PRJ - 22 - Projeto Conceitual de Aeronave
Lab 05
Atividade Individual

Cap Eng **Ney** Rafael Secco
ney@ita.br

1T Eng João A. **Dantas** de J. Ferreira
dantas@ita.br

Instruções

1. Data de entrega: 17/10/2021, 23:59;
2. Envie a solução para dantas@ita.br
3. Todos devem enviar os relatórios através do Google Classroom - enviar um aquivo compactado (.zip ou .rar) com o relatório e o código escrito - o código deve ter comentários;
4. Deduções por atraso: a cada dia, 2,0 pontos são descontados da nota final;
5. Todos devem entregar um pequeno relatório com os itens marcados em vermelho;
6. Podem discutir soluções com outros alunos, mas não podem compartilhar códigos.

1 Pesos

Nesse Lab, trabalharemos com três módulos em Python3.

1.1 Módulo de peso vazio

Esse módulo estima o peso vazio da aeronave ao adicionar contribuições de cada um de seus componentes. Nós usaremos algumas tendências históricas da Fig. 1 para isso.

	Fighters		Transport & Bomber		General aviation			
	lb/ft^2	kg/m^2	lb/ft^2	kg/m^2	lb/ft^2	kg/m^2	Multiplier	Approximate location
Wing	9	44	10	49	2.5	12	$S_{\text{exposed planform}}$	40% MAC
Horizontal tail	4	20	5.5	27	2	10	$S_{\text{exposed planform}}$	40% MAC
Vertical tail	5.3	26	5.5	27	2	10	$S_{\text{exposed planform}}$	40% MAC
Fuselage	4.8	23	5	24	1.4	7	$S_{\text{wetted area}}$	40–50% length
	Weight ratio		Weight ratio		Weight ratio			
Landing gear*	0.033		0.043		0.057		TOGW	centroid
Landing gear—Navy	0.045		—		—		TOGW	centroid
Installed engine	1.3		1.3		1.4		Engine weight	centroid
"All-else empty"	0.17		0.17		0.1		TOGW	40–50% length

*15% to nose gear, 85% to main gear; reduce gear weight by 0.014 W_0 if fixed gear.

Figura 1: Contribuições para o peso vazio estimados por Raymer (Tab. 15.2).

Nós vamos usar o mesmo módulo para estimar a posição do CG da aeronave vazia.

1.1.1 Inputs

Os inputs para este módulo são:

- Parâmetro de peso: W_0.
- Parâmetros da asa: S_w, AR_w, λ_w, Λ_w, $c_{r,w}$, $x_{m,w}$, $c_{m,w}$, e $(t/c)_{r,w}$.
- Parâmetros da empenagem horizontal: S_h, $x_{m,h}$, e $c_{m,h}$.
- Parâmetros da empenagem vertical: S_v, $x_{m,v}$, e $c_{m,v}$.
- Parâmetros da fuselagem: L_f, D_f, e $S_{wet,f}$.
- Parâmetros dos motores: n_{eng}, T_0, BPR, x_n, e L_n.
- Parâmetros do trem de pouso: x_{nlg} e x_{mlg}.

1.1.2 Outputs

Os outputs desse módulo são:

- Parâmetro de peso: W_e.
- Parâmetro de CG: $x_{CG,e}$.
- *Array* de pesos: `weightsvec`

1.1.3 Chamada da função

Para calcular o peso vazio da aeronave, cada aluno deve utilizar a função como a seguir:

```
def empty_weight(aircraft, W0_guess, T0_guess):
    .
    .
    .
  return We, xcg_e, weightsvec
```

A variável `weightsvec` é um *array* com a seguinte constituição:
`weightsvec = [W_w,  W_h, W_v, W_f, W_nlg, W_mlg, W_eng_installed, W_allelse]`

1.1.4 Caso de testes

Considere um caso de testes com os inputs padrão e:

```
W0_guess = 422712.9
T0_guess = 125600
```

Você deve obter os seguintes resultados:

```
We = 232448.8612514016
xcg_e = 17.311278299071514
weightsvec = 32939.95933267459
             4819.756583850933
             3962.4552
             69621.52083852515
             2726.498205
             15450.156495
             31067.321596350914
             71861.19300000001
```

1.2 Módulo de peso de combustível

Esse módulo calcula a contribuição do combustível para o *Maximum Takeoff Weight*, Peso Máximo de Decolagem (MTOW). Algumas das frações de combustível podem ser estimadas usando os dados históricos mostrados na Fig. 2.

Table 2.1 Suggested Fuel-Fractions For Several Mission Phases

Mission Phase No.(See Fig.2.1) / Airplane Type:	Engine Start, Warm-up 1	Taxi 2	Take-off 3	Climb 4	Descent 7	Landing Taxi, Shutdown 8
1. Homebuilt	0.998	0.998	0.998	0.995	0.995	0.995
2. Single Engine	0.995	0.997	0.998	0.992	0.993	0.993
3. Twin Engine	0.992	0.996	0.996	0.990	0.992	0.992
4. Agricultural	0.996	0.995	0.996	0.998	0.999	0.998
5. Business Jets	0.990	0.995	0.995	0.980	0.990	0.992
6. Regional TBP's	0.990	0.995	0.995	0.985	0.985	0.995
7. Transport Jets	0.990	0.990	0.995	0.980	0.990	0.992
8. Military Trainers	0.990	0.990	0.990	0.980	0.990	0.995
9. Fighters	0.990	0.990	0.990	0.96-0.90	0.990	0.995
10. Mil.Patrol, Bomb, Transport	0.990	0.990	0.995	0.980	0.990	0.992
11. Flying Boats, Amphibious, Float Airplanes	0.992	0.990	0.996	0.985	0.990	0.990
12. Supersonic Cruise	0.990	0.995	0.995	0.92-0.87	0.985	0.992

Notes: 1. The numbers in this table are based on experience or on judgment.
2. There is no substitute for common sense! If and when common sense so dictates, the reader should substitute other values for the fractions suggested in this table.

Figura 2: Frações de combustível estimados por Roskam.

1.2.1 Inputs

Os inputs para este módulo são:

- Parâmetro de peso: W_0.
- Parâmetro da asa: S_w.
- Parâmetros de cruzeiro: $C_{D0,cruise}$, K_{cruise}, h_{cruise}, M_{cruise}, R_{cruise}, e C_{cruise}.
- Parâmetro de espera: E_{loiter}.
- Parâmetros de cruzeiro alternativo: $C_{D0,altcruise}$, $K_{altcruise}$, $h_{altcruise}$, $M_{altcruise}$, $R_{altcruise}$, e $C_{altcruise}$.

1.2.2 Outputs

Os outputs deste módulo são:

- Parâmetros de peso: W_f e $M_{f,cruise}$.

1.2.3 Chamada da função

Para calcular o peso de combustível da aeronave, cada aluno deve utilizar a função como a seguir:

```
def fuel_weight(aircraft, W0_guess
                CD0_cruise, K_cruise, altitude_cruise, Mach_cruise, range_cruise, C_cruise,
                loiter_time,
                CD0_altcruise, K_altcruise, altitude_altcruise, Mach_altcruise, range_altcruise,
                                                C_altcruise):
.
.
.
return Wf, Mf_cruise
```

1.2.4 Caso de testes

Considere um caso de testes com os inputs anteriores e:

```
CD0_cruise = 0.01857763638636
K_cruise = 0.04747410535245
altitude_cruise = 11000.00000000000000
Mach_cruise = 0.77000000000000
range_cruise = 2390000.00000000000000
C_cruise = 0.00019859928416

loiter_time = 2700.00000000000000

CD0_altcruise = 0.01948073140867
K_altcruise = 0.04633848260462
altitude_altcruise = 4572.00000000000000
Mach_altcruise = 0.40000000000000
range_altcruise = 370000.00000000000000
C_altcruise = 0.00018508237527
```

Você deve obter os seguintes resultados:

```
Wf = 99509.4311185458
Mf_cruise = 0.95569551
```

1.3 Módulo do Peso Máximo de Decolagem

Esse módulo implementa o processo iterativo para calcular o MTOW usando os módulos anteriores.

1.3.1 Inputs

Os inputs para esse módulo são:

- Parâmetros de peso: $W_{0,guess}$, $W_{payload}$, e W_{crew}.
- Parâmetros geométricos da asa: S_w, AR_w, $c_{r,w}$, λ_w, Λ_w, $(t/c)_{r,w}$, $(t/c)_{t,w}$, b_w, $x_{m,w}$, e $c_{m,w}$.
- Parâmetros geométricos da empenagem horizontal: S_h, $c_{r,h}$, λ_h, $(t/c)_{r,h}$, $(t/c)_{t,h}$, $x_{m,h}$, e $c_{m,h}$.
- Parâmetros geométricos da empenagem vertical: S_v, $c_{r,v}$, λ_v, $(t/c)_{r,v}$, $(t/c)_{t,v}$, $x_{m,v}$, e $c_{m,v}$.
- Parâmetros da fuselagem: L_f, e D_f.
- Parâmetros da nacele: L_n, D_n, e x_n.
- Parâmetros do flap: $\delta_{flap,LD}$, `flap_type`, c_{flap}/c_w, e b_{flap}/b_w.
- Parâmetros do slat: $\delta_{slat,LD}$, `slat_type`, c_{slat}/c_w, e b_{slat}/b_w.
- Parâmetros do arrasto de excrescência: k_{exc}.
- Parâmetros dos motores: n_{eng}, $n_{eng,w}$, T_0, e BPR.
- Parâmetros do trem de pouso: x_{nlg} e x_{mlg}.
- Parâmetros de cruzeiro: h_{cruise}, M_{cruise}, e R_{cruise}.
- Parâmetros da espera: E_{loiter}.
- Parâmetros de cruzeiro alternativo: $h_{altcruise}$, $M_{altcruise}$, e $R_{altcruise}$.

1.3.2 Outputs

Os outputs desse módulo são:

- Parâmetros de peso: W_0, W_e, W_f, $M_{f,cruise}$, e `weightsvec`.
- Parâmetro de CG: $x_{CG,e}$.

1.3.3 Chamada da função

Para calcular o peso máximo de decolagem da aeronave, cada aluno deve utilizar a função como a seguir:

```
def weight(aircraft, W0_guess, T0_guess,
           altitude_cruise, Mach_cruise, range_cruise,
           loiter_time,
           altitude_altcruise, Mach_altcruise, range_altcruise):
.
.
.
return W0, We, Wf, Mf_cruise, xcg_e, weightsvec
```

1.3.4 Caso de testes

Considere um caso de testes com os inputs anteriores e:

```
T0_guess = 125600
W0_guess = 422712.9

altitude_cruise = 11000.0000
Mach_cruise = 0.7700000
range_cruise = 2390000.00000

loiter_time = 2700.00000

altitude_altcruise = 4572.00000
Mach_altcruise = 0.40000000
range_altcruise = 370000.00000
```

Você deve obter os seguintes resultados:

```
W0 = 439776.60009163496
We = 236794.43249269313
Wf = 102998.64759894181
Mf_cruise = 0.95569551
xcg_e = 17.274672280063154
weightsvec = 33670.24024350807
             4819.756583850933
             3962.4552
             69621.52083852515
             2835.9753065561245
             16070.526737151373
             31067.321596350914
             74746.63598675058
```

1.3.5 Exercícios

Crie um gráfico de pizza (ou similar) mostrando o *breakdown* de pesos para a aeronave do caso de testes. Esse gráfico deve mostrar as porcentagens de peso correspondentes a carga-paga, tripulação, combustível e **cada** componente da aeronave.

Agora pegamos o caso de testes e iremos variar o alongamento da asa (AR_w), mantendo os outros parâmetros (independentes do alongamento) fixos.

Gere um plot de W_0 como função de AR_w (alongamento da asa) para o intervalo $7.0 < AR_w < 14$. Qual AR_w você escolheria baseando-se unicamente nesse plot?

Lista de siglas

MTOW *Maximum Takeoff Weight*, Peso Máximo de Decolagem

A Construção do código

Para entender o funcionamento do código fornecido, acompanhe as seções a seguir:

A.1 Módulo de consumo específico de combustível

Nós utilizaremos um módulo que calcula o consumo específico de combustível (C) dos motores de acordo com suas condições operacionais.

A.1.1 Inputs

Os inputs para esse módulo são:

- Parâmetro do motor: BPR.
- Parâmetros de condição de voo: M, e h.

A.1.2 Outputs

Os outputs desse módulo são:

- Parâmetro do motor: C.

A.1.3 Chamada da função

A função é como a seguir:

```
def engineTSFC(BPR, Mach, altitude):
.
.
.
return C
```

`BPR`, `Mach`, `altitude` e `C` são *floats*.

Inicialmente utilizamos a função auxiliar fornecida no arquivo `aux_tools.py` para obter as propriedades atmosféricas:

```
T,p,rho,mi = atmosphere(altitude, 288.15)
```

Então calcule a razão σ entre a densidade do ar atual e a densidade do ar ao nível do mar (que é igual a 1.225 kg/m^3):

$$\sigma = \frac{\rho}{1.225} \tag{1}$$

Determine o consumo específico de combustível base de acordo com o tipo de motor:

- Motores de baixo *bypass ratio* (< 4.0): $C_{base} = 0.85/3600$ 1/s
- Motores de alto *bypass ratio* (≥ 4.0): $C_{base} = 0.70/3600$ 1/s

Agora compute o consumo específico de combustível nas condições atuais de voo com (Howe Eq. 3.12a):

$$C = C_{base} \cdot \left(1 - 0.15 \cdot BPR^{0.65}\right) \cdot \left(1 + 0.28 \cdot \left(1 + 0.063 \cdot BPR^2\right) \cdot M\right) \cdot \sigma^{0.08} \tag{2}$$

A.2 Módulo de peso vazio

A.2.1 Peso da asa

Calcule o fator último de carga com:

$$N_z = 1.5 \cdot 2.5 \tag{3}$$

Estime a área de superfícies de controle como 15% da área total da asa:

$$S_{csw} = 0.15 \cdot S_w \tag{4}$$

Nós podemos estimar o peso da asa com a expressão a seguir (Raymer Eq. 15.25):

$$W_w = 0.0051 \cdot (W_0 \cdot N_z)^{0.557} \cdot S_w^{0.649} \cdot AR_w^{0.55} \cdot (t/c)_{r,w}^{-0.4} \cdot (1 + \lambda_w)^{0.1} \cdot (\cos \Lambda_w)^{-1.0} \cdot S_{csw}^{0.1} \tag{5}$$

ATENÇÃO: Essa equação utiliza unidades imperiais (britânicas)! Todas as áreas estão em ft^2 e os pesos estão em lbs. Se lembrem de fazer as conversões antes e depois de usar essa equação.

A seguir, estimamos o centro de gravidade da asa de acordo com a Fig. 1:

$$x_{CG,w} = x_{m,w} + 0.4 \cdot c_{m,w} \tag{6}$$

NOTE: Aumentamos o expoente do AR_w de 0.5 para 0.55, quando comparado com o original na Eq. 15.25 no Raymer, já que não estava penalizando altos alongamentos o suficiente. Por exemplo, uma adaptação grosseira da Eq. C-1 do Torenbeek sugere que o expoente deveria estar mais próximo de 0.8.

A.2.2 Peso da empenagem horizontal

De acordo com a Fig. 1, podemos calcular o peso deste componente com:

$$W_h = 27 \cdot g \cdot S_h \tag{7}$$

na qual g é a aceleração da gravidade (9.81 m/s^2).
O centro de gravidade deste componente, também estimado com Fig. 1, é dado por:

$$x_{CG,h} = x_{m,h} + 0.4 \cdot c_{m,h} \tag{8}$$

A.2.3 Peso da empenagem vertical

De acordo com a Fig. 1, podemos calcular o peso deste componente com:

$$W_v = 27 \cdot g \cdot S_v \tag{9}$$

O centro de gravidade deste componente, também estimado com a Fig. 1, é dado por:

$$x_{CG,v} = x_{m,v} + 0.4 \cdot c_{m,v} \tag{10}$$

A.2.4 Peso da fuselagem

De acordo com a Fig. 1, podemos calcular o peso deste componente com:

$$W_f = 24 \cdot g \cdot S_{wet,f} \tag{11}$$

O centro de gravidade deste componente, também estimado com a Fig. 1, é dado por:

$$x_{CG,f} = 0.45 \cdot L_f \tag{12}$$

A.2.5 Peso do trem de pouso de nariz

De acordo com a Fig. 1, podemos calcular o peso deste componente com:

$$W_{nlg} = 0.15 \cdot 0.043 \cdot W_0 \tag{13}$$

Se lembre de usar `W0_guess` como W_0.
O centro de gravidade deste componente, também estimado com a Fig. 1, é dado por:

$$x_{CG,nlg} = x_{nlg} \tag{14}$$

A.2.6 Peso do trem de pouso principal

De acordo com a Fig. 1, podemos calcular o peso deste componente com:

$$W_{mlg} = 0.85 \cdot 0.043 \cdot W_0 \tag{15}$$

Se lembre de usar `W0_guess` como W_0.
O centro de gravidade deste componente, também estimado com a Fig. 1, é dado por:

$$x_{CG,mlg} = x_{mlg} \tag{16}$$

A.2.7 Peso do motor instalado

Primeiro, determine a tração de decolagem de apenas um motor com:

$$T_{eng,s} = \frac{T_0}{n_{eng}} \tag{17}$$

Se lembre de usar `T0_guess` como T_0 na equação acima.
O peso do motor isolado pode ser estimado com (Raymer Eq. 10.4):

$$W_{eng,s} = 14.7 \cdot g \cdot \left(\frac{T_{eng}}{1000}\right)^{1.1} \cdot e^{-0.045 \cdot BPR} \tag{18}$$

O peso do motor instalado, de acordo com a Fig. 1, é dado por:

$$W_{eng} = 1.3 \cdot n_{eng} \cdot W_{eng,s} \tag{19}$$

A Fig. 1 também nos permite estimar a posição do centro de gravidade dos motores instalados como:

$$x_{CG,eng} = x_n + 0.5 \cdot L_n \tag{20}$$

A.2.8 Peso de todo o resto

De acordo com a Fig. 1, o peso dos sistemas e componentes remanescentes pode ser calculada como:

$$W_{ae} = 0.17 \cdot W_0 \tag{21}$$

Se lembre de usar `W0_guess` como W_0.
Por agora, vamos calcular a posição do centro de gravidade correspondente usando os valores médios sugeridos pela Fig. 1:

$$x_{CG,ae} = 0.45 \cdot L_f \tag{22}$$

Vocês têm a liberdade de ajustar esse parâmetro para obter melhores margens estáticas mais tarde o projeto.

A.2.9 Peso vazio

O peso vazio total é a soma dos pesos dos componentes:

$$W_e = W_w + W_h + W_v + W_f + W_{nlg} + W_{mlg} + W_{eng} + W_{ae} \tag{23}$$

O centro de gravidade da aeronave fazia é a média ponderada dos centros de gravidade dos componentes:

$$x_{CG,e} = \frac{W_w x_{CG,w} + W_h x_{CG,h} + W_v x_{CG,v} + W_f x_{CG,f} + W_{nlg} x_{CG,nlg} + W_{mlg} x_{CG,mlg} + W_{eng} x_{CG,eng} + W_{ae} x_{CG,ae}}{W_e} \tag{24}$$

Esses são os dois outputs desse módulo.

A.3 Módulo de peso de combustível

A.3.1 Inicialização

Primeiro inicializemos a variável M_f que irá guardar os produtos de todas as frações de peso.

$$M_f = 1.0 \tag{25}$$

Nós vamos sobrescrever esse valor ao longo do código.

A.3.2 Ligar os motores

Use a Fig. 2 para obter a fração de combustível para essa fase e, em seguida, atualize o valor de M_f. Para jatos de transporte, temos:

$$M_f = M_f \cdot 0.990 \tag{26}$$

A.3.3 Taxi

Use a Fig. 2 para obter a fração de combustível para essa fase e, em seguida, atualize o valor de M_f. Para jatos de transporte, temos:

$$M_f = M_f \cdot 0.990 \tag{27}$$

A.3.4 Decolagem

Use a Fig. 2 para obter a fração de combustível para essa fase e, em seguida, atualize o valor de M_f. Para jatos de transporte, temos:

$$M_f = M_f \cdot 0.995 \tag{28}$$

A.3.5 Subida

Use a Fig. 2 para obter a fração de combustível para essa fase e, em seguida, atualize o valor de M_f. Para jatos de transporte, temos:

$$M_f = M_f \cdot 0.980 \tag{29}$$

A.3.6 Cruzeiro

Antes de qualquer cálculo acerca da fase de cruzeiro, nós precisamos armazenar a fração de combustível atual para cálculos posteriores. Portanto, defina uma variável chamada `Mf_cruise`, que irá guardar o valor atual de `Mf`:

$$M_{f,cruise} = M_f \tag{30}$$

Esse será um output desse módulo.

Podemos usar a equação de alcance de Breguet para estimar o consumo de combustível para a fase de cruzeiro. Inicialmente, precisamos determinar as propriedades do ar no nível de voo de cruzeiro. Podemos usar a função auxiliar fornecida no arquivo `aux_tools.py` para isso:

```
T,p,rho,mi = atmosphere(altitude_cruise, 288.15)
```

Assim que tivermos a temperatura do ar (T), podemos determinar a velocidade do som (a) com:

$$a_{cruise} = \sqrt{\gamma \cdot R \cdot T} \tag{31}$$

onde $\gamma = 1.4$ e $R = 287$ J/kg/K para o ar.

Então a velocidade de cruzeiro da aeronave (V_{cruise}) é:

$$V_{cruise} = M_{cruise} \cdot a_{cruise} \tag{32}$$

Podemos estimar o peso da aeronave no começo do cruzeiro com:

$$W_{cruise} = M_f \cdot W_0 \tag{33}$$

Se lembre de usar `W0_guess` como sua estimativa para W_0 na equação acima. Podemos usar a condição $L = W_{cruise}$ para calcular o coeficiente de sustentação de cruzeiro:

$$C_{L,cruise} = \frac{2 \cdot W_{cruise}}{\rho \cdot S_w \cdot V_{cruise}^2} \tag{34}$$

O coeficiente de arrasto correspondente pode ser calcular com a polar de arrasto:

$$C_{D,cruise} = C_{D0,cruise} + K_{cruise} \cdot C_{L,cruise}^2 \tag{35}$$

A fração de combustível de cruzeiro estimada pela equação de Breguet é:

$$FF_{cruise} = \exp\left(-\frac{R_{cruise} \cdot C_{cruise} \cdot C_{D,cruise}}{V_{cruise} \cdot C_{L,cruise}}\right) \tag{36}$$

Agora podemos agregar essa fração de combustível com o produto global:

$$M_f = M_f \cdot FF_{cruise} \tag{37}$$

A.3.7 Espera

O FAR 125.377 especifica que aeronaves de curto- e médio-alcance devem carregar combustível suficiente para uma espera de 45 minutos, enquanto que aeronaves de longo-alcance devem ser capaz de esperar por 10% do tempo originalmente estimado de voo.

A aeronave irá controlar sua velocidade de voo para que mantenha a melhor relação sustentação por arrasto durante a espera. O máximo L/D para a polar de arrasto parabólica é dada por:

$$L/D_{\max} = \frac{1}{2 \cdot \sqrt{C_{D0,cruise} \cdot K_{cruise}}} \tag{38}$$

Assumimos que os coeficientes da polar de arrasto vão se manter aproximadamente os mesmos entre o cruzeiro e a espera. Entretanto, podemos considerar que o consumo de combustível dos motores será reduzido em 20% devido às diferentes configurações de tração (Raymer Tab. 3.3).

$$C_{loiter} = 0.8 \cdot C_{cruise} \tag{39}$$

Agora a fração de combustível de espera será:

$$FF_{loiter} = \exp\left(-\frac{E_{loiter} \cdot C_{loiter}}{L/D_{\max}}\right) \tag{40}$$

Agora agregamos essa fração de combustível no produto global:

$$M_f = M_f \cdot FF_{loiter} \tag{41}$$

A.3.8 Descida

Use a Fig. 2 para obter a fração de combustível para essa fase e, em seguida, atualize o valor de M_f. Para jatos de transporte, temos:

$$M_f = M_f \cdot 0.990 \tag{42}$$

A.3.9 Cruzeiro alternativo

Aeronaves devem voar com combustível extra suficiente para chegar a um aeroporto de alternativa ou voar por um período de tempo adicional (see FAR 125.377). Nós podemos usar a mesma formulação da fase de cruzeiro para estimar a fração de combustível para o cruzeiro de alternativa.

Podemos usar a função auxiliar fornecida no arquivo `aux_tools.py` para obter as propriedades atmosféricas:

```
T,p,rho,mi = atmosphere(altitude_altcruise, 288.15)
```

Assim que obtemos a temperatura do ar (T), podemos determinar a velocidade do som (a) com:

$$a_{altcruise} = \sqrt{\gamma \cdot R \cdot T} \tag{43}$$

onde $\gamma = 1.4$ e $R = 287$ J/kg/K para o ar.

Então a velocidade de cruzeiro de alternativa ($V_{altcruise}$) é:

$$V_{altcruise} = M_{altcruise} \cdot a_{altcruise} \tag{44}$$

Podemos estimar o peso da aeronave no começo do cruzeiro de alternativa com:

$$W_{altcruise} = M_f \cdot W_0 \tag{45}$$

Se lembre de usar `W0_guess` como sua estimativa para W_0 na equação acima. Podemos usar a condição de $L = W_{altcruise}$ para calcular o coeficiente de sustentação de cruzeiro:

$$C_{L,altcruise} = \frac{2 \cdot W_{altcruise}}{\rho \cdot S_w \cdot V_{altcruise}^2} \tag{46}$$

O coeficiente de arrasto correspondente pode ser calculado com a polar de arrasto:

$$C_{D,altcruise} = C_{D0,altcruise} + K_{altcruise} \cdot C_{L,altcruise}^2 \tag{47}$$

A fração de combustível do cruzeiro de alternativa estimado pela equação de Breguet é:

$$FF_{altcruise} = \exp\left(-\frac{R_{altcruise} \cdot C_{altcruise} \cdot C_{D,altcruise}}{V_{altcruise} \cdot C_{L,altcruise}}\right) \tag{48}$$

Agora podemos agregar essa fração de combustível no produto global:

$$M_f = M_f \cdot FF_{altcruise} \tag{49}$$

A.3.10 Táxi de pouso e desligamento

Use a Fig. 2 para obter a fração de combustível para essa fase e, em seguida, atualize o valor de M_f. Para jatos de transporte, temos:

$$M_f = M_f \cdot 0.992 \tag{50}$$

A.3.11 Peso do Combustível

A variável M_f agora representa a razão entre os pesos final e inicial da aeronave. O peso de combustível é a diferença entre esses dois valores. Portanto, podemos usar M_f para calcular o peso de combustível como a seguir:

$$W_f = 1.06 \cdot (1 - M_f) \cdot W_0 \tag{51}$$

Mais uma vez, se lembre de usar `W0_guess` como uma estimativa para W_0 na equação acima. O fator de 6% se refere ao combustível preso nas linhas da aeronave (Raymer Eq. 3.13).

O peso de combustível (W_f) é o output final desse módulo.

A.4 Módulo do Peso Máximo de Decolagem

A.4.1 Cálculos preliminares

Primeiro precisamos obter alguns dados antes de rodar o processo iterativo para estimar o MTOW. Execute os passos seguintes:

1. Calcule o consumo específico de combustível dos motores na condição de cruzeiro (C_{cruise}) usando M_{cruise} e h_{cruise}.
2. Calcule o consumo específico de combustível na condição de cruzeiro de alternativa ($C_{altcruise}$) usando $M_{altcruise}$ e $h_{altcruise}$.
3. Defina uma variável auxiliar Δ que indique quando o processo iterativo deva parar:

$$\Delta = 1000 \tag{52}$$

A.4.2 Iteração do Peso Máximo de Decolagem

O MTOW deve ser determinado iterativamente dadas as suas relações implícitas com o peso de combustível e o peso vazio. O algoritmo 1 abaixo descreve esse processo iterativo:

Esse processo deve convergir em algumas iterações. O valor de W_0 depois que o código sair do `while` é o valor convergido para MTOW.

Algorithm 1 Iteração do MTOW

1: **while** $\Delta > 100$ **do**

2: Use o módulo de aerodinâmica para calcular $C_{D0,cruise}$ e K_{cruise} para a condição de cruzeiro usando $W_{0,guess}$ e as seguintes condições:

```
Mach = Mach_cruise
altitude = altitude_cruise
n_engines_failed = 0
flap_def = 0.0
max_flap_def = LD_flap_def
slat_def = 0.0
max_slat_def = LD_slat_def
lg_down = 0
h_ground = 0
```

3: Use o módulo de aerodinâmica para calcular $C_{D0,altcruise}$ e $K_{altcruise}$ para a condição de cruzeiro alternativo usando $W_{0,guess}$ e as seguintes condições:

```
Mach = Mach_altcruise
altitude = altitude_altcruise
n_engines_failed = 0
flap_def = 0.0
max_flap_def = LD_flap_def
slat_def = 0.0
max_slat_def = LD_slat_def
lg_down = 0
h_ground = 0
```

4: Use o módulo de peso vazio para calcular W_e usando $W_{0,guess}$

5: Use o módulo de peso de combustível para calcular W_f usando $W_{0,guess}$

6: $W_0 = W_{payload} + W_{crew} + W_f + W_e$ ▷ Recalcule o MTOW

7: $\Delta = W_0 - W_{0,guess}$ ▷ Calcule a diferença entre o valor estimado e o calculado

8: $W_{0,guess} = W_0$ ▷ Estabeleça o valor calculado como a nova estimativa