UFABC Rocket Design

Núcleo de Pesquisas

pyRocket - Implementação de plataforma integrada de cálculo para desenvolvimento de foguetes experimentais em linguagem Python

Lucas Valentim Berganton Câmara

São Bernardo do Campo - SP

2021

Lucas Valentim Berganton Câmara

pyRocket - Implementação de plataforma integrada de cálculo para desenvolvimento de foguetes experimentais em linguagem Python

Projeto de (Iniciação Científica/PDPD) realizado no Núcleo de Pesquisas da UFABC Rocket Design

UFABC Rocket Design

Núcleo de Pesquisas

Orientador(a): Prof. Dr. Wallace Gusmão Ferreira

São Bernardo do Campo - SP

2021

“Per Aspera, Ad Astra”

Resumo

As competições de foguete-modelismo crescem cada vez mais no Brasil e no mundo. São eventos que provocam interesse e conhecimentos nas áreas relacionadas, como física, matemática, administração e outros, dentro de projetos espaciais, tanto pelos jovens quanto pelo público em geral. Atualmente, a equipe UFABC Rocket Design, e não somente ela, utiliza as planilhas construídas por Richard Nakka[1], um foguete-modelista amador que constrói, testa e reúne dados desde 1972, quando lançou seu primeiro foguete. Desse modo, suas ferramentas são precisas e confiáveis. Visando essa confiabilidade no material de Nakka, e o seu uso pelas equipes de competição, o projeto é objetivado na construção de uma ferramenta de cálculo em linguagem Python, baseada nas planilhas do mesmo, com o objetivo de facilitar o uso dos dados e da sua implementação em outros programas em Python. A relevância da construção dessa ferramenta é tal que, futuramente, conforme mais programas em Python sejam desenvolvidos, mais completo pode ficar essa ferramenta e, consequentemente, mais informações ela pode trazer para o cálculo de um foguete experimental, facilitando e tornando mais eficiente sua construção.

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

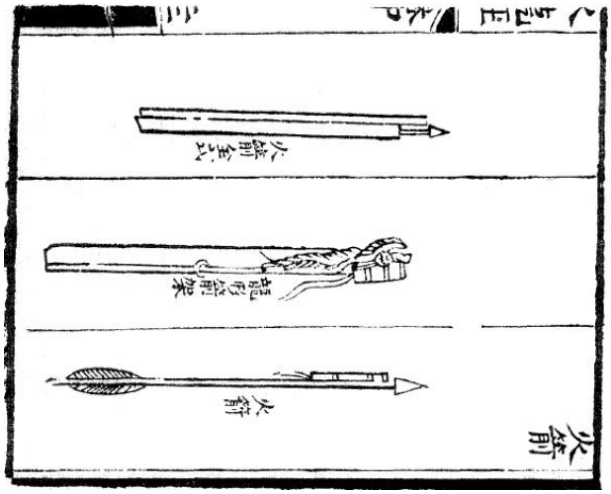
Sumário

1 Introdução

Segundo o dicionário Infopédia da Língua Portuguesa, um foguete é um projétil autopropulsionado. Eles são máquinas usadas para dar impulso suficiente para empurrar um objeto, seja ele um satélite, um explosivo ou até uma pessoa[2].

Sua origem remonta a datas próximas do século X pelos chineses, que montavam “flechas” com bambu e pólvora, como na figura 1. Eles utilizavam dessas ferramentas tanto para cerimônias religiosas como armas de guerra [3].

Figura 1: Flecha de fogo chinesa



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

É creditado que, em 1379, o italiano Muratori utilizou a palavra “rochetta” para descrever o tipo de “flechas” com propelente de pólvora usadas nos tempos medievais. Essa palavra foi traduzida para o inglês, como “rocket”, e para o português como foguete [3].

Já Newton, mesmo não intencionalmente, aplicando modelos matemáticos aos princípios da natureza descreveu como foguetes funcionam, em 1687. Ao explicar a terceira lei, sobre ação e reação, ele pode descrever como o impulso causado pela combustão do propelente gera uma força que tem uma reação oposta, movendo o foguete para frente. E, ele também teorizou que, se um objeto pudesse voar rápido o suficiente em uma altitude suficiente ele não iria cair, pois teria atingido a orbita terrestre[4].

Diante disso, e devido aos avanços da tecnologia, o foguete foi aprimorado com uma cápsula de metal, como na figura 2, e, como já era usado, participou de muito mais guerras, especialmente a Segunda guerra mundial (1939-45). Mas foi logo em seguida que os foguetes tiveram seu ápice.

A ideia de o homem chegar a Lua não é nova, há muito o ser humano tem uma fascinação por ela, como é possível ver no filme “Le Voyage dans la Lune” (Viagem à Lua), de Georges Méliès, que retrata a ida do homem à Lua de uma forma cômica.

Apesar de todas obras ficcionais o interesse era real, e foi na Guerra Fria (1947-1991) que a disputa entre Estados Unidos da América (EUA) e a União das Repúblicas Socialistas Soviéticas

Figura 2: Foguete V2 na 2ª Guerra Mundial

Fonte: https://bit.ly/3zsA5q8

(URSS), para mostrar quem era a maior potência, causou uma desenfreada busca por mostrar poderio militar e cultural [5].

Essa disputa desencadeou na corrida espacial que se iniciou na década de 50 e trouxe conquistas como a primeira nave espacial, o primeiro homem no espaço e o primeiro homem na lua[5], visível na figura 3.

Figura 3: Pouso da Apollo 11 na Lua

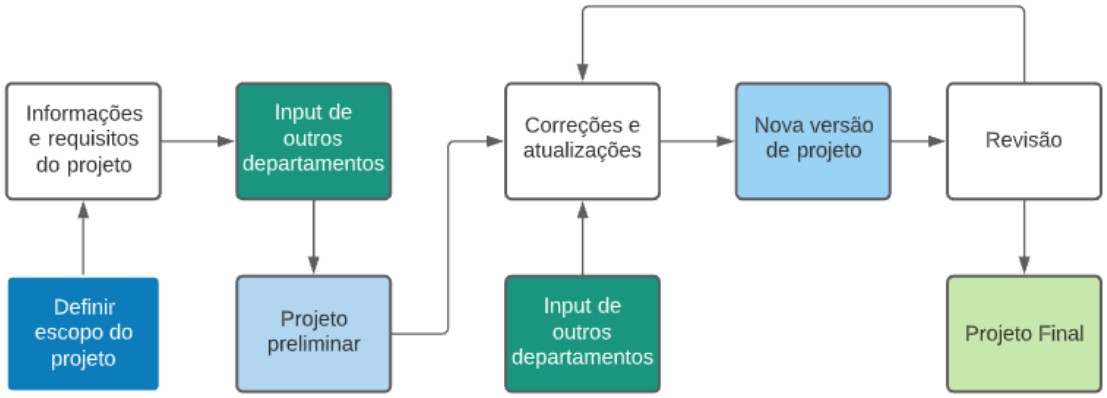
Fonte: https://www.spacestuff.org/how-rockets-work

Assim, o foguete é de extrema importância para a história da humanidade, e isso deve perdurar pelos próximos séculos na exploração espacial, não só por empresas governamentais, mas também por empresas privadas.

2 Objetivos

Durante o desenvolvimento de um foguete experimental, há alguns passos a serem seguidos. Dentro da UFABC Rocket Design, há um fluxo de projeto que guia os membros.

Figura 4: Fluxo de projeto



Fonte: própria

O primeiro passo para desenvolver um foguete experimental é saber o escopo do projeto. Para isso é necessário saber se será construído um foguete de 1 km ou de 3 km de apogeu, a carga necessária e outras informações. Após essa definição, esses dados são passados para os outros departamentos onde são implementados, nesse momento o departamento de propulsão trabalha em cima desse escopo de projeto para desenvolver o motor do foguete.

Para esse desenvolvimento são utilizadas as planilhas Casing, que permite calcular a pressão do invólucro do motor do foguete, para que ele não exploda. Solid Rocket Motor (SRM), que permite calcular a velocidade do foguete, apogeu, curva de queima e outras informações. Ezalt, que permite estimar a performance de voo. E O-ring, que permite construir o anel de vedação dos compartimentos do foguete.

Cada uma dessas planilhas gera dados que são usados nas outras, na respectiva ordem apresentada previamente. Após esses dados coletados é possível gerar uma simulação, no software Open Rocket, e ter resultados de desempenho do foguete. A partir disso modificações são feitas até atingir o projeto final.

Nesse contexto, o objetivo é a construção de uma ferramenta que facilite o desenvolvimento do foguete. A construção do pacote pyRocket, que abriga os módulos adaptando as planilhas Casing, SRM, Ezalt e O-ring, permite a troca de dados e a interação entre planilhas de forma eficiente e direta, tornando a construção dos foguetes cada vez mais fáceis.

3 Fundamentação Teórica

3.1 Fundamentação

Como explicado por Richard Nakka[6], em seu site na sessão de Teoria de motor-foguete sólido, e descrito no artigo ”Minifoguete a propelente sólido”[7], durante a operação de um foguete experimental diversos processos ocorrem, como reações químicas e efeitos físicos semelhantes ao funcionamento de um foguete real. Desde a queima do propelente até a exaustão

do gás pela tubeira.

Durante o estudo de foguetes experimentais, como os analisados por essa pesquisa, certos fundamentos são levadas em conta para alcançar um motor-foguete ideal de modo a simplificar a natureza de um foguete real.

Pode-se iniciar assumindo os seguintes fundamentos:

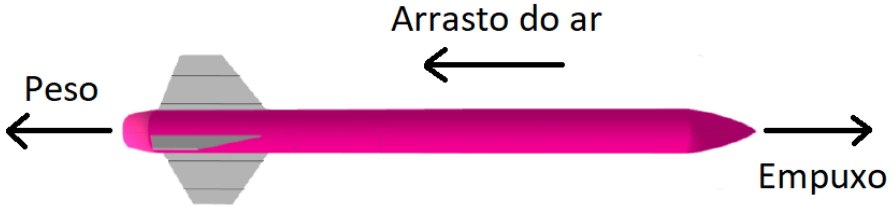
- Combustão completa do propelente
- Os produtos da combustão obedecem a lei dos gases perfeitos
- Não há fricção no escoamento da tubeira
- A combustão e o escoamento são adiabáticos (sem perda de calor)
- Condições de regime permanente existem durante a operação do motor
- Os produtos da exaustão ocorrem de maneira uniforme de contínua.
- O escoamento da tubeira é unidimensional e não rotacional
- Equilíbrio químico é estabelecido na câmara de combustão
- A queima do propelente progride perpendicularmente a superfície de queima

Tendo esses fundamentos, é possível refletir o comportamento de um foguete de uma forma muito próxima da realidade.

3.2 Forças no foguete

Considerando o movimento a superfície da Terra como referência, e o foguete se movendo na vertical, certas forças atuam nesse movimento. O empuxo, é responsável por ”empurrar”o foguete na direção vertical, a força de arrasto do ar, é responsável por causar o atrito do foguete com o ar, e a força peso, é responsável pelo peso do foguete.

Figura 5: Forças atuantes no foguete



Fonte: UFABC Rocket Design

A força de empuxo é resultante da combustão do grão-propelente, a pressão exercida no motor é conduzida para fora pela garganta da tubeira e gera uma força superior ao peso do foguete que o impulsiona verticalmente.

Quando um objeto se move a uma determinada velocidade sofre colisões com partículas da atmosfera que acaba por gerar um atríto com a superfície da coifa e, por consequência, acaba por retardar o movimento do foguete.

A força peso é dependente da massa total do foguete, definida pelos componentes presentes, variantes no tempo.

Se o tempo de queima for pequeno, pode se considerar a massa como constante e calcular a força peso através de:

$$P = m \cdot g \tag{1}$$

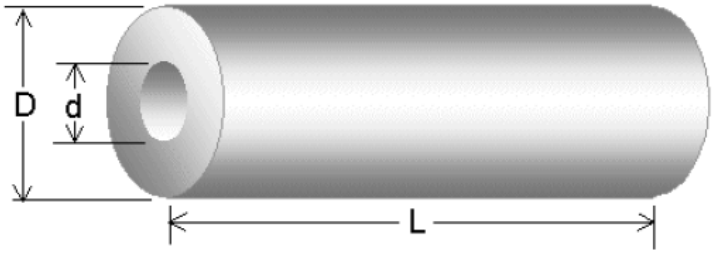
Sendo g a aceleração da gravidade, desconsiderando qualquer alteração pela altitude. Essa força tem sua direção em sentido ao referencial Terra.

3.3 Motor e propelente

O grão propelente é composta por dois constituintes, combustível e oxidante, que queimam de maneira homogênea. Os propelentes usados em foguetes experimentais são baseados em açúcares, mas ainda sim podem se tornarem bem complexos envolvendo polímeros aglomerantes, estabilizadores de fase e até solventes. Porém, independente da sua composição, um fator importante para a queima efetiva do propelente é sua forma geométrica.

O motor é constituído de três partes: a câmara de combustão, onde ocorre a queima do grão-propelente e forma a pressão interna, a garganta, local de passagem da pressão interna, sua área faz com que os gases internos saiam em uma velocidade maior, e o bocal, onde a pressão de saída diminui e a velocidade pode alcançar valores super sônicos.

Figura 6: Grão de cilindro oco

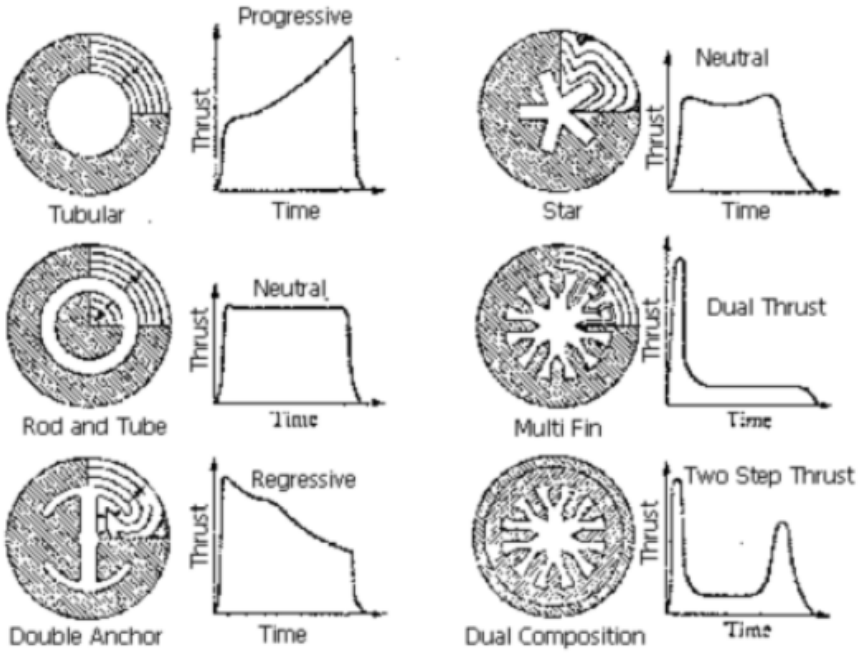


Fonte: Solid Rocket Motor Theory - Richard Nakka

Os grãos do motor possuem uma alma, o formato do interior, que percorre todo o grão transversalmente. Ela pode possuir vários formatos que afetam a performance do voo na forma empuxo-tempo. O empuxo gerado pelo motor do foguete é proporcional a área de queima em qualquer instante do tempo. A superfície de queima recua em qualquer ponto na perpendicular a superfície no ponto em específico, isso gera uma relação entre a superfície de queima e a distância da camada que foi queimada. Variando conforme os formatos da alma do grão.

A forma como a curva varia na vertical apresenta seu empuxo e na horizontal apresenta o tempo. Por exemplo, a curva do grão de alma estrela (star) apresenta curva neutra, o que é

Figura 7: Formas do núcleo e influência sobre a curva de empuxo



Fonte: Solid Rocket Motor Theory - Richard Nakka

desejável já que possibilita uma maior eficiência no impulso total resultante, devido a tubeira operar eficientemente a uma pressão constante na câmara.

A área de queima do grão é um fator determinante para a determinação do desempenho do foguete. Ao entrar em combustão, os produtos são produzidos a uma taxa de escoamento ditada por:

$$m_g = A_b \cdot p_p \cdot r \tag{2}$$

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). A área de queima está relacionada com a pressão máxima do motor, já que quanto maior a área, maior será a pressão do motor. Essa pressão definirá a estrutura.

A massa específica é usada nos cálculos de desempenho. Um propelente composto de dois constituintes (oxidante e combustível), sua massa específica pode ser dada como:

$$p_p = \frac{1}{\frac{f_o}{p_o} + \frac{f_f}{p_f}} \tag{3}$$

O símbolo P_p é a massa específica, f é a fração mássica, o sub-símbolo "o" se refere ao oxidante e "f" ao combustível. A massa específica também pode ser obtida pesando o grão, de modo a determinar sua massa (m_g), e medindo seu volume (V_g). Assim a massa específica (P_p) pode ser obtida através da razão entre a massa e o volume. E o volume sendo o cilindro oco, como na figura 6.

A variação entre a massa específica real e ideal varia por aproximadamente 5%, influenciado pela técnica de preparo e pela incerteza das medidas.

A fração de carregamento volumétrica (V_l) se refere a eficiência volumétrica do motor, dada pela razão entre o volume do grão (V_g) e o volume disponível na câmara (V_a), pode também ser dado como:

$$V_l = \frac{V_g}{V_a} = \frac{I_t}{I_{sp} \cdot p_p \cdot g \cdot V_a} \tag{4}$$

Sendo I_t o impulso total, usado para saber a altura que o foguete será propelido, dado por:

$$I_t = \int_0^{t_b} F \cdot dt \tag{5}$$

Onde I_{sp} o impulso específico, g a aceleração gravitacional (9,8 m/s²) e t_b o tempo de queima do motor.

O Impulso Específico é o critério de eficiência de um propelente, pode ser considerado como a força de empuxo produzida por uma unidade de massa de propelente sobre o tempo de queima em um segundo. Pode ser apresentado como:

$$I_{sp} = \frac{I_t}{w_p} \tag{6}$$

Onde w_p é o peso do propelente.

Sabendo I_t e I_{sp}, é possível obter uma aproximação da velocidade de exaustão dos gases da combustão, através da equação:

$$v_e = \frac{I_t}{m_p} = I_{sp} \cdot g \tag{7}$$

O impulso específico é dependente de:

- Fluxo de massa;
- Energia disponível na combustão;
- Eficiência da tubeira;
- Condições de pressão ambiente;
- Perda de calor para a estrutura;
- Perda de escoamento bifásico;
- Eficiência da combustão.

A fração de Camada (w_f) é a razão da espessura da camada de propelente para o raio externo do grão (r), dado por:

$$w_f = \frac{D - d}{D} = \frac{2 \cdot r \cdot t_b}{D} \tag{8}$$

Para maximizar a duração de queima é necessário aumentar a fração de camada, aumentar a espessura, porém fazendo isso há a diminuição do diâmetro da alma do grão.

A razão de área é dada pela área de seção transversal do canal de escoamento (Ap) e da garganta da tubeira (At).

$$\frac{A_p}{A_t} = \frac{\pi \cdot D^2 \cdot (1 - v_e)}{4 \cdot A_t} \tag{9}$$

A velocidade do gás ao longo do canal de escoamento é influenciada pela magnitude da razão de área porta-garganta. Quando a razão é 1, ocorre o escoamento bloqueado, onde a velocidade do escoamento na garganta é igual a velocidade de escoamento da porta. Frações menores, podem ser usada na extremidade superior do grão, ponto em que o fluxo de massa é mínimo.

É sugerido uma razão 2 ou 3, que apresenta um índice nos quais as tendências de queima erosiva são estabelecidas. Para projetar o motor, a razão entre comprimento e diâmetro do grão são fatores que influenciam. Quando maior a relação L/D, maior o efeito de queima erosiva, esse alto valor tende a gerar altos fluxos de massas diferentes ao longo do grão.

A velocidade do gás ao longo do canal de escoamento é influenciada pela magnitude da razão de área porta-garganta. Quando a razão é 1, ocorre o escoamento bloqueado, onde a velocidade do escoamento na garganta é igual a velocidade de escoamento da porta. Frações menores, pode ser usada na extremidade superior do grão, ponto em que o fluxo de massa é mínimo.

É sugerido uma razão 2 ou 3, que apresenta um índice nos quais as tendências de queima erosiva são estabelecidas. Para projetar o motor, a razão entre comprimento e diâmetro do grão são fatores que influenciam. Quanto maior a relação L/D (comprimento/diâmetro), maior o efeito de queima erosiva, esse alto valor tende a gerar altos fluxos de massas diferentes ao longo do grão.

Figura 8: Eixo de simetria da tubeira

Fonte: Própria

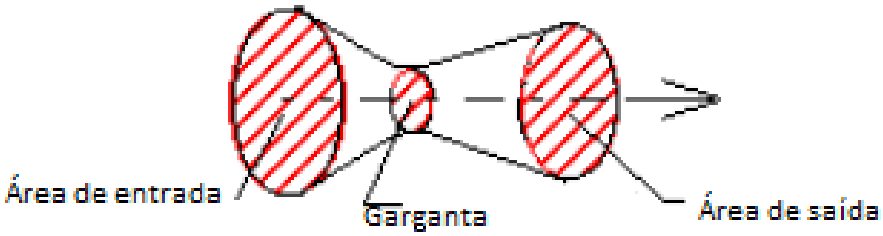
A função de uma tubeira é canalizar e acelerar os produtos produzidos pela combustão na queima do propelente, de modo a maximizar a velocidade de exaustão durante a saída. A tubeira de Laval, tubeira convergente-divergente, cumpre esse efeito de forma simples, variando a área da seção transversal (diâmetro) de forma precisa.

É considerado que o escoamento de um fluído é constante, não varia durante a queima e é unidimensional, ou seja, ao longo do eixo de simetria.

O escoamento é compressível, já que há gases se movendo em velocidades altas, geralmente supersônicas, logo, apresenta variação em sua massa específica.

De modo a simplificar, considerando esse fluído como um gás ideal, é possível usar a relação direta entre pressão, massa específica e temperatura, propriedades importantes na analise do escoamento dos gases na tubeira. Essas propriedades são afetadas pela variação na área da seção transversal, pela fricção entre o gás e a tubeira e pela perda de calor para a vizinhança.

Figura 9: Área da seção transversal

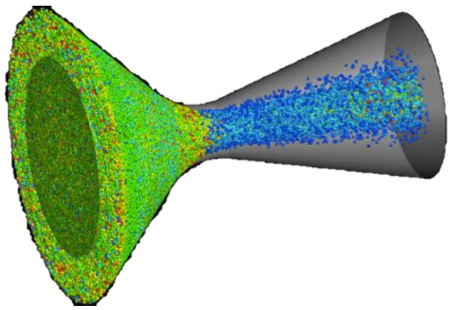


Fonte: Solid Rocket Motor Theory - Richard Nakka

Projetando a geometria da tubeira é possível acelerar os produtos da queima do propelente, sendo afetado somente pela variação da área da seção transversal quando o fluido se move.

Na porção divergente da tubeira que é atingido o escoamento supersônico.

Figura 10: Escoamento do bocal convergente-divergente



Fonte: www.mautone.eng.br

A velocidade sônica (a) e o número de Mach (M) são definidos como a razão da velocidade do escoamento para a velocidade sônica local, sendo dado por:

$$a = \sqrt{k \cdot R \cdot T} \tag{10}$$

$$M = \frac{v}{a} \tag{11}$$

A equação da continuidade, ou conservação de massa, é dada por:

$$pAv = cte = p^* \cdot A^* \cdot v^* \tag{12}$$

Sendo A a área da seção transversal da tubeira, v a velocidade do escoamento e p a pressão, a equação mostra que a massa escoada através da tubeira deve ser constante. Assim, A^* é a condição crítica, onde o número de Mach é 1, sendo a velocidade de escoamento igual a velocidade do som.

É possível expressar a razão das áreas A/A^* em termos do número de Mach. Portanto, a razão das áreas é a área da seção transversal em qualquer ponto da tubeira dividida pela área da seção transversal onde existe a condição crítica.

$$\frac{A}{A^*} = \frac{1}{M} \cdot \left(\frac{1 + \frac{k-1}{2} \cdot m^2}{1 + \frac{k-1}{2}} \right)^{\frac{k+1}{2 \cdot (k+1)}} \tag{13}$$

Ao plotar um gráfico de A/A^* por Mach, é visível que uma passagem de convergente-divergente com uma seção de área mínima é necessária para acelerar o escoamento da velocidade subsônica para supersônica. De modo simplista, é necessário ”reunir”o gás em uma área da tubeira, para depois ”espalhar”em outra área da tubeira, de modo a aumentar sua velocidade.

O ponto crítico acontece na garganta da tubeira, quando o valor de Mach atinge 1. Por esse motivo é necessária uma seção divergente para que o escoamento seja maior que a velocidade sônica.

É possível saber a velocidade na saída da tubeira através da seguinte equação:

$$v_e = \sqrt{\frac{2 \cdot k \cdot T_o}{k-1} \left(\frac{R'}{m} \right) \left[1 - \left(\frac{p_e}{p_o} \right)^{\frac{k-1}{k}} \right]} \tag{14}$$

Nessa equação temos:

- K como a razão efetiva entre calores específicos dos produtos de exaustão. Obtidos na análise de combustão;
- R’ como a constante universal dos gases (8314 N.m/Kmol.K);
- M é massa molecular efetiva dos produtos da exaustão, obtidos na análise da combustão;
- T_o é a temperatura de combustão do propelente, obtido na análise de combustão;
- P_e e P_o são a pressão na saída da tubeira e a pressão na câmara, respectivamente. Para foguetes experimentais, o valor de P_e pode ser dado como a pressão atmosférica.

Analisando o comportamento da tubeira, é possível ver que:

- A velocidade de exaustão máxima é alcançada quando P_e=0;
- Aumentar a pressão da câmara não aumenta significativamente a velocidade da exaustão;
- Uma temperatura de combustão mais alta e uma massa molecular mais baixa são igualmente benéficos.

3.4 Pressão da câmara

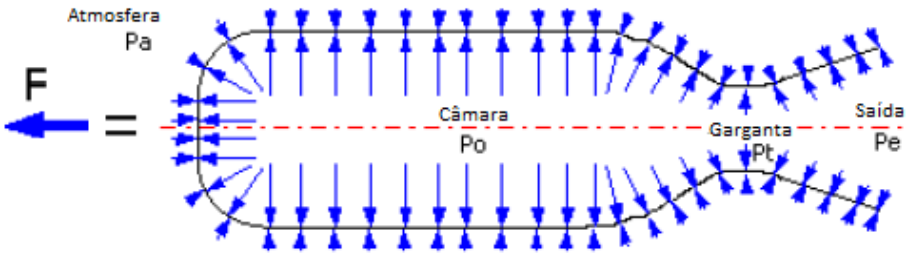
O motor-foguete opera sobre princípios básicos de conversão de energia térmica em energia cinética. A alta exaustão dos produtos da combustão saindo pela garganta do motor, tem sua energia cinética aumentada e, devido ao empuxo, o foguete é propulsionado.

O empuxo pode ser expresso por:

$$F = \int P \cdot dA \; = \; m \cdot v_e + (P_e - P_a) \cdot A_e \tag{15}$$

Sendo a integral das forças de pressão resultante atuando na câmara e tubeira, projetada em um plano normal ao eixo de simetria, como na figura 11.

Figura 11: Forças na câmara de combustão e tubeira



Fonte: Solid Rocket Motor Theory - Richard Nakka

Após a combustão, a pressão interna é maior dentro da câmara do que próximo a tubeira. Ainda na equação acima, é possível ver que o fluxo de massa (m) e a velocidade de exaustão (v_e) seguido pelo "empuxo de pressão", uma relação que é igual a zero em uma tubeira com expansão ótima, quando a pressão de saída (P_e) é igual a pressão atmosférica (P_a). E A_e a área da saída da tubeira.

Temos que, se a área da garganta é dobrada, o empuxo será dobrado (se a pressão for mantida). Se a pressão da câmara for dobrada, o empuxo também será dobrado (aproximadamente). Entretanto, se a pressão for aumentada, a estrutura deve ser reforçada.

A pressão dentro da câmara de um foguete é gerada pela combustão do grão-propelente, os gases produzidos aceleram na tentativa de escapar pela tubeira, se a garganta for pequena os gases podem não escapar rápido o suficiente e criar uma pressurização interna, podendo gerar uma explosão.

Para determinar a pressão de pressurização, deve-se primeiro notar a geração dos produtos da combustão. A taxa de consumo é equivalente a taxa de escoamento, portanto, a equação é identica a equação 2.

Durante a combustão são gerados gases, que contribuem para a pressão, e também são geradas partículas sólidas ou líquidas que são expelidas na fumaça durante a fase condensada, que contribuem para o empuxo.

A taxa na qual os produtos da combustão escoam através da tubeira é limitada pelo escoamento bloqueado, como citado anteriormente, onde o escoamento atinge a velocidade do

som (Mach 1) na garganta da tubeira. A condição é dada como bloqueada pois nesse local a velocidade nunca ultrapassa a velocidade do som. Desse modo, é possível determinar a taxa de escoamento dos produtos da combustão, através da equação:

$$m_m = P_o \cdot A^* \sqrt{\frac{k}{R \cdot T_o}} \left(\frac{2}{k+2} \right)^{\frac{k+1}{2\cdot(k-1)}} \tag{16}$$

A taxa de escoamento da tubeira é dada em função da pressão na câmara (P_o), que determina a massa específica do escoamento, da área da garganta ($A*$) e das propriedades do gás (determina a velocidade sônica).

A pressão na câmara pode ser definida em três estágio: o primeiro é a ignição, ocorre durante a operação do motor, a pressão na câmara sobe rapidamente, até o segundo estágio.

No segundo estágio a queima do propelente é quase totalmente estável e homogênea, onde a saída dos gases da combustão está em equilibrio com a produção dos gases.

O último estágio ocorre após o grão ter sido completamente consumido, inicia-se a despressurização (ponto onde a pressão interna se iguala a pressão externa), onde ainda há restos do grão-propelente. Porém, esse efeito é considerado sob a hipótese de o grão ter sido completamente consumido. Entretanto, experimentalmente, a queima desses resíduos formados durante a despressurização forma a escória, que pode se acumular na garganta da tubeira e diminuir a pressão.

3.5 Sobre Python

A linguagem Python foi criada por Guido van Rossum em 1991, é de desenvolvimento comunitário e possui uma das maiores comunidades ativas. Ela não possui padrões e espeficicações de formalidade, mas possui os PEPs (Python Enhancement Proposals - Propostas de aprimoramento de Python) que abordam funcionalidades, procedimentos e outros [8].

A linguagem foi projetada para ser versátil e eficiênte, priorizando a velocidade e a fácil leitura do código, exigindo poucas linhas para certas funções quando comparadas a outras linguagens.

Apesar de muitas pessoas fazerem referência a espécie de cobra, quando se pensa no nome da linguagem, ela na verdade faz referência ao grupo humorístico Monty Python[9].

Um *Easter Egg*[1] do Python é: quando importado o módulo ”this”é mostrado ”The Zen of Python”, que representa o guia de princípios da linguagem. [1]

```
>>import this

    The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
```

[1] Easter egg é um termo apropriado da lingua inglesa para se referir a elementos e segredos escondidos em filmes, séries, games e outros programas.

```
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

4 Metodologia

4.1 Materiais e métodos

Como apresentado anteriormente, a pesquisa se baseia nas planilhas de Richard Nakka [1], já que sua credibilidade no âmbito científico é bem grande, como é possível ver nos trabalhos de Galvão [10], Pedroni[11],[12] e Schlossmacher[13], onde Nakka é citado como fonte de pesquisa.

O processo de realização da construção do programa inicia-se com a engenharia reversa das planilhas, para isso é necessário entender os cálculos feitos nas planilhas e de onde as variáveis são tiradas. Algumas fórmulas são explicitadas, como as da planilha Casing, mas outras não tão explicitas, como as da planilha Ezalt. Posteriormente, entendendo os cálculos feitos, é possível estruturá-los na forma de código e construir o programa.

Para realizar a execução desses programas foi utilizado um notebook Lenovo ideapad S145, com processador Intel Core i5 e 8GB de RAM, rodando Windows 10 (versão 20H2), Python 3.8.3 e Spyder 4.1.4.

4.1.1 Instalação do Python

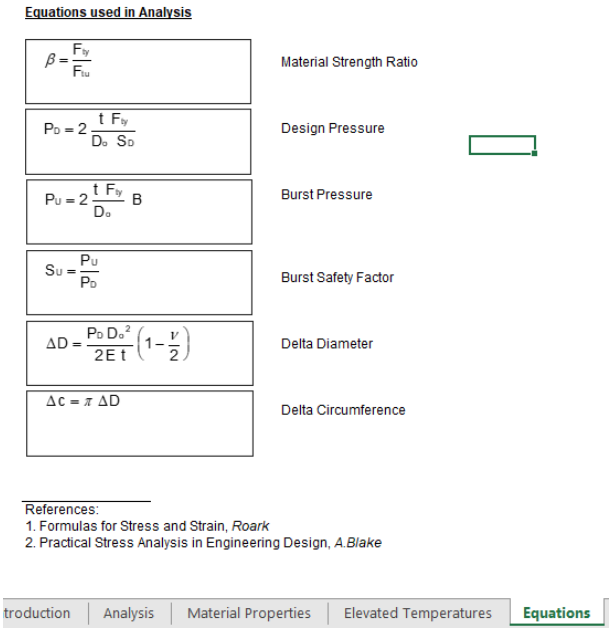
É simples a instalação do Python no Windows. É necessário entrar no site oficial (https://www.python.org/), clicar na aba download e fazer download da versão mais recente, ou procurar logo abaixo versões antigas.

Para a construção dos códigos foi utilizado a IDE Spyder, instalada a partir do navegador anaconda (https://www.anaconda.com/products/individual). Foi escolhida essa IDE por ser um

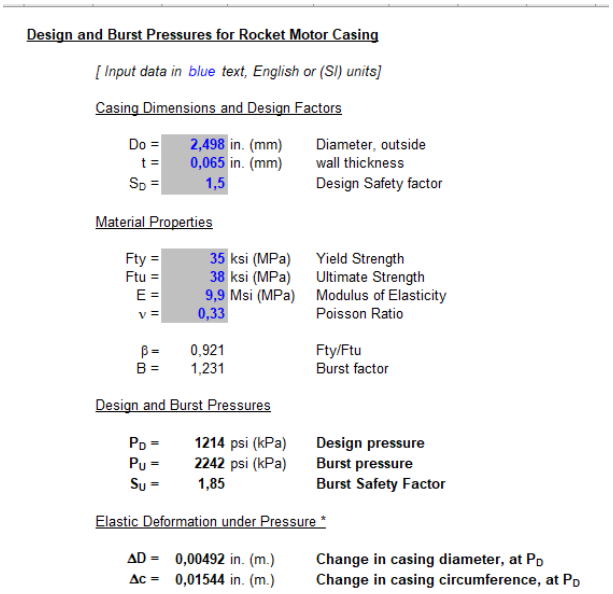
ambiente de desenvolvimento poderoso com edição avançada, testagem interativa e com recursos de debug e inspeção de código.

4.1.2 Implementação Planilha Casing

Figura 12: Planilha Casing



(a) Equações da planilha Casing



(b) Planilha Casing

Fonte: Planilha Casing - Richard Nakka

A primeira implementação foi feita na planilha Casing[1], utilizada para calcular o design de pressão e pressão de ruptura de um invólucro de um foguete experimental, assim como a expansão radial e mudança do diâmetro do invólucro devido a pressão exercida na câmada de combustão.

A partir das fórmulas apresentadas na figura 12, item a, é possível compreender os cálculos feito na planilha e no item b, entender como ela funciona.

E, entendo como funcionam, é possível construir o programa em Python (Veja o código completo no apêndice A).

```
1    class Casing():
2    def cas():
3    #Chama os pacotes usados no programa
4        import math
5        import matplotlib.pyplot  #Necessario somente caso for construir o
   grafico, entretanto o grafico nao e necessario.
6        import materials
7        from datetime import datetime
8
```

```
#Abertura do programa
        print("--Casing--")
#Solicitacao da insercao de dados
        print ("---Dimensoes do involucro e Fatores de design---")
        Do= float(input('Insira o diametro externo (mm): '))
        t= float(input('Insira a espessura da parede (mm): '))

#Opcao para escolher a entrada e, consequentemente, a saida
        print(" ")
        print("Para inserir, pressione:")
        print("0 para o valor do Fator de design de segurança ")
        print("1 para o valor do Fator de segurança de ruptura")
        op=str(input("") )

        if op=="0":
            Sd= float(input('Insira o Fator de design de segurança: '))
        elif op=="1":
            Su= float(input('Insira o Fator de segurança de ruptura: '))
```

Acima é possível ver o código iniciando com uma importação de biblioteca na qual chama os módulos math[14], que traz informações matemáticas, como o valor de π, que é usado posteriormente, o módulo matplotlib[15], que é usado para a construção de gráficos 2D, o módulo datetime[16], usado para chamar data e hora na criação do arquivo tipo texto (.txt) e o módulo materials, que contém as informações sobre os materiais disponíveis para cálculo.

Em seguida, há a inserção dos dados sobre as dimensões do invólucro como: diâmetro externo (Do) e espessura da parede (t), ambos em milímetros. E, logo após, é possível optar pela inserção do valor do Fator de design de segurança, e ter como output do programa o valor do Fator de segurança de ruptura, ou o oposto. Essa opção pode ser escolhida inserindo o valor ”0”para o primeiro e ”1”para o segundo.

O Fator de design de segurança se refere a quantidade de vezes que a câmara aguenta a pressão de operação máxima esperada (MEOP, da sigla em inglês), geralmente o invólucro é projetado para suportar uma pressão entre 1.5 vezes até, no máximo, 2.5 vezes. Esse fator ajuda a previnir possíveis acidentes fazendo com que a cápsula suporte valores maiores.

O Fator de segurança de ruptura é dado pela razão entre a pressão de ruptura pelo design de pressão, ele prevê a pressão que o invólucro suporta antes de romper.

O próximo input é a escolha do material. O código em python a seguir mostra a opção de ver, ou não, os materiais disponíveis e inserir qual o desejado.

```
    opcao=input("Deseja ver a lista de materiais? (S/N) ")
        if opcao=="S" or opcao=="s":
            for materials.mec_propk, materials.mec_propv in materials.
    mec_prop.items():
                print(f'{materials.mec_propk}')
        else:
            pass
```

```
    prop = input('Digite o tipo do material do invólucro: ')
    p=materials.mec_prop[prop]
```

Para isso é usado o pacote ”materials”, criado com os materiais e propriedades dispostos na planilha Casing. Como apresentado no código a seguir (Veja o código completo do apêndice B).

Nesse pacote temos os valores de força de rendimento (Fty), força máxima (Ftu), módulo de elasticidade (E) e coeficiente de Poisson (v) para cada combustível disponível na planilha Casing.

```
#Yield Strength (Fty) Forca de Rendimento
#Ultimate Strength (Ftu) Forca máxima
#modulo de elasticidade (E) Modulo de elasticidade
#coeficiente de poisson (v) Coeficiente de Poisson
mec_prop={
    'aco c 1010 laminacao quente':{
        'Fty':165,
        'Ftu':296,
        'E':200100,
        'v':0.32},
    'aco c 1010 laminacao fria':{
        'Fty':414,
        'Ftu':496,
        'E':200100,
        'v':0.32},
    'aco c 1015 laminacao quente':{
        'Fty':228,
        'Ftu':379,
        'E':200100,
        'v':0.32},
```

Com esses valores, é possível estabelecer a razão da força do material (b), a partir da relação $\frac{Fty}{Ftu}$, ambos dados retirados de valores experimentais por Nakka[17]. E, através da relação matemática a seguir, se obter o fator de ruptura (B).

$$B = A \cdot \beta^4 + B \cdot \beta^3 + C \cdot \beta^2 + D \cdot \beta^1 + E \tag{17}$$

Continuando o código RocketCasing.py, tem-se os cálculos e output feitos. Anteriormente, caso tenha sido escolhido a opção ”0”, para inserir o fator de design de segurança, são calculados: Design de pressão (Pd), Pressão de ruptura (Pu) e o fator de segurança de ruptura (Su). Caso tenha sido escolhido a opção ”1”, para inserir o valor do fator de segurança de ruptura, são calculados: Design de pressão (Pd), Pressão de ruptura (Pu) e fator de design de ruptura (Sd).

O design de pressão (Pd) é considerada ser a máxima pressão operacional esperada (MEOP), isso é, a pressão que normalmente não se espera que seja excedida durante a operação do motor. Esse critério de design é a ausência de deformação permanente do invólucro nas condições de operação, ou seja, não exceder a força de rendimento. É uma saída dada pelo programa.

O cálculo para o design de pressão (Pd) é dado por::

$$Pd = 2 \cdot \frac{t \cdot Fty \cdot 1000}{Do \cdot Sd} \tag{18}$$

A pressão de ruptura (Pu) é a pressão na câmara na qual o invólucro irá romper. O fator de segurança de ruptura é dado pela razão entre a pressão de ruptura e o design de pressão (um motor dever ter um limitador de pressão, como um plug de ruptura, pra prover um controle de liberação de pressão antes que a câmara atinja o valor crítico). É uma saída dada pelo programa. O cálculo para pressão de ruptura (Pu) dado por:

$$Pu = \frac{2 \cdot B \cdot t \cdot Fty \cdot 1000}{Do} \tag{19}$$

O Fator de segurança de ruptura (Su) é uma saída do programa dada pela razão entre Pu e Pd. E o fator de design de ruptura (Sd) é dado por:

$$Sd = \frac{2 \cdot (t \cdot \ Fty \cdot 1000)}{Do \cdot Pd} \tag{20}$$

A Variação no diâmetro (ΔD) para Pd e na circunferência (Δc) para Pd também são dados de saída do programa e são dados pelas equações 21 e 22, respectivamente, logo abaixo:

$$\Delta D = \frac{2 \cdot Pd \cdot (\frac{Do}{2})^2}{\frac{E}{\frac{10^6}{t \cdot (1 - \frac{\nu}{2})}}} \tag{21}$$

$$\Delta c = \pi \cdot \Delta D \tag{22}$$

O efeito prejudicial do aquecimento do invólucro sob condições operacionais pode ser levado em consideração pelo uso de curvas de redução da resistência do material a temperaturas elevadas.

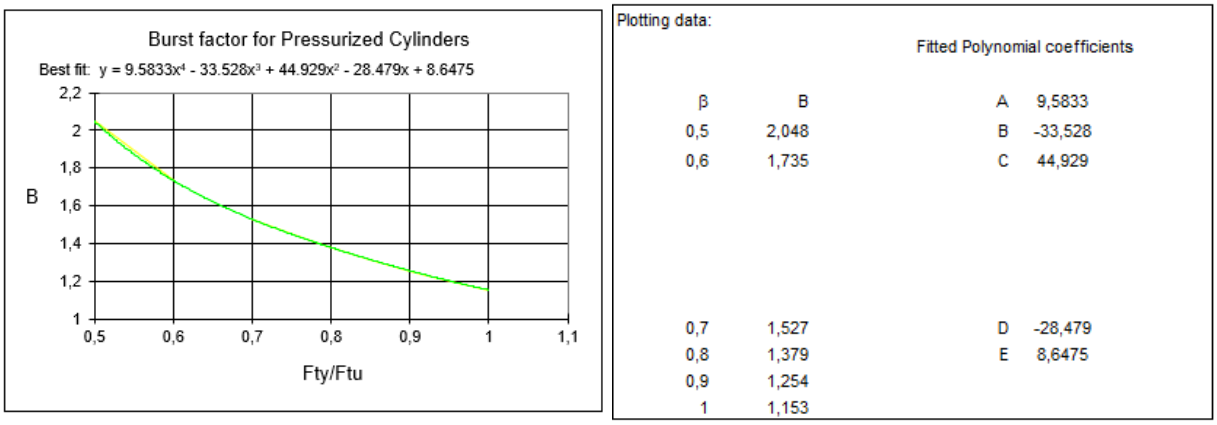
Na planilha, Richard mostra um gráfico titulado ”Fator de ruptura para cilindros pressurizados”, no qual relaciona fator de ruptura pela razão de força do material. Na construção desse gráfico, foram usados coeficientes polinomiais para ajuste de curva, apresentados também na planilha, como é possível ver na figura 13, item b. Entretanto, não é explicitado como esses coeficientes foram atingidos.

Portanto, a resolução feita foi construir o gráfico[18] da figura 14 de acordo com a planilha da figura 13, item a, a partir do código a seguir:

```
1    matplotlib.pyplot.title('Fator de ruptura para cilindros pressurizados'
     )
2    matplotlib.pyplot.xlabel('Fty/Ftu')
3    matplotlib.pyplot.ylabel('B')
4    matplotlib.pyplot.plot([0.5, 0.6, 0.7, 0.8, 0.9, 1], [2.048, 1.735,
     1.527, 1.379, 1.254, 1.153], color='g', marker='o')
```

```
5    matplotlib.pyplot.axis([0.5, 1, 1, 2.2]) # [xmin, xmax, ymin, ymax]
6
7    matplotlib.pyplot.show()
```

Figura 13: Gráfico e coeficientes planilha Casing

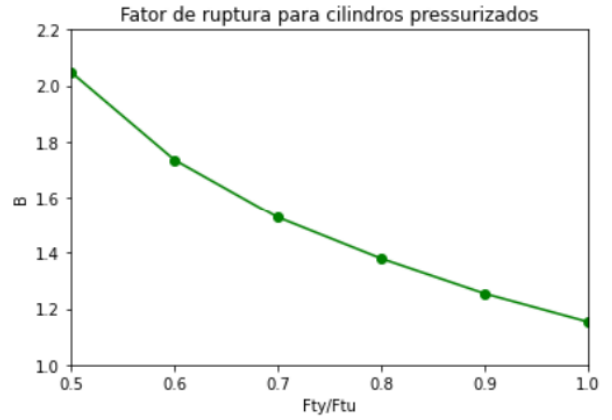


(a) Gráfico fator de ruptura

(b) Coeficiente polinomiais equipados

Fonte: https://www.nakka-rocketry.net/softw.html#SRM

Figura 14: Gráfico de ruptura para cilindros pressurizados



Fonte: própria

Após executar o programa, as informações obtidas são salvas em um arquivo .txt, através do código apresentado no início do capítulo:

A variável ”data”chama o pacote datetime, a variável ”datatxt”ajusta a características salvas no modo dia, mês, ano, hora e data. Logo a seguir, é criado o arquivo .txt, chamado ”casing results.txt”e a letra ”a”indica que os novos valores irão se escrever no mesmo arquivo.

Após isso, os valores são inseridos e organizados para serem legíveis no arquivo em texto.

Figura 15: Casing results.txt

```
20/05/2021 20:37
Material: aço inox 304
Características do material: {'Fty': 517, 'Ftu': 724, 'E': 186300, 'v': 0.27}
Razão da força do material: b= 0.7140883977900553
Fator de ruptura: B= 1.5045933904503013
Design de pressão: Pd= 172333.33333333334
Pressão de ruptura: Pu= 388937.3914314029
Fator de segurança de ruptura: Su= 2.256890085675452
Variação no diâmetro do invólucro: dD= 3.200608337806406e-06
Variação na circunferência do invólucro: dc= 1.0055007641070844e-05
--------------------------------
```

Fonte: Própria

4.1.3 Implementação Planilha Ezalt

O processo de implementação da planilha Ezalt [19] é semelhante ao da planilha Casing. O fator principal de dificuldade dessa planilha é que, diferente da Casing, as equações não estão explicitas.

Para isso, foi necessário compreender como cada equação funcionada através das células do Excel, ferramenta do pacote office.

Entendendo as fórmulas usadas, o processo de execução é semelhante ao da construção do programa RocketCasing.

A planilha Ezalt é usada para calcular uma estimativa para a performance de voo para foguetes experimentais. Os valores de input são:

- Título;
- Impulso médio do motor - F (medido em Newtons);
- Impulso total do motor - I_t (medido em Newton-segundo);
- Massa do propelente do motor - m_p (medido em quilos);
- Massa morta do foguete - m_r (medido em quilos);
- Diâmetro máximo do foguete - D (medido em centímetros);
- Coeficiente do arrasto de foguete - C_d.

A partir dessas entradas é possível calcular: O tempo de impulso do foguete, usado para saber o tempo levado para o foguete alcançar seu apogeu, dado por:

$$t = \frac{I_t}{F} \tag{23}$$

Sabendo o tempo de impulso, é possível definir a classificação do motor, baseado na tabela da figura 17.

Figura 16: Planilha Ezalt

FLIGHT PERFORMANCE ESTIMATOR FOR HOBBY ROCKETS

(valid for *subsonic* rockets only)

EzAlt.xls (MS Excel 97)
Version: 1,2
Date: Feb. 2007

*Instructions: Enter data in **blue** text (boxed cells).*

Title		example rocket		
Motor average thrust	F =	1368,1	N.	
Motor total impulse	It =	2531,9	N-sec.	
Motor propellant mass	mp =	2,123	kg.	**Input data**
Rocket dead mass	mr =	4,200	kg.	
Rocket diameter (max)	D =	10	cm.	
Rocket drag coefficient	Cd =	0,5		

Motor thrust time	t =	1,851	sec.
Motor classification		K	
Rocket avg. flight mass	mra =	5,262	kg.
Acceleration (average)	**a** =	**250**	metre/sec^2
	or **a** =	**25,5**	g's

Peak altitude (zero drag)	z2 =	11360	metres	
Time to peak altitude (zero drag)	t2 =	49,1	sec.	**Ideal** (no drag
Max velocity (zero drag)	v1 =	463	metre/sec.	resistance)
Burnout altitude (zero drag)	z1 =	428	metres	

Drag Influence number	N =	2553	Out of valid range!	
Peak altitude reduction factor	f1 =	0,179		
Time to peak reduction factor	f2 =	0,303		Drag reduction factors
Max velocity reduction factor	f3 =	0,804		
Burnout altitude reduction factor	f4 =	0,888		

Peak altitude	**Z peak** =	**2034**	metres	
Time to peak altitude	**t peak** =	**14,9**	sec.	**Predicted** (with
Max velocity	**V max** =	**373**	metre/sec.	drag)
	or **V max** =	**1341**	km/hr	
Burnout altitude	**Z bo** =	**380**	metres	

Warning: Rocket is supersonic, results may be invalid

Fonte: Planilha Ezalt - Richard Nakka

Figura 17: Sistema de classificação de motores

Rocket Motor Classification System			
Class	Total Impulse (Newton-seconds)		
A	1,3		2,5
B	2,5	-	5
C	5	-	10
D	10	-	20
E	20	-	40
F	40	-	80
G	80	-	160
H	160	-	320
I	320	-	640
J	640	-	1280
K	1280	-	2560

Fonte: Planilha Ezalt - Richard Nakka

A massa média de voo do foguete é dada por:

$$m_{ra} = \frac{m_r + m_p}{2} \tag{24}$$

Onde m_r e m_p, como apresentados anteriormente, são a massa morta do foguete e a massa morta do propelente, respectivamente.

A aceleração, é dada em m/s², e calculada por:

$$a = \frac{F}{m_{ra} - 9,808} \tag{25}$$

Após os outputs desses dados, são feitos cálculos para uma situação ideal, onde não é considerado a resistência de arrasto. Nessa sessão é calculado:

Altitude de pico, medida em metros, dada por:

$$z_2 = \frac{F \cdot z_1}{\frac{m_{ra}}{9,808}} \tag{26}$$

Tempo para atingir altitude de pico, medido em segundos, dado por:

$$t_2 = \sqrt{\frac{2 \cdot (z_2 - z_1)}{9,808}} \tag{27}$$

Velocidade máxima atingida pelo foguete, medido em m/s, dado por:

$$v_1 = \sqrt{\frac{2 \cdot z_1}{m_{ra \cdot (F - m_{ra} \cdot 9,808)}}} \tag{28}$$

Se o valor de v_1 for maior que 331, a planilha apresenta uma mensagem dizendo ”Atenção: Foguete supersônico, resultados podem ser inválidos”.

E, por último, a altitude de burnout, ponto final de queima do propelente, dado por:

$$z_1 = \frac{1}{2} \cdot \left(\frac{F}{m_{ra} - 9,808} \right) \cdot t^2 \tag{29}$$

Em seguida, são feitos cálculos considerando fatores de redução de arrasto, devido a resistência do ar. Sendo os cálculos:

O Número de influência de arrasto é usado para relacionar os valores ideais com os reais, baseando no fator de redução de arrasto, considerando o arrasto aerodinâmico aplicado aos valores ideais. É calculado por:

$$N = \frac{C_d \cdot D^2 \cdot v_1}{m_r \cdot 1000} \tag{30}$$

Se esse valor for maior que 900, a planilha apresenta a mensagem ”Fora do intervalo válido”

O fator de redução de altitude de pico, onde a altitude é reduzida pelo arrasto, é dado por: Se a equação 31 for maior que 1, o fator de redução é 1, senão é o valor dado pela equação.

$$f_1 = \frac{1}{1,049909 + 0,001719 \cdot N^{1,004225}} \tag{31}$$

O fator de redução de tempo de altitude, onde o tempo para atingir o pico é reduzido devido ao arrasto, é dado por: Se a equação 32 for maior que 1, o fator de redução é 1, senão é o valor dado pela equação.

$$f_2 = \frac{1}{1,048224 + 0,001093 \cdot N^{0,97255}} \tag{32}$$

O fator de redução de velocidade máxima, onde a velocidade máxima é reduzida pelo arrasto, é dado por: Se a equação 33 for maior que 1, o fator de redução é 1, senão é o valor dado pela equação.

$$f_3 = 0,99769 + 0,000075691 \cdot N \tag{33}$$

O fator de redução de altitude de burnout, a altitude de burnout é reduzida devido ao arrasto, é dado por: Se a equação 34 for maior que 1, o fator de redução é 1, senão é o valor dado pela equação.

$$f_4 = 0,99973 + 0,000043807 \cdot N \tag{34}$$

Após definir os valores ideais e os fatores de redução, é possível calcular as informações de output predito com arrasto. Esses cálculos são dados por:

A altitude de pico é dada por:

$$Z_{pico} = f_1 \cdot z_2 \tag{35}$$

O tempo para a altitude de pico é dado por:

$$t_{pico} = f_2 \cdot t_2 \tag{36}$$

A velocidade máxima, medida em m/s, é dada por:

$$v_{max} = f_3 \cdot v_1 \tag{37}$$

A altitude de burnout é dada por:

$$z_{bo} = f_4 \cdot z_1 \tag{38}$$

Após o estudo da planilha, e compreendendo como ela funciona, é possível construi o programa em Python para executar os mesmos cálculos (Veja o código completo no apêndice C).

```
class Ezalt():
   def ezalt():
        import math
        import matplotlib.pyplot
        from datetime import datetime

#Abertura do programa
        print(" ")
        print("--Ezalt--")

#input do programa
```

```
        title=str(input(("Insira o nome do foguete: ")))
        F=float(input(("Insira o Impulso do motor (N): ")))
        It=float(input(("Insira o Impulso Total do motor (N-sec): ")))
        Mp=float(input(("Insira a Massa do propelente do motor (kg): ")))
        Mr=float(input(("Insira a Massa morta do foguete (kg): "))) #massa
    do foguete sem o motor
        D=float(input(("Insira o diametro do foguete (cm): ")))
        Cd=float(input(("Insira o coeficiente de arrasto (tipicamente entre
     0,3 a 0,6): ")))
```

```
        Mra=Mr+Mp/2
        print("Massa media de voo do foguete: Mra= ",'{:.3f}'.format(Mra),"
     Kg")
        a=F/Mra-9.808
        print("Aceleracao: a= ",'{:.3f}'.format(a)," m/s^2")
        print(" ")
        print("Ideal -  Sem resistencia de arrasto")
        print(" ")
        z1=1/2*(F/Mra-9.808)*t**2
        print("Altitude de burnout: ",'{:.3f}'.format(z1)," m")
        z2=F*z1/Mra/9.808
        print("Altitude de pico: ",'{:.3f}'.format(z2)," m")
        t2=t+math.sqrt(2*(z2-z1)/9.808)
        print("Tempo para atingir altitude de pico: ",'{:.3f}'.format(t2),"
     s")
        v1=math.sqrt((2*z1)/Mra*(F-Mra*9.808))
        print("Velocidade maxima: ",'{:.3f}'.format(v1)," m/s")
```

O trecho do código acima, demonstra a construção do programa em Python, onde se inicia importando os módulos math e datetime (apresentados quando explicado sobre o programa RocketCasing). A seguir são colocados os inputs e os cálculos de output, apresentados anteriormente na descrição da planilha.

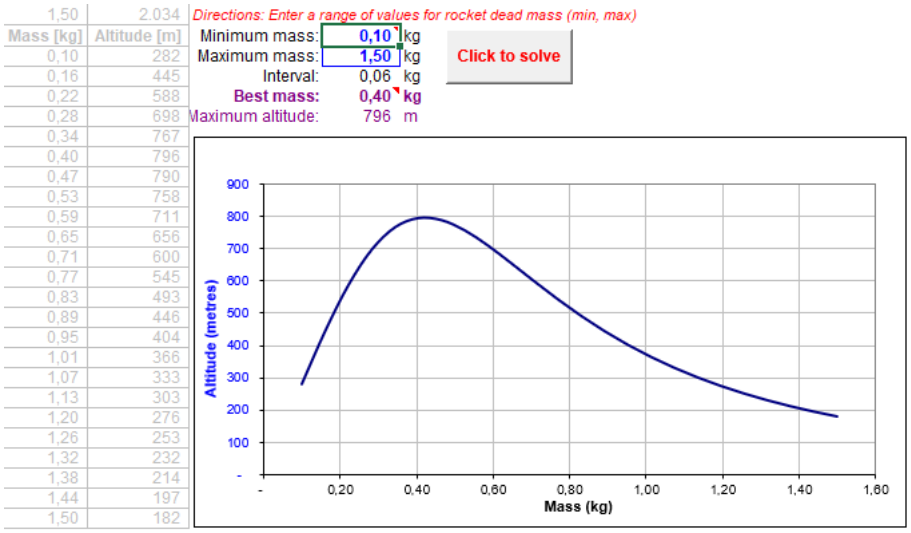
Na planilha Ezalt, ainda há outra aba chamada Best Mass, ou Melhor Massa, ela determina automaticamente a massa do foguete que vai resultar na maior altitude alcançada pelo foguete. Ela prediz valores para para foguetes experimentais típicos e é válida somente para foguetes que operam em velocidades subsônicas.

A planilha solicita dois inputs, referentes a massa morta do foguete, o primeiro é a massa mínima (que deve ser maior que 0) e o segundo é a massa máxima. Com esses dois valores é calculador o intervalo, através da fórmula:

$$intervalo = \frac{m_{max} - m_{min}}{23} \tag{39}$$

Nakka, na planilha melhor massa, cria uma tabela com os valores de massa, da menor até a maior inserida variando conforme a variável intervalo. E com os valores de altitude fixos, como é possível ver na figura 18. Nessa tabela a maior altitude é 796 metros, conforme varia os inputs

Figura 18: Planilha Ezalt



Fonte: Planilha Ezalt - Richard Nakka

de massa, o output de melhor massa também varia. Pela progressão da tabela, a maior altitude é o 5º valor, que corresponde ao valor de melhor massa, assim temos:

```
    Mmassa=(i*5)+Mmin
    print("Melhor massa: ",'{:.2f}'.format(Mmassa), ' Kg')
```

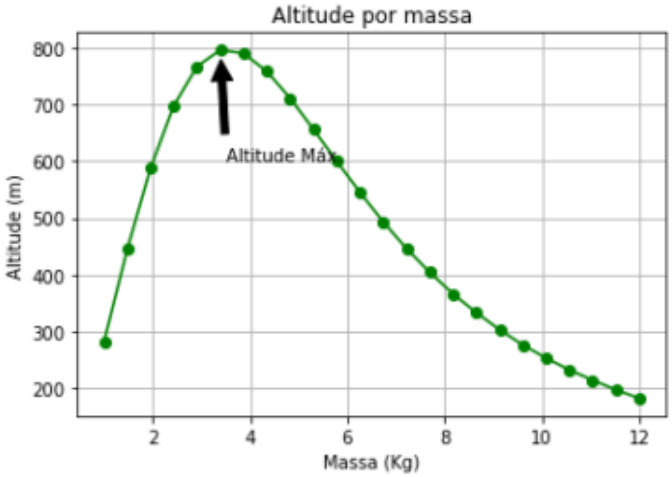
Logo após definir a melhor massa, é construido um gráfico relacionando altitude, em metros, pela massa, em quilogramas.

```
    #construção do gráfico
    m=Mmin
    matplotlib.pyplot.title('Altitude por massa')
    matplotlib.pyplot.xlabel('Massa (Kg)')
    matplotlib.pyplot.ylabel('Altitude (m)')
    matplotlib.pyplot.grid(True)
    matplotlib.pyplot.plot([m, m+i,m+2*i,m+3*i,m+4*i,m+5*i,m+6*i,m+7*i,m+8*
    i,m+9*i,m+10*i,m+11*i,m+12*i,m+13*i,m+14*i,m+15*i,m+16*i,m+17*i,m+18*i,m
    +19*i,m+20*i,m+21*i,m+22*i
    ,m+23*i], [282, 445, 588, 698, 767, 796, 790, 758, 711, 656, 600, 545,
    493, 446, 404, 366, 333, 303, 276, 253, 232, 214, 197, 182], color='g',
    marker='o')
    matplotlib.pyplot.annotate('Altitude Máx.', xy=(m+5*i, 796), xytext
    =(3.5, 600), arrowprops=dict(facecolor='black', shrink=0.1))
    matplotlib.pyplot.show()
```

O código acima mostra a construção do gráfico[18], ”m”é a variável que contém o valor da massa mínima, ela é explicita no eixo das abscissas progredindo conforme o valor do intervalo, e os valores do eixo das ordenadas são fixos pela tabela feito pelo Nakka. Assim é possível construir o gráfico, como é visível a seguir:

Posteriormente, é construido um arquivo ”.txt”, assim como foi feito no código RocketCasing, para salvar os outputs do código Ezalt.

Figura 19: Gráfico Altitude por massa - Planilha Melhor Massa



Fonte: Planilha Ezalt - Richard Nakka

Figura 20: Arquivo .txt do código Ezalt

```
17/08/2021 23:28

Título: fogueto
Impulso do motor: F= 1380.0 N
Impulso total: It= 2560.0 N-sec
Massa do propelente: Mp= 980.0 Kg
Massa morta: Mr= 400.0 Kg
Diâmetro do foguete: D= 10.0 cm
Coeficiente de Arrasto: Cd= 0.5
Tempo de impulso do motor: t= 1.855072463768116 s
Classificação do motor: K
Massa média de voo: Mra= 890.0 Kg
Aceleração: a= -8.25743820224719 m/s²
-----------------------------------

Ideal - Sem resistência de arrasto

Altitude de Burnout: z1= -14.208135633860321 m
Altitude de pico: z2= -2.246186004399899 m
Tempo para altitude de pico: t2= 3.4168736699524205 s
Velocidade máxima: v1= 15.318146230255659 m/s
-----------------------------------
```

Fonte: Própria

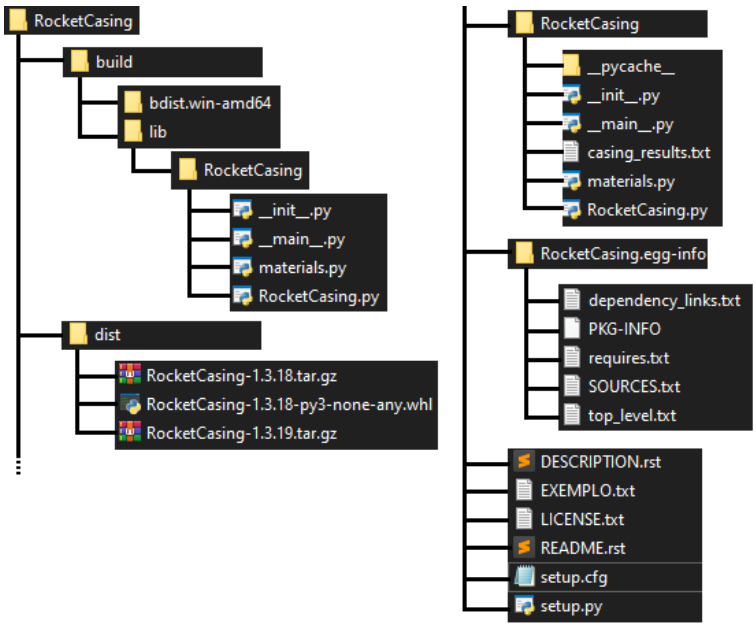
4.2 Construção dos módulos

4.2.1 Preparação da pasta

Concluida a construção dos códigos, inicia-se o upload para o PyPI[20], Python Package Index, repositório de softwares Python.

Para montar o módulo é necessário preparar o código e o local em que ele se encontra[21][22][23][24][25]. Na figura 21, é possível ver a pasta RocketCasing, ela contém os arquivos e pastas usados para ”uploadear”o código para os servidores PyPI. Dentro dessa pasta é criada outra pasta chamada Casing, ela possui o código criado (RocketCasing.py), o código materials.py, __init__.py e __main__.py.

Figura 21: Árvore Arquivos RocketCasing



Fonte: Própria

O código RocketCasing.py e materials.py foram explicados anteriormente na subseção 4.1.1, o arquivo __init__.py inicia o módulo e chama módulos necessários para rodar o programa.

```
1    from RocketCasing import materials
```

Esse trecho corresponde ao arquivo de inicialização, onde, da pasta RocketCasing, importa o código materials.

O arquivo __main__.py permite que esse arquivo seja chamado, como um import, em outros programas sem que haja a necessidade de escrever todo o código novamente dentro do programa.

```
1    from RocketCasing import Casing
2    RocketCasing.Casing()
```

Aqui, da pasta RocketCasing, é chamado o programa RocketCasing e ele executa a classe ”Casing()”.

Outros arquivos dentro da pasta principal são:

- DESCRIPTION.rst
- Exemplo.txt
- LICENSE.txt
- README.rst
- Setup.cfg
- Setup.py

O arquivo DESCRIPTION.rst traz uma breve descrição do que se trata o programa, a extensão .rst significa reStructuredText e é uma forma do programa ler o arquivo de texto. O arquivo Exemplo.txt é um bloco de texto que apresenta uma explicação de como o código funciona e mostra um exemplo com valores.

O arquivo LICENSE.txt apresenta a licença usada nesse código, copiada do PyPA (Python Package Authority).

O arquivo README.rst traz uma descrição mais detalhada do programa com os tópicos: Sobre, funcionalidade, como executar, pré-requisitos, instalado, como contribuir e licença.

Por fim, os arquivos setup. Eles trabalham com metadata, uma biblioteca que provê acesso para pacotes instalados. O Setup.py é um arquivo com as instruções, configurações e detalhes sobre os arquivos que formam o pacote, isso inclui arquivos necessários para instalação e funcionamento. O Setup.cfg é voltado para o tipo de distribuição que você deseja criar, bdist/sdist e outras classificações, para a instalação do pacote, ele age como um meio-termo entre o script de instalação e a linha de comando para o script de instalação.[26]

Figura 22: Exemplo código setup.py

```
# Always prefer setuptools over distutils
from setuptools import setup, find_packages
import pathlib

here = pathlib.Path(__file__).parent.resolve()

# Get the long description from the README file
long_description = (here / 'README.rst').read_text(encoding='utf-8')

# Arguments marked as "Required" below must be included for upload to PyPI.
# Fields marked as "Optional" may be commented out.

setup(
    # This is the name of your project. The first time you publish this
    # package, this name will be registered for you. It will determine how
    # users can install this project, e.g.:
    #
    # $ pip install sampleproject
    #
    # And where it will live on PyPI: https://pypi.org/project/sampleproject/
    #
    # There are some restrictions on what makes a valid project name
    # specification here:
    # https://packaging.python.org/specifications/core-metadata/#name
    name='RocketCasing',  # Required

    # Versions should comply with PEP 440:
    # https://www.python.org/dev/peps/pep-0440/
    #
    # For a discussion on single-sourcing the version across setup.py and the
    # project code, see
    # https://packaging.python.org/en/latest/single_source_version.html
    version='1.3.19',  # Required
```

Fonte: Própria

4.2.2 Upload para PyPI

Após criar a pasta e os arquivos apresentados, é necessário ”uploadear”essa pasta para o servidor PyPI, como comentado anteriormente. Para fazer esse upload são necessários quatro códigos rodados dentro do prompt de comando do windows.

```
1    python setup.py sdist
```

```
    python setup.py bdist
    py -m pip install --user --upgrade twine
    py -m twine upload --repository pypi dist/*
```

Os códigos sdist e bdist preparam a pasta para que seja possível fazer upload dela para o servidor. Enquanto o comando sdist cria as pastas egg e dist, o comando bdist cria a pasta build, onde salva arquivos wheel (python puro)[27].

O terceiro código instala o módulo twine[28], fazendo upgrade para a última versão caso já tenha instalado, que é utilizado para publicar pacotes Python para o PyPI. O maior motivo para usar o Twine é que ele provê uma melhor segurança e testabilidade, autenticando através do HTTPS usando sua conexão, independente da versão do Python. Enquanto o python setup.py upload apena funcionará corretamente e seguramente caso seu programa, versão do python e sistema operacional estiverem configurados propriamente[28].

O quarto, e último, código faz o upload do repositório para o servidor PyPI através da pasta dist usando o twine. Quando rodado esse comando será necessário inserir seu login do site PyPI (https://pypi.org/).

```
    C:\RocketCasing>py -m twine upload --repository pypi dist/*
Uploading distributions to https://upload.pypi.org/legacy/
Enter your username: LucasValentim
Enter your password:
Uploading RocketCasing-0.1.2-py3-none-any.whl
100%$|#####################################################|$ 14.9k/14.9k
   [00:01<00:00, 8.02kB/s]
Uploading RocketCasing-0.1.2.tar.gz
100%|#####################################################| 15.9k/15.9k
   [00:01<00:00, 13.2kB/s]

View at:
https://pypi.org/project/RocketCasing/0.1.2/
```

Esse trecho mostra a tela quando rodado o último comando, para subir o pacote para o servidor PyPI, da primeira versão do programa RocketCasing.

Logo que finalizado, é dado o link para visitar o site do PyPI que contém os pacote criados.

É possível visitar o pacote RocketCasing em: https://pypi.org/project/Rocket Casing/. E o código do pacote Ezalt em: https://pypi.org/project/Ezalt/.

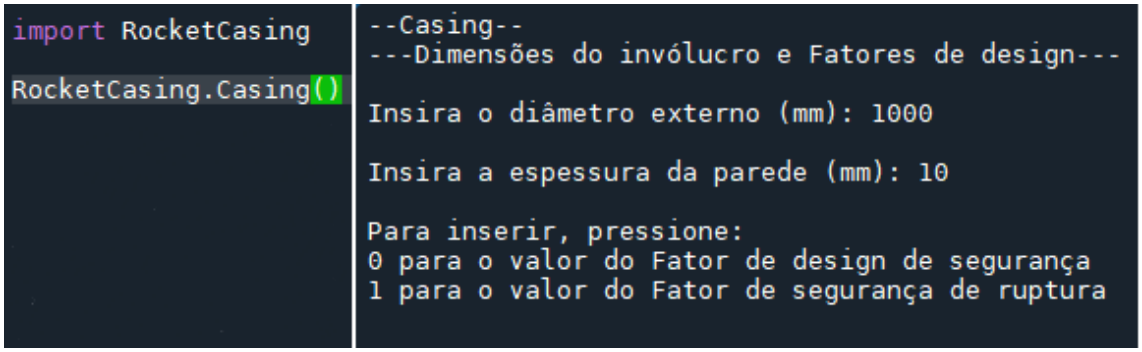
Estando o programa devidamente em um pacote e esse pacote nos servidores PyPI, é possível baixá-lo e usa-lo. No trecho abaixo é possível ver o download[29] do pacote RocketCasing (Veja o código completo no apêndice D).

```
    C:\Users\Extra\Desktop\RocketCasing\Casing>python -m pip install
   RocketCasing
Collecting RocketCasing
  Using cached RocketCasing-1.4.19.tar.gz (8.4 kB)
```

Após baixar o pacote é possível utilizá-lo[30], como apresentado a seguir.

```
1    import RocketCasing
2    RocketCasing.Casing()
```

Figura 23: Exemplo import RocketCasing



Fonte: Própria

Os códigos podem ser encontrados também em: https://github.com /LucasValentimB/py_Rocket

5 Resultados

A proposta desenvolvida nesse trabalho possibilitou compreender como as planilhas do Nakka funcionam e como elas implementam o desenvolvimento de foguetes experimentais, permitindo que conhecimentos práticos e teóricos reunidos por Nakka, desde os anos 70, sejam usados para desenvolver novas pesquisas até os dias de hoje.

A construção dos pacotes dos programas RocketCasing e Ezalt permite uma maior performance na construção de foguetes. A automatização de planilhas, como a escolha do material na planilha Casing, facilita o uso da tais informações.

Quanto mais planilhas forem transformadas, informações entre elas podem ser trocadas e a necessidade de varias planilhas passa a ser deixada de lado, conforme a integração entre elas facilita a troca de dados.

O resultado da pesquisa pode ser visto no servidor PyPI, como apresentados no item 4.2.2.

6 Perspectivas futuras

Como comentado previamente, Nakka possui diversas planilhas e os dados dessas planilhas se completam.

Com a construção dos programas é possível uní-los em um único pacote, o pyRocket, que reune todas as variáveis de forma mais eficiente, assim não é necessário alternar entre planilhas.

Adaptações podem ser feitas nas planilhas para facilitar ainda mais o uso delas. Uma delas, que seria feita no programa RocketCasing, é numerar os materiais da lista de materiais para facilitar sua seleção, assim como permitir que, caso o material desejado não esteja cadastrado, seja possível inserir as informações manualmente.

Essas adaptações, quando implementadas, são ”uploadeadas”para o servidor PyPI, possibilitando a atualização do módulo e usar suas novas funções.

Portanto, é visível, no futuro desse projeto, implementações com o objetivo de melhorar e adaptar o projeto, assim como, entregar uma ferramenta eficiente e útil.

Referências

[1] Richard Nakka. What is this web site all about?. richard nakka's experimental rocketry web site, jul 2021. Acesso em: 04 de Jan. de 2021.

[2] Britannica Escola. Foguete, dec 2020. Acesso em 22 de dez. de 2020.

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

[4] 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]. Acessado em: 14 de Jan. de 2021.

[5] Daniel Neves SILVA. Corrida espacial, fev 2021. Acesso em: 09 de fev. De 2021.

[6] Richard NAKKA. Solid rocket motor theory. *Richard Nakka's Experimental Rocketry Web Site*, jul 2001. Acesso em 13 de Out. de 2020.

[7] 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*, set 2020.

[8] Python Software Foundation. Pep 1 – pep purpose and guidelines, ago 2021. Acesso em: 27 de Ago. de 2021.

[9] Naomi Hamilton. The a-z of programmin languages: Python, ago 2008. Acesso em: 27 de Ago. de 2021.

[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: 30 de Nov. de 2020.

[11] Yan da Silva PEDRONI. Otimização da propulsão de foguetes por meio do desenvolvimento de modelagem numérica acoplada fsi. *Trabalho de conclusão de curso (bacharelado em engenharia aeroespacial) – Universidade Federal de Santa Catarina, Joinville*, jul 2016. [Orientador: Profª. Talita Sauter Possamai]. Acesso em: 14 de Dez. de 2020.

[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*, 42, nov 2020. Acesso em: 14 de Jan. de 2021.

[13] Lucas SCHLOSSMACHER. Desenvolvimento de motores-foguete para espaçomodelos. *Trabalho de conclusão de curso (bacharelado em engenharia mecânica) – Universidade Federal do Paraná, Curitiba*, jul 2015. [Orientador: Prof. Dr. Carlos Henrique Marchi]. Acesso em: 25 de Nov. de 2020.

[14] Python Software Foundation. math — mathematical functions, sep 2021. Acesso em: 25 de Set. de 2020.

[15] John Hunter, Darren Dale, Eric Firing, Michael Droettboom, and the Matplotlib development team. Pyplot tutorial - an introduction to the pyplot interface., sep 2021. Acesso em: 30 de Set. de 2020.

[16] Python Software Foundation. datetime — basic date and time types, sep 2021. Acesso em: 3 de Fev. de 2021.

[17] Richard Nakka. Planilha casing, mai 2021. Acesso em: 15 de Out. de 2020.

[18] Rodrigo Santana. Plotando gráficos de um jeito fácil com python, dez 2020. Acesso em: 14 de Nov. de 2020.

[19] Richard Nakka. Ezalt, mai 2021. Acesso em: 13 de Jun. de 2021.

[20] Python. Python package index, mai 2021. Acesso em: 20 de Maio de 2021.

[21] CODING TECH. Publishing (perfect) python packages on pypiyoutube, jul 2019. Acesso em: 25 abr. 2021.

[22] Python Software Foundation. Packaging python projects — python packaging user guide, jul 2021. Acesso em: 3 abr. 2021.

[23] JONAS NEUBERT. Publishing your first pypi package by/for the absolute beginner, sep 2017. Acesso em: 8 maio. 2021.

[24] Osvaldo Santana. Python: Criando pacotes e módulos, jan 2007. Acesso em: 2 abr. 2021.

[25] Neri Aldoir Neitzke. Boas práticas de programação: Organizando o código e seus pacotes, jun 2011. Acesso em: 29 mar. 2021.

[26] Python Sotware Foundation. Escrevendo arquivo de configuração de intalação, ago 2021. Acesso em: 20 de Ago. de 2021.

[27] Daniel Holth. Python: Criando pacotes e módulos, jun 2012. Acesso em: 19 abr. 2021.

[28] Donald Stufft and individual contributors. Twine, jul 2021. Acesso em: 21 de Ago. de 2021.

[29] The pip developers. User guide - pip documentation v21.2.dev0, jun 2020. Acesso em: 11 maio. 2021.

[30] DevFuria. Importando módulos no python (imports), jun 2012. Acesso em: 2 abr. 2021.

7 APÊNDICE A - Código RocketCasing.py

```
    class Casing():
    def cas():
    #Chama os pacotes usados no programa
        import math
        import matplotlib.pyplot  #Necessario somente caso for construir o grafico, entretanto o grafico nao e necessario.
        import materials
        from datetime import datetime

#Abertura do programa
        print("--Casing--")
#Solicitacao da insercao de dados
        print ("---Dimensoes do involucro e Fatores de design---")
        Do= float(input('Insira o diametro externo (mm): '))
        t= float(input('Insira a espessura da parede (mm): '))

#Opcao para escolher a entrada e, consequentemente, a saida
        print(" ")
        print("Para inserir, pressione:")
        print("0 para o valor do Fator de design de segurança ")
        print("1 para o valor do Fator de segurança de ruptura")
        op=str(input("") )

        if op=="0":
            Sd= float(input('Insira o Fator de design de segurança: '))
        elif op=="1":
            Su= float(input('Insira o Fator de segurança de ruptura: '))
#Mostrar lista de materiais
        opcao=input("Deseja ver a lista de materiais? (S/N) ")
        if opcao=="S" or opcao=="s":
            for materials.mec_propk, materials.mec_propv in materials.mec_prop.items():
                print(f'{materials.mec_propk}')
        else:
            pass
        prop = input('Digite o tipo do material do involucro: ')
        p=materials.mec_prop[prop]
#Saida dos resultados
        print(" ")
        print("-------------------------")
        print("Material: ", prop)
        print("Caracteristicas do material: ", '{}'.format(p))
        b=p['Fty']/p['Ftu'] if p['Fty']!=0 or p['Ftu']!=0 else 0
        print("Razao da forca do material: b= ",'{:.5f}'.format(b))
        B=(9.5833*b**4)+(-33.528*b**3)+(44.929*b**2)+(-28.479*b)+8.6475 if
```

```
    b!=0 else 0
44        print("Fator de ruptura: B= ",'{:.5f}'.format(B))
45
46        if op=="0":
47            print(" ")
48            print ("---Pressoes de design e ruptura---")
49            Pd=(2*(t*p['Fty']*1000)/(Do*Sd)) if Do!=0 or Sd!=0 else 0
50            print("Design de pressao: Pd= ",'{:.5f}'.format(Pd)," kPa")
51            Pu=(2*B*t*p['Fty']*1000/Do) if Do!=0 else 0
52            print("Pressao de ruptura: Pu= ",'{:.5f}'.format(Pu), " kPa")
53            Su=Pu/Pd if Pd!=0 else 0
54            print("Fator de seguranca de ruptura: Su= ",'{:.5f}'.format(Su)
    )
55        elif op=="1":
56            print(" ")
57            print ("---Pressoes de design e ruptura---")
58            Pu=(2*B*t*p['Fty']*1000/Do) if Do!=0 else 0
59            print("Pressao de ruptura: Pu= ",'{:.5f}'.format(Pu), " kPa")
60            Pd=Pu/Su if Su!=0 else 0
61            print("Design de pressao: Pd= ",'{:.5f}'.format(Pd)," kPa")
62            Sd=(2*(t*p['Fty']*1000)/(Do*Pd)) if Do!=0 or Pd!=0 else 0 #
    corrigir divisao por 0
63            print("Fator de design de segurança: Sd= ",'{:.5f}'.format(Sd))
64
65        print (" ")
66        print(" ")
67        print ("---Deformaçao elastica sobre pressao---")
68        dD=(2*Pd*(Do/2)**2/p['E']/10**6/t*(1-p['v']/2)) if p['E']!=0 or p['
    v']!=0 else 0
69        print("Variacao do diametro do involucro: dD= ",'{:.6f}'.format(dD)
    ," m")
70        dc=((dD*math.pi))
71        print("Variacao da circunferencia do involucro: dc= ",'{:.6f}'.
    format(dc)," m")
72
73 #Construcao grafico (de acordo com a planilha do Nakka)
74
75     matplotlib.pyplot.title('Fator de ruptura para cilindros pressurizados'
    )
76     matplotlib.pyplot.xlabel('Fty/Ftu')
77     matplotlib.pyplot.ylabel('B')
78     matplotlib.pyplot.plot([0.5, 0.6, 0.7, 0.8, 0.9, 1], [2.048, 1.735,
    1.527, 1.379, 1.254, 1.153], color='g', marker='o')
79     matplotlib.pyplot.axis([0.5, 1, 1, 2.2]) # [xmin, xmax, ymin, ymax]
80     matplotlib.pyplot.show()
81
82     #o grafico apresenta uma curva para o fator de explosao para cilindros
```

```
    pressurizados. Uma curva fixa que apresenta os melhores valores.
83
84  #construcao do arquivo em txt com os valores
85          data=datetime.now()
86          datatxt=data.strftime('%d/%m/%Y %H:%M')
87
88          arquivo = open('casing_results.txt', 'a')
89          arquivo.write('\n')
90          arquivo.write(datatxt)
91          arquivo.write('\n')
92          arquivo.write('Material: ')
93          arquivo.write(prop)
94          arquivo.write('\n')
95          arquivo.write('Caracteristicas do material: ')
96          arquivo.write('{}'.format(p))
97          arquivo.write('\n')
98          arquivo.write('Razao da forca do material: b= ')
99          arquivo.write('{}'.format(b))
100         arquivo.write('\n')
101         arquivo.write('Fator de ruptura: B= ')
102         arquivo.write('{}'.format(B))
103         arquivo.write('\n')
104         arquivo.write('Design de pressao: Pd= ')
105         arquivo.write('{}'.format(Pd))
106         arquivo.write(' kPa')
107         arquivo.write('\n')
108         arquivo.write('Pressao de ruptura: Pu= ')
109         arquivo.write('{}'.format(Pu))
110         arquivo.write(' kPa')
111         arquivo.write('\n')
112         if op=="0":
113             arquivo.write('Fator de seguranca de ruptura: Su= ')
114             arquivo.write('{}'.format(Su))
115             arquivo.write('\n')
116         if op=="1":
117             arquivo.write('Fator de design de seguranca: Sd= ')
118             arquivo.write('{}'.format(Sd))
119             arquivo.write('\n')
120         arquivo.write('Variacao no diametro do involucro: dD= ')
121         arquivo.write('{}'.format(dD))
122         arquivo.write('\n')
123         arquivo.write(' m')
124         arquivo.write('Variacao na circunferencia do involucro: dc= ')
125         arquivo.write('{}'.format(dc))
126         arquivo.write(' m')
127         arquivo.write('\n')
128         arquivo.write('--------------------------------')
```

```
        arquivo.write('\n')
        arquivo.write(' ')
        arquivo.close()

Casing.cas()
```

8 APÊNDICE B - Código materials.py

```
#Yield Strength (Fty) Forca de Rendimento
#Ultimate Strength (Ftu) Forca máxima
#modulo de elasticidade (E) Modulo de elasticidade
#coeficiente de poisson (v) Coeficiente de Poisson
mec_prop={
    'aco c 1010 laminacao quente':{
        'Fty':165,
        'Ftu':296,
        'E':200100,
        'v':0.32},
    'aco c 1010 laminacao fria':{
        'Fty':414,
        'Ftu':496,
        'E':200100,
        'v':0.32},
    'aco c 1015 laminacao quente':{
        'Fty':228,
        'Ftu':379,
        'E':200100,
        'v':0.32},
    'aco c 1015 normalizado':{
        'Fty':241,
        'Ftu':345,
        'E':200100,
        'v':0.32},
    'aco c 1025 laminacao quente':{
        'Fty':310,
        'Ftu':462,
        'E':200100,
        'v':0.32},
    'aco c 1025 normalizado':{
        'Fty':331,
        'Ftu':448,
        'E':200100,
        'v':0.32},
    'aco c 1025 laminacao fria':{
        'Fty':483,
```

```
        'Ftu':586,
        'E':200100,
        'v':0.32},
    'aco inox 304':{
        'Fty':517,
        'Ftu':724,
        'E':186300,
        'v':0.27},
    'aco inox 301':{
        'Fty':517,
        'Ftu':862,
        'E':186300,
        'v':0.27},
    'aco aisi 4130 normalizado MIL-T-6736':{
        'Fty':517,
        'Ftu':655,
        'E':200100,
        'v':0.32},
    'aco aisi 4130 laminacao fria MIL-T-6736':{
        'Fty':621,
        'Ftu':690,
        'E':200100,
        'v':0.32},
    'aluminio 6061-t4':{
        'Fty':110,
        'Ftu':207,
        'E':68310,
        'v':0.33},
    'aluminio 6061-t6':{
        'Fty':241,
        'Ftu':290,
        'E':68310,
        'v':0.33},
    'aluminio 6061-t6511 extrusado':{
        'Fty':241,
        'Ftu':290,
        'E':68310,
        'v':0.33},
    'aluminio 2024-t3':{
        'Fty':310,
        'Ftu':455,
        'E':72450,
        'v':0.33},
    'aluminio 2024-t42':{
        'Fty':262,
        'Ftu':427,
        'E':72450,
```

```
        'v':0.33},
    'aluminio 7075-t6':{
        'Fty':455,
        'Ftu':531,
        'E':71760,
        'v':0.33},
    'aluminio 7075-t73':{
        'Fty':386,
        'Ftu':455,
        'E':71760,
        'v':0.33},
    'policloreto de vinila':{
        'Fty':41,
        'Ftu':51,
        'E':2898,
        'v':0.41},
    'acrilonitrila butadieno estireno':{
        'Fty':35,
        'Ftu':41,
        'E':2001,
        'v':0.0},
    'papelao':{
        'Fty':0,
        'Ftu':14,
        'E':0,
        'v':0.0},
    'tubo de metal eletrico anelado':{
        'Fty':296,
        'Ftu':393,
        'E':200100,
        'v':0.32},
    'tubo de metal eletrico laminado':{
        'Fty':310,
        'Ftu':414,
        'E':200100,
        'v':0.32}
    }
```

9 APÊNDICE C - Código Ezalt.py

```
class Ezalt():
    def ezalt():
        import math
        import matplotlib.pyplot
        from datetime import datetime
```

```
6
7  #Abertura do programa
8          print(" ")
9          print("--Ezalt--")
10
11 #input do programa
12         title=str(input(("Insira o nome do foguete: ")))
13         F=float(input(("Insira o Impulso do motor (N): ")))
14         It=float(input(("Insira o Impulso Total do motor (N-sec): ")))
15         Mp=float(input(("Insira a Massa do propelente do motor (kg): ")))
16         Mr=float(input(("Insira a Massa morta do foguete (kg): "))) #massa
       do foguete sem o motor
17         D=float(input(("Insira o diametro do foguete (cm): ")))
18         Cd=float(input(("Insira o coeficiente de arrasto (tipicamente entre
        0,3 a 0,6): ")))
19
20         print(" ")
21 #cálculos e outputs do programa
22         print("------------------------")
23         print(" ")
24 #tempo de impulso do motor
25         t=It/F if F!=0 else 0
26         print("Tempo de impulso do motor: t= ",'{:.3f}'.format(t)," s")
27
28         if It>=1.3 and It<2.5:
29             Classe="A"
30         elif It>=2.5 and It<5:
31             Classe="B"
32         elif It>=5 and It<10:
33             Classe="C"
34         elif It>=10 and It<20:
35             Classe="D"
36         elif It>=20 and It<40:
37             Classe="E"
38         elif It>=40 and It<80:
39             Classe="F"
40         elif It>=80 and It<160:
41             Classe="G"
42         elif It>=160 and It<320:
43             Classe="H"
44         elif It>=320 and It<640:
45             Classe="I"
46         elif It>=640 and It<1280:
47             Classe="J"
48         else:
49             Classe="K"
50         print("Classificacao do motor: ", Classe)
```

```
51
52          Mra=Mr+Mp/2
53          print("Massa media de voo do foguete: Mra= ",'{:.3f}'.format(Mra)," Kg")
54
55          a=F/Mra-9.808
56          print("Aceleracao: a= ",'{:.3f}'.format(a)," m/s^2")
57
58          print(" ")
59          print("Ideal -  Sem resistencia de arrasto")
60          print(" ")
61
62          z1=1/2*(F/Mra-9.808)*t**2
63          print("Altitude de burnout: ",'{:.3f}'.format(z1)," m")
64
65          z2=F*z1/Mra/9.808
66          print("Altitude de pico: ",'{:.3f}'.format(z2)," m")
67
68
69          t2=t+math.sqrt(2*(z2-z1)/9.808)
70          print("Tempo para atingir altitude de pico: ",'{:.3f}'.format(t2)," s")
71
72          v1=math.sqrt((2*z1)/Mra*(F-Mra*9.808))
73          print("Velocidade maxima: ",'{:.3f}'.format(v1)," m/s")
74
75          print("-------------------------")
76          print("Fatores de reducao de arrasto")
77          print(" ")
78
79          N=Cd*D**2*v1**2/Mr/1000
80          print("Numero de influencia de arrasto: ",'{:.3f}'.format(N))
81
82          if N>900:
83              print("(Fora do alcance valido!)")
84              print(" ")
85  #f1: fator de reducao de altitude de pico. Altitude maxima e reduzida devido ao arrasto aerodinamico
86          if 1/(1.049909+0.001719*(N**1.0042225))>1 :
87              f1=1
88          else:
89              f1=1/(1.049909+0.001719*(N**1.0042225))
90          print("Fator de reducao de altitude de pico: ",'{:.3f}'.format(f1)," m")
91
92  #f2: fator de reducao de tempo de altitude. Tempo para atingir o pico e reduzido pelo arrasto aerodinamico
```

```
93          if 1/(1.048224+0.001093*(N**0.97255))>1 :
94              f2=1
95          else:
96              f2=1/(1.048224+0.001093*(N**0.97255))
97          print("Fator de reducao de tempo de altitude: ",'{:.3f}'.format(f2)
    ," s")
98
99  #f3: fator de reducao da velocidade maxima. A velocidade maxima e reduzida
    devido ao arrasto aerodinamico
100         if (0.99769-0.000075691*N)>1 :
101             f3=1
102         else:
103             f3=(0.99769-0.000075691*N)
104         print("Fator de reducao de velocidade maxima: ",'{:.3f}'.format(f3)
    ," m/s")
105
106 #f4: fator de reducao de altitude de burnout. A altitude do burnout e
    reduzido devido ao arrasto aerodinamico
107         if (0.99973-0.000043807*N)>1 :
108             f4=1
109         else:
110             f4=(0.99973-0.000043807*N)
111         print("Fator de reducao de altitude de burnout: ",'{:.3f}'.format(
    f4)," m")
112
113         print("-------------------------")
114         print("Predito com arrasto")
115         print(" ")
116
117         Zpico=f1*z2
118         print("Altitude de pico: Zpico= ",'{:.3f}'.format(Zpico)," m")
119
120         tpico=f2*t2
121         print("Tempo para Altitude de pico: tpico= ",'{:.3f}'.format(tpico)
    ," s")
122
123         Vmax=f3*v1
124         print("Velocidade maxima: vmax= ",'{:.3f}'.format(Vmax)," m/s")
125
126         Zbo=f4*z1
127         print("Altitude de burnout: Zbo= ",'{:.3f}'.format(Zbo)," m")
128
129         if v1>331 :
130             print(" ")
131             print("AVISO: Foguete Supersonico. Resultados podem ser
    invalidos")
132
```

```
133         print("-------------------------")
134         print("Melhor massa")
135 
136         Mmin=float(input("Insira a massa minima (kg): "))
137 
138         while Mmin<=0 :
139             print("O valor deve ser maior que zero")
140             Mmin=float(input("Insira a massa minima (kg): "))
141 
142         Mmax=float(input("Insira a massa maxima (kg): "))
143 
144         i=(Mmax-Mmin)/23
145     #print("Intervalo: ",'{:.2f}'.format(i), ' Kg'))
146 
147         Mmassa=(i*5)+Mmin
148 
149         print("Melhor massa: ",'{:.2f}'.format(Mmassa), ' Kg')
150 #construção do grafico
151         m=Mmin
152 
153         matplotlib.pyplot.title('Altitude por massa')
154         matplotlib.pyplot.xlabel('Massa (Kg)')
155         matplotlib.pyplot.ylabel('Altitude (m)')
156         matplotlib.pyplot.grid(True)
157 
158         matplotlib.pyplot.plot([m, m+i,m+2*i,m+3*i,m+4*i,m+5*i,m+6*i,m+7*i,
    m+8*i,m+9*i,m+10*i,m+11*i,m+12*i,m+13*i,m+14*i,m+15*i,m+16*i,m+17*i,m
    +18*i,m+19*i,m+20*i,m+21*i,m+22*i
159                                 ,m+23*i], [282, 445, 588, 698, 767, 796,
    790, 758, 711, 656, 600, 545, 493, 446, 404, 366, 333, 303, 276, 253,
    232, 214, 197, 182], color='g',marker='o')
160         matplotlib.pyplot.annotate('Altitude Máx.', xy=(m+5*i, 796), xytext
    =(3.5, 600), arrowprops=dict(facecolor='black', shrink=0.1))
161 
162         matplotlib.pyplot.show()
163 
164 #construção do arquivo em txt com os valores
165         data=datetime.now()
166         datatxt=data.strftime('%d/%m/%Y %H:%M')
167 
168         arquivo = open('ezalt_results.txt', 'a')
169         arquivo.write('\n')
170         arquivo.write(datatxt)
171         arquivo.write('\n')
172         arquivo.write('\n')
173         arquivo.write('Título: ')
174         arquivo.write(title)
```

```
        arquivo.write('\n')
        arquivo.write('Impulso do motor: F= ')
        arquivo.write('{}'.format(F))
        arquivo.write(' N')
        arquivo.write('\n')
        arquivo.write('Impulso total: It= ')
        arquivo.write('{}'.format(It))
        arquivo.write(' N-sec')
        arquivo.write('\n')
        arquivo.write('Massa do propelente: Mp= ')
        arquivo.write('{}'.format(Mp))
        arquivo.write(' Kg')
        arquivo.write('\n')
        arquivo.write('Massa morta: Mr= ')
        arquivo.write('{}'.format(Mr))
        arquivo.write(' Kg')
        arquivo.write('\n')
        arquivo.write('Diametro do foguete: D= ',)
        arquivo.write('{}'.format(D))
        arquivo.write(' cm')
        arquivo.write('\n')
        arquivo.write('Coeficiente de Arrasto: Cd= ')
        arquivo.write('{}'.format(Cd))
        arquivo.write('\n')
        arquivo.write('Tempo de impulso do motor: t= ')
        arquivo.write('{}'.format(t))
        arquivo.write(' s')
        arquivo.write('\n')
        arquivo.write('Classificacao do motor: ')
        arquivo.write(Classe)
        arquivo.write('\n')
        arquivo.write('Massa media de voo: Mra= ')
        arquivo.write('{}'.format(Mra))
        arquivo.write(' Kg')
        arquivo.write('\n')
        arquivo.write('Aceleracao: a= ')
        arquivo.write('{}'.format(a))
        arquivo.write(' m/s²')
        arquivo.write('\n')
        arquivo.write('-----------------------------------')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Ideal - Sem resistencia de arrasto')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Altitude de Burnout: z1= ')
        arquivo.write('{}'.format(z1))
```

```
        arquivo.write(' m')
        arquivo.write('\n')
        arquivo.write('Altitude de pico: z2= ')
        arquivo.write('{}'.format(z2))
        arquivo.write(' m')
        arquivo.write('\n')
        arquivo.write('Tempo para altitude de pico: t2= ')
        arquivo.write('{}'.format(t2))
        arquivo.write(' s')
        arquivo.write('\n')
        arquivo.write('Velocidade maxima: v1= ')
        arquivo.write('{}'.format(v1))
        arquivo.write(' m/s')
        arquivo.write('\n')
        arquivo.write('-----------------------------------')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Fatores de reducao de arrasto')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Numero de influencia de arrasto: N= ')
        arquivo.write('{}'.format(N))
        arquivo.write('\n')
        arquivo.write('Fator de reducao de altitude de pico: f1= ')
        arquivo.write('{}'.format(f1))
        arquivo.write(' m')
        arquivo.write('\n')
        arquivo.write('Fator de reducao de tempo de altitude: f2= ')
        arquivo.write('{}'.format(f2))
        arquivo.write(' s')
        arquivo.write('\n')
        arquivo.write('Fator de reducao de velocidade maxima: f3= ')
        arquivo.write('{}'.format(f3))
        arquivo.write(' m/s')
        arquivo.write('\n')
        arquivo.write('Fator de reducao de altitude de burnout: f4= ')
        arquivo.write('{}'.format(f4))
        arquivo.write(' m')
        arquivo.write('\n')
        arquivo.write('-----------------------------------')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Predito com arrasto')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Altitude de pico: Zpico= ')
        arquivo.write('{}'.format(Zpico))
```

```
        arquivo.write(' m')
        arquivo.write('\n')
        arquivo.write('Tempo para altitude de pico: tpico= ')
        arquivo.write('{}'.format(tpico))
        arquivo.write(' s')
        arquivo.write('\n')
        arquivo.write('Velocidade maxima: Vmax= ')
        arquivo.write('{}'.format(Vmax))
        arquivo.write(' m/s')
        arquivo.write('\n')
        arquivo.write('Altitude de burnout: Zbo= ')
        arquivo.write('{}'.format(Zbo))
        arquivo.write(' m')
        arquivo.write('\n')
        if v1==331:
            arquivo.write('AVISO: Foguete Supersonico. Resultados podem ser
    invalidos')
        arquivo.write('\n')
        arquivo.write('-----------------------------------')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Melhor Massa')
        arquivo.write('\n')
        arquivo.write('\n')
        arquivo.write('Melhor massa: Mm= ')
        arquivo.write('{}'.format(Mmassa))
        arquivo.write(' Kg')
        arquivo.write('\n')
        arquivo.write('===================================')
        arquivo.close()

Ezalt.ezalt()
```

10 APÊNDICE D - Trecho Download RocketCasing.py

```
    C:\Users\Extra\Desktop\RocketCasing\Casing>python -m pip install
   RocketCasing
Collecting RocketCasing
  Using cached RocketCasing-1.4.19.tar.gz (8.4 kB)
Collecting peppercorn
  Using cached peppercorn-0.6-py3-none-any.whl (4.8 kB)
Collecting matplotlib
  Downloading matplotlib-3.4.3-cp39-cp39-win_amd64.whl (7.1 MB)
     |################################| 7.1 MB 2.2 MB/s
Collecting datetime
```

```
  Using cached DateTime-4.3-py2.py3-none-any.whl (60 kB)
Collecting zope.interface
  Downloading zope.interface-5.4.0-cp39-cp39-win_amd64.whl (210 kB)
     |################################| 210 kB 2.2 MB/s
Collecting pytz
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |################################| 247 kB 2.2 MB/s
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\extra\appdata\
    roaming\python\python39\site-packages (from matplotlib->RocketCasing)
    (2.4.7)
Collecting pillow>=6.2.0
  Downloading Pillow-8.3.1-1-cp39-cp39-win_amd64.whl (3.2 MB)
     |################################| 3.2 MB 1.6 MB/s
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.1-cp39-cp39-win_amd64.whl (51 kB)
     |################################| 51 kB 33 kB/s
Collecting numpy>=1.16
  Downloading numpy-1.21.2-cp39-cp39-win_amd64.whl (14.0 MB)
     |################################| 14.0 MB 2.2 MB/s
Collecting cycler>=0.10
  Using cached cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Requirement already satisfied: six in c:\users\extra\appdata\roaming\python
    \python39\site-packages (from cycler>=0.10->matplotlib->RocketCasing)
    (1.16.0)
Requirement already satisfied: setuptools in c:\users\extra\appdata\local\
    programs\python\python39\lib\site-packages (from zope.interface->
    datetime->RocketCasing) (56.0.0)
Using legacy 'setup.py install' for RocketCasing, since package 'wheel' is
    not installed.
Installing collected packages: zope.interface, pytz, python-dateutil,
    pillow, numpy, kiwisolver, cycler, peppercorn, matplotlib, datetime,
    RocketCasing
    Running setup.py install for RocketCasing ... done
Successfully installed RocketCasing-1.4.19 cycler-0.10.0 datetime-4.3
    kiwisolver-1.3.1 matplotlib-3.4.3 numpy-1.21.2 peppercorn-0.6 pillow
    -8.3.1 python-dateutil-2.8.2 pytz-2021.1 zope.interface-5.4.0
```