ESTUDO SOBRE A INCLUSÃO DA NÃO LINEARIDADE GEOMÉTRICA EM PROJETOS DE EDIFÍCIOS

ANDRÉ LUIZ BANKI

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia Civil da Universidade Federal de Santa Catarina, como requisito parcial para a obtenção do título de Mestre em Engenharia Civil

Área de Concentração: Estruturas

Orientador: Prof. Dr. Daniel Domingues Loriggio

Florianópolis – SC
1999

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CENTRO TECNOLÓGICO

DEPARTAMENTO DE ENGENHARIA CIVIL

CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

A Comissão Examinadora, abaixo assinada, aprova a dissertação entitulada:

ESTUDO SOBRE A INCLUSÃO DA NÃO LINEARIDADE GEOMÉTRICA EM PROJETOS DE EDIFÍCIOS

por:

ENGº. CIVIL ANDRÉ LUIZ BANKI

Como requisito para a obtenção do grau de

MESTRE EM ENGENHARIA CIVIL

Daniel Domingues Loriggio – Orientador

Jucilei Cordini – Coordenador do CPGEC

Comissão Examinadora:

Prof. Eduardo Alberto Fancello, Dr.

Prof. Fábio Armando Botelho Cordovil, Dr.

Profª. Henriette Lebre La Rovere, Dra.

Florianópolis, julho de 1999

AGRADECIMENTOS

A Deus, sempre em primeiro lugar.

A meus pais, por terem me dado todas as condições necessárias para chegar até aqui.

Ao Prof. Daniel Domingues Loriggio, pela tranquilidade em sua orientação e pelo incentivo que me deu para concluí-la.

À minha esposa e filha, que souberam compreender a importância deste trabalho e tolerar o tempo que gastei em frente do computador.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com a popularização das ferramentas computacionais e com as estruturas tornando-se cada vez mais arrojadas, torna-se indispensável uma melhora imediata na qualidade das análises envolvidas no projeto de edifícios, principalmente em termos de verificação da sua estabilidade global.

Este trabalho parte do estudo dos principais processos para a inclusão da não linearidade geométrica no projeto de edifícios. São estes o Processo P-Delta, o Processo da Matriz de Rigidez Geométrica e o Processo das Funções de Estabilidade.

Os três processos são comparados através de uma série de exemplos numéricos, mostrando a aplicação destes processos em modelos de pórticos planos que representam estruturas reais. A precisão nos resultados é verificada em função da discretização e do nível de carregamento adotados. A análise desses resultados fornece indicações para a correta utilização desses métodos no modelamento de estruturas.

Os resultados também são comparados com aqueles obtidos através de um procedimento simplificado, que corresponde a majorar as solicitações através do coeficiente Gama-Z, mostrando os possíveis erros contra a segurança que podem ser cometidos.

Procura-se dar o enfoque da aplicação em projeto, abordando questões práticas como o custo de processamento, o problema da inclusão da segurança e das combinações de ações. São fornecidas indicações a respeito da extensão dos procedimentos à análise tridimensional de estruturas.

Os três processos são implementados computacionalmente em um programa para resolução de pórticos planos. Tal programa, como objetivo secundário desta dissertação, foi desenvolvido, desde sua parte linear, utilizando-se a filosofia da Programação Orientada a Objetos. Mostram-se as diversas vantagens relativas ao uso desta metodologia, com destaque para a extensão do programa de análise linear à análise não linear geométrica utilizando-se herança de classes.

ABSTRACT

The structures are growing taller and more slender. This fact, besides of the accessibility of the the computational tools, must lead to an immediate improvement in the quality of the analysis involved in the structural design. A specially important point is the global stability of the structures.

This work starts studying the best known methods for inclusion of the geometric nonlinearity in the design of buildings. They are the P-Delta Method, the Geometric Stiffness Matrix Method and the Stability Functions Method.

The three methods are compared through a series of numeric examples, showing the application of these methods in plane frames that represent real structures. The precision of the results is verified in function of the discretization and the axial load level. The analysis of those results points to the correct handling of these methods in design of structures.

The results are also compared with those obtained through the simplified procedure of applying an amplification factor called Gamma-Z to the first-order solicitations. This work shows the possible unsafe differences that can be obtained.

The focal point of this work is the practical application of the studied methods, approaching subjects like processing cost, safety inclusion and load combinations. Indications are supplied regarding the extension of the methods into the three-dimensional analysis of structures.

The three methods are implemented in a plane frame solution program. This program was an secondary goal of this dissertation and was developed from the beginning under the Object Oriented Programming approach. The several advantages obtained with the use of this methodology are shown, with spots to the extension of the linear analysis program to the geometric nonlinear analysis using class inheritance.

CAPÍTULO 1.
INTRODUÇÃO

Nas últimas duas décadas, o volume de trabalhos publicados na área de Engenharia Estrutural foi bastante grande, apresentando sensível crescimento ano após ano. A grande popularização do Método dos Elementos Finitos, aliada à rápida evolução das ferramentas computacionais, permitiu aos pesquisadores analisar uma variada gama de estruturas, sob as mais variadas ações, criando modelos que se aproximam cada vez mais dos resultados obtidos em ensaios. Pode-se citar desde não linearidades até estudos dinâmicos e com grandes deslocamentos.

Paralelamente a este grande *boom* na área de pesquisa, ao se analisar a evolução dos procedimentos na área de Projeto, no mesmo período, a nível nacional, nota-se uma certa estagnação. Os conceitos utilizados hoje no projeto de uma edificação em concreto armado ainda são, basicamente, os mesmos que nortearam o desenvolvimento da norma brasileira NBR 6118/78 ("Projeto e Execução de Estruturas de Concreto Armado"). Muito embora as estruturas sejam dia a dia mais arrojadas, com a substituição dos antigos "paredões" de alvenaria por fachadas de vidro e lajes sem vigas, conceitos como a estabilidade global das estruturas ainda são usualmente tratados em projeto através de verificações simplificadas.

Embora existam muitos profissionais experientes, para os quais os conceitos de análise global da estrutura já são conhecidos e aplicados em projetos, e que acompanham as tendências mundiais em termos de normalização e evolução tecnológica, a disseminação das ferramentas computacionais a nível comercial e as dificuldades cada vez maiores no mercado de trabalho geraram outra classe de calculistas que se limitam aos conceitos a eles apresentados durante sua formação acadêmica e à utilização de ferramentas prontas.

Existe um hiato entre as áreas de Pesquisa e Projeto, que pode ser explicado, em parte, pela dificuldade de aplicação prática imediata da maior parte dos trabalhos publicados na área. O desenvolvimento das ferramentas computacionais tem revertido este fato, mas estudos mais detalhados ainda são absolutamente proibitivos para fins práticos. O modelo de uma estrutura através de elementos finitos sólidos, por exemplo, não apenas é completamente inviável nos dias de hoje como também possui aplicação prática questionável em termos de projeto. Muitas vezes, a utilização de modelos mais simples fornece resultados não apenas mais rápidos como também mais confiáveis, visto partirem de premissas e informações mais claras.

Este trabalho de dissertação, dentro da linha de pesquisa "Projeto e Análise de Estruturas", busca diminuir a diferença entre estas duas áreas, abordando a inclusão da não linearidade geométrica no projeto de estruturas. Este enfoque do trabalho segue parte dos objetivos hoje trabalhados no Grupo de Estruturas do Curso de Pós-Graduação em Engenharia Civil da Universidade Federal de Santa Catarina (UFSC). Neste trabalho, pretende-se apresentar uma sistematização dos principais processos disponíveis para análise de estruturas utilizando efeitos não lineares geométricos, visando sua aplicação em projetos usuais. Embora sejam mencionadas basicamente as estruturas em concreto armado, visto constituírem a maior parte das obras construídas e hoje em projeto no Brasil, a maior parte dos conceitos aqui apresentados é genérica e podem ser diretamente estendida a outros tipos de estruturas, sejam estas em aço ou madeira, por exemplo.

Apesar da existência de estudos bastante avançados na área, ainda é difícil retirar destes formas simplificadas aplicáveis a projetos correntes. Ao invés disto, este trabalho tomará como base os conceitos utilizados correntemente em Projeto, baseados na Análise Elástica Linear, introduzindo-se a partir disto conceitos mais aprimorados de não linearidade geométrica.

Dentro desta filosofia, serão utilizados modelos de pórticos planos. Em primeiro lugar, os modelos reticulados, sejam planos ou espaciais, além de serem muito difundidos e fazerem parte das ferramentas mais utilizadas em projeto, apresentam um número total de graus de liberdade muito menor do que os modelos sólidos ou de chapas em Elementos Finitos, o que resulta em um custo computacional muito inferior. Em termos de projeto, os modelos baseados em pórticos, onde os elementos são representados por seus eixos geométricos, conseguem representar a estrutura de uma edificação com um grau de precisão, dentro dos parâmetros da Engenharia, que torna usualmente dispensável o uso de modelos mais elaborados. Estes últimos normalmente se restringem a estudos de verificação e análise de efeitos localizados.

Como parte deste trabalho, será descrita a implementação computacional dos procedimentos lineares e não lineares, utilizando técnicas de Programação Orientada a Objeto, facilitando a sua utilização em trabalhos posteriores.

A extensão dos procedimentos para o caso 3D, embora não esteja presente na implementação computacional a ser apresentada, será abordada na parte final do trabalho. Pode-se considerar esta extensão relativamente imediata, constituindo-se em outra vantagem dos modelos de barras. Os efeitos não lineares geométricos presentes nas estruturas planas estendem-se para o caso espacial de forma similar aos efeitos lineares. Neste trabalho, optou-se por trabalhar com modelos planos pois, além de possuírem ainda certa aplicação prática, permitem que seus resultados sejam mais facilmente visualizados. A representação gráfica de dados e resultados é mais simples, facilitando sua compreensão. Uma completa comparação entre o comportamento de uma estrutura sob análise plana ou espacial, mesmo considerando-se apenas efeitos lineares, foge ao escopo deste trabalho, o qual procura apresentar os apenas os efeitos adicionais gerados pelos deslocamentos de 2ª ordem.

1.1 - BREVE HISTÓRICO

Enquanto a área de Pesquisa em Análise Estrutural trabalha com conhecimento globalizado e publicação de artigos em âmbito nacional e internacional, a área de Projeto sempre foi balizada pela normalização corrente. O fato de que os resultados obtidos através de pesquisa são apenas lentamente inseridos na normalização favorece a distância entre as duas áreas. O hiato entre Pesquisa e Projeto é algo que pode ser considerado natural, visto que novos modelos que surgem na área de Pesquisa devem ser exaustivamente validados antes de permitirem aplicação prática em Projeto. Todavia, este hiato apresenta-se mais fortemente no cenário nacional, quando comparado com a mesma questão a nível internacional. Pode-se apontar como um dos fatores que levam a esta situação o fato de se ter uma normalização oficial hoje relativamente defasada.

Embora não se possa considerar a formação acadêmica a única fonte do conhecimento teórico adquirido por um calculista de estruturas, esta baseia-se nas normas vigentes e em bibliografia local. Aqui desponta outra questão, a da produção bibliográfica nacional, pequena na última década. A maior parte da bibliografia disponível remonta ao início da década de 80, quando ainda se preocupavam mais com o caráter prático do projeto do que com a análise global de uma edificação.

A primeira normalização brasileira na Engenharia de Estruturas foi criada pela Associação Brasileira de Cimento Portland, em 1931. Em 1940, a ABNT (Associação Brasileira de Normas Técnicas) lançou a primeira edição da NB-1, denominada "Projeto e execução de obras de concreto armado". Durante as décadas de 40 e 50, a evolução na área de Pesquisa foi suficientemente lenta para que as normas fossem gradualmente adaptadas. A revisão NB-1/60 representava bem o conhecimento da época.

Nas décadas de 60 e 70, os avanços obtidos na Engenharia de Estruturas foram bem mais significativos. Na transição da NB-1/60 para a NB-1/78, houve necessidade de uma modificação completa nos conceitos apresentados pela norma. A consideração do fenômeno da Instabilidade no dimensionamento dos pilares substituiu os conceitos antigos baseados no modelo da Flambagem. O conceito de Estados Limites e o processo Semi Probabilístico para verificação da segurança, bem como uma revisão radical das notações, também podem ser destacados.

Todavia, conforme cita FUSCO (1993): "Em relação ao projeto, o que se pode dizer é que a NB-1 sempre foi uma norma para o projeto de *peças* de concreto armado, com algumas indicações referentes a certos aspectos do projeto estrutural de alguns tipos de construção". A NB-1/78, agora indicada como NBR 6118/78, tratava a análise das estruturas apenas superficialmente em seu item 3 ("Esforços solicitantes"). Em seu item 3.2.1 ("Método de cálculo"), cita apenas "Os esforços solicitantes das estruturas lineares poderão ser determinados em regime elástico ou elasto-plástico. (...) Para o cálculo em regime elástico, admitir-se-á o módulo de deformação previsto em 8.2.5; a área e o momento de inércia das seções poderão ser calculados para a seção transversal geométrica, sem consideração da armadura. (...)".

A abordagem dada aos efeitos de 2ª ordem, às imperfeições geométricas, à deformação lenta e à instabilidade não tinha um enfoque da estrutura como um todo, limitando-se ao cálculo dos pilares. Sob o item 4.1.1.3 ("Compressão por força normal Fd – barras isoladas"), encontra-se apenas "O cálculo, que abrange tanto o caso de ruína por ruptura à compressão do concreto como o de ruína por instabilidade, será feito: – pelo processo exato (obrigatório quando $\lambda > 140$) que considera a relação momento-curvatura, baseada nos diagramas σ-ε do concreto e do aço, ou por processo aproximado devidamente justificado; (...) – pelo processo simplificado descrito no item 4.1.1.3-C quando tratar-se do caso particular deste item".

A NBR 6118/78 é ainda considerada a norma vigente e é o texto que norteia a formação acadêmica dos novos profissionais.

Citando ZERMIANI (1998), "Os conceitos da não linearidade geométrica foram bastante estudados no final da década de 60 e início da de 70, (...), porém, sua aplicação estava limitada pela ferramenta computacional disponível. Não se popularizou e nem tampouco distribuiu-se pelo mundo, pois os computadores eram de custo altíssimo e não estavam disponíveis na maioria dos escritórios de projeto". Embora conceitos de estabilidade global de estruturas, normalmente utilizando processos aproximados para estimativa da grandeza relativa dos efeitos de 2ª ordem, já fossem razoavelmente conhecidos a nível mundial, nada sobre isto foi incluído na norma.

Paralelamente a isto, o desenvolvimento da área de Pesquisa em Análise Estrutural sofreu seu grande "boom" nas décadas de 80 e 90 que se seguiram à publicação da NBR 6118/78. O desenvolvimento praticamente exponencial das ferramentas computacionais permitiu a centros de pesquisa no mundo todo obter um novo patamar de conhecimento, modelando efeitos cada vez mais complexos e estendendo a análise de estruturas de peças isoladas para estruturas inteiras e de grande complexidade.

1.2 - ESTADO ATUAL DE PROJETO

Embora a área de Análise tenha se desenvolvido cientificamente nas últimas décadas, as novas tecnologias divulgadas através de artigos em revistas internacionais especializadas possuem grau de penetração desprezível no meio técnico nacional. A isto se agrega o fato de que a maioria destes artigos trata de itens muito específicos e de pouca aplicação prática imediata em um escritório de cálculo. Com exceção de uma pequena parte dos profissionais que têm condição de investir no aprimoramento de sua formação, seja através de estudo formal ou não, a grande maioria das pessoas que hoje trabalham no Projeto de Estruturas possui grande dificuldade em ter acesso a estas tecnologias, não por não existirem meios de acesso, mas por não serem de disponibilidade prática. O profissional hoje busca ocasionalmente apoio teórico para itens específicos, mas não está atento a mudanças de paradigma em relação aos procedimentos que está acostumado a executar.

Pode-se dizer, portanto, que a área de Projeto Estrutural ainda baseia-se, hoje, no estudo de peças isoladas e na utilização de ferramental automatizado, seja através de programas comerciais ou roteiros de cálculo simplificados. A bibliografia disponível na área, embora de boa qualidade, norteia-se pelos mesmos princípios da NBR 6118/78. Por exemplo, SUSSEKIND (1982) trata a não linearidade física (através de relação momento fletor-curvatura) e o cálculo com não linearidade geométrica através de processo iterativo (representando o chamado "processo exato" da NBR 6118/78) apenas em seu item final ("Pilares esbeltos – $\lambda > 80$"), a mesma abordagem utilizada pela norma. Não é feita ligação com a estrutura como um todo.

Em termos de análise das estruturas, um dos aspectos mais significativos trata da verificação simplificada da estabilidade global das edificações. Neste ponto, na omissão da norma brasileira, a bibliografia especializada segue o critério sugerido pelo CEB (Comite Euro-International Du Beton), segundo o qual um coeficiente α representa a subestrutura de contraventamento da edificação. Caso este valor situe-se abaixo de um certo valor limite, considera-se que os efeitos de 2ª ordem não ultrapassam em mais de 10% aqueles provenientes da teoria de 1ª ordem, podendo, portanto, ser simplesmente desprezados. SUSSEKIND (1982) afirma: "Consideramos qualidade indispensável no projeto de uma estrutura de edifício o atendimento às condições de robustez mínima". Uma abordagem usualmente dada ao problema era apenas a de considerar a estrutura deslocável e com isto modificar o comprimento equivalente para cálculo do índice de esbeltez dos pilares. Estranhamente, este tipo de abordagem ainda é muito utilizada a nível internacional, constando, por exemplo, das recomendações da norma norte-americana e de diversos artigos, mesmo recentes. Outra abordagem sempre foi a de simplesmente enrijecer a estrutura até que se obtivesse um parâmetro de instabilidade inferior ao limite, ficando as análises mais elaboradas restritas a alguns casos especiais.

Uma referência mais completa sobre o assunto pode ser encontrada em FUSCO (1981), onde discorre com bastante propriedade sobre o Estado Limite Último de Instabilidade. Na apresentação de seu "Método Geral", considera tanto a não linearidade física como geométrica do sistema, citando o fato do que o mesmo processo pode ser estendido dos pilares às estruturas hiperestáticas. Todavia, o enfoque básico ainda é o do cálculo de pilares esbeltos. Também apresenta o processo α para avaliação da estabilidade global da edificação e inclui um item referente ao cálculo exato de pórticos considerando a não linearidade: "A verificação exata da segurança de pórticos hiperestáticos somente pode ser feita através do método geral, com uma análise não linear. Em cada etapa do processo de resolução, deve ser resolvido o problema hiperestático". O processo incluído sob o item "Pórticos hiperestáticos – Cálculo rigoroso" pode ser reconhecido como o Processo P-Delta que será apresentado no decorrer desta dissertação. Todavia, afirma que "As dificuldades materiais desse cálculo exato são excessivas para a sua aplicação prática".

Fundamentalmente, este é o panorama geral da área de Projeto. A formação de novos profissionais, bem como grande parte dos projetos atualmente desenvolvidos, concentra-se na consideração apenas simplificada dos efeitos de 2ª ordem no cálculo de pilares esbeltos e no máximo em uma avaliação relativa da grandeza dos efeitos globais de 2ª ordem através da verificação da robustez mínima das subestruturas de contraventamento.

Paradoxalmente, os maiores avanços técnicos têm sido incorporados aos projetos usuais pelos programas comerciais de cálculo de estruturas em concreto armado, ao mesmo tempo em que difundem a utilização de processos sem que estes tenham que ser efetivamente aprendidos por seus usuários. Na década de 80, estes programas dedicavam-se ao cálculo de elementos isolados (vigas, lajes, pilares tratados em separado), delegando ao usuário a tarefa da compreender e analisar o comportamento da estrutura como um todo, enquanto que, hoje, a difusão das ferramentas de CAD (Computer Aided Design) permite aos principais programas do mercado trabalhar com a estrutura de forma gráfica, considerando a interação entre vigas e pilares através de modelos de pórticos espaciais e modelando pavimentos inteiros através do Método dos Elementos Finitos ou da Analogia de Grelha. Verificações de estabilidade global e inclusão dos efeitos da força de vento podem ser automaticamente efetuados pelos programas. Considerações de não linearidade geométrica através do Processo P-Delta ou do coeficiente amplificador Gama-Z já estão disponíveis no projeto de estruturas complexas.

Infelizmente, esses programas ainda são utilizados basicamente como "caixas pretas" por grande parte dos calculistas, não propriamente por não esclarecerem os modelos sobre os quais trabalham, mas principalmente pela lacuna existente entre a tecnologia utilizada pelo *software* e a formação técnica do profissional que o utiliza. A aplicação de um modelo de Elementos Finitos por um profissional que jamais teve contato com suas premissas ou o uso de modelos espaciais por outro que sequer utilizou anteriormente modelos de pórticos planos não apenas é nitidamente questionável, como vem se transformando no panorama mais comum na área de Projeto. Esta é uma das preocupações que norteia este trabalho e um dos motivos pelos quais se optou em utilizar modelos mais simples, cuja compreensão é fundamental para a aplicação de modelos mais complexos.

1.3 - A NOVA NBR 6118

Conforme exposto por FRANÇA & STUCCHI (1993), a revisão de parte da NBR 6118/78 iniciou em 1990 e a elaboração do texto base procurou apoio na normas atuais NBR 6118/78 e NBR 7197/89 ("Projeto de Estruturas de Concreto Protendido") e lançou mão do Eurocode 2 e do Código Modelo do CEB (MC-90). O Eurocode 2 é a norma da Comunidade Européia e baseia-se bastante no Código Modelo MC-78, incorporando, porém, alguns avanços do MC-90. O texto base, em sua essência, ficou pronto em dezembro de 1992, tendo sido estabelecida, pouco depois, na ABNT, a Comissão de Estudos para revisão da NBR 6118. Até o momento da finalização desta dissertação, ainda não se concluíram os trabalhos, permanecendo a NBR 6118/78 como norma vigente, sendo os avanços pouco a pouco divulgados para o meio técnico. Os autores citam como um dos princípios básicos da nova NBR 6118: "Procura-se privilegiar a visão da estrutura como um todo, dando ênfase a todas as etapas do projeto, (...). Procurou-se abandonar o esquema de uma norma aparentemente voltada ao dimensionamento e verificação de seções ou de peças estruturais".

Dentro deste enfoque, a nova NBR 6118 deve incluir, entre outras modificações, um tópico específico para a Análise Estrutural, procurando distinguir suas possíveis variações e aplicabilidade, além de um tópico específico para a Instabilidade e Efeitos de Segunda Ordem, agora não mais tratando apenas os efeitos locais como fazia a NBR 6118/78.

Segundo SANTOS & FRANCO (1993), a nova NBR 6118 deverá incluir um capítulo denominado provisoriamente "Instabilidade e Efeitos de Segunda Ordem", onde afirmam que "Os efeitos de 2ª ordem, em cuja determinação deve ser levado em conta o comportamento não linear dos materiais, podem ser desprezados sempre que não representem acréscimo superior a 10% nas reações e solicitações relevantes na estrutura". Os autores classificam as estruturas como sendo de "nós fixos" (onde os efeitos de 2ª ordem não excedem o limite de 10%) ou de "nós móveis" (no caso contrário). Para fazer esta distinção, fornecem dois parâmetros simplificados que podem ser utilizados para estimar os efeitos de 2ª ordem : o Parâmetro de Estabilidade α e o Coeficiente γz. Caso a estrutura seja considerada como sendo de nós móveis, afirmam que a nova NBR 6118 deve obrigar a consideração tanto da não linearidade geométrica como da física no cálculo da estrutura.

Segundo os autores, para a consideração da não linearidade geométrica, a nova NBR 6118 permitirá o emprego do Processo P-Delta, levando em conta a não linearidade física de forma aproximada através de reduções convenientes na rigidez das peças para o cálculo de 1ª ordem. Citam ainda que: "A nova NB-1 permite uma simplificação original para a determinação dos esforços finais de 2ª ordem, válida para estruturas regulares de edifícios: os esforços totais (1ª + 2ª ordem) podem ser obtidos pela simples multiplicação dos valores dos esforços de 1ª ordem pelo coeficiente γz, desde que $\gamma z \leq 1.2$".

De acordo com FRANÇA & STUCCHI (1993), um dos objetivos da nova NBR 6118 é "sinalizar como serão aplicados procedimentos mais complexos, pois nos próximos anos deverá ocorrer um desenvolvimento acelerado de *software*." Nesta filosofia, procedimentos simplificados foram colocados ao lado de outros mais aprimorados. Faltam, talvez, comentários mais detalhados sobre estes, facilitando seu acesso à comunidade dos projetistas estruturais.

1.4 - OBJETIVOS

Neste panorama, destaca-se como objetivo principal deste trabalho de dissertação a divulgação de procedimentos que permitam a imediata inclusão da não linearidade geométrica no projeto de estruturas de concreto armado.

Para tal, passa-se pelo estudo dos diferentes processos para consideração da não linearidade geométrica. Pretende-se expor e comparar os três principais processos disponíveis, a saber, o Processo P-Delta, o Processo da Matriz de Rigidez Geométrica e o Processo das Funções de Estabilidade. Outros processos disponíveis na bibliografia serão citados, mas estes três representam os processos mais conhecidos e mais testados, sendo mais adequados para aplicação em projeto.

Objetiva-se verificar a validade de algumas das considerações e procedimentos simplificados, tanto aqueles correntes como aqueles presentes na proposta de revisão da NBR 6118. Ao final, pretende-se mostrar que as ferramentas computacionais hoje disponíveis tornam questionável a adoção de certas simplificações, incentivando o uso dos processos citados para inclusão da não linearidade geométrica.

1.5 - METODOLOGIA

Pode-se encontrar na bibliografia processos bastante avançados para inclusão dos efeitos da não linearidade geométrica no cálculo de estruturas. Grande parte destes estudos trabalha com modelos planos em Elementos Finitos para análise da interação entre vigas e pilares. O problema inserido em tal tipo de procedimento passa pela difícil extensão ao caso tridimensional, praticamente indispensável à aplicação corrente em Projeto de Estruturas. Cita-se que existem na área diversos estudos também acerca deste tipo de modelo. Neste trabalho, contudo, não se pretende explanar sobre esta enorme gama de aspectos, por considerar-se o custo computacional elevado demais para aplicação prática imediata.

Ao invés de trabalhar com os resultados mais recentes fornecidos pela área de Pesquisa, a metodologia deste trabalho será a de partir dos procedimentos atualmente utilizados pela área de Projeto e incluir a estes, gradativamente, considerações mais avançadas.

Dentro deste panorama, optou-se por utilizar modelos de barras e não elementos finitos de placas ou sólidos. Isto porque, em primeiro lugar, este tipo de modelo é o utilizado atualmente pelos escritórios de cálculo. Em segundo lugar, porque diversos estudos já apontaram a aplicabilidade deste tipo de procedimento no modelo da não linearidade geométrica.

Para facilitar a exposição dos procedimentos, optou-se por trabalhar com modelos de pórticos planos. Este tipo de modelo, além de possuir aplicação própria, pode ser estendido para o caso espacial de forma relativamente direta. Comentários sobre o assunto serão feitos quando se mostrar relevante.

Para viabilizar a aplicação em estruturas, os procedimentos serão implementados em um programa computacional. Este programa será elaborado em linguagem C++ e fará uso de técnicas de Programação Orientada a Objetos. Inicialmente, será implementada a análise linear de 1ª ordem, utilizando conceitos já bastante conhecidos da Análise Matricial de Estruturas. Como segundo passo, serão feitas as modificações necessárias para a inclusão dos efeitos da não linearidade geométrica. Como passo final, vai-se utilizar o programa elaborado para modelar alguns exemplos simples de pilares isolados e pequenos pórticos, procurando obter destes algumas conclusões a respeito da validade dos resultados e sua relação com processos simplificados conhecidos.

1.6 - CONTRIBUIÇÕES, VIABILIDADE E IMPORTÂNCIA

Como contribuição acadêmica, este trabalho irá reunir, de forma clara e sistemática, os conceitos envolvidos na consideração de efeitos não lineares geométricos nas estruturas de concreto armado, constituindo-se em um material que pode ser utilizado como complemento ou atualização da formação de profissionais da área de Cálculo Estrutural.

Pretende-se, contudo, que a contribuição principal deste trabalho seja a divulgação de procedimentos que podem ser aplicados sem dificuldade na área de Projeto. Pretende-se sinalizar aos desenvolvedores de programas para a área de Projeto Estrutural um caminho que pode ser percorrido para a inclusão dos efeitos não lineares em *softwares* existentes ou que venham a ser desenvolvidos.

O rápido desenvolvimento das ferramentas computacionais se encarregará de viabilizar, a curto prazo, a aplicação destes processos no dia-a-dia dos escritórios de projeto. Por esta razão, optou-se por adotar procedimentos próximos dos atualmente utilizados em projetos ao invés das pesquisas mais avançadas na área, seguindo uma linha de trabalho hoje defendida pelo Orientador dentro do Grupo de Estruturas no Curso de Pós-Graduação em Engenharia Civil da Universidade Federal de Santa Catarina (UFSC)..

Ao final deste trabalho, após a verificação dos exemplos e comparação com os resultados obtidos através dos processos simplificados utilizados atualmente, ficará clara a importância deste trabalho. Com a popularização das ferramentas computacionais, torna-se indispensável uma melhora imediata na qualidade das análises envolvidas no projeto de edifícios.

1.7 - ESTRUTURA DA DISSERTAÇÃO

Este trabalho de dissertação está estruturado em sete capítulos e trata essencialmente de processos para inclusão da não linearidade geométrica em pórticos planos de concreto armado. Para perfeita compreensão do assunto, recomenda-se a leitura dos capítulos em ordem, visto representarem uma evolução sistemática nos procedimentos de análise das estruturas. Exceção se faz ao quarto capítulo, onde é feita a exposição da implementação computacional, cuja leitura depende deste interesse específico.

No primeiro capítulo, introdutório, procurou-se dar uma idéia geral dos conceitos e objetivos envolvidos nesta dissertação. Um breve histórico a respeito a normalização brasileira na Engenharia de Estruturas pretendeu inserir o leitor nos assuntos a serem abordados no decorrer deste trabalho.

O segundo capítulo descreve a formulação matricial do Método dos Deslocamentos, procedimento hoje utilizado para o cálculo das estruturas em regime elástico-linear. Serão discutidas também as premissas básicas deste tipo de análise, incluindo os critérios que definem a sua aplicabilidade face às diversas fontes de não linearidade presentes no projeto de estruturas.

No terceiro capítulo, serão abordados os conceitos referentes à não linearidade geométrica. O enfoque principal será dado aos processos P-Delta, da Matriz de Rigidez Geométrica e das Funções de Estabilidade, bem como algumas outras propostas disponíveis na bibliografia. Será feita uma primeira comparação entre os processos, utilizando-se resultados analíticos.

No quarto capítulo, será descrita a implementação computacional, tanto da parte linear como dos processos destacados. Embora a implementação linear seja clássica, utiliza-se uma relativamente nova abordagem orientada a objetos, procurando mostrar as vantagens desta metodologia quando comparada à Programação Estruturada convencional.

No quinto capítulo, serão utilizados exemplos numéricos para comparar os resultados obtidos com cada processo, estudando também os fatores que influenciam a precisão obtida nos resultados. Os resultados obtidos serão relacionados com aqueles provenientes do processo simplificado do coeficiente Gama-Z, constante da proposta de revisão da NBR 6118.

O sexto capítulo tratará da aplicação da não linearidade geométrica aos projetos reais de estruturas, abordando diversos tópicos de interesse quando da aplicação da teoria apresentada. Incluem-se aqui, além do efeito tridimensional, considerações de segurança, combinações de ações e estados limites.

No sétimo e último capítulo, serão apresentadas as conclusões e recomendações obtidas através deste estudo para aplicação no Projeto de Estruturas.

Em anexo, encontram-se listagens e informações referentes à implementação computacional desenvolvida como apoio a esta dissertação.

CAPÍTULO 2.
ANÁLISE LINEAR

2.1 - PREMISSAS BÁSICAS

2.1.1 - Objetivo da Análise Estrutural

Conforme CORRÊA & RAMALHO (1993): "O principal objetivo de uma análise estrutural é determinar os efeitos das ações em uma estrutura, com a finalidade de efetuar verificações de estados limites últimos e de utilização. Assim sendo, a análise deve permitir que se estabeleçam as distribuições de esforços solicitantes internos, tensões, deformações e deslocamentos, em uma parte ou em toda a estrutura".

2.1.2 - Idealização estrutural

O modelo de cálculo escolhido para representar uma estrutura deve reproduzir, da melhor maneira possível, a geometria desta e seu comportamento frente às diversas solicitações que serão aplicadas sobre ela ao longo de sua vida útil. Ao mesmo tempo, este modelo deve ser suficientemente prático para que dele possam ser obtidas as respostas desejadas ao menor custo possível (por custo, pode-se entender também tempo). Não é mérito algum levar horas ao invés de minutos para definir que a tensão em um ponto da estrutura é de 4,12005 ao invés de 4,12. Isto porque o objetivo da Análise é o de fornecer resultados para um Projeto, no qual existe uma estrutura inteira a calcular e uma obra real a ser construída.

Além disto, deve-se lembrar que existem, em todos os tipos de análise, além de diferentes graus de simplificações, uma série de incertezas, relativas, por exemplo, ao comportamento dos materiais (obtido apenas de forma probabilística) e à real distribuição dos carregamentos. Muitas vezes, estas incertezas podem invalidar completamente a precisão adicional obtida através de uma análise mais refinada.

Com o objetivo de simplificar a análise, os elementos estruturais são divididos em três categorias (conforme exposto no item 5.2.1 do MC-90):

- Unidimensionais: quando uma das dimensões é muito maior do que as outras duas;
- Bidimensionais: quando uma das dimensões é relativamente menor do que as outras duas;
- Tridimensionais: quando não há uma dimensão predominante.

Os elementos unidimensionais, também chamados *elementos lineares* ou *elementos de barra*, são definidos como tal quando uma das dimensões, denominada *comprimento*, supera em pelo menos três vezes a maior dimensão da seção transversal. Este é o critério adotado pelo MC-90 (item 5.2.1) e que, conforme CORRÊA & RAMALHO (1993), deve ser colocado na nova NBR 6118.

São chamadas *estruturas reticuladas* aquelas que podem ser suficientemente representadas através de elementos de barra. Tais barras são ligadas entre si através de nós, que são, primariamente, pontos de intersecção entre membros, pontos de suporte ou extremidades livres das barras. Existem seis categorias de estruturas reticuladas: vigas, treliças planas, treliças espaciais, grelhas, pórticos planos e pórticos espaciais.

Os elementos que compõem as estruturas correntes de edifícios normalmente encaixam-se na definição de "elemento linear" preconizada pela norma. Exceção seja feita às lajes e às vigas parede. Conforme CORRÊA & RAMALHO (1993), a nova NBR 6118 permitirá considerar uma viga parede como componente de um sistema estrutural, representando-a também por um elemento linear, desde que se considere a deformação por cisalhamento. As lajes, para efeito de análise ao carregamento horizontal, podem ser consideradas, em conjunto, como uma chapa totalmente rígida em seu plano.

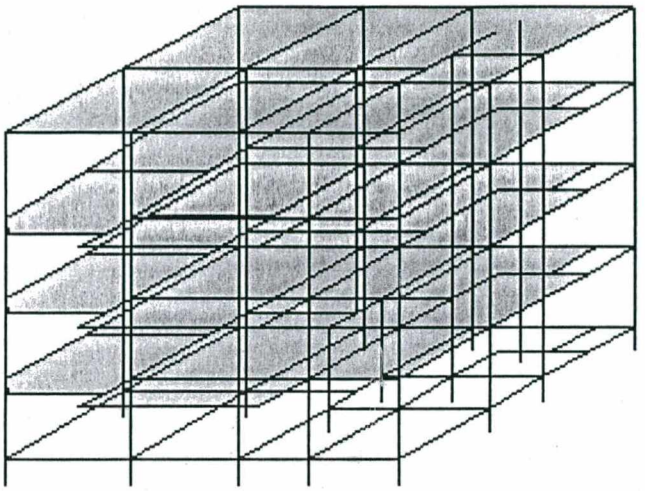


Figura 2.1 – Exemplo de uma estrutura de edificação

2.1.3 - Hipóteses básicas

São três as hipóteses básicas da Análise Estrutural:

- Condições de equilíbrio: A correta solução de qualquer das grandezas envolvidas (esforços internos, reações, deslocamentos) deve satisfazer simultaneamente a todas as condições de equilíbrio estático, não apenas da estrutura como um todo, mas também de qualquer parte desta quando analisada separadamente.
- Condições de compatibilidade: Estas condições dizem respeito à continuidade dos deslocamentos ao longo da estrutura, além de respeitar as condições de contorno (vinculação) e geométricas da estrutura.
- Carregamento monotônico: Admite-se que o carregamento definido para a estrutura seja aplicado sempre de forma proporcional e que não haja diferença entre os ciclos de carga e descarga.

2.1.4 - Análise linear

A análise de uma estrutura é dita *linear* quando admite-se comportamento elástico-linear para os materiais e quando a estrutura sofre deslocamentos suficientemente pequenos, tais que a mudança na geometria e nos pontos de aplicação das cargas não altere os resultados obtidos. Isto permite que todas as equações sejam escritas com relação à configuração indeformada da estrutura. Estes materiais obedecem à lei de Hooke, que define que a relação entre as tensões e as deformações no elemento é sempre linear.

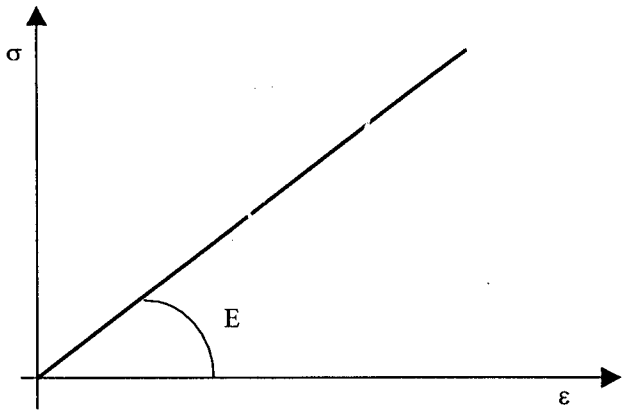


Figura 2.2 – Diagrama σ-ε para um material elástico-linear

Segundo a NBR 6118/78, item 3.2.2.1, neste tipo de análise, as características geométricas podem ser determinadas pelas seções brutas de concreto dos elementos estruturais, sem a necessidade de prévia determinação da armadura. Conforme CORRÊA & RAMALHO (1993), a nova NBR 6118 deve apresentar a mesma recomendação, mas alertando que este tipo de análise é aplicável quando se tem um nível de solicitação que provoque baixas tensões e os elementos estruturais não apresentem fissuras significativas.

2.2 - FUNDAMENTOS DE ANÁLISE MATRICIAL

2.2.1 - Abordagem matricial

O uso da notação matricial na resolução de problemas de Análise Estrutural traz duas vantagens essenciais, segundo LIVESLEY (1964):

- Do ponto de vista teórico, permite enunciar métodos de análise de uma forma compacta e precisa, mas, ao mesmo tempo, bastante genérica. Facilita, portanto, o tratamento da teoria estrutural como um conjunto único, sem que seus princípios fundamentais sejam obscurecidos por questões computacionais ou por diferenças físicas entre tipos de estruturas.
- Do ponto de vista prático, fornece uma metodologia sistemática de análise de estruturas que constitui-se em base extremamente adequada para uma implementação computacional.

Mesmo nas estruturas mais simples, a obtenção dos valores dos esforços e dos deslocamentos não pode ser feita pela simples aplicação de fórmulas algébricas conhecidas. Diversos processos podem ser aplicados para estruturas mais complexas, mas a utilização da notação matricial permite, se não reduzir o número total de operações numéricas a efetuar, mas sim estabelecer um procedimento genérico capaz de calcular tipos diferentes de estruturas mantendo o processo de cálculo encapsulado em um programa de computador. Tal enfoque é normalmente conhecido como a Análise Matricial de Estruturas.

2.2.2 - Métodos de resolução de estruturas reticuladas

A resolução destas estruturas discretizadas pode ser feita através de um dos dois métodos: o Método dos Deslocamentos (ou Método da Rigidez) e o Método das Forças (ou Método da Flexibilidade). A diferença básica entre estes dois métodos de resolução refere-se ao tipo de indeterminação considerada na análise da estrutura, sejam estas os esforços ou os deslocamentos.

Quando as ações são as incógnitas da análise, como é feito no Método das Forças, tem-se o caso de *indeterminação estática*. No caso, a indeterminação refere-se a um excesso de esforços desconhecidos quando comparados com o número de equações de equilíbrio disponíveis. Se houver mais esforços desconhecidos do que equações, a estrutura é dita *estaticamente indeterminada*, sendo estes esforços em excesso denominados de *incógnitas hiperestáticas*.

No Método dos Deslocamentos, as incógnitas passam a ser os deslocamentos dos nós da estrutura, consistindo em uma *indeterminação cinemática*. Este tipo de indeterminação é advinda do fato de que, em uma estrutura sujeita a certo carregamento, todos os seus nós sofrem deslocamentos. Em certos casos, alguns deles são conhecidos, devido ao fato de sobre eles ser imposto um deslocamento determinado (nós vinculados).

O Método das Forças baseia-se na obtenção das incógnitas hiperestáticas através de transformações na estrutura original (em número igual à quantidade de incógnitas), de onde se obtenha uma versão estaticamente determinada (sem os esforços redundantes), e de um conjunto de equações de compatibilidade, cuja solução conduz à solução da estrutura. Este método possui pouca aplicação computacional, devido ao fato de que a escolha da estrutura isostática fundamental a considerar não é única e depende de uma decisão a ser tomada.

O procedimento a adotar para a análise da estrutura consiste, primeiramente, na determinação do grau de indeterminação (quantidade de incógnitas) desta estrutura. A partir daí, são escolhidas as redundantes a liberar, criando-se, desta forma, a estrutura isostática fundamental, sendo esta estaticamente determinada. O procedimento, a partir disto, é o de calcular os deslocamentos nesta estrutura liberada (considerando-se as redundantes liberadas como cargas aplicadas, geralmente de valor unitário) e formular as equações de compatibilidade pertinentes.

O Método dos Deslocamentos, de forma oposta, é extremamente recomendado para a análise de estruturas através de métodos computacionais. Apesar das duas formulações serem matematicamente semelhantes, nesta não é necessária qualquer transformação da estrutura original, eliminando, portanto, a necessidade de qualquer decisão e constituindo-se em um algoritmo único e de fácil estruturação.

O procedimento é, de certa forma, análogo ao do método anterior, na medida em que é utilizada uma versão cinematicamente determinada da estrutura original em análise, e que utilizam-se equações de equilíbrio de forças para determinar os deslocamentos desconhecidos.

Em comparação, tornam-se óbvias as vantagens decorrentes da utilização desta última formulação. Como será visto adiante, o algoritmo de cálculo resultante é direto e de fácil codificação, diferentemente do que seria necessário para se programar computacionalmente uma resolução que utilizasse o Método das Forças.

2.2.3 - Graus de liberdade de uma estrutura

Conforme colocado por ZERMIANI (1998), "Um grau de liberdade é um tipo de movimento possível para cada uma das extremidades das barras de uma estrutura". Para o caso específico de pórticos planos, cada extremidade de uma barra possui três movimentos possíveis, que são:

- deslocamento na direção do eixo da barra;
- deslocamento na direção perpendicular ao eixo da barra; e
- giro em torno da extremidade da barra.

Como cada barra possui duas extremidades (na verdade, dois nós), terá ao total seis graus de liberdade.

É necessário adotar uma convenção para os sentidos positivo e negativo de cada um dos graus de liberdade. Neste trabalho, será adotada a convenção proposta por WEAVER & GERE (1965), apresentada na figura abaixo, onde as setas indicam o sentido positivo dos deslocamentos:

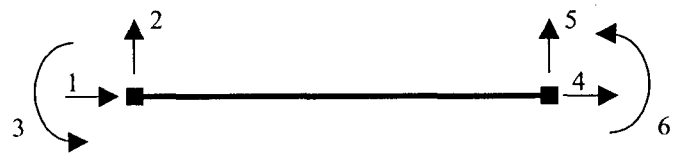


Figura 2.3 – Graus de liberdade para uma barra de pórtico plano

2.2.4 - Sistemas de referência local e global

Embora seja possível formular todo o método de forma direta, adotando um sistema de referência único e exprimindo a rigidez de cada elemento com relação a este sistema global, uma maneira de sistematizar este processo é definir um sistema de referência local a cada barra. Desta forma, pode-se separar o problema em duas partes:

- Os dados relativos apenas à própria barra em estudo (seus carregamentos e propriedades de rigidez) são expressos em relação ao sistema de referência local da barra. Com isto, obtém-se uma notação mais simples e compacta.
- Os dados relativos à estrutura como um todo (o conjunto das barras e a solução do sistema completo) são expressos em relação ao sistema de referência global.

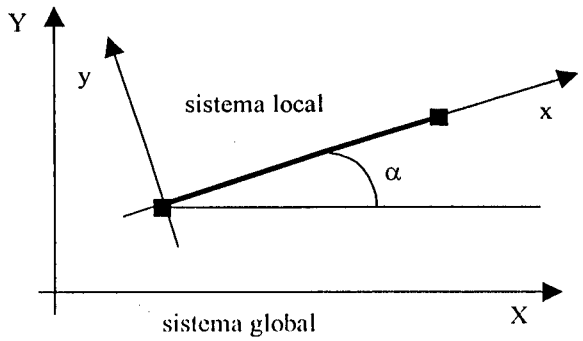


Figura 2.4 – Sistema de coordenadas locais

A partir do ângulo α definido entre o eixo da barra e o sistema de referência global, obtém-se as relações entre um vetor no plano expresso no sistema global e no local. Estas relações podem ser facilmente colocadas sob forma matricial[1].

$$\{s\} = [T].\{S\} \qquad \text{Eq. (2-1)}$$

Onde:

- {s} = vetor expresso no sistema local
- [T] = matriz de transformação de coordenadas
- {S} = vetor expresso no sistema global

Pode-se deduzir facilmente a matriz de transformação, obtendo:

$$[T] = \begin{bmatrix} \cos\alpha & \operatorname{sen}\alpha \\ -\operatorname{sen}\alpha & \cos\alpha \end{bmatrix} \qquad \text{Eq. (2-2)}$$

As mesmas relações podem ser agrupadas de forma a englobar todos os graus de liberdade de uma barra de pórtico plano. Acrescentam-se valores unitários para converter as rotações, que não variam nesta transformação de coordenadas.

$$[T] = \begin{bmatrix} \cos\alpha & \operatorname{sen}\alpha & 0 & 0 & 0 & 0 \\ -\operatorname{sen}\alpha & \cos\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\alpha & \operatorname{sen}\alpha & 0 \\ 0 & 0 & 0 & -\operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Eq. (2-3)}$$

[1] Notações adotadas conforme LORIGGIO (1992).

2.3 - SISTEMATIZAÇÃO DO MÉTODO DOS DESLOCAMENTOS

Neste item, será feita uma breve descrição da metodologia geral de resolução de uma estrutura reticulada através do Método dos Deslocamentos, utilizando uma formulação matricial. Mais detalhes podem ser encontrados em WEAVER & GERE (1965).

Pode-se definir como fases deste processo:

- Discretização da estrutura;
- Obtenção das matrizes de rigidez de cada barra;
- Montagem da matriz de rigidez da estrutura;
- Montagem do vetor de forças;
- Imposição das condições de contorno;
- Resolução do sistema de equações formado;
- Obtenção dos esforços internos.

2.3.1 - Discretização da estrutura

As condições de equilíbrio devem ser satisfeitas em todos os pontos da estrutura, mas, usualmente, são analisadas apenas em certos pontos de interesse, denominados *nós* da estrutura. Como já exposto, para que se possa representar uma estrutura através de elementos de barra, é necessário que seus membros possuam comprimento significativamente maior do que sua seção transversal. Satisfeita esta condição, são inseridos, usualmente, nós nas seguintes posições:

- mudança de direção ou união entre barras;
- alteração de seção transversal da barra;
- ponto com vinculação.

Além disso, podem ser incluídos nós em qualquer outra posição na qual se deseje conhecer, por exemplo, os deslocamentos. Dependendo da implementação adotada, pode ser necessário (ou conveniente) a inclusão de nós nos pontos onde existem cargas aplicadas ou onde ocorra uma variação no carregamento distribuído sobre uma barra.

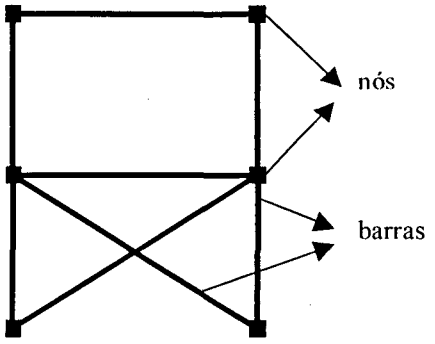


Figura 2.5 – Estrutura de um pórtico plano

2.3.2 - Matriz de rigidez da barra

Uma vez definida a discretização da estrutura, tornam-se conhecidas as coordenadas dos diversos nós e as propriedades das barras definidas. Estas propriedades podem variar de acordo com o tipo de elemento sendo analisado, consistindo, para uma barra de pórtico plano, de:

- L = comprimento;
- I = momento de inércia da seção transversal;
- A = área da seção transversal; e
- E = módulo de elasticidade do material.

A relação entre os esforços internos e os deslocamentos nas extremidades de um elemento de barra são descritas através de um conjunto de equações de equilíbrio, as quais podem ser escritas, na forma matricial, como sendo:

$$\{S\} = [r] \cdot \{d\} \qquad \text{Eq. (2-4)}$$

Onde:

- $\{S\}$ = vetor de esforços internos
- $\{d\}$ = vetor de deslocamentos
- $[r]$ = matriz de rigidez

A matriz de rigidez de um elemento de barra é função apenas de suas propriedades físicas e geométricas, variando em função do tipo de elemento (ou seja, do número de deslocamentos nodais considerados). Pode-se provar que esta matriz é sempre simétrica (devido à simetria entre o nó final e o inicial da barra).

A matriz de rigidez de uma barra de pórtico plano apresenta-se como seguinte:

$$[r]=\begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \qquad \text{Eq. (2-5)}$$

Esta matriz é referenciada a um sistema de coordenadas coincidente com seu eixo centroidal. É útil, contudo, referenciar-se a barra em um sistema global, a fim de facilitar a resolução da estrutura como um todo. Esta transformação de coordenadas pode ser facilmente realizada através da Álgebra Matricial, utilizando-se a matriz [T] de transformação, definida no item 2.2.4, da seguinte forma:

$$[r] = [T]^T . [r] . [T] \qquad \text{Eq. (2-6)}$$

2.3.3 - Matriz de rigidez da estrutura

Assim como a relação entre os esforços e os deslocamentos, através da definição da matriz de rigidez, é válida para um elemento de barra tomado isoladamente, também se aplica à estrutura como um todo. Desta forma, o vetor de deslocamentos {δ} é composto por todos os deslocamentos pertencentes à estrutura e o vetor de esforços é composto pelos carregamentos externos aplicados em cada nó e pelas reações de apoio nos nós vinculados.

A matriz de rigidez é obtida através da superposição das matrizes de rigidez das barras que a compõem, nas posições onde estas se inserem.

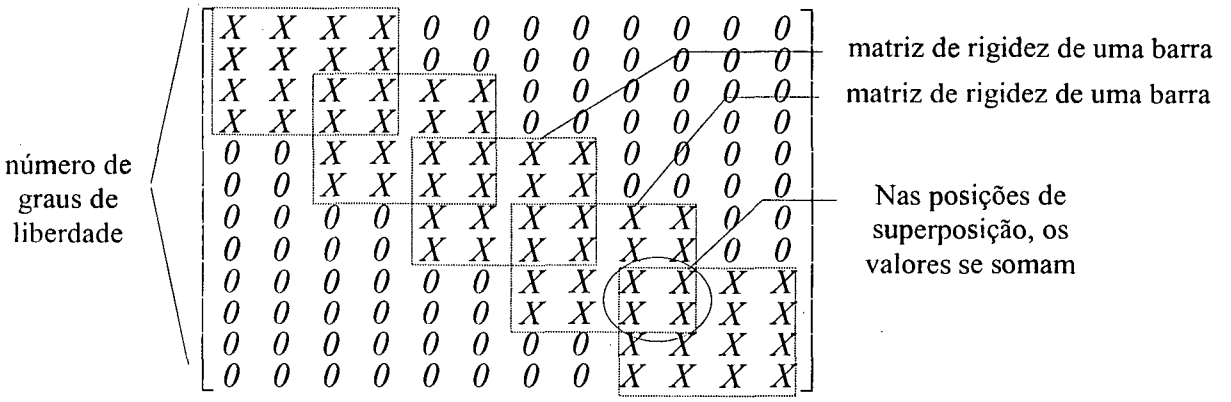


Figura 2.6 – Montagem da matriz de rigidez global

Pode-se provar que esta matriz também é simétrica e, mais ainda, que é uma matriz do tipo *banda*. Isto significa que todos os seus elementos não-nulos concentram-se em uma banda simétrica ao longo da diagonal principal da matriz de rigidez. Desta forma, pode-se armazenar apenas os valores relevantes, em uma matriz com número de colunas igual à sua largura de banda.

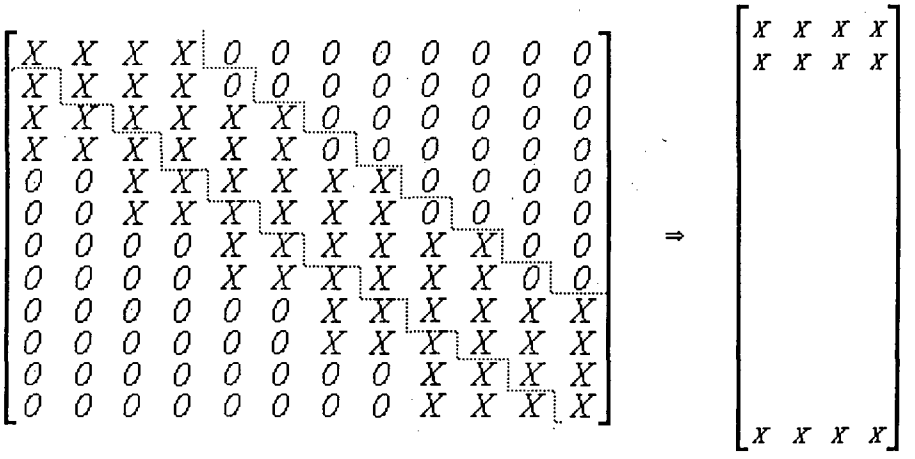


Figura 2.7 – Característica de uma matriz banda

Uma vez que apenas a semi-banda da matriz é armazenada, e levando em consideração o fato de que a matriz de rigidez da barra também é simétrica, pode-se montar a matriz da estrutura também apenas com os valores da banda superior da matriz da barra.

A matriz de rigidez de estrutura é indicada por [K] e forma a mesma equação de equilíbrio em torno dos nós:

$$\{F\} = [K] \cdot \{\delta\} \qquad \text{Eq. (2-7)}$$

Onde

- $\{F\}$ = vetor de forças nodais externas
- $\{\delta\}$ = vetor de deslocamentos nodais
- $[K]$ = matriz de rigidez global

2.3.4 - Vetor de forças

O vetor de forças $\{F\}$ é composto pelos esforços resultantes em cada um dos graus de liberdade da estrutura (sejam estas cargas aplicadas ou reações de apoio). A primeira parcela deste vetor é formada pelo conjunto de carregamentos nodais externos *{Fi}*. Estes carregamentos são usualmente referidos ao sistema de referência global. Considerando que cada nó da estrutura possui três graus de liberdade, utilizam-se:

- $Fi_{(3i-2)} = Fx_i$ = carga concentrada aplicada no nó "i", paralela ao eixo global X
- $Fi_{(3i-1)} = Fy_i$ = carga concentrada aplicada no nó "i", paralela ao eixo global Y
- $Fi_{(3i)} = Mz_i$ = momento fletor aplicado no nó "i", com sinal positivo conforme convenção definida em 2.2.3

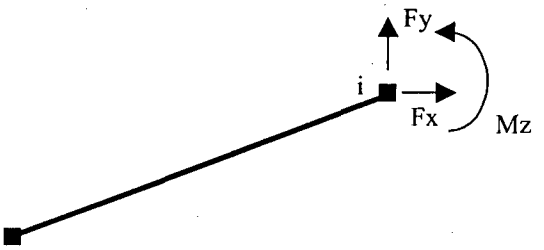


Figura 2.8 – Carregamentos nodais externos

Na formulação matricial, todos os efeitos (ações e deslocamentos) são concentrados nos nós da estrutura. No caso de existirem carregamentos distribuídos sobre as barras, é necessário definir esforços nodais equivalentes. Tais esforços possuem valores tais que provocam os mesmos deslocamentos obtidos com o carregamento original. Isto pode feito considerando-se a barra engastada em suas extremidades, com o carregamento aplicado.

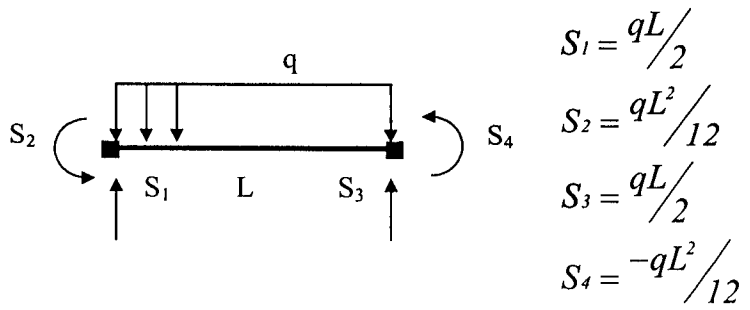


Figura 2.9 – Esforços de imobilização para uma carga uniformemente distribuída (elemento de viga)

No caso de um carregamento uniformemente distribuído sobre uma barra de pórtico plano, obtém-se o seguinte vetor:

$$\{Fo\} = \begin{bmatrix} 0 \\ \frac{qL}{2} \\ \frac{ql^2}{12} \\ 0 \\ \frac{qL}{2} \\ -\frac{qL^2}{12} \end{bmatrix} \qquad \text{Eq. (2-8)}$$

Os esforços resultantes são os esforços a serem aplicados, com sinal invertido, compondo a segunda parcela do vetor *{F}*, o vetor *{Fo}*. Este vetor é formado pela soma dos diversos vetores *{fo}* de cada barra, já transformados para o sistema global através das relações definidas em 2.2.4. Desta forma, o vetor {F} será composto por:

$$\{F\} = \{Fi\} - \{Fo\} \qquad \text{Eq. (2-9)}$$

Dependendo do caso, podem ser incluídas outras componentes no cálculo de {F}. Estas são as representações, em termos de esforços, de outros efeitos, como o da variação de temperatura, deslocamentos prescritos ou protensão nas barras.

2.3.5 - Condições de contorno

Pode-se comprovar que a matriz de rigidez global da estrutura é singular, ou seja, é uma matriz que não admite inversão. Desta forma, o sistema de equações que ela define é indeterminado. Isto ocorre porque não foi aplicada à estrutura qualquer espécie de restrição com relação aos deslocamentos de corpo rígido. Para que se possa obter uma solução para o sistema, é necessário que se estabeleça um número mínimo de vinculações, aplicadas sobre o sistema, de forma a torná-lo determinado. Tal operação se traduz em separar para a solução do sistema apenas aqueles deslocamentos (graus de liberdade) cujo valor seja desconhecido, retirando-se da matriz aqueles aos quais seja imposto um valor (normalmente, nulo), devido a vinculações.

Este procedimento pode ser feito de várias maneiras. A solução adotada neste trabalho é a de reordenar os graus de liberdade presentes no sistema, colocando todos graus restringidos no final. Obtém-se, portanto, a divisão da estrutura em duas partes:

- parte A: graus de liberdade livres, nos quais são conhecidos os esforços aplicados e desejam-se obter os deslocamentos
- parte B: graus de liberdade restringidos, nos quais são conhecidos os deslocamentos (usualmente nulos) e desejam-se obter as reações de apoio

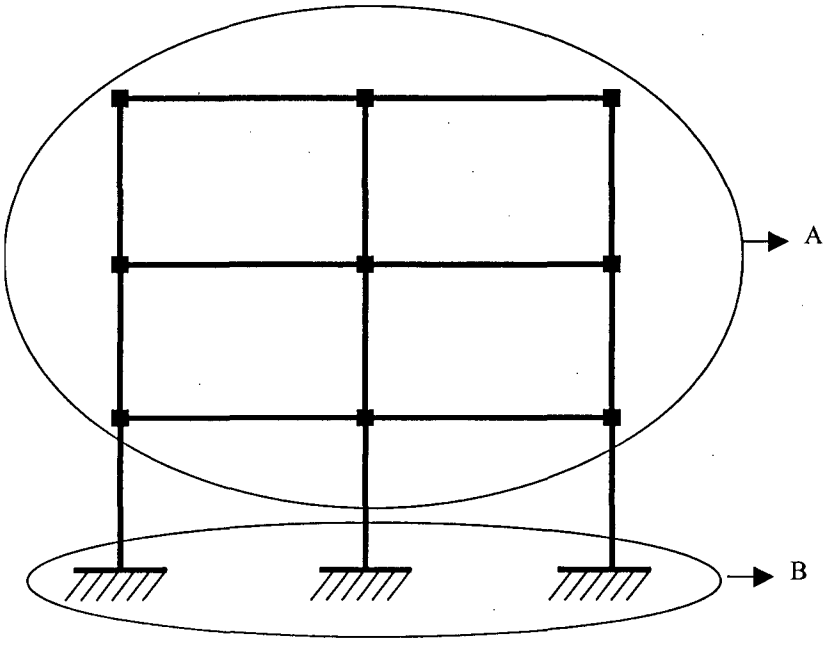


Figura 2.10 – Divisão da estrutura em parte livre e parte restringida

A matriz de rigidez [K], uma vez reordenada, pode ser expressa na forma de submatrizes:

$$[K]=\begin{bmatrix} Kaa & Kab \\ Kba & Kbb \end{bmatrix} \qquad \text{Eq. (2-10)}$$

Aplicando-se a mesma reordenação aos vetores de forças {F} e deslocamentos {δ}, pode-se reescrever a Eq. (2-7) da seguinte forma:

$$\begin{bmatrix} Fa \\ Fb \end{bmatrix}=\begin{bmatrix} Kaa & Kab \\ Kba & Kbb \end{bmatrix}\cdot\begin{bmatrix} \delta a \\ \delta b \end{bmatrix} \qquad \text{Eq. (2-11)}$$

Aplicando-se as propriedades definidas na Álgebra Matricial, pode-se obter os deslocamentos conhecidos através da seguinte equação modificada:

$$\{Fa\} = [Kaa].\{\delta a\} + [Kab].\{\delta b\} \qquad \text{Eq. (2-12)}$$

No caso de se definirem como nulos todos os deslocamentos restringidos, a equação acima se reduz a:

$$\{Fa\} = [Kaa].\{\delta a\} \qquad \text{Eq. (2-13)}$$

Onde:

- {Fa} = vetor das forças nodais referentes aos deslocamentos livres;
- [Kaa] = parte da matriz de rigidez referente aos deslocamentos livres;
- {δa} = deslocamentos desconhecidos.

Pode-se demonstrar que a submatriz [Kaa] mantém as características de simetria e banda apresentadas pela matriz [K]. Todavia, caso sejam especificadas condições de contorno suficientes, esta deixará de ser singular, tornando o sistema determinado.

2.3.6 - Resolução do sistema

Como já exposto, a resolução da estrutura baseia-se na seguinte equação de equilíbrio:

$$\{F\} = [K] \,.\, \{\delta\} \qquad \text{Eq. (2-7)}$$

Visto esta equação estar escrita na forma matricial e o fato de que a grandeza que se deseja conhecer são os deslocamentos, expressos por *{δ}*, a equação acima se traduz em um sistema de equações lineares, cuja solução deve ser encontrada. Isto pode ser feito através de diversos processos já conhecidos, como, por exemplo, o processo da eliminação de Gauss, o processo de Cholesky e o processo da solução frontal.

Devido ao fato da matriz de rigidez *[K]* ser do tipo *banda simétrica*, é interessante aproveitar esta característica, a fim de se economizar espaço de armazenamento e tempo de processamento. Desta forma, como apenas os valores em torno da diagonal principal possuem valores não-nulos, pode-se armazenar apenas esta semi-banda e, igualmente, processar apenas estes valores durante a solução do sistema. Neste programa, foi adotado, para a solução do sistema, o *Método de Cholesky*. Mais detalhes sobre este processo podem ser obtidos em WEAVER & GERE (1965).

2.3.7 - Cálculo dos esforços internos

Uma vez obtidos os deslocamentos nodais, através da resolução do sistema acima citado, pode-se aplicar novamente a equação de equilíbrio para cada barra, agora conhecidos os valores de seus deslocamentos nodais *{δ}*, a fim de se obter seus valores de esforços internos *{S}*.

$$\{S\} - \{S0\} = [r] \cdot \{\delta\} \qquad \text{Eq. (2-14)}$$

Desta vez, trata-se apenas de uma simples operação de multiplicação de matrizes. Após esta, deve-se calcular os valores dos esforços referenciados ao sistema local da barra, usando a mesma regra de transformação já citada.

2.4 – CONSIDERAÇÕES SOBRE A APLICABILIDADE DA ANÁLISE LINEAR

A Análise Linear, conforme exposta no decorrer deste capítulo, considera comportamento elástico-linear para os materiais. Considera-se, usualmente, no caso de estruturas de concreto armado, a seção bruta de concreto, sem consideração da armadura. Conforme CORRÊA & RAMALHO (1993), "Os resultados de uma análise linear são usualmente empregados para a verificação de Estados Limites de Serviço. É possível estender os resultados para verificações de Estado Limite Último, mesmo com altas tensões, desde que se garanta a ductilidade".

2.4.1 Não linearidade física

Embora assuma-se um comportamento elástico-linear para os materiais, isto absolutamente não é verdade para o caso do concreto armado. Em primeiro lugar, mesmo desprezando-se a armadura, o diagrama tensão-deformação do concreto, além de não ser linear, vale apenas para tensões de compressão, sendo que a resistência à tração é tão baixa que pode ser usualmente desprezada.

O diagrama tensão-deformação convencional do concreto à compressão é dado no item 8.2.4 da NBR 6118/78 e define-se através de uma parábola com início na origem e pico a uma deformação de 0.2% e de um patamar de 0.85 fcd para deformações superiores a 0.2%.

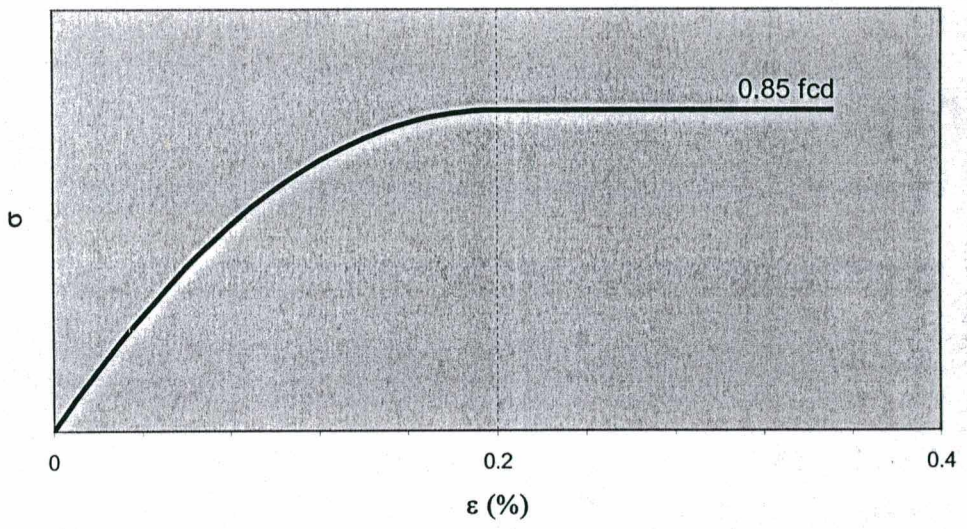


Figura 2.11 – Diagrama tensão-deformação do concreto (NBR 6118)

Este diagrama aplica-se à análise no estado limite último. Para a análise linear, pode-se assumir valores característicos. Segundo a NBR 6118/78, item 3.2.2.1, o cálculo das estruturas em regime elástico pode ser feito utilizando-se o módulo de elasticidade definido em seu item 8.2.5:

$$Ect = 6600 \sqrt{fcj} \text{ (Mpa)} \qquad \text{Eq. (2-15)}$$

$$fcj = fck + 3{,}5 \text{ MPa}$$

A norma completa, em seu item 4.2.3.1-A, afirmando que, desde que a deformação lenta seja nula ou desprezível (ou seja, no caso de ações de curta duração), o módulo de elasticidade Ec a adotar será o módulo secante do concreto, suposto igual a 0,9 do módulo secante definido na equação acima. Fica, portanto:

$$Ec = 5940 \sqrt{fck+3,5} \ (Mpa) \qquad \text{Eq. (2-16)}$$

Representando-se graficamente a equação acima e comparando-a com os diagramas tensão-deformação do concreto característico e de cálculo, para um valor de fck igual a 20 Mpa, tem-se:

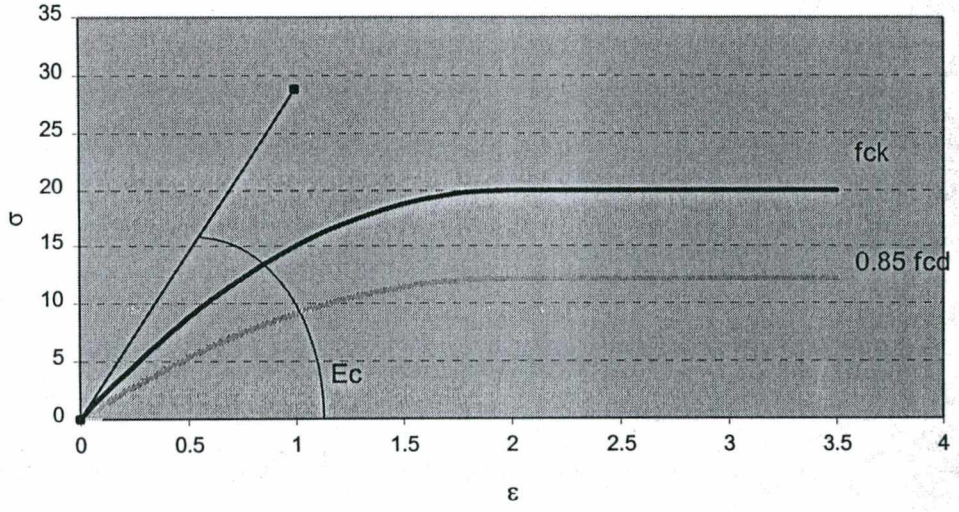


Figura 2.12 – Módulo de elasticidade secante do concreto

Pode-se notar também que, mesmo para valores inferiores a 50% de *fck*, já existe diferença considerável entre a tensão obtida através do diagrama σ-ε assumido para o concreto e o valor obtido pela hipótese linear[2]. Todavia, considerando que este diagrama é apenas convencional, não se caracteriza como sendo grande fonte de erro.

[2] Pode-se observar, a partir da figura, que esta expressão fornece valores elevados de Ec, mesmo para tensões em serviço, em comparação com valores experimentais e normas internacionais. Existem propostas de revisão deste valor para inclusão na nova NBR 6118.

A dificuldade maior, no tocante à representação do material, além da representação da armadura, é a fissuração na seção, que leva a um diagrama de tensões no concreto válido apenas para a parte comprimida e à solicitação da armadura tracionada.

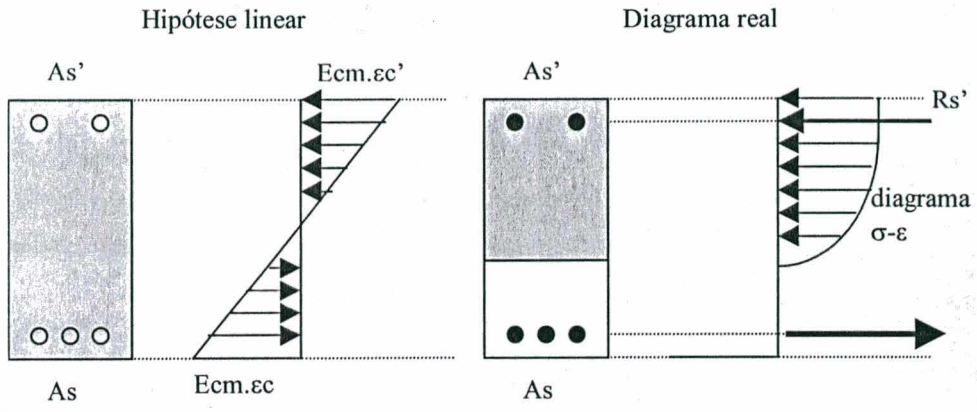


Figura 2.13 – Seção de concreto fissurada

2.4.2 Não linearidade geométrica

"Sob a ação de cargas verticais e horizontais, os nós da estrutura deslocam-se horizontalmente. Os esforços de segunda ordem decorrentes desses deslocamentos são chamados efeitos globais de 2ª ordem". "Os efeitos de 2ª ordem, em cuja determinação deve ser levado em conta o comportamento não linear dos materiais, podem ser desprezados sempre que não representem acréscimo superior a 10% nas reações e solicitações relevantes na estrutura". Esta preocupação, expressa por SANTOS & FRANCO (1993) e que, segundo os autores, deve constar da nova NBR 6118, limita a aplicação da análise de 1ª ordem conforme exposta até aqui.

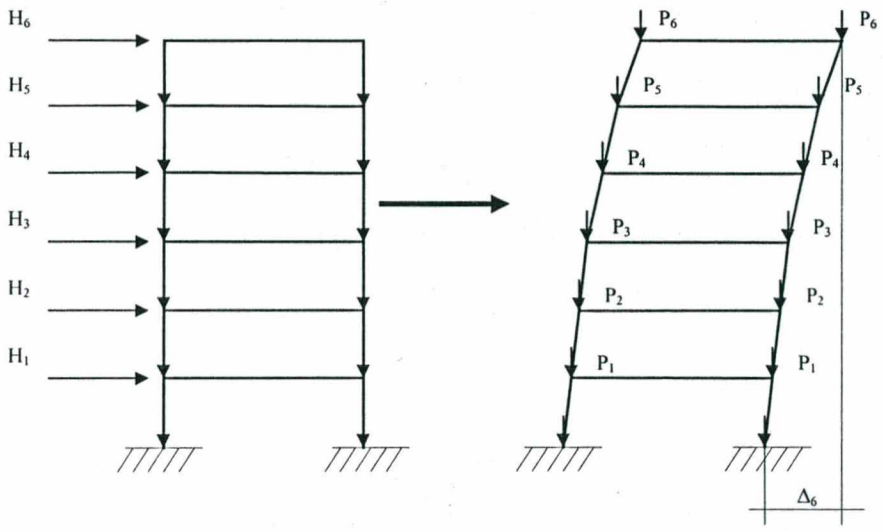


Figura 2.14 – Configuração deformada de uma estrutura

A determinação dos efeitos de 2ª ordem é assunto do capítulo a seguir, mas é interessante abordar as considerações simplificadas disponíveis para a validação da análise de 1ª ordem. Objetiva-se concluir, como base unicamente nos resultados da análise de 1ª ordem, se os efeitos de 2ª ordem são relevantes ou não. Para isto, SANTOS & FRANCO (1993) disponibilizam dois processos: o Parâmetro Alfa e o Coeficiente Gama-Z. Segundo os autores, a nova NBR 6118 aceitará ambos os processos.

2.4.3 Parâmetro de Instabilidade Alfa (α)

O parâmetro Alfa é definido por SANTOS & FRANCO (1993) da seguinte forma: "Uma estrutura reticulada poderá ser considerada de nós fixos[3] se seu parâmetro de instabilidade α for menor que o valor α1 definido a seguir".

$$\alpha = H_{tot}\sqrt{\frac{N_k}{E_c I_c}} \qquad \text{Eq. (2-17)}$$

$$\alpha 1 = \begin{cases} 0.2 + 0.1 \cdot n & \text{para } n \leq 3 \\ 0.6 & \text{para } n \geq 4 \end{cases} \qquad \text{Eq. (2-18)}$$

Onde:

[3] Uma estrutura é considerada de "nós fixos" quando os efeitos de 2ª ordem pode ser desprezados.

- n = número de níveis de barras horizontais (andares) acima da fundação ou de um nível pouco deslocável do subsolo;
- H_{tot} = altura total da estrutura, medida a partir do topo da fundação ou de um nível pouco deslocável do subsolo;
- N_k = somatória de todas as cargas verticais atuantes na estrutura (a partir do nível considerado para o cálculo de H_{tot}), com seu valor característico;
- E_cI_c = somatória das rigidezes de todos os pilares na direção considerada. No caso de estruturas de pórticos, treliças ou mistas, permite-se considerar produto de rigidez E_cI_c de um pilar equivalente de seção constante.

Para determinar a rigidez equivalente E_cI_c, procede-se da seguinte forma:

- calcula-se o deslocamento no topo da estrutura, sob a ação do carregamento horizontal característico;
- calcula-se a rigidez de um pilar equivalente de seção constante, engastado na base e livre no topo, de mesma altura H_{tot}, tal que, sob a ação do mesmo carregamento, sofra o mesmo deslocamento no topo.

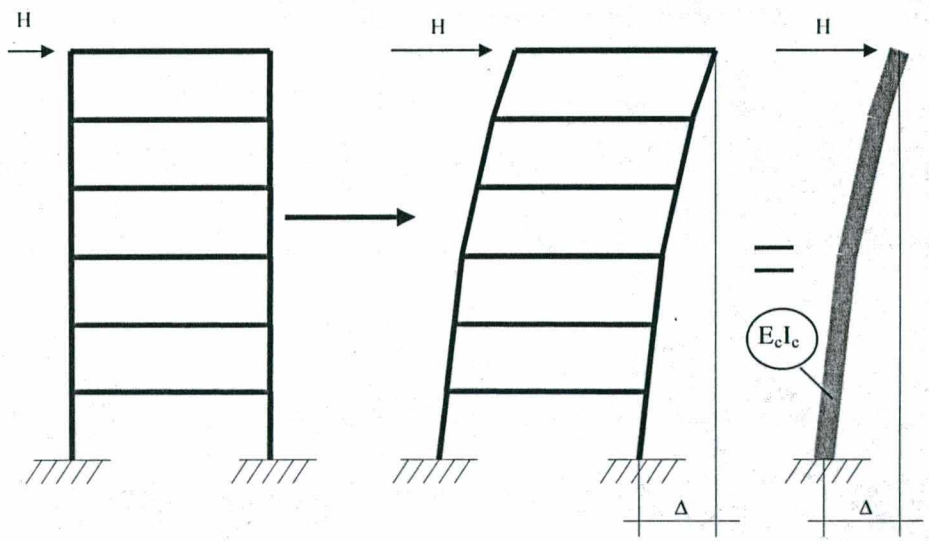


Figura 2.15 – Pilar equivalente a um pórtico plano

O parâmetro Alfa, em termos de verificação da estabilidade global de uma edificação, pode ser considerado o processo mais conhecido pelos projetistas de estruturas. O critério, adotado pela norma do CEB, consiste ainda hoje na forma mais utilizada da avaliação dos efeitos de 2ª ordem em uma estrutura de concreto armado. A isto se soma o fato de que a bibliografia nacional, em sua maior parte, apresenta este como critério de avaliação da estabilidade global.

A dedução da expressão para cálculo de α não é objetivo deste trabalho, sendo que as indicações essenciais estão apresentadas em SUSSEKIND (1982). É interessante apontar apenas que o valor limite (0.6 para $n > 3$) é obtido a partir da consideração $M_{(2a\ ordem)} = 10\%\ M_{(1a\ ordem)}$. A não linearidade física é levada em conta de maneira simplificada através da redução do módulo de rigidez E_cI_c por um fator 0.7.

2.4.4 Coeficiente Gama-Z (γz)

O coeficiente γz é definido por SANTOS & FRANCO (1993) da seguinte forma: "É possível determinar de forma aproximada o coeficiente γz de majoração dos esforços globais finais com relação aos de primeira ordem". O valor de γz é definido por:

$$\gamma z = \frac{1}{1 - \frac{\Delta M_{tot,d}}{M1_{tot,d}}} \qquad \text{Eq. (2-19)}$$

Onde:

- $M1_{tot,d}$ = momento de tombamento, ou seja, a soma dos momentos de todas as forças horizontais, com seus valores de cálculo, em relação à base da estrutura;
- $\Delta M_{tot,d}$ = soma dos produtos de todas as forças verticais atuantes na estrutura, com seus valores de cálculo, pelos deslocamentos horizontais de seus respectivos pontos de aplicação, obtidos da análise de 1ª ordem.

Uma vez que o valor de γz representa o próprio efeito de 2ª ordem, deve-se satisfazer à condição $\gamma z \leq 1.1$.

2.4.5 Considerações

O uso dos critérios simplificados para avaliação dos efeitos de 2ª ordem deve ser pouco a pouco substituído por processos que considerem a não linearidade geométrica da estrutura. Este é o assunto do capítulo a seguir. Todavia, alguns comentários devem ser feitos.

O parâmetro Alfa, embora seja correntemente o mais utilizado, pode ser considerado de formulação inferior ao coeficiente Gama-Z. Um dos problemas que este apresenta é a representação deficiente de estruturas cuja rigidez varia ao longo de sua altura.

O parâmetro Alfa correlaciona apenas o deslocamento obtido no topo com um pilar equivalente. Em diversos casos, deve-se analisar a estrutura que está sendo submetida à verificação de estabilidade para verificar se o resultado será aceitável ou não. Um problema usual é o de se ter, acima do "corpo" da estrutura, mais alguns pavimentos para casa de máquinas, reservatório superior, entre outros. Como estes pavimentos são usualmente pouco rígidos, tendem a comprometer os resultados.

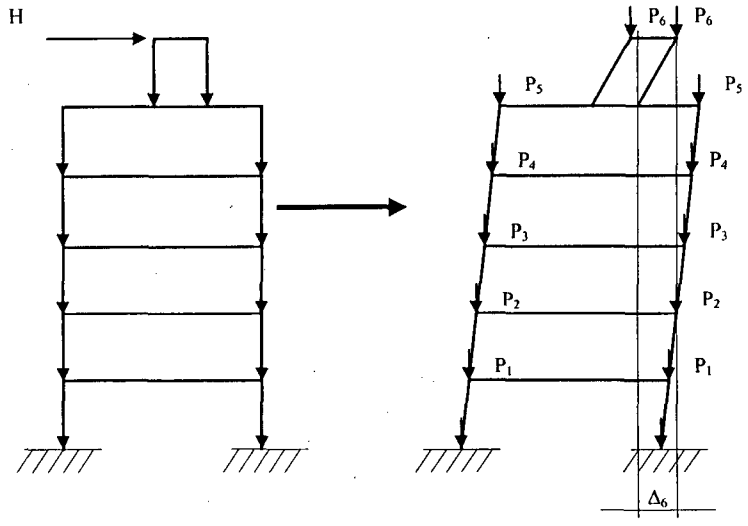


Figura 2.16 – Problema na obtenção do parâmetro Alfa

O coeficiente Gama-Z não apresenta este problema específico com tanta evidência. Isto porque a influência de cada parte da estrutura é tomada isoladamente, para depois ser computado o somatório.

Outro aspecto diz respeito ao fato de se fazer a analogia com um pilar engastado na base e livre no topo. A configuração deformada deste pilar, bastante conhecida, é normalmente muito diferente da configuração de deslocamentos apresentada por uma estrutura real.

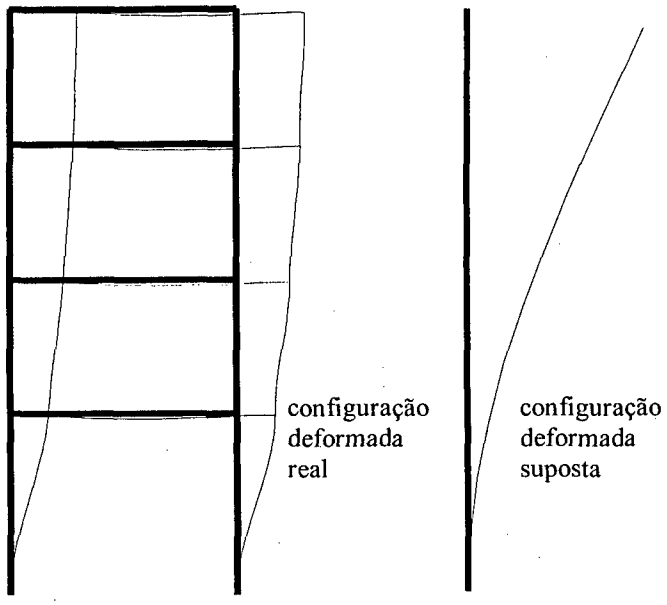


Figura 2.17 – Comparação entre a deformada real e um pilar engastado na base e livre no topo

A simples utilização do deslocamento no topo para associar as duas situações pode ser considerada muito simplista, fazendo com que os resultados divirjam mais do real do que os resultados apresentados com o uso do Gama-Z. No capítulo a seguir, após apresentar os métodos para consideração dos efeitos de 2ª ordem no cálculo da estrutura, será possível fazer uma verificação numérica destes coeficientes apresentados. Analisando apenas ambas as formulações, pode-se indicar o uso do Gama-Z na falta de consideração mais exata da não linearidade geométrica.

2.4.6 Outras fontes de não linearidade

A maior parte dos estudos a respeito da estabilidade de estruturas reticuladas centra sua pesquisa no efeito causado pelo deslocamento horizontal dos nós (o efeito P-Delta, abordado usualmente como "efeitos globais de 2ª ordem") e a influência da força axial na rigidez das barras. A consideração da não linearidade física é importante, mas sua complexidade faz com que seja usualmente tratada de maneira simplificada. Existem, contudo, outros fatores não cobertos pela Análise Linear e que podem levar à redução da capacidade portante da estrutura. Estes fatores foram reunidos por BIRNSTIEL & IFFLAND (1980) e apresentam-se como a seguir.

Efeitos geométricos

Listam-se aqui os efeitos concernentes à geometria considerada para a estrutura:

- *Efeito dos deslocamentos horizontais dos nós*: Este efeito, normalmente conhecido como Efeito P-Delta (P-Δ), corresponde ao "efeito global de 2ª ordem" conforme abordado por SANTOS & FRANCO (1993) e como deve ser colocado na nova NBR 6118. Basicamente, o problema refere-se ao fato de que, após a análise de 1ª ordem, os pontos de aplicação das cargas alteram-se de posição, criando momentos de 2ª ordem. Uma nova análise seria necessária, com base na nova configuração geométrica da estrutura, sendo esta repetida iterativamente até a convergência.
- *Influência da força axial na rigidez à flexão de cada elemento*: O efeito de uma força axial atuando em um elemento de barra é o de reduzir sua rigidez caso a força seja de compressão, ou o oposto, no caso de tração. No caso da rigidez à flexão, uma força de compressão reduz o momento necessário para rotacionar a extremidade da barra, devido ao aumento dos momentos fletores ao longo do elemento.

Estes dois primeiros efeitos são considerados os mais importantes e serão tratados no decorrer deste trabalho.

- *Imperfeições geométricas globais*: Em uma estrutura real, as peças não são construídas exatamente nas posições geometricamente previstas na análise. Das imperfeições, a mais importante refere-se ao desaprumo dos pilares acumulado ao longo da altura total da edificação, levando a um desaprumo da estrutura como um todo. A distribuição da magnitude e direção destas imperfeições pode ser considerada aleatória ao longo da estrutura. Uma abordagem padronizada deve ser efetuada, todavia, para levar em conta este efeito do cálculo da estrutura.

MARTINS & STUCCHI (1993) apontam esta preocupação ao definir que "Na verificação do estado limite último das estruturas reticuladas, devem ser consideradas as imperfeições geométricas do eixo das peças da estrutura descarregada". O critério utilizado, segundo os autores, deve ser incluído na nova NBR 6118 e equivale ao preconizado pelo MC-90, que define um ângulo de desvio no alinhamento dos pilares com base na altura do elemento e no número de elementos verticais considerado. Na análise global da estrutura, deve ser considerado um desaprumo global dos elementos verticais, como na figura a seguir:

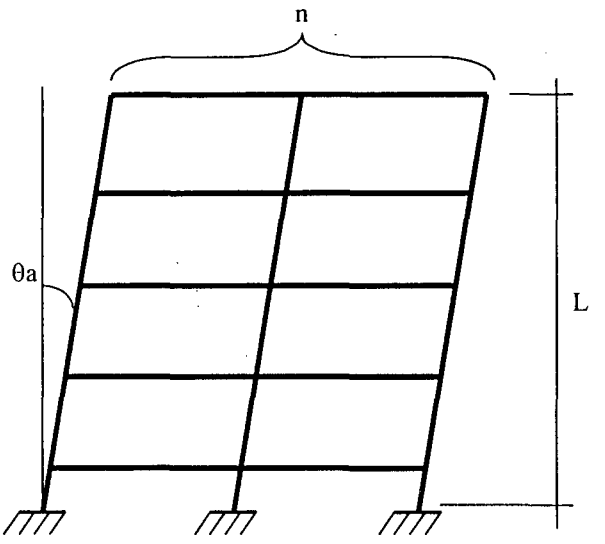


Figura 2.18 – Imperfeições geométricas globais

Define-se o valor de θ_a da seguinte forma:

$$\theta_1 = \frac{1}{100\sqrt{L}} \geq \theta 1_{min} \qquad \text{Eq. (2-20)}$$

$$\theta_a = \theta_1 \sqrt{\frac{1+1/n}{2}} \qquad \text{Eq. (2-21)}$$

Onde:

- L = altura total da estrutura em metros;
- n = número total de elementos verticais contínuos;
- $\theta 1_{min}$ = 1/400 para estruturas de nós fixos e 1/300 para estruturas de nós móveis.

Conforme colocado por MARTINS & STUCCHI (1993), permite-se, simplificadamente, substituir as imperfeições geométricas por conjuntos de cargas nodais auto-equilibradas, como mostra a figura a seguir:

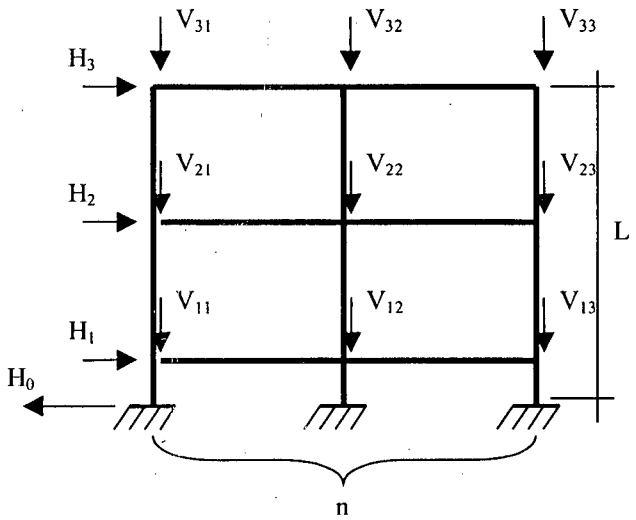


Figura 2.19 – Carregamentos nodais equivalentes a um desaprumo global

As cargas horizontais em cada andar são obtidas por:

$$H_i = \sum_{j=1}^{n} V_{ij}\theta_a \qquad \text{Eq. (2-22)}$$

$$H_0 = \sum_{i=1}^{m} H_i \qquad \text{Eq. (2-23)}$$

Onde:

- V_{ij} = carga vertical aplicada pelo andar "i" ao pilar "j".

Uma vez que o valor de θa é constante e relativo a uma condição inicial da estrutura, os carregamentos equivalentes podem ser obtidos mesmo antes da análise de 1ª ordem, acrescentando-os manualmente aos carregamentos horizontais previstos no modelo. Uma abordagem sistematizada seria a de efetuar a análise de 1ª ordem e aplicar o mesmo procedimento às barras da estrutura, com base na carga axial obtida e no valor de θa fornecido como parâmetro do problema. Isto não será feito no presente trabalho, mas não apresenta qualquer dificuldade excepcional.

Existem outras abordagens para o problema e um estudo sobre estas foi efetuado por KIM & CHEN (1996-b). Os autores citam três procedimentos disponíveis para inclusão das imperfeições geométricas: o método das imperfeições explícitas (que inclui a imperfeição diretamente na geometria do modelo), o método das cargas equivalentes (equivale ao processo descrito) e o método do módulo de deformação reduzido. Segundo os autores, as diferenças encontradas são desprezíveis, confirmando a aplicabilidade do processo descrito por MARTINS & STUCCHI (1993).

- *Imperfeições geométricas localizadas*: Além do efeito de desaprumo global, cada peça pode ter, isoladamente, efeitos secundários localizados, devidos a imperfeições na execução. Incluem-se aqui, principalmente, a falta de retinilidade (curvatura inicial) e o desaprumo de um lance de pilar em relação aos lances adjacentes.

MARTINS & STUCCHI (1993) apontam que "No caso de verificação de um lance de pilar, deve ser considerado o efeito do desaprumo ou da falta de retinilidade do eixo do pilar". O ângulo de desvio considerado é o mesmo θ_1 definido na Eq. (2-20), alterando-se apenas o valor de $\theta 1_{min}$ para 1/300.

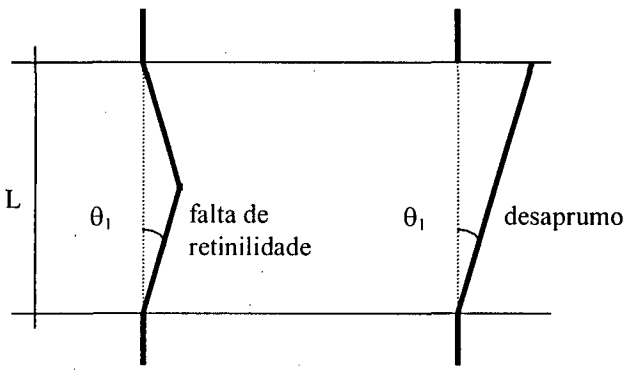


Figura 2.20 – Imperfeições geométricas localizadas

Da mesma forma como para as imperfeições globais, os autores afirmam que permite-se, simplificadamente, substituir as imperfeições geométricas por conjuntos de cargas nodais auto-equilibradas, como mostra a figura a seguir:

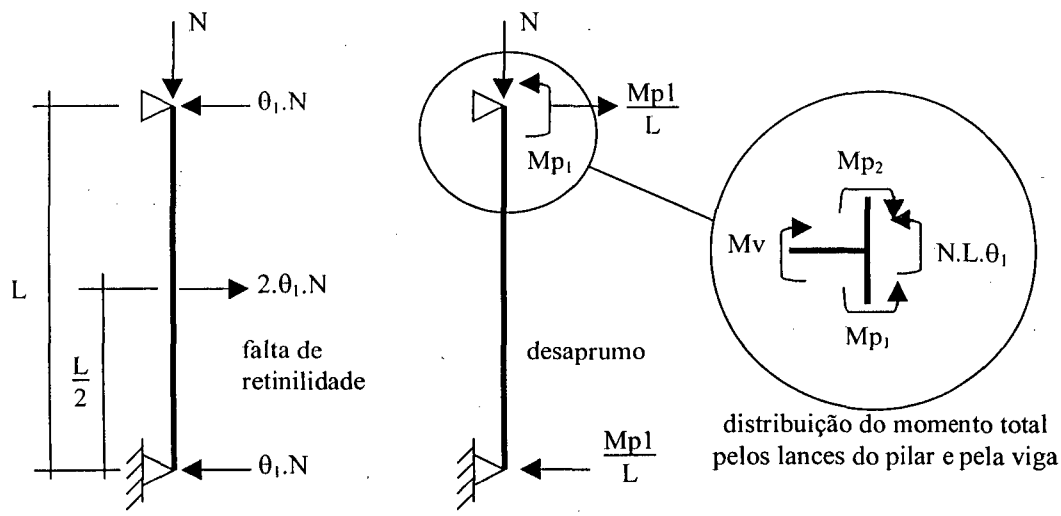


Figura 2.21 – Cargas nodais equivalentes a um desaprumo localizado

Encontram-se, na proposta de revisão da NBR 6118, algumas considerações referentes às imperfeições localizadas:

- "Admite-se que, nos casos usuais, a consideração da falta de retinilidade seja suficiente" (MARTINS & STUCCHI, 1993);
- "Nas estruturas de nós fixos, permite-se considerar cada elemento comprimido isoladamente, como barra vinculada nas extremidades aos demais elementos estruturais que ali concorrem, onde se aplicam os esforços obtidos pela análise da estrutura efetuada segundo a teoria de 1ª ordem" (SANTOS & FRANCO, 1993);
- "Em estruturas de nós móveis, (...), a análise global de 2ª ordem fornecerá apenas os esforços nas extremidades das barras, devendo então ser realizada uma análise dos efeitos locais de 2ª ordem (...). Os elementos isolados, para fins de verificação local, serão formados pelas barras comprimidas retiradas da estrutura, de mesma forma que em 15.5" (SANTOS & FRANCO, 1993).

Estas considerações permitem a consideração separada dos efeitos locais de 2ª ordem e, da mesma forma, das imperfeições geométricas localizadas. A abordagem decorrente disto é a definição de uma "excentricidade acidental" considerada disposta apenas na parte central do pilar, com o valor:

$$e_a = \theta_1 . \frac{L}{2} \qquad \text{Eq. (2-24)}$$

A consideração das imperfeições localizadas pode ser feita logo após a análise de 1ª ordem, uma vez que é necessário conhecer a carga normal atuante em cada barra. Isto difere das imperfeições globais, que podem ser modeladas com base nas cargas externas. Isto não será feito no presente trabalho, mas não apresenta qualquer dificuldade excepcional. Conforme já exposto, existem outras abordagens para o problema e um estudo sobre estas foi efetuado por KIM & CHEN (1996-a).

- *Efeito das deformações por cortante*: O efeito das deformações por cortante é normalmente muito pequeno, mas pode ganhar vulto na presença de vigas parede e consolos curtos na estrutura. A consideração destas deformações pode ser feita facilmente na dedução da matriz de rigidez da barra (vide 2.3.2). Certos coeficientes de rigidez são menores porque as deformações por cortante reduzem os esforços provocados pelos deslocamentos unitários. A dedução da matriz de rigidez para uma barra de pórtico plano, considerando as deformações por cortante (mas desprezando a força axial), é bastante conhecida e pode ser encontrada, por exemplo, em WEAVER & GERE (1965).

Não é divulgado, no momento, nenhum estudo contendo a consideração conjunta de ambos os fatores (cortante e força axial). Nos procedimentos que serão apresentados no capítulo a seguir, será desprezado o efeito das deformações por cortante.

- *Efeito de distorções localizadas*: Distorções locais após o carregamento, como, por exemplo, flambagem localizada de almas de vigas I ou flambagem lateral por flexo-torção, podem afetar a capacidade portante da peça, mas são difíceis de serem modeladas matematicamente.

Os efeitos localizados são bastante raros nas estruturas de concreto armado, razão esta pela qual o presente estudo não irá abordá-los. No caso de estruturas de aço, estes efeitos tornam-se mais relevantes, sendo definidos alguns procedimentos simplificados para limitar as dimensões da peça de forma a que estes não venham a ocorrer ou para definir um fator de redução da capacidade portante da peça no dimensionamento. Mais detalhes podem ser encontrados na NBR 8800 ("Projeto e execução de estruturas de aço de edifícios").

Efeitos físicos

Os efeitos físicos referem-se à não linearidade entre tensões e deformações apresentada pelas peças da estrutura.

- *Não linearidade física*: O fato de que a relação entre tensões e deformações (diagrama σ-ε) altera os resultados obtidos através de uma análise linear foi discutido no item 2.4.1.
- *Tensões residuais*: Tensões residuais provenientes do processo de fabricação são um problema apresentado pelas estruturas metálicas. Estas influenciam a rigidez do elemento e podem ser levadas em conta na formulação da matriz de rigidez da barra. Em termos de concreto armado, outros fatores intervenientes são a retração e a deformação lenta, cuja inclusão nos modelos também pode ser considerada bastante complexa.

Efeitos do carregamento

Caso a suposição de carregamento monotônico admitida usualmente não seja obedecida, a não linearidade da estrutura faz com que a história de carregamentos passe a ser um fator relevante na verificação da estrutura.

- *Carregamento não proporcional*: A ordem de aplicação dos diversos carregamentos verticais e horizontais da estrutura ("história do carregamento") não tem influência sobre uma análise linear, devido ao princípio da superposição das ações. Quando a análise não é linear, todavia, isto deixa de ser válido e este problema pode se tornar relevante.
- *Carregamento variável*: A não linearidade física dos materiais pode levar a estrutura a deformações plásticas permanentes. Ciclos de carga e descarga podem acumular deformações plásticas e levar a estrutura à ruína com um carregamento bastante inferior ao que seria limite caso fosse aplicado monotonicamente.

Estes dois efeitos envolvem variações no tempo e agregam grande grau de complexidade à análise, fugindo ao escopo deste trabalho. Colabora para isto o fato de que, em uma estrutura de concreto armado convencional, as cargas permanentes atuantes correspondem usualmente a mais de 80% do total, minimizando os efeitos variáveis do carregamento.

CAPÍTULO 3.
NÃO LINEARIDADE GEOMÉTRICA

3.1 - VISÃO GERAL

Existem diversos aspectos não lineares geométricos na análise estrutural, tendo uma visão geral destes sido abordada no item 2.4.6. Neste trabalho, os estudos serão limitados aos efeitos principais, considerados mais relevantes para a aplicação prática no projeto de edificações.

O primeiro e mais importante efeito é o causado pela ação das forças horizontais sobre uma edificação. Após a análise de 1ª ordem, os deslocamentos horizontais dos pontos de aplicação das cargas verticais provocam momentos adicionais em relação à base da estrutura. Estes momentos, chamados *momentos de 2a ordem,* provocam novo acréscimo aos deslocamentos horizontais da estrutura, criando um efeito iterativo de aumento nos deslocamentos e nos esforços que pode convergir para uma situação estável ou não. Este efeito, relativo à estrutura como um todo, é usualmente denominado efeito P-Δ.

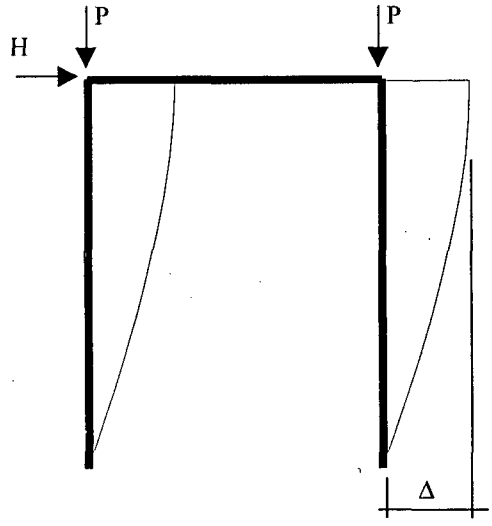


Figura 3.1 – Efeito P-Δ

Embora o efeito P-Δ seja referente aos deslocamentos nodais da estrutura, o mesmo problema aplica-se à configuração deformada de cada barra submetida a esforços axiais. O efeito das forças normais ao longo de uma barra é, algumas vezes, referenciado separadamente como efeito P-δ.

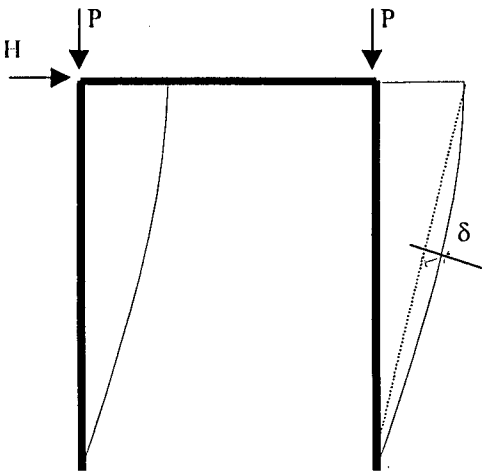


Figura 3.2 – Efeito P-δ

A outra abordagem dada ao problema é o fato de que, sob a ação de forças axiais, a rigidez à flexão de uma barra é modificada, sendo reduzida no caso de compressão e aumentada no caso de tração. Com isto, o mesmo processo pode ser encarado da seguinte forma: após a análise de 1ª ordem, computados os esforços internos em cada barra da estrutura, a presença de forças axiais faz com que a rigidez das barras utilizada para o cálculo seja modificada, obrigando um novo cálculo da estrutura. Obtidos os novos esforços, a rigidez das barras deve ser novamente corrigida, até que se obtenha a convergência.

3.2 - PROCESSO P-DELTA

3.2.1 Princípios básicos

A consideração da parcela P-Δ do comportamento não linear geométrico de uma estrutura pode ser considerada imediata. Separando-se uma barra da estrutura em sua condição deformada, nota-se o desvio Δ na aplicação da carga, considerada entre suas extremidades. Ao invés de incluir explicitamente a modificação da geometria na análise de estrutura, pode-se simplificar muito o processo substituindo o momento adicional causado pela excentricidade P.Δ por um binário de cargas horizontais auto-equilibradas.

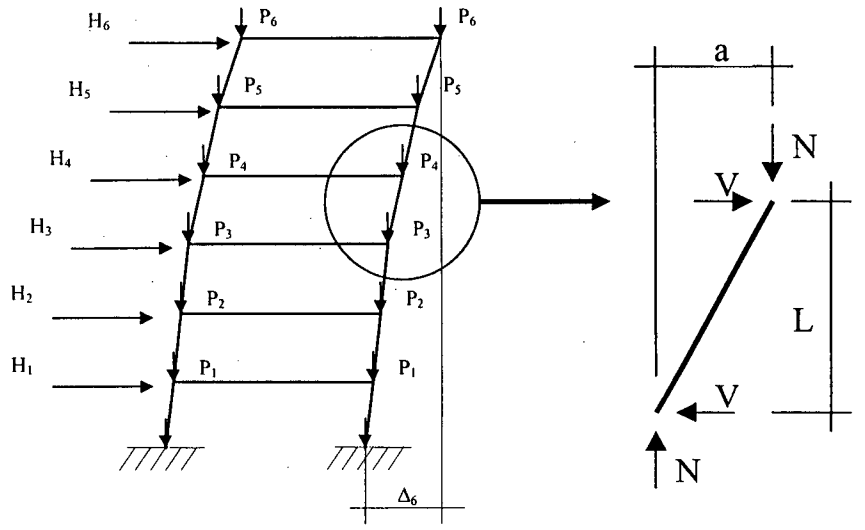


Figura 3.3 – Processo P-Delta

A cada barra da estrutura, acrescentam-se cortantes fictícios em suas extremidades, obtidos através de simples equilíbrio:

$$V = \frac{N \cdot a}{L} \qquad \text{Eq. (3-1)}$$

Onde:

- L = extensão da barra;
- N = carga normal atuante;
- a = deslocamento relativo entre suas extremidades;
- V = cortante fictício a ser acrescentado à análise.

3.2.2 Análise da estrutura

O mesmo raciocínio aplicado a uma barra isolada pode ser estendido para uma estrutura reticulada qualquer. Uma vez computados os cortantes fictícios Vi de cada barra, a diferença entre os cortantes de duas barras concorrentes em um nó gera uma carga equivalente Hi aplicada sobre o nó. O conjunto de cargas Hi atuantes sobre todos os nós irá reproduzir o efeito P-Δ da estrutura como um todo.

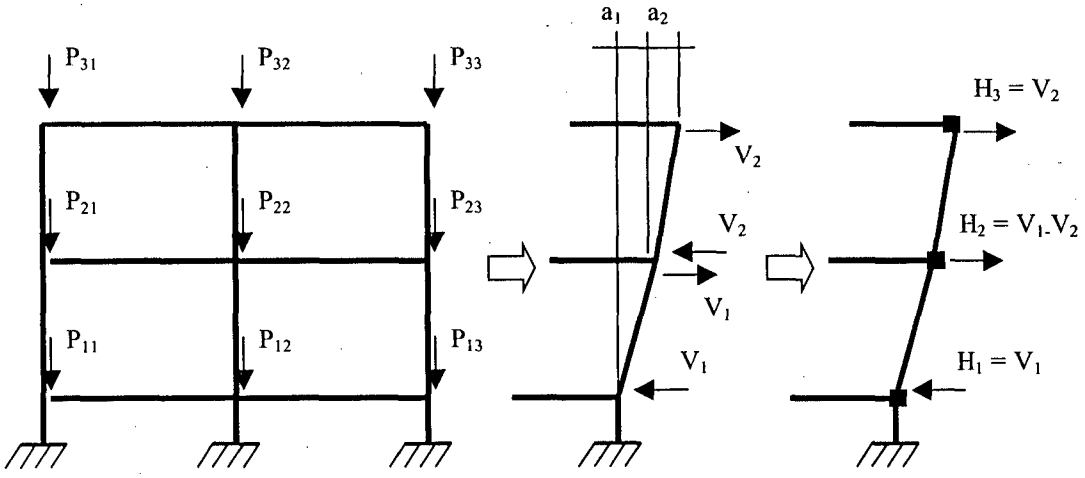


Figura 3.4 – Carregamentos nodais equivalentes ao efeito P-Δ

O Processo P-Delta pode ser considerado a forma mais imediata de abordar o efeito P-Δ de uma edificação. Seus princípios podem ser encontrados em diversas bibliografias tradicionais que tratam aspectos de projeto de estruturas tanto de concreto armado como metálicas. Em termos de Projeto, é considerado, normalmente, como uma abordagem "rigorosa" para o problema usualmente resolvido através da simples estimativa dos efeitos de 2ª ordem através de um fator característico, como o Parâmetro Alfa (vide item 2.4.3) ou o Coeficiente Gama-Z (vide item 2.4.4). Por exemplo, FUSCO (1981) descreve este processo como "Cálculo rigoroso de pórticos hiperestáticos" e GAIOTTI & SMITH (1989) o abordam simplesmente como o "Processo iterativo".

Em termos de normalização, este processo está descrito no Anexo L da NBR 8800/88 ("Projeto e Execução de Estruturas de Aço de Edifícios"). Conforme SANTOS & FRANCO (1993), a nova NBR 6118 não deve apresentar explicitamente o processo a ser utilizado (ao menos no texto base disponível no momento da elaboração deste trabalho), mas colocam: "Em estruturas de edifícios, permite-se, para a consideração da não linearidade geométrica, o emprego do processo P-Δ (também conhecido como N-a)".

A abordagem matricial do problema pode ser considerada bastante simples:

- Efetua-se a análise de 1ª ordem da forma usual, obtendo-se os esforços de 1ª ordem em cada barra;
- Obtém-se o vetor {F'} correspondente aos esforços fictícios obtidos da consideração do efeito P-Δ em cada barra;
- Acrescenta-se o vetor {F'} ao vetor de forças {F};
- Calculam-se os novos deslocamentos e esforços internos;
- Repete-se o processo, obtendo novo vetor {F'} até que a diferença no deslocamento nodal entre uma iteração e outra, medido para cada nó da estrutura, seja desprezível.

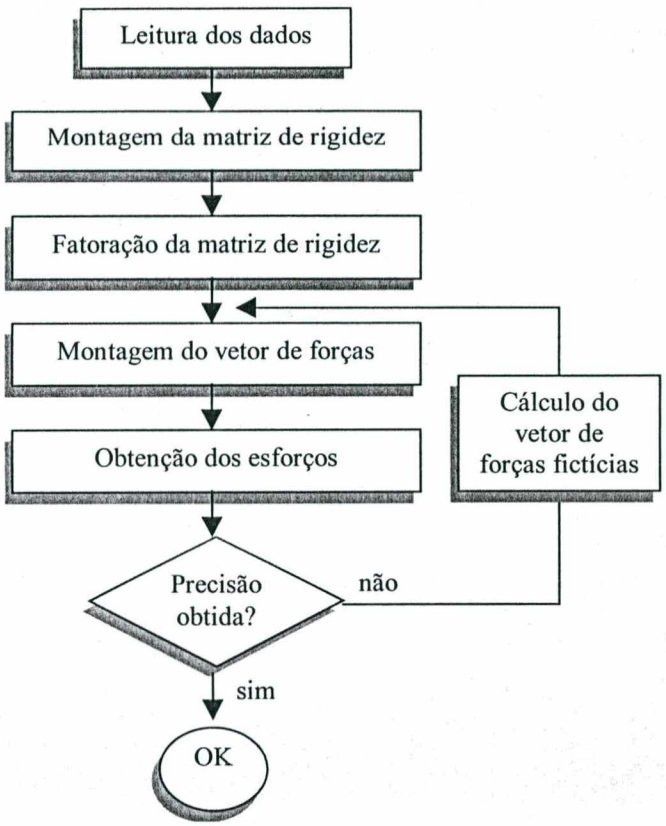


Figura 3.5 – Fluxograma representativo do Processo P-Delta

Pelo fato de ser este um processo numérico, deve-se estabelecer um critério de convergência, estabelecendo-se uma tolerância, abaixo da qual consideram-se satisfatórios os resultados obtidos. Esta tolerância é dependente do problema a ser analisado e deve ser decidido com base em seu custo computacional. Vai-se analisar a convergência do processo através de exemplos numéricos, apresentados em item posterior.

3.2.3 Aplicação às barras não verticais

Conforme já exposto, o Processo P-Delta confunde-se, em grande parte da bibliografia, especialmente aquela dedicada à área de Projeto, com o próprio efeito P-Δ apresentado por uma estrutura. O enfoque de Projeto sugere o cômputo das cargas fictícias apenas nas barras verticais (pilares), obtendo sempre carregamentos horizontais aplicados. Isto porque estes concentram normalmente quase todo o efeito de 2ª ordem. Esta recomendação é feita pela própria NBR 8800/88 em seu Anexo L.

Todavia, o processo, conforme apresentado, é genérico e pode ser aplicado para barras em qualquer orientação. A aplicação do processo apenas às barras verticais pode ser considerada de cunho prático para aplicação manual, mas pode ser desconsiderada para fins de implementação computacional. Mesmo a norma NBR 8800/88 "permite" e não "sugere" a dispensa das barras não verticais (horizontais ou inclinadas).

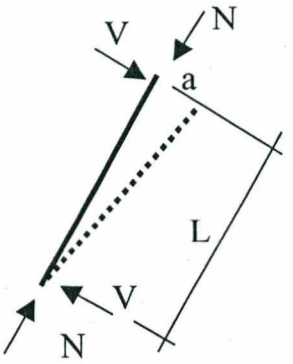


Figura 3.6 – Processo P-Delta aplicado às barras não verticais

ZERMIANI (1998) apresenta o Processo P-Δ aplicando-o apenas às barras verticais da estrutura e define um "Processo P-Δ genérico" aplicável a todas as barras independentemente de inclinação. Com alguns exemplos, afirma que os resultados obtidos são muito parecidos. Neste trabalho, assume-se diretamente o Processo P-Delta como sendo aplicável a todas as barras.

3.2.4 Efeito P-δ

A princípio, o Processo P-Delta aplica-se ao modelamento dos efeitos globais de 2ª ordem em uma estrutura. Quando é feito o equilíbrio da barra isoladamente, considera-se suas extremidades livres à rotação. Esta simplificação caracteriza o processo conforme apresentado.

Desta forma, este processo não consegue capturar o efeito P-δ ocorrido ao longo de um elemento. Pode-se verificar isto com um exemplo simples: considera-se uma barra biapoiada, sujeita a uma força normal compressiva e momentos fletores aplicados em suas extremidades.

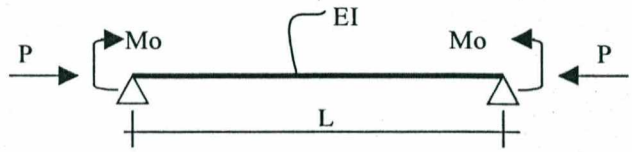


Figura 3.7 – Exemplo 1 – Viga-coluna biapoiada

Após a análise de 1ª ordem, a configuração deformada da estrutura apresenta uma variação ao longo de seu eixo, com um deslocamento máximo δ no centro do vão. Devido à presença da carga normal, os momentos de 2ª ordem provocarão aumento nos deslocamentos, em processo análogo ao efeito P-Δ apresentado pela estrutura como um todo.

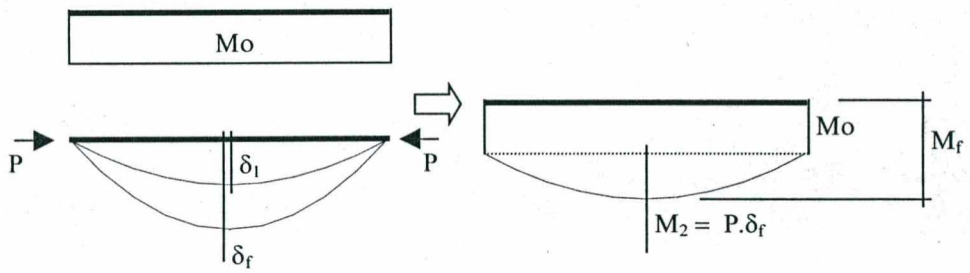


Figura 3.8 – Exemplo 1 – Efeito da 2ª ordem nos momentos fletores

Este exemplo foi inicialmente apresentado por LUI & ZHANG (1990) e apresenta uma solução analítica fechada, pela qual o momento máximo, considerando a 1ª e 2ª ordem, ocorre no meio do vão, com o valor:

$$M_f = M_0 \cdot \sec\left(\frac{L}{2}\sqrt{\frac{P}{EI}}\right) \qquad \text{Eq. (3-2)}$$

A aplicação do Processo P-Delta a esta estrutura simples resulta em esforços adicionais nulos, uma vez que o deslocamento relativo entre as extremidades da barra é nulo.

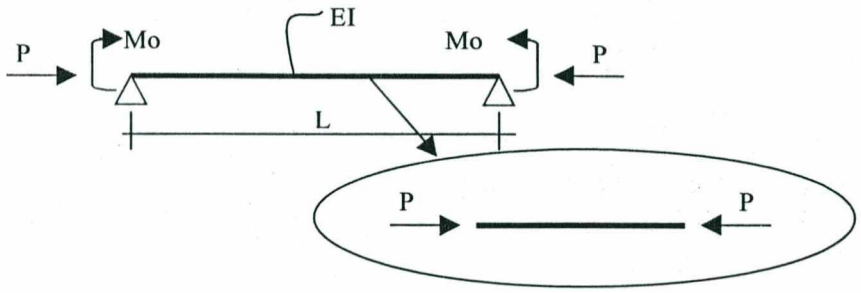


Figura 3.9 – Exemplo 1 – Aplicação do Processo P-Delta

Efeito da discretização

Este exemplo mostrou uma situação onde não se consegue modelar o comportamento da barra através do Processo P-Delta. Todavia, caso seja inserido um nó adicional no centro do vão, a viga comporta-se como uma estrutura e cada parte dela como um membro. Pode-se, portanto, capturar o efeito P-Δ concentrado no nó central, desprezando o efeito P-δ ocorrido no interior de cada parte dividida.

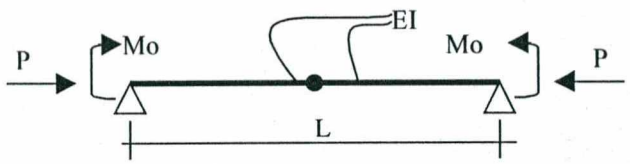


Figura 3.10 – Exemplo 1 – Divisão da estrutura em duas partes

Os resultados obtidos a partir da análise de 1ª ordem serão:

$$\delta_1 = \frac{M_0 \cdot L^2}{8 \cdot EI} \qquad \text{Eq. (3-3)}$$

$$M_1 = M_0 \qquad \text{Eq. (3-4)}$$

Dado o deslocamento δ ocorrido no nó central, podem ser obtidos os cortantes fictícios em cada barra:

$$V_2 = \frac{P \cdot \delta_1}{L/2} = \frac{P \cdot M_0 \cdot L}{2 \cdot EI} \qquad \text{Eq. (3-5)}$$

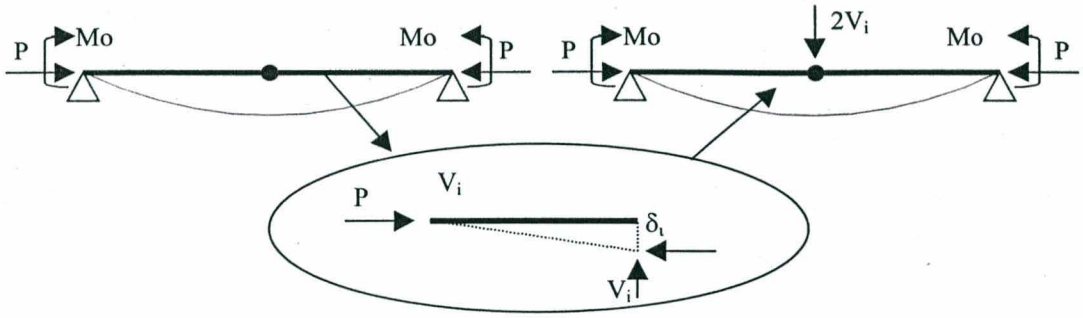


Figura 3.11 – Exemplo 1 com nó central – Aplicação do Processo P-Delta

Com base no carregamento fictício V_i aplicado à viga, obtém-se novo deslocamento δ_{i+1} de 2ª ordem[4]:

$$\delta_{i+1} = \frac{(2 \cdot V_i) \cdot L^3}{48 \cdot EI} \qquad \text{Eq. (3-6)}$$

Substituindo a Eq. (3-5) na Eq. (3-6), obtém-se uma expressão de δ_{i+1} diretamente em função de δ_i:

$$\delta_{i+1} = \frac{P \cdot L^2}{12 \cdot EI} \cdot \delta_i \qquad \text{Eq. (3-7)}$$

O valor final do deslocamento no vão será obtido através da somatória de todos os δ_i. Este valor é o de uma progressão geométrica infinita:

$$\delta_f = \delta_1 + k\,\delta_1 + k\,\delta_2 + k\,\delta_3 + \ldots$$

$$\delta_f = \delta_1 + k\,\delta_1 + k^2\,\delta_1 + k^3\,\delta_1 + \ldots$$

$$\delta_f = \delta_1 (1 + k + k^2 + \ldots) \qquad \text{Eq. (3-8)}$$

$$\text{Onde: } k = \frac{P \cdot L^2}{12 \cdot EI}$$

Existem duas situações possíveis:

- Caso $k < 1$, cada termo será menor que o anterior e a série convergirá para um valor finito;

[4] Expressão para cálculo da flecha em uma viga sujeita a uma carga concentrada no meio do vão.

- Caso $k \geq 1$, cada termo será pelo menos igual ao anterior e o valor da série tenderá ao infinito.

Supondo $k < 1$, obtém-se o seguinte valor de deslocamento final:

$$\delta_f = \frac{\delta_1}{1-k} = \frac{Mo \cdot L^2/8 \cdot EI}{1 - P \cdot L^2/12 \cdot EI} = \frac{3 \cdot Mo \cdot P \cdot L^2}{24 \cdot EI - 2 \cdot P \cdot L^2} \qquad \text{Eq. (3-9)}$$

Com base no deslocamento final, pode-se obter o valor do momento fletor final ocorrido no centro do vão:

$$M_f = Mo + P \cdot \delta_f$$

$$M_f = Mo \cdot \left(1 + \frac{3 \cdot P \cdot L^2}{24 \cdot EI - 2 \cdot P \cdot L^2}\right) \qquad \text{Eq. (3-10)}$$

Considerações

Mesmo com base neste exemplo simples, pode-se obter algumas conclusões importantes. A primeira delas é que é possível simular o efeito P-δ ocorrido no interior da uma barra com o Processo P-Delta, desde que a barra seja dividida através de nós intermediários.

A segunda delas se obtém comparando o momento final obtido através do Processo P-Delta - Eq. (3-10) – com o mesmo valor obtido através de uma solução analítica - Eq. (3-2).

$$M_f = Mo \cdot \sec\left(\frac{\sqrt{p'}}{2}\right) \quad \text{(analítica)} \qquad \text{Eq. (3-11)}$$

$$M_f = Mo \cdot \left(1 + \frac{3 \cdot p'}{24 - 2 \cdot p'}\right) \quad \text{(aproximada)} \qquad \text{Eq. (3-12)}$$

Onde:

$$p' = \frac{P \cdot L^2}{EI} \qquad \text{Eq. (3-13)}$$

A relação entre as duas expressões pode ser melhor visualizada expandindo-as em séries de Taylor:

$$Mf(\text{analitico}) := Mo + \frac{1}{8}\cdot Mo\cdot p + \frac{5}{384}\cdot Mo\cdot p^2 + \frac{61}{46080}\cdot Mo\cdot p^3 + \frac{277}{2064384}\cdot Mo\cdot p^4 + \frac{50521}{3715891200}\cdot Mo\cdot p^5$$

$$Mf(\text{aproximado}) := Mo + \frac{1}{8}\cdot Mo\cdot p + \frac{1}{96}\cdot Mo\cdot p^2 + \frac{1}{1152}\cdot Mo\cdot p^3 + \frac{1}{13824}\cdot Mo\cdot p^4 + \frac{1}{165888}\cdot Mo\cdot p^5$$

Pode-se notar que a diferença entre as duas equações não se encontra em seu formato, mas apenas nos coeficientes. Esta diferença pode ser explicada pelo fato de que o modelo utilizado concentra todo o efeito P-δ no nó central, ao invés de distribuí-lo ao longo da barra. Pode-se dizer que o Processo P-Delta tenderia ao resultado exato caso a barra fosse gradativamente subdividida em partes menores.

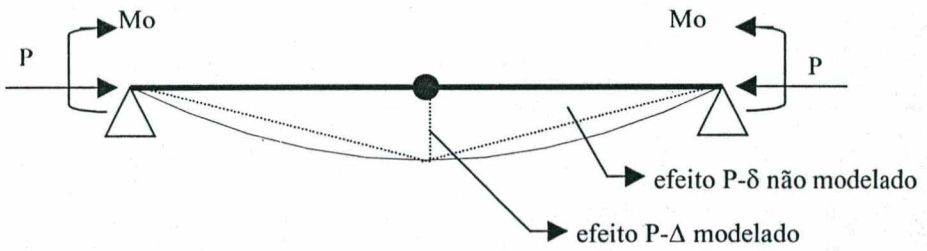


Figura 3.12 – Exemplo 1 com nó central – Análise do processo

Comparando-se os resultados obtidos através das duas expressões, pode-se verificar que a diferença entre elas não é constante, mas sim crescente de acordo com o valor de p'. O gráfico a seguir exibe a evolução do momento fletor final de acordo com o valor de p'.

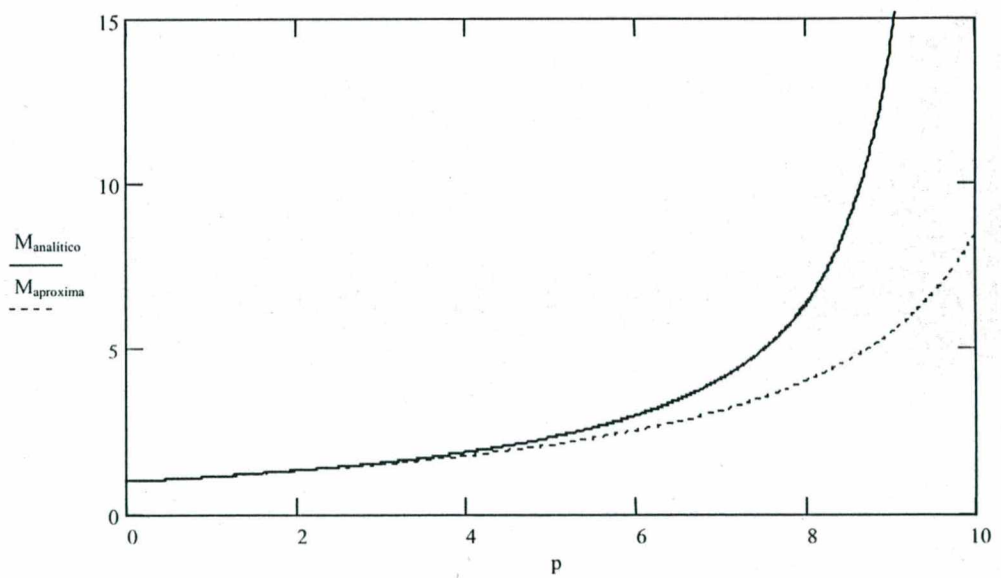


Figura 3.13 – Exemplo 1 com nó central – Evolução dos momentos fletores

A consideração final diz respeito à existência de um valor limite de p' acima do qual não existe convergência para os momentos fletores. A expressão analítica da Eq. (3-11) fornece uma singularidade para $p'=\pi^2$, chegando à conhecida expressão para a carga crítica de Euler. Neste ponto, os deslocamentos e momentos fletores tendem ao infinito.

$$Pe = \frac{\pi^2 \cdot EI}{L^2} \approx \frac{9.87 \cdot EI}{L^2} \qquad \text{Eq. (3-14)}$$

A expressão aproximada da Eq. (3-12) fornece também uma singularidade, mas para p' = 12. Analogamente, seria obtido:

$$Pe = \frac{12 \cdot EI}{L^2} \qquad \text{Eq. (3-15)}$$

Pode-se dizer, à primeira vista:

- O Processo P-Delta fornece valores de deslocamentos (e esforços) sempre inferiores aos resultados analíticos;
- A diferença cresce com a carga normal, sendo bastante significativa para níveis de carga próximos à carga crítica;
- Os valores obtidos tendem aos analíticos conforme a barra é subdividida;
- Pode-se utilizar o Processo P-Delta para obter a carga crítica de uma estrutura, mas esta será sempre superior à real, tendendo a esta conforme a discretização.

3.2.5 Influência da discretização

O trabalho manual necessário para resolver o mesmo problema com um número maior de divisões na forma literal seria desnecessariamente exaustivo. Pode-se chegar às mesmas conclusões utilizando-se de um exemplo numérico equivalente.

Resolvendo-se numericamente o problema para três relações de carregamento (p'=2, 5 e 9), obtém-se o diagrama a seguir, que apresenta as relações entre o deslocamento final obtido e o valor analítico para diversos níveis de subdivisão da barra:

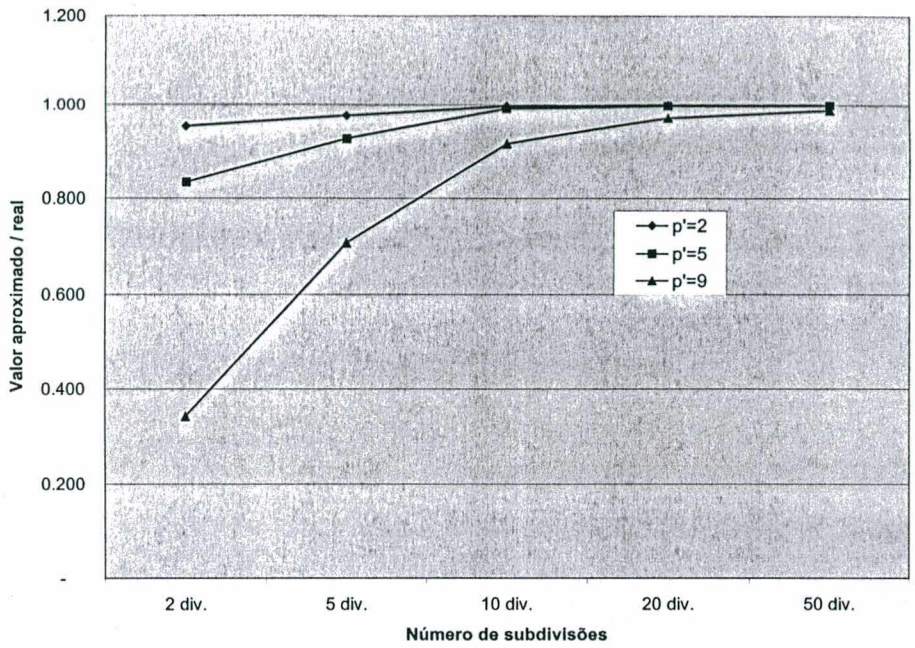


Figura 3.14 – Exemplo 1 – Efeito da discretização no Processo P-Delta

Pode-se notar que :

- O valor obtido realmente converge para o real conforme aumenta-se a discretização;
- Quanto maior o nível de força normal, maior o número de divisões necessárias para se chegar a uma precisão estabelecida (por exemplo, para p'=9, carga próxima da crítica, tem-se um erro de 8.7% mesmo com 10 divisões e um erro de 29.2% para 5 divisões).

3.3 - PROCESSO DA MATRIZ DE RIGIDEZ GEOMÉTRICA

Conforme exposto em 2.3.2, a relação entre os esforços internos e os deslocamentos nas extremidades de um elemento de barra é descrita através de um conjunto de equações de equilíbrio. Estas equações, uma vez dispostas na forma matricial, têm seus coeficientes agrupados na *matriz de rigidez* da barra.

Com a presença de uma carga normal, os esforços necessários para produzir um deslocamento unitário em um dos graus de liberdade da barra são alterados. Com isto, alteram-se os coeficientes da matriz de rigidez.

Devido à interdependência entre a rigidez e os esforços, todo problema não linear, para sua solução, será tratado como uma sequência de problemas lineares, resolvidos de forma a buscar iterativamente a solução não linear. Da mesma forma como para o Processo P-Delta, os esforços obtidos em uma análise anterior são utilizados para modificar a matriz de rigidez, obtendo-se novos esforços até a convergência.

A primeira formulação a ser apresentada será a da Matriz de Rigidez Geométrica. Nesta, a influência da força normal é suposta linear, modificando-se a Eq. (2-7) para:

$$\{F\} = ([K]+[K_G]).\{\delta\} \qquad \text{Eq. (3-16)}$$

Onde:

- [K] = matriz de rigidez elástica
- $[K_G]$ = matriz de rigidez geométrica

Para a dedução da matriz $[K_G]$, torna-se necessário o uso de princípios energéticos ao invés das relações diretas já apresentadas. Será feita, então, uma breve explanação sobre esta metodologia.

3.3.1 Formulação energética da matriz de rigidez

Os métodos energéticos são baseados na idéia de se encontrar estados consistentes de corpos ou estruturas associados com valores estacionários de uma quantidade escalar característica de corpos carregados. Em Engenharia, esta quantidade é usualmente uma medida de energia ou trabalho. Estes métodos fornecem uma abordagem genérica e muito poderosa para os problemas de Análise de Estruturas, considerada apenas um passo além da Análise Matricial convencional conforme abordada. Muitas idéias são comuns e a sistematização dos procedimentos é muito semelhante.

Não se pretende aqui explicitar toda a formulação energética de um elemento de barra. Detalhes adicionais podem ser encontrados facilmente na bibliografia especializada, sendo normalmente utilizada no Método dos Elementos Finitos. Adiante utilizam-se algumas exposições de SELKE & PEREIRA (1994), MOREIRA (1977) e ZERMIANI & LORIGGIO (1996).

Energia potencial

Pode-se definir um funcional π_p como a energia potencial de um corpo carregado. Se um corpo elástico linear está em equilíbrio, pode-se mostrar que ele assume a energia potencial mínima. A energia potencial π_p é definida como a soma da energia de deformação U e da energia potencial das forças externas W.

$$\pi_p = U + V \qquad \text{onde } V = -\Sigma F_i \delta_i \qquad \text{Eq. (3-17)}$$

Ao aplicar o princípio da energia potencial mínima, toma-se a variação de π_p e iguala-se a mesma a zero. Assumindo que o carregamento permanece constante, tem-se:

$$\delta\pi_p = \delta U + \delta V = 0 \qquad \text{Eq. (3-18)}$$

O símbolo δ indica variação, podendo ser expresso como uma série de derivadas parciais de π_p em função de seus parâmetros. Assim:

$$\textit{Supondo} \quad \pi_p = \pi_p(u_1, u_2, \ldots, u_n)$$

$$\delta\pi_p = 0 \Rightarrow \begin{bmatrix} \partial\pi_p / \partial u_1 \\ \partial\pi_p / \partial u_2 \\ \vdots \\ \partial\pi_p / \partial u_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \text{Eq. (3-19)}$$

Expressões do trabalho e energia sob forma matricial

Considere-se uma estrutura submetida a cargas externas {F}, ficando seus elementos sujeitos a esforços {S}. Estes elementos sofrerão deformações {d} e a estrutura apresentará deslocamentos nodais {δ}. O trabalho realizado pelas ações que variam de 0 até F_i, sobre os deslocamentos que variam de 0 até δ_i:

$$W = \Sigma \frac{1}{2} F_i \delta_i \qquad \text{Eq. (3-20)}$$

$$W = \Sigma \frac{1}{2} \{F\}^T \{\delta\} = \Sigma \frac{1}{2} \{\delta\}^T \{F\} \qquad \text{Eq. (3-21)}$$

A energia de deformação da estrutura, medida pelo trabalho dos esforços internos S_i face às deformações d_i:

$$U = \Sigma \frac{1}{2} S_i \, d_i \qquad \text{Eq. (3-22)}$$

$$U = \Sigma \frac{1}{2} \{S\}^T \{d\} = \Sigma \frac{1}{2} \{d\}^T \{S\} \qquad \text{Eq. (3-23)}$$

Conhecidas as matrizes de rigidez de cada elemento [r] e a matriz de rigidez global da estrutura [K], são válidas as relações expressas pela Eq. (2-4) e pela Eq. (2-7):

$$\{S\} = [r] \, . \, \{d\} \qquad \text{Eq. (2-4)}$$

$$\{F\} = [K] \, . \, \{\delta\} \qquad \text{Eq. (2-7)}$$

A partir disto, obtém-se:

$$W = \frac{1}{2} \{\delta\}^T [K] \{\delta\} \qquad \text{Eq. (3-24)}$$

$$U = \Sigma \frac{1}{2} \{d\}^T [r] \{d\} \qquad \text{Eq. (3-25)}$$

Teorema de Castigliano

Sejam $\{F\}$ e $\{\delta\}$ as ações e deslocamentos de uma estrutura, respectivamente, com matriz de rigidez [K]. O princípio da conservação da energia permite concluir que, no regime elástico e sob carregamento estático:

$$U = W = \frac{1}{2} \{\delta\}^T [K] \{\delta\} \qquad \text{Eq. (3-26)}$$

Se ocorrer variação elementar de um dos deslocamentos δ_i, tem-se:

$$\frac{\partial U}{\partial \delta_i} = \frac{1}{2} \left(\frac{\partial \{\delta\}^T}{\partial \delta_i} [K] \{\delta\} + \{\delta\}^T [K] \frac{\partial \{\delta\}}{\partial \delta_i} \right) \qquad \text{Eq. (3-27)}$$

Considerando-se que:

$$\frac{\partial\{\delta\}^T}{\partial\delta_i} = \frac{\partial}{\partial\delta_i}\begin{bmatrix}\delta_1 & \delta_2 & \dots & \delta_i & \cdots & \delta_n\end{bmatrix} = \begin{bmatrix}0 & 0 & \dots & 1 & \cdots & 0\end{bmatrix} \qquad \text{Eq. (3-28)}$$

$$\frac{\partial\{\delta\}}{\partial\delta_i} = \frac{\partial}{\partial\delta_i}\begin{bmatrix}\delta_1\\ \delta_2\\ \vdots\\ \delta_i\\ \vdots\\ \delta_n\end{bmatrix} = \begin{bmatrix}0\\ 0\\ \vdots\\ 1\\ \vdots\\ 0\end{bmatrix} \qquad \text{Eq. (3-29)}$$

Substituindo-se a Eq. (3-28) e a Eq. (3-29) na Eq. (3-27), obtém-se:

$$\frac{\partial U}{\partial\delta_i} = F_i \qquad \text{Eq. (3-30)}$$

1º Teorema de Castigliano: "A derivada da energia de deformação U em relação a um dos deslocamentos δ_i fornece a ação mecânica F_i aplicada na direção desse deslocamento".

Por outro lado, sendo:

$$F_i = K_{i1}\,\delta_1 + K_{i2}\,\delta_2 + \dots + K_{in}\,\delta_n \qquad \text{Eq. (3-31)}$$

Obtém-se:

$$\frac{\partial^2 U}{\partial\delta_i^{\,2}} = \frac{\partial F_i}{\partial\delta_i} = K_{ii} \qquad \text{Eq. (3-32)}$$

$$\frac{\partial^2 U}{\partial\delta_j\cdot\partial\delta_i} = K_{ij} = K_{ji} \qquad \text{Eq. (3-33)}$$

Os coeficientes de rigidez podem ser obtidos através da dupla derivação parcial da energia de deformação, conforme Eq. (3-32) e Eq. (3-33). Isto nos fornece outra metodologia que permite chegar à matriz de rigidez de um elemento de barra. Em muitas situações, esta abordagem facilita o desenvolvimento das equações e é normalmente utilizada para estender os conceitos abordados no item 2.3.

Funções de interpolação

Para que o tratamento matricial seja feito com base nos deslocamentos nodais {δ} ao invés do campo de deslocamentos contínuos ao longo da estrutura, é necessário definir funções que interpolam os deslocamentos nodais obtidos ao longo do comprimento do elemento. Cada tipo de elemento pode definir funções diferentes, sendo estas normalmente expressas em termos de coordenadas locais do elemento. Para o caso de um elemento de pórtico plano, contendo seis deslocamentos possíveis, definem-se as seguintes *funções de interpolação* N:

$$N_1(\xi) = 1 - \xi$$

$$N_2(\xi) = 1 - 3\,\xi^2 + 2\,\xi^3$$

$$N_3(\xi) = L\,\xi\,(1 - 2\,\xi + \xi^2) \qquad \text{Eq. (3-34)}$$

$$N_4(\xi) = \xi$$

$$N_5(\xi) = \xi^2\,(3 - 2\,\xi)$$

$$N_6(\xi) = L\,\xi^2\,(\xi - 1)$$

Onde:

$$\xi = \frac{x}{L} \qquad \text{Eq. (3-35)}$$

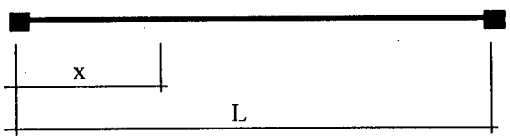


Figura 3.15 – Coordenada local de um elemento de barra

As funções de interpolação N_2, N_3, N_5 e N_6 adotadas são chamadas usualmente de funções de interpolação de Hermite. Por isto, esta formulação para o elemento de barra é também chamada de *elemento cúbico de Hermite,* largamente utilizada no Método dos Elementos Finitos.

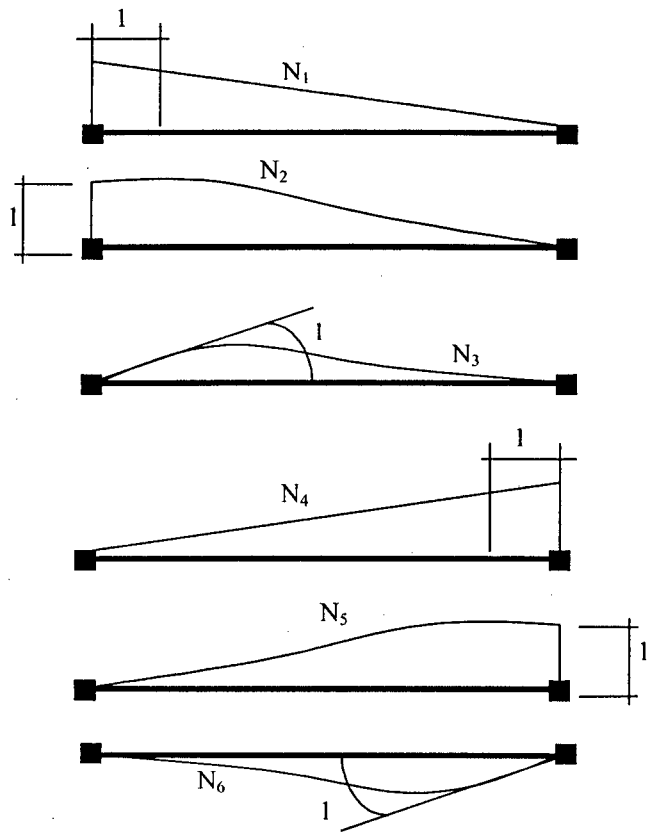


Figura 3.16 – Funções de interpolação de Hermite

3.3.2 Obtenção da matriz de rigidez geométrica

A obtenção da matriz de rigidez da barra fornece ações mecânicas correspondentes a deslocamentos unitários impostos. As ações {S} associadas a estas configurações seriam os termos da matriz de rigidez elástica conforme definida em 2.3.2. Considerando a presença de uma força normal P, tem-se:

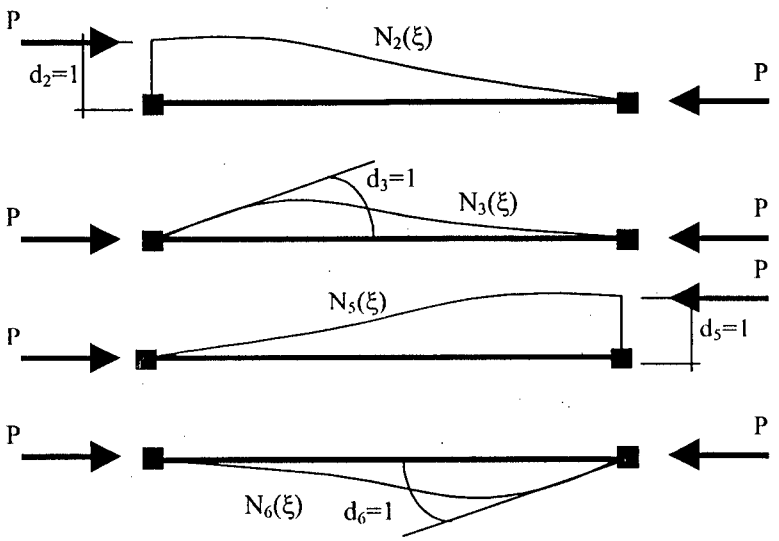


Figura 3.17 – Inclusão da força normal na dedução da rigidez

Com a presença da carga P, devem ser consideradas ações suplementares {G}, necessárias para restabelecer o equilíbrio. Supõe-se que a influência de P é linear em presença destas deformações.

Utilizando o princípio da superposição das ações, pode-se utilizar apenas a parcela da energia de deformação proveniente do produto da força P pelo deslocamento na barra.

No presente caso, tem-se:

$$U = \frac{P}{2}\int_0^L \left(\frac{dy}{dx}\right)^2 dx \qquad \text{Eq. (3-36)}$$

Onde:

- P = força normal aplicada nas extremidades do elemento
- x = coordenada local do elemento
- y = y(x) = deslocamento transversal ocorrido na barra, em função de x

Convertendo a Eq. (3-35) para a coordenada isoparamétrica ξ ao longo do elemento, esta fornece que dx = L dξ. Com isto, a Eq. (3-36) pode ser modificada para:

$$U = \frac{P}{2 \cdot L}\int_0^1 \left(\frac{dy}{d\xi}\right)^2 d\xi \qquad \text{Eq. (3-37)}$$

Utilizando-se as funções de interpolação definidas na Eq. (3-34), pode-se escrever o deslocamento y(ξ) em função dos deslocamentos nodais {d}:

$$y(\xi) = d_2.N_2(\xi) + d_3.N_3(\xi) + d_5.N_5(\xi) + d_6.N_6(\xi) \qquad \text{Eq. (3-38)}$$

Pode-se também escrever (onde o símbolo ' representa a primeira derivada da função em relação a ξ):

$$y'(\xi) = d_2.N_2'(\xi) + d_3.N_3'(\xi) + d_5.N_5'(\xi) + d_6.N_6'(\xi) \qquad \text{Eq. (3-39)}$$

Substituindo-se a Eq. (3-39) na Eq. (3-37), tem-se:

$$U_N = \frac{P}{2 \cdot L} \int_0^L \left(d_2 N_2' + d_3 N_3' + d_5 N_5' + d_6 N_6'\right)^2 d\xi$$

$$U_N = \frac{P}{2 \cdot L} \left(\begin{array}{l} d_2{}^2 \int_0^L N_2'^2 \, d\xi + \mathrm{K} + d_6{}^2 \int_0^L N_6'^2 \, d\xi + \\ 2 d_2 d_3 \int_0^L N_2' N_3' d\xi + \mathrm{K} + 2 d_3 d_4 \int_0^L N_3' N_4' d\xi \end{array} \right) \qquad \text{Eq. (3-40)}$$

Para simplificar, pode-se escrever:

$$\int_0^L N_i'^2 \, d\xi = C_{ii} \qquad \text{Eq. (3-41)}$$

$$\int_0^L N_i' N_j' d\xi = C_{ij} \qquad \text{Eq. (3-42)}$$

Desta forma, pode-se reescrever a Eq. (3-40):

$$U_N = \frac{P}{2 \cdot L} \left(\begin{array}{l} d_1{}^2 C_{11} + d_2{}^2 C_{22} + d_3{}^2 C_{33} + d_4{}^2 C_{44} + \\ 2 d_1 d_2 C_{12} + 2 d_1 d_3 C_{13} + 2 d_1 d_4 C_{14} + \\ 2 d_2 d_3 C_{23} + 2 d_2 d_4 C_{24} + 2 d_3 d_4 C_{34} \end{array} \right) \qquad \text{Eq. (3-43)}$$

Aplicando-se o Teorema de Castigliano, conforme Eq. (3-32) e Eq. (3-33), deve-se derivar parcialmente, duas vezes, a energia de deformação em relação aos deslocamentos d_i, obtendo-se:

$$\frac{\partial^2 U_N}{\partial d_i{}^2} = \frac{P}{L} C_{ii} = g_{ii} \qquad \text{Eq. (3-44)}$$

$$\frac{\partial^2 U_N}{\partial d_i d_j} = \frac{P}{L} C_{ij} = g_{ij} \qquad \text{Eq. (3-45)}$$

Logo, os termos da matriz de rigidez geométrica da barra, em coordenadas locais, serão dados substituindo-se as funções de interpolação $N_i(\xi)$ da Eq. (3-34) nas integrais da Eq. (3-41) e Eq. (3-42), obtendo os valores de C_{ii} e C_{ij}. Multiplicando-os pelo fator $-\frac{P}{L}$, conforme Eq. (3-44) e Eq. (3-45), obtém-se a matriz completa:

$$[K_G]=\frac{P}{L}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6}{5} & \frac{L}{10} & 0 & -\frac{6}{5} & \frac{L}{10} \\ 0 & \frac{L}{10} & \frac{2\cdot L^2}{15} & 0 & -\frac{L}{10} & -\frac{L^2}{30} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{6}{5} & -\frac{L}{10} & 0 & \frac{6}{5} & -\frac{L}{10} \\ 0 & \frac{L}{10} & -\frac{L^2}{30} & 0 & -\frac{L}{10} & \frac{2\cdot L^2}{15} \end{bmatrix} \qquad \text{Eq. (3-46)}$$

Observa-se que, devido às simplificações efetuadas, a matriz de rigidez geométrica é uma função *linear* da carga normal P.

3.3.3 Sistematização do processo

A abordagem matricial do problema pode ser considerada bastante simples:

- Efetua-se a análise de 1ª ordem da forma usual, obtendo-se os esforços de 1ª ordem em cada barra;
- Calcula-se a matriz de rigidez geométrica $[K_G]$ para cada elemento de barra;
- Acrescenta-se a matriz $[K_G]$ à matriz de rigidez elástica [K];
- Monta-se a nova matriz de rigidez corrigida da estrutura;
- Calculam-se os novos deslocamentos e esforços internos;
- Repete-se o processo, obtendo novas matrizes $[K_G]$ até que a diferença nos deslocamentos nodais, entre uma iteração e outra, medido para cada nó da estrutura, seja desprezível.

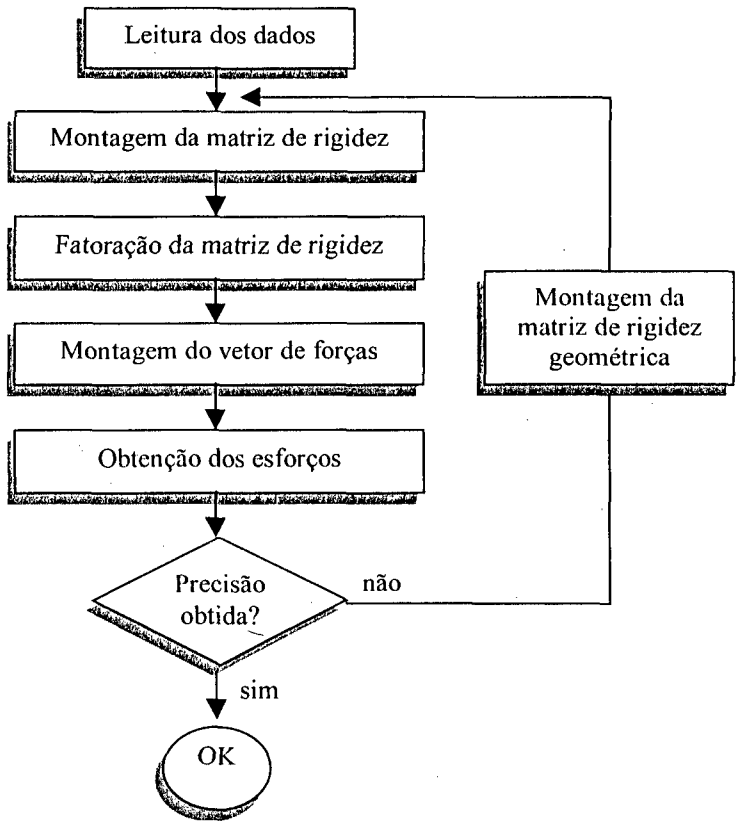


Figura 3.18 – Fluxograma representativo do processo K_G

Pelo fato de ser este um processo numérico, deve-se estabelecer um critério de convergência, estabelecendo uma tolerância abaixo da qual pode-se considerar desprezível a variação nos deslocamentos nodais. Esta precisão necessária é dependente do problema a ser analisado e deve ser decidido com base em seu custo computacional. Vai-se analisar a convergência do processo através de exemplos numéricos, apresentados em item posterior.

3.3.4 Efeito P-δ

O fato de se considerar linear a influência da força normal N sobre a rigidez da barra faz com que a solução correta só possa ser encontrada através da divisão da barra em elementos menores. Uma vez que a rigidez à flexão da barra é modificada, este processo consegue capturar o efeito P-δ ocorrido ao longo de um elemento. Com o auxílio das funções de interpolação definidas na Eq. (3-34), pode-se acompanhar o campo de deslocamentos no interior do elemento.

Pode-se verificar isto através do mesmo exemplo utilizado no item 3.2.4: considera-se uma barra biapoiada, sujeita a uma força normal compressiva e momentos fletores aplicados em suas extremidades. Reproduz-se abaixo a Figura 3.7:

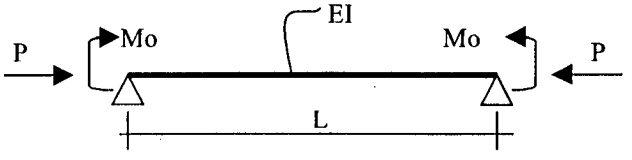


Figura 3.7 – Exemplo 1 – Viga-coluna biapoiada

Utilizando-se a relação definida pela Eq. (3-16):

$$\{F\} = ([K]+[K_G]).\{\delta\} \qquad \text{Eq. (3-16)}$$

Consideram-se as seguintes condições de contorno:

- Os deslocamentos axiais d_1 e d_4 não afetam o valor do momento de 2ª ordem, podendo ser ignorados (reduz-se a um problema de viga);
- Conhecem-se os deslocamentos $d_2=0$ e $d_5=0$.

Com isto, tomando-se apenas os deslocamentos desconhecidos d_3 e d_6, tem-se o seguinte sistema:

$$\begin{Bmatrix} -Mo \\ Mo \end{Bmatrix} = \left(\begin{bmatrix} \frac{4\cdot EI}{L} & \frac{2\cdot EI}{L} \\ \frac{2\cdot EI}{L} & \frac{4\cdot EI}{L} \end{bmatrix} - \begin{bmatrix} \frac{2\cdot P\cdot L}{15} & \frac{-P\cdot L}{30} \\ \frac{-P\cdot L}{30} & \frac{2\cdot P\cdot L}{15} \end{bmatrix} \right) \cdot \begin{Bmatrix} d_3 \\ d_6 \end{Bmatrix} \qquad \text{Eq. (3-47)}$$

Somando-se as matrizes referentes a [K] e [K_G] e invertendo o resultado, resulta:

$$\begin{Bmatrix} d_3 \\ d_6 \end{Bmatrix} = \qquad \text{Eq. (3-48)}$$

$$\begin{bmatrix} 8\cdot(30\cdot EI - P\cdot L^2)\cdot\frac{L}{720\cdot EI^2 - 72\cdot EI\cdot P\cdot L^2 + P^2\cdot L^4} & -2\cdot(60\cdot EI + P\cdot L^2)\cdot\frac{L}{720\cdot EI^2 - 72\cdot EI\cdot P\cdot L^2 + P^2\cdot L^4} \\ -2\cdot(60\cdot EI + P\cdot L^2)\cdot\frac{L}{720\cdot EI^2 - 72\cdot EI\cdot P\cdot L^2 + P^2\cdot L^4} & 8\cdot(30\cdot EI - P\cdot L^2)\cdot\frac{L}{720\cdot EI^2 - 72\cdot EI\cdot P\cdot L^2 + P^2\cdot L^4} \end{bmatrix} \begin{bmatrix} -Mo \\ Mo \end{bmatrix}$$

Efetuando-se as multiplicações, podem ser obtidos os deslocamentos incógnitos. Após simplificar as expressões, obtém-se:

$$d_3 = -d_6 = \frac{6 \cdot Mo \cdot L}{P \cdot L^2 - 12 \cdot EI} \qquad \text{Eq. (3-49)}$$

Uma vez obtidos os deslocamentos nodais, pode-se utilizar as funções de interpolação definidas na Eq. (3-34) para obter o deslocamento no meio do vão. Utilizando-se $\xi = 0.5$, obtém-se:

$$\delta = \frac{L}{8}(d_3 - d_6) \qquad \text{Eq. (3-50)}$$

Substituindo-se a Eq. (3-49) na Eq. (3-50), chega-se ao deslocamento no meio do vão. Neste caso, este já é o deslocamento final, uma vez que a força normal P atuante no elemento não será modificada. O deslocamento final será, portanto:

$$\delta_f = \frac{3 \cdot Mo \cdot L^2}{2 \cdot (12 \cdot EI - P \cdot L^2)} \qquad \text{Eq. (3-51)}$$

Com base no deslocamento final, pode-se obter o valor do momento fletor final ocorrido no centro do vão:

$$M_f = Mo + P \cdot \delta_f$$
$$M_f = Mo \cdot \left(1 + \frac{3 \cdot P \cdot L^2}{24 \cdot EI - 2 \cdot P \cdot L^2}\right) \qquad \text{Eq. (3-52)}$$

Observa-se que, coincidentemente, esta expressão é a mesma que foi obtida no caso de aplicação do Processo P-Delta ao mesmo exemplo, mas com a barra dividida em duas partes.

Considerações

Mesmo com base neste exemplo simples, pode-se obter algumas conclusões importantes. A primeira delas é que o Processo da Matriz de Rigidez Geométrica pode simular o efeito P-δ ocorrido no interior da uma barra, mesmo sem a divisão da barra em nós intermediários.

A segunda delas se obtém comparando o momento final obtido através do processo K_G - Eq. (3-52) – com o mesmo valor obtido através de uma solução analítica - Eq. (3-2).

$$M_f = Mo \cdot \sec\left(\frac{\sqrt{p'}}{2}\right) \quad \text{(analítica)} \qquad \text{Eq. (3-53)}$$

$$M_f = Mo \cdot \left(1 + \frac{3 \cdot p'}{24 - 2 \cdot p'}\right) \quad \text{(aproximada)} \qquad \text{Eq. (3-54)}$$

$$\text{Onde } p' = \frac{P \cdot L^2}{EI}$$

Todas as considerações feitas no item 3.2.4 podem ser aplicadas a este exemplo, uma vez que chegou-se à mesma expressão para o momento fletor final.

Pode-se verificar também que a carga crítica é obtida para p' = 12. Desta vez, este valor não indica a não convergência dos deslocamentos, mas sim o valor que torna singular a matriz de rigidez da barra.

3.3.5 Influência da discretização

O trabalho manual necessário para resolver o mesmo problema com um número maior de divisões na forma literal seria desnecessariamente exaustivo. Pode-se chegar às mesmas conclusões utilizando-se de um exemplo numérico equivalente.

Resolvendo-se numericamente o problema para três relações de carregamento (p'=2, 5 e 9), obtém-se o diagrama a seguir, que apresenta as relações entre o deslocamento final obtido e o valor analítico para diversos níveis de subdivisão da barra:

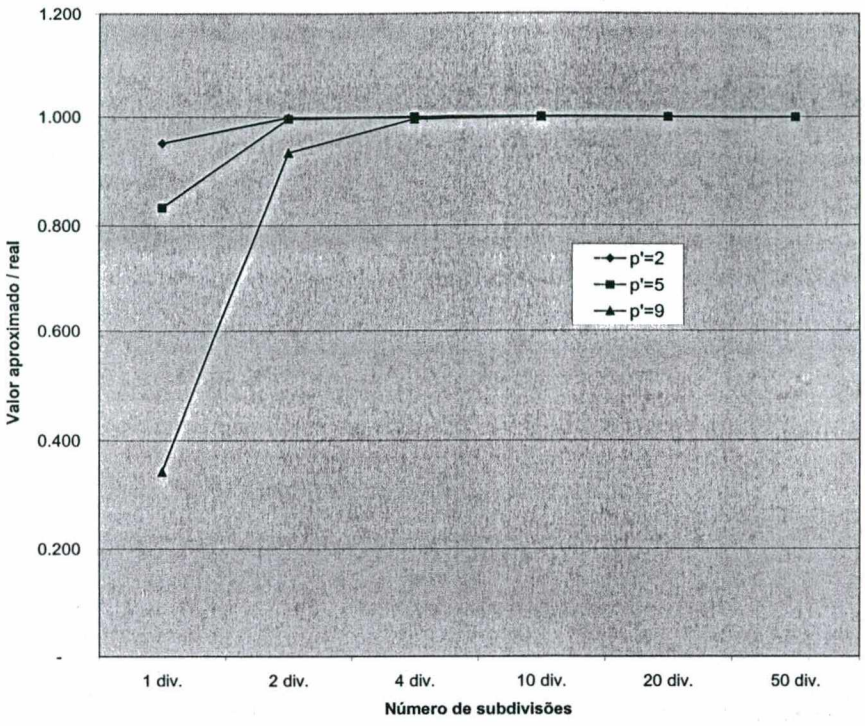


Figura 3.19 – Exemplo 1 – Efeito da discretização no processo K_G

Pode-se notar que :

- O valor obtido converge rapidamente para o real conforme aumenta-se a discretização;
- Quanto maior o nível de força normal, maior o número de divisões necessárias para se chegar a uma precisão estabelecida (por exemplo, para duas divisões, o erro é da ordem de 0.1% para p'=2, mas de 6.6% para p'=9);
- A diferença encontrada passando-se de uma para duas divisões é bastante grande, indicando que este processo converge muito rapidamente. Com duas divisões, não se obteve erro superior a 10%.

3.3.6 Comparação entre P-Delta e K_G

Analisando-se este primeiro exemplo, pode-se chegar a algumas conclusões acerca da utilização de um e outro processo. Em termos de resultados encontrados, pode-se dizer:

- Os valores encontrados pelo Processo P-Delta estão sempre abaixo daqueles obtidos através do Processo da Matriz de Rigidez Geométrica (K_G), estando ambos abaixo da solução analítica;

- Ambos os processos são influenciados pelo aumento da força normal, mas o processo K_G com menor intensidade que o P-Delta;
- O aumento do número de divisões melhora ambos os resultados, mas o processo K_G converge muito mais rapidamente.

Pode-se verificar isto analisando os resultados obtidos para p'=9. A tabela a seguir apresenta a relação entre os valores obtidos e a solução analítica:

N.º de divisões	P-Delta	K_G
1	N/A	0.343
2	0.343	0.934
4	0.708	0.995
10	0.913	1.000
20	0.969	1.000
50	0.987	1.000

Tabela 3.1 – Exemplo 1 – Comparação entre os resultados de P-Delta e K_G para p'=9

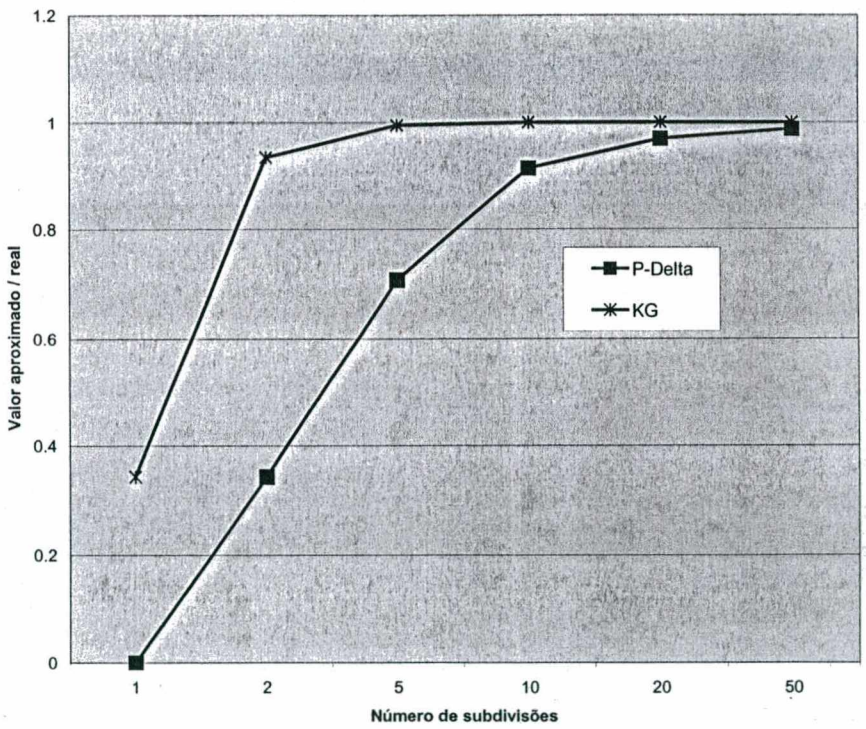


Figura 3.20 – Exemplo 1 – Comparação entre os resultados de P-Delta e K_G para p'=9

3.4 PROCESSO DA MATRIZ DE RIGIDEZ GEOMÉTRICA MODIFICADO (K_{GM})

Existem diversas formas de resolver um problema não linear através de soluções lineares iterativas. Uma forma de otimizar o Processo da Matriz de Rigidez Geométrica é o que será descrito a seguir.

Pode-se modificar a Eq. (3-16) da seguinte forma:

$$\{F\} = ([K]+[K_G]).\{\delta\} \qquad \text{Eq. (3-16)}$$

$$\{F\} - [K_G].\{\delta\} = [K].\{\delta\} \qquad \text{Eq. (3-55)}$$

A equação acima pode ser resolvida iterativamente adotando a parcela $[K_G].\{\delta\}$ correspondente ao produto da matriz de rigidez geométrica pelo vetor de deslocamentos obtido na iteração anterior, obtendo um novo vetor $\{\delta\}$. Adotando-se um esquema de solução baseado na inversão da matriz [K], tem-se:

$$\{\delta_i\} = [K]^{-1} . (\{F\} - [K_G].\{\delta_{i-1}\}) \qquad \text{Eq. (3-56)}$$

Teoricamente, este processo apresenta exatamente os mesmos resultados do Processo da Matriz de Rigidez Geométrica, desde que se utilize um número suficiente de iterações.

3.4.1 Diferença computacional

Embora todos os resultados apresentados indiquem vantagem na utilização do Processo da Matriz de Rigidez Geométrica sobre o Processo P-Delta, deve-se pesar outro ponto importante: o custo computacional.

Em ambos os processos, é efetuada uma análise linear para obter os resultados iniciais de esforços (no caso, as forças axiais). Com base nisto, são modificados alguns parâmetros do problema, efetuando-se nova análise linear. Este procedimento, executado iterativamente, simula a análise não linear.

No Processo P-Delta, forças fictícias são adicionadas aos nós da estrutura, enquanto que, no processo K_G, é feita uma modificação da matriz de rigidez. Por não alterar a matriz de rigidez, o Processo P-Delta possui a vantagem de fatorar a matriz da estrutura apenas uma vez, aproveitando a mesma solução para resolver os diversos vetores de carregamentos. Uma vez que a fatoração é a parte do processo que consome mais tempo na solução, esta diferença torna-se muito significativa no caso de sistemas maiores (projetos reais). Este é o objetivo do Processo da Matriz de Rigidez Geométrica Modificado.

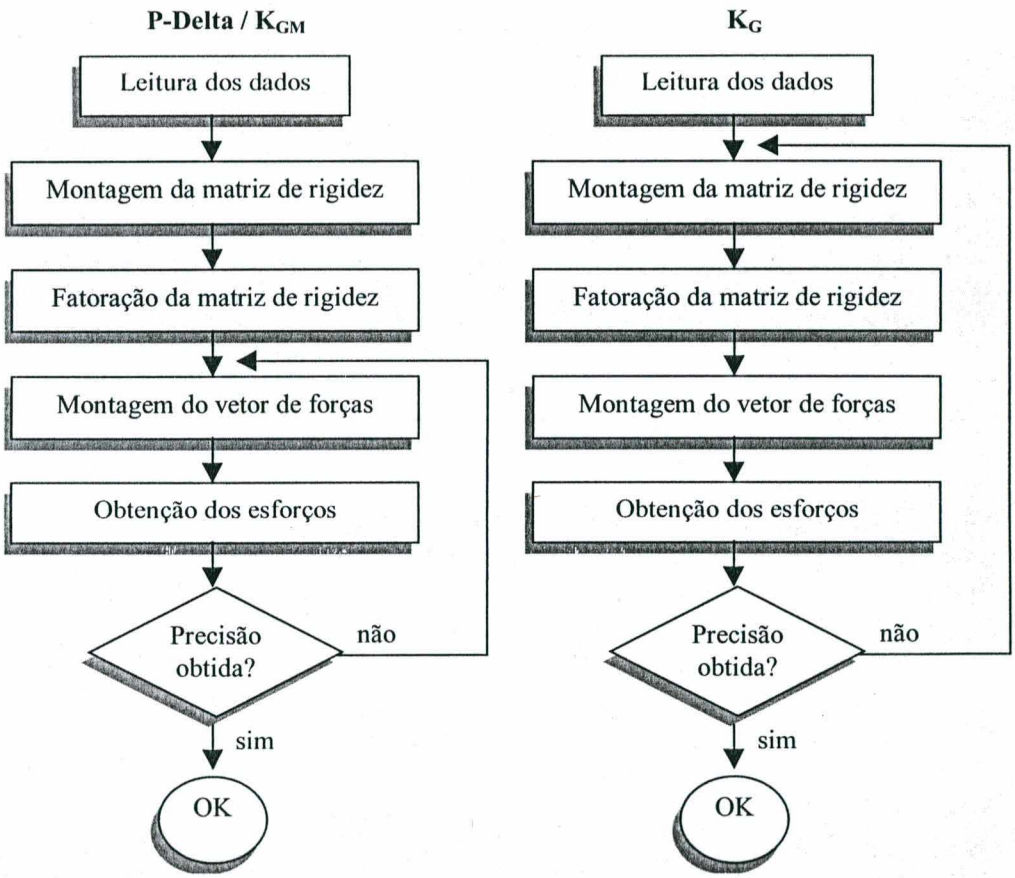


Figura 3.21 – Comparação entre os fluxogramas do Processo P-Delta e do K_G

Número de iterações

Outro ponto que pode influenciar o tempo necessário para a obtenção da solução é o número de iterações necessárias para se chegar à precisão preestabelecida. Para este exemplo, a aplicação do processo K_G resultou em apenas uma iteração após a análise de 1ª ordem. Isto porque, na configuração apresentada, a carga axial não varia com o deslocamento transversal da barra.

A diferença fundamental deriva do fato de que o Processo P-Delta depende, a cada iteração, da força normal em cada barra e do deslocamento relativo dos seus nós, enquanto que o processo K_G depende apenas da força normal nas barras. Nas estruturas usuais de edificações, o deslocamento horizontal dos nós não afeta muito a força normal nas barras, o que explica uma convergência rápida no processo K_G. Já pelo Processo P-Delta ou pelo processo K_{GM}, podem ser necessárias muitas iterações.

Analisando-se novamente o mesmo exemplo, obtém-se (considerando uma precisão mínima de 0.1%):

N.º de divisões	Processo P-Delta			Processo K_{GM}		
	p' = 2	p'= 5	p' = 9	p' = 2	p'= 5	p' = 9
2	4	8	20	4	8	20
4	5	9	35	5	9	35
10	5	10	46	5	10	46
20	5	10	48	5	10	48
50	5	10	49	5	10	49

Tabela 3.2 – Exemplo 1 – Nº de iterações necessárias para o Processo P-Delta e K_{GM}

O primeiro ponto a notar é a equivalência entre o Processo P-Delta e o processo K_{GM} em termos de convergência dos resultados. Como ambos utilizam a mesma solução inicial a cada iteração, apresentam basicamente o mesmo número de iterações necessárias para se chegar a uma precisão preestabelecida. Observa-se, portanto, que o Processo da Matriz de Rigidez Geométrica Modificado apresenta os mesmos resultados que seriam obtidos através da matriz K_G, mas utilizando a abordagem de solução do problema não linear exatamente igual à do P-Delta.

Pode-se obter também, a partir destes resultados:

- O número de iterações necessárias é influenciado fortemente pelo nível de força normal; quanto maior, mais iterações serão necessárias. Deve-se cuidar, portanto, com a imposição de um número limite de iterações. Um valor relativamente baixo como 10 ou 20 poderia indicar que o problema não converge para p'=9, o que não seria correto.
- Quanto maior o número de divisões, mais iterações são necessárias. Este efeito, embora não seja tão significativo quanto o anterior, denota as iterações necessárias para capturar o efeito P-δ no interior da barra. Independentemente do tipo de estrutura, o processo K_G não apresenta este comportamento, visto que a carga normal não varia ao longo da barra[5].

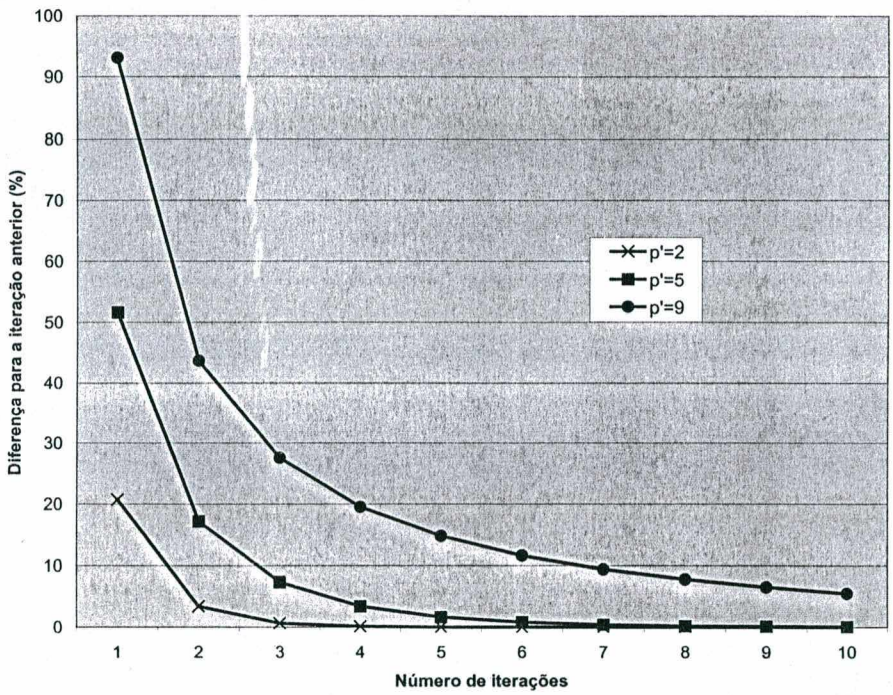


Figura 3.22 – Exemplo 1 com 10 subdivisões – Evolução da precisão obtida no Processo P-Delta

[5] Em cada iteração, as subdivisões apresentarão sempre a mesma força normal, a não ser que haja uma força normal distribuída ao longo do elemento.

3.5 - PROCESSO DAS FUNÇÕES DE ESTABILIDADE

A abordagem utilizada para obter a matriz de rigidez geométrica (item anterior) define como linear a influência da carga normal sobre a rigidez da barra. Isto simplifica o tratamento algébrico do problema, mas obriga a subdivisão da barra para se aproximar da solução correta quando a carga axial aproxima-se da carga crítica. Outra abordagem é a de considerar explicitamente a não linearidade do problema na dedução da matriz de rigidez, através do uso de *funções de estabilidade.*

3.5.1 Dedução das funções de estabilidade

A obtenção das funções de estabilidade e, por conseguinte, da matriz de rigidez da barra, é feita com base na equação diferencial de uma viga. Assumem-se pequenas rotações, com o que pode-se partir da seguinte equação:

$$M = EI\frac{d^2y}{dx^2} \qquad \text{Eq. (3-57)}$$

A dedução apresentada a seguir é sintética e pode ser encontrada por completo em BEAUFAIT et al. (1970).

Funções de rotação

Considere-se um elemento de viga submetido a uma força axial P. Aplicando-se uma rotação unitária φ_i a uma das suas extremidades, obtém-se os esforços de imobilização M_i, M_f, V_i e V_f.

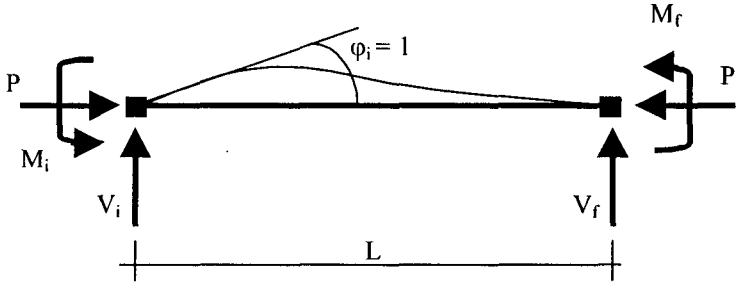


Figura 3.23 – Elemento de barra submetido a uma rotação unitária

A relação entre os momentos fletores e os esforços cortantes pode ser obtida por simples equilíbrio:

$$V_i = -V_f = \frac{M_i + M_f}{L} \qquad \text{Eq. (3-58)}$$

A equação diferencial da viga resulta em:

$$M(x) = EI\frac{d^2y(x)}{dx^2} = -\left(P \cdot y(x) + M_i - V_i \cdot x\right) \qquad \text{Eq. (3-59)}$$

Substituindo-se a Eq. (3-58) na Eq. (3-59), obtém-se:

$$EI\frac{d^2y(x)}{dx^2} = -P \cdot y(x) - M_i + \left(M_i + M_f\right) \cdot \frac{x}{L} \qquad \text{Eq. (3-60)}$$

Para simplificar, pode-se expressar a carga axial P em função da carga crítica de Euler P_E:

$$P = \Phi \cdot P_E \Rightarrow \Phi = P\frac{L^2}{\pi^2 EI} \qquad \text{Eq. (3-61)}$$

$$\frac{d^2y}{dx^2} + \frac{\pi^2 \cdot \Phi}{L^2}y = \frac{\left(M_i + M_f\right) \cdot \frac{x}{L} - M_i}{EI} \qquad \text{Eq. (3-62)}$$

A solução geral da Eq. (3-62) pode ser expressa como:

$$y = A \cdot sen\left(\frac{\mu}{L} \cdot x\right) + B \cdot \cos\left(\frac{\mu}{L} \cdot x\right) + \frac{L^2}{\mu^2 \cdot EI}\left[\left(M_i + M_f\right)\frac{x}{L} - M_i\right] \qquad \text{Eq. (3-63)}$$

Onde:

$$\mu = \pi\sqrt{\Phi} \qquad \text{Eq. (3-64)}$$

Aplicando-se as condições de contorno conhecidas (y=0 para x=0 e y=0 para x=L), pode-se obter o valor das constantes A e B:

$$A = -\frac{L^2}{\mu^2 \cdot EI}\left(M_i \cdot \cot\mu + M_f \cdot \cos ec\,\mu\right)$$
$$B = \frac{L^2}{\mu^2 \cdot EI}M_i \qquad \text{Eq. (3-65)}$$

Substituindo os valores de A e B na Eq. (3-63), obtém-se:

$$\frac{\mu^2 \cdot EI}{L^2} y = -\left(M_i \cdot \cot\mu + M_f \cdot \operatorname{cosec}\mu\right) \cdot \operatorname{sen}\left(\frac{\mu}{L} x\right) + M_i \cdot \cos\left(\frac{\mu}{L} x\right) + \left(M_i + M_f\right)\frac{x}{L} - M_i \qquad \text{Eq. (3-66)}$$

Diferenciando a Eq. (3-66) em relação a x, obtém-se uma expressão para o giro ocorrido ao longo da barra:

$$\frac{\mu^2 \cdot EI}{L} \cdot \frac{dy}{dx} = M_i\left[1 - \mu \cdot \operatorname{sen}\left(\frac{\mu}{L} x\right) - \mu \cdot \cot\mu \cdot \cos\left(\frac{\mu}{L} x\right)\right] + M_f\left[1 - \mu \cdot \operatorname{cosec}\mu \cdot \cos\left(\frac{\mu}{L} x\right)\right] \qquad \text{Eq. (3-67)}$$

Aplicando agora as condições de contorno restantes ($\frac{dy}{dx}$ =0 para x=L e $\frac{dy}{dx}$ =1 para x=0), podem ser obtidos os valores dos momentos fletores:

$$M_f = \frac{\mu - \operatorname{sen}\mu}{\operatorname{sen}\mu - \mu \cdot \cos\mu} M_i \qquad \text{Eq. (3-68)}$$

$$M_i = \frac{\mu \cdot (\operatorname{sen}\mu - \mu \cdot \cos\mu)}{2 \cdot (1 - \cos\mu) - \mu \cdot \operatorname{sen}\mu} \cdot \frac{EI}{L} \qquad \text{Eq. (3-69)}$$

Pode-se definir, a partir das equações acima, duas *funções de rotação* expressas como seguinte:

$$c = \frac{\mu - \operatorname{sen}\mu}{\operatorname{sen}\mu - \mu \cdot \cos\mu} \qquad \text{Eq. (3-70)}$$

$$r = \frac{\mu \cdot (\operatorname{sen}\mu - \mu \cdot \cos\mu)}{2 \cdot (1 - \cos\mu) - \mu \cdot \operatorname{sen}\mu} \qquad \text{Eq. (3-71)}$$

Para o caso de força axial zero, o valor de ambas as funções torna-se indeterminado, uma vez que $\mu = \cos\mu = 0$. Todavia, pode-se calcular o limite das funções, quando μ tende a zero, aplicando-se a regra de L'Hospital, de onde vem $c(0) = \frac{1}{2}$ e $r(0) = 4$. Pode-se verificar que estes valores são os mesmos determinados para a matriz de rigidez sem a presença da carga axial.

Os momentos fletores podem ser expressos em função das funções de rotação *r* e *c*. Os esforços cortantes podem ser obtidos aplicando-se a Eq. (3-58), o que resulta:

$$M_i = r \cdot \frac{EI}{L} \qquad \text{Eq. (3-72)}$$

$$M_f = c \cdot r \cdot \frac{EI}{L} \qquad \text{Eq. (3-73)}$$

$$V_i = -V_f = r \cdot (1+c) \cdot \frac{EI}{L^2} \qquad \text{Eq. (3-74)}$$

Para valores negativos de Φ, ou seja, para cargas axiais de tração, pode-se modificar a Eq. (3-64) da seguinte forma:

$$\nu = \pi\sqrt{\Phi} = i\pi\sqrt{|\Phi|} = i\mu \qquad \text{Eq. (3-75)}$$

Uma vez que ν é um valor complexo, podem ser definidas as seguintes relações:

$$\operatorname{sen}\nu = \frac{e^{i\nu} - e^{-i\nu}}{2i} = \frac{e^{-\mu} - e^{\mu}}{2i} \qquad \text{Eq. (3-76)}$$

$$\cos\nu = \frac{e^{i\nu} + e^{-i\nu}}{2} = \frac{e^{-\mu} + e^{\mu}}{2} \qquad \text{Eq. (3-77)}$$

Aplicando estas relações na Eq. (3-70) e na Eq. (3-71), podem ser obtidas expressões análogas para *r* e *c* em termos de funções hiperbólicas:

$$r = \frac{\mu \cdot (\mu \cdot \cosh\mu - \operatorname{senh}\mu)}{2 \cdot (1 - \cosh\mu) + \mu \cdot \operatorname{senh}\mu} \qquad \text{Eq. (3-78)}$$

$$c = \frac{\mu - \operatorname{senh}\mu}{\operatorname{senh}\mu - \mu \cdot \cosh\mu} \qquad \text{Eq. (3-79)}$$

Desta forma, as funções de rotação podem ser obtidas para o caso de forças de tração com o auxílio das duas equações acima, permanecendo válidas as equações Eq. (3-72) a Eq. (3-74).

Funções de translação

Considerando agora o mesmo elemento de viga submetido a uma força axial P, mas aplicando-se um deslocamento unitário δ_i a uma das suas extremidades, obtém-se os esforços de imobilização M_i, M_f, V_i e V_f.

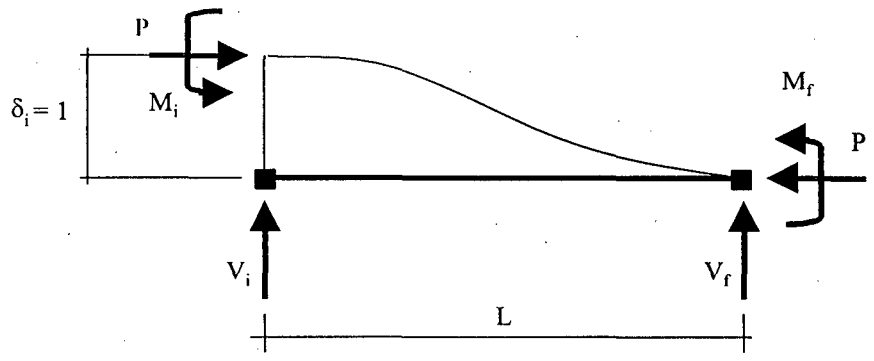


Figura 3.24 –Elemento de barra submetido a uma translação unitária

A obtenção dos esforços pode ser feita substituindo a translação δ_i por duas rotações nodais de $\frac{1}{L}$ nas extremidades. Rotacionando a barra inteira como um corpo rígido com o mesmo ângulo , obtém-se:

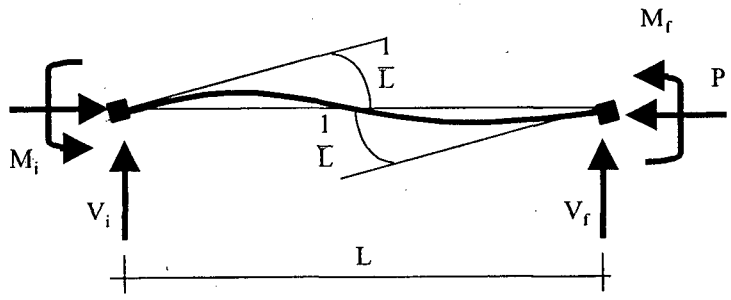


Figura 3.25 – Rotações equivalentes a uma translação unitária

Existe uma diferença entre o ângulo de aplicação da carga P de uma para outra situação. Criteriosamente, o valor da carga axial na segunda situação (Figura 3.25) seria $P.\cos(\frac{1}{L})$. Como esta diferença é muito pequena, visto serem assumidas pequenas rotações, pode ser desprezada.

Fazendo-se o equilíbrio, pode-se obter novamente os esforços cortantes em funções dos momentos fletores. Agora existe uma parcela correspondente ao momento provocado pela carga P com um braço de alavanca unitário:

$$V_i = \frac{(M_i + M_f) - P}{L} \qquad \text{Eq. (3-80)}$$

$$V_f = \frac{P - (M_i + M_f)}{L} \qquad \text{Eq. (3-81)}$$

Aplicando-se as rotações de na Eq. (3-72) e na Eq. (3-73), obtém-se (lembrando que, para uma rotação do nó final, aplicam-se as mesmas equações, mas na forma inversa):

$$M_i = r \cdot \frac{EI}{L} \cdot \frac{1}{L} + r \cdot c \cdot \frac{EI}{L} \cdot \frac{1}{L} \qquad \text{Eq. (3-82)}$$

$$M_f = r \cdot c \cdot \frac{EI}{L} \cdot \frac{1}{L} + r \cdot \frac{EI}{L} \cdot \frac{1}{L}$$

Simplificando, obtém-se:

$$M_i = M_f = r \cdot (1 + c) \cdot \frac{EI}{L^2} \qquad \text{Eq. (3-83)}$$

Substituindo a Eq. (3-83) na Eq. (3-80) e na Eq. (3-81), obtém-se os valores dos esforços cortantes:

$$V_i = -V_f = 2 \cdot r \cdot (1 + c) \cdot \frac{EI}{L^3} - \frac{P}{L} \qquad \text{Eq. (3-84)}$$

Pode-se expressar a carga axial P na equação acima através de uma *função de translação t*, o que resulta em:

$$t = 1 - \frac{\pi^2 \cdot \Phi}{2 \cdot r \cdot (1 + c)} \qquad \text{Eq. (3-85)}$$

Rescrevendo a Eq. (3-84) em função de *t*, tem-se:

$$V_i = -V_f = 2 \cdot t \cdot r \cdot (1 + c) \cdot \frac{EI}{L^3} \qquad \text{Eq. (3-86)}$$

Matriz de rigidez completa

Aplicando-se as funções de rotação e translação já definidas em todos os deslocamentos possíveis da barra, pode-se obter a matriz de rigidez completa de um elemento de viga referido ao seu sistema local:

$$[K] = EI\begin{bmatrix} \frac{2\cdot t\cdot r\cdot(1+c)}{L^3} & \frac{r\cdot(1+c)}{L^2} & -\frac{2\cdot t\cdot r\cdot(1+c)}{L^3} & \frac{r\cdot(1+c)}{L^2} \\ \frac{r\cdot(1+c)}{L^2} & \frac{r}{L} & -\frac{r\cdot(1+c)}{L^2} & \frac{c\cdot r}{L} \\ -\frac{2\cdot t\cdot r\cdot(1+c)}{L^3} & -\frac{r\cdot(1+c)}{L^2} & \frac{2\cdot t\cdot r\cdot(1+c)}{L^3} & -\frac{r\cdot(1+c)}{L^2} \\ \frac{r\cdot(1+c)}{L^2} & \frac{c\cdot r}{L} & -\frac{r\cdot(1+c)}{L^2} & \frac{r}{L} \end{bmatrix} \quad \text{Eq. (3-87)}$$

3.5.2 Efeito P-δ

Esta formulação, visto considerar a influência não linear da carga axial P na rigidez da barra, consegue capturar com exatidão o efeito P-δ ocorrido ao longo de um elemento. Pode-se verificar isto através do mesmo exemplo utilizado no item 3.2.4: considera-se uma barra biapoiada, sujeita a uma força normal compressiva e momentos fletores aplicados em suas extremidades. Reproduz-se abaixo a Figura 3.7:

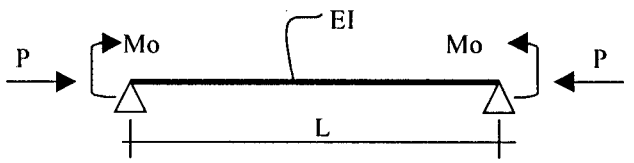


Figura 3.7 – Exemplo 1 – Viga-coluna biapoiada

Consideram-se as seguintes condições de contorno:

- Os deslocamentos axiais d_1 e d_4 não afetam o valor do momento de 2ª ordem, podendo ser ignorados (reduz-se a um problema de viga);
- Conhecem-se os deslocamentos $d_2=0$ e $d_5=0$.

Com isto, tomando-se apenas os deslocamentos desconhecidos d_3 e d_6, tem-se o seguinte sistema:

$$\begin{Bmatrix} -Mo \\ Mo \end{Bmatrix} = EI \cdot \begin{bmatrix} \frac{r}{L} & \frac{c \cdot r}{L} \\ \frac{c \cdot r}{L} & \frac{r}{L} \end{bmatrix} \cdot \begin{Bmatrix} d_3 \\ d_6 \end{Bmatrix} \qquad \text{Eq. (3-88)}$$

Onde *r* e *c* são as funções de rotação definidas na Eq. (3-70) e na Eq. (3-71) para o caso de força axial compressiva.

Substituindo as expressões para *r* e *c* e resolvendo o sistema de equações, podem ser obtidos os deslocamentos incógnitos. Após simplificar as expressões, obtém-se:

$$d_3 = -d_6 = \frac{Mo \cdot L}{EI} \cdot \frac{\operatorname{sen} \mu}{(\cos \mu - 1) \cdot \mu} \qquad \text{Eq. (3-89)}$$

Pode-se demonstrar que esta equação corresponde à solução analítica em termos do giro nas extremidades da barra. O valor do momento final no centro da barra também coincide com a solução analítica, visto que, na obtenção desta, foi adotado exatamente o mesmo procedimento do item 3.5.1.

Comentários

A primeira conclusão que se pode retirar deste problema é que a solução analítica pode ser obtida diretamente, mesmo sem a divisão da barra em elementos menores. Na falta de funções exatas para interpolação, poder-se-ia inserir um nó central, com o que se obteria exatamente o deslocamento real, ou utilizar uma função de interpolação aproximada como a definida pela Eq. (3-50).

O segundo ponto pode ser observado facilmente na equação anterior: a obtenção da carga crítica. O momento fletor final tende ao infinito quando cos $\mu = 1$, ou seja:

$$\mu = \pi \Rightarrow P = \frac{\pi^2 EI}{L^2} \qquad \text{Eq. (3-90)}$$

Este valor corresponde exatamente ao valor da carga crítica de Euler que pode ser obtido através da equação analítica.

Pode-se concluir através deste simples exemplo:

- O Processo das Funções de Estabilidade permite obter a solução analítica sem a necessidade de subdividir a barra;
- O processo permite obter exatamente a carga crítica.

Outro ponto importante pode ser observado verificando-se a variação das funções de rotação *r* e *c* conforme varia o valor de μ (e, por conseguinte, a carga axial P):

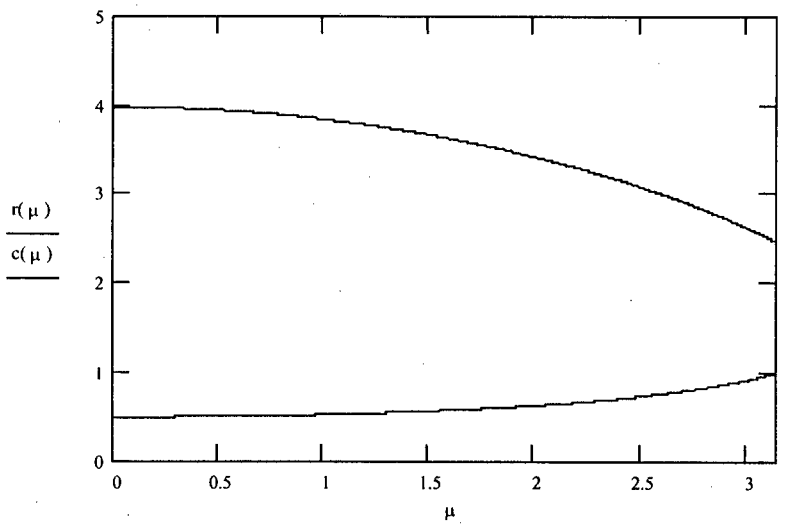


Figura 3.26 –Funções de estabilidade

Pode-se observar que a influência de P sobre a rigidez da barra não é linear, o que explica os resultados significativamente diferentes obtidos com o auxílio da matriz de rigidez geométrica. Observa-se também que esta diferença cresce conforme o valor de μ, ou seja, com o aumento da carga normal.

3.6 - OUTROS PROCESSOS

Além dos três processos apresentados para inclusão da não linearidade geométrica na resolução de pórticos planos (a saber, o Processo P-Delta, no item 3.2, o Processo da Matriz de Rigidez Geométrica, no item 3.3, e o Processo das Funções de Estabilidade, no item 3.5), existem vários outros disponíveis na bibliografia especializada. Estes três primeiros representam os processos mais conhecidos e são usualmente adotados como comparação no desenvolvimento de novos procedimentos. Por este motivo, foram utilizados neste trabalho.

Embora pouco aplicados atualmente, existem diversos métodos simplificados para a consideração do efeito de 2ª ordem nas estruturas. Será feita uma breve revisão sobre estes processos. Além disto, diversos estudos mais recentes fornecem outras formulações mais avançadas, modelando inclusive o comportamento pós-crítico da estrutura. Não se pretende aqui mostrar uma revisão completa destes trabalhos, visto a grande quantidade de material disponível, mas apenas apontar alguns estudos recentes e suas críticas sobre as formulações apresentadas.

3.6.1 Processos simplificados

O efeito P-Delta, segundo o qual os deslocamentos horizontais dos nós da estrutura geram efeitos de 2ª ordem e levam a uma nova configuração deformada, já é conhecido há muito tempo. Sua inclusão no projeto das estruturas sempre foi, por outro lado, algo por demais trabalhoso antes da popularização dos computadores domésticos. Desta forma, desenvolveram-se diversos processos que procuram levar em conta este efeito de forma mais direta possível. Um estudo feito por GAIOTTI & SMITH (1989) reúne, em ordem aproximadamente cronológica, alguns destes processos.

Processo do fator de amplificação

Provavelmente, a primeira abordagem dada ao problema foi a de relacionar os efeitos de 2ª ordem com a carga crítica de flambagem de uma barra. Parte-se de uma barra submetida a uma carga axial e carregamento lateral uniformemente distribuído:

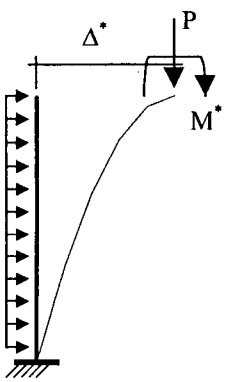


Figura 3.27 – Pilar sujeito a carregamento transversal

Da solução da equação diferencial, o deslocamento final da extremidade da barra pode ser aproximado através da expressão (TIMOSHENHO & GERE, 1961, apud GAIOTTI & SMITH, 1989):

$$\Delta^* = \Delta \cdot \left(\frac{1}{1 - \frac{P}{P_{cr}}} \right) \qquad \text{Eq. (3-91)}$$

Onde:

- Δ^* = deslocamento final na extremidade, incluindo o efeito P-Delta;
- Δ = deslocamento de 1ª ordem na extremidade da barra;
- P = carga axial atuante;
- P_{cr} = carga crítica de Euler.

Esta expressão define um fator de amplificação F que, uma vez aplicado aos deslocamentos obtidos da análise de 1ª ordem, fornece os deslocamentos finais, incluindo o efeito P-Delta.

Este processo pode ser aplicado a estruturas de edificações, obtendo-se os deslocamentos horizontais finais nos diversos andares através de uma variação na Eq. (3-91):

$$\Delta^* = \Delta \cdot \left(\frac{1}{1 - \frac{P_o}{P_{ocr}}} \right) \qquad \text{Eq. (3-92)}$$

Onde:

- P_o = carga vertical total atuante na base da estrutura;
- P_{ocr} = carga crítica da estrutura.

Supõe-se um fator de amplificação constante ao longo de toda a estrutura, tendo sido desenvolvidas algumas expressões aproximadas para o cálculo de P_{ocr}, dependendo do tipo de elemento que compõe o sistema de contraventamento da estrutura, se pilares parede, pórticos ou associações destes.

TIMOSHENHO & GERE (1961) apud GAIOTTI & SMITH (1989) demonstram também que o fator de amplificação dos momentos fletores é o mesmo dos deslocamentos, ou seja,

$$M^* = M \cdot \left(\frac{1}{1 - \frac{P_o}{P_{ocr}}} \right) \qquad \text{Eq. (3-93)}$$

Este processo é bastante aproximado e seu interesse neste trabalho pode ser considerado meramente histórico. Serviu como base, contudo, para outros procedimentos e algumas idéias ainda são aproveitadas para avaliação preliminar dos efeitos de 2ª ordem através do coeficiente de amplificação Gama-Z discutido no item 2.4.4.

Processo do carregamento gravitacional iterativo

Embora o Processo P-Delta já fosse bastante conhecido há duas décadas atrás, era considerado por demais trabalhoso para aplicação em projetos. Desenvolveram-se, portanto, algumas formas mais simplificadas de abordar o problema. Uma delas, proposta por GAIOTTI & SMITH (1989), parte da idéia fundamental de modificar as coordenadas dos nós de uma estrutura para sua posição deformada após a análise de 1ª ordem e realizar nova análise até a convergência. Este procedimento, todavia, pode ser considerado mais trabalhoso que o próprio P-Delta, implicando também em um maior custo computacional.

A proposta consiste em adicionar à estrutura um pilar fictício, com as seguintes propriedades:

- Altura igual à edificação;
- Rigidez axial virtualmente infinita;
- Rigidez à flexão nula.

Este pilar fictício seria conectado à estrutura através de ligações axialmente rígidas (mas sem rigidez à flexão) e todo o carregamento vertical atuante na estrutura seria imposto a este pilar equivalente. Desta maneira, apenas as coordenadas do pilar fictício necessitam ser alteradas a cada análise, obtendo-se basicamente os mesmos resultados mas com um esforço manual menor.

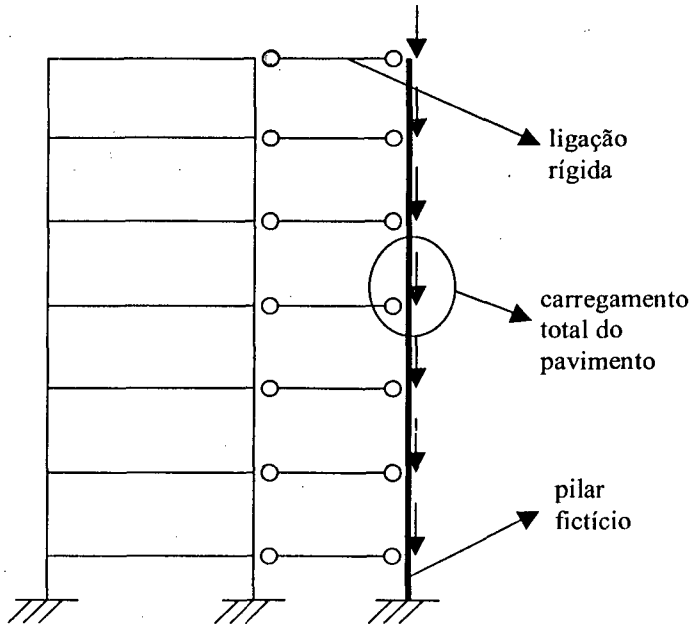


Figura 3.28 – Processo do carregamento gravitacional iterativo

Processo direto

Neste processo, procura-se evitar o procedimento iterativo implícito no processo anterior, através de um ajuste direto dos efeitos de 2ª ordem em cada pavimento, através de um fator baseado na carga vertical e na rigidez lateral do pavimento.

Parte-se das mesmas premissas admitidas para o Processo P-Delta, associando a carga normal com um aumento nos esforços cortantes (cargas horizontais) em cada pavimento:

$$\delta S_i = \frac{P_i \cdot \delta_i^*}{h_i} \qquad \text{Eq. (3-94)}$$

Onde:

- δS_i = variação nos esforços cortantes do pavimento;
- P_i = carga vertical total do pavimento;
- δ_i^* = diferença final entre o deslocamento horizontal inferior e superior do pavimento, incluindo o efeito P-Delta;
- h_i = altura do pavimento.

Quando existem apenas forças horizontais atuando, a rigidez ao cortante de um pavimento pode ser expressa como:

$$K_{si} = \frac{S_i}{\delta_i} \qquad \text{Eq. (3-95)}$$

Onde:

- K_{si} = rigidez ao cortante;
- S_i = esforço cortante total no pavimento;
- δ_i = diferença entre o deslocamento horizontal inferior e superior do pavimento, sem incluir o efeito P-Delta.

O efeito P-Delta pode ser associado ao incremento nos esforços cortantes δS_i indicado na Eq. (3-94), para fornecer uma expressão para o esforço cortante final incluindo os efeitos de 2ª ordem:

$$S_i^* = S_i + \delta S_i = S_i + \frac{P_i \cdot \delta_i^*}{h_i} \qquad \text{Eq. (3-96)}$$

Admitindo que o deslocamento horizontal da estrutura é proporcional apenas ao cortante nos pavimentos, o deslocamento final pode ser expresso, segundo GOLDBERG (1974) apud GAIOTTI & SMITH (1989), da seguinte forma:

$$\delta_i^* = \frac{\left(S_i + \frac{P_i \cdot \delta_i^*}{h_i} \right)}{K_{si}} \qquad \text{Eq. (3-97)}$$

Substituindo o valor de K_{si} dado pela Eq. (3-96) na Eq. (3-97), obtém-se a solução para o deslocamento final em um pavimento:

$$\delta_i^* = \delta_i \cdot \frac{1}{1 - \frac{P_i \cdot \delta_i}{S_i \cdot h_i}} \qquad \text{Eq. (3-98)}$$

O deslocamento lateral total em um pavimento pode ser obtido, portanto, através da somatória dos deslocamentos obtidos em cada pavimento inferior, desde a fundação. Nota-se que o efeito de 2ª ordem será variável ao longo de cada pavimento e que a Eq. (3-98) coincide com a expressão do coeficiente Gama-Z apresentado no item 2.4.4.

As imprecisões inerentes a este processo podem ser explicadas, em parte, por se limitar à análise de estruturas que se deformam primariamente em modo de cisalhamento, desprezando a deformação à flexão.

Processo do membro com propriedade negativa

Este processo, apresentado por RUTENBERG (1981) apud GAIOTTI & SMITH (1989) objetiva evitar os procedimentos iterativos, aplicando uma redução à rigidez da estrutura já na análise de 1ª ordem. Supondo conhecidas as cargas verticais atuantes em cada pavimento, objetiva-se adicionar um pilar fictício à estrutura, com propriedades de rigidez negativas e proporcionais às cargas verticais.

Este pilar fictício é adicionado ao modelo através de ligações rígidas, da mesma maneira como no processo do carregamento gravitacional iterativo (vide Figura 3.28), considerando-se neste uma rigidez ao cisalhamento negativa conforme a expressão:

$$G.A_{si} = -P_i \qquad \text{Eq. (3-99)}$$

Onde:

- $G.A_{si}$ = produto de rigidez ao cisalhamento da barra fictícia;
- P_i = carga vertical atuante no pavimento.

Comentários

Em todos os processos apresentados neste item, associa-se o efeito de 2ª ordem ao comportamento de um pavimento como um todo. Este tipo de procedimento limita-se a estruturas muito regulares, onde a distribuição de rigidez dos pilares ao longo de um pavimento seja bastante uniforme e todo o comportamento da estrutura possa ser reduzido a deslocamentos horizontais rígidos em cada pavimento.

Desta forma, não se incluem estruturas usuais, onde estas variações são significativas. Os demais processos, por se aplicarem à formulação das barras que compõem a estrutura, são muito mais genéricos e podem ser aplicados a diversos tipos de edificações.

3.6.2 Outras formulações para a matriz de rigidez

Nas últimas duas décadas, muitos estudos têm sido feitos acerca da consideração dos efeitos não lineares na análise das estruturas. A popularização do Método dos Elementos Finitos, bem como o avanço exponencial na capacidade de processamento dos computadores disponíveis, fez com que grande número de pesquisadores estendesse os conceitos desenvolvidos nos itens 3.3 e 3.5. Basicamente, o comportamento da estrutura pode ser reduzido ao comportamento das barras que a compõem através da formulação da matriz de rigidez das barras, que pode incluir diversos efeitos além dos apresentados:

- Deformações por cisalhamento;
- Grandes rotações;
- Material não linear;
- Rigidez torsional (estruturas espaciais);
- Comportamento pós-crítico; entre outros.

Mesmo limitando-se às premissas básicas já utilizadas para o Processo da Matriz de Rigidez Geométrica e para o Processo das Funções de Estabilidade, existem diversas considerações na bibliografia especializada, sendo algumas delas abordadas a seguir.

Críticas ao Processo P-Delta

O Processo P-Delta, apresentado no item 3.2, bem como os processos simplificados abordados no item 3.6.1, podem ser considerados à parte desta discussão. Isto porque estes processos representam uma abordagem basicamente de Projeto ao problema da não linearidade, associando a solução ao cálculo de estruturas usuais. Procura-se capturar o comportamento da estrutura como um todo (embora o Processo P-Delta, como já exposto no item 3.2.4, possa ser utilizado até mesmo para capturar o efeito P-δ ocorrido ao longo de cada barra individualmente) e não modelar o comportamento individual de um elemento genérico.

O fato de que o Processo P-Delta modela apenas os deslocamentos nas extremidades das barras força a subdivisão de cada barra em elementos menores para se alcançar maior precisão. Conforme constatado pelos exemplos numéricos apresentados nos itens 5.3 e 5.4, este processo pode incorrer em diferenças muito significativas conforme aumenta-se o nível de força normal. A necessidade de grande nível de discretização pode fazer com que o tempo de processamento, usualmente considerado menor que os demais processos (conforme exposto no item 3.3.6), torne-se maior para um nível de precisão preestabelecido quando comparado, por exemplo, com o Processo das Funções de Estabilidade.

Críticas ao Processo da Matriz de Rigidez Geométrica

Basicamente, a discussão reduz-se à procura de uma formulação para o elemento de barra que resulte em uma matriz de rigidez modificada. Na maior parte dos artigos disponíveis sobre o assunto, o processo da matriz K_G (referenciado usualmente como "the cubic Hermite element" [6]) é considerado a forma mais imediata de incluir a não linearidade geométrica na formulação da estrutura.

Todavia, conforme pôde ser visto através dos exemplos desenvolvidos no item 3.3.4, os erros de precisão apresentados por este processo podem ser significativos, caso aplicado em problemas de determinação da carga crítica de uma estrutura. SO & CHAN (1991) apud CHAN & ZHOU (1994) apontam que este elemento superestima a carga crítica de uma barra simplesmente apoiada por cerca de 21.6%, resultado este também obtido no Exemplo 1. ZHOU & CHAN (1996) e AL-BERMANI & KITIPORNCHAI (1990), da mesma forma como diversos outros autores, apontam para o fato de que, para se obter uma precisão satisfatória, este elemento deve ser dividido em várias partes, aumentando o custo computacional das soluções. Os autores procuram encontrar formulações que eliminem esta dificuldade.

[6] Isto porque baseia-se em estender a formulação conhecida do elemento de viga que utiliza as funções de interpolação cúbicas de Hermite para o caso não linear.

Por outro lado, BATHE & BOLOURCHI (1979) apud CHAN & ZHOU (1994) apontam que este elemento é numericamente bastante estável e pode ser utilizado para análises não lineares correntes (como, por exemplo, aquelas feitas nos projetos usuais) sem grande perda de precisão. Adequadamente abaixo da carga crítica, pode-se chegar a resultados bastante precisos.

Críticas ao Processo das Funções de Estabilidade

Embora o processo desenvolvido no item 3.5 possa obter precisamente a carga crítica usando apenas um elemento por barra (CHAN & ZHOU, 1994), possui a desvantagem de apresentar expressões diferentes para o caso de cargas de compressão e de tração. Em alguns casos, quando a força axial é pequena, pode levar o problema à instabilidade numérica. Conforme colocado por CHEN & TOMA (1994), a singularidade para N=0 causa instabilidade numérica quando a carga axial tende a zero.

Outro ponto destacado por CHAN & ZHOU (1994) refere-se ao fato de que o Processo das Funções de Estabilidade recai na solução de equações de equilíbrio e não constitui em uma abordagem de Elementos Finitos. Consequentemente, muitas técnicas, como a da integração numérica e a variação da seção transversal do elemento ao longo de seu comprimento não podem ser diretamente aplicadas.

Alternativa ao Processo das Funções de Estabilidade

Para evitar o problema da instabilidade numérica quando a carga axial tende a zero, bem como unificar as expressões para a matriz de rigidez nos casos de tração e compressão, GOTO & CHEN (1987a) apud CHEN & TOMA (1994) propõem o uso da expansão em séries das funções trigonométricas e hiperbólicas apresentadas no item 3.5.1, da seguinte forma:

$$\left.\begin{matrix}\operatorname{sen}\gamma L\\ \operatorname{senh}\gamma L\end{matrix}\right\} = \gamma L + \gamma L \cdot \sum_{n=1}^{\infty} \frac{1}{(2\cdot n+1)!}\cdot(-p)^n \qquad \text{Eq. (3-100)}$$

$$\left.\begin{matrix}\cos\gamma L\\ \cosh\gamma L\end{matrix}\right\} = 1 + \sum_{n=1}^{\infty} \frac{1}{(2\cdot n)!}\cdot(-p)^n \qquad \text{Eq. (3-101)}$$

Onde p' representa a mesma relação expressa na Eq. (3-13), ou seja:

$$p' = \frac{P \cdot L^2}{EI} \qquad \text{Eq. (3-13)}$$

Para fins práticos, as séries infinitas devem ser limitadas a um número fixo de termos. Segundo os autores, é suficiente adotar os primeiros dez termos das séries para se chegar a uma solução convergente.

Elemento de ordem superior (PEP)

Diversos autores apresentam formulações mais complexas para os elementos não lineares. Como já exposto, uma das motivações disto é a de utilizar as mesmas expressões para o caso de forças de compressão ou tração, evitando possíveis erros numéricos presentes no Processo das Funções de Estabilidade.

Uma proposta foi apresentada por CHAN & ZHOU (1994), segundo a qual, ao invés de resolver os deslocamentos diretamente através da equação diferencial de equilíbrio (como feito na dedução do Processo das Funções de Estabilidade), impõe-se o equilíbrio apenas no meio do vão. A matriz de rigidez obtida através do Método dos Elementos Finitos é simples, estável e apresenta as mesmas expressões para casos de força normal nula, compressão ou tração. O autor denomina este elemento de *elemento PEP* ("Pointwise Equilibrating Polynomial").

Seja uma barra, sujeita a uma carga normal P e a duas rotações θ_1 e θ_2:

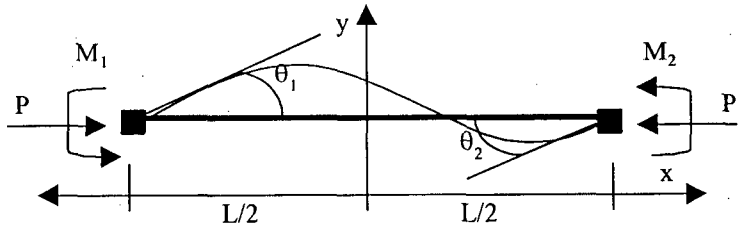


Figura 3.29 – Funções de interpolação considerando a carga normal

Impõe-se, inicialmente, quatro condições de contorno:

- Para $x = -\frac{L}{2}$, $y = 0$ e $y' = \theta_1$
- Para $x = \frac{L}{2}$, $y = 0$ e $y' = \theta_2$

Ao invés de utilizar a equação diferencial de equilíbrio para resolver os deslocamentos y ao longo de todo o comprimento da barra (como feito no item 3.5.1), aplica-se esta condição unicamente ao meio da barra (x = 0). Conforme colocado na Eq. (3-70), a equação completa apresenta-se da seguinte forma:

$$EI\frac{d^2y(x)}{dx^2} = -P\cdot y(x) - M_i + (M_i + M_f)\cdot\frac{x}{L} \qquad \text{Eq. (3-70)}$$

Substitui-se a equação completa por duas condições de contorno adicionais:

$$EI\cdot y'' = P\cdot y + \frac{M_1 + M_2}{L}\cdot\left(\frac{L}{2} + x\right) - M_1 \quad \text{para } x = 0$$

$$EI\cdot y''' = P\cdot y' + \frac{M_1 + M_2}{L} \quad \text{para } x = 0 \qquad \text{Eq. (3-102)}$$

Com base nas cinco condições de contorno definidas, pode-se obter uma função de deslocamentos polinomial de quinta ordem:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 \qquad \text{Eq. (3-103)}$$

Substituindo-se as condições de contorno definidas na Eq. (3-103), podem ser obtidos os deslocamentos:

$$y = \begin{bmatrix} N_1 & N_2 \end{bmatrix}\cdot L\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \qquad \text{Eq. (3-104)}$$

Onde N_1, N_2 são as funções de interpolação dadas por:

$$N_1 = \frac{A}{H_1} + \frac{B}{H_2} \qquad \text{Eq. (3-105)}$$

$$N_2 = \frac{A}{H_1} - \frac{B}{H_2} \qquad \text{Eq. (3-106)}$$

Estas funções são definidas a partir de:

$$A = -20\cdot\frac{x}{L} + (80 - p)\cdot\left(\frac{x}{L}\right)^3 + 4\cdot p\cdot\left(\frac{x}{L}\right)^5 \qquad \text{Eq. (3-107)}$$

$$B = 6 - \left(24 - \frac{p}{2}\right) \cdot \left(\frac{x}{L}\right)^2 - 2 \cdot p \cdot \left(\frac{x}{L}\right)^4 \qquad \text{Eq. (3-108)}$$

$$H_1 = 80 + p \qquad \text{Eq. (3-109)}$$

$$H_2 = 48 + p \qquad \text{Eq. (3-110)}$$

Onde p representa a mesma relação expressa na Eq. (3-13), ou seja:

$$p' = \frac{P \cdot L^2}{EI} \qquad \text{Eq. (3-13)}$$

Pode-se notar que as funções de interpolação reduzem-se às funções de interpolação cúbicas de Hermite (vide item 3.3.1) quando a carga axial é igual a zero. Pode-se continuar o procedimento como no item 3.3.2, a partir do princípio da energia potencial total, para se chegar a uma nova matriz de rigidez para a barra.

Segundo diversos exemplos numéricos elaborados pelos autores desta proposta, este elemento PEP apresenta resultados muito próximos àqueles obtidos através do Processo das Funções de Estabilidade (sendo superior, portanto, ao elemento cúbico utilizado no Processo da Matriz de Rigidez Geométrica), sem apresentar suas características de instabilidade numérica.

A questão das cargas laterais

Outra questão foi levantada por ZHOU & CHAN (1996): a idealização, admitida em todos os outros desenvolvimentos, de que as forças aplicadas são nodais, o que não é aplicável a problemas reais. Uma vez que o princípio da superposição das ações não é válido em análises não lineares, a utilização de cargas nodais equivalentes a um carregamento distribuído ao longo da barra pode levar a erros consideráveis.

Além da consideração das cargas laterais, permanece a preocupação, expressa no item anterior pelos mesmos autores, de incluir a própria carga normal na dedução das funções de interpolação.

Seja uma barra, sujeita a uma carga normal P, a duas rotações θ_1 e θ_2 e também a um carregamento distribuído w(x):

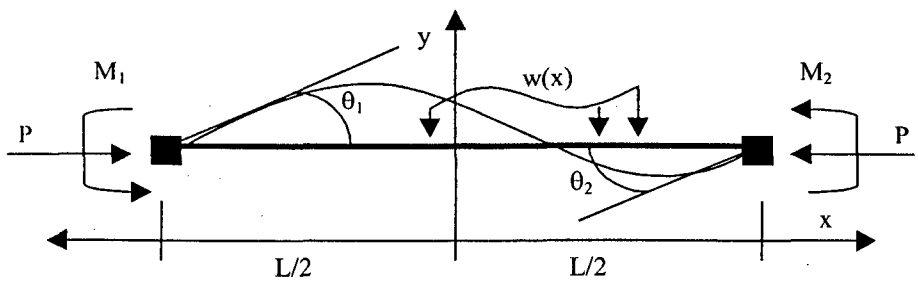


Figura 3.30 – Funções de interpolação considerando a carga normal e carregamento lateral

Impõe-se as quatro condições de contorno referentes à compatibilidade e deseja-se impor a condição de equilíbrio apenas no meio do vão, exatamente como feito no item anterior. As condições de contorno adicionais agora apresentam-se da seguinte forma:

$$EI \cdot y'' = P \cdot y + \frac{1}{2} \cdot (M_1 - M_2) - M_0 \quad \text{para } x = 0$$

$$EI \cdot y''' = P \cdot y' + \frac{M_1 + M_2}{L} \quad \text{para } x = 0 \qquad \text{Eq. (3-111)}$$

Onde:

- M_0 = momento causado no meio do vão pela ação da carga lateral w(x).

Substituindo-se as condições de contorno definidas na mesma Eq. (3-103), podem ser obtidos os deslocamentos:

$$y = N_1 \cdot L \cdot \theta_1 + N_2 \cdot L \cdot \theta_2 + N_w \cdot L \cdot \overline{M}_0 \qquad \text{Eq. (3-112)}$$

Onde N_1, N_2 são as funções de interpolação que se apresentam da mesma forma como na Eq. (3-105) e na Eq. (3-106):

$$N_1 = \frac{A}{H_1} + \frac{B}{H_2} \qquad \text{Eq. (3-105)}$$

$$N_2 = \frac{A}{H_1} - \frac{B}{H_2} \qquad \text{Eq. (3-106)}$$

Utilizando-se agora um parâmetro adimensional *t* no lugar de *x*, estas funções são definidas a partir de:

$$A = -\frac{1}{8}\cdot t\cdot\left(1-t^2\right)\cdot(p+80)+\frac{1}{8}\cdot t\cdot\left(1-t^2\right)^2\cdot p \qquad \text{Eq. (3-113)}$$

$$B = \frac{1}{8}\cdot\left(1-t^2\right)\cdot(p+48)-\frac{1}{8}\cdot\left(1-t^2\right)^2\cdot p \qquad \text{Eq. (3-114)}$$

Onde H_1, H_2 e p são definidas da mesma forma como no item anterior, pelas equações Eq. (3-109), Eq. (3-110) e Eq. (3-13), respectivamente. Apresentam-se ainda:

$$\overline{M_0} = \frac{M_0\cdot L}{EI} \qquad \text{Eq. (3-115)}$$

$$N_w = -\frac{(1-t^2)^2}{H_2} \qquad \text{Eq. (3-116)}$$

Pode-se notar que as funções de interpolação reduzem-se novamente às funções de interpolação cúbicas de Hermite (vide item 3.3.1) quando tanto a carga axial P como a carga lateral w(x) forem iguais a zero.

Pode-se continuar o procedimento como no item 3.3.2, a partir do princípio da energia potencial total, para se chegar a uma nova matriz de rigidez para a barra.

Como verificação, supõe-se uma barra sujeita a uma carga axial P e a uma carga concentrada Q em seu centro:

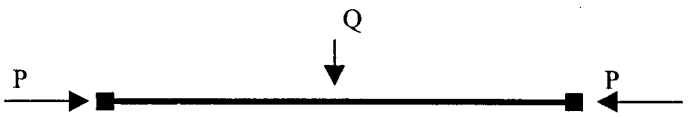


Figura 3.31 – Barra sujeita a carregamento concentrado

A solução analítica fornece a seguinte expressão para o deslocamento ao longo da barra:

$$y = \frac{Q}{2 \cdot P \cdot k} \cdot \left[\frac{\operatorname{sen}(k \cdot x)}{\cos\left(\frac{k \cdot x}{2}\right)} - k \cdot x \right] \qquad \text{Eq. (3-117)}$$

$$\text{Onde:} \qquad k = L \cdot \sqrt{\frac{P}{EI}}$$

Para uma relação entre a carga lateral a e axial de 0.1 (Q/P = 0.1), obtém-se um erro de 22.3% através do Processo da Matriz de Rigidez Geométrica e de 4.1% utilizando as funções de estabilidade, para um erro de apenas 0.04% com o processo proposto. Elevando-se a relação Q/P para 0.2 e 1, mesmo o Processo das Funções de Estabilidade apresenta erros de 7% e 17%, respectivamente.

Observa-se que, em uma situação como esta, o uso dos processos estudados incorreria em erro inaceitável. Felizmente, são raras as situações de projeto onde ocorrem concomitantemente grandes cargas axiais e laterais. Como comparação, pode-se lembrar que a carga lateral equivalente a imperfeições geométricas em pilares resulta em relações Q/P da ordem de 1/400 a 1/200.

Uma situação de interesse refere-se aos carregamentos horizontais provenientes do efeito do vento sobre a estrutura. Os autores mostram que a diferença obtida em utilizar-se formulações simplificadas, onde as cargas são transferidas para os nós, pode ser considerada significativa mesmo em condições de projeto.

3.6.3 Processo da carga lateral fictícia

O processo da carga lateral fictícia[7] apresentado por LUI & ZHANG (1990) merece um destaque em separado por apresentar-se de forma significativamente diferente das demais propostas, que baseiam-se sempre em obter novas formulações para a matriz de rigidez da barra. Ao invés disto, o processo proposto dedica-se a modelar os efeitos de 2ª ordem através de cargas laterais fictícias, de forma semelhante ao feito para o Processo P-Delta.

[7] "Pseudo load method", no original.

Base do método

Este processo foi desenvolvido com base na semelhança entre a equação diferencial de uma viga-coluna e uma viga. Para uma viga-coluna, a equação diferencial tem a seguinte forma:

$$EI \cdot \frac{d^4y}{dx^4} + P \cdot \frac{d^2y}{dx^2} = w \qquad \text{Eq. (3-118)}$$

Onde:

- y = y(x) = deslocamento ao longo da barra;
- w = w(x) = carga lateral distribuída ao longo da barra;

Pode-se rearranjar a Eq. (3-118) da seguinte forma:

$$EI \cdot \frac{d^4y}{dx^4} = \overline{w} \qquad \text{Eq. (3-119)}$$

Onde:

$$\overline{w} = w - P \cdot \frac{d^2y}{dx^2} \qquad \text{Eq. (3-120)}$$

Para pequenos deslocamentos, pode-se dizer que:

$$\frac{d^2y}{dx^2} = -\frac{M}{EI} \qquad \text{Eq. (3-121)}$$

Desta forma, pode-se reescrever a Eq. (3-120) da seguinte forma:

$$\overline{w} = w - P \cdot \frac{M}{EI} \qquad \text{Eq. (3-122)}$$

Onde:

- M = M(x) = momento ao longo da barra.

Verificando que a Eq. (3-119) representa a equação diferencial de uma viga, pode-se concluir que a não linearidade geométrica ao longo da barra (efeito P-δ) pode ser simulada através da aplicação de cargas laterais fictícias com o valor $\frac{PM}{EI}$. Esta carga fictícia pode ser obtida multiplicando-se o diagrama de momentos fletores M(x) obtido através da análise de 1ª ordem pelo fator $\frac{P}{EI}$. Uma nova análise deve ser efetuada, com as barras submetidas às cargas laterais acrescidas das cargas fictícias. Este processo precisa ser repetido iterativamente até que não ocorra diferença significativa nos momentos fletores entre duas iterações.

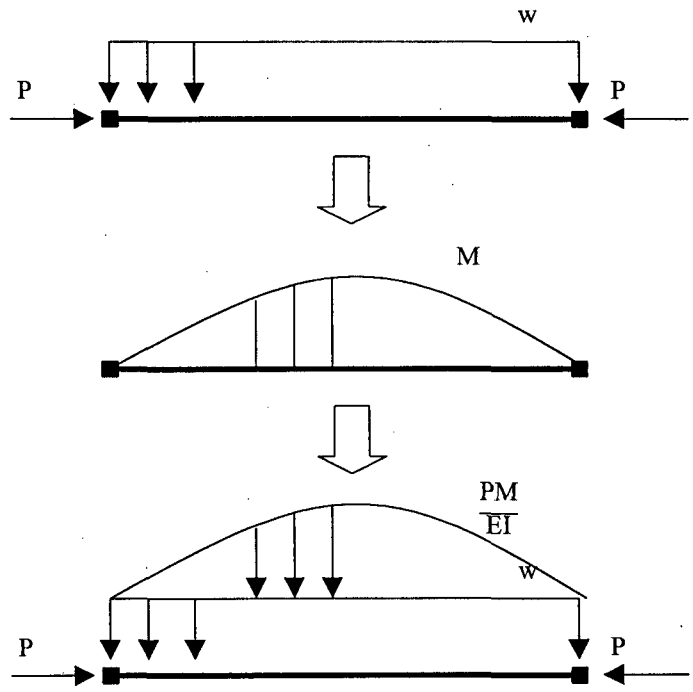


Figura 3.32 – Processo da carga lateral fictícia

Exemplo 1

Como ilustração do processo, pode-se tomar novamente o Exemplo 1: considera-se uma barra biapoiada, sujeita a uma força normal compressiva e momentos fletores aplicados em suas extremidades.

Conforme já colocado, a solução analítica deste problema fornece que o momento máximo, considerando a 1ª e 2ª ordem, ocorre no meio do vão, com o valor:

$$M_f = M_0 \cdot \sec\left(\frac{L}{2}\sqrt{\frac{P}{EI}}\right) \qquad \text{Eq. (3-2)}$$

Aplicando-se o processo proposto, será obtido, após três iterações, o seguinte valor para o momento no meio do vão:

$$M_f = M_o + \frac{1}{8}\cdot\frac{P\cdot M_0\cdot L^2}{EI} + \frac{5}{384}\cdot\frac{P^2\cdot M_0\cdot L^4}{(EI)^2} \qquad \text{Eq. (3-123)}$$

Pode-se verificar que a Eq. (3-123) corresponde exatamente à expansão em série de Taylor da Eq. (3-2). Comparando-se com os resultados obtidos para este exemplo através do Processo P-Delta (item 3.2.4), do Processo da Matriz de Rigidez Geométrica (item 3.3.4) e do Processo das Funções de Estabilidade (item 3.5.2), pode-se verificar que o processo proposto supera os dois primeiros em termos de precisão, com resultados muito próximos do Processo das Funções de Estabilidade.

Sistematização do processo

À primeira vista, pode-se dizer que as cargas fictícias viriam a induzir reações de apoio adicionais à estrutura. Estas reações adicionais (que não existem) podem ser eliminadas incluindo-se à formulação esforços cortantes fictícios nas extremidades de cada elemento. Estes esforços fictícios podem ser obtidos através do produto da carga axial P pela rotação na extremidade correspondente (conforme pode ser deduzido por integração).

Com isto, o equilíbrio torna-se garantido no interior dos elementos. Por outro lado, o equilíbrio pode ser violado nos nós da estrutura caso os cortantes fictícios em duas barras adjacentes não se anulem. Pode-se garantir o equilíbrio global acrescentando-se aos nós cargas fictícias correspondentes à diferença entre os esforços cortantes fictícios das barras adjacentes. Estas cargas simulam a não linearidade geométrica da estrutura como um todo (efeito P-Δ).

Pode-se notar como vantagens deste processo:

Com isto, o equilíbrio torna-se garantido no interior dos elementos. Por outro lado, o equilíbrio pode ser violado nos nós da estrutura caso os cortantes fictícios em duas barras adjacentes não se anulem. Pode-se garantir o equilíbrio global acrescentando-se aos nós cargas fictícias correspondentes à diferença entre os esforços cortantes fictícios das barras adjacentes. Estas cargas simulam a não linearidade geométrica da estrutura como um todo (efeito P-Δ).

Pode-se notar como vantagens deste processo:

- As iterações são efetuadas apenas sobre o vetor de carregamentos, mantendo inalterada a matriz de rigidez da estrutura, o que reduz muito o custo computacional envolvido na solução (análogo ao Processo P-Delta, conforme discutido no item 3.3.6);
- Os resultados independem da discretização do elemento, uma vez que o diagrama de momentos fletores, bem como o fator $\frac{P}{EI}$, não variará caso a barra seja subdividida;
- A precisão obtida nos resultados é comparável à do Processo das Funções de Estabilidade.

CAPÍTULO 4.
ASPECTOS DE IMPLEMENTAÇÃO ORIENTADA A OBJETOS

4.1 - FUNDAMENTOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

A implementação computacional dos procedimentos descritos no CAPÍTULO 2 pode ser considerada clássica. Listagens completas, em sua maior parte escritas em linguagem FORTRAN, podem ser encontradas, entre outros, em WEAVER & GERE (1965) e em BEAUFAIT et al. (1970).

Tendências modernas de implementação, todavia, exigem uma maior produtividade e adaptabilidade no código escrito. A reutilização deve ser feita de maneira rápida e simples. O desenvolvimento de interfaces gráficas requer "módulos" de solução estanques e facilmente intercambiáveis. Esta tendência tem disseminado o uso da Programação Orientada a Objetos (OOP) em lugar da Programação Estruturada convencional.

Por outro lado, os desenvolvimentos a nível acadêmico ainda pouco utilizam estas tendências atuais. Pode-se dizer que, nos dias de hoje, o maior número de profissionais da área de pesquisa ainda trabalha com metodologias clássicas e a tradicional linguagem FORTRAN. O grande volume de código já elaborado, embora apresente certos problemas de reaplicabilidade, induz os novos trabalhos a seguir os mesmos procedimentos. Por este motivo, optou-se por implementar este trabalho desde o início, sem reaproveitar rotinas prontas de análise matricial, sinalizando aos novos trabalhos um caminho mais moderno e eficiente de se fazer uma implementação computacional.

Foge ao escopo deste trabalho uma explanação completa sobre os princípios da OOP, mas, sinteticamente, o principal objetivo desta abordagem é criar componentes de software estanques chamados *objetos*. Um aplicativo será composto, portanto, por um conjunto de objetos.

4.2 - DEFINIÇÕES

A implementação dos conceitos fundamentais da OOP varia de linguagem para linguagem. Todavia, a sintaxe é bastante semelhante e a funcionalidade obtida é basicamente a mesma. Neste trabalho, optou-se por utilizar a linguagem C++, através do compilador Borland C++ versão 5.0. A seguir, descrevem-se os principais elementos de suas terminologia.

4.2.1 Objetos

Um *objeto* é uma entidade composta por dados e métodos. Diferentemente da programação convencional, onde o desenvolvedor precisa representar dados e procedimentos separadamente e gerenciar sua interação, a OOP encapsula tipos específicos de dados com os métodos específicos para operar sobre estes. Ou seja, projetam-se objetos que simulam um "comportamento" definido.

Pode-se criar um objeto agrupando diferentes objetos a seu conjunto de dados. Este novo objeto utiliza a funcionalidade fornecida por seus sub-objetos. A abstração fornecida pelo conceito de objetos pode ser estendida a um aplicativo completo. Um aplicativo composto por objetos (por exemplo, um módulo de resolução) pode também ser um objeto e ser incluído em uma aplicação maior.

4.2.2 Classes

Grupos de objetos que têm o mesmo tipo de dados e procedimentos são chamados uma *classe*. Desta forma, os objetos podem ser definidos como *instâncias* de uma classe.

A classe armazena as definições do modelo de dados utilizado por suas instâncias, bem como dos métodos associados.

4.2.3 Métodos

Os procedimentos referentes a uma classe são chamados *métodos*. Os métodos assemelham-se às sub-rotinas existentes na Programação Estruturada, mas com a diferença fundamental de serem aplicados sobre uma classe. Desta forma, um método só pode ser utilizado como parte de um objeto existente. Dentro da implementação do método, estão acessíveis não apenas os argumentos de entrada, mas também todos os dados do objeto ao qual pertence.

4.3 - VANTAGENS

Existem diversas vantagens referentes ao uso da OOP. Pode-se citar como algumas delas o que está a seguir.

4.3.1 Ocultamento de dados

Uma das características importantes da OOP é o fato de que um objeto possui duas áreas de armazenamento de dados e métodos:

- parte pública: pode ser acessada externamente, consistindo na forma como o objeto comunica-se com o "meio externo".
- parte privada: pode ser acessada apenas pelo código pertencente à própria classe. Incluem-se aqui todas as características peculiares à implementação específica do objeto.

Usualmente, os dados do objeto são armazenados em sua parte privada. Desta forma, podem ser acessados (ou não) apenas pelos métodos que forem definidos como públicos. Isto significa também que a implementação de um objeto pode ser modificada facilmente sem afetar o restante do programa. Com a padronização adequada da parte pública dos objetos, estes podem ser facilmente substituídos por outros de implementação completamente diferente.

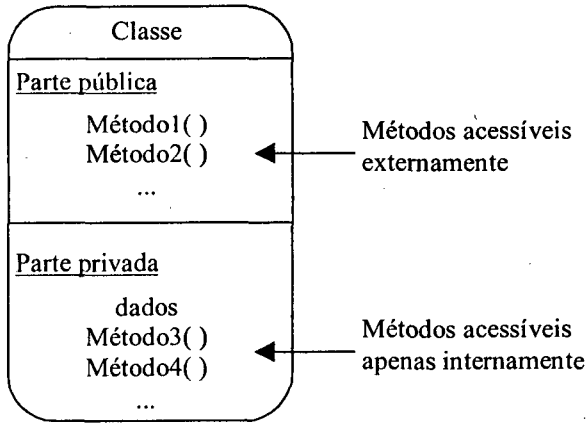


Figura 4.1 – Parte pública e privada de uma classe

4.3.2 Reaproveitamento

Na programação estruturada convencional, é possível o reaproveitamento de código na evolução de um programa desenvolvido ou na criação de outro que aproveite uma de suas partes. A divisão do programa em *módulos* e *sub-rotinas* permite, de certa forma, uma organização dos procedimentos e, com isto, permite ao desenvolvedor:

- reaproveitamento em todo de algumas sub-rotinas, sem modificação
- reaproveitamento em parte de algumas sub-rotinas, com modificações
- alteração do gerenciamento das sub-rotinas, pela inclusão ou exclusão de dados globais (parâmetros)
- criação de novas sub-rotinas

Isto constitui-se no que pode ser chamado de *reaproveitamento de código*. Em suma, toma-se código fonte escrito (seja todo ou, caso possível, apenas alguns módulos relevantes), duplica-se o que for necessário e faz-se as modificações pertinentes. Este tipo de procedimento possui algumas claras dificuldades:

- A organização do código e agrupamento das sub-rotinas em módulos é bastante subjetiva, não possuindo uma metodologia padrão e variando simplesmente de programador para programador.
- Mesmo as sub-rotinas comuns, reaproveitadas sem qualquer modificação, devem ser duplicadas, dificultando a manutenção simultânea no futuro. A elaboração de bibliotecas é algo bastante deficiente, visto serem apenas uma organização física de funções, sem funcionalidade realmente estanque.

- O aproveitamento realmente integral (sem modificações) das sub-rotinas é difícil, visto não ser esta a preocupação da Programação Estruturada.

Em contraponto, a Programação Orientada a Objetos parte de uma filosofia na qual o código deve ser estanque, ou seja, uma classe deve representar exatamente o problema ao qual se propõe. O ponto de partida é a criação de código reaproveitável. Pode-se comparar:

- a organização das classes é algo sistemático, utilizando-se de regras gerais que facilitam a padronização entre os programadores;
- pode-se aproveitar sempre a classe como um todo, sem qualquer alteração no código fonte;
- com base apenas nas definições de classe (*headers*), pode-se utilizar até mesmo o código original já compilado ao invés do código fonte, impedindo o acesso ao código original (este procedimento é muito comum na distribuição de bibliotecas comerciais desenvolvidas para fazer funções específicas).

Sintetizando, pode-se comparar:

Programação Estruturada	Programação Orientada a Objetos
- reaproveitamento de código	- reaproveitamento de classes
- estilo de programação voltado ao controle de fluxo, mas sem padronização	- estilo de programação voltado à separação das classes, funcionando como uma padronização para seu reaproveitamento
- necessidade de duplicação do código fonte, reutilizando rotinas em todo ou em parte	- possibilidade de utilização de classes inteiras, até mesmo com código já compilado
- dificuldade de manutenção posterior	- separação e facilidade de manutenção

Tabela 4.1 – Comparação entre as técnicas de reaproveitamento de código

4.4 - SINTAXE

Existem pequenas diferenças de sintaxe no que se refere à utilização da OOP em comparação com a Programação Estruturada. Para melhor expor as diferenças, optou-se por um exemplo prático:

- supõe-se a criação de um programa cujo objetivo seja o desenho de linhas no vídeo;
- a entrada de dados de pontos define as coordenadas (X, Y) de dois ou mais pontos;
- a entrada de dados define, para cada linha, o número dos seus nós e sua cor;
- Para a parte estruturada, utilizar-se-á de código em linguagem BASIC[8];
- Para a parte orientada a objetos, utilizar-se-á de código em linguagem C++[9], como feito para o desenvolvimento do *software* associado.

4.4.1 Programação Estruturada

Mesmo a Programação Estruturada possui mecanismos para organização de seus dados. Dados complexos podem ser agrupados na forma de *registros*, muito embora este estilo de programação seja pouco aproveitado na maior parte das publicações técnicas. Vide, por exemplo, WEAVER & GERE (1965), BEAUFAIT et al. (1970) e ZERMIANI (1998). Pode-se definir estruturas que representam pontos e linhas:

```
TYPE Ponto
     x : SINGLE
     y : SINGLE
END TYPE

TYPE Linha
     cor : INTEGER
     pInicial : INTEGER
     pFinal : INTEGER
END TYPE
```

Pode-se notar que a definição, embora mostre algo sobre a organização dos dados, tem como deficiências:

- representam-se apenas os dados referentes ao objeto e não seu comportamento;
- a representação é simples e limitada aos tipos básicos da linguagem (valores inteiros, caracteres ou de ponto flutuante);
- a representação dos pontos no registro Linha é feita de forma indireta, através de simples números que indexam um vetor externo.

[8] Pode-se dizer que esta consiste no principal estilo de linguagem estruturada em ambiente PC, enquanto que o FORTRAN é principal estilo de linguagem estruturada em estações de trabalho e computadores de grande porte.

[9] Pode-se dizer que esta divide com o Pascal – leia-se Borland Delphi –o título de linguagem OOP mais usada em ambiente PC.

O programa principal (sem explicitar as sub-rotinas) apresentaria-se aproximadamente como o seguinte:

```
' Número de elementos
DIM nPontos AS INTEGER
DIM nLinhas AS INTEGER
CALL LeVarGlobais()

' Vetores de elementos
DIM Pontos(nPontos) AS Ponto
CALL LePontos(nPontos, Pontos())
DIM Linhas(nLinhas) AS Linha
CALL LeLinhas(nLinhas, Linhas())

 ' Desenho das linhas
FOR i = 1 to nLinhas
  xi = Linhas(i).Pontos(Linhas(i).pInicial).x
  yi = Linhas(i).Pontos(Linhas(i).pInicial).y
  xf = Linhas(i).Pontos(Linhas(i).pFinal).x
  yf = Linhas(i).Pontos(Linhas(i).pFinal).y
  cor = Linhas(i).cor
  CALL DesenhaLinha(xi, yi, xf, xy, cor)
NEXT i
END
```

Deve-se lembrar ainda que o estilo de programação normalmente utilizado sequer faz uso dos registros, apresentando os dados apenas sob a forma de vetores. Neste caso, seria obtido algo como o seguinte:

```
' Número de elementos
DIM nPontos AS INTEGER
DIM nLinhas AS INTEGER
CALL LeVarGlobais()

' Vetores de elementos
DIM Pontos(nPontos,2) AS SINGLE
CALL LePontos(nPontos, Pontos())
DIM Linhas(nLinhas, 3) AS SINGLE
CALL LeLinhas(nLinhas, Linhas())

 ' Desenho das linhas
FOR i = 1 to nLinhas
  xi = Pontos(Linhas(i,1),1)
  yi = Pontos(Linhas(i,1),2)
  xf = Pontos(Linhas(i,2),1)
  yf = Pontos(Linhas(i,2),2)
  cor = Linhas(i,3)
  CALL DesenhaLinha(xi, yi, xf, xy, cor)
NEXT i
END
```

Tal estilo de programação apresenta claros problemas de legibilidade. A representação de dados complexos através de vetores é prática comum e depende de uma documentação muito clara para que possa ser compreendida por outro que não o próprio programador.

4.4.2 Programação orientada a objetos

No caso de se utilizar a OOP, parte-se, como requisito fundamental, da modelagem dos dados na forma de pontos e linhas. Pode-se ainda separar em classes a representação física da própria cor ao invés de usar apenas um número. Estas classes poderiam se apresentar como seguinte:

```
class Color
{
  public:
     Color(int _numero);        // construtor
     int NumeroCor();        // retorna o número da cor
     string TextoCor();      // retorna o nome da cor
     void Numero(int _numero);  // atribui novo valor

  protected;
     int numero;
};

class Ponto
{
  public:
     Ponto();     // construtor
     float X();   // retorna o valor de x
     float Y();   // retorna o valor de y
     void LeDoArquivo(char* arq);

  protected:
     float x, y;
};

class Linha
{
  public:
     Linha(Ponto* _pInicial, Ponto* _pFinal);     // construtor
     Ponto* PInicial();  // retorna o ponto inicial
     Ponto* PFinal();    // retorna o ponto final
     Color Cor();        // retorna a cor

     void LeDoArquivo(char* arq);
     void Desenha();

  protected:
     Ponto* pInicial, pFinal;
     Color cor;
};
```

Pode-se notar alguns pontos fundamentais da OOP mesmo mesta implementação simples:

- cada classe contém tanto a representação de seus dados como métodos que definem o seu comportamento;
- os dados da classe estão contidos na parte privada da classe, ficando seu acesso condicionado à existência de métodos na parte pública;

- a classe Linha possui uma representação dos seus pontos inicial e final ao invés de simples índices de vetor.

O programa principal também é uma classe e poderia se apresentar como seguinte:

```
class MyApp
{
  public:
     Executa();

  protected:
     // Métodos
     void LeArquivo(char* arq);
     void Desenha();

     // Dados
     Array<Ponto> pontos;
     Array<Linha> linhas;
};

void MyApp::Desenha()
{
  int nLinhas = linhas->GetItemsInContainer();
  for (int i=1; i <= nLinhas; i++)
     (*linhas)[i]->Desenha();
}
```

Complementando, o método Desenha da classe Linha poderia se apresentar como seguinte (considerando que existe uma classe Screen que encapsula as funções de desenho no vídeo):

```
void Linha::Desenha()
{
  float xi = pInicial->X();
  float yi = pInicial->Y();
  float xf = pFinal->X();
  float yf = pFinal->X();
  Screen dc;
  dc.DesenhaLinha(xi, yi, xf, yf, cor.Numero());
}
```

Pode-se notar:

- não é necessário, no escopo da aplicação, o conhecimento da estrutura interna da classe Linha nem da forma que esta utiliza para se desenhar no vídeo;
- no escopo da classe Linha, tem-se acesso aos seus nós inicial e final e não ao vetor completo;
- pode-se modificar facilmente a classe Linha (por exemplo, incluir a espessura da linha) sem modificar nenhuma das outras classes.

4.4.3 Ponteiros e objetos

Pode-se notar, na implementação da classe Linha, uma diferença entre seus tipos de dados:

```
protected:
     Ponto* pInicial, pFinal;
     Color cor;
```

A declaração Ponto seguida por "*" indica que esta dado é um *ponteiro* para um objeto[10]. A ausência deste indicador na declaração Color indica que este dado é um objeto.

Um ponteiro é um tipo especial de dado que fornece o endereço de memória onde o objeto está localizado. Desta forma, pode-se utilizá-lo como se fosse o próprio objeto. Na sintaxe C++, existe uma pequena diferença no uso do dado:

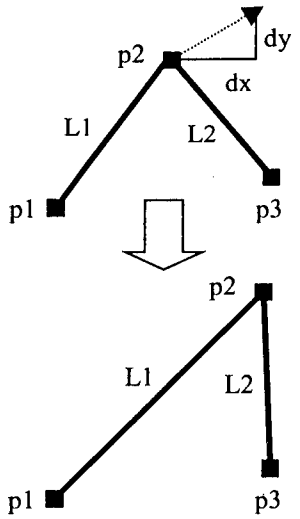
```
Ponto p1;
int x = pi.X();        // usa-se o '.'

Ponto* p2;
int x2 = p2->X();      // usa-se o "->"
int x3 = (*p2).X();    // ou "derreferencia-se" o objeto
```

Usa-se ponteiros quando não se deseja criar uma cópia de objetos que já existem externamente. Neste caso, o objeto MyApp possui um vetor de pontos lido do arquivo, bem como um vetor de linhas. Caso fosse utilizada uma referência comum, a linha teria uma *cópia* dos seus pontos extremos. Supondo que uma segunda parte do processo desejasse mudar a posição dos pontos informados:

- Usando referências, cada linha teria de ser modificada. O mesmo ponto, pertencente a várias linhas, seria alterado várias vezes (pois teriam sido criadas cópias).
- Usando ponteiros, seria suficiente modificar os próprios pontos. Uma vez que as linhas apenas *apontam* para os pontos, não teriam que ser modificadas.

O uso de ponteiros consiste em ferramenta importante da OOP e é uma das responsáveis pela clareza do código. Desta forma, pode-se separar mais facilmente os problemas. Neste caso, pode-se considerar mais lógico aplicar a modificação nos nós do que nas linhas.

[10] Estas indicações referem-se especificamente à sintaxe C++. Outras linguagens possuem formas diferentes de tratar as referências a objetos.




```
void MyApp::MovePonto(Ponto* p,
float dx, float dy)
{
  p->Move(dx, dy);
  // não é necessário mudar as
linhas
  l1->Desenha();
  l2->Desenha();
}
```

Figura 4.2 – Exemplo de uso de ponteiros

4.5 - HERANÇA

O conceito de *herança* funciona como base de todas as técnicas de reutilização de classes. Um classe pode ser definida a partir de uma classe base (ou ancestral), herdando desta todos os seus dados e métodos.

A partir disto, podem ser definidos novos dados ou métodos. Aqueles existentes podem ser modificados, inclusive substituindo métodos por outros de mesmo nome e parâmetros, mas com funcionalidade diferente. Quando um método é chamado para uma classe, o compilador verifica se esta está implementado na própria classe; caso não o encontre, procura na classe base anterior, depois na próxima e assim por diante. Desta forma, métodos podem ser *sobrescritos* ou adicionados.

Aproveitando o exemplo anterior, pode-se definir um nova classe LinhaEspessa, baseada em Linha mas com a definição de sua espessura.

```
class LinhaEspessa : public Linha
{
  public:
     LinhaEspessa(Ponto* _pInicial, Ponto* _pFinal);
       // construtor
     float Espessura();

     virtual void LeDoArquivo(char* arq);
     virtual void Desenha();

  protected:
     float espessura;
};
```

Pode-se notar:

- apenas os dados adicionais devem ser especificados;
- novos métodos (por exemplo, acesso a novos dados) podem ser adicionados;
- métodos sobrescritos são precedidos do identificador *virtual*.

Os métodos modificados seriam, neste caso, a leitura do arquivo (existe um dado adicional) e a função de desenho. Por exemplo:

```
void LinhaEspessa::Desenha()
{
  float xi = pInicial->X();
  float yi = pInicial->Y();
  float xf = pFinal->X();
  float yf = pFinal->X();
  Screen dc;
  dc.DesenhaLinha(xi, yi, xf, yf, cor.Numero(), espessura);
}
```

Nota-se que a classe LinhaEspessa utiliza indistintamente os dados da classe base Linha e seus próprios dados. Isto pode ser permitido ou não pelo programador da classe original, definindo seus dados como:

- *public*: podem ser utilizados externamente;
- *protected*: podem ser utilizados apenas pelas classes derivadas; ou
- *private*: não podem ser acessados em nenhuma situação.

4.6 - POLIMORFISMO

Outra característica fornecida pela OOP é a de permitir a utilização de diferentes classes complexas no mesmo contexto de uma classe base única. No exemplo apresentado, o programa tornaria-se ainda mais genérico se fosse capaz de lidar com tipos diferentes de objetos. Além dos pontos, podem ser desenhadas linhas, poligonais, círculos e muitos outros. Muito embora estes elementos sejam diferentes em forma, possuem alguns aspectos comuns que podem ser *encapsulados* em uma classe base. Por exemplo, poder-se-ia definir:

```
class ElementoGrafico
{
  public:
    // construtor
     ElementoGrafico();
     Ponto* PInicial();  // retorna o ponto inicial
     Color Cor();        // retorna a cor

     virtual void LeDoArquivo(char* arq);
     virtual void Desenha();

  protected:
     Ponto* pInicial;
```

```
    Color cor;
}
```

Esta classe base contém os dados comuns a todos os objetos:

- os dados cor e ponto inicial;
- os métodos de leitura e desenho.

A partir da classe base, pode-se definir outras três classes, representando linhas, retângulos e círculos, por exemplo:

```
class Linha: public ElementoGrafico
{
  public:
    // construtor
     Linha();
     Ponto* PFinal();  // retorna o ponto inicial

     virtual void LeDoArquivo(char* arq);
     virtual void Desenha();

  protected:
     Ponto* pFinal;
}


class Retangulo: public Linha
{
  public:
    // construtor
     Retangulo();

     virtual void LeDoArquivo(char* arq);
     virtual void Desenha();

  protected:
}


class Circulo: public ElementoGrafico
{
  public:
    // construtor
     Circulo();
     float Raio();  // retorna o raio

     virtual void LeDoArquivo(char* arq);
     virtual void Desenha();

  protected:
     float raio;
}
```

Em cada classe será implementado um método LeDoArquivo() e um método Desenha() utilizando seus dados particulares. Uma forma de utilização seria definir no programa principal um vetor para cada tipo de objeto. Ao invés disto, pode-se dar uma forma polimórfica, uma vez que os métodos públicos a serem acessados são os mesmos.

```
class MyApp
```

```
{
  public:
    Executa();

  protected:
    // Métodos
    void LeArquivo(char* arq);
    void Desenha();

    // Dados
    Array<Ponto> pontos;
    Array<ElementoGrafico> elems;
};
```

Supondo um método para leitura no qual cada registro possui um identificador de classe e um conjunto de dados formatado, a decisão do tipo de dado a ser incluído no vetor seria feita durante a leitura do arquivo:

```
void MyApp::LeArquivo(char* arq)
{
  ifstream arquivo;
  arquivo.open(arq);
  // lê os pontos
    ...
  // lê os objetos
  while (!eof()) {
    int id;
    arquivo << id;
    ElementoGrafico* elem;
    switch (id) {
      case LINHA:
        elem = new Linha();
      case RETANGULO:
        elem = new Retangulo();
      case CIRCULO:
        elem = new Circulo();
    }
    elem->LeDoArquivo(arquivo);
    elems->Add(elem);
  }
}
```

Embora a decisão do tipo de elemento seja feita ao nível de aplicativo, o polimorfismo permite que cada tipo de elemento leia um conjunto diferente de dados do arquivo, sem que isto tenha que ser gerenciado externamente. O mesmo comportamento reflete-se no método Desenha():

```
void MyApp::Desenha()
{
  int nElems = elems->GetItemsInContainer();
  for (int i=1; i ≤ nElems; i++)
    (*elems)[i]->Desenha();
}
```

Uma das vantagens deste procedimento é a facilidade de incluir novos objetos a partir da classe base. Neste caso, seria suficiente que todos implementassem os métodos virtuais LeDoArquivo() e Desenha() e a única modificação no código principal fica no grupo *switch*. Como comentário, outros esquemas de gravação prescindem deste tipo de modificação. Por exemplo, as *streams* do C++ permitem gravar classes que podem restaurar a si próprias, sem a necessidade de distinção por tipo.

4.7 - MODELO DE OBJETOS PARA UM PÓRTICO PLANO

Utilizando as definições já apresentadas, pode-se definir um modelo de classes e objetos com a finalidade de resolver o problema em questão, que é o de resolver um pórtico plano. Refere-se aqui a um esquema genérico, apenas para compreensão da metodologia empregada. Listagens completas podem ser encontradas em anexo.

4.7.1 Representação gráfica

A representação gráfica usual para uma classe contém os seguintes elementos:

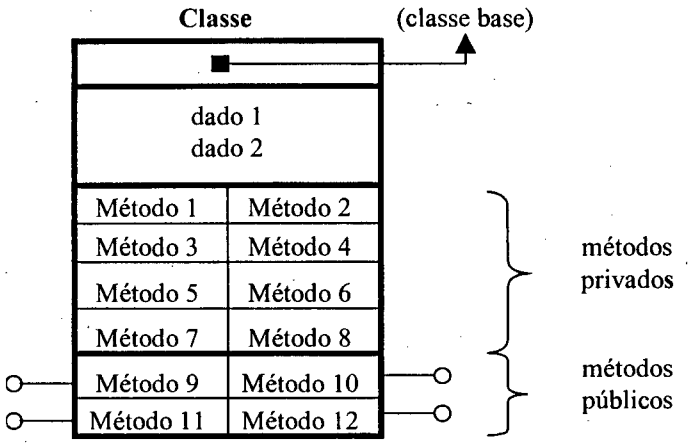


Figura 4.3 – Representação gráfica de uma classe

Entre os métodos públicos, pode-se omitir, por simplicidade, aqueles destinados para o acesso interno aos dados. Usualmente, para cada dado "item" existem dois métodos públicos: um para acessar seu valor e outro para alterá-lo.

```
class Dummy
{
  public:
     float Item() { return item; }              // acesso
     void Item(float _item) { item = _item; }   // atribuição

  protected:
```

```
        float item;
};
```

Isto nem sempre é verdade. Conforme já exposto, as técnicas de ocultamento de dados permitem restringir este tipo de acesso. Certos dados podem ser inicializados e utilizados apenas internamente, sem possuir qualquer forma de acesso externo.

4.7.2 Classe No

A classe No representa os nós da estrutura. Armazenam-se nela os dados referentes exclusivamente ao nó, que são:

- coordenadas (x, y);
- cargas aplicadas (fx, fy, mz);
- restrições de apoio (rx, ry, rz);
- reações de apoio após o cálculo (fx, fy, fz);
- deslocamentos após o cálculo (dx, dy, rz).

Além dos métodos de acesso aos próprios dados, podem-se definir métodos para leitura e escrita em arquivo.

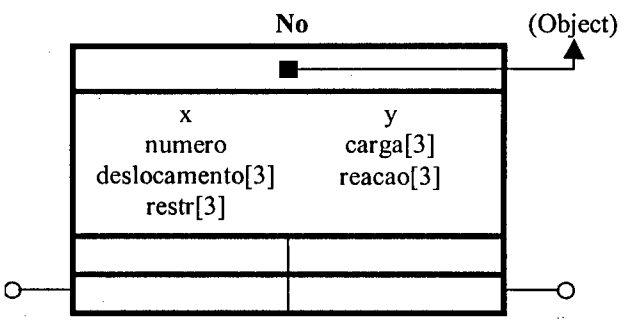


Figura 4.4 – Classe No

4.7.3 Classe Barra

A classe Barra representa uma barra ligando dois nós da estrutura. A ela referem-se os seguintes dados:

- nó inicial e nó final (ponteiros);
- propriedades geométricas (E, I, A);
- carga distribuída (Q);
- esforços internos após o cálculo (ci, vi, mi, cf, vf, mf).

Na classe Barra, encapsulam-se os métodos que definem seu comportamento. Além dos métodos de acesso aos dados e os de I/O, destacam-se:

- obtenção do vetor de forças local;
- obtenção da matriz de rigidez local;
- obtenção dos esforços internos com base nos deslocamentos nodais;
- conversão do sistema local para o global e vice-versa.

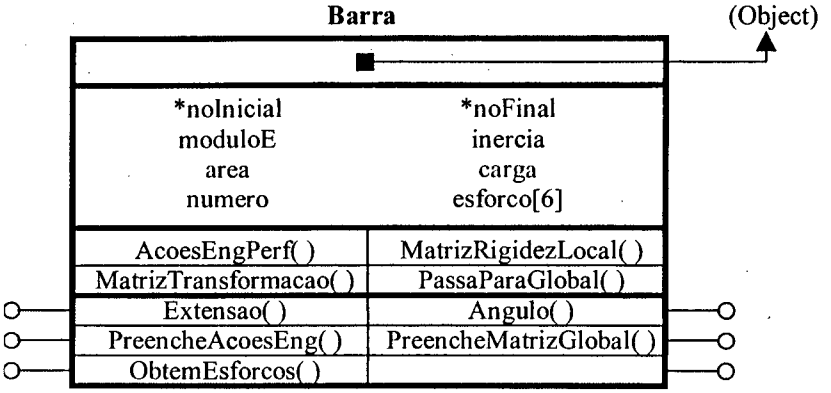


Figura 4.5 – Classe Barra

A formulação da matriz de rigidez referida ao sistema local, conforme Eq. (2-2), consiste em um exemplo perfeito da Orientação a Objetos. Esta matriz baseia-se na formulação assumida para o elemento (pequenos deslocamentos, despreza-se cortante, etc.) e utiliza apenas dados referentes ao mesmo (propriedades geométricas e extensão). O mesmo se pode dizer da montagem do vetor de forças e da conversão entre sistemas de coordenadas. O encapsulamento destes procedimentos em uma classe permite a sua fácil modificação. Os conceitos de herança e polimorfismo já abordados permitem implementar formulações mais complexas com menor esforço.

4.7.4 Classe MatrizBanda

A solução do sistema de equações lineares, um dos pontos centrais na resolução de estruturas reticuladas, pode ser feita de diversas maneiras, tendo este trabalho utilizado o Método de Cholesky. O fato de existirem vários outros tipos de procedimentos (por exemplo, o método de Gauss e o método Skyline) indica a necessidade de representar este procedimento também por uma classe.

A classe MatrizBanda possui como dados:

- a largura da banda;

- a matriz de coeficientes (no caso, a matriz de rigidez [K]);
- o vetor de valores independentes (no caso, o vetor de forças {F});
- o vetor de resultados (no caso, o vetor de deslocamentos {δ}).

Os métodos essenciais referem-se a:

- fatoração da matriz;
- obtenção dos resultados;
- armazenamento em disco;
- verificação da precisão numérica.

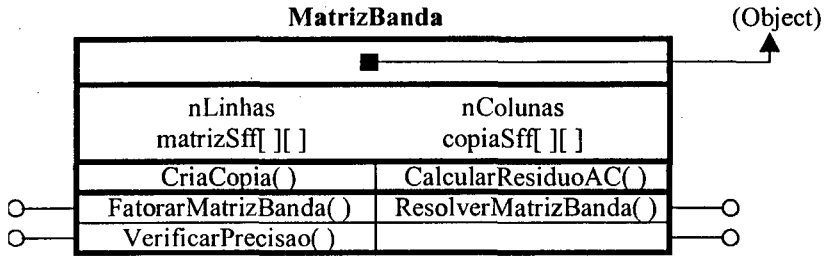


Figura 4.6 – Classe MatrizBanda

Esta classe, por sua generalidade, poderia também ser utilizada sem modificações na solução de outros tipos de problemas, como, por exemplo, treliças, pórticos espaciais ou mesmo Elementos Finitos.

4.7.5 Classe Pórtico

Englobando as classes já definidas, define-se a classe que representa o pórtico plano. Nesta, estarão presentes:

- vetor de nós da estrutura;
- vetor de barras da estrutura;
- sistema linear resultante (MatrizBanda);
- vetores globais de carregamentos e deslocamentos;
- vetores para mapear os graus de liberdade restringidos.

Os diversos métodos referentes à solução do problema constituem-se basicamente em apenas três métodos públicos:

- LePorticoDeArquivo: para a leitura dos dados do arquivo;
- Calcula: efetua todo o processamento;
- GravaResultadosArquivo: cria arquivo de resultados.

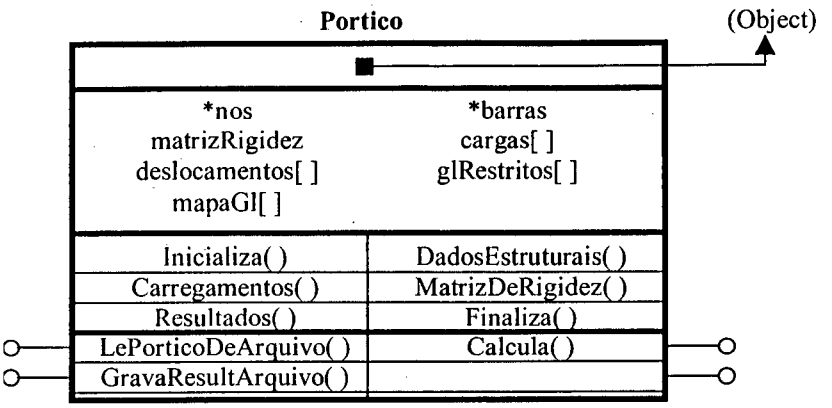


Figura 4.7 – Classe Portico

4.8 - COMPARAÇÃO COM A PROGRAMAÇÃO ESTRUTURADA

Uma comparação completa com o mesmo programa implementado através da Programação Estruturada seria tediosa e repetitiva. Pode-se definir, contudo, algumas diferenças fundamentais em termos de filosofia de trabalho:

Programação Estruturada	Programação Orientada a Objetos
- Trabalha com uma sequência linear de chamada de sub-rotinas.	- Possui hierarquia de classes, onde cada classe desempenha determinado tipo de tarefa e uma classe maior, que engloba as outras, possui um método "principal"
- Cada sub-rotina do sistema trabalha com vetores globais de dados	- Classes encapsulam dados e métodos
- Variações no programa necessitam duplicar e modificar o original	- Mudanças podem ser feitas por herança de classes, sem afetar o código original

Tabela 4.2 – Comparação entre os estilos de programação

Como comparação, pode-se tomar o trecho principal do mesmo programa para solução de pórticos planos, elaborado em linguagem BASIC, em termos de Programação Estruturada, utilizado por ZERMIANI (1998). Foram efetuadas algumas pequenas simplificações.

```
'DEFINICAO DE VARIAVEIS
DIM nos(50, 2), incid(50, 2), l(50), ang(50), matglob(50, 6,
6), matest(100, 100), mattramp(50, 6, 6)
DIM carreg(50, 2), prop(50, 3), matloc(50, 6, 6), mattran(50,
6, 6), restr(100, 7), esfor(100, 6)
DIM a(100, 100), b(100), c(100), mat(50, 6, 6), Fo(6),
resul(100, 6), F0(100), F(100), x(100)
'INICIO DO PROGRAMA
INPUT "Nome do arquivo: ", arq$
```

```
OPEN "i", #1, arq$ + ".dad"
OPEN "o", #2, arq$ + ".sad"
CALL lenos(nnos, nos())
CALL leprop(nbarras, prop())
CALL lecargas(nbarras, carreg())
CALL leincidencia(nbarras, incid())
CALL comprimento(nbarras, incid(), nos(), l(), ang())
CALL matrizlocal(nbarras, prop(), l(), matloc())
CALL matriztran(nbarras, ang(), mattramp(), mattran(),
matloc(), matglob())
CALL matrizestrut(nbarras, incid(), matglob(), matest())
CALL esforcono(F())
CALL termoindep(nbarras, nnos, carreg(), l(), incid(), ang(),
mattramp(), resul(), F(), b())
CALL lerestr(nr, restr())
CALL testaglr(nnos, ngl, restr(), matest(), b())
CALL resolmat(ngl, nbarras, incid(), matest(), b(), x())
CALL calesforcos(nbarras, resul(), matglob(), incid(), x(),
mattran(), esfor())
CALL impressao(nnos, ngl, nos(), nbarras, prop(), carreg(),
incid(), l(), ang(), matloc(), x(), esfor())
CLOSE #1
CLOSE #2
END
```

Pode-se notar, em uma análise superficial:

- A sequência de trabalho do programa é simples e bastante linear, podendo ser considerada de fácil compreensão (deve-se ter em mente, todavia, que este é um programa muito simples);
- Todos os procedimentos e variáveis são trabalhados no nível do aplicativo, dificultando seu aproveitamento como parte de outro módulo maior;
- O uso de vetores para representar os dados pode ser considerado muito pouco claro (por exemplo, a matriz Prop() armazena os valores de E, I e A para cada barra, um em cada coluna, acessando-os por Prop(i,1), Prop(i,2) e Prop(i,3), respectivamente).

Na OOP, não se trabalha com um procedimento principal. O processamento da estrutura está encapsulado na classe Portico e pode ser acessado por qualquer aplicativo. Por exemplo, pode-se incluí-lo como um item de menu de um programa de edição de textos. Depois, pode-se incluir um procedimento como a seguir:

```
void TPFrameApp::CmCalculaPortico()
{
  const char* nomeArquivo = ArquivoDados();
  if (nomeArquivo) {
    Portico* port = new Portico();
    port->LePorticoDeArquivo(nomeArquivo);
    port->Calcula();
    string resultFileName = MudaExtensao(nomeArquivo, ".sad");
    port->GravaResultadosEmArquivo(resultFileName.c_str());
    delete port;
```

```
    }
  } // CmCalculaPortico
```

Nota-se que, externamente, apenas os métodos públicos da classe Portico são acessados. Pode-se constituir facilmente um módulo "solver" contendo todo o código compilado das classes de resolução (Portico, Barra, No, MatrizBanda e classes básicas) e ser utilizado por uma aplicação qualquer.

Para a OOP, a sequência de cálculo pode ser considerada também linear, mas com diversos níveis de profundidade. Chamado o método Calcula( ) do objeto *port*, o código a seguir será executado (listagem simplificada):

```
void Portico::Calcula()
{
  DadosEstruturais();
  CriarMatrizSFF();
  MatrizDeRigidez();
  matEst->FatorarMatrizBanda();
  Carregamentos();
  matEst->ResolverMatrizBanda(ac,df);
  matEst->VerificarPrecisao(ac,df);
  Resultados();
} // Calcula
```

Este método assemelha-se ao procedimento principal do código estruturado, mas com algumas diferenças:

- o uso de dados da classe Portico, considerados globais (ac, df);
- a mudança de escopo de parte do procedimento, ficando a parte de solução do sistema linear em outra classe, a MatrizBanda;
- não necessidade de uso de grande número de parâmetros globais, como na implementação estruturada.

Como comentário, pode-se eliminar a passagem de parâmetros também na Programação Estruturada definindo os vetores de dados como globais. O problema é que estes ficariam globais a toda a aplicação e acessíveis a todas as sub-rotinas, em comparação com a OOP, onde os dados são acessados globalmente apenas no escopo onde estão inseridos.

Em seguida, pode-se analisar um dos métodos privados chamados por Calcula( ), destinado à montagem do vetor de carregamentos:

```
void Portico::Carregamentos()
{
  Vetor *aj = new Vetor(Ngl());
  Vetor *ae = new Vetor(Ngl());

  // preenche vetor AJ
  int numero;
  for (int i = 1; i <= NNos(); i++) {
```

```
        numero = (*nos)[i]->Numero();
        for (TDeslocamento j = DX; j <= RZ; j++)
         aj[(numero - 1) * Ndj() + j] = (*nos)[i]->Carga(j);
      }

      // preenche vetor AE
      for (int i = 1; i <= NBarras(); i++) {
         Barra* barra = (*barras)[i];
         barra->PreencheAcoesEngPerf(ae);
      }

      // Carregamentos nos nós combinados
      int jr;
      for (int j = 1; j <= Ngl(); j++) {
        jr = id[j];
        ac[jr] = aj[j] + ae[j];
      }

      delete ae;
      delete aj;
    } // Carregamentos
```

Pode-se notar:

- o uso do vetor global de carregamentos (ac) e do vetor que mapeia os graus de liberdade após a vinculação (id);
- a montagem das cargas nodais equivalentes devidas às cargas distribuídas nas barras através de um método da classe Barra (PreencheAcoesEngPerf(ae)).

Aqui nota-se outra grande vantagem da OOP, na medida em que os detalhes relevantes às propriedades das barras e à formulação adotada estão encapsulados na classe Barra. Caso se desejasse modificar a formulação da barra, incluindo, por exemplo, carregamentos triangulares, todas as modificações seriam feitas apenas na classe Barra. Este e os outros métodos da classe Portico ficariam inalterados.

As comparações feitas aqui são superficiais, mas seguem as tendências atuais de programação. Justifica-se sua abordagem neste ponto do trabalho por consistir em uma contribuição à forma como este tipo de trabalho vem sido apresentado pelo meio acadêmico. Em anexo, serão encontradas listagens completas referentes a todas as classes relevantes no programa.

4.9 – IMPLEMENTAÇÃO DA NÃO LINEARIDADE

As vantagens da OOP podem ficar mais claras ao se analisar a inserção da não linearidade geométrica aos procedimentos já implementados. Isto porque as vantagens aparecem mais claramente exatamente quando se deseja estender ou reaproveitar código já existente.

Parte-se da seguinte situação:

- Tem-se um programa, escrito sob os parâmetros da OOP, com o objetivo de calcular os deslocamentos e esforços internos nas barras de um pórtico plano, utilizando apenas a análise elástica linear, exatamente como descrito nos itens 2.3 e 4.7;
- Deseja-se estender este programa, fazendo o mínimo de modificações, para que neste possam ser incluídos os efeitos não lineares geométricos;
- Pretende-se ter, ao final, quatro programas funcionando completamente, sendo um para análise elástica linear e um para cada processo não linear discutido até aqui (P-Delta, K_G e Funções de Estabilidade), com o mínimo de duplicação de código.

4.9.1 Classes a alterar

A fim de entender melhor o problema, pode-se listar a definição do modelo de objetos para um pórtico plano definido no item 4.7. Basicamente, foram definidas as seguintes classes:

Classe	Dados	Comportamento
No	- coordenadas; - deslocamentos; - esforços nodais.	- apenas armazenamento.
Barra	- propriedades geométricas; - esforços internos.	- monta sua matriz de rigidez; - obtém seus vetor de carregamentos nodais equivalentes; - obtém seus esforços internos a partir dos deslocamentos nodais.
MatrizBanda	- matriz de rigidez da estrutura.	- resolve o sistema de equações lineares.
Portico	- vetor de nós; - vetor de barras; - sistema linear.	- gerencia todos os dados; - gerencia leitura, criação e gravação; - calcula a estrutura em si.

Tabela 4.3 – Classes de um pórtico plano

Pode-se observar, com base na Tabela 4.3 e no que foi discutido no item 3.3.6:

- As classes No e MatrizBanda mantém-se inalteradas, visto que a mudança no processo de cálculo não altera em nada a representação dos dados de uma estrutura e que a parte de solução do sistema linear não é afetada;
- A classe Portico deve necessariamente ser alterada, visto que o processo de cálculo, que antes era direto, passa a ser iterativo;
- A classe Barra também deve ser alterada, visto que esta tem a responsabilidade de descrever seu próprio comportamento, justamente o que deve ser alterado (mudança na matriz de rigidez, por exemplo);
- Mesmo as duas classes alteradas o serão apenas parcialmente, visto que grande parte do processo se mantém inalterada.

4.9.2 Alteração no processo de cálculo

Em primeiro lugar, deve-se lembrar que a consideração da não linearidade geométrica envolve um procedimento iterativo, onde realizam-se diversas análises lineares até que se obtenha a convergência desejada. Abaixo reproduz-se a Figura 3.21:

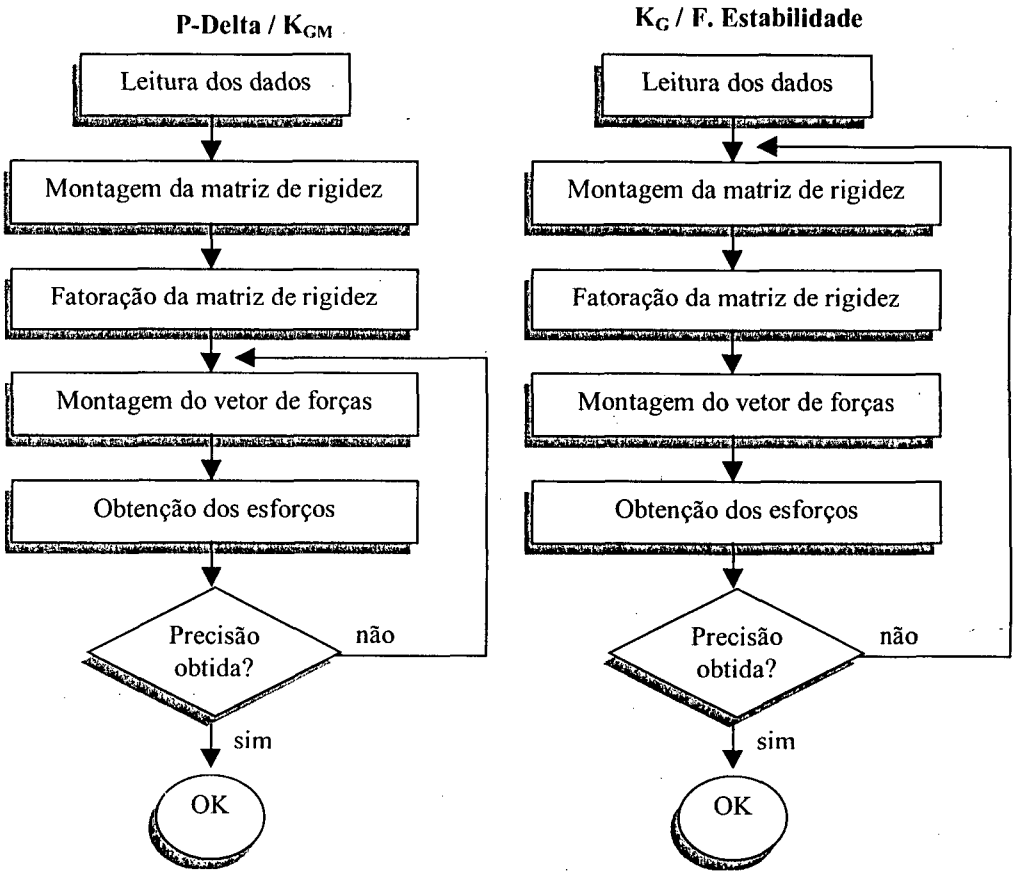


Figura 3.21 – Comparação entre os fluxogramas do Processo P-Delta e do K_G

A parte central do fluxograma corresponde ao método *Calcula( )* da classe Portico. Basicamente, apresenta-se como se segue:

```
bool Portico::Calcula()
{
 DadosEstruturais();
 CriarMatrizSFF();
 MatrizDeRigidez();
 matEst->FatorarMatrizBanda();
 Carregamentos();
 matEst->ResolverMatrizBanda(ac,df);
 matEst->VerificarPrecisao(ac,df));
 Resultados();
 DestruirMatrizes();
} // Calcula
```

O objetivo de cada uma delas é o seguinte:

- *DadosEstruturais( )*: mapeia os deslocamentos restringidos e calcula a largura de banda;
- *CriarMatrizSff( )*: aloca memória para a matriz de rigidez da estrutura;
- *MatrizDeRigidez( )*: preenche a matriz de rigidez global da estrutura;
- *FatorarMatrizBanda( )*: inverte a matriz de rigidez global;
- *Carregamentos( )*: monta o vetor de carregamentos nodais (ac);
- *ResolverMatrizBanda( )*: obtém os deslocamentos (df) com base nos carregamentos nodais (ac) e na matriz já invertida;
- *VerificarPrecisao( )*: verifica a precisão numérica do resultado obtido;
- *Resultados( )*: obtém os esforços internos nas barras;
- *DestruirMatrizes( )*: finaliza o cálculo.

Processos P-Delta e K_{GM}

Estes dois processos, o Processo P-Delta e o Processo da Matriz de Rigidez Geométrica Modificado, em termos de solução do problema não linear, podem ser tratados em conjunto, visto que possuem as mesmas premissas básicas. Ambos incluem a não linearidade geométrica na correção do vetor de deslocamentos a cada iteração.

Para a implementação destes processos, seria necessário:

- Alterar a montagem do vetor de carregamentos nodais (método *Carregamentos( )*), incluindo as cargas fictícias;
- Após obter os deslocamentos nodais, verificar a alteração em relação à iteração anterior;

- Caso necessário, obter os esforços internos (método *Resultados( )*) e voltar à montagem do vetor de carregamentos.

Deve-se notar que toda a montagem da matriz de rigidez (até o método *FatorarMatrizBanda( )*) mantém-se inalterada. O mesmo método *Calcula( )* poderia se apresentar como seguinte:

```
bool PorticoCargas::Calcula()
{
 DadosEstruturais();
 CriarMatrizSFF();
 MatrizDeRigidez();
 matEst->FatorarMatrizBanda();

 int nIter = 1;
 bool convergiu = false;
 while ((!convergiu) && (nIter < MAX_ITER)) {
   Carregamentos();
   matEst->ResolverMatrizBanda(ac,df);
   matEst->VerificarPrecisao(ac,df));
   Resultados();
   convergiu = ProcessoConvergiu();
   nIter++;
 }
 if (!convergiu)
   MessageBox("Processo não convergiu.");

  DestruirMatrizes();
} // Calcula
```

Existem três modificações:

- O método *Calcula( )*, como pode ser visto;
- O método *Carregamentos( )*, que deve incluir as cargas fictícias oriundas do Processo P-Delta ou do K_{GM};
- A inclusão de um novo método, *ProcessoConvergiu( )*, com o objetivo de verificar se a variação nos deslocamentos é significativa ou não.

Todos os demais métodos se mantém inalterados.

Processos K_G e das Funções de Estabilidade

Os outros dois processos analisados, a saber, o Processo da Matriz de Rigidez Geométrica e o Processo das Funções de Estabilidade, também podem ser tratados em conjunto. Ambos incluem a não linearidade geométrica na formulação da matriz de rigidez da barra, variando unicamente os próprios coeficientes de rigidez.

O mesmo método *Calcula( )* poderia se apresentar como se segue para ambos os processos:

```
bool PorticoRigidez::Calcula()
```

```
{
 DadosEstruturais();
 CriarMatrizSFF();

 int nIter = 1;
 bool convergiu = false;
 while ((!convergiu) && (nIter < MAX_ITER)) {
   MatrizDeRigidez();
   matEst->FatorarMatrizBanda();
   Carregamentos();
   matEst->ResolverMatrizBanda(ac,df);
   matEst->VerificarPrecisao(ac,df));
   Resultados();
   convergiu = ProcessoConvergiu();
   nIter++;
 }
 if (!convergiu)
   MessageBox("Processo não convergiu.");

  DestruirMatrizes();
} // Calcula
```

Existem três modificações:

- O método *Calcula( )*, como pode ser visto;
- O método *MatrizDeRigidez( )*, que deve refletir a nova formulação da matriz de rigidez da barra;
- A inclusão de um novo método, *ProcessoConvergiu( )*, idêntico ao já destinado ao Processo P-Delta.

Todos os demais métodos se mantém inalterados.

4.9.3 Alteração na formulação da barra

Pôde ser visto no item anterior que são necessárias três modificações adicionais para implementar os processos não lineares estudados:

- Alteração do método que monta o vetor de carregamentos (*Carregamentos( )*) no caso do Processo P-Delta e do K_{GM};
- Alteração do método que monta a matriz de rigidez global (*MatrizDeRigidez( )*) nos outros dois casos;
- Criação de um novo método para verificar a convergência do processo (*ProcessoConvergiu( )*).

Vetor de carregamentos

A montagem do vetor de carregamentos, segundo a filosofia da OOP, é de responsabilidade da classe Portico, visto que este possui o vetor de barras e o vetor de carregamentos nodais. Todavia, independe ao Portico os detalhes de como este vetor é preenchido, uma vez que isto irá variar de acordo com o carregamento presente na barra (carga total ou parcialmente distribuída, cargas concentradas, entre outros).

Desta forma, o Pórtico apenas gerencia o processo, mas é responsabilidade da classe Barra o efetivo preenchimento do vetor de carregamentos. O método *Carregamentos( )* apresenta-se basicamente como seguinte:

```
bool Portico::Carregamentos()
{
  Vetor *aj = new Vetor(Ngl());
  Vetor *ae = new Vetor(Ngl());

// Carregamentos nodais
  int numero;
  for (int i = 1; i <= NNos(); i++) {
    numero = (*nos)[i]->Numero();
    for (TDeslocamento j = DX; j <= RZ; j++)
       aj[(numero - 1) * Ndj() + j] = (*nos)[i]->Carga(j);
  }

// Carregamentos distribuídos nas barras
  for (int i = 1; i <= NBarras(); i++) {
     Barra* barra = (*barras)[i];
     barra->PreencheAcoesEngPerf(ae);
  }

  // Carregamentos totais
  int jr;
  for (int j = 1; j <= Ngl(); j++) {
    jr = id[j];
    ac[jr] = aj[j] + ae[j];
  }

  delete ae;
  delete aj;
} // Carregamentos
```

Pode-se notar que todos os detalhes referentes à montagem do vetor de carregamentos estão encapsulados no método *PreencheAcoesEngPerf( )* da classe Barra. Na verdade, não é necessário fazer qualquer modificação no método *Carregamentos( )* da classe Portico.

Passando à implementação da barra, o método *PreencheAcoesEngPerf( )* apresenta-se da seguinte forma:

```
void Barra::PreencheAcoesEngPerf(Vetor *ae)
{
```

```
    MatrizBarra *matTrans = new MatrizBarra();
    MatrizBarra *matTransposta = new MatrizBarra();
    MatrizTransformacao(matTrans,matTransposta);

    for (int lin=1; lin<=6; lin++) {
      for (TEsforco col=ESF_AXIAL_I; col<=ESF_FLETOR_F; col++)
        ae[GlGlobal(lin)]    +=    (*matTransposta)[lin][col]    *
AcoesEngPerf(col);
    }

    delete matTrans;
    delete matTransposta;
  } // PreencheAcosEngPerf
```

Mesmo neste nível mais interno, ainda pode-se notar a divisão de tarefas. Neste método, existe a chamada para outros métodos da classe Barra:

- *MatrizTransformação( )*: cria as matrizes de transformação de coordenadas do sistema local para o global e vice-versa;
- *GlGlobal( )*: mapeia os deslocamentos globais com relação ao número de seus nós inicial e final;
- *AcoesEngPerf( )*: retorna os carregamentos nodais no sistema local da barra.

Com isto, pode-se notar que este método se mantém inalterado, bem como as tarefas de mapear as coordenadas e converter os esforços para o sistema global. Na verdade, toda a parte variável do processo esta encapsulada no método *AcoesEngPerf( )*:

```
float Barra::AcoesEngPerf(TEsforco indice)
{
  float valor = 0;
  switch (indice) {
    case ESF_AXIAL_I     : valor = 0;
                           break;
    case ESF_CORTANTE_I  : valor = carga * Extensao() /2;
                           break;
    case ESF_FLETOR_I    : valor = carga * pow(Extensao(),2) /12;
                           break;
    case ESF_AXIAL_F     : valor = 0;
                           break;
    case ESF_CORTANTE_F  : valor = carga * Extensao() /2;
                           break;
    case ESF_FLETOR_F    : valor = -carga * pow(Extensao(),2) /12;
                           break;
  }
  return valor;
} // AcoesEngPerf
```

Este método retorna os esforços nodais equivalentes a um carregamento uniformemente distribuído em toda a extensão da barra. Caso se desejasse incluir uma carga concentrada, por exemplo, apenas este método deveria ser alterado. Isto ilustra a potencialidade de reaproveitamento do código apresentada pela OOP.

No caso de se incluir o Processo P-Delta na formulação da barra, este método seria alterado da seguinte forma:

```
float BarraPDelta::AcoesEngPerf(TEsforco indice)
{
  float valor = Barra::AcoesEngPerf(indice);
  float N = -Esforco(ESF_AXIAL_I);
  if (fabs(N) > 0.1) {
    float a = Deslocamento(DY_F) - Deslocamento(DY_I);
    float h = N * a / Extensao();
    switch (indice) {
      case ESF_CORTANTE_I    : valor += h;
                               break;
      case ESF_CORTANTE_F    : valor -= h;
                               break;
    }
  }
  return valor;
} // AcoesEngPerf
```

Herança e polimorfismo

O exemplo acima ilustra uma das propriedades da OOP: a herança. Deve-se notar que a utilização de uma classe derivada da classe Barra permite modificar apenas os métodos desejados. Outro ponto a notar é que uma classe mantém toda a funcionalidade de suas classes ancestrais. Neste caso, a montagem do vetor de carregamentos inicia exatamente executando o código da classe Barra para, em seguida, apenas acrescentar os esforços referentes ao efeito P-Delta. Com isto:

- No caso de uma alteração do programa para a inclusão de cargas concentradas no interior das barras, isto poderia ser feito diretamente na classe Barra;
- A classe BarraPDelta, derivada de Barra, não necessitaria de nenhuma modificação;
- Não existe duplicação de código.

Outra propriedade da OOP, o *polimorfismo* (vide item 4.6), pode também ser observada aqui. Externamente, o acesso à barra é feito através do método *PreencheAcoesEngPerf( )*. Este utiliza internamente o método *AcoesEngPerf( )* para montar o vetor no sistema local. Pode-se derivar a classe BarraPDelta sobrescrevendo apenas o método *AcoesEngPerf( )*, uma vez que a característica polimórfica da OOP consegue decidir, na execução do método *PreencheAcoesEngPerf( )*, que o método *AcoesEngPerf( )* deve ser executado a partir da classe derivada e não da classe base. Com isto, evita-se duplicar o resto do procedimento.

Para implementar o Processo da Matriz de Rigidez Geométrica Modificado, basta redefinir a formulação da barra, alterando o mesmo método para:

```
float BarraPDelta::AcoesEngPerf(TEsforco indice)
{
  MatrizBarra *mataux = new MatrizBarra();
  MatrizCorrecaoRigidez(mataux);
  LadoSimetrico(mataux);

  float valor = Barra::AcoesEngPerf(indice);
  for (TDeslocamentoBarra i=DX_I; i<= RZ_F; i++) {
    valor += (*mataux)[indice][i] * Deslocamento(i);
  }
  return valor;
} // AcoesEngPerf

void BarraKgMod::MatrizCorrecaoRigidez(MatrizBarra* matLocal)
{
  float l = Extensao();
  float N = Esforco(ESF_AXIAL_I);

  (*matLocal)[2][2] += 6 * N / (5 * l);
  (*matLocal)[2][3] += N / 10;
  (*matLocal)[2][5] += -6 * N / (5 * l);
  (*matLocal)[2][6] += N / 10;
  (*matLocal)[3][3] += 2 * N * l / 15;
  (*matLocal)[3][5] += -N / 10;
  (*matLocal)[3][6] += -N * l / 30;
  (*matLocal)[5][5] += 6 * N / (5 * l);
  (*matLocal)[5][6] += -N / 10;
  (*matLocal)[6][6] += 2 * N * l / 15;
} // MatrizCorrecaoRigidez
```

Matriz de rigidez

A montagem da matriz de rigidez da estrutura, da mesma forma como o vetor de carregamentos, é de responsabilidade da classe Portico, visto que este possui o vetor de barras e matriz de rigidez global. Todavia, interessa ao Portico os detalhes de como a matriz é preenchida, fazendo a correta superposição entre todas as barras e gerenciando o armazenamento conforme a banda, mas não os valores dos coeficientes em si, que dependerão da formulação da própria barra.

Desta forma, o Pórtico apenas gerencia o processo, mas é responsabilidade da classe Barra o efetivo preenchimento da matriz de rigidez. O método *MatrizDeRigidez()* apresenta-se basicamente como seguinte:

```
bool Portico::MatrizDeRigidez()
{
  Matriz *sff = matEst->Sff();
  for (int i = 1; i <= NBarras(); i++) {
     MatrizBarra* sms = new MatrizBarra();
     Barra* barra = (*barras)[i];
```

```
            barra->MatrizDeRigidezGlobal(sms);

            for (int j = 1; j <= NglBarra(); j++) {
             int i1 = barra->GlGlobal(j);
             if (!glRestrito[i1]) {
               for (int k = j; k <= NglBarra(); k++) {
                 int i2 = barra->GlGlobal(k);
                 if (!glRestrito[i2]) {
                   int ir = id[i1];
                   int ic = id[i2];
                   if (ir >= ic) {
                     int item = ir;
                     ir = ic;
                     ic = item;
                   }
                   ic = ic - ir + 1;
                   (*sff)[ir][ic] += (*sms)[j][k];
                 }
               }
             }
            }
            delete sms;
          }
        } // MatrizDeRigidez
```

Abstraindo-se os detalhes, o método percorre todas as barras do pórtico, obtém suas matrizes de rigidez já no sistema global (método *MatrizDeRigidezGlobal( )* da classe Barra) e as superpõe na matriz de rigidez global. Na verdade, não é necessário fazer qualquer modificação no método *MatrizDeRigidez( )* da classe Portico.

Passando à implementação da barra, o método *MatrizDeRigidezGlobal ( )* apresenta-se da seguinte forma:

```
void Barra::MatrizDeRigidezGlobal(MatrizBarra *mat, bool
rotacionar)
{
  MatrizRigidezLocal(mat);
  LadoSimetrico(mat);
  PassaParaGlobal(mat, rotacionar);
} // MatrizDeRigidezGlobal
```

Mesmo neste nível mais interno, ainda pode-se notar a divisão de tarefas. Neste método, existe a chamada para outros métodos da classe Barra:

- *MatrizRigidezLocal( )*: cria a matriz de rigidez da barra no sistema local (apenas a triangular superior);
- *LadoSimetrico( )*: copia os coeficientes para o lado simétrico da matriz;
- *PassaParaGlobal( )*: passa a matriz de rigidez definida para o sistema de coordenadas globais.

Com isto, pode-se notar que este método se mantém inalterado, bem como a tarefa de converter os coeficientes para o sistema global. Na verdade, toda a parte variável do processo está encapsulada no método *MatrizRigidezLocal( )*:

```
void Barra::MatrizRigidezLocal(MatrizBarra *matLocal)
{
  float i = Inercia();
  float e = ModuloE();
  float a = Area();
  float l = Extensao();
  (*matLocal)[1][1] = e * a / l;
  (*matLocal)[1][4] = -(*matLocal)[1][1];
  (*matLocal)[2][2] = 12 * e * i / pow(l,3);
  (*matLocal)[2][3] = 6 * e * i / pow(l,2);
  (*matLocal)[2][5] = -(*matLocal)[2][2];
  (*matLocal)[2][6] = (*matLocal)[2][3];
  (*matLocal)[3][3] = 4 * e * i / l;
  (*matLocal)[3][5] = -(*matLocal)[2][3];
  (*matLocal)[3][6] = 2 * e * i / l;
  (*matLocal)[4][4] = (*matLocal)[1][1];
  (*matLocal)[5][5] = (*matLocal)[2][2];
  (*matLocal)[5][6] = -(*matLocal)[2][3];
  (*matLocal)[6][6] = (*matLocal)[3][3];
} // MatrizRigidezLocal
```

Este método retorna a matriz de rigidez da barra referida ao seu sistema de coordenadas locais, desprezando-se a força normal e as deformações por cisalhamento, exatamente como formulado no item 2.3.2. A consideração da não linearidade geométrica, tanto pelo Processo K_G como pelo Processo das Funções de Estabilidade, seria feita neste ponto.

No caso de se incluir o Processo K_G na formulação da barra, este método seria alterado da seguinte forma:

```
void BarraKg::MatrizRigidezLocal(MatrizBarra* matLocal)
{
  Barra::MatrizRigidezLocal(matLocal);
  float l = Extensao();
  float N = -Esforco(ESF_AXIAL_I);

  (*matLocal)[2][2] += 6 * N / (5 * l);
  (*matLocal)[2][3] += N / 10;
  (*matLocal)[2][5] += -6 * N / (5 * l);
  (*matLocal)[2][6] += N / 10;
  (*matLocal)[3][3] += 2 * N * l / 15;
  (*matLocal)[3][5] += -N / 10;
  (*matLocal)[3][6] += -N * l / 30;
  (*matLocal)[5][5] += 6 * N / (5 * l);
  (*matLocal)[5][6] += -N / 10;
  (*matLocal)[6][6] += 2 * N * l / 15;
} // MatrizRigidezLocal
```

Já no caso das funções de estabilidade, teria-se algo como o seguinte[11]:

```
void    BarraFEstabilidade::MatrizRigidezLocal    (MatrizBarra*
matLocal)
{
  Barra::MatrizRigidezLocal(matLocal);
  float l = Extensao();
```

[11] Para a notação utilizada na implementação, vide WEAVER & GERE (1965).

```
    float N = Esforco(ESF_AXIAL_I);

    double k = sqrt(fabs(N)/(ModuloE()*Inercia()));
    double kL = k*l;
    double fiC = 2 - 2*cos(kL) - kL*sin(kL);
    double fiT = 2 - 2 * cosh(kL) + kL * sinh(kL);
    double s1=1, s2=1, s3=1, s4=1;

    if (N > 10) {                          //compressao
      s1 = (pow(kL,3) * sin(kL)) / (12 * fiC);
      s2 = (pow(kL,2) * (1 - cos(kL))) / (6 * fiC);
      s3 = (kL * (sin(kL) - kL * cos(kL))) / (4 * fiC);
      s4 = (kL * (kL - sin(kL))) / (2 * fiC);
    }
    else if (N < -10) {
      s1 = (pow(kL,3) * sinh(kL)) / (12 * fiT);
      s2 = (pow(kL,2) * (cosh(kL) - 1)) / (6 * fiT);
      s3 = (kL * (-sinh(kL) + kL * cosh(kL))) / (4 * fiT);
      s4 = (kL * (-kL + sinh(kL))) / (2 * fiT);
    }

    (*matLocal)[2][2] *= s1;
    (*matLocal)[2][3] *= s2;
    (*matLocal)[2][5] *= s1;
    (*matLocal)[2][6] *= s2;
    (*matLocal)[3][3] *= s3;
    (*matLocal)[3][5] *= s2;
    (*matLocal)[3][6] *= s4;
    (*matLocal)[5][5] *= s1;
    (*matLocal)[5][6] *= s2;
    (*matLocal)[6][6] *= s3;
  } // MatrizRigidezLocal
```

Todas as considerações já feitas sobre herança, polimorfismo e reaproveitamento de código podem ser notadas também neste caso.

4.9.4 Implementação

As considerações feitas nos dois itens anteriores permitem a definição de uma implementação OOP para os processos não lineares geométricos a partir de um programa de resolução de pórticos planos.

Em primeiro lugar, é necessário derivar a classe Portico a fim de refletir a mudança no algoritmo de cálculo. Uma vez que existem pontos comuns, é interessante fazer uso se uma hierarquia, lembrando que:

- A verificação de convergência dos deslocamentos (método *ProcessoConvergiu( )*) é a mesma para todos os processos;
- O método *Calcula( )* é um para os processos K_G e das funções de estabilidade, e outro para os processos P-Delta e K_{GM};
- Os demais métodos não se alteram.

Em segundo lugar, é necessário derivar a classe Barra a fim de refletir a diferença em cada formulação. Cada tipo de processo considerado utilizará uma classe Barra diferente. A fim de generalizar o processo, é conveniente simplesmente decidir o tipo de barra no momento da criação do pórtico:

```
void Portico::LeIncidencias(ifstream *arquivo,int nBarras)
{
  char coment[80];
  arquivo->getline(coment,80);
  WORD noI, noF;
  for(WORD i = 1; i <= nBarras; i++) {
    (*arquivo) >> noI;
    (*arquivo) >> noF;
    arquivo->getline(coment,80);
    Barra* barra = CriaBarra();
    barra->Numero(i);
    barra->NoInicial((*((*nos)[noI])));
    barra->NoFinal((*((*nos)[noF])));
    barras->Add(barra);
  }
}
```

O método *LeIncidencias()*, chamado a partir do método *LePorticoDeArquivo()*, cria as barras da estrutura de acordo com o arquivo de dados. Resolve-se o problema da seguinte forma:

- Os dados do arquivo são os mesmos para todos os processos (não é necessário informar o processo de cálculo no arquivo);
- Existe um método *CriaBarra()* que retorna um ponteiro para uma nova barra alocada;
- Esta nova barra é referenciada através de um ponteiro para a classe base Barra;
- Cada tipo de pórtico pode retornar um ponteiro para um tipo alocado de barra diferente;

Isto é possível porque qualquer classe pode ser referenciada por um ponteiro para uma de suas classes base, desde que não seja necessário chamar explicitamente um método que não exista nesta classe base. Caso seja utilizado um método sobrescrito (como foi descrito no item 4.9.3), a decisão será feita em tempo de execução sem problema algum. A isto se chama polimorfismo.

Desta forma, apenas um método *CriaBarra()* deve ser implementado nas classes Portico derivadas:

```
Barra* Portico::CriaBarra()
{
  return new Barra();
} // CriaBarra
```

```
Barra* PorticoPDelta::CriaBarra()
{
  return new BarraPDelta();
} // CriaBarra

Barra* PorticoKg::CriaBarra()
{
  return new BarraKg();
} // CriaBarra

Barra* PorticoKgMod::CriaBarra()
{
  return new BarraKgMod();
} // CriaBarra

Barra* PorticoFEstabilidade::CriaBarra()
{
  return new BarraFEstabilidade ();
} // CriaBarra
```

Derivação da classe Portico

Acrescentando-se também o método *CriaBarra()* do item anterior, poderia ser montada uma estrutura de classes como a seguinte:

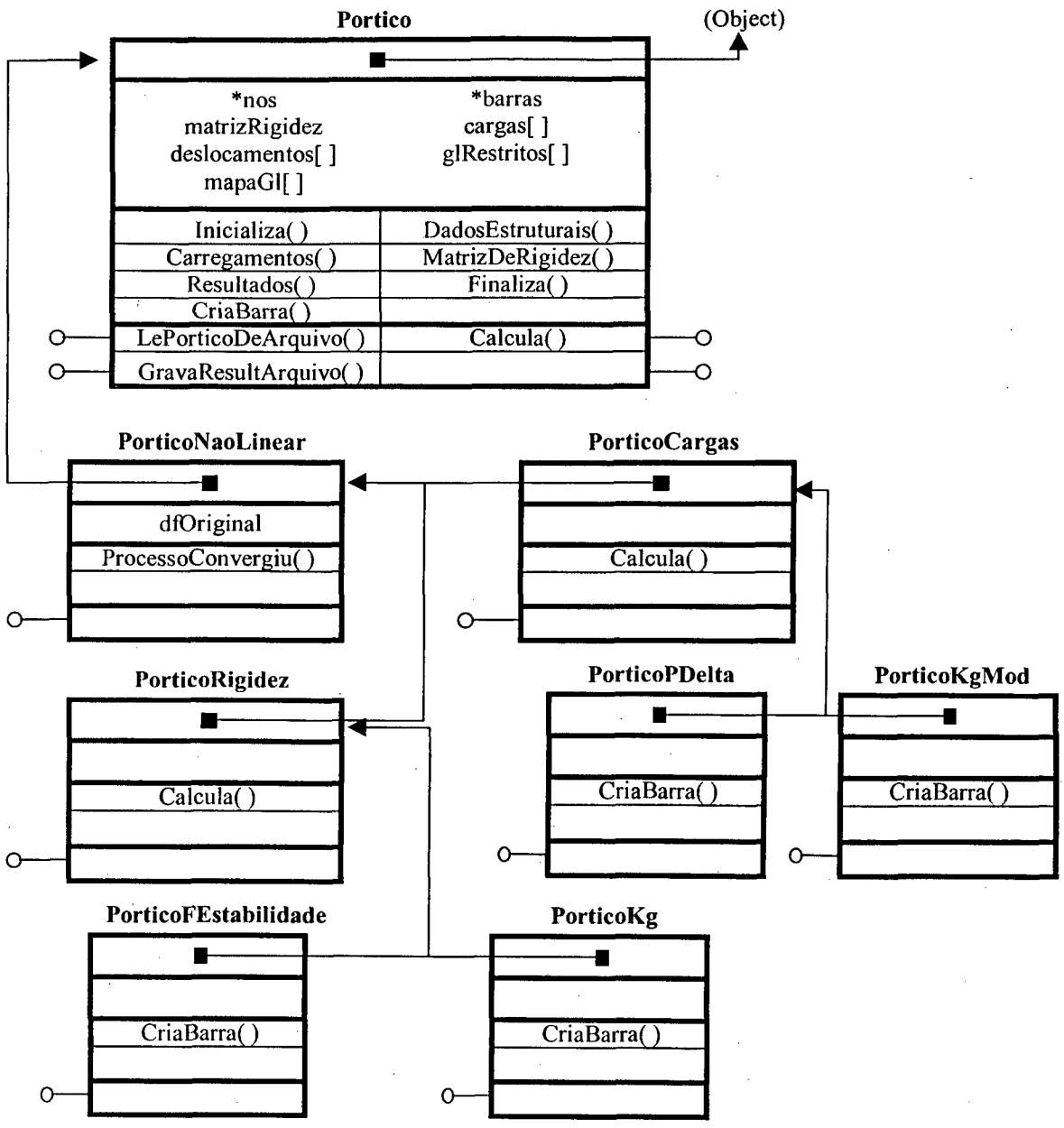


Figura 4.8 – Modelo de objetos para pórtico plano incluindo a não linearidade geométrica

Pode-se notar na estrutura de classes definida que nenhum método foi duplicado. Existem:

- três métodos *Calcula( )*, sendo um para a análise de 1ª ordem, um para os processos P-Delta e K_{GM} e um para os outros dois;
- cinco métodos *CriaBarra( )*, sendo um para cada tipo de pórtico;
- um método *ProcessoConvergiu( )*, apenas na classe base para os processos não lineares geométricos;

- uma implementação de cada método comum, todos na classe Portico.

Derivação da classe Barra

A classe base Barra deve ser derivada para refletir a diferença em cada formulação. Deve-se lembrar que algumas delas alteram o método *AcoesEngPerf( )* e outras o método *MatrizRigidezLocal( )*. Cada classe derivada pode sobrescrever apenas os métodos necessários, mantendo os demais. Poderia ser montada uma estrutura de classes como a seguinte:

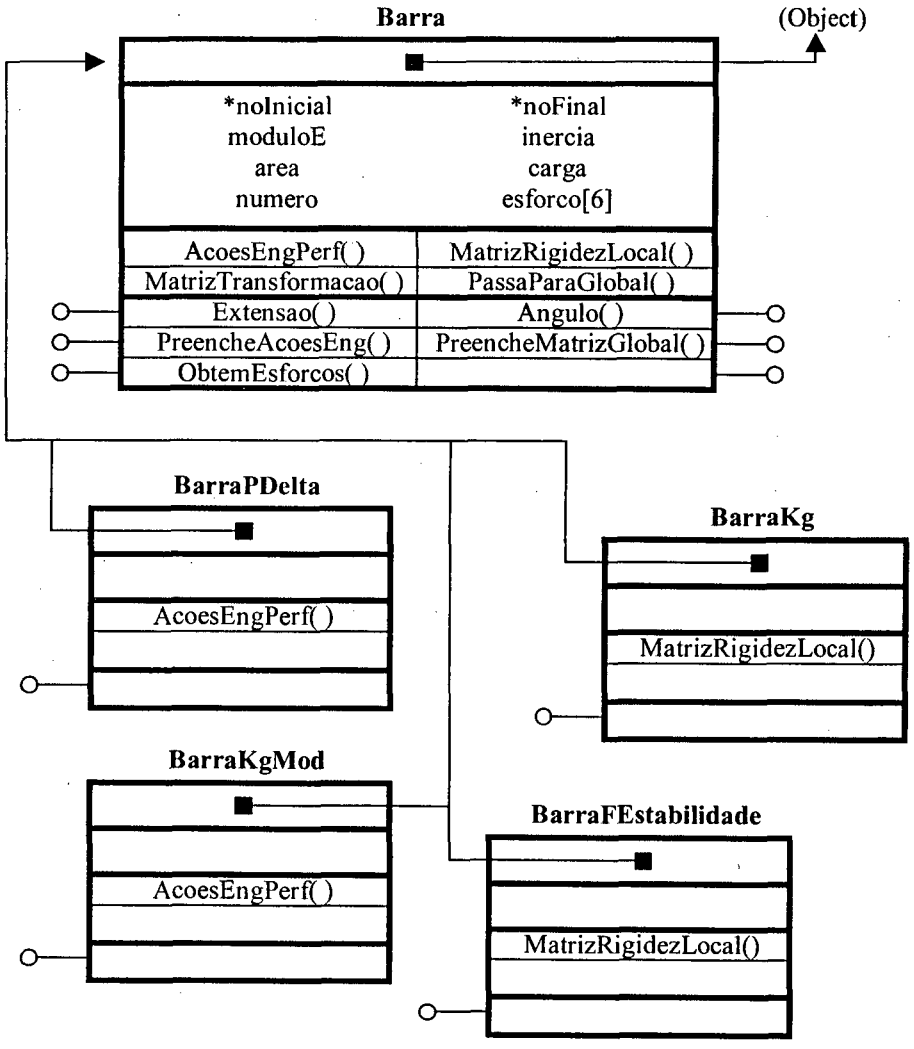


Figura 4.9 – Modelo de objetos para uma barra de pórtico plano incluindo a não linearidade geométrica

Comentários

Com o auxílio da OOP, pode-se escrever cinco programas para resolução de pórticos planos, cada um com uma maneira de considerar a não linearidade geométrica, com o mínimo de duplicação de código. Com isto, podem ser confirmadas algumas vantagens já discutidas no item 4.8:

- A clareza do código é muito maior, visto que cada classe encapsula um comportamento específico e a diferença entre cada processo pode ser visualizada mais facilmente;
- Todo o código escrito para as classes originais pode ser mantido, sem duplicação, facilitando a manutenção;
- Outras modificações (outros processos, outros tipos de carregamentos, etc.) podem ser incluídas mais facilmente.

CAPÍTULO 5.
EXEMPLOS NUMÉRICOS

Neste item, objetiva-se aplicar os princípios abordados no cálculo de alguns pórticos típicos, considerando a não linearidade geométrica das estruturas.

5.1 - PÓRTICO APOIADO NA BASE

Na figura abaixo, apresenta-se o esquema estrutural de um edifício simples, sujeito a esforços verticais e horizontais:

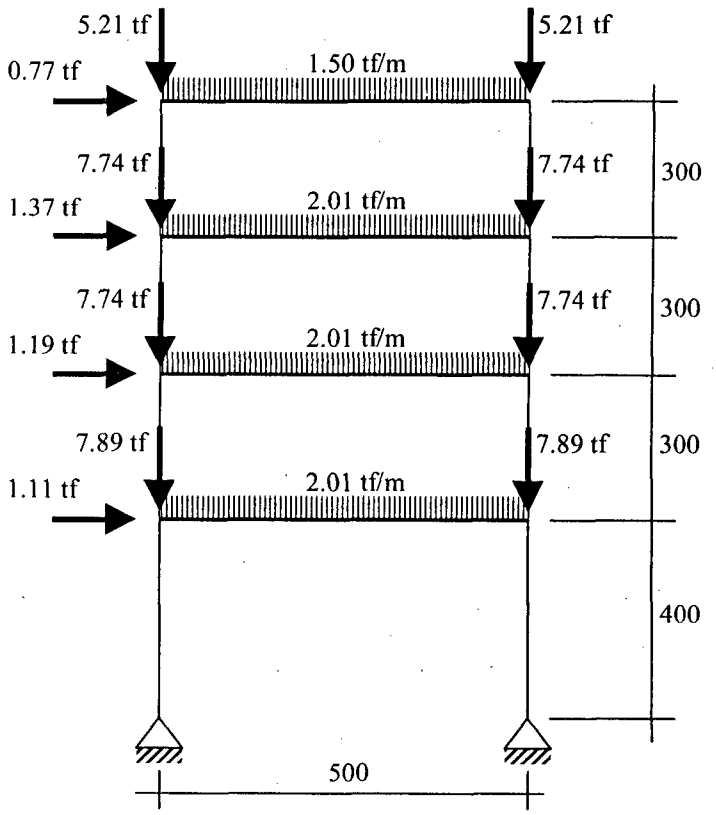


Figura 5.1 – Exemplo 2 – Pórtico apoiado na base

Serão utilizadas as seguintes propriedades:

- Módulo de elasticidade: $2{,}7 \times 10^5$ kgf/cm^2
- Seção transversal das barras: 30 x 20 cm para as barras verticais (pilares) e 13 x 55 cm para as barras horizontais (vigas).

Este exemplo foi inicialmente apresentado por PITTA (1996) e refere-se ao pórtico equivalente de uma estrutura simples, sujeita a carregamentos de vento e de utilização usuais.

5.1.1 Análise de 1 ° ordem

Os resultados obtidos a partir da análise linear da estrutura são os deslocamentos nodais de 1° ordem e os esforços internos em cada barra.

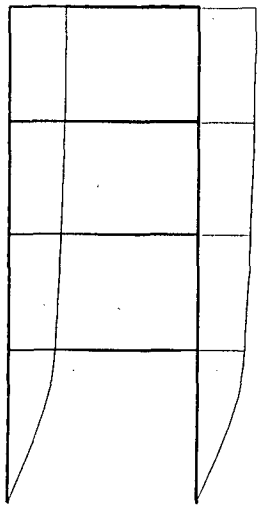


Figura 5.2 – Exemplo 2 - Estrutura deformada

Obtiveram-se os seguintes deslocamentos horizontais médios:

Nível (cm)	Deslocamento horizontal (cm)
1300	11.561
1000	11.298
700	10.657
400	9.543

Tabela 5.1 – Exemplo 2 – Deslocamentos obtidos na análise de 1ª ordem

Os momentos fletores nas barras apresentaram a seguinte configuração:

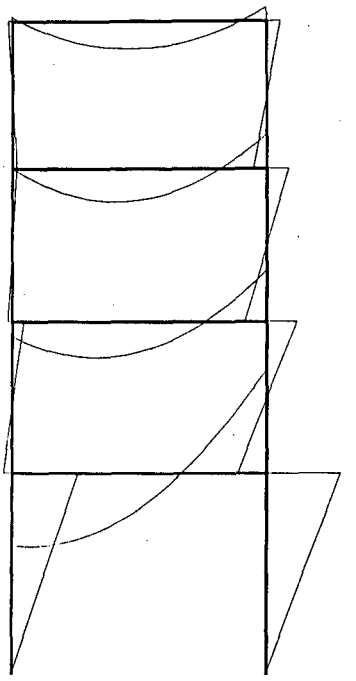


Figura 5.3 – Exemplo 2 – Diagrama de momentos fletores

5.1.2 Parâmetros de instabilidade

Com base nos resultados obtidos através da análise de 1ª ordem, pode-se avaliar a flexibilidade da estrutura através dos dois processos simplificados descritos nos itens 2.4.3 (Parâmetro Alfa) e 2.4.4 (Coeficiente Gama-Z).

Para a estrutura apresentada, obtiveram-se os seguintes valores:

- $\alpha = 1.063$
- $\gamma_z = 1.386$

O uso destes coeficientes em projeto pode indicar as seguintes conclusões sobre a estrutura calculada:

- O parâmetro α, por apresentar valor bastante superior ao limite definido de 0.6, indica que os efeitos de 2ª ordem devem ser levados em conta, mas não fornece informação sobre a magnitude destes;
- O coeficiente γ_z tem o objetivo de prever a diferença entre os esforços calculados pela teoria de 1ª ordem e de 2ª ordem, indicando aqui que esta diferença supera em muito o limite de 10% considerado admissível.

5.1.3 Análise através do Processo P-Delta

A fim de se aplicar o Processo P-Delta descrito no item 3.2, deve-se definir apenas uma tolerância limite para considerar a convergência nos resultados. Neste exemplo, vai-se utilizar um erro máximo de 1%. Esta decisão é de Engenharia e depende apenas da precisão requerida pela solução.

Aplicando-se o Processo P-Delta ao modelo, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	11.563	
1	16.590	43.483
2	19.072	14.958
3	20.313	6.507
4	20.934	3.059
5	21.246	1.487
6	21.401	0.734

Tabela 5.2 – Exemplo 2 por P-Delta – Deslocamentos obtidos em cada iteração

5.1.4 Análise utilizando o processo K_G

Aplicando-se os mesmos critérios, mas agora utilizando o Processo da Matriz de Rigidez Geométrica (K_G) descrito no item 3.3, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	11.563	
1	25.808	123.201
2	25.827	0.075

Tabela 5.3 – Exemplo 2 por K_G – Deslocamentos obtidos em cada iteração

5.1.5 Análise utilizando o processo K_{GM}

Utilizando o Processo da Matriz de Rigidez Geométrica Modificado (K_{GM}) descrito no item 3.4, seriam obtidos teoricamente os mesmos resultados do processo K_G. Na prática, uma vez que é necessário estabelecer uma precisão para a obtenção da solução, deve-se obter resultados um pouco diferentes.

Aplicando-se os mesmos critérios, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	11.563	
1	17.427	50.723
2	20.850	19.637
3	22.875	9.716
4	24.076	5.250
5	24.789	2.958
6	25.211	1.704
7	25.462	0.994

Tabela 5.4 – Exemplo 2 por K_{GM} – Deslocamentos obtidos em cada iteração

5.1.6 Análise utilizando o Processo das Funções de Estabilidade

Aplicando-se os mesmos critérios, mas agora utilizando o Processo das Funções de Estabilidade descrito no item 3.5, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	11.563	
1	25.937	124.319
2	25.964	0.102

Tabela 5.5 – Exemplo 2 pelas funções de estabilidade – Deslocamentos obtidos em cada iteração

5.1.7 Comparação dos resultados

Pode-se comparar os resultados obtidos através dos diferentes processos analisando uma série de itens na estrutura. O item mais significativo refere-se ao deslocamento do topo da estrutura, no ponto da aplicação da carga horizontal. Considera-se que o valor obtido pelo Processo das Funções de Estabilidade é o "exato", indicando nos demais a diferença apresentada.

Obtém-se os seguintes valores:

Processo	Deslocamento no topo (cm)	Diferença (%)
1ª ordem	11.563	-55.47
P-Delta	21.401	-17.57
K_G	25.827	-0.53
K_{GM}	25.462	-1.93
Funções de estabilidade	25.964	

Tabela 5.6 – Exemplo 2 – Comparação entre os processos

Observa-se, para este exemplo, que o erro obtido com a aplicação do Processo P-Delta pode ser considerado significativo, enquanto que o uso da matriz K_G chegou praticamente aos mesmos resultados do Processo das Funções de Estabilidade. O processo K_{GM} apresenta resultados muito próximos aos de KG, apenas com maior número de iterações.

Outra questão referente aos processos P-Delta e K_{GM} pode ser verificada analisando-se a evolução dos deslocamentos de acordo com o número de iterações:

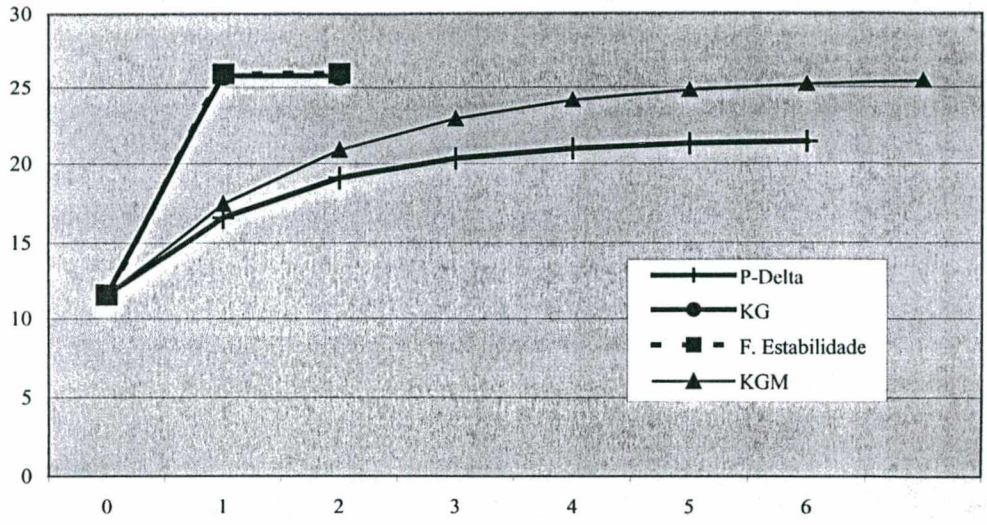


Figura 5.4 – Exemplo 2 – Evolução dos deslocamentos em cada processo

Pode-se notar que, enquanto que os processos K_G e das funções de estabilidade praticamente obtém o resultado correto já na primeira iteração, os processos P-Delta e K_{GM} possuem uma convergência muito mais lenta. Uma vez que se calcula a convergência nos deslocamentos com base em uma iteração anterior, a diferença acumulada pode ser maior do que o limite estabelecido.

Para verificar isto, pode-se calcular o mesmo exemplo pelo Processo P-Delta, mas agora utilizando uma precisão mínima de 0.05% ao invés de 1%.

Iteração	Deslocamento no topo (cm)	Diferença para a anterior (%)	Diferença acumulada (%)
5 (anterior)	21.246	1.487	0.000
6	21.401	0.734	0.734
7	21.479	0.365	1.097
8	21.519	0.182	1.285
9	21.538	0.091	1.374
10	21.548	0.046	1.421

Tabela 5.7 – Exemplo 2 por P-Delta – Deslocamentos considerando precisão adicional

Levando-se a precisão a 0.01%, o mesmo exemplo obteria um deslocamento no topo de 21.557 cm, o que representa uma diferença de 1.46% em relação ao deslocamento no qual se considerou uma precisão suficiente de 1% (5ª iteração). Deve-se notar, portanto:

- A imposição de uma precisão mínima deve ser feita considerando um valor abaixo do desejado, para levar em conta o erro acumulado posterior;
- Este problema é apresentado tanto pelo Processo P-Delta como pelo K_{GM},. dentre os apresentados, terem convergência nem sempre rápida;

Mesmo com maior precisão, o resultado final ainda está 16.97% abaixo daquele apresentado pelas funções de estabilidade. Fazendo o mesmo procedimento para o processo K_{GM}, obtém-se um valor de 25.824 a uma precisão de 0.01%. Pode-se notar que este processo realmente tende ao resultado apresentado por K_G e que, embora apresente convergência tão lenta como o processo P-Delta, os erros envolvidos são muito menores.

5.2 - PÓRTICO ENGASTADO NA BASE

Como segundo exemplo, vai-se repetir o Exemplo 2, alterando-se apenas a condição de vinculação da fundação, de apoiada para engastada. Com isto, pretende-se avaliar a influência desta vinculação na flexibilidade da estrutura e, consequentemente, nos valores obtidos pela teoria de 2ª ordem.

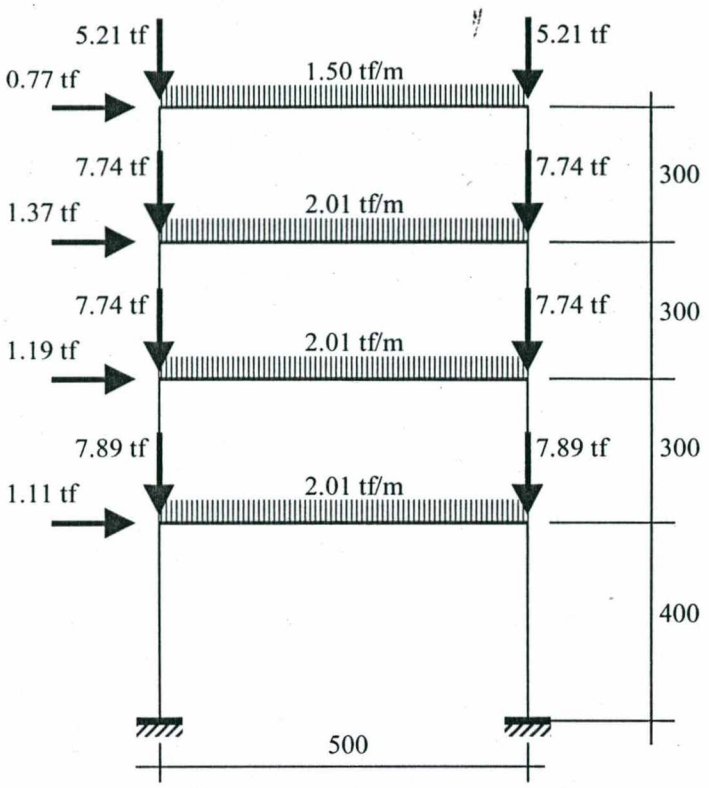


Figura 5.5 – Exemplo 3 – Pórtico engastado na base

5.2.1 Análise de 1 º ordem

Os resultados obtidos a partir da análise linear da estrutura são os deslocamentos nodais de 1º ordem e os esforços internos em cada barra.

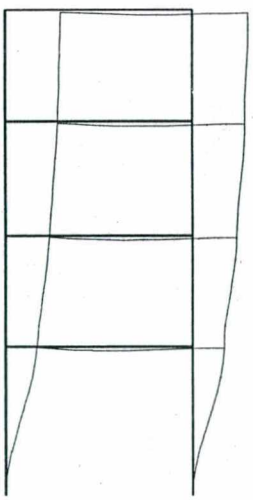


Figura 5.6 – Exemplo 3 - Estrutura deformada

Obtiveram-se os seguintes deslocamentos horizontais médios:

Nível (cm)	Deslocamento horizontal (cm)
1300	4.307
1000	4.050
700	3.418
400	2.428

Tabela 5.8 – Exemplo 3 – Deslocamentos obtidos na análise de 1ª ordem

Os momentos fletores nas barras apresentaram a seguinte configuração:

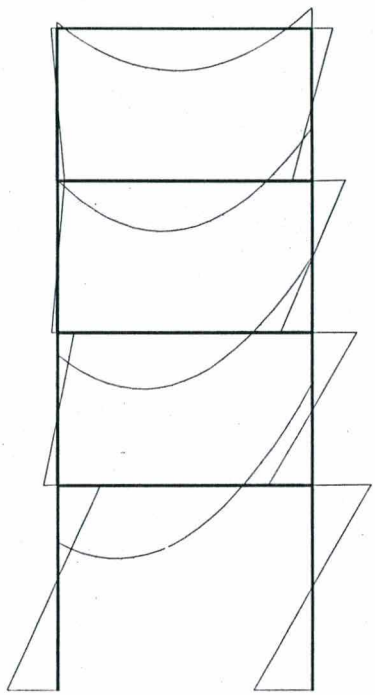


Figura 5.7 – Exemplo 3 – Diagrama de momentos fletores

5.2.2 Parâmetros de instabilidade

Com base nos resultados obtidos através da análise de 1ª ordem, pode-se avaliar a flexibilidade da estrutura através dos dois processos simplificados descritos nos itens 2.4.3 (Parâmetro Alfa) e 2.4.4 (Coeficiente Gama-Z).

Para a estrutura apresentada, obtiveram-se os seguintes valores:

- $\alpha = 0.649$
- $\gamma_z = 1.100$

O uso destes coeficientes em projeto pode indicar as seguintes conclusões sobre a estrutura calculada:

- O parâmetro α, por apresentar valor apenas ligeiramente superior ao limite definido de 0.6, poderia indicar que os efeitos de 2ª ordem são pequenos;
- O coeficiente γ_z indicaria aqui que esta estrutura estaria dentro do limite de 10% considerado admissível.

5.2.3 Análise através do Processo P-Delta

Aplicando-se o Processo P-Delta ao modelo, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	4.309	
1	4.729	43.483
2	4.776	1.000
3	4.782	0.119

Tabela 5.9 – Exemplo 3 por P-Delta – Deslocamentos obtidos em cada iteração

5.2.4 Análise utilizando o processo K_G

Aplicando-se os mesmos critérios, mas agora utilizando o Processo da Matriz de Rigidez Geométrica (K_G) descrito no item 3.3, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	4.309	
1	4.866	12.917
2	4.866	0.001

Tabela 5.10 – Exemplo 3 por K_G – Deslocamentos obtidos em cada iteração

5.2.5 Análise utilizando o processo K_{GM}

Utilizando o Processo da Matriz de Rigidez Geométrica Modificado (K_{GM}) descrito no item 3.4, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	4.309	
1	4.792	11.201
2	4.855	1.326
3	4.864	0.184

Tabela 5.11 – Exemplo 3 por K_{GM} – Deslocamentos obtidos em cada iteração

5.2.6 Análise utilizando o Processo das Funções de Estabilidade

Aplicando-se os mesmos critérios, mas agora utilizando o Processo das Funções de Estabilidade descrito no item 3.5, foram necessárias as seguintes iterações:

Iteração	Deslocamento no topo (cm)	Diferença (%)
0 (1ª ordem)	4.309	
1	4.867	12.936
2	4.867	0.001

Tabela 5.12 – Exemplo 3 pelas funções de estabilidade – Deslocamentos obtidos em cada iteração

5.2.7 Comparação dos resultados

Utilizando-se novamente o deslocamento horizontal no topo da estrutura para comparar os três processos, obtém-se os seguintes valores:

Processo	Deslocamento no topo (cm)	Diferença (%)
1ª ordem	4.309	-11.46
P-Delta	4.782	-1.75
K_G	4.866	-0.02
K_{GM}	4.864	-0.06
Funções de estabilidade	4.867	

Tabela 5.13 – Exemplo 3 – Comparação entre os processos

Observa-se, para este exemplo, que os erros obtidos com a aplicação do Processo P-Delta são muito menores que no exemplo anterior, com os processos K_G e K_{GM} apresentando erro absolutamente desprezível. Analisando-se novamente a evolução dos deslocamentos de acordo com o número de iterações, obtém-se:

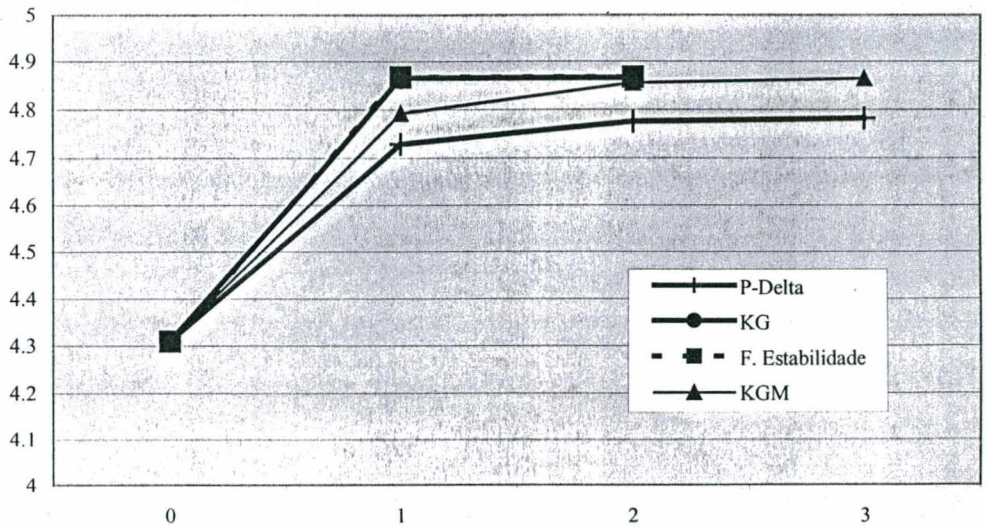


Figura 5.8 – Exemplo 3 – Evolução dos deslocamentos em cada processo

Pode-se notar que, desta vez, mesmo os processos P-Delta e K_{GM} possuem uma convergência bastante rápida. Por exemplo, refazendo-se o exemplo através do Processo P-Delta, mas considerando uma precisão mínima de 0.01%, chega-se ao mesmo resultado de 4.782.

Este exemplo, em comparação com o exemplo anterior, parece indicar que, quanto mais flexível for a estrutura (ou seja, maior for a diferença entre a análise de 1ª e 2ª ordem):

- mais iterações são necessárias para se chegar a uma precisão preestabelecida;
- maior a diferença (erro relativo) apresentada entre os processos apresentados.

5.3 - INFLUÊNCIA DA DISCRETIZAÇÃO

Conforme exposto no item 3.2.4, referente ao Processo P-Delta, e no item 3.3.5, referente ao Processo da Matriz de Rigidez Geométrica, os resultados apresentados podem ser refinados através da subdivisão da barra em elementos menores. Afirmou-se também que a necessidade de subdivisão é tanto maior quanto maiores forem os efeitos de segunda ordem, podendo usualmente ser associado à razão entre a carga normal atuante na barra e a carga crítica de Euler para a mesma barra.

Vai-se tentar observar agora o mesmo comportamento já encontrado no Exemplo 1, com uma barra isolada, nos exemplos maiores correspondentes ao Exemplo 2 (pórtico apoiado na base) e Exemplo 3 (pórtico engastado na base).

5.3.1 Discretização uniforme

Uma vez que já se constatou que a necessidade de subdivisão está relacionada com a carga normal, pode-se fazer diretamente a discretização apenas nas barras verticais (pilares), visto serem desprezíveis os esforços axiais nas barras horizontais.

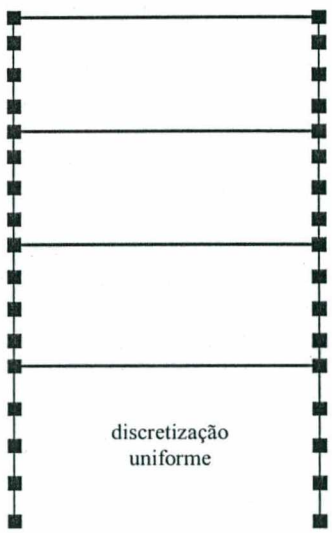


Figura 5.9 – Exemplo 2 e Exemplo 3 – Discretização da estrutura

Estipulando-se como precisão mínima o valor de 0.1% e adotando-se novamente como referência o deslocamento no topo da estrutura, obtém-se os seguintes valores para o caso do Exemplo 2 (apoiado):

Nº de subdivisões	Funções de estabilidade	P-Delta	K_G	K_{GM}
1	25.963	21.538	25.827	25.800
2	25.963	24.456	25.963	25.924
5	25.963	25.678	25.963	25.934
10	25.963	25.869	25.963	25.934
20	25.963	25.918	25.963	25.934
40	25.963	25.930	25.963	25.934
100	25.963	25.934	25.963	25.934

Tabela 5.14 – Exemplo 2 – Variação no deslocamento no topo conforme a discretização

Utilizando-se os mesmos critérios para o caso do Exemplo 3 (engastado):

Nº de subdivisões	Funções de estabilidade	P-Delta	K_G	K_{GM}
1	4.867	4.782	4.866	4.866
2	4.867	4.783	4.866	4.866
5	4.867	4.850	4.867	4.866
10	4.867	4.862	4.867	4.866
20	4.867	4.865	4.867	4.866
40	4.867	4.866	4.867	4.866
100	4.867	4.866	4.867	4.866

Tabela 5.15 – Exemplo 3 – Variação no deslocamento no topo conforme a discretização

A partir destes exemplos, pode-se obter algumas considerações:

- Os resultados obtidos através do Processo das Funções de Estabilidade independem da discretização, caracterizando-se como "exatos" mesmo sem subdivisões;
- O Processo P-Delta converge para o resultado exato conforme a subdivisão, mas sem alcançá-lo realmente;

- No caso do Exemplo 2, é extremamente recomendável a subdivisão da barra para utilização do Processo P-Delta;
- O processo K_G converge rapidamente para o resultado correto com um número de subdivisões bem inferior ao P-Delta, apresentando resultados muito próximos mesmo sem qualquer discretização;
- O processo K_{GM} também converge rapidamente para o resultado correto, mas a sua relação com o K_G é independente da discretização (mesmo com a barra discretizada, os resultados tenderão ao K_G conforme a precisão estabelecida entre as iterações);
- A relação com a carga crítica é apenas um fator indicativo no caso de barras isoladas, ficando o caso de estruturas complexas sujeitas também a outras questões como a disposição dos elementos e as condições de vinculação.

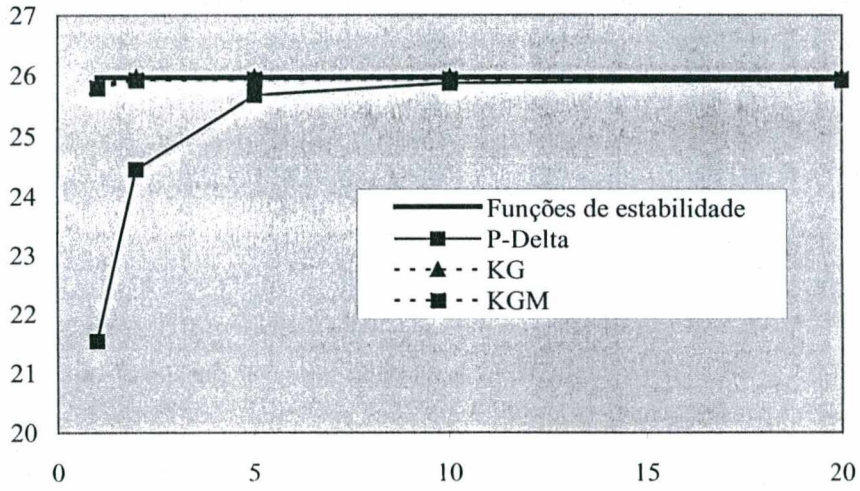


Figura 5.10 –Exemplo 2 – Variação no deslocamento no topo conforme a discretização

Refazendo-se os mesmos exemplos, mas agora aplicando-se a mesma subdivisão também às barras horizontais, observa-se a obtenção dos mesmos resultados já apresentados na Tabela 5.14 e na Tabela 5.15, o que confirma a suposição de que é suficiente discretizar as barras sujeitas a esforços normais relevantes.

5.3.2 Discretização não uniforme ao longo do elemento

Ao invés de se dividir as barras em partes iguais, deseja-se estudar o comportamento dos processos face a uma discretização não uniforme das barras. Pode-se definir, por exemplo, uma discretização onde faz-se um refinamento nas extremidades das barras, conforme a figura abaixo:

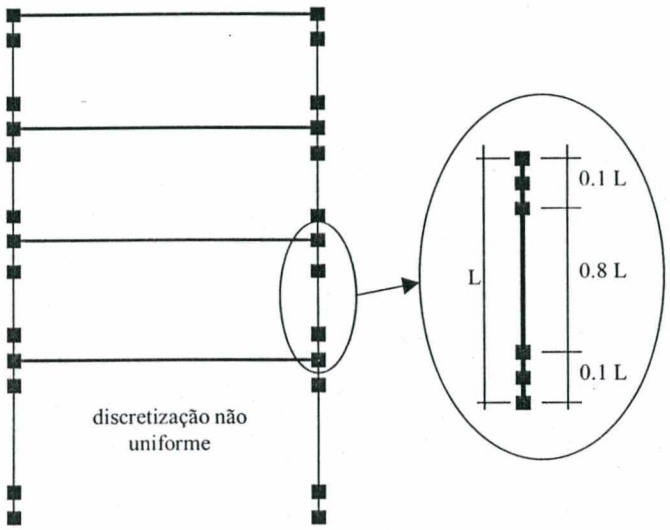


Figura 5.11 – Exemplo 2 e Exemplo 3 – Discretização da estrutura de forma desigual ao longo das barras

Verificando-se novamente o deslocamento no topo da estrutura, obtém-se:

Processo	Exemplo 2 (apoiado)	Exemplo 3 (engastado)
P-Delta	23.426	4.837
K_G	25.917	4.866
K_{GM}	25.889	4.866
Funções de estabilidade	25.963	4.867

Tabela 5.16 – Exemplo 2 e Exemplo 3 – Deslocamento no topo para discretização concentrada nas extremidades das barras

Observa-se que, com exceção do Processo das Funções de Estabilidade, que independe da discretização, os resultados obtidos foram inferiores àqueles obtidos com a mesma quantidade de barras, mas com divisões iguais (5 divisões). Verifica-se que não há vantagem em concentrar a discretização em partes das barras.

5.3.3 Discretização não uniforme ao longo da estrutura

Uma forma de otimizar a discretização da estrutura é levar em consideração o fato de que as barras sujeitas a maiores efeitos de 2ª ordem necessitam de mais subdivisões. Pode-se levar em conta este efeito, de forma aproximada, relacionando a carga normal atuante em cada barra (após a análise de 1ª ordem) com a respectiva carga crítica. Nos dois exemplos em questão, os efeitos de 2ª ordem concentram-se no lance inferior, justificando uma discretização diferenciada.

Por exemplo, vai-se definir uma discretização variável para o primeiro lance e manter os lances superiores divididos em apenas duas partes.

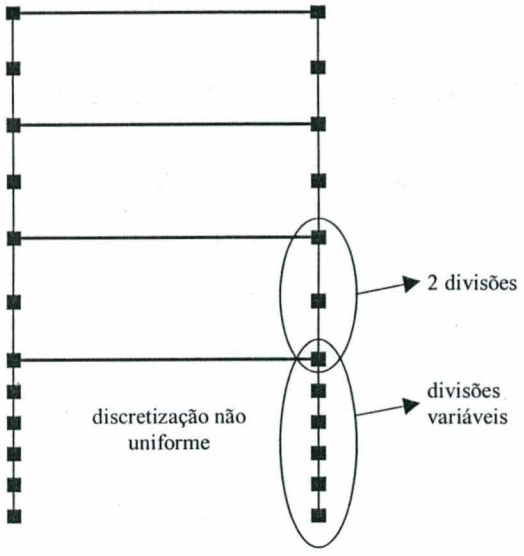


Figura 5.12 – Exemplo 2 e Exemplo 3 – Discretização da estrutura de forma desigual ao longo dos lances

Estipulando-se como precisão mínima o valor de 0.1% e adotando-se novamente como referência o deslocamento no topo da estrutura, obtém-se os seguintes valores:

	Exemplo 2 (apoiado)		Exemplo 3 (engastado)	
Nº de subdivisões	P-Delta	K_G	P-Delta	K_G
5	25.668	25.963	4.841	4.867
10	25.857	25.963	4.851	4.867
20	25.905	25.963	4.854	4.867
40	25.917	25.963	4.855	4.867

Tabela 5.17 – Exemplo 2 e Exemplo 3 – Variação no deslocamento no topo conforme a discretização (concentrada no lance inferior)

Observa-se que os resultados encontrados são muito bons para o Exemplo 2 (vide Tabela 5.14), onde os deslocamentos da estrutura concentram-se basicamente ao nível do primeiro lance. Todavia, para o Exemplo 3, as diferenças obtidas pelo Processo P-Delta já foram significativas. Não se verifica diferença com o auxílio do Processo da Matriz de Rigidez Geométrica porque, para estes exemplos, mesmo a discretização em apenas duas partes já fornece resultados praticamente iguais aos exatos.

De qualquer forma, nota-se que, em caso de discretização da estrutura, é conveniente fazê-lo de forma diferenciada, o que pode diminuir muito o custo computacional da solução.

5.4 - EFEITO DO AUMENTO DA FORÇA NORMAL

Conforme já observado no Exemplo 1, o aumento do nível de força normal em um elemento leva a maiores efeitos de 2ª ordem, aumentando também a necessidade de discretização da estrutura. Observou-se também que existe uma carga limite que a estrutura pode suportar acima da qual não se obtém resultados convergentes. Deseja-se agora verificar as mesmas conclusões nos exemplos maiores abordados.

Para esta verificação, tomar-se-á novamente o Exemplo 3 (pórtico engastado na base), mas acrescentando à estrutura duas cargas verticais P_{adic} em seu topo.

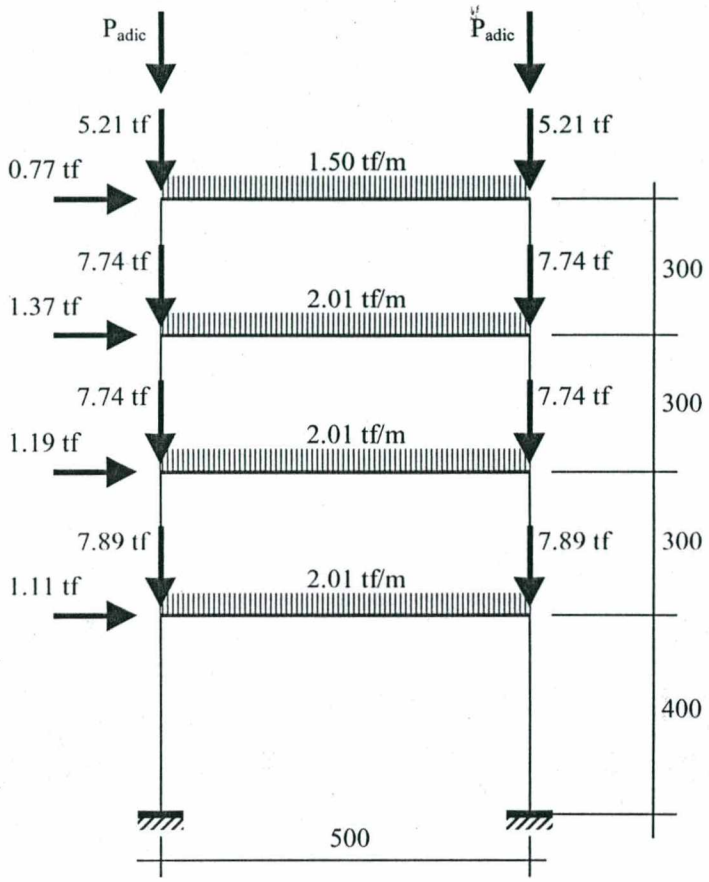


Figura 5.13 – Exemplo 3 – Cargas adicionais aplicadas no topo

Adotando valores crescentes de P_{adic}, obtém-se os seguintes resultados[12].

[12] Lembrando que os valores obtidos através do Processo das Funções de Estabilidade não variam com a discretização e podem ser considerados "exatos".

Nº de divisões	P_{adic} = 200 tf (Δ_{exato} = 15.489)			P_{adic} = 250 tf (Δ_{exato} = 48.990)			P_{adic} = 260 tf (Δ_{exato} = 100.455)		
	P-Δ	K_G	K_{GM}	P-Δ	K_G	K_{GM}	P-Δ	K_G	K_{GM}
1	10.934	15.127	15.085	16.824	42.684	42.157	18.970	73.238	71.587
2	10.975	15.285	15.238	16.965	45.205	44.590	19.157	82.684	80.506
5	14.287	15.483	15.428	34.751	48.870	48.144	51.518	99.814	96.400
10	15.125	15.489	15.434	43.831	48.982	48.246	78.750	100.414	96.986
20	15.355	15.489	15.434	47.078	48.990	48.252	91.665	100.453	97.018
40	15.414	15.489	15.434	47.975	48.990	48.252	95.837	100.455	97.020
100	15.431	15.489	15.434	48.220	48.990	48.252	97.039	100.455	97.020

Tabela 5.18 – Exemplo 3 – Variação no deslocamento no topo conforme o aumento da força normal e o nível de discretização

Como parâmetro de comparação, pode-se relacionar cada exemplo com a menor relação entre carga e carga crítica encontrada na estrutura. Neste exemplo, isto ocorre na barra 9, por ter simultaneamente o maior nível de força normal e a menor carga crítica (maior comprimento). Calculando-se a carga crítica de Euler para a barra 9, chega-se a P_E = 333.1 tf. As três variações expostas na

Nº de divisões	P_{adic} = 200 tf (Δ_{exato} = 15.489)			P_{adic} = 250 tf (Δ_{exato} = 48.990)			P_{adic} = 260 tf (Δ_{exato} = 100.455)		
	P-Δ	K_G	K_{GM}	P-Δ	K_G	K_{GM}	P-Δ	K_G	K_{GM}
1	10.934	15.127	15.085	16.824	42.684	42.157	18.970	73.238	71.587
2	10.975	15.285	15.238	16.965	45.205	44.590	19.157	82.684	80.506
5	14.287	15.483	15.428	34.751	48.870	48.144	51.518	99.814	96.400
10	15.125	15.489	15.434	43.831	48.982	48.246	78.750	100.414	96.986
20	15.355	15.489	15.434	47.078	48.990	48.252	91.665	100.453	97.018
40	15.414	15.489	15.434	47.975	48.990	48.252	95.837	100.455	97.020
100	15.431	15.489	15.434	48.220	48.990	48.252	97.039	100.455	97.020

Tabela 5.18 apresentam carga normal na barra 9 (logo após a análise de 1ª ordem) de 254.7, 302.9 e 312.9 tf, respectivamente.

Observa-se que estes valores estão bastante próximos da carga crítica, justificando, portanto, a grande diferença obtida nos resultados. Deve-se lembrar também que os efeitos de 2ª ordem aumentam a força normal nesta barra.

Para uma carga adicional de 270 tf, não é possível obter uma solução para o sistema, caracterizando-se como o ponto crítico desta estrutura.

Confirma-se, portanto, que:

- O Processo das Funções de Estabilidade independe da discretização para qualquer nível de esforço normal;
- O Processo da Matriz de Rigidez Geométrica só apresenta erros significativos para cargas próximas da crítica;
- O Processo da Matriz de Rigidez Geométrica Modificado apresenta resultados comparáveis aos de K_G, podendo-se notar que a diferença entre os dois processos cresce ligeiramente com o aumento da força normal, devido ao fato de que a convergência torna-se mais lenta;
- O Processo P-Delta é muito mais sensível ao aumento da força normal, levando a um erro considerável mesmo com P_{adic} = 200 tf;
- Quanto maior o nível de carga normal, mas subdivisões são necessárias para se obter uma precisão preestabelecida.

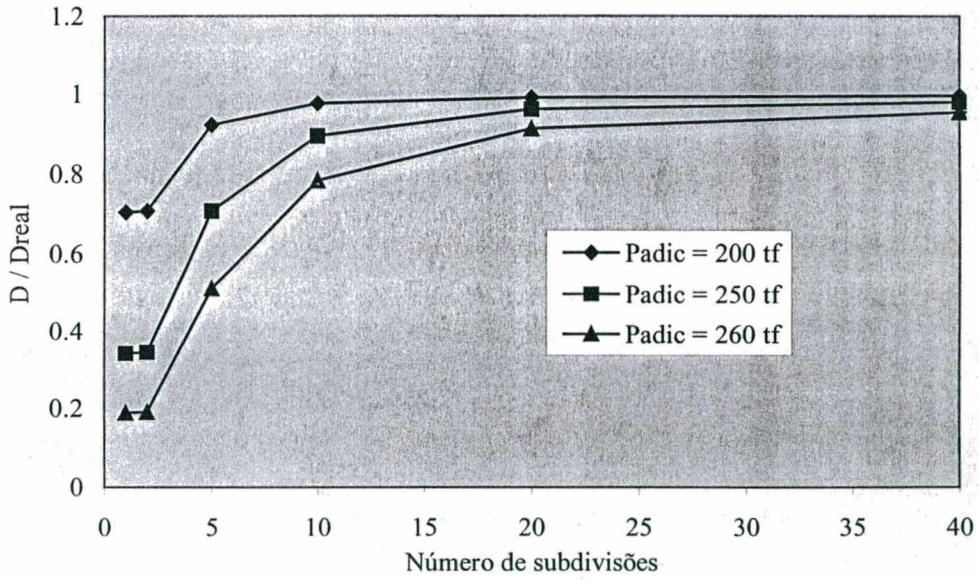


Figura 5.14 –Exemplo 3 por P-Delta – Variação no deslocamento no topo conforme o aumento da força normal e o nível de discretização

5.5 - O COEFICIENTE GAMA-Z E OS ESFORÇOS NAS BARRAS

Até o momento, utilizou-se apenas os deslocamentos como comparação entre os processos apresentados para consideração da não linearidade geométrica na solução de pórticos planos. Isto porque as diferenças obtidas praticamente repetem todas as conclusões efetuadas.

Por outro lado, SANTOS & FRANCO (1993) afirmam que é possível "determinar de forma aproximada o coeficiente γ_z de majoração dos esforços globais finais com relação aos de primeira ordem". O valor de γ_z é o definido pela Eq. (2-19):

$$\gamma z = \frac{1}{1 - \frac{\Delta M_{tot,d}}{M1_{tot,d}}} \qquad \text{Eq. (2-19)}$$

O valor de γ_z fornece uma indicação sobre a magnitude dos esforços de 2ª ordem, apontando para a necessidade ou não de uma análise mais elaborada. Mais adiante, os autores permitem estender este conceito, permitindo "a avaliação dos esforços finais (1ª + 2ª ordem) pela multiplicação por γ_z dos esforços de 1ª ordem". Isto pode ser feito apenas para estruturas "regulares" e desde que $\gamma_z \leq 1.2$. Segundo os autores, esta será a abordagem a ser dada pela nova NBR 6118.

Não é imediato o conceito de "estrutura regular" mencionado como limitação à aplicação do processo. Pode-se dizer que se refere a uma distribuição regular entre os elementos (vigas e pilares) tanto em planta quanto entre os pavimentos. Por este critério, a estrutura utilizada no Exemplo 2 e no Exemplo 3 poderia perfeitamente ser classificada como regular.

A fim de avaliar a variação nos esforços internos das barras, vai-se explicitar uma numeração para os elementos da estrutura:

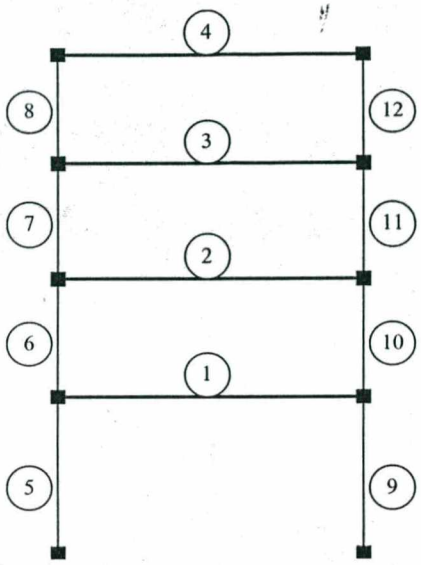


Figura 5.15 – Exemplo 2 e Exemplo 3 – Numeração das barras

Outra questão importante, não expressa claramente pelos autores, refere-se à identificação do que sejam os esforços afetados pelo efeito de 2ª ordem que podem ser relacionados diretamente ao coeficiente γ_z.

5.5.1 Análise de 1ª ordem

Como comparação, vai-se se estudar os momentos fletores ocorridos ao longo da prumada 5-8. Os resultados obtidos a partir da análise de 1ª ordem são:

	Exemplo 2 (apoiado)		Exemplo 3 (engastado)	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-576	-485	-576	-485
7	458	573	458	564
6	1035	1544	1204	1425
5	0	8354	4269	3623

Tabela 5.19 – Exemplo 2 e Exemplo 3 – Momentos fletores obtidos na análise de 1ª ordem

Nota-se um comportamento bastante semelhante nos níveis superiores, ficando a diferença nos momentos fletores concentrada basicamente no primeiro lance. A figura abaixo mostra, qualitativamente, a distribuição dos momentos fletores nos dois exemplos.

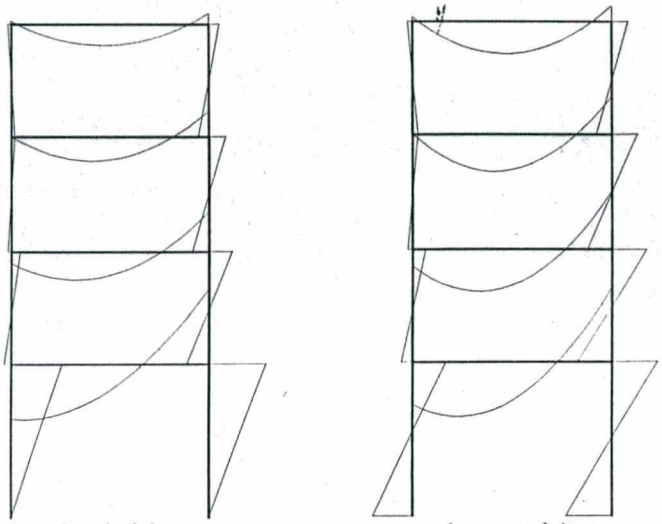


Figura 5.16 – Exemplo 2 e Exemplo 3 – Distribuição dos momentos fletores

5.5.2 Processo das Funções de Estabilidade

Resolvendo-se ambos os exemplos através do Processo das Funções de Estabilidade, obtém-se os seguintes resultados:

	Exemplo 2 (apoiado)		Exemplo 3 (engastado)	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-566	-470	-565	-470
7	536	672	530	647
6	872	2053	1378	1628
5	0	20246	4952	4333

Tabela 5.20 – Exemplo 2 e Exemplo 3 – Momentos fletores obtidos através do Processo das Funções de Estabilidade

5.5.3 Comparação através do coeficiente Gama-Z

Utilizando-se a expressão definida no item 2.4.4 para o coeficiente Gama-Z, obtém-se os seguintes valores:

- Exemplo 2: $\gamma_z = 1.386$
- Exemplo 3: $\gamma_z = 1.100$

Uma vez que um único valor representa toda a estrutura calculada, supõe-se que a diferença entre os momentos fletores ocorra da mesma forma em todas as barras. Desta forma, conforme SANTOS & FRANCO (1993), poderia-se simplesmente multiplicar os momentos da Tabela 5.19 pelo fator $\gamma_z = 1.1$ para se chegar aos esforços finais. Todavia, uma comparação entre os valores expressos na Tabela 5.19 (1ª ordem) e na Tabela 5.20 (1ª + 2ª ordem) resulta em:

	Exemplo 2 (apoiado)		Exemplo 3(engastado)	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-566(-1.7%)	-470(-3.1%)	-565(-1.9%)	-470(-3.1%)
7	536(+17.0%)	672(+17.3%)	530(+15.7%)	647(+14.7%)
6	872(-15.8%)	2053(+33.0%)	1378(+14.5%)	1628(+14.3%)
5	0	20246(+142.4%)	4952(+16.0%)	4333(+19.6%)

Tabela 5.21 – Exemplo 2 e Exemplo 3 – Diferença nos momentos fletores obtidos através do Processo das Funções de Estabilidade

Pode-se observar:

- Mesmo no Exemplo 3, considerado bastante regular, existe uma diferença significativa nos efeitos de 2ª ordem ao longo da prumada, passando desde uma variação negativa (-3.1%) na parte superior para um valor bastante superior aos 10% estimados por γ_z na parte inferior;
- No Exemplo 2, onde os efeitos de 2ª ordem concentram-se bastante no primeiro lance, não existe qualquer uniformidade em sua distribuição, ficando a variação de 38.6% predita por γ_z muito abaixo dos 142.4% realmente ocorridos no ponto mais crítico;
- A variação ocorrida nos dois lances superiores foi muito semelhante nos dois exemplos, mesmo para valores de γ_z completamente diferentes.

O diagrama a seguir mostra a diferença obtida ao se utilizar os momentos fletores simplesmente majorados por γ_z ao invés dos valores obtidos através da análise de 2ª ordem, para o caso do Exemplo 3, onde as diferenças são pequenas:

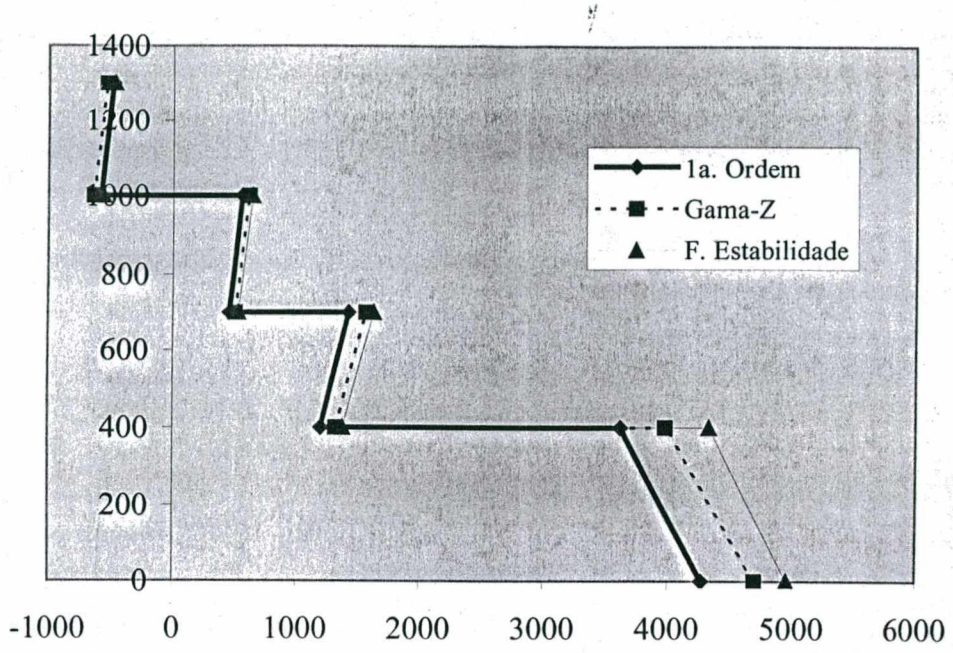


Figura 5.17 – Exemplo 3 – Diferença nos momentos fletores obtidos através do Processo das Funções de Estabilidade

O mesmo diagrama elaborado para o Exemplo 2 apresentaria-se da seguinte forma:

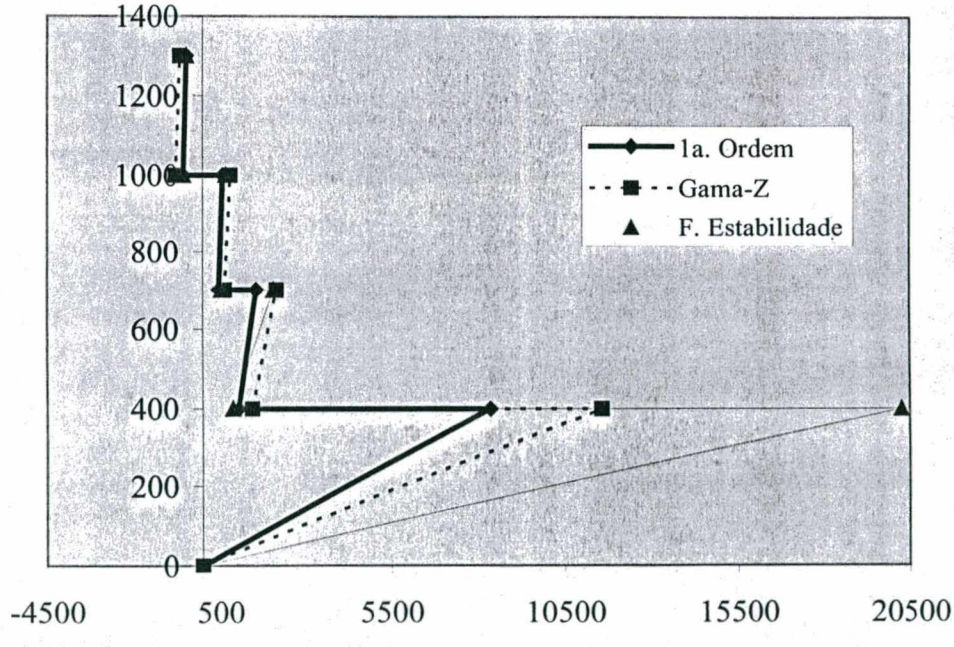


Figura 5.18 – Exemplo 2 – Diferença nos momentos fletores obtidos através do Processo das Funções de Estabilidade

A figura acima indica claramente a diferença contra a segurança que seria obtida com a aplicação direta do coeficiente γ_z sobre os esforços de 1ª ordem. Tomando apenas os lances superiores, seria visível também o erro a favor da segurança obtido:

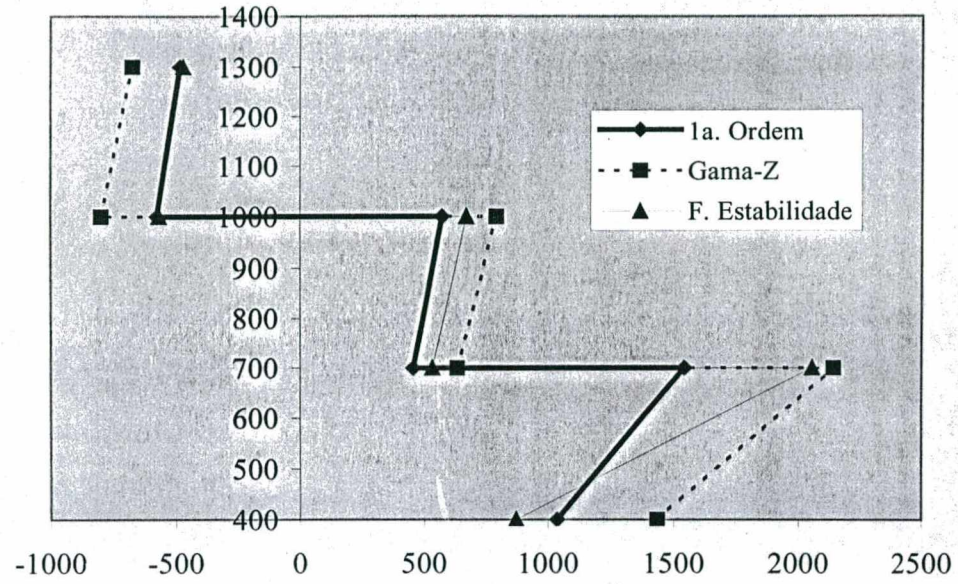


Figura 5.19 – Exemplo 2 – Diferença nos momentos fletores obtidos através do Processo das Funções de Estabilidade (excluindo o lance inferior)

Deve-se observar que a uniformização dos efeitos de 2ª ordem pode representar um erro grave contra a segurança. Mesmo no caso do Exemplo 3, o momento calculado por γ_z chegaria a ser 8.02% inferior ao momento real, o que pode ser considerado significativo. No caso do Exemplo 2, a aplicação de γ_z[13] incorreria em erro de –42.81%, absolutamente inaceitável.

5.5.4 Fatores intervenientes

A análise feita no item anterior pode levar à conclusão de que a diferença entre os momentos fletores preditos pelo coeficiente γ_z e aqueles calculados pela teoria de 2ª ordem são significativamente diferentes apenas quando o valor de γ_z é elevado ou quando a diferença de rigidez é significativa, caso das fundações apoiadas do Exemplo 2.

[13] Deve-se lembrar que este exemplo não seria permitido pelos autores, por ser $\gamma_z > 1.2$.

Como primeira verificação, pode-se tomar o Exemplo 3 (engastado) e torná-lo mais flexível, através da diminuição da seção dos pilares, de 30x20 cm para 30x15 cm. Com isto, o coeficiente γ_z apresentado pelo exemplo passa a 1.243. Obtém-se os seguintes resultados:

	1ª ordem		Funções de estabilidade	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-82	-18	-63(-23.2%)	9(-150.0%)
7	914	960	1070(+17.1%)	1122(+16.9%)
6	1758	1846	2153(+22.5%)	2278(+23.4%)
5	4319	3995	6321(+46.35%)	6038(+51.1%)

Tabela 5.22 –Exemplo 3 mais flexível – Diferença nos momentos fletores obtidos através do Processo das Funções de Estabilidade

Pode-se notar que, com o aumento da flexibilidade da estrutura, mesmo a estrutura considerada bastante regular leva a diferenças bastante significativas. Neste caso, o valor predito por γ_z estaria até 17.8% abaixo do valor real, o que pode ser considerado inaceitável mesmo para situações de projeto.

Pode-se fazer também o exemplo contrário, tomando o Exemplo 2 (apoiado) e tornando-o mais rígido, através do aumento da seção dos pilares para 30x25 cm. Com isto, o coeficiente γ_z apresentado para a estrutura reduz-se para 1.186. Obtém-se os seguintes resultados:

	1ª ordem		Funções de estabilidade	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-1005	-909	-1001(-0.4%)	-898(-1.21%)
7	93	335	133(+43.0%)	403(+20.3%)
6	390	1351	321(-17.7%)	1672(+23.8%)
5	0	8145	0	11911(+46.2%)

Tabela 5.23 –Exemplo 2 mais rígido – Diferença nos momentos fletores obtidos através do Processo das Funções de Estabilidade

Mais uma vez, observa-se que este exemplo não permite a adoção de um valor representativo para γ_z. Utilizando-se o valor calculado, chega-se a momentos fletores até 18.9% abaixo do valor real.

Destes exemplos, pode-se dizer que o uso do coeficiente γ_z para predizer os efeitos de 2ª ordem apresenta pouca confiabilidade, tendo gerado erros significativos mesmo para as estruturas bastante regulares apresentadas, e mesmo para valores relativamente baixos de γ_z. Estima-se que os resultados seriam razoáveis apenas para estruturas simultaneamente muito regulares (como a do Exemplo 3) e com valores de γ_z bastante baixos.

5.5.5 Processo P-Delta

É interessante analisar os resultados obtidos através dos outros dois processos expostos para consideração da não linearidade geométrica em função dos momentos fletores nas barras. Considerando-se como exata a solução encontrada através do Processo das Funções de Estabilidade, obtém-se os seguintes valores através do Processo P-Delta:

	Exemplo 2 (apoiado)		Exemplo 3 (engastado)	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-566(0.0%)	-470(0.0%)	-566(+0.2%)	-471(+0.2%)
7	518(-3.4%)	662(-1.5%)	525(-0.9%)	645(-0.3%)
6	1024(+17.4%)	2055(+0.1%)	1373(-0.4%)	1626(-0.1%)
5	0	17426(-13.9%)	4948(-0.1%)	4263(-1.6%)

Tabela 5.24 – Exemplo 2 e Exemplo 3 – Diferença nos momentos fletores obtidos através do Processo P-Delta e das funções de estabilidade

Comparando-se os resultados acima com os resultados expressos em termos do deslocamento no topo da estrutura (Tabela 5.6, para o Exemplo 2, e Tabela 5.13, para o Exemplo 3), observa-se a coincidência entre o erro obtido nos deslocamentos e o erro máximo obtido nos momentos fletores. De qualquer maneira, pode-se observar que a diferença entre os dois processos não é constante ao longo das barras, concentrando-se naquelas onde o efeito de 2ª ordem é mais significativo (nestes dois exemplos, as barras inferiores).

5.5.6 Processo K_G

Considerando-se novamente como exata a solução encontrada através do Processo das Funções de Estabilidade, obtém-se os seguintes valores através do processo K_G:

	Exemplo 2 (apoiado)		Exemplo 3 (engastado)	
Barra	Momento inicial (kgf.m)	Momento final (kgf.m)	Momento inicial (kgf.m)	Momento final (kgf.m)
8	-566(0.0%)	-470(0.0%)	-565(0.0%)	-470(0.0%)
7	536(-3.4%)	671(-0.1%)	530(0.0%)	648(+0.1%)
6	880(+0.9%)	2054(+0.1%)	1378(0.0%)	1628(0.0%)
5	0	19931(-1.6%)	4952(0.0%)	4332(-0.1%)

Tabela 5.25 – Exemplo 2 e Exemplo 3 – Diferença nos momentos fletores obtidos através do processo K_G e das funções de estabilidade

Comparando-se os resultados acima com os resultados expressos em termos do deslocamento no topo da estrutura (Tabela 5.6 para o Exemplo 2 e Tabela 5.13 para o Exemplo 3), observa-se novamente que a diferença entre os dois processos não é constante ao longo das barras, mas muito inferior ao erro apresentado pelo Processo P-Delta, podendo ser considerada desprezível nestes casos.

Os resultados obtidos através do processo K_{GM} basicamente reproduzem aqueles expressos na Tabela 5.25, exatamente como esperado.

Percebe-se, com o uso destes dois exemplos, que a adoção do deslocamento no topo como parâmetro de comparação entre os processos possui precisão aceitável para a comparação entre os esforços internos.

5.5.7 Outros esforços

A adoção de um coeficiente global como o γ_z para representar os efeitos de 2ª ordem em uma estrutura é algo que possui uma série de limitações. Neste caso, foi suposto inicialmente que o fator calculado deva majorar unicamente os momentos fletores ocorridos nas barras verticais (pilares). Isto por serem estes os elementos sujeitos à compressão e onde se concentram os efeitos de 2ª ordem. Na verdade, a conceituação do coeficiente γ_z obrigaria, em sua interpretação, que fossem majorados todos os esforços atuantes na estrutura, e não apenas os momentos fletores nos pilares. Da mesma forma, não é válida a hipótese contrária de se majorarem unicamente os momentos nos pilares, uma vez que a hiperestaticidade da estrutura faz com que esta variação se distribua pelos elementos, gerando variações em todos os outros esforços.

Por exemplo, sabe-se que os momentos fletores nas vigas, nos pontos de intersecção com os pilares, devem ser modificados para garantir o equilíbrio. Caso fosse realmente assumida uma distribuição constante para γ_z, o equilíbrio de momentos em cada nó modificaria os momentos fletores nas vigas.

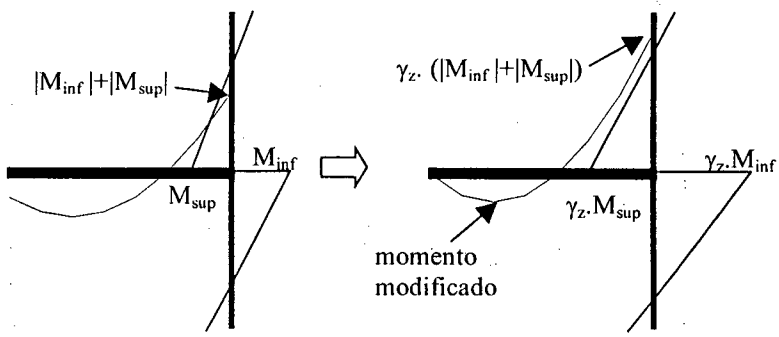


Figura 5.20 – Equilíbrio de momentos nos nós

O conceito de majorar os momentos nas vigas por γ_z seria válido apenas para os encontros com os pilares, sendo que os demais esforços deveriam ser corrigidos também por equilíbrio.

Outro efeito é o de modificar os esforços cortantes nas barras. No caso de barras sem carregamento distribuído (caso dos pilares), a majoração dos momentos por γ_z corresponde a uma majoração equivalente nos esforços cortantes.

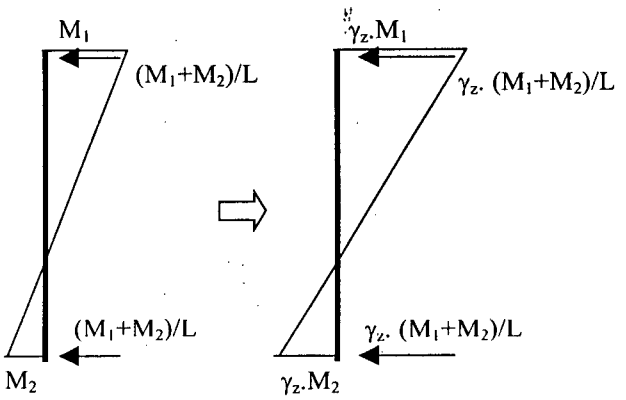


Figura 5.21 – Alteração nos esforços cortantes pelo efeito de 2ª ordem

Pode-se observar também, em diversos casos, uma variação na força normal aplicada às barras, variação esta devida ao efeito P-Δ da estrutura como um todo. Com a modificação da posição dos nós, modificam-se as cargas normais, mas de um valor não diretamente relacionado com a variação γ_z.

Para confirmar a variação nos esforços, pode-se tomar novamente os exemplos calculados e observar alguns pontos (vide Figura 5.15, contendo a numeração dos elementos).

No caso do Exemplo 3 (engastado), que apresenta $\gamma_z = 1.100$, tem-se os seguintes valores selecionados:

Esforço	Análise de 1ª ordem	Funções de estabilidade
Cortante na barra 5 (kgf)	1973.1	2025.4(+2.7%)
Normal na barra 5 (kgf)	41953.8	41467.0(-1.2%)
Normal na barra 9 (kgf)	52856.2	53343.0(+0.9%)
Fletor inicial na barra 1 (kgf.m)	-8552.6	-9365.8(+9.5%)
Fletor final na barra 1 (kgf.m)	-4827.5	-5711.0(+18.3%)

Tabela 5.26 – Exemplo 3 – Variação nos esforços internos

No caso do Exemplo 2 (apoiado), que apresenta $\gamma_z = 1.386$, tem-se os seguintes valores selecionados:

Esforço	Análise de 1ª ordem	Funções de estabilidade
Cortante na barra 5 (kgf)	2088.6	2929.8(+40.3%)
Normal na barra 5 (kgf)	40117.0	35387.3(-11.8%)
Normal na barra 9 (kgf)	54693.0	59422.7(+8.6%)
Fletor inicial na barra 1 (kgf.m)	-12907.4	-23579.8(+82.7%)
Fletor final na barra 1 (kgf.m)	-9389.8	-20896.1(+122.5%)

Tabela 5.27 – Exemplo 2 – Variação nos esforços internos

Pode-se notar, mesmo através de poucos pontos selecionados, que a variação ocorrida com os efeitos de 2ª ordem não é constante, podendo-se obter erros consideráveis na adoção de um fator de multiplicação único como γ_z. Outro ponto importante refere-se ao aumento do esforço normal atuante na barra 9. Esta variação, embora inferior à dos momentos fletores e ao próprio coeficiente γ_z, não pode ser desprezada, visto reduzir a capacidade portante do pilar em questão, comprovando a teoria de que não é possível majorar unicamente os momentos fletores nos pilares.

5.6 EXEMPLOS ADICIONAIS

Para finalizar este capítulo, pretende-se expor outros exemplos numéricos, de forma a verificar a validade das conclusões efetuadas no decorrer deste capítulo.

Nos exemplos a seguir, não se incluem, os resultados obtidos através do Processo da Matriz de Rigidez Geométrica Modificado (K_{GM}). Isto para que não se torne repetitivo, uma vez que mostrou-se que estes resultados simplesmente tendem àqueles obtidos através do Processo da Matriz de Rigidez Geométrica (K_G), conforme aumenta o número de iterações na solução.

5.6.1 Pórtico apoiado na base com viga de travamento no lance inferior

Apenas a título de ilustração, reproduzem-se aqui algumas das considerações apresentadas por PITTA (1996), por se julgar que sejam importantes para a aplicação dos conceitos envolvidos. Até o momento, foram utilizados dois exemplos numéricos, os quais diferem entre si apenas na condição de vinculação da sua fundação. Isto foi feito para se ilustrar a diferença entre uma estrutura relativamente esbelta e outra mais rígida.

Em termos de aplicação para Projeto, é importante destacar que, na prática, não é necessário passar às fundações a responsabilidade de estabilizar a estrutura. Isto pode ser considerado indesejável, visto que uma condição de engastamento no solo é algo complexo e de difícil efetividade. Ao invés disto, normalmente utilizam-se vigas de travamento no nível inferior (vigas de baldrame), ficando estas responsáveis pela rigidez do lance inferior. Reproduzindo o exemplo já apresentado por PITTA (1996), apresenta-se o esquema estrutural de um edifício simples, sujeito a esforços verticais e horizontais:

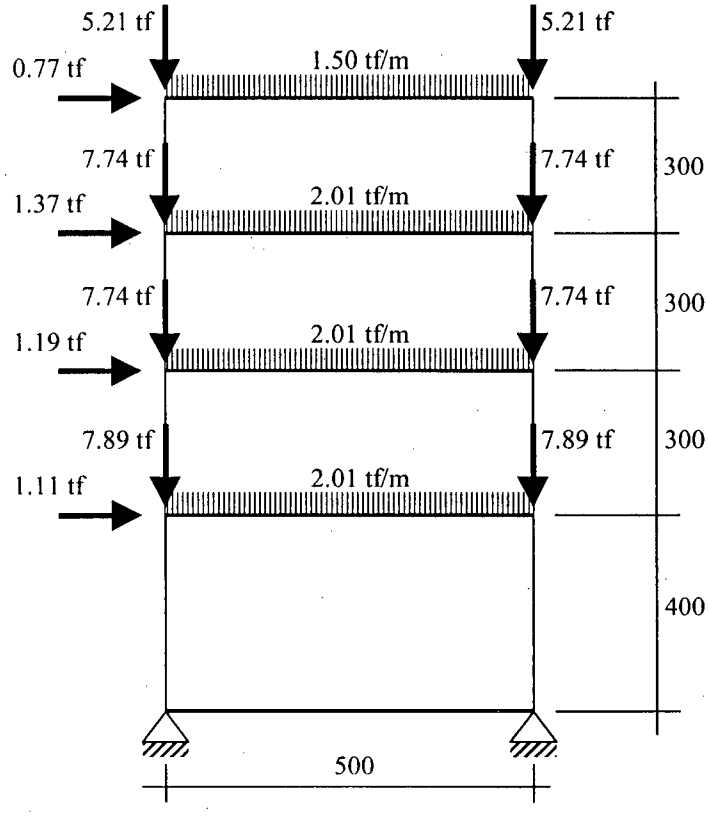


Figura 5.22 – Exemplo 4 – Pórtico apoiado na base com viga de travamento

A única diferença deste modelo para o Exemplo 2 é a viga de travamento, na qual foi utilizada exatamente a mesma seção das demais (13x55 cm). Os resultados para este modelo apresentam-se como seguinte:

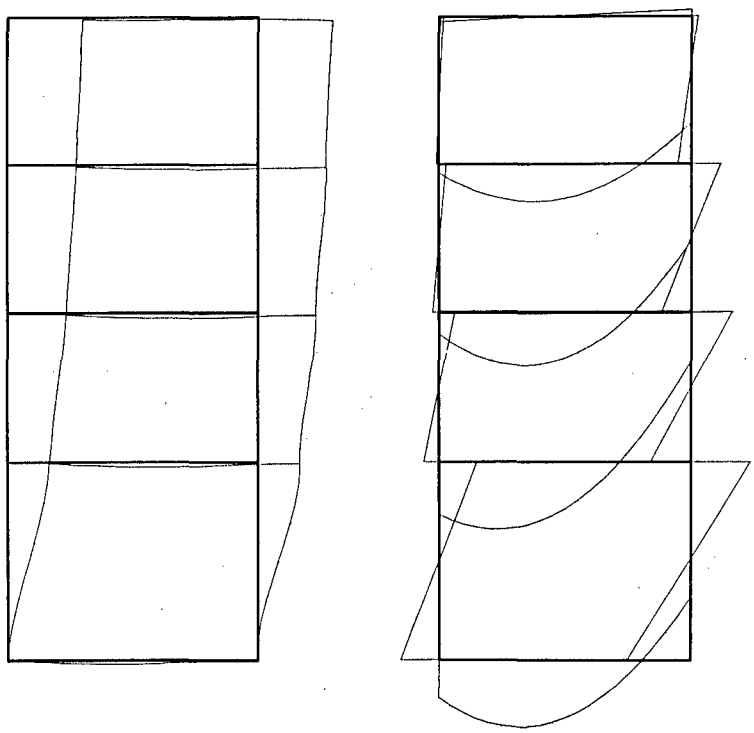


Figura 5.23 – Exemplo 4 – Estrutura deformada e diagrama de momentos fletores

Utilizando-se novamente o deslocamento horizontal no topo da estrutura para comparar os três processos, obtém-se os seguintes valores:

Processo	Deslocamento no topo (cm)	Diferença (%)
1ª ordem	4.576	-10.41
P-Delta	5.035	-1.43
K_G	5.108	0.00
Funções de estabilidade	5.108	

Tabela 5.28 – Exemplo 4 – Comparação entre os processos

Comparando estes resultados com aqueles obtidos no Exemplo 2 e no Exemplo 3, nota-se a grande eficiência da viga de travamento no nível inferior. Mesmo com as fundações rotuladas, esta estrutura apresentou um comportamento bastante próximo àquele obtido com as fundações engastadas.

Para a estrutura apresentada, obtiveram-se os seguintes valores referentes aos parâmetros de instabilidade:

- $\alpha = 0.669$
- $\gamma_z = 1.091$

Observa-se, mais uma vez, que estes valores aproximam-se daqueles obtidos considerando-se fundações engastadas. Nota-se, também, o razoável grau de precisão apresentado pelo coeficiente γ_z, caso este fosse utilizado para verificar o deslocamento no topo da edificação.

A fim de avaliar a variação nos esforços internos das barras, vai-se explicitar uma numeração para os elementos da estrutura:

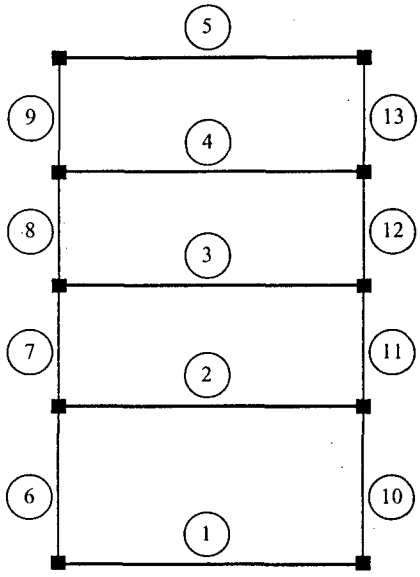


Figura 5.24 – Exemplo 4 – Numeração das barras

O gráfico a seguir exprime a variação percentual ocorrida nos esforços internos atuantes em todas as barras da estrutura.

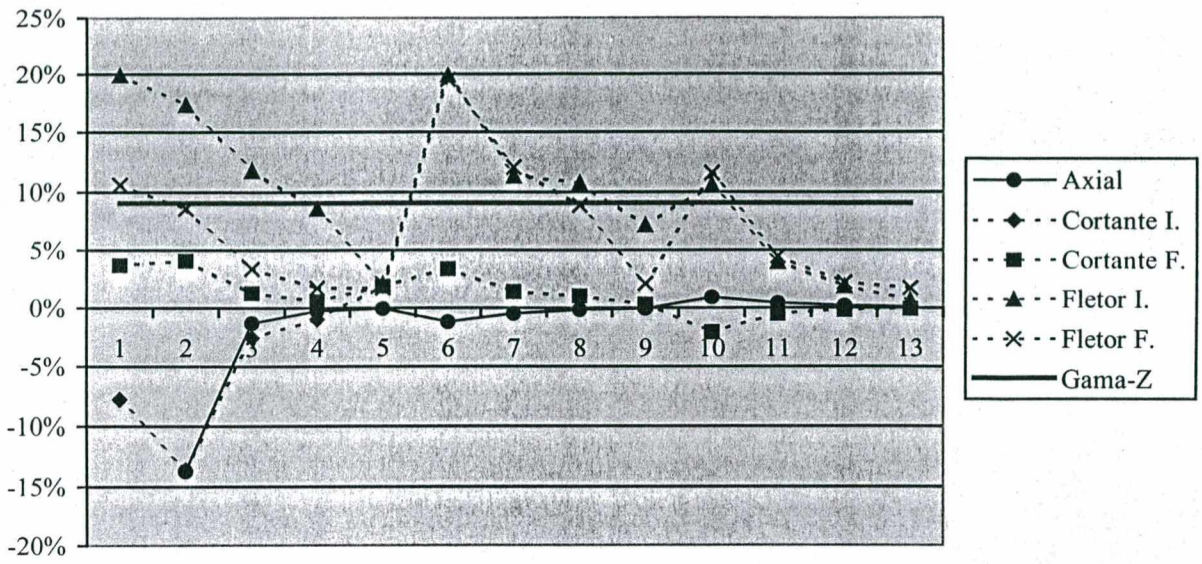


Figura 5.25 – Exemplo 4 – Variação nos esforços internos

Para analisar este diagrama, deve-se notar que, conforme colocado na Figura 5.24, os pontos numerados de 1 a 5 referem-se às vigas e os pontos 6 a 13 referem-se aos pilares. Pode-se fazer alguns comentários:

- Os efeitos de 2ª ordem causam variação em todos os esforços internos atuantes em todas as barras.
- Esta variação não é uniforme e não pode ser relacionada diretamente com o coeficiente γ_z.
- As maiores diferenças são obtidas nos momentos fletores. Apenas em torno destes pode-se identificar uma "média" que poderia ser relacionada com γ_z.
- Os efeitos de 2ª ordem são gradativamente menores em direção ao topo da estrutura, tanto para as vigas como para os pilares.
- Nota-se um sensível erro contra a segurança que seria cometido ao adotar para a barra 6 esforços de 2ª ordem através da simples majoração dos esforços de 1ª ordem por γ_z.

5.6.2 Pórtico de 15 pavimentos engastado na base

A partir dos próximos exemplos, deseja-se verificar as conclusões efetuadas até o momento utilizando modelos um pouco maiores. Inicialmente, vai-se tomar uma estrutura semelhante ao Exemplo 3 (pórtico engastado na base), mas agora utilizando-se uma edificação de 15 pavimentos no lugar dos 4 pavimentos do modelo original.

A fim de simular uma situação real de projeto, vai-se adotar seções mais robustas para os pilares, sendo estas variáveis ao longo da altura da edificação, porém constantes a cada três lances, prática usual em termos de projeto. Supõe-se também um módulo de elasticidade do concreto maior, simulando a utilização de concreto de maior resistência característica.

Serão utilizadas as seguintes propriedades:

- Módulo de elasticidade: $3{,}5 \times 10^5$ kgf/cm^2
- Seção transversal das barras: 15 x 55 cm para as barras horizontais (vigas) e variando desde 50 x 25 cm até 40 x 15 cm para as barras verticais (pilares).

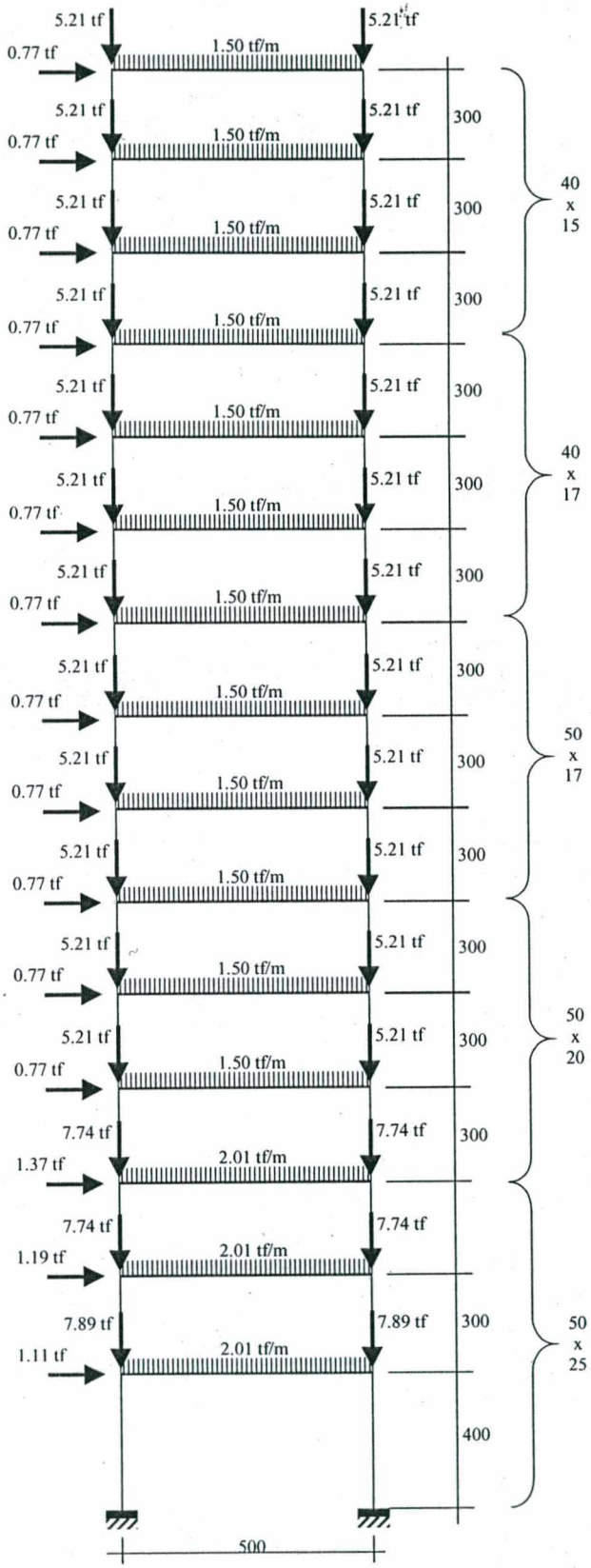


Figura 5.26 – Exemplo 5 – Pórtico de 15 pavimentos engastado na base

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 33.992 cm
- Parâmetro Alfa: $\alpha = 0.992$
- Coeficiente Gama-Z: $\gamma_z = 1.171$

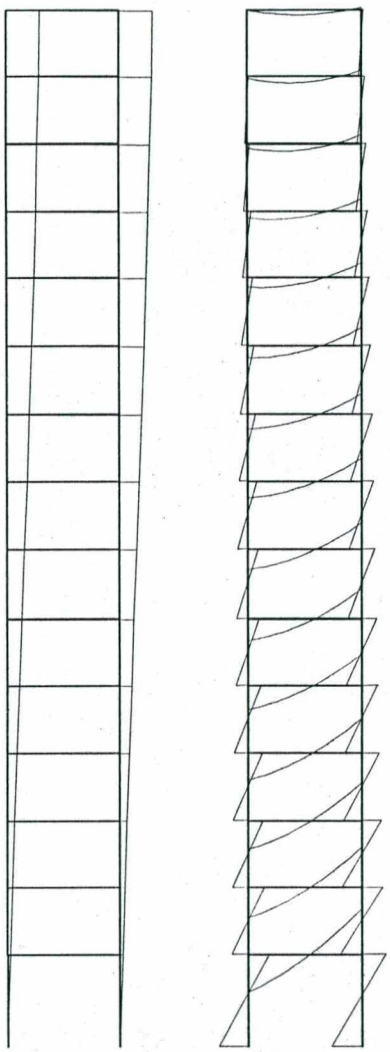


Figura 5.27 – Exemplo 5 – Estrutura deformada e diagrama de momentos fletores

Deseja-se analisar este exemplo utilizando-se os três processos apresentados (P-Delta, K_G e Funções de Estabilidade). Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:

Nº de subdivisões	Funções de estabilidade	P-Delta	K_G
1	40.326	39.780	40.320
2	40.326	39.791	40.323
5	40.326	40.215	40.326
10	40.326	40.292	40.326
20	40.326	40.312	40.326

Tabela 5.29 – Exemplo 5 – Variação no deslocamento no topo conforme a discretização

Pode-se notar, a partir deste exemplo, que as conclusões já estabelecidas para o Exemplo 2 e para o Exemplo 3 reproduzem-se aqui, mesmo utilizando-se uma estrutura significativamente maior:

- Os resultados obtidos através do Processo das Funções de Estabilidade independem da discretização, caracterizando-se como "exatos" mesmo sem subdivisões;
- O Processo P-Delta converge para o resultado exato conforme a subdivisão, mas sem alcançá-lo realmente, apresentando um erro inicial de 1.35%;
- O processo K_G converge rapidamente para o resultado correto com um número de subdivisões bem inferior ao P-Delta, apresentando resultados muito próximos mesmo sem qualquer discretização (diferença de 0.01%).

Destacam-se como contribuições mais importantes deste modelo a evidência de que a influência da discretização e a diferença apresentada entre os resultados dos diferentes processos é função da flexibilidade da própria estrutura e não da quantidade de barras. Comparando este modelo com um modelo menor equivalente, o do Exemplo 3, nota-se exatamente o mesmo comportamento. A existência de um número maior de barras no modelo não significa que se possa utilizar uma discretização mais grosseira.

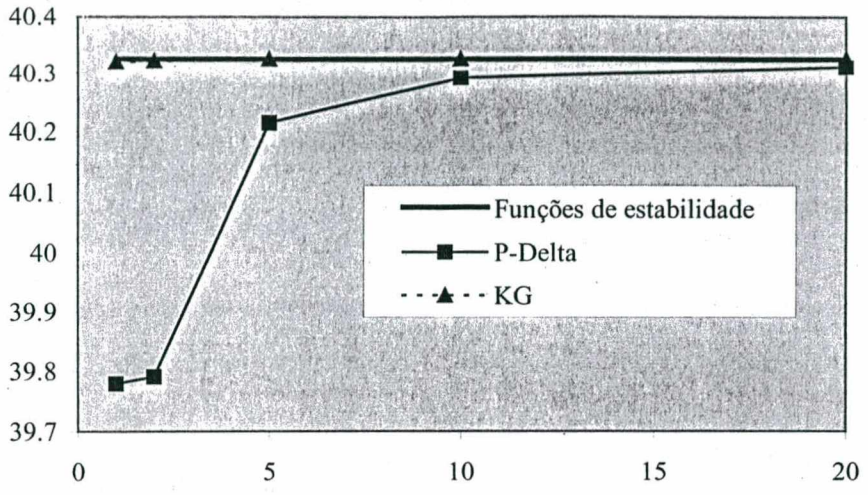


Figura 5.28 – Exemplo 5 – Variação no deslocamento no topo conforme a discretização

Mais uma vez, observa-se o razoável grau de precisão apresentado pelo coeficiente γ_z caso este fosse utilizado para verificar o deslocamento no topo da edificação. Multiplicando-se o deslocamento de 1ª ordem por γ_z, seria obtido um valor de 39.805 cm, mais próximo ao resultado "exato" que o próprio P-Delta.

Todavia, sua definição não é de predizer o deslocamento, mas sim a variação nos esforços internos da estrutura. Deseja-se mostrar a real variação destes esforços, comparando os valores obtidos através da análise de 1ª ordem com os valores obtidos através do Processo das Funções de Estabilidade.

O diagrama a seguir apresenta as variações ocorridas apenas nos pilares, agrupados por seção.

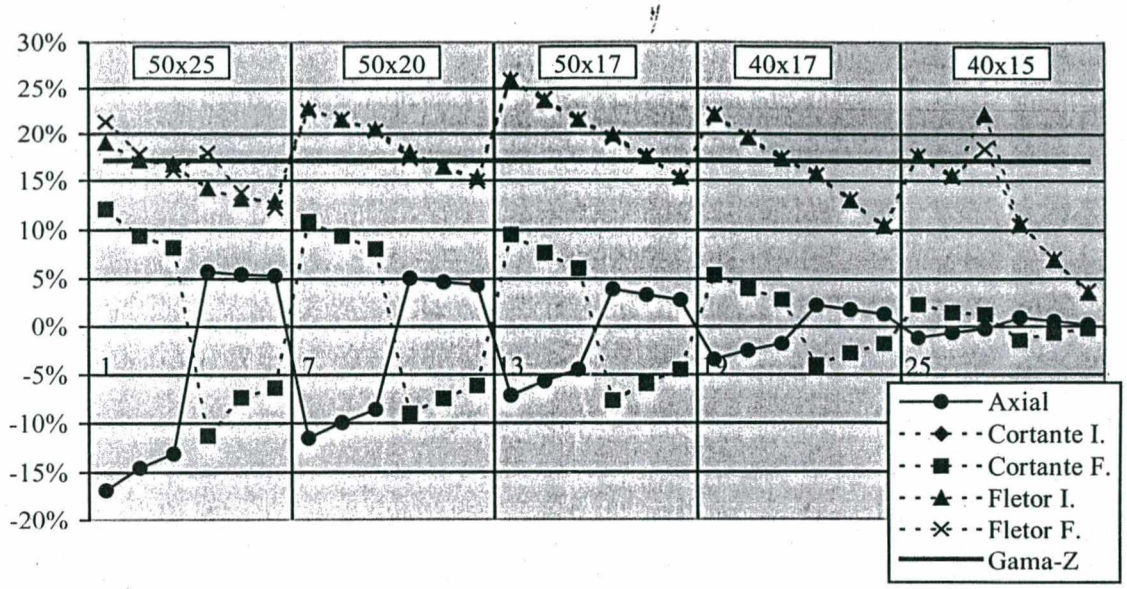


Figura 5.29 – Exemplo 5 – Variação nos esforços internos dos pilares

Pode-se fazer alguns comentários:

- Os efeitos de 2ª ordem causam variação em todos os esforços internos atuantes em todas as barras.
- Esta variação não é uniforme e não pode ser relacionada diretamente com o coeficiente γ_z, mas pode-se identificar em torno dos momentos fletores uma "média" que poderia ser relacionada com γ_z.
- A variação na carga axial é bem inferior à dos momentos fletores, mas apresenta-se relevante na base da edificação. Deve-se notar também que tanto a carga axial quanto os esforços cortantes tendem a ser reduzidos de um lado da estrutura e aumentados no outro.

Apresentando-se um diagrama análogo, desta vez englobando apenas as vigas, onde apresenta-se a numeração das barras iniciando do lance inferior e aumentando em direção ao lance superior, obteria-se:

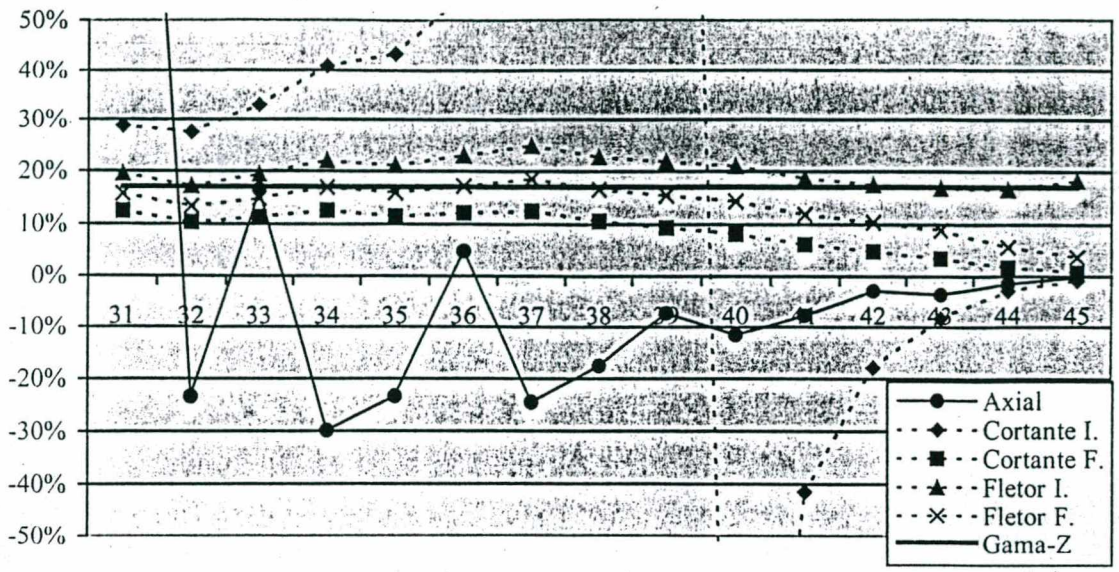


Figura 5.30 – Exemplo 5 – Variação nos esforços internos das vigas

Observa-se que:

- É bastante difícil identificar um comportamento para as vigas relacionado com o valor do coeficiente γ_z. Nota-se, porém, que a alteração nos esforços é significativa.
- Quanto maiores são os valores dos esforços internos atuantes nas vigas, em comparação com os esforços atuantes nos pilares, mais se pode identificar um comportamento médio. Quando os esforços são pequenos, como é o caso das cargas axiais, a variação percentual pode ser muito grande.

5.6.3 Pórtico de 15 pavimentos apoiado na base

Neste exemplo, objetiva-se verificar a influência do engastamento nas fundações na mesma estrutura de 15 pavimentos utilizada para o Exemplo 5. Supõe-se a mesma estrutura colocada na Figura 5.31, mas com as fundações rotuladas. Todos os demais dados são supostos iguais.

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 43.367 cm
- Parâmetro Alfa: $\alpha = 1.120$
- Coeficiente Gama-Z: $\gamma_z = 1.273$

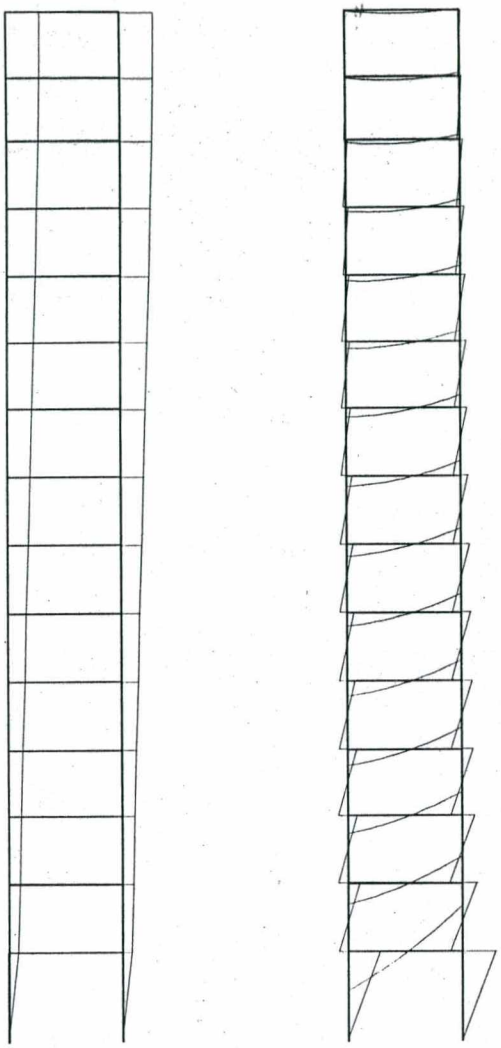
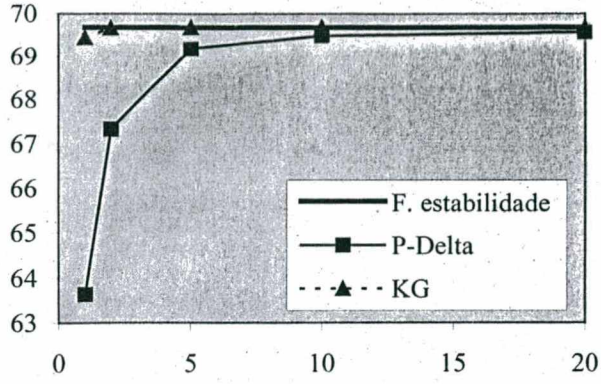


Figura 5.31 – Exemplo 6 – Estrutura deformada e diagrama de momentos fletores

Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:



Funções de estabilidade:		69.692
Nº de subdivisões	P-Delta	K_G
1	63.634	69.440
2	67.329	69.672
5	69.190	69.691
10	69.491	69.692
20	69.568	69.692

Figura 5.32 – Exemplo 6 – Variação no deslocamento no topo conforme a discretização

Nota-se, como esperado, que a diferença entre este modelo e o anterior, onde as fundações estavam engastadas, apenas reproduz exatamente as diferenças já constatadas para o Exemplo 2 e o Exemplo 3. O erro apresentado pelo Processo P-Delta, no caso de não se fazer discretização, passa para 8.7%, um valor relevante.

Destaca-se que, da mesma forma como ocorrido para o Exemplo 2, no item 5.1.7, o valor apresentado pelo coeficiente γ_z ficou abaixo da real variação no deslocamento no topo da edificação, considerado o item no qual sua aplicação apresenta maior precisão. Em ambos os casos, a ocorrência de um lance com rigidez muito diferente dos demais fez com que os resultados divergissem mais dos reais. Esta estrutura contendo um número maior de pavimentos apresentou um erro na determinação do γ_z muito superior àquela com número menor de pavimentos. Neste exemplo, o deslocamento no topo determinado através do coeficiente Gama-Z seria de 5.206 cm, valor 20.8% inferior ao obtido através do Processo das Funções de Estabilidade.

O diagrama a seguir apresenta as variações nos esforços internos ocorridas apenas nos pilares, agrupados por seção.

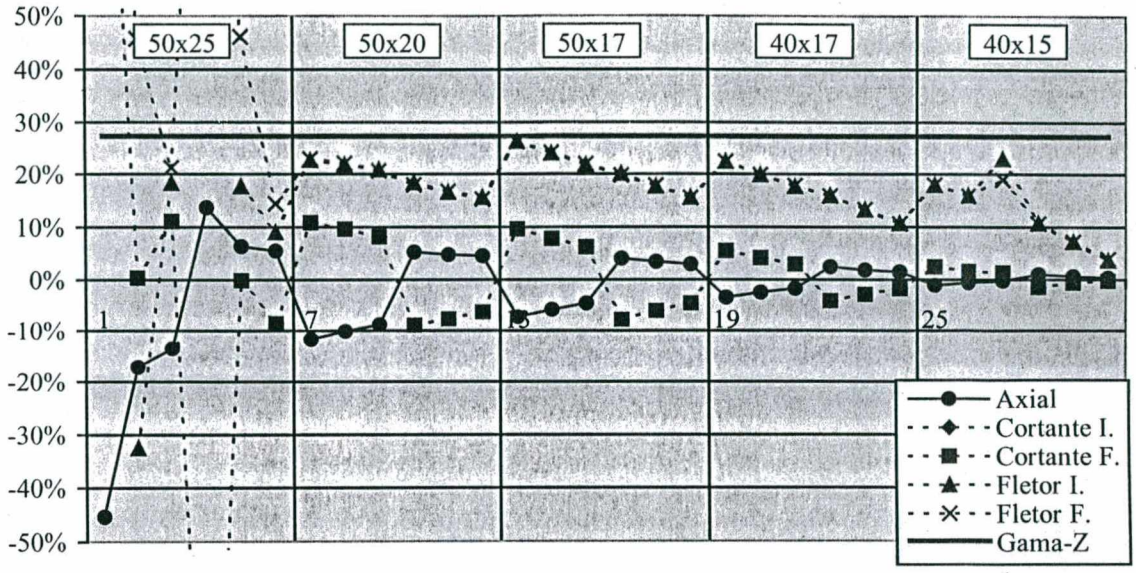


Figura 5.33 – Exemplo 6 – Variação nos esforços internos dos pilares

Nota-se, neste exemplo, que a variação concentra-se muito no primeiro lance, ficando os demais lances com variações abaixo do próprio γ_z. Obtém-se, na situação mais crítica, uma diferença nos momentos fletores na ordem de 127%, em muito superior ao que se obteria com o auxílio de γ_z.

5.6.4 Pórtico de 15 pavimentos e três linhas de pilares

Neste exemplo, objetiva-se verificar a influência do aumento no número de traves verticais na mesma estrutura de 15 pavimentos utilizada para o Exemplo 5. Supõe-se a mesma estrutura colocada na Figura 5.31, mas com uma linha de pilares a mais, mantendo-se as fundações engastadas. Todos os demais dados são supostos iguais.

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 20.899 cm
- Parâmetro Alfa: $\alpha = 1.012$
- Coeficiente Gama-Z: $\gamma_z = 1.185$

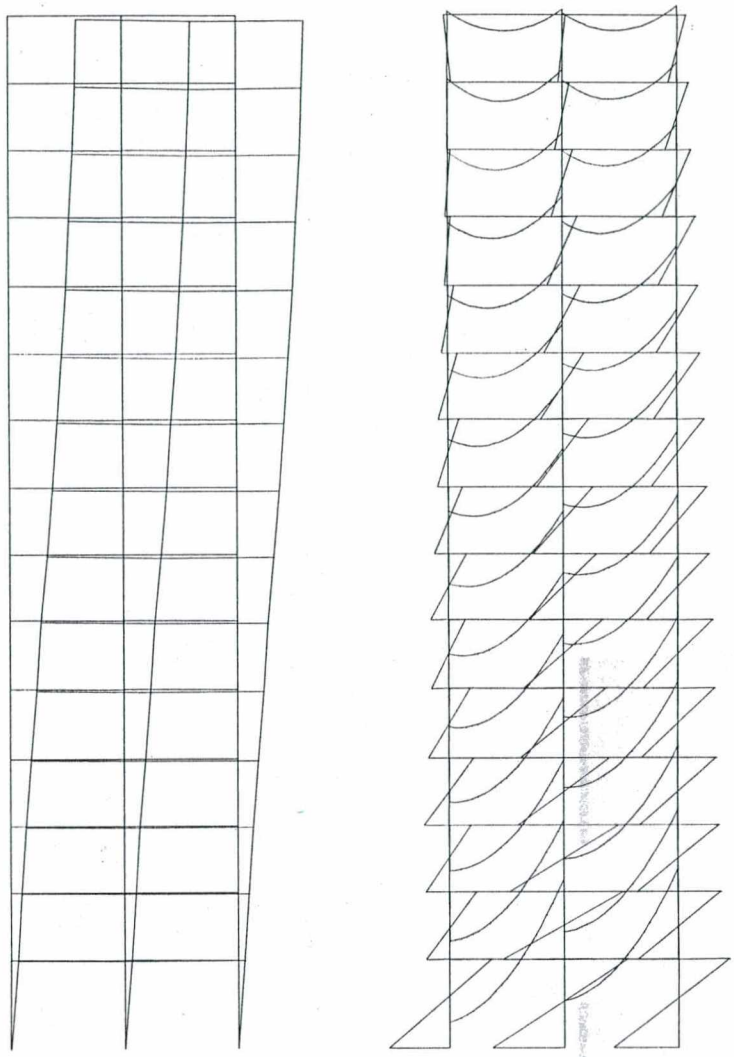
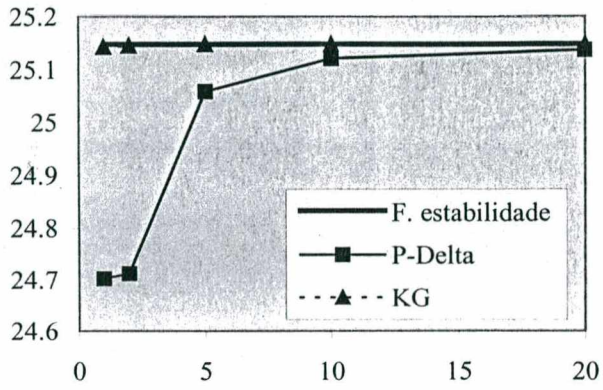


Figura 5.34 – Exemplo 7 – Estrutura deformada e diagrama de momentos fletores

Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:



Funções de estabilidade:		25.148
Nº de subdivisões	P-Delta	K_G
1	24.703	25.143
2	24.712	25.145
5	25.058	25.148
10	25.121	25.148
20	25.137	25.148

Figura 5.35 – Exemplo 7 – Variação no deslocamento no topo conforme a discretização

Neste exemplo, nota-se que o fato de se tornar a estrutura mais robusta, embora tenha reduzido o deslocamento horizontal da edificação, não modificou em nada a resposta da estrutura aos efeitos de 2ª ordem. Deve-se atentar, todavia, para o fato de que a carga vertical total aplicada à estrutura é duas vezes maior do que a do Exemplo 5, o que compensou o aumento de rigidez.

Mais uma vez, torna-se interessante analisar as variações nos esforços internos ocorridos nos pilares, através do diagrama a seguir.

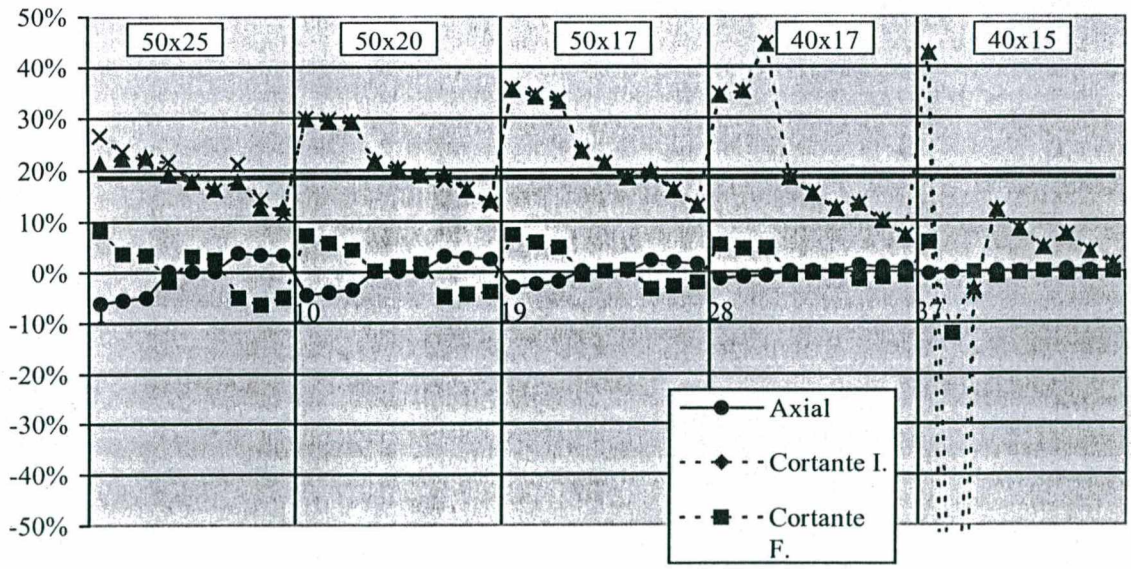


Figura 5.36 – Exemplo 7 – Variação nos esforços internos dos pilares

Nota-se, neste exemplo:

- O comportamento médio desta estrutura é igual à do Exemplo 5.
- Nos pilares dos lances superiores, por apresentarem momentos fletores muito pequenos em comparação com os demais (vide Figura 5.34), ocorre uma variação percentual muito grande.

5.6.5 Pórtico de 15 pavimentos variando ao longo da altura

Até o presente momento, foram utilizados exemplos onde, embora as características de rigidez fossem diferentes ao longo de sua altura, não existia uma variação na geometria ou disposição dos elementos. No exemplo a seguir, busca-se verificar a validade das conclusões já efetuadas em uma estrutura cuja largura é diferente em função da altura.

Utiliza-se a mesma estrutura do Exemplo 7, mas partindo-se de quatro linhas de pilares na parte inferior, reduzindo-se para três e, por fim, duas linhas de pilares. Mantém-se as seções transversais e carregamentos do Exemplo 7.

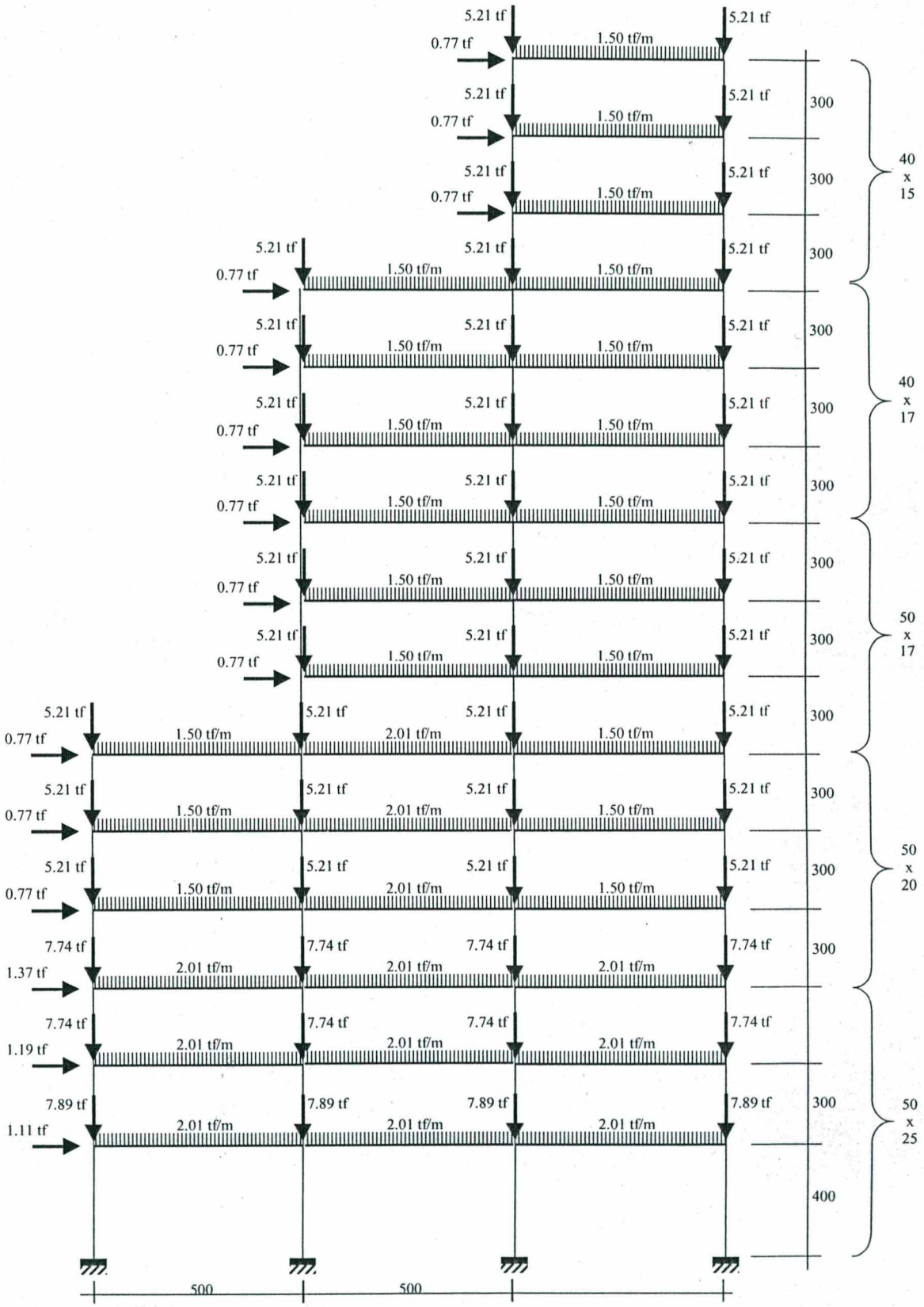


Figura 5.37 – Exemplo 8 – Pórtico de 15 pavimentos com largura variável

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 2.335 cm
- Parâmetro Alfa: $\alpha = 0.968$
- Coeficiente Gama-Z: $\gamma_z = 1.097$

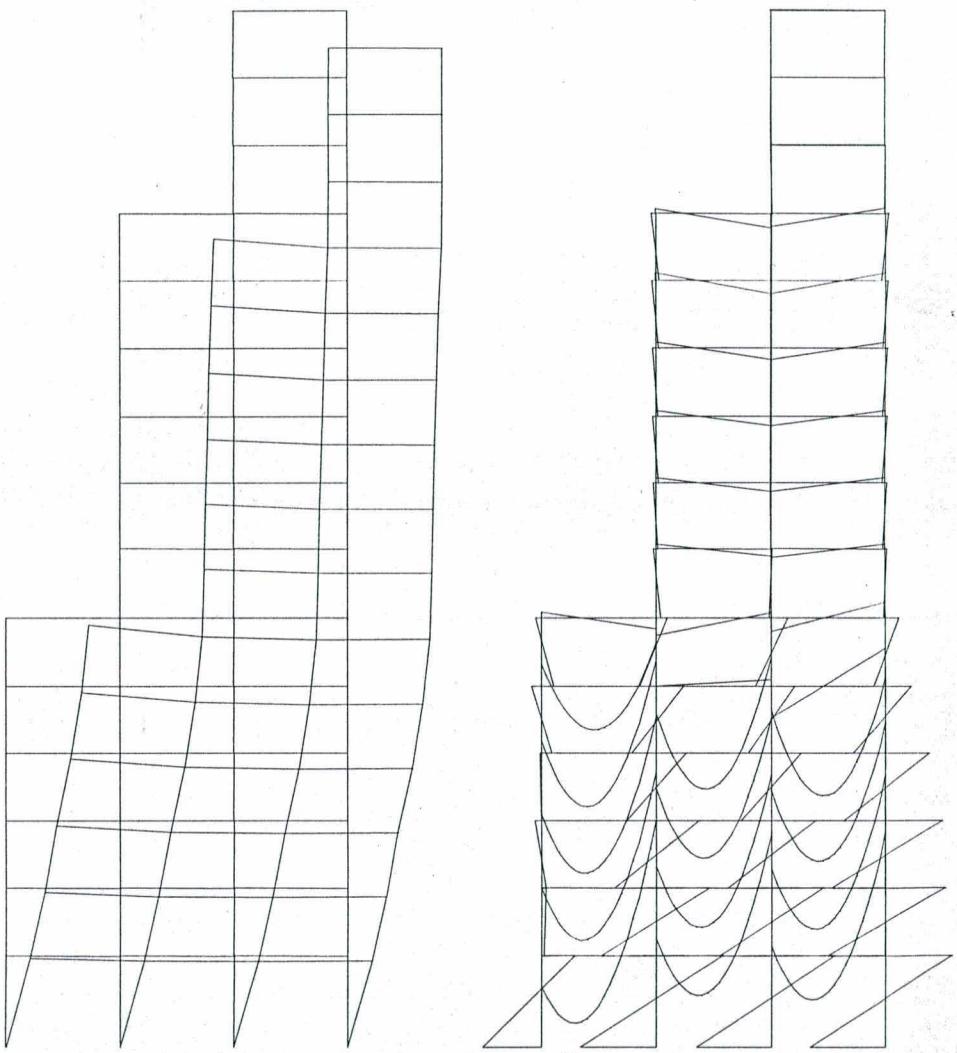
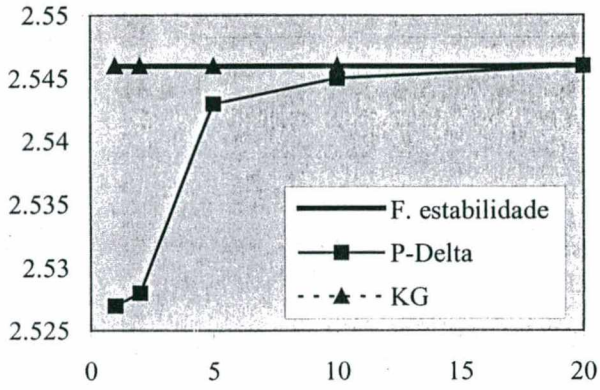


Figura 5.38 – Exemplo 8 – Estrutura deformada e diagrama de momentos fletores

Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:



Funções de estabilidade:		2.546
Nº de subdivisões	P-Delta	K_G
1	2.527	2.546
2	2.528	2.546
5	2.543	2.546
10	2.545	2.546
20	2.546	2.546

Figura 5.39 – Exemplo 8 – Variação no deslocamento no topo conforme a discretização

Nota-se um comportamento idêntico aos demais modelos. A resposta desta estrutura com relação aos efeitos de 2ª ordem é determinada quase que exclusivamente por sua parte inferior. Por haverem mais linhas de pilares, a estrutura é mais robusta e os efeitos de 2ª ordem são pequenos. Além disto, nota-se, mais uma vez, que as diferenças encontradas entre o Processo P-Delta e o Processo das Funções de Estabilidade só se mostra relevante quando os próprios efeitos de 2ª ordem são grandes, independentemente da geometria da estrutura.

Novamente, torna-se interessante analisar as variações nos esforços internos ocorridos nos pilares, através do diagrama a seguir.

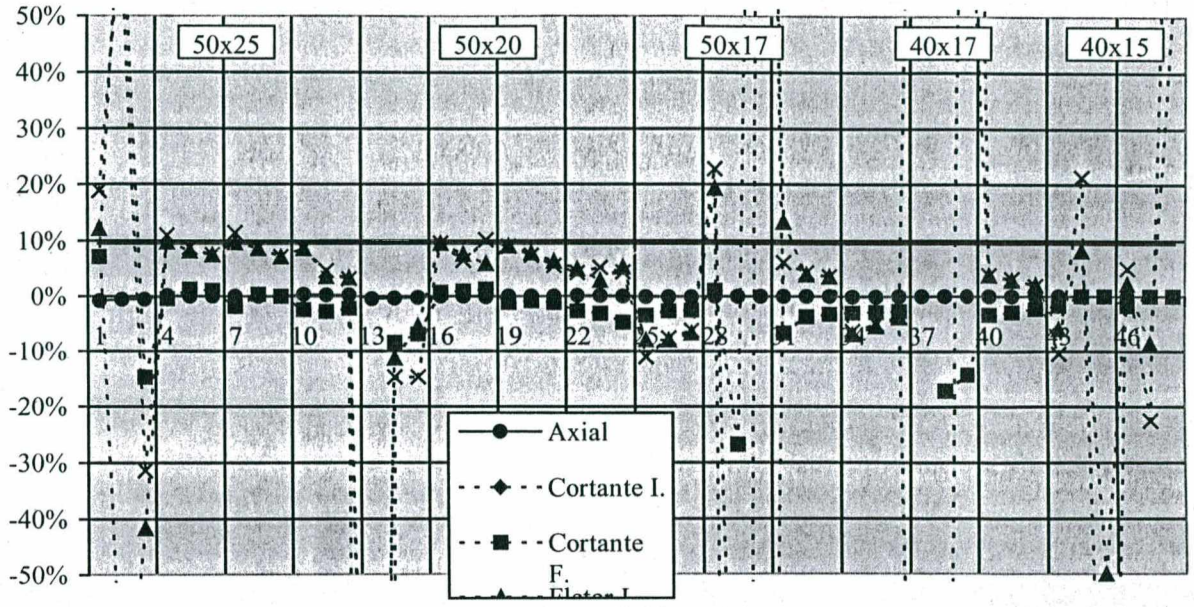


Figura 5.40 – Exemplo 8 – Variação nos esforços internos dos pilares

Nota-se, neste exemplo:

- O comportamento médio desta estrutura é semelhante aos demais exemplos.
- Não apenas nos pilares dos lances superiores, mas também em alguns pilares inferiores, novamente por apresentarem momentos fletores muito pequenos em comparação com os demais (vide Figura 5.38), ocorre uma variação percentual muito grande.

5.6.6 Pórtico com viga de transição na parte inferior

Uma situação bastante comum para fins de projeto é a ocorrência de vigas de transição, usualmente dispostas nos pavimentos inferiores, sobre as quais apoiam-se pilares provenientes do "corpo" da estrutura. Objetiva-se, com este exemplo, verificar a validade das conclusões propostas para um exemplo desta natureza.

Parte-se da mesma estrutura utilizada para o Exemplo 7, mas substituindo-se os dois lances inferiores por vigas de transição. Nestes, foram adotadas seções transversais de 50 x 50 cm para os pilares e 20 x 70 cm para as vigas. Mantém-se as demais seções transversais e carregamentos conforme o Exemplo 7.

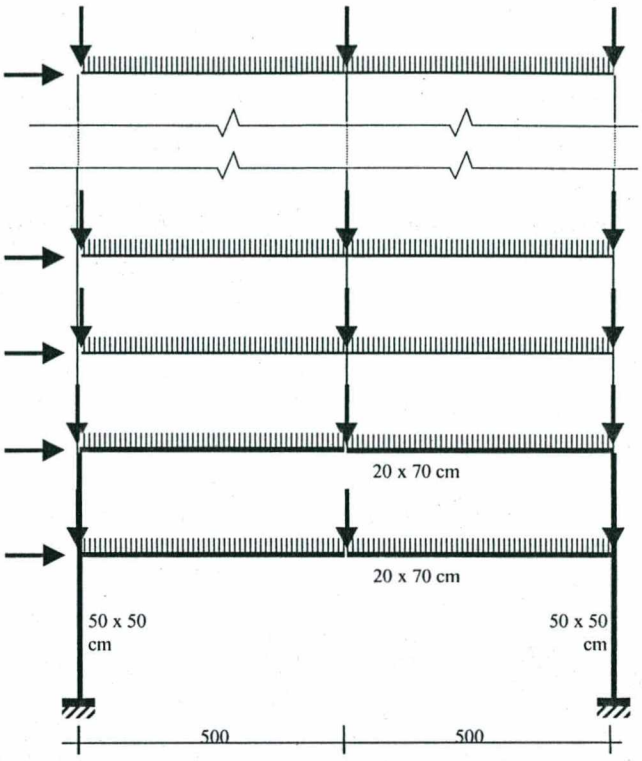


Figura 5.41 – Exemplo 9 – Pórtico de 15 pavimentos com viga de transição

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 18.498 cm
- Parâmetro Alfa: $\alpha = 0.952$
- Coeficiente Gama-Z: $\gamma_z = 1.148$

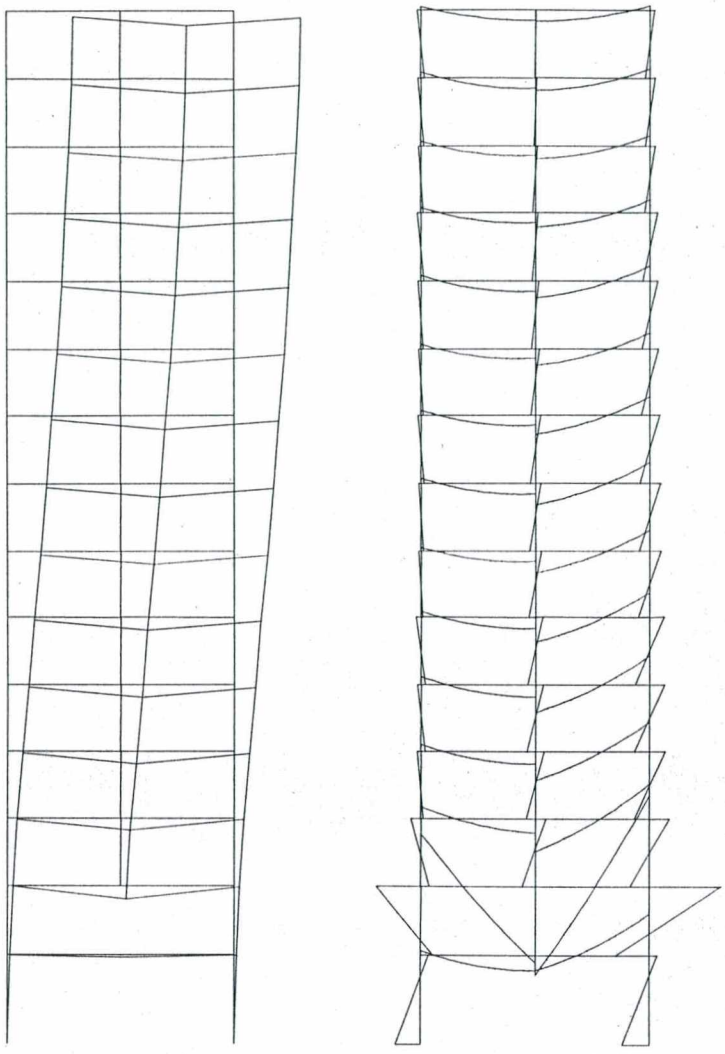
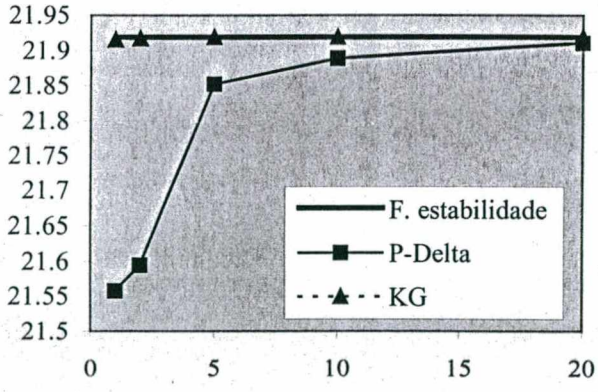


Figura 5.42 – Exemplo 9 – Estrutura deformada e diagrama de momentos fletores

Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:



Funções de estabilidade:		21.919
Nº de subdivisões	P-Delta	K_G
1	21.558	21.915
2	21.594	21.917
5	21.852	21.919
10	21.889	21.919
20	21.911	21.919

Figura 5.43 – Exemplo 8 – Variação no deslocamento no topo conforme a discretização

Em termos do deslocamento no topo da edificação, não existe qualquer diferença no comportamento deste para os modelos anteriores. Não se observou erro apreciável na utilização do Processo P-Delta, sendo os resultados obtidos através de K_G praticamente iguais aos que se obtiveram através do Processo das Funções de Estabilidade. A variação dos esforços internos nos pilares, em comparação com o coeficiente γ_z, manteve também o mesmo comportamento já observado nos exemplos anteriores.

Nesta estrutura em particular, torna-se interessante observar a influência dos efeitos de 2ª ordem nos esforços internos atuantes na viga de transição.

Esforço	Barra 46		Barra 47	
	1ª ordem	1ª + 2ª ordem	1ª ordem	1ª + 2ª ordem
Axial (tf)	16.5	16.3	25.8	26.6
Cortante inicial (tf)	35.2	34.6	-37.3	-37.9
Cortante final (tf)	25.2	24.6	-47.3	-47.9
Fletor inicial (tf.m)	-62.1	-60.2	104.0	105.6
Fletor final (tf.m)	88.7	88.3	-107.4	-110.0

Tabela 5.30 – Exemplo 4 – Comparação entre os processos

Nota-se uma variação desprezível, com aumento de apenas 1.5% no momento positivo obtido no meio do vão. Isto evidencia o fato de que não é interessante majorar os esforços da estrutura, obtidos através da análise de 1ª ordem, diretamente por γ_z.

5.6.7 Pórtico de 15 pavimentos contendo um pilar parede

Outra questão de interesse, relativa à aplicação dos procedimentos descritos no projeto de estruturas de concreto armado, diz respeito à presença de pilares parede no conjunto da edificação. Estes pilares de grandes dimensões situam-se, normalmente, na região da caixa de elevador do edifício ou de suas escadas.

A abordagem tradicional dada a este tipo de estrutura parte da divisão desta em duas partes: a parte de contraventamento e a parte contraventada. FUSCO (1981), por exemplo, sugere "fazer com que a estrutura de contraventamento, composta por dois ou mais elementos de contraventamento e pelas lajes do edifício, tenha rigidez suficiente para que os demais pilares possam ser considerados como participantes de uma estrutura com nós indeslocáveis". Segundo o autor, os elementos de contraventamento são compostos por pilares de grandes dimensões, por paredes estruturais, por treliças ou pórticos de grande rigidez. Esta divisão, comum em grande parte da bibliografia disponível, visa permitir o dimensionamento dos elementos supostos pertencentes à parte contraventada da estrutura sem a consideração dos efeitos globais de 2ª ordem.

O conceito de estrutura de contraventamento é algo presente mesmo em trabalhos recentes. PENNER & FUSCO (1997) afirmam que: "Em uma análise de estabilidade, considerar a participação de todos os pilares como integrantes da estrutura de contraventamento torna o cálculo muito trabalhoso". A abordagem dada pelos autores é a de verificar inicialmente a estabilidade global da estrutura considerando-se apenas a parte de contraventamento, efetuando a análise da estrutura como um todo apenas quando se fizer necessário. SANTOS & FRANCO (1993) chamam a atenção para o fato de que "todos os elementos resistem, alguns mais, outros menos, aos esforços decorrentes das ações horizontais". Mesmo assim, permitem identificar, "por conveniência de análise", as subestruturas de contraventamento.

Conforme colocado por SANTOS & FRANCO (1993), a nova NBR 6118 permitirá definir como "elementos isolados" aqueles elementos contraventados, sejam estes pertencentes a estruturas de nós fixos (onde os efeitos de 2ª ordem podem ser desprezados) ou mesmo de nós móveis (onde os efeitos de 2ª ordem são relevantes), aplicando às suas extremidades unicamente os esforços obtidos através da análise de 1ª ordem.

Objetiva-se, com este exemplo, analisar a real influência de um pilar de grande rigidez pertencente ao pórtico equivalente de uma edificação. Parte-se da mesma estrutura utilizada para o Exemplo 7, acrescentando-se a esta um pilar com seção transversal de 20 x 100 cm, constante ao longo da altura da edificação. Mantém-se as demais seções transversais e carregamentos conforme o Exemplo 7.

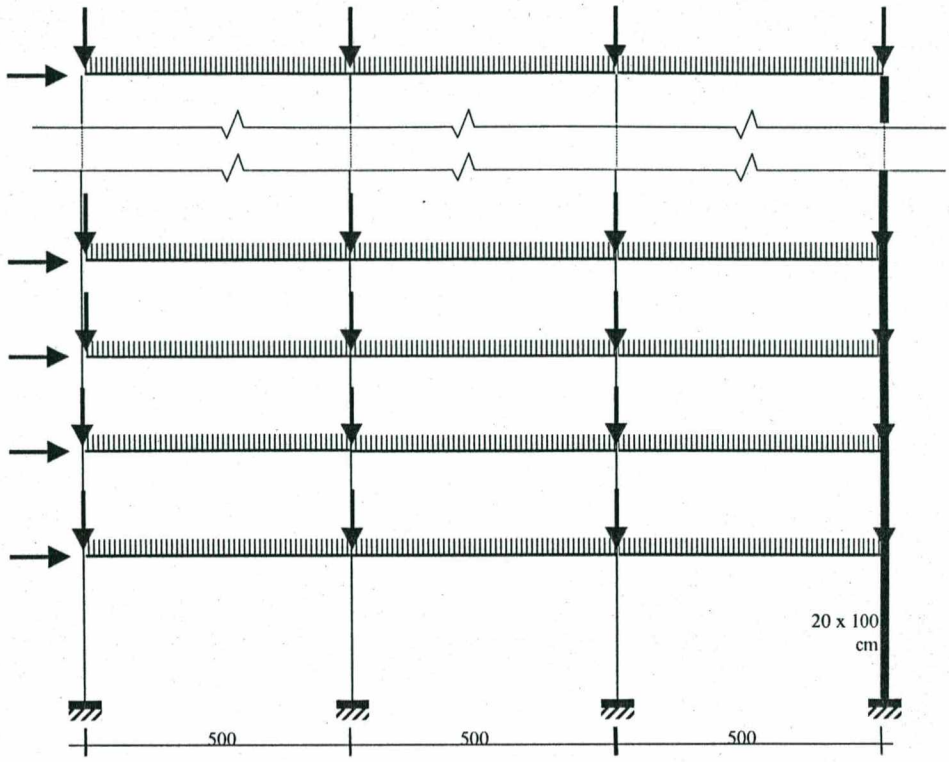


Figura 5.44 – Exemplo 10 – Pórtico de 15 pavimentos contendo um pilar parede

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 7.355 cm
- Parâmetro Alfa: $\alpha = 0.719$
- Coeficiente Gama-Z: $\gamma_z = 1.098$

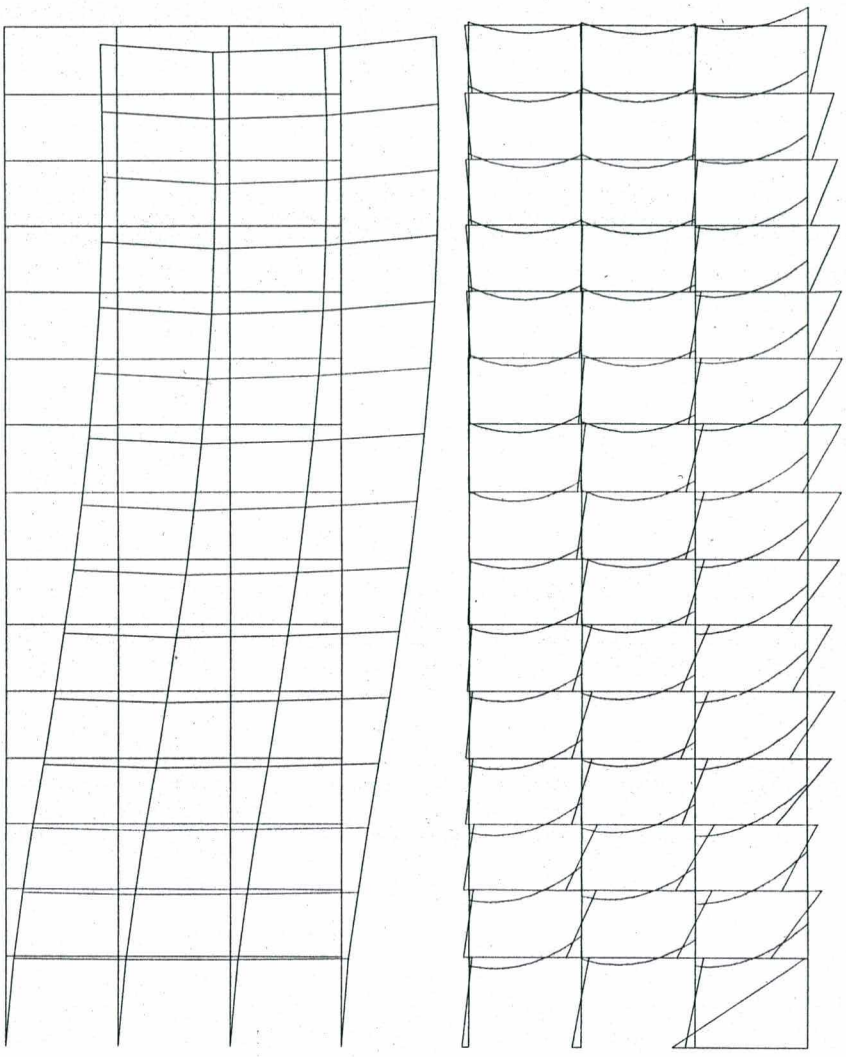
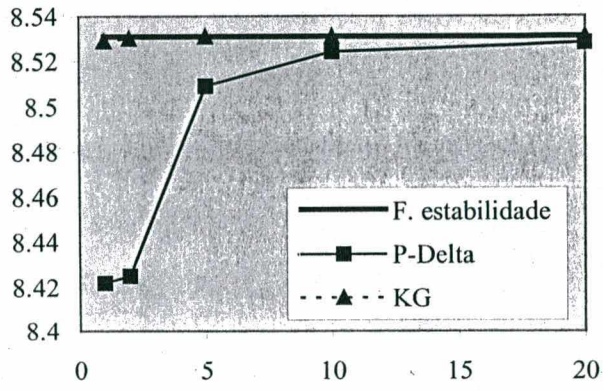


Figura 5.45 – Exemplo 10 – Estrutura deformada e diagrama de momentos fletores

Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:



Funções de estabilidade:		8.531
Nº de subdivisões	P-Delta	K_G
1	8.422	8.529
2	8.425	8.530
5	8.509	8.531
10	8.524	8.531
20	8.528	8.531

Figura 5.46 – Exemplo 10 – Variação no deslocamento no topo conforme a discretização

Em termos do deslocamento no topo da edificação, não existe qualquer diferença no comportamento deste para os modelos anteriores.

Torna-se interessante, neste ponto, verificar duas colocações:

- "A estrutura de contraventamento (no caso, formada pelo pilar parede) absorve a maior parte dos efeitos horizontais."

Pode-se utilizar como parâmetro de comparação a reação horizontal absorvida na base do pilar parede, que foi de 15.0 tf, em relação à reação horizontal absorvida pelas demais três fundações, que totalizaram 5.1 tf. Nota-se que o pilar parede absorveu sozinho 75% da reação horizontal total da edificação. Em termos de momentos fletores, apresentou um momento de 61.1 tf na base, muitas vezes superior aos demais três pilares, todos apresentando momentos fletores na ordem de 4 tf. Este pilar parede poderia, portanto, ser classificado como a subestrutura de contraventamento deste modelo.

- "Os elementos pertencentes à parte contraventada da estrutura podem ser calculados a partir dos efeitos de 1ª ordem".

O pilar parede absorve realmente a maior parcela dos efeitos horizontais. Em sua base ocorre o maior momento fletor da edificação, 61.1 tf, o qual altera-se para 66.7 tf (9.1%) após a análise de 2ª ordem. Este acréscimo apenas equivale-se ao que ocorre com o deslocamento no topo.

Analisando-se a variação nos momentos fletores nos demais pilares, vemos que a hipótese de se desconsiderarem nestes os efeitos de 2ª ordem não é válida. Na figura a seguir, indicam-se os diagramas de momentos fletores correspondentes aos 6 primeiros lances deste modelo. Ao lado de cada pilar, apresenta-se a respectiva variação ocorrida no momento fletor referido à base do pilar, após a consideração dos efeitos de 2ª ordem.

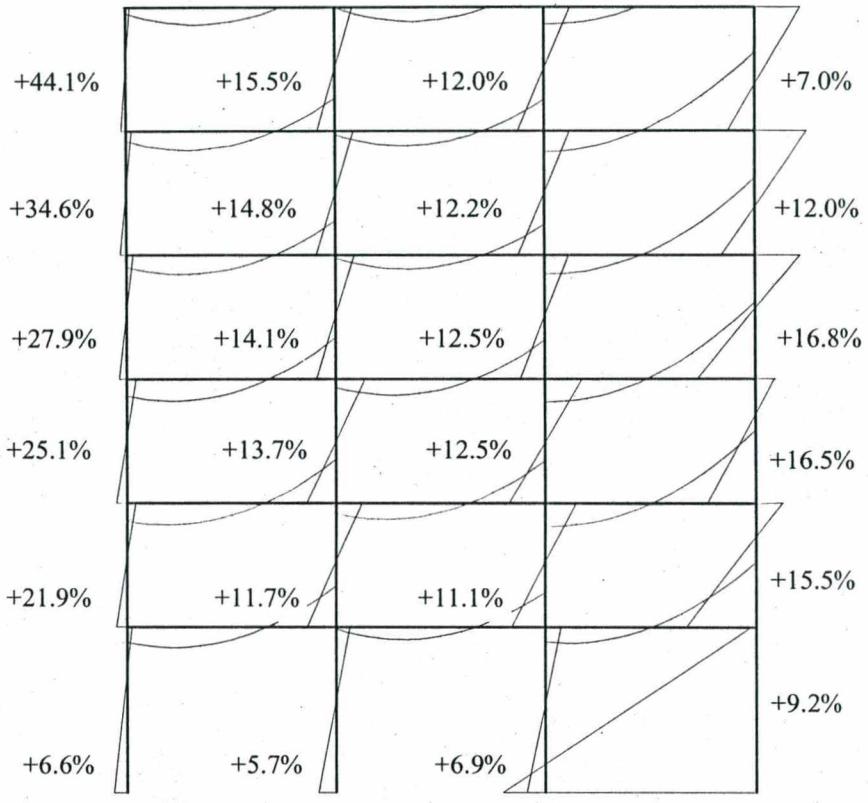


Figura 5.47 – Exemplo 10 – Variação nos momentos fletores nos pilares para os lances iniciais

Nota-se que, embora o pilar parede absorva a maior parte dos efeitos horizontais, sendo responsável, em módulo, pelo maior acréscimo de momentos fletores, o mesmo não se observa quando se consideram as variações percentuais. Mesmo desconsiderando-se os grandes aumentos (superiores a 40%) sofridos pelos pilares que apresentam momentos fletores muito baixos, ocorrem diferenças significativas em diversos pilares componentes da estrutura. Variações em torno de 12 a 15% podem ser observadas em grande número de pilares.

Estas variações não podem ser simplesmente desprezadas, por incorrerem em erro significativo contra a segurança. Deve-se atentar para o fato de que, embora os pilares ditos contraventados possuam menores esforços, apresentam também seções transversais menores e, por conseguinte, capacidade resistente menor.

5.6.8 Pórtico de 15 pavimentos associado a um pilar parede

No exemplo anterior, considerou-se a presença de um pilar parede como componente do pórtico resistente. No modelo, consideraram-se ligações rígidas entre o pilar e a estrutura restante, por intermédio de vigas em cada pavimento. Outra situação muito comum ocorre no caso de se utilizarem pórticos equivalentes para representar uma estrutura, acoplando-os através de ligações axialmente rígidas. Considera-se, desta forma, que a transmissão dos esforços horizontais entre os pórticos resistentes seja feito pelo efeito de diafragma rígido apresentado pelo painel de lajes.

Objetiva-se, com este exemplo, analisar uma situação onde acoplam-se um pórtico resistente e um pilar parede. Parte-se da mesma estrutura utilizada para o Exemplo 10, utilizando-se a mesma seção transversal de 20 x 100 cm para o pilar parede, constante ao longo da altura da edificação, mas considerando as ligações horizontais entre as duas partes da estrutura como sendo livres ao giro. Mantém-se as demais seções transversais e carregamentos..

Estas variações não podem ser simplesmente desprezadas, por incorrerem em erro significativo contra a segurança. Deve-se atentar para o fato de que, embora os pilares ditos contraventados possuam menores esforços, apresentam também seções transversais menores e, por conseguinte, capacidade resistente menor.

5.6.8 Pórtico de 15 pavimentos associado a um pilar parede

No exemplo anterior, considerou-se a presença de um pilar parede como componente do pórtico resistente. No modelo, consideraram-se ligações rígidas entre o pilar e a estrutura restante, por intermédio de vigas em cada pavimento. Outra situação muito comum ocorre no caso de se utilizarem pórticos equivalentes para representar uma estrutura, acoplando-os através de ligações axialmente rígidas. Considera-se, desta forma, que a transmissão dos esforços horizontais entre os pórticos resistentes seja feito pelo efeito de diafragma rígido apresentado pelo painel de lajes.

Objetiva-se, com este exemplo, analisar uma situação onde acoplam-se um pórtico resistente e um pilar parede. Parte-se da mesma estrutura utilizada para o Exemplo 10, utilizando-se a mesma seção transversal de 20 x 100 cm para o pilar parede, constante ao longo da altura da edificação, mas considerando as ligações horizontais entre as duas partes da estrutura como sendo livres ao giro. Mantém-se as demais seções transversais e carregamentos..

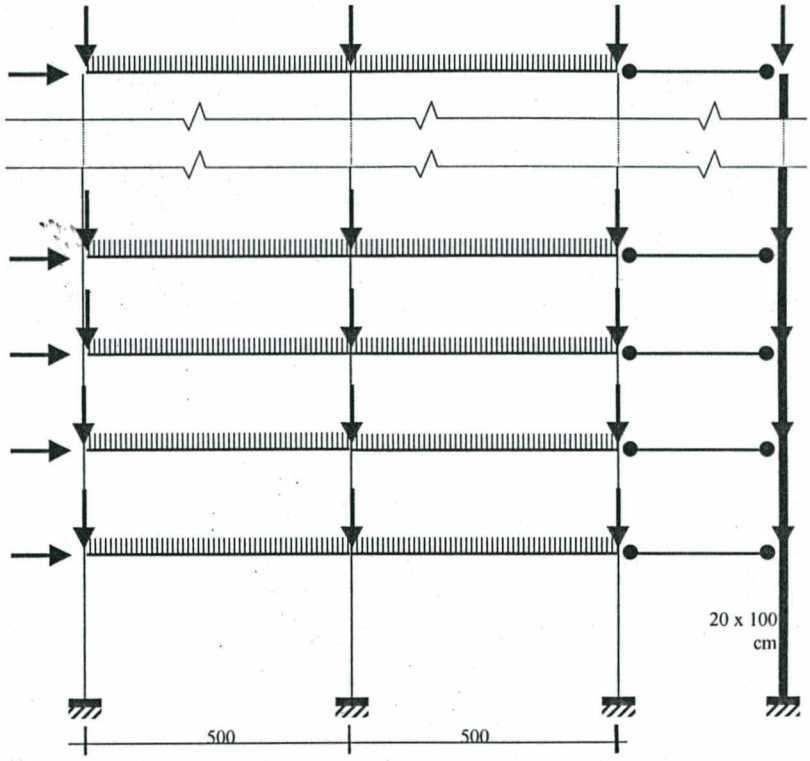


Figura 5.48 – Exemplo 11 – Pórtico de 15 pavimentos associado a um pilar parede

Efetuando-se a análise de 1ª ordem desta estrutura, obtém-se o seguinte comportamento:

- Deslocamento no topo: 22.549 cm
- Parâmetro Alfa: $\alpha = 1.248$
- Coeficiente Gama-Z: $\gamma_z = 1.300$

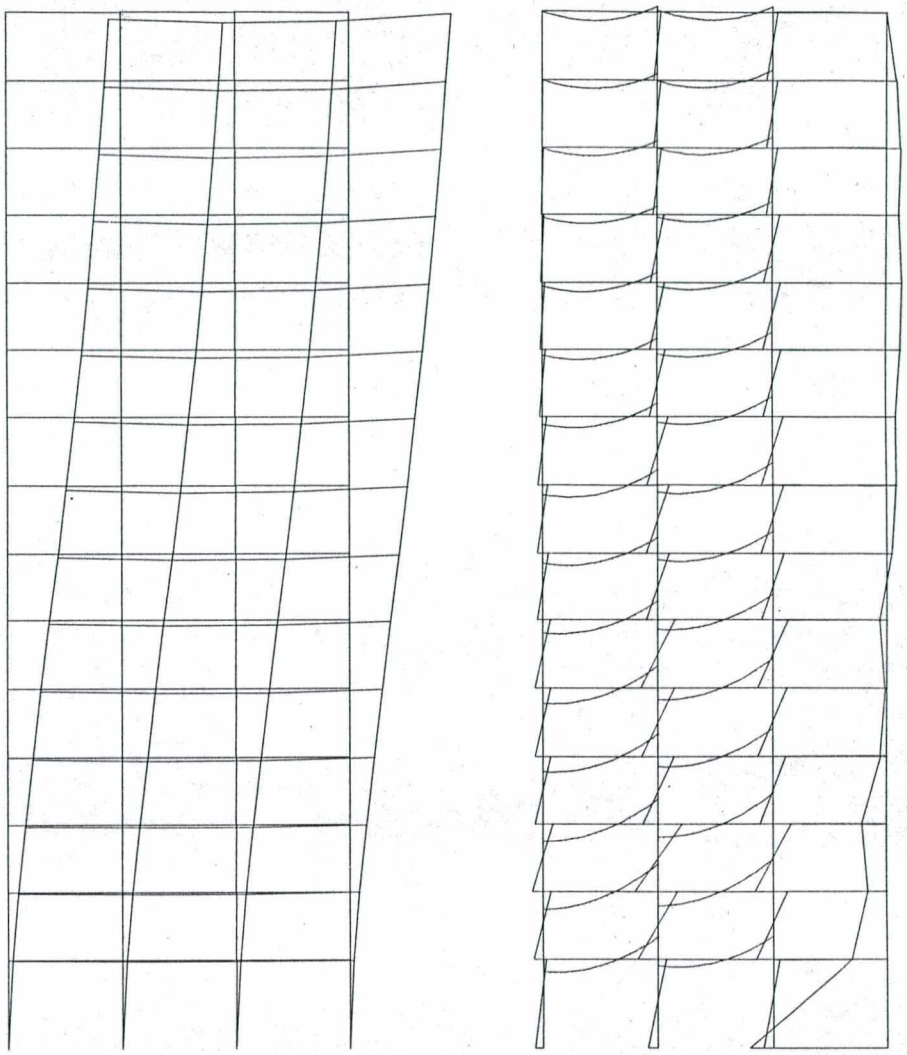
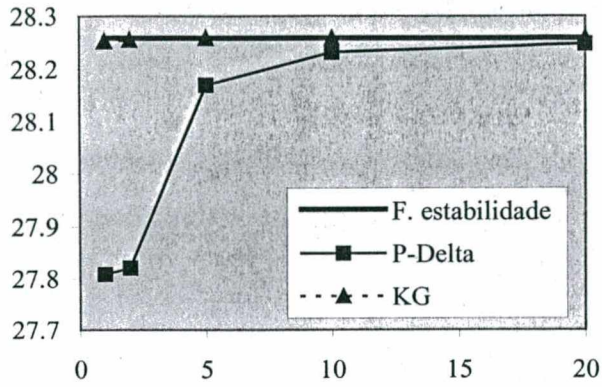


Figura 5.49 – Exemplo 11 – Estrutura deformada e diagrama de momentos fletores

Repetindo-se os procedimentos do item 5.3.1 (Discretização uniforme), obtém-se os seguintes resultados:



Funções de estabilidade:		28.077
Nº de subdivisões	P-Delta	K_G
1	27.808	28.253
2	27.821	28.255
5	28.169	28.258
10	28.232	28.258
20	28.248	28.258

Figura 5.50 – Exemplo 11 – Variação no deslocamento no topo conforme a discretização

O comportamento deste modelo, mesmo com a inclusão do pilar parede e das ligações rígidas, em nada difere dos exemplos anteriores. Com a eliminação da ligação entre as duas partes da estrutura, nota-se uma diminuição bastante grande da rigidez da estrutura como um todo. Todavia, apresenta-se também neste modelo o mesmo comportamento do exemplo anterior: embora os maiores momentos fletores ocorram no pilar parede, variações percentuais ocorrem ao longo de todos os elementos.

Pode-se verificar isso analisando a variação nos esforços internos nos pilares pertencentes à parte dita contraventada da estrutura, através do diagrama a seguir.

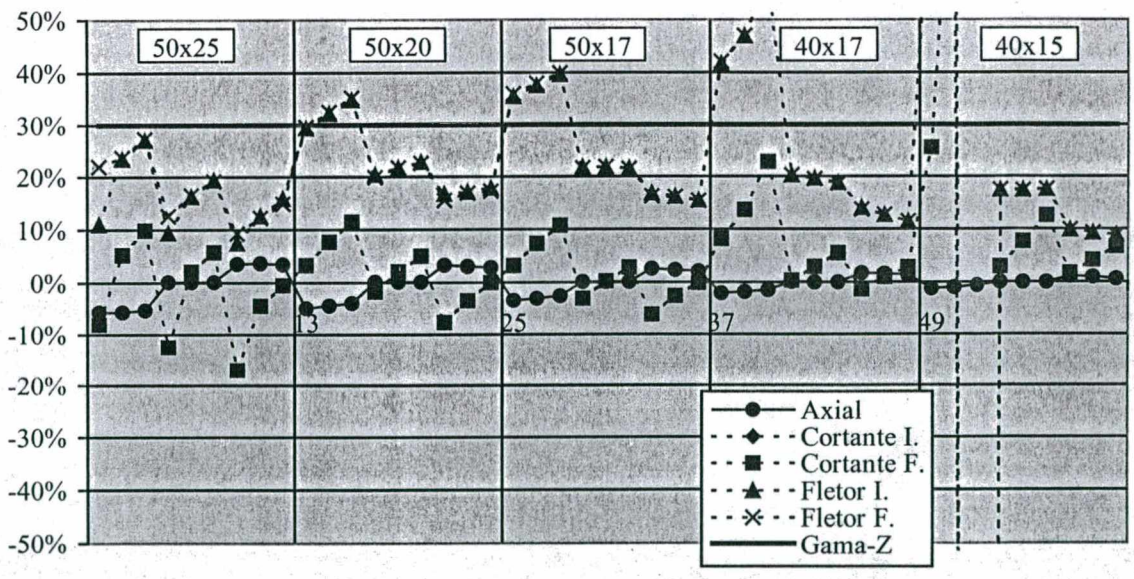


Figura 5.51 – Exemplo 11 – Variação nos esforços internos dos pilares

5.6.9 Resumo e comentários

Após a exposição deste diversos exemplos, cabe fazer uma rápida análise de alguns resultados apresentados. Inicialmente, estabelece-se como parâmetro o deslocamento ocorrido no topo da edificação e deseja-se comparar a sua variação com os valores dos parâmetros simplificados para avaliação da estabilidade global (o Parâmetro Alfa e o Coeficiente Gama-Z).

Modelo	α	γ_z	δ_0	δ_f
Exemplo 2	1.063	1.386	11.563	25.964 (Δ=2.245)
Exemplo 3	0.649	1.100	4.309	4.867 (Δ=1.129)
Exemplo 4	0.992	1.091	4.576	5.108 (Δ=1.116)
Exemplo 5	0.992	1.171	33.992	40.326 (Δ=1.186)
Exemplo 6	1.120	1.273	43.367	69.692 (Δ=1.607)
Exemplo 7	1.012	1.185	20.889	25.148 (Δ=1.204)
Exemplo 8	0.968	1.097	2.335	2.546 (Δ=1.090)
Exemplo 9	0.952	1.148	18.498	21.919 (Δ=1.185)
Exemplo 10	0.719	1.098	7.355	8.531 (Δ=1.160)
Exemplo 11	1.248	1.300	22.549	28.077 (Δ=1.245)

Tabela 5.31 – Resumo dos exemplos apresentados

Ao longo deste capítulo, foram efetuadas diversas avaliações a respeito da efetividade da aplicação do coeficiente γ_z como majorador dos esforços de 1ª ordem, visando a obtenção direta dos esforços de 2ª ordem. Notou-se que o valor de γ_z é um indicativo razoável da variação no deslocamento horizontal da edificação, mas que representa, no máximo, um comportamento médio em termos de esforços. A ocorrência de diversos elementos onde os efeitos de 2ª ordem são superiores a γ_z, aliada ao fato de que isto normalmente ocorre nos pontos mais solicitados da edificação, faz com que não seja recomendável a adoção deste coeficiente para obtenção dos esforços finais.

Além disto, pode-se notar, para fins de comparação, que o erro relativo apresentado pelo coeficiente γ_z foi inaceitavelmente grande no caso do Exemplo 2 e do Exemplo 6, os dois modelos cujas fundações foram consideradas rotuladas.

Por outro lado, embora se tenha apresentado o valor do coeficiente α para todos os modelos, este não foi comentado em nenhum ponto. Isto porque, conforme exposto no item 2.4.5, a formulação do Parâmetro Alfa pode ser considerada bastante inferior à do Coeficiente Gama-Z. Analisando-se a Tabela 5.31, nota-se a grande disparidade encontrada nos valores deste coeficiente. Por exemplo:

- O Exemplo 3 e o Exemplo 4 apresentam valores de γ_z muito próximos, mas valores completamente diferentes de α;
- O Exemplo 10 apresenta valor de α inferior ao do Exemplo 8, com o comportamento real inverso disto.

Além da falta de precisão, este parâmetro define unicamente um valor limite, não permitindo, contudo, que se avalie exatamente o quanto se está longe dele. Um coeficiente α de 0.9, por exemplo, está 50% acima do valor limite de 0.6 (o qual representa um efeito de 2ª ordem de 10%), mas isto não fornece qualquer indicativo de qual seria o efeito de 2ª ordem equivalente.

Por este motivo, considera-se que, na impossibilidade de se fazer uma análise mais completa por intermédio de um dos processos apresentados, ou na necessidade de um cálculo expedito, é mais interessante a adoção do Coeficiente Gama-Z em relação ao Parâmetro Alfa.

CAPÍTULO 6.
ASPECTOS DE PROJETO

A aplicação dos conceitos de não linearidade geométrica em projetos já representa um enorme avanço em direção à obtenção de modelos mais representativos, que possam garantir a segurança da estrutura com máximo grau de confiabilidade e que possibilitem, ao final, maior economia para a estrutura. Conforme exposto no item 2.4.2, a nova NBR 6118 deve expressar sua preocupação com os efeitos de 2ª ordem, mas de forma a permitir o uso de processos simplificados (a saber, o parâmetro Alfa e o coeficiente Gama-Z) que simplesmente estimam a magnitude destes, de forma a evitar análises mais elaboradas.

Este trabalho dedica-se a mostrar que a inclusão dos efeitos não lineares geométricos ao projeto de estruturas é simples e fornece diversas vantagens em relação aos processos simplificados, por dois motivos:

- Fornece resultados mais confiáveis, evitando possíveis erros contra a segurança, uma vez que os parâmetros simplificados podem dispensar a análise de 2ª ordem em uma estrutura onde estas diferenças podem ser relevantes. Critica-se especialmente o uso do coeficiente Gama-Z para a majoração dos esforços de 1ª ordem, conforme sugerido por SANTOS & FRANCO (1993) e, segundo os autores, permitido pela proposta de revisão da NBR 6118.
- Permite a análise de estruturas onde os efeitos de 2ª ordem devem ser considerados, possibilitando uso de estruturas mais arrojadas e evitando a necessidade de se enrijecer a estrutura até que os efeitos de 2ª ordem caiam abaixo de 10%.

A aplicação dos processos apresentados a projetos reais ainda possui limitações de ordem computacional, mas estas barreiras têm sido rapidamente eliminadas. O uso do Processo P-Delta (vide item 3.2) permite a consideração da não linearidade geométrica mesmo em estruturas com milhares de nós, com pequeno acréscimo ao custo computacional. Os outros dois processos expostos (Processo da Matriz de Rigidez Geométrica, no item 3.3, e Processo das Funções de Estabilidade, no item 3.5) possuem custo mais elevado, como exposto no item 3.3.6, mas têm aplicação justificada nos casos onde uma precisão maior é requerida.

Para efetiva aplicação em projetos reais, alguns detalhes devem ser lembrados, referentes a diversas particularidades que devem ser cobertas pelos métodos a serem utilizados. A seguir, expõem-se alguns pontos de interesse.

6.1 - ALTERAÇÕES NA FORMULAÇÃO

Ao se tentar aplicar os processos propostos às estruturas reais, diversos pontos de interesse, utilizados nas análises de 1ª ordem, devem também ser contemplados nas análises de 2ª ordem.

6.1.1 Liberação de vinculações

Uma necessidade referente aos projetos usuais diz respeito à liberação de algumas das vinculações existentes nas barras. Esta liberação diz respeito não às vinculações dos nós, mas sim aos graus de liberdade considerados para a barra.

Uma aplicação diz respeito às estruturas onde se consideram que todas as ligações entre os elementos são rotuladas, ou seja, não transmitem momentos fletores. Uma estrutura formada apenas por barras que não possuem rigidez à flexão é chamada *treliça* e pode ser calculada com um número menor de graus de liberdade do que os pórticos. Usualmente, uma mesma estrutura possui elementos com rigidez à flexão e outros não. Desta forma, uma estrutura de pórtico plano pode conter barras de treliça, normalmente definidas apenas considerando-se sua rigidez à flexão nula (momento de inércia da seção transversal igual a zero).

Outra situação mais complexa ocorre quando uma das extremidades da barra é rotulada e outra não. Em termos de projeto, esta situação pode ser considerada bastante comum. Neste caso, uma nova dedução da matriz de rigidez deve ser feita, de forma análoga ao citado no item 2.3.2.

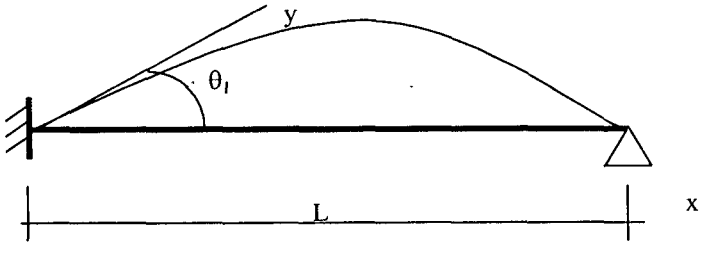


Figura 6.1 – Barra com sua extremidade final rotulada

A matriz de rigidez elástica de uma barra de pórtico plano com a sua extremidade inicial engastada e a outra rotulada, referida ao seu sistema local, apresenta-se como seguinte (BEAUFAIT et al., 1970):

$$[r]=\begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{3EI}{L^3} & -\frac{3EI}{L^2} & 0 & -\frac{3EI}{L^3} & 0 \\ 0 & -\frac{3EI}{L^2} & \frac{3EI}{L} & 0 & \frac{3EI}{L^2} & 0 \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{3EI}{L^3} & \frac{3EI}{L^2} & 0 & \frac{3EI}{L^3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{Eq. (6-124)}$$

Da mesma forma, a modificação na vinculação da barra deve ser considerada na inclusão da força axial na formulação da matriz de rigidez.

Processo P-Delta

No caso do Processo P-Delta, por não levar em conta o comportamento à rigidez da barra (vide item 3.2.4), esta diferença não é levada em conta. Na verdade, o Processo P-Delta é deduzido como se todas as barras fossem livres à flexão, sendo esta uma das razões pela qual pode ser considerado um processo mais aproximado que os demais.

Processo da Matriz de Rigidez Geométrica

Utilizando o Processo da Matriz de Rigidez Geométrica, MOREIRA (1977) apresenta a matriz K_G que deve ser acrescentada à matriz de rigidez elástica quando a barra possui sua extremidade final rotulada. Esta matriz substitui a Eq. (3-46), devendo ser acrescentada à matriz elástica expressa pela Eq. (6-124) para obter a matriz de rigidez total da barra.

$$[K_G]=\frac{P}{L}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6}{5} & -\frac{L}{5} & 0 & 0 & 0 \\ 0 & -\frac{L}{5} & \frac{6\cdot L^2}{5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{Eq. (6-125)}$$

Processo das Funções de Estabilidade

O Processo das Funções de Estabilidade também necessita ser modificado de forma análoga. Não se encontram-se facilmente dispo níveis formulações que estendam os conceitos utilizados nas Funções de Estabilidade. Pode-se considerar que esta abordagem, baseada na obtenção direta da matriz de rigidez partindo-se da equação diferencial da viga, é trabalhosa e menos sistemática.

Conforme colocado no item 3.6.2, existem diversas alternativas aos processos apresentados, mas nota-se a grande preferência dos autores pela adoção dos métodos energéticos, da mesma forma como feito para a dedução da matriz K_G, no item 3.3.1. Pode-se apontar esta como uma das vantagens do Processo da Matriz de Rigidez Geométrica em relação ao Processo das Funções de Estabilidade: pelo fato de sua formulação ser mais simples e direta, pode-se incluir a esta mais facilmente as possíveis variações. Sua formulação poderia ser facilmente estendida para barras com seção transversal variável, vinculações semi-rígidas ou outras, como é relativamente comum no Método dos Elementos Finitos.

6.1.2 Deformações por cisalhamento

O efeito das deformações por cortante, desprezado em todas as formulações apresentadas, também pode afetar a estabilidade de uma estrutura. Independentemente de carga axial, a presença de forças cortantes reduz alguns dos elementos da matriz de rigidez, porque as deformações por cisalhamento diminuem as ações de engastamento induzidas pelos deslocamentos unitários.

Pode-se deduzir facilmente a matriz de rigidez elástica de uma barra de pórtico plano considerando as deformações por cisalhamento (WEAVER & GERE, 1965, BIRNSTIEL & IFFLAND, 1980, SELKE & PEREIRA, 1994, entre outros). A matriz expressa pela Eq. (2-5) modifica-se como seguinte:

$$[r]=\begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{(1+2g)L^3} & \frac{6EI}{(1+2g)L^2} & 0 & -\frac{12EI}{(1+2g)L^3} & \frac{6EI}{(1+2g)L^2} \\ 0 & \frac{6EI}{(1+2g)L^2} & \frac{\left(1+\frac{g}{2}\right)4EI}{(1+2g)L} & 0 & -\frac{6EI}{(1+2g)L^2} & \frac{(1-g)2EI}{(1+2g)L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{(1+2g)L^3} & -\frac{6EI}{(1+2g)L^2} & 0 & \frac{12EI}{(1+2g)L^3} & -\frac{6EI}{(1+2g)L^2} \\ 0 & \frac{6EI}{(1+2g)L^2} & \frac{(1-g)2EI}{(1+2g)L} & 0 & -\frac{6EI}{(1+2g)L^2} & \frac{\left(1+\frac{g}{2}\right)4EI}{(1+2g)L} \end{bmatrix} \qquad \text{Eq. (6-126)}$$

$$g=\frac{6\cdot F\cdot E\cdot I}{G\cdot A\cdot L^2} \qquad \text{Eq. (6-127)}$$

Onde:

- F = fator de forma da seção.

Embora o efeito das deformações por cisalhamento possa ser facilmente incluído através da Eq. (6-127) e o efeito da força normal possa ser incluído de diversas maneiras discutidas ao longo deste trabalho, a consideração simultânea dos dois efeitos ainda não se encontra bem estudada. BIRNSTIEL & IFFLAND (1980) sugerem, na falta de estudos mais precisos, a multiplicação da matriz de rigidez geométrica obtida considerando-se a força normal pelos fatores (1+g) e (1+2g) devidos à deformação por cisalhamento.

Na prática, este problema possui pouca importância, visto as deformações por cisalhamento serem usualmente muito pequenas. Em termos de projeto, estas deformações tornam-se realmente relevantes apenas no caso de vigas parede. Segundo CORRÊA & RAMALHO (1193), pode-se considerar uma viga parede como componente de um sistema estrutural, representando-a também por um elemento linear, desde que se considere a deformação por cisalhamento. Como a força axial presente neste tipo de elemento é muito pequena, a aproximação sugerida por BIRNSTIEL & IFFLAND (1980) pode ser utilizada sem perda de precisão relevante.

6.2 - ESTRUTURAS ESPACIAIS

Nos dias de hoje, a análise das estruturas sob o ponto de vista tridimensional já pode ser considerada algo comum. O conceito, universalmente utilizado até uma década atrás, de dividir as estruturas em *subestruturas de contraventamento*, responsáveis pela estabilidade horizontal da edificação e calculadas como pórticos planos ou pilares parede, vem sido substituído pela análise da estrutura em modelos de pórtico espacial. Neste caso, o comportamento da estrutura pode ser descrito como um todo, eliminando a necessidade de dividir a estrutura em partes (o que facilita em muito a automatização dos processos).

Embora este trabalho verse sobre estruturas planas, isto foi feito apenas para simplificar a interpretação dos resultados. Todas as formulações apresentadas podem ser facilmente estendidas para o caso espacial.

6.2.1 Processo P-Delta

O Processo P-Delta, por se basear unicamente no equilíbrio de uma barra, pode ser aplicado diretamente, considerando-se apenas as forças horizontais fictícias em cada um dos eixos das barras.

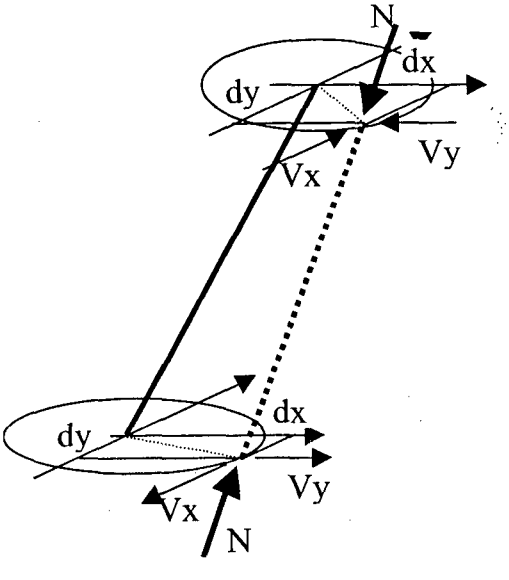


Figura 6.2 – Processo P-Delta aplicado ao caso tridimensional

Aplicando-se a Eq. (3-1) separadamente para as direções X e Y, pode-se reproduzir o comportamento não linear da estrutura sob forma espacial.

Da mesma forma como feito no item 3.2, a cada barra da estrutura acrescentam-se cortantes fictícios em suas extremidades, obtidos através de simples equilíbrio:

$$V_x = \frac{N \cdot a_x}{L} \qquad \text{Eq. (6-128)}$$

$$V_y = \frac{N \cdot a_y}{L}$$

Onde:

- L = extensão da barra;
- N = carga normal atuante;
- a_x = deslocamento relativo as extremidades da barra, contado paralelamente ao eixo X;
- a_y = deslocamento relativo as extremidades da barra, contado paralelamente ao eixo Y;
- V_x = cortante fictício a ser acrescentado à análise, paralelo à direção X;
- V_y = cortante fictício a ser acrescentado à análise, paralelo à direção Y.

A extensão do Processo P-Delta ao caso espacial em nada altera o fluxograma de aplicação deste (conforme exposto na Figura 3.5). A montagem do vetor de forças deve incluir simultaneamente os cortantes fictícios obtidos para as direções X e Y.

Embora a evolução dos deslocamentos obtidos em uma direção usualmente não afete os deslocamentos na outra direção, um comportamento espacial que resulte na torção da estrutura como um todo só pode ser analisado aplicando-se simultaneamente os cortantes fictícios nas duas direções. Além disto, a variação na força normal atuante nas barras, mesmo que seja usualmente pequena, provocada pelo aumento dos deslocamentos em uma das direções, causa variação nos resultados obtidos na outra direção. Pode-se considerar que o processo convergiu quando todos os deslocamentos (considerando as duas direções, X e Y) não apresentam mais diferença significativa em relação à iteração anterior.

6.2.2 Processo da Matriz de Rigidez Geométrica

Nas formulações que envolvem a alteração da matriz de rigidez da barra, as mesmas considerações já feitas na modificação da rigidez à flexão da barra sob carga normal podem ser feitas separadamente para as duas direções da barra, resultando nos termos adicionais da matriz de rigidez de uma barra de pórtico espacial.

No caso espacial, consideram-se seis graus de liberdade por nó (rotação e translação em cada um dos eixos X, Y e Z), totalizando doze graus de liberdade na barra.

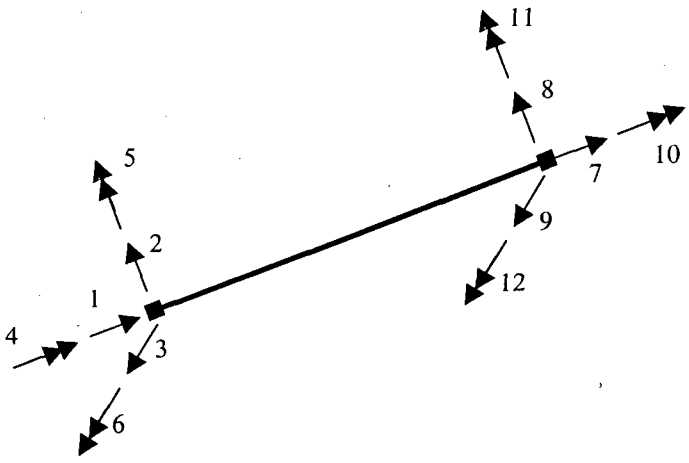


Figura 6.3 - Graus de liberdade de uma barra de pórtico espacial

A matriz de rigidez elástica da barra, referida ao seu sistema de coordenadas locais, apresenta-se como seguinte (WEAVER & GERE, 1965):

$$[r] = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ & \frac{12EI_y}{L^3} & 0 & 0 & \frac{6EI_y}{L^2} & 0 & 0 & -\frac{12EI_y}{L^3} & 0 & 0 & \frac{6EI_y}{L^2} & 0 \\ & & \frac{12EI_x}{L^3} & 0 & 0 & \frac{6EI_y}{L^2} & 0 & 0 & -\frac{12EI_x}{L^3} & 0 & 0 & \frac{6EI_x}{L^2} \\ & & & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ & & & & \frac{4EI_y}{L} & 0 & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & \frac{2EI_y}{L} & 0 \\ & & & & & \frac{4EI_x}{L} & 0 & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & \frac{2EI_x}{L} \\ & & & & & & \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & \frac{12EI_y}{L^3} & 0 & 0 & -\frac{6EI_y}{L^2} & 0 \\ & & (sim) & & & & & & \frac{12EI_x}{L^3} & 0 & 0 & -\frac{6EI_x}{L^2} \\ & & & & & & & & & \frac{GJ}{L} & 0 & 0 \\ & & & & & & & & & & \frac{4EI_y}{L} & 0 \\ & & & & & & & & & & & \frac{4EI_x}{L} \end{bmatrix} \quad \text{Eq. (6-129)}$$

Onde:

- L = comprimento;
- I_x = momento de inércia da seção transversal, referido ao seu eixo local X;
- I_y = momento de inércia da seção transversal, referido ao seu eixo local Y;
- A = área da seção transversal; e
- E = módulo de elasticidade do material.

Pode-se notar que os termos de rigidez adicionais são exatamente análogos àqueles definidos para a barra de pórtico plano, sendo independentes entre si.

Adicionalmente, apresenta-se um termo referente à rigidez à torção da barra, expressa por:

$$K_T = \frac{G \cdot J}{L} \quad \text{Eq. (6-130)}$$

Onde:

- G = módulo de rigidez à torção;
- J = momento de inércia polar da barra;
- L = comprimento da barra.

Na abordagem adotada para a dedução da matriz de rigidez geométrica, onde considera-se linear a influência da força normal, pode-se estender facilmente o raciocínio utilizado no item 3.3.2 de forma separada para as duas direções da barra. Adotando-se as mesmas funções de interpolação para ambas as direções, chega-se exatamente aos mesmos termos nos graus de liberdade análogos.

Uma vez que os deslocamentos unitários referentes à torção nas extremidades da barra (graus de liberdade 4 e 10 na Figura 6.3) não acarretam deslocamentos transversais na barra, a presença de uma força normal P não modifica os esforços de imobilização correspondentes. Chega-se, portanto, a uma matriz K_G referente a uma barra de pórtico espacial, cujos termos também são funções lineares da carga normal P.

$$[K_G]=\frac{P}{L}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & \frac{6}{5} & 0 & 0 & \frac{L}{10} & 0 & 0 & -\frac{6}{5} & 0 & 0 & \frac{L}{10} & 0 \\
 & & \frac{6}{5} & 0 & 0 & \frac{L}{10} & 0 & 0 & -\frac{6}{5} & 0 & 0 & \frac{L}{10} \\
 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & \frac{2\cdot L^2}{15} & 0 & 0 & -\frac{L}{10} & 0 & 0 & -\frac{L^2}{30} & 0 \\
 & & & & & \frac{2\cdot L^2}{15} & 0 & 0 & -\frac{L}{10} & 0 & 0 & -\frac{L^2}{30} \\
 & & & & & & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & \frac{6}{5} & 0 & 0 & -\frac{L}{10} & 0 \\
 & & & & & & & & \frac{6}{5} & 0 & 0 & -\frac{L}{10} \\
 & & & & (sim) & & & & & 0 & 0 & 0 \\
 & & & & & & & & & & \frac{2\cdot L^2}{15} & 0 \\
 & & & & & & & & & & & \frac{2\cdot L^2}{15}
\end{bmatrix} \qquad \text{Eq. (6-131)}$$

Esta matriz K_G, da mesma forma como definido em 3.3.3, deve ser adicionada à matriz de rigidez elástica, ou, conforme abordagem definida no Processo da Matriz de Rigidez Geométrica Modificado, no item 3.4, ser utilizada para majorar os deslocamentos obtidos na iteração anterior e obter o novo vetor de cargas fictícias.

6.2.3 Processo das Funções de Estabilidade

No Processo das Funções de Estabilidade, conforme exposto no item 3.5, considera-se explicitamente a não linearidade do problema na dedução da matriz de rigidez.

A obtenção das funções de estabilidade e, por conseguinte, da matriz de rigidez da barra, é feita com base na equação diferencial de uma viga:

$$M = EI\frac{d^2y}{dx^2}$$

No caso espacial, a equação acima pode ser aplicada separadamente para as duas direções principais da barra, utilizando-se os valores de I_x e I_y correspondentes. A diferença no momento de inércia da barra nas duas direções resulta em um valor diferente para a carga crítica P_E para cada direção. Analogamente, obtém-se:

$$P = \Phi_x \cdot P_{Ex} \Rightarrow \Phi_x = P\frac{L^2}{\pi^2 EI_x} \qquad \text{Eq. (6-132)}$$

$$P = \Phi_y \cdot P_{Ey} \Rightarrow \Phi_y = P\frac{L^2}{\pi^2 EI_y}$$

A necessidade de definição de duas funções Φ incorre em funções de estabilidade diferentes para os dois eixos principais da barra. Definem-se, portanto, as funções de rotação c_x, c_y, r_x e r_y e as funções de translação t_x e t_y, de forma análoga ao feito no item 3.5.1.

BIRNSTIEL & IFFLAND (1980) definem uma *função de estabilidade à torção*, análoga às *funções de rotação* desenvolvidas no item 3.5.1 para o Processo das Funções de Estabilidade. Esta função T multiplica a rigidez elástica à torção e é definida de forma diferente para o caso de forças de compressão e de tração.

Define-se, inicialmente, a relação:

$$k_P = G \cdot J - |P| \cdot r_0^2 \qquad \text{Eq. (6-133)}$$

Onde:

- P = carga axial aplicada na barra;
- r_0 = raio de giração polar da barra.

No caso de carga P compressiva e k_p positivo:

$$T = \frac{(k \cdot L)^3}{k \cdot L - 2 \cdot \tanh\left(k \cdot L/2\right)} \qquad \text{Eq. (6-134)}$$

$$\text{Onde:} \quad k = \sqrt{\frac{G \cdot J - |P| \cdot r_0^{\,2}}{E \cdot C^*}}$$

No caso de carga P compressiva e k_P negativo:

$$T = \frac{(k \cdot L)^3}{2 \cdot (1 - \cos(k \cdot L)) - k \cdot L \cdot \operatorname{sen}(k \cdot L)} \qquad \text{Eq. (6-135)}$$

$$\text{Onde:} \quad k = \sqrt{\frac{|P| \cdot r_0^{\,2} - G \cdot J}{E \cdot C^*}}$$

No caso de carga de tração:

$$T = \frac{(k \cdot L)^3}{k \cdot L - 2 \cdot \tanh\left(k \cdot L/2\right)} \qquad \text{Eq. (6-136)}$$

$$\text{Onde:} \quad k = \sqrt{\frac{G \cdot J + |P| \cdot r_0^{\,2}}{E \cdot C^*}}$$

Nas três expressões, tem-se:

- E = módulo de elasticidade;
- C^* = constante de empenamento da seção.

Podem-se notar, portanto, duas diferenças essenciais entre a formulação da matriz de rigidez geométrica e a das funções de estabilidade:

- Na primeira, não existe diferença nos termos de rigidez para as duas direções da barra, enquanto que na segunda sim, caso a rigidez à flexão da barra seja diferente nas duas direções.
- Na primeira, os termos de rigidez à torção não são afetados, enquanto que na segunda é necessário definir uma função de estabilidade à torção.

6.3 - EXEMPLOS NUMÉRICOS TRIDIMENSIONAIS

A implementação computacional dos processos para inclusão da não linearidade geométrica no projeto de estruturas no caso tridimensional não é objeto deste trabalho. Conforme já exposto, optou-se pela utilização de pórticos planos para simplificar a exposição de dados e resultados.

Todavia, dada a grande importância do tema para fins de projeto, cabe tecer alguns comentários a respeito e verificar se o comportamento já mostrado para o caso 2D reproduz-se no caso 3D. Objetiva-se verificar a existência de respostas diferentes aos efeitos de 2ª ordem no caso tridimensional, sem a intenção, contudo, de analisar todas as variantes possíveis e presentes já na análise linear.

Para este estudo, optou-se pela utilização de um programa comercial, o Eberick Master, desenvolvido pela empresa AltoQi Tecnologia em Informática Ltda. Este programa baseia-se na análise da estrutura em um modelo de pórtico espacial. Para a inclusão da não linearidade geométrica, implementa o Processo P-Delta. Apresenta também os valores do Parâmetro Alfa e do Coeficiente Gama-Z, computados separadamente para as duas direções principais, que podem também ser utilizados para fins de comparação. Nos exemplos a seguir, serão utilizados resultados obtidos através deste programa.

6.3.1 Pórtico apoiado na base

A estrutura apresentada no Exemplo 2, conforme colocado inicialmente por PITTA (1996), refere-se ao pórtico equivalente de uma suposta edificação sujeita a carregamentos de vento e utilização usuais. Como primeiro exemplo, deseja-se reproduzir esta estrutura sob sua forma espacial original e analisá-la através do Processo P-Delta.

Esta estrutura é composta por 4 pavimentos iguais em planta, com dimensão de 5.00 x 20.00 m entre eixos. São utilizadas as seguintes propriedades:

- Módulo de elasticidade: $2{,}7 \times 10^5$ kgf/cm^2;
- Seção transversal das barras: 30 x 20 cm para os pilares e 13 x 55 cm para as vigas;
- Lajes maciças de 10 cm;

- Altura de 300 cm entre os pavimentos, à exceção do primeiro nível, que apresenta 400 cm de altura.

A estrutura desta edificação apresenta-se como na figura a seguir.

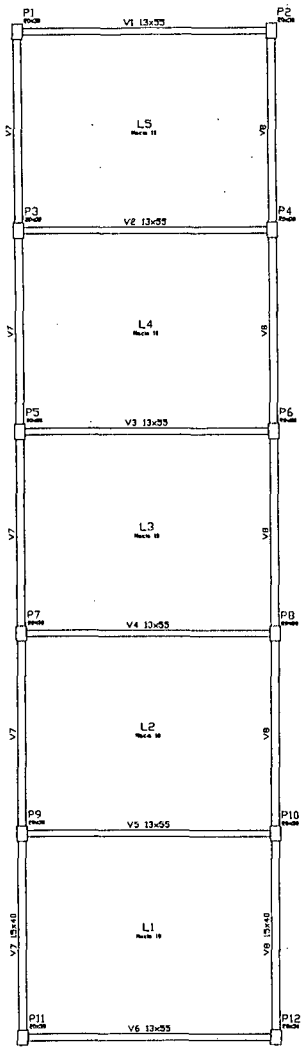


Figura 6.4 – Exemplo 12 – Pórtico apoiado na base

Tridimensionalmente, esta estrutura apresenta-se como seguinte:

Figura 6.5 – Exemplo 12 – Visualização tridimensional da estrutura

Utilizando-se os mesmos dados informados por PITTA (1996), chega-se ao pórtico equivalente do Exemplo 2, expresso na Figura 5.1, ao se considerar uma área de influência igual à semidistância entre os pilares adjacentes e se tomar um dos pórticos internos para avaliação. Segundo esta abordagem, os pórticos externos seriam bastante diferentes dos internos, ao se adotar uma área de influência para as cargas horizontais equivalente à metade dos outros, além de se terem cargas verticais concentradas diferentes, provenientes das reações de apoio das vigas longitudinais.

Em relação ao eixo X[14], coincidente com o pórtico equivalente utilizado no Exemplo 2, a análise de 1ª ordem deste modelo fornece os seguintes resultados:

- Deslocamento horizontal no topo: 9.83 cm
- Parâmetro Alfa: $\alpha = 0.99$
- Coeficiente Gama-Z: $\gamma_z = 1.92$

[14] Onde convenciona-se chamar "eixo X" o eixo horizontal da planta baixa da estrutura.

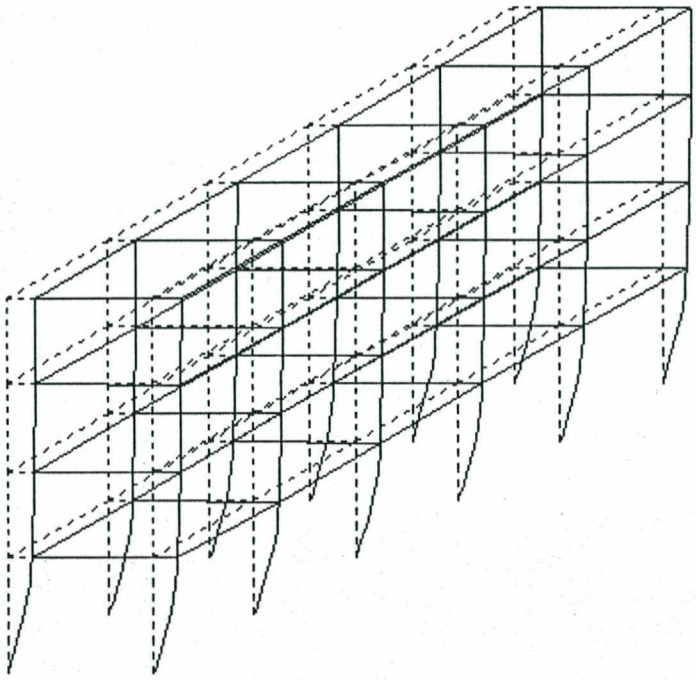


Figura 6.6 – Exemplo 12 – Estrutura deformada

Efetuada a análise de 2ª ordem através do Processo P-Delta, obteve-se um deslocamento final no topo da edificação de 16.21 cm. O deslocamento horizontal apresentado foi o mesmo em todos os pontos, pelo fato de que o programa utiliza a hipótese de diafragma rígido para consideração das lajes no modelo.

Para fins de comparação, é interessante mostrar o pórtico equivalente que seria obtido ao tomar a linha extrema de pilares da edificação.

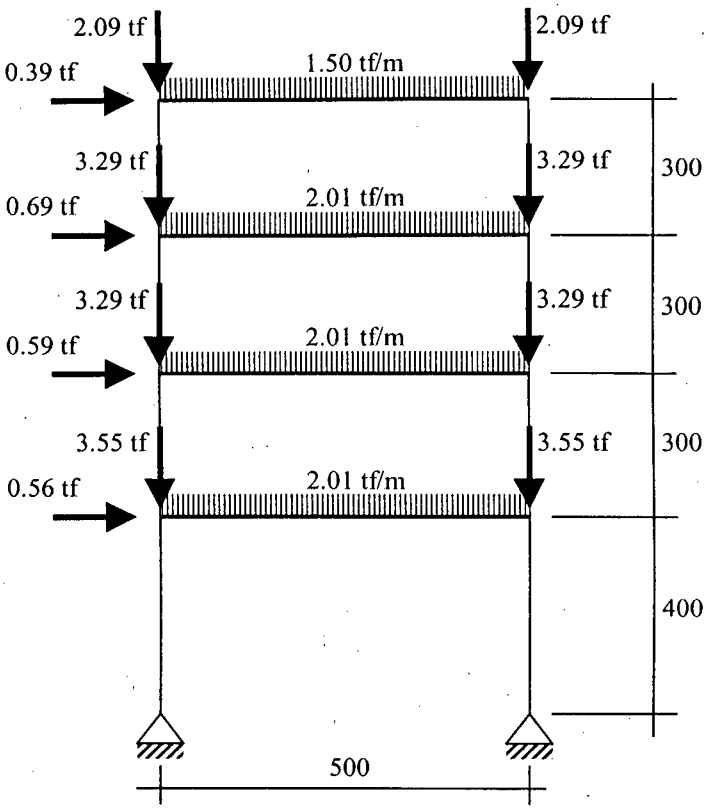


Figura 6.7 – Exemplo 12 – Pórtico plano equivalente à extrema da edificação

Para comparar os resultados obtidos no modelo 3D com o pórtico equivalente adotado para o Exemplo 2, utilizam-se os resultados obtidos através do Processo P-Delta sem discretização nas barras, obtendo a tabela a seguir:

Modelo	δ_0	δ_f
Plano	11.563	21.401 (Δ=1.851)
Plano referente à extremidade	5.820	8.271 (Δ=1.421)
Espacial	9.83	16.21 (Δ=1.649)

Tabela 6.1 – Exemplo 12 – Comparação entre os resultados obtidos no caso plano e no tridimensional

Pode-se observar que a estrutura espacial é consideravelmente mais rígida do que o pórtico equivalente utilizado no Exemplo 2, apresentando tanto um deslocamento de 1ª ordem inferior quanto efeitos de 2ª ordem menores. Isto se explica, neste exemplo, pelo fato de que a simples consideração de uma área de influência no cômputo das cargas horizontais em cada pórtico não representa a realidade. Na verdade, os seis pórticos contribuem simultaneamente para a estabilidade horizontal da edificação. Nota-se que um modelo da linha de extrema de pilares apresentaria comportamento inverso, sendo artificialmente mais rígido do que a estrutura em si.

O modelo plano poderia ser aprimorado utilizando-se a metodologia de considerar todos os pórticos em um único modelo, acoplando-os através de ligações axialmente rígidas, da mesma forma como foi feito, no item 5.6.8, para o Exemplo 11. Todavia, o aumento na dificuldade material deste procedimento, quando comparado ao pórtico equivalente simples, já torna justificável a adoção direta do modelo espacial.

Um ponto interessante a apontar é a distribuição dos efeitos de 2ª ordem ao longo do modelo espacial. Embora o efeito de diafragma rígido faça com que os deslocamentos nos topos de cada linha de pilares sejam iguais, a carga vertical atuante em cada um deles é diferente. As linhas de pilares extremas possuem cargas axiais menores, o que sugere efeitos de 2ª ordem proporcionalmente menores. O procedimento plano, conforme colocado na Tabela 6.1, sugere resultados completamente diferentes.

A tabela abaixo apresenta os maiores momentos fletores ocorridos no nível inferior das três primeiras linhas de pilares (de cima para baixo em planta), conforme colocado na Figura 6.4.

Pilar	P (tf)	M_0 (kgf.m)	M_f (kgf.m)
P2 (extremo)	30.74	200.77	379.87
P4 (intermediário)	48.71	200.78	411.01
P6 (central)	48.57	200.78	410.80

Tabela 6.2 – Exemplo 12 – Variação nos momentos fletores ocorridos nos pilares

Nota-se que o efeito de diafragma rígido faz com que os momentos fletores atuantes nos pilares, considerando apenas uma análise de 1ª ordem, sejam praticamente iguais. O fato das cargas axiais serem diferentes faz com que os efeitos de 2ª ordem concentrem-se nos pilares centrais. Esta diferença ocorre, causando momentos finais diferentes em cada pilar, mas em nível muito inferior à diferença que seria obtida analisando cada pórtico equivalente de forma separada. A análise tridimensional contribui, portanto, para a uniformização dos efeitos de 2ª ordem em relação às linhas de pilares.

6.3.2 Pórtico engastado na base

Objetiva-se, agora, verificar a influência do engastamento nas fundações quando aplicado à estrutura tridimensional utilizada no Exemplo 12. Toma-se exatamente a mesma estrutura já apresentada, simplesmente modificando-se a condição de vinculação das fundações. Deseja-se comparar os resultados deste modelo em relação ao anterior e verificar se seu comportamento reproduz a mesma diferença encontrada entre o Exemplo 2 e o Exemplo 3.

Em relação ao eixo X, coincidente com o pórtico equivalente utilizado no Exemplo 3, a análise de 1ª ordem deste modelo fornece os seguintes resultados:

- Deslocamento horizontal no topo: 3.62 cm
- Parâmetro Alfa: $\alpha = 0.60$
- Coeficiente Gama-Z: $\gamma_z = 1.18$

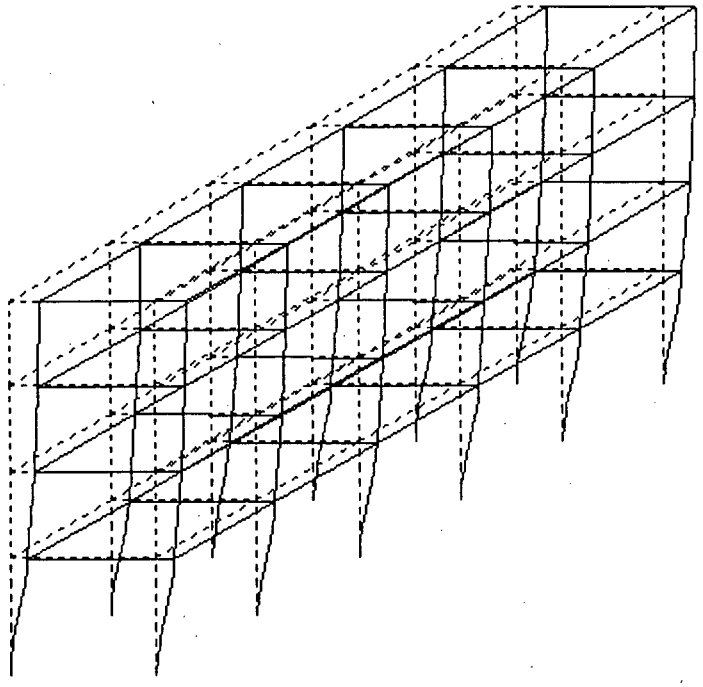


Figura 6.8 – Exemplo 13 – Estrutura deformada

Efetuada a análise de 2ª ordem através do Processo P-Delta, obteve-se um deslocamento final no topo da edificação de 3.82 cm. O deslocamento horizontal apresentado foi o mesmo em todos os pontos, pelo fato do programa utilizar a hipótese de diafragma rígido.

Para comparar os resultados obtidos no modelo 3D com o pórtico equivalente adotado para o Exemplo 2, utilizam-se os resultados obtidos através do Processo P-Delta sem discretização nas barras, obtendo a tabela a seguir:

Modelo	δ_0	δ_f
Plano	4.309	4.782 (Δ=1.109)
Espacial	3.62	3.82 (Δ=1.055)

Tabela 6.3 – Exemplo 13 – Comparação entre os resultados obtidos no caso plano e no tridimensional

Observa-se aqui exatamente o mesmo comportamento do exemplo anterior, sendo a estrutura espacial mais rígida do que o modelo plano equivalente, conforme já comentado no item anterior. As diferenças encontradas neste modelo são menores do que no modelo anterior, apoiado, visto os próprios efeitos de 2ª ordem serem menores.

A tabela abaixo apresenta os maiores momentos fletores ocorridos no nível inferior das três primeiras linhas de pilares (de cima para baixo em planta), da mesma forma como feito no item anterior.

Pilar	P (tf)	M_0 (kgf.m)	M_f (kgf.m)
P2 (extremo)	29.26	3717.90	3923.79
P4 (intermediário)	47.21	3717.90	4018.70
P6 (central)	47.09	3717.91	4019.34

Tabela 6.4 – Exemplo 13 – Variação nos momentos fletores ocorridos nos pilares

Observa-se o mesmo comportamento do modelo anterior, com o efeito de 2ª ordem fazendo com que os momentos nos pilares, inicialmente iguais após a análise de 1ª ordem, concentrem-se nos pilares internos. Todavia, embora a diferença nas cargas axiais seja a mesma do exémplo anterior, nota-se que a diferença obtida após a análise de 2ª ordem é menor, indicando que esta diferença ocorre apenas quando os efeitos de 2ª ordem são grandes. Confirma-se a idéia de que o modelo espacial contribui para uniformizar os efeitos de 2ª ordem pelas linhas de pilares da estrutura.

6.3.3 Pórtico simétrico contendo pilares parede

Objetiva-se, agora, criar um exemplo tridimensional equivalente ao utilizado para o Exemplo 10, no qual foi incluído ao modelo do pórtico resistente um pilar de grandes dimensões.

Esta estrutura é composta por 15 pavimentos iguais em planta, com dimensão de 10.00 x 20.00 m entre eixos, sendo utilizadas 6 linhas contendo 3 pilares cada. São utilizadas as seguintes propriedades:

- Módulo de elasticidade: $3{,}5 \times 10^5$ kgf/cm^2;
- Seção transversal das barras: 13 x 55 cm para as vigas e variando desde 25 x 50 cm até 15 x 40 cm para os pilares, com exceção de dois pilares parede com seção de 15 x 250 cm ao longo de toda a sua prumada;
- Lajes maciças de 10 cm;
- Altura de 300 cm entre os pavimentos, à exceção do primeiro nível, que apresenta 400 cm de altura.

A estrutura desta edificação apresenta-se como na figura a seguir.

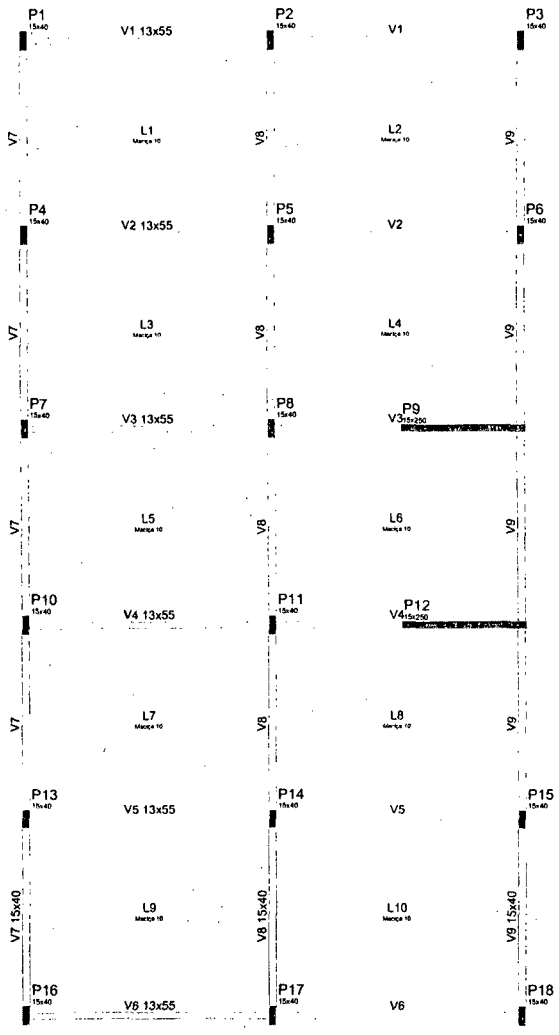


Figura 6.9 – Exemplo 14 – Pórtico tridimensional contendo pilares parede simétricos

Pode-se notar que o modelo criado é simétrico com relação ao eixo X. Com isto, esta estrutura apresenta-se tridimensionalmente como seguinte:

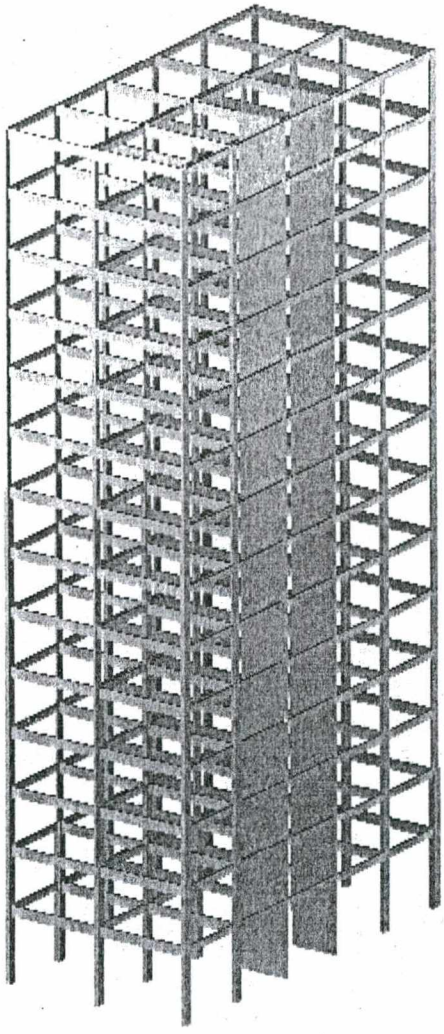


Figura 6.10 – Exemplo 14 – Visualização tridimensional da estrutura

Em relação ao eixo X, coincidente com o pórtico equivalente utilizado no Exemplo 10, a análise de 1ª ordem deste modelo fornece os seguintes resultados:

- Deslocamento horizontal no topo: 18.15 cm
- Parâmetro Alfa: $\alpha = 0.87$
- Coeficiente Gama-Z: $\gamma_z = 1.11$

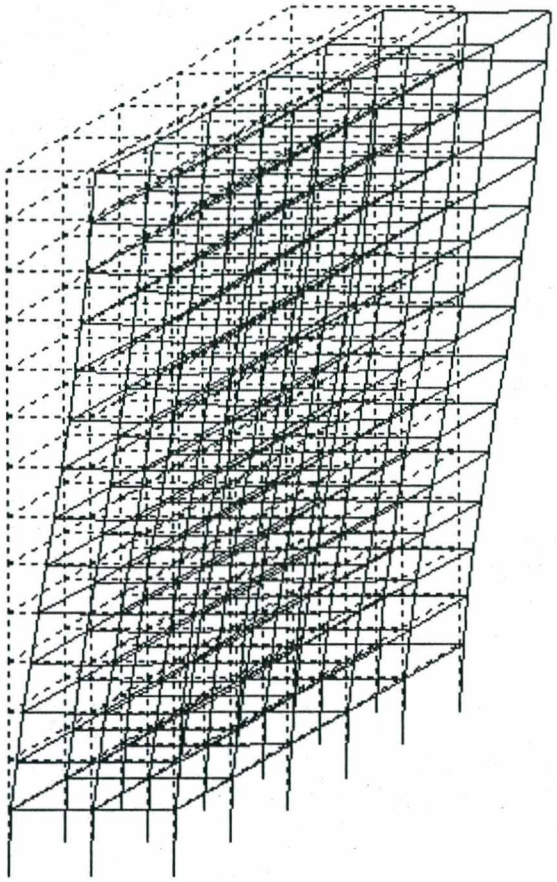


Figura 6.11 – Exemplo 14 – Estrutura deformada

Efetuada a análise de 2ª ordem através do Processo P-Delta, obteve-se um deslocamento final no topo da edificação de 20.42 cm.

Para este exemplo, torna-se interessante verificar a distribuição dos efeitos de 2ª ordem ao longo do modelo espacial. No caso do modelo plano equivalente desenvolvido no item 5.6.7, constatou-se que, embora o pilar parede absorva a maior parte dos efeitos horizontais, as variações percentuais nos demais pilares da edificação são da mesma ordem de grandeza ou, por vezes, maiores. A tabela abaixo apresenta os maiores momentos fletores ocorridos no nível inferior das três primeiras linhas de pilares (de cima para baixo em planta), conforme dispostos na Figura 6.9.

Pilar	P (tf)	M_0 (kgf.m)	M_f (kgf.m)
P3 (extremo)	134.09	2285.35	2476.81 (Δ=8.38%)
P6 (intermediário)	164.46	2280.71	2467.03 (Δ=8.56%)
P9 (pilar parede, central)	300.38	409646.31	444823.59 (Δ=8.59%)

Tabela 6.5 – Exemplo 14 – Variação nos momentos fletores ocorridos nos pilares

Nota-se, conforme esperado, que o pilar parede apresenta momentos fletores muito superiores aos demais pilares. Todavia, as variações percentuais são praticamente as mesmas, evidência que se mostrou também no exemplo plano do item 5.6.7.

6.3.4 Pórtico não simétrico contendo pilares parede

Como exemplo final, deseja-se criar um modelo cujo comportamento tridimensional não possa ser capturado através de modelos planos. Os exemplos utilizados até este ponto possuíam comportamento perfeitamente simétrico em torno do eixo em estudo. Quando isto não ocorre, a aplicação de carregamentos horizontais à estrutura causa deslocamentos não apenas na direção do carregamento, mas também na direção transversal, provocando uma "torção" na estrutura. Objetiva-se verificar a influência deste comportamento na resposta da estrutura aos efeitos de 2ª ordem.

Utiliza-se uma estrutura composta por 15 pavimentos iguais em planta, exatamente igual à adotada para o exemplo anterior, com a exceção de que os pilares parede encontram-se dispostos em um lado da estrutura.

A estrutura desta edificação apresenta-se como na figura a seguir.

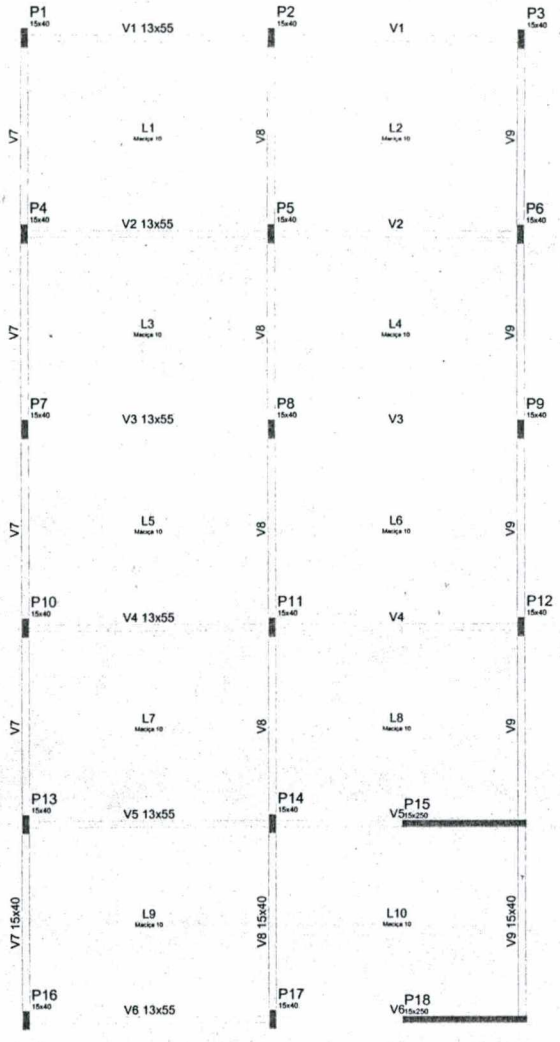


Figura 6.12 – Exemplo 15 – Pórtico tridimensional contendo pilares parede não simétricos

Esta estrutura apresenta-se tridimensionalmente como seguinte:

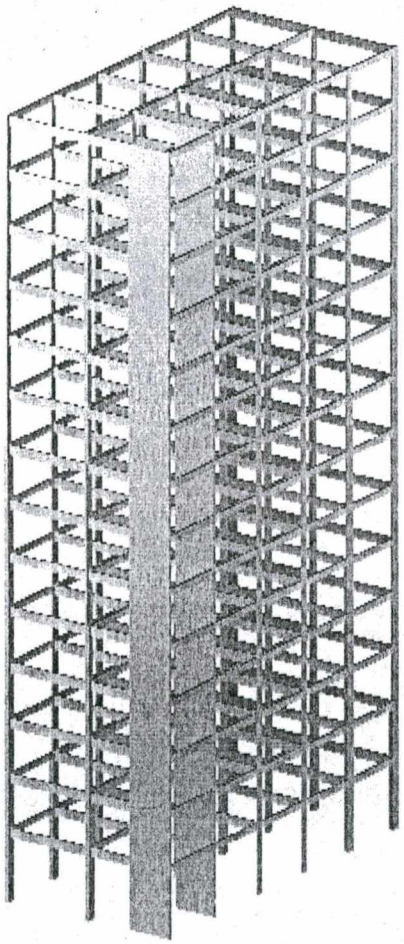


Figura 6.13 – Exemplo 15 – Visualização tridimensional da estrutura

Admite-se esta estrutura submetida apenas a carregamentos segundo o eixo X, coincidente com o pórtico equivalente utilizado no Exemplo 10. A análise de 1ª ordem deste modelo fornece os seguintes resultados, para a direção X:

- Deslocamento horizontal no topo: 20.39 cm
- Parâmetro Alfa: $\alpha = 0.92$
- Coeficiente Gama-Z: $\gamma_z = 1.14$

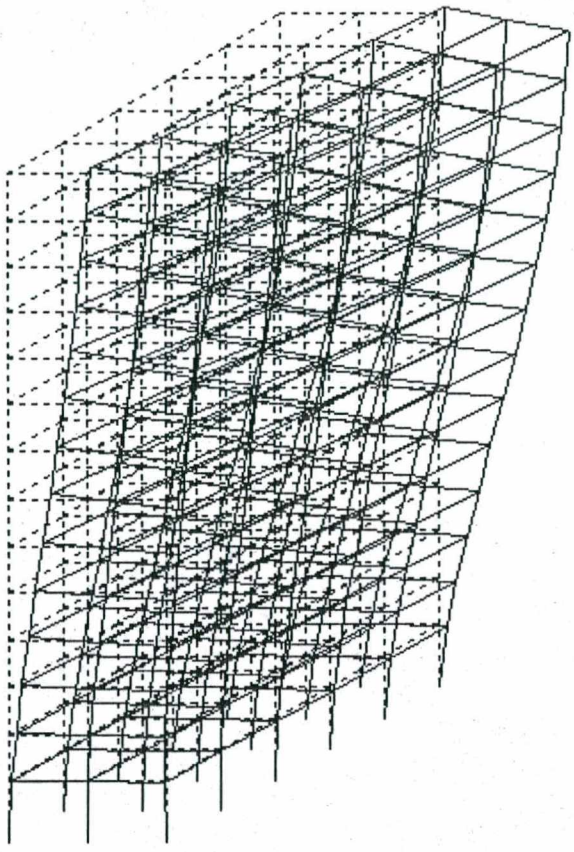


Figura 6.14 – Exemplo 15 – Estrutura deformada

Pode-se notar, mesmo visualmente, como o comportamento desta estrutura não é simétrico. Embora existam apenas carregamentos horizontais segundo a direção X, ocorrem deslocamentos também na direção Y. Efetuada a análise de 2ª ordem através do Processo P-Delta, obteve-se um deslocamento final no topo da edificação, referido ao eixo X, de 23.50 cm.

Para analisar os resultados deste modelo, pode-se selecionar dois parâmetros importantes: o deslocamento horizontal ocorrido no topo e o momento fletor ocorrido na base, para cada um dos pilares componentes da estrutura. A tabela abaixo indica os resultados obtidos para a linha de pilares à direita da edificação, onde estão contidos os pilares parede (vide Figura 6.12).

Pilar	Deslocamento X no topo (cm)		Momento fletor na base (kgf.m)	
	1ª	1ª + 2ª	1ª	1ª + 2ª
P3	26.50	31.14 (Δ=17.5%)	9793.09	11168.92 (Δ=14.1%)
P6	23.63	27.66 (Δ=17.1%)	8006.51	9084.74 (Δ=13.5%)
P9	20.75	24.17 (Δ=16.5%)	1395.98	1547.08 (Δ=10.8%)
P12	17.87	20.68 (Δ=15.7%)	1016.78	1121.95 (Δ=10.3%)
P15	14.99	17.19 (Δ=14.7%)	517816.69	578774.75 (Δ=11.8%)
P18	12.11	13.70 (Δ=13.1%)	116644.97	116724.71 (Δ=0.1%)

Tabela 6.6 – Exemplo 15 – Variação nos momentos fletores ocorridos nos pilares

Nota-se, em primeiro lugar, que a variação nos efeitos de 2ª ordem é menos uniforme do que no modelo simétrico equivalente, mas sem apresentar diferenças excepcionais. Exceção se faz ao P18, que apresentou variação praticamente nula nos valores de momento fletores entre a análise de 1ª e 2ª ordem. Isto pode ser explicado por um "efeito alavanca" tridimensional, que faz com que este absorva menos esforços horizontais do que o pilar P15. A título de comparação, deve-se dizer que a reação horizontal ocorrida na base do pilar P18 foi de apenas 29.6 tf, pequena em comparação com a reação apresentada pelo pilar P15, de 103.2 tf.

Pode-se dizer, a partir deste exemplo, que os efeitos de 2ª ordem apenas aumentam os efeitos de 1ª ordem, inclusive seus efeitos tridimensionais, sem alterar o comportamento global da estrutura. Caso os esforços horizontais provoquem uma rotação da estrutura em torno de seu eixo (como é o caso deste exemplo), o efeito de 2ª ordem simplesmente fará esta rotação maior, de forma proporcional à rigidez da estrutura em cada direção. A modificação nos esforços segue apenas as características de rigidez já abordadas.

O diagrama a seguir ilustra o comportamento da estrutura quando sujeito a cargas horizontais. Representam-se os deslocamentos horizontais ocorridos no topo da edificação, para as análises de 1ª e 2ª ordens.

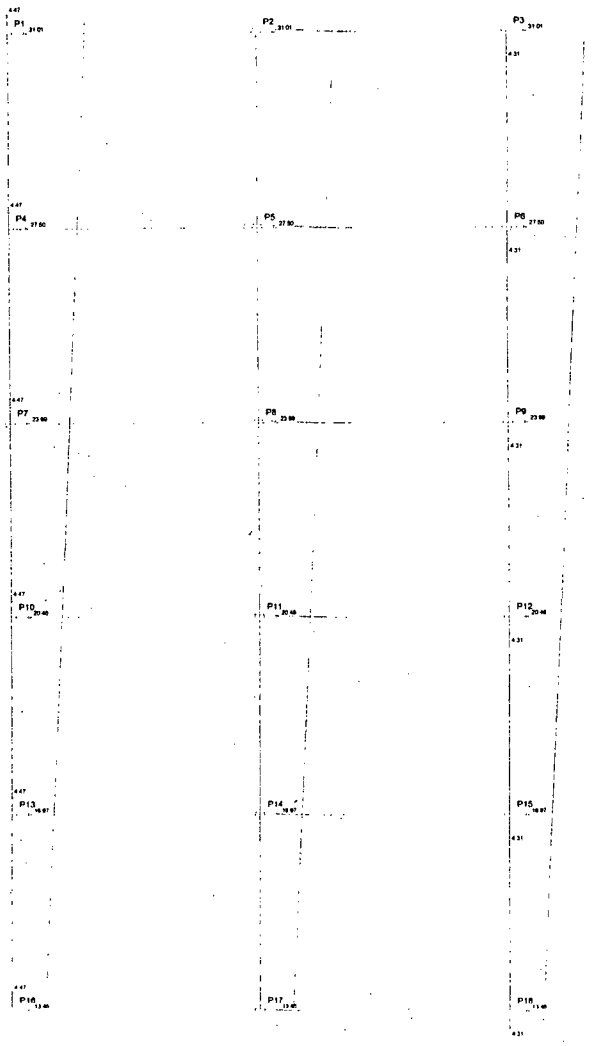


Figura 6.15 – Exemplo 15 – Deslocamentos horizontais no topo da edificação

6.3.5 Influência da rigidez à torção dos pilares

A questão da análise tridimensional de uma estrutura está intimamente ligada aos efeitos de torção da estrutura como um todo. Sempre que não existir um eixo de simetria, aplicável simultaneamente à estrutura e aos carregamentos, como no exemplo anterior, os carregamentos horizontais provocarão um giro na estrutura.

Um dado adicional existente na análise espacial que pode estar vinculado a este problema é da rigidez à torção das barras. Em um exemplo simples, contendo apenas um pilar e uma barra horizontal, seria simples verificar que a rigidez à torção determina os deslocamentos horizontais ocorridos em seu topo. Quanto menor a rigidez, maiores os deslocamentos e, portanto, maiores os efeitos de 2ª ordem.

Deseja-se verificar a influência deste fator em uma estrutura usual de edificação. Para tal, será utilizada a mesma estrutura do Exemplo 15, mas considerando-se uma rigidez à torção igual a 25% da rigidez original.

A tabela a seguir compara os deslocamentos obtidos no topo da edificação para as duas situações.

Pilar	Rigidez 100%		Rigidez 25%	
	1ª	1ª + 2ª	1ª	1ª + 2ª
P3	26.50	31.14 (Δ=17.51%)	26.57	31.24 (Δ=17.58%)
P6	23.63	27.66 (Δ=17.05%)	23.67	27.72 (Δ=17.11%)
P9	20.75	24.17 (Δ=16.48%)	20.77	24.20 (Δ=16.51%)
P12	17.87	20.68 (Δ=15.72%)	17.87	20.69 (Δ=15.78%)
P15	14.99	17.19 (Δ=14.68%)	14.97	17.17 (Δ=14.70%)
P18	12.11	13.70 (Δ=13.13%)	12.07	13.65 (Δ=13.09%)

Tabela 6.7 – Exemplo 15 – Variação nos deslocamentos com a redução da torção

Pode-se observar, através destes valores:

- A influência da rigidez à torção dos pilares neste exemplo (e nas edificações usuais) é muito pequena, ficando a resistência ao giro da estrutura como um todo a cargo da interação entre os pórticos transversais.
- Embora o efeito seja muito pequeno, pode-se observar que a redução da rigidez à torção causa aumento na tendência de giro da edificação. Em uma situação onde a rigidez fosse dada preponderantemente por um único elemento à torção, como, por exemplo, uma caixa de elevador, este efeito poderia ser mais relevante.
- Os efeitos de 2ª ordem apenas acompanham os de 1ª ordem e também apresentam uma tendência a ampliar o giro da edificação. Isto só se mostrará relevante, todavia, nas estruturas onde a determinação da rigidez à torção for importante para a própria análise linear.

6.4 - ESTADOS LIMITES

Existem dois estados limites a considerar no dimensionamento de uma estrutura:

- Estados limites últimos (ELU): "aqueles relacionados ao colapso, ou a qualquer forma de ruína estrutural, que determine a paralisação do uso da estrutura".
- Estados limites de serviço (ELS): "aqueles relacionados à durabilidade das estruturas, aparência, conforto do usuário e boa utilização funcional da mesma, seja em relação aos usuários, seja às máquinas e aos equipamentos utilizados".

No caso dos estados limites de serviço, trabalha-se com os valores característicos das ações. Os valores característicos F_k correspondem ao quantil de 5% na distribuição normal, considerado na situação mais desfavorável. Da mesma forma, utilizam-se os valores característicos f_k das resistências.

Já no caso dos estados limites últimos, utilizam-se os valores de cálculo das ações, obtidos a partir dos valores característicos, multiplicando-os pelos respectivos coeficientes de ponderação γ_f.. De forma análoga, as resistências de cálculo f_d são obtidas dividindo-se os valores característicos f_k pelo coeficiente de ponderação γ_m das resistências.

Na análise linear de 1ª ordem, é indiferente o fato de se obter os esforços internos (a partir das ações) característicos ou de cálculo, uma vez que o princípio da superposição das ações garante a linearidade dos valores. É usual fazer uma única análise, utilizando-se os valores característicos, majorando-se diretamente os resultados finais por γ_f para se obter os valores de cálculo.

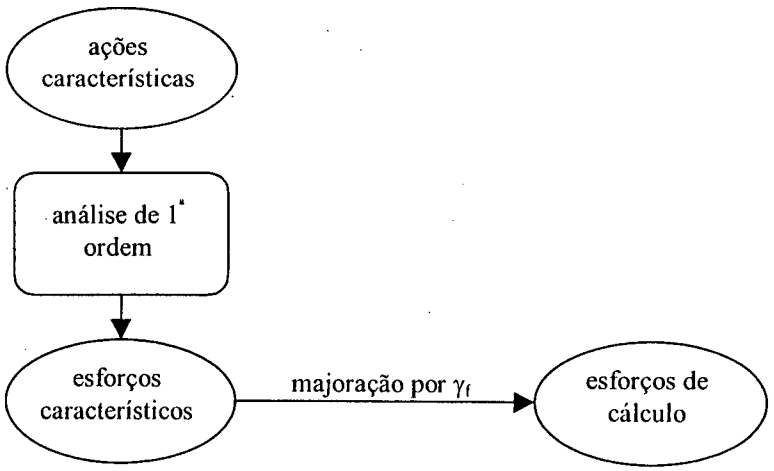


Figura 6.16 – Estados limites na análise de 1ª ordem

No caso da análise de 2ª ordem, a relação entre os deslocamentos e a carga axial não é linear, o que invalida o princípio da superposição das ações. Desta forma, para se obter os esforços finais de cálculo, é necessário fazer a análise utilizando-se as ações de cálculo. No caso de se considerarem os estados limites de serviço, devem ser utilizadas as ações características. A diferença entre os esforços característicos e de cálculo não pode ser obtida, portanto, pela simples majoração dos primeiros.

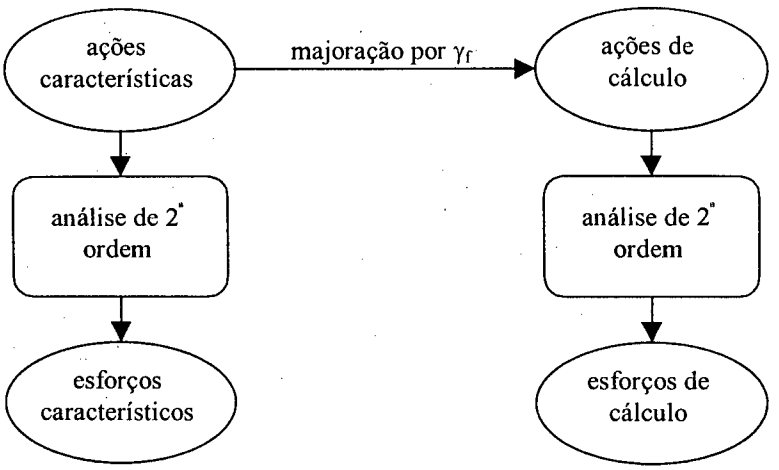


Figura 6.17 – Estados limites na análise de 2ª ordem

Pode-se observar isto voltando-se ao Exemplo 1: considera-se uma barra biapoiada, sujeita a uma força normal compressiva e momentos fletores aplicados em suas extremidades.

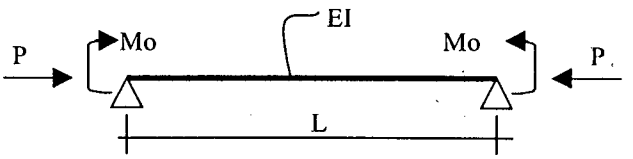


Figura 3.7 – Exemplo 1 – Viga-coluna biapoiada

Conforme exposto na Eq. (3-2), a solução analítica deste exemplo fornece o seguinte momento final no meio do vão:

$$M_f = M_0 \cdot \sec\left(\frac{L}{2}\sqrt{\frac{P}{EI}}\right) \qquad \text{Eq. (3-2)}$$

Considerando-se os valores indicados como sendo os valores característicos, a simples majoração do resultado por γ_f (como feito na análise de 1ª ordem) forneceria:

$$M_{fd} = \gamma_f \cdot \left[M_0 \cdot \sec\left(\frac{L}{2} \sqrt{\frac{P}{EI}} \right) \right] \qquad \text{Eq. (6-137)}$$

Por outro lado, ao considerar-se na análise os esforços de cálculo já majorados por γ_f, chega-se a:

$$M_{fd} = (\gamma_f \cdot M_0) \cdot \sec\left(\frac{L}{2} \sqrt{\frac{(\gamma_f \cdot P)}{EI}} \right) \qquad \text{Eq. (6-138)}$$

Pode-se notar que o resultado obtido através da Eq. (6-138) é diferente daquele que seria encontrado através da Eq. (6-137). Verificam-se dois pontos:

- A majoração do valor de M_0 é linear, ou seja, pode ser feita tanto sobre o momento M_0 aplicado quanto sobre o momento resultante, obtendo-se os mesmos valores;
- A majoração da carga P provoca aumento não linear no momento final, devendo ser obrigatoriamente feita na carga aplicada, obtendo-se a partir da carga já majorada o momento final.

Os mesmos resultados podem ser observados através dos processos numéricos apresentados. Voltando-se ao Exemplo 2 (pórtico apoiado na base) e considerando que:

- os valores apresentados na Figura 5.1 são os valores característicos;
- o coeficiente de majoração γ_f é igual a 1.4.

Processo	δ_{fk} (cm)	$\gamma_f.\delta_{fk}$ (cm)	δ_{fd} (cm)
P-Delta	21.401	29.961	48.694
K_G	25.827	36.156	83.357
Funções de estabilidade	25.964	36.350	86.254

Tabela 6.8 – Exemplo 2 – Análise considerando valores de cálculo

Pode-se observar que o aumento da força normal na análise aumenta o próprio efeito de 2ª ordem, podendo chegar a diferenças bastante grandes em relação ao processo convencional de majorar os esforços após o cálculo.

Uma forma simplificada de abordar o problema seria:

- obter inicialmente os esforços relativos às cargas verticais;

- obter a partir destes os esforços axiais de cálculo em cada barra, considerando-os constantes;
- analisar a estrutura sujeita aos carregamentos horizontais, utilizando-se, na definição dos efeitos de 2ª ordem, as cargas axiais já estabelecidas;
- combinar os resultados obtidos com as cargas verticais e horizontais.

Processo	δ_{fk} (cm)	$\gamma_f.\delta_{fk}$ (cm)	δ_{fd} (cm)
P-Delta	34.781	48.693	48.694
K_G	59.301	83.021	83.357
Funções de estabilidade	61.087	85.522	86.254

Tabela 6.9 – Exemplo 2 – Análise considerando valores de cálculo (processo alternativo)

Pode-se observar que a variação obtida é muito pequena, causada apenas pelas cargas axiais induzidas pelas cargas horizontais, cujo valor pode ser usualmente desprezado.

6.5 - COMBINAÇÕES DE AÇÕES

Outro aspecto a considerar, também proveniente do fato de que a não linearidade faz com que não se possa utilizar a superposição das ações, diz respeito às combinações de ações. Conforme MARTINS & STUCCHI (1993), "um carregamento é definido pela combinação das ações que têm probabilidade não desprezível de atuarem simultaneamente sobre a estrutura, durante um período preestabelecido. Estas combinações devem ser feitas de diferentes maneiras, de forma que possam ser determinados os efeitos mais desfavoráveis para a estrutura".

Em situações reais de projeto, estão presentes ao menos três casos de carregamento:

- carga permanente: composta pelo peso próprio de todas as partes que compõem a edificação (estrutura, alvenaria, revestimentos, etc.);
- carga acidental: carga variável proveniente da utilização da estrutura, determinada sob forma semi-probabilística;
- carga de vento: cargas estáticas equivalentes à ação do vento, dividindo-se em carregamentos distintos conforme a direção e sentido de aplicação da carga.

Podem ocorrer ainda diversas outras ações, como, por exemplo, variações de temperatura, recalques de apoio, empuxos de terra, impactos, entre outros.

Na análise de 1ª ordem, cada uma das ações implica em esforços internos correspondentes, podendo ser calculados isoladamente. A linearidade entre ações e deslocamentos permite que as combinações possam ser feitas diretamente sobre os esforços ao invés de sobre as ações.

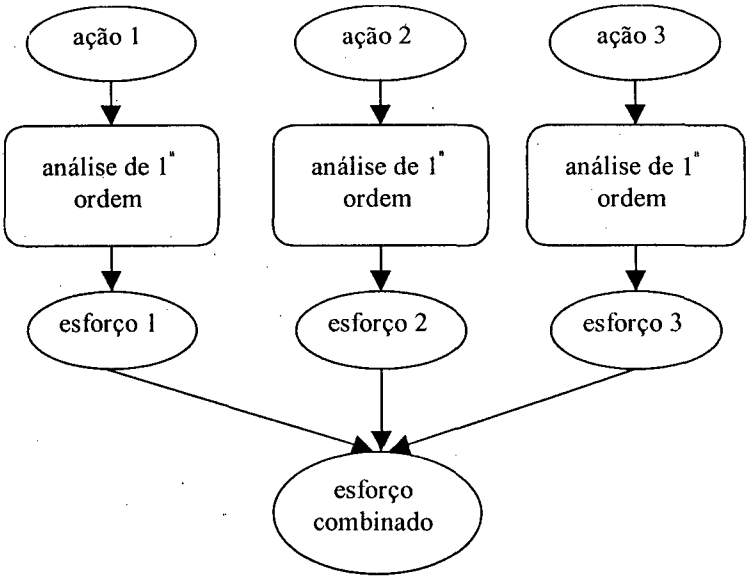


Figura 6.18 – Combinações de ações na análise de 1ª ordem

Na análise de 2ª ordem, as ações devem ser combinadas e cada combinação deve ser analisada separadamente. Conforme exposto no item anterior, também é necessário diferenciar as combinações últimas das combinações de serviço, analisando-as separadamente.

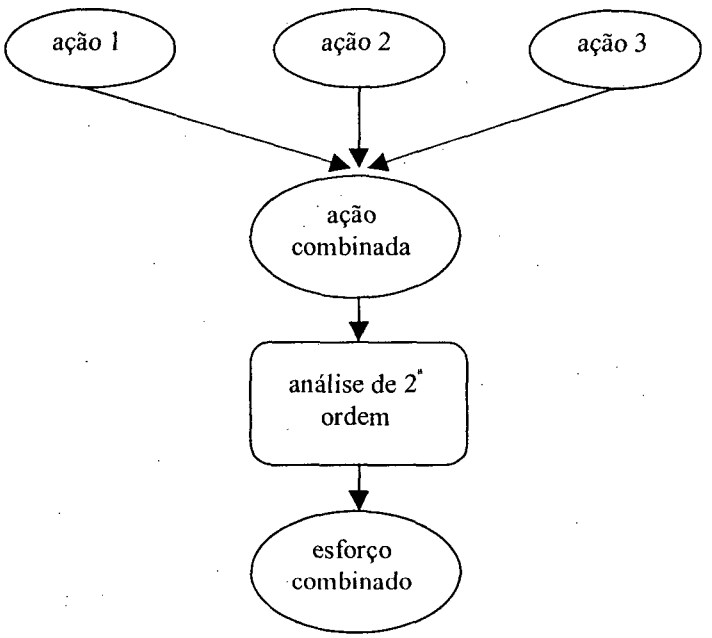


Figura 6.19 – Combinações de ações na análise de 2ª ordem

Dependendo do número de ações consideradas, o número de combinações obtido pode ser bastante grande. Na análise de 1ª ordem, um novo caso de carregamento implica apenas em uma nova análise a efetuar, enquanto que, na análise de 2ª ordem, isto pode implicar em grande número de combinações com os demais casos existentes, aumentando rapidamente o custo de processamento.

6.6 - NÃO LINEARIDADE FÍSICA

Conforme exposto no item 2.4.1, as estruturas de concreto armado não respeitam a Lei de Hooke, ou seja, a relação entre as tensões e as deformações não é linear. Esta característica, chamada *não linearidade física*, pode afetar consideravelmente os resultados obtidos em relação a uma análise linear. Basicamente, pode-se considerar que existe uma redução na rigidez da estrutura (produto de rigidez EI das barras). Adotando-se a abordagem simplificada de considerar um redução constante φ na rigidez das barras, pode-se constatar que, a partir da análise de 1ª ordem:

- os deslocamentos seriam divididos pelo fator φ (com $\varphi < 1$);
- os esforços internos manteriam-se inalterados.

Esta consideração é para qualquer valor de φ, desde que seja aplicado o mesmo valor em toda a estrutura.

No caso da análise de 2ª ordem, a redução na rigidez das peças provoca um aumento nos efeitos de 2ª ordem. Pode-se observar isto voltando-se ao Exemplo 1: considera-se uma barra biapoiada, sujeita a uma força normal compressiva e momentos fletores aplicados em suas extremidades.

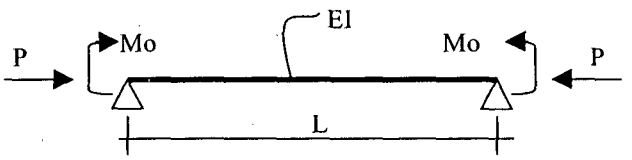


Figura 3.7 – Exemplo 1 – Viga-coluna biapoiada

Conforme exposto na Eq. (3-2), a solução analítica deste exemplo fornece o seguinte momento final no meio do vão:

$$M_f = M_0 \cdot \sec\left(\frac{L}{2}\sqrt{\frac{P}{EI}}\right) \qquad \text{Eq. (3-2)}$$

Ao considerar-se na análise a redução da rigidez pelo fator φ, chega-se a:

$$M_f = M_0 \cdot \sec\left(\frac{L}{2}\sqrt{\frac{P}{\varphi \cdot EI}}\right) \qquad \text{Eq. (6-139)}$$

Pode-se notar que o resultado obtido através do Eq. (6-139) é diferente daquele que seria encontrado através da Eq. (3-2). A redução na rigidez influencia tanto os deslocamentos como os esforços resultantes.

Segundo SANTOS & FRANCO (1993), a nova NBR 6118 deve permitir a consideração da não linearidade física de maneira aproximada na obtenção dos esforços globais de 2ª ordem, tomando-se para rigidez das peças os seguintes valores:

- vigas: $EI_{sec} = 0.5\ EI$;
- pilares: $EI_{sec} = 0.8\ EI$.

Alternativamente, quando a subestrutura de contraventamento for composta exclusivamente por vigas e pilares, permitir-se-á considerar:

- vigas: $EI_{sec} = 0.7\ EI$;
- pilares: $EI_{sec} = 0.7\ EI$.

Chamando-se as duas alternativas de Critério 1 e Critério 2, respectivamente, pode-se voltar ao Exemplo 2 (pórtico apoiado na base) e verificar a influência desta consideração no deslocamento final obtido no topo da estrutura:

Processo	Sem considerar NLF	Critério 1	Critério 2
P-Delta	21.401	39.948	52.060
K_G	25.827	56.821	94.426
Funções de estabilidade	25.964	57.734	98.513

Tabela 6.10 – Exemplo 2 – Análise considerando a não linearidade física de maneira aproximada.

Neste exemplo, visto serem grandes os efeitos de 2ª ordem, pode-se observar facilmente a influência da não linearidade física sobre os resultados obtidos considerando-se a não linearidade geométrica. Neste caso, que pode ser considerado crítico, a simples escolha entre dois processos aproximados sugeridos resulta em diferença inaceitável nos resultados obtidos. Deve-se lembrar, porém, que a redução na rigidez das peças é algo muito mais complexo, dependendo de fatores que vão desde o nível de carga normal até a própria armação da peça, tornando difícil a consideração mais exata da não linearidade física em termos de Projeto.

CAPÍTULO 7.
CONCLUSÕES E RECOMENDAÇÕES

7.1 - COMPARAÇÃO ENTRE OS PROCESSOS

O Processo P-Delta representa uma abordagem dada pela Engenharia ao problema da não linearidade geométrica. Em sua versão original, o processo destina-se à aplicação às barras verticais de uma edificação (pilares), capturando com isto o efeito P-Δ global. Desta forma, este processo não pode, por exemplo, capturar efeitos não lineares locais de uma barra simplesmente apoiada sujeita a uma força compressiva. Mostrou-se, porém, que o processo pode ser estendido não apenas a estruturas quaisquer, como também pode ser utilizado para capturar os efeitos P-δ locais às barras, mediante subdivisão destas em partes menores. Seus resultados, portanto, aproximam-se dos resultados analíticos conforme aumenta a discretização.

O Processo da Matriz de Rigidez Geométrica corresponde à inclusão do efeito não linear geométrico na obtenção da matriz de rigidez de uma barra da forma mais imediata, simplesmente modificando a formulação mais conhecida do elemento de viga que é obtida através do Método dos Elementos Finitos, através da consideração da influência da força axial de forma linear. Para isto, utilizam-se simples relações de equilíbrio. Esta formulação, embora muito mais precisa que a do Processo P-Delta, não corresponde diretamente à solução analítica, por se considerar linear a influência do esforço normal. Através da subdivisão da barra, tende-se à solução analítica. O Processo da Matriz de Rigidez Geométrica Modificado utiliza exatamente a mesma formulação, mas utilizando um esquema de solução do problema não linear similar ao do P-Delta.

O fator preponderante que determina a necessidade de discretização, quando se considera a análise de um elemento isolado, em ambos os processos, é o nível de força axial atuante. Quanto mais próximo da carga crítica da barra, maior a diferença entre o valor obtido e o valor analítico e, consequentemente, maior a necessidade de discretização. Mostrou-se, porém, que o Processo P-Delta é muito mais sensível ao aumento da força axial, apresentando erros significativos mesmo a 50% da carga crítica, enquanto que os processos K_G e K_{GM} só começam a apresentar diferenças relevantes a partir dos 90% da carga crítica.

Verificou-se também que estes últimos possuem muito mais eficiência à discretização, com grande melhoria nos resultados mesmo com apenas duas divisões, enquanto que o P-Delta converge de forma relativamente lenta para o resultado analítico conforme é aumentado o número de subdivisões da barra.

O Processo das Funções de Estabilidade, por outro lado, corresponde à própria solução analítica fornecida pela Resistência dos Materiais, pelo fato de basear a dedução da matriz de rigidez na solução da equação diferencial da viga. Deve-se lembrar que esta solução chamada analítica e referenciada ao longo do texto como "exata" também se baseia em uma forma reduzida da equação diferencial da viga, considerando-se apenas pequenas rotações. Para o tipo de problema de interesse da Engenharia Civil, contudo, pode praticamente ser tratado como a solução exata. Verificou-se o fato de que a solução obtida por intermédio deste processo independe do nível de discretização, ou seja, permite obter diretamente a solução analítica sem a necessidade de subdividir as barras.

Mostrou-se que os três processos permitem modelar uma estrutura até sua carga crítica, caracterizando-se este limite por uma não convergência nas iterações no caso do Processo P-Delta e pela singularidade na matriz de rigidez da estrutura para os outros dois processos. Todavia, apenas o Processo das Funções de Estabilidade permite obter o mesmo valor correspondente à carga crítica de Euler, ficando as outras duas soluções sempre abaixo da solução analítica, ou seja, estas conduzem a deslocamentos sempre menores que a solução analítica e apresentam uma carga crítica sempre superior à carga crítica teórica. Mesmo o Processo da Matriz de Rigidez Geométrica não se mostrou adequado para obtenção da carga crítica de uma estrutura, tendo chegado a um erro superior a 20% quando aplicado a uma barra isolada. O Processo P-Delta mostrou-se absolutamente inadequado para análise de estruturas próximas à carga crítica. Ambos os processos podem ser aplicados apenas se for feita uma discretização adequada na estrutura.

Outro ponto a destacar é a necessidade, característica à metodologia adotada, de resolver iterativamente a estrutura sob forma linear para obter a solução não linear. Verificou-se que o número de iterações necessárias aumenta conforme aumentam os efeitos de 2ª ordem (acompanhando o aumento da carga axial, por exemplo). A quantidade de iterações necessárias é similarmente pequena nos processos da Matriz de Rigidez Geométrica e das Funções de Estabilidade, enquanto que pode ser bastante grande no caso do Processo P-Delta e do Processo da Matriz de Rigidez Geométrica Modificado. Para estes dois últimos, a convergência dos resultados é relativamente lenta, podendo-se obter, mesmo após uma precisão mínima preestabelecida em relação à iteração anterior, um erro acumulado superior a esta precisão. Verifica-se também estes apresentam uma quantidade de iterações necessárias gradativamente maior conforme é aumentado o nível de discretização nas barras.

Observou-se, em todos os exemplos numéricos apresentados, que, para as condições de análise usuais, os processos K_G e K_{GM} fornecem resultados praticamente iguais aos do Processo das Funções de Estabilidade, com erros normalmente inferiores a 1% mesmo sem qualquer subdivisão nas barras. Concluiu-se que, da mesma forma como levantado por outros autores, estes dois processos podem ser usados indistintamente para fins de projeto, ficando o Processo das Funções de Estabilidade necessário apenas para problemas de obtenção da carga crítica, a não ser que se estude a discretização necessária.

Já o Processo P-Delta apresentou erros significativos mesmo a nível de projeto, chegando a diferenças próximas a 20%, contrárias à segurança, no exemplo com as fundações apoiadas onde as barras não estavam subdivididas. Verificou-se que estes erros são relevantes quando os efeitos de 2ª ordem são grandes, o que sugere a adoção de um limite superior nos efeitos de 2ª ordem obtidos, acima do qual seria obrigatória a subdivisão das barras, a fim de melhorar a precisão dos resultados.

7.2 APLICABILIDADE PRÁTICA

Quanto à implementação computacional, o Processo P-Delta apresenta a grande vantagem de sua simplicidade, com uma metodologia única independentemente da formulação e das condições de vinculação da barra. Pode-se até mesmo utilizar este processo sob forma relativamente manual, incluindo as forças horizontais fictícias diretamente sobre uma análise linear. Nos outros processos, é necessária a modificação da matriz de rigidez, de forma bastante semelhante e relativamente direta em termos de programação. Todavia, a matriz K_G pode ser considerada de formulação mais simples do que as funções de estabilidade, o que simplifica as extensões à sua formulação.

Quanto à aplicação em projeto, por outro lado, deve-se levar em conta o tempo necessário (custo computacional) para se obter uma solução por cada processo. O Processo P-Delta, embora necessite de um número maior de iterações do que o K_G, baseia-se na modificação do vetor de forças aplicado a uma estrutura, enquanto que os outros dois modificam a própria matriz de rigidez. No primeiro caso, não é necessário fazer novamente a inversão da matriz de rigidez da estrutura, efetuando iterativamente apenas multiplicações matriciais, enquanto que, no segundo caso, é necessário repetir iterativamente todo o processo. Uma vez que o tempo necessário para a inversão da matriz é muito maior do que aquele necessário para uma multiplicação, com esta diferença aumentando exponencialmente com o aumento no número de graus de liberdade da estrutura, o uso do Processo P-Delta torna-se mais interessante quanto maior a estrutura a analisar.

O processo que reúne as vantagens de P-Delta e K_G é o Processo da Matriz de Rigidez Geométrica Modificado, uma vez que este apresenta as mesmas características de solução do P-Delta e os mesmos resultados do K_G. A escolha entre K_G e K_{GM} fica, portanto, relativa apenas ao custo computacional obtido caso a caso. Quanto ao P-Delta, pode ser utilizado apenas para níveis pequenos de efeito de 2ª ordem (observaram-se bons resultados até uma proporção da ordem de 20% dos efeitos de 2ª ordem sobre os de 1ª ordem).

7.3 - ASPECTOS DE PROJETO

A aplicação dos conceitos de não linearidade geométrica em projetos representa grande avanço em direção à obtenção de modelos mais representativos, que possam estimar a segurança da estrutura com menor grau de erro. O atual estágio das ferramentas computacionais já permite que tais análises sejam aplicadas a uma estrutura como um todo. Neste panorama, este trabalho teve como um dos seus objetivos mostrar a aplicação dos processos para inclusão da não linearidade geométrica em modelos práticos, apontando cuidados a serem tomados em projetos reais.

7.3.1 Aspectos a considerar

Mesmo na análise das estruturas sob forma linear, existem considerações diversas a serem feitas de forma a adequar o modelo à estrutura em análise e a prever situações diversas de geometria e carregamento. Desta forma, a formulação apresentada no CAPÍTULO 3 não cobre todas as situações da prática, sendo necessárias diversas modificações à matriz de rigidez e aos vetores de carregamentos. Pode-se citar aqui a necessidade de liberar vinculações internas das barras (normalmente, de forma a considerar ligações livres ao giro), a consideração de ligações semi rígidas entre os elementos, o uso de fundações elásticas e a consideração dos efeitos da dilatação térmica, entre outros.

As modificações na formulação devem ser analisadas para verificar sua influência nos efeitos de 2ª ordem. No caso de se adotar o Processo P-Delta, não são necessárias modificações, visto ser um processo baseado apenas em relações de equilíbrio. Analisando a necessidade de verificar a formulação da matriz de rigidez e dos vetores de forças, a utilização do Processo da Matriz de Rigidez Geométrica apresenta vantagem sobre o Processo das Funções de Estabilidade, visto sua formulação, mais simples, necessitar de poucas modificações, sendo estas normalmente feitas sem grande dificuldade.

Outro ponto a destacar é a consideração da não linearidade física. Conforme exposto, este assunto é extenso e ainda não totalmente resolvido, não sendo objeto do presente trabalho. A grande variabilidade obtida quanto ao comportamento do material concreto armado sugere a adoção de procedimentos simplificados como os sugeridos na proposta de revisão da NBR 6118. O que se mostrou é que a rigidez à flexão da estrutura influi significativamente nos efeitos de 2ª ordem, sendo estes crescentes conforme a rigidez da estrutura é reduzida. Na falta de valores mais exatos, destaca-se apenas a importância de se fazer a redução no módulo de rigidez, mesmo sob forma simplificada.

7.3.2 A questão do Coeficiente Gama-Z

Ao longo de diversos exemplos numéricos apresentados no CAPÍTULO 5, procurou-se também fazer uma comparação entre os resultados fornecidos por cada processo e destes com os parâmetros simplificados mais conhecidos, a saber, o Parâmetro de Instabilidade Alfa e o Coeficiente Gama-Z. Ambos destacam-se por estarem presentes na proposta de revisão da NBR 6118.

Uma primeira análise indica que, na necessidade de um parâmetro simplificado, mostrou-se mais interessante a utilização do Gama-Z do que a do Alfa. Por um lado, a formulação do Alfa é mais pobre, associando a estrutura com um pilar engastado-livre, comportamento expressivamente distante do real. Isto causa erros maiores, especialmente quando existem variações na rigidez ao longo da altura da edificação. Por outro lado, o resultado que este fornece apenas correlaciona um limite acima do qual os efeitos de 2ª ordem devem ser considerados, sendo de pouca valia para determinar o quanto se está distante do limite, enquanto que o Gama-Z procura determinar exatamente a magnitude dos efeitos de 2ª ordem. No restante do trabalho, procurou-se verificar as situações onde o Coeficiente Gama-Z fornece resultados confiáveis ou não, desconsiderando o uso do Parâmetro Alfa.

Observou-se que o Coeficiente Gama-Z fornece resultados bastante próximos aos reais quando é utilizado para determinar a relação entre os deslocamentos de 1ª e 2ª ordem. Nos exemplos onde a variação de rigidez ao longo dos pavimentos não é grande, a precisão pode ser considerada plenamente aceitável. Diferenças realmente grandes foram obtidas apenas nos exemplos onde as fundações estavam rotuladas, ou seja, onde a rigidez do lance inferior era muito menor do que a dos lances superiores, concentrando neste os efeitos de 2ª ordem. A redução da rigidez em direção ao topo da edificação, comum na maioria dos projetos, não gerou diferenças significativas.

Todavia, a questão mais importante não é a dos deslocamentos, mais sim a variação dos esforços internos atuantes nas barras. O objetivo do Coeficiente Gama-Z, conforme sua definição, é justamente indicar a variação nos esforços com a consideração dos efeitos de 2ª ordem. Analisando-se a modificação nos esforços, para qualquer exemplo, por mais regular que seja, verifica-se que a modificação não é constante ao longo das barras nem apresenta-se da mesma maneira em todos os esforços. Nos exemplos bastante uniformes, observou-se que o valor do Coeficiente Gama-Z representa basicamente a média da variação dos momentos fletores nos pilares, sendo este o único esforço sobre o qual seria justificável fazer a majoração por Gama-Z. Todavia, ocorrem modificações também nos esforços cortantes, sendo estas relacionadas ao equilíbrio dos esforços e não diretamente ao Gama-Z. Outro ponto a considerar é a variação dos esforços axiais nas barras, normalmente pequena, mas que pode ser importante especialmente nos pórticos resistentes compostos por dois pilares.

A variação nos esforços internos estende-se também às vigas da estrutura, mas de forma que dificilmente pode ser relacionada ao Gama-Z. Conclui-se rapidamente que não é possível nem majorar todos os esforços por Gama-Z nem majorar apenas os momentos nos pilares. Um procedimento que poderia ser adotado seria o de majorar os momentos fletores nos pilares por Gama-Z e obter a variação nos demais esforços, incluindo as vigas, por equilíbrio. A dificuldade material relativa a tal processo já se mostra injustificável, podendo facilmente ser estendida para a utilização do Processo P-Delta.

Além disto, este comportamento médio pode ser identificado apenas nas estruturas bastante regulares. Uma situação crítica está nas estruturas que concentram os efeitos de 2ª ordem em seu lance inferior ou em outra parte desta, comportamento que não pode ser reproduzido pelo Gama-Z.

Em termos de projeto, não se pode considerar aceitável o uso do Gama-Z como um comportamento médio. Isto porque uma média pressupõe diretamente que alguns pontos apresentam efeitos de 2ª ordem abaixo desta média e outros acima delas. O que se observa é que sempre existem pontos da estrutura onde os efeitos de 2ª ordem são expressivamente maiores que o valor obtido pelo Coeficiente Gama-Z e estes são usualmente os pilares sujeitos a maior nível de força normal e maiores momentos fletores, onde os esforços adicionais são mais críticos. Os efeitos de 2ª ordem menores ocorrem justamente nos pilares sujeitos a menores esforços.

Concluindo, deseja-se estimular o uso de processos menos simplificados, que se mostraram, no decorrer deste trabalho, bastante acessíveis. No uso do critério usual de limitar os efeitos de 2ª ordem a 10% dos efeitos de 1ª ordem através do Gama-Z, deve-se lembrar que, na verdade, os efeitos serão sempre superiores a isto em parte da estrutura, especialmente se a rigidez dos lances inferiores for pequena. O uso do Coeficiente Gama-Z para obter os esforços finais em uma faixa de valores acima dos 10%, mesmo limitados a 20%, conforme constante na proposta de revisão da NBR 6118, deve ser considerado como um processo simplificado que fornece valores contrários à segurança em alguns pontos da estrutura.

7.3.3 O modelo tradicional de subestruturas de contraventamento

Em parte dos exemplos apresentados, procurou-se questionar o conceito ainda corrente de dividir uma estrutura a ser analisada em duas partes, uma chamada "de contraventamento", destinada a absorver os efeitos de 2ª ordem, e outra "contraventada", onde os efeitos globais de 2ª ordem podem ser desprezados, sendo seus elementos calculados como peças isoladas. Tais elementos de contraventamento são usualmente pilares parede ou pórticos rígidos, mas não existe um critério definido do limite a partir do qual se considera que uma subestrutura está contraventando o restante da estrutura.

Duas situações devem ser distinguidas. Uma delas refere-se à separação de uma subestrutura de contraventamento para verificar, com o auxílio de um parâmetro simplificado como o Alfa ou o Gama-Z, se esta pode garantir sozinha a estabilidade global da edificação. Neste caso, a subestrutura será sempre menos rígida do que a estrutura em si, ficando esta análise a favor da segurança. Aplicam-se aqui apenas os mesmos comentários quanto à utilização de procedimentos simplificados.

Outra situação ocorre quando se deseja avaliar os efeitos de 2ª ordem e utilizam-se para tal apenas as subestruturas de contraventamento. Mostrou-se, através de alguns exemplos, que a ocorrência de elementos mais rígidos como parte da estrutura faz com que estes absorvam a maior parte dos efeitos horizontais, mas que a variação relativa aos efeitos de 2ª ordem é praticamente igual na parte de contraventamento e na parte contraventada. O fato dos esforços serem menores em módulo não significa que os efeitos de 2ª ordem são necessariamente menos importantes.

A utilização do conceito de subestrutura de contraventamento é, portanto, questionável. Face aos recursos atualmente disponíveis, torna-se mais interessante o modelo da estrutura como um todo, permitindo que todos os elementos contribuam para a estabilidade global da edificação e garantindo que os efeitos de 2ª ordem apliquem-se igualmente a todos.

7.3.4 Comportamento espacial das estruturas

A análise das estruturas sob forma espacial vem se tornando mais comum dia a dia. Em comparação com o sistema tradicional, onde os efeitos horizontais são analisados através de diversos pórticos resistentes e cada pavimento pode ter seu comportamento verificado através de modelos de grelhas, a abordagem espacial tem a vantagem essencial de produzir um modelo único, que não depende de decisões ou divisões da estrutura para ser criado.

Apesar de não ser objeto desta dissertação uma comparação entre o modelo plano e o espacial, por existirem muitas questões em aberto mesmo quanto ao comportamento das estruturas sob análise linear, foram utilizados alguns exemplos provindos de um software comercial para verificar alguns aspectos importantes relativos aos efeitos de 2ª ordem.

Verificou-se que a extensão dos procedimentos abordados ao longo deste trabalho para a forma espacial é simples e que os resultados encontrados não apresentam surpresas. A utilização de um modelo de pórtico espacial pode incluir sem dificuldade a não linearidade geométrica. Observou-se, com alguns exemplos, que o modelo espacial é normalmente um pouco mais rígido do que seu pórtico plano equivalente, obrigando o uso de um modelo com todos os pórticos encadeados para se obter um resultado mais preciso. Todavia, tal dificuldade material já justifica o uso diretamente do modelo espacial.

Em alguns casos, a estrutura apresenta um comportamento tridimensional que não pode ser capturado através de pórticos planos. Isto ocorre sempre que não existe um plano de simetria, forçando um giro da estrutura em torno de um eixo vertical. Verificou-se que os efeitos de 2ª ordem apenas acompanham o comportamento da estrutura. A variação nos esforços internos após a inclusão dos efeitos de 2ª ordem é similar tanto nas estruturas simétricas como nas assimétricas.

Uma característica adicional que pode influir em alguns modelos é a rigidez à torção considerada para as barras verticais (pilares). Uma redução no valor da rigidez conduz a um aumento nos efeitos de 2ª ordem. Por outro lado, esta redução é uma questão que influi da mesma forma na análise linear, não se encontrando ainda completamente estudada. O que se observa é que apenas tipos específicos de estruturas sofrem influência da rigidez à torção dos pilares. Normalmente, a resistência à torção apresentada pela estrutura como um todo é fornecida pelo comportamento conjunto dos pórticos transversais e não pela rigidez à torção. Portanto, a influência deste fator nos efeitos de 2ª ordem só será relevante nas estruturas onde também for relevante para a determinação dos efeitos de 1ª ordem.

7.3.5 O problema das combinações de ações

Em termos de projeto, existe uma questão muito importante detectada no tocante à análise de estruturas sob efeitos de 2ª ordem: o fato de que não é possível utilizar o princípio da superposição das ações da mesma forma como feito na análise linear. Quando o problema é linear, é suficiente processar a estrutura sujeita a cada caso de carregamento separadamente, utilizando-se valores característicos. Pode-se então compor os resultados obtidos para cada caso em uma ou mais combinações, utilizando, para isto, qualquer coeficiente de ponderação. Os valores obtidos na combinação dos resultados serão sempre os mesmos que seriam obtidos pela combinação das ações.

Quando a análise não é linear, tal procedimento deixa de ser válido. Cada combinação de carregamentos deve ser analisada separadamente, não sendo possível combinar os valores obtidos em duas análises. Isto se observa simplesmente verificando que o acréscimo nos momentos nos pilares, causados pelas forças de vento, será tanto maior quanto maiores forem as cargas axiais induzidas pelos carregamentos permanente e acidental.

Um ponto muito importante a se observar é o estado limite ao qual se destina a análise. Para a verificação dos estados limites de utilização, aplicam-se valores característicos dos carregamentos, enquanto que, para a verificação dos estados limites últimos, utilizam-se valores de cálculo (majorados por coeficientes de segurança e de combinação). Para a consideração dos efeitos de 2ª ordem, tais análises devem ser feitas separadamente, sendo vetada a possibilidade de majorar os resultados obtidos com o uso de valores característicos para se obter os valores de cálculo, sob pena de incorrer em erro contra a segurança.

Na medida em que existem diversos carregamentos variáveis atuantes sobre uma edificação, não é interessante considerá-los atuando simultaneamente, mas sim pesquisar a situação mais crítica entre as combinações aplicáveis. Dependendo do número de casos de carregamento, este critério pode levar a grande número de combinações. A priori, cada uma delas deveria ser analisada separadamente para a correta consideração dos efeitos de 2ª ordem.

7.4 - ASPECTOS NÃO ABORDADOS

Embora se tratem ao longo deste texto acerca de análises mais precisas e análises simplificadas, cabe lembrar que estão cobertos apenas alguns aspectos não lineares apresentados por uma estrutura de concreto armado. Justifica-se tal limitação pelo interesse na aplicação prática dos preceitos abordados em projetos usuais. A evolução na qualidade dos projetos deve ser feita gradativamente, aplicando corretamente os efeitos mais simples antes de incluir os mais complexos.

7.4.1 Aspectos não lineares geométricos

Tratando-se apenas de efeitos não lineares geométricos, existem ainda alguns pontos não abordados neste trabalho. Apenas com relação aos mesmos aspectos cobertos pelos três processos já citados, existem outras propostas que almejam a obtenção de resultados mais precisos ou a obtenção da mesma precisão com maior velocidade. Conforme estes processos forem mais amplamente estudados e reconhecidos, podem também seres trazidos para o ambiente de projeto.

Outro ponto importante é a consideração das imperfeições geométricas globais e locais, que pode ser feita diretamente na análise da estrutura. Apresenta-se fortemente, nesta questão, o mesmo problema das combinações de ações.

7.4.2 Aspectos gerais de projeto

A questão da não linearidade no projeto de estruturas avança ainda bastante além dos efeitos geométricos aqui estudados. Nas estruturas de concreto armado, uma grande fonte de incerteza vem do comportamento altamente não linear físico apresentado por este material composto. A consideração mais exata do conjunto concreto-armadura, sob o efeito da fissuração e retração, é algo ainda excessivamente complexo para fins de projeto.

O material concreto armado possui ainda comportamento variável com o tempo, sofrendo influência da deformação lenta e da história de carregamentos imposta à estrutura. Efeitos localizados e determinação apenas estatística das propriedades dos materiais são problemas também comuns.

Tal grau de incerteza pode invalidar o nível de precisão que se pretende obter em uma análise dita elaborada. Isto deve ficar claro em todos os trabalhos que pretendam analisar estruturas reais.

7.5 - ASPECTOS DE IMPLEMENTAÇÃO

Embora o desenvolvimento computacional não seja o objetivo central deste trabalho, constitui-se em uma contribuição que pretende ser dada à comunidade acadêmica. A nível profissional, a tendência de desenvolvimento de interfaces gráficas tem disseminado o uso da Programação Orientada a Objetos (OOP) em lugar da Programação Estruturada convencional. Por outro lado, os desenvolvimentos a nível acadêmico ainda pouco utilizam estas tendências atuais.

No decorrer do CAPÍTULO 4, procurou-se mostrar as principais características da Programação Orientada a Objetos, apresentando o modelo de objetos utilizado para representar um programa para solução de pórticos planos. Mostrou-se que esta abordagem, embora modifique certos paradigmas, é bastante acessível, principalmente aos pesquisadores que já têm conhecimento de Programação Estruturada. A partir do programa original pronto, mostrou-se a facilidade obtida para inclusão dos efeitos não lineares geométricos.

7.5.1 Vantagens da implementação orientada a objetos

O ponto principal no tocante à comparação entre a OOP e a Programação Estruturada diz respeito à manutenção e reutilização do código. Enquanto que a Programação Estruturada permite, ao máximo, o desenvolvimento de bibliotecas de funções para a realização de tarefas específicas, a Programação Orientada a Objetos dedica-se diretamente à definição de elementos estanques denominados classes. Corretamente utilizada, a OOP permite dividir o problema em estudo de tal forma que a continuidade do desenvolvimento torna-se algo absolutamente natural.

A possibilidade de reutilização de código usando a Programação Estruturada através de bibliotecas de funções mostra apenas a vantagem desta sobre os procedimentos mais antigos ainda de programação linear. Ao depender dos critérios de organização de cada programador individualmente na definição de conjuntos de funções, as quais devem ser utilizadas de forma e em uma sequência que deve ser estabelecida e largamente documentada, tenta-se apenas simular o que é feito naturalmente ao se encapsular o código necessário em uma classe.

O objetivo deste trabalho não é o de concluir qual estilo de programação é mais vantajoso, por ser este objeto de outra área de conhecimento. Pretendeu-se apenas mostrar uma maneira suficientemente clara de entender esta filosofia e mostrar sua aplicação a esta situação real.

Embora não se possam estabelecer conclusões baseadas apenas em experiências pessoais, deseja-se ressaltar o quanto um investimento na mudança de certos paradigmas pode ser vantajoso. A dificuldade material adicional gerada pelo aprendizado de uma nova filosofia pode ser compensada largamente, não apenas pela geração de código mais reutilizável, como também mais claro e legível. Ao se dividir o problema em duas partes, o modelo de objetos e a implementação em si, pode-se dar ao pesquisador uma visão melhor do que significa a implementação computacional.

7.5.2 Quanto à migração da Programação Estruturada para a OOP

Para aqueles pesquisadores que atualmente utilizam a Programação Estruturada em seus trabalhos, pode-se fazer algumas sugestões que facilitem a transição rumo à orientação a objetos.

A primeira idéia da orientação a objetos é a limitação do escopo. Entende-se por escopo a abrangência de uma variável ou procedimento. O primeiro passo a seguir é justamente o de limitar ao máximo a acessibilidade de cada informação. Além de eliminar variáveis globais, deve-se dividir procedimentos em procedimentos menores, tais que as tarefas de cada um sejam mais facilmente identificadas. É interessante separar os procedimentos análogos em diversos arquivos, procurando identificar os procedimentos que são locais ao arquivo e quais são usados externamente.

Com um código suficientemente subdividido, já é possível identificar em cada conjunto de funções uma classe que as engloba. Deve-se tentar separar os dados que pertencem à classe dos métodos que os acessam externamente (ocultamento de dados). Neste ponto, para migrar para a Programação Orientada a Objetos, deve-se escolher uma linguagem e ambiente de desenvolvimento que as suporte. Pode-se citar os ambientes C++ e Pascal como os mais disseminados e mais adequados.

O refinamento das classes inicialmente desenvolvidas gera classes auxiliares que encapsulam comportamentos específicos, bem como permite identificar as vantagens que podem ser obtidas através de herança e polimorfismo.

Evidentemente, a conversão de um programa existente através de refinamento não é um modo eficiente de se obter um programa orientado a objetos. Todavia, é um caminho que pode ser sugerido àqueles que desejam iniciar nesta área. Na verdade, a Programação Orientada a Objetos só é utilizada em sua plenitude quando a aplicação desejada é inicialmente projetada, com seu modelo de classes definido de forma a obter a máxima eficiência, ficando a codificação em si em segundo plano.

7.6 - SUGESTÃO PARA PESQUISAS FUTURAS

Considerando que o objetivo central desta dissertação é o de melhorar a qualidade dos projetos desenvolvidos na área de estruturas, surgem dois pontos importantes que merecem ser abordados e que representam uma continuidade lógica a este trabalho. O primeiro deles seria uma comparação completa entre os resultados obtidos entre uma análise plana e uma análise espacial. Uma ampla revisão a respeito dos conceitos envolvidos e da interpretação dos resultados obtidos seria de grande valia para viabilizar a correta utilização deste tipo de modelo em projetos usuais. Sugere-se a aplicação dos processos para inclusão da não linearidade geométrica na análise espacial, complementando os resultados obtidos nesta dissertação.

A segunda linha de pesquisa a ser recomendada é a da inclusão dos efeitos não lineares físicos em adição aos efeitos não lineares geométricos. A fim de possibilitar a aplicação prática, sugere-se manter o trabalho com os modelos de barras. Uma forma de capturar o comportamento não linear físico do concreto armado seria através de diagramas momento fletor – força normal – curvatura obtidos para cada seção. Discretizando-se corretamente as barras e incluindo-se simultanemente os efeitos não lineares físicos e geométricos, pode-se tentar capturar o fenômeno da Instabilidade nas estruturas de concreto armado. Isto permitiria obter simultaneamente os efeitos de 2ª ordem globais e locais, tornando desnecessária a análise de elementos isolados e a utilização de procedimentos simplificados como o da modificação do comprimento de flambagem dos pilares.

Fica também desta dissertação a sugestão de se estender este trabalho aproveitando-se os preceitos da Programação Orientada a Objetos, incentivando sua divulgação nos meios acadêmicos.

REFERÊNCIAS BIBLIOGRÁFICAS

AL-BERMANI, Faris G. A. & KITIPORNCHAI, Sritawat. "Nonlinear Analysis of Thin-Walled Structures Using Least Element/Member". *Journal of Structural Engineering.* Vol. 116, nº 1. American Society of Civil Engineers (ASCE): USA, 1990.

ALTOQI INFORMÁTICA. *AltoQi Eberick Master - Referência.* Florianópolis, SC: 1998.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *Projeto e Execução de Estruturas de Concreto Armado. NBR 6118.* Rio de Janeiro, RJ: 1978.

_____. *Projeto e Execução de Estruturas de Concreto Armado. NBR 6118. Texto base para revisão* Rio de Janeiro, RJ: 1994.

_____. *Projeto e Execução de Estruturas de Aço de Edifícios. NBR 8800.* Rio de Janeiro, RJ: 1988.

BEAUFAIT, ROWAN, HOADLEY & HACKETT. *Computer Methods of Structural Analysis.* London, England: 1970, Prentice-Hall.

BIRNSTIEL, Charles & IFFLAND, Jerome S. B. "Factors Influencing Frame Stability". *Journal of Structural Engineering.* Vol. 106, nº 4. American Society of Civil Engineers (ASCE): USA, 1980.

CHAN, Siu Lai & ZHOU, Zhi Hua. "Pointwise Equilibrating Polynomial Element for Nonlinear Analysis of Frames". *Journal of Structural Engineering.* Vol. 120, nº 6. American Society of Civil Engineers (ASCE): USA, 1994.

CHEN, W. F. & TOMA, S. *Advanced Analysis of Steel Frames.* Florida, USA: 1994, CRC Press, Inc.

COMITE EURO-INTERNATIONAL DU BETON. *CEB-FIP Model Code 1990. MC-90.* London, England: 1993.

FRANÇA, Ricardo L. e S. & STUCCHI, Fernando R. "NB-1/93 – Um panorama geral". In: *III Simpósio EPUSP sobre Estruturas de Concreto. Anais*. São Paulo, SP: 1993.

FUSCO, Péricles B. *Solicitações Normais*. Rio de Janeiro, RJ: 1981, Ed. Guanabara Dois.

_____. "A normalização brasileira na Engenharia de Estruturas". In: *III Simpósio EPUSP sobre Estruturas de Concreto. Anais*. São Paulo, SP: 1993.

GAIOTTI, Regina & SMITH, Bryan S. "P-Delta Analysis of Building Structures". *Journal of Structural Engineering*. Vol. 115, nº 4. American Society of Civil Engineers (ASCE): USA, 1989.

KIM, Seung-Eock & CHEN, Wai-Fah. "Pratical Advanced Analysis for Braced Steel Frame Design". *Journal of Structural Engineering*. Vol. 122, nº 11. American Society of Civil Engineers (ASCE): USA, 1986.

_____. "Pratical Advanced Analysis for Unbraced Steel Frame Design". *Journal of Structural Engineering*. Vol. 122, nº 11. American Society of Civil Engineers (ASCE): USA, 1986.

LIVESLEY, R. K. *Matrix Methods of Structural Analysis*. London, England: 1964, Pergamon Press Ltd.

LORIGGIO, Daniel D. Notas de aula da cadeira de Análise Matricial de Estruturas do curso de graduação em Engenharia Civil da UFSC. Florianópolis, SC: 1992.

LUI, E. M. & ZHANG, Cheng-Yin. "Nonlinear frame analysis by the pseudo load method". *Computers and Structures*. Vol. 37, Nº 05, 1990.

MARTINS, Antônio R. & STUCCHI, Fernando R. "Ações nas estruturas de concreto". In: *III Simpósio EPUSP sobre Estruturas de Concreto. Anais*. São Paulo, SP: 1993.

MOREIRA, Domício Falcão. *Análise Matricial das Estruturas.* Rio de Janeiro, RJ: 1977, Editora da USP, Livros Técnicos e Científicos Editora S.A.

PENNER, Elisabeth & FUSCO, Péricles B. "A estabilidade de edifícios altos". *Revista Téchne.* Ano V, nº 6. Editora PINI, 1997.

PITTA, Arthur L. "Edifícios de pequena altura: normalização simplificada". *Revista IBRACON.* Ano VI, nº 6. Instituto Brasileiro do Concreto, 1996.

SANTOS, Lauro M. & FRANCO, Mário. "Instabilidade e efeitos de 2ª ordem nas estruturas de concreto". In: *III Simpósio EPUSP sobre Estruturas de Concreto. Anais.* São Paulo, SP: 1993.

SELKE, Carlos A. C. & PEREIRA, Luiz T. do V. *Introdução ao Método dos Elementos Finitos.* Florianópolis, SC: Editora da UFSC, 1994.

SUSSEKIND, José C. *Curso de Concreto.* Volume II. 2ª ed. São Paulo, SP: 1982, Editora Globo S/A.

WEAVER Jr., William & GERE, James M. *Matrix Analysis of Framed Structures.* 2nd ed. New York, USA: 1965, D. Van Nostrand Company.

ZERMIANI, Fabiano L. *Contribuição à análise não linear geométrica de pórticos planos.* Dissertação de Mestrado em Engenharia Mecânica – Universidade Federal de Santa Catarina. Florianópolis, SC: 1998.

ZERMIANI, Fabiano L. & LORIGGIO, Daniel D. "Matriz de Rigidez Geométrica para Pórticos Planos". In: *Congresso Técnico Científico de Engenharia Civil. Anais.* Florianópolis, SC: 1996.

ZHOU, Zhi Hua & CHAN, Siu Lai. "Refined Second-Order Analysis of Frames with Members under Lateral and Axial Loads". *Journal of Structural Engineering.* Vol. 122, nº 5. American Society of Civil Engineers (ASCE): USA, 1996.

BIBLIOGRAFIA ADICIONAL

BERRY, John. *Programando em C++*. São Paulo, SP: Makron Books, 1991.

CORRÊA, Márcio R. S. & RAMALHO, Márcio A. "Análise de estruturas de concreto". In: *III Simpósio EPUSP sobre Estruturas de Concreto. Anais*. São Paulo, SP: 1993.

FORDE, Bruce W. R., FOSCHI, Ricardo O. & STIEMER, Siegfried F. "Object-oriented finite element analysis". *Computers and Structures*. Vol. 34, Nº 03, 1990.

LORIGGIO, Daniel D. "Considerações sobre o Dimensionamento de Peças de Madeira submetidas à Flexo-Compressão com a Inclusão da Não Linearidade Geométrica". In: *V EBRAMEN. Anais*. Vol. 1. Belo Horizonte, MG: 1995.

_____. "Considerações sobre o Estado Limite Último de Instabilidade de Peças de Madeira". In: *IV EBRAMEN. Anais*. Vol. 2 Florianópolis, SC: 1998.

LORIGGIO, Daniel D. & SENEM, P. R. "Análise Não Linear e Análise Limite de Pórticos Planos por Métodos Numéricos". In: *XXVII Jornadas Sudamericas de Ingeniería Estructural. Anais*. Vol. 3. Tucumán, Argentina: 1995.

_____. "Limit Analysis of Plane Frames by Numerical Methods". In: *V EPMESC. Anais*. Macau: 1995.

WEAVER Jr., William & GERE, James M. *Matrix Algebra for Engineers*. 2nd ed. Monterey, California: 1983, Brooks/Cole Engineering Division.

APÊNDICE 1.
PROGRAMA IMPLEMENTADO

1.1 - OBJETIVOS

A fim de se poder desenvolver os conceitos da não linearidade geométrica em pórticos planos, optou-se pelo desenvolvimento de um programa computacional que implementasse todos os procedimentos discutidos. Embora existam comercialmente alguns programas de Análise Estrutural que contemplam alguns dos pontos apresentados, através da implementação pode-se ter a máxima liberdade na escolha dos processos e parâmetros a serem utilizados.

Uma vez que esta implementação não se destina à aplicação comercial, não existem diversas preocupações de interface e usabilidade. Na verdade, objetiva-se como contribuição deste trabalho apenas a parte de solução e não o programa como um todo. A utilização das técnicas de Orientação a Objetos facilitou a completa separação das classes destinadas à solução do problema do aplicativo em si.

Descreve-se aqui a estrutura do aplicativo para permitir a futuros pesquisadores na área a utilização deste trabalho no cálculo de modelos que envolvam a consideração de não linearidade geométrica em pórticos planos.

1.2 - FILOSOFIA DE TRABALHO

Tendo em vista o fato de que o presente projeto possui natureza puramente científica, torna-se claro que a parte fundamental do desenvolvimento é o núcleo central do programa (módulo de resolução da estrutura), tendo sido este o objetivo proposto para esta dissertação. Desta forma, optou-se por fazer unicamente a entrada e saída dos dados relevantes via arquivo texto (arquivos padrão ASCII, sem formatação).

Para a entrada dos dados, portanto, o programa requer que o usuário gere previamente um arquivo contendo os dados necessários ao programa. Com isto, quer-se dizer que não foi dada ênfase à confecção de um módulo de pré-processamento (entrada de dados), por este não pertencer ao escopo da pesquisa.

Da mesma forma, a saída de dados oferecida pelo programa constitui-se de arquivos texto contendo os resultados relevantes para o problema em questão.

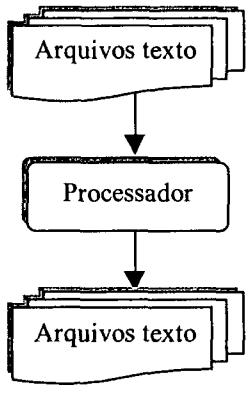


Figura A7.1 – Filosofia de trabalho do programa implementado

Paralelamente a isto, decidiu-se pela apresentação de alguns resultados sob forma gráfica. Estes são o desenho da estrutura em sua configuração deformada e os diagramas de esforços internos nas barras. Isto pôde ser feito mesmo sem a implementação de um módulo gráfico, lançando-se mão do artifício de gravar os desenhos gerados em formato DXF. O formato DXF (Data eXchange Format) é um formato público, baseado em arquivo texto padrão, onde sequências de códigos representam entidades de desenho. Estes arquivos podem ser importados posteriormente em quase todos os programas CAD do mercado, para visualização e impressão do desenho gerado.

1.2.1 Módulo processador linear

A partir de um único arquivo de entrada de dados, pode-se optar por fazer a análise do modelo em questão utilizando-se a análise linear ou um dos três processos abordados para inclusão da não linearidade geométrica.

Utilizando-se a opção "Análise linear", é necessário apenas o arquivo de entrada de dados, obtendo-se outro arquivo formato contendo os resultados obtidos. Consideram-se como resultados apresentados pelo programa:

- Deslocamentos nodais;
- Esforços internos nas barras;
- Deslocamentos e momentos fletores em formato DXF

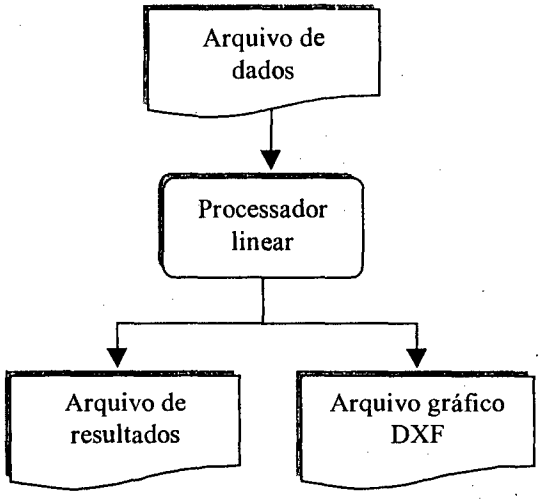


Figura A1.2 – Filosofia de trabalho do programa de análise linear

1.2.2 Módulo processador não linear

Utilizando-se a opção "Análise não linear geométrica", pode-se optar por um dos três processos descritos para inclusão dos efeitos não lineares geométricos. Para a realização da análise não linear, são necessários uma série de parâmetros, como, por exemplo, a precisão mínima preestabelecida. Optou-se por não utilizar valores fixos, mas sim presentes na entrada de dados. Uma vez que estes parâmetros são similares para diversos modelos em análise, optou-se por contê-los em um único arquivo de configuração, que também é lido durante a análise.

Efetuada a análise não linear, tem-se disponível as seguintes informações, agora referentes à análise de 2ª ordem:

- Deslocamentos nodais;
- Esforços internos nas barras;
- Deslocamentos e momentos fletores em formato DXF;
- Registro do processo utilizado para a consideração da não linearidade geométrica, indicando valores representativos de cada iteração;
- Parâmetros de instabilidade Alfa e Gama-Z.

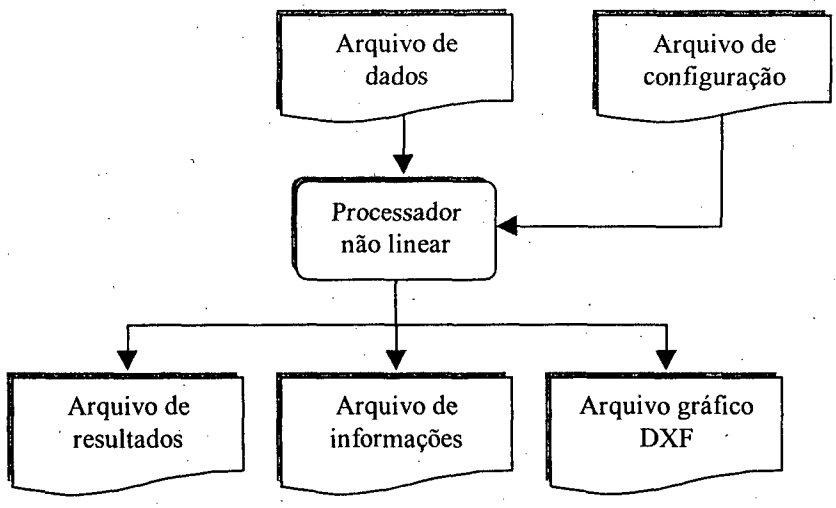


Figura A1.3 – Filosofia de trabalho do programa de análise não linear

1.3 - ARQUIVO DE ENTRADA DE DADOS

A entrada dos dados da estrutura a ser processada deve ser feita a partir de um arquivo formato texto. Este arquivo de entrada deve ser formatado de acordo com uma certa regra estabelecida pelo programa como ordenação dos dados a serem informados. Assim sendo, deve-se seguir uma estrutura modelo, descrita a seguir.

Para a criação deste arquivo, pode-se utilizar qualquer editor de textos que possua a capacidade de gravar arquivos em formato ASCII, como, por exemplo, o EDIT do MS-DOS ou o Notepad do Windows. A implementação do programa faz uso de uma biblioteca predefinida que já inclui um ambiente de edição de textos, evitando a necessidade de se utilizar um editor externo.

Para o funcionamento do programa conforme implementado, o arquivo de entrada de dados deve obrigatoriamente ter a extensão DAD. Todos os arquivos gerados pelo programa terão o nome do arquivo original e uma extensão diferente para cada caso.

Para fins de facilidade de leitura posterior, foram incluídas linhas de cabeçalhos separando as diversas seções de dados. Tais linhas são obrigatórias, mas utilizadas apenas como delimitadores. Existem, sinteticamente, as seguintes seções:

- Linha de título

```
Linha de título
```

Esta linha apenas descreve o título do problema em estudo, para referência posterior.

- Número de nós

```
Número de nós
NN
```

O valor NN indica a quantidade de nós no modelo.

- Número de barras

```
Número de barras
NB
```

O valor NB indica a quantidade de barras no modelo.

- Coordenadas nodais

```
Coordenadas nodais (X, Y)
X(I)    Y(1)
X(2)    Y(2)
M
X(NN)   Y(NN)
```

Cada linha representa um nó da estrutura, onde devem ser indicadas as coordenadas X e Y do nó. Os dois valores podem ser separados por um ou mais espaços.

- Incidências das barras

```
Incidências das barras (noI, noF)
noI(1)    noF(1)
noI(2)    noF(2)
M
noI(NB)   noF(NB)
```

Cada linha representa uma barra da estrutura, onde devem ser indicados números dos dois nós que definem a barra. Os dois valores podem ser separados por um ou mais espaços.

- Propriedades das barras

```
Propriedades (barraI, barraF, B, H, Ec)
NP
barraI(1)    barraF(1)  B(1)  H(1)  Ec(1)
barraI(2)    barraF(2)  B(2)  H(2)  Ec(2)
M
barraI(NP)   barraF(NP) B(NP) H(NP) Ec(NP)
```

Nesta seção, definem-se as propriedades geométricas das barras. Como usualmente diversas barras possuem propriedades comuns, não é necessário definir uma linha para cada uma. Definem-se tipos de seções e as barras sobre as quais elas se aplicam.

A primeira linha define o número de tipos de seções no arquivo, NP. A seguir, cada linha define as propriedades de um tipo de seção, na seguinte ordem:

- barraI e barraF: número da primeira e última barra sobre as quais aplicar o tipo de seção sendo definido;

- B e H: dimensões da seção transversal da barra (sobre estas são calculadas automaticamente as propriedades geométricas da seção, ou seja, a área e o momento de inércia);
- Ec: módulo de elasticidade longitudinal utilizado para o elemento.

- Condições de vinculação dos nós

```
Restrições de apoio (glX, glY, glZ)
NR
no(1)      glX(1)     glY(1)      glZ(1)
no(2)      glX(2)     glY(2)      glZ(2)
M
no(NR)    glX(NR)    glY(NR)     glZ(NR)
```

Nesta seção, são indicados os nós que possuem vinculações. A primeira linha define a quantidade de nós vinculados, NR. A seguir, cada linha define as condições de vinculação de um nó, na seguinte ordem:

- no: número do nó vinculado;
- glX: condição de vinculação do nó com relação ao deslocamento na direção X (onde 1 significa deslocamento restringido e 0 significa deslocamento livre);
- glY: condição de vinculação do nó com relação ao deslocamento na direção Y;
- glZ: condição de vinculação do nó com relação ao giro.

- Cargas distribuídas nas barras

```
Cargas distribuídas (barraI, barraF, valor)
NCD
barraI(1)     barraI(1)     valor(1)
barraI(2)     barraI(2)     valor(2)
M
barraI(NCD)   barraI(NCD)   valor(NCD)
```

Nesta seção, são indicadas as barras que possuem carregamento distribuído ao longo de seu comprimento. A primeira linha define a quantidade de barras que possuem carregamentos, NCD. A seguir, cada linha define as condições de carregamento de uma barra, na seguinte ordem:

- barraI e barraF: número da primeira e última barra sobre as quais aplicar a tipo de seção sendo definido;
- valor: valor da carga uniformemente distribuída por unidade de comprimento ao longo da barra.

Mesmo que não hajam carregamentos distribuídos nas barras, esta seção deve ser informada, contendo apenas o valor 0 (zero) como NCD.

- Cargas concentradas nos nós

```
Cargas concentradas (no, Fx, Fy, Mz)
NCC
no(1)     Fx(1)     Fy(1)     Mz(1)
no(2)     Fx(2)     Fy(2)     Mz(2)
M
no(NCC)   Fx(NCC)   Fy(NCC)   Mz(NCC)
```

Nesta seção, são indicados os nós que possuem cargas aplicadas. A primeira linha define a quantidade de nós que possuem carregamentos, NCC. A seguir, cada linha define as condições de carregamento de um nó, na seguinte ordem:

- no: número do nó onde o carregamento será aplicado;
- Fx: valor da carga concentrada aplicada paralelamente ao eixo X;
- Fy: valor da carga concentrada aplicada paralelamente ao eixo Y;
- Mz: valor da momento fletor aplicado.

Mesmo que não hajam carregamentos concentrados nos nós, esta seção deve ser informada, contendo apenas o valor 0 (zero) como NCC.

- Nó referência

```
No referência
REF
```

Nesta seção é requerido um único valor, que o número do nó utilizado como referência para verificar a convergência dos processos não lineares. Este valor é ignorado pelo programa de análise de 1ª ordem.

1.3.1 Exemplo

Para exemplificar a entrada de dados, pode-se tomar o Exemplo 2 já abordado anteriormente. Repetindo, apresenta-se o esquema estrutural de um edifício simples, sujeito a esforços verticais e horizontais:

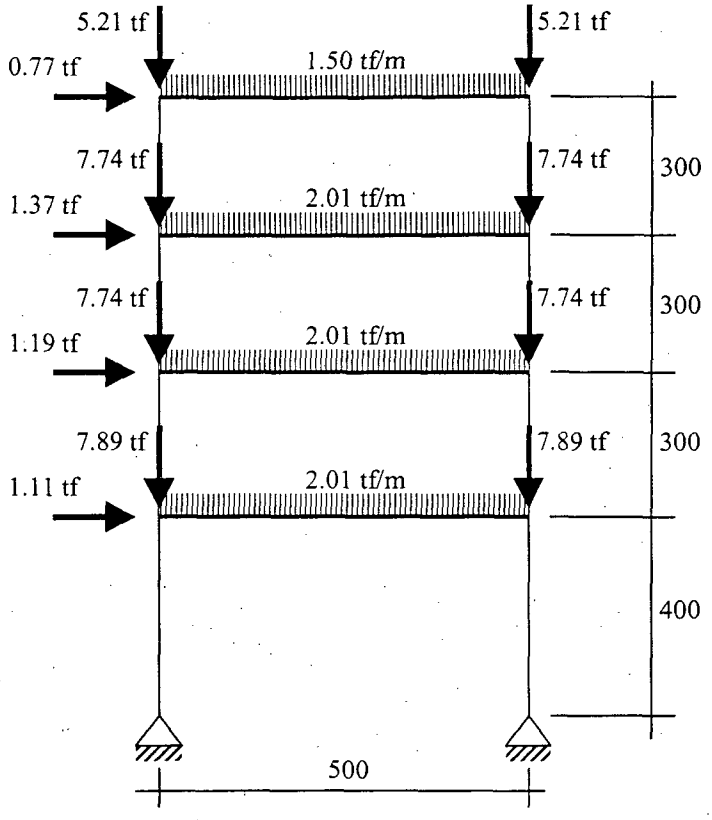


Figura 5.1 – Exemplo 2 – Pórtico apoiado na base

Serão utilizadas as seguintes propriedades:

- Módulo de elasticidade: $2{,}7 \times 10^5$ kgf/cm^2
- Seção transversal das barras: 30 x 20 cm para as barras verticais (pilares) e 13 x 55 cm para as barras horizontais (vigas).

A listagem a seguir corresponde ao arquivo de entrada de dados utilizado para calcular este exemplo.

```
**** Exemplo pórtico Apoiado
Numero de nós
10
Numero de barras
12
Coordenadas dos nós (X - Y)
0           0
0         400
0         700
0        1000
0        1300
500         0
500       400
500       700
500      1000
500      1300
Incidências das barras(noI - noF)
2   7
```

```
3    8
4    9
5   10
1    2
2    3
3    4
4    5
6    7
7    8
8    9
9   10
Secoes (barraI - barraF - B - H - Ec)
2
1     4       13.0     55.0      270000
5    12       30.0     20.0      270000
Restricoes de apoio
2
1    1 1 0
6    1 1 0
Cargas distribuídas (barraI - barraF - valor)
2
1     3       -20.1
4     4       -15.0
Cargas concentradas nos nos
9
1      550         0     0
2     1100     -7890     0
3     1190     -7740     0
4     1370     -7740     0
5      770     -5210     0
7        0     -7890     0
8        0     -7740     0
9        0     -7740     0
10       0     -5210     0
No referencia
5
```

1.3.2 Numeração dos nós

A matriz de rigidez da estrutura resultante é uma matriz esparsa, ou seja, contém grande número de valores nulos em seu interior. Mais especificamente, esta matriz possui a característica de ser uma matriz do tipo *banda*, ou seja, os valores não-nulos concentram-se em torno da diagonal principal (vide item 2.3.3). Os procedimentos para solução do sistema de equações lineares resultantes aproveitam-se desta característica para minimizar o tempo necessário à obtenção da solução.

O tempo necessário para a solução será tanto menor quanto menor for a largura da banda da matriz, sendo esta definida a partir da máxima diferença entre a numeração dos nós que concorrem em uma mesma barra. Desta maneira, a numeração adotada para os nós torna-se de suma importância em modelos de maior porte.

Para este programa, foi implementado um algoritmo de renumeração interna dos nós, independente da numeração informada pelo usuário. Desta forma, não é necessário qualquer preocupação a este respeito na geração do arquivo de entrada de dados.

1.4 - ARQUIVO DE RESULTADOS

Como principal resultado da análise de uma estrutura reticulada, tem-se os deslocamentos nodais e os esforços internos obtidos em cada barra. No caso deste programa implementado, podem ser fornecidos estes resultados para duas situações distintas:

- resultados da análise de 1ª ordem, obtidos ao se utilizar a opção "Análise linear";
- resultados da análise de 2ª ordem, após todas as iterações necessárias, obtidos ao se escolher um dos procedimentos disponíveis para análise não linear.

Os resultados apresentados encontram-se formatados e dividem-se em duas seções:

- Deslocamentos nodais: a primeira tabela apresenta os deslocamentos finais sofridos pelos nós (deslocamento em X, deslocamento em Y e rotação):

Deslocamento final dos nós			
Número	X	Y	Z
1	0.0000	0.0000	-0.0342
2	9.5429	-0.0991	-0.0032
3	10.6579	-0.1577	-0.0018
4	11.2985	-0.1958	-0.0015
5	11.5626	-0.2120	-0.0012
6	0.0000	0.0000	-0.0354
7	9.5432	-0.1350	-0.0007
8	10.6562	-0.2042	0.0002
9	11.2967	-0.2465	0.0005
10	11.5597	-0.2635	0.0008

- Esforços internos: a segunda tabela apresenta os esforços internos ocorridos em cada barra da estrutura. Na sequência, apresentam-se: esforço axial, esforços cortantes no início e no fim da barra e momentos fletores também no início e no fim da barra. A convenção de sinais adotada é a seguinte:

Figura A1.4 – Convenção de sinais para uma barra de pórtico plano

Esforços nas barras					
Número	Axial	Cortante I.	Cortante F.	Fletor I.	Fletor F.
1	-129.01	565.55	9484.45	-938984.44	-1290739.88
2	674.04	3325.16	6724.84	-200169.88	-649748.50
3	672.58	4146.52	5903.48	345.68	-439587.41
4	1123.77	3499.77	4000.23	48502.70	-173618.34
5	40117.00	2088.63	-2088.63	0.00	835451.94
6	31661.45	859.62	-859.62	103532.48	154353.86
7	20596.29	343.66	-343.66	45816.01	57281.46
8	8709.77	-353.77	353.77	-57627.14	-48502.70
9	54693.00	2341.37	-2341.37	0.00	936548.06
10	37318.55	2470.38	-2470.38	354191.81	386921.84
11	22853.71	1796.34	-1796.34	262826.62	276075.91
12	9210.23	1123.77	-1123.77	163511.50	173618.34

1.5 - ARQUIVO DE CONFIGURAÇÃO

Os parâmetros utilizados na análise não linear são armazenados em um único arquivo, também em formato texto, utilizado para todos os modelos a serem processados.

Para o funcionamento do programa conforme implementado, o arquivo de configurações deve obrigatoriamente ter o nome PORTICO.CFG e estar gravado no mesmo diretório onde está o arquivo de dados a ser processado. Caso este arquivo não seja encontrado, serão utilizados valores padrão constantes no código.

Este arquivo deve ser formatado de modo a que cada item esteja contido em duas linhas, uma contendo a descrição e outra o valor. Na ordem, tem-se os seguintes itens a incluir no arquivo:

```
Discretizacao das barras (0 - nenhum, 1 - iguais, 2 - extremos,
3 - inicio)
1
```

Define a forma como as barras serão discretizadas:

- 0 – sem discretização;
- 1 – divisões com tamanhos iguais;
- 2 – duas divisões em cada extremidade e a parte central usualmente maior; ou
- 3 – duas divisões apenas no início da barra, com o restante usualmente maior.

```
Dividir somente as barras verticais (0 - não, 1 - sim)
1
```

Define se a discretização será feita apenas sobre as barras verticais (pilares) ou sobre todas as barras da estrutura.

```
Tamanho das divisões das barras (x L)
0.1
```

Define o tamanho das divisões, em relação ao comprimento total da barra. No caso das divisões serem de tamanhos diferentes (opções 2 e 3 do primeiro item), define o tamanho relativo dos segmentos extremos.

```
Tipo NL geométrica (0-nenhuma, 1-P-Delta, 2-KG, 3-F.
estabilidade)
2
```

Define o processo a ser utilizado para inclusão da não linearidade geométrica:

- 0 – Nenhum (equivale a fazer uma análise linear);
- 1 - Processo P-Delta;
- 2 - Processo da Matriz de Rigidez Geométrica;
- 3 - Processo das Funções de Estabilidade.

```
Precisão mínima NL geométrica
0.01
```

Define a variação máxima entre um deslocamento e o mesmo deslocamento obtido na iteração anterior. Calcula-se esta variação como sendo a razão entre a diferença entre as iterações e o valor da iteração antrerior.

```
N° máximo de iterações (geométrico)
20
```

Define o número máximo de iterações a realizar. Caso não se obtenha a precisão mínima preestabelecida antes de se alcançar o número limite de iterações, diz-se que o problema não convergiu.

1.6 - ARQUIVO GRÁFICO DXF

O artifício de se gerar saídas gráficas em formato DXF é algo bastante interessante, por uma série de razões:

- Permite a visualização gráfica dos resultados sem que seja necessário implementar um módulo de pós-processamento gráfico;
- Utiliza formatação texto padrão, o que permite sua fácil manipulação em qualquer ambiente ou linguagem de programação;
- Pode ser lido em praticamente qualquer um dos pacotes CAD disponíveis no mercado;
- Sua codificação é bem documentada e bastante acessível.

Embora a padronização DXF possa incluir grande número de elementos diferentes, com diversas seções e subseções, possui um 'núcleo básico que pode ser facilmente utilizado para geração de arquivos simples. Não é objetivo aqui explanar sobre toda a formatação DXF, mas considera-se interessante mostrar o conteúdo mínimo que deve ter um arquivo DXF para que este possa ser lido posteriormente em um programa como o AutoCAD.

Deve-se seguir uma sintaxe mínima e conter ao mínimo as seções descritas a seguir. Um padrão é constante: grava-se sempre um código identificador e a seguir o valor associado a ele. Desta forma, podem ser omitidas muitas informações não essenciais, associadas a outros códigos não incluídos, normalmente presentes nos arquivos de desenho gerados pelos aplicativos comerciais.

1.6.1 Cabeçalho

No início do arquivo, existe uma seção que pode gravar variáveis internas do programa. Mesmo vazia, é obrigatória e pode apresentar-se apenas como:

```
0
SECTION
2
HEADER
0
ENDSEC
```

1.6.2 Tabelas

A seguir, gravam-se tabelas de informações relativas ao desenho. Incluem-se aqui estilos de linha, estilos de texto, parâmetros da janela, entre outros. Destas, a única tabela obrigatória é a que define os níveis (*layers*) de desenho. Pode-se separar o desenho gerado em layers específicos, facilitando sua manipulação posterior no ambiente do aplicativo.

Por exemplo, o programa implementado utiliza três layers:

- Layer 0 – estrutura indeformada;
- Layer 1 – estrutura deformada;
- Layer 2 – diagramas de momentos fletores.

A seção de tabelas inicia da seguinte forma:

```
0
SECTION
2
TABLES
```

Dentro da seção, podem existir várias tabelas. A tabela de layers inicia da seguinte forma:

```
0
TABLE
```

A seguir, deve-se incluir um código que indica a quantidade de layers que serão definidos:

```
70
3
```

Abaixo, definem-se tantas entradas quantas forem os layers definidos, na seguinte sintaxe mínima:

```
0
LAYER
2
Estrutura
70
0
62
3
6
CONTINUOUS
```

São variáveis aqui apenas o código 2, onde indica-se o nome do layer, e o código 62, que indica o número da cor associada ao layer.

Ao final, deve-se encerrar a tabela de layers e a seção de tabelas:

```
0
ENDTAB
0
ENDSEC
```

1.6.3 Entidades de desenho

Na tabela a seguir, definem-se os elementos de desenho propriamente ditos. O padrão DXF engloba dezenas de entidades possíveis, cada uma com sua codificação associada. Caso se deseje detalhes sobre a geração de cada uma delas, deve-se consultar a documentação fornecida pela AutoDesk.

Neste trabalho, apresenta-se como interessante apenas o básico, que é o desenho de linhas simples. Para tal, deve-se iniciar a seção como o seguinte:

```
0
SECTION
2
ENTITIES
```

Cada elemento possui um identificador associado e inicia sempre pelo código de controle 0. No caso da linha, esta identifica-se como o elemento "LINE" e pode ser definida a partir dos seguintes dados mínimos:

```
0
```

```
LINE
8
Estrutura
62
BYLAYER
10
xInicial
20
yInicial
11
xFinal
21
yFinal
```

O código 8 indica o nome do layer no qual o elemento está inserido. O código 62 indica a cor do elemento, que pode ser definida como um código de cor ou com a constante "BYLAYER", que faz com que seja utilizada a cor definida para o layer. Os demais códigos devem ser substituídos pelas coordenadas da linha.

Ao final, deve-se encerrar a seção de entidades e finalizar o arquivo DXF:

```
0
ENDSEC
0
EOF
```

Com estas simples instruções, pode-se gerar desenhos de linhas de qualquer complexidade e visualizá-los em praticamente qualquer programa de CAD externo.

1.7 - OUTROS RESULTADOS

Além do arquivo de resultados já mencionado, o programa implementado gera ainda outros três tipos de arquivos, todos em formato texto padrão:

- Arquivo de registro da análise de 2ª ordem (com a extensão LOG);
- Arquivo de exportação de dados para o programa SAP90 (com as extensões SAP e SP2);
- Arquivo de exportação para programas CAD, contendo a representação gráfica da estrutura deformada e dos diagramas de momentos fletores (com a extensão DXF).

1.7.1 Arquivo de "log"

Além da possibilidade de se obter os resultados em termos de deslocamentos e esforços internos tanto para a análise de 1ª como para 2ª ordem, é gerado um arquivo contendo um registro das principais informações relevantes ao processamento da análise de 2ª ordem. Este arquivo apresenta-se como seguinte:

```
*******************************************************************************
PÓRTICO PLANO
     Não linearidade geométrica:    Processo P-Delta
     Precisão mínima: 0.1%
     Discretização das barras: Nenhuma
*******************************************************************************
*** Análise de 1ª ordem: ************
Deslocamento inicial: 11.563
Parâmetro Alfa: 1.063
Coeficiente Gama-Z: 1.386
Deslocamento previsto: 16.022
**************************************
Deslocamento ref: 16.590          dif = 43.483%
Deslocamento ref: 19.072          dif = 14.958%
Deslocamento ref: 20.313          dif = 6.507%
Deslocamento ref: 20.934          dif = 3.059%
Deslocamento ref: 21.246          dif = 1.487%
Deslocamento ref: 21.401          dif = 0.734%
Deslocamento ref: 21.479          dif = 0.365%
Deslocamento ref: 21.519          dif = 0.182%
Deslocamento ref: 21.538          dif = 0.091%
Tempo total de processamento: 0'1''
```

Pode-se dividir este arquivo em três seções:

- Seção 1 – Parâmetros: no início do arquivo, registram-se o processo sendo utilizado, a precisão mínima preestabelecida entre duas iterações e a discretização utilizada para as barras.
- Seção 2 – Análise de 1ª ordem: a seguir, apresentam-se os resultados obtidos na análise de 1ª ordem:
  - Deslocamento inicial: deslocamento horizontal (X) sofrido pelo nó referência indicado no arquivo de dados;
  - Parâmetro Alfa: valor do parâmetro de instabilidade Alfa calculado para este modelo com base nos resultados da análise de 1ª ordem (conforme colocado no item 2.4.3, um valor superior a 0.6 indica efeitos de 2ª ordem superiores a 10%);
  - Coeficiente Gama-Z: valor do coeficiente Gama-Z calculado para este modelo com base nos resultados da análise de 1ª ordem (conforme colocado no item 2.4.4, este valor destina-se a prever a magnitude dos efeitos de 2ª ordem de forma direta);
  - Deslocamento previsto: com base no coeficiente Gama-Z já obtido, indica-se o valor de deslocamento no nó referência que seria obtido com o uso deste processo, para fins de comparação com os resultados reais.
- Seção 3 – Iterações: registram-se, para cada uma das iterações efetuadas antes de se obter a precisão preestabelecida, o deslocamento no nó referência e a diferença deste valor com relação ao anterior. Quando esta diferença é inferior à precisão mínima, consideram-se obtidos os resultados finais.

Com base nestas informações, pode-se montar gráficos de evolução dos resultados para cada iteração, conforme apresentado nas diversas figuras constantes no texto.

1.7.2 Arquivo de exportação SAP90

Com base nas informações contidas no arquivo de dados, pode-se gerar um arquivo equivalente para leitura em outro aplicativo que também suporte entrada de dados formato texto. Este tipo de procedimento permite executar rapidamente o mesmo modelo em outro programa.

Conhecida a formatação de dados utilizada pelo programa com o qual deseja-se fazer a interface, basta apenas gravar os dados necessários em um arquivo formato texto. Como exemplo, foi implementada uma saída de dados no formato utilizado pelo conhecido programa de análise estrutural SAP90.

Dois arquivo de dados são gerados:

- Um contendo exatamente o modelo informado, com a extensão .SAP;
- Outro contendo o modelo modificado após a discretização das barras, com a extensão .SP2.

1.8 - INTERFACE IMPLEMENTADA

Optou-se pelo desenvolvimento de uma interface padrão Windows, dedicada à manipulação dos arquivos texto necessários ao programa. A programação desta interface, bem como a manipulação dos arquivos texto, utiliza apenas classes predefinidas fornecidas com o compilador utilizado, o Borland C++.

1.8.1 Edição de textos

Este aplicativo padrão utiliza funções predefinidas para gerenciamento de arquivos texto. Possui interface MDI (Multiple Document Interface), ou seja, permite abrir diversas janelas simultamentamente, cada uma contendo um arquivo diferente.

Estão presentes as funções básicas para abrir, alterar, gravar e imprimir os arquivos, todos recursos gerados automaticamente.

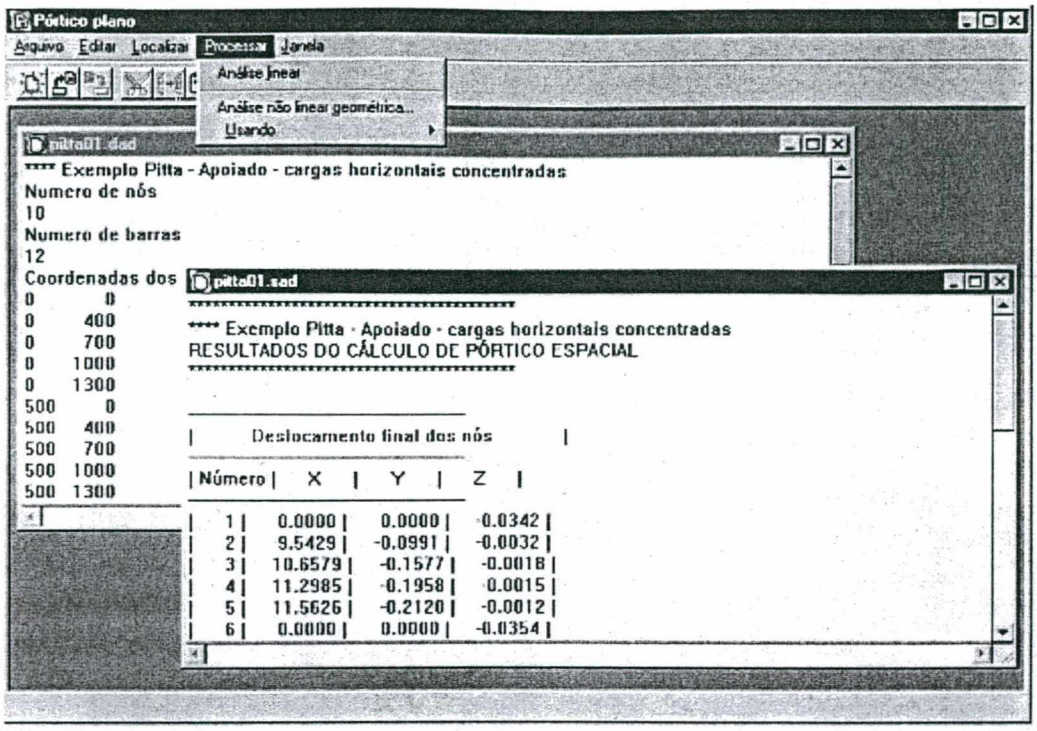


Figura A1.5 –Ambiente de edição de textos

Esta parte da implementação pode ser gerada facilmente na maior parte dos modernos ambientes de desenvolvimento para Windows.

1.8.2 Opções de processamento

As inclusões feitas ao módulo básico de edição de textos são as que fazem a análise da estrutura contida no arquivo de dados. Estas podem ser acessadas no menu "Processar". O arquivo atual (janela corrente) deve ser o arquivo de dados do modelo.

Utilizando-se a opção "Análise linear", é efetuada uma análise linear sobre o arquivo corrente. São gerados diversos arquivos, mas apenas o arquivo de resultados é aberto como uma nova janela.

Utilizando-se a opção "Análise não linear geométrica...", é aberto um diálogo onde podem ser preenchidas as opções que serão gravadas no arquivo de configurações.

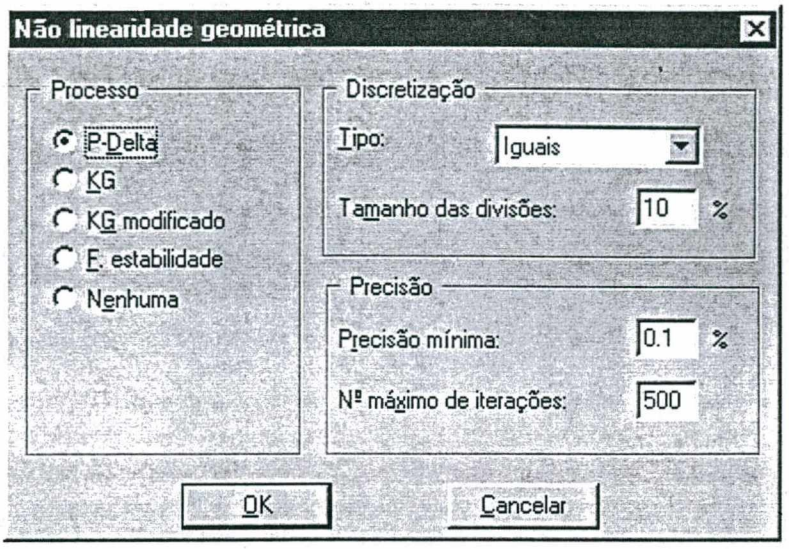


Figura A1.6 – Configurações da análise não linear

Pressionando-se o botão "OK", é efetuada uma análise de 2ª ordem sobre o modelo corrente, utilizando as opções configuradas no diálogo. Uma janela de progresso indica a evolução dos cálculos. Ao final, o conteúdo desta janela é o mesmo que está gravado no arquivo de "log".

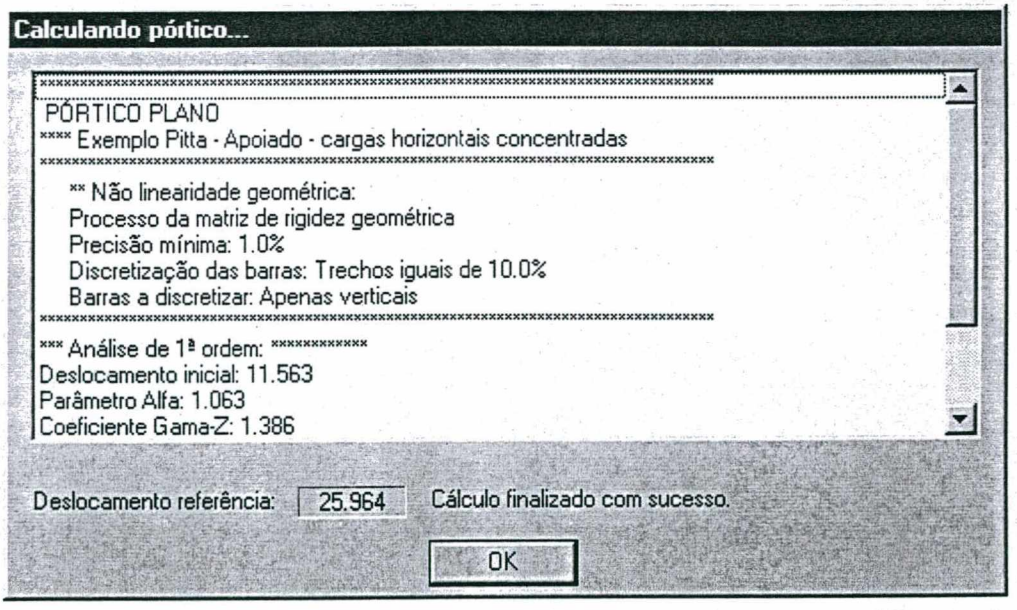


Figura A1.7 – Janela de progresso da análise não linear

Da mesma forma, são gerados diversos arquivos, mas apenas o arquivo de resultados, referente à análise de 2ª ordem, é aberto como uma nova janela.

As demais opções do menu permitem analisar o modelo corrente usando um processo para inclusão da não linearidade geométrica definido e as configurações correntes, sem passar pelo diálogo de configurações.

1.9 - COMENTÁRIOS

Conforme já exposto, a implementação deste programa centra-se apenas na parte de resolução. O código desenvolvido pode posteriormente ser aproveitado por outros pesquisadores, fazendo a este as extensões necessárias. Todavia, decidiu-se por também deixar pronto, como opção secundária, um módulo de resolução já compilado, que possa ser diretamente utilizado por outros pesquisadores que desejem apenas executar outros exemplos. Neste capítulo, foram abordadas os principais pontos que permitem a sua utilização.

O capítulo a seguir fará a exposição da implementação adotada, onde se confirmará a independência entre as classes de resolução e a interface implementada.

APÊNDICE 2.
LISTAGENS (ANÁLISE LINEAR)

2.1 VISÃO GERAL

Na filosofia da Programação Orientada a Objetos, não existe algo como um procedimentos principal e suas sub-rotinas. Diversas classes se inter-relacionam para compor o aplicativo como um todo.

O material a seguir fornece apenas informações de referência, sem grande preocupação com a sequência didática. Este material destina-se unicamente aos leitores que já possuem conhecimentos prévios sobre programação de computadores.

Alguns pontos podem ser esclarecidos:

- O compilador utilizado foi o Borland C++ 5.0 para Windows. Embora a maior parte do código, com exceção do nível da aplicação em si, seja independente do ambiente, utilizam-se certas bibliotecas padrão Borland. São estas as classes que encapsulam a API do Windows e fornecem formas de acesso aos arquivos em disco, de gerenciamento de memória, entre outros. No caso de se desejar portar o código inteiro para outro compilador, estas partes do código normalmente têm que ser modificadas, seguindo as instruções de conversão contidas no próprio compilador.
- Esta exposição não é considerada o objetivo primário deste trabalho. Mostra-se aqui uma forma de implementar um programa de resolução de pórticos planos através da Programação Orientada a Objetos, mas apenas como forma de divulgação desta metodologia. Não foi desenvolvido um modelo voltado diretamente à portabilidade, por usar elementos que dependem do compilador.
- Caso se desejasse o desenvolvimento de um núcleo de solução realmente genérico, algumas modificações deveriam ser efetuadas, mas aproveitando a maior parte das classes desenvolvidas.

Não se pretende fazer aqui uma explanação sobre a sintaxe C++. Para os leitores que estão acostumados ao uso de outras linguagens como BASIC e FORTRAN, recomenda-se, além da compreensão do modelo de objetos exposto no Capítulo 4, que se atentem para alguns pontos básicos, que representam diferenças entre estas linguagens e o C++:

- Uma instrução pode estar contida em várias linhas. O que define o fim da instrução é o ";".
- O código C++ é dito sensível ao caso, ou seja, caracteres em maiúsculas são diferentes daqueles em minúsculas, podendo, por exemplo, conter no mesmo escopo uma variável "A" e outra "a".
- As linhas que iniciam por "//" são comentários.
- As estruturas de controle de fluxo são similares às existentes em outras linguagens: blocos IF, FOR e WHILE representam o mesmo que em outras linguagens.
- Deve-se entender as variáveis como sendo objetos ao invés de tipos simples. Cada objeto pode ter associado a este um número qualquer de métodos.
- Utiliza-se largamente referências e ponteiros de memória. Uma explanação sobre este recurso foi feita no Capítulo 4.
- Cada classe possui um conjunto de dados e um conjunto de métodos. Tanto um como outro podem ser definidos como acessíveis externamente à classe (métodos públicos) ou acessíveis apenas internamente (métodos protegidos ou privados). Esta separação é altamente recomendável, por facilitar a compreensão do código e por permitir o avanço em relação ao uso de metodologia padrão de desenvolvimento de componentes.

Na sintaxe C++, existe uma separação no código de cada classe:

- Um arquivo, normalmente com a extensão ".h", define o "header" da classe. Esta representa a definição da classe, contendo todos os seus dados e métodos, tanto públicos (abaixo da seção "public") como privados (abaixo da seção "protected").
- A implementação de cada método pode estar contida em um ou mais arquivos, normalmente com a extensão ".CPP".

Em primeiro lugar, vai-se apresentar as classes utilizadas para o desenvolvimento do programa de análise linear de pórticos planos.

2.2 CLASSE PORTICO

A classe Portico representa o módulo de solução em si. Nesta classe, estão contidos os vetores de nós e barras que compõem o modelo, bem como os métodos que criam esta estrutura a partir dos arquivos em disco. Esta é a classe responsável pelo processamento da estrutura.

```
#ifndef __PORTICO_H
#define __PORTICO_H

#include "portico\no.h"
#include "portico\barra.h"

class _import ArquivoInput;
class _import ofstream;
class _import MatrizBanda;

class Portico
{
  public:
    Portico();
    virtual ~Portico();

    const VetorDeNos &Nos() const { return *nos; }
    VetorDeNos &Nos() { return *nos; }
    VetorDeBarras &Barras() { return *barras; }
    const VetorDeBarras &Barras() const { return *barras; }

    virtual bool Calcula();

//**** leitura de arquivo
    virtual bool LePorticoDeArquivo(const char *nomeArquivo);
    void LeDados(ArquivoInput& arquivo, int nNos, int nBarras);

//**** gravação de arquivo
    bool GravaResultadosEmArquivo(const char *nomeArquivo);
    bool ExportaDadosSAP(const char *nomeArquivo);
    bool ExportaDXF(const char *nomeArquivo);

//**** estabilidade global
    float CoeficienteGamaZ();
    float ParametroAlfa();

  protected:
    int NBarras() { return barras->GetItemsInContainer(); }
    int NNos() { return nos->GetItemsInContainer(); }
    int Ndj() { return 3; }   // número de vínculos possíveis em um nó
    int Ngl() { return NNos() * Ndj(); }
    int NglBarra() { return 2 * Ndj(); }
    int NLinhas() { return Ngl() - NumeroDeRestricoes(); }
    int NumeroDeRestricoes() const;
    int NumeroDeNosComRestricoes() const;
    float MenorDimensao();
    double MaiorFletor();
    double MaiorDeslocamento();
```

```
//**** métodos de cálculo
    virtual bool CriarMatrizes();
    virtual void DestruirMatrizes();
    bool CriarMatrizSFF();
    bool DadosEstruturais();
    bool MatrizDeRigidez();
    bool Carregamentos();
    bool Resultados();

//**** leitura de arquivo
    virtual Barra* CriaBarra();
    virtual SecaoTransversal* CriaSecao();
    virtual void AtribuiSecao(SecaoTransversal* secao, Barra* barra);
    virtual void LeCoordenadas(ArquivoInput& arquivo, int nNos);
    virtual void LeIncidencias(ArquivoInput& arquivo,int nBarras);
    virtual void LeSecoesBarras(ArquivoInput& arquivo);
    virtual void LeRestricoesApoio(ArquivoInput& arquivo);
    virtual void LeCargasDistribuidas(ArquivoInput& arquivo);
    virtual void LeCargasConcentradas(ArquivoInput& arquivo);

//**** estabilidade global
    float Momento2Ordem();
    float MomentoTombamento();
    float AlturaTotal();
    float CargaTotal();
    float EIequivalente();
    float FlechaTopo();

  protected:
    VetorDeNos *nos;
    VetorDeBarras *barras;
    double *df, *ac;
    MatrizBanda *matEst;
    bool calculoOk;
    int largBanda;
    bool *glRestrito;
    int *id;
    string tituloModelo;
}; // Portico

#endif __PORTICO_H
```

2.2.1 Métodos públicos para criação e processamento

Construtor da classe Portico. Inicializa dados necessários com valores vazios.

```
Portico::Portico()
{
  nos = new VetorDeNos(10,1,5);
  barras = new VetorDeBarras(10,1,5);
  matEst = NULL;
  calculoOk = false;
} // Portico
```

Destrutor da classe Portico. Limpa dados e libera memória.

```
Portico::~Portico()
{
  nos->Flush();
  delete nos;

  barras->Flush();
  delete barras;

} // ~Portico
```

Processa a estrutura. Caso a análise seja bem-sucedida, retorna valor verdadeiro e, caso, contrário, valor falso.

```
bool Portico::Calcula()
{
  calculoOk = false;
  if (!NBarras() || !NNos())
    return true;
  if (!DadosEstruturais())
    return false;
  if (!CriarMatrizSFF())
    return false;
  if (!MatrizDeRigidez()) {
    DestruirMatrizes();
    return false;
  }
  if (!matEst->FatorarMatrizBanda(NULL)) {
    ::MessageBox(NULL,"Erro na solução do sistema."
                 ,"Erro",MB_ICONSTOP | MB_OK);
    DestruirMatrizes();
    return false;
  }
  if (!Carregamentos()) {
    DestruirMatrizes();
    return false;
  }
  matEst->ResolverMatrizBanda(ac,df);
  if (!matEst->VerificarPrecisao(ac,df)) {
    ::MessageBox(NULL,"Erro de precisão numérica.",
                 "Erro",MB_ICONSTOP | MB_OK);
    DestruirMatrizes();
    return false;
  }
  if (!Resultados()) {
    DestruirMatrizes();
    return false;
  }
  DestruirMatrizes();
  calculoOk = true;
  return true;
} // Calcula
```

2.2.2 Métodos privados para processamento

Inicializa vetores de dados com o número de graus de liberdade definidos após a leitura.

```
bool Portico::CriarMatrizes()
{
  glRestrito = new bool(Ngl()+1);
  id = new int(Ngl()+1);
  df = new double(Ngl()+1);
  ac = new double(Ngl()+1);

  return true;
} // CriarMatrizes
```

Inicializa a matriz de rigidez global com o número de graus de liberdade e largura de banda definidos após a leitura.

```
bool Portico::CriarMatrizSFF()
{
  if (matEst)
    delete matEst;
  matEst = new MatrizBanda(NLinhas(),largBanda);
  return true;
} // CriarMatrizSFF
```

LISTAGENS (ANÁLISE LINEAR)

Elimina vetores de dados e libera a memória alocada.

```
void Portico::DestruirMatrizes()
{
  delete glRestrito;
  delete id;
  delete df;
  delete ac;
  if (matEst) {
    delete matEst;
    matEst = NULL;
  }
} // DestruirMatrizes
```

Retorna o número total de graus de liberdade restringidos.

```
int Portico::NumeroDeRestricoes() const
{
  int nr = 0,nNos = nos->GetItemsInContainer();
  for (int i = 1; i <= nNos; i++)
    nr += (*nos)[i]->NumeroDeRestricoes();
  return nr;
} // NumeroDeRestricoes
```

Retorna o número de nós que possuem alguma vinculação.

```
int Portico::NumeroDeNosComRestricoes() const
{
  int nrj = 0,nNos = nos->GetItemsInContainer();
  for (int i = 1; i <= nNos; i++)
    if ((*nos)[i]->PossuiRestricoes())
      nrj++;
  return nrj;
} // NumeroDeNosComRestricoes
```

Cria o vetor que mapeia os graus de liberdade (faz uma ordenação de tal forma que os graus vinculados passem para o final, ficando de fora da solução do sistema linear.

Calcula a largura de banda necessária para a matriz de rigidez global.

```
bool Portico::DadosEstruturais()
{
  if (!CriarMatrizes())
    return false;

  // vinculação dos nós
  for (int i = 1; i <= NNos(); i++) {
     int numero = (*nos)[i]->Numero();
    for (TDeslocamento j = DX; j <= RZ; j++)
       glRestrito[(numero - 1) * Ndj() + j] =
                          (*nos)[i]->Restr(j);
  }

  // índices das equações para cada grau de liberdade em
cada nó
  int n1 = 0;
  for (int j = 1; j <= Ngl(); j++) {
    n1 += glRestrito[j];
    if (!glRestrito[j])
      id[j] = j - n1;
    else
      id[j] = NLinhas() + n1;
  }

  // cálculo da largura de banda da matriz de rigidez
  int nbi;
  largBanda = 0;
  for (int i = 1; i <= NBarras(); i++) {
    nbi = Ndj() * (abs(
     (*barras)[i]->NoFinal().Numero() -
     (*barras)[i]->NoInicial().Numero()) + 1);
    if (nbi > largBanda)
      largBanda = nbi;
  }

  return true;
} // DadosEstruturais
```

Monta a matriz de rigidez global da estrutura.

Cada barra é responsável por fornecer sua matriz de rigidez já referida ao sistema de coordenadas globais, sendo feita aqui apenas a transferência para a matriz de rigidez global.

```
bool Portico::MatrizDeRigidez()
{
  Matriz* sff = matEst->Sff();

  for (int i = 1; i <= NBarras(); i++) {
     MatrizBarra* sms = new MatrizBarra();
     Barra* barra = (*barras)[i];
     barra->MatrizDeRigidezGlobal(sms);

    // transferência para a matriz de rigidez
    for (int j = 1; j <= NglBarra(); j++) {
      int i1 = barra->GlGlobal(j);
      if (!glRestrito[i1]) {
        for (int k = j; k <= NglBarra(); k++) {
          int i2 = barra->GlGlobal(k);
          if (!glRestrito[i2]) {
            int ir = id[i1];
            int ic = id[i2];
            if (ir >= ic) {
              int item = ir;
              ir = ic;
              ic = item;
            }
            ic = ic - ir + 1;
            (*sff)[ir][ic] += (*sms)[j][k];
          }
        }
      }
    }
      delete sms;
  } // for - total de barras

  return true;
} // MatrizDeRigidez
```

Monta os vetores de carregamentos.

A parcela de cargas nodais (AJ) é retirada diretamente dos dados dos nós, enquanto que a parcela referente aos esforços de imobilização (AE) nas barras é preenchida pelas próprias barras.

As duas parcelas são somadas para montar o vetor de termos independentes AC.

```
bool Portico::Carregamentos()
{
  double *aj = new double(Ngl()+1);
  double *ae = new double(Ngl()+1);

  // preenche vetor AJ
  int numero;
  for (int i = 1; i <= NNos(); i++) {
    numero = (*nos)[i]->Numero();
    for (TDeslocamento j = DX; j <= RZ; j++)
      aj[(numero-1)*Ndj()+j]=(*nos)[i]->Carga(j);
  }

  for (int i = 1; i <= NBarras(); i++) {
     Barra* barra = (*barras)[i];
     barra->PreencheAcoesEngPerf(ae);
  }

  // Carregamentos nos nós combinados
  int jr;
  for (int j = 1; j <= Ngl(); j++) {
    jr = id[j];
    ac[jr] = aj[j] + ae[j];
  }

  delete ae;
  delete aj;
  return true;
} // Carregamentos
```

Atribui os deslocamentos cálculados aos nós e obtém os esforços internos às barras. Cada barra é responsável por definir seus esforços com base nos deslocamentos calculados globalmente.

```
bool Portico::Resultados()
{
  double *dj = new double(Ngl()+1);

  for (int k = 1; k <= Ngl(); k++)
    dj[k] = df[id[k]];

  for (int j = 1; j <= NNos(); j++)
    for (TDeslocamento k = DX; k <= RZ; k++)
      (*nos)[j]->Deslocamento(k,dj[Ndj()*j-(Ndj()-k)]);

  for (int i = 1; i <= NBarras(); i++) {
    Barra* barra = (*barras)[i];
    barra->ObtemEsforcos(dj);
  }

  delete dj;
  return true;
} // Resultados
```

2.2.3 Métodos para a leitura de arquivos de dados

Método público que preenche os dados do pórtico a partir de um arquivo de dados. Utiliza como parâmetro o nome do arquivo.

Lê os parâmetros principais e chama um método privado para ler o restante.

```
bool Portico::LePorticoDeArquivo(const char *nomeArquivo)
{
  nos->Flush();
  barras->Flush();
  ArquivoInput arquivo(nomeArquivo);
  if(!arquivo.AlocacaoOK()) {
    ::MessageBox(NULL,"Não foi possível abrir
                 o arquivo.","Erro",MB_ICONSTOP | MB_OK);
    return false;
  }

  tituloModelo = arquivo.ReadLine();

  int nNos = arquivo.ReadBlockInteger();
  int nBarras = arquivo.ReadBlockInteger();

  LeDados(arquivo, nNos, nBarras);

  return true;
} // LePorticoDeArquivo
```

Os demais métodos a seguir são protegidos, ou seja, acessíveis apenas internamente. Deve-se notar o uso de uma classe auxiliar para encapsular a leitura do arquivo de dados em formato texto.

Cria uma nova barra do pórtico (método a ser redefinido nas classes derivadas).

```
Barra* Portico::CriaBarra()
{
  return new Barra();
} // CriaBarra
```

Cria uma nova seção transversal.

```
SecaoTransversal* Portico::CriaSecao()
{
  return new SecaoTransversal();
} // CriaSecao
```

Atribui a seção transversal à barra.

```
void Portico::AtribuiSecao(SecaoTransversal* secao,
Barra* barra)
{
  barra->Secao(new SecaoTransversal(*secao));
} // CriaSecao
```

Lê os dados do arquivo (subdivide a tarefa em métodos menores).

```
void Portico::LeDados(ArquivoInput& arquivo, int nNos,
int nBarras)
{
  LeCoordenadas(arquivo, nNos);
  LeIncidencias(arquivo, nBarras);
  LeSecoesBarras(arquivo);
  LeRestricoesApoio(arquivo);
  LeCargasDistribuidas(arquivo);
  LeCargasConcentradas(arquivo);
} // LeDados
```

Lê a seção de coordenadas nodais e preenche o vetor de nós do pórtico.

```
void Portico::LeCoordenadas(ArquivoInput& arquivo, int
nNos)
{
  arquivo.SkipLine();
  float x, y;
  for(WORD i = 1; i <= nNos; i++) {
    x = arquivo.ReadFloat();
    y = arquivo.ReadFloatNewLine();
    No* no = new No;
    no->Numero(i);
    no->X(x);
    no->Y(y);
    nos->Add(no);
  }
} // LeCoordenadas
```

Lê a seção de incidências das barras e preenche o vetor de barras do pórtico.

```
void Portico::LeIncidencias(ArquivoInput& arquivo,int
nBarras)
{
  arquivo.SkipLine();
  for(WORD i = 1; i <= nBarras; i++) {
    int noI = arquivo.ReadInteger();
    int noF = arquivo.ReadIntegerNewLine();
    Barra* barra = CriaBarra();
    barra->Numero(i);
    barra->NoInicial((*((*nos)[noI])));
    barra->NoFinal((*((*nos)[noF])));
    barras->Add(barra);
  }
}
```

Lê a seção de propriedades das barras. Atribui seções transversais às barras.

```
void Portico::LeSecoesBarras(ArquivoInput& arquivo)
{
  arquivo.SkipLine();
  int nSecoes = arquivo.ReadIntegerNewLine();
  for(WORD i = 1; i <= nSecoes; i++) {
    int barraI = arquivo.ReadInteger();
    int barraF = arquivo.ReadInteger();
    SecaoTransversal* secao = CriaSecao();
    secao->LeSecao(arquivo);
    for (int b = barraI; b <= barraF; b++)
      AtribuiSecao(secao, Barras()[b]);
    delete secao;
  }
} // LeSecoesBarras
```

Lê a seção de restrições de apoio. Atribui as condições de vinculação aos nós.

```
void Portico::LeRestricoesApoio(ArquivoInput& arquivo)
{
  arquivo.SkipLine();
  int nRestr = arquivo.ReadIntegerNewLine();
  for(int i = 1; i <= nRestr; i++) {
    int numero = arquivo.ReadInteger();
    int codx = arquivo.ReadInteger();
    int cody = arquivo.ReadInteger();
    int codz = arquivo.ReadIntegerNewLine();

    (*nos)[numero]->Restr(DX,codx);
    (*nos)[numero]->Restr(DY,cody);
    (*nos)[numero]->Restr(RZ,codz);
  }
} // LeRestricoesApoio
```

Lê a seção de cargas distribuídas e atribui seus valores sobre as barras do pórtico.

```
void Portico::LeCargasDistribuidas(ArquivoInput& arquivo)
{
  arquivo.SkipLine();
  int nCargas = arquivo.ReadIntegerNewLine();
  for(int i = 1; i <= nCargas; i++) {
    int barraI = arquivo.ReadInteger();
    int barraF = arquivo.ReadInteger();
    float carga = arquivo.ReadFloatNewLine();

    for(int b = barraI; b <= barraF; b++)
      (*barras)[b]->Carga(carga);
  }
} // LeCargasDistribuidas
```

Lê a seção de carregamentos concentrados e atribui seus valores sobre os nós do pórtico.

```
void Portico::LeCargasConcentradas(ArquivoInput& arquivo)
{
  arquivo.SkipLine();
  int nCargasConc = arquivo.ReadIntegerNewLine();
  for(int i = 1; i <= nCargasConc; i++) {
    int numero = arquivo.ReadInteger();
    float fx = arquivo.ReadFloat();
    float fy = arquivo.ReadFloat();
    float mz = arquivo.ReadFloatNewLine();

    (*nos)[numero]->Carga(DX,fx);
    (*nos)[numero]->Carga(DY,fy);
    (*nos)[numero]->Carga(RZ,mz);
  }
} // LeIncidencias
```

2.2.4 Método para gravação do arquivo de resultados

Este método grava o arquivo de resultados do processamento do pórtico. Recebe como parâmetro o nome do arquivo onde o relatório será gravado. A primeira seção grava os deslocamentos ocorridos em cada um dos nós e a segunda seção grava os esforços internos ocorridos em cada uma das barras.

```
bool Portico::GravaResultadosEmArquivo(const char *nomeArquivo)
{
  ofstream arquivo(nomeArquivo);
  if (!arquivo)
    return false;

  arquivo << "****************************************" << endl;
  arquivo << tituloModelo << endl;
  arquivo << "RESULTADOS DO CÁLCULO DE PÓRTICO ESPACIAL" << endl;
  arquivo << "****************************************" << endl;
  arquivo << endl;

  // Deslocamentos nos nós
  char buf[80];
  arquivo << "-------------------------------------------------" << endl;
  arquivo << "|           Deslocamento final dos nós           |" << endl;
  arquivo << "-------------------------------------------------" << endl;
  arquivo << "| Número |      X      |      Y      |      Z      |" << endl;
  arquivo << "-------------------------------------------------" << endl;

  for (int i = 1; i <= NNos(); i++) {
    sprintf(buf, "%7i", (*nos)[i]->Numero());
    arquivo << "|" << buf;
    for (TDeslocamento j = DX; j <= RZ; j++) {
      sprintf(buf, "%12.4f", (*nos)[i]->Deslocamento(j));
      arquivo << " |" << buf;
    }
    arquivo << " |" << endl;
  }
  arquivo << "-------------------------------------------------" << endl;
  arquivo << endl << endl;

  // Esforços nas barras
  arquivo << "------------------------------------------------------------------------------
------" << endl;
  arquivo << "|                                                    Esforços nas barras
|" << endl;
  arquivo << "------------------------------------------------------------------------------
------" << endl;
  arquivo << "| Número |     Axial     | Cortante I. | Cortante F. |  Fletor I.  |  Fletor
F.  |" << endl;
  arquivo << "------------------------------------------------------------------------------
------" << endl;
  for (int i = 1; i <= NBarras(); i++) {
    sprintf(buf, "%7i", (*barras)[i]->Numero());
    arquivo << "|" << buf;
    sprintf(buf, "%12.2f", (*barras)[i]->Esforco(ESF_AXIAL_I));
    arquivo << " |" << buf;
    sprintf(buf, "%12.2f", (*barras)[i]->Esforco(ESF_CORTANTE_I));
    arquivo << " |" << buf;
    sprintf(buf, "%12.2f", (*barras)[i]->Esforco(ESF_CORTANTE_F));
    arquivo << " |" << buf;
    sprintf(buf, "%12.2f", (*barras)[i]->Esforco(ESF_FLETOR_I));
    arquivo << " |" << buf;
    sprintf(buf, "%12.2f", (*barras)[i]->Esforco(ESF_FLETOR_F));
    arquivo << " |" << buf;
    arquivo << " |" << endl;
  }
  arquivo << "------------------------------------------------------------------------------
------" << endl;

  if (arquivo.good()) {
    arquivo.close();
    return true;
  } else
    return false;
} // GravaResultadosEmArquivo
```

2.2.5 Método para gravação do arquivo de exportação SAP90

Este método grava o arquivo correspondente a todos os dados do pórtico, formatado para leitura no programa de Elementos Finitos SAP90. Recebe como parâmetro o nome do arquivo onde o relatório será gravado.

Para mais informações, consulte a documentação deste programa. Este exemplo apenas ilustra uma das potencialidades da exportação de arquivos em formato texto.

```
bool Portico::ExportaDadosSAP(const char *nomeArquivo)
{
  ofstream file(nomeArquivo);
  if (!file.good()) {
    return false;
  }
  char buf[80];

  // cabeçalho
  file << tituloModelo << endl;
  file << "SYSTEM" << endl << "L=1" << endl << endl;

  // nós
  file << "JOINTS" << endl;
  VetorDeNosIterator itNos(*nos);
  while (itNos) {
    No* corrente = itNos++;
    sprintf(buf, "%4i", corrente->Numero());
    file << buf  << "    X=";
    sprintf(buf, "%-8.2f", corrente->X());
    file << buf  << "    Y=";
    sprintf(buf, "%-8.2f", corrente->Y());
    file << buf << endl;
  }

  // restrições dos nós
  file << endl << "RESTRAINTS" << endl;
  int nNos = nos->GetItemsInContainer();
  file << "   1";
  sprintf(buf, "%4i", nNos);
  file << buf << "    R=0,0,1,0,0,1"  << endl;
  itNos.Restart();
  while (itNos) {
    No* corrente = itNos++;
    if (corrente->PossuiRestricoes()) {
      sprintf(buf, "%4i", corrente->Numero());
      file << buf  << "        R=";
      for (TDeslocamento j = DX; j <= RZ; j++) {
        file << corrente->Restr(j);
        if (j < Ndj())
          file << ",";
      }
      file << endl;
    }
  }


  // barras
  file << endl << "FRAME" << endl;
```

LISTAGENS (ANÁLISE LINEAR)

```
  // características físicas
  int nBarras = barras->GetItemsInContainer();
  file << "NM=" << (int)nBarras << "        NL=" << (int)nBarras
       << endl << "C" << endl;
  VetorDeBarrasIterator itBarras(*barras);
  while (itBarras) {
    Barra* corrente = itBarras++;
    sprintf(buf, "%4i", corrente->Numero());
    file << buf  << "    A=";
    sprintf(buf, "%-8.2f", corrente->Area());
    file << buf  << "    I=";
    sprintf(buf, "%-10.2f", corrente->Inercia());
    file << buf  << "    E=";
    sprintf(buf, "%-10.2f", corrente->ModuloE());
    file << buf  << "    AS=0" << endl;
  }
  file << "C" << endl;


  itBarras.Restart();
  while (itBarras) {
    Barra* corrente = itBarras++;
    file << (int)corrente->Numero() << "        WL=0,0," <<
            -corrente->Carga() << endl;
  }
  file << "C" << endl;

  // incidência das barras
  itBarras.Restart();
  while (itBarras) {
    Barra* corrente = itBarras++;
    file << (int)corrente->Numero() << "        " <<
        (int)corrente->NoInicial().Numero() << "        " <<
        (int)corrente->NoFinal().Numero() << "        M=" <<
        (int)corrente->Numero() << "        NSL=" <<
        (int)corrente->Numero() << "        LP=1,0" << endl;
  }


    // cargas concentradas nos nós
  file << endl << "LOADS" << endl;
  itNos.Restart();
  while (itNos) {
    No* corrente = itNos++;
    if (corrente->Carga(DX) || corrente->Carga(DY) || corrente->Carga(RZ)) {
      file << (int)corrente->Numero() << "        F=";
      for (TDeslocamento j = DX; j <= RZ; j++) {
        file << corrente->Carga(j);
        if (j < Ndj())
          file << ",";
      }
      file << endl;
    }
  }
  file.close();
  return false;
} // ExportaDadosSAP
```

2.2.6 Métodos para exportação dos arquivos gráficos DXF

Este método público cria um arquivo gráfico formato DXF contendo: estrutura indeformada (a partir das coordenadas das barras), estrutura deformada (gerada pela própria barra) e momentos fletores (também gerados pelas próprias barras).

```
Bool Portico::ExportaDXF(const char *nomeArquivo)
{
  ArquivoDXF *dxf = new ArquivoDXF(nomeArquivo);
  if (!dxf->AlocacaoOK()) {
    delete dxf;
    return false;
  }
  // estrutura
  dxf->LayerCorrente(0);
  dxf->CorCorrente(15);
  VetorDeBarrasIterator itBarras(*barras);
  while (itBarras) {
    Barra* corrente = itBarras++;
    float width = 3.0;
    dxf->Line(corrente->NoInicial(),
              corrente->NoFinal(), width);
  }

  // deformada
  dxf->LayerCorrente(1);
  dxf->CorCorrente(13);
  itBarras.Restart();
  float ampliacao =
        (0.3*MenorDimensao())/MaiorDeslocamento();

  while (itBarras) {
    Barra* corrente = itBarras++;
    corrente->DeformadaDXF(dxf,ampliacao);
  }

  // momentos fletores
  dxf->LayerCorrente(2);
  dxf->CorCorrente(12);
  itBarras.Restart();
  ampliacao = (0.4*MenorDimensao())/MaiorFletor();
  while (itBarras) {
    Barra* corrente = itBarras++;
    corrente->MomentoDXF(dxf, ampliacao);
  }

  delete dxf;
  return true;
}
```

A escala relativa dos diagramas de deslocamento e momentos fletores é definida com base nas dimensões da edificação. Os métodos a seguir são privados e auxiliam na definição dos fatores de escala.

Retorna a menor dimensão do retângulo que circunscreve a geometria da estrutura.

```
float Portico::MenorDimensao()
{
  double menorX = MAXFLOAT, maiorX = -MAXFLOAT,
         menorY = MAXFLOAT, maiorY = -MAXFLOAT;
  VetorDeNosIterator itNos(*nos);
  while (itNos) {
    No* corrente = itNos++;
    menorX = min(menorX, corrente->X());
    maiorX = max(maiorX, corrente->X());
    menorY = min(menorY, corrente->Y());
    maiorY = max(maiorY, corrente->Y());
  }
  float dx = maiorX-menorX,
        dy = maiorY-menorY;
  return  (dx > 0.1)&&(dy > 0.1) ?
          min(dx, dy) : max(dx, dy);
} // MenorDimensao
```

Retorna o maior momento fletor (em módulo) ocorrido em todas as barras do pórtico. Usado para definir a escala relativa de desenho dos momentos fletores.

```
double Portico::MaiorFletor()
{
  VetorDeBarrasIterator itBarras(*barras);
  double maiorFletor = 0;
  while (itBarras) {
    Barra* corrente = itBarras++;
    maiorFletor = max(maiorFletor,
                      fabs(corrente->Esforco(ESF_FLETOR_I)));
    maiorFletor = max(maiorFletor,
                      fabs(corrente->Esforco(ESF_FLETOR_F)));
  }
  return maiorFletor;
} // MaiorFletor
```

Retorna o maior deslocamento (em módulo) ocorrido em todos os nós do pórtico. Usado para definir a escala relativa de desenho da estrutura deformada..

```
double Portico::MaiorDeslocamento()
{
  double maiorD = 0;
  VetorDeNosIterator itNos(*nos);
  while (itNos) {
    No* corrente = itNos++;
    double dx = fabs(corrente->X() -
                corrente->XFinal(1.0));
    double dy = fabs(corrente->Y() -
                corrente->YFinal(1.0));
    maiorD = max(maiorD, sqrt(dx*dx+dy*dy));
  }
  return maiorD;
} // MaiorDeslocamento
```

2.2.7 Métodos públicos para definição simplificada dos efeitos de 2ª ordem

Retorna o valor do Coeficiente Gama-Z, com base nos resultados obtidos pela análise linear da estrutura.

```
float Portico::CoeficienteGamaZ()
{
  float gamaZ;
  float mom2ordem = Momento2Ordem();
  float momTombamento = MomentoTombamento();
  if (momTombamento == 0)
    gamaZ = MAXFLOAT;      // não é possível avaliar
  else {
    float fator = mom2ordem / momTombamento;
    if (fator >= 1)
      gamaZ = MAXFLOAT;      // estrutura instável
    else
      gamaZ = 1 / (1-fator);
  }
  return gamaZ;
} // CoeficienteGamaZ
```

Retorna o valor do Parâmetro de Instabilidade Alfa, com base nos resultados obtidos pela análise linear da estrutura.

```
float Portico::ParametroAlfa()
{
  float alfa;
  float hTot = AlturaTotal();
  float N = CargaTotal();
  float EIeq = EIequivalente();
  if (EIeq == 0)
    alfa = MAXFLOAT;      // não é possível avaliar
  else {
    float fator = N / EIeq;
    if (fator <= 0)
      alfa = 0;      // tracionada
    else
      alfa = hTot * sqrt(fator);
  }
  return alfa;
} // CoeficienteGamaZ
```

2.2.8 Métodos privados para apoio à definição simplificada dos efeitos de 2ª ordem

Retorna o valor do momento total de segunda ordem calculado para a estrutura.

É calculado como a somatória do produto de todas as cargas verticais pelos respectivos deslocamentos horizontais dos seus pontos de aplicação.

```
float Portico::Momento2Ordem()
{
  double *ae = new double(Ngl()+1);
  // preenche vetor AE
  for (int i = 1; i <= NBarras(); i++) {
     Barra* barra = (*barras)[i];
     barra->PreencheAcoesEngPerf(ae);
  }

  float mom2Ordem = 0;
  for (int i = 1; i <= NNos(); i++) {
     float cargaNo = (*nos)[i]->Carga(DY);
     float cargaBarra = ae[3*i-1];
     float desloc = (*nos)[i]->Deslocamento(DX);
     mom2Ordem += (cargaNo + cargaBarra) * desloc;
  }

  delete ae;
  return fabs(mom2Ordem);
} // Momento2Ordem
```

Retorna o valor do momento total de derrubamento calculado para a estrutura.

É calculado como a somatória do produto de todas as cargas horizontais pelos respectivos braços de alavanca dos seus pontos de aplicação.

```
float Portico::MomentoTombamento()
{
  float momTomba = 0;
  for (int i = 1; i <= NNos(); i++) {
     float cargaNo = (*nos)[i]->Carga(DX);
     float y = (*nos)[i]->Y();
     momTomba += cargaNo * y;
  }
  return fabs(momTomba);
} // MomentoTombamento
```

Retorna a altura total do modelo (considerando como altura a dimensão do modelo em Y).

```
float Portico::AlturaTotal()
{
  float yMin = MAXFLOAT, yMax = -MAXFLOAT;
  for (int i = 1; i <= NNos(); i++) {
     float y = (*nos)[i]->Y();
     yMin = min(yMin,y);
     yMax = max(yMax,y);
  }
  return (yMax - yMin);
} // AlturaTotal
```

Retorna a carga vertical total aplicada ao modelo.

```
float Portico::CargaTotal()
{
  double *ae = new double(Ngl()+1);
  // preenche vetor AE
  for (int i = 1; i <= NBarras(); i++) {
     Barra* barra = (*barras)[i];
     barra->PreencheAcoesEngPerf(ae);
  }

  float carga = 0;
  for (int i = 1; i <= NNos(); i++) {
     float cargaNo = -(*nos)[i]->Carga(DY);
     float cargaBarra = -ae[3*i-1];
     carga += (cargaNo + cargaBarra);
  }

  delete ae;
  return carga;
} // CargaTotal
```

Retorna a flecha que ocorreria no topo de um pilar engastado na base e livre no topo, com altura "L" e módulo de rigidez "EI", caso fosse aplicada uma força "F" na posição "a".

```
float FlechaCantilever(float F, float L, float a, float
EI)
{
  return (-F*a*a/EI)*(a/3 + (L-a)/2);
} // FlechaCantilever
```

Retorna o módulo de rigidez equivalente à estrutura como um todo. Para isto, a flecha ocorrida no topo é associada àquela que ocorreria com o mesmo carregamento em um pilar engastado na base e livre no topo.

```
float Portico::EIequivalente()
{
  float L = AlturaTotal();
  float flechaFicticia = 0;
  float EIfic = 1000;
  for (int i = 1; i <= NNos(); i++) {
     float cargaNo = (*nos)[i]->Carga(DX);
     float y = (*nos)[i]->Y();
     flechaFicticia += FlechaCantilever(
                       cargaNo, L, y, EIfic);
  }
  float EIeq = (flechaFicticia/FlechaTopo()) * EIfic;
  return fabs(EIeq);
} // EIequivalente
```

Retorna a flecha ocorrida no topo da edificação. Utiliza-se o maior deslocamento X ocorrido entre os nós superiores do modelo.

```
float Portico::FlechaTopo()
{
  float yMax = -MAXFLOAT;
  double flecha = 0;
  for (int i = 1; i <= NNos(); i++) {
     float y = (*nos)[i]->Y();
     if (y >= yMax) {
       yMax = y;
       flecha = max (flecha,
                fabs((*nos)[i]->Deslocamento(DX)) );
     }
  }
  return flecha;
} // AlturaTotal
```

2.3 CLASSE MATRIZBANDA

A classe MatrizBanda encapsula a questão da solução do sistema de equações lineares referentes à matriz de rigidez global da estrutura. Esta classe utiliza a propriedade de banda simétrica da matriz de rigidez, armazenando apenas a semi-banda da matriz. É criada, portanto, com base em dois parâmetros: o número de graus de liberdade e a largura de banda.

A matriz de rigidez global faz parte da classe e pode ser acessada externamente. Os coeficientes são preenchidos no método MatrizDeRigidez da classe Portico. Uma vez criada a matriz, pode-se fatorá-la (corresponde a fazer sua inversão, de acordo com o Método de Cholesky utilizado). Uma vez fatorada, pode-se resolver qualquer vetor de termos independentes, obtendo o vetor de incógnitas correspondentes, sem a necessidade de inverter novamente a matriz de rigidez.

```
#ifndef __MATRIZBANDA_H
#define __MATRIZBANDA_H

//---------------------------------------------------------------------------
#include "portico\matriz.h"
class _import StatusCalculo;
//---------------------------------------------------------------------------

class MatrizBanda
{
  public:
    MatrizBanda(long nLinhas,long nColunas);
    virtual ~MatrizBanda();
    Matriz* Sff() { return _sff; }

    bool FatorarMatrizBanda(StatusCalculo* status);
    void ResolverMatrizBanda(double *ac,double *df);
    bool VerificarPrecisao(double *ac,double *df);

  protected:
    bool CriaCopia();
    void LogarProgresso(StatusCalculo* status, int j);
void CalcularResiduoEmAC(double *ac,double *df,double *deltaAC,
    double &maiorDiferenca,double &valorAC,double &valorNovoAC);

  private:
    long nLinhas,nColunas;
    Matriz *_sff;
    Matriz *_copiaSff;

}; // MatrizBanda

#endif __MATRIZBANDA_H
```

2.3.1 Métodos para criação e destruição

Cria uma nova matriz nula, com o tamanho informado, onde nColunas representa a semi-banda da matriz de rigidez.

```
MatrizBanda::MatrizBanda (long _nLinhas,long _nColunas)
{
  nLinhas = _nLinhas;
  nColunas = _nColunas;
  _sff = new Matriz(_nLinhas,_nColunas);
  _copiaSff = NULL;
} // MatrizBanda
```

Apaga a matriz de rigidez e libera a memória alocada.

```
MatrizBanda::~MatrizBanda()
{
  delete _sff;
  if (_copiaSff)
    delete _copiaSff;
} // ~MatrizDinamica
```

2.3.2 Métodos públicos para resolução de sistemas de equações lineares

Fatora a matriz de rigidez. A operação de fatoração transforma a matriz de rigidez SFF em uma matriz triangular inferior.

Este procedimento pode ser considerado equivalente à inversão da matriz de rigidez, que fica armazenada em SFF.

Caso ocorra problema na fatoração, isto significa que a matriz de rigidez é singular e este método retorna erro.

```
bool        MatrizBanda::FatorarMatrizBanda(StatusCalculo*
status)
{
  if (!_copiaSff) CriaCopia();

  int j, j1, j2, i, i1, k;
  double sum, temp;

  MessageBeep(MB_ICONEXCLAMATION);


  if ((*_sff)[1][1] <= 0)
    return false;

  for (j = 2; j <= nLinhas; j++) {

    if (status) LogarProgresso(status, j);
    j1 = j - 1;
    j2 = j - nColunas + 1;
    if (j2 < 1)
      j2 = 1;

    if (j1 != 1) {
      for (i = 2; i <= j1; i++) {
        i1 = i - 1;
        if (i1 >= j2) {
          sum = (*_sff)[i][j - i + 1];
          for (k = j2; k <= i1; k++)
            sum = sum - (*_sff)[k][i - k + 1] *
                        (*_sff)[k][j - k + 1];
          (*_sff)[i][j - i + 1] = sum;
        }
      }
    }
    sum = (*_sff)[j][1];
    for (k = j2; k <= j1; k++) {
      temp = (*_sff)[k][j - k + 1]/(*_sff)[k][1];
      sum = sum - temp * (*_sff)[k][j - k + 1];
      (*_sff)[k][j - k + 1] = temp;
    }
    if (sum <= 0)
      return false;

    (*_sff)[j][1] = sum;
  }
  return true;
} // FatorarMatrizBanda
```

Recebe como parâmetro o vetor de termos independentes AC e o vetor de incógnitas DF. Com base na matriz de rigidez previamente invertida (ou seja, é necessário sempre ter chamado o método FatorarMatrizBanda antes deste), resolve o sistema de equações lineares e preenche o vetor DF.

Este método pode ser chamado várias vezes, passando-se diversos vetores AC e obtendo-se os correspondentes vetores DF, sem que seja necessário fazer novamente a inversão da matriz.

```
void  MatrizBanda::ResolverMatrizBanda(double  *ac,double
*df)
{
  int i, j, k, i1, k1, k2;
  double sum;

  // Guardar a coluna 1 de SFF para utilizar no segundo
loop
  double *sff1 = new double(nLinhas+1);

  for (i = 1; i <= nLinhas; i++) {
    j = i - nColunas + 1;
    if (i <= nColunas)
      j = 1;
    sum = ac[i];
    k1 = i - 1;

    for (k = j; k <= k1; k++)
      sum = sum - (*_sff)[k][i - k + 1] * df[k];
    df[i] = sum;

    sff1[i] = (*_sff)[i][1];
  }

  for (i = 1; i <= nLinhas; i++) {
    df[i] = df[i] / sff1[i];
  }

  delete sff1;

  for (i1 = 1; i1 <= nLinhas; i1++) {
    i = nLinhas - i1 + 1;

    j = i + nColunas - 1;
    if (j > nLinhas)
      j = nLinhas;
    sum = df[i];
    k2 = i + 1;
    double *SFFi = (*_sff)[i];
    for (k = k2; k <= j; k++)
      sum = sum - SFFi[k - i + 1] * df[k];

    df[i] = sum;
  }
} // ResolverMatrizBanda
```

Calcula a precisão numérica da solução obtida através do método ResolverMatrizBanda.
Recebe como parâmetro os dois vetores previamente calculados.

Verifica-se a precisão multiplicando-se o vetor de incógnitas DF pela matriz de rigidez original (armazenada em CopiaSff). O resultado deve ser igual ao próprio vetor AC. Define-se internamente uma diferença mínima aceitável de 1%.

Caso ocorra uma diferença superior a 1%, este método tenta refinar o resultado obtido, em um máximo de 3 iterações. Caso não seja possível, retorna erro.

```
bool MatrizBanda::VerificarPrecisao(double *ac,double *df)
{
  double *deltaAC = new double(nLinhas+1);
  double *deltaDF = new double(nLinhas+1);

  double tolerancia = 0.01;
  double maiorDiferenca = 0,valorAC = 0,valorNovoAC = 0;
  bool diferencaOk = false;
  int nRefinamentos = 3;

  for (int i=1;i <= nRefinamentos && !diferencaOk;i++) {
    CalcularResiduoEmAC(ac,df,deltaAC,
                maiorDiferenca,valorAC,valorNovoAC);

    // verificar tolerância
    if (maiorDiferenca > tolerancia) {
      if (maiorDiferenca > 1) {
        // diferença excessiva, parar refinamento
        i = nRefinamentos + 1;
      }

      if (i > 1) {
        delete deltaDF;
        deltaDF = new double(nLinhas+1);
      }

      // determinar vetor de correção
      ResolverMatrizBanda(deltaAC,deltaDF);

      // corrigir DF
      for (int i1 = 1;i1 <= nLinhas;i1++)
        df[i1] += deltaDF[i1];

    } else {
      diferencaOk = true;
    }
  }

  delete deltaAC;
  delete deltaDF;
  return diferencaOk;
} // VerificarPrecisao
```

2.3.3 Métodos privados auxiliares

Com base no vetor de termos independentes AC e no vetor de incógnitas DF obtido previamente, obtém o vetor de resíduo DeltaAC.

```
void MatrizBanda::CalcularResiduoEmAC(double *ac, double *df, double *deltaAC, double &maiorDiferenca, double &valorAC, double &valorNovoAC)
{
  // vetor resultado da multiplicação copiaSff * DF
  double *novoAC = new double(nLinhas+1);

  for (int i = 1;i <= nLinhas;i++) {
    int i1 = i - nColunas + 1; // linha inicial
    if (i1 < 1)
      i1 = 1;

    double sum = 0;
    for (int j = i1;j <= i;j++)
      sum += (df[j] * (*_copiaSff)[j][i - j + 1]);

    int k2 = min(nColunas,(nLinhas - i + 1));
    for (int j = 2;j <= k2;j++)
      sum += (df[i + j - 1] * (*_copiaSff)[i][j]);

    novoAC[i] = sum;
  }

  maiorDiferenca = 0;
  for (int i1 = 1;i1 <= nLinhas;i1++) {
    deltaAC[i1] = ac[i1] - novoAC[i1];
```

```
    if (fabs(ac[i1]) > 10) {
      double dif = fabs((novoAC[i1] - ac[i1]) / ac[i1]);
      if (dif > maiorDiferenca) {
        maiorDiferenca = dif;
        valorAC = ac[i1];
        valorNovoAC = novoAC[i1];
      }
    }
  }
  delete novoAC;
} // CalcularResiduoEmAC
```

Copia a matriz de rigidez SFF para CopiaSFF. Isto porque SFF será alterada na fatoração e a CopiaSFF é necessária para verificar a precisão numérica dos resultados.

```
bool MatrizBanda::CriaCopia()
{
  _copiaSff = new Matriz(*_sff);
  return true;
} // MatrizDinamica
```

Indica o progresso da fatoração. StatusCalculo é uma classe que encapsula um diálogo de progresso.

```
void MatrizBanda::LogarProgresso(StatusCalculo* status,
int j)
{
  int nLinhasAviso = 20;
  sBuffer.Init();
  if (j % nLinhasAviso == 0) {
    sBuffer << "Fatorando matriz: linha " << j
            << " de " << nLinhas << ends;
    status->AlterarMensagem(buffer);
  }
} // LogarProgresso
```

2.4 CLASSE BARRA

A classe Barra representa o comportamento de um elemento de barra de pórtico plano. Nesta classe, encapsula-se o comportamento do elemento, relativo à formulação adotada. Pode-se notar que toda a classe Portico independe da formulação adotada para as barras.

Caso se deseje, por exemplo, incluir o efeito das deformações por cortante, basta alterar (ou derivar) a classe Barra, sem modificar as classes Portico ou MatrizBanda.

LISTAGENS (ANÁLISE LINEAR)

```
#ifndef __BARRA_H
#define __BARRA_H

class _import ArquivoDXF;
class _import MatrizBarra;
class _import No;
class _import Point;
class _import SecaoTransversal;

//---------------------------------------------------------------------------
typedef enum { ESF_AXIAL_I=1, ESF_CORTANTE_I, ESF_FLETOR_I, ESF_AXIAL_F,
               ESF_CORTANTE_F, ESF_FLETOR_F } TEsforco;

//---------------------------------------------------------------------------
class Barra
{
  public:
   Barra();
    Barra(No& _noInicial,No& _noFinal,int _numero = 0);
    Barra& operator =(const Barra& outraBarra);
    int operator ==(const Barra& outraBarra);
    void CopiaDados(const Barra& outraBarra);
    virtual Barra* CriaCopia();
    virtual ~Barra();

    // funções de acesso aos dados
    int Numero() const { return numero; }
    void Numero(int _numero) { numero = _numero; }

    No& NoInicial() { return *noInicial; }
    void NoInicial(No& _no)  { noInicial = &_no; }
    No& NoFinal() { return *noFinal; }
    void NoFinal(No& _no)  { noFinal = &_no; }

    SecaoTransversal* Secao() { return secao; }
    void Secao(SecaoTransversal* _secao) { secao = _secao; }

    float Carga() const { return carga; }
    void Carga(float valor) { carga = valor; }

    float Area();
    float Inercia();
    float ModuloE();

    float Extensao() const;
    bool Vertical();
    double Angulo();

    float Esforco(TEsforco indice) { return esforco[indice]; }
    void Esforco(TEsforco indice, float valor){ esforco[indice]=valor; }

    // funções de montagem da matriz de rigidez
    int GlGlobal(int glLocal);
    void PreencheAcoesEngPerf(double *ae);
    void ObtemEsforcos(double *dj);
    void MatrizDeRigidezGlobal(MatrizBarra *sm,bool rotacionar=true);

    void DeformadaDXF(ArquivoDXF *dxf, float ampliacao);
    void MomentoDXF(ArquivoDXF *dxf, float ampliacao);
```

LISTAGENS (ANÁLISE LINEAR)

```
  protected:

    // funções de montagem da matriz de rigidez
    virtual float AcoesEngPerf(TEsforco indice);
    virtual void MatrizRigidezLocal(MatrizBarra *sm);
    void LadoSimetrico(MatrizBarra *sm);
    void MatrizTransformacao(MatrizBarra *matTrans,MatrizBarra *matTransposta);
    void PassaParaGlobal(MatrizBarra *sm, bool rotacionar);

    // funções de conversão para o sistema global
    Point ValorParaCoordenada(float xLocal, float valor, float ampliacao);

// funções de acesso aos esforços internos
    float Cortante(float x);
    float Momento(float x);
    float Giro(float x);
    float Flecha(float x);

  private:

    int numero;
    No *noInicial,*noFinal;
    SecaoTransversal* secao;
    float carga;        // carga distribuída
    float esforco[7];   // esforços internos (válido após cálculo)

}; // Barra
#endif __BARRA_H
```

2.4.1 Métodos para criação, destruição e cópia

Construtor default da classe Barra. Inicia com dados nulos.

```
Barra::Barra()
{
  noInicial = noFinal = NULL;
  numero = 0;
  carga = 0;
  secao = NULL;

  memset(esforco, 0x0, 7 * sizeof(float));
} // Barra
```

Construtor da barra que já recebe referências para os dois nós de apoio.

```
Barra::Barra(No& _noInicial,No& _noFinal,int _numero)
{
  noInicial = &_noInicial;
  noFinal = &_noFinal;
  numero = _numero;
  carga = 0;
  secao = NULL;

  memset(esforco, 0x0, 7 * sizeof(float));
} // Barra
```

Operador de cópia da Barra. Permite atribuir os dados de uma barra a outra.

```
Barra& Barra::operator=(const Barra& outraBarra)
{
  CopiaDados(outraBarra);
  return *this;
} // operator=
```

Método auxiliar para cópia dos dados da barra. Pode ser redefinido virtualmente, caso se incluam dados em uma classe derivada.

```
void Barra::CopiaDados(const Barra& outraBarra)
{
  numero = outraBarra.numero;
  noInicial = outraBarra.noInicial;
  noFinal = outraBarra.noFinal;
  carga = outraBarra.carga;

  memcpy(esforco, outraBarra.esforco, 7 * sizeof(float));

} // CopiaDados
```

Retorna uma nova barra como cópia da barra corrente.

```
Barra* Barra::CriaCopia()
{
  Barra* copia = new Barra();
  (*copia) = (*this);
  return copia;
} // CriaCopia
```

Operador de comparação da classe Barra.

```
int Barra::operator ==(const Barra& outraBarra)
{
  return numero == outraBarra.numero &&
         *noInicial == *(outraBarra.noInicial) &&
         *noFinal == *(outraBarra.noFinal);
} // operator==
```

Destrutor da classe Barra.

```
Barra::~Barra()
{
  if (secao)
    delete secao;
} // ~Barra
```

2.4.2 Métodos que retornam propriedades das barras

Retorna a área da seção transversal.

```
float Barra::Area()
{
  return secao->Area();
} // Area
```

Retorna o momento de inércia da seção transversal.

```
float Barra::Inercia()
{
  return secao->Inercia();
} // Inercia
```

Retorna o módulo de elasticidade, armazenado como propriedade da seção.

```
float Barra::ModuloE()
{
  return secao->ModuloE();
} // ModuloE
```

Retorna o comprimento da barra.

```
float Barra::Extensao() const
{
  return noInicial->Distance(*noFinal);
} // Extensao
```

Retorna verdadeiro se a barra for vertical, ou seja, se os dois nós possuem o mesmo valor de X (dentro de uma pequena tolerância).

```
bool Barra::Vertical()
{
  return (fabs(NoFinal().X() - NoInicial().X()) < 0.01);
} // Vertical
```

Retorna o ângulo da barra em relação ao sistema de coordenadas globais.

```
double Barra::Angulo()
{
  float xi = noInicial->X();
  float yi = noInicial->Y();
  float xf = noFinal->X();
  float yf = noFinal->Y();
  double ang = 0;
  double pi = acos(-1.0);
  if (xf > xi && yf == yi) ang = 0;
  if (xf < xi && yf == yi) ang = pi;
  if (xf == xi && yf > yi) ang = pi/2;
  if (xf == xi && yf < yi) ang = 3*pi/2;
  if (xf > xi && yf > yi) ang = atan((yf - yi) /
                                     (xf - xi));
  if (xf < xi && yf > yi) ang = pi - atan((yf - yi) /
                                          (xf - xi));
  if (xf < xi && yf < yi) ang = pi + atan((yf - yi) /
                                          (xf - xi));
  if (xf > xi && yf < yi) ang = 2*pi - atan((yf - yi) /
                                            (xf - xi));
  return ang;
}  // Angulo
```

2.4.3 Métodos para preenchimento dos vetores de forças

Método público que recebe o vetor de forças global AE e acrescenta a estes seus esforços de imobilização.

```
void Barra::PreencheAcoesEngPerf(double *ae)
{
  MatrizBarra *matTrans = new MatrizBarra();
  MatrizBarra *matTransposta = new MatrizBarra();
  MatrizTransformacao(matTrans,matTransposta);

  for (int lin=1; lin<=6; lin++) {
    for (TEsforco col=ESF_AXIAL_I;
              col<=ESF_FLETOR_F; col++)
      ae[GlGlobal(lin)] += (*matTransposta)[lin][col] *
                           AcoesEngPerf(col);
  }

  delete matTrans;
  delete matTransposta;
} // PreencheAcosEngPerf
```

Método privado que retorna o esforço de imobilização correspondente a um grau de liberdade da barra.

Está definido de forma a incluir apenas carregamentos uniformemente distribuídos.

```
float Barra::AcoesEngPerf(TEsforco indice)
{
  float valor = 0;
  switch (indice) {
    case ESF_AXIAL_I       : valor = 0;
                             break;
    case ESF_CORTANTE_I    : valor = carga * Extensao()/2;
                             break;
    case ESF_FLETOR_I : valor=carga*pow(Extensao(),2)/12;
                             break;
    case ESF_AXIAL_F       : valor = 0;
                             break;
    case ESF_CORTANTE_F : valor = carga * Extensao() /2;
                             break;
    case ESF_FLETOR_F: valor=-carga*pow(Extensao(),2)/12;
                             break;
  }
  return valor;
} // AcoesEngPerf
```

Método privado que retorna o número do grau de liberdade global referente a um grau de liberdade local. Utilizado para preencher corretamente o vetor de forças.

```
int Barra::GlGlobal(int glLocal)
{
   int jj = NoInicial().Numero(),
       jk = NoFinal().Numero();

   int im[7];
    for (int j = 1; j <= 3; j++)
      im[j] = 3 * jj - (3-j);
   for (int j = 1; j <= 3; j++)
     im[j + 3] = 3 * jk - (3-j);

   return im[glLocal];
} // GlGlobal
```

2.4.4 Métodos para definição da matriz de rigidez

Retorna a matriz de rigidez da barra. Caso o parâmetro "rotacionar" seja verdadeiro, retorna os valores referenciado ao sistema de coordenadas globais; caso contrário, ao sistema local.

```
void Barra::MatrizDeRigidezGlobal(MatrizBarra *mat, bool
rotacionar)
{
  MatrizRigidezLocal(mat);
  LadoSimetrico(mat);
  PassaParaGlobal(mat, rotacionar);
}  //MatrizDeRigidez
```

Preenche a matriz com os termos de rigidez referentes ao sistema de coordenadas local (apenas a triangular superior).

Este método representa a formulação da matriz de rigidez.

```
void Barra::MatrizRigidezLocal(MatrizBarra *matLocal)
{
  float i = Inercia();
  float e = ModuloE();
  float a = Area();
  float l = Extensao();
  (*matLocal)[1][1] = e * a / l;
  (*matLocal)[1][4] = -(*matLocal)[1][1];
  (*matLocal)[2][2] = 12 * e * i / pow(l,3);
  (*matLocal)[2][3] = 6 * e * i / pow(l,2);
  (*matLocal)[2][5] = -(*matLocal)[2][2];
  (*matLocal)[2][6] = (*matLocal)[2][3];
  (*matLocal)[3][3] = 4 * e * i / l;
  (*matLocal)[3][5] = -(*matLocal)[2][3];
  (*matLocal)[3][6] = 2 * e * i / l;
  (*matLocal)[4][4] = (*matLocal)[1][1];
  (*matLocal)[5][5] = (*matLocal)[2][2];
  (*matLocal)[5][6] = -(*matLocal)[2][3];
  (*matLocal)[6][6] = (*matLocal)[3][3];
} // MatrizRigidezLocal
```

Preenche o lado simétrico da matriz de rigidez.

```
void Barra::LadoSimetrico(MatrizBarra *matLocal)
{
  for (int i=2; i<=6; i++)
    for (int j=1; j < i; j++)
      (*matLocal)[i][j] = (*matLocal)[j][i];
} // LadoSimetrico
```

2.4.5 Métodos auxiliares para transformação de coordenadas

Preenche a matriz de transformação de coordenadas e sua transposta.

```
void Barra::MatrizTransformacao(MatrizBarra
*matTrans,MatrizBarra *matTransposta)
{
  (*matTrans)[1][1] = cos(Angulo());
  (*matTrans)[1][2] = sin(Angulo());
  (*matTrans)[2][1] = -(*matTrans)[1][2];
  (*matTrans)[2][2] = (*matTrans)[1][1];
  (*matTrans)[3][3] = 1;
  (*matTrans)[4][4] = (*matTrans)[1][1];
  (*matTrans)[4][5] = (*matTrans)[1][2];
  (*matTrans)[5][4] = -(*matTrans)[1][2];
  (*matTrans)[5][5] = (*matTrans)[1][1];
  (*matTrans)[6][6] = 1;

  for (int i=1; i<=6; i++) {
    for (int j=1; j <=6; j++) {
      (*matTransposta)[i][j] = (*matTrans)[j][i];
    }
  }
}  // MatrizTransformacao
```

Converte a matriz informada como parâmetro para o sistema de coordenadas globais.

```
void Barra::PassaParaGlobal(MatrizBarra *sm, bool rotacionar)
{
  MatrizBarra *mataux = new MatrizBarra();
  MatrizBarra *matTrans = new MatrizBarra();
  MatrizBarra *matTransposta = new MatrizBarra();

  MatrizTransformacao(matTrans,matTransposta);

  for (int c=1; c<=6; c++) {
    for (int d=1; d <=6; d++) {
      (*mataux)[c][d] = 0;
      for (int f=1; f <=6; f++) {
        (*mataux)[c][d] = (*mataux)[c][d] +
                ((*matTransposta)[c][f] * (*sm)[f][d]);
      }
    }
  }

  for (int c=1; c<=6; c++) {
    for (int d=1; d <=6; d++) {
      if (rotacionar) {
        (*sm)[c][d] = 0;
        for (int f=1; f <=6; f++) {
            (*sm)[c][d] = (*sm)[c][d] +
                    ((*mataux)[c][f] * (*matTrans)[f][d]);
        }
      }
      else
        (*sm)[c][d] = (*mataux)[c][d];
    }
  }

  delete mataux;
  delete matTrans;
  delete matTransposta;
} // PassaParaGlobal
```

2.4.6 Método para atribuir os esforços internos

Recebe como parâmetro o vetor de deslocamentos nodais calculados e obtém a partir destes os esforços internos nas barras.

```
void Barra::ObtemEsforcos(double *dj)
{
   MatrizBarra *smrt = new MatrizBarra();
   MatrizBarra *matTrans = new MatrizBarra();
   MatrizBarra *matTransposta = new MatrizBarra();
   MatrizTransformacao(matTrans,matTransposta);
   MatrizDeRigidezGlobal(smrt);

   double aux[7];
   for (TEsforco j=ESF_AXIAL_I; j <= ESF_FLETOR_F; j++) {
      aux[j] = 0;
      for (int k = 1; k <= 6; k++) {
        int i1 = GlGlobal(k);
        aux[j] = aux[j] + (*smrt)[j][k] * dj[i1];
      }
   }
   for (TEsforco j=ESF_AXIAL_I; j <= ESF_FLETOR_F; j++) {
      double amd[7];
      amd[j] = 0;
      for (int k = 1; k <= 6; k++)
        amd[j] = amd[j] + (*matTrans)[j][k] * aux[k];

      Esforco(j,-AcoesEngPerf(j) + amd[j]);
   }

   delete smrt;
   delete matTrans;
   delete matTransposta;
} // ObtemEsforcos
```

2.4.7 Métodos públicos para geração das saídas gráficas formato DXF

Desenha a posição deformada da barra no arquivo DXF.

```
void Barra::DeformadaDXF(ArquivoDXF *dxf, float ampliacao)
{
  dxf->Line(noInicial->PFinal(ampliacao),
            noFinal->PFinal(ampliacao));
} // DeformadaDXF
```

Desenha o diagrama de momentos fletores da barra no arquivo DXF.

```
void Barra::MomentoDXF(ArquivoDXF *dxf, float ampliacao)
{
  int nDiv = carga == 0 ? 1 : 20;
  float dx = Extensao() / nDiv;
  Point pAnt = *noInicial;

  for (int i=0; i<= nDiv; i++) {
    float xLocal = i*dx;
    float mi = -Momento(xLocal);
    Point p = ValorParaCoordenada(xLocal, mi, ampliacao);
    dxf->Line(pAnt, p);
    pAnt = p;
  }
  dxf->Line(pAnt, *noFinal);

} //MomentoDXF
```

2.4.8 Métodos auxiliares para geração das saídas gráficas formato DXF

Retorna o valor do esforço cortante na posição local X da barra.

```
float Barra::Cortante(float x)
{
  float v0 = Esforco(ESF_CORTANTE_I);
  float q = carga;
  float v = v0 + q*x;
  return v;
} // Deslocamento
```

Retorna o valor do momento fletor na posição local X da barra.

```
float Barra::Momento(float x)
{
  float m0 = -Esforco(ESF_FLETOR_I);
  float v0 = Esforco(ESF_CORTANTE_I);
  float q = carga;
  float m = m0 + v0*x + q*x*x/2;
  return m;
} // Deslocamento
```

Retorna o valor do giro na posição local X da barra.

```
float Barra::Giro(float x)
{
  float fi0 = noInicial->Deslocamento(RZ);
  float m0 = -Esforco(ESF_FLETOR_I);
  float v0 = Esforco(ESF_CORTANTE_I);
  float q = carga;
  float EI = secao->Inercia() * secao->ModuloE();
  float fi = fi0 + (m0*x + v0*pow(x,2)/2 +
                    q*pow(x,3)/6)/EI;
  return fi;
} // Deslocamento
```

Retorna o valor da flecha na posição local X da barra.

```
float Barra::Flecha(float x)
{
  Line retaBarra(*noInicial, *noFinal);
  float f0 = retaBarra.Distance(noInicial->PFinal(1.0));
  float fi0 = noInicial->Deslocamento(RZ);
  float m0 = -Esforco(ESF_FLETOR_I);
  float v0 = Esforco(ESF_CORTANTE_I);
  float q = carga;
  float EI = secao->Inercia() * secao->ModuloE();
  float f = f0 + fi0*x + (m0*pow(x,2)/2 + v0*pow(x,3)/6 +
                          q*pow(x,4)/24)/EI;
  return f;
} // Deslocamento
```

Dado um valor a ser plotado e a posição X local da barra onde este ocorre, retorna o ponto equivalente no espaço.

```
Point Barra::ValorParaCoordenada(float xLocal, float
valor, float ampliacao)
{
  Line retaBarra(*noInicial, *noFinal);
  Point pBarra = noInicial->PontoIntermediario(*noFinal,
                              xLocal/Extensao());
  Line retaValor = retaBarra.ParallelLine(
                   valor*ampliacao);
  Point pValor = retaValor.Projection(pBarra);
  return pValor;
} // ValorParaCoordenada
```

2.5 CLASSE NO

A classe No representa um nó da estrutura. Basicamente, o No é um ponto, ou seja, define a posição (X,Y) do nó. Incluem-se também como dados desta classe os carregamentos, as condições de apoio e os deslocamentos ocorridos em cada nó.

Esta classe é utilizada basicamente para fins de armazenamento de dados, facilitando sua manipulação, possuindo pouco comportamento próprio. Deve-se notar o uso de operadores, característica da linguagem C++, que permitem tratar uma classe de qualquer complexidade como se fosse um tipo simples.

```
#ifndef __NO_H
#define __NO_H

#include <classlib\arrays.h>
#include <portico\points.h>

//---------------------------------------------------------------------------
typedef enum {DX=1, DY, RZ} TDeslocamento;
//---------------------------------------------------------------------------

class No : public Point
{
  public:
    No();
    No(float _x, float _y, int _numero = 0);
    No(const No& outroNo);
    virtual No& operator =(const No& outroNo);
    virtual int operator ==(const No& outroNo) const;
    virtual int operator<(const No& outroNo);
    virtual ~No();

    int Numero() const { return numero; }
    void Numero(int _numero) { numero = _numero; }
```

```
    float XFinal(float ampliacao) const { return X()+ampliacao*deslocamento[DX]; }
    float YFinal(float ampliacao) const { return Y()+ampliacao*deslocamento[DY]; }
    Point PFinal(float ampliacao) const { return
Point(XFinal(ampliacao),YFinal(ampliacao)); }
   bool Restr(TDeslocamento indice) { return restr[indice]; }
    void Restr(TDeslocamento indice, bool valor) { restr[indice]=valor; }
    bool PossuiRestricoes() const;
    int NumeroDeRestricoes() const;
    void Vincula();
    void Libera();
    float Carga(TDeslocamento indice) { return carga[indice]; }
    void Carga(TDeslocamento indice, float valor) { carga[indice]=valor; }

    float Deslocamento(TDeslocamento indice) { return deslocamento[indice]; }
    void Deslocamento(TDeslocamento indice, float valor) { deslocamento[indice]=valor; }

    float Reacao(TDeslocamento indice) { return reacao[indice]; }
    void Reacao(TDeslocamento indice, float valor) { reacao[indice]=valor; }

  private:
    int numero;
    bool restr[4];   // vínculos de translação e rotação nas direções x, y e z

    float carga[4];        // força e momento aplicados no nó nas direções x, y e z
    float deslocamento[4]; // deslocamento final no nó (válido após cálculo)
    float reacao[4];       // reações de suporte no nó (válido após cálculo)

}; // No

#endif __NO_H
```

2.5.1 Métodos para a criação e destruição

Construtor default da classe No. Utilzia os dados definidos para sua classe base Point.

```
No::No():Point()
{
  numero = 0;
  memset(restr, 0x0, 4 * sizeof(bool));

  memset(carga, 0x0, 4 * sizeof(float));
  memset(deslocamento, 0x0, 4 * sizeof(float));
  memset(reacao, 0x0, 4 * sizeof(float));
} // No
```

Construtor da classe No que recebe como parâmetro as coordenadas e o número do nó.

```
No::No(float _x,float _y, int _numero) :
  Point(_x,_y),numero(_numero)
{
  memset(restr, 0x0, 4 * sizeof(bool));

  memset(carga, 0x0, 4 * sizeof(float));
  memset(deslocamento, 0x0, 4 * sizeof(float));
  memset(reacao, 0x0, 4 * sizeof(float));
} // No
```

Construtor da classe No que recebe outro No como parâmetro.

```
No::No(const No& outroNo)
{
  numero = outroNo.numero;
  X(outroNo.X());
  Y(outroNo.Y());
  memcpy(restr, outroNo.restr, 4 * sizeof(bool));

  memcpy(carga, outroNo.carga, 4 * sizeof(float));
  memcpy(deslocamento, outroNo.deslocamento,
         4 * sizeof(float));
  memcpy(reacao, outroNo.reacao, 4 * sizeof(float));
}
```

Operador de comparação da classe No.

```
int No::operator ==(const No& outroNo) const
{
  bool result = ( X() == outroNo.X() && Y() ==
outroNo.Y() );
  return result;
} // operator==
```

Operador de atribuição da classe No.

```
No& No::operator =(const No& outroNo)
{
  numero = outroNo.numero;
  X(outroNo.X());
  Y(outroNo.Y());
  memcpy(restr, outroNo.restr, 4 * sizeof(bool));

  memcpy(carga, outroNo.carga, 4 * sizeof(float));
  memcpy(deslocamento, outroNo.deslocamento,
         4 * sizeof(float));
  memcpy(reacao, outroNo.reacao, 4 * sizeof(float));

  return *this;
} // operator =
```

Operador de operação da classe No.

```
int No::operator<(const No& _no)
{
  return numero < _no.numero;
} // operator<
```

Destrutor da classe No.

```
No::~No()
{
} // ~No
```

2.5.2 Métodos que retornam as condições de vinculação

Retorna verdadeiro se o nó possuir alguma restrição de apoio.

```
bool No::PossuiRestricoes() const
{
  return NumeroDeRestricoes();
} // PossuiRestricoes
```

Retorna o número de restrições de apoio nos nós.

```
int No::NumeroDeRestricoes() const
{
  int nr = 0;
  for (TDeslocamento i = DX; i <= RZ; i++)
    nr += restr[i] ? 1 : 0;
  return nr;
} // NumeroDeRestricoes
```

2.6 CLASSES AUXILIARES

A seguir listam-se algumas outras classes definidas como apoio à definição desta implementação. Os métodos básicos de construção, destruição e operadores não serão comentados sempre que forem análogos aos definidos para as outras classes.

2.6.1 Classe Secao

Esta classe representa a seção transversal de uma barra, sendo utilizada para armazenar suas dimensões (B e H) e retornar suas propriedades geométricas (área, momento de inércia, etc).

```
#ifndef __SECAO_H
#define __SECAO_H

class _import ArquivoInput;
//---------------------------------------------------------------------------

class SecaoTransversal
{
  public:
    SecaoTransversal();
    SecaoTransversal(float _b, float _h, float _moduloE);
    SecaoTransversal(const SecaoTransversal& _secao);
    SecaoTransversal& operator =(const SecaoTransversal& _secao);
    virtual ~SecaoTransversal();

    float B() const { return b; }
    float H() const { return h; }
    void B(float  _b) { b = _b; }
    void H(float  _h) { h = _h; }

    float ModuloE() { return moduloE; }
    void ModuloE(float _moduloE) { moduloE = _moduloE; }

    float Inercia();
    float Area();

    void LeSecao(ArquivoInput& arquivo);
    virtual void LeLinhasAdicionais(ArquivoInput& arquivo, int nLinhas);

   protected:
    float  b, h;
    float moduloE;

}; // SecaoTransversal

//---------------------------------------------------------------------------
#endif __SECAO_H
```

Retorna o momento de inércia da seção retangular.

```
float SecaoTransversal::Inercia()
{
  return b*h*h*h/12.0;
} // ~SecaoTransversal
```

Retorna a área da seção retangular.

```
float SecaoTransversal::Area()
{
  return b*h;
} // ~SecaoTransversal
```

Lê os dados da seção a partir do arquivo de dados passado como parâmetro.

```
void SecaoTransversal::LeSecao(ArquivoInput& arquivo)
{
  b = arquivo.ReadFloat();
  h = arquivo.ReadFloat();
  moduloE = arquivo.ReadFloat();

  int nLinhas = arquivo.ReadIntegerNewLine();
  LeLinhasAdicionais(arquivo, nLinhas);
} // LeSecao
```

2.6.2 Classe ArquivoInput

Esta classe representa o arquivo de entrada de dados. Embora exista uma classe predefinida para manipulação de arquivos no Borland C++ (classe ifstream), foi interessante encapsular seu comportamento em uma classe para facilitar a conversão de textos para valores numéricos e para gerenciar a existência de linhas de cabeçalho.

```
#ifndef __ARQUIVO_INPUT_H
#define __ARQUIVO_INPUT_H

#include <fstream.h>
#include <cstring.h>

#define tamanhoLinha 80
//---------------------------------------------------------------------------

class ArquivoInput
{
  public:
    ArquivoInput(const char *nomeArquivo);
    ~ArquivoInput();

    bool AlocacaoOK() { return alocacaoOK; }

    float ReadBlockFloat();
    int ReadBlockInteger();

    float ReadFloat(bool fimLinha = false);
    float ReadFloatNewLine();
    int ReadInteger(bool fimLinha = false);
    int ReadIntegerNewLine();

    string ReadLine();
    void SkipLine();

  private:
     ifstream arquivo;
     bool alocacaoOK;

}; // ArquivoInput

//---------------------------------------------------------------------------

#endif __ARQUIVO_INPUT_H
```

Construtor da classe ArquivoInput. Recebe o nome do arquivo onde estão gravados os dados e tenta abri-lo para leitura. Caso não seja possível, indica o erro na variável AlocacaoOK, que pode ser acessada externamente.

```
ArquivoInput::ArquivoInput(
  const char *nomeArquivo
)
{
  arquivo.open(nomeArquivo);
  if (arquivo.good())
    alocacaoOK = true;
  else
    alocacaoOK = false;

} // ArquivoInput
```

Destrutor da classe ArquivoInput. Fecha o arquivo de dados.

```
ArquivoInput::~ArquivoInput()
{
  arquivo.close();
} // ~ArquivoInput
```

Lê um bloco composto por uma linha de comentário e um valor real.

```
float ArquivoInput::ReadBlockFloat()
{
  char coment[tamanhoLinha];
  arquivo.getline(coment,tamanhoLinha);
  char linha[tamanhoLinha];
  arquivo.getline(linha,tamanhoLinha);
  return atof(linha);
} // ReadBlockFloat
```

Lê um bloco composto por uma linha de comentário e um valor inteiro.

```
int ArquivoInput::ReadBlockInteger()
{
  char coment[tamanhoLinha];
  arquivo.getline(coment,tamanhoLinha);
  char linha[tamanhoLinha];
  arquivo.getline(linha,tamanhoLinha);
  return atoi(linha);
} // ReadBlockInteger
```

Lê um valor real do arquivo de dados. Considera-se que os valores estão separados por espaços ou por fins de linha.

```
float ArquivoInput::ReadFloat(bool fimLinha)
{
  char dado[tamanhoLinha];
  if (fimLinha)
    arquivo.getline(dado,tamanhoLinha);
  else {
    bool ok = false;
    while (!ok) {
      arquivo.get(dado,tamanhoLinha,' ');
      ok = (dado[0]!='\0');
      if (!ok)
        arquivo.get();
    }
  }
  return atof(dado);
} // ReadFloat
```

Lê um valor real do arquivo de dados e passa para a linha seguinte.

```
float ArquivoInput::ReadFloatNewLine()
{
  return ReadFloat(true);
} // ReadFloatNewLine
```

Lê um valor inteiro do arquivo de dados. Considera-se que os valores estão separados por espaços ou por fins de linha.

```
int ArquivoInput::ReadInteger(bool fimLinha)
{
  char dado[tamanhoLinha];
  if (fimLinha)
    arquivo.getline(dado,tamanhoLinha);
  else {
    bool ok = false;
    while (!ok) {
      arquivo.get(dado,tamanhoLinha,' ');
      ok = (dado[0]!='\0');
      if (!ok)
        arquivo.get();
    }
  }
  return atoi(dado);
} // ReadInteger
```

Lê um valor inteiro do arquivo de dados e passa para a linha seguinte.

```
int ArquivoInput::ReadIntegerNewLine()
{
  return ReadInteger(true);
} // ReadIntegerNewLine
```

Lê uma linha do arquivo de dados e a retorna sem conversão.

```
string ArquivoInput::ReadLine()
{
  string ret;
  char coment[tamanhoLinha];
  arquivo.getline(coment,tamanhoLinha);
  ret = coment;
  return ret;
} // ReadLine
```

Lê uma linha do arquivo de dados sem retorná-la (usado para pular linhas de cabeçalho).

```
void ArquivoInput::SkipLine()
{
  char coment[tamanhoLinha];
  arquivo.getline(coment,tamanhoLinha);
} // SkipLine
```

2.6.3 Classe ArquivoDXF

Esta classe representa o arquivo de saída gráfica em formato DXF. Para sua utilização, basta ser criado com o nome do arquivo desejado. A partir disto, podem ser utilizados métodos que simulam desenho, os quais gravarão os códigos equivalentes aos elementos gráficos em formato DXF.

Define-se uma cor e nível correntes para o desenho. Cada elemento inserido utilizará estes parâmetros correntes, até que sejam alterados.

A funcionalidade desta classe pode ser estendida facilmente para o desenho de elementos como círculos, arcos, textos, entre outros, bastando definir métodos de desenho que gravam seus códigos correspondentes. Utilizam-se aqui apenas linhas simples (elementos "LINE") e linhas com espessura (elementos "POLYLINE", que representam poligonais).

LISTAGENS (ANÁLISE LINEAR)

```
#ifndef __DXF_H
#define __DXF_H

#include <fstream.h>
class _import Point;

//---------------------------------------------------------------------------
#define BYLAYER -1
//---------------------------------------------------------------------------

class ArquivoDXF
{
  public:
    ArquivoDXF(const char *nomeArquivo);
    ~ArquivoDXF();

    int LayerCorrente() { return layerCorrente; }
    void LayerCorrente(int layer)  { layerCorrente=layer; }
    int CorCorrente() { return corCorrente; }
    void CorCorrente(int cor)  { corCorrente=cor; }
    bool AlocacaoOK() { return alocacaoOK; }

    void Header();
    void Layer(int numero, int cor);
    void End();
    void Line(float xi, float yi, float xf, float yf);
    void Line(Point pi, Point pf);
    void Line(float xi, float yi, float xf, float yf, float width);
    void Line(Point pi, Point pf, float width);

  private:
     ofstream arquivo;
     bool alocacaoOK;
     int layerCorrente;
     int corCorrente;

}; // ArquivoDXF

//---------------------------------------------------------------------------

#endif __DXF_H
```

Construtor da classe ArquivoDXF. Recebe o nome do arquivo onde estão gravados os dados e tenta abri-lo para leitura. Caso não seja possível, indica o erro na variável AlocacaoOK, que pode ser acessada externamente.

Incluir a seção de cabeçalho padrão no início do arquivo.

```
ArquivoDXF::ArquivoDXF(const char *nomeArquivo)
{
  alocacaoOK = true;
  arquivo.open(nomeArquivo);
  if (arquivo) {
    layerCorrente = 0;
    corCorrente = BYLAYER;
    Header();
  }
  else
    alocacaoOK = false;

} // ArquivoDXF
```

Destrutor da classe ArquivoDXF. Finaliza a última seção e fecha o arquivo.

```
ArquivoDXF::~ArquivoDXF()
{
  End();
  arquivo.close();
} // ~ArquivoDXF
```

Grava o cabeçalho do arquivo. Optou-se pela definição de um número constante de layers (10 layers, cada um com uma cor diferente).

```
void ArquivoDXF::Header()
{
  arquivo << "0" << endl
          << "SECTION" << endl
          << "2" << endl
          << "HEADER" << endl
          << "0" << endl
          << "ENDSEC" << endl;
  int maxLayer=10;
  arquivo << "0" << endl
          << "SECTION" << endl
          << "2" << endl
          << "TABLES" << endl
          << "0" << endl
          << "TABLE" << endl
          << "2" << endl
          << "LAYER" << endl
          << "70" << endl
          << maxLayer << endl;
  for (int i = 0; i<=maxLayer; i++)
    Layer(i, i + 3);

  arquivo << "0" << endl
          << "ENDTAB" << endl
          << "0" << endl
          << "ENDSEC" << endl
          << "0" << endl
          << "SECTION" << endl
          << "2" << endl
          << "ENTITIES" << endl;
} // Header
```

Método privado que grava a definição de um layer.

```
void ArquivoDXF::Layer(int numero, int cor)
{
  arquivo << "0" << endl
          << "LAYER" << endl
          << "2" << endl
          << numero << endl
          << "70" << endl
          << "0" << endl
          << "62" << endl
          << cor << endl
          << "6" << endl
          << "CONTINUOUS" << endl;
} // Layer
```

Método privado que grava o fim do arquivo.

```
void ArquivoDXF::End()
{
  arquivo << "0" << endl
          << "ENDSEC" << endl
          << "0" << endl
          << "EOF" << endl;
} // End
```

Método público que grava uma linha no arquivo. Utiliza como parâmetro as coordenadas X,Y iniciais e finais da linha e utiliza a cor e nível correntes.

```
void ArquivoDXF::Line(float xi, float yi, float xf, float
yf)
{
  arquivo << "0" << endl
          << "LINE" << endl
          << "8" << endl
          << layerCorrente << endl
          << "62" << endl;

  if (corCorrente == BYLAYER)
    arquivo << "BYLAYER" << endl;
  else
    arquivo << corCorrente << endl;

  arquivo << "10" << endl
          << xi << endl
          << "20" << endl
          << yi << endl
          << "11" << endl
          << xf << endl
          << "21" << endl
          << yf << endl;
} // Line
```

Método público que grava uma linha com espessura definida no arquivo. Utiliza como parâmetro as coordenadas X,Y iniciais e finais da linha e utiliza a cor e nível correntes.

Na verdade, é gravada uma poligonal com apenas um segmento, pois o padrão DXF não aceita espessuras para linhas.

```
void ArquivoDXF::Line(float xi, float yi, float xf, float
yf, float width)
{
  arquivo << "0" << endl
          << "POLYLINE" << endl
          << "8" << endl
          << layerCorrente << endl
          << "6" << endl
          << "CONTINUOUS" << endl
          << "62" << endl;

  if (corCorrente == BYLAYER)
    arquivo << "BYLAYER" << endl;
  else
    arquivo << corCorrente << endl;

  arquivo << "66" << endl
          << "1" << endl
          << "40" << endl
          << width << endl
          << "41" << endl
          << width << endl
          << "0" << endl
          << "VERTEX" << endl
          << "8" << endl
          << layerCorrente << endl
          << "10" << endl
          << xi << endl
          << "20" << endl
          << yi << endl
          << "0" << endl
          << "VERTEX" << endl
          << "8" << endl
          << layerCorrente << endl
          << "10" << endl
          << xf << endl
          << "20" << endl
          << yf << endl
          << "0" << endl
          << "SEQEND" << endl;
} // Line
```

Método público que grava uma linha a partir de dois pontos.

```
void ArquivoDXF::Line(Point pi, Point pf)
{
  Line(pi.X(),pi.Y(),pf.X(),pf.Y());
} // Line
```

Método público que grava uma linha com espessura a partir de dois pontos.

```
void ArquivoDXF::Line(Point pi, Point pf, float width)
{
  Line(pi.X(),pi.Y(),pf.X(),pf.Y(),width );
} // Line
```

2.7 CLASSE APLICATIVO

O conjunto de classes apresentadas constitui o núcleo de resolução de um programa de pórticos planos. A classe Portico, que representa o módulo de solução em si, possui métodos públicos que permitem sua criação a partir de um arquivo texto, seu processamento e a gravação dos resultados em outro arquivo texto.

A utilização desta classe poderia ser feita com qualquer tipo de interface externa. Para simplificar a utilização, foi associada com um programa editor de arquivos texto gerado através de um "assistente" AppExpert do Borland C++. A esta aplicação pronta foi acrescentado apenas o código necessário à solução do pórtico.

Cria o Portico a partir do arquivo corrente e o transfere para o método Processa.

```
void TPFrameApp::CmCalculaPortico()
{
  const char* nomeArquivo = ArquivoDados();
  if (nomeArquivo) {
    Portico* port = new Portico();
    port->LePorticoDeArquivo(nomeArquivo);
    Processa(port,nomeArquivo);
    delete port;
  }
} // CmCalculaPortico
```

Utiliza os diversos métodos públicos da classe Portico. Grava o modelo original em um arquivo SAP90 e, em seguida, efetua o processamento.

Caso o processamento seja bem-sucedido, grava os arquivos de resultados e de saída gráfica formato DXF.

O arquivo de resultados é aberto em uma nova janela utilizando-se a classe TdocTemplate definida nas bibliotecas do Borland.

```
void TPFrameApp::Processa(Portico* port, const char*
nomeArquivo)
{
  HCURSOR cursor = SetCursor(
    ::LoadCursor(NULL, IDC_WAIT));

  String sapFileName = MudaExtensao(
    nomeArquivo, ".sap");
  port->ExportaDadosSAP(sapFileName.c_str());

  TdocTemplate* tpl;
  If (port->Calcula()) {
    string resultFileName = MudaExtensao(
      nomeArquivo, ".sad");
    port->GravaResultadosEmArquivo(
      resultFileName.c_str());
    tpl = GetDocManager()->MatchTemplate(
      resultFileName.c_str());
    if (tpl)
      GetDocManager()->CreateDoc(
        tpl,resultFileName.c_str());

      resultFileName = MudaExtensao(
        nomeArquivo, ".dxf");
      port->ExportaDXF(
        resultFileName.c_str());

      sapFileName = MudaExtensao(
         nomeArquivo, ".sp2");
      port->ExportaDadosSAP(
        sapFileName.c_str());
   }

   SetCursor(cursor);

}
```

Método auxiliar que retorna o nome do arquivo na janela corrente do aplicativo. Utiliza a classe Tdocument definida nas bibliotecas do Borland.

```
char* TPFrameApp::ArquivoDados()
{
  char* nomeArquivo = NULL;
  TDocument* doc = GetDocManager()->GetCurrentDoc();
  if (doc) {
    nomeArquivo = (char*)doc->GetDocPath();
    string extensao = Extensao(nomeArquivo);
    if (extensao != ".DAD") {
      ::MessageBox(NULL,"Nome do arquivo inválido.",
                   "Erro",MB_ICONSTOP | MB_OK);
      return NULL;
    }
    doc->Commit(true);      // grava o arquivo corrente
  }
  return nomeArquivo;
}  // ArquivoDados
```

O desenvolvimento de outro aplicativo que encapsulasse a resolução de pórticos planos necessitaria conter métodos apenas análogos a estes definidos. Todas as demais classes restantes poderiam ser utilizadas sem modificações.

APÊNDICE 3.
LISTAGENS (ANÁLISE NÃO LINEAR)

3.1 VISÃO GERAL

No Capítulo 4, foi apresentado o modelo de objetos que representa o programa de pórticos planos listado ao longo do Apêndice 2. Mostra-se uma das vantagens da Programação Orientada a Objetos ao estender estas classes para o caso não linear, redefinindo apenas os pontos necessários.

A classe Portico representa o módulo de solução linear. Definem-se três classes distintas para representar os módulos de solução que implementam o Processo P-Delta, o Processo da Matriz de Rigidez Geométrica e o Processo das Funções de Estabilidade. Ao invés de duplicar o código, pode-se criar classes que herdam todo o comportamento do pórtico e apenas acrescentam ou modificam o que for necessário. Também são utilizadas classes derivadas da classe Barra, que representam as novas formulações utilizadas.

3.2 CLASSE PORTICONAOLINEAR

Esta é uma classe base, derivada diretamente da classe Portico, onde define-se o comportamento não linear genérico de um pórtico plano, sem a implementação de nenhum processo em particular. Todos os processos não lineares geométricos serão derivados desta classe.

Além da análise não linear, implementam-se comportamentos específicos não colocados na classe base Portico. São estes a geração do arquivo de "log" e a discretização das barras. Deve-se notar o uso da classe auxiliar ConfigPortico, onde são armazenados os parâmetros para processamento da análise não linear.

```
#ifndef __PORTICO_NAOLINEAR_H
#define __PORTICO_NAOLINEAR_H
#include "portico\configPortico.h"
#include "portico\portico.h"
#include <fstream.h>

class _import StatusCalculo;
//---------------------------------------------------------------------------

class PorticoNaoLinear : public Portico
{
  public:
    PorticoNaoLinear();
    virtual ~PorticoNaoLinear();

    virtual bool Calcula();

    ConfigPortico* Config() { return config; }
    StatusCalculo* Status() { return status; }
    void Status(StatusCalculo* _status) { status = _status; }

    No NoReferencia();
    void NoReferencia(No* noRef) { noReferencia = noRef; }

    void ArquivoLog(const char *nomeArquivo);
    ofstream ArquivoLog() { return logFile; }

    bool LePorticoDeArquivo(const char *nomeArquivo);

    void DivideBarras();

  protected:
    virtual bool CriarMatrizes();
    virtual void DestruirMatrizes();

    virtual void LogarParametros();
    void Logar1aOrdem();
    void AddLog(const char* msg);
    void AddLogSBuffer();
    void TempoProcessamento(time_t tInicial);

    virtual bool Inicializa();
    bool Processa1aOrdem();
    bool Processa2aOrdem();
    virtual bool ProcessaIteracaoGeometrico();
    bool Finaliza();

    bool ProcessoConvergiu();
    void ZerarVetoresIntermediarios();

    virtual void CopiaSecao(Barra* barraOriginal, Barra* barra);
    void DivideBarra(Barra* barraOriginal, VetorFloat trechos);
    virtual Barra* CriaBarra();

  protected:
    StatusCalculo* status;
    float deslocamentoReferencia;
    ConfigPortico* config;
    ofstream logFile;

  private:
    No* noReferencia;
    double *dfOriginal;
    bool primeiraIteracao;
}; // PorticoNaoLinear
#endif __PORTICO_NAOLINEAR_H
```

3.2.1 Métodos para construção e destruição

Construtor da classe PorticoNaoLinear. Inicializa os dados da classe base Portico, mais os dados específicos.

```
PorticoNaoLinear::PorticoNaoLinear() : Portico()
{
  config = new ConfigPortico();
  config->ProcessoNLGeometrica(NLG_NENHUM);
  deslocamentoReferencia = 0;
  primeiraIteracao = true;
  status = NULL;
  noReferencia = NULL;
} // PorticoNaoLinear
```

Destrutor da classe PorticoNaoLinear.

```
PorticoNaoLinear::~PorticoNaoLinear()
{
  if (config)
    delete config;
}
```

Método que inicializa os vetores de dados. Estende o mesmo método da classe Portico (inicializa todos os dados da classe base, mais o vetor adicional criado).

Será chamado virtualmente a partir da classe Portico.

```
bool PorticoNaoLinear::CriarMatrizes()
{
  Portico::CriarMatrizes();
  dfOriginal = new double[Ngl()+1];
  return true;
} // CriarMatrizes
```

Método que desaloca os vetores de dados. Estende o mesmo método da classe Portico (libera todos os dados da classe base, mais o vetor adicional criado).

Será chamado virtualmente a partir da classe Portico.

```
void PorticoNaoLinear::DestruirMatrizes()
{
  logFile.close();
  Portico::DestruirMatrizes();
  delete[] dfOriginal;
} // DestruirMatrizes
```

3.2.2 Método público para processamento

Redefinição do método Calcula() da classe base Portico.

```
bool PorticoNaoLinear::Calcula()
{
  time_t tInicial = time(NULL);
  calculoOk = false;

  if (Inicializa())
  if (ProcessalaOrdem())
  if (Processa2aOrdem())
  if (Finaliza())
    calculoOk = true;

  TempoProcessamento(tInicial);
  DestruirMatrizes();  // fecha log
  return calculoOk;
} // Calcula
```

3.2.3 Métodos privados para processamento

Inicializa os dados do pórtico. Chama os métodos DivideBarras() e DadosEstruturais(). Este último, por ser exatamente igual ao da classe base Portico, não precisa ser redefinido.

```
bool PorticoNaoLinear::Inicializa()
{
  if (status) status->AlterarMensagem("Inicializando");
  DivideBarras();
  LogarParametros();
  if (!NBarras() || !NNos())
    return false;
  if (!DadosEstruturais())
    return false;
  return true;
} // Inicializa
```

Este método basicamente agrupa os métodos utilizados para a análise linear, que não necessitam ser redefinidos. Acrescentam-se as mensagens de progresso.

Os deslocamentos obtidos após a análise de 1ª ordem são armazenados no vetor dfOriginal.

```
bool PorticoNaoLinear::ProcessalaOrdem()
{
  if (status)
    status->AlterarMensagem("Criando matriz de rigidez");
  if (!CriarMatrizSFF())
    return false;
  if (!MatrizDeRigidez())
    return false;
  if (status)
     status->AlterarMensagem("Fatorando matriz");
  if (!matEst->FatorarMatrizBanda(status)) {
    ::MessageBox(NULL,"Erro na solução do
sistema.","Erro",MB_ICONSTOP | MB_OK);
    return false;
  }
  if (status)
    status->AlterarMensagem("Obtendo resultados");
  Carregamentos();
  matEst->ResolverMatrizBanda(ac,df);
  Resultados();

  // guarda dfOriginal
  memcpy(dfOriginal, df, (Ngl()+1)*sizeof(double));

  deslocamentoReferencia =
                        NoReferencia().Deslocamento(DX);

  LogarlaOrdem();
  return true;
} // ProcessalaOrdem
```

Este método efetua a análise de 2ª ordem do pórtico. Independentemente do processo adotado (esta classe não implementa nenhum deles), é necessário efetuar uma iteração e comparar os resultados obtidos com a iteração anterior.

Este método controla as iterações e chama o método responsável pelo processo não linear adotado.

```
bool PorticoNaoLinear::Processa2aOrdem()
{
  if (config->ProcessoNLGeometrica() != NLG_NENHUM), {
    bool convergiu = false;
    int nIter = 1;
    while ((!convergiu) &&
           (nIter < config->IterMaxGeometrica())) {
       ProcessaIteracaoGeometrico();
       convergiu = ProcessoConvergiu();
       nIter++;
    }
    if (!convergiu) {
      AddLog("Processo não convergiu.");
      ::MessageBox(NULL,"Processo não convergiu.",
                   "Erro",MB_ICONSTOP | MB_OK);
      return false;
    }
  }
  return true;
} // Processa2aOrdem
```

Este método representa o processo adotado para a análise não linear. Está implementado em branco, podendo ser redefinido em suas classes derivadas.

```
bool PorticoNaoLinear::ProcessaIteracaoGeometrico()
{
  return true;
} // ProcessaIteracaoGeometrico
```

Efetua os últimos procedimentos após a análise linear. Aqui, é feita a verificação da precisão numérica dos resultados obtidos.

```
bool PorticoNaoLinear::Finaliza()
{
  if (!matEst->VerificarPrecisao(ac,df))
    return false;
  return true;
} // Finaliza
```

Método auxiliar que reinicializa os vetores de dados necessários em cada iteração da análise não linear.

```
void PorticoNaoLinear::ZerarVetoresIntermediarios()
{
  delete[] df;
  delete[] ac;

  df = new double[Ngl()+1];
  ac = new double[Ngl()+1];
} // ZerarVetoresIntermediarios
```

3.2.4 Métodos para controle das iterações

Método que retorna o nó utilizado como referência para definir a diferença entre os deslocamentos ocorridos em cada iteração da análise não linear.

Caso o nó tenha sido definido na leitura do arquivo, retorna o nó indicado; caso contrário, retorna o nó com maior deslocamento em X.

```
No PorticoNaoLinear::NoReferencia()
{
  if (!noReferencia) {  // obter nó referência
    No* noRef = (*nos)[1];
    VetorDeNosIterator itNos(*nos);
    while (itNos) {
      No* corrente = itNos++;
      if (fabs(corrente->Deslocamento(DX)) >
          fabs(noRef->Deslocamento(DX)))
        noRef = corrente;
    }
    if (fabs(noRef->Deslocamento(DX)))
      noReferencia = noRef;
    else
      return *noRef;
  }
  return *noReferencia;
} // NoReferencia
```

Método que calcula a diferença ocorrida entre os deslocamentos calculados na iteração corrente e na iteração anterior.

Retorna verdadeiro quando esta diferença for inferior ao valor limite configurado.

```
bool PorticoNaoLinear::ProcessoConvergiu()
{
  bool retorno = false;
  float novoDeslocRef = NoReferencia().Deslocamento(DX);
  if (status) status->DeslocRef(novoDeslocRef);
  if (deslocamentoReferencia != 0) {
    float dif = fabs( (novoDeslocRef -
        deslocamentoReferencia)/deslocamentoReferencia );
    sBuffer.Init(3,ios::fixed);
    sBuffer << "Deslocamento ref: " << novoDeslocRef
            << "           dif = " << (100*dif) << "%";
    AddLogSBuffer();
    if ( dif <= config->PrecisaoGeometrica() )
      retorno = true;
  }
  deslocamentoReferencia = novoDeslocRef;
  return retorno;
} // ProcessoConvergiu
```

3.2.5 Métodos que efetuam a discretização do pórtico

Método que efetua a discretização das barras do pórtico, com base nos parâmetros configurados.

Para cada barra, define os parâmetros de discretização e chama o método DivideBarra().

Ao final, os nós gerados são renumerados automaticamente.

```
void PorticoNaoLinear::DivideBarras()
{
  float tamExtremos, tamCentral;
  int nb = NBarras();
  for(int b = 1; b <= nb; b++) {
    VetorFloat *trechos = new VetorFloat(10,1,5);
  switch (config->DiscrBarras()) {
    case DIV_NENHUM:
         break;
    case DIV_IGUAIS: {
        int nDivisoes = int(
                    (1/config->TamanhoDivisao())+0.5);
        tamCentral = 1.0/nDivisoes;
        for (int i=1; i<=nDivisoes;i++)
          trechos->Add(tamCentral);
        }
         break;
    case DIV_EXTREMOS:
        tamExtremos = config->TamanhoDivisao();
        tamCentral = 1-4*config->TamanhoDivisao();
        trechos->Add(tamExtremos);
        trechos->Add(tamExtremos);
        trechos->Add(tamCentral);
        trechos->Add(tamExtremos);
        trechos->Add(tamExtremos);
         break;
    case DIV_INICIO:
        tamExtremos = config->TamanhoDivisao();
        tamCentral = 0.5*(1-3*config->TamanhoDivisao());
        trechos->Add(tamExtremos);
        trechos->Add(tamExtremos);
        trechos->Add(tamExtremos);
        trechos->Add(tamCentral);
        trechos->Add(tamCentral);
         break;
    }
    if (!config->DividirSomenteVerticais() ||
        (*barras)[b]->Vertical())
      DivideBarra((*barras)[b], *trechos);
  }
  if ((config->DiscrBarras() != DIV_NENHUM) &&
      (config->TamanhoDivisao() < 1)) {
    RenumeradorPortico renumerador(nos, barras);
    renumerador.RenumeraNos();
  }

} // DivideBarras
```

Método que recebe como parâmetro uma barra do pórtico e um vetor de dados para discretização, efetua as subdivisões necessárias e acrescenta as novas barras ao pórtico.

```
void PorticoNaoLinear::DivideBarra(Barra* barraOriginal,
VetorFloat trechos)
{
  No* noInicial = &barraOriginal->NoInicial();
  No* noFinal = &barraOriginal->NoFinal();
  No* noAnterior = noInicial;
  int nDivisoes = trechos.GetItemsInContainer();

  for (int i=1; i<=(nDivisoes-1); i++) {
    float x = noAnterior->X() +
              (noFinal->X()-noInicial->X()) * trechos[i];
    float y = noAnterior->Y() +
              (noFinal->Y()-noInicial->Y()) * trechos[i];
    int numeroNo = nos->GetItemsInContainer()+1;
    No* novoNo = new No(x, y, numeroNo);
    nos->Add(novoNo);

    Barra* barra = barraOriginal->CriaCopia();
    CopiaSecao(barraOriginal, barra);
    int numeroBarra = barras->GetItemsInContainer()+1;
    barra->Numero(numeroBarra);
    barra->NoInicial(*noAnterior);
    barra->NoFinal(*novoNo);
    barras->Add(barra);
    noAnterior = novoNo;
  }
  barraOriginal->NoInicial(*noAnterior);

} // DivideBarra
```

3.2.6 Métodos para a criação do pórtico a partir do arquivo

Método que lê os dados do pórtico a partir de um arquivo texto. Redefine o mesmo método da classe Portico, acrescentando a este a leitura do nó referência para a análise não linear.

```
bool PorticoNaoLinear::LePorticoDeArquivo(const char
*nomeArquivo)
{
  nos->Flush();
  barras->Flush();
  ArquivoInput arquivo(nomeArquivo);
  if(!arquivo.AlocacaoOK()) {
    ::MessageBox(NULL,"Não foi possível abrir
                 o arquivo.","Erro",MB_ICONSTOP | MB_OK);
    return false;
  }

  tituloModelo = arquivo.ReadLine();

  int nNos = arquivo.ReadBlockInteger();
  int nBarras = arquivo.ReadBlockInteger();

  LeDados(arquivo, nNos, nBarras);

  int noRef = arquivo.ReadBlockInteger(); // nó
referência
  if (noRef != 0)  // 0 -> atribui no cálculo
    NoReferencia(Nos()[noRef]);

  return true;
} // LePorticoDeArquivo
```

Método que cria uma nova barra para o pórtico.

```
Barra* PorticoNaoLinear::CriaBarra()
{
  return new Barra;
} // CriaBarra
```

3.2.7 Métodos para a criação do arquivo de "log"

Método público que inicializa o arquivo de "log", com base no nome do arquivo texto passado como parâmetro.

```
void PorticoNaoLinear::ArquivoLog(const char *nomeArquivo)
{
  logFile.open(nomeArquivo);
  logFile.precision(3);
  logFile.setf(ios::fixed);
} // ArquivoLog
```

Método privado que adiciona uma mensagem passada por parâmetro ao arquivo de "log". A mesma mensagem é passada ao diálogo indicador de progresso, caso este tenha sido criado.

```
void PorticoNaoLinear::AddLog(const char* msg)
{
  logFile << msg << endl;
  if (status)
    status->LogarMensagem(msg);
} // AddLog
```

Método privado que adiciona uma mensagem armazenada na stream global sBuffer ao arquivo de "log". A mesma mensagem é passada ao diálogo indicador de progresso, caso este tenha sido criado.

```
void PorticoNaoLinear::AddLogSBuffer()
{
  sBuffer << ends;
  logFile << buffer << endl;
  if (status)
    status->LogarMensagem(buffer);
} // AddLog
```

Calcula o tempo necessário para o processamento do pórtico e adiciona esta mensagem ao log. O tempo inicial é atribuído no início do método Calcula() e passado por parâmetro.

```
void PorticoNaoLinear::TempoProcessamento(time_t tInicial)
{
  time_t tFinal;
  tFinal = time(NULL);
  int minutos = (tFinal-tInicial)/60;
  int segundos = (tFinal-tInicial)%60;
  sBuffer.Init();
  sBuffer << "Tempo total de processamento: " <<
             minutos << "'"
          << segundos << "''";
  AddLogSBuffer();
  if (status) {
    if (calculoOk)
      status->AlterarMensagem("Cálculo finalizado com
                               sucesso.");
    else
      status->AlterarMensagem("Erro na solução do
                               problema.");
  }
} // TempoProcessamento
```

Adiciona ao arquivo de "log" os parâmetros do modelo.

Estes incluem: título do modelo, processo não linear utilizado, precisão mínima e informações sobre a discretização adotada.

```
void PorticoNaoLinear::LogarParametros()
{
  AddLog("**********************************");
  AddLog(" PÓRTICO PLANO ");
  AddLog(tituloModelo.c_str());
  AddLog("**********************************");

  AddLog("     ** Não linearidade geométrica: ");
  sBuffer.Init();
  switch (config->ProcessoNLGeometrica()) {
    case NLG_NENHUM:
      sBuffer << "     Nenhuma";
      break;
    case NLG_PDELTA:
      sBuffer << "     Processo P-Delta";
      break;
    case NLG_KG:
      sBuffer << "     Processo da Matriz de
                 Rigidez Geométrica";
      break;
    case NLG_F_ESTABILIDADE:
      sBuffer << "     Processo das Funções
                 de Estabilidade";
      break;
  }
  AddLogSBuffer();

  if (config->ProcessoNLGeometrica() != NLG_NENHUM) {
    sBuffer.Init(1,ios::fixed);
    sBuffer << "     Precisão mínima: " <<
               (100*config->PrecisaoGeometrica()) << "%";
    AddLogSBuffer();
  }

  sBuffer.Init(1,ios::fixed);
  sBuffer << "     Discretização das barras: ";
  switch (config->DiscrBarras()) {
    case DIV_NENHUM:   sBuffer << "Nenhuma";
                       break;
    case DIV_IGUAIS:   sBuffer << "Trechos iguais de " <<
                      (100*config->TamanhoDivisao()) << "%";
                       break;
    case DIV_EXTREMOS: sBuffer << "Extremos de " <<
                      (100*config->TamanhoDivisao()) << "%";
                       break;
    case DIV_INICIO:   sBuffer << "Duas de " <<
         (100*config->TamanhoDivisao()) << "% no início";
                       break;
  }
  AddLogSBuffer();

  sBuffer.Init();
  sBuffer << "     Barras a discretizar: ";
  if (config->DividirSomenteVerticais())
     sBuffer << "Apenas verticais";
  else
     sBuffer << "Todas";
  AddLogSBuffer();

  AddLog("**********************************");

} // LogarParametros
```

Adiciona ao arquivo de "log" os resultados obtidos a partir da análise de 1ª ordem..

Estes incluem: deslocamento de 1ª ordem utilizado como referência, valor do parâmetro Alfa, valor do coeficiente Gama-Z e o deslocamento que seria obtido ao simplesmente majorar o deslocamento de 1ª ordem por Gama-Z.

```
void PorticoNaoLinear::Logar1aOrdem()
{
  AddLog("*** Análise de 1ª ordem: ************");

  sBuffer.Init(3,ios::fixed);
  sBuffer << "Deslocamento inicial: " <<
             deslocamentoReferencia;
  AddLogSBuffer();

  float alfa = ParametroAlfa();
  sBuffer.Init(3,ios::fixed);
  sBuffer << "Parâmetro Alfa: ";
  if (alfa == MAXFLOAT)
    sBuffer << "N/A";
  else
    sBuffer << alfa;
  AddLogSBuffer();

  float gamaZ = CoeficienteGamaZ();
  sBuffer.Init(3,ios::fixed);
  sBuffer << "Coeficiente Gama-Z: ";
  if (gamaZ == MAXFLOAT)
    sBuffer << "N/A";
  else
    sBuffer << gamaZ;
  AddLogSBuffer();

  sBuffer.Init(3,ios::fixed);
  sBuffer << "Deslocamento previsto: ";
  if (gamaZ == MAXFLOAT)
    sBuffer << "N/A";
  else
    sBuffer << (deslocamentoReferencia*gamaZ);
  AddLogSBuffer();

  AddLog("****************************************");
} // Logar1aOrdem
```

3.3 CLASSE PORTICOCARGAS

Os processos P-Delta e da Matriz de Rigidez Geométrica Modificado, embora diferentes em formulação, adotam a mesma abordagem para solução do problema não linear. Desta forma, tornou-se interessante utilizar uma classe intermediária onde definem-se os métodos comuns aos dois processos.

Não são listados, deste ponto em diante, os métodos construtores e destrutores, por não conterem dados adicionais.

```
#ifndef __PORTICO_CARGAS_H
#define __PORTICO_CARGAS_H
#include "portico\porticoNaoLinear.h"

class PorticoNLCargas : public PorticoNaoLinear
{
   public:
    PorticoNLCargas();
    virtual ~PorticoNLCargas();
   protected:
    virtual bool ProcessaIteracaoGeometrico();

}; // PorticoNLCargas
#endif __PORTICO_CARGAS_H
```

3.3.1 Método que efetua a análise não linear

Método que efetua as operações necessárias à solução do problema não linear. É necessário, em cada iteração, reinicializar o vetor de carregamentos (onde já serão acrescentadas as cargas fictícias) e obter os novos deslocamentos.

```
bool PorticoCargas::ProcessaIteracaoGeometrico()
{
   if (status)
    status->AlterarMensagem("Obtendo vetor de cargas");
   ZerarVetoresIntermediarios();
   Carregamentos();
   matEst->ResolverMatrizBanda(ac,df);
   if (status)
    status->AlterarMensagem("Obtendo esforços");
   Resultados();
   return true;
} // ProcessaIteracaoGeometrico
```

3.4 CLASSE PORTICOPDELTA

Este classe deriva-se a partir da classe PorticoCargas definida anteriormente, implementando apenas o método que define o tipo de barra a ser utilizado.

```
#ifndef __PORTICO_PDELTA_H
#define __PORTICO_PDELTA_H

#include "portico\PorticoCargas.h"

class PorticoPDelta : public PorticoCargas
{
  public:
    PorticoPDelta();
    virtual ~PorticoPDelta();
    virtual Barra* CriaBarra();
}; // PorticoPDelta

#endif __PORTICO_PDELTA_H
```

3.4.1 Método que define o tipo de barra

Método virtual que cria uma nova barra para o pórtico. Retorna uma BarraPDelta, que é a classe que redefine a Barra para o Processo P-Delta.

```
Barra* PorticoPDelta::CriaBarra()
{
  return new BarraPDelta;
} // CriaBarra
```

3.5 CLASSE PORTICOKGMOD

Este classe deriva-se a partir da classe PorticoCargas definida anteriomente, implementando apenas o método que define o tipo de barra a ser utilizado.

```
#ifndef __PORTICO_KG_MOD_H
#define __PORTICO_KG_MOD_H

#include "portico\PorticoNLCargas.h"

class PorticoKGMod : public PorticoNLCargas
{
  public:
    PorticoKGMod();
    virtual ~PorticoKGMod();

    virtual Barra* CriaBarra();

}; // PorticoKGMod

#endif __PORTICO_KG_MOD_H
```

3.5.1 Método que define o tipo de barra

Método virtual que cria uma nova barra para o pórtico. Retorna uma BarraKgMod, que é a classe que redefine a Barra para o processo K_{GM}.

```
Barra* PorticoKgMod::CriaBarra()
{
  return new BarraKgMod;
} // CriaBarra
```

3.6 CLASSE PORTICORIGIDEZ

Os dois outros processos para inclusão da não linearidade geométrica, o Processo da Matriz de Rigidez Geométrica e o Processo das Funções de Estabilidade, embora diferentes em formulação, também são muito semelhantes em termos de análise. Desta forma, tornou-se interessante utilizar uma classe intermediária onde definem-se os métodos comuns aos dois processos.

```
#ifndef __PORTICO_RIGIDEZ_H
#define __PORTICO_RIGIDEZ_H

#include "portico\porticoNaoLinear.h"

class PorticoRigidez : public PorticoNaoLinear
{
  public:
    PorticoRigidez();
    virtual ~PorticoRigidez();

  protected:

    virtual bool ProcessaIteracaoGeometrico();

}; // PorticoRigidez

#endif __PORTICO_RIGIDEZ_H
```

3.6.1 Método que efetua a análise não linear

Método que efetua as operações necessárias, iguais nos dois processos. É necessário, em cada iteração, montar a nova matriz de rigidez (com base na nova formulação), reinicializar o vetor de carregamentos e obter os novos deslocamentos.

```
bool PorticoRigidez::ProcessaIteracaoGeometrico()
{
   if (status)
    status->AlterarMensagem("Criando matriz de rigidez");
   ZerarVetoresIntermediarios();
   CriarMatrizSFF();
   MatrizDeRigidez();
   if (status)
    status->AlterarMensagem("Fatorando matriz");
   if (!matEst->FatorarMatrizBanda(status)) {
     ::MessageBox(NULL,"Erro na solução do sistema.",
                  "Erro",MB_ICONSTOP | MB_OK);
     return false;
   }
   if (status)
    status->AlterarMensagem("Obtendo resultados");
   Carregamentos();
   matEst->ResolverMatrizBanda(ac,df);
   Resultados();
   return true;
} // ProcessaIteracaoGeometrico
```

3.7 CLASSE PORTICOKG

Este classe deriva-se a partir da classe PorticoRigidez definida anteriomente, implementando apenas o método que define o tipo de barra a ser utilizado.

```
#ifndef __PORTICO_KG_H
#define __PORTICO_KG_H

#include "portico\porticoRigidez.h"

//---------------------------------------------------------------------------
class PorticoKg : public PorticoRigidez
{
  public:
    PorticoKg();
    virtual ~PorticoKg();

    virtual Barra* CriaBarra();

  protected:

}; // PorticoKg

#endif __PORTICO_KG_H
```

3.7.1 Método que define o tipo de barra

Método virtual que cria uma nova barra para o pórtico. Retorna uma BarraKg, que é a classe que redefine a Barra para o Processo da Matriz de Rigidez Geométrica.

```
Barra* PorticoKg::CriaBarra()
{
  return new BarraKg;
} // CriaBarra
```

3.8 CLASSE PORTICOESTABILIDADE

Este classe deriva-se a partir da classe PorticoRigidez definida anteriomente, implementando apenas o método que define o tipo de barra a ser utilizado.

```
#ifndef __PORTICO_FESTABILIDADE_H
#define __PORTICO_FESTABILIDADE_H

#include "portico\porticoRigidez.h"

//---------------------------------------------------------------------------
class PorticoFEstabilidade : public PorticoRigidez
{
  public:
    PorticoFEstabilidade();
    virtual ~PorticoFEstabilidade();

    virtual Barra* CriaBarra();

  protected:

}; // PorticoFEstabilidade
#endif __PORTICO_FESTABILIDADE_H
```

3.8.1 Método que define o tipo de barra

Método virtual que cria uma nova barra para o pórtico. Retorna uma BarraKg, que é a classe que redefine a Barra para o Processo das Funções de Estabilidade.

```
Barra* PorticoFEstabilidade::CriaBarra()
{
  return new BarraFEstabilidade;
} // CriaBarra
```

3.9 CLASSE BARRAPDELTA

Esta classe redefine o comportamento da classe Barra, alterando apenas os métodos necessários para a implementação do Processo P-Delta. Sendo este baseado na inclusão de esforços cortantes fictícios à barra, é necessário apenas redefinir o método que calcula os esforços de imobilização da barra.

```
class BarraPDelta : public Barra
{
  public:
    BarraPDelta();
    BarraPDelta(No& _noInicial,No& _noFinal,int _numero = 0);
    virtual Barra* CriaCopia();

  protected:
    virtual float AcoesEngPerf(TEsforco indice);

  private:

}; // BarraPDelta
```

3.9.1 Método que define os esforços de imobilização

Método que obtém os esforços de imobilização da barra.

Inicialmente, definem-se os esforços baseados na formulação original da barra.

Caso a barra seja vertical, acrescenta os esforços cortantes calculados com base na carga axial e nos deslocamentos horizontais ocorridos.

```
float BarraPDelta::AcoesEngPerf(TEsforco indice)
{
  float valor = Barra::AcoesEngPerf(indice);
  if (Vertical()) {
     float N = Esforco(ESF_AXIAL_I);
     if (fabs(N) > 0.1) {
        float a = NoFinal().Deslocamento(DX) -
                  NoInicial().Deslocamento(DX);
        float h = N * a / Extensao();
        switch (indice) {
          case ESF_CORTANTE_I   : valor += h;
                                  break;
          case ESF_CORTANTE_F   : valor -= h;
                                  break;
        }
     }
  }
  return valor;
} // AcoesEngPerf
```

3.10 CLASSE BARRAKGMOD

Esta classe redefine o comportamento da classe Barra, alterando apenas os métodos necessários para a implementação do Processo da Matriz de Rigidez Geométrica Modificado. É necessário apenas redefinir o método que calcula os esforços de imobilização da barra, acrescentando a este os esforços fictícios obtidos através da multiplicação do vetor de deslocamentos obtido na iteração anterior pela matriz de rigidez geométrica..

```
class BarraKgMod : public Barra
{
  public:
    BarraKgMod();
    BarraKgMod(No& _noInicial,No& _noFinal,int _numero = 0);
    virtual Barra* CriaCopia();

  protected:
    virtual float AcoesEngPerf(TEsforco indice);
    void BarraKgMod::MatrizCorrecaoRigidez (MatrizBarra* matLocal)

  private:

}; // BarraPDelta
```

3.10.1 Método que define os esforços de imobilização

Método que obtém os esforços de imobilização da barra.

Inicialmente, definem-se os esforços baseados na formulação original da barra, acrescentando-se a estes os esforços fictícios.

O vetor Deslocamento() contém os deslocamentos nodais sofridos pela barra, referidos ao seu sistema de coordenadas locais.

```
float BarraKgMod::AcoesEngPerf(TEsforco indice)
{
  MatrizBarra *mataux = new MatrizBarra();
  MatrizCorrecaoRigidez(mataux);
  LadoSimetrico(mataux);

  float valor = Barra::AcoesEngPerf(indice);
  for (TDeslocamentoBarra i=DX_I;i<=RZ_F;i++)
   valor += (*mataux)[indice][i]
            * Deslocamento(i);
  return valor;
} // AcoesEngPerf
```

Método auxiliar que contém a formulação da Matriz de Rigidez Geométrica utilizada para obtenção do vetor de carregamentos fictícios.

```
void BarraKgMod::MatrizCorrecaoRigidez
(MatrizBarra* matLocal)
{
  float l = Extensao();
  float N = Esforco(ESF_AXIAL_I);

  (*matLocal)[2][2] += 6 * N / (5 * l);
  (*matLocal)[2][3] += N / 10;
  (*matLocal)[2][5] += -6 * N / (5 * l);
  (*matLocal)[2][6] += N / 10;
  (*matLocal)[3][3] += 2 * N * l / 15;
  (*matLocal)[3][5] += -N / 10;
  (*matLocal)[3][6] += -N * l / 30;
  (*matLocal)[5][5] += 6 * N / (5 * l);
  (*matLocal)[5][6] += -N / 10;
  (*matLocal)[6][6] += 2 * N * l / 15;
} // MatrizCorrecaoRigidez
```

3.11 CLASSE BARRAKG

Esta classe redefine o comportamento da classe Barra, alterando apenas os métodos necessários para a implementação do Processo da Matriz de Rigidez Geométrica. Este processo baseia-se apenas na modificação da matriz de rigidez da barra.

```
#ifndef __BARRAKG_H
#define __BARRAKG_H

#include "portico\barra.h"

//---------------------------------------------------------------------------
class BarraKg : public Barra
{
  public:
    BarraKg();
    BarraKg(No& _noInicial,No& _noFinal,int _numero = 0);
    virtual Barra* CriaCopia();

  protected:
    virtual void MatrizRigidezLocal(MatrizBarra* sm);
  private:

}; // BarraKg

//---------------------------------------------------------------------------
#endif __BARRAKG_H
```

3.11.1 Método que obtém a matriz de rigidez local

Método que define a matriz de rigidez da estrutura referida ao seu sistema local.

Inicialmente, são preenchidos os termos da matriz de rigidez elástica, conforme a implementação feita na classe Barra, sendo a estes acrescentados os termos da matriz de rigidez geométrica.

```
void BarraKg::MatrizRigidezLocal(MatrizBarra* matLocal)
{
  Barra::MatrizRigidezLocal(matLocal);
  float l = Extensao();
  float N = -Esforco(ESF_AXIAL_I);

  (*matLocal)[2][2] += 6 * N / (5 * l);
  (*matLocal)[2][3] += N / 10;
  (*matLocal)[2][5] += -6 * N / (5 * l);
  (*matLocal)[2][6] += N / 10;
  (*matLocal)[3][3] += 2 * N * l / 15;
  (*matLocal)[3][5] += -N / 10;
  (*matLocal)[3][6] += -N * l / 30;
  (*matLocal)[5][5] += 6 * N / (5 * l);
  (*matLocal)[5][6] += -N / 10;
  (*matLocal)[6][6] += 2 * N * l / 15;
} // MatrizRigidezLocal
```

3.12 CLASSE BARRAESTABILIDADE

Esta classe redefine o comportamento da classe Barra, alterando apenas os métodos necessários para a implementação do Processo das Funções de Estabilidade Geométrica. Este processo baseia-se apenas na modificação da matriz de rigidez da barra.

```
#ifndef __BARRA_F_ESTABILIDADE_H
#define __BARRA_F_ESTABILIDADE_H
#include "portico\barra.h"

//---------------------------------------------------------------------------
class BarraFEstabilidade : public Barra
{
  public:
    BarraFEstabilidade();
    BarraFEstabilidade(No& _noInicial,No& _noFinal,int _numero = 0);
    virtual Barra* CriaCopia();

  protected:
    virtual float AcoesEngPerf(TEsforco indice);
    virtual void MatrizRigidezLocal(MatrizBarra* sm);
  private:
}; // BarraFEstabilidade

//---------------------------------------------------------------------------
#endif __BARRA_F_ESTABILIDADE_H
```

3.12.1 Método que obtém a matriz de rigidez local

Método que define a matriz de rigidez da estrutura referida ao seu sistema local.

Inicialmente, são preenchidos os termos da matriz de rigidez elástica, conforme a implementação feita na classe Barra, sendo estes multiplicados pelas funções de rotação e translação definidas.

Para evitar problemas de instabilidade numérica (que ocorrem quando a carga axial tende a zero), foi estabelecida uma carga mínima abaixo da qual a matriz de rigidez elástica não é modificada.

```
void BarraFEstabilidade::MatrizRigidezLocal(MatrizBarra*
matLocal)
{
  Barra::MatrizRigidezLocal(matLocal);
  float l = Extensao();
  float N = Esforco(ESF_AXIAL_I);

  double k = sqrt(fabs(N)/(ModuloE()*Inercia()));
  double kL = k*l;
  double fiC = 2 - 2*cos(kL) - kL*sin(kL);
  double fiT = 2 - 2 * cosh(kL) + kL * sinh(kL);
  double s1=1, s2=1, s3=1, s4=1;

  if (N > 10) {                    //compressao
    s1 = (pow(kL,3) * sin(kL)) / (12 * fiC);
    s2 = (pow(kL,2) * (1 - cos(kL))) / (6 * fiC);
    s3 = (kL * (sin(kL) - kL * cos(kL))) / (4 * fiC);
    s4 = (kL * (kL - sin(kL))) / (2 * fiC);
  }
  else if (N < -10) {
    s1 = (pow(kL,3) * sinh(kL)) / (12 * fiT);
    s2 = (pow(kL,2) * (cosh(kL) - 1)) / (6 * fiT);
    s3 = (kL * (-sinh(kL) + kL * cosh(kL))) / (4 * fiT);
    s4 = (kL * (-kL + sinh(kL))) / (2 * fiT);
  }

  (*matLocal)[2][2] *= s1;
  (*matLocal)[2][3] *= s2;
  (*matLocal)[2][5] *= s1;
  (*matLocal)[2][6] *= s2;
  (*matLocal)[3][3] *= s3;
  (*matLocal)[3][5] *= s2;
  (*matLocal)[3][6] *= s4;
  (*matLocal)[5][5] *= s1;
  (*matLocal)[5][6] *= s2;
  (*matLocal)[6][6] *= s3;
} // MatrizRigidezLocal
```

3.13 CLASSE APLICATIVO

Uma vez que a definição do aplicativo externo não faz parte do objetivo do trabalho, apresenta-se aqui apenas os métodos básicos que gerenciam a análise do pórtico.

Optou-se por definir uma única classe Aplicativo que permite calcular o pórtico contido em um arquivo de dados tanto pela análise linear como através de um dos processos descritos para análise não linear.

Utilizam-se os métodos já definidos para a análise linear: ArquivoDados() e Processa().

Calcula o modelo utilizando o Processo P-Delta.

```
void TPFrameApp::CmCalculaPorticoPDelta()
{
  CalcularPortico(NLG_PDELTA);
} // CmCalculaPorticoPDelta
```

Calcula o modelo utilizando o Processo da Matriz de Rigidez Geométrica

```
void TPFrameApp::CmCalculaPorticoKg()
{
  CalcularPortico(NLG_KG);
} // CmCalculaPorticoKg
```

Calcula o modelo utilizando o Processo das Funções de Estabilidade.

```
void TPFrameApp::CmCalculaPorticoFEstabilidade()
{
  CalcularPortico(NLG_F_ESTABILIDADE);
} // CmCalculaPorticoKg
```

Método que executa um diálogo onde podem ser configuradas as opções da análise não linear. Estas opções são gravadas e passadas para o pórtico.

```
void TPFrameApp::CmCalculaPorticoDialogoNLG()
{
  const char* nomeArquivo = ArquivoDados();
  if (nomeArquivo) {
    PFrameConfigNLG configDlg(
                    GetMainWindow(), nomeArquivo);
    bool ok = (configDlg.Execute() == IDOK);
    // Lê as opções configuradas
    ConfigPortico config;
    ok = ok && config.Importa(nomeArquivo);
    if (ok)
      CalcularPortico(config.ProcessoNLGeometrica());
  }
} // CmCalculaPorticoDialogoNLG
```

Método privado que cria a classe correta do pórtico a ser analisado, com base no processo escolhido.

O pórtico criado é passado para o método ProcessaStatus().

```
void          TPFrameApp::CalcularPortico(tipoNLGeometrica
processoNLGeometrica)
{
  const char* nomeArquivo = ArquivoDados();
  if (nomeArquivo) {
    PorticoNaoLinear* port;
    switch (processoNLGeometrica) {
      case NLG_NENHUM: port = new PorticoNaoLinear();
                       break;
      case NLG_PDELTA: port = new PorticoPDelta();
                       break;
      case NLG_KG:     port = new PorticoKg();
                       break;
      case NLG_F_ESTABILIDADE:
                       port = new PorticoFEstabilidade();
                       break;
    }
    ProcessaStatus(port, nomeArquivo);
    delete port;
  }
}
```

Método privado que processa o pórtico não linear passado como parâmetro.

Efetua as operações específicas para os pórticos não lineares: inicializa os dados de configuração. inicializa o arquivo de "log", lê o arquivo de dados e cria a janela de progresso do cálculo.

O pórtico recém inicializado é passado para o método Processa(), onde é calculado da mesma forma que um pórtico linear.

```
void  TPFrameApp::ProcessaStatus(PorticoNaoLinear*  port,
const char* nomeArquivo)
{
  // Inicializa configuração
  port->Config()->Importa(nomeArquivo);
  // Inicializa arquivo LOG
  string logFileName = MudaExtensao(nomeArquivo, ".log");
  port->ArquivoLog(logFileName.c_str());
  // Lê dados do pórtico
  port->LePorticoDeArquivo(nomeArquivo);
  // Cria janela de progresso
  StatusCalculo* status =
                 new StatusCalculo(GetMainWindow());
  port->Status(status);
  status->Create();
  // Calcula o pórtico
  Processa(port,nomeArquivo);
  // Finaliza
  status->Finalizar();

} // ProcessaStatus
```