FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

DEPARTAMENTO DE
ENGENHARIA MECÂNICA

Estudo Hidrodinâmico de um veículo subaquático

Dissertação apresentada para a obtenção do grau de Mestre em Engenharia Mecânica na Especialidade de Produção e Projecto

Autor

Tiago Filipe Costa Cristóvão

Orientadores

Almerindo Domingues Ferreira

Micael Santos Couceiro

Júri

Presidente	**Professor Doutor Cristóvão Silva** **Professor Auxiliar da Universidade de Coimbra**
Vogais	**Professor Doutor Marta Cristina Cardoso de Oliveira** **Professor Auxiliar da Universidade de Coimbra**
Orientador	**Professor Doutor Almerindo Domingues Ferreira** **Professor Auxiliar da Universidade de Coimbra**

Colaboração Institucional

Matereospace, Lda

MATEREO
LIFETIME PERFORMANCE AND RELIABILITY DESIGN

Coimbra, Fevereiro, 2016

Agradecimentos

Agradeço aos meus pais, Isabel Maria Costa e Francisco José Cristóvão, e ao meu irmão Alexandre Rafael Cristóvão por me apoiarem durante o meu percurso académico. Ao Jorge Alexandre Vieira, ao professor Almerindo Domingues Ferreira e ao Micael Santos Couceiro agradeço por me darem a oportunidade de trabalhar em algo que, para além de interessante, influencia diretamente o meu percurso profissional.

Resumo

O estágio proposto pela entidade Matereospace Lda, designada por Matereo no presente documento, enquadra-se no âmbito de um estudo de um sistema multi-veículo denominado ECHO5. Este sistema é composto por um veículo subaquático acoplado fisicamente a um veículo de superfície mediante um cabo umbilical. O estudo, a desenvolver no âmbito do estágio, focou-se no estudo hidrodinâmico do veículo subaquático. Para o efeito, dividiu-se a tarefa em duas fases. A primeira fase, dada a necessidade de enquadrar o sistema multi-veículo ECHO5 no mercado, teve como objetivo a familiarização com os diversos veículos subaquáticos existentes. Nesse sentido, o veículo subaquático foi classificado segundo o tipo de controlo e enquadrado numa de três áreas de ação apresentadas no capítulo dois, nomeadamente a área da investigação científica, a área comercial e a área da defesa. O tipo de controlo das soluções apresentadas nessas diversas áreas pode ser remoto (*e.g.*, sistemas completamente teleoperados), autónomo e híbrido. A partir das áreas de ação e classes de veículos subaquáticos, concluiu-se que o veículo subaquático em estudo pertence à categoria de sistemas híbridos e encontra-se integrado na área de ação comercial.

A segunda fase, teve como objetivo a exploração das ferramentas OpenFOAM, OpenSCad e Matlab, com a finalidade de se criar uma plataforma integrada onde é possível, a partir de um conjunto de parâmetros, gerar uma geometria, um domínio de simulação, simular e obter os resultados da mesma. Para que todo o processo seja autónomo, foi desenvolvido um conjunto de *script*s em Matlab. A partir do trabalho apresentado, torna-se assim possível controlar as diversas ferramentas e realizar um estudo mais pormenorizado da fuselagem do veículo subaquático.

Palavras-chave: Veículo Subaquático, Hidrodinâmica, *Software* Matlab, *Software* OpenSCad, *Software* OpenFOAM.

Abstract

The internship proposed by the entity Matereospace Lda, herein referred to as Matereo, falls within the scope of a multi-vehicle system study called ECHO5, which is composed of an underwater vehicle attached physically to a surface vehicle by an umbilical cord. The study carried out in the internship was focused on the hydrodynamic study of the underwater vehicle. This task was divided into two phases. The first phase, given the need to frame the ECHO5 multi-vehicle system on the market, was aimed at the familiarization with the various existing underwater vehicles. In this sense, in chapter two, the underwater vehicle was classified according to the type of control and framed within three action areas, namely scientific research, commercial and defense areas. The control type of the solutions presented in these areas may be remote (ie, completely remotely operated systems), autonomous or hybrid. From the action fields and underwater vehicle classes, we came to the conclusion that the underwater vehicle under study belongs to the category of hybrid systems and aimed at the commercial action area.

The second phase, aimed at exploring the OpenFOAM tools, OpenSCad and Matlab, with the purpose of creating an integrated platform where it is possible to, from a set of parameters, generate a geometry, a simulation domain, a simulation and obtain the results thereof. For the whole process to be independent, a set of MATLAB scripts were developed. From the presented work, it becomes possible to control the various software and perform a more detailed study of the underwater vehicle fuselage.

Keywords Underwater Vehicle, Hydrodynamics, Matlab, OpenSCad, OpenFOAM.

Índice

Índice de Figuras v
Índice de Tabelas vii
1. INTRODUÇÃO 8
2. ESTADO DA ARTE 11
2.1. Veículos subaquáticos 11
2.1.1. Áreas de ação dos veículos subaquáticos 11
2.1.2. Classificação de veículos subaquáticos 14
2.2. Geometria da Fuselagem 17
2.2.1. Modelo geométrico da fuselagem 18
3. MODELAÇÃO DA FUSELAGEM 20
3.1. Aplicação das equações de Myring ao *Software* OpenSCad 20
3.2. Integração do software OpenSCad no software Matlab 24
4. ANÁLISE NUMÉRICA DA FORÇA E DO COEFICIENTE DE ARRASTO 27
4.1. Pré-processamento 27
4.1.1. Condições de Fronteira 28
4.1.2. Propriedades do fluido 29
4.1.3. Modelo de Turbulência 29
4.1.4. Independência da Malha de elementos finitos 30
4.2. Validação de resultados 32
4.3. Obtenção da força e coeficiente de arrasto 33
4.4. Integração do software OpenFOAM no software Matlab 34
4.4.1. Copiar a fuselagem para a pasta da simulação 38
4.4.2. Geração da Malha de Elementos Finitos 38
4.4.3. Simulação 39
4.4.4. Obter a Força e o Coeficiente de Arrasto 39
4.4.5. Limpar Diretoria de Simulação 40
5. RESULTADOS 42
5.1. Estudo numérico da fuselagem do veículo subaquático 42
5.2. Fuselagem do veículo subaquático final 44
6. CONCLUSÕES 46
7. REFERÊNCIAS BIBLIOGRÁFICAS 47
ANEXO A – Código do script run.m 49
ANEXO B- Resultados obtidos para vários parâmetros de n, θ. 53
ANEXO C- Resultados obtidos para n=1,65 e diferentes valores de θ 54
APÊNDICE A – Script para a obtenção das forças de arrasto (gerado pelo Matlab) 55
APÊNDICE B – Script para a obtenção dos coeficientes de arrasto (gerado pelo Matlab) 56

ÍNDICE DE FIGURAS

Figura 1 Sistema multi-veículo ECHO5 9
Figura 2 Veículo subaquático do Sistema multi-veículo ECHO5. 9
Figura 3 Monitorização da qualidade da água (Lim et al., 2009). 10
Figura 4 Imagens retiradas pelo veículo subaquático SeaBED (Singh et al., 2002). 10
Figura 5 Foto do AUV Maya (Mascarenhas, et al., 2015) 11
Figura 6 Mapeamento da variação da turbidez ao longo do estuário Mandovi (Mascarenhas, et al., 2015). 12
Figura 7 AUV Iver3-580 da OceanServer (Ocean Server Technology, Inc., 2015). 13
Figura 8 Planeamento de uma missão de mapeamento do AUV IVER3 (Ocean Server Technology, Inc., 2015) 13
Figura 9 AUV Remus 100 (Charles, et al., 2011). 14
Figura 10 Esquema de utilização de um ROV (National Oceanic and Atmospheric Administration, 2002). 15
Figura 11 Lançamento do AUV REMUS 100 pela Marinha Norte Americana (Nicholson e Healey, 2008). 16
Figura 12 Esquema classificativo do sistema HURV (Xiang et al., 2015). 16
Figura 13 Operação de uma missão como sistema HURV (Xiang et al., 2015). 17
Figura 14 Parâmetros geométricos segundo as equações de Myring (Sousa et al., 2014). 18
Figura 15 Diferentes curvas do nariz em função de n (Sousa et al., 2014). 19
Figura 16 Diferentes curvas da cauda em função de θ (Sousa et al., 2014). 19
Figura 17 Extrusão de uma geometria 2D em 3D (OpenSCad User Manual, 2015). 20
Figura 18 Código para a geração da curva do nariz 21
Figura 19 Geometria 2D do nariz. 22
Figura 20 Código para a geração da curva da cauda 22
Figura 21 Geometria 2D da cauda. 22
Figura 22 Curvas do nariz e cauda na sua posição final. 23
Figura 23 Extrusão radial das curvas de Myring do nariz e da cauda. 23
Figura 24 Código para o posicionamento e extrusão radial do nariz, cauda e secção central. 23
Figura 25 Fuselagem final. 24

Figura 26 Fuselagem Seccionado a 90°. 24
Figura 27 Comando para a geração da fuselagem. 24
Figura 28 Código para geração da fuselagem pelo *software* Matlab. 25
Figura 29 Domínio de simulação. 28
Figura 30 Discretização 1. 31
Figura 31 Código para o cálculo do coeficiente de atrito, ficheiro controlDict. 33
Figura 32 Código para o cálculo da força de atrito introduzido no ficheiro controlDict. 34
Figura 33 Arquitetura do processo de simulação. 35
Figura 34 Pastas necessárias para o processo de simulação. 35
Figura 35 Ficheiros da pasta 0. 35
Figura 36 Ficheiros da pasta geo. 35
Figura 37 Ficheiros da pasta sim. 36
Figura 38 Ficheiros da pasta constant. 36
Figura 39 Ficheiros da pasta Results. 36
Figura 40 Ficheiros da pasta system. 37
Figura 41 Scripts do Matlab para realizar uma simulação. 37
Figura 42 Código para copiar a fuselagem para a pasta da simulação. 38
Figura 43 Código para a geração da malha de elementos finitos. 38
Figura 44 Código para realizar uma simulação em paralelo e obter os resultados de $y+$. 39
Figura 45 Código para retirar a Força, o Coeficiente de Arrasto e o y+. 40
Figura 46 Código para apagar todas as pastas e ficheiros não necessários para a simulação seguinte. 41
Figura 47 Coeficiente de arrasto obtido para diferentes valores de n e θ. 42
Figura 48 Coeficiente de arrasto obtido para diferentes valores de n com $\theta = 23º$. 43
Figura 49 Coeficiente de arrasto para valores de n= [21;25]. 43
Figura 50 Representação da malha de elementos finitos da fuselagem final. 44
Figura 51 Refinamento de malha de elementos finitos e camadas. 45

ÍNDICE DE TABELAS

Tabela 1 Sensores utilizados no AUV Maya (Desa e Maya Team, 2007). 12
Tabela 2 Geometria do domínio de simulação. 28
Tabela 3 Condições de Fronteira. 29
Tabela 4 Propriedades do fluido considerado. 29
Tabela 5 Nº de elementos da discretização 1. 30
Tabela 6 Refinamento volumétrico considerado em todas as malhas de elementos finitos de simulação. 30
Tabela 7 Resultados obtidos da simulação na discretização 1. 31
Tabela 8 Parâmetros da discretização 2. 32
Tabela 9 Parâmetros da discretização 3. 32
Tabela 10 Resultados obtidos na simulação do cilindro para três discretizações diferentes. 32
Tabela 11 Erro percentual dos resultados obtidos em comparação com o resultado experimental. 33
Tabela 12 Parâmetros finais da fuselagem do veículo subaquático. 44

1. INTRODUÇÃO

A monitorização da qualidade da água não só é uma das linhas de atuação prioritárias da União Europeia mas também é uma das chaves para o desenvolvimento sustentável (Prammer, 1998). O projeto *Undersee* (Matereospace Lda, 2016), cooperação entre a Matereo[1], sedeada no Instituto Pedro Nunes e a Ingeniarius[2], sedeada no Centro Tecnológico da Cerâmica e do Vidro, tem como objetivo o desenvolvimento do sistema multi-veículo ECHO5, apresentado na Figura 1. Este sistema tem como objetivo prestar serviços nas áreas de monitorização da qualidade da água, inspeção de infraestruturas e levantamentos topográficos. A empresa Matereo iniciou a sua atividade em Agosto de 2014 tendo como área de ação a investigação e desenvolvimento de produto e serviços em áreas da engenharia de estruturas e engenharia mecânica. Com a evolução da empresa e a criação de conhecimento nas diversas áreas de atuação, iniciou-se o projeto *Undersee* para responder às necessidades identificadas no âmbito da prestação de serviços de monitorização da qualidade da água de sistemas em "alta" (*e.g.,* emissários submarinos de industrias poluentes) (Matereospace Lda, 2016). Destas necessidades surgiu o projeto Undersee. (Matereospace Lda, 2016). A empresa Ingeniarius iniciou a sua atividade em Janeiro de 2014, tendo como área de ação a investigação e desenvolvimento de produto em áreas como a robótica e a automação (Ingeniarius Lda, 2016).

O estágio, denominado "Estudo hidrodinâmico de um veículo subaquático", visa o estudo do veículo subaquático, apresentado na Figura 2, que está inserido no sistema multi-veículo ECHO5, responsável por todas as atividades subaquáticas.

[1] http://matereo.com/
[2] http://ingeniarius.pt/

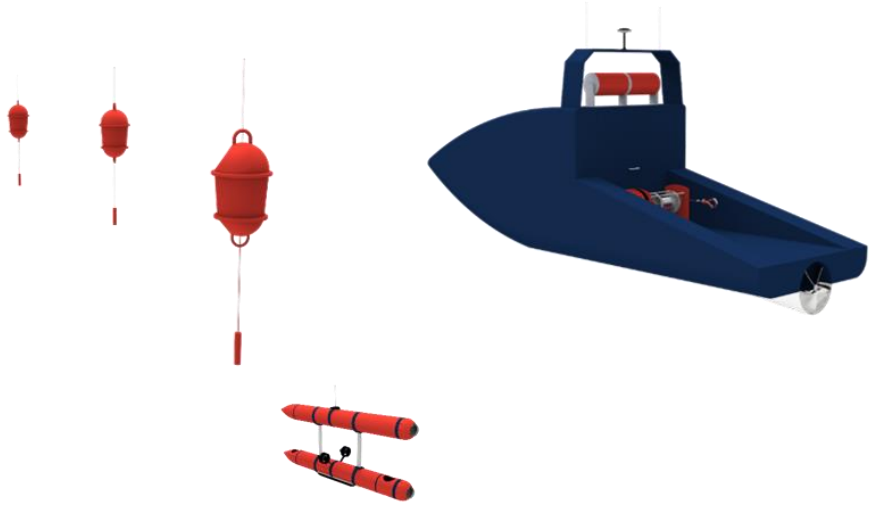

Figura 1 Sistema multi-veículo ECHO5

O ECHO5 é um sistema multi-veículo composto por um veículo de superfície, um conjunto de sonobóias e um veículo subaquático, como representado na Figura 1,que integra diferentes tipos de sensores à superfície e em profundidade.

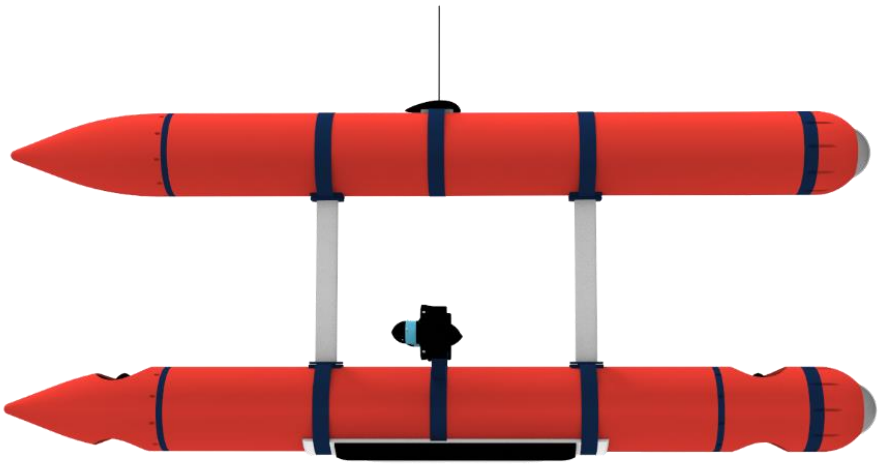

Figura 2 Veículo subaquático do Sistema multi-veículo ECHO5.

O sistema multi-veículo ECHO5 tem as seguintes funcionalidades principais:

1. A monitorização da qualidade da água, apresentada na Figura 3, de modo a melhorar o processo de tomada de decisão assegurando aos nossos clientes informação em tempo real.

Figura 3 Monitorização da qualidade da água (Lim et al., 2009).

2. A inspeção costeira de infraestruturas, em tempo real, a partir de um conjunto de câmaras específicas para o efeito.
3. Levantamentos topográficos costeiros através de um sonar de varrimento lateral como apresentado na Figura 4.

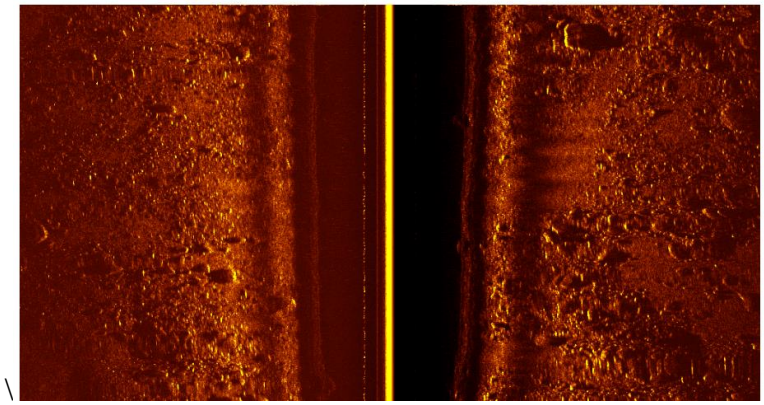

Figura 4 Imagens retiradas pelo veículo subaquático SeaBED (Singh et al., 2002).

2. ESTADO DA ARTE

2.1. Veículos subaquáticos

Os veículos subaquáticos abrangem uma grande diversidade de missões, desde a recolha de imagens de fontes hidrotermais (Marsh et al., 2013), até à deteção de minas (Healey e Nicholson, 2008), sendo assim dotados de sistemas sensoriais distintos. Para uma melhor compreensão dos veículos subaquáticos e respetiva constituição, as próximas secções apresentam um enquadramento dentro da respetiva área de ação de cada um, classificando-os segundo o tipo de controlo.

2.1.1. Áreas de ação dos veículos subaquáticos

Os veículos subaquáticos têm um vasto espectro de utilização. Este pode ser dividido em três áreas de ação: Investigação Científica, Comercial e Defesa (Ruiz, 2010):

2.1.1.1. Área da investigação científica

Os veículos subaquáticos utilizados na investigação científica são dotados de sensores específicos. Um exemplo disso é o veículo subaquático autónomo (*autonomous underwater vehicle*, ou AUV) Maya, apresentado na Figura 5. Este sistemafoi desenvolvido pelo Instituto Nacional Oceanográfico Indiano para utilização em água doce, sendo posteriormente modificado para utilização em água salgada (Sahu e Subudhi, 2014).

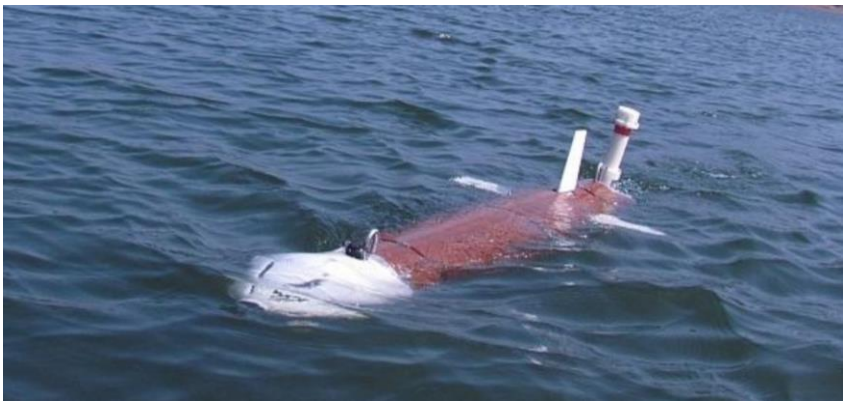

Figura 5 Foto do AUV Maya (Mascarenhas, et al., 2015)

O AUV Maya utiliza os sensores apresentados na tabela seguinte:

Tabela 1 Sensores utilizados no AUV Maya (Desa e Maya Team, 2007).

Sensor used on Maya	**Manufacturer**	**Range & Accuracy**	**Size & Weight**
Dissolved Oxygen	Aanderraa, Norway	0-500uM (<8uM or 5%)	36 x 86mm. 120 gms
Chlorophyll	WetLabs, USA	0.02–60µg/l	63 x 133mm, 80 gms
Turbidity		0–25 NTU	
Conductivity, Temperature, Depth	RBR, Canada	0 to70 mS/cm, (± 0.003 mS/cm)	400mm x 64mm, 389 gms
		-5 °C to 35 °C, (± 0.002 °C)	
		0 to 740 dBars, (<0.001% FS)	
Hyper spectral Radiometers	TRIOS, Germany	Irr: 280-500nm, (6 to 10 %)	47 x 260mm, 838 gms
		Rad: 320-950nm (6%)	47 x 297mm, 913 gms

A partir dos sensores apresentados na Tabela 1, foi possível nos dias 8 e 11 de junho de 2015 a realização de uma missão, por parte da Divisão de Instrumentação Marinha do Instituto Nacional de Oceanografia de Goa, em que o AUV MAYA adquiriu as propriedades físicas, químicas e biológicas da superfície das águas dos estuários Zuari e Mandovi. Um exemplo dos resultados obtidos é a variação de turbidez, conforme apresentado na Figura 6.

Figura 6 Mapeamento da variação da turbidez ao longo do estuário Mandovi (Mascarenhas, et al., 2015).

2.1.1.2. Área comercial

Os equipamentos comerciais para além de disponibilizarem aos centros de investigação uma plataforma de trabalho para a realização das suas missões específicas, realçam a capacidade modular de forma a facilmente se adaptarem aos requisitos do cliente. Um exemplo disso é o AUV Iver3 da empresa OceanServer. Esta empresa dispõe de um modelo base Iver3-580-Standard AUV (Ocean Server Technology, Inc., 2015) que pode ser configurado como mostrado na Figura 7.

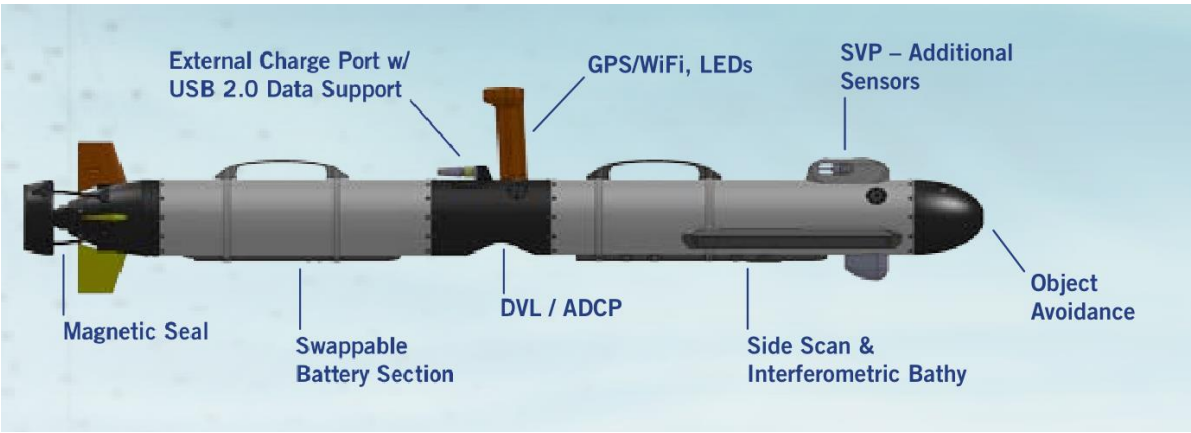

Figura 7 AUV Iver3-580 da OceanServer (Ocean Server Technology, Inc., 2015).

Dada a modularidade do AUV Iver3 a sua utilização pode variar desde o mapeamento topográfico, como demonstrado na Figura 8, até à recolha de imagens subaquáticas.

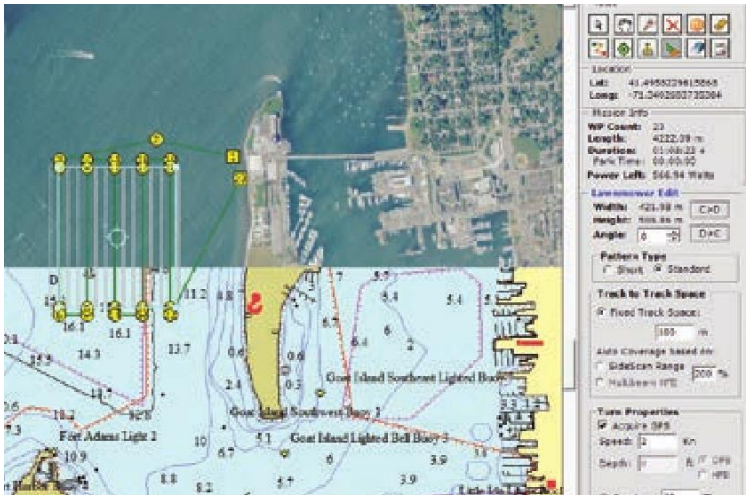

Figura 8 Planeamento de uma missão de mapeamento do AUV IVER3 (Ocean Server Technology, Inc., 2015)

2.1.1.3. Defesa

A literatura em equipamentos militares é escassa, dado o contexto confidencial do seu desenvolvimento. Alguns dos exemplos encontrados baseiam-se na modificação de soluções comerciais. Estes sistemas executam missões como a segurança e proteção de instalações costeiras, acompanhamento de navios, mapeamento de escombros ou deteção de minas. Um exemplo deste tipo de missões é descrito em Charles (2011) com a modificação do AUV Remus 100 para a deteção de explosivos em ambiente marinho utilizando um biossensor, como apresentado na Figura 9.

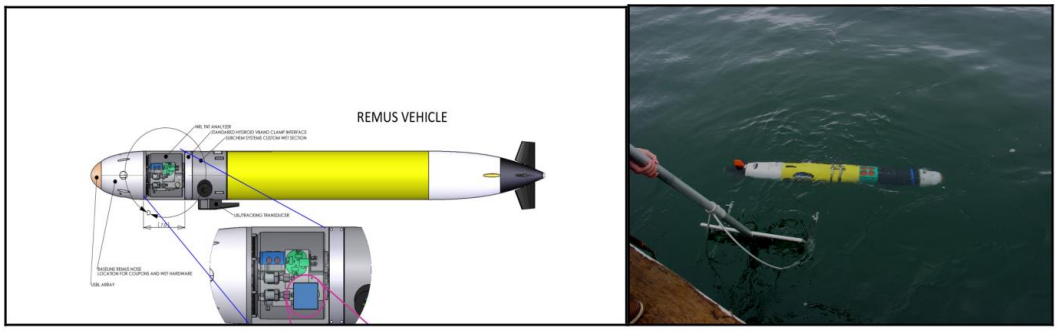

Figura 9 AUV Remus 100 (Charles, et al., 2011).

2.1.2. Classificação de veículos subaquáticos

Os veículos subaquáticos, dadas as múltiplas áreas de ação, podem ser classificados segundo o tipo de missão que executam, a sua geometria, a profundidade a que operam, o material utilizado na construção da fuselagem, a dimensão característica, a velocidade máxima e o tipo de controlo (Ruiz, 2010). De forma a classificar o veículo subaquático em estudo é necessário não só definir as características referidas como também o tipo de controlo, *i.e.*, se esse é remoto, autónomo, ou híbrido (Ruiz, 2010).

2.1.2.1. Veículo subaquático operado remotamente (ROV)

Um ROV é caracterizado pela capacidade de executar tarefas que outrora somente mergulhadores especializados estavam aptos. O primeiro veículo subaquático denominado ROV foi criado pelo cientista Francês Dimitri Rebikoff, em 1953 (Ruiz, 2010). A primeira tentativa de financiamento na tecnologia foi feita por parte da Marinha Norte Americana em 1960 num veículo subaquático de resgate controlado por cabo (*cable-controlled underwater recovery vehicle*, ou CURV). Este sistema estava habilitado a realizar operações de resgate e missões de recolha de objetos do fundo oceânico.

O controlo remoto de um ROV, por parte do operador, é realizado a partir da ligação umbilical à embarcação de apoio, como representado na Figura 10.

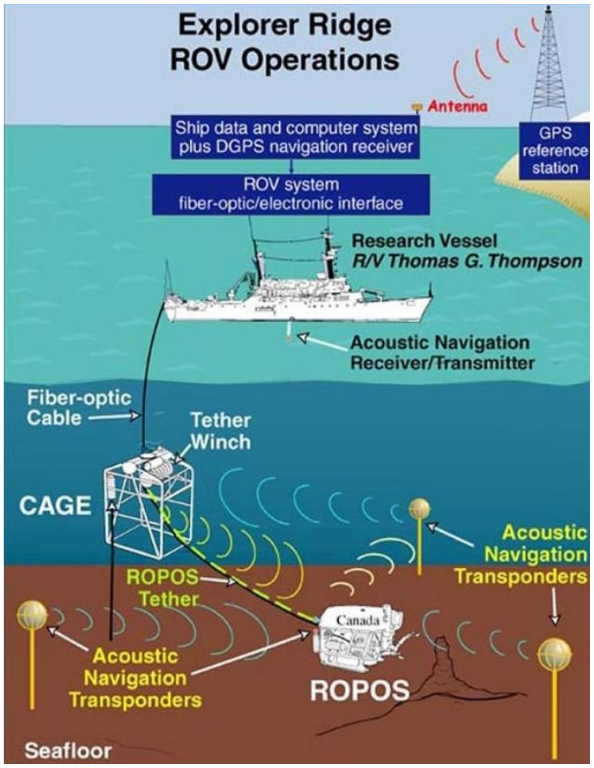

Figura 10 Esquema de utilização de um ROV (National Oceanic and Atmospheric Administration, 2002).

2.1.2.2. Veículo Subaquático Autónomo (AUV)

Um AUV é caracterizado pela capacidade de executar uma missão sem interveniência do operador. O primeiro veículo subaquático classificado como AUV, foi apresentado como um veículo de investigação subaquático concebido para um propósito específico (*special purpose underwater research vehicle*, ou SPURV) (Gafurov e Klochkov, 2015). O SPURV foi desenvolvido nos Estados Unidos da América, em 1957, pelo Laboratório de Física Aplicada da Universidade de Washington. Este sistema foi usado com sucesso em investigação oceanográfica até 1979, provando assim ser uma ferramenta eficaz na exploração oceânica (Gafurov e Klochkov, 2015).

De momento, o desenvolvimento de AUVs é estimulado pela necessidade de se realizarem missões de mapeamento topográfico, caracterização do fundo oceânico, inspeção do casco de navios e aplicações militares, bem como na deteção de minas (Nicholson e Healey, 2008). O grande fator de constrangimento no desenvolvimento de AUV reside na autonomia. Este fator limita não só o alcance de cada missão, como também a quantidade de dados recolhidos (Nicholson e Healey, 2008).

Figura 11 Lançamento do AUV REMUS 100 pela Marinha Norte Americana (Nicholson e Healey, 2008).

2.1.2.3. Veículo Robótico Subaquático Híbrido (HURV)

Como descrito anteriormente, os veículos subaquáticos podem ser divididos em duas grandes áreas: os ROV (remotamente controlados) e os AUV (autónomos). Uma das vantagens da utilização de um AUV incide na ausência de ligações físicas entre o sistema e o veículo de superfície. A desvantagem reside em não ser possível realizar intervenções durante um longo período de funcionamento. Em contrapartida, a vantagem da utilização de um ROV é a capacidade de fazer intervenções durante um longo período de tempo, já que o sistema se encontra fisicamente ligado ao veículo de superfície por um cabo umbilical. A desvantagem deste sistema prende-se pela possibilidade do cordão umbilical ficar preso em algum obstáculo, limitando o alcance da missão a realizar (Xiang et al., 2015). Considerando que ambas as soluções apresentam vantagens e desvantagens, diversos trabalhos mais recentes têm apresentado modelos híbridos com vista a mitigar as referidas desvantagens de cada um (Xiang et al., 2015).

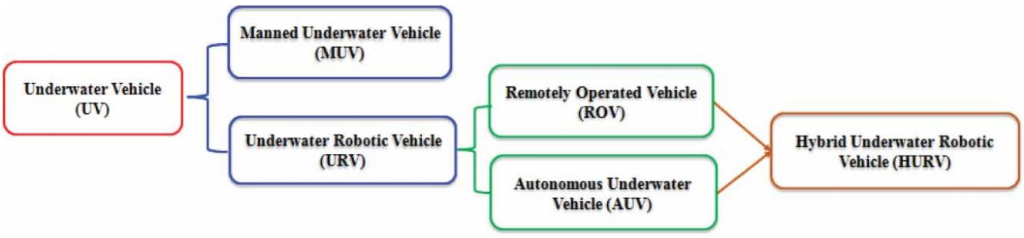

Figura 12 Esquema classificativo do sistema HURV (Xiang et al., 2015).

O sistema multi-veículo, apresentado na Figura 12, classificado como veículo robótico subaquático híbrido (*hybrid underwater robotic vehicles*, ou HURV), combina as vantagens do ROV e do AUV. O HURV providencia assim a capacidade de fazer intervenções durante um longo período de tempo, devido à existência de uma ligação física com o veículo de superfície, podendo este ser, ou não, remotamente controlado pelo operador. Por outro lado, e de forma a aumentar o nível de autonomia em situações pontuais (*e.g.,* acesso a zonas de difícil acesso), o HURV pode, temporariamente, quebrar a ligação física com o veículo de superfície, aumentando o seu nível de mobilidade e autonomia (Xiang et al, 2015).

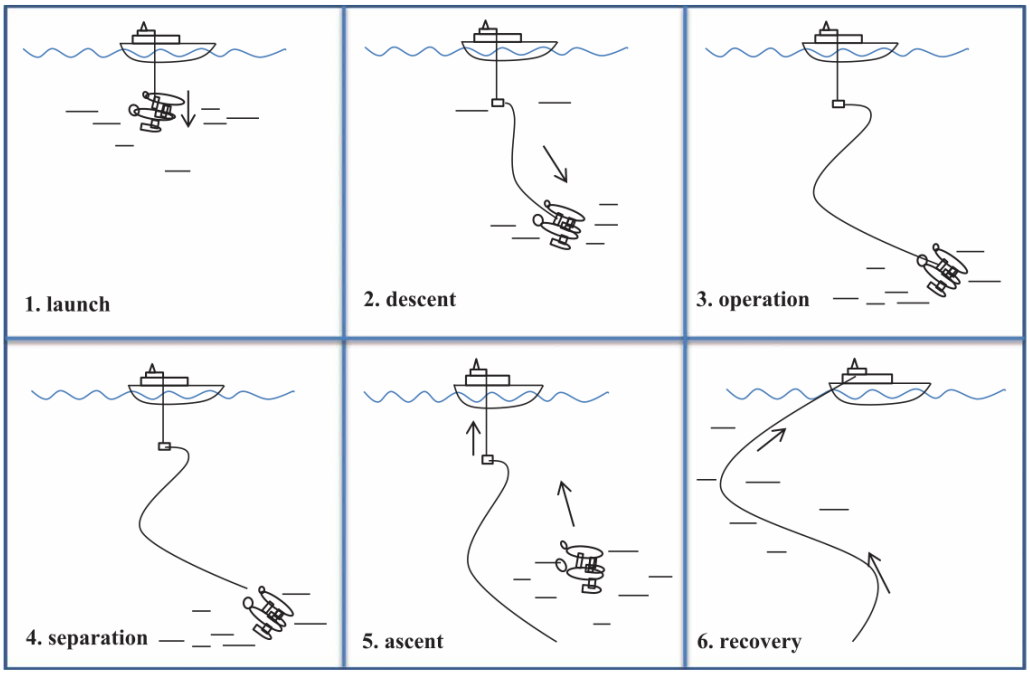

Figura 13 Operação de uma missão como sistema HURV (Xiang et al., 2015).

Conclui-se que o sistema multi-veículo ECHO5 se enquadra na classificação de HURV (Xiang et al., 2015).

2.2. Geometria da Fuselagem

A geometria da fuselagem do veículo subaquático é um fator de extrema importância na sua autonomia (Sarkar et al, 1997). Da literatura encontrada, foi recorrente, por parte de autores como Sousa et al., (2014) e Joung et al., (2009), a utilização das equações de Myring, para definir a geometria da fuselagem de um veículo subaquático.

2.2.1. Modelo geométrico da fuselagem

O modelo geométrico de um veículo subaquático, definido pelas equações de Myring, apresentado na Figura 14, é definido por três secções. O nariz, representado pela equação $r_1(x)$ de comprimento a, a secção central de comprimento b e a cauda representada pela equação $r_2(x)$ de comprimento c (Sousa et al., 2014).

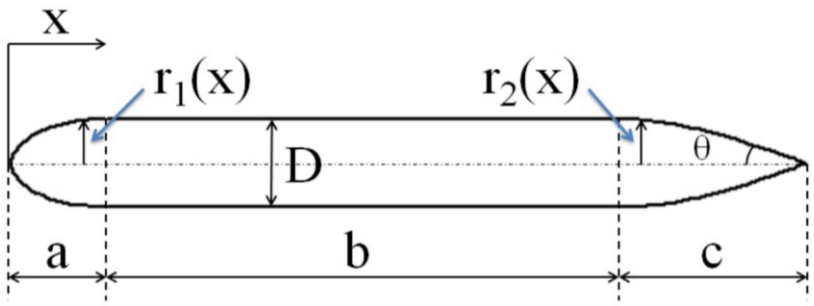

Figura 14 Parâmetros geométricos segundo as equações de Myring (Sousa et al., 2014).

A equação 2.1, que define o nariz do veículo subaquático, depende dos seguintes parâmetros geométricos, diâmetro da fuselagem D, comprimento do nariz a e por um parâmetro não geométrico n.

$$r_1(x) = \frac{1}{2}D[1 - (\frac{x-a}{a})^2]^{1/n} \quad (2.1)$$

A equação 2.2, que define a cauda do veículo subaquático, depende dos seguintes parâmetros geométricos, diâmetro da fuselagem D, comprimento da cauda, c, da secção central b e do nariz a e do ângulo θ.

$$r_2(x) = \frac{1}{2}D - \left[\frac{3D}{2c^2} - \frac{tg\theta}{c^2}\right](x-a-b)^2 + \left[\frac{D}{c^3} - \frac{tg\theta}{c^2}\right](x-a-b)^3 \quad (2.2)$$

Ao variar os parâmetros a, c, n e θ, das equações 2.1 e 2.2, torna possível gerar diferentes configurações do nariz e da cauda de um veículo com um diâmetro fixo D e comprimento fixo b da secção central.

De forma a compreender as diferentes geometrias possíveis com a variação do parâmetro não geométrico n, considerando um comprimento a e um diâmetro D constante, resultam as configurações apresentadas na Figura 15.

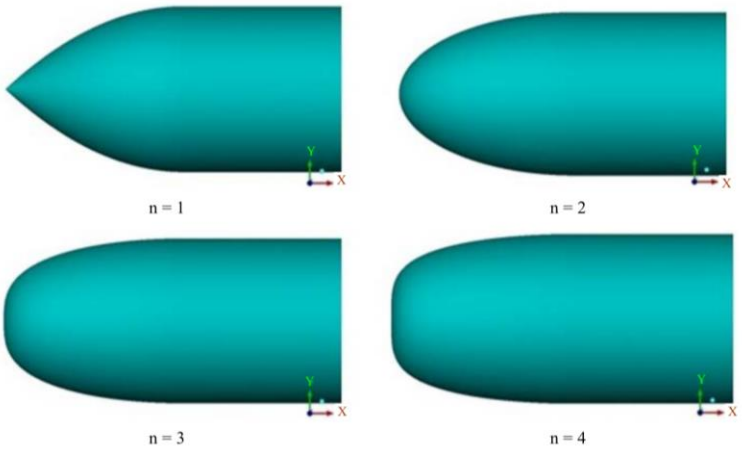

Figura 15 Diferentes curvas do nariz em função de $\boldsymbol{n}$ (Sousa et al., 2014).

Da mesma forma que foram apresentadas configurações do nariz para diferentes valores de n, apresenta-se na Figura 16, configurações da cauda para diferentes valores do parâmetro geométrico θ mantendo constantes o comprimento c e o diâmetro D.

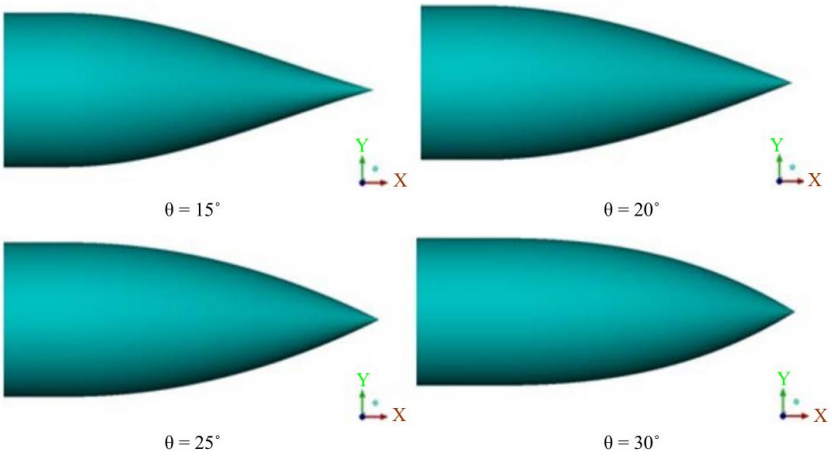

Figura 16 Diferentes curvas da cauda em função de θ (Sousa et al., 2014).

No estudo efetuado por Sousa *et al*., (2014) chegou à conclusão que, de entre os parâmetros $n = \{1, 2, 3, 4\}$ e $\theta = \{15, 20, 25, 30\}$, o coeficiente de arrasto de um veículo subaquático é minimizado para $\theta = 20$ e $n = 2$. Neste caso o coeficiente de arrasto é de $C_d = 0{,}1230$, considerando o comprimento do nariz com $a = 215mm$, a secção central com $b = 1155mm$, a cauda com $c = 430mm$ e diâmetro $D = 200mm$.

A partir dos resultados obtidos por Sousa et al., (2014) adotou-se uma gama de parâmetros para uma fuselagem de $n = \{1{,}5;\ 2{,}5\}$ e de $\theta = \{15°;\ 25°\}$.

3. MODELAÇÃO DA FUSELAGEM

Para a modelação da fuselagem foi utilizado o *software* OpenSCad (OpenSCad, 2015) e o sistema operativo ElementaryOS (ElementaryOs, 2015). Nos capítulos 3.1 e 3.2 é descrita a aplicação das equações de Myring no Software OpenSCad e o controlo da mesma pelo *software* Matlab, respetivamente.

3.1. Aplicação das equações de Myring ao *Software* OpenSCad

O *software* OpenSCad é um programa de código aberto que gera um sólido ou uma superfície a partir de um *script*. Ao contrário de outros *softwares* CAD (Desenho assistido por Computador), o utilizador não tem comandos interativos à sua disposição mas sim uma linguagem de programação, baseada em comandos e funções matemáticas. O utilizador tem duas técnicas de modelação possíveis: o CSG (Geometria Sólida Construtiva) e a função de extrudir geometrias 2D (OpenSCad, 2015).

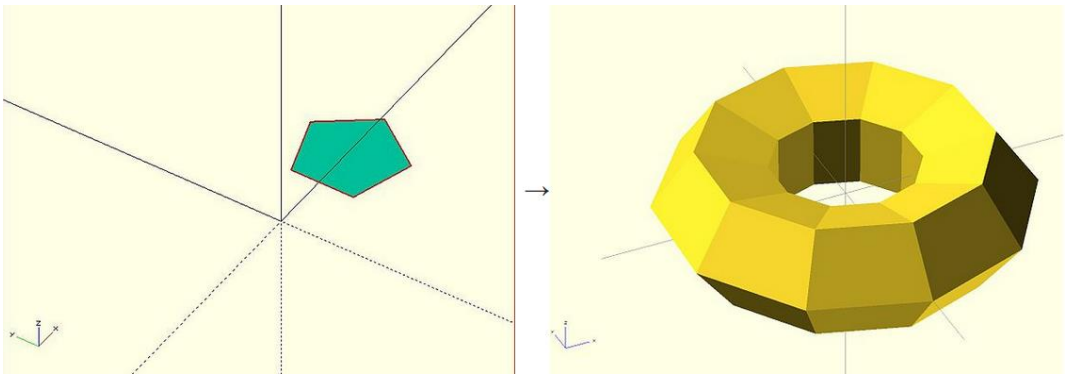

Figura 17 Extrusão de uma geometria 2D em 3D (OpenSCad User Manual, 2015).

O CSG, é uma técnica de modelação baseada em operações booleanas entre sólidos primitivos como, por exemplo, paralelepípedos, cilindros, esferas ou cones. A extrusão de geometrias 2D seja linear ou radial permite a geração de geometrias não primitivas e a utilização de equações matemáticas na definição geometrias 2D (Figura 17). Estas ferramentas conferem ao utilizador a possibilidade de facilmente alterar os parâmetros que definem uma geometria e obter diferentes configurações com simples comandos no terminal.

O processo utilizado para gerar a superfície da fuselagem foi a divisão deste em três secções, nomeadamente, nariz, seção central e cauda.

A geração do nariz e da cauda do veículo subaquático começa pela discretização da função escolhida em polígonos de comprimento, segundo o eixo x, dx. Este processo é tão mais rápido quanto maior for o incremento, dx, escolhido. Em contrapartida se o incremento, dx, for elevado, menor será a qualidade da curva obtida. A escolha do incremento dx foi feita com base na suavidade da superfície por inspeção a olho nu, não obstante de posterior estudo em futuros trabalhos. Na Figura 18 é apresentado o código para a geração da curva do nariz, para isso inicializou-se a equação 2.1 na função $frente(x)$, e definiu-se o início do ciclo com $start = 0$ e o incremento para a criação de um polígono com um incremento dx de:

$$dx = \frac{a - start}{steps} \tag{3.1}$$

em que a representa o comprimento do nariz e $steps$ o incremento definido para a criação da geometria.

```
module frente(steps, a, D, n)
{
        function frente(x)=(0.5)*D*pow((1-pow((x-a)/a,2)),(1/n));
        start=0;
       dx = (a-start)/steps;
        for(x = [start:dx:a])
        {
                rotate([180,0,0])
               translate([0,-a,0])
               rotate([0,180,0])
               rotate([0,0,90])
               polygon( points=[[0,0],[x,frente(x)],[x+dx,frente(x+dx)],[x+dx,0] ]);
        }
}
```

Figura 18 Código para a geração da curva do nariz

Após a inicialização da função $frente(x)$ e respetivas constantes realizou-se um ciclo onde são criados sucessivos polígonos. Estes são translacionados e rodados de forma a posicionarem correntemente a nossa curva, ou seja, no plano em que o *software* OpenSCad executa extrusões radiais. Após o posicionamento do nariz resultou a curva apresentada na Figura 19.

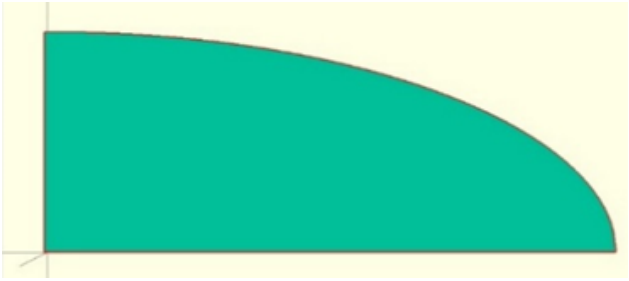

Figura 19 Geometria 2D do nariz.

O código para criação da curva da cauda é apresentado na Figura 20. Realizou-se, de igual forma, a curva do nariz, com a exceção que a função inicializada é a $ftras(x)$. Obteve-se, assim, à curva apresentada na Figura 21.

```
module tras(steps, a, b, c, D, teta)
{
        function ftras(x)=0.5*D-( (3*D)/(2*pow(c,2)) - tan(teta)/c )*pow(x-a-b,2)
        + ( D/(pow(c,3)) - tan(teta)/pow(c,2) )*pow(x-a-b,3);
        start=0;
        dx = (c-start)/steps;
        for(x = [start:dx:c])
        {
                rotate([180,0,0])
                rotate([0,180,0])
                rotate([0,0,90])
                polygon( points=[[0,0],[x,ftras(x)],[x+dx,ftras(x+dx)],[x+dx,0]]);
        }
}
```

Figura 20 Código para a geração da curva da cauda

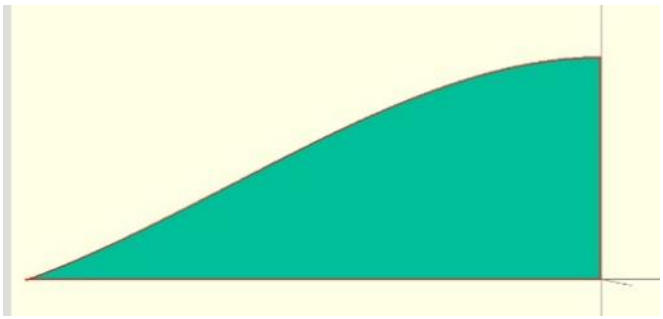

Figura 21 Geometria 2D da cauda.

Após a criação das curvas, que descrevem o nariz e a cauda, são colocaram-se na sua posição final (Figura 22), de forma a realizar uma extrusão radial com um incremento denominado, $stepR$, tal como apresentado na Figura 23. O valor do incremento $stepR$ é escolhido com base na suavidade da superfície por inspeção a olho nu, não obstante de posterior estudo em futuros trabalhos.

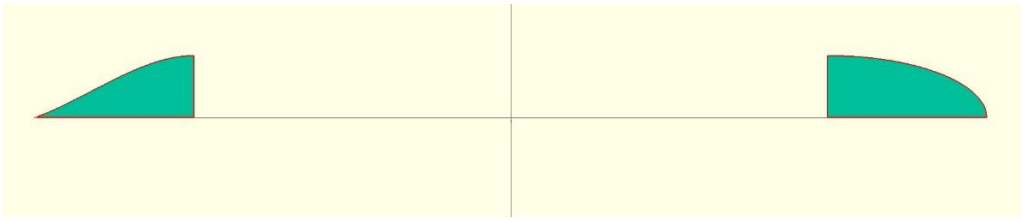

Figura 22 Curvas do nariz e cauda na sua posição final.

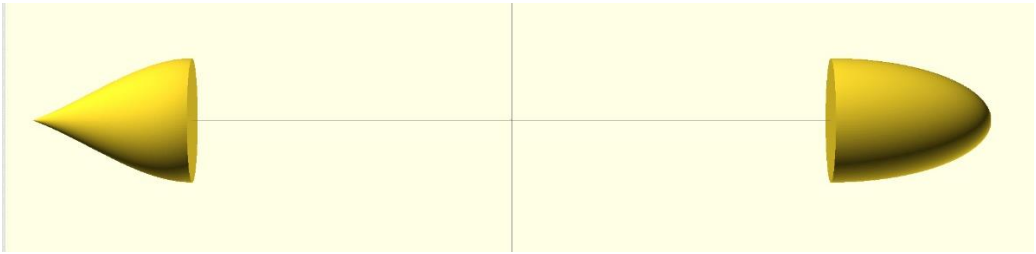

Figura 23 Extrusão radial das curvas de Myring do nariz e da cauda.

Cria-se, a partir da função $cilinder()$, a secção central e transaciona-se para a posição central ao longo do eixo x, como apresentado na Figura 24.

```
module torpedo(stepR, step, a, D, n, b, c, teta, comp)
{

//_____________________Frente_____________________//
   translate([comp/2,0,0])
   rotate([0,90,0])
   rotate_extrude($fn = stepR)
   frente(step, a, D, n);

//_____________________Tubo_____________________//
   translate([-comp/2,0,0])
   rotate([0,90,0])
   cylinder(comp, $fn = stepR)

//_____________________Tras_____________________//
   translate([-comp/2,0,0])
   rotate([0,90,0])
   rotate_extrude($fn = stepR)
   tras(step, 0, b, c, D, teta);
}
```

Figura 24 Código para o posicionamento e extrusão radial do nariz, cauda e secção central.

Após a geração da fuselagem, apresentada na Figura 25, é seccionada em 90° já que o domínio de simulação é um quarto do total, como representado na Figura 26.

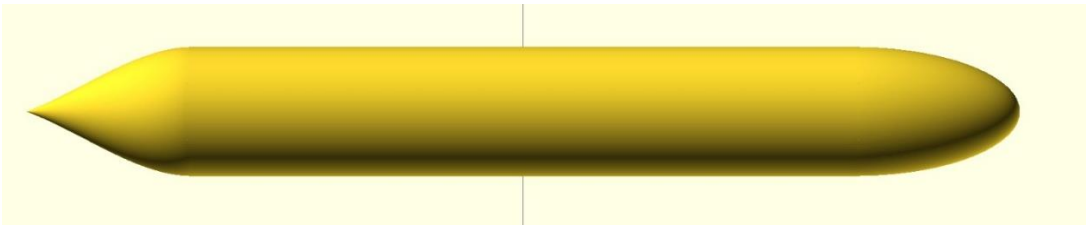

Figura 25 Fuselagem final.

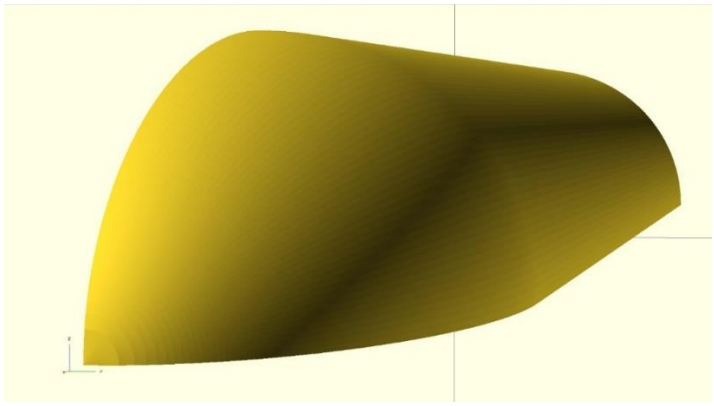

Figura 26 Fuselagem Seccionado a $\mathbf{90}^{\circ}$**.**

Para a criação da fuselagem e geração do ficheiro em formato $.stl$ executou-se no terminal, aberto na diretoria onde se encontra o script $.scad$, o comando apresentado na Figura 27.

```
openscad -o geo1.stl -D 'stepR=200' -D 'step=200' -D 'a=250' -D 'D=155' -D
'n=1.65' -D 'b=0' -D 'c=250' -D 'teta=24' -D 'comp=750' geo.scad
```

Figura 27 Comando para a geração da fuselagem.

3.2. Integração do software OpenSCad no software Matlab

Dado que o processo de otimização será realizado no *software* Matlab é imperativo, para que o processo seja autónomo, que a geração de todos os comandos seja feita pelo Matlab. Para a criação da fuselagem são necessários os parâmetros geométricos $a, D, b, c, teta$ e $comp$, o parâmetro adimensional n e os parâmetros numéricos $stepR$ e $step$. Estes representam respetivamente o incremento radial $stepR$, o incremento em x $step$, o comprimento do nariz a, o diâmetro da fuselagem D, o parâmetro adimensional do nariz

n, o comprimento da secção central b, comprimento da cauda c, o parâmetro da cauda θ e comprimento da secção central $comp$.

Dado que se utiliza um sólido primitivo na criação da secção central, o valor de b será sempre igual a $b = 0$ e o comprimento da secção central é atribuído à variável $comp$. Durante o processo apresentado na Figura 28 utilizaram-se não só os *softwares* Matlab e o OpenSCad como também o software OpenFOAM (OpenCFD Ltd, 2016).

```
[Cf, fx]=run(n, teta)

% _____Criar_Geometria_____
%_____Parametros constantes_____
 stepR=200;
 step=200;
 D=155;
 comp=750;
 a=250;
 c=250;
 b=0;


SstepR = num2str(stepR);
Sstep= num2str(step);
Sa= num2str(a);
SD= num2str(D);
Sn= num2str(n);
Sb= num2str(b);
Sc= num2str(c);
Steta = num2str(teta);
Scomp = num2str(comp);

 command0='cd geo && ';

 command1='openscad -o geo1.stl';

 command2=' geo.scad';

SstepR = strcat(' -D ''stepR=',SstepR,'''');
Sstep  = strcat(' -D ''step=',Sstep,'''');
Sa     = strcat(' -D ''a=',Sa,'''');
SD     = strcat(' -D ''D=',SD,'''');
Sn     = strcat(' -D ''n=',Sn,'''');
Sb     = strcat(' -D ''b=',Sb,'''');
Sc     = strcat(' -D ''c=',Sc,'''');
Steta  = strcat(' -D ''teta=',Steta,'''');
Scomp  = strcat(' -D ''comp=',Scomp,'''');

system(strcat(command0, command1, SstepR, Sstep, Sa,

SD, Sn, Sb, Sc, Steta, Scomp, command2));

system(strcat(command0 , '. /opt/openfoam240/etc/bashrc

&& surfaceTransformPoints -scale ''(0.001 0.001 0.001)'' geo1.stl geo.stl'));
```

Figura 28 Código para geração da fuselagem pelo *software* Matlab.

Criou-se um conjunto de *strings* e três comandos para a geração do ficheiro geo.stl referente à fuselagem. O $command0$ entra na pasta onde está o ficheiro geo.scad, o $command1$ e $command2$ executam o *software* OpenSCad e criam uma fuselagem denominada geo1.stl, respetivamente.

Dado que o *software* OpenSCad trabalha em milímetros e o *software* OpenFOAM em metros, é necessário converter as unidades. Para isso utiliza-se o comando $surfaceTransformPoints$ pertencente ao OpenFOAM que converte o ficheiro geo1.stl $[mm]$ para geo.stl em $[m]$.

4. ANÁLISE NUMÉRICA DA FORÇA E DO COEFICIENTE DE ARRASTO

A análise numérica da força e do coeficiente de arrasto foi executada no *software* OpenFOAM. Este é capaz de, por exemplo, resolver escoamentos com reações químicas, turbulência e transferência de calor (OpenCFD Ltd, 2016).

Os ficheiros necessários para a execução de uma simulação foram criados pelo *software* HELYX-OS (Engys, 2016). Este é um interface gráfico do OpenFOAM que agiliza o processo simulação desde a geração da malha de elementos finitos até a aplicação das condições de fronteira.

Neste capítulo apresenta-se o pré-processamento com as respetivas condições de fronteira, propriedades do fluido e o modelo de turbulência. De forma a validar os parâmetros da malha de elementos finitos, analisaram-se três simulações com diferentes discretizações de um cilindro, de dimensões $D = 0{,}2\ e\ L = 1{,}8$, e comparou-se com o resultado experimental obtido na literatura. Por final analisou-se o código utilizado para realizar um conjunto de simulações de forma autónoma.

4.1. Pré-processamento

De forma a obter uma simulação com condições de fronteira, propriedades do fluido e modelo de turbulência passíveis à variação dos parâmetros geométricos apresentados no capítulo 2.1.1, e obter resultados de coeficiente de arrasto que viabilizem o estudo apresentado no capítulo 5, consideraram-se as condições de fronteira, propriedades do fluido considerado e modelo de turbulência descritos no presente capítulo. Para o fazer comparou-se o resultado experimental do coeficiente de arrasto obtido na literatura para um cilindro de diâmetro $D = 0{,}2$ e de comprimento $L = 1{,}8m$ com a simulação.

4.1.1. Condições de Fronteira

O domínio utilizado na simulação tem as dimensões apresentadas na Tabela 2.

Tabela 2 Geometria do domínio de simulação**.**

	x[m]	y[m]	z[m]
Mínimo	$-2{,}3$	0	0
Máximo	1,9	1,2	1,2

De forma a diminuir o tempo de simulação, simulou-se um quarto do escoamento. Para isso criou-se planos de simetria em ffminz e ffminy, representados na Figura 29.

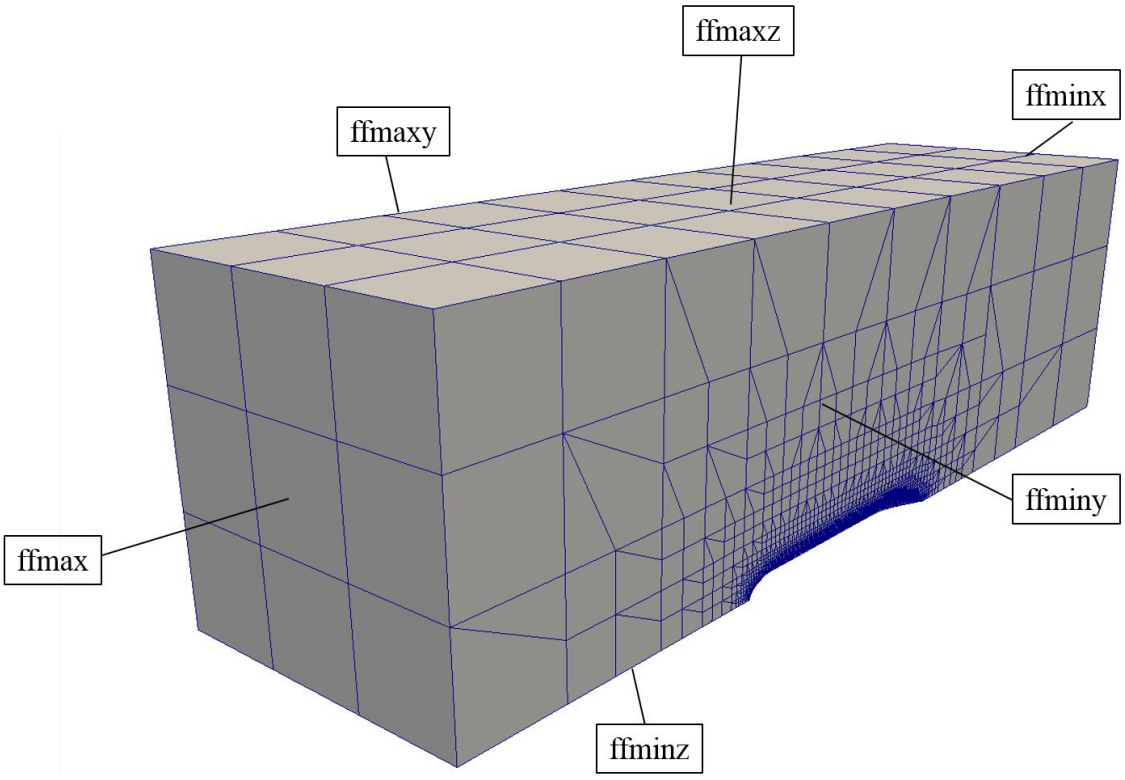

Figura 29 Domínio de simulação**.**

As condições de fronteiras utilizadas em todas as simulações deste trabalho são apresentadas na Figura 29. O plano a montante do cilindro, ffminx tem uma velocidade de escoamento de $-2m/s$ e um gradiente nulo de pressão. O plano a jusante do cilindro tem um gradiente nulo de velocidade e uma pressão de 0. Os *patch*es complanares às faces do cilindro seccionado, ffminy e ffminz são planos de simetria. Os planos restantes ffmaxy e

ffmaxz têm um gradiente nulo de velocidade, denominado com escorregamento na Tabela 3.

Tabela 3 Condições de Fronteira**.**

Patch	Condições de fronteira		Plano de simetria	Parede
	Velocidade [m/s]	Pressão		
ffminx	Gradiente nulo	0		
ffmax	$(-2,0,0)$	Gradiente nulo		
ffminy			✓	
ffmaxy				Com escorregamento
ffminz			✓	
ffmaxz				Com escorregamento
Cilindro				Sem escorregamento

4.1.2. Propriedades do fluido

Utilizaram-se as propriedades da água salgada apresentadas na Tabela 4 (Sousa et al., 2014).

Tabela 4 Propriedades do fluido considerado.

	Densidade $[kg/m^3]$	Viscosidade dinâmica $[Pa.s]$
Água salgada	1027	0,00125

4.1.3. Modelo de Turbulência

No cálculo da força e do coeficiente de arrasto em escoamentos livres, incompressíveis completamente turbulentos, é recomendado por (Sousa et al., 2014) e (Phillips et al., 2007) a utilização do modelo de turbulência SST (*Shear Stress Transport*). Este é uma variante do modelo $k-\omega$, que combina os modelos padrão $k-\omega$ e $k-\varepsilon$ nas zonas mais próximas de parede e mais afastadas, respetivamente. O modelo de turbulência $k-\varepsilon$ utiliza duas variáveis, a energia cinética turbulenta k e a taxa de dissipação da turbulência ε. O modelo de turbulência $k-\omega$ difere do modelo $k-\varepsilon$ já que utiliza uma versão modificada da equação de transporte de k e uma equação de transporte diferente para a taxa de dissipação específica ω (Lima, 2013). Foi utilizado o *solver* simpleFoam com a

capacidade de resolver escoamentos turbulentos em regime permanente de fluidos incompressíveis (OpenFOAM Foundation, 2016) em todas as simulações apresentadas neste trabalho.

4.1.4. Independência da Malha de elementos finitos

A malha de elementos finitos, considerada nas simulações, tem os parâmetros apresentados na Tabela 5.

Tabela 5 Parâmetros da discretização 1.

	$x[m]$	$y[m]$	$z[m]$
Mínimo	$-2{,}3$	0	0
Máximo	1,9	1,2	1,2
Nº de elementos	14	4	4
Tamanho [m]	0,3	0,3	0,3

Aplicou-se um refinamento à superfície do cilindro, máximo e mínimo, de nível 6. Dado que por cada nível de refinamento o elemento é dividido em 4, no nível 6 -se a um tamanho de $0{,}009375m$. Para além do refinamento na superfície do cilindro, não só se aplicou 9 camadas com uma espessura final de 40% do elemento, com um coeficiente de expansão de 1,25 como também se utilizou um refinamento volumétrico em função da distância à superfície, com os parâmetros apresentados na Tabela 6. Os parâmetros para a geração da malha de elementos finitos são guardados no ficheiro *snappyHexMeshDict* do OpenFOAM.

Tabela 6 Refinamento volumétrico considerado em todas as malhas de elementos finitos de simulação.

Distância [m]	Nível
0,02	6
0,06	5
0,12	4
0,3	3
0,5	2

A partir do refinamento, em função da distância ao cilindro, da Tabela 6 obteve-se a discretização apresentada na Figura 30.

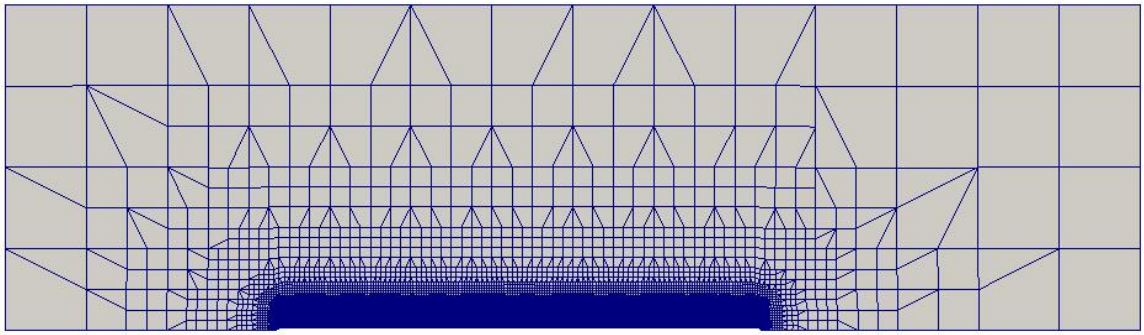

Figura 30 Discretização 1.

A partir das condições de fronteira, propriedades do fluido e da discretização 1, obtiveram-se os resultados apresentados na Tabela 7.

Tabela 7 Resultados obtidos da simulação na discretização 1.

Discretização	Nº de elementos	Tempo $[min]$	$y^+ \, médio$	c_d	$F_{pressão}\,[N]$	$F_{viscosa}\,[N]$
1	207942	14,74	10,9947	0,95058	13,93440	0,97038

O Coeficiente de arrasto C_d é função da força causada pelos efeitos viscosos e pela pressão como apresentado na equação 4.1.

$$C_d = C_{df} + C_{dp} = \frac{F_{viscosa}}{\frac{1}{2}\rho U A_{ref}} + \frac{F_{pressão}}{\frac{1}{2}\rho U A_{ref}} \tag{4.1}$$

O parâmetro adimensional $y^+médio$ relaciona a distância e a velocidade do escoamento no elemento mais próximo da superfície. Encontrou-se na literatura um valor de referência de $y^+ < 2{,}5$ (Eça, 2004). O valor de $y^+ \, médio = 10{,}9947$ não verifica o valor de referência. Dada a necessidade de validar o processo de controlo por parte do *software* Matlab não se otimizou este parâmetro. Não obstante, geraram-se duas outras discretizações como base de comparação. Para tal criou-se a discretização 2 mais refinada e a discretização 3 mais grosseira com os parâmetros apresentados na Tabela 8 e Tabela 9.

Tabela 8 Parâmetros da discretização 2.

	$x[m]$	$y[m]$	$z[m]$
Mínimo	$-2,3$	0	0
Máximo	1,9	1,2	1,2
Nº de elementos	21	6	6
Tamanho [m]	0,2	0,2	0,2

Tabela 9 Parâmetros da discretização 3.

	$x[m]$	$y[m]$	$z[m]$
Mínimo	$-2,3$	0	0
Máximo	1,9	1,2	1,2
Nº de elementos	10	3	3
Tamanho [m]	0,42	0,4	0,4

Considerando os parâmetros de refinamento apresentados na Tabela 6 e as nove camadas junto à superfície do cilindro, obtiveram-se os resultados apresentados na Tabela 10.

Tabela 10 Resultados obtidos na simulação do cilindro para três discretizações diferentes.

Domínio	Nº de elementos	Tempo $[s]$	$y^+ \, médio$	cd	$F_{pressão}\,[N]$	$F_{viscosa}\,[N]$	$F_{arrasto}\,[N]$
1	207942	884,56	10,9947	0,95058	13,93440	0,97038	14,90478
2	572817	4205,47	7,02220	0,94271	13,81277	0,96858	14,78628
3	92943	218,76	15,60447	0,94232	13,80177	0,97352	14,77529

4.2. Validação de resultados

Compararam-se os resultados obtidos com o valor experimental do coeficiente de arrasto de $c_d = 0,85$ (Sousa et al., 2014) referente a um cilindro de dimensões iguais, de onde se obteve os erros percentuais apresentados na Tabela 11. O autor Sousa et al (2014) não só simula uma fuselagem como apresentado no capítulo 2.1.1 como também validou os seus resultados simulando um cilindro nas mesmas condições que a sua fuselagem. Para

validação de resultados aplicou-se a mesma metodologia tendo como prioridade um baixo tempo de comutação.

Tabela 11 Erro percentual dos resultados obtidos em comparação com o resultado experimental.

Domínio	c_d	Erro [%]	Tempo [min]
1	0,95058	10,90706	14,74
2	0,94271	10,86118	70,09
3	0,94232	11,83294	3,646

A partir da Tabela 11 conclui-se que o domínio com a discretização 3, mais grosseira, tem uma pequena diferença de erro em comparação à discretização 2, mais refinada. Dado que o tempo de computação é o fator decisivo considerou-se a discretização 3 em todas as simulações do capítulo 5.

4.3. Obtenção da força e coeficiente de arrasto

Obteve-se a força e coeficiente de arrasto a partir da modificação do ficheiro controlDict. A este adicionou-se todos os parâmetros necessários para o cálculo da força e do coeficiente de atrito para um *patch* selecionado, como apresentado nas Figuras 31 e 32.

```
forceCoeffs
{
    type forceCoeffs;
    functionObjectLibs ( "libforces.so" );
    outputControl timeStep;
    outputInterval 1;
    writeFormat ascii;
    patches ( "cilindro.*" );
    pName p;
    UName U;
    rhoName rhoInf;
    log true;
    liftDir ( 0 0 1);
    dragDir ( -1 0 0);
    CofR ( 0 0 0);
    pitchAxis ( 0 1 0);
    magUInf 2.00;
    rhoInf 1027;
    lRef 1.8;
    Aref 0.00785398163;
}
```

Figura 31 Código para o cálculo do coeficiente de atrito, ficheiro controlDict.

Para o cálculo do coeficiente de atrito, do *patch* cilindro, considerou-se como área de referência $Aref = 0{,}00785398163m^2$, calculada para um quarto da área de um cilindro com um diâmetro de $D = 0{,}2m$, um comprimento de referência $lRef = 1{,}8m$, uma velocidade $magUInf = 2m/s$ e uma densidade de $rhoInf = 1027\text{kg/m}^3$, como apresentado na Figura 31. De igual forma, foi utilizado para o cálculo da força de atrito uma densidade de $rhoInf = 1027\text{kg/m}^3$, como apresentado na Figura 32.

```
forces
{
    type forces;
    functionObjectLibs ( "libforces.so");
    outputControl timeStep;
    outputInterval 1;
    format ascii;
    patches ( "cilindro.*" );
    pName p;
    UName U;
    rhoName rhoInf;
    log true;
    CofR ( 0 0 0);
    rhoInf 1027;
}
```

Figura 32 Código para o cálculo da força de atrito introduzido no ficheiro controlDict.

4.4. Integração do software OpenFOAM no software Matlab

De forma a executar uma simulação de forma completamente autónoma criaram-se os comandos necessários para a geração da fuselagem, a cópia da fuselagem, a pasta de simulação, a geração da malha de elementos finitos, a simulação, a obtenção da força e coeficiente de arrasto e a limpeza da diretoria de simulação, como apresentado no apêndice A, com a arquitetura apresentada na Figura 33.

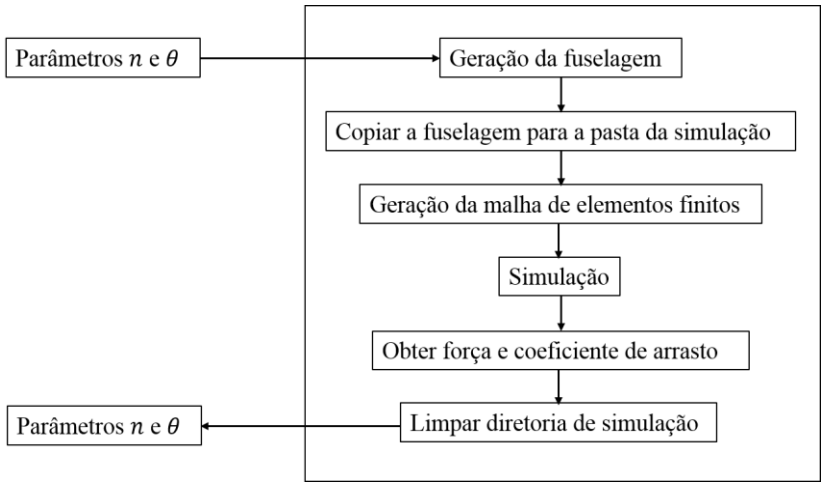

Figura 33 Arquitetura do processo de simulação.

Realizou-se todo o processo na diretoria */matlab* que contém as pastas, apresentadas na Figura 34.

Figura 34 Pastas necessárias para o processo de simulação.

A pasta 0 contém as condições iniciais de pressão, velocidade e os parâmetros de turbulência, como apresentado na Figura 35.

Figura 35 Ficheiros da pasta 0.

A pasta *geo* contém todos os ficheiros necessários para a geração da fuselagem, apresentados na Figura 36.

Figura 36 Ficheiros da pasta geo.

A pasta *sim*, apresentada na Figura 37, é a diretoria onde o OpenFOAM foi executado, contendo: uma pasta denominada 0, com as condições iniciais, apresentada na figura 36; uma pasta denominada *constant*, com todos os parâmetros da malha de elementos finitos e propriedades do modelo de turbulência utilizado, apresentados na Figura 38; uma pasta denominada *Results*, com todos os resultados obtidos, apresentada na Figura 39; e uma pasta denominada *system* com todas as funções utilizadas desde os parâmetros para a geração da malha de elementos finitos até às funções para obter os coeficientes de arrasto, apresentados na Figura 40.

Figura 37 Ficheiros da pasta sim.

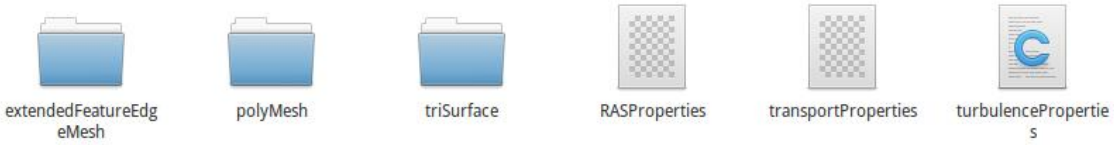

Figura 38 Ficheiros da pasta constant.

Figura 39 Ficheiros da pasta Results.

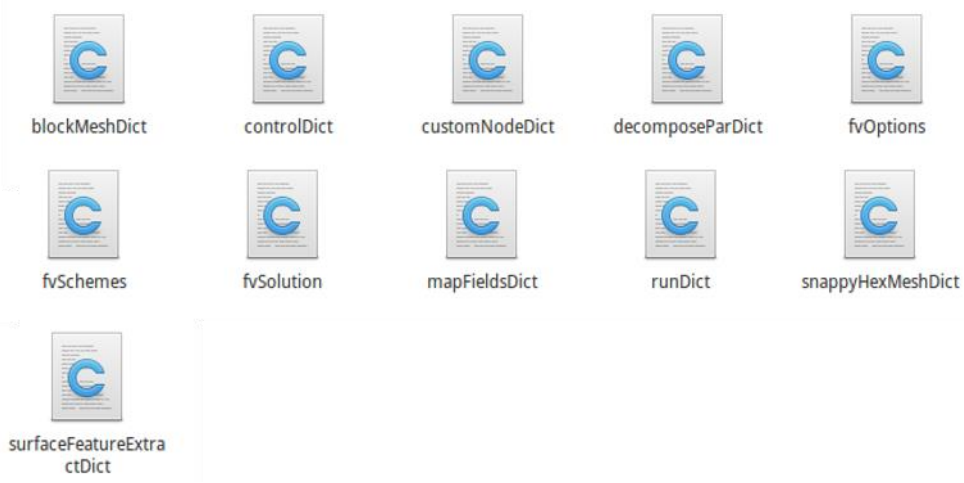

Figura 40 Ficheiros da pasta system.

Para o controlo de todo o processo criaram-se os seguintes *scripts,* apresentados na Figura 41, que se encontram na diretoria /matlab.

Figura 41 Scripts do Matlab para realizar uma simulação.

Os *scripts* importforces.m e importForcesCoeffs.m, mostrados no apêndice A e B, criados em Matlab, obtêm as forças e coeficientes de arrasto dos ficheiros do tipo .dat criados pelo OpenFOAM. O *script* run.m corre todo o processo apresentado na Figura 33. O *script* simu.m controla a entrada dos parâmetros n e θ e guarda os resultados.

O processo apresentado na Figura 33 é descrito em detalhe nos subcapítulos seguintes. Dado que a geração da fuselagem foi explicada no capítulo 3.1 começou-se pelo processo: copiar a fuselagem para a pasta da simulação.

4.4.1. Copiar a fuselagem para a pasta da simulação

Após a geração da fuselagem, copiou-se a da diretoria /geo para sim/constant/trisurface, como apresentado na Figura 42.

```
%_____________________Copiar geometria para pasta da simulação_____

system('cp geo/geo.stl sim/constant/triSurface/');
```

Figura 42 Código para copiar a fuselagem para a pasta da simulação.

4.4.2. Geração da Malha de Elementos Finitos

O processo de discretização é realizado por um conjunto de comandos, apresentados na Figura 43.

```
%_____________________________Criação da Malha_____________________________

 command3=' cd sim && . /opt/openfoam240/etc/bashrc && ';
 system('mv sim/0 sim/0.org');
 system('mkdir sim/0');

 system(strcat(command3,' blockMesh'));
 system(strcat(command3,' surfaceFeatureExtract '))
 system(strcat(command3,'decomposePar -force'));
 system(strcat(command3,'mpirun -np 4 snappyHexMesh -overwrite -parallel'))
 system(strcat(command3,'reconstructParMesh -constant'));

system(' rm -r sim/processor0');
system(' rm -r sim/processor1');
system(' rm -r sim/processor2');
system(' rm -r sim/processor3');
system(' rm -r sim/0');

system('mv sim/0.org sim/0')
system(strcat(command3,'decomposePar -force'));
system(strcat(command3, 'mpirun -n 4 renumberMesh -overwrite -parallel'));
```

Figura 43 Código para a geração da malha de elementos finitos.

De forma a facilitar todo o processo criou-se um comando denominado $command3$ que entra na diretoria de simulação e inicializa o OpenFOAM.

Em seguida renomeou-se a pasta 0 para 0.org de forma a não interferir com o processo de geração da malha de elementos finitos e gerou-se o domínio base, a partir do comando blockMesh, com os parâmetros apresentados na Tabela 5. Logo depois separou-se o domínio pelos quatro processadores, com o comando decomposePar –force e extraiu-se o perfil da superfície, da fuselagem seccionada a 90°, de forma a obtermos as 9 camadas de

forma mais congruente a partir do comando surfaceFeatureExtract. É então criada a malha de elementos finitos, em paralelo, com os parâmetros referidos no capítulo 4.1.4.

De forma a evitar erros, agregaram-se as malhas de elementos finitos criadas em paralelo, numa só, apagou-se todas as diretorias dos processadores e renomeou-se a pasta 0.org para 0. Por fim, dividiu-se a malha de elementos finitos pelos quatro processadores para que as simulações se realizem em paralelo.

4.4.3. Simulação

De forma a facilitar a execução dos comandos necessários criou-se a variável $command4$ que gera uma *string* com o nome da simulação. A simulação realizou-se em paralelo a partir do comando mpirun.

Por fim, agregou-se os resultados dos vários processadores e calculou-se o parâmetro y^+ com o comando yPlusRAS. Os resultados deste guardaram-se no ficheiro, denominado yPlusRAS_ e o nome da simulação criado pelo comando4, como apresentado na Figura 44.

```
%%_____________________________simulação_____________________________

command4=strcat('SR',num2str(stepR),'S',num2str(step),'a',num2str(a),
system('chmod 777 sim');
system(strcat(command3,'mpirun -np 4 simpleFoam -parallel'));
%system(strcat(command3,'mpirun -np 4 simpleFoam -parallel 2>','log_',
%system(strcat(command3,32,'tail -f',32,command4));
system('chmod 777 sim');
system(strcat(command3, ' reconstructParMesh -constant'));
system(strcat(command3, ' reconstructPar '));
system(strcat(command3,'yPlusRAS>','yPlusRAS_',command4));
```

Figura 44 Código para realizar uma simulação em paralelo e obter os resultados de y^+.

4.4.4. Obter a Força e o Coeficiente de Arrasto

De forma a obter-se a força e o coeficiente de arrasto, criaram-se dois *scripts,* que ao serem copiados para a pasta de pós-processamento, retiram o valor da última iteração. Dado que todas as simulações apresentadas na Tabela 10 convergiram antes da milésima iteração foi considerado que se a solução não for divergente não se realizam mais de 2000 iterações, contudo consideração anterior não é extrapolável para simulações com parâmetros diferentes. Assim sendo, o Matlab, ao executar a função importForcesCoeffs, devolve o último valor do coeficiente de arrasto. De igual forma a função importForces devolve o

último valor de força, referente aos efeitos viscosos. Foi utilizado o comando command4 para adicionar ao nome dos ficheiros forces e forceCoeffs o nome da respetiva simulação.

Os três últimos comandos da Figura 45 copiam os ficheiros de força, de coeficiente de arrasto e de y^+ para uma pasta com o nome da simulação, dentro da pasta *Results*.

```
%%_____Retirar Força de Arrasto e Coeficiente de arrasto_____

system('chmod 777 sim/postProcessing/forceCoeffs/0');
system('chmod 777 sim/postProcessing/forces/0');

system(' cp importForces.m       sim/postProcessing/forces/0/');
system(' cp importForcesCoeffs.m sim/postProcessing/forceCoeffs/0');
cd sim/postProcessing/forceCoeffs/0;
forceCoeffs = importForcesCoeffs('forceCoeffs.dat', 1, 2000);
t=forceCoeffs(end, 1);
Cf=forceCoeffs(end, 3);
cd ..
cd ..
cd forces/0/
forces=importForces('forces.dat', 1, 2000);
fx=forces(end);
fx=cell2mat(fx);
system(strcat('mv forces.dat',32,'forces_',command4,'.dat'));
cd ..
cd ..
cd forceCoeffs/0/
system(strcat('mv forceCoeffs.dat',32,'forceCoeffs_',command4,'.dat'));
cd ..
cd ..
cd ..
system(strcat('mv',32,'postProcessing/forces/0/','forces_',command4,
'.dat',32, 'Results/',command4,'/','forces_',command4,'.dat'));
system(strcat('mv',32,'postProcessing/forceCoeffs/0/','forceCoeffs_',
command4,'.dat',32, 'Results/',command4,'/','forceCoeffs_',command4,'.dat'));
system(strcat('mv',32,'yPlusRAS_',command4,32, 'Results/',command4,'/',
'/','yPlusRAS_',command4));
```

Figura 45 Código para retirar a Força, o Coeficiente de Arrasto e o y+.

4.4.5. Limpar Diretoria de Simulação

Para que o processo possa ser repetido é necessário repor a pasta sim à sua formatação inicial. Para isso foi criado um ciclo que lê os nomes das diretorias e elimina todas as pastas e ficheiros que não sejam necessários, para a próxima simulação, como apresentado na Figura 46. No caso da necessidade de rever todos os ficheiros de todas as simulações é possível copia-los para a pasta Results com o mesmo método apresentado na Figura 45.

```
%_____Apagar pastas_____
system(' rm -r postProcessing');
system(' rm -r processor0');
system(' rm -r processor1');
system(' rm -r processor2');
system(' rm -r processor3');
system(' rm -r 0');

cd ..
s=dir('sim');
d=size(s);
d=d(1);

    n=1:d
name=str2num(s(n,1).name);
       isnumeric(name)==1 || strcmp('NaN', name)==1
             name~=0
            system(strcat('cd sim &&',32,'rm -r',32, num2str(name)));
            system('cd ..');

system(' cp -r 0 sim/');
```

Figura 46 Código para apagar todas as pastas e ficheiros não necessários para a simulação seguinte.

5. RESULTADOS

5.1. Estudo numérico da fuselagem do veículo subaquático

Como base de estudo, dos parâmetros necessários à caracterização da fuselagem, consideraram-se constantes, devido a constrangimentos de projeto, os parâmetros geométricos: $a = 250mm, D = 155mm, b = 0, c = 250mm,\ comp = 750mm,$ e os parâmetros numéricos $stepR = 200$ e $step = 200$. Simularam-se várias configurações da fuselagem variando os parâmetros n e θ entre n={1,5;3,5} e $\theta = \{15; 25\}$, para os quais se obtiveram os resultados do anexo B, apresentados graficamente na Figura 47.

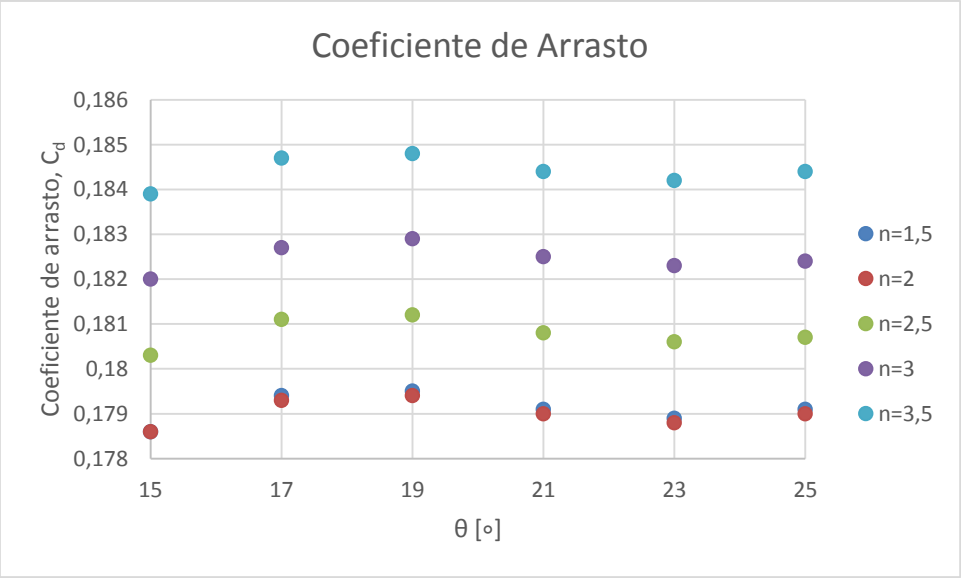

Figura 47 Coeficiente de arrasto obtido para diferentes valores de n e $\boldsymbol{\theta}$.

A partir da Figura 47 conclui-se que o menor coeficiente de arrasto obtido ocorre para valores de $\theta = 23°$e $\theta = 15°$ e $n = [1{,}5; 2]$. A partir dos valores destes simulou-se, para um valor fixo de $\theta = 23°$, a variação do parâmetro n entre $n = \{1{,}5; 1{,}7\}$. Deste modo, obtiveram-se os resultados apresentados na Figura 48.

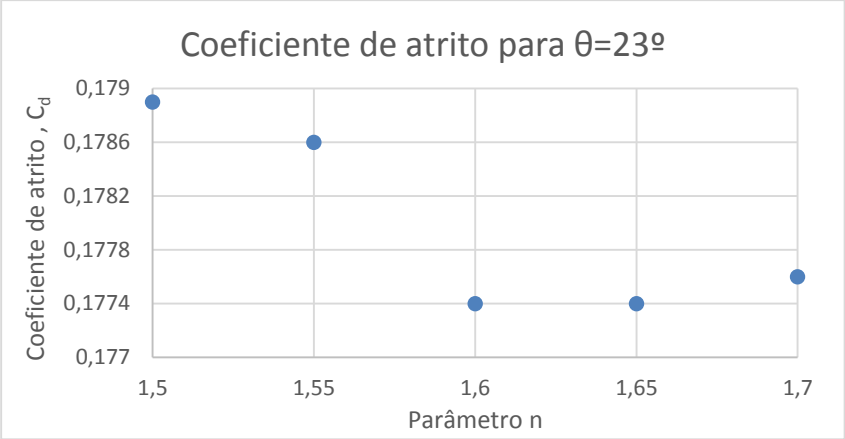

Figura 48 Coeficiente de arrasto obtido para diferentes valores de n com $\boldsymbol{\theta = 23º}$.

O menor valor de coeficiente de arrasto, é obtido para valores de $\theta = 23º$ e $n = 1{,}6$ e $n = 1{,}65$ de $C_d = 0{,}1774$. A partir deste resultado escolheu-se um parâmetro $n = 1{,}65$. Em seguida variou-se o ângulo θ entre $\theta = \{21; 25\}$ e obtiveram-se os resultados apresentados na Figura 49.

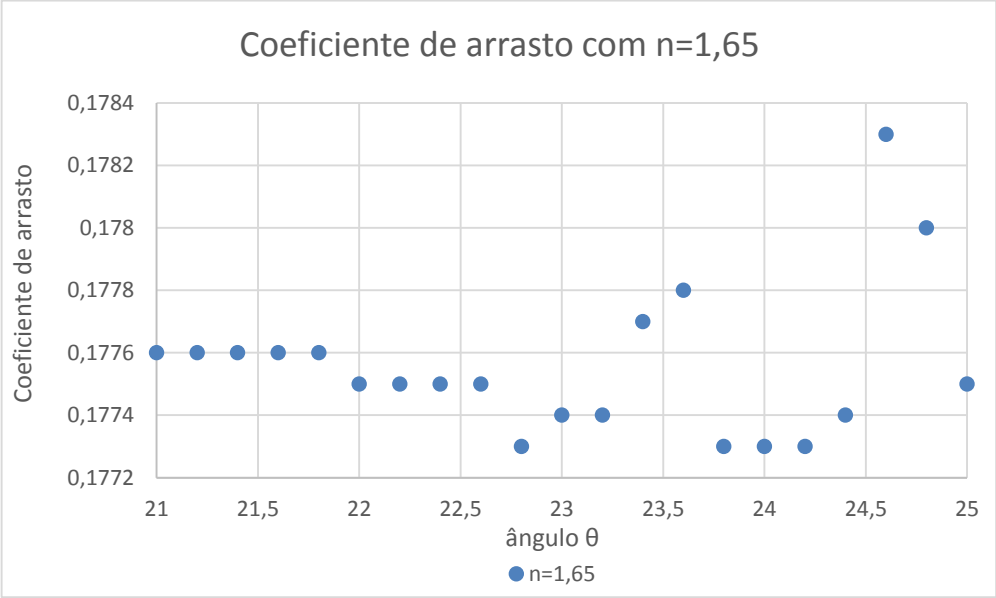

Figura 49 Coeficiente de arrasto para valores de n= [21;25].

Dada a pequena amplitude de resultados de coeficiente de arrasto apresentados na Figura 49 escolheu-se um valor de $\theta = 24$ e um $n = 1{,}65$ como possíveis valores para a fuselagem do veículo subaquático.

5.2. Fuselagem do veículo subaquático final

Após o estudo do capítulo 5.1 escolheu-se, como possíveis parâmetros para a fuselagem do veículo subaquático, os apresentados na Tabela 12.

Tabela 12 Parâmetros finais da fuselagem do veículo subaquático**.**

$a\ [mm]$	250
$seção\ central\ [mm]$	750
$c\ [mm]$	250
$D[mm]$	155
$n[mm]$	1,65
$\theta[º]$	24
$\boldsymbol{C_D}$	0,17728
$\boldsymbol{f_{pressão}}\ [\boldsymbol{N}]$	0,47019
$\boldsymbol{f_{viscosa}}\ [\boldsymbol{N}]$	1,24768
$\boldsymbol{f_{arrasto}}\ [\boldsymbol{N}]$	1,71787
$\boldsymbol{y+}$	16,95

A aplicação dos parâmetros do domínio 3 na fuselagem final resultou na malha de elementos finitos apresentada na Figura 50.

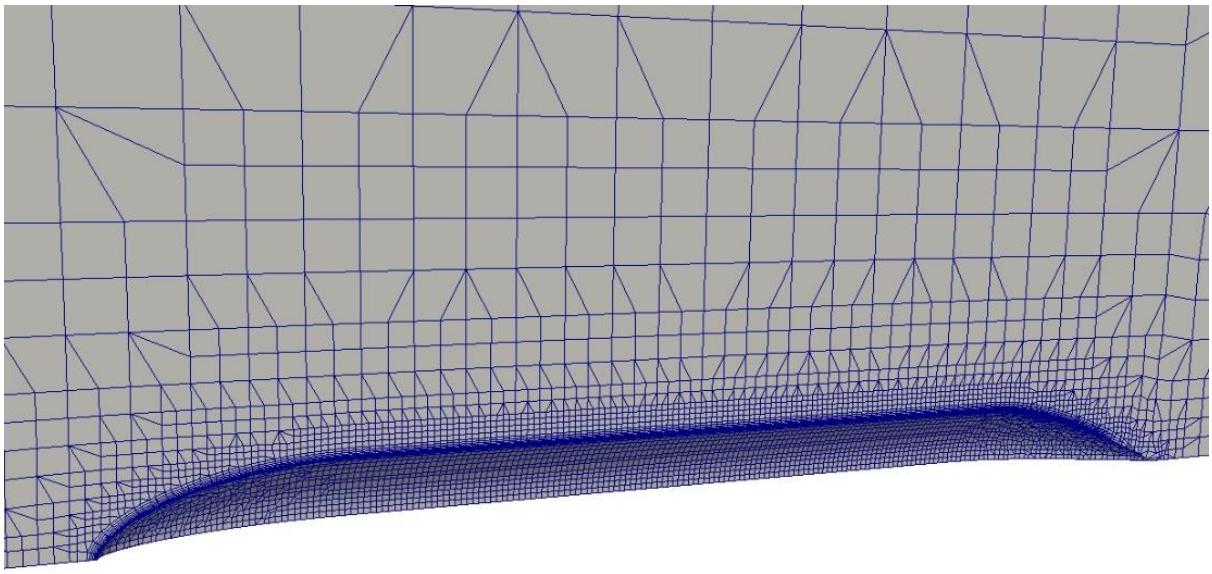

Figura 50 Representação da malha de elementos finitos da fuselagem final.

Na Figura 51 pode se constatar o refinamento da malhe em nove camadas.

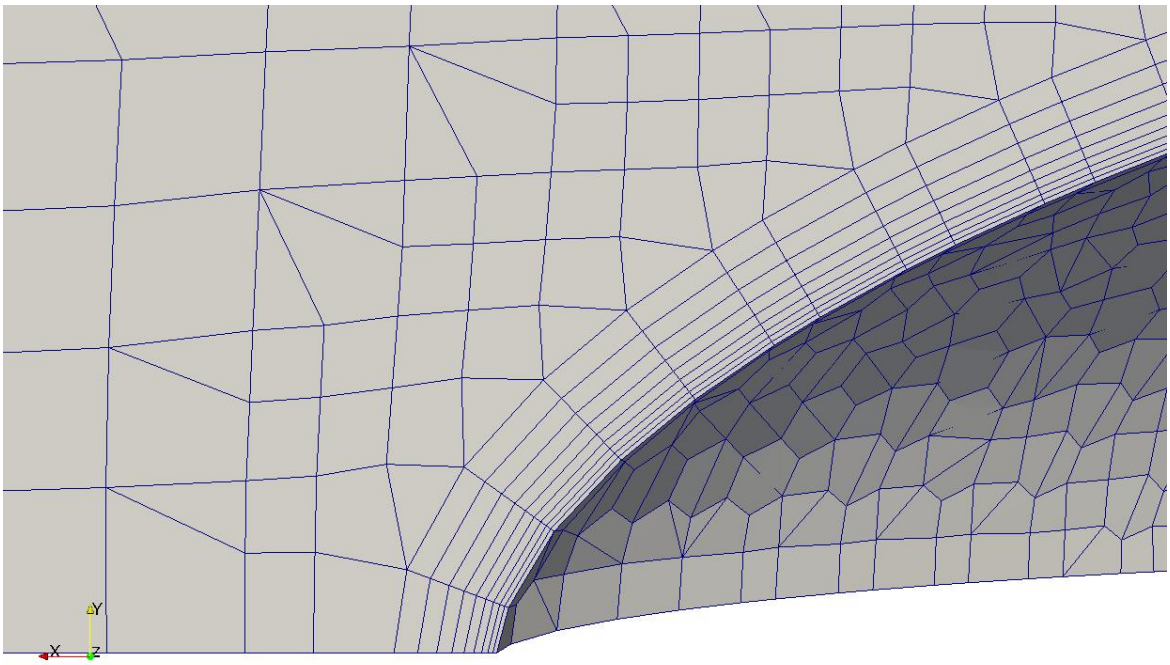

Figura 51 Refinamento de malha de elementos finitos e camadas.

6. CONCLUSÕES

A partir do estudo do estado da arte de veículos subaquáticos não só se concluiu que o veículo subaquático em estudo pertence à categoria de sistemas híbridos a integrar numa área de ação comercial como também se mostrou um panorama das soluções apresentadas pelo mercado a nível global.

A partir da integração dos *softwares* OpenFOAM, OpenSCad e Matlab, tornou-se possível a realização de um conjunto de simulações para diferentes fuselagens de forma autónoma. A partir desta capacidade, não só efetuou-se um estudo à hidrodinâmica da fuselagem do veículo subaquático no qual se obteve uma possível fuselagem, como também abriu-se a porta à possibilidade de otimização. A partir dos scripts em Matlab desenvolvidos, é possível controlar todos os parâmetros da fuselagem. No futuro, ambiciona-se desenvolver novos *scripts* em Matlab para modificar os parâmetros de pré-processamento e processamento do OpenFOAM. Desta forma será possível variar, de forma autónoma, o modelo de turbulência, e os parâmetros da malha de elementos finitos.

7. REFERÊNCIAS BIBLIOGRÁFICAS

Charles, P., Adams, A., Deschamps, J., Veitch, S., Hanson, A., e Kusterbeck, A. (2011). Explosives detection in the marine environment using UUV-modified immunosensor. *Chemical, Biological, Radiological, Nuclear, and Explosives (CBRNE) Sensing XII*.

Desa, E., e Maya Team. (2007). The small AUV Maya: Initial Field Results.

Eça, L. (2004). *On The Grid Sensitivity Of The Wall Boundary Condition Of The k − ω Turbulence Model.*

ElementaryOs. (2015). *ElementaryOs*. Obtido de https://elementary.io

Engys. (2016). Obtido de http://engys.com/products/helyx-os

Gafurov, S., e Klochkov, E. (2015). Autonomous unmanned underwater vehicles development tendencies. *Procedia Engineering*, 141 – 148.

Ingeniarius Lda. (2016). Obtido de http://www.ingeniarius.pt/

Joung, T., Sammut, K., He, F., e Lee, S.-K. (2009). A Study on the Design Optimization of an AUV by Using Computational Fluid Dynamic Analysis. *Proceedings of the Nineteenth International Offshore and Polar Engineering Conference*, 696-702.

Lim, H., MatJafri, M., e Abdullah, K. (2009). Monitoring Turbidity in Prai River Estuary Using Digital Camera Imagery. *International Conference on IT to Celebrate S. Charmonman's 72nd Birthday*, 14.1-14.7.

Lima, R. (2013). *Avaliação de modelos de turbulência na previsão do desempenho de ejetores.*

Marsh, L., Copley, J. T., Huvenne, V. A., e Tyler, P. A. (2013). Getting the bigger picture: Using precision Remotely Operated Vehicle (ROV) videography to acquire high-definition mosaic images of newly discovered hydrothermal vents in the Southern Ocean. *Deep-Sea Research II*, 124–135.

Mascarenhas, A., Afzulpurkar, S., Navelkar, G., Maurya, P., Fernandes, L., Sa, E., . . . Dias, A. (2015). Use of Marine Robot 'MAYA' for Monitoring the Zuari and Mandovi Estuarine Systems. *Dynamics of the Indian Ocean: Perspective and Retrospective: Proceedings of International Symposium on Indian Ocean.*

Matereospace Lda. (2016). Coimbra. Obtido de http://matereo.com/

National Oceanic and Atmospheric Administration. (10 de Janeiro de 2002). Obtido de http://oceanexplorer.noaa.gov/explorations/02fire/background/rovs_auvs/media/rovops.html

Nicholson, J., e Healey, A. (2008). The Present State of Autonomous Underwater Vehicle (AUV) Applications and Technologies. *Marine Technology Society Journal*, 44-51.

Ocean Server Technology, Inc. (2015). *Iver-auv.* Obtido de http://www.iver-auv.com/iver3S.html

OpenCFD Ltd. (2016). Obtido de http://www.openfoam.com/

OpenFOAM Foundation. (2016). Obtido de http://www.openfoam.org/features/standard-solvers.php

OpenSCad. (2015). *OpenSCad*. Obtido de http://www.openscad.org

OpenSCad User Manual. (2015). Obtido de https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Using_the_2D_Subsystem

Phillips, A., Furlong, M., e Turnock , S. (2007). *The Use of Computational Fluid Dynsmics to Assess the Hull Resistence of Concept Autonomous Underwater Vehicles.*

Prammer. (3 de Novembro de 1998). DIRECTIVA 98/83/CE. *Qualidade da água destinada ao consumo humano*. Bruxelas.

Ruiz, D. (2010). *Estado Del Arte En Robotica Submarina.*

Sahu, B., e Subudhi, B. (2014). The State of Art of Autonomous Underwater Vehicles in Current and Future Decades.

Sarkar, T., Sayer, P., e Fraser, S. (1997). A Study Of Autonomous Underwater Vehicle Hull Forms Using Computational Fluid Dynamics. *International Journal For Numerical Methods In Fluids, Vol. 25*, 1301–1313.

Singh, H., Eustice, R., Roman, C., e Pizarro, O. (2002). The SeaBED AUV – A Platform for High Resolution Imaging.

Sousa, J., Macêdo, A., Junior, W., e Lima, A. (2014). Numerical Analysis of Turbulent Fluid Flow and Drag Coefficient for Optimizing the AUV Hull Design. *Open Journal of Fluid Dynamics*, 263-277.

Xiang, X., Niu, Z., Lapierre, L., e Zuo, M. (2015). Hybrid underwater robotic vehicles: the state-of-the-art and future trends. *HKIE Transactions*, 103–116.

ANEXO A – CÓDIGO DO SCRIPT RUN.M

```
function [Cf, fx]=run(n, teta)


% _______________Criar_Geometria_____________________________
%________Parametros constantes_____________________
 stepR=200;
 step=200;
 D=155;
 comp=750;
 a=250;
 c=250;
 b=0;

SstepR = num2str(stepR);
Sstep= num2str(step);
Sa= num2str(a);
SD= num2str(D);
Sn= num2str(n);
Sb= num2str(b);
Sc= num2str(c);
Steta = num2str(teta);
Scomp = num2str(comp);

 command0='cd geo && ';

 command1='openscad -o geo1.stl';

 command2=' geo.scad';

SstepR = strcat(' -D "stepR=',SstepR,"");
Sstep  = strcat(' -D "step=',Sstep,"");
Sa     = strcat(' -D "a=',Sa,"");
SD     = strcat(' -D "D=',SD,"");
Sn     = strcat(' -D "n=',Sn,"");
Sb     = strcat(' -D "b=',Sb,"");
Sc     = strcat(' -D "c=',Sc,"");
Steta  = strcat(' -D "teta=',Steta,"");
Scomp  = strcat(' -D "comp=',Scomp,"");

system(strcat(command0, command1, SstepR, Sstep, Sa, SD, Sn, Sb, Sc, Steta, Scomp,
command2));
```

```
system(strcat(command0 , '. /opt/openfoam240/etc/bashrc && surfaceTransformPoints -
scale "(0.001 0.001 0.001)" geo1.stl geo.stl'));

%_____Copiar geometria para pasta da simulação_____

system('cp geo/geo.stl sim/constant/triSurface/');

%_____Criação da Malha_____

 command3=' cd sim && . /opt/openfoam240/etc/bashrc && ';
 system('mv sim/0 sim/0.org');
 system('mkdir sim/0');

 system(strcat(command3,' blockMesh'));
 system(strcat(command3,' surfaceFeatureExtract '))
 system(strcat(command3,'decomposePar -force'));
 system(strcat(command3,'mpirun -np 4 snappyHexMesh -overwrite -parallel'));
 system(strcat(command3,'reconstructParMesh -constant'));

system(' rm -r sim/processor0');
system(' rm -r sim/processor1');
system(' rm -r sim/processor2');
system(' rm -r sim/processor3');
system(' rm -r sim/0');

system('mv sim/0.org sim/0')
system(strcat(command3,'decomposePar -force'));
system(strcat(command3, 'mpirun -n 4 renumberMesh -overwrite -parallel'));

%%_____simulação_____

command4=strcat('SR',num2str(stepR),'S',num2str(step),'a',num2str(a),
'd',num2str(D),'n',num2str(n),'c',num2str(c),'t',num2str(teta),'comp',num2str(comp));
system('chmod 777 sim');
system(strcat(command3,'mpirun -np 4 simpleFoam -parallel'));
%system(strcat(command3,'mpirun -np 4 simpleFoam -parallel 2>','log_',command4,32,' |
tee -a'));
%system(strcat(command3,32,'tail -f',32,command4));
system('chmod 777 sim');
system(strcat(command3, ' reconstructParMesh -constant'));
system(strcat(command3, ' reconstructPar '));
system(strcat(command3,'yPlusRAS>','yPlusRAS_',command4));

%%_____Retirar Força de Arrasto e Coeficiente de arrasto_____

 system('chmod 777 sim/postProcessing/forceCoeffs/0');
 system('chmod 777 sim/postProcessing/forces/0');
```

```
 system(' cp importForces.m       sim/postProcessing/forces/0/');
 system(' cp importForcesCoeffs.m sim/postProcessing/forceCoeffs/0');
 cd sim/postProcessing/forceCoeffs/0;
 forceCoeffs = importForcesCoeffs('forceCoeffs.dat', 1, 2000);
 t=forceCoeffs(end, 1);
 Cf=forceCoeffs(end, 3);
 cd ..
 cd ..
 cd forces/0/
 forces=importForces('forces.dat', 1, 2000);
 fx=forces(end);
 fx=cell2mat(fx);
 system(strcat('mv forces.dat',32,'forces_',command4,'.dat'));
 cd ..
 cd ..
 cd forceCoeffs/0/
 system(strcat('mv forceCoeffs.dat',32,'forceCoeffs_',command4,'.dat'));
 cd ..
 cd ..
 cd ..
 system(strcat('mkdir Results/', command4));
 system(strcat('mv',32,'postProcessing/forces/0/','forces_',command4,'.dat',32,
'Results/',command4,'/','forces_',command4,'.dat'));
 system(strcat('mv',32,'postProcessing/forceCoeffs/0/','forceCoeffs_',command4,'.dat',32,
'Results/',command4,'/','forceCoeffs_',command4,'.dat'));
 system(strcat('mv',32,'yPlusRAS_',command4,32,
'Results/',command4,'/','yPlusRAS_',command4));
% system(strcat('mv',32,'log_',command4,32, 'Results/',command4,'/','log_',command4));

%______Apagar pastas______
system(' rm -r postProcessing');
system(' rm -r processor0');
system(' rm -r processor1');
system(' rm -r processor2');
system(' rm -r processor3');
system(' rm -r 0');
cd ..
s=dir('sim');
d=size(s);
d=d(1);

for n=1:d
name=str2num(s(n,1).name);
   if isnumeric(name)==1 || strcmp('NaN', name)==1
      if   name~=0
         system(strcat('cd sim &&',32,'rm -r',32, num2str(name)));
         system('cd ..');
      end
```

```
    end
end
system(' cp -r 0 sim/');
end
```

ANEXO B- RESULTADOS OBTIDOS PARA VÁRIOS PARÂMETROS DE N, Θ.

n	θ	cd	$f_{pressao}$
1,5	15	0,1786	-0,5043
1,5	17	0,1794	-0,5055
1,5	19	0,1795	-0,5035
1,5	21	0,1791	-0,4979
1,5	23	0,1789	-0,4917
1,5	25	0,1791	-0,4938
2	15	0,1786	-0,4902
2	17	0,1793	-0,4912
2	19	0,1794	-0,4892
2	21	0,1790	-0,4834
2	23	0,1788	-0,4774
2	25	0,1790	-0,4795
2,5	15	0,1803	-0,5017
2,5	17	0,1811	-0,5029
2,5	19	0,1812	-0,5008
2,5	21	0,1808	-0,4949
2,5	23	0,1806	-0,4889
2,5	25	0,1807	-0,4911
3	15	0,1820	-0,5160
3	17	0,1827	-0,5169
3	19	0,1829	-0,5154
3	21	0,1825	-0,5092
3	23	0,1823	-0,5030
3	25	0,1824	-0,5051
3,5	15	0,1839	-0,5342
3,5	17	0,1847	-0,5354
3,5	19	0,1848	-0,5338
3,5	21	0,1844	-0,5275
3,5	23	0,1842	-0,5215
3,5	25	0,1844	-0,5237

ANEXO C- RESULTADOS OBTIDOS PARA N=1,65 E DIFERENTES VALORES DE Θ

n	θ	cd	$f_{pressao}$
1,65	21	0,1776	-0,4743
1,65	21,2	0,1776	-0,474
1,65	21,4	0,1776	-0,4739
1,65	21,6	0,1776	-0,4735
1,65	21,8	0,1776	-0,4749
1,65	22	0,1775	-0,4725
1,65	22,2	0,1775	-0,4724
1,65	22,4	0,1775	-0,4719
1,65	22,6	0,1775	-0,4711
1,65	22,8	0,1773	-0,4678
1,65	23	0,1774	-0,4682
1,65	23,2	0,1774	-0,4682
1,65	23,4	0,1777	-0,4706
1,65	23,6	0,1778	-0,4744
1,65	23,8	0,1773	-0,4705
1,65	24	0,1773	-0,4701
1,65	24,2	0,1773	-0,4701
1,65	24,4	0,1774	-0,4707
1,65	24,6	0,1783	-0,4662
1,65	24,8	0,178	-0,4723
1,65	25	0,1775	-0,4704

APÊNDICE A – SCRIPT PARA A OBTENÇÃO DAS FORÇAS DE ARRASTO (GERADO PELO MATLAB)

```
function forces1 = importfile1(filename, startRow, endRow)

delimiter = {' ',';','('};
if nargin<=2
    startRow = 496;
    endRow = 496;
end

formatSpec =
'%*s%*s%f%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s
%*s%*s%*s%*s%*s%[^\n\r]';

fileID = fopen(filename,'r');

dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter,
'HeaderLines', startRow(1)-1, 'ReturnOnError', false);
for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1,
'Delimiter', delimiter, 'HeaderLines', startRow(block)-1, 'ReturnOnError', false);
    dataArray{1} = [dataArray{1};dataArrayBlock{1}];
end

fclose(fileID);

dataArray = cellfun(@(x) num2cell(x), dataArray, 'UniformOutput', false);
forces1 = [dataArray{1:end-1}];
```

APÊNDICE B – SCRIPT PARA A OBTENÇÃO DOS COEFICIENTES DE ARRASTO (GERADO PELO MATLAB)

```
function forceCoeffs = importForcesCoeffs(filename, startRow, endRow)
delimiter = '\t';
if nargin<=2
    startRow = 9;
    endRow = 504;
end

formatSpec = '%s%s%s%s%s%s%[^\n\r]';

fileID = fopen(filename,'r');

dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1, 'Delimiter', delimiter,
'HeaderLines', startRow(1)-1, 'ReturnOnError', false);
for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-startRow(block)+1,
'Delimiter', delimiter, 'HeaderLines', startRow(block)-1, 'ReturnOnError', false);
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

fclose(fileID);

raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
for col=1:length(dataArray)-1
    raw(1:length(dataArray{col}),col) = dataArray{col};
end
numericData = NaN(size(dataArray{1},1),size(dataArray,2));

for col=[1,2,3,4,5,6]
    rawData = dataArray{col};
    for row=1:size(rawData, 1);
        regexstr = '(?<prefix>.*?)(?<numbers>([-]*(\d+[\,]*)+[\.]{0,1}\d*[eEdD]{0,1}[-
+]*\d*[i]{0,1})|([-]*(\d+[\,]*)*[\.]{1,1}\d+[eEdD]{0,1}[-+]*\d*[i]{0,1}))(?<suffix>.*)';
        try
            result = regexp(rawData{row}, regexstr, 'names');
            numbers = result.numbers;
```

```
            invalidThousandsSeparator = false;
            if any(numbers==',');
                thousandsRegExp = '^\d+?(\,\d{3})*\.{0,1}\d*$';
                if isempty(regexp(thousandsRegExp, ',', 'once'));
                    numbers = NaN;
                    invalidThousandsSeparator = true;
                end
            end
            if ~invalidThousandsSeparator;
                numbers = textscan(strrep(numbers, ',', ''), '%f');
                numericData(row, col) = numbers{1};
                raw{row, col} = numbers{1};
            end
        catch me
        end
    end
end

R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),raw); % Find non-numeric cells
raw(R) = {NaN}; % Replace non-numeric cells

%% Create output variable
forceCoeffs = cell2mat(raw);
```