Imperial College
London

Supporting Document: Model Documentation

Imperial College London

Department of Aeronautics

Future Pathways for eVTOLs: A Design Optimization Perspective

Author:
Johannes Janning
CID: 02518943

Abstract

This document summarizes the core functions and interrelationships of the eVTOL Technical-Economical Operations (TEO) Framework. An interactive optimization framework for future eVTOL Operations Planning, Economcical & Environmental Analysis and Technology refinement. Document Version: v010924

Contents

List of Figures

List of Tables

Glossary

$Limit_{\mathbf{certification}}$ Limiting Certification Mass (kg). 7

$M_{\mathbf{battery}}$ Battery Mass (kg). 7

$M_{\mathbf{lug}}$ Single Luggage Mass (kg). 8

$M_{\mathbf{pax}}$ Passenger Mass (kg). 8

M_{empty} Aircraft Empty Mass (kg). 7

$M_{payload}$ Payload Mass (kg). 7

$n_{\mathbf{seats}}$ Number of seats available (-). 8

LF Load Factor (-). 8

MTOM Maximum Take-Off Mass (kg). 7

TAS True Airspeed. 2

1 Mission Profile & Operation Parameters

Assumptions and Limitations:

- The simulation assumes a zero-credit descent, the cruise phase proceeds until overhead of the destination where the reserve would begin or the hover landing begins.

The flight profile in this framework accounts for a standard eVTOL mission for urban air mobility [1] and regional air mobility. Fig. 1 illustrates the segments of the eVTOL mission, Table 1 is describing each segment.

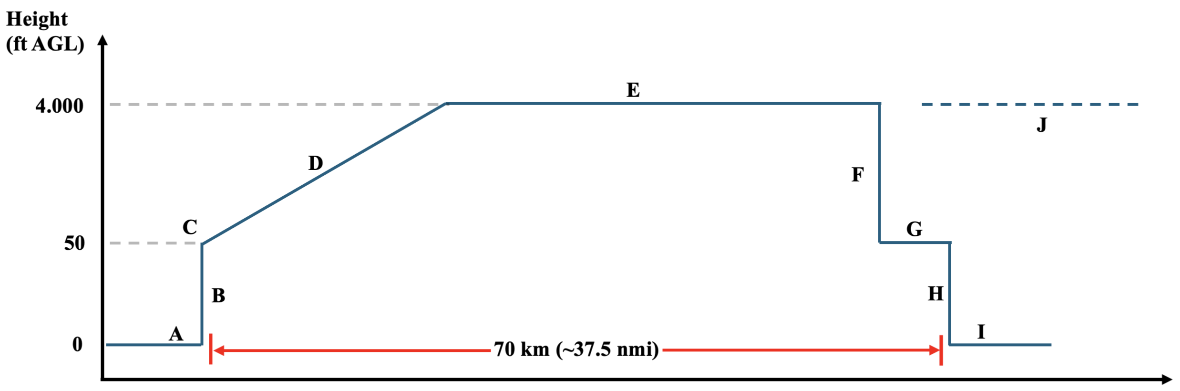


Figure 1: Sizing mission profile.

Segment	Description
A	Take-off
B	Vertical climb to 50ft (30s hover)
C	Transition
D	Climb
E	Cruise at 4.000ft
F	No credit descent
G	Transition
H	Vertical descent from 50ft (30s hover)
I	Landing
J	30 min reserve

Table 1: Sizing mission segments.

1.1 Operating Speeds

Assumptions and Limitations:

- The optimization framework presented does not consider wind.
- However, during climb and cruise phase, wind can be integrated by adding the average tailwind to the speed and subtracting the average headwind from the airspeed.
- The required climb speed V_{climb} and cruise speed V_{cruise} are computed according to the standard lift equation, depending on the aircraft mass $MTOM$, air density ρ, lift coefficient C_L and wing area S, see eq. 76 and 71 respectively.

True Airspeed (TAS) is the calibrated airspeed (CAS) corrected for altitude and non-standard temperature, representing the speed of the aircraft relative to the surrounding airmass [2]. TAS is used to calculate the power required for each flight phase. Within this model, the decisive speeds are the horizontal cruise speed V_{cruise} and the horizontal climb speed V_{climb} as described in eq. 76 and 71 respectively. The vertical climb speed, named rate of climb (ROC), which is used for energy requirement modeling during climb phase, is computed below, using α_{cimb} (see Chapter 3), assuming zero wind:

$$ROC = tan(\alpha_{climb}) \cdot V_{climb} \tag{1}$$

As consideration: in low energy segments, namely reserve or holding flight, the loiter speed can be used as defined by Ref. [1].

$$V_{loiter} = \left(\frac{1}{3}\right)^{\frac{1}{4}} V_{cruise} \,[\text{kts}] \tag{2}$$

When accounting for wind, the TAS is summed up with wind speed (head/tail wind), resulting in aircraft ground speed (GS). In this model we can assume adjustable direct head- or tailwind in cruise phase as well as climb phase. If taken into account, the headwind on a route can be considered as a margin factor on the speed over ground (i.e. relevant for the time calculation and the aerodynamic model).

1.2 Altitude Settings

Assumptions and Limitations:

- The initial and final mission altitude $h_{takeoff-landing}$ above mean sea level is assumed to be 0m.
- Final hover height h_{hover} is assumed to be 50ft (15.25m) above ground level, according Ref. [3].
- Cruise altitude is assumed for 1,000ft (304.7m), covering EASA minimum safe altitude above congested and rural areas.
- To simulate other altitudes, these can simply be adjusted in the framework. The change in air density with altitude is taken into account via the atmospheric model.

Within this framework, there are three altitude to consider:

$$h_{hover} = \text{hover height [ft]}\, AGL \tag{3}$$

$$h_{cruise} = \text{cruise height}\,[\text{ft}]\,AGL \tag{4}$$

$$h_{takeoff-landing} = \text{initial altitude}\,[\text{ft}]\,AMSL \tag{5}$$

Eq. 3 is the initial hover height above ground level at which transition occurs, it is assumed to be 100 ft AGL according Ref. [3]. Eq. 4 illustrates the cruise altitude above ground level. Eq. 5 is referred to the initial take-off and landing altitude, above mean sea level. These height setting does influence the air density within the atmospheric model, see Chapter 3.3.

1.3 Distances

Assumptions and Limitations:

- The presented eVTOL optimization uses a constant circuitry ratio Ω of 1 due to simplicity and improved comparability.
- For further analysis, the author's determined circuitry ratios for eVTOL from Table 2 can be used, which are based on flight planning taking into account the airspace and flight procedures.
- The circuitry ratios Ω of the means of transport used in the transport type comparison are shown in Table 3 and are widely based on own computations.

The direct air distance D_{direct} from a starting point to end point is the only distance to externally set within this framework. D_{direct} can be influenced by the circuitry ratio, which is developed separately. By selecting the desired operational mode, the model will adapt the setting for the trip ground distance in eq. 6

$$D_{tripground} = D_{direct} * \Omega\,[\text{km}] \tag{6}$$

where Ω is the circuitry ratio, defined by:

$$\Omega = f(D_{direct}, OperatingMode) \tag{7}$$

Ω varies, depending on IFR, VFR or no selected specific type of operation, for which it is 1.

	<35km	**<120km**	**>120km**
VFR uncontrolled	1,09	1,04	1,04
VFR controlled	1,40	1,05	1,05
IFR	2,98	1,64	1,05
Direct Air Distance	1,00	1,00	1,00

Table 2: Circuity Ratios for eVTOL Operations

The respected travel segment times and distances within the trip are computed with the equations below (using SI-units):

$$t_{climb} = \frac{h_{cruise} - h_{hover}}{ROC} \tag{8}$$

$$D_{climb} = V_{climb} * t_{climb}\,[\text{km}] \tag{9}$$

and

$$D_{cruise} = D_{tripground} - D_{climb} \tag{10}$$

$$t_{cruise} = \frac{D_{cruise}}{V_{cruise}} \tag{11}$$

The total time for one trip is then calculated by adding up:

$$t_{trip} = t_{hover_{TO}} + t_{climb} + t_{cruise} + t_{hover_{LDG}} + t_{holding} \tag{12}$$

where it can be assumed that $t_hover = t_{hover_{TO}} + t_{hover_{LDG}} = 60 seconds$, and $t_{holding}$ be set as required.

	Distance	**Ratio**	**Source**
Cars	up to 180km	1.30	computed
Cars	above 180km	1.20	computed
Train	all	1.20	computed
Airplane	all	1.05	Ref. [4]
Bicycle	all	1.28	computed
Bus	up to 100km	1.60	computed
Bus	above 100km	1.25	computed

Table 3: Circuity Ratios for Different Modes of Transport

1.4 Operations Model

Assumptions and Limitations:

- The following operational assumptions are made within the provided simulation.
- The assumptions can be adjusted according individual operational aspects.
- The provided operational framework evaluates the implementation of battery-swapping technology. The baseline scenario provides, build-in batteries that need stationary recharge before flight.

- The eVTOL operates within a daily operation window, defined by a fixed value of T_D (hours).
- Within this operation window the eVTOL can perform flights, characterized by travel time t_{trip} and battery depth-of-discharge (DOD), defined in chapter (4.3).
- The time between each flight is defined as the turnaround-time T_T. This time accounts for battery recharging as well as passenger/freight swap, but within the model only battery recharging is considered.
- Time for battery recharge is defined by the charging rate $C_{rate_{charge}}$, defined in chapter (4.3).

$$T_T = \frac{1}{C_{rate_{charge}}} \cdot DoD = \frac{1}{C_{rate_{charge}}} \cdot \frac{E_{trip}}{E_{Battery}} \tag{13}$$

with:

$$DoD = \frac{E_{trip}}{E_{Battery}} \tag{14}$$

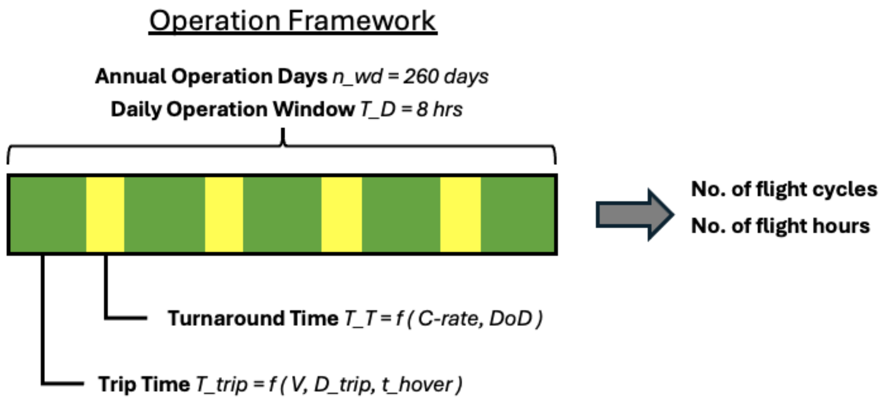
The Total Cycle Time Factor (CT), measures the efficiency of transportation operations. It reflects the ratio of non-productive turnaround time to productive trip time, with the "+1" accounting for the complete cycle. A lower value indicates higher operational efficiency by minimizing turnaround time relative to trip time. It is defined by:

$$CT = \frac{T_T}{t_{trip}} + 1 \tag{15}$$

Fig. (2) depicts an operational framework for eVTOL vehicles, based on Uber Elevate requirements, Ref. [5]. It includes:

- **Annual Operation Days (n_{wd})**: 260 days/year.
- **Daily Operation Window (T_D)**: 8 hours/day.
- **Turnaround Time (T_T)**: Function of C-rate and Depth of Discharge (DoD).
- **Trip Time (T_{trip})**: Depends on vehicle speed (V), trip distance (D_{trip}), and hover time (t_{hover}).

The framework calculates the number of flight cycles and total flight hours within these constraints. Operational parameters like annual-trip-time ratio, annual flight hours, daily flight hours and flight cycles are defined below:



Operation Framework
Annual Operation Days n_wd = 260 days
Daily Operation Window T_D = 8 hrs
No. of flight cycles
No. of flight hours
Turnaround Time T_T = f (C-rate, DoD)
Trip Time T_trip = f (V, D_trip, t_hover)

Figure 2: Operations Window.

- λ_{trip} = **annual-trip-time ratio**, defined by equation 16 and 21:

$$\lambda_{trip} = \frac{t_{trip}}{FH_{annual}} \tag{16}$$

- FH_{annual} (hours) = annual operating flight hours of one eVTOL, defined by equation 17 and 20:

$$FH_{annual} = t_{trip} \cdot FC_{annual} = t_{trip} \cdot n_{WD} \cdot FC_{day} \tag{17}$$

- FC_{annual} (-) = annual operating flight cycles of one eVTOL, defined by:

$$FC_{annual} = n_{WD} \cdot FC_{day} = n_{WD} \cdot \frac{T_D}{t_{trip} \cdot CT} \tag{18}$$

- n_{WD} (-) = number of operating days per year, set to 260, Ref. [6].

- FC_{day} (-) = daily operating flight cycles of one eVTOL, defined by:

$$FC_{day} = \frac{T_D}{t_{trip} \cdot CT} \tag{19}$$

$$FH_{annual} = n_{WD} \cdot \frac{T_D}{CT} \tag{20}$$

and

$$\lambda_{trip} = \frac{1}{FC_{annual}} = \frac{t_{trip} \cdot CT}{n_{WD} \cdot T_D} \tag{21}$$

2 Aircraft Mass Budget

2.1 Maximum Take-Off Mass

Assumptions and Limitations:

- Due to the preliminary design nature of the optimisation, MTOM should be treated as a first estimate.
- The proposed design optimization allows a deviation by 2% of eq. 22.

The total mass of an eVTOL aircraft can be generally described by (expressed in kg):

$$MTOM = M_{payload} + M_{empty} + M_{\text{battery}} \leq Limit_{\text{certification}} \tag{22}$$

with $MTOM$ being the maximum take-off mass, $M_{payload}$ the payload mass, M_{empty} the empty mass, and $M_{battery}$ the battery mass. According EASA's latest certification specifications for eVTOL, Ref. [3], the maximum certifiable weight is limited to $Limit_{certification} = 5.700kg$.

2.2 Battery Mass

Assumptions and Limitations:

- Battery Mass accounts for design mission energy requirements, as sum of reserve and trip energy. Reserve energy can be adjusted by the user by selecting reserve time, which is assumed to be 30min in cruise power.
- 20% of unusable battery capacity in terms of state of charge ceiling and floor are considered (see Chapter 4.3).
- The end-of-life battery status is considered, defined by 80% of begin-of-life condition (see Chapter 4.3).

Historically, $M_{battery}$ would be considered as part of M_{empty}, but for enormous impact of $M_{battery}$ on MTOM, it is considered separately. $M_{battery}$ can be calculated by eq. 23, considering the design mission energy needs and battery energy desnity ρ_{bat} in Wh/kg. The factor of 0.64 accounts for unusable battery energy (see Chapter 4.3).

$$M_{battery} = \frac{E_{useable}}{e_{usable}} = \frac{E_{trip} + E_{res}}{0.64 \cdot \rho_{bat}} \text{ [kg]} \tag{23}$$

2.3 Payload

Assumptions and Limitations:

- Payload is defined by load factor, number of passengers and their average mass.
- Within this framework a constant payload and a load factor of 1 is considered, according equation 25.

$M_{payload}$ should count for an average passenger weight M_{pax} and luggage weight per passenger M_{lug}. The mean Passenger and Luggage weights for commercial air transportation are assumed M_{pax}=82.2kg and M_{lug}=16kg [7]. n_{seats} is the maximum seat capacity of the aircraft. LF is the load factor of the aircraft.

$$M_{payload} = (M_{\text{pax}} + M_{\text{lug}}) \cdot N_{\text{seats}} \cdot LF \,[\text{kg}] \tag{24}$$

In a fully loaded 4-seater eVTOL this results in:

$$M_{payload} = (82.2 + 16) \cdot 4 \cdot 1 \,[\text{kg}] = 392.8 \,[\text{kg}] \tag{25}$$

2.4 Empty Mass Method 1

Assumptions and Limitations:

- This method is used within the presented optimization.
- Statistical methods highly depend on used data set for model fit. The provided methods are based on general aviation aircraft.
- The provided equations are highly unit sensitive. Use the provided conversion table to transfer metric units into imperial units, insert the imperial units into the models and re-transfer the results from pounds to kilogram.
- Wihtin this framework, statistical mass estimations according Raymer [8] and Nicolai [9] are averaged.

The empty mass in this method is defined as:

$$M_{empty} = M_{wing} + M_{rotor} + M_{motor} + M_{fuselage} + M_{systems} + M_{furnish} + M_{crew} \tag{26}$$

Where crew mass M_{crew} is based on average masses for single pilot operation [7]. Within this method, the values in Table 4 are used for unit conversion.

Unit	**Metric**	**Imperial**
Weight	1 kg	2.20462 lb
Length	1 m	3.28084 ft
Area	1 m^2	10.7639 ft^2
Speed	1 m/s	1.94384 kt

Table 4: Conversions from Metric to Imperial Units

2.4.1 Wing Mass

Assumptions and Limitations:

- Within optimization model, rectangular wing with NACA2412 airfoil is assumed, taper ratio λ is assumed 1 and sweep angle $\Lambda_{c/4}$ to be 0°.
- Airfoil data for NACA0012 is provided in Chapter 3 and can be adjusted in the provided optimization code.
- Thickness to chord t/c ratio is assumed for 0.12 (NACA2412).
- Maximum level airspeed V_H is assumed to equal V_{cruise}.
- The calculation of the wing mass is based solely on statistical mass estimation, based on historical aircraft data from general aviation. Detailed structural and geometric characteristics of the wings are NOT taken into account. The integration of an FEM model, for example, for a more accurate calculation of the wing mass, taking into account more precise wing design parameters, has the potential to significantly complement the proposed model in future work.

Raymer:

$$M_{W_R} = 0.036 \cdot S_W^{0.758} W_{FW}^{-0.0035} \left(\frac{AR_W}{\cos^2 \Lambda_{C/4}}\right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100 \cdot t/c}{\cos \Lambda_{C/4}}\right)^{-0.3} (n_z W_O)^{0.49} \tag{27}$$

Nicolai:

$$M_{W_N} = 96.948 \cdot \left(\frac{n_z W_O}{10^5}\right)^{0.65} \left(\frac{AR_W}{\cos^2 \Lambda_{C/4}}\right)^{0.57} \left(\frac{S_W}{100}\right)^{0.61} \left(\frac{1+\lambda}{2(t/c)}\right)^{0.36} \left(\sqrt{1 + \frac{V_H}{500}}\right)^{0.993} \tag{28}$$

In the given equations:

- M_W is the predicted weight of the wing in pounds-force (lbf).
- S_W is the trapezoidal wing area in square feet (ft^2).
- W_{FW} is the weight of fuel in the wing in pounds (lbs) (if $W_{FW} = 0$, then let $W_{FW}^{0.0035} = 1$).
- AR denotes the Aspect Ratio of the wing, HT, or VT, as per the appropriate subscripts.
- $\Lambda_{C/4}$ is the wing sweep at 25% Mean Geometric Chord (MGC).
- q represents the dynamic pressure at cruise.
- λ is the wing taper ratio.
- t/c refers to the wing thickness to chord ratio.
- n_z is the ultimate load factor.
- W_0 is the design gross weight in pounds (lbs).
- V_H is the maximum level airspeed at Sea Level (S-L) in Knots Equivalent Airspeed (KEAS).

Resulting in:

$$M_{wing} = \frac{M_{W_R} + M_{W_N}}{2} \tag{29}$$

2.4.2 Rotor Mass

The rotor weight model from [10], which deals with UAM, was adopted and adapted for passenger transporting EVTOL using a correction factor. The average weight and radius of a series of MT propellers from table 5 was used as a reference to identify a matchpoint where the radius of 1.1m hits 18.0kg. In addition the model was damped by adjusting the potency of the disproportionality to moderate the exponential increase in rotor weight with rotor size.

$$M_R = k_{rotor} \cdot \left[n_h \cdot \left(0.7484 \cdot R_{\text{hover}}^{1.2} - 0.0403 \cdot R_{\text{hover}}\right) + n_c \cdot \left(0.7484 \cdot R_{\text{cruise}}^{1.2} - 0.0403 \cdot R_{\text{cruise}}\right)\right] \quad (30)$$

where n_h and n_c are the number of rotors in the hover configuration and cruise configuration, respectively. During climb, the cruise propeller is used. R_i represents the radius of each propeller, with i indicating either the hover or cruise configuration.

Type	**MT-11**	**MT-15**	**MTV-21**	**MTV-17**	**MTV-20**	**Average**
D [cm]	190	260	203	190	210	
r [cm]	95	130	101.5	95	105	1.1
M [kg]	16	25	12	16	21	18
Source	[11]	[12]	[13]	[14]	[15]	

Table 5: Technical Data for MT Propellers

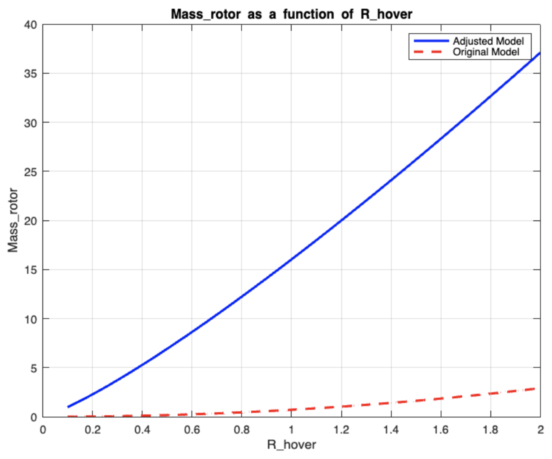


Figure 3: Rotor mass model.

2.4.3 Landing Gear Mass

Assumptions and Limitations:

- The provided equations are highly unit sensitive. Use the provided conversion table to transfer metric units into imperial units, insert the imperial units into the models and re-transfer the results from pounds to kilogram.

The Landing gear mass is computed by averaging statistical models. Raymer provides to equations, one for main landing gear and one for nose landing gear. Nicolai provides a model for the entire landing gear. The landing gear weight is driven by total aircraft mass $MTOM$ and length of the gear struts L_m and L_n, summarized as L_i. L_i is defined by the cruise propeller radius $R_{prop_{cruise}}$ and the minimum propeller clearance of 7 inches (0.1778m) as required by EASA, Ref. [16], and FAA, Ref. [17].

Raymer:

$$W_{MLG_R} = 0.095 \cdot (n_l \cdot W_l)^{0.768} \left(\frac{L_m}{12}\right)^{0.409} \tag{31}$$

$$M_{NLG_R} = 0.125 \cdot (n_l W_l)^{0.566} \left(\frac{L_n}{12}\right)^{0.845} \tag{32}$$

where:

- W_{MLG} = predicted weight of the main landing gear in lbf
- W_{NLG} = predicted weight of the nose landing gear in lbf
- n_l = ultimate landing load factor (set to $1.5 \cdot 2.5 = 3.75$, with 2.5 as maximum load factor 1.5 as safety factor).
- L_m = length of the main landing gear strut in inches
- W_l = design landing weight in lbf
- L_n = length of the nose landing gear strut in inches

Nicolai:

$$M_{MNLG_N} = 0.054 \cdot (n_l W_l)^{0.684} \left(\frac{L_m}{12}\right)^{0.601} \tag{33}$$

where:

- n_l = ultimate landing load factor
- W_l = design landing weight in lbf
- L_m = length of the main landing gear strut in inches
- W_{MNLG} = predicted weight of the entire landing gear in lbf

Resulting in:

$$M_{gear} = \frac{M_{MLG_R} + M_{NLG_R} + M_{MNLG_N}}{2} \tag{34}$$

2.4.4 Motor Mass

Assumptions and Limitations:

- The motor mass model, based on Ref. [18], is valid for the power range from 10 kW to 260 kW per motor.

The model is based on Ref. [18] and solely depends on the number of motors, which is set to the number of rotors, aligning with the power architecture of the eVTOL shown in Fig. (4).

$$M_{motor} = n_h \cdot 0.6756 \left(\frac{p_{\text{hover}}}{n_{p_{\text{hover}}} \cdot 745.7}\right)^{0.783} + n_c \cdot 0.6756 \left(\frac{p_{\text{climb}}}{n_{p_{\text{horizontal}}} \cdot 745.7}\right)^{0.783} \tag{35}$$

where P is the power, which can be input directly in watts, and the output is the weight in kilograms (kg). n_h and n_c are the number of rotors in the hover configuration and cruise configuration, respectively.

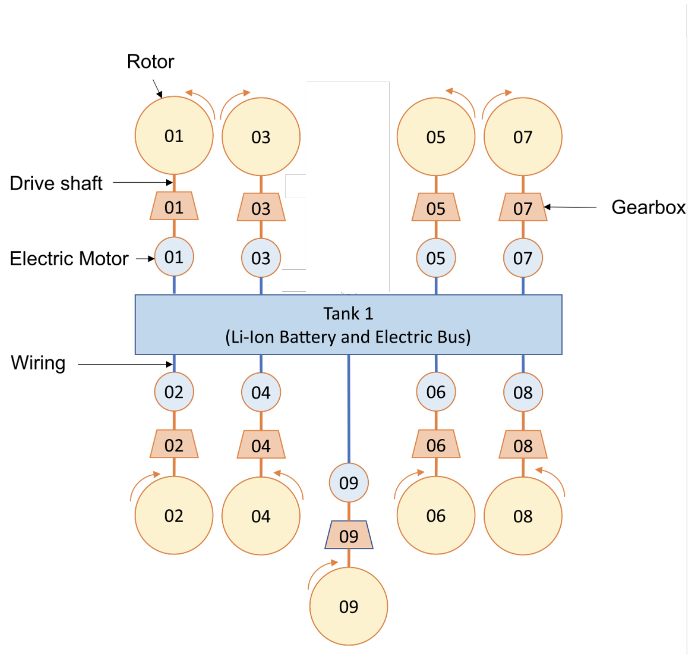


Figure 4: Motor architecture concept, from Ref. [19].

2.4.5 Fuselage Mass

Assumptions and Limitations:

- The provided equations are highly unit sensitive. Use the provided conversion table to transfer metric units into imperial units, insert the imperial units into the models and re-transfer the results from pounds to kilogram.
- The calculation of the fuselage mass is based on statistical mass estimation methods. The structural and geometric properties of the stringers, frames and skins are NOT taken into account. Consideration of the structural calculation may contribute to a significant future extension of the model.

The fuselage mass accounts for the structural mass of the frames, stringer, and skin of the main fuselage.

Raymer:

$$W_{\text{FUS}} = 0.052 \cdot S_{\text{FUS}}^{0.086} \cdot (n_z W_0)^{0.177} l_{\text{HT}}^{-0.051} \left(\frac{l_{\text{FS}}}{d_{\text{FS}}}\right)^{-0.072} q^{0.241} + 11.9 \cdot (V_P \Delta P)^{0.271} \tag{36}$$

Nicolai:

$$W_{\text{FUS}} = 200 \left[\left(\frac{n_z W_0}{10^5}\right)^{0.286} \left(\frac{l_F}{10}\right)^{0.857} \left(\frac{w_F + d_F}{10}\right) \left(\frac{V_H}{100}\right)^{0.338} \right]^{1.1} \tag{37}$$

where:

- W_{FUS} = predicted weight of the fuselage in lbf

- S_{FUS} = fuselage wetted area in ft^2
- l_{FS} = length of fuselage structure (forward bulkhead to aft frame) in ft
- d_{FS} = depth of fuselage structure in ft
- V_P = volume of pressurized cabin section in ft^3
- ΔP = cabin pressure differential, in psi (typically 8 psi)
- l_F = fuselage length in ft
- w_F = fuselage max width in ft
- d_F = fuselage max depth in ft
- l_{HT} = length of fuselage structure (not explicitly listed in the overview, but assumed from context)
- n_z = ultimate load factor
- W_0 = design gross weight in lbf
- q = dynamic pressure at cruise
- V_H = maximum level airspeed at Sea Level (S-L) in Knots Equivalent Airspeed (KEAS)

2.4.6 Systems Mass

Assumptions and Limitations:

- The provided equations are highly unit sensitive. Use the provided conversion table to transfer metric units into imperial units, insert the imperial units into the models and re-transfer the results from pounds to kilogram.

The systems mass accounts for the flight control system and avionics. We assume it also covers air conditioning, lights and other non-flying electronics.

Raymer:

$$M_{system_R} = 0.053 \cdot l_{\text{FS}}^{1.536} \cdot b^{0.371} \cdot \left(n_z W_0 \cdot 10^{-4}\right)^{0.80} \tag{38}$$

Nicolai:

$$M_{system_N} = 1.08 W_0^{0.7} \tag{39}$$

where:

- W_{CTRL} = predicted weight of the flight control system in lbf
- l_{FS} = length of fuselage structure in ft
- b = wingspan in ft
- n_z = ultimate load factor
- W_0 = design gross weight in lbf

Resulting in:

$$M_{system} = \frac{M_{system_R} + M_{system_N}}{2} \tag{40}$$

2.4.7 Furnish Mass

Assumptions and Limitations:

- The provided equations are highly unit sensitive. Use the provided conversion table to transfer metric units into imperial units, insert the imperial units into the models and re-transfer the results from pounds to kilogram.

The furnish mass accounts for interior of the eVTOL, like seating and other travel-experience related equipment like entertaiment.

Raymer:

$$M_{furn_R} = 0.0582 W_0 - 65 \tag{41}$$

Nicolai:

$$M_{furn_N} = 34.5 N_{\text{CREW}} q_H^{0.25} \tag{42}$$

where:

- W_{FURN} = predicted weight of furnishings in lbf
- W_0 = design gross weight (maximum take-off mass, MTOM) in lbf
- N_{CREW} = number of crew
- q_H = dynamic pressure at max level airspeed, in lbf/ft^2

Resulting in:

$$M_{furnish} = \frac{M_{furn_R} + M_{furn_N}}{2} \tag{43}$$

2.5 Empty Mass Method 2

Assumptions and Limitations:

- This method is identical to Empty Mass Method 1, despite only the Raymer or Nicolai Method is used, so no average is taken.
- This method is prone to under- and over estimation of component mass, as it highly dependent on used data sets for statistical method fit.

2.6 Empty Mass Method 3

Assumptions and Limitations:

- This method is based on models used for the calculation of UAVs and on estimates by the author, but provides a simpler but possibly less accurate estimate of the eVTOL empty weight.

The empty weight in this model is defined as:

$$M_{empty} = M_{wing} + M_{rotor} + M_{motor} + M_{other} \tag{44}$$

The wing (eq. 45) and rotor (eq. 46) mass models are adapted from RR Martins [10]. The motor mass model (eq. **??**) is the same in method 1. S being the wing reference area in m^2, N_P the number of propellers and R the rotor diameter in m for hover and cruise configuration:

$$M_{wing} = -0.0802 + 2.2854 \cdot S \tag{45}$$

$$M_{rotor} = \sum_i N_{P_i} \cdot (0.7484 \cdot R_i^2 - 0.0403 \cdot R_i) \tag{46}$$

M_{other} is assumed fix, accounting for fuselage, avionics, wiring, interior, empennage and painting. It is estimated by subtracting the wing, rotor, and motor weight using the above models, from the Tecnam P2006T's total empty weight of 819kg Ref. [20], which is a similiar in-size aircraft to future eVTOL. For the Tecnam P2006T following is assumed Ref. [20]: s = 11.4m , $A_{wing} = 14.8m^2$, 2 propellers, $R_{prop} = 0.9m$. This results in a fixed other empty weight of:

$$M_{other} = 668kg \tag{47}$$

which aligns with estimations from Ref. [21] for eVTOL empty mass.

3 Aerodynamic Model

In this model we use a simplified aerodynamic lift and drag model to calculate lift coefficient C_L and drag coefficient C_D of the eVTOL wing, based on Ref. [22] and Ref. [23]. The airfoil data is taken from Ref. [24, 25] at $Re = 200,000$.

3.1 Lift Model

Within this framework only the pre-stall aerodynamics are considered. A linear pre-stall lift curve is assumed with a stall angle of attack at 15°. According [23], the finite-wing-lift-curve slope α_{wing} is described by:

$$\alpha_{wing} = \frac{\alpha_{airfoil}}{1 + \frac{\alpha_{airfoil}}{\pi \cdot AR \cdot e}} \tag{48}$$

where AR is the wing aspect ratio, e is the oswald span efficiency, set to 0.8, and $\alpha_{airfoil}$ is the airfoil-lift-curve slope. During the development of this model there are two airfoil profiles in consideration. Based on Ref. [23] and in the interest of simplicity the symmetric NACA 0012 profile (0% camber, 12% max thickness at 30% chord, Ref. [23]) can be used. To simulate an airfoil with camber, the NACA 2412 (can be used as it provides more lift at lower angle of attacks compared to the NACA 0012 (2% max camber at 40% chord, 12% max thickness at 30% chord, Ref. [23]) making it more efficient in urban and regional air mobility. For detailed airfoil info see Appendix A. The finite-wing lift coefficient can be written as:

$$C_L = \frac{\alpha_{airfoil}}{1 + \frac{\alpha_{airfoil}}{\pi \cdot AR \cdot e}} \cdot \alpha + C_{L_0} = \alpha_{wing} \cdot \alpha + C_{L_0} \tag{49}$$

where C_{L_0} is the lift coefficient at zero angle of attack (equals 0 for NACA 0012) and α is the angle of attack. The slope of the lift coefficient of the airfoil NACA 0012 and NACA 2412 are computed with linear regression using the data from Table 12 and 13 in the Appendix. The airfoils can be described as followed:

$$\text{NACA0012} : C_L = 5.4225(1/rad) \cdot \alpha(rad) \tag{50}$$

$$\text{NACA2412} : C_L = 4.0923(1/rad) \cdot \alpha(rad) + 0.415 \tag{51}$$

This results in total lift, where ρ is the air density (modelled with the atmospheric model, see Chapter 3.3), V is the aircraft's true air speed, S is the wings reference area:

$$L = \frac{1}{2} \cdot \rho \cdot V^2 \cdot S \cdot C_L = \frac{1}{2} \cdot \rho \cdot V^2 \cdot S \cdot \left(\frac{\alpha_{airfoil}}{1 + \frac{\alpha_{airfoil}}{\pi \cdot AR \cdot e}} \cdot \alpha + C_{L_0} \right) \tag{52}$$

3.2 Drag Model

The model used is based on the lifting line theroy [23] and assumes finite-wing-drag coefficient as composition of induced drag C_{D_i} and parasite drag $C_{D_{min}}$:

$$C_{D_{min}} = C_{D_{min}} + C_{D_i} \tag{53}$$

with

$$C_{D_i} = \frac{C_L^2}{\pi \cdot AR \cdot e} \tag{54}$$

and $C_{D_{min}} = 0.0397$ based on the approach by Ref. [22] and the wind-tunnel experiments of Ref.[26], accounting for additional drag of the lifting rotors of a lift+cruise configuration. Resulting in following drag model:

$$C_{D_{min}} = 0.0397 + \frac{C_L^2}{\pi \cdot AR \cdot e} \tag{55}$$

and total drag:

$$D = \frac{1}{2} \cdot \rho \cdot V^2 \cdot S \cdot C_D = \frac{1}{2} \cdot \rho \cdot V^2 \cdot S \cdot \left(0.0397 + \frac{C_L^2}{\pi \cdot AR \cdot e}\right) \tag{56}$$

The resulting lift and drag coefficient models are shown in Fig. 5. Within the model, a constant angle of attack during climb $\alpha_{climb} = 8°$ and $\alpha_{climb} = 3°$ (subject to change).

3.3 Atmospheric Model

Based on [27], the model of the international standard atmosphere (ISA) is implemented in the aerodynamic model, influencing the air density at selected heights. The air density ρ is described as a function of height, where ρ_{MSL} is the standard air density at mean sea level, $\rho_{takeoff-landing}$ is the air density at the takeoff- and landing site altitude $h_{takeoff-laning}$, measured AMSL (above mean sea level), ρ_{cruise} is the air density at cruise altitude H_{cruise}, measured AGL (above ground level), such that $h_{cruise} + h_{takeoff-laning}$ give the cruise altitude AMSL. ρ_{climb} is the air density at average climb altitude and which is assumed as the mean of h_{cruise} plus $h_{takeoff-laning}$. The calculated densities are in $\frac{\text{kg}}{\text{m}^3}$.

$$\rho_{MSL} = 1.225 \left[\frac{\text{kg}}{\text{m}^3}\right] \tag{57}$$

and

$$\rho_{takeoff-landing} = \rho_{MSL} \left(1 - 22.558 \times 10^{-6} \times h_{takeoff-landing}\right)^{4.2559} \tag{58}$$

and

$$\rho_{cruise} = \rho_{MSL} \left(1 - 22.558 \times 10^{-6} \times (h_{cruise} + h_{takeoff-landing})\right)^{4.2559} \tag{59}$$

and

$$\rho_{Climb} = \rho_{MSL} \left(1 - 22.558 \times 10^{-6} \times (h_{cruise}/2 + h_{takeoff-landing})\right)^{4.2559} \tag{60}$$

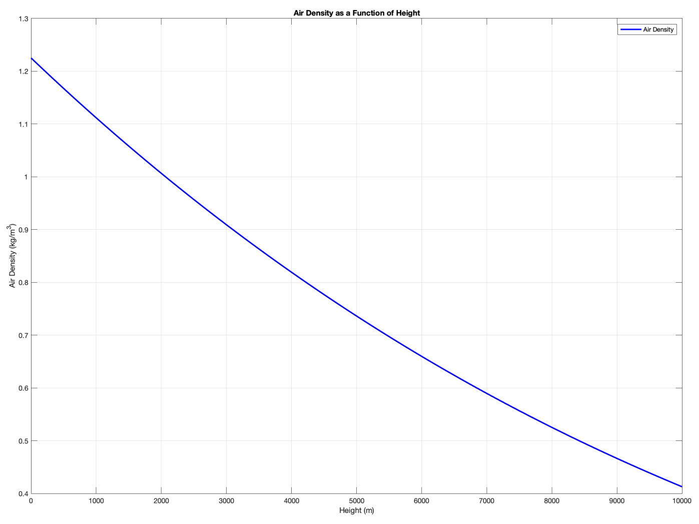
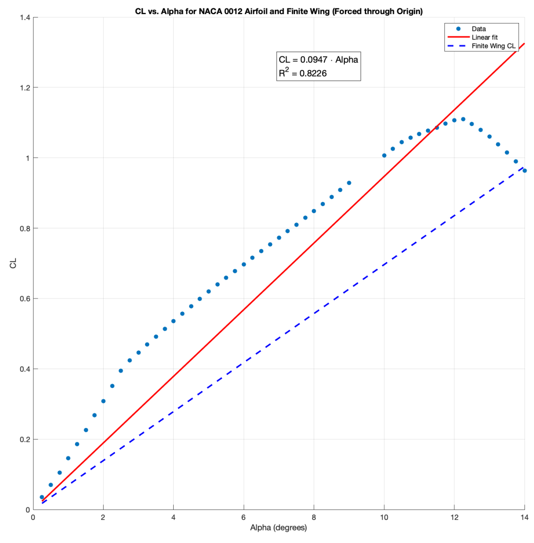
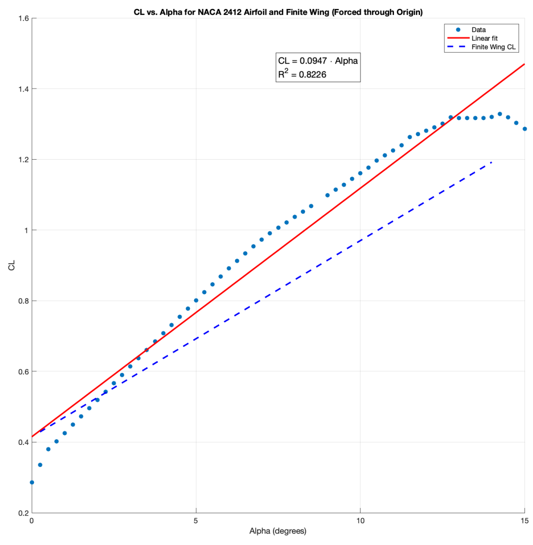


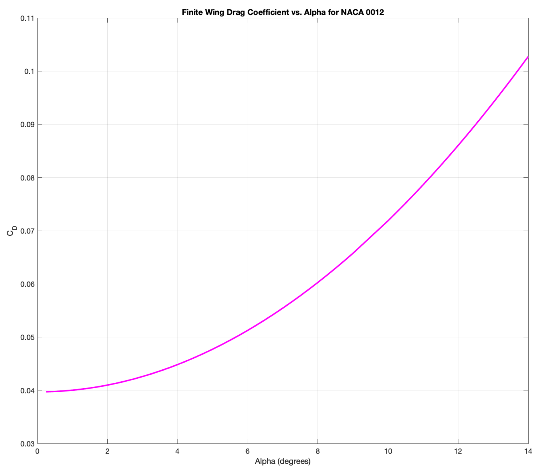
Figure 6: Air density as function of height.



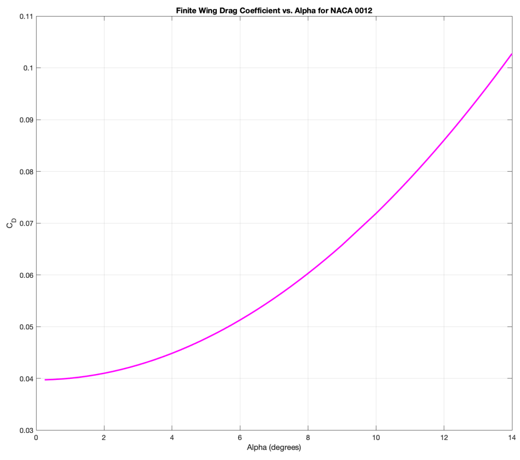
(a) Lift coefficient model for NACA0012.



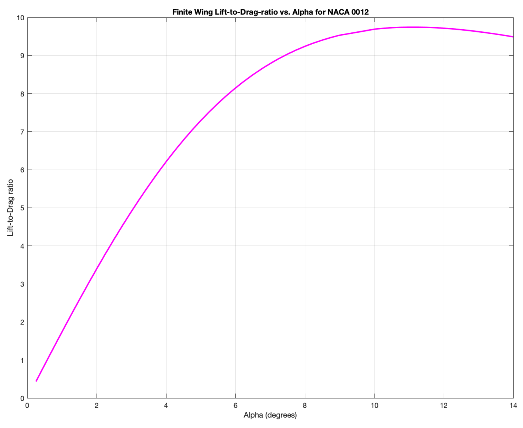
(b) Lift coefficient model for NACA2412.



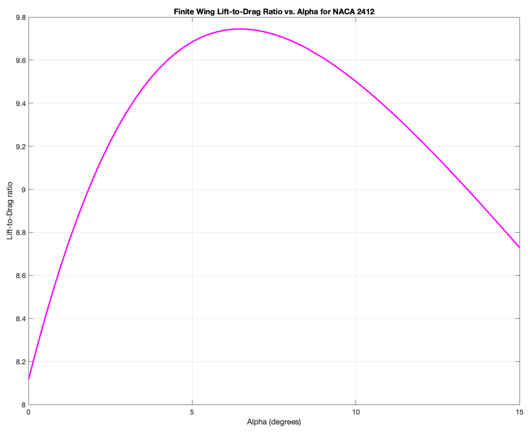
(c) Drag coefficient model for NACA0012.



(d) Drag coefficient model for NACA2412.



(e) Lift-to-drag ratio model for NACA0012.



(f) Lift-to-drag ratio model for NACA2412.

Figure 5: Airfoils regression models.

4 Power and Energy Model

4.1 Aircraft Efficiencies

Assumptions and Limitations:

- The provided framework assumes fix system efficiencies. These can be changed as required for customised simulations.

Item	Symbol	Value	Source
Electric Efficiency	η_e	0.9	[1]
Hover Propulsive Efficiency	η_{hp}	0.7	[6]
Propulsive Efficiency	η_p	0.85	[1]
Hover System Efficiency	$\eta_h = \eta_e \cdot \eta_{hp}$	0.63	[6]
Overall System Efficiency	$\eta_c = \eta_p \cdot \eta_e$	0.77	[6]

Table 6: System efficiencies.

4.2 Momentum Theory Power Model

Assumptions and Limitations:

- The power modelling within the provided framework is based on propeller momentum theory, using flight equilibrium relationships in hover, climb and cruise flight. For improved rotor dynamics accuracy, one should enhance this model using Blade Element Momentum Theory.
- Steady, unaccelerated flight (forces equilibrium) in each flight phase is assumed.
- For simulation, number of propeller in climb/cruise configuration is assumed to be 1. The number of vertical lift propeller is assumed to be 8.

4.2.1 Hover Power Required:

Power requirements in hover flight account for steady hover, where thrust and aircraft mass are balanced and power is needed to overcome propeller induced velocity, as illustrated in Fig. (7):

$$T_{required_{hover}} = MTOM \cdot g \tag{61}$$

resulting in a power requirement of:

$$P_{required_{hover}} = \frac{T_{total} \cdot v_i}{\eta_h} = \frac{MTOM \cdot g}{\eta_h} \cdot \sqrt{\frac{\sigma}{2 \cdot \rho}} \tag{62}$$

- v_i (m/s^2) = propeller induced velocity, defined by:

$$v_i = \sqrt{\frac{\sigma}{2 \cdot \rho}} \tag{63}$$

- ρ (kg/m^2) = air density in hover, defined by eq. 58.
- σ (N/m^2) = propeller disk loading, defined by:

$$\sigma = \frac{T_{prop}}{A_{prop}} = \frac{T_{prop}}{\pi \cdot R_{prop}^2} \tag{64}$$

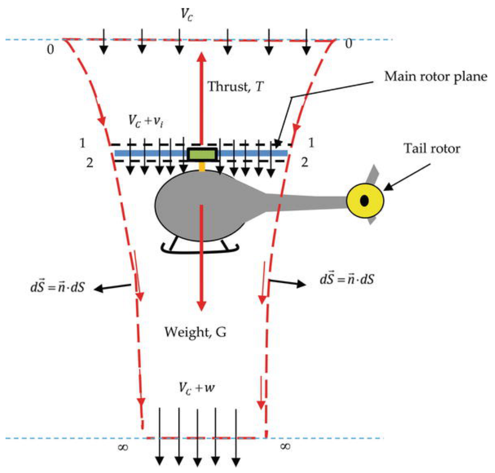


Figure 7: Forces in Hover, from Ref. [28].

4.2.2 Climb Power Required:

Power required in climb is defined by:

$$P_{climb} = P_{req_{flight}} + P_{ind_{prop}} = \frac{T_{req_{climb}} \cdot V_{climb}}{\eta_c} + \frac{T_{prop_{climb}} \cdot v_i}{\eta_c} \cdot n_{prop} \tag{65}$$

- $P_{req_{flight}}$ (W) = power required to maintain flight, defined by:

$$P_{req_{flight}} = \frac{T_{req_{climb}} \cdot V_{climb}}{\eta_c} \tag{66}$$

- $T_{req_{climb}}$ (N) = thrust required in climb, illustrated in Fig. (8) and defined by:

$$T_{required_{climb}} = D_{climb} + MTOM \cdot sin(\theta) \tag{67}$$

- D_{climb} (N) = aircraft drag in climb flight (eq. (56)), using V_{climb} (eq. (71)) and α_{climb}.
- MTOM (kg) = max. take-off mass (eq. (22)).
- θ (°) = climb angle, assumed to be equal to angle of attack in climb α_{climb}.
- $P_{ind_{porp}}$ (W) = propeller induced power required, defined by:

$$P_{ind_{prop}} = \frac{T_{prop_{climb}} \cdot v_i}{\eta_c} \cdot n_{prop} \tag{68}$$

- $T_{prop_{climb}}$ (N) = $T_{req_{climb}}/n_{prop}$ = thrust required per propeller.
- n_{prop} (-) = number of propeller.
- v_i (m/s) = propeller induced velocity, defined by equation (63

The horizontal speed during climb phase (Climb Speed), accounting for climb angle $\theta = \alpha_{climb}$, assuming zero wind, is defined by:

$$\text{Climb Condition: } MTOM \cdot g \cdot cos(\theta) = L = \frac{1}{2} \cdot \rho \cdot V_{climb}^2 \cdot S \cdot C_{L_{climb}} \tag{69}$$

Resulting in the trivial conditions during climb flight:

$$\text{Mass Condition: } MTOM = \frac{1}{2 \cdot g \cdot cos(\theta)} \cdot \rho \cdot V_{climb}^2 \cdot S \cdot C_{L_{climb}} \tag{70}$$

and,

$$\text{Climb Speed Condition: } V_{climb} = \sqrt{\frac{2 \cdot MTOM \cdot g \cdot cos(\theta)}{\rho \cdot S \cdot C_{L_{climb}}}} \tag{71}$$

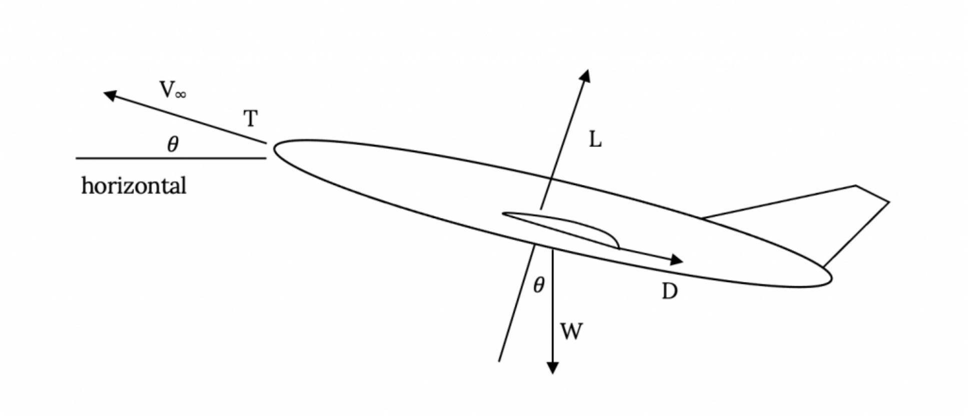


Figure 8: Forces in Climb, from Ref. [29].

4.2.3 Cruise Power Required:

Power required to maintain steady cruise flight is similar derived as cruise power, defined by:

$$P_{cruise} = P_{req_{flight}} + P_{ind_{prop}} = \frac{T_{req_{cruise}} \cdot V_{cruise}}{\eta_c} + \frac{T_{prop_{cruise}} \cdot v_i}{\eta_c} \cdot n_{prop} \tag{72}$$

- $T_{req_{cruise}}$ (N) = thrust required in cruise, illustrated in Fig. (9) and defined by:

$$T_{required_{cruise}} = D_{cruise} \tag{73}$$

- D_{climb} (N) = aircraft drag in cruise flight (eq. (56)), using V_{cruise} (eq. (76))

During horizontal, unaccelerated flight, the cruise speed is defined by:

$$\text{Cruise Condition: } MTOM \cdot g = L = \frac{1}{2} \cdot \rho \cdot V_{cruise}^2 \cdot S \cdot C_{L_{cruise}} \tag{74}$$

Resulting in the trivial conditions during cruise flight:

$$\text{Mass Condition: } MTOM = \frac{1}{2 \cdot g} \cdot \rho \cdot V_{cruise}^2 \cdot S \cdot C_{L_{cruise}} \tag{75}$$

and,

$$\text{Cruise Speed Condition: } V_{cruise} = \sqrt{\frac{2 \cdot MTOM \cdot g}{\rho \cdot S \cdot C_{L_{cruise}}}} \tag{76}$$

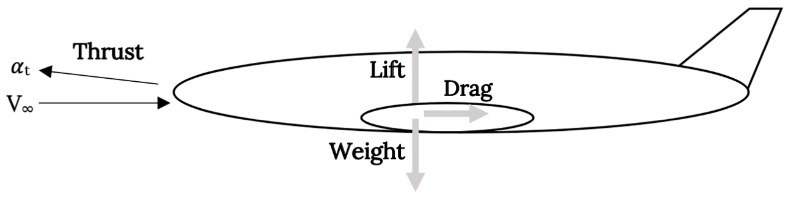


Figure 9: Static Force Balance in Straight and Level Flight, from Ref. [29].

4.3 Energy Requirements

The required energy (Wh) in each flight phase can be computed using the relationship:

$$E_i = \sum_i \mathrm{P}_i \cdot t_i \tag{77}$$

The specific battery energy can be calculated when dividing the energy by battery mass:

$$e_i = \frac{E_i}{M_{bat}} \tag{78}$$

- M_{bat} (kg) = battery mass, defined by eq. (23), with:

$$E_{useable} = E_{trip} + E_{reserve} \tag{79}$$

- E_{trip} (Wh) = sum of energy required in hover, climb and cruise, using eq. (77).
- $E_{reserve}$ (Wh) = energy required in reserve flight, accounting for 30 min for VFR and 45min for IFR flight with cruise power according FAA.

Within this model the end-of-life battery status is considered, which is set to 80% of BOL (begin-of-life) capacity:

$$e_{BAT_{EOL}} = 0.8 \cdot \rho_{bat} \tag{80}$$

The battery ceiling is set as 90% SOC and the floor as 10% SOC, making these portions unusable for trips [6]. Resulting in usable specific energy:

$$e_{\text{usable}} = 0.8 \cdot e_{BAT_{EOL}} = 0.64 \cdot \rho_{bat} = e_{trip_{design}} + e_{reserve} \tag{81}$$

or expressed as the battery specific energy budget for the trip:

$$e_{trip_{design}} = 0.64 \cdot \rho_{bat} - e_{reserve} \tag{82}$$

or

$$e_{trip} \leq 0.64 \cdot \rho_{bat} - e_{reserve} \tag{83}$$

$$1.5623 \cdot (e_{trip} + e_{reserve}) \leq \rho_{bat} \tag{84}$$

Resulting in total battery energy storage capacity of:

$$E_{Bat_{total}} = \rho_{bat} \cdot M_{bat} \tag{85}$$

4.4 Battery Degradation Model

Assumptions and Limitations:

- Within the provided framework, two methods are proposed to account for degrading battery energy storage capacity over operation time.
- **Method 1** encounters an self-developed empirical model to connect battery cycle life with depth-of-discharge (DoD), and C-rate during charge and discharge. This method is used within the provided simulations.
- **Method 2** is derived using a fixed degradation (Ref. [30]) factor at 1C discharge rate and is not accounting for DoD and charging rate.
- It must be mentioned that the simulation results of the battery consumption (numbers of the batteries) depend very much on the battery degradation model used. A reasonable enhancement of the model could be the integration of a more accurate and efficient battery degradation model in future work.

4.4.1 Method 1

This section provides a brief overview of the empirical model developed to estimate the cycle life of a battery as a function of Depth of Discharge (DoD), charging C-rate, and discharging C-rate. The model is based on data DoD and cycle-life data of Ref. [31] and [32], interpolated for accuracy, and adjusted for harsh environmental conditions such as those encountered in eVTOL operations.

Assumptions and Limitations:

- **Data Adjustment**: The provided cycle life data was reduced by a factor of 4 to simulate the more extreme conditions expected in eVTOL applications and account for degradation down to 80% of begin-of-life (BOL) capacity.
- **Interpolation**: A piecewise linear interpolation was used to fit the DoD data. This approach ensures that the model captures the observed trend accurately.The data was segmented at DoD = 0.8. Separate linear fits were applied to the segments below and above this threshold.
- **Empirical Formula**: The provided model incorporates the effects of DoD, charging C-rate, and discharging C-rate, as a result of interpolation and estimation. Further research required to capture more accurate models for battery degradation of eVTOL.

The empirical model for cycle life N_{cycles} as function of DoD is illustrated in Fig. (10) and defined as:

$$N_{\text{cycles}}(\text{DoD}) = \begin{cases} a_1 \cdot \text{DoD} + b_1, & \text{for DoD} \leq 0.8 \\ a_2 \cdot \text{DoD} + b_2, & \text{for DoD} > 0.8 \end{cases}$$

- The ‘pchip‘ (Piecewise Cubic Hermite Interpolating Polynomial) method is used for smooth interpolation of the given data points, and basic linear fitting (‘polyfit‘) is applied to derive the segmented linear model.

In this model, a DoD over 80% is avoided according to Ref. [6], therefore the model is reduced to:

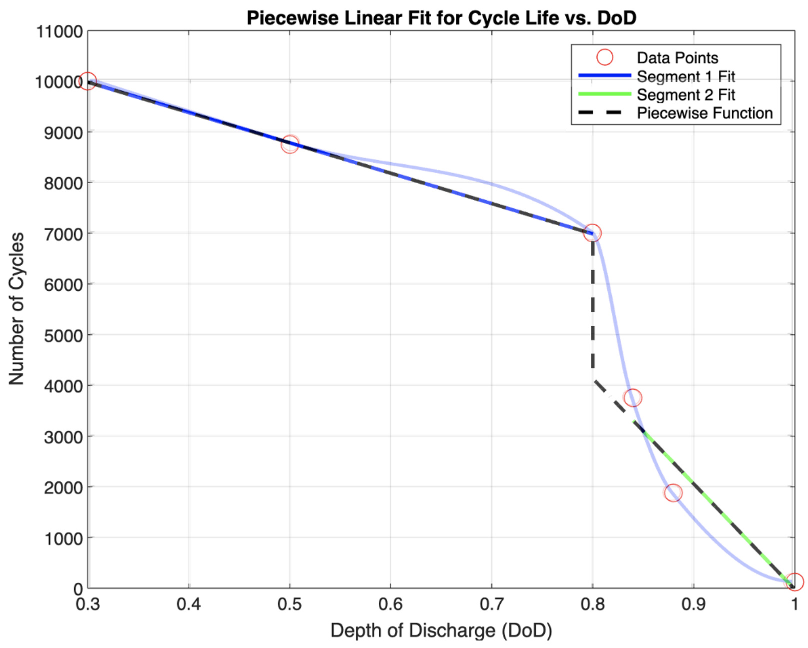


Figure 10: Interpolated Curve for Cycle Life vs. DoD

$$N_{cycles}(\text{DoD}) = a_1 \cdot DoD + b_1 \tag{86}$$

This is further adjusted by the following, assuming that the charge rates are linked to the cycles via a power ratio:

$$N_{\text{cycles}} = N_{\text{cycles}}(\text{DoD}) \cdot \left(\frac{1}{\text{C-rate}^{c}_{\text{discharge}}} \right) \cdot \left(\frac{0.5}{\text{C-rate}^{d}_{\text{charge}}} \right)$$

Where:

- a_1, b_1, a_2, b_2 are the fitted parameters for the piecewise linear segments.
- $c = 1.1$ and $d = 1.2$ are empirically estimated coefficients that account for the effects of discharging and charging rates, respectively.
- **Fitted Parameters**:

$$\text{For DoD} \leq 0.8 : a_1 = -5986.8421, b_1 = 11776.3158$$

$$\text{For DoD} > 0.8 : a_2 = -20793.2692, b_2 = 20769.2307$$

This results in the follwing used equation:

$$N_{\text{cycles}} = (-5986.8421 \cdot DoD + 11776.3158) \cdot \left(\frac{1}{\text{C-rate}^{1.1}_{\text{discharge}}} \right) \cdot \left(\frac{0.5}{\text{C-rate}^{1.2}_{\text{charge}}} \right) \tag{87}$$

With:

- **Discharge Rate:** $C - rate_{discharge} = \frac{C_{\text{rate\_hover}} \cdot t_{\text{hover}} + C_{\text{rate\_climb}} \cdot t_{\text{cl}} + C_{\text{rate\_cruise}} \cdot t_{\text{cr}}}{t_{\text{tot}}}$: Average C-rate across the different phases of flight.
- $C_{\text{rate\_hover}} = \frac{\text{Power}_{\text{hover}}}{E_{\text{battery}}}$: C-rate during hover, defined as the ratio of hover power to total battery energy.

- $C_{\text{rate\_climb}} = \frac{\text{Power}_{\text{climb}}}{E_{\text{battery}}}$: C-rate during climb, defined as the ratio of climb power to total battery energy.
- $C_{\text{rate\_cruise}} = \frac{\text{Power}_{\text{cruise}}}{E_{\text{battery}}}$: C-rate during cruise, defined as the ratio of cruise power to total battery energy.
- $E_{\text{battery}} = \rho_{\text{bat}} \cdot m_{\text{bat}}$: Total energy capacity of the battery in Wh.
- **Charge Rate:** $C - rate_{charge}$ is assumed to be a design variable within the framework, so set fixed and changed as required.
- **DoD:** $DOD = \frac{E_{trip}}{E_{Battery}}$, ratio of energy used to total stored energy.

The resulted battery cycle-life envelope is illustrated in Fig. (11) for a constant $C-rate_{charge}$ of 2C. The empirical model developed provides a first estimation of battery cycle life across different DoD and C-rate conditions, based on battery data. The piecewise linear approach effectively captures the sharp decrease in cycle life observed at higher DoD levels.

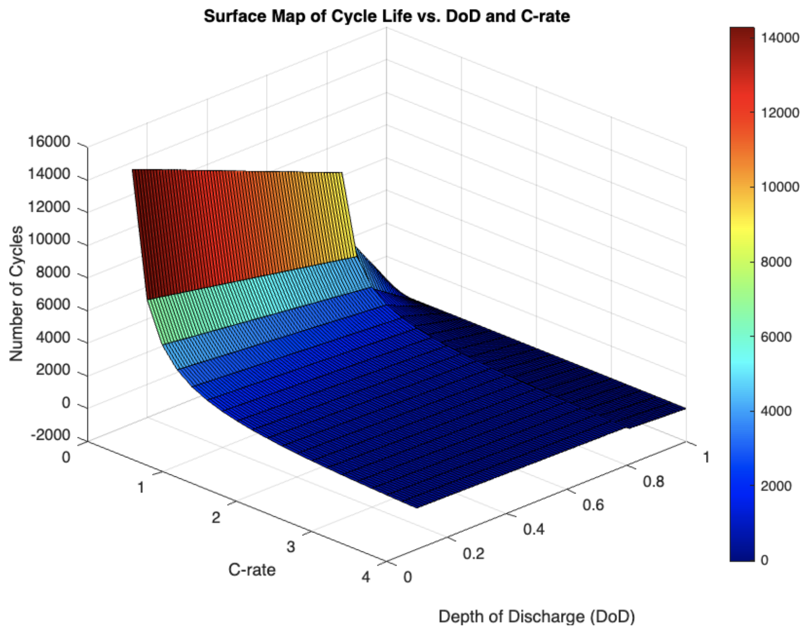


Figure 11: Piecewise Linear Fit for Cycle Life vs. DoD

4.4.2 Method 2

Assumptions and Limitations:

- This method delivers a simpler and static approach for battery cycle-life estimation, making the model less dynamic.

From Ref. [30] the average degradation of the battery per charging cycle is $Q_{cycle} = 0.048\%$. Assuming an allowed degradation of $D = 20\%$, the number of life cycles for the eVTOL is:

$$N_{cycles} = \frac{D}{Q} = \frac{20}{0.048} = 416.67 \tag{88}$$

5 Constraining Models

5.1 Wing-Rotor Geometry Constraint

The lifting rotor radius tends to reach the upper limit during optimization design because a lower disk loading reduces power consumption [22]. A geometry constraint is implemented to prevent interferences between the lifting rotors, their attachment to the wings and minimum distances between the lifting rotors and between the lifting rotors and the fuselage (d), where the fuselage width is w. d is set fix to 0.0125m, as required by EASA, Ref. [16]. This condition applies to both the rotor radii $R_{prop_{hover}}$ and the span b. Eq. 89 and 90 are based on an lift plus cruise condition with 8 lifting rotors, as used in the demonstrated framework. Eq. 91 illustrates this approach for an undefined number of lifting rotors, but only considers $n_{lifter} \geq 4$ and $n_{lifter} \in 4\mathbb{Z}$ Fig. 12 illustrates this restriction.

$$\text{Wingspan Condition: } \frac{b}{2} \geq (3 \cdot R_{prop_{hover}} + 2 \cdot d + \frac{w}{2}) \tag{89}$$

resulting in minimum wing span b_{min},

$$b_{min} = 2 \cdot (3 \cdot R_{prop_{hover}} + 2 \cdot d + \frac{w}{2}) \tag{90}$$

or for undefined number of rotors:

$$b_{min} = 2 \cdot \left(\left(\frac{1}{4} \cdot n_{lifter} \cdot 2 \cdot R_{prop_{hover}} - R_{prop_{hover}} \right) + \frac{1}{4} \cdot n_{lifter} \cdot d + \frac{w}{2} \right) \tag{91}$$

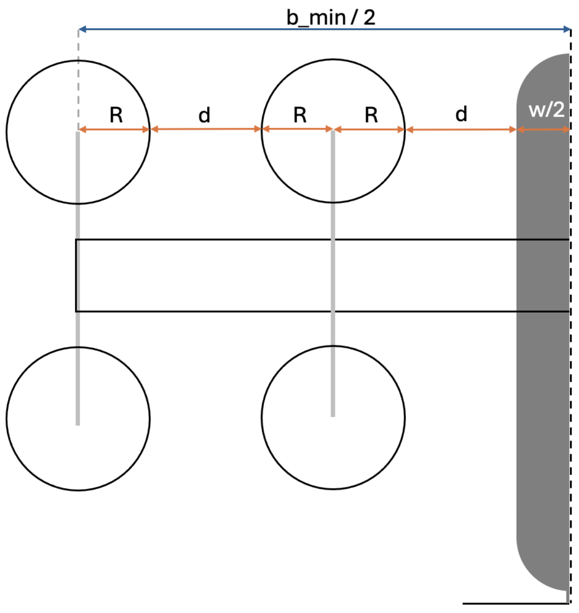


Figure 12: Geometry constraint on minimal wing span, lifting rotor and fuselage interference

5.2 Vertiport Constraint

Since the construction of new vertiports involves considerable financial investment and in many cases complex authorization procedures, a successful initial eVTOL operation can be planned on the basis of a design that meets the requirements of existing heliports. According an author-led expert interview with the helipad data provider "Helipaddy", the average D-value for existing helipads, potentially be used for UAM-operations, can be assumed for 15m. Therefore, this is assumed to be the maximum dimension of the EVTOL to maximize the likelihood of successful integration into existing air traffic infrastructure. This constraint is expressed in the following equations, where $R_{prop_{hover}}$ is the lifting rotor radius, d is the distance between lifting rotors and fuselage, w is the width of the fuselage, b is the eVTOL wing span:

$$\text{Vertiport Constraint: } 18 \geq 2 \cdot (2 \cdot d + 4 \cdot R_{prop_{hover}} + \frac{w}{2}) \tag{92}$$

$$\text{Vertiport Constraint: } 15 \geq b \tag{93}$$

Within the demonstrated framework it is assumed that $d = 0.2m$ and $w = 1.5m$. Fig. (13) illustrates the D-value, defined by EASA, Ref. [33], as the minimum space needed for an eVTOL to land, taxi, and park. If an eVTOL's D-value exceeds the as average assumed 15m, it may not fit within existing heliports, necessitating larger or specially designed vertiports. This will impacts the design and adaptability of vertiport infrastructure.

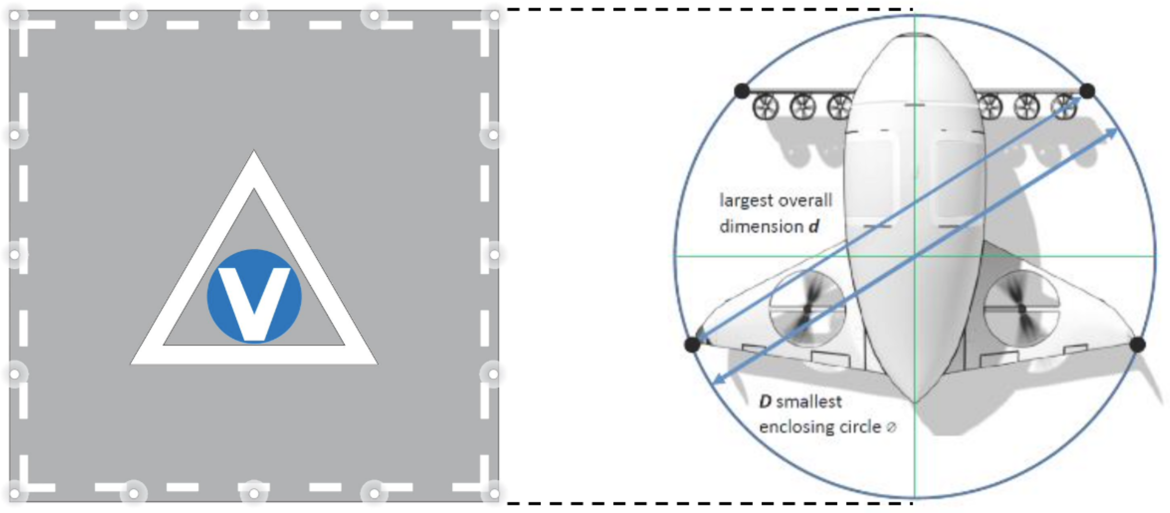

largest overall dimension *d*
D smallest enclosing circle ⌀

Figure 13: D-value for Vertiports, from Ref. [33].

6 Operational Cost Model

Assumptions and Limitations:

- The total operating cost model is structured according the ATA-67 Formula for Direct Operating cost, Ref. [34].
- The sub-assumptions are based on highly variable operational assumptions in real world applications, which can be adjusted depending on the operational business.

6.1 Total Operating Cost

The total operating costs of an aircraft per trip are divided into Cash Operating Cost (COC), Cost of Ownership (COO) and Indirect Operating Cost (IOC), according Ref. [34]:

$$TOC = COC + COO + IOC \tag{94}$$

Each cost element is further subdivided into its components, as shown in figure 14, in order to obtain as detailed a picture as possible of the cost structure of an eVTOL. The cost calculation is based on the fact that the flight hour is the only value-adding activity of an operator. Therefore, all costs associated with flight operations must be allocated to the flight hour. The sum of COC and COO is also regarded as Direct Operating Cost (DOC), similar to variable costs, and IOC as fixed costs.

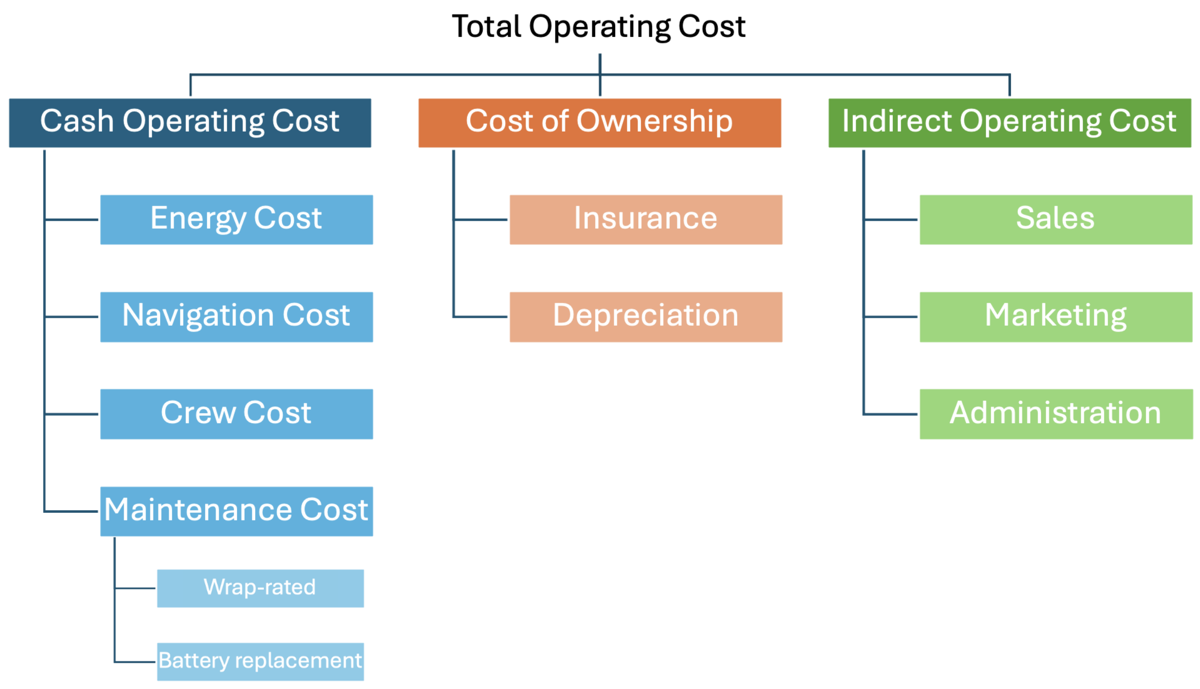


Figure 14: Total operating cost structure.

6.2 Cash Operating Cost

Cash operating costs in aviation, akin to variable costs, encompass immediate, out-of-pocket expenses. In this model, they are considered on a trip base and include energy cost C_E, maintenance cost C_M, crew cost C_C, and navigation cost C_N:

$$COO = C_E + C_M + C_C + C_N \tag{95}$$

6.2.1 Energy Cost

Assumptions and Limitations:

- The framework assumes the energy price and electricity CO_2 emissions of Germany, accounting 0.096 €/kWh and 0.3784 kg-CO_2/kWh respectively.
- A dataset for energy prices and CO_2 emissions is provided on the authors GitHub and can be used for geographic specific analysis.

Energy cost, comparable to fuel cost of combustion-engine aircraft, are defined by the simple relationship:

$$C_E = E_{trip} \cdot P_{energy} \tag{96}$$

with trip energy E_{trip} (kWh) and current energy price P_{energy} (€/kWh). E_{trip} is the used energy amount of the trip flown, and is direct related to power requirements in each flight phase and flight time, thus to mission profile design (velocity, altitude etc.). It is considered, that P_{energy} can vary depending on procedures for generation of electricity, differing in geographical regions, countries and energy policy.

6.2.2 Maintenance Cost

Assumptions and Limitations:

- The computation of this model is based on several static operational assumptions that can significantly influence the potential result.
- t_{trip} in this section is considered in hours.

The maintenance costs, C_M, are defined by:

$$C_M = C_{M_{WR}} + C_{M_B} \tag{97}$$

with $C_{M_{WR}}$ are the **wrap-rated maintenance costs**:

$$C_{M_{WR}} = MF \cdot MWR \cdot t_{trip} = 0.6 \cdot 55 \cdot t_{trip} = 33 \cdot t_{trip} \tag{98}$$

- MF (-) = maintenance man-hours per flight hour ratio, assumed as 0.6, Ref. [1].
- MWR (€/hr) = maintenance wrap-rate, assumed fix at 55€, Ref. [1].
- t_{trip} (hr) = trip duration of the considered flight, see eq. 12.

C_{M_B} is considered as **battery replacement cost**, defined by:

$$C_{M_B} = n_{Bat} \cdot P_{Bat_s} \cdot E_{BAT_{total}} \cdot \lambda_{trip} \tag{99}$$

- $E_{BAT_{total}}$ (kWh) = **total energy storage capacity** of the eVTOL-installed battery, defined by eq. (85) in chapter 4.3.
- P_{bat_s} (€/kWh) = **energy specific battery acquisition cost**. According to [30], the Tesla Model S battery cost was estimated at 155 €/kWh in 2020 and 55 €/kWh in 2030, with current costs ranging from 108 €/kWh to 142 €/kWh. Future eVTOL batteries will likely have higher capacities, requiring special attention, as battery costs can significantly impact the total cost.

Assumptions and Limitations:

- The following trivial assumption applies: Travelling a design trip distance corresponds to discharging one battery charge. In other words, the battery of the eVTOL is designed in such a way that it is sufficient to cover the design trip distance, after which it must be recharged to cover the same distance again. The discharge in the design trip distance does not mean a complete battery discharge, as unusable energy and reserves are also taken into account.
- It is assumed that the reserve is not used, but is considered as energy reserve in battery design.
- When considering the cycle life of the battery, it is assumed that the eVTOL flies the design distance on every flight.

- n_{Bat} (-) = **number of required batteries per year**. This is highly dependent on the specific battery characteristics and discharge rates. A model for battery cycle-life is proposed and derived in the following:

$$n_{Bat} = \frac{CC_{req_{eVTOL}}}{N_{cycles}} \tag{100}$$

- $CC_{req_{eVTOL}}$ (-) = **annually required charging cycles** of the eVTOL, defined by:

$$CC_{req} = \frac{FC_{annual}}{FC_{Bat}} = n_{WD} \cdot \frac{T_D}{t_{trip} \cdot DH} \cdot \frac{SE_{trip}}{SE_{trip_{design}}} \tag{101}$$

- FC_{Bat} (-) = number of flight cycles that can be flown with one battery charge, corresponds to the ratio of design trip specific energy available $SE_{trip_{design}}$ to flown trip energy SE_{trip}, where $SE_{trip_{design}}$ is defined by equation 82 in chapter 4.3:

$$FC_{Bat} = \frac{SE_{trip_{design}}}{SE_{trip}} = \frac{E_{trip_{design}}}{E_{trip}} \tag{102}$$

Assumptions and Limitations:

- The charge cycle amount of Li-Ion batteries for EV applications is dependent on the allowed end-of-life losses, set to 20% (equals 80% of begin-of-life battery capacity), and average discharge rate (C_{rate}).
- Within the provided framework we establish an empirical model to estimate the battery cycle life according equation (**??**).
- As alternate approach, one can consider a constant charging loss rate $Loss_{charge}$ of 0.048% according Ref. [30].

- N_{cycles} (-) = **number of life cycles available per battery**, based on the average C-rate $C_{rate_{vaerage}}$, depth-of-discharge (DoD) and chraging rate, until 20% degradation relative to begin-of-life (BOL) condition is met. $CC_{avlb_{Bat}}$ is determined empirical within this framework, based on eq. (87) in chapter (4.3).

$$n_{bat_{req}} = \frac{1}{N_{cycles}} \cdot \frac{n_{WD} \cdot T_D}{t_{trip} \cdot DH} \cdot \frac{e_{\text{trip}}}{e_{\text{tripm}}} \tag{103}$$

- Substituting eq: (21) and eq. (103) in eq. (99) yields the battery replacement cost function, delivering €/trip:

$$C_{M_B} = \frac{1}{N_{cycles}} \cdot P_{Bat_s} \cdot E_{BAT_{total}} \cdot \frac{e_{\text{trip}}}{e_{\text{tripm}}} \tag{104}$$

6.2.3 Navigation Cost

Assumptions and Limitations:

- The framework proposes two navigation cost computation methods, both based on german air traffic control charge authority (DFS), Ref. [35].
- The proposed simulation uses Method 1 with eq. (105) to map the navigation fees as accurately as possible.
- Eq. (108) is proposed to use to count for uncertainities in airport charges, as they can differ dependent on aircraft characterisitcs, averaging the navigation fees for all eVTOL aircraft and making them soley trip distance dependent.

Method 1: In this model the navigation charges for terminal services C_{TS} and en-route services C_{ER} of DFS [35] are considered.

$$C_N = C_{TS} + C_{ER} \tag{105}$$

with

$$C_{TS} = \left(\frac{MTOM}{1000 \cdot 50}\right)^{0.7} \cdot unitrate \tag{106}$$

and

$$C_{ER} = \left(\frac{MTOM}{1000 \cdot 50}\right)^{0.5} \cdot \frac{D_{trip}}{100} \cdot unitrate \tag{107}$$

Method 2: Actually, there will be additional airport specific landing and take-off charges in real-world operation. As these can differ significantly depending on region of operation, the DFS charges can be considered for the maximum MTOM certifiable of 5.700kg accroding EASA SC-VTOL Ref. [3] and a unit rate of 80.14€ (2024) [35] to simplify the model, resulting in total navigation charges for one trip:

$$C_N = 17.526 + 0.2706 \cdot D_{trip} \tag{108}$$

6.2.4 Crew Cost

Assumptions and Limitations:

- There are two proposed crew cost model methods. Method 1 in eq. (110) is used within the demonstrated analysis.
- Method2, eq. (112), is more simplified and not accounts for bunker based pilots.

Method 1: The crew cost per trip is defined by:

$$C_C = C_{Crew} + C_{Cabin} \tag{109}$$

Due to the predicted nature of operation for eVTOL, no cabin crew will be considered in this model, reducing eq. 109 to:

$$C_C = C_{Crew} = S_P \cdot n_{pilot} \cdot \lambda_{trip} = S_P \cdot \frac{t_{trip} \cdot CT}{U_{pilot} \cdot n_{AC}} \tag{110}$$

- S_P (€/year) = annually eVTOL pilot salary. Assumed to equal helicopter pilot salary of 80.850€ per year in 2024 in Germany, Ref. [36].
- λ_{trip} (-) = annual-trip-time ratio, as defined by equation 16 and 21.
- n_{pilot} (-) = number of pilots needed to operate the eVTOL over the year.

$$n_{pilot} = n_{WD} \cdot \frac{T_D}{U_{pilot} \cdot n_{AC}} \tag{111}$$

- U_{pilot} (hrs) = annually utilization of one pilot, assuming 2.000 hours per pilot per year.
- n_{AC} (-) = number of aircraft controlled by one pilot/operator. For pilot on-board assume $n_{AC} = 1$. For bunker-based pilot operation assume n_{AC} up to 8.

Method 2: An alternate and simplified model for crew cost would assume as fixed hourly salary of 39€ [36], resulting in:

$$C_{Crew} = 39 \cdot \frac{t_{trip}}{N_{AC}} \tag{112}$$

6.3 Cost of Ownership

Assumptions and Limitations:

- Accounting for costs of annual depreciation and aircraft insurance, distributed on the flight hour.
- Annual depreciation directly depends on eVTOL acquisition cost. The eVTOL acquisition cost is derived using the empty mass specific price, Ref. [6].
- The estimated values can differ from real world negotiated aircraft purchase prices. Further research is required at a time more data on eVTOL prices is available.
- For the annuity, fixed financial assumptions are made.

This model covers the cost of ownership in aircraft, considering depreciation C_{dep} and insurance C_{ins}, including the financial loss from the aircraft's decreasing value over time and the regular payments for coverage against potential damages or liabilities:

$$COO = C_{dep} + C_{ins} = j \cdot P_{S_{empty}} \cdot M_{empty} \cdot \frac{t_{trip} \cdot CT}{n_{WD} \cdot T_D} + x_{ins} \cdot COC \tag{113}$$

- C_{dep} (€) = **depreciation cost**, defined by:

$$C_{dep} = DEP_a \cdot \lambda_{trip} = DEP_a \cdot \frac{t_{trip} \cdot CT}{n_{WD} \cdot T_D} = j \cdot P_{S_{empty}} \cdot M_{empty} \cdot \frac{t_{trip} \cdot CT}{n_{WD} \cdot T_D} \tag{114}$$

where DEP_a is the annual depreciation costs [37], defined by:

$$DEP_a = TDA \cdot \frac{i \cdot (1 - RV) \cdot (1+i)^n}{(1+i)^n - 1} = j \cdot TDA = j \cdot P_{S_{empty}} \cdot M_{empty} \tag{115}$$

- TDA (€) = $P_{S_{empty}} \cdot M_{empty}$ = total depreciation amount, assumed to be equal to the acquisition price of an eVTOL. According [38], considering a cummulative order value of €7.4bn for 2.770 vehicles of EVE Air Mobility eVTOL's, the average investment was €2.8m. In this model we assume a relationship between the empty weight M_{empty} (eq. 44 in chapter 2) and eVTOL acquisition costs in line with Ref. [6], with an specific cost factor of $P_{s_{empty}}$=1.436,5 $€/kg_{empty}$ per kg empty weight.
- j (-) = annuity factor. Annual fraction of the total depreciation amount (TDA). Considering the made assumptions, this is set within the model for $j = 0.0796$.
- i (%) = interest rate percentage, set to 3%.
- RV (%) = percentage residual value of investment, set to 5%.
- n (years) = annuity number of years, set to 15 years.
- C_{ins} (€) = **insurance cost**, defined by:

$$C_{ins} = x_{ins} \cdot COC \tag{116}$$

- x_{ins} (%) = insurance factor. Assumed fix at 6%, Ref.[30].

Assumptions and Limitations:

- Within the provided simulation, follwing is assumed:
- $j = 0.0796$
- $x_{ins} = 0.06$
- $P_{S_{empty}}$ = 1.436,5 €/kg
- Resulting in a cost of ownership model of:

$$COO = C_{dep} + C_{ins} = 114.345 \cdot M_{empty} \cdot t_{trip} \cdot 4.61538 \cdot 10^{-4} + 0.06 \cdot COC \tag{117}$$

6.4 Indirect Operating Cost

Indirect operating costs for aircraft include expenses such as reservation and sales expenses, advertising and publicity, and general and administrative costs, which are necessary for the overall support and management of aviation operations but not directly tied to flight activities. In this model it is assumed, that these costs can be derived as fraction of the DOC [39]:

$$IOC = C_{I_{RS}} + C_{I_{AP}} + C_{I_{GA}} = (x_{IRS} + x_{IAP} + x_{IGA}) \cdot DOC \tag{118}$$

- DOC (€) = $COC + COO$ = direct operating cost, sum of cash operating cost (COO) and cost of ownership (COO).

- $C_{I_{RS}}$ (€) = reservation and sales expanses (14% of DOC)
  - x_{IRS} (%) = reservation & sales factor.
- $C_{I_{AP}}$ (€) = advertising and publicity costs (2% of DOC)
  - x_{IAP} (%) = advertising & publicity factor.
- $C_{i_{GA}}$ (€) = general and administrative costs (6% of DOC)
  - x_{IGA} (%) = general & administration factor.

Assumptions and Limitations:

- The provided framework assumes the following, Ref. [39]:
- $x_{IRS} = 14\%$
- $x_{IAP} = 2\%$
- $x_{IGA} = 6\ \%$

$$COO = (0.14 + 0.02 + 0.06) \cdot DOC = 0.22 \cdot (COC + COO) \tag{119}$$

6.5 Sepcific Cost

Using specific total operating costs, so TOC per trip divided by kilometers and seats (eq. 120) or adjusted by load factor, the passenger specific operating costs (eq. 121), allows for a more precise comparison of cost efficiency across different routes and aircraft, reflecting both operational and market performance.

$$TOC_s = \frac{COC + COO + IOC}{N_s \cdot D_{trip}} = \frac{TOC}{N_s \cdot D_{trip}}(eur/skm) \tag{120}$$

$$TOC_sp = \frac{COC + COO + IOC}{N_s \cdot LF \cdot D_{trip}} = \frac{TOC}{N_s \cdot LF \cdot D_{trip}}(eur/skm) \tag{121}$$

- N_s is the number of seats of the aircraft
- LF is the load factor during the journey

Taking eq. 121 per trip we can consider this as the break-even ticket price for one seat for one trip, which a flight operator must charge in order to avoid losses:

$$TOC_{P_{trip}} = \frac{TOC}{N_s \cdot LF}(eur) \tag{122}$$

6.6 Revenue Model

This eVTOL revenue and profit model is designed to simulate the profitability and economic incentives for future eVTOL operators. The revenue (Rev) generated from a single flight is calculated as the product of the trip distance (d_{trip}) and a fixed trip rate ($fare$) based on London ground taxi services, Ref. [40], assumed to be £1.70, which is approximately €1.98 as of August 2024.:

$$Rev = fare \cdot d_{\text{trip}}$$

The profit from each flight ($Profit_{\text{flight}}$) is then computed by subtracting the total operating cost (TOC) from the revenue:

$$Profit_{\text{flight}} = Rev - TOC$$

To determine the annual profit ($Profit_{\text{annual}}$), the profit per flight is multiplied by the annual number of flight cycles (FC_{annual}):

$$Profit_{\text{annual}} = (Rev - TOC) \cdot FC_{\text{annual}}$$

This model enables the assessment of the financial viability of eVTOL operations by evaluating revenue potential and operational costs across different trip scenarios.

The author is aware that the real-world revenue for an eVTOL trip would include an additional VAT, as in Germany approx. 19% VAT. However, as this may vary by region, it is not taken into account in this model.

7 Model Parameters

The following table summarizes the applied fixed parameters.

Item	Symbol	Value	Unit	Source
acceleration due to gravity	g	9.81	m/s^2	-
take-off/landing height AMSL	h_{tofldg}	0	m	-
hover height AGL	h_{hover}	15.24	m	[41]
cruise height AGL	h_{cruise}	1219.2	m	[41]
air density at MSL	ρ_{msl}	1.225	kg/m^3	-

Table 7: Environmental Parameters

Item	Symbol	Value	Unit	Source
wing angle of attack in cruise	$\alpha_{\text{deg\_cruise}}$	3	degree	-
wing angle of attack in climb	$\alpha_{\text{deg\_climb}}$	8	degree	-
max wing angle of attack	$\alpha_{\text{deg\_max}}$	15	degree	[10]

Table 8: Aerodynamic Parameters

Item	Symbol	Value	Unit	Source
mass per passenger	M_{pax}	82	kg	[7]
mass per luggage	M_{lug}	16	kg	[7]
EASA standard crew weight	m_{crew}	96.5	kg	[7]
Battery Density	ρ_{bat}	300/400	Wh/kg	[5]
Discharge C-rate	C_{charge}	1 a 4	1/h	[6]
Electric Efficiency	η_{e}	0.9	-	[1]
Horizontal propulsive efficiency	η_{p}	0.85	-	[1]
Hover propulsion efficiency	η_{hp}	0.7	-	[1]

Table 9: Various Operational Parameters

Item	Symbol	Value	Unit	Source
Number of seats	N_{s}	4	-	-
Mass specific aircraft acquisition price	$P_{\mathrm{s\_empty}}$	1436.5	€/kg	[6]
Number of working days per year	N_{wd}	260	days	[5]
Daily hourly flight window	T_{D}	8	hours	[5]
Energy price	P_{e}	0.096668	€/kWh	[42]
Battery replacement price	$P_{\mathrm{bat\_s}}$	115	€/kWh	[30]
VFR reserve time	t_{res}	1800	seconds	[43]
Hover time	t_{hover}	60	seconds	[41]
Annual salary for one pilot	S_{P}	45400	€	[36]
Number of aircraft controlled by one pilot	N_{AC}	1	-	[1]
Annual pilot utilization	U_{pilot}	7200000	seconds	[5]
Battery life cycle global warming impact	GWP_{battery}	124.5	kg CO2e/kWh	[44]
Electricity generation GWP for battery charging	GWP_{energy}	0.37896	kg CO2/kWh	[42]
Load factor	LF	0.67	-	[5]
Fare per km per passenger	$fare_{\mathrm{km}}$	1.98	€/km	[40]

Table 10: Economic Assumptions

Item	Symbol	Value	Unit	Source
time weighting factor	$time_{\mathrm{weight}}$	0.333	-	-
CO2 weighting factor	$co2_{\mathrm{weight}}$	0.333	-	-
costs weighting factor	$costs_{\mathrm{weight}}$	0.333	-	-

Table 11: Transportation Mode Comparison Figure of Merit Weighting

8 Multiobjective Optimization MATLAB Code Documentation

General Assumptions and Limitations:

- Transitional eVTOL: The provided Framework is designed for transition-capable electric vertical take-off and landing aircraft. Thus, multicopter are not directly covered within the framework.
- Vectored Thrust: The models used within this framework mapping the behaviour of lift+cruise eVTOL configuration. For the purpose to simulate for vectored-thrust vehicle, an additional transitional-power consumption and transition-mechanics weight estimation model should be integrated.
- Baseline: Using the baseline scenario (K10), the max. design range is determined at 120km. Above 120km, the optimization does not deliver results as constraints can't be met.

8.1 Code Overview

8.1.1 Purpose

The purpose of this optimization code is to design and optimize an electric vertical takeoff and landing (eVTOL) aircraft. The main goal is to find the optimal design variables that minimize both the trip energy and trip time. The optimization framework employs a multi-objective genetic algorithm to balance these objectives while satisfying various constraints related to aerodynamic performance, mass, and geometric limitations. The optimization problem is illustrated in Fig. 15 using extended design structure matrix (xdsm):

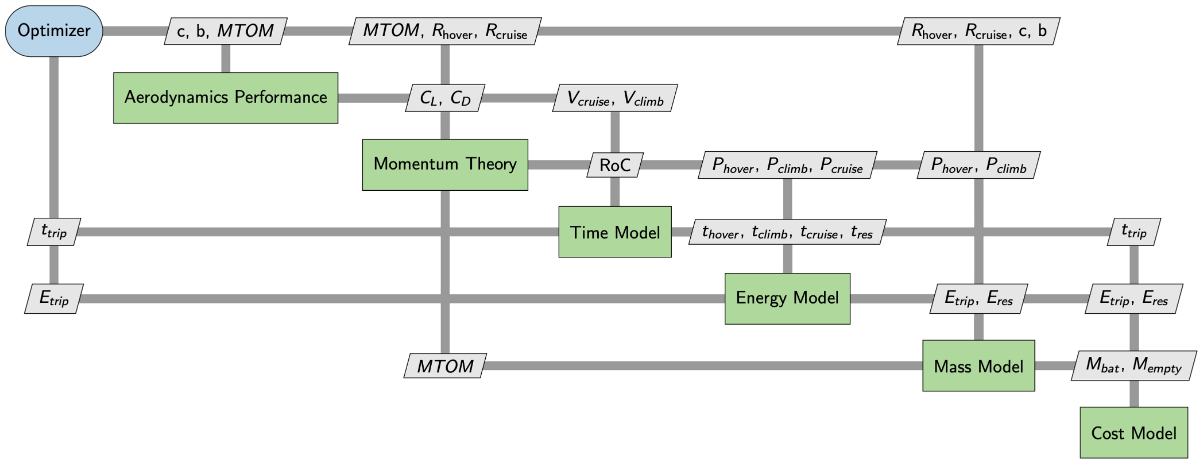


Figure 15: XDSM for initial optimization problem

8.1.2 General Code Structure

The code consists of several main components. The workflow is illustrated in Fig. 16:

1. **Optimization Execution Code:** This is the main script that sets up and runs the optimization process using the `gamultiobj` function.
2. **Fixed Parameters Setting:** A function (`getFixedParameters`) that defines and returns a set of fixed parameters used throughout the optimization process.
3. **Multi-Objective Function:** A function (`multiObjectiveFunction`) that defines the objectives to be minimized (trip energy and trip time).
4. **Nonlinear Constraints Function:** A function (`nonlinearConstraints`) that defines the constraints that must be satisfied during the optimization.

8.2 Optimization Execution Code

8.2.1 Clearing the Workspace and Importing Necessary Functions:

Purpose: The clc command clears the Command Window, and clear removes all variables, globals, functions, and MEX links from the workspace, ensuring that any previous data or functions do not interfere with the current execution. This is crucial for avoiding unexpected behavior due to residual variables or states from previous runs. The addpath commands add directories to the MATLAB search path, allowing the functions and scripts within those directories to be accessible. This is essential for modular code organization, where different functionalities (e.g., aerodynamics model, power model) are separated into different files.

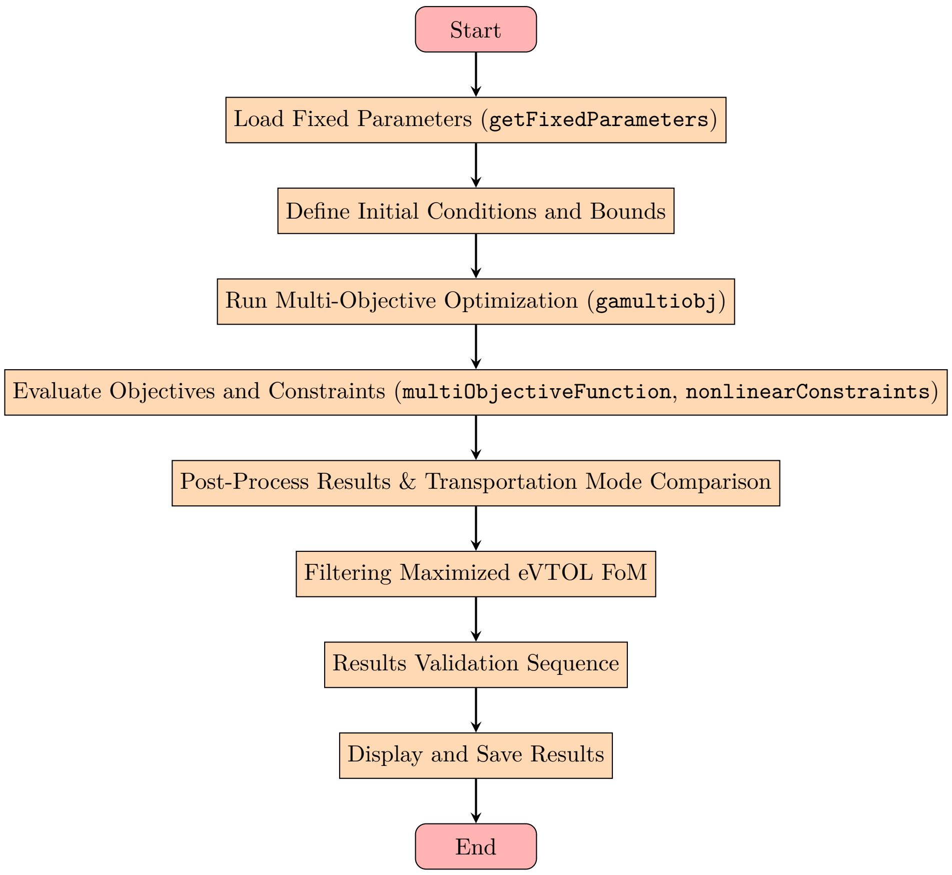

Start
Load Fixed Parameters (getFixedParameters)
Define Initial Conditions and Bounds
Run Multi-Objective Optimization (gamultiobj)
Evaluate Objectives and Constraints (multiObjectiveFunction, nonlinearConstraints)
Post-Process Results & Transportation Mode Comparison
Filtering Maximized eVTOL FoM
Results Validation Sequence
Display and Save Results
End

Figure 16: Workflow Diagram of the Optimization Code

```
clc;
clear;

% Importing the model functions
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/MDO_MOO_Model_v100724/');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Aerodynamics_Model');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Power_Model');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Time_Model');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Energy_Model');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Weight_Model');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Cost_Model/Cash
    Operating Costs');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Cost_Model/Cost of
    Ownership and IOC');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/Cost_Model/Total
    Operating Costs');
addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro
    (2))/Documents/MATLAB/Master_Project/Model_Functions/BEM_Model');
```

8.2.2 Loading Fixed Parameters:

Loading Fixed Parameters: The function `getFixedParameters` defines and returns a set of fixed parameters that are used throughout the optimization process. These parameters include physical constants, mission profile settings, aerodynamic parameters, and efficiencies. The function in described in detail in chapter 8.3.

```matlab
% Load fixed parameters
params = getFixedParameters();
```

Listing 1: Loading fixed parameters

8.2.3 Initial Guesses for Design Variables:

Initial Guesses for Design Variables: This section of the code sets initial guesses for the design variables and calculates initial estimates for various component weights of the aircraft. These initial values are used to start the optimization process. By estimating the weights of various components and calculating the initial MTOM, the code provides a starting point that helps guide the optimization algorithm towards feasible and efficient aircraft designs. These initial estimates ensure that the algorithm does not start from unrealistic values, thereby improving the efficiency and effectiveness of the optimization process.

```matlab
% Pre-calculate initial weights
c_initial = 1.5; % Initial guess for chord length
b_initial = 15;  % Initial guess for wingspan
num_props_hover_initial = 8;
num_props_cruise_initial = 1;
R_prop_hover_initial = 1.5;
R_prop_cruise_initial = 1.5;

m_wing_initial = M_wing(c_initial, b_initial);
m_rotor_initial = M_rotor(num_props_hover_initial, num_props_cruise_initial,
    R_prop_hover_initial, R_prop_cruise_initial);
m_motor_initial = M_motor(100000, 100000); % power values for initialization
m_crew = 82.5 + 14; % EASA standard crew weight
m_other = 668; % fuselage weight, and others (seats etc.). from tecnam P2006T

m_empty_initial = m_wing_initial + m_rotor_initial + m_motor_initial + m_crew
    + m_other;
m_bat_initial = M_bat(params.rho_bat, 100000, 50000); % Example energy values
    for initialization
m_pay = M_pay(params.N_s, params.M_pax, params.M_lug);

MTOM_lower_bound = m_empty_initial + m_pay + m_bat_initial;
```

8.2.4 Defining Bounds:

Bounds for Each Design Variable: These bounds define the allowable range for each design variable (e.g., cruise speed, climb speed, propeller radii, chord length, wingspan). The optimization algorithm will search within these ranges for the optimal values. Bounds ensure that the algorithm only explores feasible and physically meaningful regions of the design space, preventing it from considering unrealistic or impractical designs.

```matlab
% Define the bounds of the optimized design variables
bounds = [500, 5700;   % MTOM
          1, 2;        % R_cruise
          1, 2;        % R_hover
          1.3, 4;      % c
          8, 40;];     % b
```

Listing 2: Defining the bounds of the optimized design variables

8.2.5 Genetic Algorithm Options:

Genetic Algorithm Options: The `PopulationSize` is set to 100, meaning each generation of the genetic algorithm will consist of 100 individuals. A larger population size allows for a more diverse search of the design space, which can lead to better optimization results. However, this also increases the computational cost. The `Display` option is set to `'iter'`, which controls the level of output displayed during the optimization. Setting it to `'iter'` means that information will be displayed at each iteration of the algorithm, allowing the user to monitor the progress of the optimization. The `optimoptions` function is used to create and configure optimization options for various solvers in MATLAB. In this case, the solver is `gamultiobj`, which stands for Genetic Algorithm Multi-Objective. The genetic algorithm is chosen for its ability to handle complex, multi-modal, and non-differentiable objective functions, making it suitable for multi-objective optimization problems where multiple conflicting objectives need to be optimized simultaneously.

```
% GA options using ga for multiobjective
options = optimoptions('gamultiobj', 'PopulationSize', 100, 'Display', 'iter');
```

Listing 3: GA options using `ga` for multiobjective optimization

Executing the Genetic Algorithm: The `gamultiobj` function is used to perform multi-objective optimization using a genetic algorithm. This function is particularly useful for problems where there are multiple objectives to be optimized simultaneously, and it returns a set of optimal solutions known as the Pareto front.

```
% Run the multi-objective GA
[x, fval, exitFlag, output, population, scores] =
    gamultiobj(@(x)multiObjectiveFunction(x, params), 5, [], [], [], [],
    bounds(:,1), bounds(:,2), @(x)nonlinearConstraints(x, params), options);
```

In this command:

- The function `multiObjectiveFunction(x, params)` calculates the objectives to be minimized, based on the design variables `x` and fixed parameters `params`.
- The number `6` indicates the number of design variables to be optimized.
- The bounds for the design variables are specified by `bounds(:,1)` and `bounds(:,2)`, which define the lower and upper bounds, respectively.
- The function `nonlinearConstraints(x, params)` ensures that the design variables meet all specified constraints.
- The `options` configured earlier are applied to the genetic algorithm.

The outputs of this command include:

- `x`: The optimized design variables.
- `fval`: The objective function values corresponding to the optimized design variables.
- `exitFlag`: An integer identifying the reason the algorithm terminated.

- `output`: A structure containing information about the optimization process.
- `population`: The final population of the genetic algorithm.
- `scores`: The objective function values for the final population.

By using `optimoptions` and `gamultiobj`, we can effectively configure and execute a multi-objective genetic algorithm to explore and optimize the design space, ultimately finding a set of optimal solutions that balance multiple conflicting objectives.

8.2.6 Capturing and Storing Results:

Capturing and Storing Results: Display the optimization results, including the optimal design variables and the corresponding objective values.

```matlab
% Display results
disp('Optimized Design Variables (x):');
disp(x);
disp('Objective Values:');
disp(fval);
```

8.2.7 Post-Processing

Purpose: Once the optimization process is complete, the post-processing step recalculates more detailed parameters for each optimized solution. This includes economic modeling based on optimization results, along with figure of merit computation and post-processing of other relevant parameters like component weights, power requirements, energy consumption, and costs.

- **Verification and Validation:** The detailed recalculations ensure that the optimized solutions are feasible and adhere to all physical and operational constraints. This step verifies the initial results provided by the optimization.
- **Detailed Analysis:** Provides a more comprehensive understanding of each solution, allowing for a deeper analysis of trade-offs and performance metrics.
- **Final Selection:** Helps in selecting the most suitable design solutions by comparing additional criteria not directly used in the optimization process.

```matlab
function results = processOptimizationResults(x, fval, params,
    set_trip_distance)
    results = [];
    for i = 1:size(x, 1)
        MTOM = x(i, 1);
        R_prop_cruise = x(i, 2);
        R_prop_hover = x(i, 3);
        c = x(i, 4);
        b = x(i, 5);
        % rho_bat = x(i, 6);

        % Use the fixed parameters
        alpha_deg_cruise = params.alpha_deg_cruise;
        alpha_deg_climb = params.alpha_deg_climb;
        num_props_cruise = params.num_props_cruise;
        num_props_hover = params.num_props_hover;
        g = params.g;
        rho_hover = params.rho_hover;
        rho = params.rho;
        rho_msl = params.rho_msl;
        rho_climb = params.rho_climb;
        h_hover = params.h_hover;
        h_cruise = params.h_cruise;
        N_s = params.N_s;
        M_pax = params.M_pax;
        M_lug = params.M_lug;
        P_s_empty = params.P_s_empty;
        N_wd = params.N_wd;
        T_D = params.T_D;
        N_df = params.N_df;
        P_e = params.P_e;
        P_bat_s = params.P_bat_s;
        t_res = params.t_res;
        t_hover = params.t_hover;
        D_trip = set_trip_distance;
        eta_h = params.eta_h;
        eta_c = params.eta_c;
        m_crew = params.m_crew;
        time_weight = params.time_weight;
        co2_weight = params.co2_weight;
        energy_weight = params.energy_weight;
        costs_weight = params.costs_weight;
        l_fus_m = params.l_fus_m;
        r_fus_m = params.r_fus_m;
        rho_bat = params.rho_bat;
        C_charge = params.C_charge;
        U_pilot = params.U_pilot;
        N_AC = params.N_AC;
        T_D = params.T_D;
        S_P = params.S_P;
        GWP_battery = params.GWP_battery;
        GWP_energy = params.GWP_energy;
        LF = params.LF;
        fare_km = params.fare_km;

        % Disciplinary Analysis

        % Aerodynamic Processing
        c_l_cruise = Lift_Coefficient(alpha_deg_cruise, c, b);
        c_l_climb = Lift_Coefficient(alpha_deg_climb, c, b);
        c_d_cruise = Drag_Coefficient(c_l_cruise, c, b);
        c_d_climb = Drag_Coefficient(c_l_climb, c, b);
        V_cruise = sqrt(2*MTOM*g/(c_l_cruise*c*b*rho));
        V_climb = sqrt(2*MTOM*g/(c_l_climb*c*b*rho));
        drag_cruise = Drag(rho, V_cruise, c, b, c_d_cruise);
        roc = ROC(alpha_deg_climb, V_climb);

        AR = b / c; % aspect ratio
        WL = MTOM / (b * c); % wing loading in kg / m^2
        DL = MTOM / (num_props_hover * pi()*R_prop_hover^2); % disk loading in
            kg/m^2
        LDcruise = c_l_cruise / c_d_cruise; % LD ratio in cruise
        LDclimb = c_l_climb / c_d_climb; % LD ratio in climb

        % Power Processing (via Momentum Theory)
        Power_hover = Power_Hover_MT(MTOM, rho_hover, R_prop_hover,
            num_props_hover, eta_h);
        Power_climb = Power_Climb_MT(MTOM, rho_climb, c, b, V_climb, roc,
            c_d_climb, R_prop_cruise, num_props_cruise, eta_c, alpha_deg_climb);
        Power_cruise = Power_Cruise_MT(MTOM, rho, drag_cruise, V_cruise,
            R_prop_cruise, num_props_cruise, eta_c, 0);

        % Time and distance processing
        t_cl = t_climb(h_cruise, h_hover, roc);
        d_cl = D_climb(V_climb, t_cl);
        d_cru = D_cruise(D_trip, d_cl, 0);
        t_cr = t_cruise(d_cru, V_cruise);
        t_tot = t_cl + t_hover + 0 + t_cr;

        % Energy processing
        e_hvr = E_hover(Power_hover, t_hover);
        e_cl = E_climb(Power_climb, t_cl);
        e_cru = E_cruise(Power_cruise, t_cr);
        e_trip = E_trip(e_hvr, e_cl, e_cru, 0);
        e_res = E_res(Power_cruise, t_res);


        % Mass processing
        m_furnish = M_furnish(MTOM, V_cruise, rho);
        m_gear = M_gear(MTOM, R_prop_cruise, r_fus_m);
        m_fuselage = M_fuselage(l_fus_m, r_fus_m, MTOM, rho, rho_msl,
            V_cruise);
        m_system = M_system(l_fus_m, b, MTOM);
        m_bat = M_bat(rho_bat, e_trip, e_res);
        m_motor = M_motor(R_prop_hover, Power_hover, R_prop_cruise,
            Power_climb);
        m_wing = M_wing(c, b, V_cruise, rho, MTOM);
        m_pay = M_pay (N_s, M_pax, M_lug);
        m_rotor = M_rotor(num_props_hover, num_props_cruise, R_prop_hover,
            R_prop_cruise);
        m_empty = m_wing + m_motor + m_rotor + m_crew + m_furnish + m_fuselage
            + m_system + m_gear;
        omega_empty = m_empty / MTOM;

        e_tot = E_tot(rho_bat, m_bat); % total battery energy
        e_tripm = E_trip_max(rho_bat, m_bat, e_res); % max trip energy


        % Battery and Operations Design Module

        E_battery = rho_bat * m_bat; % total energy capcity of the battery in
            Wh
        C_rate_hover = Power_hover / E_battery;
        C_rate_climb = Power_climb / E_battery;
        C_rate_cruise = Power_cruise / E_battery;
        C_rate_average = (C_rate_hover * t_hover + C_rate_climb * t_cl +
            C_rate_cruise * t_cr) / t_tot; % c-rate discharge
        DOD = e_trip / E_battery; % depth of discharge

        T_T = 1 / C_charge * DOD * 3600;  % [seconds] turnaround time
        DH = T_T / t_tot + 1; % deadhead ratio

        n_cycles = N_cycles(DOD, C_rate_average, C_charge); % battery usage
            cycles, before replacement (degradation to 80% BOL)
        n_bat_req = 1/n_cycles * N_wd * T_D / (t_tot*DH) * e_trip / e_tripm; %
            number of batteries required to fullfill number of flight ; 416.67

        FC_d = T_D / (t_tot * DH); % daly flight cycles
        FC_a = N_wd * FC_d; % flight cycles flown per year
        FH_d = FC_d * t_tot / 3600; % daily flight hours
        FH_a = FC_a * t_tot / 3600; % annual flight hours
        P_bat_single = P_bat_s * e_tot/1000; % single battery price
        P_bat_annual = n_bat_req * P_bat_single; % yearly battery price



        % --> Cost processing

        % Cash Operating Costs
        c_e = C_E(e_trip, P_e);
        c_n = C_N(MTOM ,D_trip);
        c_cob = C_Cob(N_wd, T_D, U_pilot, N_AC, t_tot, DH, S_P);
        c_mwr = C_Mwr(t_tot);
        %supplementary energy processing

        c_mb = C_MB(P_bat_s, e_tot, n_bat_req, t_tot, DH, T_D, N_wd);
        c_m = C_M(c_mwr, c_mb);
        coc = COC(c_e, c_m, c_n, c_cob);

        % Cost of Wwnership & Indirect Operating Costs
        coo = COO(coc, omega_empty, MTOM, P_s_empty, N_wd, T_D, t_tot, DH);
        ioc = IOC(coc, omega_empty, MTOM, P_s_empty, N_wd, T_D, t_tot, DH);

        % Total Operating Costs
        toc = TOC(coc, coo, ioc);
        toc_s = TOC_s(toc, N_s, D_trip);


        % Economics & Environment

        GWP_flight = GWP_cycle(E_battery, n_bat_req, FC_a, e_trip,
            GWP_battery, GWP_energy);
        GWP_annual = GWP_flight * FC_a;
        GWP_energy_fraction = (e_trip/1000 * GWP_energy) / GWP_flight;



        revenue = fare_km * D_trip * N_s * LF; % estimated revenue per trip in

        ticket_price_pax = revenue / (N_s*LF); % estimated ticket price in
        Profit_flight = (revenue-toc); % profit per flight in
        Profit_annual = (revenue-toc) * FC_a; % annual profit in


        %%% Efficiency Metrices
        DOC = coc + coo; %      - direct operating cost per flight

        ECE = DOC / (e_trip/1000); %    /kWh - Energy Cost Efficiency
        BCE = c_mb / (e_trip/1000); %    /kWh - Battery Cost Efficienty
        RpK = revenue / D_trip; % revenue per km ratio
        PR = Profit_flight / toc; % profit ratio

        EpK = e_trip/1000 / D_trip; % kWh/km
        GWPpK = GWP_flight / D_trip; % kg CO2e / km
        CEE = GWP_flight / (e_trip/1000); % Carbon Efficiency of Energy (CEE)

        AC_cost = m_empty * P_s_empty; % aircraft acquisition cost

        CSI = DOC * GWP_flight / D_trip; % Combined Sustainability Index

        COI = (toc * GWP_flight) / (D_trip * Profit_annual/1000); % combined
            operation index


        % -> Transportation Mode Comparison processing

```

```matlab
        % Calculate eVTOL specific parameters
        time_requirement = t_tot/60; % trip time in min
        eVTOL_CO2_kg_skm = e_trip / 1000 / 4 / D_trip * 0.378; % kg CO2 per seat km
        eVTOL_energy_consumption = e_trip / 4 / D_trip; % Wh/km
        eVTOL_costs_eur_skm = toc_s; % EUR/seat km

        % Transportation mode comparison data
        modes = {'Airplane', 'Gasonline Vehicle (1)', 'Diesel Vehicle (1)', ...
                 'Electric Vehicle (1)', 'Gasonline Vehicle (5)', 'Diesel Vehicle (5)', ...
                 'Public Bus (100%)', 'Electric Vehicle (5)', 'Train (100%)', ...
                 'Bicycle', 'Airplane (79.6%)', 'Diesel Vehicle (1.3)', ...
                 'Electric Vehicle (1.3)', 'Gasonline Vehicle (1.3)', ...
                 'Public Bus (60%)', 'Train (50%)', 'eVTOL'};

        CO2_kg_skm = [0.198, 0.157, 0.128, 0.065, 0.031, 0.026, 0.013, 0.013, 0.007, 0.000, ...
                      0.249, 0.099, 0.050, 0.120, 0.022, 0.012, eVTOL_CO2_kg_skm];
        energy_consumption = [216.06, 632.40, 480.00, 172.70, 126.48, 96.00, 49.42, 34.54, 57.02, -111.11, ...
                              340.99, 369.23, 132.85, 486.46, 82.84, 114.04, eVTOL_energy_consumption];
        costs_eur_skm = [0.46, 0.117, 0.083, 0.105, 0.023, 0.017, 0.060, 0.021, 0.200, -0.491, ...
                         0.579, 0.064, 0.081, 0.090, 0.104, 0.402, eVTOL_costs_eur_skm];

        velocities = struct('Cars_up_to_60km', 60.00, ...
                            'Cars_above_60km', 85.00, ...
                            'Airplane_below_400km', 74.00, ...
                            'Airplane_above_400km', 151.00, ...
                            'Bicycle', 18.80, ...
                            'Public_Bus_up_to_60km', 39.70, ...
                            'Public_Bus_above_60km', 64.00, ...
                            'Train_up_to_60km', 49.10, ...
                            'Train_above_60km', 99.00);

        circuity_ratios = struct('Cars_up_to_180km', 1.30, ...
                                 'Cars_above_180km', 1.20, ...
                                 'Train_all', 1.20, ...
                                 'Airplane_all', 1.05, ...
                                 'Bicycle_all', 1.28, ...
                                 'Bus_up_to_100km', 1.60, ...
                                 'Bus_above_100km', 1.25);

        adjusted_trip_distance = zeros(size(modes));
        for j = 1:length(modes)
            mode = modes{j};
            if contains(mode, 'Gasonline Vehicle') || contains(mode, 'Diesel Vehicle') || contains(mode, 'Electric Vehicle')
                if D_trip <= 180
                    adjusted_trip_distance(j) = D_trip * circuity_ratios.Cars_up_to_180km;
                else
                    adjusted_trip_distance(j) = D_trip * circuity_ratios.Cars_above_180km;
                end
            elseif contains(mode, 'Airplane')
                adjusted_trip_distance(j) = D_trip * circuity_ratios.Airplane_all;
            elseif strcmp(mode, 'Bicycle')
                adjusted_trip_distance(j) = D_trip * circuity_ratios.Bicycle_all;
            elseif contains(mode, 'Public Bus')
                if D_trip <= 100
                    adjusted_trip_distance(j) = D_trip * circuity_ratios.Bus_up_to_100km;
                else
                    adjusted_trip_distance(j) = D_trip * circuity_ratios.Bus_above_100km;
                end
            elseif contains(mode, 'Train')
                adjusted_trip_distance(j) = D_trip * circuity_ratios.Train_all;
            elseif strcmp(mode, 'eVTOL')
                % eVTOL does not have a circuity ratio adjustment
                adjusted_trip_distance(j) = D_trip;
            end
        end

        CO2_total = CO2_kg_skm .* adjusted_trip_distance;
        energy_total = energy_consumption .* adjusted_trip_distance;
        costs_total = costs_eur_skm .* adjusted_trip_distance;

        CO2_total(end) = eVTOL_CO2_kg_skm * adjusted_trip_distance(end);
        energy_total(end) = eVTOL_energy_consumption * adjusted_trip_distance(end);
        costs_total(end) = eVTOL_costs_eur_skm * adjusted_trip_distance(end);

        % Determine time demand based on average velocity and adjusted trip distance
        time_demand = zeros(size(modes));
        for j = 1:length(modes)
            mode = modes{j};
            if contains(mode, 'Gasonline Vehicle') || contains(mode, 'Diesel Vehicle') || contains(mode, 'Electric Vehicle')
                if D_trip <= 60
                    time_demand(j) = adjusted_trip_distance(j) / velocities.Cars_up_to_60km * 60; % converting hours to minutes
                else
                    time_demand(j) = adjusted_trip_distance(j) / velocities.Cars_above_60km * 60; % converting hours to minutes
                end
            elseif contains(mode, 'Airplane')
                if D_trip <= 400
                    time_demand(j) = (adjusted_trip_distance(j) / velocities.Airplane_below_400km * 60) + 120; % converting hours to minutes and adding 2h airport time
                else
                    time_demand(j) = (adjusted_trip_distance(j) / velocities.Airplane_above_400km * 60) + 120; % converting hours to minutes and adding 2h airport time
                end
            elseif strcmp(mode, 'Bicycle')
                time_demand(j) = adjusted_trip_distance(j) / velocities.Bicycle * 60; % converting hours to minutes
            elseif contains(mode, 'Public Bus')
                if D_trip <= 60
                    time_demand(j) = adjusted_trip_distance(j) / velocities.Public_Bus_up_to_60km * 60; % converting hours to minutes
                else
```

```
                    time_demand(j) = adjusted_trip_distance(j) /
                        velocities.Public_Bus_above_60km * 60; % converting
                        hours to minutes
                end
            elseif contains(mode, 'Train')
                if D_trip <= 60
                    time_demand(j) = adjusted_trip_distance(j) /
                        velocities.Train_up_to_60km * 60; % converting hours to
                        minutes
                else
                    time_demand(j) = adjusted_trip_distance(j) /
                        velocities.Train_above_60km * 60; % converting hours to
                        minutes
                end
            elseif strcmp(mode, 'eVTOL')
                % Using eVTOL velocity
                time_demand(j) = time_requirement; % converting hours to
                    minutes
            end
        end

        % Calculate ratings between 1 and 10 for each mode
        calculate_rating = @(x, X_min, X_max) (1 * (x - X_min) - 10 * (x -
            X_max)) / (X_max - X_min);

        time_rating = calculate_rating(time_demand, min(time_demand),
            max(time_demand));
        co2_rating = calculate_rating(CO2_total, min(CO2_total),
            max(CO2_total));
        energy_rating = calculate_rating(energy_total, min(energy_total),
            max(energy_total));
        costs_rating = calculate_rating(costs_total, min(costs_total),
            max(costs_total));

        % Calculate the Figure of Merit (FoM) for each mode
        FoM = (time_weight * time_rating') + ...
        (co2_weight * co2_rating') + ...
        (energy_weight * energy_rating') + ...
        (costs_weight * costs_rating');

        % Store the FoM for the eVTOL
        eVTOL_FoM = FoM(end);

        % Append the results
        results = [results; fval(i, :), x(i, :), rho_bat, V_cruise, V_climb,
            Power_hover, Power_climb, Power_cruise, E_battery/1000, EpK,
            m_empty, m_pay, m_bat, m_motor, m_rotor, m_wing, m_gear, m_furnish,
            m_fuselage, m_system, c_l_cruise, c_l_climb, c_d_cruise, c_d_climb,
            LDcruise, LDclimb, roc*196.85, AR, WL, DL, toc_s, toc, DOC, coc,
            c_e, c_m, c_mwr, c_mb, c_n, c_cob, coo, ioc, AC_cost, ECE, BCE,
            n_bat_req, n_cycles, DOD, C_rate_hover, C_rate_climb,
            C_rate_cruise, C_rate_average, C_charge, T_T, DH, P_bat_single,
            P_bat_annual, GWP_flight, GWP_annual, GWPpK, CEE,
            GWP_energy_fraction, CSI, COI, FC_a, FC_d, FH_a, FH_d,
            Profit_annual, Profit_flight, PR, RpK, ticket_price_pax, d_cru,
            d_cl, eVTOL_FoM];
    end
end
```

Listing 4: Processing optimization results

8.2.8 Results and Pareto-Front

After the optimization and post-processing steps, the results are organized and analyzed using a structured approach. This involves defining the names of the objectives, design variables, and calculated values, creating a table for the results, sorting the table, and visualizing the Pareto front. **Defining Names**

- A table is created for the results with custom names assigned to each column for clarity.
- The table is sorted by the first objective function, which is 'Trip Energy', to prioritize solutions based on energy efficiency.
- A Pareto front is plotted to visualize the trade-off between 'Trip Energy' and 'Trip Time'.
- The x-axis represents 'Trip Energy' (Wh), and the y-axis represents 'Trip Time' (minutes).
- The plot helps in identifying the optimal solutions that balance both objectives effectively.
- The sorted results table is saved to an Excel file for further analysis and record-keeping.

```

% Defining the names of the objectives, design variables, and calculated values
objectiveNames = {'Trip_Energy [Wh]', 'Trip_Time [sec]'};
designVarNames = {'MTOM [kg]', 'R_cruise [m]', 'R_hover [m]', 'chord [m]',
    'wingspan_b [m]'};
calculatedVarNames = {'rho_bat [Wh/kg]', 'V_cruise [m/s]', 'V_climb [m/s]',
    'Power_hover [W]', 'Power_climb [W]', 'Power_cruise [W]', 'E_battery
    [kWh]', 'EpK [kWh/km]', 'm_empty [kg]', 'm_pay [kg]', 'm_bat [kg]',
    'm_motor [kg]', 'm_rotor [kg]', 'm_wing [kg]', 'm_gear [kg]', 'm_furnish
    [kg]', 'm_fuselage [kg]', 'm_system [kg]', 'c_l_cruise [-]', 'c_l_climb
    [-]', 'c_d_cruise [-]', 'c_d_climb [-]', 'LD_cruise [-]', 'LD_climb [-]',
    'ROC [ft/min]', 'Aspect Ratio [-]', 'Wing Loading [kg/m^2]', 'Disk Loading
    [kg/m^2]', 'TOC_skm [   /skm]', 'TOC [   ]', 'DOC [   ]', 'COC [   ]',
    'C_energy [   ]', 'C_m [   ]', 'C_mwr [   ]', 'C_mb [   ]', 'C_nav [   ]',
    'C_crew [   ]', 'COO [   ]', 'IOC [   ]', 'AC_cost_unit [   ]', 'ECE
    [   /kWh]', 'BCE [   /kWh]', 'n_bat_req_a [-]', 'n_bat_cycles [-]', 'DOD
    [%/100]', 'C_rate_hover [1/h]', 'C_rate_climb [1/h]', 'C_rate_cruise
    [1/h]', 'C_rate_average [1/h]', 'C_charge [1/h]', 'T_T [sec]', 'DH [-]',
    'P_bat_single [   ]', 'P_bat_annual [   ]', 'GWP_flight [CO2e kg]',
    'GWP_annual [CO2e kg]', 'GWPpK [CO2e kg/km]', 'CEE [CO2e kg/kWh]',
    'GWP_trip_energy_fraction [-]', 'CSI [   *kgCO2e/km]', 'COI [-]', 'FC_a
    [-]', 'FC_d [-]', 'FH_a [hrs]', 'FH_d [hrs]', 'Profit_annual [   ]',
    'Profit_flight [   ]', 'PR [-]', 'RpK [   /km]', 'ticket_price_pax [   ]',
    'd_cru [m]', 'd_cl [m]', 'eVTOL_FoM [-]'};

% Creating a table for the results with custom names
resultsTable = array2table(results, 'VariableNames', [objectiveNames,
    designVarNames, calculatedVarNames]);

% Sorting the table by the first objective function (change the index to sort
    by a different objective)
sortedResultsTable = sortrows(resultsTable, 'Trip_Energy [Wh]');

% to display the sorted table
disp('Sorted Results Table:');
disp(sortedResultsTable);

% Plotting the Pareto front
figure;
scatter(fval(:, 1), fval(:, 2)/60, 'filled');
title('Pareto Front');
```

```
xlabel('Trip Energy (Wh)');
ylabel('Trip time (minutes)');
grid on;
```

Listing 5: Defining objectives, design variables, and calculated values; creating and sorting results table; plotting the Pareto front

8.3 Fixed Parameters Settings

Purpose and Importance: The fixed parameters are crucial for the optimization process as they provide constant values that define the environmental settings, mission profile, and vehicle characteristics. These parameters ensure consistency and reliability in the simulation and optimization models by providing baseline values for calculations. **Detailed Parameter List:**

- **Standard Environmental Settings:**
  - `params.g = 9.81;` - Acceleration due to gravity in m/s^2.
- **Mission Profile Settings:**
  - `params.h_hover = 30.5;` - Hover height in meters Above Ground Level (AGL).
  - `params.h_cruise = 304.7;` - Cruise height in meters AGL.
- **Atmospheric Model:**
  - `rho_msl = 1.225;` - Air density at mean sea level in kg/m^3.
  - `params.rho_hover = rho_tofldg;` - Air density at take-off and landing height.
  - `params.rho = rho_cruise;` - Air density at cruise height.
  - `params.rho_climb = rho_climb;` - Air density during climb.
- **Aerodynamic Parameters:**
  - `params.alpha_deg_cruise = 3;` - Wing angle of attack in cruise in degrees.
  - `params.alpha_deg_climb = 8;` - Wing angle of attack in climb in degrees.
- **eVTOL Configuration:**
  - `params.num_props_cruise = 1;` - Number of propellers in cruise configuration.
  - `params.num_props_hover = 8;` - Number of propellers in hover configuration.
- **Operational Parameters:**
  - `params.N_s = 4;` - Number of seats.
  - `params.M_pax = 82;` - Mass per passenger in kg.
  - `params.M_lug = 16;` - Mass per luggage in kg.
  - `params.P_s_empty = 1436.5;` - Cost per kg of empty mass in euros.
  - `params.N_wd = 260;` - Number of working days per year.
  - `params.N_df = 6;` - Number of daily flights.
  - `params.P_e = 0.09;` - Cost per kWh of electricity in euros.
  - `params.P_bat_s = 115;` - Specific price of battery in euros per kWh.
  - `params.t_res = 60 * 30;` - 30 minutes reserve time for VFR according to FAA in seconds.

  - `params.t_hover = 60;` - Example hover time in seconds.
  - `params.D_trip = 50;` - Example trip distance in km.

- **Efficiencies:**
  - `params.eta_e = 0.9;` - Electric efficiency.
  - `params.eta_p = 0.85;` - Horizontal propulsive efficiency.
  - `params.eta_hp = 0.7;` - Hover propulsion efficiency.
  - `params.eta_h = params.eta_hp * params.eta_e;` - Combined hover efficiency.
  - `params.eta_c = params.eta_p * params.eta_e;` - Combined cruise efficiency.

- **Battery Parameters:**
  - `params.rho_bat = 300;` - Battery energy density in Wh/kg.

Parameter Initialization Function: The function `getFixedParameters` initializes and returns a structured array `params` containing all fixed parameters used in the optimization process. These parameters include environmental settings, mission profiles, aerodynamic properties, eVTOL configuration, operational parameters, and efficiencies. By centralizing these constants, the function ensures consistency and simplifies modifications for future adjustments.

```matlab
function params = getFixedParameters()
  % Fixed parameters as derived and defined in the model documentation
  addpath('/Users/johannes/Dropbox/Mein Mac (Johanness MacBook Pro (2))/Documents/MATLAB/Master_Project/MDO_MOO_Model_v100724/');

  % Standard Environmental Settings
  params.g = 9.81; % Acceleration due to gravity in m/s^2

  % Model Flight Height Settings
  params.h_tofldg = 0; % [m], take-off and landing height AMSL, (above mean sea level)
  params.h_hover = 15.24; % [m] 30.5m = 100ft, 15.24m = 50ft, Hover height AGL, (above ground level)
  params.h_cruise = 1219.2; % [m], 304.7m = 1000ft, Cruise height AGL (above ground level), 1219.2m = 4000ft

  % Atmospheric Model
  rho_msl = 1.225; % [kg/m^3], air density at main sea level (MSL)
  rho_tofldg = rho_msl*(1-22.558*10^(-6)*params.h_tofldg)^4.2559; % [kg/m^3], air density at take-off and landing area
  rho_cruise = rho_msl*(1-22.558*10^(-6)*(params.h_cruise + params.h_tofldg))^4.2559; % [kg/m^3], air density at cruise height
  rho_climb = rho_msl*(1-22.558*10^(-6)*(params.h_cruise/2 + params.h_tofldg))^4.2559; % [kg/m^3], assumed air density during climb
  params.rho_hover = rho_tofldg;
  params.rho = rho_cruise;
  params.rho_climb = rho_climb;
  params.rho_msl = rho_msl;

  % Aerodynamic Paramaters
  params.alpha_deg_cruise = 3; % wing angle of attack in cruise [-]
  params.alpha_deg_climb = 8; % wing angle of attack in climb [-]
  params.alpha_deg_max = 15; % max wing angle of attack pre stall [-]

  % Design: Number of Props in eVTOL Configuration
  params.num_props_cruise = 1; % number of props in cruise (horizontal flight) configuration -
  params.num_props_hover = 8; % number of prop in hover (vertical flight) configuration -
```

```
    params.d_tip = 0.025; % [m] % spacing between rotors and fuselage
    params.w_fuselage = 1.5; % [m] width of fuselage
    params.l_fus_m = 6; % assuming 9m for fuselage length (= aircraft length) [see Tecnam P2006T]
    params.r_fus_m = 0.75; % assuming 1.5m for cabin width [see Tecnam P2006T]

    % Mass assumptions
    params.M_pax = 82; % [kg], Mass per passenger
    params.M_lugg = 16; % [kg], Mass per luggage
    params.m_crew = 82.5 + 14; % [kg], EASA standard crew weight
    params.m_other = 668; % [kg], fuselage weight, and others (seats etc.). from tecnam P2006T
    params.m_motor_initial = 500; % [kg], initial guess for motor mass
    params.m_bat_initial = 200; % [kg], initial guess for battery mass

    % Battery assumptions
    params.rho_bat = 400; % [Wh/kg] fixed assumption on battery density (400Wh/kg acc. Uber Eleveate requirement)
    params.C_charge =1; % [1/h] C-rate (discharge rate of battery)


    % Economic assumptions
    params.N_s = 4; % Number of seats
    params.P_s_empty = 1436.5; % [   /kg of M_empty], weight specific aircraft acquisition price
    params.N_wd = 260; % [days],  Number of working days per year, based on uber
    params.N_df = 6; % [-], Number of daily flights
    params.P_e = 0.096668; % [   /kWh], energy price
    params.P_bat_s = 115; % [   /kWh], energy capacity specific battery replacement price
    params.t_res = 60 * 30; % [seconds], 30min reserve for VFR acc. FAA
    params.t_hover = 60; % [seconds], hover time in seconds
    params.T_D = 8*60*60; % assuming a fixed operation window of 8hours daily, based on uber
    params.S_P = 45400; % [   ] annual salary for one pilot, based on Uber
    params.N_AC = 1; % [-] number of aircraft controlled by one pilot (>1, bunker based operation)
    params.U_pilot = 2000*60*60; %[sec] annual pilot utilization in seconds (2000hrs)
    params.GWP_battery = 124.5, % [kg CO2e/kWh] LIB NMC battery life cycle global warming impact
    params.GWP_energy = 0.37896; %[kg CO2/kWh] electrcity generation GWP for battery charging
    params.LF = 0.67; % load factor, based on uber
    params.fare_km = 1.98; % euro per km per passenger, based on london taxi day fare



    % Efficiencies
    params.eta_e = 0.9; % electric efficiency
    params.eta_p = 0.85; % horizontal propulsive efficiency
    params.eta_hp = 0.7; % hover propulsion efficiency
    params.eta_h = params.eta_hp * params.eta_e; % Hover efficiency
    params.eta_c = params.eta_p * params.eta_e; % horizontal flight efficiency

    % Transportation Mode Comparison Figure of Merit Weighting
    params.time_weight = 0.333;
    params.co2_weight = 0.333;
    params.energy_weight = 0;
    params.costs_weight = 0.333;

```

```
end
```

Listing 6: Defining fixed parameters in the model

8.4 Multi-Objective Function

Objective Function Overview: The multi-objective function plays a crucial role in the optimization process by evaluating the performance of design solutions based on multiple criteria. In this context, the primary objectives being optimized are *trip energy* and *trip time*. The function aims to minimize both of these objectives, guiding the optimization algorithm towards the most efficient and effective designs. The `multiObjectiveFunction` is implemented to extract and utilize design variables for detailed performance calculations. By systematically analyzing these variables and their interactions, the `multiObjectiveFunction` ensures that the optimization algorithm can identify design solutions that minimize both trip energy and trip time, leading to efficient and viable aircraft designs.

```
  function f = multiObjectiveFunction(x, params, set_trip_distance)
    % Extracting design variables
    MTOM = x(1);
    R_prop_cruise = x(2);
    R_prop_hover = x(3);
    c = x(4);
    b = x(5);

    % Using the fixed parameters
        alpha_deg_cruise = params.alpha_deg_cruise;
        alpha_deg_climb = params.alpha_deg_climb;
        num_props_cruise = params.num_props_cruise;
        num_props_hover = params.num_props_hover;
        g = params.g;
        rho_hover = params.rho_hover;
        rho = params.rho;
        rho_msl = params.rho_msl;
        rho_climb = params.rho_climb;
        h_hover = params.h_hover;
        h_cruise = params.h_cruise;
        N_s = params.N_s;
        M_pax = params.M_pax;
        M_lug = params.M_lug;
        P_s_empty = params.P_s_empty;
        N_wd = params.N_wd;
        T_D = params.T_D;
        N_df = params.N_df;
        P_e = params.P_e;
        P_bat_s = params.P_bat_s;
        t_res = params.t_res;
        t_hover = params.t_hover;
        D_trip = set_trip_distance;
        eta_h = params.eta_h;
        eta_c = params.eta_c;
        m_crew = params.m_crew;
        time_weight = params.time_weight;
        co2_weight = params.co2_weight;
        energy_weight = params.energy_weight;
        costs_weight = params.costs_weight;
        l_fus_m = params.l_fus_m;
        r_fus_m = params.r_fus_m;
        rho_bat = params.rho_bat;


    % Aerodynamic & Performance Processing
    c_l_cruise = Lift_Coefficient(alpha_deg_cruise, c, b);
    c_l_climb = Lift_Coefficient(alpha_deg_climb, c, b);
    c_d_cruise = Drag_Coefficient(c_l_cruise, c, b);
    c_d_climb = Drag_Coefficient(c_l_climb, c, b);
    V_cruise = sqrt(2*MTOM*g/(c_l_cruise*c*b*rho));
    V_climb = sqrt(2*MTOM*g/(c_l_climb*c*b*rho));
    drag_cruise = Drag(rho, V_cruise, c, b, c_d_cruise);
    roc = ROC(alpha_deg_climb, V_climb);

    % Power Processing (via Momentum Theory)
    Power_hover = Power_Hover_MT(MTOM, rho_hover, R_prop_hover,
        num_props_hover, eta_h);
    Power_climb = Power_Climb_MT(MTOM, rho_climb, c, b, V_climb, roc,
        c_d_climb, R_prop_cruise, num_props_cruise, eta_c, alpha_deg_climb);
    Power_cruise = Power_Cruise_MT(MTOM, rho, drag_cruise, V_cruise,
        R_prop_cruise, num_props_cruise, eta_c, 0);

    % Time and distance Processing
    t_cl = t_climb(h_cruise, h_hover, roc);
    d_cl = D_climb(V_climb, t_cl);
    d_cru = D_cruise(D_trip, d_cl, 0);
    t_cr = t_cruise(d_cru, V_cruise);
    t_tot = (t_cl + t_hover + t_cr);

    % Energy processing
    e_hvr = E_hover(Power_hover, t_hover);
    e_cl = E_climb(Power_climb, t_cl);
    e_cru = E_cruise(Power_cruise, t_cr);
    e_trip = E_trip(e_hvr, e_cl, e_cru, 0);

    % Define objective functions
    f = [e_trip, t_tot]; % Minimizing energy trip and trip time
end
```

Listing 7: Multi-objective function for optimization

8.5 Nonlinear Constraints Function

Constraints are essential in the optimization process as they ensure that the solutions generated by the optimization algorithm adhere to physical, operational, and regulatory limits. These constraints help in maintaining the feasibility and practicality of the design solutions. The `nonlinearConstraints` function is implemented to enforce various constraints on the design variables during the optimization. The specific constraints applied include aerodynamic, geometric, and regulatory constraints, each ensuring the design remains within permissible bounds. The function interacts with design variables by extracting them and using them in conjunction with fixed parameters to compute the constraints.

```
 function [c, ceq] = nonlinearConstraints(x, params, set_trip_distance)
    % Extract design variables
    MTOM = x(1);
    R_prop_cruise = x(2);
    R_prop_hover = x(3);
    c = x(4);
    b = x(5);
    % rho_bat = x(6);


    % Use the fixed parameters
    alpha_deg_cruise = params.alpha_deg_cruise;
    alpha_deg_climb = params.alpha_deg_climb;
    num_props_cruise = params.num_props_cruise;
    num_props_hover = params.num_props_hover;
    g = params.g;
    rho = params.rho;
    rho_msl = params.rho_msl;
    rho_hover = params.rho_hover;
    rho_climb = params.rho_climb;
    h_hover = params.h_hover;
    h_cruise = params.h_cruise;
    N_s = params.N_s;
    M_pax = params.M_pax;
    M_lug = params.M_lug;
    t_res = params.t_res;
    t_hover = params.t_hover;
    D_trip = set_trip_distance;
    eta_h = params.eta_h;
    eta_c = params.eta_c;
    m_crew = params.m_crew;
    d_tip = params.d_tip;
    w_fuselage = params.w_fuselage;
    l_fus_m = params.l_fus_m;
    r_fus_m = params.r_fus_m;
    rho_bat = params.rho_bat;


    % --> Disciplinary Analysis:

    % Aerodynamic Processing
    c_l_cruise = Lift_Coefficient(alpha_deg_cruise, c, b);
    c_l_climb = Lift_Coefficient(alpha_deg_climb, c, b);
    c_d_cruise = Drag_Coefficient(c_l_cruise, c, b);
    c_d_climb = Drag_Coefficient(c_l_climb, c, b);
    V_cruise = sqrt(2*MTOM*g/(c_l_cruise*c*b*rho));
    V_climb = sqrt(2*MTOM*g/(c_l_climb*c*b*rho));
    drag_cruise = Drag(rho, V_cruise, c, b, c_d_cruise);
    roc = ROC(alpha_deg_climb, V_climb);

    % Power Processing (via Momentum Theory)
    Power_hover = Power_Hover_MT(MTOM, rho_hover, R_prop_hover,
        num_props_hover, eta_h);
    Power_climb = Power_Climb_MT(MTOM, rho_climb, c, b, V_climb, roc,
        c_d_climb, R_prop_cruise, num_props_cruise, eta_c, alpha_deg_climb);
    Power_cruise = Power_Cruise_MT(MTOM, rho, drag_cruise, V_cruise,
        R_prop_cruise, num_props_cruise, eta_c, 0);

    % Time and distance processing
    t_cl = t_climb(h_cruise, h_hover, roc);
    d_cl = D_climb(V_climb, t_cl);
    d_cru = D_cruise(D_trip, d_cl, 0);
    t_cr = t_cruise(d_cru, V_cruise);


    % Energy processing
    e_hvr = E_hover(Power_hover, t_hover);
    e_cl = E_climb(Power_climb, t_cl);
    e_cru = E_cruise(Power_cruise, t_cr);
    e_trip = E_trip(e_hvr, e_cl, e_cru, 0);
    e_res = E_res(Power_cruise, t_res);

    % Mass processing
    m_furnish = M_furnish(MTOM, V_cruise, rho);
    m_gear = M_gear(MTOM, R_prop_cruise, r_fus_m);
    m_fuselage = M_fuselage(l_fus_m, r_fus_m, MTOM, rho, rho_msl, V_cruise);
    m_system = M_system(l_fus_m, b, MTOM);
    m_bat = M_bat(rho_bat, e_trip, e_res);
    m_motor = M_motor(R_prop_hover, Power_hover, R_prop_cruise, Power_climb);
    m_wing = M_wing(c, b, V_cruise, rho, MTOM);
    m_pay = M_pay (N_s, M_pax, M_lug);
    m_rotor = M_rotor(num_props_hover, num_props_cruise, R_prop_hover,
        R_prop_cruise);
    m_empty = m_wing + m_motor + m_rotor + m_crew + m_furnish + m_fuselage +
        m_system + m_gear;


    % -> Constraint Definition:


    geometry_constraint = 2 * (num_props_hover / 4 * 2 * R_prop_hover * 3/4 +
        2 * d_tip + w_fuselage / 2) - b; % constraint on lifting rotor and
        fuselage distance
    vertiport_constraint1 = 2 * (2 * d_tip + 4 * R_prop_hover + w_fuselage /
        2) - 15; % 15 m vertiport constraint (according to EASA PTS VTP) with
        10% margin
    vertiport_constraint2 = b - 15; %15 m vertiport constraint (according to
        EASA PTS VTP) with 10% margin
    tolerance_percentage = 0.025; % 11% Tolerance
    tolerance_value = tolerance_percentage * MTOM;
    MTOM_constraint2 = abs(MTOM - (m_bat + m_empty + m_pay)) - tolerance_value;
    wing = b/c - 15;

    ceq = [];
    c = [ % insert scenario constraints here
         geometry_constraint;
         MTOM_constraint2];
end
```

Listing 8: Nonlinear constraints function for optimization

8.6 Conclusion and Recommendations

The provided MATLAB code facilitates a multi-objective optimization framework for the design of an eVTOL aircraft. The main components and flow of the code are as follows:

- **Initialization:**
  - Clear the workspace and import necessary model functions.
  - Load fixed parameters using the `getFixedParameters` function.
- **Pre-calculation of Initial Weights:**
  - Make initial guesses for design variables such as chord length, wingspan, and propeller radii.
  - Calculate initial component weights to provide a starting point for further iterations.
- **Defining Optimization Bounds:**
  - Specify bounds for the design variables to guide the optimization process within realistic and feasible ranges.
- **Multi-Objective Optimization:**
  - Utilize a genetic algorithm (`gamultiobj`) to optimize two main objectives: trip energy and trip time.

  - The `multiObjectiveFunction` evaluates each candidate solution based on detailed aerodynamic, mass, power, and energy calculations.

- **Nonlinear Constraints:**
  - The `nonlinearConstraints` function ensures that all design solutions adhere to physical, geometric, and regulatory constraints, maintaining feasibility and compliance with standards.

- **Post-Processing:**
  - Recalculate the MTOM and other parameters for each optimized solution.
  - Organize the results into a table, sort, and visualize to highlight the trade-offs between objectives.

- **Results Presentation:**
  - Display final results, including detailed metrics and visualizations such as the Pareto front.
  - Save results to an Excel file for further analysis and decision-making.

Recommendations for Future Use

- **Enhancement of Objective Functions:**
  - Incorporate more detailed and diverse objective functions to capture additional performance metrics and improve the robustness of the optimization.

- **Integration of More Constraints:**
  - Add more constraints related to safety, environmental impact, and operational limits to further refine the design space and ensure compliance with a broader range of requirements.

- **Optimization Algorithms:**
  - Explore the use of other optimization algorithms, such as particle swarm optimization or differential evolution, to compare performance and potentially achieve better solutions.

- **Modularization and Code Efficiency:**
  - Refactor the code into more modular functions to enhance readability and maintainability.
  - Focus on optimizing the computational efficiency of the calculations, especially for large-scale problems.

- **Validation and Testing:**
  - Implement rigorous validation and testing of the models and optimization results against real-world data and scenarios to ensure accuracy and reliability.

- **User Interface and Automation:**
  - Develop a user-friendly interface to facilitate easier input of parameters and viewing of results.
  - Automate the process to allow for seamless execution and analysis, particularly for iterative design processes.

References

[1] A. Brown and W. L. Harris. *A Vehicle Design and Optimization Model for On-Demand Aviation.* Reston, VA: American Institute of Aeronautics and Astronautics, 2018.

[2] SKYbrary. *True Airspeed.* `https://skybrary.aero/articles/true-airspeed`. Accessed: 2024-06-12. 2024. URL: `https://skybrary.aero/articles/true-airspeed`.

[3] European Union Aviation Safety Agency. *Special Condition VTOL.* Accessed: 2024-06-12. 2024. URL: `https://www.easa.europa.eu/en/document-library/product-certification-consultations/special-condition-vtol`.

[4] Alfred Meilus. *Airline Efficiency and Air Passenger Trip Circuity Trend.* Report. Washington, DC: Federal Aviation Administration, 2014.

[5] Jeff Holden and Nikhil Goel. *Fast-Forwarding to a Future of On-Demand Urban Air Transportation.* Report. Uber Elevate, 2016.

[6] Chi Yang et al. 'Challenges and key requirements of batteries for electric vertical takeoff and landing aircraft'. In: *Joule* 5.8 (2021). Accessed: 2024-06-12, pp. 1884–1900. DOI: `10.1016/j.joule.2021.05.011`. URL: `https://www.cell.com/joule/fulltext/S2542-4351(21)00205-1?_returnURL=https%3A%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS2542435121002051%3Fshowall%3Dtrue`.

[7] European Union Aviation Safety Agency. *EASA review of standard passenger weights 2022 shows no significant change.* Accessed: 2024-06-12. 2022. URL: `https://www.easa.europa.eu/en/newsroom-and-events/news/easa-review-standard-passenger-weights-2022-shows-no-significant-change`.

[8] Daniel P. Raymer. *Aircraft Design: A Conceptual Approach.* 2nd. AIAA Education Series. American Institute of Aeronautics and Astronautics, 1992. ISBN: 9781600860156.

[9] Snorri Gudmundsson. *General Aviation Aircraft Design: Applied Methods and Procedures.* Accessed: 2024-08-08. Butterworth-Heinemann, 2013. ISBN: 9780123973085. URL: `https://learning.oreilly.com/library/view/general-aviation-aircraft/9780123973085/xhtml/CHP006.html#CESECTITLE0100`.

[10] S. Kaneko and J. R. R. A. Martins. 'Simultaneous Optimization of Conceptual Design and Takeoff Trajectory of a Lift-Plus-Cruise UAV'. In: *Unpublished Manuscript or Conference Paper* (2023). Available upon request or in press.

[11] MT-Propeller. *Technical Data Sheet: MTV-11 Propeller.* `https://www.mt-propeller.com/pdf/TCDS_EASA/MTV-11.pdf`. Accessed: August 10, 2024. n.d.

[12] MT-Propeller. *Technical Data Sheet: MTV-15 Propeller.* `https://www.mt-propeller.com/pdf/TCDS_EASA/MTV-15.pdf`. Accessed: August 10, 2024. n.d.

[13] MT-Propeller. *Technical Data Sheet: MTV-21 Propeller.* `https://www.mt-propeller.com/pdf/TCDS_EASA/MTV-21.pdf`. Accessed: August 10, 2024. n.d.

[14] MT-Propeller. *Technical Data Sheet: MTV-17 Propeller.* `https://www.mt-propeller.com/pdf/TCDS_EASA/MTV-17.pdf`. Accessed: August 10, 2024. n.d.

[15] MT-Propeller. *Technical Data Sheet: MTV-20 Propeller.* `https://www.mt-propeller.com/pdf/TCDS_EASA/MTV-20.pdf`. Accessed: August 10, 2024. n.d.

[16] European Union Aviation Safety Agency (EASA). *Easy Access Rules for Large Aeroplanes (CS-25)-CS 25.925.* Accessed: August 10, 2024. 2024. URL: `https://www.easa.europa.eu/en/document-library/easy-access-rules/online-publications/easy-access-rules-large-aeroplanes-cs-25?kw=propeller%20clearance`.

[17] Federal Aviation Administration. *14 CFR § 25.925 - Propeller Clearance.* Accessed: August 10, 2024. 2024. URL: https://www.ecfr.gov/current/title-14/chapter-I/subchapter-C/part-25/subpart-E/subject-group-ECFR3db216ad9d52259/section-25.925.

[18] B. Govindarajan and A. Sridharan. 'Conceptual Sizing of Vertical Lift Package Delivery Platforms'. In: *Journal of Aircraft* 57.6 (2020), p. 1170. DOI: 10.2514/1.c035805.

[19] Christopher Silva et al. *VTOL Urban Air Mobility Concept Vehicles for Technology Development.* Report. NASA, 2018.

[20] Tecnam. *P2006T Information Manual.* Costruzioni Aeronautiche Tecnam S.p.A. 2015. URL: https://tecnam.com/wp-content/uploads/2015/05/P2006T-14-w.pdf.

[21] Jiechao Zhang et al. 'Conceptual Design and System Level Analysis of Tilt-Duct eVTOL Aircraft'. In: *The Proceedings of the 2021 Asia-Pacific International Symposium on Aerospace Technology (APISAT 2021), Volume 1.* Ed. by Sangchul Lee et al. Singapore: Springer Nature Singapore, 2023, pp. 189–198.

[22] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization.* Cambridge University Press, 2021. ISBN: 9781108833417. URL: https://mdobook.github.io.

[23] S. S. Chauhan and J. R. R. A. Martins. 'Tilt-Wing eVTOL Takeoff Trajectory Optimization'. In: *Journal of Aircraft* 57.1 (2019), p. 93. DOI: 10.2514/1.c035476.

[24] Airfoil Tools. *NACA 0012 Airfoil Polars.* http://airfoiltools.com/polar/details?polar=xf-n0012-il-200000. Accessed: 2024-07-16. 2024.

[25] Airfoil Tools. *NACA 2412 Airfoil Polars.* http://airfoiltools.com/polar/details?polar=xf-naca2412-il-200000. Accessed: 2024-07-16. 2024.

[26] Alessandro Bacchini et al. 'Impact of lift propeller drag on the performance of eVTOL lift+cruise aircraft'. In: *Aerospace Science and Technology* 109 (2021), p. 106429. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2020.106429. URL: https://www.sciencedirect.com/science/article/pii/S1270963820311111.

[27] LibreTexts Engineering. *ISA Equations.* https://eng.libretexts.org/Bookshelves/Aerospace_Engineering/Fundamentals_of_Aerospace_Engineering_(Arnedo)/02%3A_Generalities/2.03%3A_Standard_atmosphere/2.3.03%3A_ISA_equations. Accessed: 2024-06-12. 2024. URL: https://eng.libretexts.org/Bookshelves/Aerospace_Engineering/Fundamentals_of_Aerospace_Engineering_(Arnedo)/02%5C%3A_Generalities/2.03%5C%3A_Standard_atmosphere/2.3.03%5C%3A_ISA_equations.

[28] C Rotaru and M Todorov. *Helicopter Flight Physics.* InTech, 2018. URL: http://dx.doi.org/10.5772/intechopen.71516.

[29] James F. Marchman III. 'Altitude Change: Climb and Glide'. In: *Aerodynamics and Aircraft Performance, 3rd edition.* James F. Marchman III in association with the University Libraries at Virginia Tech, 2021. Chap. 5. DOI: http://hdl.handle.net/10919/96525. URL: http://hdl.handle.net/10919/96525.

[30] Y. Mihara et al. 'Cost Analysis of eVTOL Configuration Design for an Air Ambulance System in Japan'. In: *Journal of Advanced Transportation* 2021 (2021). Available online, Article ID 8821234. DOI: 10.1155/2021/8821234.

[31] Michael J. Duffy et al. 'A Study in Reducing the Cost of Vertical Flight with Electric Propulsion'. In: *American Institute of Aeronautics and Astronautics Conference.* Accessed: 2024-08-13. The Boeing Company. 2017. URL: https://www.researchgate.net/publication/318235979.

[32] Chris Wetherell. *Let's Talk About the Panasonic NCR18650B.* Accessed: 2024-08-13. 2018. URL: https://turtleherding.com/wp-content/uploads/2018/06/Lets-talk-about-the-Panasonic-NCR18650B.pdf.

[33] *Vertiports: Prototype Technical Specifications for the Design of VFR Vertiports for Operation with Manned VTOL-Capable Aircraft Certified in the Enhanced Category (PTS-VPT-DSN).* Report. EASA European Aviation Safety Agency, Mar. 2022.

[34] H. B. Faulkner. *The ATA-67 Formula for Direct Operating cost.* Tech. rep. NASA-TM-X-58083. NASA Technical Memorandum. NASA, May 1973. URL: https://ntrs.nasa.gov/api/citations/19730024119/downloads/19730024119.pdf.

[35] DFS Deutsche Flugsicherung. *Charges - Legal Framework.* Accessed: 2024-06-15. 2024. URL: https://www.dfs.de/homepage/en/air-traffic-control/legal-framework/charges/.

[36] Economic Research Institute (ERI). *Helicopter Pilot Salary in Germany.* Accessed: 2024-06-15. 2024. URL: https://www.erieri.com/salary/job/helicopter-pilot/germany#:~:text=The%20average%20salary%20range%20for,conducted%20and%20researched%20by%20ERI..

[37] Investopedia. *Annuity Method of Depreciation.* Accessed: 2024-06-15. 2023. URL: https://www.investopedia.com/terms/a/annuity-method-of-depreciation.asp#:~:text=The%20annuity%20method%20of%20depreciation%20is%20a%20process%20used%20to,least%20constant)%20rate%20of%20return..

[38] Valor International. *Orders for Eve's Flying Car Total $8.3bn.* Accessed: 2024-06-15. 2023. URL: https://valorinternational.globo.com/business/news/2023/05/02/orders-for-eves-flying-car-total-83bn.ghtml.

[39] International Civil Aviation Organization (ICAO). *Airlines Operating Costs and Productivity.* Accessed: 2024-06-16. 2017. URL: https://www.icao.int/mid/documents/2017/aviation%20data%20and%20analysis%20seminar/ppt3%20-%20airlines%20operating%20costs%20and%20productivity.pdf.

[40] *Taxi Fare Calculator London.* https://www.bettertaxi.com/taxi-fare-calculator/london/. Accessed: 2024-08-19.

[41] Michael D. Patterson, Kevin R. Antcliff and Lee W. Kohlman. *A Proposed Approach to Studying Urban Air Mobility Missions Including an Initial Exploration of Mission Requirements.* NASA Technical Report. 2018.

[42] International Energy Agency (IEA). *Germany - Countries Regions.* https://www.iea.org/countries/germany. Accessed: 2024-08-29. 2024.

[43] Bjorn Fehrm. *Bjorn's Corner: Sustainable Air Transport. Part 43. eVTOL IFR range.* Accessed: 2024-08-29. 2022. URL: https://leehamnews.com/2022/10/28/bjorns-corner-sustainable-air-transport-part-43-evtol-ifr-range/.

[44] F. Arshad et al. 'Life Cycle Assessment of Lithium-ion Batteries: A Critical Review'. In: *Resources, Conservation and Recycling* 180 (2022). DOI: 10.1016/j.resconrec.2022.106164.

A Appendix

A.1 Airfoil Data

NACA 0012 Data according [24].

α	C_L	C_D	C_{Dp}	C_M	Top_Xtr	Bot_Xtr
0.000	0.0000	0.01020	0.00520	0.0000	0.9046	0.9047
0.250	0.0350	0.01022	0.00521	-0.0004	0.8886	0.9182
0.500	0.0700	0.01025	0.00522	-0.0009	0.8696	0.9304
0.750	0.1048	0.01028	0.00521	-0.0015	0.8482	0.9424
1.000	0.1454	0.01034	0.00519	-0.0033	0.8255	0.9510
1.250	0.1861	0.01040	0.00518	-0.0054	0.7978	0.9593
1.500	0.2255	0.01048	0.00517	-0.0072	0.7684	0.9686
1.750	0.2684	0.01056	0.00512	-0.0100	0.7357	0.9756
2.000	0.3085	0.01066	0.00509	-0.0122	0.7012	0.9843
2.250	0.3513	0.01075	0.00504	-0.0152	0.6637	0.9915
2.500	0.3942	0.01085	0.00497	-0.0183	0.6249	0.9979
2.750	0.4235	0.01094	0.00492	-0.0188	0.5886	1.0000
3.000	0.4462	0.01104	0.00489	-0.0180	0.5545	1.0000
3.250	0.4689	0.01118	0.00489	-0.0172	0.5202	1.0000
3.500	0.4915	0.01134	0.00492	-0.0163	0.4857	1.0000
3.750	0.5138	0.01153	0.00498	-0.0155	0.4509	1.0000
4.000	0.5357	0.01176	0.00507	-0.0145	0.4160	1.0000
4.250	0.5571	0.01203	0.00521	-0.0134	0.3810	1.0000
4.500	0.5782	0.01234	0.00536	-0.0123	0.3456	1.0000
4.750	0.5990	0.01268	0.00556	-0.0111	0.3101	1.0000
5.000	0.6195	0.01306	0.00580	-0.0099	0.2744	1.0000
5.250	0.6395	0.01350	0.00609	-0.0086	0.2398	1.0000
5.500	0.6591	0.01400	0.00643	-0.0072	0.2077	1.0000
5.750	0.6784	0.01455	0.00683	-0.0058	0.1796	1.0000
6.000	0.6971	0.01516	0.00729	-0.0043	0.1563	1.0000
6.250	0.7161	0.01577	0.00781	-0.0028	0.1374	1.0000
6.500	0.7351	0.01641	0.00837	-0.0013	0.1223	1.0000
6.750	0.7535	0.01712	0.00900	0.0003	0.1104	1.0000
7.000	0.7725	0.01776	0.00961	0.0018	0.1005	1.0000
7.250	0.7919	0.01844	0.01031	0.0033	0.0925	1.0000
7.500	0.8096	0.01936	0.01114	0.0049	0.0858	1.0000
7.750	0.8300	0.01995	0.01181	0.0062	0.0799	1.0000
8.000	0.8481	0.02112	0.01285	0.0077	0.0749	1.0000
8.250	0.8690	0.02173	0.01362	0.0089	0.0707	1.0000
8.500	0.8890	0.02251	0.01438	0.0101	0.0667	1.0000
8.750	0.9087	0.02387	0.01574	0.0112	0.0635	1.0000
9.000	0.9291	0.02472	0.01674	0.0124	0.0604	1.0000
10.000	1.0067	0.02966	0.02206	0.0168	0.0511	1.0000
10.250	1.0251	0.03077	0.02323	0.0179	0.0492	1.0000
10.500	1.0445	0.03262	0.02503	0.0185	0.0474	1.0000
10.750	1.0571	0.03445	0.02718	0.0203	0.0462	1.0000
11.000	1.0681	0.03636	0.02942	0.0222	0.0451	1.0000
11.250	1.0773	0.03842	0.03177	0.0241	0.0441	1.0000
11.500	1.0860	0.04035	0.03390	0.0258	0.0430	1.0000
11.750	1.0969	0.04194	0.03558	0.0272	0.0419	1.0000
12.000	1.1068	0.04366	0.03735	0.0286	0.0410	1.0000
12.250	1.1101	0.04727	0.04108	0.0298	0.0401	1.0000
12.500	1.0959	0.04960	0.04372	0.0334	0.0399	1.0000
12.750	1.0790	0.05258	0.04700	0.0360	0.0397	1.0000
13.000	1.0599	0.05626	0.05095	0.0372	0.0396	1.0000
13.250	1.0385	0.06069	0.05564	0.0369	0.0396	1.0000
13.500	1.0150	0.06603	0.06120	0.0352	0.0396	1.0000
13.750	0.9898	0.07237	0.06774	0.0319	0.0398	1.0000
14.000	0.9632	0.07976	0.07532	0.0275	0.0399	1.0000

Table 12: XFOIL Calculated Polar for NACA 0012 Airfoil

NACA 2412 Data according [25].

α	C_L	C_D	C_{Dp}	C_M	Top_Xtr	Bot_Xtr
0.000	0.2860	0.01001	0.00485	-0.0608	0.8392	0.9789
0.250	0.3359	0.00994	0.00469	-0.0650	0.8247	0.9917
0.500	0.3796	0.00985	0.00451	-0.0682	0.8082	1.0000
0.750	0.4025	0.00981	0.00436	-0.0672	0.7901	1.0000
1.000	0.4257	0.00980	0.00424	-0.0662	0.7719	1.0000
1.250	0.4491	0.00982	0.00415	-0.0652	0.7537	1.0000
1.500	0.4727	0.00988	0.00410	-0.0642	0.7356	1.0000
1.750	0.4959	0.00996	0.00409	-0.0631	0.7163	1.0000
2.000	0.5193	0.01005	0.00410	-0.0621	0.6973	1.0000
2.250	0.5428	0.01017	0.00413	-0.0611	0.6786	1.0000
2.500	0.5663	0.01030	0.00417	-0.0601	0.6604	1.0000
2.750	0.5899	0.01045	0.00424	-0.0592	0.6427	1.0000
3.000	0.6136	0.01062	0.00432	-0.0582	0.6255	1.0000
3.250	0.6373	0.01080	0.00443	-0.0572	0.6088	1.0000
3.500	0.6610	0.01099	0.00456	-0.0563	0.5924	1.0000
3.750	0.6847	0.01119	0.00470	-0.0554	0.5757	1.0000
4.000	0.7080	0.01141	0.00483	-0.0543	0.5577	1.0000
4.250	0.7311	0.01161	0.00498	-0.0533	0.5375	1.0000
4.500	0.7543	0.01182	0.00515	-0.0522	0.5178	1.0000
4.750	0.7778	0.01205	0.00534	-0.0513	0.4995	1.0000
5.000	0.8010	0.01229	0.00554	-0.0503	0.4800	1.0000
5.250	0.8238	0.01256	0.00572	-0.0493	0.4589	1.0000
5.500	0.8465	0.01279	0.00593	-0.0482	0.4349	1.0000
5.750	0.8687	0.01309	0.00616	-0.0471	0.4112	1.0000
6.000	0.8911	0.01338	0.00642	-0.0461	0.3854	1.0000
6.250	0.9129	0.01373	0.00670	-0.0450	0.3582	1.0000
6.500	0.9339	0.01415	0.00702	-0.0438	0.3278	1.0000
6.750	0.9541	0.01465	0.00740	-0.0425	0.2937	1.0000
7.000	0.9731	0.01526	0.00785	-0.0411	0.2573	1.0000
7.250	0.9908	0.01600	0.00841	-0.0396	0.2186	1.0000
7.500	1.0067	0.01693	0.00909	-0.0378	0.1782	1.0000
7.750	1.0218	0.01795	0.00991	-0.0360	0.1446	1.0000
8.000	1.0370	0.01896	0.01076	-0.0341	0.1225	1.0000
8.250	1.0523	0.01995	0.01166	-0.0323	0.1086	1.0000
8.500	1.0679	0.02091	0.01261	-0.0305	0.0995	1.0000
9.000	1.0982	0.02292	0.01462	-0.0268	0.0875	1.0000
9.250	1.1137	0.02387	0.01556	-0.0251	0.0830	1.0000
9.500	1.1282	0.02523	0.01687	-0.0234	0.0794	1.0000
9.750	1.1448	0.02608	0.01784	-0.0218	0.0764	1.0000
10.000	1.1606	0.02701	0.01883	-0.0201	0.0733	1.0000
10.250	1.1764	0.02814	0.01993	-0.0187	0.0705	1.0000
10.500	1.1960	0.02971	0.02152	-0.0179	0.0678	1.0000
10.750	1.2108	0.03066	0.02263	-0.0162	0.0656	1.0000
11.000	1.2253	0.03171	0.02378	-0.0146	0.0632	1.0000
11.250	1.2401	0.03283	0.02494	-0.0132	0.0610	1.0000
11.500	1.2632	0.03501	0.02707	-0.0133	0.0582	1.0000
11.750	1.2711	0.03605	0.02834	-0.0109	0.0568	1.0000
12.000	1.2809	0.03736	0.02983	-0.0090	0.0549	1.0000
12.250	1.2900	0.03870	0.03129	-0.0072	0.0530	1.0000
12.500	1.3004	0.04002	0.03266	-0.0058	0.0513	1.0000
12.750	1.3189	0.04267	0.03530	-0.0055	0.0491	1.0000
13.000	1.3172	0.04432	0.03723	-0.0029	0.0482	1.0000
13.250	1.3163	0.04631	0.03947	-0.0008	0.0469	1.0000
13.500	1.3164	0.04839	0.04175	0.0008	0.0456	1.0000
13.750	1.3172	0.05044	0.04393	0.0021	0.0443	1.0000
14.000	1.3198	0.05245	0.04601	0.0031	0.0432	1.0000
14.250	1.3283	0.05480	0.04836	0.0038	0.0419	1.0000
14.500	1.3192	0.05878	0.05258	0.0048	0.0411	1.0000
14.750	1.3035	0.06249	0.05658	0.0053	0.0407	1.0000
15.000	1.2862	0.06682	0.06119	0.0051	0.0403	1.0000

Table 13: XFOIL Calculated Polar for NACA 2412 Airfoil