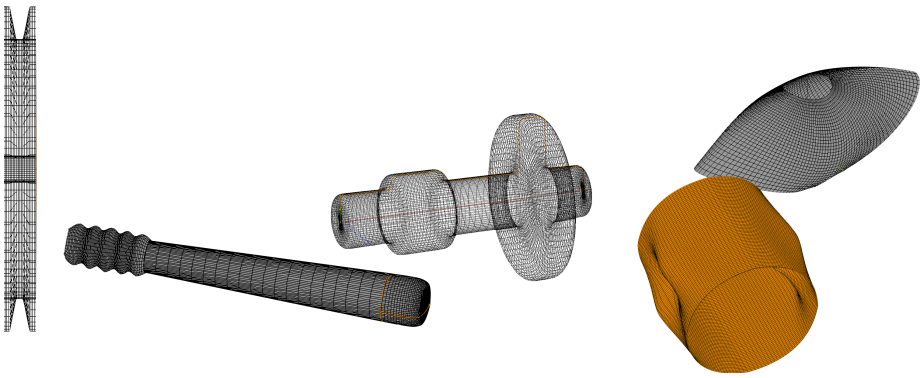
Universidade de Aveiro Departamento de Engenharia Mecânica
2012

Alexandre Seixas
Pinho Peralta

Análise isogeométrica de estruturas do tipo placa e casca



Alexandre Seixas Pinho Peralta

Análise isogeométrica de estruturas do tipo placa e casca

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação cientifica de Rui Pedro Ramos Cardoso, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President	**Prof. Doutor Francisco José Malheiro Queirós de Melo** Professor Associado da Universidade de Aveiro
Vogais / Committee	**Prof. Doutor Rui Pedro Ramos Cardoso** Professor Auxiliar da Universidade de Aveiro (orientador)
	Prof. Doutor José Manuel de Ameida Cesar de Sá Professor Catedrático do Dep. Eng. Mecânica da Faculdade de Engenharia da Universidade do Porto

Agradecimentos / Acknowledgements

Dedico este trabalho a todos os que estão de alguma forma presentes para o aplaudir. À minha família como forma de retribuir o esforço feito para me proporcionar esta formação académica. À Mafalda Nunes Branco pelo apoio e motivação incondicional durante o decorrer deste trabalho. Agradeço em especial ao Professor Rui Cardoso pela oportunidade que me foi dada para trabalhar sob a sua orientação.

Palavras-chave

Non-Uniform Rational B-Splines (NURBS); Análise Isogeométrica; Estruturas Placa e Casca; Modelação Sólida de Objectos NURBS; Programação Orientada por Objectos

Resumo

A análise isogeométrica vem optimizar um modelo de cálculo já utilizado: o Método dos Elementos Finitos. Com as devidas modificações, é possível obter bons resultados dispendendo menos tempo em todo o processo, bem como obter taxas de convergência mais elevadas. Para ser possível realizar essa análise é necessário que o código tradicional do Método dos Elementos Finitos sofra as alterações convenientes. Paralelamente deve ser feita a preparação dos modelos para submeter a posterior análise, tendo sido implementadas as devidas funcionalidades num *software* já existente que suporta principalmente investigação no campo dos métodos numéricos. Para tal foram implementas ferramentas que permitem a modelação de geometrias baseadas em NURBS, e a exportação dos dados das mesmas. Esta ferramenta de pré e pós-processamento está disponível para *Machintosh* dada a versatilidade da linguagem de programação orientada a objectos em que está escrito. Essa linguagem é denominada *Objective-C* e também esta matéria será abordada no presente trabalho ainda que de forma sucinta.

Keywords Non-Uniform Rational B-Splines (NURBS); Isogeometric Analysis; Thin Shell Structures; NURBS modeling; Object Oriented Programming

Abstract Isogeometric analysis came up to optimize a classical method, the Finite Element Method. With the proper modifications, it's possible get good results spending less time on entire process, as well as getting great convergence rates. To be able to perform such analysis is necessary that traditional Finite Element code undergoes some changes. Simultaneously must be made the preprocesing work on a software previously created for investigation purposes on numerical methods area. To accomplish that goal, NURBS modeling tools were developed to graphicaly create those geometries, and be able to export the necessary data for the rest of the process. This software it's only avaliable for Machintosh due of the versatility of the Object Oriented Programing language which it is written. This language is called Objective-C and will be addressed in this work, even briefly.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução e Estado da Arte

A análise numérica assumiu, desde o seu aparecimento, um papel fundamental no campo das ciências, seja ele qual for. Desde os mais simples cálculos estruturais às mais complexas simulações de fluídos, os estudos numéricos são feitos recorrendo à modelação matemática do problema, geração de uma malha e a sua posterior análise.

O presente trabalho foca-se apenas nas simulações estruturais, que partem de um modelo tridimensional digital. Na industria aeronáutica, o interesse por esta matéria deu-se entre os anos 1950 e 1960[2], tendo sido nesta época que surgiram os primeiros programas de desenho assistido por computador (DAC).

Contudo, as modelações que se apresentam fiéis aos modelos reais, não podiam ser utilizadas directamente nas análises numéricas, havendo portanto a necessidade de criar malhas a partir do zero para a análise. As malhas apresentavam portanto diferenças relativamente ao modelo que lhes deu origem, sendo que as malhas apresentam uma geometria aproximada.

Havia-se criado uma distância entre as dias áreas. Esta separação, tornou as alterações nas malhas em alguns casos impossível devido a não existir um *feedback* entre as áreas já mencionadas.[2]

Uma vez que era necessário recomeçar os trabalhos de representação das geometrias do ponto zero, não houve aqui interesse em manter as áreas juntas, dando-se o inevitável afastamento e consequente evolução de forma separada. [4]

Com a intenção de as aproximar novamente, surgiram as análises isogeométricas (AIG). Para conseguir eliminar esse mesmo vazio existente, é necessário fazer com que ambas as partes tenham um diálogo mais próximo e o processo de criação das geometrias seja transversal a ambas as áreas. Isto é, fazer com que a base para obtenção tanto das geometrias 3D dos modelos de DAC, como das malhas para a Análise por Elementos Finitos (AEF) sejam comuns.

Apesar de haver enumeras geometrias computacionais que poderiam servir de base à análise isogeométrica, as *Non-Uniform Rational B-Spline* (NURBS) foram escolhidas uma vez que a sua formulação assenta numa base muito interessante: são usadas para criar o maior leque possível de geometrias computacionais.[1] É também através destas geometrias que se geram ficheiros para trocar informação de DAC [6] nomeadamente os ficheiros do tipo "*.iges".

Neste momento é maior o desafio de gerar elementos finitos fiéis ao modelo de DAC, do que propriamente o de realizar a análise. [1]

No campo das análises numéricas, a falta de fidelidade entre os modelos leva a problemas de instabilidade, sendo bastante notório nos casos de análise de cascas. [2]

É de salientar que tudo na indústria tem o seu custo associado, e as análises numéricas não são excepção. Na indústria automóvel a etapa de geração de uma malha para o veículo completo (é portanto uma malha bastante complexa) pode levar até 4 meses. [2]

Com os presentes métodos de elementos finitos (MEF) uma análise toma os seguintes passos e percentagens mostradas na tabela 1.1.

Tabela 1.1: Processo de cálculo pelo MEF [1]

Etapa	Descrição	Percentagem
1	Criação de um modelo sólido	4%
2	Análise do modelo sólido	20%
3	Decomposição da geometria	32%
4	Criação da malha	14%
5	Manipulação da malha	6%
6	Atribuição de parâmetros ao modelo	6%
7	Construção do modelo para simulação	8%
8	Execução da simulação	4%
9	Análise dos resultados da simulação	5%
10	Gravação dos resultados	1%

Se nos concentrarmos na etapa 3 e 4, vemos que juntas representam 46% do tempo de todo este processo.

Num modelo construído com base em NURBS, reduzimos esta percentagem a um valor insignificante, ou quase nulo uma vez que a informação necessária para gerar o modelo serve de base ao cálculo numérico inerente à análise, eliminando assim a necessidade dessas etapas. [6]

Se estivermos perante um problema de contacto a diferença é logo notória uma vez que a malha se mantém muito mais fiel ao modelo real e evitamos assim problemas de algoritmo. [2]

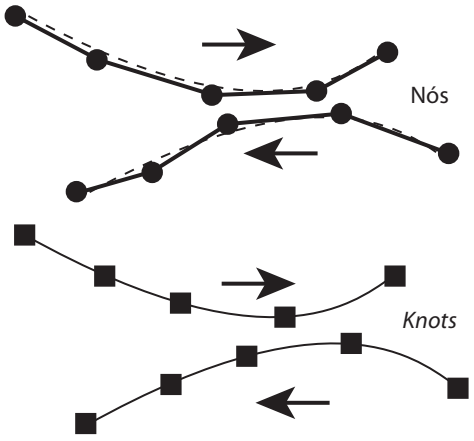
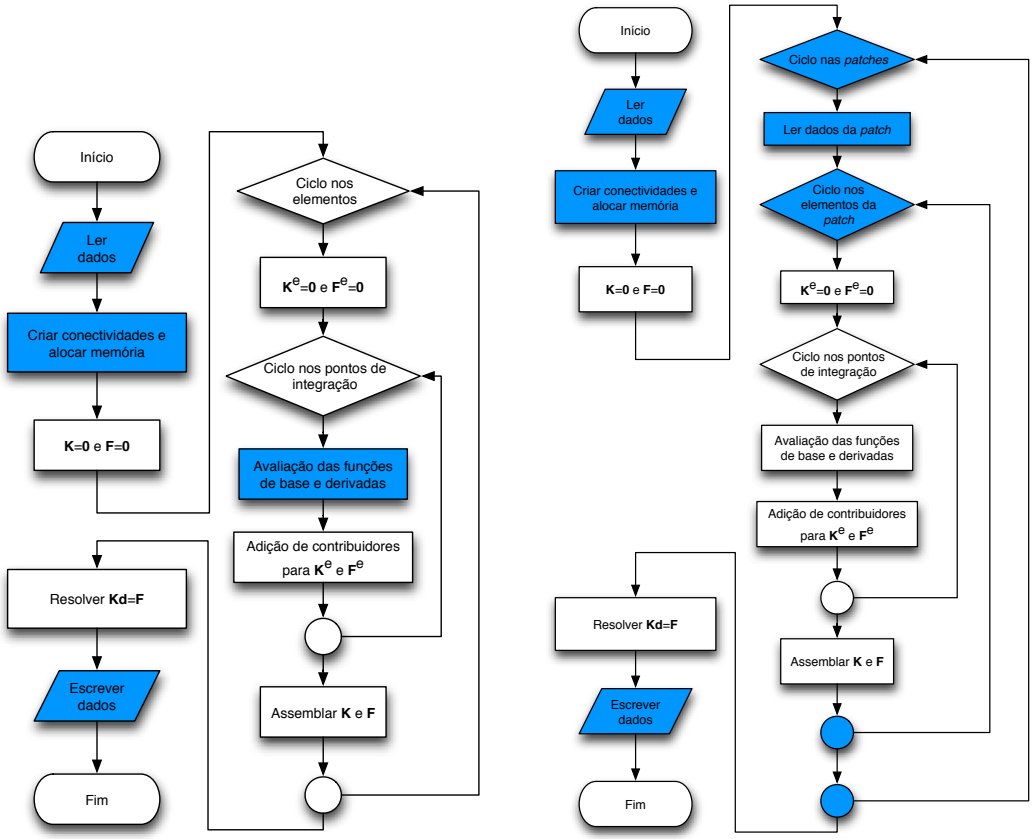


Figura 1.1: Nós de uma malha vs. *knots*[2]



(a) MEF Clássico (b) Isogeométrico

Figura 1.2: Modificação de um algoritmo do MEF clássico para a AIG [2]

Para testar e explorar estas mesmas teorias, foi adaptado um *software* já existente e inicialmente criado apenas com a finalidade de investigação, para *Machintosh*, de forma a ser possível modelar geometrias recorrendo a NURBS. Esse *software* também será alvo de estudo no presente trabalho, uma vez que a linguagem em que está escrito em *Objective-C* uma linguagem de programação muito versátil, e de fácil implementação.

Existe actualmente uma variedade de *software* que permite a modelação de geometrias NURBS, destacando os seguintes:

- Power NURBS
- GID
- CATIA
- Blender
- Rhino 3D
- Autodesk AliasStudio
- Autodesk Maya

Contudo estes *softwares* são, evidentemente, de código fechado ficam-se apenas pela modelação das geometrias, e não permitem assim o acesso a toda a informação que é desejável para explorar convenientemente este tema, nomeadamente aceder aos dados necessários ao *input* do *software* que fará o cálculo.

A opção de um *software* não comercial para uso neste trabalho teve como fim a aquisição de novos conhecimentos bem como consolidar diversos conhecimentos obtidos durante a formação académica, a nível de programação, de cálculo computacional, de modelação numérica – desde a geração da mais simples malha à representação gráfica de geometrias NURBS recorrendo a *OpenGL*.

O *software* utilizado é denominado *Numerical Dynamics* e assume o conceito de pré e pós-processador.

Existe um *software* comercial que detém o mesmo conceito do *Numerical Dynamics*, o GiD.[1]

Para efectuar a análise numérica sobre a malha gerada, foi usado um *software* de nome CEREBRO© [7], que dado um ficheiro gerado pelo pré-processador, efectua os cálculos numéricos e gera um ficheiro de resultados. Este ficheiro de resultados é lido novamente pelo programa que assume agora um papel de pós-processador.

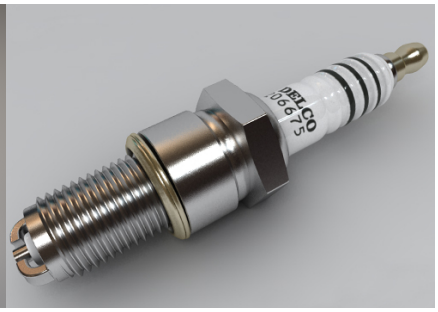
Como trabalho futuro propõe-se que seja feita a adaptação do *software Numerical Dynamics* à importação e exportação de ficheiros do tipo "*.iges". Esta adaptação permitirá uma enorme flexibilidade na interligação deste *software* com *softwares* já existentes, tais como *CATIA* ou *SolidWORKS*. A importação de ficheiros permitirá preparar as modelações feitas com esses *softwares* para uma análise numérica mais específica, como é p.ex. uma análise numérica de um processo de estampagem, ou uma análise de fractura.

A escrita personalização de ficheiros também é um trabalho futuro fulcral para melhorar a flexibilidade deste *software* tornando este trabalho ainda mais abrangente.

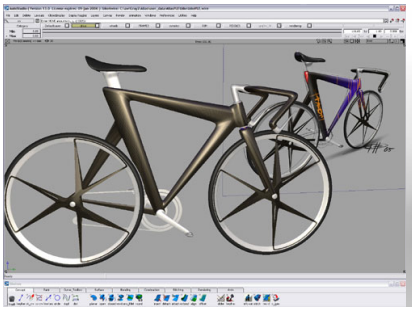
A interactividade com o utilizador também é um aspecto a melhorar nomeadamente na selecção de objectos. Esta deverá ser feita directamente com o clique do rato sem que haja a necessidade de escolher menus e botões para poder alcançar tais funcionalidades.

[1] Como já foi referido atrás, a implementação computacional de todos estes conceitos visa consolidar os conceitos matemáticos inerentes às NURBS para posteriormente melhor compreender a análise isogeométrica.

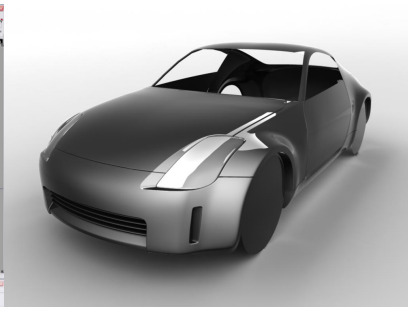
(a) Power NURBS



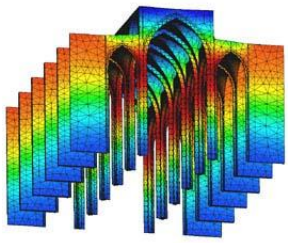
(b) Rhino 3D



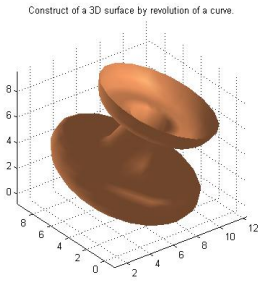
(c) Autodesk AliasStudio



(d) Autodesk Maya



(e) GiD



(f) MATLAB

Figura 1.3: Outras alternativas de *software*

Capítulo 2

Curvas Paramétricas

Para enquadrar este trabalho, é necessário definir alguns conceitos fundamentais às curvas paramétricas, dentro das quais se inserem as *Non-Uniform Rational B-Splines* – NURBS. Comecemos por descrever uma curva paramétrica simples, avançando até uma curva NURBS.

2.1 Curvas de Bézier

Seja uma curva definida em $\mathbb{R}^3$ pelas seguintes funções:

$$x = f(t) \tag{2.1}$$
$$y = g(t) \tag{2.2}$$
$$z = h(t) \tag{2.3}$$

onde t é o parâmetro que pode ser normalizado para valores entre 0 e 1.
O vector que define um ponto da curva é dado por

$$P(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} \tag{2.4}$$

Exemplificando com uma hélice circular, temos os parâmetros

$$x(t) = rcos(t) \tag{2.5}$$
$$y(t) = rsin(t) \tag{2.6}$$
$$z(t) = bt \tag{2.7}$$

ficando assim definido $P(t)$ como:

$$P(t) = \begin{bmatrix} rcos(t) & rsin(t) & bt \end{bmatrix} \tag{2.8}$$

Um caso especial de uma NURBS é uma curva de Bézier, em que as suas funções base pertencem ao domínio $\mathbb{R}$, o grau que define a curva é 1 valor inferior ao número de vértices do polígono de Bézier, também denominados pontos de controlo da curva. A curva segue a forma do polígono, e o primeiro e último ponto de controlo são coincidentes respectivamente com o ponto inicial e final da curva.

Seja $B_0 = [\ 1 \quad 1\ ]$, $B_1 = [\ 2 \quad 3\ ]$, $B_2 = [\ 4 \quad 3\ ]$, $B_3 = [\ 3 \quad 1\ ]$, pontos de controlo da curva de Bézier. Para determinar t pontos da curva de Bézier apliquemos a seguinte expressão:

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t) \tag{2.9}$$

onde

$$J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{2.10}$$

e

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{2.11}$$

como o polígono de Bézier é definido por 4 vértices, o grau n do polinómio que define a curva é $n = 4 - 1 = 3$.

$$\binom{n}{i} = \binom{3}{i} = \frac{6}{i!(3-i)!} \tag{2.12}$$

e as funções base ficam assim definidas como:

$$J_{3,0}(t) = (1)t^0(1-t)^3 = (1-t)^3 \tag{2.13}$$
$$J_{3,1}(t) = 3t(1-t)^2 \tag{2.14}$$
$$J_{3,2}(t) = 3t^2(1-t) \tag{2.15}$$
$$J_{3,3}(t) = t^3 \tag{2.16}$$

dando assim origem à equação $P(t)$ que define a curva de grau 3 como sendo:

$$P(t) = B_0 J_{3,0} + B_1 J_{3,1} + B_2 J_{3,2} + B_3 J_{3,3} \tag{2.17}$$
$$P(t) = (1-t)^3 B_0 + 3t(1-t)^2 B_1 + 3t^2(1-t) B_2 + t^3 B_3 \tag{2.18}$$

Para calcular os valores de sete pontos da curva, foram calculados e determinados os valores em resumo na tabela 2.1.

Tabela 2.1: Valores das funções base para a curva do exemplo

t	$J_{3,0}$	$J_{3,1}$	$J_{3,2}$	$J_{3,3}$
0	1	0	0	0
0.15	0.614	0.325	0.058	0.003
0.35	0.275	0.444	0.239	0.042
0.50	0.125	0.375	0.375	0.125
0.65	0.042	0.239	0.444	0.275
0.85	0.003	0.058	0.325	0.614
1	0	0	0	1

Com estes valores já é possível calcular os pontos da curva.

$$P(0.00) = B_0 = [\ 1 \quad 1\ ] \tag{2.19}$$
$$P(0.15) = 0.614B_0 + 0.325B_1 + 0.058B_2 + 0.003B_3 = [\ 1.500 \quad 1.756\ ] \tag{2.20}$$
$$P(0.35) = 0.275B_0 + 0.444B_1 + 0.239B_2 + 0.042B_3 = [\ 2.248 \quad 2.367\ ] \tag{2.21}$$
$$P(0.50) = 0.125B_0 + 0.375B_1 + 0.375B_2 + 0.125B_3 = [\ 2.750 \quad 2.500\ ] \tag{2.22}$$
$$P(0.65) = 0.042B_0 + 0.239B_1 + 0.444B_2 + 0.275B_3 = [\ 3.122 \quad 2.367\ ] \tag{2.23}$$
$$P(0.85) = 0.003B_0 + 0.058B_1 + 0.325B_2 + 0.641B_3 = [\ 3.248 \quad 1.765\ ] \tag{2.24}$$
$$P(1.00) = B_3 = [\ 3 \quad 1\ ] \tag{2.25}$$

Os valores da funções base ditam o peso que cada vértice do polígono de controlo irá ter no cálculo da curva.

O resultado da curva é visivel na figura 2.1

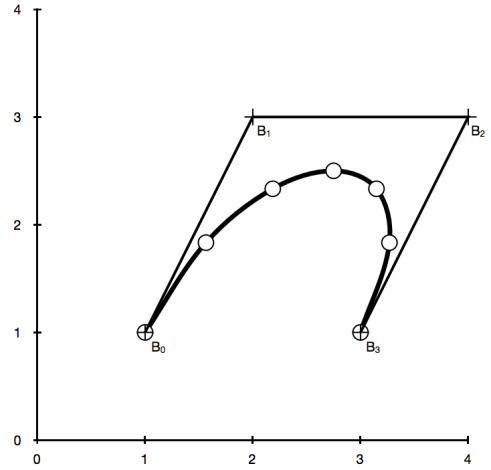


Figura 2.1: Polígono de Bézier e curva resultante

Com este exemplo simples é possível ter uma ideia de como se obtem as curvas paramétricas.

2.2 Curvas B-Spline

Avançando agora para uma curva do tipo *B-Spline* , seja $P(t)$ um vector posição na curva em função do parâmetro t, uma curva *B-Spline* é obtida com a seguinte expressão:

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t) \quad t_{min} \leq t \leq t_{max}, \quad 2 \leq k \leq n+1 \tag{2.26}$$

onde B_i são os vectores de posição dos $n+1$ vértices do polígono de controlo e $N_{i,k}$ são as funções base normalizadas das curvas de Bézier.

Para cada i função base normalizada de ordem k (grau $k-1$) a função base $N_{i,k}$ é definida por:

$$N_{i,1}(t) = \begin{cases} 1 & \text{se } x_i \leq t \leq x_{i+1} \\ 0 & \text{restantes} \end{cases} \tag{2.27}$$

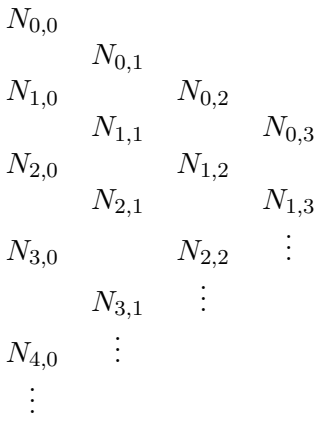


Figura 2.2: Relação encadeada para a obtenção das funções de forma [3]

quando o grau da função é 1 e nos restantes casos é definida de forma iterativa seguindo o fluxo de cálculo apresentado na figura 2.2 sendo o primeiro termo dado por:

$$N_{i,k}(t) = \frac{(t - x_i)N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}} \tag{2.28}$$

e os valores de x_i são elementos de um vector de *knots* que satisfazem a condição $x_i \leq x_{i+1}$, ou seja terão sempre que ser crescentes ou iguais ao seu antecessor. Por uma questão de ambiguidade iremos usar o termo *knots* para distinguir estes últimos dos nós usados em malhas de elementos finitos.

Como esquematizado na figura 2.2 para calcular uma função de base, requer que se calculem as suas antecedentes, fazendo com que haja assim uma relação encadeada entre as funções base. Se quisermos obter a função $N_{1,2}$ temos que calcular previamente as funções $N_{1,0}$, $N_{2,0}$, $N_{3,0}$, $N_{1,1}$, $N_{2,1}$. Para consolidar esta situação temos o exemplo seguinte [3]:

Seja $U = \{u_0 = 0, u_1 = 0, u_2 = 0, u_3 = 1, u_4 = 1, u_5 = 1\}$ um vectore de *knots* e $p = 2$ o grau das funções de base, temos as próximas funções base com os graus 0, 1 e 2:

No caso de ser apresentada uma indeterminação $\frac{0}{0}$ assumimos que o resultado final será zero.

$$N_{0,0} = N_{1,0} = 0 \qquad -\infty < u < \infty \tag{2.29}$$

$$N_{2,0} = \begin{cases} 1 & \text{se } 0 \leq u < 1 \\ 0 & \text{restantes} \end{cases} \tag{2.30}$$

$$N_{3,0} = N_{4,0} = 0 \qquad -\infty < u < \infty \tag{2.31}$$

$$N_{0,1} = \frac{u-0}{0-0}N_{0,0} + \frac{0-u}{0-0}N_{1,0} = 0 \qquad -\infty < u < \infty \tag{2.32}$$

$$N_{1,1} = \frac{u-0}{0-0}N_{1,0} + \frac{1-u}{1-0}N_{2,0} = \begin{cases} 1-u & 0 \leq u < 1 \\ 0 & \text{restantes} \end{cases} \tag{2.33}$$

$$N_{2,1} = \frac{u-0}{1-0}N_{2,0} + \frac{1-u}{1-1}N_{3,0} = \begin{cases} u & 0 \leq u < 1 \\ 0 & \text{restantes} \end{cases} \tag{2.34}$$

$$N_{3,1} = \frac{u-1}{1-1}N_{3,0} + \frac{1-u}{1-1}N_{4,0} = 0 \qquad -\infty < u < \infty \tag{2.35}$$

$$N_{0,2} = \frac{u-0}{0-0}N_{0,1} + \frac{1-u}{1-0}N_{1,1} = \begin{cases} (1-u)^2 & 0 \leq u < 1 \\ 0 & \text{restantes} \end{cases} \tag{2.36}$$

$$N_{1,2} = \frac{u-0}{1-0}N_{1,1} + \frac{1-u}{1-0}N_{2,1} = \begin{cases} 2u(1-u) & 0 \leq u < 1 \\ 0 & \text{restantes} \end{cases} \tag{2.37}$$

$$N_{2,2} = \frac{u-0}{1-0}N_{2,1} + \frac{1-u}{1-1}N_{3,1} = \begin{cases} u^2 & 0 \leq u < 1 \\ 0 & \text{restantes} \end{cases} \tag{2.38}$$

2.2.1 Derivadas das funções de forma

A primeira derivada de uma função de forma é dada por:

$$N'_{i,p} = \frac{p}{u_{i+p} - u_i}N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - u_{i+1}}N_{i+1,p-1}(u) \tag{2.39}$$

Se generalizarmos para uma ordem de derivação k, temos a seguinte expressão:

$$N^{(k)}_{i,p}(u) = p\left(\frac{N^{(k-1)}_{i,p-1}}{u_{i+p} - u_i} - \frac{N^{(k-1)}_{i+1,p-1}}{u_{i+p+1} - u_{i+1}}\right) \tag{2.40}$$

A equação 2.41 calcula a derivada de ordem k com base em $N_{i,p-k}, ..., N_{i+k,p-k}$

$$N^k_{i,p} = \frac{p!}{(p-k)!}\sum_{j=0}^{k} a_{k,j}N_{i+j,p-k} \tag{2.41}$$

com

$$a_{0,0} = 1 \tag{2.42}$$

$$a_{k,0} = \frac{a_{k-1,0}}{u_{i+p+k} - u_i} \tag{2.43}$$

$$a_{k,j} = \frac{a_{k-1,j} - a_{k-1,j-1}}{u_{i+p+j-k} - u_{i+j}} \; ; \; j = 1, ..., k-1 \tag{2.44}$$

$$a_{k,k} = \frac{-\, a_{k-1,k-1}}{u_{i+p+1} - u_{i+k}} \tag{2.45}$$

com esta reescrita podemos tirar as seguintes conclusões:

- k nunca deve exceder p (todas as derivadas seguintes serão 0);
- os denominadores podem ser zero, e nesse caso o quociente é zero por defeito.

2.2.2 Propriedades convexas das curvas

As curvas *B-Splines* partilham uma propriedade interessante do ponto de vista visual que é a existência de fronteiras virtuais, entre as quais a curva ficará contida. Como vemos na figura 2.3 essas fronteiras variam consoante o grau $k-1$ das funções base. Quanto mais baixo for o valor de k mais a curva se aproxima do seu polígono de controlo. Esta característica permite alterar a curva sem se ter que alterar os pontos de controlo, existindo outras alternativas ao controlo da curva como veremos mais a frente neste trabalho. [4]

2.2.3 Vectores de *knots*

Um vector de *knots* têm influência directa nas funções de base $N_{i,k}(t)$ como foi explicito na equação 2.26 e por sua vez na curva resultante. Como já foi referido anteriormente o único requisito que este vector tem que satisfazer é ser um vector de números monótonos crescentes. Por convenção começam em 0 e são normalizados para terminar em 1, e todos os seus valores ficarem igualmente espaçados dentro destes limites.

$$\Xi = \; [0 \quad 1 \quad 2 \quad 3 \quad 4] \tag{2.46}$$

$$\Xi = \; [-0.2 \quad -0.1 \quad 0 \quad 1 \quad 2] \tag{2.47}$$

$$\Xi = \; [0 \quad 0.25 \quad 0.5 \quad 0.75 \quad 1] \tag{2.48}$$

Os exemplos 2.46, 2.47 e 2.48 são vectores de *knots* válidos sendo que 2.48 é o mais convencional e recomendado de utilizar, pois os seus valores estão entre 0 e 1.

Estes vectores são vectores periódicos, que por definição são vectores que não repetem o primeiro e o último *knot* k vezes, sendo k a ordem das funções de base da curva.

Um vector uniforme aberto tem a multiplicidade de cada *knot* no início e no final igual à ordem k da função de base. Os restantes valores são igualmente espaçados.[4]

Um vector uniforme aberto é construído da seguinte forma:

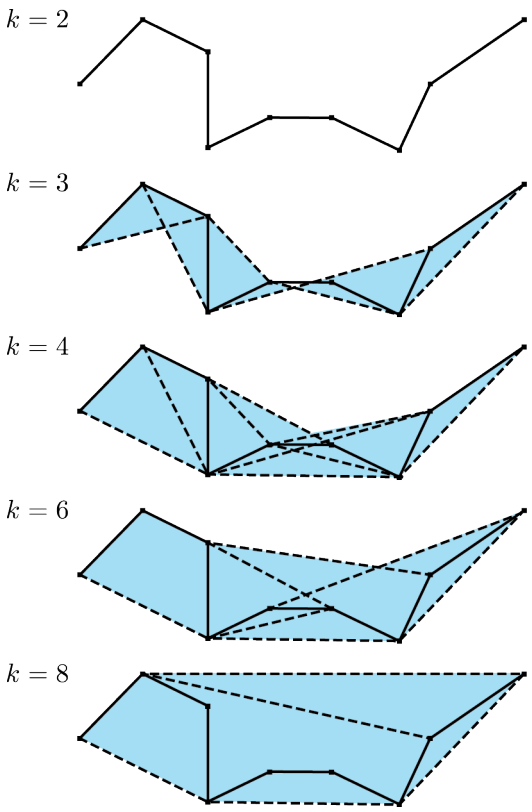


Figura 2.3: Propriedades convexas de uma *B-Spline* [4]

$$x_i = 0 \qquad 1 \leq i \leq k \tag{2.49}$$
$$x_i = i - k \qquad k+1 \leq i \leq n+1 \tag{2.50}$$
$$x_i = n - k + 2 \qquad n+2 \leq i \leq n+k+1 \tag{2.51}$$

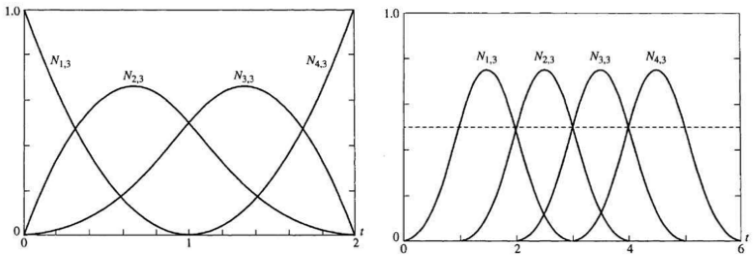
Concretizando o algoritmo anterior, temos:

$$\text{Para } k = 2 \qquad \Xi = [\ 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 4\ ] \tag{2.52}$$
$$\text{Para } k = 3 \qquad \Xi = [\ 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 3 \quad 3\ ] \tag{2.53}$$
$$\text{Para } k = 4 \qquad \Xi = [\ 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2\ ] \tag{2.54}$$

normalizando os elementos entre os valores 0 e 1 ficando assim:

$$\text{Para } k = 2 \qquad \Xi = [\ 0 \quad 0 \quad 1/4 \quad 1/2 \quad 3/4 \quad 1 \quad 1\ ] \tag{2.55}$$
$$\text{Para } k = 3 \qquad \Xi = [\ 0 \quad 0 \quad 0 \quad 1/3 \quad 2/3 \quad 1 \quad 1 \quad 1\ ] \tag{2.56}$$
$$\text{Para } k = 4 \qquad \Xi = [\ 0 \quad 0 \quad 0 \quad 0 \quad 1/2 \quad 1 \quad 1 \quad 1 \quad 1\ ] \tag{2.57}$$

Outro aspecto bastante importante é o controlo da curva *B-Spline* em termos de ajuste ao seu polígono de controlo. Existem no entanto outras formas para alterar a curva. Enuncia-se de seguida essas possibilidades:



(a) (b)

Figura 2.4: Funções de base para vectores uniformes abertos a) periódicos b)

- Modificação do tipo de vector de *knots*, alternando entre um vector periódico uniforme, aberto uniforme, e não uniforme;
- Alteração do valor k das funções base, ou seja a sua ordem;
- Adição ou eliminação de pontos de controlo, bem como alteração as suas posições;
- Criação de multiplicidade de um vértice do polígono de controlo, ou eliminação dessa multiplicidade;
- Utilização de repetidos valores de um *knot* no vector de *knots*.

Na figura 2.5 podemos ver as alterações que uma curva sofre quando se promovem algumas das alterações referidas atrás. No caso particular da figura 2.5 d) foi alterado o vector de *knots* da equação 2.58, que origina a curva a), para o vector de *knots* da equação 2.59

$$[X] = [\ 0 \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 3 \quad 3\ ] \tag{2.58}$$

para

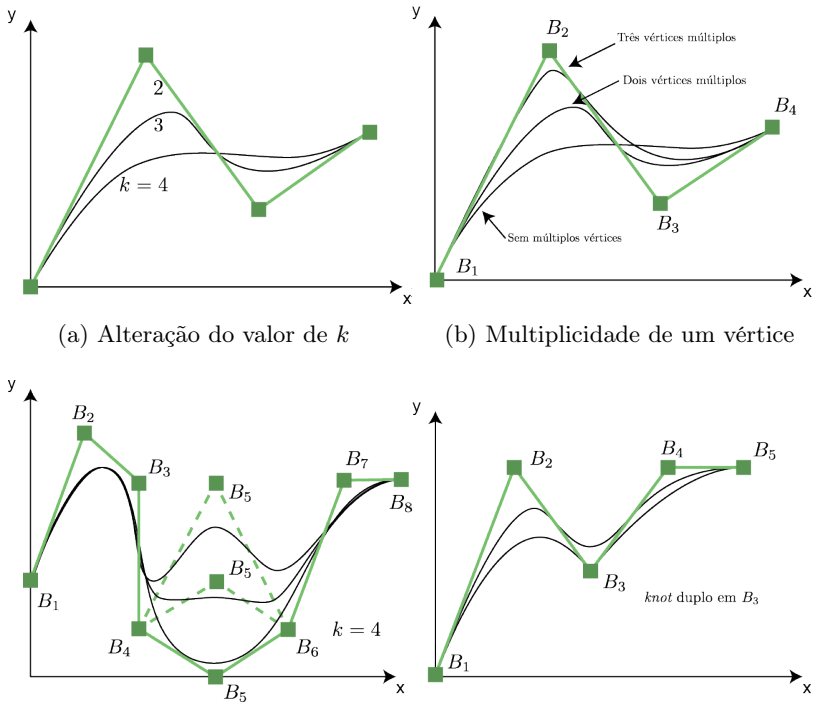
$$[X] = [\ 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 3 \quad 3 \quad 3\ ] \tag{2.59}$$

2.2.4 Modificação e refinamento da curva

Existem várias formas de modificar uma curva, seja com a intenção de apenas alterar a mesma, seja com a intenção de a refinar. Estando perante uma curva esse refinamento é feito segundo a única direcção paramétrica existente, ξ.

Podemos, p.ex., fazer este refinamento através da introdução de um novo *knot*. Os *knots* podem ser inseridos sem alterar a curva geometricamente ou paramétricamente.

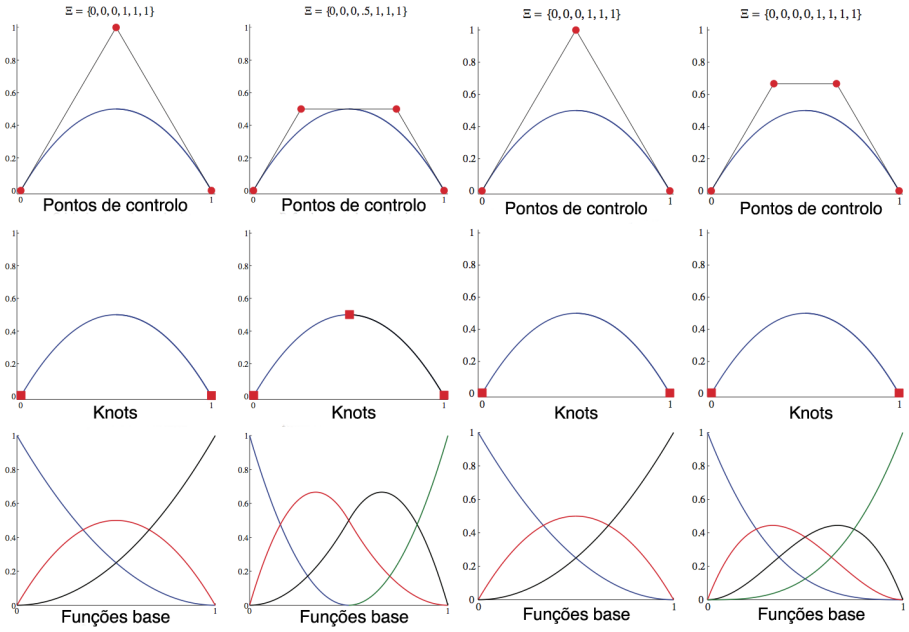
Uma outra forma de refinar a curva é a elevação da ordem das funções de base. Esta técnica consiste em aumentar uma vez cada *knot*, não inserindo qualquer novo valor no vector de *knots* ao contrário da introdução de *knots*.



(a) Alteração do valor de k (b) Multiplicidade de um vértice

(c) Modificação de um vértice (d) Modificação de vector de *knots*

Figura 2.5: Diferentes ajustamentos das curvas *B-Spline* [3]



(a) Introdução de um *knot* (b) Elevação do grau k

Figura 2.6: Refinamento de uma curva NURBS [1]

2.3 Non-Uniform Rational B-Splines: NURBS

Para se calcular todos os pontos de uma curva NURBS, é necessário interpolar os pontos de controlo, os pesos de cada um dos pontos de controlo e as funções de base que por sua vez são definidas pelos vectores de *knots*.

Seja $C(u)$ uma curva NURBS de grau p o que significa que as suas funções base são polinómios de grau p.

$$\boldsymbol{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \boldsymbol{P_i}}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \quad a \leq u \leq b \tag{2.60}$$

e o vector de *knots* U definido por:

$$U = [\underbrace{a, ..., a}_{p+1}, u_{p+1}, ..., u_{m-p-1}, \underbrace{b, ..., b}_{p+1}] \tag{2.61}$$

Os valores a e b serão por padrão 0 e 1 respectivamente, e o peso de cada nó, w_i, será sempre superior ou igual a zero.

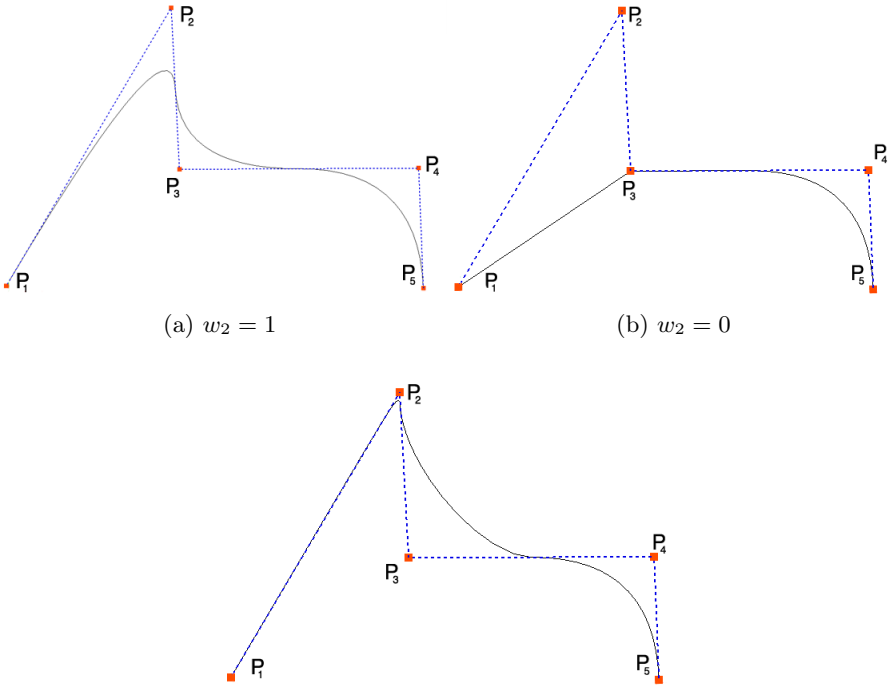
O peso atribuído a cada ponto de controlo faz variar a forma da curva como vemos na figura 2.7. Se todos os pesos fossem iguais a 1, $w_i = 1$ estaríamos perante uma *B-Spline*.

Com os pesos de cada ponto de controlo, podemos definir uma nova função base R dada pela seguinte expressão:

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{j=0}^{n} N_{j,p}(u) w_j} \tag{2.62}$$

com a expressão 2.62 é possível rescrever a equação 2.60 da seguinte forma:

$$\boldsymbol{C}(u) = \sum_{i=0}^{n} R_{i,p}(u) \boldsymbol{P_i} \tag{2.63}$$



(a) $w_2 = 1$ (b) $w_2 = 0$

(c) $w_2 = 3$

Figura 2.7: Variação do peso do ponto de controlo P_2

2.4 Interpolação de curvas, superfícies e sólidos

Dado um conjunto de pontos, no espaço XY é possível criar uma curva que ajustará ao mesmos, bem como dados pontos no espaço XYZ uma superfície que se ajuste aos mesmos. O mesmo é possível para os sólidos dadas as superfícies de controlo segundo as quais o sólido ficará contido. Para satisfazer estas condições, existe a variante da aproximação e da interpolação.

A aproximação é utilizada para um conjunto de pontos em que não é obrigatório todos os pontos pertencerem à superfície como é o caso de pontos gerados por um dispositivo de medição por coordenadas, que poderá introduzir algum ruído nas medições ou mesmo existir uma tolerância que é aceitável e apenas alguns pontos críticos terem que forçosamente pertencer à superfície.

Para os requisitos do presente trabalho, é de maior interesse a interpolação dos pontos, uma vez que nos interessa criar do zero essa curva, superfície ou sólido de forma precisa, e não diante da reconstrução da mesma a partir de um modelo físico já existente.

2.4.1 Interpolação de curvas

Dado um conjunto de n pontos $\{\mathbf{Q}_k\}$ com $k = 0, ..., n$ podemos interpolar uma curva de grau p com um parâmetro $\bar{u}_k$ para cada ponto $\mathbf{Q}_k$, conjuntamente com um vector de *knots* $U = u_0, ..., u_m$ podemos obter um sistema de equações lineares com as dimensões $(n+1) \times (n+1)$ através da seguinte expressão:

$$\mathbf{Q}_k = \mathbf{C}(\bar{u}_k) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\mathbf{P}_i \tag{2.64}$$

Este sistema tem como incógnitas os $n+1$ pontos de controlo $\mathbf{P}_i$.

Outra questão importante é a determinação dos valores de $\bar{u}_k$. Neste ponto é fundamental optar por um vector cujos valores são definidos pelo método da corda de circulo.

O método das corda de círculo permite determinar esses mesmos valores utilizando as seguintes expressões:

$$d = \sum_{k=1}^{n} |\mathbf{Q_k} - \mathbf{Q_{k-1}}| \tag{2.65}$$

$$\bar{u}_0 = 0 \quad \bar{u}_n = 1$$

$$\bar{u}_k = \bar{u}_{k-1} + \frac{|\mathbf{Q_k} - \mathbf{Q_{k-1}}|}{d} \qquad k = 1, ..., n-1 \tag{2.66}$$

Para determinar, por fim, o vector de *knots*, utiliza-se um método que é o reflexo da distribuição dos pontos de $\bar{u}_k$ e que é obtido com as expressões seguintes:

$$u_0 = ... = u_p = 0 \qquad u_{m-p} = ... = u_m = 1$$

$$u_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p-1} \bar{u}_i \qquad j = 1, ..., n-p \tag{2.67}$$

2.4.2 Interpolação de superfícies

Seguindo o raciocínio da secção 2.4.1 a interpolação de superfícies faz-se incluindo mais uma dimensão no cálculo dos pontos que, neste caso, irão definir a superfície. Concretizando numa equação este raciocínio, temos o seguinte:

$$\mathbf{Q}_{k,l} = \mathbf{S}(\bar{u}_k, \bar{v}_l) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(\bar{u}_k)N_{j,q}(\bar{v}_l)\mathbf{P}_{i,j} \tag{2.68}$$

As funções $\bar{u}_k$ e $\bar{v}_l$ são, no caso das superfícies, dadas por um método de cálculo descrito a seguir:

$$\bar{u}_k = \frac{1}{m+1}\sum_{l=0}^{m} \bar{u}_k^l \qquad\qquad k = 0, ..., n \tag{2.69}$$

$$\bar{v}_l = \frac{1}{n+1}\sum_{k=0}^{n} \bar{v}_l^k \qquad\qquad l = 0, ..., m \tag{2.70}$$

Com estas funções, fica a faltar apenas, calcular os vectores de *knots* segundo ξ e η que são obtidos pela equação 2.67 à semelhança das curvas.

Como $\mathbf{S}(u, v)$ é uma superfície dada por um produto tensores, a mesma pode ser rescrita como uma sequência de interpolações. Fixando o valor de l podemos rescrever a equação 2.68:

$$\mathbf{Q}_{k,l} = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\left(\sum_{j=0}^{m} N_{j,k}(\bar{v}_l)\mathbf{P}_{i,j}\right) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k)\mathbf{R}_{i,l} \tag{2.71}$$

com $\mathbf{R}_{i,l}$ definido por:

$$\mathbf{R}_{i,l} = \sum_{j=0}^{m} N_{j,k}(\bar{v}_l)\mathbf{P}_{i,j} \tag{2.72}$$

A superfície resultante é comutativa, isto é, podemos interpolar primeiro segundo x e depois segundo y ou vice-versa, que o resultado será o mesmo.

Por forma a melhor ilustrar o que foi mencionado atrás, podemos ver o exemplo da figura 2.8.

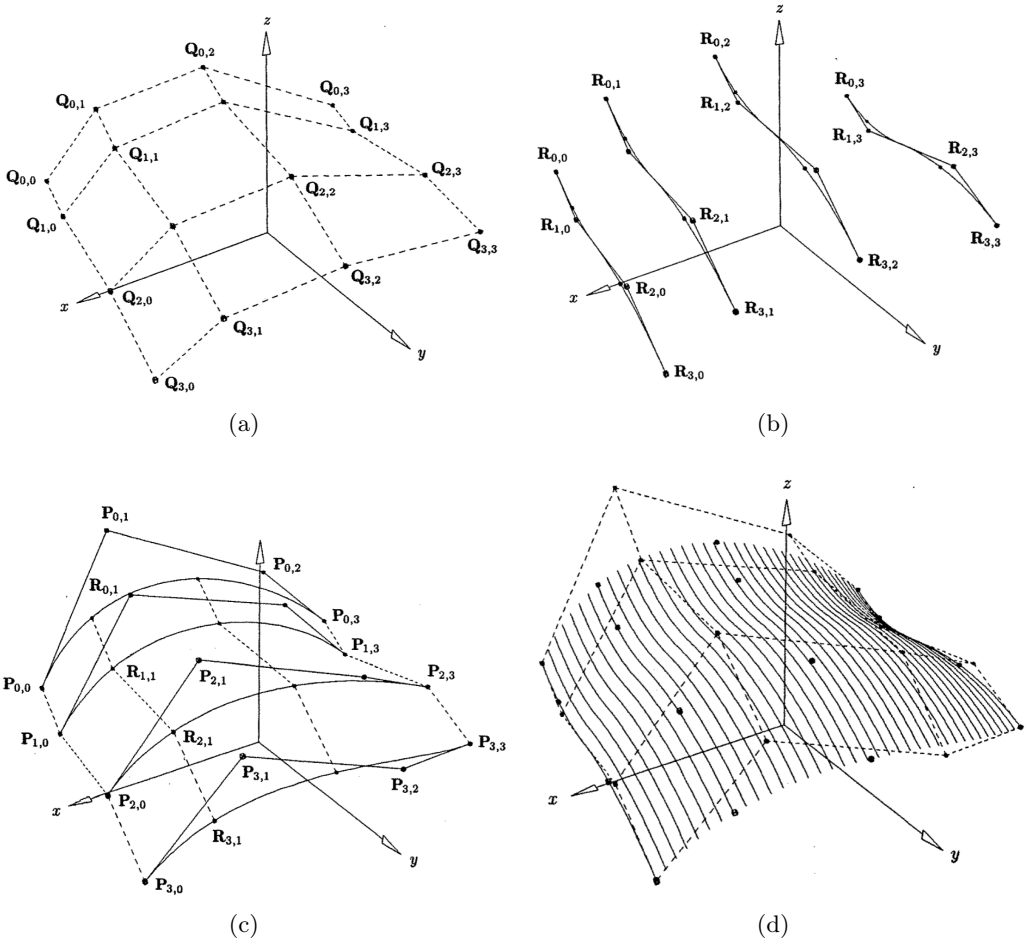


Figura 2.8: Interpolação de superfícies [3]

2.4.3 Interpolação de sólidos

Tendo já sido apresentado atrás a interpolação de curvas e de superfícies, fica a faltar apenas apresentar a interpolação de sólidos para se conseguir obter um modelo NURBS de um objecto físico.

Até agora, vimos que podemos interpolar uma curva que possui apenas uma dimensão paramétrica, uma superfície que tem duas dimensões paramétricas, e veremos de seguida como fazer uma interpolação que origina um sólido, portanto um objecto com três dimensões paramétricas.

Seguindo a linha de raciocínio mantida até agora, tanto na interpolação de curvas como na interpolação de superfícies, se for feita nova interpolação à expressão 2.68 segundo uma nova dimensão, temos como resultado uma NURBS que descreve um sólido.

Para interpolar um sólido basta apenas reformular a expressão 2.68 acrescentando mais uma dimensão, isto é, aplicando um somatório a toda a expressão. Com essa alteração obtemos a seguinte equação:

$$\mathbf{Q}_{k,l,t} = \mathbf{S}(\bar{u}_k, \bar{v}_l, \bar{w}_t) = \sum_{i=0}^{n} N_{i,p}(\bar{u}_k) \left(\sum_{j=0}^{m} N_{j,q}(\bar{v}_l) \left(\sum_{c=0}^{e} N_{c,r}(\bar{w}_t) \mathbf{P}_{c,i,j} \right) \right) \tag{2.73}$$

O termo $\bar{w}_t$ na expressão 2.73 é obtido tal com na expressão 2.69, sendo que este assume a seguinte forma:

$$\bar{w}_t = \frac{1}{e+1} \sum_{c=0}^{e} \bar{w}_t^c \qquad t = 0, ..., e \tag{2.74}$$

Este conceito é também conhecido como NURBS de elevada ordem, e torna-se difícil de visualizar graficamente o resultado final. Para tal, podemos atentar à figura 2.9 que demonstra precisamente esse caso.

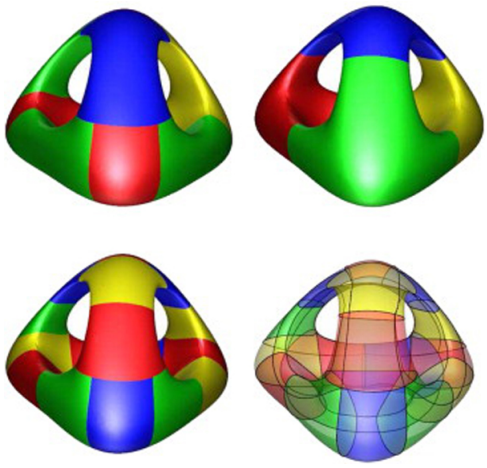


Figura 2.9: Sólido representado por NURBS [5]

Capítulo 3

Cálculo Isogeométrico

No capítulo anterior foi descrita a versatilidade das NURBS tal como a sua precisão na representação de geometrias. Neste capítulo veremos a ligação das duas áreas de interesse neste trabalho, DAC e AEF, através da análise isogeométrica.

Inicialmente foi referido que as geometrias representadas por NURBS são bastante precisas e fiéis ao modelo físico que as mesmas representam. Isso deve-se às funções de base que permitem calcular os vários pontos da curva. São essas funções que na análise numérica serão utilizadas para efectuar os cálculos e obter portanto os resultados pretendidos.

Já as AEF contrastam com estas pois as funções base usadas no cálculo numérico, não são fiéis na recriação da geometria do modelo físico, e quando essa recriação é feita, existe uma considerável discrepância. Esta margem é tanto mais alargada quanto mais complexas forem as formas da geometria em causa.

Tabela 3.1: Diferenças entre a AIG e o MEF [1]

Análise isogeométrica	Método dos elementos finitos
Geometria exacta	Geometria aproximada
Pontos de controlo	Nós
Variáveis de controlo	Variáveis nodais
Funções não interpolatórias	Funções interpolatórias
Baseado em NURBS	Baseado em polinómios
Alta continuidade (controlável)	Continuidade C^0 fixa
Funções base sempre positivas	Funções base não obrigatóriamente positivas
Propriedades de envolvente convexa	Sem propriedades
Variações geométricas suaves	Variações abruptas

Das diferenças mencionadas na tabela 3.1 a que mais relevância tem é sem dúvida a exactidão com que se representa a geometria. De seguida destaca-se o refinamento das malhas. Aqui o mesmo factor que proporciona uma exactidão ímpar da geometria, também tem implicações ao nível do refinamento da geometria que no caso da AIG é independente de um modelo original, pois a informação é a mesma. Por fim destaca-se o facto de nem os pontos de controlo nem as variáveis de controlo serem, no caso da AIG, interpolatórias ao contrário do que sucede na AEF com os nós e as variáveis nodais.

3.1 Método dos resíduos pesados - Método de Galerkin

É importante lembrar mais uma vez que numa análise numérica os resultados obtidos nunca são exactos, sendo por sua vez aproximados e havendo sempre um erro associado aos mesmos. Será mostrado de seguida um método que visa minimizar esse mesmo erro. Mais precisamente minimizar o resíduo, uma vez que não estamos perante uma igualdade. Ao minimizarmos o resíduo estamos a aproximarmo-nos do valor exacto e consequentemente a diminuir o erro.

Se tivermos um problema definido por

$$Au = f \qquad \text{num domínio } \Omega \tag{3.1}$$

$$Bu = r \quad \text{a definição da fronteira num domínio } \Gamma \tag{3.2}$$

$$\hat{u}_n = \psi + \sum_{i=1}^{n} a_i \phi_i \tag{3.3}$$

for uma função que aproxima do valor de u podemos rescrever a expressão que define o problema, da seguinte forma:

$$A\hat{u}_n - f = R_n \Leftrightarrow \tag{3.4}$$

$$\Leftrightarrow A\psi + \sum_{i=1}^{n} a_i \phi_i - f = R_n \approx 0 \tag{3.5}$$

e R_n é a n-ésima aproximação, logo quantos mais termos tivermos melhor será a aproximação

Para reduzir este resíduo o máximo possível, impõe-se um número de integrais do resíduo no domínio Ω afectado de um peso, seja zero:

$$\int_{\Omega} w_i R_n d\Omega = \int_{\Omega} w_i \left\{ A\psi + \sum_{j=1}^{n} a_j \varphi_j - f \right\} d\Omega = 0, \ i = 1, ..., m \tag{3.6}$$

w_i com i a variar entre 1 e m são funções de peso arbitrárias. Quando $m = n$ temos um sistema de n equações e n incógnitas a_j.

Se utilizarmos como funções de peso w_i as funções de forma ϕ_i estamos a utilizar o método de Galerkin. Este método é bastante conveniente uma vez que leva a uma matriz dos coeficientes simétrica, simplificando deste modo a solução.

3.2 Formulação fraca

Também é possível obter os coeficientes de uma expansão que nos dará a aproximação à solução da equação diferencial pretendida, mesmo que não se satisfaça à partida as condições fronteira. Não obstante este processo é mais elaborado, pois para manter a precisão da formulação forte, vista anteriormente, são necessárias mais funções de forma. Destaca-se também o facto de poderem existir condições fronteira que envolvem derivadas da própria função, o que complicará o problema caso as fronteiras sejam irregulares, como é o caso de uma geometria curvilínea.

Retomemos a expressão 3.6:

$$\int_{\Omega} w^{\Omega} R_n^{\Omega} d\Omega = \int_{\Omega} w^{\Omega}(A_n - f) d\Omega \tag{3.7}$$

$$\int_{\Gamma} w^{\Gamma} R_n^{\Gamma} d\Gamma = \int_{\Gamma} w^{\Gamma}(B_n - r) d\Gamma \tag{3.8}$$

$$\int_{\Omega} w^{\Omega} R_n^{\Omega} d\Omega + \int_{\Gamma} w^{\Gamma} R_n^{\Gamma} d\Gamma = 0 \tag{3.9}$$

Na equação 3.7 se substituirmos o termo que contém A a equação pode ser rescrita da seguinte maneira:

$$\int_{\Omega} w^{\Omega} A_n d\Omega = \int_{\Omega} (C w^{\Omega})(D_n) d\Omega + \int_{\Gamma} w^{\Omega}(E_n) d\Gamma \tag{3.10}$$

Esta expressão é chamada então de formulação fraca dos resíduos pesados, sendo C, D, E operadores lineares de ordem de derivação inferior a A. As condicionantes das funções de forma a utilizar, quanto à ordem de derivação são menores.

3.3 Aplicação da formulação fraca à Teoria da Elasticidade

Seja σ_{ij} o tensor das tensões e F_i o vector das forças que actuam no corpo. A expressão da qual resulta o equilíbrio da elasticidade é a seguinte:

$$\frac{\partial \sigma_{ij}}{\partial X_j} + F_i = 0, \ (i, j = 0, 1, 2, 3) \tag{3.11}$$

Esta expressão garante que haja sempre equilíbrio estático com a evolução das tensões aplicadas.

Recorrendo à Lei de Hooke, obtêm-se as tensões com a seguinte expressão:

$$\sigma = \mathbf{D}\epsilon \tag{3.12}$$

Em que $\mathbf{D}$ é a matriz definida na equação 3.13:

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{3.13}$$

Esta matriz relaciona as tensões com as deformações e ϵ_{rs} é o tensor das deformações que é possível obter com a expressão apresentada na equação seguinte, sendo u_r e u_s são vectores de deslocamentos:

$$\epsilon_{rs} = \frac{1}{2}\left(\frac{\partial u_r}{\partial X_s} + \frac{\partial u_s}{\partial X_r}\right), \ (r,s = 1,2,3) \tag{3.14}$$

Por sua vez o tensor das deformações pode ser descrito também de forma matricial:

$$\begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{xx} \\ \epsilon_{xx} \\ 2\epsilon_{xy} \\ 2\epsilon_{xz} \\ 2\epsilon_{yz} \end{Bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial z} & 0 \\ \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix}}_{\mathcal{L}} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \mathcal{L} \cdot \boldsymbol{u} \tag{3.15}$$

Voltando à equação da Lei de Hooke, é possível rescrever a mesma, ficando assim:

$$\boldsymbol{\sigma} = \boldsymbol{D\epsilon} = \boldsymbol{D\mathcal{L}u} \tag{3.16}$$

Posto isto, para aproximar a solução da equação de equilíbrio da Teoria da Elasticidade, será escolhida uma função aproximante $\hat{\sigma}_{ij}$ ao tensor das tensões, obtendo desta forma o resíduo R^{Ω}:

$$R^{\Omega} = \frac{\partial \hat{\sigma}_{ij}}{\partial X_j} + F_i \tag{3.17}$$

Será este resíduo que se pretende minimizar o mais possível, dentro de um certa tolerância, isto é de um valor aceitável.

Supondo agora que estamos perante as seguintes condições fronteira:

$$\sigma_{ij} \cdot n_j = T_i \text{ em } \Gamma_\sigma \tag{3.18}$$

$$u_i = \bar{u} \text{ em } \Gamma_u \tag{3.19}$$

A fronteira Γ_u será satisfeita de forma exacta pela escolha certa da função ψ e das funções de forma N_i. Já Γ_σ será satisfeita de forma aproximada, originando um resíduo na fronteira dado por:

$$R^\Gamma = \hat{\sigma}_{ij} \cdot n_j - T_i \tag{3.20}$$

Aplicando o Método de Gelerkin é possível minimizar os resíduos no domínio e na fronteira aplicando os seguintes integrais:

$$\int_\Omega \left(\frac{\partial \hat{\sigma}_{ij}}{\partial X_j} + F_i \right) W_i^l d\Omega + \int_{\Gamma_\sigma} (\hat{\sigma}_{ij} \cdot n_j - T_i) \overline{W}_i^l d\Gamma_\sigma = 0 \tag{3.21}$$

Seja W_i^l a função de peso no domínio, $\overline{W}_i^l$ a função de peso na fronteira e o índice l os l termos que inclui o somatório da função aproximante.

Aplicando o teorema de Green resulta a seguinte expressão:

$$-\int_\Omega \hat{\sigma}_{ij} \frac{\partial W_i^l}{\partial X_j} d\Omega + \int_\Omega F_i W_i^l d\Omega + \int_{\Gamma_u + \Gamma_\sigma} \hat{\sigma}_{ij} \cdot n_j W_i^l d\Gamma + \int_{\Gamma_\sigma} (\hat{\sigma}_{ij} \cdot n_j - T_i) \overline{W}_i^l d\Gamma = 0 \tag{3.22}$$

Ao escolher como funções de peso as seguintes funções

$$W_i^l = 0 \text{ em } \Gamma_u \tag{3.23}$$

$$\overline{W}_i^l = -W_i^l \text{ em } \Gamma_\sigma \tag{3.24}$$

obtemos a condição fronteira $\sigma_{ij} \cdot n_j$ como sendo uma uma condição fronteira natural, dando assim origem à formulação fraca que se segue

$$\int_\Omega \hat{\sigma}_{ij} \frac{\partial W_i^l}{\partial X_j} d\Omega = \int_\Omega F_i W_i^l d\Omega + \int_{\Gamma_\sigma} T_i W_i^l d\Gamma \tag{3.25}$$

ou na forma matricial

$$\int_\Omega (\mathcal{L} \boldsymbol{W}_l)^T \hat{\boldsymbol{\sigma}} d\Omega = \int_\Omega \boldsymbol{W}_l \boldsymbol{F} d\Omega + \int_{\Gamma_\sigma} \boldsymbol{W}_l \boldsymbol{T} d\Gamma \tag{3.26}$$

Por sua vez o campo de deslocamentos tem uma função aproximante do tipo

$$\hat{\mathbf{u}} = \sum_{k=1}^{m} N_k \mathbf{u}_k \tag{3.27}$$

ou na forma matricial

$$\left\{ \begin{array}{c} \hat{u} \\ \hat{v} \\ \hat{w} \end{array} \right\} = \left[\begin{array}{ccccccc} N_1 & 0 & 0 & ... & N_m & 0 & 0 \\ 0 & N_1 & 0 & ... & 0 & N_m & 0 \\ 0 & 0 & N_1 & ... & 0 & 0 & N_m \end{array} \right] \left\{ \begin{array}{c} u_1 \\ v_1 \\ w_1 \\ . \\ . \\ . \\ u_m \\ v_m \\ w_m \end{array} \right\} \tag{3.28}$$

Por fim as deformações também provêem de uma função aproximante

$$\hat{\epsilon} = \mathcal{L} \cdot \hat{\mathbf{u}} = \mathcal{L} \cdot \mathbf{N}_k \cdot \mathbf{u}_k \tag{3.29}$$

$$N_k = \left[\begin{array}{ccc} N_k & 0 & 0 \\ 0 & N_k & 0 \\ 0 & 0 & N_k \end{array} \right] \tag{3.30}$$

$$\mathbf{u}_k = \left\{ \begin{array}{c} u_k \\ v_k \\ w_k \end{array} \right\} \tag{3.31}$$

rescrevendo a função aproximante das tensões

$$\hat{\boldsymbol{\sigma}} = \mathbf{D}\mathcal{L}\mathbf{N}_k\mathbf{u}_k \tag{3.32}$$

onde $\mathcal{L}\mathbf{N}_k = \mathbf{B}$ e permite rescrever assim a expressão anterior da seguinte forma:

$$\hat{\boldsymbol{\sigma}} = \mathbf{DBu} \tag{3.33}$$

A matriz $\mathbf{B}$ é a matriz das derivadas das funções de forma das NURBS já demostrada em 2.63, e é obtida da seguinte forma:

$$\mathbf{B}_J^i = \left[\begin{array}{ccc} \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial x} & 0 & 0 \\ 0 & \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial y} & 0 \\ 0 & 0 & \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial z} \\ \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial y} & \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial x} & 0 \\ \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial z} & 0 & \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial x} \\ 0 & \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial z} & \dfrac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial y} \end{array} \right] (J = 1 \sim 9) \tag{3.34}$$

e por sua vez os elementos que a constituem são dados por:

$$\frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial x} = \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \zeta}\frac{\partial \zeta}{\partial x} \tag{3.35}$$

$$\frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial y} = \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \zeta}\frac{\partial \zeta}{\partial y} \tag{3.36}$$

$$\frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial z} = \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \eta}\frac{\partial \eta}{\partial z} + \frac{\partial R_J(\xi_i,\eta_i,\zeta_i)}{\partial \zeta}\frac{\partial \zeta}{\partial z} \tag{3.37}$$

Recorrendo ao método de Galerkin utiliza-se como funções de peso as funções de forma, isto é, $W_l = R_l$. Desta forma é podssível rescrever a formulação fraca da seguinte forma:

$$\sum_{k=1}^{m}\left(\int_\Omega \mathbf{B}^T\mathbf{D}\mathbf{B}d\Omega\right)\widehat{\mathbf{u}}_k = \int_\Omega \mathbf{N}_l\mathbf{F}d\Omega + \int_{\Gamma_\sigma}\mathbf{N}_l\mathbf{T}d\Gamma \tag{3.38}$$

que não é mais do que um sistema na forma $\mathbf{K_{lk}}\cdot\mathbf{\hat{u}_k} = \mathbf{R_l}$, onde $\mathbf{K}$ é a matriz de rigidez do sistema que é obtida a partir da seguinte equação

$$\mathbf{K} = \sum_{k=1}^{m}\left(\int_\Omega \mathbf{B}^T\mathbf{D}\mathbf{B}d\Omega\right) \tag{3.39}$$

Este sistema é resolvido para as incógnitas $\mathbf{\hat{u}}_k$

3.4 Derivação e integração numérica na análise isogeométrica

Computacionalmente é necessário realizar operações de integração e derivação, contudo não é possível o computador realizar directamente estas operações, uma vez que não são operações aritméticas. Estas operações são realizadas apenas num referencial paramétrico denominado por referencial rst. Existem outros dois referenciais sendo um o referencial físico xyz e o referencial de índices $\xi\eta\zeta$. É possível fazer as referidas operações, recorrendo a uma aproximação através dos somatórios apresentados nas equações seguintes:

$$\frac{\partial u}{\partial x} \approx \frac{\partial \hat{u}_n}{\partial x} = \sum_{i=1}^{n}\frac{\partial N_i}{\partial x}u_i \tag{3.40}$$

$$\frac{\partial u}{\partial y} \approx \frac{\partial \hat{u}_n}{\partial y} = \sum_{i=1}^{n}\frac{\partial N_i}{\partial y}u_i \tag{3.41}$$

$$\frac{\partial u}{\partial z} \approx \frac{\partial \hat{u}_n}{\partial z} = \sum_{i=1}^{n}\frac{\partial N_i}{\partial z}u_i \tag{3.42}$$

em que

$$\frac{\partial N_i}{\partial x} = \frac{\partial N_i}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial N_i}{\partial s}\frac{\partial s}{\partial x} + \frac{\partial N_i}{\partial t}\frac{\partial t}{\partial x} \tag{3.43}$$

$$\frac{\partial N_i}{\partial y} = \frac{\partial N_i}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial N_i}{\partial s}\frac{\partial s}{\partial y} + \frac{\partial N_i}{\partial t}\frac{\partial t}{\partial y} \tag{3.44}$$

$$\frac{\partial N_i}{\partial z} = \frac{\partial N_i}{\partial r}\frac{\partial r}{\partial z} + \frac{\partial N_i}{\partial s}\frac{\partial s}{\partial z} + \frac{\partial N_i}{\partial t}\frac{\partial t}{\partial z} \tag{3.45}$$

ou escrevendo de forma mais conveniente

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial s}{\partial x} & \frac{\partial t}{\partial x} \\ \frac{\partial r}{\partial y} & \frac{\partial s}{\partial y} & \frac{\partial t}{\partial y} \\ \frac{\partial r}{\partial z} & \frac{\partial s}{\partial z} & \frac{\partial t}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial r} \\ \frac{\partial N_i}{\partial s} \\ \frac{\partial N_i}{\partial t} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial r} \\ \frac{\partial N_i}{\partial s} \\ \frac{\partial N_i}{\partial t} \end{bmatrix} \tag{3.46}$$

e a matriz $\mathbf{J}$ é dada por

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial y}{\partial r} & \frac{\partial z}{\partial r} \\ \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} & \frac{\partial z}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} & \frac{\partial z}{\partial t} \end{bmatrix} \tag{3.47}$$

por fim

$$x = \sum_{i=1}^{n} N_i x_i \tag{3.48}$$

$$y = \sum_{i=1}^{n} N_i y_i \tag{3.49}$$

$$z = \sum_{i=1}^{n} N_i z_i \tag{3.50}$$

Rescrevendo 3.47 como sendo um somatório das várias funções

$$\mathbf{J} = \sum_{i=1}^{n} \begin{bmatrix} \frac{\partial N_i}{\partial r} x_i & \frac{\partial N_i}{\partial r} y_i & \frac{\partial N_i}{\partial r} z_i \\ \frac{\partial N_i}{\partial s} x_i & \frac{\partial N_i}{\partial s} y_i & \frac{\partial N_i}{\partial s} z_i \\ \frac{\partial N_i}{\partial t} x_i & \frac{\partial N_i}{\partial t} y_i & \frac{\partial N_i}{\partial t} z_i \end{bmatrix} \tag{3.51}$$

Os valores de r, s, t variam sempre entre -1 e 1.

$$r, s, t \in [-1; 1]$$

Quanto à integração numérica a mesma é feita num referencial paramétrico r, s, t segundo pontos conhecidos e denominados pontos de Gauss.

$$\int_{\Omega^e} f(x,y,z)dxdydz = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(r,s,t)\det\mathbf{J}dtdsdr \tag{3.52}$$

Os referenciais mencionados podem ser vistos no exemplo da figura 3.1.

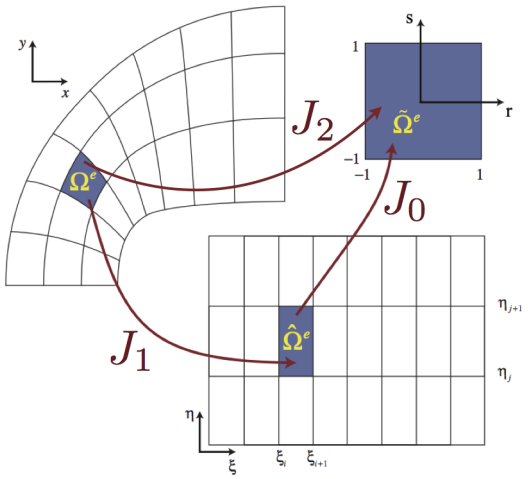


Figura 3.1: Diferentes referenciais existentes [2]

3.5 Mudanças de referenciais

A mudança de referenciais é feita recorrendo às derivações em cadeia como veremos a seguir. Para passar do espaço paramétrico para o espaço dos indíces recorremos às seguintes derivações

$$\frac{\partial}{\partial r} = \frac{\partial}{\partial \xi} \cdot \frac{\partial \xi}{\partial r} + \frac{\partial}{\partial \eta} \cdot \frac{\partial \eta}{\partial r} + \frac{\partial}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial r} \tag{3.53}$$

$$\frac{\partial}{\partial s} = \frac{\partial}{\partial \xi} \cdot \frac{\partial \xi}{\partial s} + \frac{\partial}{\partial \eta} \cdot \frac{\partial \eta}{\partial s} + \frac{\partial}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial s} \tag{3.54}$$

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial \xi} \cdot \frac{\partial \xi}{\partial t} + \frac{\partial}{\partial \eta} \cdot \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial t} \tag{3.55}$$

Estas derivações formam a matriz $\mathbf{J}_0$ dada por:

$$\boldsymbol{J}_0 = \begin{bmatrix} \frac{\partial \xi}{\partial r} & \frac{\partial \eta}{\partial r} & \frac{\partial \zeta}{\partial r} \\ \frac{\partial \xi}{\partial s} & \frac{\partial \eta}{\partial s} & \frac{\partial \zeta}{\partial s} \\ \frac{\partial \xi}{\partial t} & \frac{\partial \eta}{\partial t} & \frac{\partial \zeta}{\partial t} \end{bmatrix} \tag{3.56}$$

De forma análoga, procede-se à passagem entre o referencial físico e o referencial paramétrico, formando a matriz $\mathbf{J}_1$ que realiza essa mesma passagem, com as seguintes derivadas parciais:

$$\frac{\partial \boldsymbol{x}}{\partial \xi} = \sum_{I=0}^{n} \sum_{J=0}^{m} \sum_{K=0}^{l} \frac{\partial R_{IJK}(\xi, \eta, \zeta)}{\partial \xi} \boldsymbol{x}_{IJK} \tag{3.57}$$

$$\frac{\partial \boldsymbol{x}}{\partial \eta} = \sum_{I=0}^{n} \sum_{J=0}^{m} \sum_{K=0}^{l} \frac{\partial R_{IJK}(\xi, \eta, \zeta)}{\partial \eta} \boldsymbol{x}_{IJK} \tag{3.58}$$

$$\frac{\partial \boldsymbol{x}}{\partial \zeta} = \sum_{I=0}^{n} \sum_{J=0}^{m} \sum_{K=0}^{l} \frac{\partial R_{IJK}(\xi, \eta, \zeta)}{\partial \zeta} \boldsymbol{x}_{IJK} \tag{3.59}$$

$$\boldsymbol{J}_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \tag{3.60}$$

$$\begin{Bmatrix} \dfrac{\partial}{\partial \xi} \\ \dfrac{\partial}{\partial \eta} \\ \dfrac{\partial}{\partial \zeta} \end{Bmatrix} = \mathbf{J}_1 \begin{Bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{Bmatrix} \tag{3.61}$$

Recorrendo à regra de derivação em cadeia é possível passar do referencial físico XYZ para o referencial paramétrico rst.

$$\frac{\partial \boldsymbol{x}}{\partial r} = \frac{\partial x}{\partial \xi} \cdot \frac{\partial \xi}{\partial r} + \frac{\partial x}{\partial \eta} \cdot \frac{\partial \eta}{\partial r} + \frac{\partial x}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial r} \tag{3.62}$$

$$\frac{\partial \boldsymbol{x}}{\partial s} = \frac{\partial x}{\partial \xi} \cdot \frac{\partial \xi}{\partial s} + \frac{\partial x}{\partial \eta} \cdot \frac{\partial \eta}{\partial s} + \frac{\partial x}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial s} \tag{3.63}$$

$$\frac{\partial \boldsymbol{x}}{\partial t} = \frac{\partial x}{\partial \xi} \cdot \frac{\partial \xi}{\partial t} + \frac{\partial x}{\partial \eta} \cdot \frac{\partial \eta}{\partial t} + \frac{\partial x}{\partial \zeta} \cdot \frac{\partial \zeta}{\partial t} \tag{3.64}$$

Combinando estas duas matrizes, é possível a passagem de um referencial físico para o espaço de índices, sendo para isso necessário multiplicar as duas matrizes $\mathbf{J}_0$ e $\mathbf{J}_1$ obtendo como resultado uma matriz $\mathbf{J}_2$ que permite a passagem do espaço físico para o espaço dos índices de forma directa:

$$\begin{aligned} \boldsymbol{J}_2 &= \boldsymbol{J}_0 \cdot \boldsymbol{J}_1 \\ &= \begin{bmatrix} \dfrac{\partial \xi}{\partial r}\dfrac{\partial x}{\partial \xi} + \dfrac{\partial \eta}{\partial r}\dfrac{\partial x}{\partial \eta} + \dfrac{\partial \zeta}{\partial r}\dfrac{\partial x}{\partial \zeta} & \dfrac{\partial \xi}{\partial r}\dfrac{\partial y}{\partial \xi} + \dfrac{\partial \eta}{\partial r}\dfrac{\partial y}{\partial \eta} + \dfrac{\partial \zeta}{\partial r}\dfrac{\partial y}{\partial \zeta} & \dfrac{\partial \xi}{\partial r}\dfrac{\partial z}{\partial \xi} + \dfrac{\partial \eta}{\partial r}\dfrac{\partial z}{\partial \eta} + \dfrac{\partial \zeta}{\partial r}\dfrac{\partial z}{\partial \zeta} \\ \dfrac{\partial \xi}{\partial s}\dfrac{\partial x}{\partial \xi} + \dfrac{\partial \eta}{\partial s}\dfrac{\partial x}{\partial \eta} + \dfrac{\partial \zeta}{\partial s}\dfrac{\partial x}{\partial \zeta} & \dfrac{\partial \xi}{\partial s}\dfrac{\partial y}{\partial \xi} + \dfrac{\partial \eta}{\partial s}\dfrac{\partial y}{\partial \eta} + \dfrac{\partial \zeta}{\partial s}\dfrac{\partial y}{\partial \zeta} & \dfrac{\partial \xi}{\partial s}\dfrac{\partial z}{\partial \xi} + \dfrac{\partial \eta}{\partial s}\dfrac{\partial z}{\partial \eta} + \dfrac{\partial \zeta}{\partial s}\dfrac{\partial z}{\partial \zeta} \\ \dfrac{\partial \xi}{\partial t}\dfrac{\partial x}{\partial \xi} + \dfrac{\partial \eta}{\partial t}\dfrac{\partial x}{\partial \eta} + \dfrac{\partial \zeta}{\partial t}\dfrac{\partial x}{\partial \zeta} & \dfrac{\partial \xi}{\partial t}\dfrac{\partial y}{\partial \xi} + \dfrac{\partial \eta}{\partial t}\dfrac{\partial y}{\partial \eta} + \dfrac{\partial \zeta}{\partial t}\dfrac{\partial y}{\partial \zeta} & \dfrac{\partial \xi}{\partial t}\dfrac{\partial z}{\partial \xi} + \dfrac{\partial \eta}{\partial t}\dfrac{\partial z}{\partial \eta} + \dfrac{\partial \zeta}{\partial t}\dfrac{\partial z}{\partial \zeta} \end{bmatrix} \end{aligned} \tag{3.65}$$

Esta passagem do espaço físico para o espaço de índices é fundamental uma vez que é neste espaço que é feita a integração de Gauss.

$$\begin{Bmatrix} \dfrac{\partial}{\partial r} \\ \dfrac{\partial}{\partial s} \\ \dfrac{\partial}{\partial t} \end{Bmatrix} = \mathbf{J}_2 \begin{Bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{Bmatrix} \tag{3.66}$$

Capítulo 4

Programação Orientada a Objectos

O *software* no qual assenta este trabalho é um *software* desenvolvido com o propósito de servir de base à investigação, nomeadamente à geração e manipulação de malhas (pré-processamento) e visualização de resultados (pós-processamento) e geração de imagens vectoriais dos mesmos. Até ao início deste trabalho este *software* apenas contemplava os MEF e era denominado CerebroPre (figura 4.1), havendo portanto a necessidade de tornar o mesmo mais versátil e intuitivo para receber a modelação gráfica de NURBS. Como essa mudança tornou o programa mais dinâmico o como tal o seu nome foi também mudado para *Numerical Dynamics* (figura 4.2). É desta forma que este *software* é referido em todo o trabalho.

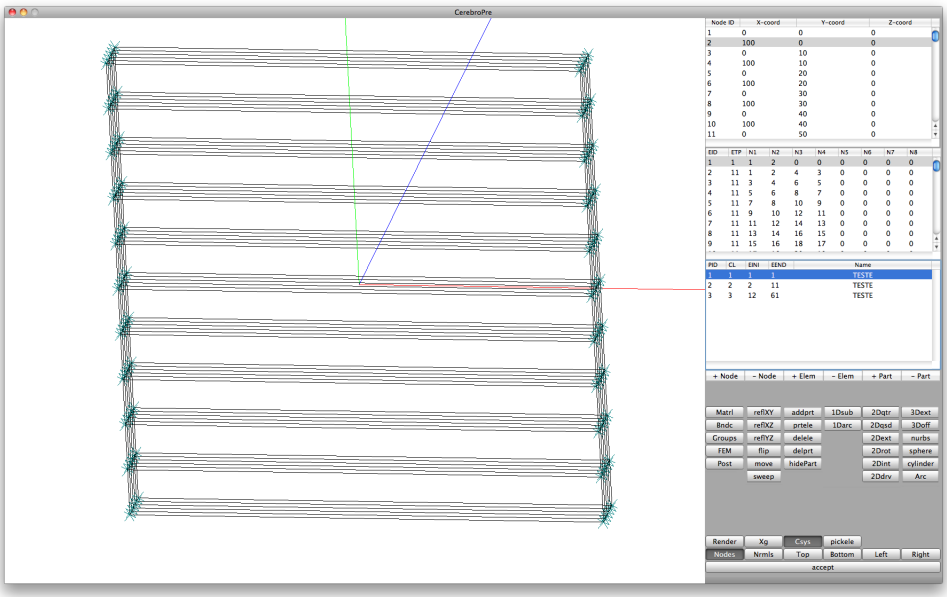


Figura 4.1: Janela principal do *software CerebroPre*

O *software* é unicamente compatível com o sistema operativo *Apple OS X* e tal facto deve-se à estabilidade, performance, e facilidade com que se consegue desenvolver e fazer modificações na estrutura do *software*. Uma vez que o código é escrito em *Objective-C* a sua portabilidade para outros sistemas operativos é possível mas fica fora do âmbito deste

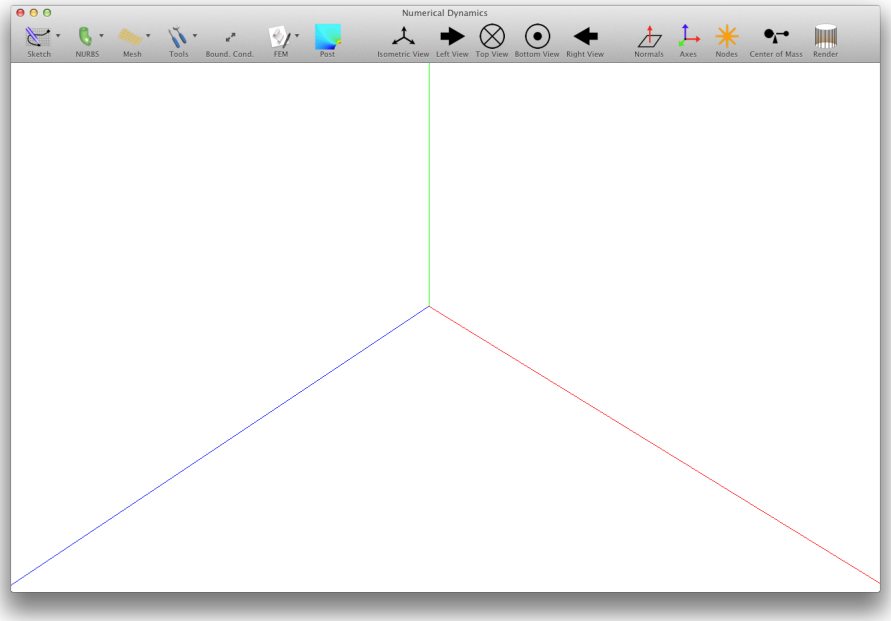


Figura 4.2: Janela principal do *software Numerical Dynamics*

trabalho, pelo que não será mencionado nenhum aspecto relativamente a essa mesma portabilidade, nem compatibilidade de código com outros sistemas operativos.

Para desenvolver *software* nativo para *Mac OS X* recorremos ao ambiente de desenvolvimento *Xcode* cujas bibliotecas de funções disponíveis, estão escritas na linguagem *Objective-C* fazendo com que a implementação seja mais simples dado o suporte existente tanto no próprio *software* como *online*.

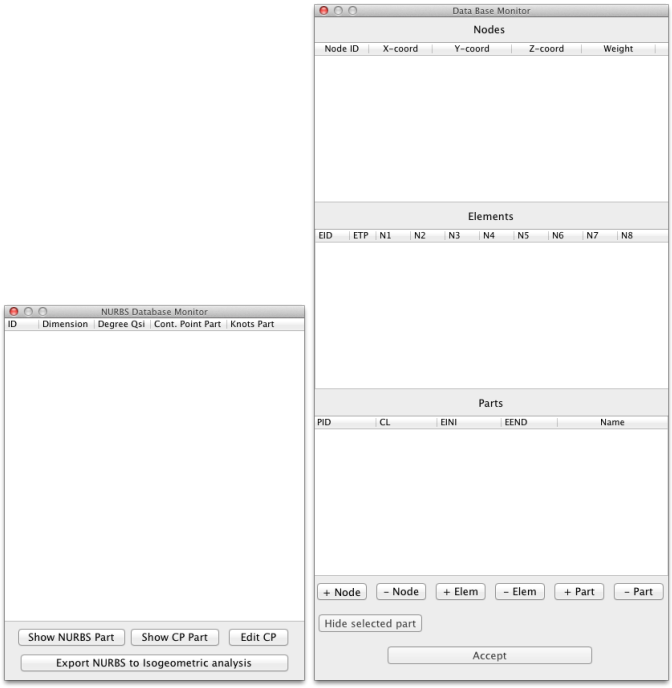
4.1 *Objective-C*

A principal razão que levou a optar por esta linguagem de programação para desenvolver o *software* foi em primeiro lugar ser uma linguagem orientada a objectos, o que a torna muito versátil. O potencial que se encontra nas bibliotecas de funções só estará disponível com o uso das técnicas desta mesma linguagem.

O segundo motivo que levou a esta escolha, foi o facto desta linguagem ser uma extensão do padrão ANSI C [8], ou seja os programas existentes C podem ser adaptados para usar estas estruturas de *software* sem perder o trabalho que foi feito no seu desenvolvimento. Porque o *Objective-C* incorpora C, existem todos os benefícios do C quando se programa com *Objective-C*. Desta forma, é sempre possível sempre escolher o método que se pretende seguir, tomando como exemplo a criação de uma classe numa vertente orientada a objectos, ou outras técnicas de programação, como estruturas de dados e funções.

Além disso, o *Objective-C* é uma linguagem muito simples, sendo a sua sintaxe curta, inequívoca e fácil de aprender. A curva de de aprendizagem desta linguagem é bastante acentuada o que torna qualquer pessoa familiar com a linguagem em pouco tempo. [9]

Em *Objective-C* uma mensagem tem a seguinte estrutura:



(a) (b)

Figura 4.3: Janelas de monitorização e manipulação das bases de dados

```
[objectoNaMemoria metodoAExecutar:argumentoAPassarAoMetodo];
```

O objecto previamente existente na memória com o nome "objectoNaMemoria"irá receber o comando tomado como exemplo, e irá executar o método com o nome "metodoAExecutar". Neste exemplo estamos a invocar um método que requer apenas um argumento, e esse argumento foi indicado como sendo "argumentoAPassarAoMetodo".

Caso o método devolva algum resultado o mesmo é atribuído a uma variável. Se for conveniente determinar o numero de elementos num vector, é possível faze-lo da seguinte forma:

```
numeroDeElementos = [arrayDeElementos count];
```

Uma boa prática de programação orientada a objecto é o uso do paradigma Modelo Vista Controlador – MVC. Este paradigma refere que o código deve ser separado em três categorias e tendo cada uma delas um papel bem definido.

O Modelo como o próprio nome indica define um padrão de informação, para quando se cria uma instância desse objecto modelo, se possa criar nova informação e guardar respectivamente essa informação.

A Vista destina-se a mostrar a informação ao utilizador, e também perceber quais as ordens deste.

Já o controlador situa-se no meio de ambos fazendo um papel de interpretador. É este que determina como o Modelo vai ser representado na Vista e também que informação pedir ao modelo e coloca-la na vista.

A Vista e o Modelo nunca falam diretamente. [10]

Comparado a outras linguagens orientadas a objectos baseadas em C, o *Objective-C* é muito dinâmico. O compilador preserva uma grande quantidade de informações sobre os próprios objectos para uso durante a execução. Decisões que de outra forma poderiam ser feitas durante a compilação pode ser adiadas até que o programa esteja em execução. Este dinamismo dá aos programas desenvolvidos na linguagem *Objective-C* uma flexibilidade enorme.

Com o *Objective-C* conseguimos dois grandes benefícios que são difíceis de obter com outras linguagens orientadas as objectos, sendo elas as *bindings* e as ferramentas de monitorização e optimização.

Objective-C suporta um estilo aberto de ligação dinâmica, um estilo que pode incorporar uma arquitectura simples para interfaces interactivas. As mensagens não são, necessariamente, transportadas por qualquer classe do receptor ou mesmo desencadeadas por um método, permitindo assim que através da interface de utilização haja uma comunicação entre objectos. [11]

Este dinamismo permite a construção de ferramentas de desenvolvimento de sofisticadas. Uma interface para com o sistema operativo e consequentemente com o *software* em execução fornece acesso a informações sobre o funcionamento das aplicações, e deste modo é possível intervir a fim de optimizar e encontrar erros ou imperfeições no código (*bugs*).[9]

4.2 Interfaces gráficas

Referido anteriormente na tabela 1.1 a primeira etapa para uma simulação numérica é a criação de um modelo sólido.

É aqui que assume importância uma boa interface. O tempo despendido na utilização de uma interface não intuitiva e dinâmica, pode parecer irrisório quando se tem em conta apenas uma simulação, mas para optimizar todo um processo é necessário levar em conta todas as etapas e todos os detalhes.

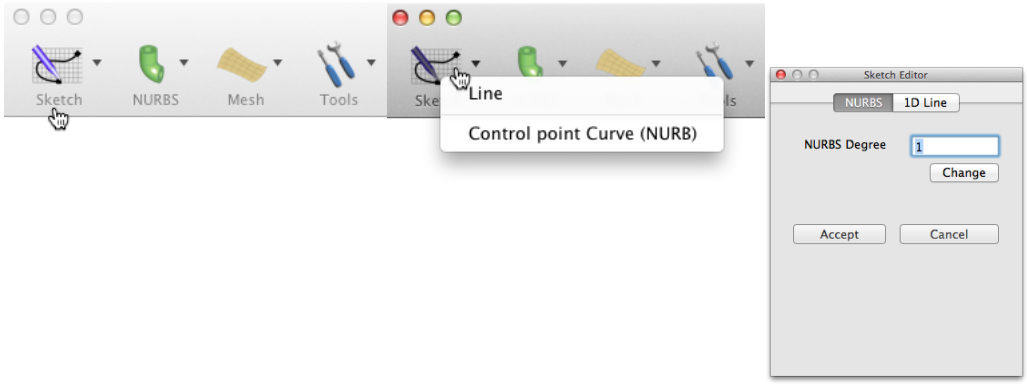
Uma interface "limpa"e simples ajudará o utilizador a perceber mais facilmente os passos que terá de seguir, evitando assim uma exploração ou navegação por longos menus que fará com que o tempo gasto nesta tarefa seja maior. Aliando também um conceito de inteligência, que faz surgir os comandos e as caixas de diálogo na altura exacta, consegue-se também tornar a modelação bastante mais célere.

Outro aspecto deveras importante é o *feedback* que a interface promove ao utilizador, sendo que neste caso foi opção utilizar a cor como mecanismo de realce e resposta à interacção do utilizador.

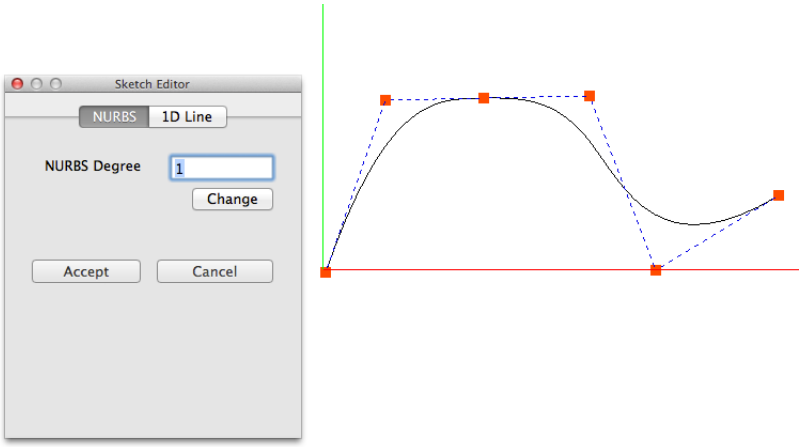
Por fim os atalhos de teclado intuitivos conferem à interface uma utilização a duas mãos, fazendo com que com menos cliques do rato se consiga activar funções, ou mesmo manipular o modelo.

De seguida irão ser sumariadas as funcionalidades ao dispor no *software Numerical Dynamics* para que se consigam obter as malhas pretendidas para análise.

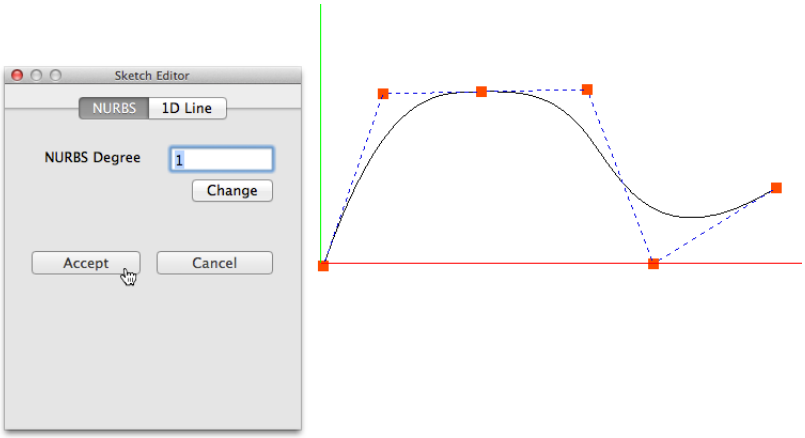
Tal como noutros sistemas de DAC os modelos 3D começam por uma base 2D e essa base teve origem 1D. Comecemos então por desenhar uma entidade 1D:



(a) Escolha do menu "Sketch" (b) Curva NURB por definição dos pontos de controlo (c) Escolha do grau da curva



(d) Desenho do polígono de controlo



(e) Clique em "Accept"para criar a curva

Figura 4.4: Passos para criar uma curva NURBS com base na definição dos pontos de controlo

Como a curva desenhada poderá necessitar de ajustes, é possível editar a curva até obter o resultado pretendido. Activando a janela "NURBS Database Monitor"4.5 através do menu "View"surge o seguinte menu:

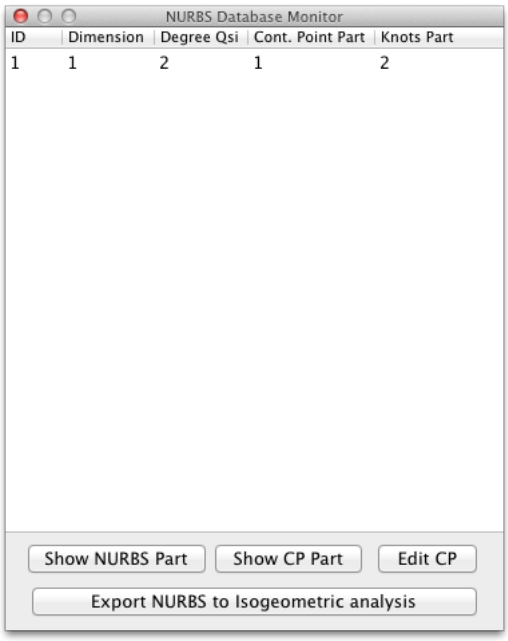
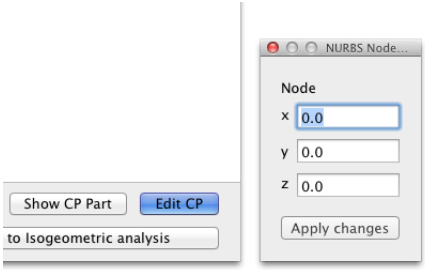
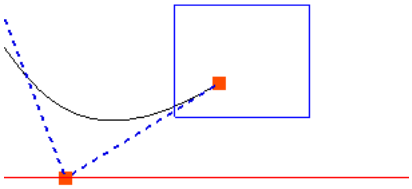


Figura 4.5: NURBS Database Monitor

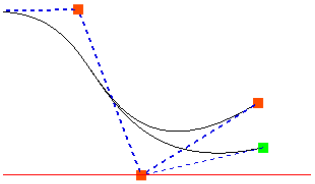
Seleccionando a curva pretendida e clicando em "Show CP Part"é mostrado o polígono de controlo realçado, e de seguida é possível activar o modo de edição clicando em "Edit CP". surge por fim uma janela que monitoriza as coordenadas no ponto que iremos editar. Os passos para editar um ponto de controlo estão descritos na figura 4.6.



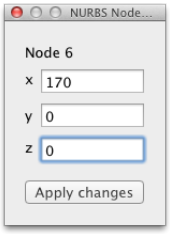
(a) Activar modo de edição



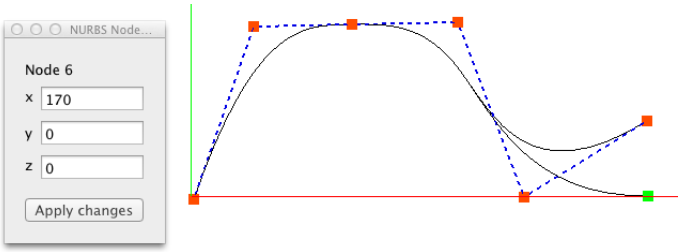
(b) Selecção do ponto



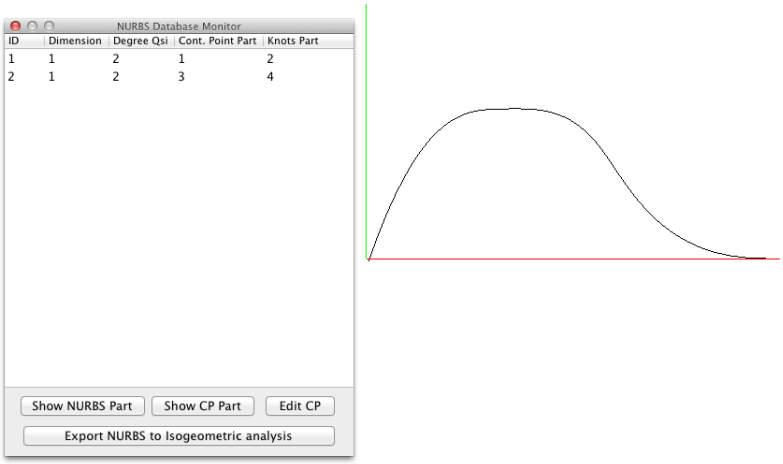
(c) Identificação do ponto



(d) Ajuste das coordenadas manualmente



(e) Aplicar as alterações feitas manualmente



(f) Sair do modo de edição e gravar as alterações

Figura 4.6: Passos para editar uma curva NURBS

Depois de concluído o ajuste do ponto pretendido, caso não seja do interesse ajustar mais nenhum outro ponto desta curva, estamos perante as condições necessárias à obtenção de um modelo 2D através das opções disponíveis e listadas no menu "Mesh"que vemos na figura 4.7:

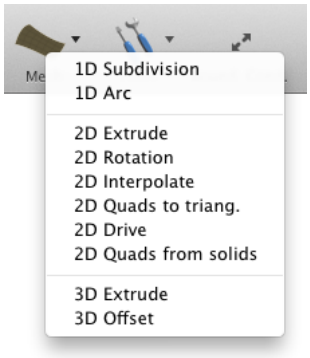


Figura 4.7: Opções para obter um modelo a 2D

Para o presente exemplo foi escolhida a opção "Extrude". É possível selecionar a curva através do botão "Select Part". Depois da mesma estar seleccionada, fica realçada na cor amarela. Quando o utilizador tiver finalmente escolhido a curva pretendida clica em "Upload Sel"para validar a escolha. Por fim completa-se a restante informação, fazendo a seguinte leitura da mesma: dado um vector que parte do plano de origem da curva, até ao valor especificado, pretende-se criar N divisões igualmente espaçadas.

No caso do exemplo demonstrado passo a passo na figura 4.9 irão ser criadas 10 divisões num vector que parte do plano $z = 0$ até $z = 100$. O resultado desta operação pode ser visto na figura 4.8

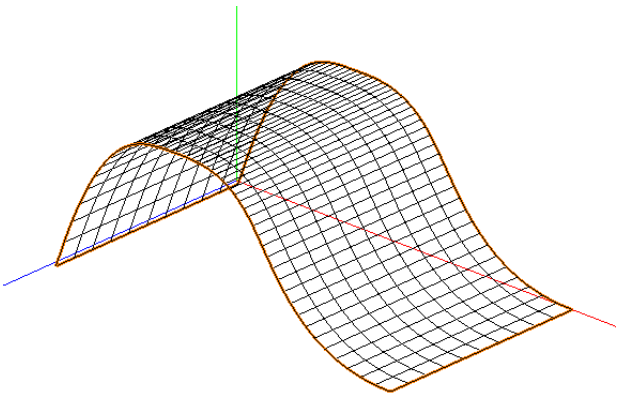
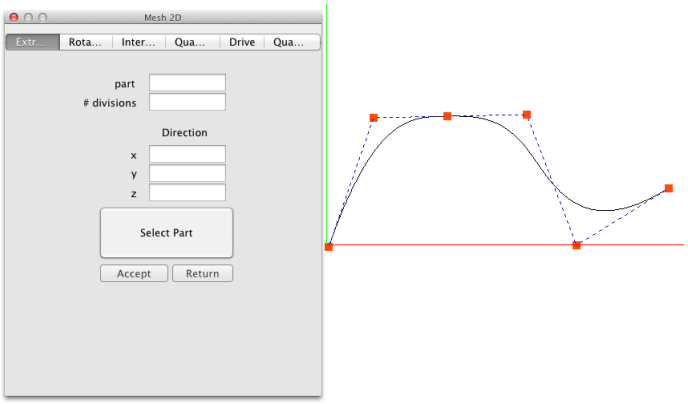
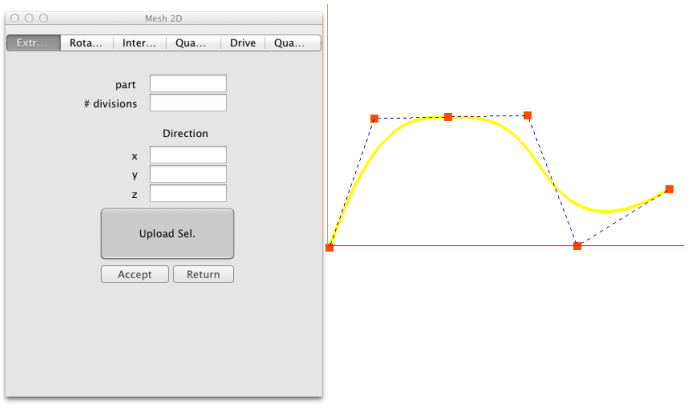


Figura 4.8: Resultado de uma extrusão para duas dimensões

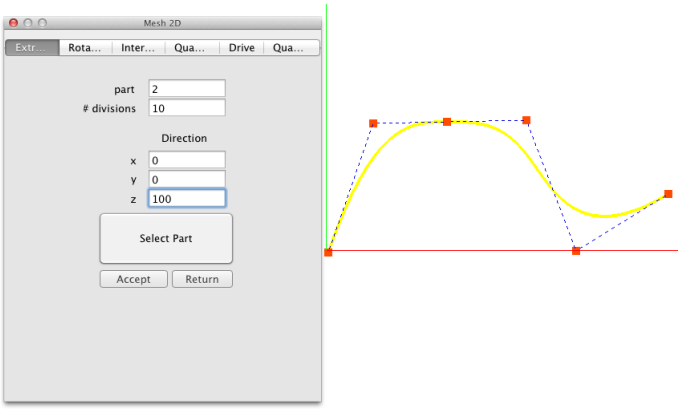
No final desta operação resultaram 5 entidades na base de dados como vemos na figura 4.10



(a) Menu das opções "Extrude"



(b) Selecção da "part"pretendida



(c) Preenchimento das opções e por fim "Accept"

Figura 4.9: Passos para criar um modelo 2D

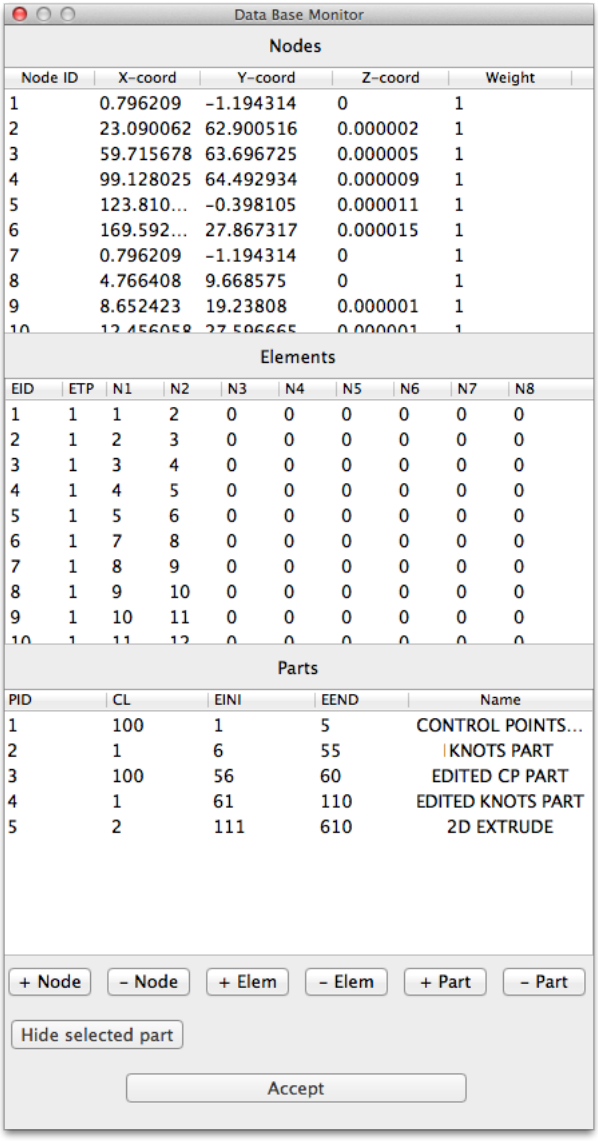


Figura 4.10: Identificação de todos os elementos desenhados

Por fim e para obter uma entidade com três dimensões, voltamos ao menu "Mesh"e escolhemos a opção "3D Offset"4.11.

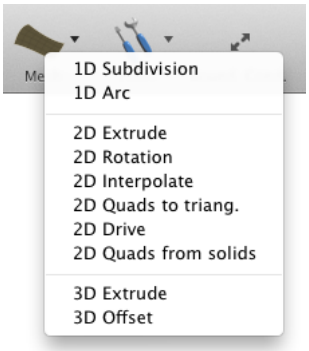
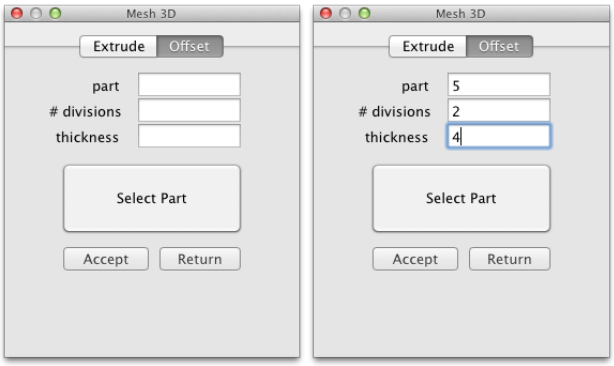


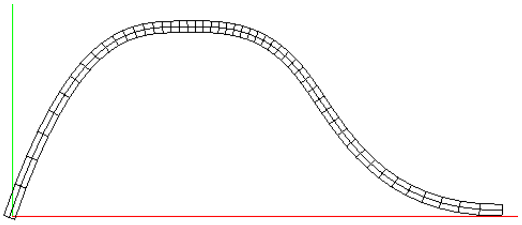
Figura 4.11: Passagem de 2D para 3D

Com esta opção teremos apenas que especificar qual a entidade à qual pretendemos aplicar a transformação, o número de divisões e a espessura total final. Podemos utilizar a janela da base de dados como vemos representado em parte na figura 4.10 e que está acessível também através do menu "View"para desta forma preencher os campos de entrada, sem termos que seleccionar através do clique a geometria pretendida. Por vezes pode-se tornar mais rápido o uso desta mesma metodologia.

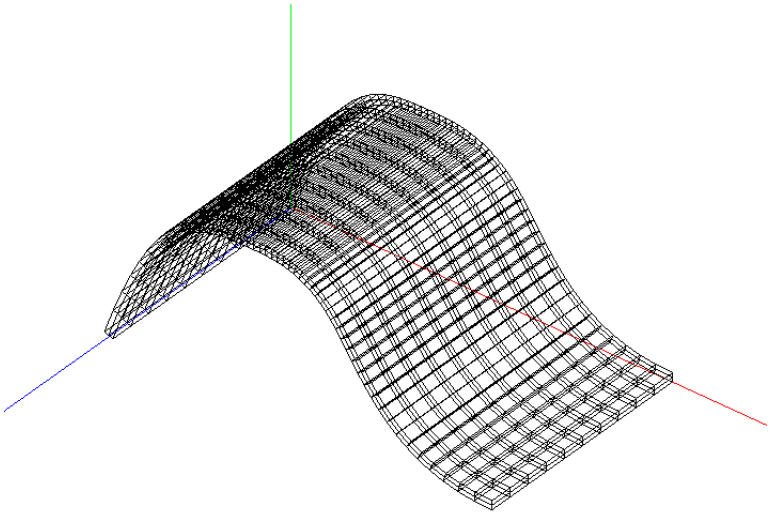


(a) (b)

Figura 4.12: Criação de uma entidade 3D



(a)



(b)

Figura 4.13: Resultado do "3D Offset"

4.2.1 Ferramentas de modelação disponíveis no *software Numerical Dynamics*

Além das ferramentas utilizadas para demonstrar a interface do *software* utilizado, existem outras ferramentas/opções que são de todo fundamentais para a modelação de problemas.

Estas ferramentas ainda se encontram num estado elementar, pois o seu propósito é apenas testar a implementação das respectivas funcionalidades, e apresentam uma grande margem de evolução.

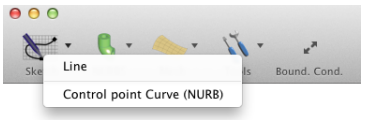


Figura 4.14: Ferramentas de *Sketch*

Na figura 4.14 podemos ver as opções que dispomos para criar uma entidade de forma gráfica, isto é, através do uso do rato. Se escolhermos a opção "1D Line", surge-nos a janela que vemos na figura 4.15. Com essa opção, é definida uma linha no plano normal à vista actual, e nesse plano é possível especificar as coordenadas x e y dos pontos extremos da linha. Quando as coordenadas dos pontos são introduzidas manualmente e depois de já ter sido desenhada a linha, é possível obter uma pré-visualização do resultado final, clicando no botão existente para esse mesmo fim. É colocado ao dispor do utilizador a possibilidade de especificar o número de divisões que essa linha terá, o que posteriormente dará origem a elementos sobre essa mesma linha desenhada.

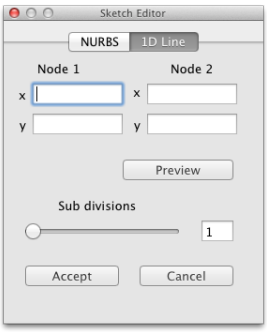


Figura 4.15: Opções para desenho de uma linha a uma dimensão

Outra possibilidade que o utilizador tem ao dispor para desenhar uma entidade a uma dimensão, é através do menu "Control Point Curve (NURBS)"que mostra a janela que vemos na figura 4.16. Essa funcionalidade é simples e intuitiva, bastando clicar onde se pretende criar pontos de controlo e de imediato surge a curva que esses mesmos pontos dão origem. É possível ao utilizador visualizar e alterar o grau da curva até esta estar terminada. Para terminar a curva basta clicar com o botão direito do rato em qualquer parte da janela.

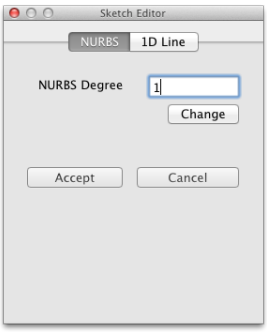


Figura 4.16: Opções para desenho de uma curva NURBS através de pontos de controlo

Para criar geometrias provenientes de polígonos de controlo ou superfícies de controlo, utilizamos as funções disponíveis no menu "NURBS". Essas opões estão listadas na figura 4.17 e destacam-se as seguintes: *Sphere*, *Cylinder*, *Arc*, *Fit curve*, *Fit surface* e *Fit solid*, sendo que as restantes são irrelevantes para o presente trabalho.

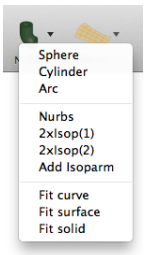


Figura 4.17: Ferramentas para criar geometrias com base em NURBS

Escolhendo a opção da esfera, de imediato surge no ecrã a janela que vemos na figura 4.18 e preenchendo as caixas de texto com os parâmetros de entrada desta função, é criada uma esfera com centro nas coordenadas especificadas xyz no referencial global. Os parâmetros de "Tesselation"irão definir as divisões (e por consequência o refinamento) que a forma final terá. Quanto mais elevados forem estes parâmetros mais o objecto se aproximará de uma esfera. Porém mais pesará no cálculo computacional se o refinamento for excessivo.

Figura 4.18: Opções para criar uma esfera através de NURBS

Se o utilizador pretender criar um cilindro, deve optar pela segunda opção do menu. À semelhança da esfera surge uma janela com os dados dos de entrada para a criação de um cilindro. Com a janela que surge e que está representada na figura 4.19, é criado um cilindro no referencial global, cujas bases terão o seu centro nos pontos xyz especificados e os raios poderão ser diferentes. Esta flexibilidade trás a vantagem de se poder também criar objectos cónicos. Nos ajustes da superfície, é possível definir as divisões dos polígonos de controlo, segundo as direcções paramétricas ξ e η, sendo ξ a direcção longitudinal e η a radial.

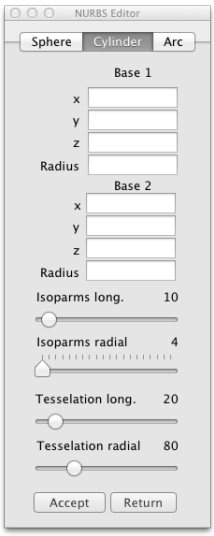


Figura 4.19: Opções para criar um cilindro através de NURBS

Também um arco de circunferência pode ser criado através deste menu. Seleccionando a respectiva opção, indicamos as coordenadas xyz do seu centro, ponto inicial e final, tudo isto no referencial global. Como já explicado atrás, é permitido ajustar o refinamento tanto do polígono de controlo, como da curva que irá ser criada. Estas opções estão visíveis na janela que está representada na figura 4.20.

Figura 4.20: Opções para criar um arco através de NURBS

A função de ajuste de uma curva a um conjunto de pontos previamente definidos é das mais importantes deste trabalho. Com esta função é necessário escolher a "Part"existente na base de dados (visível através da janela "Data Base Monitor") e introduzir em "Part to fit", escolhendo de seguida o grau da curva que se pretende, e introduzir esse valor em "Degree"e por fim o número de divisões (refinamento) que a curva terá e esse valor será introduzido em "Tesselation".

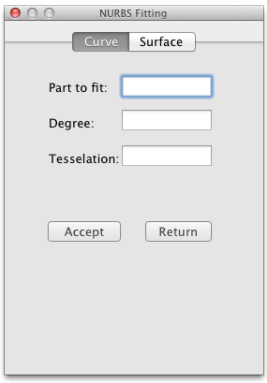


Figura 4.21: Opções para criar uma curva através de ajuste a pontos já definidos

À semelhança do ajuste de curvas a função que será apresentada de seguida é bastante importante para este trabalho. A referida função é precisamente o ajuste de uma superfície NURBS a um conjunto de pontos já existente, sendo ele proveniente de qualquer tipo de função utilizada. De seguida indicamos qual o nó da entidade que será o nó inicial para a interpolação e os graus que pretendemos para a superfície segundo as direcções ξ e η. Por fim, e também nesta funcionalidade podemos definir o refinamento da superfície com os parâmetros "Tesselation"para ambas as direcções.

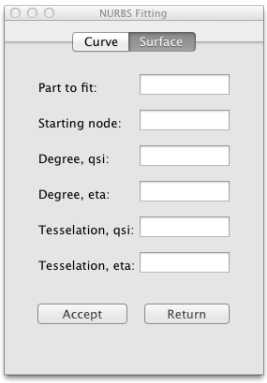


Figura 4.22: Opções para criar uma superfície através de ajuste a pontos já definidos

A função que é fundamental a este trabalho, é sem dúvida a função que se segue, cuja finalidade é o ajuste de um sólido a um conjunto de fronteiras criadas sob a forma de uma entidade do tipo 3D. Com este ajuste é possível converter toda e qualquer geometria modelada em NURBS e dessa forma termos os dados necessário à análise isogeométrica. Usando as referências da base de dados dos objectos já modelados e introduzimos a entidade ("Part") que contém os pontos que queremos que pertençam à superfície NURBS.

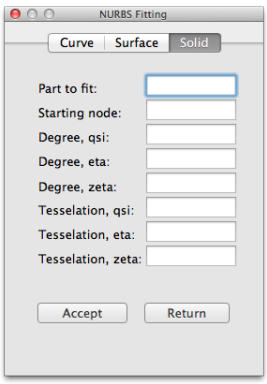


Figura 4.23: Opções para criar um sólido através de ajuste superfícies já definidas

Com pelo menos uma entidade do tipo 1D criada, é possível continuar a trabalhar essa entidade segundo novas direcções. Para tal recorremos ao menu "Mesh"que disponibiliza ao utilizador as opções *1D Subdivision*, *1D Arc*, *2D Extrude*, *2D Rotation*, *2D Interpolate*, *2D Quadrilaterals to triangles*, *2D Drive*, *3D Extrude*, *3D Offset*, que vemos na figura 4.24. A funcionalidade *2D Quadrilaterals from solids* não é relevante para o presente trabalho, e não será detalhada.

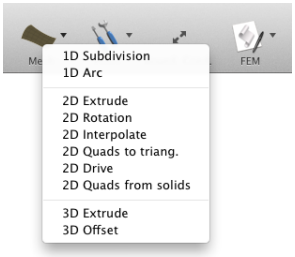


Figura 4.24: Ferramentas para criar geometrias a partir de outras previamente criadas

Se já existir uma entidade 1D e se o utilizador pretender subdividir pode utilizar a opção "1D Subdivision"cuja janela de opções está apresentada na figura 4.25. Procedendo da mesma forma indicada atrás para a escolha da entidade a subdividir, indicamos essa mesma entidade, e o número de divisões em que pretende subdividir.

Figura 4.25: Divisão de uma entidade 1D já existente

É possível aproximar um arco de circunferência através da criação de vários segmentos de recta. Esta opção está disponível no menu "1D Arc". Indicando o número de divisões que melhor aproximam o arco de circunferência, os números dos nós inicial e final entre os quais a interpolação será feita, e também o centro do arco. As entradas para as coordenadas xyz do centro do arco serão no espaço do referencial global.

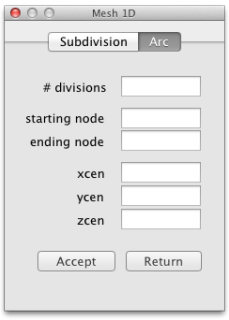


Figura 4.26: Opções para criar um arco a partir de uma entidade 1D já existente

Uma técnica bastante comum para criar formas em duas dimensões é a revolução de um perfil segundo um eixo de revolução. No *software* utilizado também é possível utilizar esta técnica. Acedendo ao menu "2D Revolution"surge uma janela onde se especifica a entidade que contém o perfil do objecto resultante, o número de divisões que irá aproximar esta forma, à semelhança do que já foi explicado no caso do arco de circunferência, e também o angulo de revolução. Para indicar qual o vector da revolução indicamos a sua origem e as suas coordenadas xyz no referencial global. É visível na figura 4.27 o que foi atrás mencionado.

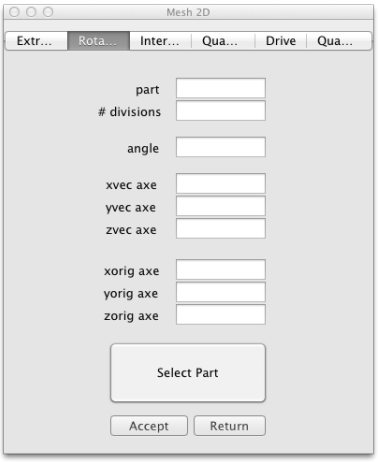


Figura 4.27: Opções para criar uma superfície de revolução

A funcionalidade denominada "2D Interpolation"utiliza quatro entidades para criar uma malha completa entre as referidas entidades. Basta para isso que se indique quais as entidades, pertencentes à base de dados, que queremos para cada uma das curvas. A janela que surge para executar esta função está na figura 4.28.

Figura 4.28: Opções para criar uma superfície através da interpolação de 4 entidades

Se for conveniente utilizar uma malha de elementos triangulares, é possível converter elementos de quatro nós em elementos de três nós e obter dessa forma uma malha de elementos triangulares. Na figura 4.29 vemos que basta especificar a entidade que pretendemos converter e essa mesma conversão é realizada dividindo automaticamente os elementos pelos nós diagonalmente opostos.

Figura 4.29: Opções para criar uma malha de triângulos a partir de uma malha de quadriláteros

Existem objectos reais que possuem contornos ou elementos que derivam de um perfil que é constante e é "conduzido"segundo ou outro perfil, seja ele rectilíneo, ou com contornos. Para modelar essas mesmas situações, existe a opção "2D Drive"visível na figura 4.30 que permite indicar quais as entidades que desempenham o papel de perfil e de contorno. Mais umas vez recorre-se à base de dados para indicar quais as entidades que irão ser usadas.

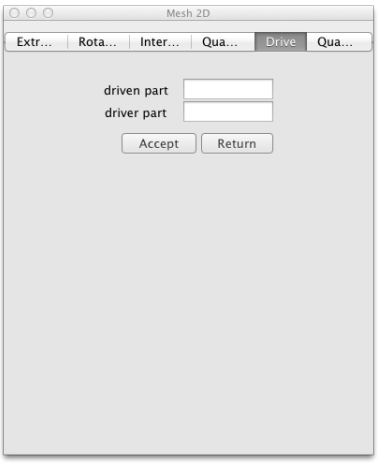


Figura 4.30: Opções para criar uma superfície através da propagação de uma entidade sobre um perfil

Analogamente ao que foi referido no exemplo da secção 4.2 sobre a extrusão para duas dimensões, também é possível partir de uma entidade a duas dimensões e obter uma a três dimensões por extrusão. Na figura 4.31 é visível que para tal basta especificar qual a entidade da base de dados que queremos extrudir, o número de divisões que pretendidas num intervalo que é definido pelo vector de extrusão. Esse mesmo vector será definido no referencial global.

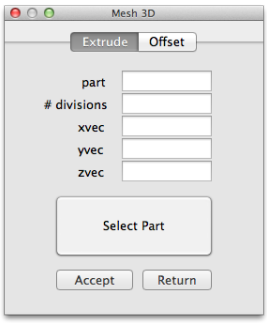


Figura 4.31: Opções para criar uma entidade a três dimensões através de extrusão

Apresentadas estas ferramentas de modelação directa, foram também desenvolvidas outras potencialidades que permitem de certa forma manipular os objectos já desenhados.

Essas funcionalidades estão agrupadas no menu "Tools"que se vê na figura 4.32. As funcionalidades *Join Parts*, *Delete Part*, *Sweep nodes*, *Reflect ...*, *Move mesh* serão detalhadas de seguida. A funcionalidade *Flip* não será detalhada uma vez que não tem especial relevância para o presente trabalho.

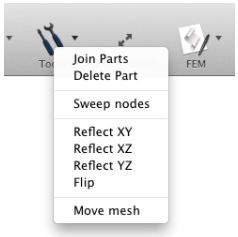


Figura 4.32: Ferramentas de auxílio à modelação

Para se agrupar duas entidades numa só, e ser possível obter resultados mais rápidos em determinadas operações, recorre-se ao menu "Join Parts"que faz surgir no ecrã a janela da figura 4.33. Esta ferramenta é de utilização simples bastando apenas indicar quais as entidades que se pretende agrupar.

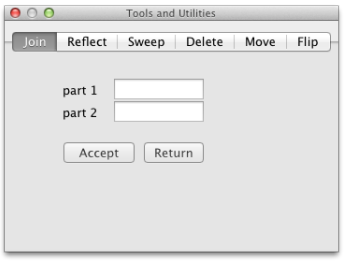


Figura 4.33: Opções para agrupar duas entidades numa só

Na presença de simetrias, é conveniente modelar apenas parte da geometria. No entanto para melhor se visualizarem os resultados ou efeitos dessa simplificação, feita com a aplicação de simetrias, é necessário reflectir o modelo segundo uma dada referência. No caso do *software* utilizado apenas é possível fazer essa mesma simetria, tendo por base um dos planos cartesianos. Ao escolher a funcionalidade "Reflect ..."surge a janela da figura 4.34 que dado o plano escolhido à partida, XY, XZ ou YZ, activa o respectivo separador. Depois basta simplesmente indicar o número da entidade que na qual se pretende aplicar a simetria.

Figura 4.34: Opções para criar simetrias segundos um plano cartesiano selecionado

Existem aproximações feitas nos cálculos que dão origem a nós com coordenadas muito próximas quando deveriam ser coincidentes e portanto unificados para não dar origem a nós órfãos. Para fazer essa mesma unificação e resolver esse problema existe uma ferramenta disponível no menu "Sweep". Escolhendo essa ferramenta, surgem as opões da figura 4.35 que dada uma tolerância irá analisar os nós seleccionados com o rato através do botão "Pick"ou especificando uma entidade de início e fim para essa mesma análise. No final os nós inúteis serão eliminados, e as conectividades actualizadas com a nova informação da base de dados.

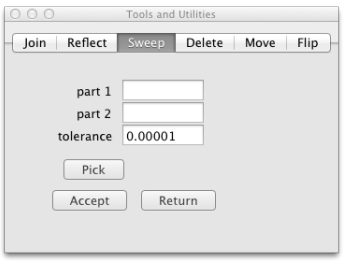


Figura 4.35: Opções para unificar nós com uma tolerância de posicionamento

Quando é necessário eliminar alguma entidade, recorre-se à ferramenta que permite a correcta gestão da base de dados neste ponto. A importância desta ferramenta prende-se com a correcta manutenção da base de dados ao eliminar uma entidade, no sentido em que todas as conectividades são recalculadas. É de salientar que quando se está perante uma entidade do tipo NURBS a eliminação é também propagada ao seu polígono de controlo, e vice-versa. A janela que possibilita essa operação é mostrada na figura 4.36.

Figura 4.36: Opções para eliminar uma entidade

Por vezes uma geometria é criada segundo determinadas condições mas depois surge a necessidade de a reposicionar para cumprir novos requisitos. É possível realizar esta tarefa com a ferramenta que o *software* dispõe para esse fim. É uma ferramenta que, como vemos na figura 4.37, requer como parâmetros de entrada a entidade à qual se pretende aplicar o deslocamento, e para que ponto absoluto, segundo o referencial global, se pretende fazer essa transformação.

Figura 4.37: Opções para criar deslocar uma entidade

Para aplicação das condições de fronteira nas entidades modeladas que seguirão para posterior análise, o *software* tem ao dispor, um menu intitulado "Boundary Conditions"que permite ao utilizador aplicar restrições à liberdade dos nós, aplicar cargas pontuais nodais, bem como cargas distribuídas nas faces dos elementos.

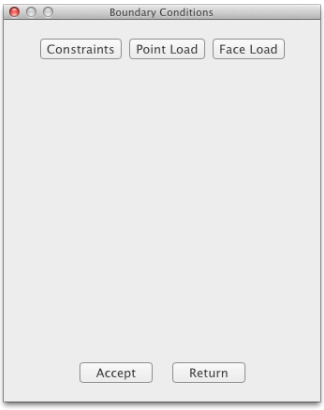


Figura 4.38: Ferramentas de aplicação das condições de fronteira

No menu "Contraints"estão disponíveis as funções que vemos na figura 4.39 que irão fazer com que os nós da entidade escolhida, ou os nós seleccionados com o rato, tenham os graus de liberdade bloqueados conforme a escolha feita com as caixas de validação que vemos na imagem. Os graus de liberdade 1 a 3 dizem respeito à translação segundo X, Y, Z respectivamente, e os graus 4 a 6 às rotações também segundo X, Y, Z respectivamente. É possível ver também uma representação gráfica dos graus de liberdade já restritos clicando no botão "Show Constraints".

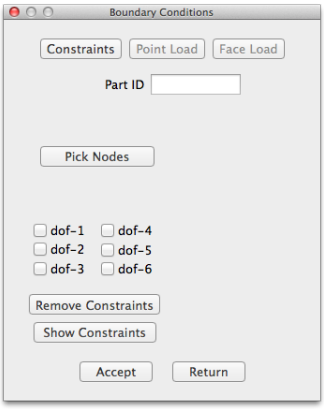


Figura 4.39: Opções para aplicar restrições aos nós quando à sua liberdade de movimentos

A aplicação de cargas pontuais é em tudo semelhante à restrição dos graus de liberdade descrita anteriormente. Fica apenas destacada a diferença, tal como vemos na figura 4.40, que é possível especificar o valor da carga pontual a aplicar em cada nó.

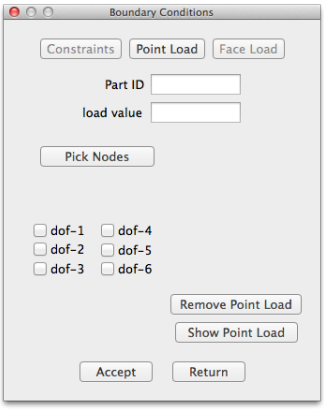


Figura 4.40: Opções para criar uma carga pontual nodal

Bastante comum é também a aplicação de uma carga distribuída segundo uma face do elemento. Neste módulo basta apenas que o utilizador indique a entidade à qual pretende aplicar a carga em todos os elementos, ou pelo contrário, seleccione os elementos com o auxílio do rato. Existe a possibilidade de mostrar essas mesmas cargas activando o modo de visualização das cargas com o botão "Show Face Load".

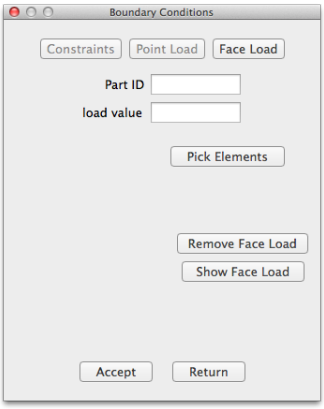


Figura 4.41: Opções para criar uma carga distribuída por face dos elementos

Em todos os menus está disponível a opção de remover as restrições ou cargas aplicadas. Para tal basta que se proceda da mesma forma que foi já explicada para a selecção, mas no final opta-se por clicar no botão "Remove...", aplicando-se a cada um dos três casos possíveis.

A definição da espessura de um nó assume por defeito o valor zero, isto é cada nó é apenas um ponto. No entanto em certas análises numéricas é conveniente definir um valor de espessura nodal diferente de zero. Com o auxílio desta funcionalidade presente no *software* , é possível o utilizador indicar a entidade cujos respectivos nós serão afectados da alteração pretendida, ou seleccionar apenas alguns nós com o auxílio do rato. A janela que surge quando se pretende efectuar essa mudança está visível na figura 4.42.

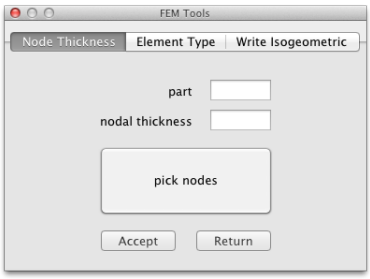


Figura 4.42: Opções para definir a espessura que os nós representam

Como referido atrás neste trabalho, uma análise divide-se em três etapas e todas as ferramentas / funcionalidades apresentadas até agora concentram-se na etapa do pré-processamento. Para que o *software* de cálculo consiga identificar qual o algoritmo numérico e formulação a utilizar, é necessário definir o seu tipo. Para realizar essa atribuição de elemento, recorre-se ao menu que está na figura 4.43 que apresenta duas possibilidades de selecção dos elementos das entidades presentes na base de dados. É novamente possível especificar qual a entidade que terá todos os seu elementos afectados da escolha, ou seleccionar com o rato os elementos nos quais se pretende aplicar essa mesma escolha.

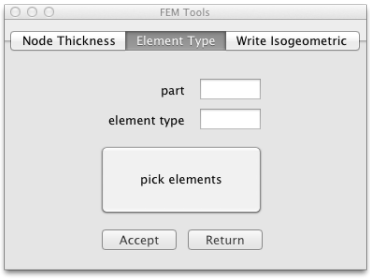


Figura 4.43: Opções para atribuir uma espessura aos nós em casos especiais

Os dados que serão submetidos a análise, só estarão completos quando for definido o material que constitui o modelo físico que estamos a representar. Para esse fim, existe um menu que apresenta ao utilizador as opções visíveis na figura 4.44. Para dar início à definição do material, é necessário clicar em "New"para criar um novo material. Os valores padrão são para um aço genérico. É importante aqui salientar como é feita a atribuição dos materiais aos respectivos elementos. De forma análoga ao que já foi descrito atrás podemos escolher uma entidade e clicar no botão "assign elements"para marcar todos os elementos dessa entidade como elementos alvo da operação, ou em alternativa, seleccionar com o rato esses mesmos elementos. De uma forma ou de outra é possível ter controlo sobre o número de elementos que serão afectados dessa alteração.

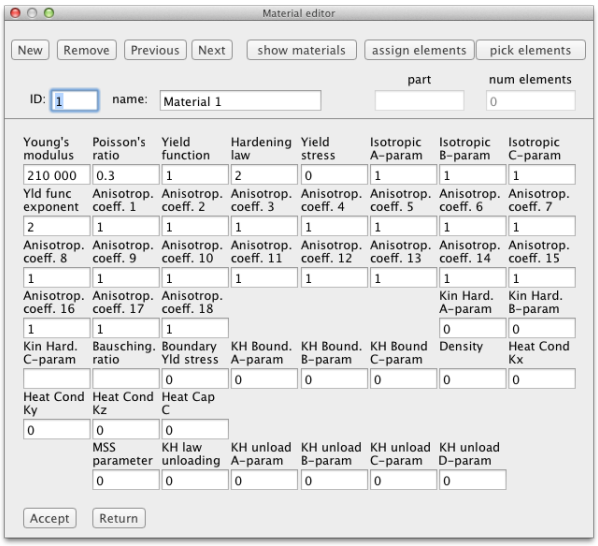


Figura 4.44: Opções para definição de materiais e atribuição dos respectivos elementos

Passadas as fazes do pré-processamento e do cálculo numérico é agora altura de visualizar os resultados provenientes dessa mesma análise. Conforme os resultados escolhidos para se obter como resultado da análise, é possível visualizar os seus valores, activando, de forma alternada, as opções que são mostradas na figura 4.45. Nessa mesma figura vemos que existe um marcador que podemos movimentar para visualizar as várias etapas da análise. Por defeito vemos sempre a deformação que ocorre a cada etapa da análise.

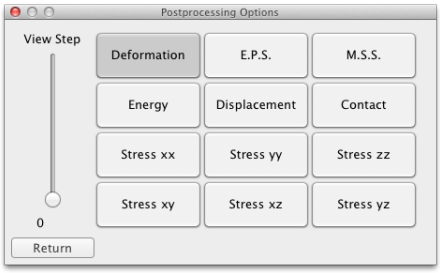


Figura 4.45: Opções para visualização de resultados – pós-processamento

Com as ferramentas atrás apresentadas, é possível criar modelos que descrevem a forma e as proporções de um objecto dentro de padrões aceitáveis.

Nas figuras seguintes, fica visível o potencial que o presente *software* já apresenta e resultado disso, são as geometrias que se seguem.

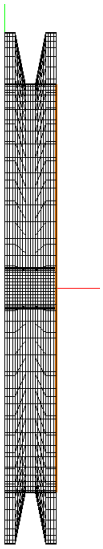


Figura 4.46: Exemplo 1 - polia em vista frontal

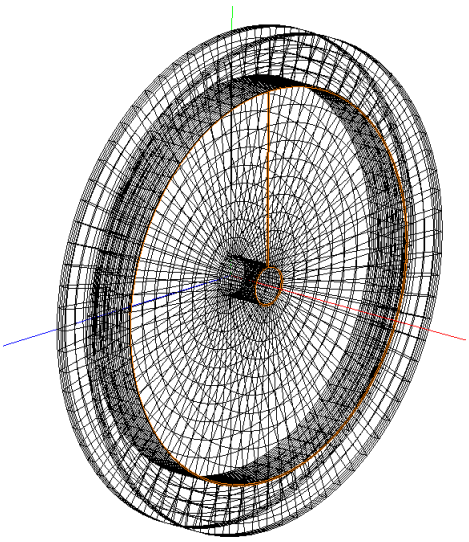


Figura 4.47: Exemplo 1 - malha da polia em perspectiva

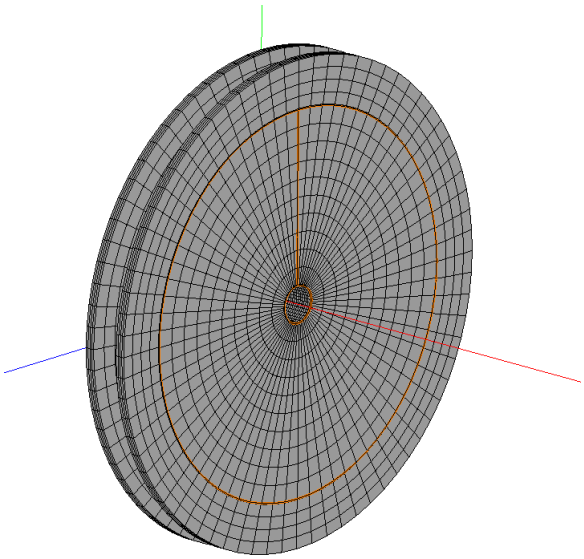


Figura 4.48: Exemplo 1 - malha da polia em perspectiva e renderizada

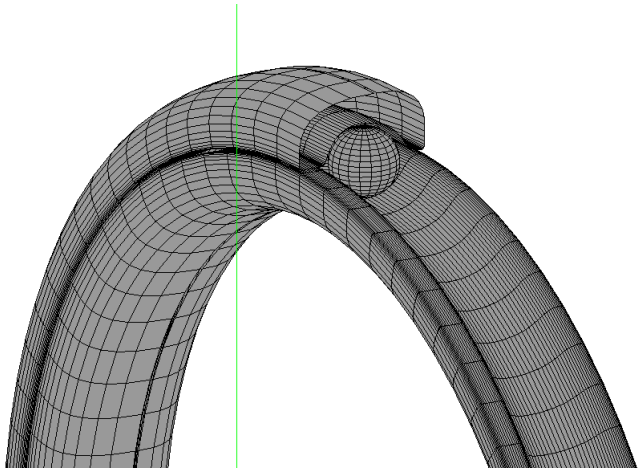


Figura 4.49: Exemplo 2 - vista parcial de um rolamento

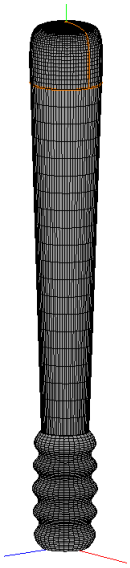


Figura 4.50: Exemplo 3 - taco de *baseball* renderizado

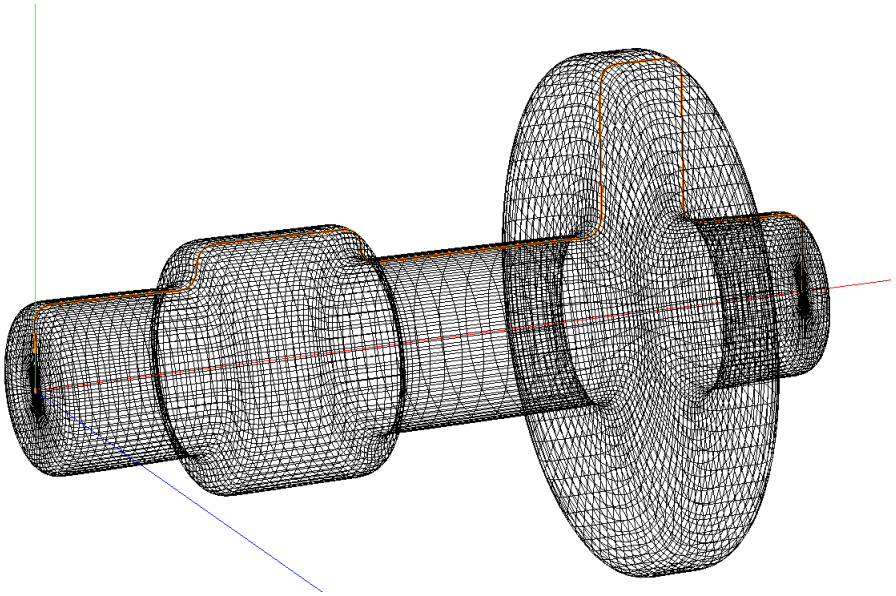


Figura 4.51: Exemplo 4 - malha de um veio em perspectiva

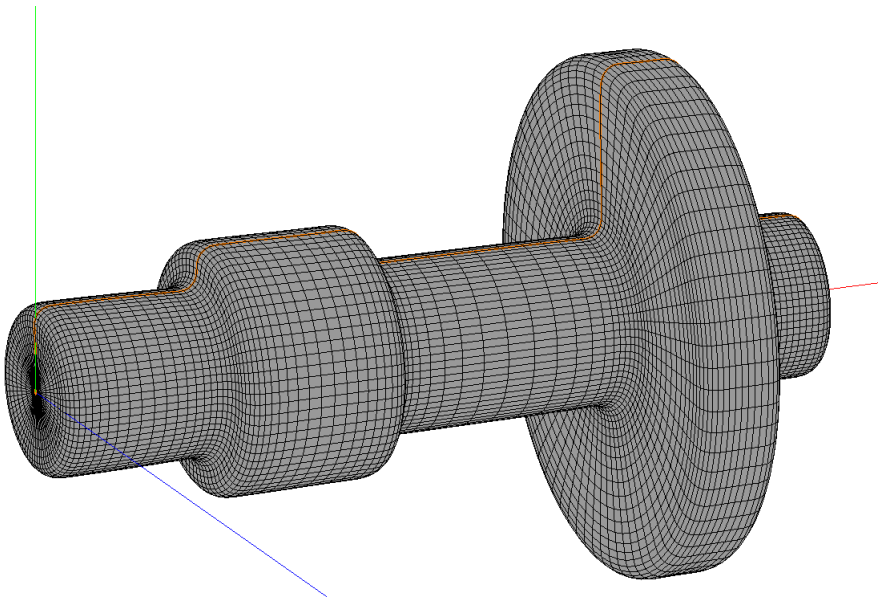


Figura 4.52: Exemplo 4 - malha de um veio em perspectiva e renderizado

4.3 *OpenGL*

A *OpenGL* é uma interface de *software* para controlar o hardware gráfico. Esta interface consiste em cerca de 150 comandos distintos que são usados nas operações necessárias para produzir aplicações tridimensionais interactivas.

A *OpenGL* é concebida para correr de forma suave e rápida, a sua interface é independente do hardware e pode ser implementada em mais que uma plataforma de hardware diferente. Para alcançar as características mencionadas, não existem comandos para executar tarefas nas janelas ou obter *inputs* do usuário. Em vez disso, os mesmos são geridos através do sistema operativo em que está a correr o *software* que implementa a *OpenGL*.

Da mesma forma, a *OpenGL* não fornece comandos de alto nível para descrever modelos de objetos tridimensionais. Tais comandos poderiam permitir que se especificassem formas relativamente complexas, como automóveis, partes do corpo, aviões, ou moléculas mas com a *OpenGL* , esses mesmos modelos são criados a partir de um pequeno conjunto de primitivas geométricas - pontos, linhas e polígonos.

Para tal foi criada uma biblioteca que fornece esses mesmos recursos, construída em *OpenGL*. A biblioteca de Utilitários *OpenGL* (UGL) oferece muitos dos recursos de modelação, tais como superfícies quadráticas, curvas e superfícies NURBS. A UGL é um *standard* de cada implementação de *OpenGL*, variando com o sistema operativo em questão. [12]

Ao nível da *OpenGL* foram implementadas funcionalidades que permitem realizar as operações elementares ao nível da representação geométrica, operações de manipulação de objectos nomeadamente rotações, *zoom*, e deslocamentos dos objectos desenhados no ecrã. Para tal foi necessário criar matrizes de projecção, de perspectiva. Funcionalidades mais avançadas, tais como iluminação e sombras, aplicação de padrões nas faces dos objectos, não foram contempladas pois na presente fase deste trabalho não se mostram como uma mais valia.

Comecemos por destacar que a *OpenGL* se baseia numa máquina de estados. Como tal sempre que queremos fazer uma alteração em algum dos seus recursos, temos que implicitamente dar essa ordem.

Para tal, foram estudadas as funcionalidades que permitem activar / desactivar os estados e os seus macros do compilador, renderizar pontos, linhas, linhas com tracejado, cores, espessuras das linhas, bem como todas as funções que preparam e inicializam as variáveis de estado da janela com a vista *OpenGL*.

Como exemplo se quisermos desenhar uma linha entre o ponto $a = 3,3$ e $b = 4,5$ usamos as seguintes instruções, para renderizar a linha:

```
a[2]={3,3};
b[2]={4,5};

glBegin(GL_LINES);
glVertex2i(a[0], a[1]);
glVertex2i(b[0], b[1]);
glEnd();
```

À medida que a complexidade das geometrias aumenta, ou mesmo o número de objectos que queremos renderizar em simultâneo, aumenta também o esforço por parte do sistema para em tempo útil conseguir executar essas tarefas.

Como tal foi também medida a performance do programa em execução para ser a implementação feita estava a ser demasiado penosa para o sistema. Esta medição foi feita para a representação de uma curva NURBS, em que a cada movimento do rato, é necessário recalcular todos os valores para poder representar a curva e termos assim a pré-visualização da mesma. O resultado dessa medição está na figura 4.53 e podemos ver que os picos de solicitação do processador apenas ocorrem quando o programa inicia e quando as alterações são guardadas.

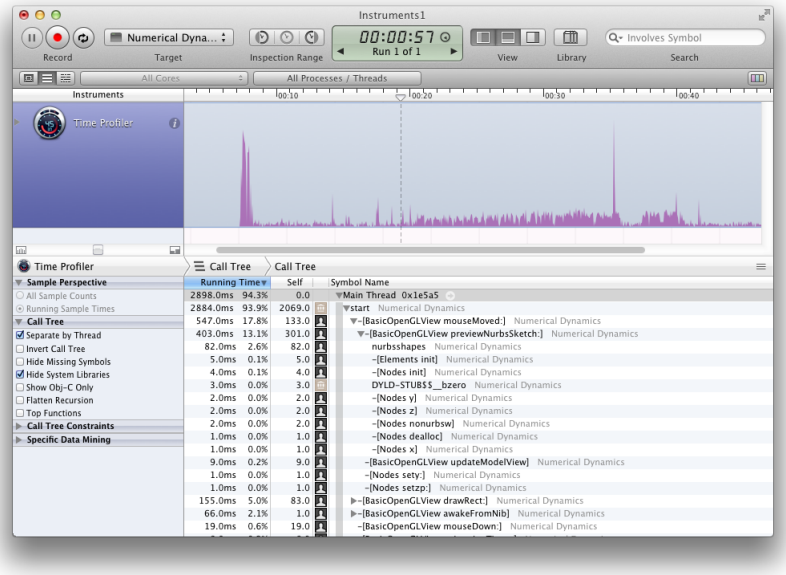


Figura 4.53: *Instruments* – monitorização do desempenho do programa

Capítulo 5

Resultados Numéricos

Para testar toda a teoria da AIG vista até agora, recorreu-se ao *software* desenvolvido e apresentado no presente trabalho, onde foram implementados dois problemas que servem de referência para validar a implementação do *software Numerical Dynamics* já apresentado.

Com os módulos e as ferramentas apresentados na secção 4 modelaram-se os problemas e foi feita a sua conversão para objectos do tipo NURBS.

No *software Numerical Dynamics* foi implementada uma rotina que exporta a entidade, que se pretende analisar, para um ficheiro, escrevendo as informações necessárias de acordo com as seguintes regras:

```
*NODES
...

*KNOTS
... direcção paramétrica Xi

*KNOTS
... direcção paramétrica Eta

*KNOTS
... direcção paramétrica Zeta

*PATCH
...

*CONTROL POINTS
...

*MATERIAL
...

*BOUNDARY=ISOGPOINTDISP
...
```

∗LOAD=ISOGPOINTLOAD
. . .

∗TABLE=LINEAR
. . .

∗JOB=ISOGEOMETRIC
. . .

∗ANALYSIS=UPDCNT

Posteriormente o *software* CEREBRO© [7] analisa o ficheiro com estes dados, executa os cálculos, e por fim devolve os resultados indicados no ficheiro.

Os dois problemas que foram submetidos para análise e os resultados obtidos destas simulações são avaliados de seguida.

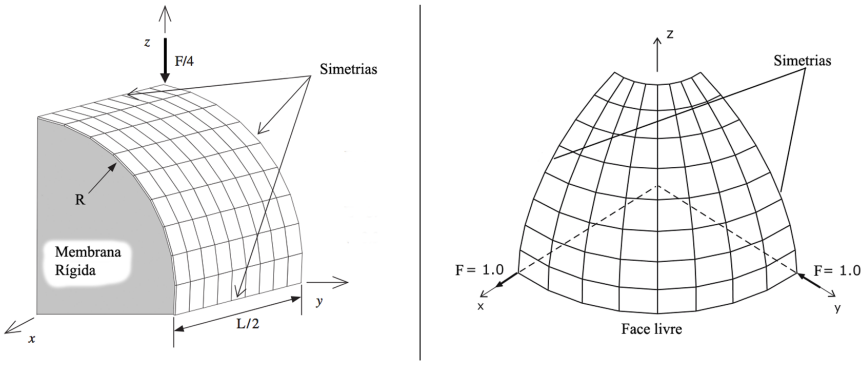


Figura 5.1: Dados dos problemas submetidos para análise

5.1 Problemas submetidos para análise

Submeteu-se a simulação os seguintes casos: uma casca semi-esférica, cum um furo, resultante da modelação de uma curva e posterior revolução; um cilindro pressionado lateralmente resultante da modelação de uma curva e posterior extrusão. Na figura 5.1 podemos ver a geometria da semi-esfera a) e do cilindro b).

Ambos os casos foram submetidos para análise com funções de base de grau dois e três. Desta forma consegue-se traçar as respectivas curvas de convergência, permitindo assim analisar de que forma o grau das funções de base influenciam a convergência da análise.

Problema a): um cilindro com membrana rígida nos topos, isto é não permite nenhuma translação dos nós das faces dos topos, pressionado com duas forças F centradas e diametralmente opostas. Por questões de simetria apenas foi modelado 1/8 do cilindro e utilizado um valor para a força de F/4. Para essa simetria os valores que a caracterizam são:

$$
\begin{aligned}
L &= 600 \text{ mm} \\
R &= 300 \text{ mm} \\
t &= 3 \text{ mm} \\
E &= 3 \times 10^6 \text{ MPa} \\
\nu &= 0.3 \\
F &= 1.0 \text{ N}
\end{aligned}
$$

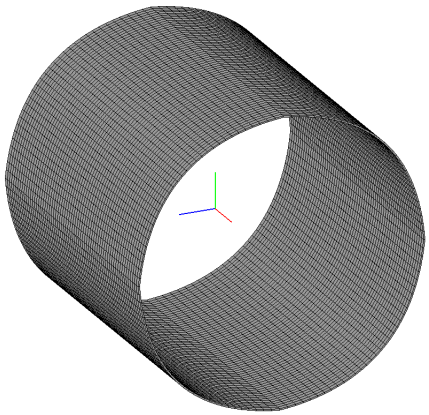


Figura 5.2: Caso de estudo b): cilindro

Problema b) uma semi esfera modelada por 1/4 da sua geometria dadas as simetrias existentes nos planos XZ e YZ, exercendo-se uma forma segundo a direcção negativa de Y e outra força segundo a direção positiva de X, mantendo-se os restantes graus de

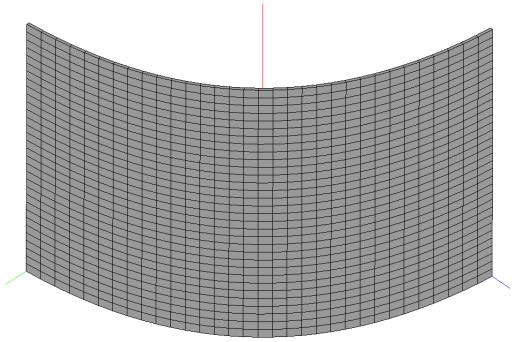


Figura 5.3: Aplicação de tripla simetria

liberdade sem qualquer restrição.

$$
\begin{aligned}
R &= 10 \text{ mm} \\
t &= 0.04 \text{ mm} \\
E &= 6.852 \times 10^7 \text{ MPa} \\
\nu &= 0.3 \\
F &= 1.0 \text{ N}
\end{aligned}
$$

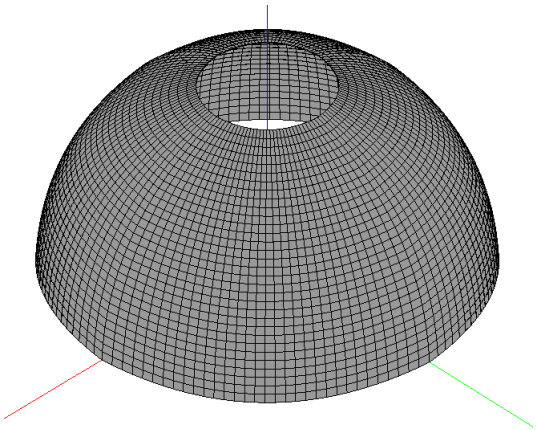


Figura 5.4: Caso de estudo a): semi-esfera com furo

No caso de estarmos perante uma geometria obtida com polinómios de grau dois, utiliza-se a integração de Gauss 3×3, e no caso de uma geometria obtida com polinómios de grau três, utiliza-se a integração de Gauss 4×4. Estas integrações são feitas no plano, uma vez que se tratam de abordagens do tipo placa e casca, e apenas nos queremos focar sobre o que se passa no plano.

Recorrendo ao *software Numerical Dynamics* importou-se o ficheiro de resultados para visualização dos *outputs* escolhidos, no caso dos presentes problemas foram apenas escolhidos os deslocamentos como *output*.

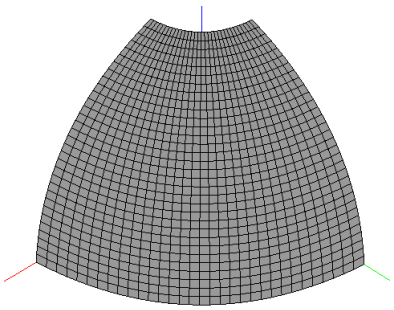
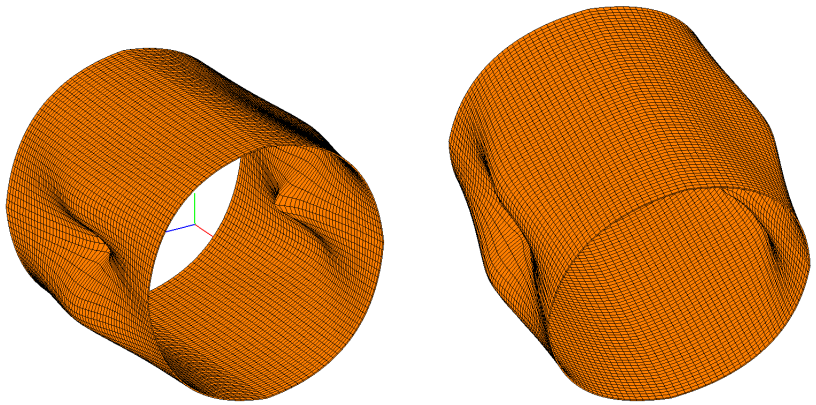


Figura 5.5: Aplicação de dupla simetria

5.2 Resultados obtidos: Cilindro

No caso do cilindro foi adoptada tripla simetria, ou seja simetria segundo os três planos cartesianos.



(a) (b)

Figura 5.6: Deformada do cilindro em duas perspectivas diferentes

Na figura 5.6 a) e b) vemos a deformada do cilindro, com a aplicação de um factor de escala de 100, uma vez que as deformações resultantes da aplicação de uma força com um valor desta ordem, são visualmente são imperceptíveis.

À semelhança dos resultados obtidos por Hughes [1], também os resultados obtidos neste trabalho apresentam uma deformação bastante localizada.

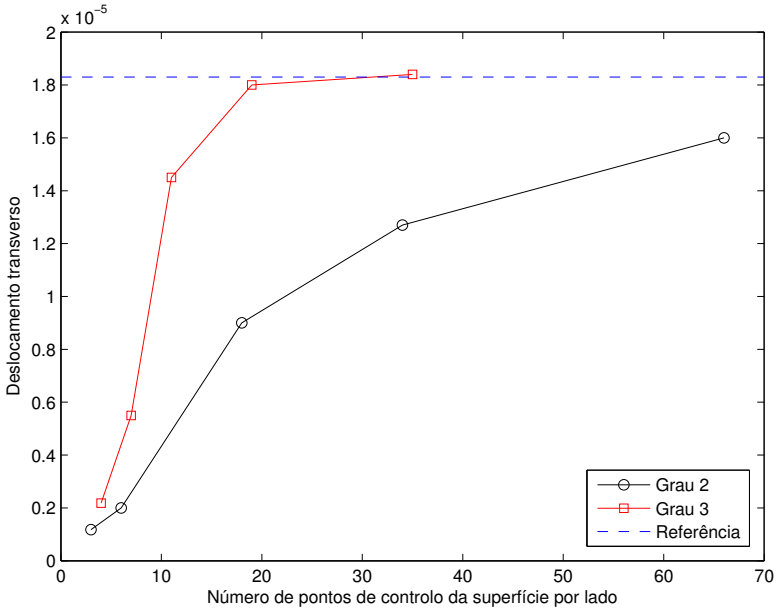


Figura 5.7: Convergência da solução para o caso do cilindro

Através do gráfico da figura 5.7 podemos verificar que a solução no caso do cilindro converge para o valor de referência quando usamos funções de grau três, e com funções de grau dois essa convergência não é conseguida. Esta convergência da solução que é visível no gráfico, valida também a implementação da análise isogeométrica efectuada para problemas do tipo placa e casca.

5.3 Resultados obtidos: Semi-esfera com furo

Perante o caso da semi-esfera foi adoptada dupla simetria, sendo ela segundo os planos XZ e YZ.

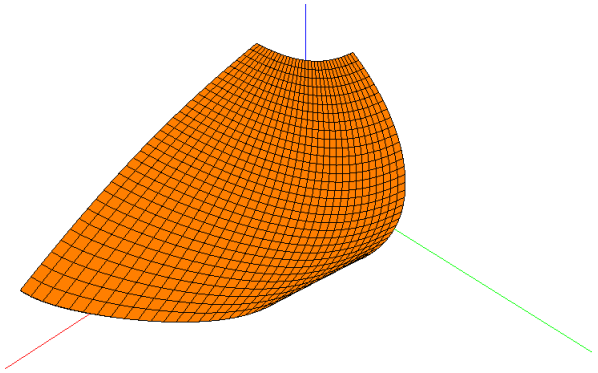
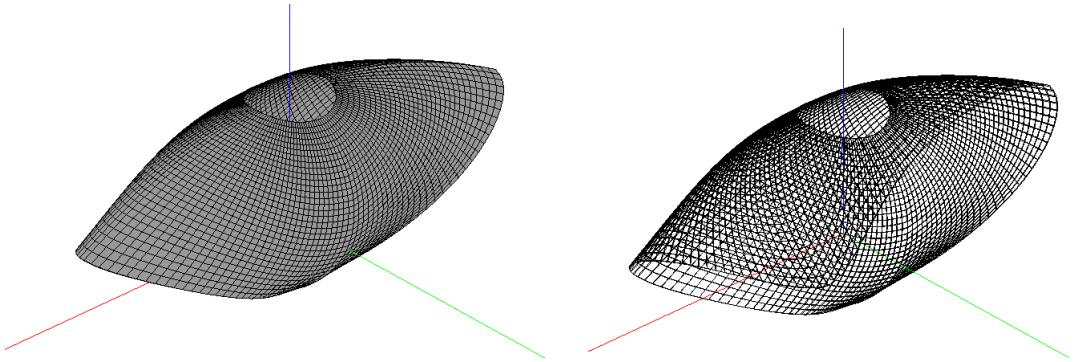


Figura 5.8: Deformada da dupla simetria da semi-esfera

Na figura 5.9 a) e b) vemos a deformada da semi-esfera, com a aplicação de um factor de escala de 100, uma vez que as deformações resultantes do valor da força aplicada, visualmente são imperceptíveis, tal como no caso do cilindro.



(a) (b)

Figura 5.9: Deformada da semi-esfera com e sem aplicação de cor na geometria

Uma vez que é de todo o interesse verificar a convergência da solução, para o valor de referência numa AIG, foram registados os valores dos deslocamentos transversos e dos pontos de controlo por lado do presente problema. Essa convergência é mostrada no gráfico da figura 5.10

Através do gráfico da figura 5.10 podemos verificar que a solução converge para o valor de referência, e que essa convergência é tanto mais rápida quando maior for o grau

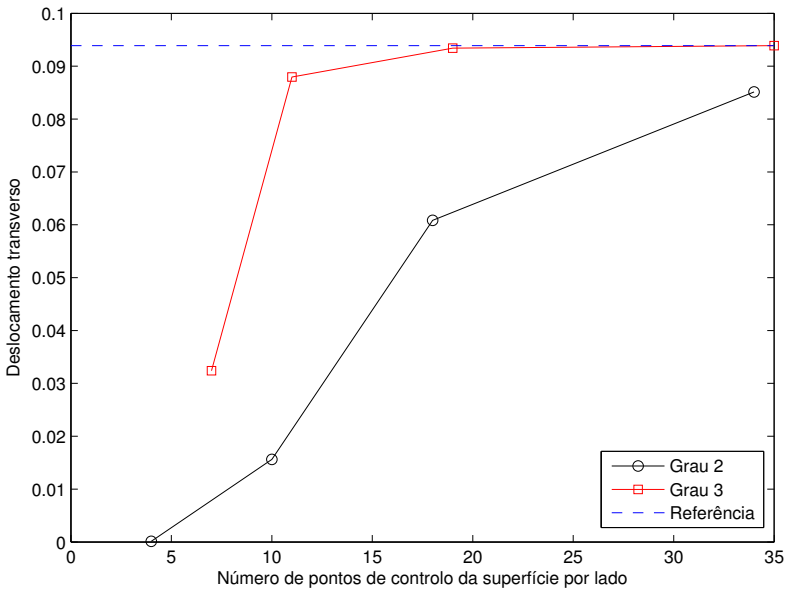


Figura 5.10: Convergência da solução da semi-esfera

dos polinómios que formam as funções de base. Esta convergência da solução que é visível no gráfico, valida também a implementação da análise isogeométrica efectuada para problemas do tipo placa e casca.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho é uma busca incessante pela optimização de métodos e aperfeiçoamento de técnicas que permitam tornar as simulações mais rápidas e mais precisas. Como tal, nunca podemos dizer que o trabalho está concluído, podemos sim dizer que os objectivos foram atingidos.

6.1 Conclusões

A AIG é, como foi referido várias vezes ao longo deste trabalho, um caminho bastante promissor no campo das análises numéricas. Apesar dos trabalhos já feitos nesta área ainda há um longo caminho a percorrer até esta matéria estar plenamente implementada e difundida.

Com os resultados obtidos e já apresentados, conclui-se que, recorrendo à AIG podem-se obter os resultados da simulação de forma bastante mais rápida, a nível do tratamento e processamento da informação requerida para todo o processo.

É de salientar que as geometrias da deformada das superfícies, apresentam bastante rigor na representação da sua geometria devido ao facto de serem superfícies construídas com base em NURBS.

Estas análises apresentam taxas de convergência bastante elevadas, conseguindo-se esse aumento com a elevação do grau das funções base, para um mesmo número de pontos de controlo por face.

Na análise do cilindro não se obteve convergência com funções de grau dois, tendo sido conseguida essa mesma convergência apenas com funções de grau três. Das convergências referidas retira-se também a conclusão que a implementação efectuada da análise isogemétrica é válida para as estruturas do tipo placa e casca.

O pressuposto inicial, de que este tipo de análise comparativamente com o método tradicional do elementos finitos é benéfico quanto à performance de toda a análise, bem como na representação da geometria do corpo, fica confirmado com base nestes resultados.

Foi também alvo deste trabalho a adaptação do *software Numerical Dynamics* para a possibilidade de desenhar curvas NURBS bem como de optimizar o *software* em alguns aspectos gerais ao nível do funcionamento do programa. Neste ponto os objectivos impostos à partida foram cumpridos, na medida em que o *software* permite obter dados para posterior análise.

6.2 Trabalhos Futuros

Por forma a dar continuidade a este trabalho, novas e melhoradas ferramentas devem ser implementadas para tornar a interacção com o utilizador mais satisfatória.

Uma ferramenta de cotagem, permitiria também uma maior versatilidade na utilização do programa, bem como a possibilidade de realizar geometrias mais complexas.

Uma rotina para importação e exportação de dados a no formato "*.iges"tornará bastante versatilidade o *software* pois permitirá integrar o mesmo com outros já existentes, e também o processamento de modelações já existentes nesse formato, mesmo que feitas recorrendo a outro *software* existente. A importação de ficheiros permitirá análises às modelações já feitas com esses *softwares* para uma análise numérica mais específica, como é p.ex. uma análise numérica de um processo de estampagem, ou uma análise de fractura, entre outros.

No âmbito da investigação, este *software* pode tornar-se mais versátil após a possibilidade de personalizar a forma como os dados são escritos num ficheiro, para serem submetidos a uma análise feita com *software* próprio.

Por fim e por motivos claros, a implementação das rotinas de cálculo no próprio programa *Numerical Dynamics* fariam deste *software* uma poderosa ferramenta para uso industrial ou trabalhos de projecto.

Outra caminho igualmente viável seria a criação de um *software* como o que vemos na figura 6.1 que é bastante mais minimalista, no entanto é mais orientado para quem pretende apenas correr simulações num ficheiro do tipo "*.iges". Depois de definir os carregamentos, as condições fronteira, material, entre outros parâmetros, o utilizador poderá fazer uma pré-visualização do que será submetido para análise e aguardar que essa mesma análise seja executada.

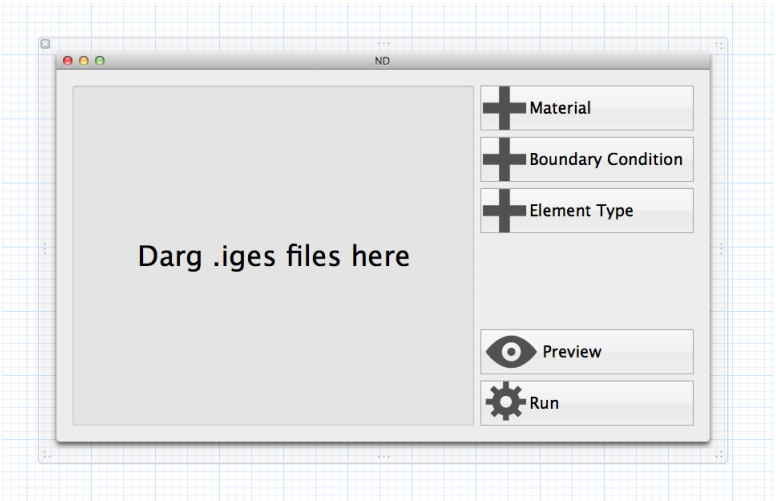


Figura 6.1: Exemplo do referido trabalho futuro

Bibliografia

[1] Cottrell JA, Hughes TJR, Bazilevs Y. Isogeometric Analysis: Toward Integration of CAD and FEA. John Wiley & Sons; 2009.

[2] Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering. 2005;194:4135–4195.

[3] Piegl LA, Tiller W. The Nurbs Book. Monographs in Visual Communication. Springer; 1997.

[4] Rogers DF. An Introduction to NURBS: With Historical Perspective. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Elsevier Science; 2000.

[5] Li B, Qin H. Component-aware tensor-product trivariate splines of arbitrary topology. Computers & Graphics. 2012;36(5):329 – 340.

[6] Cardoso RPR, de Sa JMAC. The enhanced assumed strain method for the isogeometric analysis of nearly incompressible deformation of solids. International Journal for Numerical Methods in Engineering. 2012;92:56–78.

[7] Cardoso RPR. CEREBRO - finite element analysis software. Licence ASSOFT No. 1169/D/04. 2004;.

[8] Hillegass A. Cocoa Programming for Mac OS X. Addison-Wesley; 2008.

[9] Inc A. Object-Oriented Programming with Objective-C: Why Objective-C?; 2012. Available from: `https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/OOP_ObjC/Articles/ooWhy.html` [cited 2012-09-18].

[10] Stevenson S. Cocoa and Objective-C: Up and Running. O'Reilly; 2010.

[11] Hillegass A. Objective-C Programming: The Big Nerd Ranch Guide. Big Nerd Ranch Guides. Addison Wesley Professional; 2011.

[12] Shreiner D. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1. OpenGL Series. Addison-Wesley; 2010.