U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

CÁLCULO ORGÂNICO DE SECÇÕES QUAISQUER EM FLEXÃO DESVIADA SEGUNDO O EUROCÓDIGO 2

HÉLDER DAVID FERNANDES MIRANDA

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM ESTRUTURAS

Orientador: Professor Doutor Álvaro Ferreira Marques Azevedo

Co-Orientador: Professor Doutor José Manuel de Sena Cruz

FEVEREIRO DE 2008

MESTRADO INTEGRADO EM ENGENHARIA CIVIL 2007/2008

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

Fax +351-22-508 1446

✉ miec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

Fax +351-22-508 1440

✉ feup@fe.up.pt

http://www.fe.up.pt

Aos meus Irmãos,

AGRADECIMENTOS

Agradeço aos Professores Álvaro Azevedo e José Sena Cruz, por todo o entusiasmo e rigor científico que me souberam incutir.

RESUMO

Este trabalho consiste no desenvolvimento de um código computacional destinado à resolução de problemas de verificação e dimensionamento de secções compósitas genéricas sujeitas a flexão composta desviada, de acordo com o Eurocódigo 2. O programa de computador implementado é muito robusto, permitindo abordar a generalidade dos casos práticos. A secção utilizada no cálculo pode ter uma qualquer geometria e ser constituída por vários tipos de materiais, como por exemplo o aço ou o FRP (com ou sem pré-esforço). O programa desenvolvido é de utilização simples, uma vez que está dotado de uma interface gráfica intuitiva. A quase totalidade dos dados pode ser introduzida pela interface gráfica e pode em seguida ser guardada em ficheiro. As fases de cálculo e de visualização dos resultados encontram-se também integradas na interface gráfica.

Foi efectuada a sintetização da informação relativa ao comportamento dos materiais correntemente utilizados em secções compósitas. Foram estudados modelos estruturais do comportamento de vigas sujeitas a esforços normais e de flexão, incluindo modelos propostos pela regulamentação existente.

A informação recolhida permitiu estabelecer um modelo de cálculo baseado num sistema de equações não lineares correspondente ao equilíbrio estático da peça. Para evitar a manipulação de expressões demasiado complexas e para que o modelo possa ser aplicado a qualquer caso foi concebido um algoritmo simplificado que avalia o erro associado a cada equação do sistema. Este procedimento é utilizado pelo método de Newton-Raphson, que constitui um processo iterativo destinado a calcular a solução do problema.

Foi em seguida necessário escolher uma linguagem de programação e uma biblioteca destinada a facilitar o desenvolvimento de uma interface gráfica intuitiva. A escolha recaiu na linguagem C++ e na biblioteca Microsoft Foundation Classes (MFC), tendo toda a programação sido efectuada com base em objectos.

Foram efectuados testes para averiguar individualmente o correcto funcionamento de cada um dos módulos, bem como testes ao funcionamento geral do programa. Os resultados obtidos foram validados com base em tabelas e ábacos de situações correntes, bem como em alguns resultados publicados que correspondem a casos de flexão composta desviada em secções não correntes.

PALAVRAS-CHAVE: modelação numérica de estruturas, estados limites últimos, flexão composta desviada, dimensionamento de secções compósitas, programação orientada por objectos.

ABSTRACT

In this work, a computational model for the analysis and design of composite members of arbitrary shape subjected to an axial force and biaxial bending was developed. For the case of concrete and steel, the Eurocode 2 recommendations are followed. The behavior of other materials must be carefully adapted to the numerical model. The resulting computer program is very robust and is able to address most practical design cases.

The cross section of the beam may be of an arbitrary shape and may be reinforced with any type of rebar, e.g., steel or FRP. These rebars may be prestressed. The utilization of the computer program that implements this model is very intuitive, since it has a modern graphical user interface. Practically all the required data can be defined with the graphical user interface and may be stored in a data file for later modifications. The calculation and result visualization phases are also integrated in the same graphical user interface.

The first phase of this work consisted on the appraisal of the behavior of several types of materials that can be used in composite members, e.g., CFRP, GRC. Several models for the analysis and design of composite beams subject to an axial force and biaxial loading were also studied, mainly those that are suggested by the design recommendations.

The analysis or the design of the member is based on a system of nonlinear equations that corresponds to the static equilibrium of all the cross section forces. In order to avoid the manipulation of complex expressions and to allow for the application of the model to a cross section of any shape, an algorithm to calculate each unbalanced force or moment was developed. This procedure is used by the Newton-Raphson method, which provides an iterative algorithm that converges to the solution of the system of nonlinear equations.

In order to provide an efficient and useful tool for any practicing engineer, a suitable computer language and a library of tools for the implementation of a graphical user interface add to be selected. A good compromise between efficiency, versatility and ease of development can be found in the C++ computer language and the Microsoft Foundation Classes (MFC). In order to benefit from the main characteristics of these tools, all the developed code is object oriented.

All the components of the developed code were individually tested, along with the full application and its graphical user interface. The results obtained with the developed computer program were compared with published results corresponding to classical cross section shapes and typical reinforcement layouts. Some validations were also carried out with biaxially bent beams with nonconventional cross sections.

KEY-WORDS: numerical modeling of structures, ultimate limit states, biaxial bending, design of composite beams, object oriented programming.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SÍMBOLOS

E_d - valor de cálculo do esforço actuante

R_d - valor de cálculo do esforço resistente

γ_g - coeficiente de segurança para acções permanentes

γ_q - coeficiente de segurança para acções variáveis

E_{Gik} - valor característico do esforço resultante de uma acção permanente

E_{Q1k} - valor característico do esforço resultante da acção variável considerada como acção base da combinação

E_{Qjk} - valor característico do esforço resultante de uma acção variável distinta da acção base

γ_{gi} - coeficiente de segurança relativo às acções permanentes

γ_{qi} - coeficiente de segurança relativo às acções variáveis

ψ_{0j} - coeficiente ψ correspondentes à acção variável de ordem *j*

f_{ck} – resistência característica do betão à compressão

f_{cd} - resistência de cálculo do betão à compressão

ε_{uk} - valor característico da extensão do aço da armadura para betão armado ou de pré-esforço na carga máxima

α_{cc} - coeficiente destinado a considerar os efeitos a longo prazo e as consequências desfavoráveis resultantes do modo como a carga é aplicada e afecta a tensão máxima de cálculo do betão

β - inclinação do eixo neutro de um campo de extensões relativamente à direcção horizontal

D - parâmetro definidor do tipo de rotura que ocorre numa secção em flexão composta desviada

A_s - área de armadura

N - esforço axial

M_x - momento flector relativamente ao eixo dos xx

M_y - momento flector relativamente ao eixo dos yy

1
INTRODUÇÃO

O trabalho apresentado neste documento consiste no desenvolvimento de um código computacional, para cálculo de secções compósitas à ruptura, sujeitas a esforços normais e de flexão.

Em certos casos simples é possível deduzir expressões analíticas que permitem resolver o referido problema. Aliás existem expressões, tabelas e ábacos adequados a casos particulares de secções de betão armado, calculados de acordo com o REBAP [1]. Vários autores encontram-se a preparar tabelas adequadas ao mesmo tipo de cálculo, mas segundo o EC2 [2].

No entanto é necessário referir que este tipo de problema na sua forma mais genérica, i.e., em casos de flexão composta desviada em secções de geometria quaisquer, não tem solução analítica, nem seria viável construir tabelas ou ábacos que abrangessem todos os casos possíveis. Esta situação torna-se ainda mais evidente se as relações tensão-extensão e, consequentemente, os tipos de material não forem fixados à partida, sendo um dado do problema.

Compreende-se assim o interesse inerente ao desenvolvimento de um programa que permite efectuar cálculos em elementos como: vigas caixão de pontes, núcleos rígidos de edifícios ou ainda em secções que utilizem materiais menos convencionais. O programa não deixa no entanto de facilitar a resolução dos problemas mais convencionais e de se encontrar preparado para respeitar quer a regulamentação do REBAP [1], quer a regulamentação do EC2 [2].

Por outro lado, pretendia-se um programa de cálculo autónomo que pudesse ser utilizado na prática com grande simplicidade e que apresentasse uma elevada portabilidade. Deste modo justifica-se o inevitável envolvimento de conceitos relacionados com o desenho assistido por computador ou a aplicação do paradigma da programação orientada por objectos, de forma a conseguir-se uma maior facilidade para o utilizador na definição dos problemas, bem como uma eficaz representação das soluções obtidas.

2
ENQUADRAMENTO DO ESTUDO

2.1. INTRODUÇÃO

Neste trabalho é apenas abordada a verificação do estado limite último de resistência. Contudo, numa situação real de projecto, para se garantir a segurança e a qualidade da estrutura, é necessário verificar também os estados limites de encurvadura, fendilhação e deformação, entre outros.

2.2. ESTADO LIMITE ÚLTIMO DE RESISTÊNCIA

A verificação de segurança em relação ao estado limite último de resistência deve, em geral, ser feita com base nos esforços respeitando a condição (1), sendo E_d o valor de cálculo do efeito das acções e R_d o valor de cálculo do esforço resistente [3].

$$E_d \leq R_d \tag{1}$$

Os valores de cálculo do efeito das acções são determinados com base num conjunto de combinações de acções, admitindo-se elasticidade perfeita dos materiais, afectadas de um coeficiente de segurança (γ_G para acções permanentes e γ_Q para acções variáveis). Estes coeficientes de segurança estão regulamentados e indicados na Tabela 1 para situações de projecto persistentes e transitórias [3].

Tabela 1 – Coeficientes de segurança parciais em ELU

E.L.U.	Acções permanentes	Acções variáveis	Pré-esforço
	γ_G	γ_Q	γ_P
Efeito favorável	1.00	0.00	1.00
Efeito desfavorável	1.35	1.50	1.00

Para o cálculo do esforço actuante, devem ser consideradas as combinações de acções, cuja actuação simultânea seja verosímil e que produzam na estrutura, os efeitos mais desfavoráveis.

Em geral tem-se

$$\sum_{i=1}^{m} \gamma_{G,i} G_{k,i} \text{ "+" } \gamma_P P \text{ "+" } \gamma_Q \left[Q_{k,1} + \sum_{j=2}^{n} \Psi_{0,j} Q_{k,j} \right] \tag{2}$$

sendo

$G_{k,i}$ - valor característico de uma acção permanente

$Q_{k,1}$ - valor característico da acção variável considerada como acção base da combinação

$Q_{k,j}$ - valor característico do esforço resultante de uma acção variável distinta da acção base

$\gamma_{G,i}$ - coeficiente parcial relativo às acções permanentes

γ_Q - coeficiente parcial relativo às acções variáveis

$\psi_{0,j}$ - coeficiente ψ correspondentes à acção variável de ordem j.

Os valores de cálculo dos esforços resistentes devem ser determinados por intermédio de uma teoria do comportamento e partir das propriedades dos materiais que constituem a estrutura, convenientemente quantificadas para atender à segurança pretendida. Estes valores são calculados a partir do valor característico da resistência dos materiais reduzido de um factor γ_m, cujos valores estão indicados na Tabela 2 [2].

Tabela 2 – Coeficientes parciais relativos aos materiais para ELU

Situações de Projecto	Betão	Aço das armaduras para betão armado	Aço de Pré-Esforço
	γ_C	γ_S	γ_P
Persistente	1.5	1.15	1.15

2.3. ANÁLISE COMPARATIVA ENTRE O REBAP E O EUROCÓDIGO 2

2.3.1. BETÃO COMPRIMIDO

Segundo o REBAP [1], as relações tensões-extensões de cálculo do betão à compressão a considerar na determinação dos valores de cálculo dos esforços resistentes para a verificação do Estado Limite Último (ELU) de resistência devem em geral ser as indicadas na Figura 1.

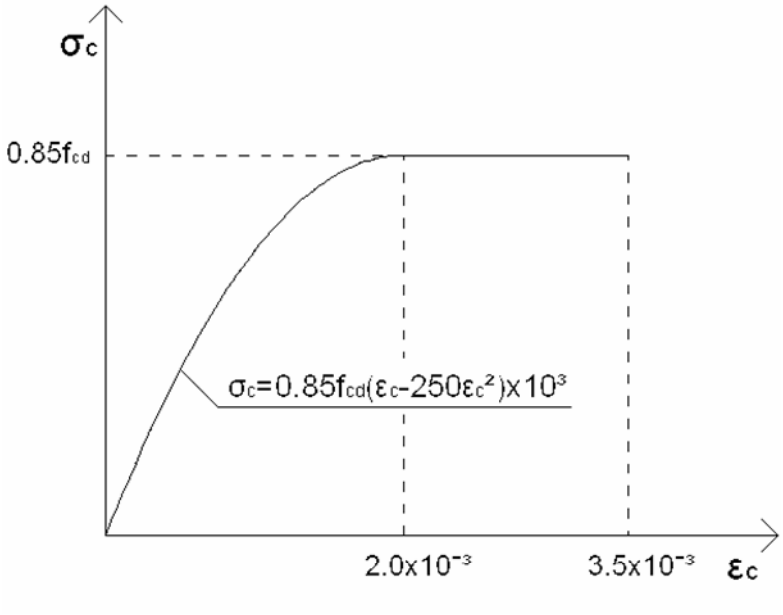


Fig.1 – Curva tensão-extensão de cálculo segundo o REBAP

Até uma extensão máxima de 2×10^{-3} o valor da tensão máxima de cálculo do betão à compressão é dado pela expressão indicada na Figura 1. Para extensões superiores e até ao valor máximo de 3.5×10^{-3}, a tensão máxima deve ser limitada a $0.85f_{cd}$, para considerar uma possível diminuição da tensão de rotura do betão quando sujeito prolongadamente a tensões elevadas.

No Eurocódigo 2 [2] as relações tensões-extensões de cálculo admitidas são as indicadas na Figura 2.

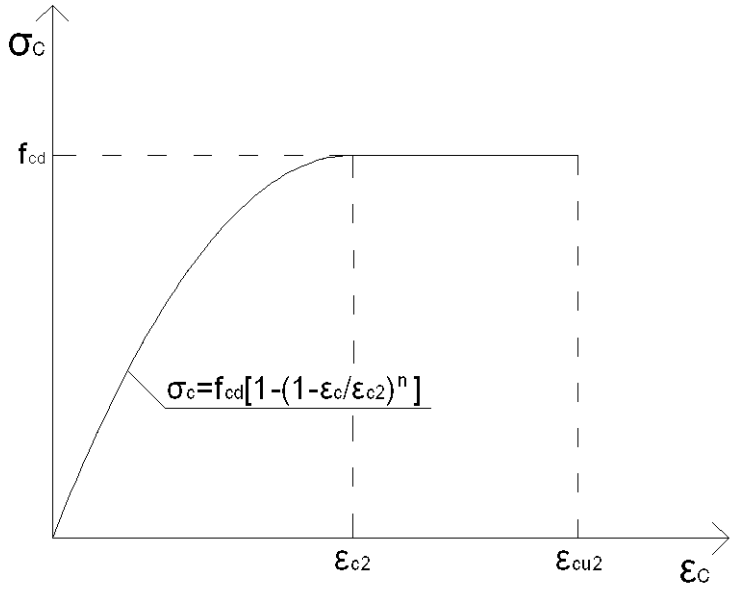


Fig.2 – Curva tensão-extensão de cálculo segundo o Eurocódigo 2

As tensões são definidas pela expressão indicada na Figura 2, que é válida até uma extensão máxima ε_{c2}, tomando a partir dessa extensão, e até à extensão última ε_{cu2}, o valor de f_{cd}.

$$f_{cd} = \alpha_{cc} \frac{f_{ck}}{\gamma_c} \tag{3}$$

Assim, pode-se constatar que no EC2 [2], contrariamente ao estipulado pelo REBAP [1], o valor das extensões máximas permitidas variam de acordo com a classe de betão a utilizar. No EC2 [2] o coeficiente α_{cc} destina-se a considerar os efeitos a longo prazo e as consequências desfavoráveis resultantes do modo como a carga é aplicada e afecta a tensão máxima de cálculo do betão. Este coeficiente varia entre 0.8 e 1. Os valores a atribuir aos parâmetros atrás referidos encontram-se indicados na Tabela 3.

Tabela 3 – Características de resistência e de deformação do betão

Classes de Resistência do Betão														
f_{ck} (MPa)	12	16	20	25	30	35	40	45	50	55	60	70	80	90
$f_{ck,cubo}$ (MPa)	15	20	25	30	37	45	50	55	60	67	75	85	95	105
ε_{c2} (‰)	2									2.2	2.3	2.4	2.5	2.6
ε_{cu2} (‰)	3.5									3.1	2.9	2.7	2.6	2.6
n	2									1.75	1.6	1.45	1.4	1.4

2.3.2. AÇO PARA BETÃO ARMADO

Segundo o REBAP [1], as relações tensões-extensões de cálculo dos diversos tipos de aço estão limitadas, independentemente das classes de aço e betão a ser utilizado, aos valores de 3.5×10^{-3} (extensão última do betão) à compressão e a 10×10^{-3} no caso de se encontrar traccionado (ver Figura 3).

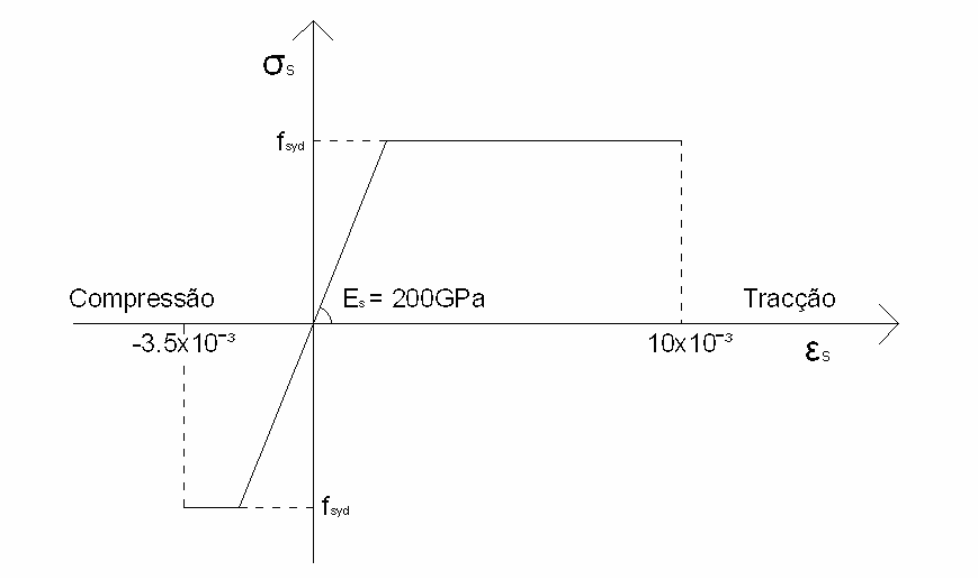


Fig.3 – Diagrama bilinear do aço de acordo com o REBAP

O EC2 admite para o cálculo corrente qualquer uma das seguintes hipóteses (ver Figura 4):

a) um ramo superior inclinado com uma extensão limite ε_{ud} e uma tensão máxima $k f_{yk}/\gamma_s$ para ε_{uk}, em que $k=(f_t/f_y)_k$. O valor limite da extensão de cálculo ε_{ud} depende de normas específicas para cada país. O valor definido para Portugal e também o recomendado pelo EC2 é de $0.9\varepsilon_{uk}$. Por sua vez os valores de ε_{uk} e de k dependem de uma classe de ductilidade (A, B ou C) atribuída ao aço em causa.

b) um ramo superior horizontal sem necessidade de verificação do limite de extensão.

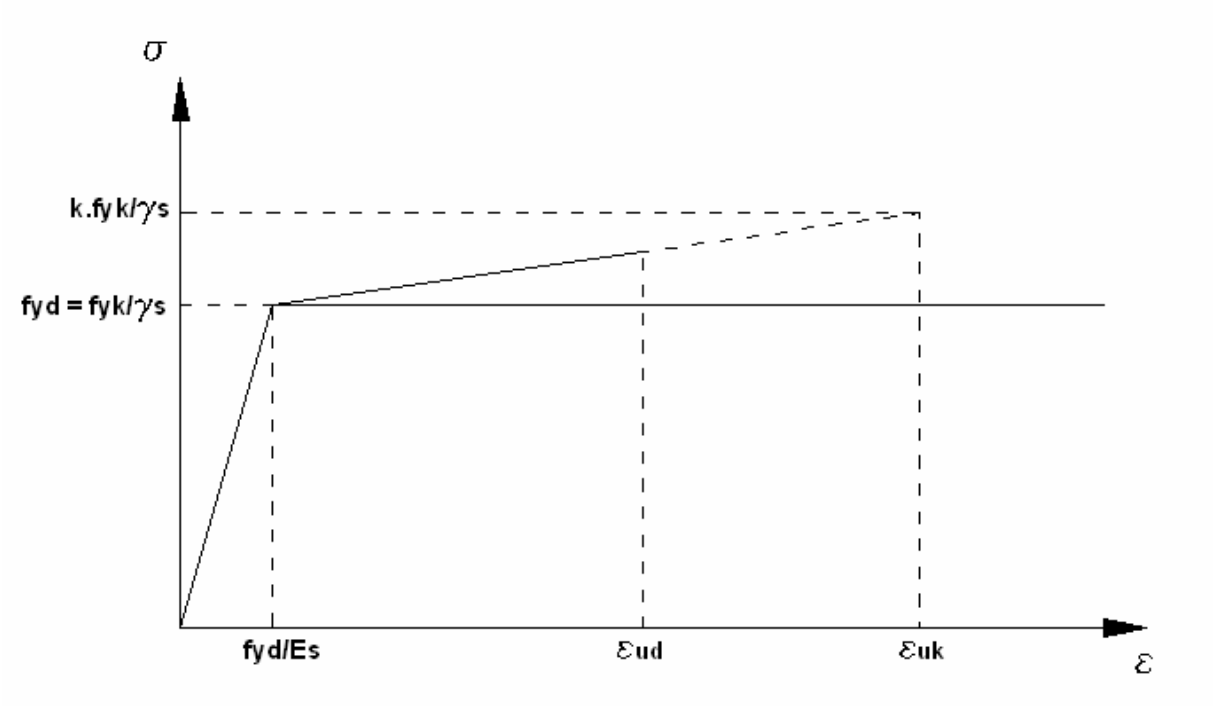


Fig.4 – Diagrama bilinear do aço segundo o EC2

A hipótese b) é semelhante à já adoptada pelo REBAP. É também mais conservativa do que a hipótese de cálculo a) dado que ignora alguma da capacidade resistente desenvolvida pelo aço, já depois deste entrar em cedência.

2.4. MATERIAIS FRP

Nas últimas décadas os materiais compósitos têm vindo a assumir uma maior importância como materiais estruturais. A necessidade de procurar meios cada vez mais práticos de reforçar estruturas existentes e alternativas aos materiais convencionais tem levado a uma crescente utilização de materiais baseados em polímeros reforçados com fibras (FRP).

Os materiais FRP consistem em aglomerados de finas fibras não metálicas, contínuas e direccionadas, aglutinadas por uma matriz de resina. Dependendo do tipo de fibra podem classificar-se em:

AFRP – baseados em fibras de aramida

CFRP – baseados em fibras de carbono

GFRP – baseados em fibras de vidro

A utilização de materiais FRP apresenta inúmeras vantagens entre as quais se destacam as seguintes:

- não estão sujeitos à corrosão;
- baixo peso volúmico (cerca de um quarto do valor correspondente ao aço), reduzindo as cargas na estrutura e facilitando a sua aplicação;
- tensão de ruptura elevada;
- disponíveis com grande diversidade de formas e dimensões.

Este tipo de material apresenta no entanto algumas desvantagens que não devem ser negligenciadas, nomeadamente:

- os FRP apresentam um funcionamento elástico até a ruptura, sem desenvolver deformação plástica, o que faz com que tenham uma ductilidade muito reduzida;
- alguns tipos de FRP, por exemplo em carbono e aramida, apresentam um coeficiente de expansão térmica significativamente diferente do correspondente ao betão, o que pode induzir elevadas tensões nos materiais;
- quando exposto a elevadas temperaturas, o FRP fica sujeito a uma degradação prematura que pode eventualmente provocar o colapso da estrutura;
- os custos associados ao material FRP são muito elevados comparativamente com os correspondentes ao aço.

É portanto recomendável ponderar a utilização do FRP, considerando entre outros aspectos o processo construtivo e a duração a longo prazo e não somente o comportamento resistente, do ponto de vista mecânico. Esta discussão ultrapassa no entanto o âmbito deste trabalho.

O diagrama tensão-extensão idealizado e típico de um material FRP é linear até á ruptura, tal como se mostra na Figura 5.

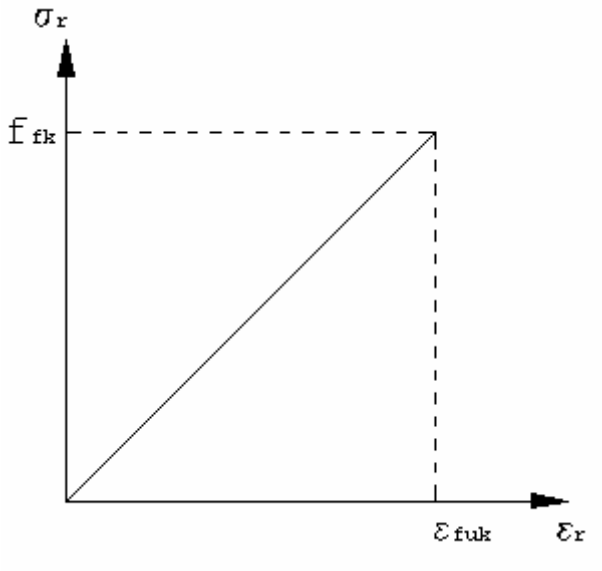


Fig.5 – Diagrama tensão-extensão idealizado para o material FRP

Os parâmetros f_{fk} e ε_{fuk} indicados na Figura 5 são, respectivamente, os valores característicos da resistência e extensão últimas. Assim, o módulo de elasticidade E_{fu} pode ser calculado por intermédio da expressão que se segue.

$$E_{fu} = \frac{f_{fk}}{\varepsilon_{fuk}} \tag{4}$$

Os coeficientes de segurança a utilizar para os materiais FRP variam consoante o tipo de fibra utilizado e o modo e tipo de cuidados com que são aplicados. Os coeficientes de segurança que devem ser adoptados são os indicados na Tabela 4 [5].

Tabela 4 – Coeficientes de segurança parciais para o material FRP

Tipo de FRP	Aplicação tipo A	Aplicação tipo B
CFRP	1.2	1.35
AFRP	1.25	1.45
GFRP	1.3	1.5

Na Tabela 4 as aplicações do tipo A são aplicações com controlo de qualidade normal em sistemas pré-fabricados, ou aplicações *in-situ* sob condições de elevado controlo de qualidade. As aplicações do tipo B são aplicações executadas *in-situ* em que o controlo de qualidade esteja abaixo do normal, ou em qualquer tipo de aplicação em que existam difíceis condições de trabalho no local.

3
DESCRIÇÃO DO MODELO

3.1. INTRODUÇÃO

O equilíbrio estático de uma secção compósita genérica (incluindo de betão armado) sujeita à flexão composta desviada pode ser traduzido matematicamente pelo sistema que se segue.

$$\begin{cases} \sum N = 0 \\ \sum M_x = 0 \\ \sum M_y = 0 \end{cases} \quad (5)$$

No âmbito do presente trabalho são sempre consideradas como incógnitas a inclinação do eixo neutro e o tipo de rotura que ocorre na secção, definidas com recurso a duas variáveis, β e D, respectivamente. É necessário escolher uma terceira incógnita, que pode ser a área de armaduras a dimensionar ou um dos esforços resistentes, dado que o sistema possui três equações.

Uma vez que as equações que constituem o sistema (5) não são lineares, utiliza-se um método iterativo para a sua resolução numérica com base em aproximações sucessivas (método de Newton-Raphson) [6].

No âmbito do software desenvolvido apenas é necessário criar uma subrotina que fornece os três desequilíbrios (ΔN, ΔMx, ΔMy) e juntá-la ao programa genérico de resolução de um qualquer sistema de equações não lineares. Os referidos desequilíbrios são calculados a partir das resultantes das tensões nos dois materiais e respectivos pontos de aplicação para cada conjunto de valores das incógnitas.

Assim, fornecendo ao programa as características dos materiais utilizados bem como os contornos da secção e posicionamento das armaduras, obtêm-se os esforços resistentes ou a área de armaduras a dimensionar.

3.2. DEFINIÇÃO DA SECÇÃO

O modelo desenvolvido aplica-se a uma secção qualquer, a qual deve estar definida pelos vértices constituintes dos polígonos que a delimitam e pela respectiva posição das armaduras. Estas coordenadas podem ser apresentadas segundo um qualquer referencial cartesiano, tendo em atenção que os esforços são aplicados na origem do mesmo. São considerados como positivos os esforços

axiais de tracção e os momentos que possuam o respectivo vector na direcção dos eixos, segundo a regra do "saca-rolhas". São consideradas positivas as extensões de alongamento.

3.2.1. MATERIAL BASE

No modelo utilizado define-se material base como o material pelo qual a secção é essencialmente constituída. Assim por exemplo no caso de uma secção em betão armado, o material base é o betão.

Um dado material base é caracterizado por uma curva tensão-extensão própria.

No programa desenvolvido é possível definir estas curvas recorrendo a ficheiros correspondentes aos materiais base, contendo um dos dois conjuntos de informação:

a) os vários parâmetros definidores de uma classe de betão de acordo com o EC2, no caso do material de base ser betão;

b) uma lista de coordenadas correspondentes a pontos da curva tensão-extensão.

3.2.2. MATERIAIS PARA ARMADURA

Os materiais que constituem as armaduras da secção (por exemplo aço ou FRP) são igualmente caracterizados por curvas tensão-extensão próprias.

No modelo adoptado admite-se que estas relações podem ser definidas recorrendo às coordenadas de cinco pontos, de acordo com a figura que se segue.

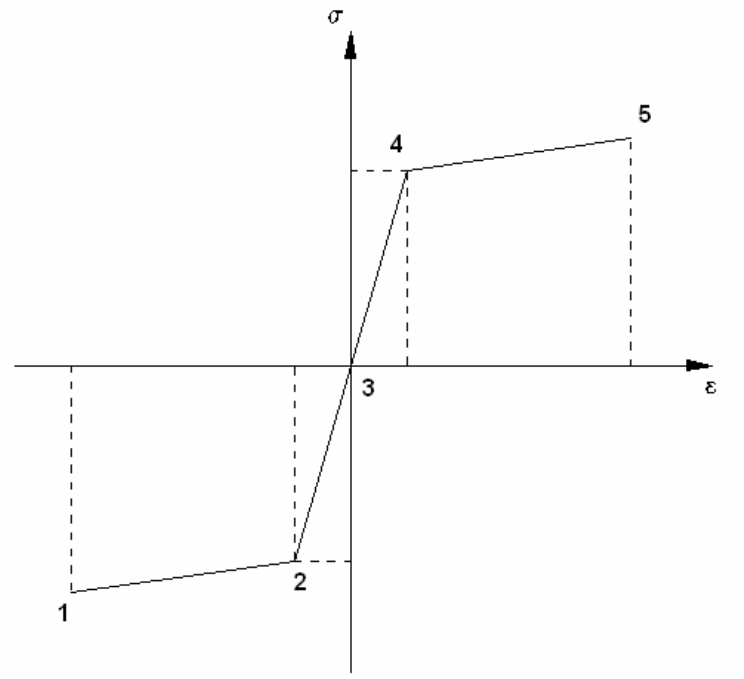


Fig.6 – Diagrama tensão-extensão genérico para um material de armadura

Estes dados correspondentes a cada material de armadura são guardados em ficheiros apropriados para facilitar a sua introdução no programa.

Se as armaduras em causa estiverem sob o efeito de uma tensão inicial correspondente a um pré-esforço é igualmente possível estabelecer esse valor de tensão inicial na altura em que se define a secção a calcular.

3.3. EQUAÇÕES DE EQUILÍBRIO

Para satisfazer a condição de dimensionamento em ELU é necessário resolver o sistema de equações (6), que pode ser traduzido como um sistema não linear.

$$\begin{cases} N = \sum F_c + \sum F_s + \sum F_p \\ M_x = \sum F_{ci} z_{cxi} + \sum F_{sj} z_{sxj} + \sum F_{pk} z_{pxk} \\ M_y = \sum F_{ci} z_{cyi} + \sum F_{sj} z_{syj} + \sum F_{pk} z_{pyk} \end{cases} \quad (6)$$

Este sistema tem como incógnitas o domínio (D), a inclinação do eixo neutro (β) e um esforço resistente (N, M_x, M_y) ou a área de armadura (A_s), como se ilustra na Figura 7. As componentes principais do sistema são: F_c a força resultante das tensões no material base, F_s a força resultante na armadura e F_p a força resultante nas armaduras pré-esforçadas.

3.4. DOMÍNIOS DE DEFORMAÇÃO

O conjunto dos campos de extensão compatíveis com a ruptura da secção (domínios de deformação) foi parametrizado recorrendo a uma variável real D (ver Figura 8). Esta toma valores entre um e quatro, representando cada valor inteiro a transição entre diferentes tipos de rotura [7].

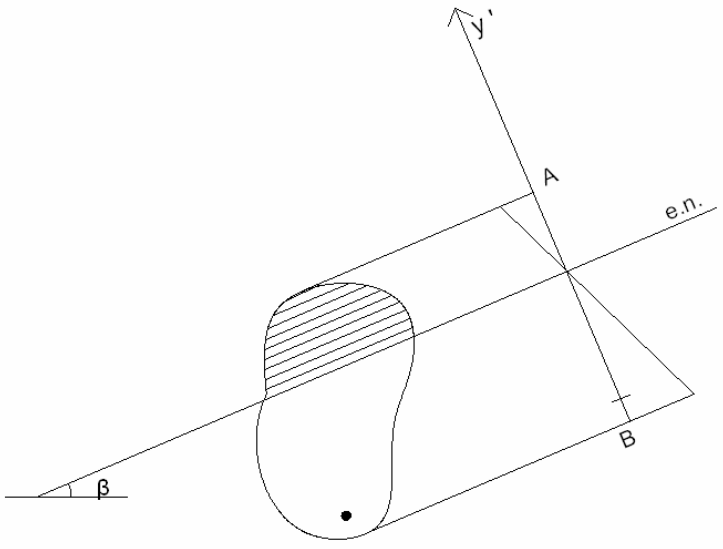


Fig.7 – Secção genérica com diagrama de extensões

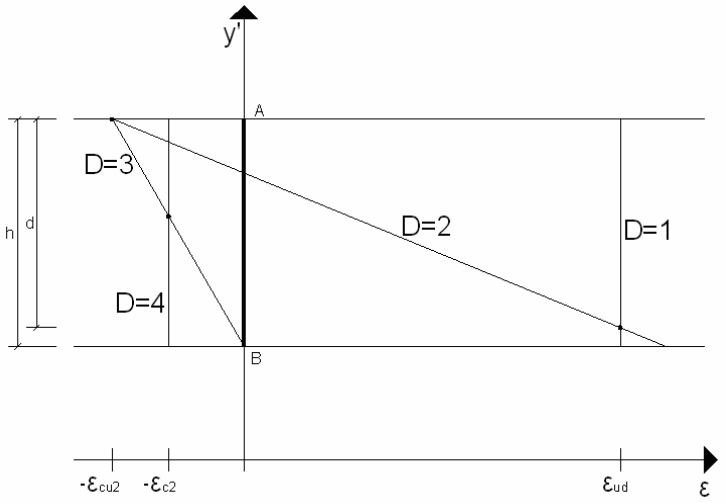


Fig.8 – Diagrama dos domínios de deformação

O domínio 1 é aquele onde a rotura ocorre por tracção simples, quando esta atinge a extensão última em algum dos pontos em que esteja colocada armadura. O domínio 2 ocorre quando as extensões da secção atingem os valores últimos tanto na zona comprimida como na zona traccionada. O domínio 3 corresponde a uma transição em que a secção deixa de estar parcialmente em tracção e passa a estar totalmente comprimida. Finalmente, o domínio 4 corresponde à rotura por compressão simples estando limitada a um determinado valor que depende do material de base.

Na Tabela 5 são apresentados os valores das extensões para cada valor inteiro de *D*.

Tabela 5 – Domínios de deformação

D	ε_{sup}	ε_{inf}
1	ε_{ud}	ε_{ud}
2	ε_{cu2}	K
3	ε_{cu2}	0
4	ε_{c2}	ε_{c2}

E em que valor de *K* é determinado como sendo a maior extensão de tracção possível na fibra do material base mais abaixo do eixo neutro, sem que nenhuma das armaduras ultrapasse o seu valor limite de tracção. Note-se que o valor de *K* não é forçosamente condicionado pela armadura em tracção mais abaixo do eixo neutro.

O valor *K* é calculado como o menor dos valores K^i determinados para cada ponto de colocação de armadura de índice *i* de acordo com a expressão que se segue.

$$K^i = \frac{(-\varepsilon_{cu2} + \varepsilon^i_{ud})}{d^i} h + \varepsilon_{cu2} \qquad (7)$$

Sendo *h* a altura da secção medida na perpendicular ao eixo neutro, ε_{cu2} a extensão máxima de compressão permitida no material base, ε^i_{ud} a extensão máxima de tracção permitida pelo material da

armadura localizada no ponto i e d_i a respectiva distancia à fibra mais comprimida do material base, medida na perpendicular ao eixo neutro. Assim consegue-se garantir o valor máximo de K, sem que para nenhuma das armaduras o limite máximo de extensão de tracção ε^i_{ud} seja excedido.

Para outro qualquer valor do domínio as correspondentes extensões no material base ε_{inf} e ε_{sup}, podem ser obtidas por interpolação linear, entre cada um dos domínios definidos na Tabela 3. Apresenta-se em seguida a forma de cálculo para cada um dos domínios

Extensões na fibra superior

$$D:1\to 2 \quad \Rightarrow \quad \varepsilon_{\sup} = \varepsilon_{ud} - (\varepsilon_{ud} + \varepsilon_{cu2})(D-1) \tag{8}$$

$$D:2\to 3 \quad \Rightarrow \quad \varepsilon_{\sup} = \varepsilon_{cu2} \tag{9}$$

$$D:3\to 4 \quad \Rightarrow \quad \varepsilon_{\sup} = \varepsilon_{cu2} - (\varepsilon_{cu2} - \varepsilon_{c2})(D-3) \tag{10}$$

Extensões na fibra inferior

$$D:1\to 2 \quad \Rightarrow \quad \varepsilon_{\inf} = \varepsilon_{ud} + (K - \varepsilon_{ud})(D-1) \tag{11}$$

$$D:2\to 3 \quad \Rightarrow \quad \varepsilon_{\inf} = K - K(D-2) \tag{12}$$

$$D:3\to 4 \quad \Rightarrow \quad \varepsilon_{\inf} = \varepsilon_{cu2}(D-3) \tag{13}$$

3.5. ESFORÇOS RESULTANTES NO MATERIAL BASE

A força e os momentos correspondentes ao funcionamento mecânico do material base resultam da actuação de um campo de tensões. As suas resultantes são assim calculadas recorrendo a integrais de superfície, referidos nas expressões que se seguem.

$$F_x = \int_A \sigma(x,y)\,dA \tag{14}$$

$$M_x = \int_A y\,\sigma(x,y)\,dA \tag{15}$$

$$M_y = -\int_A x\,\sigma(x,y)\,dA \tag{16}$$

As curvas tensão-extensão do material base não são definidas à partida por expressões analíticas e domínio de integração limitado pelos contornos da secção pode ser irregular. Esta situação leva a que

os integrais referidos nas expressões (14), (15) e (16) sejam calculados recorrendo a processos numéricos.

O processo adoptado consiste essencialmente no seguinte:

1) Determinar as coordenadas dos vértices da secção num sistema de eixos definido deforma a que o eixo das abcissas fique paralelo ao eixo neutro (ver Figura 9). Assim consegue-se que o campo de tensões varie apenas segundo o eixo das ordenadas.
2) Dividir a secção em varias bandas uniformemente espaçadas e paralelas ao eixo das abcissas.
3) Calcular para cada banda de abcissa y, o valor da tensão $\sigma(y)$ e o valor de cada um dos integrais (14) (15) e (16), sobre o domínio da própria banda, admitindo para tal tensão constante.
4) Somar devidamente as parcelas obtidas no ponto 3 para cada um dos integrais que se pretende calcular.
5) Calcular os momentos relativamente ao sistema de eixos original.

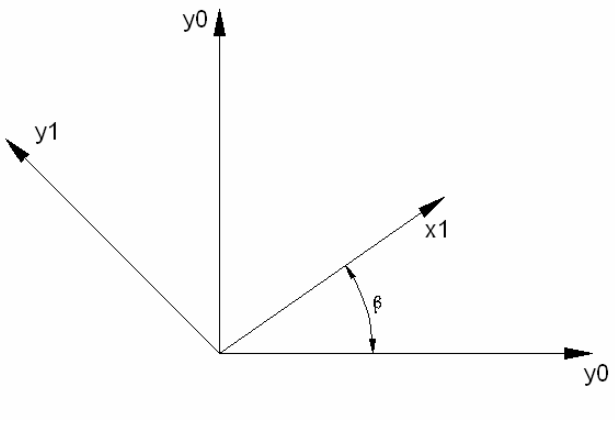


Fig.9 – Sistema de coordenadas rodado de um ângulo β relativamente ao original

As referidas mudanças entre sistemas de coordenadas de pontos e vectores são conseguidas recorrendo a transformações definidas pelas expressões que se seguem.

$$M_{x0} = M_{x1}\cos(\beta) - M_{y1}\sin(\beta) \tag{17}$$

$$M_{y0} = M_{x1}\sin(\beta) + M_{y1}\cos(\beta) \tag{18}$$

Sendo M_{x1} e M_{y1} coordenadas num sistema de eixos x_1Oy_1 que se encontra rodado de um ângulo β relativamente a um outro sistema x_0Oy_0 (fig. 9) e em que as respectivas coordenadas são M_{x0} e M_{y0}.

3.6. ESFORÇOS RESULTANTES NAS ARMADURAS

Os esforços resultantes nas armaduras são calculados com base nas respectivas relações tensão-extensão de cada material. Assim conhecida a extensão em cada ponto em que a armadura é colocada é possível obter a tensão. Multiplicando o valor da tensão pela área obtém-se a respectiva força actuante no ponto. O momento correspondente a cada ponto calcula-se multiplicado a força pelo respectivo braço.

No caso de a armadura em causa ser pré-esforçada, a força correspondente ao pré-esforço é composta por duas parcelas, uma devida à tensão aplicada inicialmente e outra causada pela variação de extensão (variação da tensão do pré-esforço efectivo). Tendo em atenção que o pré-esforço aplicado está na sua totalidade no lado da resistência e que se trata de um dimensionamento ou verificação de uma estrutura hiperestática, no momento actuante (M_{Ed}) deve ser adicionada a parcela correspondente ao momento hiperestático de pré-esforço.

3.7. RESOLUÇÃO DO SISTEMA DE EQUAÇÕES DE EQUILÍBRIO

Para a resolução do sistema de equações não lineares recorre-se a um processo numérico iterativo baseado no método de Newton-Raphson. Tendo em vista um aumento da robustez do algoritmo foi implementada uma técnica de "line-search" [6]. O método de Newton-Raphson destina-se, na sua forma unidimensional, ao cálculo iterativo das raízes de uma qualquer equação, que de forma genérica pode ser traduzida por

$$g(x) = 0 \tag{19}$$

Considerando os dois primeiros termos do desenvolvimento em série de Taylor da função g(x), tem-se

$$g(x)^{r+1} \cong g(x)^r + g'(x)^r \Delta x^r \tag{20}$$

Adoptando um valor inicial x^0 e conhecida a derivada da função, pode-se chegar a um valor mais próximo da raiz usando as seguintes expressões

$$x^{r+1} = x^r + \Delta x^r \tag{21}$$

$$\Delta x^r = -\frac{g(x)^r}{g'(x)^r} \tag{22}$$

Este processo é repetido sucessivamente até se obter a convergência para uma das soluções.

Generalizando o método de Newton-Raphson de forma a resolver um sistema de equações não lineares de *n* equações a *n* incógnitas, tem-se

$$\begin{cases} g_1(x_1,\ldots,x_n) = 0 \\ \quad\vdots \qquad\qquad \vdots \\ g_n(x_1,\ldots,x_n) = 0 \end{cases} \tag{23}$$

Desenvolvendo cada equação em série de Taylor resulta

$$\begin{cases} g_1(x_1,\ldots,x_n)^{r+1} \cong g_1(x_1,\ldots,x_n)^r + \left(\frac{\partial g_1}{\partial x_1}\right)^r \Delta x_1^r + \ldots + \left(\frac{\partial g_1}{\partial x_n}\right)^r \Delta x_n^r = 0 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ g_n(x_1,\ldots,x_n)^{r+1} \cong g_n(x_1,\ldots,x_n)^r + \left(\frac{\partial g_n}{\partial x_1}\right)^r \Delta x_1^r + \ldots + \left(\frac{\partial g_n}{\partial x_n}\right)^r \Delta x_n^r = 0 \end{cases} \quad (24)$$

Cada uma destas equações pode ser escrita na forma vectorial, resultando

$$g_i^{r+1} \cong g_i^r + \sum_j \left(\frac{\partial g_i}{\partial x_j}\right)^r \Delta x_j^r = 0 \quad (25)$$

Assim, a partir de uma solução aproximada, x^r, pode calcular-se uma solução mais próxima da exacta, x^{r+1}, aplicando a seguinte expressão.

$$\underset{\sim}{x}^{r+1} = \underset{\sim}{x}^r + \Delta \underset{\sim}{x}^r \quad (26)$$

O sistema de equações lineares que resulta da aplicação do método de Newton-Raphson é, neste contexto, sempre constituído por três equações. Por esse motivo, e por simplicidade, é utilizada a regra de Cramer [8] para se obter a correspondente solução. Convém ainda referir que é utilizada uma técnica designada de line-search para facilitar a convergência do processo iterativo [6]. Os acréscimos a cada solução corrente são pré-multiplicados por um factor de *line-search*, determinado no intervalo]0,1], de forma a obter uma solução mais próxima da solução exacta do que fazendo apenas uma correcção directa segundo (25).

4

PROGRAMAÇÃO ORIENTADA POR OBJECTOS

4.1. INTRODUÇÃO

Este capítulo destina-se a dar uma perspectiva geral sobre o paradigma da programação orientada por objectos, uma vez que o programa desenvolvido baseia-se em grande medida no emprego de várias técnicas de programação, que são adiante descritas. São apresentadas as principais metodologias e objectivos da programação orientada por objectos, fazendo-se também referência a alguns tipos de estruturas de dados genéricas, com elevado interesse no contexto deste trabalho.

4.2. PARADIGMAS DE PROGRAMAÇÃO

A primeira etapa a ultrapassar quando alguém se pretende iniciar na escrita de programas de computador, consiste em disciplinar o raciocínio de forma a conseguir traduzir a resolução de um problema segundo uma sequência lógica de operações sobre estruturas de dados, cuidadosamente escolhidas. Esta sequência lógica de operações constitui o chamado algoritmo.

Vencida essa primeira etapa é possível escrever alguns programas. No entanto quando os problemas a resolver assumem elevado grau de complexidade, tanto ao nível das estruturas de dados utilizadas, como das operações a realizar, torna-se necessário aprender a estruturar o programa sob uma forma adequada, de forma a reduzir essa mesma complexidade.

A um conjunto de estratégias e de estilos utilizados para estruturar os programas dá-se o nome de paradigma da programação. As linguagens e os paradigmas de programação têm evoluído, acompanhando a evolução tecnológica e a capacidade de cálculo, capaz de abordar problemas cada vez mais complexos.

Na tabela que se segue são apresentadas as principais gerações de linguagens e paradigmas que lhes deram origem [9].

Tabela 6 – Evolução das linguagens e paradigmas da programação

Geração	Década	Paradigmas de programação	Linguagens de suporte
1ª	50 e 60	Programação não estruturada	Cobol, Fortran, Basic
2ª	70	Procedimental	Pascal, linguagem C
3ª	80	Modular	Modula II
4ª	80	Abstracção de tipos de dados	ADA
5ª	80 e 90	Programação orientada por objectos	Smalltalk, C++, Java

Geralmente uma linguagem que suporta um dado paradigma de programação suporta igualmente paradigmas anteriores. Além disso, a linguagem adoptada não obriga a que se utilize um determinado estilo de programação.

4.2.1. PROGRAMAÇÃO PROCEDIMENTAL

O paradigma procedimental permite reduzir a complexidade dos algoritmos através da estruturação do programa em procedimentos mais simples. Cada procedimento ou função permite realizar subtarefas do programa e é escrito de forma pouco complexa, uma vez que existe a possibilidade de esse procedimento invocar outros procedimentos para realizar outras tarefas. Assim, o programa global é escrito de forma simplificada com base em procedimentos, que por sua vez são escritos com base noutros procedimentos, e assim sucessivamente. O programa é assim estruturado em sub-algoritmos que realizam tarefas de complexidade reduzida.

4.2.2. PROGRAMAÇÃO MODULAR

A programação modular assenta em princípios de encapsulamento de dados. O programa é decomposto em módulos de funcionamento que são constituídos por dados e procedimentos. Os referidos módulos são concebidos para que grande parte dos respectivos dados não possa ser acedida directamente, a não ser pelo próprio módulo, conseguindo-se assim uma maior simplicidade e eficácia na manipulação dos dados e na realização de tarefas.

Neste paradigma de programação o acesso aos dados de cada módulo só pode ser, na maior parte dos casos, efectuado por intermédio de uma interface constituída por um conjunto de procedimentos adequados.

4.2.3. ABSTRACÇÃO DE TIPOS DE DADOS

Este paradigma permite que o programador crie tipos de dados abstractos com as propriedades desejadas. Para tal é necessário definir esse tipo de dados recorrendo a um conjunto de outros dados e a um conjunto completo de operações que podem ser realizadas com esse tipo de dados.

Um exemplo típico é a declaração de um tipo de dados capaz de operar com números complexos. Um número complexo *z* pode ser escrito da seguinte forma

$$z = a + bi \tag{27}$$

Na expressão (27) *i* é a unidade imaginária e *a* e *b* são números reais, correspondentes à parte real e imaginária, respectivamente.

Assim, neste caso podem ser utilizados dois números reais para armazenar as partes reais e imaginária do número, sendo ainda necessário definir algoritmos para operações como a soma, a subtracção ou a multiplicação de números complexos.

4.2.4. PROGRAMAÇÃO ORIENTADA POR OBJECTOS

O paradigma da programação orientada por objectos consiste em estruturar o programa em classes e em dotar essas classes com um conjunto completo de operações, tornando explícitas características comuns às várias classes. Esta explicitação das características comuns entre classes é efectuada por intermédio de mecanismos de herança e polimorfismo.

No caso de não se estabelecerem características comuns entre classes de objectos o paradigma degenera na abstracção de dados.

A programação orientada por objectos apresenta uma metodologia inovadora de estruturação de programas, aproximando-se das actividades humanas. Por exemplo, quando uma fábrica pretende produzir em série um determinado produto não necessita fabricar todos os seus componentes, uma vez que possivelmente alguns destes já se encontram disponíveis no mercado. Também no caso da programação orientada por objectos é possível reaproveitar classes de objectos já existentes.

A terminologia adoptada quando se recorre à utilização de objectos é também muito característica, referindo-se alguns exemplos [9]:

- um objecto definido pelo programador é referido como uma instância de uma classe;
- os membros-função de uma classe são referidos como métodos dessa classe;
- os membros-dados de uma classe são referidos como atributos dessa classe;
- invocar um método de uma classe é referido como sendo o envio de uma mensagem a um objecto dessa classe. O objecto sobre o qual se invoca um método é referido como receptor da mensagem.

4.3. CLASSES DE OBJECTOS

As classes são agregados de membros de dados (atributos) e membros correspondentes a procedimentos ou funções que permitem interagir com o objecto a declarar (métodos).

Na definição das classes, interessa ter em consideração dois factores:

- as condições iniciais a impor para os objectos a criar;
- os métodos ou operações que são necessárias para interactuar com esses objectos.

A criação de um objecto e as condições a impor são satisfeitas por intermédio de métodos específicos designados construtores. Os construtores permitem executar operações de inicialização, como reservar

memória para o objecto ou estabelecer valores iniciais para os atributos. Por vezes pode igualmente ser necessário declarar um método complementar do construtor que permite "destruir" o objecto (método destrutor). As operações executadas por um método destrutor podem ser, por exemplo, libertar memória reservada para o objecto ou enviar uma mensagem a outro objecto que indique que este foi destruído.

4.3.1. DERIVAÇÃO DE CLASSES

Ao definir um conjunto de classes adequadas a implementar uma dada aplicação é desejável encontrar classes com propriedades comuns, de modo que a definição de algumas dessas classes possa ser feita com herança de atributos e métodos de outras classes genéricas.

A classe herdeira denomina-se classe derivada. A classe da qual a nova classe herda membros denomina-se classe base [10]. Uma classe pode derivar de uma ou mais classes base.

A classe derivada normalmente adiciona novos atributos e métodos à classe base da qual é derivada, sendo por isso mais rica.

4.3.2. POLIMORFISMO

Uma acção diz-se polimórfica se for executada de diferentes formas dependendo do contexto em que é invocada. No âmbito da programação por objectos, o polimorfismo traduz-se na possibilidade de redefinir nas classes derivadas os métodos declarados na classe base, com o mesmo nome e parâmetros, de modo que quando postos em execução sejam invocadas funções em concordância com o tipo de objecto. Para que tal seja possível é necessário definir os métodos da classe base como virtuais.

4.4. ESTRUTURAS DE DADOS

É frequente a utilização em programação de variáveis correspondentes a números, caracteres, etc. Estes tipos de variáveis possuem a vantagem de permitir ao programador abstrair-se da forma como são armazenados e manipulados pelo computador, i.e., em grupos de bits (cada um destes bits pode apresentar somente um de dois valores, 0 ou 1). Esta forma de abstracção pode ser generalizada para definir outros tipos de dados compostos por variáveis do tipo atrás referido, como por exemplo, vectores e matrizes de números. Nas linguagens de programação mais modernas é possível levar este tipo de abstracção ainda mais longe e definir tipos de dados especificamente ajustados para manipular a informação do programa a conceber, podendo assim construir-se estruturas com alguma complexidade, designadas estruturas de dados.

4.4.1. LISTAS LIGADAS

Os vectores são tipos de dados muito utilizados, apesar de possuírem as seguintes limitações:

- o tamanho máximo de um vector não pode aumentar depois de ele ter sido iniciado em memória;
- para inserir ou eliminar itens de um vector pode ser necessário deslocar vários outros itens do mesmo vector, para que se mantenha o seu tipo de organização.

Estas limitações podem ser ultrapassadas recorrendo a estruturas de dados designadas listas ligadas. Embora existam várias maneiras de implementar listas ligadas, pode referir-se que uma lista ligada é essencialmente um conjunto de objectos, designados nós, que se encontram ordenados sequencialmente (ver a Figura 10). Cada um dos nós contém um determinado tipo de dados e ainda uma variável do tipo apontador, que permite referenciar o nó que lhe sucede na lista.

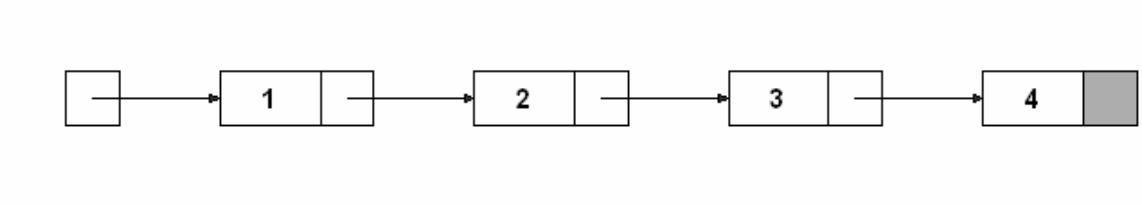


Fig.10 – Representação esquemática de uma lista ligada com quatro nós

Numa estrutura deste tipo é possível adicionar e remover facilmente nós ou sequências de nós simplesmente redireccionando apontadores em alguns dos nós da cadeia, tal como se indica nas Figuras 11 e 12.

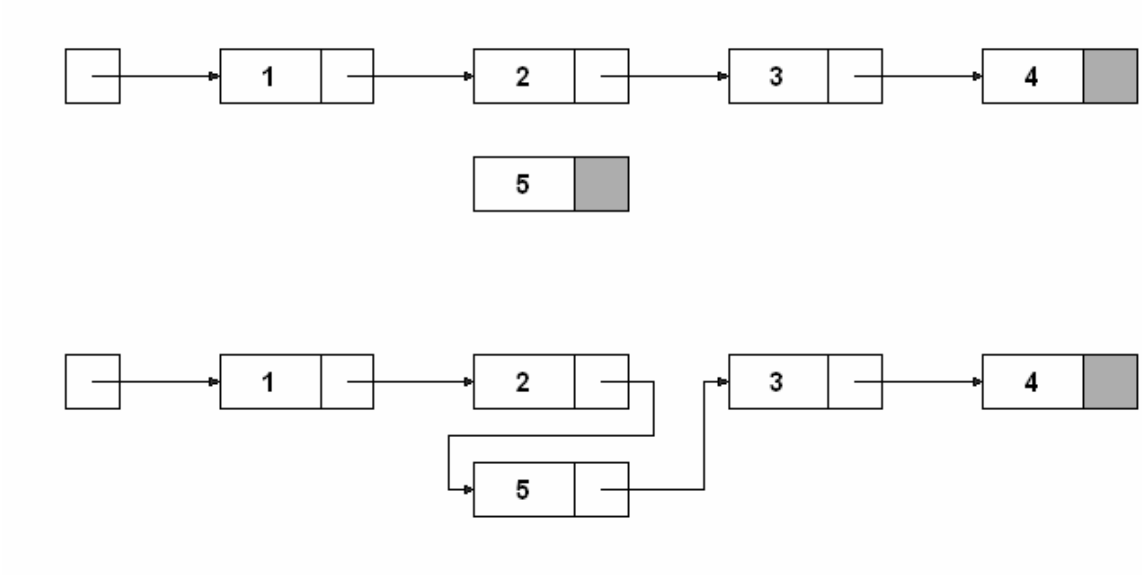


Fig.11 – Inserção do nó 5 entre os nós 2 e 3 de uma lista ligada

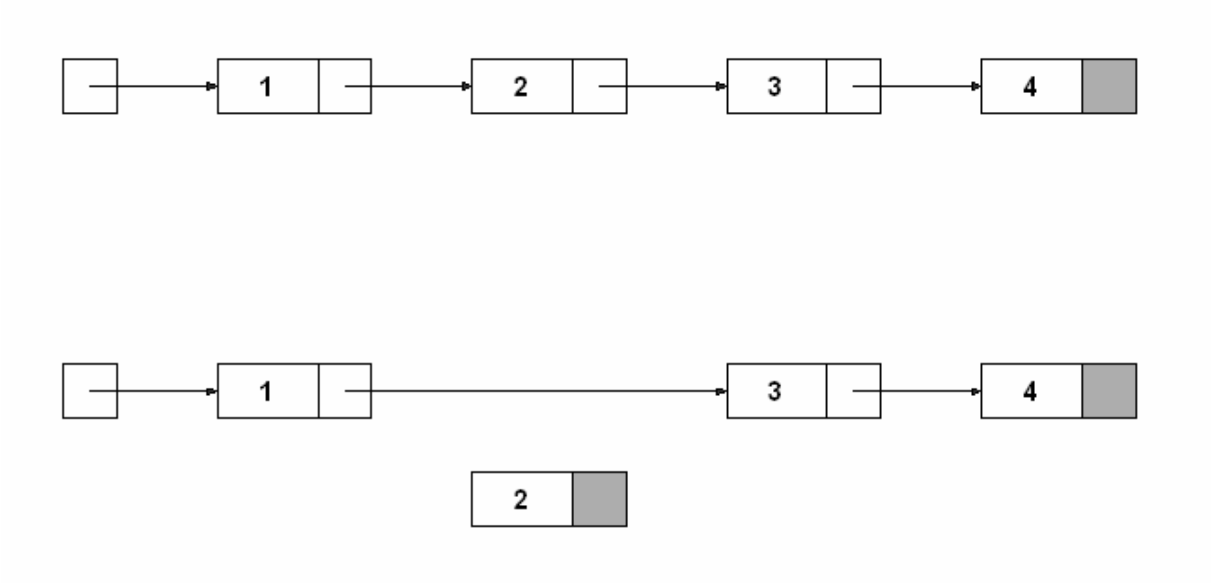


Fig.12 – Remoção do nó 2 de uma lista ligada

Se cada nó possui somente um vínculo para o seu sucessor, tal como no exemplo referido, tem-se uma lista simplesmente ligada. Existem no entanto outros tipos de listas ligadas, como por exemplo:

- listas duplamente ligadas;
- listas circulares;
- listas de salto.

As listas ligadas apresentam grandes vantagens em relação aos vectores, uma vez que facilmente permitem adicionar e remover dados, mantendo a coerência da estrutura. No entanto os seus nós de dados só podem ser acedidos de uma forma sequencial, aspecto que constitui a sua principal desvantagem.

5

CONCEPÇÃO DO PROGRAMA CSANALYSIS

5.1. INTRODUÇÃO

Neste capítulo faz-se referência a questões relacionadas com a concepção do programa CSAnalysis. Para além de incluir o modelo numérico descrito no Capítulo 3, este programa apresenta um certo nível de complexidade, inerente ao modo de interacção do utilizador com o núcleo de cálculo e à necessidade de gerir diversas estruturas de dados. São ainda referidos alguns detalhes referentes a algoritmos do núcleo de cálculo que estão relacionados com o modelo numérico adoptado.

É no entanto necessário referir que, dado o nível de complexidade da aplicação aqui descrita, seria demasiado extenso abordar todos os detalhes do programa, optando-se por referir apenas os aspectos conceptuais em que se baseia o extenso código que foi desenvolvido. A obtenção de um esclarecimento mais aprofundado requer a leitura do código do programa.

5.2. CARACTERÍSTICAS DO PROGRAMA

Os requisitos iniciais associados ao desenvolvimento do programa CSAnalysis referem a necessidade de o programa de cálculo ser suficientemente robusto para permitir analisar a maior generalidade possível dos casos de flexão composta desviada. Mas simultaneamente deve também ser aplicável à resolução dos casos mais correntes de forma suficientemente prática. Assim, cedo se tornou claro que para além de um núcleo de cálculo robusto o programa deve ainda possuir as seguintes características:

- permitir abrir e guardar as secções em ficheiros;
- permitir criar e utilizar materiais armazenados em ficheiros;
- apresentar uma arquitectura baseada num programa de CAD, permitindo a introdução e modificação de dados segundo uma ordem o mais arbitrária possível, quer pelo rato, quer pelo teclado;
- mostrar uma grelha de apoio à introdução de coordenadas e mecanismos de controlo das condições de visualização;
- permitir o cálculo e a representação das soluções correspondentes ao Estado Limite Último;
- permitir o cálculo da posição do centro de gravidade da secção considerando apenas o material base, bem como outras características geométricas da secção;
-

- dispor de um assistente para a criação rápida das secções mais correntes (circular, rectangular ou em T).

Trata-se assim de uma aplicação fácil de utilizar, baseada numa arquitectura de programa de desenho assistido por computador.

5.3. ESTRUTURA GLOBAL DO PROGRAMA

As características pretendidas para o programa CSAnalysis levaram a que fosse preferível estruturar a aplicação recorrendo à programação orientada por objectos, uma vez que não se pretende um modo de funcionamento essencialmente do tipo sequencial mas sim relacional entre as diferentes partes que constituem a aplicação. Apesar disso, o núcleo de cálculo encontra-se definido segundo um paradigma essencialmente procedimental, uma vez que executa operações de cálculo sequenciais através de uma composição de procedimentos.

O programa encontra-se por esse motivo dividido em duas partes fundamentais:

- núcleo de cálculo essencialmente procedimental, não deixando no entanto de se recorrer à programação de alguns objectos para gerir as estruturas de dados envolvidas no cálculo e em certos casos pontuais;
- interface com o utilizador essencialmente estruturada segundo uma lógica de objectos.

5.4. ESTRUTURAS DE DADOS

As estruturas de dados existentes no programa foram implementadas como objectos, tirando-se assim vantagem das características do paradigma de programação adoptado, como o encapsulamento e a possibilidade de derivação dos tipos de dados (ver o Capítulo 4). Estas estruturas podem agrupar-se segundo três objectivos:

a) caracterização das curvas tensão-extensão dos materiais;

b) caracterização da secção a calcular;

c) armazenamento temporário de informação associada ao processamento.

As estruturas do tipo a) e b) referido encontram-se dotadas de métodos específicos para guardar e retirar dados de ficheiros (métodos *Save*() e *Load*()). Nas estruturas do tipo c) não existem estes métodos uma vez que se trata de informação temporária.

Os objectos correspondentes às curvas tensão-extensão possuem ainda métodos para avaliar facilmente a tensão em função da extensão, dada uma determinada tensão de pré-esforço inicial (método *Eval*()).

Já a caracterização da secção é feita por intermédio de duas estruturas de dados de topo, que representam em paralelo a mesma informação, mas organizada de dois modos distintos, a que correspondem as seguintes classes de objectos:

a) *CSection* – estrutura de dados para manipulação e visualização da secção;

b) *CCalcSection* – estrutura de dados da secção tendo em vista a realização de cálculos.

A estrutura *CSection*, representada no diagrama de classes do Anexo A1, é particularmente adequada para a manipulação dos dados da secção, uma vez que facilmente permite inserir e retirar vértices (*Vertex*), arestas (*Edge*) e pontos de colocação de armadura (*Rebar*) de forma arbitrária, mantendo a coerência da estrutura de dados. Esta característica deve-se ao facto de terem sido utilizadas classes contentoras do tipo "Lista Ligada".

A estrutura *CSection* apresenta no entanto o inconveniente de apenas permitir um acesso sequencial aos dados. Para permitir um acesso não sequencial, importante para efeitos de cálculo, houve a necessidade de criar a estrutura *CCalcSection* baseada em vectores de dados. Esta segunda estrutura, *CCalcSection*, é reconstruída com base na estrutura *CSection* sempre que é necessário proceder a um cálculo e os dados tenham sido alterados. Com este objectivo é utilizado o método *Rebuild*().

5.5. INTERFACE COM O UTILIZADOR

Como já foi referido, a interface com o utilizador foi escrita segundo uma lógica de objectos. A estrutura essencial de funcionamento encontra-se esquematizada no Anexo A2. As principais classes de objectos utilizadas são as seguintes:

- estados de edição e funcionamento da aplicação;
- sistema de coordenadas;
- janela de edição e visualização da secção;
- documento representativo da secção;
- outras classes (estruturas de dados, desenho, controlo de caixas de diálogo, etc.).

O programa CSAnalysis apresenta características especiais na medida em que funciona sob o controlo de um sistema operativo baseado em eventos. Como tal, este tipo de programa não é executado de forma totalmente "autónoma" como os programas mais clássicos. Essencialmente, o sistema operativo envia informação ao programa sob a forma de "mensagens" a que correspondem "eventos", como por exemplo: o utilizador alterou a posição do ponteiro do rato, ou uma janela sobrepôs-se à janela do programa. Nestas circunstâncias, o programa deve responder de forma adequada às "mensagens" que lhe são enviadas. Daí a ideia de criar objectos correspondentes a diferentes estados de edição e funcionamento da aplicação contendo métodos específicos para gerir as "mensagens" enviadas pelo sistema, conforme o utilizador esteja por exemplo a introduzir vértices ou a mudar as condições de visualização da secção. Estes objectos tiram partido do polimorfismo e da derivação de classes, próprios da programação orientada por objectos, uma vez que derivam todos de um mesmo objecto que corresponde à resposta predefinida dada pelo programa para as mensagens que lhe são enviadas.

O sistema de coordenadas que está a ser utilizado é também definido por intermédio de um objecto apropriado. Este objecto encapsula dados relativos às coordenadas do modelo real e às coordenadas de representação em ecrã (pixeis), correspondentes ao rectângulo que está a ser visualizado em determinado momento. No objecto correspondente ao sistema de coordenadas foram definidos métodos apropriados para:

- estabelecer enquadramentos de imagem (algoritmo de *mapping*);
- estabelecer relações geométricas correspondentes a pontos ou vectores entre os dois sistemas de coordenadas;
- alterar o rectângulo de visualização.

A janela de edição e visualização permite fazer a representação da secção em que se está a trabalhar. O objecto correspondente ao documento serve essencialmente para gerir o ficheiro e a informação correspondente à secção. Existem ainda outras classes para gerir caixas de diálogo, efectuar operações de desenho na janela de visualização ou controlar estruturas de dados (ver Secção 5.4).

5.6. NÚCLEO DE CÁLCULO

O núcleo de cálculo constitui a implementação do modelo numérico adoptado (ver o Capítulo 3). Trata-se de uma parte do programa desenvolvida numa lógica de programação essencialmente procedimental, uma vez que consiste numa execução sequencial de cálculos obtida por composição de procedimentos, não deixando no entanto de se recorrer a alguns objectos para as estruturas de dados e a um objecto de cálculo (*CSlice*).

O núcleo de cálculo é essencialmente constituído pelos seguintes módulos:

- *ConcreteDiseq* – calcula os esforços resultantes do funcionamento mecânico do material base (ver o código no Anexo A4);
- *Correction* – calcula as correcções a fazer à solução corrente do problema;
- *Disequilibrium* – determina os desequilíbrios de esforços na secção (ΔN, ΔM_x e ΔM_y);
- *MainCalcProcedure* – procedimento principal que controla o núcleo de cálculo, retornando as soluções do problema de flexão composta desviada (ver o código no Anexo A4);
- *Rotate* – executa a transformação linear das coordenadas de um determinado conjunto de pontos, correspondendo à rotação do sistema de eixos (ver o código no Anexo A4);
- *SolGenerator* – consiste num gerador de soluções iniciais que se encontrem próximas das soluções correctas. Estas soluções aproximadas são necessárias para iniciar o processo iterativo de resolução;
- *Solve* – calcula as soluções do problema principal, recorrendo a um processo iterativo;
- *SteelDiseq* – calcula os esforços resultantes do funcionamento mecânico das armaduras;
- *StrainField* – define o campo de extensões, tendo em conta a geometria da secção, a inclinação do eixo neutro (β) e o parâmetro correspondente ao tipo de ruptura (D) (ver o código no Anexo A4).

A interdependência entre os referidos módulos encontra-se esquematizada no diagrama que se encontra no Anexo A3.

5.6.1. CÁLCULO DOS ESFORÇOS RESULTANTES NO MATERIAL BASE

A determinação dos esforços resultantes no material base é efectuada de acordo com a sequência de cálculo referida na Secção 3.4. Nesta secção é descrito o algoritmo utilizado para efectuar a divisão da secção em bandas. Para cada banda são efectuados os procedimentos que se seguem, recorrendo para tal ao objecto auxiliar *CSlice*.

1) É calculada a ordenada y correspondente ao eixo médio da banda.
2) São calculados os pontos de possível intercepção entre a linha correspondente ao eixo médio da banda e cada uma das semi-rectas correspondentes aos contornos da secção.

2.1) Cálculo do ponto de intercepção entre a linha média da banda e a recta que contem a semi-recta a interceptar, no caso da semi-recta não ser paralela.

2.2) Se o ponto de intercepção se encontra entre os dois pontos extremos da semi-recta, então admite-se que se trata de um ponto a considerar na definição da banda. Se tal se verificar é "enviado" o valor da abcissa do ponto obtido ao objecto *CSlice* (método *Insert*()).

3) As abcissas dos pontos obtidos em 2), e armazenadas no objecto *CSlice*, são ordenadas por ordem crescente (método *Sort*()), de acordo com o indicado na Figura 13.

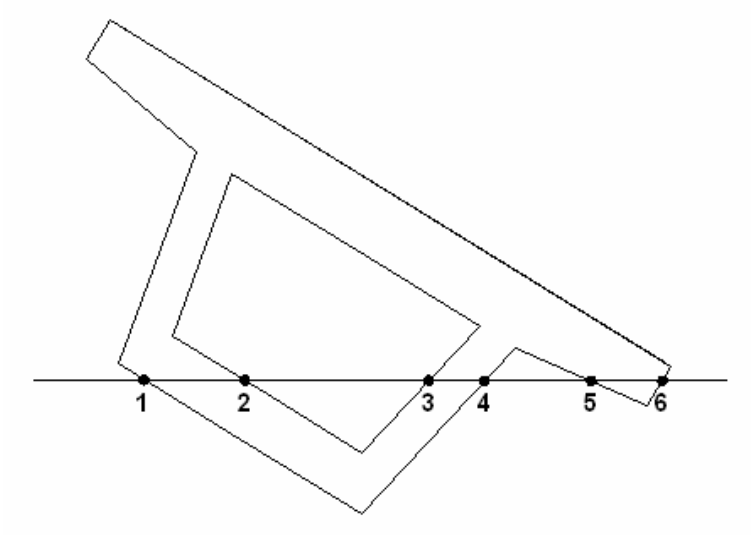


Fig.13 – Conjunto de pontos de intercepção, depois de ordenados

4) Tendo em consideração que os pares de abcissas que vão de um ponto ímpar para um ponto par correspondem a segmentos internos da secção, é possível determinar o comprimento e os momentos de primeira ordem relativamente aos eixos horizontal e vertical, por unidade de largura da banda (métodos *Length*(), *Momentum*() e *Momentum*2()). Esses valores são em seguida multiplicados pela largura da banda e pela tensão instalada, que é admitida constante, obtendo-se a força e os momentos correspondentes a essa banda.

5.6.2. CÁLCULO DOS ESFORÇOS RESULTANTES NAS ARMADURAS

O processo de cálculo dos esforços resultantes nas armaduras é em seguida apresentado.

Para cada ponto central de uma armadura:

1) é calculado o valor da extensão no ponto central da armadura;

2) é calculado o valor de tensão correspondente à extensão obtida no ponto anterior e atendendo à curva tensão-extensão do material utilizado;

3) a força F, o momento M_x e o momento M_y são obtidos com as seguintes expressões

$$F = \sigma A \tag{28}$$

$$M_x = F\, y \tag{29}$$

$$M_y = F\, x \tag{30}$$

Sendo σ e A a tensão e a área correspondentes à armadura e (x,y) as coordenadas do seu ponto central.

Os valores de F, M_x e M_y obtidos em 3) vão sendo sucessivamente adicionados, obtendo-se no final do processo os esforços resultantes da contribuição de todas as armaduras.

5.6.3. GERADOR DE SOLUÇÕES INICIAIS

Como foi referido no Capítulo 3, a determinação de soluções para o problema da flexão composta desviada por intermédio da resolução do sistema (6) é feita recorrendo a um processo iterativo (método de Newton-Raphson). Uma vez que se executa um processo iterativo é necessária uma solução inicial aproximada para o sistema. Para tal existe um procedimento que gera soluções inicias (função *SolGenerator*()), permitindo obter soluções aproximadas e assim iniciar o processo iterativo.

O gerador de soluções foi concebido tendo em atenção o facto de o ângulo de inclinação do eixo neutro, β, poder apenas variar entre 0 e 2π e a variável D poder variar entre 1 e 4. Assim, o gerador de soluções efectua o cálculo do desequilíbrio de esforços sobre uma malha de pontos definida neste domínio de soluções (ver Figura 14), sendo a terceira incógnita determinada em cada ponto de forma a minimizar a soma dos quadrados dos desequilíbrios obtidos.

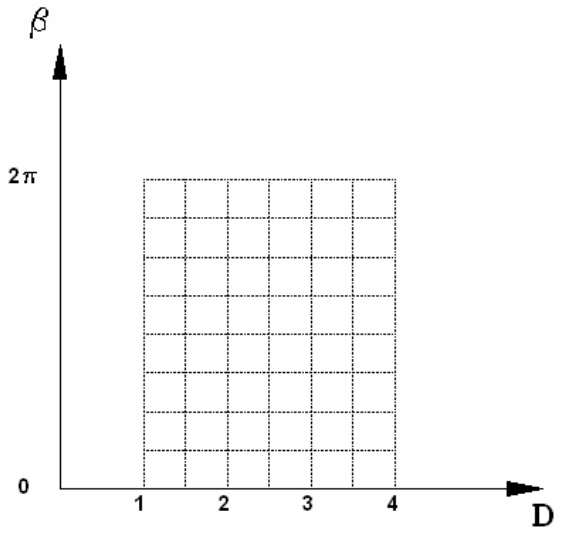


Fig.14 – Malha de pontos utilizada pelo gerador de soluções iniciais

De todos os pontos calculados são seleccionados para soluções iniciais aqueles aos quais correspondem menores valores da soma dos quadrados dos desequilíbrios de esforços.

5.6.4. RESOLUÇÃO DO SISTEMA DE EQUAÇÕES

A resolução do sistema de equações é feita de acordo com o processo iterativo já referido na secção 3.7. Este procedimento encontra-se implementado por intermédio da função *Solve*() do núcleo de

cálculo, recebendo num dos parâmetros de entrada soluções aproximadas fornecidas pelo gerador de soluções iniciais. São efectuadas sucessivas iterações dos seguintes passos:

1) cálculo dos desequilíbrios da solução corrente;
2) cálculo da soma dos quadrados dos desequilíbrios da solução corrente. Se esta soma for inferior a uma dada tolerância termina o ciclo sendo devolvida a solução;
3) se o número máximo de iterações foi ultrapassado, o processo é interrompido sem ser apresentada qualquer solução;
4) cálculo numérico das derivadas parciais da função que devolve os desequilíbrios de esforços, no ponto correspondente à solução corrente, recorrendo a diferenças finitas;
5) montagem da matriz Jacobiana da transformação referida em 4 e resolução do sistema (25) obtendo as correcções a fazer à solução corrente;
6) soma à solução corrente da correcção determinada no passo 5 depois de multiplicada por um parâmetro α (parâmetro de *line-search*), que assume valores entre 0 e 1, sendo determinado de forma a tornar a solução mais próxima da solução exacta;
7) incremento do número de iterações e retorno ao passo 1.

6

UTILIZAÇÃO DO PROGRAMA CSANALYSIS

6.1. INTRODUÇÃO

Neste capítulo é apresentado o programa de computador que efectivamente foi realizado no âmbito deste projecto. Refere-se também o seu modo de funcionamento e as operações básicas que é possível realizar.

6.2. APRESENTAÇÃO GERAL DO PROGRAMA

Como já foi referido o programa CSAnalysis é uma aplicação dotada de uma interface gráfica fácil de utilizar. O programa incorpora caixas de diálogo, barras de ferramentas e menus por intermédio dos quais é possível realizar tarefas. O aspecto geral do programa encontra-se na Figura 15.

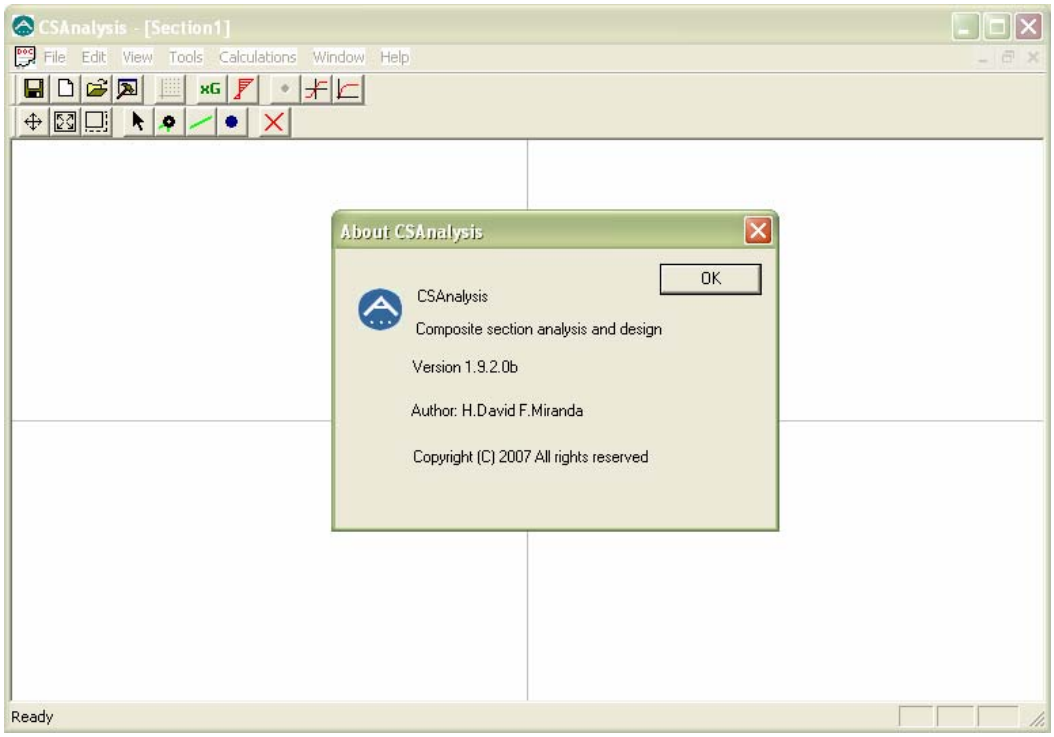


Fig.15 – Interface do programa CSAnalysis

A barra de ferramentas contém botões de acesso rápido às funcionalidades mais úteis da aplicação. No entanto todas estas funcionalidades podem igualmente ser acedidas por intermédio do menu existente. O utilizador pode inclusive desactivar as barras de ferramentas.

6.3. ABRIR, CRIAR E GUARDAR SECÇÕES

Para abrir, criar e guardar secções num ficheiro já aberto ou guardá-las num novo ficheiro, deve aceder-se ao menu "File" (Figura 16) e escolher a respectiva opção "Open…", "New", "Save" ou "Save As…".

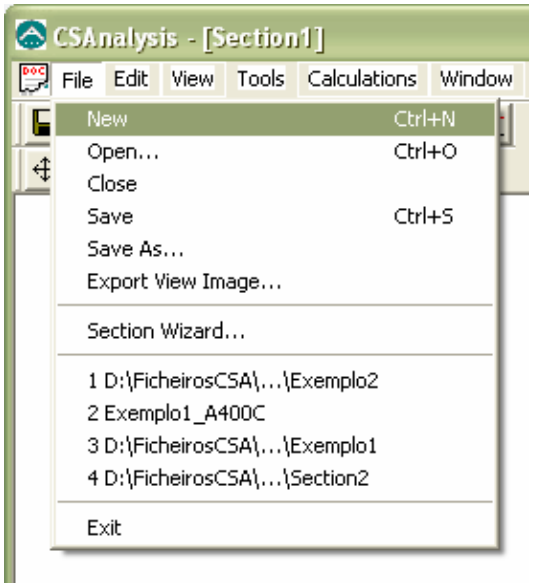


Fig.16 – Opções do menu *File*

No caso de se escolher abrir ou guardar a secção num novo ficheiro, é aberta uma caixa de diálogo onde se deve escolher o nome do ficheiro, tal como se mostra na Figura 17.

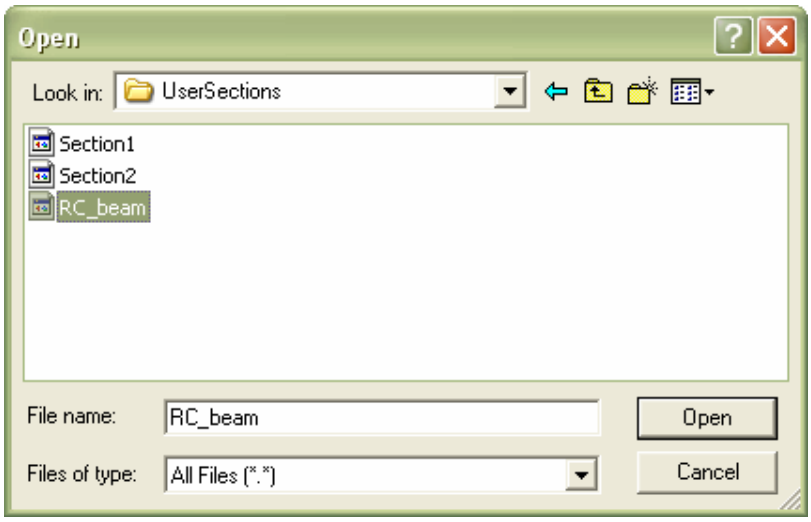


Fig.17 – Caixa de diálogo para escolha do ficheiro

6.4. ALTERAR OS MATERIAIS CONSTITUINTES DA SECÇÃO

Para escolher os materiais constituintes, bem como atribuir um nome para a secção e escrever alguns comentários relativos ao problema a resolver, deve escolher-se a opção do menu "Edit > Section properties", surgindo assim uma caixa de diálogo como a indicada na Figura 18.

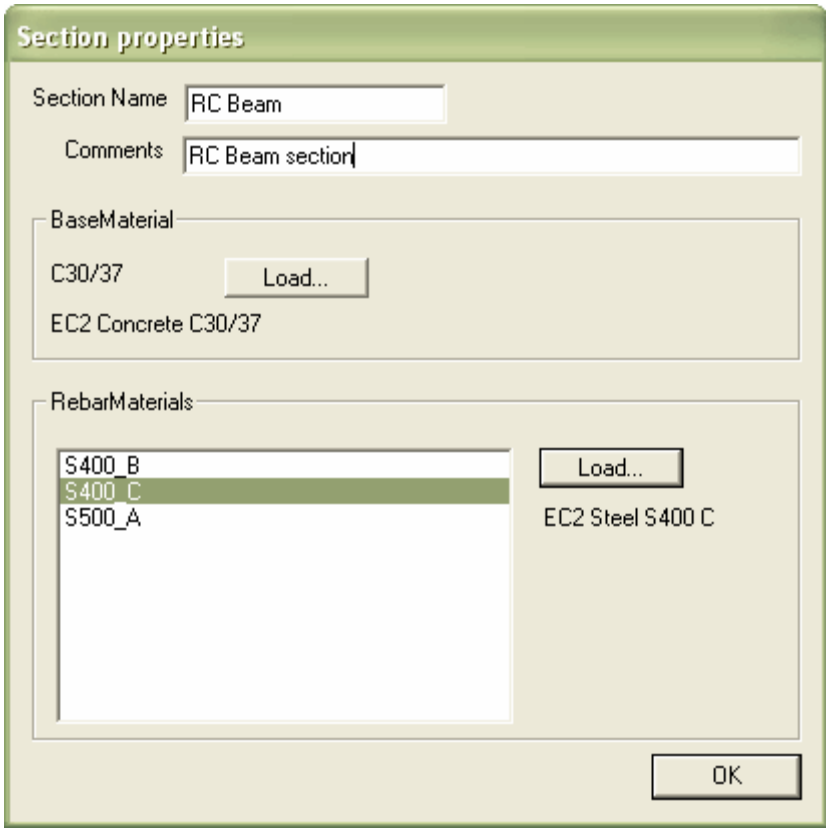


Fig.18 – Caixa de diálogo *Section Properties*

Para seleccionar um material existente em ficheiro, deve-se pressionar o respectivo botão "Load…", surgindo uma caixa de diálogo (como a já apresentada na Figura 18) em que se requer a selecção do ficheiro correspondente ao material. Apenas é possível seleccionar um material base para a secção, podendo no entanto ser seleccionados distintos materiais para as armaduras de reforço. Junto do nome do material são indicados os respectivos comentários.

6.5. DEFINIÇÃO DA GEOMETRIA DA SECÇÃO

A geometria de uma secção é definida por intermédio dos vértices e arestas que a limitam, bem como pelo posicionamento e características das armaduras, tal como se mostra na Figura 19.

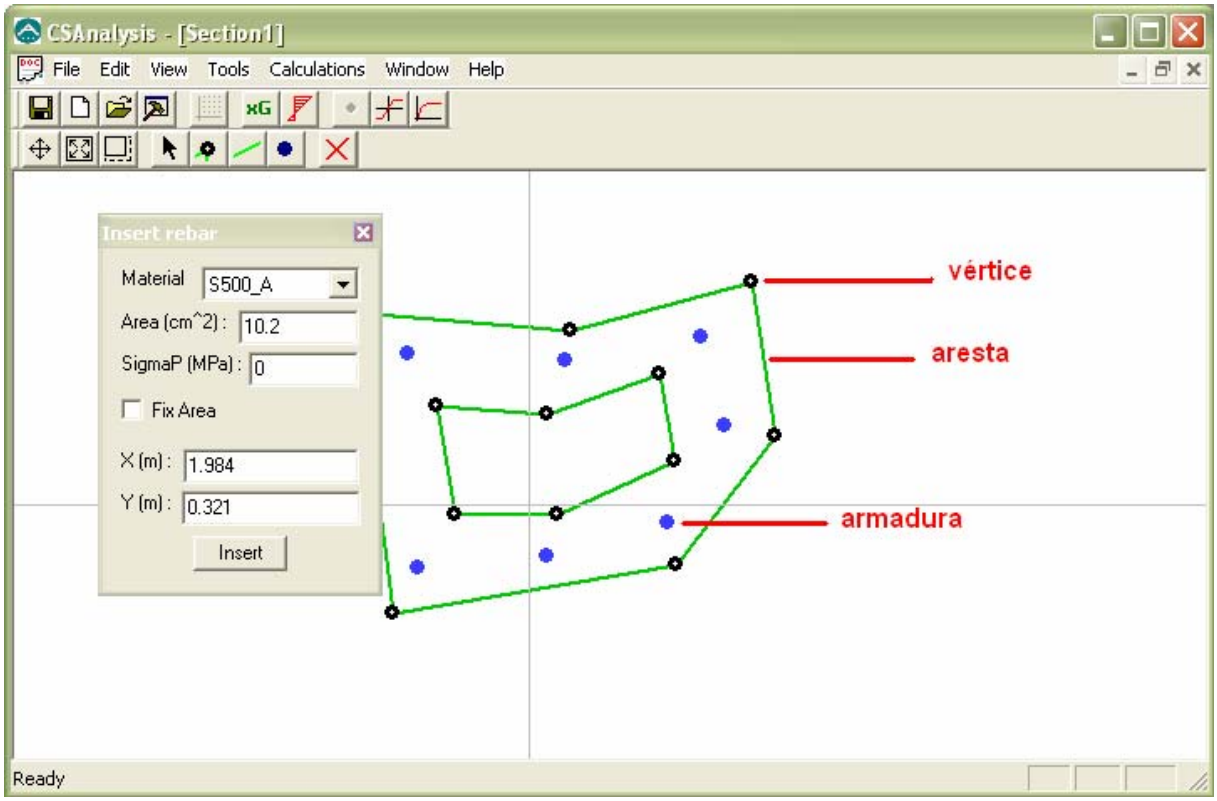


Fig.19 – Definição de uma secção genérica

Os vértices da secção devem ser inseridos recorrendo à opção do menu "Edit > Insert Vertex", podendo introduzir-se os vértices graficamente recorrendo ao rato ou inserir as suas coordenadas pelo teclado na correspondente caixa de diálogo. As armaduras devem ser definidas recorrendo à opção do menu "Edit > Insert Rebar". É necessário escolher o tipo de material, a área de armadura, a tensão de pré-esforço (no caso da armadura se encontrar pré-esforçada) e ainda indicar as coordenadas do ponto central do varão. Pode também definir-se que, no caso do dimensionamento, a área do varão de armadura se encontra fixa, sendo para tal necessário assinalar a opção "Fix Area" (ver Figura 19). A definição das coordenadas das armaduras pode ser feita tal como as dos vértices, i.e., graficamente recorrendo ao rato, ou introduzidas pelo teclado.

A introdução de arestas é feita recorrendo à opção "Edit > Insert Edge". Após activada a opção deve indicar-se através do rato quais os dois vértices extremos da aresta a definir.

Depois de já definido algum dos elementos da secção é possível proceder a modificações, escolhendo a opção "Edit > Select/Modify" e seleccionar vértices, arestas ou armaduras para posteriormente "arrastar" recorrendo ao rato ou então proceder a alterações pelo teclado, recorrendo a caixas de diálogo com se indica na Figura 20.

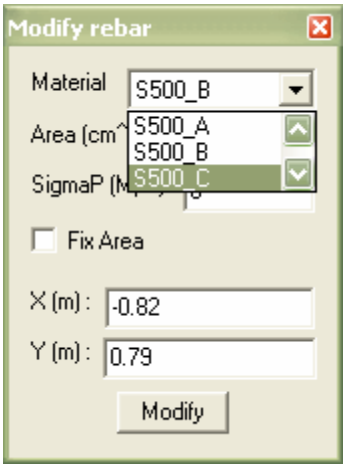


Fig.20 – Modificação das características de um varão de armadura

É também possível eliminar os elementos seleccionados recorrendo à opção "Edit > Delete Selected Object".

6.6. ALTERAR AS CONDIÇÕES DE VISUALIZAÇÃO

Para facilitar a definição da secção, é possível ajustar o rectângulo de visualização, recorrendo às seguintes opções do sub-menu "View".

- "Pan View" – Permite "arrastar" o rectângulo correspondente à janela de visualização.
- "Zoom Fit" – Define o rectângulo de visualização de forma que seja representado o desenho integral da secção, com a maior dimensão possível.
- "Zoom Box" – Permite seleccionar graficamente o rectângulo a visualizar (Figuras 21 e 22).

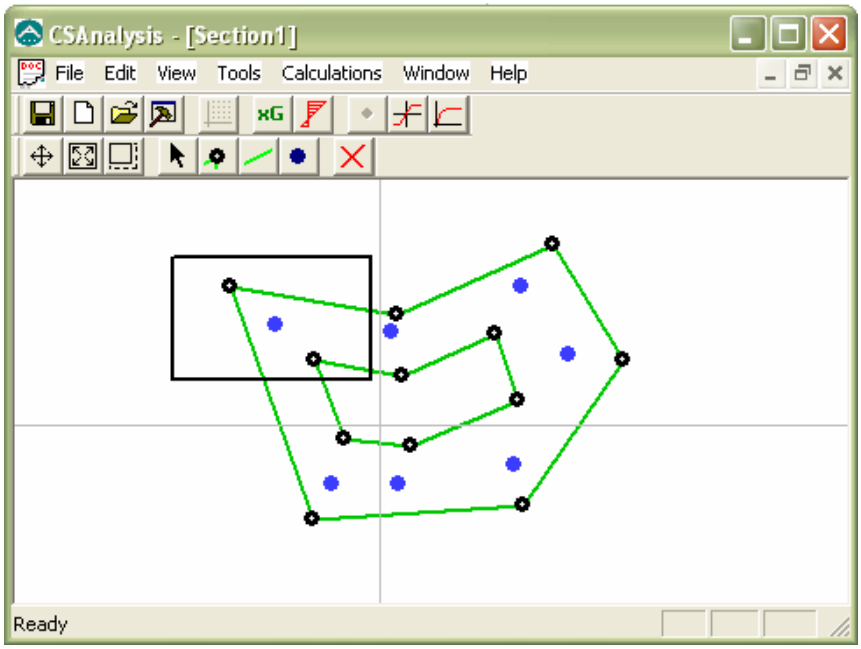


Fig.21 – Selecção do rectângulo a visualizar com a opção *Zoom Box*

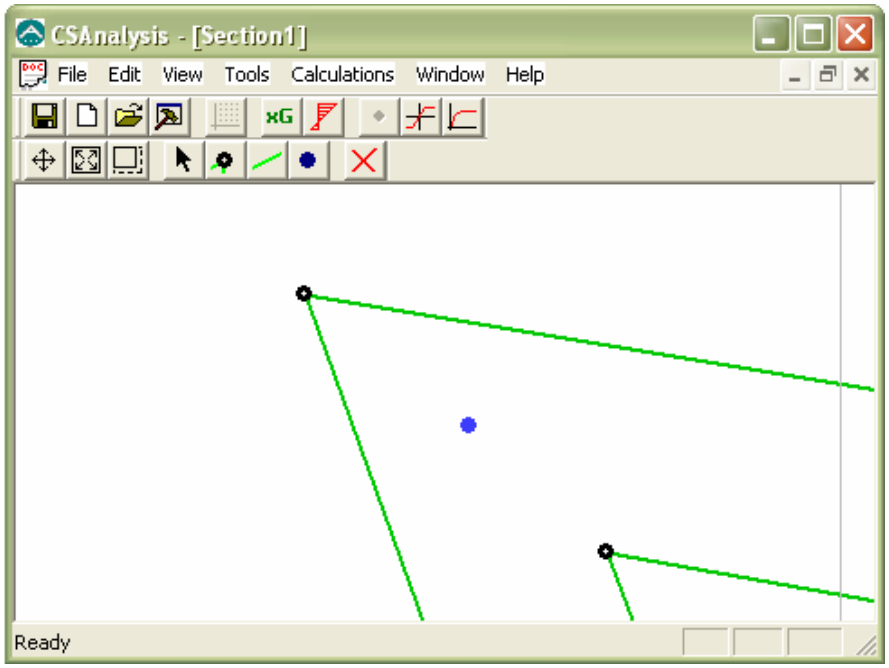


Fig.22 – Visualização do rectângulo escolhido recorrendo à opção *Zoom Box*

É também possível proceder ao aumento ou à diminuição da escala da representação por intermédio da rotação da roda do rato.

6.7. VISUALIZAÇÃO DO SISTEMA DE EIXOS E GRELHA DE APOIO

No sentido de facilitar a introdução de coordenadas pela via gráfica, é possível utilizar uma grelha de apoio recorrendo à opção "View > Grid/Axis", surgindo a caixa de diálogo representada na Figura 23.

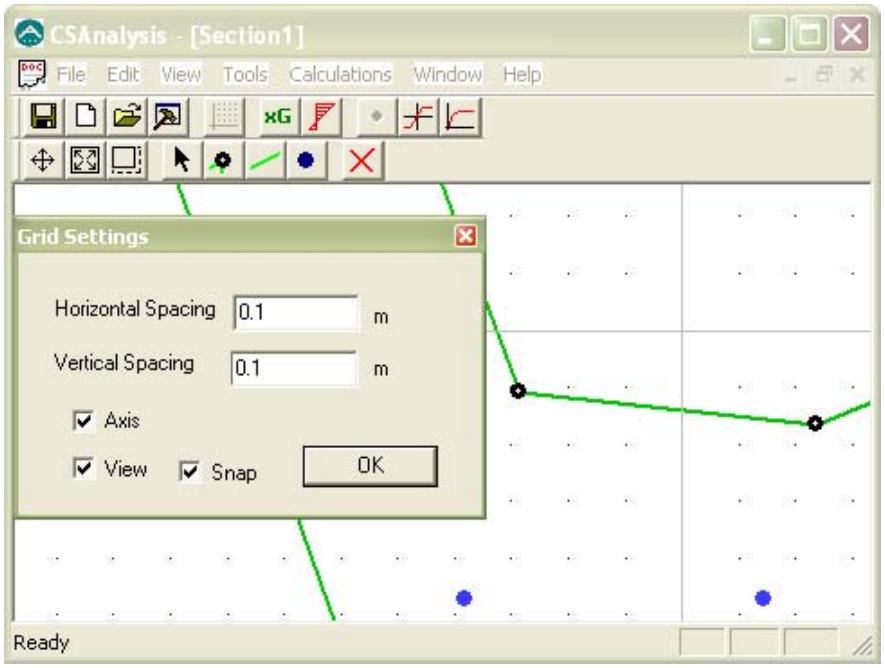


Fig.23 – Caixa de configuração da grelha e dos eixos

Na referida caixa de diálogo é possível configurar o espaçamento horizontal e vertical da grelha, bem como o facto de esta se encontrar ou não visível (opção "View"). É também possível optar pelo ajustamento dos pontos introduzidos a pontos da grelha durante a introdução de coordenadas (opção "Snap"). É igualmente possível definir se os eixos se encontram visíveis (opção "Axis").

6.8. CÁLCULO DAS PROPRIEDADES GEOMÉTRICAS DA SECÇÃO

Para calcular as propriedades geométricas da secção, como as coordenadas do centro de massa do material base ou os momentos estáticos e momentos de inércia, deve ser utilizada a opção "Calculations > Mass Properties…". As propriedades geométricas são exibidas numa caixa de diálogo, tal como representado da Figura 24.

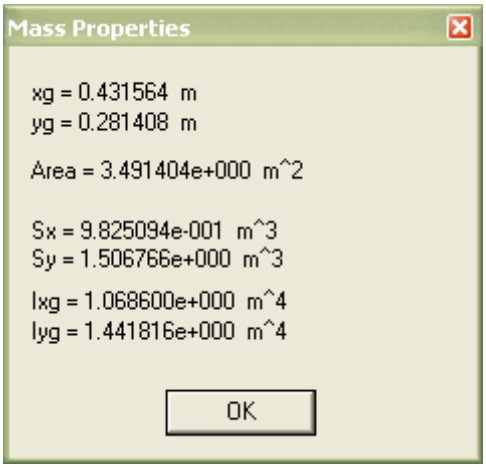


Fig.24 – Caixa de diálogo exibindo as propriedades geométricas da secção

Se a secção não for válida, por exemplo, se a envolvente for uma poligonal aberta, não é possível proceder ao cálculo, sendo exibida uma mensagem de erro.

6.9. CÁLCULO DA SECÇÃO EM ESTADO LIMITE ÚLTIMO

O cálculo da secção sujeita a flexão composta desviada em Estado Limite Último deve ser efectuado recorrendo à opção "Calculations > Calculate Section…". Ao seleccionar esta opção é exibida uma caixa de diálogo onde são pedidos os valores dos esforços a que a secção está sujeita (N, M_x e M_y) e qual a incógnita do problema (ver a Figura 25).

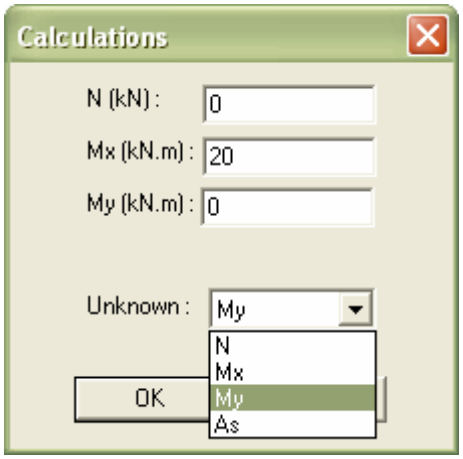


Fig.25 – Caixa de diálogo para introdução dos esforços e escolha da incógnita do problema

Se A_s for seleccionado para incógnita, o programa determina o valor mínimo pelo qual se devem multiplicar as áreas de armadura correspondentes aos varões que não foram definidos com áreas fixas, de forma a verificar segurança para os esforços dados (problema de dimensionamento). Se um dos esforços for seleccionado como incógnita do problema, o programa determina, para as áreas de armadura que foram fixadas, quais os valores do esforço desconhecido que condicionam o início da ruptura. Neste último caso, o valor que se introduz na caixa de diálogo para o esforço desconhecido é irrelevante.

As soluções determinadas são exibidas numa caixa de diálogo semelhante à representada na Figura 26.

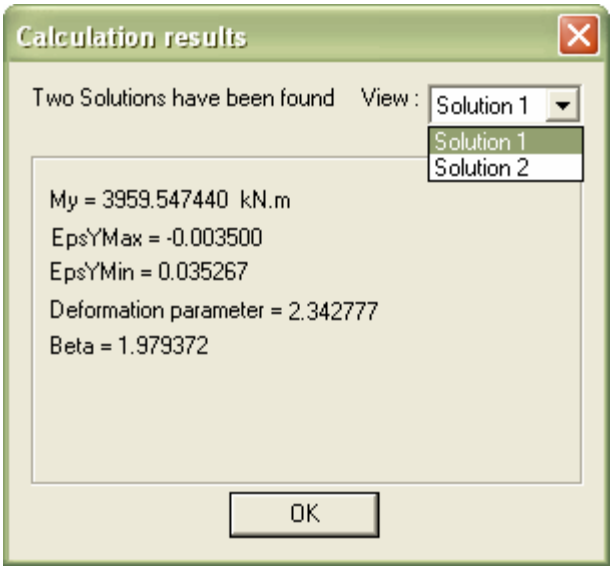


Fig.26 – Caixa de diálogo com as soluções obtidas

No caso de ter sido obtida mais do que uma solução, o utilizador pode escolher aquela à qual vão corresponder os diagramas de extensão e de tensão no material base (ver a Figura 27).

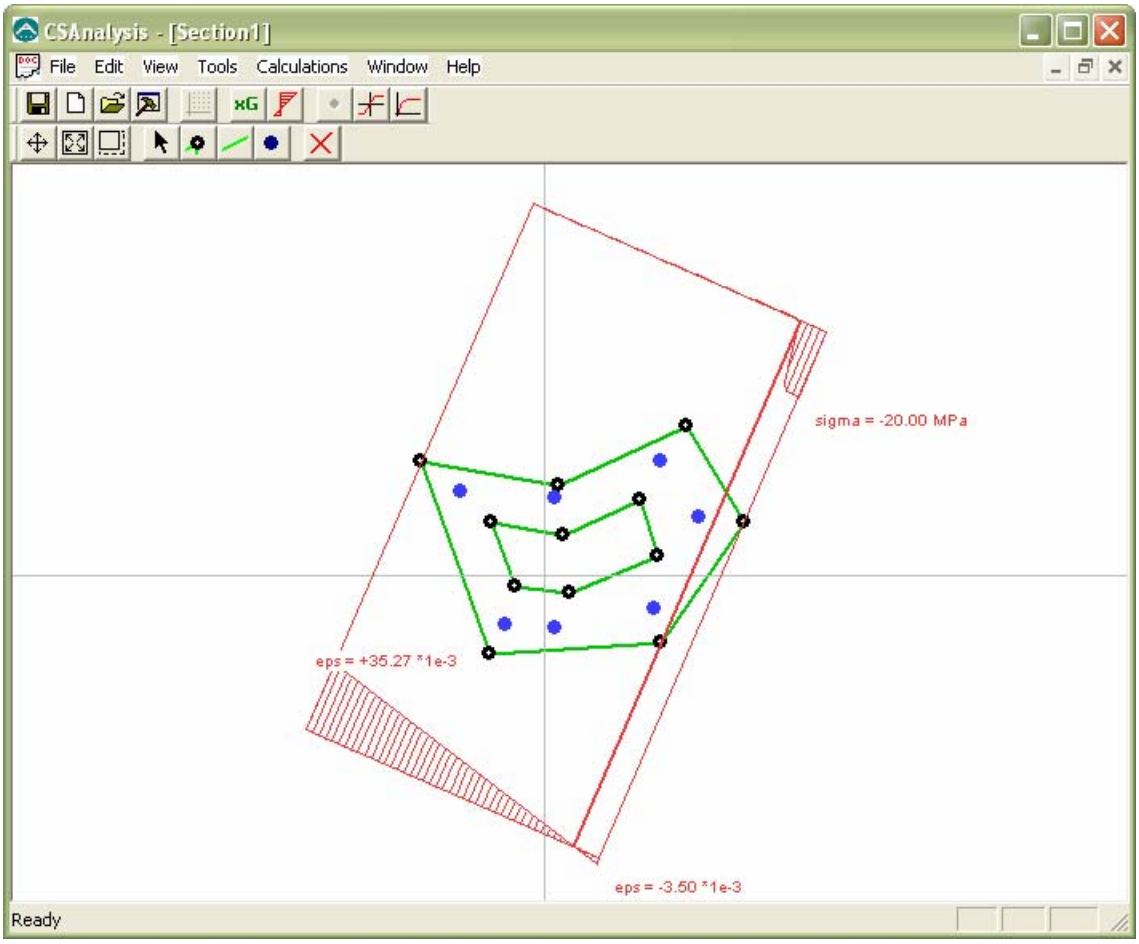


Fig.27 – Representação dos diagramas de extensão e de tensão no material base

Se a secção não for válida, por exemplo, se a envolvente for uma poligonal aberta, não é possível proceder ao cálculo, sendo exibida uma mensagem de erro.

6.10. ASSISTENTE PARA MODELAÇÃO DE SECÇÕES CIRCULARES, RECTANGULARES E EM T

O assistente de modelação de secções serve para facilitar a definição daquelas que são mais correntemente utilizadas, evitando-se assim o procedimento de introdução directa de vértices, arestas e armaduras. Este assistente permite definir rapidamente secções circulares, rectangulares e em T. Para utilizar o assistente deve escolher-se a opção "File > Section Wizard", devendo depois escolher-se qual o tipo de secção a criar (Figura 28).

Fig.28 – Caixa de diálogo para selecção do tipo de secção

Dependendo do tipo de secção escolhida (Figuras 29, 30 e 31) são requeridos alguns parâmetros definidores da secção, por exemplo, a espessura de recobrimento e o número e diâmetro dos varões a utilizar.

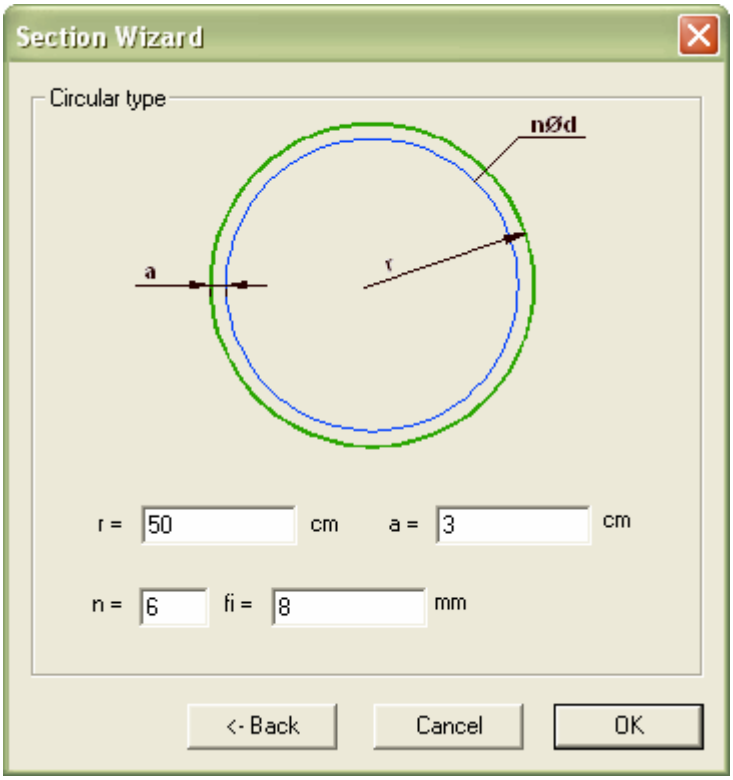


Fig.29 – Caixa de diálogo para introdução dos parâmetros definidores de uma secção circular

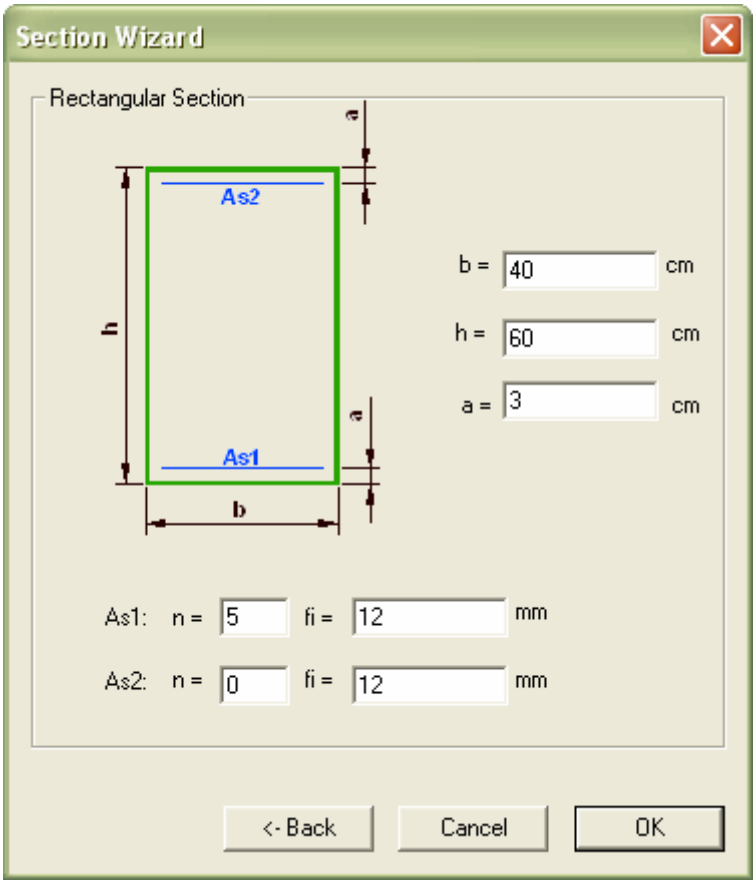


Fig.30 – Caixa de diálogo para introdução dos parâmetros definidores de uma secção rectangular

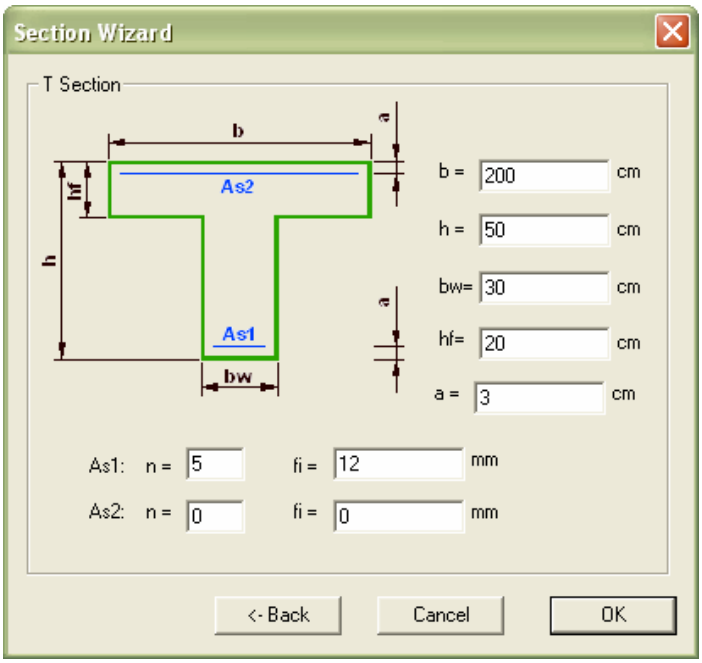


Fig.31 – Caixa de diálogo para introdução dos parâmetros definidores de uma secção em T

Após a introdução dos dados a secção pretendida é gerada e fica pronta a ser calculada. No entanto é ainda possível proceder a modificações, tal como se faria em qualquer outra secção, podendo assim por exemplo colocar-se armaduras pré-esforçadas ou introduzir aberturas na secção não previstas pelo assistente.

6.11. ASSISTENTE DE MATERIAIS BASE E ASSISTENTE DE MATERIAIS PARA ARMADURA

Encontram-se preparados ficheiros relativos a aços e betões em conformidade com as idealizações propostas pelos regulamentos REBAP e EC2. No entanto o programa permite que se utilizem outras idealizações de curvas tensão-extensão ou outros materiais definidos pelo utilizador, desde que existam para tal os ficheiros que caracterizam essas curvas. Assim, para facilitar a criação de ficheiros correspondentes aos materiais e as suas curvas tensão-extensão, são disponibilizados pelo programa um assistente de materiais base e um assistente de materiais para armadura.

O assistente de materiais base pode ser utilizado seleccionando "Tools > Base Material Creator…" (Figura 32).

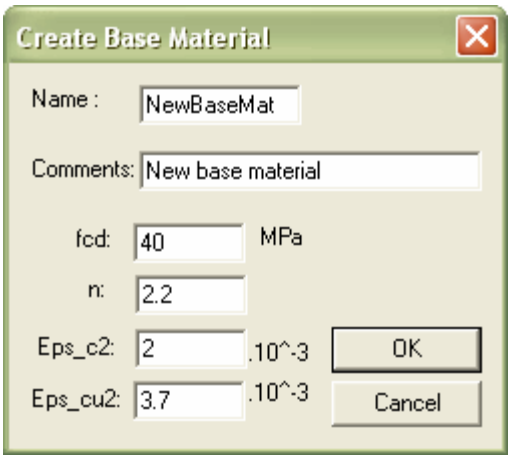


Fig.32 – Assistente de materiais base

Após introduzir os parâmetros definidores da curva tensão-extensão, o nome e o comentário opcional, escolhe-se o nome do ficheiro onde vão ser guardados estes dados.

O assistente de materiais para armadura pode ser invocado seleccionando "Tools > Rebar Material Creator…" (Figura 33).

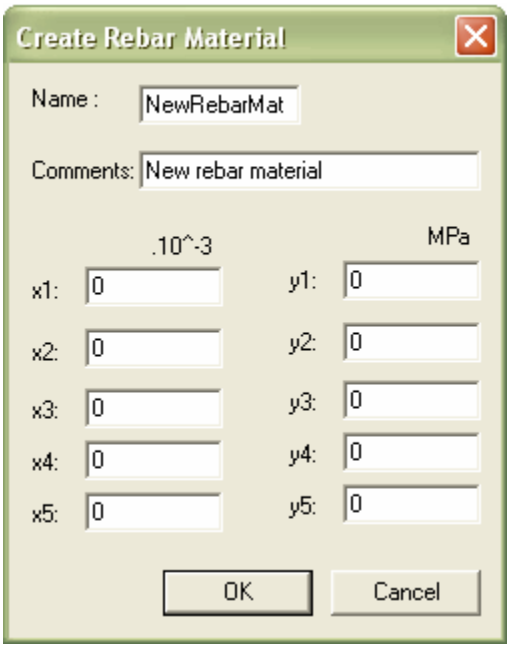


Fig.33 – Assistente de materiais para armaduras

Deve se introduzido o nome do material e um comentário opcional. A curva tensão-extensão é definida por intermédio da introdução das coordenadas de cinco pontos, de acordo com as indicações já referidas na Secção 3.2.2. Seguidamente escolhe-se o nome do ficheiro onde vão ser guardados os dados.

A curva tensão extensão do material base pode também ser definida pelas coordenadas de um conjunto de pontos. No entanto, não existe nenhum assistente para criar um material base nestas condições, uma vez que o número de pontos a introduzir poderia ser elevado. Assim, os ficheiros correspondentes a materiais base definidos por conjuntos de pontos devem ser preparados pelo utilizador.

6.12. EXPORTAR IMAGENS DO PROGRAMA PARA FICHEIRO

A exportação do desenho representado na janela para um ficheiro "BMP" pode ser efectuada por intermédio da opção "File > Export View Image".

7
EXEMPLOS DE CÁLCULO

7.1. INTRODUÇÃO

Neste capítulo apresentam-se alguns exemplos de cálculo, procurando-se escolher secções realistas do ponto de vista da engenharia civil. Os exemplos apresentados exploram as potencialidades do programa CSAnalysis e permitem comprovar os seus resultados.

7.2. EXEMPLO 1 – DIMENSIONAMENTO DE UMA SECÇÃO RECTANGULAR EM FLEXÃO SIMPLES

Pretende-se fazer o dimensionamento das armaduras de uma secção rectangular, nas condições da Figura 34.

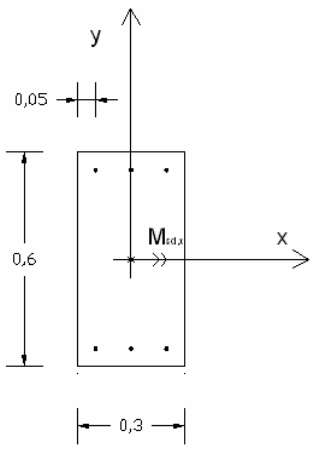


Fig.34 – Secção rectangular cuja armadura se pretende dimensionar (dimensões em metros)

O material base utilizado é o betão da classe C20/25 e o material das armaduras é o aço S400 B (valor característico da tensão de cedência igual a 400 MPa e classe de ductilidade B). Os varões de armadura devem ser dispostos de modo que 80% da área total de armadura fique localizada no banzo inferior e os restantes 20% no banzo superior. A secção deverá resistir a um momento flector de cálculo relativamente ao eixo horizontal de 250 kN·m.

Para obter a solução recorrendo ao programa CSAnalysis, modelou-se uma secção nas condições da Figura 34, com varões de 3Φ16 na face inferior e 3Φ8 na face superior. O resultado do

dimensionamento e os correspondentes diagramas de extensões e tensões no material base são indicados nas Figuras 35 e 36.

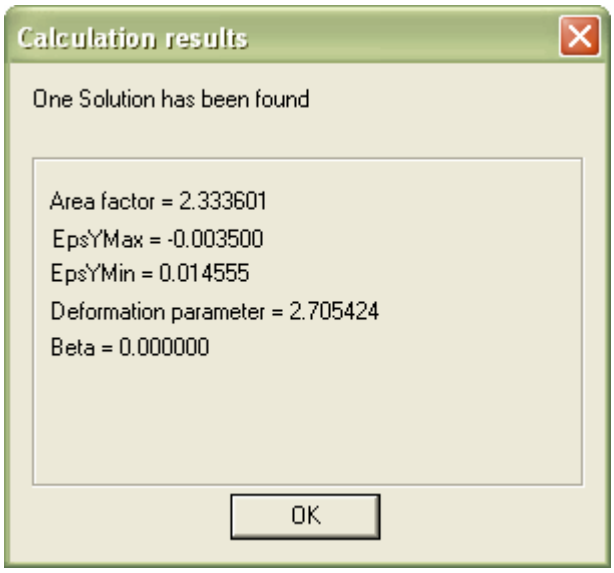


Fig.35 – Resultados do dimensionamento da secção com o programa CSAnalysis

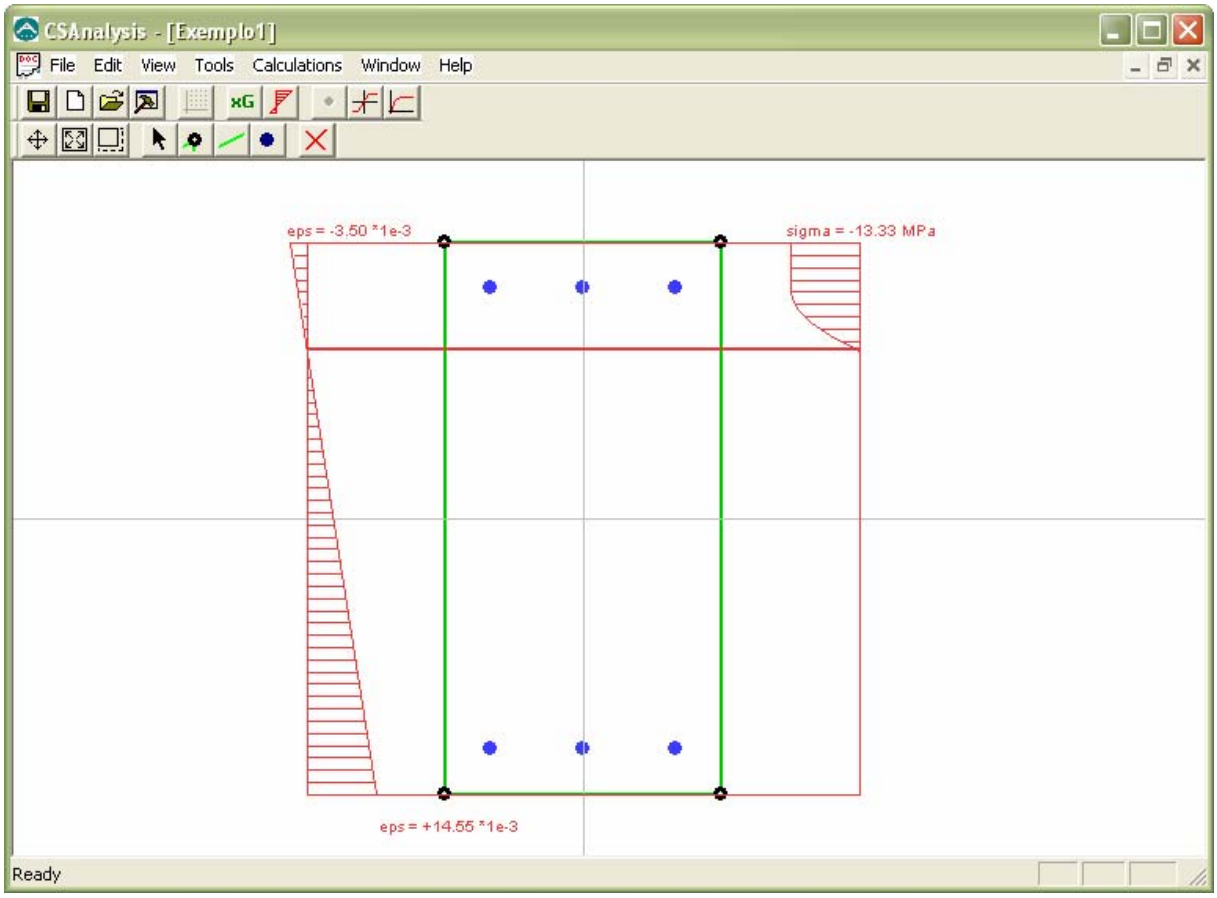


Fig.36 – Diagramas de extensões e tensões no material base para a secção dimensionada

Concluindo-se que é necessária uma área total de armadura 2.334 vezes superior à adoptada inicialmente. Sendo assim é necessária uma área total de 17.61 cm^2 de armadura. Conclui-se igualmente que a ruptura ocorre com aço no banzo inferior em situação de cedência.

7.3. EXEMPLO 2 – DIMENSIONAMENTO EM FLEXÃO COMPOSTA DE UMA SECÇÃO EM T

Pretende-se fazer o dimensionamento das armaduras de uma secção em T, com a geometria indicada na Figura 37.

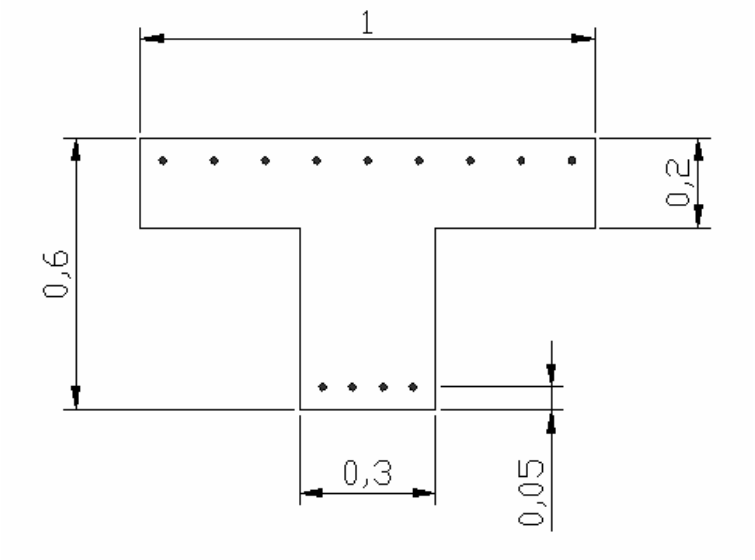


Fig.37 – Secção em T cuja armadura se pretende dimensionar (dimensões em metros)

O material base utilizado é o betão da classe C25/30 e o material das armaduras é o aço S500 C. Os varões de armadura devem ser dispostos de modo que 50% da área total de armadura se localize no banzo inferior e os restantes 50% no banzo superior. Em estado limite último a secção deverá resistir aos seguintes esforços de cálculo: N_{Ed}= -400 kN, $M_{Ed,x}$=-356 kN·m e $M_{Ed,y}$= 0 kN·m (convenção de sinais de acordo com a regra da mão direita).

A secção foi modelada com o programa dispondo 4Φ12 no banzo inferior e 9Φ8 no banzo superior, de modo a dispor metade da área total de armaduras em cada banzo. O resultado do dimensionamento e os correspondentes diagramas de extensões e tensões no material base são indicados nas Figuras 38 e 39.

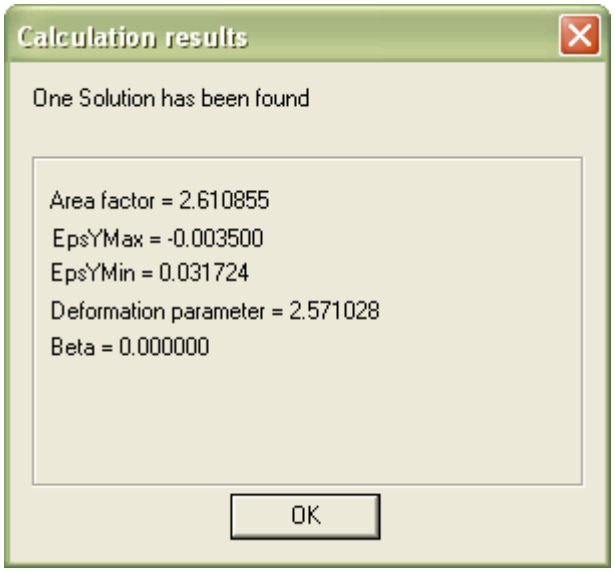


Fig.38 – Resultados do dimensionamento da secção com o programa CSAnalysis

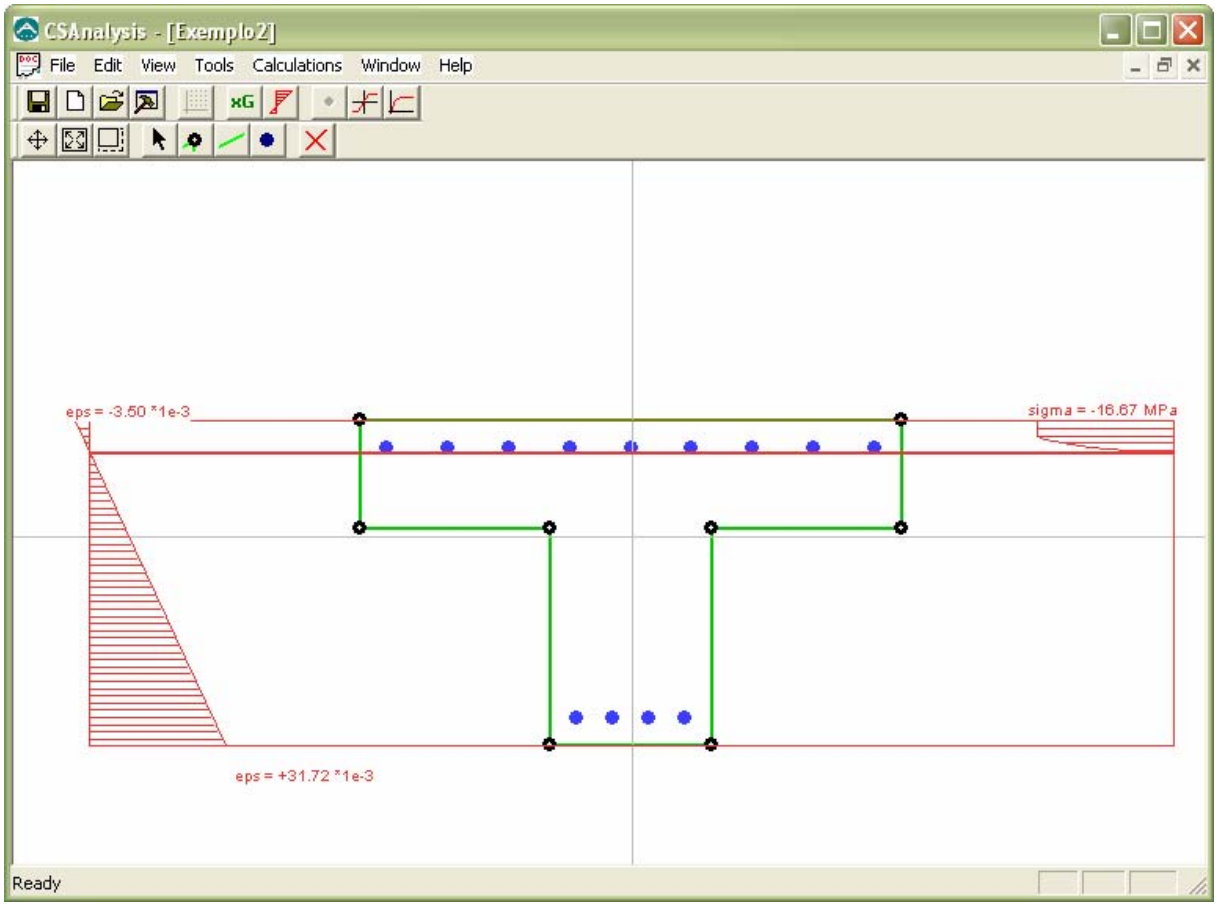


Fig.39 – Diagramas de extensões e tensões no material base para a secção dimensionada

Concluindo-se que é necessária uma área total de armaduras 2.611 vezes superior à adoptada. Sendo assim, é necessária uma área total de 23.6 cm^2 de armadura. Conclui-se igualmente que a ruptura ocorre com aço no banzo inferior em situação de cedência.

7.4. EXEMPLO 3 – VERIFICAÇÃO DA CAPACIDADE RESISTENTE DE UMA SECÇÃO CIRCULAR EM COMPRESSÃO SIMPLES

Pretende-se verificar a capacidade resistente da secção circular indicada na Figura 40.

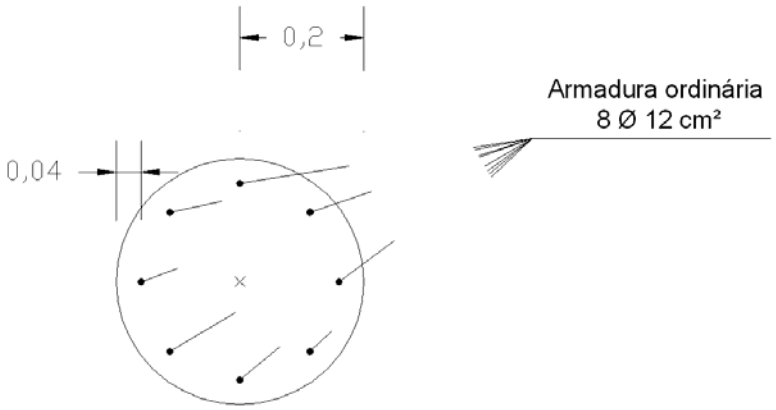


Fig.40 – Secção circular (dimensões em metros)

A secção fica sujeita a um esforço axial de compressão simples. Os materiais utilizados são o betão da classe C25/30 e o aço A500 C.

O programa CSAnalysis forneceu duas situações extremas em que a secção atinge o limite da capacidade resistente, tal como se encontra indicado nas Figuras 41 e 42.

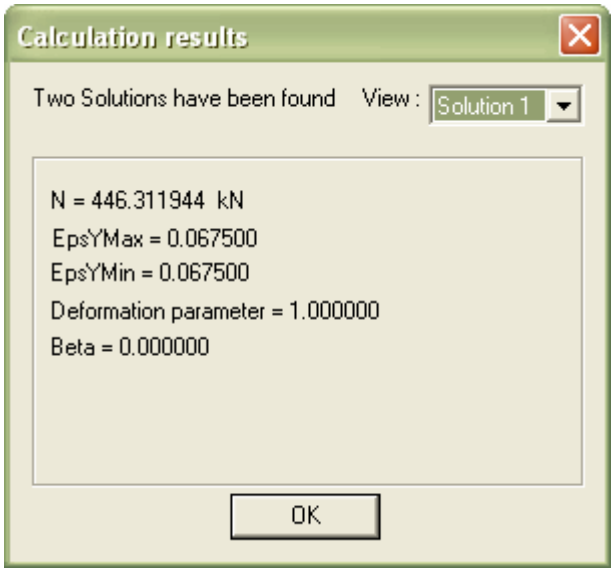


Fig.41 – Primeira solução obtida com o programa CSAnalysis

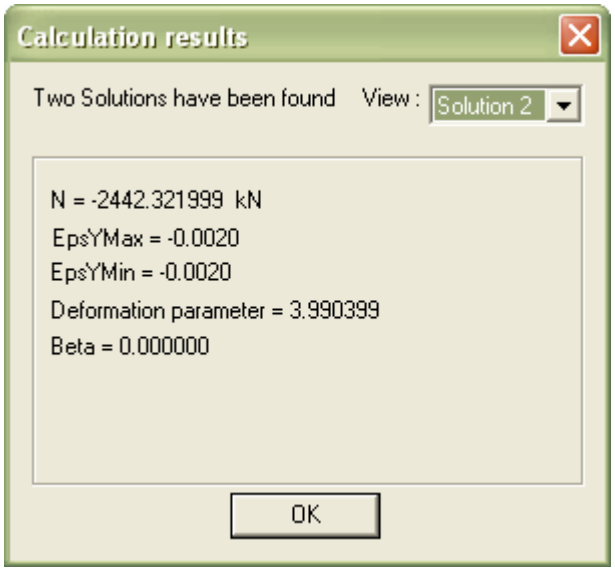


Fig.42 – Segunda solução obtida com o programa CSAnalysis

A primeira solução corresponde ao funcionamento da peça como tirante, i.e., em tracção simples. A correspondente força de tracção máxima é de 446 kN.

A segunda solução corresponde a um funcionamento em compressão simples, sendo suportado um esforço axial de compressão máximo de 2442 kN. Os correspondentes diagramas de extensões e tensões no material base encontram-se na Figura 43.

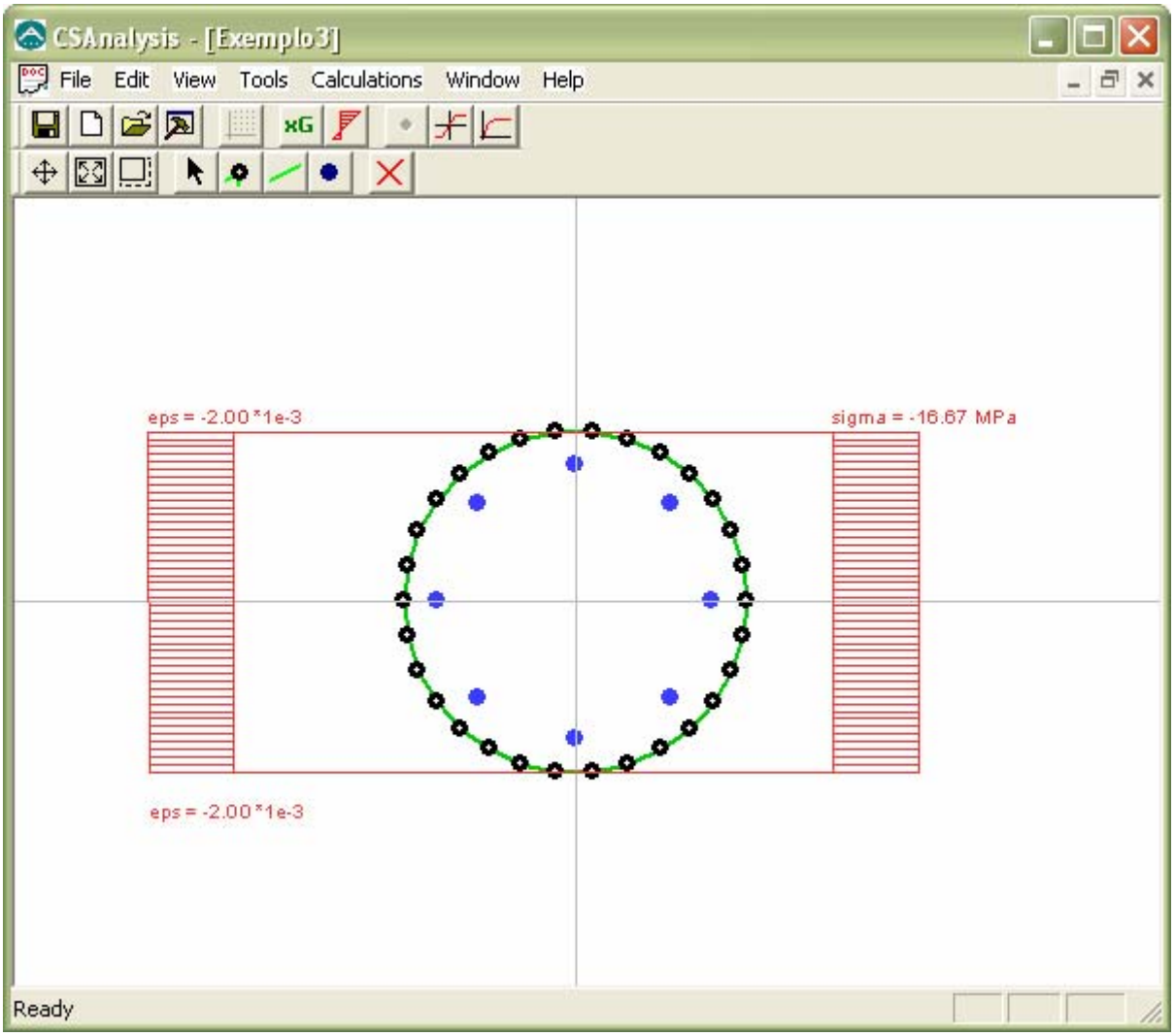


Fig.43 – Diagramas de extensões e tensões em compressão simples

Como se pode verificar, em toda a área abrangida pelo material base é atingido o limite de extensão em compressão simples permitido pelo EC2 [2].

7.5. EXEMPLO 4 – VERIFICAÇÃO DA SEGURANÇA DE UMA VIGA CAIXÃO PRÉ-ESFORÇADA

Pretende-se fazer uma verificação de segurança da viga caixão pré-esforçada indicada na Figura 44.

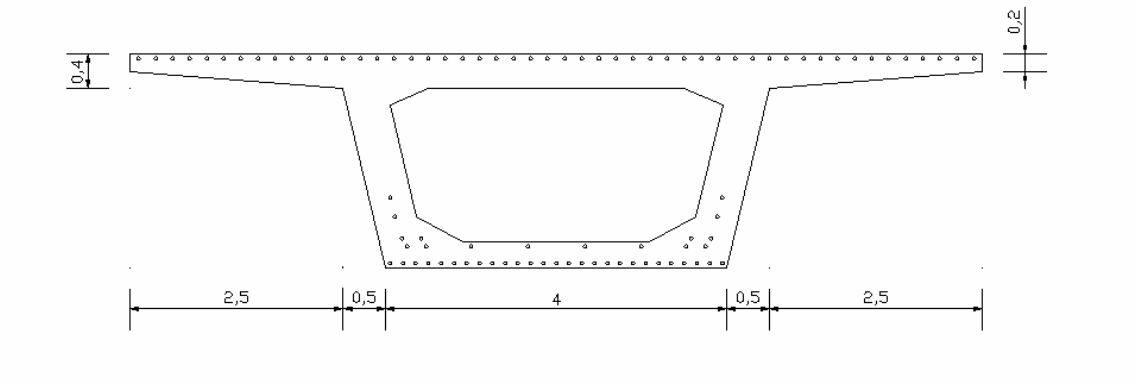


Fig.44 – Viga caixão (dimensões em metros)

Os materiais utilizados são o betão C25/30 como material base, aço A400 C para as armaduras ordinárias e aço Ap1860 para as armaduras pré-esforcadas. As armaduras de pré-esforço utilizadas possuem uma tensão última da cálculo f_{pud}= 1452.2 MPa e uma tensão de pré-esforço a tempo infinito σ_{pinf}= 1100 MPa.

As armaduras ordinárias são distribuídas da seguinte forma: 27Φ16 na face inferior e 50Φ10 na face superior. A armadura de pré-esforço é distribuída por 16 cabos com 11.16 cm^2 de área cada.

Em estado limite último a secção está sujeita aos seguintes esforços: N_{Ed}= 0 kN, $M_{Ed,x}$=-50 000 kN·m e $M_{Ed,y}$=-25 000 kN·m (convenção de sinais de acordo com a regra da mão direita).

O modelo da secção representado pelo programa CSAnalysis encontra-se na Figura 45.

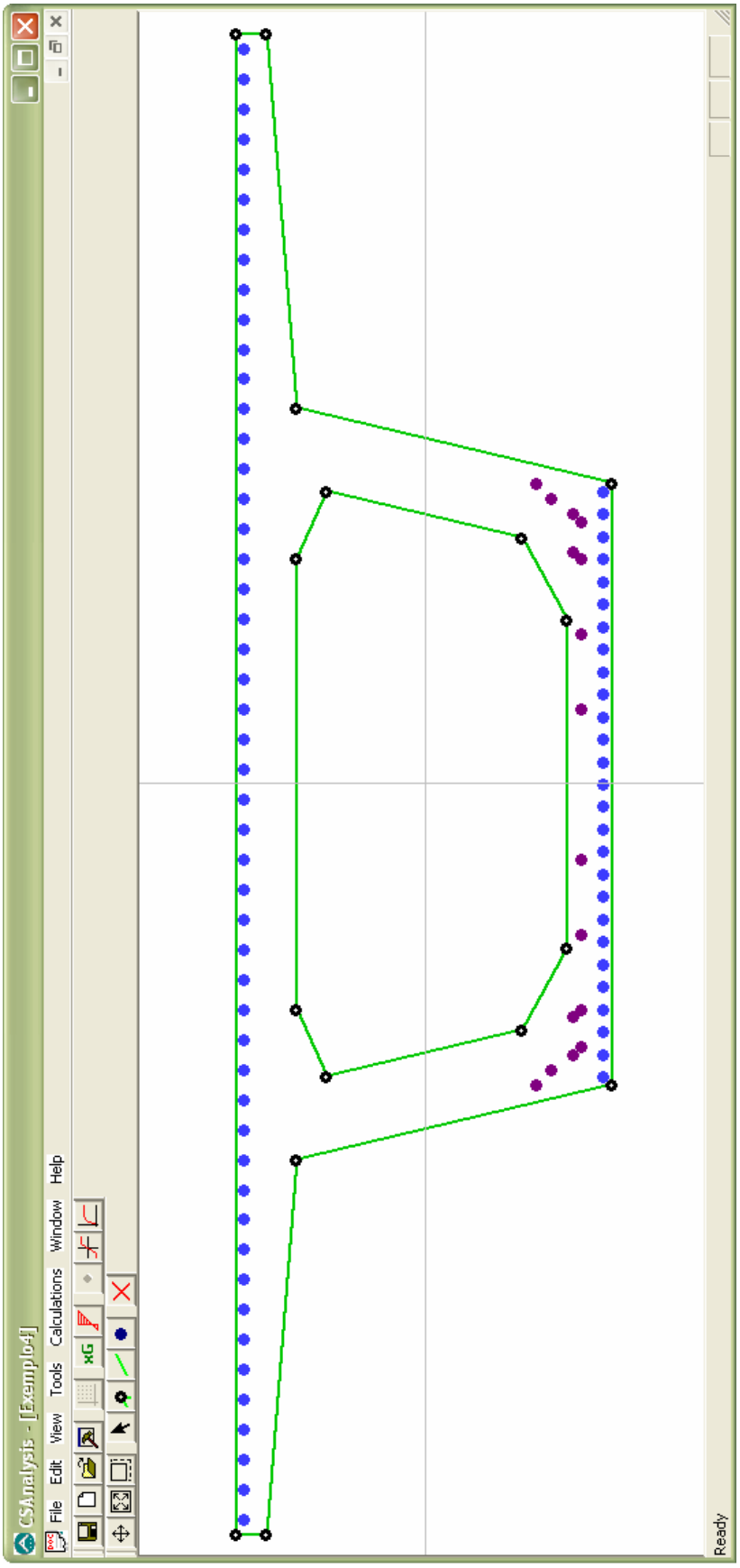


Fig.45 – Modelo CSAnalysis da viga caixão pré-esforçada

Se o esforço axial N_{Ed} for considerado fixo, tal como o momento $M_{Ed,y}$, pode calcular-se o momento máximo $M_{Rd,x}$ a que secção resiste nessas condições. Assim, obtêm-se dois valores extremos para os quais se dá o início da ruptura, tal como se indica nas Figuras 46 e 47.

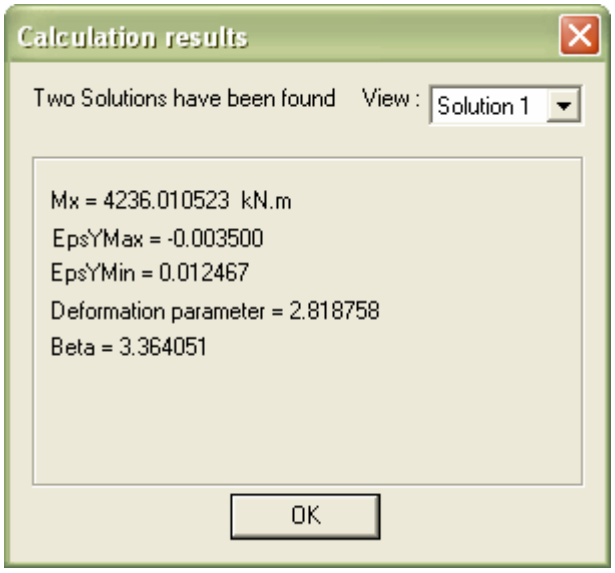


Fig.46 – Primeira solução obtida com o programa

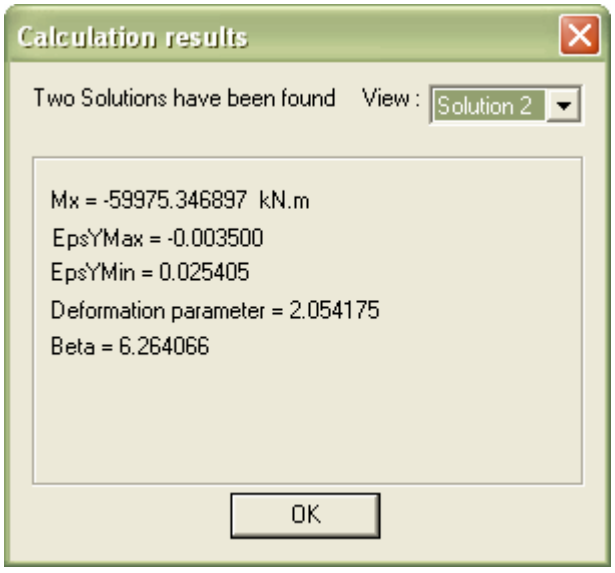


Fig.47 – Segunda solução obtida com o programa

Os correspondentes diagramas de extensão e tensão no material base são, respectivamente, os que se apresentam nas Figuras 48 e 49.

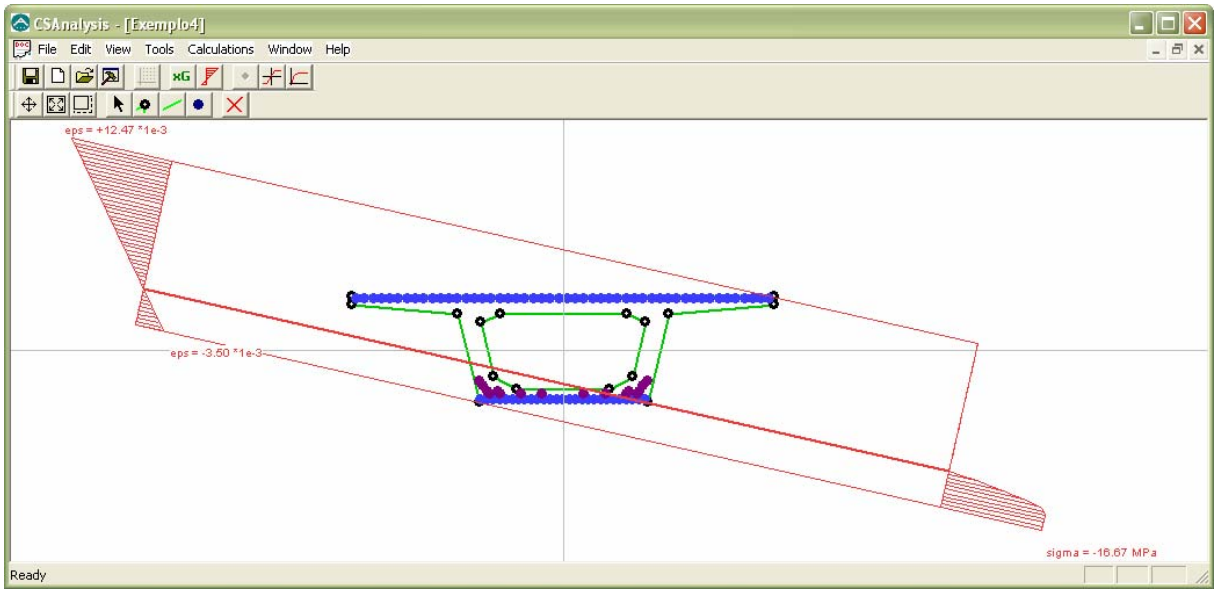


Fig.48 – Diagramas de extensões e tensões no material base correspondentes à primeira solução

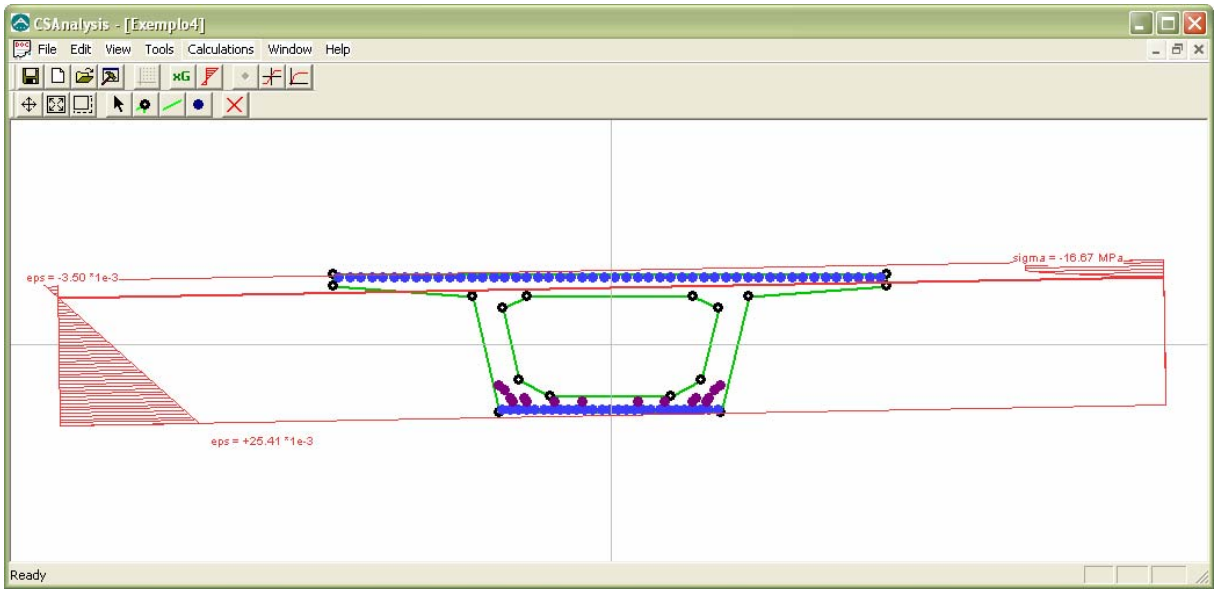


Fig.49 – Diagramas de extensões e tensões no material base correspondentes à segunda solução

Conclui-se que para os esforços N_{Ed}= 0 kN e $M_{Ed,y}$=-25 000 kN·m , o momento $M_{Rd,x}$ pode variar entre 4 236 kN m e -59 975 kN m verificando-se sempre a existência de segurança.

7.6. EXEMPLO 5 – CÁLCULO DA RESITÊNCIA DE UMA SECÇÃO REFORÇADA COM CFRP

Pretende-se calcular a resistência de uma secção em betão reforçada com aço e com um material FRP, baseado em fibras de carbono (CFRP), tal como se indica na Figura 50.

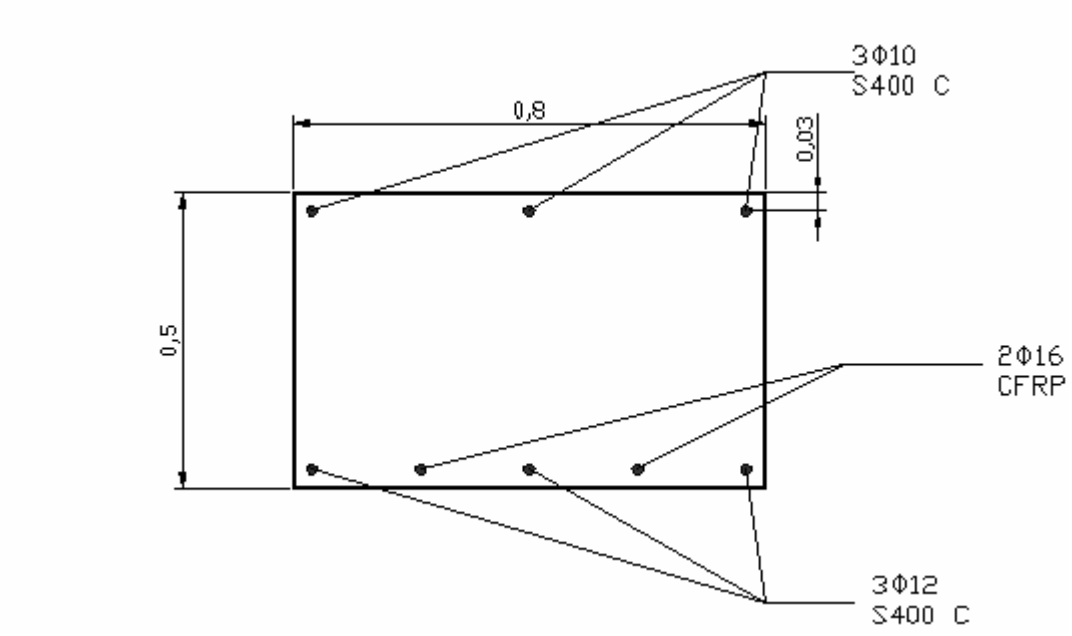


Fig.50 – Secção reforçada com aço e CFRP (dimensões em metros)

O material base é o betão C30/37. Os restantes materiais são o aço S400 C, 3Φ10 na face superior e 3Φ12 na face inferior, e o CFRP, 2Φ16 na face inferior. O CFRP utilizado apresenta uma resistência característica f_{fk}= 3000 MPa e módulo de elasticidade E= 210 GPa. O coeficiente de segurança adoptado para o material CFRP foi γ_m= 1.35, considerando os coeficientes da Tabela 4 e um fraco controlo de qualidade nas condições de aplicação do material. Assim sendo, é necessário criar um novo material de armadura correspondente ao CFRP, tal como se indica na Figura 51.

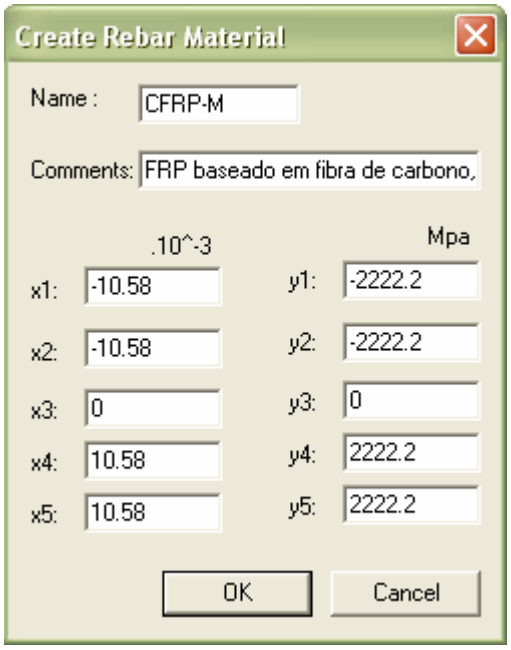


Fig.51 – Introdução de um novo material correspondente ao CFRP

Procedendo aos cálculos da resistência da secção em flexão simples, obtém-se os resultados que se apresentam nas Figura 52 e 53.

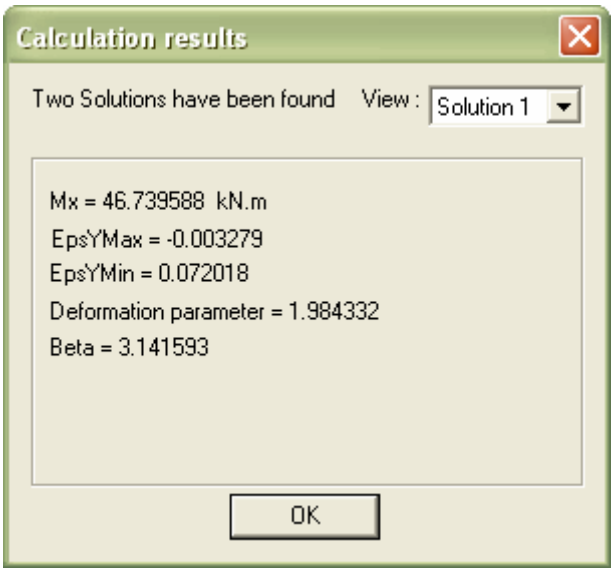


Fig.52 – Primeira solução obtida com o programa

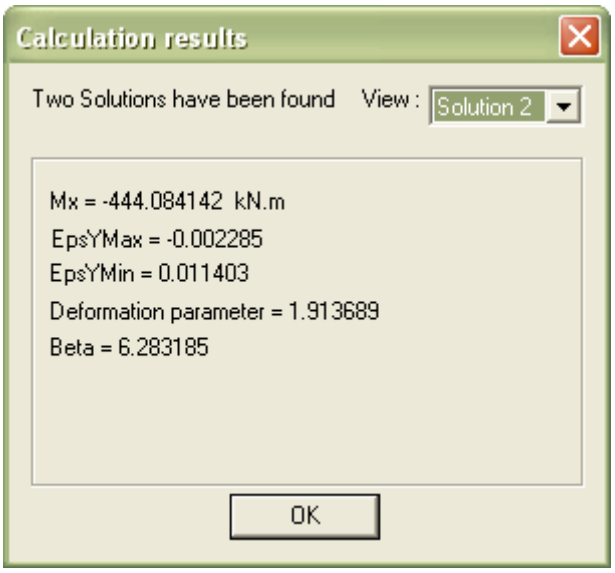


Fig.53 – Segunda solução obtida com o programa

Conclui-se que a secção resiste a momentos $M_{Ed,x}$ entre 47 kN·m e -444 kN·m. Os diagramas de tensões e extensões no material base, correspondentes a um momento igual a -444 kN·m, encontram-se representados na Figura 54.

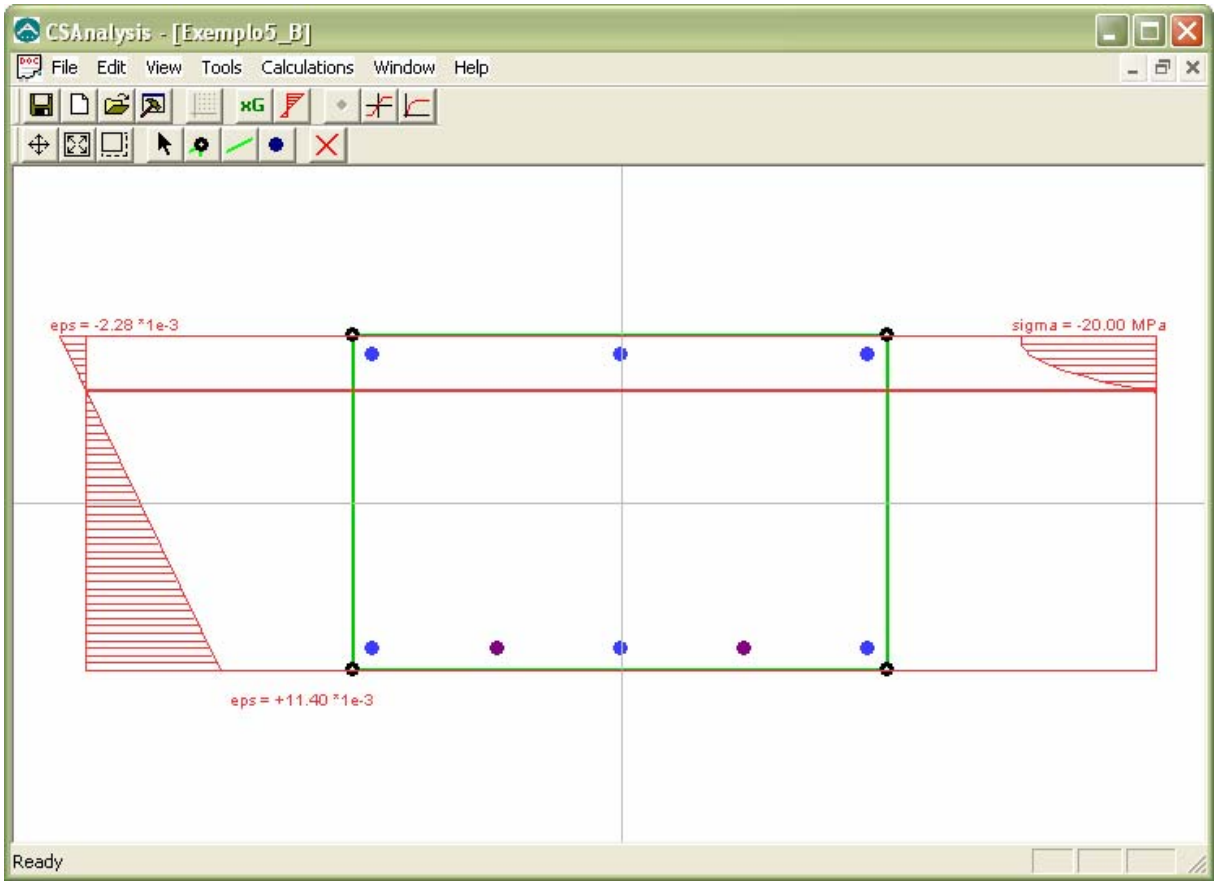


Fig.54 – Diagramas de extensões e tensões no material base para M_x= -444 kN·m

8
CONCLUSÕES

São muito breves as conclusões deste trabalho, uma vez que o desenvolvimento do código computacional descrito teve como objectivos essenciais, para além da utilidade do próprio programa, a síntese e aprendizagem de conceitos, bem como o desenvolvimento de capacidades ligadas à modelação numérica. Assim, a generalidade do conhecimento adquirido encontra-se envolto numa complexidade intuitiva detida pelo autor, uma vez que consiste sobretudo no desenvolvimento de uma criatividade própria para o tratamento de problemas de modelação numérica e da aptidão para lidar com paradigmas e formas de estruturar informação, bem como os correspondentes processos de cálculo. Apesar disso são apresentadas ao longo do texto as principais questões e as soluções adoptadas para os problemas que foram surgindo no decorrer do trabalho. Pode assim dizer-se que este trabalho poderá de alguma forma inspirar a criatividade de alguém que pretenda conceber modelos computacionais para resolver problemas de engenharia civil ou de áreas contíguas, fundamentalmente se a pessoa em causa possuir alguma experiência na construção de algoritmos e na programação orientada por objectos.

A programação orientada por objectos é sobretudo um paradigma de programação, mas constitui também uma forma inovadora de abordagem na modelação dos fenómenos. Este tipo de programação vai muito para além da execução sequencial de instruções e encontra-se ainda pouco explorada no domínio da engenharia civil. Ao permitir que se modelem objectos autónomos, que podem interagir entre si segundo uma sequência ou um esquema global que não é conhecido à partida, possibilita-se a concepção de novas abordagens para os problemas de engenharia, completamente distintas das tradicionais.

O programa CSAnalysis apresenta mais de 7500 linhas de código distribuídas por 138 ficheiros e ainda outros ficheiros de apoio que não contêm código computacional propriamente dito. O facto de o código ter sido escrito com base num paradigma orientado por objectos permite uma fácil adaptação a novas situações que não tenham sido previstas no código original, adicionando e/ou adaptando classes de objectos utilizadas pelo programa. Se se pretender rescrever o programa de tal forma que os materiais de reforço apresentem curvas tensão-extensão que não se encontrem definidas por troços lineares, mas sim por uma função parametrizada, é suficiente modificar a classe de objectos correspondente ao material. Algumas das classes podem além disso ser reaproveitadas para outros programas orientados por objectos. Assim, se por exemplo existir a necessidade de relacionar dois sistemas de coordenadas no plano, é possível reaproveitar na íntegra a classe que foi escrita no âmbito deste programa, adicionando-lhe, se necessário, novas funcionalidades.

O programa desenvolvido cumpre todos os requisitos de robustez, generalidade de problemas que pode resolver e facilidade de utilização para os quais foi concebido. Apresenta ainda tempos de

execução muito baixos na resolução do problema de flexão composta desviada, sendo esta a fase mais exigente em termos de cálculo. Alguns módulos podem ser incorporados em outras aplicações, como por exemplo em programas de cálculo de estruturas reticuladas, podendo inclusive ser adoptados por outras plataformas e sistemas operativos, uma vez que se encontram escritos em linguagem C++, que é uma das linguagens mais utilizadas em software comercial, encontrando-se amplamente divulgada.

BIBLIOGRAFIA

[1] *Regulamento de Estruturas de Betão Armado e Pré-Esforçado*, Decreto-Lei n.º 349-C/83, 30 de Julho.

[2] *Eurocode 2: Design of concrete structures - Part 1-1: General rules and rules for buildings*. Comité Européen de Normalisation (CEN), EN 1992-1-1:2004: E, 2004.

[3] *Eurocode - Basis of structural design*. Comité Européen de Normalisation (CEN), EN 1990:2002 E, 2002.

[4] *Regulamento de Segurança e Acções para Estruturas de Edifícios e Pontes*, Decreto-Lei n.º 235/83, 31 de Maio.

[5] Audenaert, K., Balázs, G., Blashko, M., Blontrock, H., Czaderski, C., David, Duckett W., Hordijk, D., Leeming, M., E., Matthys, S., Meier, H., Monti, G., Moss, R., Neubauer, U., Niedermeier, R., Tomasso, A., Talisten, B., Triantafillou, T., Zehetmaier, G., Zilch, K., *Externally bounded FRP reinforcement for RC structures*, International Federation for Structural Concrete, 2001.

[6] Azevedo, A. F. M., *Optimização de Estruturas com Comportamento Linear e Não Linear*. Dissertação para Doutoramento em Engenharia Civil, Faculdade de Engenharia da Universidade do Porto, 1994.

[7] Ferreira, A. E. V., Azevedo, A. F. M., *Cálculo da Capacidade Resistente e Dimensionamento de Armaduras de Secções Quaisquer de Betão Armado e Pré-esforçado Sujeitas a Flexão Composta Desviada*, 2º Encontro Nacional Sobre Estruturas Pré-Esforçadas, Porto, 1988.
http://civil.fe.up.pt/alvaro/pdf/1988_ENEPE_Calc_Armad_Sec_Quaisquer_Flex_Desv.pdf

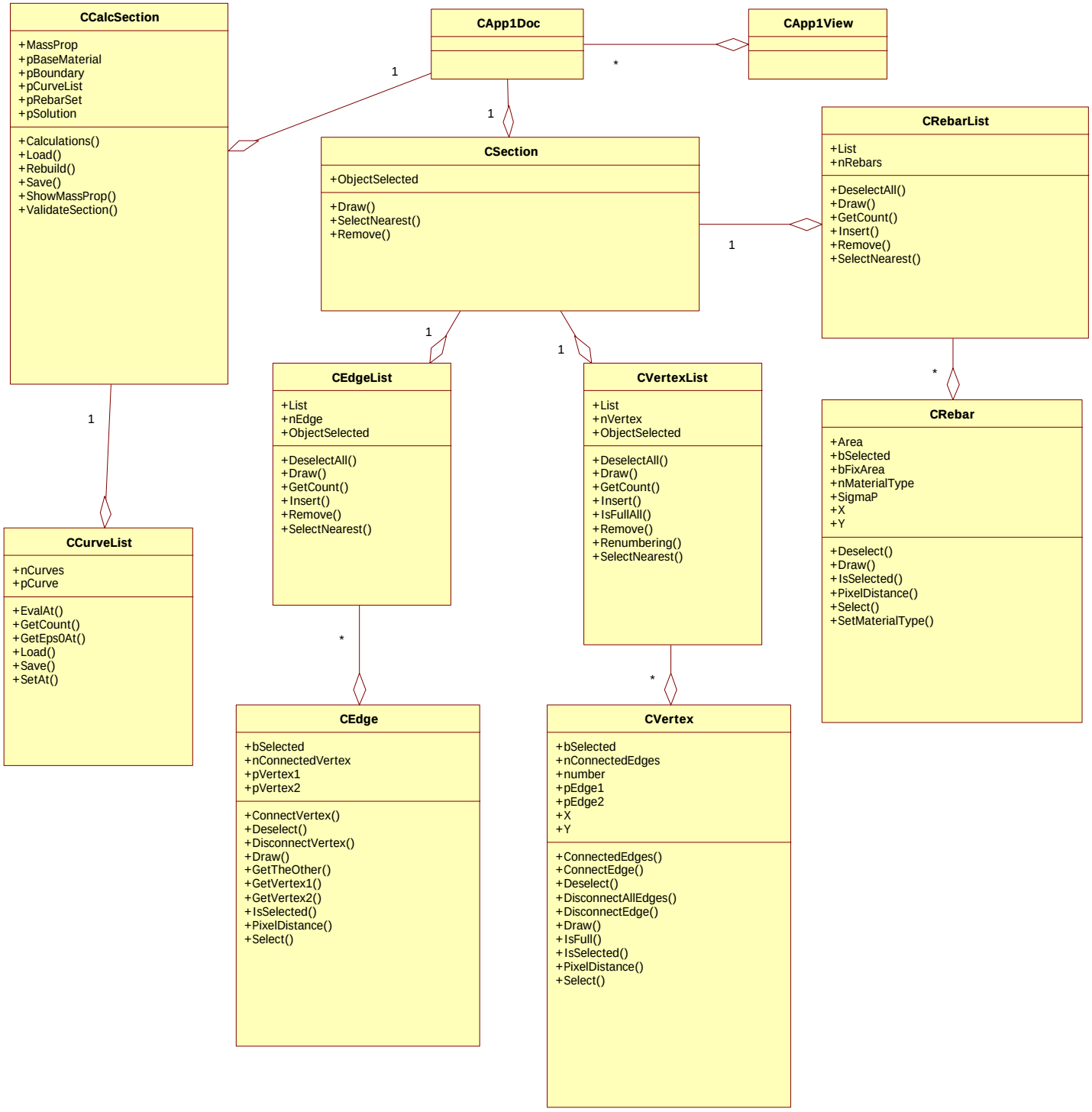
[8] Apostol, T. *Calculus*, Volume 2, Editorial Reverte, 1999.

[9] Pereira, P., Rodrigues, P., Sousa, M., *Programação em C++ Conceitos básicos e Algoritmos*, FCA-Editora de Informática, 1998.

[10] Pereira, P., Rodrigues, P., Sousa, M., *Algoritmos e Estruturas de Dados*, FCA-Editora de Informática, 1998.

ANEXO A1

DIAGRAMA DE CLASSES DA ESTRUTURA DE DADOS



Simbologia

Classe

Nome da classe

+Atributo1
+Atributo2
+...

+método1()
+método2()
+...()

Cardinalidade (#)

1 - exactamente um

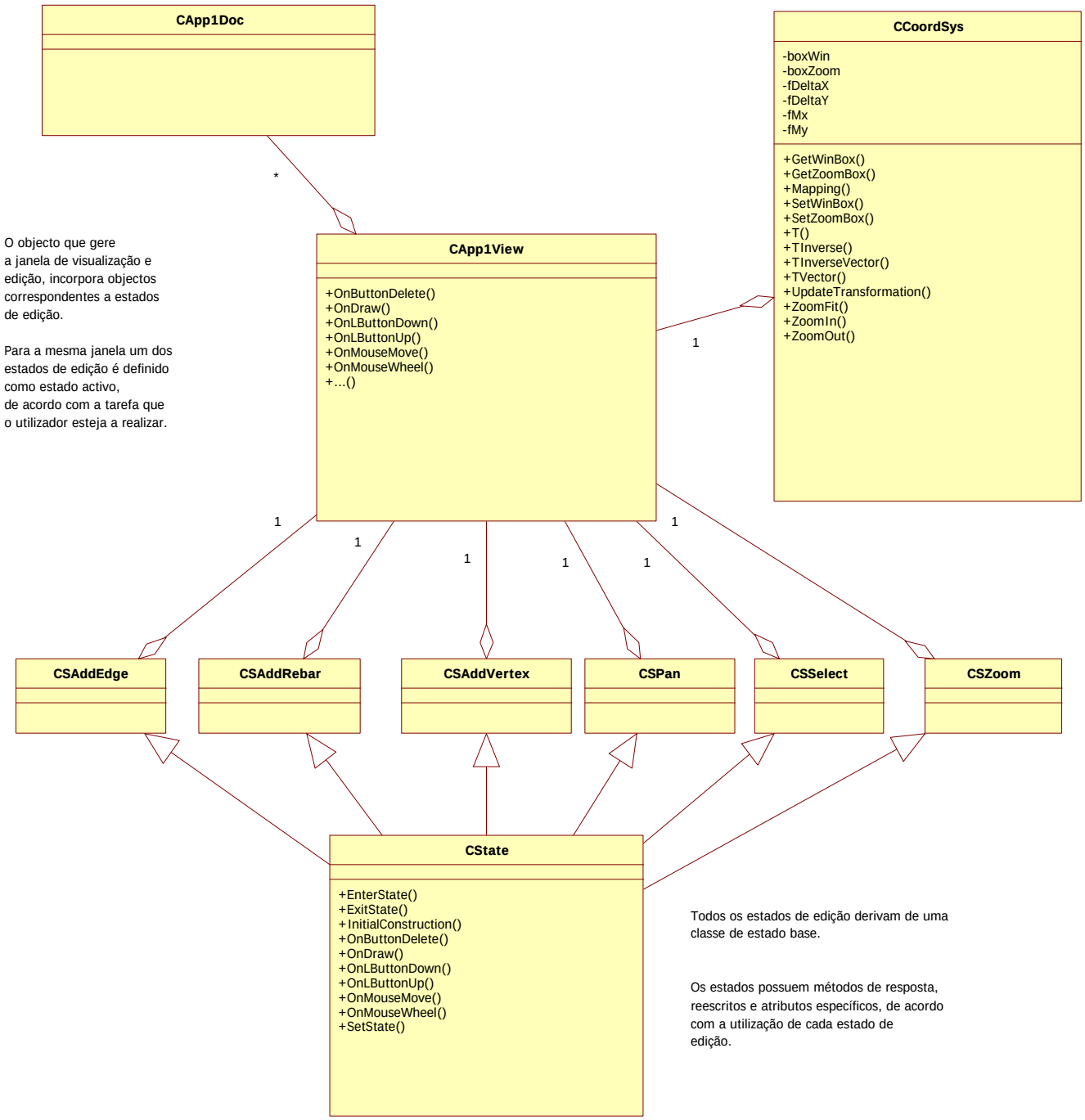
* - zero, um ou muitos

Classe principal — Agregação — **Classe agregada**

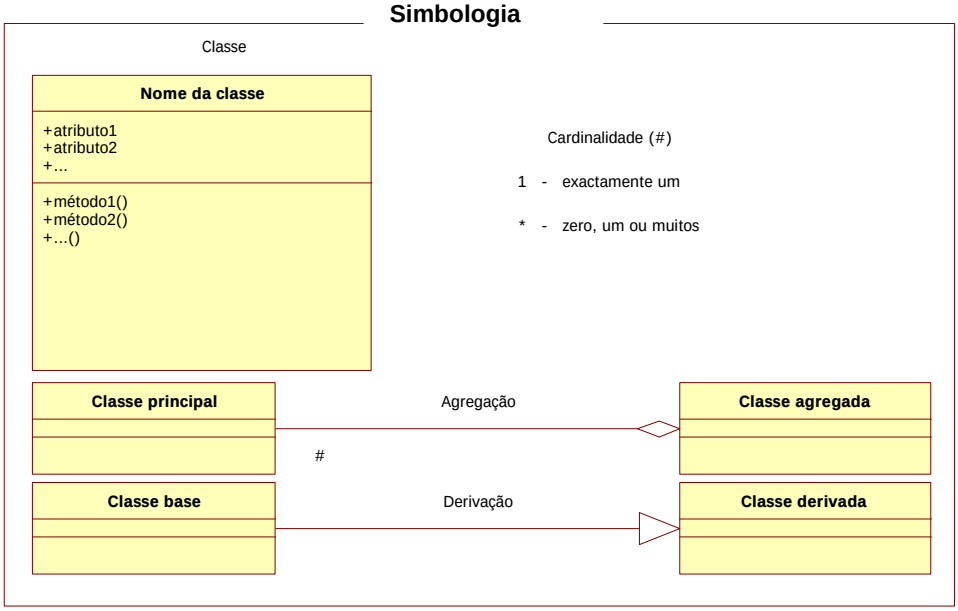
\#

ANEXO A2

DIAGRAMA DE CLASSES DA INTERFACE GRÁFÍCA

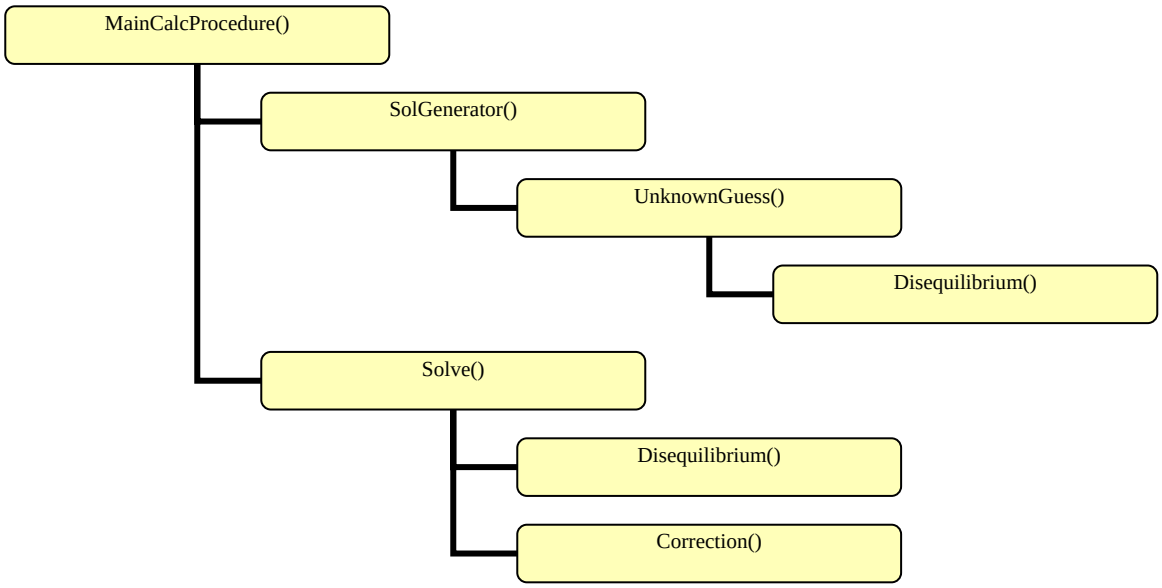

CApp1Doc
CCoordSys
-boxWin
-boxZoom
-fDeltaX
-fDeltaY
-fMx
-fMy
+GetWinBox()
+GetZoomBox()
+Mapping()
+SetWinBox()
+SetZoomBox()
+T()
+TInverse()
+TInverseVector()
+TVector()
+UpdateTransformation()
+ZoomFit()
+ZoomIn()
+ZoomOut()
*
1
O objecto que gere
a janela de visualização e
edição, incorpora objectos
correspondentes a estados
de edição.
Para a mesma janela um dos
estados de edição é definido
como estado activo,
de acordo com a tarefa que
o utilizador esteja a realizar.
CApp1View
+OnButtonDelete()
+OnDraw()
+OnLButtonDown()
+OnLButtonUp()
+OnMouseMove()
+OnMouseWheel()
+...()
1
1
1
1
1
1
1
CSAddEdge
CSAddRebar
CSAddVertex
CSPan
CSSelect
CSZoom
CState
+EnterState()
+ExitState()
+InitialConstruction()
+OnButtonDelete()
+OnDraw()
+OnLButtonDown()
+OnLButtonUp()
+OnMouseMove()
+OnMouseWheel()
+SetState()
Todos os estados de edição derivam de uma
classe de estado base.
Os estados possuem métodos de resposta,
reescritos e atributos específicos, de acordo
com a utilização de cada estado de
edição.

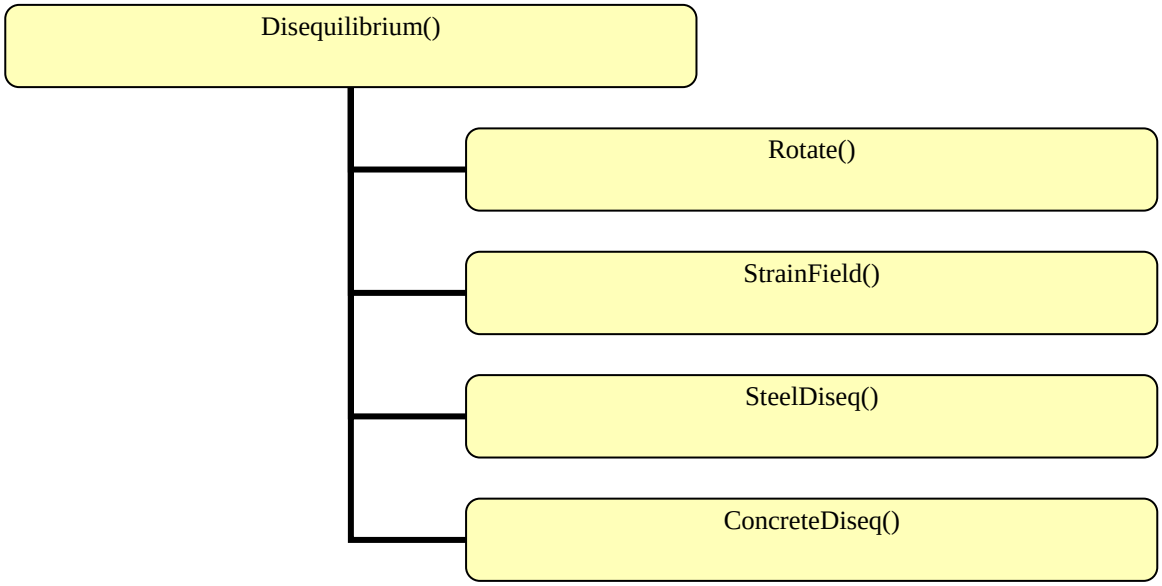
Simbologia


Classe
Nome da classe
+atributo1
+atributo2
+...
+método1()
+método2()
+...()
Cardinalidade (#)
1 - exactamente um
* - zero, um ou muitos
Classe principal
Agregação
Classe agregada
#
Classe base
Derivação
Classe derivada

ANEXO A3

DIAGRAMA DE HIERARQUIA ENTRE PROCEDIMENTOS DE CÁLCULO


MainCalcProcedure()
SolGenerator()
UnknownGuess()
Disequilibrium()
Solve()
Disequilibrium()
Correction()


Disequilibrium()
Rotate()
StrainField()
SteelDiseq()
ConcreteDiseq()

ANEXO A4

CÓDIGO DE ALGUNS MÓDULOS DO PROGRAMA CSANALYSIS

```
/*       1         2         3         4         5         6         7       */
/* 456789_123456789_123456789_123456789_123456789_123456789_123456789_123456 */

/*****************************************************************************/
/*                                                                           */
/* MainCalcProcedure.cpp                                                     */
/* Procedimento de calculo principal                                         */
/*                                                                           */
/*                                                                           */
/* Version:                                                                  */
/* Date   :    29 Outubro 2007                                               */
/* Authors:    H.David Miranda                                               */
/*                                                                           */
/*****************************************************************************/

#include "stdafx.h"
#include "App1.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

#include "DataStructs.h"
#include "CurveList.h"
#include "RebarSet.h"
#include "BaseMaterial.h"
#include "Stress.h"
#include "Diseq.h"
#include "Solution.h"
#include "Bondary.h"

void SolGenerator(double matInitialSol[50+1][3+1],

                  CBondary* pBondary,
                  CRebarSet* pRebarSet,
                  CCurveList* pCurveList,
                  CStress *pStress,
                  CBaseMaterial* pBaseMaterial,
                  CDiseq* pDiseq,
                  CSolution *pSolution);

int Solve(CBondary* pBondary,
          CRebarSet* pRebarSet,
          CCurveList* pCurveList,
          CStress *pStress,
          CBaseMaterial* pBaseMaterial,
          CDiseq* pDiseq,
          CSolution *pSolution,
          double* vecGuess);


void MainCalcProcedure( CBondary* pBondary,
                        CRebarSet* pRebarSet,
                        CCurveList* pCurveList,
                        CStress *pStress,
                        CBaseMaterial* pBaseMaterial,
                        CSolution *pSolution)
{
    CDiseq Diseq;
    double matInitialSol[50+1][3+1];

    SolGenerator(matInitialSol,
                 pBondary,
                 pRebarSet,
                 pCurveList,
                 pStress,
                 pBaseMaterial,
                 &Diseq,
                 pSolution);

    double f1,f2,f3; // variaveis temporarias
    int a,b;
    int nMaxSol = (pStress->chUnknownNum == AreaFactor ? 1 : 2);
    for(a=1; (a<=50)&&(pSolution->nSolutions<nMaxSol); a++)
    {
        b = Solve(  pBondary,
                    pRebarSet,
```

```
                    pCurveList,
                    pStress,
                    pBaseMaterial,
                    &Diseq,
                    pSolution,
                    matInitialSol[a]);
        if (!b)
        {
            if (pSolution->nSolutions>0)
            {
                // verificar se a solucao e ou nao repetida
                f1 = (pSolution->beta[1]-pSolution->beta[2]);
                f2 = (pSolution->D[1]-pSolution->D[2]);
                f3 = (pSolution->Unk[1]-pSolution->Unk[2]);
                if (f1*f1 + f2*f2 + f3*f3 > 1e-4 )
                {
                    pSolution->nSolutions++;
                    return;
                }

            }
            else
            {
                // AreaFactor tem de ser possitivo.
                pSolution->nSolutions+=
                    ((pStress->chUnknownNum != AreaFactor) ||
                    ((pSolution->Unk[1])>-1e-3));
            }
        }
    }
}
```

```
/*       1         2         3         4         5         6         7       */
/* 456789_123456789_123456789_123456789_123456789_123456789_123456789_123456 */

/*************************************************************************/
/*                                                                       */
/* StrainField.cpp                                                       */
/* Define o campo de extensoes                                           */
/* Retorna o valor 0 em caso de sucesso                                  */
/*                                                                       */
/* Version:                                                              */
/* Date   :                                                              */
/* Authors:    H.David Miranda                                           */
/* Co-authors:                                                           */
/*                                                                       */
/*************************************************************************/

#include "stdafx.h"
#include "App1.h"

#ifdef  DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


#include <stdio.h>
#include <math.h>

#include "Bondary.h"
#include "RebarSet.h"
#include "CurveList.h"
#include "BaseMaterial.h"
#include "Solution.h"


int StrainField(double f_Domin, CBondary* pSection,
                CRebarSet* pRebarSet, CCurveList* pCurveList,
                CBaseMaterial* pBaseMaterial,
                double *pfEpsM,double *pfEpsB,CSolution *pSolution)
{

    int a;
    double f_h; /* altura da seccao medida na vertical*/
    double f_d; /* altura util da peça */

    /* extensao maxima com a secao toda em tracao */
    double f_epsMax=1;

    // extensao maxima de traccao no material base
    //compativel com a estensao limite de todas as rebars
    double f_epsCMax=1;
    double f_eps;

    double fSectionYMax, fSectionYMin;
    double fSectionXMax, fSectionXMin;

    /* extensões nos pontos extremos da seccao */
    double fEpsYMax, fEpsYMin;

    /* calculo dos pontos extremos da seccao */
    fSectionXMin = fSectionXMax = pSection->x[1];
    fSectionYMin = fSectionYMax = pSection->y[1];
    for (a=2; a<=(pSection->nVertex); a++)
    {
        if (pSection->y[a]>fSectionYMax)
            fSectionYMax=pSection->y[a];

        if (pSection->y[a]<fSectionYMin)
            fSectionYMin=pSection->y[a];

        if (pSection->x[a]>fSectionXMax)
            fSectionXMax=pSection->x[a];

        if (pSection->x[a]<fSectionXMin)
            fSectionXMin=pSection->x[a];

    }
```

```
f_h=(fSectionYMax-fSectionYMin);

a=pSolution->nSolutions+1;
pSolution->Ly[a] = fSectionYMin;
pSolution->Uy[a] = fSectionYMax;
pSolution->Lx[a] = fSectionXMin;
pSolution->Ux[a] = fSectionXMax;


/* Calculo da estensao maxima com toda a secção em traccao */
for (a=1; a<=(pRebarSet->nNumPt); a++)
{
    // pre-esforco na segunda parcela
    f_eps = pCurveList->GetEpsMaxAt(pRebarSet->vecCurve[a])
        -pCurveList->GetEps0At(pRebarSet->vecCurve[a],
        pRebarSet->vecSigmaP[a] );
    if (f_epsMax > f_eps)
    {
            f_epsMax = f_eps;
    }
}

if (f_Domin<=1)
{
    fEpsYMax=f_epsMax;
    fEpsYMin=f_epsMax;
}
else if (f_Domin<=2)
{
    fEpsYMax= f_epsMax - (f_Domin-1)* (f_epsMax - pBaseMaterial->f_epsMin);

    f_d = (fSectionYMax - pRebarSet->y[1]);
    // pre-esforco na segunda parcela
    f_eps = pCurveList->GetEpsMaxAt(pRebarSet->vecCurve[1])
        -pCurveList->GetEps0At(pRebarSet->vecCurve[1],
        pRebarSet->vecSigmaP[1] );
    fEpsYMin = (f_eps - fEpsYMax)/f_d*f_h + fEpsYMax;
    for (a=2; a<=(pRebarSet->nNumPt); a++)
    {
        f_d = (fSectionYMax - pRebarSet->y[a]);
        // pre-esforco na segunda parcela
        f_eps = pCurveList->GetEpsMaxAt(pRebarSet->vecCurve[a])
            -pCurveList->GetEps0At(pRebarSet->vecCurve[a],
            pRebarSet->vecSigmaP[a] );
        f_epsCMax = (f_eps - fEpsYMax)/f_d*f_h + fEpsYMax;
        if (fEpsYMin > f_epsCMax)
        {
            fEpsYMin = f_epsCMax;
        }
    }
}
else if (f_Domin<=3)
{
    fEpsYMax= pBaseMaterial->f_epsMin;

    f_d = (fSectionYMax - pRebarSet->y[1]);
    // pre-esforco na segunda parcela
    f_eps = pCurveList->GetEpsMaxAt(pRebarSet->vecCurve[1])
        -pCurveList->GetEps0At(pRebarSet->vecCurve[1],
        pRebarSet->vecSigmaP[1] );
    fEpsYMin = (f_eps - fEpsYMax)/f_d*f_h + fEpsYMax;
    for (a=2; a<=(pRebarSet->nNumPt); a++)
    {
        f_d = (fSectionYMax - pRebarSet->y[a]);
        // pre-esforco na segunda parcela
        f_eps = pCurveList->GetEpsMaxAt(pRebarSet->vecCurve[a])
            -pCurveList->GetEps0At(pRebarSet->vecCurve[a],
            pRebarSet->vecSigmaP[a] );
        f_epsCMax = (f_eps - fEpsYMax)/f_d*f_h + fEpsYMax;
        if (fEpsYMin > f_epsCMax)
        {
            fEpsYMin = f_epsCMax;
        }
    }

    fEpsYMin = fEpsYMin - (f_Domin-2)*(fEpsYMin-0);
}
else if (f_Domin<4)
{
```

```
        fEpsYMin = pBaseMaterial->f_epsc2;
        fEpsYMax = pBaseMaterial->f_epsMin -
            (f_Domin-3)*(pBaseMaterial->f_epsMin-fEpsYMin);
        fEpsYMin = 0 - (f_Domin-3)*(0-fEpsYMin);
    }
    else
    {
        fEpsYMax= pBaseMaterial->f_epsc2;
        fEpsYMin= fEpsYMax;
    }

    /* Definicao do campo de extensões epsilon(y)= (*pfEpsM)*y + (*pfEpsB) */
    *pfEpsM=(fEpsYMax-fEpsYMin)/f_h;
    *pfEpsB=fEpsYMax-(*pfEpsM)*fSectionYMax;
    a=pSolution->nSolutions+1;
    pSolution->EpsYMax[a] = fEpsYMax;
    pSolution->EpsYMin[a] = fEpsYMin;

    if (fabs(fEpsYMax-fEpsYMin)<1e-12)
    {
        pSolution->bShow[a] = 0;

        pSolution->x1[a] = 0;
        pSolution->x2[a] = 10;
        pSolution->y1[a] = pSolution->y2[a] = (fSectionYMin + fSectionYMax) / 2;

    }
    else
    {
        pSolution->bShow[a] = 1;

        pSolution->x1[a]=0;
        pSolution->x2[a]=10;
        pSolution->y1[a] = pSolution->y2[a] = -(*pfEpsB)/(*pfEpsM);
    }


    /* verificacao das condicoes especiais de traccao */
    /* se todas as armaduras estiver com tensão maxima pode considerar-se */
    /* dominio igual a 1 */
    pSolution->bDominEqualTo1[a] = false;
    if (f_Domin<=2)
    {
        double sigma_max;
        double sigma;
        int c;
        int b;
        // verifica se em todos os pontos
        // a tensao maxima de traccao e verificada
        pSolution->bDominEqualTo1[a] = true;
        for (b=1; b<=(pRebarSet->nNumPt); b++)
        {
            c = pRebarSet->vecCurve[b];

            sigma_max = pCurveList->EvalAt(c, pCurveList->GetEpsMaxAt(c));
            sigma = pCurveList->EvalAt(c,
                (*pfEpsM)*(pRebarSet->y[b]) +
                *pfEpsB +pCurveList->GetEps0At(c,pRebarSet->vecSigmaP[b] ));
            if ((sigma_max-1e-7)>sigma)
            {
                pSolution->bDominEqualTo1[a] = false;
            }

        }
    }

    return 0;
} /* StrainField */
```

```
/*       1         2         3         4         5         6         7       */
/* 456789_123456789_123456789_123456789_123456789_123456789_123456789_123456 */

/*************************************************************************/
/*                                                                       */
/* Rotate.cpp                                                            */
/* Converte as coordenadas dadas para                                    */
/* coordenadas num sistema de eixos rodado de um ângulo teta             */
/*                                                                       */
/*                                                                       */
/*                                                                       */
/* Version:    1.1                                                       */
/* Date:       2 Outubro 2007                                            */
/* Authors:    H.David Miranda                                           */
/*                                                                       */
/*************************************************************************/

#include "stdafx.h"
#include "App1.h"

#ifdef _DEBUG
#undef THIS FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif


#include <stdio.h>
#include <stdlib.h>
#include <math.h> /* sin,cos */


void Rotate(double f_teta, double **px, double **py, int nNumPt)
{
    int a;

    double f_sin=sin(-f_teta);
    double f_cos=cos(-f_teta);

    double x,y;

    /* efectua a rotacao para todos os pontos */
    for (a=1;a<=nNumPt;a++)
    {
        x = (*px)[a];
        y = (*py)[a];

        (*px)[a] = x * f_cos + y * f_sin;
        (*py)[a] = y * f_cos - x * f_sin;
    }
}

void Rotate(double f_teta, double *px, double *py)
{
    double x = *px;
    double y = *py;
    double f_sin=sin(-f_teta);
    double f_cos=cos(-f_teta);

    *px =  x * f_cos + y * f_sin;
    *py = y * f_cos - x * f_sin;
}
```

```
/*       1         2         3         4         5         6         7       */
/* 456789_123456789_123456789_123456789_123456789_123456789_123456789_123456 */

/*************************************************************************/
/*                                                                       */
/* ConcreteDiseq.cpp                                                     */
/* Calcula os esforços resistentes correspondentes no material base,     */
/* relativamente ao referencial com o eixo neutro paralelo ao eixo dos xx.   */
/* Retorna o valor 0 em caso de sucesso                                  */
/*                                                                       */
/* Version:    2.1                                                       */
/* Date:       10 Outubro 2007                                           */
/* Authors:    H.David Miranda                                           */
/*                                                                       */
/*************************************************************************/

#include "stdafx.h"
#include "App1.h"

#ifdef  DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

#include <math.h>
#include <stdio.h>

#include "Diseq.h"
#include "BaseMaterial.h"
#include "Bondary.h"
#include "Selice.h"

int ConcreatDiseq(CBondary *pBondary, CBaseMaterial *pBaseMaterial,
                  double fEpsM, double fEpsB, CDiseq *pDiseq)
{
    // largura de cada fatia em m
    double deltay = 0.001;

    // desiquilibrios = 0
    pDiseq->delta[N]=pDiseq->delta[Mx]=pDiseq->delta[My]=0;

    double sigma;
    int a;
    // calculo dos extremos da secçao
    double ymax = pBondary->y[1];
    double ymin = pBondary->y[1];
    for (a=2; a<=pBondary->nVertex; a++)
    {
        if (pBondary->y[a] > ymax)
            ymax = pBondary->y[a];
        if (pBondary->y[a] < ymin)
            ymin = pBondary->y[a];
    }

    CSelice Selice(pBondary->nEdges);

    double y;
    double yA,yB,xA,xB;

    // percorrer a secçao de cima para baixo
    for (y=ymax+deltay/2; y>=ymin; y-=deltay)
    {
        for (a=1; a<=pBondary->nEdges; a++)
        {
            yA = pBondary->y[pBondary->P[a]];
            xA = pBondary->x[pBondary->P[a]];
            yB = pBondary->y[pBondary->Q[a]];
            xB = pBondary->x[pBondary->Q[a]];

            if ( (yA>y && yB<y) || (yB>y && yA<y) )
            {
                // se a recta não e' horizontal
                if (fabs(yB-yA)>deltay*1e-6)
                {
                    // inserir ordenada do ponto de intercepcao
                    Selice.Insert(  xA + (y-yA)*(xB-xA)/(yB-yA)  );
                }
```

```
                // se a recta  e' horizontal ou
                // a intrecpcao e' proxima de um vertice
                if ( (fabs(yA-y)<=deltay*1e-6) || (fabs(yB-y)<=deltay*1e-6) )
                {
                    // excepção
                    // a maneira eficaz de resolver o problema
                    // e' dar um pequeno deslocamento a y
                    // e reiniciar a montagem da selice (fatia)
                    y += deltay*1e-6;
                    Selice.Reset();
                    a = 0; // reiniciar o ciclo controlado pela variavel a
                }
            }
        }
        double len=Selice.Length();
        sigma = pBaseMaterial->Eval(fEpsM*y + fEpsB);
        pDiseq->delta[N] += sigma * len;
        pDiseq->delta[Mx] += y * sigma * len;
        pDiseq->delta[My] -= sigma* Selice.Momentum();
        Selice.Reset();
    }
    pDiseq->delta[N] *= deltay;
    pDiseq->delta[Mx] *= deltay;
    pDiseq->delta[My] *= deltay;

    return 0;
}
```