U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

DESENVOLVIMENTO DE APLICAÇÕES COMPUTACIONAIS DIDÁTICAS NA ÁREA DAS ESTRUTURAS

MÁRIO PAULO MACHADO TEIXEIRA

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM ESTRUTURAS

Orientador: Professor Doutor Miguel Ângelo Carvalho Ferraz

JUNHO DE 2015

MESTRADO INTEGRADO EM ENGENHARIA CIVIL2014/2015

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

Fax +351-22-5081446

✉ 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 +351-22-5081440

✉ feup@fe.up.pt

http://www.fe.up.pt

Reproduções parciais deste documento serão autorizadas na condição que seja mencionado o Autor e feita referência a *Mestrado Integrado em Engenharia Civil - 2014/2015 - Departamento de Engenharia Civil, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2015*.

As opiniões e informações incluídas neste documento representam unicamente o ponto de vista do respetivo Autor, não podendo o Editor aceitar qualquer responsabilidade legal ou outra em relação a erros ou omissões que possam existir.

Este documento foi produzido a partir de versão eletrónica fornecida pelo respetivo Autor.

Ao meu Avó Damásio Teixeira

… I firmly believe that any man's finest hours – His greatest fulfillment of all that he holds dear – is that moment when he has worked his hearth out in good cause and lies exhausted on the field of battle – victorious

Vince Lombardi

AGRADECIMENTOS

É um facto que a realização desta dissertação não teria sido de forma alguma possível sem o apoio incondicional e o incentivo das pessoas enumeradas nesta pequena folha de papel.

Antes de mais e sobretudo agradeço ao meu orientador o Professor Doutor Miguel Ferraz que demonstrou ao longo da produção deste trabalho uma disponibilidade, paciência e um profissionalismo sem igual. Com a sua capacidade de compreensão e sugestões, possibilitaram criar um rumo de auxílio para a realização deste documento.

Tenho que agradecer sem dúvida a uma pessoa, que ao longo da minha vida se tem mostrado disponível para me ouvir, ajudar e incentivar, a minha irmã. Sem ela tudo seria um pouco mais cinzento.

Gostava de deixar uma palavra de apreço aos meus amigos, que apesar da minha ausência, eles mantiveram uma forte amizade comigo e sempre demonstram interesse em conviver e apoiar-me durante esta fase da minha vida, muito obrigado. Sim Dave estas incluído aqui. Tu também Brig.

Gostaria de deixar um agradecimento especial aos meus irmãos e camaradas, que apesar de já não convivermos no dia-a-dia, as experiências que vivemos juntos mudaram alguns objetivos na minha vida. Não poderia deixar de mencionar o meu grande amigo Nelson Martins que me concedeu, através da sua perseverança e ambição mas sempre com respeito e humildade, uma grande inspiração para os meus alvos na vida. A eficiência e fulgor que o meu grande amigo Robert Alves demonstrou durante o tempo em que passamos juntos, foi mais uma vez, uma inspiração para mim, sem ele não seria quem sou hoje também. Por fim, ao meu grande amigo Flávio Teixeira, que tem demonstrado ser um amigo muito leal e prestável que jamais irei esquecer. Desejo a estes meus camaradas o mais sucesso possível e que tudo de bom lhes aconteça, obrigado meus irmãos.

Agradeço também ao casal Gonçalves pela sua simpatia e auxílio que me deram durante a minha estadia na casa deles, que sem qualquer dúvida, a sua ajuda foi imensurável e sem eles esta aventura pela Faculdade de Engenharia do Porto não teria sido possível, muito obrigado.

Agradeço do fundo do meu coração há minha família, pais, avós, tios, primos, pelo grande apoio e motivação que me deram ao longo dos anos. Sem vocês não teria sido possível chegar onde cheguei nem caminhar até onde chegarei, sobretudo, á minha avó Teresa e ao meu avô Manuel, que nunca deixaram de acreditar em mim, e sempre demonstram apreço por tudo o que fiz de bom ao longo da minha vida e onde nos maus momentos me ajudaram por incentivos morais a ultrapassar os meus obstáculos, muito obrigado.

Finalmente mas não menos importante, queira deixar um agradecimento especial ao meu grande amigo de longa data, Daniel Silvério. É importantíssimo referir que hoje não estaria nesta posição caso não tivesse conhecido este senhor. O Daniel nunca deixou de ser um amigo que partilha os mesmos interesses e que me cativou a chegar mais longe. Possibilitou que sonhasse histórias e contos com que de outra forma nunca haveria de os ter sonhado. É um amigo que sem o mesmo não poderia haver *duo* em *vagrant*. Meu amigo até já, pois sem dúvida alguma, que ambos permaneceremos amigos enquanto caminharmos por esta terra e continuaremos a sê-lo muito depois de termos deixado de a caminhar.

Por fim, agradeço à Faculdade de Engenharia do Porto pelos conhecimentos que adquiri, e pela minha entrada no mundo entusiasmante da engenharia. Fizeram crescer na minha pessoa novos alvos e propósitos para o futuro, um sincero obrigado.

RESUMO

O seguinte documento apresenta um trabalho de Desenvolvimento de aplicações computacionais didáticas na área da Engenharia Civil, realizado através da utilização de uma linguagem computacional para a criação de uma ferramenta de cálculo, que visa auxiliar os estudantes a abordar e resolver diversos casos teóricos ou práticos no seu ato de aprendizagem, de forma simples, intuitiva e de fácil acesso.

O resultado do trabalho foi a criação de uma ferramenta de cálculo desenvolvida utilizado a linguagem de programação Csharp, com o auxílio de uma plataforma de programação denominada por Unity3D. A ferramenta calcula interactivamente as tensões tangenciais e normais, resultantes da aplicação de momentos e esforços axialmente aplicados em secções padronizadas como também a secções de qualquer tipo de geometrias,

Com o objectivo de criar uma ferramenta o mais intuitiva possível, os cálculos são apresentados sem transições e em tempo real, podendo o utilizador alterar a geometria de forma fácil e dinâmica, promovendo o espírito crítico por parte do utilizador, ao ver as alterações dos diagramas de tensões sem recurso a janelas extra, nem "*restarts*" do programa.

O corpo desde documento aborda a escolha da plataforma de programação anteriormente referida, sempre associada à linguagem de programação utilizada e às dificuldades inerentes da programação para a utilização em diferentes plataformas móveis. Apresenta também o processo de conceção de ferramenta tal como exemplos diversos de utilização e de cálculo.

O mesmo documento termina com uma reflexão sobre os resultados obtidos na realização deste trabalho, como também avalia as diferentes aplicações possíveis para a ferramenta desenvolvida, quer a nível académico, quer a outros níveis, comentando as suas deficiências bem como a sua potencialidade de expansão para a resolução de outros tipos de problemas.

PALAVRAS-CHAVE: Engenharia Civil, ferramenta didática, flexão composta desviada, secções de geometria variável, propriedades geométricas, tensões normais, tensões tangenciais.

ABSTRACT

The following document describes the work of developing academic computer applications in the field of civil engineering through the use of a computer language to create a calculation tool, which helps students to approach and solve a variety of theoretical or practical problems, in a simple, intuitive and easily accessible form.

The result of the work was the creation of a calculation tool developed using the CSharp programming language with the aid of a programming platform called Unity3D. The tool calculates the shear stress and plane stress as results of axial forces and biaxial bending actions applied to standard sections as well as sections of any kind of geometry.

In order to create a more intuitive tool possible, the calculations are presented seamless without transitions. The user can change the geometry easily and dynamically, thus inducing a critical view by the user when the changes occur to the diagrams of tensions in real time, without using extra windows or restarts to the program.

The body of this document addresses the choice of the programming platform mentioned above, also associated to the topic of the programming language used and the difficulties inherent in programming for the use on different mobile platforms, as well as the design process behind the program and some application examples.

This document concludes with a reflection on the results obtained in this work, and also evaluates the different possible applications for the developed tool, either academically or at another level, with a brief discussion about the tool's shortcomings, as well as the possibility of expansion to other types of problems.

KEYWORDS: Civil engineering, didactic tool, biaxial bending, sections with variable geometry, geometrical properties, normal stress, tangential stress.

ÍNDICE GERAL

AGRADECIMENTOS i
RESUMO iii
ABSTRACT v

1. INTRODUÇÃO 1
2.1. OBJECTIVOS 1
2.2. PRINCÍPIOS DE DESENVOLVIMENTO 1

2. PROGRAMAÇÃO 3
2.1. INTRODUÇÃO 3
2.2. PARADIGMA ATUAL PARA A CONCEÇÃO DA ESTRUTURA DO PROGRAMA 3
2.3. LINGUAGENS E PARADIGMAS DE PROGRAMAÇÃO 3
2.3.1. GRUPOS DE LINGUAGENS 5
2.3.2. CATEGORIA DE LINGUAGENS 5
2.3.2.1. PROGRAMAÇÃO PROCEDIMENTAL 5
2.3.2.2.PROGRAMAÇÃO MODULAR 6
2.3.2.3. ABSTRAÇÃO DE TIPOS DE DADOS 6
2.3.2.4. PROGRAMAÇÃO ORIENTADA POR OBJETOS 6
2.3.3. LINGUAGEM DE PROGRAMAÇÃO *JAVA* 8
2.3.4. LINGUAGEM DE PROGRAMAÇÃO C SHARP 8
2.4. ESTRUTURA DO PROGRAMA PARA VÁRIAS PLATAFORMAS 8
2.5. ESTRUTURA DO PROGRAMA PARA *WEBSITES* 8
2.5.1 PROTOCOLOS DE COMUNICAÇÃO 9
2.5.1.1 CATEGORIAS DE PROTOCOLOS 9
2.5.1.2. TIPO DE PROTOCOLOS 9
2.5.2. LINGUAGEM HTML 10
2.5.3. LINGUAGEM *JAVASCRIPT* 11
2.5.4. LINGUAGEM PHP 11
2.5.5. DEFININDO PROGRAMAÇÃO NA *WEB* 11
2.6. *WEB HOSTING* 12
2.7. *FRAMEWORK UNITY3D* 12
2.7.1. OBJETIVOS DE IMPLEMENTAÇÃO 13

3. PROBLEMA MECÁNICO 15

3.1. INTRODUÇÃO 15

3.2. MÉTODO DE CÁLCULO DAS FLEXÕES DE SIMPLES, DESVIADA E COMPOSTAS 15

3.2.1. TENSÃO SOB CARREGAMENTO AXIAL; PRINCÍPIO *SAINT-VENANT* 16

3.2.2. TENSÃO NORMAL 18

3.2.3. FLEXÃO COMPOSTA DESVIADA 22

3.3. TENSÕES TANGENCIAIS 26

3.4. ESTADO PLANO DE TENSÕES 30

3.4.1. PLANO DE TENSÕES PRINCIPAIS 33

3.4.2. TENSÃO MÁXIMA DE CORTE 35

3.4.3. CÍRCULO DE *MOHR* 36

4. CONCEÇÃO DA FERRAMENTA DE CÁLCULO 39

4.1. INTRODUÇÃO 39

4.2. CARACTERÍSTICAS DA FERRAMENTA 39

4.3. ESTRUTURAÇÃO DA FERRAMENTA 40

4.4. ESTRUTURA DA *INTERFACE* 41

4.4.1. ERGONOMIA DE SISTEMA MULTIMÉDIA 41

4.5. METODOLOGIA DE CÁLCULO 42

4.5.1. METODOLOGIA DE CÁLCULO PARA TENSÕES NORMAIS E FLEXÕES COMPOSTAS DESVIADAS 43

4.5.1.1. COMPUTAÇÃO DE ÁREAS, CENTRÓIDE E INÉRCIAS 43

4.5.1.2. METODOLOGIA DE DESENHO DOS DIAGRAMAS DE TENSÕES 45

4.5.2. METODOLOGIA DE CÁLCULO DAS TENSÕES DE CORTE 45

4.5.3. METODOLOGIA PARA A OBTENÇÃO DA REPRESENTAÇÃO GRÁFICA DO CÍRCULO DE *MOHR* 46

5. UTILIZAÇÃO DA FERRAMENTA DE CÁLCULO 49

5.1. INTRODUÇÃO 49

5.2. DIREÇÕES DE ABERTURA 49

5.3. APRESENTAÇÃO DA *INTERFACE* DE ESCOLHA DE TAREFAS 51

5.5. MÉTODOS PARA A DEFINIÇÃO DA GEOMETRIA NO MÓDULO DE DESENHO POLINOMIAL 53

5.6. SECÇÕES PADRONIZADAS 56

5.7. DESENHO DO CÍRCULO DE *MOHR* 59

5.8. EXPANSIBILIDADE E NOTAS 60

6. EXEMPLOS DE APLICAÇÃO 61

6.1. INTRODUÇÃO 61

6.2. EXERCÍCIO 1 – CÁLCULO DE UMA SECÇÃO EM T SUBMETIDA A FLEXÃO SIMPLES 61

6.3. EXERCÍCIO 2 – CÁLCULO DE UMA SECÇÃO CIRCULAR SUBMETIDA A FLEXÃO SIMPLES 63

6.4. EXERCÍCIO 3 – DETERMINAÇÃO DA TENSÃO DE CORTE NUMA SECÇÃO EM I 64

6.5. EXERCÍCIO 4 – DESENHO E CÁLCULO DO CÍRCULO DE *MOHR* 66

6.6. EXERCÍCIO 5 – CÁLCULO DE SECÇÕES SUBMETIDAS A FLEXÃO COMPOSTA DESVIADA 67

7. CONSIDERAÇÕES FINAIS 69

REFERÊNCIAS BIBLIOGRÁFICAS 73

ANEXO A1. PARCELA DE CODIGO DO PROGRAMA 77

ÍNDICE DE FIGURAS

Fig. 1.1 – Uma única linguagem compatível com diversos dispositivos. 2
Fig. 2.1 – Polimorfismo....7
Fig. 2.2 – Nomenclatura de uma Pagina Web 11
Fig. 3.1. – Diagrama Tensão – Extensão....15
Fig. 3.2. – Elemento sofrendo um carregamento axial por parte de forças P 16
Fig. 3.3. – Representação sintética do princípio de *Saint-Venant* 17
Fig. 3.4. – Representação da distribuição de tensões em diversas secções 17
Fig. 3.5. – Deformação de um segmento de viga infinitesimal 18
Fig. 3.6. – Representação do cálculo da resultante da tensão normal numa secção transversal 20
Fig. 3.7. – Representação da tensão máxima 22
Fig. 3.8. – Resultantes de um carregamento excêntrico 22
Fig. 3.9. – Diagrama de tensões sobre um elemento onde o eixo neutro está fora do referencial 23
Fig. 3.10. – Diagrama de tensões sobre um elemento com representação do eixo neutro 23
Fig. 3.11. – Secção transversal com simetria vertical, sujeita a um momento flector não coincidente com os eixos principais 24
Fig. 3.12. – Elemento submetido a uma força excêntrica P 25
Fig. 3.13. – Secção transversal sujeita a uma força P e a um momento M 26
Fig. 3.14 – Variação do primeiro momento Q da área A' em torno do eixo neutro 28
Fig. 3.15 – Representação da determinação da tensão de corte 29
Fig. 3.16. – Plano biaxial do estado de tensões. Componentes atuando nas suas direções positivas....30
Fig. 3.17. – Representação da transformação das coordenadas xy do estado planos de tensão, para coordenadas x'y' 31
Fig. 3.18. – Método geométrico para a determinação dos ângulos θ1 e θ2....34
Fig. 3.19 – Estados equivalentes de tensão num ponto 35
Fig. 3.20. – Círculo de *Mohr* e transformação de estados planos de tensão 37
Fig. 4.1 – Delimitação da área de um polinómio....45
Fig. 4.2 – Metodologia de determinação do diagrama de tensões 46
Fig. 4.3. – Construção do círculo através dos componentes de tensão 47
Fig. 5.1. – Página *Web* onde o programa está hospedado 49
Fig. 5.2. – Interface inicial da ferramenta de cálculo no *website*....50
Fig. 5.3. – Interface para escolha de módulos 51
Fig. 5.4. – Representação do Separador "Sobre" 51
Fig. 5.5. – Interface de cálculo para o módulo de desenho polinomial....52
Fig. 5.6. – Introdução de pontos....53

Fig. 5.7. – Inserção manual de pontos 54
Fig. 5.8. – Obtenção de resultados 54
Fig. 5.9. – Representação do diagrama de tensões 55
Fig. 5.10. – Abertura de uma secção 56
Fig. 5.11 – Cálculo de secções padronizadas, I 56
Fig. 5.12. – Cálculo da tensão de corte, para secção em I 57
Fig. 5.13. – Cálculo da tensão de corte, para secção em T, com abertura na secção 58
Fig. 5.14. - Cálculo da tensão normal e de corte 58
Fig. 5.15. – Cálculo de tensões num círculo com dois materiais 59
Fig. 5.16. – Módulo de cálculo e desenho do círculo de *Mohr* 59
Fig. 6.1. – Viga submetida a esforço axial e a flexão simples, secção transversal em milímetros 61
Fig. 6.2. – Diagrama de tensões para flexão simples e carga axial numa secção T 62
Fig. 6.3. – Resultados numéricos das tensões normais para flexão composta e carga axial numa secção T 62
Fig. 6.4. – Secção circular composta por dois materiais sujeita a flexão simples 63
Fig. 6.5. – Resultados para uma secção círculo sujeita a flexão simples 63
Fig. 6.6. – Secção I sujeita a esforço transverso e momento fletor 64
Fig. 6.7. – Resultados obtidos numa secção em I sujeita a esforço transverso e momento fletor 64
Fig. 6.8. – Resultados obtidos numa Secção I com vazio sujeita a esforço transverso e momento fletor 65
Fig. 6.9. – Estado plano de tensões de um dado ponto num elemento 66
Fig. 6.10. – Representação gráfica do círculo de *Mohr* 66
Fig. 6.11. – Detalhes das tensões principais representadas no círculo de *Mohr* 67
Fig. 6.12. – Flexão composta desviada sobre uma secção genérica 67
Fig. 6.13 – Resultados obtidos numa secção genérica sujeita a flexão composta 68

ÍNDICE DE TABELAS

Quadro1 – Tipos de linguagens de programação.....4

SÍMBOLOS, ACRÓNIMOS E ABREVIATURAS

A – Área

A’ – Área parcial de uma secção

b – Largura

c – Distancia do eixo neutro a fibra mais extrema

d – Distancia

E – Modulo de elasticidade

F – Força

H – Força horizontal

h – Altura

I – Momento de inércia

M – Momento fletor

Mx – Momento fletor relativamente ao eixo dos xx

My – Momento fletor relativamente ao eixo dos yy

N – Esforço Axial

P – Força

Q – Primeiro momento da área

R – Força resultante; Raio

T – Força de torção

t – Espessura

V – Esforço Transverso

x, y, z – Coordenadas retangulares

x’, y’, z’ – Coordenadas do centroide; Centro de gravidade

α,β,θ – Ângulos

ρ – Raio de curvatura

ε – Extensão

σ - Tensão Normal

$\sigma 1, \sigma 2$ – Tensões principais

τ – Tensão de corte

1
INTRODUÇÃO

1.1. OBJECTIVOS

O seguinte documento descreve um trabalho realizado no âmbito do mestrado integrado em engenharia civil no ramo de especialização em estruturas. Tal como o tema da dissertação indica, este trabalho consistiu no desenvolvimento de aplicações computacionais didáticas não apenas no ramo das estruturas, mas em todo o espectro da Engenharia Civil, através da utilização de uma linguagem computacional para a criação de uma ferramenta de cálculo, que permitisse auxiliar os estudantes a abordar diversos casos teóricos ou práticos no seu ato de aprendizagem.

Sendo um dos objectivos pretendidos, a indução nos utilizadores da aplicação um espírito dinâmico e um raciocínio critico perante os resultados apresentados, com a possibilidade de alterar os dados de forma fácil e rápida, sem ser necessário o auxilio de interface extras.

1.2. PRINCÍPIOS DE DESENVOLVIMENTO

Tal como mencionado no ponto anterior foi desenvolvido através de um código computacional, mais precisamente na linguagem *CSharp*, uma pequena ferramenta de cálculo didático com o objetivo de aplicar os métodos e técnicas ensinados aos alunos de Engenharia Civil. De forma a obter uma interface simples, direta e intuitiva de forma dinâmica e interativa. Pretendendo-se que a ferramenta funcione em diversas situações, nomeadamente:

- Computador *off-line*;
- Dispositivos moveis *off-line*;
- Pagina Web acedida através de:
    i. Dispositivos de *hardware* fixos;
    ii. Dispositivos de *hardware* moveis;
- Compatibilidade com *IOS* e *Android*;

De forma a permitir que a ferramenta fosse utilizada por qualquer tipo de dispositivo, com ou sem acesso à internet, foi utilizado uma interface de programação com um grande leque de compatibilidade de formatos e linguagens de programação pelo nome de *Unity3D*, a qual será abordada com maior detalhe no capítulo dois. Em auxílio à ferramenta, foi de igual modo criado um *website* no qual está inserida a ferramenta no próprio design da página, com o propósito de ser consultado para a utilização da mesma, por outros tipos de dispositivos informáticos que sejam capazes de aceder ao programa sem que seja necessário o seu *download* e a sua instalação.

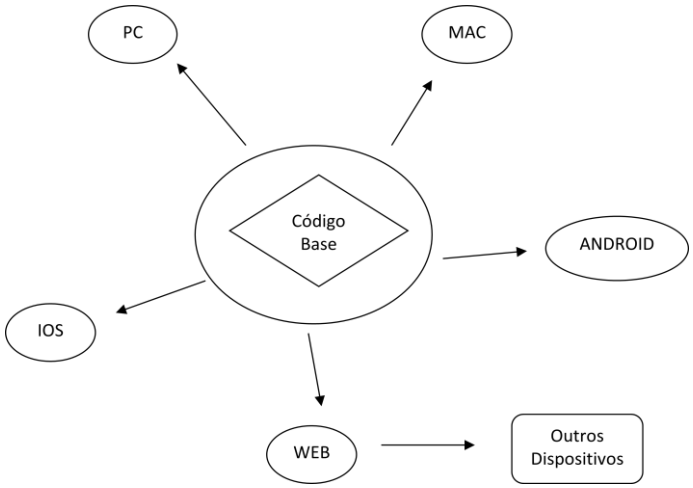


Fig.1.1 – Uma única linguagem compatível com diversos dispositivos

Para que o programa fosse acessível sem download no website, foi utilizador a tecnologia de WebGL, que se trata de uma interface de programação de aplicações em JavaScript a partir do elemento de delimitação de áreas para renderização dinâmica de gráficos do HTML5. O WebGL permite que seja implementado em uma aplicação sem a necessidade de plug-ings no navegador.

O desenvolvimento da ferramenta focou-se desta forma no cálculo de tensões normais e tangenciais em secções padronizadas e na forma mais genérica de qualquer tipo de sujeitas a flexões simples ou a flexões compostas desviadas. É de referir a utilidade de facilitar e resolver os problemas mais convencionais no âmbito do ensino académico, no entanto, a facilidade de alteração geométrica das secções constitui por si só ser uma ferramenta de grande utilidade para ser aplicada na prática de Engenharia Civil. Outro aspeto a destacar é a capacidade de ser acessível em qualquer ponto do mundo a qualquer altura.

Estas foram as bases de conceção para a obtenção de uma ferramenta autónoma com grande versatilidade que fosse o mais intuitiva possível no cálculo orgânico de secções e que apresentasse uma interatividade simples, com uma apresentação direta dos resultados obtidos, permitindo induzir nos seus utilizadores um raciocínio crítico, alterando a geometria ou as solicitações aplicadas de forma instantânea.

2
PROGRAMAÇÃO

2.1. INTRODUÇÃO

O seguinte capítulo pretende ilustrar as razões que levaram à utilização da plataforma de programação*Unity3D*, bem como as principais vantagens e desvantagens de cada tipo de linguagem aplicadas à interface estrutural, e para o desenvolvimento de uma ferramenta no paradigma atual das interfaces móveis e dos vários sistemas operativos em questão, ilustrando os mecanismos que levaram à sua seleção tanto a nível de programação de raiz como a nível de realização de *websites*.

2.2. PARADIGMA ATUAL PARA A CONCEÇÃO DA ESTRUTURA DO PROGRAMA

Hoje em dia assistimos cada vez mais à generalização das redes e dos aparelhos móveis (telemóvel, computador portátil, *tablet*), e dessa forma, a maneira de aceder e interagir com a informação também tem sofrido alterações de modo a se moldar com a tendência atual. Nesta fase de crescimento dos sistemas computadorizados e com a ampliação do uso de tecnologias de transmissão de dados através de redes sem fio, e com a generalização do uso da internet, foi contemplado mediante o tema do desenvolvimento de aplicações didáticas, a criação de uma ferramenta (ou aplicativo) passível de ser utilizada nos principais sistemas operativos, podendo a mesma ser acedida através de qualquer aparelho móvel com ou sem internet.

Visando este objeto, foi estabelecido a integração da ferramenta como uma aplicação acedida através de um *website* ou carregamento como um aplicativo *offline*. Desta forma, o aplicativo poderá ser utilizado em qualquer plataforma, em qualquer sítio, com ou sem acesso à internet, num computador fixo ou num dispositivo móvel.

Por conseguinte, são descritas várias técnicas e metodologias de programação, as quais são essenciais no que diz respeito ao enquadramento do desenvolvimento da dita ferramenta de cálculo, ou noutras palavras do programa, pretendendo-se expor uma perspetiva informativa sobre os métodos utilizados para a sua conceção.

2.3. LINGUAGENS E PARADIGMAS DE PROGRAMAÇÃO

Por definição, a linguagem de programação é definida por um sistema de comunicações que utiliza um conjunto de instruções e regras de sintaxe. Segundo a execução de uma sequência lógica de operações, estas podem ser executadas por um programa seguindo um conjunto finito de regras através das quais se pode dar execução a um dado processo com bases sintáticas e semânticas (denominado por algoritmo), que comunica com dispositivos de *hardware* e

*software*as instruções do programador e desta forma possibilita a comunicação entre o algoritmo escrito pelo programador e o computador [5].

A linguagem utilizada pelo processador consiste numa série de dados binários (valores de 0 e 1), não sendo ela compreensível para os utilizadores humanos, e por essa razão foram desenvolvidas linguagens intermediárias compreensíveis para o homem. O código escrito nestes tipos de linguagens é transformado em código binário para que o processador o consiga processar. No entanto, cada tipo de processador pode ter a sua própria linguagem de processamento. O termo de portabilidade é usualmente utilizado nestes casos para descrever a capacidade de utilizar um programa de *software* em diferentes tipos de equipamentos. Por vezes, para utilizar um programa de *software* escrito em determinado código de programação ou noutro equipamento, é necessário reescrever todo o programa [5].

Quadro 1 – Tipos de linguagens de programação

Linguagem	**Paradigma de programação**	**Categoria**	**Principais áreas de aplicação**
Fortran	Não estruturada	Linguagem Compilada	Cálculo
Cobol	Não estruturada	Linguagem Compilada	Gestão
Basic	Não estruturada	Linguagem Interpretada	Programação para fins educacionais
Pascal	Procedimental	Linguagem Compilada	Educação
C	Procedimental	Linguagem Compilada	Programação de sistemas
Modula *II*	Modular	Linguagem Compilada	Computação simbólica
ADA	Abstração de Tipos de Dados	Linguagem Compilada	*Software* de tempo real
Java	Programação Orientada por Objetos	Linguagem Intermédia	Programação orientada para internet
C++	Programação Orientada por Objetos	Linguagem Compilada	Programação de sistema orientado para objeto

Assim sendo, perante a relevância do tema discutido é imperativo que a linguagem de programação escolhida permita criar um algoritmo com a maior portabilidade possível, mas que permita também ao programador estruturar o programa de forma a conseguir traduzir a resolução de problemas de elevado grau de complexidade, em vários estilos de programação.

2.3.1. GRUPOS DE LINGUAGENS

As linguagens de programação são geralmente divididas em dois grupos principais com base no processamento dos seus comandos.

- Linguagem de programação imperativa. Uma linguagem imperativa, programa mediante uma série de comandos, agrupados em blocos e compostos de ordens condicionais que permitem ao programa retornar a um bloco de comandos caso se cumpra a condição.

As linguagens imperativas carecem de flexibilidade devido à sequência das instruções, ainda por demais este princípio foi utilizado pelas primeiras linguagens de programação e ainda hoje em dia é utilizado pelas linguagens modernas.

- Linguagem de programação funcional. Trata-se de uma linguagem que cria programas mediante funções (por vezes, denominada por linguagem procedimental), devolve um novo estado de resultados e recebe como entrada o resultado das outras funções.

2.3.2. CATEGORIA DE LINGUAGENS

Neste aspeto, as linguagens de programação podem ser divididas em duas categorias, a linguagem interpretada e a linguagem compilada.

- Linguagem interpretada. Como já foi mencionado anteriormente a linguagem de programação é distinta do código de máquina. Este tem que ser traduzido para que o processador perceba o código. Um programa escrito numa linguagem interpretada requer um programa adicional, ou seja, um programa que atue como o "intérprete" e traduza os comandos conforme necessário.

- Linguagem compilada. Um programa escrito numa linguagem compilada é traduzido por um programa adicional chamado compilador, o que cria um novo arquivo autónomo que não requer qualquer outro programa para se executar a si mesmo, esse ficheiro é chamado de executável. A vantagem de não necessitar de um programa adicional para executá-lo uma vez que tenha sido compilado reflete-se na compilação, como a tradução só precisa de ser feita uma vez a sua execução é muito mais rápida.

- Línguas intermédias. Algumas línguas pertencem a ambas as categorias. Um programa escrito nestas linguagens pode em alguns caso passar por uma compilação intermédia [6].

2.3.2.1. PROGRAMAÇÃO PROCEDIMENTAL

A programação modular, permite através da estruturação do programa em procedimentos mais elementares, faz com que o modelo procedimental reduza a complexidade dos algoritmos. Cada função possibilita a realização de subtarefas do programa, que por sua vez é registado de forma menos complexa, visto que existe a possibilidade de que esse procedimento invoque outros procedimentos para executar outras funções. Em suma, o programa total é registado de forma simplificada com base em procedimentos, que por sua vez, são registados com base noutros procedimentos, e assim progressivamente. Este processo é assim organizado em sub-algoritmos que executam funções de complexidade reduzida [5].

2.3.2.2.PROGRAMAÇÃO MODULAR

A programação modular baseia-se em princípios de encapsulamento de dados. Os módulos de funcionamento que decompõem o programa, para além de serem estabelecidos por dados e procedimentos, são formados para que uma enorme porção desses dados não possa ser acedida diretamente, a não ser pelo próprio módulo. Este procedimento permite uma maior simplicidade e eficiência na preparação dos dados, bem como na produção de tarefas.

De igual modo, a obtenção dos dados de cada módulo na maioria das vezes, só pode ser cumprida por meio de uma interface organizada por um agrupado de procedimentos apropriados [5].

2.3.2.3. ABSTRAÇÃO DE TIPOS DE DADOS

De acordo com este método, o programador é capacitado para criar módulos de dados abstratos com as propriedades pretendidas. Porém, para atingi-lo é necessário que se deva definir esse género de dados recorrendo a outros dados, assim como a um conjunto completo de operações que visam esse género de dados. Na conceção de um programa baseado neste paradigma, começa-se por definir os tipos de variáveis mais adequados ao objetivo, munindo-os do conjunto de operadores e métodos de acesso que mais se lhe ajustem, fazendo ênfase na definição de tipos de objetos e na criação posterior de objetos desses tipos [5].

2.3.2.4. PROGRAMAÇÃO ORIENTADA POR OBJETOS

No que diz respeito à programação orientada por objetos, este consiste na estruturação do programa em categorias. Essas categorias podem ser definidas por um conjunto completo de operações, convertendo essas características explícitas e comuns nas diversas categorias existentes. Poderemos indicar que estas características comuns são efetuadas por meio de organismos de legado e polimorfismo. Porém, poderá existir o facto de não se estabelecer características comuns entre as categorias, mas neste caso, o paradigma degenera na abstração de dados.

Poderemos concluir que a programação orientada por objetos não só manifesta uma metodologia inovadora de estruturação de programas, como também se aproxima das atividades humanas [5].

PARADIGMAS DE PROGRAMAÇÃO

Um paradigma de programação é definido por um conjunto de estratégias e estilos que são utilizados para realizar a estrutura do programa. Como visto anteriormente no Quadro 1, existem vários paradigmas da programação. Antes de os enunciar é necessário explicitar alguns conceitos.

- Objeto. Um objeto nos termos em questão é a encapsulação de dados que representam uma entidade física no mundo real, juntamente com as operações para manipular dados. Objetos interagem com outros objetos para fornecer ou adquirir serviços. Um objeto pode ser definido como uma instância de classe [8].

- Classe. Uma classe é um modelo para objetos, sendo os mesmos considerados como instâncias de uma classe. Assim sendo os objetos que partilham a mesma funcionalidade e características podem ser agrupados numa classe [10].

- Encapsulamento. Os dados de um objeto só podem ser acedidos pelas suas operações, isto é, o encapsulamento é uma embalagem que compreende estado e comportamento. O estado fica acessível apenas através de operações invocadas por mensagens. A interface externa do objeto é separada da sua implementação:
  - Pode mudar-se a implementação sem alterar a interface.
  - Ajuda a reduzir o efeito das alterações ao código.
  - Quem usa o objeto não precisa de conhecer os detalhes de implementação [9].

- Herança. Trata-se de um mecanismo de partilha de comportamento onde uma subclasse herda as variáveis e os métodos da superclasse. As instâncias da subclasse são idênticas às instâncias da superclasse exceto nas diferenças estabelecidas pela definição da subclasse. Uma classe pode:
  - Acrescentar variáveis.
  - Acrescentar métodos.
  - Redefinir métodos. [9]

- Polimorfismo. O polimorfismo permite implementar sistemas mais facilmente extensíveis, sendo que os programas podem ser escritos para processar todos os objetos de uma hierarquia de forma genérica. Ele é assim utilizado para descrever a habilidade de enviar mensagens para diversos objetos, que por sua vez, irão lidar com a informação enviada da sua própria maneira especial.

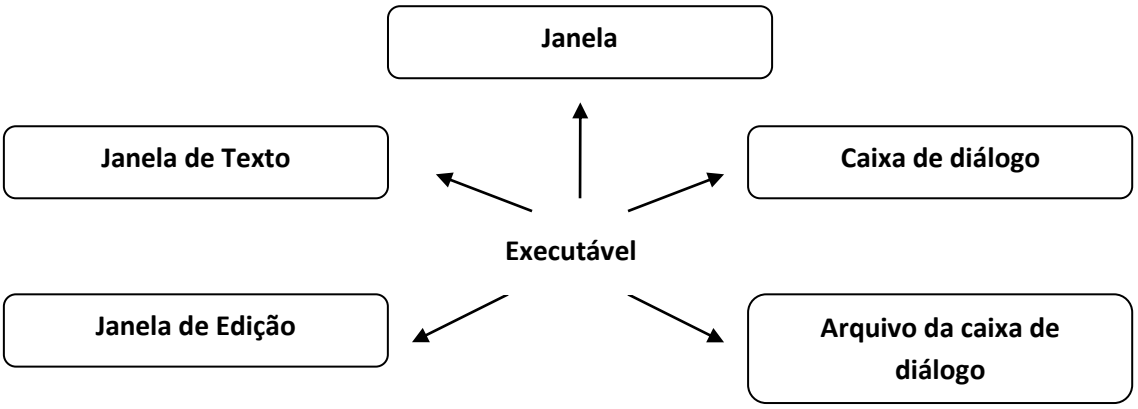


Fig.2.1 - Polimorfismo

Os detalhes da resposta de um objeto para uma mensagem genérica são subjetivos ao objeto.

2.3.3. LINGUAGEM DE PROGRAMAÇÃO *JAVA*

A linguagem de programação em *Java* é orientada para a produção de aplicações para intranet e para a *World Wide Web* (WWW), sendo a mesma compilada para um código intermédio, interpretado pelas máquinas virtuais de *Java* existentes para cada plataforma. Assim sendo, *Java* é uma linguagem orientada a objetos. Os programas que utilizam nesta linguagem de programação têm como base a manipulação de classes, objetos, atributos e métodos. [9]

2.3.4. LINGUAGEM DE PROGRAMAÇÃO C SHARP

C# é derivado da linguagem de programação C, semelhante ao *Java* é uma linguagem orientada por objetos e não oferece variáveis ou funções globais, tudo é envolvido num sistema de classes, suportando vários tipos de polimorfismo, e permite a separação de interfaces das implementações [12]. Estas características, combinadas com poderosas ferramentas de desenvolvimento de suporte de multiplataforma, fazem com que o C# seja uma boa escolha para diversos tipos de desenvolvimento de *software*, incluindo projetos de desenvolvimento rápido de aplicações, aplicações de internet, e projeto com rigorosos requisitos de complexidade. Devido à sua capacidade de compatibilidade e flexibilidade, esta linguagem de programação foi escolhida e utilizada para concretizar a programação da ferramenta de cálculo em discussão neste documento.

2.4. ESTRUTURA DO PROGRAMA PARA VÁRIAS PLATAFORMAS

É de certa forma argumentável que o conteúdo é mais importante que o aspeto, mas isto não significa que a apresentação é dispensável. Quando se analisa a conceção de uma estrutura para multiplataformas este tema torna-se pertinente, visto que subscreve uma necessidade de corresponder às expectativas do utilizador da ferramenta, na sua representação visual como no uso de figuras e diagramas "limpos", sem muito ruído visual, tal como a apresentação de informação tem que ser feita de forma intuitiva e direta, no qual o utilizador se encontre num ambiente familiar em que lhe permita explorar e reunir informações. Um grande desafio em projetar uma aplicação para diferentes plataformas é sobre a definição de *layouts*, ou seja, a representação visual da ferramenta, se se considerar, por exemplo, o tamanho visual da janela de um monitor de um computador para o ecrã de um telemóvel, existe a possibilidade do utilizador do telemóvel ter que se deslocar para cima e para baixo na janela do navegador para encontrar uma imagem ou um extrato de informação.

O principal objeto em relação a este tema será sempre a antecipação destes desafios e projetar a ferramenta de modo a que a informação projetada seja apresentada de forma apelativa e intuitiva, com transições suaves entre as secções de texto e gráficas para guiar suavemente os olhos do utilizador para as áreas de interesse da ferramenta.

2.5. ESTRUTURA DO PROGRAMA PARA *WEBSITES*

Pretendendo-se projetar a interface de utilizador para uma página *Web*, pode parecer de certa forma supérfluo, pelo facto que a interface de utilizador para uma página da *Web* esteja em grande parte associada ao fornecimento da interface pelo *Web browser*, e definindo o *design* para uma página que esteja a ser acedida por dispositivo de *hardware* fixos. As definições de monitores são pouco divergentes e existe de certa forma uma padronização de *design*.

Agora se for considerado que cada plataforma de *hardware* sem fios tem um navegador diferente lançado por parte do fornecedor da plataforma, pode-se estar a lidar com pelo menos três navegadores diferentes escritos por concorrentes diferentes. Seguindo este raciocínio, os

navegadores dos dispositivos móveis tendem a manter a sua interface simples, a fim de fazer o melhor uso possível do espaço de ecrã limitado. Da mesma forma, o conteúdo *wireless* exige projetos simples de páginas *Web*, apesar que hoje em dia e cada vez mais se sinta uma evolução a nível de processamento da plataformas de *hardware* que permitam um desenvolvimento muito mais robusto de páginas *Web*. Existem exemplos com o *wireless application protocol*, onde o navegador em telemóveis não tem interface própria móvel, e em muitos destes casos o navegador domina todo o visor. É necessário criar uma página que se adapte a ambas estas realidades e que não cause transtorno ao seu utilizador. Para projetar uma página é necessário adquirir alguns conhecimentos básicos, que serão descritos nos próximos pontos.

2.5.1 Protocolos de Comunicação

A linguagem que um computador utiliza para comunicar com outros computadores não possui correlação nenhuma com a linguagem de programação mencionada anteriormente, ela é, no entanto, denominada por protocolo de comunicação. Assim sendo, um protocolo é um método padrão que permite através de um conjunto de regras e procedimentos, a comunicação entre processos que poderão estar a ocorrer simultaneamente por computadores diferentes e desta forma enviar e receber dados através de uma rede [11].

2.5.1.1 Categorias de Protocolos

Os protocolos definem exclusivamente como devem comunicar os computadores entre si, no formato e na sequência de dados que vão trocar uns com os outros. As especificações dos protocolos nunca são exaustivas, pelo que é comum que as implementações estejam sujeitas a uma determinada interpretação das mesmas. Assim sendo, os protocolos são geralmente classificados em duas categorias segundo o controlo de dados requerido.

Protocolos sem Conexão

Este tipo de protocolos permite que exista uma comunicação por parte de um computador sem que o mesmo notifique o computador recetor, isto é, trata-se de um método que envia dados à entidade recetora sem a notificar, sendo que a mesma recebe os dados sem enviar um aviso de receção ao remetente. Neste método de comunicação, os dados são enviados em forma de blocos de dados.

Protocolos com Conexão

Estes protocolos controlam a transmissão de dados durante uma comunicação estabelecida entre dois computadores, isto é, o computador recetor envia um aviso de receção durante a comunicação pelo qual o computador emissor é responsável pela validade dos dados enviados. Neste método os dados são enviados como fluxo de dados.

2.5.1.2. Tipo de Protocolos

Conforme se espera realizar a comunicação, diversos tipos de protocolos podem ser utilizados. Alguns protocolos especializam-se no intercâmbio de arquivos, onde outros se especializam na administração do estado da transmissão. Os protocolos utilizados pertencem a uma série de protocolos ou um conjunto de protocolos inter-relacionados, sendo que este conjunto é

denominado por TCP/IP. Entre outros, estes são alguns protocolos de maior relevância para o documento em questão:

- HTTP – *Hyper Text Transfer Protocol*, define como as mensagens são formatadas e transmitidas, determinando quais as ações que os *Webservers* e os *browsers* devem tomar em resposta a vários comandos.

- FTP - *File Transfer Protocol*, é um protocolo utilizado na troca de um ficheiro na internet. É utilizado mais comummente em *downloads* de ficheiros de um servidor utilizando a internet, ou a carregar informação a um servidor.

- IP - *Internet Protocol*, especifica o formato dos pacotes de informação, por si só funciona como um serviço de correios que permite enviar pacotes ao sistema.

- TCP – *Transmission Control Protocol*, define dois *hosts* que estabelecem a conexão e a troça de informação. Este protocolo garante que a transmissão de informação e de pacotes seja recebida pela mesma ordem pela qual foi enviada.

- SMTP - *Simple Mail Transfer Protocol*, permite o envio de mensagens entre servidores.

- UDP – *User Datagram Protocol*, corre por cima dos protocolos IP, e oferece uma maneira direta de enviar e receber informação.

- ARP – *Address Resolution Protocol*, permite converter um endereço IP num endereço físico.

- ICMP - *Internet Control Message Protocol*, suporta pacotes de erros, de controlo e de mensagens informativas [9].

Tal como foi mencionado anteriormente, um protocolo define apenas o formato e a sequência de dados que devem ser trocados. Por outro lado, um protocolo não define como se programa o *software* para que ele seja compatível com o protocolo, a isto se denomina a implementação ou a conversão de um protocolo a uma linguagem de programação. Mais sobre este tópico será abordado no Capítulo 3.

2.5.2. LINGUAGEM HTML

HTML é a abreviatura da expressão inglesa *Hyper Text Markup Language*. Trata-se de uma linguagem de marcação e não de uma linguagem de programação, ou seja, ela é constituída por um código que delimita conteúdos específicos segundo uma sintaxe própria. Mais especificamente, este código possibilita a criação de páginas na *Web*, sendo o seu principal objetivo a publicação de conteúdo na *Web*, que seja por marcação de texto, imagem, vídeo, áudio ou outras formas de representação gráfica de conteúdo [10].

2.5.3. LINGUAGEM *JAVASCRIPT*

O *JavaScript* é uma linguagem de programação interpretada, isto é, é uma linguagem que permite que os *scripts* pudessem ser executados do lado do cliente sem que este passe pelo servidor. O seu uso primário é escrever funções que estão incluídas em páginas HTML.

Algumas das potencialidades do *Javascript* são:

- Gerar HTML;
- Reagir a ações do visitante;
- Gerar conteúdos dinâmicos;
- Criar efeitos gráficos.

Este assunto é abordado em maior detalhe no ponto 2.5.6. [9].

2.5.4. LINGUAGEM PHP

PHP ou *Hyper text Preprocessor*, é uma linguagem interpretada. Esta linguagem é usada para o desenvolvimento de aplicações presentes no lado do servidor, capazes de gerar conteúdo dinâmico, sendo utilizada em documentos HTML. Mais detalhe no ponto adiante [9].

2.5.5. DEFININDO PROGRAMAÇÃO NA *WEB*

Ao longo do tempo foram sendo adicionadas novas funcionalidades no HTML, uma delas foi a criação de documentos inteligentes, que tinha como funcionalidade exibir conteúdos conforme uma base de dados. A partir destas necessidades, algumas linguagens de programação passaram a ser utilizadas, como são os exemplos mais comuns de PHP, JAVASCRIPT.

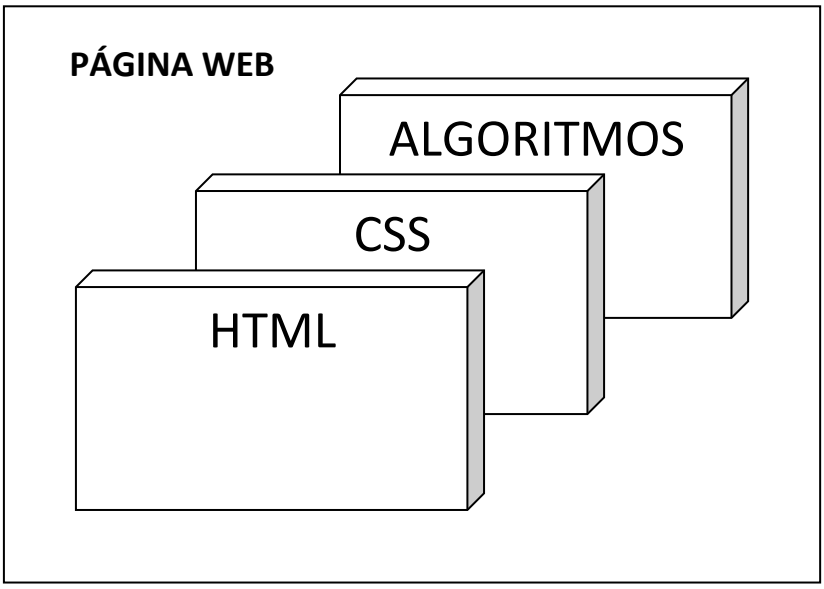


Fig.2.2 – Nomenclatura de uma Pagina Web

Para se clarificar um pouco da questão, pode-se realizar o seguinte exercício de subdividir a *Web* em três camadas, a primeira é a responsável pelo conteúdo, e que contém o HTML. A segunda camada é responsável pela apresentação do documento, e que se encontram nestas camadas os códigos em CSS (linguagens de folha de estilo), a qual permitem controlar a aparência de elementos HTML. A terceira camada é a responsável pelo comportamento da página, nesta camada encontra-se toda a lógica e programação. Ou seja, as linguagens de programação ficam responsáveis pelo desenvolvimento de algoritmos e funções, já o HTML fica responsável pela formação e estruturação do documento em si.

2.6. *WEB HOSTING*

O *Web hosting* refere-se à hospedagem de um *website* ou seja, da sua publicação *on-line*. Para tal ser realizado e a fim de publicar o *Website* é necessário subscrever um serviço de hospedagem (ou *Web Hosting*). O nome do domínio é ligado a um endereço de IP que aponta para um computador específico, como por exemplo, fe.up.pt. Quando o utilizador colocar o nome do domínio no seu navegador, o endereço de IP é localizado e o *website* é carregado a partir da hospedagem *web*. Um serviço de hospedeiro pode ter entre um a milhares de computadores que executam *software Web*, tais como *Apache*, *OS X Server*, *Windows Server*, sendo que a maioria dos *websites* são acedidos a partir de um servidor partilhado, o que representa um único computador que hospeda vários *websites*.

Websites com elevado tráfego necessitam de um servidor dedicado para suster o tráfego intenso que o mesmo possa obter, dessa forma, é necessário realizar uma análise crítica das características do *website* a construir, tais como:

- Qual será a carga de tráfego esperada, e qual o montante de largura de banda que o *website* irá utilizar.
- O *host* oferece um bom suporte técnico e garante pouco tempo de inatividade.
- Qual será o espectro temporal de hospedagem, em relação à longevidade necessária para a realização do projeto.
- Quais são as ferramentas suportadas por cada plataforma que se pretende utilizar no desenvolvimento do *website*, no ponto de vista da orientação que se ambiciona alcançar.

Atendendo a estas questões, foi utilizado um serviço de hospedagem *Apache*, desenvolvido num domínio com finalidade exclusivamente académica, bem como para fins meramente ilustrativos.

2.7. *FRAMEWORK UNITY3D*

A plataforma de programação *Unity3D* pode também ser definida por uma interface de *design*, ou, por outras palavras, uma plataforma de desenvolvimento de *software*. A sua principal vantagem é o seu grande leque de compatibilidade de formatos, tanto a nível de formatos de texto, imagem, áudio e vídeo como de linguagens de programação. Demais, trata-se de uma plataforma completamente grátis de utilização.

No que diz respeito ao espectro deste documento e das intenções de criar uma ferramenta acessível para múltiplas plataformas, esta plataforma de programação permite uma programação flexível e eficiente para multi-plataformas, ou seja, permite uma programação utilizando *JavaScript* e C# em uníssono, sem restringir qualquer que seja a linguagem de programação que se deseje utilizar apesar de se poder tornar um pouco desorganizado caso se utilize várias linguagens. No entanto, o ambiente de trabalho é intuitivo na utilização de *scripts* podendo ser utilizado pelo sistema operativo *Windows*, *Mac* e *Linux*. Uma ferramenta importante

incorporada na plataforma é o *WEBGL*, o que permite visualizar o conteúdo criado com o *Unity* diretamente no navegador com relativa acessibilidade, suportando vários motores como o *Internet Explorer*, *Mozilla Firefox*, *Google Chrome*, *Safari* e *Opera*, utilizando *HTML5*, sendo que este ultimo é a quinta versão da linguagem HTML.

2.7.1. OBJETIVOS DE IMPLEMENTAÇÃO

Tal como mencionado anteriormente, a intenção seria de criar uma ferramenta versátil que pode ser acedida através de:

- Computador *off-line*;
- Dispositivos Móveis *off-line*;
- Página *web* acedida através de:
    i) Dispositivos Fixos;
    ii) Dispositivos Móveis;
- Compatibilidade com IOS e *Android*.

Tendo em vista a criação da ferramenta para todos estes tipos de formatos, torna-se deveras pouco eficiente programar diferentes tipos de versões para diferentes tipos de dispositivos, não apenas isso, como seria necessário utilizar outras linguagens de programação, como por exemplo, *Java*, *C Sharp* e *X-Code*.

A grande capacidade de portabilidade entre formatos que a plataforma de programação *Unity3D* possui, foi o motivo principal da sua escolha. Graças à flexibilidade da plataforma foi possível criar um fluxo de trabalho com uma velocidade e eficácia que caso contrário não teria sido possível. A particularidade da comunicação entre objetos demonstra que a utilização de C# para a programação em várias plataformas é impossível sem o auxílio de programas de exportação pouco fiáveis e passíveis de conter erros. Nesta particular plataforma de programação é necessário a realização de uma programação orientada especificamente para a mesma, algo que pode tornar a curva de aprendizagem mais árdua, no que diz respeito ao *know-how* (conhecimento especializado) de algumas nuances características deste *Framework*.

Assim sendo, esta plataforma de programação permite que apenas com uma única linguagem de programação, nesta instância o *C Sharp*, se utilize uma programação generalizada para todos os componentes mencionados no início deste ponto, o que se trata sem dúvida de uma mais-valia de grande significância.

3
PROBLEMA MECÁNICO

3.1. INTRODUÇÃO

Este capítulo destina-se a enunciar e abordar os conceitos teóricos dos métodos de cálculo utilizados para a obtenção do estado de tensões e o respetivo desenho do Círculo de *Mohr* em secções planas sujeitas a flexão composta desviada.

3.2. MÉTODO DE CÁLCULO DAS FLEXÕES DE SIMPLES, DESVIADA E COMPOSTAS

Antes de mais é necessário referir que a maioria das estruturas de engenharia é dimensionada para suportar deformações relativamente pequenas, no que diz respeito à parcela linear do diagrama tensão – deformação como pode ser visionado na Figura 3.1. A fase inicial do diagrama, ou seja, a fase elástica representa o limite de proporcionalidade (atá ao ponto P) entre a tensão σ e a extensão ε, denominada essa relação por Lei de *Hooke* e podendo se escrever da seguinte forma.

$$\sigma = E \cdot \varepsilon \tag{3.1}$$

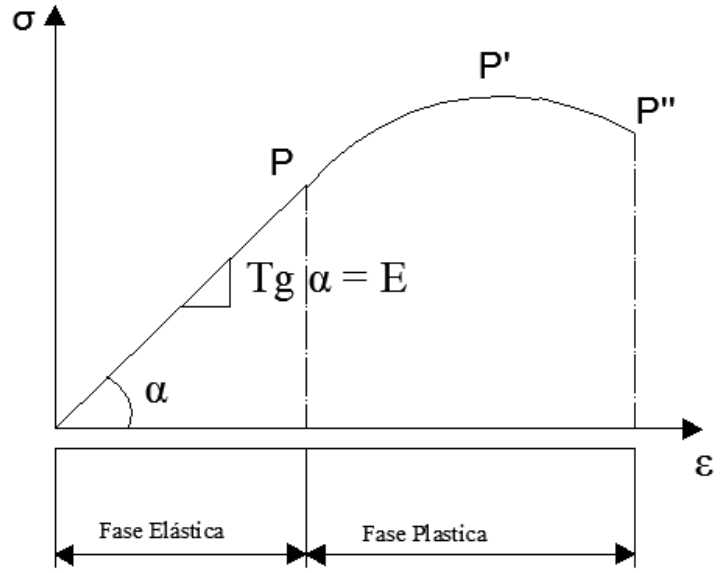


Fig.3.1. – Diagrama Tensão – Extensão; P – Limite de proporcionalidade; P' – Limite de Resistência; P'' – Limite de Rutura

3.2.1. TENSÃO SOB CARREGAMENTO AXIAL; PRINCÍPIO *SAINT-VENANT*

Se considerarmos a Figura 3.2., este elemento sofre um carregamento utilizando duas placas rígidas para transmitir cargas para o mesmo, supondo também que o elemento é constituído por um material homogéneo e isotrópico.

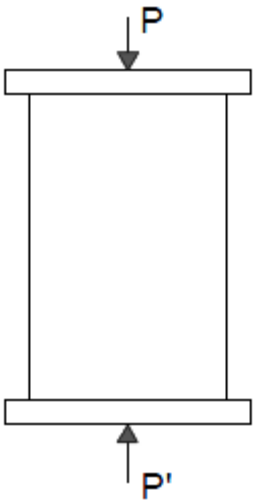


Fig.3.2. – Elemento sofrendo um carregamento axial por parte de forças P

Atendendo que as forças aplicadas no elemento são centralizadas e que as placas aproximam-se uma da outra sem rodarem, irá, por conseguinte, dar origem a um encurtamento por parte do elemento, pelo que a sua espessura e largura aumentarão. Admitindo que o elemento durante a deformação permanece retilíneo e que as secções transversais mantêm-se planas e paralelas entre si, as condições de extremidade mantêm a compatibilidade. Considerando a deformação idêntica por parte das secções transversais pode-se obter uma distribuição uniforme de formações no elemento, ou seja, as extensões longitudinais ε_z e lateral,

$$\varepsilon_x = -\nu \cdot \varepsilon_z \tag{3.2}$$

são constantes. Agora se for considerado que as tensões não excedem o limite de proporcionalidade, aplica-se a lei de *Hooke*,

$$\sigma_z = E \cdot \varepsilon_z \tag{3.3}$$

pelo que é possível concluir que a tensão normal σ_z também é constante. Portanto, a distribuição de tensões é uniforme ao longo do elemento e em cada ponto,

$$\sigma_y = (\sigma_y)_{med} = \frac{P}{A} \tag{3.4}$$

Por consequência, se as cargas forem concentradas como indica a Figura 3.3., os elementos na proximidade imediatas dos pontos de aplicação das cargas ficam submetidos a tensões elevadas, enquanto os outros elementos na proximidade das extremidades da barra não são afetados pelas cargas. Ao se considerar secções mais afastadas das extremidades, é possível notar uma progressiva igualização das deformações, ou seja, uma distribuição de deformações e tensões quase uniforme na secção transversal [2].

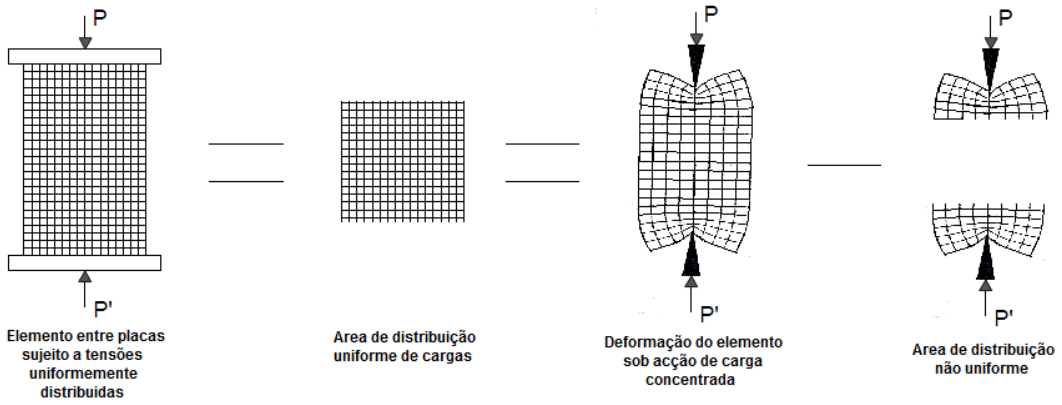
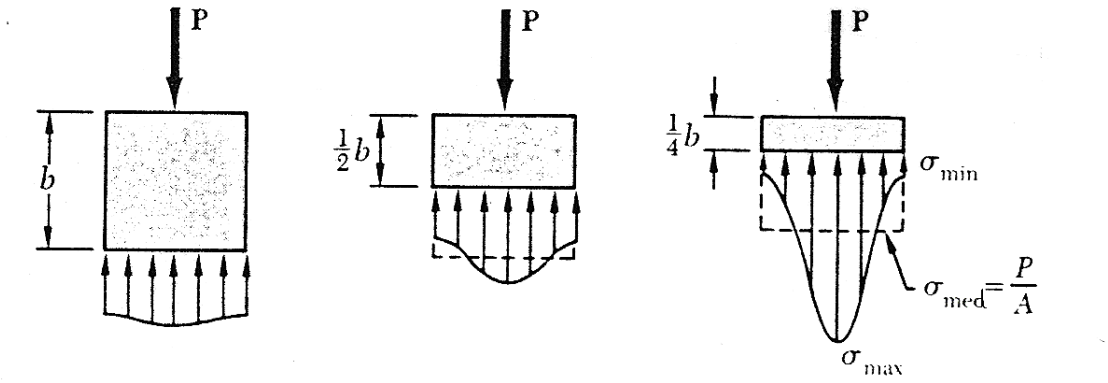


Fig.3.3. – Representação sintética do princípio de *Saint-Venant*

Do mesmo modo, quanto mais próximo se estiver do ponto de aplicação da força, menos uniforme será a distribuição das tensões numa determinada secção transversal, tal como ilustra a Figura 3.4., onde as distribuições das tensões estão representadas em diversas secções transversais, neste caso, tratando-se de uma placa retangular fina a qual sofre um carregamento concentrado.



$$\sigma_{min} = 0.973\sigma_{med} \qquad \sigma_{min} = 0.668\sigma_{med} \qquad \sigma_{min} = 0.198\sigma_{med}$$
$$\sigma_{max} = 1.027\sigma_{med} \qquad \sigma_{max} = 1.387\sigma_{med} \qquad \sigma_{max} = 2.575\sigma_{med}$$

Fig.3.4. – Representação da distribuição de tensões em diversas secções [2]

Tendo em conta a Figura 3.4., é possível observar que a largura da placa, encontra-se a uma distância b da superfície de aplicação da carga, desta forma, é possível denotar que a distribuição de tensões é praticamente uniforme nesta mesma secção. Considerando esses

fatores nesta instância, a tensão σ_y é igual a uma tensão média em qualquer ponto dessa secção transversal.

Assim sendo, ao respeitar as condições de afastamento mínimo do ponto de aplicação da carga concentrada é possível reconhecer qualquer elemento submetido à aplicação de uma carga de qualquer natureza, como um elemento localizado entre duas placas sobre as quais é aplicado uma tensão uniformemente distribuída que por consequência também está uniformemente distribuída ao longo de cada uma das secções transversais do elemento, ou seja, à exceção da vizinhança imediata dos pontos de aplicação das cargas, é possível admitir-se que a distribuição das tensões é independente do modo de aplicação das cargas [2].

Este princípio é conhecido por *Saint-Venant*, o qual permite tratar diferentes tipos de carregamentos de uma maneira relativamente semelhante desde que não sejam consideradas as tensões na vizinhança do ponto de aplicação da carga, sendo que as mesmas não possuem distribuição uniforme. Este princípio enfatiza ainda que a desconsideração das tensões nessa mesma região não deverá modificar a condição estática do sistema, mantendo, portanto, estaticamente equivalentes a relação entre o carregamento real e o carregamento utilizado nos cálculos.

3.2.2. FLEXÃO NORMAL

Assumindo que as secções transversais permanecem perpendiculares ao eixo de um elemento, a superfície neutra torna-se curva durante a sua deformação (Figura 3.5.). O raio de curvatura é representado por ρ. É denotado que a distância entre as secções transversais é medida ao longo da superfície neutra e permanece inalterado em dx, supondo-se que o eixo do elemento não altera o seu comprimento. Desse modo, as fibras longitudinais que localizam-se na superfície neutra não são deformadas, enquanto as fibras acima da superfície são comprimidas e as fibras abaixo são tracionadas, como representado na figura 7.

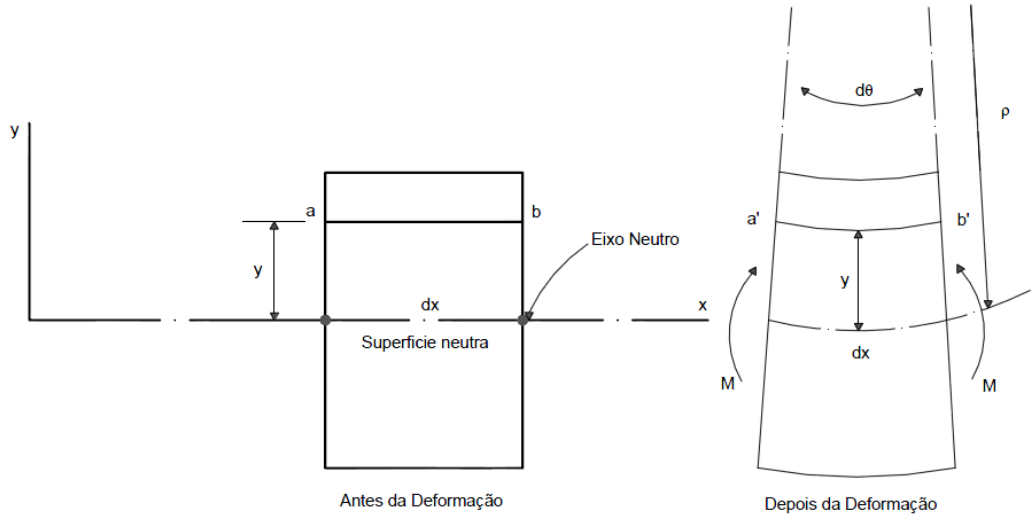


Fig.3.5. – Deformação de um segmento de viga infinitesimal

Se considerarmos a deformação da fibra longitudinal ab, que se situa a uma distância y acima da superfície neutra (ou eixo neutro), como exemplificado na figura acima, no estado deformado, a

fibra forma o arco a'b' de raio (ρ-y), com o ângulo dθ. Atendendo a estes fatores, o seu comprimento deformado é,

$$a'b' = (\rho - y) \cdot d\theta \tag{3.5}$$

Sendo o comprimento original da fibra igual a,

$$a' = dx = \rho \cdot d\theta \tag{3.6}$$

A extensão normal da fibra é determinada dividindo a mudança do comprimento pelo comprimento original, obtendo-se

$$\varepsilon = \frac{a'b' - ab}{ab} = \frac{(\rho - y) \cdot d\theta - \rho \cdot d\theta}{\rho \cdot d\theta} = -\frac{y}{\rho} \tag{3.7}$$

Assumindo agora que a tensão é inferior ao limite de proporcionalidade (ver figura 3), pode-se obter a tensão normal na fibra ab a partir da lei de *Hooke* (2.1).

$$\sigma = E \cdot \varepsilon = -\frac{E}{\rho} y \tag{3.8}$$

Desta forma, observa-se que a distância y da fibra situada no eixo neutro é proporcional à tensão normal da fibra longitudinal. A convenção de sinais indica que quando o sinal for negativo o momento fletor positivo provoca uma tensão de compressão quando y é positivo (nas fibras acima da superfície neutra) e quando y é negativo as tensões são de tração [4].

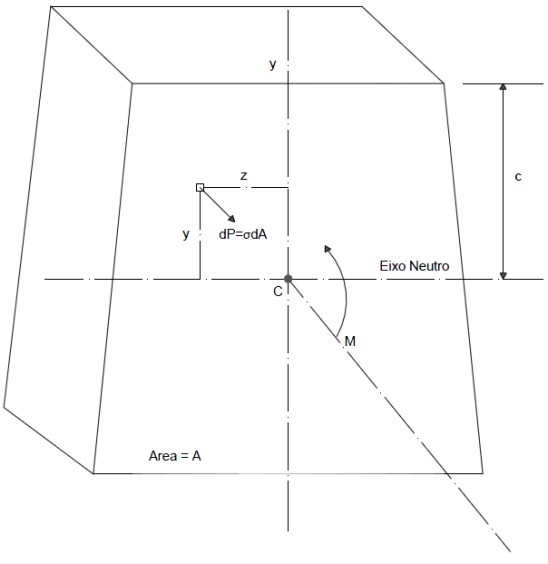


Fig.3.6. – Representação do cálculo da resultante da tensão normal numa secção transversal

Continuando a derivação da formula de flexão, é necessário localizar-se o eixo neutro da secção transversal e derivar a relação de ρ e M, o que pode ser conseguido através da aplicação das condições de equilíbrio. A força normal aplicada na área infinitesimal da secção transversal é

$$dP = \sigma \cdot dA \tag{3.9}$$

Substituindo σ pela equação (3.8), obtêm-se,

$$dP = -\frac{E}{\rho} y \cdot dA \tag{3.10}$$

Onde y é a distância de dA desde o eixo neutro, assim sendo, a resultante da distribuição da tensão normal na secção transversal tem que ser idêntica ao momento fletor M_x, que atua sobre o eixo neutro, por outras palavras

$$-\int_A y \cdot dP = M_x \tag{3.11}$$

O sinal negativo na expressão indica que o momento dP e M positivo possuem sentidos opostos, além disso, a força axial resultante e o momento fletor resultante em torno do eixo y deve ser igual a zero, isto é,

$$\int_A dP = -\frac{E}{\rho}\int_A y \cdot dA = 0 \tag{3.12}$$

Uma vez que a condição para a força axial deve ser igual a zero e porque E/ρ difere de zero, ela só pode ser satisfeita se,

$$\int_A y \cdot dA = 0 \tag{3.13}$$

O integral da equação (3.13) é o primeiro momento da área da secção transversal em torno do eixo neutro, podendo a mesma ser igual apenas se o eixo neutro passar através do centróide C da área da secção transversal. Ora, para que a resultante do momento fletor sobre o eixo y desaparece, a condição é a seguinte,

$$\int_A z \cdot dP = -\frac{E}{\rho}\int_A z \cdot y \cdot dA = 0 \tag{3.14}$$

Onde o produto da inércia da secção transversal é dado por, $\int_A z \cdot y \cdot dA$, considerando que o eixo y é um eixo de simetria da secção, o integral da equação (3.14) é zero.

Notando que a resultante do momento sobre o eixo neutro tem que ser igual a M_x, a resultante do momento sobre o eixo z é,

$$\int_A y \cdot dP = -\frac{E}{\rho}\int_A y^2 \cdot dA = M_x \tag{3.15}$$

Reconhecendo que o momento de inércia da área de uma secção transversal em torno do eixo neutro é,

$$\int_A y^2 \cdot dA = I_x \tag{3.16}$$

Obtêm-se a relação momento, curvatura.

$$\rho = \frac{E \cdot I_x}{M_x} \tag{3.17}$$

Substituindo a equação (3.8) pela equação (3.17), obtêm-se a equação de flexão.

$$\sigma_z = -\frac{M_x}{I_x} \cdot y \tag{3.18}$$

Assinalando que um momento de flexão M_x positivo, causa uma tensão negativa, poderá se compreendida por uma tensão de compressão acima da linha neutra e uma tensão de tração (M_x negativo) abaixo do eixo neutro.

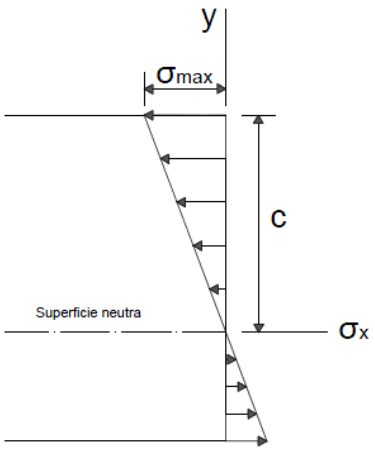


Fig.3.7. – Representação da tensão máxima

Assim sendo e segundo a Figura 3.7., o valor máximo da tensão sem considerar o sinal é dado por,

$$\sigma_x = \frac{|M_{max}| \cdot C}{I_x} \tag{3.19}$$

Onde C representa a distância entre o eixo neutro e o ponto mais afastado na secção transversal [4].

3.2.3. FLEXÃO COMPOSTA E FLEXÃO COMPOSTA DESVIADA

Neste capítulo foi abordado até agora o conceito de um carregamento centrado, ou seja, que a linha de ação das forças induzidas num elemento passava pelo centróide das secções transversais. Nesta instância, analisa-se a distribuição de tensões quando a linha de ação das forças (ou seja do carregamento) não passa pelo centróide, isto é, quando o carregamento é excêntrico na secção transversal.

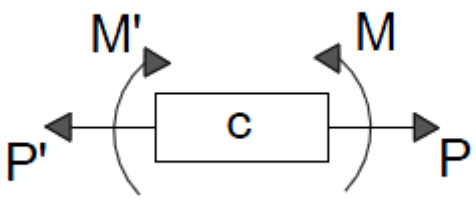


Fig.3.8. – Resultantes de um carregamento excêntrico

Considere-se o carregamento de uma parcela de um elemento descrito na Figura 3.8., ilustrando que as mesmas podem ser tratadas como as resultantes de um carregamento excêntrico, sendo assim, pode-se obter a sobreposição da distribuição de tensões uniformes, que corresponde às cargas centradas P e P', e as tensões lineares dos binários M e M', representado por,

$$\sigma_z = \sigma_{x,axial} + \sigma_{x,\,flexão} \tag{3.20}$$

Aplicando as Eq. (18) e (4), resulta na seguinte expressão.

$$\sigma_z = \frac{P}{A} - \frac{M_x}{I_x} \cdot y \tag{3.21}$$

Relembrando que A representa a área da secção transversal, I representa o momento de inércia em torno do eixo x e y representa a distância entre o ponto de tensão a conhecer e o eixo central da secção transversal.

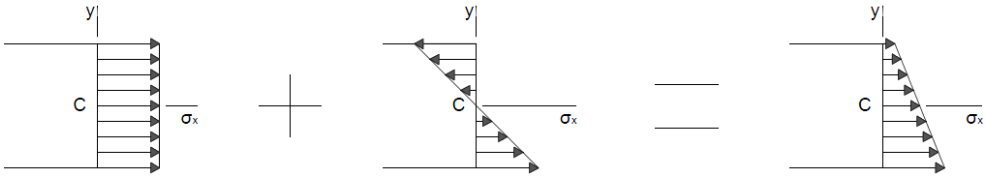


Fig.3.9. – Diagrama de tensões sobre um elemento onde o eixo neutro está fora do referencial

Tal como indica a Figura 3.9., a distribuição das tensões obtidas através da relação (3.21), varia linearmente, atendendo para que neste exemplo as tensões combinadas (3.20) apresentam o mesmo sinal.

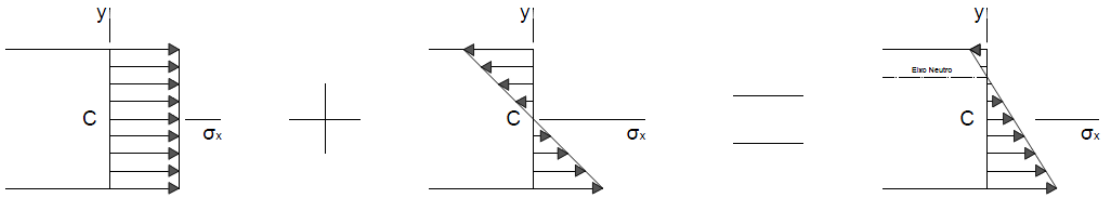


Fig.3.10. – Diagrama de tensões sobre um elemento com representação do eixo neutro

A combinação das tensões (3.20), também pode desenhar um diagrama de tensões como o indicado na Fig.3.10. Deste modo, pode-se concluir que dependendo da geometria da secção transversal e da excentricidade da carga, a combinação de tensões podem variar grandemente dependendo do seu sinal individual, podendo o eixo da secção transversal não coincidir com o eixo neutro sempre que (3.4) e (3.18) sejam diferentes de zero.

É importante reiterar nesta instância, que os resultados aqui apresentados apenas são válidos aplicando o princípio de *Saint-Venant*, ou seja, que as tensões não ultrapassem o limite de proporcionalidade do material e que se mantenha num regime elástico, como ilustrado pela Figura 3 [2].

Utilizando estes mesmos princípios e respeitando o princípio da sobreposição, poderemos considerar a Figura 3.11.

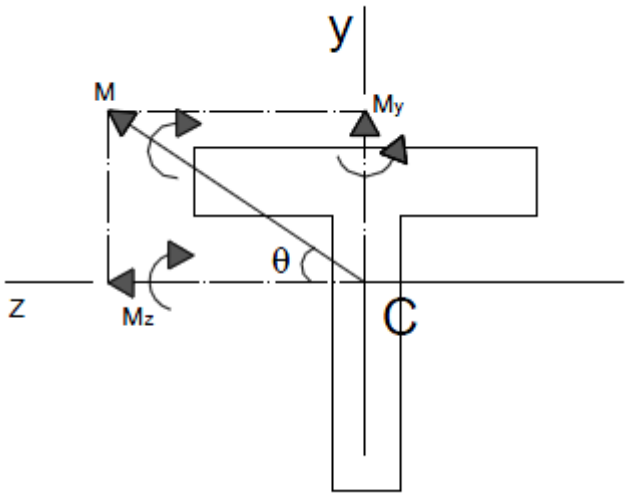


Fig.3.11. – Secção transversal com simetria vertical, sujeita a um momento flector não coincidente com os eixos principais.

Como verificado na Figura 3.11., a secção transversal está submetida a um momento M, que faz um ângulo θ com o eixo horizontal, o qual representa as forças atuantes numa dada secção transversal. Este momento M pode ser decomposto em duas componentes segundo o eixo z e y, representados segundo o referencial apresentado por,

$$M_x = M \cdot \cos\theta \tag{3.22}$$

e,

$$M_y = M \cdot \sin\theta \tag{3.23}$$

Sabendo que os eixos x e y, são os eixos principais de inércia da secção transversal, pode-se aplicar a Equação (3.18) para obter as tensões resultantes das componentes Mx e My. Desta forma, a componente Mx do vetor de momento M, dá origem a uma flexão em torno do eixo z, onde a sua tensão resultará da expressão,

$$\sigma_z = -\frac{M_x \cdot y}{I_x} \tag{3.24}$$

Onde Iz representa o momento de inércia da secção transversal em relação ao eixo z. O sinal negativo indica o facto de haver compressões para y superior a zero e trações para y inferior a zero, por outro lado, a componente My do vetor de momento M, atua num plano horizontal e em torno do eixo y, onde a sua tensão é obtida através da seguinte expressão,

$$\sigma_z = +\frac{M_y \cdot x}{I_y} \tag{3.25}$$

Neste caso, Iy representa o momento de inércia da secção transversal em torno do eixo y, novamente o sinal positivo indica que existem trações quando x é superior a zero e compressões quando x é inferior a zero, assim sendo, a tensões causadas pelo vetor de momento M com o ângulo θ em relação ao eixo x é dado pela conjunção das expressões (3.24.) e (3.25.), sendo ele,

$$\sigma_x = -\frac{M_z \cdot y}{I_z} + \frac{M_y \cdot z}{I_y} \tag{3.26}$$

Apontando de igual forma que a expressão (3.26.) pode também ser usada para determinar tensões em secções transversais assimétricas.

De forma a consolidar os seguintes conceitos, considere-se o seguinte exemplo ilustrado pela Figura 3.12.

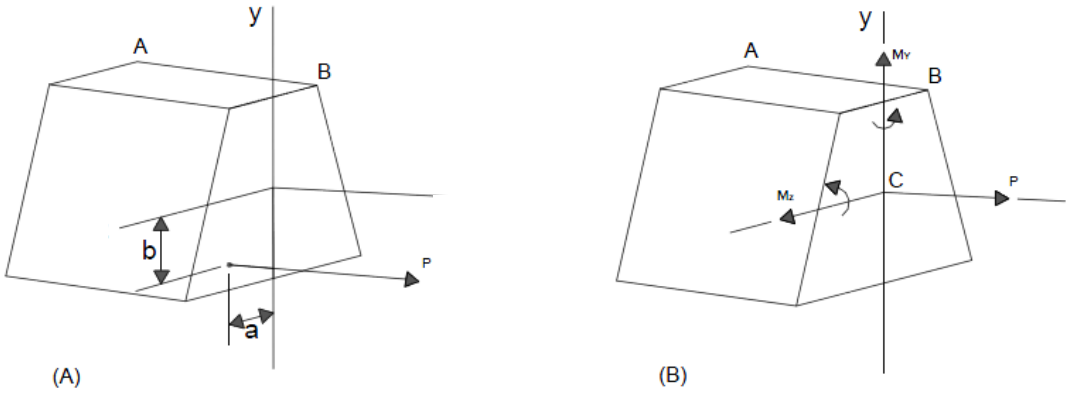


Fig.3.12. – Elemento submetido a uma força excêntrica P

Neste elemento está aplicada uma força "P", afastada do centro da secção transversal, o que causa um momento em torno do eixo "y" que é dado pela força "P" multiplicada pela distância "a", e um momento em torno do eixo "x" dado novamente pela multiplicação da força "P" pela distância "b", como pode ser visto no ponto (B) da Figura 3.12. O carregamento apresentado no ponto (B) é estaticamente equivalente ao carregamento apresentado no ponto (A), através desses dados, pode-se obter por sobreposição as tensões que correspondem à força P e aos momentos M_x e M_y. Aplicando-se as equações (3.4), (3.24) e (3.25), obtêm-se,

$$\sigma_z = \frac{P}{A} - \frac{M_x \cdot y}{I_x} + \frac{M_y \cdot x}{I_y} \tag{3.27}$$

Desta forma e tomando este caso genérico de uma força axial excêntrica, pode-se observar a substituição da carga por um sistema equivalente localizado no centróide da secção transversal, onde os vetores de momento se encontram orientados para os eixos principais de inércia, sendo que x e y são obtidos através desses mesmos eixos de inércia da secção transversal. Finalmente por sobreposição das tensões resultantes da carga axial excêntrica, se poderá obter a expressão das tensões da flexão composta (3.27.).

3.3. TENSÕES TANGENCIAIS

Considere-se que um carregamento transversal a uma viga impõe um aparecimento de tensões normais e tangenciais, sendo que as tensões normais são devidas ao momento fletor M e as tensões tangenciais devido ao esforço transverso V. Para ilustrar um pouco mais, pode-se considerar uma viga simplesmente apoiada na qual é aplicada um esforço transverso, e por consequente, pode-se considerar que a Figura 15 representa a secção transversal da viga submetida as duas forças horizontais P e F, onde P representa a força axial devida à tensão de flexão que atua sobre a área A' da secção transversal e onde M é a resultante da tensão tangencial que atua sobre a superfície horizontal.

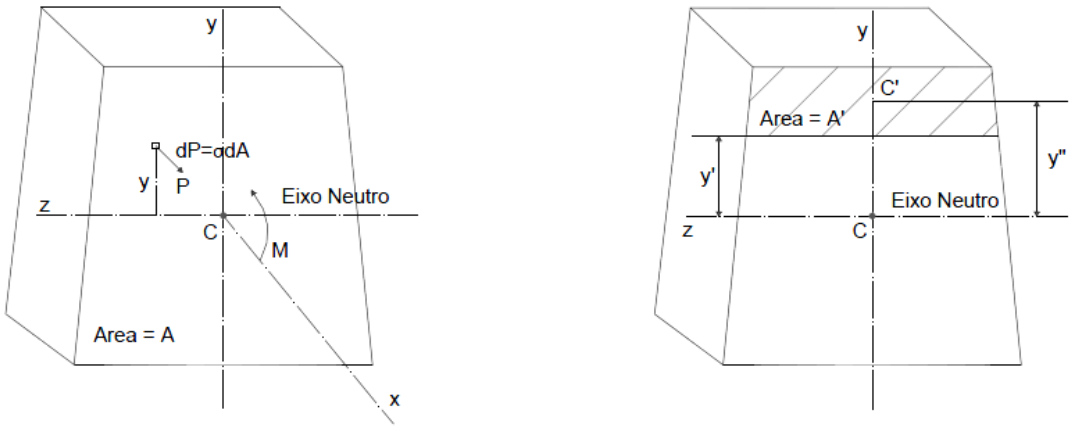


Fig.3.13. – Secção transversal sujeita a uma força P e a um momento M

Segundo a Figura 3.13. identifica-se que a força axial sobre o elemento e atendendo à área da secção transversal pode ser traduzido por,

$$dP = \sigma \cdot dA \tag{3.28}$$

O momento de flexão atuado na secção transversal M, pode ser dado pela Equação (3.18), onde y é a distância a partir do eixo neutro e I é o momento de inércia da área total da secção transversal em torno do eixo neutro, desta maneira pode-se escrever,

$$dP = \frac{M_x \cdot y}{I_x} \cdot dA \tag{3.29}$$

Integrando pela área A' obtêm-se,

$$P = \int_{A'} dP = -\frac{M_x}{I_x} \int_{A'} y \cdot dA = -\frac{M_x \cdot Q}{I_x} \tag{3.30}$$

Onde,

$$Q = \int_{A'} y \cdot dA \tag{3.31}$$

É definido como o primeiro momento de área A em torno do eixo neutro, ainda assinalando que o sinal negativo na equação (3.30) do momento M_x positivo demonstra que a direção da força P é inversa ao representado na Figura 3.13.

Denotando a distância entre o eixo neutro e o centróide C' da área A' por y'', pode-se escrever a expressão (3.31) como,

$$Q = A' \cdot y'' \tag{3.32}$$

Onde Q representa o primeiro momento da área da secção transversal que fica localizada acima da linha y', visto que o primeiro momento da área total da secção transversal em torno do eixo neutro é zero, enquanto que o primeiro momento da área abaixo de y' é –Q. Portanto, a magnitude de Q pode ser calculada usando a área acima ou abaixo de y', sendo que o valor máximo de Q ocorre no eixo neutro onde y'=0. Desta forma, resulta que a força de corte horizontal é maior na superfície neutra, a variação de Q com y' é ilustrada na Figura 3.14.

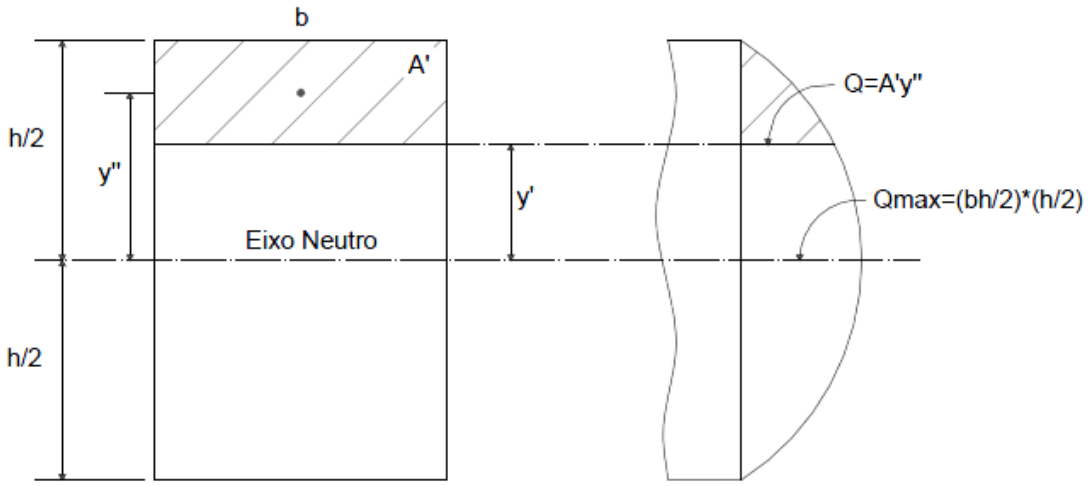


Fig.3.14 – Variação do primeiro momento Q da área A' em torno do eixo neutro

Analisando o diagrama de corpo livre da zona sombreada da Figura 3.15., ele é delimitado pelas secções 1 e 2 que estão separadas pela distância dx infinitesimal e um plano horizontal localizado a uma distância y' acima do eixo neutro da secção transversal. Se o momento de flexão do ponto 1 é M, então a força resultante que atua sobre a face de 1 do corpo é dado por,

$$P = -M_x \cdot \frac{Q}{I_x} \tag{3.33}$$

Tal como mencionado anteriormente, Q representa o primeiro momento da área A'. A ação do momento de flexão na secção 2 é M_x+dM_x, onde dM_x é a mudança infinitesimal em M_x sobre a distância dz.

Examinando, portanto, a ação normal atuante na face 2 do elemento pode-se escrever a seguinte expressão,

$$P + dP = -(M_x + dM_x) \cdot \frac{Q}{I_x} \tag{3.34}$$

Visto que as força diferem por,

$$(P + dP) - P = -(M_x + dM_x) \cdot \frac{Q}{I_x} - (M_x \cdot \frac{Q}{I_x}) = dM_x \cdot \frac{Q}{I_x} \tag{3.35}$$

O equilíbrio só pode ocorrer se existir uma força de corte igual e oposta dF atuante na superfície horizontal.

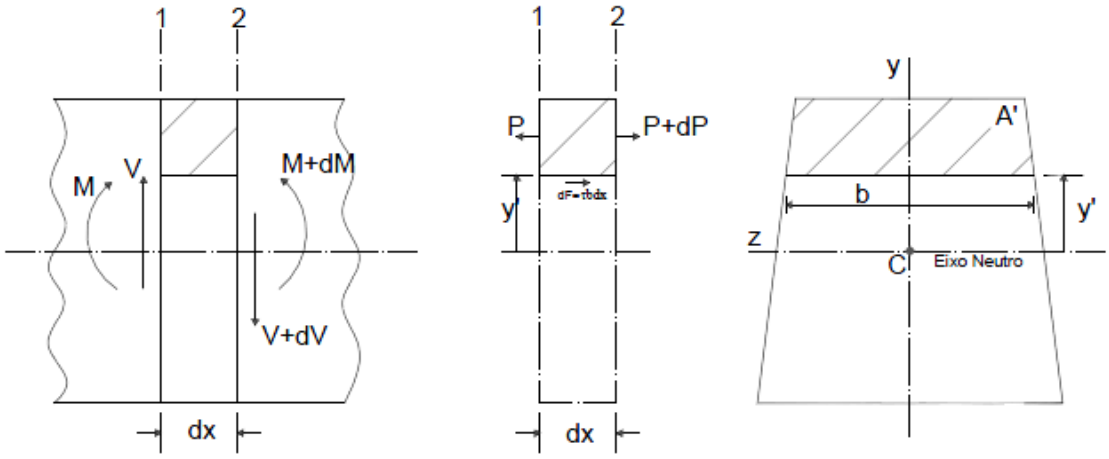


Fig.3.15 – Representação da determinação da tensão de corte

Assumindo que τ representa a tensão de corte média atuante no plano horizontal, a sua resultante é,

$$dF = \tau \cdot b \cdot dx \tag{3.36}$$

Onde b é a largura da secção transversal em y=y' com apresentado na Figura 3.15., o equilíbrio necessário para as forças horizontais é,

$$\sum F = 0 : (P + dP) - P + \tau \cdot b \cdot dx = 0 \tag{3.37}$$

Substituindo então na equação (3.35.) tem-se,

$$- dM \cdot \frac{Q}{I} + \tau \cdot b \cdot dx = 0 \tag{3.38}$$

O que dá,

$$\tau = \frac{dM}{dx} \frac{Q}{I \cdot b} \tag{3.39}$$

Relembrando que *numa viga submetida à força cortante V, aparecem tensões de corte quer nas secções transversais, quer nas longitudinais, isto é, perpendiculares aquelas. Numa determinada secção transversal, as tensões de corte, que nela atuam, tem como resultante, ou são equivalentes, a força cortante V* [3]. Podemos escrever que a relação entre a força de corte e o momento de flexão é dado por,

$$V = \frac{dM}{dx} \tag{3.40}$$

Obtemos a tensão de corte horizontal,

$$\tau_{zy} = \frac{V_y \cdot Q}{I_x \cdot b} \tag{3.41}$$

Denota-se então que a fórmula (3.41) prevê que o maior esforço de corte de uma viga prismática ocorra na secção transversal que acarreta a maior força de corte vertical V. Do mesmo modo, o valor da tensão máxima tangencial é determinado pela razão Q/b, visto que Q é sempre máximo para y' zero, ou seja, no eixo neutro. No entanto, se a largura b no eixo neutro for maior do que em outras partes da secção transversal, será necessário calcular τ em dois ou mais valores de y' antes de se conseguir determinar o seu valor máximo.

A tensão tangencial (ou tensão de corte), apresenta tal como as tensões normais uma concentração maior de tensões próximas de cantos retos e buraco na secção transversal, acrescendo que na junção entre a alma e o banzo apresentam, de igual modo, concentrações de tensões [4].

3.4. ESTADO PLANO DE TENSÕES

O estado de tensão num ponto é definido pelos componentes de tensão atuantes sobre os lados de um elemento de volume diferencial que inclui esse mesmo ponto. Sabendo o estado de tensão num ponto permite calcular os componentes das tensões que atuam sobre qualquer plano que passa através desse ponto, o qual, por sua vez, permite encontrar as tensões máximas de um elemento (nota-se que não existe correlação entre o referencial apresentado no ponto 3.4 com o referencial apresentado nos pontos anteriores)

Portanto, a obtenção dos estados de tensão num ponto, pode ser conseguida através dos componentes que foram referidos até agora, isto é, σ_x e σ_y, que podem ser obtidos utilizando a expressão (3.27) e τ_{xy} calculado através da utilização da expressão (3.41), mencionada anteriormente. A convenção do descrito, pode ser observado utilizando a Figura 3.16.

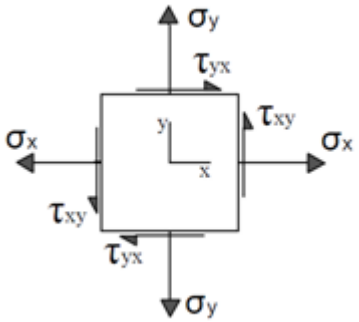


Fig.3.16. – Plano biaxial do estado de tensões. Componentes atuando nas suas direções positivas

Conforme a representação da Figura 3.16. indica, σx denota a tensão normal atuando na face x, do mesmo modo que σy denota a tensão normal atuando na face y, que por sua vez, τ_{xy} representa a tensão de corte na face x atuando na direção y e vice-versa, sendo que a magnitude dos esforços das tensões tangenciais (ou de corte) são idênticos, tal como referido no ponto 2.3.

É importante referir uma convenção de sinais para as tensões representadas no plano biaxial. Continuando a utilizar a Figura 3.16. como referência, denota-se que a cada face do elemento lhe é atribuída uma notação x ou y para as tensões. Estas são, por sua vez, perpendiculares aos eixos do referencial do elemento. Define-se então, que os pontos normais exteriores na direção de coordenadas positivas, são positivos, pelo contrário, os pontos normais exteriores na direção de coordenadas negativas, são negativos, sendo que as tensões são iguais e opostas às suas contrapartes em cada uma das faces. Pode-se então definir, que a convenção de sinais é a seguinte:

- As tensões de compressão são representadas pela direcção do vetor, segundo a sua face respetiva.
- As tensões de tração são representadas pelo sentido do vetor, iterando novamente que a sua direção é para o exterior da sua face respetiva.
- Em relação a representação da tensão de corte o primeiro vetor indica a face sobre o qual ele atua, enquanto o segundo indica o sentido da tensão.

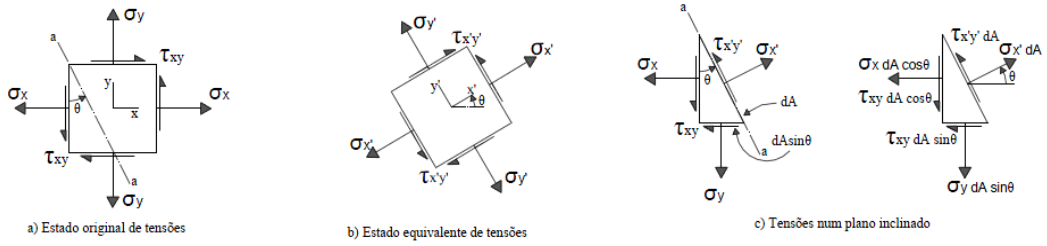


Fig.3.17. – Representação da transformação das coordenadas xy do estado planos de tensão, para coordenadas x'y'

Como foi mencionado anteriormente, o estado de tensão num ponto é representado pelas tensões que atuam na face de um elemento do qual o seu volume engloba o ponto em questão, do mesmo modo que as tensões num ponto dependem das inclinações dos planos sobre os quais atuam, as tensões sobre as faces do elemento variam conforme a orientação do elemento é alterada. Essas mesmas variações das suas relações matemáticas são chamadas de equações de transformação, as quais são utilizadas para encontrar as tensões máximas e mínimas num ponto.

Atendendo para a Figura 3.17. a), ela demonstra o estado plano de tensões num ponto. Onde as faces do elemento são perpendiculares aos eixos x e y, por sua vez, a figura b) representa o estado plano de tensão no mesmo ponto, mas agora as faces do elemento são perpendiculares ao eixo x' e y', assim, a sua orientação difere pelo ângulo θ (positivo no sentido anti-horário partindo do eixo x para o eixo x') em relação à orientação dos dois conjuntos de eixos. Em conformidade com o referencial das direções positivas apresentado pela Figura 3.16., as componentes da Fig.3.17. a) e b), estão representadas de acordo com esse mesmo referencial, ou seja no sentido positivo. As tensões tangenciais de τ_{xy} dos planos complementares podem-se dizer que são iguais a τ_{yx}, da mesma maneira que $\tau_{x'y'}$ também é igual a $\tau_{y'x'}$, por demais, pode-se afirmar que a Fig.3.17. a) e b) possuem equivalentes estados de tensão, representando, portanto, o mesmo estado de tensão, apenas referindo dois conjuntos diferentes de eixos de coordenadas.

O plano apresentado em c) coincide com o eixo x' do elemento mostrado em b), devido a isso as tensões atuantes nesse plano são σ x' e τx'y'. Se forem aplicadas as equações de equilíbrio para o elemento c) e derivar σ x' e τx'y' em função do ângulo θ, deixando que a área da face inclinada seja dA, descobre-se que as áreas de x e y são as faces dAcosθ e dAsenθ, tal como ilustrado. Multiplicando as tensões pelas áreas em que atuam obtêm-se as forças demonstradas no diagrama d), desta forma, as equações de equilibro são,

$$\Sigma Fx = 0 \Leftrightarrow \sigma_{x'} \cdot dA \cdot \cos\theta - \tau_{x'\cdot y'} \cdot dA \cdot \sin\theta - \sigma_x \cdot dA \cdot \cos\theta - \tau_{xy} \cdot dA \cdot \sin\theta = 0 \quad (3.42)$$

$$\Sigma Fy = 0 \Leftrightarrow \sigma x' dA\cos\theta + \tau x' y' dA\sin\theta - \sigma y dA\cos\theta - \tau xy dA\sin\theta = 0 \quad (3.43)$$

Resolvendo as tensões no plano c), têm-se,

$$\sigma_{x'} = \sigma_x \cdot \cos^2\theta + \sigma_y \cdot \sin^2\theta + 2 \cdot \tau_{xy} \cdot \sin\theta \cdot \cos\theta = 0 \quad (3.44)$$

$$\tau_{x'y'} = -(\sigma_x - \sigma_y) \cdot \sin\theta \cdot \cos\theta + \tau_{xy} \cdot (\cos^2\theta - \sin^2\theta) \quad (3.45)$$

Assim sendo, a tensão normal que atua sobre a face y' do elemento da Fig.3.17. b) pode ser obtida por substituição de θ por θ+90º na equação (3.44),

$$\sigma_{y'} = \sigma_x \cdot \sin^2\theta + \sigma_y \cdot \cos^2\theta - 2 \cdot \tau_{xy} \cdot \sin\theta \cdot \cos\theta = 0 \quad (3.46)$$

As equações (3.44), (3.45) e (3.46) são equações de transformação. Utilizando as relações trigonométricas nas ditas equações, podemos obter as formas mais comuns de apresentação das equações de transformação dos estados planos de tensão,

$$\sigma_{x'} = \frac{\sigma_x + \sigma_y}{2} + \frac{\sigma_x - \sigma_y}{2} \cdot \cos 2\theta + \tau_{xy} \cdot \sin 2\theta \quad (3.47)$$

$$\sigma_{y'} = \frac{\sigma_x + \sigma_y}{2} + \frac{\sigma_x - \sigma_y}{2} \cdot \cos 2\theta + \tau \cdot \sin 2\theta \quad (3.48)$$

$$\tau_{x'y'} = -\frac{\sigma_x - \sigma_y}{2} \cdot \sin 2\theta + \tau_{xy} \cdot \cos 2\theta \quad (3.49)$$

Estas expressões mostram que se o estado de tensões num determinado ponto for conhecido (em termos de σ_x, σ_y e τ_{xy}), é possível calcular as tensões que atuam em qualquer plano que passe por esse ponto, daí resulta o plano de tensões originais representado na Fig.3.17. a) for conhecido, então as equações de transformação permitem obter o estado equivalente de tensões, mostrado em b). Ressalta-se que a expressão (3.47) e (3.48) indica que a soma das tensões normais é invariável, ou seja, não depende da orientação do elemento. Isto pode ser apresentado pela seguinte expressão,

$$\sigma_{x'} + \sigma_{y'} = \sigma_x + \sigma_y \tag{3.50}$$

3.4.1. PLANO DE TENSÕES PRINCIPAIS

No plano de tensões principais, as tensões normais atuantes no ponto em questão são consideradas máximas e mínimas. As direções que são perpendiculares aos planos principais são chamados de direções principais, onde os valores do ângulo θ definem a direção principal através da condição,

$$\frac{d\sigma_{x'}}{d\theta} = 0 \tag{3.51}$$

Se for utilizado a expressão (3.47) em função de $\sigma_{x'}$, esta condição torna-se em,

$$\frac{d\sigma_{x'}}{d\theta} = -(\sigma_x - \sigma_y) \cdot \sin 2\theta + 2 \cdot \tau_{xy} \cdot \cos 2\theta = 0 \tag{3.52}$$

Que resulta em,

$$\tan 2\theta = \frac{2 \cdot \tau_{xy}}{\sigma_x - \sigma_y} \tag{3.53}$$

Atendendo a esta mesma expressão, ela produz duas soluções para 2θ que diferem entre si 180º, então pode-se apresentar uma solução por $2\theta_1$ e outra por $2\theta_2$, sendo esta última igual a 2θ1 mais 180º, desta forma, as duas direções principais diferem em 90º.

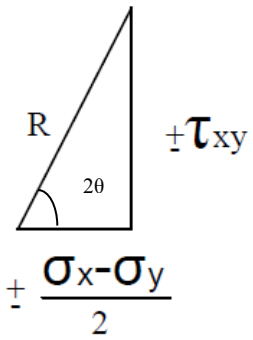


Fig.3.18. – Método geométrico para a determinação dos ângulos θ_1 e θ_2

Os senos e co-senos de $2\theta_1$ e $2\theta_2$ podem ser obtidos através da utilização do elemento representado na Figura 3.18., ou seja,

$$\begin{matrix}\sin 2\theta_1 \\ \sin 2\theta_2\end{matrix} \mapsto = \pm \frac{\tau_{xy}}{R} \tag{3.54}$$

$$\begin{matrix}\cos 2\theta_1 \\ \cos 2\theta_2\end{matrix} \mapsto = \pm \frac{\sigma_x - \sigma_y}{2 \cdot R} \tag{3.55}$$

Onde,

$$R = \sqrt{(\frac{\sigma_x - \sigma_y}{2})^2 + \tau^2{}_{xy}} \tag{3.56}$$

Substituindo as equações (3.54) e (3.55) na equação (3.47) e, desta forma, simplificando, obtêm-se,

$$\begin{matrix}\sigma_1 \\ \sigma_2\end{matrix} \mapsto = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{(\frac{\sigma_x - \sigma_y}{2})^2 + \tau^2{}_{xy}} \tag{3.57}$$

Onde σ_1 e σ_2 atuam nos planos respetivamente definidos por θ_1 e θ_2. A tensão de corte atuante sobre os planos principais de tensão é obtida através da substituição da equação (3.54) e (3.55), na equação (3.49), tal que,

$$\begin{matrix}\tau_{12}\\ \tau_{21}\end{matrix} \mapsto = -\frac{\sigma_x - \sigma_y}{2} \cdot (\pm\frac{\tau_{xy}}{R}) + \tau_{xy} \cdot (\pm\frac{\sigma_x - \sigma_y}{2R}) = 0 \tag{3.58}$$

O que mostra que não há tensões de cisalhamento sobre os planos principais de tensão.

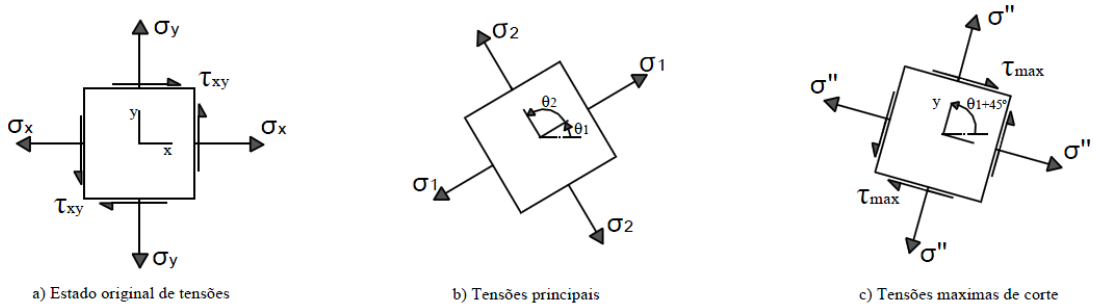


Fig.3.19 – Estados equivalentes de tensão num ponto

Ao observar a Figura 3.19., denota-se que a alinha a) mostra o estado original de tensões (segundo o referencial enunciado anteriormente) para um sistema de coordenadas xy. No ponto b) nota-se que os dois planos principais diferem em 90º e não possuem tensões de cisalhamento. Como mencionado anteriormente, este é o estado equivalente de tensões para o estado original de tensões. Denota-se novamente que a alinha a) e b) da Figura 3.19. exibe duas representações do mesmo estado de tensão.

3.4.2. TENSÃO MÁXIMA DE CORTE

É chamado de tensão máxima de corte a maior magnitude de $\tau_{x'y'}$ em determinado ponto, o qual é representado por τ_{max}. Os valores de θ que definem os planos de tensão de corte máxima encontram-se a partir da equação,

$$\frac{d\sigma_{x'}}{d\theta} = 0 \tag{3.59}$$

Onde τx'y' é dado pela equação (3.49), colocando a derivada igual a zero e resolvendo em função do ângulo θ, obtêm-se a seguinte expressão,

$$\tan 2\theta = -\frac{\sigma_x - \sigma_y}{2 \cdot \tau_{xy}} \tag{3.60}$$

Na semelhança do referido no ponto 2.4.1. para a equação (3.53), a expressão (3.60) produz duas soluções para 2θ que diferem por 180º, desta forma, existem dois valores de θ que diferem

por 90º, observando assim que a expressão (3.60) é o inverso da expressão (3.53), o que significa que os ângulos 2θ definidos por estas duas equações diferem por 90º. Pode-se então concluir que o estado de tensão máxima está inclinado a 45º em relação aos planos principais de tensão.

Usando um triângulo similar ao apresentado na Figura 3.18., atesta-se que é de facto possível de se encontrar o ângulo de sen2θ e cos2θ. Substituindo o resultados nas equações (3.47), (3.48) e (3.49), obtêm-se a tensão máxima de corte, dada por,

$$\tau_{\max} = R = \sqrt{(\frac{\sigma_x - \sigma_y}{2})^2 + \tau^2{}_{xy}} \tag{3.61}$$

As tensões normais que atuam sobre os planos de tensão máxima de corte, são,

$$\sigma'' = \frac{\sigma_x + \sigma_y}{2} \tag{3.62}$$

Estes resultados são demonstrados na Figura 3.19. c), e comparando as expressões (3.61) e (3.58), podemos chegar à conclusão de que a expressão para a obtenção da tensão máxima de corte pode ser dada por,

$$\tau_{\max} = \frac{|\sigma_1 - \sigma_2|}{2} \tag{3.63}$$

3.4.3. CÍRCULO DE *MOHR*

As expressões gerais definidas anteriormente de σ e τ, fornecem as tensões no plano definido pelo ângulo θ, as quais podem ter uma representação gráfica simples através da construção do círculo de *Mohr*. Considerando-se um sistema de coordenadas onde se representam, no eixo das abcissas as tensões normais e no eixo das ordenadas as tensões de corte, podem-se marcar depois pontos, pelo qual a circunferência do círculo de *Mohr* passará, tendo a mesma o seu centro no eixo das abcissas, ou seja, das tensões normais. Desta forma, obtêm-se então o círculo de *Mohr*, tal como ilustrado pela Figura 3.20. adiante.

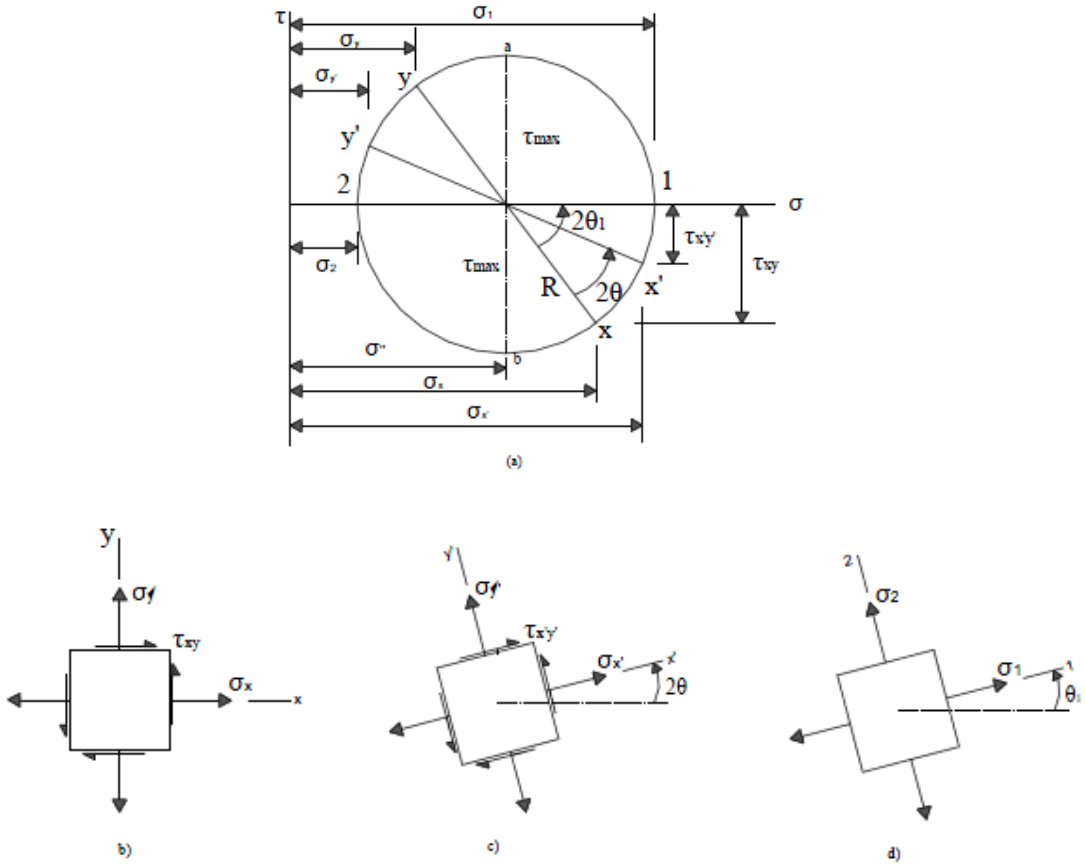


Fig.3.20. – Círculo de *Mohr* e transformação de estados planos de tensão

É necessário reiterar novamente a convenção de sinais para o círculo de *Mohr*, que de forma resumida consta no seguinte. As tensões de tração são positivas e as de compressão negativas. Quando as tensões de corte são positivas o elemento tende a girar no sentido anti-horário, quando as mesmas são negativas o elemento gira no sentido horário. Utilizando a Figura 3.20. como referência, o eixo horizontal do círculo é dado pelas tensões normais, e o eixo vertical é medido através das tensões de corte, assim sendo, as coordenadas de cada ponto na circunferência representam as tensões normais e as tensões de corte que atuam sobre um plano específico que passa através do ponto selecionado do corpo. Desta forma, é necessário identificar cada ponto colocado sobre o círculo de *Mohr* de forma a se conseguir identificar o plano que representa. Então para calcular as tensões normais σ e as tensões de corte τ, num plano inclinado a θ no sentido anti-horário em relação ao eixo dos x, marca-se o ângulo 2θ a partir do diâmetro de dois pontos do círculo de *Mohr*. Na Figura 3.20., seria os pontos X e Y, visto serem os pontos extremos do diâmetro (sendo os mesmos utilizados para desenhar o diâmetro do círculo de *Mohr*). Representam as tensões nos planos paralelos aos eixo x e y, ou seja, σ_x, τ_{xy}, e σ_x, $-\tau_{xy}$, ressaltando que o ângulo 2θ entre os dois diâmetros no círculo é o dobro da transformação do ângulo θ, com ambos os ângulos medidos na mesma direção, isto é, no sentido horário ou anti-horário.

De modo a descrever o procedimento para a utilização do círculo de *Mohr* para a transformação dos componentes de tensão a partir dos eixos xy para os eixos x'y', continuamos a observar a Figura 3.20. Tendo em conta o sentido e a magnitude de ângulo θ entre os eixos xy e x'y' do ponto c), o sentido de θ toma a direção na qual os eixos xy têm que ser rodados para coincidir com os eixos x'y'. Girando o diâmetro X-Y do círculo de *Mohr* através do ângulo 2θ no mesmo

sentido que o ângulo θ, obtêm-se os pontos finais do novo diâmetro, os quais são X' e Y', como mostrado no ponto a). Por consequente, as coordenadas do ponto X' são ($\sigma_{x'}$, $\tau_{x'y'}$), e as coordenadas do ponto Y' são ($\sigma_{y'}$, $\tau_{x'y'}$).

Usando o círculo de *Mohr* é possível identificar as tensões e direções principais. Observando-se o ponto d) da Figura 3.20., determina-se que os planos principais dos planos de tensão (máximo e mínimos normais) são identificados respetivamente como (1) e (2). As coordenadas das tensões normais deste pontos são,

$$\sigma_1 = \sigma'' + R \tag{3.64}$$

De igual forma,

$$\sigma_2 = \sigma'' - R \tag{3.65}$$

As coordenadas das tensões de corte para estas orientações são zero. Conforme mencionado anteriormente, as direções principais diferem da direção das coordenadas xy pela grandeza do ângulo θ_1. A sua magnitude e sentido são determinados pelo ângulo $2\theta_1$ no círculo de *Mohr*, alem disso, é possível de ser observado através do círculo de *Mohr* apresentado na Figura 3.20. a), que a máxima tensão de corte, ou seja, τ_{max} é igual ao raio R do círculo. Os planos nos quais τ_{max} atua, são representados pelos pontos (a) e (b) no círculo em questão. Observando-se que as coordenadas de σ em (a) e (b) são σ'', os pontos que se inserem no círculo e que correspondem aos planos de tensão máxima de corte, diferem por 90º a partir dos pontos que representam os planos principais de tensão. Portanto, a diferença entre os planos de corte máximo e os planos principais é de 45º, tal como mencionado anteriormente e como esperado.

4
CONCEÇÃO DA FERRAMENTA DE CÁLCULO

4.1. INTRODUÇÃO

Este capítulo tem como finalidade, apresentar e ilustrar as metodologias adotadas, no ponto de vista do paradigma de programação e no sentido de conceção da ferramenta de cálculo, designada por "*Solve & Draw"*. Devido à inerente extensão do algoritmo, os detalhes que serão aqui apontados terão como objetivo uma configuração mais conceptual, de forma a poder esclarecer os aspetos mais pertinentes do desenvolvimento, e a apresentar metodologias que auxiliem na perceção inerente das dificuldades e da procura de alternativas para as soluções menos ortodoxas.

Todo o método de interação com o utilizador será descrito mais adiante no Capítulo 5, pelo que neste capítulo apenas serão feitas breves menções à introdução de dados, de forma a referir ideologias de rotinas de cálculo.

4.2. CARACTERÍSTICAS DA FERRAMENTA

A necessidade inicial foi a da criação de um modelo que permitisse o cálculo orgânico de qualquer secção no espaço de duas dimensões para qualquer tipo de análise de flexão composta desviada, sem deixar de parte a otimização da interface para os diversos aparelhos de *hardware* (computador, *tablet*, página *Web*). A fluidez com que se permite fazer alterações a qualquer figura teria que ser simples e intuitiva, bem como o método de inserção de dados no programa *Solve&Draw*. Tornou-se óbvio que seria necessário criar diversos métodos de utilização, sendo que a componente virtual de "*touch*", ou seja, a utilização maioritariamente através do rato, teria que ter primazia em relação às restastes, devido em grande parte à padronização atual dos aparelhos móveis.

Para além deste ponto central de cálculo, o mesmo teve que ser também aplicável aos casos mais correntes, tal como figuras retangulares, em T, círculos, ou em I, em que a sua resolução não deveria necessitar o seu desenho devido à sua inerente padronização. Seguindo então, estes princípios de conceção, foi definido que a ferramenta deveria ter as seguintes características:

- Ser manuseada através de *input* (inserção de dados) pelo rato, pelo teclado e por *touchscreen* (utilizando dispositivos manuseados com os dedos).
- Criar um manuseamento familiar para os utilizadores ligados à engenharia civil, ou seja, a criação de um manuseamento através do rato idêntico ao do programa AutoCAD (*Computer Aided Design*, ou em português, desenho assistido por computador), no sentido de permitir *zoom in* e *zoom out*, por meio da rotação da roda do rato, e *pan*

(deslocar a imagem) mantendo pressionado a tecla interior do rato, podendo ser a mesma a própria roda do rato.

- Criação de uma grelha genérica segundo os eixos de coordenadas, de forma a apoiar o desenho "livre" de secções, fazendo com que a mesma acompanhe os movimentos do utilizador, ou seja, quando se faça *zoom out* a grelha diminua, quando se faça *zoom in* a grelha aumente.
- Criação de uma zona de seleção rápida de secções padronizadas, tal como mencionado anteriormente (retângulo, circulo, T, I).
- Representação visual do centróide nas secções desenhadas e de igual forma apresentar os diagramas de tensões normais junto da secção sem abrir novas janelas.
- Permitir a visualização de ambos os diagramas das tensões de corte e das tensões normais.
- Alteração fluida das soluções, ou seja, após a indicação para se realizar o cálculo, a rotina continua a correr e a variar as soluções conforme o utilizador vai arrastando um ou mais pontos.
- Permitir exportar e guardar os valores obtidos no cálculo.
- Permitir a criação e cálculo de secções com aberturas interiores.
- Criação de um "zona" de seleção rápida para o tipo de cálculo que se deseja efetuar.

Assim sendo, demonstra-se que se trata de uma ferramenta de fácil utilização e com certa familiaridade aos mecanismos de *input* utilizados pelos *softwares* mais comuns. Um ponto a ressaltar encontra-se no facto de não "existir" um *input* de eliminação de pontos. Esta decisão de *design* da ferramenta, pode parecer contraintuitivo, e talvez seja encarado como tal para um utilizador que esteja habituado a corrigir pontos que tenha introduzido incorretamente por eliminação, mas à semelhança do *software* AutoCad, é possível selecionar um ponto e alterar a sua posição através de inserção de novas coordenadas ou por arrastamento do ponto, sem ser necessário eliminar o ponto e posteriormente volta-lo a inserir. Da mesma forma e também á semelhança do método de utilização do rato na interface do AutoCad, a movimentação da tela de desenho principal pode ser movimentada através do pressionamento constante da roda do meio do rato e posteriormente graças à movimentação do rato em si, a imagem também se desloca. De igual forma pode ser executado um *zoom out* ou *zoom in* através da movimentação da roda do rato para cima ou para baixo, resultando respetivamente numa aproximação da imagem ou um afastamento dela. Estes fatores ajudam ao utilizador a poder manusear com mais familiaridade o programa, e a criar uma afinidade mais rápida com o programa.

4.3. ESTRUTURAÇÃO DA FERRAMENTA

Devido aos fatores inerentes associados com a conceção da ferramenta mencionados anteriormente, foi preferível estruturar o programa por módulos, recorrendo à programação orientada por objetos. O código fonte, ou a sequência de cálculo, mantem-se a mesma, apenas a forma como ela interage com os objetos se altera, no entanto, alguns módulos possuem um paradigma procedimental único e diferenciado. A grande vantagem da programação por módulos encontra-se na facilidade de introduzir novos mecanismos de cálculo, ou de criar novos módulos sem ter que inserir uma linha de código no programa global. Obviamente a contra partida é a interação entre módulos que não pode ser gerada sem um armazenamento de informação de processamento individualizado por módulo, o que pode criar alguma complexidade no acesso sequencial aos dados, não desprezando com isto a grande vantagem para expansão da ferramenta sem ter que se reutilizar ou alterar as linhas de código anteriores.

Desta forma, o programa está subdividido essencialmente em duas partes conceptuais, sendo elas:

- Cálculo procedimental, ou seja, desenho livre sem linha sequencial de operações, podendo-se obter resultados por parcelas, com alterações de geometria.
- Cálculo sequencial direcionado para um tipo específico de lógica de estruturação de resolução.

4.4. ESTRUTURA DA *INTERFACE*

Como já foi mencionado várias vezes neste documento, a necessidade de criar uma interface intuitiva parte do princípio do funcionamento em multiplataformas, onde nem todo o tipo de visualizações é ideal para a perceção dos sistemas de cálculo ou de resultados. Usualmente, é recorrido a janelas "*drop down*", nas quais se podem escolher determinadas operações, ou a ícones com determinados desenhos que indicam o funcionamento de opção. Ora para certos tipos de *hardware*, estes tipos de interações podem prejudicar a interação do utilizador com a ferramenta, podendo causar frustração e transtorno, portanto, partindo desta perspetiva foi decidido apresentar as opções para o utilizador de forma maior do que o normal e com o nome do que cada opção fará.

De seguida, é necessário averiguar qual o método mais eficaz para a representação do sistema de coordenas e para a visualização da secção. Recorrendo à implementação de um sistema de ampliação, redução e ajuste de imagem, possibilitasse que cada utilizador possa escolher o "tamanho" de visualização que pretende, partindo, como é óbvio, de uma visualização padrão.

No que se trata de representação de resultados, eles são representados graficamente através de diagramas, e também numericamente. Nesta ocorrência, é pertinente discutir qual a metodologia de apresentação e qual a filosofia de utilização, isto é, suponhamos que o utilizador apenas deseja resolver um dado problema, é mais acessível para o mesmo que exista um *menu* no qual se apresentam os resultados e que eles sejam exportados, ou copiados para o exterior do programa, no entanto, se o utilizador pretender fazer um análise teórica ou prática de uma determinada secção e pretender visualizar as alterações dos resultados frequentemente é preferível que tenha a visualização direta dos valores, de modo a que lhe seja permitido retirar as suas ilações com maior facilidade. De modo a se compreender melhor esta situação e para que se alcance uma solução ideal, é necessário abordar o conceito de ruído visual e de ergonomia.

4.4.1. ERGONOMIA DE SISTEMA MULTIMÉDIA

Ergonomia ou por outras palavras a ciência da interação de seres humanos com sistemas multimédia tem como objetivo otimizar a utilização dos mesmos, e demonstra os princípios pelos quais se transmite informação reduzindo ao máximo o ruído visual. Antes de mais, é necessário entender quais as etapas por qual cada utilizador passa para levar a cabo uma certa linha de ação numa determinada aplicação, podendo passar pela seguintes etapas,

i) Formar o objetivo;
ii) Formular hipóteses de ação;
iii) Especificar uma ação;
iv) Executar a ação;
v) Perceber o resultado no seu contexto;
vi) Interpretar o resultado;
vii) Validar o resultado.

Assim, num sistema multimédia interativo todas as ações têm inerente um ciclo de estados que começa com um determinado objetivo na mente do utilizador. Para atingir esse objetivo, o utilizador vai pensar numa forma de cumpri-lo. Leva, então a cabo a ação que pensa ser a mais indicada para cumprir o objetivo. A ação é executada e o utilizador tem um conjunto de perceções acerca dos seus efeitos, que interpreta e avalia [13].

Desta forma, é necessário seguir algumas diretrizes no que diz respeito à produção de uma interface otimizada para o utilizador. Alguns dos fatores são,

- Destacar os elementos que sejam mais relevantes de uma ferramenta e torná-los facilmente visíveis.
- Receção de *feedback* acerca do estado da ação que o utilizador efetuou.
- Consistência da representação gráfica da informação e dos mecanismos de ação.
- Bom mapeamento das opções de ação, para que o utilizador perceba a que ação se encontra associada ao elemento.
- Agrupar a informação por blocos de acordo com os itens lógicos as que se referem, para que se torne mais fácil a leitura, a compreensão e memorização do texto.
- Agrupar os elementos por proximidade física de acordo com os seus significados.
- Efeito de estica, isto é, tornar o programa agradável de utilizar.
- Velocidade no acesso a informação, que permitirá ao utilizador não perder muito tempo.

Certos sistemas de visionamento de informação utilizam um barra de deslizamento para visualizar informação, mas tem vindo a ser cada vez menos utilizada pelo facto que os utilizadores têm a tendência para considerar apenas o conteúdo que é apresentado assim que a aplicação é aberta. Deste modo, é vantajoso construir *interfaces* do tamanho do ecrã de visionamento. Não sendo possível atender à exposição de toda a informação no ecrã, deverão se colocar os conteúdos mais importantes nas imediações visíveis da janela.

Atendendo a todos estes fatores e voltando ao tópico discutido no final do ponto 4.4., acerca do ruído visual, este conceito pode ser interpretado de forma errónea no sentido de não apresentar muita informação, mas pelo contrário, é vantajoso apresentar a maior quantidade possível de informação organizada e direcionada para a ação, voltando a reiterar informação organizada, sem ruído visual, ou seja, informação agrupada coerentemente. Desta forma, foi decidido apresentar toda a informação de forma direta, encontrando-se todas as caixas de inserção de valores diretamente visíveis, sem ter que se selecionar nenhuma opção extra, da mesma forma os resultados principais são apresentados diretamente e variam dinamicamente conforme o utilizador constrói um dado problema. Desta maneira, cria-se um ambiente que incentiva o espírito crítico do utilizador, atendendo ao modo como os resultados variam dependendo do seu *input*.

4.5. METODOLOGIA DE CÁLCULO

A obtenção dos resultados de determinados módulos é efetuado conforme uma sequência de cálculo lógico. Os princípios que serão discutidos neste ponto irão ilustrar o procedimento utilizado para determinar os resultados das tensões para qualquer secção transversal, como também irá ser mencionado a metodologia para a obtenção dos resultados de cada módulo em particular. É necessário relembrar de antemão que as metodologias que serão apresentadas, baseiam-se num paradigma direcionado para a programação, e em virtude disso, a sua apresentação ira refletir nesse mesmo fator.

Todos os pontos marcados respeitam um sistema de coordenadas locais, o qual é posteriormente transformado para um sistema corrente de coordenadas, para cada tipo de aplicação. Dessa

forma, a lógica de programação terá em conta esse fator no processo de criação do algoritmo, como se verá mais adiante.

4.5.1. METODOLOGIA DE CÁLCULO PARA FLEXÕES NORMAIS E FLEXÕES COMPOSTAS DESVIADAS

Para o cálculo destas tensões é necessário recordar a fórmula 3.27. localizada no Capítulo 3, considerando que existem momentos numa determinada secção transversal. Primeiro determina-se a área do elemento, e posteriormente, calcula-se a tensão utilizando a expressão, 3.18, tendo sempre em conta o sinal da força aplicada. Se existir agora um momento no eixo horizontal que pode ser designado por y, será necessário proceder à seguinte metodologia para determinar os esforços de flexão num determinado ponto, sendo elas,

- Determinar o momento de flexão, com o respetivo sinal correto conforme o referencial da secção que o contem pode indicar.
- Determinar a localização do eixo neutro.
- Calcular o momento de inércia I da área da secção transversal em torno do eixo neutro em questão.
- Determinar a coordenada y do ponto dado, sendo que y será positivo caso o ponto se encontre acima do eixo neutro e negativo se ele se encontrar abaixo da linha neutra.
- Calcular por fim a tensão através da equação (3.24.), se o sinal correto for utilizado para M e y, então a tensão também terá o sinal correto. (Relembra-se que tensão de tração tem sinal positivo, e tensões de compressão possuem sinal negativo).

Caso exista um momento em torno do eixo z, o cálculo da sua parcela será semelhante ao descrito acima. Apesar de estes princípios aparentarem ser claros, quando se trata de programar para qualquer secção transversal, é necessário a utilização de métodos mais específicos. Ora vejamos então.

4.5.1.1. COMPUTAÇÃO DE ÁREAS, CENTRÓIDE E INÉRCIAS

Tal como referido no Capitulo 3 deste documento, as secções são desenhadas a partir do posicionamento dos pontos que englobam a figura (ou seja, os seus vértices). Tendo em conta o caso apresentado pela Figura 4.1. é possível determinar a área através das coordenadas de cada ponto, pela utilização da seguinte expressão,

$$A = \frac{1}{2} \left| \sum_{i=0}^{n-1} (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \right| \tag{4.1}$$

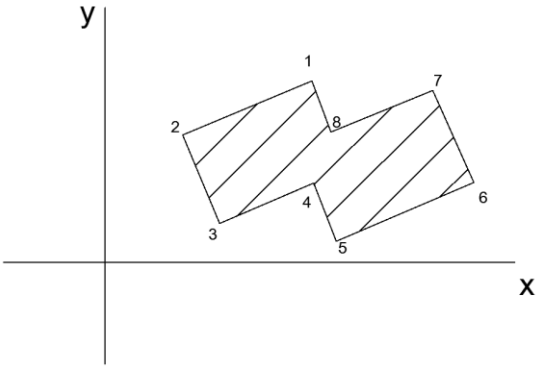


Fig.4.1 – Delimitação da área de um polinómio

Onde x_i será x_1, que é a coordenada x do vértice 1 e y_{i+1} será y_2, que é a coordenada do seguinte ponto, e assim por diante, sendo que o último ponto envolve o primeiro ponto novamente. Dependendo do sentido horário ou anti-horário da progressão dos pontos, o resultado final poderia ser negativo, mas correto em termos de valor absoluto. Quando se calcula as posições do centróide, terá que se utilizar o valor resultante com o seu respetivo sinal, neste caso, o mesmo seria negativo, portanto, deve-se utilizar esse mesmo valor nas seguintes expressões,

$$C_x = \frac{1}{6 \cdot A} \sum_{i=0}^{n-1} (x_i + x_{i+1}) \cdot (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \tag{4.2.}$$

$$C_y = \frac{1}{6 \cdot A} \sum_{i=0}^{n-1} (y_i + y_{i+1}) \cdot (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \tag{4.3.}$$

Do mesmo modo, os segundos momentos de inércia podem ser calculados em uma forma genérica pela soma das contribuições de cada segmento de um polinómio. Desta forma, as inércias segundo cada eixo podem ser obtidas através das seguintes equações,

$$I_x = \frac{1}{12} \sum_{i=1}^{i=N} (y_i^2 + y_i \cdot y_{i+1} + y_{i+1}^2) \cdot (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \tag{4.4.}$$

$$I_y = \frac{1}{12} \sum_{i=1}^{i=N} (x_i^2 + x_i \cdot x_{i+1} + x_{i+1}^2) \cdot (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \tag{4.5.}$$

$$I_{xy} = \frac{1}{24} \sum_{i=1}^{i=N} (x_i \cdot y_{i+1} + 2 \cdot x_i \cdot y_i + 2 \cdot x_{i+1} \cdot y_{i+1} + x_{i+1} \cdot y_i) \cdot (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \tag{4.6.}$$

Da mesma maneira, se os pontos estiverem ordenados com um sentido horário os valores retornarão positivos, caso contrário os valores serão de grandeza idênticas, no entanto, negativos. Determinados estes valores para um número indeterminado de pontos, será necessário armazenar os resultados de cada ponto, para que seja possível identificar os valores máximos, e os resultados em cada um dos pontos inseridos. A sua consulta é possível através da opção encontrada no canto superior esquerdo com o nome e detalhes.

4.5.1.2. METODOLOGIA DE DESENHO DOS DIAGRAMAS DE TENSÕES

A fim de desenhar o diagrama das tensões é necessário encontrar os pontos da secção mais afastados no eixo neutro, portanto é necessário realizar uma verificação por vetores perpendiculares de cada ponto em relação ao eixo neutro, medindo os seus valores, de modo a que os pontos dos extremos sejam descobertos. Para isso, é necessário num primeiro plano descobrir a equação da linha neutra, a qual pode ser obtida resolvendo a equação (3.27) em função de x e y, de forma a se obter o declive do eixo neutro e assim ser possível criar os ditos vetores perpendiculares.

Utilizando essas medidas, é possível desenhar os diagramas por extensão de linhas paralelas em relação ao eixo neutro provenientes dos pontos extremos (pontos que contêm as tensões máximas) da secção transversal. A ilustração desta metodologia pode ser observada através da Figura 4.2..

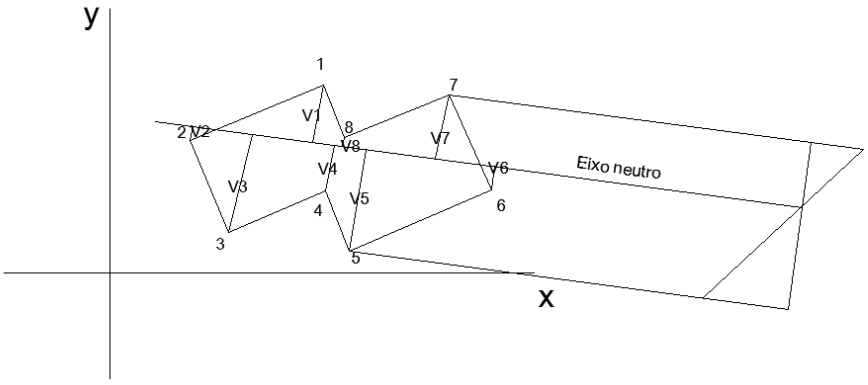


Fig.4.2 – Metodologia de determinação do diagrama de tensões

4.5.2. METODOLOGIA DE CÁLCULO DAS TENSÕES DE CORTE

O módulo da tensão de corte encontra-se integrado no algoritmo das tensões normais, e pode ser determinado para qualquer um dos pontos definidos pelo utilizador. O procedimento utilizado para a sua determinação é o seguinte,

- Determinar o esforço tangencial V_y, que atua na secção transversal e que contem o ponto especificado.
- Localizar o eixo neutro e calcular o momento de inércia I_x da área da secção transversal em torno do eixo neutro em questão.

- Calcular o primeiro momento Q_x da área da secção transversal que se encontra acima (ou abaixo) do ponto especificado.
- Determinar a tensão de corte a partir da expressão 3.41, em b é a largura da secção transversal no ponto em questão.

Relembra-se que τ é a tensão de corte "real", apenas se b se mantiver constante em toda a secção transversal, caso contrário, τ representa a tensão tangencial média. A tensão máxima de corte, ou seja, τ_{max} numa determinada secção, ocorre onde a relação matemática entre o primeiro momento da área e da largura b é maior. Assim sendo, se a largura de b for constante, então o τ_{max} está localizado no eixo neutro, devido a que nesse local Q apresenta o seu valor máximo. Caso "b" não seja constante, então é necessário determinar a tensão de corte em mais que um ponto de modo a determinar o valor máximo da tensão de corte, o que poderá ocorrer numa zona de redução de largura na secção transversal, ou num ponto de descontinuidade em torno de um vazio no elemento.

4.5.3. METODOLOGIA PARA A OBTENÇÃO DA REPRESENTAÇÃO GRÁFICA DO CÍRCULO DE *MOHR*

As expressões para desenhar o círculo de *Mohr* já foram deduzidas no Capítulo 3, no qual foi dado igualmente uma breve menção da metodologia para o desenho do círculo. Para reiterar este ponto no sentido metodológico, observe a Figura 4.3.

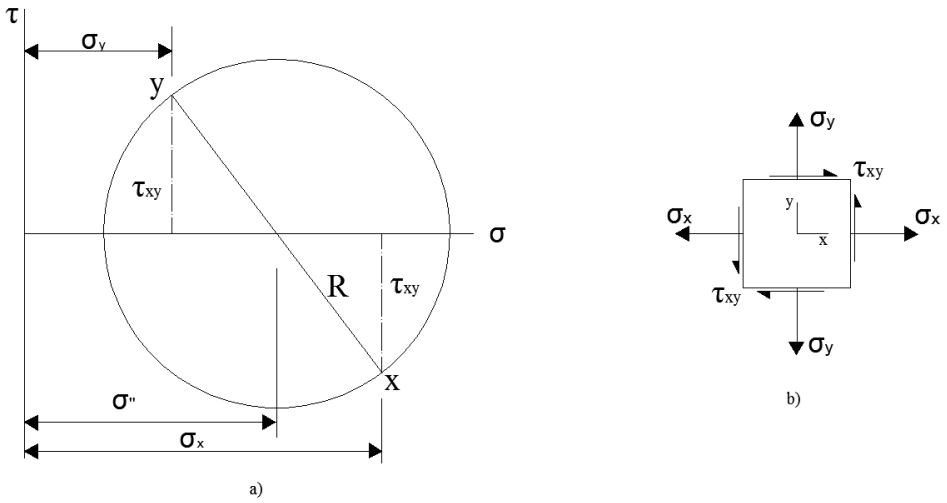


Fig.4.3. – Construção do círculo de Mohr através das componentes do estado de tensão

Considerando o estado plano de tensões num ponto definido pelo ponto b) da Figura 4.3., que é caracterizado pelos componentes de tensão σ_x, σ_y, e τ_{xy}. Para este estado de tensão o círculo de *Mohr* é construído seguindo a seguinte metodologia,

- Desenhar um conjunto de eixos, no qual o eixo das abcissas representa a tensão normal e o eixo das ordenadas representa as tensões tangenciais de corte.
- Marcar o ponto X de coordenadas (σ_x, $-\tau_{xy}$), e marcar o ponto Y de coordenadas (σ_y, $-\tau_{xy}$). As coordenadas destes pontos representam as tensões atuantes na face x e y do

elemento. Esta sua representação deve ser acompanhada pelos seus valores de coordenadas, de modo a facilitar a sua identificação.

- Unem-se os pontos X e Y com uma linha reta, e desenha-se o círculo utilizando este diâmetro, ficando essencialmente desenhado o círculo de *Mohr*.
- Identifica-se o raio do círculo pela sigla R, obtido através da expressão 3.56, o centro do círculo localizado em (σ'', 0), sendo''σ obtido através da equação 3.62, identifica-se de igual forma o ângulo alfa.
- Determina-se as tensões principais utilizando a transformação dos eixos xy para x'y', e identificação o ponto (σ_1, 0) e o ponto (σ_2, 0).
- Por meio de τ_{max} = R, aponta-se o ponto para qual a tensão de corte é máxima, pelas coordenadas ($\sigma_{y'}$, $\tau_{x'y'}$).

5
UTILIZAÇÃO DA FERRAMENTA DE CÁLCULO

5.1. INTRODUÇÃO

Neste capítulo pretende-se demonstrar o funcionamento da ferramenta de cálculo, as formas de acesso à ferramenta, e as nuances de utilização. Atende-se, desde já ao facto, que o domínio do *website* apresentado possui características meramente indicativas e académicas, visto que a sua hospedagem é virtual e acompanhada de um serviço de *hosting* privado. A mesma poderá já não se encontrar futuramente *online* no endereço apresentado, pelo que a sua disponibilidade também pode ser obtida num regime *off-line* para uso no computador ou num dispositivo móvel.

5.2. DIREÇÕES DE ABERTURA

Como mencionado anteriormente, esta ferramenta pode ser aberta através de um motor de busca, e dessa forma, utilizar a própria ferramenta num âmbito de um *Web browser*. Portanto, primeiro abre-se um explorador e coloca-se o endereço do *website*.

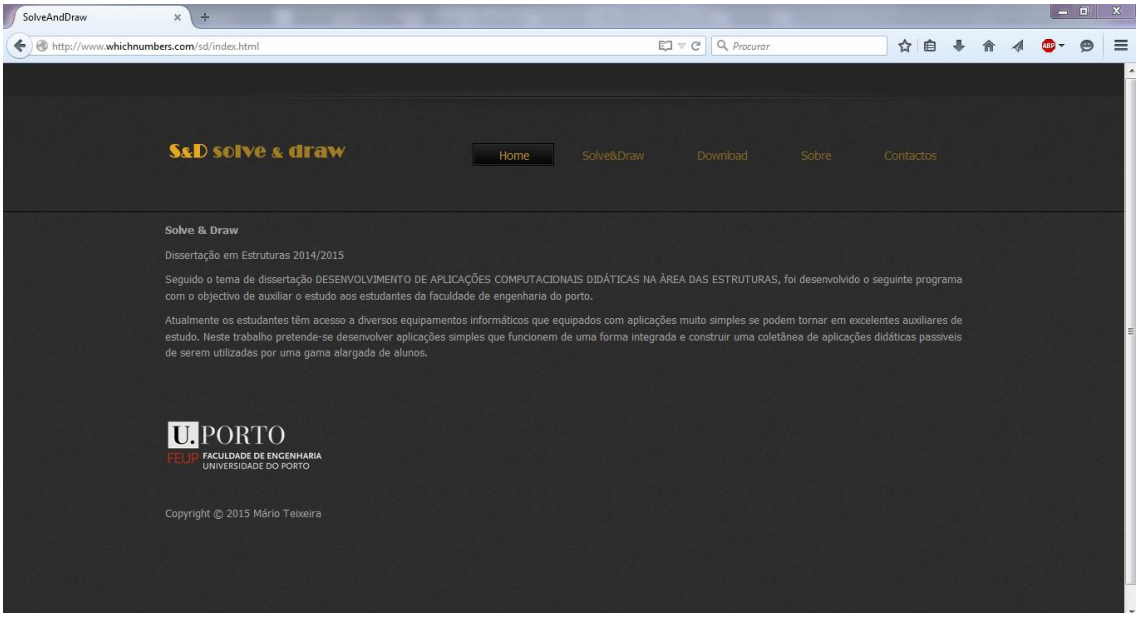


Fig.5.1. – Página *Web* onde o programa está hospedado

O mesmo foi acedido no dia 23 de Junho de 2015, representado pela Figura 5.1., com o nome de www.whichnumbers.com/sd.

Existem cinco menus para folhear, sendo os mesmos,

- *Home*. Página principal de abertura.
- *Solve&Draw*. Acesso ao programa através do *website*, e uso do mesmo através do *Web browser*.
- *Download*. Parcela que contem versões do programa para descarga e uso direto através do dispositivo informático compatível ao que foi descarregado.
- *Sobre*. Parcela contendo uma pequena síntese das faculdades inerentes ao programa (Fig. 5.4).
- *Contactos*. Informação sobre as formas de contactar o autor deste documento acerca de qualquer dúvida de funcionamento ou problemas na utilização do *website*, ou do programa.

Selecionando a opção *Solve&Draw*, tem-se acesso ao programa embutido na página *Web*, demonstrado na Figura 5.2.

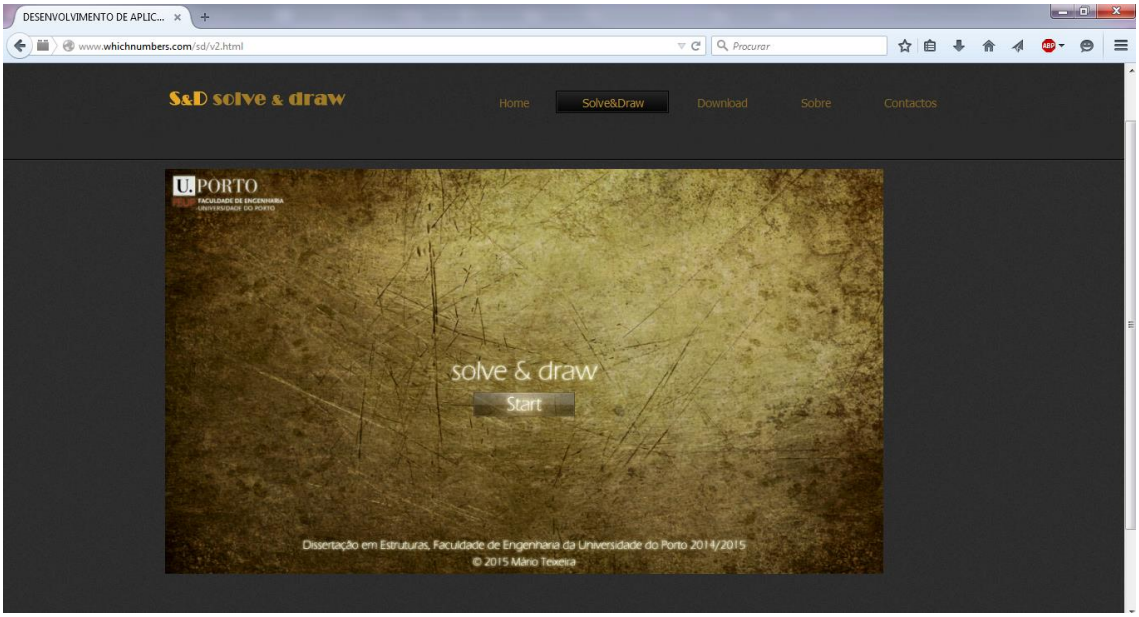


Fig.5.2. – Interface inicial da ferramenta de cálculo no *website*

Como se pode observar pela Figura 5.2., a aplicação apresenta uma interface gráfica orientada para dispositivos móveis (está mesma interface é idêntica para todas as plataformas) no sentido de se permitir obter uma apresentação atraente bem como percetiva e simples de utilizar. A imagem de entrada na ferramenta tem a funcionalidade de criar uma separação entre a abertura da aplicação e as opções, de maneira a ambientar o utilizador desde logo o início para o esquema visual e o ambiente contido pela ferramenta.

O esquema da interface é idêntico para todos os tipos de dispositivos, podendo apenas variar posicionamentos pontuais de certas ações ou de certas representações gráficas, apenas percetíveis por comparação entre sistemas operativos.

5.3. APRESENTAÇÃO DA *INTERFACE* DE ESCOLHA DE TAREFAS

O princípio de execução desta página foi de escolha intuitiva, por parte do utilizador através da visualização de objetos que lhe transmitam sensorialmente o módulo a que ele pretende utilizar, como pode ser visto pela Figura 5.3.



Fig.5.3. – Interface para escolha de módulos

As opções visuais representam uma forma fácil e rápida de escolher secções padronizadas mais correntemente utilizadas, evitando assim o seu desenho. As secções disponiveis serão abordadas em maior detalhe mais adiante.

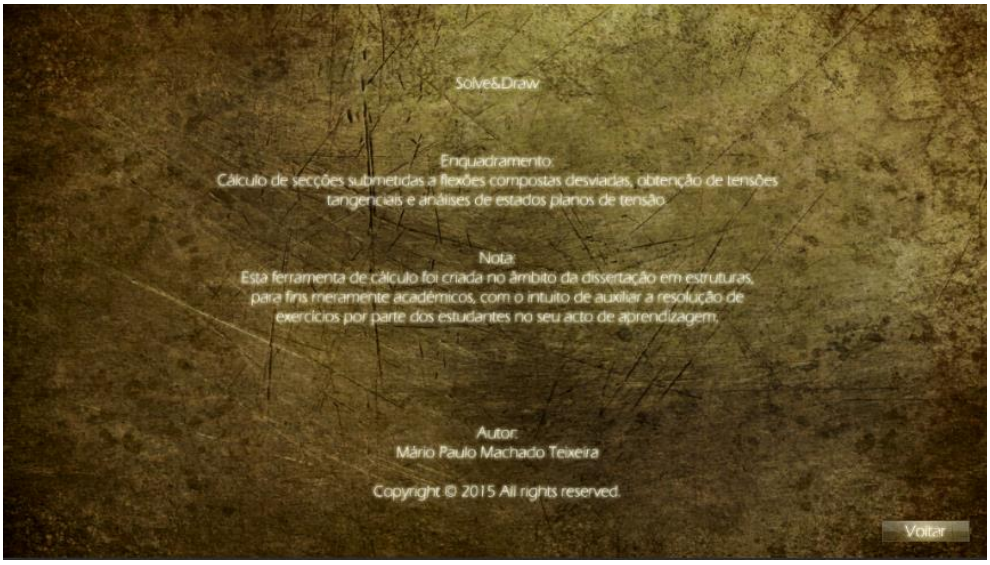


Fig.5.4. – Representação do Separador “Sobre”.

O desenho polinomial permite desenhar de forma livre uma secção transversal de quaisquer dimensões e formas, e a opção círculo de *Mohr*, como o nome indica, permite executar os cálculos inerentes aos estados planos de tensão para desenho do círculo de *Mohr*.

No canto inferior direito é possível observar as opções "*Website"* e *Sobre*, tal como o nome indica, a opção *Website* redireciona o utilizador para o *website* definido acima de forma direta pela abertura de uma janela do motor de busca padrão definido pelo dispositivo informático do utilizador. O ponto "Sobre" descreve um pouco o funcionamento da ferramenta e o âmbito da sua realização como pode ser observado pela Figura 5.5.

5.4. *INTERFACE* DO MÓDULO DE DESENHO POLINOMIAL

A interface do desenho livre de secções transversais é bastante semelhante aos de outros módulos, apresentando apenas pequenas diferenças de utilização, pelo que a definição de pontos nestas últimas secções já foram definidos pelo próprio programa.

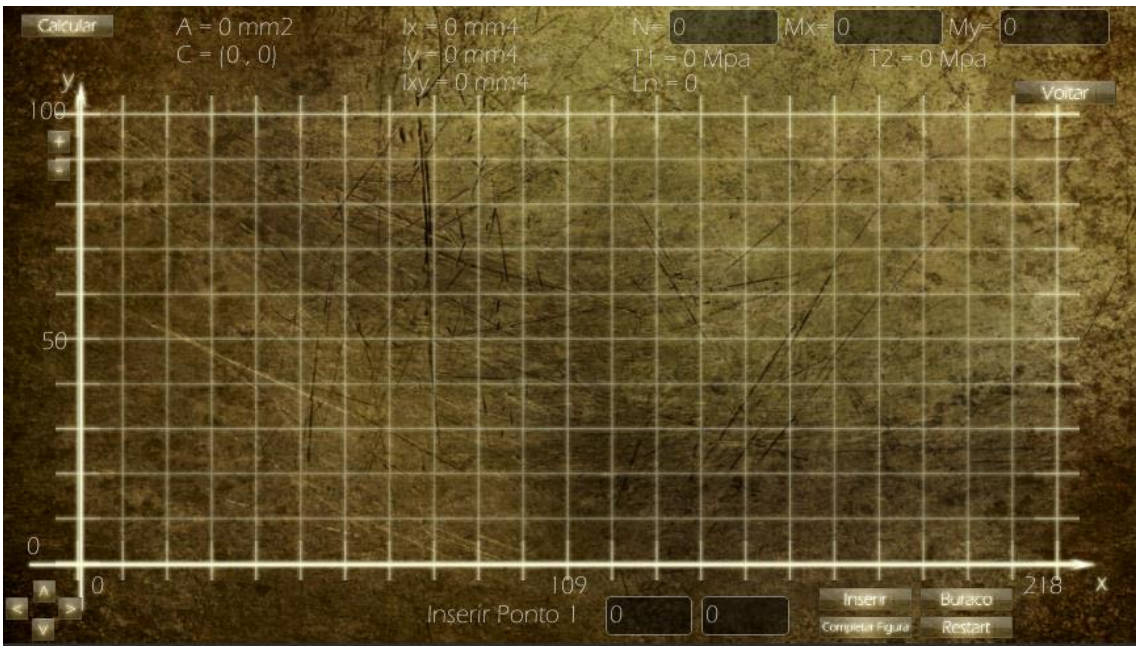


Fig.5.5. – Interface de cálculo para o módulo de desenho polinomial

Olhando para a Figura 5.5., podemos observar de imediato a malha e a escala de eixos x e y. Esta pode ser alterada com o auxílio ao *zoom out* e *zoom in*, por intermédio da roda do rato ou pela utilização dos botões existentes no canto superior esquerdo, onde o + simula a rotação da roda do rato para cima, e o sinal – simula a rotação da roda do rato para baixo. Para realizar *pan*, ou seja, para mover a imagem, pode ser utilizado novamente a roda do rato, mantendo-a pressionada e movendo o rato, ou através das setas no canto inferior esquerdo que simulam esta mesma funcionalidade descrita.

No canto superior esquerdo, encontra-se a tecla "Calcula" que dá início ao cálculo da secção desenhada. De seguida, encontram-se já representados A, que indica a área da secção, C que indica as coordenadas do centróide, Ix, que demonstra a inércia em torno do eixo x; a mesma semelhança que Iy e Ixy apresentam as inércias em tordo do eixo representado.

No canto superior direito, pode-se observar três caixas de inserção de dados, sendo eles, N, esforço normal, Mx, momento fletor em torno do eixo x, e M_y, momento fletor em torno do eixo y. Logo por debaixo dessas "caixas", encontram-se apresentados os valores das tensões principais apresentadas como T1 e T2 que terão a representação visual no diagrama de tensões. Ainda por baixo dos mesmos, está apresentando LN, que irá expor a expressão do eixo neutro. Na imediação, encontra-se a tecla voltar, que retorna a vista anterior de escolha de módulos apresentado pela Figura 5.4.

Na parte inferior da interface está apresentado "Inserir Ponto 1", onde aparecem duas caixas de *input*, sendo as mesmas as coordenadas x e as coordenas y respetivamente, caso se pretenda uma inserção de valores numéricos. É também possível a inserção de pontos através do rato, ou do dedo, por *click* no ponto onde se pretende inserir o primeiro ponto da secção a desenhar. Na sua proximidade encontram-se quatro opções, "inserir", como o nome indica, que insere o ponto de coordenadas introduzidas nas caixas de *input*. A opção "Completar Figura" permite unir automaticamente os dois últimos pontos sem ser necessário o *input* de mais coordenadas, algo que também pode ser realizado com o auxílio do pressionamento do botão direito do rato que representa uma tecla rápida para a execução dessa tarefa. A opção "Buraco" cria uma nova ação de inserção de pontos para criar uma abertura na secção desenhada. Finalmente, o botão "*Restart*" reinicia o desenho da secção.

5.5. MÉTODOS PARA A DEFINIÇÃO DA GEOMETRIA NO MÓDULO DE DESENHO POLINOMIAL

Como já mencionado, existem várias formas de inserção de pontos, um deles é realizado por inserção numérica como pode ser visto na Figura 5.6.

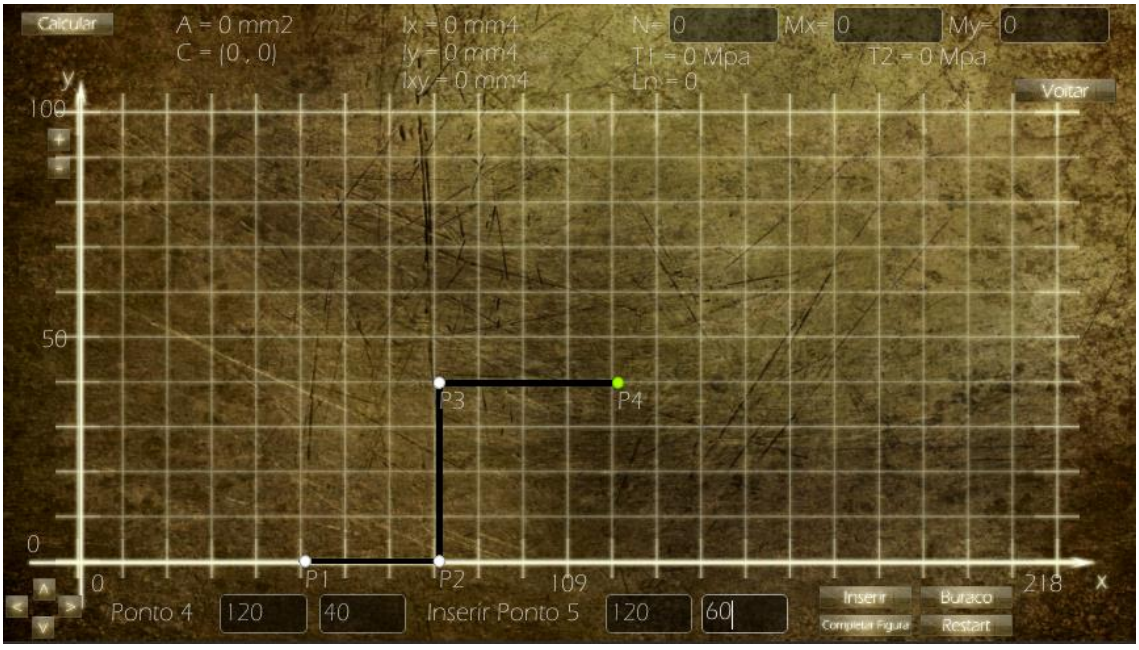


Fig.5.6. – Introdução de pontos

A introdução numérica será sempre mais fidedigna da secção que se pretende desenhar, visto que existe um erro inerente da sua colocação à mão livre, por meio do rato ou por meio de introdução por *touch* com o dedo. Refira-se o aparecimento no canto inferior esquerdo de duas novas caixas de *input* de coordenadas, as quais possibilitam a correção do ponto selecionado a

verde, por defeito está definido o ponto anteriormente colocado, podendo o novo ser selecionado através da secção por meio do rato a qualquer um dos outros pontos. Este processo torna mais vantajoso, em termos de tempo, o desenho à mão livre (tal com ilustram as figuras 5.7 e 5.8), fazendo posteriormente uma correção rápida dos pontos inseridos, não sendo necessário proceder ao pressionamento de qualquer opção, uma vez que as coordenadas dos pontos mudam automaticamente a posição dos pontos.

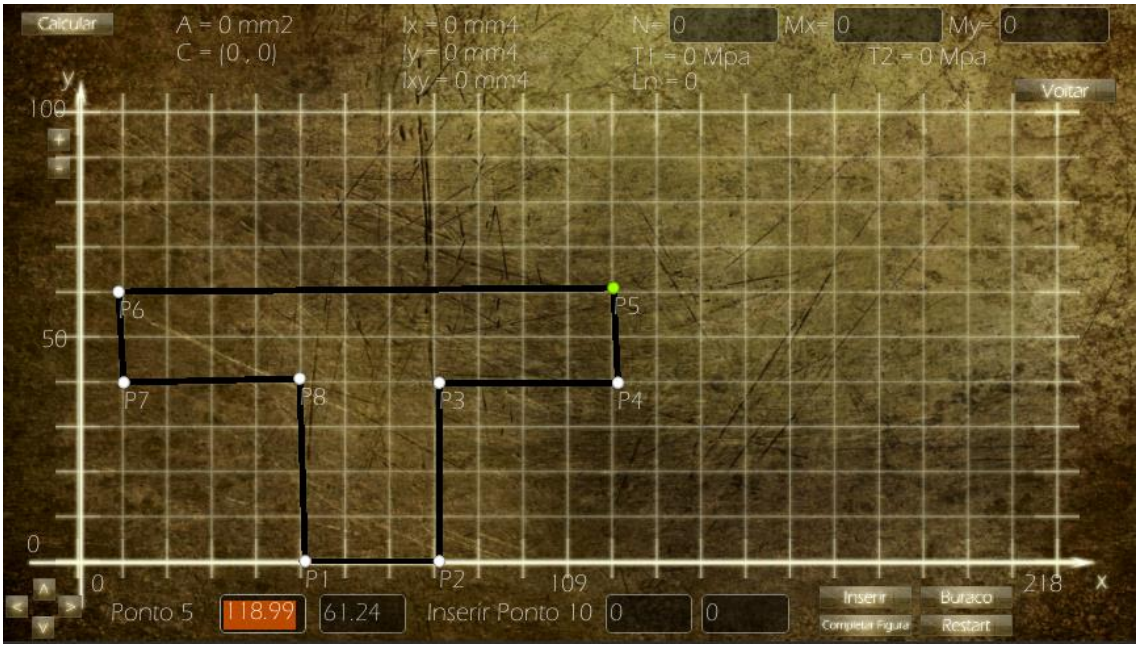


Fig.5.7. – Inserção manual de pontos

Rapidamente, usando o método de colocação de pontos que se realizou na secção, agora procede-se à seleção de cada ponto introduzido de modo a corrigir as coordenadas.

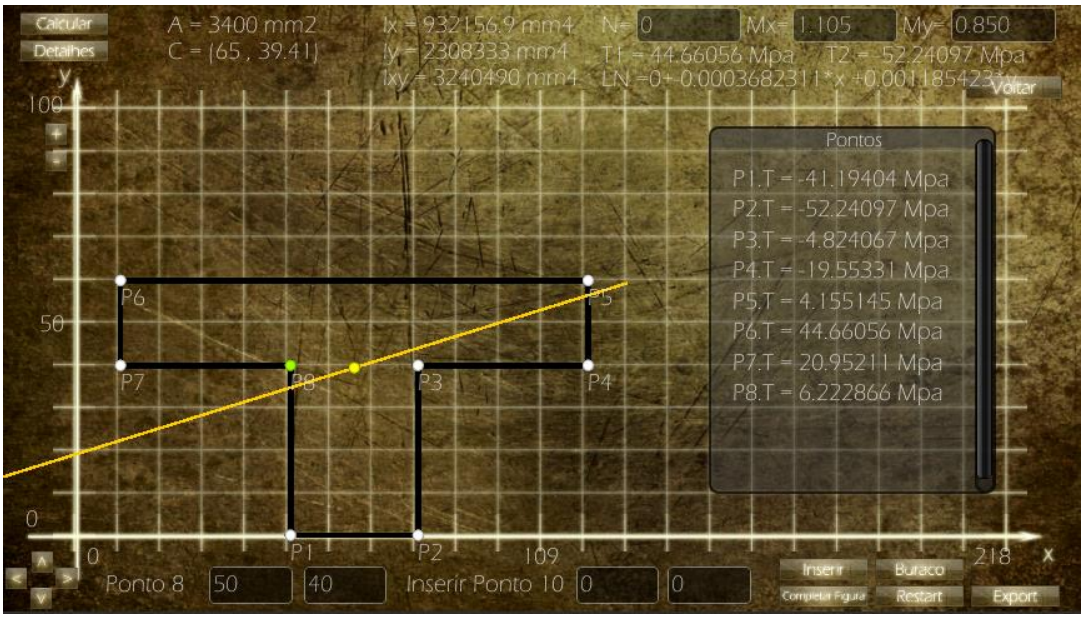


Fig.5.8. – Obtenção de resultados

Após corrigir os pontos, têm-se a secção representada com os pontos nas suas coordenadas exatas, pelo que posteriormente se podem colocar os valores de N, M_x, e M_y. Neste caso, podemos observar a ocorrência representada na Figura 5.8., a de uma flexão composta desviada.

Ao carregar no botão "calcular", é desenhado o diagrama de tensões e a linha neutra representada pela linha amarela. Para obter mais informação das tensões nos outros pontos pode-se pressionar a tecla "Detalhes" que fará com que apareça uma janela amovível no qual estarão descritas as tensões em cada um dos pontos desenhados. Para obter a informação numa zona específica bastaria introduzir um nó, ou por outras palavras, um ponto. No canto inferior direito, também surgiu uma nova opção para exportar os valores para um bloco de notas.

Como demonstrado pela Figura 5.8, o diagrama foi desenhado fora da área de visualização, portanto, procedendo a um *zoom out*, e a um consecutivo *pan*, consegue-se obter o resultado representado pela Figura 5.9.

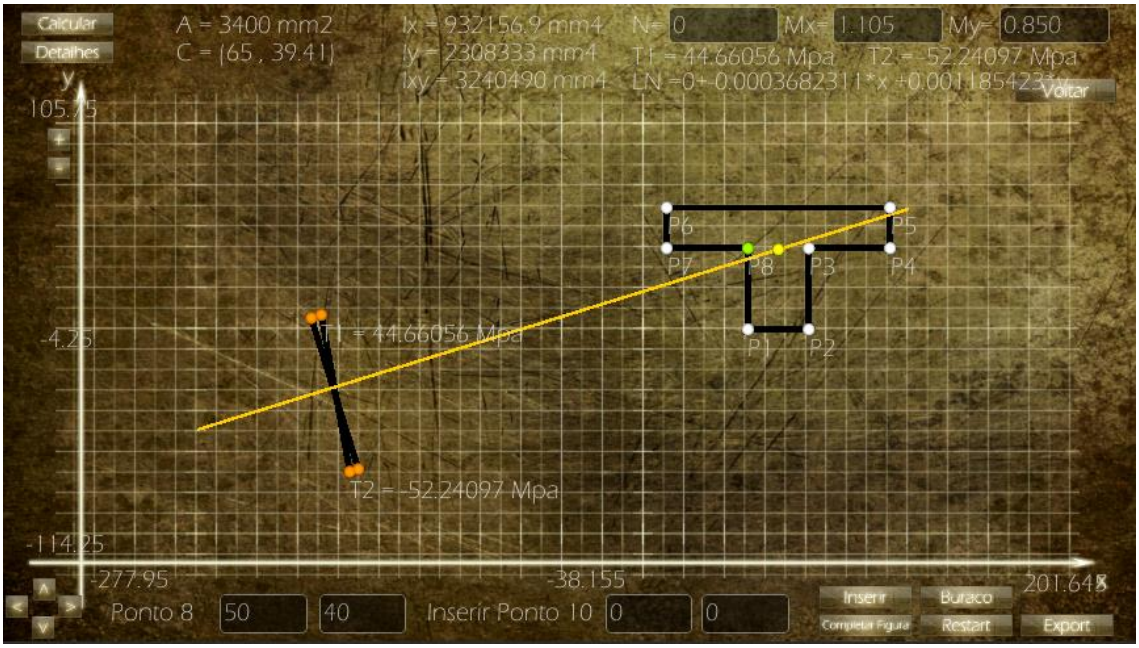


Fig.5.9. – Representação do diagrama de tensões

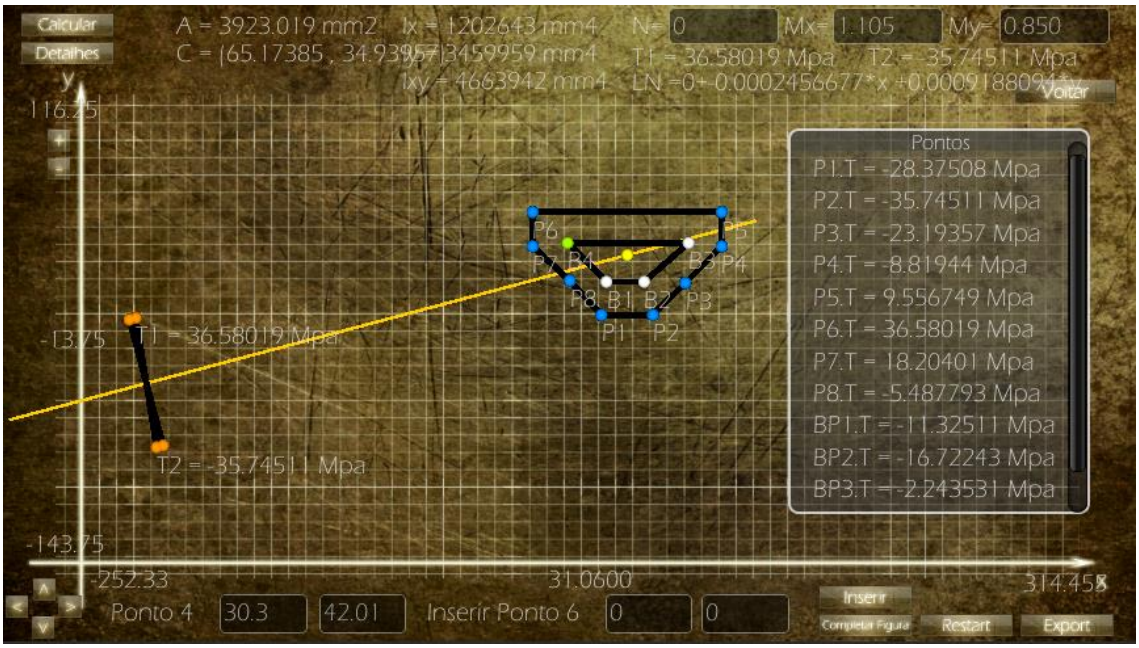


Fig.5.10. – Abertura de uma secção

Pode-se nesta instância, por exemplo, proceder à alteração da secção e transformá-la num trapézio. Este procedimento é realizado selecionando e arrastando os pontos 8 e 3. Posteriormente, introduz-se uma abertura na secção pela utilização da opção "Buraco". Assim, podemos obter o que mostra a Figura 5.10. As alterações são apresentadas de forma dinâmica, no qual o diagrama e a linha neutra são alterados conforme se introduz os valores.

5.6. SECÇÕES PADRONIZADAS

Recorrendo ao menu apresentado na Figura 5.4., pode-se selecionar uma secção em I, no qual se passará para o mostrado pela Figura 5.11.

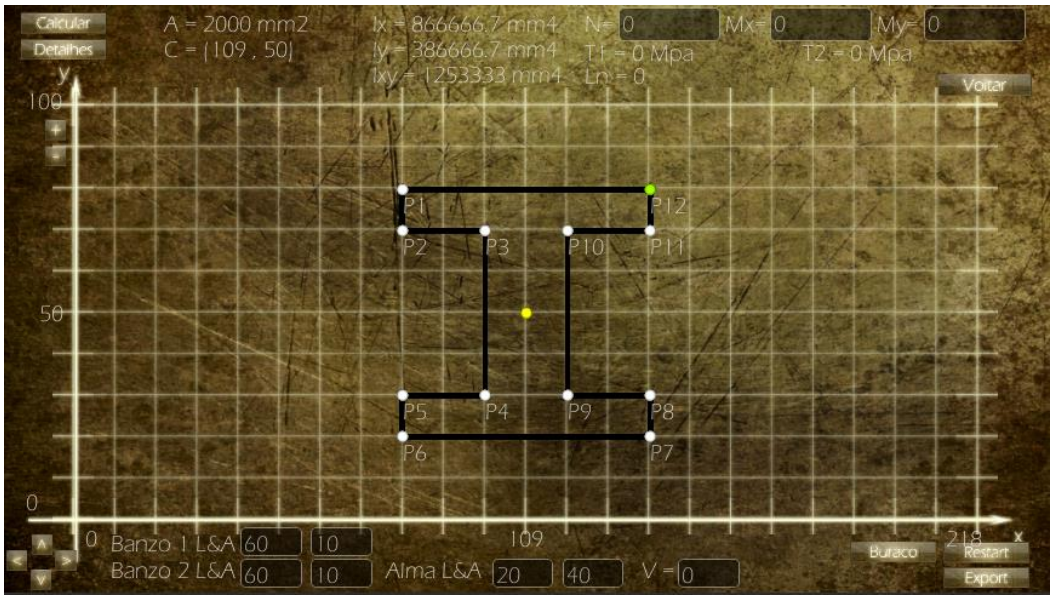


Fig.5.11 – Cálculo de secções padronizadas, I

Segundo o que foi mostrado na Figura 5.5. e para o que foi apresentado pela Figura 5.11., as principais alterações verificam-se no canto inferior da imagem, onde se pode verificar que já não é possível inserir pontos, mas agora é permitida uma inserção de valores mais genérica, definindo apenas a geometria das parcelas da secção por largura e altura, respetivamente o eixo x e y. Ademais, existe num *input box* uma nova representação por V, o que indica o esforço transverso na direção y atuante na secção. Desta forma, se assim se pretender consegue-se obter o diagrama das flexões em conjunção com o diagrama das tensões de corte.

É possível colocar uma abertura na secção por intermédio da opção "buraco", no entanto, esta estará restrita a um desenho simétrico na alma da secção, onde na altura da sua seleção, será permitido a introdução de valores para o seu posicionamento em relação a um dos banzos, tal como a sua largura e altura, não permitindo que estes valores sejam superiores aos valores introduzidos para as dimensões da alma. O comportamento é semelhante para as restantes secções padronizadas, sendo uma exceção ao círculo onde o desenho das tensões é menos pertinente e onde se introduz a possibilidade de cálculo de uma secção com dois materiais. Pode-se então observar o descrito através das Figuras, 5.12., 5.13., 5.14. e 5.15.

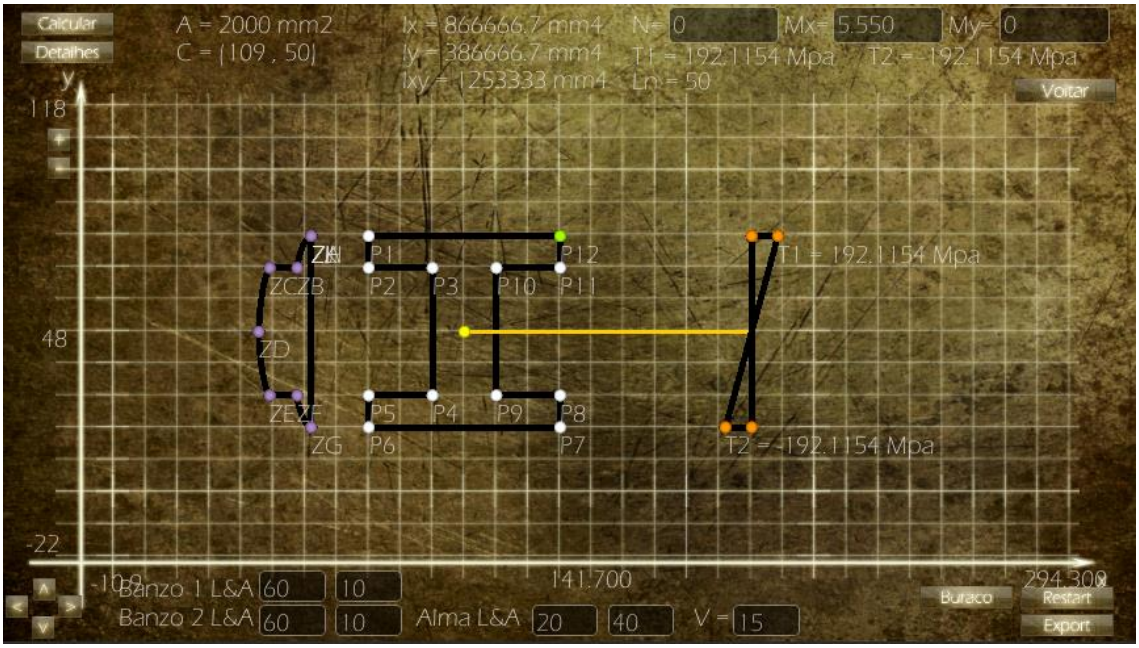


Fig.5.12. – Cálculo da tensão de corte, para secção em I

De modo a evitar uma sobreposição, os valores das tensões de corte estão descritos no menu "Detalhes", onde é possível também obter os valores das tensões normais.

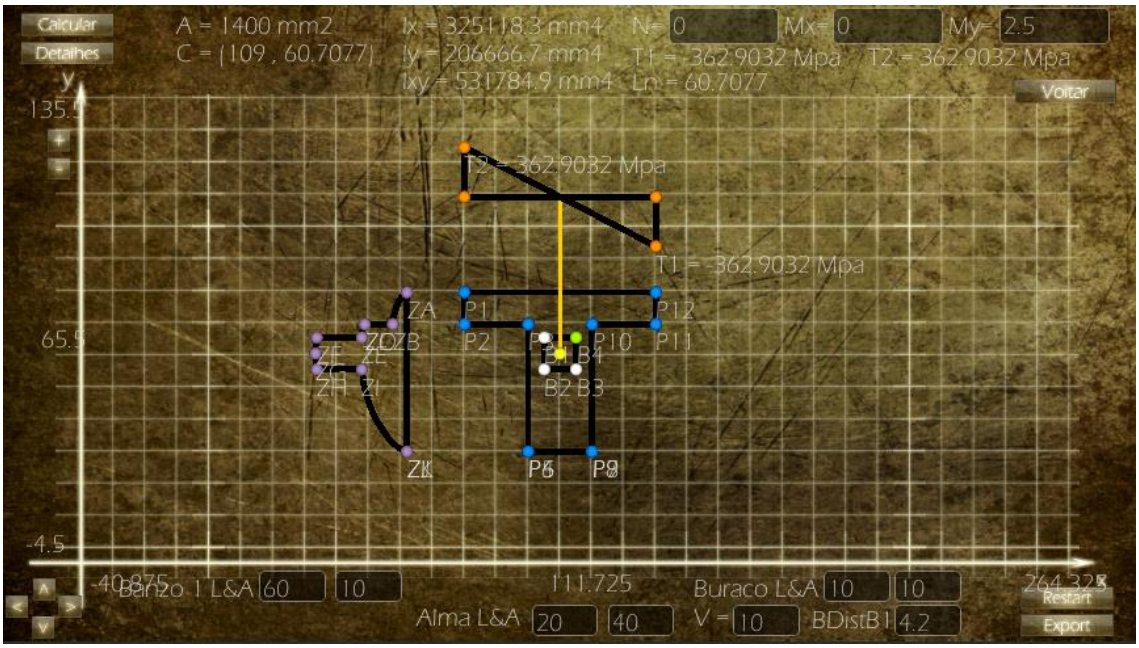


Fig.5.13. – Cálculo da tensão de corte, para secção em T, com abertura na secção

Como ilustrado pela Figura 5.13. e mencionado anteriormente, a seleção do buraco cria três novas caixas de *input* representadas no canto inferior direito, onde se pode ler "Buraco L&A", que significa a largura e altura do buraco. Na sua proximidade pode-se observar "BDistB1", no qual apresenta a distância do buraco ao banzo 1, para as quais as suas medidas são definidas a partir de *input box* no canto inferior esquerdo.

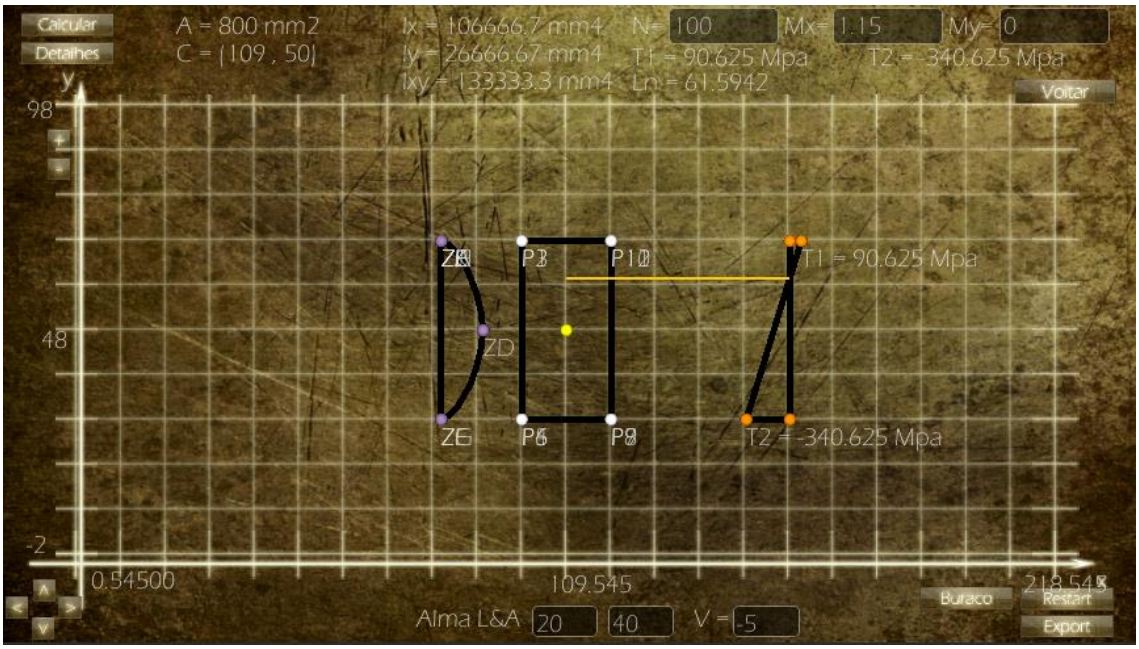


Fig.5.14. - Cálculo da tensão normal e de corte

No exemplo demonstrado pela Figura 5.14., a tensão de corte está invertida e a linha neutra encontra-se localizada acima do centróide da secção.

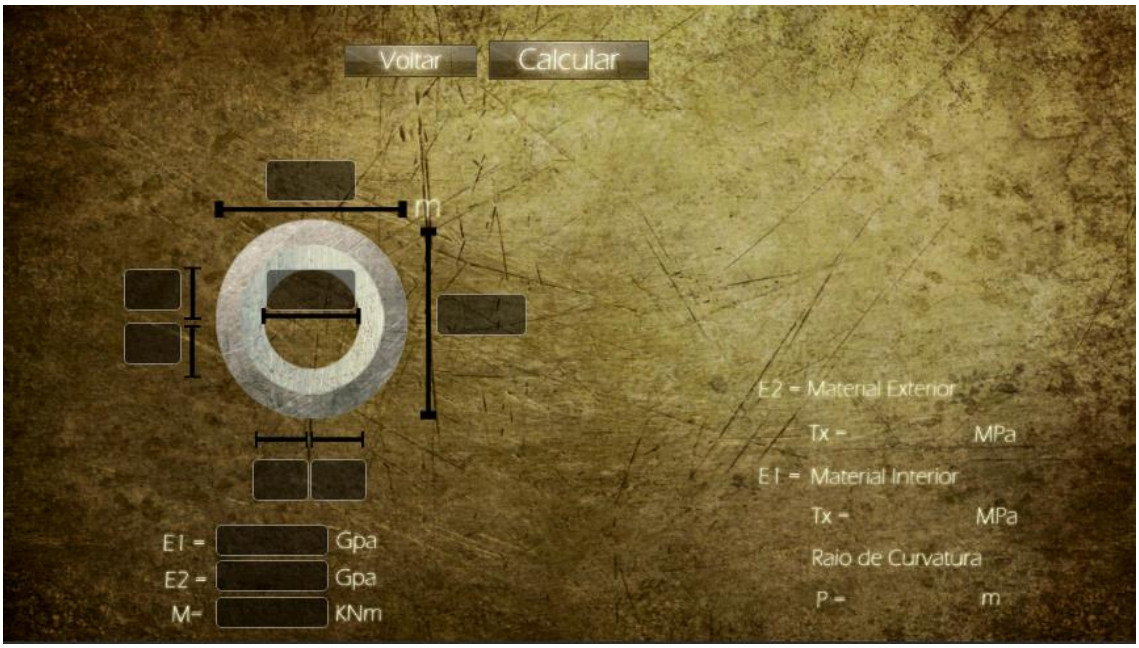


Fig.5.15. – Cálculo de tensões num círculo com dois materiais

As medidas aparentemente extensas do círculo são devido à possibilidade do exterior ou interior ser elipsoidal em vez de circular. E1 e E2 identificam o módulo de elasticidade de cada um dos elementos caso o hajam. Um exemplo é demonstrado no ponto 6.3.

5.7. DESENHO DO CÍRCULO DE *MOHR*

Selecionando o módulo de desenho do círculo de *Mohr* através da interface apresentada pela Figura 5.4., obtêm-se o representado na Figura 5.16.

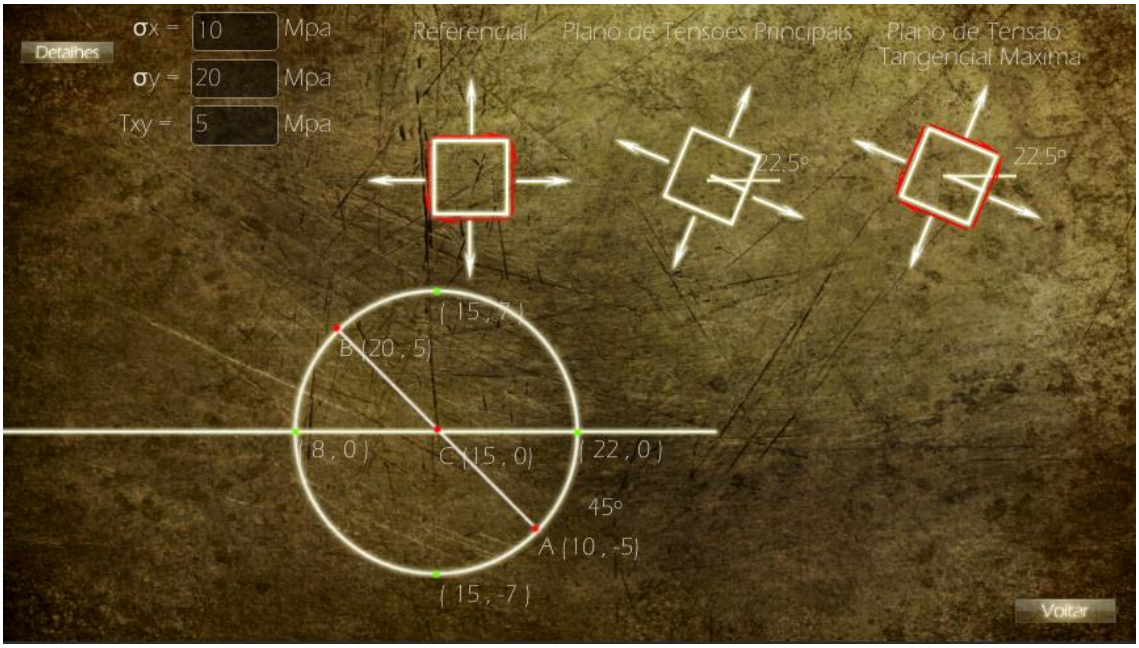


Fig.5.16. – Módulo de cálculo e desenho do círculo de *Mohr*

Ao contrário das suas contrapartes, este módulo não possui uma tecla para o início do cálculo. Assim que são inseridos os valores, os resultados são gerados automaticamente. No canto superior, pode-se observar o referencial que indica os sentidos das tensões atuantes seguindo o referencial de trações e compressões definidos no Capítulo 4, variando os sentidos em função das suas tensões serem negativas ou positivas.

De seguida, estão representados os planos de tensões principais e os planos principais de tensão tangencial máxima. No que diz respeito ao desenho do círculo de *Mohr* em si, ele é definido pela reta que contem o seu diâmetro BA e pelo seu ângulo α igual 45º. No exemplo da Figura 5.16., os pontos a verde representam os máximos valores das tensões. Do mesmo modo que os módulos anteriores, existe uma opção para obter os detalhes relevantes inerentes do cálculo para a obtenção dos valores apresentados graficamente, nomeadamente as tensões em cada um dos pontos desenhados.

5.8. EXPANSIBILIDADE E NOTAS

Como foi demonstrado neste capítulo, a utilização desta ferramenta é simples e direta, entregando um *output* imediato ao *input* do utilizador. Ainda assim existe a possibilidade de expansão da ferramenta através da incrementação de novos módulos na interface da Figura 5.3. Da mesma maneira como foi apresentado o seu funcionamento na página *Web*, a utilização de outros dispositivos é feita diretamente sem sofrer alterações muito significativas no seu código ou no seu aspeto gráfico visual.

6
EXEMPLOS DE APLICAÇÃO

6.1. INTRODUÇÃO

Tal como o título indica este capítulo, destina-se a apresentar alguns exemplos de aplicação a finalidade de verificar o bom funcionamento da ferramenta de cálculo desenvolvida. Os exemplos expostos apresentam as funcionalidades do programa e apontam para as potencialidades de resolução de cada módulo.

6.2. EXEMPLO 1 – CÁLCULO DE UMA SECÇÃO EM T SUBMETIDA A FLEXÃO SIMPLES

Cálculo de uma secção transversal em forma T, com as suas características indicadas na Figura 6.1. sujeito a flexão composta.

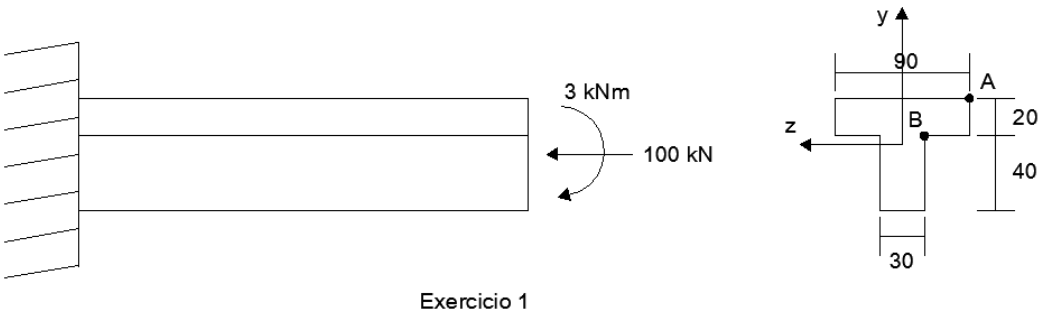


Fig.6.1. – Viga submetida a esforço axial e a flexão simples, secção transversal em milímetros

Atendendo às cargas aplicadas na viga, averiguam-se através da ferramenta *Solde&Draw* os valores máximos da tensão normal de tração e de compressão, e posteriormente, determinam-se os valores das tensões nos pontos A e B.

Para este cálculo, recorre-se ao desenho padronizado de uma secção em T, na interface de escolha, obtendo os resultados apresentados na Figura 6.2.

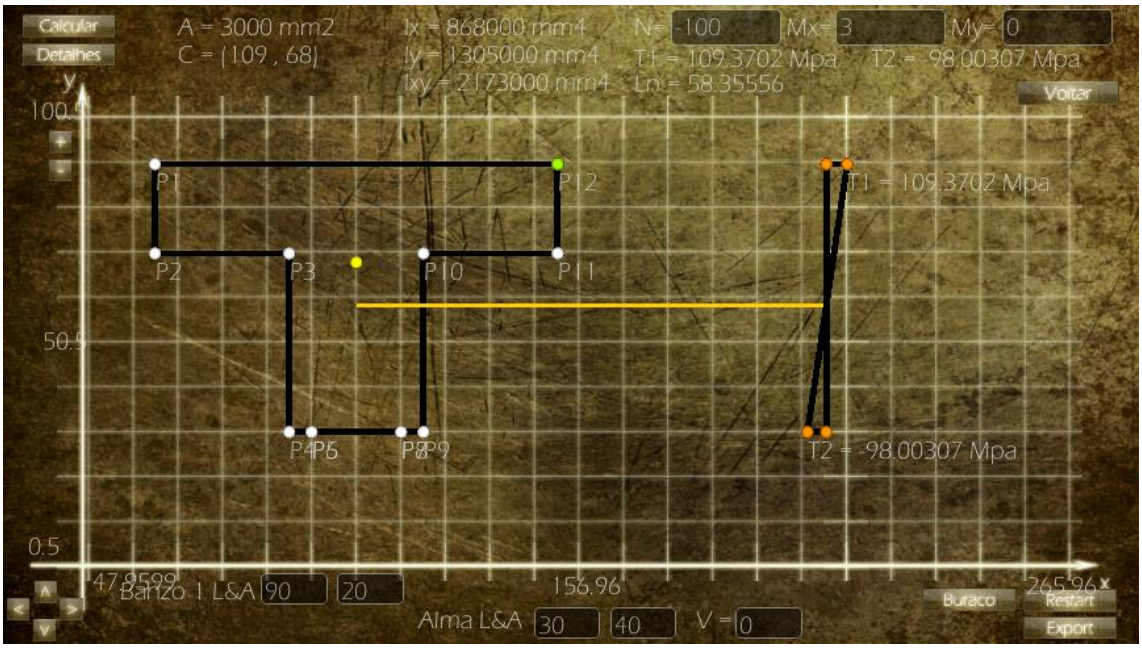


Fig.6.2. – Diagrama de tensões normais para flexão composta e carga axial numa secção T

O valor máximo da tensão na fibra superior é de 109.37 MPa (tração) e o valor máximo na fibra inferior é de -98 MPa (compressão). O eixo neutro encontra-se a 10.35 mm de distância do centróide da secção. Observando-se na Figura 6.3, que a tensão normal em A corresponde ao ponto 12, (a tensão na fibra superior) e que, a tensão normal em B corresponde ao ponto 10 (tensão normal na fibra inferior).

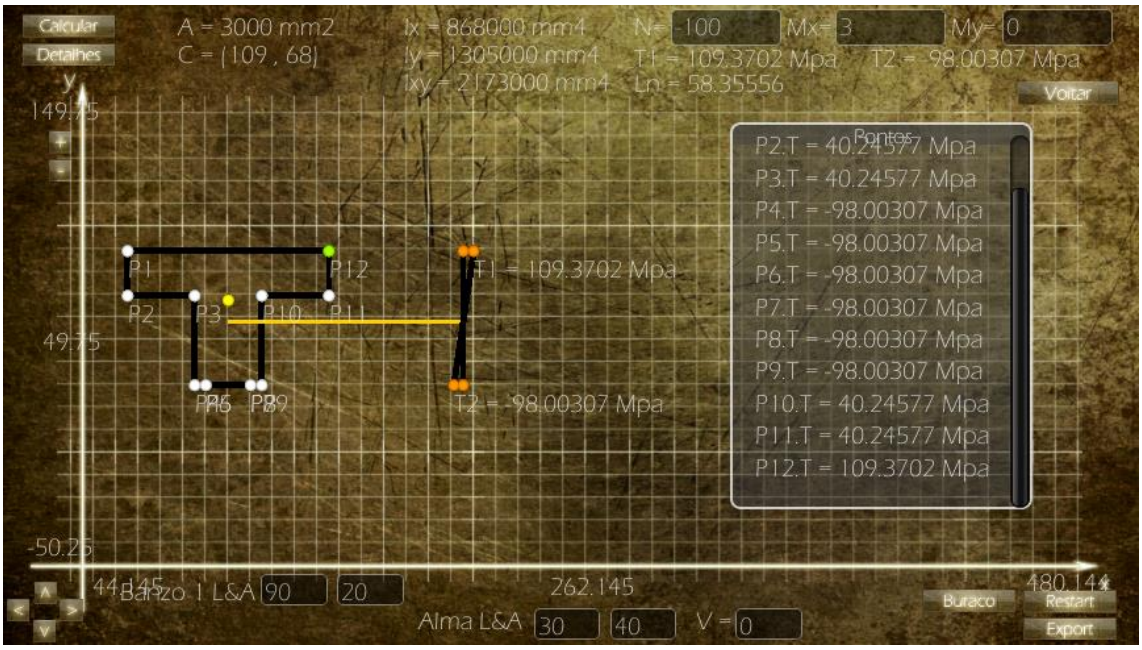


Fig.6.3. – Resultados numéricos das tensões normais para flexão composta e carga axial numa secção T

6.3. EXEMPLO 2 – CÁLCULO DE UMA SECÇÃO CIRCULAR SUBMETIDA A FLEXÃO SIMPLES

Este exemplo possui a particularidade da secção circular ser constituída por dois materiais diferentes e, desta forma, apresentando resistências elásticas diferentes, pelo que é necessário ser executada a homogeneização da secção.

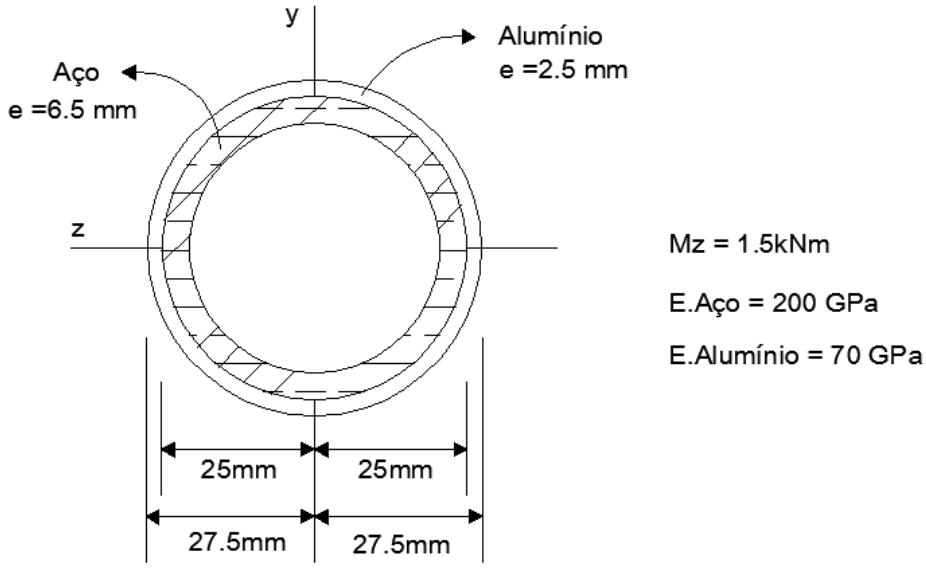


Fig.6.4. – Secção circular composta por dois materiais sujeita a flexão simples

Pretende-se obter as tensões normais máximas para ambos os materiais, em resposta ao momento aplicado de 1.5 kNm, conforme exposto na Figura 6.4.

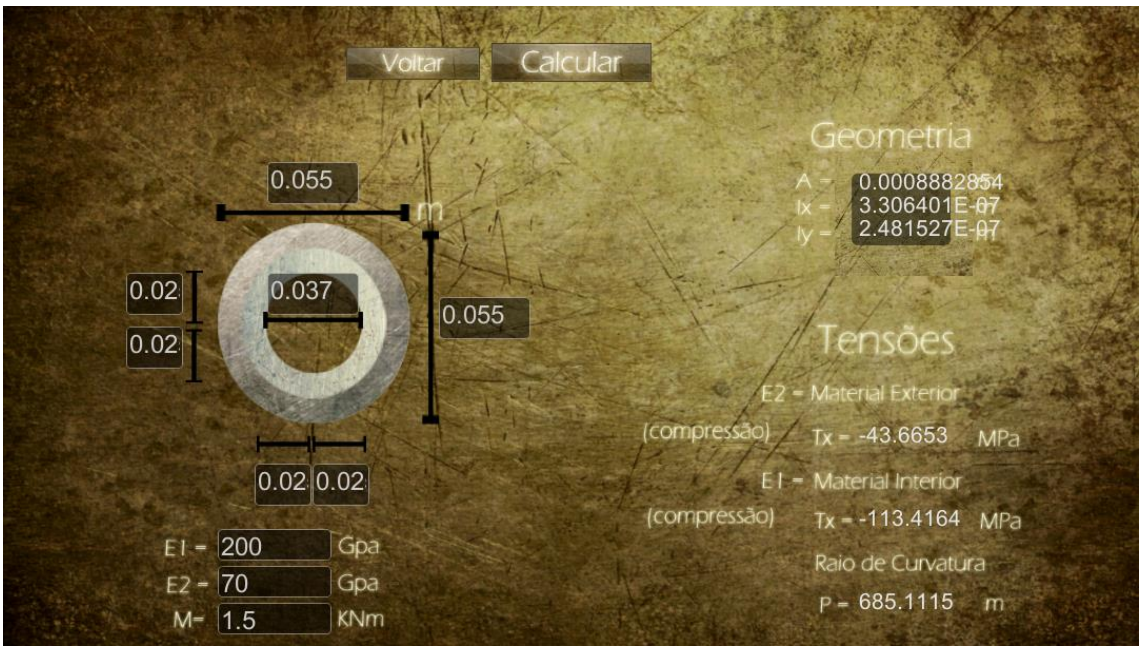


Fig.6.5. – Resultados obtidos numa secção circular composta por dois materiais sujeita a flexão simples

A inserção das medidas é executada em metros, e assim, obtêm-se os valores das tensões máximas para cada material. Deste modo, a tensão normal normal máxima no alumínio é de 43.67 MPa e no aço, a sua tensão máxima e de -113.42 MPa. Tal como seria esperado, a tensão no aço é superior à tensão no alumínio.

6.4. EXEMPLO 3 – DETERMINAÇÃO DAS TENSÕES NUMA SECÇÃO EM I

Neste exemplo demonstrado pela Figura 6.6., faz-se a comparação entre a variação de valores pela introdução de um vazio na alma da secção em I. A solicitação aplicada corresponde à acção simultânea de um esforço transverso vertical e de um momento fletor em torno do eixo x.

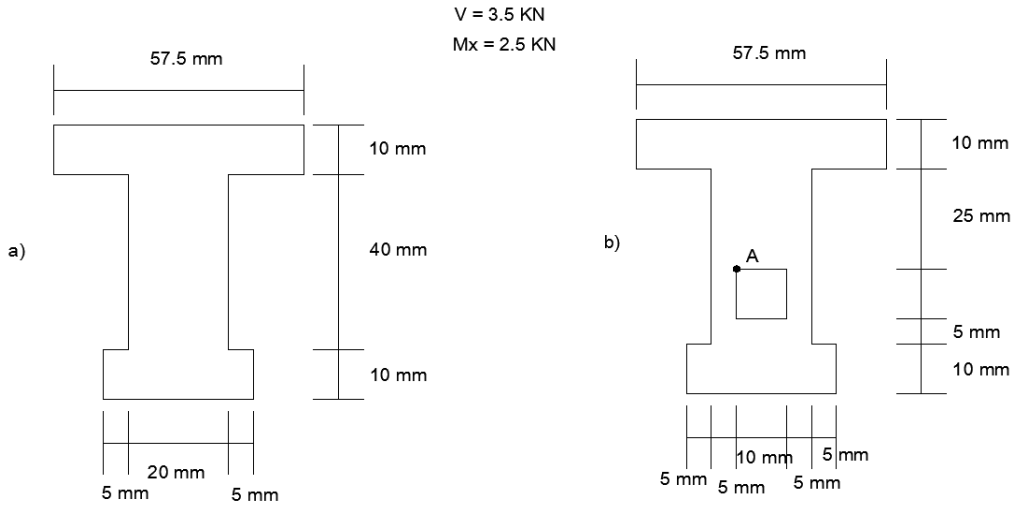


Fig.6.6. – Secção I sujeita a esforço transverso e momento fletor

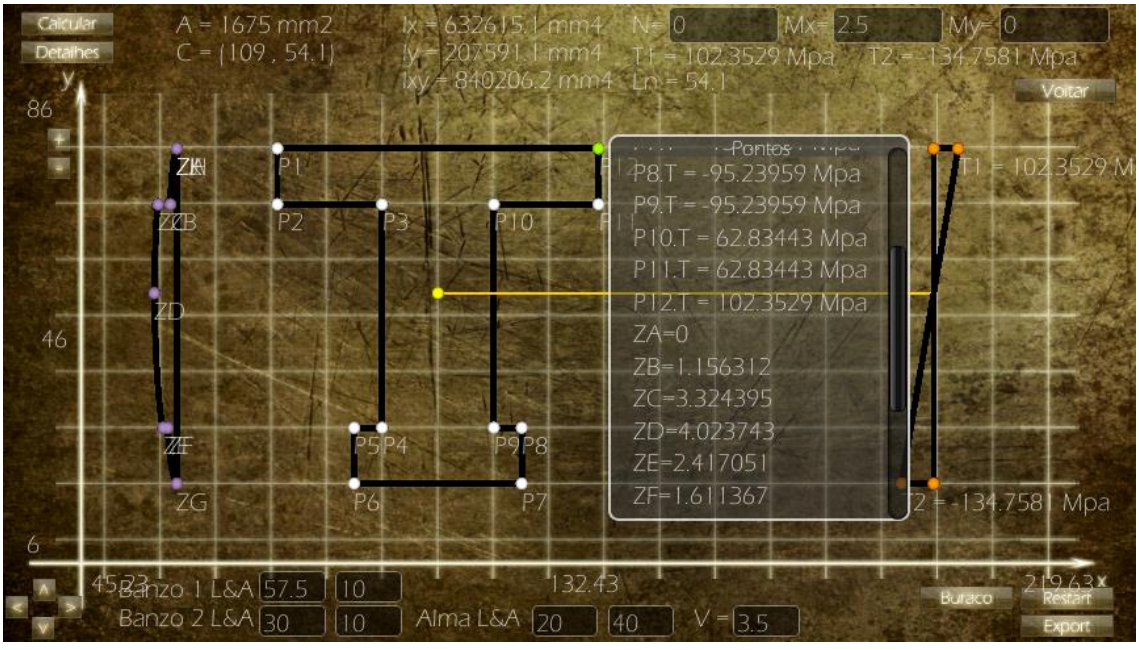


Fig.6.7. – Resultados obtidos numa secção em I sujeita a esforço transverso e momento fletor

Segundo a Figura 6.7, a tensão máxima de corte que se verifica na secção I sem abertura é de 4.03 MPa, a qual se localiza no eixo neutro das tensões normais. Da mesma forma, a tensão normal na fibra superior é de 102.35 MPa (tracção) e a tensão normal na fibra inferior é de -134.76 MPa (compressão). Seguindo para a hipótese b), insere-se o vazio na secção.

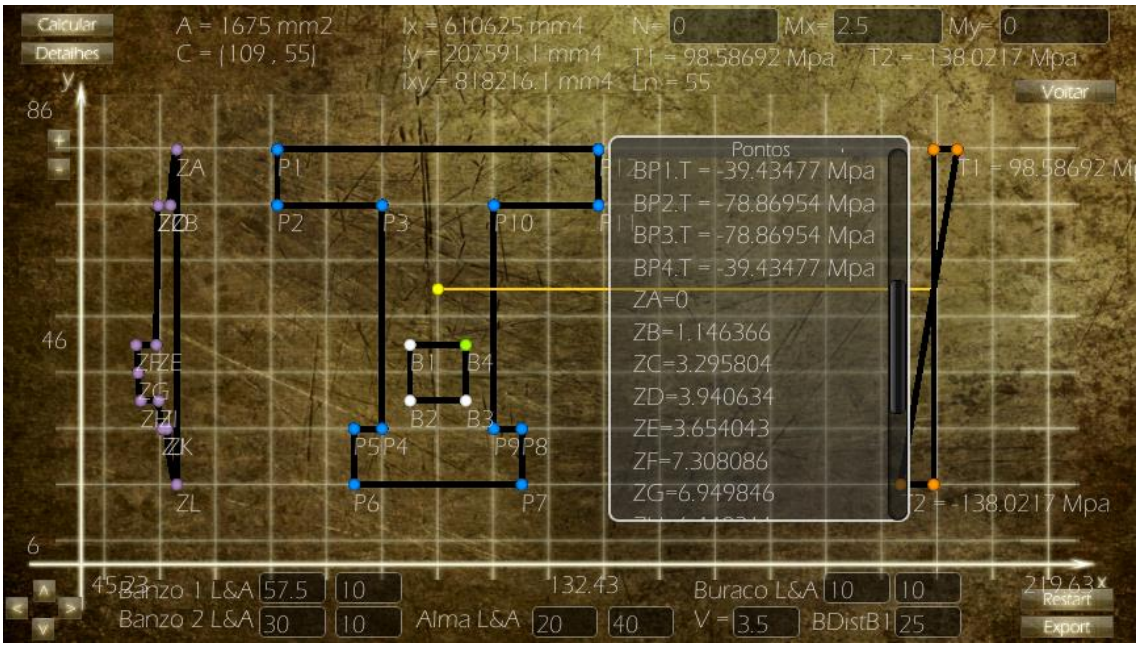


Fig.6.8. – Resultados obtidos numa Secção I com vazio sujeita a esforço transverso e momento fletor

A tensão de corte no ponto A (ver Figura 6.6.) tem o valor de 7.31MPa e coincide com a máxima tensão de corte na secção transversal. A tensão normal nesse ponto é de -39.43 MPa (compressão). A tensão da fibra superior é agora de 98.59 MPa (tração) decrescendo o seu valor em relação ao obtido na hipótese a), e a tensão na fibra inferior é de -138.02 Mpa (compressão), superior ao obtido na hipótese a). Evidencia-se assim, que a diminuição da secção na zona do vazio cria maiores tensões de corte confere o mencionado no Capítulo 4.

6.5. EXEMPLO 4 – DESENHO E CÁLCULO DO CÍRCULO DE *MOHR*

Estudando o estado plano de tensões apresentado pela Figura 6.9, pode-se desde logo identificar segundo a convenção de sinais que a tensão normal em x é positiva, a tensão normal em y é negativa e a tensão tangencial é positiva.

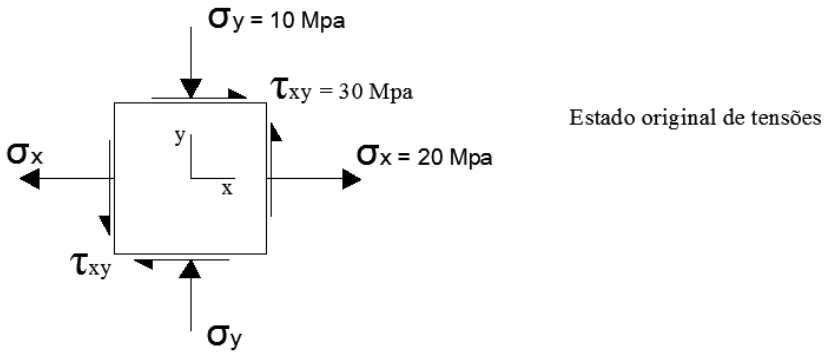


Fig.6.9. – Estado plano de tensões de um dado ponto num elemento

Com auxílio da ferramenta *Solve&Draw*, é possivel descobrir o plano das tensões principais e a tensão de corte máxima procedendo ao desenho do círculo de *Mohr*, como pode ser observado na Figura 6.10.

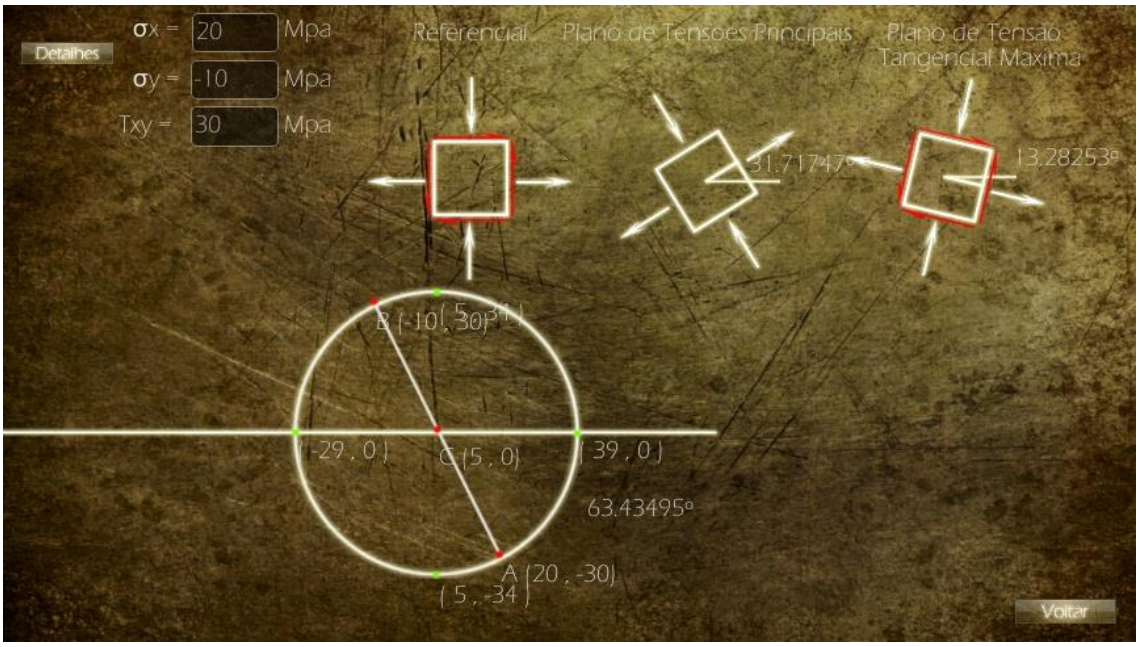


Fig.6.10. – Representação gráfica do círculo de *Mohr*

Ora então, o valor das tensões principais é respetivamente $\sigma_{x'}$ igual a 39 MPa e $\sigma_{y'}$ igual a -29 Mpa. No entanto, para se observar em mais detalhe pode-se utilizar a opção "Detalhes" no canto superior esquerdo.

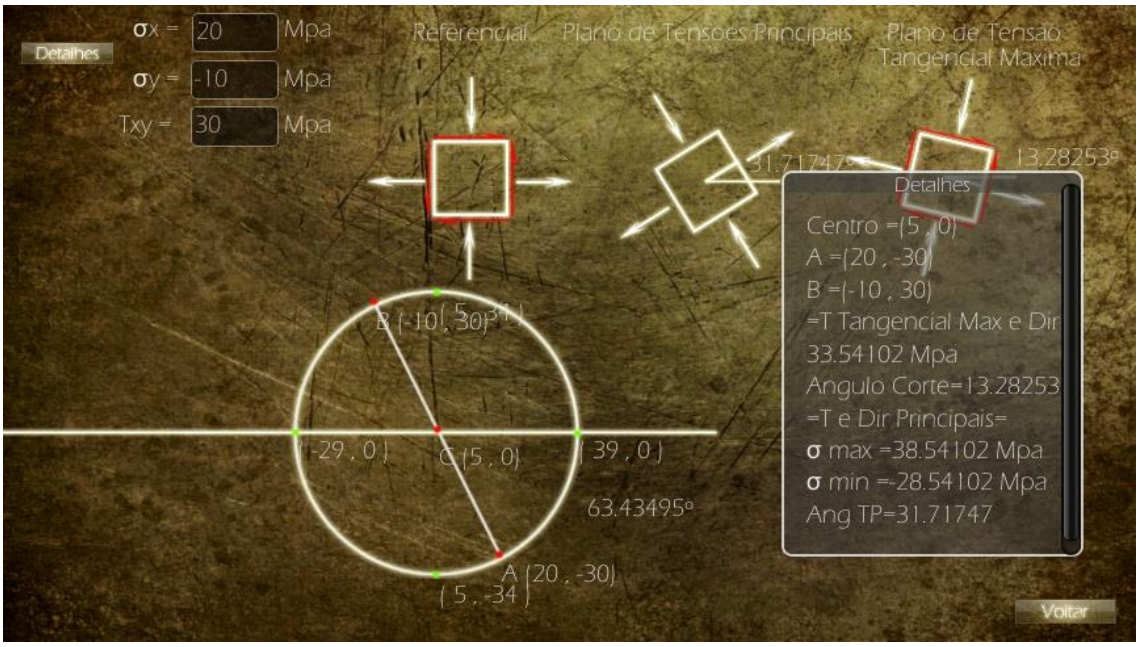


Fig.6.11. – Detalhes das tensões principais representadas no círculo de *Mohr*

O raio do círculo é 33.54 MPa, que corresponde ao valor da tensão tangencial máxima do ângulo dado pela reta entre os pontos A e B é de 63.4º e correspondente às direcções principais de tensão.

6.6. EXEMPLO 5 – CÁLCULO DE SECÇÕES SUBMETIDAS A FLEXÃO COMPOSTA DESVIADA

Na Figura 6.12., apresenta-se uma secção genérica sujeita a dois momentos.

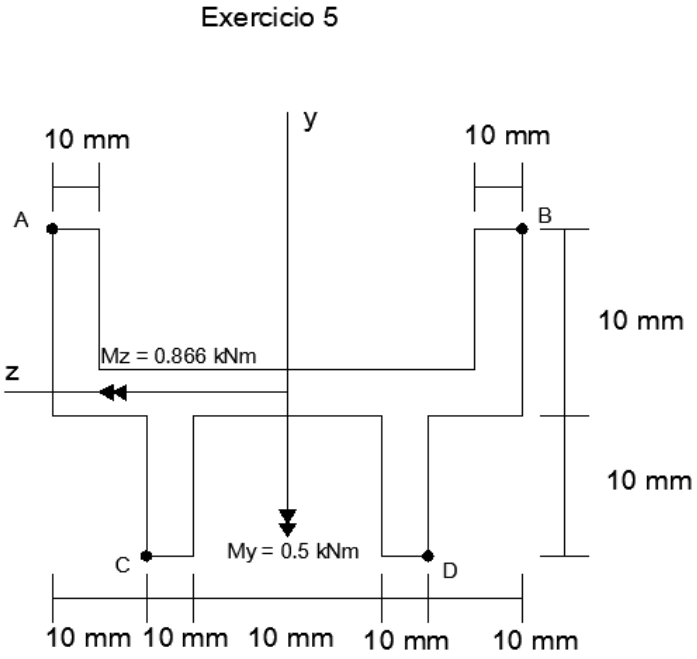


Fig.6.12. – Flexão composta desviada sobre uma secção genérica

Utilizando o módulo de desenho polinomial, é possivel obter a posição do centróide, a equação do eixo neutro, e as tensões nos pontos A, B,C e D.

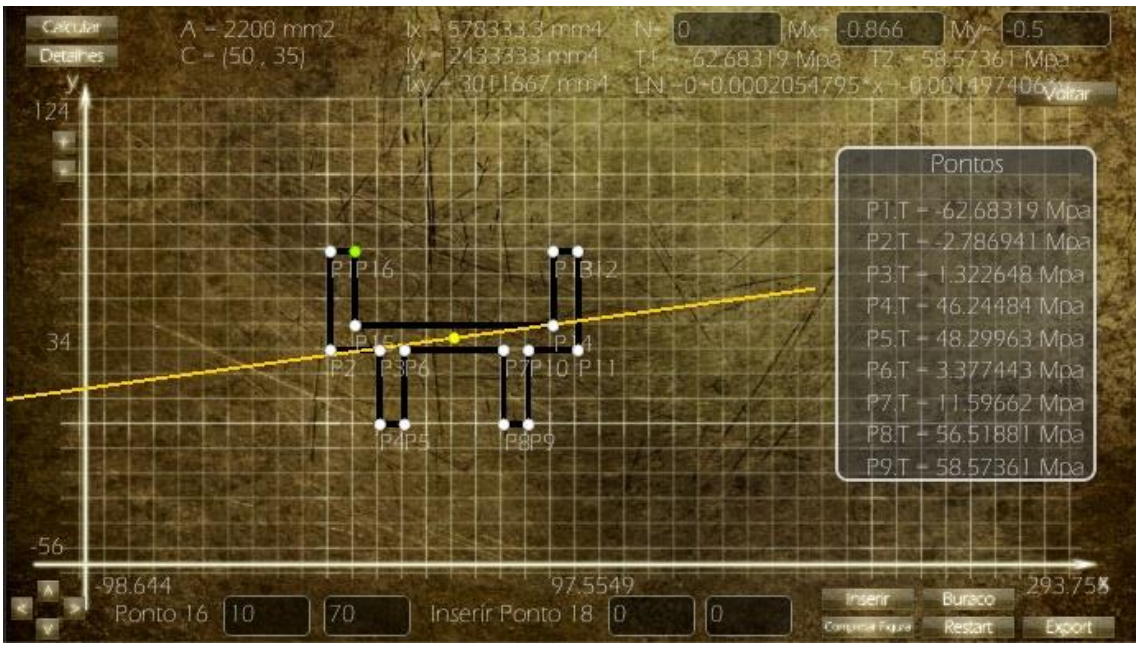


Fig.6.13 – Resultados obtidos numa secção genérica sujeita a flexão composta

Observando a Figura 6.13, retira-se a informação que o centróide está localizado em (50, 35) e que a equação do eixo neutra, é igual a 0.00020548x – 0.0014971y. As tensões de A, B, C e D, são respetivamente -62.7 Mpa, -42.1 Mpa, 46.2 Mpa e 58.6 Mpa.

Uma das grandes vantagens da utilização do módulo de desenho polinomial é a capacidade de alterar a secção muito facilmente e, graças a tal, testar diferentes opções para um determinado carregamento, tal como foi visto nas figuras 5.9 e 5.10.

7
CONSIDERAÇÕES FINAIS

Neste capítulo pretende-se a retirar uma ilação ou ilações do que foi realizado neste trabalho e apresentar algumas propostas para desenvolvimento futuro. O foco deste trabalho foi a criação de uma ferramenta didática, com o objetivo de auxiliar o estudo aos estudantes não só da Faculdade de Engenharia do Porto, mas para qualquer pessoa familiarizada com engenharia.

Pretendeu-se desenvolver aplicações simples que funcionassem de forma integrada, e modular possíveis de serem utilizadas por uma gama alargada de utilizadores. Nesse sentido, e devido à quantidade e variedade de equipamentos informáticos, atualmente disponiveis foi necessária uma programação direccionada para ser utilizada em múltiplos sistemas operativos. Apostou-se na utilização *online* através de um *website*, que devido à sua formatação permitisse a utilização da ferramenta através de uma vasta quantidade de motores de busca.

Em suma, o objetivo deste trabalho foi criar um programa que facilitasse a resolução de problemas e que demonstrasse uma grande compatibilidade com diversos equipamentos informáticos, sejam eles móveis ou não. Desta forma, a grande parte do conhecimento obtido encontra-se na detenção do próprio autor, ganho sobretudo de uma forma intuitiva em todo o processo de desenvolvimento, tanto ao nível de utilização de diferentes linguagens de programação, como no desenvolvimento de páginas usando *Html*.

Foram também adquiridas competencias na criação de domínios e servidores, na programação orientada para cada tipo de plataforma, na utilização dos *softwares* utilizados no processo criativo do programa. Estas competências permitiram resolver problemas na criação de métodos para ultrapassar diversos problemas, sendo eles relativos a um paradigma de programação, a um problema de estruturação de informação ou ao processo de criação de algoritmo.

Ao longo deste documento foram apresentadas as opções principais que o autor tomou de forma a introduzir as principais soluções a nível macro que foram adotadas e para resolver os problemas que surgiram durante a conceção da ferramenta. Desta forma houve uma pesquisa para determinar qual a melhor linguagem a utilizar, tendo a opção em primeira instância recaído sobre o *javascript*, de modo a criar a aplicação diretamente imbuída no *website*.

Em paralelo, criou-se uma aplicação para aparelhos móveis, compatíveis com o *website*. Rapidamente, foi concluido que o único método possível de ser exequível seria o de juntar os dois projetos em um, através de um único desenvolvimento que permitisse a criação de uma ferramenta para múltiplas plataformas. Neste processo foi utilizado a plataforma de desenvolvimento Unity3D, cuja capacidade de produção de programas compatíveis para diversas plataformas foi a principal razão para a sua escolha.

Como já referido acima, todas as escolhas que resolvem um problema geram por vezes outros problemas, neste caso, foi a compatibilidade do serviço de hospedagem com as permissões dos utilizadores. O servidor baseado no sistema operativo *Windows* não permitia a criação de

ficheiros no próprio servidor a partir de mecanismos automáticos, portanto foi necessária uma pesquisa mais elaborada no que diz respeito ao funcionamento desses serviços, tal como foi ilustrado no Capítulo 2. Assim sendo, para abrir as portas para a execução de determinadas ações foi alterado já durante a fase de desenvolvimento o tipo de sistema a operar o servidor, de *Windows* mudou-se para *Apache*, ou seja, um serviço baseado no sistema operativo *Linux*. O autor teve de desenvolver novos conhecimentos para lidar com este novo sistema e portanto, tal como foi descrito no início deste capítulo, o principal detentor de ganhos a nível de conhecimento trata-se do próprio autor deste documento.

Espera-se que de alguma forma, este documento transmita algum do conhecimento adquirido por parte do seu autor, visando promover o desenvolvimento de mais aplicações com ênfase na resolução de problemas de engenharia civil, ou até mesmo de outras áreas com orientação para os dispositivos móveis que já se usam com grande banalidade hoje em dia, ou ainda que sirva de inspiração para que os engenheiros civis se atrevam a navegar um pouco pelos paradigmas da programação e criem e desenvolvam novos métodos e novas técnicas para a evolução tecnológica na área da engenharia.

No que diz respeito à utilização da linguagem C Sharp, ela foi escolhida como linguagem a utilizar devido a sua grande versatilidade e compatibilidade com o Unity3D e também devido ao seu vasto suporte que pode ser facilmente consultado. A mesma acaba por ser uma linguagem simples e moderna, orientada para uma programação por objetos sendo uma das línguas mais utilizadas e amplamente divulgadas para a criação e desenvolvimento de *software* comercial. Em suma, trata-se de uma linguagem que providencia a simplicidade do visual *Basic* com o poder do seu precedente C++.

É pertinente ainda referir que para que um programa seja portável para várias plataformas é necessária a criação de uma *interface* que para além de ser orientada para um uso de multiplataformas, seja intuitiva e que transmita informação com pouco transtorno e ruído visual para o utilizador. Portanto, foi claramente evidenciado que para uma utilização mais confortável e prática para um utilizador desse tipo de aparelho informático, que as opções teriam que ser transmitidas de forma direta e intuitiva visualmente. As ações possíveis de se tomar também teriam um maior tamanho e estariam logo representadas na tela da ferramenta, da mesma forma, os resultados teriam que ser apresentados sem *popups*, ou seja, sem o auxílio a janelas extra, mas que infelizmente foi necessário a criação de uma janela desse género constituída pelo nome detalhes, mas que não é de forma alguma essencial na sua abertura devido às informações principais que estão representadas diretamente e que variam organicamente com o *input* do utilizador.

O programa com o nome *Solve&Draw* foi desenvolvido, com o auxílio de cerca de trinta *scripts* contendo entre eles mais do que sete mil linhas de código, possui, no entanto, a capacidade de ser expandido por meio de novos módulos completamente autónomos e sem alterar as linhas de código já escritas. A velocidade de resolução é alta, podendo por vezes transacionar um pouco mais lentamente entre os módulos devido à pré-existência de valores inseridos, que têm com o objetivo ajudar o utilizador a reconhecer mais facilmente as *input box*. A exportação dos valores é feita através de um bloco de notas e pode ser utilizada por outro programa noutro tipo de cálculo, uma grande mais-valia conseguida foi o seu desenvolvimento para vários sistemas operativos, tanto fixos como móveis, o que requereu um cuidado extra com a interface gráfica.

Desta forma e conforme o descrito ao longo do documento, cumprindo-se o objetivo de criar aplicações computacionais didáticas na área de estruturas para uma gama alargada de utilizadores, em várias plataformas informáticas, sendo que está ferramenta pode ainda ser expandida para resolver outro tipo de problemas pela introdução de novos módulos. O próximo

passo lógico seria a expansão da ferramenta para o cálculo e desenho de armaduras no desenho polinomial tanto como nos outros módulos descritos ao longo deste documento.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Maia, Raynerson Ferreira. *Desenvolvimento de uma aplicação web para apoio ao cálculo de estruturas metálicas.* Dissertação de Mestrado, Faculdade de Engenharia do Porto, 2012.

[2] Beer F.P., Johnston E. Russel, DeWolf J.T., *Mecânica dos Materiais*, McGraw-Hill de Portugal, Lisboa, 2003.

[3] William A. Nash, *Resistencia dos Materiais*, McGraw-Hill Ltda, São Paulo, 1990.

[4] Pytel A., Kiusalaas J., *Mechanics of Materials*, Brooks/Cole, a division of Thomson Learning Inc, Canada, 2003.

[5] Rodrigues P., Pereira P., Sousa M., *Programação em C++ conceitos básicos e Algoritmos*, FCA – Editora de Informática LDA, Lisboa, 2000.

[6] Gudwin R.R., *Linguagens de Programação*, Universidade Estadual de Campinas, Faculdade de Engenharia Eléctrica e de Computação, 1997.

[7] Rodrigues P., Pereira P., Sousa M., *Programação em C++ Algoritmos e Estruturas de Dados*, FCA – Editora de Informática LDA, Lisboa, 2000.

[8] Bindu R. Rao, *Object-Oriented Databases: Technology, Applications and Products*, McGraw-Hill, Inc., Singapore, 1994.

[9] Pereira A., Poupa C., *Linguagens Web*, Edições Sílabo Lda, Lisboa, 2005.

[10] Musciano C., Kennedy B., *HTML AND XHTMAL: The definitive Guide*, O'Reilly & Associates, INC, United States of America, 2000.

[11] Rischpater R., *Wireless Web Development*, Springer-Verlag, United States of America, 2000.

[12] Wille C., *Presenting C#*, Sams Publishing, United States of America, 2000.

[13] Tavares J., Castro J., *Princípios relacionados com a ergonomia de sistemas multimédia – uma sistematização possível.* Sítio da Internet (www.joanacastro.com/versao02/Principios.pdf). 20 de Junho 2015.

ANEXO A1

PARCELA DE CÓDIGO DO PROGRAMA

MÓDULO DE DESENHO POLINOMIAL

```
using UnityEngine;
using System.Collections;
using System.Text.RegularExpressions;
using System;
using System.IO;
public class CriadorDePoints : MonoBehaviour {
.
.
.
public void CallSpawnDiagram(){
float Nforcetemp = float.Parse (Nforce);
float Mxtemp = float.Parse (Mx);
float Mytemp = float.Parse (My);
GameObject PontoMaisADireita = FindPontoMaisADireita ();
GameObject PontoMaisAEsquerda = FindPontoMaisAEsquerda ();
GameObject PontoMaisABaixo = FindPontoMaisABaixo ();
GameObject PontoMaisACima = FindPontoMaisACima  ();
if (Mxtemp != 0 && Mytemp != 0) {
PontoMaisACima = FindPontoMaisACimaDiagonal  ();
PontoMaisABaixo = FindPontoMaisABaixoDiagonal  ();
PontoMaisADireita = FindPontoMaisABaixoDiagonal ();
PontoMaisAEsquerda = FindPontoMaisACimaDiagonal ();                }
float DiagramT1X = PontoMaisACima.GetComponent<PontoInteractions>().virtualx - centroidcoordX;
float DiagramT1Y = PontoMaisACima.GetComponent<PontoInteractions>().virtualy - centroidcoordY;
float DiagramT2X = - (centroidcoordX - PontoMaisABaixo.GetComponent<PontoInteractions>().virtualx) ;
float DiagramT2Y = - (centroidcoordY - PontoMaisABaixo.GetComponent<PontoInteractions>().virtualy) ;
if (Mxtemp == 0 && Mytemp != 0) {
DiagramT1X = PontoMaisADireita.GetComponent<PontoInteractions>().virtualx - centroidcoordX;
DiagramT1Y = PontoMaisADireita.GetComponent<PontoInteractions>().virtualy - centroidcoordY;
```

```
DiagramT2X = - (centroidcoordX - PontoMaisAEsquerda.GetComponent<PontoInteractions>().virtualx) ;

DiagramT2Y = - (centroidcoordY - PontoMaisAEsquerda.GetComponent<PontoInteractions>().virtualy) ;

}

if (Mxtemp != 0 && Mytemp != 0) {

if(PontoMaisABaixo.GetComponent<PontoInteractions>().virtualx == PontoMaisACima.GetComponent<PontoInteractions>().virtualx && PontoMaisABaixo.GetComponent<PontoInteractions>().virtualy == PontoMaisACima.GetComponent<PontoInteractions>().virtualy){

DiagramT1X = PontoMaisADireita.GetComponent<PontoInteractions>().virtualx - centroidcoordX;

DiagramT1Y = PontoMaisADireita.GetComponent<PontoInteractions>().virtualy - centroidcoordY;

DiagramT2X = - (centroidcoordX - PontoMaisAEsquerda.GetComponent<PontoInteractions>().virtualx) ;

DiagramT2Y = - (centroidcoordY - PontoMaisAEsquerda.GetComponent<PontoInteractions>().virtualy) ;

}

DiagramT2X = PontoMaisABaixo.GetComponent<PontoInteractions>().virtualx - centroidcoordX;

DiagramT2Y = PontoMaisABaixo.GetComponent<PontoInteractions>().virtualy - centroidcoordY;

DiagramT1X = - (centroidcoordX - PontoMaisACima.GetComponent<PontoInteractions>().virtualx) ;

DiagramT1Y = - (centroidcoordY - PontoMaisACima.GetComponent<PontoInteractions>().virtualy) ;

if(T1 == T2){

DiagramT1X = centroidcoordX - PontoMaisACima.GetComponent<PontoInteractions>().virtualx ;

DiagramT1Y = centroidcoordY - PontoMaisACima.GetComponent<PontoInteractions>().virtualy ;

DiagramT2X = centroidcoordX - PontoMaisABaixo.GetComponent<PontoInteractions>().virtualx ;

DiagramT2Y = centroidcoordY - PontoMaisABaixo.GetComponent<PontoInteractions>().virtualy ;

}

}

CalcMomentx();

Inertia= momentx;
```

```
InertiaY= momenty;

T1 = (Nforcetemp) / (areadashape-areadashape2) - ((Mytemp*1000) / (InertiaY-InertiaY2)) * DiagramT1X +  ((Mxtemp*1000) / (Inertia-Inertia2)) * DiagramT1Y ;

T2 = (Nforcetemp) / (areadashape-areadashape2) - ((Mytemp*1000) / (InertiaY-InertiaY2)) * DiagramT2X +  ((Mxtemp*1000) / (Inertia-Inertia2)) * DiagramT2Y ;

T1 *= 1000;

T2 *= 1000;

////////DIAGRAMA HORIZONTAL

if (Mxtemp !=0 && Mytemp==0){

DiagramHeight = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy - PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualy;

float T1temp = T1;

float T2temp = T2;

T1Length = (T1temp * (DiagramHeight / 2)) / DiagramTMaxLength;

T2Length = (T2temp * (DiagramHeight / 2)) / DiagramTMaxLength;

if (T1temp > DiagramTMaxLength) {T1Length=DiagramHeight / 2;}

if (T1temp < -DiagramTMaxLength) {T1Length= - DiagramHeight / 2;}

if (T2temp > DiagramTMaxLength) {T2Length=DiagramHeight / 2;}

if (T2temp < -DiagramTMaxLength) {T2Length= - DiagramHeight / 2;}

TRatio = T2Length / T1Length;

TRatioAplicadoAHeight = DiagramHeight * TRatio;

DistLinhaNeutraAoTopPoint= -T1Length / (  (T2Length-T1Length)/DiagramHeight  ) ;

DiagramPoint2X = PontoMaisADireita.GetComponent<PontoInteractions> ().virtualx + DiagramHeight;

DiagramPoint2Y = PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualy;

DiagramPoint1X = DiagramPoint2X + T2Length;

DiagramPoint1Y = PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualy;

DiagramPoint3X = PontoMaisADireita.GetComponent<PontoInteractions> ().virtualx + DiagramHeight;

DiagramPoint3Y = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy;

DiagramPoint4X = DiagramPoint2X + T1Length;

DiagramPoint4Y = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy;

DiagramPoint5X = DiagramPoint1X;

DiagramPoint5Y = DiagramPoint1Y;

LinhaNeutraPoint1X = centroidcoordX;

LinhaNeutraPoint1Y = DiagramPoint3Y- DistLinhaNeutraAoTopPoint;
```

```
LinhaNeutraPoint2X = LinhaNeutraPoint1X;
LinhaNeutraPoint2Y = LinhaNeutraPoint1Y;
LinhaNeutraPoint3X = DiagramPoint3X;
LinhaNeutraPoint3Y = DiagramPoint3Y- DistLinhaNeutraAoTopPoint;
LinhaNeutraPoint4X = LinhaNeutraPoint3X;
LinhaNeutraPoint4Y = LinhaNeutraPoint3Y;
if ((T1temp > 0 && T2temp > 0) || (T1temp < 0 && T2temp < 0)) {
NaoHaLinhaNeutra=true;
}else{
NaoHaLinhaNeutra=false;
}
}
/////////DIAGRAMA VERTICAL
if (Mxtemp ==0 && Mytemp!=0){
DiagramHeight = PontoMaisADireita.GetComponent<PontoInteractions> ().virtualx - PontoMaisAEsquerda.GetComponent<PontoInteractions> ().virtualx;
float T1temp = T1;
float T2temp = T2;
T1Length = (T1temp * (DiagramHeight / 2)) / DiagramTMaxLength;
T2Length = (T2temp * (DiagramHeight / 2)) / DiagramTMaxLength;
if (T1temp > DiagramTMaxLength) {T1Length=DiagramHeight / 2;}
if (T1temp < -DiagramTMaxLength) {T1Length= - DiagramHeight / 2;}
if (T2temp > DiagramTMaxLength) {T2Length=DiagramHeight / 2;}
if (T2temp < -DiagramTMaxLength) {T2Length= - DiagramHeight / 2;}
TRatio = T2Length / T1Length;
TRatioAplicadoAHeight = DiagramHeight * TRatio;
DistLinhaNeutraAoTopPoint= -T1Length / (  (T2Length-T1Length)/DiagramHeight  ) ;
DiagramPoint2X = PontoMaisAEsquerda.GetComponent<PontoInteractions> ().virtualx;
DiagramPoint2Y = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy + DiagramHeight/2;
DiagramPoint1X = PontoMaisAEsquerda.GetComponent<PontoInteractions> ().virtualx;
DiagramPoint1Y = DiagramPoint2Y + T2Length;
DiagramPoint3X = PontoMaisADireita.GetComponent<PontoInteractions> ().virtualx;
DiagramPoint3Y = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy + DiagramHeight/2;
DiagramPoint4X = PontoMaisADireita.GetComponent<PontoInteractions> ().virtualx;
```

```
DiagramPoint4Y = DiagramPoint2Y + T1Length;
DiagramPoint5X = DiagramPoint1X;
DiagramPoint5Y = DiagramPoint1Y;
LinhaNeutraPoint1X = DiagramPoint3X- DistLinhaNeutraAoTopPoint;
LinhaNeutraPoint1Y = centroidcoordY;
LinhaNeutraPoint2X = LinhaNeutraPoint1X;
LinhaNeutraPoint2Y = LinhaNeutraPoint1Y;
LinhaNeutraPoint3X = DiagramPoint3X- DistLinhaNeutraAoTopPoint;
LinhaNeutraPoint3Y = DiagramPoint3Y;
LinhaNeutraPoint4X = LinhaNeutraPoint3X;
LinhaNeutraPoint4Y = LinhaNeutraPoint3Y;
if ((T1temp > 0 && T2temp > 0) || (T1temp < 0 && T2temp < 0)) {
NaoHaLinhaNeutra=true;
}else{
NaoHaLinhaNeutra=false; }}
////////// DIAGRAMA OBLIQUO
if (Mxtemp !=0 && Mytemp!=0){
DiagramHeight = PontoMaisACima.GetComponent<PontoInteractions>().virtuald + PontoMaisABaixo.GetComponent<PontoInteractions>().virtuald;
Debug.Log ("diagramhueight=" +DiagramHeight);
float T1temp = T1;
float T2temp = T2;
T1Length = (T1temp * (DiagramHeight / 2)) / DiagramTMaxLength;
T2Length = (T2temp * (DiagramHeight / 2)) / DiagramTMaxLength;
if (T1temp > DiagramTMaxLength) {T1Length=DiagramHeight / 2;}
if (T1temp < -DiagramTMaxLength) {T1Length= - DiagramHeight / 2;}
if (T2temp > DiagramTMaxLength) {T2Length=DiagramHeight / 2;}
if (T2temp < -DiagramTMaxLength) {T2Length= - DiagramHeight / 2;}
TRatio = T2Length / T1Length;
TRatioAplicadoAHeight = DiagramHeight * TRatio;
DistLinhaNeutraAoTopPoint= -T1Length / (  (T2Length-T1Length)/DiagramHeight  ) ;
float LNdPoint1X=  (  (  (  (Nforcetemp)/(areadashape-areadashape2)  ) + ((Mxtemp*1000) / (Inertia-Inertia2))  *  20  )/  ((Mytemp*1000) / (InertiaY-InertiaY2))  );
LinhaNeutraPoint1X = centroidcoordX + LNdPoint1X ;
LinhaNeutraPoint1Y = centroidcoordY + 20;
```

```
float LNdPoint2X= ( ( ( (Nforcetemp)/(areadashape-areadashape2) ) + ((Mxtemp*1000) / (Inertia-Inertia2)) * (-20) )/ ((Mytemp*1000) / (InertiaY-InertiaY2)) );

LinhaNeutraPoint2X = centroidcoordX + LNdPoint2X ;

LinhaNeutraPoint2Y = centroidcoordY - 20;

DecliveDaLinhaNeutra=(LinhaNeutraPoint1Y-LinhaNeutraPoint2Y) / (LinhaNeutraPoint1X-LinhaNeutraPoint2X);

LinhaNeutraPoint3X = LinhaNeutraPoint2X - (DiagramHeight/2)/DecliveDaLinhaNeutra;

LinhaNeutraPoint3Y = LinhaNeutraPoint2Y - (DiagramHeight/2);

LinhaNeutraPoint4X = LinhaNeutraPoint3X;

LinhaNeutraPoint4Y = LinhaNeutraPoint3Y;

DiagramPoint2X = PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualx - (DiagramHeight/2)/DecliveDaLinhaNeutra;

DiagramPoint2Y = PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualy - (DiagramHeight/2);

DiagramPoint1X = DiagramPoint2X - T1Length / Mathf.Sqrt( 1 + Mathf.Pow(DecliveDaLinhaNeutra,2) );

float cucu =  Mathf.Abs( DiagramPoint1X - DiagramPoint2X );

float coco = cucu*DecliveDaLinhaNeutra;

if (T1Length<0){ DiagramPoint1Y = DiagramPoint2Y + coco;}

if (T1Length>0){ DiagramPoint1Y = DiagramPoint2Y - coco;}

float d;

float AX = PontoMaisACima.GetComponent<PontoInteractions> ().virtualx;

float AY = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy;

float BX = PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualx;

float BY = PontoMaisABaixo.GetComponent<PontoInteractions> ().virtualy;

float CX = DiagramPoint2X;

float CY = DiagramPoint2Y;

float VX;

float VY;

float F;

float dist;

d =  Mathf.Pow((AX - BX),2) + Mathf.Pow((AY - BY),2) ;

VX =  BX - CX;

VY = BY - CY;

F =  ((AX-BX)*VX + (AY-BY)*VY ) / Mathf.Sqrt( Mathf.Pow(VX,2)+Mathf.Pow(VY,2) );

dist =  Mathf.Sqrt ( d+ Mathf.Pow(F,2) ) ;
```

```
float z = DiagramPoint2Y+  DiagramPoint2X/DecliveDaLinhaNeutra;

float g = PontoMaisACima.GetComponent<PontoInteractions> ().virtualy - PontoMaisACima.GetComponent<PontoInteractions> ().virtualx*DecliveDaLinhaNeutra;

DiagramPoint3X = (z-g)/ (  ( Mathf.Pow(DecliveDaLinhaNeutra,2)+1) / DecliveDaLinhaNeutra ) ;

DiagramPoint3Y =  DecliveDaLinhaNeutra* DiagramPoint3X  + g;

DiagramPoint4X = DiagramPoint3X - T2Length / Mathf.Sqrt( 1 + Mathf.Pow(DecliveDaLinhaNeutra,2) );

float ca=  Mathf.Abs( DiagramPoint4X - DiagramPoint3X );

float cece = ca*DecliveDaLinhaNeutra;

if ( T2Length>0){ DiagramPoint4Y = DiagramPoint3Y - cece;}

if ( T2Length<0){ DiagramPoint4Y = DiagramPoint3Y + cece;}

DiagramPoint5X = DiagramPoint1X;

DiagramPoint5Y = DiagramPoint1Y;

///////LINHANEUTRAFIX

//LinhaNeutraPoint1X=centroidcoordX;

//LinhaNeutraPoint1Y=centroidcoordY;

if ((T1temp > 0 && T2temp > 0) || (T1temp < 0 && T2temp < 0)) {

NaoHaLinhaNeutra=true;

}else{

NaoHaLinhaNeutra=false; } }

if (Mxtemp != 0 || Mytemp!=0) {

if (!DiagramSpawned) {

drawerlinhaneutradiagramdecoy.SetActive(true);

drawerlinhaneutradiagramdecoy.GetComponent<LineRenderer>().enabled=true;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>().CallpaSpawnLinhaNeutra Points=true;

drawerdiagramdecoy.SetActive(true);

drawerdiagramdecoy.GetComponent<CriadorDePoints>().CallpaSpawnDiagram=true;  }

if (DiagramSpawned) {

drawerlinhaneutradiagramdecoy.SetActive(true);

drawerlinhaneutradiagramdecoy.GetComponent<LineRenderer>().enabled=true;

drawerdiagramdecoy.SetActive(true);

drawerdiagramdecoy.GetComponent<CriadorDePoints>().CallPaRevealAllPoints=true;

drawerdiagramdecoy.GetComponent<CriadorDePoints> ().pontos [0].GetComponent<PontoInteractions> ().virtualx = DiagramPoint1X;
```

```
drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[0].GetComponent<PontoInteractions> ().virtualy = DiagramPoint1Y;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[1].GetComponent<PontoInteractions> ().virtualx = DiagramPoint2X;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[1].GetComponent<PontoInteractions> ().virtualy = DiagramPoint2Y;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[2].GetComponent<PontoInteractions> ().virtualx = DiagramPoint3X;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[2].GetComponent<PontoInteractions> ().virtualy = DiagramPoint3Y;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[3].GetComponent<PontoInteractions> ().virtualx = DiagramPoint4X;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[3].GetComponent<PontoInteractions> ().virtualy = DiagramPoint4Y;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[4].GetComponent<PontoInteractions> ().virtualx = DiagramPoint5X;

drawerdiagramdecoy.GetComponent<CriadorDePoints>                    ().pontos
[4].GetComponent<PontoInteractions> ().virtualy = DiagramPoint5Y;

drawerdiagramdecoy.GetComponent<CriadorDePoints> ().scaling = true;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[0].GetComponent<PontoInteractions> ().virtualx = LinhaNeutraPoint1X;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[0].GetComponent<PontoInteractions> ().virtualy = LinhaNeutraPoint1Y;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[1].GetComponent<PontoInteractions> ().virtualx = LinhaNeutraPoint2X;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[1].GetComponent<PontoInteractions> ().virtualy = LinhaNeutraPoint2Y;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[2].GetComponent<PontoInteractions> ().virtualx = LinhaNeutraPoint3X;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[2].GetComponent<PontoInteractions> ().virtualy = LinhaNeutraPoint3Y;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[3].GetComponent<PontoInteractions> ().virtualx = LinhaNeutraPoint4X;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints>          ().pontos
[3].GetComponent<PontoInteractions> ().virtualy = LinhaNeutraPoint4Y;

drawerlinhaneutradiagramdecoy.GetComponent<CriadorDePoints> ().scaling = true;
                                                                     }

DiagramSpawned = true; }

if (Mxtemp==0 && Mytemp==0){

drawerlinhaneutradiagramdecoy.GetComponent<LineRenderer>().enabled=false;
```

```
//drawerlinhaneutradiagramdecoy.SetActive(false);
drawerdiagramdecoy.GetComponent<CriadorDePoints>().CallPaHideAllPoints=true;}
if (NaoHaLinhaNeutra) {
drawerlinhaneutradiagramdecoy.GetComponent<LineRenderer>().enabled=false;
/drawerlinhaneutradiagramdecoy.SetActive(false);}}
public void SpawnLinhaNeutraPoints(){
SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);
public void SpawnDiagram(){SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);
SpawnDiagramPointOnCoords (0, 0);}
public void SpawnDiagramPointOnCoords(float FeedCordX, float FeedCordY){
float pontox = FeedCordX;
pontox = pontox/(xscalemax/15.268818f);
pontox +=-7.634409f;
float pontoy = FeedCordY;
pontoy = pontoy/(yscalemax/6.989248f);
pontoy +=-3.74552f;
Vector3 v2 = new Vector3 (pontox, pontoy, 0);
pontos [arraysizepoobj] = Instantiate (designdoponto, v2, Quaternion.identity) as GameObject;
PontoInteractions temppts = pontos [arraysizepoobj].GetComponent<PontoInteractions>();
temppts.insertedpoint=true;
temppts.simsoudiagrampoint=true;
if (linhaneutradiagramdecoy) {
pontos [arraysizepoobj].GetComponent<MeshRenderer>().enabled=false;}
highlightedponto = pontos [arraysizepoobj];
temppts.virtualx = FeedCordX;
temppts.virtualy = FeedCordY;
if (arraysize == 0) {
lineRenderer.SetVertexCount (1);
lineRenderer.SetPosition (arraysize, pontos [arraysizepoobj].transform.position);
```

```
lastposition = pontos [arraysizepoobj].transform.position;
arraysize += 1;
arraysizepoobj += 1;
} else {
lineRenderer.SetVertexCount (arraysize + 3);
lineRenderer.SetPosition (arraysize, pontos [arraysizepoobj].transform.position);
lineRenderer.SetPosition (arraysize + 1, lastposition);
lineRenderer.SetPosition (arraysize + 2, pontos [arraysizepoobj].transform.position);
lastposition = pontos [arraysizepoobj].transform.position;
arraysize += 3;
arraysizepoobj += 1;}}
public GameObject FindPontoMaisADireita(){
GameObject PontmaisDir = pontos [0];
for (int i= 0; i < arraysizepoobj; i++) {
if( pontos[i].GetComponent<PontoInteractions>().virtualx > PontmaisDir.GetComponent<PontoInteractions>().virtualx ){
PontmaisDir=pontos[i];} }
return PontmaisDir;}
public GameObject FindPontoMaisAEsquerda(){
GameObject PontmaisEsq = pontos [0];
for (int i= 0; i < arraysizepoobj; i++) {
if( pontos[i].GetComponent<PontoInteractions>().virtualx < PontmaisEsq.GetComponent<PontoInteractions>().virtualx ){
PontmaisEsq=pontos[i];} }
return PontmaisEsq;}
public GameObject FindPontoMaisABaixo(){
GameObject PontmaisBai = pontos [0];
for (int i= 0; i < arraysizepoobj; i++) {
if( pontos[i].GetComponent<PontoInteractions>().virtualy < PontmaisBai.GetComponent<PontoInteractions>().virtualy ){
PontmaisBai=pontos[i]; } }
return PontmaisBai;}
public GameObject FindPontoMaisACima(){
GameObject PontmaisCim = pontos [0];
for (int i= 0; i < arraysizepoobj; i++) {
```

```
if( pontos[i].GetComponent<PontoInteractions>().virtualy > PontmaisCim.GetComponent<PontoInteractions>().virtualy ){

PontmaisCim=pontos[i];}}

return PontmaisCim;} public GameObject FindPontoMaisACimaDiagonal(){

GameObject PontmaisCimaDia = pontos [0];

for (int i= 0; i < arraysizepoobj; i++) {

float b;

float TlnX= pontos[i].GetComponent<PontoInteractions>().virtualx - centroidcoordX;

float TlnY= pontos[i].GetComponent<PontoInteractions>().virtualy - centroidcoordY;

b = TlnY + TlnX * DecliveDaLinhaNeutra;

float TlnX2= b / (2*DecliveDaLinhaNeutra);

float TlnY2= b / 2;

float d = Mathf.Sqrt(  (Mathf.Pow((TlnX2-TlnX),2)) +  (Mathf.Pow((TlnY2-TlnY),2))  );

pontos [i].GetComponent<PontoInteractions>().ADireitaDaLN = isRight( LinhaNeutraPoint1X, LinhaNeutraPoint1Y, LinhaNeutraPoint2X, LinhaNeutraPoint2Y, pontos[i].GetComponent<PontoInteractions>().virtualx, pontos[i].GetComponent<PontoInteractions>().virtualy);

//float tempi=i+1;

//Debug.Log("pontos " + tempi + " - d= " + d + " , Declive= "+ DecliveDoPonto );

pontos [i].GetComponent<PontoInteractions>().virtuald= d;

if( pontos[i].GetComponent<PontoInteractions>().ADireitaDaLN==false && pontos[i].GetComponent<PontoInteractions>().virtuald > PontmaisCimaDia.GetComponent<PontoInteractions>().virtuald ){

PontmaisCimaDia=pontos[i];}}

return PontmaisCimaDia;}

public GameObject FindPontoMaisABaixoDiagonal(){

GameObject PontmaisBai = pontos [0];

for (int i= 0; i < arraysizepoobj; i++) {

float b;

float TlnX= pontos[i].GetComponent<PontoInteractions>().virtualx - centroidcoordX;

float TlnY= pontos[i].GetComponent<PontoInteractions>().virtualy - centroidcoordY;

b = TlnY + TlnX * DecliveDaLinhaNeutra;

float TlnX2= b / (2*DecliveDaLinhaNeutra);

float TlnY2= b / 2;


float d = Mathf.Sqrt(  (Mathf.Pow((TlnX2-TlnX),2)) +  (Mathf.Pow((TlnY2-TlnY),2))  );
```

```
pontos [i].GetComponent<PontoInteractions>().ADireitaDaLN = isRight( LinhaNeutraPoint1X, LinhaNeutraPoint1Y, LinhaNeutraPoint2X, LinhaNeutraPoint2Y, pontos[i].GetComponent<PontoInteractions>().virtualx, pontos[i].GetComponent<PontoInteractions>().virtualy);

//float tempi=i+1;

//Debug.Log("declive da ln= " + DecliveDaLinhaNeutra);

//Debug.Log("pontos " + tempi + " - d= " + d + " , a direita?= "+ pontos [i].GetComponent<PontoInteractions>().ADireitaDaLN );

pontos [i].GetComponent<PontoInteractions>().virtuald= d;

if( pontos[i].GetComponent<PontoInteractions>().ADireitaDaLN==true && pontos[i].GetComponent<PontoInteractions>().virtuald > PontmaisBai.GetComponent<PontoInteractions>().virtuald ){

PontmaisBai=pontos[i];

                                        }

                                    }

return PontmaisBai;


                                }
```