UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

CURSO DE GRADUAÇÃO EM ENGENHARIA MECÂNICA

JEFFERSON MATHEUS DA SILVA BEZERRA

IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO DE RESFRIAMENTO REGENERATIVO EM UM MOTOR FOGUETE BIPROPELENTE LÍQUIDO

FORTALEZA

2019

JEFFERSON MATHEUS DA SILVA BEZERRA

IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO DE RESFRIAMENTO REGENERATIVO EM UM MOTOR FOGUETE BIPROPELENTE LÍQUIDO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecânica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Claus Franz Wehmann

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B469i Bezerra, Jefferson Matheus da Silva.
IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO DE RESFRIAMENTOREGENERATIVO EM UM MOTOR FOGUETE BIPROPELENTE LÍQUIDO / Jefferson Matheus da Silva Bezerra. – 2019.
101 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Mecânica, Fortaleza, 2019.
Orientação: Prof. Dr. Claus Franz Wehmann.

1. Motor foguete. 2. Resfriamento regenerativo. 3. Transferência de calor. 4. Programa de computador. 5. Gradiente descendente. I. Título.

CDD 620.1

JEFFERSON MATHEUS DA SILVA BEZERRA

IMPLEMENTAÇÃO DE UM MODELO NUMÉRICO DE RESFRIAMENTO REGENERATIVO EM UM MOTOR FOGUETE BIPROPELENTE LÍQUIDO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecânica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Mecânica.

Aprovada em: 25 de Novembro de 2019

BANCA EXAMINADORA

Prof. Dr. Claus Franz Wehmann (Orientador)
Universidade Federal do Ceará (UFC)

Prof. M. Eng. Antonio Carlos Foltran
Universidade Federal do Paraná (UFPR)

Prof. Dr. Clodoaldo de Oliveira Carvalho Filho
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Indubitavelmente à minha mãe, professora e bibliotecária, por me levar a biblioteca desde de cedo. O que me fez enxergar valor nos livros e na educação. Por oferecer a mim apoio integral e incondicional durante toda está jornada, mesmo que por vezes contestando as minhas escolhas profissionais. Fazendo sempre mais do que era viável. Me presenteando com a segurança necessária para desfrutar plenamente de uma graduação, sem precisar me preocupar com outras coisas.

À Talita, por seu essencial apoio durante este percurso e por sua compreensão do imenso tempo despendido em atividades acadêmicas durante todos esses anos. Tempo este que eu preferiria estar na companhia dela.

À Cleiton e Ilana, meu tio e minha tia, à Layane e Lívia, minhas primas, que, no início, me receberam nesta cidade e sempre deixaram a porta de sua casa aberta para mim.

À tia Jacinta, por possibilitar o lugar onde residi em maior parte da graduação, afirmando seu apoio desde o princípio desta etapa.

Ao Pedim, meu excepcional amigo, uma constante em mundo de amizades cada vez mais descartáveis.

Aos amigos que a universidade promoveu: Samyo, Baiano, Daniel, Tito, Marcelo, Thiaguinho, Thiagão, Victor, Leozin, Rodolfo, Bruno, Jean, Magro, Carlos, JV e tantos outros. Obrigado por toda a convivência dentro e fora dos muros da universidade.

Ao Prof. Claus Franz Wehmann pela oportunidade e orientação neste trabalho. Aos Profs. Antonio Carlos Foltran e Clodoaldo de Oliveira Carvalho Filho pela participação na banca e as valiosas críticas ao trabalho.

Por fim, a todas as pessoas que não tornaram o caminho mais tortuoso, pois muito ajuda quem não atrapalha.

"O mundo faz muito menos sentido do que você pensa. A coerência deriva principalmente do modo como sua mente funciona."

(Daniel Kahneman)

RESUMO

Na nova corrida espacial, os motores foguete continuam a ser componentes essenciais. No entanto, as elevadas temperaturas alcançadas nesta categoria de motor estão acima do ponto de fusão dos materiais comumente utilizados para a fabricação dos motores. Por isso, estratégias de resfriamento precisam ser adotadas para garantir o pleno funcionamento do dispositivo. O trabalho busca implementar um programa de computador para calcular uma solução numérica de resfriamento regenerativo de um motor foguete bipropelente e verificar o resultado obtido. A implementação é proposta neste trabalho como uma solução para cálculo do perfil de temperaturas para motores foguete bipropelente e também propõe aplicar isso para motores com diferentes características. O modelo e a interface gráfica são implementados em linguagem Python e integrados com outras ferramentas para realizar o cálculo do perfil de temperaturas do motor. O programa possui um otimizador baseado no método do gradiente descendente, que permite encontrar uma configuração dos canais que maximize o fluxo de calor, tendo em vista maximizar a troca de calor da parede do motor para o líquido de arrefecimento. A verificação do código é realizada utilizando dados do motor foguete bipropelente L-75, desenvolvido pela Força Aérea Brasileira, através da comparação do fluxo de calor ao longo do motor e de um comparativo do perfil de temperaturas com os resultados de outros autores. Uma geometria otimizada é gerada para outro motor e analisada. A aplicação e verificação do modelo a um motor foguete com diferentes características demonstrou a capacidade do programa de lidar com diferentes geometrias, propelentes e propriedades. Através do uso da técnica de otimização, valores de fluxo de calor foram maximizados para um motor. Resultando em um método relativamente simples para a obtenção de uma melhor geometria.

Palavras-chave: Motor foguete. Resfriamento regenerativo. Transferência de calor. Programa de computador. Método do Gradiente Descendente

ABSTRACT

Rocket engines remain essential components in the new space race. However, the temperatures achieved in these applications are above the melting point of the materials commonly used for these engines. Therefore, cooling strategies are essential in order to maintain the device working properly. This work aims to implement a numerical regenerative solution of a bi-propellant rocket engine and verify the result. The implementation is proposed in this work as a possible solution for the temperature profile calculation for two propellant rocket engines and it also proposes scaling this possibility for engines with different characteristics. A model and a graphical interface are implemented in Python language and integrated with other tools to perform the engine temperature profile calculation. There's also a code developed to find a geometry that matches the engine used. Using a optimization technique based on the descending gradient method to maximize the heat exchange from the wall to the refrigerant. Code verification is performed using data from the L-75 bi-propellant rocket engine developed by Brazilian Air Force and comparing the heat flux throughout the engine and the temperature profile with results found in the literature. At this point, an optimal geometry developed for another engine and analyzed. The application and verification of the model to a rocket engine with other parameters demonstrated the program's ability to handle different geometries, propellants and properties. Using the optimization technique heat flow values were maximized for an engine, resulting in a relatively simple method for obtaining a better geometry.

Keywords: Rocket engine. Regenerative cooling. Heat transfer. Program computer. Gradient descent.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CEA	NASA Chemical Equilibrium with Applications
CSV	*Comma-separated values*
FAB	Força Aérea Brasileira
GDAe	Grupo de Desenvolvimento Aeroespacial
IAE	Instituto de Tecnologia Aeronaútica
o/f	razão combustível-oxidante
RP-1	Querosene Refinado 1
SI	Sistema Internacional de Unidade
UFC	Universidade Federal do Ceará

LISTA DE SÍMBOLOS

c_p	Calor específico a pressão constante
e	Rugosidade na superfície interna do tubo
g	Aceleração da gravidade
h_{lt}	Perda de carga total
k	Coeficiente de condução
M	Número de Mach
Nu	Número de Nusselt
o/f	Razão de mistura do combustível
P	Perímetro molhado do tubo
Pr	Número de Prandtl
p	Pressão
ρ	Massa específica
R_c	Resistência térmica de convecção nas aletas
R_w	Resistência térmica de condução da parede
R_g	Resistência térmica de convecção na parede interna
T_c	Temperatura do líquido de arrefecimento
T_{wc}	Temperatura na base da superfície estendida
T_{wg}	Temperatura na parede interna do motor
T_{aw}	Temperatura dos produtos da combustão
t	Espessura da parede interna
V	Velocidade
α	Coeficiente de energia cinética
η_f	Desempenho da aleta
η_o	Desempenho global da aleta
μ	Viscosidade dinâmica
γ	Coeficiente de expansão adiabática dos gases quentes

SUMÁRIO

1 INTRODUÇÃO

Uma nova corrida espacial progride, antes fomentada por governos, agora impulsionada por empresas. A viabilidade econômica é o novo foco e, em busca desse objetivo, novas técnicas e nichos são explorados, como os pousos verticais da SpaceX e as investidas em turismo espacial da Blue Origin (BOWLER, 2019).

A propulsão a jato ainda é peça central neste cenário. A física deste fenômeno pode ser explicada através da mecânica newtoniana. O propulsor a jato, devido à conservação da quantidade de momento, gera uma força de reação chamada de empuxo (HILL; PETERSON, 2014).

Os motores a propulsão a jato podem ser divididos em várias categorias, como: *turbojets*, *turbofans*, *ramjets* e motores foguete. O motor foguete se difere dos outros, segundo Hill e Peterson (2014), devido a sua capacidade de carregar consigo todo propelente necessário para sua operação. Esta característica permite seu uso em elevadas altitudes, inclusive além da atmosfera terrestre. Em conjunto com o alto empuxo em relação a seu peso, esta categoria de motor tem um desempenho superior em comparação as outras categorias e tem seu próprio campo de aplicação, sem competição com outros motores (SUTTON; BIBLARZ, 2011). Suas principais aplicações são o lançamento e propulsão de veículos espaciais, como os que levaram o homem à lua, e mísseis militares que podem ter como carga desde explosivos a armas de destruição em massa.

Existem várias maneiras de classificar os motores foguete, seja pela sua sua fonte de energia, por sua aplicação, por tipo de combustível ou maneira pela qual o empuxo é produzido. Focando especificamente neste último, quando o empuxo é produzido como uma consequência das reações químicas, em que um conjunto de reações produzem gases de exaustão que atingem altas velocidades e também elevadas temperaturas (entre 2500 °C a 4100 °C), ao mesmo tempo em que ocorre à expansão dentro da câmara de combustão e ao longo do motor (SUTTON; BIBLARZ, 2011).

As elevadas temperaturas passíveis de serem alcançadas podem influenciar nas propriedades mecânicas de resistência e fazer com que as tensões causem deformações significativas no motor. Além disto, as temperaturas alcançadas estão acima do ponto de fusão dos materiais comumente utilizados para a fabricação dos motores, como as ligas de cobre. Por isso, estratégias de resfriamento precisam ser adotadas para garantir o pleno funcionamento do dispositivo. No entanto, dependendo do combustível utilizado, existem diferentes métodos para tentar reduzir a

temperatura do motor. Em foguetes que utilizam combustível líquido, duas técnicas são comumente utilizadas para a diminuição da temperatura: o resfriamento por filme e o resfriamento regenerativo. Cabe destacar que estas técnicas também podem ser utilizadas concomitantemente.

O primeiro método consiste na injeção de uma fina camada de um dos propelentes por sobre a superfície interna da câmara de combustão do motor, para reduzir a troca de calor entre os gases quentes e o material da parede. Já o resfriamento regenerativo tem este nome porque utiliza o combustível ou oxidante antes de ser injetado na câmara de combustão, para percorrer internamente a parede que compõe o motor e absorver calor, diminuindo a temperatura do material.

Durante o projeto de um motor foguete bipropelente, é necessário investigar o perfil de temperaturas ocasionado pelo fluxo de calor dos gases quentes que são expandidos para gerar empuxo. É possível realizar esta análise através de várias técnicas como: solução analítica, numérica e volumes finitos. Cada uma destas técnicas têm suas vantagens e desvantagens. Realizar uma análise analítica muitas vezes pode não ser viável ou sequer possível; um experimento, apesar de fornecer resultados mais confiáveis, pode ter um custo proibitivo; e simulações com volumes finitos podem demandar um alto poder computacional.

A implementação de um modelo numérico é proposta neste trabalho como uma solução para o cálculo do perfil de temperaturas para motores foguete bipropelente. O modelo desenvolvido em Foltran e Blavier (2018) é implementado utilizando a linguagem Pyhton, no intuito de criar um *software* em uma linguagem flexível para modularidade e extensões. A verificação do código é realizada utilizando o motor foguete biproplente L-75, em desenvolvimento pela Força Aérea Brasileira (FAB). Esta verificação é feita através da comparação do fluxo de calor ao longo do motor com os resultados de Almeida e Shimote (1999) e de Foltran e Blavier (2018), além de um comparativo do perfil de temperaturas com o último.

Devido à gama de possibilidades existentes, encontrar a geometria que maximize a troca de calor pode ser um desafio. Logo, para encontrar uma geometria satisfatória dos canais de resfriamento, a função objetivo é modelada tendo em vista maximizar a troca de calor entre a parede e o líquido de arrefecimento, utilizando uma técnica de otimização baseada no método do gradiente descendente. Este é um método iterativo de minimização que é simples e direto para a obtenção de soluções, mesmo podendo oferecer dificuldade para a convergência. (LUENBERGER; YE, 2008).

A técnica de otimização citada é aplicada ao motor bipropelente Ogum, atualmente

em desenvolvimento na Universidade Federal do Ceará (UFC) pelo Grupo de Desenvolvimento Aeroespacial (GDAe), para uma primeira averiguação do perfil de temperaturas e fluxo de calor relacionados ao motor.

Finalmente, para auxiliar na investigação do perfil de temperaturas, o *software* desenvolvido também calcula a perda de carga ocasionada pela geometria dos canais de resfriamento escolhida e qual a espessura mínima necessária para a parede interna suportar os esforços mecânicos devido à pressão no interior do motor.

2 OBJETIVOS

2.1 Objetivo geral

Implementar um programa de computador para calcular uma solução numérica de resfriamento regenerativo de um motor foguete bipropelente e verificar o resultado obtido.

2.2 Objetivos específicos

- Implementar o código do *software* utilizando uma linguagem de uso geral que facilite a extensão de suas funcionalidades.
- Apresentar um programa capaz de lidar com motores de diferentes características.
- Implementar uma metodologia de otimização da geometria dos canais internos do motor, visando maximizar a troca de calor.
- Desenvolver um *software* que seja adequado para o uso do cliente final.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Transferência de Calor em Cilindros Ocos

O cilindro oco pode ser considerado uma aproximação conveniente para modelar situações reais. Quando se trata da transferência de calor, pode-se considerar que o cilindro oco tem gradiente de temperatura apenas em sua direção radial, caso as paredes interna e externa tenham temperaturas diferentes e uniformes em toda a sua extensão. (INCROPERA *et al.*, 2008).

3.1.1 Condução

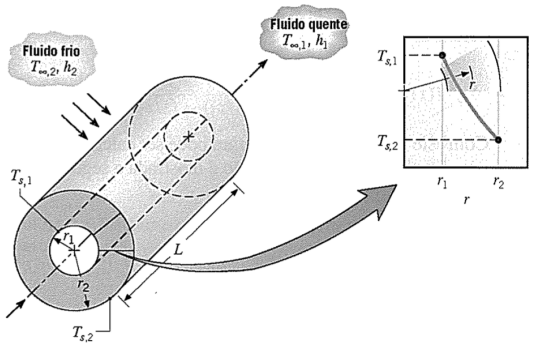
Em um cilindro oco, cujas superfícies estão expostas a diferentes temperaturas, considerando a hipótese de regime permanente e não havendo geração de calor, é possível realizar uma análise térmica neste corpo a partir da equação 3.1.

$$q_r = \frac{2\pi\, Lk\, (T_{s,1} - T_{s,2})}{ln(r_2/r_1)} \tag{3.1}$$

Na equação 3.1, $T_{s,1}$ e $T_{s,2}$ correspondem respectivamente a temperatura das superfícies interna e externa; L é o comprimento do cilindro; r_2 e r_1 são os raios externo e interno, respectivamente; q_r a taxa de transferência de calor entre a superfície interna e externa; e k representa o coeficiente de condução ou condutividade térmica, uma propriedade do material, que também irá governar a magnitude do fluxo de calor. Logo, nota-se que em regime permanente e considerando uma parede de geometria cilíndrica, o fluxo de calor em relação às temperaturas da superfície, não se comporta de forma linear, mas logarítmica.

Para uma melhor compreensão, na figura 1 está ilustrado um cilindro base para o desenvolvimento da seção 3.1.2.

Figura 1 – Representação de um cilindro oco com temperaturas constantes em sua superfície interna e externa



Fonte: Adaptado de Incropera *et al.* (2008).

3.1.2 Convecção

Já em regime permanente, com um coeficiente de convecção constante, a equação 3.2 é utilizada para calcular o fluxo de calor na convecção (INCROPERA *et al.*, 2008).

$$q = hA(T_s - T_\infty) \tag{3.2}$$

Na equação 3.2 o q corresponde a taxa de transferência de calor, T_s à temperatura na superfície e T_∞ à temperatura do fluido. A geometria do corpo aonde acontece a convecção influencia o coeficiente de convecção, por isso uma correlação adequada às condições do processo deve ser utilizada para estimar o coeficiente de convecção. A área da superfície onde o fluido escoa também é alterada pela geometria.

$$A = L2\pi r \tag{3.3}$$

Para o cálculo da taxa de transferência de calor através da parede cilíndrica, apenas deve-se substituir A na equação 3.2 pela equação 3.3, correspondente a área da parede externa do cilindro. Isto por que, neste caso, o fluxo de calor é modelado apenas na direção radial do objeto.

O coeficiente de convecção pode ser determinado através do número de Nusselt (Nu) descrito na equação 3.4. Pode ser considerado uma razão da transferência de calor por convecção em relação a transferência de calor por condução (INCROPERA *et al.*, 2008).

$$Nu = \frac{hL}{k_f} \tag{3.4}$$

Na equação 3.4, o k_f é o coeficiente de condução do fluído, h é o coeficiente de convecção e L é o comprimento característico da geometria. A partir de correlações que estimem o número de Nusselt pode-se encontrar o coeficiente de convecção h.

3.1.3 Correlação de Bartz

No estudo de Bartz (1957) foi proposta uma forma de calcular o coeficiente de convecção local no bocal de foguetes. A equação adaptada para o Sistema Internacional de Unidade (SI) está descrita em 3.5.

$$h = 0,026 \left(\frac{\mu_0}{D_t}\right)^{0,2} \frac{c_{p0}}{Pr_0^{0,6}} \left(\frac{p_0}{c_t}\right)^{0,8} \left(\frac{A_t}{A}\right) \sigma \tag{3.5}$$

O valor *Dt* corresponde ao diâmetro na garganta do motor, *At* a área da garganta e *A* a área da seção para qual o coeficiente de convecção é calculado. Os valores subscritos com 0 são referentes a valores de estagnação na câmara de combustão.

O σ, presente na equação 3.5, é um fator de correção que leva em conta os efeitos de compressibilidade sobre a camada limite e está descrito na equação 3.6.

$$\sigma = \left[\frac{T_{wg}}{2T_0}\left(1+\frac{\gamma-1}{2}M^2\right)+\frac{1}{2}\right]^{-0,68}\left[\left(1+\frac{\gamma-1}{2}M^2\right)\right]^{0,12} \tag{3.6}$$

A equação da correlação Bartz (1957) pode ser ajustada a partir de dados experimentais para uma melhor descrição dos fenômenos de transferência de calor.

3.1.4 Correlação de Sieder-Tate

Em escoamentos internos turbulentos plenamente desenvolvidos, correlações empíricas podem ser utilizadas para determinar o número de Nusselt. Já em escoamentos caracterizados por grandes mudanças nas propriedades, Incropera *et al.* (2008) recomenda a utilização da correlação de Sieder-Tate, mostrada na equação 3.7.

$$Nu_D = 0,027 Re_D^{4/5} Pr^{1/3}\left(\frac{\mu}{\mu_s}\right)^{0,14} \tag{3.7}$$

O μ representa a viscosidade do fluido na sua temperatura atual, enquanto o μ_s corresponde a viscosidade dinâmica do fluído avaliado na temperatura da superfície adjacente ao escoamento. As variáveis com o subscrito *D* indicam que o número de Nusselt é baseado no diâmetro hidráulico.

A correlação de Sieder-Tate apesar de ser sugerida para escoamentos internos, também pode ser aplicada em uma superfície que tenha temperatura e fluxo térmico constante, obtendo uma adequada aproximação (INCROPERA *et al.*, 2008).

É importante ressaltar que a correlação de Sieder-Tate tem uma restrição para seu uso. O número de Prandtl (Pr) deve estar no intervalo $0,7 \leq Pr \leq 16700$, o número de Reynolds local (Re_D) deve ser superior a dez mil, $Re_D > 10000$, e a razão entre o comprimento e o diâmetro deve ser superior a dez, $L/D \geq 10$. Segundo Incropera *et al.* (2008), os valores dessas propriedades devem estar aproximadamente respeitando estas restrições.

3.1.5 Superfícies Estendidas

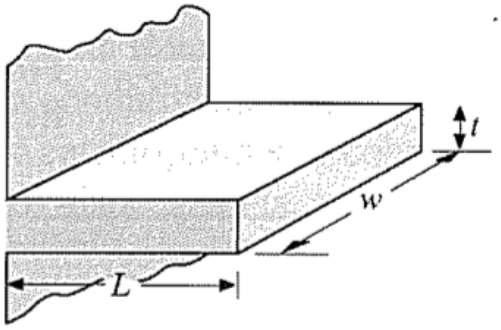
O desempenho em relação à superfície do sólido que a aleta ocupa e a quantidade de incremento de área correspondente a troca de calor ocasionada por ela é representada pela eficiência da aleta (η_f).

Para cada perfil de aleta, há um cálculo de desempenho. No caso da aleta plana retangular com ponta adiabática, η_f é calculado usando a equação 3.8.

$$\eta_f = \frac{tanh(mL_c)}{mL_c} \tag{3.8}$$

Onde $m = (2h/kt)^{(1/2)}$ e L_c é o comprimento da aleta corrigido. Para aletas com pontas adiabáticas, pode ser considerado apenas L. Sendo h o coeficiente de convecção e k o coeficiente de condução ou condutibilidade térmica. Na figura 2 observamos a dimensão t na geometria, que é a espessura da aleta.

Figura 2 – Principais dimensões de uma aleta plana retangular



Fonte: Incropera *et al.* (2008).

Quando utilizamos um conjunto de aletas, podemos calcular o desempenho do conjunto de aletas η_o utilizando a equação 3.9.

$$\eta_o = 1 - \frac{NA_a}{A_t}(1 - \eta_f) \tag{3.9}$$

Para isso é necessário calcular a área total da superfície. Isto pode ser feito utilizando a equação 3.10, aonde A_a é a área da aleta, A_b a área da superfície exposta da base das aletas e N o número de aletas na superfície.

$$A_t = NA_a + A_b \tag{3.10}$$

Com o desempenho do conjunto de aletas calculado toda a superfície da aleta pode ser expressa por uma resistência térmica, como mostra a equação 3.11.

$$R_a = \frac{1}{\eta_o h A_t} \tag{3.11}$$

A resistência R_a obtida da equação 3.11 pode ser considerada uma resistência efetiva, pois engloba a transferência por condução e por convecção na superfície primária (INCROPERA *et al.*, 2008).

3.2 Foguete ideal e relações termodinâmicas

A modelagem de um foguete em condições ideais é útil para expressar as suas condições físicas reais de forma simplificada, utilizando relações termodinâmicas. Em foguetes cuja propulsão é feita por reações químicas, seu desempenho real está de 1% a 6% abaixo do valor calculado utilizando a modelagem do foguete ideal. Utilizar as simplificações do foguete ideal para novos projetos é uma prática comum (SUTTON; BIBLARZ, 2011).

Abaixo estão descritas as hipóteses que compõe o modelo, conforme Sutton e Biblarz (2011).

1. Os produtos da reação química de combustão são homogêneos.
2. Todas as espécies químicas que compõe o fluido de trabalho são gasosas. Fases condensadas adicionam uma quantidade insignificante à massa total.
3. O fluido de trabalho é um gás ideal.
4. Não há transferência de calor entre o motor e a sua vizinhança. Os gases quentes da combustão são considerados um reservatório térmico.
5. O atrito do fluido de trabalho e os efeitos da camada limite são negligenciados.
6. Não há ondas de choque ou descontinuidades no escoamento.
7. A vazão mássica do propelente é constante. O trabalho realizado pelo fluido devido a expansão é uniforme. Os efeitos do regime transitório são de curta duração e podem ser negligenciados.
8. Todos os gases de escape saem na direção normal em relação ao divergente.
9. A velocidade do gás, pressão, temperatura e densidade são uniformes em qualquer seção que é normal em relação ao eixo longitudinal do motor.

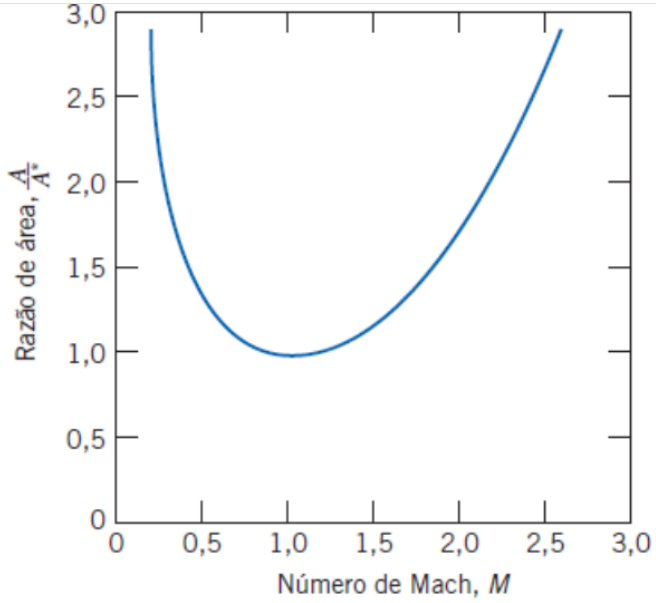
10. O fluido de trabalho está em equilíbrio químico e a composição do gás não muda (*frozen composition*).
11. Os propelentes armazenados estão à temperatura ambiente. Propelentes criogênicos estão em seu ponto de ebulição.

As considerações acima trazem algumas suposições, como a expansão dos gases ser isentrópica e o escoamento dos gases ser termodinamicamente reversível (SUTTON; BIBLARZ, 2011).

No subsequente desenvolvimento é conveniente usar o estado de estagnação como referência, denotado pelo subscrito 0. No estado de estagnação, consideram-se as propriedades do fluxo com velocidade zero (FOX *et al.*, 2014). Quando a velocidade na câmara de combustão está próxima de zero, a temperatura e a pressão se aproximam do estado de estagnação (SUTTON; BIBLARZ, 2011).

Em um escoamento compressível isentrópico unidimensional de um gás ideal, o número de Mach varia em função da razão de áreas, como pode-se ver no gráfico da figura 3. Em um motor foguete, podemos entender esta razão como sendo a área da seção dividida pela área da garganta.

Figura 3 – Variação do número de Mach em um escoamento compressível



Fonte: Fox *et al.* (2014).

Na condição de escoamento bloqueado (*choked flow*), um fluxo ao passar por uma constrição em velocidade sônica e pressão a jusante da constrição mais baixa que a montante, tem sua velocidade aumentada. Havendo pressão suficiente para "bloquear"o motor, então antes da

seção da garganta o escoamento é subsônico, exatamente na seção da garganta o atinge número de Mach 1 e a partir desse ponto torna-se supersônico.

O número de Mach do escoamento na condição de *choked flow* para as suposições citadas anteriormente é calculado utilizando a equação 3.12 (BORGNAKKE; SONNTAG, 2014).

$$\left(\frac{A}{A_t}\right) = \frac{1}{M}\left[\left(\frac{2}{\gamma+1}\right)\left(1+\frac{\gamma-1}{2}M^2\right)\right]^{(\gamma+1)/(\gamma-1)} \tag{3.12}$$

O γ corresponde ao coeficiente de expansão adiabática dos gases quentes. Como a equação 3.12 é transcendental, para obter o valor de M é necessário recorrer a métodos numéricos.

Se considerarmos os efeitos da camada limite no escoamento e a alta velocidade dos gases quentes, a temperatura de parede adiabática pode ser calculada utilizando a equação 3.13 (FOLTRAN; BLAVIER, 2018).

$$T_{aw} = T_0\left[\frac{1+Pr_0^{1/3}\frac{(\gamma-1)}{2}M^2}{1+\frac{(\gamma-1)}{2}M^2}\right] \tag{3.13}$$

Aonde T_{aw} é a temperatura dos produtos da combustão, Pr_0 o número de Prandtl em estagnação e T_0 a temperatura de estagnação.

3.3 Perda de Carga em Tubos

Quando o escoamento interno de um fluido viscoso flui em um tubo ou duto, há resistência do fluido a este escoamento. Por causa desse atrito, uma parcela da energia mecânica do fluido é convertida em calor. Esta perda de carga pode ser prevista através dos princípios da conservação de energia (FOX *et al.*, 2014).

$$\left(\frac{p_1}{\rho}+\alpha_1\frac{V_1^2}{2}+gz_1\right)-\left(\frac{p_2}{\rho}+\alpha_2\frac{V_2^2}{2}+gz_2\right)=h_{l_T} \tag{3.14}$$

Na equação 3.14 o termo h_{l_T} representa a perda de carga total que é a soma de duas partes: as perdas distribuídas (h_l) e as perdas localizadas (h_{l_m}), logo $h_{l_T} = h_l + h_{l_m}$. As outras variáveis são semelhantes às utilizadas na equação de Bernoulli, exceto o termo α que representa o coeficiente de energia cinética.

Em escoamentos com grandes números de Reynolds, α é próximo a um. Quando é adicionado a característica de escoamento em tubos, a variação de energia cinética é tipicamente pequena neste tipo de escoamento, portanto pode-se considerar $\alpha = 1$ (FOX *et al.*, 2014).

Não havendo mudança de energia potencial do sistema, a equação 3.14 torna-se a equação .

$$\frac{\Delta p}{\rho} = \frac{p_1 - p_2}{\rho} = h_{l_T} \tag{3.15}$$

Logo, é possível relacionar a perda de carga com a diferença de pressão em relação à entrada e a saída do tubo.

De acordo com Fox *et al.* (2014), para escoamentos turbulentos completamente desenvolvidos a perda de carga é calculada utilizando a equação 3.16, aonde L é o comprimento do tubo, D o diâmetro, V a velocidade e f o fator de atrito.

$$h_{l_T} = f\frac{L}{D}\frac{V^2}{2} \tag{3.16}$$

Em Fox *et al.* (2014) continua com o desenvolvimento dos cálculos e sugere formas para determinar o fator de atrito. Segundo ele, a equação de Coolebrook, equação (3.17), é a mais largamente utilizada. Nela a variável e representa a rugosidade na superfície interna do tubo.

$$\frac{1}{f^{0,5}} = -2log\left(\frac{e/D}{3,7} + \frac{2,51}{Ref^{0,5}}\right) \tag{3.17}$$

A equação (3.17) é transcendental e para resolve-la é necessária a utilização de métodos numéricos.

Em seu livro, Fox *et al.* (2014) mostra ainda que todo o desenvolvimento desta seção também pode ser utilizado para dutos não circulares, desde que as seções retas (transversais ao escoamento) não sejam extremamente grandes. Na equação 3.18 é apresentado o cálculo do diâmetro hidráulico D_h, que pode ser utilizado no lugar do diâmetro D, para dutos não circulares.

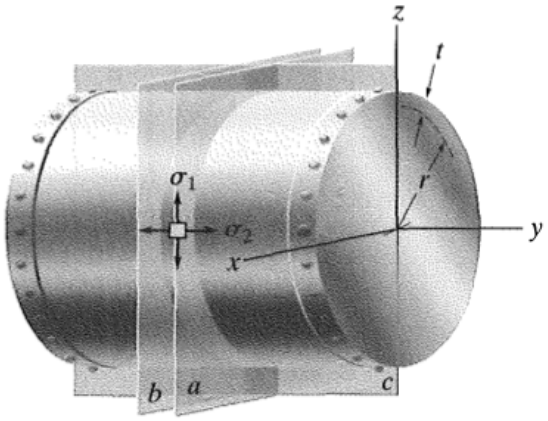
$$D_h = \frac{4A}{P} \tag{3.18}$$

Na equação 3.18, A representa a área e P o perímetro molhado do tubo.

3.4 Vasos de Pressão

Vasos de pressão são estruturas cilíndricas ou esféricas onde a diferença de pressão entre o interior e o exterior da estrutura causa tensão em todas as direções do material. No entanto, a análise do vaso de pressão pode ser simplificada se a condição de paredes finas for assumida, tornando-se um estado de tensão biaxial, conforme figura 4. Para se considerar a hipótese de paredes finas, o raio interno r dividido pela espessura t deve ser maior ou igual a dez (HIBBELER, 2009).

Figura 4 – Representação de um elemento de tensão biaxial em um vaso de pressão cilíndrico



Fonte: Hibbeler (2009).

Conforme Hibbeler (2009), as tensões normais na direção circunferencial e longitudinal são obtidas das equações 3.19 e 3.20, aonde p corresponde a pressão no interior do cilindro. Todas as pressões avaliadas são manométricas e na parede do cilindro as duas tensões são consideradas constantes.

$$\sigma_1 = \frac{pr}{t} \tag{3.19}$$

$$\sigma_2 = \frac{pr}{2t} \tag{3.20}$$

A teoria da energia de distorção relaciona as tensões principais no material e produz a tensão σ_e equivalente a uma tensão axial. Está tensão pode ser comparada mais facilmente a dados experimentais. A partir disso, pode ser utilizada como critério de falha para materiais

dúcteis. A equação 3.21 é utilizada neste critério de falha (HIBBELER, 2009).

$$\sigma_e^2 = \sigma_1^2 - \sigma_1\sigma_2 + \sigma_2^2 \tag{3.21}$$

O termo σ_e também é conhecido como tensão de Von Mises. A equação 3.21 é usada para material sob um estado biaxial de tensões.

3.5 Métodos Numéricos

Esta seção apresenta dois métodos numéricos, sendo o primeiro para o cálculo de zeros em funções e o segundo com objetivo de minimizar o valor de funções através do gradiente.

3.5.1 Bisseção

Em funções reais não lineares pode ser dificultoso ou impossível encontrar a sua solução de maneira analítica. Um dos métodos utilizados para solucionar este problema é o da bisseção.

Como descrito em Ruggiero e Lopes (1997), a partir de uma função $f(x)$ e um intervalo $[a,b]$, quando $f(a)f(b) < 0$ existirá pelo menos uma raiz no intervalo. Neste método o intervalo deverá ser criteriosamente escolhido para que o intervalo contenha uma raiz.

O objetivo do método reside em sucessivamente diminuir o intervalo ao meio, conduzir o teste supracitado, encontrando em qual dos subintervalos ocorre a raiz. Seleciona-se o subintervalo que contém a raiz e descarta-se o outro. Faz-se isso sucessivamente até a precisão ser atingida.

O algoritmo da bisseção, proposto por (RUGGIERO; LOPES, 1997), está mostrado na figura 5.

Figura 5 – Descrição do algoritmo da bisseção

1. Dados iniciais:
   a) intervalo inicial $[a,b]$
   b) precisão ε
2. Se $(b-a) < \varepsilon$, então escolha um x para qualquer $x \in [a,b]$. FIM.
3. $k = 1$
4. $M = f(a)$
5. $x = a+b/2$
6. Se $Mf(x) > 0$, faça $a = x$. Vá para o passo 8.
7. $b = x$
8. Se $(b-a) < \varepsilon$, escolha x para qualquer $x \in [a,b]$. FIM.
9. $k = k+1$. Volte ao passo 5.

Fonte: (RUGGIERO; LOPES, 1997)

O método da bisseção gera uma sequência convergente, desde que $f(x)$ seja contínua e $f(a)f(b) < 0$.

3.5.2 Gradiente descendente

Em problemas de minimização, o método do gradiente descendente pode ser utilizado para a otimização de problemas. Trata-se de uma técnica antiga e simples, por isso é normalmente a primeira a ser utilizada em um novo problema, também servindo como referência para outras técnicas (LUENBERGER; YE, 2008).

O operador matemático gradiente pode ser entendido como o vetor que tem o sentido e direção para onde se tem a maior variação no valor da função f. Sua definição para uma função de duas variáveis é encontrada na equação 3.22, conforme Stewart (2010). É importante ressaltar que a função f deve ter derivadas parciais continuas.

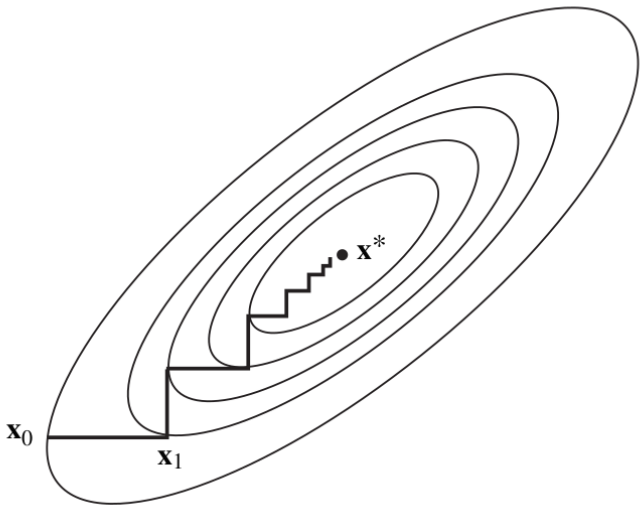
$$\nabla f(x,y) = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} \tag{3.22}$$

Como descrito em Luenberger e Ye (2008), o método consiste em sucessivas iterações nas quais a função objetivo f é levada a atingir um máximo ou mínimo local. A equação 3.23 mostra o algoritmo do método e a figura 6 sua representação gráfica.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{x}_k \tag{3.23}$$

Os vetores $\mathbf{x}$ correspondem ao valor de entrada da função objetivo f, aonde k indica o número da iteração a que o vetor se refere. O valor de α, passo da iteração, é um argumento do algoritmo e está relacionado com a velocidade deste até a sua convergência.

Figura 6 – Caminho percorrido pelo método na função objetivo



Fonte: Luenberger e Ye (2008).

Apesar de existirem técnicas para determinar um valor ótimo de α, uma boa aproximação pode ser obtida ao se utilizar um valor α entre 0 e 1. A convergência pode ser verificada através do teorema de convergência global pormenorizado em Luenberger e Ye (2008).

4 METODOLOGIA

Nesta seção será mostrado todo processo feito para o desenvolvimento do *software*, a integração realizada com o NASA Chemical Equilibrium with Applications (CEA) utilizando o pyCEA e as ferramentas utilizadas. Também serão descritos os dois algoritmos utilizados para o cálculo do perfil de temperaturas e para a otimização da troca de calor.

4.1 Ferramentas utilizadas

4.1.1 Linguagem Python e bibliotecas

O ambiente de desenvolvimento do presente estudo utiliza a linguagem Python em sua versão 3.5.3 e o sistema operacional Linux Debian 9. Duas bibliotecas relevantes foram utilizadas o pyCEA e o SciPy. Na seção 4.1.3 é descrita a utilização da biblioteca pyCEA. O SciPy é um conjunto de *softwares* de código aberto, na forma de bibliotecas, voltadas para a matemática, ciência e engenharia (SCIPY, 2019). Ele foi utilizado para auxiliar na resolução de equações, usando o método da bisseção e o cálculo numérico do gradiente.

4.1.2 CEA

O CEA é um código desenvolvido no Lewis Research Center, um centro de pesquisa da NASA, atualmente chamado Glenn Research Center e várias propriedades dos gases quentes são obtidas através dele. A captura é feita de maneira automática utilizando a biblioteca pyCEA. Os detalhes desta integração são explicados na seção 4.1.3. O código original do CEA é utilizado, sem nenhuma modificação. Dois modos do CEA foram utilizados para a obtenção das propriedades: o tp (temperatura e pressão especificadas) e o *rkt* (modo de foguete). O manual de usuário de Gordon e McBride (1994) foi utilizado como base para a operação do programa.

A figura 7 é um exemplo de um código de *input* para o CEA, demonstrando a maneira que foi utilizado o modo tp. As principais entradas são a temperatura (K), pressão (bar), o combustível, o oxidante e razão combustível-oxidante (o/f) em fração mássica. Na saída utiliza-se o $siunits$ para que as unidades estejam de acordo com o SI e propriedades de transporte sejam calculadas, pois as propriedades dos produtos de combustão γ e o c_p são utilizadas a partir da saída desta execução. Na linha 8 a opção *short* é importante para que o CEA gere um relatório menor, o que diminuirá o tempo de gravação e leitura dos arquivos pelo algoritmo que

irá calcular o perfil de temperaturas.

Figura 7 – Código de *input* para o CEA - modo *tp*

```
problem case=1 tp
  p(bar) = 60.0
  t(k) = 3318.7968981157937
react
  fuel=RP-1 wt=1
  oxid=O2(L) wt=2.42
output
  siunits short, transport
  plot t gam cp
end
```

Fonte: Elaborado pelo autor.

Na figura 8 está um exemplo de *input* do CEA utilizado o modo *rkt*. Algumas entradas são semelhantes ao código anterior, como combustível e oxidante. Como a composição química homogênea é assumida, a opção *frozen* é adicionada para este caso.

Figura 8 – Código de *input* para o CEA - modo *rkt*

```
problem    o/f=2.42,
    rocket  fac tcest,k=3800
  p,bar=60,
react
  fuel=RP-1
  oxid=O2(L)
output  transport
end
```

Fonte: Elaborado pelo autor.

4.1.3 Biblioteca PyCEA

A biblioteca PyCEA, desenvolvida por (ESPINOZA, 2019), é um invólucro para permitir o uso do CEA no Python. Apesar do clássico *software* de equilíbrio químico ser imple-

mentando em FORTRAN, a biblioteca consegue executar este código e capturar as propriedades de saída diretamente para a linguagem Python.

Entretanto, originalmente a biblioteca PyCEA não foi projetada para funcionar com os modos *t p* e *rkt* do CEA. Logo, foi preciso realizar uma extensão das funcionalidades originais da biblioteca para que ela pudesse satisfazer as demandas de modos e propriedades desejadas. A biblioteca PyCEA após as modificações está detalhada no apêndice 8.

É importante ressaltar que a biblioteca PyCEA não faz modificações no código-fonte ou execução do CEA, além de não trazer consigo o próprio código. Uma pasta com o nome *CEA+FORTRAN* contendo todo o código do programa de cálculo de equilíbrio químico existe no diretório de execução do programa e é necessária para o funcionamento correto da biblioteca.

Ao executar a biblioteca para a captura de uma propriedade a pasta *CEAoutput* é criada, caso não exista anteriormente, e ela conterá todos os arquivos de *input* e *output* utilizados na execução. Isto porque a cada iteração, toda vez que uma propriedade vinda do CEA é necessária, o código dele é executado e a propriedade é capturada. Apenas se existir um arquivo já gerado por uma iteração anterior, o CEA não é executado. Devido a isso há um ganho desempenho na leitura e gravação de arquivos e, consequentemente, na execução do programa. Este processo ocorre diversas vezes durante a execução da solução de um perfil de temperaturas.

4.1.4 WebPlotDigitizer

Para converter as figuras com a geometria do motor a ser validado para um formato discretizado em pontos num plano cartesiano, utilizou-se o *WebPlotDigitizer* 4.1.4. O formato de saída desta aplicação é um arquivo *Comma-separated values* (CSV). Esta ferramenta é de suma importância para a obtenção das geometrias utilizadas no presente trabalho. As geometrias utilizadas no trabalho estão disponíveis nos apêndices 11 e 12.

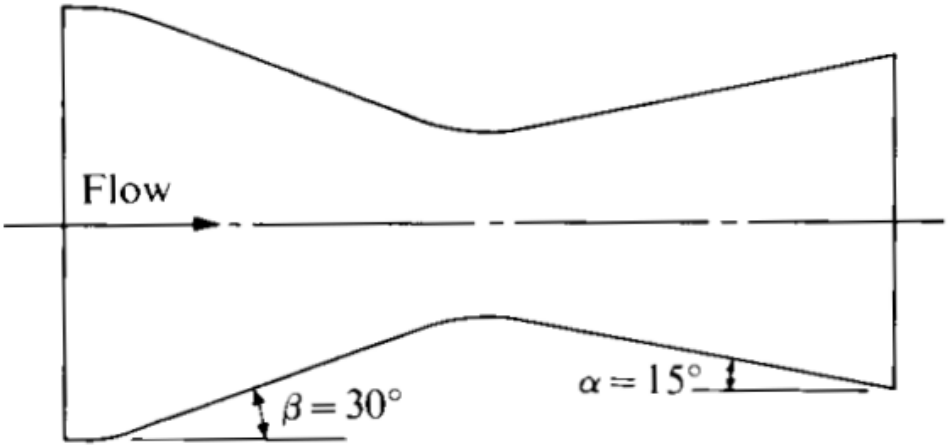
4.2 Modelo Númerico Implementado

O trabalho de Foltran e Blavier (2018) apresenta um modelo numérico para solução do perfil de temperaturas ao longo de um motor foguete bipropelente. Este modelo foi escolhido para ser implementado no presente trabalho. Esta subseção a seguir trata das equações e hipóteses utilizadas a partir do modelo escolhido.

Na figura 9, nota-se que o motor foguete tem, a grosso modo, um formato próximo ao

de um cilindro. Entretanto, as variações de área e de propriedades ao longo do eixo longitudinal não podem ser desprezadas. Em razão disso, o escoamento dos gases de combustão é quase-unidimensional e todo o motor é dividido em várias seções cilíndricas de acordo com seu comprimento.

Figura 9 – Câmara de combustão de motor foguete



Fonte: Hill e Peterson (2014).

A geometria dos canais de resfriamento pode ser observada na figura 10. Os canais são espaçados igualmente ao redor do motor e cabe destacar três das suas dimensões: a largura do canal, altura do canal e a espessura da aleta. O diâmetro da câmara de combustão é a variável independente. A espessura das aletas ou a largura dos canais se alteram em acordo com espessura da câmara e o número de canais.

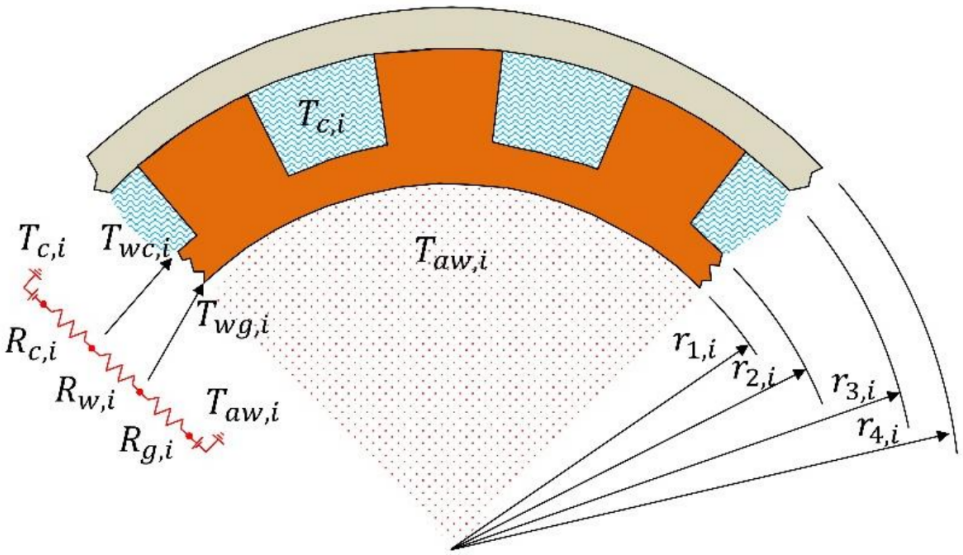
O CEA é utilizado para se obter propriedades dos gases quentes, como o calor específico à pressão constante (c_p) e o coeficiente de expansão adiabática (γ), nas várias seções em que o motor foi dividido. Também é usado para as propriedades de estagnação como a temperatura dos gases (T_0), número de Prandtl (Pr_0) e viscosidade dinâmica (μ_0).

No caso do motor foguete L-75, utilizado na validação do modelo, a propriedade γ relativa aos gases quentes foi obtida do trabalho de Blavier (2016). Esta propriedade é calculada em função da temperatura e está descrita na equação 4.1.

$$\gamma = 1,23854 \times 10^{-8} T_{aw}^2 - 8,09028 \times 10^{-5} T_{aw} + 1,34563 \tag{4.1}$$

As propriedades do combustível Querosene Refinado 1 (RP-1) são provenientes do trabalho de Boysan (2008), e estão descritas de maneira analítica para uma relevante faixa de

Figura 10 – Resistências térmicas e principais temperaturas na seção de um motor foguete com resfriamento regenerativo



Fonte: Foltran e Blavier (2018).

temperatura, 300 K a 800 K. No caso do motor foguete Ogum, que utiliza etanol como combustível, foram utilizadas propriedades constantes para 1 atm e temperatura de 273 K, são elas: massa específica (ρ), calor específico à pressão constante ($c_{p,c}$), coeficiente de condutividade térmica (k_c) e viscosidade dinâmica (μ_c). O subscrito c indica que a propriedade é do líquido de arrefecimento.

Em cada umas das seções é assumida a condição de *choked flow* e a equação 3.12 é usada para calcular o número de Mach para cada uma delas, utilizando o método da bisseção para encontrar as raízes da equação.

As condições iniciais utilizadas no método da bisseção para encontrar as raízes da equação 3.12 são diferentes para as seções à montante e à jusante da garganta. Antes o escoamento é subsônico e após a garganta é supersônico.

Em todo o modelo considera-se a operação em regime permanente e que a parede externa do motor não troca calor com o meio externo. A associação de resistências térmicas é feita na direção transversal do motor e o motor é subdividido em seções transversais ao longo de todo o seu comprimento. Considerando um número relativamente grande de seções transversais, é esperado que o perfil transversal de temperaturas não seja significativamente diferente entre duas associações adjacentes. Desta forma, considera-se que todo o calor transferido ocorre na direção radial apenas.

É possível modelar a troca de calor como um circuito térmico com três resistências,

como pode ser visto na figura 10. A resistência mais externa R_c é relacionada a convecção em uma superfície estendida, enquanto a do meio, R_w, é relativa à condução através da parede do motor, já a mais interna, R_g, corresponde a convecção entre os gases quentes e a parede interna do motor. A condução é calculada através da equação 3.1, utilizando as propriedades do material e as dimensões geométricas do motor.

Tabela 1 – Descrição das temperaturas do modelo de Foltran e Blavier (2018)

T_c	Líquido de Arrefecimento
T_{wc}	Base da superfície estendida
T_{wg}	Parede interna do motor
T_{aw}	Produtos da combustão

Fonte: Foltran e Blavier (2018).

Para o cálculo da resistência de convecção (R_g) é utilizada a equação 4.2, que modela a convecção de calor como uma resistência térmica. Já na superfície estendida é utilizada as equações descritas na seção 3.1.5. No cálculo das resistências de convecção e de superfície estendida, existe um coeficiente de convecção diferente para cada condição em que a troca de calor acontece.

No cálculo de R_c, na superfície estendida, é utilizado o coeficiente de convecção de Sieder-Tate, descrito na seção 3.1.4. Na resistência de convecção dos gases quentes, a correlação de convecção de Bartz (1957) é utilizada e está descrita em mais detalhes na seção 3.1.3.

O valor do fluxo de calor é calculado utilizando os gases quentes como um reservatório térmico. Dessa forma, o fluxo de calor pode ser obtido através da equação 4.2, da qual, com as devidas manipulações algébricas, podem ser obtidas as demais temperaturas.

$$q_i = \frac{T_{aw,i} - T_{c,i}}{R_{g,i} + R_{w,i} + R_{c,i}} = \frac{T_{aw,i} - T_{wg,i}}{R_{g,i}} = \frac{T_{aw,i} - T_{wc,i}}{R_{g,i} + R_{w,i}} \tag{4.2}$$

Várias propriedades, como as temperaturas, são dependentes de outras propriedades, impossibilitando o cálculo analítico. Por isso, na primeira iteração as temperaturas assumem valores iniciais que são recalculados nas próximas iterações.

Cabe destacar a temperatura do líquido de arrefecimento que está contracorrente, pois, em cada iteração, é recalculada utilizando um balanço de energia. Analisando uma única seção dentre as que o motor é dividido, é feito um volume de controle para calcular a temperatura do líquido de arrefecimento a partir do fluxo de calor. Na equação 4.3, $T'_{c,i}$ é a temperatura de

entrada do líquido de arrefecimento na seção e $T''_{c,i}$ é a temperatura de saída.

$$T''_{c,i} = \frac{q_i}{\dot{m}_c c_{p,i}} + T'_{c,i} \tag{4.3}$$

A temperatura $T_{c,i}$, calculada com a equação 4.4, é definida como a média da temperatura de entrada e saída do líquido de arrefecimento na seção.

$$T_{c,i} = \frac{T''_{c,i} + T'_{c,i}}{2} \tag{4.4}$$

Na seção adjacente, seguindo o sentido fluxo do líquido de arrefecimento da saída do divergente para a câmara de combustão, a temperatura de entrada dela é a temperatura de saída da seção anterior. Isto ocorre, até que todas as temperaturas do líquido de arrefecimento, em cada seção, sejam calculadas.

As iterações são feitas até que a erro da norma L_1 de todas as temperaturas seja menor que a tolerância, ou o máximo de iterações seja atingido.

4.3 Algoritmo para Cálculo das Temperaturas

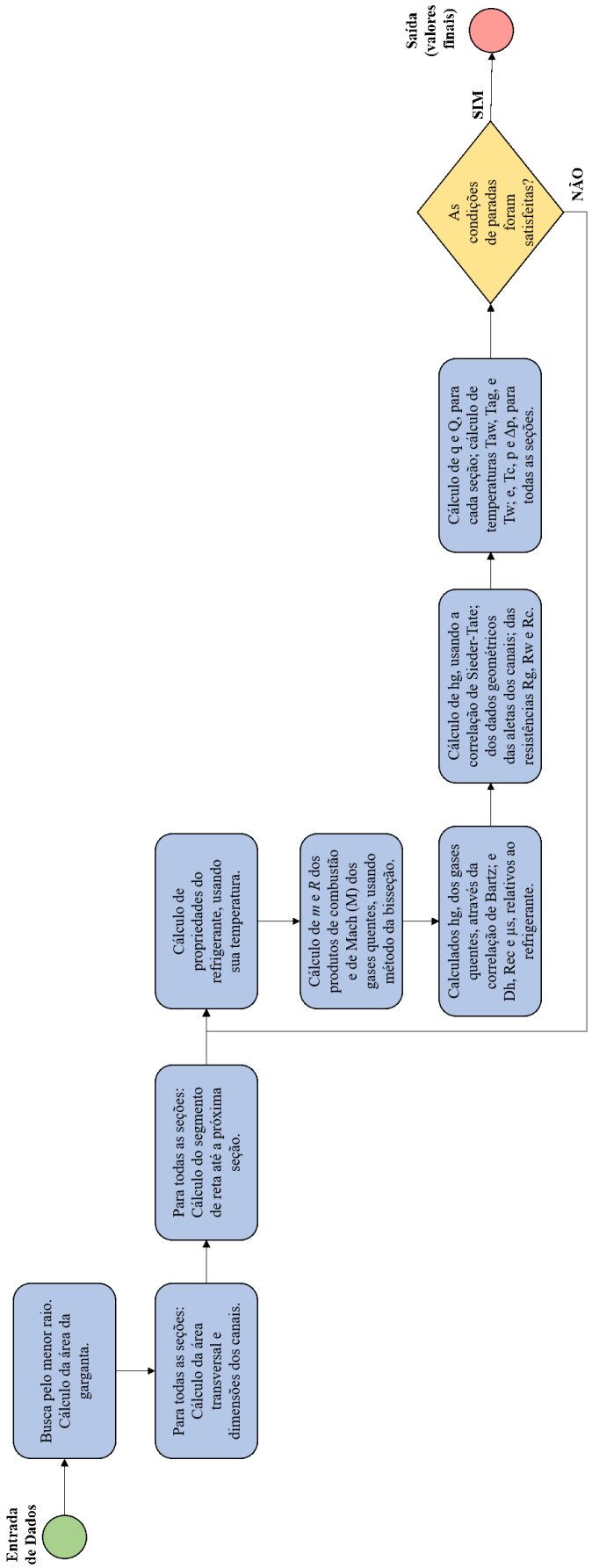
Para o funcionamento do algoritmo de cálculo das temperaturas do modelo, é necessária a entrada de vários dados sobre o motor como: combustível, oxidante, razão mássica entre oxidante e combustível (o/f), estimativa inicial das temperaturas, espessura da parede interna, condutividade térmica do material da câmara, vazão mássica do líquido de arrefecimento, limite de escoamento do material e rugosidade dos canais.

Também são necessárias três dimensões relacionadas as aletas e canais: a largura do canal, largura da aleta e a altura do canal, igual à altura da aleta. O número de canais e a altura do canal são uma entrada do algoritmo assim como uma das duas outras dimensões, que deve ser escolhida como fixa. O programa calculará a terceira dimensão em função da dimensão fixa considerando todos os canais equidistantes.

As condições de parada utilizadas são: quando o número máximo de iterações é atingido ou quando a norma L_1 das temperaturas é menor ou igual a tolerância especificada para o processo. O número máximo de iterações e a tolerância também são entradas.

O código fonte está integralmente disponibilizado no apêndice 7, seu fluxograma está mostrado na figura 11 e brevemente descrito abaixo.

Figura 11 – Caminho percorrido pelo método na função objetivo



Fonte: Elaborado pelo autor.

1. Carrega as informações de entrada necessárias, valores iniciais estimados de temperatura, o arquivo de geometria em formato CSV e conta a quantidade de pontos existentes.
2. Dados de propriedades de estagnação do produtos da combustão como pressão (p_0), temperatura (T_0), número de Prandtl (Pr_0), viscosidade dinâmica (μ_0) e calor específico a pressão constante ($c_{p,0}$) são obtidas do CEA.
3. Utilizando os dados de geometria carregados, busca pelo menor raio e calcula a área da garganta. Então, em todas as seções, os raios, a área transversal e dimensões dos canais são calculadas.
4. Em cada seção, um segmento de reta até a próxima seção no sentido de escoamento do líquido de arrefecimento, é calculado utilizando teorema de Pitágoras para representar o comprimento do canal.
5. As propriedades do líquido de arrefecimento como calor específico a pressão constante ($c_{p,c}$), condutividade térmica (k_c), viscosidade dinâmica (μ_c), número de Prandtl (Pr_c) são obtidas na temperatura atual do fluido.
6. Valores de calor específico a pressão constante (c_p) e o coeficiente de expansão adiabática (γ) dos produtos da combustão são obtidos através do CEA. Vazão mássica $\dot{m}$ e constante do ideal dos gases dos produtos de combustão (R) é calculada.
7. O valor do número de Mach (M) dos gases quentes é calculado utilizando o método da bisseção.
8. O coeficiente de convecção dos gases quentes (h_g) é calculado através da correlação de Bartz.
9. Os valores do diâmetro hidráulico (D_h), número de Reynolds (Re_c) e viscosidade dinâmica na temperatura da superfície do canal relativos ao líquido de arrefecimento são calculados.
10. O coeficiente de convecção do líquido de arrefecimento para o interior dos canais (h_g) é calculado usando a correlação de Sieder-Tate.
11. Dados geométricos das aletas dos canais como área da aleta (A_a), área total do conjunto de aletas (A_t), eficiência de aleta (η_o) e eficiência global (η_f) são calculados.
12. As resistências térmicas de convecção dos gases quentes (R_g), parede interna da câmara (R_w) e superfície estendidas onde o líquido de arrefecimento escoa (R_c) são calculadas.
13. A taxa de transferência de calor (q) e o fluxo de calor (Q) para cada seção são calculados.
14. As temperaturas dos gases quentes (T_{aw}), parede interna(T_{wg}), superfície dos canais (T_{wc}) são calculadas.

15. A temperatura do líquido T_c é calculada conforme o modelo para cada seção onde já existam todas propriedades calculadas.
16. A pressão estática (p) é calculada para cada seção.
17. A perda de carga (Δp) é calculada em cada seção para cada canal e seu total é contabilizado.
18. É verificado se as condições de parada foram atendidas, caso contrário, retorno ao passo 5.
19. Os valores finais de todo o processo são retornados ao programa principal.

4.4 Algoritmo de Otimização

O algoritmo de otimização baseado no método do gradiente, explicado na seção 3.5.2, é descrito abaixo e está disponível no apêndice 7.

Quase todas as entradas necessárias no algoritmo para o cálculo das temperaturas também devem estar presentes no algoritmo de otimização, exceto a largura da aleta ou do canal, dependendo de qual dimensão foi fixada. A altura do canal é utilizada como estimativa inicial para os cálculos. O passo w do gradiente descendente também é um valor de entrada para o algoritmo.

Dados mínimos da altura do canal e largura do canal ou aleta devem ser entrada para evitar a possibilidade de resultados demasiadamente pequenos.

A estimativa inicial da largura do canal ou aleta não é necessária como parâmetro de entrada, isto porque o algoritmo sugere uma estimativa baseada na largura máxima e mínima do canal ou aleta. A partir do número de canais é calculada uma largura máxima possível para o canal ou aleta e com a informação do valor mínimo de largura da aleta ou canal, o valor intermediário entre os dois é calculado e utilizado como estimativa.

A função objetivo utilizada na otimização tem como domínio o $\mathbb{R}^2$, sendo a altura do canal e a largura da aleta ou canal, dependendo de qual foi fixada como entrada, as suas dimensões. A imagem da função é o fluxo de calor (Q) ao qual se procura minimizar. A função objetivo está descrita 4.5.

$$f(h,l) = Q \tag{4.5}$$

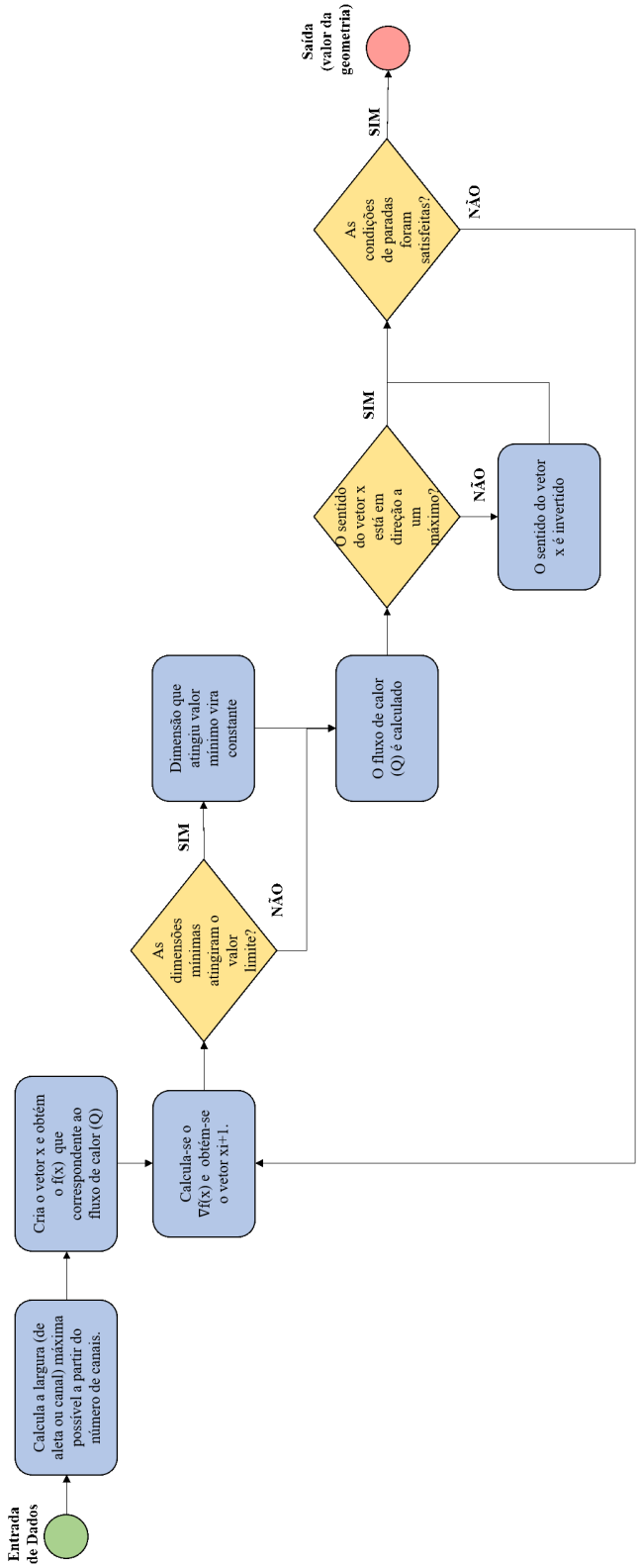
Onde h é a altura do canal e l a largura do canal ou aleta, já a função f é calculada pelo algoritmo para o cálculo das temperaturas descrito na seção 4.3.

O algoritmo termina quando o número máximo de iterações é atingido ou quando a tolerância para Q é igualada ou superada. Tanto a tolerância quanto o número máximo de

iterações são dados de entrada.

O fluxo do algoritmo está visto na figura 12 e seus passos estão descritos mais detalhadamente logo abaixo.

Figura 12 – Caminho percorrido pelo método na função objetivo



Fonte: Elaborado pelo autor.

1. Carrega os dados de entrada para a otimização.
2. Calcula a largura (de aleta ou canal) máxima possível a partir do número de canais.
3. Cria o vetor **x** que contém os valores de entrada da função objetivo e obtém o $f(\mathbf{x})$ que correspondente ao fluxo de calor (Q).
4. Calcula o $\nabla f(\mathbf{x})$ e com equação 3.23, é obtido o vetor $\mathbf{x}_{i+1}$.
5. É verificado se nenhuma das dimensões mínimas atingiu o valor limite máximo ou mínimo. Se a verificação for verdade, esta dimensão é tornada constante.
6. Utilizando o algoritmo de cálculo das temperaturas, o fluxo de calor (Q) é calculado.
7. É verificado o sentido do vetor **x**, caso esteja em direção a um máximo, o sentido é invertido.
8. É verificada se alguma das condições de parada é satisfeita, em caso negativo voltar ao passo 4.
9. Retorna o valor da geometria contendo altura do canal e largura da aleta ou canal ao programa principal.

5 RESULTADOS

A implementação do modelo culminou em uma biblioteca integrada com o CEA para a linguagem Python e em uma interface gráfica. Com esta biblioteca é possível calcular para um foguete bipropelente: os perfis de temperatura, fluxo de calor e propriedades relacionadas. A biblioteca e os dois códigos, relativos ao L-75 e o Ogum, estão disponíveis nos apêndices 7, 9 e 10, respectivamente.

5.1 Validação do Algoritmo para Cálculo das Temperaturas

A validação do *software* é feita por comparação com os resultados do fluxo de calor e perfis de temperatura obtidos por Foltran e Blavier (2018), que validou seu modelo através do fluxo de calor calculado analiticamente por Almeida e Shimote (1999). O objeto de validação utilizado é o motor foguete L-75 desenvolvido pelo Instituto de Tecnologia Aeronaútica (IAE).

Os dados da tabela 2 e 3 são os dados de entrada necessários para o *software* calcular os perfis de temperaturas ao longo do motor.

Tabela 2 – Dados do motor L-75

Dimensão constante	Largura da aleta
Altura do canal	0,0015 m
Largura da aleta	0,001 m
Espessura da parede interna	0.0015 m
Líquido de Arrefecimento	Querosene RP-1
Combustível	Querosene RP-1
Oxidante	Oxigênio Líquido
Razão de mistura	2,42
Vazão mássica do combustível	6,4 kg/s
Pressão de câmara	6 MPa
Condutividade térmica da câmara e aletas	290 W m^{-1} K^{-1}

Fonte: Foltran e Blavier (2018).

A tabela 3 contém as estimativas de temperaturas que são utilizadas para o início das iterações e os dados referentes à condição de parada do algoritmo, como a tolerância e o número máximo de iterações. Na geometria utilizada, a densidade de pontos que descreve o perfil do motor tem uma densidade maior próxima ao bocal devido a grandes alterações de propriedades nesta parte.

Tabela 3 – Dados relativos a validação assumidos para o motor L-75

Estimativa de temperatura dos gases quentes (T_{aw})	3000 K
Estimativa de temperatura da parede interna (T_{wg})	1000 K
Estimativa de temperatura da base da aleta (T_{wc})	600 K
Estimativa de temperatura do líquido de arrefecimento (T_c)	303 K
Tolerância relativa as iterações	1×10^{-6}
Número máximo de iterações	100
Número de seções axiais	166

Fonte: Elaborado pelo autor.

No motor utilizado, o número de aletas varia conforme a seção e sua configuração, como descritas na tabela 4. Estes valores também são entradas no programa para o cálculo dos perfis de temperatura.

Tabela 4 – Configuração do número de aletas do motor L-75

Distância longitudinal do injetor (mm)	**Número de canais**
$z < 293$	170
$293 \leq z < 425$	148
$425 \leq z < 523$	300
$523 \leq z < 1491$	458

Fonte: Foltran e Blavier (2018).

As demais informações de geometria estão descritas na tabela 5, no entanto, elas não são entradas diretas para o programa, pois esses dados são carregados através de um arquivo CSV.

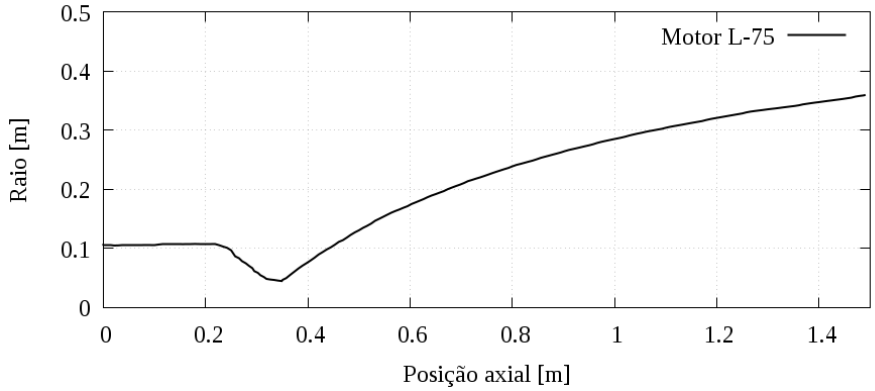
Tabela 5 – Dados de geometria do motor L-75

Comprimento total do motor	1491 mm
Comprimento total da câmara de combustão	285 mm
Diâmetro da câmara de combustão	211 mm
Diâmetro da gargata	90 mm
Diâmetro da saída	719 mm

Fonte: Foltran e Blavier (2018).

A geometria do L-75, obtida através do trabalho de Foltran e Blavier (2018), é plotada na figura 13 utilizando o Rohatgi (2019).

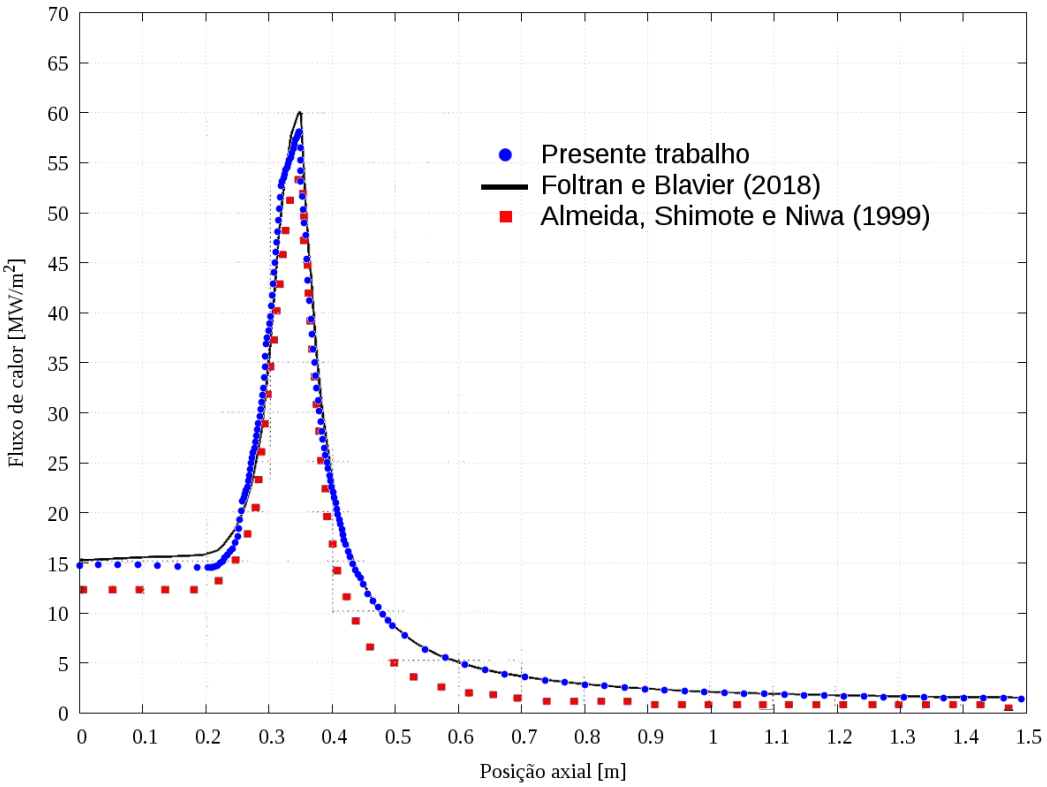
Figura 13 – Perfil externo do motor foguete L-75



Fonte: Elaborado pelo autor.

Na figura 14 estão plotadas três curvas de fluxo de calor para fim de comparação. Os resultados de Almeida e Shimote (1999) está em vermelho, o trabalho de Almeida e Shimote (1999) Foltran e Blavier (2018) em preto e o presente trabalho em azul.

Figura 14 – Fluxo de calor ao longo seção axial do motor foguete L-75



Fonte: Elaborado pelo autor.

As curvas do trabalho atual e de Foltran e Blavier (2018) sobrestimam as obtidas por Almeida e Shimote (1999), calculado com outra metodologia. Isto possivelmente explica esta diferença.

As curvas obtidas neste trabalho e em Foltran e Blavier (2018) estão concordantes após a garganta. Entretanto, apesar de utilizarem a mesma metodologia, há diferença entre as curvas nas seções antes da garganta ou exatamente nela.

Visando a melhoria da geometria foram feitas correções manuais, utilizando como base as dimensões presentes em (BLAVIER, 2016), porém as diferenças permaneceram. Estas diferenças são causadas pelo modo que a geometria é obtida, feita neste estudo de maneira gráfica utilizando o Rohatgi (2019). As pequenas diferenças da geometria, geradas para o cálculo, causam mudanças significativas nas curvas produzidas pelas iterações.

Ainda assim, a curva obtida foi bastante aceitável e seus erros relativos estão listados na tabela 6.

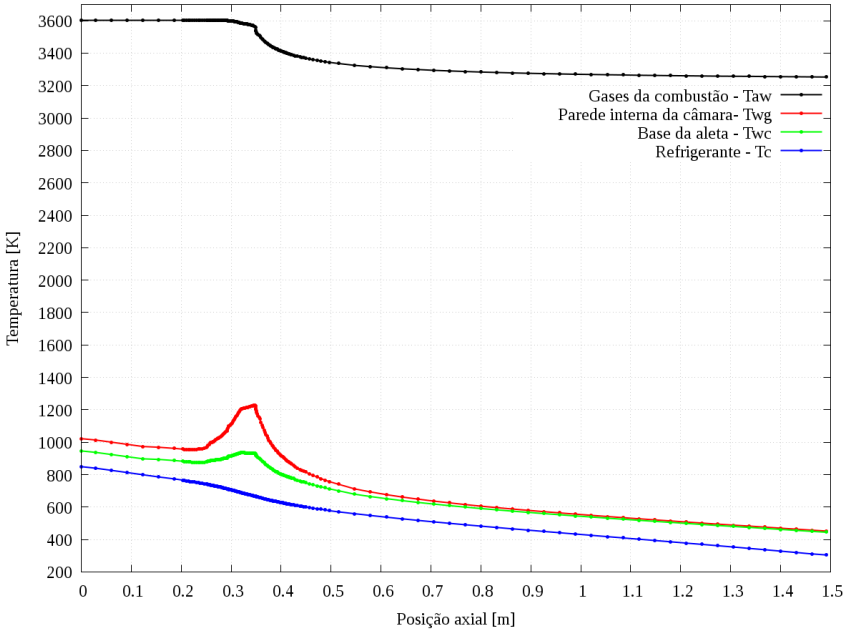
Tabela 6 – Erros relativos na validação utilizando o motor foguete L-75

Distância longitudinal do injetor de combustível [m]	Erro relativo do fluxo de calor
0,000	1,44 %
0,349	-0,84 %
0,500	5,39 %
1,491	23,26 %

Fonte: Elaborado pelo autor.

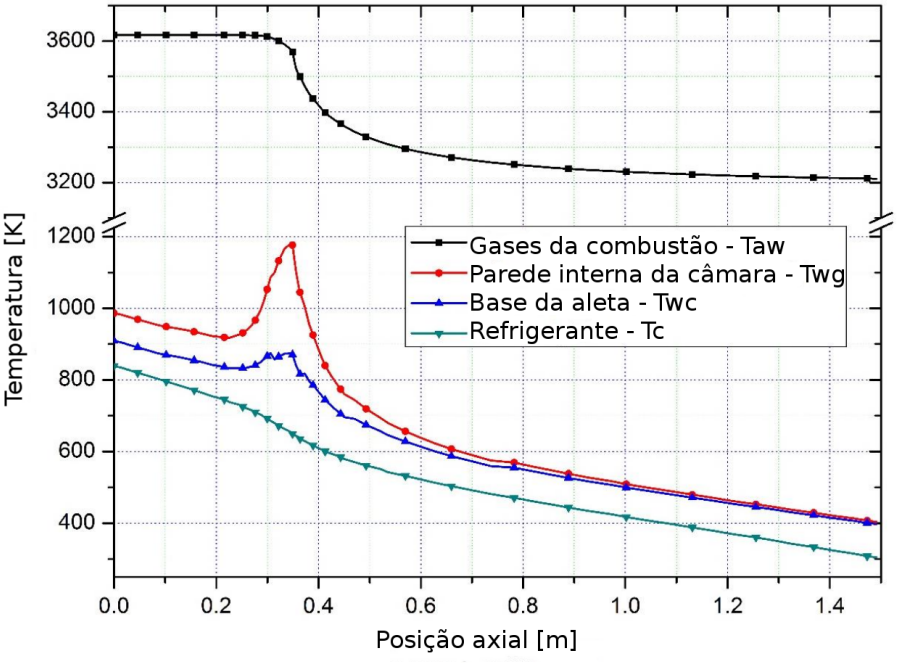
O perfil de temperaturas para o L-75 gerado neste trabalho está mostrando na figura 15, já o obtido por Foltran e Blavier (2018) está na figura 16. Dado que houve uma discrepância entre a curva do fluxo de calor gerada neste trabalho e a de Foltran e Blavier (2018), uma diferença também é esperada no perfil de temperaturas.

Figura 15 – Perfis de temperaturas do motor L-75 obtidas obtidas neste trabalho



Fonte: Elaborado pelo autor.

Figura 16 – Perfis de temperaturas do motor L-75 obtidas obtidas por Foltran e Blavier (2018)



Fonte: Foltran e Blavier (2018).

Nota-se que a temperatura dos gases de combustão tem valores idênticos ao de Foltran e Blavier (2018) no início da câmara de combustão, isto indica que o programa desenvolvido neste trabalho encontra uma temperatura de estagnação dos gases quentes condizente com Foltran e Blavier (2018). No entanto, na seção de exaustão, a temperatura em Foltran e Blavier (2018)

tem uma queda para um valor próximo a 3200 K. Portanto, há uma diferença de mais 50 K em relação aos resultados do atual estudo.

Esta diferença, observada após a garganta nos gases de combustão, também está presente em todas as outras curvas. As temperaturas na parede interna, na base da aleta e do líquido de arrefecimento também apresentam em toda sua extensão uma diferença em torno de 50 K para mais.

As variações encontradas nos perfis de temperatura também podem ser explicadas devido à maneira que a geometria do L-75 foi obtida. No entanto, apresentam bons valores quando comparadas as obtidas por Foltran e Blavier (2018). Na tabela 7 estão presentes os dados de saída relacionados ao motor L-75.

Tabela 7 – Dados de saída para o motor foguete L-75

Fluxo de calor máximo	58,095 MW/m^{-}2
Temperatura máxima da parede interna T_{wg}	1227,4 K
Perda de carga do combustível	637 kPa
Largura de canal máxima	3,96 mm
Iterações necessárias	18

Fonte: Elaborado pelo autor.

A tabela 6 contém um comparativo dos erros relativos entre os resultados de Foltran e Blavier (2018) e os deste trabalho. Os quatro pontos mostrados correspondem a posições importantes ao longo da câmara de combustão, aonde a posição zero corresponde ao injetor de combustível.

O comparativo através dos erros relativos reforça a observação feita no gráfico 16. Vale destacar que o maior erro, de 23,26% na saída do divergente, ocorre devido a pequenas variações quantitativas no valor diminuto do fluxo de calor na saída do convergente, ocasionando maior variação percentual.

O comparativo entre as curvas obtidas neste trabalho e as do objeto de validação apresentam diferenças não significativas, ocasionadas pela forma gráfica utilizada para obtenção das curvas. Apesar dessas pequenas disparidades entre as curvas, podemos considerar o modelo validado e apto a ser utilizado na verificação de outros motores foguete bipropelentes, desde que sejam válidas as mesmas hipóteses simplificadoras.

5.2 Otimização da Geometria do Motor Ogum

Ogum é um motor foguete bipropelente em desenvolvimento pelo GDAe. Um primeiro estudo do resfriamento do Ogum é realizado neste trabalho. A partir da implementação validada na seção 5.1, uma investigação da geometria, dimensão e quantidade de canais foi realizada utilizando o algoritmo de otimização descrito na seção 4.4.

O combustível utilizado neste motor é o etanol e suas propriedades constantes para a temperatura e pressão ambiente obtidas de Engineering ToolBox (2019) foram utilizadas como dados de entrada no programa. Os dados de entrada para a otimização estão descritos na tabela 8.

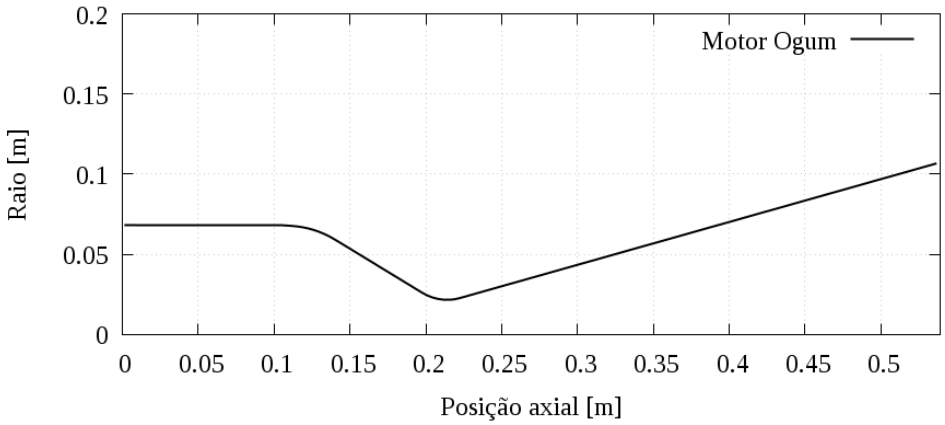
Tabela 8 – Dados de entrada do algoritmo relacionados ao motor Ogum

Dimensão constante	Largura do canal
Estimativa inicial de altura do canal	0,002 m
Espessura da parede interna	0,001 m
Estimativa de temperatura dos gases quentes (T_{aw})	3000 K
Estimativa de temperatura da parede interna (T_{wg})	1000 K
Estimativa de temperatura da base da aleta (T_{wc})	600 K
Estimativa de temperatura do líquido de arrefecimento (T_c)	303 K
Líquido de arrefecimento	Etanol
Combustível	Etanol
Oxidante	Óxido Nitroso
Razão de mistura	5,65
Vazão mássica do combustível	1,0 kg/s
Pressão de câmara	1 MPa
Material da câmara e aletas	Aço Inoxidável 304
Limite de ruptura	505 MPa
Condutividade térmica da câmara e aletas	16,2 W m^{-1} K^{-1}
Rugosidade dos canais	$1,6 \times 10^{-6}$ (Classe N7)
Coeficiente segurança estrutural da câmara	1.2
Tolerância relativa as iterações	1×10^{-6}
Número de seções axiais	166
Largura mínima de aleta	0,001 mm
Largura mínima de canal	0,001 mm
Altura mínima de canal	0,001 mm
Passo das iterações de otimização	0,05

Fonte: Elaborado pelo autor.

Além do cálculo dos perfis de temperatura e fluxo de calor, a perda de carga ocasionada pelos canais e a espessura mínima da parede interna também são calculadas.

Figura 17 – Perfil externo do motor foguete Ogum



Fonte: Elaborado pelo autor.

Ao gerar os pontos relativos à geometria de entrada, procurou-se adicionar uma maior densidade de pontos na região da garganta do motor, pois nesta área ocorrem as maiores mudanças de propriedade. Os dados de geometria estão descritos na tabela 9.

Tabela 9 – Dados de geometria do motor Ogum

Comprimento total do motor	539 mm
Comprimento total da câmara de combustão	125 mm
Diâmetro da câmara de combustão	136 mm
Diâmetro da garganta	43 mm
Diâmetro da saída	215 mm

Fonte: Elaborado pelo autor.

O algoritmo de otimização foi utilizado para encontrar o maior resultado de transferência de calor em um número de canais variando de 14 ao máximo geometricamente possível, neste caso 71.

O aumento do número de canais provoca uma diminuição da temperatura da parede interna, devido principalmente a ampliação da área de contato para a troca de calor. Como pode ser observado na tabela 10.

Também é possível notar que para valores pequenos de quantidade de canais, a altura do canal é a mínima limite, escolhida pelo usuário na entrada de dados, porém, a largura do canal começa mais de oito vezes maior que a largura mínima e, progressivamente, diminui até chegar ao seu valor mínimo limite.

Tabela 10 – Resultado de otimizações para várias quantidades de canais no motor foguete Ogum

Canais	T_{wg} (max) [K]	Q [MW]	Altura [mm]	Largura [mm]	Perda de carga [kPa]
14	1590,8	8,212	1,07	8,88	2,1
20	1584,0	8,264	1,06	6,11	3,1
30	1563,0	8,423	1,00	3,65	3,8
40	1569,2	8,376	1,01	1,01	1,1
50	1561,94	8,431	1,02	1,02	2,2
60	1554,1	8,491	1,03	1,03	3,9
71	1541,40	8,588	1,00	1,00	6,2

Fonte: Elaborado pelo autor.

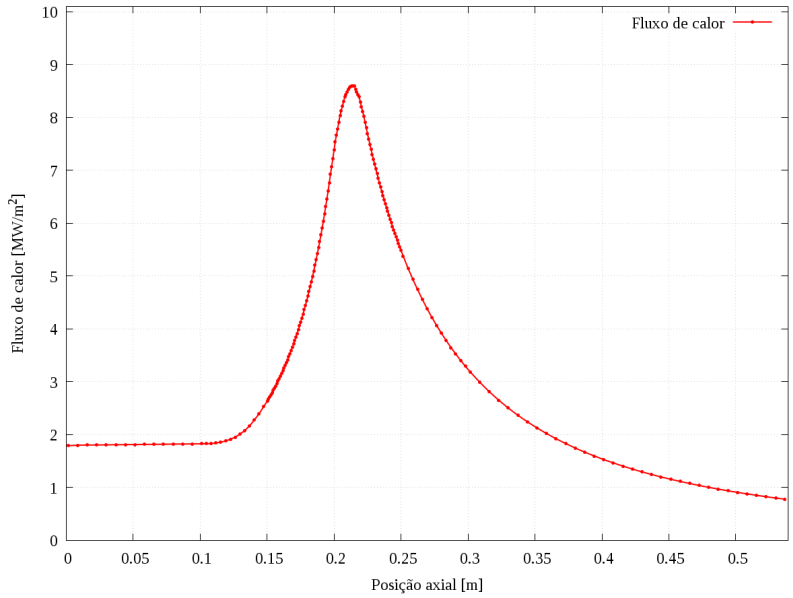
A tendência da melhor troca de calor acontecer na maior redução da altura e largura do canal pode ser explicada devido aos coeficientes de transferência de calor serem inversamente proporcionais ao diâmetro hidráulico. Um grande aumento da transferência de calor acontece quando a área transversal do canal sofre considerável diminuição (INCROPERA *et al.*, 2008).

De uma maneira geral, a perda de carga mostra tendência de aumento a medida que o número de canais cresce para cada otimização, e cai bruscamente ao chegar a 40 canais. É muito importante observar que esta queda brusca é também resultado da diminuição brusca da largura de canal que acontece de 30 para 40 canais, pois a geometria do canal não se mantém fixa ao variarmos o número de canais. Caso a geometria estivesse fixa, seria observado apenas o incremento da perda de carga a cada aumento do número de canais.

É importante também salientar que a perda de carga nos canais é tipicamente entre 5% e 25% da pressão de câmara, conforme Sutton e Biblarz (2011).

A geometria do resultado de 71 canais foi utilizada, pois, apresentou a menor temperatura máxima na parede interna. As figuras 18 e 19 são referentes a esta geometria.

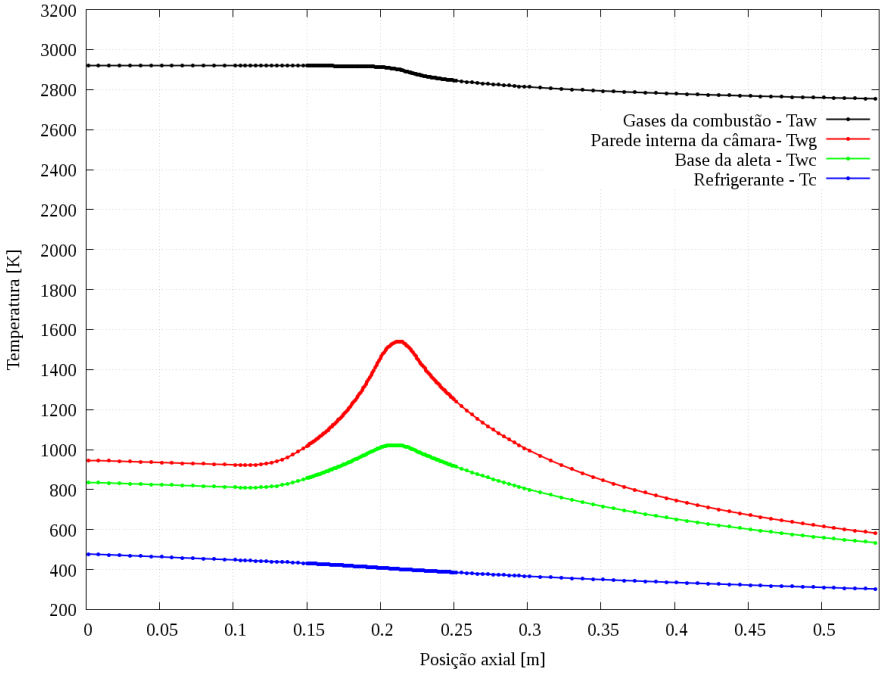
Figura 18 – Fluxo de calor ao longo do eixo longitudinal do motor Ogum



Fonte: Elaborado pelo autor.

O fluxo de calor ao longo do motor Ogum confirma o perfil característico desta curva, pois o maior valor situa-se na garganta do motor.

Figura 19 – Perfis de temperatura obtidos para o melhor resultado do motor Ogum.



Fonte: Elaborado pelo autor.

Os perfis de temperatura obtidos são condizentes com o esperado de um motor foguete, no entanto, neste caso específico, a temperatura da parede interna superou a temperatura de fusão do material, que é de 1400 °C, conforme ASM - Aerospace Specification Metals (2019).

A temperatura final do líquido de arrefecimento também é um parâmetro importante para a avaliar se o excesso de calor não está degradando o combustível antes de sua entrada na câmara. Não foram encontrados dados para um limite admissível no caso do etanol.

Os dados de saída da tabela 11 também mostram que a espessura mínima da parede interna da câmara é bem menor em relação ao mínimo valor admissível, mesmo utilizando o coeficiente de segurança de 1,2.

É importante que a espessura da parede interna seja a mínima possível, como mostra Almeida e Shimote (1999 apud KESSAEV, 1997).

Tabela 11 – Dados de saída para o motor foguete Ogum

Fluxo de calor máximo	8,602 MW/m$^-$2
Temperatura máxima da parede interna T_{wg}	1540,2 K
Perda de carga do combustível	6,617 kPa
Espessura mínima da parede interna	0,12 mm
Iterações necessárias	8
Número de canais	71
Largura do canal	1 mm
Altura do canal	1 mm

Fonte: Elaborado pelo autor.

No trabalho de Dinçer *et al.* (2016), temos os materiais que são comumente utilizados na fabricação de motores foguetes. O material utilizado como entrada para o Ogum foi o Aço Inox 304 que tem um baixo coeficiente de condução de calor se comparado as ligas de cobre comumente utilizadas. Segundo ele, enquanto o material utilizando tem 16,2 W/m.K a liga de cobre Glidcop-Al-15 tem 365 W/m.K.

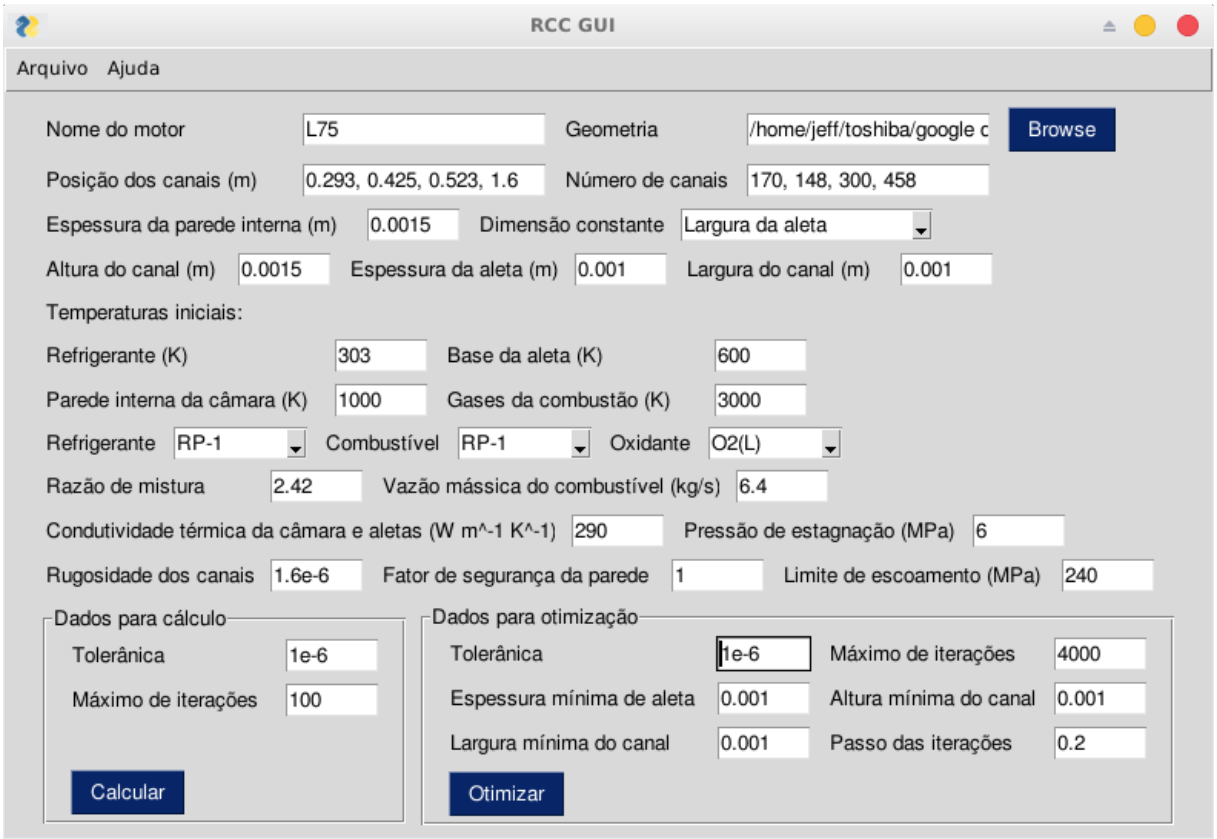
A utilização de propriedades do líquido de arrefecimento como função da temperatura pode melhorar a acurácia dos resultados e a utilização de um material com maior coeficiente de condução certamente diminuirá a temperatura máxima na parede interna. Isto também trará um aumento na temperatura do combustível antes de sua chegada na câmara, reforçando a necessidade da avaliação das temperaturas máximas admissíveis do etanol.

Aliada a outras técnicas como a de resfriamento por filme, a temperatura da parede interna (T_{wg}) pode chegar a valores aceitáveis, onde a fusão do material não ocorre.

5.3 Interface Gráfica

Visando uma melhor interação humano-computador, e para disponibilizar outro modo de utilização da biblioteca, uma interface gráfica foi construída. Ela contém todo o mecanismo utilizado para a computação dos resultados anteriores e necessita das mesmas entradas para seu funcionamento.

Figura 20 – Interface gráfica construída para utilização da biblioteca.



Fonte: Elaborado pelo autor.

A interface é disponibilizada na forma de um repositório no *GitHub*. Mais informações sobre sua instalação e utilização podem ser encontradas em Bezerra (2019).

6 CONCLUSÃO

A principal proposta do presente trabalho é realizar a implementação de um modelo numérico para calcular o perfil de temperaturas em um motor foguete bipropelente e dimensionar os canais de resfriamento regenerativo para o motor em questão. Além disso, foi objetivado escalar modelo para motores com diferentes características, otimizar a geometria dos canais de resfriamento e produzir um *software* finalizado.

A implementação numérica de um modelo de resfriamento regenerativo foi possível na linguagem Python, a partir do modelo de Foltran e Blavier (2018), e a utilização de ferramentas como o CEA para a obtenção das propriedades dos gases de combustão.

A validação da implementação foi realizada utilizando o trabalho de Foltran e Blavier (2018). Foram comparadas a curva de fluxo de calor e os perfis de temperaturas do motor foguete L-75, obtidas numericamente em Foltran e Blavier (2018), e o fluxo de calor, analiticamente em Almeida e Shimote (1999).

A linguagem de uso geral Python foi escolhida também devido a sua capacidade de integração, bastante eficiente ao acoplar o CEA ao *software* desenvolvido. A implementação, na forma de uma biblioteca e uma interface gráfica, garante a possiblidade de reuso do código, de extensão de suas funcionalidades e uma maior facilidade de utilização. O trabalho resultou em um produto finalizado disponibilizado no GitHub, o que permitirá o acesso e uso por demais pesquisadores.

Observa-se que a aplicação do modelo a motores foguete demonstrou a capacidade do programa em lidar com diferentes geometrias ao receber como entrada: arquivos do L-75 e do Ogum, em formato CSV; diferentes propelentes ao aplicar o modelo utilizando RP-1, oxigênio líquido, etanol e óxido nitroso; e, diversas propriedades como temperaturas, condutividade térmica, razão de mistura, pressão de estagnação e limite de escoamento.

Através do uso do gradiente descendente como técnica de otimização, valores de fluxo de calor foram maximizados para o motor o Ogum, resultando em um método relativamente simples para a obtenção de uma melhor geometria.

O resfriamento por filme e a utilização de um material com maior coeficiente de condução no motor Ogum pode resultar em temperaturas admissíveis na parede interna.

Uma melhoria para a biblioteca pode ser feita através do cálculo da troca de calor por radiação e a utilização de mais propriedades em função da temperatura e do combustível utilizado, no caso do motor Ogum. Estas melhorias podem trazer uma maior acurácia nos

resultados obtidos. Outra sugestão de trabalho futuro é a validação experimental, que pode ser feita para comparação dos resultados obtidos com os dados experimentais. Este trabalho pode trazer uma melhoria na biblioteca e no modelo ao ser implementado.

REFERÊNCIAS

ALMEIDA, D. S.; SHIMOTE, W. K. Selection of materials for combustion chamber of liquid propelant rocket engine. **XV Congresso Brasileiro de Engenharia Mecânica**, 1999.

ASM - AEROSPACE SPECIFICATION METALS. **AISI Type 304 Stainless Steel**. 2019. Disponível em: <http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=mq304a>. Acesso em: 08 ago. 2019.

BARTZ, D. R. A simple equation for rapid estimation of rocket nozzle convective heat transfer coefficients. **Jet Propulsion**, 1957.

BEZERRA, J. **RCC GUI - Rocket Cooling Calculator**. GitHub, 2019. Disponível em: <https://github.com/jeffersonmsb/rocket-cooling-calculator>. Acesso em: 27 ago. 2019.

BLAVIER, J. P. **Projeto térmico de um motor foguete com refrigeração regenerativa**. Dissertação (Trabalho de Conclusão de Curso (Bacharelado em Engenharia Mecânica)) — Universidade Positivo, 2016.

BORGNAKKE, C.; SONNTAG, R. E. **Fundamentals of Thermodynamics**. [S.l.]: Wiley, 2014. Sétima Edição.

BOWLER, T. **A nova corrida espacial - que agora é disputada por empresas**. British Broadcasting Corporation, 2019. Disponível em: <https://www.bbc.com/portuguese/geral-41275401>. Acesso em: 27 ago. 2019.

BOYSAN, M. E. **Analysis of regenerative cooling in liquid propellant rocket engines**. Dissertação (M. Sc. thesis) — Ankara: Middle East Technical University, 2008.

DINçER, O.; PEHLIVANOĞLU, M. K.; DERICIOĞLU, A. F. High strength copper alloys for extreme temperature conditions. **Uluslararası Metalurji ve Malzeme Kongresi**, 2016.

ENGINEERING TOOLBOX. **Ethanol - Thermophysical properties**. 2019. Disponível em: <https://www.engineeringtoolbox.com/ethanol-ethyl-alcohol-properties-C2H6O-d_2027.html>. Acesso em: 08 ago. 2019.

ESPINOZA, N. **pyCEA**. 2019. Disponível em: <https://github.com/nespinoza/pyCEA>. Acesso em: 15 jul. 2019.

FOLTRAN, A. C.; BLAVIER, J. P. Numerical model of heat transfer in a regenaratively cooled rocket engine. **I Congresso Aeroespacial Brasileiro**, 2018.

FOX, R. W.; MCDONALD, A. T.; J, P. P.; LEYLEGIAN, J. C. **Introdução à Mecânica dos Fluidos**. Oitava edição. [S.l.]: LTC, 2014.

GORDON, S.; MCBRIDE, J. **Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Application**. [S.l.]: , 1994. RP1311.

HIBBELER, R. C. **Resistência dos Materiais**. Quinta edição. [S.l.]: Pearson Prentice Hall, 2009.

HILL, P. G.; PETERSON, C. R. **Mechanics and Thermodynamics of Propulsion**. [S.l.]: Pearson, 2014. Segunda Edição.

INCROPERA, F. P.; DEWITT, D. P.; BERGMAN, T. L.; S, L. A. **Fundamentos de Transferência de Calor e Massa**. [S.l.]: LTC, 2008. v. 6 ed.

KESSAEV, J. V. **Theory and calculation of liquid propellant engines**. [S.l.]: Moscow Aviation Institute, 1997.

LUENBERGER, D. G.; YE, Y. **Linear and Nonlinear Programming**. [S.l.]: Springer, 2008. v. 3 ed.

ROHATGI, A. **WebPlotDigitizer**. 2019. Disponível em: <https://apps.automeris.io/wpd/>. Acesso em: 15 jul. 2019.

RUGGIERO, M. A. G.; LOPES, V. L. R. **Cálculo Numérico: Aspectos teóricos e computacionais**. Segunda edição. [S.l.]: Pearson Prentice Hall, 1997. v. 2.

SCIPY. **SciPy**. 2019. Disponível em: <https://www.scipy.org/>. Acesso em: 15 jul. 2019.

STEWART, J. **Cálculo**. Quinta edição. [S.l.]: Cengage Learning, 2010. v. 2.

SUTTON, G. P.; BIBLARZ, O. **Rocket Propulsion Elements**. [S.l.]: Addison-Wesley, 2011. v. 1993.

7 BIBLIOTECA ROCKET COOLING CALCULATOR

```
'''
Biblioteca para calculo de resfriamento regenerativa em
   motores foguetes bipropelentes
Jefferson Bezerra
https://github.com/jeffersonmsb/rocket-cooling-calculator
'''

import csv
import numpy as np
import math
import pyCEA
from scipy import optimize
import os
import subprocess

def geometry(data_in, data_out):
    with open(data_in['geometry_path']) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        data_out['geometry'] = list(csv_reader)

        data_out['size'] = len(data_out['geometry'])

        #Buscar geometria da garganta
        Rt = float(data_out['geometry'][0][1])
        zt = float(data_out['geometry'][0][0])
        for row in data_out['geometry']:
            if float(row[1]) < Rt:
                Rt = float(row[1])
                zt = float(row[0])
        data_out['Rt'] = Rt
```

```
30          data_out['zt'] = zt
31          data_out['At'] = np.pi*np.power(Rt,2)
32
33          #C lculo das raz es de  rea
34          data_out['r1'] = []
35          data_out['r2'] = []
36          data_out['r3'] = []
37          data_out['Ae'] = []
38          data_out['Ae/At'] = []
39          data_out['z'] = []
40          data_out['N'] = []
41          data_out['CCH'] = []
42          data_out['CCW'] = []
43          data_out['FT'] = []
44          n = 0
45
46          for row in data_out['geometry']:
47              A = np.pi*np.power(float(row[1]),2)
48              data_out['r1'].append(float(row[1]))
49              r2 = float(row[1]) + data_in['IWT']
50              data_out['r2'].append(r2)
51              data_out['r3'].append(float(row[1]) + data_in['
                    IWT'] + data_in['CCH'])
52              data_out['Ae'].append(A)
53              data_out['Ae/At'].append(A/data_out['At'])
54              data_out['z'].append(float(row[0]))
55
56              if float(row[0]) > data_in['channel_number'][n
                    ][0]:
57                  n = n + 1
58
59              N = data_in['channel_number'][n][1]
```

```
60              data_out['N'].append(N)
61
62              data_out['CCH'].append(data_in['CCH'])
63
64              if data_in['dim_constant'] == 'FT':
65                  data_out['FT'].append(data_in['FT'])
66                  aux = (2*np.pi*r2)/N - data_in['FT']
67                  if aux <= 0:
68                      data_out['error_code'] = 1
69                      return
70                  data_out['CCW'].append(aux)
71              else:
72                  data_out['CCW'].append(data_in['CCW'])
73                  aux = (2*np.pi*r2)/N - data_in['CCW']
74                  data_out['FT'].append(aux)
75
76          data_out['L'] = []
77          for i in range(0, data_out['size']):
78              if(i==0):
79                  A = 0.5*(data_out['z'][i+1]+data_out['z'][i
                        ]) - data_out['z'][i]
80                  B = 0.5*(data_out['r1'][i+1]+data_out['r1'
                        ][i]) - data_out['r1'][i]
81                  data_out['L'].append(math.sqrt(A**2 + B**2)
                        )
82              else:
83                  if(i!=(data_out['size']-1)):
84                      A = 0.5*(data_out['z'][i+1]+data_out['z
                            '][i]) - 0.5*(data_out['z'][i]+
                            data_out['z'][i-1])
85                      B = 0.5*(data_out['r1'][i+1]+data_out['
                            r1'][i]) - 0.5*(data_out['r1'][i]+
```

```
                        data_out['r1'][i-1])
86                      data_out['L'].append(math.sqrt(A**2 + B
                        **2))
87                  else:
88                      A = data_out['z'][i] - 0.5*(data_out['z
                        '][i]+data_out['z'][i-1])
89                      B = data_out['r1'][i] - 0.5*(data_out['
                        r1'][i]+data_out['r1'][i-1])
90                      data_out['L'].append(math.sqrt(A**2 + B
                        **2))
91      data_out['error_code'] = 0
92
93  def coolant_prop(coolant_name, prop_name, temperature):
94      if coolant_name == 'RP-1':
95          if temperature > 800:
96              temperature = 800
97          if temperature < 300:
98              temperature = 300
99
100         if prop_name == 'ro':
101             return 820
102         if prop_name == 'cp':
103             return -2.82649e-3*temperature**2.0 + 6.77751e0
                    *temperature - 2.45234e1 #BOYSAN
104         if prop_name == 'k':
105             return 9.64e-8*temperature**2-2.95e-4*
                    temperature+0.261 #BOYSAN
106         if prop_name == 'mi':
107             return -1.46e-11*temperature**3+3.22e-8*
                    temperature**2-2.39e-5*temperature+6E-3 #
                    BOYSAN
108     if coolant_name == 'C2H5OH(L)':
```

```
109         if prop_name == 'ro':
110             return 785.3
111         if prop_name == 'cp':
112             return 2570
113         if prop_name == 'k':
114             return 0.167
115         if prop_name == 'mi':
116             return 1.36e-3
117     else:
118         print('Coolant proprieties not found')
119         return -1
120 
121 def create_prop(data_in, data_out):
122     data_out['Tc'] = data_out['size']*[data_in['Tc_primary']]
123     data_out['Twg'] = data_out['size']*[data_in['Twg_primary']]
124     data_out['Twc'] = data_out['size']*[data_in['Twc_primary']]
125     data_out['Taw'] = data_out['size']*[data_in['Taw_primary']]
126 
127     data_out['cp_c'] = data_out['size']*[None]
128     data_out['k_c'] = data_out['size']*[None]
129     data_out['mi_c'] = data_out['size']*[None]
130     data_out['Pr_c'] = data_out['size']*[None]
131     data_out['gama'] = data_out['size']*[None]
132     data_out['M'] = data_out['size']*[None]
133     data_out['cp'] = data_out['size']*[None]
134     data_out['R'] = data_out['size']*[None]
135     data_out['h_g'] = data_out['size']*[None]
136     data_out['Re_c'] = data_out['size']*[None]
```

```
137     data_out['D_h'] = data_out['size']*[None]
138     data_out['mi_s'] = data_out['size']*[None]
139     data_out['h_c'] = data_out['size']*[None]
140     data_out['Aa'] = data_out['size']*[None]
141     data_out['Atotal'] = data_out['size']*[None]
142     data_out['m'] = data_out['size']*[None]
143     data_out['eta_f'] = data_out['size']*[None]
144     data_out['eta_o'] = data_out['size']*[None]
145     data_out['R_c'] = data_out['size']*[None]
146     data_out['R_g'] = data_out['size']*[None]
147     data_out['R_w'] = data_out['size']*[None]
148     data_out['q'] = data_out['size']*[None]
149     data_out['Q'] = data_out['size']*[None]
150     data_out['f'] = data_out['size']*[None]
151     data_out['ro'] = data_out['size']*[None]
152     data_out['V_c'] = data_out['size']*[None]
153     data_out['hl'] = data_out['size']*[None]
154     data_out['deltap'] = data_out['size']*[None]
155     data_out['T_static'] = data_out['size']*[None]
156     data_out['p_static'] = data_out['size']*[6000000]
157
158 def calc_prop(data_in, data_out):
159
160     data_in['p0_pyCEA'] = data_in['p0']/1e5 #Convers o de
            [Pa] para [bar]
161     pyCEA.calcPropStagnationCEA(data_in['p0_pyCEA'],data_in
            ['fuel'], data_in['oxidizer'],data_in['of'], data_in
            ['motor_name'])
162     T0 = pyCEA.readPropStagnationCEA('t', data_in['p0_pyCEA
            '], data_in['fuel'], data_in['oxidizer'], data_in['
            of'], data_in['motor_name'])
```

```
163     cp0 = pyCEA.readPropStagnationCEA('cp', data_in['
          p0_pyCEA'], data_in['fuel'], data_in['oxidizer'],
          data_in['of'], data_in['motor_name'])
164     Pr0 = pyCEA.readPropStagnationCEA('pr', data_in['
          p0_pyCEA'], data_in['fuel'], data_in['oxidizer'],
          data_in['of'], data_in['motor_name'])
165     mi0 = pyCEA.readPropStagnationCEA('mi', data_in['
          p0_pyCEA'], data_in['fuel'], data_in['oxidizer'],
          data_in['of'], data_in['motor_name'])
166
167     Tc1 = data_in['Tc_primary']
168     IWT = data_in['IWT']
169     k_w = data_in['k_w']
170     mponto_c = data_in['m._c']
171     e = data_in['e']
172     p0 = data_in['p0']
173
174     Re_c = data_out['Re_c']
175     N = data_out['N']
176     mi_c = data_out['mi_c']
177     CCW = data_out['CCW']
178     FT = data_out['FT']
179     D_h = data_out['D_h']
180     mi_s = data_out['mi_s']
181     Tc = data_out['Tc']
182     Twg = data_out['Twg']
183     Twc = data_out['Twc']
184     Taw = data_out['Taw']
185     h_c = data_out['h_c']
186     k_c = data_out['k_c']
187     Pr_c = data_out['Pr_c']
188     Aa = data_out['Aa']
```

```
189     L = data_out['L']
190     r1 = data_out['r1']
191     r2 = data_out['r2']
192     r3 = data_out['r3']
193     Atotal = data_out['Atotal']
194     m = data_out['m']
195     eta_f = data_out['eta_f']
196     eta_o = data_out['eta_o']
197     R_c = data_out['R_c']
198     R_g = data_out['R_g']
199     R_w = data_out['R_w']
200     h_g = data_out['h_g']
201     q = data_out['q']
202     Q = data_out['Q']
203     cp_c = data_out['cp_c']
204     k_c = data_out['k_c']
205     f = data_out['f']
206     ro = data_out['ro']
207     V_c = data_out['V_c']
208     hl = data_out['hl']
209     deltap = data_out['deltap']
210     T_static = data_out['T_static']
211     p_static = data_out['p_static']
212     gama = data_out['gama']
213     M = data_out['M']
214     CCH = data_out['CCH']
215
216     data_out['p_drop'] = 0
217
218     def f_mach(M):
219         A = 2/(data_out['gama'][i]+1)
220         B = 1+(((data_out['gama'][i]-1)/2)*(M**2))
```

```
221         C = (data_out['gama'][i]+1)/(data_out['gama'][i]-1)
222         D = (data_out['Ae/At'][i]*M)**2
223         return ( (A*B)**C-D )
224 
225     def f_coolebrook(f):
226         return (1/(-2*math.log(((e/D_h[i])/3.7)+(2.51/(Re_c
                [i]*f**0.5)), 10))**2-f)
227 
228     for i in reversed(range(0,data_out['size'])):
229         cp_c[i] = coolant_prop(data_in['coolant'], 'cp', Tc
                [i])
230         k_c[i] = coolant_prop(data_in['coolant'], 'k',
                data_out['Tc'][i])
231         data_out['mi_c'][i] = coolant_prop(data_in['coolant
                '], 'mi', data_out['Tc'][i])
232         data_out['Pr_c'][i] = data_out['cp_c'][i]*data_out[
                'mi_c'][i]/data_out['k_c'][i]
233 
234         pyCEA.calcPropCEA(data_out['Taw'][i] , data_in['
                p0_pyCEA'], data_in['fuel'], data_in['oxidizer'
                ], data_in['of'], data_in['motor_name'])
235         data_out['cp'][i] = pyCEA.readPropCEA('cp',
                data_out['Taw'][i], data_in['p0_pyCEA'], data_in
                ['fuel'], data_in['oxidizer'], data_in['of'],
                data_in['motor_name'])
236         #data_out['cp'][i] = -5.84399e-05*data_out['Taw'][i
                ]**2.0 + 4.23454e-01*data_out['Taw'][i] +
                1.29256e+03
237         data_out['gama'][i] = 1.23854e-8*data_out['Taw'][i
                ]**2 - 8.09028e-5*data_out['Taw'][i] + 1.34563
238         #Gama para o L-75
```

```
239         #data_out['gama'][i] = pyCEA.readPropCEA('gama',
               data_out['Taw'][i], data_in['p0_pyCEA'], data_in
               ['fuel'], data_in['oxidizer'], data_in['of'],
               data_in['motor_name'])
240         data_out['R'][i] = (data_out['cp'][i]*(1 - 1/
               data_out['gama'][i]))
241         mponto = data_in['p0']*data_out['At']*((data_out['
               gama'][i]/(data_out['R'][i]*T0))*(2/(data_out['
               gama'][i]+1))**((data_out['gama'][i]+1)/(
               data_out['gama'][i]-1)))**0.5
242         c = (data_in['p0']*data_out['At'])/mponto
243         if(data_out['z'][i] > data_out['zt']):
244             a = 1
245             b = 25
246         else:
247             a = 0
248             b = 1
249         data_out['M'][i] = optimize.bisect(f_mach, a, b,
               rtol=8.881784197001252e-16)
250         aux1 = 1 + ((data_out['gama'][i]-1)/2)*data_out['M'
               ][i]**2
251         sigma = ((data_out['Twg'][i]/(2*T0))*aux1+0.5  )
               **-0.68  *  aux1**-0.12
252         data_out['h_g'][i] = (  0.026  *  ((mi0/(2*data_out
               ['Rt']))**0.2)  *  (cp0/(Pr0**0.6))  *  (data_in
               ['p0']/c)**0.8  *  (data_out['At']/data_out['Ae'
               ][i])**0.9  *  sigma  )
253
254         D_h[i] = (4*CCW[i]*CCH[i])/(2*(CCW[i]+CCH[i]))
255         Re_c[i] = (4*mponto)/(N[i]*mi_c[i]*2*(CCW[i]+CCH[i
               ]))
256
```

```
257         mi_s[i] = coolant_prop(data_in['coolant'], 'mi',
               Twc[i])
258         h_c[i] = ((k_c[i]/D_h[i]) * 0.027 * Re_c[i]**0.8 *
               Pr_c[i]**(1/3) * (mi_c[i]/mi_s[i])**0.14 )
259
260         Aa[i] = (2*CCH[i]*L[i])
261         Atotal[i] = (N[i]*Aa[i] + L[i]*(2*math.pi*r2[i]-N[i
               ]*FT[i]))
262         m[i] = math.sqrt((2*h_c[i])/(k_c[i]*FT[i]))
263         eta_f[i] = (math.tanh(m[i]*CCH[i])/(m[i]*CCH[i]))
264         eta_o[i] = 1-((N[i]*Aa[i]*(1-eta_f[i])) / Atotal[i
               ])
265
266         R_g[i] = (1/(2*math.pi*r1[i]*L[i]*h_g[i]))
267
268         R_w[i] = (math.log(r2[i]/r1[i]) / (2*math.pi*L[i]*
               k_w))
269
270         R_c[i] = (1 / (eta_o[i]*h_c[i]*Atotal[i]))
271
272         q[i] = ((Taw[i] - Tc[i]) / (R_g[i] + R_w[i] + R_c[i
               ]))
273         Q[i] =  ( q[i]/(2*math.pi*r1[i]*L[i])/1000000 )
274
275         aux = 0.5*(data_out['gama'][i] - 1)*data_out['M'][i
               ]**2
276         Taw[i] = (T0 * ((1 + Pr0**(1/3)*aux) / (1 + aux)))
277         Twg[i] = -R_g[i]*q[i]+Taw[i]
278         Twc[i] = -q[i]*(R_g[i]+R_w[i])+Taw[i]
279
280         lista = reversed(range( i,data_out['size']))
281         Tc1 = 303
```

```
282         for j in lista:
283             Tc2 = (q[j] / (mponto_c*cp_c[j])) + Tc1
284             Tc[j] = (Tc2+Tc1)/2
285             Tc1 = Tc2
286 
287 
288         p_static[i] = p0*(1+((gama[i]-1)/2)*M[i]**2)**-(
                gama[i]/(gama[i]-1))
289 
290         #C lculo da perda de carga
291         f[i] = optimize.bisect(f_coolebrook, 0.00001, 2,
                rtol=8.881784197001252e-16)
292         ro[i] = coolant_prop(data_in['coolant'], 'ro', Tc[i
                ])
293         V_c[i] = mponto_c/(ro[i]*CCH[i]*CCW[i]*N[i])
294         hl[i] = f[i]*((L[i]/D_h[i])/(V_c[i]**2/2))
295         deltap[i] = ro[i]*hl[i]*N[i]
296         data_out['p_drop'] += deltap[i]
297 
298         #C lculo da temperatura est tica e press o
                est tica
299         T_static[i] = T0*(1+((gama[i]-1)/2)*M[i]**2)**-1
300 
301 def iteration(data_in , data_out):
302     geometry(data_in, data_out)
303     if data_out['error_code'] != 0:
304         print('CCW <= 0')
305         return
306     create_prop(data_in, data_out)
307     for i in range(0,data_in['max_iterations']):
308         print('Iteration {}'.format(i+1))
309         calc_prop(data_in, data_out)
```

```
310         if i==0:
311             Tc_0 = sum(data_out['Q'])
312             Twg_0 = sum(data_out['Twg'])
313             Twc_0 = sum(data_out['Twc'])
314             Taw_0 = sum(data_out['Taw'])
315
316             Tc_prev = Tc_0
317             Twg_prev = Twg_0
318             Twc_prev = Twc_0
319             Taw_prev = Taw_0
320         else:
321             Tc = sum(data_out['Q'])
322             Twg = sum(data_out['Twg'])
323             Twc = sum(data_out['Twc'])
324             Taw = sum(data_out['Taw'])
325
326             Tc_L1 = abs(Tc-Tc_prev)/Tc_0
327             Twg_L1 = abs(Twg-Twg_prev)/Twg_0
328             Twc_L1 = abs(Twc-Twc_prev)/Twc_0
329             Taw_L1 = abs(Taw-Taw_prev)/Taw_0
330
331             Tc_prev = Tc
332             Twg_prev = Twg
333             Twc_prev = Twc
334             Taw_prev = Taw
335
336             if Tc_L1 <= data_in['tol'] and Twg_L1 <=
                   data_in['tol'] and Twc_L1 <= data_in['tol']
                   and Taw_L1 <= data_in['tol']:
337                 break
338     print('Total Iteration Temperature: ' + str(i+1))
339
```

```
340
341 def optimize_channel2(data_in, data_out):
342     flag1 = False
343     flag2 = False
344     if data_in['dim_constant'] == 'FT':
345         dim_const = 'FT'
346         dim_var = 'CCW'
347     else:
348         dim_const = 'CCW'
349         dim_var = 'FT'
350
351     geometry(data_in, data_out)
352     m = 0
353     for i in range(0, data_out['size']):
354         if data_out['r2'][i] < data_out['r2'][m]:
355             m = i
356     dim_max = (2*np.pi*data_out['r2'][m])/data_out['N'][m]
        - data_in[dim_var + '_min']
357
358     if dim_max-data_in[dim_const + '_min'] <= 0:
359         print('Maior dimens o geom trica    menor que
            dimens o m nima.')
360         return
361
362     dim = (dim_max+data_in[dim_const + '_min'])/2
363     x = np.array([data_in['CCH'] , dim])
364     data_in[dim_const] = dim
365     iteration(data_in, data_out)
366     Q = max(data_out['Q'])
367     Q_prev = Q
368     Q0 = Q
369
```

```
370     w = data_in['w']
371
372     for opt in range(0,data_in['max_iterations_opt']):
373         grad = np.gradient(x)
374         xn = x - w*grad
375
376         if xn[0] <= data_in['CCH_min'] and flag1 == False:
377             flag1 = True
378             print('CCH_min')
379         if (xn[1] <= data_in[dim_const+'_min'] or xn[1] >=
               dim_max) and flag2 == False:
380             flag2 = True
381             print(dim_const+' min or max')
382
383         if flag1 == True:
384             xn[0] = x[0]
385
386         if flag2 == True:
387             xn[1] = x[1]
388
389         data_in['CCH'] = xn[0]
390         data_in[dim_const] = xn[1]
391         iteration(data_in, data_out)
392         Q = max(data_out['Q'])
393         if Q-Q_prev < 0:
394             w=w*-1
395             print(w)
396             continue
397
398         x = xn
399         print('Opt #{} Q:{} CCH:{} {}:{}'.format(opt, Q, x
               [0], dim_const, x[1]))
```

```
400
401             Q_diff = abs(Q-Q_prev)/Q0
402             Q_prev = Q
403
404             if Q_diff <= data_in['tol_opt']:
405                 break
406
407 def plot(data_out):
408     data = []
409     for i in range(0,data_out['size']):
410         data_row = [ data_out['z'][i], data_out['Q'][i],
                data_out['Taw'][i], data_out['Twg'][i], data_out
                ['Twc'][i], data_out['Tc'][i]]
411         data.append(data_row)
412
413     with open('rcc_plot_data.csv', mode='w', encoding='utf
            -8') as data_file:
414         csv_writer = csv.writer(data_file, delimiter=',')
415         csv_writer.writerows(data)
416
417     p = subprocess.Popen("gnuplot \'rcc_plot_config.gnu\'",
            shell = True)
418     os.waitpid(p.pid, 0)
419     '''p2 = subprocess.Popen("ristretto \'temps.png\'",
            shell = True)
420     os.waitpid(p2.pid, 1)'''
421
422 def calc_wall_thickness(p, path, sigmae, n=1):
423
424     with open(path) as csv_file:
425         csv_reader = csv.reader(csv_file, delimiter=',')
426         r = float(list(csv_reader)[0][1])
```

```
427
428     def f_t(t):
429         sigma1 = (p*r)/t
430         sigma2 = (p*r)/(2*t)
431         return math.sqrt(sigma1**2-sigma1*sigma2+sigma2**2)
                -sigmae
432
433     return (optimize.bisect(f_t, 1e-8, 1, rtol
           =8.881784197001252e-16))*n
```

8 BIBLIOTECA PYCEA MODIFICADA

```
# -*- coding: utf-8 -*-
import numpy as np
import subprocess,os

def checkCEA(compiler='gfortran'):
    """
    Check that all the fortran CEA codes have their
    corresponding executables:
    """
    cwd = os.getcwd()
    os.chdir('CEA+FORTRAN')
    if not os.path.exists('FCEA2'):
        subprocess.Popen(compiler+' cea2.f -o FCEA2',\
                         shell = True).wait()
    if not os.path.exists('b1b2b3'):
        subprocess.Popen(compiler+' b1b2b3.f -o b1b2b3',\
                         shell = True).wait()
    if not os.path.exists('syntax'):
        subprocess.Popen(compiler+' syntax.f -o syntax',\
                         shell = True).wait()
    if not os.path.exists('thermo.lib'):
        p = subprocess.Popen(['./FCEA2'], stdout=subprocess
            .PIPE, \
                             stdin=subprocess.PIPE, stderr=
                                subprocess.PIPE)
        stdout_data = p.communicate(input='thermo')[0]
    if not os.path.exists('trans.lib'):
        p = subprocess.Popen(['./FCEA2'], stdout=subprocess
            .PIPE, \
```

```
27                                  stdin=subprocess.PIPE, stderr=
                                      subprocess.PIPE)
28          stdout_data = p.communicate(input='trans')[0]
29
30      os.chdir(cwd)
31      #print('CEA up and running!')
32
33  def runCEA(filename):
34      filename_no_extension = (filename.split('.inp')[0]).
          split('/')[1]
35      subprocess.Popen('cp '+filename+' CEA+FORTRAN/.',shell
          = True).wait()
36      cwd = os.getcwd()
37      os.chdir('CEA+FORTRAN')
38      p = subprocess.Popen(['./FCEA2'], stdout=subprocess.
          PIPE, stdin=subprocess.PIPE, stderr=subprocess.PIPE)
39      stdout_data = p.communicate(input=filename_no_extension
          .encode())[0]
40      subprocess.Popen('mv '+filename_no_extension+'.out ../
          CEAoutput/.',shell = True).wait()
41      subprocess.Popen('rm '+filename_no_extension+'.inp',
          shell = True).wait()
42      os.chdir(cwd)
43
44  def read_abundances(filename):
45      """
46      This code reads abundances from a file containing Z,
          name, A(X) and error on this
47      number, where A(X) = log N(X)/N(H) + 12, and N(X) is
          the number of atoms of element
48      X. The code returns:
49
```

```
50          Z               Atomic number of the element X
51          Name            Name of the element X
52          10**A           10 to the (log N(X)/N(H) - 12), where N
               (X) is the number of atoms of element X,
53                          and N(H) is the number of hydrogen
                               atoms (thus, this takes 1e12 for
                               hydrogen
54                          by definition).
55      """
56      Z = np.array([])
57      name = np.array([])
58      logN = np.array([])
59
60      f = open(filename,'r')
61      while True:
62          line = f.readline()
63          if line != '':
64              if line[0] != '#':
65                  data = line.split()
66                  Z = np.append(Z,np.double(data[0]))
67                  name = np.append(name,data[1])
68                  logN = np.append(logN,np.double(data[2]))
69          else:
70              break
71      f.close()
72      return Z,name,10**(logN)
73
74  def calcCEA(T,P,name,N,only_consider_these,prefix='
       benchmark',ions = False):
75      """
76      This function generates input files for CEA, runs them,
            and
```

```
77      spits out the equilibrium compositions:
78      """
79
80      if not os.path.exists('CEAoutput'):
81          os.mkdir('CEAoutput')
82      f = open('CEAoutput/'+prefix+'_'+str(T)+'_'+str(P)+'.
           inp','w')
83      f.write('# problem dataset: transpec\n')
84      if ions:
85          f.write('problem tp ions\n')
86      else:
87          f.write('problem tp\n')
88      f.write('  p(bar) = '+str(P)+'\n')
89      f.write('  t(k) = '+str(T)+'\n')
90      f.write('reac\n')
91      for i in range(len(name)):
92          f.write('  na '+name[i]+' moles = '+str(N[i]).upper
               ()+'\n')
93      f.write('only '+only_consider_these)
94      f.write('output calories siunits trace 1e-20\n')
95      f.write('end')
96      f.close()
97      runCEA('CEAoutput/'+prefix+'_'+str(T)+'_'+str(P)+'.inp'
           )
98
99  def readCEA(T,P,prefix='benchmark'):
100     f = open('CEAoutput/'+prefix+'_'+str(T)+'_'+str(P)+'.
           out','r')
101     species = []
102     moles = []
103     while True:
104         line = f.readline()
```

```
105         if 'MOLE FRACTIONS' in line:
106             f.readline()
107             break
108         if line[:21] == ' CALCULATIONS STOPPED' or line ==
               '':
109             f.close()
110             return species,moles
111     while True:
112         line = f.readline()
113         vec = line.split()
114         if len(vec) == 0:
115             break
116         elif line[:21] == ' CALCULATIONS STOPPED':
117             break
118         else:
119             if '*' not in vec[1]:
120                 species.append(vec[0])
121                 if '-' in vec[1]:
122                     num,exp = vec[1].split('-')
123                     moles.append(np.double(num+'e-'+exp))
124                 else:
125                     moles.append(np.double(vec[1]))
126     f.close()
127     return species,moles
128
129 def check_elements(name,only_consider_these):
130     idx = []
131     for i in range(len(name)):
132         c_name = name[i]
133         length = len(c_name)
134         for j in range(len(only_consider_these)-length):
```

```
135             if only_consider_these[j:j+length] == c_name
                   and only_consider_these[j-1]!='(' \
136                                 and only_consider_these[j-1]!='
                                       I':
137                  if length == 1:
138                      if not only_consider_these[j+1] ==
                            only_consider_these[j+1].lower():
139                          if c_name == 'C':
140                              if only_consider_these[j+1] ==
                                    'L':
141                                  if only_consider_these[j+1:
                                        j+3] == 'Li' or\
142                                      only_consider_these[j+1:
                                            j+3] == 'La' or\
143                                      only_consider_these[j+1:
                                            j+3] == 'Lu':
144                                      idx.append(i)
145                                      break
146                              else:
147                                  idx.append(i)
148                                  break
149                          else:
150                              idx.append(i)
151                              break
152                      else:
153                          if only_consider_these[j+1] == ' ':
154                              idx.append(i)
155                              break
156                  else:
157                      idx.append(i)
158                      break
159     return idx
```

```
160
161 def readPropCEA(prop,t,p,fuel,oxid,of,prefix='benchmark'):
162     filename = 'CEAoutput/'+prefix+'_'+str(t)+'_'+str(p)+'_
        '+str(fuel)+'_'+oxid+'_'+str(of)+'.out'
163     filename = filename.replace('(','').replace(')','')
164     f = open(filename,'r')
165
166     if prop == 'gama':
167         while True:
168             line = f.readline()
169             if 'GAMMAs' in line:
170                 gama = float(line.split(' ')[-1][:-1])
171                 break
172         f.close()
173         return gama
174     if prop == 'cp':
175         while True:
176             line = f.readline()
177             if 'WITH FROZEN REACTIONS' in line:
178                 f.readline()
179                 line = f.readline()
180                 cp = round(float(line.split(' ')[-1][:-1])
                    *1000,1)
181                 break
182         f.close()
183         return cp
184
185 def calcPropCEA(t,p,fuel,oxid,of,prefix='benchmark'):
186     filename = 'CEAoutput/'+prefix+'_'+str(t)+'_'+str(p)+'_
        '+str(fuel)+'_'+oxid+'_'+str(of)+'.inp'
187     filename = filename.replace('(','').replace(')','')
188
```

```
189     if os.path.isfile(filename):
190         return
191 
192     if not os.path.exists('CEAoutput'):
193         os.mkdir('CEAoutput')
194     f = open(filename,'w')
195     f.write('problem case=1 tp\n')
196     f.write('  p(bar) = '+str(p)+'\n')
197     f.write('  t(k) = '+str(t)+'\n')
198     f.write('reac\n')
199     f.write('  fuel='+str(fuel)+' wt=1\n')
200     f.write('  oxid='+str(oxid)+' wt='+str(of)+'\n')
201     f.write('output\n')
202     f.write('  siunits short, transport\n')
203     f.write('  t gam cp\n')
204     f.write('end\n')
205     f.close()
206     runCEA(filename)
207 
208 def calcPropStagnationCEA(p,fuel,oxid,of,prefix='benchmark'
    ):
209     filename = 'CEAoutput/'+prefix+'_'+str(p)+'_'+str(fuel)
        +'_'+oxid+'_'+str(of)+'_rkt.inp'
210     filename = filename.replace('(','').replace(')','')
211 
212     if os.path.isfile(filename):
213         return
214 
215     if not os.path.exists('CEAoutput'):
216         os.mkdir('CEAoutput')
217     f = open(filename,'w')
218     f.write('problem    o/f='+str(of)+',\n')
```

```
219     f.write('    rocket  fac   ac/at=1  tcest,k=3800\n')
220     f.write('  p,bar='+str(p)+',\n')
221     f.write('react  \n')
222     f.write('  fuel='+str(fuel)+' \n')
223     f.write('  oxid='+str(oxid)+' \n')
224     f.write('output  transport \n')
225     f.write('end\n')
226     f.close()
227     runCEA(filename)
228 
229 def readPropStagnationCEA(prop,p,fuel,oxid,of,prefix='
    benchmark'):
230     filename = 'CEAoutput/'+prefix+'_'+str(p)+'_'+str(fuel)
        +'_'+oxid+'_'+str(of)+'_rkt.out'
231     filename = filename.replace('(','').replace(')','')
232     f = open(filename,'r')
233 
234     if prop == 't':
235         while True:
236             line = f.readline()
237             if 'T, K' in line:
238                 t = float(line.split('  ')[-3])
239                 break
240         f.close()
241         return t
242     if prop == 'cp':
243         while True:
244             line = f.readline()
245             if 'WITH FROZEN REACTIONS' in line:
246                 f.readline()
247                 line = f.readline()
```

```
248                 cp = round(float(line.split('   ')[-3])
                       *1000,1)
249                 break
250         f.close()
251         return cp
252     if prop == 'pr':
253         while True:
254             line = f.readline()
255             if 'WITH FROZEN REACTIONS' in line:
256                 f.readline()
257                 f.readline()
258                 f.readline()
259                 line = f.readline()
260                 pr = float(line.split('   ')[-3])
261                 break
262         f.close()
263         return pr
264     if prop == 'mi':
265         while True:
266             line = f.readline()
267             if 'VISC,MILLIPOISE' in line:
268                 mi = float(line.split('   ')[-3])/1000
269                 break
270         f.close()
271         return mi
272
273 checkCEA()
```

9 CÓDIGO PRINCIPAL PARA O MOTOR FOGUETE L-75

```
print('Rocket Cooling Channel')
import librcc as rcc
import csv
import pyCEA

entrada = {}
saida = {}

entrada['motor_name'] = 'L75'
entrada['geometry_path'] = 'l75.csv'
entrada['channel_number'] = [ [0.293, 170], [0.425, 148],
   [0.523, 300], [1.600, 458] ] #[m]
entrada['dim_constant'] = 'FT'
entrada['CCH'] = 0.0015 #[m]
entrada['FT'] = 0.001 #[m]
entrada['IWT'] = 0.0015 #[m]
entrada['Tc_primary'] = 303 #[K]
entrada['Twg_primary'] = 1000 #[K]
entrada['Twc_primary'] = 600 #[K]
entrada['Taw_primary'] = 3000 #[K]
entrada['coolant'] = 'RP-1'
entrada['fuel'] = 'RP-1'
entrada['oxidizer'] = 'O2(L)'
entrada['of'] = 2.42
entrada['p0'] = 6000000 #[Pa] Press o de estagna  o
entrada['k_w'] = 290
entrada['m._c'] = 6.4
entrada['e'] = 1.6e-6 #Classe N7 Tornear NBR 4287 https://
   edisciplinas.usp.br/pluginfile.php/3344004/mod_resource/
   content/1/DTM%20I%20-%20aula%2007%20-%20Rugosidade%20e
```

```
   %20toler%C3%A2ncia.pdf
entrada['fs_wall'] = 1
entrada['sigmae'] = 240e6 #[Pa] #glidcop al 15

entrada['tol'] = 1e-6
entrada['max_iterations'] = 100
entrada['tol_opt'] = 1e-6
entrada['max_iterations_opt'] = 4000
entrada['FT_min'] = 0.001
entrada['CCH_min'] = 0.001
entrada['CCW_min'] = 0.001
entrada['w'] = 0.1

rcc.iteration(entrada, saida)
#rcc.optimize_channel2(entrada, saida)
print('Perda de carga: {} MPa'.format(saida['p_drop']/1e6))
print('Q(max):{} MW  Twg(max): {} K  CCW(max):{} CCW(min)
   :{}'.format(max(saida['Q']), max(saida['Twg']), max(
   saida['CCW']), min(saida['CCW'])))
t = rcc.calc_wall_thickness(entrada['p0'], entrada['
   geometry_path'], entrada['sigmae'], entrada['fs_wall'])
print('Espessura m nima da parede interna: {} m'.format(t)
   )
rcc.plot(saida)

for i in range(0,saida['size']):
    print('z:{} Q:{}'.format(saida['z'][i], saida['Q'][i]))
```

10 CÓDIGO PRINCIPAL PARA O MOTOR FOGUETE OGUM

```
print('Rocket Cooling Channel')
import librcc as rcc
import csv
import pyCEA

entrada = {}
saida = {}

entrada['motor_name'] = 'ogum'
entrada['geometry_path'] = 'ogum.csv'
entrada['channel_number'] = [ [1.600, 71] ]
entrada['dim_constant'] = 'CCW'
entrada['CCW'] = 0.001
entrada['CCH'] = 0.001
entrada['FT'] = 0.001
entrada['IWT'] = 0.001
entrada['Tc_primary'] = 303 #[K]
entrada['Twg_primary'] = 1000 #[K]
entrada['Twc_primary'] = 400 #[K]
entrada['Taw_primary'] = 3600 #[K]
entrada['coolant'] = 'C2H5OH(L)'
entrada['fuel'] = 'C2H5OH(L)'
entrada['oxidizer'] = 'N2O'
entrada['of'] = 5.65
entrada['p0'] = 1000000 #[Pa] Press o de estagna  o
entrada['k_w'] = 365
entrada['m._c'] = 1
entrada['e'] = 1.6e-6 #Classe N7 Tornear NBR 4287 https://
   edisciplinas.usp.br/pluginfile.php/3344004/mod_resource/
   content/1/DTM%20I%20-%20aula%2007%20-%20Rugosidade%20e
```

```
   %20toler%C3%A2ncia.pdf
entrada['fs_wall'] = 1
entrada['sigmae'] = 505e6 #[Pa] #glidcop al 15
#http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=
   mq304a
#https://www.makeitfrom.com/material-properties/Half-Hard
   -304-Stainless-Steel
#Otimiza  o
entrada['tol'] = 1e-5
entrada['max_iterations'] = 100
entrada['tol_opt'] = 1e-5
entrada['max_iterations_opt'] = 5000
entrada['FT_min'] = 0.001
entrada['CCH_min'] = 0.001
entrada['CCW_min'] = 0.001
entrada['w'] = 0.05

t = rcc.calc_wall_thickness(entrada['p0'], entrada['
   geometry_path'], entrada['sigmae'], entrada['fs_wall'])
print('Espessura m nima da parede interna: {} m'.format(t)
   )

rcc.iteration(entrada, saida)
#rcc.optimize_channel2(entrada, saida)
rcc.plot(saida)
print('Perda de carga: {} MPa'.format(saida['p_drop']/1e6))
print('Q(max):{} Twg(max): {} FT(max):{} FT(min):{}'.format
   (max(saida['Q']), max(saida['Twg']), max(saida['FT']),
   min(saida['FT'])))

with open('ogumopt.csv', mode='w', encoding='utf-8') as
   data_file:
```

```
53      csv_writer = csv.writer(data_file, delimiter=',')
54      lista = [14, 20, 30, 40, 50, 60,71]
55      for nch in lista:
56          entrada['CCH'] = 0.002
57          entrada['channel_number'] = [ [1.600, nch] ]
58          rcc.optimize_channel2(entrada, saida)
59
60          area_total_canais = 0
61
62          for j in range(0,saida['size']):
63              area_total_canais += 2*saida['L'][j]*saida['CCH
                    '][j]+2*saida['L'][j]*saida['CCW'][j]
64
65          area_total_canais = area_total_canais*nch
66
67          data_row = [ entrada['channel_number'][0][1], max(
                saida['Twg']), max(saida['Q']), entrada['CCH'],
                entrada['CCW'], saida['p_drop']/1e6,
                area_total_canais, saida['D_h'][0], (1/nch)
                /(820*entrada['CCH']*entrada['CCW'])]
68          print(data_row)
69          csv_writer.writerow(data_row)
```

11 GEOMETRIA DO MOTOR FOGUETE L-75

0.0003488756189256,0.10528142799023592
0.02882945413533035,0.10518180431244905
0.060418588027230546,0.1056799227013832
0.09200772191913076,0.10627766476810407
0.12359685581103094,0.10687540683482498
0.1551859870293113,0.10737352522375914
0.1867751259483134,0.10807089096826682
0.2025696954078144,0.10807089096826682
0.20651833227726896,0.10807089096826682
0.21046697401375647,0.10807089096826682
0.214415615750244,0.10782183177379977
0.21836425748673155,0.10757277257933272
0.22231289922321906,0.10662634764035796
0.22626154095970658,0.1056799227013832
0.23021018269619412,0.10448443856794137
0.23415882443268166,0.10328895443449953
0.23810746616916917,0.10219309397884455
0.24205610790565668,0.10109723352318955
0.2460047496421442,0.09900137306753454
0.2499533913786317,0.09690551261187951
0.2519277122468755,0.09438248830349869
0.2539020331151193,0.09185946399511787
0.2558763539836306,0.08933643968673705
0.2578506748516068,0.0868134153783563
0.25975448426027037,0.08599863601359983
0.261658293668934,0.08518385664884336
0.26356210307759764,0.08436907728408688
0.2654659124862613,0.08355429791933039
0.26692314931758404,0.08231611792397996
0.2683803861489068,0.08107793792862952

0.2698376229802296,0.07983975793327903
0.2712948598115524,0.0786015779379286
0.2731575375354609,0.07767827992415434
0.2750202152593695,0.07675498191038005
0.27688289298327806,0.07583168389660576
0.2787455707071866,0.0749083858828315
0.28026744304312445,0.07386767424880851
0.2817893153790624,0.0728269626147855
0.2833111877150003,0.07178625098076248
0.28483306005093817,0.0707455393467395
0.2864783274411413,0.06976887150557941
0.2881235948313444,0.06879220366441936
0.2897688622215476,0.06781553582325928
0.2914141296117507,0.0668388679820992
0.2924012900458726,0.06554998665073222
0.2933884504799945,0.06426110531936523
0.2943756109141635,0.06297222398799832
0.29536277134823824,0.061683342656631335
0.2970903021079515,0.0609922033919853
0.29881783286766483,0.06030106412733924
0.30054536362737816,0.059609924862693175
0.30227289438709143,0.05891878559804714
0.3035068449297438,0.05791756763628961
0.30474079547239613,0.056916349674532074
0.3059747460150485,0.05591513171277454
0.30720869655770083,0.05491391375101701
0.3089009715876241,0.05413257947808895
0.3105932466175473,0.05335124520516092
0.31228552164747053,0.052569910932232865
0.3139777966773938,0.05178857665930481
0.3152470029498362,0.05099336623125645
0.31651620922227863,0.05019815580320809

0.31778541549472106,0.049402945375159726
0.3190546217671634,0.0486077349471114
0.32086441589638687,0.048371128712367706
0.3226742100256103,0.04813452247762401
0.32448400415483375,0.04789791624288031
0.3262937982840572,0.047661310008136615
0.3282681191523009,0.04746206265256299
0.3302424400205447,0.04726281529698936
0.3322167608887885,0.04706356794141574
0.33419108175703227,0.04686432058584211
0.33616540262527606,0.04659035547192835
0.3381397234935198,0.0463163903580146
0.3401140443617635,0.046042425244100856
0.3420883652300073,0.045768460130187094
0.34381627392250547,0.04557634509764032
0.34554418261500364,0.045384230065093546
0.3472720913075018,0.045192115032546776
0.349,0.045
0.34937803356696184,0.04549098606590727
0.34975606713392365,0.045981972131814465
0.35013410070088546,0.04647295819772166
0.35051213426784733,0.04696394426362893
0.35228902304926674,0.04769617829536205
0.3540659118306861 4,0.048428412327095174
0.3558428006121055,0.04916064635882827
0.3576196893935249,0.04989288039056139
0.35970469188620047,0.051341046397662474
0.3617896943788761,0.05278921240476359
0.3638746968715517,0.05423737841186471
0.36595969936422734,0.05568554441896584
0.36787086855486734,0.05698367112952135
0.36978203774550733,0.05828179784007687

0.3716932069361474,0.05957992455063243
0.3736043761267874,0.06087805126118795
0.3755969777438112,0.06221881992473555
0.3775895793608349,0.0635595885882831
0.37958218097785873,0.06490035725183069
0.3815747825948253,0.06624112591537828
0.3835491034631263,0.06743661004882012
0.3855234243313701,0.06863209418226195
0.3874977451996139,0.06982757831570378
0.3894720660678576,0.07102306244914558
0.39150122918244146,0.07210646994507722
0.3935303922970253 6,0.07318987744100888
0.3955955541160925,0.07427328493694055
0.3975887185261931,0.07535669243287219
0.3994752918002927,0.07643262815296983
0.40136186507439225,0.07750856387306748
0.40324843834849183,0.07858449959316513
0.4051350116225914,0.07966043531326278
0.40710933249083514,0.08085591944670459
0.4090836533590789,0.08205140358014643
0.4110579742273227,0.08324688771358826
0.4130322950956644,0.08444237184703007
0.4150395213116142 6,0.08580721623270947
0.4170467475276621 3,0.0871720606183889
0.4190539737437099 5,0.0885369050040683
0.4210611999597577 7,0.08990174938974771
0.4248688187770850 4,0.09185152708357544
0.428676437594412 3,0.09380130477740317
0.4330011404486605 6,0.09619227304428683
0.4373258433029087 5,0.09858324131117047
0.4409571834712856 5,0.10019501009822147
0.4445885236396626,0.10180677888527251

0.44885446694426073,0.10420842111763334
0.45683401378674593,0.10891563989306051
0.4645620697567287,0.11265579739625703
0.4724959147272638,0.11630866558177369
0.4800923493060303,0.12076611356503536
0.48795307128144527,0.125182763280251
0.496132400592741,0.12911078257584557
0.5001280499689484,0.1308613129140997
0.5153350214183974,0.13797017392188765
0.547135689689038,0.15340472800185972
0.5790068694192587,0.16591461554109022
0.610596003311159,0.17786945687550848
0.6421851372030591,0.18952542717656623
0.6737742710949595,0.20028478437754263
0.7053634049868596,0.21064564686737175
0.7369525388787597,0.22020951993490634
0.76854167277066,0.2296737693246541
0.8001308066625601,0.23883914768104142
0.8317199405544604,0.24670941822620007
0.8633090744463605,0.254978183482506
0.8948982083382608,0.26235033563873056
0.9264873422301609,0.2696228641171683
0.958076476122061,0.27619802685109834
0.9896656100139614,0.2831716842961757
1.0212547439058615,0.28904948128559793
1.052843877977616,0.29552502034174116
1.0844330116896619,0.3010043226200162
1.116022145581562,0.30678249593165163
1.1476112794734623,0.31176367982099257
1.1792004133653624,0.3171433584214808
1.2107895472572627,0.3223237896663953
1.2423786811491628,0.32710572620016265

1.273967815041063,0.33228615744507717

1.3055569489329633,0.33577298616761586

1.3371460828248634,0.3392598148901545

1.3687352167167637,0.3435436330349877

1.4003243506086638,0.3474289564686736

1.431913484500564,0.3511150325467859

1.4635026183924642,0.354900732302685

1.491,0.3595

12 GEOMETRIA DO MOTOR FOGUETE OGUM

0.0013566253742490798,0.06829576784556021
0.008484182773852312,0.06806070295886471
0.015612119769558404,0.06806070295886471
0.0227400567652645,0.06806070295886471
0.02986799376097058,0.06806070295886471
0.03699593075667668,0.06806070295886471
0.04412386775238278,0.06806070295886471
0.05125180474808887,0.06806070295886471
0.05837974174379495,0.06806070295886471
0.06550767873950106,0.06806070295886471
0.07263561573520715,0.06806070295886471
0.07976355273091323,0.06806070295886471
0.08689148972661932,0.06806070295886471
0.09401942672232542,0.06806070295886471
0.10114736371803151,0.06806070295886471
0.10471133221588456,0.06806070295886471
0.10827527698898118,0.06804601140344624
0.11183881844121851,0.06778156340591379
0.11540224126967369,0.06744365763128901
0.11896537940105173,0.06692945319164262
0.12252818538583976,0.06620956697613764
0.12609070667355068,0.06531338209561105
0.12965280091564585,0.06415274921755203
0.13321451556163816,0.0627570514527975
0.1367758031620148,0.06109690569051054
0.14033663999201923,0.05915762037527267
0.14389735819824154,0.057144877282942444
0.14745798150543815,0.05507336796893836
0.15012848457297023,0.053541773316563
0.1510186522621476,0.0530312417657712

0.15190881402013587,0.052517037326124796
0.15279897577812412,0.052002832886478396
0.15368913753611238,0.051488628446832
0.15457929929410064,0.05097442400718559
0.1554694610520889,0.05046021956753918
0.15635962281007715,0.04994601512789278
0.1572497856806543,0.04943181068824636
0.1581399463260537,0.04891760624859996
0.1590301080840419 4,0.048403401808953556
0.15992026984203017,0.04788919736930716
0.16081043160001843,0.047374992929660756
0.16170059335800668,0.04686078849001435
0.16259076104718406,0.04635025693922255
0.1634809287363614,0.04583972538843077
0.16437109642553877,0.04532919383763899
0.16526126411471614,0.044818662286847194
0.1661514199415153,0.04430078495834617
0.16704157576831447,0.043782907629845144
0.1679317315951136,0.04326503030134413
0.16882188742191276,0.04274715297284311
0.1697120551110901,0.042236621422051326
0.1706022228002675,0.04172608987125953
0.17149239048944487,0.041215558320467735
0.17238255817862222,0.04070502676967595
0.17327271993661048,0.040190822330029546
0.17416288169459873,0.03967661789038314
0.175053043452587,0.03916241345073673
0.17594320521057524,0.038648209011090326
0.176833669685635,0.03813400457144392
0.17772352872655175,0.03761980013179752
0.17861369048454,0.03710559569215112
0.17950385224252827,0.03659139125250471

0.18039401993170562,0.03608085970171293
0.181284187620883,0.035570328150921135
0.18217435531006038,0.03505979660012934
0.18306452299923773,0.03454926504933756
0.1839546788260369,0.03403138772083653
0.18484483465283602,0.03351351039233552
0.1857349047963515,0.03299563306383451
0.18662514630643431,0.032477755735333486
0.18751531399561167,0.031967224184541704
0.18840548168478904,0.03145669263374991
0.18929564937396642,0.030946161082958112
0.1901858170631438,0.030435629532166317
0.19107597882113206,0.029921425092519913
0.19196614057912031,0.02940722065287351
0.19285630233710857,0.028893016213227107
0.19374646409509683,0.028378811773580703
0.19463662585308508,0.027864607333934296
0.19552678761107334,0.02735040289428789
0.1964169493690616,0.026836198454641483
0.19730711112704985,0.026321994014995076
0.19819736778406383,0.025866555797022545
0.1990876244410778,0.025411117579050013
0.19997788109809178,0.024955679361077482
0.20086813775510573,0.024500241143104964
0.20175859607254937,0.02416968114618941
0.20264905438999298,0.023839121149273862
0.2035395127074366,0.023508561152358318
0.2044299710248802,0.023178001155442773
0.20532060727799703,0.022957627824165744
0.20621124353111386,0.02273725449288871
0.20710187978423067,0.022516881161611685
0.2079925160373475,0.022296507830334655

0.20888330650138115,0.022171629609277668

0.20977409696541477,0.022046751388220685

0.2106648874294484,0.0219218731671637

0.211555677893482,0.02179699494610672

0.2124466284996215,0.0217712847241244

0.213337579105761,0.02174557450214208

0.21422852971190054,0.021719864280159762

0.2151194803180400 4,0.021694154058177442

0.2160106029286636 4,0.02177495761297902

0.2169017255392872 7,0.021855761167780596

0.217792848149910 9,0.021936564722582173

0.218683970760534 5,0.02201736827738375

0.219575259444453 1,0.022201012720114602

0.220466548128371 74,0.02238465716284546

0.221357836812290 38,0.02256830160557632

0.222249125496208 98,0.022751946048307173

0.223140503147964 21,0.02299068382385729

0.224031880799719 45,0.023229421599407406

0.224923258451474 68,0.023468159374957522

0.225814636103229 9,0.02370689715050764

0.226706013754985 11,0.02394563492605775

0.227597391406740 35,0.024184372701607875

0.228488769058495 58,0.024423110477158

0.229380146710250 8,0.02466184825270812

0.230271524362006 04,0.02490058602825824

0.231162902013761 27,0.025139323803808355

0.232054279665516 5,0.025378061579358475

0.232945657317271 7,0.025616799354908584

0.233837034969026 94,0.025855537130458697

0.234728412620782 17,0.02609427490600881

0.235619790272537 4,0.026333012681558923

0.236511167924292 64,0.026571750457109036

0.23740253964485875,0.026806815343804535
0.23829391136542488,0.02704188023050004
0.23918528308599102,0.027276945117195545
0.24007665480655713,0.027512010003891044
0.24096803245831233,0.027750747779441153
0.24185941011006756,0.027989485554991273
0.2427507877618228,0.028228223330541393
0.24364216541357803,0.02846696110609151
0.24453354306533326,0.028705698881641626
0.2454249207170885,0.028944436657191742
0.24631629836884372,0.02918317443274186
0.24720767602059895,0.029421912208291975
0.2480990536723542,0.029660649983842095
0.24899043132410942,0.029899387759392215
0.24988180897586465,0.030138125534942335
0.25166456427937510,0.030615601086042575
0.2552300689552069,0.03156687929938842
0.2587955617686605,0.03251081173502502
0.2623610723756814,0.0334657628372255
0.2659265891389137,0.03442438682828058
0.26949211138329054,0.035386683708190286
0.27305760419674413,0.03633061614382689
0.2766231088725759,0.037281894357172746
0.28018860761721864,0.038229499681663975
0.28375413601780686,0.03919546945042829
0.2873196406936387,0.04014674766377414
0.2908851335070923,0.04109068009941076
0.29445064411411315,0.04204563120161123
0.2980161547211341,0.04300058230381169
0.301581665328155,0.043955533406012166
0.30871266281744036,0.04585074405499462
0.31584368996267126,0.04776431914825017

0.3229746993143349,0.04966687557494187
0.33010570866599853,0.05156943200163356
0.3372367417424186,0.05348667998374374
0.3443677451628931,0.05538556352158082
0.3514987663769349,0.05729546572598175
0.35862976386622025,0.05919067637496422
0.3657608080501854,0.06111527013478363
0.3728918003631148,0.06300680789491145
0.3800228097147784,0.06490936432160316
0.38715384279119847,0.06682661230371333
0.39428482841810564,0.06871447717498656
0.4014158674257148,0.07063539804595136
0.4085468649150002,0.07253060869493382
0.4156779039226093 4,0.0744515295658986
0.4228089014118947,0.07634674021488105
0.4299399107635583 6,0.07824929664157276
0.437070943839978 4,0.08016654462368293
0.444201929466885 5,0.08205440949495615
0.451332968474494 7,0.08397533036592096
0.458463965963780 1,0.0858705410149034
0.465594987177821 9,0.08778044321930434
0.472726002460674 64,0.08968667253485066
0.479857011812338 25,0.09158922896154235
0.486988027095190 96,0.09349545827708867
0.494119048309232 76,0.09540536048148962
0.501250069523274 6,0.09731526268589055
0.508381067012559 9,0.09921047333487298
0.515512088226601 8,0.10112037553927392
0.522643103509454 5,0.10302660485482028
0.529774124723496 4,0.10493650705922122
0.536662028275391 9,0.10677221690875885