APLICAÇÃO ORIENTADA A OBJETOS PARA ANÁLISE FÍSICAMENTE NÃO-LINEAR COM MODELOS RETICULADOS DE SEÇÕES TRANSVERSAIS COMPOSTAS

Marcos Torres da Fonseca

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ESTRUTURAS

"APLICAÇÃO ORIENTADA A OBJETOS PARA ANÁLISE FÍSICAMENTE NÃO-LINEAR COM MODELOS RETICULADOS DE SEÇÕES TRANSVERSAIS COMPOSTAS"

Marcos Torres da Fonseca

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Estruturas da Escola de Engenharia da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do título de "Mestre em Engenharia de Estruturas".

Comissão Examinadora:

_____
Prof. Dr. Roque Luiz da Silva Pitangueira
DEES - UFMG - (Orientador)

_____
Prof. Dr. Alcebíades de Vasconcellos Filho
DEES - UFMG

_____
Prof. Dr. Armando Cesar Campos Lavall
DEES-UFMG

_____
Prof. Dr. Rogério José Marczak
UFRGS

Belo Horizonte, 06 de outubro de 2006

Para ser grande, sê inteiro: nada
Teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.
Assim em cada lago a lua toda
Brilha, porque alta vive.

*(*Fernando Pessoa*)*

Dedico este trabalho a meus pais e avós.

Índice

Lista de Tabelas

Lista de Figuras

Resumo

Esta dissertação de mestrado refere-se à implementação computacional, segundo o paradigma orientado a objetos, da solução de modelos estruturais reticulados fisicamente não-lineares, com seções transversais de qualquer geometria e compostas por vários materiais, através do Método dos Elementos Finitos.

Estas seções são representadas através de uma decomposição em várias áreas menores, sendo monitoradas em cada uma as relações tensão-deformação não-lineares dos materiais. Desta forma, é possível determinar a equação constitutiva de cada seção, a matriz de rigidez e os esforços internos do elemento finito. São então obtidas, por meio de um processo incremental-iterativo, as trajetórias de equilíbrio para determinados graus de liberdade do modelo.

As formulações para os modelos matemáticos e discretos, baseados nas teorias de flexão de Timoshenko e Euler-Bernoulli, são apresentadas, considerando-se o tipo mais geral de modelo estrutural reticulado, o pórtico espacial. Discute-se ainda, brevemente, a solução de equações não-lineares de equilíbrio.

Através da utilização de várias soluções tecnológicas para desenvolvimento de software, a referida implementação é feita com a progressiva fatoração e ampliação do núcleo numérico do **INSANE** (*INteractive Structural ANalysis Environment*), um ambiente computacional segmentado e amigável a mudanças. A concepção do núcleo numérico e o conjunto de segmentos que representam as diversas abstrações necessárias a uma resolução numérica de um modelo de elementos finitos são discutidos.

Várias simulações numéricas são apresentadas de maneira a validar a implementação e mostrar os potenciais de modelagem da aplicação.

Abstract

This master's thesis refers to the computational implementation, according to object-oriented paradigm, of material nonlinear solution of finite element models of framed structures with composed and geometrically arbitrary cross sections.

These sections are represented by decomposition in smaller areas, in which the materials' nonlinear stress-strain relations are monitored. In such way, it is possible to determine the constitutive equation of each section, the stiffness matrix and the internal efforts of the finite element. Then, the equilibrium paths for certain degrees of freedom of the model are obtained through an incremental-iterative procedure.

The mathematical and discrete models formulations, based on Timoshenko and Euler-Bernoulli bending theories, are presented, considering the most general framed structural model, the space frame. The nonlinear solution of equilibrium equations is discussed briefly.

Through the use of various software development solutions, the related implementation takes place with the progressive rearrangement and extension of the numerical nucleus of **INSANE** (*INteractive Structural ANalysis Environment*), a computational environment segmented and friendly to changes. The numerical nucleus conception and the set of segments that represent the necessary abstractions to a numerical resolution of a finite element model are discussed.

Some numerical simulations are presented to validate the implementation and to show the application modeling potential.

Agradecimentos

A *DEUS* por sempre iluminar minha vida e colocar à minha volta pessoas tão especiais.

A *meus pais* pelo amor e apoio incondicionais, pelos conselhos e momentos de paz, e por me propiciarem condições de chegar até aqui.

A *meus avós* que tanto amo, eternos exemplos de vida e valores.

À *minha família* por sempre acreditar na realização desta conquista.

À *Flávia*, minha motivação maior, pelo incentivo constante, cheio de amor, paciência e carinho.

Ao professor *Roque Luiz da Silva Pitangueira*, orientador e amigo, agradeço por me apresentar a programação orientada a objetos e por ser profissional exemplar, sempre disponível e empolgado.

Aos *Insanos* pela amizade e pelas divertidas horas frente ao computador.

Aos *meus amigos* que compreenderam minha ausência nos momentos de festa e eventos futebolísticos, e que sempre me apoiaram nesta conquista.

Aos *professores* e *funcionários* do Departamento de Engenharia de Estruturas da $UFMG$ pela disponibilidade e atenção em todos os momentos.

A todos aqueles que de alguma forma contribuíram para a realização deste trabalho.

À *CAPES* pelo apoio financeiro.

Capítulo 1

INTRODUÇÃO

Em mecânica estrutural, um problema é dito *não-linear* quando a rigidez depende dos deslocamentos da estrutura. Esta dependência é dita *fisicamente* não-linear quando a rigidez é afetada pela resposta do material ao estado de deformação a que o mesmo está submetido.

Uma poderosa ferramenta para a análise fisicamente não-linear de estruturas é o Método dos Elementos Finitos, que permite o modelamento de diferentes estruturas sob o efeito de diferentes solicitações e restrições. Ao empregar-se esta difundida ferramenta de análise estrutural, deve-se atentar para a necessidade de modelos constitutivos capazes de descrever o comportamento dos materiais e para a necessidade da obtenção de trajetórias de equilíbrio que possam predizer a resposta da estrutura até e além da falha de componentes individuais.

A formulação de elementos finitos unidimensionais é uma alternativa amplamente reconhecida para a aproximação da resposta de estruturas reticuladas, bastante comuns em problemas de engenharia. Nestas estruturas, deve-se incluir, quando necessário, os efeitos das deformações de cisalhamento, pois podem ser significativos na análise de seu comportamento.

No caso unidimensional do Método dos Elementos Finitos é possível representar seções transversais compostas e de qualquer geometria decompondo-as em várias áreas menores. Em cada uma dessas áreas são monitoradas as relações tensão-deformação não-lineares dos materiais e, por meio de um somatório, simplifica-se a integração dos esforços, estimando-se a resposta da seção ao estado de deformação existente. Encontradas as deformações, pode ser determinada a equação constitutiva de cada seção transversal e a matriz de rigidez de cada elemento. Podem então ser avaliadas as forças internas e obtidas as trajetórias de equilíbrio.

A resolução de problemas fisicamente não-lineares via Método dos Elementos Finitos com modelos estruturais reticulados de seções transversais compostas e de qualquer geometria envolve conceitos freqüentemente encontrados em análises mais gerais e complexas. Assim, desenvolver um software amigável à mudanças, que atenda desde modelos mais simples até modelos extremamente sofisticados de elementos finitos, sem ter que recomeçar o processo a cada novo aperfeiçoamento, é um desafio na área de métodos computacionais aplicados à mecânica estrutural. Enfrentar este desafio é imperativo para que a pesquisa na área disponha de recursos computacionais que, a partir de conceitos já consolidados, possibilitem o aprimoramento progressivo dos modelos através da ampliação de complexidades.

Algumas iniciativas de desenvolvimento de software pela comunidade acadêmica ao longo do tempo resultaram em produtos dependentes de sistema operacional, pouco amigáveis, escritos em linguagens de programação não apropriadas, de expansão, distribuição e manutenção difíceis, desenvolvidos por equipes fechadas, com documentação deficiente, entre outras limitações. Isto pode ser creditado à falta de disposição da comunidade em se apropriar das tecnologias emergentes ou mesmo à inexistência das mesmas. Esta constatação confronta-se com o surgimento e aprimoramento de recursos para desenvolvimento de software, como o paradigma de programação orientada a objetos, linguagem Java, XML (*eXtensible Markup Language*), UML (*Unified Modelling Language*), CVS (*Concurrent Version System*), testes unitários, padrões de projeto de software, entre outras.

Portanto, o desafio de desenvolver sistemas computacionais utilizando estes recursos é condição obrigatória para o aprimoramento, com agilidade e criatividade, da pesquisa na área da mecânica computacional.

1.1 Objetivos do Trabalho

1.1.1 Objetivos Gerais

As possibilidades oferecidas pelos recursos tecnológicos para desenvolvimento de software constituem amplo campo de pesquisa na área de métodos numéricos e computacionais aplicados à engenharia.

O domínio destes recursos e a aplicação dos mesmos no aprimoramento progressivo dos modelos de análise estrutural requerem um ambiente computacional segmentado, amigável a mudanças e escalável em complexidade, como proposto pelo programa **INSANE** (*INteractive Structural ANalysis Environment*), desenvolvido no Departamento de Engenharia de Estruturas da Escola de Engenharia da Universidade Federal de Minas Gerais e disponível em `http://www.dees.ufmg.br/insane`.

O ambiente computacional **INSANE** é constituído de três grandes aplicações: pré-processador, processador e pós-processador, todas implementadas em linguagem Java. O pré e o pós-processador são aplicações gráficas interativas que disponibilizam, respectivamente, ferramentas de pré e pós-processamento de diferentes modelos discretos. O processador é a aplicação que representa o núcleo numérico do sistema e é a responsável pela obtenção dos resultados de diferentes modelos discretos de análise estrutural.

Cada uma destas aplicações é implementada segundo o paradigma de programação orientada a objetos, que é uma técnica de programação baseada em classes e objetos. Os atributos e métodos são encapsulados nos objetos, sendo eles intimamente amarrados entre si e com a propriedade de ocultar informações. Ou seja, apesar de se comunicarem uns com os outros através de interfaces bem definidas, geralmente os objetos não têm permissão para conhecer como outros objetos estão implementados.

Isto permite que os programas possam ser divididos em módulos independentes, possibilitando o trabalho em conjunto de diversas pessoas em diferentes locais e épocas. Desta forma, a manutenção e expansão através da constante fatoração progressiva do código são facilmente realizadas em um programa desenvolvido com programação orientada a objetos. E com auxílio

dos recursos disponíveis para automação, gerenciamento, teste e controle de versões no desenvolvimento de software, é possível desenvolver de forma colaborativa e dinâmica um programa estável, confiável, com qualidade e, principalmente, útil para a engenharia estrutural.

É no projeto de expansão do programa **INSANE** que este trabalho se insere, com a progressiva fatoração e ampliação do núcleo numérico do sistema, apoiadas na implementação da formulação paramétrica de elementos finitos existente no programa, desenvolvida por Almeida (2005), e demais implementações existentes, desenvolvidas por Gonçalves (2004), Pitangueira et al. (2004) e Fuina (2006). A partir de conceitos já consolidados, complexidades são ampliadas sem ter que recomeçar todo o processo a cada novo aperfeiçoamento.

1.1.2 Objetivos Específicos

A dissertação que aqui se apresenta refere-se à implementação computacional no núcleo numérico do programa **INSANE** da resolução através do Métodos dos Elementos Finitos de problemas fisicamente não-lineares de modelos estruturais reticulados, baseados nas teorias de flexão de Timoshenko e de Euler-Bernoulli, com seções transversais de qualquer geometria e compostas por vários materiais.

A resolução destes problemas envolve a definição e a discretização de uma seção de geometria arbitrária e vários materiais, a obtenção da matriz constitutiva da seção e a solução não-linear de problemas de elementos finitos paramétricos unidimensionais.

1.2 Organização do Texto

Este trabalho está organizado em 7 capítulos.

No Capítulo 2 é apresentada a formulação matemática de pórtico espacial, que é a mais geral das estruturas reticuladas, com seções transversais compostas e de geometria qualquer. Considera-se primeiro a teoria de flexão de Timoshenko e em seguida a teoria de flexão de Euler-Bernoulli.

Em seguida, são apresentadas no Capítulo 3 as formulações para os modelos discretos de

elementos finitos paramétricos unidimensionais para pórtico espacial, considerando as formulações matemáticas vistas no capítulo anterior.

O Capítulo 4 trata da análise fisicamente não-linear de estruturas. Este capítulo inicia-se apresentando exemplos de leis constitutivas não-lineares de alguns materiais que são adotados posteriormente nas simulações numéricas.

Capítulo 2

MODELOS MATEMÁTICOS DE PÓRTICO ESPACIAL

2.1 Introdução

Segundo Weaver, Jr. e Gere (1980), uma estrutura reticulada é formada por barras prismáticas, as quais têm um eixo reto e são longas em comparação à sua seção transversal. Os pontos de interseção destas barras, assim como os pontos de apoio das barras de extremidades livres, são os nós da estrutura reticulada.

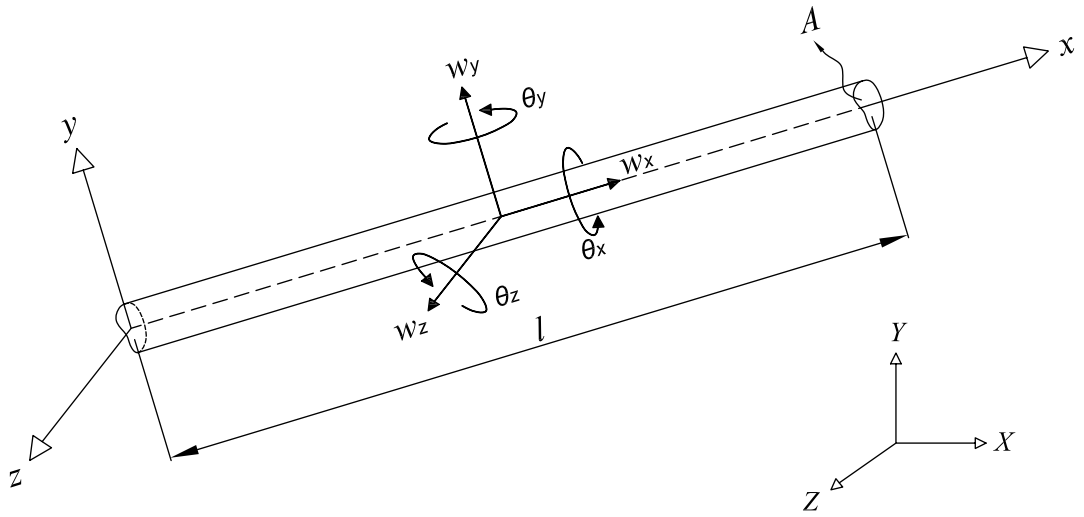


Figura 2.1: Sistema de coordenadas e deslocamentos de uma barra de pórtico espacial.

Um pórtico espacial é o tipo mais geral de estrutura reticulada, no que diz respeito tanto a geometria quanto a cargas. A Figura 2.1 mostra uma barra, de comprimento l e seção transversal de área A, orientada de maneira arbitrária no espaço. A orientação de seu sistema

local de coordenadas é de tal forma que os planos xy e xz são os planos principais de flexão. Considera-se que a seção transversal tem dois eixos coincidentes com os eixos locais y e z, de modo que a flexão e a torção tomem lugar independentemente uma da outra.

Uma barra típica de pórtico espacial pode suportar forças axiais, forças cortantes nas direções principais, momentos de torção e momentos fletores nas direções principais. Um ponto qualquer do eixo da barra pode ter até seis deslocamentos, sendo três translações (w_x, w_y e w_z) e três rotações (θ_x, θ_y e θ_z), conforme Figura 2.1.

Portanto, na análise de um pórtico espacial, consideram-se deformações axiais, deformações devido à flexão e devido à torção. As deformações devido aos esforços cortantes muitas vezes são desprezadas, mas podem ser significativas e devem ser incluídas na análise de tais estruturas, quando necessário.

A incorporação dos efeitos relacionados ao cisalhamento pode ser feita através da consideração da teoria de flexão de vigas de Timoshenko, a qual inclui uma aproximação para a distorção da seção transversal devido aos esforços cortantes. Esforços estes desprezados na teoria clássica de flexão de vigas esbeltas de Euler-Bernoulli.

Neste capítulo, é apresentada a formulação de pórtico espacial considerando, em primeiro lugar, a teoria de Timoshenko e, em seguida, a formulação considerando a teoria de Euler-Bernoulli. É considerada ainda a não-linearidade dos materiais da estrutura e o fato das seções das barras serem compostas e de geometria qualquer.

2.2 Pórtico Espacial considerando a Teoria de Timoshenko

2.2.1 Hipóteses

Para um pórtico espacial considerando a teoria de Timoshenko são adotadas as seguintes hipóteses:

1. Seções transversais normais ao eixo da barra antes da flexão, permanecem planas, mas não permanecem necessariamente ortogonais a tal eixo, depois da flexão;

2. A barra suporta deformação axial, deformações devido ao cisalhamento nas duas direções principais, deformação devido à torção e deformações devido à flexão nas duas direções principais;

3. Admite-se constante a distribuição de tensões tangenciais ao longo da seção transversal, porém modificada por um fator de correção de cisalhamento α da seção, de maneira que o trabalho da deformação tangencial constante coincida com o trabalho da deformação exata.

4. Existe uma completa interação entre os vários materiais da barra na seção transversal. As deformações nas interfaces dos materiais são consideradas compatíveis.

2.2.2 Campo de Deslocamentos

Seja a barra de pórtico espacial submetida a cargas no plano xy, a cargas no plano xz e submetida a esforços axiais e de torção. O comportamento da barra, antes e depois de deformada, está representado nas Figuras 2.2, 2.3 e 2.4.

A partir das hipóteses adotadas para a teoria de Timoshenko e da observação das Figuras 2.2, 2.3 e 2.4, pode-se obter o seguinte campo de deslocamentos para um ponto qualquer da barra:

$$u_x(x,y,z) \;=\; w_x(x) - y \cdot \theta_z(x) + z \cdot \theta_y(x) \tag{2.1}$$

$$u_y(x,y,z) \;=\; w_y(x) - z \cdot \theta_x(x) \tag{2.2}$$

$$u_z(x,y,z) \;=\; w_z(x) + y \cdot \theta_x(x) \tag{2.3}$$

2.2.3 Relações Deformações-Deslocamentos

A partir do campo de deslocamentos descrito e considerando a teoria da elasticidade clássica para pequenos deslocamentos e deformações, obtêm-se as seguintes relações entre deformações e deslocamentos para um ponto qualquer da barra:

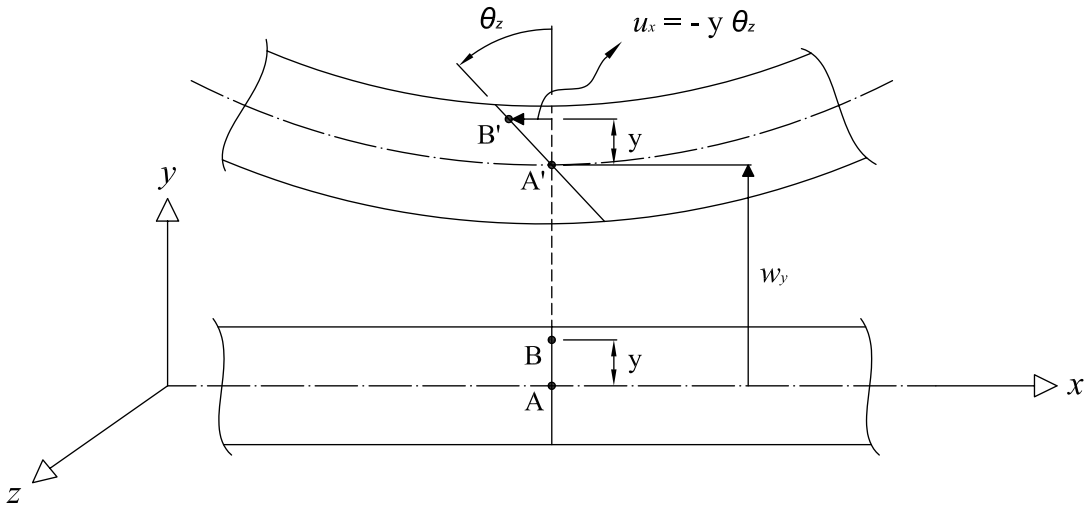


Figura 2.2: Seção antes e depois da flexão no plano *xy*.

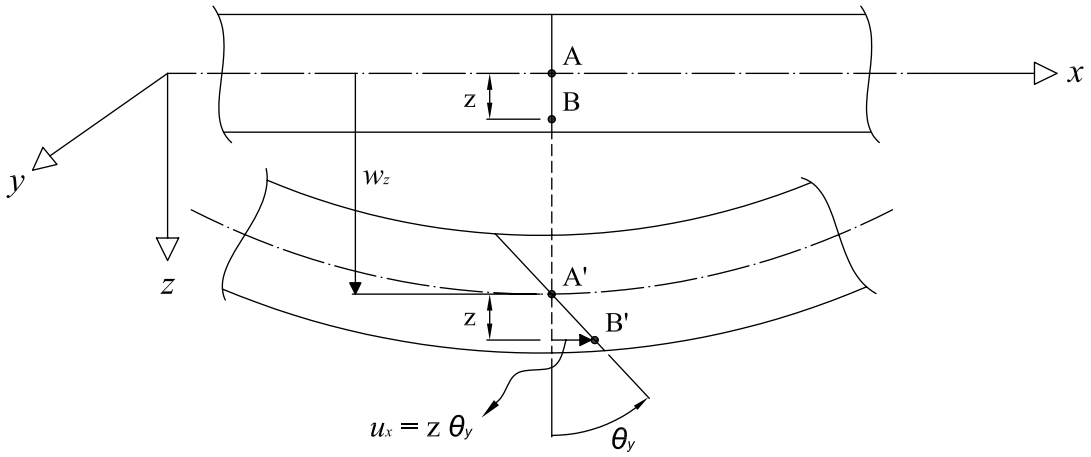


Figura 2.3: Seção antes e depois da flexão no plano *xz*.

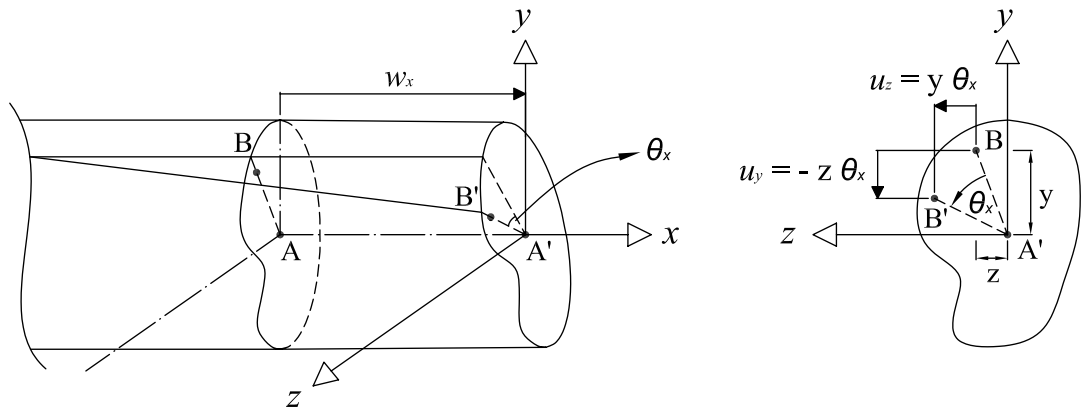


Figura 2.4: Seção antes e depois da deformação axial e de torção.

$$\varepsilon_{xx} \;=\; \frac{\partial\, u_x(x,y,z)}{\partial\, x} \;=\; \frac{\partial\, w_x}{\partial\, x} - y\,\frac{\partial\, \theta_z}{\partial\, x} + z\,\frac{\partial\, \theta_y}{\partial\, x} \tag{2.4}$$

$$\gamma_{xy} \;=\; \frac{\partial\, u_y(x,y,z)}{\partial\, x} + \frac{\partial\, u_x(x,y,z)}{\partial\, y} \;=\; \frac{\partial\, w_y}{\partial\, x} - z\,\frac{\partial\, \theta_x}{\partial\, x} - \theta_z \tag{2.5}$$

$$\gamma_{xz} \;=\; \frac{\partial\, u_z(x,y,z)}{\partial\, x} + \frac{\partial\, u_x(x,y,z)}{\partial\, z} \;=\; \frac{\partial\, w_z}{\partial\, x} + y\,\frac{\partial\, \theta_x}{\partial\, x} + \theta_y \tag{2.6}$$

Definindo-se as deformações generalizadas axial ε_a, de cisalhamento γ_y e γ_z, de torção ψ e de flexão (curvaturas) κ_y e κ_z como:

$$\varepsilon_a = \frac{\partial\, w_x}{\partial\, x}, \quad \gamma_y = \frac{\partial\, w_y}{\partial\, x} - \theta_z, \quad \gamma_z = \frac{\partial\, w_z}{\partial\, x} + \theta_y, \quad \psi = \frac{\partial\, \theta_x}{\partial\, x}, \quad \kappa_y = \frac{\partial\, \theta_y}{\partial\, x} \quad e \quad \kappa_z = \frac{\partial\, \theta_z}{\partial\, x}, \tag{2.7}$$

obtém-se então que:

$$\varepsilon_{xx} \;=\; \varepsilon_a - y \cdot \kappa_z + z \cdot \kappa_y \tag{2.8}$$

$$\gamma_{xy} \;=\; \gamma_y - z \cdot \psi \tag{2.9}$$

$$\gamma_{xz} \;=\; \gamma_z + y \cdot \psi \tag{2.10}$$

2.2.4 Relações Tensões-Deformações

Admitindo-se que a relação entre tensões e deformações dos materiais é única e não depende da geometria da estrutura, tal relação é dita sua lei constitutiva, uma propriedade do material que deve ser prescrita como um dado externo à análise.

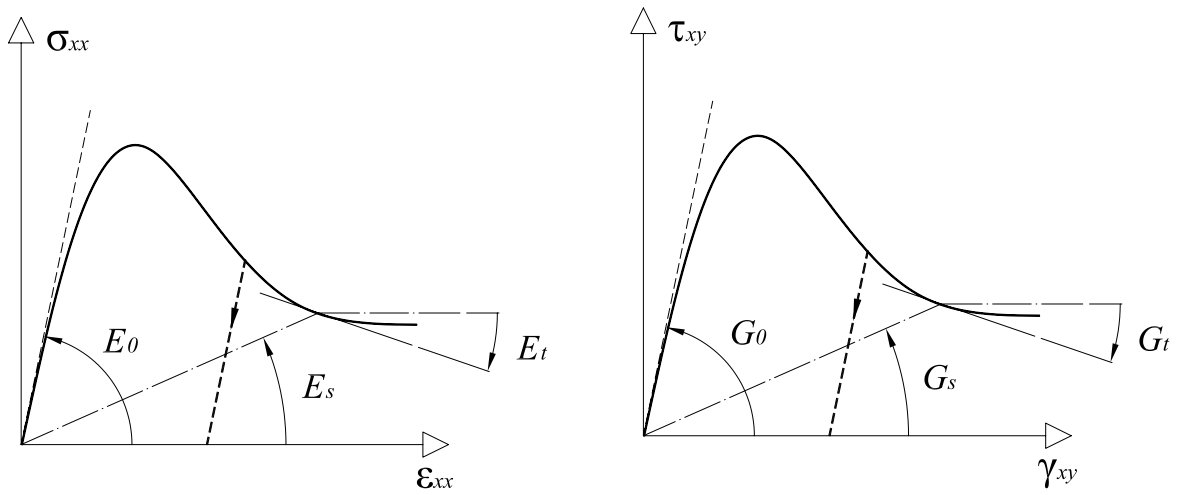
Considerando materiais cujas leis constitutivas para tensões normais e tangenciais são não-lineares, como as representadas na Figura 2.5, têm-se as seguintes relações entre tensões e deformações para um ponto qualquer da barra:

$$\sigma_{xx} = E_s(\varepsilon_{xx}) \cdot \varepsilon_{xx} \tag{2.11}$$

$$\tau_{xy} = G_s(\gamma_{xy}) \cdot \gamma_{xy} \tag{2.12}$$

$$\tau_{xz} = G_s(\gamma_{xz}) \cdot \gamma_{xz} \tag{2.13}$$

onde E_s e G_s são, respectivamente, os módulos de elasticidade secantes longitudinal e transversal do material (Figura 2.5).



(a) Tensão Normal (b) Tensão Tangencial

Figura 2.5: Exemplos de leis constitutivas não-lineares.

Relações incrementais entre tensões e deformações também podem ser escritas na forma:

$$d\sigma_{xx} = E_t(\varepsilon_{xx}) \cdot d\varepsilon_{xx} \tag{2.14}$$

$$d\tau_{xy} = G_t(\gamma_{xy}) \cdot d\gamma_{xy} \tag{2.15}$$

$$d\tau_{xz} = G_t(\gamma_{xz}) \cdot d\gamma_{xz} \tag{2.16}$$

onde E_t e G_t são, respectivamente, os módulos de elasticidade tangentes longitudinal e transversal do material (Figura 2.5).

2.2.5 Esforços Internos

A integração das tensões leva aos esforços internos nas seções da barra (força axial N, os esforços cortantes V_y e V_z, momento de torção T e momentos fletores M_y e M_z) dados por:

$$N = \int_A \sigma_{xx} \; dA \tag{2.17}$$

$$V_y = \int_A \tau_{xy} \; dA \tag{2.18}$$

$$V_z = \int_A \tau_{xz} \; dA \tag{2.19}$$

$$T = \int_A (\tau_{xz} \cdot y - \tau_{xy} \cdot z) \; dA \tag{2.20}$$

$$M_y = \int_A \sigma_{xx} \cdot z \; dA \tag{2.21}$$

$$M_z = \int_A -\sigma_{xx} \cdot y \; dA \tag{2.22}$$

Substituindo as Equações de 2.8 a 2.10 nas Equações de 2.11 a 2.13, e o resultado nas equações acima, tem-se que:

$$N = \int_A (\varepsilon_a - y \cdot \kappa_z + z \cdot \kappa_y) \cdot E_s \; dA \tag{2.23}$$

$$V_y = \int_A (\alpha_y \gamma_y - z \cdot \psi) \cdot G_s \; dA \tag{2.24}$$

$$V_z = \int_A (\alpha_z \gamma_z + y \cdot \psi) \cdot G_s \; dA \tag{2.25}$$

$$T = \int_A [(\gamma_z + y \cdot \psi) \cdot y - (\gamma_y - z \cdot \psi) \cdot z \,] \cdot G_s \; dA \tag{2.26}$$

$$M_y = \int_A (\varepsilon_a - y \cdot \kappa_z + z \cdot \kappa_y) \cdot E_s \cdot z \; dA \tag{2.27}$$

$$M_z = \int_A -(\varepsilon_a - y \cdot \kappa_z + z \cdot \kappa_y) \cdot E_s \cdot y \; dA \tag{2.28}$$

Nas Equações 2.24 e 2.25, pela hipótese (*3*), foram acrescentados os fatores de correção de cisalhamento α_y e α_z para as duas direções principais da seção transversal.

Para uma seção composta de vários materiais, distribuídos ao longo de uma área de geometria qualquer, é possível simplificar a integração dos esforços ao longo da seção decompondo-a em $\boldsymbol{q}$ pequenas áreas (Galgoul (1979), Sfakianakis (2001), Romero et al. (2002)), como na Figura 2.6. Este tipo de aproximação permite que, a partir do monitoramento das tensões e deformações em cada uma destas áreas, se aproxime a resposta da seção transversal pela soma de suas contribuições. É possível então obter os esforços internos por:

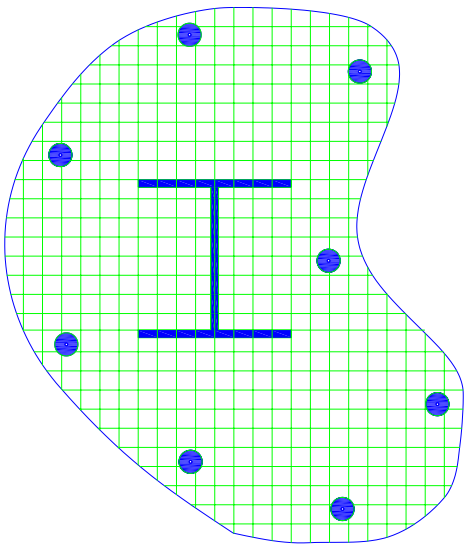


Figura 2.6: Decomposição da seção transversal.

$$N = \sum_{i=1}^{q} A_i \cdot (\varepsilon_a - y_i \cdot \kappa_z + z_i \cdot \kappa_y) \cdot E_{si} \tag{2.29}$$

$$V_y = \sum_{i=1}^{q} A_i \cdot (\alpha_y \gamma_y - z_i \cdot \psi) \cdot G_{si} \tag{2.30}$$

$$V_z = \sum_{i=1}^{q} A_i \cdot (\alpha_z \gamma_z + y_i \cdot \psi) \cdot G_{si} \tag{2.31}$$

$$T = \sum_{i=1}^{q} A_i \cdot [(\gamma_z + y_i \cdot \psi) y_i - (\gamma_y - z_i \cdot \psi) z_i] \cdot G_{si} \tag{2.32}$$

$$M_y = \sum_{i=1}^{q} A_i \cdot (\varepsilon_a - y_i \cdot \kappa_z + z_i \cdot \kappa_y) \cdot E_{si} \cdot z_i \tag{2.33}$$

$$M_z = -\sum_{i=1}^{q} A_i \cdot (\varepsilon_a - y_i \cdot \kappa_z + z_i \cdot \kappa_y) \cdot E_{si} \cdot y_i \tag{2.34}$$

2.3 Pórtico Espacial considerando a Teoria de Euler-Bernoulli

2.3.1 Hipóteses

Para um pórtico espacial considerando a teoria de Euler-Bernoulli, assumem-se as seguintes hipóteses:

1. Seções transversais normais ao eixo da barra antes da flexão, permanecem planas e ortogonais a tal eixo depois da flexão;

2. A barra suporta deformação axial, deformação devido à torção, e deformações devido à flexão nas duas direções principais;

Estas diferem das hipóteses feitas para a teoria de Timoshenko (Seção 2.2.1) quanto à ortogonalidade das seções e quanto à consideração de deformações de cisalhamento. Portanto, não há necessidade de uma hipótese equivalente à hipótese (*3*) feita para aquela teoria. Assume-se ainda como válida a hipótese (*4*).

2.3.2 Campo de Deslocamentos

Seja a barra de pórtico espacial submetida a cargas no plano xy, a cargas no plano xz, e submetida a esforços axiais e de torção. O seu comportamento antes e depois de deformada é semelhante ao mostrado nas Figuras 2.2, 2.3 e 2.4 referentes à teoria de Timoshenko.

No entanto, pela hipótese (*1*) para a teoria de Euler-Bernoulli, as rotações θ_z e θ_y da seção transversal são iguais, respectivamente, às derivadas das translações w_y e w_z do eixo da barra. Portanto, o deslocamento u_x fica, para flexão no plano xy:

$$u_x \;=\; -y \cdot \theta_z \;=\; -y\,\frac{\partial w_y}{\partial x} \tag{2.35}$$

e para flexão no plano xz:

$$u_x \;=\; z \cdot \theta_y \;=\; -z\,\frac{\partial w_z}{\partial x} \tag{2.36}$$

A partir das hipóteses adotadas e da observação das Figuras 2.2, 2.3 e 2.4 e das Equações 2.35 e 2.36, pode-se obter o seguinte campo de deslocamentos para um ponto qualquer da barra:

$$u_x(x,y,z) \;=\; w_x(x) - y\,\frac{\partial\, w_y(x)}{\partial\, x} - z\,\frac{\partial\, w_z(x)}{\partial\, x} \tag{2.37}$$

$$u_y(x,y,z) \;=\; w_y(x) - z \cdot \theta_x(x) \tag{2.38}$$

$$u_z(x,y,z) \;=\; w_z(x) + y \cdot \theta_x(x) \tag{2.39}$$

2.3.3 Relações Deformações-Deslocamentos

A partir do campo de deslocamentos descrito e tomando as mesmas considerações feitas na formulação anterior, obtêm-se as seguintes relações entre deformações e deslocamentos para um ponto qualquer da barra:

$$\varepsilon_{xx} \;=\; \frac{\partial\, u_x(x,y,z)}{\partial\, x} \;=\; \frac{\partial\, w_x}{\partial\, x} - y\,\frac{\partial^2\, w_y}{\partial\, x^2} - z\,\frac{\partial^2\, w_z}{\partial\, x^2} \tag{2.40}$$

$$\gamma_{xy} \;=\; \frac{\partial\; u_y(x,y,z)}{\partial\; x} + \frac{\partial\; u_x(x,y,z)}{\partial\; y} \;=\; -\; z\; \frac{\partial\; \theta_x}{\partial\; x} \tag{2.41}$$

$$\gamma_{xz} \;=\; \frac{\partial\; u_z(x,y,z)}{\partial\; x} + \frac{\partial\; u_x(x,y,z)}{\partial\; z} \;=\; y\; \frac{\partial\; \theta_x}{\partial\; x} \tag{2.42}$$

Definindo-se as deformações generalizadas axial ε_a, de torção ψ e de flexão (curvaturas) κ_y e κ_z como:

$$\varepsilon_a = \frac{\partial\; w_x}{\partial\; x}, \quad \psi = \frac{\partial\; \theta_x}{\partial\; x}, \quad \kappa_y = -\; \frac{\partial^2\; w_z}{\partial\; x^2} \quad e \quad \kappa_z = \frac{\partial^2\; w_y}{\partial\; x^2}, \tag{2.43}$$

obtém-se então que:

$$\varepsilon_{xx} \;=\; \varepsilon_a - y \cdot \kappa_z + z \cdot \kappa_y \tag{2.44}$$

$$\gamma_{xy} \;=\; -z \cdot \psi \tag{2.45}$$

$$\gamma_{xz} \;=\; y \cdot \psi \tag{2.46}$$

2.3.4 Esforços Internos

Como as relações tensões-deformações são independentes das hipóteses cinemáticas, a integração das tensões leva aos esforços internos nas seções da barra (força axial N, momento de torção T e momentos fletores M_y e M_z), analogamente à formulação segundo a teoria de Timoshenko, exceto pela consideração dos esforços cortantes.

Para a força axial N e os momentos fletores M_y e M_z, a simplificação da integração leva, respectivamente, aos mesmos somatórios das Equações 2.29, 2.33 e 2.34. Devido à desconsideração do cisalhamento, o momento de torção T é obtido puramente a partir das deformações de torção, como segue:

$$T \;=\; \int_A (y_i^2 \cdot \psi + z_i^2 \cdot \psi) \cdot G_s \; dA \;=\; \sum_{i=1}^{q} A_i \cdot (y_i^2 \cdot \psi + z_i^2 \cdot \psi) \cdot G_{si} \tag{2.47}$$

Capítulo 3

MODELOS DISCRETOS DE PÓRTICO ESPACIAL

3.1 Introdução

Em uma análise estrutural, o problema de meio contínuo da estrutura real é substituído por um modelo matemático utilizando-se hipóteses simplificadoras, como visto no capítulo anterior. Tal modelo matemático é expresso por equações diferenciais cujas soluções, ditas soluções analíticas, são conhecidas apenas para alguns casos simples. Para superar as limitações da resolução das equações diferenciais associadas às soluções analíticas, adota-se um modelo numérico aproximado dito modelo discreto (Pitangueira, 2000).

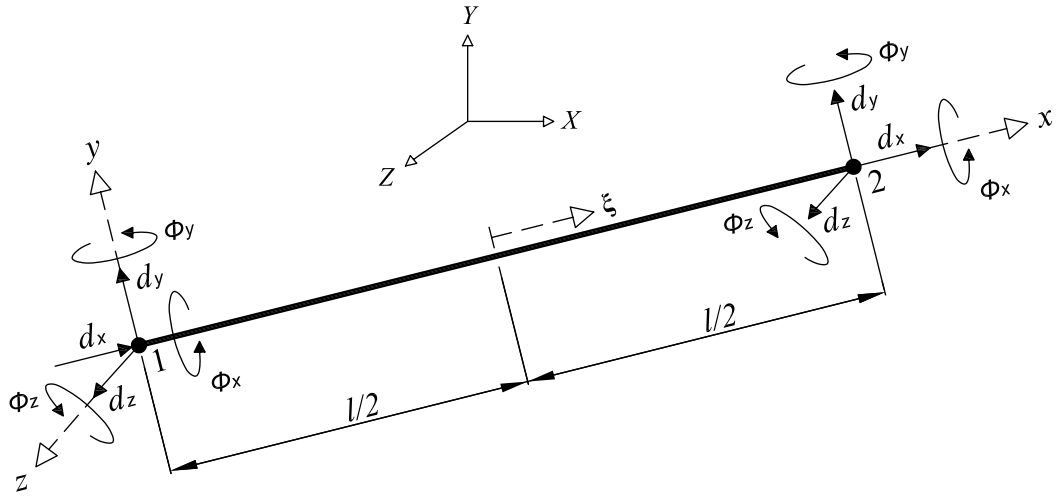
Nos modelos discretos as equações são algébricas e as incógnitas são determinadas em um número finito de pontos, diferentemente das soluções analíticas que permitem avaliar as incógnitas em um número infinito de pontos no domínio do problema.

Dentre os métodos discretos, o Método dos Elementos Finitos (MEF) é o mais difundido. No modelo de deslocamentos do MEF o problema é dividido em sub-domínios de dimensões finitas, denominados elementos finitos, onde o campo de deslocamentos é arbitrado. Escrevendo-se o campo de deslocamentos de cada elemento em função dos deslocamentos nodais, obtém-se um sistema de equações que permite solucionar o problema.

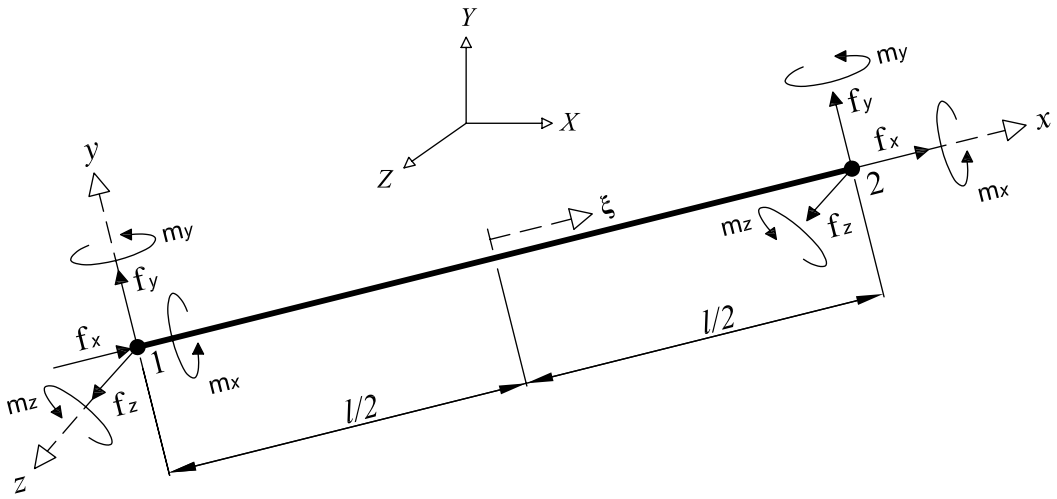
A formulação paramétrica do MEF generaliza este conceito, descrevendo as referidas aproximações em função de parâmetros adimensionais de um sistema de coordenadas congruente com a geometria do elemento. Esta estratégia, além de simplificar o cálculo de derivadas e

integrais inerentes ao método, confere ao mesmo grande generalidade. Isto significa que as diversas grandezas do modelo podem ser calculadas através de procedimentos unificados, que contemplam a maioria das hipóteses relativas à geometria, cinemática do contínuo, comportamento do material e carregamento (Almeida, 2005).

Deve-se atentar, no entanto, que, para a acurácia do modelo discreto, é necessário um modelo matemático adequado e a definição de um meio contínuo coerente com a estrutura real.



(a) Deslocamentos nodais



(b) Forças Nodais

Figura 3.1: Elemento finito de pórtico espacial de 2 nós.

Apresenta-se, neste capítulo, o elemento finito paramétrico de pórtico espacial considerando a teoria de Timoshenko, de acordo com a formulação apresentada na Seção 2.2, e em seguida o elemento finito considerando a teoria de Euler-Bernoulli, apresentada na Seção 2.3.

Tomar-se-á um elemento de comprimento l, seção transversal de geometria qualquer e composta por vários materiais de comportamento não-linear, com 6 graus de liberdade (3 translações e 3 rotações) por nó. Um elemento de dois nós está representado na Figura 3.1, tanto em termos de deslocamentos nodais como em termos de forças nodais.

3.2 Modelo Discreto considerando a Teoria de Timoshenko

3.2.1 Aproximação de Deslocamentos

Para um elemento finito unidimensional, formulado em termos cinemáticos, os deslocamentos em um ponto qualquer do eixo do elemento (armazenados em um vetor $\underline{\mathrm{w}}$) são obtidos a partir dos valores dos deslocamentos nodais (armazenados em um vetor $\underline{\mathrm{d}}$), através da seguinte aproximação:

$$\underline{\mathrm{w}} = \underline{\mathrm{N}}\ \underline{\mathrm{d}} \tag{3.1}$$

onde

$$\underline{\mathrm{w}} = \begin{bmatrix} w_x \\ w_y \\ w_z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}, \qquad \underline{\mathrm{d}} = \begin{bmatrix} \underline{\mathrm{d}}_1 \\ \vdots \\ \underline{\mathrm{d}}_n \end{bmatrix} \qquad \textit{sendo} \qquad \underline{\mathrm{d}}_i = \begin{bmatrix} d_{ix} \\ d_{iy} \\ d_{iz} \\ \phi_{ix} \\ \phi_{iy} \\ \phi_{iz} \end{bmatrix} \tag{3.2}$$

e n sendo o número de nós do elemento.

A matriz $\underline{\mathrm{N}}$ é uma matriz que contém as chamadas funções de forma do elemento, uma para cada deslocamento permitido por nó, dada por:

$$\underline{\mathrm{N}} = \begin{bmatrix} \underline{\mathrm{N}}_1 & \cdots & \underline{\mathrm{N}}_n \end{bmatrix} \qquad \textit{sendo} \qquad \underline{\mathrm{N}}_i = \begin{bmatrix} N_i & 0 & 0 & 0 & 0 & 0 \\ 0 & N_i & 0 & 0 & 0 & 0 \\ 0 & 0 & N_i & 0 & 0 & 0 \\ 0 & 0 & 0 & N_i & 0 & 0 \\ 0 & 0 & 0 & 0 & N_i & 0 \\ 0 & 0 & 0 & 0 & 0 & N_i \end{bmatrix} \tag{3.3}$$

e N_i a função polinomial aproximadora associada aos deslocamentos do nó i.

No caso da formulação paramétrica unidimensional do MEF, esta função é descrita em função da coordenada natural ξ (Figura 3.1).

Os deslocamentos em um ponto qualquer de coordenadas x, y e z (armazenados em um vetor $\underline{\mathrm{u}}$) podem ser obtidos utilizando o operador $\underline{l}_u$ (deduzido a partir da observação das Equações 2.1 a 2.3) sobre a aproximação da Equação 3.1, da seguinte forma:

$$\underline{\mathrm{u}} = \underline{l}_u \, \underline{\mathrm{w}} = \underline{l}_u \, \underline{\mathrm{N}} \, \underline{\mathrm{d}} \tag{3.4}$$

onde

$$\underline{\mathrm{u}} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \qquad e \qquad \underline{l}_u = \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & 0 \\ 0 & 0 & 1 & y & 0 & 0 \end{bmatrix} \tag{3.5}$$

3.2.2 Relação Deformação-Deslocamento

Utilizando a aproximação dada na Equação 3.1, as componentes de deformação generalizada da seção transversal (armazenadas em um vetor $\underline{\chi}$) podem ser calculadas através da relação:

$$\underline{\chi} = \underline{\mathrm{B}}\ \underline{\mathrm{d}} \qquad onde \qquad \underline{\chi} = \begin{bmatrix} \varepsilon_a \\ \gamma_y \\ \gamma_z \\ \psi \\ \kappa_y \\ \kappa_z \end{bmatrix} \qquad e \tag{3.6}$$

$$\underline{\mathrm{B}} = \begin{bmatrix} \underline{\mathrm{B}}_1 & \cdots & \underline{\mathrm{B}}_n \end{bmatrix}, \qquad sendo \qquad \underline{\mathrm{B}}_i = \begin{bmatrix} N_i' & 0 & 0 & 0 & 0 & 0 \\ 0 & N_i' & 0 & 0 & 0 & -N_i \\ 0 & 0 & N_i' & 0 & N_i & 0 \\ 0 & 0 & 0 & N_i' & 0 & 0 \\ 0 & 0 & 0 & 0 & N_i' & 0 \\ 0 & 0 & 0 & 0 & 0 & N_i' \end{bmatrix} \tag{3.7}$$

e N'_i a derivada das funções de forma em relação à coordenada cartesiana x.

As deformações em um ponto qualquer de coordenadas x, y e z (armazenados em um vetor $\underline{\varepsilon}$) podem, a partir da observação das Equações 2.8 a 2.10, ser obtidas utilizando o operador $\underline{l}_\varepsilon$ (neste caso, idêntico ao operador $\underline{l}_u$) sobre a aproximação da Equação 3.6:

$$\underline{\varepsilon} = \underline{l}_\varepsilon\ \underline{\chi} = \underline{l}_\varepsilon\ \underline{\mathrm{B}}\ \underline{\mathrm{d}} \qquad onde \qquad \underline{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \gamma_{xy} \\ \gamma_{xz} \end{bmatrix} \tag{3.8}$$

3.2.3 Relação Tensão-Deformação

Adotando-se as hipóteses relativas ao comportamento não-linear dos materiais, as tensões em um ponto qualquer de coordenadas x, y e z (armazenados em um vetor $\underline{\sigma}$) podem, a partir da observação das Equações 2.11 a 2.13, ser obtidas por:

$$\underline{\sigma} = \underline{E}_s\ \underline{\varepsilon} \tag{3.9}$$

onde

$$\underline{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \tau_{xy} \\ \tau_{xz} \end{bmatrix} \qquad e \qquad \underline{E}_s = \begin{bmatrix} E_s & 0 & 0 \\ 0 & G_s & 0 \\ 0 & 0 & G_s \end{bmatrix} \tag{3.10}$$

Ou, na forma incremental:

$$d\underline{\sigma} = \underline{E}_t \; d\underline{\varepsilon} \qquad com \qquad \underline{E}_t = \begin{bmatrix} E_t & 0 & 0 \\ 0 & G_t & 0 \\ 0 & 0 & G_t \end{bmatrix} \tag{3.11}$$

3.2.4 Esforços Internos

A partir da análise das Equações 2.29 a 2.34, as tensões generalizadas (vetor $\underline{\mathrm{M}}$) podem ser calculadas através da multiplicação da matriz constitutiva secante generalizada da seção transversal $\underline{C}_s$ pelas deformações generalizadas:

$$\underline{\mathrm{M}} = \underline{C}_s \; \underline{\chi} \qquad onde \qquad \underline{\mathrm{M}} = \begin{bmatrix} N \\ V_y \\ V_z \\ T \\ M_y \\ M_z \end{bmatrix} \qquad e \tag{3.12}$$

$$\underline{C}_s = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{si} & 0 & 0 & 0 & z_i E_{si} & -y_i E_{si} \\ 0 & \alpha_y G_{si} & 0 & -z_i G_{si} & 0 & 0 \\ 0 & 0 & \alpha_z G_{si} & y_i G_{si} & 0 & 0 \\ 0 & -z_i G_{si} & y_i G_{si} & (y_i^2 + z_i^2) G_{si} & 0 & 0 \\ z_i E_{si} & 0 & 0 & 0 & z_i^2 E_{si} & -y_i z_i E_{si} \\ -y_i E_{si} & 0 & 0 & 0 & -y_i z_i E_{si} & y_i^2 E_{si} \end{bmatrix} \tag{3.13}$$

Ou, na forma incremental:

$$d\underline{\mathrm{M}} = \underline{C}_t \; d\underline{\chi} \tag{3.14}$$

onde $\underline{C_t}$ é a matriz constitutiva tangente generalizada da seção transversal, dada por:

$$\underline{C_t} = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{ti} & 0 & 0 & 0 & z_i E_{ti} & -y_i E_{ti} \\ 0 & \alpha_y G_{ti} & 0 & -z_i G_{ti} & 0 & 0 \\ 0 & 0 & \alpha_z G_{ti} & y_i G_{ti} & 0 & 0 \\ 0 & -z_i G_{ti} & y_i G_{ti} & (y_i^2 + z_i^2) G_{ti} & 0 & 0 \\ z_i E_{ti} & 0 & 0 & 0 & z_i^2 E_{ti} & -y_i z_i E_{ti} \\ -y_i E_{ti} & 0 & 0 & 0 & -y_i z_i E_{ti} & y_i^2 E_{ti} \end{bmatrix} \tag{3.15}$$

3.3 Modelo Discreto considerando a Teoria de Euler-Bernoulli

3.3.1 Aproximação de Deslocamentos

Para um elemento finito unidimensional, considerando a teoria de Euler-Bernoulli, a matriz de funções de forma $\underline{\mathrm{N}}$, utilizada para aproximação dos deslocamentos de um ponto qualquer do eixo do elemento (Equação 3.1), é dada por:

$$\underline{\mathrm{N}} = \begin{bmatrix} \underline{\mathrm{N}}_1 & \cdots & \underline{\mathrm{N}}_n \end{bmatrix} \qquad \textit{sendo} \qquad \underline{\mathrm{N}}_i = \begin{bmatrix} N_i & 0 & 0 & 0 & 0 & 0 \\ 0 & \overline{N}_i & 0 & 0 & 0 & \overline{\overline{N}}_i \\ 0 & 0 & \overline{N}_i & 0 & \overline{\overline{N}}_i & 0 \\ 0 & 0 & 0 & N_i & 0 & 0 \\ 0 & 0 & \overline{N'}_i & 0 & \overline{\overline{N'}}_i & 0 \\ 0 & \overline{N'}_i & 0 & 0 & 0 & \overline{\overline{N'}}_i \end{bmatrix} \tag{3.16}$$

e N_i, $\overline{N}_i$ e $\overline{\overline{N}}_i$ as funções polinomiais aproximadoras associadas, respectivamente, aos deslocamentos d_x e ϕ_x, d_y e d_z, e ϕ_y e ϕ_z do nó i. $\overline{N}_i$' e $\overline{\overline{N}}_i$' são as derivadas das funções de forma em relação à coordenada cartesiana x.

Para obter-se os deslocamentos de um ponto qualquer de coordenadas x, y e z, utiliza-se na Equação 3.4 o operador $\underline{l}_u$ (deduzido a partir da observação das Equações 2.37 a 2.39), que é dado da seguinte forma:

$$\underline{l}_u = \begin{bmatrix} 1 & -y\frac{\partial}{\partial x} & z\frac{\partial}{\partial x} & 0 \\ 0 & 1 & 0 & -z \\ 0 & 0 & 1 & y \end{bmatrix} \tag{3.17}$$

3.3.2 Relação Deformação-Deslocamento

Para a teoria de Euler-Bernoulli, o vetor $\underline{\chi}$ das componentes de deformação generalizada da seção transversal resume-se a:

$$\underline{\chi} = \begin{bmatrix} \varepsilon_a \\ \psi \\ \kappa_y \\ \kappa_z \end{bmatrix} \tag{3.18}$$

E pode ser calculado pela Equação 3.6 com a matriz $\underline{\mathrm{B}}$ na seguinte forma:

$$\underline{\mathrm{B}} = \begin{bmatrix} \underline{\mathrm{B}}_1 & \cdots & \underline{\mathrm{B}}_n \end{bmatrix}, \qquad \textit{sendo} \qquad \underline{\mathrm{B}}_i = \begin{bmatrix} N_i' & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & N_i' & 0 & 0 \\ 0 & 0 & -\overline{N}_i'' & 0 & \overline{\overline{N}}_i'' & 0 \\ 0 & \overline{N}_i'' & 0 & 0 & 0 & \overline{\overline{N}}_i'' \end{bmatrix} \tag{3.19}$$

e N_i', $\overline{N}_i$" e $\overline{\overline{N}}_i$" as derivadas das funções de forma em relação à coordenada cartesiana x.

A partir da observação das Equações 2.44 a 2.46, obtém-se o operador $\underline{l}_\varepsilon$, através do qual é possível calcular as deformações em um ponto qualquer de coordenadas x, y e z (Equação 3.8):

$$\underline{l}_\varepsilon = \begin{bmatrix} 1 & 0 & z & -y \\ 0 & -z & 0 & 0 \\ 0 & y & 0 & 0 \end{bmatrix} \tag{3.20}$$

3.3.3 Esforços Internos

Sendo as relações tensão-deformação independentes dos modelos matemáticos, pela observação das Equações 2.29, 2.33, 2.34 e 2.47 obtém-se, para a teoria de Euler-Bernoulli, a matriz

constitutiva secante generalizada da seção transversal $\underline{C_s}$. É a partir dela que se obtém o vetor das tensões generalizadas $\underline{\mathrm{M}}$:

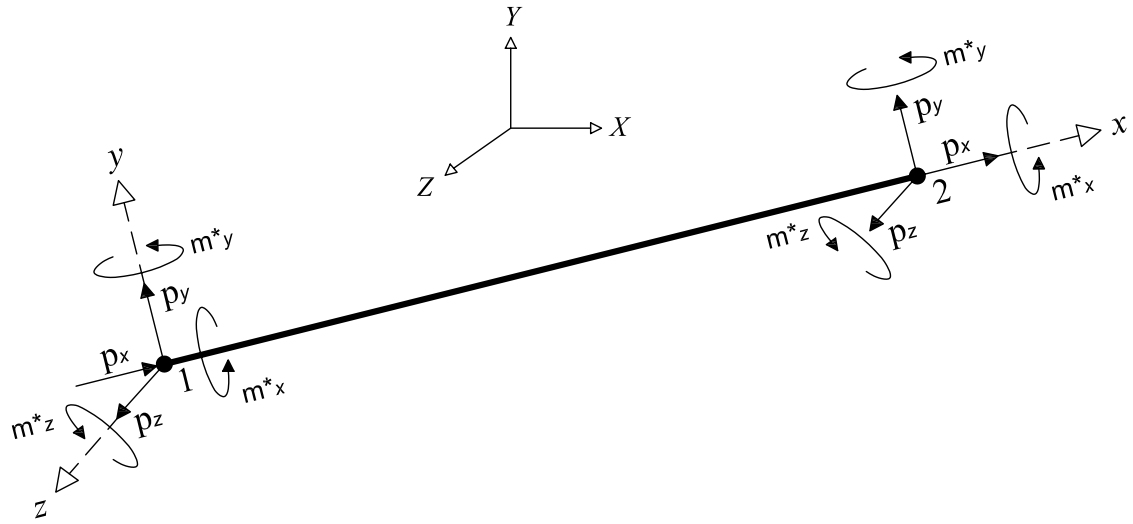
$$\underline{\mathrm{M}} = \underline{C_s}\ \underline{\chi} \qquad onde \qquad \underline{\mathrm{M}} = \begin{bmatrix} N \\ T \\ M_y \\ M_z \end{bmatrix} \qquad e \tag{3.21}$$

$$\underline{C_s} = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{si} & 0 & z_i E_{si} & -y_i E_{si} \\ 0 & (y_i^2 + z_i^2)\, G_{si} & 0 & 0 \\ z_i E_{si} & 0 & z_i^2 E_{si} & -y_i z_i E_{si} \\ -y_i E_{si} & 0 & -y_i z_i E_{si} & y_i^2 E_{si} \end{bmatrix} \tag{3.22}$$

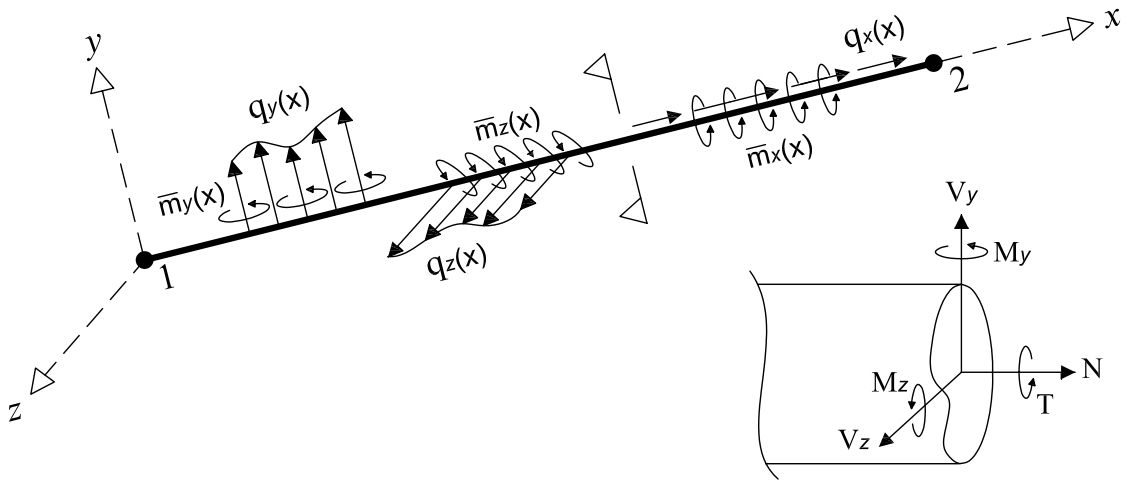
Ou, para a forma incremental (Equação 3.14), através da matriz constitutiva tangente generalizada da seção transversal $\underline{C_t}$, dada por:

$$\underline{C_t} = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{ti} & 0 & z_i E_{ti} & -y_i E_{ti} \\ 0 & (y_i^2 + z_i^2)\, G_{ti} & 0 & 0 \\ z_i E_{ti} & 0 & z_i^2 E_{ti} & -y_i z_i E_{ti} \\ -y_i E_{ti} & 0 & -y_i z_i E_{ti} & y_i^2 E_{ti} \end{bmatrix} \tag{3.23}$$

3.4 Equações de Equilíbrio do Elemento



(a) Esforços de extremidade



(b) Cargas de corpo e esforços internos

Figura 3.2: Carregamentos em um elemento finito de pórtico espacial de 2 nós.

Para obtenção das equações de equilíbrio do elemento utiliza-se o Princípio dos Trabalhos Virtuais. Seja o elemento sujeito a esforços de extremidade (armazenados em um vetor $\underline{\mathrm{p}}$), a cargas distribuídas no corpo do elemento (armazenadas em um vetor $\underline{\mathrm{b}}(x)$) e a esforços internos pré-existentes (armazenadas em um vetor $\underline{\mathrm{M}}_0$), como mostram a Figura 3.2 e as

equações abaixo:

$$\underline{\mathrm{p}} = \begin{bmatrix} \underline{\mathrm{p}}_1 \\ \vdots \\ \underline{\mathrm{p}}_n \end{bmatrix} \qquad sendo \qquad \underline{\mathrm{p}}_i = \begin{bmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \\ m^*_{ix} \\ m^*_{iy} \\ m^*_{iz} \end{bmatrix}, \tag{3.24}$$

$$\underline{\mathrm{b}}(x) = \begin{bmatrix} q_x(x) \\ q_y(x) \\ q_z(x) \\ \overline{m}_x(x) \\ \overline{m}_y(x) \\ \overline{m}_z(x) \end{bmatrix} \qquad e \qquad \underline{\mathrm{M}}_0 = \begin{bmatrix} N_0 \\ V_{y0} \\ V_{z0} \\ T_0 \\ M_{y0} \\ M_{z0} \end{bmatrix} \tag{3.25}$$

Impondo um campo de deslocamentos virtuais compatíveis ($\underline{\delta w}$), dado por:

$$\underline{\delta w} = \underline{\mathrm{N}}\ \underline{\delta d} \tag{3.26}$$

onde

$$\underline{\delta w} = \begin{bmatrix} \delta w_x \\ \delta w_y \\ \delta w_z \\ \delta \theta_x \\ \delta \theta_y \\ \delta \theta_z \end{bmatrix} \quad e\ os\ deslocamentos\ nodais\ virtuais \quad \underline{\delta d} = \begin{bmatrix} \delta d_x \\ \delta d_y \\ \delta d_z \\ \delta \phi_x \\ \delta \phi_y \\ \delta \phi_z \end{bmatrix}, \tag{3.27}$$

pode-se calcular o trabalho virtual externo ($\underline{W}_e$):

$$\underline{W}_e = \underline{\delta d}^T \cdot \underline{\mathrm{p}} + \int_L \underline{\delta w}^T \cdot \underline{\mathrm{b}}(x)\ dx \tag{3.28}$$

Substituindo a Equação 3.26 em 3.28 tem-se:

$$\underline{W_e} = \underline{\delta d}^T \cdot \underline{\mathrm{p}} + \int_L \underline{\delta d}^T \cdot \underline{\mathrm{N}}^T \cdot \underline{\mathrm{b}}(x)\ dx \tag{3.29}$$

O vetor $\underline{\mathrm{b}}(x)$ pode ser obtido pela aproximação:

$$\underline{\mathrm{b}}(x) = \underline{\mathrm{N}}\ \underline{\mathrm{b}}_N \tag{3.30}$$

onde $\underline{\mathrm{b}}_N$ é o vetor de valores nodais das cargas de corpo do elemento ($\underline{\mathrm{b}}(x)$). Substituindo a Equação 3.30 em 3.29, chega-se a:

$$\underline{W_e} = \underline{\delta d}^T \cdot \underline{\mathrm{p}} + \int_0^L \underline{\delta d}^T \cdot \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ dx \tag{3.31}$$

O trabalho virtual interno ($\underline{W_i}$) é dado por:

$$\underline{W_i} = \int_V \underline{\delta\varepsilon}^T \cdot \underline{\sigma}\ dV + \int_V \underline{\delta\varepsilon}^T \cdot \underline{\sigma_0}\ dV \tag{3.32}$$

onde o vetor que representa um estado de deformações virtuais ($\underline{\delta\varepsilon}$) e o vetor que representa um estado de tensões pré-existente ($\underline{\sigma_0}$) são:

$$\underline{\delta\varepsilon} = \begin{bmatrix} \delta\varepsilon_{xx} \\ \delta\gamma_{xy} \\ \delta\gamma_{xz} \end{bmatrix} \quad e \quad \underline{\sigma_0} = \begin{bmatrix} \sigma_{xx0} \\ \tau_{xy0} \\ \tau_{xz0} \end{bmatrix} \tag{3.33}$$

O vetor $\underline{\delta\varepsilon}$ pode ser obtido utilizando o operador $\underline{l_\varepsilon}$:

$$\underline{\delta\varepsilon} = \underline{l_\varepsilon}\ \underline{\delta\chi} \tag{3.34}$$

onde o vetor de deformações virtuais generalizadas ($\underline{\delta\chi}$) é:

$$\underline{\delta\chi} = \begin{bmatrix} \delta\varepsilon_a \\ \delta\gamma_y \\ \delta\gamma_z \\ \delta\psi \\ \delta\kappa_y \\ \delta\kappa_z \end{bmatrix} \tag{3.35}$$

Substituindo as Equações 3.8, 3.9 e 3.34 em 3.32, tem-se:

$$\underline{W}_i = \int_V \underline{\delta\chi}^T \cdot \underline{l}_\varepsilon^T \cdot \underline{E}_s \cdot \underline{l}_\varepsilon \cdot \underline{\chi}\ dV + \int_V \underline{\delta\chi}^T \cdot \underline{l}_\varepsilon^T \cdot \underline{E}_{s0} \cdot \underline{l}_\varepsilon \cdot \underline{\chi}_0\ dV \tag{3.36}$$

Igualando-se as Equações 3.31 e 3.36, segundo o Princípio dos Trabalhos Virtuais, obtém-se:

$$\int_V \underline{\delta\chi}^T \cdot \underline{l}_\varepsilon^T \cdot \underline{E} \cdot \underline{l}_\varepsilon \cdot \underline{\chi}\ dV + \int_V \underline{\delta\chi}^T \cdot \underline{l}_\varepsilon^T \cdot \underline{E}_{s0} \cdot \underline{l}_\varepsilon \cdot \underline{\chi}_0\ dV$$

$$= \underline{\delta d}^T \cdot \underline{\mathrm{p}} + \int_0^L \underline{\delta d}^T \cdot \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ dx \tag{3.37}$$

As integrais em 3.37 podem ser simplificadas levando em consideração que os termos dos vetores $\underline{\delta\chi}$ e $\underline{\chi}$ são constantes na seção transversal e variáveis ao longo do comprimento. Substituindo também a relação da Equação 3.6, tem-se:

$$\int_0^L \underline{\delta d}^T \cdot [\ \underline{\mathrm{B}}^T \cdot (\int_A \underline{l}_\varepsilon^T \cdot \underline{E}_s \cdot \underline{l}_\varepsilon\ dA) \cdot \underline{\mathrm{B}} \cdot \underline{d}\ +\ \underline{\mathrm{B}}^T \cdot (\int_A \underline{l}_\varepsilon^T \cdot \underline{E}_{s0} \cdot \underline{l}_\varepsilon\ dA) \cdot \underline{\chi}_0\ ]\ dx$$

$$= \underline{\delta d}^T \cdot \underline{\mathrm{p}}\ + \int_0^L \underline{\delta d}^T \cdot \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ dx \tag{3.38}$$

Considerando que $\underline{\delta d}$ e $\underline{d}$ são valores nodais, obtém-se:

$$\underline{\delta d}^T \cdot \int_0^L [\ \underline{\mathrm{B}}^T \cdot (\int_A \underline{l}_\varepsilon^T \cdot \underline{E}_s \cdot \underline{l}_\varepsilon\ dA) \cdot \underline{\mathrm{B}} \cdot \underline{d}\ +\ \underline{\mathrm{B}}^T \cdot (\int_A \underline{l}_\varepsilon^T \cdot \underline{E}_{s0} \cdot \underline{l}_\varepsilon\ dA) \cdot \chi_0\ ]\ dx$$

$$= \underline{\delta d}^T \cdot \underline{\mathrm{p}}\ +\ \underline{\delta d}^T \int_0^L \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ \ dx \tag{3.39}$$

Como $\underline{\delta d}$ é o vetor de deslocamentos nodais virtuais arbitrários, a Equação 3.39 somente se verifica se

$$\int_0^L [\ \underline{\mathrm{B}}^T \cdot (\int_A \underline{l}_\varepsilon^T \cdot \underline{E}_s \cdot \underline{l}_\varepsilon\ dA) \cdot \underline{\mathrm{B}} \cdot \underline{d}\ +\ \underline{\mathrm{B}}^T \cdot (\int_A \underline{l}_\varepsilon^T \cdot \underline{E}_{s0} \cdot \underline{l}_\varepsilon\ dA) \cdot \chi_0\ ]\ dx$$

$$= \underline{\mathrm{p}}\ +\ \int_0^L \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ dx \tag{3.40}$$

Demonstra-se, utilizando-se a discretização da seção em q pequenas áreas, que a matriz constitutiva secante generalizada da seção transversal é:

$$\underline{C}_s = \int_A \underline{l}_\varepsilon^T \cdot \underline{E}_s \cdot \underline{l}_\varepsilon\ \ dA \tag{3.41}$$

Sendo, para a teoria de Timoshenko e já acrescentados os fatores de correção de cisalhamento da seção transversal:

$$\underline{C}_s = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{si} & 0 & 0 & 0 & z_i E_{si} & -y_i E_{si} \\ 0 & \alpha_y G_{si} & 0 & -z_i G_{si} & 0 & 0 \\ 0 & 0 & \alpha_z G_{si} & y_i G_{si} & 0 & 0 \\ 0 & -z_i G_{si} & y_i G_{si} & (y_i^2 + z_i^2) G_{si} & 0 & 0 \\ z_i E_{si} & 0 & 0 & 0 & z_i^2 E_{si} & -y_i z_i E_{si} \\ -y_i E_{si} & 0 & 0 & 0 & -y_i z_i E_{si} & y_i^2 E_{si} \end{bmatrix}$$

como já obtido em 3.13. E, para a teoria de Euler-Bernoulli:

$$\underline{C}_s = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{si} & 0 & z_i E_{si} & -y_i E_{si} \\ 0 & (y_i^2 + z_i^2)\, G_{si} & 0 & 0 \\ z_i E_{si} & 0 & z_i^2 E_{si} & -y_i z_i E_{si} \\ -y_i E_{si} & 0 & -y_i z_i E_{si} & y_i^2 E_{si} \end{bmatrix}$$

como já obtido em 3.22.

A Equação 3.40 fica então:

$$\left(\int_0^L \underline{\mathrm{B}}^T \cdot \underline{C}_s \cdot \underline{\mathrm{B}}\ dx \right) \cdot \underline{d}\ +\ \int_0^L \underline{\mathrm{B}}^T \cdot \underline{C}_{s0} \cdot \chi_0\ dx\ =\ \underline{\mathrm{p}}\ +\ \int_0^L \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ dx \tag{3.42}$$

É possível identificar a matriz de rigidez do elemento $\underline{k}^e$, o vetor de cargas nodais equivalentes às forças de corpo $\underline{\mathrm{f}}^e_{eq}$ e o vetor de forças nodais equivalentes aos esforços internos pré-existentes $\underline{\mathrm{f}}^e_0$ como:

$$\underline{k}^e = \int_0^L \underline{\mathrm{B}}^T \cdot \underline{C}_s \cdot \underline{\mathrm{B}}\ dx \tag{3.43}$$

$$\underline{\mathrm{f}}^e_{eq} = \int_0^L \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N\ \ dx \tag{3.44}$$

$$\underline{\mathrm{f}}^e_0 = \int_0^L \underline{\mathrm{B}}^T \cdot \underline{C}_{s0} \cdot \chi_0\ dx = \int_0^L \underline{\mathrm{B}}^T \cdot \underline{\mathrm{M}}_0\ \ dx \tag{3.45}$$

Chega-se então à equação matricial de equilíbrio de cada elemento, dada por:

$$\underline{k}^e \cdot \underline{\mathrm{d}} = \underline{\mathrm{p}}\ +\ \underline{\mathrm{f}}^e_{eq}\ -\ \underline{\mathrm{f}}^e_0 \tag{3.46}$$

Transformando as Equações 3.43, 3.44 e 3.45 para o sistema de coordenadas naturais através do operador jacobiano J, que no caso unidimensional se resume a:

$$J = \frac{\partial x}{\partial \xi} \tag{3.47}$$

tem-se:

$$\underline{k}^e = \int_{-1}^{1} \underline{\mathrm{B}}^T \cdot \underline{C}_s \cdot \underline{\mathrm{B}} \cdot J \; d\xi \tag{3.48}$$

$$\underline{\mathrm{f}}_{eq}^e = \int_{-1}^{1} \underline{\mathrm{N}}^T \cdot \underline{\mathrm{N}} \cdot \underline{\mathrm{b}}_N \cdot J \; d\xi \tag{3.49}$$

$$\underline{\mathrm{f}}_0^e = \int_{-1}^{1} \underline{\mathrm{B}}^T \cdot \underline{\mathrm{M}}_0 \cdot J \; d\xi \tag{3.50}$$

Utilizando-se a Quadratura de Gauss para integração numérica, obtém-se:

$$\underline{k}^e = \sum_{j=1}^{m} \omega_j \cdot \underline{\mathrm{B}}_j^T \cdot \underline{C}_{sj} \cdot \underline{\mathrm{B}}_j \cdot J_j \tag{3.51}$$

$$\underline{\mathrm{f}}_{eq}^e = \sum_{j=1}^{m} \omega_j \cdot \underline{\mathrm{N}}_j^T \cdot \underline{\mathrm{N}}_j \cdot \underline{\mathrm{b}}_{Nj} \cdot J_j \tag{3.52}$$

$$\underline{\mathrm{f}}_0^e = \sum_{j=1}^{m} \omega_j \cdot \underline{\mathrm{B}}_j^T \cdot \underline{\mathrm{M}}_{0j} \cdot J_j \tag{3.53}$$

onde m é o número de pontos de Gauss adotado para a integração e ω_j é o fator-peso.

Para o caso de equações de equilíbrio incrementais, a matriz de rigidez tangente do elemento fica:

$$\underline{k}_t^e = \int_{-1}^{1} \underline{\mathrm{B}}^T \cdot \underline{C}_t \cdot \underline{\mathrm{B}} \cdot J \; d\xi = \sum_{j=1}^{m} \omega_j \cdot \underline{\mathrm{B}}_j^T \cdot \underline{C}_{tj} \cdot \underline{\mathrm{B}}_j \cdot J_j \tag{3.54}$$

onde a matriz constitutiva tangente generalizada da seção transversal $\underline{C}_t$ é:

$$\underline{C}_t = \int_A \underline{l}_\varepsilon^T \cdot \underline{E}_t \cdot \underline{l}_\varepsilon \; dA \tag{3.55}$$

Sendo, para a teoria de Timoshenko e já acrescentados os fatores de correção de cisalhamento da seção transversal:

$$\underline{C}_t = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{ti} & 0 & 0 & 0 & z_i E_{ti} & -y_i E_{ti} \\ 0 & \alpha_y G_{ti} & 0 & -z_i G_{ti} & 0 & 0 \\ 0 & 0 & \alpha_z G_{ti} & y_i G_{ti} & 0 & 0 \\ 0 & -z_i G_{ti} & y_i G_{ti} & (y_i^2 + z_i^2) G_{ti} & 0 & 0 \\ z_i E_{ti} & 0 & 0 & 0 & z_i^2 E_{ti} & -y_i z_i E_{ti} \\ -y_i E_{ti} & 0 & 0 & 0 & -y_i z_i E_{ti} & y_i^2 E_{ti} \end{bmatrix}$$

como já obtido em 3.15. E, para a teoria de Euler-Bernoulli:

$$\underline{C}_t = \sum_{i=1}^{q} A_i \begin{bmatrix} E_{ti} & 0 & z_i E_{ti} & -y_i E_{ti} \\ 0 & (y_i^2 + z_i^2) G_{ti} & 0 & 0 \\ z_i E_{ti} & 0 & z_i^2 E_{ti} & -y_i z_i E_{ti} \\ -y_i E_{ti} & 0 & -y_i z_i E_{ti} & y_i^2 E_{ti} \end{bmatrix}$$

como já obtido em 3.23.

3.5 Equação de Equilíbrio do Modelo

A combinação das rigidezes e carregamentos dos vários elementos leva às equações de equilíbrio dos nós do modelo:

$$\underline{K} \cdot \underline{\mathrm{D}} = \underline{\mathrm{P}} \; + \; \underline{\mathrm{F}}_{eq} \; - \; \underline{\mathrm{F}}_0 \tag{3.56}$$

onde $\underline{\mathrm{D}}$ e $\underline{\mathrm{P}}$ são os vetores de deslocamentos e cargas nodais referidos ao sistema global de coordenadas e

$$\underline{K} = \sum_{e=1}^{nc} \underline{K}^e \tag{3.57}$$

$$\underline{\mathrm{F}}_{eq} = \sum_{e=1}^{nc} \underline{\mathrm{F}}_{eq}^e \tag{3.58}$$

$$\underline{\mathrm{F}}_0 = \sum_{e=1}^{nc} \underline{\mathrm{F}}_0^e \tag{3.59}$$

sendo ne o número de elementos do modelo. $\underline{K}^e$ é a matriz de rigidez global do elemento $\boldsymbol{e}$. $\underline{\mathrm{F}}^e_{eq}$ e $\underline{\mathrm{F}}^e_0$ são, respectivamente, os vetores de cargas nodais equivalentes às forças de corpo e de forças nodais equivalentes aos esforços internos pré-existentes do elemento $\boldsymbol{e}$, referidos ao sistema global de coordenadas.

Para obtenção das matrizes referidas ao sistema global de coordenadas é necessário realizar a transformação, dada por

$$\underline{K}^e = \underline{\mathrm{T}}^T \cdot \underline{k}^e \cdot \underline{\mathrm{T}} \tag{3.60}$$

$$\underline{\mathrm{F}}^e_{eq} = \underline{\mathrm{T}}^T \cdot \underline{\mathrm{f}}^e_{eq} \tag{3.61}$$

$$\underline{\mathrm{F}}^e_0 = \underline{\mathrm{T}}^T \cdot \underline{\mathrm{f}}^e_0 \tag{3.62}$$

A matriz $\underline{\mathrm{T}}$ é a matriz de transformação entre os sistemas local e global de coordenadas e tem a seguinte forma:

$$\underline{\mathrm{T}} = \begin{bmatrix} \underline{\mathrm{R}}_1 & & \\ & \ddots & \\ & & \underline{\mathrm{R}}_n \end{bmatrix} \tag{3.63}$$

onde $\underline{\mathrm{R}}_i$ é uma matriz 6x6 formada pelos cossenos diretores e n o número de nós do elemento.

Capítulo 4

ANÁLISE FISICAMENTE NÃO-LINEAR DE ESTRUTURAS

4.1 Introdução

Como discutido em Pitangueira (1998), é normal, na análise do comportamento não-linear de materiais, considerá-los, inicialmente, como homogêneos, elásticos e isotrópicos, admitindo-se que, com aplicação de carga e conseqüentes deformações, os materiais deixam de ser elásticos e isotrópicos e tornam-se heterogêneos pela deterioração nas regiões mais solicitadas. O fenômeno ocorre de tal maneira que, durante o processo de deterioração da estrutura, alguns pontos do domínio apresentam características mecânicas distintas dos demais, observando-se que esta combinação de materiais com características muito diversas (regiões danificadas junto a outras com as características do material homogêneo inicial) causa efeitos não-lineares pronunciados na resposta da estrutura.

Em um meio contínuo e homogêneo, todos os pontos do domínio têm as mesmas propriedades e, portanto, reagem da mesma maneira às ações externas. Assim, a relação entre tensões e deformações é única e não depende da geometria da amostra. Esta única relação tensão-deformação é dita uma propriedade do material, sua lei constitutiva. Esta é a idéia usual da análise por elementos finitos, que considera o meio contínuo, o material inicialmente homogêneo e a lei tensão-deformação conhecida a priori.

Conforme Fuina (2004) e Pitangueira (1998), a representação do comportamento não-linear de estruturas no espaço parâmetro de carga-deslocamentos envolve fenômenos de aumento de deslocamentos com decréscimos de cargas ou mesmo decréscimo de deslocamentos com decréscimo de cargas, como mostram as trajetórias de equilíbrio da Figura 4.1. Para a representação completa de tais trajetórias, os procedimentos numéricos empregados devem ser capazes de detectar a ocorrência de pontos limites de carga (ponto B na Figura 4.1) e de pontos limites de deslocamento (ponto C na Figura 4.1).

Na análise não-linear de uma estrutura, deseja-se obter trajetórias de equilíbrio para determinados graus de liberdade da discretização, executando-se um processo incremental-iterativo nas variáveis do problema.

Assim, dado um campo de deslocamentos $\{U\}$ e um fator de carga proporcional λ, equivalentes a um ponto da trajetória de equilíbrio (ponto A na Figura 4.1), deseja-se encontrar outro ponto de equilíbrio (ponto B na Figura 4.1) de modo que a variação de determinadas grandezas do problema no passo incremental (do ponto A ao ponto B) seja controlada.

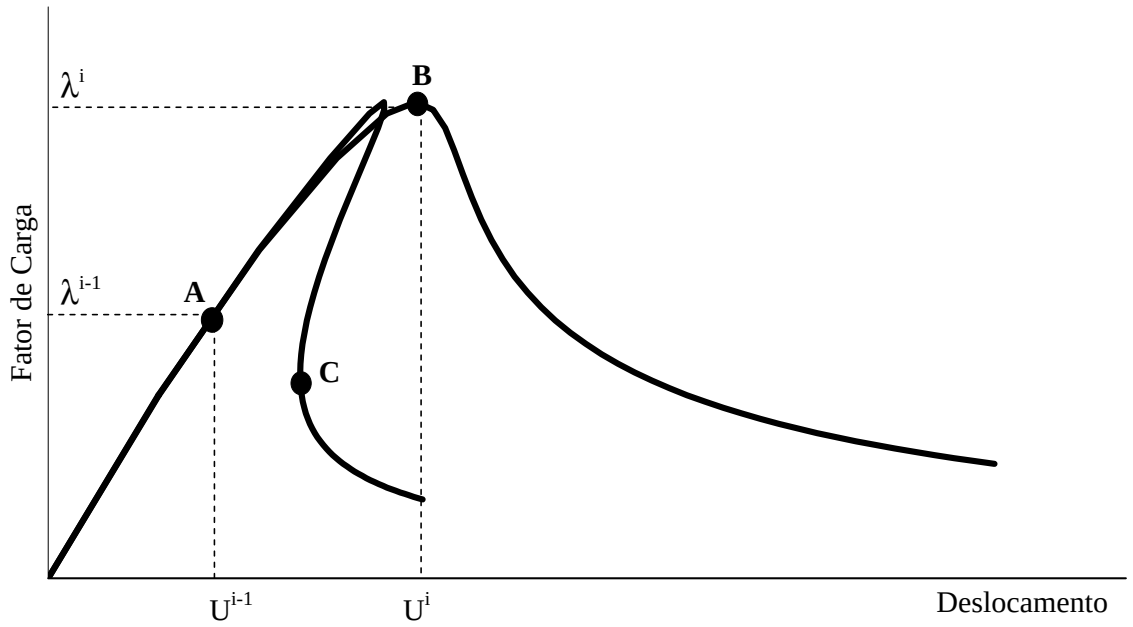


Figura 4.1: Trajetórias de equilíbrio típicas em problemas não-lineares (Pitangueira, 1998).

Este capítulo apresenta, primeiramente, exemplos de leis constitutivas não-lineares para alguns materiais que são adotados nas simulações numéricas apresentadas na Seção 6.3. A seguir, discute-se brevemente a formulação usada para solução de equações não-lineares de equilíbrio, bem como os diferentes métodos de controle para obtenção das trajetórias de equilíbrio. Tanto as leis constitutivas quanto a solução de equações não-lineares encontram-se implementadas no programa **INSANE**, conforme discutido no Capítulo 5.

4.2 Leis Constitutivas Não-Lineares

Para os exemplos numéricos apresentados neste trabalho, serão consideradas, além do material linear elástico, duas leis constitutivas para o concreto, a lei da NBR6118 (2003) e a lei proposta por Carreira e Chu (1985), e uma lei constitutiva bilinear para o aço.

4.2.1 Concreto segundo a NBR6118 (2003)

A lei tensão-deformação do concreto para compressão (Figura 4.2), conforme o item 8.2.10 da NBR6118, tem dois trechos distintos: o primeiro, para deformações inferiores a 0,2%, é dado pela Equação 4.1; o segundo é constante, para deformações entre 0,2% e 0,35%.

$$\sigma_c = 0,85 \cdot f_{cd} \cdot [1 - (1 - \frac{\varepsilon_c}{0,002})^2] \tag{4.1}$$

Na Equação 4.1, f_{cd} representa a resistência à compressão de cálculo do concreto, dada no item 12.3.3 da norma, e é igual à resistência característica à compressão do concreto f_{ck} dividida por 1,4. O módulo de elasticidade inicial do concreto E_{c0} é dado por:

$$E_{c0} = 5600 \cdot \sqrt{f_{ck}} \tag{4.2}$$

com E_{c0} e f_{ck} dados em *MPa*.

Para tração, a lei tensão-deformação do concreto é bilinear conforme a Figura 4.2. f_{ctk} é a resistência característica à tração, conforme o item 8.2.5, e a deformação última é 0,15%. A norma também estabelece o módulo de poisson υ igual à 0,2.

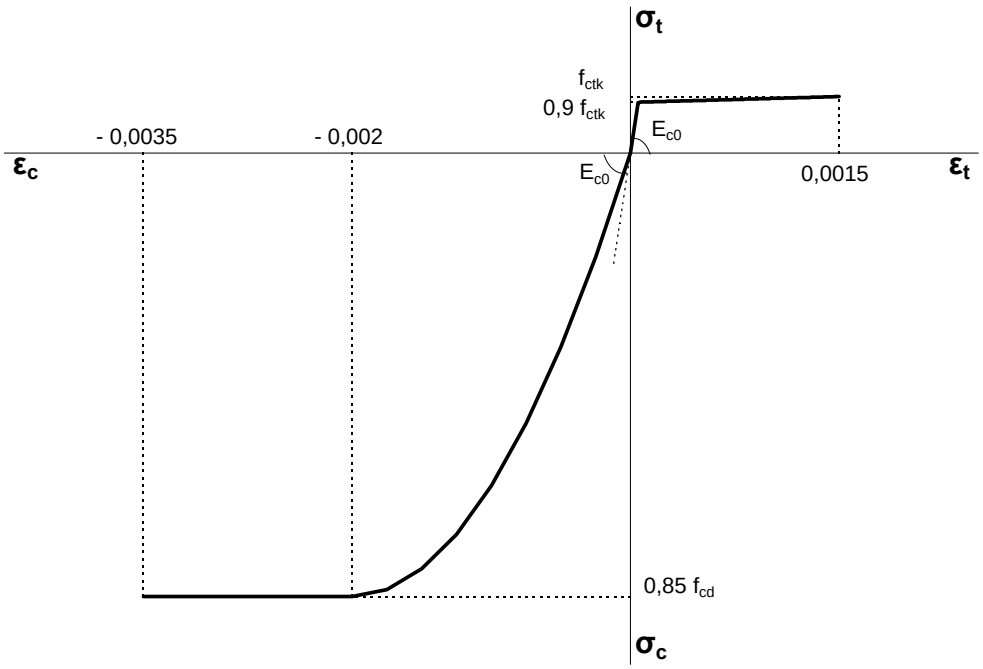


Figura 4.2: Lei constitutiva para o concreto segundo a NBR6118 (2003).

4.2.2 Concreto segundo Carreira e Chu

A lei constitutiva do concreto proposta por Carreira e Chu (1985), mostrada na Figura 4.3, é dada pelas relações:

$$\sigma_c = f_c \cdot \frac{k_c \cdot (\frac{\varepsilon}{\varepsilon_c})}{k_c - 1 + (\frac{\varepsilon}{\varepsilon_c})^{k_c}}, \qquad com \qquad k_c = \frac{1}{1 - (\frac{f_c}{\varepsilon_c E_0})} \tag{4.3}$$

para compressão, e

$$\sigma_t = f_t \cdot \frac{k_t \cdot (\frac{\varepsilon}{\varepsilon_t})}{k_t - 1 + (\frac{\varepsilon}{\varepsilon_t})^{k_t}}, \qquad com \qquad k_t = \frac{1}{1 - (\frac{f_t}{\varepsilon_t E_0})} \tag{4.4}$$

para tração.

E_0 é o módulo de elasticidade inicial do concreto, f_c a resistência à compressão do concreto, f_t a resistência à tração do concreto, ε_c é a deformação relativa à tensão máxima de compressão e ε_t é a deformação relativa à tensão máxima de tração.

Pode-se observar que a lei tensão-deformação de Carreira e Chu leva em conta o amolecimento do concreto na compressão, enquanto a norma brasileira considera um patamar de escoamento constante até um limite de deformação último. Ambas as leis consideram a capacidade de resistência à tração do concreto, ainda que pequena em relação a resistência à compressão.

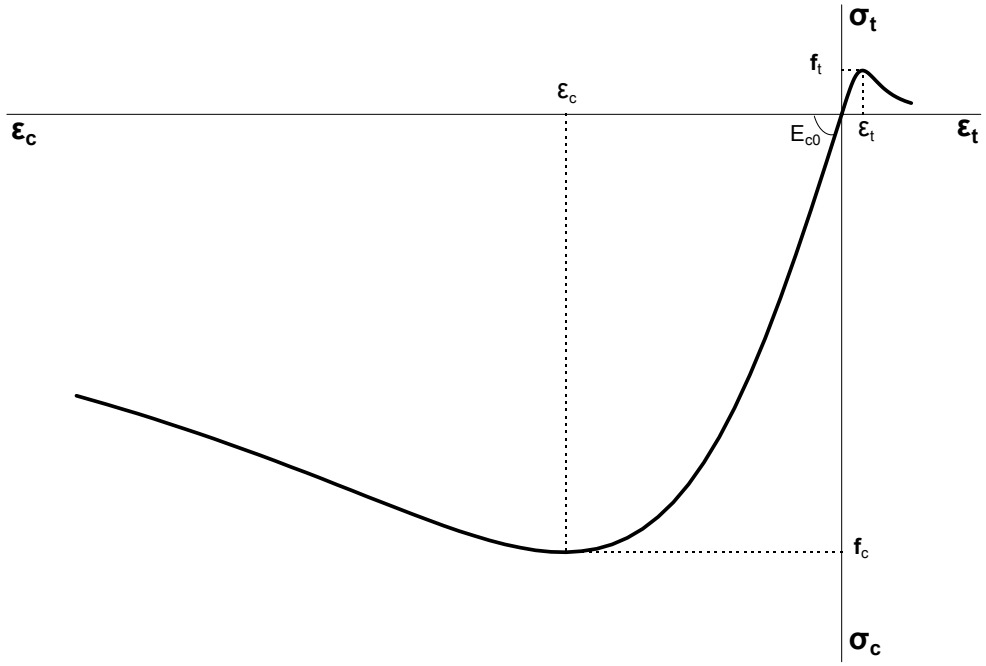


Figura 4.3: Lei constitutiva para o concreto segundo Carreira e Chu (1985).

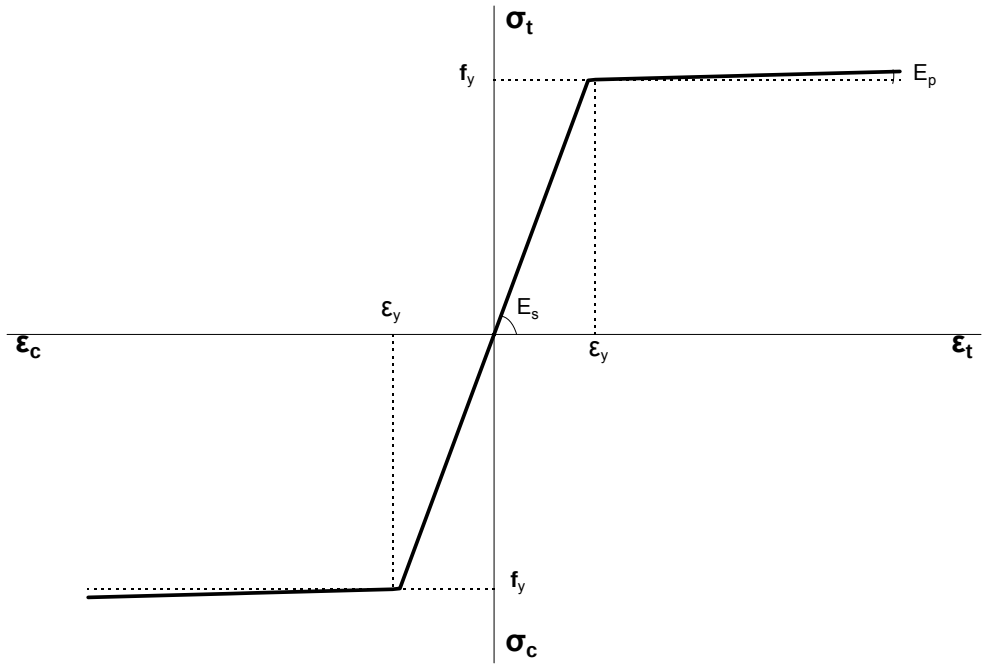


Figura 4.4: Lei constitutiva para o aço.

4.2.3 Aço Elasto-Plástico

O comportamento do aço da armadura em compressão e tração é representado por uma lei bilinear (Figura 4.4). E_s é o módulo de elasticidade do aço, E_p é o módulo elastoplástico, f_y é a tensão de escoamento e ε_y a deformação de escoamento. Observa-se que, se o módulo elastoplástico for igual a zero, o escoamento do aço se dá por um patamar constante, sem ocorrer endurecimento.

4.3 Solução de Equações Não-Lineares de Equilíbrio

Em uma análise não-linear, confronta-se com o problema de resolver o sistema de N+1 incógnitas (N deslocamentos incrementais e um incremento no fator de carga) e N+1 equações (N equações de equilíbrio e uma equação de restrição).

Um processo incremental-iterativo torna-se necessário para solucionar o problema. Para este fim, a equação de equilíbrio incremental correspondente a iteração j do passo i pode ser escrita na forma abaixo:

$$[K]^i_{j-1} \cdot \{\delta U\}^i_j = \delta\lambda^i_j \cdot \{P\} + \{Q\}^i_{j-1} \tag{4.5}$$

onde:

$[K]^i_{j-1}$ é a matriz de rigidez tangente na iteração j-1 do passo i, função do campo de deslocamentos $\{U\}^i_{j-1}$;
$\{\delta U\}^i_j$ é o vetor de deslocamentos incrementais da iteração j do passo i;
$\delta\lambda^i_j$ é o incremento do fator de cargas na iteração j do passo i;
$\{P\}$ é o vetor de cargas de referência;
$\{Q\}^i_{j-1}$ é o vetor de forças residuais da iteração j-1 do passo i.

Inicialmente, estabelece-se, em função do parâmetro de controle, um valor para o incremento do fator de carga $\delta\lambda_j$, podendo-se então obter $\{\delta U\}_j$, o qual pode ser decomposto nas

parcelas associadas à carga de referência, $\{\delta U\}_j^P$, e à carga residual $\{\delta U\}_j^Q$, na forma:

$$\{\delta U\}_j = \delta\lambda_j \cdot \{\delta U\}_j^P + \{\delta U\}_j^Q \tag{4.6}$$

com

$$[K]_{j-1} \cdot \{\delta U\}_j^P = \{P\} \tag{4.7}$$

e

$$[K]_{j-1} \cdot \{\delta U\}_j^Q = \{Q\}_{j-1} \tag{4.8}$$

Ao final de cada iteração, a convergência é verificada através da magnitude do vetor de forças residuais $\{Q\}_j$ e/ou da magnitude do vetor de deslocamentos iterativos $\{\delta U\}_j$ e o processo iterativo continua até que determinado critério de convergência seja atendido. Se uma nova iteração for necessária, após calculados $\{\delta U\}_j^P$ e $\{\delta U\}_j^Q$ utilizando-se as Equações 4.7 e 4.8, o valor de $\delta\lambda_j$ deve ser obtido com uma equação de restrição que envolve combinações das grandezas do problema.

A atualização das variáveis é feita da seguinte forma:

$$\lambda_j = \lambda_{j-1} + \delta\lambda_j \tag{4.9}$$

$$\{U\}_j = \{U\}_{j-1} + \{\delta U\}_j \tag{4.10}$$

O vetor de cargas residuais da iteração j é dado por:

$$\{Q\}_j = \lambda_j \cdot \{P\} + \{P_0\} - \{F\}_j \tag{4.11}$$

onde $\{F\}_j$ é o vetor de forças equivalentes às tensões internas ao final da iteração j e $\{P_0\}$ é o vetor de cargas constantes.

Observa-se que, na primeira iteração de cada passo, o vetor de cargas residuais $\{Q\}_{j-1}$ é nulo.

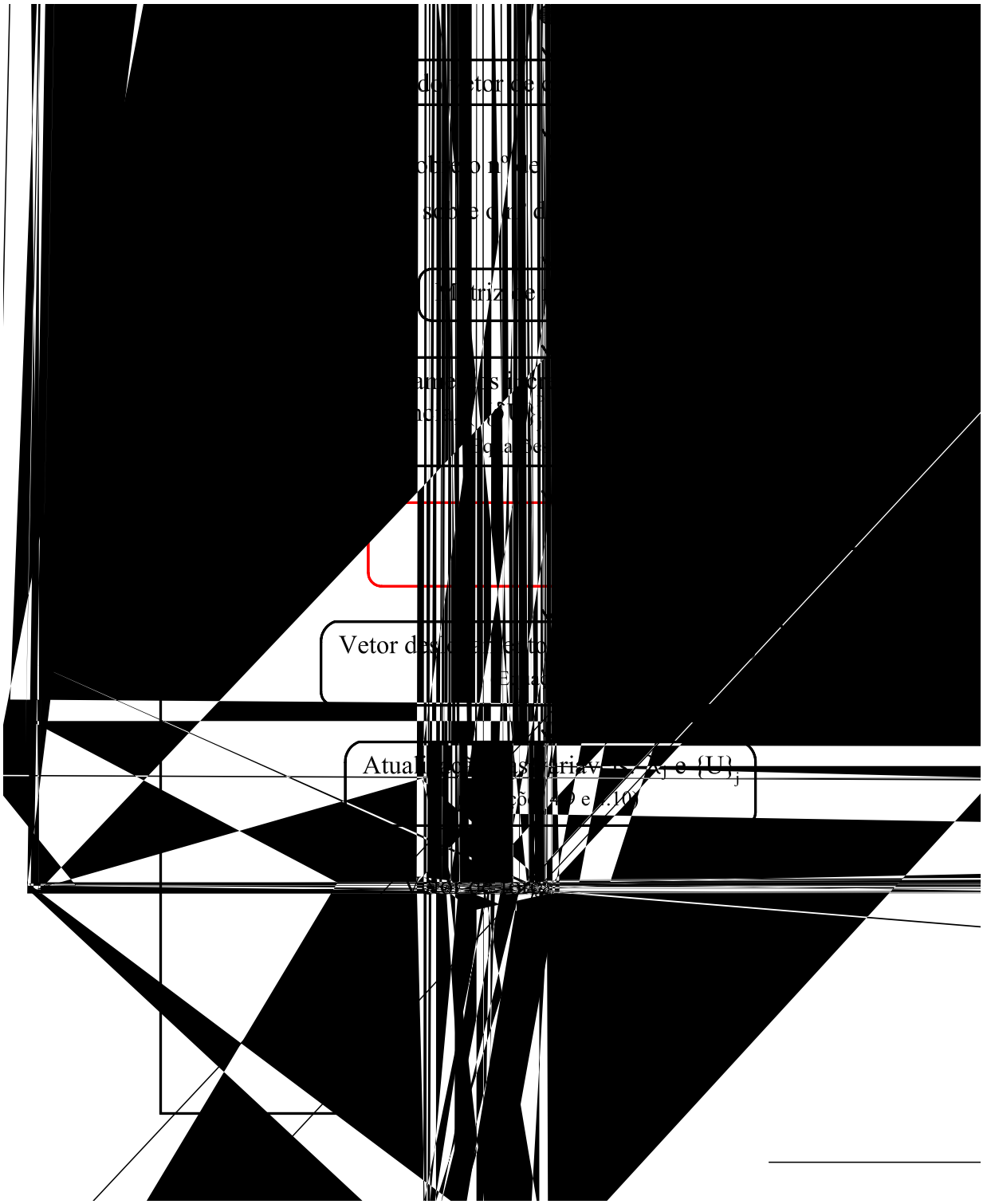
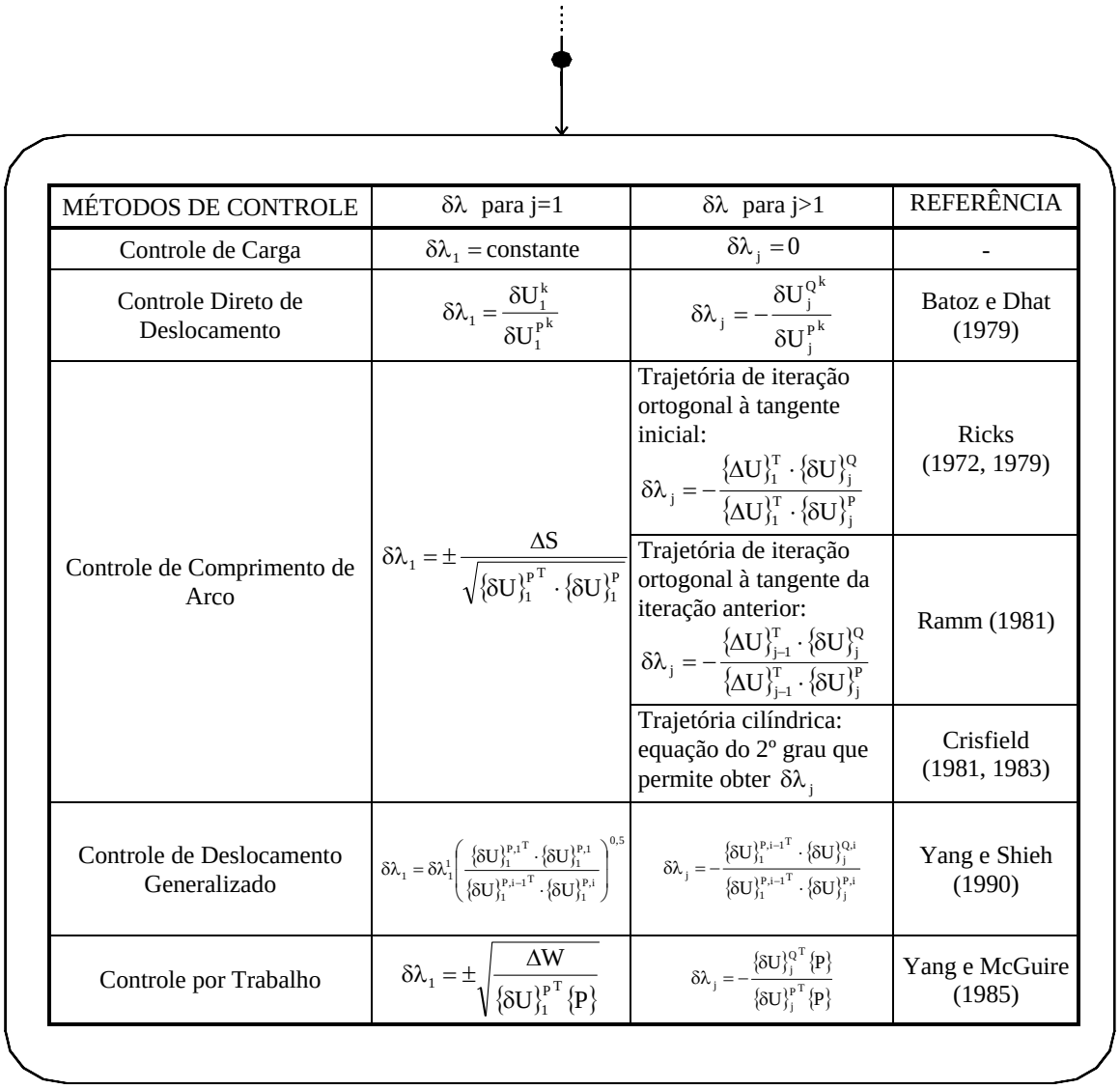


Figura 4.5: Diagrama de atividades do algoritmo genérico para métodos de controle (Fuina, 2004).



MÉTODOS DE CONTROLE	$\delta\lambda$ para j=1	$\delta\lambda$ para j>1	REFERÊNCIA
Controle de Carga	$\delta\lambda_1 = \text{constante}$	$\delta\lambda_j = 0$	-
Controle Direto de Deslocamento	$\delta\lambda_1 = \dfrac{\delta U_1^k}{\delta U_1^{P\,k}}$	$\delta\lambda_j = -\dfrac{\delta U_j^{Q\,k}}{\delta U_j^{P\,k}}$	Batoz e Dhat (1979)
Controle de Comprimento de Arco	$\delta\lambda_1 = \pm\dfrac{\Delta S}{\sqrt{\{\delta U\}_1^{P\,T}\cdot\{\delta U\}_1^{P}}}$	Trajetória de iteração ortogonal à tangente inicial: $\delta\lambda_j = -\dfrac{\{\Delta U\}_1^{T}\cdot\{\delta U\}_j^{Q}}{\{\Delta U\}_1^{T}\cdot\{\delta U\}_j^{P}}$	Ricks (1972, 1979)
		Trajetória de iteração ortogonal à tangente da iteração anterior: $\delta\lambda_j = -\dfrac{\{\Delta U\}_{j-1}^{T}\cdot\{\delta U\}_j^{Q}}{\{\Delta U\}_{j-1}^{T}\cdot\{\delta U\}_j^{P}}$	Ramm (1981)
		Trajetória cilíndrica: equação do 2º grau que permite obter $\delta\lambda_j$	Crisfield (1981, 1983)
Controle de Deslocamento Generalizado	$\delta\lambda_1 = \delta\lambda_1^1\left(\dfrac{\{\delta U\}_1^{P,1\,T}\cdot\{\delta U\}_1^{P,1}}{\{\delta U\}_1^{P,i-1\,T}\cdot\{\delta U\}_1^{P,i}}\right)^{0,5}$	$\delta\lambda_j = -\dfrac{\{\delta U\}_1^{P,i-1\,T}\cdot\{\delta U\}_j^{Q,i}}{\{\delta U\}_1^{P,i-1\,T}\cdot\{\delta U\}_j^{P,i}}$	Yang e Shieh (1990)
Controle por Trabalho	$\delta\lambda_1 = \pm\sqrt{\dfrac{\Delta W}{\{\delta U\}_1^{P\,T}\{P\}}}$	$\delta\lambda_j = -\dfrac{\{\delta U\}_j^{Q\,T}\{P\}}{\{\delta U\}_j^{P\,T}\{P\}}$	Yang e McGuire (1985)

Figura 4.6: Detalhamento do diagrama de atividades da Figura 4.5 (Fuina, 2004).

A formulação acima descrita é completamente genérica e se aplica aos vários métodos de controle, bastando que se redefina a equação de restrição. O diagrama de atividades da Figura 4.5 mostra os principais passos do algoritmo genérico para solução de equações incrementais de equilíbrio. Pode-se observar que o diagrama possui um procedimento em destaque. Este refere-se à obtenção do parâmetro de carga $\delta\lambda_j^i$, que depende do método de controle adotado.

Diferentes métodos incrementais têm sido empregados para análise não-linear de estruturas, destacando-se os métodos de controle de carga, de controle direto de deslocamento,

de controle de comprimento de arco, de controle de deslocamento generalizado, de controle por trabalho e de controle de resíduo ortogonal. A Figura 4.6 mostra um detalhamento dos parâmetros de cargas obtidos para cada método de controle nas interações j=1 e j>1.

No método de controle de carga, a carga externa é incrementada de um valor constante somente na 1ª iteração (j=1) de cada passo. Para as demais iterações (j>1), o incremento de carga é feito igual a zero, implicando num carregamento externo sempre constante. Quando a carga externa ultrapassa o valor correspondente ao ponto limite (ponto B na Figura 4.1), a linha horizontal que controla a trajetória de iteração nunca cruza a trajetória de equilíbrio e nenhum ponto de convergência pode ser obtido. Evidentemente, uma instabilidade numérica deve ocorrer próximo aos pontos limites. Assim, a utilização deste método falha na passagem por pontos limites de carga.

O método de controle direto de deslocamentos (Batoz e Dhat, 1979) supõe que as iterações são processadas a um deslocamento constante. Uma componente de deslocamento é escolhida como o parâmetro de controle. Para que possa ser escolhido adequadamente o grau de liberdade a ser usado para o controle, é necessário o conhecimento aproximado da estrutura a ser analisada. Da mesma forma que o método de controle de carga não permite a passagem por pontos limites, o controle direto de deslocamento é ineficiente se o deslocamento de controle experimenta diminuição de um nível de carga para outro, com redução de carga acompanhada de redução de deslocamentos (ponto C na Figura 4.1). Isto se deve ao fato da trajetória de iteração, controlada por uma linha vertical, nunca cruzar a trajetória de equilíbrio.

Apesar das limitações dos métodos de controle de carga e de deslocamento, estes métodos passaram a constituir um padrão para o desenvolvimento de outros métodos mais gerais e eficazes. Para solucionar as dificuldades destes métodos, o uso de combinações de deslocamentos e fator de carga, para controlar a trajetória de iteração, tem sido adotado nos métodos de controle de comprimento de arco. Nestes métodos, o processo iterativo é controlado através de uma combinação geométrica entre as variáveis deslocamentos e fator de carga proporcional. Dentre as combinações mais utilizadas destacam-se aquelas nas quais: (1) a trajetória de iteração é ortogonal à tangente inicial (Ricks (1972, 1979)), (2) a trajetória de iteração é

ortogonal à tangente da iteração anterior (Ramm, 1981) e (3) a trajetória de iteração é um arco de circunferência (Crisfield (1981, 1983)).

Além dos métodos de comprimento de arco, outros métodos, que também utilizam combinações de deslocamentos e fator de carga, têm sido adotados com êxito na solução de problemas não-lineares. Dentre estes métodos pode-se citar o de controle de deslocamento generalizado (Yang e Shieh, 1990) e o de controle de trabalho (Yang e McGuire, 1985).

Capítulo 5

PROJETO ORIENTADO A OBJETOS DO NÚCLEO NUMÉRICO INSANE

5.1 Introdução

O processador é a aplicação que representa o núcleo numérico do sistema **INSANE** e é o responsável pela obtenção dos resultados de diferentes modelos discretos de análise estrutural. Esta aplicação faz uso dos diversos conceitos do paradigma de programação orientada a objetos (classes, herança, polimorfismo, etc.) de modo a possuir a segmentação necessária ao aprimoramento progressivo do sistema.

O núcleo numérico é formado por interfaces que representam as diversas abstrações de uma resolução numérica de modelos discretos, cada qual com sua hierarquia de classes responsável por cumprir o seu devido papel no processamento. Atualmente, ele encontra-se, como mostra a Figura 5.1, organizado em função das seguintes interfaces: `Assembler`, `Solution`, `Model` e `Persistence`.

A interface `Assembler` é a responsável por montar o seguinte sistema matricial de segunda ordem com o qual é possível representar diversos tipos de problemas discretos:

$$\underline{A}\,\underline{\ddot{X}} + \underline{B}\,\underline{\dot{X}} + \underline{C}\,\underline{X} = \underline{R} - \underline{F} \tag{5.1}$$

onde $\underline{X}$ é o vetor de variáveis de estado do problema; $\underline{\dot{X}}$ e $\underline{\ddot{X}}$ são os vetores com, respectivamente, a primeira e a segunda variação temporal da variável de estado; $\underline{A}$, $\underline{B}$ e $\underline{C}$ são as

matrizes dos coeficientes, que podem ou não depender da variável de estado e suas derivadas; e $\underline{R}$ e $\underline{F}$ representam os termos independentes do sistema de equações.

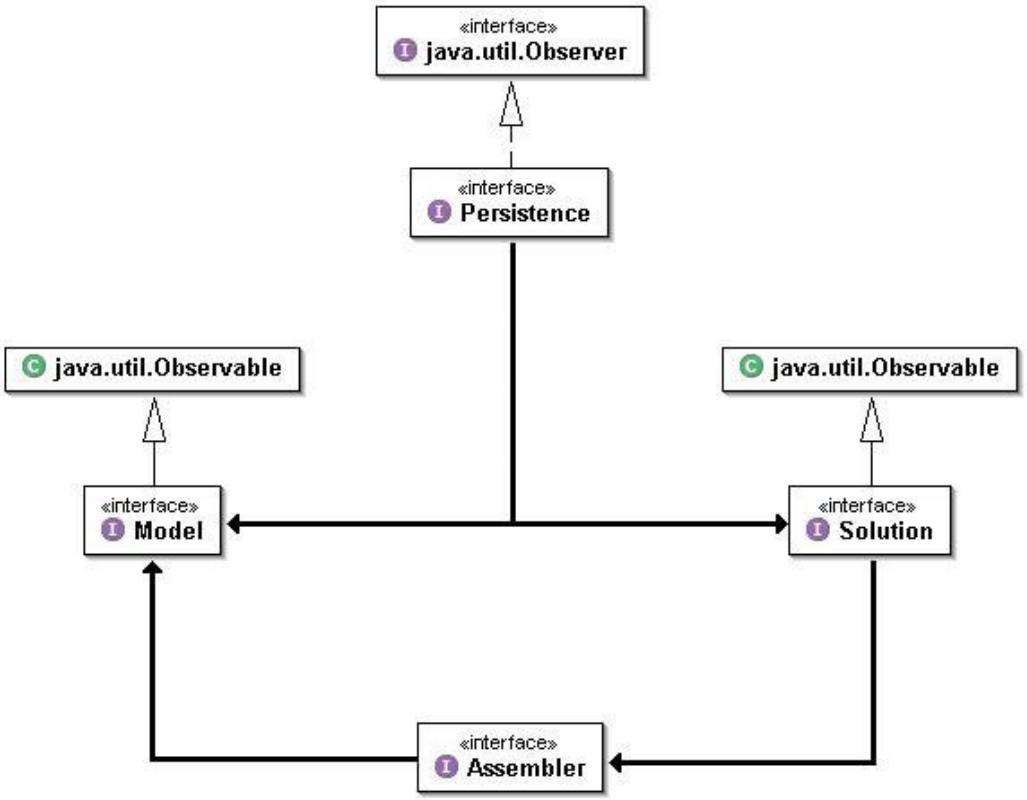


Figura 5.1: Organização do núcleo numérico do **INSANE**.

A interface `Solution`, implementada por Fuina (2006), é quem desencadeia o processo de solução, possuindo os recursos necessários para resolver este sistema matricial, seja o sistema linear ou não-linear.

`Model` possui os dados relativos ao modelo discreto a ser analisado e fornece para `Assembler` todas as informações necessárias para montar a equação do modelo, que será resolvida por `Solution`.

Tanto `Model` como `Solution` se comunicam com a interface `Persistence`, que trata os dados de entrada e, principalmente, persiste os dados de saída para as demais aplicações sempre que observa alterações no estado do modelo discreto.

Este processo de observação de alterações ocorre segundo o padrão de projeto *Observer-Observable*, o qual é um mecanismo de propagação de mudanças. Quando um objeto dito **observador** (que implementa a interface `java.util.Observer`) é criado, ele é inscrito na lista de

observadores dos objetos ditos **observados** (que estendem a classe `java.util.Observable`). Quando alguma mudança ocorre no estado de um objeto observado, é disparado então o mecanismo de propagação de mudanças, que se encarrega de notificar os objetos observadores para se atualizarem. Isto garante a consistência e a comunicação entre o componente observador (`Persistence`) e os componentes observados (`Solution` e `Model`).

Neste capítulo, é apresentado o projeto orientado a objetos de todas as interfaces que compõem o núcleo numérico do sistema **INSANE**, tendo como principal motivação a formulação proposta neste trabalho. São identificadas as principais classes e instâncias, bem como suas respectivas responsabilidades, destacando-se os atributos e métodos de maior relevância.

Para isso, são utilizados diagramas de classes, que permitem conhecer a hierarquia das mesmas e como elas se comunicam para desempenhar suas funções, e diagramas de seqüência, que mostram como as atividades ocorrem e como estas são desenvolvidas ao longo do tempo. Os diagramas seguem a proposta da *Unified Modelling Language* (UML), linguagem padronizada para a modelagem de sistemas de software orientados a objetos. Maiores informações sobre a UML e outras tecnologias de desenvolvimento de software utilizadas na implementação podem ser vistas no Apêndice A.

5.2 Interface `Assembler`

A interface `Assembler` (Figura 5.2) possui os métodos necessários para montar as matrizes e vetores do modelo conforme a Equação 5.1. Ela é implementada pela classe `FemAssembler`, que é apropriada aos diversos tipos de problemas que podem ser modelados através do Método dos Elementos Finitos. A classe `FemAssembler` tem como atributo um objeto do tipo `Model`, que é o modelo de elementos finitos para o qual deve montar a equação.

Para o método dos elementos finitos aplicado à análise estática, como é o caso da formulação discutida neste trabalho, a Equação 5.1 se resume a uma equação de primeira ordem, que é a equação de equilíbrio do modelo dada pela Equação 3.56, e pode ser representada em

termos de sub-matrizes:

$$\begin{bmatrix} \underline{C}_{uu} & \underline{C}_{up} \\ \underline{C}_{pu} & \underline{C}_{pp} \end{bmatrix} \begin{bmatrix} \underline{X}_u \\ \underline{X}_p \end{bmatrix} = \begin{bmatrix} \underline{R}_p \\ \underline{R}_u \end{bmatrix} - \begin{bmatrix} \underline{F}_p \\ \underline{F}_u \end{bmatrix}, \quad onde \tag{5.2}$$

$$\begin{bmatrix} \underline{R}_p \\ \underline{R}_u \end{bmatrix} = \begin{bmatrix} \underline{N}_p \\ \underline{N}_u \end{bmatrix} + \begin{bmatrix} \underline{E}_p \\ \underline{E}_u \end{bmatrix} \tag{5.3}$$

Correlacionando com a Equação 3.56, a matriz $\underline{C}$ é a matriz de rigidez do modelo, $\underline{X}$ o vetor de deslocamentos nodais, $\underline{R}$ é vetor de cargas nodais e $\underline{F}$ o vetor de forças nodais equivalentes aos esforços internos. O vetor $\underline{R}$ é composto por duas parcelas: o vetor $\underline{N}$ e vetor $\underline{E}$ que são, respectivamente, o vetor de forças aplicadas diretamente nos nós e o vetor de forças nodais equivalentes às cargas de corpo. Os índices u e p indicam, respectivamente, se a sub-matriz é referente a valores desconhecidos ou prescritos.

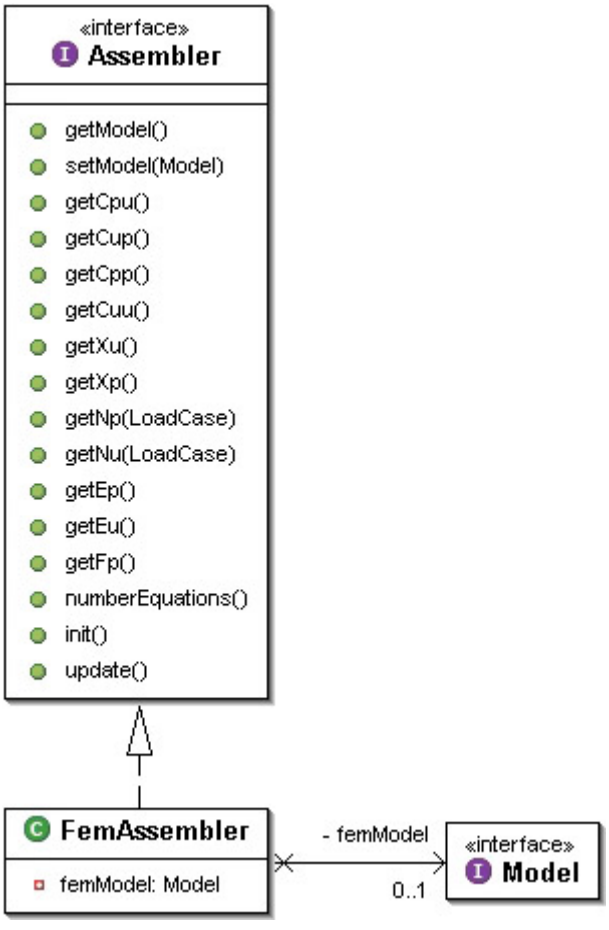


Figura 5.2: Diagrama de classe para `Assembler`.

Na Figura 5.2 estão exemplificados alguns métodos da interface `Assembler` que, quando invocados, são capazes de fornecer qualquer uma das partes deste sistema matricial. É importante ainda citar o método *numberEquations()*, que numera as equações do modelo, organizando-as segundo o critério de valores desconhecidos ou prescritos. Desta forma, `Assembler` é capaz de montar, por exemplo, a matriz de rigidez reduzida do modelo ($\underline{\underline{C}}_{uu}$) para obter os deslocamentos nodais desconhecidos ($\underline{X}_u$). Os métodos *init()* e *update()* estão relacionados com o processo da análise não-linear e, portanto, são discutidos na Seção 5.6.

5.3 Interface `Solution`

Uma vez montada a equação do problema, fica a cargo da interface `Solution` (Figura 5.3) resolvê-la. As classes que a implementam são `SteadyState`, responsável pela solução de problemas lineares, e `EquilibriumPath`, responsável pela solução de problemas não-lineares através de um processo incremental-iterativo.

Um objeto `SteadyState` possui um objeto do tipo `Assembler` e um objeto do tipo `LinearEquationSystem`, que é responsável por obter a solução de um sistema de equações algébricas lineares, como a montada no caso de uma solução estática linear (Figura 5.3).

Já a classe `EquilibriumPath` possui um objeto do tipo `Step`, que possui os métodos para resolução do processo incremental-iterativo, e uma lista de `IterativeStrategy`, informada pelo usuário, que define os métodos de controle para obtenção das trajetórias de equilíbrio a serem utilizados. `EquilibriumPath` não possui um objeto do tipo `Assembler`, porém possui o método *setAssembler(*`Assembler`*)* que atribui ao seu objeto `Step` um objeto `Assembler` para o qual deve resolver o processo incremental-iterativo. Possui ainda o método *execute()* que terá sua seqüência de atividades discutida na Seção 5.6. `EquilibriumPath` estende a classe `Observable`, uma vez que é observada pela persistência, e implementa a interface `Observer`, uma vez que é observadora das mudanças de estado de seu objeto `Step`.

A Figura 5.4 mostra o diagrama de classe para a interface `Step`, implementada pela classe `StandardNewtonRaphson`, onde o método de Newton Raphson Padrão é empregado durante o processo incremental-iterativo da solução não-linear.

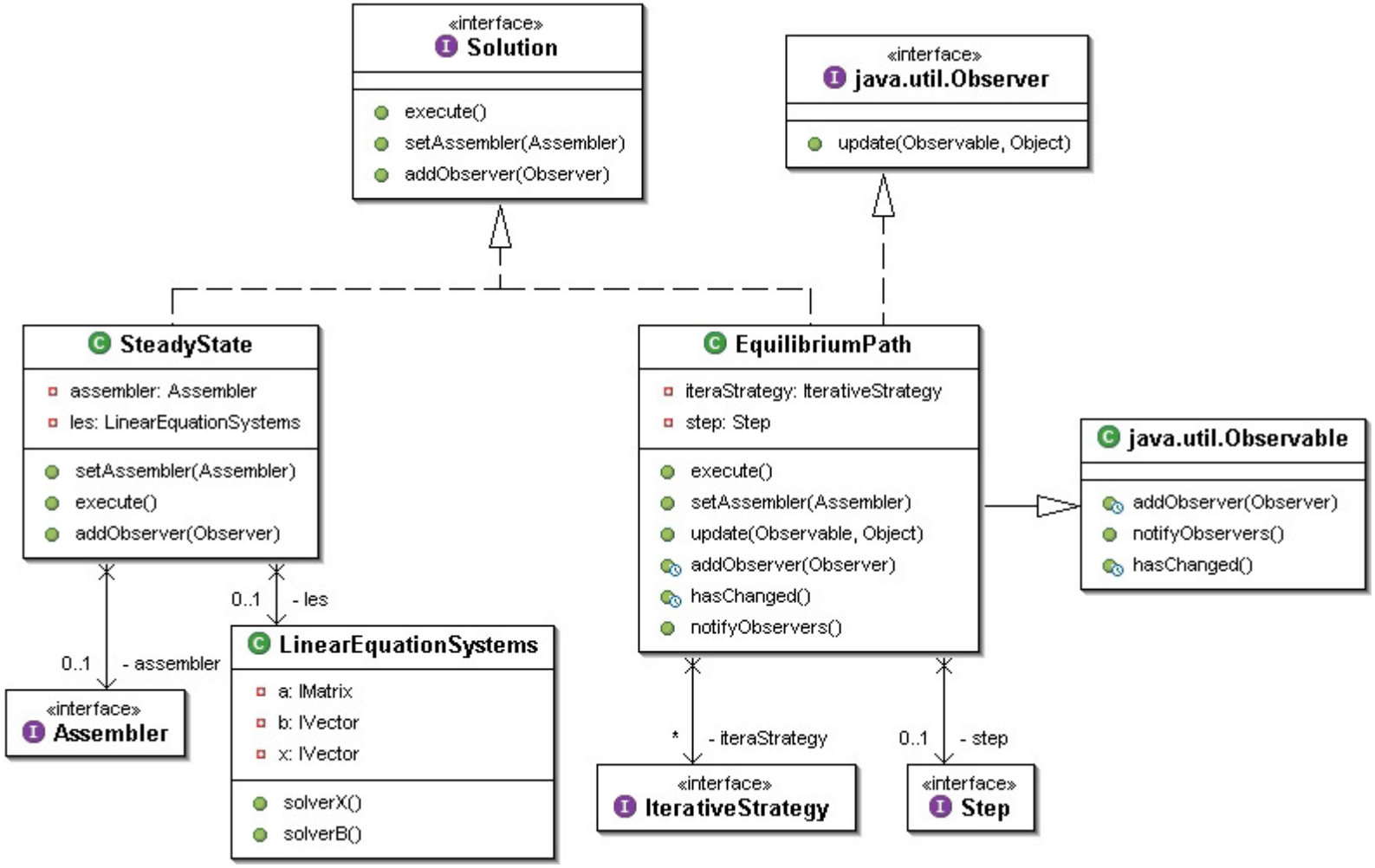


Figura 5.3: Diagrama de classe para `Solution`.

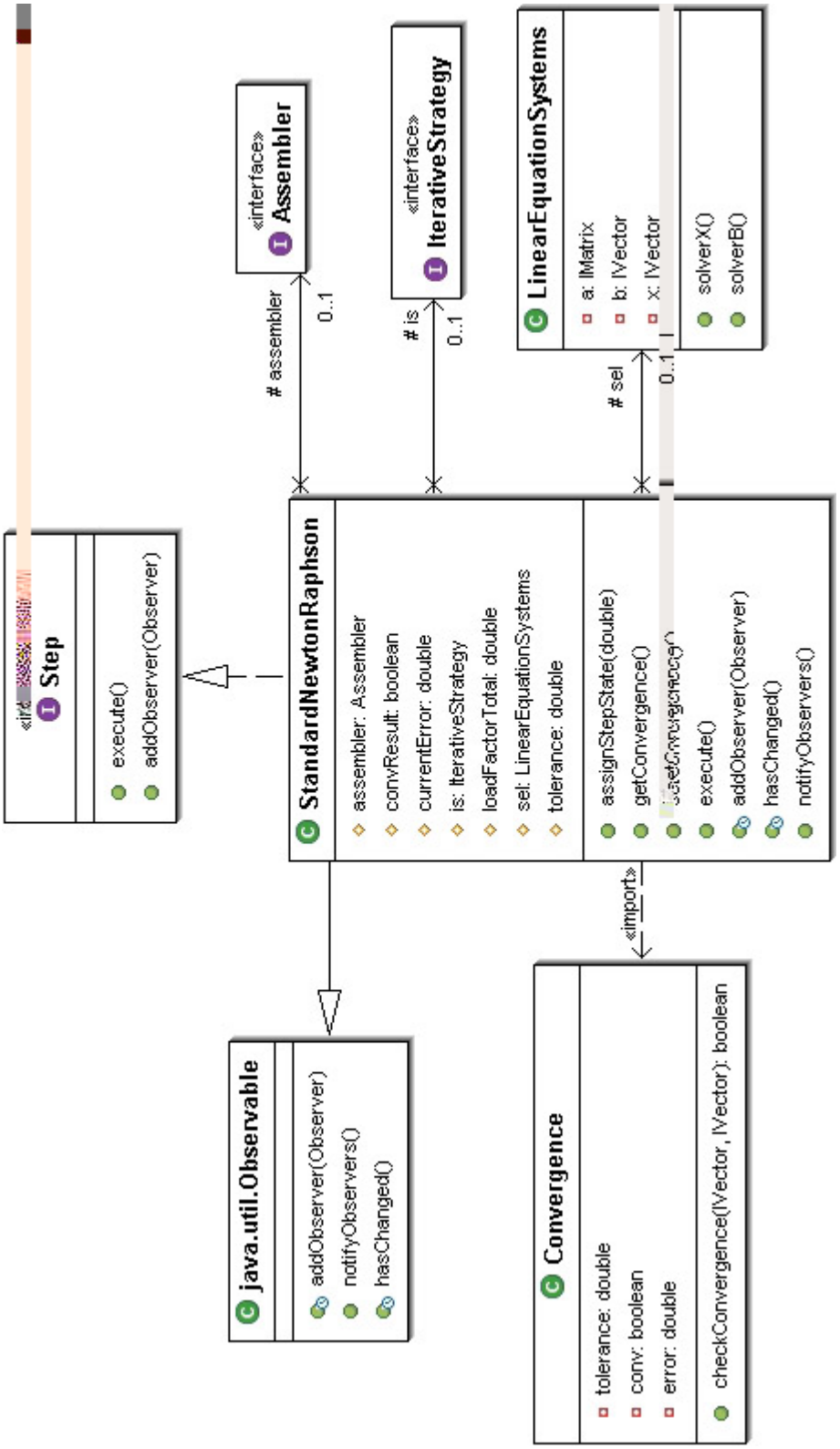


Figura 5.4: Diagrama de classe para `Step`.

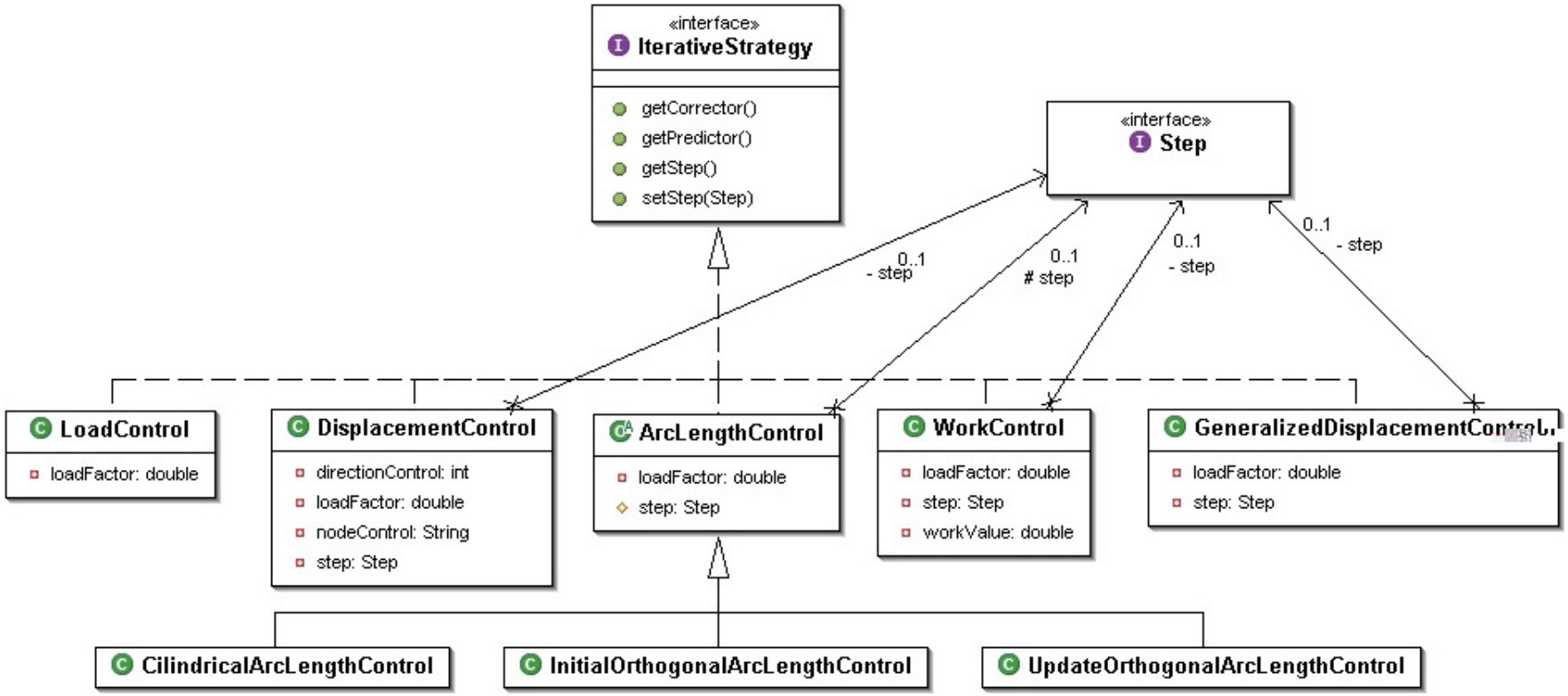


Figura 5.5: Diagrama de classe para `IterativeStrategy`.

`StandardNewtonRaphson` tem todos os atributos necessários para obter a convergência no passo, destacando-se um objeto `Assembler`, capaz de informar as matrizes e vetores da equação a ser resolvida em cada iteração do passo; um objeto `LinearEquationSystem`, capaz de resolver o sistema de equações algébricas lineares de cada iteração; e um objeto `IterativeStrategy`, que representa a estratégia de iteração corrente. `StandardNewtonRaphson` também estende a classe `Observable`, sendo observada por `EquilibriumPath`.

A convergência é verificada através do método *getConvergence()* e *setConvergence()* de `StandardNewtonRaphson`. *setConvergence()*, quando invocado, cria um objeto `Convergence`, que, por meio do método *checkConvergence()*, calcula a convergência baseada em força, deslocamento, ou ambos, conforme informado pelo usuário.

O diagrama de classe da interface `IterativeStrategy` é representado na Figura 5.5. Em sua hierarquia estão as classes que representam os vários métodos de controle implementados por Fuina (2006): método de controle de carga, método de controle direto de deslocamento, método de controle por trabalho, método de controle de deslocamento generalizado e método de controle de comprimento de arco e suas particularizações. Cada uma destas classes possui os métodos *getPredictor()* e *getCorrector()* que calculam o fator de carga da primeira iteração e das demais iterações, respectivamente, conforme a Figura 4.6 do Capítulo 4. Exceto a classe `LoadControl`, todas as outras classes possuem um objeto do tipo `Step`, pois precisam deste para saber informações sobre o passo anterior durante o cálculo do fator de carga.

5.4 Interface `Model`

O modelo discreto a ser analisado é representado no projeto orientado a objetos do núcleo numérico pela interface `Model` (Figura 5.6). Uma classe que implemente `Model`, para representar da forma mais geral possível um modelo discreto, é constituída por listas de objetos inerentes a este modelo e contém métodos de acesso e manipulação destas listas.

`Model` é implementada pela classe `FemModel` que representa o modelo de elementos finitos propriamente dito. Um objeto `FemModel` tem listas de nós, elementos, funções de forma, ordens

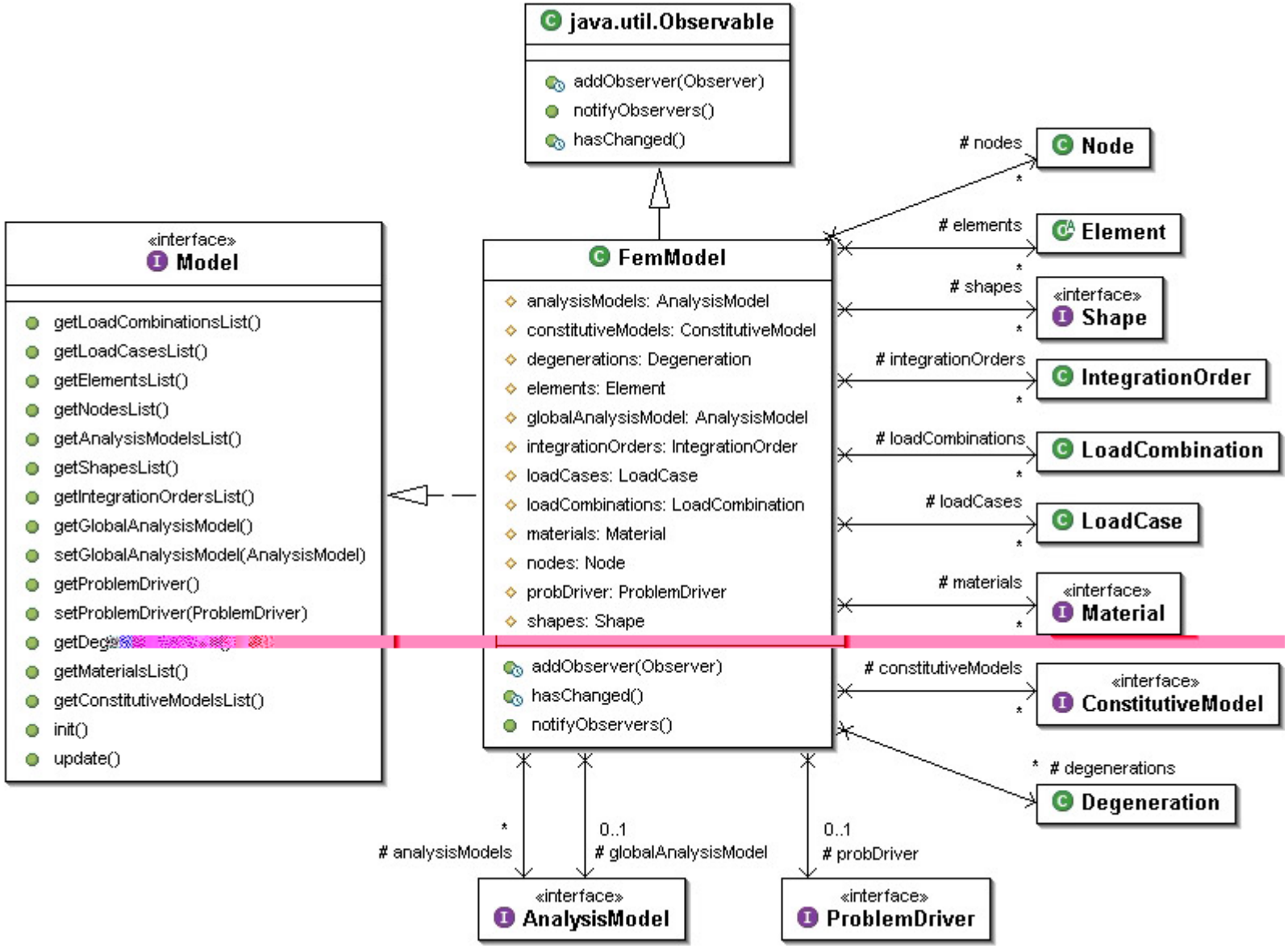


Figura 5.6: Diagrama de classe para `Model`.

de integração, casos de carga, combinações de carregamento, modelos de análise, materiais, modelos constitutivos e degenerações (o conceito de degeneração será discutido adiante). Tem ainda dois atributos: um modelo de análise global do tipo `AnalysisModel` e um objeto `ProblemDriver`. `FemModel` estende também a classe `Observable`, pois é observada pela persistência.

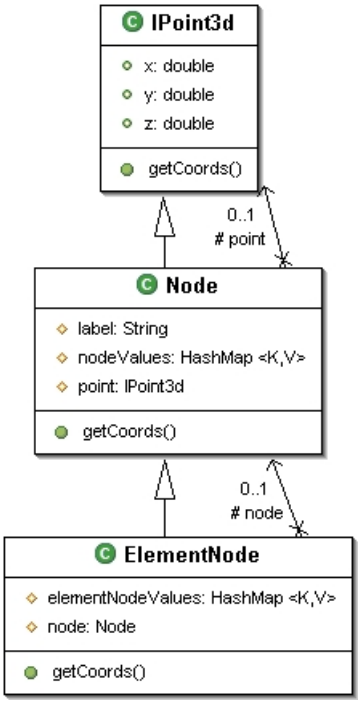


Figura 5.7: Diagrama de classe para `Node`.

Juntamente à interface `Model` estão implementadas as classes `Node`, `Element` e `ProblemDriver`. A Figura 5.7 mostra o diagrama de classe para `Node`, que possui um *label* e uma coleção de valores do tipo `java.util.HashMap` que armazena as variáveis do nó como, por exemplo, deslocamentos e forças. Além de estender a classe `IPoint3d` e conseqüentemente possuir todos os métodos e atributos desta classe, `Node` tem também como atributo um objeto deste tipo para representar um ponto no espaço. Os métodos herdados são sobrecarregados para que acessem diretamente o objeto `IPoint3d`, e não os atributos herdados. Por exemplo, o método *getCoords()* de `Node` acessa o objeto do tipo `IPoint3d` e invoca seu método *getCoords()*, obtendo diretamente as coordenadas do ponto no espaço que representa o nó. Este encadeamento permite a reutilização do código e simplifica a manipulação de um objeto `Node`.

Da mesma forma, como especialização do nó, a classe `ElementNode` (Figura 5.7) estende

`Node` e também possui um objeto `Node`, além de uma coleção de valores do tipo `HashMap`, que armazena as forças equivalentes às cargas de corpo do elemento. `ElementNode` sobrecarrega os métodos herdados, acessando diretamente os atributos de seu objeto `Node`. Como exemplo, seu método *getCoords()* invoca o método *getCoords()* de seu objeto `Node`, obtendo, através do encadeamento, as coordenadas do nó.

A classe `Element` representa os elementos finitos e é estendida pela classe `Parametric-Element` que representa os elementos finitos paramétricos. Esta, por sua vez, é estendida por outras classes representando os vários tipos de elementos finitos paramétricos, como mostrado na Figura 5.8. Um objeto `Element` tem como atributos uma lista de `ElementNode`, que representa sua incidência, uma lista de `Degeneration`, que representa seus pontos de integração e sua constituição geométrica e física, um objeto `AnalysisModel`, que representa seu modelo de análise, um objeto `Shape`, que representa sua função de forma, um objeto `Constitutive-Model`, que representa seu modelo constitutivo, e um objeto `ProblemDriver`, que armazena informações relativas ao tipo de problema que o elemento modela.

A interface `ProblemDriver` (Figura 5.9) possui os métodos necessários para informar a `Assembler` as parcelas de cada elemento na equação do modelo, sejam elas incrementais ou totais. Em sua hierarquia são representados diversos tipos de problemas que podem ser resolvidos através de modelos discretos, sendo o caso aqui estudado um problema fisicamente não-linear de mecânica dos sólidos com elementos finitos paramétricos, representado pela classe `ParametricPhysicallyNonLinearSolidMech`. Desta maneira, a classe `Element` é uma classe bastante geral, independente do problema que deve ser capaz de representar, mas que é capaz de passar todas as informações necessárias ao modelo.

A classe `ParametricElement` possui, além dos atributos de `Element`, um objeto `Integra-tionOrder` que representa a sua ordem para integração numérica. Em sua hierarquia estão as classes que representam os elementos finitos paramétricos, separadas de acordo com sua geometria: elementos de barra, elementos planos triangulares e quadrilaterais, e elementos sólidos tetraédricos e hexaédricos. Estas sub-classes implementam os métodos relativos à integração numérica (*addDegenerations(*`Degeneration`*)* e *initDegenerations()*), discutidos na Seção 5.6.

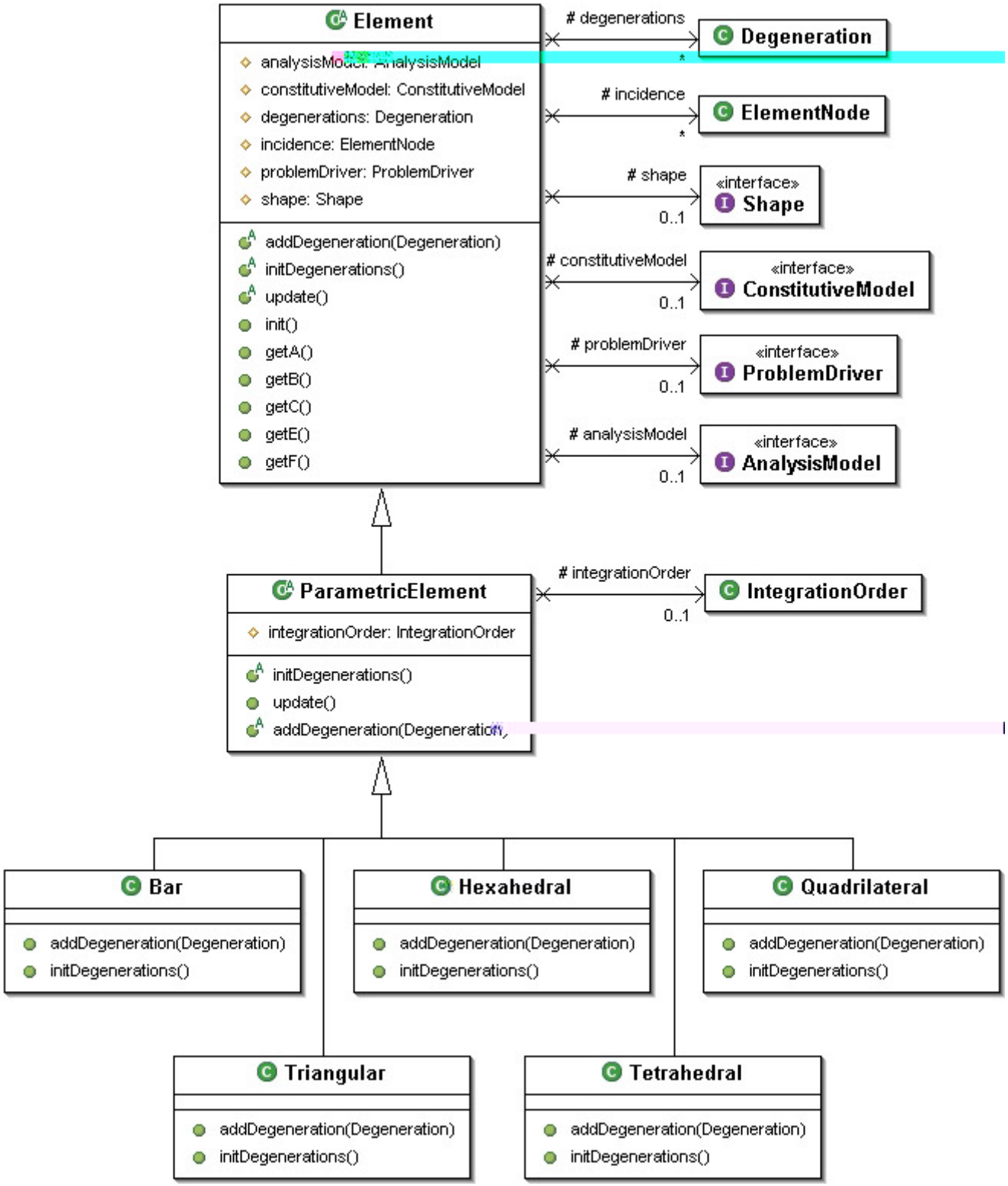


Figura 5.8: Diagrama de classe para `Element`.

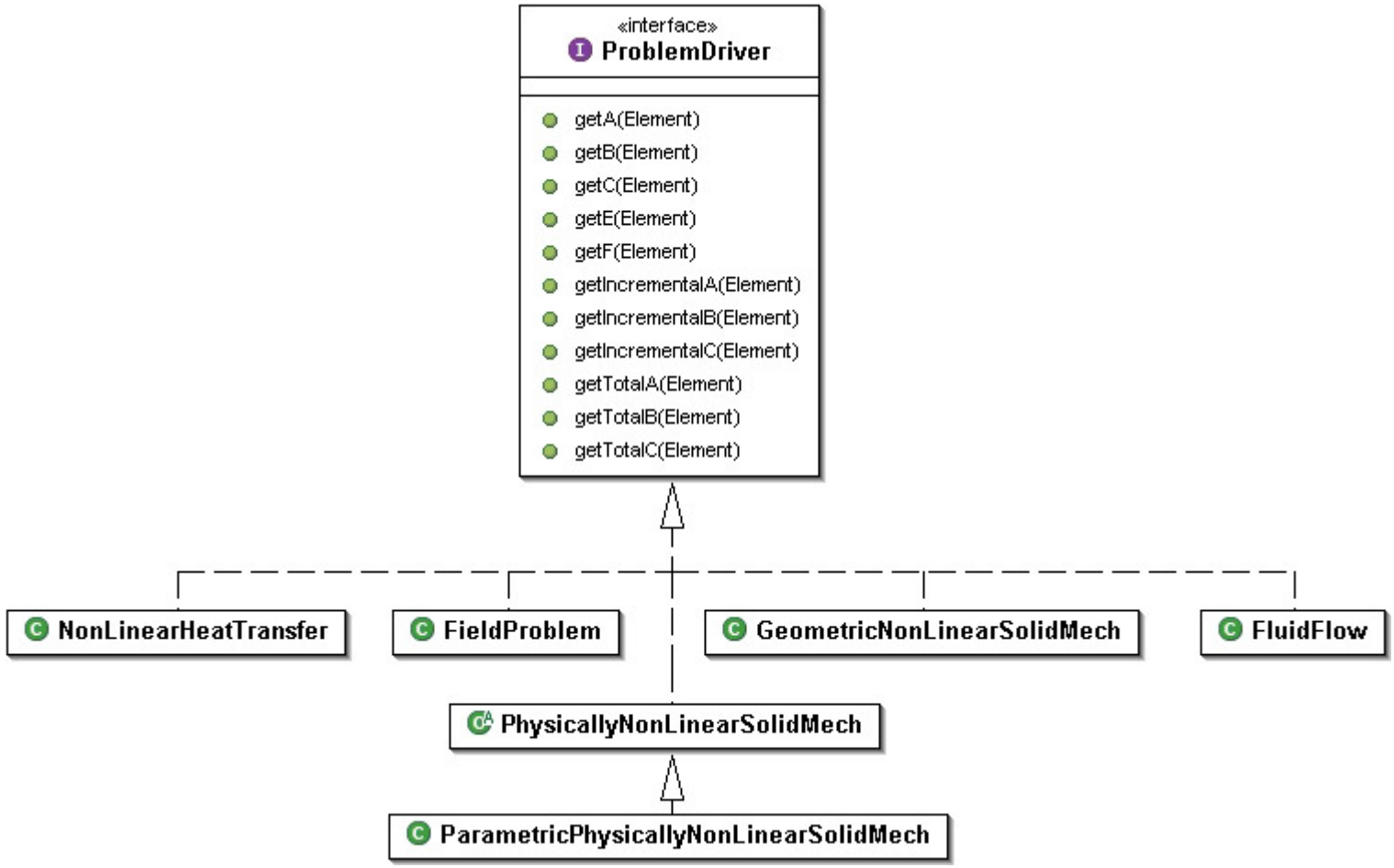


Figura 5.9: Diagrama de classe para `ProblemDriver`.

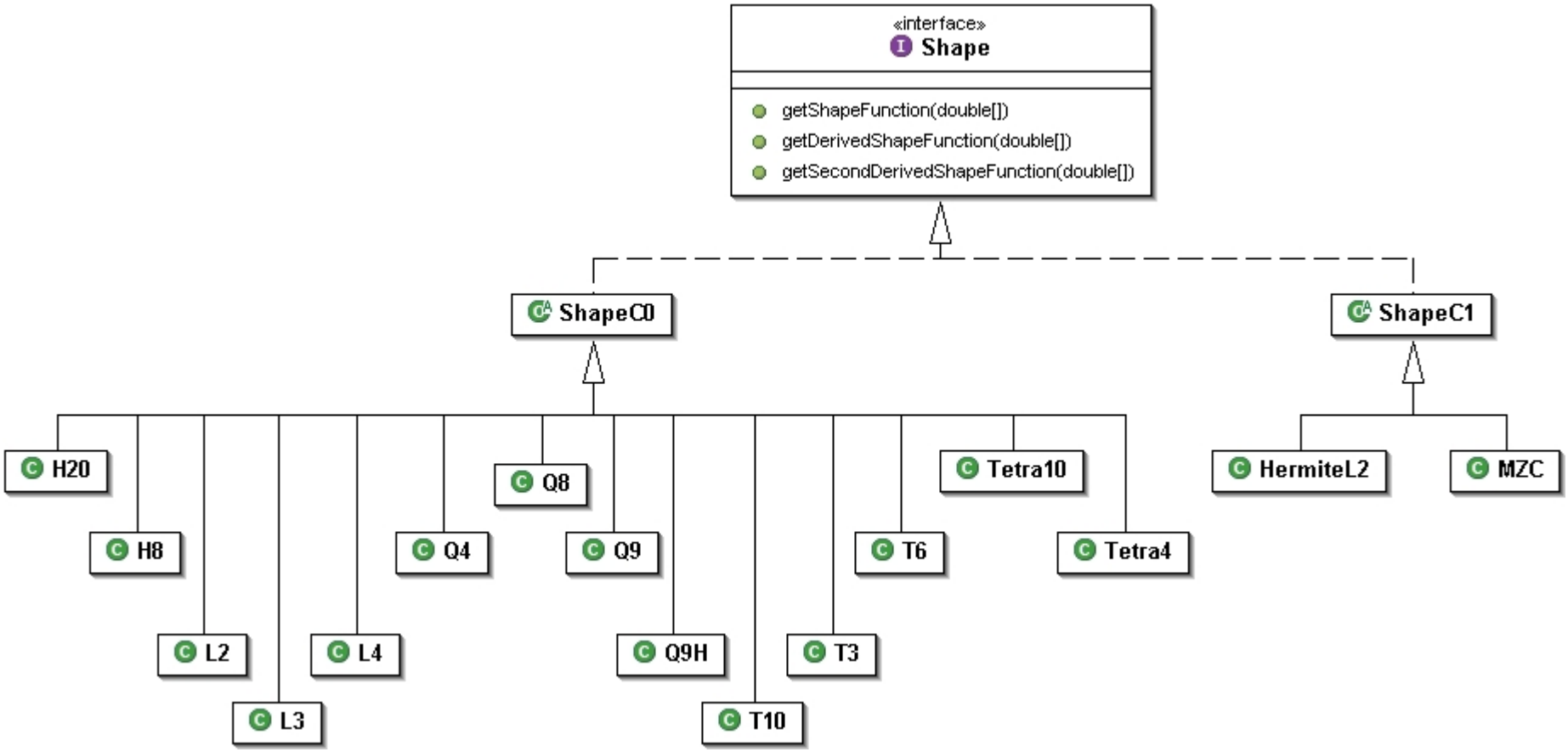


Figura 5.10: Diagrama de classe para `Shape`.

5.4.1 Interface `Shape`

As funções de aproximação ou funções de forma dos elementos finitos estão agrupadas na hierarquia da interface `Shape` (Figura 5.10), que possui métodos responsáveis por fornecer as funções de forma, suas primeiras derivadas e suas segundas derivadas. A hierarquia é dividida segundo a continuidade dos elementos finitos por quem são utilizadas, se elas são de continuidade C0, como, por exemplo, o elemento de pórtico espacial de Timoshenko, ou continuidade C1, como, por exemplo, o elemento de pórtico espacial de Euler-Bernoulli. Em um segundo nível da hierarquia, há as classes que representam as funções de forma de acordo com a geometria do elemento e o seu número de nós. Isso mostra mais uma vez a generalidade da classe `Element`, que não depende da continuidade da função de forma e nem do número de nós que possui.

5.4.2 Pacote `MaterialMedia`

O pacote `MaterialMedia` contém interfaces e classes necessárias para representar da forma mais geral possível a constituição geométrica e física dos elementos finitos.

A interface `Material` (Figura 5.11) possui os métodos necessários para obter informações sobre as propriedades dos diferentes materiais, sejam elas propriedades secantes, tangentes ou de descarregamento. Os diferentes materiais são representados pelas classes que implementam a interface `Material`, e que têm como atributos seu *label* e uma coleção do tipo `HashMap` com os valores necessários para caracterizar aquele material. A Figura 5.11 mostra o diagrama de classe da interface `Material` e algumas de suas sub-classes implementadas para os materiais descritos na Seção 4.2.

Existe, ainda, a classe `MaterialPoint` (Figura 5.12) que tem o papel de representar um ponto no meio material e que tem como propriedades um *label*, um objeto `IPoint3d`, que o representa como um ponto no espaço, um objeto `IVolume`, que representa sua propriedade geométrica, um objeto `Material`, um objeto `AnalysisModel` e um objeto `ConstitutiveModel`. Possui ainda duas coleções do tipo `HashMap` que armazenam dois tipos de variáveis dependentes do seu modelo constitutivo: as variáveis constitutivas atuais e as variáveis constitutivas prévias.

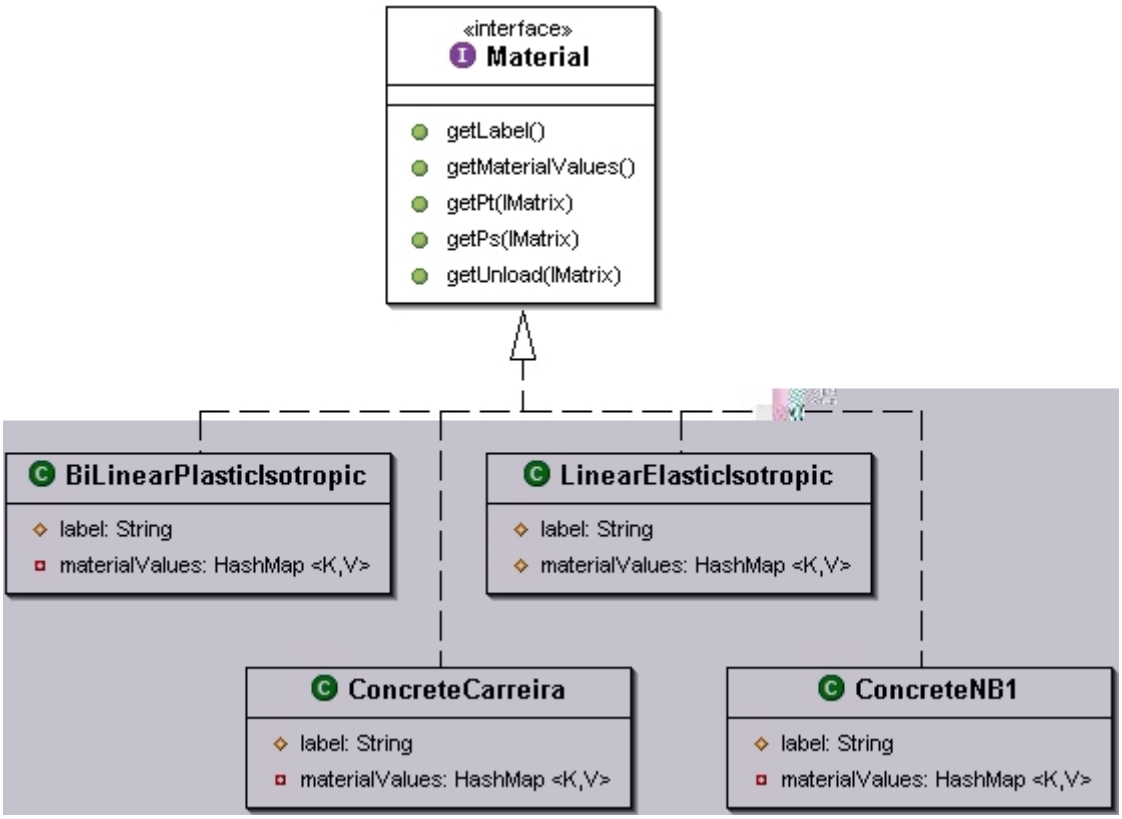


Figura 5.11: Diagrama de classe para `Material`.

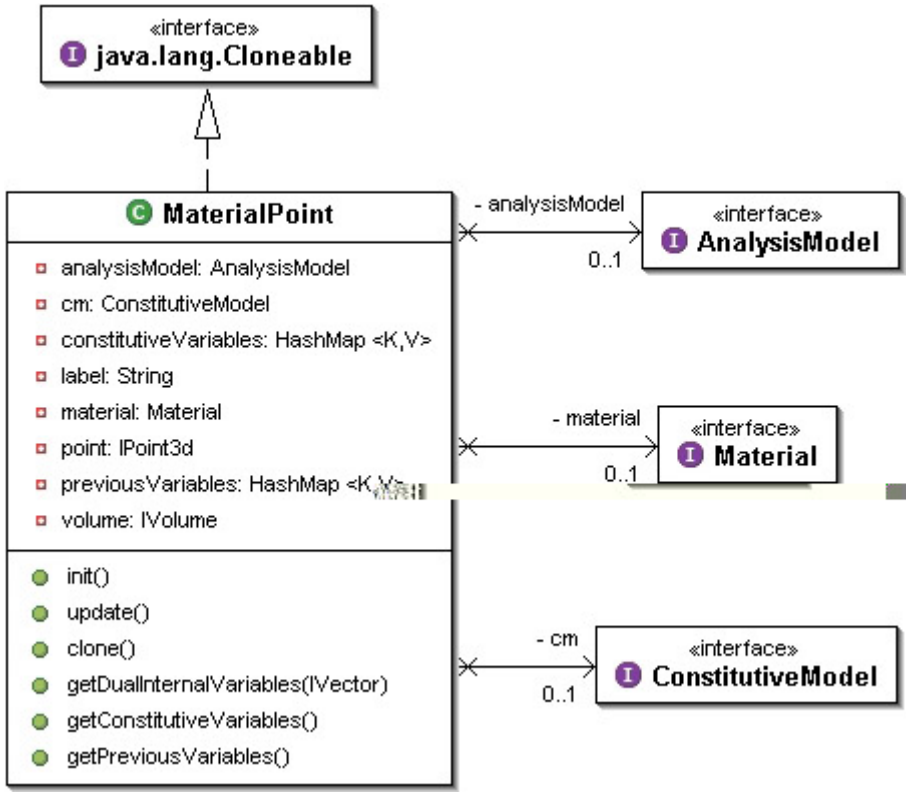


Figura 5.12: Diagrama de classe para `MaterialPoint`.

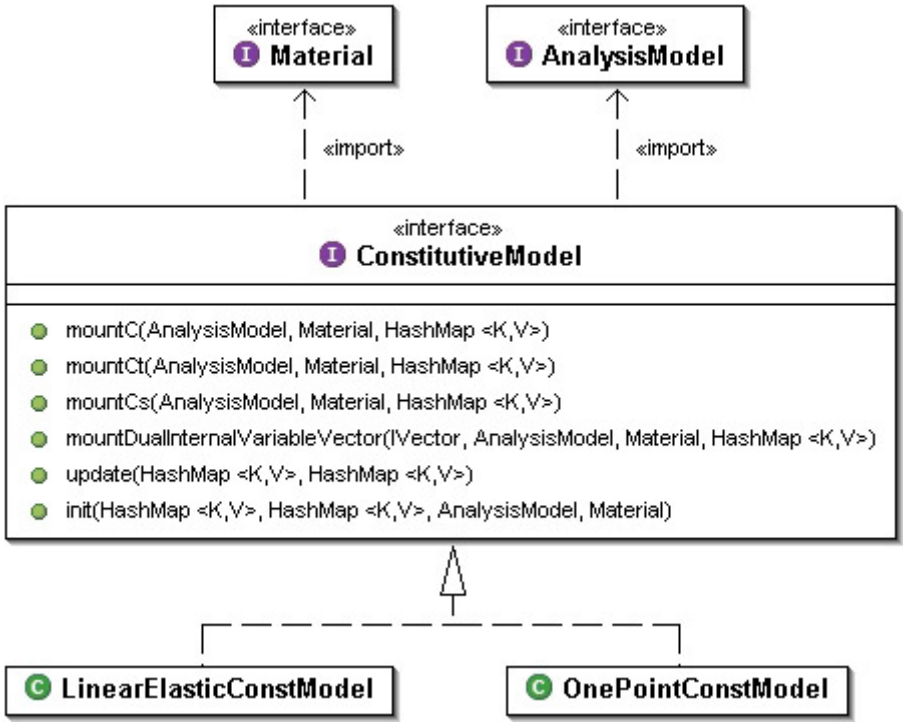


Figura 5.13: Diagrama de classe para `ConstitutiveModel`.

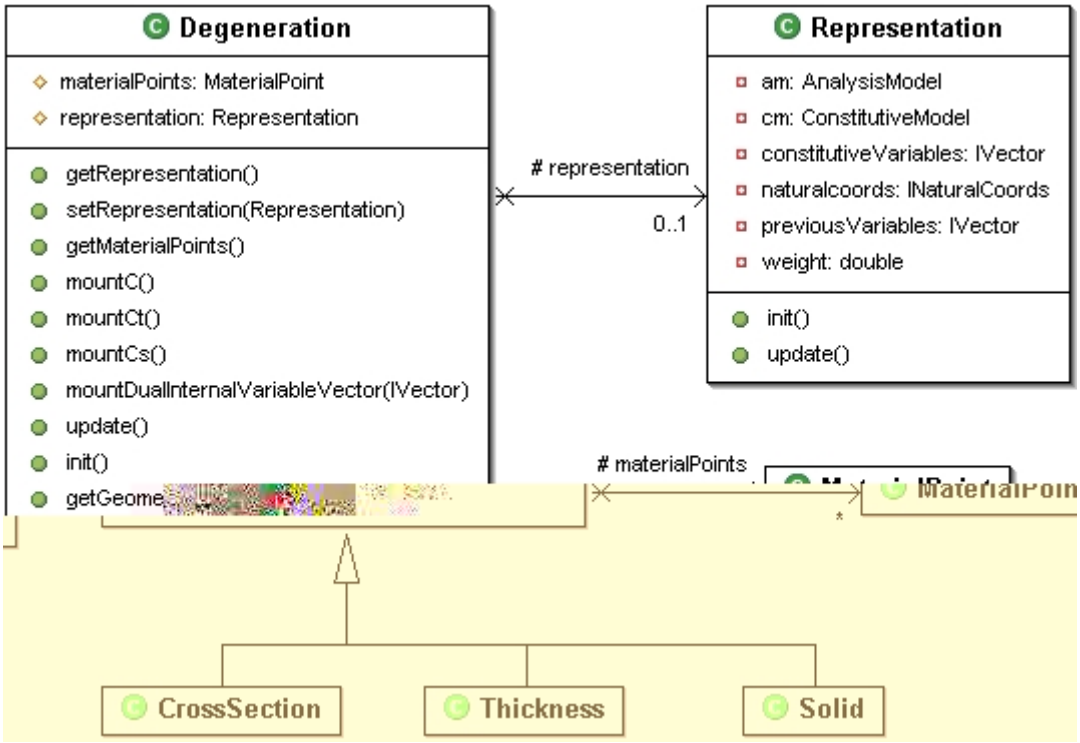


Figura 5.14: Diagrama de classe para `Degeneration`.

Estas duas coleções são capazes de informar, durante a análise, o estado do ponto material no momento corrente e no passo anterior. Um objeto `MaterialPoint` implementa a interface `java.lang.Cloneable`, possuindo a propriedade de ser clonado de forma seletiva. Esta propriedade permite a cópia de somente alguns atributos de um objeto `MaterialPoint`, através da implementação do método *clone()*.

A Figura 5.13 mostra o diagrama de classe da interface `ConstitutiveModel`, responsável por montar as matrizes constitutivas e calcular as tensões das degenerações e dos pontos materiais através de informações de suas geometrias, materiais e modelos de análise. Em sua hierarquia, estão implementados os modelos constitutivos para o caso linear elástico e para pontos de uma seção transversal. Um objeto do tipo `ConstitutiveModel` monta tanto as matrizes constitutivas secantes e tangentes quanto o vetor de tensões referentes às variáveis de estado correntes. É importante ressaltar a importância da dependência do modelo constitutivo quanto aos materiais e modelos de análise, pois sem essas informações não é possível obter as matrizes constitutivas.

Finalmente, a classe `Degeneration` (Figura 5.14) representa a degeneração na geometria do elemento. As degenerações são compostas por uma lista de pontos materiais e são representadas por objetos do tipo `Representation`. Um exemplo de `Degeneration` é a classe `CrossSection` que representa a degeneração causada pela discretização em elementos finitos unidimensionais, na qual simplifica-se uma geometria tridimensional em apenas uma linha, como mostra a Figura 5.15. Perdem-se as informações sobre a seção transversal (as duas dimensões que foram simplificadas), principalmente se ela for de geometria qualquer e composta. A classe `CrossSection` contém uma lista de pontos materiais fornecida pelo usuário que descreve de maneira aproximada a geometria e a composição da seção, permitindo que o modelo constitutivo dessa degeneração seja aproximado pelo somatório da contribuição de cada um dos pontos. Para cada ponto de integração do elemento há uma degeneração, o que permite aproximar o modelo constitutivo ao longo do elemento e assim obter a sua matriz de rigidez.

Um objeto `Representation` (Figura 5.14), para representar uma degeneração, possui um modelo constitutivo, um modelo de análise e dois vetores, um com suas variáveis constitutivas

e outro com as variáveis prévias. Estes atributos dizem respeito à degeneração como um todo, generalizando para o conjunto de pontos materiais. A representação cumpre ainda o papel de ponto de integração numérica, possuindo os devidos atributos necessários à integração numérica.

Figura 5.15: Degeneração da geometria de um elemento finito unidimensional.

Quando solicitado por seu elemento, um objeto `Degeneration` monta, através de sua lista de pontos materiais e de sua representação, tanto as suas matrizes constitutivas secante e tangente quanto o seu vetor de tensões referente às variáveis de estado correntes.

5.4.3 Interface `AnalysisModel`

A interface `AnalysisModel` (Figura 5.16) possui os métodos para fornecer as informações, dependentes do modelo de análise, necessárias aos elementos finitos, aos pontos materiais e às representações. Ela é implementada por classes representantes dos diversos modelos de análise, tanto no nível do elemento quanto no nível de seus pontos materiais. Por exemplo, para um elemento com o modelo de análise do tipo `TimoSpaceFrame` ou `EulerSpaceFrame`, são objetos destas classes que informam ao elemento o número de graus de liberdade para cada nó. Em uma degeneração do elemento, o número de deformações generalizadas é informado pelo modelo de análise da representação, que é sempre o mesmo do elemento, já que ela representa um ponto de integração do mesmo. Em um ponto material pertencente a uma degeneração `CrossSection`, cujo modelo de análise é `TimoPoint` ou `EulerPoint`, são objetos destas classes que informam os operadores necessários para obtenção das matrizes constitutivas em cada ponto.

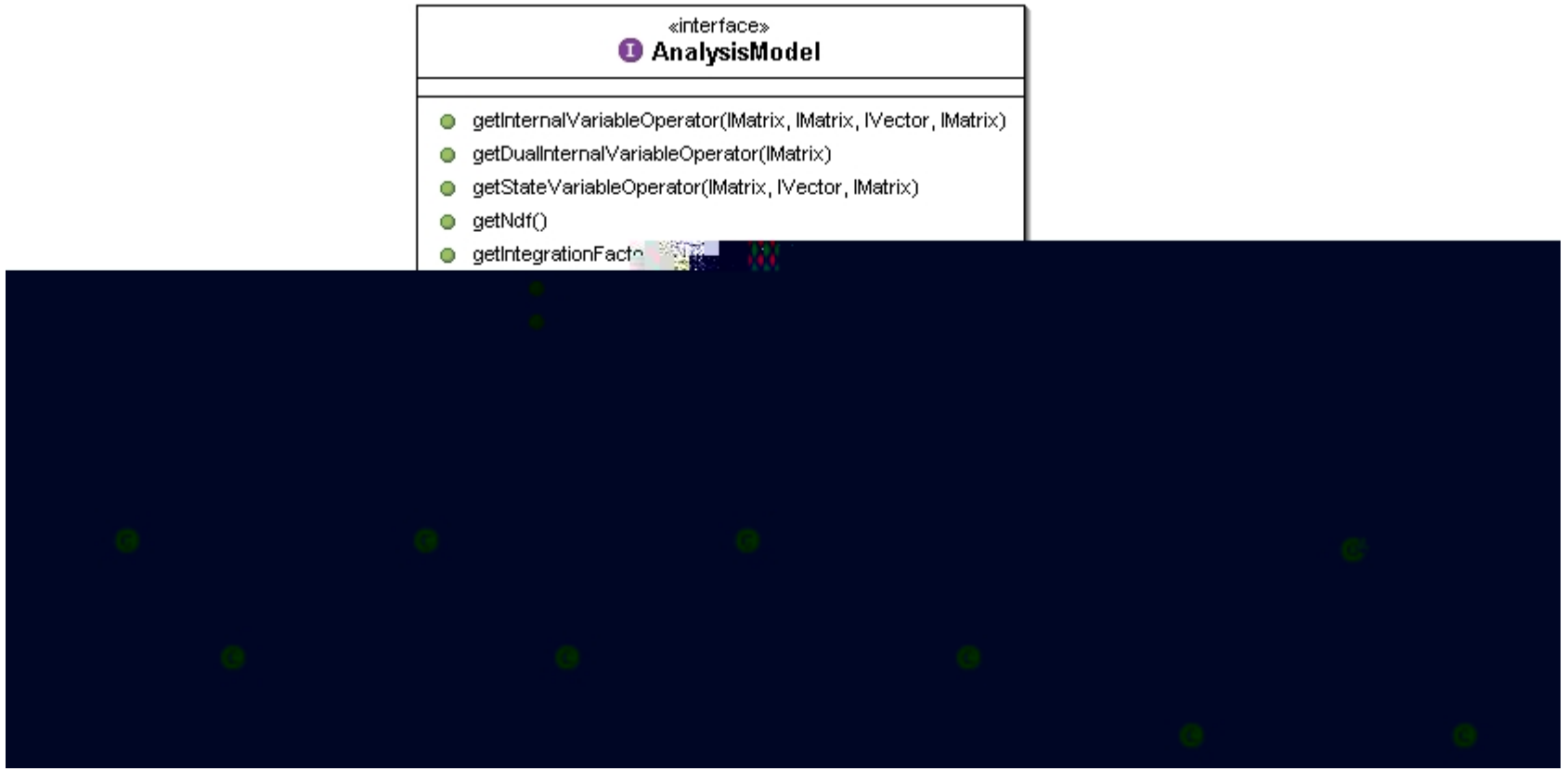


Figura 5.16: Diagrama de classe para `AnalysisModel`.

5.5 Interface `Persistence`

A função da interface `Persistence` (Figura 5.17) é receber dados de entrada através de arquivos, que podem ter sido gerados, por exemplo, por uma aplicação de pré-processamento; e gerar arquivos através de dados de saída do núcleo numérico para posterior utilização nas demais aplicações do sistema, como por exemplo, um pós-processador. Uma vez gerados, os dados necessários à descrição do modelo preenchem o objeto `Model` e os dados necessários à descrição da solução preenchem o objeto `Solution`. Isto torna possível ao objeto `Assembler` montar as matrizes e vetores das equações do modelo e ao objeto `Solution` resolvê-las. Os resultados obtidos são persistidos então em arquivos de saída, sempre que ocorre uma alteração no estado do modelo.

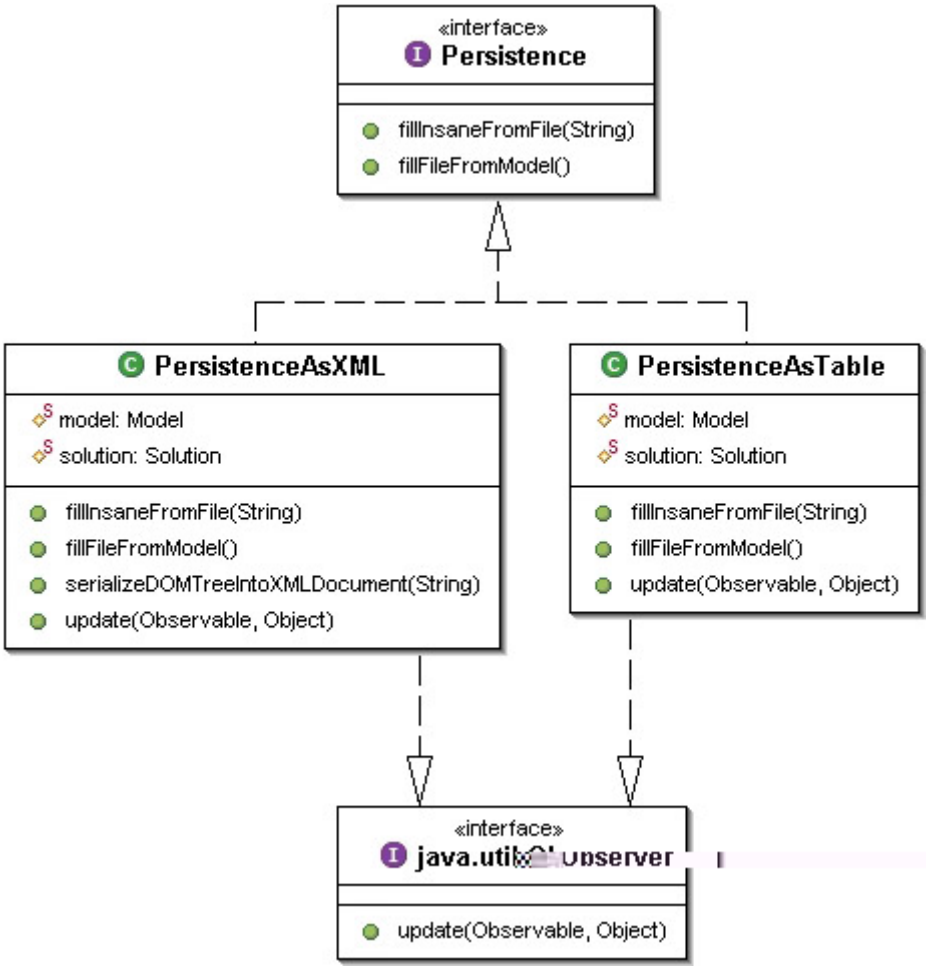


Figura 5.17: Diagrama de classe para `Persistence`.

A interface possui métodos que cumprem os papéis de persistência de entrada e de saída e é particularizada segundo o tipo de arquivo a persistir (Figura 5.17). Um objeto do tipo `Persistence` possui um atributo do tipo `Model` e outro do tipo `Solution`, que são os objetos com os quais se comunicam. E ainda implementa a interface `Observer`, sendo componente observador do mecanismo de propagação de mudanças segundo o padrão *Observer-Observable*.

A persistência de dados entre as aplicações mais utilizada é baseada em arquivos XML, sigla esta que é uma abreviação de *eXtensible Markup Language*, ou seja, linguagem de marcação estendida. A XML permite criar dados estruturados, baseada em um arquivo texto, o que difere de outras linguagens de marcação, como a HTML, pelo fato das regras de marcação serem definidas pelo programador. A opção de fazer a persistência dos dados em arquivos XML e a segmentação propiciada pela utilização de programação orientada a objetos permitirá que o sistema, ou partes deste, possa futuramente ser utilizado através da internet, uma vez que a XML vem sendo adotada como padrão para troca de documentos através da rede (Pitangueira et al., 2004).

5.6 Seqüência de Atividades

Os processos do núcleo numérico, essenciais para a obtenção de resultados de um modelo de elementos finitos paramétricos em um problema não-linear de análise estrutural, são apresentados a seguir. Para isto são utilizados diagramas de seqüência, que mostram como ocorrem os processos e como os objetos interagem ao longo do tempo.

As primeiras atividades a serem apresentadas são do método *init()*, que inicializa as variáveis do modelo para sua conseguinte resolução. Em seguida, são apresentadas as atividades do método de execução do processo incremental-iterativo da solução, do método para obtenção da matriz de rigidez incremental de um elemento finito paramétrico e do método para obtenção do vetor de forças nodais equivalentes aos esforços internos de um elemento finito paramétrico. Por último, as atividades do método *update()*, que atualiza as variáveis do modelo quando é obtida a solução de um passo do processo incremental-iterativo.

5.6.1 Método *init()*

Após o objeto `Persistence` preencher o modelo de elementos finitos, quem dispara o processo de inicialização das variáveis do modelo é a interface `Solution`. Ressalta-se aqui que os objetos do modelo são criados e inicializados pelo método *fillInsaneFromFile()* da interface `Persistence`, e que o método *init()* trata da inicialização das variáveis dos objetos do modelo.

`Solution`, antes de começar o processo de solução, solicita ao seu objeto `FemAssembler` que inicialize suas variáveis. `FemAssembler` invoca então o seu método *numberEquations(),* que numera as equações do seu objeto `FemModel`, e solicita que este inicialize suas variáveis, como pode ser visto no diagrama da Figura 5.18. `FemModel` acessa sua lista de `Element` e solicita que todos inicializem suas variáveis, como mostrado na Figura 5.19.

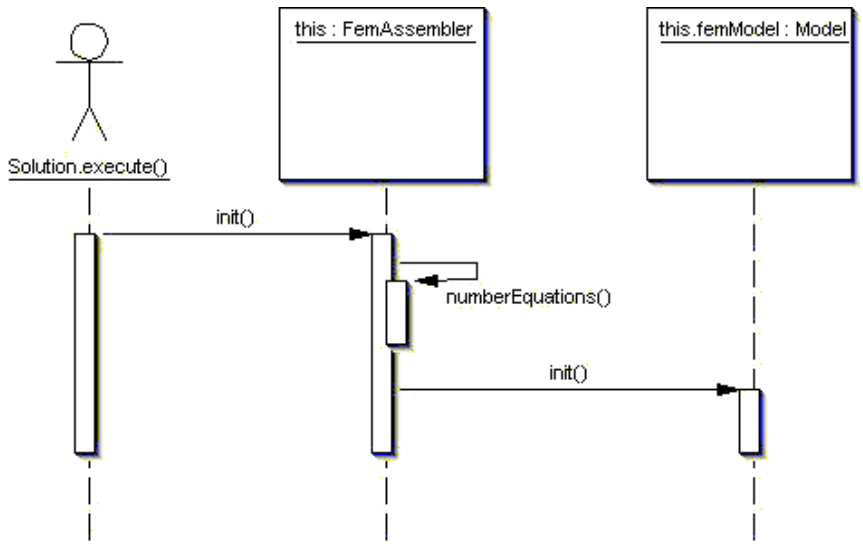


Figura 5.18: Diagrama de seqüência para *FemAssembler.init()*.

O elemento paramétrico, quando criado pela persistência, é informado de seu número de pontos de integração e também tem atribuídas tantas degenerações quantos pontos de integração. Isto é feito através do método *addDegenerations(*`Degeneration`*)*, que clona a lista de pontos materiais de uma degeneração da lista do modelo e a atribui a uma degeneração referente a um ponto de integração do elemento.

Quando `FemModel` é invocado, o método *init()* de cada elemento inicializa, através do método *initDegenerations()*, as variáveis relativas à integração numérica, que são armazenadas

nas representações de suas degenerações, como pode ser visto nas Figuras 5.20 e 5.21 para o caso de um elemento de barra. Este método atribui, na representação da degeneração relativa a cada ponto de integração, as variáveis relativas às coordenadas naturais e ao fator-peso para quadratura de Gauss. Atribui ainda à representação o mesmo modelo de análise de elemento do qual faz parte. Cada elemento solicita, então, que suas degenerações inicializem suas variáveis.

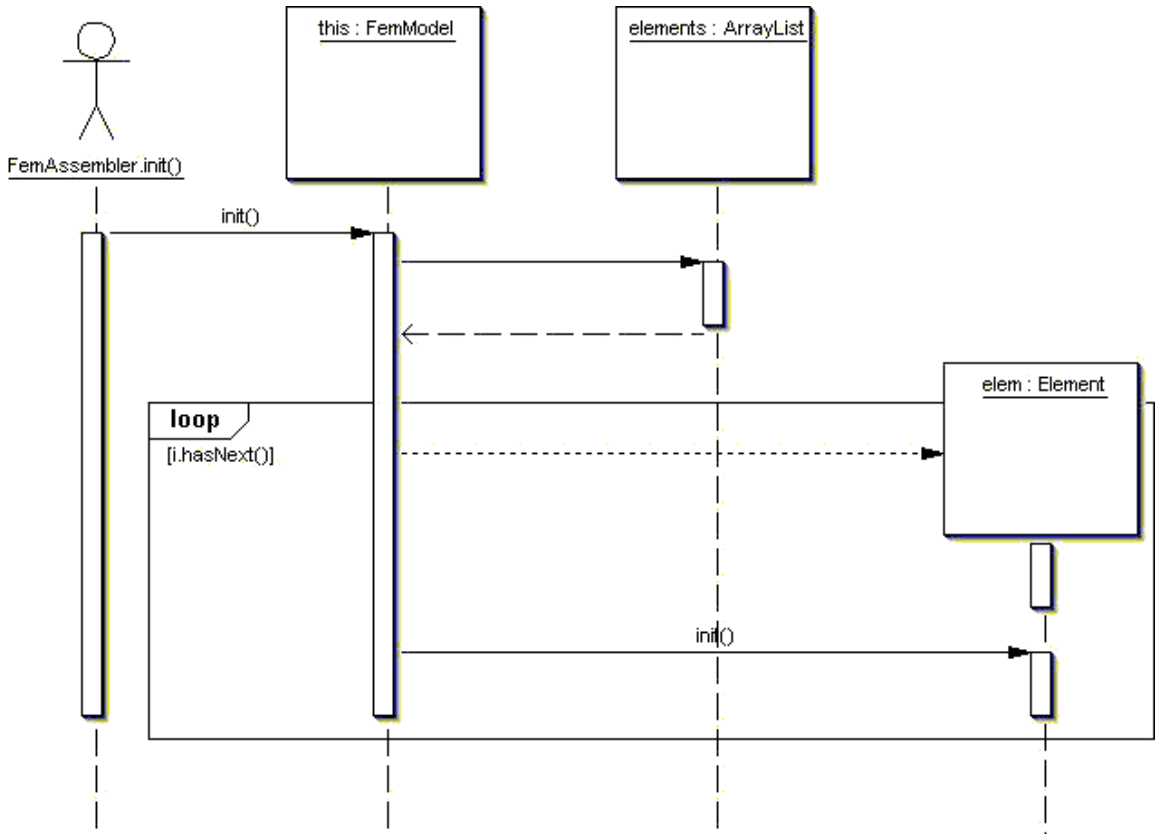


Figura 5.19: Diagrama de seqüência para *FemModel.init()*.

A Figura 5.22 mostra o diagrama de seqüência do método *init()* de uma degeneração do tipo `CrossSection`. A partir do modelo de análise de sua representação, cada degeneração inicializa suas variáveis constitutivas e as variáveis constitutivas prévias, armazenadas em objetos `HashMap`. Em seguida, solicita a todos seus pontos materiais que também se inicializem.

Como mostra a Figura 5.23, cada ponto material consulta em seu modelo constitutivo o número de variáveis constitutivas que possui e, a partir dessa informação, cria seus objetos `HashMap` relativos às variáveis constitutivas e as variáveis constitutivas prévias. Estes mapas, ainda com objetos nulos, são passados pelo ponto material como parâmetros no método *init()*

de seu modelo constitutivo, juntamente com seu modelo de análise e material, para que possam ser inicializados. É o objeto modelo constitutivo que, a partir do modelo de análise passado como parâmetro, inicializa os valores dos mapas, como mostra o diagrama da Figura 5.24 para o modelo constitutivo `OnePointConstModel`. Desta forma, todas as variáveis do modelo foram inicializadas e pode-se iniciar o processo de solução.

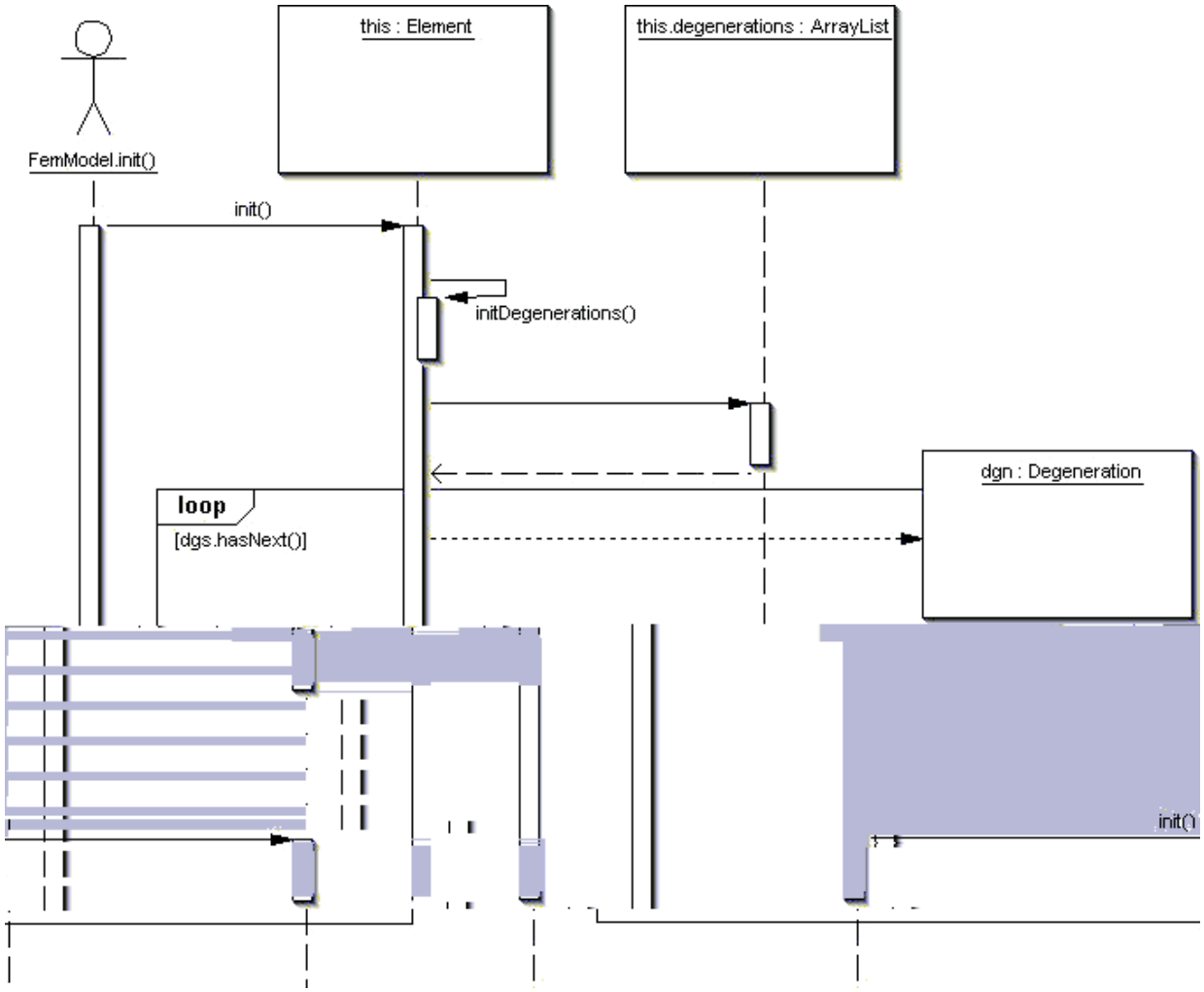


Figura 5.20: Diagrama de seqüência para *Element.init()*.

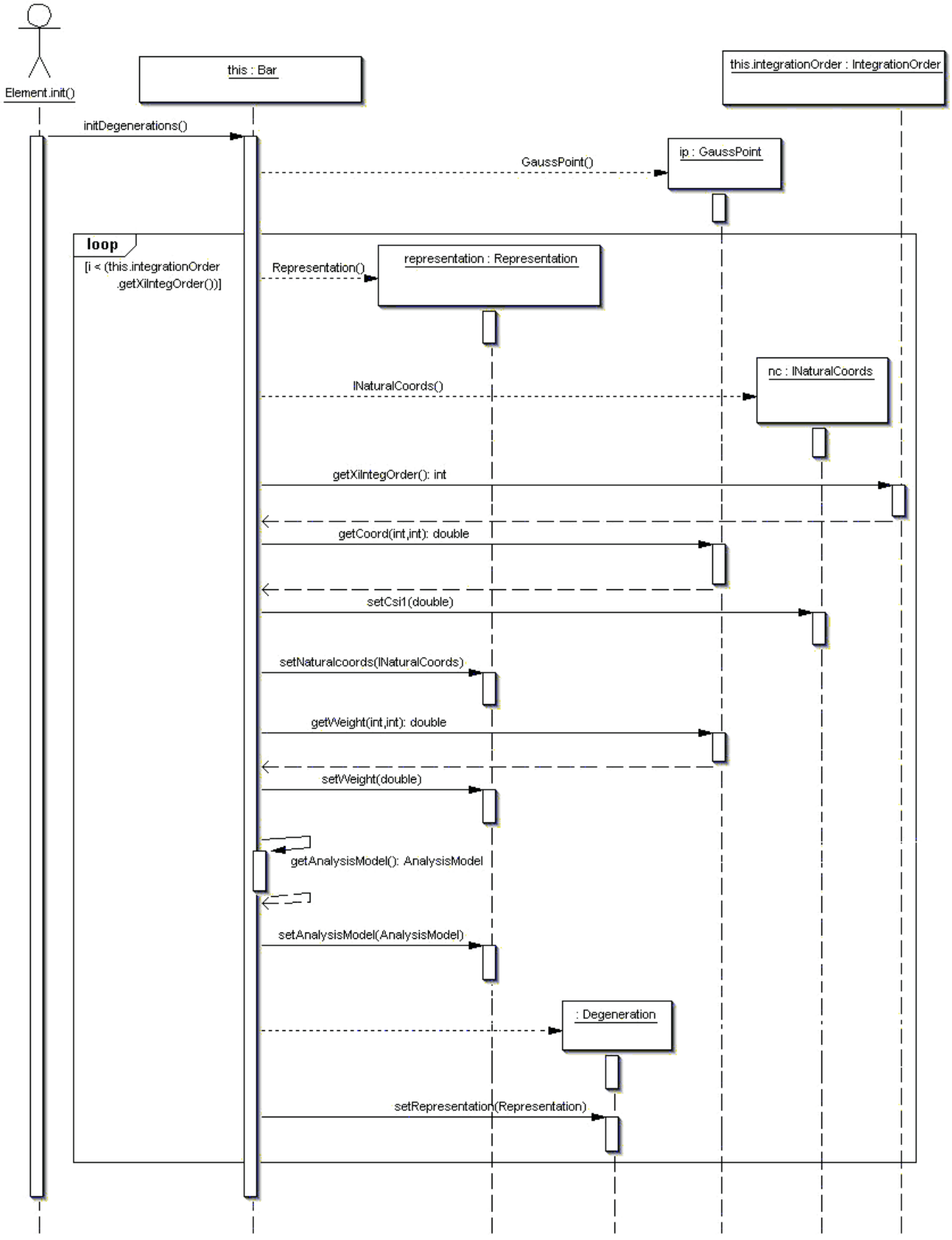


Figura 5.21: Diagrama de seqüência para *Bar.initDegenerations()*.

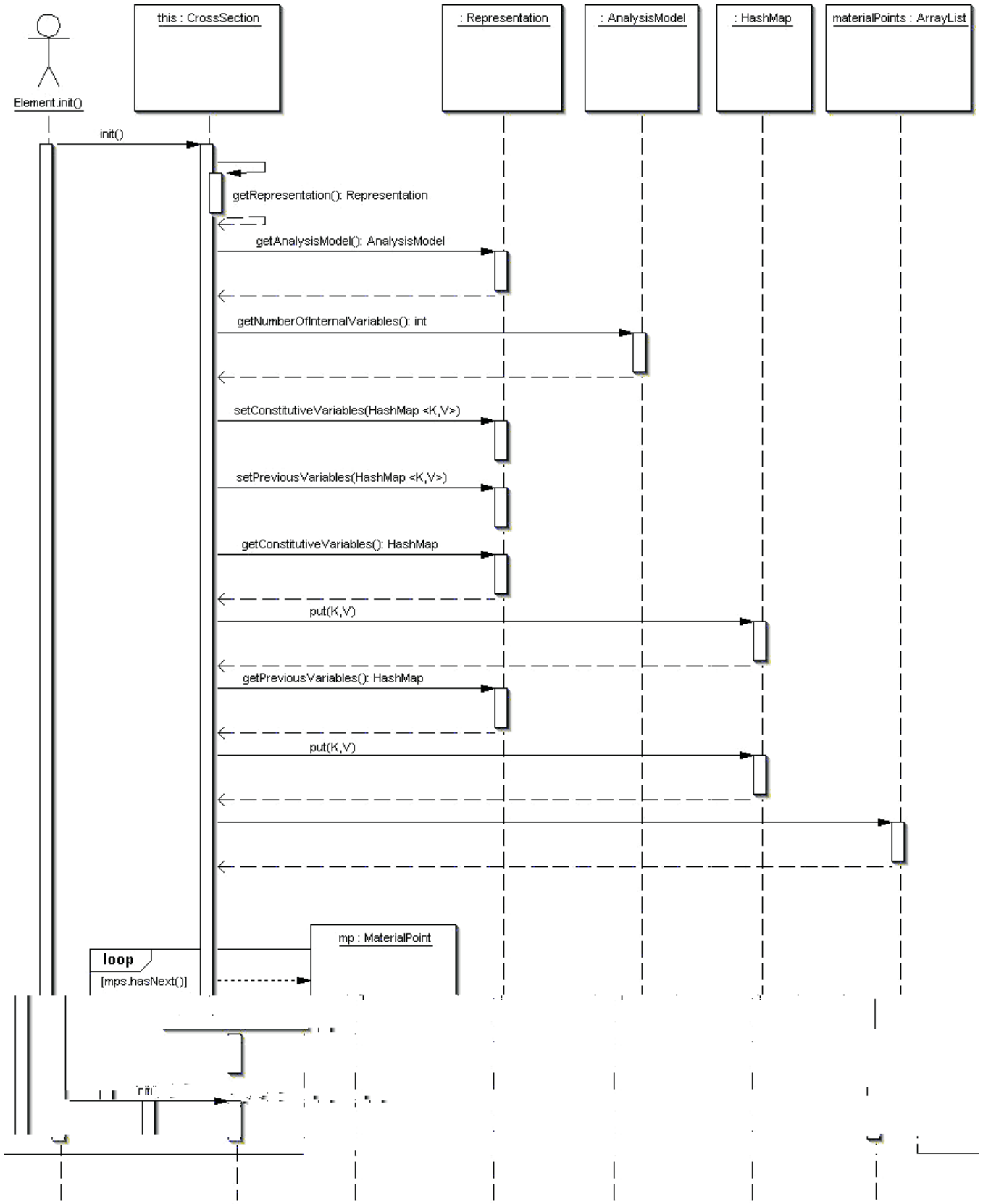


Figura 5.22: Diagrama de seqüência para *CrossSection.init()*.

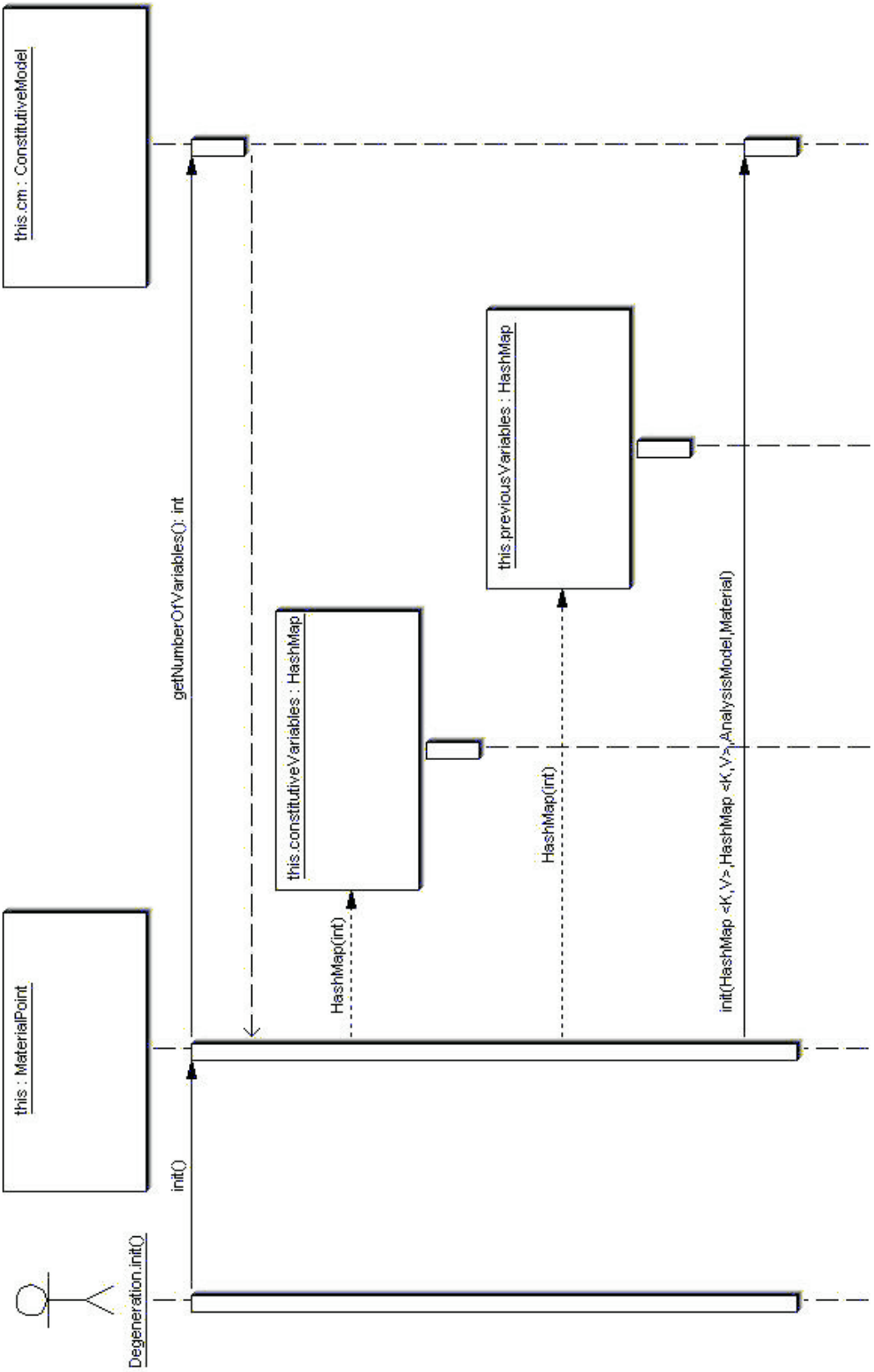


Figura 5.23: Diagrama de seqüência para *MaterialPoint.init()*.

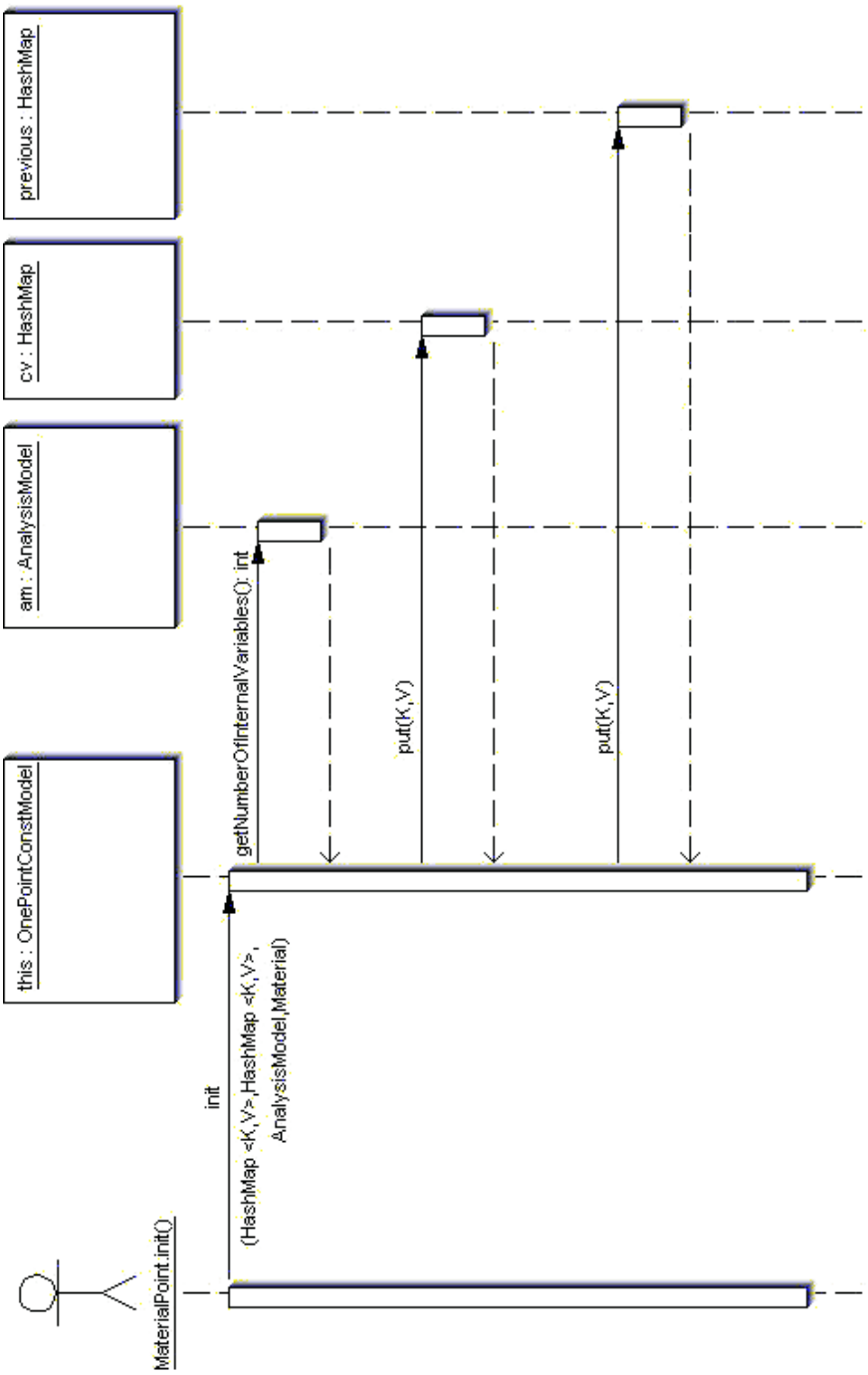


Figura 5.24: Diagrama de seqüência para *OnePointConstMod.init()*.

5.6.2 Método *execute()*

O processo de solução não-linear é disparado ao invocar-se o método *execute()* da classe `EquilibriumPath`, responsável pelo *loop* sobre o número de incrementos, o qual é informado pelo usuário. Este método, além de solicitar ao objeto `StandardNewtonRaphson` do tipo `Step` que inicialize seu `Assembler`, também inscreve `EquilibriumPath` como observador de `StandardNewtonRaphson`.

Dentro deste primeiro *loop* sobre o número de incrementos, representado na Figura 5.25, é chamado o método *execute()* de `StandardNewtonRaphson`, que é responsável pelo *loop* sobre o número de iterações, também informado pelo usuário. Antes de iniciar o *loop* das iterações, `StandardNewtonRaphson`, solicita ao seu `Assembler` o vetor de deslocamentos nodais atuais e também armazena em um vetor interno a carga de referência multiplicada pelo fator de carga, mais as cargas constantes (Figura 5.26).

Uma vez iniciado o loop sobre o número de iterações, representado nas Figuras 5.26 e 5.27, `StandardNewtonRaphson` solicita que `Assembler` calcule, através do método *getIncrementalCuu()*, a matriz de rigidez incremental reduzida do modelo. É então calculado, através do método *getXP()*, o vetor de deslocamentos iterativos devido à carga de referência. Para a primeira iteração de cada passo, é calculado o fator de carga através do método *getPredictor()* do objeto `IterativeStrategy`, que representa o método de controle corrente. Com o fator de carga, o método *assignStepState()* atualiza o fator de carga, conforme a Equação 4.9, calcula os deslocamentos iterativos, conforme a Equação 4.6 e atualiza os deslocamentos nodais, conforme a Equação 4.10.

Feito isto, em todas as iterações, é solicitado a `Assembler` que calcule o vetor de forças nodais equivalentes aos esforços internos e a matriz de rigidez incremental reduzida do modelo para os novos deslocamentos e é calculado, através do método *getUnbalancedX()*, o vetor de deslocamentos iterativos devido à carga desbalanceada. Exceto na primeira iteração de cada passo, nas demais iterações é calculado o fator de carga corrente através do método *getCorrector()* do objeto `IterativeStrategy`. O método *assignStepState()* é novamente invocado para atualizar o fator de carga e os deslocamentos. É verificada então a convergência através

do método *setConvergence()* de `StandardNewtonRaphson`, baseada em força, deslocamento, ou ambos, conforme informado pelo usuário.

Caso a convergência não seja alcançada, `StandardNewtonRaphson` invoca o método *setChanged()*, para sinalizar que seu estado foi alterado, e o método *notifyObservers()*, que notifica a seu observador `EquilibriumPath` que sofreu alterações. Neste momento, `EquilibriumPath`, que também é observado, somente informa a seu observador `Persistence` que a convergência não foi alcançada. Assim, o loop das iterações continua até que se obtenha a convergência ou até que o número máximo de iterações, fornecido pelo usuário, seja atingido.

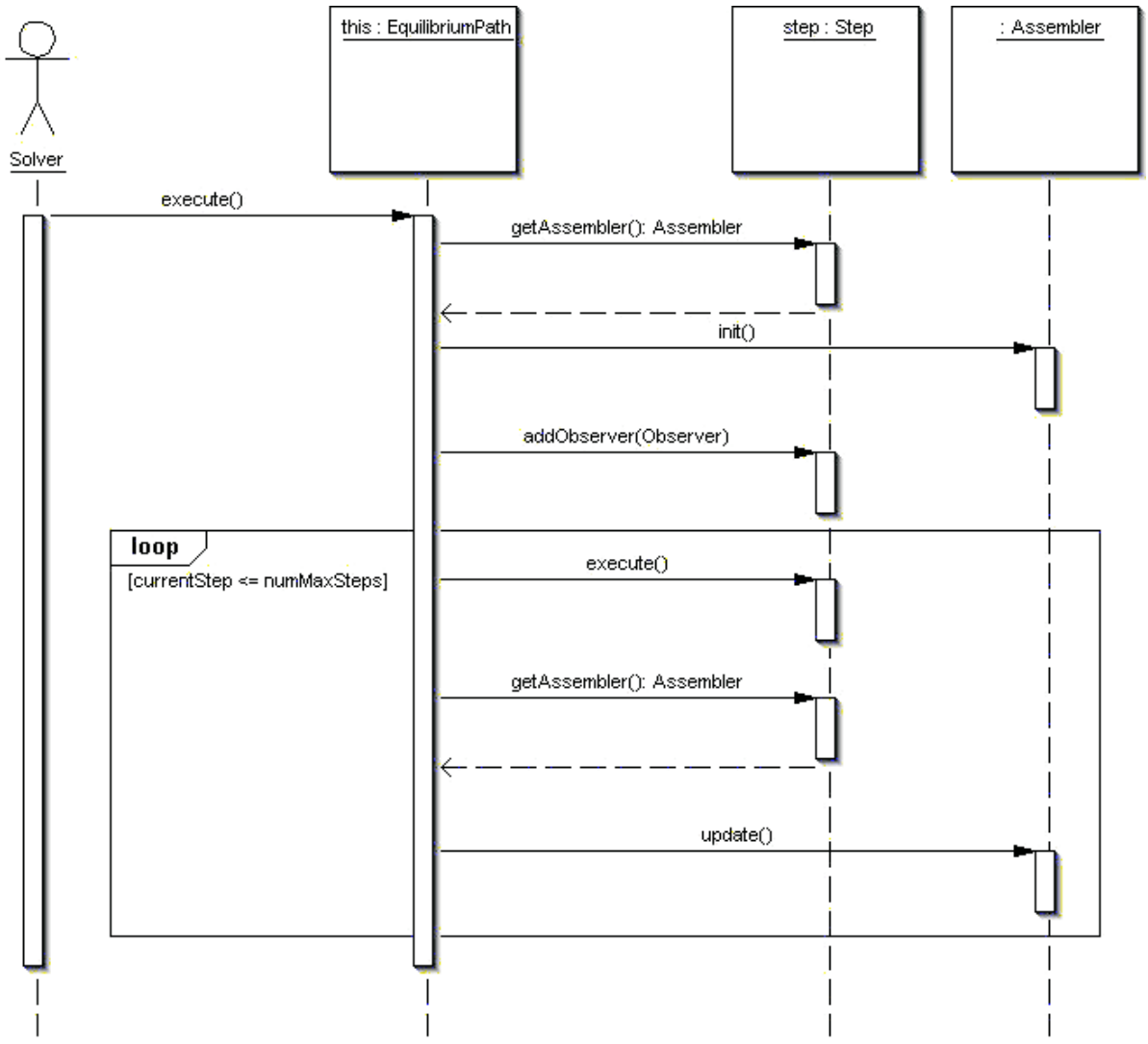


Figura 5.25: Diagrama de seqüência para *EquilibriumPath.execute()*.

Caso a convergência seja alcançada, `StandardNewtonRaphson` realiza o mesmo processo para notificar a seu observador `EquilibriumPath` que sofreu alterações. `EquilibriumPath`, além de informar a seu observador `Persistence` que a convergência foi alcançada, também solicita que `Assembler` atualize as variáveis do modelo, disparando assim o mecanismo de propagação de mudança do núcleo numérico. É neste momento que a persistência, através de seu método *update()*, persiste os dados de saída do modelo para um passo, sendo possível traçar a trajetória de equilíbrio do problema.

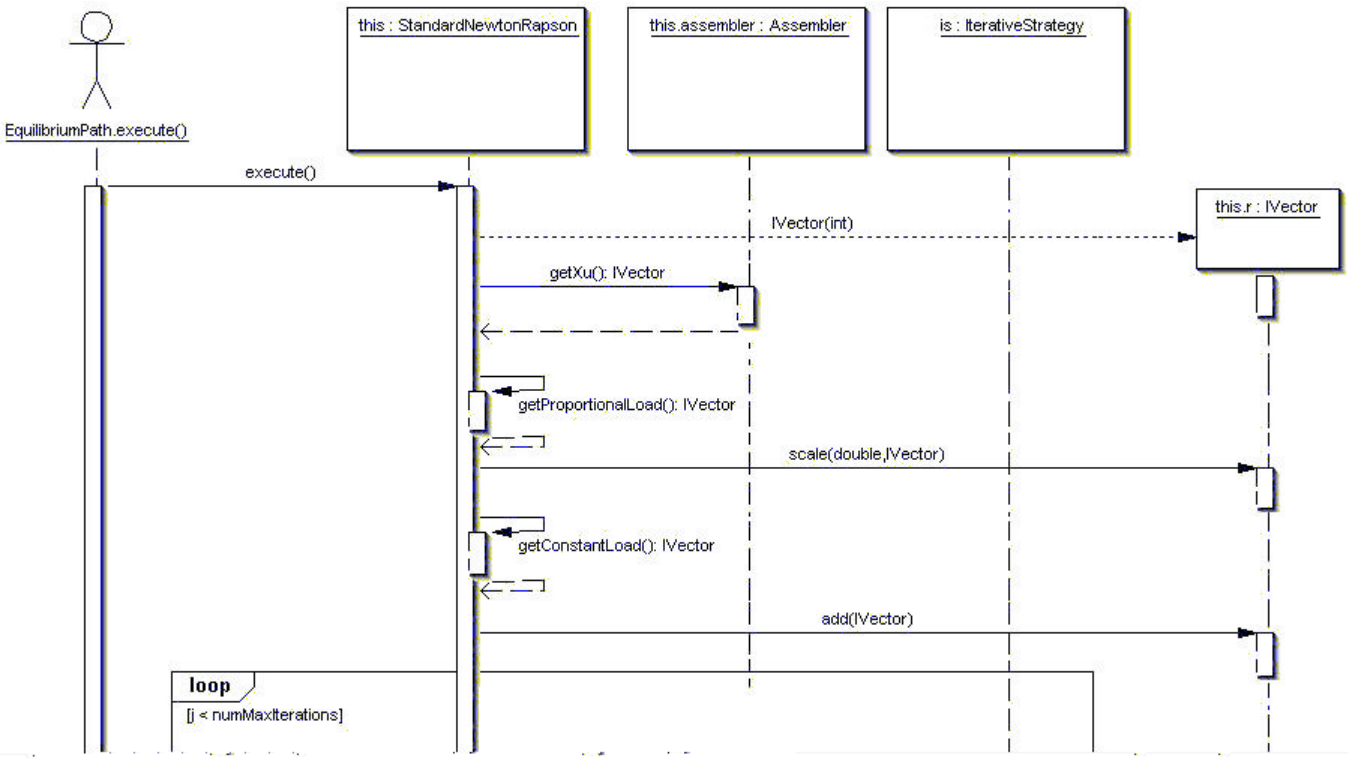


Figura 5.26: Diagrama de seq¨

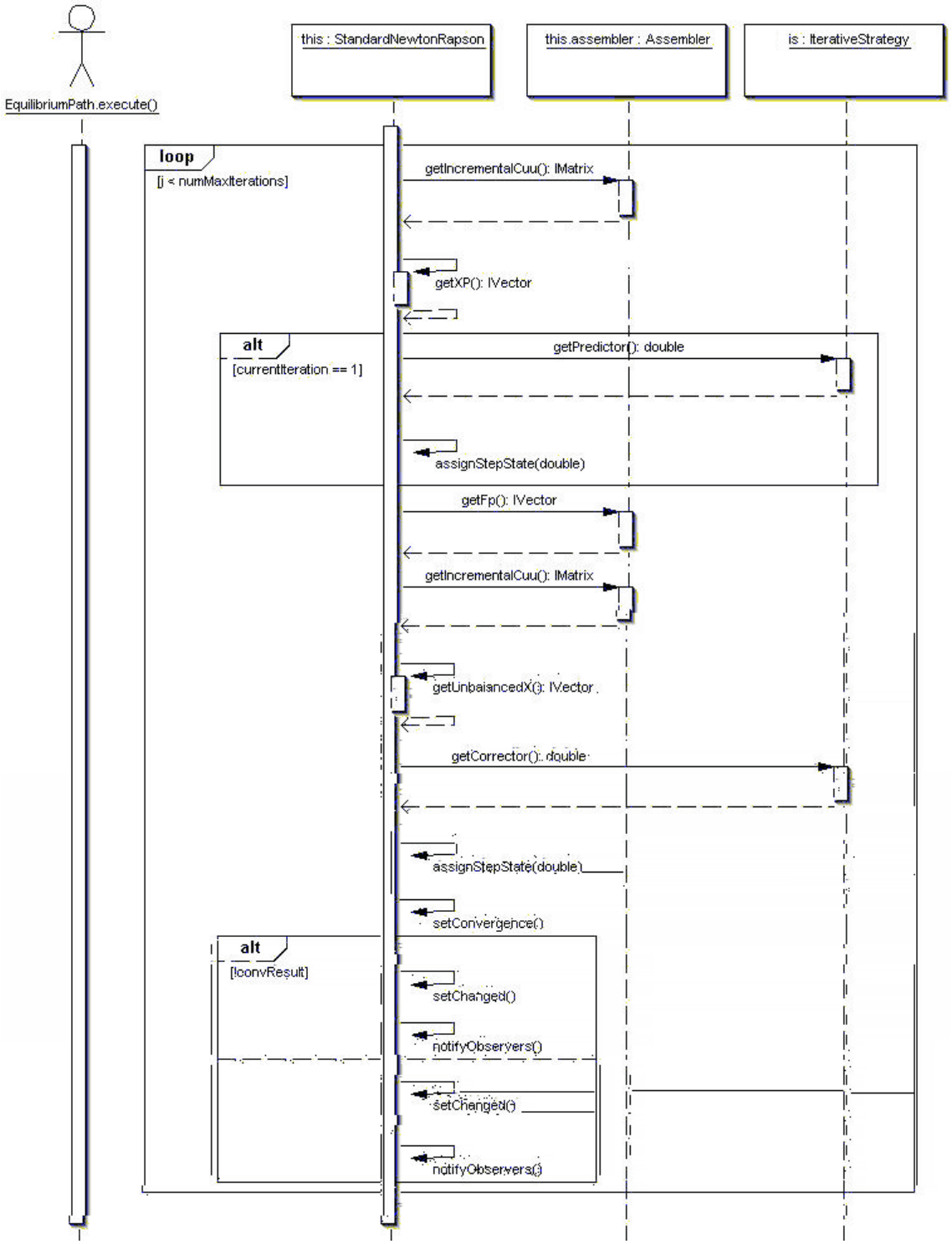


Figura 5.27: Diagrama de seqüência para *StandardNewtonRaphson.execute()* (2^a parte).

5.6.3 Método *getIncrementalC()*

Toda vez que `Assembler` precisa montar a matriz de rigidez incremental, ele a solicita a cada um dos elementos do modelo. Cada elemento, por sua vez, solicita a seu objeto `ProblemDriver` que monte sua matriz, de acordo com o problema que representa.

Nas Figuras 5.28 e 5.29 está representado o diagrama de seqüência resumido do método da classe `ParametricPhysicallyNonLinearSolidMech` para obtenção da matriz de rigidez incremental de um elemento finito paramétrico. Primeiro, é criada uma matriz quadrada de tamanho igual ao número de graus de liberdade do elemento, na qual serão somadas as parcelas da integração numérica. Em seguida, o elemento paramétrico acessa sua lista de degenerações e solicita a cada uma as informações que necessita para o somatório.

Começa requisitando à representação da degeneração as suas coordenadas naturais e, posteriormente, o fator-peso para integração numérica. De posse das coordenadas naturais, solicita ao objeto `Shape` as funções de forma e suas derivadas primeiras e segundas. Com estas, pede ao objeto `AnalysisModel` que monte a matriz das derivadas das funções de forma. É solicitado então à degeneração que monte sua matriz constitutiva tangente. No caso de uma degeneração `CrossSection`, esta matriz é obtida pelo somatório da matriz constitutiva de cada ponto material da lista de pontos materiais que descreve a seção transversal.

Solicita-se ainda ao modelo de análise o fator de integração, o operador jacobiano, e a matriz de transformação. Feitas as devidas multiplicações matriciais e as multiplicações da integração numérica, a matriz obtida é adicionada à matriz criada no início do processo. Percorridas todas as degenerações do elemento paramétrico, obtém-se assim a sua matriz de rigidez incremental.

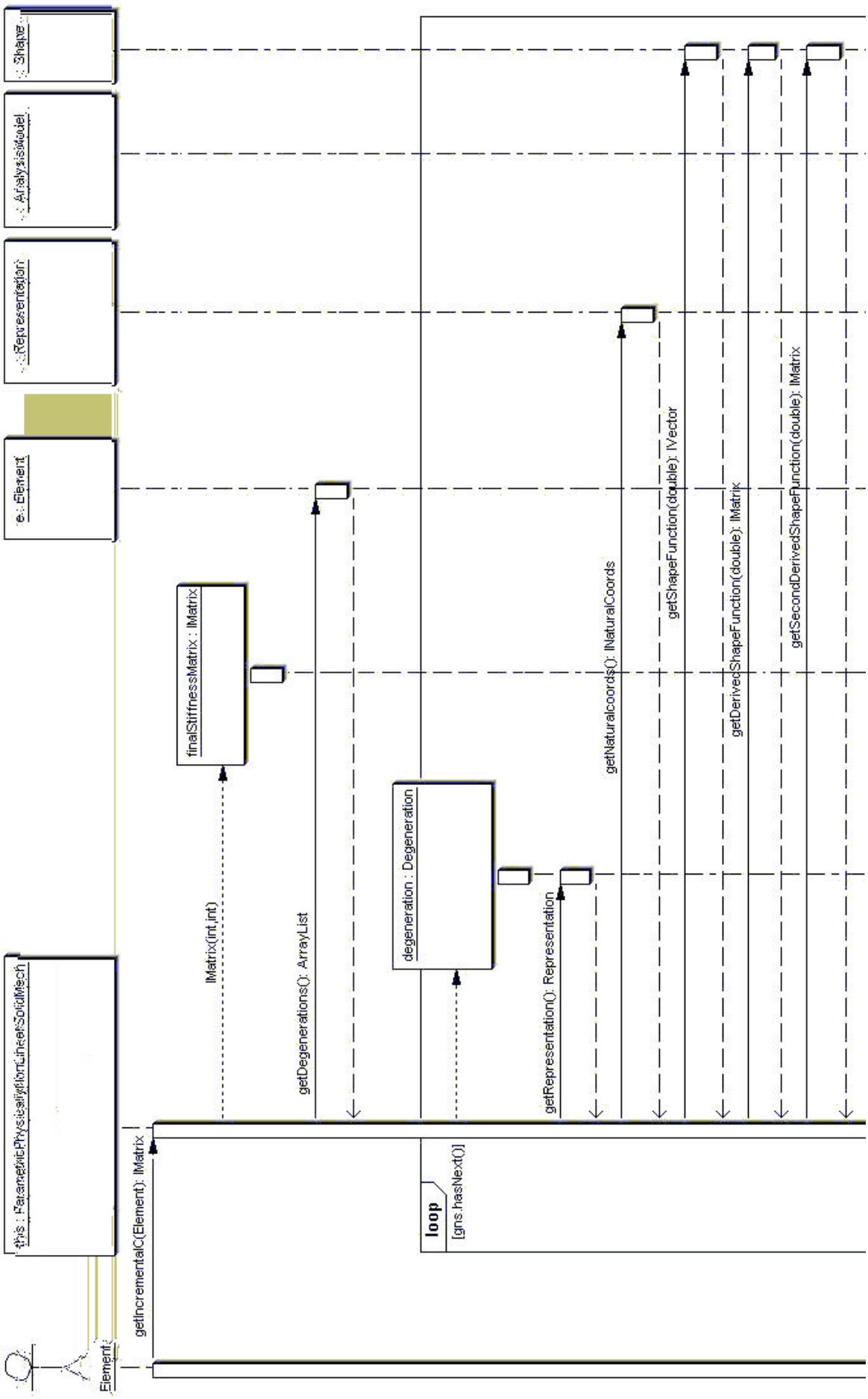


Figura 5.28: Diagrama de seqüência para *ParametricPhysicallyNonLinearSolidMech.getIncrementalC()* (1^a parte).

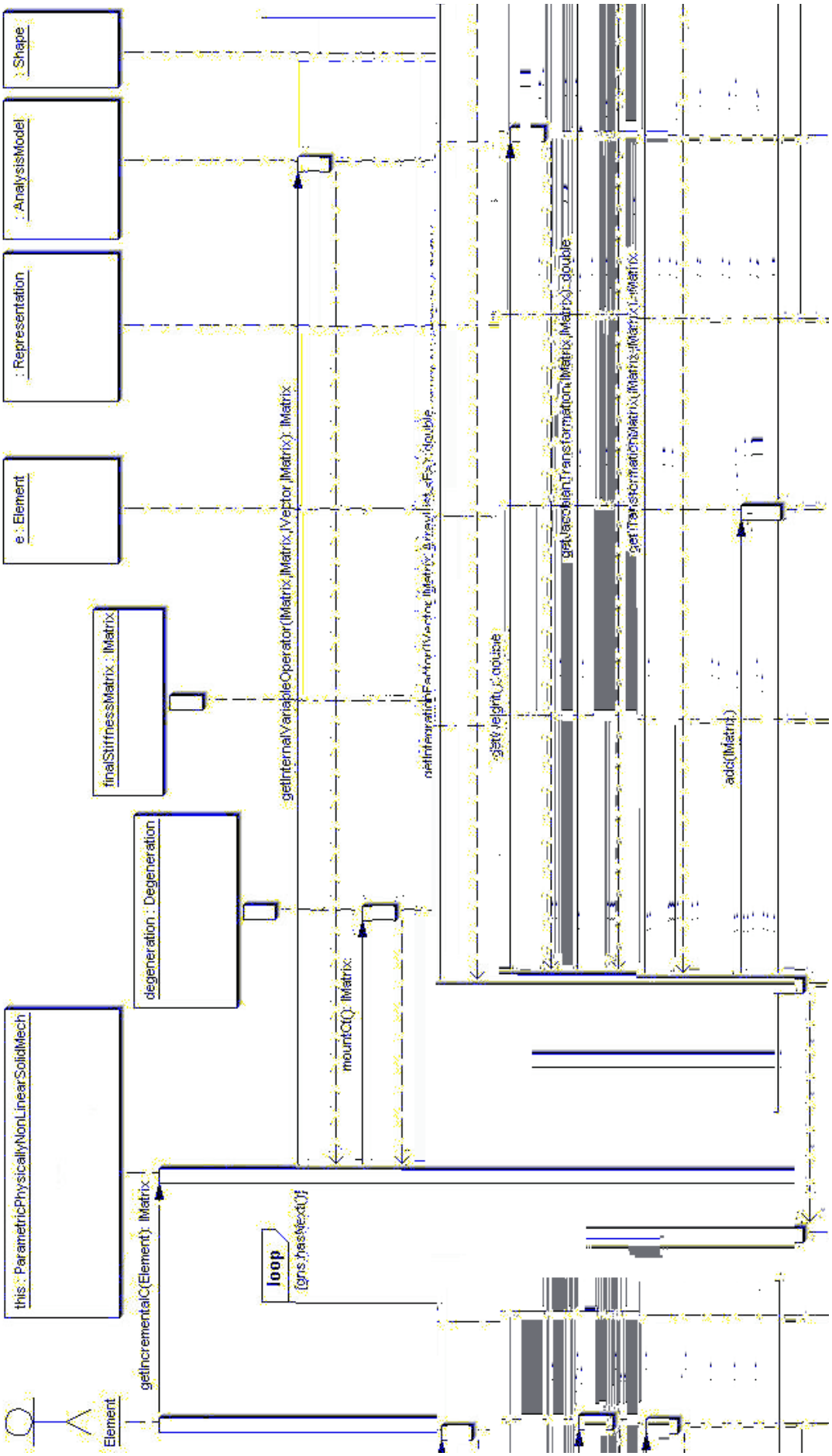


Figura 5.29: Diagrama de seqüência para *ParametricPhysicallyNonLinearSolidMech.getIncrementalC()* (2^a parte).

5.6.4 Método *getFp()*

A seqüência de atividades para obtenção do vetor de forças nodais equivalentes aos esforços internos de um elemento finito paramétrico é semelhante à seqüência para obtenção da matriz de rigidez incremental. Toda vez que `Assembler` precisa montar o vetor de forças nodais equivalentes aos esforços internos do modelo, ele o solicita a cada um dos elementos do modelo. Cada elemento, por sua vez, solicita a seu objeto `ProblemDriver` que monte seu vetor, de acordo com o problema que representa.

Nas Figuras 5.30 e 5.31 está representado o diagrama de seqüência resumido do método da classe `ParametricPhysicallyNonLinearSolidMech` para obtenção do vetor de forças nodais equivalentes aos esforços internos de um elemento finito paramétrico. Primeiro, é criado um vetor de tamanho igual ao número de graus de liberdade do elemento, no qual serão somadas as parcelas da integração numérica. Em seguida, o elemento paramétrico acessa sua lista de degenerações e solicita a cada uma as informações que necessita para o somatório.

Começa requisitando à representação da degeneração as suas coordenadas naturais e, posteriormente, o fator-peso para integração numérica. De posse das coordenadas naturais, solicita ao objeto `Shape` as funções de forma e suas derivadas primeiras e segundas. Com estas, pede ao objeto `AnalysisModel` que monte a matriz das derivadas das funções de forma. É solicitado então ao elemento que monte para aquela degeneração o vetor de esforços internos. Para isso, o elemento calcula o vetor de deformações generalizadas naquela degeneração e multiplica pela sua matriz constitutiva secante. Neste momento, o vetor de esforços internos anterior é armazenado como variável prévia na representação da degeneração.

Solicita-se ainda ao modelo de análise o fator de integração, o operador jacobiano, e a matriz de transformação. Feitas as devidas multiplicações entre matrizes e vetores e as multiplicações da integração numérica, o vetor obtido é adicionado ao vetor criado no início do processo. Percorridas todas as degenerações do elemento paramétrico, obtém-se assim o vetor de forças nodais equivalentes aos esforços internos.

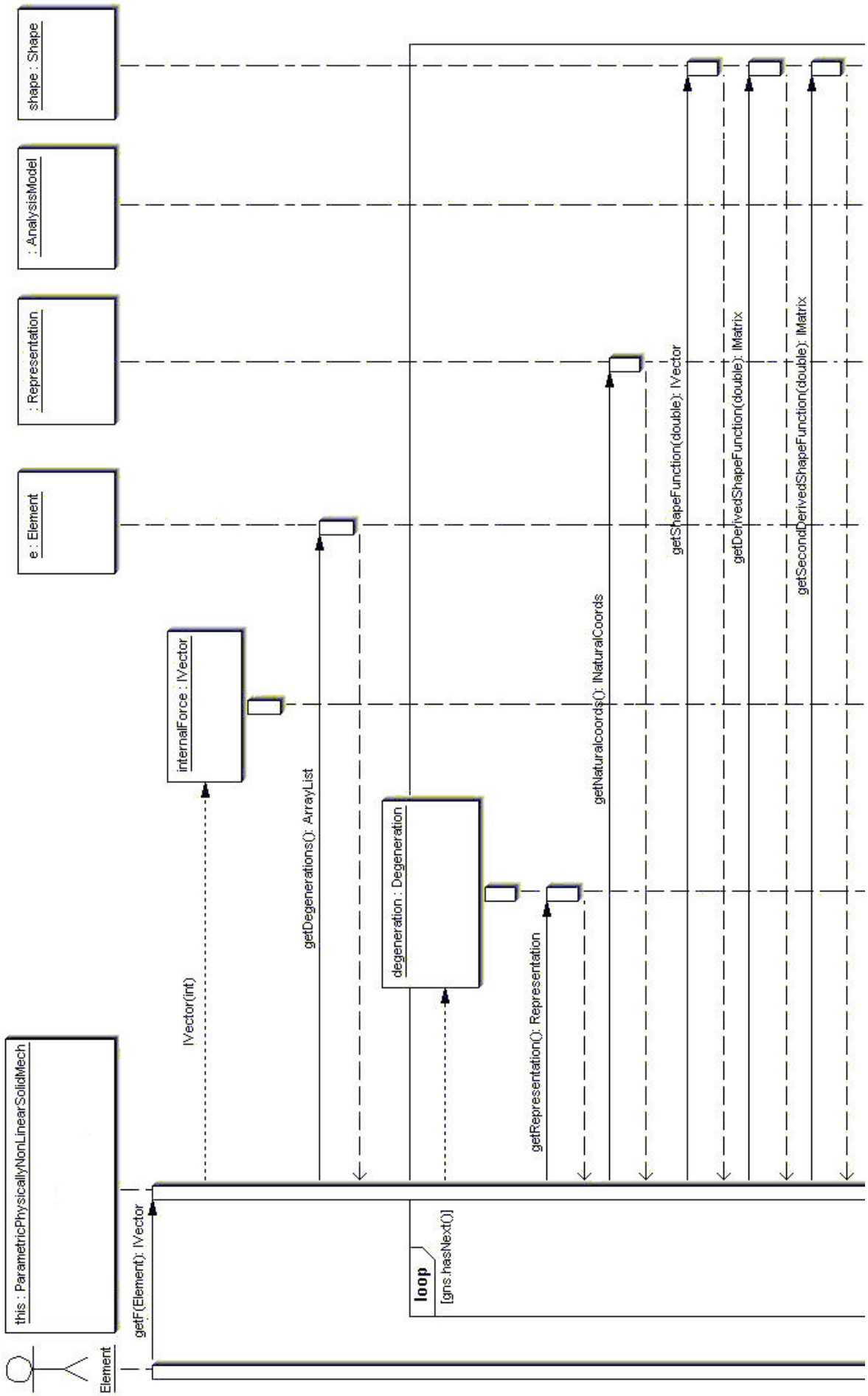


Figura 5.30: Diagrama de seqüência para *ParametricPhysicallyNonLinearSolidMech.getF()* (1^a parte).

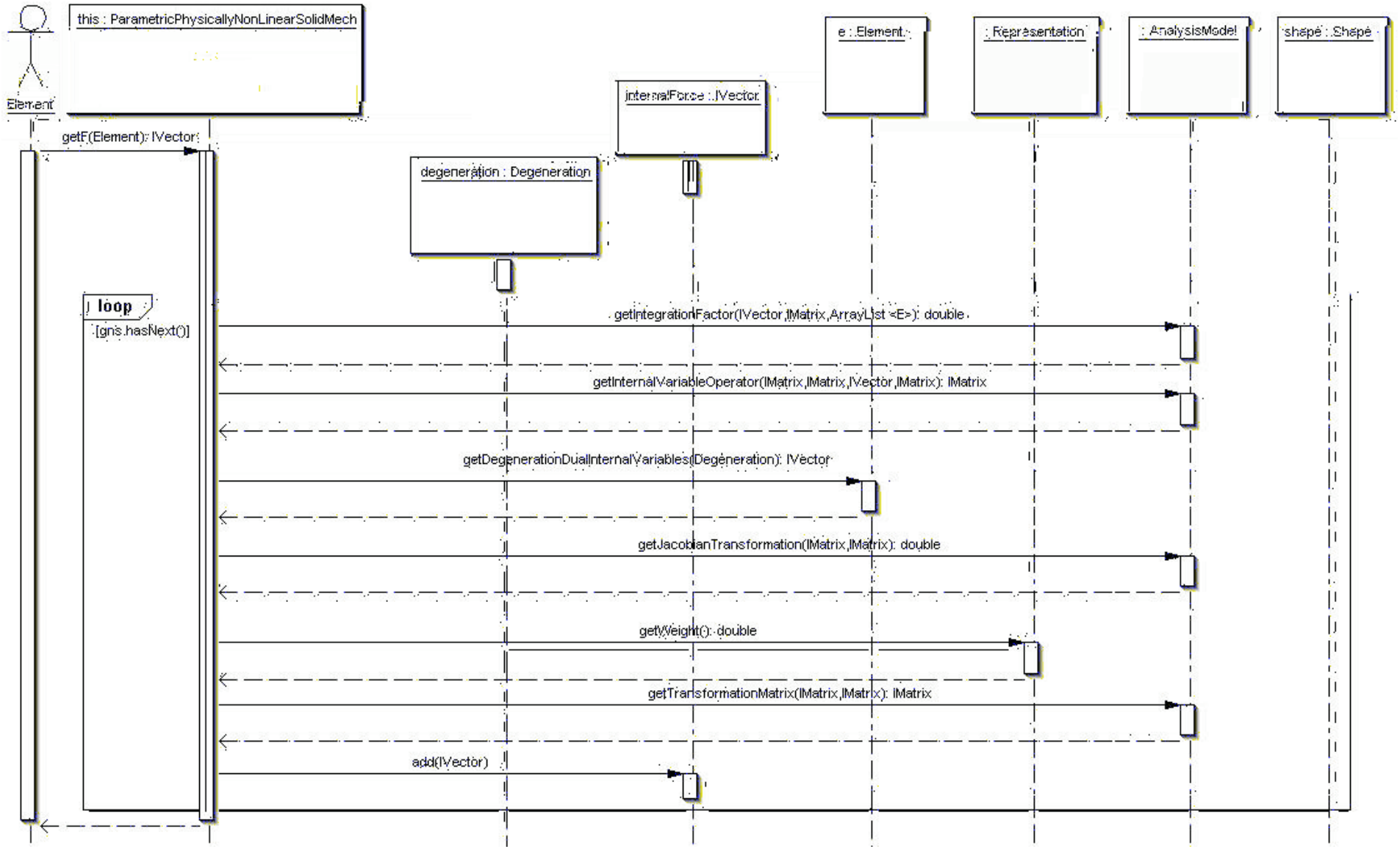


Figura 5.31: Diagrama de seqüência para *ParametricPhysicallyNonLinearSolidMech.getF()* (2^a parte).

5.6.5 Método *update()*

Terminada a solução de um passo do processo incremental-iterativo, `Solution` dispara o processo de atualização das variáveis do modelo de elementos finitos, solicitando a `Assembler` que se atualize. Este solicita, então, a `Model` que atualize suas variáveis, como mostra o diagrama da Figura 5.32.

Como `Model` é um objeto que estende a classe `Observable`, ele invoca os métodos *setChanged()*, que através de um `boolean` sinaliza que foi alterado, e *notifyObservers()*, que notifica seus observadores que sofreu alterações. É neste momento, indicado na Figura 5.33, que os componentes observadores, como a persistência, têm seu método *update()* invocado. `Model` prossegue percorrendo sua lista de elementos e atualizando-os.

Cada `ParametricElement` solicita que cada uma das degenerações de sua lista atualize as suas variáveis, como indicado na Figura 5.34. No caso de uma degeneração `CrossSection`, é solicitado à representação que armazene suas variáveis constitutivas atuais em seu mapa de variáveis prévias, como mostrado na Figura 5.35. Em seguida, é solicitado a todos pontos materiais que também atualizem suas variáveis.

Como representado nos diagramas das Figuras 5.36 e 5.37, o ponto material invoca o método *update()* de seu modelo constitutivo, passando como parâmetro seus dois mapas. O modelo constitutivo `OnePointConstModel` armazena os valores do mapa de variáveis constitutivas do ponto material no mapa de variáveis prévias. Desta forma, as variáveis constitutivas correntes de todo o modelo foram atualizadas, sendo armazenadas como variáveis prévias, processo esse necessário à obtenção da convergência no próximo passo do processo incremental-iterativo.

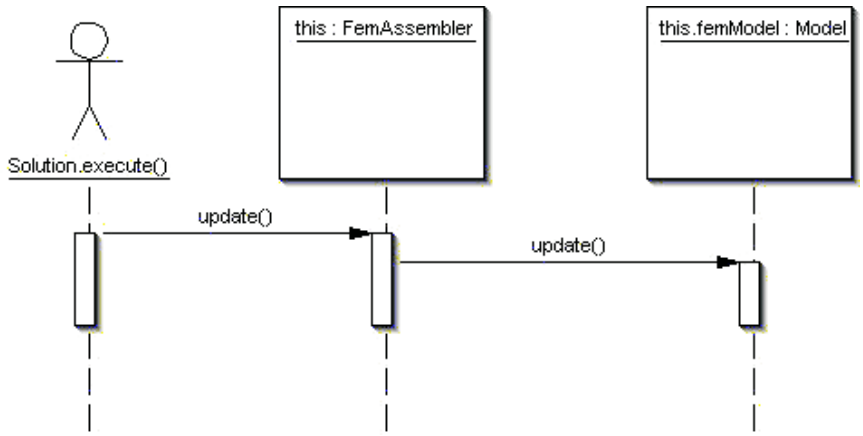


Figura 5.32: Diagrama de seqüência para *FemAssembler.update()*.

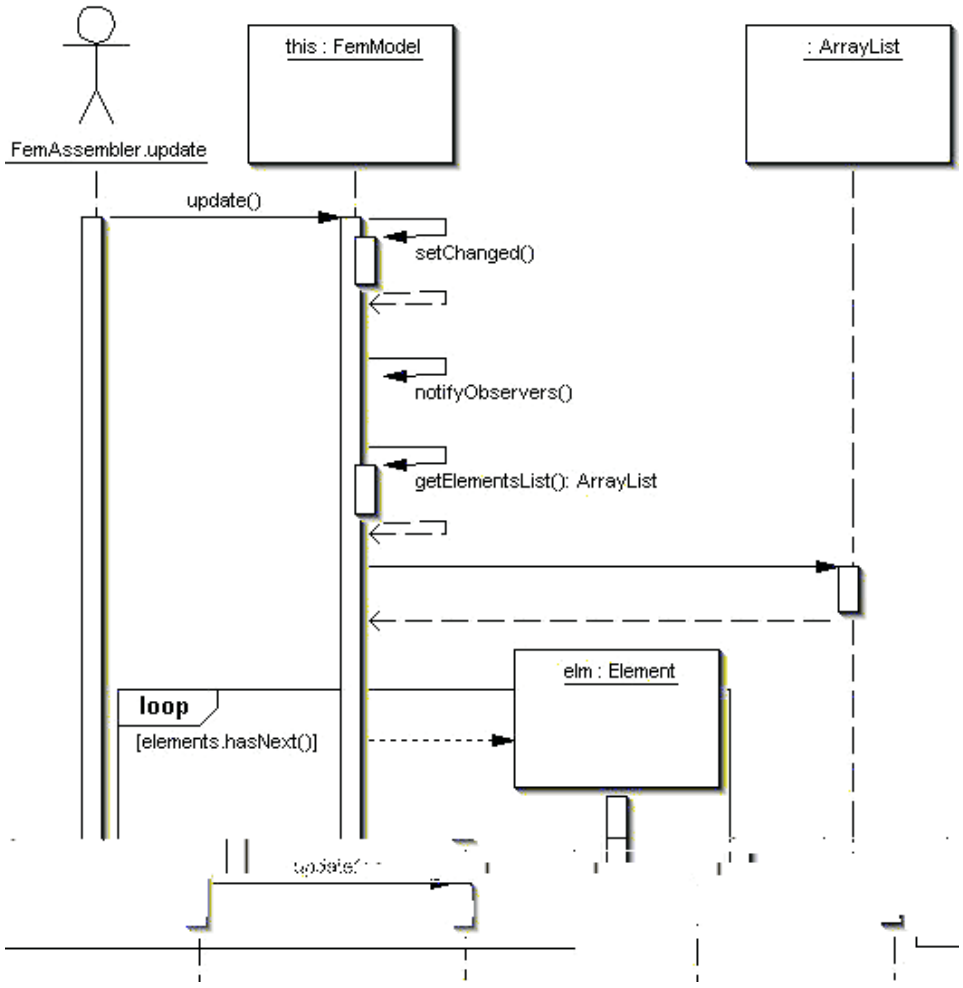


Figura 5.33: Diagrama de seqüência para *FemModel.update()*.

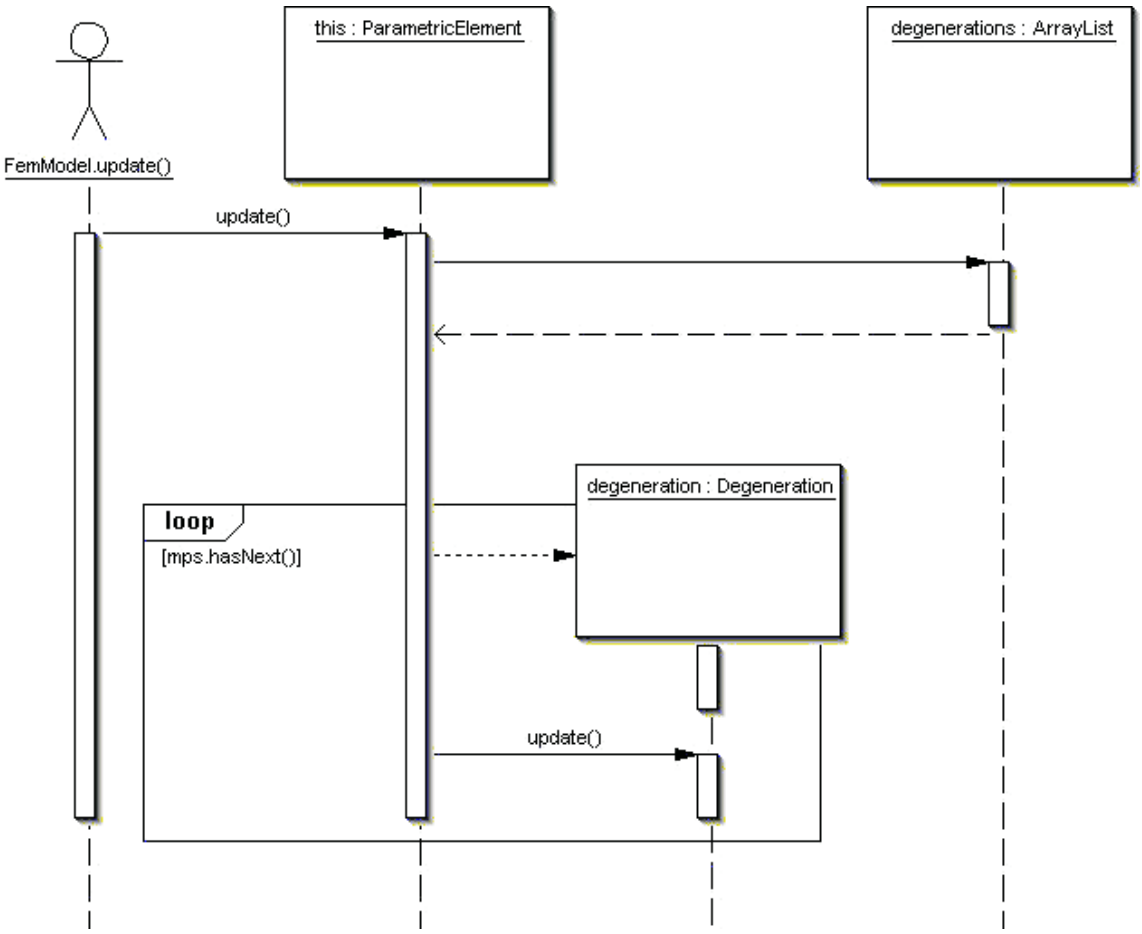


Figura 5.34: Diagrama de seqüência para *ParametricElement.update()*.

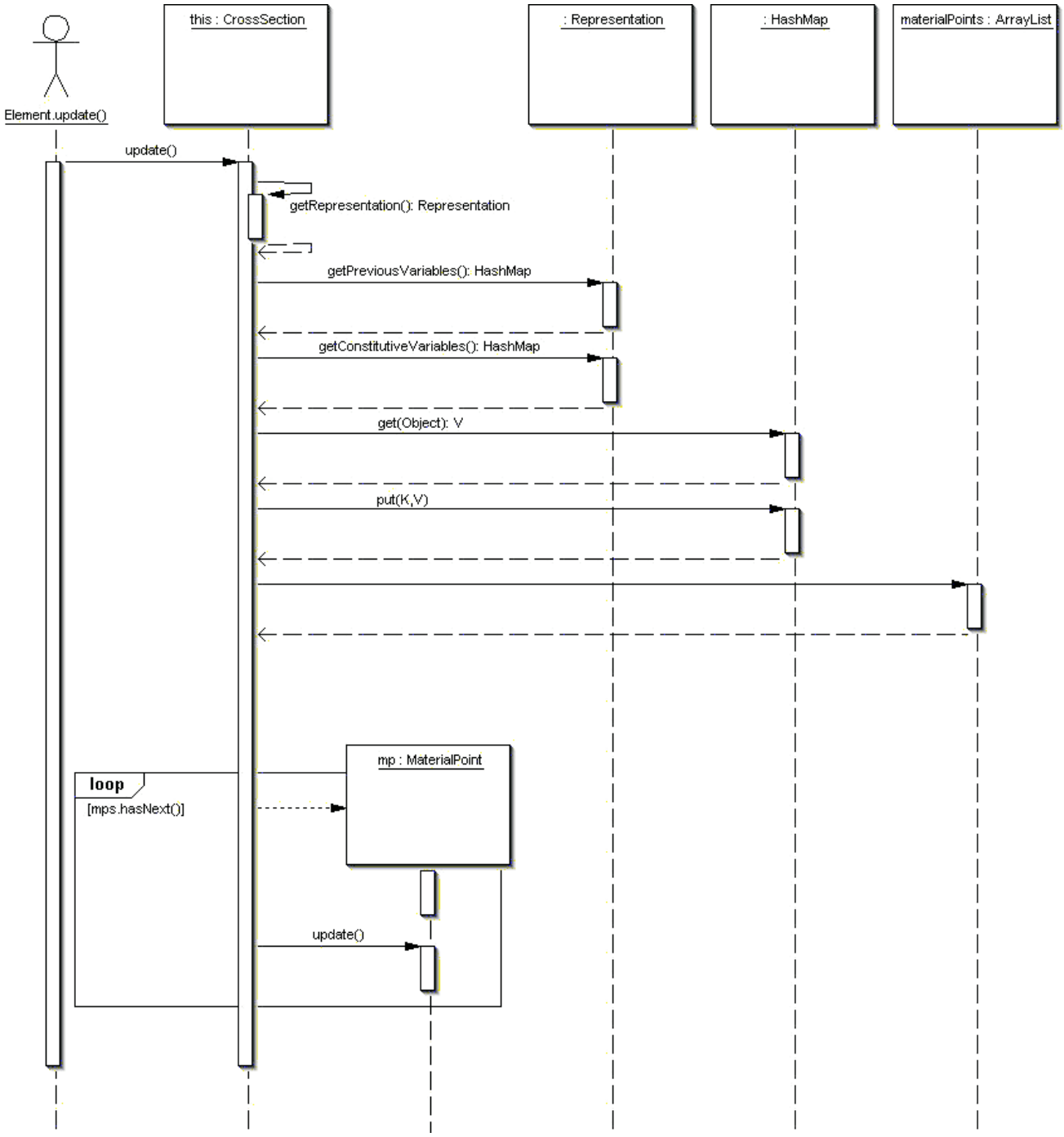


Figura 5.35: Diagrama de seqüência para *CrossSection.update()*.

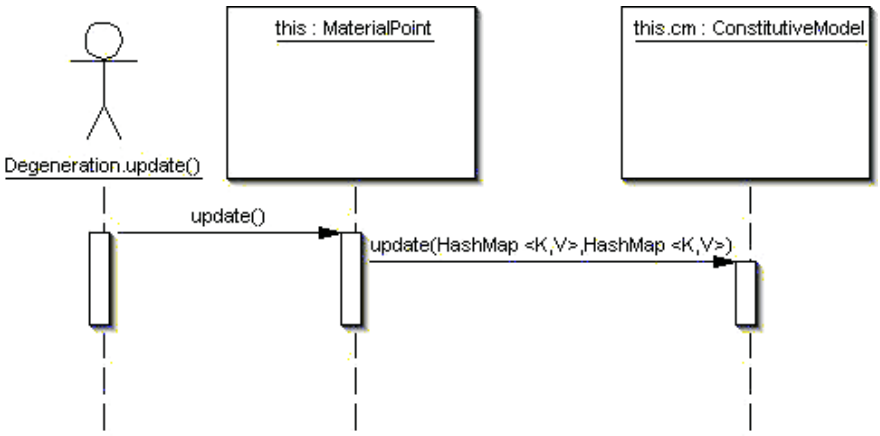


Figura 5.36: Diagrama de seqüência para *MaterialPoint.update()*.

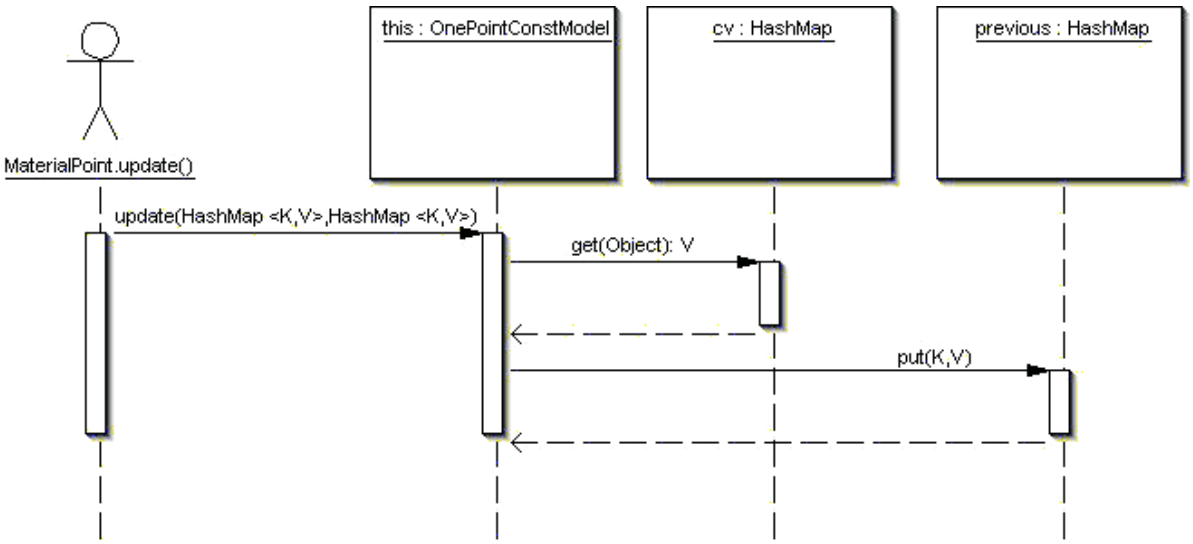


Figura 5.37: Diagrama de seqüência para *OnePointConstModel.update()*.

Capítulo 6

SIMULAÇÕES NUMÉRICAS

6.1 Introdução

Com o objetivo de ilustrar e validar a implementação da formulação proposta, são apresentadas neste capítulo simulações numéricas de problemas elásticos lineares e de problemas fisicamente não-lineares, nos quais empregam-se os recursos disponibilizados no **INSANE**.

Os exemplos elásticos lineares confrontam os resultados obtidos com as soluções analíticas correspondentes, visando validar a formulação proposta e o modelo de elementos finitos implementado. São discutidas as limitações da formulação e as limitações e aplicações dos elementos finitos disponíveis no sistema.

Nas simulações numéricas de problemas fisicamente não-lineares, comparam-se os resultados obtidos com resultados experimentais da literatura. Outras simulações são apresentadas para ilustrar os recursos disponíveis. Pretende-se verificar a eficiência da implementação da formulação proposta na descrição das trajetórias de equilíbrio dos modelos. Todos estas simulações tratam de estruturas de concreto armado, uma vez que são estruturas de seções compostas cujos exemplos são abundantes na literatura. As leis constitutivas consideradas para os materiais estão discutidas no Capítulo 4.

A persistência de dados dos exemplos realizou-se através de arquivos no formato XML. No Apêndice B são mostrados, como exemplo, um arquivo de entrada de dados e seu respectivo arquivo de saída de dados para a simulação da Seção 6.3.1.

6.2 Problemas Elásticos Lineares

6.2.1 Estruturas Reticuladas

Em um primeiro momento na validação da implementação, foram analisados exemplos de estruturas reticuladas cujos materiais são elásticos lineares. Os exemplos foram retirados da literatura e estão representados nas Figuras de 6.1 a 6.6: viga contínua, treliça plana, pórtico plano, treliça espacial (Weaver, Jr. e Gere, 1980), pórtico espacial (Dawe, 1984) e grelha (Vasconcellos Filho, 1986). As figuras mostram também, para cada exemplo, as propriedades dos materiais e a discretização das seções transversais das barras em $\boldsymbol{q}$ áreas. Em Weaver, Jr. e Gere (1980), as propriedades geométricas das seções são dadas numericamente e foi necessário descrever uma seção cujas propriedades fossem iguais às do problema descrito.

Os exemplos foram analisados segundo a teoria de Euler-Bernoulli, utilizando-se um elemento finito de Hermite de 2 nós por barra, com 2 pontos de integração cada. Na análise, utilizou-se primeiro o recurso de análise linear do **INSANE** e em seguida o recurso de análise não-linear com um único passo e uma única iteração. Os resultados obtidos pelos dois recursos foram idênticos, mostrando coerência entre os mesmos.

A Tabela 6.1 mostra, para determinados deslocamentos, o erro percentual dos resultados obtidos pelo **INSANE** em relação aos resultados da literatura. Mostra também os valores das propriedades das seções transversais dados nas referências em comparação às propriedades geométricas calculadas com as discretizações das seções adotadas.

A discretização da seção resulta em um erro percentual nos valores dos momentos de inércia e momentos polares em relação ao valor exato. Essa diferença faz com que os resultados obtidos para os deslocamentos pelo **INSANE** tenham também um erro em relação aos valores calculados analiticamente nas referências.

Era de se esperar que, com a discretização em elementos finitos utilizada, os resultados obtidos fossem bastante próximos aos exatos. Porém, devido à discretização das seções transversais e a conseqüente aproximação de suas propriedades geométricas, que influem diretamente na rigidez da estrutura, os resultados refletiram o erro das propriedades geométricas

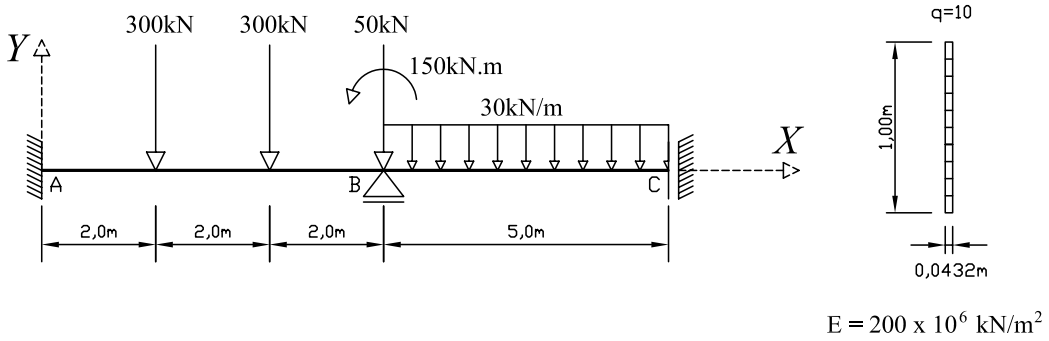


Figura 6.1: Viga Contínua.

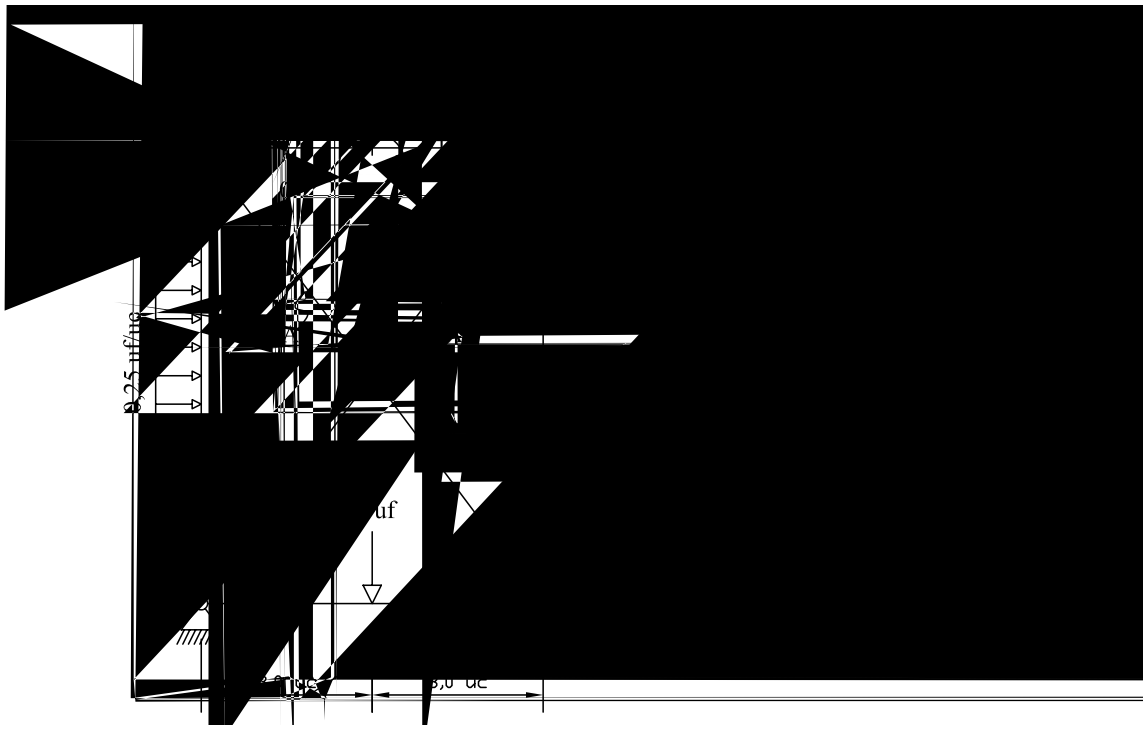


Figura 6.2: Treliça Plana (em unidades de comprimento *uc* e unidades de força *uf*).

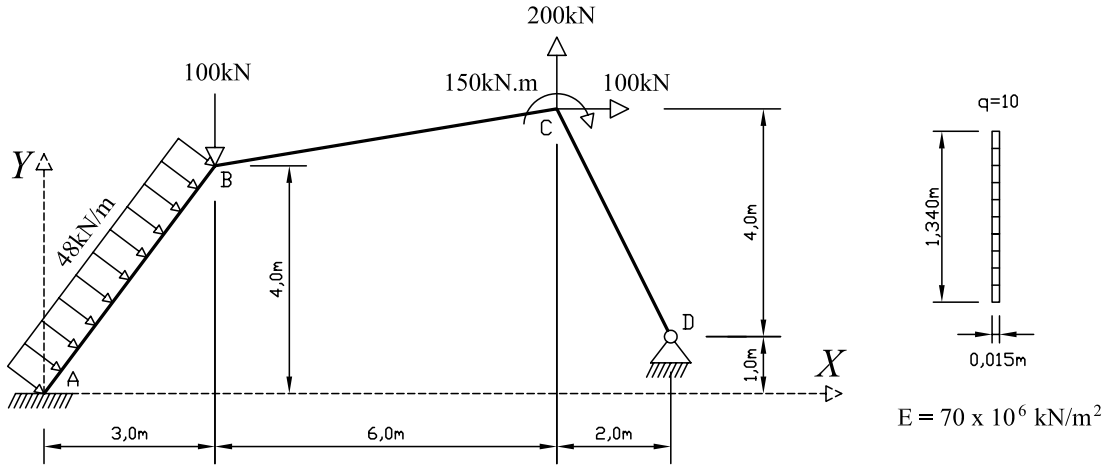


Figura 6.3: Pórtico Plano

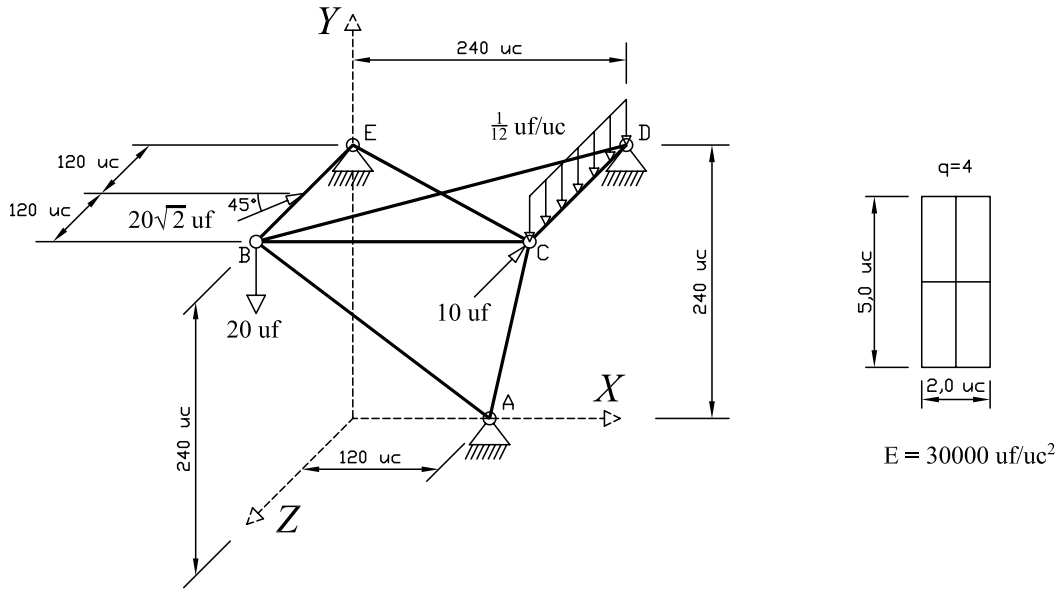


Figura 6.4: Treliça Espacial (em unidades de comprimento *uc* e unidades de força *uf*).

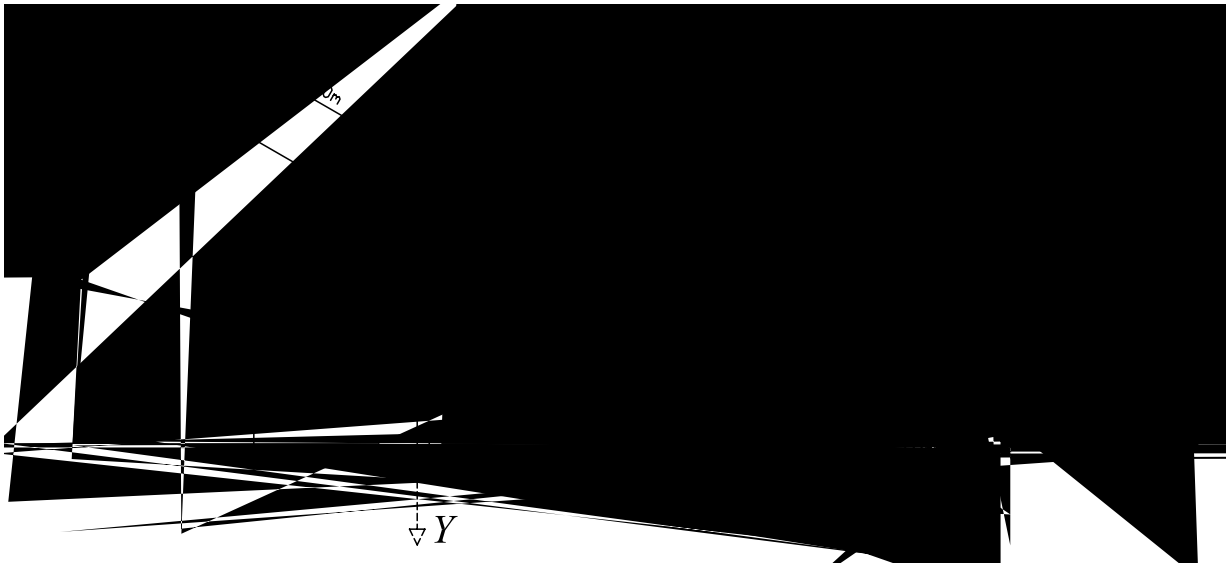


Figura 6.5: Pórtico Espacial.

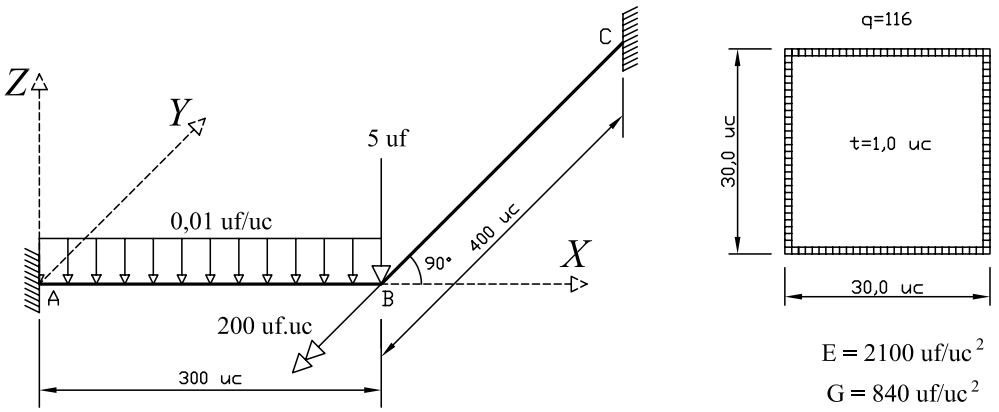


Figura 6.6: Grelha (em unidades de comprimento *uc* e unidades de força *uf*).

nos valores dos deslocamentos obtidos, como pode ser visto na Tabela 6.1. Este erro poderia ser facilmente reduzido com o refinamento da discretização das seções transversais, o que melhoraria a aproximação para suas propriedades geométricas e, conseqüentemente, a aproximação para os deslocamentos da estrutura.

Tabela 6.1: Comparação dos resultados para exemplos elásticos lineares.

Nó	*Desloc.*	*Resultado Referência*	*Resultado Obtido*	*Erro Relativo (%)*	*Propriedades Seção Transv.*	*Valor dado na Referência*	*Valor calculado INSANE*	*Erro Relativo (%)*
VIGA CONTÍNUA								
C	**dy**	*1.169E-04 m*	*1.180E-04 m*	*0.94*	**Inércia Z (m^4)**	*3.600E-03*	*3.564E-03*	*-1.00*
TRELIÇA PLANA								
A	**dx**	*1.000E-01 uc*	*9.986E-02 uc*	*-0.14*	**Área (uc^2)**	*1.000E+01*	*1.000E+01*	*0.00*
	dy	*4.147E-02 uc*	*4.142E-02 uc*	*-0.12*				
B	**dx**	*1.061E-01 uc*	*1.060E-01 uc*	*-0.09*				
	dy	*-4.020E-02 uc*	*-4.015E-02 uc*	*-0.12*				
PÓRTICO PLANO								
B	**dx**	*7.368E-03 m*	*7.461E-03 m*	*1.26*	**Área (m^2)**	*2.000E-02*	*2.000E-02*	*0.00*
	dy	*-5.192E-03 m*	*-5.262E-03 m*	*1.35*	**Inércia Z (m^4)**	*3.000E-03*	*2.963E-03*	*-1.23*
	Θz	*-1.335E-04 rad*	*-1.359E-04 rad*	*1.80*				
C	**dx**	*6.410E-03 m*	*6.485E-03 m*	*1.17*				
	dy	*3.421E-03 m*	*3.458E-03 m*	*1.08*				
	Θz	*-2.685E-04 rad*	*-2.709E-04 rad*	*0.89*				
TRELIÇA ESPACIAL								
B	**dx**	*1.477E-02 uc*	*1.477E-02 uc*	*0.00*	**Área (uc^2)**	*1.000E+01*	*1.000E+01*	*0.00*
	dy	*-5.638E-02 uc*	*-5.637E-02 uc*	*-0.02*				
	dz	*9.769E-03 uc*	*9.768E-03 uc*	*-0.01*				
C	**dx**	*1.654E-02 uc*	*1.654E-02 uc*	*0.00*				
	dy	*-2.504E-02 uc*	*-2.505E-02 uc*	*0.04*				
	dz	*-1.023E-02 uc*	*-1.023E-02 uc*	*0.00*				
PÓRTICO ESPACIAL								
A	**dx**	*1.106E-05 m*	*1.107E-05 m*	*0.09*				
	dy	*7.373E-08 m*	*7.284E-08 m*	*-1.21*	**Área (m^2)**	*2.827E-03*	*2.827E-03*	*0.00*
	dz	*-5.931E-05 m*	*-5.932E-05 m*	*0.02*	**Inércia Y (m^4)**	*2.898E-06*	*2.863E-06*	*-1.22*
	Θx	*2.031E-05 rad*	*2.032E-05 rad*	*0.05*	**Inércia Z (m^4)**	*2.898E-06*	*2.863E-06*	*-1.22*
	Θy	*7.885E-03 rad*	*7.983E-03 rad*	*1.24*	**Inércia Polar (m^4)**	*5.796E-06*	*5.726E-06*	*-1.22*
	Θz	*3.810E-06 rad*	*3.810E-06 rad*	*0.00*				
GRELHA								
B	**dz**	*-6.724E-01 uc*	*-6.719E-01 uc*	*-0.07*	**Inércia Y (uc^4)**	*16259.0*	*16270.0*	*0.07*
	Θx	*1.990E-03*	*1.989E-03*	*-0.05*	**Inércia Polar (uc^4)**	*32518.0*	*32540.0*	*0.07*
	Θy	*2.398E-03*	*2.396E-03*	*-0.08*				

Os resultados obtidos foram, mesmo assim, bastante satisfatórios já que se utilizou uma discretização das seções transversais com um número pequeno de áreas. Confirmou-se a adequação da formulação de pórtico espacial para os tipos mais simples de estruturas reticuladas e validou-se, portanto, a implementação desta formulação no caso de modelos reticulados cujos materiais são elásticos lineares. Destaca-se ainda a capacidade da formulação proposta neste trabalho de representar uma seção transversal de geometria qualquer. No entanto, ressalta-se que deve ser sempre considerada a influência da discretização da seção transversal nos resultados obtidos para os deslocamentos da estrutura.

6.2.2 Efeito de Bloqueio da Solução

Apresenta-se a seguir a comparação entre a solução exata e a solução de elementos finitos para uma viga em balanço submetida a uma carga concentrada na extremidade, como mostra a Figura 6.7. Esta comparação tem o intuito de demonstrar, como sugerido por Oñate (1995), o efeito de bloqueio da solução, que ocorre quando se utiliza para a discretização de uma barra esbelta um elemento de 2 nós baseado na teoria de Timoshenko. E com a demonstração deste fenômeno, pretende-se, além de validar a implementação dos elementos finitos apresentados no Capítulo 3, destacar os cuidados que devem ser tomados na sua utilização.

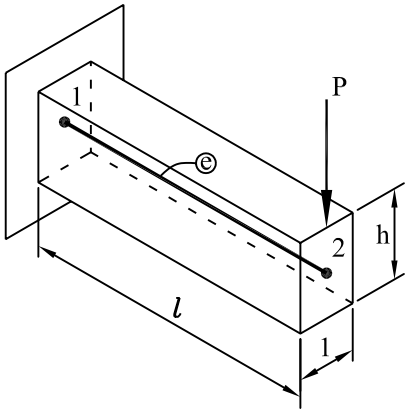


Figura 6.7: Viga em balanço submetida a uma carga concentrada na extremidade.

A solução para o deslocamento vertical da extremidade livre da viga segundo a teoria clássica de vigas, dita solução exata, é dada por:

$$w^f_{exata} = \frac{l^3}{3EI}\ P \tag{6.1}$$

Considerando os efeitos dos esforços cortantes, e sendo constante a distribuição de tensões tangenciais ao longo da seção transversal, a solução é acrescida de uma parcela da seguinte forma:

$$w^c_{exata} = \frac{l^3}{3EI}\ P + \frac{l}{\alpha GA}\ P \tag{6.2}$$

onde α é o fator de correção de cisalhamento da seção transversal.

Para a viga em balanço, obteve-se as soluções exatas através das Equações 6.1 e 6.2

variando-se os valores do coeficiente de esbeltez da viga, dado por:

$$\lambda = \frac{l}{h} \tag{6.3}$$

Através do **INSANE**, a viga foi analisada com 1 elemento finito de 2 nós de Euler-Bernoulli e com 1 elemento finito de 2 nós de Timoshenko, ambos com 2 pontos de integração. A seção transversal da viga foi discretizada de forma que a inércia calculada pelo programa fosse próxima o suficiente da inércia exata. Na análise, obtiveram-se valores do deslocamento vertical da extremidade livre da viga para vários coeficientes de esbeltez.

Para comparação dos resultados obtidos, foi tomado como referência o valor da solução exata da teoria clássica de vigas, através da razão

$$\varphi = \frac{w}{w^f_{exata}} \tag{6.4}$$

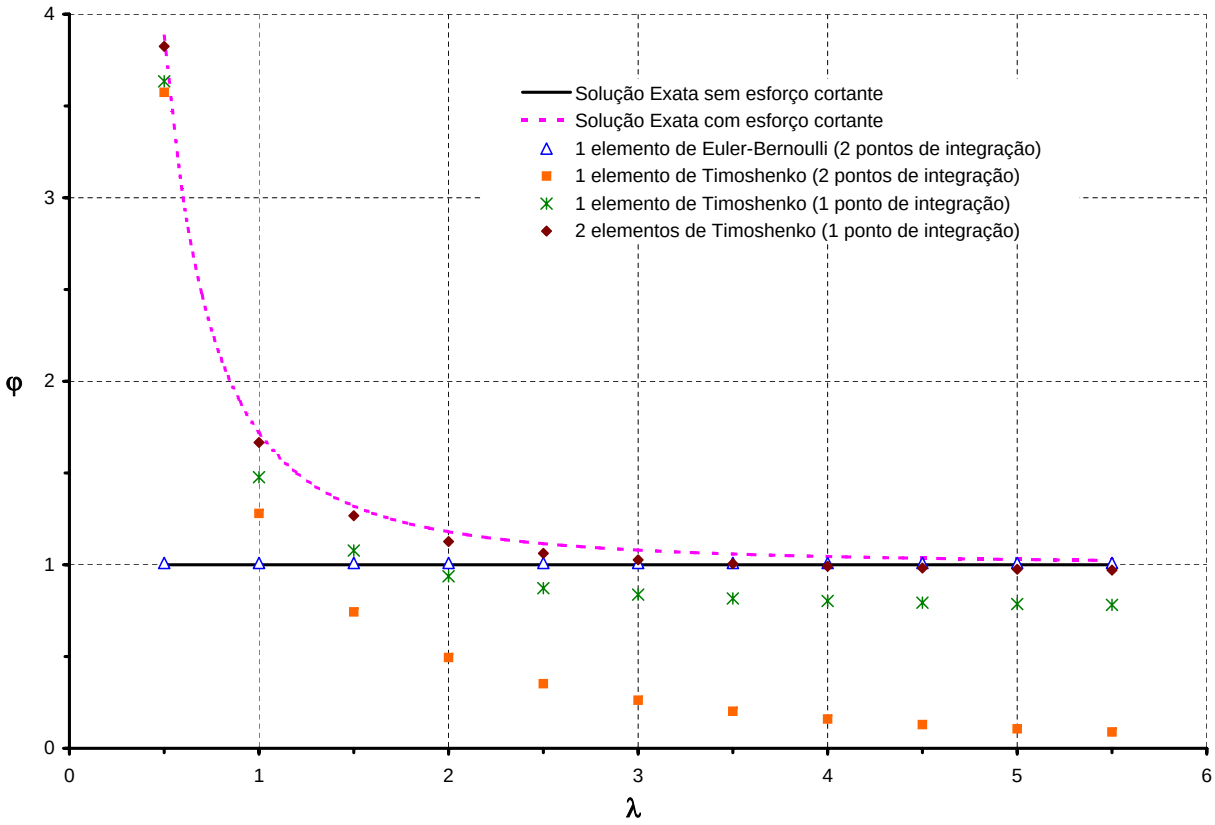


Figura 6.8: Variação da razão φ com o coeficiente de esbeltez λ para uma viga em balanço.

O gráfico da Figura 6.8 mostra a variação da razão entre as soluções obtidas e a solução exata da teoria clássica de vigas com o coeficiente de esbeltez da viga.

É possível observar no gráfico que, como já é conhecido para uma viga esbelta, os efeitos dos esforços cortantes são insignificantes e, portanto, o resultado obtido pela Equação 6.2 deve coincidir com o resultado da Equação 6.1. Observa-se ainda que os resultados obtidos com 1 elemento de 2 nós segundo a teoria de Euler-Bernoulli coincidiram com os resultados da solução exata da teoria clássica de vigas, como esperado.

No entanto, com o aumento do coeficiente de esbeltez da viga, o elemento finito de 2 nós segundo a teoria de Timoshenko, com 2 pontos de integração, não foi capaz de convergir para solução exata da teoria clássica de vigas. Isso se deve a um fenômeno de superestimativa numérica da rigidez, o qual bloqueia a solução à medida que a esbeltez da viga aumenta. Daí conclui-se que esse elemento só é adequado para vigas com baixo coeficiente de esbeltez, ditas vigas altas.

Uma técnica usual para solucionar este inconveniente é a adoção da integração reduzida. Com ela objetiva-se, nesse caso, subintegrar-se a parcela da matriz de rigidez devido a cortante, utilizando uma ordem de integração inferior ao necessário para obter o seu cálculo exato. Assim a flexibilidade da estrutura deve aumentar, contrabalanceando a excessiva rigidez introduzida pelo cortante.

Para comprovar esse fato, utilizou-se na análise da viga, para vários coeficientes de esbeltez, 1 elemento finito de 2 nós de Timoshenko, com 1 ponto de integração somente. Com a integração reduzida, os resultados obtidos com o aumento do coeficiente de esbeltez foram mais próximos da solução da teoria clássica de vigas. Como a solução obtida com elementos baseados na teoria de Timoshenko não é exata, pode-se obter a convergência aumentando o número de elementos finitos. Isto foi comprovado analisando a viga com 2 elementos finitos de 2 nós de Timoshenko, com 1 ponto de integração cada.

6.2.3 Elementos Finitos de Timoshenko

Como visto no exemplo anterior, os elementos finitos baseados na teoria de Euler-Bernoulli são inadequados para vigas com coeficiente de esbeltez baixo, uma vez que não consideram os efeitos dos esforços cortantes. Já os elementos baseados na teoria de Timoshenko contemplam em sua formulação este tipo de efeito, sendo adequados para análise de vigas altas. Porém, a precisão da solução obtida com elementos finitos depende do grau do polinômio de aproximação utilizado, que deve ser, no mínimo, da mesma ordem da função polinomial da solução exata.

No **INSANE**, estão disponíveis elementos finitos baseados na teoria de Timoshenko de 2, 3 e 4 nós. Para discutir a precisão destes 3 tipos de elementos, e validar a sua implementação, serão analisadas duas vigas altas, sendo uma viga em balanço com carga concentrada na extremidade e uma viga bi-apoiada com carga distribuída (Figura 6.9). Os resultados obtidos serão comparados com as soluções analíticas da teoria da elasticidade que consideram o efeito dos esforços cortantes retiradas de Timoshenko e Goodier (1980).

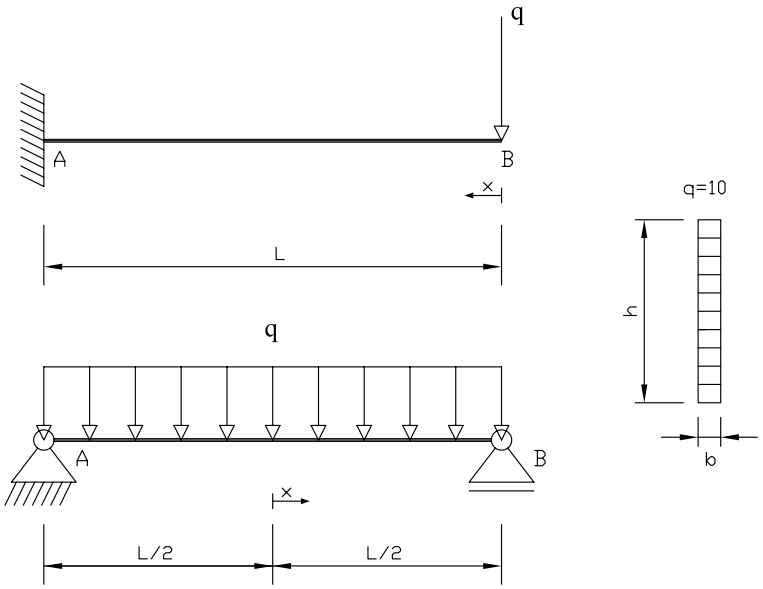


Figura 6.9: Viga alta em balanço e viga alta bi-apoiada.

A Figura 6.9 mostra as vigas em estudo, sendo, em unidades de comprimento (uc) e unidades de força (uf), L=2,0 uc, h=1,0 uc e b=0,12 uc, com o coeficiente de esbeltez igual a 2. Para a viga em balanço, q=10,0 uf, e para a viga bi-apoiada, q=10,0 uf/uc. Foi considerado material elástico linear com módulo de elasticidade longitudinal igual a 12,0 uf/uc^2, coeficiente de poisson igual a 0,2 e módulo de elasticidade transversal igual a 5,0 uf/uc^2. A seção transversal foi discretizada em 10 áreas e, como é uma seção retangular, tem o fator de correção de cisalhamento igual a $\frac{5}{6}$. Com a discretização adotada para a seção transversal, obteve-se um erro relativo de 1,0% entre a inércia aproximada calculada pelo programa e a inércia exata.

Para a viga em balanço, a solução exata da teoria da elasticidade para o deslocamento vertical é dada por:

$$w(x) = [\ \frac{x^3}{6E} - \frac{L^2 x}{2E} + \frac{L^3}{3E} + \frac{h^2}{8G}\ (L - x)\ ]\ \frac{q}{I} \tag{6.5}$$

A viga em balanço foi analisada no **INSANE** com os 3 tipos de elementos de Timoshenko disponíveis: 1 elemento finito de 2 nós com 2 pontos de integração, 1 elemento finito de 3 nós com 3 pontos de integração e 1 elemento finito de 4 nós com 4 pontos de integração. A Figura 6.10 mostra os resultados, em unidades adimensionais, dos deslocamentos verticais dos nós calculados pelo programa para as três malhas, juntamente com a solução exata da teoria da elasticidade.

Como a Equação 6.5 mostra, a solução do deslocamento vertical para uma viga em balanço

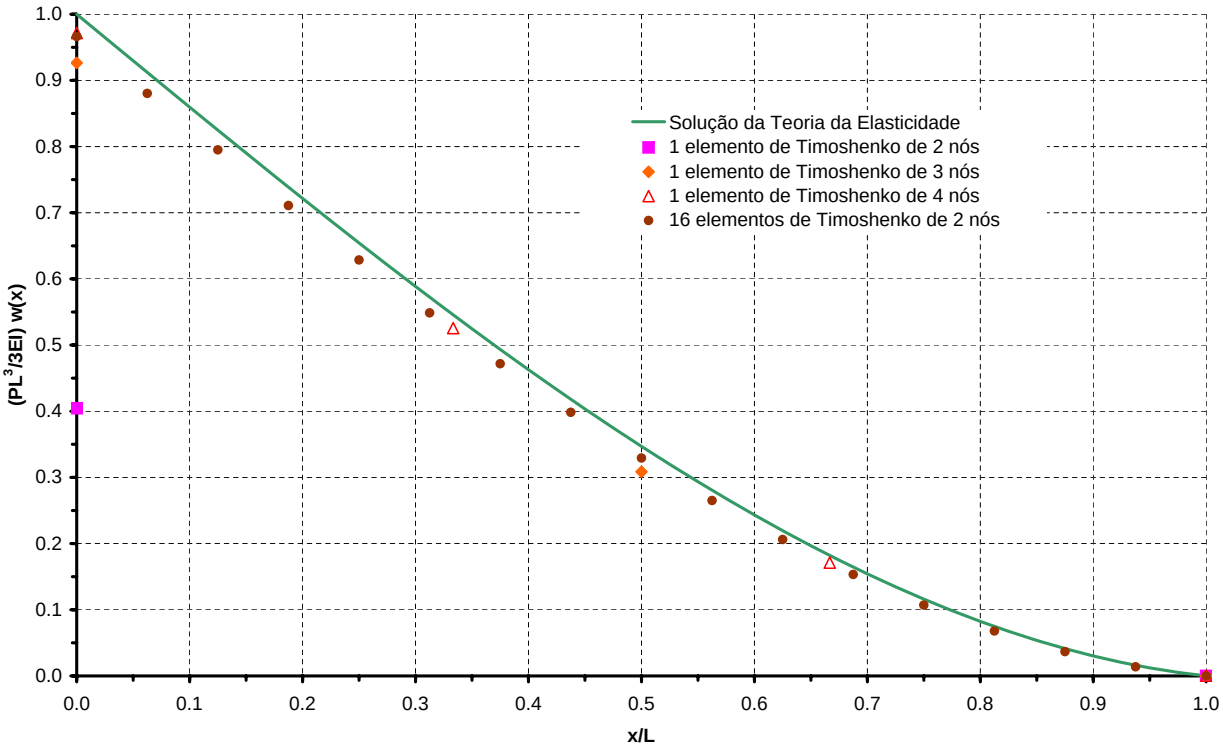


Figura 6.10: Resultados para viga alta em balanço.

No entanto, a solução obtida com 1 elemento finito de Timoshenko de 4 nós, mesmo sendo satisfatória, não coincide com a solução exata por duas razões: primeiro, a influência da discretização da seção na rigidez da estrutura; e segundo, a consideração na teoria da elasticidade do efeito do empenamento da seção transversal devido aos esforços cortantes sobre as deflexões da viga, enquanto a hipótese adotada na formulação do elemento finito de Timoshenko diz que as seções normais ao eixo da barra permanecem planas após a flexão.

Para os elementos com funções aproximadoras mais simples, como o elemento de Timoshenko de 2 nós, é possível obter uma boa aproximação para soluções de alto grau utilizando-se um maior número de elementos. Isto é demonstrado pela solução satisfatória obtida no **INSANE** com 16 elementos finitos de 2 nós, como também mostra a Figura 6.10. A opção de utilizar uma malha com vários elementos simples ou uma malha com poucos elementos de ordem superior fica a cargo do analista, que, em análises mais complexas, deve fazer uso de sua experiência para pesar entre esforço de cálculo e precisão.

No caso da viga bi-apoiada, a solução exata da teoria da Elasticidade para o deslocamento vertical é dada por:

$$w(x) = \delta + \frac{q}{2EI} \left[\frac{L^2 x^2}{8} - \frac{x^4}{12} - \frac{h^2 x^2}{20} + \frac{h^2 x^2}{4} \left(1 + \frac{\upsilon}{2}\right) \right] \tag{6.6}$$

onde

$$\delta = \frac{5qL^4}{384EI} \left[1 + \frac{12h^2}{5L^2} \left(\frac{4}{5} + \frac{\upsilon}{2}\right) \right] \tag{6.7}$$

Esta solução é uma equação de quarto grau, sendo, portanto, de ordem maior que todas as funções aproximadoras dos elementos finitos de Timoshenko disponíveis no **INSANE**. Para a obtenção de resultados precisos, é necessário então utilizar mais de um elemento. Para efeito de comparação, foram analisadas quatro malhas distintas, sendo: 2 elementos finitos de 2 nós com 2 pontos de integração cada, 2 elementos finitos de 3 nós com 3 pontos de integração cada, 2 elementos finitos de 4 nós com 4 pontos de integração cada, e, finalmente, 16 elementos finitos de 2 nós com 2 pontos de integração cada. A Figura 6.11 mostra os resultados, em unidades adimensionais, dos deslocamentos verticais dos nós calculados pelo programa para as quatro malhas, juntamente com a solução exata da teoria da elasticidade.

Os resultados para 2 elementos de 3 e 4 nós foram bastante satisfatórios, sendo que o aumento do número de elementos compensou a deficiência da ordem da função aproximadora. Já o resultado com 2 elementos de 2 nós não se aproximou da solução exata, devido ao baixo grau de sua função, sendo necessário aumentar o número de elementos para 16, obtendo-se assim um resultado satisfatório.

Nas análises dos dois exemplos de vigas altas, contatou-se que é possível obter bons resultados com os elementos finitos baseados na teoria de Timoshenko disponíveis no **INSANE**. Mesmo com o elemento mais simples isto é possível, ainda que necessário utilizar vários elementos na discretização do problema para uma melhor aproximação dos resultados.

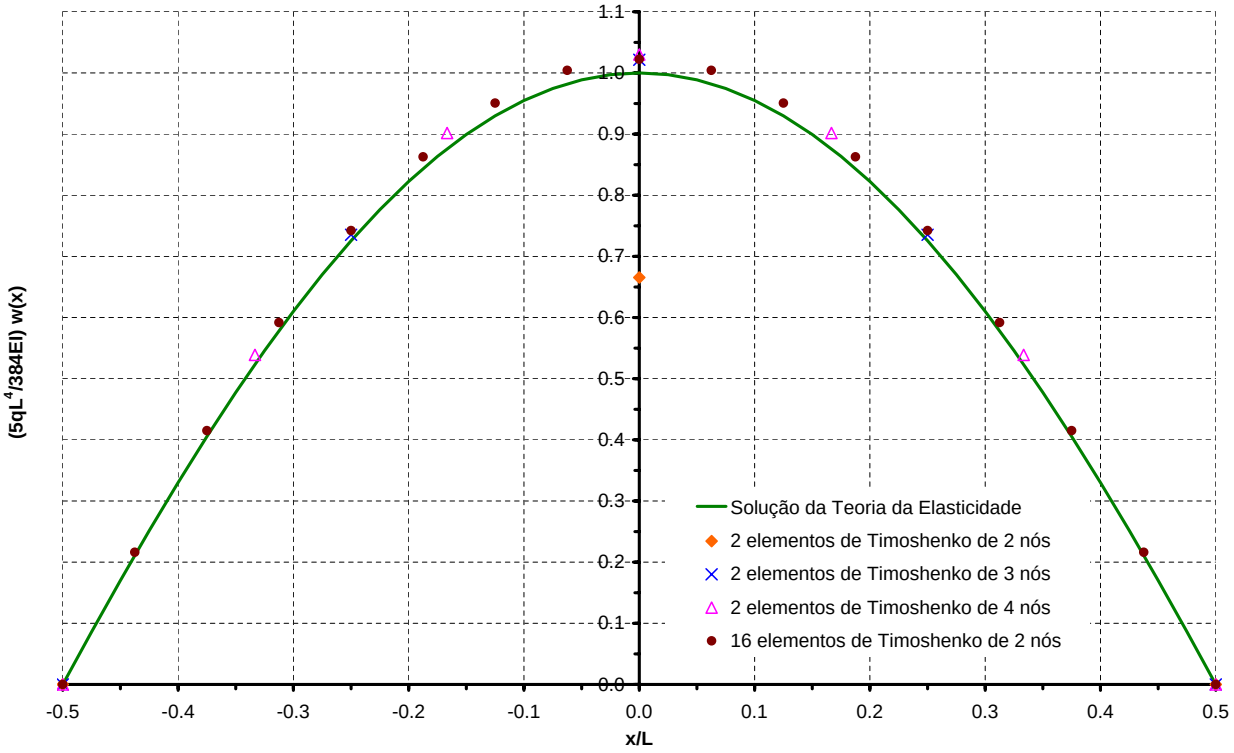


Figura 6.11: Resultados para viga alta bi-apoiada.

6.2.4 Verificação do Cálculo de Tensões e Deformações

Este exemplo tem o objetivo de verificar os valores calculados pelo **INSANE** para as tensões e deformações generalizadas da seção transversal de um dos pontos de integração do elemento e para a distribuição de tensões ao longo desta seção. Para isso, foi modelada uma barra em balanço que tem aplicadas em sua extremidade uma carga concentrada na direção de seu eixo, duas cargas concentradas perpendiculares ao eixo da barra e perpendiculares entre si e um momento de torção, como mostra a Figura 6.12. Desta forma a barra está submetida a todos os esforços de um pórtico espacial.

A barra tem comprimento l=20,0 uc e seção circular de diâmetro d=2,0 uc, com o coeficiente de esbeltez igual a 10. As cargas concentradas $\boldsymbol{P}$ têm valor unitário, sendo todas no sentido negativo do eixo em que estão aplicadas, e o momento de torção tem valor unitário e sentido positivo. Foi considerado material elástico linear com módulo de elasticidade longitudinal igual a 12,0 uf/uc^2, coeficiente de poisson igual a 0,2 e módulo de elasticidade transversal

igual a 5,0 uf/uc^2. A seção transversal foi discretizada em 360 áreas e, como é uma seção circular, tem o fator de correção de cisalhamento igual a $\frac{6}{7}$. Com a discretização adotada para a seção transversal, obteve-se um erro relativo de 0,5% entre as inércias aproximadas calculadas pelo programa e as inércias exatas.

Na análise, utilizou-se um elemento finito baseado na teoria de Timoshenko de 4 nós. Apesar da barra ser esbelta, optou-se por um elemento que considera os esforços cortantes para que estes também possam ser verificados. Para evitar o bloqueio da solução recorreu-se ao artifício da integração reduzida, utilizando-se 3 pontos de integração.

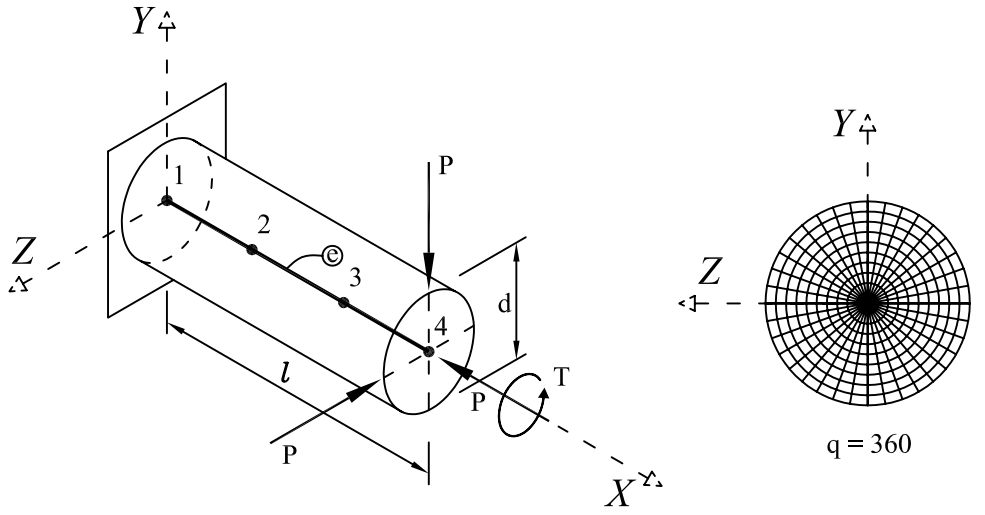


Figura 6.12: Barra em balanço submetida a todos os esforços de um pórtico espacial.

A Tabela 6.2 mostra a comparação entre os resultados analíticos segundo a teoria de Timoshenko e os resultados obtidos pelo **INSANE** para os deslocamentos nodais e as tensões e deformações generalizadas na seção transversal do ponto de integração localizado no meio da barra. Pode-se observar nos resultados dos deslocamentos e deformações generalizadas a influência da aproximação das propriedades da seção transversal, provocando nas variáveis dependentes destas propriedades um erro relativo de 0,5%. Os deslocamentos nas direções y e z foram também influenciados pela integração reduzida adotada, que evita o efeito de bloqueio, mas que também perde em precisão da integração numérica. Já os resultados obtidos para as tensões generalizadas coincidiram com os valores da solução analítica.

Tabela 6.2: Comparação dos resultados para a barra espacial.

Ponto	*Variável*	*Solução Analítica*	*Resultado Obtido*	*Erro Relativo (%)*
Nó 4 (x=*l*)	dx	-5.305E-01 uc	-5.305E-01 uc	0.00
	dy	-2.829E+02 uc	-2.858E+02 uc	1.03
	dz	-2.829E+02 uc	-2.858E+02 uc	1.03
	Θx	2.546E+00	2.559E+00	0.50
	Θy	2.122E+01	2.133E+01	0.50
	Θz	-2.122E+01	-2.133E+01	0.50
x = *l*/2	N	-1.000E+00 uf	-1.000E+00 uf	0.00
	Vy	-1.000E+00 uf	-1.000E+00 uf	0.00
	Vz	-1.000E+00 uf	-1.000E+00 uf	0.00
	T	1.000E+00 uf.uc	1.000E+00 uf.uc	0.00
	My	1.000E+01 uf.uc	1.000E+01 uf.uc	0.00
	Mz	-1.000E+01 uf.uc	-1.000E+01 uf.uc	0.00
	ε_a	-2.653E-02	-2.653E-02	0.02
	γ_y	-7.427E-02	-7.428E-02	0.01
	γ_z	-7.427E-02	-7.428E-02	0.01
	ψ	1.273E-01	1.280E-01	0.53
	κ_y	1.061E+00	1.066E+00	0.47
	κ_z	-1.061E+00	-1.066E+00	0.47

Uma vez analisadas as tensões e deformações generalizadas na seção transversal e constatados resultados satisfatórios, as distribuições de tensões e deformações ao longo da seção têm seus resultados garantidos, pois são funções destas variáveis. As Figuras 6.13 e 6.14 mostram, respectivamente, os resultados para tensões normais axiais e para tensões tangenciais ao longo do eixo y da seção transversal do meio da barra. Como esperado, os resultados obtidos coincidiram com as soluções analíticas. Observa-se ainda que a tensão tangencial σ_{xy} ao longo do eixo y da seção é constante, como adotado por hipótese, enquanto a tensão tangencial σ_{xz} é acrescida do efeito de cisalhamento devido à torção.

O **INSANE** mostrou-se capaz de fornecer com precisão os valores de tensões e deformações nos pontos de integração do elemento finito, sejam tensões e deformações generalizadas da seção transversal, sejam tensões e deformações de um dos pontos da seção. No entanto, para que isso aconteça, é imprescindível garantir uma aproximação suficientemente precisa para os deslocamentos nodais.

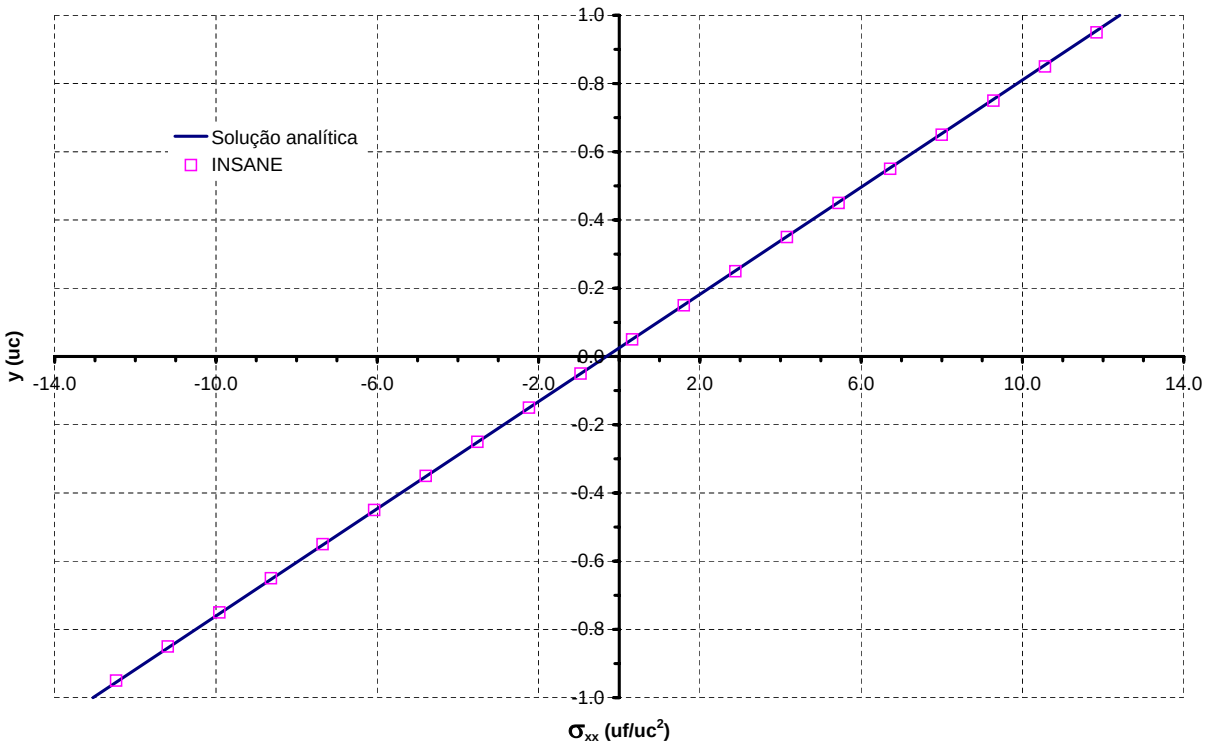


Figura 6.13: Resultados para tensões normais axiais ao longo do eixo y da seção transversal do meio da barra.

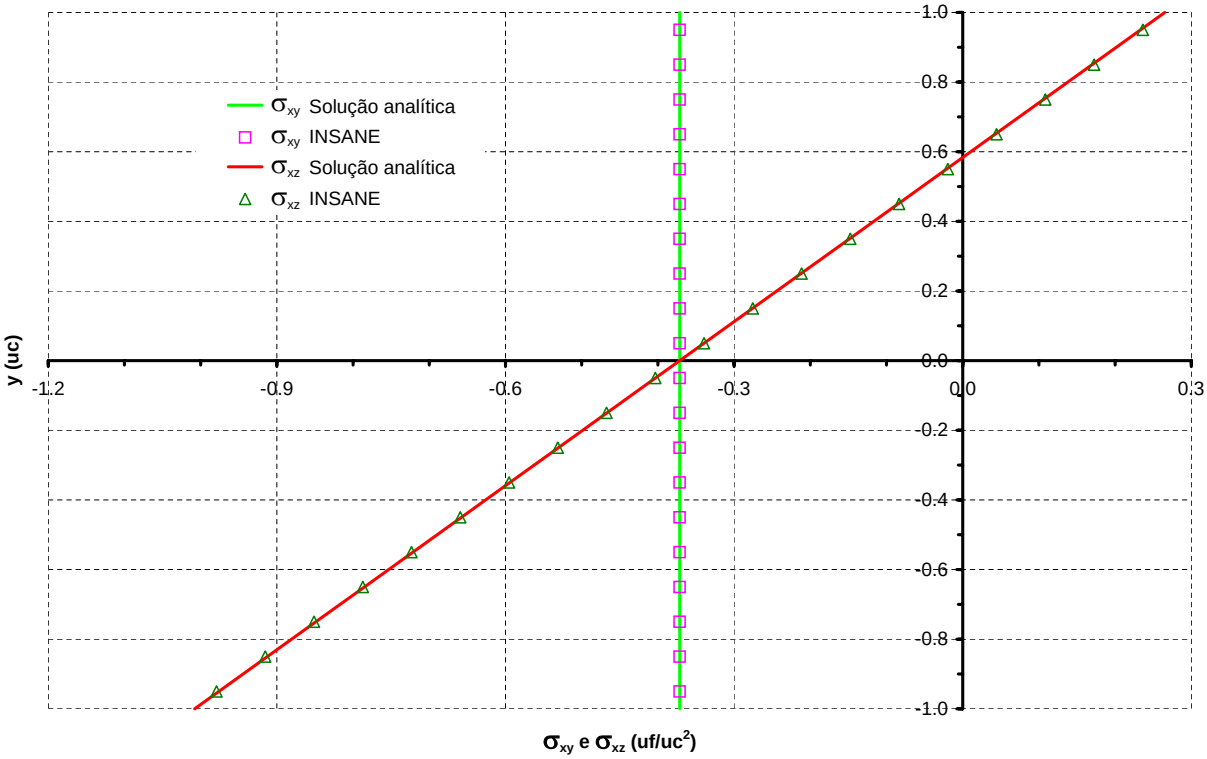


Figura 6.14: Resultados para tensões tangenciais ao longo do eixo y da seção transversal do meio da barra.

6.3 Problemas Fisicamente Não-Lineares

6.3.1 Pilar Circular

A Figura 6.15 mostra a configuração geométrica de um pilar de concreto armado com seção circular sujeito à compressão centrada, descrito em Simão (2003).

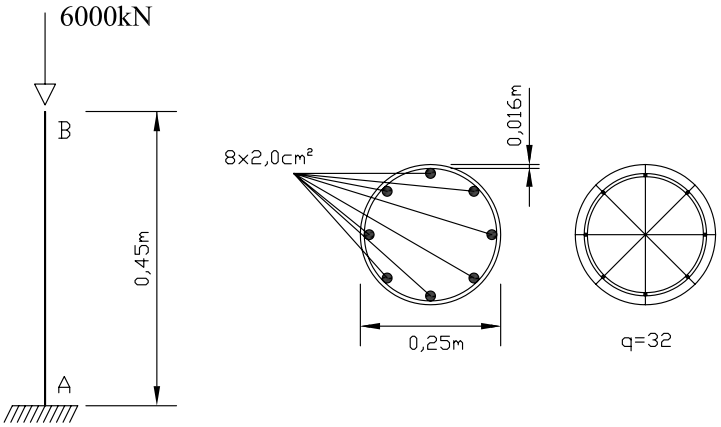


Figura 6.15: Pilar circular de concreto armado.

O pilar foi analisado através do **INSANE**, adotando-se em sua discretização 1 elemento finito de Euler-Bernoulli de 2 nós com dois pontos de integração. A seção transversal circular

O resultado obtido com o concreto da NBR6118 (2003) aproximou-se mais do resultado experimental, uma vez que não leva em consideração o amolecimento do concreto, apresentando um comportamento dúctil após a carga máxima. Quando considerada a lei proposta por Carreira e Chu (1985), observa-se o efeito do amolecimento do concreto e a conseqüente redução da ductilidade do pilar. Pode-se ainda observar que quando aumenta-se a deformação relativa à tensão máxima de compressão ε_c, suaviza-se o amolecimento do concreto. Mesmo assim, o modelo constitutivo proposto por Carreira e Chu (1985) não foi capaz de representar o comportamento pós-crítico do pilar.

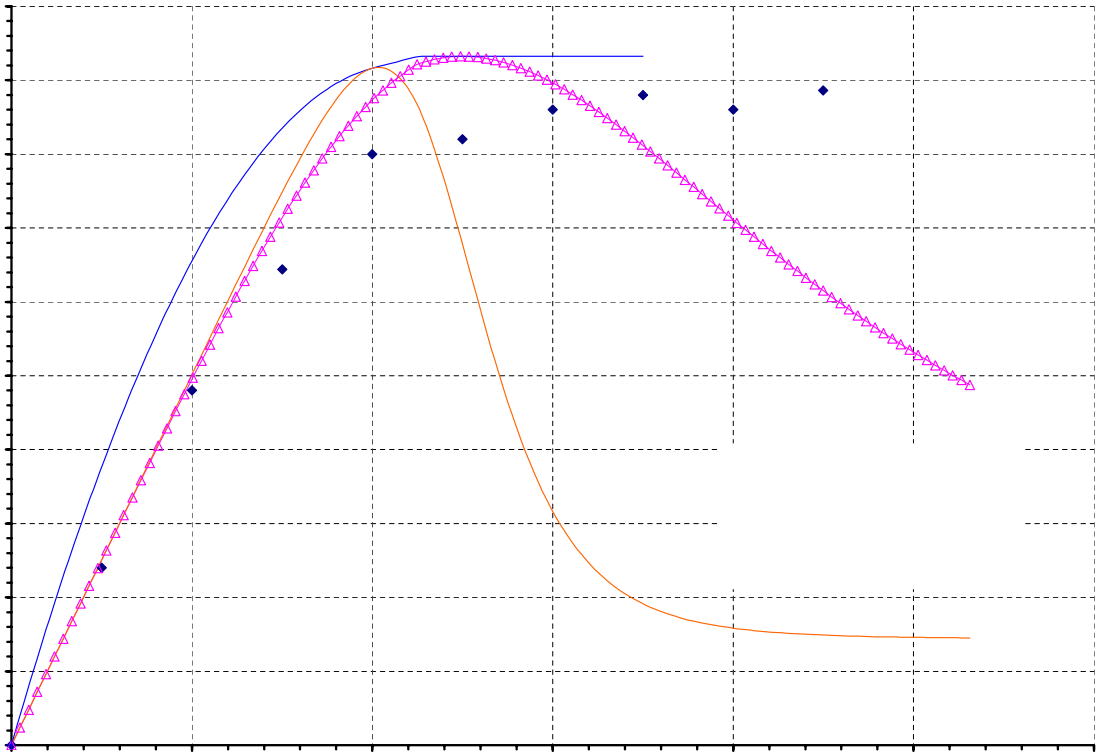


Figura 6.16: Trajetórias de equilíbrio para o pilar circular.

6.3.2 Viga I

Analisou-se um viga de concreto armado com seção I, bi-apoiada, sobre o efeito de duas cargas concentradas, descrita por Neves (2000), e que está representada na Figura 6.17. A viga, mesmo sendo uma viga esbelta, foi analisada segundo a teoria de Euler-Bernoulli e segundo a teoria de Timoshenko.

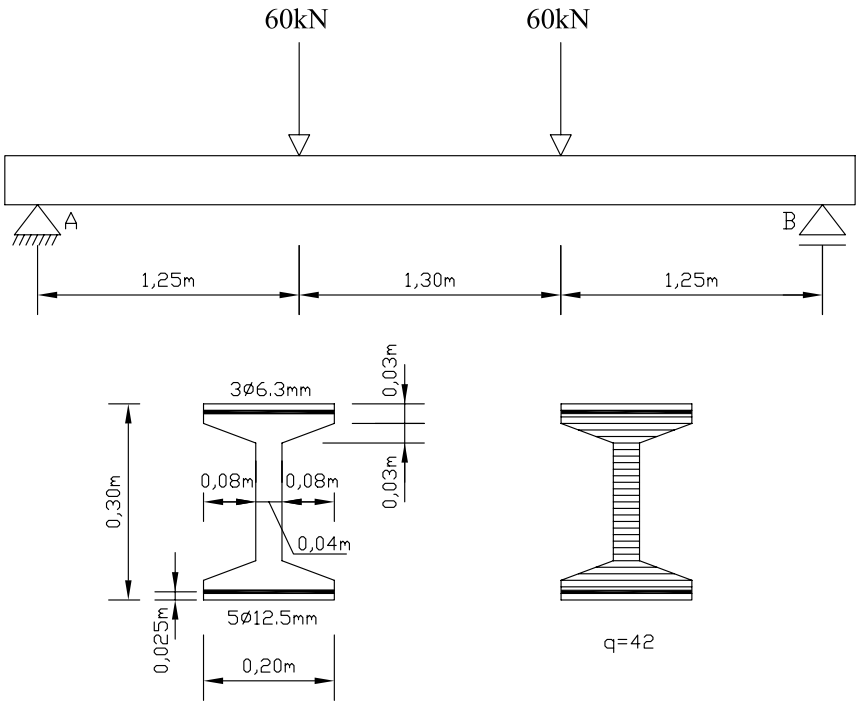


Figura 6.17: Viga I de concreto armado.

Para a discretização segundo a teoria de Euler-Bernoulli foram utilizados 6 elementos de 2 nós com 2 pontos de integração cada. Já para a discretização segundo a teoria de Timoshenko foram utilizados 12 elementos de 2 nós com integração reduzida (1 ponto de integração cada) para evitar o efeito de bloqueio da solução. A seção transversal da viga foi discretizada em 42 áreas ao longo de sua altura, sendo 2 delas representando as armaduras da viga.

Considerou-se a lei constitutiva proposta por Carreira e Chu (1985), sendo f_c=21MPa, f_t=2,1MPa, E_0=29632MPa, ε_c=0,002 e ε_t=0,0002. Para o aço considerou-se comportamento elasto-plástico sem endurecimento com E_s=177890MPa e f_y=500MPa.

Na análise não-linear da viga, adotou-se o método de controle de deslocamento generalizado, com incremento de 2×10^{-1} e com uma tolerância para convergência de 1×10^{-4}.

A Figura 6.18 mostra as trajetórias de equilíbrio correspondentes ao deslocamento vertical do meio do vão da viga. Vale ressaltar que a resposta com elementos segundo a teoria de Euler-Bernoulli e a resposta com elementos segundo a teoria de Timoshenko coincidiram, pois a malha com o 12 elementos de Timoshenko com integração reduzida é capaz de aproximar a resposta de um viga esbelta da mesma forma que a metade de elementos de Euler-Bernoulli.

Observa-se uma satisfatória concordância entre os resultados numéricos e os resultados experimentais adaptados de Neves (2000). Cabe ainda ressaltar que a simulação numérica foi capaz de descrever o regime pós-crítico da viga. No entanto, não há dados experimentais equivalentes para comparação.

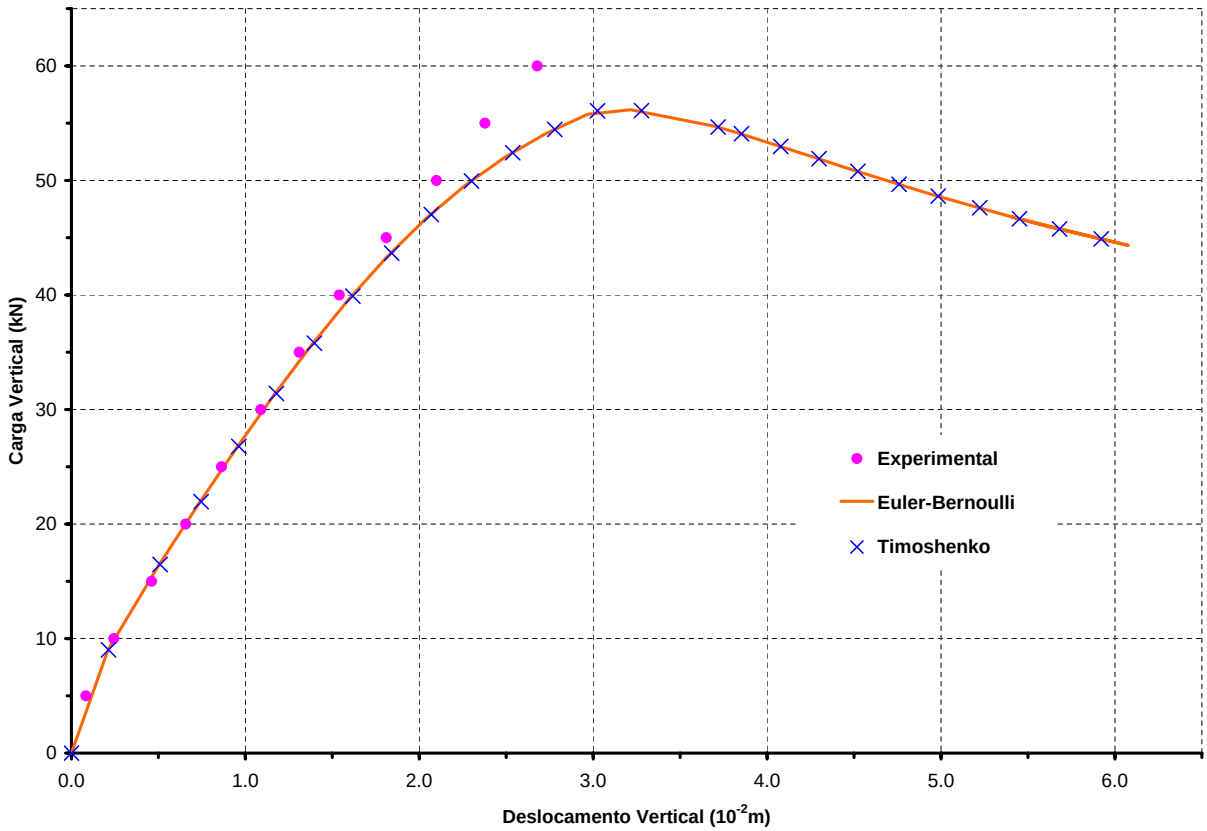


Figura 6.18: Trajetórias de equilíbrio para viga I.

6.3.3 Pórtico Plano de 2 Andares

A Figura 6.19 mostra o pórtico estudado numérica e experimentalmente por Vecchio e Emara (1992) e modelado neste exemplo. O pórtico foi analisado segundo as duas teorias propostas neste trabalho, sendo discretizado da seguinte forma: 28 elementos de 2 nós com 2 pontos de integração para a teoria de Euler-Bernoulli, sendo 6 elementos em cada pilar e 8 em cada viga; e em 56 elementos de 2 nós com integração reduzida para a teoria de Timoshenko, sendo 12 elementos em cada pilar e 16 em cada viga. As seções transversais dos pilares e das vigas foram discretizadas da mesma maneira, com um total de 42 áreas ao longo de sua altura, sendo 2 delas representando as armaduras. Como é uma seção retangular, tem o fator de correção de cisalhamento igual a $\frac{5}{6}$.

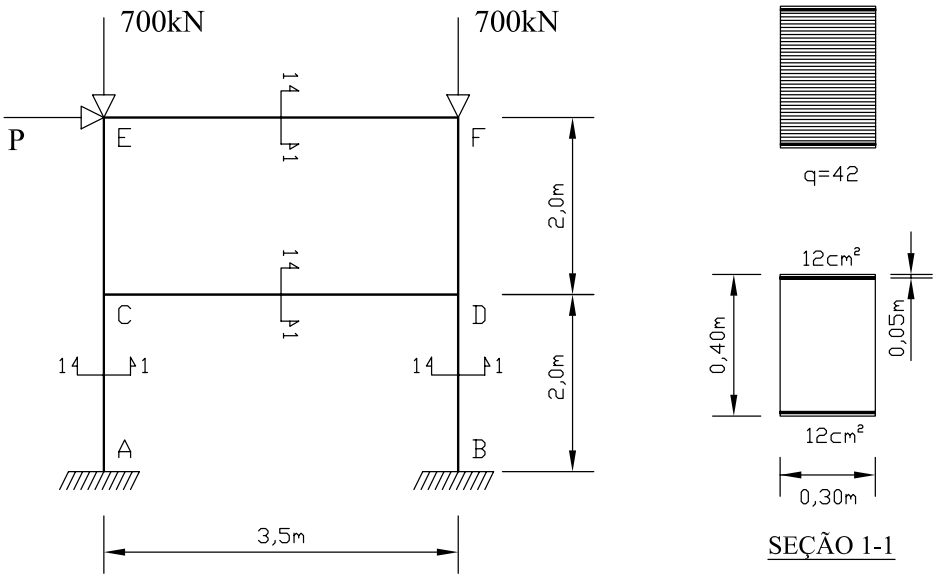


Figura 6.19: Pórtico Plano de concreto armado.

Considerou-se a lei constitutiva proposta por Carreira e Chu (1985), sendo f_c=30MPa, f_t=3MPa, E_0=30000MPa, ε_c=0,002 e ε_t=0,0002. Para o aço considerou-se comportamento elasto-plástico sem endurecimento com E_s=192500MPa e f_y=418MPa.

Na análise não-linear do pórtico, adotou-se o método de controle direto de deslocamentos, com o qual controlou-se o deslocamento horizontal do ponto E da Figura 6.19 incrementado-o de 1,5$\times 10^{-3}m$, com uma tolerância para convergência de 1×10^{-3}. O pórtico está sujeito a

cargas constantes de 700 kN nos pontos E e F, e sujeito a uma carga lateral incremental $\boldsymbol{P}$ no ponto E.

As trajetórias de equilíbrio correspondentes ao deslocamento horizontal do ponto E do pórtico estão representadas na Figura 6.20. Os resultados numéricos obtidos foram satisfatórios quando comparados com os resultados experimentais. Apesar de simularem uma resposta mais rígida da estrutura, no regime crítico demonstraram o mesmo comportamento dúctil dos resultados experimentais.

Os resultados mostram que a formulação considerando a teoria de Timoshenko é mais adequada à solução do problema, uma vez que flexibiliza a estrutura devido à influência do cisalhamento. Esta influência pode ser vista na perda de capacidade de carga do pórtico, particularmente próximo às cargas máximas, sendo, portanto, o cisalhamento fator significativo nos deslocamentos totais da estrutura.

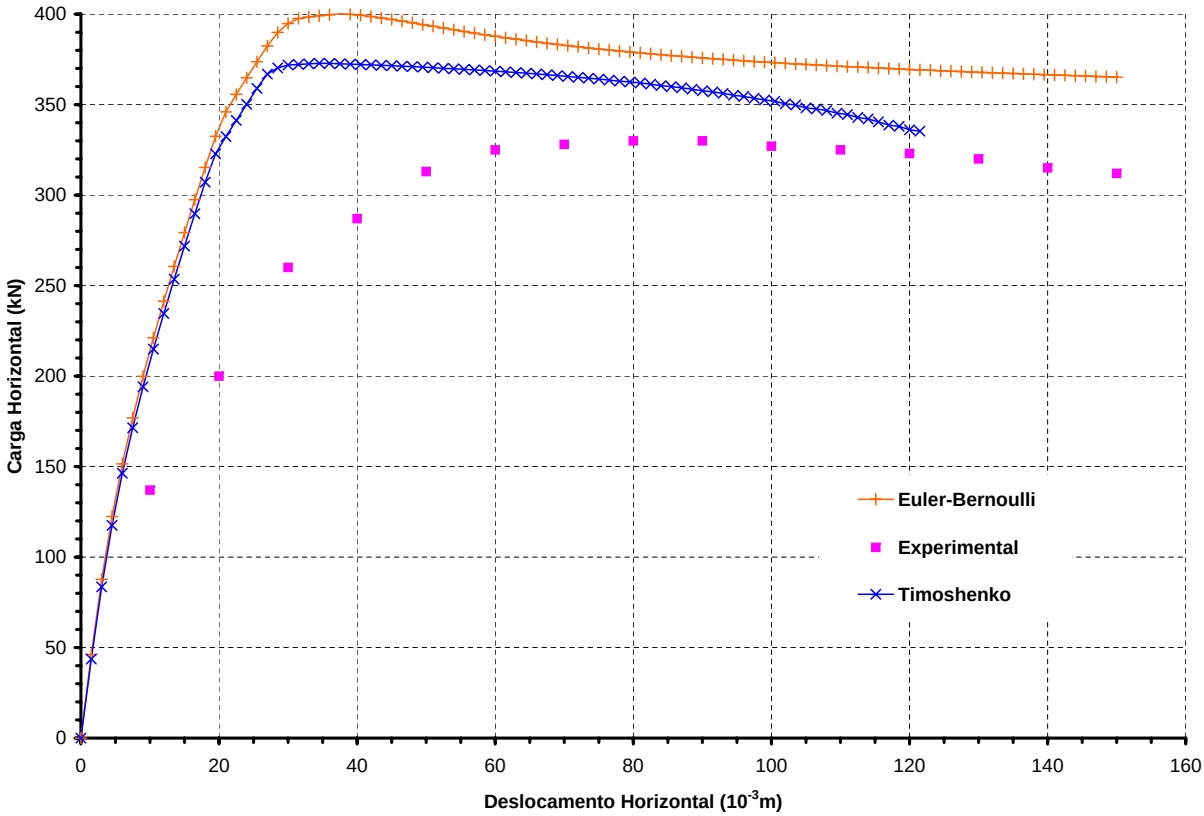


Figura 6.20: Trajetórias de equilíbrio para ponto E do pórtico.

6.3.4 Viga em Balanço de Concreto Armado

Para demonstrar o mecanismo de ruptura da viga, são mostrados para cada caso a distribuição de deformações na seção mais solicitada e os históricos de tensões e deformações axiais para a armadura tracionada e para a fibra mais comprimida.

Para a viga em concreto simples, a ruptura se deu por tração do concreto após atingir uma carga máxima referente ao fator igual à 0,66, como mostram as Figuras 6.23 e 6.24. Observa-se que, a medida que a carga foi incrementada, a linha neutra movimentou-se, reduzindo consideravelmente a área de concreto comprimida e produzindo o colapso da viga.

Quando considerada uma armadura com A_s igual à 1,5 cm^2, a ruptura da seção mais solicitada foi devido à deformação plástica excessiva da armadura, que, após atingir uma carga máxima referente ao fator igual à 1,00, alcançou a tensão de escoamento. Com isso, a viga perdeu capacidade de carga até alcançar o alongamento limite último, como mostram as Figuras 6.25 e 6.26. Mesmo com a ocorrência do escoamento da armadura, o concreto não atingiu sua tensão limite de compressão, como pode ser visto na Figura 6.27, caracterizando uma seção sub-armada correspondente ao domínio de deformação *2* da NBR6118.

Na Figura 6.28 é mostrada a distribuição de deformações na seção mais solicitada para a viga com A_s igual à 3,0 cm^2. As Figuras 6.29 e 6.30 mostram os históricos de tensões e deformações axiais para a armadura tracionada e para a fibra mais comprimida da seção, respectivamente. Pode ser visto que, como o caso anterior, a ruptura da seção mais solicitada deu-se devido à deformação plástica excessiva da armadura. No entanto, a fibra mais comprimida de concreto armado já havia alcançado sua tensão limite de compressão, mas ainda sem atingir seu limite último de deformação, o que novamente caracteriza uma seção sub-armada no domínio de deformação *2*.

Já a viga com A_s igual à 6,0 cm^2 rompeu com ocorrência simultânea do escoamento da armadura e esmagamento do concreto, experimentando um processo de movimentação da linha neutra que levou a armadura até o seu limite de alongamento e o concreto até seu limite último de deformação, como é mostrado nas Figuras 6.31, 6.32 e 6.33. Este processo corresponde ao limite entre os domínios de deformação *2* e *3* da NBR6118, caracterizando ainda uma seção sub-armada.

A viga com A_s igual à 12,0 cm^2 foi capaz de suportar maior nível de carga, com um fator de carga igual à 4,32. O seu mecanismo de ruptura deu-se, a partir desse ponto, pelo esmagamento do concreto, que após atingir a tensão limite de compressão chegou ao seu limite último de deformação, levando a viga à ruína sem que houvesse o escoamento do aço, como mostram as Figuras 6.34, 6.35 e 6.36. Neste caso, a seção caracterizou-se como super-armada, correspondendo ao domínio de deformação *4*.

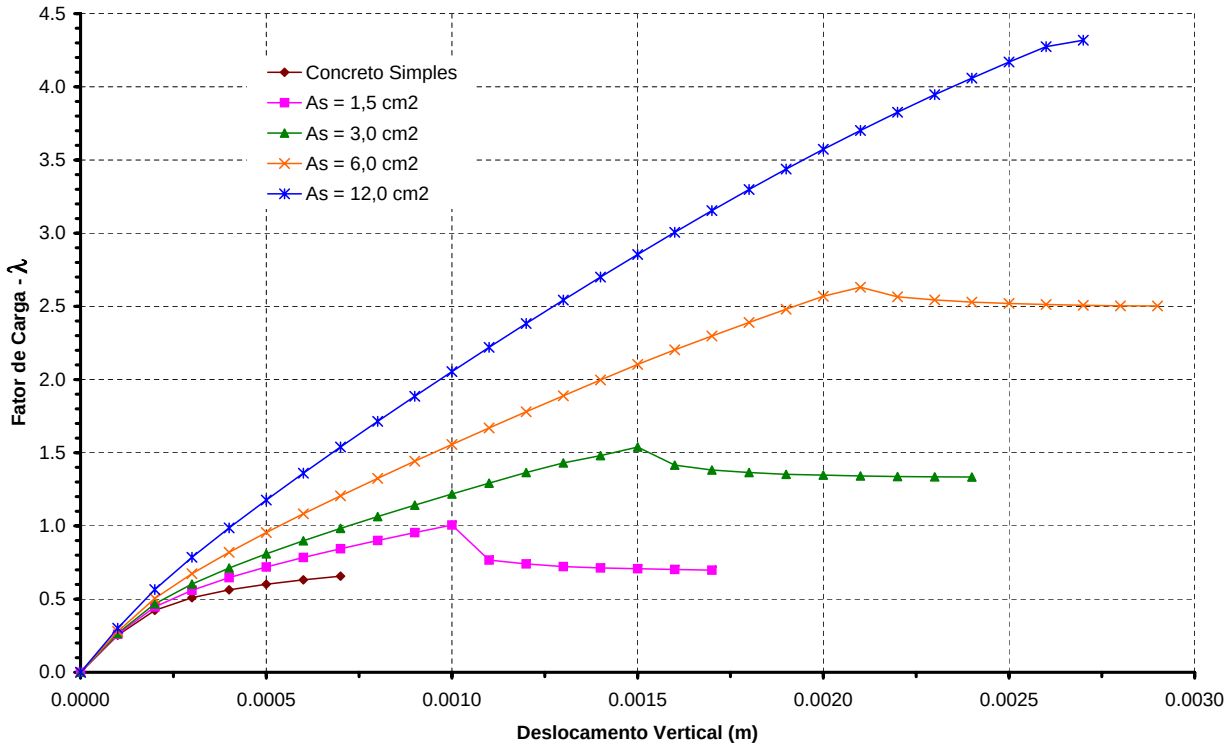


Figura 6.22: Trajetórias de equilíbrio para a viga em balanço de concreto armado.

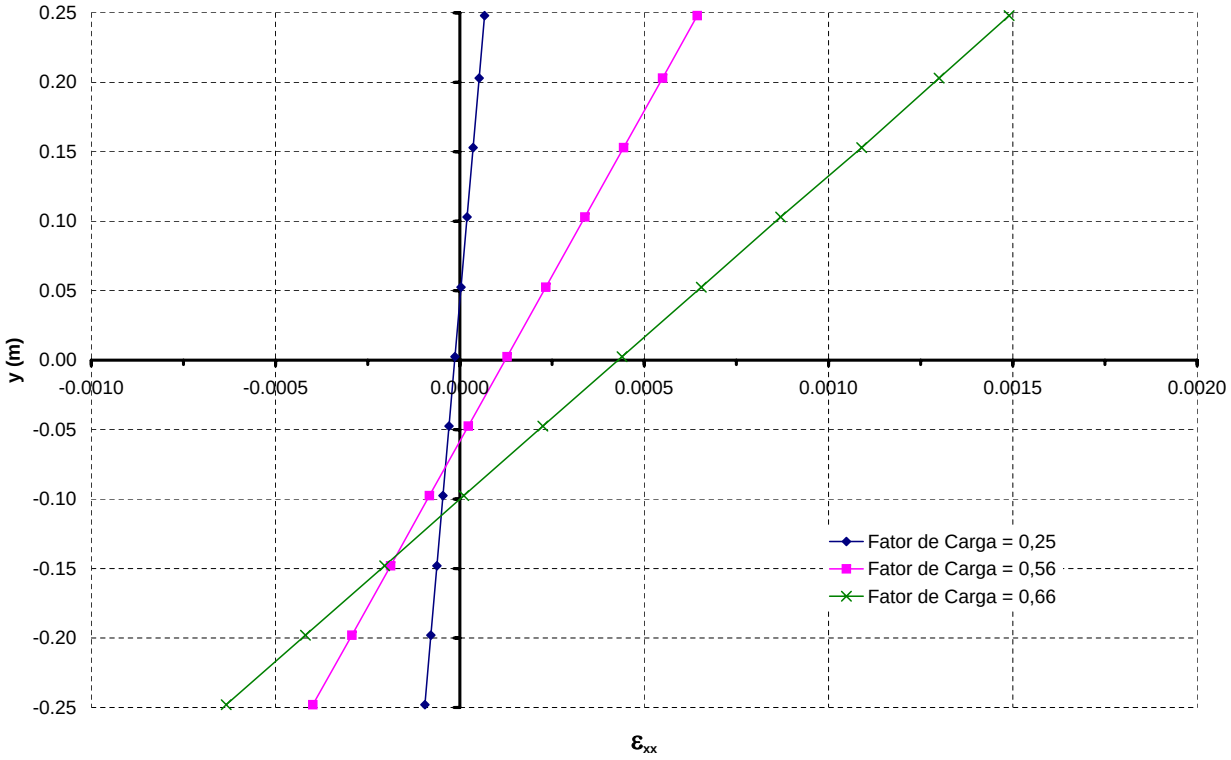


Figura 6.23: Deformações na seção transversal mais solicitada para viga em concreto simples.

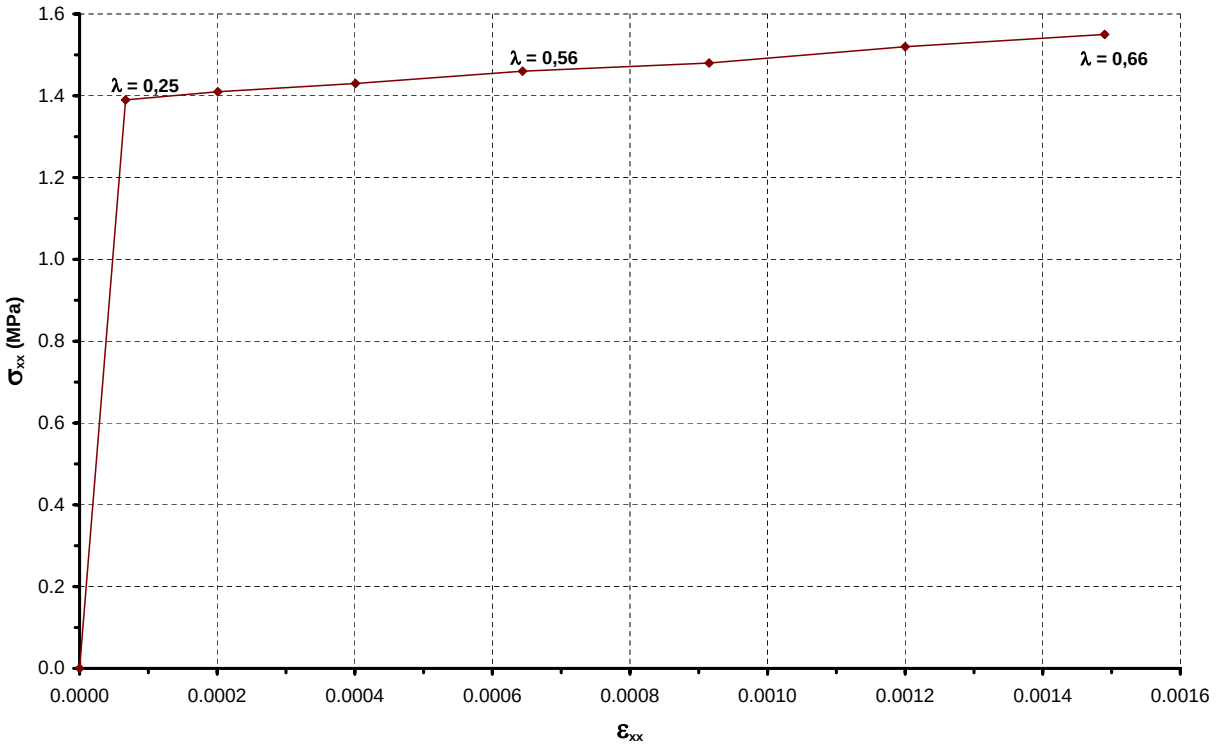


Figura 6.24:

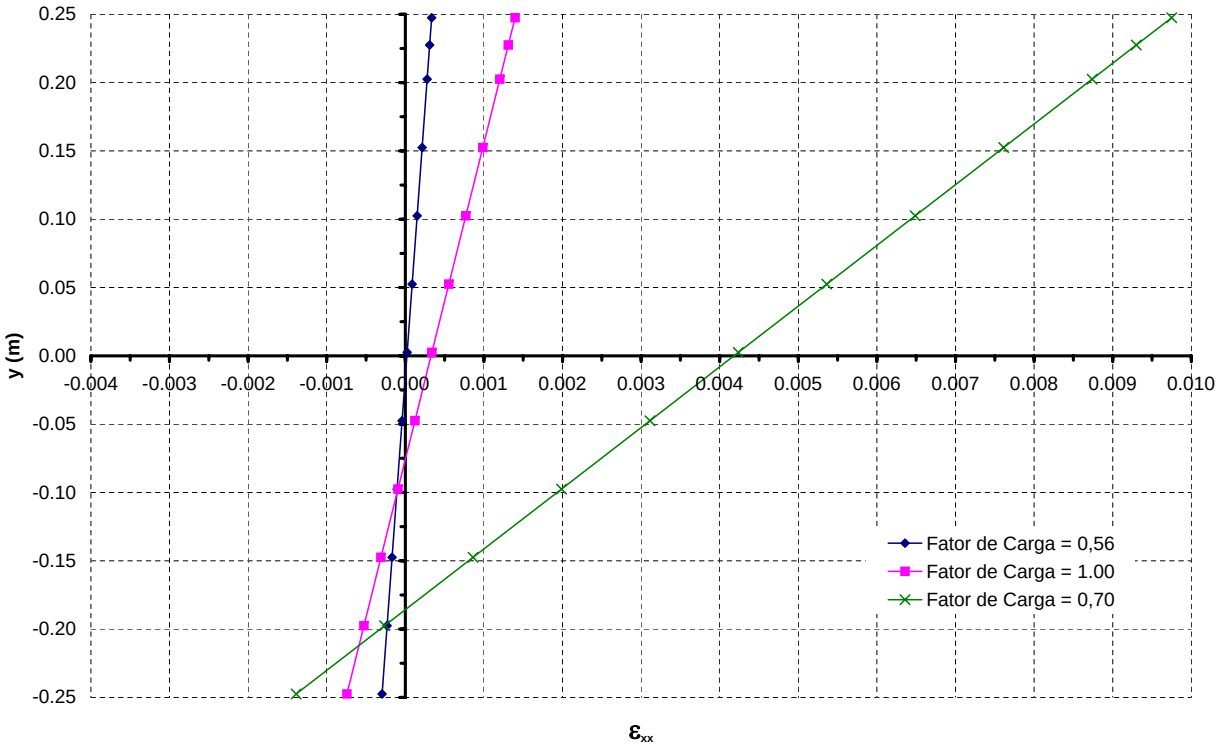


Figura 6.25: Deformações na seção transversal mais solicitada para A_s=1,5 cm^2.

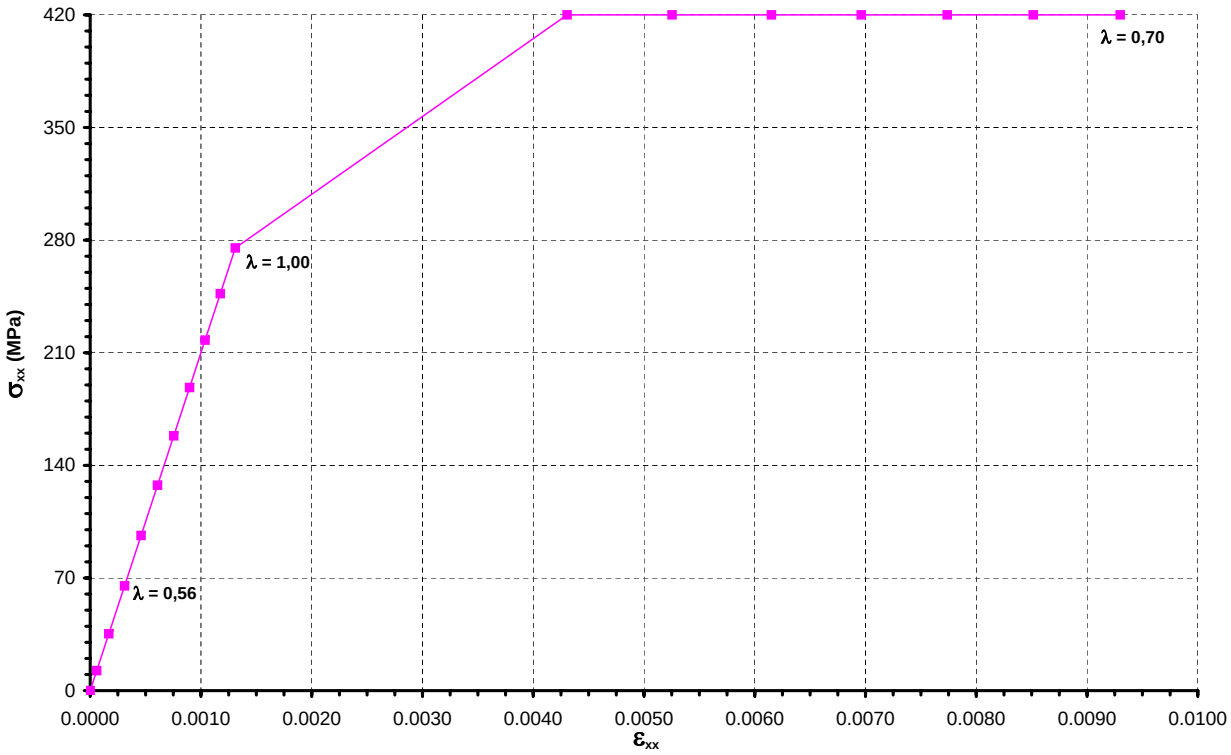


Figura 6.26: Histórico de tensões e deformações da armadura tracionada para A_s=1,5 cm^2.

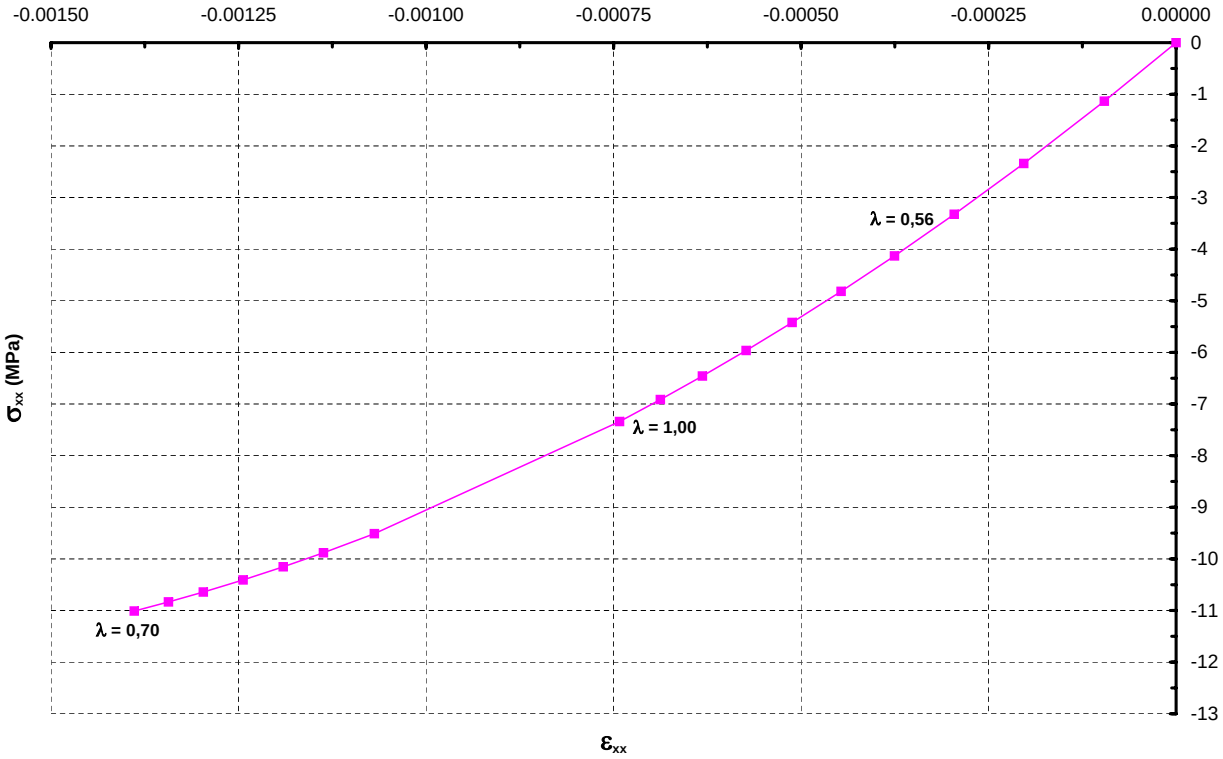


Figura 6.27: Histórico de tensões e deformações da fibra de concreto mais comprimida para A_s=1,5 cm^2.

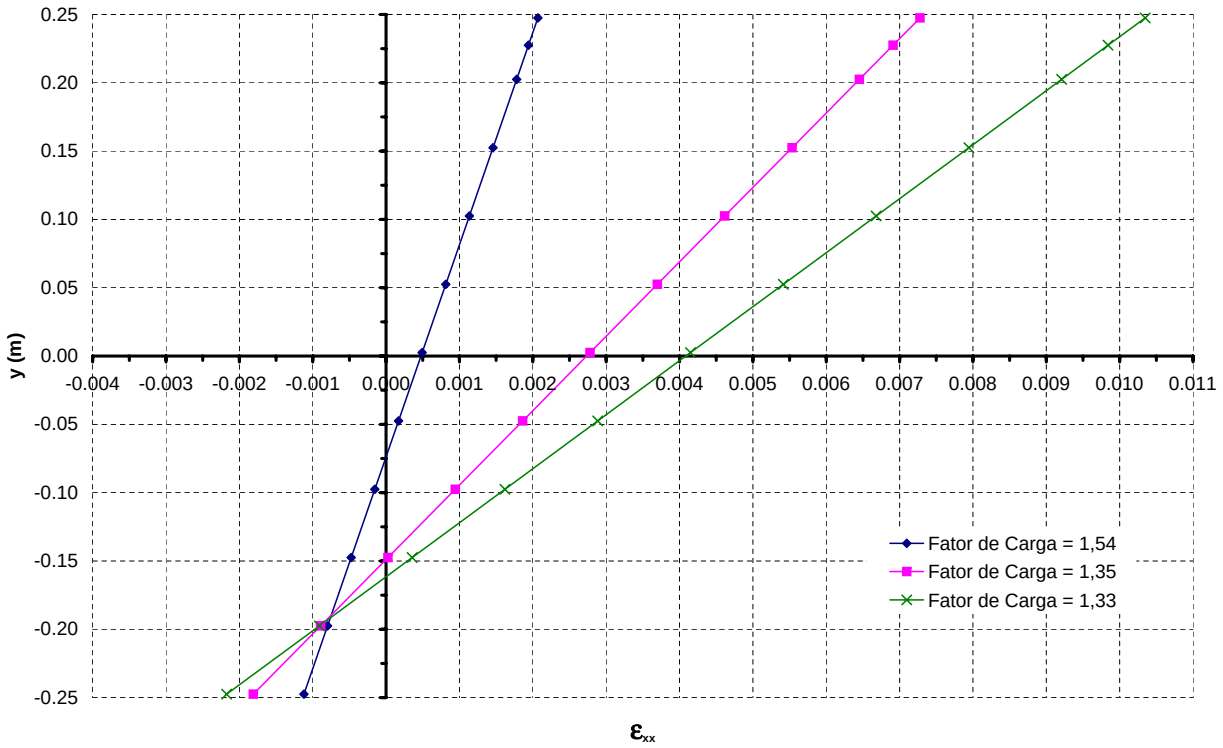


Figura 6.28: Deformações na seção transversal mais solicitada para A_s=3,0 cm^2.

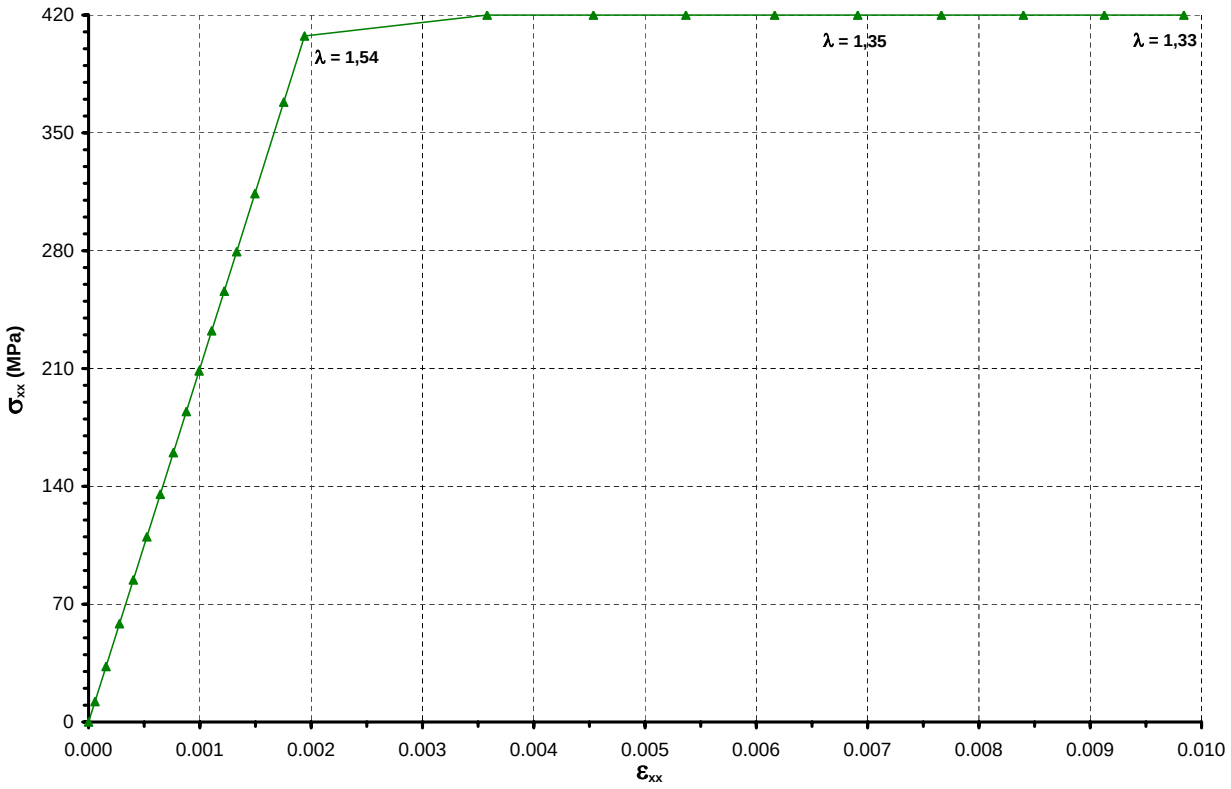


Figura 6.29: Histórico de tensões e deformações da armadura tracionada para A_s=3,0 cm^2.

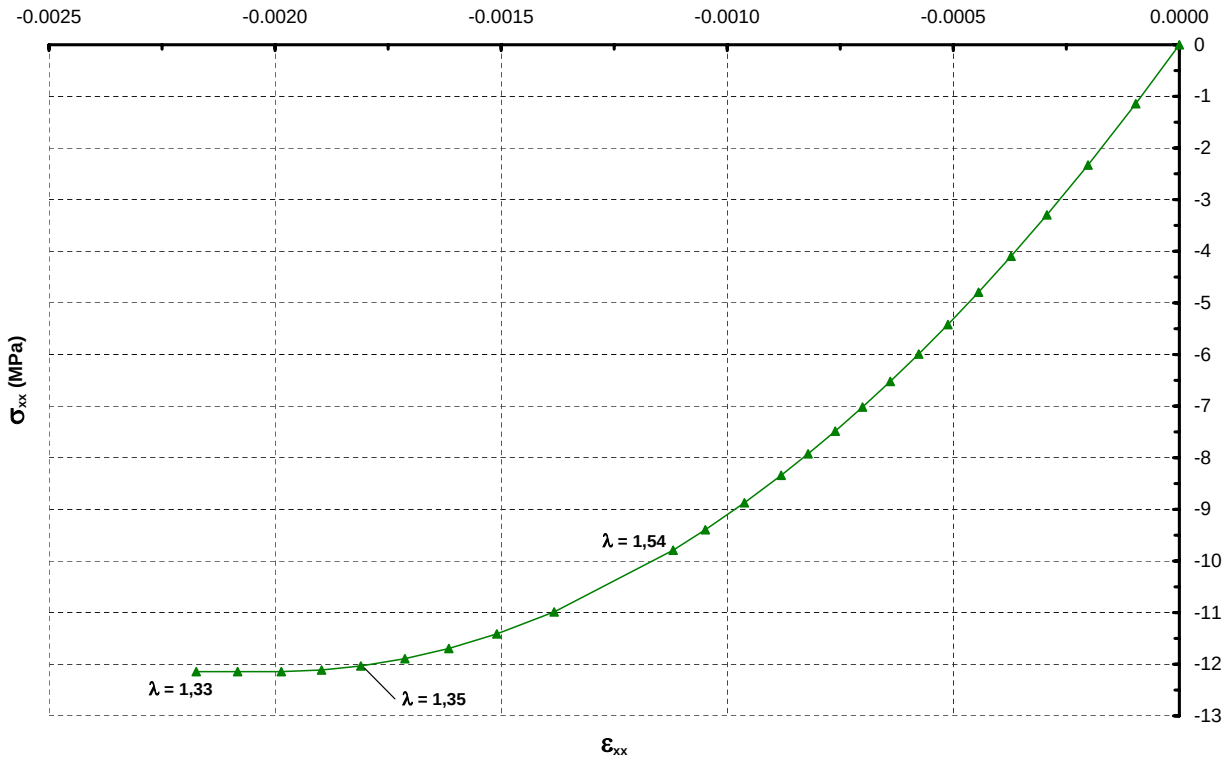


Figura 6.30: Histórico de tensões e deformações da fibra de concreto mais comprimida para A_s=3,0 cm^2.

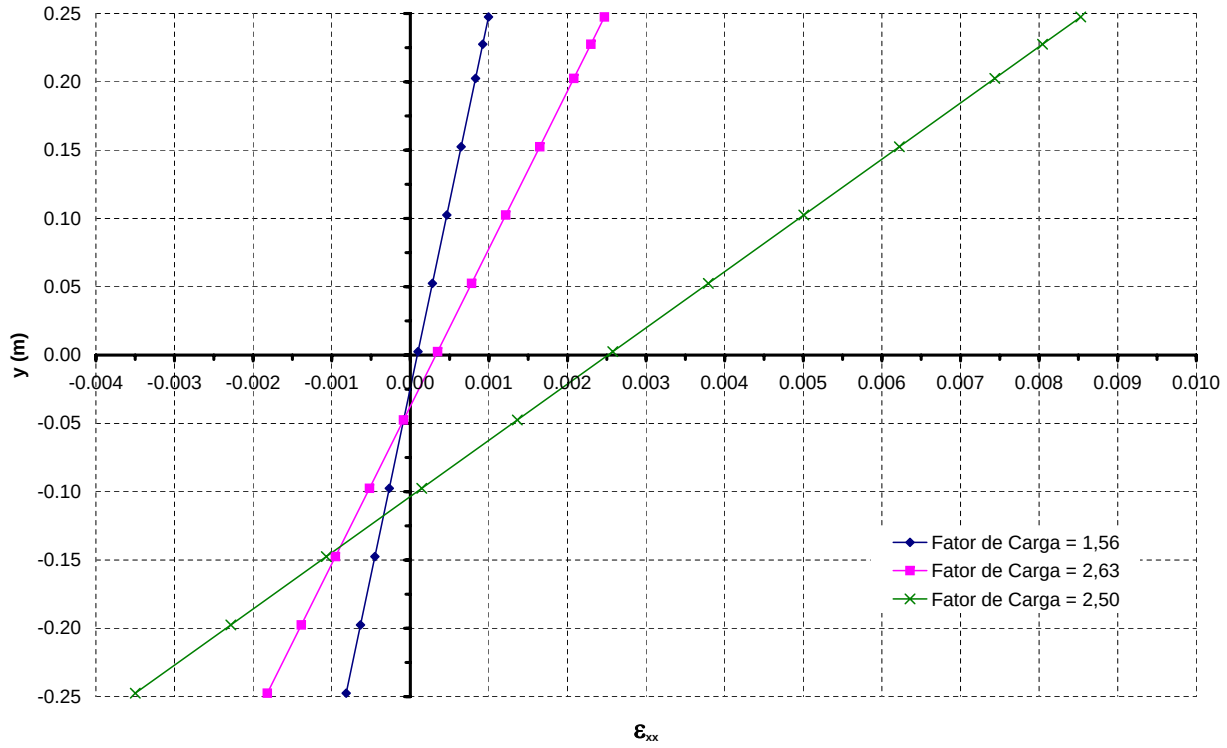


Figura 6.31: Deformações na seção transversal mais solicitada para A_s=6,0 cm^2.

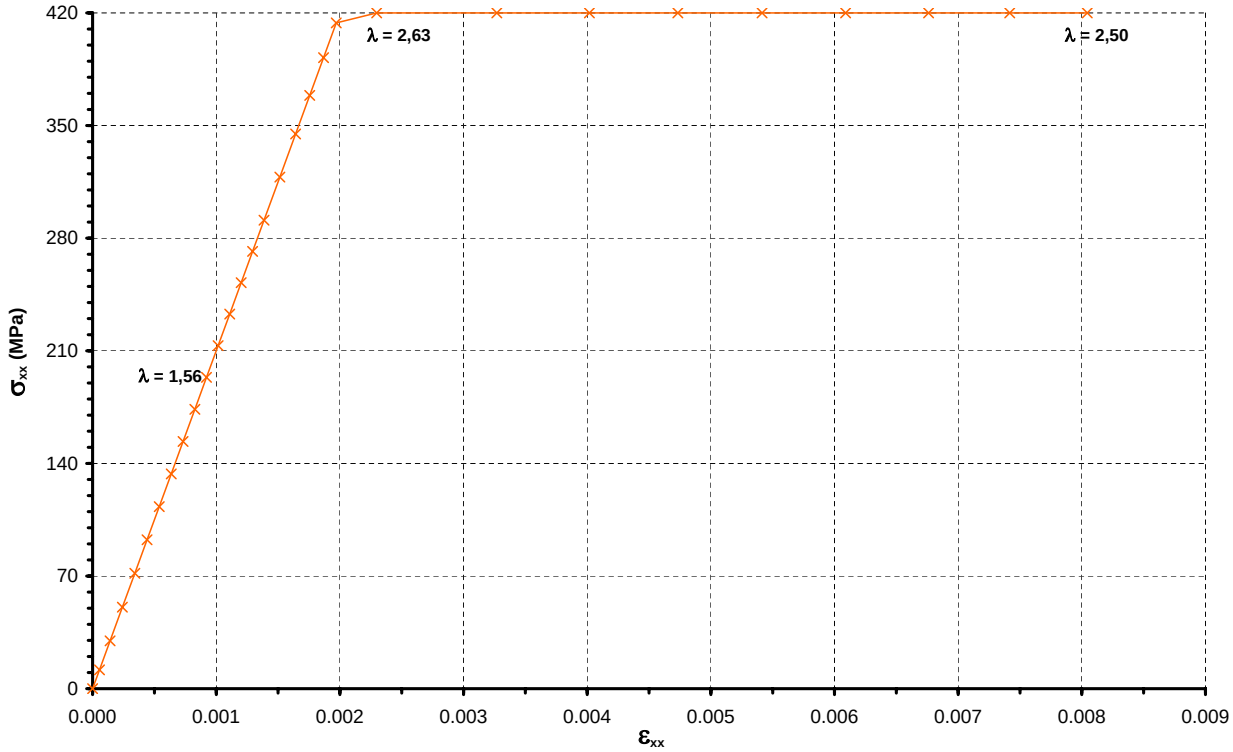


Figura 6.32: Histórico de tensões e deformações da armadura tracionada para A_s=6,0 cm^2.

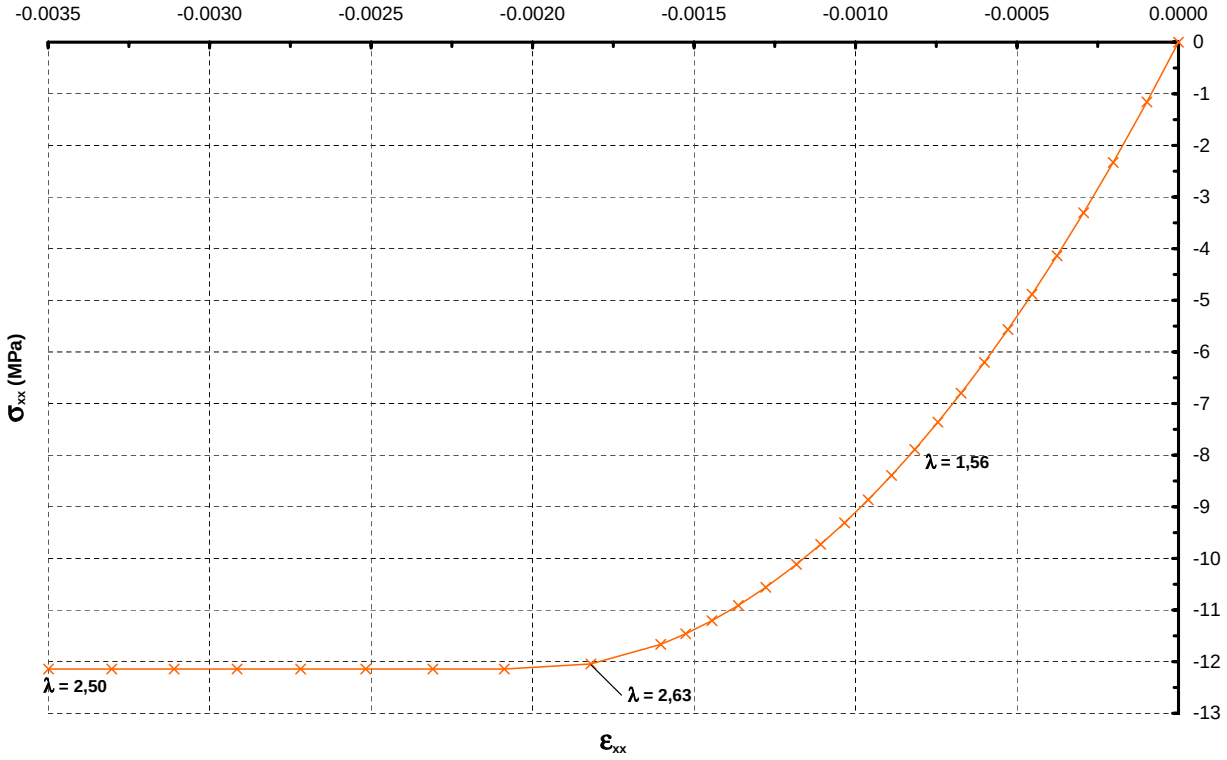


Figura 6.33: Histórico de tensões e deformações da fibra de concreto mais comprimida para A_s=6,0 cm^2.

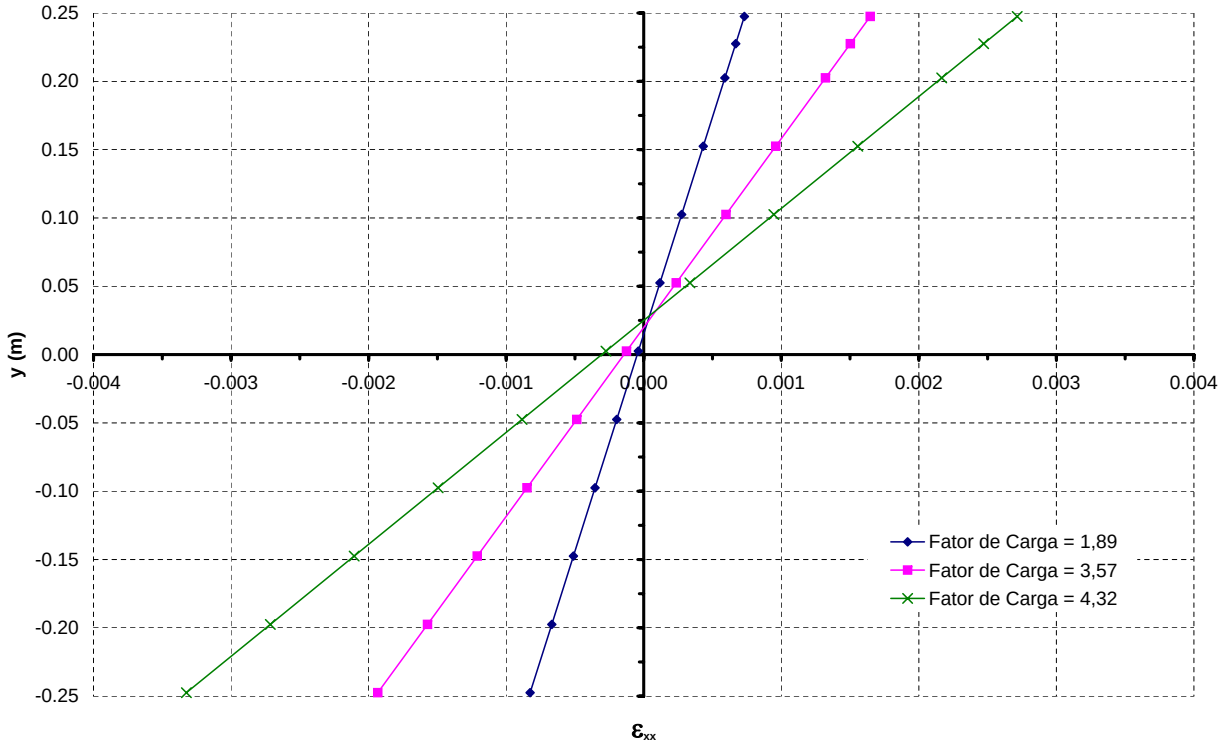


Figura 6.34: Deformações na seção transversal mais solicitada para A_s=12,0 cm^2.

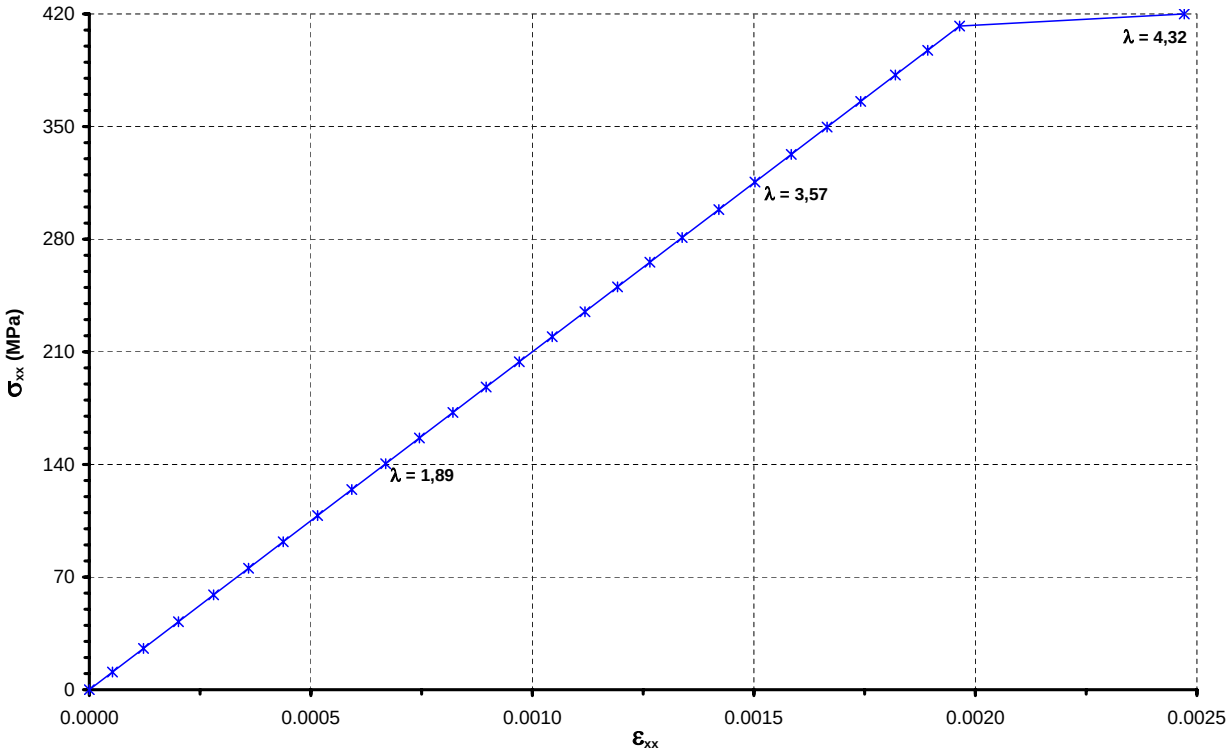


Figura 6.35: Histórico de tensões e deformações da armadura tracionada para A_s=12,0 cm^2.

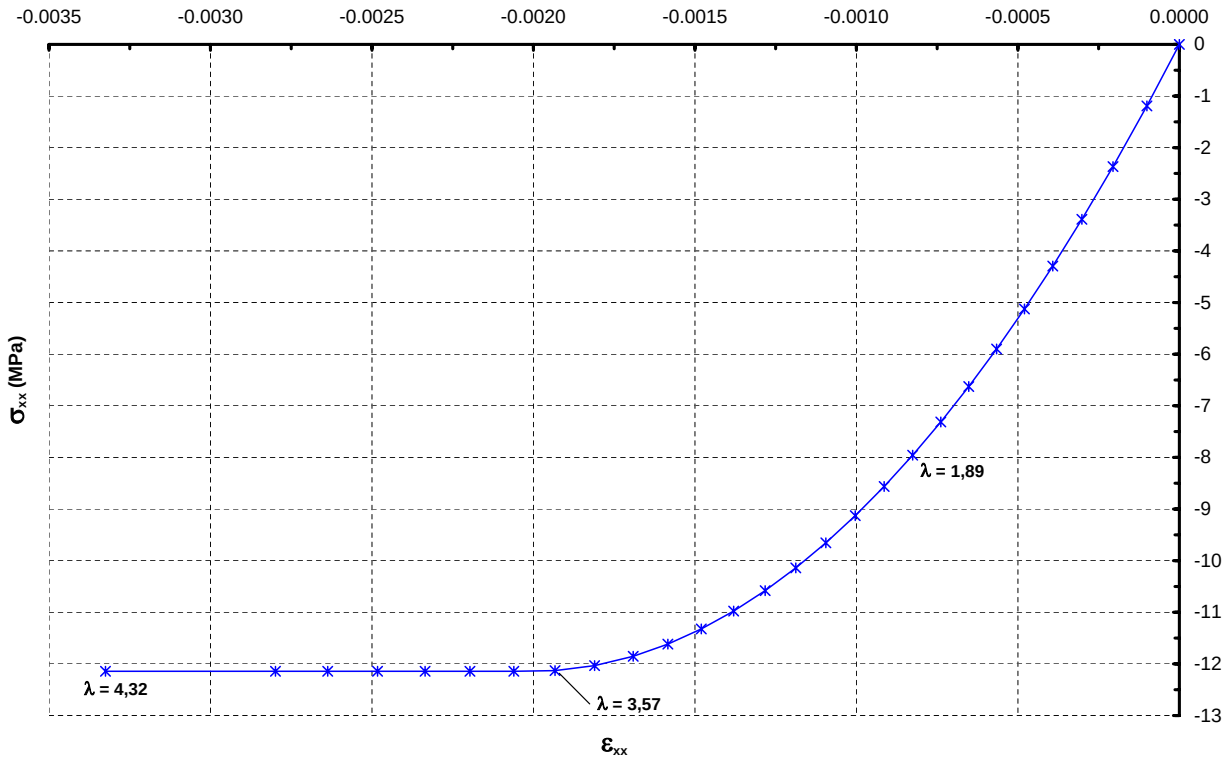


Figura 6.36: Histórico de tensões e deformações da fibra de concreto mais comprimida para A_s=12,0 cm^2.

6.3.5 Pórtico Espacial

Neste exemplo é modelado um pórtico espacial de concreto armado, cuja geometria e carregamento foram adaptados de Izzuddin et al. (2002). A Figura 6.37 mostra a configuração geométrica do pórtico, que possui as colunas armadas uniformemente e as vigas com armadura variável. Ele está sujeito a cargas horizontais incrementais $\mathbf{P}_x$ e $\mathbf{P}_y$, componentes de uma carga horizontal $\mathbf{P}$ que faz um ângulo de 45^o com o eixo global X. Está também sujeito a cargas verticais constantes, sendo a carga concentrada $\mathbf{Q}$ aplicada nos pilares igual à 200 kN e a carga distribuída $\mathbf{q}$ sobre as vigas igual à 10 kN/m.

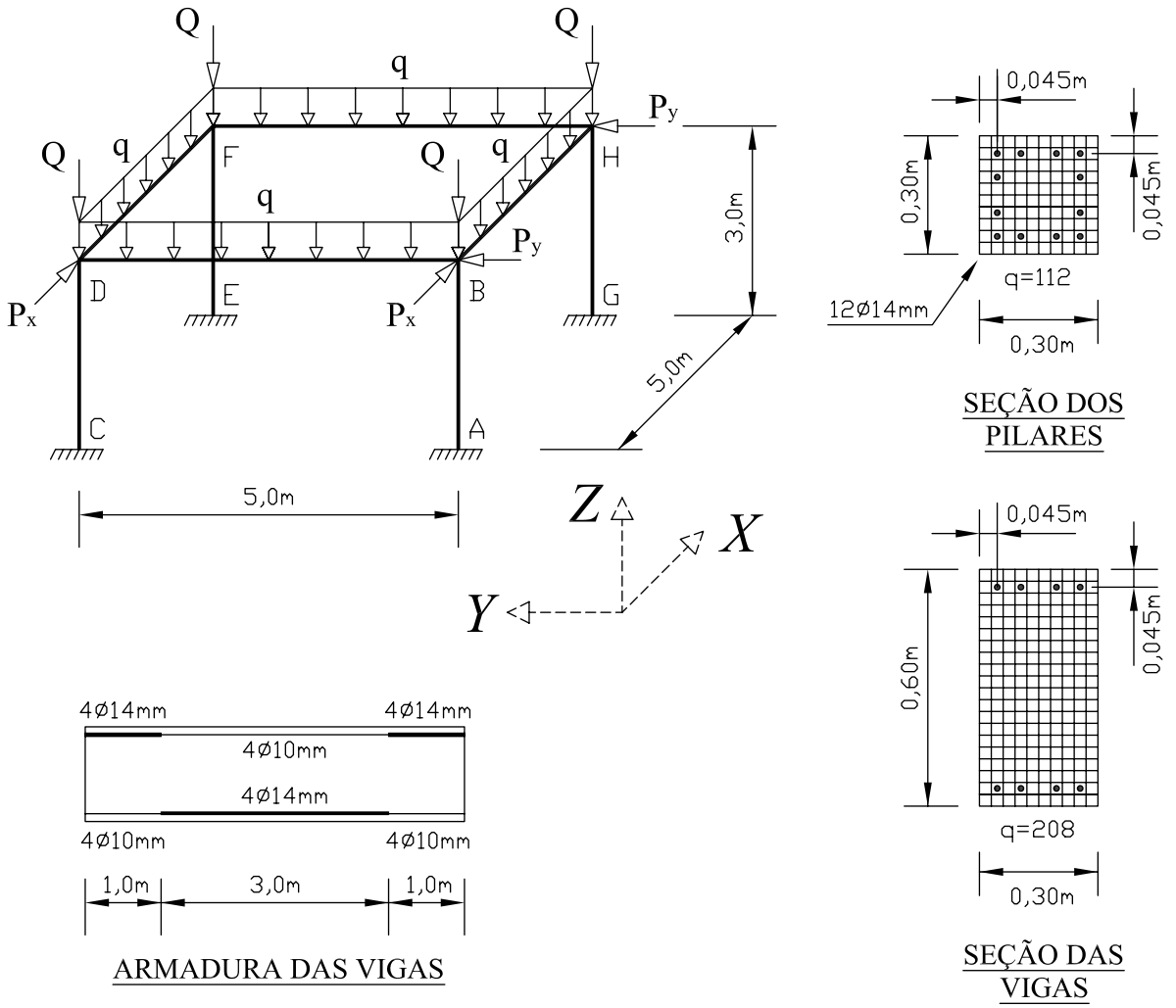
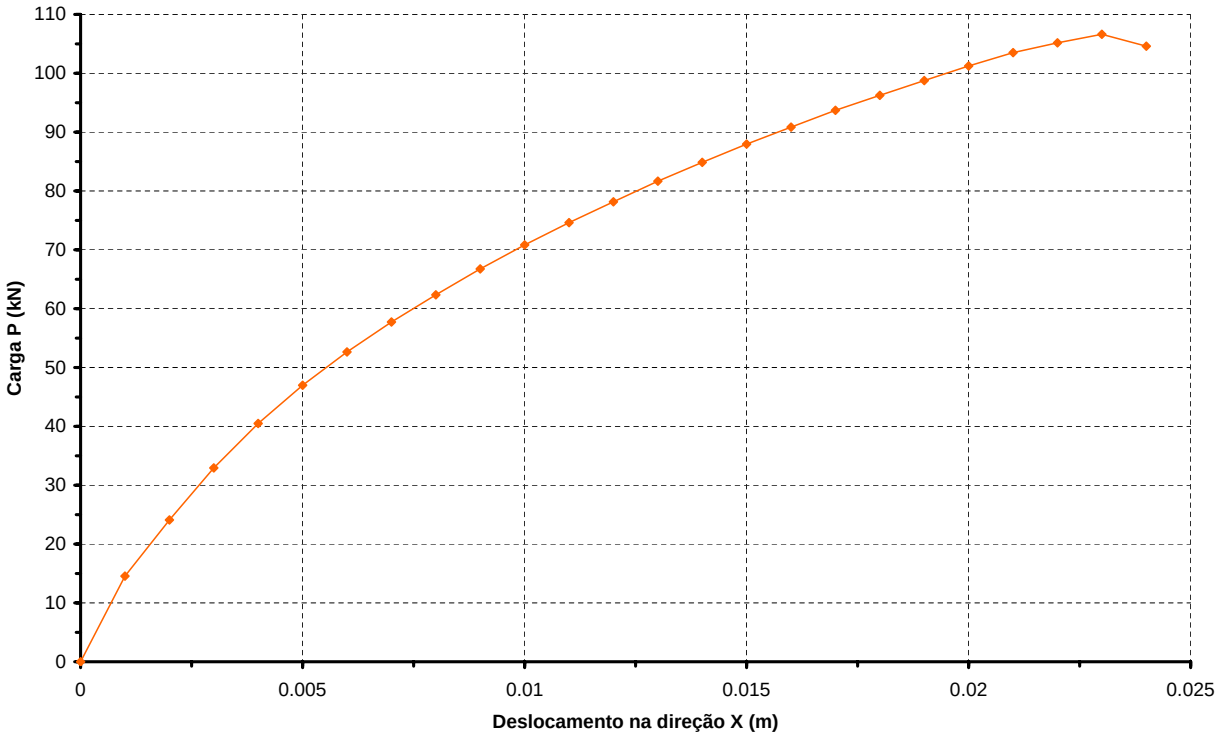
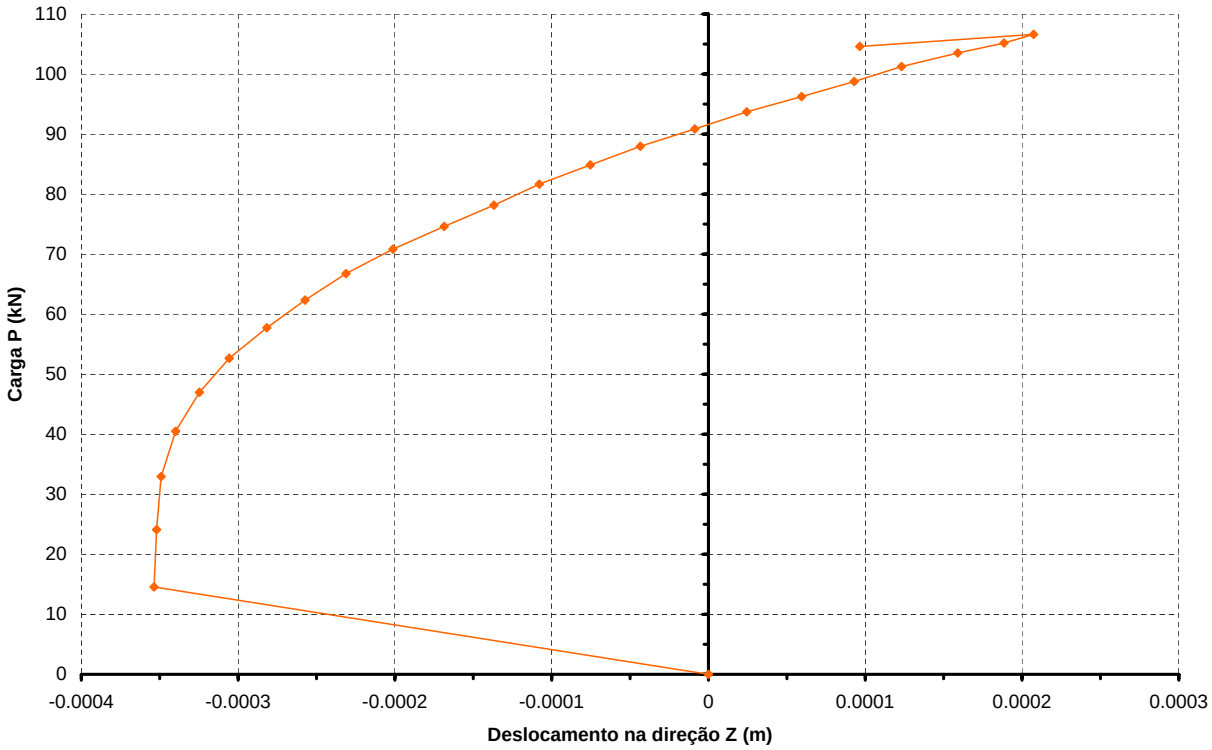


Figura 6.37: Pórtico espacial de concreto armado

O pórtico foi discretizado utilizando-se 10 elementos de 2 nós de Euler-Bernoulli por membro. A seção transversal das vigas foi discretizada em 208 áreas, sendo 8 delas representando



(a)



(b)

Figura 6.38: Trajetórias de equilíbrio do ponto B do pórtico espacial.

Quando **P** atingiu o valor de 105kN, a fibra de concreto mais comprimida ultrapassou seu limite último de deformação, sofrendo ruptura. Como se trata de uma fibra de área pequena em relação à seção, sua perda de resistência não afetou o equilíbrio da seção, sendo um indício do processo de esmagamento do concreto, uma vez que já se alcançou a tensão limite de compressão até a fibra de concreto em X'=10,6cm, como mostra a Figura 6.43. Para a mesma carga **P**, houve ainda o início da deformação plástica por compressão na armadura em X'=14,8cm e por tração na armadura em X'=-10,6cm, como pode ser visto nas Figuras 6.45 e 6.46.

Após a carga máxima, dá-se então o esmagamento do concreto acompanhado pelo escoamento das armaduras de aço mais solicitadas, levando à ruptura da seção mais solicitada do pilar AB. O mecanismo de ruptura foi característico do domínio de deformação *3* da NBR6118, sendo a seção do pilar sub-armada.

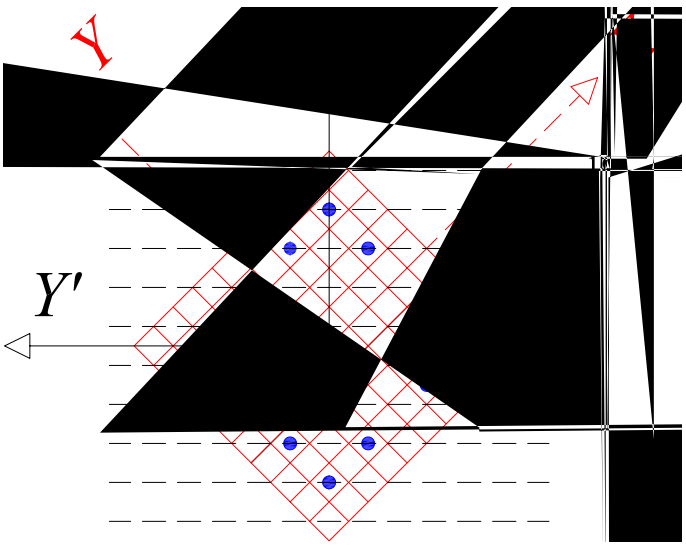


Figura 6.39: Seção da base do pilar AB e coordenadas dos pontos monitorados ao longo do eixo X'.

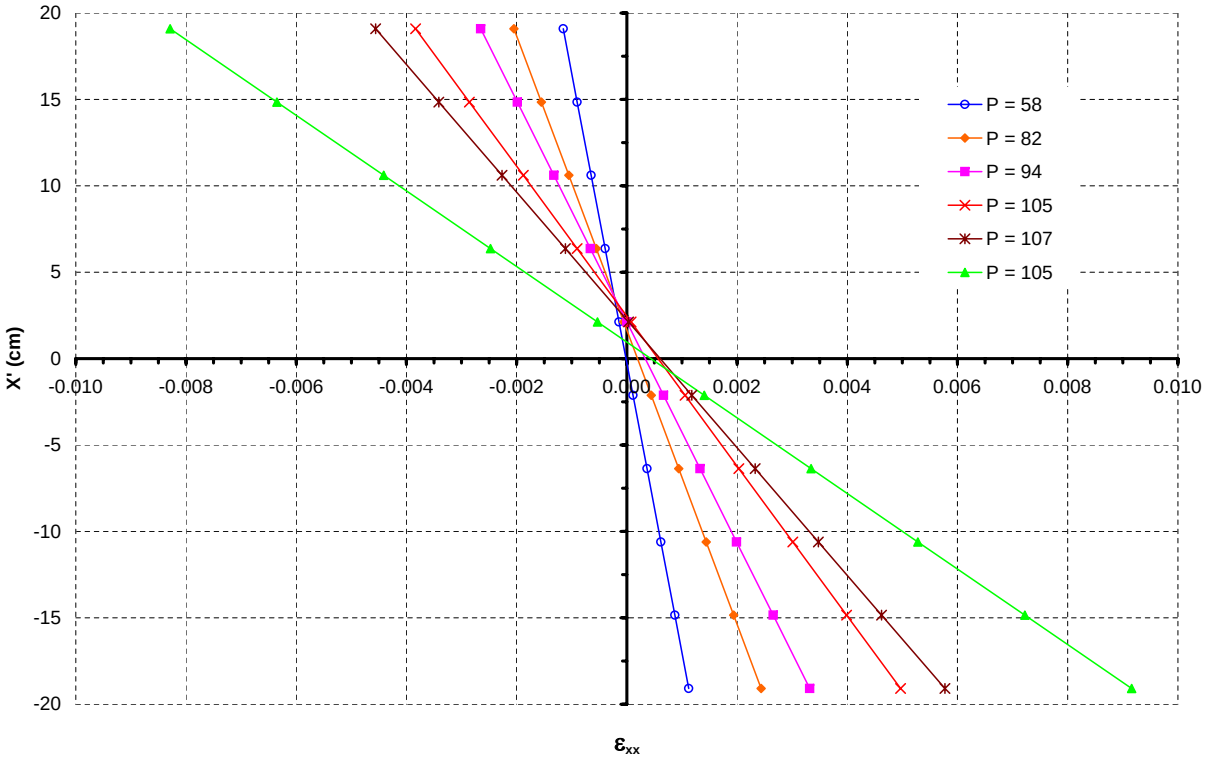


Figura 6.40: Deformações na seção transversal da base do pilar AB.

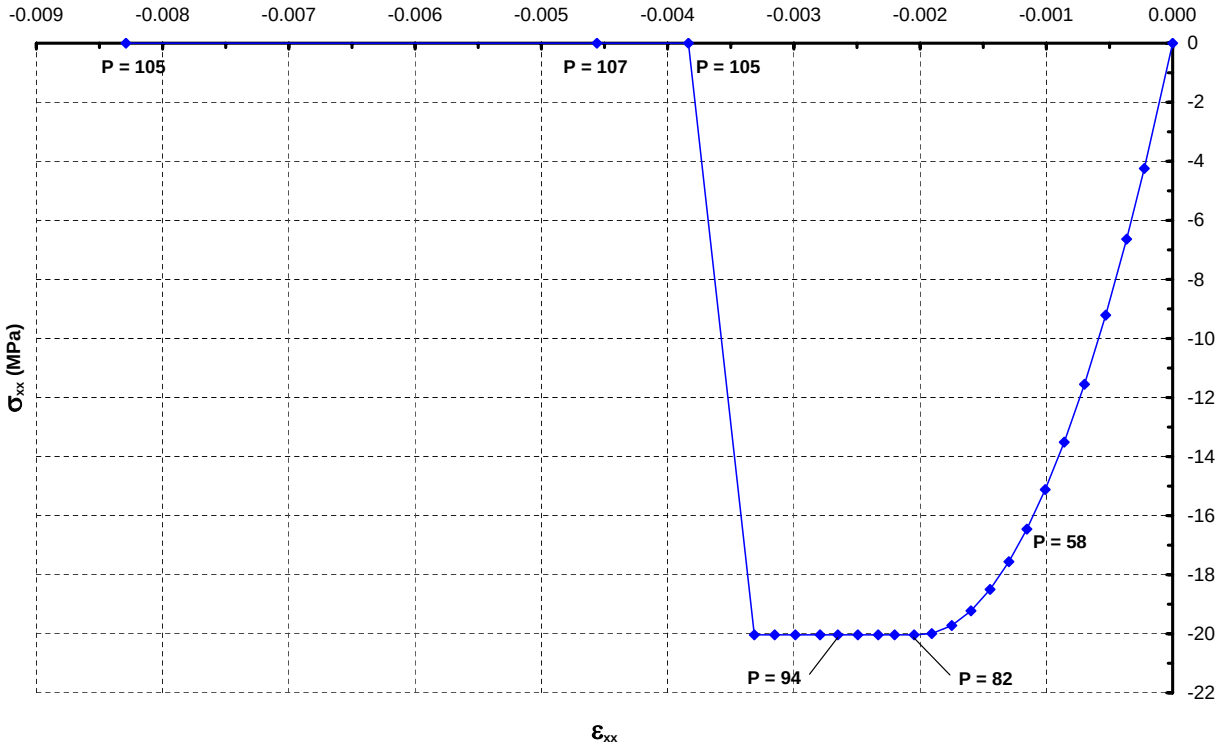


Figura 6.41: Histórico de tensões e deformações da fibra de concreto mais comprimida ($X´ = 19{,}1cm$).

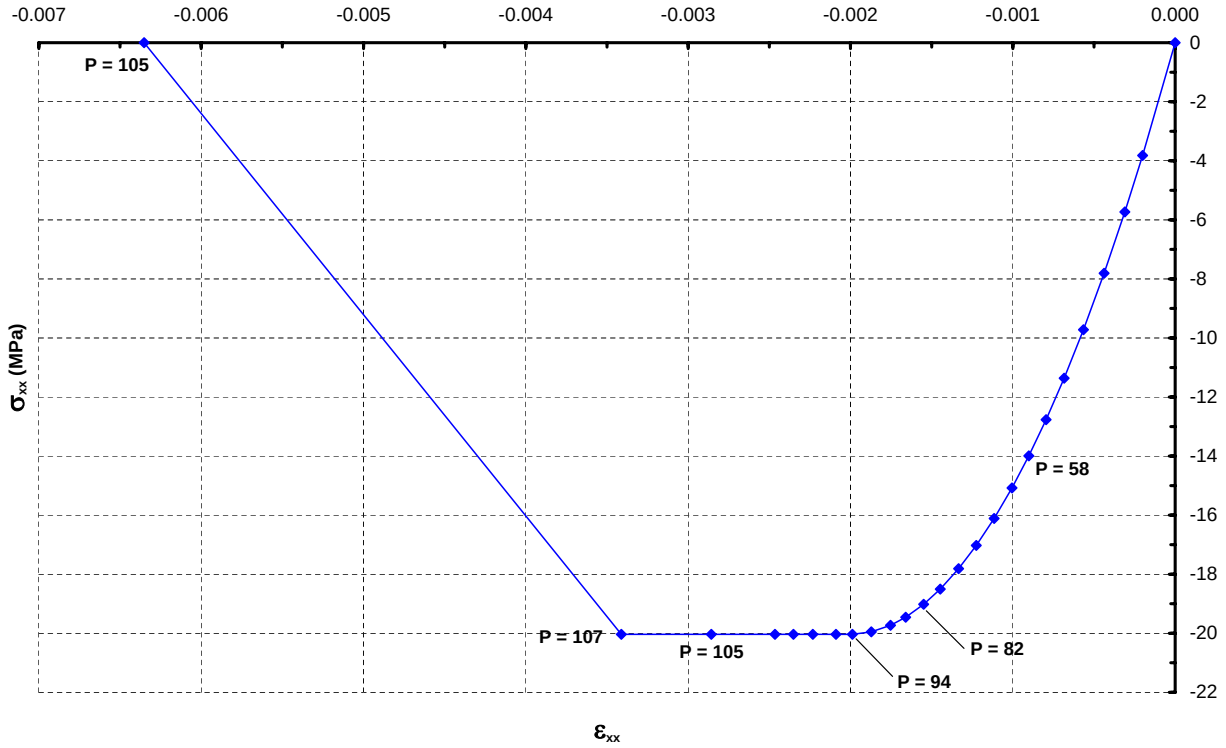


Figura 6.42: Histórico de tensões e deformações da fibra de concreto em $X^{'} = 14{,}8cm$.

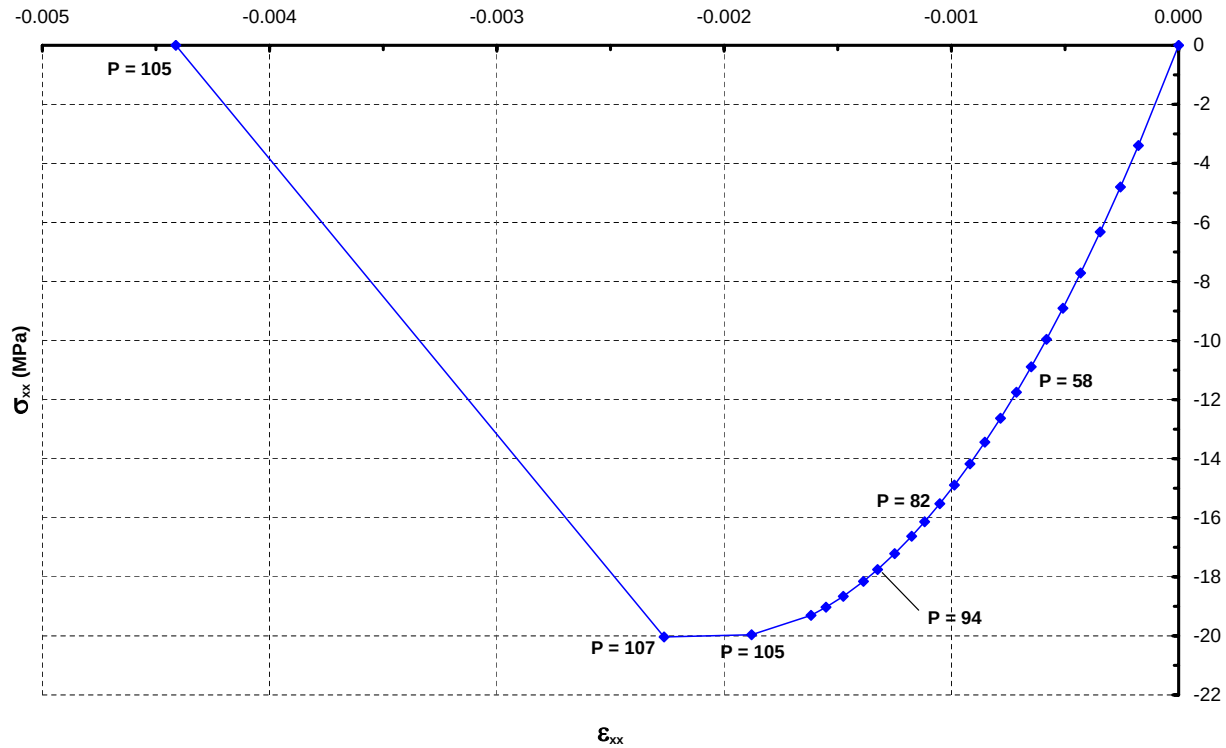


Figura 6.43: Histórico de tensões e deformações da fibra de concreto em $X^{'} = 10{,}6cm$.

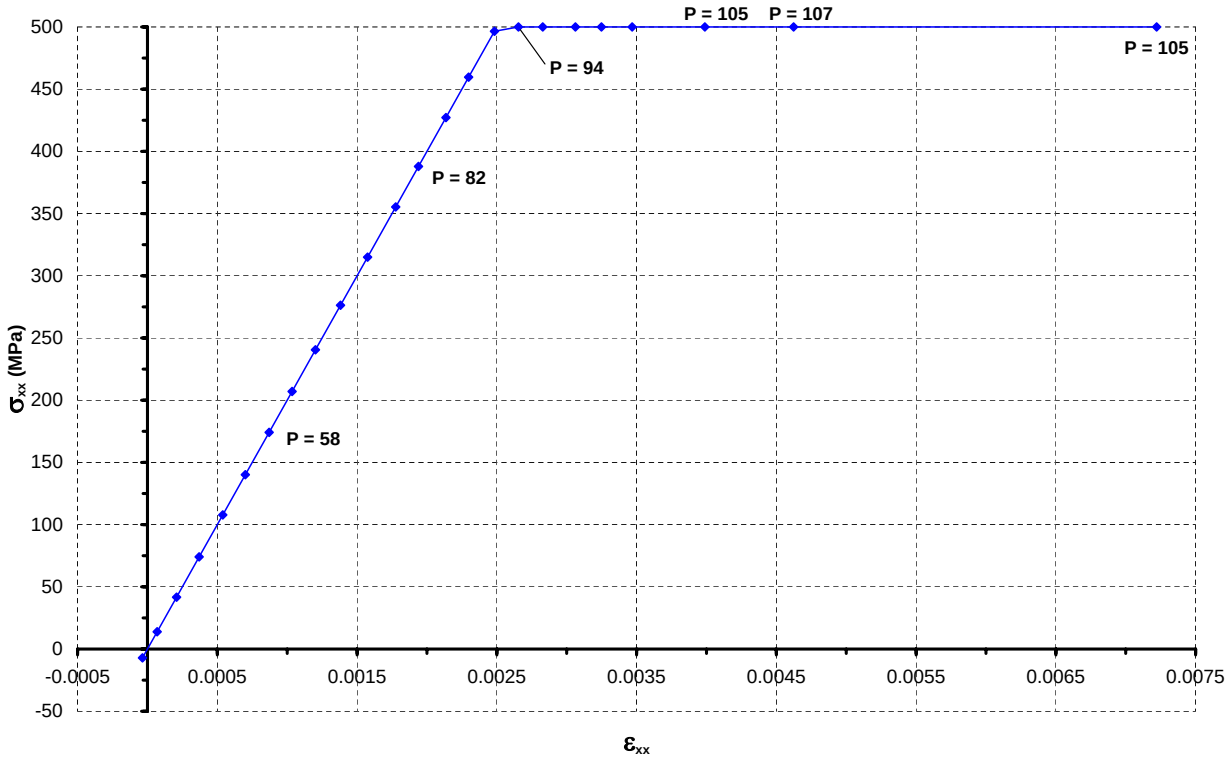


Figura 6.44: Histórico de tensões e deformações da armadura mais tracionada ($X´ = -14{,}8cm$).

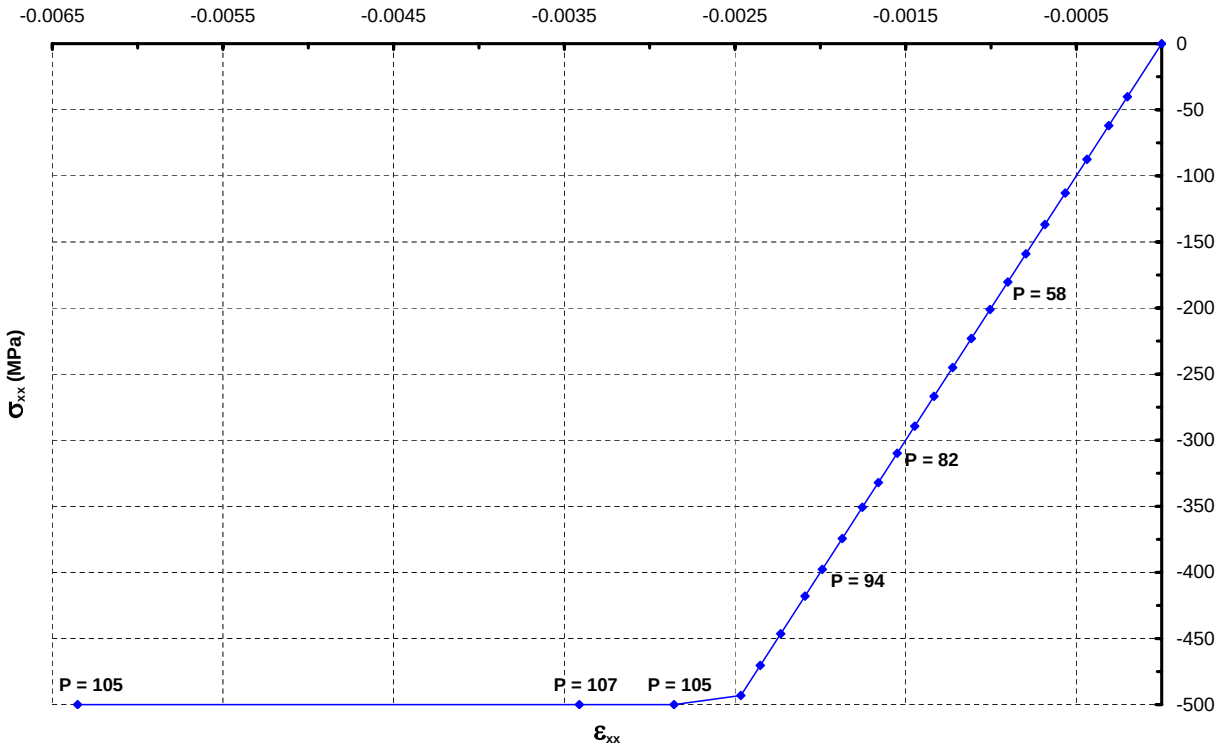


Figura 6.45: Histórico de tensões e deformações da armadura mais comprimida ($X´ = 14{,}8cm$).

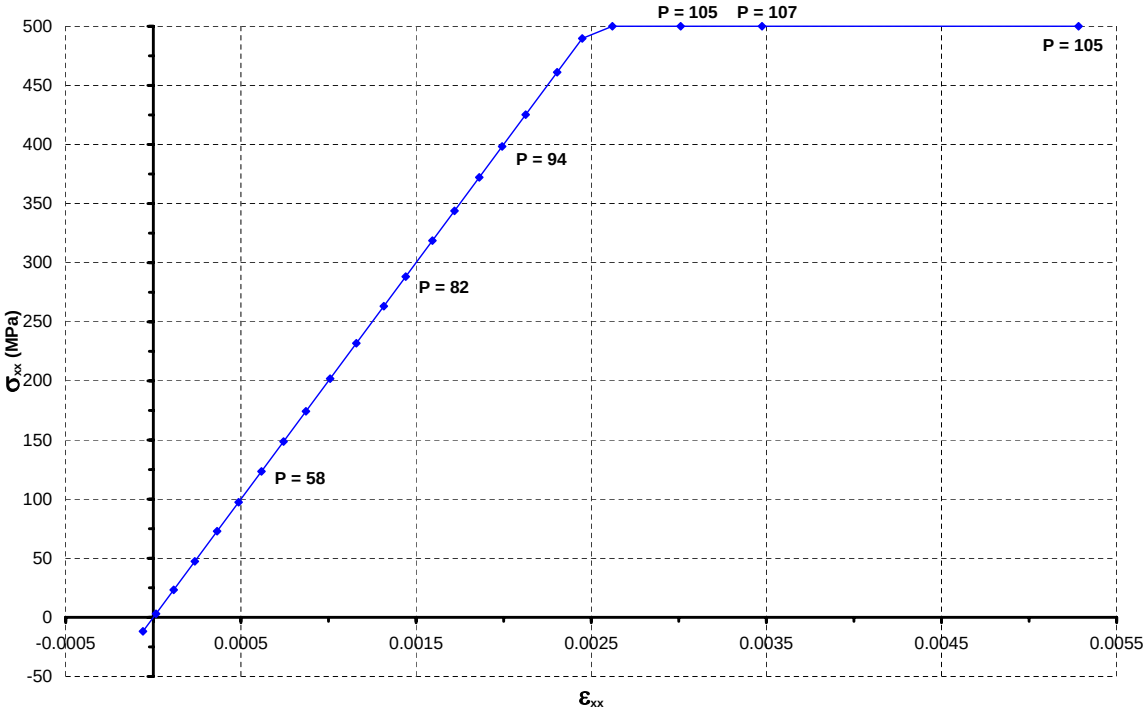


Figura 6.46: Histórico de tensões e deformações da armadura em $X' = -10{,}6cm$.

Capítulo 7

CONSIDERAÇÕES FINAIS

A partir dos recursos existentes no programa **INSANE**, o núcleo numérico do sistema foi fatorado e ampliado de forma progressiva através da implementação da resolução de problemas fisicamente não-lineares de modelos estruturais reticulados com seções transversais de qualquer geometria e compostas por vários materiais.

Para a resolução de problemas desta natureza foram apresentados os modelos matemáticos e discretos de pórtico espacial, baseados nas teorias de flexão de Timoshenko e Euler-Bernoulli, nos quais as seções foram decompostas em áreas menores. A análise fisicamente não-linear de estruturas foi discutida brevemente, apresentando exemplos de leis constitutivas não-lineares e o processo incremental-iterativo para solução de equações não-lineares de equilíbrio.

As simulações numéricas apresentadas validaram a implementação para diversos modelos estruturais reticulados, além de mostrar sua capacidade de representar seções genéricas. Comprovou-se ainda a eficiência da implementação na descrição das trajetórias de equilíbrio dos modelos e foram ilustrados os diversos recursos implementados, ressaltando-se as limitações dos mesmos.

A formulação de elementos finitos paramétricos unidimensionais para pórtico espacial apresentada, embora pareça ser conceitualmente mais simples, exige maior capacidade de abstração ao ser implementada, uma vez que se deseja, no projeto **INSANE**, uma implementação genérica, adaptável à resolução de qualquer modelo estrutural baseado no Método dos Elementos Finitos.

A generalidade da implementação foi obtida através da organização dos componentes do

núcleo numérico segundo abstrações identificadas nas diversas etapas de uma resolução numérica de modelos discretos e da definição de forma clara da interação entre estes componentes. O núcleo numérico ganhou em modularidade, tornando o software ainda mais reutilizável, uma vez que os módulos foram projetados para resolver problemas genéricos de análise estrutural através do Método dos Elementos Finitos.

A reutilização do código, além de economizar esforço de programação e diminuir possibilidades de erros, confere ao sistema a capacidade de evolução em resposta a novos e mais complexos problemas. Portanto, a partir de conceitos já consolidados, complexidades podem ser ampliadas sem ter que recomeçar todo o processo a cada novo aperfeiçoamento.

Este trabalho procurou, acima de tudo, conferir ao núcleo numérico do **INSANE** a capacidade de evoluir com o aprimoramento progressivo dos modelos de análise estrutural baseados no Método dos Elementos Finitos. Aplicações que auxiliem as pesquisas na área de métodos numéricos e computacionais podem ser desenvolvidas em um sistema computacional bem segmentado, reutilizável e adaptável. A habilidade de modificar e estender de forma amigável o sistema **INSANE** é imprescindível para mantê-lo atualizado com as tecnologias de elementos finitos e de sistemas computacionais, que têm se desenvolvido cada vez mais rápido.

7.1 Desenvolvimento Colaborativo

O projeto **INSANE** é desenvolvido de forma colaborativa, fazendo uso dos diversos conceitos da programação orientada a objetos e contando com o auxílio dos recursos emergentes de desenvolvimento de software. O projeto envolve diversos alunos em diferentes estágios de conhecimento, que desenvolvem em colaboração trabalhos em uma das três grandes aplicações do sistema (pré-processador, processador e pós-processador).

Os trabalhos desenvolvidos no processador, ou núcleo numérico, por Almeida (2005), Pitangueira et al. (2004) e Germanio (2005), já concluídos, necessitam ter sua implementação adequada à nova modulação proveniente da fatoração progressiva desenvolvida no trabalho aqui apresentado. Também deve ser adequada a implementação, também já concluída, da aplicação de pré-processamento desenvolvida por Gonçalves (2004).

Relacionados ao núcleo numérico, encontram-se em andamento os seguintes trabalhos: implementação do modelo constitutivo de microplanos baseados no contínuo de Cosserat (Fuina, 2006), implementação de elementos finitos de placas (Saliba, 2006) e a implementação do método dos elementos finitos estendido (XFEM) (Wolff, 2006).

A aplicação de pós-processamento encontra-se em desenvolvimento para a análise não-linear de elementos finitos unidimensionais e planos (Penna, 2006). O projeto **INSANE** conta ainda com um recurso de processamento interativo para ensino do método dos elementos finitos (Moreira, 2006), e com o desenvolvimento de um serviço *web* para o método dos elementos finitos, no qual o núcleo numérico é executado remotamente.

7.2 Sugestões para Trabalhos Futuros

Como sugestões para trabalhos futuros no núcleo numérico, propõe-se a implementação da solução de análises geometricamente não-lineares, a implementação de elementos finitos de casca e a implementação de outros tipos de modelos constitutivos, como o modelo constitutivo de fissuras distribuídas.

Convém ressaltar algumas facilidades para implementação no núcleo numérico dos trabalhos citados. A implementação de novos modelos constitutivos é realizada com facilidade através da particularização da interface `ConstitutiveModel`, discutida na Seção 5.4.2, assim como a implementação de novos elementos finitos é realizada através da particularização da classe `Element`, discutida na Seção 5.4. Novos tipos de soluções podem ser implementados com a especialização da interface `Solution` e da interface `ProblemDriver`, discutidas nas Seções 5.3 e 5.4, respectivamente. Já a implementação do método dos elementos finitos estendido envolve a alteração em tempo de execução de atributos do elemento finito, como a função de forma e tipo de problema, já implementados nas interfaces `Shape` e `ProblemDriver`, mostradas na Seção 5.4.

Sugere-se ainda, como trabalho futuro, o desenvolvimento de uma aplicação de pré e pós-processamento para modelos tridimensionais.

Apêndice A

Tecnologias de Desenvolvimento de Software

Para obter uma maior produtividade e velocidade no processo de fatoração e ampliação do núcleo numérico do **INSANE**, foi utilizado o IDE (*Integrated Development Environment*) ***Eclipse*** (`http://www.eclipse.org/`), ferramenta de desenvolvimento Java de código livre mais utilizada no mundo. Para agilizar este processo, o *Eclipse* oferece editor de código-fonte, gerador de código, compilador, debugador e montador, dentre outros inúmeros recursos, como os diversos plug-ins de código livre disponíveis gratuitamente na internet, os quais realizam várias funções adicionais como testes unitários, controle de versões e diagramas UML.

Juntamente ao *Eclipse*, foi utilizado o ***Apache Maven*** (`http://maven.apache.org/`), para automação e gerenciamento dos vários módulos do **INSANE**. Ele é similar à ferramenta *Ant* (`http://ant.apache.org/`), mas possui um modelo de configuração mais simples, baseado no formato XML. O *Maven* utiliza uma construção conhecida como *Project Object Model* para descrever o projeto de software em construção, suas dependências de outros módulos e componentes e a sua seqüência de construção. Ele contém tarefas pré-definidas que realizam funções bem conhecidas, como compilação, empacotamento de código e execução de testes unitários, automatizando o processo de montagem dos vários módulos em um sistema único e uniforme.

Com o objetivo de garantir a qualidade e validar a implementação do programa, foram realizados testes unitários através do framework ***JUnit*** (`http://www.junit.org/`) na forma

de plug-in no *Eclipse*. Ele oferece a possibilidade de testar o código antes de utilizá-lo efetivamente, realizando testes automatizados das aplicações em suas menores unidades. Isto facilita a correção de eventuais erros durante a implementação de métodos e objetos. Os testes unitários são executados continuamente com auxílio do *Maven*, garantindo a estabilidade e a confiabilidade do sistema e de alterações em sua implementação.

Para o controle de versões do sistema utilizou-se o *Concurrent Versions System* (**CVS**), que é uma ferramenta de apoio ao desenvolvimento colaborativo de software cuja principal função é identificar e controlar sistematicamente as modificações realizadas nos arquivos de um projeto ao longo do tempo. Através de um mecanismo automatizado, ele gerencia a evolução das mudanças e garante a integridade e rastreabilidade das modificações do sistema. O CVS (`http://www.cvshome.org/`) é visto como uma extensão natural do processo de desenvolvimento, permitindo a realização de modificações paralelas de forma coerente e padronizada, especialmente em se tratando de equipes geograficamente dispersas (Caetano, 2004).

Como ferramenta para visualizar a comunicação e relações entre os objetos das aplicações foi adotada a proposta da *Unified Modelling Language* (**UML**), linguagem padronizada para a modelagem de sistemas de software orientados a objetos. Dentre as diversas linguagens gráficas disponíveis, ela é a mais sistematicamente elaborada, e também a mais aceita. As representações gráficas das hierarquias de classes deste trabalho foram confeccionadas com o auxílio do plug-in ***EclipseUML Free Edition*** (`http://www.omondo.com/`), segundo os padrões de diagramas UML.

A documentação do código do sistema **INSANE** encontra-se disponível no site do projeto: `http://www.dees.ufmg.br/insane`.

Apêndice B

Formato do Arquivo XML para Persistência

Conforme discutido na Seção 5.5, a persistência de dados mais utilizada entre as aplicações do **INSANE** é a realizada por meio da linguagem padronizada de arquivos XML. É apresentado a seguir um exemplo da estrutura XML adotada para representar os dados referentes a um modelo paramétrico de elementos finitos.

O exemplo apresentado refere-se ao pilar de concreto armado com seção circular sujeito à compressão centrada, considerando a lei constitutiva do concreto proposta por Carreira e Chu (1985), sendo f_c=83MPa, E_0=46060MPa e ε_c=0,002, apresentado na Seção 6.3.1. O formato do arquivo XML é mostrado nas Figuras B.1 e B.2. É possível visualizar este mesmo arquivo em um formato resumido como mostra a Figura B.3.

Após a resolução do problema, o processador cria um novo arquivo XML para cada passo da análise não-linear, contendo todos os resultados obtidos. O arquivo para o primeiro passo da análise é mostrado na Figura B.4.

```
- <Insane>
  - <OutPut>
    - <OutPutNode label="2">
        <OutPutDisplacement>dx</OutPutDisplacement>
      </OutPutNode>
    </OutPut>
  - <Solution class="EquilibriumPath">
      <NumMaxSteps>120</NumMaxSteps>
    - <Step class="StandardNewtonRapson">
        <NumMaxIterations>100</NumMaxIterations>
        <Tolerance>0.0001</Tolerance>
        <ConvergenceType>1</ConvergenceType>
      </Step>
    - <IteractiveStrategyList>
      - <IteractiveStrategy class="DisplacementControl" LoadFactor="-0.00002">
          <NodeControl>2</NodeControl>
          <DirectionControl>1</DirectionControl>
        </IteractiveStrategy>
      </IteractiveStrategyList>
      <ChangeLoadFactor>-0.00001</ChangeLoadFactor>
      <ChangeTolerance>0.0001</ChangeTolerance>
      <ChangeNumMaxIterations>100</ChangeNumMaxIterations>
    </Solution>
  - <Model class="FemModel">
      <ProblemDriver>ParametricFisicallyNonLinearSolidMech</ProblemDriver>
    - <MaterialList>
      - <Material class="ConcreteCarreira" label="Concrete">
          <Fc>83000</Fc>
          <Ft>8300</Ft>
          <ec>0.002</ec>
          <et>0.0002</et>
          <E0>46060000</E0>
        </Material>
      - <Material class="LinearPlasticIsotropic" label="Steel">
          <Elasticity>200000000</Elasticity>
          <Poisson>0.3</Poisson>
          <ShearModulus>20000000</ShearModulus>
          <Yielding>450000</Yielding>
        </Material>
      </MaterialList>
    - <CrossSectionList>
      - <CrossSection label="section1">
        - <CrossSectionPoint SectionCoord="4.477E-02 1.081E-01" label="1">
            <Area>1.470E-03 1.000E00</Area>
            <PointMaterial>Concrete</PointMaterial>
            <PointConstitutiveModel
                                                    <
                                                К CrossSectionPoint
```

Figura B.1: Formato do arquivo XML de entrada de dados (1^a parte).

```
        − <CrossSectionPoint SectionCoord="-8.273E-02 8.273E-02" label="32">
            <Area>2.000E-04 1.000E00</Area>
            <PointMaterial>Steel</PointMaterial>
            <PointConstitutiveModel>OnePointConstModel</PointConstitutiveModel>
            <PointAnalysisModel>EulerPoint</PointAnalysisModel>
          </CrossSectionPoint>
        </CrossSection>
      </CrossSectionList>
    − <NodeList>
      − <Node label="2">
          <Coord>0.450 0 0</Coord>
        − <NodeValues>
            <Restraints>false true true true true true</Restraints>
          − <Loads>
              <Case label="1">-6000 0 0 0 0 0</Case>
            </Loads>
          </NodeValues>
        </Node>
      − <Node label="1">
          <Coord>0 0 0</Coord>
        − <NodeValues>
            <Restraints>true true true true true true</Restraints>
          </NodeValues>
        </Node>
      </NodeList>
    − <ElementList>
      − <Element class="ParametricElement.Bar.HermiteL2" label="1">
          <Incidence>1 2</Incidence>
          <AnalysisModel>EulerSpaceFrame</AnalysisModel>
          <IntegrationOrder>2 1 1</IntegrationOrder>
          <ElmCrossSections>section1 section1</ElmCrossSections>
        </Element>
      </ElementList>
    − <LoadCombinations>
      − <LoadCombination label="1">
          <LoadCase label="1" inc="true" factor="1.0"/>
        </LoadCombination>
      </LoadCombinations>
      <GlobalAnalysisModel>EulerSpaceFrame</GlobalAnalysisModel>
    </Model>
  </Insane>
```

Figura B.2: Formato do arquivo XML de entrada de dados (2^a parte).

```
− <Insane>
  − <OutPut>
     + <OutPutNode label="2"></OutPutNode>
    </OutPut>
  − <Solution class="EquilibriumPath">
       <NumMaxSteps>120</NumMaxSteps>
     + <Step class="StandardNewtonRapson"></Step>
     − <IteractiveStrategyList>
       + <IteractiveStrategy class="DisplacementControl" LoadFactor="-0.00002"></IteractiveStrategy>
      </IteractiveStrategyList>
      <ChangeLoadFactor>-0.00001</ChangeLoadFactor>
      <ChangeTolerance>0.0001</ChangeTolerance>
      <ChangeNumMaxIterations>100</ChangeNumMaxIterations>
    </Solution>
  − <Model class="FemModel">
       <ProblemDriver>ParametricFisicallyNonLinearSolidMech</ProblemDriver>
     − <MaterialList>
       + <Material class="ConcreteCarreira" label="Concrete"></Material>
       + <Material class="LinearPlasticIsotropic" label="Steel"></Material>
      </MaterialList>
     − <CrossSectionList>
       + <CrossSection label="section1"></CrossSection>
      </CrossSectionList>
     − <NodeList>
       + <Node label="2"></Node>
       + <Node label="1"></Node>
      </NodeList>
     − <ElementList>
       + <Element class="ParametricElement.Bar.HermiteL2" label="1"></Element>
      </ElementList>
     − <LoadCombinations>
       + <LoadCombination label="1"></LoadCombination>
      </LoadCombinations>
      <GlobalAnalysisModel>EulerSpaceFrame</GlobalAnalysisModel>
    </Model>
  </Insane>
```

Figura B.3: Formato resumido do arquivo XML de entrada de dados.

```
− <Insane>
  − <Solution class="EquilibriumPath">
      <NumMaxSteps>120</NumMaxSteps>
    + <Step class="StandardNewtonRapson"></Step>
    − <IteractiveStrategyList>
      + <IteractiveStrategy LoadFactor="-2.0E-5" class="DisplacementControl"></IteractiveStrategy>
      </IteractiveStrategyList>
      <StepNumber>1</StepNumber>
      <FinalLoadFactor>1.857E-02</FinalLoadFactor>
    </Solution>
  − <Model class="FemModel">
      <ProblemDriver>ParametricFisicallyNonLinearSolidMech</ProblemDriver>
      <GlobalAnalysisModel>EulerSpaceFrame</GlobalAnalysisModel>
    + <MaterialList></MaterialList>
    − <CrossSectionList>
      − <CrossSection label="section1">
        − <CalculatedGeometricProperties>
            <SectionArea>4.909E-02</SectionArea>
            <InertiaY>1.833E-04</InertiaY>
            <InertiaZ>1.833E-04</InertiaZ>
            <PolarInertia>3.666E-04</PolarInertia>
          </CalculatedGeometricProperties>
        </CrossSection>
      </CrossSectionList>
    − <NodeList>
      − <Node label="2">
          <NodeValues/>
          <Coord>4.500E-01 0.000E00 0.000E00</Coord>
          <Restraints>false true true true true true</Restraints>
        + <Loads></Loads>
        − <Displacements>
            -2.000E-05 0.000E00 0.000E00 0.000E00 0.000E00 0.000E00
          </Displacements>
        </Node>
      + <Node label="1"></Node>
      </NodeList>
    − <ElementList>
      − <Element class="ParametricElement.Bar.HermiteL2" label="1">
          <Incidence>1 2</Incidence>
          <AnalysisModel>EulerSpaceFrame</AnalysisModel>
          <IntegrationOrder>2 1 1</IntegrationOrder>
          <ElmCrossSections>section1 section1 </ElmCrossSections>
        − <DegenerationList>
          − <Degeneration label="IP-1">
              <GeneralizedStrains>-4.444E-05 0.000E00 0.000E00 0.000E00 </GeneralizedStrains>
              <GeneralizedStress>-1.114E02 0.000E00 0.000E00 -2.021E-17 </GeneralizedStress>
            − <SectionPoint label="MP-1">
                <PointStrains>-4.444E-05 0.000E00 0.000E00</PointStrains>
                <PointStress>-2.047E03 0.000E00 0.000E00</PointStress>
              </SectionPoint>
            + <SectionPoint label="MP-2"></SectionPoint>
            + <SectionPoint label="MP-3"></SectionPoint>
            + <SectionPoint label="MP-4"></SectionPoint>
            + <SectionPoint label="MP-5"></SectionPoint>
            + <SectionPoint label="MP-6"></SectionPoint>
            + <SectionPoint label="MP-7"></SectionPoint>
            + <SectionPoint label="MP-8"></SectionPoint>
            + <SectionPoint label="MP-9"></SectionPoint>
            + <SectionPoint label="MP-10"></SectionPoint>
            + <SectionPoint label="MP-11"></SectionPoint>
            + <SectionPoint label="MP-12"></SectionPoint>
            + <SectionPoint label="MP-13"></SectionPoint>
            + <SectionPoint label="MP-14"></SectionPoint>
            + <SectionPoint label="MP-15"></SectionPoint>
            + <SectionPoint label="MP-16"></SectionPoint>
            + <SectionPoint label="MP-17"></SectionPoint>
            + <SectionPoint label="MP-18"></SectionPoint>
            + <SectionPoint label="MP-19"></SectionPoint>
            + <SectionPoint label="MP-20"></SectionPoint>
            + <SectionPoint label="MP-21"></SectionPoint>
            + <SectionPoint label="MP-22"></SectionPoint>
            + <SectionPoint label="MP-23"></SectionPoint>
            + <SectionPoint label="MP-24"></SectionPoint>
            + <SectionPoint label="MP-25"></SectionPoint>
            + <SectionPoint label="MP-26"></SectionPoint>
            + <SectionPoint label="MP-27"></SectionPoint>
            + <SectionPoint label="MP-28"></SectionPoint>
            + <SectionPoint label="MP-29"></SectionPoint>
            + <SectionPoint label="MP-30"></SectionPoint>
            + <SectionPoint label="MP-31"></SectionPoint>
            + <SectionPoint label="MP-32"></SectionPoint>
            </Degeneration>
          + <Degeneration label="IP-2"></Degeneration>
          </DegenerationList>
        </Element>
      </ElementList>
    </Model>
  + <LoadCombinations></LoadCombinations>
  </Insane>
```


Figura B.4: Formato resumido do arquivo XML de saída de dados.

Referências Bibliográficas

Almeida, M. L., 2005. Elementos finitos paramétricos implementados em java. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

Batoz, J. L. e Dhat, G., 1979. 'Incremental displacement algorithms for nonlinear problems'. *International Journal for Numerical Methods in Engineering*, vol. 14, pp. 1262–1267. (citado em Fuina (2004)).

Caetano, C., 2004. *CVS - Controle de Versões e Desenvolvimento Colaborativo de Software*. Novatec Editora, São Paulo.

Carreira, D. J. e Chu, K., 1985. 'Stress-strain relationship for plain concrete in compression'. *ACI Journal*, vol. 82, pp. 797–804.

Crisfield, M. A., 1981. 'A fast incremental-iteractive solution procedure that handles snap-through'. *Computers & Structures*, vol. 13, pp. 55–62. (citado em Fuina (2004)).

Crisfield, M. A., 1983. 'An arc length method including line searches and acelerations'. *International Journal for Numerical Methods in Engineering*, vol. 19, pp. 1269–1289. (citado em Fuina (2004)).

Dawe, D. J., 1984. *Matrix and Finite Element Displacement Analysis of Structure*. Clarendon Press, Oxford.

Fuina, J. S., 2004. Métodos de controle de deformações para análise não-linear de estruturas. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

Fuina, J. S., 2006. Modelagem de Meios Parcialmente Frágeis Heterogêneos Utilizando o Contínuo de Cosserat e o Modelo de Microplanos. Tese de Doutorado, Universidade Federal de Minas Gerais, Belo Horizonte. (em desenvolvimento).

Galgoul, N. S., 1979. 'Dimensionamento de seção qualquer à flexão composta'. *Revista Estrutura*, vol. 86, pp. 99–112.

Germanio, L., 2005. Implementação orientada a objetos da solução de problemas estruturais dinâmicos via método dos elementos finitos. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

Gonçalves, M. A., 2004. Geração de malhas bidimensionais de elementos finitos baseada em mapeamentos transfinitos. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

Izzuddin, B. A., Siyam, A. A. F. M. e Smith, D. L., 2002. 'An efficient beam-column formulation for 3d reinforced concrete frames'. *Computers and Structures*, vol. 80, pp. 659–676.

Moreira, R. N., 2006. Sistema gráfico interativo para ensino do método dos elementos finitos. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

NBR6118, 2003. *Projeto de Estruturas de Concreto Armado*. Associação Brasileira de Normas Técnicas - ABNT.

Neves, R. A., 2000. Cálculo de esforços e deslocamentos em estruturas de pisos de edifícios, considerando-se a influência das tensões cisalhantes. Dissertação de Mestrado, Escola de Engenharia de São Carlos - Universidade de São Paulo, São Carlos.

Oñate, E., 1995. *Cálculo de Estructuras por el Método de Elementos Finitos - Análisis Estático Lineal*. Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona.

Penna, S. S., 2006. Pós-processador para resultados de análise não-linear de modelos do método dos elementos finitos. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte. (em desenvolvimento).

Pitangueira, R. L., 1998. Mecânica de Estruturas de Concreto com Inclusão de Efeitos de Tamanho e Heterogeneidade. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

Pitangueira, R. L., 2000. *Introdução ao Método dos Elementos Finitos - Notas de Aula*. Departamento de Engenharia de Estruturas da UFMG, Belo Horizonte.

Pitangueira, R. L., Filho, A. V. e Fonseca, F. T., 2004, Implementação de modelos estruturais de barras como casos particulares do método de elementos finitos, *in* 'Simpósio Mineiro de Mecânica Computacional - SIMMEC 2004', Itajubá, Minas Gerais.

Ramm, E., 1981. 'Strategies for tracing the nonlinear response near limit points, in nonlinear finite element analysis in structural mechanics'. , pp. 63–83. (citado em Fuina (2004)).

Ricks, E., 1972. 'The application of newton method to the problem of elastic stability'. *Journal of Applied Mechanics*, pp. 1060–1065. (citado em Fuina (2004)).

Ricks, E., 1979. 'An incremental approach to the solution of snapping and buckling problems'. *International Journal of Solids and Structures*, vol. 15, pp. 529–551. (citado em Fuina (2004)).

Romero, M. L., Miguel, P. F. e Cano, J. J., 2002. 'A parallel procedure for nonlinear analysis of reinforced concrete three-dimensional frames'. *Computers and Structures*, vol. 80, pp. 1337–1350.

Saliba, S. S., 2006. Implementação computacional e análise crítica de elementos finitos de placas. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte. (em desenvolvimento).

Sfakianakis, M. G., 2001. 'Biaxial bending with axial force of reinforced, composite and repaired concrete sections of arbitrary shape by fiber model and computer graphics'. *Advances in Engineering Software*, vol. 33, pp. 227–242.

Simão, W. I., 2003. Modelos de armadura e aderência para análise não-linear de estruturas de concreto armado. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

Timoshenko, S. P. e Goodier, J. N., 1980. *Teoria da Elasticidade*. Guanabara Dois, Rio de Janeiro.

Vasconcellos Filho, A., 1986. *Teoria das Estruturas - Método dos Deslocamentos, Processo de Cross e Tabelas*. Escola de Engenharia da UFMG, Belo Horizonte.

Vecchio, F. J. e Emara, M. B., 1992. 'Shear deformations in reinforced concrete frames'. *ACI Structural Journal*, vol. 89, pp. 46–56.

Weaver, Jr., W. e Gere, J. M., 1980. *Matrix Analysis of Framed Structures*. D. Van Nostrand Company, New York.

Wolff, K. P., 2006. Implementação de um modelo para fissuração baseado no método dos elementos finitos estendido. Dissertação de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte. (em desenvolvimento).

Yang, Y. B. e McGuire, W., 1985. 'A work control method for geometrically nonlinear analysis'. *Proceedings of the International Conference on Numerical Methods in Engineering: Theory and Application*, pp. 913–921. (citado em Fuina (2004)).

Yang, Y. B. e Shieh, M. S., 1990. 'Solution method for nonlinear problems with multiple critical points'. *AIAA Journal*, vol. 28(12), pp. 2110–2116. (citado em Fuina (2004)).

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo