UNIVERSIDADE FEDERAL DE MINAS GERAIS

Escola de Engenharia

Departamento de Engenharia de Estruturas

Curso de Pós-Graduação em Engenharia de Estruturas

IMPLEMENTAÇÃO ORIENTADA A OBJETOS DA SOLUÇÃO DE PROBLEMAS ESTRUTURAIS DINÂMICOS VIA MÉTODO DOS ELEMENTOS FINITOS

Leibnitz Germanio

Dissertação apresentada ao curso de Pós-Graduação em Engenharia de Estruturas da UNIVERSIDADE FEDERAL DE MINAS GERAIS, como parte dos requisitos para obtenção do título de MESTRE EM ENGENHARIA DE ESTRUTURAS

Orientador: Roque Luiz da Silva Pitangueira

Co-Orientador: Fernando Amorim de Paula

Belo Horizonte

AGOSTO DE 2005

RESUMO

Freqüentemente, ao se propor uma expansão de um modelo numérico-computacional de engenharia, é necessário implementar todo o código para a solução do problema, e não somente a parte específica relativa ao estudo. Isso ocorre principalmente devido à utilização de tecnologias inadequadas que não permitem a fácil separação entre diferentes partes do programa.

O INSANE é o projeto de um aplicativo de propósito geral para problemas de engenharia desenvolvido por um grupo de pesquisadores ligados ao departamento de Engenharia de Estruturas da UFMG. O INSANE adotou a linguagem JAVA, que suporta o paradigma da Programação Orientada a Objetos. A programação orientada a objetos permite a modularidade do programa, facilitando o escalonamento em complexidade no desenvolvimento do programa.

No estágio atual, o projeto INSANE possui uma implementação para solução de problemas estruturais estáticos via Método dos Elementos Finitos. O trabalho que aqui se apresenta implementa uma expansão do INSANE para solução de problemas estruturais dinâmicos. Essa expansão trata da implementação de métodos para solução de modal de sistemas algébricos, integração numérica da equação do movimento utilizando processos de decomposição modal e métodos para integração direta da equação do movimento.

A implementação feita é validada comparando-se as soluções de problemas escolhidos com aquelas obtidas utilizando a classe de elementos finitos paramétricos do INSANE, com os resultados disponíveis na literatura.

ABSTRACT

Often when trying to develop an expansion of a numerical-computational model for engineering problems, the researcher has to implement the whole code for the numerical solution, not only the part respective to the study. This occurs mainly because the use of not suitable technologies, which does not permit an easy separation of the code parts.

The INSANE project is a general purpose application for the engineering field developed by a research group associated with the UFMG Structural Engineering department. The project has adopted the JAVA programming language, which supports the Object Oriented Paradigm. The Object Oriented programming permits the program separation into modules, easing the program build up on complexity.

On its current status, the INSANE project has an implementation of a solver for static structural problems with the Finite Element Method. The work here presented implements an expansion in the INSANE to deal with dynamic structural problems. This expansion accounts for the implementation of modal solvers for the resulting algebraic equations, numerical integrators of the equation of motion based on the modal decomposition and direct integration methods of the equation of motion.

The implementation was validated comparing the solution of selected problems, which were solved using the parametric finite element class of the INSANE, against the results available in the literature.

À minha família, especialmente aos meus pais.

AGRADECIMENTOS

Pelo empenho, pelo comprometimento, pela amizade, pelo incentivo, pelos vários finais de semana não gozados com sua família, um agradecimento muito especial ao meu orientador Roque Pitangueira.

Pela contribuição e participação em momentos importantes, ao meu co-orientador Fernando Amorim de Paula.

Ao intangível, imensurável, inexpugnável, magnífico e assustador, o Desconhecido.

À minha família por propiciar os meus momentos mais alegres.

Aos meus colegas Insanos.

A todos os meus amigos de faculdade, em especial ao Carlos Alberto, Bruno Chumbinho, Marcelo Lucas, Roberta Ferraz, Plínio Bueno e Felipe Augusto.

A todos os professores que me ajudaram nessa caminhada, em especial aos professores Utsch, Rogério Ribeiro, Kouri, Gilcione, Denise, Marco Túlio e Ulisses.

Aos meus amigos da EMBRAER pelo apoio e incentivo

ÍNDICE ANALÍTICO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SÍMBOLOS

Números

1, 2 e 3 eixos coordenados cartesianos.

Alfabeto Latino

a	coeficiente de Rayleigh
$\overline{a}$	constante de amortecimento viscoso médio
a_{cr}	amortecimento viscoso crítico
a_{r1}	primeira constante do amortecimento de Rayleigh
a_{r2}	segunda constante do amortecimento de Rayleigh
$\mathbf{b}$	força de corpo por unidade de massa
c_i	constante de normalização dos autovetores
c_{ij}	componentes do tensor da lei de Hooke
$\mathbf{f_e}$	vetor força externa sobre uma partícula
f_{ep}	componente do vetor de forças externas principal
f_{est}	força de amortecimento estrutural
f_v	força viscosa em um sistema unidimensional
g	funções arbitrárias
i	índice, número irracional puro
j	índice
k	rigidez
k_p	rigidez principal
m	massa
n	número de modos naturais de um sistema discreto
r	vetor deslocamento ao longo de um circuito fechado, razão de freqüências

t	tempo
u	deslocamento nodal
$\mathbf{u_{pe}}$	vetor de deslocamento nodal pseudo-estático
x	posição ao longo de barras; eixo coordenado.
$\mathbf{x}$	vetor coordenadas
$\mathbf{x_s}$	vetor coordenadas ao longo da fronteira
y	eixo coordenado
z	eixo coordenado
$\mathbf{z}$	vetor posição, vetor deslocamento de partícula.
$\hat{z}$	função de deslocamento aproximada
A	área da seção transversal
$\mathbf{A}$	matriz de amortecimento viscoso
$\mathbf{A^e}$	matriz de amortecimento viscoso do elemento
$\mathbf{A_p}$	matriz de amortecimento viscoso principal
$\mathbf{B}$	vetor derivada da função de forma
$\mathbf{D}$	matriz constitutiva do material, matriz dinâmica
E	módulo de elasticidade
$\mathbf{F_d}$	vetor de forças dissipativas
$\mathbf{F_{ep}}$	vetor de forças externas principal
$\mathbf{F_{eq}}$	vetor de forças externas equivalentes
$\mathbf{F_{eq}^e}$	vetor de forças externas equivalentes do elemento
$\tilde{\mathbf{F}}$	vetor dos pseudo carregamentos do método de *Newmark-*β
$\tilde{\mathbf{F}}_{\alpha}$	vetor dos pseudo carregamento do método de *Newmark-*β com a modificação *Hilber-*α
$\hat{\mathbf{F}}_{\mathbf{ep}}$	vetor de forças principais truncado
I	Inércia rotacional de área
$\mathbf{I}$	matriz identidade
$\mathbf{J}$	matriz jacobiana
$\mathbf{K}$	matriz de rigidez
$\mathbf{K^e}$	matriz de rigidez do elemento
$\mathbf{K_p}$	matriz de rigidez principal

$\tilde{\mathbf{K}}$ pseudo matriz de rigidez do método de *Newmark-*β

$\tilde{\mathbf{K}}_{\alpha}$ pseudo matriz de rigidez do método de *Newmark-*β com a modificação *Hilber-*α

$\overline{\mathbf{K}}$ matriz de rigidez do problema de subespaço

L comprimento

$\mathbf{M}$ matriz de massa

$\mathbf{M}^{\mathbf{e}}$ matriz de massa do elemento

$\overline{\mathbf{M}}$ matriz de massa do problema de subespaço

N número de modos truncado

$\mathbf{N}$ vetor das funções de forma do elemento finito

$\mathbf{Q}$ autovetor do problema de subespaço, matriz auxiliar nos métodos de integração direta

R subespaço

$\mathbf{R}$ matriz auxiliar nos métodos de integração direta

T_n período natural

$\mathbf{U}$ vetor de deslocamentos nodais no método de integração direta

V volume

W peso de integração

Alfabeto Grego

α variável do método *Hilber-*α*,* ângulo de fase de sistemas amortecidos

β variável do método *Newmark-*β

χ coordenada normalizada

ε deformação

ϕ componente do autovetor normalizado

$\boldsymbol{\varphi}$ autovetor normalizado

$\overline{\boldsymbol{\varphi}}$ vetor modal tentativa da iteração inversa

$\overline{\overline{\boldsymbol{\varphi}}}$ vetor modal não normalizado da iteração inversa

γ variável do método *Newmark-*β

η coordenada principal

$\hat{\lambda}$ variável auxiliar para translação espectral

$\overline{\lambda}$ autovalor do método da iteração inversa

$\overline{\hat{\lambda}}$ variável $\hat{\lambda}$ após sofrer uma translação espectral

λ autovalor

μ constante de translação espectral

ν coeficiente de Poisson

θ coordenada normalizada

θ_x rotação ao longo do eixo *x*

θ_y rotação ao longo do eixo y

θ_z rotação ao longo do eixo z

ρ densidade

σ tensão

τ variável de integração na integral de *Duhamel*

υ componente do autovetor não normalizado

ω_d freqüência natural amortecida

ω_n freqüência natural

ξ coordenada normalizada

ζ razão de amortecimento

Φ Matriz modal normalizada

$\hat{\mathbf{\Phi}}$ matriz modal truncada

$\overline{\hat{\mathbf{\Phi}}}$ matriz modal tentativa truncada

$\overline{\overline{\hat{\mathbf{\Phi}}}}$ matriz modal truncada não normalizada

$\hat{\boldsymbol{\eta}}$ vetor de coordenadas principais truncado

Λ Matriz espectral

$\hat{\mathbf{\Lambda}}$ matriz espectral truncada

$\mathbf{Y}$ Autovetor não normalizado

Ω freqüência de carregamento harmônico externo

Z matriz das razões de amortecimento principais

$\hat{\mathbf{Z}}$ matriz das razões de amortecimento principais truncada

GLOSSÁRIO

- API (Application Programming Interface) - Aplicativo da linguagem Java. Muitas APIs são disponibilizadas pela própria SUN Microsystems. Essa grande disponibilidade de APIs é um dos pontos fortes da estratégia de difusão da linguagem Java.
- C1, C2... : classe de derivação da função, onde o número indica quantas vezes a função pode ser derivada.
- debug – processo de eliminação de erros do código compilado.
- DTD (Data Type Definition) – Arquivo utilizado em linguagens de marcações como a GML, SGML e XML para definir a visualização do conteúdo de um documento. Dentro de uma DTD, são criadas regras para exibição dos dados dos arquivos a partir dos campos existentes nos mesmos.
- Elemento finito L2 – elemento com dois graus de liberdade na direção do eixo
- Elemento finito L3 – elemento finito paramétrico com três nós e três graus de liberdade ao longo do eixo do elemento
- Elemento finito Q4 – elemento finito plano com 4 nós e dois graus de liberdade por nó.
- Elemento finito Q8 – elemento finito plano com 8 nós e dois graus de liberdade por nó.
- GDL – Graus De Liberdade
- General Markup Language (GML) - Linguagem para arquivo computacional com marcações para identificação dos campos de um documento. Nos documentos impressos, tradicionalmente, o visual da informação é importante para contextualização da mesma. Essa característica não é adequada para arquivos eletrônicos. Na GML, marcações são utilizadas para identificar o início e o fim de um campo, bem como qual é o contexto do mesmo (título, parágrafo, nota de rodapé, etc).
- Hipertexto – textos com chamadas automáticas a elementos externos.
- HTML – HiperText Markup Language: linguagem simplificada para arquivo computacional derivada da SGML. Essa linguagem popularizou o conceito de hipertexto da SGML e GML.

- Modularidade: Divisão do programa em partes que funcionam independentemente. A remoção e inserção de um módulo não afeta o funcionamento interno dos demais módulos.
- nGDL – n graus de liberdade onde n é um número qualquer.
- Paradigma Funcional (veja antes *Paradigma de Programação*): modelo de programação baseado na teoria de funções matemáticas. O código como um todo como cada uma das suas partes até a mais elementar delas trabalha retornando um resultado de um conjunto contradomínio a partir de um valor de entrada no conjunto domínio. O conceito de variável é mais abstrato uma vez que, para esse paradigma, um dado é definido como uma função que retorna o seu conteúdo quando evocado.
- Paradigma Imperativo (veja antes *Paradigma de Programação*): modelo de programação que basicamente possui variáveis e operações. As variáveis armazenam dados enquanto as operações modificam os dados armazenados nas variáveis.
- Paradigma Lógico (veja antes *Paradigma de Programação*): modelo de programação no qual a solução de um problema, que deve ser declarado como uma proposição lógica, é realizada através da busca na lista de axiomas do programa que satisfaça a proposição lógica.
- Paradigma de Orientação a Objetos ou Orientação a Objetos (veja antes *Paradigma de Programação*) – metodologia para elaboração de modelos na qual os dados existentes no modelo ficam contidos (encapsulados) em objetos. O acesso à informação de cada objeto por outros objetos não é direto, mas indiretamente através de métodos definidos pelo objeto detentor da informação.
- Paradigma de Programação: modelo que define regras para montagem do código. Nos paradigmas utilizados na computação atualmente pode se dizer que as regras básicas estão relacionadas ao acesso aos dados e à forma de execução de operações.
- Plataforma – no ambiente computacional, esse termo não é definido de forma rigorosa, podendo ter muitos significados. No escopo desse trabalho, plataforma é qualquer aplicativo que é base para outros aplicativos, podendo ser um sistema operacional de uma máquina, uma interface gráfica, um editor de texto que trabalha com arquivos eletrônicos diferentes, etc.

- Programação Orientada a Objetos (POO) – geração de códigos computacionais baseados no paradigma de orientação a objetos.
- Programação Estruturada: Forma de programação no qual a seqüência de elaboração do código é a mesma da seqüência de execução do código.
- SGML – Standard General Markup Language – uma evolução a partir da GML
- solução modal – no escopo desse trabalho o termo solução modal foi adotado como sendo a obtenção dos deslocamentos na direção global a partir da integração dos deslocamentos nas coordenadas principais.
- XML – eXtensible Markup Language: Linguagem para arquivo computacional derivado da HTML.

Capítulo 1

INTRODUÇÃO

O lançamento em 1877 por *John William Strutt*, o *Lorde Rayleigh*, do seu *Theory Of Sound* pode ser considerado o marco da consolidação das bases físicas e matemáticas da dinâmica de estruturas. Entretanto, o desenvolvimento científico e tecnológico que vem ocorrendo desde então contribui para manter e renovar o interesse sobre essa área da ciência e da engenharia.

A princípio, o surgimento de máquinas e equipamentos com partes sujeitas a movimentos cíclicos de grandes velocidades demandou o estudo de vibrações mecânicas. Praticamente ao mesmo tempo, o crescimento do porte das construções, conjugado com o emprego de materiais mais leves, tornaram-nas mais flexíveis, demandando o aprimoramento das análises de tais estruturas. No primeiro quarto do Século XX, o surgimento da indústria aeronáutica impulsionou o estudo da dinâmica de estruturas devido ao uso intensivo de estruturas leves, complexas, submetidas a carregamentos essencialmente dinâmicos.

A partir da década de 1950, a engenharia de estruturas como um todo passou a sofrer uma verdadeira revolução com o desenvolvimento dos computadores digitais e de novas técnicas associadas ao seu uso. É nesse contexto que surge o método dos elementos finitos. Por volta de 1970, essa técnica já estava consolidada e seu uso bastante popularizado na engenharia de estruturas.

A partir desse ponto, o desenvolvimento da dinâmica de estruturas passa a se dar pelos mais diversos motivos. O desenvolvimento tecnológico de tradicionais áreas afins ainda era acentuado; o aperfeiçoamento da instrumentação possibilitou o melhor monitoramento dos fenômenos, ao mesmo tempo em que técnicas como o ultra-som ampliavam o emprego da dinâmica e das vibrações mecânicas. Paralelamente, a capacidade de processamento dos computadores digitais aumentava exponencialmente e novos paradigmas eram erigidos. Por exemplo, a ergonomia de máquinas e edificações passou a ser um aspecto relevante nos projetos.

Todos esses fatos fazem do estudo da dinâmica e da dinâmica das estruturas um tema bastante atual e amplo. Alguns tópicos, porém, merecem especial interesse:

i. Estudo das propriedades mecânicas dos materiais.

ii. Modelagem de fenômenos como atrito, impacto, transmissibilidade em juntas e uniões.

iii. Não linearidades físicas e geométricas, efeitos de plasticidade.

iv. Escala dos problemas numérico-computacionais.

Alguns desses tópicos não são exclusivos da dinâmica de estruturas, mas são de especial importância para esta. As propriedades dos materiais necessárias para se realizar uma análise dinâmica de estruturas são, em geral, mais difíceis de se obter do que para um problema similar estático. Além disso, como no problema dinâmico também se deve trabalhar com a dimensão temporal, a escala do problema numérico-computacional é geralmente maior.

É notório que o trabalho em engenharia de estruturas se tornou bastante vinculado à análise numérica computacional. Ao longo desses aproximadamente 50 anos da revolução ocasionada pela computação digital, diversos aplicativos foram desenvolvidos para tratar os mais variados problemas, inclusive pacotes comerciais de grande aceitação como o ANSYS, ABAQUS, ADYNA, CATIA, NASTRAN, entre outros. A utilização de tais pacotes apresenta muitas vantagens, dentre as quais pode-se citar:

i. Suportados por grandes equipes.

ii. Alta confiabilidade.

iii. Interfaces amigáveis que, em geral, conciliam as três fases básicas da solução de problemas: pré-processamento, solução e pós-processamento.

iv. Trabalham com um número grande de problemas.

v. Multidisciplinares.

vi. Facilidade de difusão devido à estrutura comercial dos projetos.

Entretanto, a programação computacional para análise de problemas estruturais está longe de ser esgotada. Primeiramente, há uma acentuada evolução científica e tecnológica da engenharia de estruturas e de áreas afins (cálculo numérico, computação digital, ciência da computação, engenharia de materiais, instrumentação, etc) possibilitam soluções aprimoradas para problemas já conhecidos e abrem a possibilitam o tratamento de problemas novos com complexidades adicionais. Além disso, existem dificuldades relacionadas à utilização de pacotes comerciais, sendo algumas delas crônicas:

i. Ineficiência: o propósito geral desses programas os torna não-ótimos.

ii. Custo elevado, às vezes proibitivo para países em desenvolvimento, centro de estudos (principalmente os não comerciais), etc. Além disso, para se obter a licença de utilização do software é necessário pagar por todo o pacote, mesmo que se utilize alguns poucos recursos.

iii. Demora no desenvolvimento e na solução de falhas (*bugs*).

iv. Portabilidade: os programas demandam altos recursos computacionais e com freqüência dependem de plataformas específicas.

v. Documentação e transparência: devido ao propósito comercial dos programas, a documentação não é completa por motivos óbvios. Os usuários não têm detalhes do comportamento do software. Isso também cria nos usuários uma dependência em relação aos detentores do software.

Ao se desenvolver um novo aplicativo para problemas de dinâmica de estruturas, é pertinente identificar quais são os recursos computacionais disponíveis para implementação de um aplicativo para análise dinâmica de estruturas. Os recursos computacionais que servem de critério para a definição de um projeto são:

i. Recursos físicos: capacidade de armazenamento e capacidade de processamento dos computadores. A capacidade computacional já foi o aspecto fundamental na ciência da computação. Embora continue sendo um aspecto importante, atualmente não é o principal aspecto a ser considerado em muitos casos práticos.

ii. Recursos computacionais previamente disponibilizados: atualmente, apenas aplicativos simples, em geral para emprego em computadores de uso dedicado, são feitos de

maneira a gerenciar toda a máquina. Em computadores de uso geral, vários aplicativos são necessários, simultaneamente ou não, para executar uma tarefa.

iii. Recursos humanos: é um aspecto amplo e que muitas vezes é conseqüência dos dois aspectos anteriores; por exemplo, quando se necessita de grande capacidade de processamento, pode ser necessário utilizar processamento paralelo; mas também existem aspectos ligados a esse tipo de recurso que independem dos dois primeiros; por exemplo, a multidisciplinariedade de um projeto pode tornar quase obrigatório o emprego de trabalho em equipe.

A linguagem de programação a ser utilizada se inter-relaciona fortemente com todas as três classes de recursos disponíveis. Cada linguagem tem aspectos tecnológicos e econômicos que a torna mais ou menos adequada para uma dada aplicação. A interface homem-máquina é um desses aspectos.

Em termos da interface homem-máquina, as linguagens de programação se dividem nos seguintes tipos.

i. Programação de máquina;

ii. Programação estruturada;

iii. Programação Orientada a Objetos (POO);

iv. Programação Funcional;

v. Programação Lógica.

Atualmente, a programação em linguagem de máquina é de uso bastante restrito. Exemplos são aplicativos de pequeno porte, casos onde há o emprego de computadores não programáveis ou para desenvolvimento dos compiladores das linguagens de alto nível. O paradigma da programação funcional é o mais utilizado para problemas envolvendo inteligência artificial, principalmente por propiciar uma modelagem simples desse tipo de problema. A programação lógica se baseia na criação de declarações lógicas ao invés de comandos do tipo *entrada* e *saída*.

Quanto à programação estruturada e a POO, não existe, a princípio, distinção entre o tipo de problema em que se aplicaria a uma ou outra. Até porque, a POO é uma metodologia de programação que estende o conceito da programação estruturada. O que justifica a escolha de

uma ou outra é o ciclo de desenvolvimento e manutenção do programa. A POO possui abstrações adicionais que a torna mais adequada para casos como:

i. Programas de grande porte;

ii. Programas multidisciplinares;

iii. Programas que envolvem equipes de programadores;

iv. Programas com ciclo de vida longo.

1.1 O Projeto INSANE

A pesquisa na área de métodos numérico-computacionais para modelos discretos procura aumentar as possibilidades da metodologia utilizada. Como qualquer outra área do conhecimento, os conceitos já consolidados são o ponto de partida para os novos estudos. Entretanto, no que diz respeito à implementação computacional dos métodos numéricos, observa-se sempre um recomeço do processo ao se recriarem ferramentas relativas a tecnologias dominadas. Isso porque, ao longo do tempo, várias iniciativas de desenvolvimento de aplicativos pela comunidade acadêmica foram descontinuadas. Dentre as causas para esses insucessos, pode-se citar:

i. Dependência de sistemas operacionais pouco amigáveis;

ii. Uso de linguagens de programação não apropriadas, de expansão, distribuição e manutenção difíceis;

iii. Equipes de desenvolvimento fechadas;

iv. Documentação deficiente ou inexistente.

Em suma, é possível concluir que esses insucessos em muito foram ocasionados pela pouca disposição da comunidade acadêmica de se apropriar de tecnologias emergentes ou a inexistência de recursos mais adequados.

Um projeto que permita um aprimoramento progressivo dos modelos de maneira que os novos desenvolvimentos possam se apropriar de soluções já implementadas, requer um ambiente computacional segmentado, amigável a mudanças e escalável em complexidade. Foi

com esse propósito que um grupo de pesquisadores do Departamento de Engenharia de Estruturas da UFMG criou o projeto INSANE (*INteractive Structural ANalysis Enviroment*).

O projeto INSANE se divide em três grandes segmentos: pré-processador, processador e pós-processador. O pré e o pós-processador são aplicações gráficas interativas que disponibilizam recursos diversos para diferentes modelos discretos. O processador é o núcleo numérico do sistema, responsável pela obtenção dos resultados de diferentes modelos discretos de análise.

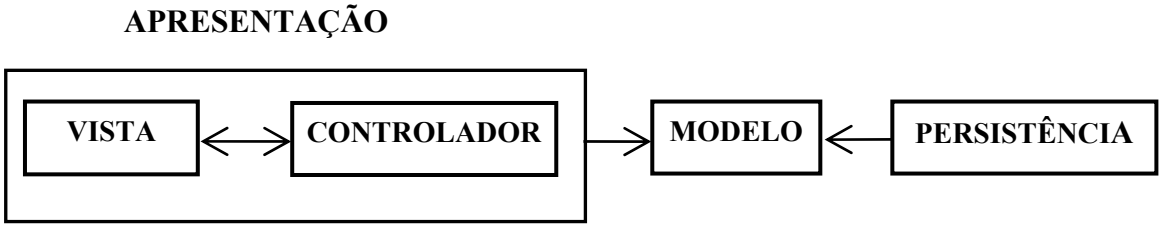


FIGURA 1.1: Arquitetura de camadas do INSANE

A FIGURA 1.1 mostra a combinação de arquitetura em camadas e padrões de projeto de software adotados para o INSANE. Essas camadas lógicas do sistema foram definidas utilizando-se o padrão de projeto de software denominado Modelo-Vista-Controlador (MVC) (Gamma et al., 1995). Este padrão é bastante apropriado para o presente caso, uma vez que preconiza a separação do processamento da informação de sua representação gráfica (Pietro, 2001) facilitando assim os trabalhos de expansão e manutenção da aplicação.

No projeto INSANE, a segmentação também é aplicada nas partes mais especializadas do sistema. Duas são as premissas básicas para o desenvolvimento das diferentes partes do programa. A primeira é que um recurso que possa ser aplicado por partes diferentes do programa deve estar disponível a todas essas partes num único lugar. A segunda é a padronização na construção das diferentes partes e na forma como os recursos de cada uma dessas partes serão disponibilizados.[1]. A implementação de tal paradigma de projeto seria

[1] A padronização de projeto proposta pelo INSANE não se resume a esse caso. Na verdade, todo o projeto vem sendo elaborado utilizando técnicas de desenvolvimento de software, como coordenação do projeto, controle de versões, divisão planejamento-desenvolvimento-simulação etc.

inviável numa linguagem que não fosse orientada a objetos. Restava, então, somente adotar uma linguagem de programação adequada. A opção foi pela linguagem Java.

A busca pela padronização e transmissão de informação também norteou a adoção da XML (*eXtensible Markup Language*) para a persistência dos dados. A XML é um derivado mais simples do SGML (*Standart Generalized Markup Language*). A grande diferença desse tipo de arquivo é que conteúdo e formatação são separados. O conteúdo tem uma forma tipada na qual é possível poucas variações. A forma com o que o documento é apresentado varia conforme sua DTD (*Data Type Definition*). Para processamento digital, essa forma representa uma grande vantagem, pois a informação pode ser trocada mais facilmente por diferentes sistemas.

O projeto INSANE objetiva com essa segmentação e padronização, canalizar esforços independentes de vários centros de pesquisa, de modo a criar uma plataforma de trabalho única. Dessa maneira, pesquisadores com objetivos específicos podem aproveitar o trabalho de desenvolvimento realizado por outros sem a necessidade de implementar todo o aplicativo para seu problema numérico-computacional específico. Esses novos trabalhos, por sua vez, passam a fazer parte da plataforma do INSANE.

1.2 Objetivos

A dissertação de mestrado que aqui se apresenta trata da expansão da camada Modelo (FIGURA 1.1) de maneira a incluir a análise estrutural dinâmica através do método dos elementos finitos, especificamente, para a família dos elementos finitos paramétricos.

Uma vez que a camada modelo utiliza o paradigma da POO, pode se dividir a expansão em três partes principais:

i. Expansão das atribuições do modelo discreto para se determinar a inércia e o amortecimento dos elementos finitos, uma vez que a rigidez já é considerada na análise estática; introdução dos graus de liberdade (GDLs) de velocidade e aceleração nodais; expansão das classes responsáveis pelo carregamento para permitir carregamentos variáveis com o tempo.

ii. Expansão da classe responsável pela resolução do modelo discreto para inclusão dos métodos associados à análise dinâmica. Para a solução modal, são implementados os métodos de iteração inversa e iteração no subespaço. Para a solução por superposição modal, são implementados os métodos de superposição de deslocamentos e superposição de acelerações. Para integração direta, é implementado o método de *Newmark-*β com a variante *Hilber-*α.

iii. Modificação da persistência de dados para as necessidades da análise dinâmica.

Uma vez concluída a fase de implementação, o aplicativo é validado resolvendo-se problemas selecionados.

1.3 Organização do Texto

O texto deste trabalho está dividido em quatro partes: formulação básica e revisão bibliográfica, detalhamento do projeto, validação do projeto e conclusões.

A primeira parte é constituída dos capítulos 2 e 3. No capítulo 2, Análise Dinâmica de Estruturas Através do Método dos Elementos Finitos (MEF), introduz-se os conceitos básicos da dinâmica e evolui-se até as equações de movimento para um sistema discretizado conforme o MEF. A partir daí, apresenta-se a característica modal do movimento dos sistemas dinâmicos lineares, apresentando-se os métodos de solução modal a seguir. O capítulo termina apresentando os métodos de integração direta, a alternativa aos métodos de superposição modal. No capítulo 3, é feita uma apresentação dos recursos computacionais adotados no projeto com ênfase para a POO.

No capítulo 4, Projeto Orientado a Objetos, os conceitos abordados nos dois capítulos anteriores são concretamente utilizados no projeto orientado a objetos para expansão do INSANE de modo a incluir a solução de problemas dinâmicos.

A validação do projeto é apresentada no capítulo 5, Solução de Problemas Selecionados e Análise de Resultados, no qual as soluções obtidas pela expansão do INSANE são confrontadas com as soluções analíticas e soluções disponíveis na literatura.

No capítulo 6 é apresentada a conclusão do trabalho que discute basicamente três pontos: resposta aos objetivos do trabalho; adequação do resultado final alcançado à filosofia do INSANE e da POO; e sugestões para trabalhos futuros.

Foram incluídos três apêndices ao trabalho. O Apêndice 1 apresenta uma visão panorâmica do projeto INSANE mostrando o andamento dos demais projetos do grupo. O Apêndice 2 apresenta a forma dos arquivos de persistência dos modelos e suas soluções. O Apêndice 3 apresenta uma análise da solução utilizando-se distribuição geométrica de massa versus distribuição consistente.

Capítulo 2

ANÁLISE DINÂMICA DE ESTRUTURAS ATRAVÉS DO MEF

2.1 Introdução

O equilíbrio dinâmico de um corpo infinitesimal localizado em $\mathbf{x} = (x_1,x_2,x_3)$ no instante t é dado pela seguinte equação:

$$\rho(\mathbf{x},t)\ddot{\mathbf{z}}(\mathbf{x},t)dV = \mathbf{df_e}\left(\mathbf{x},t\right) \tag{2.1}$$

onde ρ é a densidade do material, $\mathbf{z} = (z_1,z_2,z_3)$ é o vetor de deslocamentos, dV é o elemento infinitesimal de volume e $\mathbf{df_e}$ é o diferencial do conjunto de forças externas atuando sobre o corpo. Essa equação é a forma matemática da 2ª lei de *Newton* em coordenadas cartesianas.

Um meio contínuo pode ser tratado como um arranjo de infinitos corpos infinitesimais. Nesse caso, os esforços dentro do meio são comunicados através do campo de tensão gerado. Escrevendo a Eq

$$\rho(\mathbf{x},t)\ddot{\mathbf{z}}(\mathbf{x},t)dV = \mathbf{df_e}\left(\mathbf{x},t\right) \tag{2.1}$$

segundo a mecânica do contínuo, tem-se:

$$\sum_{j=1}^{3} \sigma_{ji,j} + \rho(\mathbf{x},t) b_i(\mathbf{x},t) = \rho(\mathbf{x},t) \ddot{z}_i(\mathbf{x},t) \qquad i = 1, 2, 3 \tag{2.2}$$

onde $i = 1, 2, 3$ são as direções cartesianas, σ_{ij} são as componentes de tensão sobre o elemento infinitesimal e b_i representa as forças de corpo por unidade de massa.

O primeiro passo para se resolver a Eq. (2.2) é relacionar o campo de tensões com o campo de deformações. Assumindo um movimento adiabático (não existe troca de energia nas fronteiras do corpo) e isotérmico (a energia interna do corpo não varia) pode-se lançar mão da lei de *Hooke* generalizada:

$$\{\sigma_i\} = [c_{ij}]\{\varepsilon_i\} \qquad i, j = 1, ..., 6 \tag{2.3}$$

onde $\{\sigma\} = [\sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \sigma_{12} \quad \sigma_{13} \quad \sigma_{23}]^T$, c_{ij} são constantes e $\{\varepsilon\} = [\varepsilon_1 \quad \varepsilon_2 \quad \varepsilon_3 \quad \varepsilon_{12} \quad \varepsilon_{13} \quad \varepsilon_{23}]^T$ representa o campo de deformações do corpo. Pode se notar que a lei de *Hooke* consiste numa linearização das relações tensão-deformação. A forma mais geral da lei de *Hooke* leva a um sistema no qual as 36 constantes c_{ij} na Eq. (2.3) são independentes. Caso se possa definir uma função para a energia interna de deformação do corpo, é possível provar que existem apenas 21 constantes c_{ij} independentes.

Supondo regime de pequenos deslocamentos e pequenas deformações, a medida de deformação do corpo é aproximadamente independente do referencial. Nesse caso, o tensor de deformações fica:

$$\varepsilon_{ij} = \frac{1}{2}\left(z_{i,j} + z_{j,i}\right) \qquad i, j = 1, 2, 3 \tag{2.4}$$[1]

A forma linear da Eq. (2.4) resulta diretamente da hipótese de pequenas deformações anteriormente adotada.

Substituindo-se as Eqs.(2.3) e (2.4) em (2.2), resulta em uma equação apenas em termos dos deslocamentos e suas derivadas:

---

[1] A forma linear da Eq. (2.4) resulta diretamente da hipótese de pequenas deformações anteriormente adotada.

$$\rho(\mathbf{x},t)b_i(\mathbf{x},t)\left(\sum_{j=1}^{3}\sum_{k=1}^{3}c_{jk}\frac{1}{2}\left(z_{k,ij}+z_{i,kj}\right)\right)=\rho\left(\mathbf{x},t\right)\ddot{z}_i\left(\mathbf{x},t\right) \tag{2.5}$$

i = 1, 2, 3

Para se resolver a Eq. (2.5), é necessário introduzir as condições de contorno essenciais e naturais do problema:

$$z\left(\mathbf{x}_\mathbf{s},t\right)=\mathbf{g}_1\left(\mathbf{x}_\mathbf{s},t\right) \tag{2.6}$$

$$\boldsymbol{\sigma}\left(\mathbf{x}_\mathbf{s},t\right)=\mathbf{g}_2\left(\mathbf{x}_\mathbf{s},t\right) \tag{2.7}$$

e condições iniciais:

$$\mathbf{z}\left(\mathbf{x},0\right)=\mathbf{g}_3\left(\mathbf{x}\right) \tag{2.8}$$

$$\dot{\mathbf{z}}\left(\mathbf{x},0\right)=\mathbf{g}_4\left(\mathbf{x}\right) \tag{2.9}$$

onde de (2.6) a (2.9), $\mathbf{x_s}$ denota as coordenadas da fronteira do corpo e $\mathbf{g_i}$ são funções arbitrárias.

Em muitos casos, não é possível ou não é prático tratar o problema utilizando-se diretamente a forma geral da Eq. (2.5), sendo necessário, portanto, utilizar um tratamento matemático simplificado. Assim, fazem-se hipóteses sobre o comportamento cinemático da estrutura, a geometria do problema e sobre o comportamento do material. Essas hipóteses dão origem a diversos modelos matemáticos de análise de estruturas.

Ainda assim, soluções analíticas dos problemas de dinâmica em engenharia de estruturas só são possíveis para alguns poucos casos, sendo que na maioria das vezes, o nível de simplificação necessário para permitir um tratamento analítico implica em modelos pouco representativos do comportamento real da estrutura, fazendo-se necessária a utilização de algum método numérico de solução.

Existem vários métodos numéricos para se resolver os problemas derivados da Eq. (2.5). Ao longo do tempo, os métodos numéricos que utilizam alguma forma de discretização do domínio ganharam espaço por serem de fácil generalização. Esses métodos consistem em se

escrever a equação diferencial do problema em função das incógnitas de alguns pontos do domínio, transformando o sistema de equações diferenciais do problema (2.5) em um sistema de equações algébricas. Exemplos são o método de diferenças finitas, de elementos finitos, de volumes finitos, de elementos de contorno, entre outros. No caso particular do MEF, o problema (2.5) é escrito como:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{F_d} + \mathbf{K}\mathbf{u} = \mathbf{F_{eq}} \tag{2.10}$$

Onde $\mathbf{M} = [m_{ij}]$ é a matriz de massa, $\mathbf{u} = \{u_i\}$ é o vetor de deslocamentos nodais, $\mathbf{F_d} = \{f_d^i\}$ é o vetor de amortecimento oriundo de efeitos internos à estrutura, $\mathbf{K} = [k_{ij}]$ é a matriz de rigidez, $F_{eq} = \{f_{eq}^i\}$ é o vetor de carregamentos externos equivalente e i, j =1, …, nGDL (número de graus de liberdade) do modelo discreto.

O MEF é um caso particular dos métodos que assumem, *a priori*, uma forma para a solução do problema, ou seja:

$$\mathbf{z}(\mathbf{x}, t) \approx \hat{\mathbf{z}}(\mathbf{x}, t) \tag{2.11}$$

onde $\hat{\mathbf{z}}$ representa uma família de funções que se supõem serem uma boa aproximação para a função solução do problema. Levando-se a Eq. (2.11) na Eq (2.5), tenta se obter qual a função particular da família de funções $\hat{\mathbf{z}}$ melhor aproxima **z**, sendo que essa aproximação é definida através de algum critério adequado.

De maneira resumida, uma análise via MEF consiste em:

i. Dividir o domínio em subdomínios, sendo que cada subdomínio constitui um elemento da malha de elementos finitos.

ii. Assumir uma função $\hat{\mathbf{z}}$ para cada elemento finito.

iii. Escrever a Eq. (2.10) para cada elemento finito.

iv. Montar a Eq. (2.10) para todo o modelo discreto a partir das equações locais.

v. Resolver a equação resultante e analisar os resultados.

Essa forma de composição do problema global a partir da montagem de subproblemas dentro de cada elemento finito é uma das principais vantagens do MEF, pois permite uma

generalização mais fácil do processo de solução para diferentes problemas. Numa análise via MEF, uma mudança em parte do modelo não implica no recomeço de todo o processo de modelagem.

Independente do critério adotado para se determinar $\hat{\mathbf{z}}$, as matrizes de massa e rigidez locais para a família dos elementos finitos paramétricos são dadas, respectivamente, por:

$$\mathbf{M^e} = \left[m_{ij}\right]^e = \int_V \rho \mathbf{N^T N} dV \tag{2.12}$$

$$\mathbf{K^e} = \left[k_{ij}\right]^e = \int_V \mathbf{B^T D B} dV \tag{2.13}$$

onde **N** é a matriz que contém as funções de forma (funções assumidas) dos elementos finitos, **B** é matriz que contém as derivadas das funções de forma, **D** é a matriz constitutiva do elemento, função do modelo de análise adotado (estado plano, sólido, axissimétrico, etc) e das propriedades do material, e '*V*' é o domínio de integração do elemento.

O carregamento nodal equivalente é composto por duas partes, os carregamentos aplicados diretamente nos nós e os carregamentos aplicados fora dos nós. Para a segunda parte do carregamento, que pode ser constituído tanto por carregamentos concentrados quanto por carregamentos distribuídos, deve-se utilizar algum critério adequado para transferi-los para os nós do elemento. Isso é feito pela seguinte equação:

$$\mathbf{F^e_{eq}} = \int_V \mathbf{N^T b} dV \tag{2.14}$$

onde **b** expressa a variação do carregamento no domínio do elemento.

Para se chegar à equação de movimento discretizada falta ainda definir a forma para cálculo de $\mathbf{F_d}$. O fenômeno do amortecimento em estruturas não é tão bem compreendido ou tão bem modelado quanto outras propriedades mecânicas, como a rigidez ou a inércia (Weaver e Johnston, 1987; Craig, 1981), de modo que se faz necessária uma análise mais detalhada desse fenômeno.

2.2 Forças Dissipativas Internas - O Amortecimento Constitutivo e o Amortecimento Viscoso

O movimento relativo interno dos materiais faz com que esses dissipem parte da energia mecânica do movimento. Em muitos problemas, é difícil se isolar esse amortecimento constitutivo de outros fenômenos dissipativos que ocorrem dentro de uma estrutura, como, por exemplo, pequenas deformações plásticas localizadas ou fricção nas uniões. Em outros casos, mesmo fenômenos não dissipativos se confundem, devido às limitações do modelo adotado, com fenômenos de dissipação. Um exemplo disso é a transmissão de energia em suportes modelados como perfeitamente rígidos. Adicione-se a isso o fato de não se compreender bem o mecanismo interno de dissipação de energia dos materiais.

As dificuldades de se modelar o amortecimento nos sistemas mecânicos são grandemente compensadas pelo fato de, em geral, esses sistemas possuírem baixos níveis de amortecimento, fazendo com ele possa ser negligenciado em muitos problemas práticos. Mesmo quando este não pode ser negligenciado, a determinação precisa da forma do amortecimento nem sempre é necessária. Tendo isso em vista, é comum considerar o amortecimento na estrutura como sendo a soma de todos os fenômenos, dissipativos ou não, que removem energia mecânica do modelo considerado, sem identificar qual é a origem desse amortecimento.

Uma forma matemática que atende bem a essa soma de fenômenos tratada como amortecimento estrutural é o amortecimento viscoso linear. O amortecimento viscoso linear, f_v, de uma partícula em um movimento unidimensional é dado por:

$$f_v = a\dot{z}(t) \tag{2.15}$$

onde 'a' é uma constante.

O amortecimento viscoso é observado no movimento interno dos fluidos ou na interface sólido-fluido quando existe movimento relativo entre esses. Para baixas velocidades, o amortecimento viscoso se aproxima do comportamento linear. A constante 'a' em (2.15) depende das propriedades do fluido e da rugosidade da superfície do sólido.

Em engenharia de estruturas o que se faz é adotar um valor médio '$\overline{a}$' de modo a igualar o efeito global de dissipação de energia, ou seja:

$$\oint_r \sum f_{dissipativa} dr = \oint_r \bar{a}\dot{u}(t) dr \tag{2.16}$$

onde $\Sigma f_{dissipativa}$ representa as forças dissipativas da estrutura e '*r*' é o circuito percorrido pela partícula. Como dito anteriormente, em geral, $\Sigma f_{dissipativa}$ engloba tanto os fenômenos dissipativos quanto aqueles que, por questões de modelagem, são tratados como dissipadores de energia para o modelo em questão.

O amortecimento viscoso linear tem duas características que o tornam especialmente atrativo: em primeiro lugar, o seu tratamento matemático é relativamente simples; em segundo lugar, herda-se bastante informação dos estudos feitos para o caso de movimento de sólidos em meios fluidos.

Considerando-se o caso de amortecimento viscoso linear em sistemas mecânicos de 1GDL, pode-se reescrever a Eq. (2.1) da seguinte maneira:

$$\ddot{z} + 2\zeta\omega_n \dot{z} + \omega_n^2 z = \frac{f_e}{m} \tag{2.17}$$

onde:

$$\zeta = \frac{a}{a_{cr}} \tag{2.18}$$

$$a_{cr} = 2m\omega_n \tag{2.19}$$

$$\omega_n = \sqrt{\frac{k}{m}} \tag{2.20}$$

sendo '*k*', '*m*' e a_{cr} a rigidez, a massa o amortecimento crítico do sistema, respectivamente. O amortecimento crítico é uma propriedade do sistema que permite definir se o movimento livre do mesmo é oscilatório ou não. Para amortecimentos abaixo do crítico (sistemas subamortecidos), o sistema oscila em torno do ponto de equilíbrio após uma perturbação inicial enquanto que, para sistemas com amortecimento acima do crítico (sistemas superamortecidos), o sistema converge assintoticamente para o ponto de equilíbrio após a mesma perturbação inicial.

A razão de amortecimento, 'ζ', é definida como a razão entre o amortecimento de um sistema e seu amortecimento crítico. No caso de sistemas subamortecidos, a freqüência de vibração livre pode ser expressa em função da razão de amortecimento do sistema por:

$$\omega_d = \omega_n \sqrt{1-\zeta^2} \tag{2.21}$$

No caso de sistemas mecânicos discretos com nGDL, a Eq. (2.10) pode ser reescrita supondo amortecimento viscoso da seguinte maneira:

$$\mathbf{M\ddot{u} + A\dot{u} + Ku = F_{eq}} \tag{2.22}$$

onde **A** é a matriz de amortecimento. O amortecimento de um único elemento finito paramétrico é dado por:

$$\mathbf{A^e} = \left[a_{ij}\right]^e = \int_V a\mathbf{N}^\mathbf{T}\mathbf{N}dV \tag{2.23}$$

onde 'a' é uma propriedade do material[1].

2.3 Análise Modal de Sistemas Lineares

Quando $\mathbf{F_e} = \mathbf{0}$ na Eq. (2.10), o sistema mecânico está em vibração livre. Considerando-se, um caso particular no qual as forças dissipativas também são nulas, tem-se:

$$\mathbf{M\ddot{u} + Ku = 0} \tag{2.24}$$

Assumindo-se uma solução da forma:

$$\mathbf{u} = \cos\omega t\mathbf{Y} = \cos\omega t\left[\upsilon_1 \quad \upsilon_2 \quad \ldots \quad \upsilon_n\right]^\mathbf{T} \tag{2.25}$$

[1] A Eq. (2.23) supõe que seja possível associar alguma propriedade de amortecimento viscoso intrínseca ao material.

onde ω é um escalar e $\mathbf{Y}$ é um vetor ainda a serem determinados. Nota-se que a solução proposta é harmônica de freqüência ω.

Levando-se (2.25) em (2.24), chega-se ao problema de autovalor generalizado do sistema:

$$\left(-\omega^2\mathbf{M}+\mathbf{K}\right)\mathbf{Y}=\mathbf{0}$$

donde:

$$\left(\mathbf{M}-\lambda\mathbf{K}\right)\mathbf{Y}=\mathbf{0} \tag{2.26}$$

sendo:

$$\lambda=\frac{1}{\omega^2} \tag{2.27}$$

Para que (2.26) possua solução não trivial, deve-se ter:

$$\left|\mathbf{M}-\lambda\mathbf{K}\right|=0 \tag{2.28}$$

A Eq. (2.28) é denominada equação característica do problema. Seja agora $\mathbf{Y_i}$ o iésimo vetor que satisfaça (2.26) para um dado ω_i^2; esse vetor, conhecido como autovetor ou modo natural de vibração, não é único; isto é, $\mathbf{Y_i}$ admite uma única direção, mas a sua magnitude pode variar livremente. O mais comum, entretanto, é se trabalhar com autovetores normalizados. Uma normalização bastante utilizada é feita com relação à matriz de massa, como segue:

$$\boldsymbol{\varphi}_\mathbf{i}=c_i\mathbf{Y}_\mathbf{i} \tag{2.29}$$

onde:

$$c_i=\frac{1}{\sqrt{\mathbf{Y}_\mathbf{i}^\mathbf{T}\mathbf{M}\mathbf{Y}_\mathbf{i}}} \tag{2.30}$$

e $\boldsymbol{\varphi}_\mathbf{i}$ é autovetor normalizado.

Fisicamente, os autovetores $\mathbf{Y_i}$ ou $\boldsymbol{\varphi}_\mathbf{i}$ são modos de vibração livre do sistema mecânico. Para a normalização adotada, tem-se que:

$$\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{M}\boldsymbol{\varphi}_\mathbf{i} = 1 \tag{2.31}$$

A matriz modal normalizada é aquela na qual os autovetores normalizados são arranjados em colunas na ordem decrescente dos respectivos autovalores:

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_\mathbf{1} & \boldsymbol{\varphi}_\mathbf{2} & \cdots & \boldsymbol{\varphi}_\mathbf{n} \end{bmatrix} \qquad \lambda_1 \geq \lambda_2 \geq \ldots \lambda_n \tag{2.32}$$

Uma das propriedades fundamentais dos autovetores é a sua ortogonalidade em relação às matrizes de massa e de rigidez:

$$\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{M}\boldsymbol{\varphi}_\mathbf{j} = \boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{K}\boldsymbol{\varphi}_\mathbf{j} = 0 \qquad i \neq j \tag{2.33}$$

A Eq. (2.28) é um polinômio de grau '*n*' denominado polinômio característico e os autovalores (2.27) são as raízes desse polinômio. A Eq. (2.28) pode apresentar raízes múltiplas, de modo que um sistema com '*n*' GDL possui '*m*' autovalores diferentes, sendo $m \leq n$. Isso implica em dizer que um sistema mecânico discreto tem '*m*' freqüências naturais. Porém, sempre é possível obter um conjunto de '*n*' autovetores que satisfaçam as equações (2.31) e (2.33).

Combinando-se a Eq. (2.33) com a Eq. (2.31), obtém-se o seguinte resultado:

$$\boldsymbol{\Phi}^\mathbf{T}\mathbf{M}\boldsymbol{\Phi} = \mathbf{I} \tag{2.34}$$

onde **I** é a matriz identidade.

O resultado obtido na Eq. (2.34) deriva diretamente da escolha feita para a normalização dos autovetores. No caso geral, o resultado da operação feita em (2.34) sobre a matriz **M** será uma matriz diagonal denominada matriz de massa modal ou principal ($\mathbf{M_P}$).

Analogamente, a matriz de rigidez principal, $\mathbf{K_p}$, é definida por:

$$\mathbf{K_p} = \boldsymbol{\Phi}^\mathbf{T}\mathbf{K}\boldsymbol{\Phi} \tag{2.35}$$

sendo que $\mathbf{K_p}$ também é uma matriz diagonal.

Retomando-se a Eq. (2.35) com $\mathbf{Y} = \boldsymbol{\varphi}_\mathbf{i}$

$$\left(\mathbf{M} - \lambda\mathbf{K}\right)\boldsymbol{\varphi}_\mathbf{i} = 0$$

a pré-multiplicação por $\boldsymbol{\varphi}_\mathbf{i}$ resulta em:

$$\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{K}\boldsymbol{\varphi}_\mathbf{i} = \omega_i^2 \tag{2.36}$$

Levando-se (2.36) em (2.35) conclui-se que os elementos $k_{p\,ii}$ da diagonal principal de $\mathbf{K_p}$ para a normalização em relação à matriz de massa adotada, Eq. (2.31), são dados por:

$$k_{p\,ii} = \omega_i^2 \tag{2.37}$$

onde ω_i são as freqüências naturais do sistema.

Seja a variável auxiliar η_i definida por:

$$\eta_i = \varepsilon_i \cos \omega_i t \tag{2.38}$$

onde ε_i é um escalar arbitrário. Substituindo-se (2.25), (2.29), (2.36) e (2.38) em (2.24), tem-se:

$$\mathbf{M}\boldsymbol{\varphi}_\mathbf{i}\ddot{\eta}_i + \mathbf{K}\boldsymbol{\varphi}_\mathbf{i}\eta_i = \mathbf{0} \tag{2.39}$$

Pré-multiplicando (2.39) por $\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}$, obtém-se:

$$\ddot{\eta}_i + \omega_i^2\eta_i = 0 \tag{2.40}$$

A Eq. (2.40) mostra que, na direção $\boldsymbol{\varphi}_\mathbf{i}$, é possível escrever a equação do movimento discretizada segundo o MEF como uma única equação diferencial de 2ª ordem na variável η_i. Essa equação possui um significado físico bastante interessante. Sabe-se que um sistema mecânico livre de forças externas que realizem trabalho sobre si oscila em torno do seu ponto de equilíbrio de maneira que a energia mecânica do sistema permute entre formas cinética e potencial. Conclui-se então que essa permutação só é possível em modos específicos $\boldsymbol{\varphi}_\mathbf{i}$ e com freqüências específicas $\omega_\mathbf{i}$ associadas a esses modos.

Fica implícito, também, que todo o trabalho de uma força externa realizado segundo um dos modos naturais da estrutura será totalmente convertido em energia mecânica interna do

sistema, pois o sistema não “rejeita” energia mecânica quando se movimenta conforme seus modos naturais; é o fenômeno da ressonância. Uma força que realize trabalho fora dos modos ou freqüências naturais da estrutura resultará no agente desse trabalho tendo que absorver a energia rejeitada pelo sistema ao oscilar fora do seu modo natural.

Considerando-se o caso particular do sistema representado pela Eq. (2.10), onde as forças dissipativas são nulas, mas os carregamento externos são diferentes de zero, tem-se

$$\mathbf{M\ddot{u} + Ku = F_e} \tag{2.41}$$

Considerando-se agora que o vetor solução **u** possa ser escrito por:

$$\mathbf{u} = \boldsymbol{\varphi}_\mathbf{i}\eta_i \tag{2.42}$$

substituindo-se (2.42) em (2.41) e pré-multiplicando-se por $\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}$, tem-se:

$$\ddot{\eta} + \omega_i\eta = f_{ep}^i \tag{2.43}$$

onde f_{ep}, denominada força principal, é dada por:

$$f_{ep}^i = \boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{F_e} \tag{2.44}$$

A Eq. (2.43) é uma equação diferencial linear de 2ª ordem não homogênea na variável η_i. A solução dessa equação pode ser obtida pela integral de *Duhamel*, como segue:

$$\eta_i = \frac{1}{\omega_i}\int_0^t f_{ep}^i \operatorname{sen}\omega_i(t-\tau)d\tau \tag{2.45}$$

onde τ é uma variável de integração muda.

Por fim, considere-se agora o caso mais geral para um sistema discretizado segundo o MEF, onde, além de carregamentos externos, também haja forças dissipativas internas à estrutura. Admitindo-se que o amortecimento interno seja do tipo viscoso linear, retorna-se, então, à Eq. (2.22), aqui repetida:

$$\mathbf{M\ddot{u} + A\dot{u} + K\ddot{u} = F_e} \tag{2.22}$$

No caso de análise modal de sistemas lineares, mesmo a forma relativamente simples de amortecimento adotado na Eq. (2.22) pode implicar em uma dificuldade adicional: não existem técnicas gerais de separação modal para um amortecimento linear qualquer, sendo que, em geral, os autovalores da Eq. (2.22) são complexos. Para se evitar tal dificuldade, é comum se adotar, a priori, uma matriz de amortecimento **A** que essa seja diagonalizável pela matriz modal $\mathbf{\Phi}$. Esse tipo de amortecimento é denominado *amortecimento modal*, pois atua segundo os modos naturais do sistema não-amortecido. *Rayleigh* propôs uma matriz de amortecimento modal proporcional à massa e à rigidez do sistema:

$$\mathbf{A} = a_{r1}\mathbf{M} + a_{r2}\mathbf{K} \tag{2.46}$$

nde 'a_{r1}' e 'a_{r2}' são constantes denominadas coeficientes de amortecimento de *Rayleigh*. Para a escolha feita de normalização dos autovetores, tem-se:

$$\mathbf{A_p} = a_{r1}\mathbf{I} + a_{r2}\mathbf{K_p} \tag{2.47}$$

onde $\mathbf{A_p}$ é a matriz de amortecimento principal.

Levando-se agora (2.42) em (2.22),

$$\mathbf{M\varphi_i}\ddot{\eta}_i + \mathbf{A\varphi_i}\ddot{\eta}_i + \mathbf{K\varphi_i}\ddot{\eta}_i = \mathbf{F_e}$$

pré-multiplicando-se por $\boldsymbol{\varphi}_i^{\mathrm{T}}$ e considerando o amortecimento de *Rayleigh*, tem-se:

$$\ddot{\eta}_i + a_i\dot{\eta}_i + \omega_i^2\eta_i = f_{ep}^{\,i} \tag{2.48}$$

onde:

$$a_i = a_{r1} + a_{r2}\omega_i^2 \tag{2.49}$$

Analogamente ao caso não amortecido, a solução da Eq. (2.48) pode ser obtida pela integral de *Duhamel* adaptada para o caso de amortecimento viscoso linear:

$$\eta_i = \frac{\mathrm{e}^{-\zeta_i\omega_i t}}{\omega_d^i}\int_0^t \mathrm{e}^{\zeta_i\tau}\ f_{ep}^{\,i}\ \mathrm{sen}\ \omega_d^i\,(t-\tau)d\tau \tag{2.50}$$

2.3.1 O Método de Superposição de Deslocamentos Modais

Uma vez que o sistema representado por (2.48) é linear, a sua solução pode escrita como a combinação linear de todas as soluções modais do sistema:

$$\mathbf{u} = \sum_{i=1}^{n} \boldsymbol{\varphi}_\mathbf{i} \eta_i \tag{2.51}$$

ou em forma matricial:

$$\mathbf{u} = \boldsymbol{\Phi}\boldsymbol{\eta} \tag{2.52}$$

substituindo-se (2.52) em (2.22) e pré-multiplicando-se por $\boldsymbol{\Phi}^{\mathbf{T}}$, tem-se:

$$\ddot{\boldsymbol{\eta}} + \mathbf{A}_\mathbf{p}\dot{\boldsymbol{\eta}} + \boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} = \mathbf{F}_\mathbf{ep} \tag{2.53}$$

onde:

$$\begin{cases} \Lambda_{ii} = \lambda_i \\ \Lambda_{ij} = 0; \quad i \neq j \end{cases} \tag{2.54}$$

De modo a se obter uma analogia plena com o sistema de 1GDL dado pela Eq. (2.17), assuma-se que seja possível escrever a $\mathbf{A_p}$ da seguinte maneira[1]:

$$\mathbf{A}_\mathbf{p} = 2\mathbf{Z}\boldsymbol{\Lambda}^{-\frac{1}{2}} \tag{2.55}$$

sendo que:

$$\begin{cases} Z_{ii} = \zeta_i \\ Z_{ij} = 0; \quad i \neq j \end{cases} \tag{2.56}$$

então, a Eq. (2.53) pode ser rescrita como:

---

[1] É possível escrever qualquer matriz diagonal com a forma (2.55). Assim, seria possível chegar na Eq. (2.53) mesmo que o amortecimento modal proposto não fosse o amortecimento de *Rayleigh*.

$$\ddot{\boldsymbol{\eta}} + 2\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{Z}\dot{\boldsymbol{\eta}} + \boldsymbol{\Lambda}^{-1}\boldsymbol{\eta} = \mathbf{F_{ep}} \tag{2.57}$$

Como as matrizes $\mathbf{Z}$ e $\boldsymbol{\Lambda}$ são diagonais, a Eq. (2.57) representa '*n*' problemas de 1 GDL em η_i; ou seja, a técnica de superposição de deslocamentos modais permite a solução no tempo de um sistema *n*-dimensional a partir da solução de '*n*' sistemas de 1 GDL[1].

Uma propriedade física que permite aumentar significativamente o desempenho do método da superposição de deslocamentos modais é que, para um movimento arbitrário, os primeiros modos naturais (modos associados às freqüências mais baixas) são predominantes, armazenando a maior parte da energia do sistema. Assim, ao se fazer a expansão dada por (2.52), não é necessário se levar em consideração todos os modos naturais da estrutura, ou seja:

$$\mathbf{u} \approx \hat{\boldsymbol{\Phi}}\hat{\boldsymbol{\eta}} \tag{2.58}$$

onde:

$$\hat{\boldsymbol{\Phi}} = \begin{bmatrix} \boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_2 & \cdots & \boldsymbol{\varphi}_N \end{bmatrix} \tag{2.59}$$

$$\hat{\boldsymbol{\eta}} = \begin{bmatrix} \eta_1 & \eta_2 & \cdots & \eta_N \end{bmatrix}^{\mathbf{T}} \tag{2.60}$$

sendo $N \leq n$.

Não necessariamente, '*N*' deve representar os '*N*' primeiros primeiro modos do sistema. Pode-se incluir em (2.59) e (2.60), de modo conveniente, os modos que melhor descrevem um movimento particular do sistema.

Fazendo-se a mesma análise que levou à Eq. (2.57), mas agora utilizando a solução modal truncada (2.58), obtém-se:

---

[1] Essa propriedade também é válida para sistemas contínuos sendo que esse tipo de sistema apresenta infinitos modos e respectivas freqüências naturais.

$$\ddot{\hat{\boldsymbol{\eta}}} + 2\hat{\boldsymbol{\Lambda}}\hat{\mathbf{Z}}^{-\frac{1}{2}}\dot{\hat{\boldsymbol{\eta}}} + \hat{\boldsymbol{\Lambda}}^{-1}\hat{\boldsymbol{\eta}} = \hat{\mathbf{F}}_{\mathbf{ep}} \tag{2.61}$$

onde:

$$2\hat{\boldsymbol{\Lambda}}^{-\frac{1}{2}}\hat{\mathbf{Z}} = \hat{\boldsymbol{\Phi}}^{\mathbf{T}}\mathbf{A}\hat{\boldsymbol{\Phi}} \tag{2.62}$$

$$\hat{\boldsymbol{\Lambda}}^{-1} = \hat{\boldsymbol{\Phi}}^{\mathbf{T}}\mathbf{K}\hat{\boldsymbol{\Phi}} \tag{2.63}$$

$$\hat{\mathbf{F}}_{\mathbf{ep}} = \hat{\boldsymbol{\Phi}}^{\mathbf{T}}\mathbf{F}_{\mathbf{e}} \tag{2.64}$$

A Eq.(2.61) significa que pode se resolver um problema *n*-dimensional de forma aproximada através da solução de '*N*' sistemas massa mola. Essa técnica é denominada truncamento modal. O número de modos a ser considerado depende da precisão requerida e das particularidades de cada problema. Quando $N = n$, chega-se à solução exata do problema discreto. Craig (1981) observa, porém, que no processo de discretização, o erro nos modos superiores é relativamente maior do que o erro nos primeiros modos, enquanto Hughes (2000) salienta que os últimos modos de um sistema discretizado não têm correlação com o sistema real, tratando-se de um efeito espúrio do processo de modelagem. Portanto, caso o sistema discreto seja resultado de uma discretização de um sistema contínuo, os modos superiores devem ser necessariamente desconsiderados.

2.3.2 O Método de Superposição de Acelerações Modais

O método da aceleração modal é uma derivação do método de superposição modal que, além de requerer menor número de modos naturais para se obter uma boa convergência, é exato para carregamentos estáticos independentemente do número de modos considerados.

A partir da Eq.(2.22), pode se escrever:

$$\mathbf{u} = \mathbf{u}_{\mathbf{pe}} - \mathbf{K}^{-1}\left(\mathbf{M}\ddot{\mathbf{u}} + \mathbf{A}\dot{\mathbf{u}}\right) \tag{2.65}$$

onde $\mathbf{u}_{\mathbf{pe}}$ é denominado a resposta pseudo-estática do sistema dada por:

$$\mathbf{u}_{\mathbf{pe}} = \mathbf{K}^{-1}\mathbf{F}_{\mathbf{e}} \tag{2.66}$$

A partir de (2.26), é possível escrever:

$$\lambda \mathbf{Y} = \mathbf{K}^{-1}\mathbf{M}\mathbf{Y}$$

ou de modo a considerar todos os autovalores:

$$\mathbf{\Phi\Lambda\eta} = \mathbf{K}^{-1}\mathbf{M\Phi\eta} \tag{2.67}$$

Substituindo-se (2.62) e (2.67) no lado direito da Eq. (2.65), tem-se:

$$\mathbf{u} = \mathbf{u_{pe}} - \mathbf{\Phi\Lambda\ddot{\eta}} + \mathbf{K^{-1}A\Phi\dot{\eta}}$$

De (2.67), pode se notar que $\mathbf{K}^{-1} = \mathbf{\Phi\Lambda\Phi}^{-1}\mathbf{M}^{-1}$. Utilizando a propriedade $\mathbf{M}^{-1} = \mathbf{\Phi\Phi}^{\mathbf{T}}$ e adotando-se amortecimento modal com $\mathbf{A_p} = \mathbf{\Phi}^{\mathbf{T}}\mathbf{A\Phi}$, chega-se a:

$$\mathbf{u} = \mathbf{u_{pe}} - \mathbf{\Phi\Lambda\ddot{\eta}} + \mathbf{\Phi\Lambda A_p\dot{\eta}} \tag{2.68}$$

Substituindo agora (2.55):

$$\mathbf{u} = \mathbf{u_{pe}} - \mathbf{\Phi\Lambda\ddot{\eta}} + 2\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{Z\Phi\dot{\eta}} \tag{2.69}$$

obtém-se a expressão geral para o método de superposição de acelerações modais. A velocidade modal, $\dot{\boldsymbol{\eta}}$, pode ser obtida derivando-se a expressão (2.50) uma vez em relação ao tempo:

$$\dot{\eta}_i = \frac{\mathrm{e}^{-\zeta_i\omega_i t}}{\omega_{di}}\int_0^t \mathrm{e}^{\zeta_i\omega_i\tau}\ f_{epi}\left[\cos\omega_{di}(t-\tau) + \omega_i \operatorname{sen}\omega_{di}(t-\tau)\right]d\tau \tag{2.70}$$

enquanto $\ddot{\boldsymbol{\eta}}$ pode ser isolado na Eq. (2.57), chegando-se a:

$$\ddot{\boldsymbol{\eta}} = \mathbf{F_{ep}} - 2\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Z\dot{\eta}} - \mathbf{\Lambda}^{-1}\boldsymbol{\eta} \tag{2.71}$$

A aceleração modal poderia ser obtida derivando-se a Eq. (2.70) uma vez mais com relação ao tempo ao invés de se usar Eq. (2.57). Porém, ao proceder-se dessa maneira, só seria possível

utilizar a Eq. (2.69) garantindo-se que a velocidade fosse pelo menos ‘C1’. Embora nos sistemas físicos reais isso seja sempre verdade, é comum se utilizar modelos matemáticos onde isso não ocorre como, por exemplo, modelos com forças impulsivas, as quais causam uma descontinuidade na curva de aceleração e um bico na curva de velocidade do sistema. Conclui-se, então, que, para problemas com forças impulsivas, no qual a velocidade experimenta mudanças do tipo degrau, a Eq. (2.69) só será válida se o efeito do amortecimento for desconsiderado.

A resposta pseudo-estática, ($\mathbf{u_{pe}}$) é a responsável pelo método de superposição de acelerações modais dar resultados exatos para carregamentos estáticos. Esse fato é bastante importante para carregamentos dinâmicos que possuem uma parcela constante. A convergência mais rápida desse método em relação à superposição de deslocamentos modais fica evidente ao se escrever a Eq. (2.69) numa forma alternativa:

$$\mathbf{u} = \mathbf{u_{pe}} - \sum_i \frac{1}{\omega_i^2} \boldsymbol{\varphi}_\mathbf{i} \ddot{\eta}_i \tag{2.72}$$

onde foi considerado um sistema não amortecido.

Em (2.72), pode se observar que peso de cada modo é reduzido na razão do quadrado da sua respectiva freqüência natural (Craig, 1981). Isso corrobora a observação feita por Craddock (2000) que salienta que o termo da resposta pseudo-estática leva em consideração todos os modos do sistema mecânico, por isso a contribuição da segunda parcela da direita em (2.72) é menor.

2.3.3 Solução do Problema de Autovalor

O primeiro passo para a solução de um problema pelos métodos de decomposição modal é a obtenção dos modos do sistema. Existem vários métodos para resolução do problema de autovalor dado pela Eq. (2.26). Craig (1981) divide esses métodos em três grandes grupos:

i. Métodos de iteração vetorial: consiste em sistematicamente assumir vetores que satisfaçam a o problema de autovalor (2.26).

ii. Métodos de transformação matricial: consistes de métodos que utilizam as propriedades de ortogonalidade dos autovetores, que são apresentadas de forma reduzida nas equações (2.34) e (2.63).

iii. Métodos de determinação das raízes do polinômio característico: baseiam-se na procura das raízes do polinômio característico associado à Eq. (2.28).

A seguir são apresentados dois métodos para a solução de problemas de autovalor, o Método da Iteração Inversa, que é um método de iteração vetorial, e o método de iteração no subespaço, que conjuga características dos métodos de iteração vetorial com características dos métodos de transformação matricial.

2.3.4 O Método da Iteração Inversa

Reescrevendo-se a Eq. (2.26) de maneira ligeiramente modificada, tem-se:

$$\lambda \mathbf{K}\boldsymbol{\varphi} = \mathbf{M}\boldsymbol{\varphi} \tag{2.73}$$

Pode-se notar que o autovetor $\boldsymbol{\varphi}$ aparece em ambos os lados da equação. Isso sugere que ao se adotar um valor tentativa $\overline{\boldsymbol{\varphi}}$ e levá-lo na Eq.

$$\lambda \mathbf{K}\boldsymbol{\varphi} = \mathbf{M}\boldsymbol{\varphi} \tag{2.73}$$

, obtém-se uma melhor aproximação $\overline{\overline{\boldsymbol{\varphi}}}$ para o autovetor procurado a menos de um fator $\overline{\lambda}$, ou seja:

$$\overline{\overline{\boldsymbol{\varphi}}}_{\mathbf{i+1}} = \mathbf{D}\overline{\boldsymbol{\varphi}}_{\mathbf{i}} \tag{2.74}$$

e

$$\overline{\boldsymbol{\lambda}}_{\mathbf{i+1}}\overline{\boldsymbol{\varphi}}_{\mathbf{i+1}} = \overline{\overline{\boldsymbol{\varphi}}}_{\mathbf{i+1}} \tag{2.75}$$

onde **D**, conhecida como matriz dinâmica, é dada por:

$$\mathbf{D} = \mathbf{K}^{-1}\mathbf{M} \tag{2.76}$$

A maneira mais eficiente de se calcular $\overline{\lambda}_{i+1}$, mas não a única, é através do quociente de *Rayleigh* dado por:

$$\overline{\lambda}_{i+1} = \frac{\overline{\overline{\boldsymbol{\varphi}}}_{\mathbf{i+1}}^{\mathbf{T}} \overline{\overline{\mathbf{M}}} \overline{\boldsymbol{\varphi}}_{\mathbf{i+1}}}{\overline{\overline{\boldsymbol{\varphi}}}_{\mathbf{i+1}}^{\mathbf{T}} \overline{\overline{\mathbf{K}}} \overline{\boldsymbol{\varphi}}_{\mathbf{i+1}}} \tag{2.77}$$

As Eqs.(2.74), (2.75) (2.77)e constituem um passo do método da iteração inversa. Pode-se demonstrar que o método da iteração inversa converge sempre para o maior autovalor do sistema, ou seja, $\overline{\boldsymbol{\varphi}}$ converge para o modo fundamental.

O método da iteração inversa pode ser adaptado para se obter modos superiores utilizando-se a técnica do deslocamento espectral. A idéia por trás dessa técnica pode ser visualizada na Figura 2.1. Através de uma translação ao longo do eixo $\hat{\lambda}$, onde $\hat{\lambda}$ é uma variável auxiliar dada por:

$$\hat{\lambda} = \frac{1}{\lambda} \tag{2.78}$$

é possível trazer uma freqüência diferente da fundamental para mais próximo da origem, tornando-a a de menor valor absoluto dentro do espectro de freqüências nesta nova referência:

$$\left(\overline{\mathbf{K}} - \hat{\lambda}\mathbf{M}\right)\boldsymbol{\varphi}_{\mathbf{i}} = \mathbf{0} \tag{2.79}$$

onde:

$$\overline{\mathbf{K}} = \mathbf{K} - \mu\mathbf{M} \tag{2.80}$$

$$\overline{\hat{\lambda}} = \frac{1}{\lambda} - \mu \tag{2.81}$$

e μ é um escalar escolhido de modo que o autovalor mais próximo da nova origem seja o próximo autovalor a ser encontrado. Na prática, o que se faz é adotar μ cada vez maiores até que se consiga convergir para o autovalor desejado. O método de translação espectral também pode ser utilizado para o caso de sistemas com modos de corpo rígido para se eliminar a singularidade da matriz de rigidez.

Outro método para se obter convergência para os autovalores superiores é o de deflação de matriz. Esse método consiste em ortogonalizar a matriz dinâmica com relação aos autovetores já obtidos como segue:

$$\mathbf{D_{i+1}} = \mathbf{D_i} - \frac{\lambda_i}{\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{M}\boldsymbol{\varphi}_\mathbf{i}}\boldsymbol{\varphi}_\mathbf{i}\boldsymbol{\varphi}_\mathbf{i}^\mathbf{T}\mathbf{M} \tag{2.82}$$

onde $\mathbf{D_i}$ é a matriz dinâmica utilizada para cálculo do iézimo autovalor, λ_i, e o seu respectivo autovetor, $\boldsymbol{\varphi}_\mathbf{i}$.

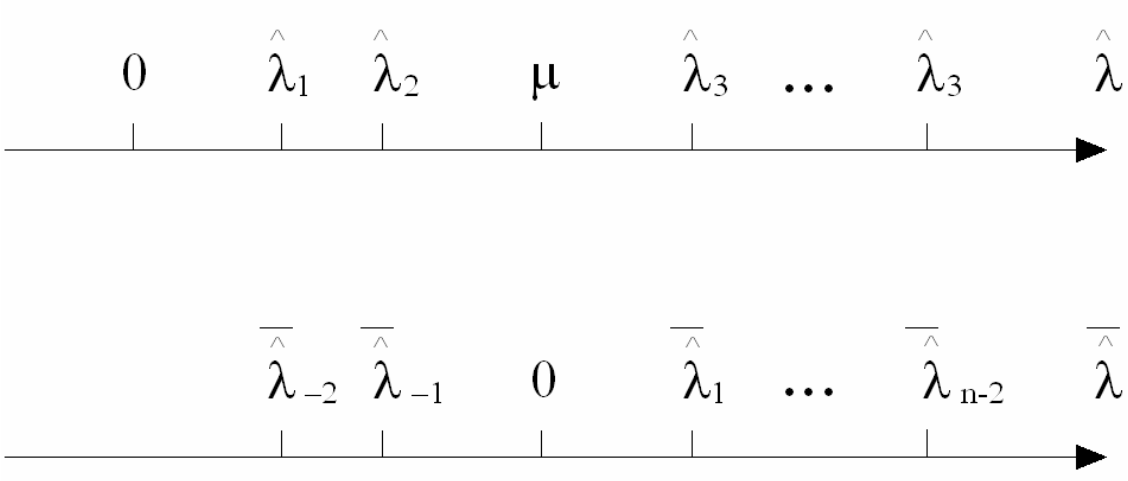


FIGURA 2.1: Exemplo de translação espectral

2.3.5 Método da Iteração no Subespaço

O problema de autovalor generalizado (2.26) pode ser escrito de forma condensada como segue:

$$\mathbf{K\Lambda\Phi} = \mathbf{M\Phi} \tag{2.83}$$

ou se considerando um sistema modal truncado:

$$\mathbf{K\hat{\Lambda}\hat{\Phi}} = \mathbf{M\hat{\Phi}} \tag{2.84}$$

Analisando-se a Eq. (2.84) à luz do método da iteração inversa, é possível notar que o método consiste em se interagir num espaço de dimensão '*N*' de bases **K** e **M** ortogonais até que se convirja para o conjunto de autovetores do problema dinâmico. Os '*N*' autovetores em

$$\mathbf{K}\hat{\mathbf{\Lambda}}\hat{\mathbf{\Phi}} = \mathbf{M}\hat{\mathbf{\Phi}} \tag{2.84}$$

(2.84) são, portanto, uma base para espaço R^N.

K. J. Bathe, em seu trabalho "*Solution Methods of Large Generalized Eingenvalue Problems in Structural Engineering*" de 1971, observou que era mais fácil se escolher um conjunto de vetores arbitrários próximo da base do espaço R^N do que um conjunto de vetores que estivessem próximos aos autovetores do problema de autovetor-autovalor. Baseado nisso, ele desenvolveu um método no qual as iterações procuravam gerar a base para R^N e, a partir daí, gerar os autovetores $\hat{\mathbf{\Phi}}$. Daí o nome de *método de iteração no subespaço*.

O primeiro passo do método desenvolvido por *Bathe* é adotar um subconjunto $\overline{\hat{\mathbf{\Phi}}}$ e obter uma aproximação melhorada $\overline{\overline{\hat{\mathbf{\Phi}}}}$ como segue:

$$\mathbf{K}\overline{\overline{\hat{\mathbf{\Phi}}}}_{\mathbf{j+1}} = \mathbf{M}\overline{\hat{\mathbf{\Phi}}}_{\mathbf{j}} \tag{2.85}$$

Esse passo é basicamente a primeira iteração do método da iteração inversa. Então, obtêm-se as projeções de **M** e **K** no subespaço R^N. Como ao longo do processo R^N vai sendo atualizado, é conveniente se referir ao mesmo como sendo o subespaço da iteração corrente, isto é, R^N_{j+1}:

$$\overline{\mathbf{K}}_{\mathbf{j+1}} = \overline{\overline{\hat{\mathbf{\Phi}}}}^{\mathbf{T}}_{\mathbf{j+1}}\mathbf{K}\overline{\overline{\hat{\mathbf{\Phi}}}}_{\mathbf{j+1}} \tag{2.86}$$

$$\overline{\mathbf{M}}_{\mathbf{j+1}} = \overline{\overline{\hat{\mathbf{\Phi}}}}^{\mathbf{T}}_{\mathbf{j+1}}\mathbf{M}\overline{\overline{\hat{\mathbf{\Phi}}}}_{\mathbf{j+1}} \tag{2.87}$$

A seguir, obtém-se a solução do problema de autovalor no subespaço R^N_{j+1} através de algum método conveniente que pode ser, inclusive, o próprio método da iteração inversa:

$$\overline{\mathbf{K}}_{\mathbf{j+1}}\mathbf{Q}_{\mathbf{j+1}}\hat{\mathbf{\Lambda}}_{\mathbf{j+1}} = \overline{\mathbf{M}}_{\mathbf{j+1}}\mathbf{Q}_{\mathbf{j+1}} \tag{2.88}$$

onde $\mathbf{Q_{j+1}}$ é uma variável auxiliar representando os autovetores do problema de subespaço.

O valor de $\overline{\hat{\mathbf{\Phi}}}$ é então atualizado por :

$$\overline{\hat{\mathbf{\Phi}}}_{\mathbf{j+1}} = \overline{\overline{\hat{\mathbf{\Phi}}}}_{\mathbf{j+1}}\mathbf{Q}_{\mathbf{j+1}} \tag{2.89}$$

Pode se provar que $\overline{\hat{\mathbf{\Phi}}}_{\mathbf{j}}$ converge para os '*N*' primeiros autovetores, desde que $\overline{\hat{\mathbf{\Phi}}}_{\mathbf{1}}$ não possua nenhum vetor ortogonal aos autovetores procurados.

Fica evidente que uma iteração no método do subespaço é computacionalmente mais custosa que uma iteração no método da iteração inversa. O sub-problema (2.88) é outro problema de autovalor dentro do problema de autovalor original. Porém, o problema de autovalor (2.88) é de ordem '*N*'. Como o número de passos para que $\overline{\hat{\mathbf{\Phi}}}$ se aproxime dos valores procurados é significativamente reduzido, no computo geral, o método de *iteração no subespaço* é mais eficiente que o método da *iteração inversa* para problemas modais de médio para grande porte.

2.4 Métodos de Integração Direta

Os métodos de integração direta, como o próprio nome sugere, resolvem diretamente a equação diferencial sem manipulações adicionais como os métodos baseados na decomposição modal. No caso particular desse trabalho, isso significa resolver diretamente a equação diferencial do problema discreto dada por (2.10). A principal vantagem desses métodos é o fato de serem mais gerais, tratando de forma idêntica problemas com variáveis acopladas ou desacopladas, problemas nos quais os carregamentos não possuem uma forma

analítica, problemas cujas equações características possuem raízes complexas, além de sua extensão para problemas não-lineares ser bastante simples.[1]

Os métodos de integração direta podem ser classificados explícitos ou implícitos. Um método de integração explícita tem a forma:

---

[1] O método da superposição modal pode ser usado em problemas não-lineares dividindo-se o problema em subintervalos lineares.

$$\mathbf{U_{j+1}} = \mathbf{f}\left(\mathbf{U_j}, \dot{\mathbf{U}}_\mathbf{j}, \ddot{\mathbf{U}}_\mathbf{j}, \mathbf{U_{j-1}}, \cdots\right) \tag{2.90}$$

enquanto os métodos de integração implícita possuem a forma:

$$\mathbf{U_{j+1}} = \mathbf{f}\left(\dot{\mathbf{U}}_\mathbf{j+1}, \ddot{\mathbf{U}}_\mathbf{j+1}, \mathbf{U_j}, \cdots\right) \tag{2.91}$$

Em suma, os métodos explícitos dependem dos valores de deslocamento e suas derivadas obtidos no passo de tempo anterior ao passo corrente para se atualizar o deslocamento, enquanto os métodos implícitos dependem das derivadas dos deslocamentos do passo corrente para atualizar o deslocamento. O método mais adequado a cada caso é dependente do problema. Weaver e Johnston (1987) analisaram vários métodos e concluíram que, de maneira geral, o método *Newmark-*β com a modificação *Hilber-*α é o método mais eficiente.

2.4.1 Método *Newmark-*β

Considere a equação do movimento discretizada segundo o MEF dada por (2.22), aqui repetida:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{A}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F_e} \tag{2.22}$$

sendo as condições iniciais $\mathbf{u_0}$ e $\dot{\mathbf{u}}_\mathbf{0}$ conhecidas. Considere agora a aceleração no intervalo $\Delta t_i = t_{i+1} - t_i$ como sendo a média entre a aceleração no começo e no final do intervalo:

$$\ddot{\mathbf{u}} = \frac{1}{2}\left(\ddot{\mathbf{u}}_\mathbf{i} + \ddot{\mathbf{u}}_\mathbf{i+1}\right) \tag{2.92}$$

Integrando-se a Eq. (2.92) no intervalo Δt_i duas vezes leva a:

$$\dot{\mathbf{u}}_\mathbf{i+1} = \dot{\mathbf{u}}_\mathbf{i} + \frac{\Delta t_i}{2}\left(\ddot{\mathbf{u}}_\mathbf{i} + \ddot{\mathbf{u}}_\mathbf{i+1}\right) \tag{2.93}$$

e

$$\mathbf{u}_\mathbf{i+1} = \mathbf{u}_\mathbf{i} + \dot{\mathbf{u}}_\mathbf{i}\Delta t + \frac{\Delta t_i^2}{4}\left(\ddot{\mathbf{u}}_\mathbf{i} + \ddot{\mathbf{u}}_\mathbf{i+1}\right) \tag{2.94}$$

Reescrevendo (2.22) em termos das variações das variáveis de estado, tem-se:

$$\mathbf{M}\Delta\ddot{\mathbf{u}} + \mathbf{A}\Delta\dot{\mathbf{u}} + \mathbf{K}\Delta\mathbf{u} = \Delta\mathbf{F}_\mathbf{e} \tag{2.95}$$

Combinando-se as equações (2.93), (2.94) e (2.95), chega-se às seguintes equações:

$$\tilde{\mathbf{K}}_\mathbf{i}\Delta\mathbf{u}_i = \Delta\tilde{\mathbf{F}}_\mathbf{i} \tag{2.96}$$

onde:

$$\tilde{\mathbf{K}}_\mathbf{i} = \mathbf{K} + \frac{2}{\Delta t_i}\mathbf{A} + \frac{4}{\Delta t_i^2}\mathbf{M} \tag{2.97}$$

$$\Delta\tilde{\mathbf{F}}_\mathbf{i} = \Delta\mathbf{F}_\mathbf{i} + \left(\frac{4}{\Delta t_i}\mathbf{M} + 2\mathbf{A}\right)\dot{\mathbf{u}}_i + 2\mathbf{M}\ddot{\mathbf{u}}_\mathbf{i} \tag{2.98}$$

Esse método, denominado método da aceleração média, possui precisão de 1ª ordem.

Considere, agora, que a aceleração varia linearmente, em Δt_i:

$$\dot{\mathbf{u}}_{\mathbf{i+1}} = \dot{\mathbf{u}}_\mathbf{i} + \frac{\Delta t_i}{2}\left(\ddot{\mathbf{u}}_\mathbf{i} + \ddot{\mathbf{u}}_{\mathbf{i+1}}\right) \tag{2.99}$$

$$\mathbf{u}_{\mathbf{i+1}} = \mathbf{u}_\mathbf{i} + \dot{\mathbf{u}}_\mathbf{i}\Delta t + \frac{\Delta t_i^2}{6}\left(2\ddot{\mathbf{u}}_\mathbf{i} + \ddot{\mathbf{u}}_{\mathbf{i+1}}\right) \tag{2.100}$$

$$\tilde{\mathbf{K}}_\mathbf{i} = \mathbf{K} + \frac{3}{\Delta t_i}\mathbf{A} + \frac{6}{\Delta t_i^2}\mathbf{M} \tag{2.101}$$

$$\Delta\tilde{\mathbf{F}}_\mathbf{i} = \Delta\mathbf{F}_\mathbf{i} + \left(\frac{6}{\Delta t_i}\mathbf{M} + 3\mathbf{A}\right)\dot{\mathbf{u}}_i + \left(3\mathbf{M} + \frac{1}{2}\mathbf{A}\right)\ddot{\mathbf{u}}_\mathbf{i} \tag{2.102}$$

Essa formulação, conhecida como método de aceleração linear, o qual é um método de integração implícito, conforme pode se notar das equações (2.99) a (2.102), possui precisão de 2ª ordem. Os métodos de aceleração constante e aceleração linear podem ser expressos por um único conjunto de equações, como segue:

$$\tilde{\mathbf{K}}\Delta\mathbf{U}_{\mathbf{j}} = \Delta\tilde{\mathbf{F}}_{\mathbf{j}} \tag{2.103}$$

$$\Delta\dot{\mathbf{U}}_{\mathbf{j}} = \frac{\gamma}{\beta\Delta t_j}\Delta\mathbf{U}_{\mathbf{j}} - \mathbf{R}_{\mathbf{j}} \tag{2.104}$$

$$\Delta\ddot{\mathbf{U}}_{\mathbf{j}} = \frac{1}{\beta\left(\Delta t_j\right)^2}\Delta\mathbf{U}_{\mathbf{j}} - \mathbf{Q}_{\mathbf{j}} \tag{2.105}$$

$$\tilde{\mathbf{K}} = \mathbf{K} + \frac{1}{\beta\left(\Delta t_j\right)^2}\mathbf{M} + \frac{\gamma}{\beta\Delta t_j}\mathbf{A} \tag{2.106}$$

$$\Delta\tilde{\mathbf{F}}_{\mathbf{j}} = \Delta\mathbf{F}_{\mathbf{j}} + \mathbf{M}\mathbf{Q}_{\mathbf{j}} + \mathbf{C}\mathbf{R}_{\mathbf{j}} \tag{2.107}$$

$$\mathbf{Q}_{\mathbf{j}} = \frac{1}{\beta\Delta t_j}\dot{\mathbf{U}}_{\mathbf{j}} + \frac{1}{2\beta}\ddot{\mathbf{U}}_{\mathbf{j}} \tag{2.108}$$

$$\mathbf{R}_{\mathbf{j}} = \frac{\gamma}{\beta}\dot{\mathbf{U}}_{\mathbf{j}} + \left(\frac{\gamma}{2\beta} - 1\right)\Delta t_j\ddot{\mathbf{U}}_{\mathbf{j}} \tag{2.109}$$

Ao se fixar $\gamma = \frac{1}{2}$ e fazer $\beta = \frac{1}{4}$ ou $\beta = \frac{1}{6}$, as equações (2.105) a (2.109) resultam no método da aceleração constante ou no método da aceleração linear respectivamente. O método de *Newmark-*β consiste em se variar livremente γ e β de modo a se determinar empiricamente qual par (γ,β) obtém a melhor relação entre precisão e estabilidade numérica. Zienkiewicz (1977) demonstra que o método de *Newmark-*β corresponde a se aplicar uma discretização pelo MEF da variável '*t*' utilizando elementos finitos de tempo de 2ª ordem.

Para $\gamma \neq 0{,}5$, observa-se um amortecimento artificial, denominado amortecimento numérico. Esse amortecimento diminui a precisão do resultado, e poder-se-ia supor indesejado. Porém, uma certa quantidade de amortecimento numérico é desejável para se reduzir o efeito espúrio dos modos mais altos em sistemas não amortecidos ou com baixo amortecimento (Cook, Malkus e Plesha, 1989). Cook, Malkus e Plesha (1989) também salientam que o método de

Newmark-β possui precisão de 2ª ordem para $\gamma = \frac{1}{2}$ e $\beta = \frac{1}{6}$, e para qualquer outra escolha do par (γ,β), o resultado tem precisão inferior à de 2ª ordem.

Os métodos de integração direta podem apresentar problemas de estabilidade numérica à medida que se aumenta o passo de tempo Δt. Dependendo do par (γ,β) selecionado, método de *Newmark-*β pode ser incondicionalmente convergente, condicionalmente convergente ou sempre divergente. Zienkiewicz (1977) demonstra que as seguintes relações devem ser respeitadas para que o método de *Newmark-*β seja incondicionalmente convergente:

$$\beta \geq \frac{1}{4}\left(\frac{1}{2} + \gamma\right)^2 \tag{2.110}$$

$$\gamma \geq \frac{1}{2} \tag{2.111}$$

Caso (γ,β) sejam tais que ainda se tenha estabilidade condicional, existe um Δt e um ΔF_e a partir do qual o método é convergente. Esse valor Δt é denominado a *variação crítica* para esse par (γ,β). Se Δt e ΔF_e forem maiores que os valores críticos para o par (γ,β) adotado, o método não convergirá.

2.4.2 A Variação *Hilber-*α do Método *Newmark-*β

Hilber, et al., *apud* Weaver e Johnston (1987), derivaram equações similares às derivadas para a obtenção do método de *Newmark-*β, porém com a aceleração variando quadraticamente ou cubicamente no intervalo de tempo Δt, ou seja, a precisão máxima possível é de 4ª ordem. Zienkiewicz (1977) demonstra que o método *Hilber-*α equivale a se adotar um elemento finito de tempo de 4ª ordem.

As expressões para o do método *Hilber-*α são:

$$\tilde{\mathbf{K}}_{\boldsymbol{\alpha}} \Delta \mathbf{U}_{\mathbf{j}} = \Delta \tilde{\mathbf{F}}_{\boldsymbol{\alpha}\mathbf{j}} \tag{2.112}$$

onde:

$$\tilde{\mathbf{K}}_{\boldsymbol{\alpha}} = \tilde{\mathbf{K}} + \alpha\mathbf{K} \tag{2.113}$$

$$\Delta\tilde{\mathbf{F}}_{\boldsymbol{\alpha j}} = \Delta\tilde{\mathbf{F}}_{\mathbf{j}} + \alpha\mathbf{K}\Delta\mathbf{U}_{\mathbf{j-1}} \tag{2.114}$$

A variável α é o grau de liberdade introduzido para expandir a interpolação da variável tempo até a 4ª ordem. A variação *Hilber-*α consiste em deixar a tríade (α,β,γ) variar livremente de modo semelhante ao feito para o método do *Newmark-*β. Pode se concluir, então, que a melhor precisão possível para esse método é a de 4ª ordem.

Weaver e Johnston (1987) realizaram uma análise comparativa entre alguns métodos de integração direta e concluíram que, de maneira geral, o método de *Newmark-*β com a variação de *Hilber-*α leva a melhores resultados. Além disso, eles apontam α = -0,1, β = 0,3025 e γ = 0,6 como uma seleção ótima para um caso geral.

2.5 Integração Numérica – A Quadratura de Gauss

Tanto no caso dos métodos de decomposição modal quanto nos métodos de integração direta da equação do movimento, as matrizes de massa, rigidez e amortecimento são integrais que, no processo computacional, devem ser obtidas numericamente. No caso dos métodos de superposição modal, a integração dos deslocamentos nas direções principais também deve ser feita numericamente.

Para o caso particular de uma análise dinâmica utilizando-se elementos finitos paramétricos, as seguintes integrais devem ser obtidas pelas Eqs. (2.12), (2.13) e (2.14) aqui repetidas:

$$\mathbf{M}^{\mathbf{e}} = \left[m_{ij}\right]^{e} = \int_{V} \rho\mathbf{N}^{\mathbf{T}}\mathbf{N}dV \tag{2.12}$$

$$\mathbf{K}^{\mathbf{e}} = \left[k_{ij}\right]^{e} = \int_{V} \mathbf{B}^{\mathbf{T}}\mathbf{D}\mathbf{B}dV \tag{2.13}$$

$$\mathbf{f}_{\mathbf{eq}}^{\mathbf{e}} = \int_V \mathbf{N}^{\mathbf{T}} \mathbf{b} dV \tag{2.14}$$

Para o caso de um elemento finito paramétrico com amortecimento viscoso linear, deve-se utilizar a Eq. (2.23) aqui repetida[1]:

$$\mathbf{A}^{\mathbf{e}} = \left[a_{ij}\right]^e = \int_V a\mathbf{N}^{\mathbf{T}}\mathbf{N} dV \tag{2.23}$$

Para os casos de separação modal, deve-se utilizar alguma técnica para se integrar os deslocamentos nas direções principais. No caso do presente trabalho, essa integração é feita utilizando-se a integral de *Duhamel* dada pela Eq. (2.50) aqui repetida:

$$\eta_i = \frac{\mathrm{e}^{-\zeta_i \omega_i t}}{\omega_d^i} \int_0^t \mathrm{e}^{\zeta_i \tau} \; f_{ep}^{\,i} \operatorname{sin} \omega_d^i (t-\tau) d\tau \tag{2.50}$$

Ao se utilizar o método de superposição de acelerações modais, deve-se obter também as velocidades nas direções principais. No presente trabalho, as velocidades são obtidas através da Eq. (2.70) aqui repetida:

$$\dot{\eta}_i = \frac{\mathrm{e}^{-\zeta_i \omega_i t}}{\omega_{di}} \int_0^t \mathrm{e}^{\zeta_i \omega_i \tau} \; f_{epi} \left[\cos \omega_{di}(t-\tau) + \omega_i \operatorname{sen} \omega_{di}(t-\tau)\right] d\tau \tag{2.70}$$

Os métodos de integração numérica possuem a seguinte forma geral:

---

[1] No caso de amortecimento de *Rayleigh*, é preferível utilizar diretamente a Eq. (2.46).

$$I = \int_{\mathbf{x}} f(\mathbf{x}) d\mathbf{x} \approx$$

$$\approx \sum_{h=1}^{m} \sum_{i=1}^{n} \cdots \sum_{j=1}^{o} \cdots \sum_{k}^{p} f\left(x_1, x_2, \cdots, x_\varphi, \cdots, x_\kappa\right) W_h^1 W_i^2 \cdots W_j^\varphi \cdots W_k^\kappa$$

(2.115)

A Eq. (2.115) aproxima uma integral através de um somatório onde a função no integrando é calculada em alguns pontos, $\left(x_1, x_2, \cdots, x_\varphi, \cdots, x_\kappa\right)$, multiplicado por pesos $\left(W_h^1, W_i^2, \cdots W_j^\varphi, \cdots W_k^\kappa\right)$ adequados. Não é necessário, mas é a prática geral, adotar-se $m = n = \cdots = o = \cdots = p$.

A quadratura de Gauss é o processo de integração numérica ótimo para integrais uni e bidimensionais de polinômios. De forma a generalizar o método, é conveniente fixar o intervalo de integração como sendo entre -1 e +1, sendo necessário, portanto, realizar uma mudança de variáveis de maneira que o domínio de integração satisfaça a esse requisito. Esse processo de transformação do domínio de integração pode ser feito através de uma parametrização, onde as variáveis ξ, θ e χ mapeiam o elemento nas direções 1, 2 e 3 do sistema cartesiano, respectivamente. Nesse sistema de variáveis auxiliar, as integrais (2.12), (2.13), (2.14) e (2.23) ficam:

$$\mathbf{M^e} = \left[m_{ij}\right]^e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \rho \mathbf{N^T N} \left|\mathbf{J}\right| d\xi d\theta d\chi \tag{2.116}$$

$$\mathbf{K^e} = \left[k_{ij}\right]^e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \mathbf{B^T D B} \left|\mathbf{J}\right| d\xi d\theta d\chi \tag{2.117}$$

$$\mathbf{f_{eq}^e} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \mathbf{N^T b}\left(\xi, \eta, \zeta\right) \left|\mathbf{J}\right| d\xi d\theta d\chi \tag{2.118}$$

$$\mathbf{A^e} = \left[a_{ij}\right]^e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} a \mathbf{N^T N} \left|\mathbf{J}\right| d\xi d\theta d\chi \tag{2.119}$$

sendo **J** é a matriz jacobiana da transformação do domínio original para o domínio (ξ, θ, χ).

Para o caso das integrais no tempo:

$$\eta_i = \frac{\mathrm{e}^{-\zeta_i t}}{\omega_d^i} \int_{-1}^{1} \mathrm{e}^{\zeta_i \tau(\xi)} \; f_{ep}^{\,i}\left(\tau(\xi)\right) \operatorname{sen} \omega_d^i \left(t - \tau(\xi)\right) J d\xi \tag{2.120}$$

$$\dot{\eta}_i = \frac{\mathrm{e}^{-\zeta_i \omega_n^i t}}{\omega_n^i} \int_{-1}^{1} \mathrm{e}^{\zeta_i \omega_n^i \tau(\xi)} \; f_{ep}^{\,i}\left(\tau(\xi)\right) \Pi_i J d\xi \tag{2.121}$$

onde

$$\Pi_i = \cos \omega_d^i \left(t - \tau(\xi)\right) + \omega_n^i \operatorname{sen} \omega_d^i \left(t - \tau(\xi)\right) \tag{2.122}$$

Utilizando-se as expressões (2.116) a (2.122), basta aplicar a Eq. (2.115) com os pontos e pesos da quadratura de Gauss para calcular as respectivas integrais.

Capítulo 3

MÉTODOS E RECURSOS PARA IMPLEMENTAÇÃO COMPUTACIONAL

3.1 Introdução

Dentre possíveis definições para computação, uma que atende aos propósitos deste trabalho é de se trata de um processo lógico no qual se procura separar, organizar, classificar e realizar cálculos com o objetivo de solucionar um problema. A computação é uma das atividades intelectuais mais antigas da humanidade. A linguagem falada, por exemplo, envolve um processo computacional bastante sofisticado.

A computação como processo abstrato independente, ou seja, como processo mental consciente e intencional do ser humano, também é uma atividade bastante antiga. Afinal, esse processo surgiu junto com a necessidade de quantificar, classificar, comparar, associar e armazenar informação (dados). Nesse instante, o ser humano passou a lidar com o problema de qual seria a melhor metodologia para desempenhar essas tarefas.

O desenvolvimento da matemática demandou que o processo computacional se tornasse mais sistemático, surgindo, inclusive, uma diferenciação clara entre os métodos computacionais matemáticos dos demais. Embora não haja exclusão entre essas duas categorias, os métodos computacionais ligados à matemática possuem uma formalização que é de extensão bastante difícil aos métodos computacionais em geral, enquanto os métodos computacionais de uso geral são demasiadamente subjetivos, não tendo muita utilidade para os problemas matemáticos.

Os processos computacionais, embora em contínuo desenvolvimento, não se constituíram formalmente como áreas independentes do conhecimento até a época da Revolução Industrial. O automatismo introduzido conjugado com processos mais extensos e complexos, impulsionou o estudo de modelos computacionais. A computação deixava de ser um processo auxiliar na realização de uma dada tarefa para ser o processo construtor da tarefa.

O advento da computação digital na década de 1950, aumentou dramaticamente as possibilidades da computação. Embora limitada com relação às abstrações suportadas, a velocidade e a capacidade de armazenamento dos computadores digitais permitia a modelagem de sistemas bastante complexos por composição das poucas operações básicas. A tarefa de programação computacional no começo da década de 1950 consistia em implementar códigos que correspondiam fielmente à realidade interna da máquina, exigindo que o programador conhecesse os princípios de construção das mesmas. O baixo nível da linguagem (nível de abstrações suportadas), tornava a programação acessível apenas para especialistas.

Logo se viu a necessidade de se desenvolver linguagens de mais alto nível. O primeiro passo para isso foi o desenvolvimento de interpretadores e compiladores capazes de transformar uma linguagem simbólica em linguagem de máquina. O uso de uma linguagem simbólica aumentou significativamente o nível de abstração possível, ficando a parte mais especializada e maçante de se descrever um algoritmo através dos poucos recursos elementares do computador a cargo dos compiladores e interpretadores. Até hoje, as linguagens simbólicas suportadas pelos compiladores são bastante limitadas, mas elas são, indubitavelmente, mais amigáveis ao processo cognitivo humano do que a linguagem de máquina.

A primeira linguagem simbólica desenvolvida, o FORTRAN, utilizava o paradigma imperativo. A abstração da programação imperativa consiste em se possuir variáveis que

apresentam estados transitórios modificados por comandos. Os códigos que utilizam a programação imperativa são, geralmente, *estruturados*. A *programação estruturada* consiste em criar o código na mesma seqüência em que este será executado. A programação imperativa, quando o código é feito de maneira estruturada, reflete basicamente o processo computacional feito pelo computador.

Os paradigmas computacionais que se seguiram ao imperativo foram o paradigma funcional, o paradigma de orientação a objetos e o paradigma lógico. O paradigma funcional e o paradigma lógico, talvez devido ao baixo desempenho e o alto nível de abstração, são pouco utilizados na engenharia.

A programação orientada a objetos (POO) surgiu na segunda metade da década de 1960 com a linguagem SIMULA. O paradigma de orientação a objetos é, na realidade, implementado sobre uma plataforma de programação imperativa, tanto que é possível fazer uma programação estruturada utilizando uma linguagem que suporte a POO.

3.2 Programação Orientada a Objeto

A programação imperativa possui três características importantes:

i. Eficiência: como já foi dito, a linguagem imperativa é próxima ao processo interno das máquinas. Por essa similaridade, os códigos feitos em linguagem imperativa são mais eficientes.
ii. Fácil modelagem dos problemas: a programação estruturada é natural para o ser humano, e os modelos tendem a ser de concepção e implementação simples.
iii. Método consolidado: existe grande experiência no uso do método e as tecnologias estão bem amadurecidas.

Entretanto, a programação estruturada, a forma mais simples de programação imperativa, possui uma característica que cria dificuldades à medida que o código cresce: tem-se total liberdade para manipular as variáveis do ambiente, podendo as mesmas ser acessadas a qualquer momento e em qualquer ponto até mesmo por operações não permitidas por essas variáveis. Essa liberdade dificulta o controle e a legibilidade do código.

A FIGURA 3.1 mostra um diagrama que ajuda a entender o inconveniente de se ter livre acesso ao conteúdo de uma variável. O **Bloco** de comandos **A** manipula a variável 'ÂNGULO' e, ao final do bloco, essa variável dá a medida de um ângulo em radianos. Ao se implementar o **Bloco B** de comandos, deve-se estar ciente que 'ÂNGULO' está em radianos. Se ao final de **B**, 'ÂNGULO' é transformada em graus, no **Bloco C** de comandos, ocorrerá um erro, pois **C** também necessita que 'ÂNGULO' seja dado em radianos.

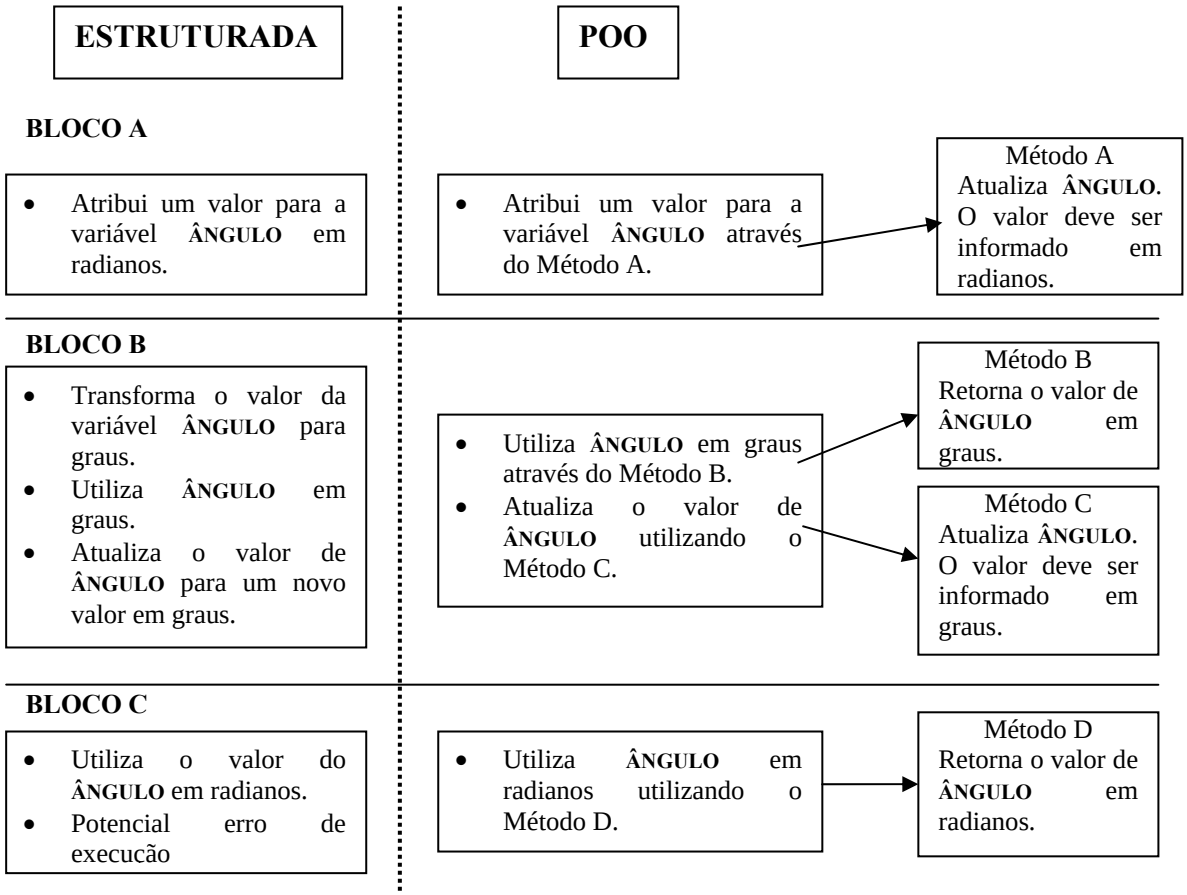


FIGURA 3.1: Diagrama com blocos de comando numa linguagem estruturada e na POO

A programação estruturada apresenta as seguintes dificuldades:

i. Não é possível isolar partes do programa. Para se implementar parte do código, é necessário conhecer ou ter algum tipo de controle sobre o restante do código.

ii. A implementação de uma parte do programa pode afetar outras partes já implementadas corretamente.

iii. A manutenção e a expansão são difíceis, o programador que implementa uma nova funcionalidade em um programa já operacional deve compreender bem o mesmo. Essa nova implementação pode obrigar que uma grande extensão do código seja revisada. A manutenção e a expansão podem ser mais trabalhosas do que a implementação básica.

iv. Quando se trata de um aplicativo desenvolvido por um único desenvolvedor, os riscos inerentes à liberdade de se ter livre acesso às variáveis são menores, mas todo o controle terá que ser feito por esse desenvolvedor, uma vez que o método não provê formas de protegê-lo.

Certamente, a POO não é muito atraente quando o problema a ser resolvido e o código resultante são simples. Nesses casos, erros como o mostrado na Figura 3.1 são facilmente evitados. Em um ambiente orientado a objetos, o desenvolvedor deve respeitar o rigor organizacional da POO. O controle e a modularidade passam a ser interessantes quando:

i. Os Programas são extensos.

ii. Os Programas são desenvolvidos por equipes de programadores.

iii. Os programas possuem um ciclo de vida longo, sujeitos à manutenção e a expansões.

Quando se fala de controle na POO, refere-se tanto ao controle de cada parte do código, como também o controle do projeto. Como se verá adiante, a POO e a divisão do problema em classes são quase sinônimos. Assim, a POO já apresenta uma forma propícia à elaboração de um projeto para o código.

A seguir, apresentar-se-á alguns conceitos básicos da POO.

3.2.1 Objeto

Horstman (1997) define que objetos são entidades que armazenam dados, definem métodos e se relacionam entre si conforme um conjunto de regras.

As três características básicas de um objeto são estado, operações e identidade. A configuração em um dado instante das informações armazenadas num objeto define seu estado; o objeto suporta um determinado conjunto de operações; a identidade, por fim, distingue objetos que eventualmente possuam o mesmo estado, suportem o mesmo conjunto de operações, mas sejam diferentes, podendo, em outro instante, possuir estados distintos.

O objeto encerra em si todo o paradigma da POO. Eles são os responsáveis pela segmentação de um código. Na programação estruturada, define-se variáveis e operações capazes de manipular essas variáveis. Na POO, o objeto possui suas próprias variáveis e define as operações que ele suporta. As demais características de uma linguagem que suporte a POO são conseqüência das características básicas definidas pelo conceito de objeto. Outras características comuns à maioria das linguagens de programação não são inerentes à POO.

Seja um objeto *solução* que se constitua na solução do sistema algébrico $AX = B$. Esse objeto deve possuir os campos[1] para armazenar a matriz A e o vetor B, métodos que permitam que sejam informados seus campos e um método que calcule e retorne o vetor solução X. Os objetos externos ao objeto *solução* não têm acesso à forma como o objeto obtém a solução do sistema, estando essa, portanto, isolada do resto do programa. Não é importante para o objeto *solução* saber se esse sistema algébrico representa um sistema de cargas e fontes num circuito elétrico, ou uma estrutura discretizada, enquanto que, para o objeto que gerou o sistema algébrico, não interessa saber qual é o método matemático está implementado dentro do objeto *solução*.

É dessa segregação que vem o conceito de "resultado contratado" na POO. Desde que os objetos externos forneçam as informações que o objeto *solução* precisa, esse retornará o resultado dentro dos padrões especificados. Essa característica também é o "inconveniente" de se utilizar a POO. Se o programador, por exemplo, tiver um problema em que 'A' é singular e queira um método de solução que elimine a singularidade e trabalhe com as informações relevantes de 'A', o objeto *solução* será inútil, pois o programador não terá como manipulá-lo. Isso torna as implementações iniciais em POO mais difíceis.

3.2.2 Classe

A maioria das linguagens de programação orientada a objetos suporta o conceito de classe. A classe define uma coleção de objetos do mesmo tipo. Ou seja, os mesmos campos, os mesmos estados admissíveis e diferentes identidades. O que se faz, na prática é definir os campos e métodos de objetos nas classes às quais eles pertencem. O objeto é um espécime dessa classe

[1] Santos (2003) denomina as variáveis declaradas num objeto de campos. Essa notação evita confusões entre variáveis locais, que podem ser declaradas em vários pontos, sendo a sua visibilidade dependente do ponto onde a mesma é declarada, e variáveis de classe, que são visíveis apenas dentro do objeto.

e é dito ser uma *instância* da mesma. O conceito de classe dentro da POO também é útil para definir regras de relacionamento entre os objetos de uma mesma classe e entre objetos de classes distintas.

3.2.3 Herança

Uma classe pode ser muito similar à outra. Eventualmente, pode ser uma versão adaptada ou mais especializada de uma dada classe. Em POO, é possível criar uma nova classe, denominada subclasse, a partir de uma classe já existente, dita superclasse, classe pai ou classe base. Esse processo, no qual os objetos da nova classe constituem um subgrupo da classe original, é denominado herança. Os objetos da subclasse devem possuir todos os atributos dos objetos da superclasse, mas podem possuir pequenas diferenças nesses atributos, bem como possuir atributos adicionais não existentes nos objetos da classe original. Legalmente, o objeto de uma subclasse deve ser invocável em todos os casos nos quais é possível se invocar um objeto da superclasse, mas a recíproca não é verdadeira.

3.2.4 Associação e Agregação

Quando duas classes se relacionam, é dito haver uma associação entre elas. Associação entre classes ocorre quando há operações dos objetos de uma classe sobre os objetos da outra. A agregação é um caso particular de associação que é plenamente atendido quando os objetos de uma classe ou contém ou exclusivamente operam objetos de uma outra classe.

3.2.5 Encapsulamento – Campos Públicos, Protegidos e Privados

Ao se declarar uma classe, existem três tipos de atributos possíveis para os seus campos e classes: *público*, *protegido* ou *privado*. O atributo *público* significa que o campo ou o método pode ser acessado ou invocado por qualquer objeto do programa diretamente. O atributo *protegido* significa que só os objetos da classe ou de classes herdeiras podem acessar diretamente os campos e invocar os métodos dos objetos dessa classe. O atributo *privado* significa que somente os objetos da classe têm acesso direto aos campos e podem invocar os métodos dessa classe.

Os atributos *protegido* e *privado* dão origem ao encapsulamento na POO. Métodos, em geral, possuem atributos públicos, embora possa ser útil criar métodos com atributo de *protegido* ou *privado*. Porém, variáveis só devem ser declaradas com atributos de *protegidas* ou *privadas*.

Mas qual a razão de não se declarar campos como sendo *públicos*? A razão para isso resume em si o paradigma da POO. Se uma variável pode ser acessada em qualquer ponto do programa, para qualquer modificação que se faça, será necessário se revisar todo o programa para se saber o impacto daquela mudança.

3.2.6 Polimorfismo e Sobrecarga

As linguagens que suportam a POO permitem que se declarem métodos e operadores diferentes com o mesmo nome. Quando isso é feito, diz-se haver uma *sobrecarga* de métodos ou operadores. A sobrecarga é um tipo de *polimorfismo*. Os compiladores são capazes de saber qual método ou operador está sendo invocado a partir da lista de argumentos. A lista de argumentos é conhecida como *assinatura*. A sobrecarga é útil quando métodos ou operadores similares diferem no tipo de argumento que é passado.

O conceito de polimorfismo, porém, é mais amplo que o de sobrecarga. À medida que se criam novas classes por herança, as subclasses podem redefinir métodos e campos já presentes na superclasse. Assim como no caso de sobrecarga, os compiladores são capazes de diferenciar as entidades corretas a partir dos argumentos passados ou a partir dos objetos a partir dos quais os métodos são chamados ou os campos são acessados.

3.2.7 Classes Abstratas e Interfaces

Classes abstratas podem ser usadas para declarar métodos abstratos. A classe abstrata declara o método, mas não o implementa, ficando isso como uma obrigatoriedade para todas as classes que herdem da classe abstrata. As classes abstratas não podem ser instanciadas. Esse recurso é útil quando um método é necessário a uma dada classe, mas é preferível delegar a sua implementação às suas subclasses mais especializadas.

A interface possui um mecanismo semelhante ao das classes abstratas, porém, todos os métodos declarados numa interface são abstratos. A interface não declara campos, o que é possível numa classe abstrata, e a implementação de uma interface não se dá pelo mecanismo de herança, podendo ser feito por qualquer classe. Outra diferença importante, é que uma classe pode implementar qualquer número de interfaces, enquanto pode herdar apenas de uma única classe.

As interfaces são úteis por permitirem um controle maior sobre uma dada classe. Se uma classe implementa uma interface, sabe-se que todos os métodos dessa interface foram implementados.

3.3 Representação Gráfica na POO – A UML

A apresentação gráfica de um programa orientado a objeto é um artifício que facilita a entendimento das entidades do programa e suas relações. A linguagem gráfica para modelagem de problemas via orientação a objetos mais aceita é a *Unified Modeling Language* (UML).

A UML surgiu da união de três metodologias de modelagem: o método de *Booch*, de Grady Booch, o método *Object Modeling Technique* (OMT), de Ivar Jacobson e o método OOSE (*Object-Oriented Software Engineering*), de Jim Rumbaugh. Essas eram as três metodologias de modelagem orientada a objetos mais populares entre os profissionais da área de engenharia de software até meados da década de 1990. A união dessas metodologias contou com o amplo apoio da *Rational Software*, que também financiou o projeto. Em 1997, a *Object Management Group* (OMG) adotou a UML como padrão de modelagem e se encarregou do seu desenvolvimento (Guedes, 2005).

“A UML é bem mais ampla do que uma ferramenta para visualização da arquitetura de um programa orientado a objetos. Segundo seus desenvolvedores (Booch, Rumabaugh e Jacobbson), a finalidade da UML é proporcionar um padrão de planejamento de arquitetura de projetos de sistemas, incluindo aspectos conceituais, como processos de negócios e funções de sistema, além de itens concretos, como as classes escritas em determinada linguagem de programação, esquemas de bancos de dados e componentes de software reutilizáveis” (Apud Medeiros, 2004).

A UML não indica a codificação do programa em si, sendo um artifício para modelagens num nível mais alto. Dessa maneira, a UML não está vinculada a nenhuma linguagem, sendo possível usá-la, inclusive, para linguagens de programação que não suportem implicitamente o paradigma de orientação a objetos. Essa visão mais ampla também permite que a UML seja acessível aos não programadores. Isso facilita uma maior inclusão do usuário final no processo de programação quando esses não são programadores. “Em geral, será necessária

uma documentação adicional para se sair do diagrama e se chegar efetivamente ao código, porém, essa comunicação deve ser minimalista. Uma documentação extensa nessa passagem pode, inclusive, ser um indício de que a diagramação deve ser revisada" (Medeiros, 2004).

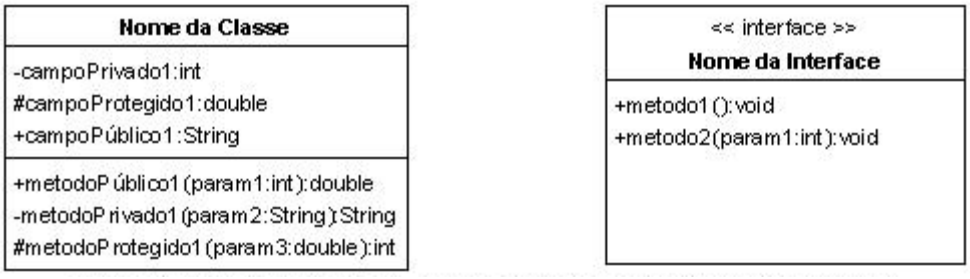
A UML, na sua versão 2.0, consta de 14 diagramas. A multiplicidade de diagramas permite várias visões do problema, sendo os diagramas, portanto, complementares entre si. Porém, é comum os diagramas possuírem áreas de sobreposição. A sobreposição é útil para a detecção de erros na modelagem (Guedes 2005). Porém, não é necessário construir os 14 diagramas para se realizar uma análise utilizando a UML.

Para o presente trabalho, limitar-se-á aos Diagramas de Classe e ao Diagrama de Seqüência. Embora outros diagramas possam ser úteis para a representação da expansão feita, os dois diagramas citados são suficientes para dar uma visão geral do trabalho.

3.3.1 Diagrama de Classes

A Figura 3.2(a) mostra a representação de uma classe na UML. O diagrama pode ser dividido em até três partes. Na primeira divisão coloca-se o nome da classe. Na segunda divisão, coloca-se o nome dos campos, e na última divisão, os operadores (métodos) dessa classe. O símbolo que precede o nome do campo ou do método indica a sua visibilidade: '-' para privado, '#' para protegido e '+' para público. No caso dos campos da classe, o tipo de campo é colocado após o nome do mesmo. Para os métodos, escrevem-se os parâmetros de chamada e o tipo retornado pela chamada do mesmo.

Os métodos abstratos são anotados em itálico. Para a interface - Figura 3.2(b) - isso não é necessário pois todos os métodos são abstratos.



(a) (b)

FIGURA 3.2: Exemplos de diagrama de classe na UML

Algumas classes possuem vários métodos e campos, não sendo prático escrever todos eles no diagrama. Limita-se, então, a escrever os campos e métodos mais relevantes da classe.

Em algumas linguagens de programação orientada a objetos, existe também o recurso dos *pacotes* que permitem organizar classes afins que não se relacionam pelo mecanismo de herança. Embora possa haver classes dentro de um pacote que não se relacionam por herança, a herança só é possível dentro de um mesmo pacote. A Figura 3.3 mostra um exemplo de pacote contendo uma classe e uma interface.

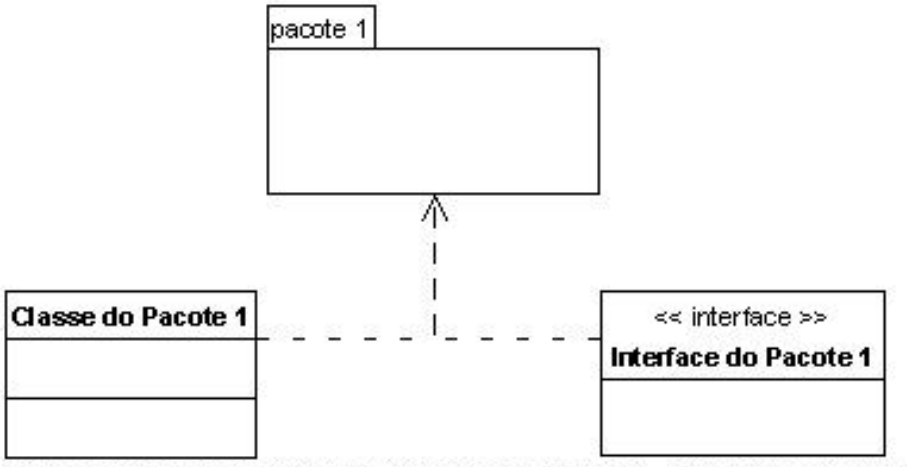


FIGURA 3.3: Exemplo de pacote

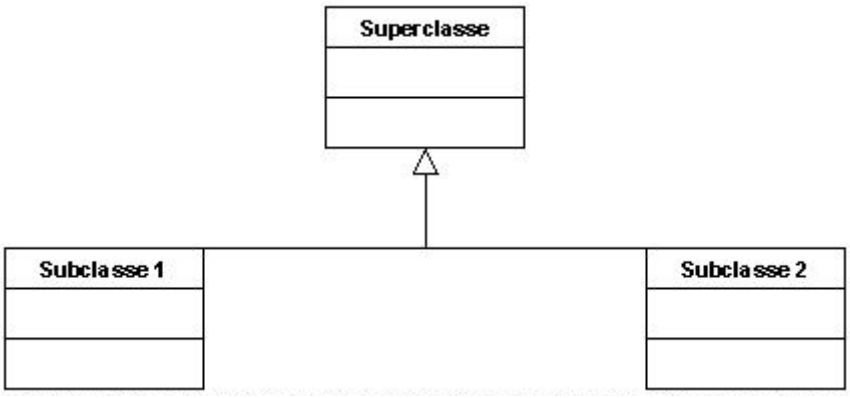


FIGURA 3.4: Exemplo de herança

Um exemplo de herança é mostrado na Figura 3.4. O sentido da seta é obrigatório para indicar o sentido da herança. A Figura 3.5 mostra algumas relações possíveis entre classes. Essas relações são descritas na Tabela 3.1. Porém, o tipo de relacionamento entre as classes pode ser bem vasto, de modo que poucos casos possuem uma diagramação específica. Nos demais

casos, pode se utilizar o recurso dos *estereótipos* que consiste da anotação da relação dentro dos sinais “<< >>”. O emprego dos estereótipos não se limita a explicar relações de classes, sendo utilizados para detalhar qualquer relação em qualquer dos diagramas UML. Mesmo quando se tem uma diagramação específica para demonstrar a relação, os estereótipos podem ser utilizados para prover informações adicionais.

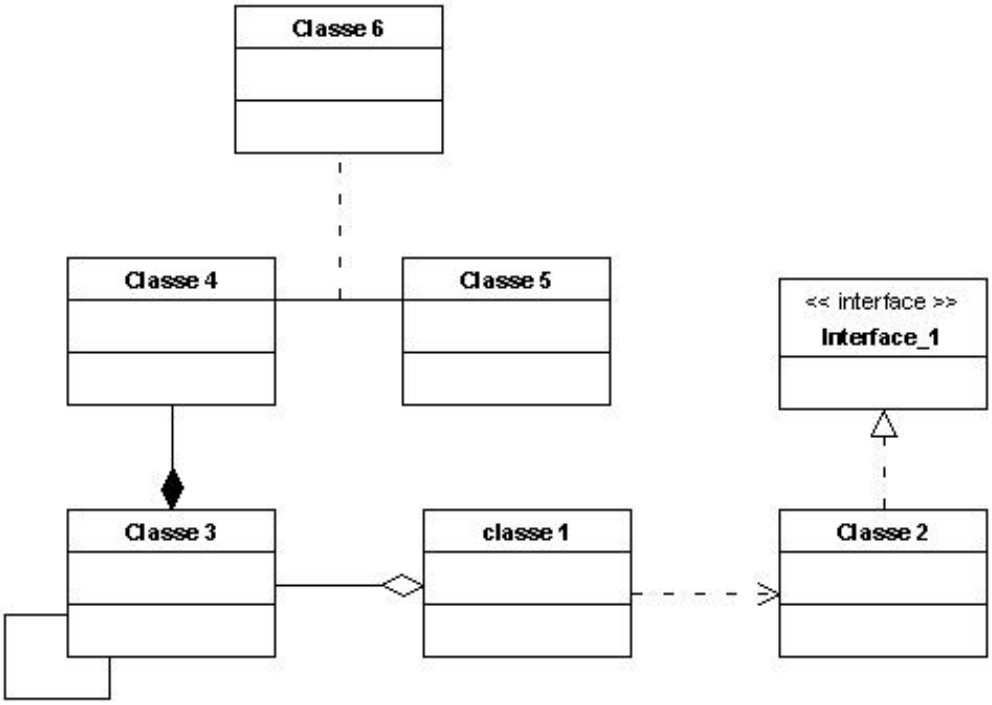


FIGURA 3.5: Exemplo de relações entre classes

TABELA 3.1: Relações entre as classes mostradas na Figura 3.5

CLASSES RELACIONADAS	RELAÇÃO	COMENTÁRIO
1 e 2	Dependência	Alguma funcionalidade da classe 1 depende de uma funcionalidade disponível na classe 2
1 e 3	Agregação	A classe 1 agrega objetos da classe 3.
2 e interface 1	Realização	A classe 2 implementa a interface 1.

CLASSES RELACIONADAS	RELAÇÃO	COMENTÁRIO
3	Associação unária ou reflexiva.	Algum caso de associação ocorre entre objetos da mesma classe
3 e 4	Composição	Objetos da classe 4 compõem algum objeto da classe 3.
4, 5 e 6	Colaboração	A associação das classes 4 e 5 tem uma colaboração por parte da classe 6.
4 e 5	Associação	A associação entre as classes 4 e 5 não se enquadra dentro de outras formas de associação. Geralmente, a natureza da associação é explicada com o uso de estereótipo.

A Figura 3.6 mostra um diagrama de classes bastante completo. Nessa figura também é mostrada a anotação da multiplicidade da relação. A multiplicidade indica quantas instâncias de uma classe se relacionam com a classe instanciadora. A Tabela 3.2 explica as relações de multiplicidade da Figura 3.6.

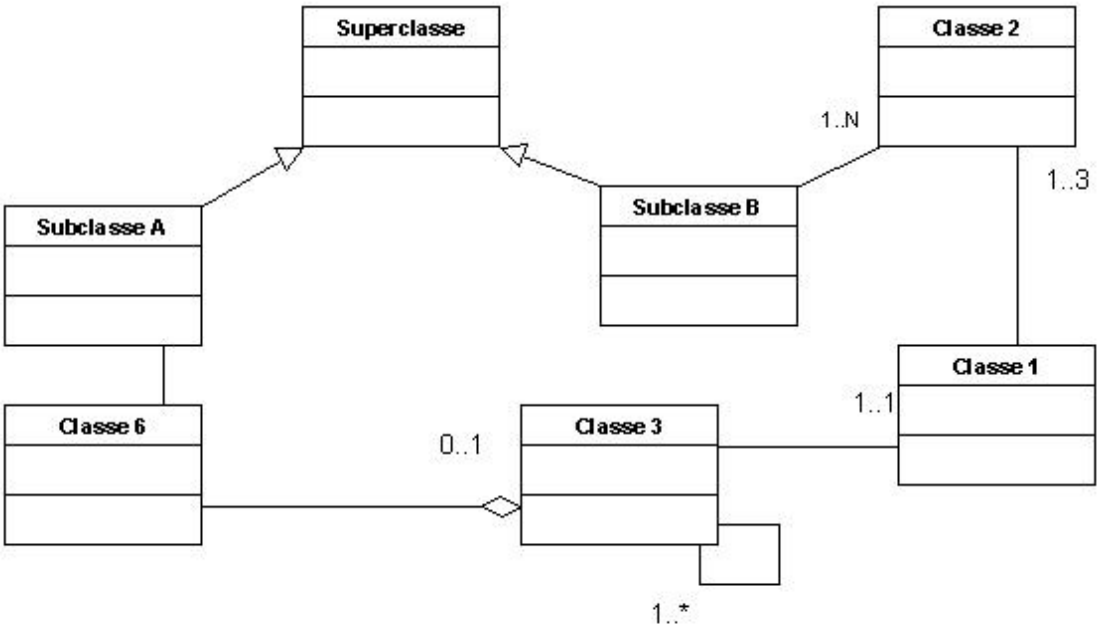


FIGURA 3.6: Diagrama de classes mostrando relações diversas e suas multiplicidades

TABELA 3.2: Relações de multiplicidade da Figura 3.6

CLASSES	MULTIPLICIDADE	COMENTÁRIO
1 e 2	1..3	Um objeto da Classe 2 se relaciona com no mínimo 1 e no máximo 3 instâncias da Classe 1
1 e 3	1..1	Uma instância da Classe 1 se relaciona com 1 instância da Classe 3.
3	1..*	Uma instância da Classe 3 se relaciona com no mínimo 1 e o número máximo de relacionamentos com instâncias da própria classe 3 é indefinido.
3 e 6	0..1	Uma instância da Classe 2 pode agregar nenhuma ou 1 instância da Classe 6.
2 e B	1..N	Uma instância da Classe 2 se relaciona no mínimo com 1 e no máximo com N instâncias da Subclasse B. Esse exemplo mostra que é comum utilizar notação algébrica para representar a multiplicidade.

É comum se dividir os diagramas de classes em subdiagramas de maneira a melhorar a legibilidade. Além disso, não é necessário incluir todos os relacionamentos e informações nos diagramas, podendo se limitar às informações mais relevantes. Portanto, os diagramas de classe permitem certa flexibilidade para o analista.

3.3.2 Diagrama de Seqüência

Esse diagrama permite visualizar a linha de tempo do programa, sendo possível, inclusive, verificar quais objetos foram criados, se eles estão ativos ou inativos, ou se eles já foram destruídos.

A linha de vida de um objeto representa o tempo em que o objeto esteve presente no programa. Ela é representada por uma linha fina tracejada, conforme mostrado na Figura 3.7.

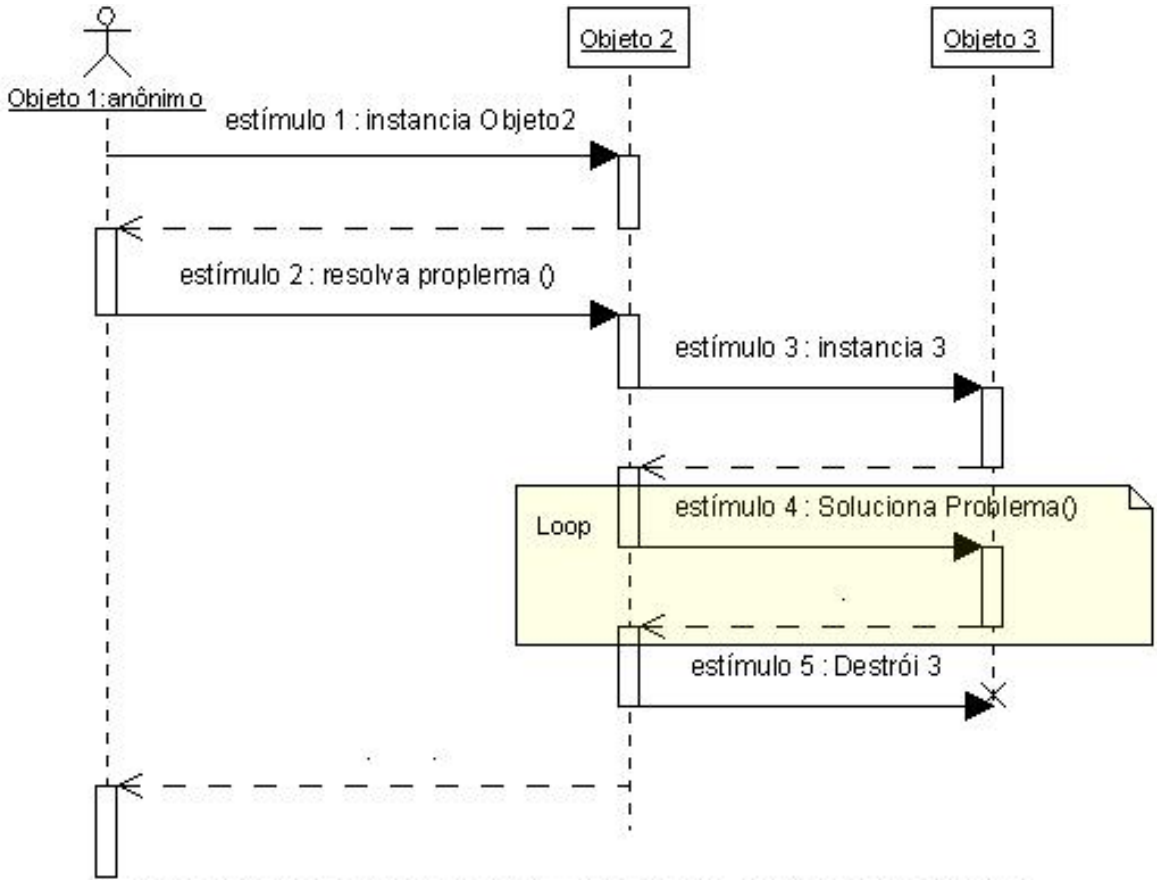


FIGURA 3.7: Diagrama de Seqüência

Os Focos de Controle ou Ativação representam os momentos em que um objeto participou ativamente do processo. São representados por linhas grossas ou barras. No exemplo da Figura 3.7, o Objeto 3 está ativo entre os momentos em que o Objeto 2 aciona o método ‘Soluciona Problema ()’ e o tempo que 3 leva para retornar a solução. As setas de linhas cheias, chamadas de estímulos, são usadas para demonstrar a ocorrência de eventos. Os eventos são, em geral, chamada de métodos, como o ‘estímulo 4’. A resposta a um estímulo também é um estímulo, mas é indicado com uma linha tracejada. O ‘estímulo 5’ ocasiona a destruição de um objeto. Nesses casos, a linha de vida dos objetos é interrompida.

Caso seja necessário, é possível se inserir comentários no diagrama. Na Figura 3.7, um comentário foi utilizado para indicar uma repetição (*loop*).

3.4 A Escolha da Linguagem JAVA

JAVA foi desenvolvida por pesquisadores da *SUN Microsystems* no início da década de 1990 como parte de um projeto que visava desenvolver ‘equipamentos inteligentes’. “Como JAVA foi projetada para dispositivos eletrônicos incorporados, e não para PCs de ponta, ela tinha que ser pequena, eficiente e facilmente portável para uma ampla variedade de dispositivos de hardware” (Lemay, L. e Cadenhead, R.; 2000).

JAVA não vinha obtendo sucesso como linguagem para dispositivos, mas o advento da Internet alavancou a linguagem, pois as qualidades necessárias como linguagem de dispositivos – tamanho, segurança e portabilidade – também eram adequadas para a rede mundial de computadores. Os navegadores passaram a baixar, junto com as páginas em HTML, pequenos aplicativos JAVA para suportar animações, tabelas, efeitos interativos, etc.

Embora o interesse pela linguagem tenha surgido da sua aplicação para a Internet, o desenvolvimento da linguagem seguiu no sentido de se torná-la uma linguagem de uso geral totalmente orientada a objetos.

A linguagem JAVA tem como premissa ser neutra em relação aos programas com os quais interage. Sistemas operacionais diferentes podem fazer, e geralmente o fazem, que o código compilado seja específico para aquele sistema operacional. Para contornar esse problema, JAVA criou uma máquina virtual. Essa máquina virtual permite que se compile um programa em uma linguagem específica e única para máquinas virtuais JAVA. Quando o programa for chamado, a máquina virtual daquela plataforma irá interpretá-lo para o sistema operacional daquele dispositivo. Esse passo intermediário envolvendo a máquina virtual JAVA é o principal motivo para sua eficiência ser menor do que outras linguagens como C++ ou Visual Basic.

O desempenho é um aspecto crítico para vários problemas. Mas existe um grande esforço dos desenvolvedores JAVA para aumentar sua eficiência.

A *SUN Microsystems* montou uma estratégia bastante atraente para a popularização de JAVA: a linguagem é de livre distribuição e uso. Esse fato, aliado ao caráter neutro da linguagem, torna JAVA especialmente interessante para o desenvolvimento de softwares livres. Velloso

(2005) mostra que JAVA já é a segunda mais usada para o desenvolvimento de softwares livres, e a tendência é que em breve se torne a primeira.

JAVA possui um poderoso recurso não existente em outras linguagens orientadas a objetos. Durante a implementação, existe uma sintaxe para comentários de um tipo especial destinados a documentar o código. Posteriormente, ferramentas disponíveis na maioria dos ambientes de desenvolvimento JAVA geram a documentação do código em formato HTML (*HyperText Markup Language*) automaticamente, baseadas na estrutura do programa e nos comentários para documentação. Essas ferramentas geram a documentação num formato padrão que inclui índice, lista de campos e métodos das classes, lista de heranças, bem como informações sobre autoria e versões das classes.

Esses aspectos tornam JAVA uma escolha natural para o desenvolvimento do projeto INSANE.

3.5 Recursos Tecnológicos Auxiliares

Como o INSANE é um projeto de software, e não uma mera aplicação, a escolha de recursos computacionais adicionais é uma parte conceitualmente relevante. Duas ferramentas computacionais importantes para o projeto são o controlador de versão CVS[1] e a interface de programação Eclipse.

A interdependência entre as partes do trabalho de vários pesquisadores dentro do projeto INSANE torna o controle de versão uma tarefa difícil sem alguma ferramenta automática. O CVS (*Concurrent Version System*) é um programa aberto para controle de versão de documentos eletrônicos. Esse programa cria um banco de dados (repositório) que contém a versão corrente dos arquivos digitais, no caso particular de um software, o código fonte do programa. O sistema então controla o acesso dos usuários ao código fonte. Cada usuário, dentro dos privilégios controlados por um administrador, baixa as partes do código fonte que

[1] A expansão para o tratamento de problemas de dinâmica no INSANE ainda não foi migrada para o ambiente CVS.

lhe são permitidas. O programador faz alterações no código e as submete ao administrador que incorpora ou não as modificações propostas, atualizando o repositório.

O sistema também controla as versões sendo revisadas. Se dois programadores estão trabalhando na mesma parte do código, quando a modificação de um for aceita, o outro usuário é notificado, permitindo que esse programador atualize o seu repositório.

O Eclipse é um software livre "projetado para construção de Ambientes de Desenvolvimento Integrados[1] que podem ser utilizados para aplicações tão diversas quanto *sites* na Internet, aplicativos JAVA ou programas em C++" (IBM *Corporation*, 2001).

A introdução das linguagens simbólicas facilitou a tarefa da programação, mas isso não quer dizer que a sintaxe das linguagens simbólicas seja totalmente amigável. As linguagens simbólicas não eliminam totalmente as barreiras entre o usuário, aquele que detém a lógica do problema, e a programação em uma linguagem tipada. A utilização de ambientes gráficos de programação e compiladores interativos facilita em muito o processo de programação. No caso particular da POO, a plataforma Eclipse integra os ambientes de edição, de processamento e de *debug*. O *debug* interativo Eclipse é bastante sofisticado, oferecendo recursos como ajuda automática, organização automática da importação de classes, conserto automático de erros sintáticos simples, etc. A expansão do INSANE para o tratamento de problemas de dinâmica de estruturas apresentada neste trabalho utilizou a plataforma Eclipse.

3.6 A Persistência de Dados

Persistir dados consiste em gravar as informações de maneira que elas existam fora do programa que a gerou. Os arquivos eletrônicos são a forma básica para persistência de dados. O projeto INSANE possui uma estrutura similar a muitos aplicativos de engenharia na qual os dados que descrevem o problema a ser resolvido estão persistidos em algum documento. Esse documento é lido de modo que as informações do problema sejam disponibilizadas para a

[1] Ambiente de Desenvolvimento Integrado, ou *Integrated Development Environment* – IDE do original em inglês, consiste de uma estrutura onde são organizados os elementos de um projeto computacional. Um grande *site* na Internet, por exemplo, necessita de um ambiente onde possa se organizar as páginas, os aplicativos e o conteúdo disponibilizado, isso pode ser feito numa IDE.

parte do programa responsável pela sua solução. Uma vez obtida a solução, essa é persistida em outro arquivo eletrônico.

A forma como os arquivos são persistidos dependem das interfaces de leitura e gravação, e não do problema em si. Interfaces automáticas de leitura e gravação precisam de uma forma tipada para conseguir interpretar os dados. Esse fato gera um grande problema: interfaces diferentes utilizam tipagens diferentes, o que torna os dados não intercanbiáveis. A adoção de uma tipagem padrão para organização da informação também não é uma solução simples. Uma dada tipagem pode ser adequada para um tipo de informação e ser de difícil utilização em outra. Como fazer, por exemplo, uma informação que ao mesmo tempo seja de fácil leitura por um sistema automático e pelo ser humano?

Esse problema mobiliza um grande esforço por parte da comunidade de desenvolvedores. A GML (*General Markup Language*) em 1969, e posteriormente a SGML (*Standart General Markup Language*) em 1980, são tentativas de se criar uma linguagem universal para o arquivamento eletrônico de informação. A GML e a SGML criaram um novo paradigma no qual a informação e a forma de apresentação da informação são separadas. O conteúdo da informação possui uma marcação padronizada que define o seu contexto. A formatação da informação é definida externamente á informação através de um documento específico de formatação da informação. Essa estrutura é bastante conveniente: para se aplicar uma outra apresentação à informação, não é necessário trabalhar a informação em si, e sim o documento que define a formatação. Isso atendia tanto a necessidade de se dar uma formatação amigável para o homem, como também à necessidade dos sistemas computacionais que só são capazes de ler textos numa formatação específica.

A HTML é uma derivação mais simples da SGML que não possui um documento de formatação associado. Os estilos a serem aplicados fazem parte da marcação do documento. A XML (*eXtensible Markup Language*) é uma evolução a partir da HTML que vem se tornando bastante popular para transmissão de dados na Internet. A XML retoma a idéia original da SGML de separar a informação da sua formatação.

Devido ao caráter aberto do projeto INSANE, a comunicação de informações é um aspecto bastante significativo. Os arquivos que definem o problema a ser resolvido, arquivo de pré-processamento, e o arquivo com as soluções, arquivo de pós-processamento, podem ser gerados e lidos por sistemas diferentes. Assim, é necessário um arquivo que possa ser lido e gerado por diferentes sistemas computacionais e cuja leitura direta pelo homem seja fácil.

Além disso, uma vez que a Internet atualmente é um dos principais veículos para a transmissão de informação, uma boa portabilidade para a Internet é desejável. A XML reúne todas essas características.

Capítulo 4

PROJETO DE EXPANSÃO PARA PROBLEMAS DE DINÂMICA

4.1 Classes de Alto Nível para Solução Dinâmica Através do INSANE

No nível mais alto possível, a resolução de um problema de dinâmica de estruturas pelo INSANE consiste em quatro fases distintas:

i. Ler um problema de dinâmica de estruturas discretizado pelo MEF através de um arquivo XML persistido do problema;

ii. Montar a equação discreta do movimento (2.10) repetida abaixo:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{F_d} + \mathbf{K}\mathbf{u} = \mathbf{F_e} \tag{2.10}$$

iii. Solucionar a equação discreta do movimento através de algum método apropriado;

iv. Persistir a solução num arquivo XML.

Assim, identificam-se as classes mais genéricas e abstratas do programa orientado a objetos que aqui se projeta. A classe principal do aplicativo, *Main*, é responsável pela entrada e saída de dados. Ela endereça as informações persistidas no arquivo XML de entrada para as classes pertinentes. As informações do modelo de elementos finitos são armazenadas na classe *FemModel*.

Uma vez construído o modelo de elementos finitos na classe *FemModel*, através de uma classe controladora, a classe *Driver*, a Eq. (2.10) é montada e passada à classe responsável pela solução do problema, a classe *Solution*. Uma vez obtida a solução, a classe *Driver* endereça a solução obtida para os elementos pertinentes da classe *FemModel*. Terminado o processamento, o programa retorna à classe *Main* que persiste a solução do problema no arquivo XML de solução.

As três classes aqui projetadas, uma vez que a classe *Main* é uma imposição da linguagem, necessitam de classes mais especializadas para desempenhar a tarefa de solução de um problema de dinâmica de estruturas. Um método natural para se identificar várias dessas classes, é percorrer um fluxo para solução de um problema típico.

4.2 Determinação de Classes a Partir de um Caso Típico

O roteiro de solução de um problema estrutural pelo MEF a seguir é uma adaptação dos roteiros propostos por Logan (1992), Bathe (1982) e Soriano e Lima (1999). Esse roteiro assume que o problema de meio contínuo já foi identificado.

4.2.1 Construção do Modelo Físico-Matemático Simplificado

O primeiro passo para a modelagem do problema é descrever sua geometria e os fenômenos físicos envolvidos. É impossível para o analista captar todos os aspectos da geometria de um problema real, bem como conceber um problema físico generalizado. O primeiro passo, então, é determinar os aspectos geométricos e os fenômenos físicos relevantes. A partir dessa representação simplificada, é possível fazer hipóteses com relação ao comportamento cinemático da estrutura. Se uma estrutura será modelada como uma estrutura em três, duas ou uma única dimensão, depende de como se supõe que a estrutura irá se deslocar. A dimensão

do problema e as leis cinemáticas da formulação adotada necessitam de uma classe específica para representá-las, a classe *AnalysisModel*.

Para finalizar a construção do modelo físico-matemático, resta definir o comportamento dos materiais. Homogeneidade, variação direcional de propriedades, propriedades mecânicas, entre outras. Identifica-se aí a classe *Material*.

4.2.2 Construção do Modelo de Elementos Finitos a Partir do Modelo Físico-Matemático

É preciso escolher um elemento capaz de representar o comportamento esperado pelo modelo. A classe *Element* contém os elementos finitos da análise. Um elemento encapsula quais são os nós que o compõem, as leis de formação das matrizes e vetores locais e as propriedades pertinentes ao elemento, material e carregamentos sobre o elemento.

Uma vez escolhido o elemento, é possível definir a geometria da malha de elementos finitos. A malha é constituída pelos elementos e seus nós, donde se detecta a classe *Node*. A classe *Node* encapsula as informações de suas coordenadas e as grandezas nodais: deslocamentos, massa, amortecimento, rigidez, carregamento e demais condições de contorno.

Os carregamentos nodais são representados por uma classe específica, a classe *PointForce*, consistindo basicamente dos módulos dos carregamento nodais em cada direção (x, y, ,z, θ_x, θ_y, θ_z). Cada módulo do carregamento pode variar no tempo. Por definição de projeto, os carregamentos e condições de contorno variáveis no tempo são descritos por funções analíticas escalares. Essas funções são armazenadas na classe *ScalarFunction*.

Também por definição de projeto, os carregamentos distribuídos no INSANE são representados por carregamentos nodais cuja distribuição ao longo do elemento é feita através das funções de forma do elemento, isto é:

$$\mathbf{F}(x, y, z) = \sum_i \mathbf{F_i} N_i \tag{4.1}$$

onde **F** é o vetor força num ponto qualquer do elemento, $\mathbf{F_i}$ é o vetor força no nó '*i*' do elemento e N_i é a função de forma associada ao nó '*i*'. A classe *ElementForce* armazena o caso do carregamento (linear, superfície ou volume) e os carregamentos nodais correspondentes ao caso.

A Eq. (4.1) apresenta pela primeira vez ao longo do encadeamento de um problema de dinâmica de estruturas as funções de forma da análise via MEF. As funções de forma são armazenadas na classe *Shape*.

Os objetos das classes *Element* e *Node* são armazenados na classe *FemModel*.

4.2.3 Obtenção das Matrizes e Vetores do Sistema

Uma vez construído o modelo discreto do problema, deve-se montar a representação matemática desse modelo, ou seja, deve-se montar a Eq. (2.22) aqui repetida:

$$\mathbf{M\ddot{u} + A\dot{u} + K\ddot{u} = F_{eq}} \tag{2.22}$$

As matrizes globais são montadas pela classe *Driver* seguindo o fluxo mostrado na Figura 4.1, onde se nota que, primeiramente, deve se obter as matrizes locais de cada elemento. A classe *Element* encapsula os métodos para geração das suas matrizes locais. As matrizes locais são obtidas através da quadratura de Gauss, onde é utilizada a classe *IntegrationPoint*. Essa classe armazena as coordenadas dos pontos de Gauss e os seus respectivos pesos de integração. Por definição de projeto, optou-se por manter a ordem de integração encapsulada em outra classe, a *IntegrationOrder*, que apenas armazena a ordem de integração em cada direção do espaço.

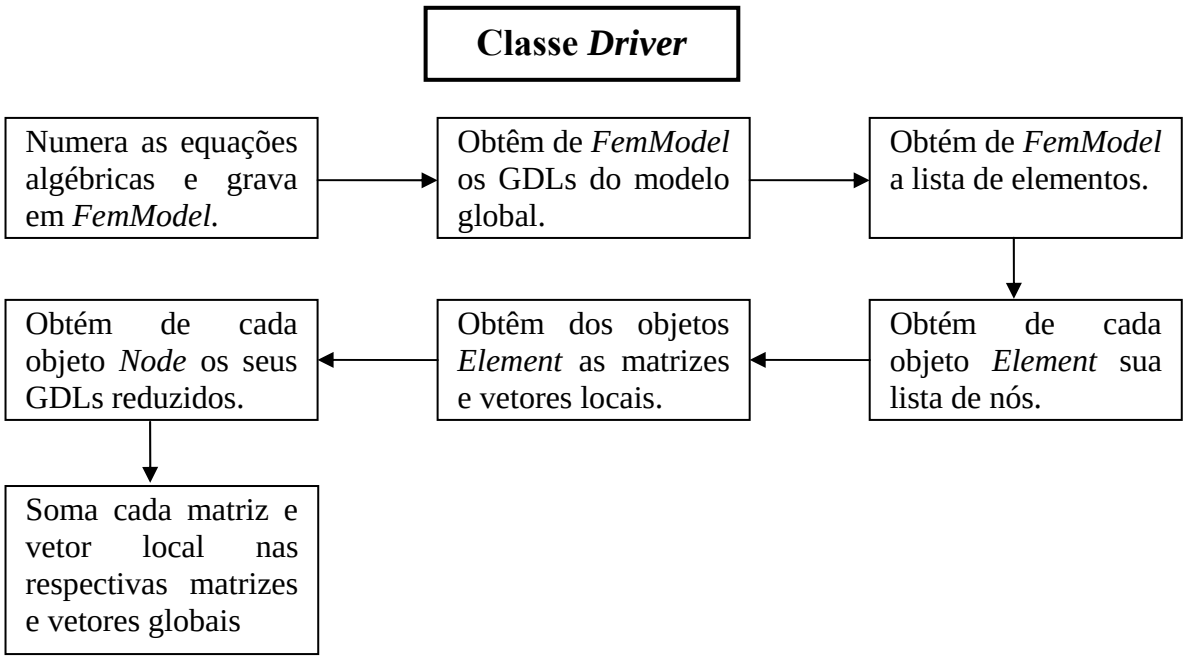


FIGURA 4.1: Processo de montagem das equação do movimento dentro da classe *Driver*

A principal vantagem de se trabalhar com elementos paramétricos é a generalização que eles possibilitam para obtenção de matrizes e vetores locais, permitindo que os métodos de integração de todos os elementos paramétricos sejam encapsulados em uma única classe, a *ParametricIntegration*.

4.2.4 Solução da Equação do Movimento

A classe *Driver* passa as matrizes e vetores necessários para a classe *Solution* realizar o processo de integração no tempo. Por definição de projeto, a classe *Solution* recebe a informação para realizar um passo de integração no tempo, após o quê, a classe *Driver* persiste todo o objeto *FemModel*. As matrizes de massa, amortecimento e rigidez são passadas à classe *Solution* apenas no primeiro passo de integração, enquanto o vetor de tempo e o vetor de carregamentos reduzidos são atualizados pela classe *Driver* a cada passo da integração no tempo.

No caso de soluções por decomposição modal, a classe *Solution* resolve, primeiramente, o problema de autovalor associado. O problema de autovalor não é uma particularidade de sistemas dinâmicos, sendo preferível definir uma classe *IModal*[1] para encapsular a solução modal fora da classe *Solution*. A classe *Solution* passa a matriz de massa e rigidez para o objeto *IModal*, enquanto o número de modos a ser obtido é informado pela classe *Main* ao ler o modelo persistido de entrada.

4.3 Demais Classes para Análise Dinâmica de Estruturas

No processo indutivo das seções 4.1 e 4.2, não foi possível identificar algumas classes devido à justificativa para as mesmas ser mais ligada ao processo computacional do que ao método de análise. Um exemplo disso é a classe *IPoint3d*, que encapsula as coordenadas de um ponto. Como ao longo da análise dinâmica, várias classes utilizam coordenadas, é preferível , então, que uma classe dedicada encapsule essa informação para permitir a padronizando na implementação de campos e métodos nas classes que utilizam coordenadas.

---

1 O prefixo ‘I’ deriva de INSANE.

O projeto INSANE desenvolveu suas próprias classes para manipulação de vetores e matrizes: as classes *IVector* e *IMatrix* respectivamente[1].

Uma classe bastante específica para a análise dinâmica é a *ResultsWriter*, a qual, por sua vez, implementa uma interface desenvolvida para o problema dinâmico, a *ResultsFile*. A classe *ResultsWriter* encapsula os métodos para persistir objetos ao longo da integração no tempo.

Por fim, criou-se a classe *CrossSection* para encapsular as propriedades geométricas das seções transversais de elementos unidimensionais e bidimensionais.

4.4 Heranças para Análise Dinâmica de Estruturas

Algumas das classes definidas anteriormente são, ainda, bastante gerais, de modo que se optou por criar classes mais especializadas através do mecanismo de herança.

4.4.1 Heranças da Classe *DiscreteModel*

O primeiro passo identificado para a modelagem do problema é a obtenção do modelo físico-matemático. A modelagem via MEF é um método de discretização adequado para certos problemas. Outras técnicas, como método dos elementos de contorno, são mais adequadas para outros casos. Mas os métodos de discretização do domínio não são a primeira abordagem num processo de análise. Em alguns problemas como os de mecânica do contínuo, pode-se optar por solucionar o problema através de funções aproximadoras quaisquer, sem utilizar uma técnica de discretização do domínio.

O diagrama de classes da Figura 4.2 representa o estágio atual do INSANE. A classe *BemModel*, para análises utilizando elementos de contorno, ainda não foi desenvolvida, mas já foi incluída no projeto prevendo uma futura expansão.

---

[1] JAVA possui uma API (Application Programming Interface) para esse tipo de objeto, a GMatrix. Optou-se por não utilizar essa classe por dois motivos: essa API utiliza muitos pacotes que não são necessários ao INSANE e foi encontrado um erro nesse aplicativo que permitia a inversão de matrizes singulares.

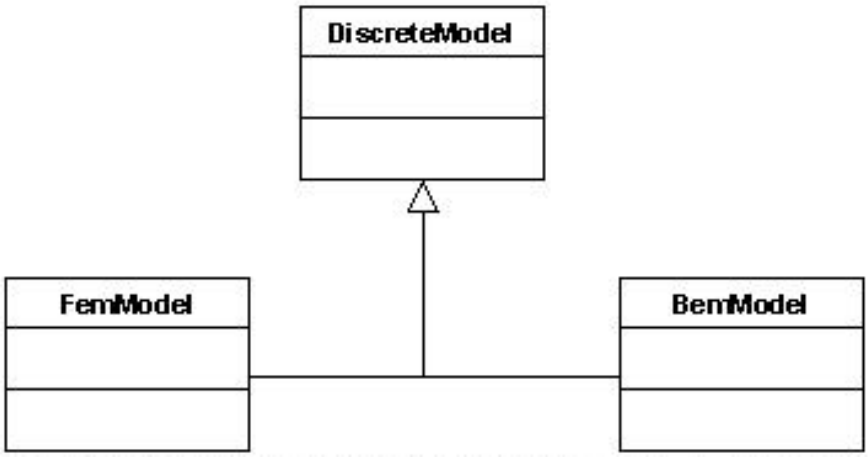


FIGURA 4.2: Heranças da classe *DiscreteModel*

4.4.2 Heranças da Classe *Material*

A classe *Material* possui quatro subclasses diretas: *Anisotropic*, *Isotropic*, *Orthotropic* e *MSpring*. A classe *MSpring* tem o objetivo de encapsular as propriedades de um suporte flexível. O propósito das demais classes fica evidente a partir de seus nomes.

A Figura 4.3 apresenta o diagrama de herança da classe *Material*, mas apenas os campos principais da classe *Isotropic* estão listados, visto que, nos exemplos para validação deste trabalho, apenas materiais isotrópicos são considerados. Na classe *Material*, quatro campos foram destacados cujo significado fica evidente pelos seus nomes à exceção do campo *type*, o qual armazena a que classe o material pertence. Na superclasse *Material* foram alocados os campos *density*, *rayleighFactorA* e *rayleighFactorB*. Não há dúvida que o primeiro desses campos deve ser alocado na superclasse, pois a densidade é uma propriedade escalar. Mas com relação aos coeficientes de *Rayleigh*, pode haver dúvidas. O amortecimento pode ser uma propriedade direcional. Porém, o amortecimento de *Rayleigh* é mais uma simplificação matematicamente conveniente do que uma tentativa de representar com exatidão o amortecimento. A possibilidade de se ter os fatores de *Rayleigh* variando com a direção teria pouca utilidade. Por fim, devido à lei de formação da matriz de amortecimento de *Rayleigh*, ela acompanhará a variação direcional da matriz de rigidez.

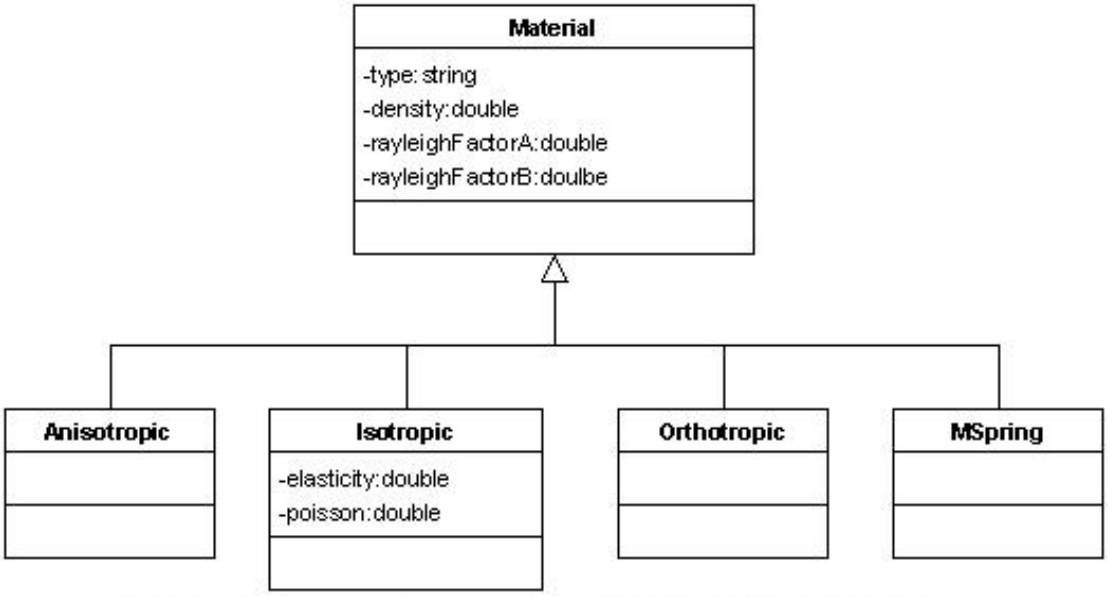


FIGURA 4.3: Diagrama de heranças da classe *Material*

4.4.3 Heranças da Classe *Element*

A Figura 4.4 mostra um diagrama de heranças incompleto da classe *Element*. Pode-se observar que a classe *ParametricElement* disponibiliza para as classes herdeiras os métodos para a determinação das matrizes e vetores do elemento. Cada elemento necessita de uma classe para encapsular as suas particularidades. A classe *FrameElement*, uma especialização para estruturas reticulares, não é utilizada nesse projeto.

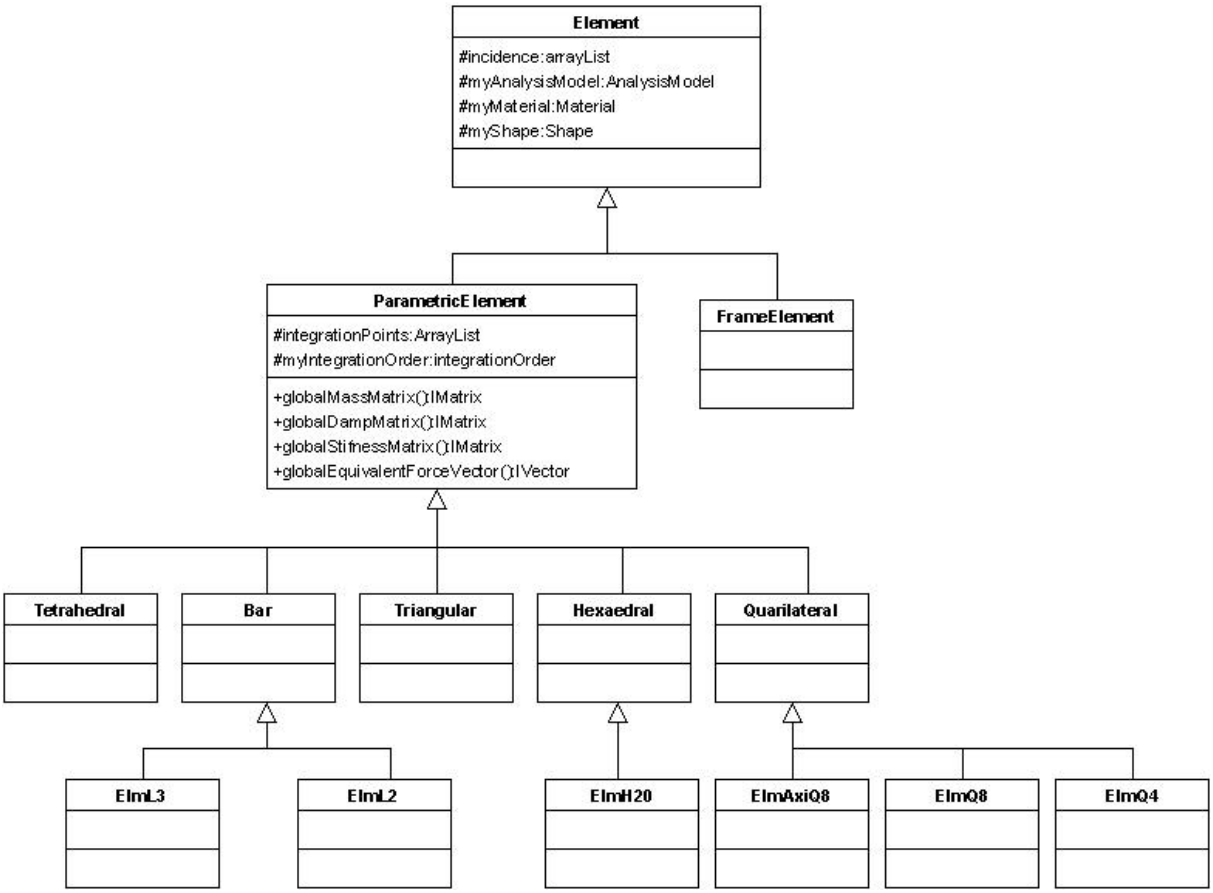


FIGURA 4.4: Diagrama de Heranças da classe *Element*

O diagrama de heranças da classe *Shape* é bastante similar ao da classe *Element*, com a diferença que as funções de forma para os elementos não têm influência do tipo de análise. Assim, não é necessário ter uma subclasse de *Shape* para cada elemento axissimétrico. Essa mesmo observação vale para a classe *ParametricIntegration*.

4.4.4 Heranças da Classe *IntegrationPoint*

A classe *IntegrationPoint* possui três heranças conforme mostrado na Figura 4.5, todas elas utilizando esquemas de quadratura de Gauss. A caracterização de cada uma das três subclasses está relacionada ao tipo de coordenada usada para representar os pontos de integração: linear, de área ou volume.

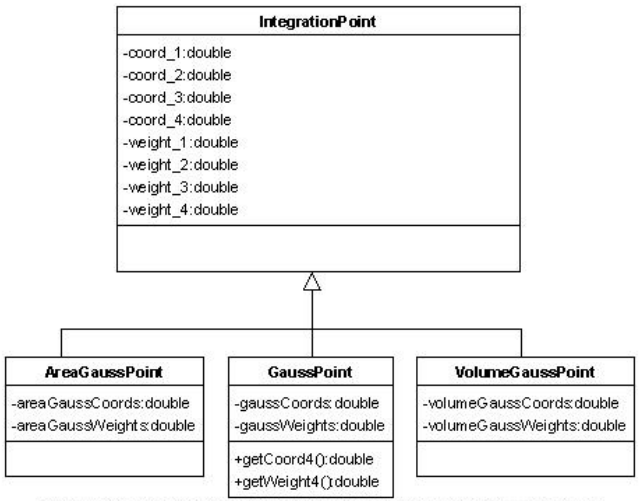


FIGURA 4.5: Diagrama de heranças para a classe *IntegrationPoint*

4.4.5 Heranças da Classe *Driver*

O diagrama de heranças da classe *Driver* indica que futuras expansões para outros problemas são esperadas, de modo que a classe é bem genérica. No estágio atual do INSANE, apenas as aplicações de dinâmica de estruturas foram incorporadas. A classe *StructuralMech* encapsula toda a implementação necessária para um problema estático. A subclasse *DynamicStructuralMech* estende a classe *StructuralMech* de modo a incluir a implementação básica de um problema de dinâmica:

i. Gera as matrizes de massa e amortecimento.

ii. Determina o vetor de tempo[1] para cada passo de integração no tempo.

iii. Gera os vetores de carregamento para cada passo de integração.

iv. Faz o encadeamento da integração numérica no tempo:

Como se observa na Figura 4.6, a finalidade das subclasses da *DynamicStructuralMech* é implementar uma forma polimórfica do método *AnalyseModel*. Cada uma das subclasses para análise dinâmica instancia um tipo de solução dinâmica. As formas polimórficas do método *AnalyseModel* passas as informações e chamas os métodos relativos ao seu objeto *Solution* específico.

[1] As coordenadas de tempo a serem utilizadas na integração de Gauss são armazenadas em um vetor.

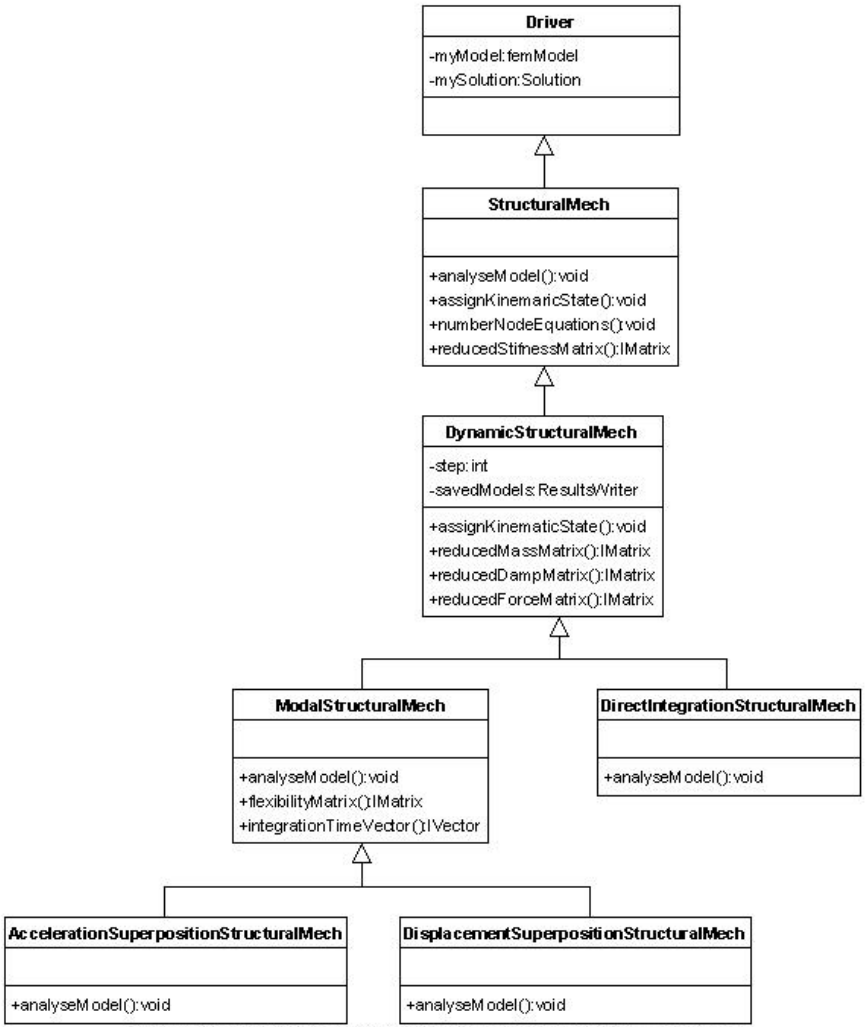


FIGURA 4.6: Diagrama de heranças da classe *Driver*

Para permitir a persistência do objeto FemModel a cada passo da integração numérica, a classe *DynamicStructuralMech* disponibiliza o campo *savedModels* para suas subclasses.

4.4.6 Heranças da Classe *Solution*

De maneira análoga à observada para a classe *Driver*, a superclasse *Solution* (Figura 4.7) é bastante genérica, não implementando nenhum método de solução sequer, ficando isso a cargo das subclasses mais especializadas. A sua única subclasse, porém, é uma classe vazia. Os formatos da classe *Solution* e da classe *Equilibrium* evidenciam as futuras expansões planejadas. A subclasse *OnePointEq* encapsula a solução estática, enquanto as duas outras subclasses implementam a solução dinâmica. O núcleo da solução é o método *kinematicSolver*. No caso das soluções modais, a classe *ModalEq* encapsula a solução do problema de autovalor associado que é feita por sua implementação do método desse método.

A superclasse *ModalEq* também encapsula os métodos de integração que são comuns as sua subclasses *AccelerationSuperposition* e *DisplacementSuperposition*.

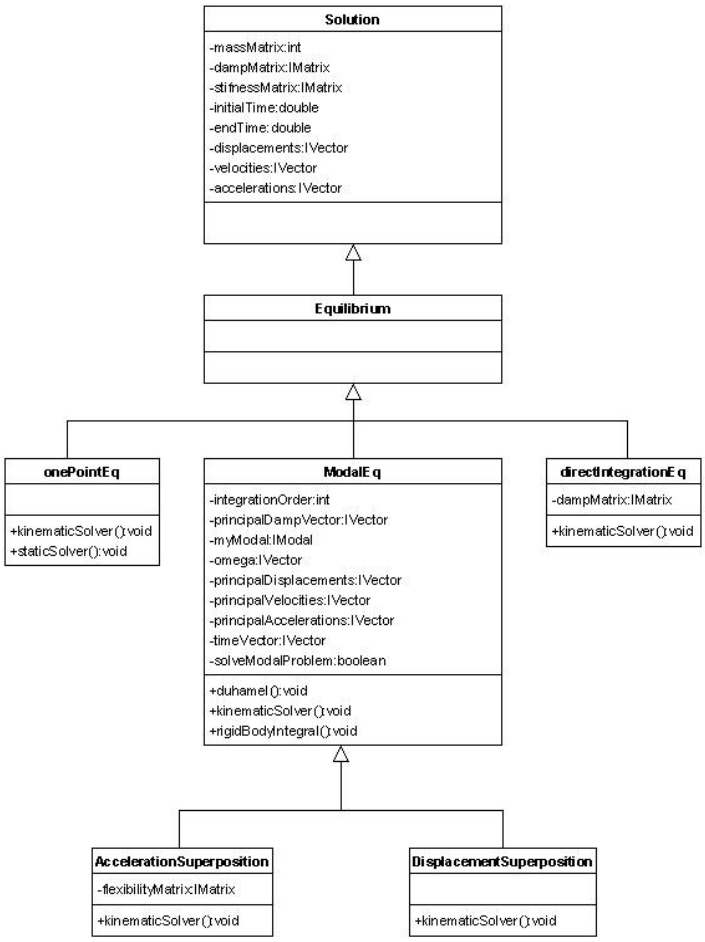


FIGURA 4.7: Diagrama de heranças da classe *Solution*

4.4.7 Heranças da Classe *IModal*

As subclasses de *IModal*, Figura 4.8, as classes *IIIModal* (*INSANE Inverse Iteration Modal*) e *ISIModal* (*INSANE Subspace Iteration Modal*) implementam os solucionadores modais da iteração inversa e da iteração no subespaço respectivamente. Vale destacar o campo *myIIIModal* na classe *ISIModal*, evidenciando que o problema de autovalor no subespaço será resolvido pelo método da iteração inversa.

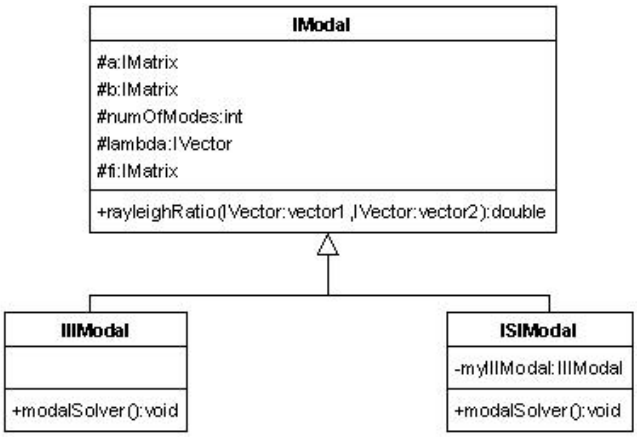


FIGURA 4.8: Diagrama de heranças da classe *IModal*

4.4.8 Heranças da Classe *ScalarFunction*

As subclasses da classe *ScalarFunction*, Figura 4.9, mostram os casos de carregamento que podem ser representados, sendo que cada caso está implementado numa classe. A classe *CompositeFunction* permite compor um carregamento a partir dos casos de carregamento básicos. O método *getValue (time)* retorna o valor escalar da função para o tempo passado como argumento.

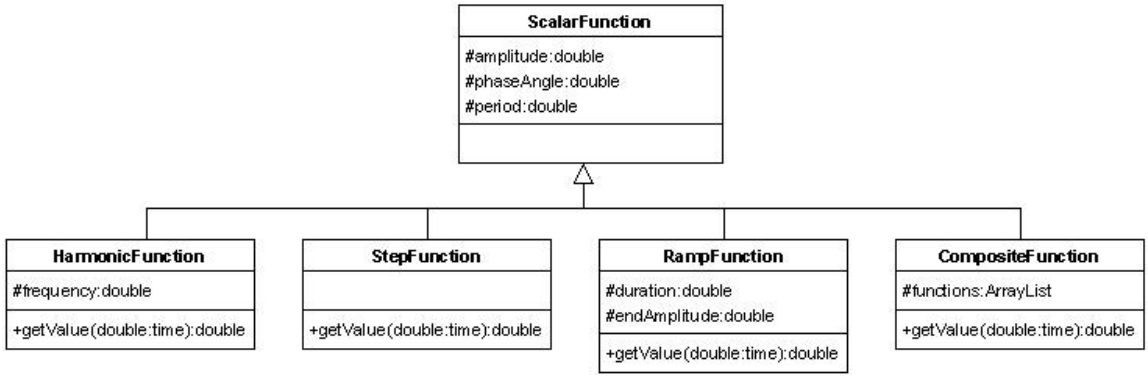


FIGURA 4.9: Diagrama de heranças para a classe *ScalarFunction*

4.5 Associação de Classes para Obtenção da Solução

4.5.1 Associações da Classe *Element* e *PointForce*

A Figura 4.10 mostra algumas associações da classe *Element*, enquanto a Figura 4.11 faz o mesmo para a classe *PointForce*. Algumas relações de multiplicidade da classe *PointForce* não são óbvias. A Tabela 4.1 explica de maneira sucinta essas associações.

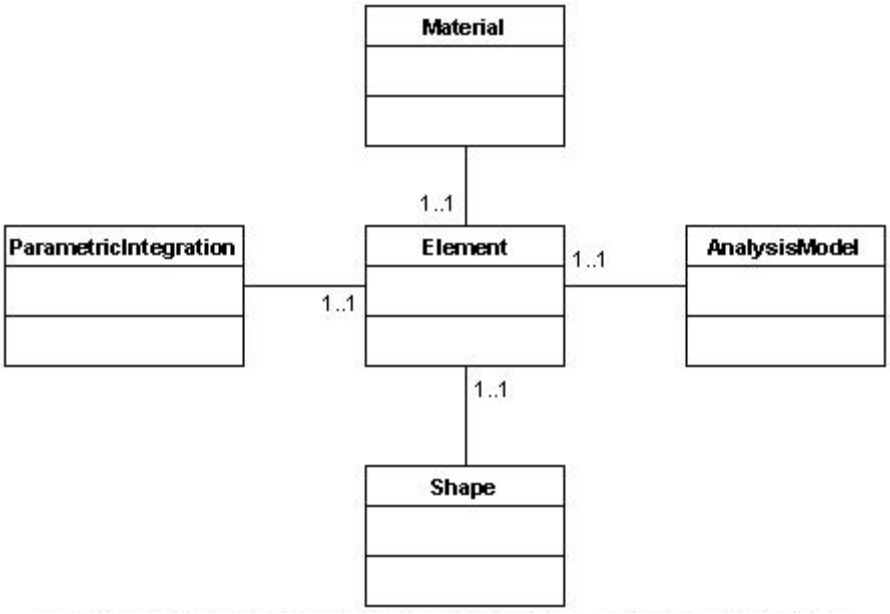


FIGURA 4.10: Diagrama de classes mostrando algumas associações da classe *Element*

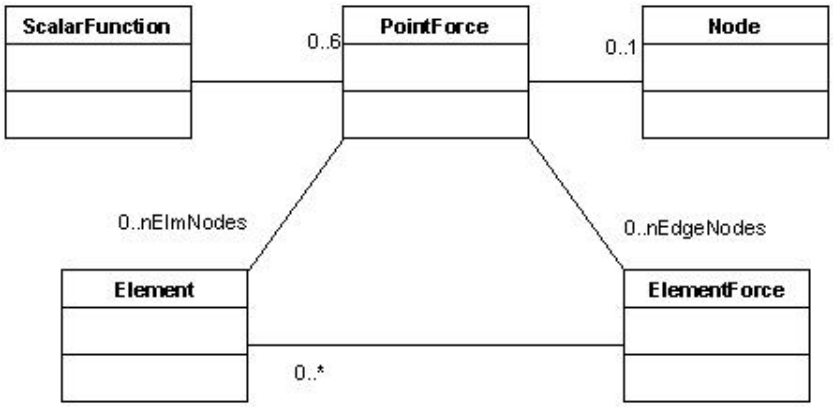


FIGURA 4.11: Diagrama de classes mostrando algumas associações da classe *PointForce*

TABELA 4.1: Descrição de algumas relações de multiplicidade mostradas na Figura 4.11

CLASSES	DETALHAMENTO
Element e *PointForce*	Cada nó do elemento pode possuir um ou nenhum *PointForce*, daí a relação 0..1.
Element e *ElementForce*	Não existe limite para o número de forças de elemento numa aresta, superfície ou volume, bem como é possível que não haja nenhuma força
PointForce e *ScalarFunction*	Cada módulo de uma instância de *PointForce* pode instanciar um ou nenhum *ScalarFunction*.
PointForce e *ElementForce*	Cada aresta ou superfície do elemento pode ter forças de elemento associadas. Cada nó dessa aresta ou superfície instancia um *PointForce*. Para o caso de forças de volume, todos os nós do elemento instanciam um *PointForce*.

4.5.2 Visão Geral da Associação das Classes do INSANE

A Figura 4.12 mostra associações das classes no INSANE num nível mais amplo. Novamente, algumas multiplicidades não são óbvias, e são resumidamente explicadas na Tabela 4.2.

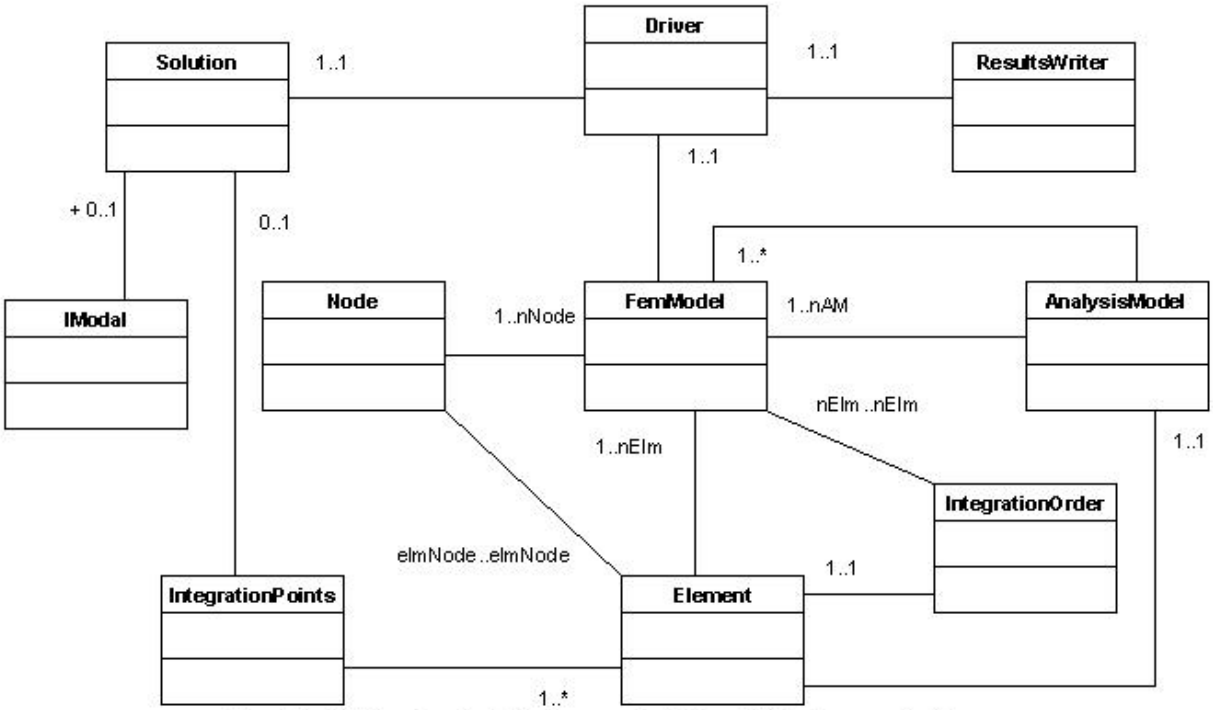


FIGURA 4.12: Visão geral da associação das classes no INSANE

TABELA 4.2: Descrição de algumas associações numa visão geral do INSANE

ELEMENT	
IntegrationPoint	O elemento instanciará tantos objetos *IntegrationPoint* quantos forem necessários para as ordens de integração especificadas para esse elemento em cada uma das direções.
SOLUTION	
IModal	Apenas no caso de solução modal se instanciará um objeto da classe *IModal*.
IntegrationPoint	Apenas a solução modal precisa instanciar um *IntegrationPoint*, pois a solução por integração direta não segue um esquema de quadratura.
FEMMODEL	
AnalysisModel	A classe *FemModel* possui instancias da classe *AnalysisModel* em campos bem distintos. Em um deles, a classe *FemModel* armazena todas as análises existentes no modelo. Em um outro, guarda o modelo de análise global.
IntegrationOrder	A classe *FemModel* armazena as ordens de integração de cada elemento.

4.6 Diagrama de Seqüências para o INSANE

A Figura 4.13 mostra o diagrama de seqüências do INSANE. Para tornar o diagrama mais legível, alguns estímulos agrupam uma série de ações. Essas ações serão, então, detalhadas na Tabela 4.3. A figura mostra o diagrama de seqüência para a integração no tempo feita pela classe *Driver*.

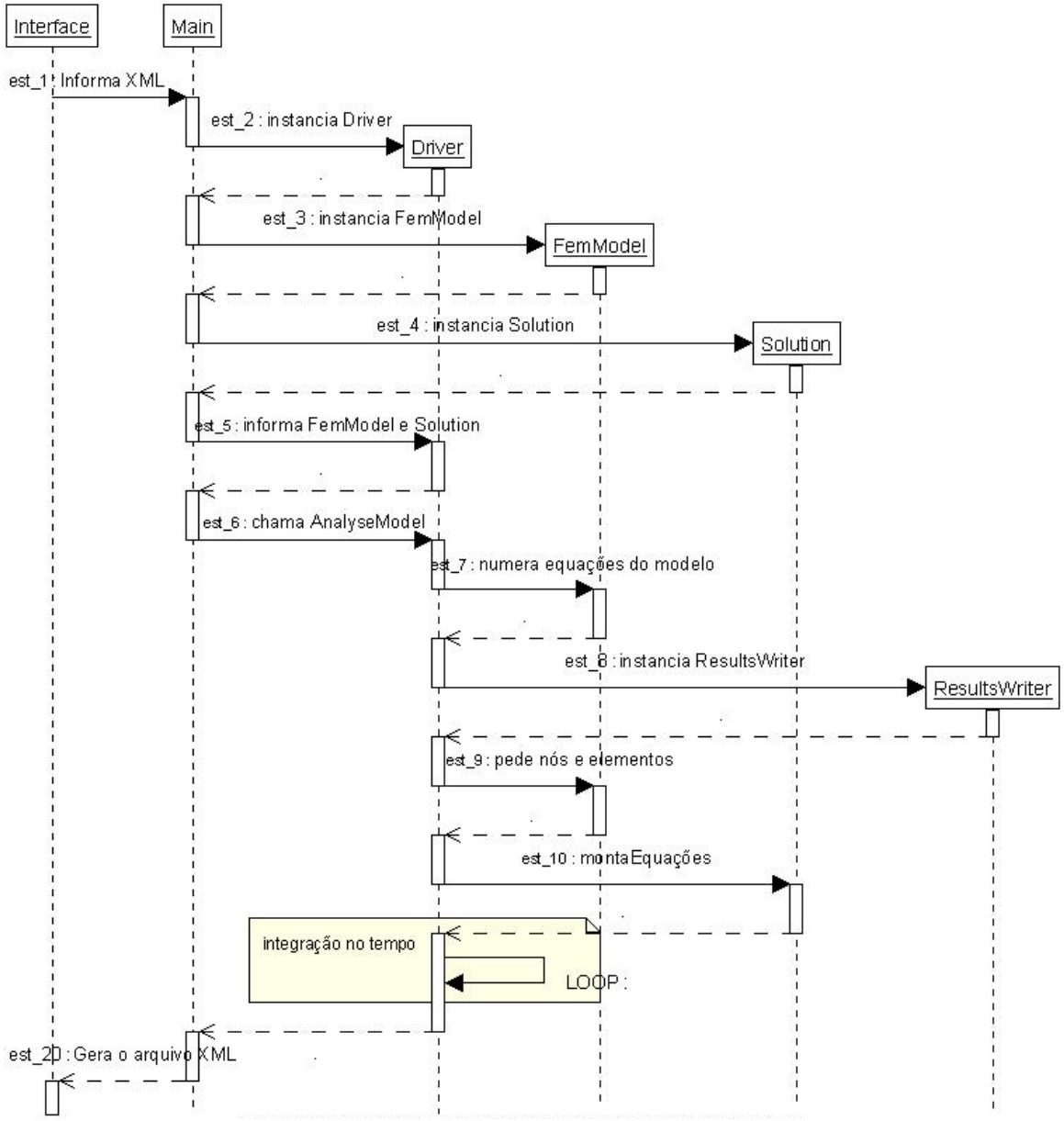


FIGURA 4.13: Diagrama de seqüências do INSANE

TABELA 4.3: Detalhamento de alguns estímulos no diagrama de seqüência do INSANE

ESTÍMULO	DETALHAMENTO DAS AÇÕES
3. *Main* instancia *FemModel*	A classe *Main* instancia *FemModel* e informa as propriedades do modelo que estão persistidas no arquivo XML. Posteriormente, a classe *Main* informa o objeto *FemModel* para a classe *Driver*.
4. *Main* instancia *Solution*	*Main* instancia a classe *Solution* e passa informações a esse objeto que estavam persistidas no arquivo XML. Feito isso, o objeto *Solution* é informado para a classe *Driver*.
20. Gera o arquivo XML da solução	Após *Driver* executar o método *analyseModel*, a classe principal executa várias chamadas às classes *Driver*, *FemModel* e *Solution* para buscar as informações da solução. Além disso, a classe *Main* cria sua própria instância de *ResultsWriter* para ler as instâncias de *FemModel* persistidas no disco rígido para cada passo de tempo.

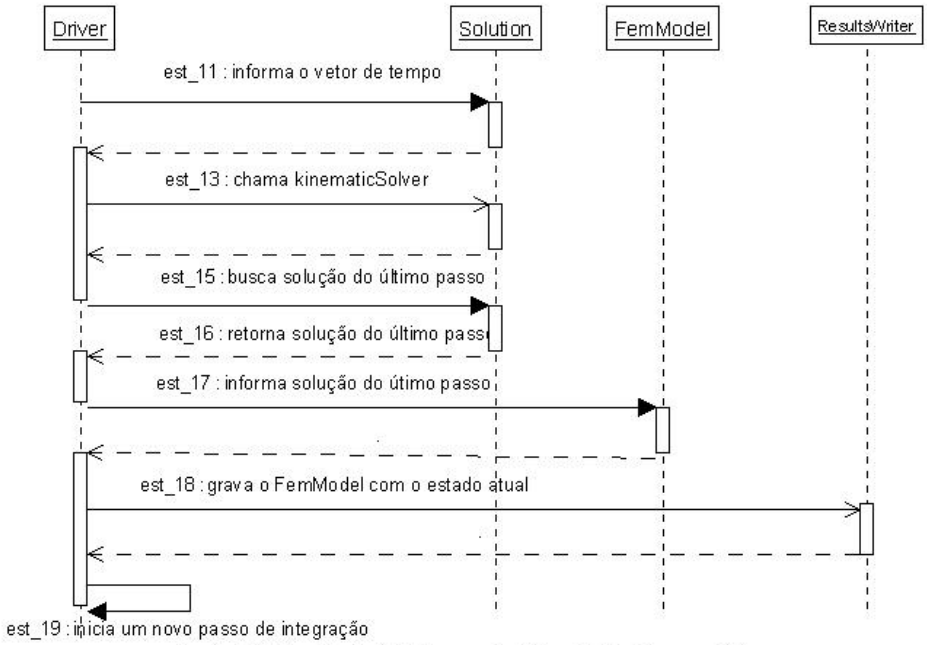


FIGURA 4.14: Diagrama de seqüência para a classe o *Loop* de integração no tempo

4.7 Sumário do Projeto de Expansão

Até o presente momento, não se preocupou em identificar qual trabalho feito nas classes apresentadas, uma vez que a solução de problemas de dinâmica foi montada sobre uma plataforma já funcional para um problema estático. No trabalho de expansão, houve três casos distintos no que se refere às classes. O primeiro foram modificações de pouca monta em algumas das classes existentes para fazer pequenas adaptações ou corrigir erros de implementação identificados. O segundo caso foi uma intervenção mais profunda para representação do caso dinâmico. Por fim, foi necessário criar classes totalmente novas dedicadas especificamente ao caso dinâmico.

O trabalho de expansão do INSANE do caso estático para o dinâmico foi condensado em duas tabelas. A Tabela 4.4 agrupa as classes que se enquadram no segundo caso descrito, ou seja, classes que foram adaptadas para permitirem o tratamento de problemas de dinâmica. Uma breve explicação sobre a modificação introduzida é apresentada. A Tabela 4.5 apresenta as classes que foram criadas especificamente para o tratamento dos casos de dinâmica de estruturas. As classes que pertencem ao primeiro grupo são omitidas.

TABELA 4.4: Classes modificadas durante o trabalho de expansão do INSANE

CLASSE	SUMÁRIO DA MODIFICAÇÃO
Main	A classe *Main* foi bastante afetada, uma vez que ela é a responsável pela entrada e saída dos dados. Todas as grandezas necessárias ao caso dinâmico tiveram que ser incluídas. Além disso, devido ao fato de uma solução dinâmica funcionar como uma série de soluções estáticas torna a forma de persistir os dados no arquivo XML diferente.
Driver	Pequenas adaptações e pequenos reparos. As diferenciações para o caso dinâmico foram feitas através do mecanismo de herança.

CLASSE	SUMÁRIO DA MODIFICAÇÃO
FemModel	Houve a inclusão de novos campos. Embora tenha se demonstrado de pouca utilidade para a expansão feita, criou-se uma subclasse da *FemModel* capaz de suportar casos mais gerais de análise dinâmica não implementados nesse trabalho.
Solution	Houve a inserção de campos nas superclasses e se utilizou também o mecanismo de herança.
Node	Criação de campos para os graus de liberdade de velocidade e aceleração. As adaptações para condições de contorno no tempo demandaram uma análise mais aprofundada. O mesmo ocorreu para os carregamentos nodais.
Element	A adaptação para prever os carregamentos do elemento variáveis no tempo requereu bastante análise. Campos específicos foram criados para suportar esse tipo de carregamento. Esses campos consistem de listas de *PointForces*.
ParametricElement	Foi modificada para a inclusão dos métodos de obtenção das matrizes de massa e amortecimento.
PointForce	For adaptada para permitir o carregamento variável no tempo. A modificação consistiu na inserção de um campo para funções escalares e a criação dos métodos pertinentes.
GaussPoint	Modificada para incluir uma coordenada específica para o eixo do tempo.
Material	Modificada para incluir os campos para a densidade e os coeficientes de Rayleigh.

TABELA 4.5: Classes criadas especificamente para a expansão para o caso de sistemas dinâmicos

CLASSE	SUMÁRIO DA MODIFICAÇÃO
DynamicStructuralMech	Implementação dos métodos para a construção das matrizes e vetores globais para todas as análises dinâmicas. As suas subclasses implementaram os métodos de encadeamento da solução (*AnalyseModel*) para cada tipo de solução dinâmica, incluindo aí um método para persistência das soluções intermediárias utilizando um objeto *ResultsWriter*.
ModalEq	Implementação da integração de *Duhamel* e métodos para suportar cálculos intermediários.
Displacement-SuperpositionEq e *Acceleration-SuperpositionEq*	Implementação dos métodos de superposição modal.
DirectIntegrationEq	Implementação do método *Newmark-*β com a modificação *Hilber-*α*.*
IModal	Implementação da solução de problemas de autovalor.
ScalarFunction	Funções analíticas básicas dos problemas dinâmicos.
ResultsWriter	Classe para gravação de soluções intermediárias.

Capítulo 5

PROBLEMAS SELECIONADOS E ANÁLISE DE RESULTADOS

Os problemas escolhidos tiveram por objetivo permitir a identificação de erros de implementação e demonstrar os recursos implementados. A Tabela 5.1 a seguir apresenta um sumário dos problemas resolvidos utilizando a implementação feita neste trabalho.

TABELA 5.1: Sumário dos problemas resolvidos para a validação do INSANE

PROBLEMA	DESCRIÇÃO	PROPRIEDADE TESTADA
5.1) Sistemas de 1GDL	Simulação de um sistema massa-mola utilizando um único elemento L2	a) Sistemas de 1GDL com massa concentrada. b) Sistemas de 1GDL com massa concentrada e amortecimento. c) Sistemas de 1GDL com massa consistente
5.2) Sistema Linear com nGDL	Simulação de sistemas unidimensionais de nGDL através de 4 elementos L2 e massa concentrada nos nós.	a) Verificação da formação do carregamento para o caso harmônico. b) Validação do integrador modal para um sistema com nGDL. c) Validação da superposição de acelerações modais.

PROBLEMA	DESCRIÇÃO	PROPRIEDADE TESTADA
5.3) Sistema Unidimensional com Elemento L3	Simulação de um sistema unidimensional com um único elemento L3	Formação da matriz de massa consistente para o elemento L3.
5.4) Treliça Plana	Treliça plana de 3 nós e 3 GDL formada por elementos de barra L2.	a) Extensão do problema para o caso plano. b) Verificação da formação do carregamento e da matriz de massa para um caso um pouco mais geral. c) Verificação do integrador modal para um sistema com n-GDL.
5.5) Vibração Transversal de uma Viga com Elementos de Estado Plano	Viga engastada simulada com uma malha de elementos Q4 com massa e carregamento degrau na extremidade livre. Os casos de estado plano de tensão e de deformação foram verificados.	Validação
5.6) Problema de Estado Plano	Um elemento de máquina foi simulado com elementos Q8 em estado plano de deformação.	Validação

Como observação, cabe dizer que todos os exemplos são apresentados sem unidades, uma vez que foram utilizadas unidades compatíveis em suas execuções.

5.1 Solução do Sistema de 1GDL

O primeiro passo para validação da implementação dinâmica do INSANE foi a solução de problemas de 1GDL. Três modelos de 1GDL foram escolhidos para testar diferentes partes do programa.

O primeiro problema de 1GDL consistiu de um sistema massa-mola preso a uma base fixa em uma extremidade e solicitado por um carregamento degrau unitário na extremidade livre, conforme ilustrado na Figura 5.1a. A massa é concentrada na extremidade livre, as condições iniciais são nulas e não existe atrito na base. Esse sistema foi modelado com um único elemento L2, conforme mostra a Figura 5.1b.

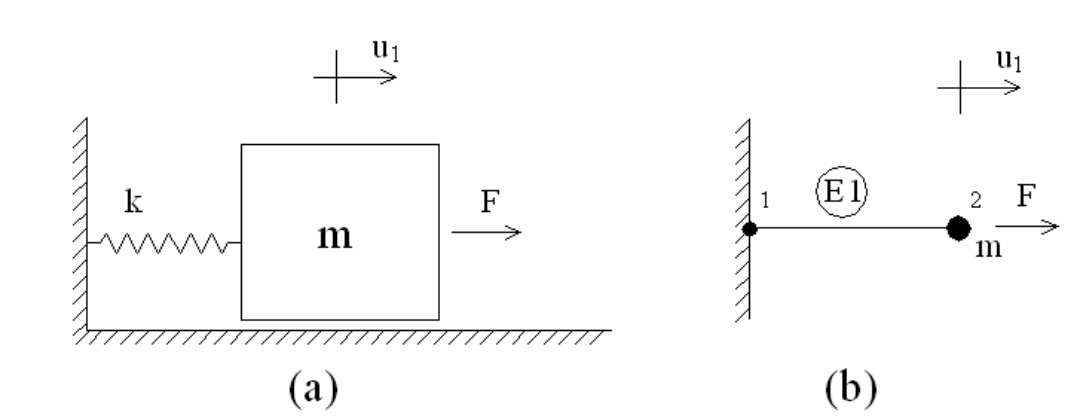


FIGURA 5.1: (a) Sistema massa mola e (b) modelo de elementos finitos do problema

Os dados do problema são:

- F é um degrau unitário
- m = 1,0
- k = 1,0

A solução analítica desse problema para condições iniciais nulas é:

$$u_2(t) = \sum_{i=1}^{3} \frac{\phi_{2,i}}{\omega_i^2}\left(1 - \cos \omega_i t\right) \tag{5.1}$$

sendo ω = 1,0.

O segundo sistema de 1GDL foi um sistema massa-mola-amortecedor que consiste basicamente da adição de amortecimento no problema anterior, conforme ilustrado na Figura 5.2. O diagrama do modelo de elementos finitos para esse problema é exatamente igual ao do caso anterior, mas existe uma diferença importante na forma como o amortecimento é tratado num modelo de solução modal e num modelo de solução por integração direta. Na solução modal, a matriz de amortecimento modal é informada a priori no arquivo persistido de entrada. A classe de interface externa *Main* lê a matriz modal nesse arquivo e ela é, então, passada diretamente para a classe de solução modal a ser utilizada (*ModalEq*, *DisplacementSuperposition* ou *AccelerationSuperposition*). Isso significa que o amortecimento na solução modal do INSANE é uma propriedade da solução.

No caso de um problema por integração direta, o arquivo persistido de entrada contém as propriedades do material. A classe *Main* lê esses parâmetros e passa essa informação para a classe *Material*. A classe *Driver* da solução por integração direta monta a matriz de amortecimento e passa a mesma para classe de solução *DirectIntegrationEq*. Portanto, no caso de soluções por integração direta, o amortecimento é uma propriedade do modelo.

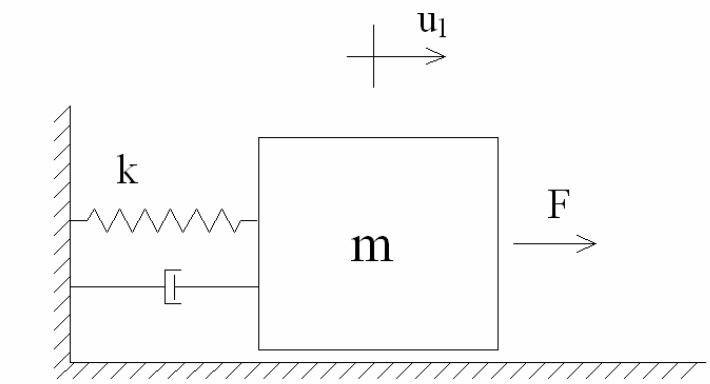


FIGURA 5.2: Sistema massa mola de 1GDL com amortecimento

A solução analítica desse problema para carregamento degrau é dada por:

$$u(t) = \frac{f_0}{k}\left[1 - e^{-\zeta\omega_n t}\left(\cos\omega_d t + \frac{\zeta\omega_n}{\omega_d} sen\,\omega_d t\right)\right] \tag{5.2}$$

onde f_0 é a amplitude do carregamento degrau.

O amortecimento modal adotado para esse problema foi $\zeta = 0{,}25$. Os demais parâmetros são idênticos ao caso anterior.

O terceiro problema de 1GDL considerou o caso de um sistema com massa distribuída. Ao discretizar o problema de meio contínuo da Figura 5.3 com um único elemento L2, o problema de MEF resultante se transforma num problema de 1GDL idêntico ao representado pela FIGURA 5.1b.

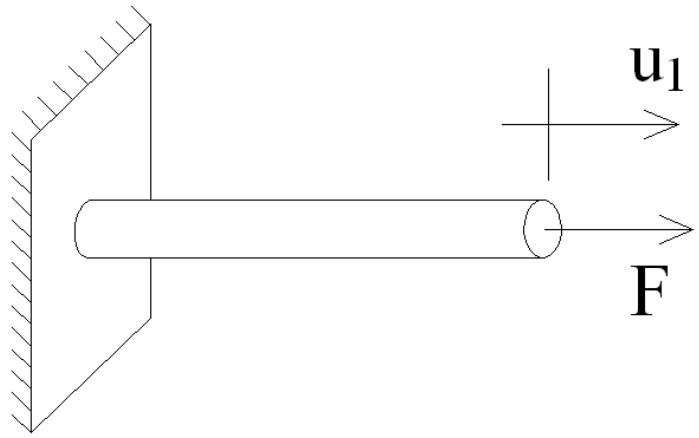


FIGURA 5.3: Modelo de barra solicitada axialmente

As matrizes de massa e rigidez do elemento L2 são dadas, respectivamente por:

$$\mathbf{M} = \frac{\rho L}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{5.3}$$

$$\mathbf{K} = \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{5.4}$$

Adotando-se os seguintes parâmetros:

- $\rho = 3{,}0$
- $L = 1{,}0$
- $A = 1{,}0$
- $E = 1{,}0$

o problema discreto se reduz ao caso do sistema de 1GDL com $m = 1$ e $k = 1$. O objetivo desse problema é validar a implementação da formação das matrizes de massa para a família de elementos paramétricos.

5.1.1 Solução Modal

O primeiro passo para fazer a integração modal do problema é obter o modo natural e a respectiva freqüência natural. Para tal, utilizou-se os solucionadores modais de iteração inversa e iteração no subespaço com precisão[1] de 1,0x10^{-6}.

Em ambos os casos, obtiveram-se os mesmos resultados:

- ω = 1,000
- ϕ = 1,000

Não existe nenhum ganho entre se utilizar o método de iteração no subespaço em relação ao de iteração inversa num problema de 1GDL, mas sua utilização se justifica pelo fato de se tratar de um processo de validação.

A partir daí, as classes para integração no tempo, *DisplacementSuperposition* e *AccelerationSuperposition*. É preciso, então, definir o tamanho do passo de tempo na integração. Craig (1981) recomenda que este seja inferior a 0,1T_n, onde T_n é o período natural. No presente trabalho, adotou-se como regra um passo de tempo nunca superior a 0,05T_n de modo a evitar que erros inerentes à ao processo numérico se confundissem com possíveis erros de implementação.

Os parâmetros da solução numérica utilizados foram:

- $\Delta t \approx \frac{T_n}{20}$
- Ordem de Integração Temporal = 3

A Figura 5.4 mostra o comportamento da solução analítica dada por (5.1) comparada com a solução obtida pelo INSANE. O tempo na geração desses resultados foi normalizado com relação ao período natural, como segue:

$$\bar{t} = \frac{t}{T_n} = \frac{\omega t}{2\pi} \tag{5.5}$$

---

[1] A precisão foi definida por $\frac{|\lambda_{n+1} - \lambda_n|}{\lambda_n}$

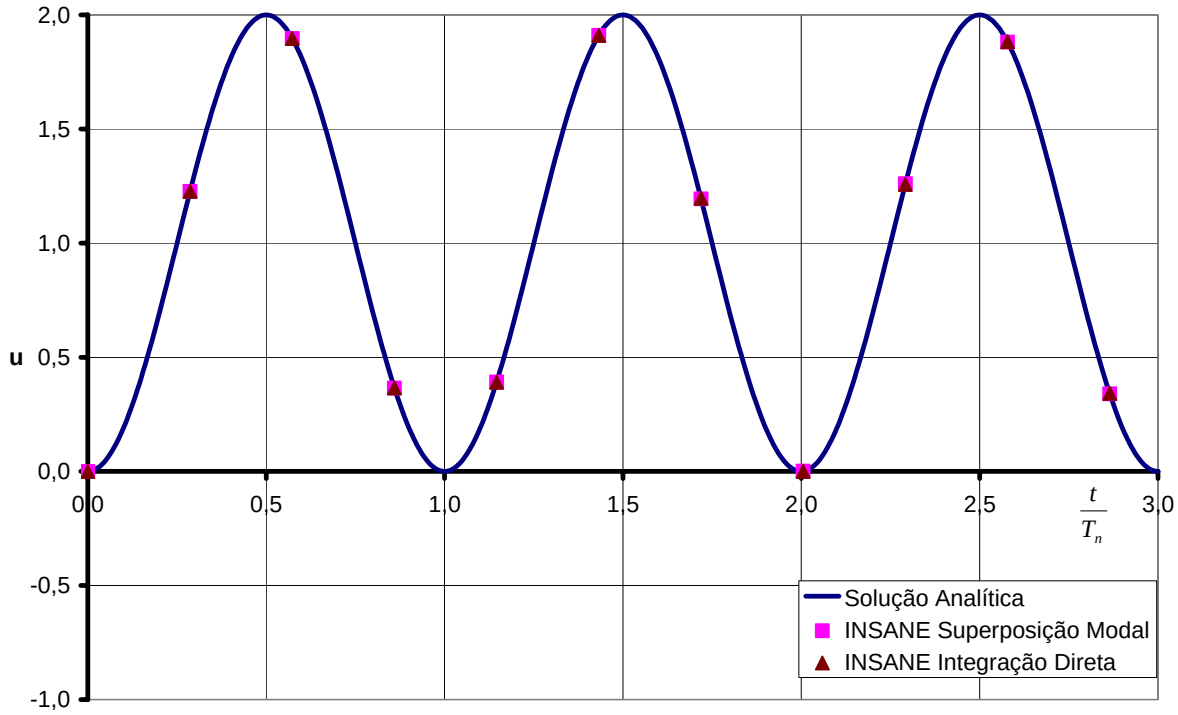


FIGURA 5.4: Deslocamento x tempo (normal) do problema de 1 GDL

Como era de se esperar, a solução por superposição de deslocamentos e superposição de acelerações levaram exatamente aos mesmos resultados.

Esses mesmos parâmetros de solução numérica foram utilizados para se obter a solução do problema de 1GDL com amortecimento da Figura 5.2, e o sistema contínuo da Figura 5.3. Para o caso do sistema com amortecimento, os resultados obtidos pelo INSANE foram levados no gráfico da FIGURA 5.5 e comparados com a Eq.(5.2). É possível notar uma excelente concordância.

Para o caso do sistema com matriz de massa consistente, a comparação é a mesma já feita para o problema da Figura 5.1a, uma vez que o sistema discreto resultante é o mesmo daquele caso. Conforme esperado, os resultados obtidos foram idênticos aos obtidos para o problema da Figura 5.1a.

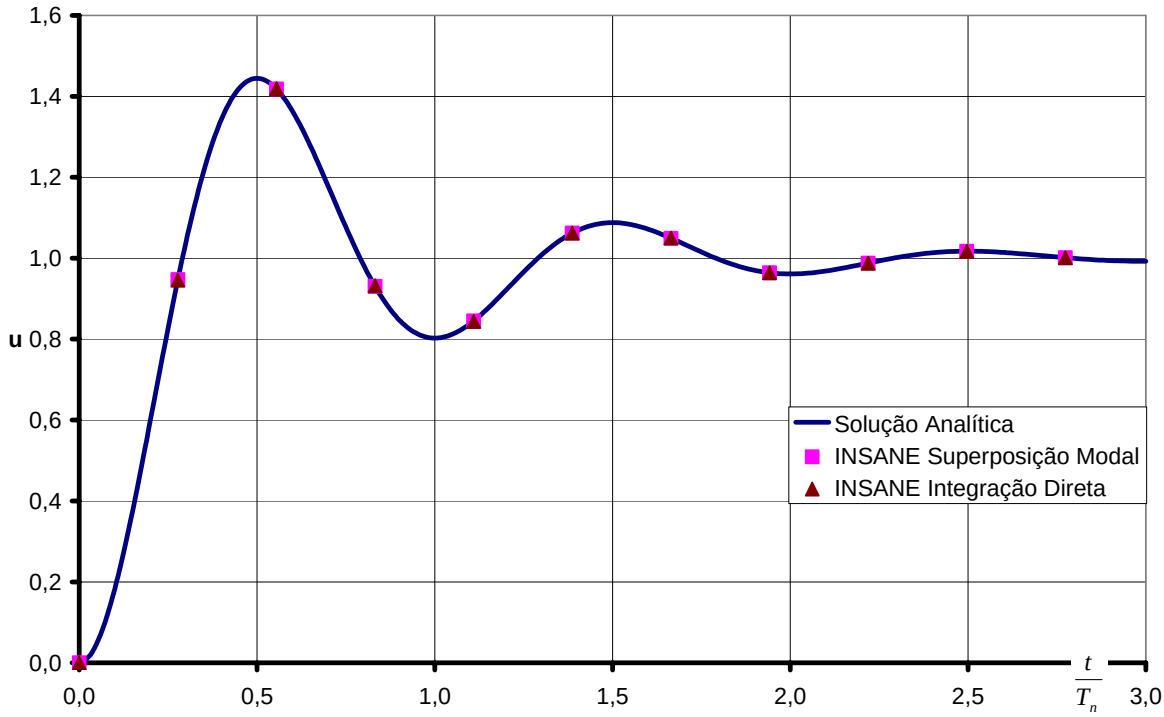


FIGURA 5.5: Comparativo entre o INSANE e a solução exata do problema de 1GDL com amortecimento modal

5.1.2 Solução por Integração Direta

Resolveu-se o sistema de 1GDL sem amortecimento com o mesmo passo de tempo da solução modal. Os demais parâmetros da solução por integração direta foram:

- $\beta = 0{,}25$
- $\gamma = 0{,}5$
- $\alpha = 0$

Os resultados obtidos por esse esquema de integração foram levados no gráfico da Figura 5.4. Como se pode notar, obtiveram-se praticamente os mesmos resultados obtidos com os métodos de superposição modal. Repetiu-se a solução anterior utilizando, agora, os seguintes parâmetros do método de *Newmark-*β:

- $\beta = 0{,}3025$
- $\gamma = 0{,}6$
- $\alpha = -0{,}1$

A diferença obtida foi da ordem de 0,001%.

Para o caso do sistema de 1GDL com amortecimento, para se obter uma matriz de amortecimento que correspondesse a ζ = 0,25 utilizando na solução modal, adotou-se o parâmetro 'a_{r2}' de Rayleigh como sendo 0,5. Os resultados obtidos na integração direta mostrados Figura 5.5 coincidem com os resultados obtidos pela integração modal para o mesmo problema.

5.2 Sistema Linear com nGDL

Uma vez validado o INSANE para problemas de 1GDL, partiu-se para sua validação para sistemas com nGDL. O problema escolhido consiste de um edifício de quatro andares com massas concentradas e carga do tipo degrau unitário na extremidade superior, conforme mostrado na Figura 5.6a. A malha de elementos finitos utilizando-se elementos L2 está mostrada na Figura 5.6b.

Os dados do problema são:

- m_1 = 1000
- m_2 = 2000
- m_3 = 2000
- m_4 = 3000
- k_1 = 800 x 10^3
- k_2 = 1600 x 10^3
- k_3 = 2400 x 10^3
- k_4 = 3200 x 10^3
- $F = f_0 \operatorname{sin} \Omega t$
- f_0 = 1,0
- Ω = 6,6468

A solução analítica desse problema para o caso não amortecido é dada por:

$$u_1(t) = \sum_{i=1}^{4} \frac{f_0 \phi_{1,i}^2}{\omega_i^2} \left[\frac{\operatorname{sen} \Omega t - r_i \operatorname{sen} \omega_i t}{\left(1 - r_i^2\right)} \right] \tag{5.6}$$

Nota-se que a Eq. (5.6) é o método de superposição de deslocamentos para o problema. Craig (1981) encontrou os seguintes valores para as freqüências naturais e seus respectivos modos:

- ω_1 = 13,29
- ω_2 = 29,66
- ω_3 = 41,08
- ω_4 = 55,88

- $\boldsymbol{\Phi} = \begin{bmatrix} 0{,}5899 & 0{,}6777 & -0{,}4315 & 0{,}0808 \\ 0{,}4595 & -0{,}0675 & 0{,}4787 & -0{,}2348 \\ 0{,}2929 & -0{,}3659 & -0{,}0759 & 0{,}5239 \\ 0{,}1387 & -0{,}2966 & -0{,}3389 & -0{,}3337 \end{bmatrix}$

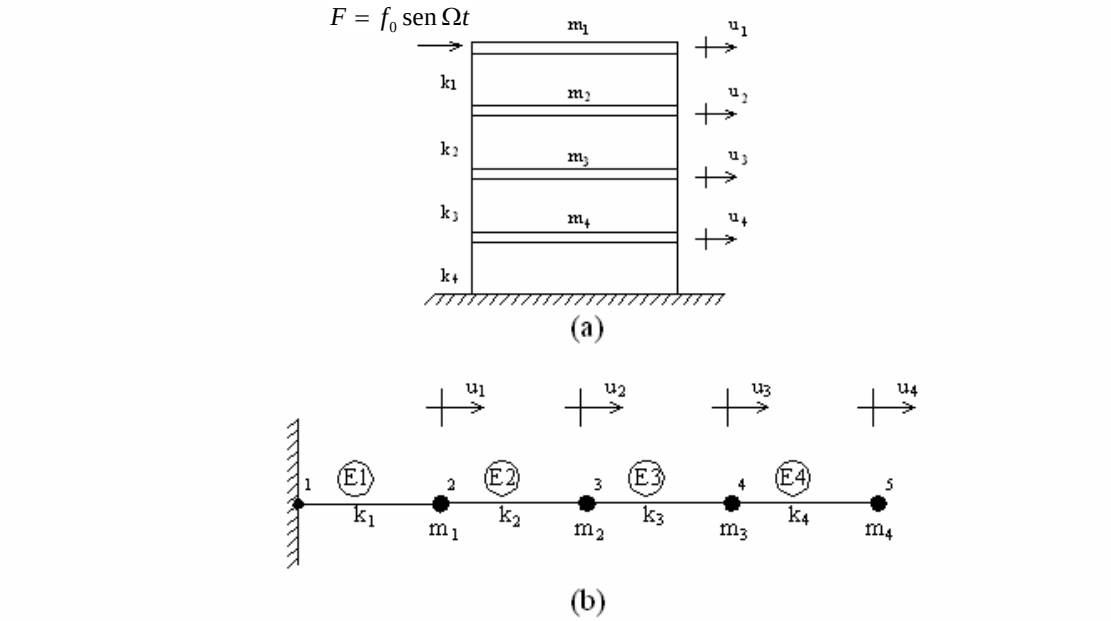


FIGURA 5.6: (a) Modelo de um edifício de quatro andares.(b) Modelo de elementos finitos

Posteriormente, resolveu-se o problema para o caso de amortecimento modal dado por:

- $\zeta_1 = 0{,}1614$
- $\zeta_2 = 0{,}1726$
- $\zeta_3 = 0{,}2097$
- $\zeta_4 = 0{,}2747$

Esses valores correspondem a se adotar os coeficientes de Rayleigh 'a_{r1}' e 'a_{r2}' como 1 e $1\text{x}10^{-3}$ respectivamente. A solução analítica desse problema, considerando condições iniciais homogêneas, é dada pelo método da superposição de deslocamentos por:

$$u_1(t) = \sum_{i=1}^{4} U_i \left[\operatorname{sen}(\Omega t - \alpha_i) + e^{-\zeta_i \omega_i t} \left(A_{1i} \cos \omega_d t + A_{2i} \operatorname{sen} \omega_d t \right) \right] \quad (5.7)$$

onde:

$$U_i = \frac{f_0 \phi_{1i}^2}{\left[\left(1 - r_i^{\,2}\right)^2 + \left(2\zeta_i r_i\right)^2\right]^{\frac{1}{2}}} \tag{5.8}$$

$$A_{1i} = \operatorname{sen} \alpha_i \tag{5.9}$$

$$A_{2i} = \zeta_i \frac{\omega_{ni}}{\omega_{di}} A_{1i} - \frac{\Omega}{\omega_d} \cos \alpha_i \tag{5.10}$$

$$\alpha_i = \operatorname{arcotan} \frac{2\zeta_i r_i}{1 - r_i^{\,2}} \tag{5.11}$$

Utilizando as freqüências obtidas por Craig (1981) e a Eq. (2.21), chega-se ao seguintes valores para as freqüências naturais amortecidas:

- ω_{d1} = 13,12
- ω_{d2} = 29,21
- ω_{d3} = 40,17
- ω_{d4} = 53,73

5.2.1 Solução Modal

Os parâmetros da solução modal foram os seguintes:

- Precisão de 1 x 10^{-6}
- $\Delta t \approx \frac{T_{n4}}{20}$
- Ordem de Integração Temporal = 3

Obteve-se uma excelente concordância com os resultados obtidos por Craig (1981), seja pelo método da iteração inversa quanto pelo método da iteração no subespaço.

Na Figura 5.7, a solução obtida através do método da superposição de acelerações modais com diferentes truncamentos modais é comparada com a Eq.(5.6), o tempo está normalizado com relação ao período natural do primeiro modo. Nota-se que, mesmo com um único modo, a solução aproxima bastante da solução analítica. Isso se deve ao fato do carregamento ser aplicado de maneira relativamente lenta (Ω = 0,5 ω_1). Dessa maneira, o termo pseudo-estático da solução por superposição de acelerações modais é responsável por uma parte considerável da resposta.

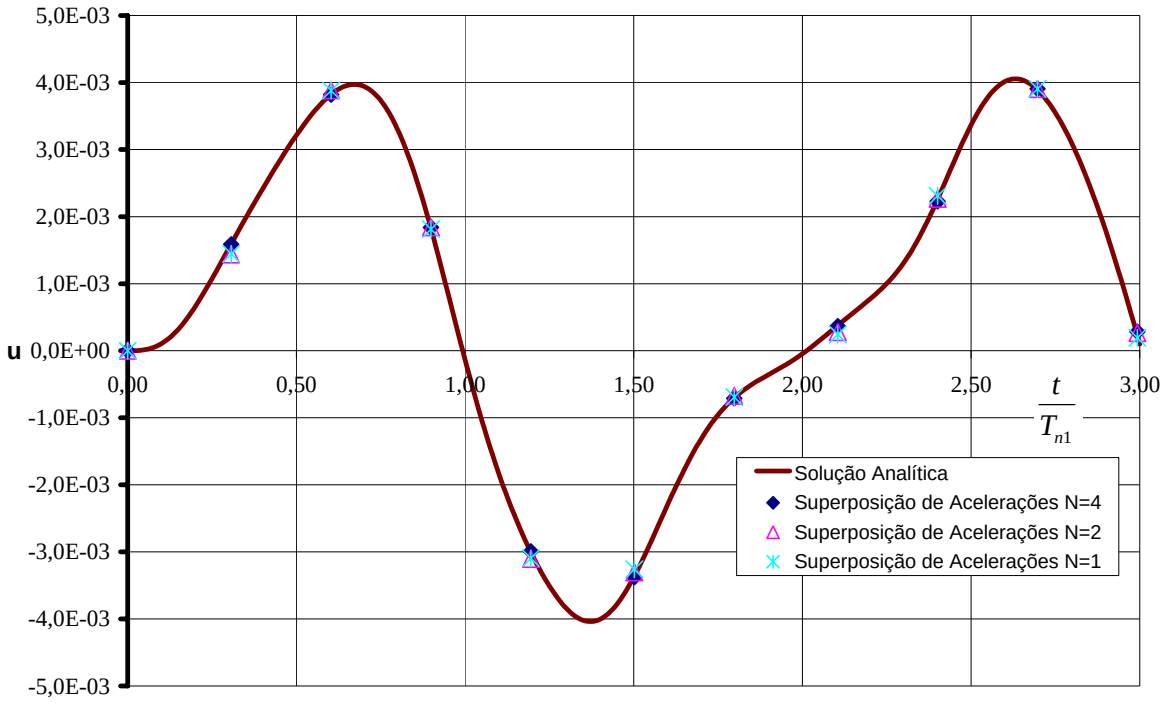


FIGURA 5.7: Solução analítica versus solução no INSANE por superposição de acelerações

Na Figura 5.8, comparou-se a solução obtida pelo programa utilizando-se superposição de deslocamentos modais com diferentes truncamentos modais com a Eq. (5.6). Novamente é possível notar uma excelente concordância entre a solução analítica e a solução numérica, mas para o caso de truncamento modal N = 1, nota-se um resultado inferior ao que foi obtido com esse mesmo truncamento para o método de superposição de acelerações modais. Esse resultado é esperado devido ao fato, já mencionado, da presença do termo de resposta pseudo-estática no método de superposição de acelerações modais.

Na Figura 5.9, comparou-se os resultados obtidos pelo método de superposição de acelerações modais sem truncamento para o caso do edifício com amortecimento modal com a Eq. (5.7). Novamente nota-se uma excelente correlação de resultados.

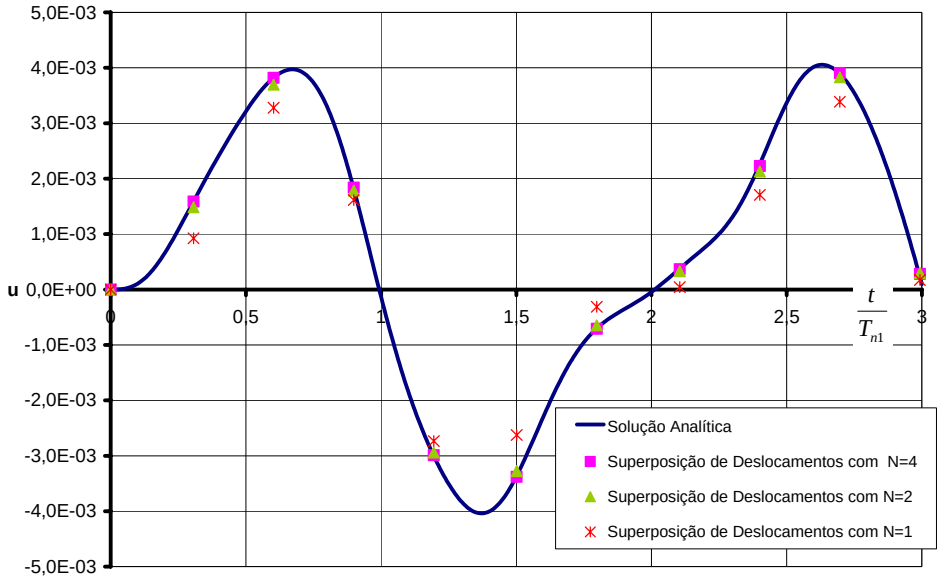


FIGURA 5.8: Solução analítica versus solução no INSANE via superposição de deslocamentos modais

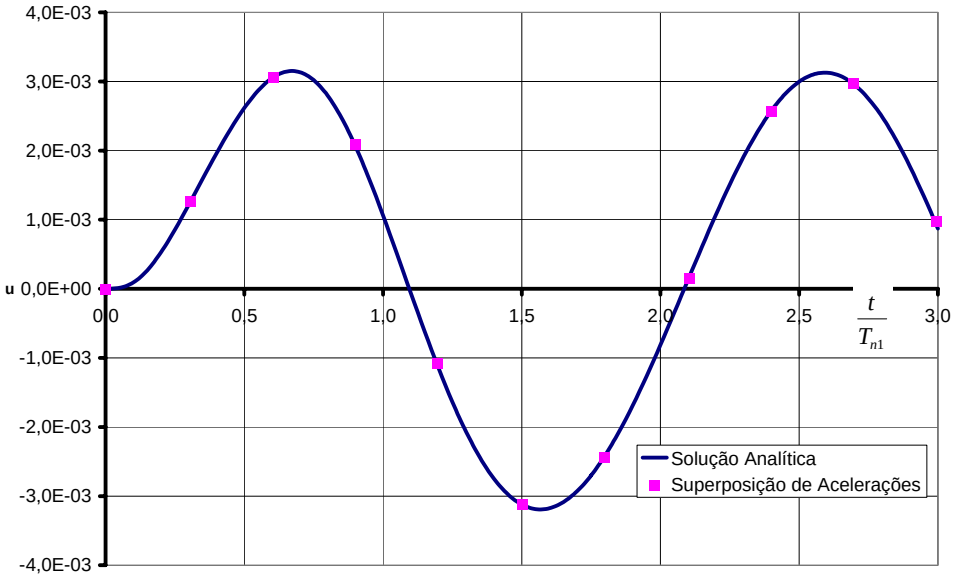


FIGURA 5.9: Comparativo entre solução analítica e superposição de acelerações sem truncamento modal

5.2.2 Solução por Integração Direta

A solução por integração direta do problema sem amortecimento foi feita com três diferentes valores para o passo de tempo. A princípio, utilizou-se o mesmo passo de tempo utilizado na integração modal, ou seja, $\Delta t \approx 0{,}05T_{n4}$. Posteriormente, dobrou-se o passo de tempo e, por

fim, utilizou-se um passo de tempo quatro vezes maior. Como pode se observar na Figura 5.10, a solução numérica não sofreu muita influência devido à essa variação., donde é possível concluir que o critério adotado, $\Delta t \approx 0{,}05T_{n4}$, foi adequado para o propósito desse trabalho, onde erros inerentes ao processo de integração numérica no tempo poderiam dificultar a análise dos resultados.

Deve se ressaltar um fato que torna a escolha para o passo de tempo para esse problema ainda mais conservadora: o critério para sua escolha foi feito em cima do período natural do quarto modo. Entretanto, quando se analisa a Figura 5.7 e a Figura 5.8, nota-se que a contribuição do terceiro e quarto modos são desprezíveis. Esse fato também contribuiu para que o aumento de Δt, não influenciasse significativamente nos resultados mostrados na Figura 5.10.

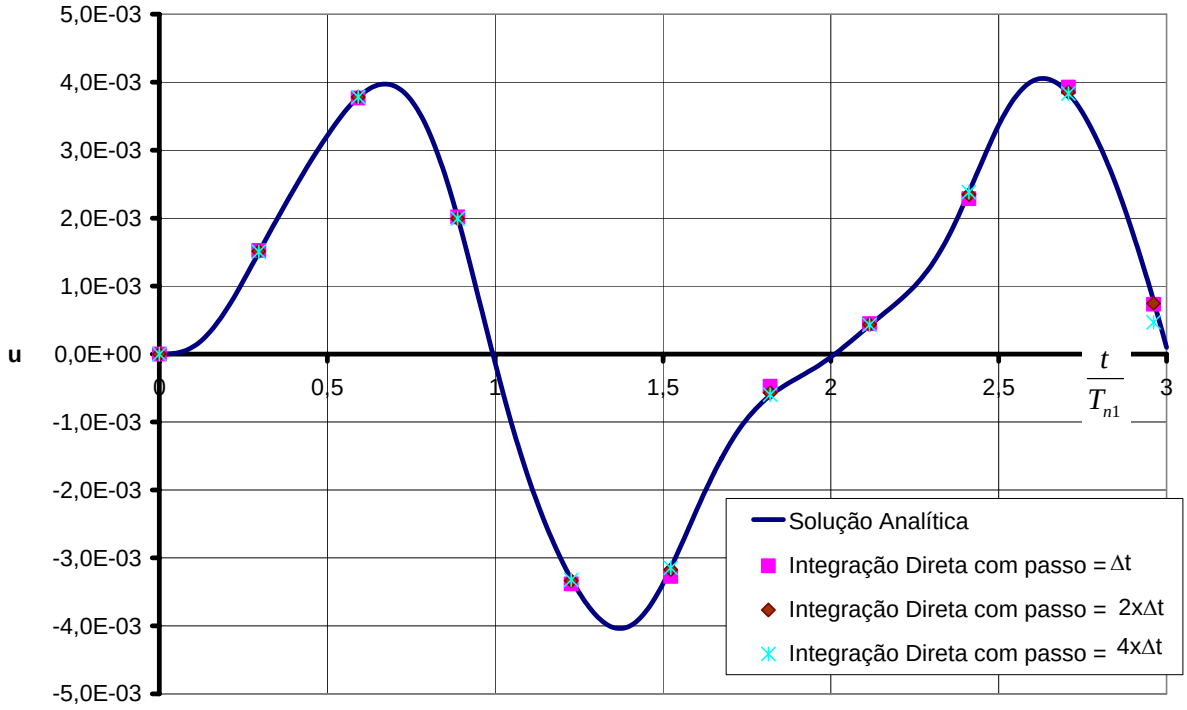


FIGURA 5.10: Solução por integração direta do edifício de quatro andares

Na Figura 5.11 comparou-se o resultado o resultado obtido para o edifício com amortecimento modal pelo método da integração direta com o resultado obtido pela superposição de acelerações modais sem truncamento modal. O objetivo é certificar que a formação da matriz de amortecimento montada no caso das soluções por integração direta está coerente com a matriz de amortecimento modal utilizada nas soluções por decomposição

modal[1]. O mesmo passo de tempo, $\Delta t \approx 0{,}05T_{n4}$, foi utilizado em ambos os casos. Pode se notar uma excelente correspondência entre ambas soluções.

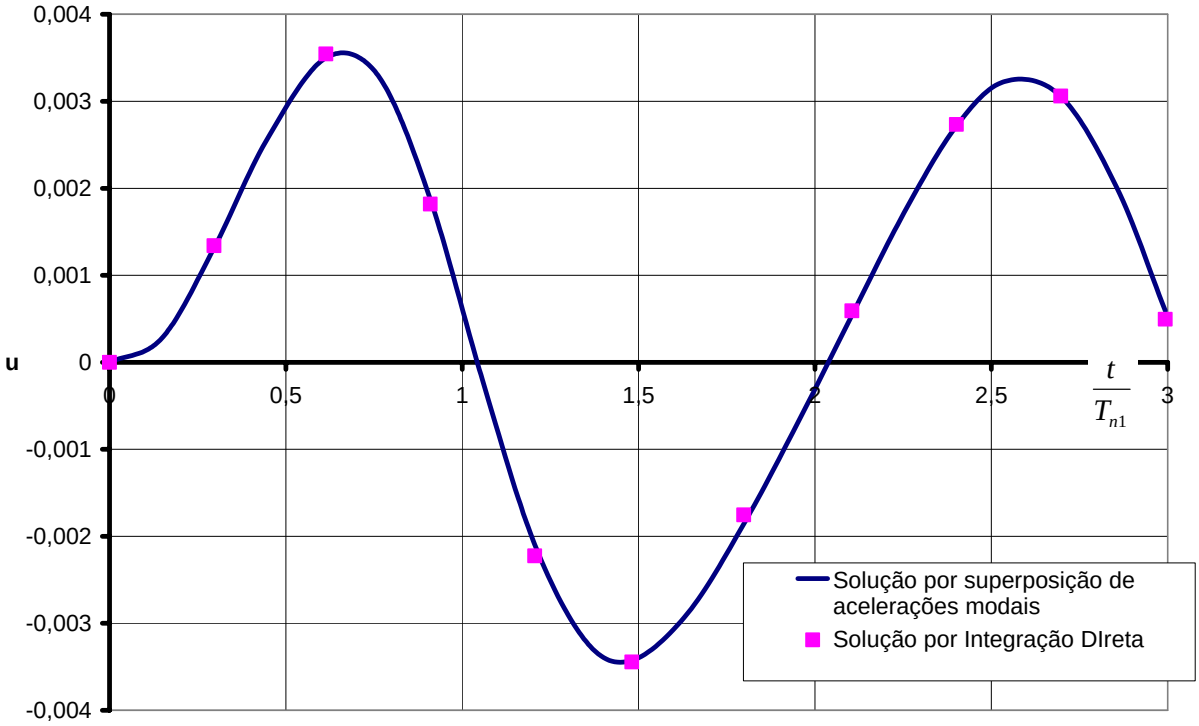


FIGURA 5.11: Comparativo da solução por integração modal e solução por integração direta

5.3 Sistema Unidimensional com Elemento L3

O objetivo desse exemplo é validar o INSANE também para o elemento linear de três nós L3 de modo a concluir a validação do programa para sistemas unidimensionais. O principal aspecto a analisar é a implementação da formação da matriz de massa do elemento paramétrico L3. O problema escolhido é um sistema contínuo como o da Figura 5.3 que, nesse exemplo, é discretizado com um único elemento L3, conforme mostrado na Figura 5.12.

---

[1] Embora o método de integração direta não seja limitado a casos de amortecimento modal, implementou-se o amortecimento de *Rayleigh* para esse método justamente para permitir comparações com os métodos de decomposição modal.

Na extremidade livre da barra atua um carregamento harmônico de amplitude unitária e freqüência $\Omega = 2$.

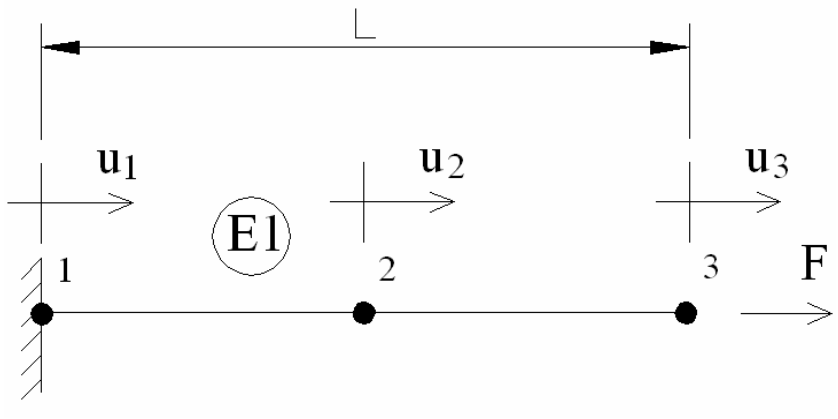


FIGURA 5.12: Barra simulada por um elemento linear de três nós

As matrizes de rigidez e massa desse elemento são dadas, respectivamente, por:

$$\mathbf{K} = \frac{EA}{3L}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix} \tag{5.12}$$

$$\mathbf{M} = \frac{\rho L}{30}\begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix} \tag{5.13}$$

Uma vez que um dos graus de liberdade está restrito, o problema modelado pelo MEF se resume à solução de um sistema massa-mola com dois graus de liberdade e massas concentradas nos nós. A solução analítica do modelo discreto resultante é a mesma dada pela Eq. (5.6).

A equação característica desse problema é um polinômio do 2^o grau. Solucionando essa equação para $\rho = 30$, $L = 1$, $A = 1$ e $E = 3$, chega-se a:

- $\omega_1 = 0{,}4985$
- $\omega_2 = 1{,}7939$
- $\mathbf{\Phi} = \begin{bmatrix} 0{,}2598 & 0{,}4463 \\ 0{,}1836 & -0{,}1815 \end{bmatrix}$

5.3.1 Solução por Superposição de Acelerações

A solução numérica para as freqüências e os modos naturais utilizando precisão de $1x10^{-6}$ coincidiram com a solução previamente obtida. Na FIGURA 5.13, comparou-se o resultado obtido pelo método de superposição de acelerações com ordem de integração 4, passo de aproximadamente $0{,}05T_{2n}$ e sem truncamento modal com a solução analítica dada pela Eq. (5.6). O tempo está normalizado com relação ao período natural do primeiro modo. Houve uma excelente correspondência entre a solução numérica do INSANE e a solução analítica.

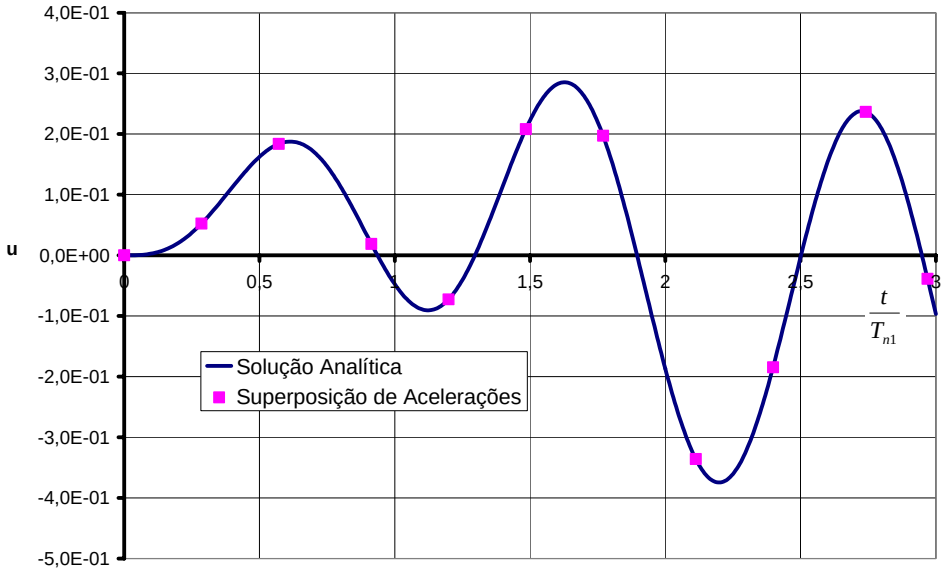


FIGURA 5.13: Solução analítica para barra discretizada usando elementos L3 versus solução do INSANE com superposição de acelerações

5.4 Treliça Plana

A treliça plana da Figura 5.14 foi submetida a duas condições distintas. Primeiramente, aplicou-se um carregamento degrau unitário na extremidade superior na direção horizontal, (Figura 5.14(a)) enquanto que, no segundo caso (Figura 5.14(b)), a treliça foi submetida à condições iniciais não nulas em todos os seus graus de liberdade e, depois, foi deixada em vibração livre. Mais precisamente, os deslocamentos aplicados aos graus de liberdade foram de 0,1. A treliça foi modela com três elementos L2. O objetivo básico desse exemplo é validar os elementos lineares para o caso plano.

Para o primeiro caso, a solução analítica é obtida aplicando-se a Eq. (5.1) nas direções principais da estrutura e projetando-se os deslocamentos principais sobre as coordenadas globais. Nesse caso, está-se interessado no deslocamento horizontal do nó ‘2’.

$$u_2(t) = \sum_{i=1}^{3} \frac{\phi_{2,i}}{\omega_i^2}\left(1 - \cos \omega_i t\right) \tag{5.14}$$

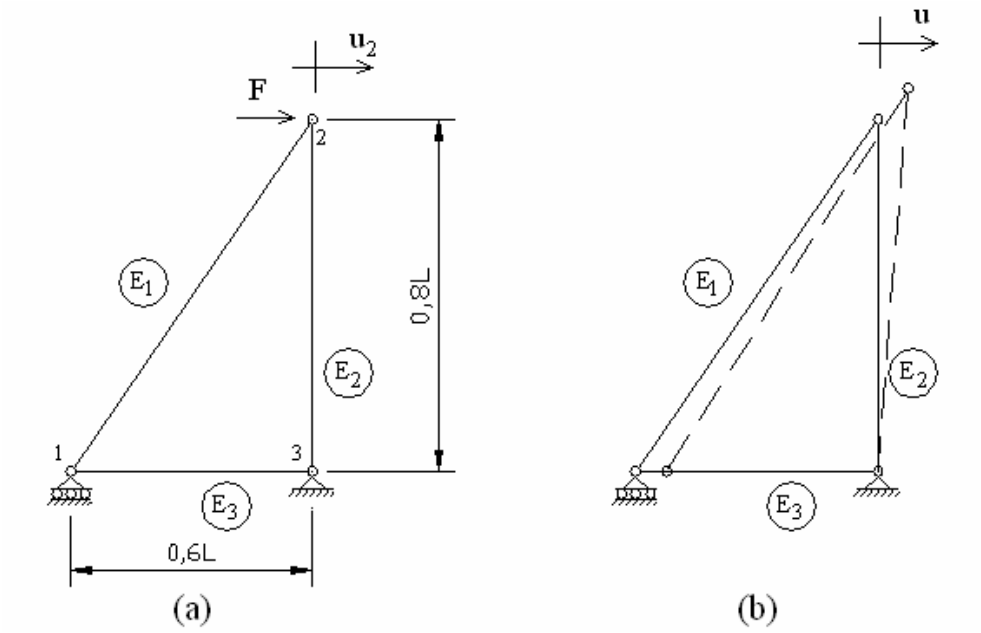


FIGURA 5.14: (a) Treliça plana submetida a carregamento degrau. (b) Treliça com deslocamentos iniciais não nulos em vibração livre

Para o segundo caso, embora também possa se obter uma solução fechada similar à Eq. (5.14), será adotada a solução de Weaver e Johnston (1987), que resolveram esse problema tanto pelo método de superposição modal quanto pelo método da integração direta *Hilber-*α*.*

Os parâmetros do problema são:

- $E = 30\text{x}10^6$
- $\rho = 7{,}35\text{x}10^{-4}$
- $L = 250$
- $A_1 = 10{,}0$
- $A_2 = 8{,}0$
- $A_3 = 6{,}0$

onde A_i é a área de cada elemento.

Weaver e Johnston (1987) encontraram os seguintes valores para a vibração livre da estrutura:

- $\omega_1 = 420$
- $\omega_2 = 1168$
- $\omega_3 = 1862$

- $\mathbf{\Phi} = \begin{bmatrix} 0{,}2068 & 0{,}6921 & -0{,}9113 \\ 0{,}8936 & -0{,}1369 & 0{,}5513 \\ -0{,}2209 & 0{,}7983 & 0{,}5561 \end{bmatrix}$

5.4.1 Solução Modal

Para o problema da treliça sujeita a carregamento degrau unitário, adotou-se $\Delta t \approx 0{,}05T_1$. Optou-se por tomar como referência o primeiro modo na escolha do passo da integração numérica devido a se esperar que apenas o primeiro modo contribua significativamente para o deslocamento. Um carregamento degrau tem o efeito de mudar o ponto de equilíbrio da estrutura e introduzir energia cinética na mesma, donde se conclui que o efeito de um carregamento degrau é a vibração livre da estrutura em torno de um ponto de equilíbrio deslocado. Nessas condições, espera-se que apenas o primeiro modo seja significativo.

Utilizando precisão de $1\text{x}10^{-8}$ chegou-se a praticamente os mesmos valores encontrados por Weaver e Johnston (1987) para as freqüências e os modos da estrutura. A partir daí, utilizando ordem de integração 4 no tempo, obteve-se os deslocamentos tanto pelo método da superposição de acelerações quanto pelo método de superposição de deslocamentos. Como não foi utilizado truncamento modal, não houve diferenças entre ambos. Os resultados obtidos foram, então, comparados com a Eq. (5.14), (Figura 5.15) onde se nota uma excelente correspondência entre as soluções numéricas e a solução analítica. Na figura, o tempo foi normalizado com relação ao período natural do primeiro modo.

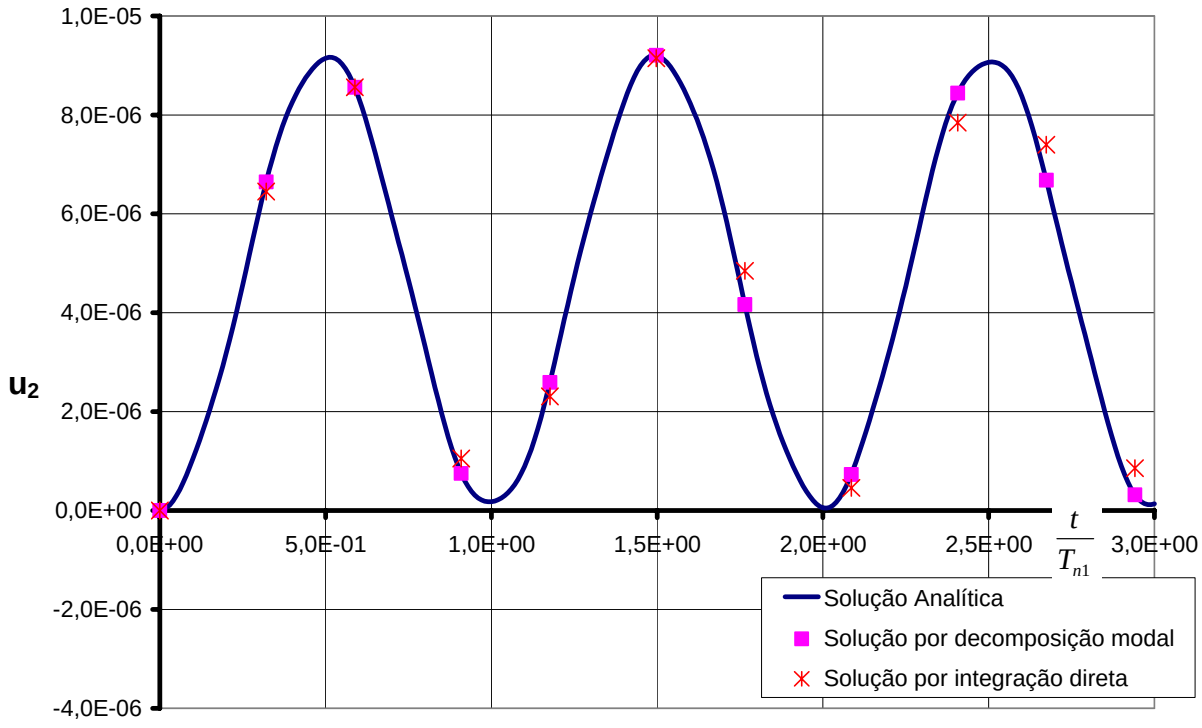


FIGURA 5.15: Treliça plana submetida a carregamento degrau

A seguir, resolveu-se o problema da treliça em vibração livre sujeita a deslocamentos iniciais de 0,1 em todos os seus GDLs. Dessa vez, adotou-se $\Delta t = 0{,}01$, que foi o mesmo valor utilizado por Weaver e Johnston. A Figura 5.16 mostra a comparação dos resultados obtidos pelo INSANE com os resultados obtidos por Weaver e Johnston. Como não se usou truncamento modal, tanto o método de superposição de acelerações quanto o método de superposição de deslocamentos modais levaram ao mesmo resultado. Utilizaram-se duas ordens de integração no tempo diferentes: um e três. Praticamente não houve diferenças em função da mudança na ordem de integração no tempo.

Já os referidos autores usaram superposição de deslocamentos modais, também sem truncamento modal, sendo que os deslocamentos principais foram obtidos pela quadratura de *Euler*. Existe uma boa correspondência entre as respostas da referência e as respostas obtidas pelo programa. Porém, essa correspondência foi inferior à obtida quando se compararam os resultados do programa com a solução analítica do problema da treliça plana sujeita a carregamento degrau. A princípio, a explicação para essa diferença é o fato de Weaver e Johnston utilizarem a quadratura de *Euler*, enquanto no INSANE utiliza a quadratura de *Gauss*, mais precisa do que a de *Euler*. Como o passo de tempo adotado é relativamente grande, uma vez que T_{n1} = 0,015, ambos os métodos de integração estão longe da

convergência. Isso faz com que pequenas diferenças impliquem em grandes diferenças relativas.

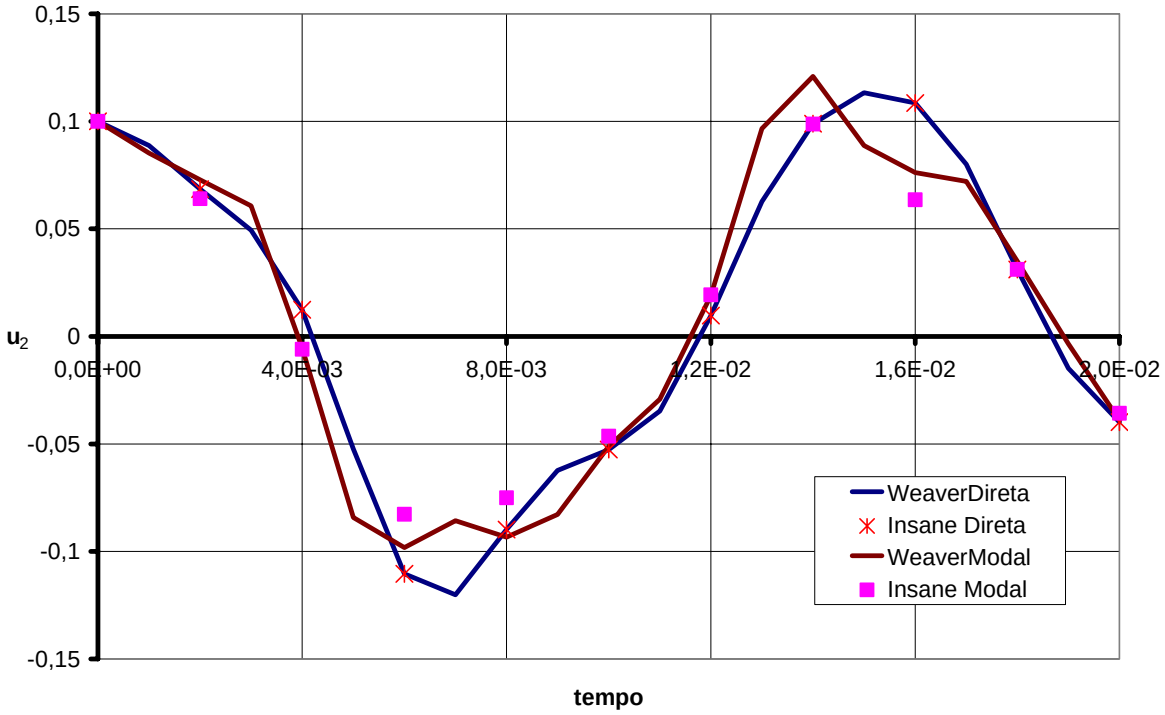


FIGURA 5.16: Comparativo de soluções para a treliça plana submetida à condições iniciais não nulas

5.4.2 Solução por Integração Direta

Utilizando o mesmo passo de tempo da solução modal e com os parâmetros da integração direta de *Newmark* sendo $\beta = 0{,}3025$, $\gamma = 0{,}6$ e $\alpha = -0{,}1$, obtiveram-se os resultados mostrados na Figura 5.15 para o caso da treliça plana submetida à carga do tipo degrau. Pode-se notar que há uma excelente concordância entre a solução por integração direta e a solução analítica nos primeiros ciclos. Nota-se também que, a partir do terceiro ciclo, a solução modal apresenta resultados um pouco melhores do que a solução por integração direta. Esse resultado é esperado, uma vez que nos métodos de superposição modal, utilizou-se a quadratura de *Gauss* com ordem de integração temporal de 4, o que significa que a precisão

do resultado é de 7ª ordem[1]. Já o método de *Newmark-*β com a modificação *Hilber-*α tem, no máximo, precisão de 4ª ordem.

Para o problema da treliça com deslocamentos iniciais não nulos, utilizou-se o mesmo passo de tempo e os mesmos parâmetros da integração direta utilizados por Weaver e Johnston: $\Delta t = 0{,}01$, β = 0,3025, γ = 0,6 e α = -0,1. Na Figura 5.16 é possível notar que o resultado obtido via INSANE é coincidente com o da referência uma vez que, para esse caso, o esquema de integração é o mesmo adotado por esses autores.

5.5 Vibração Transversal de uma Viga com Elementos de Estado Plano

Uma vez obtidos resultados para elementos paramétricos unidimensionais, partiu-se para a validação de problemas bidimensionais. O primeiro problema escolhido foi a simulação da vibração transversal de uma viga com elementos de estado plano.

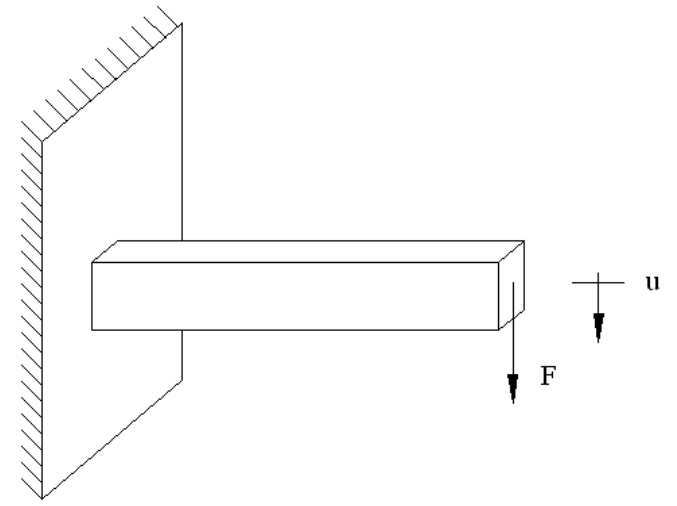


FIGURA 5.17: Viga engastada submetida a carga degrau na extremidade livre

[1] A quadratura de Gauss é exata para integração de polinômios de grau 2n-1, onde n é o número de pontos de integração.

O deslocamento de uma massa concentrada em uma viga como a da Figura 5.17 é equivalente a um problema massa-mola de 1GDL. Assim, é possível obter uma solução analítica para a resposta no tempo, a qual será utilizada para validar a simulação feita no INSANE. No caso de uma viga engastada, a constante de mola da sua extremidade livre é dada por:

$$k = \frac{3EI}{L^3} \tag{5.15}$$

onde I é a inércia à flexão, E é o módulo de elasticidade e L é o comprimento da viga.

Sejam os seguintes parâmetros de uma viga de seção retangular 1,0×1,0:

- $E = 2{,}5\text{x}10^9$
- $I = 0{,}08333$
- $L = 10{,}0$

e supondo massa concentrada igual a 2,0, tem-se $\omega_n = 559{,}0$.

Supondo carregamento degrau na extremidade e condições iniciais nulas, a solução analítica para a resposta no tempo será dada pela Eq.(5.1). Para a modelagem da viga com elementos planos, utilizou-se uma malha com 40 elementos Q4 quadrados (Figura 5.18). Essa malha foi escolhida após uma análise de convergência dentro do próprio INSANE. Porém, os estudos de convergência para um modelo parecido realizados por Cook, Malkus e Plesha (1989) e por Logan (1992) foram utilizados. O carregamento e a massa na extremidade livre foram distribuídos parabolicamente, o que resultou na seguinte distribuição nodal.

- $m_1 = 0{,}375$
- $m_2 = 1{,}25$
- $F_1 = 0{,}375$
- $F_2 = 1{,}25$

Além disso, como se trata de elementos de estado plano, foi necessário adotar um valor para o coeficiente de Poisson. Em estudo semelhante realizado por Cook, Plesha e Malkus (1989), adotou-se $\nu = 0{,}25$.

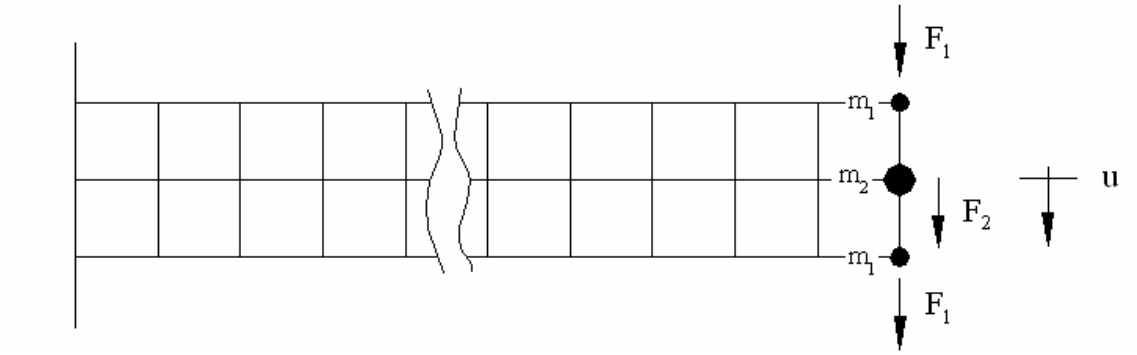


FIGURA 5.18: Malha de elementos Q4 para viga em balanço

5.5.1 Solução Modal

A obtenção da freqüência e do modo natural do modelo de elementos finitos da Figura 5.18, tanto pela método da iteração inversa quanto pela método da iteração no subespaço levou à praticamente os mesmos resultados. Também não houve diferenças significativas quando se utilizaram elementos de estado plano de tensão ou estado plano de deformação. Com uma precisão modal de $1x10^{-8}$, obteve-se $\omega_n = 579,0$.

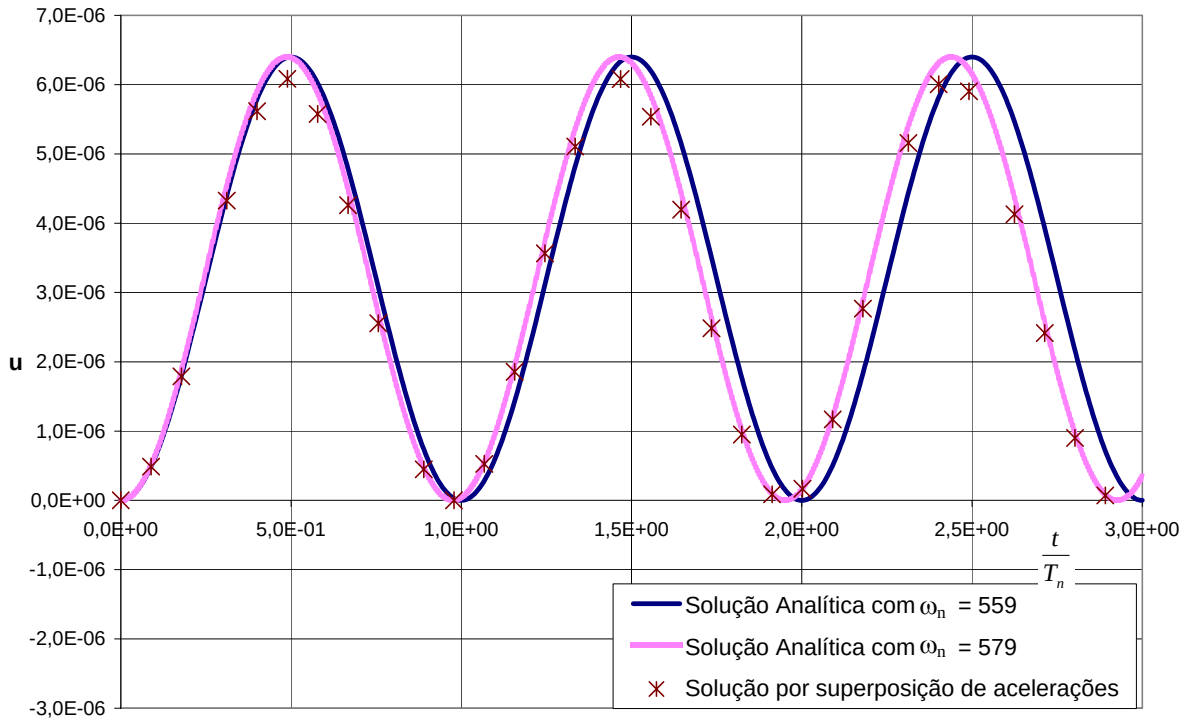


FIGURA 5.19: Resposta da viga em balanço submetido a um carregamento degrau

De posse da freqüência natural e do seu respectivo modo, obteve-se a resposta no tempo da vibração da extremidade livre da viga utilizando o método de superposição de acelerações modais com ordem de integração igual a 4. Novamente, adotou-se um passo de tempo de aproximadamente $0{,}05T_n$. Os resultados obtidos nesse esquema de integração foram comparados com os resultados da Eq. (5.1) com ω_n = 559. Esses resultados estão mostrados na Figura 5.19, onde se pode notar que, mesmo havendo uma defasagem entre a solução analítica e a solução numérica, houve uma boa concordância de resultados. O tempo está adimensionalizado com relação ao período natural da solução analítica com ω_n = 559.

Entretanto, a defasagem observada se deve ao fato da Eq. (5.15) ser derivada da formulação de vigas de *Euler-Bernouli*, que não pode ser totalmente representada pelo modelo de estado plano, e não a erros na implementação computacional. Isso fica bem evidenciado quando se utiliza na Eq. (5.1) a freqüência natural obtida no modelo de estado plano, conforme mostrado na Figura 5.19. Pode se notar que existe uma perfeita concordância entre os resultados analíticos e os obtidos pelo INSANE.

5.5.2 Solução por Integração Direta

Devido ao fato de se ter adotado um modelo com massa concentrada apenas na extremidade livre da viga, a matriz de massa desse sistema é singular. A singularidade da matriz de massa impede que sejam calculadas as acelerações iniciais necessárias no processo de integração direta.

5.6 Problema de Estado Plano

Weaver e Johnston (1987) sugeriram o problema de estado plano de deformações mostrado na Figura 5.20. O componente de máquina foi modelado com elementos Q8 e o carregamento aplicado é do tipo degrau, com distribuição parabólica ao longo da face.

Os parâmetros do problema são:

- E = $207\text{x}10^9$
- ν = 0,3
- ρ = 7850
- $2R_1 = R_2 = 0{,}02$

- espessura = 1,0
- $b_x = 1000$
- $\zeta = 0{,}05$[1]

sendo b_x o valor máximo atingido pelo carregamento parabólico.

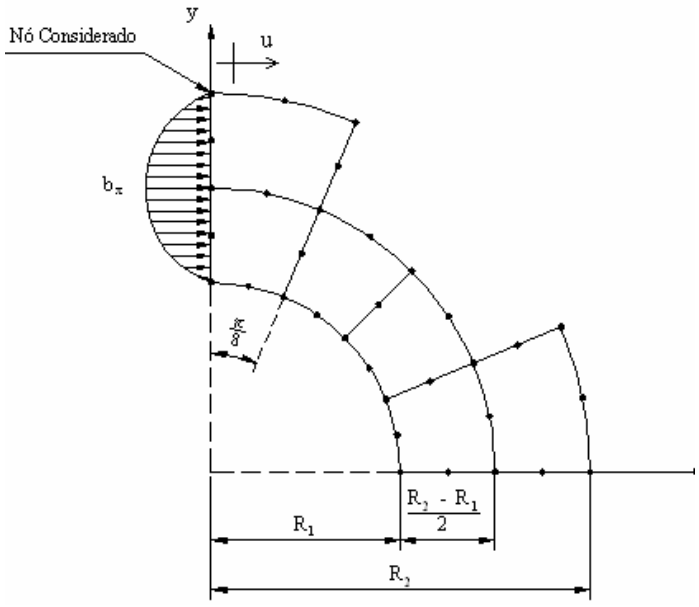


FIGURA 5.20: Componente de máquina submetido à estado plano de deformação

5.6.1 Solução Modal

A solução modal desse problema foi obtida com uma precisão de $1x10^{-9}$. Como era de se esperar, ambos os solucionadores modais chegaram aos mesmos resultados. A Figura 5.21 mostra o primeiro modo da peça, que é consistente com o que se pode intuir da geometria da mesma. A freqüência circular do 1º modo é 54000.

Posteriormente, obteve-se a resposta no tempo para o carregamento proposto. Três soluções numéricas foram obtidas utilizando passo de tempo de $5x10^{-6}$ e ordem de integração 4. Na primeira solução, utilizou-se o método de superposição de acelerações com truncamento no 5º modo. O número elevado de modos se justifica, pois essa solução será adotada como referência[2]. As outras duas soluções obtidas são por superposição de acelerações e de

---

[1] Weaver e Johnston resolvem o problema por integração direta mas não deixam claro como transformaram o amortecimento modal para gerar a matriz de amortecimento nas coordenadas globais.

[2] A forma como a solução desse problema foi apresentada por Weaver e Johnston (1987) só permitiu que alguns poucos pontos ao longo do tempo fossem determinados, por isso preferiu-se adotar uma solução do INSANE como sendo a referência e comparar os pontos identificados na solução de Weaver e Johnston com essa solução de referência.

deslocamentos truncando no 1º modo. Pode-se notar pela Figura 5.22 que mesmo para um único modo, os resultados obtidos tanto pela superposição de acelerações quanto pela superposição de deslocamentos já é bastante razoável. O motivo para isso é, novamente, o carregamento do tipo degrau, conforme discutido na seção 5.4.1. Também é possível notar uma excelente concordância com os resultados de Weaver e Johnston (1987), sendo que esses autores utilizaram o método da integração direta de *Newmark* com a modificação *Hilber-*α.

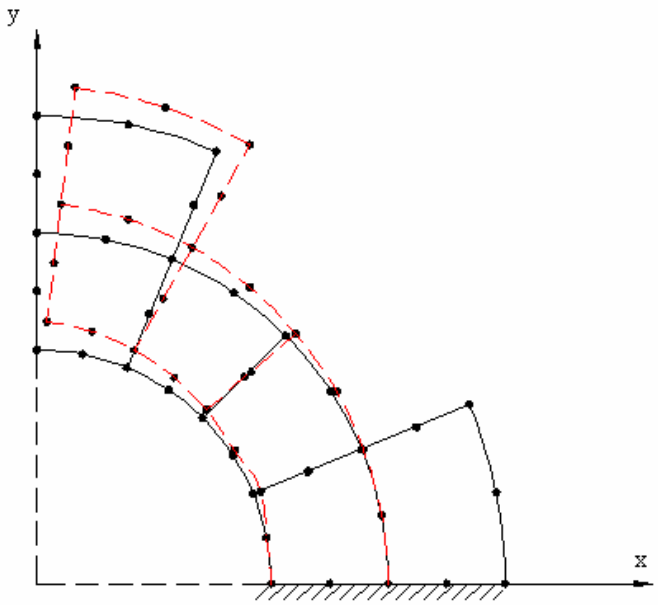


FIGURA 5.21: Primeiro modo do elemento de máquina

5.6.2 Solução Por Integração Direta

A solução por integração direta foi obtida utilizando o mesmo passo de tempo utilizado na solução modal. Os parâmetros da solução por integração direta são $\beta = 0{,}3025$, $\gamma = 0{,}6$ e $\alpha = -0{,}1$. Uma seleção para os coeficientes de Rayleigh que produzem um amortecimento modal $\zeta = 0{,}05$ para o primeiro modo é $a_{r1} = 0$ e $a_{r2} = 0{,}000001739$. O resultado da integração direta leva, praticamente, aos mesmos resultados das soluções modais.

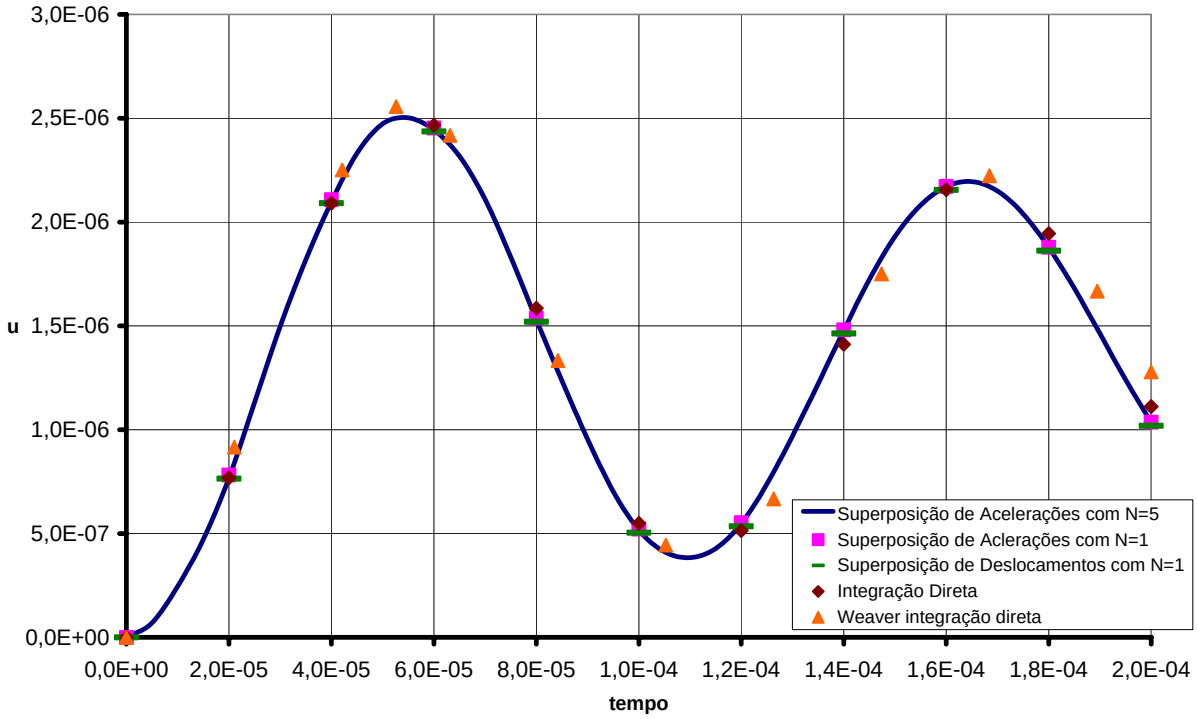


FIGURA 5.22: Deslocamento do nó de referência ao longo do tempo

Capítulo 6

CONSIDERAÇÕES FINAIS

6.1 Conclusões

A expansão para análise dinâmica do INSANE consistiu de:

i. Adaptação da interface de leitura e gravação de dados para as características do modelo dinâmico.

ii. Foram implementados dois algoritmos de solução modal, o método da iteração inversa e o método da iteração no subespaço; dois métodos de integração de sistemas dinâmicos por decomposição modal, o método de superposição de deslocamentos e o método de superposição de acelerações; e foi implementado o método *Newmark-*β com a modificação *Hilber-*α para integração direta da equação do movimento.

iii. Expandiu-se a classe dos elementos paramétricos de forma que essas pudessem calcular as matrizes de massa e amortecimento.

Uma vez feita a expansão, as implementações foram validadas comparando-se os resultados numérico-computacionais com soluções de referência. A concordância de resultados permite afirmar que a expansão do INSANE para análise dinâmica via elementos finitos paramétricos foi bem sucedida.

Uma análise importante a ser feita é a contribuição desse trabalho para os objetivos maiores do INSANE. Como o projeto ainda está numa fase inicial, as pesquisas atualmente em curso terão um impacto grande na estruturação da plataforma. Alguma reestruturação de classes ainda é relativamente simples. Se o crescimento esperado para o projeto for alcançado, as reestruturações serão mais difíceis, devendo ser evitadas. É preciso reconhecer que a utilização de uma linguagem orientada a objetos e de conceitos da POO não garante, per si, que se esteja explorando toda a potencialidade do paradigma de orientação a objetos. Nesse sentido, a criação de classes genéricas para encapsular funcionalidades de possível uso comum deve ser procurada, enquanto que funcionalidades específicas devem ser encapsuladas por classes especializadas.

A expansão realizada neste projeto atendeu a esse propósito da POO quando:

i. Implementou fora das classes de solução do problema dinâmico a solução do problema de autovalor.

ii. Implementou classes genéricas para tratar as funções analíticas.

iii. Implementou a classe para solução modal como superclasse dos métodos de solução por decomposição modal. Outros problemas modais em engenharia de estruturas poderão utilizar essa classe sem maiores mudanças.

Algumas implementações, por outro lado, podem ter encapsulado funcionalidades úteis em outros problemas fora de dinâmica de estruturas, e uma revisão dessas classes pode ser necessária. É o caso da integração de *Duhamel*, que foi implementada dentro da *ModalEq*, mas pode ser utilizada em outros problemas de engenharia.

6.2 Sugestões para Trabalhos Futuros

Existem características presentes na implementação atual que não foram totalmente testadas ou não estão completamente concluídas. Esses itens complementares e pequenas adaptações consistem de:

i. Verificar mais alguns casos de problemas com elementos paramétricos de modo a se ter uma segurança maior quanto à correta implementação do cálculo da massa e do amortecimento nesses elementos.

ii. Validar o caso com condições de contorno variáveis no tempo.

iii. Terminar a implementação e validar os métodos de decomposição modal para problemas com movimento de corpo rígido.

iv. Validar a implementação dos cálculos das reações de apoio.

v. Fazer modificações na interface de saída de dados para permitir a gravação de reações de apoio e tensões ao longo do processo de integração. Atualmente, apenas os valores das reações e tensões do último passo de tempo são persistidos.

vi. Modificar a leitura e a gravação de dados na classe *Main* e na lógica de formação dos vetores de tempo de modo a permitir a análise de casos com carregamentos genéricos, não analíticos, especificados pelo usuário.

vii. Implementar o tratamento de exceções para a expansão feita.

viii. Expandir a interface gráfica do INSANE para contemplar os problemas de dinâmica.

Outro tipo de trabalho que poderia dar seqüência ao presente projeto são aqueles que possuem objetivos mais abrangentes. Algumas possibilidades são:

i. Realizar um estudo dedicado da organização de classes do INSANE de modo a tornar mais fáceis futuras expansões.

ii. Implementar o estudo de sistemas não lineares.

iii. Realizar um estudo e implementar casos de amortecimento estrutural baseados em modelos constitutivos do mesmo.

iv. Realizar um estudo de desempenho da implementação e fazer as melhorias pertinentes.

Uma outra sugestão que não se encaixa exatamente nos dois grupos anteriores seria fazer um estudo e incluir metodologias de condensação de matrizes de modo a permitir que problemas com matrizes de massa singulares como no caso do modelo de viga utilizado nesse trabalho (seção5.5).

REFERÊNCIAS BIBLIOGRÁFICAS

Adhikari, S. Wagner, N. (2004); Direct Time-Domain Integration Method for Exponentially Damped Linear Systems, Computers and Structures Journal, publicações Elsevier publications.

Bathe, K. J. (1982); Finite Element Procedures in Engineering Analysis; Prentice-Hall, New Jersey, EUA.

Boyce, W. E, DiPrima, R. C. (1996), Equações Diferenciais Elementares e Problemas de Valores de Contorno, 6ª edição.

Camarão, C., Figueiredo, L. (2003); Programação de Computadores em JAVA, Editora LTC, Rio de Janeiro.

Cook, R. D., Malkus, D. S. e Plesha, M. E. (1989); Concepts and Applications of Finite Element Analysis; 3ª edição; John Wiley & Sons; Madison, EUA.

Craddock, J. N. (2000); Notas de aula da disciplina Finite Element Analysis do curso de pós-graduação em Engenharia Civil da Southern Illinois University, Carbondale, EUA.

Craig Jr., R. R. (1981); Structural Dynamics, An Introduction to Computer Methods, John Wiley & Sons Inc., Nova Iorque, EUA.

Dalim, H. (2002); Vibration Analysis and Control of Smart Structures, tese de doutorado, Newcastle University, New South Wales, Austrália.

Gamma, E., et al, (1995); Design Patterns – Elements of Reusable Object-Oriented Software; Addison-Wesley.

Guedes, G. T. A. (2005); UML 2 – Guia de Consulta Rápida, Novatec Editora, São Paulo.

Holzner, S. (2002); C++ Black Book; Makron Books, São Paulo.

Hortsmann, C. S. (1997); Practical Object-Oriented Development in C++ and Java, Nova Iorque, EUA.

Hughes, T. J. R. (2000); The Finite Element Method, Dover Publications, Nova Iorque, EUA..

Kijewski, T., Kareem, A. (2000); Estimation and Modeling of Damping and Auxiliary Damping Systems in Civil Engineering Structures: An Overview, Notre Dame University, Notre Dame, Indiana, EUA.

Lemay, L., Cadenhead, R. (2001); Aprenda em 21 Dias Java 2 - Professional Reference, Editora Campus, Rio de Janeiro.

Logan, D. L. (1992); A First Course in the Finite Element Method; PWS Publishing Company, Boston, EUA.

Martha, L. F. e Parente Junior, E. (2002); An Object Oriented Framework for Finite Element Programming; Fifth World Congress on Computational Mechanics; Viena, Austria.

Mase, G. E. (1970); Theory and Problems of Continuum Mechanics, McGraw Hill Book Company, Nova Iorque, EUA.

Medeiros, E. S. (2004); Desenvolvendo Software com UML 2.0, Makron Books, São Paulo.

Object Management Group (2003); OMG Unified Language Modeling Specification, OMG Publishing, Needham.

Pantalé, O. (2002); An Object-oriented Programming of an Explicit Dynamics Code: Application to Impact Simulation; Advances in Engineering Software 2002, Elsevier Science Ltd.

Pesterev, A., Bergman, L. A. (1999); Application of the Mode Acceleration Technique to the Solution of the Moving Oscillator Problem, Proceedings of the ASME Design Engineering Technical Conferences, Nevada, EUA.

Pietro, G. A. (2001); Utilização de Padrões de Projeto na Reegenharia de Sistemas; tese de mestrado da Universidade Federal de São Carlos, São Carlos.

Pitangueira, R. L. S. P (2003); Métodos Matemáticos para Engenheiros, Apostila do Curso Métodos Matemáticos da Pós Graduação em Engenharia de Estruturas, UFMG, Belo Horizonte.

Pitangueira, R. L. S. P. (2005); Desenvolvimento Colaborativo no Projeto Insane; Projeto de Pesquisa para Solicitação de Bolsas junto ao CNPQ, Escola de Engenharia da UFMG, Belo Horizonte.

Santos, R. (2003); Introdução à Programação Orientada a Objetos Usando JAVA, Editora Campus, Rio de Janeiro.

Scanlan, R. H., Rosenbaum, R. (1951); Introduction to the Study of Aircraft Vibration and Flutter, McMillan Co.; Nova Iorque, EUA.

Silva, R. A., Carvalho, L. A. V. (1993); Otimização de Estruturas Treliçadas Sujeitas à Restrições Dinâmicas, XII Congresso Brasileiro de Engenharia Mecânica, ABCM, Brasília.

Soriano, H. L. e Lima, S. S. (1999), Método de Elementos Finitos em Análise de Estruturas, UFRJ, Rio de Janeiro.

Smart Draw Tutorial Versão 6; How to Draw UML Diagrams; Smart Draw; San Diego, EUA.

Timoshenko, S. P., Young, D. H. e Weaver Jr., W. (1972); Vibration Problems in Engineering; 4ª edição; John Wiley & Sons; Nova Iorque, EUA.

Velloso, F. (2005); Java Brilha em Projetos Open Source, Revista Java Magazine, 25ª edição, ano III, Junho de 2005, Editora DevMedia

Weaver Jr., W e Johnston, P. R. (1987); Structural Dynamics by Finite Elements, Prentice-Hall, Inc, New Jersey, EUA.

Zienckiewicz, O. C. (1989); The Finite Element Methdod, 3rd Edition, McGraw Hill, New Delhi, India.

APÊNDICES

A1 DESENVOLVIMENTO COLABORATIVO NO INSANE

Iniciado em agosto de 2002, o INSANE acumula doze subprojetos que envolvem alunos de graduação, mestrado e doutorado. Os subprojetos INSANE são associados a uma ou mais vertentes complementares, conforme mostra a Tabela 6.1.

TABELA 6.1: Divisão de vertentes dentro do projeto INSANE

NOME DA VERTENTE	DESCRIÇÃO
Núcleo Numérico	Vertente voltada para a implementação, no núcleo numérico do sistema, dos recursos fundamentais do MEF.
Ensino do MEF	Vertente voltada para a criação de recursos para apoio ao ensino do MEF em cursos de graduação e pós-graduação.
Computação Gráfica	Vertente que contempla a disponibilização de recursos gráficos interativos para apoio à pesquisa envolvendo o MEF.
Recurso para Pesquisa	Vertente que utiliza os recursos disponíveis no sistema para a pesquisa, ampliando a complexidade do mesmo.
Internet	Vertente que visa disponibilizar através da WEB os recursos gerados pelo projeto, de maneira a permitir usufruir deste novo paradigma de comunicação em favor da pesquisa.

Os subprojetos concluídos, aqueles em desenvolvimento e os que estão em fase de elaboração são brevemente apresentados a seguir.

A1.1 Subprojetos Concluídos

1) Análise de meios Semi-Frágeis Heterogêneos através de Descrição Cinemática de Cosserat e Descrição Estática de Microplanos

Aluna: Jamile Salim Fuina

Nível: Tese de Doutorado

Início: 03/2004

Término: 02/2008

Vertente: Recurso para Pesquisa

"Este subprojeto INSANE é parte da tese de doutorado da colaboradora. Para dar suporte ao trabalho acadêmico, o núcleo numérico do INSANE deve contemplar aspectos complexos da modelagem através do método dos elementos finitos, destacando-se: processos de solução das equações de equilíbrio não-lineares, descrição cinemática baseada no contínuo micropolar e modelos constitutivos de microplanos" Fuina e Pitangueira (2004), *apud* Pitangueira (2005).

1) Análise Fisicamente Não-Linear com Modelos Reticulados de Seções Transversais Compostas.

Aluno: Flavio Torres da Fonseca

Nível: Dissertação de Mestrado

Início: 08/2004

Término: 02/2006

Vertente: Núcleo Numérico e Recurso para Pesquisa

"Este subprojeto, parte do trabalho de mestrado do colaborador, introduz no INSANE recursos apropriados à análise fisicamente não-linear de estruturas de concreto armado, aço e/ou mistas através de modelos estruturais reticulados. Dentre estes recursos, destacam-se a definição de seções transversais compostas genéricas e decomposição da mesma, utilizando recursos de computação gráfica interativa e os processos incremental-iterativos para obtenção da resposta não-linear do modelo e de cada seção transversal que o compõe" Fonseca (2005) *apud* Pitangueira (2005).

2) Elementos Finitos para Modelos Estruturais de Barras

Aluno: Flavio Torres da Fonseca

Nível: Iniciação Científica

Início: 01/2003

Término: 08/2004

Vertente: Núcleo Numérico, Computação Gráfica Aplicada, Ensino do MEF

“Este trabalho de iniciação científica trata modelos estruturais de barras como casos particulares do método dos elementos finitos. Disponibiliza no núcleo numérico do INSANE, modelos de treliças planas e espaciais, pórticos planos e espaciais, vigas e grelhas. O pré e pós-processamentos destes modelos ainda estão restritos aos modelos planos. Persistência de dados como objetos Java ou em arquivos XML e geração de relatórios em formato PDF, também são tratados neste trabalho” Fonseca e Pitangueira (2004), *apud* Pitangueira (2005).

3) Gerador de Malhas Bidimensionais de Elementos Finitos Baseado em Mapeamentos Transfinitos

Aluno: Marco Antônio Brugiolo Gonçalves

Nível: Dissertação de Mestrado

Início: 08/2002

Término: 10/2004

Vertente: Computação Gráfica Aplicada

“Este subprojeto, parte do trabalho de mestrado do colaborador, refere-se ao desenvolvimento de um pré-processador que gera malhas bidimensionais de elementos finitos, persistindo-as como objetos Java ou arquivos XML, que podem ser exportados para o núcleo numérico do sistema. Cada tarefa do pré-processador é implementada através de um par vista-controlador, como no padrão MVC, que altera e consulta um modelo de geração de malhas baseado em mapeamentos transfinitos” Gonçalves (2004) *apud* Pitangueira (2005).

4) Elementos Finitos Paramétricos Implementados em JAVA

Aluno: Marcelo Lucas de Almeida

Nível: Dissertação de Mestrado

Início: 08/2003

Término: 08/2005

Vertente: Núcleo Numérico

“A dissertação de mestrado deste colaborador objetiva implementar a formulação paramétrica do método dos elementos finitos. Observando experiências de outros sistemas orientados a objetos, relatadas na literatura, abstrações de classes apropriadas são implementadas em Java de maneira a disponibilizar no INSANE, alguns elementos finitos de classe C0: unidimensionais, triangulares, quadrilaterais, tetraédricos e hexaédricos, com diferentes graus de interpolação Almeida” Almeida (2005) *apud* Pitangueira (2005).

5) Abordagem Orientada a Objetos para Análise Dinâmica via Método dos Elementos Finitos – É a dissertação aqui apresentada

A1.2 Subprojetos em Desenvolvimento

1) Análise de meios Semi-Frágeis Heterogêneos através de Descrição Cinemática de Cosserat e Descrição Estática de Microplanos

Aluna: Jamile Salim Fuina

Nível: Tese de Doutorado

Início: 03/2004

Término: 02/2008

Vertente: Recurso para Pesquisa

“Este subprojeto INSANE é parte da tese de doutorado da colaboradora. Para dar suporte ao trabalho acadêmico, o núcleo numérico do INSANE deve contemplar aspectos complexos da modelagem através do método dos elementos finitos, destacando-se: processos de solução das equações de equilíbrio não-lineares, descrição cinemática baseada no contínuo micropolar e modelos constitutivos de microplanos” Fuina e Pitangueira (2004), *apud* Pitangueira (2005).

2) Análise Fisicamente Não-Linear com Modelos Reticulados de Seções Transversais Compostas.

Aluno: Flavio Torres da Fonseca

Nível: Dissertação de Mestrado

Início: 08/2004

Término: 02/2006

Vertente: Núcleo Numérico e Recurso para Pesquisa

“Este subprojeto, parte do trabalho de mestrado do colaborador, introduz no INSANE recursos apropriados à análise fisicamente não-linear de estruturas de concreto armado, aço e/ou mistas através de modelos estruturais reticulados. Dentre estes recursos, destacam-se a definição de seções transversais compostas genéricas e decomposição da mesma, utilizando recursos de computação gráfica interativa e os processos incremental-iterativos para obtenção da resposta não-linear do modelo e de cada seção transversal que o compõe” Fonseca (2005) *apud* Pitangueira (2005).

3) Sistema Gráfico Interativo para Ensino do Método dos Elementos Finitos

Aluno: Renata Nicoliello Moreira

Nível: Dissertação de Mestrado

Início: 08/2004

Término: 02/2006

Vertente: Computação Gráfica e Ensino do MEF

“Neste subprojeto INSANE, parte da dissertação de mestrado da colaboradora, deseja-se que o processo de montagem e resolução de modelos discretos de elementos finitos tenha a interferência do usuário. Desta forma as entidades matemáticas pertinentes a modelos de elementos finitos de barras ou paramétricos planos podem ser interativamente alteradas e visualizadas, de maneira a enriquecer o processo de aprendizagem deste método, tanto em cursos de graduação quanto em cursos de pós-graduação” Moreira (2005) *apud* Pitangueira (2005).

A1.3 Subprojetos em Fase de Elaboração

1) Serviço WEB para o Método dos Elementos Finitos

Aluna: Luciana Sampaio Câmara

Nível: Dissertação de Mestrado

Início: 08/2005

Término: 02/2007

Vertente: Internet e Ensino do MEF

Este subprojeto INSANE, parte da dissertação de mestrado da colaboradora, refere-se à disponibilização de um serviço WEB para análise através do método dos elementos finitos. O serviço a ser disponibilizado requer a generalização do modelo da aplicação, para que o sistema possa contemplar diversos tipos de problemas, como, por exemplo, os problemas de dinâmica estrutural, os de campo generalizados e os de mecânica dos fluidos. Também é necessário o emprego de modernas tecnologias de banco de dados, Internet e Java para que o sistema funcione em uma arquitetura cliente-servidor com três camadas físicas: uma interface gráfica web para o pré e pós-processamento no lado do cliente, um processador localizado em um servidor web e arquivos textos e/ou binários persistidos em disco.

2) Pós-Processador para Resultados de Análise Não-Linear de Modelo do Método dos Elementos Finitos

Aluno: Samuel Silva Penna

Nível: Dissertação de Mestrado

Início: 08/2005

Término: 02/2007

Vertente: Computação Gráfica

Este subprojeto INSANE, parte da dissertação de mestrado do colaborador, refere-se ao desenvolvimento de um pós-processador que trata resultados oriundos de análise não-linear do método dos elementos finitos. A partir de modelos persistidos em objetos Java ou em arquivos XML, a aplicação deverá ser capaz de traduzir para o usuário, através de recursos gráficos interativos sofisticados as grandezas estáticas e cinemáticas disponíveis nos modelos. Cada tarefa do pós-processador é implementada através de um par vista-controlador, como no padrão MVC, que consulta o modelo persistido e gera imagens representativas de cada uma das grandezas.

3) Implementação de Elementos Finitos de Placas

Aluno: Samir Silva Saliba

Nível: Dissertação de Mestrado

Início: 08/2005

Término: previsto para 02/2007

Vertente: Núcleo Numérico

“Modelagens com elementos finitos de placas estão presentes em muitas aplicações da engenharia. Assim, não se concebe um sistema para fomentar pesquisa, sem estes elementos finitos. A dissertação de mestrado deste colaborador irá supri esta deficiência do INSANE, disponibilizando os diversos elementos finitos de placas de Kirchhoff (placas finas) e de Reissner-Mindlin (placas grossas), disponíveis na literatura” Pitangueira (2005).

4) Modelos de Elementos Finitos para Fissuração

Aluno: Kelson Pothin Wolff

Nível: Dissertação de Mestrado

Início: 08/2005

Término: previsto para 02/2007

Vertente: Recurso para Pesquisa

“Disponibilizar no núcleo numérico do INSANE modelos diversos para tratamento de fissuração é o principal objetivo da dissertação de mestrado deste colaborador. Pretende-se incluir os modelos de fissuração distribuída baseados na mecânica do dano, alguns modelos de microplanos, um modelo de fissuras discretas e um modelo de interface entre fissuras. Com este trabalho espera-se que o INSANE seja capaz de processar análise não-linear de meios constituídos de materiais parcialmente frágeis como concretos, argamassas e rochas” Pitangueira (2005).

A2 FORMATO DOS ARQUIVOS DE PERSISTÊNCIA

A2.1 Persistência do Modelo

```
- <Model class="DynamicFemModel">
  - <NodeList>
    - <Node label="1">
      <Coord>0.0 0.0 0.0</Coord>
      <Restraints>false true true true true true</Restraints>
      <Displacements>0.1 0.0 0.0 0.0 0.0 0.0</Displacements>
      - <MySpring>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MySpring>
      - <MyMass>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MyMass>
      - <MyDamp>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MyDamp>
      <Angle>0.0</Angle>
    </Node>
      - <Node label="2">
      <Coord>150 200 0.0</Coord>
      <Restraints>false false true true true true</Restraints>
      <Displacements>0.1 0.1 0.0 0.0 0.0 0.0</Displacements>
      - <MySpring>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MySpring>
      - <MyMass>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MyMass>
      - <MyDamp>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MyDamp>
      <Angle>0.0</Angle>
    </Node>
    - <Node label="3">
      <Coord>150 0.0 0.0</Coord>
      <Restraints>true true true true true true</Restraints>
      <Displacements>0.0 0.0 0.0 0.0 0.0 0.0</Displacements>
      - <MySpring>
      0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
      </MySpring>
```

```
            - <MyMass>
            0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
            </MyMass>
            - <MyDamp>
            0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
            </MyDamp>
            <Angle>0.0</Angle>
        </Node>
</NodeList>
- <MaterialList>
    - <Material class="Isotropic" label="mat">
        <Elasticity>30000000</Elasticity>
        <Density>0.000735</Density>
        <Poisson>0.3</Poisson>
        <RayleighCoeficientA>0.0</RayleighCoeficientA>
        <RayleighCoeficientB>0.0</RayleighCoeficientB>
    </Material>
</MaterialList>
<AnalysisModelList/>
- <CrossSectionList>
    - <CrossSection label="sec1">
        <Area>10</Area>
        <Ix>0.0</Ix>
        <Thickness>1.0</Thickness>
    </CrossSection>
    - <CrossSection label="sec2">
        <Area>8</Area>
        <Ix>0.0</Ix>
        <Thickness>1.0</Thickness>
    </CrossSection>
        - <CrossSection label="sec3">
        <Area>6</Area>
        <Ix>0.0</Ix>
        <Thickness>1.0</Thickness>
    </CrossSection>
</CrossSectionList>
- <ElementList>
        - <Element class="ParametricElement.Bar.L2" label="E1">
            <Type>Q S</Type>
            <Incidence>1 2</Incidence>
            <IntegrationOrder>2 2 0</IntegrationOrder>
            <MyMaterial>mat</MyMaterial>
            <MyCrossSection>sec1</MyCrossSection>
            <MyAnalysisModel>Line_2D</MyAnalysisModel>
            <MyElmPointForce/>
        </Element>
      - <Element class="ParametricElement.Bar.L2" label="E2">
            <Type>Q S</Type>
            <Incidence>2 3</Incidence>
```

```
            <IntegrationOrder>2 2 0</IntegrationOrder>
            <MyMaterial>mat</MyMaterial>
            <MyCrossSection>sec2</MyCrossSection>
            <MyAnalysisModel>Line_2D</MyAnalysisModel>
            <MyElmPointForce/>
            </Element>
        - <Element class="ParametricElement.Bar.L2" label="E3">
            <Type>Q S</Type>
            <Incidence>1 3</Incidence>
            <IntegrationOrder>2 2 0</IntegrationOrder>
            <MyMaterial>mat</MyMaterial>
            <MyCrossSection>sec3</MyCrossSection>
            <MyAnalysisModel>Line_2D</MyAnalysisModel>
            <MyElmPointForce/>
        </Element>
</ElementList>
<GlobalAnalysisModel>Line_2D</GlobalAnalysisModel>
- <Solution class="AccelerationSuperposition">
        <TimeIntegrationOrder>1</TimeIntegrationOrder>
        <DampVector>0.0 0.0 0.0</DampVector>
        <RayleighCoeficients>0.0 0.0</RayleighCoeficients>
        - <ModalProblem type="ISIModal">
            <Precision>0.000001</Precision>
            <NumberOfModes>3</NumberOfModes>
        </ModalProblem>
</Solution>
<DeltaTime>0.001</DeltaTime>
<GlobalEndTime>0.02</GlobalEndTime>
<Driver>AccelerationSuperpositionStructuralMech</Driver>
</Model>
```

A2.2 Persistência da Solução

- <JMefData>
- <Driver class="class
model.discrete.driver.AccelerationSuperpositionStructuralMech">
- <Solution class="AccelerationSuperposition">
<TimeIntegrationOrder>1</TimeIntegrationOrder>
- <ModalProblem>
<Precision>1.000E-06</Precision>
<NumOfModes>3.000E00</NumOfModes>
- <NaturalModes>
<Mode Label="1">2.068E-01 8.936E-01 -2.209E-01</Mode>
<Mode Label="2">6.925E-01 -1.372E-01 7.981E-01</Mode>
<Mode Label="3">9.105E-01 -5.511E-01 -5.570E-01</Mode>
</NaturalModes>
<NaturalFrequencies>4.200E02 1.168E03 1.862E03</NaturalFrequencies>
<NaturalDampedFrequencies>4.200E02 1.168E03 1.862E03
</NaturalDampedFrequencies>
</ModalProblem>
</Solution>
<DeltaTime>1.000E-03</DeltaTime>
<GlobalEndTime>2.000E-02</GlobalEndTime>
<NumberOfIntegrationSteps>20</NumberOfIntegrationSteps>
- <FemModel class="class
model.discrete.femmodel.DynamicFemModel">
<NumberOfEquations>3</NumberOfEquations>
<NumberOfRestraints>3</NumberOfRestraints>
- <NodeList>
- <Node label="1">
<Coord>0.000E00 0.000E00 0.000E00</Coord>
<Restraints>false true true true true true</Restraints>
<TimePreDisplacements/>
<TimePreVelocity/>
<TimePreAcceleration/>
<TimeForce/>
- <TimeHistory>
- <Step number="0">
- <Displacements>1.000E-01 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00
</Displacements>
- <Velocities>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00
</Velocities>
- <Accelerations>-5.985E04 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00
</Accelerations>
</Step>

OS PASSOS DE TEMPO DE 1 A 19 FORAM SUPRIMIDOS.

- <Step number="20">
- <Displacements>-3.878E-02 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</Displacements>
- <Velocities>7.662E01 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</Velocities>
- <Accelerations>6.376E04 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</Accelerations>
</Step>
</TimeHistory>
- <mySpring>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</mySpring>
- <myDamp>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</myDamp>
- <myMass>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</myMass>
<ForceOnSpring>0.000 -0.000 -0.000 -0.000 -0.000 -0.000</ForceOnSpring>
<Reactions>0.000 0.000 0.000 0.000 0.000 0.000</Reactions>
<Angle>0.0</Angle>
</Node>
- <Node label="2">
<Coord>1.500E02 2.000E02 0.000E00</Coord>
<Restraints>false false true true true true</Restraints>
<TimePreDisplacements/>
<TimePreVelocity/>
<TimePreAcceleration/>
<TimeForce/>
- <TimeHistory>
- <Step number="0">
- <Displacements>1.000E-01 1.000E-01 0.000E00 0.000E00 0.000E00 0.000E00</Displacements>
- <Velocities>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00</Velocities>
</Step>

OS PASSOS DE TEMPO DE 1 A 19 FORAM SUPRIMIDOS.

- <Step number="20">
- <Displacements>-3.565E-02 -1.482E-03 0.000E00 0.000E00 0.000E00 0.000E00</Displacements>
- <Velocities>-4.411E01 1.229E02 0.000E00 0.000E00 0.000E00 0.000E00</Velocities>

- <Accelerations>-1.993E04 1.105E03 0.000E00 0.000E00 0.000E00
0.000E00</Accelerations>
</Step>
</TimeHistory>
- <mySpring>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</mySpring>
- <myDamp>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</myDamp>
- <myMass>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</myMass>
<ForceOnSpring>0.000 0.000 -0.000 -0.000 -0.000 -0.000</ForceOnSpring>
<Reactions>0.000 0.000 0.000 0.000 0.000 0.000</Reactions>
<Angle>0.0</Angle>
</Node>
- <Node label="3">
<Coord>1.500E02 0.000E00 0.000E00</Coord>
<Restraints>true true true true true true</Restraints>
<TimePreDisplacements/>
<TimePreVelocity/>
<TimePreAcceleration/>
<TimeForce/>
- <TimeHistory>
- <Step number="0">
- <Displacements>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</Displacements>
- <Velocities>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</Velocities>
- <Accelerations>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</Accelerations>
</Step>

OS PASSOS DE TEMPO DE 1 A 19 FORAM SUPRIMIDOS.

- <Step number="20">
- <Displacements>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</Displacements>
- <Velocities>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</Velocities>
- <Accelerations>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</Accelerations>
</Step>
</TimeHistory>
- <mySpring>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</mySpring>
- <myDamp>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</myDamp>

- <myMass>0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
0.000E00</myMass>
<ForceOnSpring>-0.000 -0.000 -0.000 -0.000 -0.000
-0.000</ForceOnSpring>
<Reactions>0.000 0.000 0.000 0.000 0.000 0.000</Reactions>
<Angle>0.0</Angle>
</Node>
</NodeList>
- <MaterialList>
- <Material class="class model.discrete.material.Isotropic"
label="mat">
<Density>7.350E-04</Density>
<RayleighCoeficients>0.000E00
0.000E00</RayleighCoeficients>
<Elasticity>3.0E7</Elasticity>
<Poisson>0.3</Poisson>
</Material>
</MaterialList>
- <CrossSectionList>
- <CrossSection class="class model.discrete.crosssection.CrossSection" label="sec1">
<Area>10.0</Area>
<Thickness>1.0</Thickness>
</CrossSection>
- <CrossSection class="class model.discrete.crosssection.CrossSection" label="sec2">
<Area>8.0</Area>
<Thickness>1.0</Thickness>
</CrossSection>
- <CrossSection class="class model.discrete.crosssection.CrossSection" label="sec3">
<Area>6.0</Area>
<Thickness>1.0</Thickness>
</CrossSection>
</CrossSectionList>
- <AnalysisModelList>
<AnalysisModel class="class model.discrete.analysismodel.Line_2D" label="Line_2D"/>
</AnalysisModelList>
- <ElementList>
- <Element class="class model.discrete.element.ElmL2"
label="E1">
<Incidence>1 2</Incidence>
<IntegrationOrder>2 2 0</IntegrationOrder>
<MyMaterial>mat</MyMaterial>
<MyCrossSection>sec1</MyCrossSection>
<MyAnalysisModel>Line_2D</MyAnalysisModel>
- <GaussPointList>
- <GaussPointResults>
<GPCartesianCoords>x y z</GPCartesianCoords>
<GPStrains>epsilonXX</GPStrains>
<GPStresses>sigmaXX</GPStresses>
</GaussPointResults>
- <GaussPoint label="null">

<GPCartesianCoord>7.5000E01 1.0000E02 0.0000E00</GPCartesianCoord>
<GPStrain>2.755E-06</GPStrain>
<GPStress>8.266194E01</GPStress>
</GaussPoint>
- <GaussPoint label="null">
<GPCartesianCoord>7.5000E01 1.0000E02 0.0000E00</GPCartesianCoord>
<GPStrain>2.755E-06</GPStrain>
<GPStress>8.266194E01</GPStress>
</GaussPoint>
</GaussPointList>
- <NodalStressesAndStrains>
<NodeCoords>x y z</NodeCoords>
<NodeStrains>epsilonXX</NodeStrains>
<NodeStresses>sigmaXX</NodeStresses>
<Node label="1"/>
<NodeCoord>0.000E00 0.000E00
0.000E00</NodeCoord>
<NodeStrain>2.755E-06</NodeStrain>
<NodeStress>8.266E01</NodeStress>
<Node label="2"/>
<NodeCoord>1.500E02 2.000E02 0.000E00</NodeCoord>
<NodeStrain>2.755E-06</NodeStrain>
<NodeStress>8.266E01</NodeStress>
</NodalStressesAndStrains>
</Element>
- <Element class="class model.discrete.element.ElmL2"
label="E2">
<Incidence>2 3</Incidence>
<IntegrationOrder>2 2 0</IntegrationOrder>
<MyMaterial>mat</MyMaterial>
<MyCrossSection>sec2</MyCrossSection>
<MyAnalysisModel>Line_2D</MyAnalysisModel>
- <GaussPointList>
- <GaussPointResults>
<GPCartesianCoords>x y z</GPCartesianCoords>
<GPStrains>epsilonXX</GPStrains>
<GPStresses>sigmaXX</GPStresses>
</GaussPointResults>
- <GaussPoint label="null">
<GPCartesianCoord>1.5000E02 1.0000E02 0.0000E00</GPCartesianCoord>
<GPStrain>-7.410E-06</GPStrain>
<GPStress>-2.223060E02</GPStress>
</GaussPoint>
- <GaussPoint label="null">
<GPCartesianCoord>1.5000E02 1.0000E02 0.0000E00</GPCartesianCoord>
<GPStrain>-7.410E-06</GPStrain>
<GPStress>-2.223060E02</GPStress>
</GaussPoint>
</GaussPointList>
- <NodalStressesAndStrains>

<NodeCoords>x y z</NodeCoords>
<NodeStrains>epsilonXX</NodeStrains>
<NodeStresses>sigmaXX</NodeStresses>
<Node label="2"/>
<NodeCoord>1.500E02 2.000E02 0.000E00</NodeCoord>
<NodeStrain>-7.410E-06</NodeStrain>
<NodeStress>-2.223E02</NodeStress>
<Node label="3"/>
<NodeCoord>1.500E02 0.000E00 0.000E00</NodeCoord>
<NodeStrain>-7.410E-06</NodeStrain>
<NodeStress>-2.223E02</NodeStress>
</NodalStressesAndStrains>
</Element>
- <Element class="class model.discrete.element.ElmL2"
label="E3">
<Incidence>1 3</Incidence>
<IntegrationOrder>2 2 0</IntegrationOrder>
<MyMaterial>mat</MyMaterial>
<MyCrossSection>sec3</MyCrossSection>
<MyAnalysisModel>Line_2D</MyAnalysisModel>
- <GaussPointList>
- <GaussPointResults>
<GPCartesianCoords>x y z</GPCartesianCoords>
<GPStrains>epsilonXX</GPStrains>
<GPStresses>sigmaXX</GPStresses>
</GaussPointResults>
- <GaussPoint label="null">
<GPCartesianCoord>7.5000E01 0.0000E00 0.0000E00</GPCartesianCoord>
<GPStrain>2.585E-04</GPStrain>
<GPStress>7.755161E03</GPStress>
</GaussPoint>
- <GaussPoint label="null">
<GPCartesianCoord>7.5000E01 0.0000E00 0.0000E00</GPCartesianCoord>
<GPStrain>2.585E-04</GPStrain>
<GPStress>7.755161E03</GPStress>
</GaussPoint>
</GaussPointList>
- <NodalStressesAndStrains>
<NodeCoords>x y z</NodeCoords>
<NodeStrains>epsilonXX</NodeStrains>
<NodeStresses>sigmaXX</NodeStresses>
<Node label="1"/>
<NodeCoord>0.000E00 0.000E00
0.000E00</NodeCoord>
<NodeStrain>2.585E-04</NodeStrain>
<NodeStress>7.755E03</NodeStress>
<Node label="3"/>
<NodeCoord>1.500E02 0.000E00 0.000E00</NodeCoord>
<NodeStrain>2.585E-04</NodeStrain>
<NodeStress>7.755E03</NodeStress>

```
</NodalStressesAndStrains>
</Element>
</ElementList>
<GlobalAnalysisModel>Line_2D</GlobalAnalysisModel>
</FemModel>
</Driver>
</JMefData>
```

A3 COMPARATIVO DE MODELOS COM MATRIZ DE MASSA GEOMETRICAMENTE DISTRIBUÍDA E COM DISTRIBUIÇÃO CONSISTENTE

Considere as equações (2.12) e (2.13) aqui repetidas:

$$\mathbf{M}^{\mathbf{e}} = \left[m_{ij}\right]^{e} = \int_{\Gamma} \rho \mathbf{N}^{\mathbf{T}} \mathbf{N} d\Gamma \tag{2.12}$$

$$\mathbf{K}^{\mathbf{e}} = \left[k_{ij}\right]^{e} = \int_{\Gamma} \mathbf{B}^{\mathbf{T}} \mathbf{D} \mathbf{B} d\Gamma \tag{2.13}$$

Essas equações determinam a distribuição de massa e rigidez para sistemas discretizados pelo MEF. Escrevendo-se:

$$\frac{d}{dt}\left(\dot{\mathbf{u}}\mathbf{M}^{\mathbf{e}}\dot{\mathbf{u}}^{\mathbf{T}} + \mathbf{u}\mathbf{K}^{\mathbf{e}}\mathbf{u}^{\mathbf{T}}\right) = \frac{dV}{dt} = 0 \tag{A.1}$$

onde V é a energia mecânica total do sistema, é possível notar um significado físico claro para as equações (2.12) e (2.13). Essas equações distribuem a massa e a rigidez entre os graus de liberdade do sistema de maneira a minimizar a energia potencial mecânica do sistema discreto.

Nessas equações, se $\mathbf{B} = \mathbf{N}'$, diz-se que a distribuição de massa é consistente. Porém, essa não é a única possibilidade, uma vez que ainda é possível se resolver a Eq. (A.1) mesmo quando se adota distribuições de massa e rigidez não consistentes. Em dinâmica de estruturas, é muito comum se utilizar métodos de distribuição de massa geométricos. Na prática, à medida que se refina a malha do modelo discreto, a distribuição geométrica tende a aproximar melhor a distribuição consistente.

Na seção 5.3 resolveu-se um problema de deslocamento axial de uma barra com um único elemento L3. Ao longo da implementação do INSANE, adotaram-se sempre as mesmas funções de interpolação para determinar massa, rigidez e amortecimento. Portanto, os resultados obtidos no Capítulo 5 são para distribuições consistentes de massa.

A seguir, faz-se um comparativo dos resultados obtidos a partir de a três modelos com elementos L2 e massas concentradas conforme ilustrado na Figura A.1. Solucionou-se o mesmo problema 5.3 para três modelos - com 2, 4 e 6 elementos L2 – utilizando-se os mesmos parâmetros: $\rho = 30$, $L = 1$, $A = 1$ e $E = 3$.

As soluções analíticas para as freqüências e os modos da barra uniforme são, respectivamente:

$$\omega_i = \frac{(2i-1)\pi}{2L}\sqrt{\frac{E}{\rho}} \tag{A.2}$$

$$\phi_i(x) = c_i \,\mathrm{sen}\left((2i-1)\frac{\pi x}{L}\right) \tag{A.3}$$

onde i é o número do modo e c_i é uma constante arbitrária.

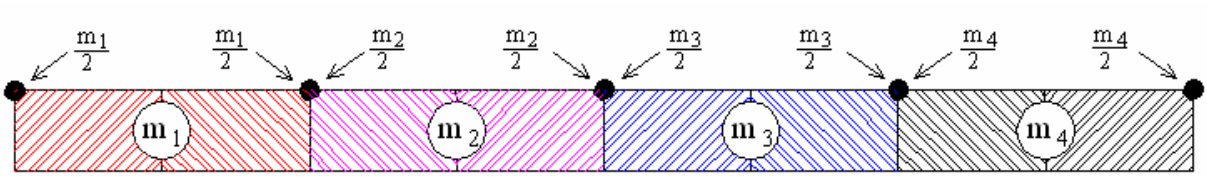


FIGURA A.1: Distribuição geométrica da massa para quatro elementos L2

Obteve-se a solução modal pelo INSANE com precisão de $1x10^{-8}$ para cada um desses modelos. A TABELA A.1 mostra o as freqüências encontradas e o erro encontrado em relação a Eq.(A.2). Na Figura A.2, comparou-se o modo fundamental obtido pelo INSANE e o modo dado pela Eq. (A.3). Na Figura b, é mostrado o erro da solução via MEF comparado com a solução analítica.

Como se pode notar, mesmo com apenas 2 elementos L2, já se obtém uma aproximação muito boa para o primeiro modo. Com 6 elementos L2, obtém-se uma aproximação melhor para a freqüência fundamental do que com um único elemento L3.

TABELA A.1: Comparativo entre freqüências naturais obtidas pelo INSANE e analiticamente

	Analítica	2 Elementos L2	4 Elementos L2	6 Elementos L2	1 Elemento L3 e massa consistente.
ω_1	0,4967	0,4841	0,5019	0,4953	0,4986
Erro (%)	-	2,53	1,05	0,28	0,38

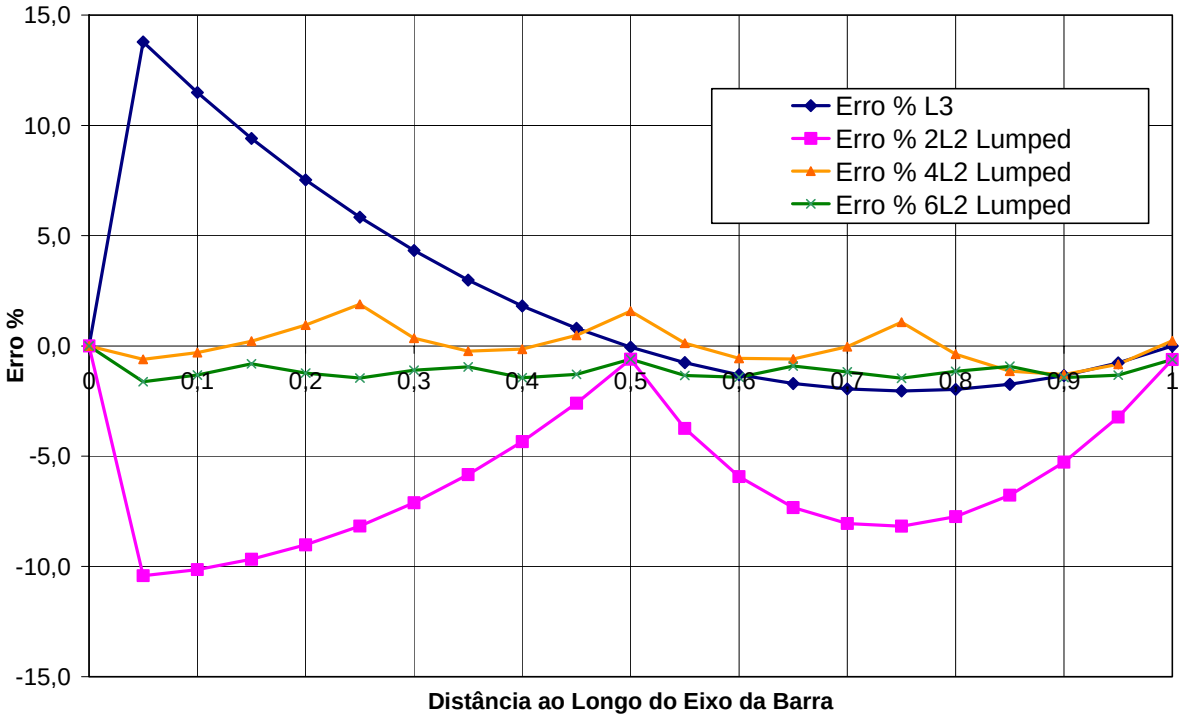
O sistema contínuo funciona como infinitas molas em série, sendo que cada modo natural constitui uma dessas molas. Os modos naturais são os mecanismos pelos quais o sistema consegue absorver trabalho externo. Dessa maneira, a solução analítica para o deslocamento axial da barra homogênea para um carregamento harmônico será dada por:

$$u(x,t) = \frac{E}{A\rho^2 L^2} S_N \tag{A.4}$$

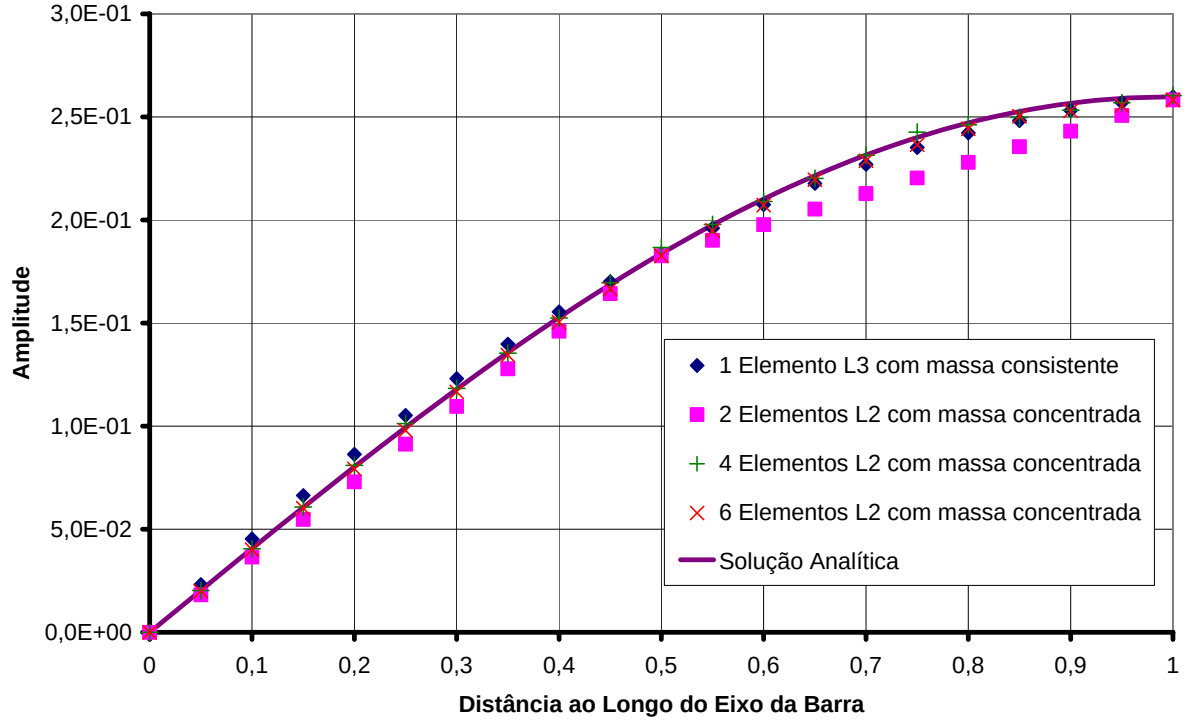
$$S_N = \sum_{i=1}^{N} (-1)^{i+1} \operatorname{sen}\left((2i-1)\pi \frac{x}{L} \right) \frac{(2i-1)^2 \pi^2}{2} \frac{\operatorname{sen}\Omega t - r \operatorname{sen}\omega_i t}{1-r^2} \tag{A.5}$$

Na Figura A.3(a), comparou-se a integração no tempo utilizando o método de superposição de acelerações truncada no primeiro modo, e a solução analítica dada utilizando-se os 20 primeiros modos. Considerou-se um carregamento harmônico unitário com Ω = 0,1 na extremidade da barra. Pode-se observar uma boa concordância entre a solução analítica e as soluções do MEF, seja com elementos com massa consistente ou com massas concentradas.

Na Figura A.3(b), compara-se o erro obtido em relação à solução analítica. Nota-se claramente que os elementos com distribuição geométrica de massa apresentam melhores resultados à medida que se refina a malha. Os grandes erros iniciais em todos os modelos podem ser explicados devido a se ter pequenos deslocamentos. Assim pequenas variações provocam grandes variações relativas.

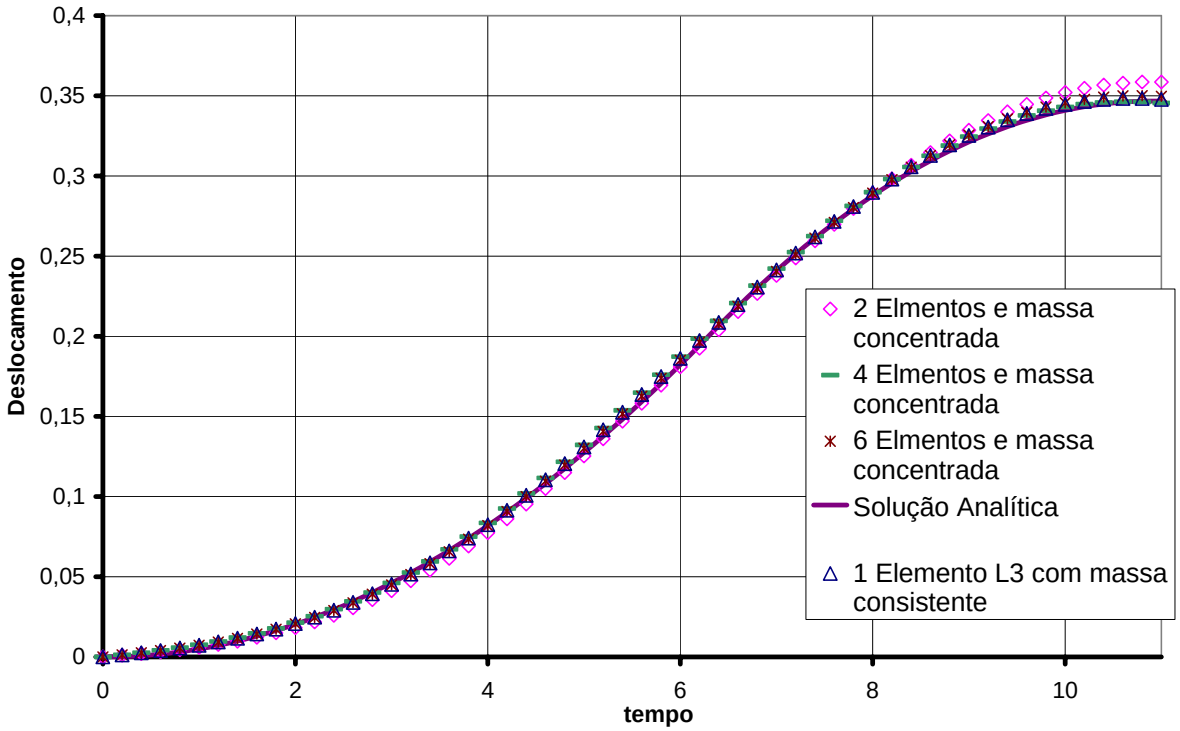


(a)

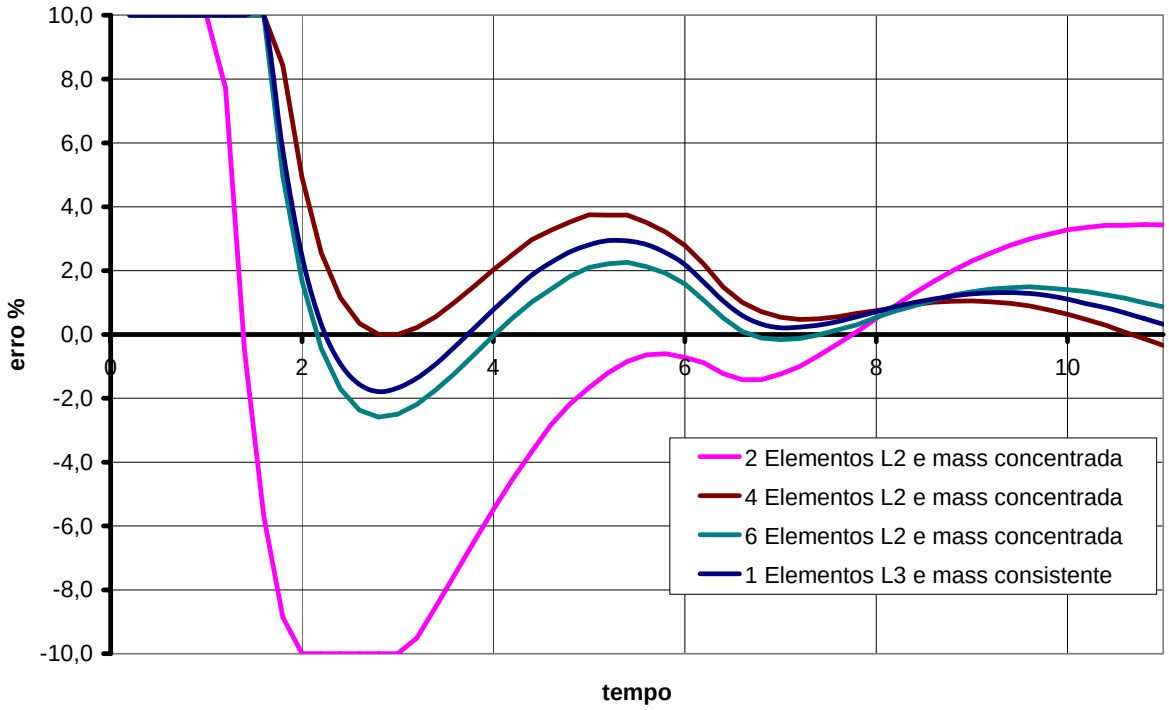


(b)

FIGURA A.2: (a) Modo fundamental da barra homogênea para diversos modelos e (b) erro percentual no cálculo do modo fundamental no MEF ao longo do eixo da barra



(a)



(b)

FIGURA A.3: (a) Deslocamento na ponta da barra homogênea e (b) erro no deslocamento para modelagens da barra homogênea com massa consistente e distribuição geométrica