e

$$\Phi_i(x_j) = 1 \; se \; i = j$$

As funções formam uma base para o espaço das funções contínuas que, em cada intervalo da partição, são polinômios lineares. Usando na aproximação em (A.35) e a propriedade (A.36), chega-se a

$$U_i \; = \; U_k(x_i)$$

Essa escolha mostra-se muito consistente, pois reúne simplicidade e eficiência. Simplicidade, pois o cálculo das integrais em (A.37) e (A.38) são fáceis de calcular com essas funções de forma. Eficiência porque geram sistemas lineares tri-diagonais, uma vez que os produtos $\Phi_i' \, \Phi_j'$, $\Phi_i \;\; \Phi_j'$ são diferentes de zero apenas se i e j forem nós adjacentes, isto é, quando$|i - j| \leq 1$. Concluímos então que, para essa escolha de base, $k_{ij} \; = \; 0$ se $|i - j| \geq 2$. Essa base, bem como sua implementação computacional em códigos de elementos finitos é simples.

Pode-se tomar elementos com três ou mais nós, cujas funções são quadráticas ou cúbicas. Esses elementos teriam os dois nós extremos como os anteriores, e outros nós dividiriam o elemento em partes iguais. Para elementos quadráticos com três nós, por exemplo $i - 1, i, i \; + \; 1$, podemos associar três funções polinomiais Φ_{i-1}, Φ_i, Φ_{i+1}correspondentes aos nós dos elementos e que fariam parte da base H^h. Essas funções, restritas ao elemento em questão, são definidas por:

$$\Phi_{i-1}(x) = \begin{cases} \dfrac{(x - x_i)(x - x_i + 1)}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} & x_{i-1} \leq x \leq x_{i+1} \\ 0 & no \; restante \end{cases}$$

$$\Phi_i(x) = \begin{cases} \dfrac{(x - x_{i-1})(x - x_i + 1)}{(x_i - x_{i-1})(x_i - x_{i+1})} & x_{i-1} \leq x \leq x_{i+1} \\ 0 & no \; restante \end{cases}$$

$$\Phi_{i-1}(x) = \begin{cases} \dfrac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_{i+1})} & x_{i-1} \leq x \leq x_{i+1} \\ 0 & no \; restante \end{cases}$$

Aos elementos com quatro nós, podem ser associados polinômios de terceiro grau e assim por diante. A figura A.2, p.101, apresenta as funções de forma para elementos com dois e três nós.

Denominamos os intervalos $\left[x_i, x_{i+p}\right], i \; = \; 1, p \ldots, N \; - \; p,$ juntamente com seus nós, de elementos finitos. Se $p \; = \; 1$, o elemento é dito linear e a base usada é formada por polinômios lineares. Se $p \; = \; 2$, o elemento é dito quadrático, e a base utilizada é formada por polinômios quadráticos, e assim por diante. Os elementos lineares possuem dois nós, os quadráticos três, e assim por diante. O conjunto de todos os elementos forma uma malha. A dimensão do espaço H^h é igual ao número de nós da malha. O número $h = max\{x_i - x_{i-1}, i = 1,2, \ldots, N\}$ é chamado de tamanho ou norma da malha.

A.5 Cálculo das matrizes de elementos

O raciocínio apresentado fixa-se nos elementos lineares para calcular a matriz de rigidez e o vetor de cargas. Das equações (A.37) e (A.38), nota-se que, para calcular k_{ij} e F_i é preciso calcular as integrais:

$$\overline{k_{ij}} = \int_a^b (k\Phi_j'\Phi_i' + b\Phi_j\ \Phi_i\ )\,dx \qquad (A.40)$$

e

$$\overline{F_i} = \int_a^b \bar{f}\,\Phi_i dx \qquad (A.41)$$

Da propriedade aditiva das integrais, obtém-se que:

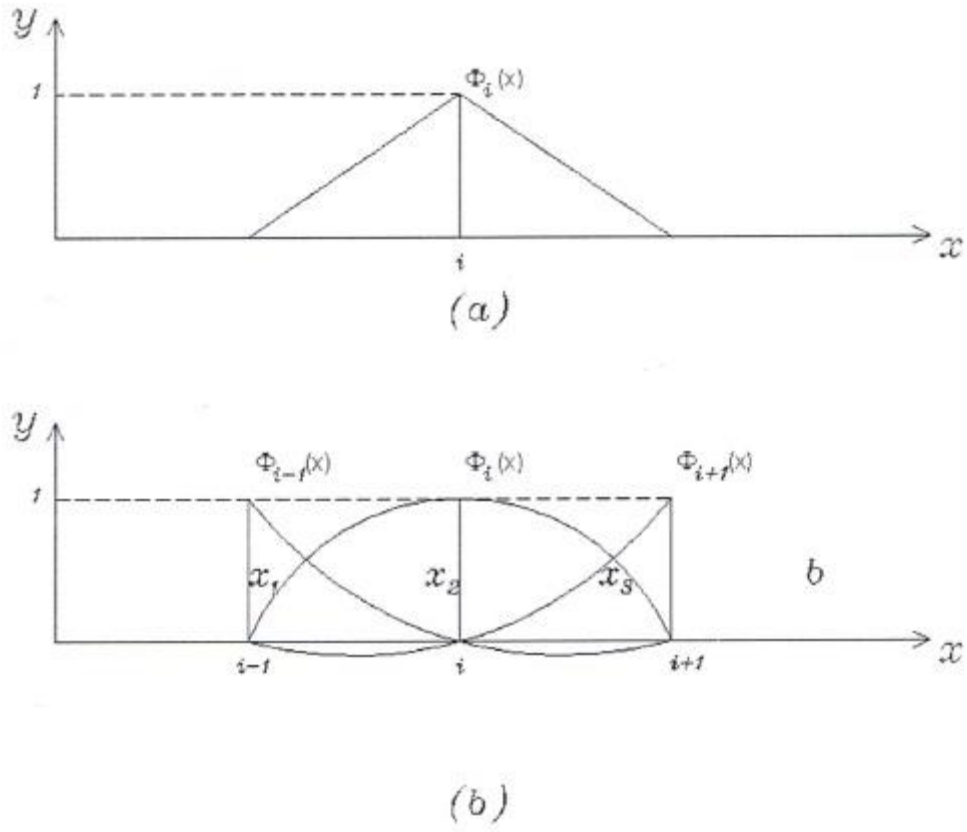

Figura A.2: a) Funções de forma para elementos lineares (com dois nós). b) Funções de forma para elementos lineares quadráticos (com três nós).

$$\overline{k_{ij}} = \sum_{e=1}^{N_e} \int_{\Omega_e} \left(k\Phi_j'\Phi_i' + b\Phi_j\ \Phi_i\ \right)dx = \sum_{e=1}^{N_e} \overline{k_{ij}}^{\,e} \qquad (A.42)$$

$$\overline{F_i} = \sum_{e=1}^{N_e} \int_{\Omega_e} \bar{f}\, \Phi_i dx = \sum_{e=1}^{N_e} \overline{F_i^e} \qquad (A.43)$$

em que N_e é o número de elementos da malha e $\Omega_e = [x_{e-1}, x_e]$ é um elemento finito. Essas equações nos dizem que é possível calcular as integrais em cada elemento e somá-las em todos os elementos para obter $\overline{k_{ij}}$ e $\overline{F_i}$. Como $\Phi_j \Phi_i = 0$ se $|i-j| \geq 2$, somente as funções de forma correspondentes aos nós do elemento trarão uma contribuição às integrais (A.42) e (A.43). Para calcular essas duas equações, costuma-se mapear o elemento para um elemento padrão, chamado de elemento mestre. Esse mapeamento é realizado mediante uma mudança de variáveis e permite colocar cada elemento $\Omega_e = [x_{e-1}, x_e]$, $e = 1,2,\dots,N$, em função do elemento padrão $\widehat{\Omega} = [-1, +1]$, dada por

$$x = \frac{1}{2}\left[(1-\xi)_{x_{e-1}} + (1-\xi)_{x_e}\right] \qquad (A.44)$$

Com essa mudança de variáveis, as funções $\Phi_{e-1}^e \Phi_e^e$ (restritas ao elemento Ω_e) serão mapeadas nas funções

$$\widehat{\psi_1}(\xi) = \frac{1}{2}(1-\xi)\ e\ \widehat{\psi_2}(\xi) = \frac{1}{2}(1+\xi) \qquad (A.45)$$

Com isto,

$$\Phi_{e-1}^e = \widehat{\psi_1}(\xi)\ e\ \Phi_e^e = \widehat{\psi_2}(\xi) \qquad (A.46)$$

sendo a relação entre x e ç dada pela equação (A.44).

Para que se possa calcular a integral

$$I = \int_{\Omega_e} g(x)dx \qquad (A.47)$$

em que g é uma função contínua qualquer, pode-se usar o teorema da mudança da variáveis para integrais, dado por

$$I = \int_{\widehat{\Omega}} \hat{g}(\xi)\frac{dx}{d\xi}d\xi = \frac{1}{2}(x_e - x_{e-1})\int_{-1}^{1} \hat{g}(\xi)d\xi \qquad (A.48)$$

em que $\hat{g}(\xi) = g(x)$, com x e ξ relacionados pela equação (A.44). Pode-se usar a quadratura gaussiana para aproximar essa integral para obter

$$I \cong \frac{1}{2}(x_e - x_{e-1})\sum_{i=1}^{M} \hat{g}(\xi_i)w_i \qquad (A.49)$$

em que M é a ordem da quadratura, $\xi_i \in [-1,1]$ são os pontos nos quais se toma a função e w_i são os pesos desses pontos. A formula (A.49) permite automatizar o cálculo de $\overline{k_{ij}}$ e F_i. Prosseguindo esse desenvolvimento, considere-se um elemento genérico $\Omega_e = [x_{e-1}, x_e]$ e denote-se por $\psi_1^e(x)$ e $\psi_2^e(x)$ as funções $\Phi_{e-1}^e(x)$ e $\Phi_e^e(x)$, respectivamente, restritas ao elementoΩ_e. Dessa suposição, segue que

$$\psi_1^e(x) = \Phi_{e-1}^e(x) = \widehat{\psi_1}(\xi)$$

e

$$\psi_2^e(x) = \Phi_e^e(x) = \widehat{\psi_2}(\xi) \qquad (A.50)$$

para x em Ω_e. Como K_{ij}^e e F_i^e só não são nulos se i e j estiverem no conjunto $(e-1, e)$ de (A.42) e (A.43), tem-se que

$$\overline{k_{e,e}^e} = \int_{\Omega_e} (k\Phi_e'\Phi_e' + c\Phi_e'\Phi_e + b\,\Phi_e\,\Phi_e)dx = \int_{\Omega_e} (k\psi_e'\psi_e' + c\psi_e'\psi_e + b\,\psi_e\,\psi_e)dx =$$

$$= \int_{\Omega_e} (\hat{k}\widehat{\psi_2^{e\prime}}\widehat{\psi_2^{e\prime}}\left(\frac{d\xi}{dx}\right)^2 + \hat{c}\widehat{\psi_2^{e\prime}}\widehat{\psi_2^e}\left(\frac{d\xi}{dx}\right) + \hat{b}\widehat{\psi_2^e}\widehat{\psi_2^e}\frac{d\xi}{dx}d\xi = k_{22}^e \qquad (A.51)$$

sendo $\hat{k}(\xi) = k(x)$, $\hat{c}(\xi) = c(x)$ e $\hat{b}(\xi) = b(x)$, com x e ξ relacionados pela equação (A.44). Analogamente,

$$\overline{k_{e-1,e}^e} = \int_{\widehat{\Omega}} (\hat{k}\,\widehat{\psi_1'}\widehat{\psi_2'}\left(\frac{d\xi}{dx}\right)^2 + \hat{c}\widehat{\psi_1'}\widehat{\psi_2^e}\left(\frac{d\xi}{dx}\right) + \hat{b}\widehat{\psi_1}\,\widehat{\psi_2})\frac{d\xi}{dx}\,d\xi = k_{12}^e \quad (A.52)$$

$$\overline{k_{e,e-1}^e} = \int_{\widehat{\Omega}} (\hat{k}\,\widehat{\psi_2'}\widehat{\psi_1'}\left(\frac{d\xi}{dx}\right)^2 + \hat{c}\widehat{\psi_2'}\widehat{\psi_1^e}\left(\frac{d\xi}{dx}\right) + \hat{b}\widehat{\psi_2}\,\widehat{\psi_1})\frac{d\xi}{dx}\,d\xi = k_{21}^e \quad (A.53)$$

$$\overline{k_{e,e-1}^e} = k_{11}^e \qquad (A.54)$$

e

$$\overline{Fk_{e,e-1}^e} = \frac{1}{2}(x_e - x_{e-1})\int_{\widehat{\Omega}} \hat{f}\,\widehat{\psi_1}\,d\xi = f_1^e \qquad (A.55)$$

$$\overline{Fk_{e,e-1}^e} = \frac{1}{2}(x_e - x_{e-1})\int_{\widehat{\Omega}} \hat{f}\,\widehat{\psi_2}\,d\xi = f_1^e \qquad (A.56)$$

Os demais termos k_{ij}^e e F_i^e são nulos. Essas condições permitem que se calculem as matrizes de rigidez e vetor de cargas locais

$$k^e = \begin{bmatrix} k_{11}^e & k_{12}^e \\ k_{21}^e & k_{22}^e \end{bmatrix} \; e \; F^e = \begin{bmatrix} f_1^e \\ f_2^e \end{bmatrix} \qquad (A.57)$$

com isto, pode-se construir a discretização local do elemento dada por

$$\begin{aligned} k_{11}^e U_1^e + k_{12}^e U_2^e &= f_1^e + \sigma(x_i) \\ k_{21}^e U_1^e + k_{22}^e U_2^e &= f_2^e - \sigma(x_{i+1}) \end{aligned} \qquad (A.58)$$

e assim, pode-se montar a equação global, como se pode observar a seguir.

Para uma malha com *N* elementos numerados consecutivamente, conforme a figura A.3, e utilizando-se a equação (A. 58) em cada elemento e somando-se a contribuição de todos esses elementos, chega-se a um sistema de equações lineares com *N* equações e *N* incógnitas.

Para que se possa mostrar como a contribuição de cada elemento é somada para formar a matriz de rigidez $k = [k_{ij}]$ e o vetor de cargas $F = [F_i]$, tome-se uma malha de quatro elementos. Para cada elemento E_i, $i = 1,2,3,4,$ têm-se duas equações na forma da equação (A. 58) que são, respectivamente,

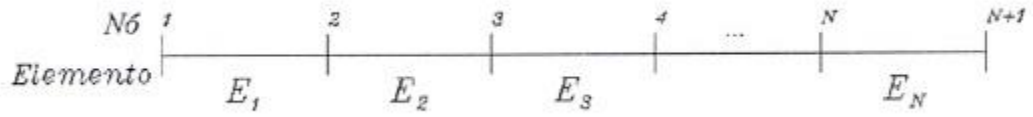

Figura A.3: Malha de elementos finitos de N elementos e N + 1 nós.

$$\begin{aligned} k_{11}^1 U_1^1 + k_{12}^1 U_2^1 &= f_1^1 + \sigma(x_1) \\ k_{21}^1 U_1^1 + k_{22}^1 U_2^1 &= f_2^1 - \sigma(x_2) \end{aligned} \quad , \quad \begin{aligned} k_{11}^2 U_2^2 + k_{12}^2 U_3^2 &= f_1^2 + \sigma(x_2) \\ k_{21}^2 U_2^2 + k_{22}^2 U_3^2 &= f_2^2 - \sigma(x_3) \end{aligned}$$

$$\begin{aligned} k_{11}^3 U_3^3 + k_{12}^3 U_4^3 &= f_1^3 + \sigma(x_3) \\ k_{21}^3 U_3^3 + k_{22}^3 U_4^3 &= f_2^3 - \sigma(x_4) \end{aligned} \quad , \quad \begin{aligned} k_{11}^4 U_4^4 + k_{12}^4 U_5^4 &= f_1^4 + \sigma(x_4) \\ k_{21}^4 U_4^4 + k_{22}^4 U_5^4 &= f_2^4 - \sigma(x_5) \end{aligned}$$

Somando-se a contribuição de cada elemento, tem-se

$$\begin{aligned} k_{11}^1 U_1 + k_{12}^1 U_2 &= f_1^1 + \sigma(x_1) \\ k_{21}^1 U_1 + (k_{22}^1 + k_{11}^2) U_2 + k_{12}^2 U_3 &= f_2^1 + f_1^2 \\ k_{21}^2 U_2 + (k_{22}^2 + k_{11}^3) U_3 + k_{12}^3 U_4 &= f_2^2 + f_1^3 \\ k_{21}^3 U_3 + (k_{22}^3 + k_{11}^4) U_4 + k_{12}^4 U_5 &= f_2^3 + f_1^4 \\ k_{21}^4 U_4 + k_{22}^4 U_5 &= f_2^4 - \sigma(x_5) \end{aligned} \qquad (A.59)$$

Com isto, o sistema de equações lineares para a malha apresenta a seguinte forma

$$\begin{bmatrix} \overline{k_{11}} & \overline{k_{12}} & & & \\ \overline{k_{21}} & \overline{k_{22}} & \overline{k_{23}} & & \\ & \overline{k_{32}} & \overline{k_{33}} & \overline{k_{34}} & \\ & & \overline{k_{43}} & \overline{k_{44}} & \\ & & & \overline{k_{54}k_{55}} & \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{bmatrix} = \begin{bmatrix} f_1^1 + \sigma(x_1) \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ f_2^3 + f_1^4 \\ f_2^4 - \sigma(x_5) \end{bmatrix} \qquad (A.60)$$

Para uma malha com *(N* - 1) elementos, ter-se-ia

$$\begin{bmatrix} \overline{k_{11}} & \overline{k_{12}} & & & \\ \overline{k_{21}} & \overline{k_{22}} & \overline{k_{23}} & & \\ & \overline{k_{32}} & \overline{k_{33}} & \overline{k_{34}} & \\ - & - & - & - & - \\ & & & \overline{k_{N-1,N-1}} & \overline{k_{N-1,N}} \\ & & & \overline{k_{N,N-1}} & \overline{k_{N,N}} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ - \\ U_{N-1} \\ U_N \end{bmatrix} = \begin{bmatrix} f_1^1 + \sigma(x_1) \\ f_2^1 + f_1^2 \\ f_2^2 + f_1^3 \\ - \\ f_2^{N-2} + f_1^{N-1} \\ f_2^{N-1} - \sigma(x_N) \end{bmatrix} \qquad (A.61)$$

onde $\overline{k_{11}} = k_{11}^1$, $\overline{k_{NN}} = k_{22}^N$, $k_{ij} = 0$, se $|i - j| \geq 2$, e $\overline{k_{\iota,\iota-1}} = k_{2i}^{i-1}$, $\overline{k_{\iota,\iota+1}} = k_{12}^i$, $k_{ii} = k_{22}^{i-1} + k_{11}^i$, para $i \neq 1$ e $i \neq N$.

A.6 Condições de contorno

Nesse tópico discute-se a introdução das condições de contorno no sistema da equação (A.61). O primeiro tipo de condição de contorno abordada é a condição. Essas tem a forma:

$$\begin{aligned} \alpha_1 U'(x_1) + \beta_1 U(x_1) &= \gamma_1 \\ \alpha_N U'(x_N) + \beta_N U(x_N) &= \gamma_N \end{aligned} \qquad (A.62)$$

Dessas expressões, pode-se calcular $\sigma(x_1)$ e $\sigma(x_N)$, explicitando-se $U'(x_1)$ e $U'(x_N)$

$$\sigma(x) = -k(x)U'(x) \qquad (A.63)$$

substituindo-se (A.63) e (A.62), tem-se

$$\begin{aligned} \sigma(x_1) = -k(x_1)U'(x_1) &= \frac{-k_1(\gamma_1 - \beta_1 U_1)}{\alpha_1} \\ \sigma(x_N) = -k(x_N)U'(x_N) &= \frac{-k_N(\gamma_N - \beta_N U_N)}{\alpha_N} \end{aligned} \qquad (A.64)$$

levando-se (A.64) e (A.61) obtém-se

$$\begin{bmatrix} k_{11} & k_{12} & & & & \\ k_{21} & k_{22} & k_{23} & & & \\ & k_{32} & k_{33} & k_{34} & & \\ - & - & - & - & - & \\ & & & k_{N-1,N-1} & k_{N-1,N} & \\ & & & k_{N,N-1} & k_{N,N} & \end{bmatrix} \begin{bmatrix} U_I \\ U_2 \\ U_3 \\ - \\ U_{N-1} \\ U_N \end{bmatrix} = \begin{bmatrix} F_I \\ F_2 \\ F_3 \\ - \\ F_{N-1} \\ F_N \end{bmatrix}$$

em que

$$k_{11} = \overline{k_{11}} - \frac{\beta_1}{\alpha_1} k_1 \, , k_{NN} = \overline{k_{NN}} - \frac{\beta_N}{\alpha_N} k_N$$

$$F_1 = f_1^1 + \frac{\gamma_1}{\alpha_1} k_1 \, , F_N = f_2^{N-1} + \frac{\gamma_N}{\alpha_N} k_N$$

e $U(x_1) = U_1 \; e \; U(x_2) = U_2$

Nesse caso, a primeira e a última equação são eliminadas, e os termos em U_1 e U_N na segunda e penúltima equação são transferidos para o segundo membro de (A.61). Com isso o sistema (A.61) apresenta a forma:

$$\begin{bmatrix} k_{22} & k_{23} & & & \\ k_{32} & k_{33} & k_{34} & & \\ - & - & - & - & \\ & & k_{N-1,N-2} & k_{N-1,N-1} & \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \\ - \\ U_{N-1} \end{bmatrix} = \begin{bmatrix} f_2^1 + f_1^2 - k_{21} U_1 \\ f_2^2 + f_1^3 \\ - \\ f_2^{N-2} + f_1^{N-1} - k_{N-1,N} U_N \end{bmatrix} \qquad (A.65)$$

A segunda condição abordada nesse trabalho é a condição de *Neumann,* que é um caso particular da condição natural. Essa condição e expressa por

$$U'(x_1) = U_1' \; e \; U'(x_N) = U_N' \qquad (A.66)$$

Ela conduz a problemas de valor de contorno que não possuem solução discretizada única. Se tomarmos uma equação diferencial do tipo

$$-[kU']' = f \qquad (A.67)$$

e $U(x)$ for uma solução do problema definido pelas equações (A.66) e (A.67), então $U(x) + \; c$, sendo c uma constante, também será solução do sistema. Nesse caso, para evitar uma indeterminação no sistema discretizado, arbitra-se o valor de *U* em algum ponto da malha, que em geral é em x_0. Com isso, diminui-se o sistema algébrico em uma equação, que passa então a

$$\begin{bmatrix} k_{22} & k_{23} & & \\ k_{32} & k_{33} & k_{34} & \\ - & - & - & - \\ & & k_{N,N-1} & k_{N,N} \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \\ - \\ U_N \end{bmatrix} = \begin{bmatrix} f_2^1 + f_1^2 - k_{21} C_o \\ f_2^2 + f_1^3 \\ - \\ f_2^{N-1} + k_N U_N' \end{bmatrix} \quad (A.68)$$

Pode-se então resolver o sistema para $U_2, U_3, \ldots, U_N$.

O último tipo de condição de contorno é a condição do tipo mista. Diz-se que uma condição é do tipo mista quando as condições de *Dirichlet* são aplicadas em uma extremidade e as condições naturais são aplicadas na outra extremidade do elemento. Como exemplo, tem-se

$$U(x_1) = U_1 \; e \; U'(x_N) = U_N'$$

$$\alpha_1 U'(x_1) + \beta_1 U(x_1) = \gamma_1 \; U(x_N) = U_N \qquad (A.69)$$

com essa condições, tem-se o sistema

$$\begin{bmatrix} k_{22} & k_{23} & & \\ k_{32} & k_{33} & k_{34} & \\ - & - & - & - \\ & & k_{N,N-1} & k_{N,N} \end{bmatrix} \begin{bmatrix} U_2 \\ U_2 \\ - \\ U_N \end{bmatrix} = \begin{bmatrix} f_2^1 + f_1^2 - k_{21} U_1 \\ f_2^2 + f_1^3 \\ - \\ f_2^{N-1} + U_N \end{bmatrix} \qquad (A.70)$$

A.7 Estimativa de erro

É possível demonstrar a priori que se a solução do problema de valor de contorno é tal que as suas derivadas de ordem n ($n > 1$) são de quadrado integrável sobre o domínio Ω e as de ordem superior a *n* não o são, e as funções de forma Φ_i são polinômios completos de graus $\leq k$, e ainda a malha é uniforme com elementos de tamanho *h,* a aproximação do erro na norma H^1 é dada por

$$\left|\left|u - u_h\right|\right|_{H^1} \leq C h^{\mu}$$

em que C é uma constante e

$$\mu = \min(k,n)$$

Com respeito à norma do L_2, o erro é uma ordem melhor

$$\left|\left|u - u_h\right|\right|_2 \leq C_1 h^{\mu+1}$$

em que C_1 é uma constante.

Apêndice B

Linguagem *C++*

C++ é uma linguagem de programação que implementa a filosofia de orientação para objetos. Os conceitos básicos desse enfoque foram já introduzidos pela linguagem de programação *Simula* desenvolvida na década de 60 por O. J. Dahl e Kristen Nygaard [5]. A linguagem de programação *Simula* foi desenvolvida para realizar simulações computacionais de processos reais, e nela, a elaboração de módulos é central. A construção desses módulos baseia-se nos objetos físicos a serem modelados [3], [2], [17].

Essa linguagem não obteve sucesso no desenvolvimento de programas de propósitos gerais, mas os conceitos nela utilizados foram explorados por várias linguagens posteriores, entre as quais se destacam a linguagem *SmallTalk* e a linguagem *C++*. A *SmallTalk* foi desenvolvida pela *Xerox PARC* (PaIo Alto Research Center, California) na década de 70, por uma equipe coordenada por Alan Kay [8], [12], [10].

O desenvolvimento da linguagem *C++* iniciou-se em 1980, nos laboratórios da *Bell,* por Bjarne Stroustrup [18]. A linguagem foi originalmente desenvolvida para solucionar algumas simulações motivadas por eventos, muito rigorosas, para as quais as considerações sobre eficiência impediam o uso de outras linguagens. A linguagem *C++* foi utilizado primeiramente por uma equipe não pertencente ao grupo de desenvolvimento coordenado por Stroustrup em 1983 e, no verão de 1987, a linguagem estava ainda em pleno desenvolvimento, passando por refinamentos e evoluções naturais.

Um objetivo chave do projeto *C++* era manter a compatibilidade com C. A idéia era preservar a integridade de milhões de linhas de código já escritas e depuradas, a integridade de muitas bibliotecas C existentes e a utilidade de ferramentas C já desenvolvidas. Devido ao alto grau de sucesso na obtenção desse objetivo, a transição de C para o *C++* mostrou-se muito simples. A melhoria mais significativa na linguagem *C++* é seu suporte à filosofia de programação orientada para objetos (OOP).

As linguagens *SmallTalk, Java* e *C++* são as mais utilizadas atualmente, embora possuam diferentes enfoques para realizar a orientação para objetos. Essa é considerada uma linguagem pura, enquanto que o C + + é híbrida, no sentido de que nela coexistem características de uma linguagem estruturada (o C) juntamente com características de orientação para objetos.

O fato de ser muito difundida faz da linguagem *C++* uma das linguagens que mais se desenvolve. A sintaxe de *C++* é muito rica, dando ao programador várias opções para escrever a mesma instrução. Comandos muitas vezes escritos com muitas linhas, podem ser reescritos com poucas linhas, o que se traduz não só em redução no tempo de implementação, mas também execução, pois o compilador criará um código mais eficiente.

B.1 Orientação a objetos

A filosofia de orientação à objetos teve sua maior penetração na indústria de software. Porém seu uso no âmbito de computação científica ainda é pequeno.

Essa filosofia de projeto de sistemas aborda os problemas em estudo sob um ponto de vista estruturado (i.e. não orientado a objetos), procura-se estabelecer um conjunto de

dados que definem o estado do sistema e uma posterior sequência de eventos que acarretará a transformação desse estado.

Numa abordagem orientada a objetos definem-se estruturas abstratas, denominadas classes, responsáveis por partes da modelagem do problema. Cada classe incorpora tanto dados (forma) como métodos (comportamentos), sendo esses necessários e suficientes para tratar as responsabilidades da classe.

Sendo a classe uma abstração, a simples definição da mesma não é suficiente para promover a solução do problema. Faz-se necessário criar uma instância dessa classe, chamado objeto. Esse é uma entidade auto-suficiente, incorpora forma e comportamento conforme sua classe.

A correta interação entre vários objetos, de classes responsáveis por diferentes partes do problema global, resultará na solução do problema.B.2 Elementos da Programação Orientada a Objetos

B.2.1 Classes

Segundo Furlan [7], uma classe é "uma coleção de objetos que podem ser descritos com os mesmos' atributos e as mesmas operações". Determinam-se os objetos que compartilham as mesmas necessidades e comportamentos, definindo a responsabilidade da classe.

Essa precisa definição de escopos e responsabilidades torna o planejamento e a elaboração do sistema simples e de fácil entendimento. Além disso, cada classe é responsável por uma porção da solução e uma classe incorpora abstratamente forma e comportamento, não sendo necessário que outras classes tenham acesso ao esquema de funcionamento de uma determinada classe.

Uma classe pode conter objetos e funções. Isso permite estruturar melhor os objetos e limita o acesso a dados por meio de funções próprias. Com isso, definem-se as propriedades e atributos que determinam o comportamento de um objeto.

As variáveis e as funções de uma classe podem ser protegidas, privadas ou públicas, conforme o nível de proteção ou controle que se deseja atribuir a elas.

Uma classe tem dois métodos que principais. O primeiro é chamado de construtor, que ocorre quando um objeto de uma classe é inicializado. Esse método é responsável pela inicialização das variáveis do objeto. O segundo método é denominado destrutor, e é chamado sempre que um objeto alcança o final do seu escopo.

B.2.2 Objetos

Os objetos, como o próprio nome sugere, são a base da Programação Orientada a Objetos. São estruturas que podem armazenar dados e possuem um comportamento definido pelos métodos da classe a que pertencem.

B.2.3 Mensagens

Uma mensagem é uma forma de comunicação entre objetos, que permite a troca de informações entre esses. Com essa mediação, pode-se criar uma associação entre objetos. Uma mesma mensagem pode ser enviada a objetos diferentes que, devido a seus métodos próprios para tratamento dessa mensagem, poderá responder de forma diferente.

Uma mensagem, para cumprir sua mediação, exige basicamente três componentes:

- Objeto para receber a mensagem
- Nome da operação (método) a ser executado pelo objeto
- Parâmetro (argumento)

B.2.4 Métodos

Como mencionado anteriormente, os dados e funções de uma classe podem ser privados, protegidos ou públicos. A forma mais segura de se manipularem os dados de uma classe é por meio dos métodos de uma classe. Para que um método possa ser acessado em qualquer ponto do código, é necessário que ele seja declarado como público. Se for declarado como privado ou protegido, só pode ser acessado por meio de métodos da própria classe e/ou por métodos de classes derivadas.

B.3 Principais características da programação orientada a objetos

B.3.1 Encapsulamento

O encapsulamento está relacionado à maneira como cada objeto é definido. Tipicamente, essa definição faz parte de uma classe *C++* e inclui uma descrição da estrutura interna do objeto e forma de proteção que isola os detalhes funcionais do objeto em relação ao exterior da classe.

Geralmente, o encapsulamento de dados possui três finalidades:

- Proteger os dados de exposição excessiva
- Ocultar detalhes do armazenamento de dados
- Facilitar a reutilização do código em outro projeto

B.3.2 Herança

Na abstração de classes, há a possibilidade de ocorrer uma conexão semântica entre mãe e filho na qual uma classe filha (subclasse) herda as propriedades de sua mãe (superclasse) direta ou indiretamente. Cada classe 'pode ter suas propriedades particulares herdadas diretamente da classe mãe ou substituídas/mascaradas nessa transição. Assim, somente propriedades diferentes serão declaradas na classe filha. Quando é criada uma nova classe a partir de uma já existente, a nova classe automaticamente herda as características da classe base.

A abstração de uma classe-base estabelece uma interface comum para um grupo de classes semelhantes. A herança é a capacidade de um novo objeto tomar atributos e operações de um objeto existente, permitindo criar classes complexas sem repetir código.

B.3.3 Polimorfismo

Duas situações caracterizam o polimorfismo, e são diferenciadas pelo escopo em que ocorrem.

Numa mesma classe pode-se definir a mesma função mais de uma vez. Essas redefinições variam quanto à quantidade e/ou tipo de argumentos passados. Dependendo dos parâmetros passados, *C++* interpretará qual das definições será utilizada. Por exemplo

Um produto vetorial entre dois vetores pode ser escrito para retomar o resultado final a um terceiro vetar ou, caso o valor do primeiro possa ser perdido, retomar o resultado ao primeiro vetor passado como parâmetro. As duas declarações seriam:

- Cross_Product(double * array _one, double * array _two)

Pode ser implementado para retomar o resultado na variável array - one.

- Cross_Product(double * array _one, double * array _two, double * result)

Pode ser implementado para retomar o resultado na variável resulto

O que diferencia as duas chamadas é a quantidade de parâmetros.

Classes filhas originadas da mesma classe base, poderão conter uma especialização de uma função existente na classe base. Essa redefinição proveniente da especialização da classe base caracteriza também um polimorfismo.

Apêndice C

Formulação utilizada

C.1 Introdução

Nesse capítulo são apresentados os conceitos e as considerações utilizados para desenvolvimento do trabalho. Parte-se dos conceitos de tensão, deformação, a relação entre tem deformação e do conceito de viga utilizados para a implementação dos códigos.

C.2 Deslocamentos

Observa-se na figura C.1, que o deslocamento fundamental de um ponto *P,* contido nesse sólido, pode ser definido pelas três componentes do vetor de deslocamentos de translação.

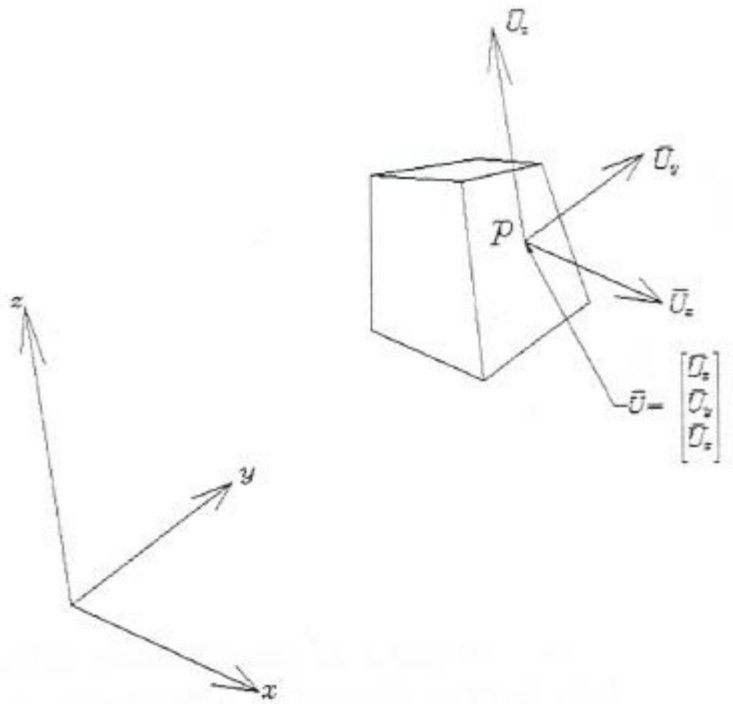

Figura C.1: Sólido tridimensional.

dessa forma, escreve-se o vetor deslocamento *u* como

$$u = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$$

C.3 Deformação

Em problemas de engenharia, nos quais há pequenos deslocamentos de pontos materiais, utiliza-se o conceito de deformação infinitesimal, para o estudo da deformação.

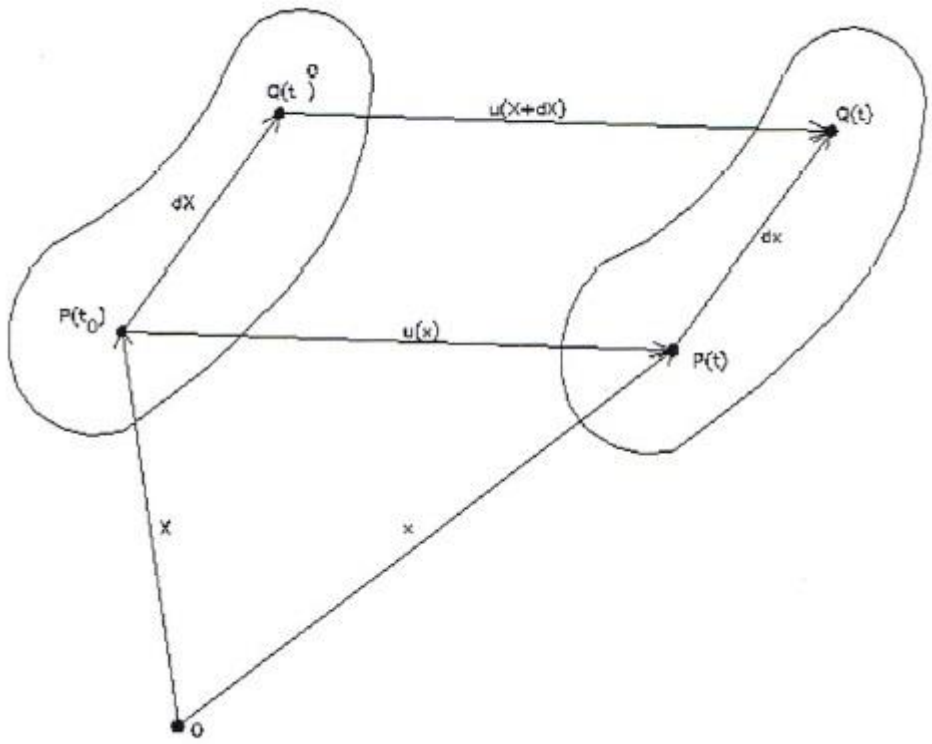

Figura C.2: Deformação de um corpo

Considere-se um corpo B, figura C.2 pagina 116, que tem uma configuração inicial apresenta uma mudança nessa configuração passando a ter uma configuração deformada. Pode-se observar na figura, que o ponto P tem um deslocamento u, passando para a posição:

$$x = X + u(X) \qquad (C.1)$$

Para um ponto Q, na vizinhança de P, em uma posição X+dX que apresenta uma variação de posição x+dx, podem ser relacionadas as mudanças de posição dos pontos materiais pela expressão:

$$x + dx = X + dX + u(X + dX) \qquad (C.2)$$

Subtraindo a expressão (C.1) da expressão (C.2), tem-se que

$$dx = dX + u(x + dX) - u(X) \qquad (C.3)$$

Utilizando-se a definição de gradiente de uma função vetorial, pode-se escrever a equação (C.3) na forma

$$dx = dX + (\nabla u)dX + O(dX) \qquad (C.4)$$

sendo *O(dx)* o resíduo da aproximação.

A definição de gradiente de uma função é dada por

$$dx \equiv u(X + dX) - u(X) \equiv (\nabla u)dX + O(X)$$

O gradiente de *u* é um tensor de segunda ordem, denominado gradiente do deslocamento. Em coordenadas cartesianas retangulares, a representação matricial de ∇u, com $X = X_i e_i$ e $u = u_i e_i$ é

$$[u] = \begin{bmatrix} \frac{\partial u_1}{\partial X_1} & \frac{\partial u_1}{\partial X_2} & \frac{\partial u_1}{\partial X_3} \\ \frac{\partial u_2}{\partial X_1} & \frac{\partial u_2}{\partial X_2} & \frac{\partial u_2}{\partial X_3} \\ \frac{\partial u_3}{\partial X_1} & \frac{\partial u_3}{\partial X_3} & \frac{\partial u_3}{\partial X_3} \end{bmatrix}$$

É possível escrever a equação (C.4), na forma:

$$dx = FdX \qquad (C.5)$$

em que

$$F = I + \nabla u \qquad (C.6)$$

Para determinar a relação entre *ds* e o comprimento *dx* e *dS* e o comprimento *dX,* toma-se o produto interno da equação (C.5), sobre ela mesma, dado por

$$dx \cdot dx = FdX \cdot FdX = dX \cdot F^T F \cdot dX$$

$$(dS)^2 = dX \cdot F^T F \cdot dX \qquad (C.7)$$

Sendo F um tensor ortogonal, o que corresponde a um movimento de corpo rígido, tem-se que $F^T F = I$. Com isto, pode-se escrever a equação (C.7) na forma

$$(ds)^2 = (dS)^2$$

Ainda, da equação (C.6), tem-se

$$F^T F = (I + \nabla u)^T (I + \nabla u) = I + \nabla u + (\nabla u)^T + (\nabla u)^T \nabla u \qquad (C.8)$$

Nesse trabalho, consideraram-se os casos em que as componentes do vetor deslocamento são muito pequenas, pois se trata de deslocamentos infinitesimais. Consequentemente suas derivadas parciais também são muito pequenas. Dessa consideração resulta que são muito pequenos, com relação a ∇u os valores absolutos de todas as componentes de $(\nabla u)^T \nabla u$. Por isso, costuma-se utilizar a seguinte aproximação:

$$F^T F \approx I + \nabla u + (\nabla u)^T \equiv I + 2E \qquad (C.9)$$

onde o tensor E é igual a

$$E = \frac{1}{2}[(\nabla u)^T + \nabla u] \qquad (C.10)$$

O tensor E é denominado *Tensor* de *Deformação Infinitesimal* e suas componentes são obtidas por meio das componentes de gradiente u, descritas em coordenadas retangulares por:

$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i}\right)$$

$$[E] = \begin{bmatrix} \frac{\partial u_1}{\partial X_1} & \frac{1}{2}\left(\frac{\partial u_1}{\partial X_2} + \frac{\partial u_2}{\partial X_1}\right) & \frac{1}{2}\left(\frac{\partial u_1}{\partial X_3} + \frac{\partial u_3}{\partial X_1}\right) \\ \frac{1}{2}\left(\frac{\partial u_1}{\partial X_2} + \frac{\partial u_2}{\partial X_1}\right) & \frac{\partial u_2}{\partial X_2} & \frac{1}{2}\left(\frac{\partial u_2}{\partial X_3} + \frac{\partial u_3}{\partial X_2}\right) \\ \frac{1}{2}\left(\frac{\partial u_1}{\partial X_3} + \frac{\partial u_3}{\partial X_1}\right) & \frac{1}{2}\left(\frac{\partial u_2}{\partial X_3} + \frac{\partial u_3}{\partial X_2}\right) & \frac{\partial u_3}{\partial X_3} \end{bmatrix}$$

Demonstra-se que as componentes E_{ii} do tensor E representam o alongamento ou o encurtamento de uma fibra na direção ei e as componentes E_{ij} representam a metade da mudança de ângulo entre uma fibra alinhada com a direção i e j.

C.4 Tensão

Na mecânica do contínuo, a ideia de tensão é baseada nas hipóteses de *Cauchy*. Para demonstrar o conceito, consideremos um corpo B como o da figura C.3, constituído de duas partes. Para um ponto P, pertencente a um plano o: que corte o corpo B existe uma força por unidade de área. A resultante de força sobre o plano o: é a integral das forças por unidade de área. A força por unidade de área é chamada de tensão.

Ao se separar o corpo B em duas partes, pela lei da ação e reação, teremos duas forças, de mesma intensidade e com sentidos contrários, agindo em cada face do corpo.

Existem três efeitos de forças que podem agir sobre um corpo, que são:

1. Forças de contato entre corpos;

2. Forças aplicadas na superfície;

3. Forças internas ao corpo.

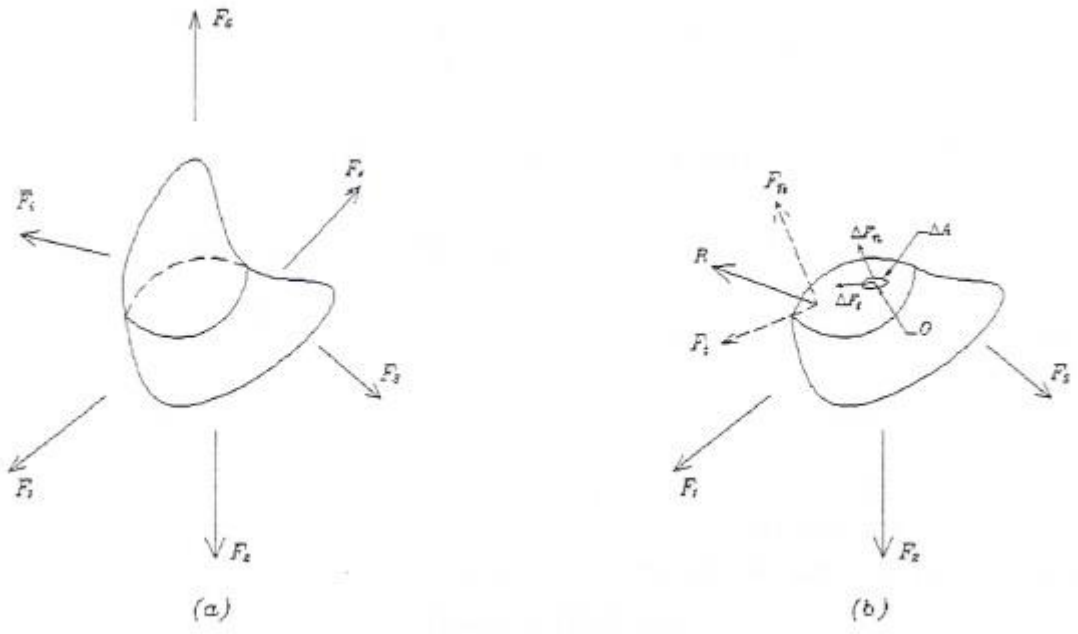

Figura *C.3:* Estado de tensão em um corpo.

Dada a existência do conceito de tensão, é possível calcular uma força resultante em qualquer ponto de um corpo. Ao se isolar uma porção P, em qualquer ponto da superfície dessa porção P existe uma tensão σ_n. A força total que atua em P é a integral sobre o contorno da região P, em que

$$\int_{\partial P} \sigma_n(x) dw$$

O somatório total das forças de volume *b(x)* sobre *P* é

$$\int_P b(x) d\Omega$$

Utilizando o conceito de forças de superfície e forças de volume, podem ser calculadas as forças resultantes sobre P, dado por .

$$f(P) = \int_{\partial P} \sigma_n dw + \int_P b(x) d\,\Omega$$

$$m(P) = \int_{\partial P} r \times \sigma_n dw + \int_P r \times b(x) d\,\Omega$$

Para um corpo estar em equilíbrio é necessário que $f(P) = 0$ e $\mathrm{m(P)}$, para qualquer parte P do corpo.

Demonstra-se que o tensor de tensão tem as seguintes propriedades:

1. Existe um tensor *T(x),* tal que $\sigma_n(P) = T(x)n$*;*

2. O tensor $T(x)$ é simétrico;

3. O tensor $T(x)$ satisfaz a lei de equilíbrio, dada por:

$$divT(x) + b = 0$$

C.5 Relação tensão deformação

A caracterização da relação entre uma medida de tensão e uma medida de deformação em um corpo, é necessária para que se possam resolver problemas de equilíbrio de corpos elásticos. As equações apresentadas nos itens *C.3* e C.4 não são suficientes para determinar a solução para esse tipo de problema, sendo necessário, portanto, a determinação de relações adicionais entre a tensão e a deformação. Para esse propósito, é necessário utilizar um modelo que represente as propriedades elásticas do corpo. Nesse trabalho, considerou-se o caso de pequenas deformações, um corpo contínuo elástico que obedeça ã lei de *Hook.[13]*

C.5.1 Estado de tensão tri dimensional

Para um sólido elástico linear, demonstra-se que a relação constitutiva entre a tensão e a deformação é dada por

$$T = T(E) \qquad (C.11)$$

em que *T* é o tensor de deformações de *Cauchy* e *E* é o tensor de deformações infinitesimais.

Como a relação entre *T* e *E* é linear, pode-se escrever a equação (C.11) como

$$T = CE \qquad (C.12)$$

Sendo C um tensor de quarta ordem, denominado tensor de elasticidade. . Em notação indicial, é escrito por:

$$T_{ij} = C_{ijkl}E_{kl} \qquad (C.13)$$

A equação *(C.13)* possui 81 coeficientes C_{ijkl}. Devido ao fato de que o tensor de deformação *E* e o tensor de tensão *T* são simétricos, tem-se, pela condição de simetria, que [4]:

$$C_{ijkl} = C_{jikl} = C_{ijlk} = C_{jilk} \qquad (C.14)$$

Dessa forma obtém-se uma redução do número de constantes, caindo de 81 para 36.

Assumindo que o conceito de elasticidade é associado com a existência de uma junção energia de deformação $U(E_{ij})$, tal que $T_{ij} = \frac{\partial U}{\partial E_{ij}}$, pode-se mostrar que $C_{ijkl} = C_{klij}$, o que reduz o número de termos independentes de 36 para 21.

Material isotrópico

Para um sólido elástico linear isotrópico, as constantes elásticas da equação *(C.12)* têm de ser as mesmas em qualquer direção. Assim, C_{ijkl} tem de ser um tensor isotrópico de quarta ordem. É possível mostrar que esse tensor pode ser representado, na sua forma mais geral, pela equação

$$C_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu\left(\delta_{ik}\delta_{kl} + \delta_{ik}\delta_{jl}\right) + \alpha\left(\delta_{ik}\delta_{jl} - \delta_{il}\delta_{jk}\right) \qquad (C.15)$$

Onde λ, μ *e* α e a são constantes escalares do material, denominadas *Coeficientes de Lamé,* e δ é o delta de *Kronecker.*

Desde que C_{ijkl} satisfaça as condições de simetria da relação (C.14), tem-se que $\alpha = 0$. Com isso a equação (C.15) se reduz a:

$$C_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu\left(\delta_{ik}\delta_{kl} + \delta_{ik}\delta_{jl}\right) \qquad (C.16)$$

Das equações (C.12) e (C.16), tem-se que

$$T_{ij} = \lambda\delta_{ij}\delta_{kl}\varepsilon_{kl} + \mu\left(\delta_{ik}\delta_{kl} + \delta_{ik}\delta_{jl}\right)\varepsilon_{kl} \qquad (C.17)$$

ou

$$T_{ij} = \lambda\delta_{ij}e_{kk} + 2\mu\varepsilon_{ij} \qquad (C.18)$$

Em que e é $trE = \sum_i e_{ii.}$, $\varepsilon = E$ e as constantes λ e μ verificam a relação $\lambda = \frac{vE}{(1+v)(1-2v)}$, na qual *E* é o módulo de elasticidade do material, e $\mu = G$.

A expressão (C.18) pode ser invertida e assim, obtém-se uma relação da deformação *E* em termos da tensão *T,* dada por [4]

$$\varepsilon_{ij} = \frac{1}{2\mu}\left[T_{ij} - \frac{\lambda}{(2\mu + 3\lambda)}T_{kk}\delta_{ij}\right] \qquad (C.19)$$

Quando se deseja aplicar métodos numéricos para determinar os valores de ε e *T,* é conveniente utilizar uma representação matricial para a relação entre a tensão e a deformação [4]. As componentes da tensão e da deformação podem ser definidas em dois vetores $\{\sigma\}$ e $\{\varepsilon\}$, respectivamente, representados por

$$\{\sigma\} = \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} \quad e \quad \{\varepsilon\} = \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{Bmatrix} \qquad (C.20)$$

Com esses dois vetores, pode-se reescrever a equação (C.12) na forma

$$\{\sigma\} = [D]\{\varepsilon\} \qquad (C.21)$$

Sendo [D] uma matrix denominada *matrix constitutiva elástica*, que tem a seguinte forma:

$$[D] = \frac{E}{(1+v)(1-2v)} \begin{bmatrix} (1-v) & v & v & 0 & 0 & 0 \\ v & (1-v) & v & 0 & 0 & 0 \\ v & v & (1-v) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2v)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2v)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2v)}{2} \end{bmatrix} \qquad (C.22)$$

Na qual v é o coeficiente de *Poisson,* que caracteriza a taxa de deformação nas direções perpendiculares ao sentido de aplicação da tensão, e E é o módulo de elasticidade do material.

As componentes do tensor de deformação podem ser escritas na forma

$$\varepsilon_x = \frac{1}{E}\left[\sigma_x - v(\sigma_y + \sigma_z)\right]$$

$$\varepsilon_y = \frac{1}{E}\left[\sigma_y - v(\sigma_x + \sigma_z)\right]$$

$$\varepsilon_z = \frac{1}{E}\left[\sigma_z - v(\sigma_x + \sigma_y)\right]$$

$$\varepsilon_{yz} = \frac{\tau_{yz}}{2G}$$

$$\varepsilon_{xz} = \frac{\tau_{xz}}{2G}$$

$$\varepsilon_{xy} = \frac{\tau_{xy}}{2G}$$

Na qual $G = \frac{E}{2(1+v)}$ que representa o Módulo de Elasticidade Transversal do material. As componentes da tensão em função das componentes da deformação são

$$\sigma_x = \frac{E\left[\varepsilon_x + v\left(-\varepsilon_x + \varepsilon_y + \varepsilon_z\right)\right]}{(1+v)(-1+2v)}$$

$$\sigma_y = \frac{E\left[\varepsilon_y + v\left(\varepsilon_x - \varepsilon_y + \varepsilon_z\right)\right]}{(1+v)(-1+2v)}$$

$$\sigma_z = \frac{E\left[\varepsilon_z + v\left(\varepsilon_x + \varepsilon_y - \varepsilon_z\right)\right]}{(1+v)(-1+2v)}$$

$$\tau_{xy} = 2G\varepsilon_{xy}$$
$$\tau_{yz} = 2G\varepsilon_{yz}$$
$$\tau_{zx} = 2G\varepsilon_{zx}$$

Material ortotrópico

Para um sólido linear ortotrópico, demonstra-se que, a partir da equação (C.12) e da relação descrita em (C.14), é possível reduzir o número de coeficientes independentes de 21 para 9. Tal manipulação é possível pois o material ortotrópico apresenta simetria em três planos mutuamente perpendiculares entre si e com relação aos eixos *x, y* e *z,* o que reduz o número de coeficientes [4]. Além disso, pela condição de simetria de *Green* para um material elástico, tem-se que

$$C_{(ij)(kl)} = C_{(kl)(ij)} \qquad (C.23)$$

Com essa relação pode-se reduzir o número de constantes independentes para 9.

Dessa maneira, a representação matricial da equação (C.21) entre a tensão e a deformação fica assim:

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ & C_{22} & C_{23} & 0 & 0 & 0 \\ & & C_{33} & 0 & 0 & 0 \\ & & & C_{44} & 0 & 0 \\ & S & & & C_{55} & 0 \\ & & & & & C_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{Bmatrix} \qquad (C.24)$$

Para estudos de engenharia, pode-se utilizar uma outra forma de representação matricial dada por

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{Bmatrix} = \begin{bmatrix} \frac{1}{E_x} & -\frac{v_{yx}}{E_y} & -\frac{v_{zx}}{E_z} & 0 & 0 & 0 \\ & \frac{1}{E_y} & -\frac{v_{zy}}{E_z} & 0 & 0 & 0 \\ & & \frac{1}{E_z} & 0 & 0 & 0 \\ & & & \frac{1}{2G_{xy}} & 0 & 0 \\ & S & & & \frac{1}{2G_{yz}} & 0 \\ & & & & & \frac{1}{2G_{zx}} \end{bmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{Bmatrix} \qquad (C.25)$$

Na qual E_x, E_y, E_z representam o Módulo de *Young* nas direções *x, y,* e *z,* respectivamente, $G_{xy}, G_{yz}\ e\ G_{zx}$representam o Módulo de elasticidade transversal, paralelos aos planos $x-y, y-z\ e\ x-y$ e $v_{ij}(i,j=x,y,z)$ representam o Coeficientes de *Poisson,* que caracterizam a taxa de deformação na direção perpendicular ao sentido de aplicação da tensão.

Devido à simetria de Green para material elástico, tem-se

$$\begin{aligned} E_x v_{yx} &= E_y v_{xy} \\ E_y v_{zy} &= E_z v_{yz} \qquad (C.26) \\ E_z v_{xz} &= E_x v_{zx} \end{aligned}$$

As relações (C.25) contém 12 constantes elásticas, porém apenas 9 são realmente independentes, devido às relações determinadas em (C.26).

Com isto, as componentes da deformação podem ser escritas como:

$$\varepsilon_x = \frac{\sigma_x}{E_x} - \frac{v_{yx}}{E_y}\sigma_y - \frac{v_{zx}}{E_z}\sigma_z$$

$$\varepsilon_y = -\frac{v_{xy}}{E_x}\sigma_x + \frac{\sigma_y}{E_y} - \frac{v_{zy}}{E_z}\sigma_z$$

$$\varepsilon_z = -\frac{v_{zx}}{E_x}\sigma_x - \frac{v_{zy}}{E_y}\sigma_y + \frac{\sigma_z}{E_z}$$

$$\varepsilon_{yz} = \frac{\tau_{xy}}{2G_{xy}}$$

$$\varepsilon_{xz} = \frac{\tau_{yz}}{2G_{yz}}$$

$$\varepsilon_{xz} = \frac{\tau_{zx}}{2G_{zx}}$$

As expressões para a determinação da tensão em função da deformação são

$$\sigma_x = \frac{-E_x\left[\varepsilon_x\left(1 - v_{yz}^2\right) + \varepsilon_y\left(v_{yx} + v_{yz}v_{zx}\right) + \varepsilon_z\left(v_{zx} + v_{yx}v_{zy}\right)\right]}{-1 + v_{xy}^2 + v_{yz}^2 + v_{zx}^2 + 2v_{xy}v_{yz}v_{zx}}$$

$$\sigma_y = \frac{-E_y\left[\varepsilon_y(1 - v_{xz}^2) + \varepsilon_x\left(v_{xy} + v_{xz}v_{zy}\right) + \varepsilon_z\left(v_{zy} + v_{xy}v_{zx}\right)\right]}{-1 + v_{xy}^2 + v_{yz}^2 + v_{zx}^2 + 2v_{xy}v_{yz}v_{zx}}$$

$$\sigma_y = \frac{-E_y\left[\varepsilon_y(1 - v_{xz}^2) + \varepsilon_x\left(v_{xy} + v_{xz}v_{zy}\right) + \varepsilon_z\left(v_{zy} + v_{xy}v_{zx}\right)\right]}{-1 + v_{xy}^2 + v_{yz}^2 + v_{zx}^2 + 2v_{xy}v_{yz}v_{zx}}$$

$$\tau_{xy} = 2G_{xy}\varepsilon_{yz}$$

$$\tau_{yz} = 2G_{yz}\varepsilon_{xz}$$

$$\tau_{zx} = 2G_{zx}\varepsilon_{xy}$$

C.5.2 Estado plano de tensão

A deformação no estado plano de tensão é obtida a partir da deformação no estado triaxial de tensão, substituindo-se $\sigma_z = \tau_{yz} = \tau_{zx} = 0$ na equação (C.20).

Material isotrópico

Para um sólido linear composto de material isotrópico, submetido a um estado plano de tensões, pode-se demonstrar, a partir da equação (C.21), que a relação entre a tensão e a deformação é dada por

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \end{Bmatrix} = \frac{1}{E}\begin{bmatrix} 1 & -v & 0 \\ -v & 1 & 0 \\ 0 & 0 & 2(1+v) \end{bmatrix}\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} \qquad (C.27)$$

e

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{Bmatrix} = \frac{E}{(1-v^2)}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \dfrac{(1-v)}{2} \end{bmatrix}\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \end{Bmatrix} \qquad (C.28)$$

Tem-se então as seguintes expressões para as componentes da deformação:

$$\varepsilon_x = \frac{1}{E}\left(\sigma_x - v\sigma_y\right)$$

$$\varepsilon_y = \frac{1}{E}\left(\sigma_y - v\sigma_x\right)$$

$$\varepsilon_z = -\frac{v}{E}\left(\sigma_x + \sigma_y\right)$$

$$\varepsilon_{xy} = \frac{\tau_{xy}}{2G}$$

$$\varepsilon_{yz} = \frac{\tau_{yz}}{2G}$$

$$\varepsilon_{zx} = \frac{\tau_{zx}}{2G}$$

É importante notar que a componente da deformação ε_z não é nula, sendo uma função linear das componentes ε_x e ε_y.

As componentes da tenção em função das componentes da deformação são:

$$\sigma_x = \frac{E\left(\varepsilon_x + v\varepsilon_y\right)}{\left(-1 + v^2\right)}$$

$$\sigma_y = \frac{E\left(\varepsilon_y + v\varepsilon_x\right)}{\left(-1 + v^2\right)}$$

$$\tau_{xy} = 2G\varepsilon_{xy}$$

$$\tau_{yz} = 2G\varepsilon_{yz}$$

$$\tau_{zx} = 2G\varepsilon_{zx}$$

Material ortotrópico

Para um corpo constituído de material ortotrópico, submetido a um estado plano de tensões, demonstra-se que, a partir de equação *(C.25),* a relação entre tensão e deformação é fornecida pela expressão:

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{Bmatrix} = \begin{bmatrix} \frac{1}{E_x} & -\frac{v_{yx}}{E_y} & -\frac{v_{zx}}{E_z} & 0 & 0 & 0 \\ & \frac{1}{E_y} & -\frac{v_{zy}}{E_z} & 0 & 0 & 0 \\ & & \frac{1}{E_z} & 0 & 0 & 0 \\ & & & \frac{1}{2G_{xy}} & 0 & 0 \\ & S & & & \frac{1}{2G_{yz}} & 0 \\ & & & & & \frac{1}{2G_{zx}} \end{bmatrix} \begin{Bmatrix} \sigma_x \\ \sigma_y \\ 0 \\ \tau_{xy} \\ 0 \\ 0 \end{Bmatrix} \qquad (C.29)$$

As expressões para as componentes da deformação são dadas por

$$\varepsilon_x = \frac{\sigma_x}{E_x} - \frac{v_{yx}}{E_y}\sigma_y$$

$$\varepsilon_y = -\frac{v_{xy}}{E_x}\sigma_x + \frac{\sigma_y}{E_y}$$

$$\varepsilon_z = -\frac{v_{xz}}{E_x}\sigma_x - \frac{v_{yx}}{E_y}\sigma_y$$

$$\varepsilon_{yz} = \frac{\tau_{xy}}{2G_{xy}}$$

$$\varepsilon_{xz} = \frac{\tau_{yz}}{2G_{yz}}$$

$$\varepsilon_{xy} = \frac{\tau_{zx}}{2G_{zx}}$$

A relação entre as tensões e as deformações para esse caso são dadas pelas expressões:

$$\sigma_x = \frac{E_x\left(\varepsilon_x + v_{yx}\varepsilon_y\right)}{\left(-1 + v_{xy}^2\right)}$$

$$\sigma_y = \frac{E_y\left(\varepsilon_y + v_{xy}\varepsilon_x\right)}{\left(-1 + v_{xy}^2\right)}$$

$$\tau_{xy} = 2G_{xy}\varepsilon_{yz}$$

$$\tau_{yz} = 2G_{yz}\varepsilon_{xz}$$

$$\tau_{zx} = 2G_{zx}\varepsilon_{xy}$$

C.5.3 Estado uniaxial de tensão

Em um estado uniaxial de tensão, tem-se $\sigma_y = \sigma_z = \tau_{yz} = 0$. Com isto na equação (C.19), para $T_{22} = T_{33} = 0$, tem-se que as expressões da deformação em função da tensão são calculadas por

$$\varepsilon_x = \frac{\sigma_x}{E}$$

$$\varepsilon_y = -\frac{\nu\sigma_x}{E}$$

$$\varepsilon_z = -\frac{\nu\sigma_x}{E}$$

$$\varepsilon_{xy} = \frac{\tau_{xy}}{2G}$$

$$\varepsilon_{yz} = \frac{\tau_{yz}}{2G}$$

$$\varepsilon_{zx} = \frac{\tau_{zx}}{2G}$$

Finalmente, a relação entre a tensão e as componentes da deformação são obtidas pelas expressões

$$\sigma_x = E\varepsilon_x$$

$$\tau_{xy} = 2G\varepsilon_{xy}$$

$$\tau_{yz} = 2G\varepsilon_{yz}$$

$$\tau_{zx} = 2G\varepsilon_{yz}$$

C.6 Princípio dos trabalhos virtuais

Seja um corpo deformável n, como ilustra a figura C.4, cujo contorno é dado por Γ, submetido a um conjunto de forças de corpo F_i no domínio de Ω, de forças T_i no contorno Γ_A e de restrições aos deslocamentos no contorno Γ_B.

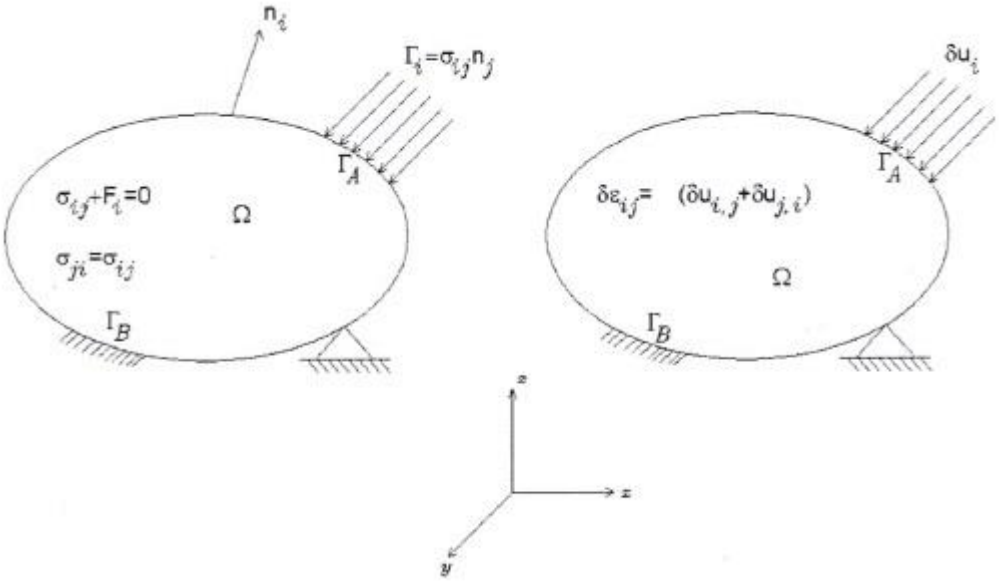

Figura *C.4:* Aplicação do PTV para um corpo em equilíbrio.

As equações de equilíbrio desse corpo, em uma dada direção, podem ser representadas pelas equações:

$$\sigma_{ij,j} + F_i = 0 \; em \; \Omega$$

$$\sigma_{ij} = \sigma_{ji} \qquad (C.30)$$

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 a fórmula (C.30) seja satisfeita quando sujeita às condições de contorno abaixo descritas, em que h_i são as componentes dos deslocamentos prescritos no contorno Γ_B e n_i são as componentes do vetor normal à superfície Γ.

$$u_i = h_i \; em \; \Gamma_B$$

$$\sigma_{ij} n_j = T_i \; em \; \Gamma_A \qquad (C.31)$$

C.6.1 Equação do princípio dos trabalhos virtuais

O termo virtual é utilizado para designar o trabalho realizado por forças externas *(ditas reais)* com uma variação de deslocamentos *(ditos virtuais).*

Admitindo um estado de equilíbrio de forças e um estado independente de deslocamentos compatíveis, e aplicando-se o *Princípio dos Trabalhos Virtuais* entre esses estados, tem-se que:

$$\int_V (\sigma_{ij,j} + F_i)\, \delta u_i dV = 0$$

$$\int_V (\sigma_{ij}\delta u_i)dV = \int_V (\sigma_{ij}\delta u_i)_{,j} - \sigma_{ij}\delta u_{i,j}dV$$

Aplicando-se o teorema da divergência [4], tem-se

$$\int_A \sigma_{ij}n_j\delta u_i\, dA - \int_v \sigma_{ij}\delta\varepsilon_{ij}dV + \int_v F_i\delta u_i dV = 0$$

$$\int_v \sigma_{ij}\delta\varepsilon_{ij}dV = \int_A \sigma_{ij}n_j\delta u_i\, dA + \int_v F_i\delta u_i dV \qquad (C.32)$$

Em que as deformações $\delta\varepsilon_{ij}$ representam um conjunto de deformações calculadas com base em um deslocamento virtual δu_i, para os pontos de aplicação das forças externas T_i e F_i.

$$\sigma_{ij}n_j = T_i \qquad (C.33)$$

Substituindo-se (C.33) em (C.32), tem-se o trabalho externo das forças para uma variação de deslocamentos

$$\int_v \sigma_{ij}\delta\varepsilon_{ij}dV = \int_A T_i\delta u_i dA + \int_v F_i\delta u_i dV \qquad (C.34)$$

Onde as quantidades T_i e F_i são forças de superfície e de corpo, respectivamente.

Na equação (C.34), os termos da direita e da esquerda da igualdade representam, respectivamente,

$$W_{ext} = \int_A T_i\delta u_i dA + \int_v F_i\delta u_i dV \qquad (C.35)$$

$$W_{ext} = \int_V \sigma_{ij}\delta u_{i,j}dV \qquad (C.36)$$

C.7 Elemento de barra

Entende-se por uma barra, um elemento estrutural, tridimensional, no qual observa-se que uma das três dimensões é muito maior do que as outras duas. Devido a essa característica principal, uma barra pode ser considerada como um elemento unidimensional e o seu comportamento pode ser analisado ao longo do sentido paralelo à sua direção maior. Nesse trabalho, orientou-se o sistema de coordenadas com a direção *x* paralela à direção principal, conforme ilustra a Figura C.5.

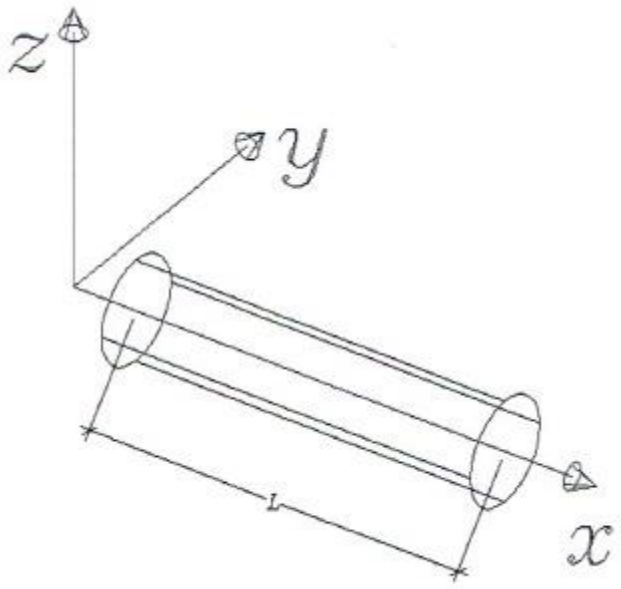

Figura *C.5:* Elemento de barra.

Finalmente, os exemplos apresentados representam apenas uma das inúmeras possibilidades de desenvolvimento de ferramentas que se pode disponibilizar para o trabalho dos profissionais da área de Engenharia.

Com o intuito de melhorar o que já foi feito até o presente momento, seria muito útil trabalhar no desenvolvimento dos seguintes tópicos:

- Criação de um elemento unidimensional para ser posicionado entre um elemento de placa e um elemento de barra para permitir modificar a vinculação entre uma barra e uma placa;

- Criar um elemento pontual entre uma barra e um nó para controlar a vinculação entre esses elementos;

- Criar a representação do offset entre os elementos de barra e de placa;

- Desenvolver um conjunto de caixas de diálogo para facilitar a utilização do programa.