Fundação Universidade Federal do ABC
Pró reitoria de pesquisa
Av. dos Estados, 5001, Santa Terezinha, Santo André/SP, CEP 09210-580
Bloco L, 3ºAndar, Fone (11) 3356-7617
iniciacao@ufabc.edu.br

Relatório Final de Iniciação Científica
referente ao Edital: 07/2021

Nome do aluno: Julia Belucci de Oliveira

Assinatura do aluno: 

Nome do Orientador: Prof. Dr. Wallace Gusmão Ferreira

Assinatura do orientador: 

Título do projeto: pyRocket - Implementação de plataforma integrada de cálculo para desenvolvimento de foguetes experimentais em linguagem Python

Palavras-chaves: Cálculo, Python, Foguete, Foguete-modelismo, Foguete experimental, espaço-modelismo, programação.

Área do conhecimento: Projeto de Foguetes Experimentais

Bolsista: Não

São Bernardo do Campo - SP

2022

“Per Aspera, Ad Astra”

Resumo

As competições de foguete-modelismo possuem um grande potêncial de despertar a curiosidade de pessoas de diversas idades, isso porque o espaço é pouco explorado e ainda restam muitas dúvidas sobre como ele funciona e o que podemos encontrar lá. Assim, a ciência aeroespacial ainda possui muito potêncial para se desenvolver, mas para que isso seja possível é necessário investimento em pesquisas que sejam capazes de questionar o que parece ser impossível. Sabendo disso Richard Nakka, um grande intusiasta do espaço modelismo, disponibilizou através de seu site todas as suas experiências no desenvolvimento de foguetes experimentais. Dados extremamente valiosos, que têm por objetivo facilitar a elaboração de espaço modelos, foram disponibilazados no formato de planilhas em Excel e atualmente são utilizados por pessoas do mundo todo incluindo pela UFABC Rocket Desing. Dessa maneira, esse projeto tem por objetivo tonar os cálculos desssas planilhas integrados através da linguagem de programação Python, visando facilitar o trabalho feito por essa entidade da UFABC assim como por outros que se interessarem nesse ramo, incentivando o desenvolvimento das competições de foguete modelismo brasileiras e, consequentemente, a evolução da ciência aeroespacial.

Palavras-chave: Cálculo, Python, Foguete, Foguete-modelismo, Foguete experimental, espaço-modelismo, programação.

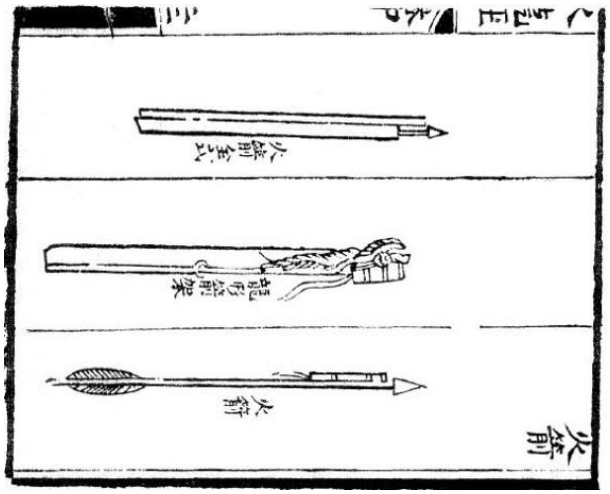
Sumário

1 Introdução

Os foguetes espaciais atualmente conhecidos são resultados de mais de dois mil anos de invenções, experimentos e descobertas. Inicialmente a observação e inspiração da natureza e mais tarde a pesquisa metódica formaram a base para o surgimento dos foguetes modernos [3].

Os primeiros modelos de foguetes que se tem notícia foram usados pelos chineses, em 1232, presos à flechas para afastar os invasores mongóis. Esses foguetes primitivos eram formados por tubos de bambu e couro cheios de pólvora. A ciência por trás dos foguetes surgiu anos depois com Galileu Galilei criando o conceito de inércia, posteriormente incorporado em uma das Leis de Isaac Newton, que é usada como base para a construção de motores-foguetes até os dias atuais [3].

Figura 1: Flechas de fogo



Fonte: https://www.spacestuff.org/how-rockets-work

Durante a corrida espacial ocorreu o lançamento do primeiro satélite na órbita da terra, esse acontecimento contribuiu para uma nova era de descoberta que nos proporcionou grandes avanços tecnológicos como os hardwares e softawers de comunicação, além de nos ajudar a compreender com maior facilidade fenômenos naturais, tornando possível prevê-los [5]. Durante esse período não era esperado que os novos avanços mudariam o rumo da história da humanidade e muitas vezes o esforço para explorar o espaço foi questionado, mas graças a curiosidade de cientistas como Joule Verne, Konstantin Tsiolkovsky, Neil Armstrong e muitos outros aos poucos o que parecia impossível se tornarou realidade [3].

Figura 2: O primeiro foguete americano em órbita.

Fonte: https://www.nasa.gov/missions/solarsystem/Why_We_01pt1.html

Segundo Giorgio Giacaglia, os motores foguetes construídos atualmente são uma simples melhoria dos que eram construídos vinte anos atrás e para uma nova descontinuidade no processo tecnológico é necessário mudanças de larga escala na eficiência e nos custos desses artefatos, ou seja, novos motores e fontes de energia, assim como tecnologia mais segura e limpa [5]. Dessa forma, a evolução dos foguetes depende do descobrimento, da necessidade e da experimentação, por isso essa ela não é linear sendo necessário buscar novos desafios que tragam grandes saltos para a ciência, como aconteceu na guerra fria [6].

Richard Nakka desenvolve e testa foguetes experimentais desde 1972, em seus projetos ele é responsável por construir desde o motor até o propelente usado. Através de sua grande experiência coletou dados suficientes para criar padrões e ajudar no desenvolvimento de novos foguetes e hoje em seu site essas informações são disponibilizadas e usadas por pessoas do mundo todo no ramo de foguete modelismo. O objetivo da criação da página é inspirar uma nova geração de cientistas e engenheiros, afinal o espaço modelismo pode ser praticado por diversas idades, podendo ser usado para estudar novas alternativas para foguetes de maior porte, ou apenas para o uso recreativo [1].

Figura 3: Flight C-31, Sept. 1938

Fonte: http://www.nakka-rocketry.net/gallery1/photo1.html

No Brasil, o espaçomodelismo é pouco valorizado assim como o desenvolvimento da ciência aeroespacial. Existem poucos fabricantes de foguetes motores brasileiros, dificultando o acesso a esse tipo de atividade que aguça a curiosidade para a ciência, e por isso é essencial o desenvolvimento de pesquisas que impulsionem a difusão da importância dessa área [7].

2 Objetivos

Dentro da UFABC Rocket Desing há um fluxo de projetos que guia os membros para a criação de novos motores-foguetes. Na imagem abaixo é possível encontrar algumas partes importantes do processo de criação de novos projetos.

Figura 4: Fluxo de projeto

Fonte: própria

Neste processo são utilizadas as planilhas disponibilizadas por Richard Nakka que ajudam a desenvolver o foguete experimental com o melhor desempenho possível evitando erros durante o seu funcionamento.

A planilha Casing é utilizada para calcular a pressão do invólucro do motor foguete, o que ajuda a decidir os melhores materiais a serem utilizados, evitando que ocorra uma explosão durante o seu funcionamento. A Ezalt nos permite estimar a performance de voo, enquanto a O-ring permite construir o anel de vedação para alguns compartimentos do foguete. Por fim, a Solid Rocket Motor (SRM) permite definir a velocidade do foguete bem como o seu apogeu, sua curva de queima, a geometria do bocal, a curva de impulso em função do tempo e a

classificação do motor foguete tomando como base o impulso total calculado.

Apesar de arquivos diferentes, cada planilha gera dados que são utilizados para obter os resultados de uma nova planilha, conforme é apresentado no fluxo de projetos da UFABC Rocket Desing. Isto é, para alcançar melhores resultados é essencial o uso integrado dos cálculos desenvolvidos por Richard Nakka.

Dessa forma, o objetivo é dar continuidade ao pacote pyRocket que abriga os módulos em linguagem python da Casing e Ezalt através da adaptação dos arquivos SRM e O-ring, buscando a troca de dados e a interação entre as planilhas de maneira direta e eficaz, facilitando o processo de construção dos foguetes.

3 Fundamentação Teórica

3.1 Fundamentação

Como dito por Richard Nakka[8], a operação de um motor foguete é um processo extremamente complexo, e mesmo sendo mais simples do que o funcionamento de foguetes “profissionais” inclui diversas reações químicas e processos de emissões de gases que devem ser avaliados para que um melhor desempenho possa ser obtido.

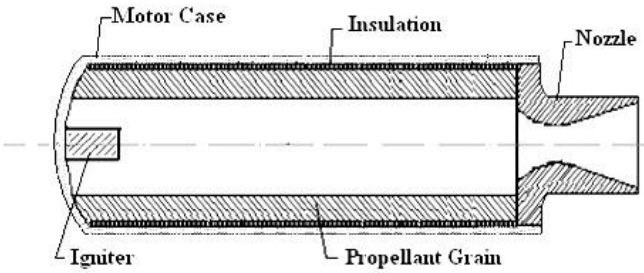
O projeto de um motor-foguete é essencial para aumentar a eficiência de um lançamento, e um passo muito importante que torna possível a criação de um foguete experimental é compreender o seu funcionamento e quais são os processos que ocorrem após a ignição do motor.

Para poder analisar teoricamente os processos que ocorrem com o foguete propelente durante a combustão do propelente e com os gases e partículas emitidas após a ignição do motor-foguete precisamos considerar algumas simplificações. Iremos assumir que durante o processo de funcionamento a combustão do propelente é completa e ocorre de maneira perpendicular à superfície de queima, e além disso o processo de escoamento e expansão dos produtos da combustão, que obedecem às leis dos gases ideais, ocorre de maneira uniforme, adiabática e unidimensional através da tubeira. Por fim, é importante estabelecer que durante todo esse processo é estabelecido um equilíbrio químico na câmara de combustão.

3.2 Estruturas do foguete

Um foguete amador é composto por duas estruturas importantes: a Câmara de Combustão e a Tubeira, como mostrado na figura 5. Elas são essenciais para que o foguete consiga realizar o seu principal objetivo, que é transformar energia térmica em energia cinética.

Figura 5: Estrutura motor-foguete



Fonte: https://www.researchgate.net/figure/Main-Parts-of-a-Solid-Rocket-Motor-1_fig1_305632527

Na Câmara de Combustão fica localizado o nosso propelente, que funcionará como combustível para o motor-foguete. Após a ignição essa substância entrará em combustão e esse processo exotérmico será responsável pela liberação de energia em forma de calor.

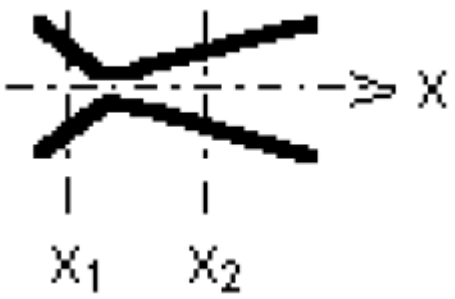
A pressão que é gerada dentro da Câmara de Combustão durante esse processo é crucial para uma operação bem sucedida, isso porque ela influencia a taxa de queima do propelente, a eficiência termodinâmica e o empuxo.

A tubeira tem a função de acelerar os produtos da combustão a maior velocidade possível durante a saída, processo que também será essencial para o aumento do empuxo resultante do motor foguete.

Para realizar a sua função é necessário que o escoamento através da tubeira seja dependente somente da área da seção transversal, ou seja, que o escoamento seja invíscido e adiabático (sem perda de calor), assim é necessário minimizar efeitos friccionais, perturbações e condições que podem levar perdas por choque.

A geometria da tubeira é composta por uma garganta, região onde as paredes da tubeira se encontram mais próximas, e o seu formato pode ser ilustrado pela figura abaixo.

Figura 6: Geometria da Tubeira



Fonte: Solid Rocket Motor Theory - Richard Nakka

Para o escoamento adiabático entre dois pontos x_1 e x_2 podemos assumir a relação abaixo, na qual adotamos o h como a entalpia do fluido, v a velocidade de

escoamento na direção x, Cp o calor específico do fluído e T a temperatura no ponto.

$$h_1 - h_2 = \frac{1}{2}(v_2{}^2 - v_1{}^2) = Cp(T_1 - T_2) \quad (1)$$

Observando essa relação é possível concluir que a diminuição da entalpia do fluido será igual ao aumento da energia cinética e à diminuição da temperatura durante o escoamento, o que ilustra que o calor do fluido será utilizado para acelerar o escoamento através da tubeira aumentando o empuxo resultante.

O Empuxo, por sua vez, é a força que o motor exerce e que propulsiona o foguete para cima. Essa força é gerada através do escoamento dos produtos da combustão do propelente pela tubeira em alta velocidade, ou seja, quando a pressão interna da câmara do motor é conduzida pela garganta da tubeira ela gera uma força superior à força peso que é capaz de impulsioná-lo verticalmente.

A expressão para o empuxo é dada pela seguinte relação:

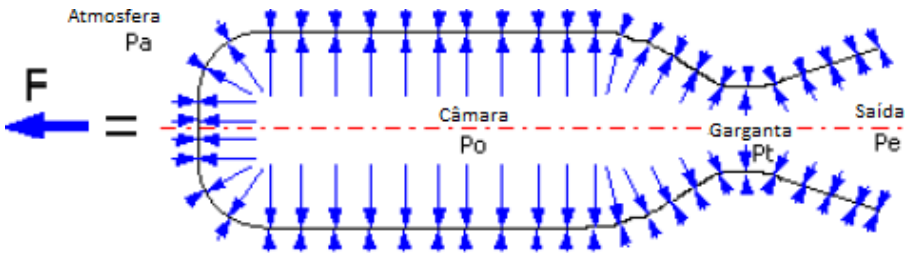
$$F = \int P \cdot dA = \dot{m} \cdot v_e + (P_e - P_a) \cdot A_e \quad (2)$$

Nessa equação podemos considerar P como a pressão resultante que atua na câmara e na tubeira, e A a área de saída da tubeira. Na segunda parte da equação, $\dot{m}$ representa o fluxo de massa e v_e a velocidade de exaustão, o termo que está sendo multiplicado pela área de saída da tubeira (A_e) é uma relação de "empuxo pressão" que seria igual a zero para uma tubeira com expansão ótima, ou seja, quando a pressão de saída (P_e) e a pressão atmosférica (P_a) são iguais.

Considerando esta relação podemos concluir que ao dobrar a área da garganta e manter a pressão constante, o empuxo será dobrado, entretanto essa alteração envolve variações importantes como um aumento na área de queima do grão. Da mesma forma, quando a pressão é dobrado o empuxo resultante também será aproximadamente dobrado, entretando nesse caso a estrutura deverá ser reforçada para suportar a nova pressão estabelecida.

Assim, é possível concluir que o empuxo é diretamente proporcional à área da garganta e quase diretamente proporcional à pressão da câmara. E além disso, como a integral das forças de pressão resultante atuando sobre a câmara e tubeira, é projetada em um plano normal ao eixo de simetria do motor-foguete é possível obter a seguinte ilustração:

Figura 7: Forças na câmara de combustão e tubeira



Fonte: Solid Rocket Motor Theory - Richard Nakka

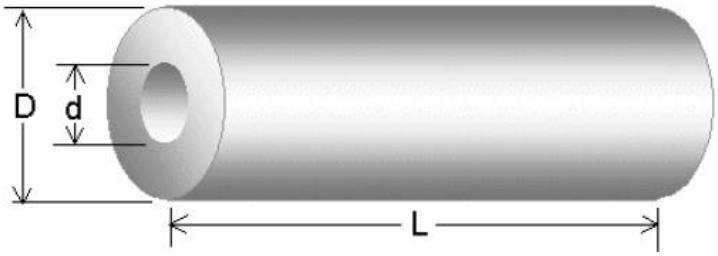
3.3 Grão propelente

Os propelentes comumente utilizados nas competições de foguete modelismo são sólidos e possuem como base açúcares. Além disso, são compostos por dois constituintes principais: o combustível e o oxidante. Em alguns casos o propelente pode ser constituído por uma mistura de oxidantes com granulometrias distintas, polímeros aglomerados e até metais.

De maneira geral, os propelentes são processados em um formato cilíndrico para que haja um melhor ajuste no motor-foguete e maior eficiência de queima. A essa forma geométrica similar nos referimos como grão propelente.

Além disso, é comum que seja introduzida uma alma ao centro do grão que pode ser composta por um único segmento cilíndrico ou pode conter vários segmentos transversais, o que afeta diretamente a performance de voo na forma empuxo-tempo.

Figura 8: Grão de cilindro oco



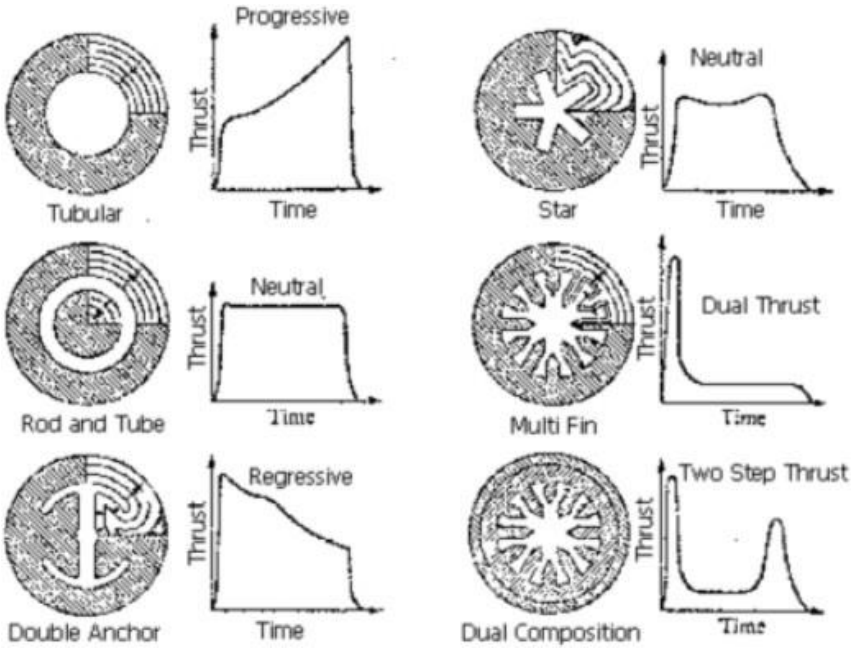
Fonte: Solid Rocket Motor Theory - Richard Nakka

Sendo o empuxo, bem como a pressão da câmara, proporcional à área de queima do grão em qualquer instante de tempo, e sabendo que a superfície de queima em qualquer ponto recua na direção normal à superfície do ponto específico, cria-se uma relação entre a superfície de queima e a distância da camada que foi queimada.

A área de queima total não depende apenas da geometria do grão, mas também do uso de inibidores. Inibidor é um material que reveste uma superfície e impede que essa região seja consumida durante a combustão do motor. Visto que, com uma maior taxa de queima há também um aumento na força de empuxo, é sempre idealizada uma maior superfície de queima, porém é interessante que a estrutura do motor-foguete esteja protegida durante a combustão e por isso é comum que a parte mais externa do grão seja inibida.

Assim, como a distância de queima depende quase inteiramente da forma inicial do grão e dos contornos inibidos, concluímos que variando o formato da alma somos capazes de alterar a curva de empuxo em função do tempo.

Figura 8: Formas do núcleo e influência sobre a curva de empuxo



Fonte: Solid Rocket Motor Theory - Richard Nakka

Na figura acima são apresentados os gráficos de empuxo em função do tempo para alguns formatos de alma de grão propelente. O grão conhecido por ter a alma no formato estrela (nomeado como star na figura) possui uma curva neutra, o que é desejável pois nesse caso a tubeira opera com uma pressão constante na câmara de combustão proporcionando uma maior eficiência no impulso resultante total.

A principal função do grão propelente é produzir produtos combustão que realizem uma taxa de escoamento de acordo com a relação apresentada abaixo, onde a taxa de escoamento (m_g) é igual ao produto entre a massa específica do propelente (p_p), a área de queima (A_b) e a taxa de queima do propelente (r).

$$m_{\mathrm{g}} = A_{\mathrm{b}} \cdot \rho_{\mathrm{p}} \cdot r \tag{3}$$

Levando em consideração que estamos tratando de propelentes compostos por um oxidante e um combustível, a sua massa específica poderá ser encontrada através do seguinte cálculo:

$$\rho_{\mathrm{p}} = \frac{1}{\frac{f_{\mathrm{o}}}{\rho_{\mathrm{o}}} + \frac{f_{\mathrm{f}}}{\rho_{\mathrm{f}}}} \tag{4}$$

Onde ρ representa a massa específica e f é a fração de massa, bem como os subscritos o e f representam, respectivamente, o oxidante e o combustível.

É importante lembrar que tipicamente a massa específica real do grão é percentualmente menor do que a ideal, calculada anteriormente. Portanto para

obter a massa específica real do grão é realizada a sua pesagem de forma experimental para encontrar a sua massa real (m_g) e o volume real (V_g) é calculado através da relação:

$$V_g = \frac{\pi}{4} \cdot (D^2 - d^2) \cdot L \tag{5}$$

Dessa forma, considerando o diâmetro externo (D), o diâmetro da alma (d) e o comprimento (L) do grão, conclui-se que a densidade real do grão ($\rho_{p\ real}$) pode ser expressa através da relação fundamental da densidade.

$$\rho_{p\ real} = \frac{m_g}{V_g} \tag{6}$$

Uma maneira de maximizar a duração da queima é aumentando a expressura e a fração de camada (w_f), valor que pode ser calculado através da expressão abaixo considerando t_b como o tempo de queima e r como o raio externo do grão. Entretanto, com isso haverá uma diminuição no diâmetro da alma do grão.

$$w_f = \frac{D-d}{D} = \frac{2 \cdot r \cdot t_b}{D} \tag{7}$$

Além disso, para encontrar a eficiência volumétrica do motor, ou seja, a parte do motor que está sendo ocupada pelo grão propelente, é utilizada a fração de carregamento volumétrica (V_l), valor calculado através da razão entre o volume real do grão (V_g) e o volume total da câmara (V_a).

Outro parâmetro importante a ser considerado durante o projeto do motor é a razão entre o comprimento e o diâmetro do grão, isso porque quanto maior a relação L/D maior será o efeito de queima erosiva negativa, ou seja, pode gerar altos fluxos de massa diferentes ao longo do grão, o que deve ser evitado.

3.4 Sobre Python

Python é uma linguagem de programação criada por Guido van Rossum e lançada em 1991. Ela ganhou destaque por utilizar menos linhas de código para conseguir expressar conceitos quando comparada com as outras linguagens de programação amplamente utilizadas, além de ser facilmente aprimorada pelos usuários através da disponibilização de bibliotecas com diferentes funcionalidades, cumprindo com o seu objetivo de fornecer produtividade avançada ao desenvolvedor [9].

Inicialmente a criação da nova linguagem de programação era apenas um projeto para manter o criador ocupado durante do período do Natal e o objetivo era criar um sucessor para a linguagem de programação ABC, que também foi criada com a ajuda de van Rossum, aprimorando e resolvendo alguns problemas

encontrados nessa primeira linguagem. O nome da linguagem faz referência a série "Monty Python's Flying Circus" da BBC, a qual o criador da linguagem era muito fã [9].

Atualmente Python é reconhecida pela sua elegância e simplicidade e por isso é utilizada para developing, scripting, geração e teste de softwares. Essas características fizeram do Python a linguagem de codificação mais popular do mundo e atualmente grandes empresas de tecnologias utilizam essa linguagem de programação, são elas: Dropbox, Google, Quora, Mozilla, Hewlett-Packard, Qualcomm, IBM e Cisco [9].

4 Metodologia

4.1 Materiais e métodos

Como apresentado anteriormente, a pesquisa se baseia nas planilhas "Solid Rocket Motor" e "O-ring" desenvolvidas por Richard Nakka [1], visto a sua credibilidade no ramo de foguetemodelismo.

Ao início do projeto foi realizado um estudo sobre a linguagem de programação a ser utilizada para a criação do programa, e para isso foram utilizadas informações disponíveis gratuitamente na internet, como vídeos da plataforma "YouTube" (https://www.youtube.com) e sites de dúvidas sobre a utilização da linguagem Python como o "Stack Overflow" (https://www.stackoverflow.org/).

Inicia-se a construção do programa com a engenharia reversa das planilhas, ou seja, é necessário compreender todos os cálculos que são realizados, a fucionalidade de cada base de dados disponibilizada, as referências de cada variável e por fim o raciocínio por trás de cada automação. Posteriormente, compreendendo a estrutura da planilha, é possível estruturá-la como código e construir o programa.

Para o desenvolvimento da pesquisa foi utilizado um notebook Lenovo Ideapad 330, com processador Intel Core i5 com 4GB de memória RAM rodando Windowns 10.

4.1.1 Instalação do Python

A instalação do Python pode ser realizada através do site oficial (https://www.python.org/), para isso basta clicar na aba download e a versão mais atualizada será instalada no seu computador.

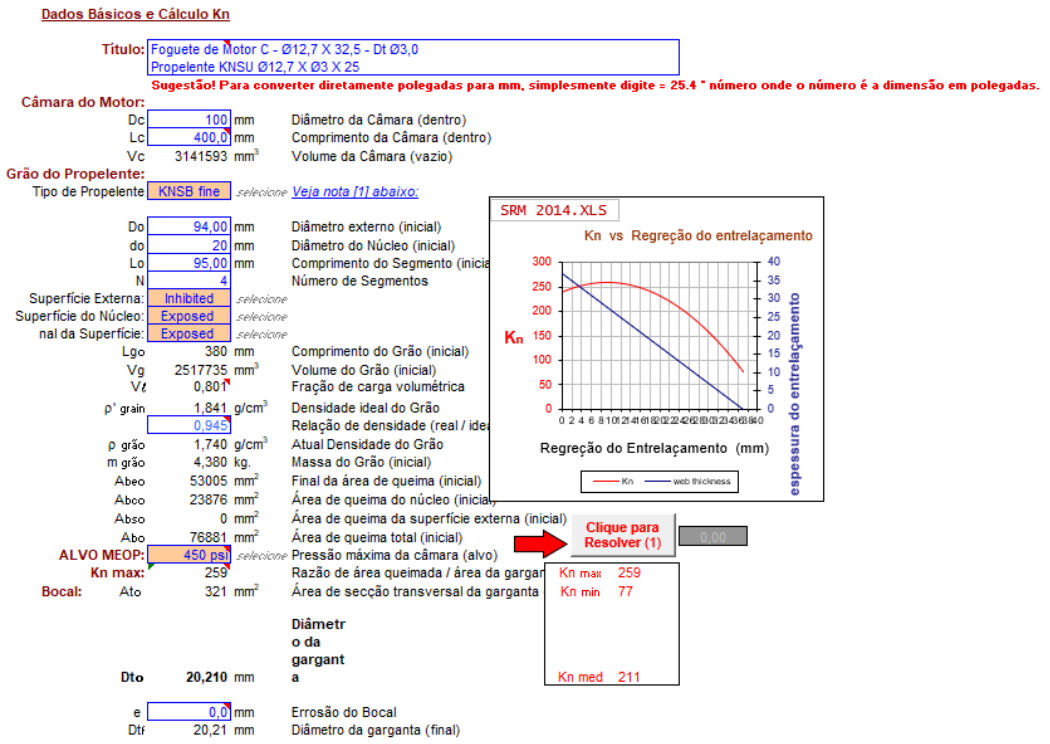
É importante saber que os códigos em Python precisam de uma plataforma param serem escritos, e elas são conhecidas como IDEs. A IDE selecionada para a criação do programa foi a Jupyter Notebook, que foi escolhida por ser uma ferramenta gratuita e intuitiva, podendo ser instalada através do navegador anaconda (https://www.anaconda.com/products/individual).

4.1.2 Implementação Data and Kn

A planilha Solid Rocket Motor[1], também conhecida por SRM, foi desenvolvida para prever o desempenho de um motor foguete que utiliza um grão propulsor de formato cilíndrico oco e que é feito à base de “açúcar”. As previsões de pressão e desempenho realizadas são baseadas na termodinâmica padrão e equações de fluxo e fluido, em conjunto com parâmetros obtidos empiricamente que relacionam a taxa de queima com o tipo de propelente escolhido.

Alguns fatores afetarão o desempenho real quando comparado com o que é previsto na planilha, entre eles a preparação do propulsor e a construção do bico, porém os cálculos realizados mostram uma aproximação razoável quando comparados com desempenho alcançado.

Figura 9: Planilha SRM



Fonte: Planilha SRM – Richard Nakka

A SRM é composta por algumas abas e dentre elas podemos citar a “Data and Kn”, que foi a aba implementada e que é apresentada na imagem acima. Nela são realizados os cálculos para gerar o gráfico da curva de queima, ou seja, ela determina o Kn com relação à regressão do entrelaçamento do grão, estabelecendo a curva de empuxo realizado pelo motor em cada espaço de tempo.

```
#Biblioteca utilizada para acessar base de dados
import pandas as pd
#Biblioteca utilizada para os cálculos
import math
#Bibliotecas utilizadas para plotar o gráfico
import matplotlib.pyplot as plt
import numpy as np
```

O código inicia com a importação das bibliotecas que auxiliam trazendo informações que o torna mais curto. A primeira biblioteca importada é chamada Pandas[14], ela é uma ferramenta de análise e manipulação de dados de códigos abertos, permitindo que o código leia base de dados que não são codificadas em linguagem Python.

Utilizamos também o módulo math[13] que fornece acesso às funções matemáticas que facilitarão cálculos futuros. A interface matplotlib.pyplot[15] será utilizada para plotar gráficos 2D, e por fim é chamado o módulo numpy[16] que também será utilizado para a análise de dados.

4.1.2.1 Base de Dados Definida Pelo Usuário

Após importar as bibliotecas são inseridos alguns dados pelo usuário. Para isso usamos a função “print”, capaz de exibir mensagens que identificam as próximas informações a serem inseridas, e em seguida, utilizamos a função “input” que permite ao programa associar os valores inseridos com as variáveis definidas.

```
title = input("Coloque um título: ")

#Câmara do Motor
print("Dados da Câmara do Motor")
print("")
#Diâmetro interno da câmara
Dc = int(input("Diâmetro interno da câmara em milímetro: "))
#Comprimento interno da câmara
Lc = float(input("Comprimento interno da câmara em milímetro: "))
```

A primeira informação a ser declarada é o nome do projeto, que é utilizado apenas para identificação do foguete que está sendo testado. Feito isso, os próximos elementos a serem informados serão utilizados em cálculos futuros.

Os primeiros dados numéricos a serem inseridos são as dimensões da câmara de combustão em milímetros, definidos pela variável D_c, que representa o diâmetro interno, e L_c representando o comprimento da câmara.

Em seguida é determinado o tipo de grão propelente que será utilizado, o que é feito relacionando o dado inserido com um valor numérico para a variável *fueltype*, como é ilustrado abaixo.

```
#Grão Propelente
print("Grão do Propelente")
print("")
#Tipo de grão propelente
fueltype = 0
while fueltype == 0:
    prop = input("""Escreva o tipo do propelente:
    Escolha entre as opções
    KNDX
    KNSB fine
    KNSB coarse
    KNSU
    KNER coarse
    KNMN coarse
    """)
    if prop == "KNDX":
        fueltype = 1
    elif prop == "KNSB fine":
        fueltype = 2
    elif prop == "KNSB coarse":
        fueltype = 3
    elif prop == "KNSU":
        fueltype = 4
    elif prop == "KNER" or prop == "KNER coarse":
        fueltype = 5
    elif prop == "KNMN" or prop == "KNMN coarse":
        fueltype = 6
    else:
        fueltype = 0
        print("Resposta inválida")
```

A função “while” cria um looping garantindo que o input seja uma das opções listadas ao usuário, assim caso a resposta seja inválida o sistema retornará a mensagem “Resposta inválida” e deverá ser informando um novo input.

Após garantir que o valor inserido está correto, o programa utiliza o condicional “if” para associar o nome do grão propelente escolhido com um valor para a variável *fueltype*. Essa variável será utilizada para relacionar o propelente escolhido com as suas propriedades físicas pré-determinadas por Richard Nakka. Assim, tomando como base a tabela disponível na aba Dados do Propelente conseguimos determinar a densidade ideal do grão propelente.

Figura 10: Propriedades físicas de produtos de combustão e propelentes

Dados do Propelente

Propriedades fisicas / térmicas de produtos de combustão e propelentes								
Tipo de Propelente	simb.	unid	notas	KNDX	KNSB	KNSU	KNER	KNMN
Densidade da massa de grão, ideal	ρ_p	g/cm^3		1,879	1,841	1,889	1,820	1,854
Razão de calores específicos, 2-ph.	k_{2ph}	-	[1]	1,043	1,042	1,044	1,043	1,042
Razão de calores específicos, mistura	k	-	[2]	1,131	1,136	1,133	1,139	1,136
Peso molecular efetivo.	M	kg/kmol	[3]	42,39	39,90	41,98	38,78	39,83
Temperatura da câmara	To	K	[4]	1710	1600	1720	1608	1616

Fonte: Planilha SRM – Richard Nakka

Para conseguir utilizar a tabela apresentada acima como uma base de dados foi criado um arquivo no modelo excel, nomeado como “Densidade ideal”, com as informações disponibilizadas no modelo original. Essa planilha será interpretada pela biblioteca pandas o que possibilitará importar todos esses dados para o programa no formato de uma matriz.

Para que esse processo possa ser realizado corretamente foi necessário algumas simplificações para evitar problemas de leitura pelo módulo, ou seja, foram trocados os rótulos das últimas cinco colunas dos nome dos propelentes para os respectivos valores, determinados anteriormente, para a variável fueltype. Dessa forma, o valor da varíavel pode ser usado como índice para encontrar as informações na matriz.

Figura 11: Base de dados Densidade ideal

Tipos de Propelente	simb.	unid	notas	1	2	3	4	5	6
Densidade da massa de grao, ideal	r p	g/cm3		1,879	1,841	1,841	1,889	1,82	1,854
Razao de calores específicos, 2-ph.	k2ph	-	[1]	1,043	1,042	1,042	1,044	1,043	1,042
Razao de calores específicos, mistura	k	-	[2]	1,131	1,136	1,136	1,133	1,139	1,136
Peso molecular efetivo.	M	kg/kmol	[3]	42,39	39,9	39,9	41,98	38,78	39,83
Temperatura da câmara	To	K	[4]	1710	1600	1600	1720	1608	1616

Fonte: Própria

Após o processo de identificar o tipo de grão propelente são inseridas informações sobre a geometria do grão propelente em milímetros como: diâmetro externo (D_o), diâmetro interno (d_o), comprimento (L_o), número de segmentos (N) e densidade real do grão (d_g).

Por fim, será determinada se a inibição das superfícies do propelente sólido serão inibidas ou não, processo que ocorrerá novamente através do uso do looping "while" e da condicional "if", associando a resposta do usuário às variáveis osi, ci e ei valores que serão utilizados em diversos cálculos realizados futuramente. Esse processo pode ser visto com mais detalhes abaixo.

```
#Inibidores
#Superfície externa
osi = -1
while osi < 0:
    SE = input("A superfície externa é Exposta ou Inibida? ")
    if SE == "Exposta":
        osi = 1
    elif SE == "Inibida":
        osi = 0
    else:
        osi = osi
        print("Resposta inválida")
#Superfície do núcleo
ci = -1
while ci < 0:
    SN = input("A superfície do núcleo é Exposta ou Inibida? ")
    if SN == "Exposta":
        ci = 1
    elif SN == "Inibida":
        ci = 0
    else:
        osi = osi
        print("Resposta inválida")
#Superfície final
ei = -1
while ei < 0:
    FS = input("O final da superfície é Exposta ou Inibida? ")
    if FS == "Exposta":
        ei = 1
    elif FS == "Inibida":
        ei = 0
    else:
        osi = osi
        print("Resposta inválida")
print("")
```

A pressão máxima causada pelos produtos da combustão, que será definida como P e é medida em mil pascal, também é um fator influenciado diretamente pela escolha do propulsor. Assim, é importante determinar qual é a pressão máxima da câmara de combustão corretamente, e para isso iremos utilizar novamente uma base de dados disponibilizada na planilha SRM.

Será utilizada uma listagem que relaciona cada pressão máxima inserida pelo usuário em libra-força por polegada quadrada com o seu valor respectivo em Mpa, garantindo que a pressão esteja sempre com a unidade de medida correta para realizar os cálculos.

```
#Densidade ideal do grão
Dados_propelente = pd.read_excel("Densidade ideal.xlsx")
pig = Dados_propelente[fueltype].loc[0]
print("A densidade ideal do grão é ", pig, "g/cm³")
```

Durante o processo de combustão do propelente pode ocorrer erosão no bocal, portanto essa é a última informação a ser declarada pelo usuário. Isso será feito através da variável e que representa a erosão do bocal em milímetros, assim quando o seu valor é igual a zero podemos afirmar que a área do bocal é a mesma durante todo o processo de funcionamento do motor foguete.

4.1.2.2 Cálculos Prévios

A partir dos dados inseridos o programa se torna capaz de realizar os cálculos

prévios que são essenciais para desenhar a curva de queima do motor.

Nesse momento serão realizados os cálculos que envolvem as propriedades e características físicas do grão propelente, bem como as dimensões da câmara de combustão e do bocal.

Conforme dito anteriormente, a pressão na câmara é causada pelos produtos liberados durante a combustão, processo que está diretamente relacionado com o empuxo resultante do motor foguete. Por isso é essencial compreender como os fluidos se comportam após serem liberados.

Sendo assim, inicialmente é necessário calcular as propriedades químicas e físicas do grão propelente, ou seja, iremos determinar o espaço disponível para alocar o propelente e assim será possível definir o tamanho do grão, bem como o espaço a ser ocupado pelos fluidos no processo de ignição.

O espaço para alocar o propulsor é definido como o volume interno da câmara de combustão (V_c). Além disso, considerando que o volume do cilíndro pode ser calculado através da área da base e do comprimento, para encontrar o volume inicial do grão (V_g) é necessário definir o comprimento inicial do grão (L_{go}). Dessa forma encontramos as seguintes relações:

$$V_c = \frac{\pi}{4} \cdot {D_c}^2 \cdot L_c \tag{8}$$

$$L_{go} = N \cdot L_o \tag{9}$$

$$V_g = \frac{\pi}{4} \cdot ({D_o}^2 - {d_o}^2) \cdot L_{go} \tag{10}$$

A quantidade de produtos gerados no processo de ignição também será influenciada pelas propriedades químicas do propulsor. Isso porque a massa inicial de propelente (m_g) é diretamente proporcional à densidade atual do grão (ρ_g), valor determinado pelo produto entre a densidade ideal e a densidade real do grão, e considerando que a densidade é fundamentalmente definida como a razão entre a massa e o volume podemos encontrar a massa inicial do grão em quilogramas através da equação abaixo.

$$m_g = \rho_g \cdot \frac{V_g}{10^6} \tag{11}$$

Além disso, a fração de volume utilizado no motor foguete (V_l) pode ser determinada pela razão entre o volume do grão propelente (V_g) e da câmara de combustão (V_c), parâmetro importante que interfere na maneira como ocorre o escoamento, bem como a velocidade com que os produtos da combustão se deslocam.

$$V_l = \frac{V_g}{V_c} \tag{12}$$

É notável que quanto maior é a área de queima total do grão, maior é o empuxo resultante e por isso é imprescindível determinar a área total de queima levando em consideração as superfícies inibidas para a determinação do Kn do motor.

Para isso, as superfícies de queima são classificadas de acordo com a área do

grão que pode ou não ser inibida, são elas: área final de queima (A_{beo}), área de queima do núcleo (A_{bco}), área de queima da superfície externa (A_{bso}) e área total de queima (A_{bo}).

$$A_{beo} = N \cdot 2 \cdot ei \cdot \frac{\pi}{4} \cdot (D_o{}^2 - d_o{}^2) \quad (13)$$

$$A_{bco} = N \cdot ci \cdot \pi \cdot d_o \cdot L_o \quad (14)$$

$$A_{bso} = N \cdot osi \cdot \pi \cdot D_o \cdot L_o \quad (15)$$

$$A_{bo} = A_{beo} + A_{bco} + A_{bso} \quad (16)$$

Acima é possível encontrar as relações utilizadas para o cálculo da superfície de queima, conforme explicado anteriormente. Entretanto é importante notar que nesses cálculos são utilizadas as variáveis ei, ci e osi, que foram determinadas anteriormente quando foi definido quais seriam as superfícies inibidas. Por esse motivo, essas variáveis são sempre definidas com o valor de 1 ou zero, assim quando a superfície é inibida o valor da variável será zero e, consequentemente, a área de queima será zero também.

```
#Final da área de queima inicial
Abeo = N*2*ei*math.pi/4*(Do**2-do**2)
print("O final da área de queima inicial é ""%.0f"% Abeo, "mm²")

#Área de queima inicial do núcleo
Abco = N*ci*math.pi*do*Lo
print("A área de queima inicial do núcleo é ""%.0f"% Abco, "mm²")

#Área de queima da superfície externa inicial
Abso = N*osi*math.pi*Do*Lo
print("A área de queima inicial da superfície externa é ""%.0f"% Abso, "mm²")

#Área de queima total inicial
Abo = Abeo+Abco+Abso
print("Área de queima total inicial é ""%.0f"% Abo, "mm²")
```

A pressão máxima da câmara de combustão é um fator importante a ser determinado, pois ele influencia diretamente no empuxo máximo resultante que será definido como $Kn_{máx}$, assim para encontra-lo utilizaremos a pressão máxima definida anteriormente através da equação:

$$Kn = a + b \cdot P + c \cdot P^2 + d \cdot P^3 + e \cdot P^4 + f \cdot P^5 + g \cdot P^6 \quad (17)$$

Em sua planilha, Richard Nakka disponibiliza uma tabela que relaciona o tipo de propelente escolhido e a pressão máxima alcançada no motor, com os coeficientes utilizados no cálculo.

Figura 12: Coeficientes para cálculo de Kn

	Polynomial coefficients for Least Squares Fit of Kn as fcn of Pressure data										
	Kn init	order	a	b	c	d	e	f	g	*calc Kn*	*Range incr*
KNDX		3	43,500	0,24168	50,484	-9,9115				234	150-400 psi
		2	163,80	22,224	-0,2240					231	450-850 psi
		4	5095,0	-2460,4	453,48	-35,532	1,0175			860	900-1300 psi
KNSB fine		5	34,500	-10,975	69,667	-19,664	2,1478	-8,146E-02		259	0-1300 psi
KNSB coarse		2	38,000	92,391	-2,0438					305	0-1300 psi
KNSU		2	32,954	44,108	-1,1025					159	0-1300 psi
KNER coarse		5	50,410	211,194	-47,7976	8,779018	-0,830282	0,030519		440	0-1300 psi
KNMN coarse		2	38,000	92,391	-2,0438					305	0-1300 psi

Fonte: Planilha SRM – Richard Nakka

Na imagem acima é possível ver a relação entre o tipo de grão e os coeficientes a serem utilizados, e assim como feito anteriormente, as informações contidas na tabela foram adaptadas em uma nova planilha com o objetivo de facilitar a interpretação dos dados pelo código, processo que é apresentado na imagem abaixo.

Figura 13: Planilha Kn max

Propelentes	order	a	b	c	d	e	f	g
KNDX 1	1,1	43,5000	0,2417	50,4841	-9,9115	0,0000	0,0000	0,0000
KNDX 2	1,2	163,8000	22,2241	-0,2240	0,0000	0,0000	0,0000	0,0000
KNDX 3	1,3	5095,0000	-2460,4260	453,4805	-35,5324	1,0175	0,0000	0,0000
KNSB fine	2	34,5000	-10,9750	69,6670	-19,6640	2,1478	-0,0815	0,0000
KNSB coarse	3	38,0000	92,3910	-2,0438	0,0000	0,0000	0,0000	0,0000
KNSU	4	32,9539	44,1079	-1,1025	0,0000	0,0000	0,0000	0,0000
KNER coarse	5	50,4104	211,1940	-47,7976	8,7790	-0,8303	0,0305	0,0000
KNMN coarse	6	38,0000	92,3910	-2,0438	0,0000	0,0000	0,0000	0,0000

Fonte: própria

No código a variável *fueltype*, definida anteriormente, será essencial para relacionar o grão propelente com os coeficientes a serem utilizados na equação do Kn, o que é mostrado abaixo.

```
#Kn max
Kn_eq = pd.read_excel("Kn max.xlsx")
filtro_kn = Kn_eq.set_index("order")
a = filtro_kn["a"].loc[fueltype]
b = filtro_kn["b"].loc[fueltype]
c = filtro_kn["c"].loc[fueltype]
d = filtro_kn["d"].loc[fueltype]
e = filtro_kn["e"].loc[fueltype]
f = filtro_kn["f"].loc[fueltype]
g = filtro_kn["g"].loc[fueltype]
Kn = a+b*P+c*P**2+d*P**3+e*P**4+f*P**5+g*P**6
print("O Kn máximo é ""%.0f"% Kn)
```

É importante citar que apenas para o propelente KNDX os coeficientes mudam de acordo com a pressão máxima inserida, por isso foi utilizado a condicional garantindo que a variável *fueltype* seja ressignificada de acordo com o valor determinado para a variável P, para todos os casos em que o grão é o KNDX.

```
P = filtro["Mpa"].loc[Press]
if P < 3 and fueltype == 1:
    fueltype = 1.1
elif P > 3 and P < 6 and fueltype == 1:
    fueltype = 1.2
elif P > 6 and fueltype == 1:
    fueltype = 1.3
```

4.1.2.3 Determinação do Kn

Como dito anteriormente o gráfico de Kn em função da regressão de entrelaçamento (x) é equivalente ao gráfico de empuxo em função do tempo, assim a curva de queima do grão propelente poderá ser definida determinando um desses gráficos.

O cálculo do Kn é realizado através da razão entre a área da tubeira e a área total de queima, entretanto para encontrar esses valores é necessário determinar as dimensões do grão durante o processo de queima. Para isso, Nakka

desenvolveu uma tabela que inclui uma sequência de cálculos que possibilitam definir os valores para o Kn em função da taxa de regressão do entrelaçamento.

Na figura abaixo é possível visualizar a tabela criada por Nakka. Nela cada coluna representa um cálculo que é repetido para todas as suas linhas e os valores subsequentes dependem da determinação dos valores das colunas anteriores. Pensando nisso, é essencial encontrar os possíveis resultados para a regressão do entrelaçamento para obter os resultados das próximas colunas.

Figura 14: Tabela gráfico Kn vs Regreção do Entrelaçamento

xinc	0,740 mm		Tabela Kn								
Intervalo	x	d	D	L	tweb	Abe	Abc	Abs	Ab total	At	Kn
	mm	mm	mm	mm	mm	mm²	mm²	mm²	mm²	mm²	
0	0,00	20,00	94,00	380,0	37,00	53005	23876	0	76881	320,8	239,7
1	0,74	21,48	94,00	374,1	36,26	52619	25243	0	77863	320,8	242,7
2	1,48	22,96	94,00	368,2	35,52	52206	26556	0	78762	320,8	245,5
3	2,22	24,44	94,00	362,2	34,78	51765	27813	0	79578	320,8	248,1
4	2,96	25,92	94,00	356,3	34,04	51297	29015	0	80312	320,8	250,4
5	3,70	27,40	94,00	350,4	33,30	50801	30162	0	80963	320,8	252,4
6	4,44	28,88	94,00	344,5	32,56	50278	31254	0	81532	320,8	254,2
7	5,18	30,36	94,00	338,6	31,82	49727	32291	0	82018	320,8	255,7
8	5,92	31,84	94,00	332,6	31,08	49148	33273	0	82422	320,8	256,9
9	6,66	33,32	94,00	326,7	30,34	48542	34200	0	82743	320,8	257,9
10	7,40	34,80	94,00	320,8	29,60	47909	35072	0	82981	320,8	258,7
11	8,14	36,28	94,00	314,9	28,86	47248	35889	0	83137	320,8	259,2
12	8,88	37,76	94,00	309,0	28,12	46560	36651	0	83210	320,8	259,4
13	9,62	39,24	94,00	303,0	27,38	45844	37358	0	83201	320,8	259,4
14	10,36	40,72	94,00	297,1	26,64	45100	38009	0	83109	320,8	259,1
15	11,10	42,20	94,00	291,2	25,90	44329	38606	0	82935	320,8	258,5
16	11,84	43,68	94,00	285,3	25,16	43530	39147	0	82678	320,8	257,7
17	12,58	45,16	94,00	279,4	24,42	42704	39634	0	82338	320,8	256,7
18	13,32	46,64	94,00	273,4	23,68	41850	40065	0	81916	320,8	255,4
19	14,06	48,12	94,00	267,5	22,94	40969	40442	0	81411	320,8	253,8
20	14,80	49,60	94,00	261,6	22,20	40061	40763	0	80824	320,8	252,0
21	15,54	51,08	94,00	255,7	21,46	39124	41030	0	80154	320,8	249,9
22	16,28	52,56	94,00	249,8	20,72	38161	41241	0	79401	320,8	247,5
23	17,02	54,04	94,00	243,8	19,98	37169	41397	0	78566	320,8	244,9
24	17,76	55,52	94,00	237,9	19,24	36150	41498	0	77649	320,8	242,1
25	18,50	57,00	94,00	232,0	18,50	35104	41544	0	76649	320,8	238,9

Fonte: Planilha SRM – Richard Nakka

Após análise foi possível compreender que os valores presentes na coluna x são determinados através de uma progressão aritmética com a razão sendo a variável x_{inc} e o termo inicial sempre como zero. Além disso, nota-se que a coluna nomeada como T_{web} é responsável por calcular a espessura do entrelaçamento em função da taxa de regressão, o que pode ser visualizado através da seguinte relação:

$$Tweb = \frac{D-d}{2} \tag{18}$$

A partir dessas informações um outro padrão foi percebido, a tabela completa contém cinquenta linhas e o quinquagésimo termo da progressão deve sempre ser igual ao primeiro valor da coluna T_{web}, e com essas informações encontra-se uma relação para a determinar a variável x_{inc}:

$$x_{\text{inc}} = \frac{D_0-d_0}{100\cdot(ci+osi)} \tag{19}$$

Após encontrar o valor do x_{inc} é possível determinar os novos valores para as dimensões do grão em função da taxa de regreção (x). Assim, considerando d como

o diâmetro interno, D como diâmetro externo e L como comprimento, obtem-se as seguintes relações:

$$d = do + (ci \cdot 2 \cdot x) \quad (20)$$
$$D = Do - (osi \cdot 2 \cdot x) \quad (21)$$
$$L = Lgo - (ei \cdot 2 \cdot x) \quad (22)$$

Como feito ateriormente para determinar a área de queima total do grão, utilizamos os valores encontrados nas funções acima para determinar a área de queima da superfície externa (Ab_s), do núcleo (Ab_c), e da superfície final (Ab_e) em função de x, sendo a área de queima total (Ab_t) definida como o somatório de toda a superfície de queima.

$$Ab_c = ci \cdot \pi \cdot d \cdot L \quad (23)$$
$$Ab_s = osi \cdot \pi \cdot D \cdot L \quad (24)$$
$$Ab_e = ei \cdot N \cdot 2 \cdot \frac{\pi}{4} \cdot (D^2 - d^2) \quad (25)$$
$$Ab_t = Ab_c + Ab_s + Ab_e \quad (26)$$

Visto que os cálculos realizados devem ser repetidos continuamente para cada uma das linhas da tabela, foi utilizada a função “for”, responsável por criar um looping limitado a uma quantidade de vezes pré definida. Além disso, os resultados foram armazenados em listas que futuramente serão utilizadas para a formação de uma base de dados, como é apresentado na imagem abaixo.

```
#Tabela
x = [0]
x_inc = (Do-do)/(100*(ci+osi))
for i in range(1,51):
    xf = x_inc*i
    x.append(xf)

d = []
for i in range(0,51):
    df = do+(ci*2*(x[i]))
    d.append(df)

D = []
for i in range(0,51):
    Df = Do-(osi*2*(x[i]))
    D.append(Df)

L = []
for i in range(0,51):
    Lf = Lgo-(ei*N*2*(x[i]))
    L.append(Lf)

T_web = []
for i in range(0,51):
    T_webf = (D[i]-d[i])/2
    T_web.append(T_webf)

Abc = []
for i in range(0,51):
    Abcf = (ci*math.pi*d[i]*L[i])
    Abc.append(Abcf)

Abs = []
for i in range(0,51):
    Absf = (osi*math.pi*D[i]*L[i])
    Abs.append(Absf)

Abe = []
for i in range(0,51):
    Abef = ((ei*2*N*math.pi/4*(D[i]**2-d[i]**2)))
    Abe.append(Abef)
```

Através desse processo é realizado uma análise dos valores encontrados, o que possibilita a determinação do valor máximo para a área de queima durante o processo de regressão, fator definido como Ab_{tmax} e que é extremamente importante para definir as características de queima do motor.

O projeto do bocal é essencial para um bom desempenho do motor foguete, assim é importante que a geometria da tubeira seja definida levando em consideração todas as informações definidas anteriormente sobre as características do grão e da câmara de combustão.

Para compreender qual deve ser a área inicial da secção transversal da garganta (At_o) é necessário levar em consideração a área total de queima máxima e o Kn máximo. Assim definimos a variável At_o como:

$$At_o = \frac{Ab_{tmax}}{Kn_{max}} \tag{27}$$

Sabendo que a àrea da circunferência é o produto entre o número de pi e o raio da circunferência ao quadrado, e assumindo que o diâmetro será sempre o dobro do tamanho do raio, é possível calcular o diâmetro inicial da garganta (Dt_o) através da seguinte relação:

$$Dt_o = \sqrt{\frac{4 \cdot At_o}{\pi}} \tag{28}$$

Nos casos onde há erosão no bocal para determinar o diâmetro final da garganta (Dt_{final}) deve ser levado em consideração o diâmetro inicial e a erosão final, como mostrado abaixo.

$$Dt_{final} = Dt_o + e \tag{29}$$

Após definir os detalhes da geometria do bocal podemos encontrar a área total da garganta em função da regressão do entrelaçamento (At), entretanto nos casos em que não ocorre erosão no bocal a área deve se manter constante e sabendo que T_{webo} será o valor de T_{web} quando $x = 0$, encontramos a seguinte equação:

$$At = \frac{\pi}{4} \cdot \left(Dt_o + e \cdot \frac{T_{webo} - T_{web}}{T_{webo}} \right)^2 \tag{30}$$

Por fim, é possível calcular o Kn em função da variável x, informação base para a construção do gráfico e determinação da curva de queima do grão.

$$Kn = \frac{Ab_t}{At} \tag{31}$$

4.1.2.4 Informações do Gráfico

Para construir o gráfico foram utilizadas as equações definidas previamente para determinar a função de Kn e de espessura de entrelaçamento com relação a

taxa de regressão, o que pode ser visto na figura abaixo.

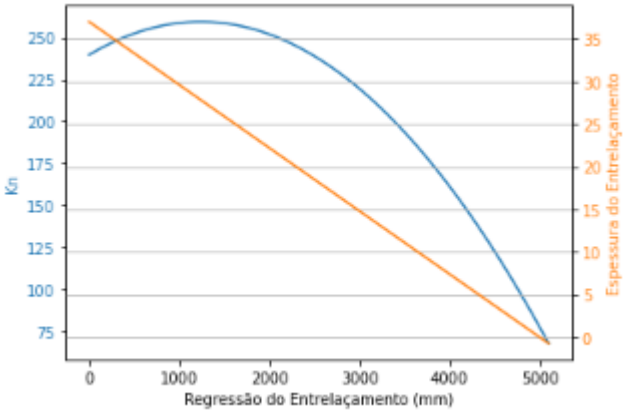
```
#Informações Gráfico
def Kn_cal(x_inc,z):
    D_cal = Do-osi*2*x_inc*z
    d_cal = do+ci*2*x_inc*z
    Dt_cal = Do-osi*2*x_inc
    dt_cal = do+ci*2*x_inc
    L_cal = Lgo-ei*2*N*x_inc*z
    two_cal = (Dt_cal-dt_cal)/2
    tweb_cal = (D_cal-d_cal)/2
    Abe_cal = (ei*2*N*math.pi)/4*(D_cal**2-d_cal**2)
    Abc_cal = ci*math.pi*d_cal*L_cal
    Abs_cal = osi*math.pi*D_cal*L_cal
    Abt_cal = Abe_cal+Abc_cal+Abs_cal
    At_cal = math.pi/4*(Dto+ef*(two_cal-tweb_cal)/two_cal)**2
    return Abt_cal/At_cal
def t_web_cal(x_inc,z):
    D_cal = Do-osi*2*x_inc*z
    d_cal = do+ci*2*x_inc*z
    return (D_cal-d_cal)/2
```

Além disso, como dito anteriormente foi utilizado a biblioteca matplotlib.pyplot, módulo que disponibiliza diversos modelos de gráficos (https://matplotlib.org/stable/tutorials/introductory/index.html), para determinar as características do gráfico, bem como o tipo de gráfico, as cores das linhas e os rótulos dos eixos do gráfico, o que pode ser visto abaixo.

```
zig = np.arange(0,51,0.01)

fig, ax1 = plt.subplots()

color = 'tab:blue'
ax1.set_xlabel('Regressão do Entrelaçamento (mm)')
ax1.set_ylabel('Kn', color=color)
ax1.plot(Kn_cal(x_inc, zig), color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()

color = 'tab:orange'
ax2.set_ylabel('Espessura do Entrelaçamento', color=color)
ax2.plot(t_web_cal(x_inc,zig), color=color)
ax2.tick_params(axis = 'y', labelcolor=color)

fig.tight_layout()
plt.grid()
plt.show()
```

Dessa forma, o programa é capaz de construir o gráfico com a curva de queima, o que é visível a seguir:

Figura 15: Gráfico construído pelo programa



Fonte: própria

Por fim, é disponibilizada ao usuário a tabela que é utilizada como base de dados para a construção do gráfico, validando os valores mostrados e garantindo que o usuário tenha acesso ao máximo de informações possíveis. Isso é feito através das listas que armazenaram os valores calculados para cada linha da tabela, e que agora podem ser apresentadas em formato semelhante ao encontrado na figura 9.

```
data = {'x': x,
        'd': d,
        'D': D,
        'L': L,
        'tweb': T_web,
        'Abe': Abe,
        'Abc': Abc,
        'Abs': Abs,
        'Abt': Abt,
        'At': At,
        'Kn': Knl}
tabela_completa = pd.DataFrame(data)
print(tabela_completa)
```

4.2 Disponibilizando o programa

O código e os arquivos que foram desenvolvidos durante essa pesquisa têm como objetivo facilitar o acesso à ferramentas que ajudam no desenvolvimento de foguetes experimentais, portanto era importante que todo o material fosse disponibilizado de maneira eficaz.

Dessa forma, foi selecionada a plataforma GitHub para disponibilizar todo o programa desenvolvido, visto que esse web site tem como objetivo facilitar o contato entre programadores que estão dispostos a compartilhar seus códigos de forma que eles possam ser aprimorados.

Assim, o código completo pode ser encontrado nos apêndices desse relatório e também através do link https://github.com/juliabelucci/py_Rocket.

5 Resultados

A proposta desenvolvida nesse trabalho possibilitou o entendimento das planilhas de Richard Nakka, do funcionamento de motores-foguete e, consequentemente, do processo de construção de foguetes experimentais. Além disso, permitiu desenvolver a capacidade de criar programas utilizando a linguagem de programação Python de maneira prática e eficaz simplificando processos.

Dar continuidade ao pacote pyRocket é essencial para alcançar uma maior performance no desenvolvimento de motores foguete, além disso a criação de programas que facilitem a criação de foguetes experimentais abre porta para que mais pessoas consigam participar de competições de foguete modelismo incentivando o desenvolvimento científico e o crescimento da área no território brasileiro.

O resultado da pesquisa pode ser encontrado na plataforma GitHub, como

apresentados no item 4.2.

6 Perspectivas futuras

Como dito anteriormente, Richard Nakka possui diversas planilhas com dados que se complementam e apenas uma parte delas foram transformadas até o atual momento, portanto o ideal é que mais arquivos sejam traduzidos para linguagem Python possibilitando a automação e integração completa entre eles, o que tornará o processo de criação de foguetes experimentais cada vez mais simples.

Além disso, é essencial que todos os códigos criados sejam reunidos em um único módulo, o que facilitaria o acesso a todo o trabalho realizando evitando perca de informações e a criação de uma ferramenta eficiente.

Referências

[1] Richard Nakka. What is this web site all about?. Richard Nakka's Experimental Rocketry Web Site, dez 2021. Acesso em: 04 de Dez. de 2021.

[2] Richard Nakka. Who I Am. Richard Nakka's Experimental Rocketry Web Site, dez 2021. Acesso em: 10 de Fev. de 2022.

[3] NASA. A Pictorial History of Rockets. Nasa.gov. Acesso em: 10 de Fev. de 2022.

[4] Steven J. Dick. The Importance of Exploration. Nasa.gov. Acesso em: 10 de Fev. de 2022.

[5] Giorgio Giacaglia. A Indústria Aeroespacial: Questões Econômicas, Tecnológicas e Sociais. Estudos Avançados, vol. 8, Apr. 1994, pp. 42-49. Acesso em: 10 de Fev. de 2022.

[6] Michael F. Lembeck. Why Space Exploration Is Important to the United States. Space.com, jun 2006. Acesso em: 26 de Jan. de 2022.

[7] Lucas SCHLOSSMACHER. Desenvolvimento de motores-foguete para espaçomodelos. Trabalho de conclusão de curso (bacharelado em engenharia mecânica) – Universidade Federal do Paraná, Curitiba, 2015. Acesso em: 25 de Nov. 2020.

[8] Richard Nakka. Solid rocket motor theory. *Richard Nakka's Experimen- tal Rocketry Web Site*, jul 2001. Acesso em: 10 de Fev. de 2022.

[9] History of Python. Geeks for geeks, mai 2022. Acesso em: 23 de Ago. de 2022.

[10] Mateus Ferreira Torres GALVÃO. Projeto estrutural de um motor-foguete acadêmico a combustível sólido. *Trabalho de conclusão de curso (bacharelado em engenharia mecânica) – Universidade Federal do Rio Grande do Norte, Natal*, jul 2018. [Orientador: Prof. Dr. João Carlos Arantes Costa Júnior].. Acesso em: 12 de Fev. de 2022.

[11] João Felisardo MACHADO. Utilizando as ciências espaciais e a astronáutica na construção de atividades práticas em ensino de física. *Dissertação de pós-graduação (Mestrado em física e ciências naturais) –*

Universidade Federal do Rio Grande do Norte, Natal, 42, nov 2006. [Orientador: Profº. Dr. Gilvan Luiz Borba]. Acesso em: 14 de Fev. de 2022.

[12]André Luíz Alvez, Anderson Nunes Paneto, Kaio Alan Littike, Sérgio Souza Bento, and Carlos Henrique Marchi. Minifoguete a propelente sólido: aspectos teóricos e propostas experimentais para o ensino de física. *Revista Brasileira de Ensino de Física*, dez 2021.

[13]Python Software Foundation. math — mathematical functions, set 2022. Acesso em: 18 de Set. de 2022.

[14]Python Data Analysis Library. pandas; set 2022. Acesso em: 18 de Set de 2022.

[15]Python Matplotlib Documentation. matplotlib tutorials; set 2022. Acesso em: 18 de Set. de 2022.

[16]Python Numpy. About us; set 2022. Acesso em: 18 de Set. de 2022.

[17]Cliff Lethbridge. History of rocketry chapter 1 ancient times through the 17th century, dez 2021. Acesso em: 22 de dez. de 2020.

[18]Richard Nakka. Planilha SRM, set 2022. Acesso em: 18 de Set. de 2022.

[19]Rodrigo Santana. Plotando gráficos de um jeito fácil com python, dez 2020. Acesso em: 20 de Jan. de 2022.

[20]Python. Python package index, set 2021. Acesso em: 30 de Set. de 2021.

7 APÊNDICE A - Código SRM – Data and Kn

```
#Biblioteca utilizada para acessar base de dados
import pandas as pd
#Biblioteca utilizada para os cálculos
import math
#Bibliotecas utilizadas para plotar o gráfico
import matplotlib.pyplot as plt
import numpy as np

title = input("Coloque um título: ")

#Câmara do Motor
print("Dados da Câmara do Motor")
print("")
#Diâmetro interno da câmara
Dc = int(input("Diâmetro interno da câmara em milímetro: "))
#Comprimento interno da câmara
Lc = float(input("Comprimento interno da câmara em milímetro: "))
#Volume da câmara vazia
Vc = (math.pi/4)*(Dc**2)*(Lc)
print("O Volume da câmara é ""%.0f"% Vc,"mm³")
print("")

#Grão Propelente
print("Grão do Propelente")
print("")
#Tipo de grão propelente
fueltype = 0
while fueltype == 0:
    prop = input("""Escreva o tipo do propelente:
    Escolha entre as opções
    KNDX
    KNSB fine
    KNSB coarse
    KNSU
    KNER coarse
    KNMN coarse
    """)
    if prop == "KNDX":
        fueltype = 1
    elif prop == "KNSB fine":
        fueltype = 2
    elif prop == "KNSB coarse":
        fueltype = 3
    elif prop == "KNSU":
        fueltype = 4
    elif prop == "KNER" or prop == "KNER coarse":
        fueltype = 5
```

```
    elif prop == "KNMN" or prop == "KNMN coarse":
        fueltype = 6
    else:
        fueltype = 0
        print("Resposta inválida")
#Diâmetro inicial externo do grão
Do = float(input("Diâmetro externo inicial em milímetro: "))
#Diâmetro inicial do núcleo do grão
do = int(input("Diâmetro do núcleo inicial em milímetro: "))
#Comprimento inicial do segmento do grão
Lo = float(input("Comprimento do segmento inicial em milímetro: "))
#Número de segmentos do grão
N = int(input("Número de segmentos: "))

#Inibidores
#Superfície externa
osi = -1
while osi < 0:
    SE = input("A superfície externa é Exposta ou Inibida? ")
    if SE == "Exposta":
        osi = 1
    elif SE == "Inibida":
        osi = 0
    else:
        osi = osi
        print("Resposta inválida")
#Superfícia do núcleo
ci = -1
while ci < 0:
    SN = input("A superfície do núcleo é Exposta ou Inibida? ")
    if SN == "Exposta":
        ci = 1
    elif SN == "Inibida":
        ci = 0
    else:
        osi = osi
        print("Resposta inválida")
#Superfície final
ei = -1
while ei < 0:
    FS = input("O final da superfície é Exposta ou Inibida? ")
    if FS == "Exposta":
        ei = 1
    elif FS == "Inibida":
        ei = 0
    else:
        osi = osi
        print("Resposta inválida")
print("")
```

```
#Comprimento do grão inicial
Lgo = N*Lo
print("O comprimento do grão inicial é", Lgo, "mm")

#Volume do grão inicial
Vgd = ((Do**2)-(do**2))*Lgo
Vg = (math.pi/4)*Vgd
print("O volume do grão inicial é ""%.0f"% Vg, "mm³")

#Fração de volume utilizada
Vl = Vg/Vc
print("A fração de volume utilizada é ""%.3f"% Vl)

#Densidade ideal do grão
Dados_propelente = pd.read_excel("Densidade ideal.xlsx")
pig = Dados_propelente[fueltype].loc[0]
print("A densidade ideal do grão é ", pig, "g/cm³")

#Relação de densidade (real/ideal)
dg = float(input("Densidade real do grão "))

#Atual densidade do grão
pg = pig*dg
print("A atual densidade do grão é ""%.3f"%pg,"g/cm³")

#Massa do grão inicial
mg = pg*(Vg/10**6)
print("A massa do grão do inicial é ""%.3f"% mg, "kg")

#Final da área de queima inicial
Abeo = N*2*ei*math.pi/4*(Do**2-do**2)
print("O final da área de queima inicial é ""%.0f"% Abeo, "mm²")

#Área de queima inicial do núcleo
Abco = N*ci*math.pi*do*Lo
print("A área de queima inicial do núcleo é ""%.0f"% Abco, "mm²")

#Área de queima da superfície externa inicial
Abso = N*osi*math.pi*Do*Lo
print("A área de queima inicial da superfície externa é ""%.0f"% Abso,
 "mm²")

#Área de queima total inicial
Abo = Abeo+Abco+Abso
print("Área de queima total inicial é ""%.0f"% Abo, "mm²")

#Alvo Meop
Press = (input("Digite a pressão com a unidade de medida como Mpa ou p
si: "))
Pressures = pd.read_excel("Pressao.xlsx")
```

```
filtro = Pressures.set_index("Pressure")
P = filtro["Mpa"].loc[Press]
if P < 3 and fueltype == 1:
    fueltype = 1.1
elif P > 3 and P < 6 and fueltype == 1:
    fueltype = 1.2
elif P > 6 and fueltype == 1:
    fueltype = 1.3
ef = float(input("Ocorre erosão no bocal? Caso a resposta seja negativ
a insira o valor zero: "))

#Kn max
Kn_eq = pd.read_excel("Kn max.xlsx")
filtro_kn = Kn_eq.set_index("order")
a = filtro_kn["a"].loc[fueltype]
b = filtro_kn["b"].loc[fueltype]
c = filtro_kn["c"].loc[fueltype]
d = filtro_kn["d"].loc[fueltype]
e = filtro_kn["e"].loc[fueltype]
f = filtro_kn["f"].loc[fueltype]
g = filtro_kn["g"].loc[fueltype]
Kn = a+b*P+c*P**2+d*P**3+e*P**4+f*P**5+g*P**6
print("O Kn máximo é ""%.0f"% Kn)

#Tabela
x = [0]
x_inc = (Do-do)/(100*(ci+osi))
for i in range(1,51):
    xf = x_inc*i
    x.append(xf)

d = []
for i in range(0,51):
    df = do+(ci*2*(x[i]))
    d.append(df)

D = []
for i in range(0,51):
    Df = Do-(osi*2*(x[i]))
    D.append(Df)

L = []
for i in range(0,51):
    Lf = Lgo-(ei*N*2*(x[i]))
    L.append(Lf)

T_web = []
for i in range(0,51):
    T_webf = (D[i]-d[i])/2
    T_web.append(T_webf)
```

```
Abc = []
for i in range(0,51):
    Abcf = (ci*math.pi*d[i]*L[i])
    Abc.append(Abcf)

Abs = []
for i in range(0,51):
    Absf = (osi*math.pi*D[i]*L[i])
    Abs.append(Absf)

Abe = []
for i in range(0,51):
    Abef = ((ei*2*N*math.pi/4*(D[i]**2-d[i]**2)))
    Abe.append(Abef)

Abt = []
for i in range(0,51):
    Abtf = (ci*math.pi*d[i]*L[i]+osi*math.pi*D[i]*L[i]+(ei*2*N*math.pi
/4*(D[i]**2-d[i]**2)))
    Abt.append(Abtf)

Abtm = max(Abt)

#Bocal
print("Informações do Bocal")
Ato = Abtm/Kn
print("A área da secção transversal da garganta inicial em mm² é ""%.0
f"%Ato)
Dto = math.sqrt((4*Ato)/math.pi)
print("O diâmetro da garganta inicial é ""%.3f"%Dto)
Dtofinal = Dto + ef
print("O diâmetro da garganta final é ""%.2f"%Dtofinal)

#Final Tabela
At = []
for i in range(0,51):
    Atf = math.pi/4*(Dto+ef*(T_web[0]-T_web[i])/T_web[0])**2
    At.append(Atf)

Knl = []
for i in range(0,51):
    Knlf = (Abt[i]/At[i])
    Knl.append(Knlf)

#Informações Gráfico
def Kn_cal(x_inc,z):
    D_cal = Do-osi*2*x_inc*z
    d_cal = do+ci*2*x_inc*z
    Dt_cal = Do-osi*2*x_inc
```

```
    dt_cal = do+ci*2*x_inc
    L_cal = Lgo-ei*2*N*x_inc*z
    two_cal = (Dt_cal-dt_cal)/2
    tweb_cal = (D_cal-d_cal)/2
    Abe_cal = (ei*2*N*math.pi)/4*(D_cal**2-d_cal**2)
    Abc_cal = ci*math.pi*d_cal*L_cal
    Abs_cal = osi*math.pi*D_cal*L_cal
    Abt_cal = Abe_cal+Abc_cal+Abs_cal
    At_cal = math.pi/4*(Dto+ef*(two_cal-tweb_cal)/two_cal)**2
    return Abt_cal/At_cal
def t_web_cal(x_inc,z):
    D_cal = Do-osi*2*x_inc*z
    d_cal = do+ci*2*x_inc*z
    return (D_cal-d_cal)/2

zig = np.arange(0,51,0.01)

fig, ax1 = plt.subplots()

color = 'tab:blue'
ax1.set_xlabel('Regressão do Entrelaçamento (mm)')
ax1.set_ylabel('Kn', color=color)
ax1.plot(Kn_cal(x_inc, zig), color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()

color = 'tab:orange'
ax2.set_ylabel('Espessura do Entrelaçamento', color=color)
ax2.plot(t_web_cal(x_inc,zig), color=color)
ax2.tick_params(axis = 'y', labelcolor=color)

fig.tight_layout()
plt.grid()
plt.show()

#Tabela plot
print(title)
x = [round(i,2) for i in x]
d = [round(i,2) for i in d]
D = [round(i,2) for i in D]
L = [round(i,1) for i in L]
T_web = [round(i,2) for i in T_web]
Abe = [round(i,0) for i in Abe]
Abc = [round(i,0) for i in Abc]
Abs = [round(i,0) for i in Abs]
Abt = [round(i,0) for i in Abt]
At = [round(i,1) for i in At]
Knl = [round(i,1) for i in Knl]
data = {'x': x,
```

```
            'd': d,
            'D': D,
            'L': L,
            'tweb': T_web,
            'Abe': Abe,
            'Abc': Abc,
            'Abs': Abs,
            'Abt': Abt,
            'At': At,
            'Kn': Knl}
tabela_completa = pd.DataFrame(data)
print(tabela_completa)
```

8 APÊNDICE B – Kn max.xlsx

Propelentes	order	a	b	C	d	e	f	g
KNDX 1	1,1	43,5000	0,2417	50,4841	-9,9115	0,0000	0,0000	0,0000
KNDX 2	1,2	163,8000	22,2241	-0,2240	0,0000	0,0000	0,0000	0,0000
KNDX 3	1,3	5095,0000	-2460,4260	453,4805	-35,5324	1,0175	0,0000	0,0000
KNSB fine	2	34,5000	-10,9750	69,6670	-19,6640	2,1478	-0,0815	0,0000
KNSB coarse	3	38,0000	92,3910	-2,0438	0,0000	0,0000	0,0000	0,0000
KNSU	4	32,9539	44,1079	-1,1025	0,0000	0,0000	0,0000	0,0000
KNER coarse	5	50,4104	211,1940	-47,7976	8,7790	-0,8303	0,0305	0,0000
KNMN coarse	6	38,0000	92,3910	-2,0438	0,0000	0,0000	0,0000	0,0000

9 APÊNDICE C – Pressao.xlsx

Pressure	Mpa
9 Mpa	9,000
8.5 Mpa	8,500
8 Mpa	8,000
7.5 Mpa	7,500
7 Mpa	7,000
6.5 Mpa	6,500
6 Mpa	6,000
5.5 Mpa	5,500
5 Mpa	5,000

4.5 Mpa	4,500
4 Mpa	4,000
3.5 Mpa	3,500
3 Mpa	3,000
2.5 Mpa	2,500
2 Mpa	2,000
1.5 Mpa	1,500
1 Mpa	1,000
1300 psi	8,964
1250 psi	8,619
1200 psi	8,274
1150 psi	7,929
1100 psi	7,585
1050 psi	7,240
1000 psi	6,895
950 psi	6,550
900 psi	6,206
850 psi	5,861
800 psi	5,516
750 psi	5,171
700 psi	4,827
650 psi	4,482
600 psi	4,137
550 psi	3,792
500 psi	3,448
450 psi	3,103
400 psi	2,758
350 psi	2,413
300 psi	2,069
250 psi	1,724
200 psi	1,379
150 psi	1,034

10 APÊNDICE D – Densidade ideal.xlsx

Tipos de Propelente	simb.	unid	notas	1	2	3	4	5	6
Densidade da massa de grão, ideal	ρ_p	g/cm^3		1,879	1,841	1,841	1,889	1,820	1,854
Razão de calores específicos, 2-ph.	k_{2ph}	-	[1]	1,043	1,042	1,042	1,044	1,043	1,042
Razão de calores específicos, mistura	k	-	[2]	1,131	1,136	1,136	1,133	1,139	1,136
Peso molecular efetivo.	M	kg/kmol	[3]	42,39	39,90	39,90	41,98	38,78	39,83
Temperatura da câmara	To	K	[4]	1710	1600	1600	1720	1608	1616