UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE ENGENHARIAS DA MOBILIDADE
CURSO DE ENGENHARIA NAVAL

Thomaz Carvalho Lascala

GERAÇÃO GEOMÉTRICA E MODELAGEM AUTOMATIZADA DE PROPULSORES DA SERIE-B DE WAGENINGEN

Joinville
2016

Thomaz Carvalho Lascala

GERAÇÃO GEOMÉTRICA E MODELAGEM AUTOMATIZADA PROPULSOR B-SERIE DE WAGENINGEN

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Naval no curso de Engenharia Naval da Universidade Federal de Santa Catarina, Campus de Joinville.

Orientador: Dr. Lucas Weihmann

Joinville

2016

AGRADECIMENTOS

À minha mãe Marisa Martins Carvalho por todo suporte e amor durante toda minha vida, e por sempre acreditar e confiar.

Ao meu pai Nelson Thomaz Lascala Junior, por seus exemplos e peripécias, seu amor e carinho.

Ao meu irmão Rodrigo Lascala, no qual confio que sempre poderi contar em todas as situações.

As minhas falecidas avós, Dirceia e Leonice, que deveriam estar presentes neste momento.

Ao meu orientador Lucas Weihmann por seu tempo e apoio na confecção do presente trabalho.

Aos meus empregadores Yank Marine e Oceânica Offshore pelas oportunidades e crescimento pessoal e profissional.

À Universidade Federal de Santa Catarina, e todo seu corpo docente e administrativo que proporcionaram a minha formação.

Aos grandes amigos de fiz em Santa Catarina que estiveram aos meu lado nos últimos anos e me farão voltar muitas vezes.

A Deus pela oportunidade de estar aqui e tudo de melhor que me acontece.

I wonder why, I wonder why.
I wonder why I wonder.
I wonder ***why*** I wonder why.
I wonder why I wonder.
-Richard P. Feynman

RESUMO

O presente trabalho consiste na automatização da geração de geometria e modelagem tridimensional de propulsores do tipo B-Series usando os softwares MATLAB e Rhinoceros.

Uma rotina de MATLAB foi criada baseada na publicação original de Kuipper (1992) a respeito das Séries *Wageningen*. Todos os dados de entrada, como diâmetro, relação passo sobre diâmetro, razão de áreas expandidas e número de pás devem ser informados pelo usuário na interface gráfica e um arquivo contendo a geometria da pá do hélice será gerado. Para a modelagem automatizada, uma rotina em *IronPython* foi desenvolvida para ser executada no Rhinoceros com o arquivo gerado pelo MATLAB.

Para se obter uma modelagem com superfícies de alta qualidade, foram desenvolvidas no MATLAB, durante a rotina de geração da geometria, curvas de tendência e interpolação de valores não existentes na publicação original sobre a série. Com isso pode-se suavizar as curvas que representam cada uma das seções, criando mais pontos para descrevê-las, seguindo uma curva do tipo *spline* mais suave que a obtida utilizando apenas os dados originais. Além da criação de seções extras para frações de raios intermediárias afim de se obter um maior controle sobre a geometria.

A validação da automatização da geração geométrica e modelagem 3D se deu comparando-se um propulsor já projetado pela *Marine Technical Services* com um propulsor desenvolvido através das duas rotinas desenvolvidas para esse trabalho. Essa comparação foi feita de forma quantitativa comparando-se o valor de massa entre o propulsor desenvolvido e o de referência. Além do mais, inspeções visuais auxiliadas por ferramentas de análise do Rhinoceros foram usadas de forma qualitativa para validação da qualidade das formas e superfícies desenvolvidas. O comportamento da curva de áreas seccinionais também será analisado nessa etapa.

Um erro percentual relativo muito pequeno quanto a massa dos propulsores foi encontrado, indício que o propulsor modelado no presente trabalho está muito semelhante ao projeto original. As análises visuais e curvas seccionais indicam a ausência defeitos graves nas superfícies.

Palavras chave: Propulsor, Hélice, Automatização, B-Series, Wageningen.

ABSTRACT

The current work consists in automating the generation and three-dimensional modeling of B-Series type thrusters using MATLAB and Rhinoceros 3D.

A MATLAB script was created based on the original publication of Kuipper (1992) about the Wageningen Series. All the input data, as propeller diameter, pitch and diameter ratio, expanded area ratio and number of blades must be informed by the user in the graphic interface and the output will be a file with the blade geometry. For the automated modeling a script was developed in Iron Python to be run in Rhinoceros with the file generated by MATLAB.

To obtain a high-quality surface when modelling, was developed in MATLAB inside the geometric generating script trend curves of data and interpolation of non-existing values in the original publication about the series. Using the fitted curves was possible to smooth out the curves representing each one of the sections, by creating more points do describe them and so following a spline curve smother than the one resulted from the original data. In addition, were created extra sections between the original radius fractions in order to obtain higher control about the geometry.

In order to validate the whole process of geometric generating and automated modeling a propeller already designed by the Marine Technical Services will be compared with a propeller fully developed through the scripts of the present work. That comparation will be made qualitatively looking into the result for the propeller mass and polar moment of inertia between the developed propeller and the reference one. Furthermore, visual inspections assisted by the Rhinoceros analysis tolls will be used to evaluate, in a qualitatively way, the developed surfaces and forms. It will also be observed the sectional curve of areas behavior during this step.

A tiny relative percent error was found about the mass of the propellers, an indication that the propeller developed in the present work is very similar to the original design. The visual analysis and the curve of sectional areas also indicates the absence of seriues erros in the surfaces.

Keywords: Propeller, Thruster, Automatization, B-Series, Wageningen.

LISTA DE ABREVIATURAS E SIGLAS

CAD – *Computer Aided Design*

CAM – *Computer Aided Manufacturing*

cr – Comprimento de Corda

csv – *Comma Separated Values*

CNC – *Computer Numerical Control*

EAR – Razão de áreas expandidas

MARIN – *Maritime Research Institute Netherlands*

MTS – *Marine Technical Services*

NACA - *National Advisory Committe for Aeronautics*

NASA - *National Aeronautics and Space Administration*

P/D – Razão passo diâmetro

r/R – Razão entre o raio da seção local e o raio total da lamina

NURBS – *Non-Uniform Rational B-Splines*

$t_{l.e.}$ – Espessura bordo ataque para seção

t_{max} – Espessura máxima para seção

VB. NET – *Visual Basics .NET*

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

Sumário

1. INTRODUÇÃO

Com o desenvolvimento das ferramentas de desenho e fabricação assistido por computador (CAD/CAM do inglês, *Computer-Aided Design/Manufacturing*) muitas formas e tolerâncias no passado consideradas impossíveis, se tornaram realidade. Essa interação entre a modelagem computacional e produção é um avanço em termos de tempo e expectativa/realidade de produto. Softwares do tipo CAD/CAM permitem o desenvolvimento do produto em modelagem tridimensionais e tecnologias como comando numérico computadorizado (CNC do inglês, *Computer Numerical Control*) é responsável pelo processo de manufatura (AUTODESK, 2012).

O uso dessas tecnologias é aplicável em diversas áreas da engenharia naval, inclusive em um importante subsistema do sistema propulsivo de embarcações, o propulsor ou hélice, peça na qual qualquer distorção pode causar perdas de eficiência e problemas vibracionais, ao mesmo tempo que melhorias podem trazer resultados positivos em eficiência, o que implica diretamente no fator econômico da navegação.

O processo de modelagem não é algo trivial devido a sua forma não uniforme, porém esse processo pode ser separado em tarefas, de forma que possibilite a automatização dessa modelagem. Previamente ao processo de modelagem, estão os requerimentos e expectativas a serem atingidos, um processo optimizado de seleção do propulsor deve ser realizado. Após feita a seleção do hélice, o mesmo deverá ser modelado tridimensionalmente, para isso será necessário que sejam informados os pontos que definem a geometria ou plano de linhas, de acordo com a tolerância esperada de projeto.

O presente trabalho desenvolverá ferramentas computacionais para geração dos planos de linhas de propulsores do tipo B e BB-Series, conforme os parâmetros de entrada necessários para essa série sistemática. Uma série sistemática é o resultado de múltiplos ensaios e estudos realizados com a variação sistemática de uma ou mais variáveis, os primeiros ensaios referentes a B-Serie foram feitos com 5 hélices de 4 pás, com uma variação sistemática de passo (KUIPPER,1992). Para essa etapa será utilizado o software MATLAB R2015a.

Conforme a geometria definida pelo processo descrito anteriormente, a modelagem tridimensional se dará de forma automatizada através do software Rhinoceros. A automatização se dará por meio de uma rotina de programação em IronPython, plug-in de editor de programação interno nativo do Rhinoceros, proporcionando uma modelagem de forma muito mais rápida que uma modelagem manual.

A programação permite a automatização de ações, tomadas de decisões, execução de cálculos complexos e manipulações geométricas baseada em condições dinâmicas, tarefas que demoram consideravelmente mais conforme a complexidade se feitas por uma pessoa manualmente (RUTEN, 2010).

As cotas ou *offsets*, de um hélice, mesmo este sendo um elemento pequeno, quando comparado ao tamanho de uma embarcação, são muitas. Seu formato é complexo o que necessita de muitos pontos para descrevê-lo corretamente, o que acarreta em outras preocupações quanto a modelagem manual, devido ao fato da modelagem se iniciar a partir de uma nuvem de pontos, esse grande número de pontos em um ambiente de 3D pode ocasionar erros de modelagem. Os pontos no espaço constantemente se sobrepõe uns aos outros nos diferentes ângulos de visão o que pode fazer com que o usuário passe despercebido por um ponto oculto por uma linha ou outro ponto, ou ainda selecionar pontos incorretos por possuírem a mesma localização em um ângulo de visão, porém diferirem de localização no espaço. Mais que isso, o fator tempo é determinante em vários aspectos, por exemplo, quando se precisa comparar e testar diferentes modelos 3D em softwares de dinâmica de fluidos computacionais (CFD do inglês, *Computational Fluid Dynamics*), tendo a disponibilidade de um processo de modelagem rápida é possível testar diferentes dados de entradas, diferentes hélices, o que talvez fosse impossível se tivesse que ser modelado manualmente um a um, ou ao menos limitaria de forma expressiva a quantidade de variações testadas.

1.1. OBJETIVO PRINCIPAL

Desenvolvimento de um processo automatizado para geração de geometria e modelagem 3D de propulsor do tipo hélice, da série sistemática B e BB.

1.2. OBJETIVOS ESPECÍFICOS

- Desenvolver uma rotina de programação MATLAB script para B-Series;
- Gerar cotas e nuvem de pontos do propulsor conforme dados de entrada;
- Desenvolver uma rotina de programação em IronPython para Rhinoceros;
- Quantificar propriedades do modelo;
- Validar resultados de modelagem com literatura.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. HÉLICE PROPULSOR

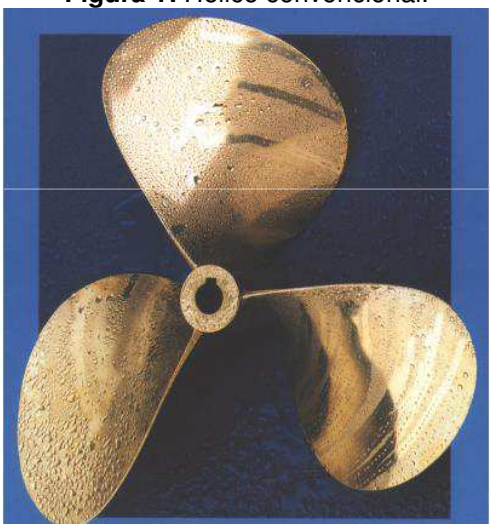
A finalidade de um propulsor é a de movimentar a embarcação, através da geração de uma força propulsiva por meio da variação da quantidade de movimento do fluido, no caso água doce ou salgada. Existem diversos tipos de propulsores navais, o mais convencional e antigo é o hélice, apesar de ter passado por muitos estudos e análises desde sua primeira aparição, seu princípio básico continua o mesmo.

O sistema propulsivo de uma embarcação é composto por uma diversidade de elementos e acopladores, destes elementos o propulsor é claramente o elemento de menor eficiência do sistema. O grande motivo da baixa eficiência do propulsor é devido ao seu método de geração de força, exatamente a variação da quantidade de movimento do fluido acarreta em energia cinética perdida para o meio na esteira.

Considerando que a eficiência do sistema propulsivo de uma embarcação está diretamente relacionada com seu desempenho econômico em um mercado altamente competitivo, a melhora da relação entre potência entregue e potência transmitida é de grande importância.

O hélice propulsor pode ser encontrado em diferentes configurações, sempre projetado para que seu ponto ótimo seja na condição de operação da embarcação. Assim sendo, um hélice eficiente em uma embarcação é especialmente projetado para essa embarcação, sua transição para outra embarcação acarretará em uma perda de eficiência considerável caso as embarcações não forem bastante semelhantes, ou suas funções sejam diferentes.

Figura 1: Hélice convencional.



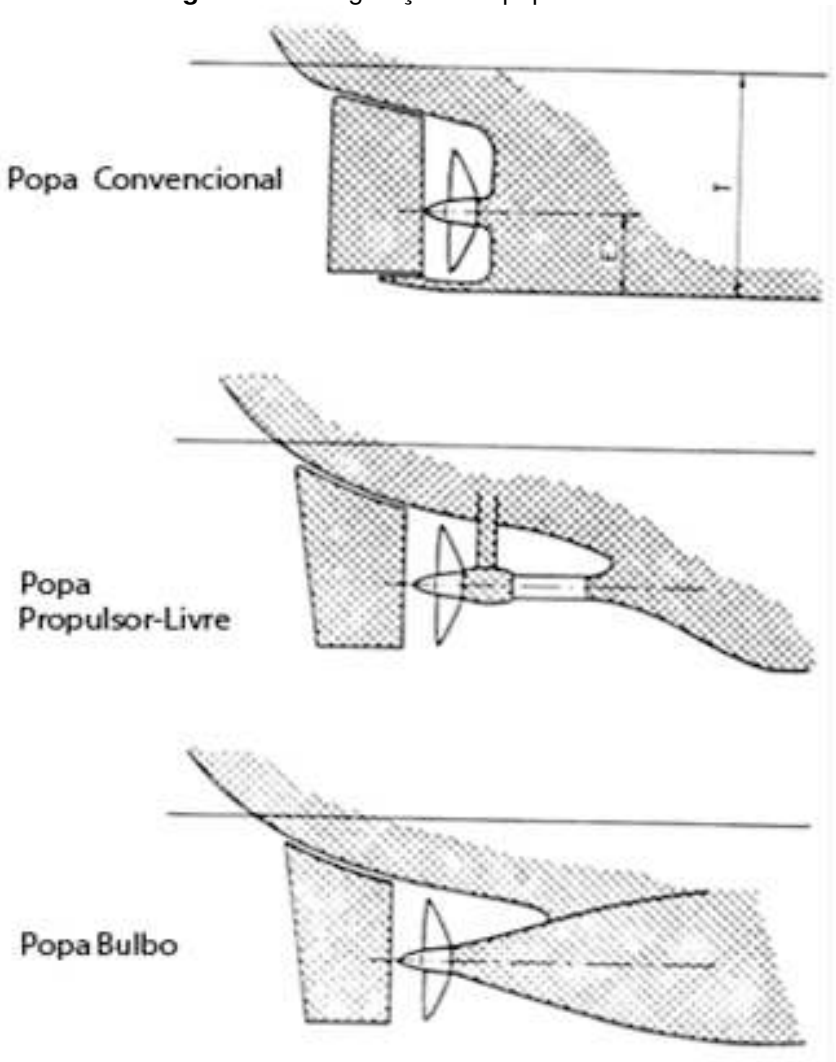
Fonte: GEER (2001).

O funcionamento do hélice se dá através de suas pás operando como superfícies sustentadoras. As mesmas são distribuídas simetricamente em torno do cubo, como mostrado na Figura 1. Cada pá se comporta como uma asa gerando sustentação conforme o ângulo de ataque para o escoamento e a força de sustentação contribui para a força propulsiva axial e para o binário resistente ao eixo de transmissão.

Apesar de que a parte principal de um projeto propulsivo seja a definição do hélice e suas propriedades geométricas, sua configuração de instalação também faz parte do projeto, essa interação casco-hélice é determinante para garantir uma boa eficiência. O hélice é instalado a popa do navio, isto é, a ré do navio, e considerações de rendimento e resistência justificam essa decisão. Caso fosse instalado na proa do navio, isto é, a vante do navio, a esteira do hélice causaria grande aumento na resistência total ao movimento do casco.

Diferentes formas de popa foram desenvolvidas com o intuito de melhorar a necessidade de potência de uma embarcação, como ilustrado na Figura 2. Porém, nem sempre a forma de popa está relacionada com resistência ao avanço ou eficiência do sistema propulsivo, por exemplo a popa *transom* usada em navios porta-containers tem por objetivo aumentar o volume de carga carregado no navio, se resumindo novamente a relação econômica e razão de carga transportada por potência instalada.

Figura 2: Configurações de popa e hélice.



Fonte: CARLTON (2007) – Adaptada.

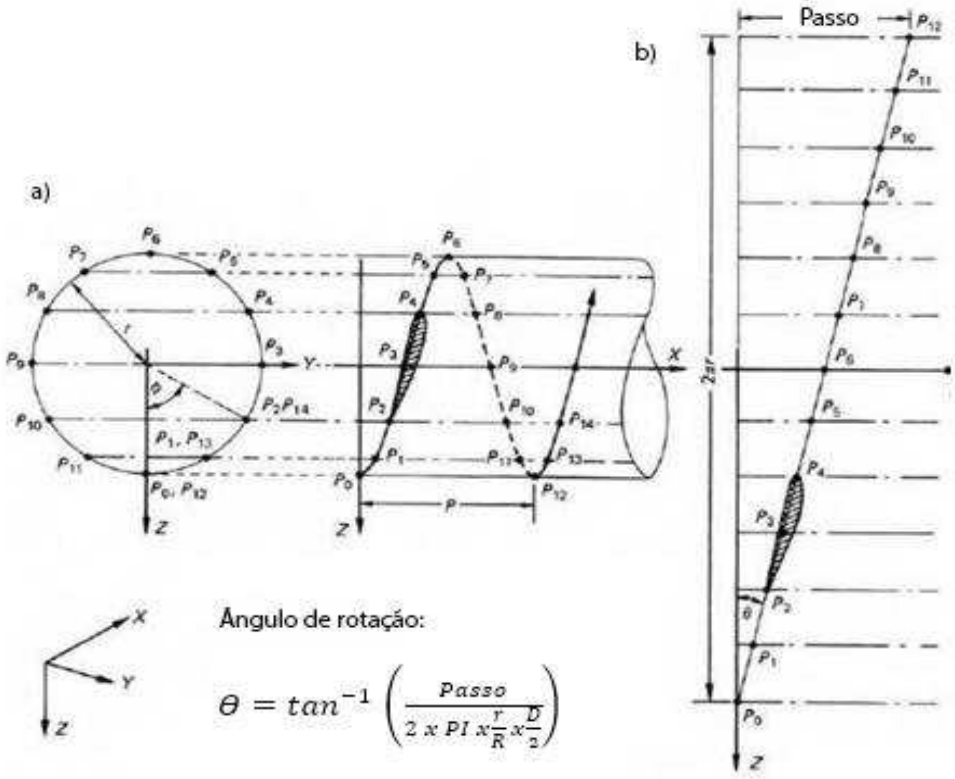
2.2. GEOMETRIA DO HÉLICE

Diversos fatores geométricos definem um hélice. Os mais importantes do ponto de vista hidrodinâmico e de eficiência são o número de pás, o passo, o caimento, a assimetria do contorno de pá e o perfilda pá.

O número de pás é um dos parâmetros que afeta diretamente todas as curvas de eficiência de um hélice. Não existe um melhor número de pás para cada aplicação, mas as mais comumente utilizadas, são 4, 5 ou 6 pás para navios mercantes, 2 e 3 para lanchas e barcos menores, apesar de existirem também propulsores de 7 e 8 pás (CARLTON, 2007). Quanto maior a área projetada pelas pás, ou seja, quanto mais próximo da área de um disco, maior a força de sustentação porém a resistência por atrito cresce proporcionalmente.

O passo do hélice é o valor de deslocamento na direção axial de um ponto na sua extremidade ao efetuar uma revolução completa, 360 graus (GEER, 1989). Conforme a Figura 3(a), podemos verificar que, para cada ângulo de rotação o deslocamento axial vai crescendo, chegando ao que se chama de passo ao final da revolução completa. E na Figura 3(b) que para um passo constante o ângulo de passo varia afim de garantir essa característica.

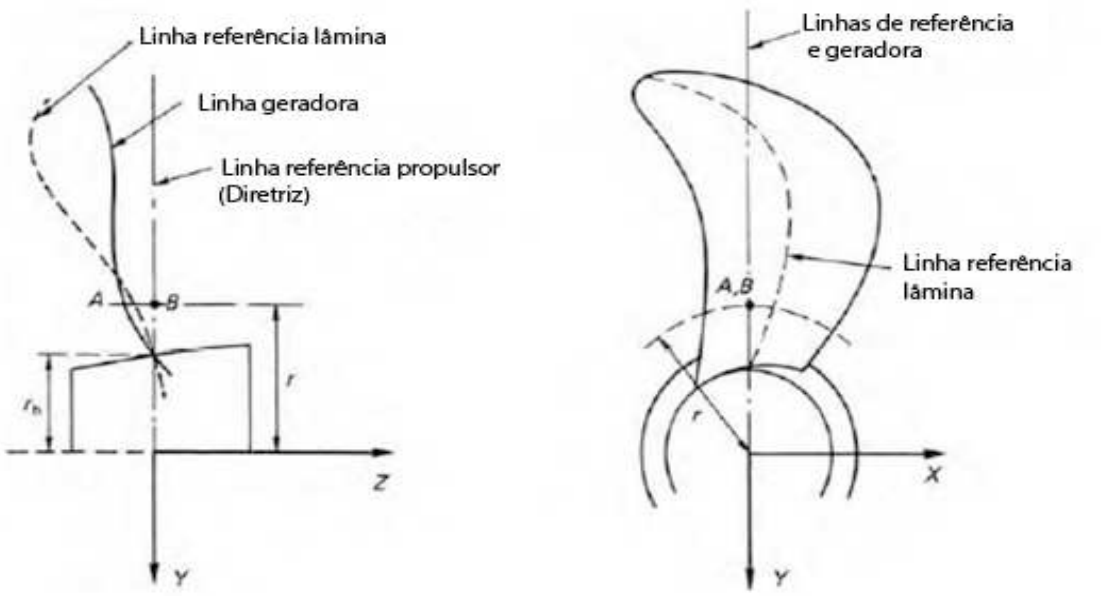
Figura 3: Passo hélice.



Fonte: CARLTON (2007) – Adaptada.

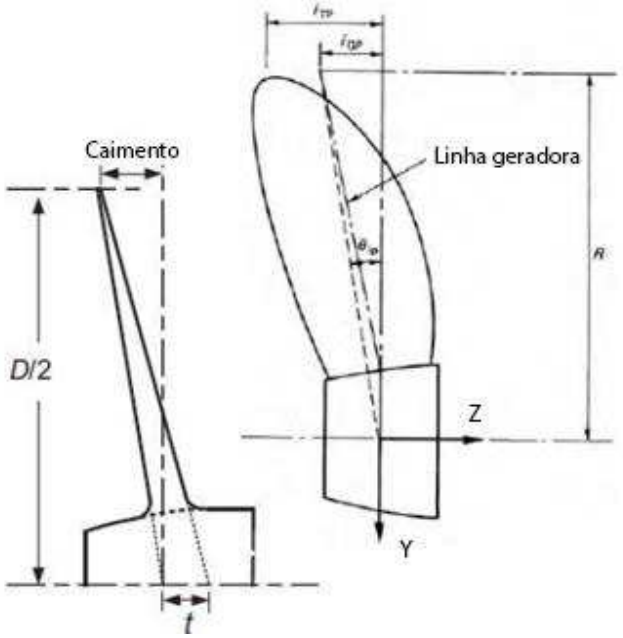
O caimento axial da secção é a distância da linha geratriz da pá ao ponto de intersecção da linha de referência do hélice com o cilindro da secção (GEER, 1989). As linhas de referência de pá, geradora e diretriz podem ser observadas na Figura 4. Já na Figura 5 observa-se o caimento axial como sendo um deslocamento axial das seções de forma. Para hélices da série B, o ângulo de caimento vale sempre 15 graus.

Figura 4: Linhas de referência.



Fonte: CARLTON (2007) – Adaptada.

Figura 5: Caimento (em ingês, R*ake*).

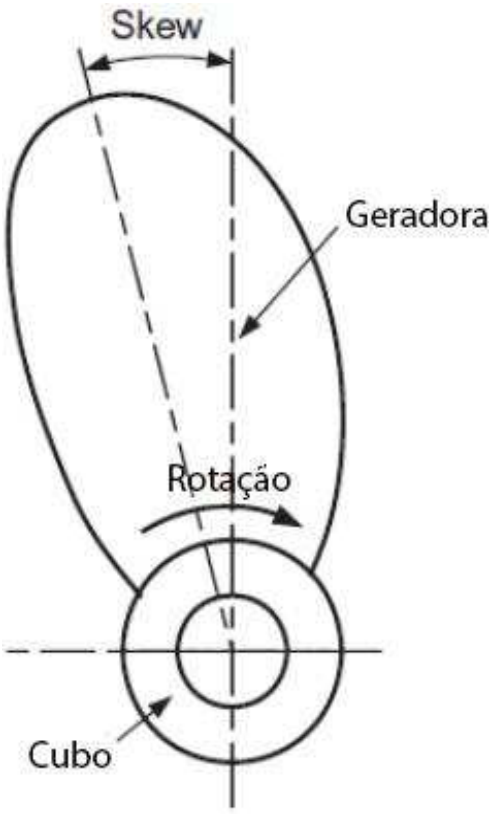


Fonte: CARLTON (2007) – Adaptada.

Conforme observado nas linhas de referência do hélice na Figura 4 a projeção da distância entre a linha geradora e a linha de referência da pá no plano do hélice é chamada de assimetria de contorno de pá, ou deslocamento circunferencial da seção da pá (GEER, 1989). O termo em inglês, *Skew*, será usado no decorrer do trabalho por ser amplamente mais utilizado que suas traduções. O

valor de *Skew* possui sinal negativo se for na direção negativa de giro e positivo se conforme a direção positiva de giro. Na Figura 6 podemos observar a assimetria em relação a linha geradora, conforme descrito acima, a assimetria é negativa, pois está no sentido negativo de rotação.

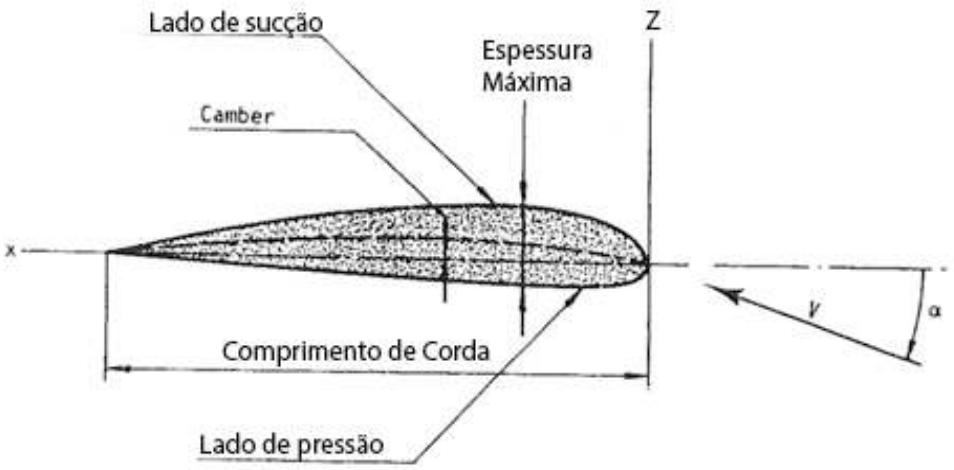
Figura 6: *Skew.*



Fonte: CARLTON (2007) – Adaptada.

Os perfis das pás do hélice para um determinado raio, possuem o formato de um aerofólio e são sua intersecção com uma superfície cilíndrica de mesmo raio. Um corte, de formato genérico de secção de pá é ilustrado na Figura 7 com o objetivo de definir alguns parâmetros característicos do perfil.

Figura 7: Perfil da seção.



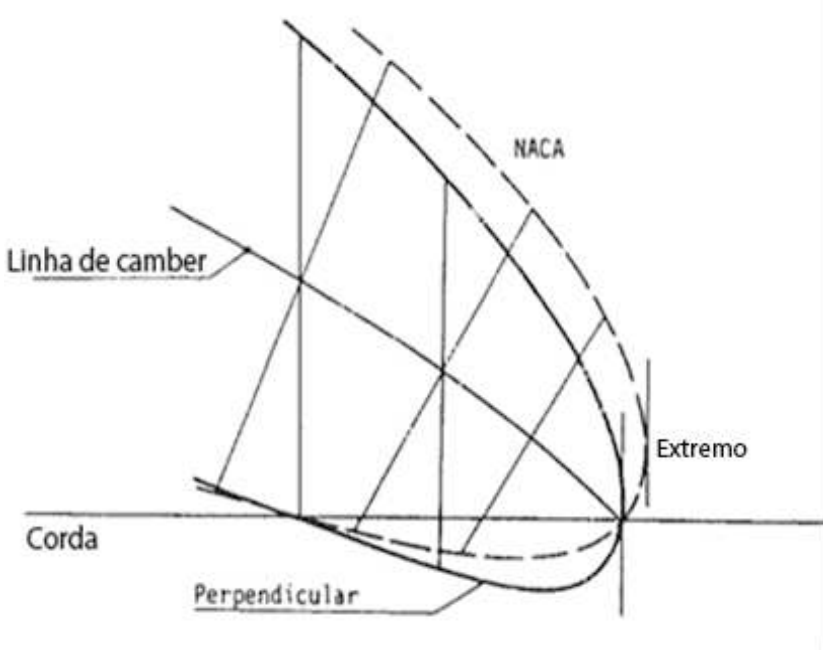
Fonte: KUIPPER (1992) – Adaptada.

O bordo em contato com o escoamento é o bordo de ataque, e o ângulo desse encontro é o ângulo de ataque, sendo esse positivo no sentido horário, como ilustrado na Figura 7. Um ângulo de ataque positivo resulta em uma pressão menor na parte de posterior do perfil, nomeando o mesmo por lado de sucção, e o inferior por lado de pressão (KUIPER,1992). Enquanto o bordo de ataque possui um formato arredondado, o bordo oposto, bordo de fuga, possui geralmente um formato afiado.

A distância em linha reta entre os pontos mais distantes do bordo de ataque e de fuga caracterizam o tamanho de corda da secção. As distâncias entre os lados de sucção e pressão descrevem a distribuição de espessura da secção, e a linha que passa pela metade dessas distâncias é conhecida por linha de *camber*, sendo a distância vertical dessa linha para uma linha horizontal entre os bordos de ataque e fuga conhecida por distribuição do *camber*.

Alternativamente existe outra maneira de se definir as espessuras e o *camber* do perfil. A antiga *National Advisory Committe for Aeronautics* (NACA), hoje em dia conhecida por *National Aeronautics and Space Administration* (NASA), conduziu na década de 30, um estudo baseado na geometria de aerofólios, destes, alguns perfis são ainda hoje adotados para o projeto de propulsores marítimos (CARLTON, 2007). Sendo para o perfil NACA a espessura uma medida perpendicular a linha de *camber* e o bordo de ataque passa a ser a intersecção da linha de *camber* com a linha de contorno, sendo por sua vez o ponto extremo de contorno mais adiante, como ilustrado na Figura 8.

Figura 8: Perfil pá NACA.



Fonte: KUIPPER (1992) – Adaptada.

Os resultados desenvolvidos pela NACA, e conhecido por séries NACA, são amplamente usados como base para projeto de propulsores, possuem um baixo arrasto e boas características de cavitação. São perfis caracterizados por uma espessura base e distribuição de *camber*, porém ambos podem ser modificados independentemente um do outro, facilitando assim o projeto de perfis com propriedades específicas (TAKINACI, 2015). Detalhes de todas as séries NACA podem ser encontradas na referência [6]. As próprias séries de Wageningen C e D são baseadas em perfis NACA 66 (DANG, 2013).

2.3. SÉRIE SISTEMÁTICA: B-SERIES

Com a expansão do setor marítimo na economia mundial, cresceu a evolução tecnológica e a busca por maiores eficiências, originando-se assim a necessidade dos projetistas navais de terem disponíveis dados confiáveis sobre propulsores. Nesse cenário surgiu o interesse pelo estudo do comportamento de propulsores e no que acarretam suas mais diversas configurações. Em 1936 foi feita a primeira publicação pelo *Maritime Research Institude Netherlands* (MARIN) a respeito dos resultados dos primeiros testes sistemáticos de propulsores, cinco

propulsores com 4 pás tiveram seu passo variado sistematicamente e testados. Muitas outras publicações foram feitas a respeito desse tópico, expandindo o número de propulsores e características contemplados pela série, chegando a um total de 130 propulsores (KUIPER,1992).

A publicação mais completa e desenvolvida foi feita em 1969, na qual foram feitas correções nos diagramas de águas abertas conforme o número de Reynolds, assim como polinômios foram desenvolvidos para representar as curvas dos diagramas anteriormente publicados. Com o desenvolvimento dos polinômios, tornou-se mais fácil o uso da série como uma alternativa ao uso de diagramas visualmente, tarefa bastante demorada e imprecisa.

A nomenclatura utilizada pela Séries B informa o número de pás, e a razão de áreas expandidas do propulsor, no caso de um B4-40, significa que esse propulsor é formado por quatro pás e sua razão de áreas expandida é 0.4.

Quando falamos a respeito da geometria dos propulsores das séries, apenas são de interesse os parâmetros que influenciam nas características propulsivas dos propulsores. Distribuição do *camber* e o *camber* máximo são parâmetros importantes. A distribuição de carregamento radial tem influência moderada porém perceptível no desempenho do propulsor, principalmente determinada pela distribuição radial do passo. As variáveis geométricas mais importantes são as variadas sistematicamente: razão entre passo e diâmetro, área expandida e número de pás.

A B-Series é usada para prever e otimizar característica propulsivas como impulso gerado, torque e número de revoluções. Caso haja interesse em estudar o efeito da cavitação, uma detalhada descrição geométrica do propulsor é necessária. Características de *rake*, contorno de pá, distribuição de espessura e pequena assimetria são importantes para a análise de cavitação porém são de baixa influência nas características de desempenho do propulsor. Durante o desenvolvimento da série, os propulsores testados não foram submetidos a carregamentos extremos em suas extremidades, portanto para esses casos, a eficiência real pode ser inferior a prevista pela série (KUIPER, 1992).

Apesar de métodos teóricos para projeto de propulsores, como a teoria da linha/superfície de sustentação, as séries continuam sendo vastamente usadas para projetos preliminares. Sendo uma opção também quando se tem uma limitação

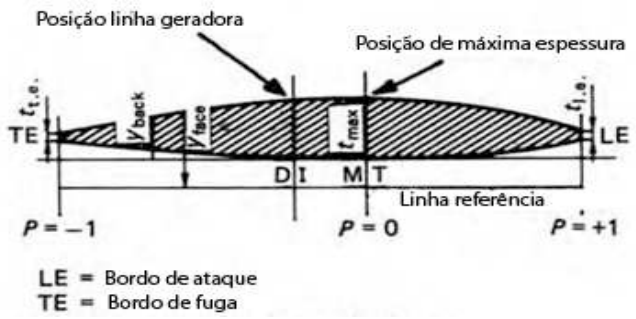
financeira para aquisição de softwares específicos, entre as séries sistemáticas a Séries B de Wageningen é uma das mais comumente utilizadas em projetos navais.

2.3.1 Geometria B-Series

A geometria das pás de um propulsor do tipo B-Series sem passo é definido através de uma série de tabelas e coeficientes. Os dados de entrada são simples, como o diâmetro do propulsor, razão de áreas expandidas e número de pás.

O plano de linhas, ou cotas do propulsor, são estabelecidas para cada razão de r/R, onde r é a posição local e R o raio total da pá. Sendo assim são calculados os comprimentos de corda, a distribuição de espessura, os pontos do lado de pressão e sucção do hélice. O plano de referência para as seções da pá é uma linha no lado de pressão da pá, sendo essa a base para modelagem do mesmo. Sua geometria é separada em quatro partes, do bordo de fuga a espessura máxima e do bordo de ataque a espessura máxima em ambos os lados. A espessura máxima é considerada o ponto zero e todas as coordenadas são expressas em porcentagens da mesma, como ilustrado na Figura 9.

Figura 9: Perfil pá NACA.



Fonte: CARLTON (2007) – Adaptada.

$$y_{face} = V_1\,(t_{max} - t_{l.e.}) \quad (1)$$

$$t_r = V_2\left(t_{max} - t_{l,e,}\right) + t_{l.e.} \quad (2)$$

A geometria de cada uma das seções, frações de r/R, é dada pelas Equações 1 e 2. Sendo y_{face} os pontos que descrevem o lado de pressão da pá, em relação a linha de referência, e t_r são as espessuras em cada ponto, através da soma dos dois se tem a localização dos pontos do lado de sucção. A espessura

máxima de cada seção é t_{max} e a espessura do bordo de ataque, $t_{l.e.}$ vale cerca de 20% da espessura máxima.

No presente trabalho a linha de referência será tratada como estando sobre o eixo X, e as posições dadas pelas Equações 1 e 2 no eixo Z. As tabelas de Kuipper (1992) tratam nessa etapa o eixo X como variando de -100% , bordo de fuga, a 100%, bordo de ataque, sendo o 0% no ponto de espessura máxima da seção.

Os coeficientes V_1 e V_2 são provenientes das tabelas propostas por KUIPPER (1992). Com o uso das Equações 1 e 2, são gerados 9 pontos descrevendo a geometria do ponto 0 ao -100%, e 10 pontos descrevendo a geometria de 0 a 100%, sendo o décimo ponto o valor de $t_{l.e.}$. Nessa etapa os pontos gerados no eixo Z levam em consideração a configuração do propulsor enquanto que os valores de X são descritos como porcentagens da corda da seção.

2.4. SOFTWARES

Um levantamento dos softwares já existentes no mercado relacionados ao tema abordado no presente trabalho será apresentado a seguir, assim como os softwares utilizados para o desenvolvimento do mesmo.

2.4.1 Softwares relacionados

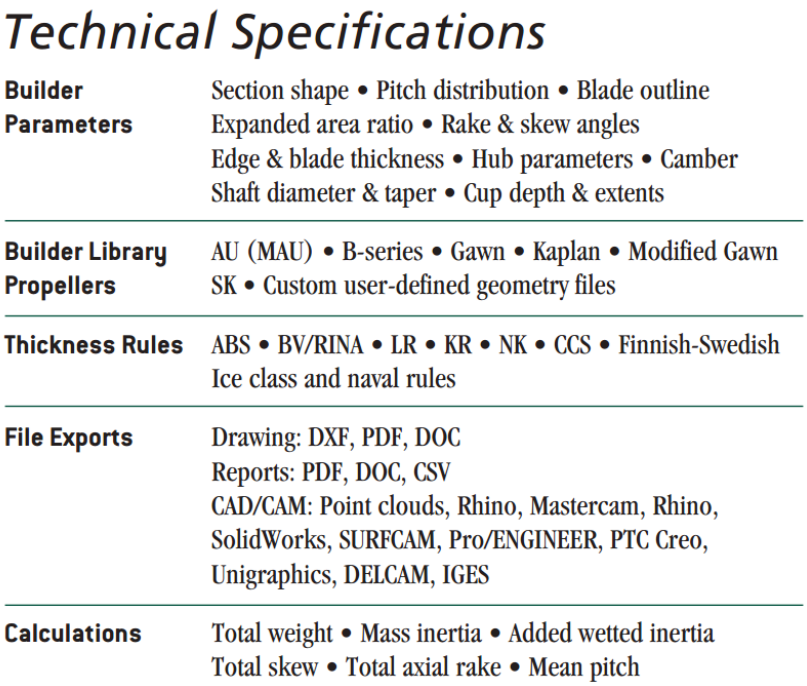
Constatou-se que o mercado atualmente carece de softwares específicos relacionados a modelagem e seleção de propulsores. Apesar de existirem diversos trabalhos acadêmicos relacionados à otimização da seleção de propulsores, em relação à modelagem de propulsores a oferta de trabalhos acadêmicos é consideravelmente limitada, e os que existem são relacionados a modelagem automatizada ou semi-automatizada em outros softwares CAD de uso geral, como AutoCAD, Rhinoceros ou SolidWorks,

A Hydrocomp é uma empresa especializada em softwares de previsões de desempenho para indústria naval, sendo responsável por softwares como PropCad, PropExpert e NavCad. Apesar de conhecidos mundialmente, seus softwares não são amplamente utilizados devido aos seus custos de aquisição.

O PropExpert é uma ferramenta de seleção de propulsores para barcos de trabalho e lazer, faz a seleção das características dos principais componentes do sistema propulsivo. Já o NavCad não tem o foco no propulsor, e sim em análises de resistência ao avanço e características propulsivas, porém também recomenda certas características do propulsor a ser utilizado.

O ProCad é uma ferramenta que trata da modelagem geométrica e tridimensional do propulsor, suas funcionalidades são apresentadas na Figura 10. Foi o software encontrado que apresenta maior aderência com a proposta desse trabalho, sendo utilizado para modelagem e preparação de desenhos em 2D, visualização 3D de propulsores marítimos, hélices e impulsores. Sua interface é relativamente simples, possuindo uma grande área para visualização do propulsor em 2D ou 3D e uma área para a entrada de dados. São necessários entradas como tipo de propulsor, sentido de rotação, número de pás, diâmetro do propulsor, passo nominal e *rake*. Fornece suporte aos requerimentos das principais sociedades classificadoras e uma biblioteca com diversas séries sistemáticas, além de B e BB-Series. Possui uma fácil integração com os formatos mais comuns da CAD, podendo exportar sua geometria em diversos formatos.

Figura 10: PropCAD – Especificações técnicas.



Technical Specifications

Builder Parameters	Section shape • Pitch distribution • Blade outline Expanded area ratio • Rake & skew angles Edge & blade thickness • Hub parameters • Camber Shaft diameter & taper • Cup depth & extents
Builder Library Propellers	AU (MAU) • B-series • Gawn • Kaplan • Modified Gawn SK • Custom user-defined geometry files
Thickness Rules	ABS • BV/RINA • LR • KR • NK • CCS • Finnish-Swedish Ice class and naval rules
File Exports	Drawing: DXF, PDF, DOC Reports: PDF, DOC, CSV CAD/CAM: Point clouds, Rhino, Mastercam, Rhino, SolidWorks, SURFCAM, Pro/ENGINEER, PTC Creo, Unigraphics, DELCAM, IGES
Calculations	Total weight • Mass inertia • Added wetted inertia Total skew • Total axial rake • Mean pitch

Fonte: ProCAD technical specifications (Hydrocomp, Inc, 2016).

2.4.2 Softwares utilizados no desenvolvimento

Os programas utilizados no desenvolvimento do trabalho foram MATLAB e Rhinoceros. Ambos os softwares foram escolhidos devido a sua disponibilidade na Universidade Federal de Santa Catarina – Campus Joinville e por possuírem características que se mostraram adequadas para execução do trabalho proposto.

O MATLAB, acrônimo de MATrix LABoratory, desenvolvido e comercializado pela MathWorks é um software interativo de alta performance voltado para o cálculo numérico, ambiente de programação de alto nível possuindo características voltadas a facilitar a vida do usuário, onde não são necessárias declarações de variáveis, alocação de memória ou utilização de ponteiros. Por ser um ambiente de programação de alto nível, apresenta uma série de funções matemáticas já implementadas que podem ser utilizadas pelo usuário, além de *Apps (*ou *Tollboxes)*, que tem por objetivo de facilitar mais ainda ao usuário visualizando algumas funções, como o *Curve Fitting* ou *Distribution Fitting*.

O MATLAB será usado na geração da geometria do propulsor, isto é, a geração de uma nuvem de pontos que descreve a forma da pá do hélice. Devido a sua reconhecida capacidade matemática, é utilizado em diversas empresas de

engenharia ao redor do mundo, foi-se considerada uma ferramenta confiável para essa tarefa.

Rhinoceros é um software comercial voltando principalmente para modelagens em três dimensões, desenvolvido pela Robert McNeel & Associates. Sua principal distinção perante outros softwares do tipo é o fato dele trabalhar com *Non- Uniform Rational B-Splines* (NURBS), representações matemáticas de geometrias 3D que podem precisamente descrever formatos em 2D. Descrições mais detalhadas sobre o assunto serão expostas na próxima seção. Possui grande capacidade de precisão, sendo um software amplamente empregado nos setores de projetos marítimos, industriais, automobilísticos, sendo utilizado para modelar desde joias até aeronaves. Além do fato de que seu custo de aquisição é baixo, uma licença estudantil custa apenas U$195 (Windows ou MAC), possui compatibilidade com os mais conhecidos formatos de documentos.

Para a automatização da modelagem 3D do hélice propulsor, era necessário um software que aceitasse linhas de comando, e melhor que isso, rotinas de programação. Através de um *plug-in* nativo do Rhinoceros 5, é possível editar e rodar rotinas de programação escritas em IronPython internamente no CAD. Através desse *plug-in* é possível gerar rotinas de programação capazes de automatizar tarefas, tomada de decisões, realizar cálculos e manipulações geométricas. IronPython é uma linguagem de código aberto, de alto nível multiplataforma (Windows, Linux, Mac) e orientada a objetos. Pode-se utilizar tanto das bibliotecas Python como .NET Framework.

2.5 TIPOS DE CURVAS

Conforme dito anteriormente a geração da geometria do hélice se dará por pontos dos perfis que descrevem a pá do hélice, porém apenas os pontos definidos pela série sistemática não descrevem uma superfície. Para isso, tomando como referência esses pontos, serão geradas curvas interpoladas ou aproximadas afim de que uma superfície possa ser descrita.

Uma curva é definida por seus pontos de controle, quando a curva passa por todos os pontos de controle, significa que uma interpolação foi realizada, caso o contrário uma aproximação foi feita. Caso valores fora do limite inicial e final forem gerados, uma extrapolação foi realizada. Curvas podem ter controle global ou local, .

Em uma curva com controle global, caso um só ponto de controle for modificado, irá afetar a curva como um todo. Caso apenas uma parte dela for afetada, ela possui controle local (VILLAS BOAS, 2006).

Uma importante propriedade de uma curva é sua continuidade, podendo essa ser separada em 4 tipos: continuidade geométrica zero, onde o ponto final de uma curva é o inicial de outra; continuidade geométrica um, onde a direção dos vetores tangentes são iguais; continuidade paramétrica um, onde os vetores tangentes são iguais em amplitude; a continuidade paramétrica de ordem “n”, onde a derivada de ordem “n” dos vetores tangentes é igual em amplitude (VILLAS BOAS, 2006).

A forma mais comum de representação de curvas é aquela em que uma coordenada é obtida em função de outra, $y = F(x)$. Os ajustes lineares e polinomiais são normalmente utilizados para representação matemática dessas curvas. Os ajustes lineares nada mais são do que polinômios de grau um, retas. Podem existir restrições de ponto, ângulo ou curvatura conforme a aplicação da mesma. Restrições de ângulo e curvatura são geralmente colocadas no final de uma curva, com o objetivo de garantir uma transição suave entre curvas de diferentes polinômios. Se houver mais do que $n + 1$ restrições (onde n é o grau do polinômio), não há certeza que a função polinomial atenderá todas as restrições (GUEST, 2012).

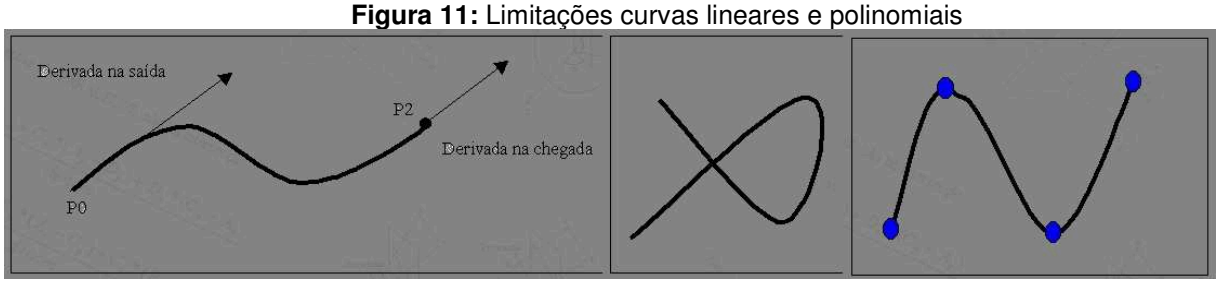
O método dos mínimos quadrados é uma maneira de avaliar como o polinômio se adequa a distribuição dos pontos, através da comparação dos resíduos. Os resíduos são as diferenças entre os valores originais e os valores que a curva ajustada resultam para a variável dependente em função da variável independente. O método efetua a soma dos quadrados dos desvios, quanto mais próximo de um, mais próxima a curva ajustada se aproxima dos pontos originais.

Caso a soma dos quadrados dos resíduos seja igual a um, a curva ajustada passa por todos os pontos originais. No entando, existem diversas razões para se preferir um ajuste aproximado com um polinômio de grau menor a um ajuste exato usando um polinômio de grau elevado. Polinômios de alta ordem são altamente oscilatórios, o número máximo de pontos de inflexão de uma curva polinomial é $n-2$, isso significa que um polinômio de grau quinze pode ter até treze pontos de inflexão, pontos em que a curvatura muda de positiva para negativa, ou vice-versa, assim como pode ter qualquer número inferior até zero. Portanto

polinômios de baixar ordem tendem a ser mais suaves e polinômios de alta ordem a ser mais irregulares (GUEST, 2012).

Essas curvas serão amplamente utilizadas no decorrer desse trabalho afim extrapolar e interpolar coeficientes para geração da geometria do propulsor. Porém quando estamos trabalhando com modelagem geométrica esses tipos de curvas possuem certas limitações e inconvenientes. É difícil definir a equação de uma curva desse tipo através de seus pontos e derivadas nestes pontos, técnica bastante utilizada em modelagem geométrica, Figura 11. Além de ser impossível criar curvas com laços e apesar de não impossível, é bastante complicado a criação de uma curva suave passando por um conjunto de pontos (COHEN,2002).

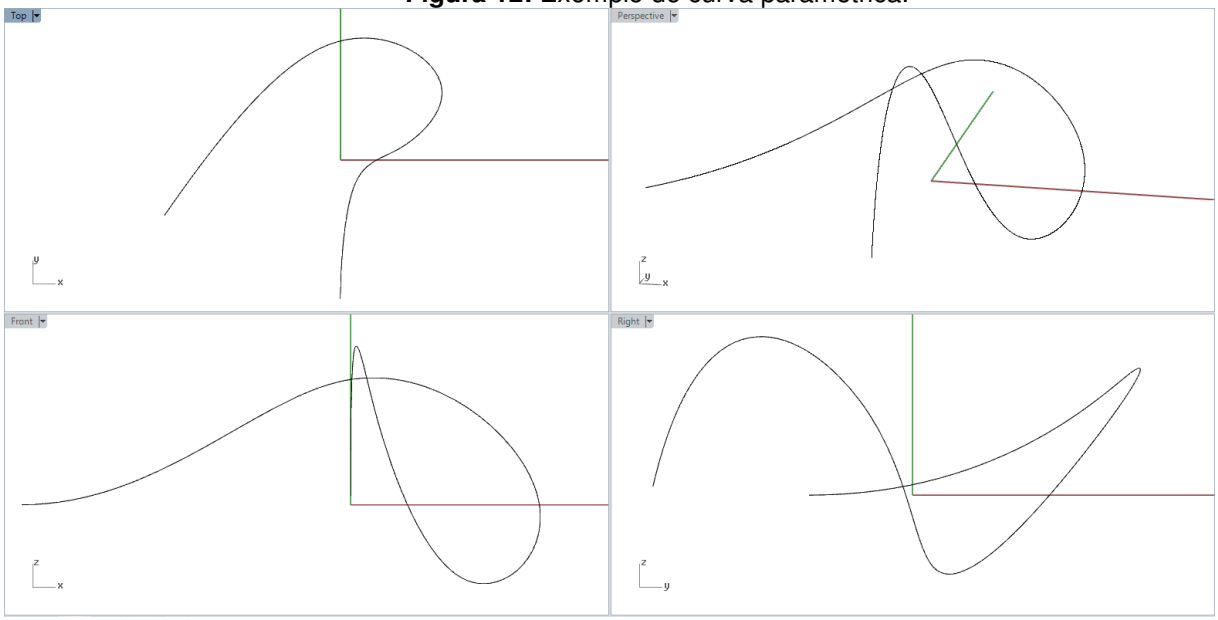
Figura 11: Limitações curvas lineares e polinomiais



Fonte: Cohen (2002)

Para aplicações onde a formulação polinomial, ou y = F(x), não é suficiente, existem as curvas paramétricas. Na parametrização o comportamento da curva é definido por funções independes para cada um dos eixos, sendo, $x = F_1(u)$, $y = F_2(u)$, $z = F_3(u)$. Um exemplo simples são, trajetórias de projetil, as posições verticais e horizontais são dadas em função do tempo, ou então e a equação paramétrica de um círculo de raio um, com centro em *(0,0),* são elas $x = cos(t)$ e $y = sen(t)$. Na Figura 12 observa-se a mesma curva quando apresentada em x,y,z (3-D) e quando apresentada apenas em suas projeções 2-D.

Figura 12: Exemplo de curva paramétrica.



Fonte: Autor (2016).

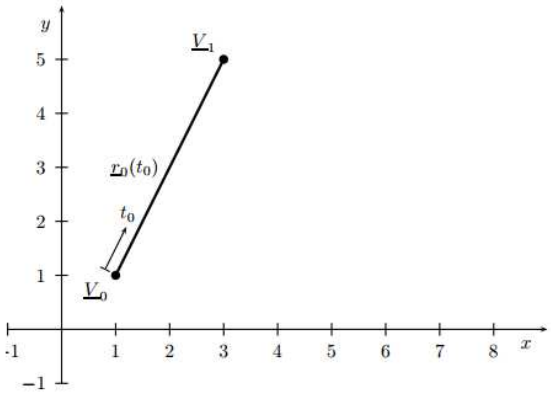
Para que curvas paramétricas sejam capazes de ser criadas a partir de pontos por onde elas devem passar, ou ser criadas dados os pontos e as derivadas da curva nestes pontos, composições ponderadas são necessárias (COHEN, 2002). As três formulações que mais se destacam e são mais utilizadas em programas CAD são: curvas Bérzier, curvas Hermite e curvas Spline (B-Spline) (VILLAS BOAS, 2006).

2.5.1 Curvas Bérzier

Uma linha reta conectando dois pontos é a forma mais simples de uma curva Bérzier. Foram desenvolvidas pelo engenheiro francês Pierre Étienne Bérzier (1910-1999) a serviço da montadora de automóveis Renault sendo um pioneiro no desenvolvimento de ferramentas CAD/CAM para o projeto de automóveis.

No caso de uma linha reta conectando dois pontos, Figura 13, esses pontos são seus vértices, $\underline{V_0}$ e $\underline{V_1}$ e ao mesmo tempo seus pontos de controle. Um fator é atribuído a cada vértice, no geral, fatores são funções dos parâmetros da curva, chamadas de funções base. As funções bases das curvas Bérzier são polinomiais.

Figura 13: Curva de Bérzier de grau *p=1* e ordem *k=2*



Fonte: L. BIRK (2014)

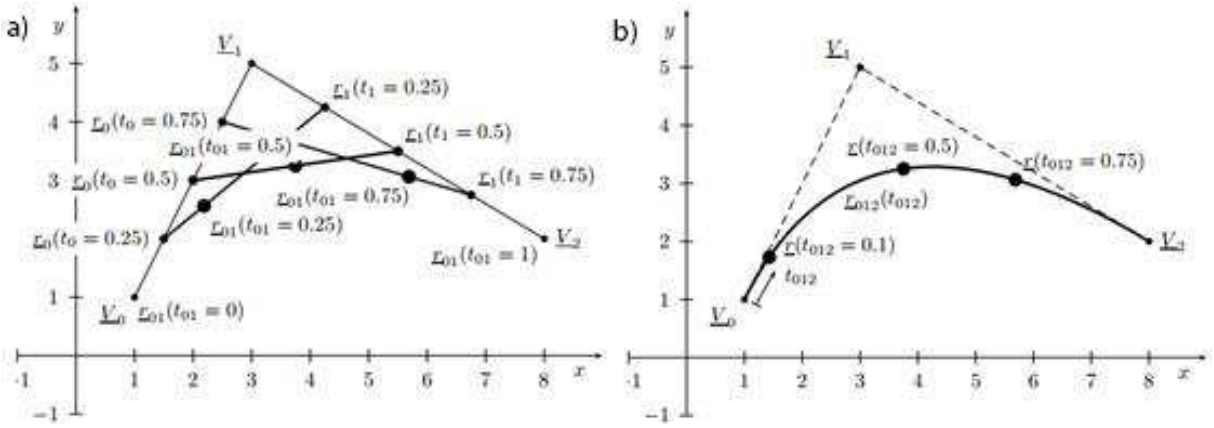
As funções bases polinomiais são uma combinação de baricentros, sendo a curva da Figura 14 descrita pelas Equações 3 e 4.

$$\underline{r}_0(t_0) = \underline{V}_0 \cdot B_{02}(t_0) + \underline{V}_1 \cdot B_{12}(t_0) = \sum_{i=0}^{1} \underline{V}_i \cdot B_{i2}(t_0) \quad (3)$$

$$B_{02}(t_0) = (1 - t_0) \quad \text{e} \quad B_{12}(t_0) = t_0 \quad (4)$$

O processo de construção para curvas de Bérzier de graus maiores persiste na mesma combinação de baricentros como ilustrado nas Figuras 14(a) e 14(b). A linha ligando o vértice V_1 e V_2 também pode ser descrita de maneira independente como pelas Equações 3 e 4, as linhas $r_0(t_0)$ e $r_1(t_1)$ representam um infinito conjunto de pontos. Para geração da curva deliberadamente escolhemos unir pontos de mesmo parâmetro *t*, as retas originais são casos de $t = 0$ e $t = 1$. Nas Equações 5 e 6 pode-se observar seu desenvolvimento matemático.

Figura 14: Construção de curva de Bérzier de grau *p=2* e ordem *k=3*



Fonte: L. BIRK (2014)

$$\underline{r}(t) = (1 - t_{01}) \left[(1 - t_0)\underline{V_0} + t_0\underline{V_1} \right] + t_{01} \left[(1 - t_1)\underline{V_1} + t_1 \underline{V_2} \right. \quad (5)$$

Para $t_0 = t_1 = t_{01} = t$

$$\underline{r}(t) = (1 - t)^2 \underline{V_0} + 2(1 - t)\, t\, \underline{V_1} + t^2 \underline{V_2} \quad (6)$$

As curvas de Bézier são amplamente utilizadas em programas de vetorização gráfica como CorelDraw, Inkscape e Adobe Illustrator. Porém em modelagem 3D elas foram quase que em sua totalidade substituídas por B-Splines e NUBRS.

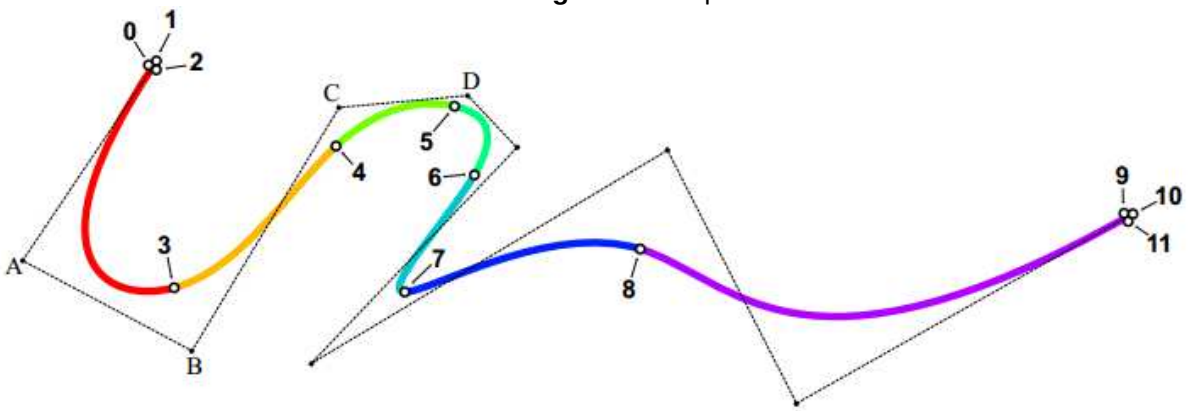
Uma das desvantagens das Curvas de Bérzier é por possuir controle global, em alguns casos pode ser positivo, mas para os propósitos usados por arquitetos navais normalmente é uma desvantagem (BIRK, 2014). A outra grande desvantagem é o fato de o número de vértices estar atrelado ao grau da curva, cada vértice extra adiciona um level de combinação baricêntrica e eleva o grau da curva. Em relação a suavidade e continuidade, cúbicas são seriam suficientes, porém para representar corpos mais complexos aumentaria muito o custo computacional devido as funções base.

2.5.2 Curvas *B-Splines*

Esse tipo de curva começou a ser investigada no começo do século 19 por N. Lobachevsky com distribuições de probabilidade. Em 1946 Isaac Jacob Schoenberg usou B-Splines para ajustar uma curva a dados estatísticos. Esse trabalho marca o início do desenvolvimento da teoria moderna de *splines*. Em 1972 C. de Boor publicou algoritmos recursivos para encontrar as funções bases desse tipo de curva. As curvas do tipo B-Splines e sua extensão, NURBS, são os elementos base na maioria dos sistemas de modelagem 3-D atualmente existentes (BIRK, 2014).

O conceito de funções polinomiais definidas por partes, são as chamadas *splines,* e são a solução para as limitações antes impostas pelas curvas de Bérzier, com *splines* possui-se controle local e o grau da curva não depende do número de vértices. A ideia consiste em dividir a curva $\underline{r}(t)$ em segmentos menores, como destacado por cores na Figura 15.

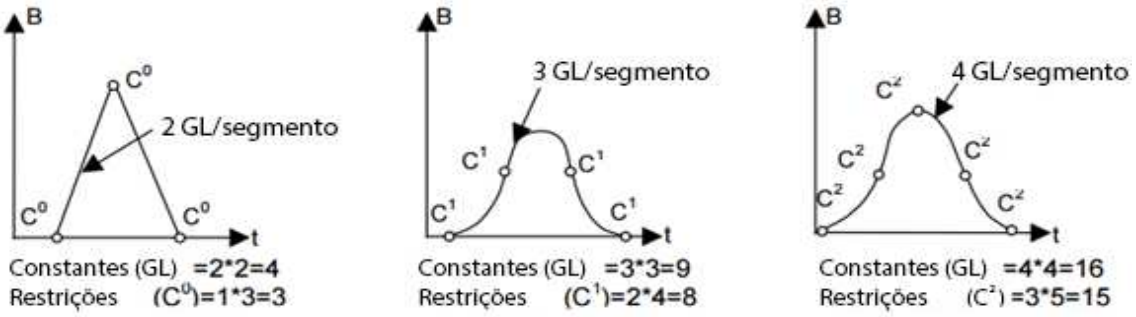
Figura 15: B-Spline



Fonte: Rhino Pythom Primer (Robert McNeel & Associates)

Alguns requerimentos de continuidade são necessários nas transições entre segmentos: para uma função linear, continuidade do tipo C^0, o ponto final do seguimento $r_i(y)$ ser o mesmo do ponto inicial do seguimento $r_{i+1}(t)$; para uma quadrática, C^1, que inclui C^0, e tangentes nos pontos serem iguais; para uma cubica, C^1 e derivativas de segunda ordem, curvatura, serem iguais. Para cada grau de continuidade (GL), uma constante se adiciona a curva, o número de constantes menos o número de constantes de continuidade será um, conforme ilustrado na Figura 16.

Figura 16: Constantes de continuidade segmentos B-Spline



Fonte: Craig (2009) – Adaptada.

São três os parâmetros que definem uma curva do tipo B-Spline: a ordem da curva, *k*, o número de vértices, e um vetor nó. Uma curva do tipo B-Spline é descrita pela Equação 8, e suas funções base são calculadas recursivamente pelas Equações 9 e 10. O comportamento das funções base é ilustrado na Figura 17, note que para qualquer valor de *t*, a soma das funções bases correspondem a 1.

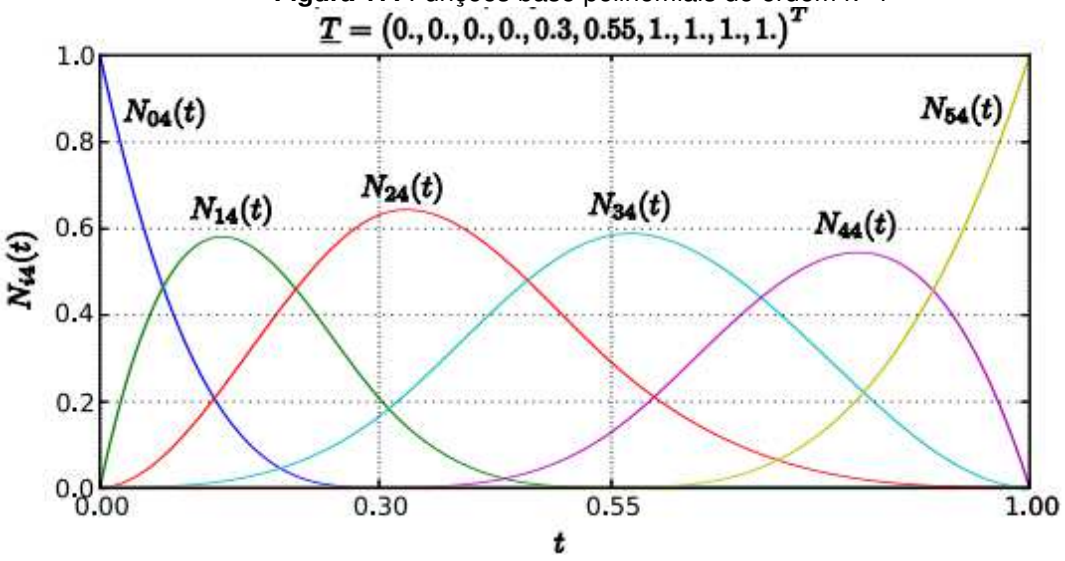
$$\underline{r} = \sum_{i=0}^{n} \underline{V}_i \cdot N_{i,k}(t) \tag{8}$$

$$N_{i,j}(t) = \begin{cases} 1 & \text{for } t \in [t_i, t_{i+1}) \\ 0 & \text{para qualquer outro } t \end{cases} \tag{9}$$

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j-1} - t_i} \cdot N_{i,j+1}(t) + \frac{t_{i+j} - t}{t_{i+j} - t_{i+1}} \cdot N_{i+1,j-1}(t) \quad j \in [2,k] \text{ e } i \in [0,n] \tag{10}$$

Os vértices são enumerados a partir do zero, então $V_0, V_1 \ldots V_n$, logo o número de vértices é igual a $n+1$. A ordem das funções base é j, e $j-1$ é o grau delas, sendo que j vai de 1 a k, sendo o k a ordem da curva $r(t)$. Para propósitos matemáticos, quando usada a Equação 10, "$0/0 = 0$". Uma curva B-Spline possui n-k +2 segmentos e o comprimento do vetor nó é de $n+1+k$ (BIRK, 2014).

Figura 17: Funções base polinomiais de ordem k=4

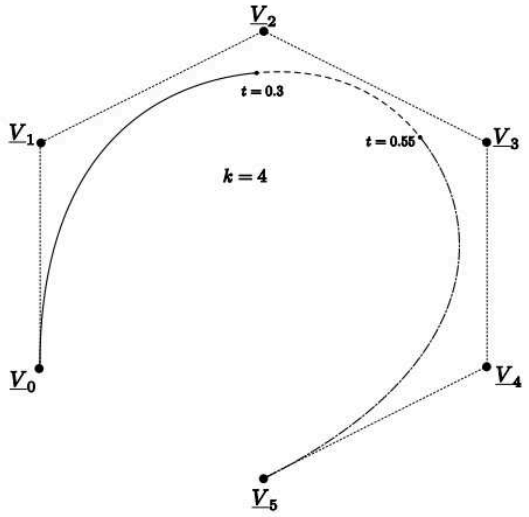


Fonte: Rhino Python Primer (Robert McNeel & Associates)

Um vetor nó, identificado por T na parte superior da Figura 17, é uma sequência crescente de números que irão definir o formato da curva. Para se criar um vetor nó, inicialmente a distância paramétrica da curva deve ser dividida em $n-k+2$ segmentos. Sendo que para um vetor nó uniforme, os intervalos paramétricos deveram ser de igual comprimento, enquanto que para um vetor nó não uniforme, os intervalos podem variar.

Na Figura 18 observa-se a curva B-Spline aberta gerada pelas funções bases ilustradas na Figura 17.

Figura 18: Curva B-Spline conforme funções Figura 17



Fonte: Rhino Pythom Primer (Robert McNeel & Associates)

Além do que ja foi discutido anteriormente, algumas outras propriedades das B-Splines são interessantes de serem apresentadas, são elas:

- Geometria Invariante: devido ao fato da curva ser determinada pelos seus vértices e vetor nó, a mesma não está relacioanda com o sistema de coordenadas, sendo assim invariante a transformação de coordenadas.

- Independência da ordem: já elucidado anteriormente, a ordem não está atrelada ao número de vertices, o único requerimento é que o número de vértices seja maior que a ordem da curva.

- Correspondência a curva de Bézier: caso o número de vértices seja igual a ordem, uma B-Spline se reduz a uma curva de Bézier, com um vetor nó de comprimento igual a ordem e uma curva com apenas um segmento. Deformabilidade local é perdida nesse caso.

- Propriedade dos pontos finais: para que a curva do tipo B-Spline realmente se encerre nos pontos extremos, deve haver uma multiplicidade de vérticies de valor k nesses pontos. Conforme observou-se na Figura 15.

2.5.3. *Non-Uniform Rational B-Spline* (NURBS)

NURBS são consideradas derivativas das *B-Splines*. O *Non-Uniform* em seu nome se refere ao vetor nó que as descreve, podendo estes serem não-uniformes,

porém não precisam estritamente ser, ou seja, NURBS podem também ser construídas com vetores uniformes. B-Splines são compostas por funções polinomiais, enquanto NURBS são compostas por funções racionais, isto é, razões polinomiais. NURBS devem ser usadas apenas em modelagens onde B-Splines polinomiais não conseguem representar a forma desejada. Por exemplo: cônicas ou curvas criadas pela intersecção de planos com diferentes orientações a um cone. A única cônica capaz de ser representada por polinômios é a parábola.

Uma curva do tipo NURBS é representada pela Equação 11:

$$r(t) = \frac{\sum_{i=0}^{n} w_i V_i N_{i,k}(t)}{\sum_{i=0}^{n} w_i N_{i,k}(t)} \quad (11)$$

Os fatores adicionais w_i são chamados de pesos. Sendo assim possível definir a intensidade com que cada ponto de controle influencia no formato da curva. Aumentando-se o peso referente a um ponto, a curva é atraida mais intensamente na direção dele, permitindo assim maior poder de modelagem e flexibilidade.

2.5. TRABALHOS RELACIONADOS

Devido ao fato de que não existem muitas referências a respeito da automatização da modelagem de propulsores, uma breve revisão será feita a respeito dos métodos propostos por quatro diferentes autores, Lim (2001), Zurcher (2009), Fowler (2012) e Boddy (2014) provenientes da *University of South Walles,* Austrália, todos orientados pelo professor P. J. Helmore. Em todos esses trabalhos o hélice apresentado na Tabela 3 foi usado como referência, o que permitiu a comparação de suas modelagens com os dados do propulsor original e entre eles. O propulsor de referência foi projetado pela *Marine Technical Services* (MTS) para a embarcação MV SDB, seu desenho pode ser encontrado no Anexo A.

Tabela 1: B5-60 Referência

Diâmetro:	2438,00 mm
Passo:	1390,00 mm
Número de pás:	5
Relação P/D:	0,57
Área da pá expandida:	2,801 m^2
Razão de áreas expandidas (EAR):	0,60
Material:	Fundido Mn-Al-Bronze
Motorização:	744 kW - 750 rpm
Relação caixa de embreagem:	1:3,02
Massa:	1030 kg
Momento polar de inercia:	238 $kg.m^2$

Fonte: Zucher, 2009

Uma breve apresentação de cada método será feita adiante, de forma cronológica. Os resultados de cada autor serão apresentados e discutidos no Capítulo 5 do presente trabalho.

Lim (2001) desenvolveu em sua tese de mestrado dois métodos usando o software PTC's Pro Engineer, intitulados por ele por "*importing*" e "*generating*". O método "*importing*" consiste em uma tabela de cotas obtidas através do desenho do propulsor original, uma planilha do Microsoft Excel para transformar as cotas em coordenadas 3D e uma rotina criada e executada pelo software. O usuário manualmente conforma as seções ao redor do cilindro para cada fração de raio e o software desenha o propulsor em 3D. Já o método "*generating*" consiste em mais operações serem executadas no ProEngineering, porém o usuário precisa informar características adicionais para ajudar o processo de modelagem, como ponto de partida dos planos para cada fração de raio, linha de caimento e pontos de relação. Seu método de automatização não foi divulgado abertamente.

Zurcher (2009) desenvolveu seu trabalho de bacharelado utilizando Microsoft Excel e software CATIA. O usuário entra com os dados de entrada como diâmetro, passo, número de pás, razão de áreas expandidas, caimento, material e dimensões do cubo. Uma combinação de equações do Excel e códigos do tipo *Visual Basic for Applications (*VBA) calculam a geometria para iniciar a modelagem automatizada. Assim como Lim (2001), seu método não foi divulgado ao público.

Fowler (2012) desenvolveu seu trabalho de bacharelado utilizando os programas Microsoft Excel e AUTOCAD. Como por Zurcher (2009) a geometria é calculada por uma planilha do Excel, porém o processo de modelagem automatizado foi programado em *Visual Basic .NET* (VB .NET). Como dados de saída tem-se o

propulsor modelado em 3D e informações sobre massa e momento de inércia. Seu método de automatização foi divulgado em seu trabalho.

Boddy (2014) desenvolveu seu trabalho de bacharelado apenas na área de modelagem. A geometria usada por ele é proveniente dos autores citados anteriormente. É considerado um processo semi-automatizado, utiliza-se do plug-in não nativo do Rhinoceros, Grasshopper, como interface gráfica para a modelagem.

3. METODOLOGIA

O desenvolvimento deste trabalho foi realizado majoritariamente em um notebook com as especificações técnicas descritas pela Tabela 1, para o mesmo desenvolvimento foram-se usados os softwares descritos na Tabela 2 em suas respectivas versões.

Tabela 2: Hardware

Fabricante/Modelo	Lenovo Y-40
Sistema Operacional	Windows 10 Home
Arquitetura Sistema	64-bit, Processador x64
Processador	Intel® Core™ i7-4510 CPU@ 2.00GHz 2.60GHz
Memória (RAM)	8 GB

Fonte: Autor (2016)

Tabela 3: Softwares

MATLAB	R2015a (8.5.0 .197613)
Rhinoceros	Version 5 SR12 64-bit (5.12.50810.13095, 8/10/2015)

Fonte: Autor (2016)

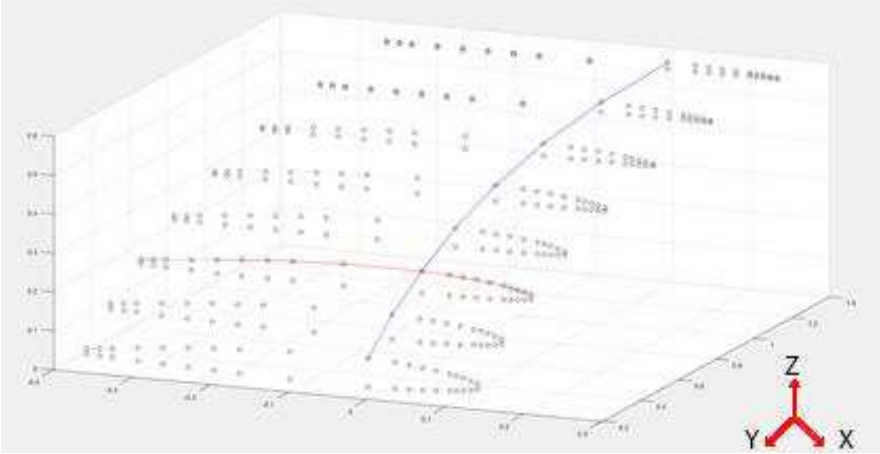
Inicialmente a programação de uma rotina em MATLAB será responsável pela criação da geometria do propulsor a ser modelado. Essa rotina será capaz de abranger todos os propulsores das séries sistemáticas B e BB. Para essa geração serão usadas as tabelas, coeficientes e equações propostas por Kuipper (1992). Os dados de entrada informados pelo usuário serão: diâmetro do propulsor, razão de área expandida, número de pás e passo.

A fim de se obter maior precisão na posterior modelagem 3D serão gerados pontos adicionais para descrever melhor as superfícies de sucção e de pressão do propulsor. Para essa etapa a *Toolbox* do MATLAB, *Curve Fitting*, será utilizada afim de encontrar a melhor equação que represente as formas geradas inicialmente. Para isso serão testadas diferentes curvas de tendência, procurando-se a que melhor se enquadre aos pontos gerados inicialmente.

Essas curvas de tendência são curvas ajustadas aos pontos originais e serão geradas tanto para os pontos que descrevem as secções nos eixos X-Z quanto em X-Y-X. Na Figura 19, observa-se pontos conectados por linhas definindo a forma da face de sucção da seção r/R =0.4, e também pontos conectados no

espaço entre seções. A curva de tendência para os pontos nos eixos X-Z irão ser usadas para suavizar as formas das seções e criar mais pontos para descreve-las quando forem usadas na modelagem, enquanto que a curva de tendência para os pontos entre as seções no espaço, serão usados para gerar novas seções em frações de raios intermediários.

Figura 19: Interpolação geométrica secções



Fonte: Autor (2016)

O método e apresentação das curvas ajustadas para essas interpolações serão discutidas adiante na etapa de desenvolvimento da rotina de geração de geometria pelo MATLAB.

Conforme dito anteriormente, a partir das curvas de tendência e suas equações, serão calculados os pontos adicionais para representar secções intermediárias as dadas pelas tabelas de Kuipper (1992), dessa forma garantindo uma metodologia matemática mais minuciosa e conhecida para a criação das curvas que darão origem as superfícies, não deixando as mesmas a cargo de funções de código fechado do software CAD.

Os coeficientes não existentes nas tabelas de Kuipper (1992) serão extrapolados afim de obtermos as mesmas quantidades de dados sobre cada secção.

A próxima etapa será a modelagem 3D automatizada do propulsor. A modelagem se dará a partir da nuvem de pontos exportada pelo MATLAB e importada pelo Rhinoceros com apenas o caimento aplicado, ou seja, seções planificadas. A aplicação do passo, rotação e conformação cilíndrica das seções serão feitas no processo de modelagem no Rhinoceros.

A automatização da modelagem será programada em uma rotina usando o ambiente de programação por rotina interno do Rhinoceros, o mesmo pode ser aberto pelo comando *EditPythonScript*, o plug-in responsável pela mesma é padrão do Rhinoceros 3D, sendo a linguagem Iron Python.

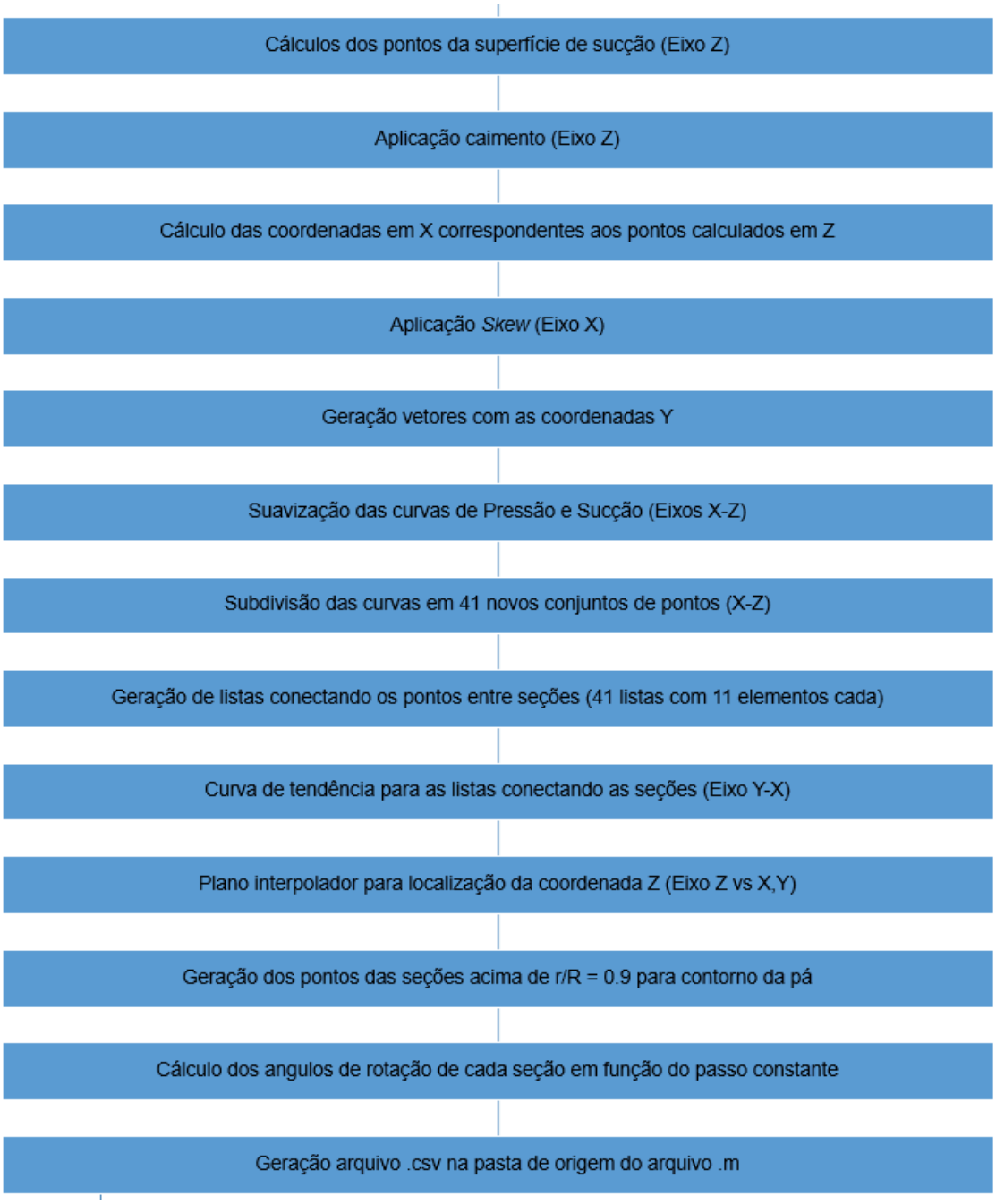
Afim de se obter uma validação de todo o desenvolvimento proposto, será usado um propulsor base já estudado por outros autores citados na revisão bibliográfica. A modelagem do mesmo será através da rotina proposta o valor de massa do hélice modelado será comparado com os originais de projeto, Anexo A, e cruzadas com as mesmas obtidas por outros autores em suas respectivas modelagens.

Inspeções visuais a respeito da suavidade das curvas também serão levadas em consideração para avaliar o êxito da proposta. O Rhinoceros possui ferramentas para fazer esta avaliação e considerações sobre o assunto serão apresentadas nos resultados. O comportamento da curva de áreas seccionais também será analisado ao final da modelagem.

4. DESENVOLVIMENTO

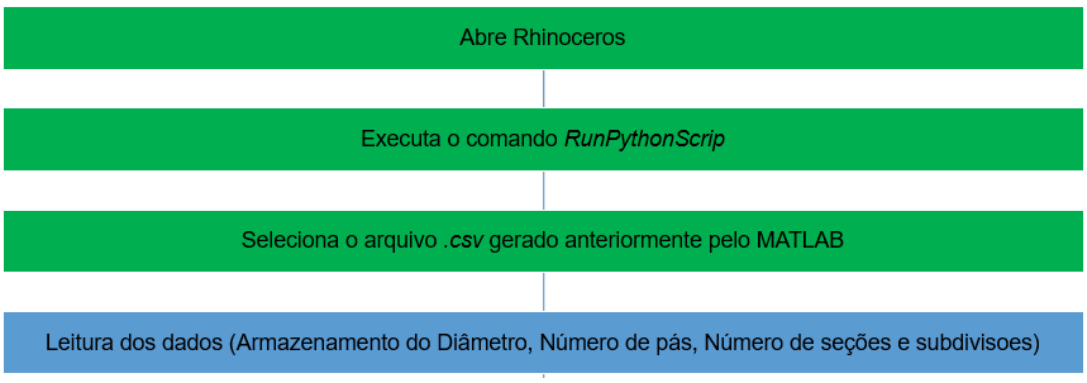
O processo de geração de geometria e modelagem automatizada serão descritos de forma minuciosa nos tópicos a seguir. As Figuras 20 e 21 apresentam fluxogramas com o objetivo de ilustrar rapidamente os processos desenvolvidos pelo MATLAB e Rhinoceros. As caixas em verde são as tarefas de responsabilidade do usuário e em azul os processos automatizados. As imagens usadas para ilustrar os processos nos tópicos seguintes são da geração geométrica e modelagem do propulsor B4-85 com as seguintes características: relação P/D = 0.9 e EAR = 0.85.

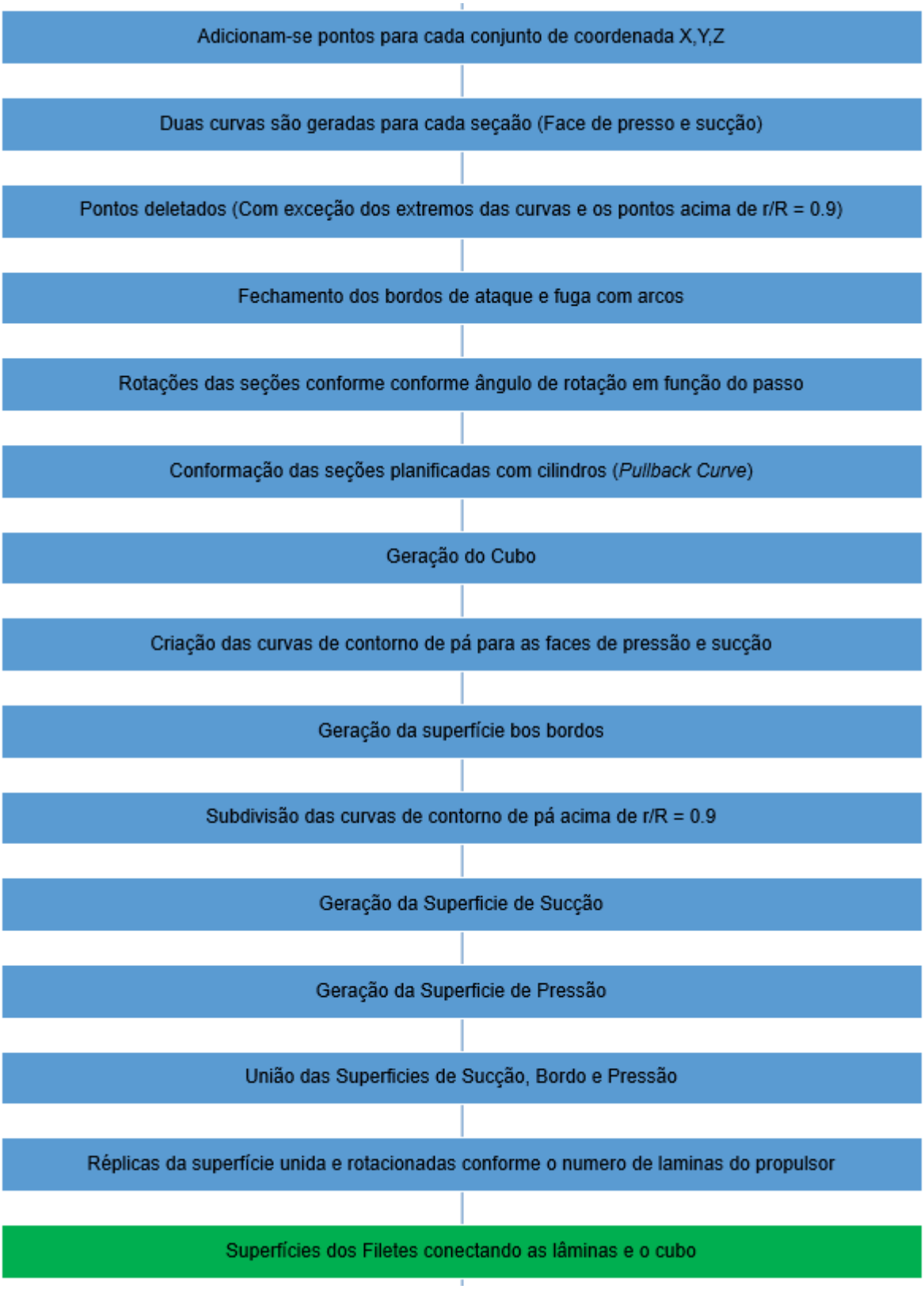
Figura 20: Fluxograma MATLAB.



Fonte: Autor (2016)

Figura 21: Fluxograma Rhinoceros.



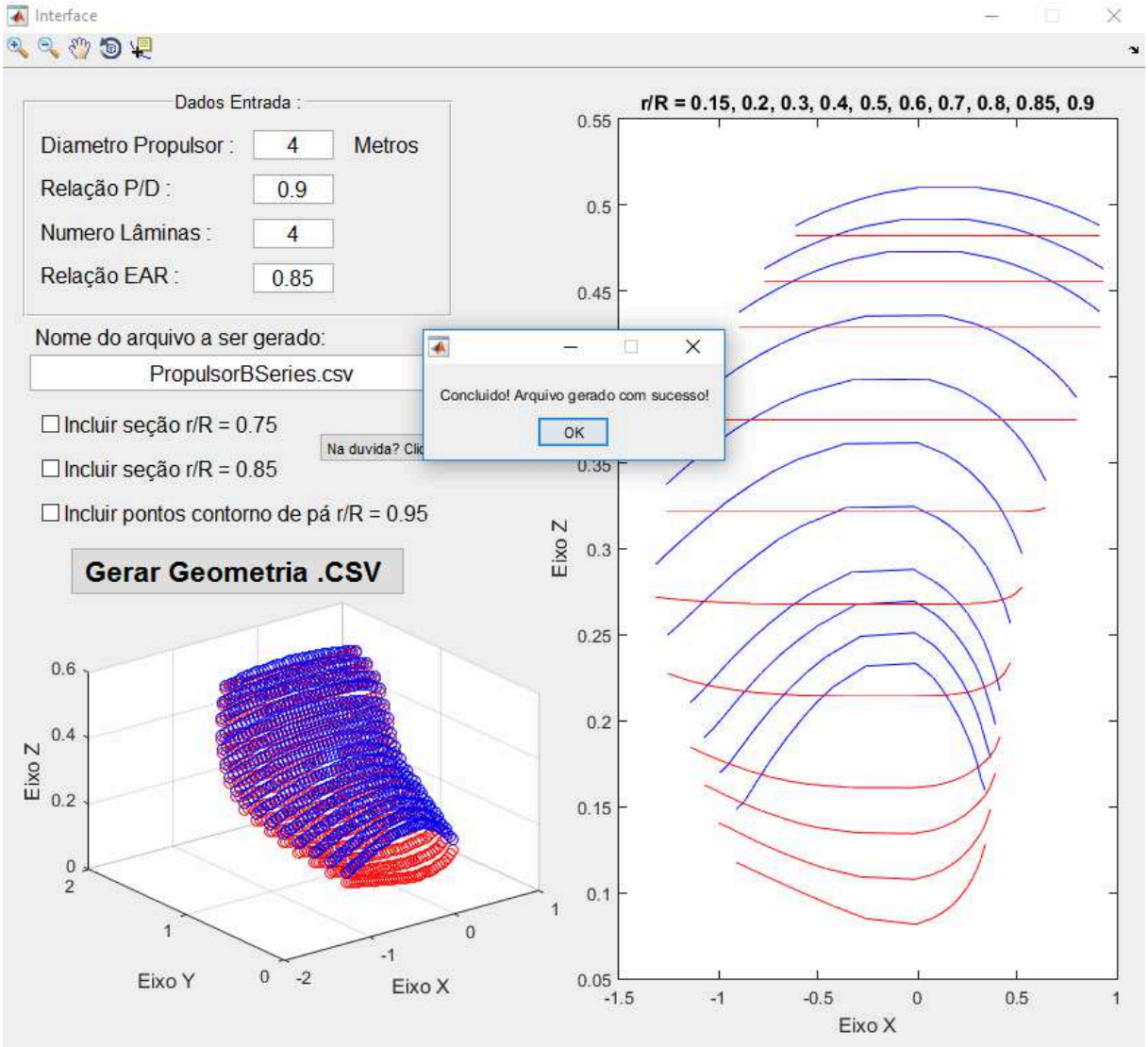


Fonte: Autor (2016)

4.1. GERAÇÃO DA GEOMETRIA NO MATLAB

A rotina do MATLAB é responsável por receber os dados de entrada do usuário em uma interface gráfica chamada *GUIDE,* sendo eles: diâmetro hélice, razão de áreas expandidas (EAR), número de pás e razão passo sobre diâmetro (P/D), e como saída ocorre a geração de um arquivo *.csv* salvo na mesma pasta que o arquivo Interface.m foi executado. Na Figura 22 pode-se observar a interface gráfica. Após a execução as seções são plotadas em 2D e 3D e abre-se uma janela de mensagem informando o usuário que o processo foi concluído.

Figura 22: Interface Gráfica MATLAB



Fonte: Autor (2016)

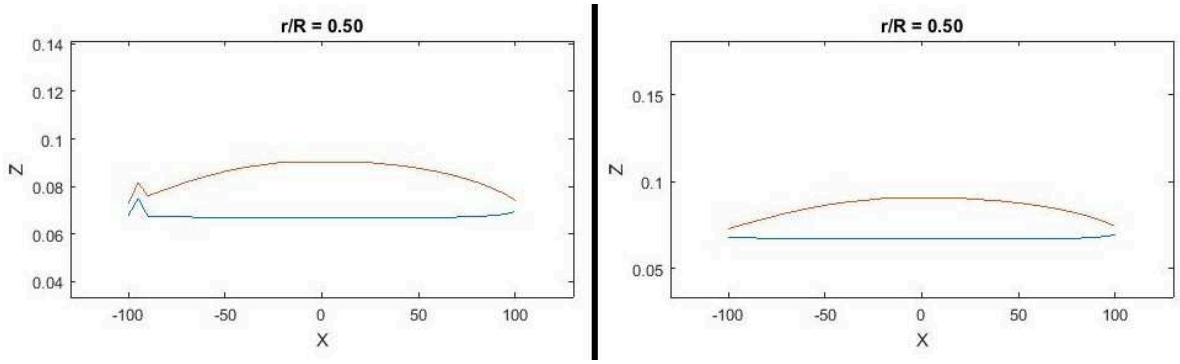
Para execução correta da rotina com interface gráfica, os arquivos Interface.m e Interface.fig devem estar na mesma pasta, e deve-se executar primeiramente o arquivo Interface.m. A rotina de programação está associada ao arquivo Interface.m, portanto o arquivo Interface.m é responsável por chamar o arquivo Interface.fig. Caso o arquivo Interface.fig seja executado primeiramente apenas uma janela gráfica sem funcionalidade será executada.

O código completo do arquivo Interface.m pode ser encontrado no Apêndice A. Para facilitar a leitura do código e descrever o processo que é executado pelo código foram inseridos comentários ao longo do programa.

Após clicar no botão “Gerar Geometria .CSV” a geração da geometria pela rotina segue como foi descrita anteriormente na revisão bibliográfica a respeito das séries B de Wageningen. As tabelas propostas por Kuipper (1992) foram programadas como variáveis, assim como as operações necessárias para o desenolvimento. As numerações da publicação original foram mantidas no código para facilitar qualquer conferência.

Foram feitas duas modificações as tabelas originais de Kuipper (1992) inicilamente, pois foi constatado um possível erro de digitação na Tabela 4.7, para 95% e r/R = 0.5. O valor de 0.42 resultava em um ponto fora da curva, como pode ser observado na Figura 23. Na esquerda da Figura 23 o uso do valor 0.42 e na direita a modificação desse valor para 0.042, modificação essa também feita nas tabelas de coeficientes para B-Series apresentadas por Carlton (2007), Figura 24.

Figura 23: Correção seção r/R = 0.5



Fonte: Autor (2016)

Figura 24: Coeficientes V_1

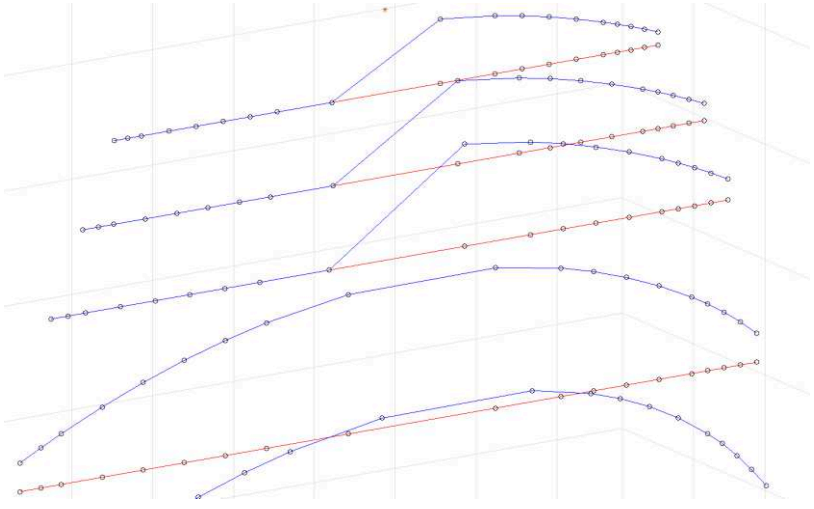
r/R P	−1.0	−0.95	−0.9	−0.8	−0.7	−0.6	−0.5	−0.4	−0.2	0
0.7–1.0	0	0	0	0	0	0	0	0	0	0
0.6	0	0	0	0	0	0	0	0	0	0
0.5	0.0522	0.0420	0.0330	0.0190	0.0100	0.0040	0.0012	0	0	0
0.4	0.1467	0.1200	0.0972	0.0630	0.0395	0.0214	0.0116	0.0044	0	0
0.3	0.2306	0.2040	0.1790	0.1333	0.0943	0.0623	0.0376	0.0202	0.0033	0
0.25	0.2598	0.2372	0.2115	0.1651	0.1246	0.0899	0.0579	0.0350	0.0084	0
0.2	0.2826	0.2630	0.2400	0.1967	0.1570	0.1207	0.0880	0.0592	0.0172	0
0.15	0.3000	0.2824	0.2650	0.2300	0.1950	0.1610	0.1280	0.0955	0.0365	0

r/R P	+1.0	+0.95	+0.9	+0.85	+0.8	+0.7	+0.6	+0.5	+0.4	+0.2	0
0.7–1.0	0	0	0	0	0	0	0	0	0	0	0
0.6	0.0382	0.0169	0.0067	0.0022	0.0006	0	0	0	0	0	0
0.5	0.1278	0.0778	0.0500	0.0328	0.0211	0.0085	0.0034	0.0008	0	0	0
0.4	0.2181	0.1467	0.1088	0.0833	0.0637	0.0357	0.0189	0.0090	0.0033	0	0
0.3	0.2923	0.2186	0.1760	0.1445	0.1191	0.0790	0.0503	0.0300	0.0148	0.0027	0
0.25	0.3256	0.2513	0.2068	0.1747	0.1465	0.1008	0.0669	0.0417	0.0224	0.0031	0
0.2	0.3560	0.2821	0.2353	0.2000	0.1685	0.1180	0.0804	0.0520	0.0304	0.0049	0
0.15	0.3860	0.3150	0.2642	0.2230	0.1870	0.1320	0.0920	0.0615	0.0384	0.0096	0

Fonte: Carlton (2007)

A segunda modificação foi feita na Tabela 4.9 de Kuipper (1992) e consistiu em adotar os mesmos coeficientes V_2 de r/R = 0.7 para as seções acima. Sem essa adaptação as seções acima de r/R = 0.7 ficariam como as apresentadas na Figura 25. Na Figura 26 apresenta-se a tabela de coeficientes de V_2 fornecida por Carlton (2007).

Figura 25: Seções acima de r/R = 0.7 sem adaptação.



Fonte: Autor (2016)

Figura 26: Coeficientes V_2

Values of V_2 for use in the equations

r/R	P	−1.0	−0.95	−0.9	−0.8	−0.7	−0.6	−0.5	−0.4	−0.2	0
0.9–1.0		0	0.0975	0.19	0.36	0.51	0.64	0.75	0.84	0.96	1
0.85		0	0.0975	0.19	0.36	0.51	0.64	0.75	0.84	0.96	1
0.8		0	0.0975	0.19	0.36	0.51	0.64	0.75	0.84	0.96	1
0.7		0	0.0975	0.19	0.36	0.51	0.64	0.75	0.84	0.96	1
0.6		0	0.0965	0.1885	0.3585	0.5110	0.6415	0.7530	0.8426	0.9613	1
0.5		0	0.0950	0.1865	0.3569	0.5140	0.6439	0.7580	0.8456	0.9639	1
0.4		0	0.0905	0.1810	0.3500	0.5040	0.6353	0.7525	0.8415	0.9645	1
0.3		0	0.0800	0.1670	0.3360	0.4885	0.6195	0.7335	0.8265	0.9583	1
0.25		0	0.0725	0.1567	0.3228	0.4740	0.6050	0.7184	0.8139	0.9519	1
0.2		0	0.0640	0.1455	0.3060	0.4535	0.5842	0.6995	0.7984	0.9446	1
0.15		0	0.0540	0.1325	0.2870	0.4280	0.5585	0.6770	0.7805	0.9360	1

r/R	P	+1.0	+0.95	+0.9	+0.85	+0.8	+0.7	+0.6	+0.5	+0.4	+0.2	0
0.9–1.0		0	0.0975	0.1900	0.2775	0.3600	0.51	0.6400	0.75	0.8400	0.9600	1
0.85		0	0.1000	0.1950	0.2830	0.3660	0.5160	0.6455	0.7550	0.8450	0.9615	1
0.8		0	0.1050	0.2028	0.2925	0.3765	0.5265	0.6545	0.7635	0.8520	0.9635	1
0.7		0	0.1240	0.2337	0.3300	0.4140	0.5615	0.6840	0.7850	0.8660	0.9675	1
0.6		0	0.1485	0.2720	0.3775	0.4620	0.6060	0.7200	0.8090	0.8790	0.9690	1
0.5		0	0.1750	0.3056	0.4135	0.5039	0.6430	0.7478	0.8275	0.8880	0.9710	1
0.4		0	0.1935	0.3235	0.4335	0.5220	0.6590	0.7593	0.8345	0.8933	0.9725	1
0.3		0	0.1890	0.3197	0.4265	0.5130	0.6505	0.7520	0.8315	0.8020	0.9750	1
0.25		0	0.1758	0.3042	0.4108	0.4982	0.6359	0.7415	0.8259	0.8899	0.9751	1
0.2		0	0.1560	0.2840	0.3905	0.4777	0.6190	0.7277	0.8170	0.8875	0.9750	1
0.15		0	0.1300	0.2600	0.3665	0.4520	0.5995	0.7105	0.8055	0.8825	0.9760	1

Fonte: Carlton (2007)

A Figura 24 contém os coeficientes das Tabelas 4.7 e 4.8 de Kuipper (1992), e a Figura 26 contém os coeficientes das Tabelas 4.9 e 4.10 de Kuipper (1992). Elas serão usadas para explicar o cálculo do formato das seções mais adiante. O processo de geração será descrito por etapas, conforme as etapas foram apresentadas no fluxograma da Figura 20.

A primeira etapa executada pela rotina após o usuário informar os dados de entrada do propulsor e mandar gerar o arquivo é a interpolação e extrapolação de algumas das tabelas originais de Kuipper (1992). Conforme observa-se pelas Figuras 24 e 26, existem coeficientes V_1 e V_2 para as razões r/R = 0.15, 0.25 e 0.85, porém não existem correspondência para essas frações nas Tabelas apresentadas pela Figura 27.

Figura 27: Coeficientes V_2

r/R	$K(r)$	$skew/c_r$
0.2	1.600	0.081
0.3	1.832	0.084
0.4	2.023	0.080
0.5	2.163	0.070
0.6	2.243	0.052
0.7	2.247	0.024
0.8	2.132	-0.020
0.85	2.005	-0.052
0.9	1.798	-0.098
0.95	1.434	-0.182
0.975	1.220	-0.273

Tabela 4.2 - Contorno Pá Série-BB

radius	A_r	B_r
0.2	0.0526	0.0040
0.3	0.0464	0.0035
0.4	0.0402	0.0030
0.5	0.0340	0.0025
0.6	0.0278	0.0020
0.7	0.0216	0.0015
0.8	0.0154	0.0010
0.9	0.0092	0.0005
1.0	0.0030	0.0000

Tabela 4.12 - Coeficientes para distribuição de espessuras

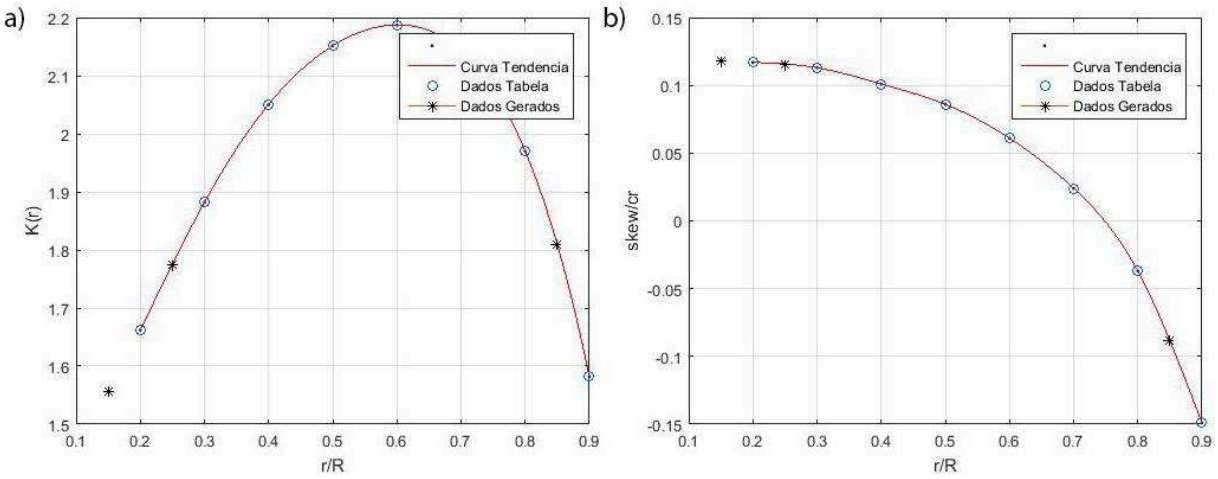
r/R	t_{max}/D					x_{tmax}/c_r
	3-bladed	4-bladed	5-bladed	6-bladed	7-bladed	
.2	.0406	.0366	.0326	.0286	.0246	.350
.3	.0359	.0324	.0289	.0254	.0219	.350
.4	.0312	.0282	.0252	.0222	.0192	.351
.5	.0265	.0240	.0215	.0190	.0165	.355
.6	.0218	.0198	.0178	.0158	.0138	.389
.7	.0171	.0156	.0141	.0126	.0111	.443
.8	.0124	.0114	.0104	.0094	.0084	.486
.9	.0077	.0072	.0067	.0062	.0057	.500
1.0	.0030	.0030	.0030	.0030	.0030	.500

Tabela 4.11 - Maior espessura e posição da maior espessura

Fonte: Kuipper (1992) – Adaptado

As curvas ajustadas para os coeficientes de *K(r)* e *skew/cr* da Tabela 4.2 de Kuipper (1992) são apresentados na Figura 28(a) e 28(b) respectivamente, foram extrapolados coeficientes para r/R = 0.15 e interpolados para r/R = 0.25.

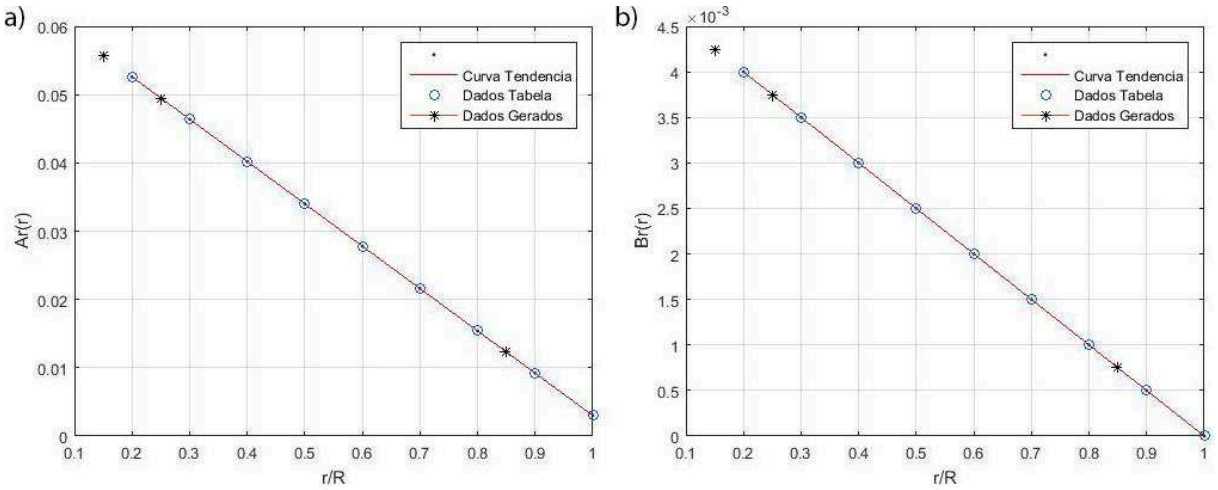
Figura 28: Curvas ajustadas - Tabela 4.2 de Kuipper (1992).



Fonte: Autor (2016)

Seguindo o mesmo procedimento, um ajuste linear foi efetuada para os coeficientes da Tabela 4.12 de Kuipper (1992) como observa-se na Figura 29 e os valores r/R = 0,25 e 0,85 foram interpolados enquanto r/R = 0,15 foi extrapolado.

Figura 29: Retas interpoladas – Tabela 4.12



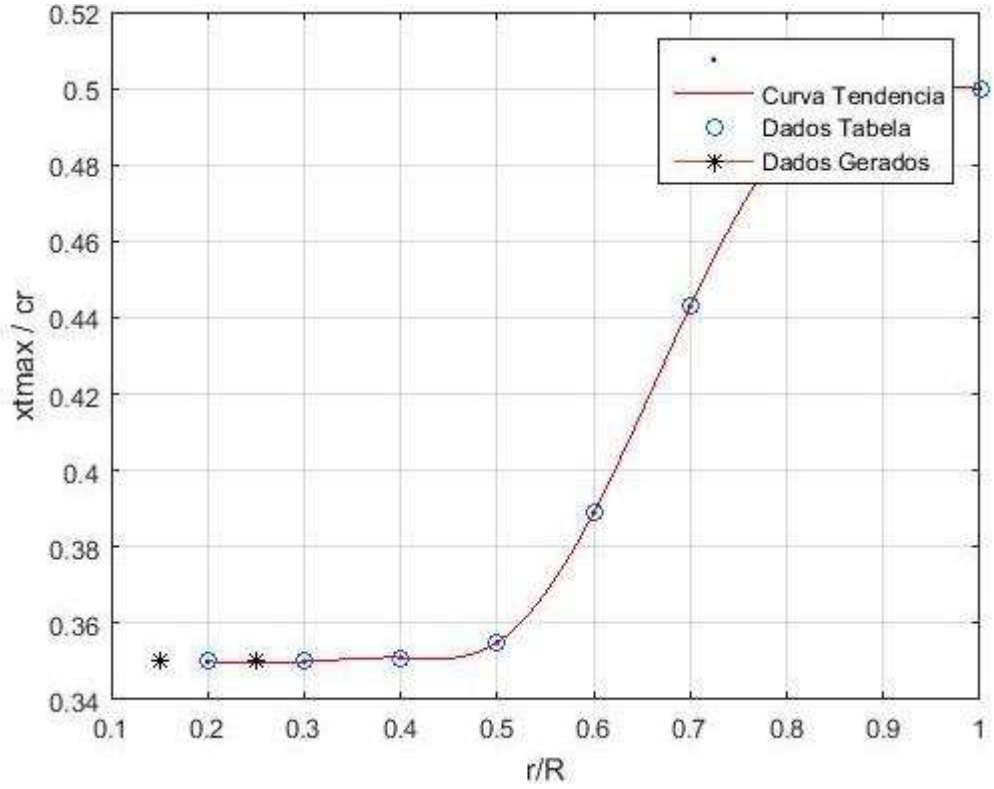
Fonte: Autor (2016)

Parte da Tabela 4.11 de Kuipper (1992) é redundante a Tabela 4.12 de Kuipper (1992), os coeficientes apresentandos na Tabela 4.11 para cada pá resultam na mesma distribuição de espessuras que quando usados os coeficientes da Tabela 4.12 com a Equação 12. Portanto para fins de programação, optou-se pelo uso da Tabela 4.12 para calcular a distribuição de espessuras. Na Equação 12, t_r é espessura propriamente dita, D o diâmetro do propulsor, Z o numero de pás e Ar e Br são os coeficientes da Tabela 4.12.

$$\frac{t_r}{D} = A_r - B_r Z \quad (12)$$

Porém, a ultima coluna da Tabela 4.11 de Kuipper (1992) é essencial para localizar o ponto onde se localiza a maior espessura da seção. Portanto esses valores forem ajustados pela curva apresentada na Figura 30. Porém apenas para r/R = 0.85 foi feita uma interpolação, os coeficientes para r/R = 0.15 e 0.25 foram assumidos como iguais aos de r/R = 0.20 e 0.30, xtmax/cr = 0.350. Isso significa que os pontos de espessura máximos até a fração de raio r/R = 0.30 estão localizados a 35% do comprimento de corda da seção em relação ao bordo de ataque.

Figura 30: Curva ajustada xtmax/cr – Tabela 4.11



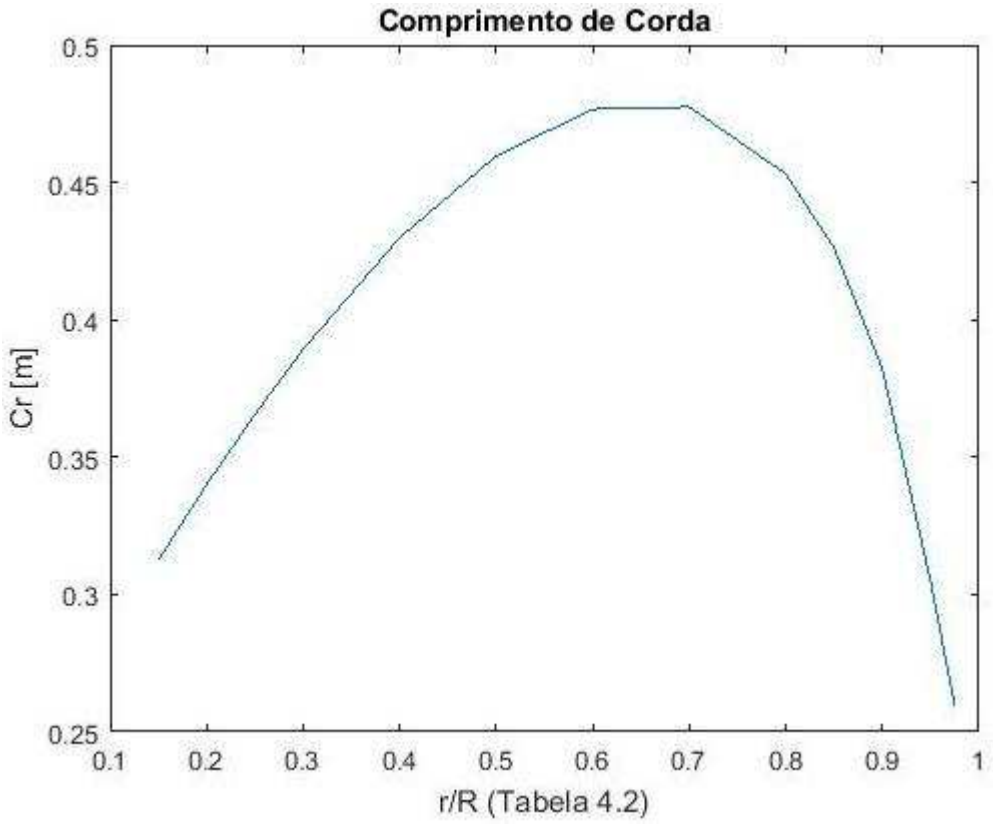
Fonte: Autor (2016)

A próxima etapa executada pela rotina é o cálculo dos comprimentos de corda de cada seção através da Equação 13. Onde K(r) é proveniente da Tabela 4.2 de Kuipper (1992) e D,EAR e Z são dados de entrada que foram informados pelo usuário na interface gráfica.

$$c(r) = \frac{K(r)\ x\ D\ x\ EAR}{Z} \quad (13)$$

Nessa etapa poderia ser usado tanto a Tabela 4.1 quanto 4.2 de Kuipper (1992), porém, por recomendação de Kuipper (1992) o uso da Tabela 4.2 causa uma variação no contorno de pá favorável aos critérios de cavitação sem afetar as características propulsivas do propulsor, sendo assim essa será usada no decorrer do trabalho. Pelo uso da Tabela 4.12 o propulsor gerado é em teoria um propulsor do tipo BB e não B, porém usualmente, não se usa o termo BB. Os resultados são apresentados na Figura 31.

Figura 31: Comprimento de Corda em função de r/R



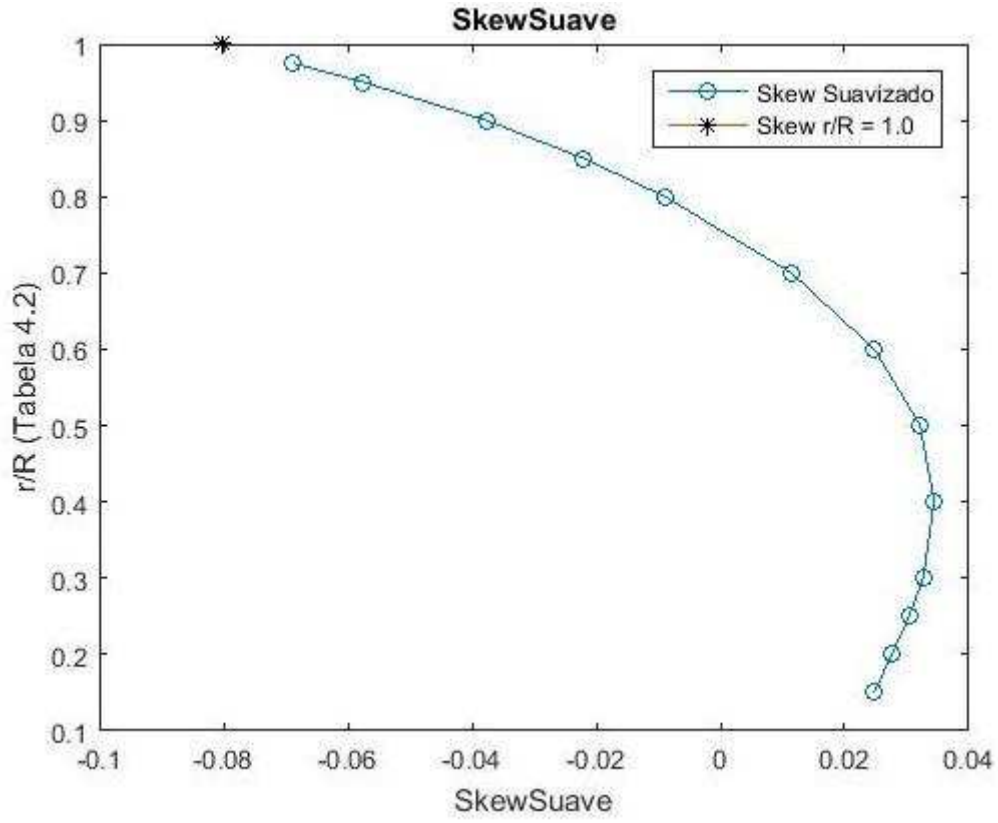
Fonte: Autor (2016)

A etapa seguinte consiste no cálculo dos valores de skew. Usando novamente a Tabela 4.2 de Kuipper (1992), a coluna *skew/cr* é multiplicada pelos valores de *cr* calculados na etapa anterior. Devido ao uso de casas decimais limitadas nos coeficientes das tabelas, é possível que os arredondamentos da última casa decimal cause flutuações pontuais e não deixem a curva de assimetria de contorno de pá tão suave como poderia.

Como os valores de s*kew* serão aplicados a todas as seções, este influenciam diretamente no contorno de pá. Com o objetivo de atingir um contorno

da pá com uma melhor curvatura, os valores de *skew* calculados foram ajustados por uma curva do tipo *Spline* mais suave e recalculados para cada seção de r/R, além de extrapolado para o valor de r/R = 1.0, Figura 32.

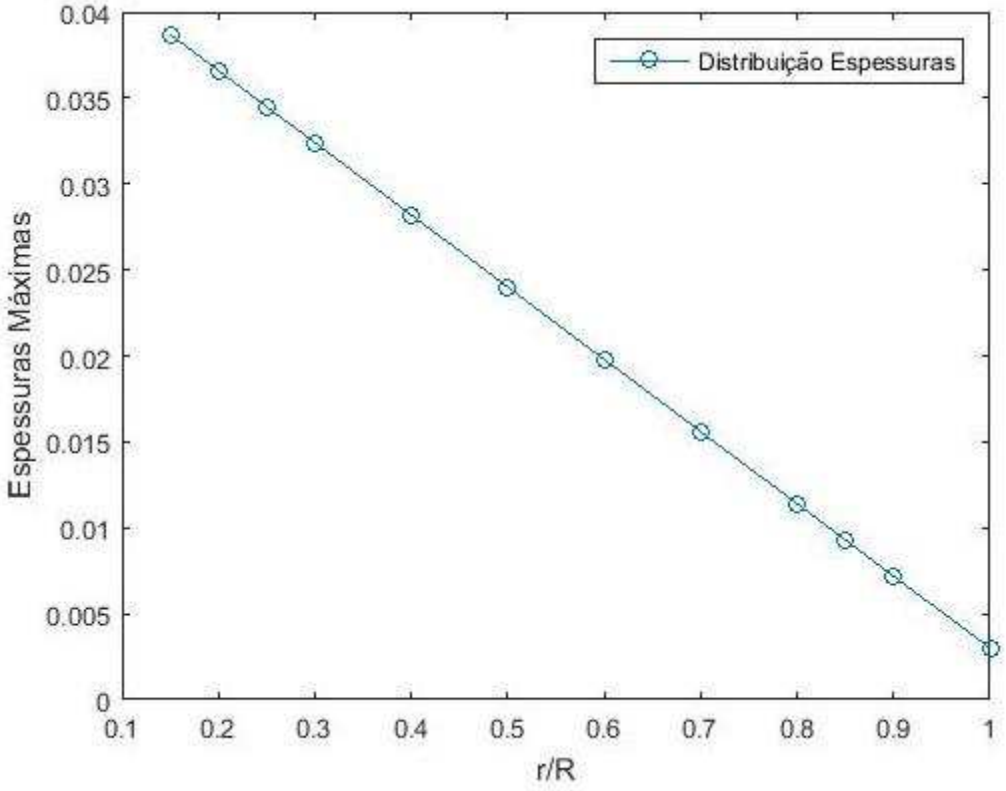
Figura 32: Skew B4-85



Fonte: Autor (2016)

A próxima etapa executada é o cálculo da distribuições de espessuras máximas das seções com a Equação 12 e Tabela 4.12 de Kuipper (1992). As espessuras máximas de cada seção descrescem linearmente conforme r/R cresce até r=R e é apresentada na Figura 33.

Figura 33: Espessuras Máximas – B4-85.



Fonte: Autor (2016)

Na próxima etapa calcula-se efetivamente a geometria de cada seção. Ao calcular os pontos no eixo X-Z obtêm-se a forma das faces de pressão e sucção para cada seção. Nessa etapa são usadas as Equações 1 e 2, e o procedimento descrito na revisão bibliográfica a respeito da geometria das séries B, seção 2.3.1 do Capitulo 2. Os coeficientes V1 e V2 utilizados nas Equações 1 e 2 são os encontrados nas Figuras 24 e 26.

São calculados os pontos no Eixo Z para cada seção, r/R, e para o bordo de fuga e ataque, considerando o 0 como o ponto de máxima espessura. Pela Equação 1, calculam-se as posições dos pontos do lado de pressão, e pela Equação 2 as espessuras para cada ponto, ao somá-los obtém-se a posição do lado de sucção.

Após a localização dos pontos que descrevem cada uma das faces de cada fração de r/R o caimento é aplicado sobre todos os pontos na coordenada Z. Séries B sempre possuem um ângulo de caimento igual a 15 graus, portanto o caimento para qualquer r/R pode ser calculado pela Equação 14. Os valores de caimento são positivos e somados ao ponto calculado anteriormente.

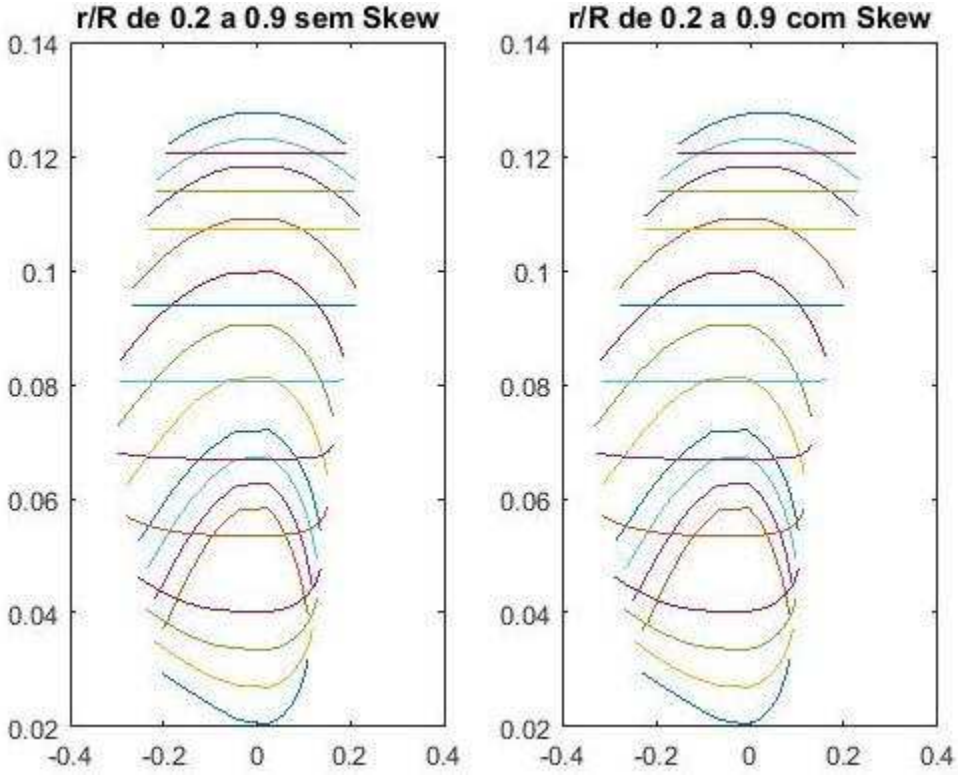
$$Rake\left(\frac{r}{R}\right) = \tan(15\ graus) * \frac{r}{R} * \frac{D}{2} \quad (14)$$

Após a localização dos pontos no eixo Z, resta saber a localização correspondente desses pontos no eixo X. Nessa etapa usam-se os valores de *xtmax/cr* da Tabela 4.11 de Kuipper (1992) para localizar o ponto de espessura máxima no eixo X. O ponto de espessura máxima é tomado como 0%.

As coordenadas de X dadas em porcentagens da corda da seção, entre o ponto de espessura máxima e o bordo de ataque eles estão distribuindos da seguinte maneira: 20%, 40%, 50%, 60%, 70%, 80%, 90%, 95% e 100%. Entre o ponto de espessura máxima e o bordo de fuga eles estão distribuídos em -20%, -40%, -50%, -60%, -70%, -80%, -85%, -90%, -95% e -100% .

Após a multiplicação das porcentagens com os respectivos comprimentos de corda obtemos as coordenadas em X reais de cada um dos 19 pontos que representam as curvas do lado de pressão e sucção. Com um procedimento análogo ao da aplicação do caimento no eixo Z, o *skew,* é aplicado a todos os pontos de cada seção na coordenada X. No entando, os valores de *skew* são subtraídos e não somados. O efeito da aplicação dos *skew* nas seções é ilustrado pela Figura 34.

Figura 34: Efeito do *Skew*

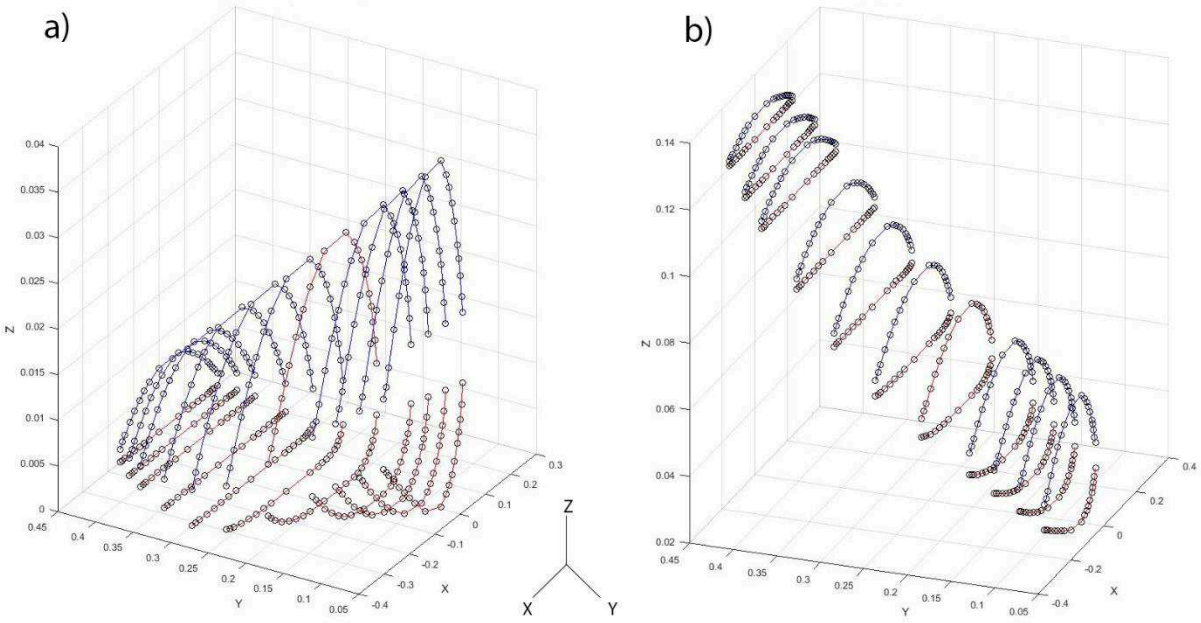


Fonte: Autor (2016)

A próxima etapa é a geração dos valores de Y, esses valores são conhecidos, já que são as posição das seções r/R e estamos trabalhando com seções planificadas, isto é, todos os pontos da mesma seção possuem mesmo valor na coordenada Y. Vetores de zeros com comprimentos iguais a 19 são criados e soma-se o valor da multiplição de cada r/R por R, obtendo-se a as coordendas das seções no eixo Y. Agora, cada um dos 19 pontos já possui um conjunto de coordenadas X,Y,Z.

Com a geração dos conjuntos de coordenadas X,Y,Z é possível que sejam plotados os pontos em três dimensões, na Figura 35 ilustra-se a diferença entre a aplicação ou não do caimento nas coordenadas Z. A Figura 35(a) não teve caimento aplicado e a Figura 35(b) tem caimento, ambas estão com o *skew* já aplicados.

Figura 35: Efeito do caimento



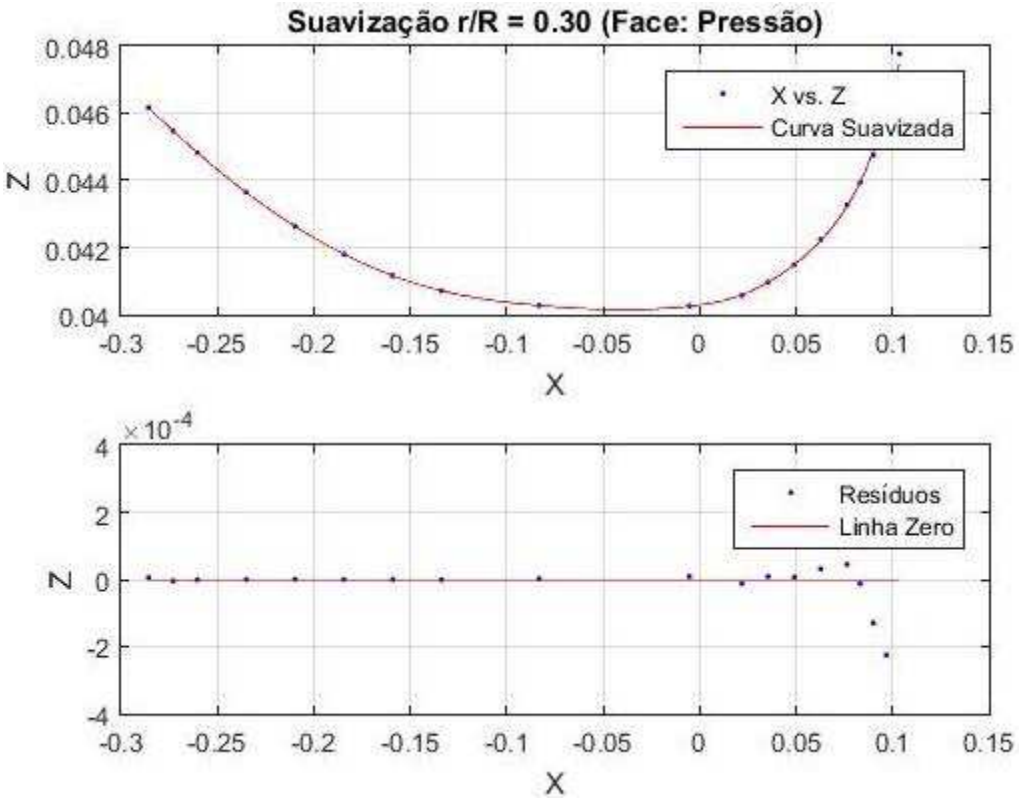
Fonte: Autor (2016)

A próxima etapa parte do mesmo princípio usado para na suavização da curva de *skew* da pá. Com o intuito de termos seções mais suavizadas, sem mudanças bruscas, foram-se efetuadas curvas de tendência usando *splines* suaves, com resíduos máximos na ordem de 0.1 milímetros para um hélice com diâmetro igual a 4m.

Com a *spline* ajustada aos pontos originais foram gerados mais pontos para descrever a curva, de 19 para 41 pontos. Com maior número de pontos descrevendo

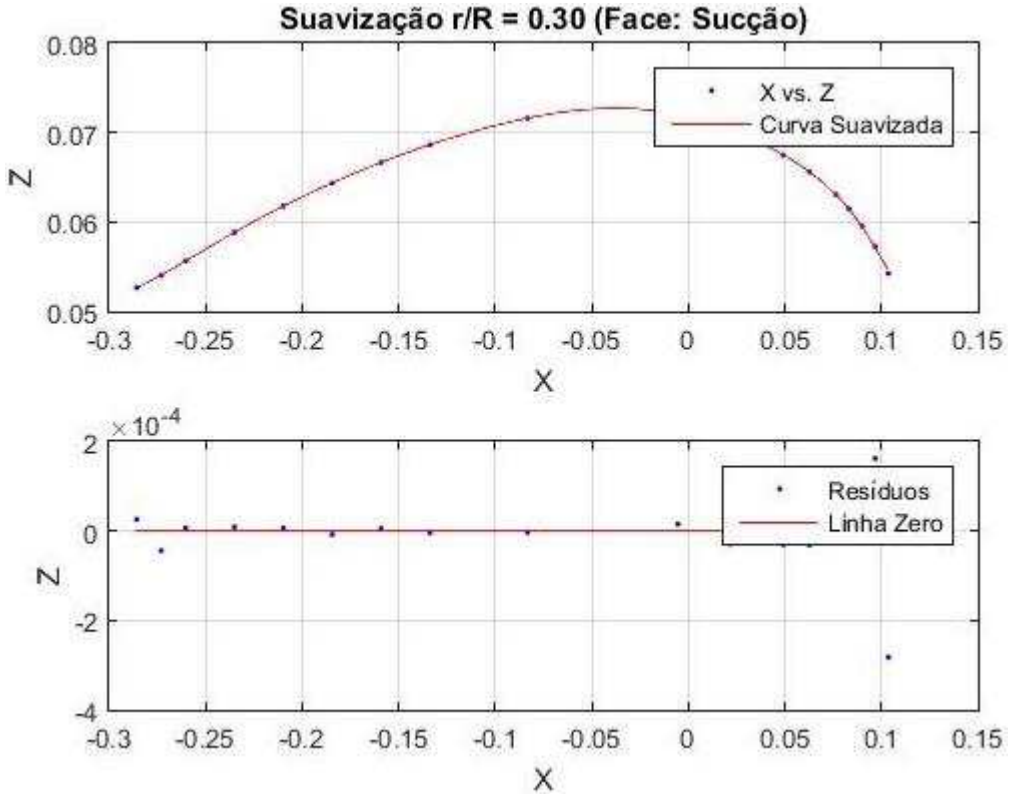
as cruvas possíveis desvios de curvatura na modelagem CAD são mitigados. Para ilustrar esse processo as curvas ajustadas para as seções de r/R = 0.3 e 0.6 são plotadas nas Figuras 36,37 e 38,39 respectivamente.

Figura 36: Face Pressão r/R = 0.3



Fonte: Autor (2016)

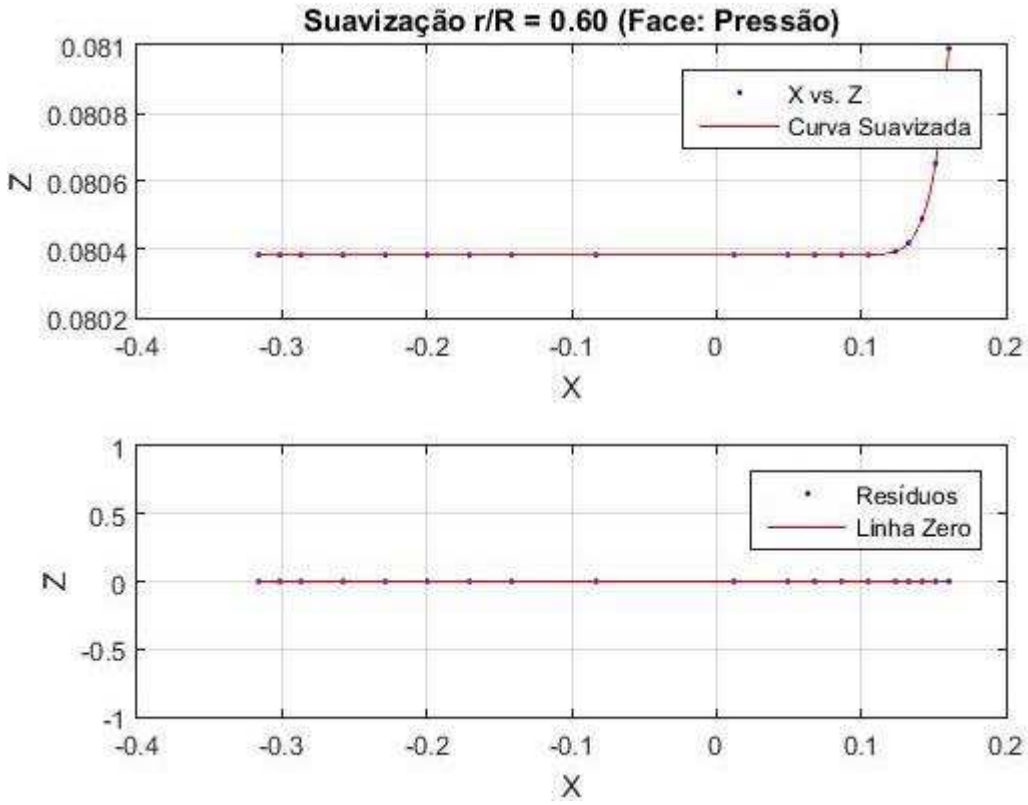
Figura 37: Face Sucção r/R = 0.3



Fonte: Autor (2016)

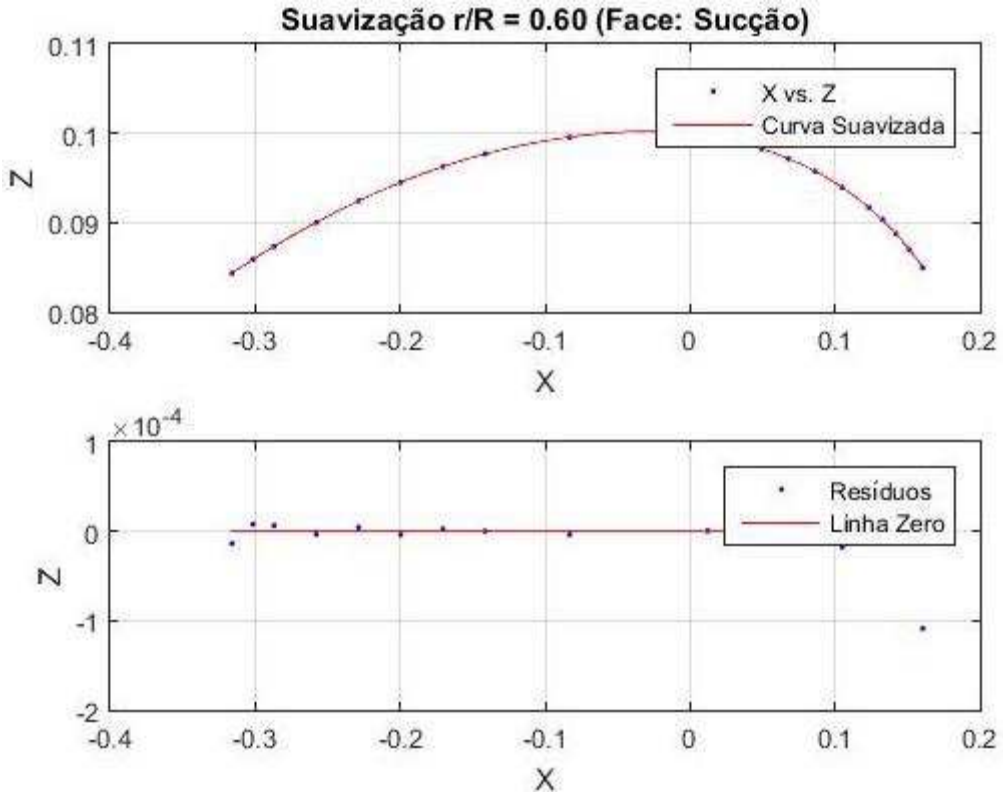
Com exceção da seção r/R = 0.6, todas as interpolações foram feitas com splines suaves do MATLAB enquanto que no caso da r/R = 0.6, foi-se necessário o uso de uma spline interpolada para a face de pressão. Isso ocorre nessa seção devido aos coeficientes da Tabela 4.8 de Kuipper (1992) serem zero até a coluna de 80%, portanto se usada uma spline ajustada como curva de tendência, resultaria em uma curva com um desvio descrecente antes de crescer como esperado em base dos pontos originais.

Figura 38: Face Pressão r/R = 0.6.



Fonte: Autor (2016)

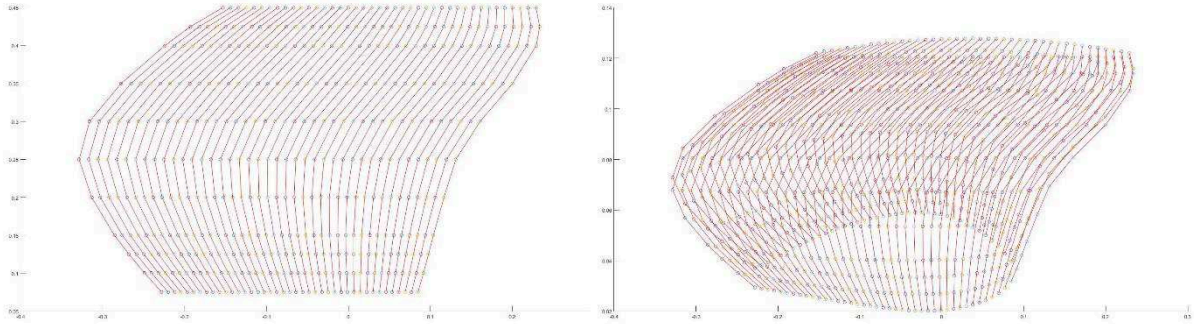
Figura 39: Face Sucção r/R = 0.6.



Fonte: Autor (2016)

Nessa etapa existem 11 seções, cada seção composta por duas curvas, cada curva descrita por 41 pontos. Na próxima etapa serão criadas listas de valores unindo os pontos no sentindo do cubo para a extremidade da pá, não mais pontos da mesma seção. Portanto serão criadas 41 listas, com 11 elementos cada. O intuito da criação dessas listas é poder gerar seções intermediárias as que a publicação original da série fornece, o resultado dessas listas está plotado na Figura 40, os eixos são X-Y e X-Z, da esquerda para a direita do leitor.

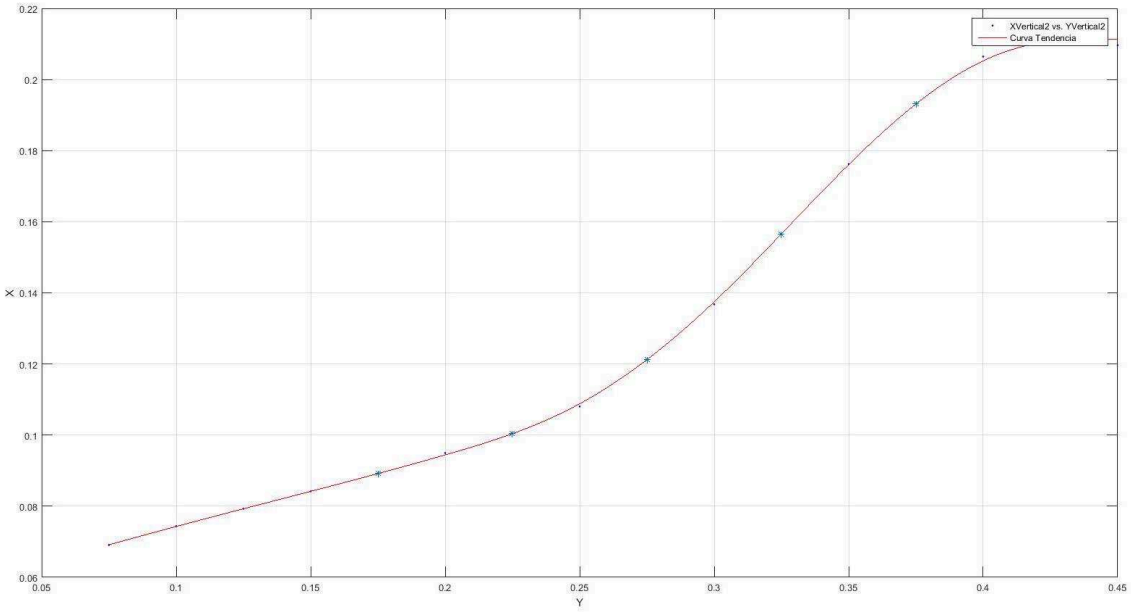
Figura 40: Pontos conectados entre seções



Fonte: Autor (2016)

A próxima etapa consiste efetivamente na criação das novas seções intermediárias, para isso foram-se criadas curvas ajustadas de tendência para os valores de X em função de Y, conforme ilustrado na Figura 41. Tudo isso dentro de um *loop* programado de 1 a 41, uma curva de tendência para uma das listas.

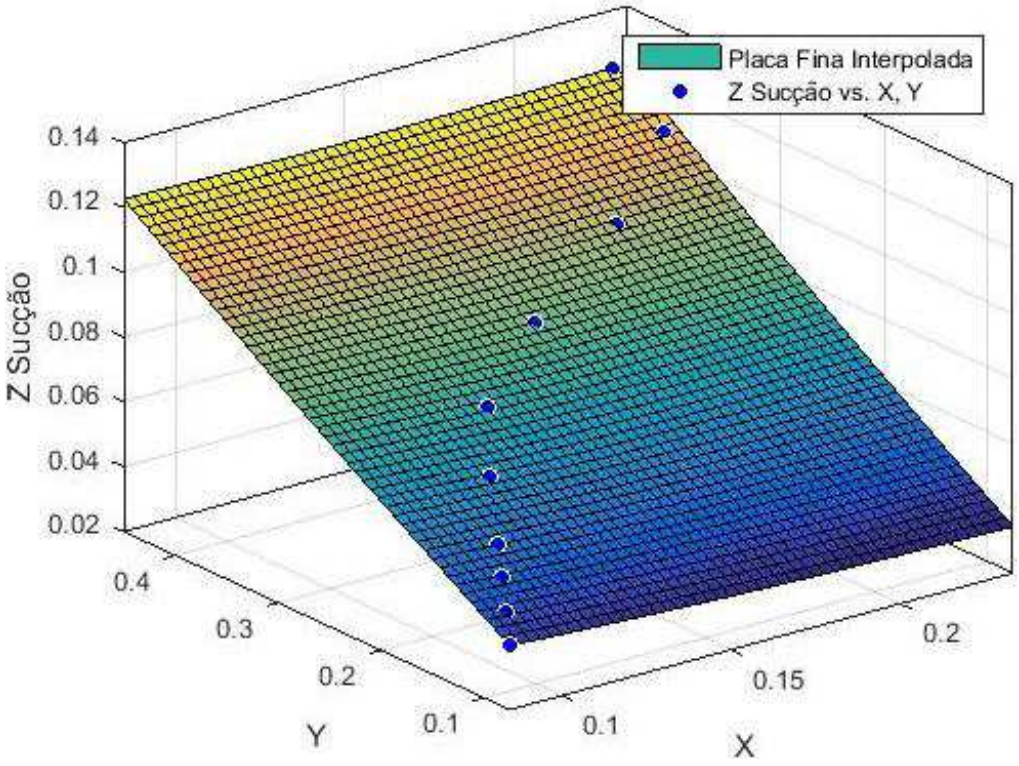
Figura 41: Spline de tendência Y(X) para uma das curvas da Fig.40



Fonte: Autor (2016)

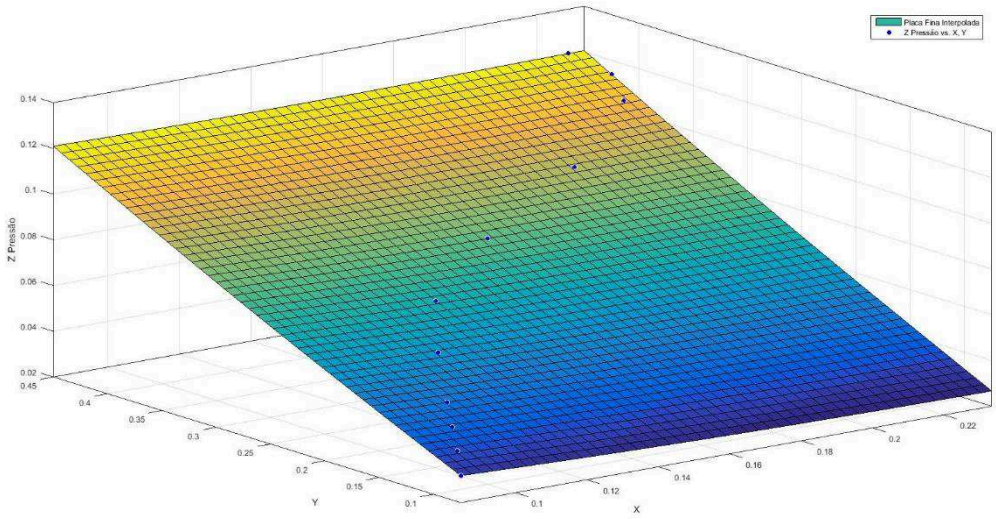
As seções a serem criadas são para r/R = 0.35, 0.45, 0.55, 0.65 e 0.75. Usando as curva geradas, como a da Figura 41, foram-se calculados os pontos de correspondentes em X. De posse das coordenadas X,Y foi-se constatado que os valores de Z para cada par X,Y pode ser calculado com o uso de uma superfície interpoladora. Os valores se dispunham sobre uma superfície plana inclinada, conforme ilustrado na Figura 42 para o lado de sucção e na Figura 43 para o lado de pressão.

Figura 42: Plano interpolador – Coordenada Z Sucção



Fonte: Autor (2016)

Figura 43: Plano interpolador – Coordenada Z Pressão

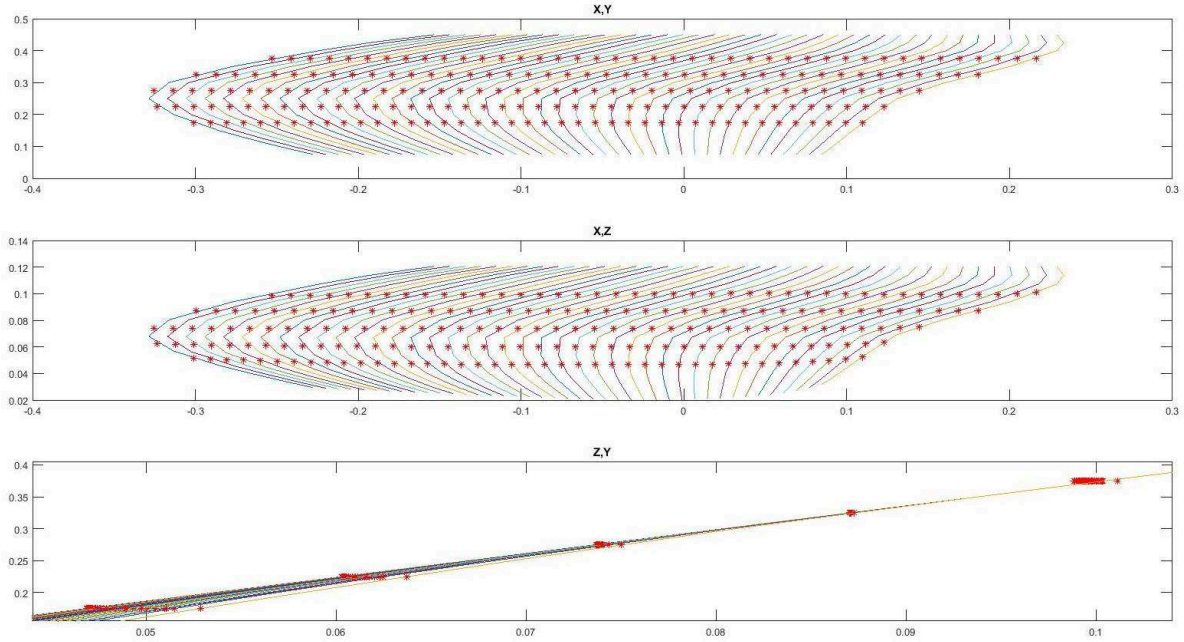


Fonte: Autor (2016)

Como a interpolação foi efetuada incialmente em dois planos e posteriormente levada ao domínio de três dimensões, a Figura 44 foi gerada com o intuito de validar a criação dessas novas seções. Observa-se que em todas as projeções os pontos gerados apareceram nos locais corretos, sendo assim estão corretamente posicionados no espaço.

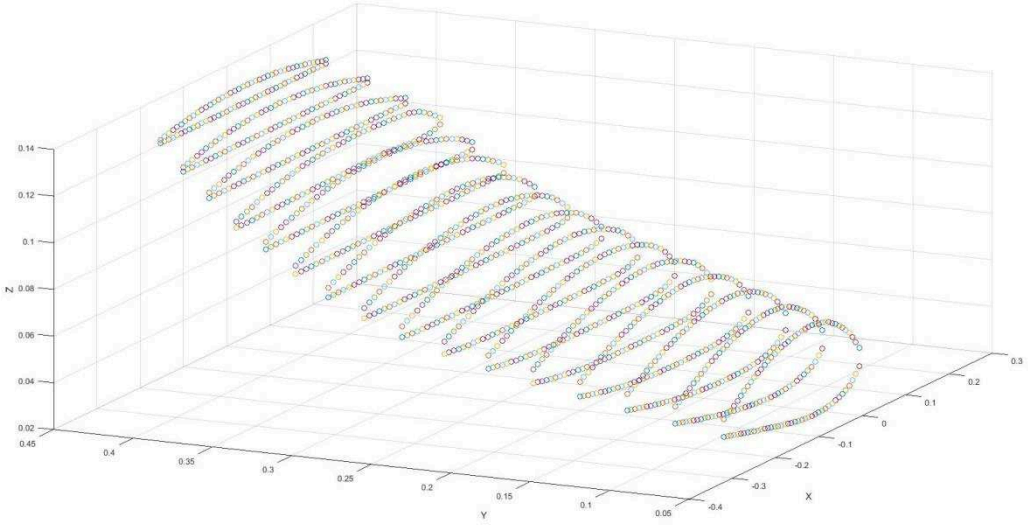
Os pontos não estão exatamente sobre as retas quando olhados os eixos X-Y e X-Z exatamente pelo fato de serem retas, a conexão entre os pontos originais está feita em segmentos de retas na Figura 44, enquanto que a geração deles foi feita por splines, o que gera um pequeno desvio para as seções intermediárias. As novas seções geradas nessa etapa são mostradas na Figura 45 assim como as seções originais.

Figura 44: Projeções 2D das seções originais e extras.



Fonte: Autor (2016)

Figura 45: Seções extras e originais.



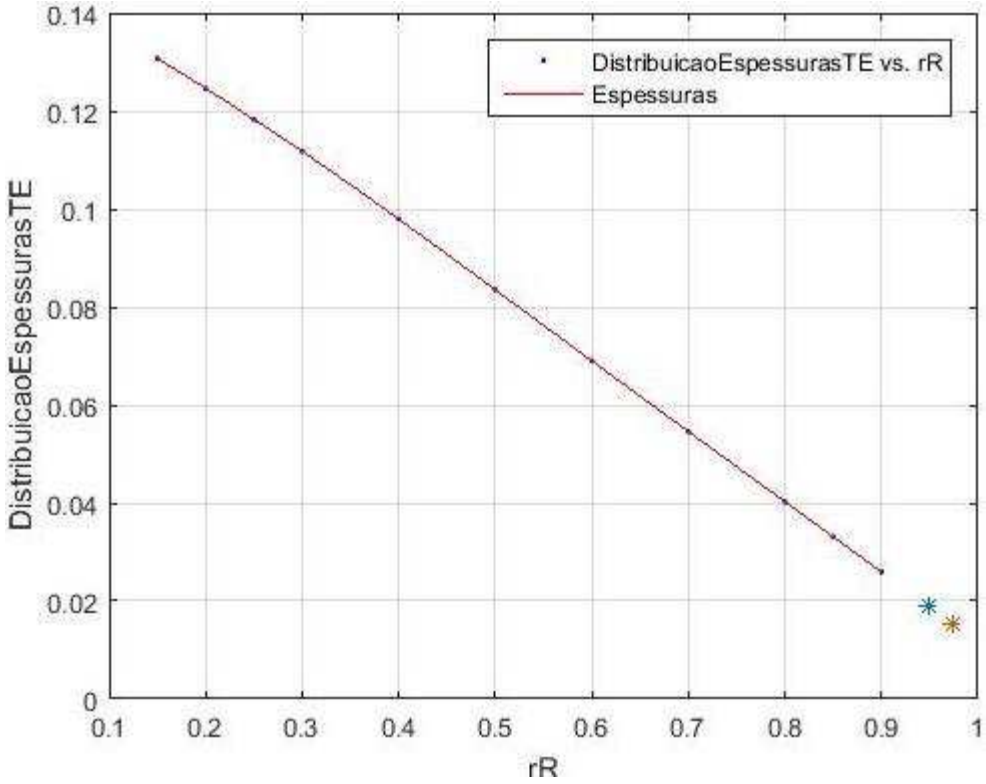
Fonte: Autor (2016)

A próxima etapa, consiste na geração dos pontos do contorno da pá acima da seção de r/R = 0.9. O objetivo dessa etapa é a criação de dois pontos adicionais para cada face, em cada bordo, na altura de r/R = 0.95 e 0.975, além de um ponto adicional em cada face para extremidade r/R = 1.0. O ponto extremo de r/R = 1.0 na face de pressão já possui todas as coordenadas conhecidas nessa etapa, a

interpolação da *skew* para r/R = 1.0 nos da sua posição em X, sua posição em Y é igual ao raio do propulsor e sua posição em Z é o caimento para r/R = 1.0, para o ponto na face de sucção apenas adiciona-se a espessura para r/R = 1 na coordenada Z.

As espessuras máximas das as seções de r/R = 0.95 e 0.975 podem ser obtidas através da interpolação da distribuição de espessuras, Figura 33, decrescendo linearmente até r/R = 1.0 por um polinômio grau um. Para o bordo de ataque a espessura vale 20% da espessura máxima da seção, porém não existe nenhuma citação a respeito das espessuras do bordo de fuga. Portanto as espessuras dos extremos do bordo de fuga que foram calculados anteriormente pela Equação 2 foram plotados e extrapolados para r/R = 0.95 e 0.975, conforme Figura 46.

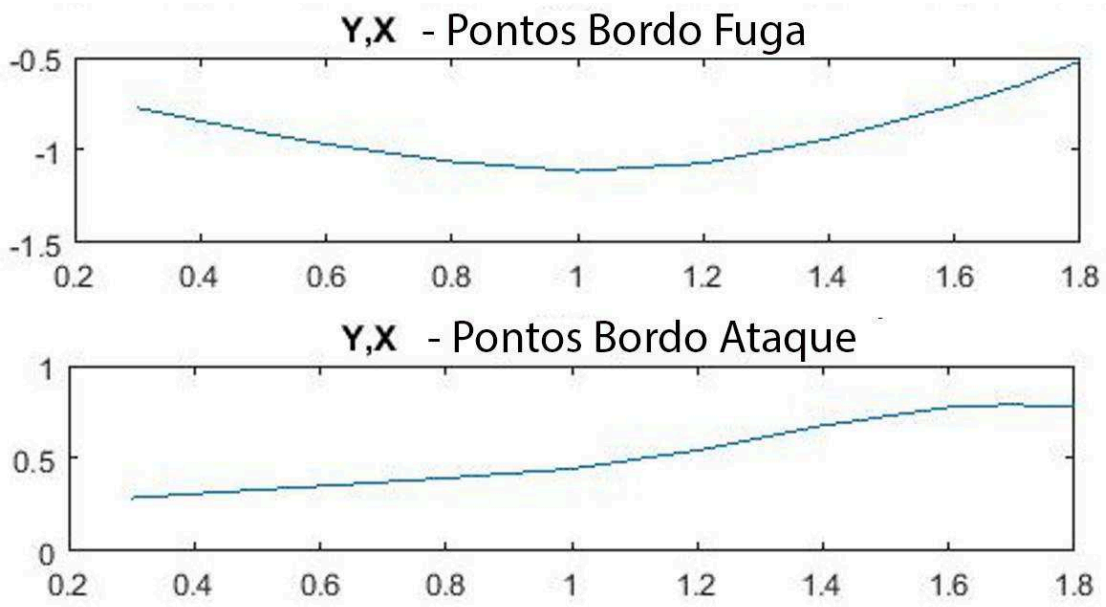
Figura 46: Distribuição espessuras - Bordo de fuga



Fonte: Autor (2016)

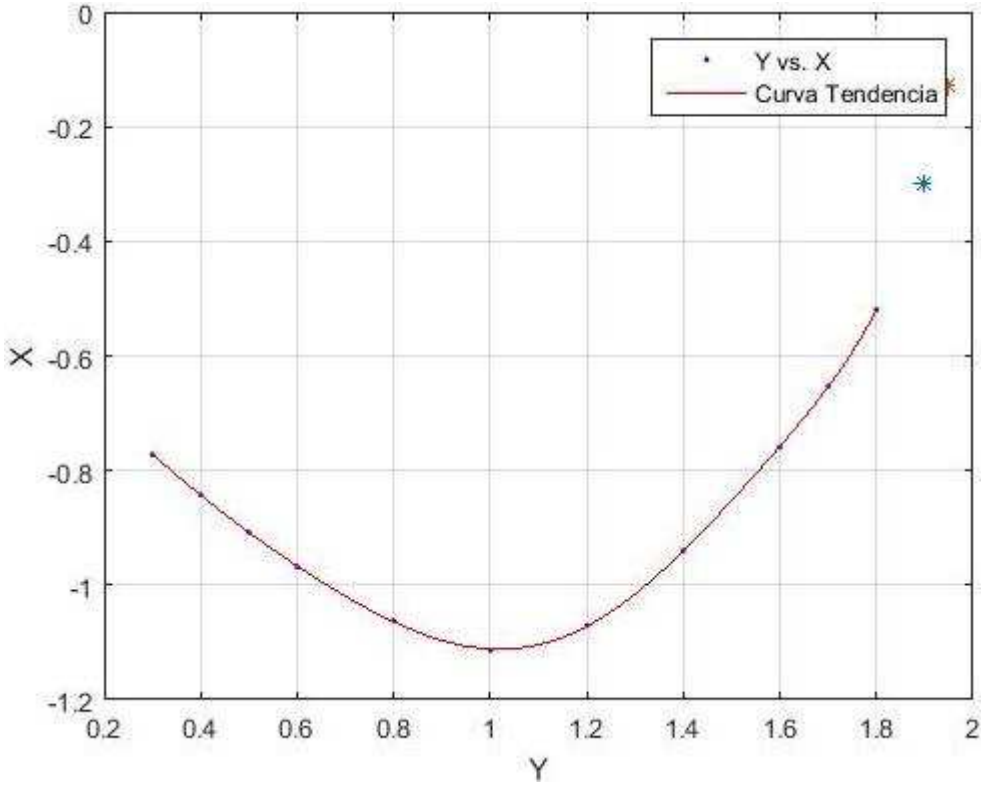
Para se criar os pontos desejados dos bordos de fuga das seções acima de r/R = 0.9, os pontos extremos dos bordos de fuga e ataque de cada seção foram plotados em projeções 2D. Na Figura 47 pode-se observar o comportamento dos pontos extremos dos bordos nos eixos Y-X, assim foram-se ajustadas curvas splines sobre os pontos originais como apresentados nas Figuras 48 e 49.

Figura 47: Pontos extremos



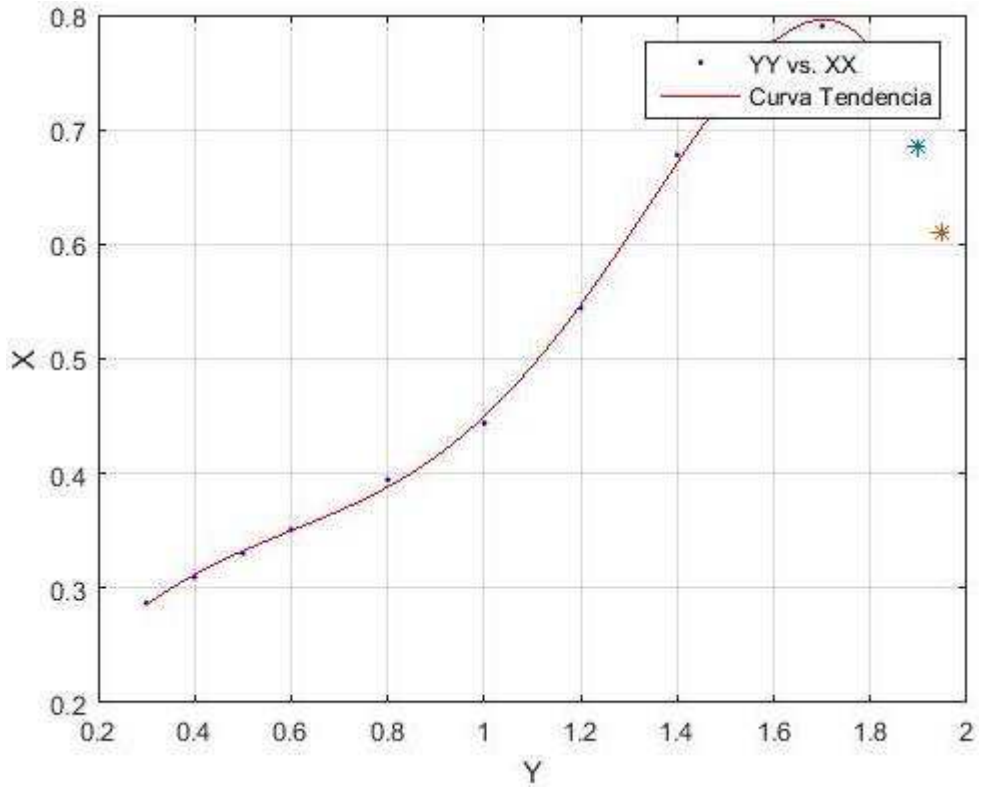
Fonte: Autor (2016)

Figura 48: Curva tendência X(Y) – Bordo fuga



Fonte: Autor (2016)

Figura 49: Curva tendência X(Y) – Bordo ataque

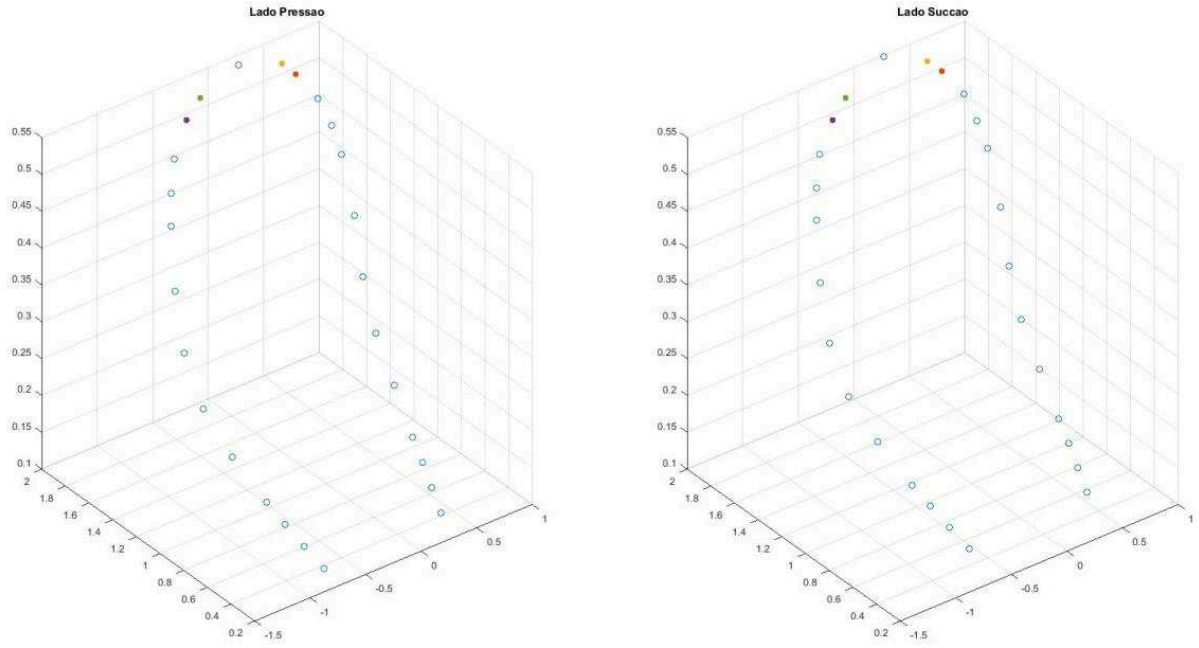


Fonte: Autor (2016)

Usando das curvas ajustadas apresentadas nas Figuras 48 e 49 foram extrapoladas as posições de X em função de Y, para os pontos em r/R = 0.95 e 0.975.

Portanto, seguindo a mesma lógica usada para posicionar o ponto extremo de r/R = 1.0, temos que as coordenadas em X foram-se descobertas nas Figuras 48 e 49, a coordenada Y é a seção desejada, e as coordenadas Z de pressão e sucção são o caimento para a determinada seção com a adição da espessura para o ponto de sucção, descobertas nas Figuras 46 para o bordo de fuga e sendo 20% da espessura máxima da seção para o bordo de ataque. O posicionamento desses pontos no espaço está apresentado na Figura 50.

Figura 50: Pontos extras posicionados



Fonte: Autor (2016)

Como última etapa de calculo são calculados os ângulos de passo, que são os ângulos em que cada seção será rotacionada durante a modelagem no Rhinoceros. Os propulsores do tipo B-Series possuem um passo constate para todos os raios, por isso os ângulos de rotação de cada seção mudam. A única exceção são os propulsores de 4 pás que possuem uma redução do passo de r/R = 0.5 até o cubo, sendo que o passo no cubo é de cerca de 80% do passo de r/R = 1.0.

O ângulo de rotação é calculado pela Equação 15 e no caso de propulsores de 4 pás, a equação permanece a mesma, a única alteração é no valor do passo, para isso o código reconhece caso seja inserido como dado de entrada um propulsor de 4 hélices, multiplicando as seções de r/R menores que 0.5 por um coeficiente que vai de 0.822 até 1.

$$\theta = tan^{-1}\left(\frac{Passo}{2\ x\ PI\ x\frac{r}{R}x\frac{D}{2}}\right) \quad (15)$$

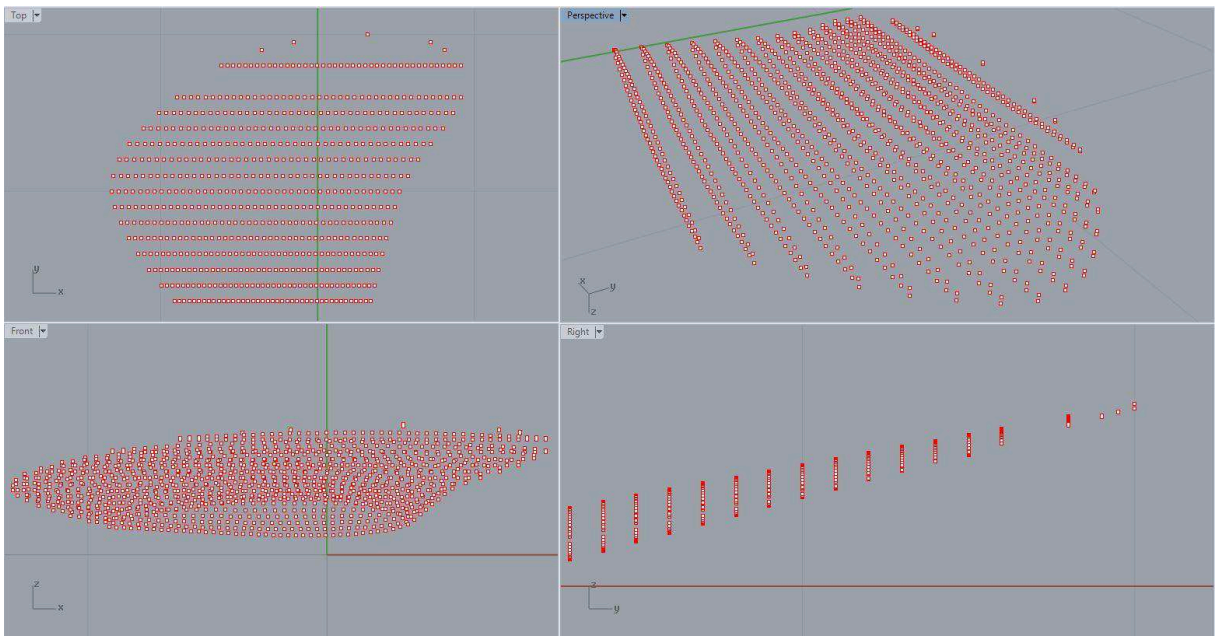
Por fim os dados desenvolvidos são juntados e gravados conforme o nome inserido na interface gráfica, por padrão PropulsorBseries.csv na mesma pasta em que foi executado o código do MATLAB.

4.2. MODELAGEM AUTOMATIZADA RHINOCEROS

A rotina de programação em IronPython responsável pela modelagem automatizada está em sua íntegra no Apêndice B. Para iniciar o processo, a primeira etapa no Rhinoceros é a execução do comando *RunPythonScrip,* que abrirá a janela para selecionar o arquivo Modelagem.py para ser executado. Ao executá-lo uma nova janela “Selecione o arquivo .CSV” será aberta, basta escolher o arquivo gerado pelo MATLAB.

A rotina será responsável por ler o arquivo *.csv* e extrair dele todas as informações geradas pelo MATLAB, alocando-as conforme programado e retornando as seguintes informações no *Output*: diâmetro propulsor, número de pás, quantidade de seções (r/R), quantidade de pontos descrevendo cada seção. A primeira etapa do processo consiste em adicionar os pontos no ambiente de trabalho do Rhinocers, Figura 51. O sistema de coordenadas utilizado permanece o mesmo.

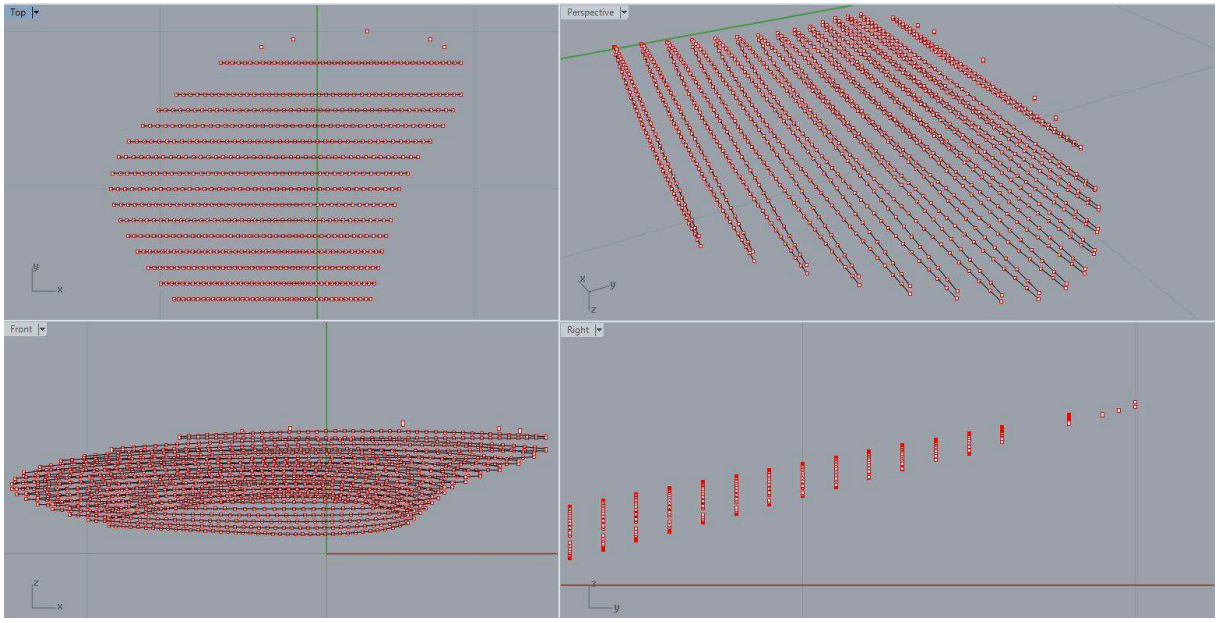
Figura 51: Importando-se os pontos no Rhinoceros



Fonte: Autor (2016)

Após a importação dos pontos, são geradas duas curvas de grau três para cada seção, uma do lado de pressão e uma do lado de sucção do hélice. As curvas podem ser vistas na Figura 52.

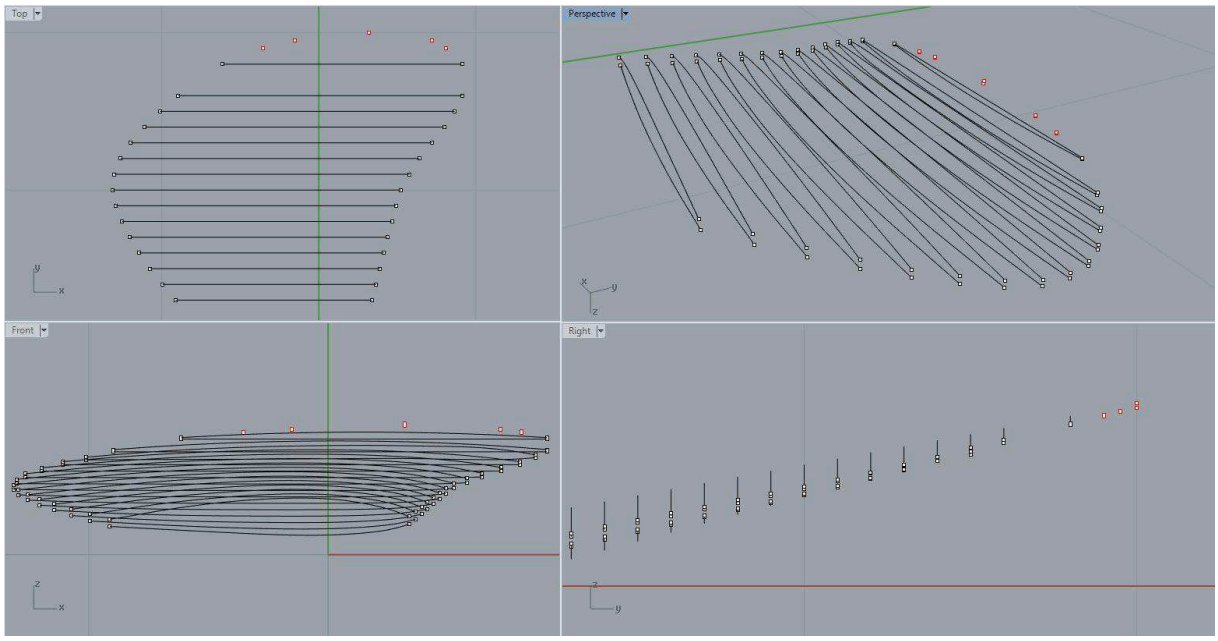
Figura 52: Curvas das seções



Fonte: Autor (2016)

Posteriormente todos pontos importados são deletados, e geram-se os pontos de início e final das curvas, sendo esses os bordos de ataque e fuga para cada lado do propulsor, Figura 53.

Figura 53: Pontos dos bordos de fuga e ataque

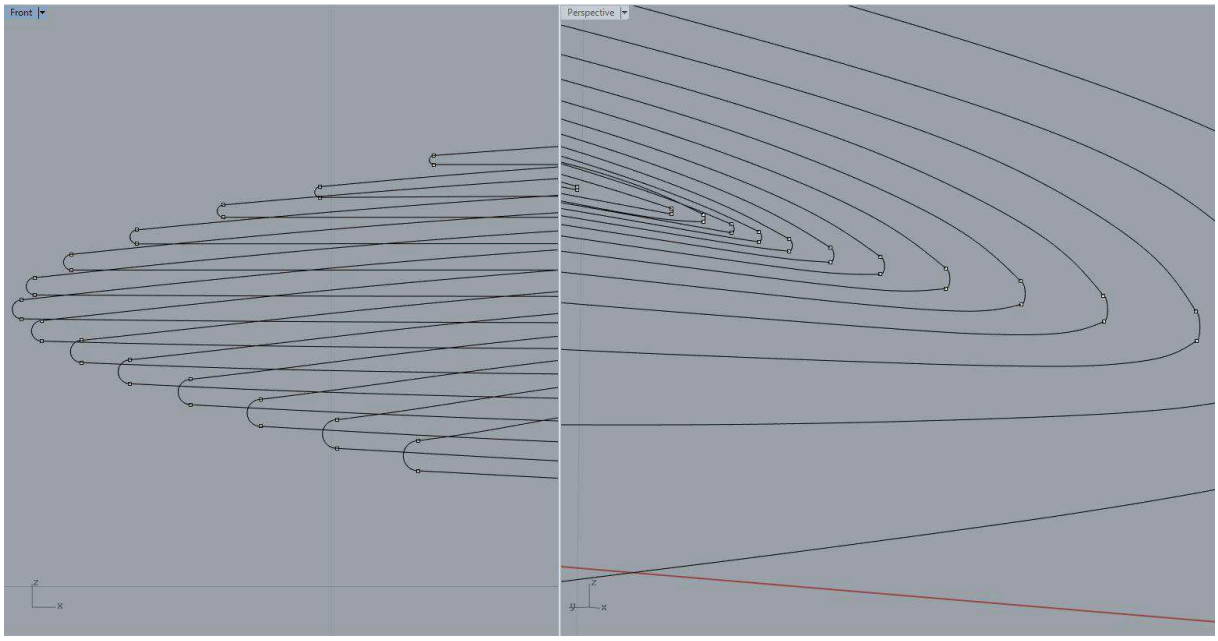


Fonte: Autor (2016)

Usando-se dos identificadores dos pontos finais de cada curva os bordos são fechados com arcos de início e fim nos lados de pressão e sucção, e as

direções positivas em Z para o bordo de ataque e negativas em Z para o bordo de fuga, Figura 54.

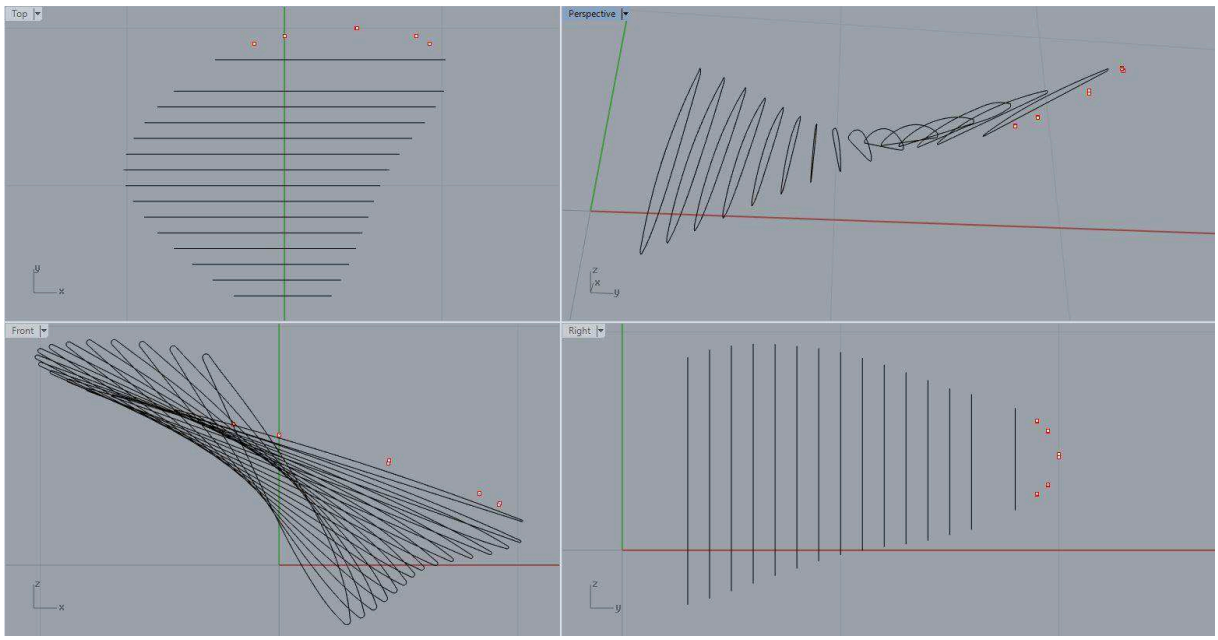
Figura 54: Fechamento dos bordos



Fonte: Autor (2016)

Nessa etapa as seções já possuem seu contorno completo e são aplicadas rotações nas mesmas em torno do eixo Y, devido ao fato que o passo das B-Series é constante, o ângulo de passo varia. Os pontos e arcos referentes ao contorno de borda das seções acima de r/R = 0.90 também são rotacionados, Figura 55.

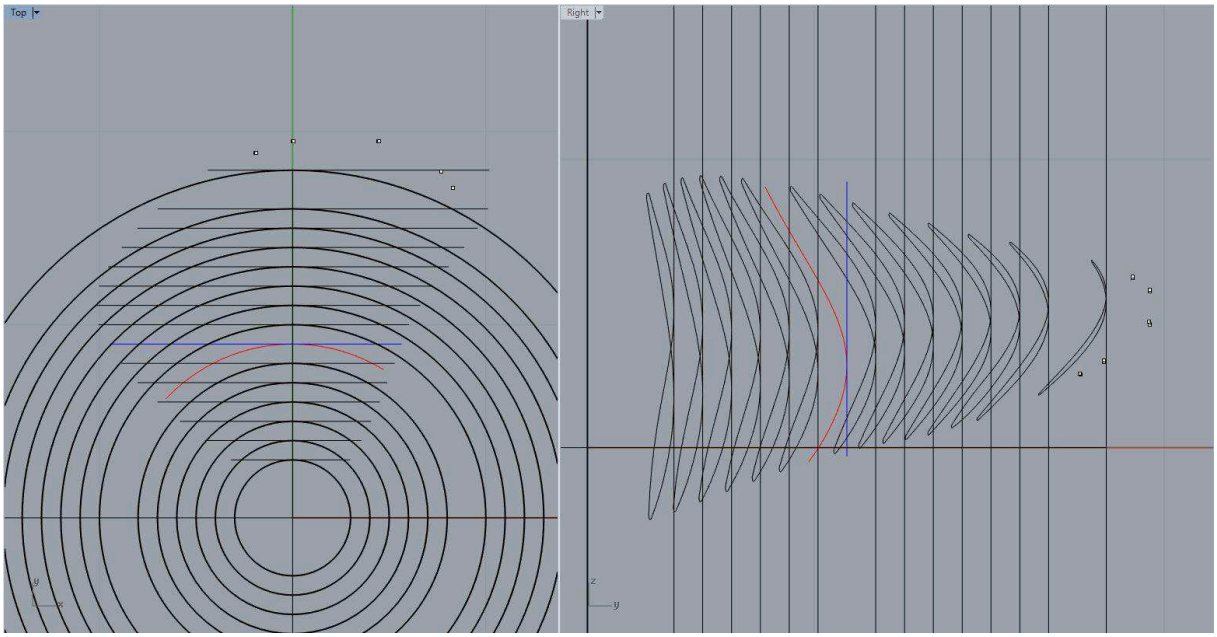
Figura 55: Rotações de passo



Fonte: Autor (2016)

Agora que as seções já estão em suas posições corretas elas devem ser conformadas cilindricamente, já que as seções geradas pelo MATLAB são planificadas, ao conformá-las, todos seus pontos estarão a uma distância de r/R iguais do cubo. Na Figura 56 podemos observar no r/R = 0.5, a reta em azul, que ao ser conformada gerou a curva em vermelho, o cilindro correspondente a r/R = 0.5 e a curva do lado de pressão foram deletados para clarificar a visualização.

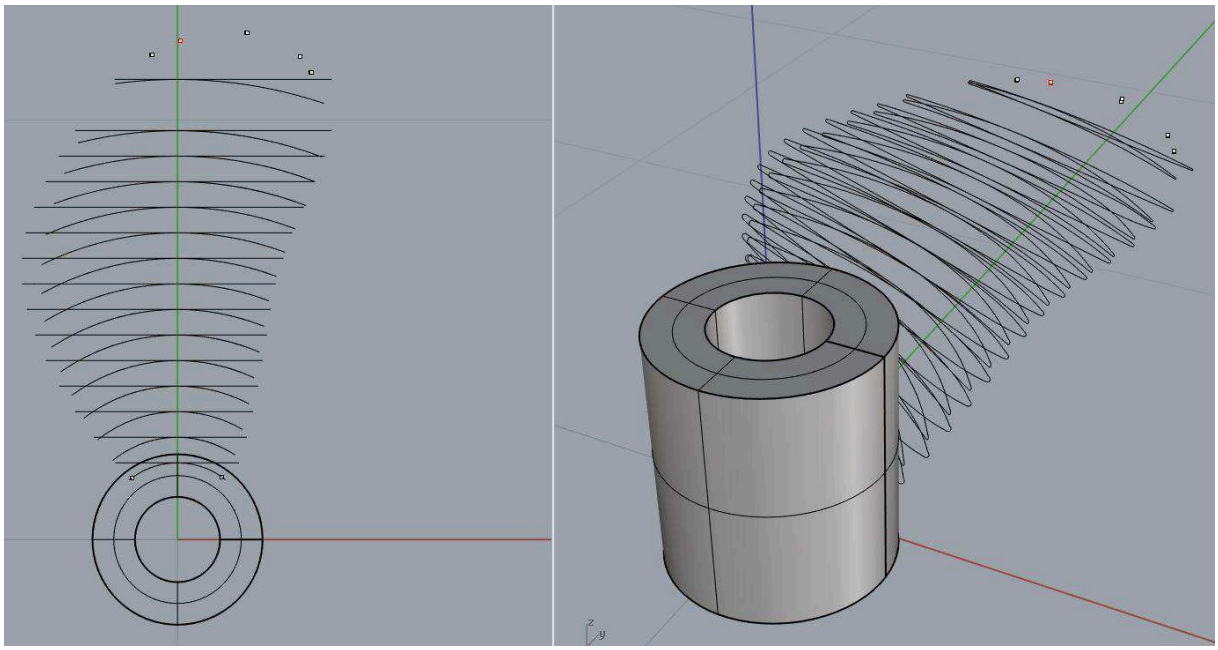
Figura 56: Conformação cilíndrica



Fonte: Autor (2016)

Utilizando-se agora dos arcos dos bordos de fuga e ataque para r/R = 0.15 será definido um comprimento de cubo de tamanho 30% maior que a distância entre os bordos. Conforme Kuipper (1992), o diâmetro do cubo de um propulsor série B é 1/6 (16,67%) de seu diâmetro, com exceção dos propulsores de 3 pás que possuem cubos com diâmetro de 9/50 (18%) de seu diâmetro. Portanto um cilindro desse diâmetro é criado no centro das pás, conforme Figura 57.

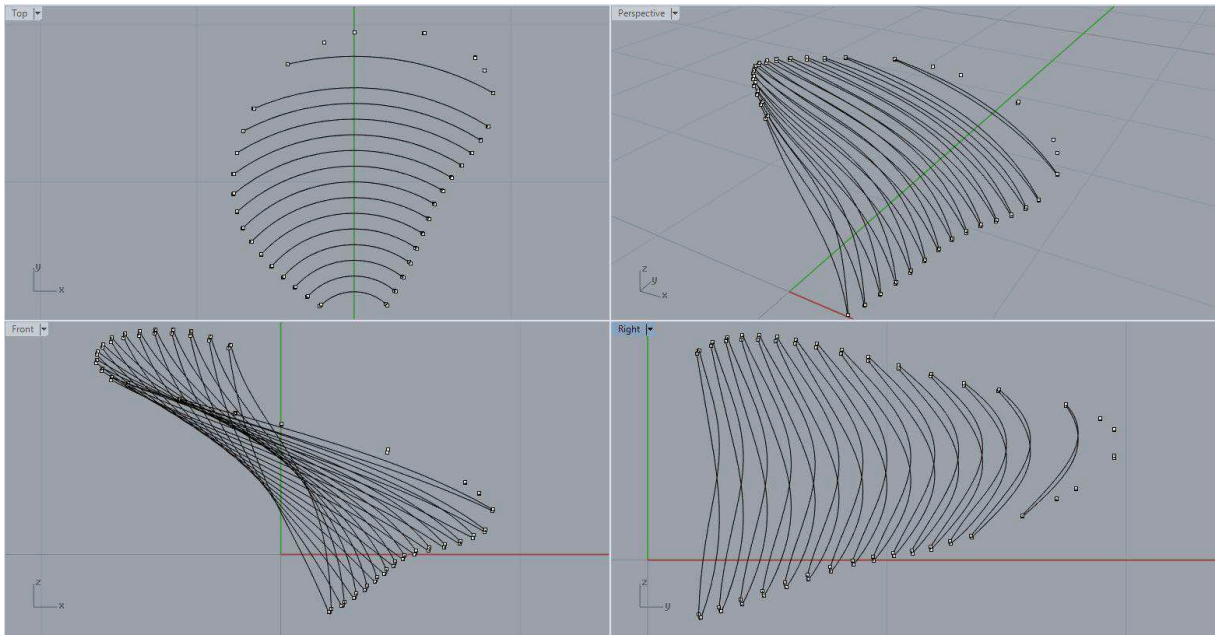
Figura 57: Curvas conformadas e pontos finai/iniciais



Fonte: Autor (2016)

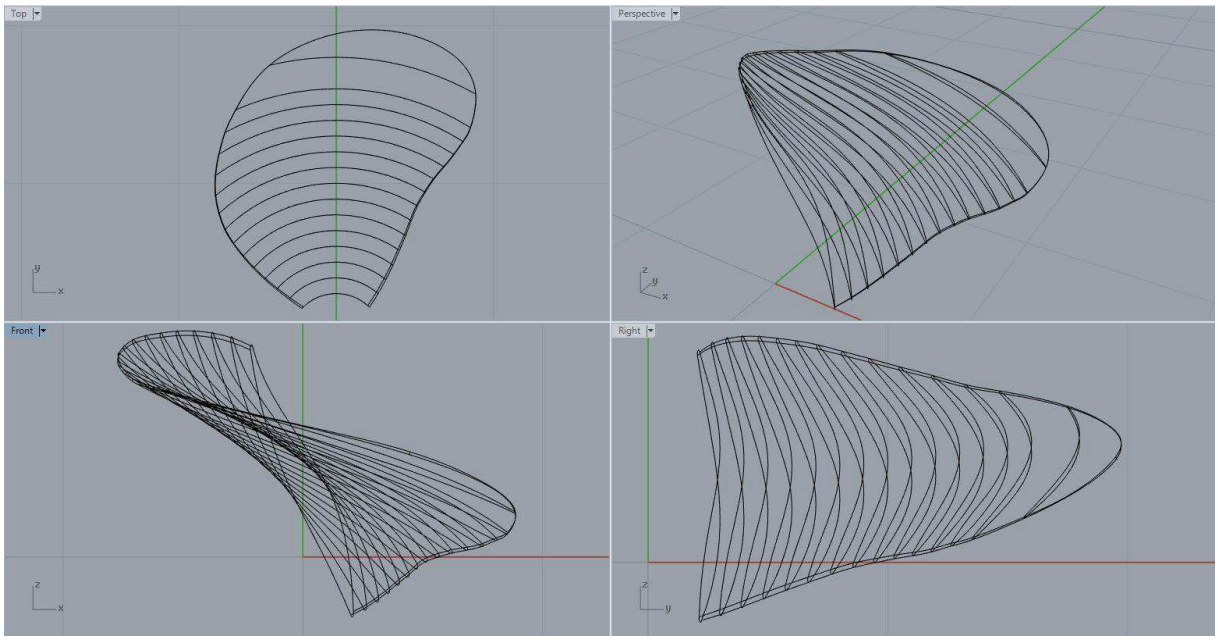
Deletando-se os cilindros e escondendo o cubo criado, adiciona-se novamente os pontos iniciais e finais das curvas, agora conformadas, obtendo a Figura 58. Os pontos iniciais e finais das curvas serão usados para gerar uma curva interpolada de grau três definindo o contorno dos bordos de ataque e fuga da pá, conforme ilustrado na Figura 58.

Figura 58: Curvas conformadas e pontos iniciais/finais



Fonte: Autor (2016)

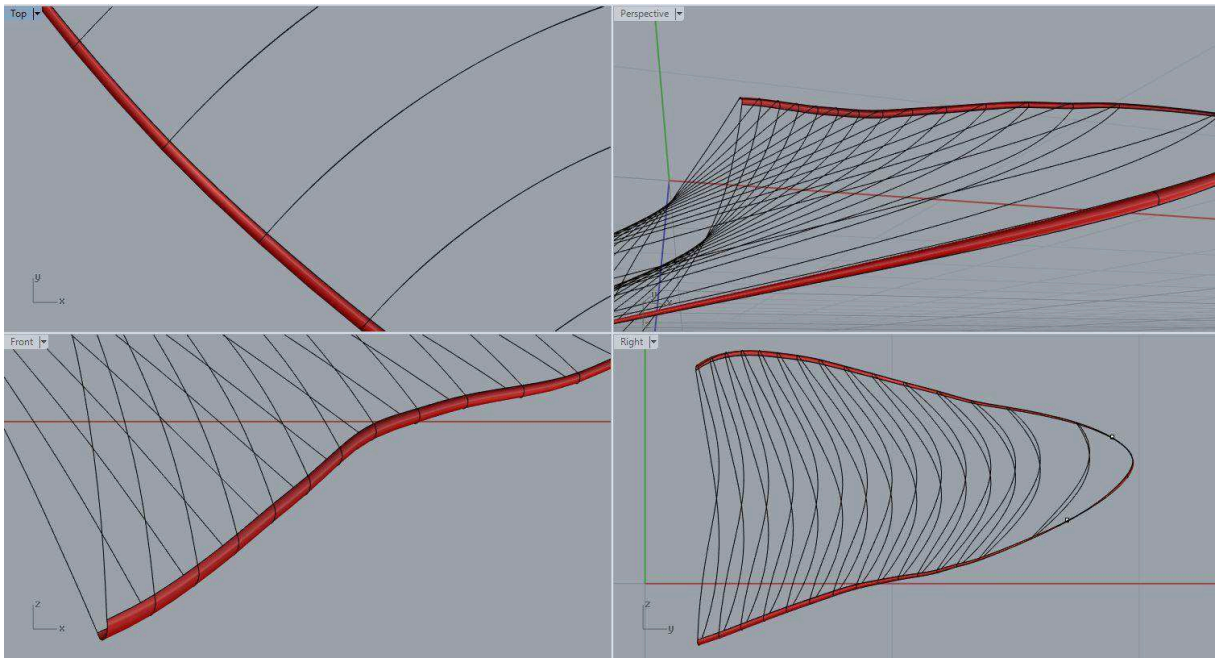
Figura 59: Contorno dos bordos



Fonte: Autor (2016)

Nessa etapa já temos os contornos de todas as seções e contorno da pá, serão geradas superfícies através da função *Curve Network.* Para uso desse comando são necessárias curvas em três direções. No caso da superfície dos bordos, usaremos as curvas de contorno de pressão e sucção, e o conjunto das curvas de fechamentos dos bordos, os arcos adicionados anteriormente e depois conformados juntamente com as curvas das seções. A superfície gerada foi sombreada em vermelho para visualização na Figura 60.

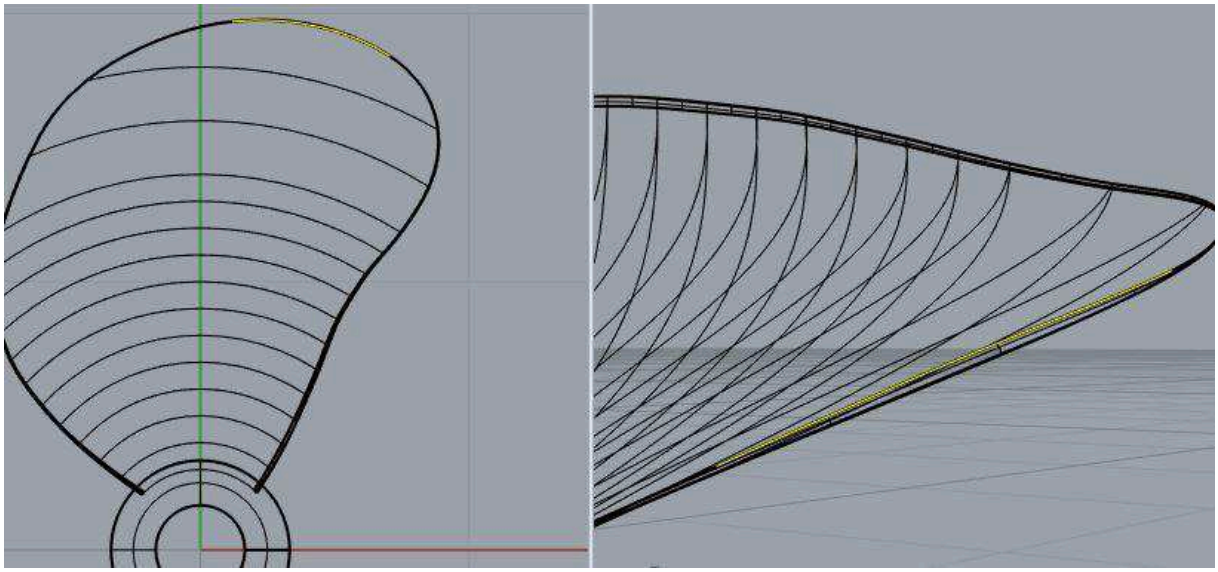
Figura 60: Importando-se os pontos no Rhinoceros



Fonte: Autor (2016)

Devido a exigência de curvas em três direções, as curvas do bordo de ataque e fuga serão agora subdivididas em três partes, pela função *Split*. A subdivisão ocorrerá acima da seção de r/R = 0.9. A primeira parte da subdivisão será do bordo de ataque, a segunda parte será entre o ponto final/inicial do bordo de ataque da seção r/R = 0.9 e do ponto final/inicial da mesma seção no bordo de fuga, e a terceira subdivisão será a curva no bordo de fuga, conforme Figura 61.

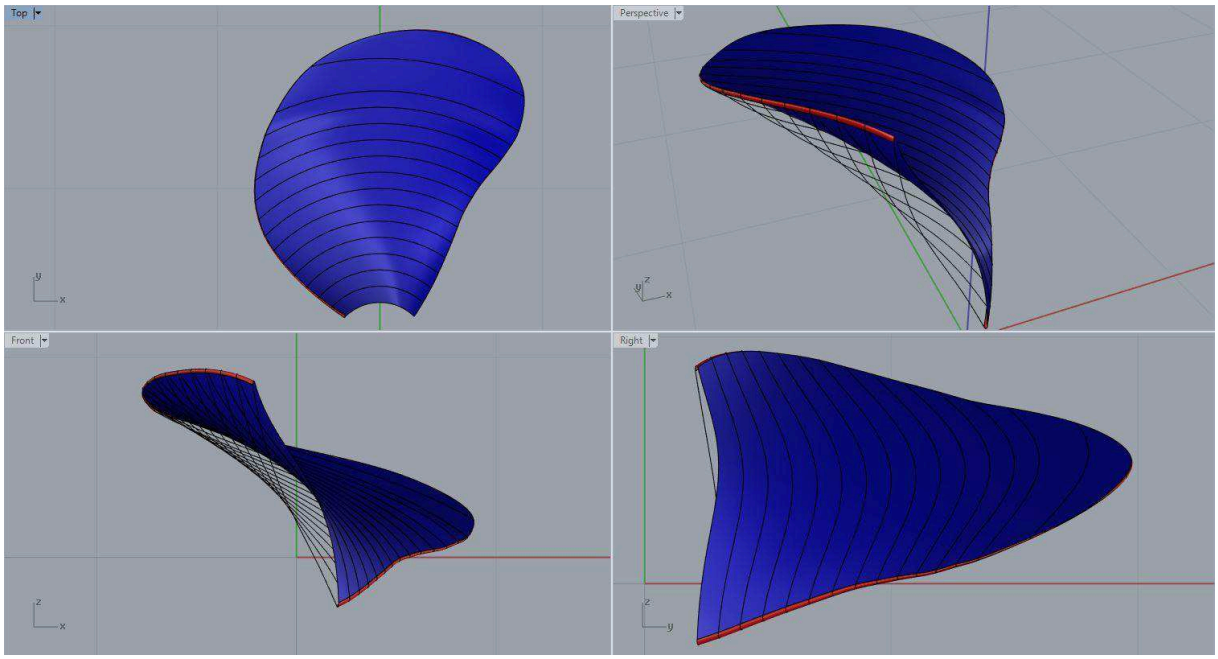
Figura 61: Split da curva de bordo



Fonte: Autor (2016)

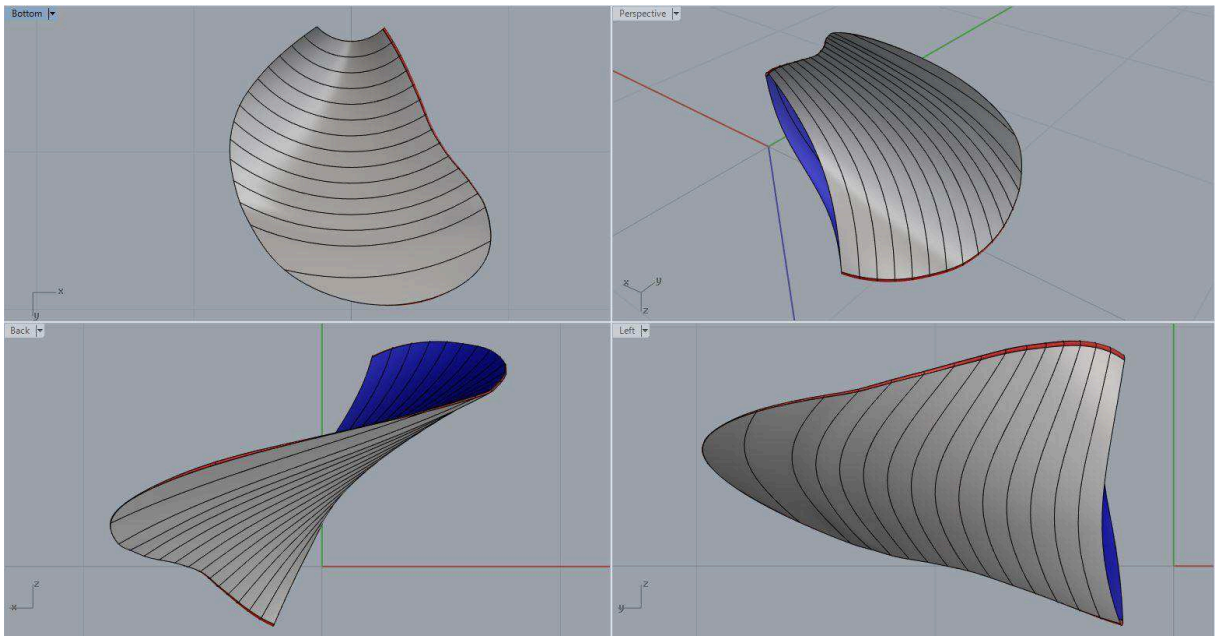
Para geração da superfície pelo *Curve Network*, a segunda parte da curva subdivida, entrará nas curvas de mesma direção das curvas das seções e as outras duas direções serão as curvas de ataque e fuga, gerando assim a superfície em azul na Figura 62 e em cinza na Figura 63.

Figura 62: Superfície de sucção



Fonte: Autor (2016)

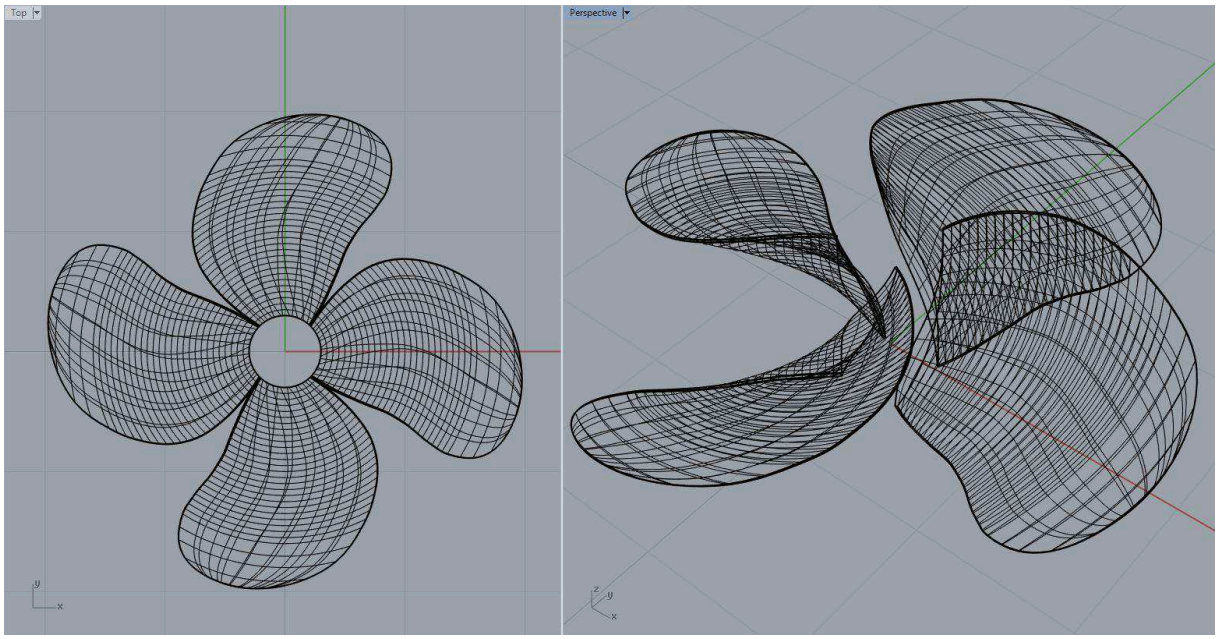
Figura 63: Superfície de pressão



Fonte: Autor (2016)

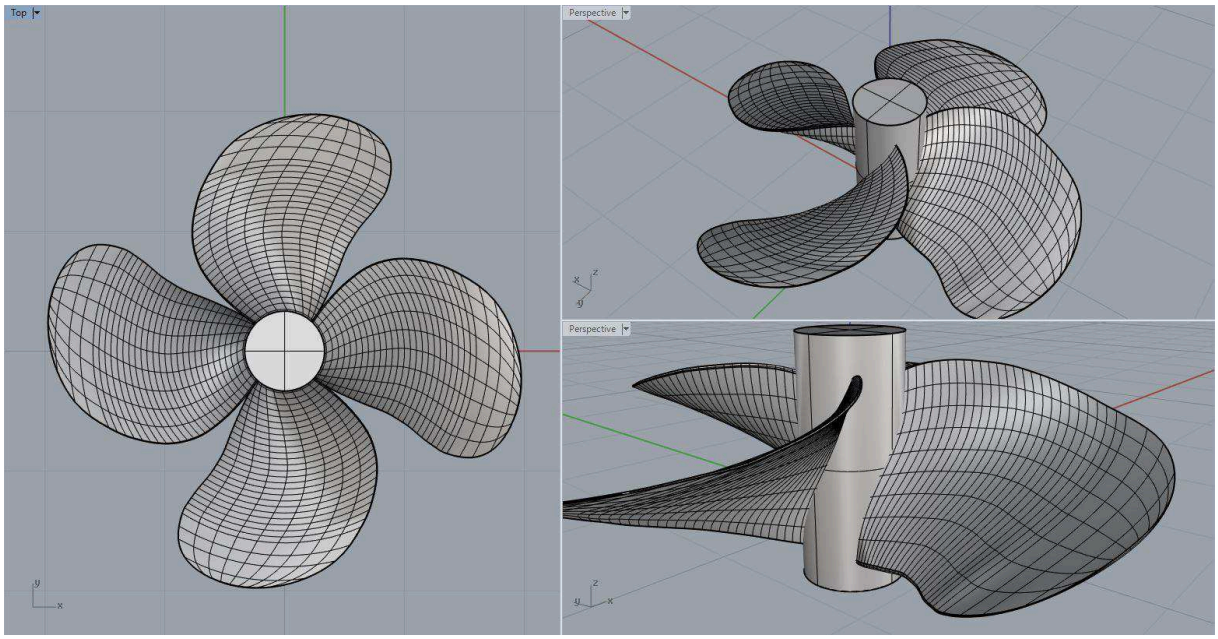
Com a pá modelada, a rotina usará a informação de quantas pás possui o propulsor para definir os ângulos de rotação com cópia que deverão ser efetuadas ao redor do eixo Z, para replicarmos as pás igualmente espaçadas, resultando na Figura 64.

Figura 64: Replicando as pás



Fonte: Autor (2016)

Figura 65: Resultado final modelagem



Fonte: Autor (2016)

Após a replicação das pás, conclui-se a modelagem automatizada, Figura 65. O código possui uma limitação quanto a geração das superfícies de filetes conectando as pás e os cubos, portanto essa etapa será de responsabilidade do usuário.

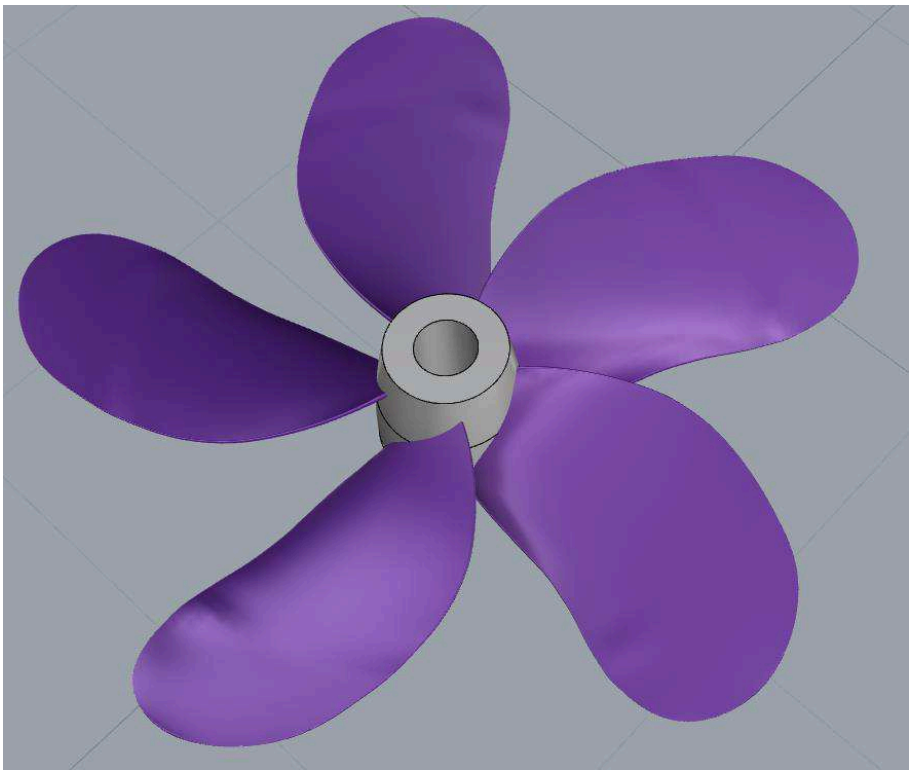
5. RESULTADOS E DISCUSSÕES

5.1 COMPARAÇÃO COM OUTROS AUTORES

Com o objetivo de validar a geração de geometria e a automatização da modelagem, o mesmo hélice usado pelos autores citados na referência bibliográfica, Seção 2.5, será modelado através dos códigos de presente trabalho e o valor de massa será comparado.

Usando os dados de diâmetro, relação passo/diâmetro, razão de áreas expandidas e número de pás da Tabela 3, foi gerada a geometria no MATLAB e esta selecionada ao executar a rotina do Rhinoceros para modelagem, resultando no hélice da Figura 66.

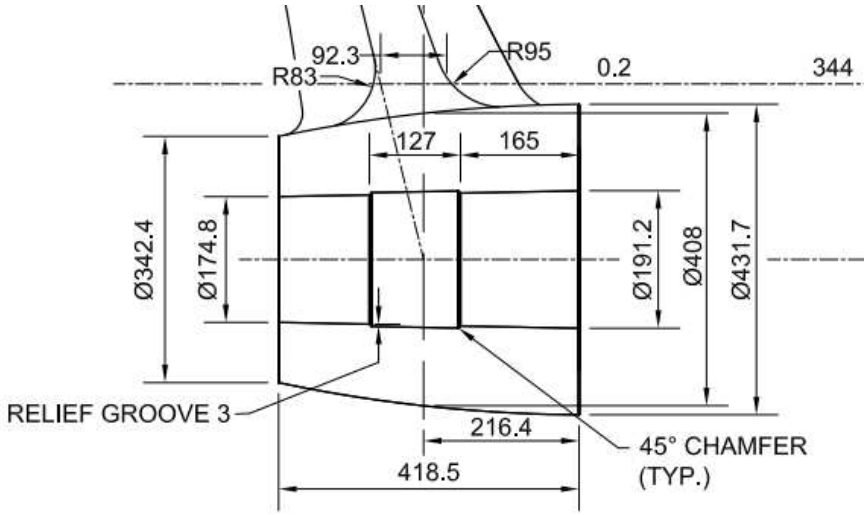
Figura 66: Hélice B5-60 modelado



Fonte: Autor (2016)

Para poder comparar com de forma mais exata o valor de massa, o cubo originário da automatização foi deletado e o cubo com as dimensões reais do cubo de referencia foi modelado manualmente, conforme Figura 67.

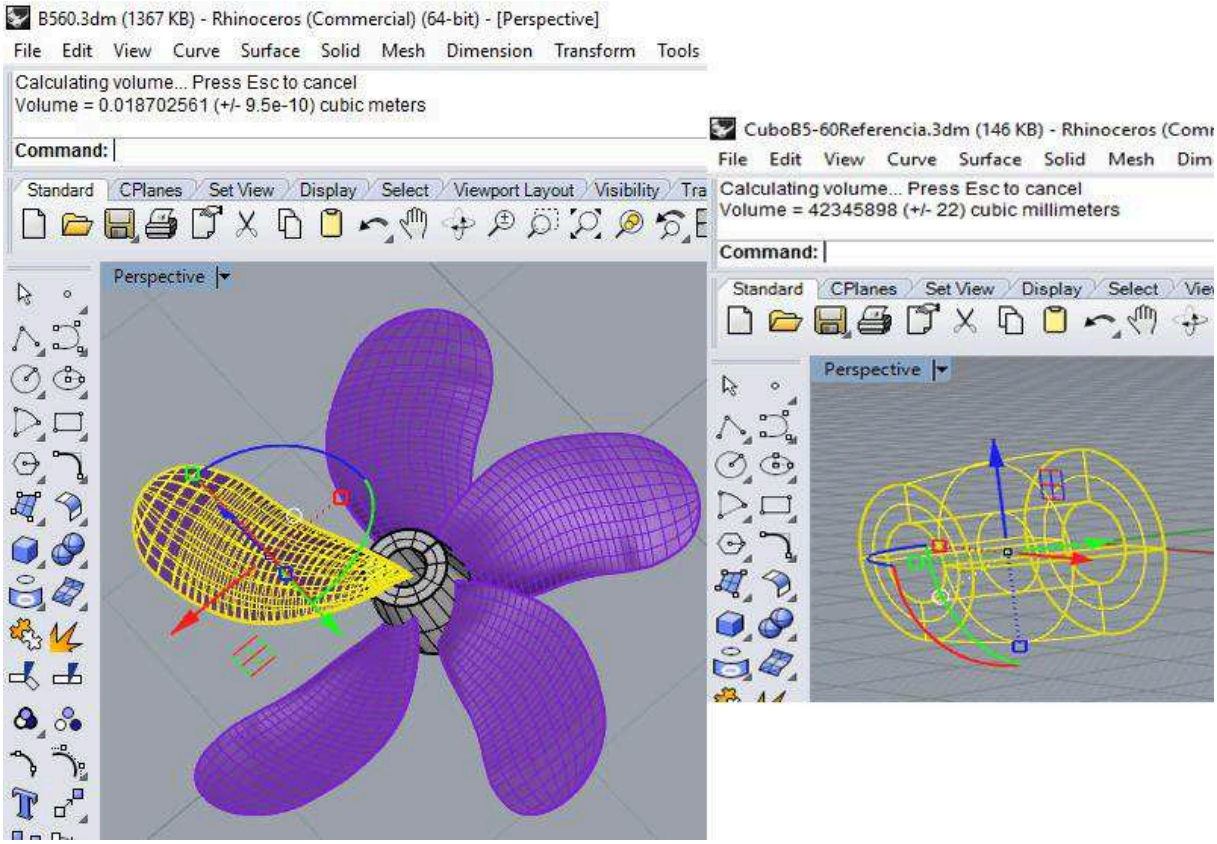
Figura 67: Cubo hélice referência



Fonte: Boddy (2014)

Usando as ferramentas de análise de massa do Rhinoceros constatou-se conforme Figura 68 que o volume de cada pá do hélice é de 0.018702561 m^3 e que o volume do cubo é 0.042345898 m^3, resultando em um volume total igual a 135.8514 cm^3. Segundo Boddy (2014) o material do hélice de referência possui uma densidade de 7.5 g / cm^3, portando o hélice modelado tem um peso igual a 1018.885 kg.

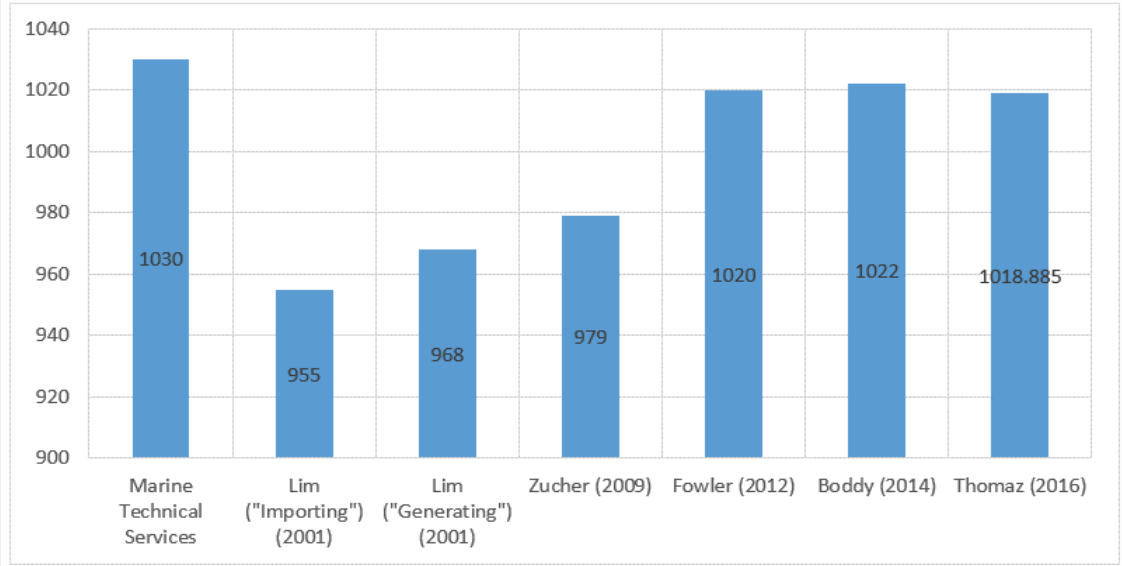
Figura 68: Volume hélice



Fonte: Autor (2016)

Através da Figura 69 pode-se observar uma comparação dos valores de massa e os seus erros relativos aos quais cada autor chegou com seu método de modelagem usando o hélice da Tabela 3 como referência. Todas as massas consideram os hélices e o cubo, filetes foram desconsiderados no caso de Boddy (2014) que foi o único a desenvolvê-los em seu método.

Figura 69: Comparação dos valores de massa dos autores



	MTS	Lim ("*Importing*") *(2001)*	Lim ("Generating") (2001)	Zucher (2009)	Fowler (2012)	Boddy (2014)	Thomaz (2016)
Erro Relativo:	**0%**	**-7.2816%**	**-6.0194%**	**-4.9515%**	**-0.9709%**	**-0.7767%**	**-0.7456%**

Fonte: Autor (2016)

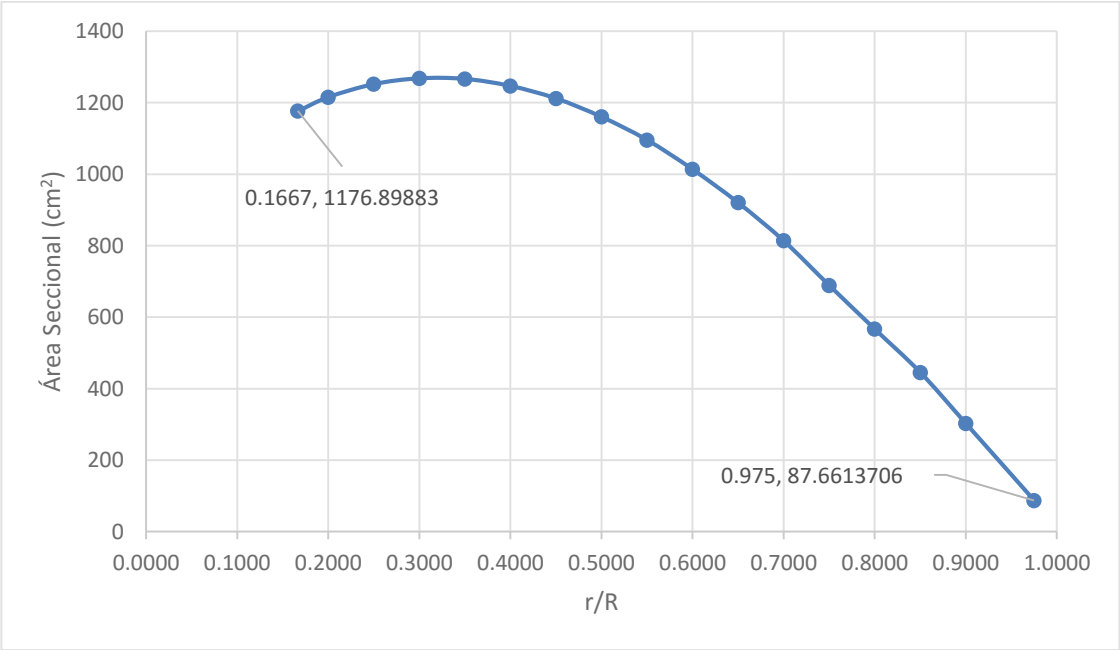
5.2 CURVA DE ÁREAS

A curva de áreas seccionadas de uma embarcação expressa a esbelteza da carena do casco (CACHO, 2014). Para curvas seccionadas abaixo da linha d'agua de uma embarcação é esperado uma curva crescente da popa até a meia-nau e decrescente da meia-nau até a proa.

No caso de uma pá do hélice da série sistemática estudada espera-se que a curva tenha um leve crescimento desde o cubo até por volta das seção r/R = 0.3 ~ 0.4 e a partir desse ponto as áreas sejam decrescente até sua extremidade final.

Em ambas as situações é esperado que a curva de áreas seccionais varie de forma suave, não havendo pontos em que bruscamente reduzem-se ou aumentam-se as áreas, ou flutuações pontuais sem acompanhar a tendência. A curva de áreas seccionais do hélice usado para demonstração dos procedimentos, B4-85 é expressa na Figura 70. A curva obtida dá uma boa indicação que as áreas seccionais do propulsor modelado seguem uma variação desejada e suave.

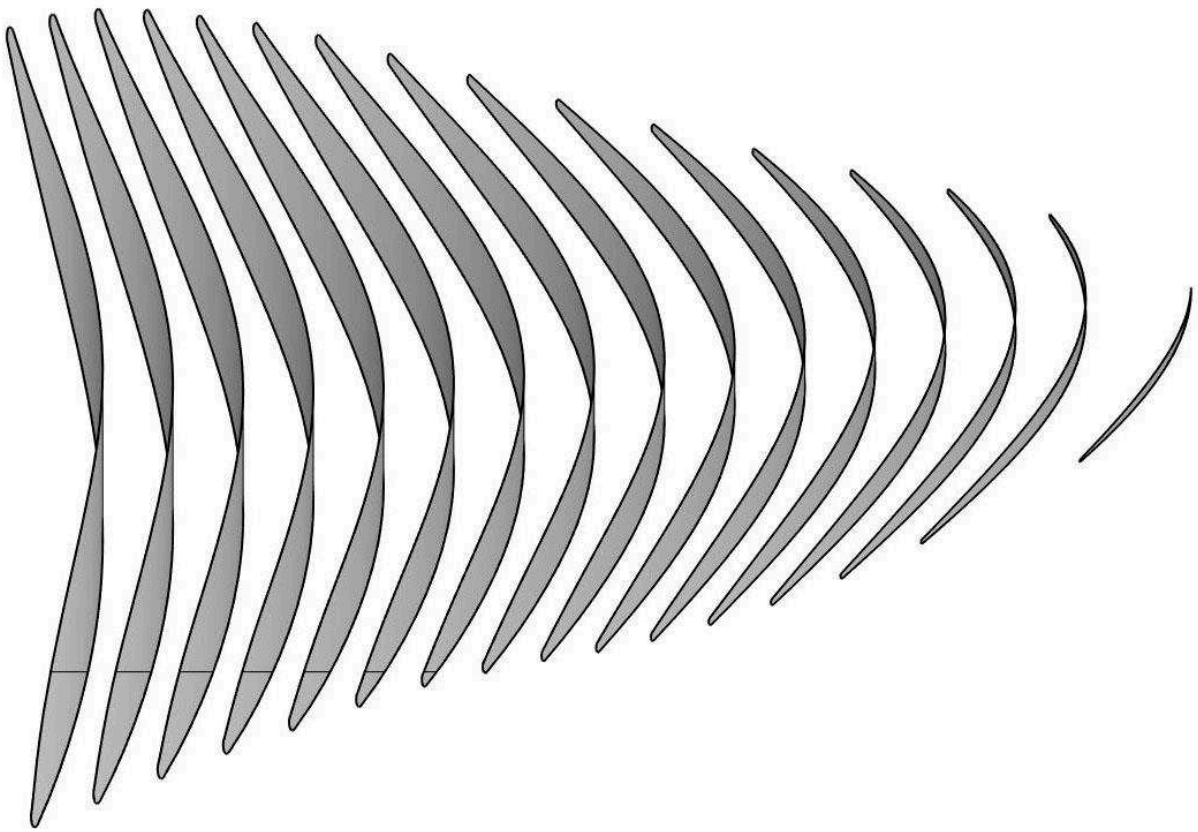
Figura 70: Curva áreas seccionais B4-85



Fonte: Autor (2016)

As áreas plotadas na Figura 70 foram seccionadas da lamina conforme ilustrado na Figura 71.

Figura 71: Áreas seccionais B4-85

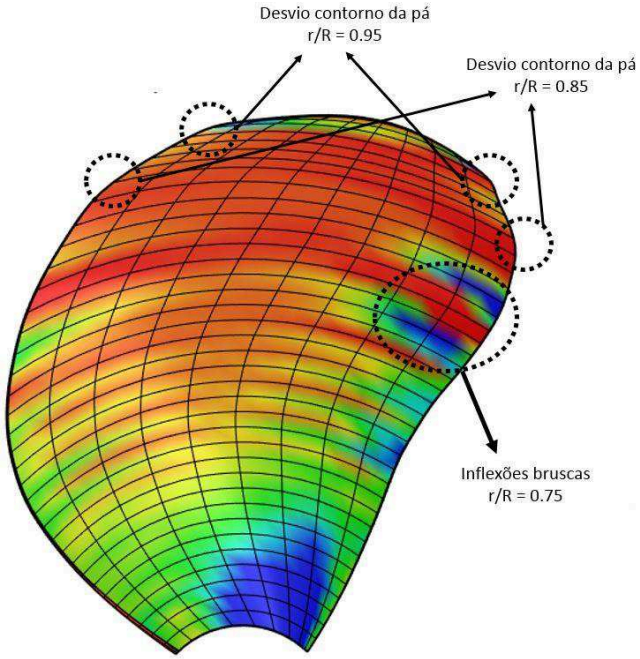


Fonte: Autor (2016)

5.3 INSPEÇÃO VISUAL

Através de inspeção visual com o auxílio das ferramentas de análise de superfície do Rhinoceros, constatou-se que a melhor modelagem se dava quando se excluíam as seções de r/R = 0.75 e r/R = 0.85, além dos pontos de contorno de pá para a seção de r/R = 0.95. Essas seções estavam causando inflexões e desvios de curvatura não desejados na modelagem, conforme ilustra-se nas Figuras 72.

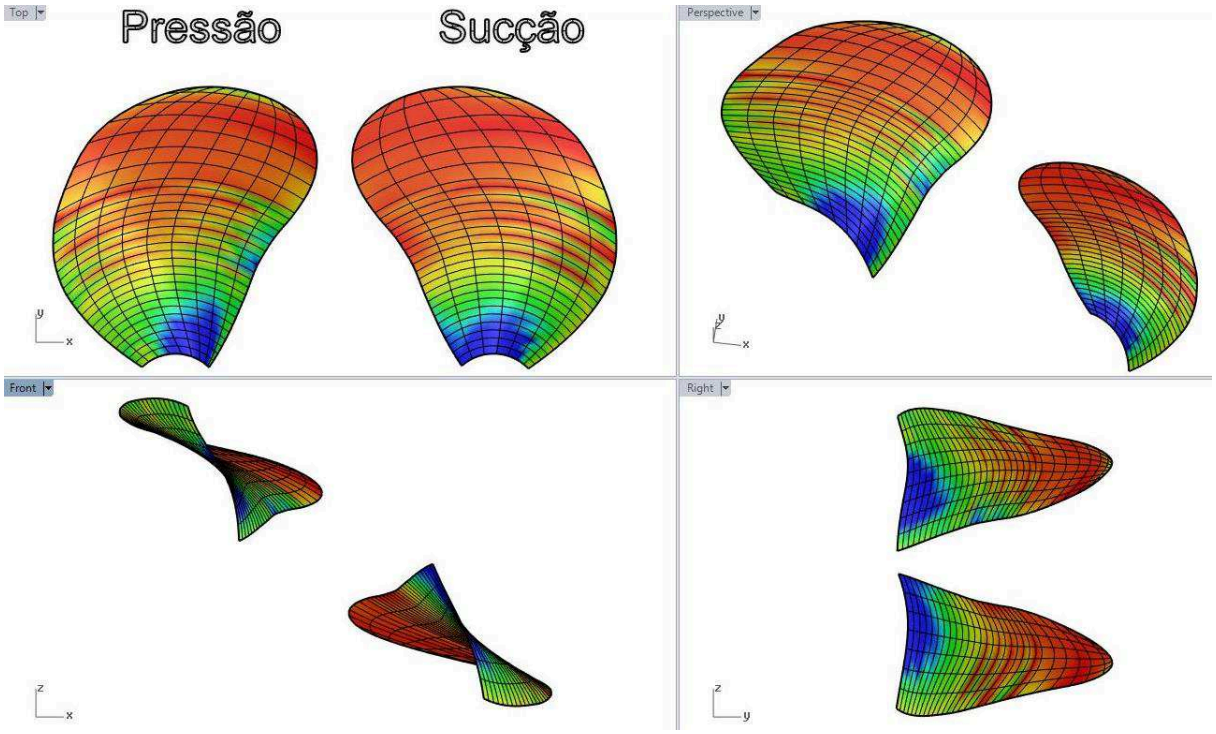
Figura 72: Seções removidas



Fonte: Autor (2016)

Após a remoção das seções descritas acima, inspecionou-se a qualidade da modelagem final através das funções visuais de análise de qualidade de superfície *CurvatureAnalysis, EMap e Zebra,* além da função de análise de objeto *DraftAngleAnalysis.*

Figura 73: Rhinoceros *CurvatureAnalysis*

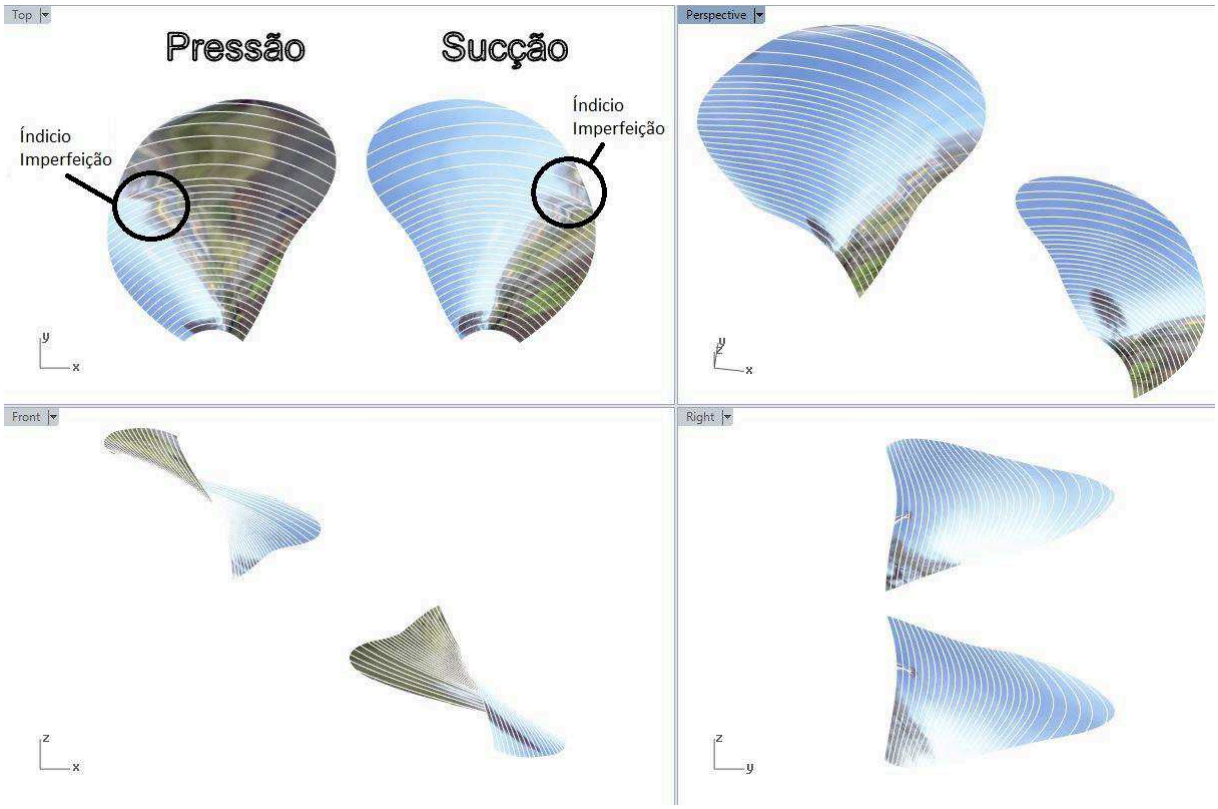


Fonte: Autor (2016)

A função *CuvatureAnalysis* avalia as curvaturas usando cores, sendo as curvaturas positivas em vermelho e as negativas em azul, enquanto as transições são apresentadas gradativamente entre vermelho e azul. A curvatura apresentada por essa análise é calculada pelo produto das principais curvaturas da superfície. São indesejadas mudanças de cores bruscas do vermelho para o azul, ou do azul para o vermelho, pois isso seriam pontos de inflexão bruscos.

Como pode ser observardo na Figura 73, para o hélice B4-85, não são apresentados regiões com inversão de curvatura bruscas, um índício qualitativo a respeito da suavidade da superfície gerada. Comparando-se as Figuras 72 e 73 fica claro que a qualidade da superfície melhorou com a retirada dos pontos da seção r/R =0.75 da modelagem.

Figura 74: Rhinoceros *E-Map*

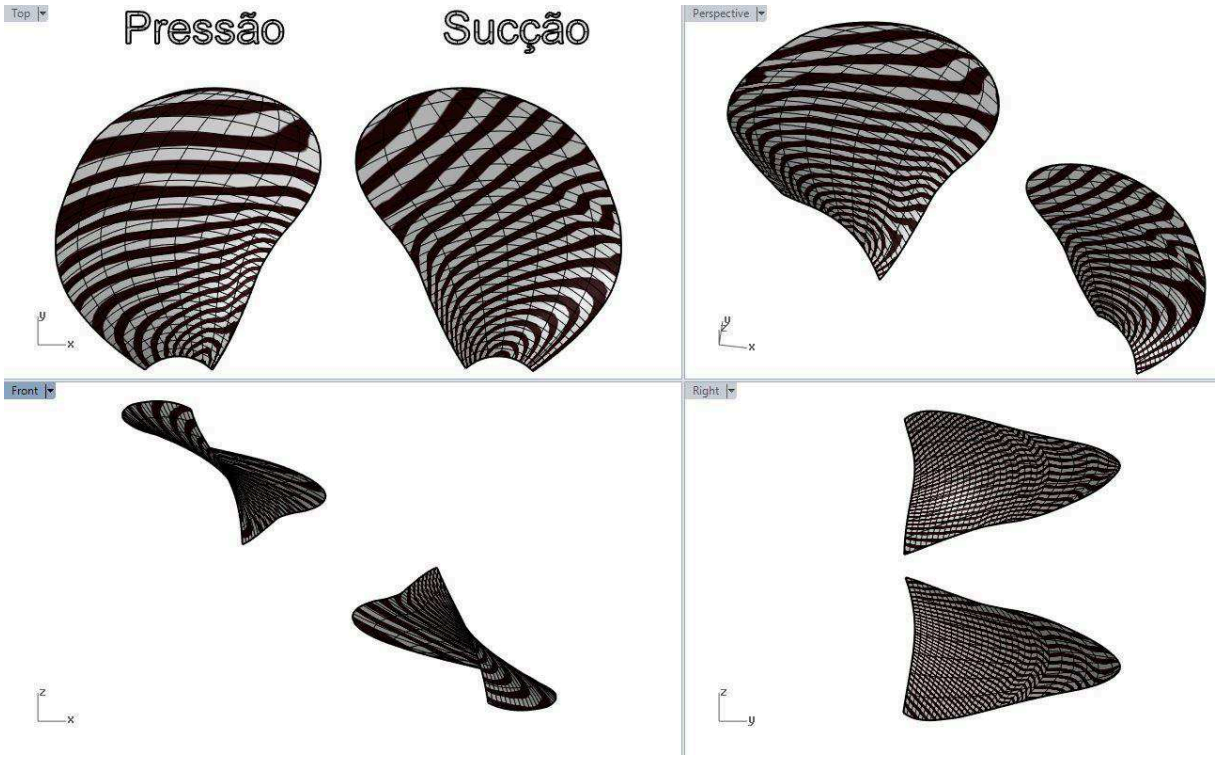


Fonte: Autor (2016)

A função *E-Map,* Figura 74, possibilita ao usuário inspecionar a suavidade da curvatura de uma superfície simulando um ambiente refletido sobre ela. Pequenas deformações são difíceis de serem observadas a olho nu, porém ao observa-se um reflexo em uma superfície polida, pequenas distorções são consideravelmente acentuadas, sendo essa a ideia por trás dessa análise.

Segundo a Figura 74, observa-se uma pequena distorção no reflexo nas regiões destacadas por círculos, por volta de r/R = 0.6, em ambas as faces do propulsor. Sendo este um índicio de uma pequena imperfeição na superfície.

Figura 75: Rhinoceros *Zebra*



Fonte: Autor (2016)

A função *Zebra,* Figura 75, avalia a suavidade e continuidade de uma superfície usando um mapa de faixas. Caso as faixas sejam continuas por toda a superfície isso expressa uma continuidade da superfície do tipo G2, isto é, continuidade de posição, curvatura e tangencial. Conforme observamos na Figura 75, existe um pequeno indício de imperfeição, já notados na Figura 74, na altura de r/R = 0.6, entretanto a continuidade das linhas nos garante que a superfície é contínua como um todo em cada face.

6. CONCLUSÃO

Os método de geração de geometria e modelagem automatizada do presente trabalho foram satisfatórios e apresentaram resultados consistentes com as informações disponíveis na literatura. Obteve-se o o valor da massa do hélice mais próximo do real, erro relativo de -0.7%, em comparação com os resultados obtidos por outros autores. Outro método que poderia ter sido utilizado para validação da modelagem é o valor do momento de inércia em relação ao eixo de rotação do propulsor, seu valor nas referências é de 238 kg-m^2 , porém o Rhinoceros por trabalhar com superfícies em vez de sólidos não tem uma funcionalidade que permita avalia-lo.

O processo de modelagem é extremamente rápido, em questão de segundos o MATLAB gera o arquivo .csv que também em questão de segundos é lido e modelado no Rhinoceros. A interface gráfica do MATLAB é intuitiva e ao mesmo tempo permite que, caso o usuário tenha interesse em alterar qualquer parte da programação, está facilmente acessível. O mesmo vale para o Rhinoceros, a rotina pode ser executada sem expor o código, porém existe também a possibilidade de abri-la e executá-la diretamente no editor, alterando-a como for conveniente.

A única limitação da automatização da modelagem é em questão dos filetes conectando as pás ao cubo, limitação essa também presente nos trabalhos de automatização dos autores citados anteriormente. Além do fato de não existir uma interligação direta entre o MATLAB e Rhinoceros.

Considerando a abrangência da Série B a respeito do número de pás e faixas de valores aceitos para EAR e P/D é enorme o número de configuração possíveis. Foram conduzidos testes com uma quantidade limitada de configurações, portanto, são necessários mais testes para validar a geração e modelagem sobre toda a abrangência da série. Caso o usuário insira valores menores do que 2 ou maiores que 7 para número de pás, o software realizará os cálculos, porém uma mensagem será exibida alertando sobre a não consistendia dos resultados.

Através da inspeção visual pelas ferramentas do Rhinoceros fica claro que apesar de não possuir uma curvatura perfeitamente suave, não foi verificada nenhuma mudança brusca de curvatura ou imperfeições pontuais, resultando em

superfícies com qualidade adequada para o trabalho proposto. A curva de áreas seccionais também se comportou da forma desejada.

O trabalho desenvolvido pode ser usado de forma acadêmica, auxiliando os estudantes a respeito do processo de modelagem em CAD de propulsores, tema pouco encontrado na literatura. Atualmente a UFSC-Joinville dispõe do Orca 3D, plug-in do Rhinoceros, e do Maxsurf capazes de gerar cascos de embarcações rapidamente em CAD, porém carece de qualquer ferramenta com esse intuito a respeito de propulsores.

6.1 TRABALHOS FUTUROS

- Integração com software CNC;
- Conversão da programação em MATLAB para Python;
- Conversão da rotina em Iron Python para plug-in do Rhinoceros;
- Ampliar a base de dados inserindo outras séries sistemáticas;
- Integração da interface com trabalho já desenvolvido na área de optimização da seleção, tornando uma ferramenta abrangente desde a seleção até a modelagem de propulsores.
- Automatização de filetes de raiz e cubo conforme normas.

REFERÊNCIAS

ABBOTT, I.H., DOENHOFF, A.E. von. **Theory of Wing Sections**. Dover, New York, 1959.

BIRK, L. ***Curved Surface Design.*** New Orleans, Louisiana. Notas de Aula. 2014

BODDY, T. A. S. ***Automated Drawing of Marine Screw Propellers Using Rhinoceros.*** Sydney, Austrália. 2014.

CARLTON, John. ***Marine propellers and propulsion***. Butterworth-Heinemann, 2012.

COHEN, Marcelo. **Computação Gráfica** ;Turma 480. 22-abr a 08-jul de 2012. Notas de Aula.

CRAIG, J. I. **Fundamentals of Computer Aided Engineering and Design**; AE4375 & AE6380. Set, 2009. Notas de Aula.

CACHO, A.J.S.C, SANTOS,T.A.R.S.,SILVA, S.B. N. R.S. **Arquitectura Naval.** Técnico Lisboa, Novembro, 2014. Notas de Aula.

DANG, J., BOOM, H.J.J van den, LIGTELIN, J. TH.. **The Wageningen C-and D-Series Propellers**. Holanda. 2013.

FOWLER, D. B. **Automated Drawing of Marine Screw Propellers Using AutoCAD.** Sydney, Austrália. 2012.

GEER, VE. ***Propeller handbook***, 1989.

GUEST, Philip George. **Numerical Methods of Curve Fitting**. 2012

KUIPER, G. ***The Wageningen Propeller Series,*** 1992.

LIM, J. **Automating Drawing of Ship Propellers with Computer-Aided Drafting.** Sydney, Austrália. 2001.

RUTEN, David. ***Python for Rhinoceros 5***, 2011.

TAKINACI, Ali Can. **Advanced Propulsion System**; GEM 423E. Notas de Aula, 2015. Disponível em: < http://160.75.46.2/staff/takinaci/advpropsys_index.html>. Acesso em: 2016-06-29.

VILLAS BOAS, Fábio. **Desenvolvimento de uma ferramenta de CAD aplicada ao projeto de hélices para veículos aquáticos não tripulados**. 2006. Dissertação (Mestrado em Engenharia Mecânica) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2006. Disponível em:

<http://www.teses.usp.br/teses/disponiveis/3/3132/tde-06062006-105913/>. Acesso em: 2016-06-10.

ZUCHER, K. **Automated Drawing of Marine Screw Propellers.** Sydney, Australia. 2009.

APÊNDICE A

Rotina Interface.m

```
function varargout = Interface(varargin)
% INTERFACE MATLAB code for Interface.fig
%      INTERFACE, by itself, creates a new INTERFACE or raises the existing
%      singleton*.
%
%      H = INTERFACE returns the handle to a new INTERFACE or the handle to
%      the existing singleton*.
%
%      INTERFACE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INTERFACE.M with the given input arguments.
%
%      INTERFACE('Property','Value',...) creates a new INTERFACE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Interface_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Interface_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Interface

% Last Modified by GUIDE v2.5 28-Jun-2016 22:04:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Interface_OpeningFcn, ...
                   'gui_OutputFcn',  @Interface_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Interface is made visible.
function Interface_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Interface (see VARARGIN)

% Choose default command line output for Interface
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Interface wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Interface_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in chb_Zero75.
function chb_Zero75_Callback(hObject, eventdata, handles)
% hObject    handle to chb_Zero75 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chb_Zero75


% --- Executes on button press in chb_Zero85.
function chb_Zero85_Callback(hObject, eventdata, handles)
% hObject    handle to chb_Zero85 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chb_Zero85


% --- Executes on button press in chb_Zero95.
function chb_Zero95_Callback(hObject, eventdata, handles)
% hObject    handle to chb_Zero95 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of chb_Zero95


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% ENTRADA DOS USUARIO
% Obtenção dos dados informados na interface gráfica pelo usuário

D = str2num(get(handles.etxt_D,'string'));
EAR = str2num(get(handles.etxt_EAR,'string'));
Z = str2num(get(handles.etxt_Z,'string'));
if Z > 7
    msgbox 'Abrangência B-Series 2 a 7 pás. Fora disso o arquivo será gerado,
porem não será correto. Aguarde o termino da execução e modifique o valor.'
end
PD = str2num(get(handles.etxt_PD,'string'));

% D = 4;
% EAR = 0.85;
% Z = 4;
% PD = 0.9;

NumeroLaminas = Z;
RakeValue = 15;

Entradas = table(D,EAR,Z,RakeValue,PD);
display('Geração de Geometria baseada em:'),Entradas

%% TABELAS KUIPPER (1992) - THE WAGENINGE PROPELLER SERIES

% Table 4.2: Blade contour of the BB-series propellers.
Tabela4Ponto02.rR = [0.2 ; 0.3 ; 0.4 ; 0.5 ; 0.6 ; 0.7 ; 0.8 ;0.85 ; 0.9; 0.95; 0.975]; %
r/R
Tabela4Ponto02.Kr = [1.6 ; 1.832 ; 2.023; 2.163 ; 2.243 ; 2.247 ; 2.132; 2.005; 1.798
; 1.434; 1.220]; % K(r)
Tabela4Ponto02.skewCr = [0.081; 0.084 ; 0.080; 0.070; 0.052; 0.024; -0.020 ; -0.052;
-0.098; -0.182; -0.273]; %skew over cr
Tabela4Ponto02.Table = table (Tabela4Ponto02.rR,Tabela4Ponto02.Kr ,
Tabela4Ponto02.skewCr);

% Table 4.5: Suction side section geometry from position of maximum
% thickness to the trailing edge.
Tabela4Ponto05.rR = [0.2 ; 0.3 ; 0.4 ; 0.5 ; 0.6 ; 0.7 ; 0.8 ; 0.9]; % r/R
```

```
Tabela4Ponto05.Oitenta = [53.35; 50.95; 47.70; 43.40; 40.20; 39.40; 40.95; 45.15];
% 80%
Tabela4Ponto05.Sessenta = [72.65 ; 71.60 ; 70.25 ; 68.40; 67.15; 66.90; 67.80; 70];
% 60%
Tabela4Ponto05.Quarenta = [86.9; 86.8; 86.55; 86.10; 85.40; 84.90; 85.30; 87];      %
40%
Tabela4Ponto05.Vinte = [96.45; 96.80; 97; 96.95; 96.8; 96.65; 96.7; 97];           %
20%
Tabela4Ponto05.Table = table(Tabela4Ponto05.rR,Tabela4Ponto05.Oitenta,...
    Tabela4Ponto05.Sessenta, Tabela4Ponto05.Quarenta,Tabela4Ponto05.Vinte);

% Table 4.7: Coefficients V1 of sextion geometry between point of maximum
% thickness and trailing edge.
Tabela4Ponto07.rR = [0.7; 0.6; 0.5; 0.4; 0.3; 0.25; 0.20; 0.15]; % r/R
Tabela4Ponto07.TE = [0; 0; 0.0522; 0.1467; 0.2306; 0.2598; 0.2826; 0.3];
%T.E.
Tabela4Ponto07.Noventa5 = [0; 0; 0.042; 0.12; 0.2040; 0.2372; 0.2630; 0.2824];
% 95%
Tabela4Ponto07.Noventa = [0; 0; 0.0330; 0.0972; 0.1790; 0.2115; 0.24; 0.2650];
% 90%
Tabela4Ponto07.Oitenta = [0; 0; 0.0190; 0.0630; 0.1333; 0.1651; 0.1967; 0.23];     %
80%
Tabela4Ponto07.Setenta = [0; 0; 0.01; 0.0395; 0.0943; 0.1246; 0.1570; 0.1950];     %
70%
Tabela4Ponto07.Sessenta = [0; 0; 0.0040; 0.0214; 0.0623; 0.0899; 0.1207; 0.1610];
% 60%
Tabela4Ponto07.Cinquenta = [0; 0; 0.0012; 0.0116; 0.0376; 0.0579; 0.0880; 0.1280];
% 50%
Tabela4Ponto07.Quarenta = [0; 0; 0; 0.0044; 0.0202; 0.0350; 0.0592; 0.0955];      %
40%
Tabela4Ponto07.Vinte = [0; 0; 0; 0; 0.0033; 0.0084; 0.0172; 0.0365];              % 20%
Tabela4Ponto07.Table = table (Tabela4Ponto07.TE, Tabela4Ponto07.Noventa5,...
    Tabela4Ponto07.Noventa, Tabela4Ponto07.Oitenta, Tabela4Ponto07.Setenta, ...
    Tabela4Ponto07.Sessenta, Tabela4Ponto07.Cinquenta,
Tabela4Ponto07.Quarenta,...
    Tabela4Ponto07.Vinte);

% Table 4.8: Coefficients V1 of section geometry between point and maximum
% thickness and leading edge.
Tabela4Ponto08.rR = [1.0; 0.9;0.85; 0.8; 0.7; 0.6; 0.5; 0.4; 0.3; 0.25; 0.20; 0.15]; %
r/R
Tabela4Ponto08.Vinte = [0;0; 0; 0;0;0; 0; 0; 0.0027; 0.0031; 0.0049; 0.0096];
% 20%
Tabela4Ponto08.Quarenta = [0;0; 0; 0;0; 0; 0; 0.0033; 0.0148; 0.0224; 0.0304;
0.0384];          % 40%
Tabela4Ponto08.Cinquenta = [0;0; 0; 0;0; 0; 0.0008; 0.0090; 0.0300; 0.0417; 0.0520;
0.0615];     % 50%
Tabela4Ponto08.Sessenta = [0;0; 0; 0;0;0; 0.0034; 0.0189; 0.0503; 0.0669; 0.0804;
0.0920];       % 60%
```

```
Tabela4Ponto08.Setenta = [0;0; 0; 0;0;0; 0.0085; 0.0357; 0.0790; 0.1008; 0.118;
0.1320];        % 70%
Tabela4Ponto08.Oitenta = [0;0; 0; 0;0; 0.0006; 0.0211; 0.0637; 0.1191; 0.1465;
0.1685; 0.1870];  % 80%
Tabela4Ponto08.Oitenta5 = [0;0; 0; 0;0; 0.0022; 0.0328; 0.0833; 0.1445; 0.1747; 0.2;
0.223];     % 85%
Tabela4Ponto08.Noventa = [0;0; 0; 0;0; 0.0067; 0.0500; 0.1088; 0.1760; 0.2068;
0.2353; 0.2642];  % 90%
Tabela4Ponto08.Noventa5 = [0;0; 0; 0;0; 0.0169; 0.0778; 0.1467; 0.2186; 0.2513;
0.2821; 0.3150]; % 95%
Tabela4Ponto08.LE = [0;0; 0; 0;0; 0.0382; 0.1278; 0.2181; 0.2923; 0.3256; 0.3560;
0.3860];       % L.E.
Tabela4Ponto08.Table = table (Tabela4Ponto08.Vinte, Tabela4Ponto08.Quarenta,...
    Tabela4Ponto08.Cinquenta, Tabela4Ponto08.Sessenta, Tabela4Ponto08.Setenta,
...
    Tabela4Ponto08.Oitenta, Tabela4Ponto08.Oitenta5, Tabela4Ponto08.Noventa, ...
    Tabela4Ponto08.Noventa5, Tabela4Ponto08.LE);

% Table 4.9: Coefficients V2 of section geometry between point of maximum
% thickness and trailing edge.
Tabela4Ponto09.rR = [0.9; 0.85; 0.8; 0.7; 0.6; 0.5; 0.4; 0.3; 0.25; 0.20; 0.15]; %r/R
Tabela4Ponto09.TE = [0; 0; 0;0; 0; 0; 0; 0; 0; 0; 0]; % T.E.
Tabela4Ponto09.Noventa5 = [0.0975;0.0975;0.0975;0.0975; 0.0965; 0.0950; 0.0905;
0.0800; 0.0725; 0.0640; 0.0540]; % 95%
Tabela4Ponto09.Noventa = [0.19;0.19;0.19;0.19; 0.1885; 0.1865; 0.1810; 0.1670;
0.1567; 0.1455; 0.1325];    % 90%
Tabela4Ponto09.Oitenta = [0.36;0.36;0.36;0.36; 0.3585; 0.3569; 0.35; 0.3360;
0.3228; 0.3060; 0.2870];      % 80%
Tabela4Ponto09.Setenta = [0.51;0.51;0.51;0.51; 0.5110; 0.5140; 0.5040; 0.4885;
0.4740; 0.4535; 0.4280];    % 70%
Tabela4Ponto09.Sessenta = [0.64;0.64;0.64;0.64; 0.6415; 0.6439; 0.6353; 0.6195;
0.6050; 0.5842; 0.5585];   % 60%
Tabela4Ponto09.Cinquenta = [0.75;0.75;0.75;0.75; 0.7530; 0.7580; 0.7525; 0.7335;
0.7184; 0.6995; 0.6770];  % 50%
Tabela4Ponto09.Quarenta = [0.84;0.84;0.84;0.84; 0.8426; 0.8456; 0.8415; 0.8265;
0.8139; 0.7984; 0.7805];   % 40%
Tabela4Ponto09.Vinte = [0.96;0.96;0.96;0.96; 0.9613; 0.9639; 0.9645; 0.9583;
0.9519; 0.9446; 0.9360];      % 20%
Tabela4Ponto09.Table = table (Tabela4Ponto09.TE, Tabela4Ponto09.Noventa5,...
    Tabela4Ponto09.Noventa,Tabela4Ponto09.Oitenta, Tabela4Ponto09.Setenta,...
    Tabela4Ponto09.Sessenta, Tabela4Ponto09.Cinquenta,
Tabela4Ponto09.Quarenta, ...
    Tabela4Ponto09.Vinte);

% Table 4.10: Coefficients V2 of section geometry between point of maximum
% thickness and leading edge.
Tabela4Ponto10.rR = [0.9; 0.85; 0.80; 0.7; 0.6; 0.5; 0.4; 0.3; 0.25; 0.20; 0.15]; % r/R
Tabela4Ponto10.Vinte = [0.96; 0.9615; 0.9635; 0.9675; 0.9690; 0.9710; 0.9725;
0.9750; 0.9751; 0.9750; 0.9760];     % 20%
```

```
Tabela4Ponto10.Quarenta = [0.84; 0.8450; 0.8520; 0.8660; 0.8790; 0.888; 0.8933;
0.8920; 0.8899; 0.8875; 0.8825];   % 40%
Tabela4Ponto10.Cinquenta = [0.75; 0.7550; 0.7635; 0.7850; 0.8090; 0.8275; 0.8345;
0.8315; 0.8259; 0.8170; 0.8055]; % 50%
Tabela4Ponto10.Sessenta = [0.64; 0.6455; 0.6545; 0.6840; 0.72; 0.7478; 0.7593;
0.7520; 0.7415; 0.7277; 0.7105];    % 60%
Tabela4Ponto10.Setenta = [0.51; 0.5160; 0.5265; 0.5615; 0.6060; 0.6430; 0.6590;
0.6505; 0.6359; 0.6190; 0.5995];   % 70%
Tabela4Ponto10.Oitenta = [0.36; 0.3660; 0.3765; 0.4140; 0.4620; 0.5039; 0.5220;
0.5130; 0.4982; 0.4777; 0.4520];   % 80%
Tabela4Ponto10.Oitenta5 = [0.2775; 0.2830; 0.2925; 0.33; 0.3775; 0.4135; 0.4335;
0.4265; 0.4108; 0.3905; 0.3665];  % 85%
Tabela4Ponto10.Noventa = [0.19; 0.1950; 0.2028; 0.2337; 0.2720; 0.3056; 0.3235;
0.3197; 0.3042; 0.2840; 0.2600];   % 90%
Tabela4Ponto10.Noventa5 = [0.0975; 0.1; 0.1050; 0.1240; 0.1485; 0.1750; 0.1935;
0.1890; 0.1750; 0.1560; 0.13];     % 95%
Tabela4Ponto10.LE = [0;0;0;0;0;0;0;0;0;0;0];
Tabela4Ponto10.Table = table (Tabela4Ponto10.Vinte, Tabela4Ponto10.Quarenta,...
    Tabela4Ponto10.Cinquenta, Tabela4Ponto10.Sessenta, Tabela4Ponto10.Setenta,
...
    Tabela4Ponto10.Oitenta,Tabela4Ponto10.Oitenta5, Tabela4Ponto10.Noventa,...
    Tabela4Ponto10.Noventa5,Tabela4Ponto10.LE);

% Table 4.11: Maximum thickness and position of maximum thickness.
Tabela4Ponto11.rR = [0.2 ; 0.3 ; 0.4 ; 0.5 ; 0.6 ; 0.7 ; 0.8 ; 0.9 ; 1.0]; % r/R
Tabela4Ponto11.bladed3 = [0.0406; 0.359; 0.0312; 0.0265; 0.0218; 0.0171; 0.0124;
0.0077; 0.0030];  %tmax/D para 3-laminas
Tabela4Ponto11.bladed4 = [0.0366; 0.0324; 0.0282; 0.0240; 0.0198; 0.0156; 0.0114;
0.0072; 0.0030]; %tmax/D para 4-laminas
Tabela4Ponto11.bladed5 = [0.0326; 0.0289; 0.0252; 0.0215; 0.0178; 0.0141; 0.0104;
0.0067; 0.0030]; %tmax/D para 5-laminas
Tabela4Ponto11.bladed6 = [0.0286; 0.0254; 0.0222; 0.0190; 0.0158; 0.0126; 0.0094;
0.0062; 0.0030]; %tmax/D para 6-laminas
Tabela4Ponto11.bladed7 = [0.0246; 0.0219; 0.0192; 0.0165; 0.0138; 0.0111; 0.0084;
0.0057; 0.0030]; %tmax/D para 7-laminas
Tabela4Ponto11.xtmaxcr = [0.350; 0.350; 0.351; 0.355; 0.389; 0.443; 0.486; 0.5;
0.5]; %xtmax/cr
Tabela4Ponto11.Table = table (Tabela4Ponto11.bladed3,
Tabela4Ponto11.bladed4,...
    Tabela4Ponto11.bladed5, Tabela4Ponto11.bladed6, Tabela4Ponto11.bladed7,...
    Tabela4Ponto11.xtmaxcr);

% Table 4.12: Coefficients for thickness distribution of B-series
% propellers
Tabela4Ponto12.radius = [0.2 ; 0.3 ; 0.4 ; 0.5 ; 0.6 ; 0.7 ; 0.8 ; 0.9; 1.0]; % r/R
Tabela4Ponto12.Ar = [0.0526 ; 0.0464; 0.0402 ; 0.0340; 0.0278; 0.0216; 0.0154;
0.0092; 0.0030]; %Ar
Tabela4Ponto12.Br = [0.0040; 0.0035; 0.0030; 0.0025; 0.0020; 0.0015; 0.0010;
0.0005; 0.000]; % Br
```

```
Tabela4Ponto12.Table =
table(Tabela4Ponto12.radius,Tabela4Ponto12.Ar,Tabela4Ponto12.Br);

%% TABELAS MODIFICADAS POR INTERPOLAÇÕES/EXTRAPOLAÇÕES

% Tabela 4.2
% Foram extrapolados e interpolados os pontos para r/R = 0.15 e 0.25

X = [0.2 ; 0.3 ; 0.4 ; 0.5 ; 0.6 ; 0.7 ; 0.8 ;0.85 ; 0.9; 0.95; 0.975];
Y1 = Tabela4Ponto02.Kr;
Y2 = Tabela4Ponto02.skewCr;

% Fit: 'Tabela 4.2 - K(r)'.
    [xData, yData] = prepareCurveData( X, Y1 );
    % Set up fittype and options.
    ft = fittype( 'poly6' );
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft );
    InterpoladoZero15 = fitresult(0.15);
    InterpoladoZero25 = fitresult(0.25);
    PolinomioKr = fitresult;
    Tabela4Ponto02.KrInterpolado = [InterpoladoZero15; 1.6; InterpoladoZero25;...
        1.832 ; 2.023; 2.163 ; 2.243 ; 2.247 ; 2.132; 2.005; 1.798 ; 1.434; 1.220];
    clear Y1
    clear xData
    clear yData
    clear ft
    clear fitresult
    clear gof
    clear InterpoladoZero15
    clear InterpoladoZero25
    clear InterpoladoZero85

% Fit: 'Tabela 4.2 - skew/cr'.
    [xData, yData] = prepareCurveData( X, Y2 );
    % Set up fittype and options.
    ft = fittype( 'smoothingspline' );
    opts = fitoptions( 'Method', 'SmoothingSpline' );
    opts.SmoothingParam = 1;
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft, opts );
    InterpoladoZero15 = fitresult(0.15);
    InterpoladoZero25 = fitresult(0.25);
    PolinomioSkewCr = fitresult;
    Tabela4Ponto02.skewCrInterpolado = [InterpoladoZero15; 0.081;...
        InterpoladoZero25; 0.084 ; 0.080; 0.070; 0.052; 0.024; -0.020 ;...
        -0.052; -0.098; -0.182; -0.273];
    clear Y2
    clear xData
    clear yData
```

```
    clear ft
    clear opts
    clear fitresult
    clear gof
    clear InterpoladoZero15
    clear InterpoladoZero25
    clear InterpoladoZero85

clear X
clear PolinomioKr
clear PolinomioSkewCr

Tabela4Ponto02.rRInterpolado = [0.15; 0.2;0.25; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8;0.85; 0.9;
0.95; 0.975];

% Tabela 4.11
% Foram interpolados os pontos para r/R = 0.15, 0.25 e 0.85
% Apenas para confirmar que os valores seriam iguais aos obtidos se
% usada a tabela 4.12

X = [0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0];
Y2 = Tabela4Ponto11.xtmaxcr;

% Fit: 'Tabela4.11 - xtmax / cr'.
    [xData, yData] = prepareCurveData( X, Y2 );
    % Set up fittype and options.
    ft = fittype( 'smoothingspline' );
    opts = fitoptions( 'Method', 'SmoothingSpline' );
    opts.Normalize = 'on';
    opts.SmoothingParam = 1;
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft, opts );
    % Os valores interpolados para 0.15 e 0.25 não seriam corretos,
    % portanto a esses dois foi-se atribuido manualmente o valor e a
    % interpolação somente foi efetuada para 0.85
    InterpoladoZero15 = 0.350;
    InterpoladoZero25 = 0.350;
    InterpoladoZero85 = fitresult(0.85);
    Tabela4Ponto11.xtmaxcr = [InterpoladoZero15; 0.350; InterpoladoZero25;
        0.350; 0.351; 0.355; 0.389; 0.443; 0.486; InterpoladoZero85; 0.500; 0.500];
    clear InterpoladoZero15
    clear InterpoladoZero25
    clear InterpoladoZero85
    clear xData
    clear yData
    clear ft
    clear opts
    clear fitresult
    clear gof
    clear Y2
```

```
clear X

% Tabela 4.12
% Foram extrapolados e interpolados os pontos para r/R = 0.15, 0.25 e 0.85

X = [0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0];
Y1 = Tabela4Ponto12.Ar;
Y2 = Tabela4Ponto12.Br;
Tabela4Ponto12.radiusInterpolado = [0.15; 0.2; 0.25; 0.3; 0.4; 0.5; 0.6;...
    0.7; 0.8; 0.85; 0.9; 1.0];
    % Fit: 'Tabela 4.12 - Ar'.
    [xData, yData] = prepareCurveData( X, Y1 );
    % Set up fittype and options.
    ft = fittype( 'poly1' );
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft );
    InterpoladoZero15 = fitresult(0.15);
    InterpoladoZero25 = fitresult(0.25);
    InterpoladoZero85 = fitresult(0.85);
    PolinomioAr = fitresult;
    Tabela4Ponto12.ArInterpolado = [InterpoladoZero15; 0.0526; InterpoladoZero25;
        0.0464; 0.0402; 0.0340; 0.0278; 0.0216; 0.0154; InterpoladoZero85;...
        0.0092; 0.0030];
    clear InterpoladoZero15
    clear InterpoladoZero25
    clear InterpoladoZero85
    clear xData
    clear yData
    clear ft
    clear fitresult
    clear gof
    clear Y1

    % Fit: 'Tabela 4.12 - Br'.
    [xData, yData] = prepareCurveData( X, Y2 );
    % Set up fittype and options.
    ft = fittype( 'poly1' );
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft );
    InterpoladoZero15 = fitresult(0.15);
    InterpoladoZero25 = fitresult(0.25);
    InterpoladoZero85 = fitresult(0.85);
    PolinomioBr = fitresult;
    Tabela4Ponto12.BrInterpolado = [InterpoladoZero15; 0.0040; InterpoladoZero25;...
        0.0035; 0.0030; 0.0025; 0.0020; 0.0015; 0.0010; InterpoladoZero85;...
        0.0005; 0.0000];
    clear InterpoladoZero15
    clear InterpoladoZero25
    clear InterpoladoZero85
    clear xData
```

```
    clear yData
    clear ft
    clear fitresult
    clear gof
    clear Y2
clear X
clear PolinomioBr
clear PolinomioAr

%% GEOMETRIA HÉLICE
% Equação (4.1): Comprimento de corda em função do raio:
% c(r) = (K(r) x D x EAR) / z

cr = Tabela4Ponto02.KrInterpolado*D*EAR/Z;

% Skew é a distancia entre a corda média da seção e a linha geradora.
% Skew positivo significa que a corda média está posicionada a frente
% da linha geradora.

skew = Tabela4Ponto02.skewCrInterpolado.*cr;

% Extrapolação para Skew em r/R = 1.0
X = Tabela4Ponto02.rRInterpolado;
[xData, yData] = prepareCurveData( X, skew );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
Tip.Skew100 = fitresult(1);
SkewSuave = fitresult(Tabela4Ponto02.rRInterpolado);
    clear xData
    clear yData
    clear ft
    clear fitresult
    clear gof
    clear X

% Equação (4.4): Espessura máxima encontrada em cada seção
% A posição delas está em função do comprimento de corda,
% "Tabela4Ponto12.xtmaxcr", xtmax/cr em relação ao bordo de
% ataque (leading edge)

% trD = Tabela4Ponto12.Ar - Tabela4Ponto12.Br*Z; % tr/D = Ar - Br*Z
% tr = trD*D; % tr = tmax

trD_Interpolado = Tabela4Ponto12.ArInterpolado - Tabela4Ponto12.BrInterpolado*Z;
% tr/D = Ar - Br*Z
tr_Interpolado = trD_Interpolado*D; % tr = tmax

clear trD_Interpolado
```

```
tmax = tr_Interpolado;
% As espessura máximas podem ser calculadas tanto pela Tabela 4.11 quanto 4.12.
%Resultam nos mesmos valores, mesmo com as interpolações

%% COORDENADAS Z

% Equação (4.2): Formulacão da geometria do lado de pressão
% Pressure side geometry is:  yface = V1 (tmax - tl.e.)

yface.TrailingEdge.Zero15 = transpose(Tabela4Ponto07.Table{8,:}.*((tmax(1) -
0.2*tmax(1)))); %r/R = 0.15
yface.TrailingEdge.Zero20 = transpose(Tabela4Ponto07.Table{7,:}.*((tmax(2) -
0.2*tmax(2))));  %r/R = 0.2
yface.TrailingEdge.Zero25 = transpose(Tabela4Ponto07.Table{6,:}.*((tmax(3) -
0.2*tmax(3)))); %r/R = 0.25
yface.TrailingEdge.Zero30 = transpose(Tabela4Ponto07.Table{5,:}.*((tmax(4) -
0.2*tmax(4))));  %r/R = 0.3
yface.TrailingEdge.Zero40 = transpose(Tabela4Ponto07.Table{4,:}.*((tmax(5) -
0.2*tmax(5))));  %r/R = 0.4
yface.TrailingEdge.Zero50 = transpose(Tabela4Ponto07.Table{3,:}.*((tmax(6) -
0.2*tmax(6))));  %r/R = 0.5
yface.TrailingEdge.Zero60 = transpose(Tabela4Ponto07.Table{2,:}.*((tmax(7) -
0.2*tmax(7))));  %r/R = 0.6
yface.TrailingEdge.Zero70 = transpose(Tabela4Ponto07.Table{1,:}.*((tmax(8) -
0.2*tmax(8))));  %r/R = 0.7

yface.LeadingEdge.Zero15 = transpose(Tabela4Ponto08.Table{12,:}.*((tmax(1) -
0.2*tmax(1)))); %r/R = 0.15
yface.LeadingEdge.Zero20 = transpose(Tabela4Ponto08.Table{11,:}.*((tmax(2) -
0.2*tmax(2)))); %r/R = 0.2
yface.LeadingEdge.Zero25 = transpose(Tabela4Ponto08.Table{10,:}.*((tmax(3) -
0.2*tmax(3)))); %r/R = 0.25
yface.LeadingEdge.Zero30 = transpose(Tabela4Ponto08.Table{9,:}.*((tmax(4) -
0.2*tmax(4))));  %r/R = 0.3
yface.LeadingEdge.Zero40 = transpose(Tabela4Ponto08.Table{8,:}.*((tmax(5) -
0.2*tmax(5))));  %r/R = 0.4
yface.LeadingEdge.Zero50 = transpose(Tabela4Ponto08.Table{7,:}.*((tmax(6) -
0.2*tmax(6))));  %r/R = 0.5
yface.LeadingEdge.Zero60 = transpose(Tabela4Ponto08.Table{6,:}.*((tmax(7) -
0.2*tmax(7))));  %r/R = 0.6
yface.LeadingEdge.Zero70 = transpose(Tabela4Ponto08.Table{5,:}.*((tmax(8) -
0.2*tmax(8))));  %r/R = 0.7

% Equação (4.3): Distribuição de espessura
% Thickness distribution is: tr = V2 (tmax - tl.e.) + tl.e.

tr1.TrailingEdge.Zero15 = (transpose(Tabela4Ponto09.Table{11,:}.*
(tmax(1)*0.8)))+(0.2*tmax(1)); %r/R = 0.15
```

```
tr1.TrailingEdge.Zero20 = (transpose(Tabela4Ponto09.Table{10,:}.*
(tmax(2)*0.8)))+(0.2*tmax(2)); %r/R = 0.2
tr1.TrailingEdge.Zero25 = (transpose(Tabela4Ponto09.Table{9,:}.*
(tmax(3)*0.8)))+(0.2*tmax(3)); %r/R = 0.25
tr1.TrailingEdge.Zero30 = (transpose(Tabela4Ponto09.Table{8,:}.*
(tmax(4)*0.8)))+(0.2*tmax(4));  %r/R = 0.3
tr1.TrailingEdge.Zero40 = (transpose(Tabela4Ponto09.Table{7,:}.*
(tmax(5)*0.8)))+(0.2*tmax(5));  %r/R = 0.4
tr1.TrailingEdge.Zero50 = (transpose(Tabela4Ponto09.Table{6,:}.*
(tmax(6)*0.8)))+(0.2*tmax(6));  %r/R = 0.5
tr1.TrailingEdge.Zero60 = (transpose(Tabela4Ponto09.Table{5,:}.*
(tmax(7)*0.8)))+(0.2*tmax(7));  %r/R = 0.6
tr1.TrailingEdge.Zero70 = (transpose(Tabela4Ponto09.Table{4,:}.*
(tmax(8)*0.8)))+(0.2*tmax(8));  %r/R = 0.7
tr1.TrailingEdge.Zero80 = (transpose(Tabela4Ponto09.Table{3,:}.*
(tmax(9)*0.8)))+(0.2*tmax(9));  %r/R = 0.8
tr1.TrailingEdge.Zero85 = (transpose(Tabela4Ponto09.Table{2,:}.*
(tmax(10)*0.8)))+(0.2*tmax(10));  %r/R = 0.85
tr1.TrailingEdge.Zero90 = (transpose(Tabela4Ponto09.Table{1,:}.*
(tmax(11)*0.8)))+(0.2*tmax(11));  %r/R = 0.90

tr1.LeadingEdge.Zero15 = (transpose(Tabela4Ponto10.Table{11,:}.*
(tmax(1)*0.8)))+(0.2*tmax(1)); %r/R = 0.15
tr1.LeadingEdge.Zero20 = (transpose(Tabela4Ponto10.Table{10,:}.*
(tmax(2)*0.8)))+(0.2*tmax(2)); %r/R = 0.2
tr1.LeadingEdge.Zero25 = (transpose(Tabela4Ponto10.Table{9,:}.*
(tmax(3)*0.8)))+(0.2*tmax(3)); %r/R = 0.25
tr1.LeadingEdge.Zero30 = (transpose(Tabela4Ponto10.Table{8,:}.*
(tmax(4)*0.8)))+(0.2*tmax(4)); %r/R = 0.3
tr1.LeadingEdge.Zero40 = (transpose(Tabela4Ponto10.Table{7,:}.*
(tmax(5)*0.8)))+(0.2*tmax(5)); %r/R = 0.4
tr1.LeadingEdge.Zero50 = (transpose(Tabela4Ponto10.Table{6,:}.*
(tmax(6)*0.8)))+(0.2*tmax(6)); %r/R = 0.5
tr1.LeadingEdge.Zero60 = (transpose(Tabela4Ponto10.Table{5,:}.*
(tmax(7)*0.8)))+(0.2*tmax(7)); %r/R = 0.6
tr1.LeadingEdge.Zero70 = (transpose(Tabela4Ponto10.Table{4,:}.*
(tmax(8)*0.8)))+(0.2*tmax(8)); %r/R = 0.7
tr1.LeadingEdge.Zero80 = (transpose(Tabela4Ponto10.Table{3,:}.*
(tmax(9)*0.8)))+(0.2*tmax(9)); %r/R = 0.8
tr1.LeadingEdge.Zero85 = (transpose(Tabela4Ponto10.Table{2,:}.*
(tmax(10)*0.8)))+(0.2*tmax(10)); %r/R = 0.85
tr1.LeadingEdge.Zero90 = (transpose(Tabela4Ponto10.Table{1,:}.*
(tmax(11)*0.8)))+(0.2*tmax(11)); %r/R = 0.9

% Geometry of the suction side or back of the blade is:  yface + tr

SuctionSide.TrailingEdge.Zero15 = yface.TrailingEdge.Zero15 +
tr1.TrailingEdge.Zero15;
SuctionSide.TrailingEdge.Zero20 = yface.TrailingEdge.Zero20 +
tr1.TrailingEdge.Zero20;
```

```
SuctionSide.TrailingEdge.Zero25 = yface.TrailingEdge.Zero25 +
tr1.TrailingEdge.Zero25;
SuctionSide.TrailingEdge.Zero30 = yface.TrailingEdge.Zero30 +
tr1.TrailingEdge.Zero30;
SuctionSide.TrailingEdge.Zero40 = yface.TrailingEdge.Zero40 +
tr1.TrailingEdge.Zero40;
SuctionSide.TrailingEdge.Zero50 = yface.TrailingEdge.Zero50 +
tr1.TrailingEdge.Zero50;
SuctionSide.TrailingEdge.Zero60 = yface.TrailingEdge.Zero60 +
tr1.TrailingEdge.Zero60;
SuctionSide.TrailingEdge.Zero70 = yface.TrailingEdge.Zero70 +
tr1.TrailingEdge.Zero70;
SuctionSide.TrailingEdge.Zero80 = tr1.TrailingEdge.Zero80;
SuctionSide.TrailingEdge.Zero85 = tr1.TrailingEdge.Zero85;
SuctionSide.TrailingEdge.Zero90 = tr1.TrailingEdge.Zero90;

SuctionSide.LeadingEdge.Zero15 = yface.LeadingEdge.Zero15 +
tr1.LeadingEdge.Zero15;
SuctionSide.LeadingEdge.Zero20 = yface.LeadingEdge.Zero20 +
tr1.LeadingEdge.Zero20;
SuctionSide.LeadingEdge.Zero25 = yface.LeadingEdge.Zero25 +
tr1.LeadingEdge.Zero25;
SuctionSide.LeadingEdge.Zero30 = yface.LeadingEdge.Zero30 +
tr1.LeadingEdge.Zero30;
SuctionSide.LeadingEdge.Zero40 = yface.LeadingEdge.Zero40 +
tr1.LeadingEdge.Zero40;
SuctionSide.LeadingEdge.Zero50 = yface.LeadingEdge.Zero50 +
tr1.LeadingEdge.Zero50;
SuctionSide.LeadingEdge.Zero60 = yface.LeadingEdge.Zero60 +
tr1.LeadingEdge.Zero60;
SuctionSide.LeadingEdge.Zero70 = yface.LeadingEdge.Zero70 +
tr1.LeadingEdge.Zero70;
SuctionSide.LeadingEdge.Zero80 = tr1.LeadingEdge.Zero80;
SuctionSide.LeadingEdge.Zero85 = tr1.LeadingEdge.Zero85;
SuctionSide.LeadingEdge.Zero90 = tr1.LeadingEdge.Zero90;

% Aplicando o caimento nas coordenadas Z

Rake.rR = [0.15; 0.2 ; 0.25; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.85; 0.9; 1];
Rake.Z = [tand(RakeValue)*(Rake.rR(1))*D/2; tand(RakeValue)*(Rake.rR(2))*D/2;...
    tand(RakeValue)*(Rake.rR(3))*D/2; ...
    tand(RakeValue)*(Rake.rR(4))*D/2; tand(RakeValue)*(Rake.rR(5))*D/2; ...
    tand(RakeValue)*(Rake.rR(6))*D/2; tand(RakeValue)*(Rake.rR(7))*D/2; ...
    tand(RakeValue)*(Rake.rR(8))*D/2; tand(RakeValue)*(Rake.rR(9))*D/2;...
    tand(RakeValue)*(Rake.rR(10))*D/2 ; tand(RakeValue)*(Rake.rR(11))*D/2;...
    tand(RakeValue)*(Rake.rR(12))*D/2];

yface.All.Zero15 = [yface.TrailingEdge.Zero15;
yface.LeadingEdge.Zero15]+Rake.Z(1);
```

```
yface.All.Zero20 = [yface.TrailingEdge.Zero20;
yface.LeadingEdge.Zero20]+Rake.Z(2);
yface.All.Zero25 = [yface.TrailingEdge.Zero25;
yface.LeadingEdge.Zero25]+Rake.Z(3);
yface.All.Zero30 = [yface.TrailingEdge.Zero30;
yface.LeadingEdge.Zero30]+Rake.Z(4);
yface.All.Zero40 = [yface.TrailingEdge.Zero40;
yface.LeadingEdge.Zero40]+Rake.Z(5);
yface.All.Zero50 = [yface.TrailingEdge.Zero50;
yface.LeadingEdge.Zero50]+Rake.Z(6);
yface.All.Zero60 = [yface.TrailingEdge.Zero60;
yface.LeadingEdge.Zero60]+Rake.Z(7);
yface.All.Zero70 = [yface.TrailingEdge.Zero70;
yface.LeadingEdge.Zero70]+Rake.Z(8);
yface.All.Zero80 = transpose(zeros(1,19)+Rake.Z(9));
yface.All.Zero85 = transpose(zeros(1,19)+Rake.Z(10));
yface.All.Zero90 = transpose(zeros(1,19)+Rake.Z(11));

SuctionSide.All.Zero15 =
[SuctionSide.TrailingEdge.Zero15;SuctionSide.LeadingEdge.Zero15]+Rake.Z(1);
SuctionSide.All.Zero20 =
[SuctionSide.TrailingEdge.Zero20;SuctionSide.LeadingEdge.Zero20]+Rake.Z(2);
SuctionSide.All.Zero25 =
[SuctionSide.TrailingEdge.Zero25;SuctionSide.LeadingEdge.Zero25]+Rake.Z(3);
SuctionSide.All.Zero30 =
[SuctionSide.TrailingEdge.Zero30;SuctionSide.LeadingEdge.Zero30]+Rake.Z(4);
SuctionSide.All.Zero40 =
[SuctionSide.TrailingEdge.Zero40;SuctionSide.LeadingEdge.Zero40]+Rake.Z(5);
SuctionSide.All.Zero50 =
[SuctionSide.TrailingEdge.Zero50;SuctionSide.LeadingEdge.Zero50]+Rake.Z(6);
SuctionSide.All.Zero60 =
[SuctionSide.TrailingEdge.Zero60;SuctionSide.LeadingEdge.Zero60]+Rake.Z(7);
SuctionSide.All.Zero70 =
[SuctionSide.TrailingEdge.Zero70;SuctionSide.LeadingEdge.Zero70]+Rake.Z(8);
SuctionSide.All.Zero80 =
[SuctionSide.TrailingEdge.Zero80;SuctionSide.LeadingEdge.Zero80]+Rake.Z(9);
SuctionSide.All.Zero85 =
[SuctionSide.TrailingEdge.Zero85;SuctionSide.LeadingEdge.Zero85]+Rake.Z(10);
SuctionSide.All.Zero90 =
[SuctionSide.TrailingEdge.Zero90;SuctionSide.LeadingEdge.Zero90]+Rake.Z(11);

%% Coordenadas X

% TrailingEdge (Bordo de Fuga considerado negativo por estar a esquerda do
% ponto de maior espessura, considerado como o 0%).

% Vetores TrailingEdge e LeadingEdge são referentes as porcentagens vindas
% das tabelas 4.7 a 4.10.

TrailingEdge = [-100; -95; -90; -80; -70; -60; -50; -40; -20];
```

```
LeadingEdge = [20; 40; 50; 60; 70; 80; 85; 90; 95; 100];
XPercent = [TrailingEdge ; LeadingEdge];

XCoordinate.TrailingEdge.Zero15 = cr(1)*(1-
Tabela4Ponto11.xtmaxcr(1,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero20 = cr(2)*(1-
Tabela4Ponto11.xtmaxcr(2,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero25 = cr(3)*(1-
Tabela4Ponto11.xtmaxcr(3,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero30 = cr(4)*(1-
Tabela4Ponto11.xtmaxcr(4,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero40 = cr(5)*(1-
Tabela4Ponto11.xtmaxcr(5,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero50 = cr(6)*(1-
Tabela4Ponto11.xtmaxcr(6,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero60 = cr(7)*(1-
Tabela4Ponto11.xtmaxcr(7,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero70 = cr(8)*(1-
Tabela4Ponto11.xtmaxcr(8,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero80 = cr(9)*(1-
Tabela4Ponto11.xtmaxcr(9,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero85 = cr(10)*(1-
Tabela4Ponto11.xtmaxcr(10,1)).*(TrailingEdge/100);
XCoordinate.TrailingEdge.Zero90 = cr(11)*(1-
Tabela4Ponto11.xtmaxcr(11,1)).*(TrailingEdge/100);

XCoordinate.LeadingEdge.Zero15 =
cr(1)*Tabela4Ponto11.xtmaxcr(1,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero20 =
cr(2)*Tabela4Ponto11.xtmaxcr(2,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero25 =
cr(3)*Tabela4Ponto11.xtmaxcr(3,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero30 =
cr(4)*Tabela4Ponto11.xtmaxcr(4,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero40 =
cr(5)*Tabela4Ponto11.xtmaxcr(5,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero50 =
cr(6)*Tabela4Ponto11.xtmaxcr(6,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero60 =
cr(7)*Tabela4Ponto11.xtmaxcr(7,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero70 =
cr(8)*Tabela4Ponto11.xtmaxcr(8,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero80 =
cr(9)*Tabela4Ponto11.xtmaxcr(9,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero85 =
cr(10)*Tabela4Ponto11.xtmaxcr(10,1).*(LeadingEdge/100);
XCoordinate.LeadingEdge.Zero90 =
cr(11)*Tabela4Ponto11.xtmaxcr(11,1).*(LeadingEdge/100);
```

```
XCoordinate.Both.Zero15 = [XCoordinate.TrailingEdge.Zero15 ;
XCoordinate.LeadingEdge.Zero15];
XCoordinate.Both.Zero20 = [XCoordinate.TrailingEdge.Zero20 ;
XCoordinate.LeadingEdge.Zero20];
XCoordinate.Both.Zero25 = [XCoordinate.TrailingEdge.Zero25 ;
XCoordinate.LeadingEdge.Zero25];
XCoordinate.Both.Zero30 = [XCoordinate.TrailingEdge.Zero30 ;
XCoordinate.LeadingEdge.Zero30];
XCoordinate.Both.Zero40 = [XCoordinate.TrailingEdge.Zero40 ;
XCoordinate.LeadingEdge.Zero40];
XCoordinate.Both.Zero50 = [XCoordinate.TrailingEdge.Zero50 ;
XCoordinate.LeadingEdge.Zero50];
XCoordinate.Both.Zero60 = [XCoordinate.TrailingEdge.Zero60 ;
XCoordinate.LeadingEdge.Zero60];
XCoordinate.Both.Zero70 = [XCoordinate.TrailingEdge.Zero70 ;
XCoordinate.LeadingEdge.Zero70];
XCoordinate.Both.Zero80 = [XCoordinate.TrailingEdge.Zero80 ;
XCoordinate.LeadingEdge.Zero80];
XCoordinate.Both.Zero85 = [XCoordinate.TrailingEdge.Zero85 ;
XCoordinate.LeadingEdge.Zero85];
XCoordinate.Both.Zero90 = [XCoordinate.TrailingEdge.Zero90 ;
XCoordinate.LeadingEdge.Zero90];

XCoordinateSkewSuave.BothSkewed.Zero15 = XCoordinate.Both.Zero15 -
SkewSuave(1);
XCoordinateSkewSuave.BothSkewed.Zero20 = XCoordinate.Both.Zero20 -
SkewSuave(2);
XCoordinateSkewSuave.BothSkewed.Zero25 = XCoordinate.Both.Zero25 -
SkewSuave(3);
XCoordinateSkewSuave.BothSkewed.Zero30 = XCoordinate.Both.Zero30 -
SkewSuave(4);
XCoordinateSkewSuave.BothSkewed.Zero40 = XCoordinate.Both.Zero40 -
SkewSuave(5);
XCoordinateSkewSuave.BothSkewed.Zero50 = XCoordinate.Both.Zero50 -
SkewSuave(6);
XCoordinateSkewSuave.BothSkewed.Zero60 = XCoordinate.Both.Zero60 -
SkewSuave(7);
XCoordinateSkewSuave.BothSkewed.Zero70 = XCoordinate.Both.Zero70 -
SkewSuave(8);
XCoordinateSkewSuave.BothSkewed.Zero80 = XCoordinate.Both.Zero80 -
SkewSuave(9);
XCoordinateSkewSuave.BothSkewed.Zero85 = XCoordinate.Both.Zero85 -
SkewSuave(10);
XCoordinateSkewSuave.BothSkewed.Zero90 = XCoordinate.Both.Zero90 -
SkewSuave(11);

%% COORDENADAS Y

Y = [0.15; 0.2 ; 0.25; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.85; 0.9]*(D/2);
```

```
YPlot.Zero15 = zeros(1,19) + Y(1);
YPlot.Zero20 = zeros(1,19) + Y(2);
YPlot.Zero25 = zeros(1,19) + Y(3);
YPlot.Zero30 = zeros(1,19) + Y(4);
YPlot.Zero40 = zeros(1,19) + Y(5);
YPlot.Zero50 = zeros(1,19) + Y(6);
YPlot.Zero60 = zeros(1,19) + Y(7);
YPlot.Zero70 = zeros(1,19) + Y(8);
YPlot.Zero80 = zeros(1,19) + Y(9);
YPlot.Zero85 = zeros(1,19) + Y(10);
YPlot.Zero90 = zeros(1,19) + Y(11);

%% PLOTS

axes(handles.axes2)
    plot(XCoordinateSkewSuave.BothSkewed.Zero15,yface.All.Zero15,'r')
    hold on
    plot(XCoordinateSkewSuave.BothSkewed.Zero15,SuctionSide.All.Zero15,'b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero20,yface.All.Zero20,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero20,SuctionSide.All.Zero20,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero25,yface.All.Zero25,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero25,SuctionSide.All.Zero25,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero30,yface.All.Zero30,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero30,SuctionSide.All.Zero30,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero40,yface.All.Zero40,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero40,SuctionSide.All.Zero40,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero50,yface.All.Zero50,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero50,SuctionSide.All.Zero50,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero60,yface.All.Zero60,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero60,SuctionSide.All.Zero60,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero70,yface.All.Zero70,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero70,SuctionSide.All.Zero70,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero80,yface.All.Zero80,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero80,SuctionSide.All.Zero80,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero85,yface.All.Zero85,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero85,SuctionSide.All.Zero85,'Color','b')
    plot(XCoordinateSkewSuave.BothSkewed.Zero90,yface.All.Zero90,'Color','r')

plot(XCoordinateSkewSuave.BothSkewed.Zero90,SuctionSide.All.Zero90,'Color','b')
    title('r/R = 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9')
    xlabel ('Eixo X')
```

```
    ylabel ('Eixo Z')

%% SUAVIZAÇÃO DAS SEÇÕES (ELEMENTOS EM Z)
Subdivisoes = 41;
X = XCoordinateSkewSuave.BothSkewed.Zero15;
Y = yface.All.Zero15;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero15_2 =
transpose(linspace(min(X),max(X),Subdivisoes));
yface.All.Zero15_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero15_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero15;
Y = SuctionSide.All.Zero15;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero15_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero15_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero15_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero20 ;
Y = yface.All.Zero20;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero20_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero20_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero20_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero20;
Y = SuctionSide.All.Zero20;
% Fit: 'untitled fit 1'.
```

```
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero20_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero20_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero20_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero25;
Y = yface.All.Zero25;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero25_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero25_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero25_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero25;
Y = SuctionSide.All.Zero25;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero25_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero25_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero25_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero30;
Y = yface.All.Zero30;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
```

```
XCoordinateSkewSuave.BothSkewed.Zero30_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero30_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero30_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero30;
Y = SuctionSide.All.Zero30;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero30_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero30_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero30_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero40;
Y = yface.All.Zero40;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero40_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero40_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero40_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero40;
Y = SuctionSide.All.Zero40;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero40_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero40_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero40_2);
clear X
clear Y
```

```
X = XCoordinateSkewSuave.BothSkewed.Zero50;
Y = yface.All.Zero50;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero50_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero50_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero50_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero50;
Y = SuctionSide.All.Zero50;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero50_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero50_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero50_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero60;
Y = yface.All.Zero60;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
opts = fitoptions( 'Method', 'SmoothingSpline' );
opts.SmoothingParam = 1;
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft, opts );
XCoordinateSkewSuave.BothSkewed.Zero60_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero60_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero60_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero60;
Y = SuctionSide.All.Zero60;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
```

```
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero60_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero60_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero60_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero70;
Y = yface.All.Zero70;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero70_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero70_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero70_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero70;
Y = SuctionSide.All.Zero70;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero70_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero70_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero70_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero80;
Y = yface.All.Zero80;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero80_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero80_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero80_2);
```

```
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero80;
Y = SuctionSide.All.Zero80;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero80_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero80_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero80_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero85;
Y = yface.All.Zero85;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero85_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero85_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero85_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero85;
Y = SuctionSide.All.Zero85;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero85_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero85_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero85_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero90;
Y = yface.All.Zero90;
% Fit: 'untitled fit 1'.
```

```
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero90_2 =
linspace(min(X),max(X),Subdivisoes);
yface.All.Zero90_Suave = fitresult(XCoordinateSkewSuave.BothSkewed.Zero90_2);
clear X
clear Y

X = XCoordinateSkewSuave.BothSkewed.Zero90;
Y = SuctionSide.All.Zero90;
% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
XCoordinateSkewSuave.BothSkewed.Zero90_2 =
linspace(min(X),max(X),Subdivisoes);
SuctionSide.All.Zero90_Suave =
fitresult(XCoordinateSkewSuave.BothSkewed.Zero90_2);
clear X
clear Y
clear ft
clear fitresult
clear xData
clear yData
clear opts

Y = [0.15; 0.2 ; 0.25; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.85; 0.9]*(D/2);
YPlot.Zero15_2 = zeros(1,Subdivisoes) + Y(1);
YPlot.Zero20_2 = zeros(1,Subdivisoes) + Y(2);
YPlot.Zero25_2 = zeros(1,Subdivisoes) + Y(3);
YPlot.Zero30_2 = zeros(1,Subdivisoes) + Y(4);
YPlot.Zero40_2 = zeros(1,Subdivisoes) + Y(5);
YPlot.Zero50_2 = zeros(1,Subdivisoes) + Y(6);
YPlot.Zero60_2 = zeros(1,Subdivisoes) + Y(7);
YPlot.Zero70_2 = zeros(1,Subdivisoes) + Y(8);
YPlot.Zero80_2 = zeros(1,Subdivisoes) + Y(9);
YPlot.Zero85_2 = zeros(1,Subdivisoes) + Y(10);
YPlot.Zero90_2 = zeros(1,Subdivisoes) + Y(11);

%% Criação de Listas "Verticais"
% Anteriomente as listas possuiam os elementos correspondentes a uma seção
% planificada.
% Nessa etapa as listas possuem um elemento de cada seção, isto é, o Y de
% cada elemento na lista é diferente, são 19 listas, numero de
% elementos que formam uma seção, contendo 11 elementos cada lista, um
```

```
% elemento para cada r/R.

%Com Skew e Rake
for i = 1:length(SuctionSide.All.Zero20_Suave)
    ZVerticalSuctionSide_Suave(:,i) =
[SuctionSide.All.Zero15_Suave(i);SuctionSide.All.Zero20_Suave(i);...
        SuctionSide.All.Zero25_Suave(i);SuctionSide.All.Zero30_Suave(i);...
        SuctionSide.All.Zero40_Suave(i);SuctionSide.All.Zero50_Suave(i);...
        SuctionSide.All.Zero60_Suave(i);SuctionSide.All.Zero70_Suave(i);...
        SuctionSide.All.Zero80_Suave(i);SuctionSide.All.Zero85_Suave(i);...
        SuctionSide.All.Zero90_Suave(i)];

    ZVerticalyface_Suave(:,i) =
[yface.All.Zero15_Suave(i);yface.All.Zero20_Suave(i);...
        yface.All.Zero25_Suave(i);yface.All.Zero30_Suave(i);...
        yface.All.Zero40_Suave(i);yface.All.Zero50_Suave(i);...
        yface.All.Zero60_Suave(i);yface.All.Zero70_Suave(i);...
        yface.All.Zero80_Suave(i);yface.All.Zero85_Suave(i);...
        yface.All.Zero90_Suave(i)];

    YVertical_Suave(:,i) = [YPlot.Zero15_2(i);YPlot.Zero20_2(i); YPlot.Zero25_2(i);...
        YPlot.Zero30_2(i);YPlot.Zero40_2(i); YPlot.Zero50_2(i);YPlot.Zero60_2(i);...
        YPlot.Zero70_2(i);YPlot.Zero80_2(i);YPlot.Zero85_2(i); YPlot.Zero90_2(i)];

    XVertical_Suave(:,i) =
[XCoordinateSkewSuave.BothSkewed.Zero15_2(i);XCoordinateSkewSuave.BothSke
wed.Zero20_2(i);...

XCoordinateSkewSuave.BothSkewed.Zero25_2(i);XCoordinateSkewSuave.BothSke
wed.Zero30_2(i);...

XCoordinateSkewSuave.BothSkewed.Zero40_2(i);XCoordinateSkewSuave.BothSke
wed.Zero50_2(i);...

XCoordinateSkewSuave.BothSkewed.Zero60_2(i);XCoordinateSkewSuave.BothSke
wed.Zero70_2(i);...

XCoordinateSkewSuave.BothSkewed.Zero80_2(i);XCoordinateSkewSuave.BothSke
wed.Zero85_2(i);...
        XCoordinateSkewSuave.BothSkewed.Zero90_2(i)];
end

%% CRIAÇÃO DE NOVAS SEÇÕES (r/R = 0.35; 0.45; 0.55; 0.65; 0.75)

for i=1:length(SuctionSide.All.Zero20_Suave);
    XVertical2 = XVertical_Suave(:,i);
    YVertical2 = YVertical_Suave(:,i);
    ZVerticalyface2 = ZVerticalyface_Suave(:,i);
    ZVerticalSuction = ZVerticalSuctionSide_Suave(:,i);
    % Fit: 'X(Y)'.
```

```
    [xData, yData] = prepareCurveData( YVertical2, XVertical2 );
    % Set up fittype and options.
    ft = fittype( 'smoothingspline' );
    % Fit model to data.
    [fitresult, gof] = fit( xData, yData, ft, 'Normalize', 'on' );
    X1 = fitresult((D/2)*0.35);
    X2 = fitresult((D/2)*0.45);
    X3 = fitresult((D/2)*0.55);
    X4 = fitresult((D/2)*0.65);
    X5 = fitresult((D/2)*0.75);
    X = [X1;X2;X3;X4;X5];
    Y = [(D/2)*0.35; (D/2)*0.45; (D/2)*0.55; (D/2)*0.65; (D/2)*0.75];

    % Fit: 'Z(X,Y)'.
    [xData, yData, zData] = prepareSurfaceData( XVertical2, YVertical2,
ZVerticalyface2 );
    % Set up fittype and options.
    ft = 'thinplateinterp';
    % Fit model to data.
    [fitresult, gof] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );
    Z1 = fitresult(X1,Y(1));
    Z2 = fitresult(X2,Y(2));
    Z3 = fitresult(X3,Y(3));
    Z4 = fitresult(X4,Y(4));
    Z5 = fitresult(X1,Y(5));
    Z = [Z1;Z2;Z3;Z4;Z5];

    % Fit: 'Z(X,Y)'.
    [xData, yData, zData] = prepareSurfaceData( XVertical2, YVertical2,
ZVerticalSuction );
    % Set up fittype and options.
    ft = 'thinplateinterp';
    % Fit model to data.
    [fitresult, gof] = fit( [xData, yData], zData, ft, 'Normalize', 'on' );
    Z1_Suction = fitresult(X1,Y(1));
    Z2_Suction = fitresult(X2,Y(2));
    Z3_Suction = fitresult(X3,Y(3));
    Z4_Suction = fitresult(X4,Y(4));
    Z5_Suction = fitresult(X1,Y(5));
    Z_Suction = [Z1_Suction;Z2_Suction;Z3_Suction;Z4_Suction;...
        Z5_Suction];

     axes(handles.axes1)
        scatter3(XVertical2, YVertical2,ZVerticalyface2,'MarkerEdgeColor','r')
        hold on
        scatter3(XVertical2, YVertical2,ZVerticalSuction,'MarkerEdgeColor','b')
        hold on
        scatter3(X,Y,Z,'MarkerEdgeColor','r')
        hold on
        scatter3(X,Y,Z_Suction,'MarkerEdgeColor','b')
```

```
        xlabel ('Eixo X')
        ylabel ('Eixo Y')
        zlabel ('Eixo Z')

    for j= 1:length(X)
     PontosX(j,i) = X(j);
     PontosY(j,i) = Y(j);
     PontosZ(j,i) = Z(j);
     PontosZ_Suction(j,i) = Z_Suction(j);
    end

end
for i = 1:length(SuctionSide.All.Zero20_Suave)
    PontosExtrasZero35_X(i,1) = PontosX(1,i);
    PontosExtrasZero45_X(i,1) = PontosX(2,i);
    PontosExtrasZero55_X(i,1) = PontosX(3,i);
    PontosExtrasZero65_X(i,1) = PontosX(4,i);
    PontosExtrasZero75_X(i,1) = PontosX(5,i);
    PontosExtrasZero35_Y(i) = PontosY(1,i);
    PontosExtrasZero45_Y(i) = PontosY(2,i);
    PontosExtrasZero55_Y(i) = PontosY(3,i);
    PontosExtrasZero65_Y(i) = PontosY(4,i);
    PontosExtrasZero75_Y(i) = PontosY(5,i);
    PontosExtrasZero35_Z(i,1) = PontosZ(1,i);
    PontosExtrasZero45_Z(i,1) = PontosZ(2,i);
    PontosExtrasZero55_Z(i,1) = PontosZ(3,i);
    PontosExtrasZero65_Z(i,1) = PontosZ(4,i);
    PontosExtrasZero75_Z(i,1) = PontosZ(5,i);
    PontosExtrasZero35_Z_Suction(i,1) = PontosZ_Suction(1,i);
    PontosExtrasZero45_Z_Suction(i,1) = PontosZ_Suction(2,i);
    PontosExtrasZero55_Z_Suction(i,1) = PontosZ_Suction(3,i);
    PontosExtrasZero65_Z_Suction(i,1) = PontosZ_Suction(4,i);
    PontosExtrasZero75_Z_Suction(i,1) = PontosZ_Suction(5,i);
end

clear Size
clear PontosX
clear PontosY
clear PontosZ
clear PontosZ_Suction

%% Contorno da Pá

% Interpolação: Espessura r/R = 0.95 e 0.975
X = Tabela4Ponto12.radiusInterpolado;
% Fit: 'Distribuição Espessuras'.
[xData, yData] = prepareCurveData( X, tr_Interpolado );
% Set up fittype and options.
ft = fittype( {'(sin(x-pi))', '((x-10)^2)', '1'}, 'independent', 'x', 'dependent', 'y',
'coefficients', {'a', 'b', 'c'} );
```

```
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
Tip.Espessura95 = fitresult(0.95);
Tip.Espessura975 = fitresult(0.975);

clear X
clear xData
clear yData
clear ft
clear fitresult


% Pontos Tip (r/R = 1.0)
Tip.rR100 = [0-Tip.Skew100, (D/2), Rake.Z(12); 0-Tip.Skew100,...
    (D/2), 0+tr_Interpolado(12)+Rake.Z(12)];

% Pontos Trailing Edge(Index=1)
PontosTrailingPressao =
[XVertical_Suave(:,1),YVertical_Suave(:,1),ZVerticalyface_Suave(:,1)];
PontosTrailingSuccao =
[XVertical_Suave(:,1),YVertical_Suave(:,1),ZVerticalSuctionSide_Suave(:,1)];

% Extrapolando valores de X para os r/R acima de 0.9 (Trailing Edge)
X = YVertical_Suave(:,1);
Y = XVertical_Suave(:,1);
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'poly7' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
Tip.TE_XZero95 = fitresult(0.95*(D/2));
Tip.TE_XZero975 = fitresult(0.975*(D/2));

clear X
clear Y
clear xData
clear yData
clear ft
clear fitresult

%Distribuição de Espessuras Trailing Edge
UltimoTE = length(tr1.TrailingEdge.Zero15);
DistribuicaoEspessurasTE = [tr1.TrailingEdge.Zero15(UltimoTE);
tr1.TrailingEdge.Zero20(UltimoTE);...
    tr1.TrailingEdge.Zero25(UltimoTE); tr1.TrailingEdge.Zero30(UltimoTE);...
    tr1.TrailingEdge.Zero40(UltimoTE); tr1.TrailingEdge.Zero50(UltimoTE);...
    tr1.TrailingEdge.Zero60(UltimoTE); tr1.TrailingEdge.Zero70(UltimoTE);...
    tr1.TrailingEdge.Zero80(UltimoTE); tr1.TrailingEdge.Zero85(UltimoTE);...
    tr1.TrailingEdge.Zero90(UltimoTE)];
rR = [0.15, 0.20, 0.25, 0.30, 0.40, 0.5, 0.6, 0.7, 0.8, 0.85, 0.9];
```

```
%Extrapolação Espessuras r/R = 0.95 e 0.975
[xData, yData] = prepareCurveData( rR, DistribuicaoEspessurasTE );
% Set up fittype and options.
ft = fittype( 'smoothingspline' );
opts = fitoptions( 'Method', 'SmoothingSpline' );
opts.SmoothingParam = 1;
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft, opts );
Tip.Espessura95_TE = fitresult(0.95);
Tip.Espessura975_TE = fitresult(0.975);

clear rR
clear UltimoTE
clear DistribuicaoEspessurasTE
clear xData
clear yData
clear ft
clear fitresult
clear opts

Tip.rR95_TE = [Tip.TE_XZero95, 0.95*(D/2), 0+tand(RakeValue)*(0.95*D/2);...
    Tip.TE_XZero95, 0.95*(D/2),
(Tip.Espessura95_TE*0.2)+tand(RakeValue)*(0.95*D/2)];
Tip.rR975_TE = [Tip.TE_XZero975, 0.975*(D/2), 0+tand(RakeValue)*(0.975*D/2);...
    Tip.TE_XZero975, 0.975*(D/2),
(Tip.Espessura975_TE*0.2)+tand(RakeValue)*(0.975*D/2)];

% Pontos Leading Edge(Index=19)
PontosLeadingPressao =
[XVertical_Suave(:,length(SuctionSide.All.Zero20_Suave)),YVertical_Suave(:,length(
SuctionSide.All.Zero20_Suave)),ZVerticalyface_Suave(:,length(SuctionSide.All.Zero
20_Suave))];
PontosLeadingSuccao =
[XVertical_Suave(:,length(SuctionSide.All.Zero20_Suave)),YVertical_Suave(:,length(
SuctionSide.All.Zero20_Suave)),ZVerticalSuctionSide_Suave(:,length(SuctionSide.Al
l.Zero20_Suave))];

% Extrapolando valores de X para os r/R acima de 0.9 (Leading Edge)
X = YVertical_Suave(:,length(SuctionSide.All.Zero20_Suave));
Y = XVertical_Suave(:,length(SuctionSide.All.Zero20_Suave));
[xData, yData] = prepareCurveData( X, Y );
% Set up fittype and options.
ft = fittype( 'poly5' );
% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft );
Tip.LE_XZero95 = fitresult(0.95*(D/2));
Tip.LE_XZero975 = fitresult(0.975*(D/2));

clear X
```

```
clear Y
clear xData
clear yData
clear ft
clear fitresult

Tip.rR95_LE = [Tip.LE_XZero95, 0.95*(D/2), 0+tand(RakeValue)*(0.95*D/2);...
    Tip.LE_XZero95, 0.95*(D/2), (Tip.Espessura95*0.2)+tand(RakeValue)*(0.95*D/2)];
Tip.rR975_LE = [Tip.LE_XZero975, 0.975*(D/2), 0+tand(RakeValue)*(0.975*D/2);...
    Tip.LE_XZero975, 0.975*(D/2),
(Tip.Espessura975*0.2)+tand(RakeValue)*(0.975*D/2)];

PontosPressao = [PontosTrailingPressao; Tip.rR100(1,:); PontosLeadingPressao];
PontosSuccao = [PontosTrailingSuccao; Tip.rR100(2,:); PontosLeadingSuccao];

clear PontosLeadingPressao
clear PontosLeadingSuccao
clear PontosTrailingPressao
clear PontosTrailingSuccao

%% Calculo dos angulos de passo

% Sem r/R = 0.75 e 0.85
rR = [0.15, 0.2, 0.25, 0.30, 0.35, 0.4, 0.45, 0.5, 0.55,  0.6, 0.65,...
     0.7, 0.8, 0.9, 0.95, 0.975 1];

 % Com r/R = 0.75
Zero75 = get(handles.chb_Zero75,'Value');
if Zero75 == 1
    rR = [0.15, 0.2, 0.25, 0.30, 0.35, 0.4, 0.45, 0.5, 0.55,  0.6, 0.65,...
        0.7, 0.75, 0.8, 0.9, 0.95, 0.975 1];
end

% Com r/R = 0.85
Zero85 = get(handles.chb_Zero85,'Value');
if Zero85 == 1
    rR = [0.15, 0.2, 0.25, 0.30, 0.35, 0.4, 0.45, 0.5, 0.55,  0.6, 0.65,...
        0.7, 0.8,0.85, 0.9, 0.95, 0.975 1];
end

%Com r/R = 0.75 e 0.85
if Zero75 == 1 && Zero85 ==1
     rR = [0.15, 0.2, 0.25, 0.30, 0.35, 0.4, 0.45, 0.5, 0.55,  0.6, 0.65,...
         0.7, 0.75,0.8,0.85, 0.9, 0.95, 0.975 1];
end

P = PD*D;

X = [0.2; 0.6];
Y = [0.822; 1];
```

```
[xData, yData] = prepareCurveData( X, Y );
ft = fittype( 'poly1' );
[fitresult, gof] = fit( xData, yData, ft );

if Z ==4
  for i = 1:length(rR)
     Nada(i) = 0;
     if rR(i) >= 0.5
        Teta(i) = atand(P/(2*pi*rR(i)*D/2));
     else
        Teta(i) = atand(P*fitresult(rR(i))/(2*pi*rR(i)*D/2));
     end
  end
else
   for i = 1:length(rR)
      Teta(i) = atand(P/(2*pi*rR(i)*D/2));
      Nada(i) = 0;
   end
end

Passo = transpose([rR; Teta; Nada]);

clear X
clear Y
clear Teta
clear Nada
clear rR
clear xData
clear yData
clear ft
clear fitresult
clear gof


%% Parametros com PONTOS EXTRAS + SUAVIZAÇÃO DAS SECOES

%Sem 0.75 e 0.85
X1Parameters_41Extras =
[transpose(XCoordinateSkewSuave.BothSkewed.Zero15_2);...
   transpose(XCoordinateSkewSuave.BothSkewed.Zero20_2);...
   transpose(XCoordinateSkewSuave.BothSkewed.Zero25_2);...
   transpose(XCoordinateSkewSuave.BothSkewed.Zero30_2);
PontosExtrasZero35_X;...
   transpose(XCoordinateSkewSuave.BothSkewed.Zero40_2);
PontosExtrasZero45_X;...
   transpose(XCoordinateSkewSuave.BothSkewed.Zero50_2);
PontosExtrasZero55_X;...
   transpose(XCoordinateSkewSuave.BothSkewed.Zero60_2);
PontosExtrasZero65_X;...
```

```
    transpose(XCoordinateSkewSuave.BothSkewed.Zero70_2);
    transpose(XCoordinateSkewSuave.BothSkewed.Zero80_2);...
    transpose(XCoordinateSkewSuave.BothSkewed.Zero90_2)];

Z1Parameters_41Extras = [yface.All.Zero15_Suave; yface.All.Zero20_Suave;
yface.All.Zero25_Suave;...
    yface.All.Zero30_Suave; PontosExtrasZero35_Z;...
    yface.All.Zero40_Suave; PontosExtrasZero45_Z;...
    yface.All.Zero50_Suave; PontosExtrasZero55_Z;...
    yface.All.Zero60_Suave; PontosExtrasZero65_Z;...
    yface.All.Zero70_Suave;
    yface.All.Zero80_Suave; yface.All.Zero90_Suave];

Z2Paramaters_41Extras = [SuctionSide.All.Zero15_Suave;
SuctionSide.All.Zero20_Suave; ...
    SuctionSide.All.Zero25_Suave;
    SuctionSide.All.Zero30_Suave; PontosExtrasZero35_Z_Suction;...
    SuctionSide.All.Zero40_Suave; PontosExtrasZero45_Z_Suction;...
    SuctionSide.All.Zero50_Suave; PontosExtrasZero55_Z_Suction;...
    SuctionSide.All.Zero60_Suave; PontosExtrasZero65_Z_Suction;...
    SuctionSide.All.Zero70_Suave;
    SuctionSide.All.Zero80_Suave; SuctionSide.All.Zero90_Suave];

Y1Parameters_41Extras =
transpose([YPlot.Zero15_2,YPlot.Zero20_2,YPlot.Zero25_2,...
    YPlot.Zero30_2, PontosExtrasZero35_Y,...
    YPlot.Zero40_2, PontosExtrasZero45_Y,...
    YPlot.Zero50_2, PontosExtrasZero55_Y,...
    YPlot.Zero60_2, PontosExtrasZero65_Y,...
    YPlot.Zero70_2,...
    YPlot.Zero80_2,YPlot.Zero90_2]);

% Com 0.75
if Zero75  == 1
    X1Parameters_41Extras =
[transpose(XCoordinateSkewSuave.BothSkewed.Zero15_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero20_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero25_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero30_2);
PontosExtrasZero35_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero40_2);
PontosExtrasZero45_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero50_2);
PontosExtrasZero55_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero60_2);
PontosExtrasZero65_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero70_2);
PontosExtrasZero75_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero80_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero90_2)];
```

```
    Z1Parameters_41Extras = [yface.All.Zero15_Suave; yface.All.Zero20_Suave;
yface.All.Zero25_Suave;...
        yface.All.Zero30_Suave; PontosExtrasZero35_Z;...
        yface.All.Zero40_Suave; PontosExtrasZero45_Z;...
        yface.All.Zero50_Suave; PontosExtrasZero55_Z;...
        yface.All.Zero60_Suave; PontosExtrasZero65_Z;...
        yface.All.Zero70_Suave; PontosExtrasZero75_Z;...
        yface.All.Zero80_Suave; yface.All.Zero90_Suave];

    Z2Paramaters_41Extras = [SuctionSide.All.Zero15_Suave;
SuctionSide.All.Zero20_Suave; ...
        SuctionSide.All.Zero25_Suave;
        SuctionSide.All.Zero30_Suave; PontosExtrasZero35_Z_Suction;...
        SuctionSide.All.Zero40_Suave; PontosExtrasZero45_Z_Suction;...
        SuctionSide.All.Zero50_Suave; PontosExtrasZero55_Z_Suction;...
        SuctionSide.All.Zero60_Suave; PontosExtrasZero65_Z_Suction;...
        SuctionSide.All.Zero70_Suave; PontosExtrasZero75_Z_Suction;...
        SuctionSide.All.Zero80_Suave; SuctionSide.All.Zero90_Suave];

    Y1Parameters_41Extras =
transpose([YPlot.Zero15_2,YPlot.Zero20_2,YPlot.Zero25_2,...
        YPlot.Zero30_2, PontosExtrasZero35_Y,...
        YPlot.Zero40_2, PontosExtrasZero45_Y,...
        YPlot.Zero50_2, PontosExtrasZero55_Y,...
        YPlot.Zero60_2, PontosExtrasZero65_Y,...
        YPlot.Zero70_2, PontosExtrasZero75_Y,...
        YPlot.Zero80_2,YPlot.Zero90_2]);
end

% Com 0.85
if Zero85 == 1
    X1Parameters_41Extras =
[transpose(XCoordinateSkewSuave.BothSkewed.Zero15_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero20_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero25_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero30_2);
PontosExtrasZero35_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero40_2);
PontosExtrasZero45_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero50_2);
PontosExtrasZero55_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero60_2);
PontosExtrasZero65_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero70_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero80_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero85_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero90_2)];
```

```
    Z1Parameters_41Extras = [yface.All.Zero15_Suave; yface.All.Zero20_Suave;
yface.All.Zero25_Suave;...
        yface.All.Zero30_Suave; PontosExtrasZero35_Z;...
        yface.All.Zero40_Suave; PontosExtrasZero45_Z;...
        yface.All.Zero50_Suave; PontosExtrasZero55_Z;...
        yface.All.Zero60_Suave; PontosExtrasZero65_Z;...
        yface.All.Zero70_Suave;
        yface.All.Zero80_Suave; yface.All.Zero85_Suave;
        yface.All.Zero90_Suave];

    Z2Paramaters_41Extras = [SuctionSide.All.Zero15_Suave;
SuctionSide.All.Zero20_Suave; ...
        SuctionSide.All.Zero25_Suave;
        SuctionSide.All.Zero30_Suave; PontosExtrasZero35_Z_Suction;...
        SuctionSide.All.Zero40_Suave; PontosExtrasZero45_Z_Suction;...
        SuctionSide.All.Zero50_Suave; PontosExtrasZero55_Z_Suction;...
        SuctionSide.All.Zero60_Suave; PontosExtrasZero65_Z_Suction;...
        SuctionSide.All.Zero70_Suave;
        SuctionSide.All.Zero80_Suave; SuctionSide.All.Zero85_Suave;
        SuctionSide.All.Zero90_Suave];

    Y1Parameters_41Extras =
transpose([YPlot.Zero15_2,YPlot.Zero20_2,YPlot.Zero25_2,...
        YPlot.Zero30_2, PontosExtrasZero35_Y,...
        YPlot.Zero40_2, PontosExtrasZero45_Y,...
        YPlot.Zero50_2, PontosExtrasZero55_Y,...
        YPlot.Zero60_2, PontosExtrasZero65_Y,...
        YPlot.Zero70_2,...
        YPlot.Zero80_2,YPlot.Zero85_2,YPlot.Zero90_2]);
end

if Zero85 == 1 && Zero75 ==1
    X1Parameters_41Extras =
[transpose(XCoordinateSkewSuave.BothSkewed.Zero15_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero20_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero25_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero30_2);
PontosExtrasZero35_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero40_2);
PontosExtrasZero45_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero50_2);
PontosExtrasZero55_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero60_2);
PontosExtrasZero65_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero70_2);
PontosExtrasZero75_X;...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero80_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero85_2);...
        transpose(XCoordinateSkewSuave.BothSkewed.Zero90_2)];
```

```
    Z1Parameters_41Extras = [yface.All.Zero15_Suave; yface.All.Zero20_Suave;
yface.All.Zero25_Suave;...
        yface.All.Zero30_Suave; PontosExtrasZero35_Z;...
        yface.All.Zero40_Suave; PontosExtrasZero45_Z;...
        yface.All.Zero50_Suave; PontosExtrasZero55_Z;...
        yface.All.Zero60_Suave; PontosExtrasZero65_Z;...
        yface.All.Zero70_Suave; PontosExtrasZero75_Z;...
        yface.All.Zero80_Suave; yface.All.Zero85_Suave;
        yface.All.Zero90_Suave];

    Z2Paramaters_41Extras = [SuctionSide.All.Zero15_Suave;
SuctionSide.All.Zero20_Suave; ...
        SuctionSide.All.Zero25_Suave;
        SuctionSide.All.Zero30_Suave; PontosExtrasZero35_Z_Suction;...
        SuctionSide.All.Zero40_Suave; PontosExtrasZero45_Z_Suction;...
        SuctionSide.All.Zero50_Suave; PontosExtrasZero55_Z_Suction;...
        SuctionSide.All.Zero60_Suave; PontosExtrasZero65_Z_Suction;...
        SuctionSide.All.Zero70_Suave; PontosExtrasZero75_Z_Suction;...
        SuctionSide.All.Zero80_Suave; SuctionSide.All.Zero85_Suave;
        SuctionSide.All.Zero90_Suave];

    Y1Parameters_41Extras =
transpose([YPlot.Zero15_2,YPlot.Zero20_2,YPlot.Zero25_2,...
        YPlot.Zero30_2, PontosExtrasZero35_Y,...
        YPlot.Zero40_2, PontosExtrasZero45_Y,...
        YPlot.Zero50_2, PontosExtrasZero55_Y,...
        YPlot.Zero60_2, PontosExtrasZero65_Y,...
        YPlot.Zero70_2, PontosExtrasZero75_Y,...
        YPlot.Zero80_2,YPlot.Zero85_2,YPlot.Zero90_2]);
end


X2Parameters_41Extras = [X1Parameters_41Extras; X1Parameters_41Extras];
ZParameters_41Extras = [Z1Parameters_41Extras; Z2Paramaters_41Extras];
Y2Parameters_41Extras = [Y1Parameters_41Extras; Y1Parameters_41Extras];

%% Geração arquivo .CSV

TipPoints = [Tip.rR95_TE; Tip.rR975_TE; Tip.rR100; Tip.rR975_LE; Tip.rR95_LE];

Parameters = [Passo; TipPoints; X2Parameters_41Extras, Y2Parameters_41Extras,
ZParameters_41Extras]; % Parametros Suavizados + Pontos

NumeroSecoes = unique(Parameters(:,2));
for i=length(NumeroSecoes):-1:1
    if NumeroSecoes(i) > (D/2)
        NumeroSecoes(i) = [];
    end
end
NumeroSecoes = length(NumeroSecoes)-3;
```

```
TamanhoSecoes = 0;
for i = 1:length(Parameters)
    if Parameters(i,2) == (D/2*0.15)
        TamanhoSecoes = TamanhoSecoes + 1;
    end
end

TamanhoSecoes = TamanhoSecoes/2;

Info = [10,0,0; D,0,0; NumeroLaminas,0,0; NumeroSecoes,0,0 ;
TamanhoSecoes,0,0];

Zero95 = get(handles.chb_Zero95,'Value');

if Zero95 == 1
    Info = [95,0,0; D,0,0; NumeroLaminas,0,0; NumeroSecoes,0,0 ;
TamanhoSecoes,0,0];
end

Parameters = [Info ; Parameters];
NomeSalvo = get(handles.etxt_Nome,'string')
csvwrite(fullfile(NomeSalvo), Parameters)
msgbox ' Concluido! Arquivo gerado com sucesso! '
% Salvando o arquivo no mesmo diretório da Rotina MATLAB


function etxt_D_Callback(hObject, eventdata, handles)
% hObject    handle to etxt_D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etxt_D as text
%        str2double(get(hObject,'String')) returns contents of etxt_D as a double


% --- Executes during object creation, after setting all properties.
function etxt_D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etxt_D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function etxt_PD_Callback(hObject, eventdata, handles)
% hObject    handle to etxt_PD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etxt_PD as text
%        str2double(get(hObject,'String')) returns contents of etxt_PD as a double


% --- Executes during object creation, after setting all properties.
function etxt_PD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etxt_PD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etxt_Z_Callback(hObject, eventdata, handles)
% hObject    handle to etxt_Z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etxt_Z as text
%        str2double(get(hObject,'String')) returns contents of etxt_Z as a double


% --- Executes during object creation, after setting all properties.
function etxt_Z_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etxt_Z (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function etxt_EAR_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to etxt_EAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etxt_EAR as text
%        str2double(get(hObject,'String')) returns contents of etxt_EAR as a double


% --- Executes during object creation, after setting all properties.
function etxt_EAR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etxt_EAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
msgbox 'Leia o Tópico 5 do TCC. Nas modelagens testadas mostram-se melhores
resultados sem a presença das seções r/R = 0.75, 0.85 e 0.95, porem devido ao fato
de ser impossivel testar todas as combinações, caso o usuário prefira, existe a
opção de adicionar esses pontos.'



function etxt_Nome_Callback(hObject, eventdata, handles)
% hObject    handle to etxt_Nome (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of etxt_Nome as text
%        str2double(get(hObject,'String')) returns contents of etxt_Nome as a double


% --- Executes during object creation, after setting all properties.
function etxt_Nome_CreateFcn(hObject, eventdata, handles)
% hObject    handle to etxt_Nome (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

APÊNDICE B

Rotina Rhinoceros (Iron Python)

```
# Importa as bibliotecas necessarias

import rhinoscriptsyntax as rs
import Rhino

# Configura o Rhino para unidade de medida em Metros e Tolerancia = 0.1 mm
rs.UnitSystem(4, False, True)
rs.UnitAbsoluteTolerance(0.001)
rs.UnitAngleTolerance(0.001)
rs.UnitDistanceDisplayPrecision(0.01)
rs.UnitRelativeTolerance(0.01)

#Seleciona o arquivo gerado pelo MATLAB
filename = rs.OpenFileName("Open CSV file","*.csv|", None, None, None)
file = open(filename, 'r')
lines = file.readlines()
file.close()

#Deletar cabecalhos
#del lines[0]

Zero95 = int((lines[0].split(','))[0])
del lines[0]

ptInfo = []
for i in range(0,4):
    ptInfo.append(float((lines[i].split(','))[0]))

for i in range(0,4):
    del lines[0]

D = float(ptInfo[0])
print "Diametro =", float(ptInfo[0]) , "Metros"
Z = int(ptInfo[1])
print "Numero de Laminas =", Z
NumeroSecoes = int(ptInfo[2])
print "Quantidade de secoes (r/R) =", NumeroSecoes
PontosPorSecao = int(ptInfo[3])
print "Quantidade de pontos para cada lado da secao =", PontosPorSecao

# Considerando que nao tenham secoes acima de 0.9 e os angulos extras sao 0.95
e 1.0
QuantidadeAngulos = NumeroSecoes + 3

AnguloPasso = []
Raios = []
```

```
for i in range (0,QuantidadeAngulos):
    ptInfo1 = lines[i].split(',')
    AnguloPasso.append (float(ptInfo1[1]))
    Raios.append (float(ptInfo1[0]))

for i in range (0,QuantidadeAngulos):
    del lines[0]

#Caso queira-se conferir os dados importados.
#print(lines)

# VermelhoRGB (Codigo RGB referente a cor)
VermelhoRGB = [255,0,0]

# ListaIDPontos (Lista com as ID's de cada ponto importado pelo .CSV)
ListaIDPontos = []

# ListaCoordendasPontos (Lista com as coordenadas dos pontos importados pelo
.CSV)
ListaCoordenadasPontos = []

# Desativa a visualizacao passo a passo do scrip no Rhino
rs.EnableRedraw(False)

# Adicona os pontos importados ao Rhino
for line in lines:
    #remove the \n
    #line = line.strip()
    #split the line by the comma
    ptInfo = line.split(',')
    x = float(ptInfo[0])
    y = float(ptInfo[1])
    z = float(ptInfo[2])
    pt = rs.AddPoint(x,y,z)
    ListaIDPontos.append(pt)
    rs.ObjectColor(pt, VermelhoRGB)
    pts = "%.4f,%.4f,%.4f" %(x,y,z)
    ListaCoordenadasPontos.append(pts)

rs.Redraw()

#TipIDPontos (ID Pontos correspondentes a r/R = 0.95 ; 0.975 ; 1.0
TipIDPontos = []
#TipCoordenadasPontos (Coordenadas Pontos correspondentes a r/R = 0.95 e 1.0)
TipCoordenadasPontos = []

for i in range (0,10):
    TipIDPontos.append(ListaIDPontos[i])
    TipCoordenadasPontos.append(ListaCoordenadasPontos[i])
```

```
# Delete os pontos da TIP que ja foram armazenados nas listas acima
for i in range (0,10):
    del(ListaIDPontos[0])
    del(ListaCoordenadasPontos[0])

# Final e Intervalo(numero de pontos que define as superficies de pressao/succao)
Inicio = 0
Final = PontosPorSecao
Intervalo = PontosPorSecao

Lista1 = []
ListaIDCurvas = []

# Gera uma curva entre os pontos para cada r/R
for j in range(0,NumeroSecoes*2):
    for i in range(Inicio,Final):
        Lista1.append(ListaCoordenadasPontos[i])
    CurveID = rs.AddCurve(Lista1,3)
#    CurveID = rs.AddInterpCurve(Lista1,3)
    ListaIDCurvas.append(CurveID)
    Lista1= []
    Inicio = Final
    Final = Final + Intervalo

CurvasPressaoID = []
CurvasSuccaoID = []

# Sepando as curvas do lado de pressao e succao
for i in range (0,NumeroSecoes):
    CurvasPressaoID.append(ListaIDCurvas[i])

for i in range (NumeroSecoes,NumeroSecoes*2):
#for i in range (21,10,-1):
    CurvasSuccaoID.append(ListaIDCurvas[i])

PontosBordoAtaque_Succao = []
PontosBordoFuga_Succao = []
PontosBordoAtaque_Pressao = []
PontosBordoFuga_Pressao = []

# Gerando pontos na extremidades iniciais e finais das curvas
for i in range(0,NumeroSecoes):

PontosBordoAtaque_Succao.append(rs.AddPoint(rs.CurveEndPoint(CurvasSucca
oID[i])))

PontosBordoFuga_Succao.append(rs.AddPoint(rs.CurveStartPoint(CurvasSuccao
ID[i])))
```

```
PontosBordoAtaque_Pressao.append(rs.AddPoint(rs.CurveEndPoint(CurvasPressaoID[i])))

PontosBordoFuga_Pressao.append(rs.AddPoint(rs.CurveStartPoint(CurvasPressaoID[i])))

rs.DeleteObjects(ListaIDPontos)

PontosBordoSuccao = []
PontosBordoPressao =[]

for i in range (0,NumeroSecoes):
    PontosBordoSuccao.append(PontosBordoAtaque_Succao[i])
    PontosBordoPressao.append(PontosBordoAtaque_Pressao[i])

for i in range (NumeroSecoes-1,-1,-1):
    PontosBordoSuccao.append(PontosBordoFuga_Succao[i])
    PontosBordoPressao.append(PontosBordoFuga_Pressao[i])

ArcosPlanificadosAtaque = []
ArcosPlanificadosFuga = []

# Gerando arcos com inicio, final e direcao para fechar os bordos de ataque e fuga
DirecaoArco1 = (1,0,0)
DirecaoArco2 = (-1,0,0)
DirecaoArco3 = (0,1,0)
for i in range (0,NumeroSecoes):

ArcosPlanificadosAtaque.append(rs.AddArcPtTanPt(PontosBordoSuccao[i],DirecaoArco1,PontosBordoPressao[i]))
for i in range (NumeroSecoes,NumeroSecoes*2):

ArcosPlanificadosFuga.append(rs.AddArcPtTanPt(PontosBordoSuccao[i],DirecaoArco2,PontosBordoPressao[i]))

ArcosPlanificadosFuga_Reverso = list(reversed(ArcosPlanificadosFuga))

# Aplicando o passo para cada r/R
Origem = (0,0,0)
EixoY = (0,1,0)

for j in range (0,NumeroSecoes):
    rs.RotateObjects(CurvasSuccaoID[j],Origem,AnguloPasso[j],axis=EixoY)
    rs.RotateObjects(CurvasPressaoID[j],Origem,AnguloPasso[j],axis=EixoY)
    rs.RotateObjects(ArcosPlanificadosAtaque[j],Origem,AnguloPasso[j],axis=EixoY)

for j in range (0,NumeroSecoes):
        rs.RotateObjects(ArcosPlanificadosFuga_Reverso[j],Origem,AnguloPasso[j],axis=EixoY)
```

```
# Pontos Tip
TipArcoTE = []
TipArcoTE.append(rs.AddArcPtTanPt(TipIDPontos[0], DirecaoArco2,
TipIDPontos[1]))
TipArcoTE.append(rs.AddArcPtTanPt(TipIDPontos[2], DirecaoArco2,
TipIDPontos[3]))
TipArcoLE = []
TipArcoLE.append(rs.AddArcPtTanPt(TipIDPontos[8], DirecaoArco1,
TipIDPontos[9]))
TipArcoLE.append(rs.AddArcPtTanPt(TipIDPontos[6], DirecaoArco1,
TipIDPontos[7]))

rs.RotateObjects(TipIDPontos[0], Origem, AnguloPasso[-3],axis=EixoY)
rs.RotateObjects(TipIDPontos[1], Origem, AnguloPasso[-3],axis=EixoY)
rs.RotateObjects(TipIDPontos[2], Origem, AnguloPasso[-2],axis=EixoY)
rs.RotateObjects(TipIDPontos[3], Origem, AnguloPasso[-2],axis=EixoY)
rs.RotateObjects(TipIDPontos[4], Origem, AnguloPasso[-1],axis=EixoY)
rs.RotateObjects(TipIDPontos[5], Origem, AnguloPasso[-1],axis=EixoY)
rs.RotateObjects(TipIDPontos[6], Origem, AnguloPasso[-2],axis=EixoY)
rs.RotateObjects(TipIDPontos[7], Origem, AnguloPasso[-2],axis=EixoY)
rs.RotateObjects(TipIDPontos[8], Origem, AnguloPasso[-3],axis=EixoY)
rs.RotateObjects(TipIDPontos[9], Origem, AnguloPasso[-3],axis=EixoY)

rs.RotateObjects(TipArcoTE[0], Origem, AnguloPasso[-3], axis=EixoY)
rs.RotateObjects(TipArcoTE[1], Origem, AnguloPasso[-2], axis=EixoY)
rs.RotateObjects(TipArcoLE[0], Origem, AnguloPasso[-3], axis=EixoY)
rs.RotateObjects(TipArcoLE[1], Origem, AnguloPasso[-2], axis=EixoY)

rs.DeleteObjects(PontosBordoAtaque_Succao)
rs.DeleteObjects(PontosBordoFuga_Succao)
rs.DeleteObjects(PontosBordoAtaque_Pressao)
rs.DeleteObjects(PontosBordoFuga_Pressao)

Cilindros = []
CurvasPullPressao = []
CurvasPullSuccao = []
ArcosPullFuga = []
ArcosPullAtaque = []
OrigemCilindro = [0,0,-2.5]

# Conformando cilindricamente as curvas e os arcos
for i in range(0,NumeroSecoes):
    CilindroID = rs.AddCylinder(OrigemCilindro, 5, Raios[i]*(D/2))
    Cilindros.append(CilindroID)
    PulledCurveSuccao = rs.PullCurve(CilindroID,CurvasSuccaoID[i])
    CurvasPullSuccao.append(PulledCurveSuccao)
    PulledCurvePressao = rs.PullCurve(CilindroID,CurvasPressaoID[i])
    CurvasPullPressao.append(PulledCurvePressao)
    PullArcosAtaque = rs.PullCurve(Cilindros[i],ArcosPlanificadosAtaque[i])
```

```
    ArcosPullAtaque.append(PullArcosAtaque)
    PullArcosFuga = rs.PullCurve(Cilindros[i],ArcosPlanificadosFuga_Reverso[i])
    ArcosPullFuga.append(PullArcosFuga)

Cilindro95 = rs.AddCylinder(OrigemCilindro, 5, 0.95*(D/2))
Cilindro975 = rs.AddCylinder(OrigemCilindro, 5, 0.975*(D/2))
Cilindro100 = rs.AddCylinder(OrigemCilindro, 5, 1*(D/2))

SuperficieCilindro95 = rs.ExplodePolysurfaces(Cilindro95)
SuperficieCilindro975 = rs.ExplodePolysurfaces(Cilindro975)
SuperficieCilindro100 = rs.ExplodePolysurfaces(Cilindro100)

TipTE95 = []
TipTE95.append(TipIDPontos[0])
TipTE95.append(TipIDPontos[1])
TipTE975 = []
TipTE975.append(TipIDPontos[2])
TipTE975.append(TipIDPontos[3])
Tip100 = []
Tip100.append(TipIDPontos[4])
Tip100.append(TipIDPontos[5])
TipLE975 = []
TipLE975.append(TipIDPontos[6])
TipLE975.append(TipIDPontos[7])
TipLE95 = []
TipLE95.append(TipIDPontos[8])
TipLE95.append(TipIDPontos[9])

TipPulledPointsTE95 =
rs.AddPoints(rs.PullPoints(SuperficieCilindro95[0],TipTE95))
TipPulledPointsTE975 =
rs.AddPoints(rs.PullPoints(SuperficieCilindro975[0],TipTE975))
TipPulledPoints100 = rs.AddPoints(rs.PullPoints(SuperficieCilindro100[0],Tip100))
TipPulledPointsLE975 =
rs.AddPoints(rs.PullPoints(SuperficieCilindro975[0],TipLE975))
TipPulledPointsLE95 =
rs.AddPoints(rs.PullPoints(SuperficieCilindro95[0],TipLE95))

TipArcoTEPulled = []
TipArcoTEPulled.append(rs.PullCurve(Cilindro95, TipArcoTE[0]))
TipArcoTEPulled.append(rs.PullCurve(Cilindro975, TipArcoTE[1]))
TipArcoLEPulled = []
TipArcoLEPulled.append(rs.PullCurve(Cilindro95, TipArcoLE[0]))
TipArcoLEPulled.append(rs.PullCurve(Cilindro975, TipArcoLE[1]))

ArcoTip =
rs.AddArcPtTanPt(TipPulledPoints100[0],DirecaoArco3,TipPulledPoints100[1])

#Criacao do Cubo
if Z == 3:
```

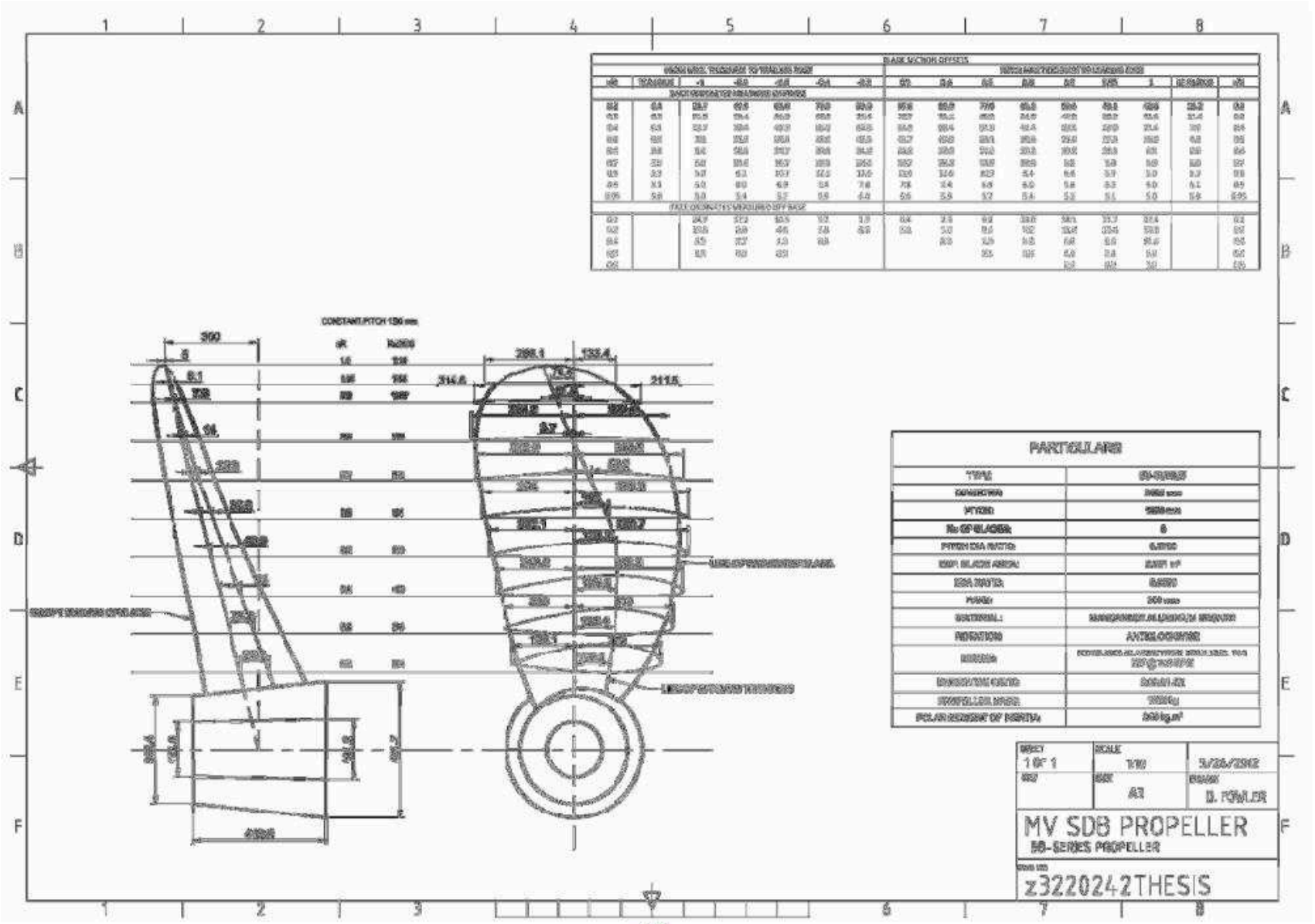
```
    HubDiameter = 0.18*D
HubDiameter = (1/6)*D
PontoIDFinalHub = rs.AddPoint(rs.CurveStartPoint(ArcosPullFuga[0]))
PontoIDInicialHub = rs.AddPoint(rs.CurveEndPoint(ArcosPullAtaque[0]))
PontoFinalHub = rs.PointCoordinates(PontoIDFinalHub)
PontoInicialHub = rs.PointCoordinates(PontoIDInicialHub)
PontoMedio = (PontoFinalHub[2] - PontoInicialHub[2])/2
ComprimentoMinHub = (PontoFinalHub[2] - PontoInicialHub[2])*1.3
OrigemZ = PontoInicialHub[2] + PontoMedio - ComprimentoMinHub/2

OrigemHub = (0,0,OrigemZ)
FinalHub = (0,0,OrigemHub[2]+ComprimentoMinHub)

CirculoInicioInterno = rs.AddCircle(OrigemHub, HubDiameter/4)
CirculoInicioExterno = rs.AddCircle(OrigemHub, HubDiameter/2)
CirculoFinalInterno = rs.AddCircle(FinalHub, HubDiameter/4)
CirculoFinalExterno = rs.AddCircle(FinalHub, HubDiameter/2)

SuperficieExterna = rs.ExtrudeCurveStraight(CirculoInicioExterno, OrigemHub,
FinalHub)
SuperficieInterna = rs.ExtrudeCurveStraight(CirculoInicioInterno, OrigemHub,
FinalHub)

Topo = (CirculoInicioInterno, CirculoInicioExterno)
TopoSrf = rs.AddLoftSrf(Topo)
Baixo = (CirculoFinalInterno, CirculoFinalExterno)
BaixoSrf = rs.AddLoftSrf(Baixo)
CuboSuperficies = (SuperficieExterna,SuperficieInterna, TopoSrf, BaixoSrf)
CuboSrf = rs.JoinSurfaces(CuboSuperficies)

rs.DeleteObjects(CuboSuperficies)
rs.DeleteObject(PontoIDFinalHub)
rs.DeleteObject(PontoIDInicialHub)
rs.DeleteObject(CirculoInicioInterno)
rs.DeleteObject(CirculoInicioExterno)
rs.DeleteObject(CirculoFinalInterno)
rs.DeleteObject(CirculoFinalExterno)
rs.DeleteObjects(TipLE95)
rs.DeleteObjects(TipLE975)
rs.DeleteObjects(TipTE95)
rs.DeleteObjects(TipTE975)
rs.DeleteObjects(TipArcoTE)
rs.DeleteObjects(TipArcoLE)
rs.DeleteObjects(ArcosPlanificadosAtaque)
rs.DeleteObjects(ArcosPlanificadosFuga)
rs.DeleteObjects(Cilindros)
rs.DeleteObjects(CurvasSuccaoID)
rs.DeleteObjects(CurvasPressaoID)

rs.DeleteObjects(SuperficieCilindro95)
```

```
rs.DeleteObjects(SuperficieCilindro975)
rs.DeleteObjects(SuperficieCilindro100)
rs.DeleteObject(Cilindro95)
rs.DeleteObject(Cilindro975)
rs.DeleteObject(Cilindro100)
rs.DeleteObjects(Tip100)

PontosFinaisAtaque = []
PontosFinaisFuga = []
PontosFinais = []
PontosIniciaisAtaque = []
PontosIniciaisFuga = []
PontosIniciais = []

#Pontos finais e inicias das curvas
for i in range (0,NumeroSecoes):
    EndPointsArcoAtaque = rs.CurveEndPoint(ArcosPullAtaque[i])
    PontosFinaisAtaque.append(rs.AddPoint(EndPointsArcoAtaque))
    EndPointsArcoFuga = rs.CurveEndPoint(ArcosPullFuga[i])
    PontosFinaisFuga.append(rs.AddPoint(EndPointsArcoFuga))
    StartPointsArcoAtaque = rs.CurveStartPoint(ArcosPullAtaque[i])
    PontosIniciaisAtaque.append(rs.AddPoint(StartPointsArcoAtaque))
    StartPointsArcoFuga = rs.CurveStartPoint(ArcosPullFuga[i])
    PontosIniciaisFuga.append(rs.AddPoint(StartPointsArcoFuga))

PontosFinaisFuga_Reverso = list(reversed(PontosFinaisFuga))
PontosIniciaisFuga_Reverso = list(reversed(PontosIniciaisFuga))

PontosFinais = PontosFinaisAtaque
if Zero95 == 95:
    PontosFinais.append(TipPulledPointsLE95[0])
PontosFinais.append(TipPulledPointsLE975[0])
PontosFinais.append(TipPulledPoints100[0])
PontosFinais.append(TipPulledPointsTE975[0])
if Zero95 == 95:
    PontosFinais.append(TipPulledPointsTE95[0])
PontosIniciais = PontosIniciaisAtaque
if Zero95 == 95:
    PontosIniciais.append(TipPulledPointsLE95[1])
PontosIniciais.append(TipPulledPointsLE975[1])
PontosIniciais.append(TipPulledPoints100[1])
PontosIniciais.append(TipPulledPointsTE975[1])
if Zero95 == 95:
    PontosIniciais.append(TipPulledPointsTE95[1])

for i in range (0,NumeroSecoes):
    PontosFinais.append(PontosFinaisFuga_Reverso[i])
    PontosIniciais.append(PontosIniciaisFuga_Reverso[i])

# Linhas dos bordos
```

```
BordoPressao = rs.AddInterpCurve(PontosFinais,3)
BordoSuccao = rs.AddInterpCurve(PontosIniciais)

rs.DeleteObjects(TipIDPontos)
rs.DeleteObjects(PontosFinais)
rs.DeleteObjects(PontosIniciais)
rs.DeleteObjects(TipPulledPoints100)

CurvasBordo = ArcosPullAtaque
if Zero95 == 95:
    CurvasBordo.append(TipArcoLEPulled[0])
CurvasBordo.append(TipArcoLEPulled[1])
CurvasBordo.append(ArcoTip)
CurvasBordo.append(TipArcoTEPulled[1])
if Zero95 == 95:
    CurvasBordo.append(TipArcoTEPulled[0])

for i in range (0,NumeroSecoes):
    CurvasBordo.append(ArcosPullFuga[i])

CurvasBordo.append(BordoPressao)
CurvasBordo.append(BordoSuccao)

#Superficie dos bordos
SrfBordo = rs.AddNetworkSrf(CurvasBordo, continuity = 3)

#Lado de Succao
domainSuccao = rs.CurveDomain(BordoSuccao)
parameterSuccao = (domainSuccao[1] / 2)*0.95
BordoSuccaoSplitted = rs.SplitCurve(BordoSuccao, parameterSuccao)

domainSuccao2 = rs.CurveDomain(BordoSuccaoSplitted[1])
parameterSuccao2 = (domainSuccao2[1] / 2)*1.05
BordoSuccaoSplitted2 = rs.SplitCurve(BordoSuccaoSplitted[1], parameterSuccao2)

CurvasSuccao = CurvasPullSuccao
CurvasSuccao.append(BordoSuccaoSplitted[0])
CurvasSuccao.append(BordoSuccaoSplitted2[0])
CurvasSuccao.append(BordoSuccaoSplitted2[1])

#Superficie de succao
SrfSuccao = rs.AddNetworkSrf(CurvasSuccao,continuity=3)

#Lado de Pressao
domainPressao = rs.CurveDomain(BordoPressao)
parameterPressao = (domainPressao[1] /2) * 0.95
BordoPressaoSplitted = rs.SplitCurve(BordoPressao, parameterPressao)

domainPressao2 = rs.CurveDomain(BordoPressaoSplitted[1])
parameterPressao2 = (domainPressao2[1]/2)*1.05
```

```
BordoPressaoSplitted2 = rs.SplitCurve(BordoPressaoSplitted[1],
parameterPressao2)

CurvasPressao = CurvasPullPressao
CurvasPressao.append(BordoPressaoSplitted[0])
CurvasPressao.append(BordoPressaoSplitted2[0])
CurvasPressao.append(BordoPressaoSplitted2[1])

#Superficie de pressao
SrfPressao = rs.AddNetworkSrf(CurvasPressao,continuity=3)

rs.DeleteObjects(CurvasPressao)
rs.DeleteObjects(CurvasSuccao)
rs.DeleteObjects(CurvasBordo)

Superficies = (SrfPressao, SrfSuccao, SrfBordo)
JointedSrf = rs.JoinSurfaces(Superficies)

rs.DeleteObjects(SrfPressao)
rs.DeleteObjects(SrfSuccao)
rs.DeleteObjects(SrfBordo)

CenterPoint = (0,0,0)
EixoZ= (0,0,1)

Angulo = 0
RotationalAngles = []
for i in range (0,Z-1):
    Incremento = 360/Z
    Angulo = Angulo + Incremento
    RotationalAngles.append(Angulo)

for i in range (0,Z-1):
    rs.RotateObject(JointedSrf, CenterPoint, RotationalAngles[i], axis=EixoZ,
copy=True)

if Zero95 == 10:
    rs.DeleteObjects(TipPulledPointsLE95)
    rs.DeleteObjects(TipPulledPointsTE95)
    rs.DeleteObject(TipArcoLEPulled[0])
    rs.DeleteObject(TipArcoTEPulled[0])

print "Concluido!"
```

ANEXO I


PARTICULARS
MV SDB PROPELLER
SB-SERIES PROPELLER
z3220242THESIS
B. FOWLER