Anais do VI Simpósio EPUSP sobre Estruturas de Concreto
Abril / 2006 ISBN 85-86686-36-0
Modelagem Computacional de Estruturas de Concreto
Trabalho SIMP0098 - p. 102-121

AMBIENTE GRÁFICO INTERATIVO PARA DIAGRAMAS DE INTERAÇÃO DE SEÇÕES DE CONCRETO ARMADO

Interactive environment for obtaining interactive diagrams of reinforced concrete sections

Filipe Braga Silva(1); Flavio Torres da Fonseca(2); Leandro Augusto Campelo de Barros(3); Marcelo Paixão Pinto Rodrigues(4); Roque Luiz Pitangueira(5);

(1), (2), (3) e (4) Alunos de Graduação em Engenharia Civil, Universidade Federal de Minas Gerais
Email (1): filipesilva@globo.com
Email (2): flaviotf@uai.com.br
Emai (3) : leandroacbarros@globo.com
Email (4) : mpaixao@waymail.com.br

(5) Professor, Departamento de Engenharia de Estruturas
Universidade Federal de Minas Gerais
email: roque@dees.ufmg.br

Correspondência: Avenida do Contorno, 842 -- andar 2. Belo Horizonte, MG 30110-060

Resumo

Este trabalho apresenta a implementação de um ambiente gráfico interativo para obtenção de diagramas de interação de seções de concreto armado. As seções podem ter qualquer forma geométrica poligonal e qualquer disposição de barras de aço, podendo estar solicitada por um conjunto de esforço normal e momentos fletores, caracterizando desde uma flexão simples até uma flexão oblíqua composta. Em seções de concreto armado submetidas à flexão obliqua é necessária a definição da profundidade e da inclinação da linha neutra. Para cada posição da linha neutra que satisfaça as condições de equilíbrio são calculados os esforços resistentes, os quais são comparados com os esforços fornecidos, até que eles se equivalham dentro de certa tolerância. A implementação do ambiente é feita segundo a metodologia de programação orientada a objetos utilizando Java. A obtenção dos diagramas de interação baseia-se em uma discretização da seção em uma lista de pontos neste plano.
Palavras-Chave: *concreto armado; diagrama de interação; programação orientada a objetos.*

Abstract

This work aims at the implementation of a graphical and interactive environment for obtaining interactive diagrams of reinforced concrete sections. The section may have any polygonal geometry and any steel bars arrangement. The section can be solicited from normal force and moments, which is enough to represent from a simple flexion to an oblique flexion. To find the position of the neutral line is necessary to obtain both the translation and the rotation of the neutral line through an iterative process. For each position of the neutral line which satisfies the equilibrium, the solicitant stress state is calculated and compared to the given one until they become equal to each other within a certain tolerance. The implementation of the graphic environment follows the Objected Oriented Programming methodology using Java. The obtained diagrams are based on the discretization of the section into a list of points on its plane.
Keywords*: reinforced concrete; interactive diagrams; Objected Oriented Programming*

1 Introdução

Este artigo apresenta parte do trabalho desenvolvido na disciplina “Trabalho Integralizador Multidisciplinar III”, do curso de Engenharia Civil da Universidade Federal de Minas Gerais, no segundo semestre do ano 2005. Trata-se do desenvolvimento de um software livre para geração de diagramas de interação de seções genéricas de concreto armado, submetidas a esforços de flexão oblíqua.

A comunidade acadêmica e os projetistas de estruturas de concreto armado estão em fase de adequação à Nova Norma Brasileira para Projeto e Execução de Obras de Concreto Armado (NBR-6118/2003). Segundo esta nova norma, a ocorrência da flexão oblíqua composta deve ser considerada em situações não tratadas adequadamente em sua antiga versão. Peças submetidas a esse tipo de solicitação são tradicionalmente dimensionadas através de ábacos construídos para seções retangulares, com distribuição e posicionamento pré-definidos de armadura. Assim, o software proposto traz grande generalização em relação a esses ábacos, uma vez que não se limita a seções retangulares, disposições ou posicionamento de armadura. Por se tratar de um software livre, o mesmo pode servir à comunidade acadêmica no tocante às disciplinas, tanto da graduação quanto da pós-graduação, que se propõem a ensinar o dimensionamento de estruturas de concreto armado. Essa validade pode-se estender também ao meio profissional, disseminando-se entre os vários escritórios de projetos que ainda se prendem aos tradicionais ábacos.

Em seções de concreto armado submetidas à flexão obliqua, a linha neutra tem sua posição desconhecida tanto pela profundidade (deslocamento em relação a um referencial), como pela rotação. Dessa forma necessita-se de dois parâmetros geométricos para determinar a posição da linha neutra: a profundidade e a inclinação. TEPEDINO (1986) desenvolveu um trabalho que oferece uma sistemática para o dimensionamento ou verificação da resistência, em estado limite último, bem como a verificação das tensões e deformações, em estado limite de utilização. Para estender esse estudo a qualquer forma de seção e qualquer disposição de armaduras, Tepedino (1986) divide a seção em elementos retangulares. Pela falta de ferramental computacional apropriado, em seus estudos, os elementos nem sempre possuem dimensões reduzidas, diferentemente do trabalho aqui proposto, onde o tamanho do elemento da malha escolhida pelo usuário poderá ser bastante reduzido uma vez que os cálculos serão feitos computacionalmente.

O posicionamento da linha neutra é obtido através de um processo iterativo que, para cada inclinação da linha, varia seu posicionamento em relação a um sistema de coordenadas, localizado no centro de gravidade da seção transversal e alinhado com a referida inclinação. Cada par de valores obtidos para a inclinação e profundidade representa uma posição da linha neutra que satisfaz as condições de equilíbrio e corresponde a um par de momentos de flexão, em relação ao sistema global de coordenadas da seção transversal.

Além da liberdade em relação ao refinamento da subdivisão da seção em retângulos, o software proposto disponibiliza recursos gráficos interativos modernos de maneira a torná-lo amigável ao usuário.

A seguir, apresenta-se uma breve caracterização da Flexão Oblíqua Composta, demonstram-se as etapas de cálculo que compõe a elaboração do software e as simplificações adotadas, discute-se a implementação do sistema, segundo o paradigma da Programação Orientada a Objetos e comparam-se resultados obtidos pelo programa, para exemplos retirados da literatura de maneira a validar o software desenvolvido.

2 Flexão Composta Oblíqua de Seções de Concreto Armado Discretizada por Pontos

Para o dimensionamento de qualquer peça em concreto armado, deve-se verificar se os esforços solicitantes provenientes das ações externas se equilibram com os esforços resistentes internos. Para se calcular os esforços resistentes, é necessário saber qual é a tensão atuante em cada ponto da seção transversal. Calcula-se esta tensão a partir de uma relação entre tensão e deformação, estabelecidas pelas normas técnicas. Portanto, a grandeza incógnita do problema é a deformação, que é facilmente obtida a partir da posição da linha neutra.

Todavia, ao contrário do estudo de seções submetidas à flexão simples ou normal composta, em que a linha neutra tem direção pré-determinada (no caso, perpendicular ao plano de atuação do momento fletor), a flexão oblíqua vai criar um problema extra para a definição da deformação atuante em cada ponto da seção transversal.

Quando há atuação de dois momentos fletores perpendiculares entre si, a direção da linha também será desconhecida. Com isso, para peças estruturais submetidas a estes tipos de esforços, a localização da linha neutra compreende a obtenção de duas incógnitas: posição da L.N. e sua inclinação em relação aos eixos principais da seção.

2.1 Hipóteses Básicas

i) A seção de concreto permanece plana após a deformação;

ii) As armaduras estão sujeitas a variação de deformações iguais às deformações específicas do concreto adjacente suposto não fissurado;

iii) A resistência à tração do concreto é desprezada;

iv) A ruptura é definida pelas seguintes deformações específicas:

Ruína do aço: $\varepsilon = 1{,}0\%$

Ruína do concreto: Peça parcialmente comprimida: $\varepsilon = 0{,}35\%$

Peça completamente comprimida: $\varepsilon = 0{,}2\%$ (ocorrendo a 3/7 da altura total da seção)

v) São previstos para o aço e para o concreto os diagramas tensão x deformação recomendados pela NBR-6118/03 e reproduzidos na figura 1;

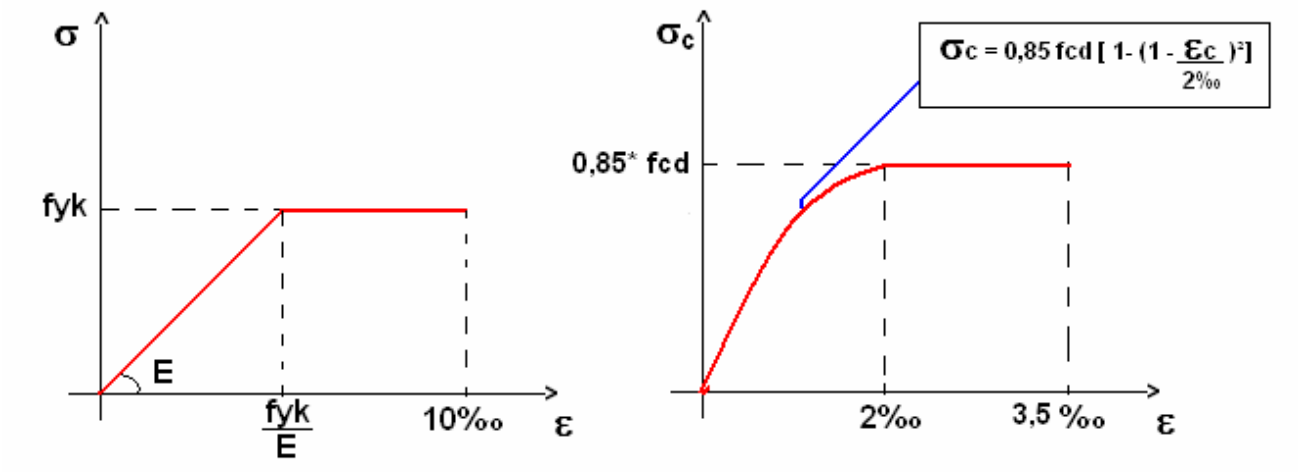


Figura 1: Diagrama tensão-deformação do aço e do concreto

2.2 Discretização da Seção

Visando à simplificação do trabalho de programação computacional, foi adotada uma discretização que dividiu a seção transversal em pequenos quadrados, cuja dimensão pode ser estipulada pelo usuário. Observando-se as figuras 2 e 3, que fazem parte do programa desenvolvido, é possível fazer um paralelo entre o que significa trabalhar com uma seção contínua e o que é a seção discretizada.

É claro que, quanto menor for a dimensão do elemento discretizado, menores serão os erros obtidos ao se utilizar o programa.

O programa irá identificar cada quadrado discretizado como sendo um elemento diferente, seja ele concreto, perfil metálico ou espaço vazio, de acordo com a posição do seu centróide. Já as barras de aço, serão identificadas como um ponto determinado pelo seu centróide.

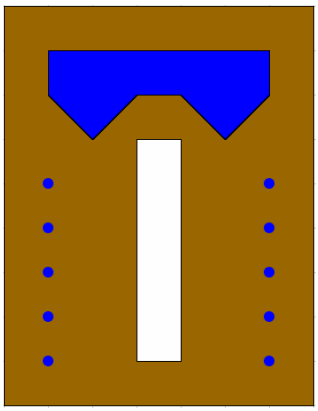


Figura 2: Seção contínua

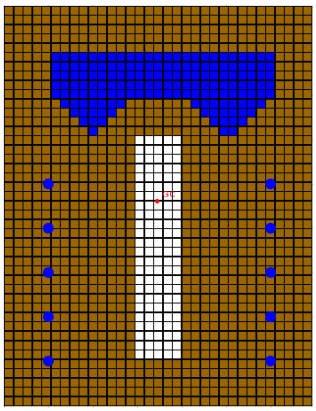


Figura 3: Seção discretizada

2.3 Etapas de Cálculo

a) Cálculo do centróide

Como a força normal e os momentos fletores estão referenciados ao centróide e aos eixos centroidais, respectivamente, o primeiro passo é determinar as coordenadas do centróide da figura. Para a seção discretizada, as equações são:

$$Ycg = \frac{\sum A_{el} \times y_{el}}{\sum A_{el}} \quad (1)$$

$$Xcg = \frac{\sum A_{el} \times x_{el}}{\sum A_{el}} \quad (2)$$

Sendo: A_{el}: Área de cada elemento discretizado;
Y_{el}: Coordenada “y” do centróide do elemento discretizado;
X_{el}: Coordenada “x” do centróide do elemento discretizado;

b) Localização da Linha Neutra

Como já foi discutido anteriormente, no caso de Flexão Composta Oblíqua a localização da Linha Neutra se resume à determinação de duas incógnitas: o ângulo (θ) e sua translação (X_0) em relação aos eixos ortogonais determinados no item anterior.

Como o método de cálculo realizado é iterativo, isto é, são testadas todas as possibilidades possíveis e se verifica qual está de acordo com o enunciado do problema, deve-se fixar uma das incógnitas, variando a outra até encontrar o par (θ, X_0) que verifica a hipótese do problema.

O parâmetro fixado é "θ" e o variável é X_0. Portanto, para cada valor do ângulo encontraremos um valor de X_0, que iguala (dentro de certa tolerância) a força normal suportada pela seção àquela determinada pelo usuário. Com a linha neutra localizada, a partir das deformações determinadas por ela, é possível encontrar o terno de esforços solicitantes suportados pela seção, composto pela força normal e os momentos fletores em relação aos eixos "x" e "y" (N, Mx, My).

Em termos da lógica do programa deve-se, para cada valor de θ fixado, seguir os passos abaixo:

i. Obter o valor de X_{min}, pois o valor inicial do parâmetro variável, X_0, será igual a X_{min}, variando–o até encontrar um valor de normal suportada pela seção que esteja dentro dos limites determinados pelo usuário (N_d e tolerância)

ii. Para cada elemento discretizado da seção deve-se calcular qual é a distância perpendicular à linha neutra, segundo a equação abaixo:

$$d_i = (X - X_o) \times \cos\theta + sen\theta \qquad (3)$$

iii. Obter os valores de di máximo e mínimo. Com isso, tem-se:

h = dimax – dimin

dn = dimax

Os parâmetros definidos anteriormente podem ser melhores visualizados na figura 4, a seguir:

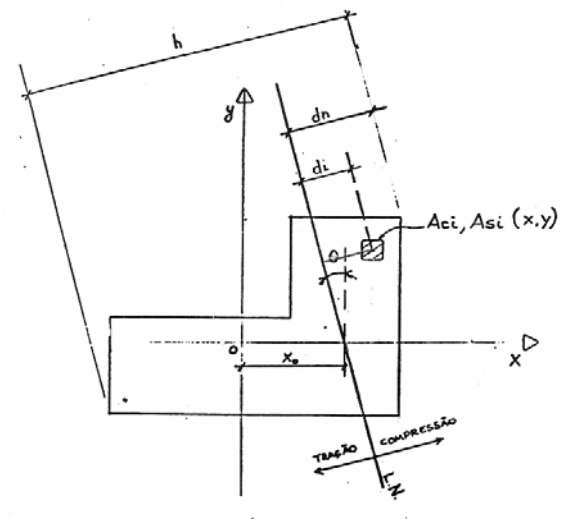


Figura 4: Visualização dos parâmetros do modelo discreto (TEPEDINO, 1986).

c) Cálculo da Deformação em cada elemento discretizado

Nota-se que o valor de dn, calculado no passo anterior, serve para localizar a linha neutra na seção. Comparando-o com o parâmetro h, podemos determinar como a seção está sendo solicitada no que diz respeito às tensões de tração ou de compressão.

dn = h : Seção completamente tracionada;
dn < h : Seção está sofrendo compressão e tração simultaneamente;
dn ≥ h : Seção completamente comprimida;

É claro que, segundo o tipo de solicitação à qual a seção está sujeita, temos duas opções de deformações máximas que podem aparecer sem que a peça estrutural entre em colapso. Isso ocorre pelo fato de o aço (que resiste a esforços de tração e de compressão) e o concreto (que resiste somente à compressão) possuírem deformações máximas diferentes para a sua ruína, como foi abordado nas hipóteses básicas. Com isso, a deformação máxima, εcd, será calculada conforme as equações 4 e 5 abaixo:

Para $d_n < h$:
$$\varepsilon_{cd} = \frac{0{,}01}{\frac{d_i}{d_n} - 1} \tag{4}$$

Para $d_n \geq h$:
$$\varepsilon_{cd} = \frac{0{,}002}{1 - \frac{3}{7} \times \frac{h}{d_n}} \tag{5}$$

Levando em consideração que as deformações possuem variação linear, pode-se dizer que:

$$\varepsilon_i = \varepsilon_{cd} \times \frac{d_i}{d_n} \tag{6}$$

Calculadas as deformações, segundo a equação 6, para encontrar a tensão atuante em cada elemento discretizado, basta utilizar os gráficos de tensão x deformações, apresentados na figura 1.

No caso de o elemento em questão ser classificado como material concreto, temos:

Se $\varepsilon_u \leq 0 \Rightarrow \sigma_{cd} = 0$ (7)

Se $0 < \varepsilon_{cd} < 0{,}002 \Rightarrow \sigma_{cd} = f_c \left[1 - \left(1 - \frac{\varepsilon_i}{0{,}002} \right)^2 \right]$ (8)

Se $\varepsilon_i \geq 0{,}002 \Rightarrow \sigma_{cd} = f_c$ (9)

OBS.: $f_c = \frac{0{,}85 \times f_{ck}}{\gamma_c}$, sendo γc o fator de minoração do concreto.

No caso dos perfis metálicos ou das barras de aço, nas quais os elementos são classificados como aços, têm-se:

Se $\varepsilon_s > \frac{f_{yk}}{E} \Rightarrow \sigma_s = f_{yk}$ (10)

Se $\varepsilon_s \leq \frac{f_{yk}}{E} \Rightarrow \sigma_s = E \times \varepsilon_s$ (11)

Sendo: E: Módulo de Elasticidade transversal do aço
fyk : Tensão característica no aço

d) Cálculo dos Esforços Solicitantes Resistidos pela Seção (N, Mx, My)

O esforço solicitante que coordena a aceitação ou não da posição X_0 da linha neutra é o esforço normal. Caso o esforço normal esteja dentro da tolerância estabelecida pelo usuário em relação à força normal, Nd, que também é um parâmetro de entrada do programa, a posição X_0 será aceita e calcula-se Mx e My resultantes dessa posição da linha neutra para compor o gráfico interativo. Os passos seguintes esclarecem o processo:

1) Cálculo de Nd: Segundo a equação 12:

$$N_d = \sum \left(A_{ci} \times \sigma_{cd} + A_{si} \times \sigma_{sd} \right) + \sum_{barras} \left(- A_s \times \sigma_{cd} + A_s \times \sigma_{sd} \right) \quad (12)$$

2) Teste de Aceitação de X_0:

a) Se $\left| \frac{N_d - N_{d0}}{N_{do}} \right| > \delta$,

Repete-se todo o procedimento, desde o início (a partir do item b), dando um incremento de ΔX, determinado pelo usuário, com isso Xn=Xn-1+ΔX;

b) Se $\left| \frac{N_d - N_{d0}}{N_{do}} \right| < \delta$, passa-se à etapa 3

3) Cálculo dos Momentos Fletores resistidos:

$$M_{yd} = \sum \left[\left(A_{ci} \times \sigma_{cd} + A_{si} \times \sigma_{sd} \right) \times x_i \right] + \sum_{barras} \left[\left(- A_s \times \sigma_{cd} + A_s \times \sigma_{sd} \right) \times x_i \right] \quad (13)$$

$$M_{xd} = \sum \left[\left(A_{ci} \times \sigma_{cd} + A_{si} \times \sigma_{sd} \right) \times y_i \right] + \sum_{barras} \left[\left(- A_s \times \sigma_{cd} + A_s \times \sigma_{sd} \right) \times y_i \right] \quad (14)$$

Com os valores de Mx e My em mão, plota-se o par ordenado no gráfico iterativo (ver figura 6) e repete-se todo o procedimento para cada valor pré-determinado de θ_0. Portanto, cada valor de ângulo fornecerá um ponto do gráfico, mostrado na figura a seguir.

Com o intuito de facilitar a compreensão das etapas de cálculo e a implementação do programa, apresentada a seguir, montou-se um diagrama de atividades, mostrado na figura 5.

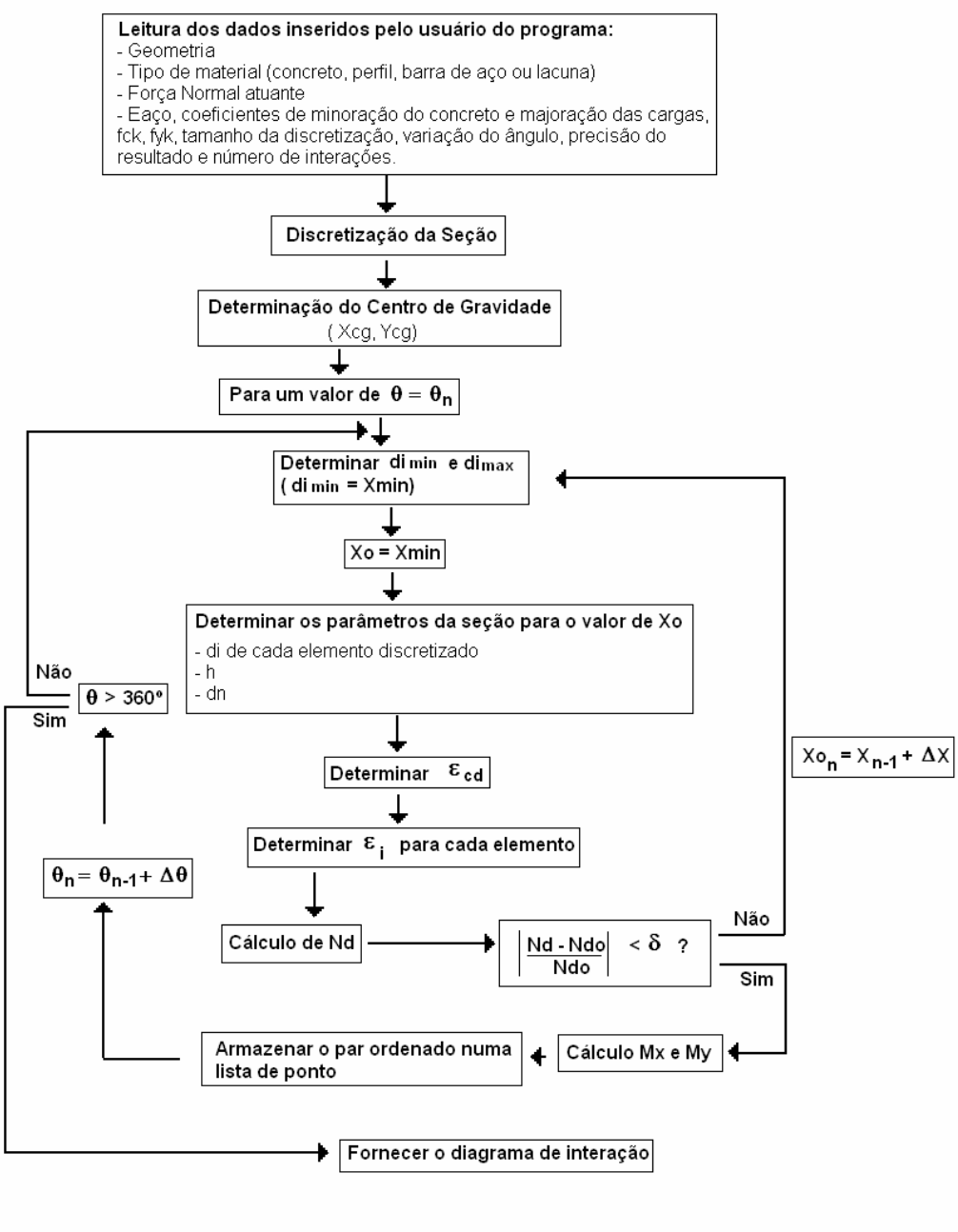


Figura 5: Diagrama de atividades do software

3 Implementação Orientada a Objetos do Sistema

3.1 Programação Orientada a Objetos e Linguagem Java

A programação orientada a objetos (POO) é uma técnica de programação baseada em classes e objetos. Os dados e métodos são encapsulados nos objetos, sendo eles intimamente amarrados entre si. Os objetos apresentam a propriedade de ocultar informações. Ou seja, apesar de eles se comunicarem uns com os outros através de interfaces bem definidas, geralmente um objeto não tem permissão para conhecer como outros objetos se comportam. Isto permite que os programas possam ser divididos em módulos independentes, facilitando o trabalho em conjunto de diversas pessoas em diferentes locais e épocas. Desta forma, a manutenção e expansão do código são muito

mais fáceis de serem feitas em um programa desenvolvido com POO do que em um feito com linguagem estruturada.

A linguagem de programação Java é uma linguagem orientada a objetos, apresentando todas as vantagens deste tipo de programação. Além disso, ela foi desenvolvida de forma a ser independente de plataforma. Essa característica é denominada portabilidade e é um dos principais atrativos desta linguagem. Um programa desenvolvido em Java pode ser compilado em um sistema operacional e executado em outro, sem prejuízos.

Devido a todas as vantagens citadas acima, escolheu-se utilizar a linguagem Java, que suporta o paradigma da programação orientada a objetos, para a implementação do sistema.

3.2 Arquitetura em Camadas e Padrões de Projeto de Software

A figura 6 mostra a combinação da arquitetura em camadas e padrões de projeto de software adotados para o programa desenvolvido, que recebeu o nome de GERADIA. Como pode ser visto na figura, a versão atual do sistema possui quatro camadas lógicas e duas camadas físicas. Três das camadas lógicas do sistema foram definidas utilizando-se o padrão de projeto de software (GAMMA ET AL., 1995) denominado `Modelo-Vista-Controlador` (MVC). Este padrão é bastante apropriado uma vez que preconiza a separação do processamento da informação de sua representação gráfica (PIETRO, 2001), facilitando assim os trabalhos de expansão e manutenção da aplicação. A quarta camada lógica é a camada de persistência.

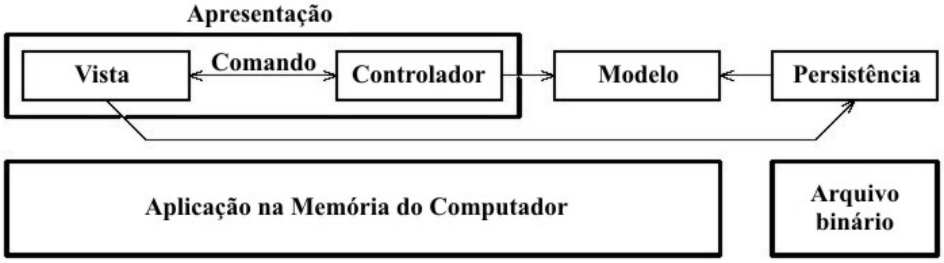


Figura 6 - Arquitetura em Camadas e Padrões de Projeto adotados no GERADIA

Em termos físicos, o GERADIA possui atualmente somente duas camadas: uma aplicação carregada na memória do computador (compreendendo as camadas lógicas `Modelo, Vista e Controlador`) e arquivos binários persistidos em disco.

O inter-relacionamento entre as camadas é conseguido, principalmente, através da implementação do padrão de projeto de software denominado Comando (GAMMA ET AL., 1995). A figura 7 exemplifica este relacionamento para o caso da tarefa de adição de uma entidade geométrica ao modelo corrente e sua visualização. Como pode ser visto na figura, o fluxo de informações para realização de tal tarefa ocorre em quatro etapas. Na primeira etapa, o objeto `Comando`, responsável pela tarefa, aciona o `Controlador` ativo informando a requisição. A seguir (2), o `Controlador` cria o objeto correspondente à entidade geométrica e o adiciona ao `Modelo` pertinente. Na etapa 3, o `Controlador` cria objetos de desenho representativos dos objetos do `Modelo`. Finalmente, na etapa 4, os objetos de desenho pertencentes ao `Controlador` são apresentados na área de desenho da `Vista`.

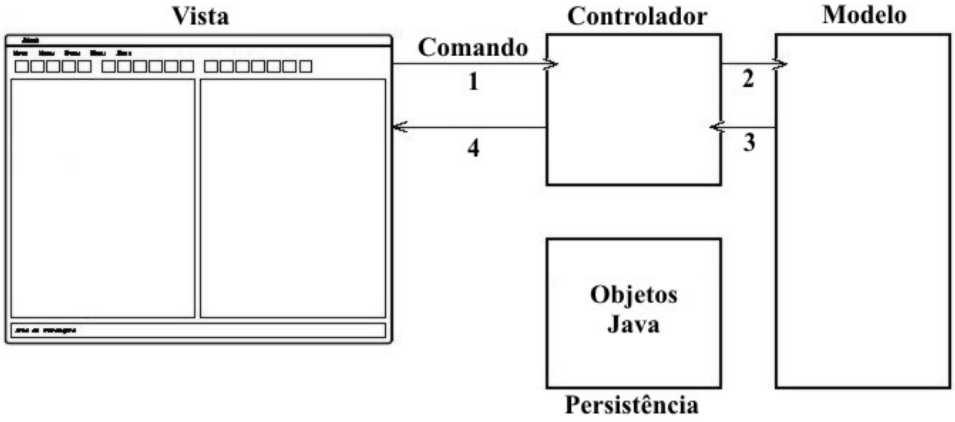


Figura 7 - Relacionamento entre camadas do GERADIA, para adição de uma entidade geométrica.

3.3 Implementação do Padrão MVC

Para implementação do Padrão MVC no sistema, foram criadas as classes Vista, Modelo e Controlador, conforme os diagramas UML (*Unified Modeling Language* (BOOCH ET AL., 2000)) da figura 8.

A classe Vista (figura 8(a)) possui um objeto da classe `Modelo`, um da classe `Controlador`, um da classe `Área de Desenho`, um da classe `Configuração`, vários objetos da classe `Comando` e vários objetos representativos de elementos que compõem a interface gráfica com o usuário (GUI- *Graphical User Interface*).

A classe `Modelo` (figura 8(b)) é composta por instâncias de classes que implementam a lógica da resolução do problema através da formulação apresentada no capítulo 3 e por um objeto da classe `Configuração`, que armazena informações comuns a todas as classes de modelagem. Entre as classes de modelagem, a classe `Modelo Geométrico` cuida da estrutura de dados representativa da geometria do modelo. A classe `Modelo Discreto` representa a discretização da seção em pequenos elementos e em barras circulares de aço.

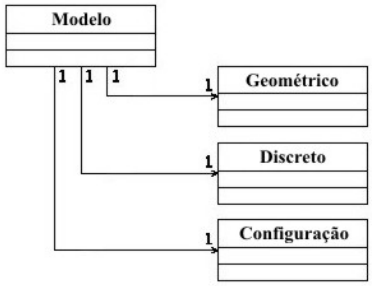
A classe `Controlador` (figura 8(c)), como recomenda GRAND (1998), é uma classe totalmente abstrata (em Java, uma interface) que faz referência ao `Modelo` corrente, à `Vista` e à `Área de Desenho`.

Cada uma das classes que implementam a interface Controlador (ver figura 8(d)) possui listas (instâncias de classes da API Collections (HORSTMANN & CORNELL, 2000)) contendo instâncias de `Objeto de Desenho`, que são extensões de classes da API gráfica Java2D (ROWE, 2001).

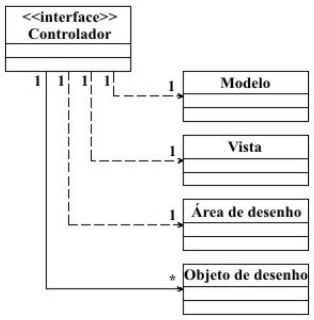
Os objetos `Área de Desenho` e `Elemento` de GUI (ver figuras 8(a) e 8(c)) são instâncias de classes da `API Java` Swing (HORSTMANN & CORNELL, 2000). O objeto `Área de Desenho` é um componente da Swing, onde um objeto `Graphics2D`, composto das instâncias de Objeto de Desenho do Controlador (ver figura 8(c)), é desenhado.

Como dito anteriormente, o relacionamento entre as classes criadas para implementação do MVC é conseguido com o uso do padrão Comando. A figura 9, que detalha a figura 7, exemplifica este relacionamento.

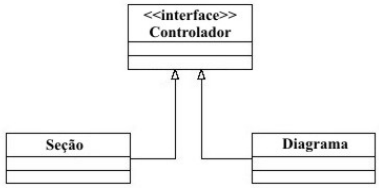
(a) Instâncias da Vista



(b) Instâncias do Modelo



(c) Instâncias do Controlador



(d) Herança do Controlador

Figura 8 - Diagramas do projeto orientado a objetos do GERADIA

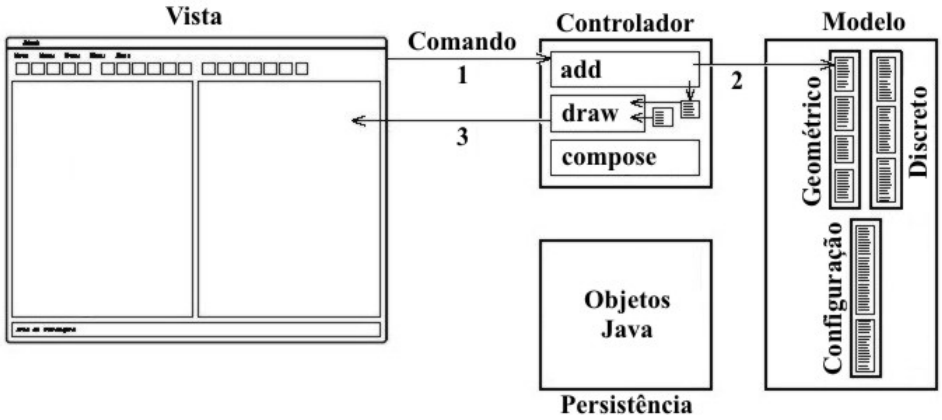


Figura 9 - Relação entre camadas para operação de adição de objeto no GERADIA

Tendo o Usuário requisitado a introdução de uma entidade geométrica ao modelo, o `Controlador` correspondente (ver figura 8(d)) é acionado (etapa 1 das figuras 7 e 10).

Com informações relativas à entidade a ser adicionada, o `Controlador` solicita a adição da mesma no `Modelo` (etapa 2 das figuras 7 e 10).

Ainda possuidor do controle da operação, o objeto `Controlador` adiciona uma instância de `Objeto de Desenho`, correspondente à entidade geométrica, em sua lista (ver método add na figura 11 equivalente à etapa 3 da figura 7) e repassa o controle para o `Comando`.

Tal `Comando` solicita à Vista que atualize sua `Área de Desenho`. Para tanto, o `Controlador` corrente é acionado para que construa o objeto `Graphics2D` correspondente, a ser desenhado (etapa 3 da figura 9 e etapa 4 da figura 7).

A figura 10 ilustra o diagrama UML de seqüência para a requisição de adição de um ponto no `Modelo Geométrico`.

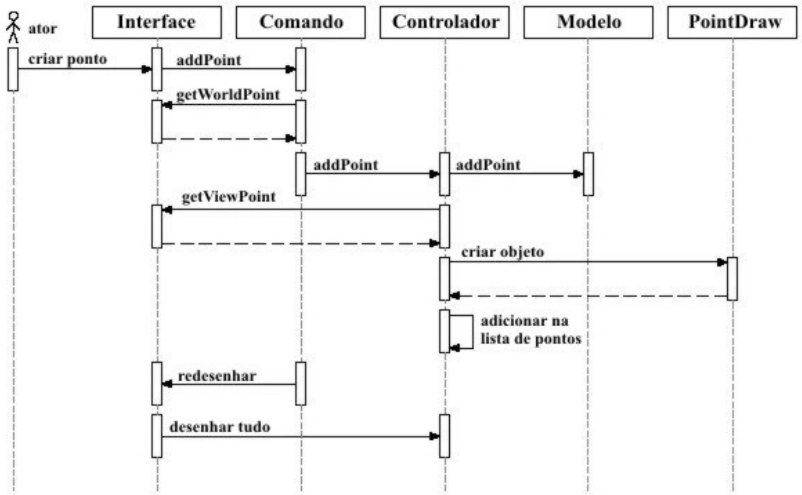
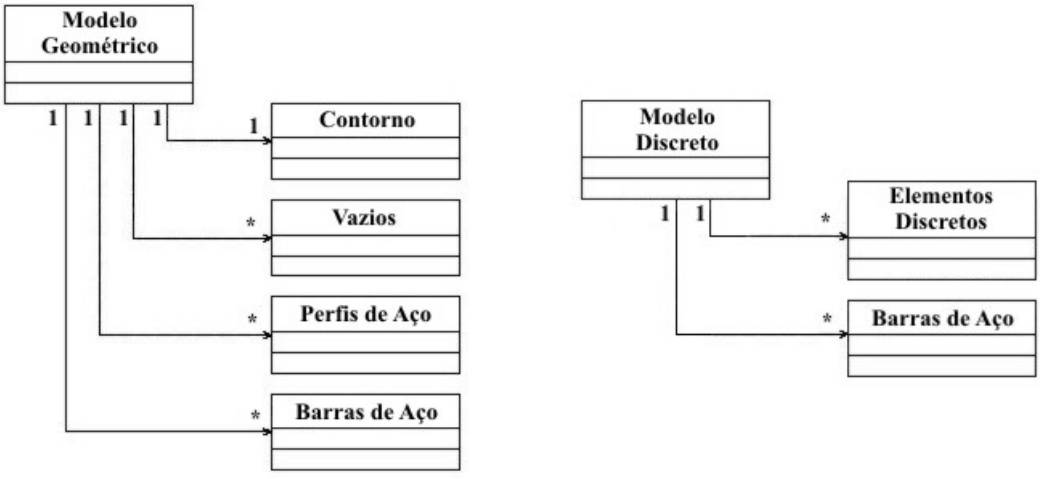


Figura 10 - Diagrama de seqüência para operação de adição de objeto no GERADIA

3.4 Detalhamento da Camada Modelo

Conforme mostrado na figura 8(b), a classe `Modelo` é constituída por uma instância da classe `Modelo Geométrico`, uma de `Modelo Discreto` e uma de `Estado do Modelo`, que armazena suas configurações. A classe `Modelo Geométrico` (Figura 11(a)) possui um objeto `GeneralPath` do pacote `Swing`, que representa o contorno da seção. Ela possui também duas listas de objetos `GeneralPath` que representam os vazios da seção e os perfis de aço. Há ainda uma lista de objetos `SteelBar`, que representam as barras de aço.



(a) Classe Modelo Geométrico (b) Classe Modelo Discreto

Figura 11 – Detalhamento das classes constituintes de Modelo.

A classe Modelo Discreto possui apenas duas listas. A primeira contém objetos da classe `DiscreteElement`, que representam os elementos discretos. A segunda contém objetos da classe `DiscreteSteelBars`, que representam as barras de aço.

3.5 Detalhamento da Camada Vista

Deve-se fazer uma diferenciação entre a camada `Vista`, preconizada pelo padrão MVC, e a classe `Vista` implementada. A camada Vista representa tudo aquilo que é desenhado na tela, ou seja, a interface gráfica. Já a classe Vista é uma pequena parte desta camada, e pode ser simplificada como uma `Área de Desenho` e os objetos necessários para que esta desempenhe sua função, que é representar graficamente dados armazenados na memória do computador. A figura 12 apresenta a interface gráfica do GERADIA e alguns de seus componentes. Ela contém duas `Áreas de Desenho`, uma para a seção e outra para o diagrama, alguns menus e botões de comando, um *prompt* de comando e um painel de mensagens.

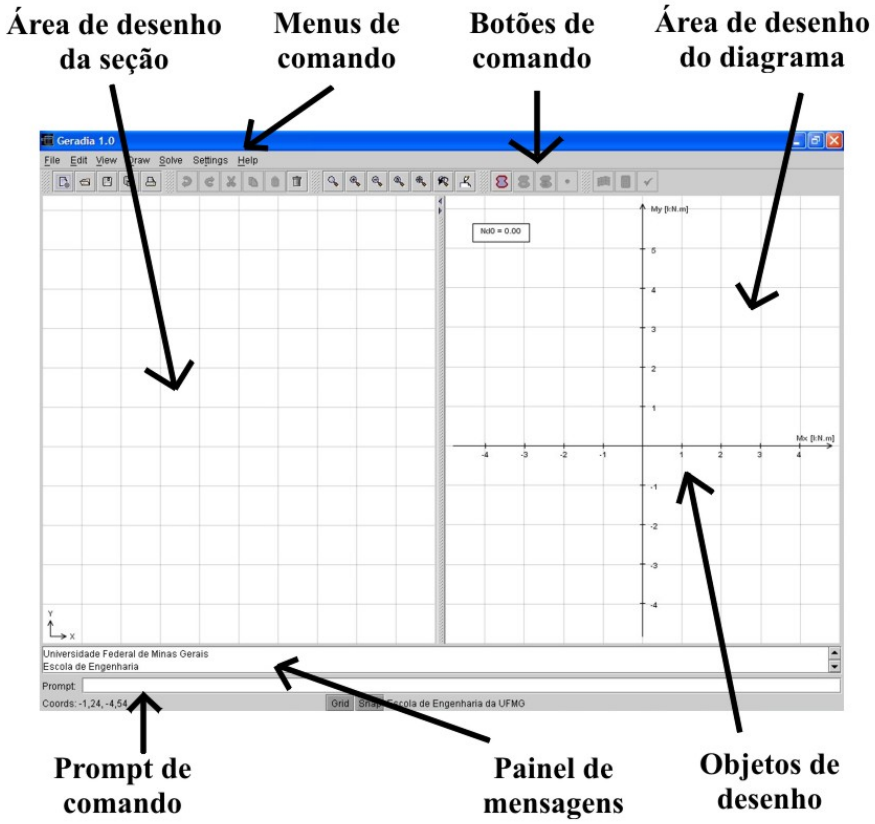
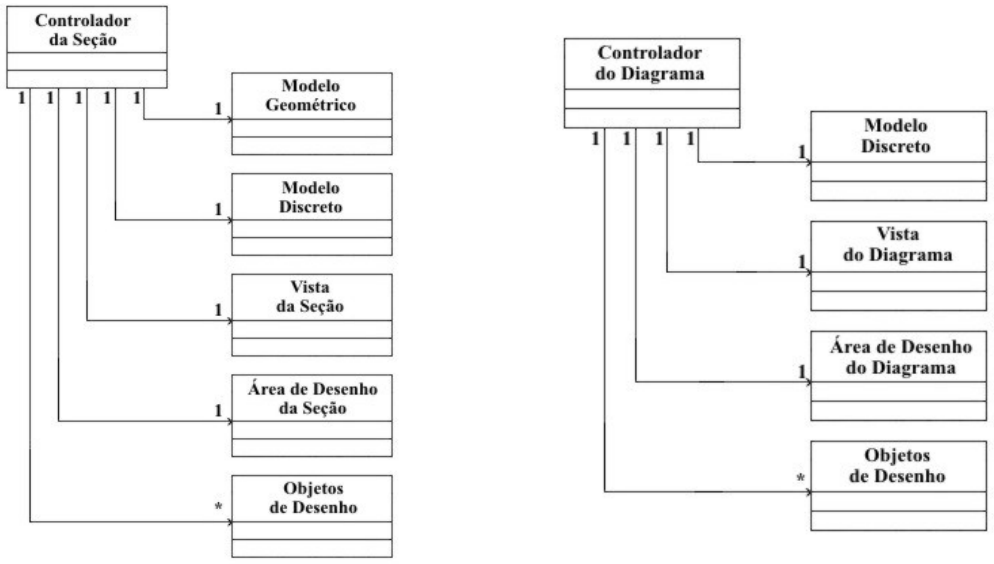


Figura 12– Componentes da interface gráfica do GERADIA.

Através dos menus e botões, pode-se acessar os diversos comandos disponibilizados no programa. O *prompt* de comando tem esta mesma função e, além disso, pode ser utilizado para a entrada de alguns dados (coordenadas das barras de aço, por exemplo) com maior precisão do que se utilizando o mouse. O painel de mensagens é uma área onde o usuário recebe informações a respeito do andamento do processo, pedidos de entrada de dados e avisos de erros. Pode-se ver na figura 12 que a `Área de Desenho` contém alguns `Objetos de Desenho`, no caso, representantes dos eixos cartesianos e do valor da força normal aplicada à seção.

3.6 Detalhamento da Camada Controlador

Existem dois tipos de Controladores no GERADIA: o `Controlador da Seção` e o `Controlador do Diagrama`. Como os próprios nomes dizem, o primeiro é responsável pela entrada de dados referentes à seção e pela sua representação gráfica na tela. O segundo trata de representar os resultados obtidos pelo processamento do programa na forma de um diagrama de interação. A figura 13 apresenta o detalhamento destas classes.



(a) `Controlador da Seção` (b) `Controlador do Diagrama`

Figura 13 – Detalhamento das subclasses de Controlador.

O `Controlador da Seção` possui referências tanto ao `Modelo Geométrico` quanto ao `Modelo Discreto`, para que seja possível representar a seção em sua forma contínua e em sua forma discretizada, de acordo com a preferência do usuário. Ele também possui uma Vista, uma `Área de Desenho` e vários `Objetos de Desenho`.

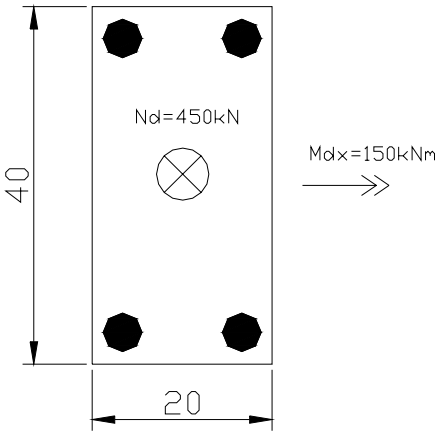
O `Controlador do Diagrama` possui referência apenas ao `Modelo Discreto`. Assim como o `Controlador da Seção`, ele também possui uma `Vista`, uma `Área de Desenho` e vários `Objetos de Desenho`.

Estes dois controladores encontram-se ativos durante toda a execução do programa. Isto possibilita a visualização conjunta da seção e do diagrama.

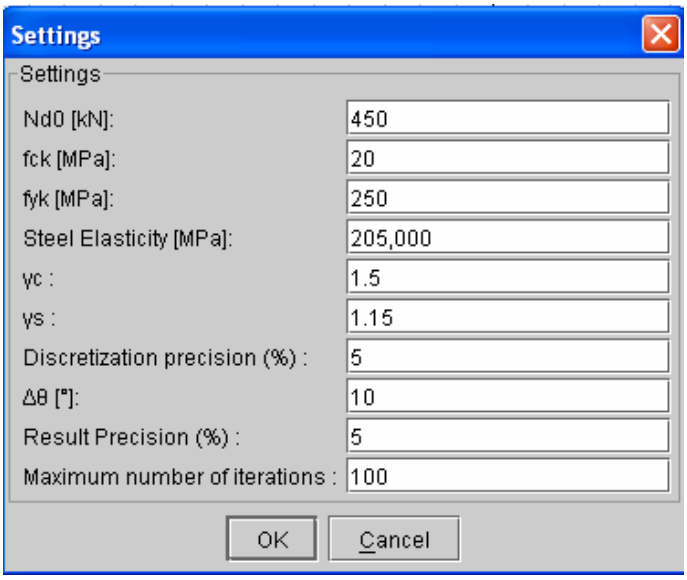
4 Exemplos

A seguir apresentam-se cinco exemplos de verificação do programa. Os quatro primeiros comparam os resultados obtidos pelo programa com os obtidos em PFEIL (1976), VENTURINE e BORTOLIN (1992) e MONTOYA, MESEGUER E CABRÉ (1979). O último exemplo mostra uma seção não simétrica que foi calculada manualmente para efeito de verificação de todos os passos do programa.

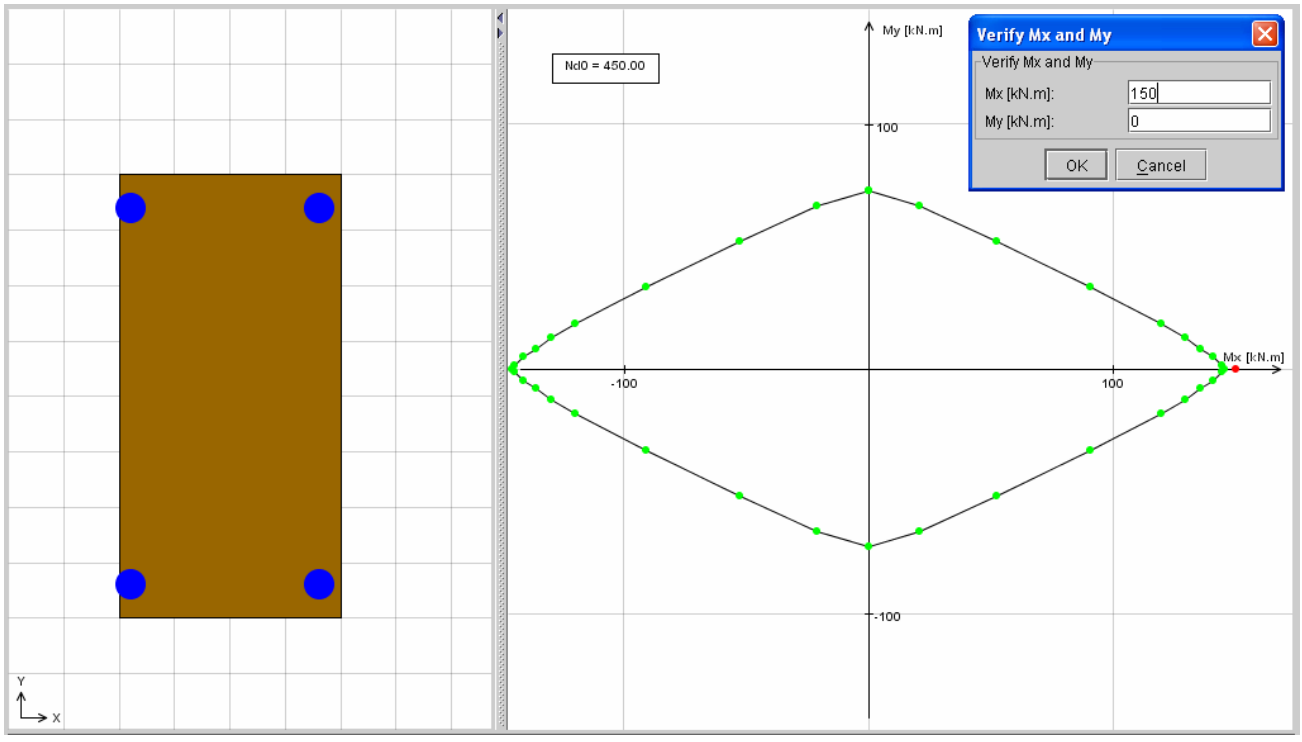
4.1 Exemplo 1(PFEIL, 1976)



(a) Seção Transversal (4 ϕ 30 mm)



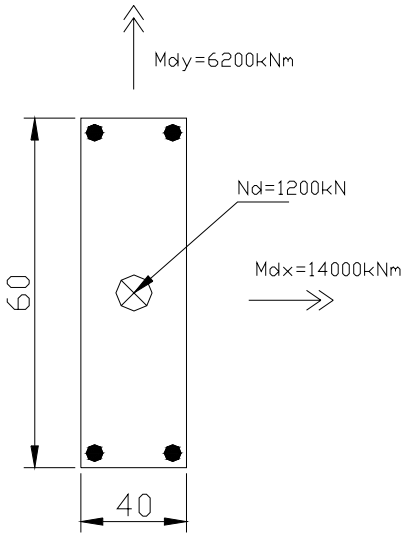
(b) Dados do Problema



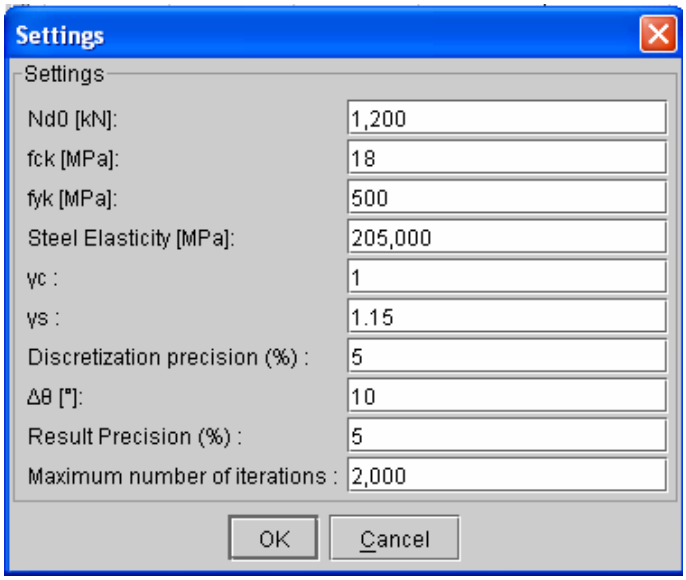
(c) – Diagrama de Interação obtido e Verificação

Figura 14 – Exemplo 1: Flexão Composta Reta (PFEIL, 1976).

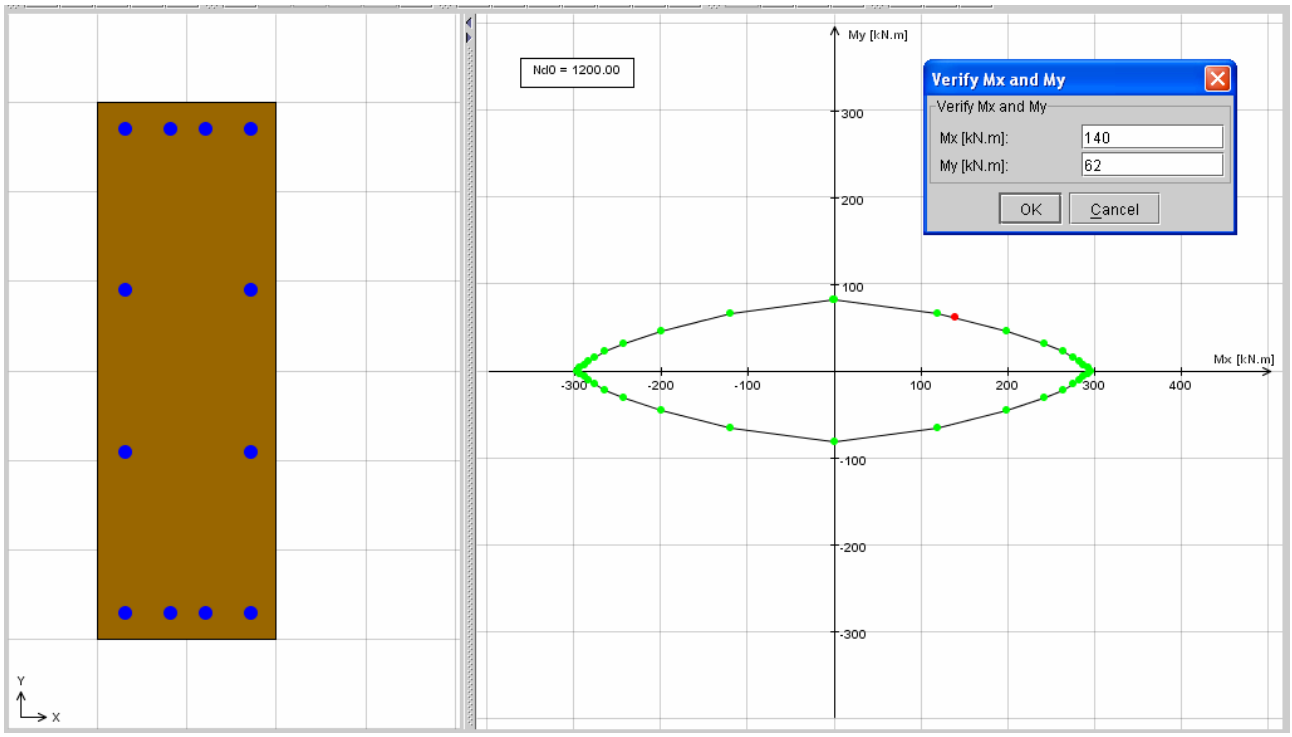
4.2 Exemplo 2 (VENTURINI E BORTOLIN, 1992)



(a) Seção Transversal (12 ϕ 16 mm)



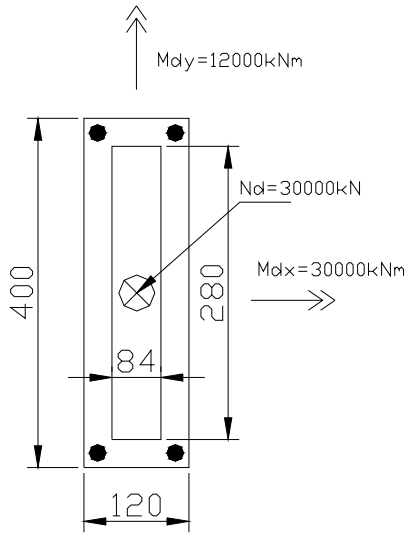
(b) Dados do Problema



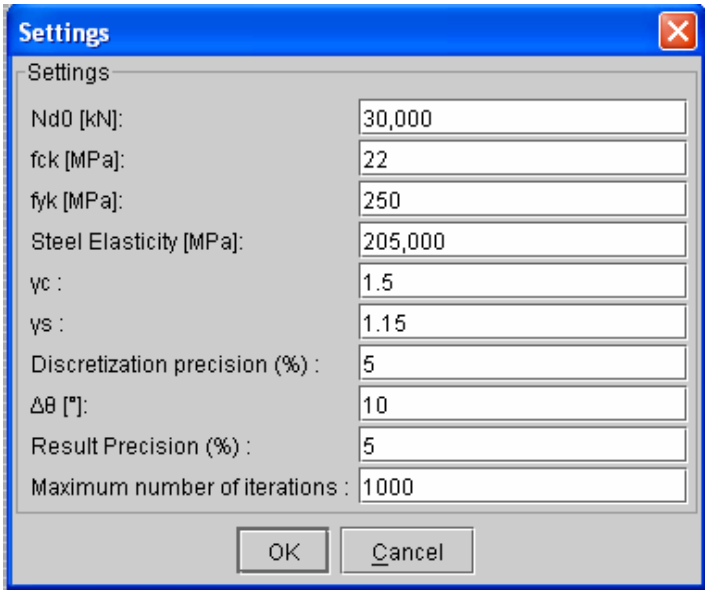
(c) – Diagrama de Interação obtido e Verificação

Figura 15 – Exemplo 2: Flexão Composta Oblíqua (VENTURINI E BORTOLIN, 1992).

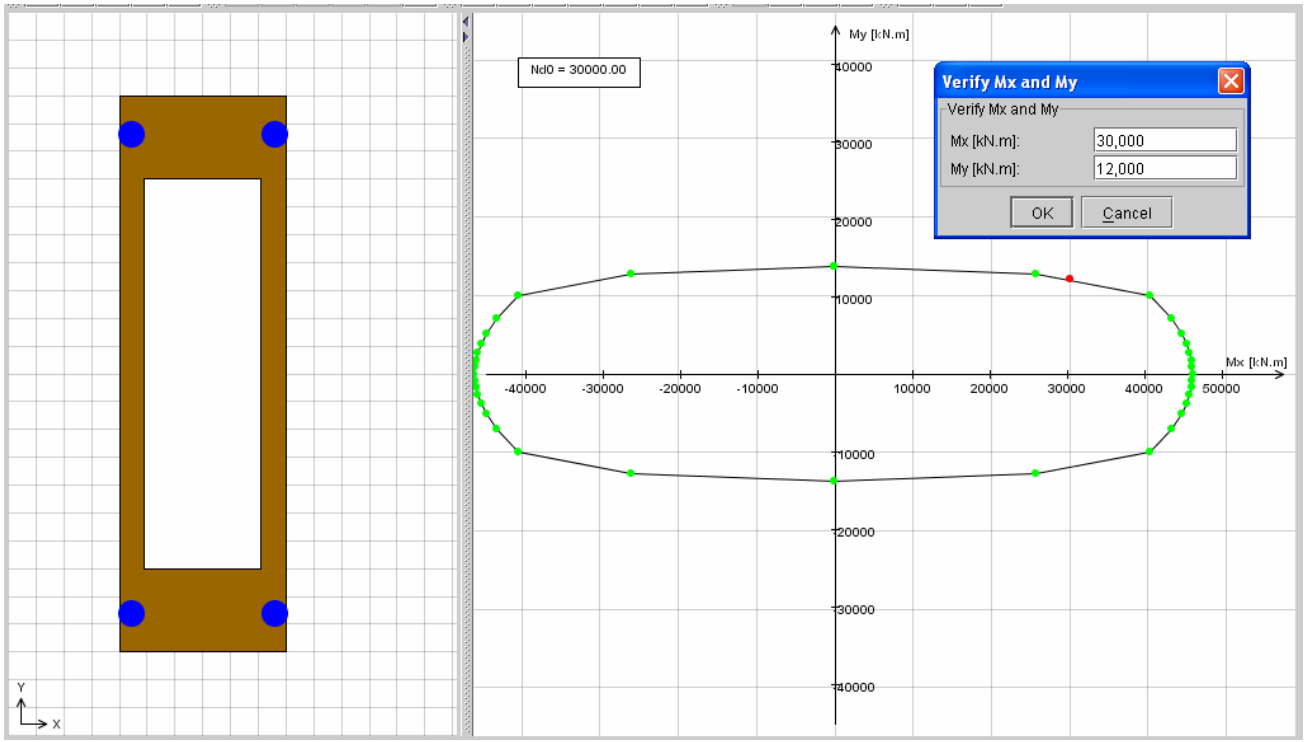
4.3 Exemplo 3 (PFEIL, 1976)



(a) Seção Transversal (4 ϕ 210 mm)



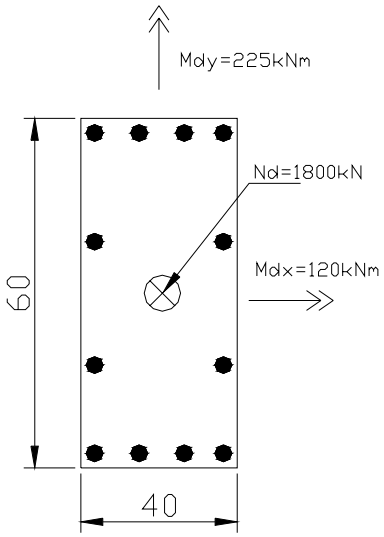
(b) Dados do Problema



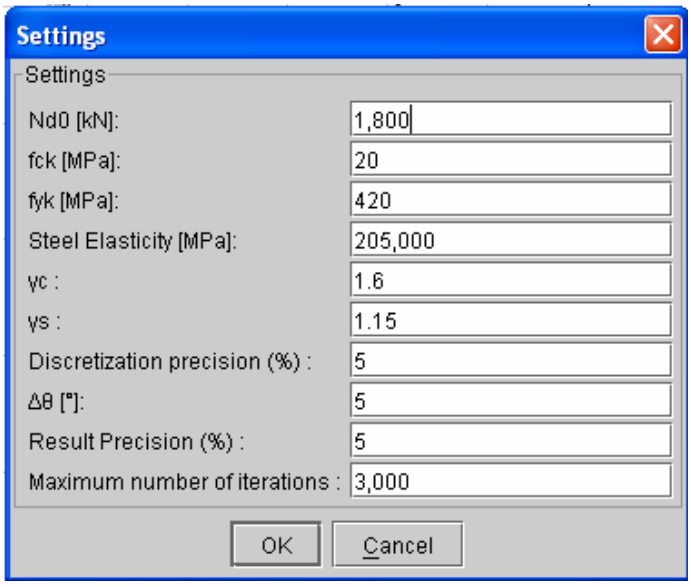
(c) – Diagrama de Interação obtido e Verificação

Figura 16 – Exemplo 3: Flexão Composta Oblíqua (PFEIL, 1976).

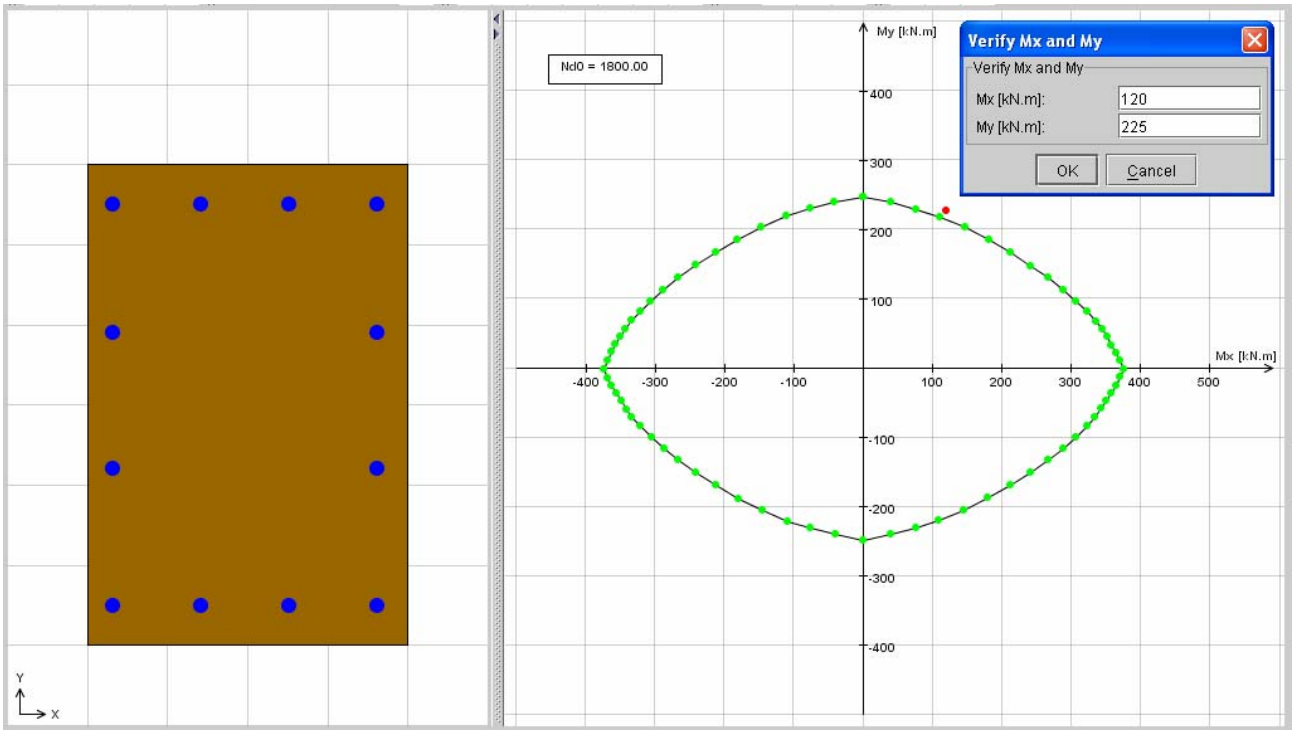
4.4 Exemplo 3 (MONTOYA, MESEGUER E CABRÉ, 1979)



(a) Seção Transversal (12 ϕ 20 mm)



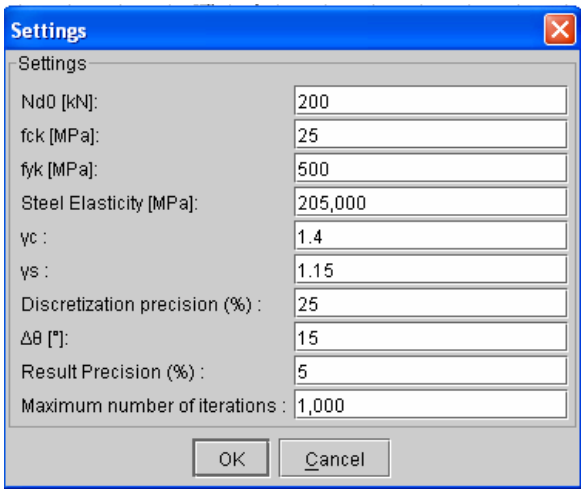
(b) Dados do Problema



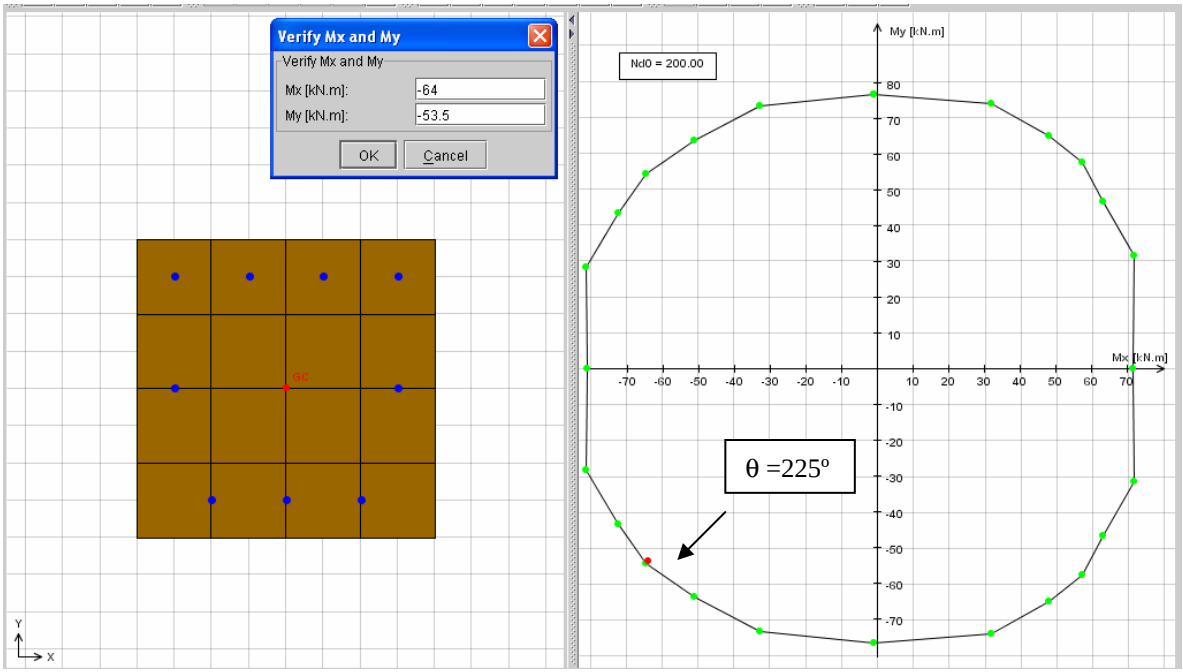
(c) – Diagrama de Interação obtido e Verificação

Figura 17 – Exemplo 4: Flexão Composta Oblíqua (MONTOYA, MESEGUER E CABRÉ, 1979).

4.5 Exemplo 4 - Seção não-simétrica



(a) Dados do Problema (9 ϕ 10 mm)



(b) – Diagrama de Interação obtido e Verificação

Figura 18 – Exemplo 5: Flexão Composta Oblíqua.

5 Considerações Finais

O artigo apresentou um ambiente gráfico interativo para obtenção de diagramas de interação de seções de concreto armado submetidas a esforços de flexão oblíqua composta: o software GERADIA, implementado através da metodologia de Programação Orientada a Objetos utilizando a linguagem de programação Java.

O software desenvolvido apresenta uma interface gráfica bastante amigável ao usuário, facilitando sua utilização. Após fornecer uma seção poligonal genérica de concreto e uma distribuição aleatória de barras de aço, ou perfis de aço, é gerado um

gráfico com todos os possíveis pares de momentos fletores, com vetores perpendiculares entre si, que a seção é capaz de suportar.

Para validar os resultados obtidos pelo GERADIA foram realizadas inúmeras comparações entre as respostas geradas pelo programa e as obtidas através de ábacos disponíveis na literatura. Nestas comparações pode-se notar que os valores encontrados aproximaram-se bastante, quando não foram exatos, dos calculados pelo GERADIA. Um dos motivos para a variação nos valores obtidos pode estar no pequeno erro gerado na discretização da seção. Outra fonte de erro que se pode levar em consideração, seria a imprecisão na obtenção de valores nos ábacos, já que isso é feito por aproximações e interpolações lineares entre curvas plotadas.

6 Referências Bibliográficas

BOOCH, G., RUMBAUGH, J., & JABOBSON, I., 2000. UML - **Guia do Usuário**. Editora Campus.

GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J., 1995. **Design Paterns - Elements of Reusable Object-Oriented Software**. Addison-Wesley.

GRAND, M., 1998. **Patterns in Java - Volume 1**. Jonh Wiley and Sons.

HORSTMANN, C. S. & CORNELL, G., 2000. **Core Java 2 Volume II - Advanced Features**. Editora Prentice Hall.

MONTOYA, P. J., MESEGUER, A. G. & CABRÉ, F. H., 1979. **Hormigón Armado**, Editora Gustavo Gili S.A., Barcelona.

PFEIL, W.,1976. **Dimensionamento do Concreto Armado à Flexão Composta**, Livros Técnicos e Científicos Editora S.A., Rio de Janeiro.

PIETRO, G. A., 2001. **Utilização de padrões de projeto na reengenharia de sistemas**. Dissertação de Mestrado, Universidade Federal de São Carlos, São Carlos, SP, Brasil.

ROWE, G. W., 2001. **Computer Graphics with Java**. Palgrave.

TEPEDINO, J. M., 1986. **Flexão Composta Oblíqua. Notas de Aula**, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

VENTURINI, W. S. & BORTOLIN, A. A., 1992. **Dimensionamento de Peças Retangulares de Concreto Armado Solicitados à Flexão Obliqua**, Universidade de São Paulo, Escola de Engenharia de São Carlos, Publicação 031/92.