UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Mission Planning Application Software for Solar Powered UAVs

(Versão corrigida após defesa de dissertação)

Pedro Rodrigues Nunes

Dissertação para obtenção do Grau de Mestre em
Engenharia Aeronáutica
(Ciclo de Estudos Integrado)

Orientador: Prof. Doutor Pedro Vieira Gamboa

Covilhã, fevereiro de 2020

Aos meus pais, Fernando e Assunção.

À minha irmã, Inês, e a todos os que permaneceram.

Acknowledgments

This thesis is the culmination of five wonderful years at University of Beira Interior, and the product of eight months of research and work. Good and bad times, I experienced them always with the support and company of my family, friends and teachers, and I am forever grateful for them, not only for this period of my life, but for the last five years.

I would like to thank my PhD advisor professor Pedro Vieira Gamboa, from the Department of Aerospace Sciences at UBI, for his readiness in supporting me whenever I had problems, and steering me in the right direction, all the while always letting me do my own work.

I would also like to thank my friends, old and new, for giving me comfort, fun and happiness throughout this time, while also motivating me to keep up the hard work.

Lastly, I would not have the chance to be at this stage without my parents, sister and rest of my family. They get the credit and respect, for my education, for my growth, and for letting me be the person I am now. A big thank you.

Resumo

A crescente procura por veículos aéreos não tripulados (UAV) para uso civil na última década tem atraído a atenção de investigadores e engenheiros um pouco por todo o mundo. É importante realçar que a sua desnecessidade de pilotagem manual é idealmente adequada à realização de missões “sujas”, perigosas, monótonas (longa autonomia) ou de grande escala (uso de “enxames” de UAVs) [1], contudo exige uma maior atenção ao desenvolvimento de tecnologias que permitam e facilitem o planeamento, operação e gestão destes veículos. Bastantes avanços têm sido feitos em UAVs movidos a energia solar, que prometem uma operação de baixo custo energético, silenciosa e limpa. Contudo, por mais que a energia solar seja livre e abundante, o presente custo, complexidade, eficiência dos sistemas de captação solar, do armazenamento e da tração usando energia elétrica, bem como a consequente necessidade de veículos de grande tamanho, restringe muito a aplicação extensiva destes veículos [2], para além das dificuldades acrescidas pela ausência de um piloto humano. Não obstante, esta dissertação abrange o desenvolvimento de um interface gráfico de utilizador (GUI) associado ao aperfeiçoamento de um software de planeamento de missões criado a partir de projetos passados, aliando a flexibilidade e rapidez à eficiência de planeamento da operação de UAVs solares. Para além de facilitar a introdução de dados necessários à otimização de uma rota predefinida, este interface permite exportar a rota otimizada para o programa *open-source* de estação de controlo de solo (GCS) “MissionPlanner” (MP) [3]. Para além disso, o software conjunto final foi também executado como parte de um teste exaustivo, provando as suas capacidades e limitações numa situação real de operação.

Palavras-chave

Veículos Aéreos Não-Tripulados, Planeamento de missões, Operação de UAVs solares, ArduPlane, ArduPilot, Autopiloto

Abstract

The growing demand for unmanned aerial vehicles (UAV) for dedicated civilian use over the last decade has attracted the attention of investigators and engineers all over the world. It is important to note that the non-necessity of manual piloting is ideally suited to the operation of dirty, dangerous, dull (long autonomy) or large scale missions (use of swarms of UAVs) [1], however it demands a greater level of attention to the development of technologies that allow and ease the planning, operation and management of such vehicles. A lot of improvement has been made in the development of solar-powered UAVs, which promise a low-energy cost, silent and clean operation. However, despite solar energy being free and abundant, among many the present cost, complexity, solar energy capture systems' efficiency, electric storage and traction efficiency, as well as the consequent requirement for large-size vehicles, greatly restricts the extensive use of these UAVs [2], besides the added difficulties from the absence of a human pilot. Nevertheless, the present work covers the development of a graphical user interface (GUI) associated to the improvement of a mission planning software created by past work, allying flexibility and quickness to the planning efficiency of solar UAV operations. Beyond facilitating the input of necessary data to the optimization of a pre-set route, this interface allows to export the optimized route to the open-source ground control station (GCS) program "MissionPlanner" (MP) [3]. In addition, as part of an exhaustive testing process, the final ensembled software was run several times, proving its capabilities and limitations in a real operational situation.

Keywords

Unmanned Aerial Vehicles, Mission Planning, Operation of solar UAVs, ArduPlane, ArduPilot, Autopilot

Contents

List of Figures

List of Tables

List of Acronyms

AEROG	Aeronautics and Astronautics Research Centre (UBI)
API	Application Programming Interface
CCTAE	Centro de Ciências e Tecnologias Aeronáuticas e Espaciais (IST)
CPU	Central Processing Unit
DLR	German Aerospace Centre
EADS	European Aeronautic Defence and Space Company (Astrium)
ENU	East-North-Up coordinates system
ERAST	Environmental Research Aircraft and Sensor Technology (NASA)
ESC	Electronic Speed Controller
ETHZ	Swiss Federal Institute of Technology Zurich
FAI	Fédération Aéronautique Internationale
FFSQP	Fortran Feasible Sequential Quadratic Programming
GCS	Ground Control Station
GEO	Geodetic coordinates system
GPL	General Public License
GUI	Graphical User Interface
HALE	High Altitude Long Endurance
HALSOL	High Altitude Solar
IDMEC	Instituto de Engenharia Mecânica (IST)
INEGI	Instituto de Ciência e Inovação em Engenharia Mecânica e Industrial
ISA	International Standard Atmosphere
IST	Instituto Superior Técnico
LAETA	Laboratório Associado de Energia, Transportes e Aeronáutica
LEEUAV	Long Endurance Electric Unmanned Aerial Vehicle
MDP	Markov Decision Process
MetPASS	Meteorology-aware Path Planning and Analysis software for Solar-powered UAVs (*Atlantiksolar*)
MMS	Mission Management System
MP	MissionPlanner (open-source software by Michael Oborne)
MPP	Mission Planner Program
MSL	Mean Sea Level
NASA	National Aeronautics and Space Administration (USA)
PV	Photovoltaic
RC	Radio-controlled
R&D	Research and Development
RPAS	Remotely Piloted Aircraft Systems

SQP	Sequential Quadratic Programming
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UBI	Universidade da Beira Interior
URL	Uniform Resource Locator (web address)
UTC	Coordinated Universal Time

Nomenclature

ROMAN SYMBOLS

$\bar{A}$,$\overline{B}$,$\bar{C}$,$\overline{D}$,$\bar{E}$	Coefficient vectors
C_{batt}	Battery capacity $[Ah]$
C_{cell}	Cell capacity $[Ah]$
C_D	Drag coefficient
C_F	Fuselage skin friction coefficient
C_L	Lift coefficient
$C_{L_{max}}$	Maximum lift coefficient
$C_{L_{takeoff}}$	Take-off lift coefficient
C_p	Power coefficient
$C_{p,0}$	Power coefficient at null advance ratio
d	Propeller diameter $[m]$
d_n	Day of the year
E	Mission consumed energy $[J]$
E_{left}	Battery energy left $[J]$
E_{ref}	Battery energy left constraint reference value
$E_{solar\|total}$	Total solar energy harvested by photovoltaic panels $[J]$
$fore_{final}$	Mission final weather forecast hour $[h]$
$fore_{init}$	Mission initial weather forecast hour $[h]$
$fore_{now}$	Current weather forecast hour $[h]$
H	Hour of the day $[h]$
h	Altitude $[m]$
h_{ref}	Reference altitude $[m]$
I	Input current $[A]$
I_0	No load current $[A]$
$I_{batt_{max}}$	Maximum battery current $[A]$
$I_{cell_{max}}$	Maximum cell current $[A]$
I_{eff}	Effective input current $[A]$
I_{max}	Maximum input current $[A]$
I_{ref}	Reference input current $[A]$
J	Solar irradiation $[W/m^2]$
J_{prop}	Propeller advance ratio

K_C	Clear sky index $[\%]$
K_t	Motor torque constant
K_v	Motor speed constant
Lat	Latitude coordinate $[deg]$
Lon	Longitude coordinate $[deg]$
m_{batt}	Battery mass $[kg]$
m_{cell}	Single cell mass $[kg]$
m_{eng}	Mass of engine/motor $[kg]$
m_{prop}	Mass of propeller $[kg]$
N	Propeller/Engine speed $[rpm]$
N_0	Minimum engine speed at idle throttle $[rpm]$
N_{max}	Maximum engine speed $[rpm]$
n	Propeller/Engine speed $[rps]$
n_{blades}	Number of propeller blades
$n_{cells,parallel}$	Number of battery cells in parallel
$n_{cells,series}$	Number of battery cells in series
n_{eng}	Number of engines/motors
$n_{weather}$	Number of weather files
p	Propeller pitch $[m]$
P_{eff}	Effective power $[W]$
P_{ele}	Electric power $[W]$
P_{max}	Maximum engine power $[W]$
P_{ref}	Required power reference value $[W]$
P_{req}	Required power $[W]$
P_{shaft}	Motor power at the shaft $[W]$
P_{sys}	Systems power $[W]$
$P_{sys,eng}$	Engine systems power $[W]$
P_T	Total electric power $[W]$
Q_{Clouds_i}	Cloud index $[\%]$
Q_{ele_i}	Elevation value $[m]$
$Q_{Elevation}$	Elevation data
Q_{lat_i}	Latitude coordinates of the elevation/weather map $[deg]$
Q_{lon_i}	Longitude coordinates of the elevation/weather map $[deg]$
Q_{motor}	Motor torque at the shaft $[Nm]$
Q_{Temp_i}	Temperature $[K]$
$Q_{Weather}$	Weather data

$Q_{winddirection_i}$	Wind direction $[deg]$
$Q_{windspeed_i}$	Wind speed $[m/s]$
R	Electric resistance $[\Omega]$
R_{cell}	Cell internal resistance $[\Omega]$
R_{ESC}	Electronic speed controller resistance $[\Omega]$
R_{motor}	Motor internal resistance $[\Omega]$
r_{gear}	Gearbox ratio
R_U	Battery resistance $[\Omega]$
S	Wing reference area $[m^2]$
S_F	Fuselage cross-section area $[m^2]$
sfc	Specific fuel consumption $[kg/Ws]$
S_{PV}	Photovoltaic panels area $[m^2]$
S_{Wet}	Fuselage wetted area $[m^2]$
t	Time $[s]$
t_{final_h}	Mission final hour $[h]$
t_{init}	Internal mission initial hour $[h]$
$t_{init_{day}}$	Mission date day
t_{init_h}	Mission date initial hour $[h]$
$t_{init_{year}}$	Mission date year
U	Input voltage $[V]$
U_0	No load voltage $[V]$
u_{batt}	Battery specific energy $[Wh/kg]$
U_{batt}	Nominal battery voltage $[V]$
$U_{batt_{max}}$	Maximum battery voltage $[V]$
$U_{batt_{min}}$	Minimum battery voltage $[V]$
U_{cell}	Nominal cell voltage $[V]$
$U_{cell_{max}}$	Maximum cell voltage $[V]$
$U_{cell_{min}}$	Minimum cell voltage $[V]$
U_{eff}	Effective input voltage $[V]$
U_{max}	Maximum input voltage $[V]$
V	Aircraft linear velocity $[m/s]$

GREEK SYMBOLS

δ_{limit}	Engine/Motor throttle limit
δ_{set}	Engine/Motor throttle setting
η_{gear}	Gearbox efficiency

η_{prop}	Propeller efficiency
η_{PV}	Photovoltaic panels efficiency
θ_i	Angle of incidence of the sun $[deg]$
λ	Longitude $[deg]$
ρ	Air relative density $[kg.\,m^{-3}]$
φ	Latitude $[deg]$

Chapter 1

Introduction

1.1 Contextualization

The present work comes in line with the University of Beira Interior (UBI) and Instituto Superior Técnico (IST) recent years' research and development of a solar-powered Long Endurance Electric UAV (LEEUAV). As many as 8 authors' thesis were considered as a base of research accomplished so far, and on which this thesis builds on.

The LEEUAV is a particular type of unmanned aerial vehicle (UAV) that was especially designed to fly uninterruptedly for at least 8 hours during the Spring or Autumn equinoxes, while also being able to carry up to 1kg of payload, to take-off and land in a short distance, to climb 1000 meters in 10 minutes, and to descend without power [2]. As an object of research and development (R&D), the LEEUAV project started out as a consortium involving UBI and IST, and whose members include the "Energy, Transportation and Aeronautics Associated Laboratory" (LAETA), namely the "Centre of Aeronautics and Space Sciences and Technologies" (CCTAE), the "Aeronautics and Astronautics Research Centre" (AEROG), the Institute of Mechanical Engineering (IDMEC) and the Institute of Science and Innovation in Mechanical and Industrial Engineering (INEGI). In brief, the line of work started in 2014 with the successful conceptual and preliminary design, and development testing of a newly built prototype, made by Cândido [4].

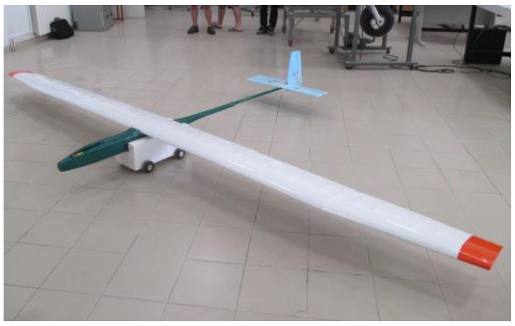


Figure 1 – First LEEUAV prototype [4]

In 2015, a solar component for the electric propulsion system was added to the wings of the prototype and validated by Sousa [5].

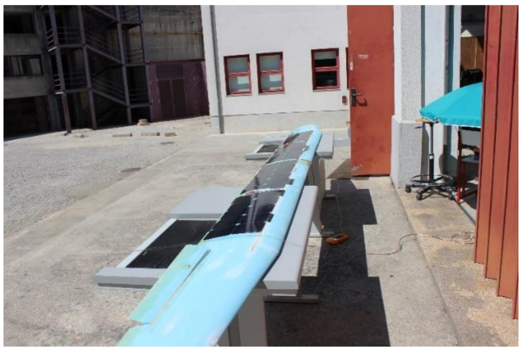


Figure 2 – Sousa's Solar module attached to the LEEUAV wing prototype [5]

In 2016, Duarte [6] specifically designed and built a new structural fuselage, tail boom and tail control surfaces for the LEEUAV, in order to withstand missions with payloads up to 1kg.

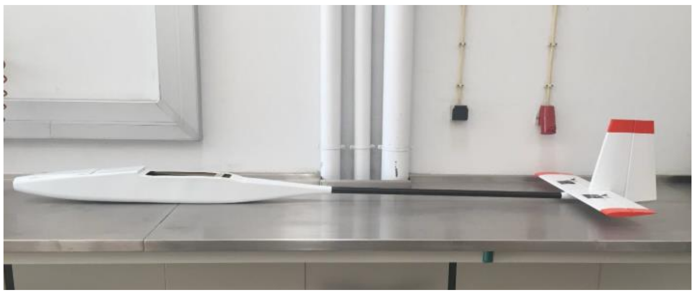


Figure 3 – Improved LEEUAV fuselage, tail boom and tail control surfaces made by Duarte [6]

Also, in the same year, Parada [7] developed a different conceptual and preliminary design for a new LEEUAV, choosing the V-tail airplane concept over the conventional tail configuration.

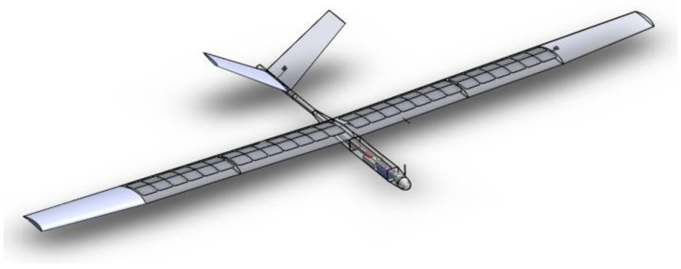


Figure 4 – Parada's new V-tail LEEUAV design [7]

In 2017, the conventional-tail LEEUAV with the latest fuselage, tail and wings was successfully assembled, tested and modified by Rodrigues [8], following integration of all electronic components, including the repositioning of the solar panels based on the work of Freitas [9]. At the same time, Moutinho designed a real-time energy management system to estimate the remaining flight time, based on the total energy balance, while also developing the data forwarding and receiving to and from the UAV, plus the visualization on the GCS "MissionPlanner" software [10].

Lastly, in 2019, the work on which this thesis is primarily based, Coelho created a Mission Planner Program (MPP) for a solar powered UAV, using the FFSQP algorithm and taking into account several models for mission planning: UAV performance constraints, ground elevation, atmospheric data and solar panels, sun and vehicle orientation [11].

1.2 Motivation

The reasons to the development of this thesis can be summarized into two main factors: planning for in-flight operational efficiency and flight safety.

Although UAV utilization is soaring all over the world, the bulk of development accomplished so far was made by several military institutions, much like in the earlier days of aviation. Though essential to the general development of the UAV as an airplane class, military development only ensures operations of the kind in the field, and normally prioritises assertion of power over profitability. An efficient planning before a mission ensures the success of a civilian operation, as this kind of operation must have economic viability. The creation and integration of a user-friendly and interactive graphical user interface (GUI) in a mission planner program makes the input process easy, fast and reliable to the user, where many different route optimizations can be made before a mission starts, and the user can choose which route is best to perform a specific mission.

The absence of a human pilot in autonomous UAVs dismisses many assurances normally present in the safe operation of manned aircraft. Human piloting, though many times limiting and disadvantageous, presents many advantages to a safe operation, namely the independent and quick decision making in the event of an emergency, the ability to break rules when needed, to think analytically and creatively to solve unexpected and never-before-occurred problems [12], things an autopilot would not yet be able to do in today's level of automation. Therefore, safety can be better achieved if the input of data into a mission planning program is correct and good environmental conditions are assured. Again, the development of a clear, understandable and reliable GUI is essential to a safe planned mission.

1.3 Objectives

This thesis follows the line of work of past LEEUAV-related works already mentioned before, in particular that of Coelho [11]. One could consider this thesis to be the second part of the development of a Mission Planner Program (MPP) (not to be confused with "MissionPlanner", an open-source software developed by Michael Oborne [3]). In essence, the main objectives of the present work can be summarized as follows:

- Improve the MPP developed by Coelho, adding new features or improving present ones;

- Ease the data input, output data retrieval and visual presentation of the MPP, by developing a graphical user interface (GUI), explaining its functioning structure, capacities and limitations;
- Perform various test runs of the ensembled Mission Planner Program, which includes Coelho's Mission Planner (also referred to as MPP or 'Optimizer' throughout this thesis) and the GUI, analyse and compare optimization results. This is done to validate the work developed thus far;
- Draw conclusions and suggestions for future work.

1.4 Thesis Outline

Including the present introductory chapter, this thesis is divided into 6 main chapters, which are briefly described below:

Chapter 1 covers the contextualization, motivation and objectives of the work to be done;

Chapter 2 focuses on a historical and state of the art review of the operation and mission path planning of solar-powered UAVs;

Chapter 3 covers the explanation of the Mission Planner Program by Coelho and the addition of new features, such as a new *mission initial hour* optimization design parameter and modifications to the minimization of energy flow. This chapter is crucial to understanding how the GUI will be added afterwards;

Chapter 4 explains the developing process of the GUI and its functioning structure, as well as a brief explanation of the system of exporting/importing data to/from the MissionPlanner freeware by Michael Oborne;

Chapter 5 describes how various optimization tests of a route from Terlamonte (Covilhã) to Castelo Branco and back were performed, according to a certain objective function, design parameters, and day of operation. Following that, a detailed analysis and comparation of results is made;

Chapter 6 provides the conclusions to draw from this work, as well as what future work could be made to improve planning and operation of solar-powered UAVs and the LEEUAV.

Chapter 2

State of the Art

2.1 History of solar-powered aircraft

Though aviation has been around since the 19th century, solar-powered aircraft operations have a very recent history. Electric solar propulsion has lagged behind conventional fossil fuels' propulsion because it is still mainly limited by the current level of development and efficiency of batteries and photovoltaic (PV) cells.

2.1.1 Early developments

The use of electric propulsion for aircraft was first introduced as early as 1884, with the hydrogen-filled dirigible *La France*. At the time, electric propulsion was superior to its rival, steam propulsion. However, the technology went undeveloped and abandoned for almost a century, in favour of gasoline propulsion [13]. The first officially recorded electric powered radio-controlled flight was achieved on 30 June 1957, with the RC model "Radio Queen", by UK Colonel H. J. Taplin. Also, in October 1957, German pioneer Fred Militky first achieved a successful flight with a free flight model. Photovoltaic (PV) cells first appeared in 1954 at Bell Telephone Laboratories [14]. However, the technology to harness solar light as energy to viably power an aircraft only really took off in the 1970s.

Starting in 1970, R. J. Boucher and his brother Roland Boucher from Astro Flight Inc. began their experiments with electric flight. On 4 November 1974, *Sunrise I*, the first ever solar-powered model aircraft, took off from a dry lake at Camp Irwin, California, USA. The *Sunrise I* had a wingspan of 9.75m and weighed 12.47kg. Following the partial destruction of the airplane in a sandstorm, an improved second model *Sunrise II* was built and flown on 12 September 1975, at Nellis AFB. Having the same wingspan as its predecessor, the *Sunrise II* weighed 10.21kg and its 14% efficient PV cells could produce 150W more power than the 450W of *Sunrise I*. These breakthroughs set the stage for later developments in solar aviation [13].

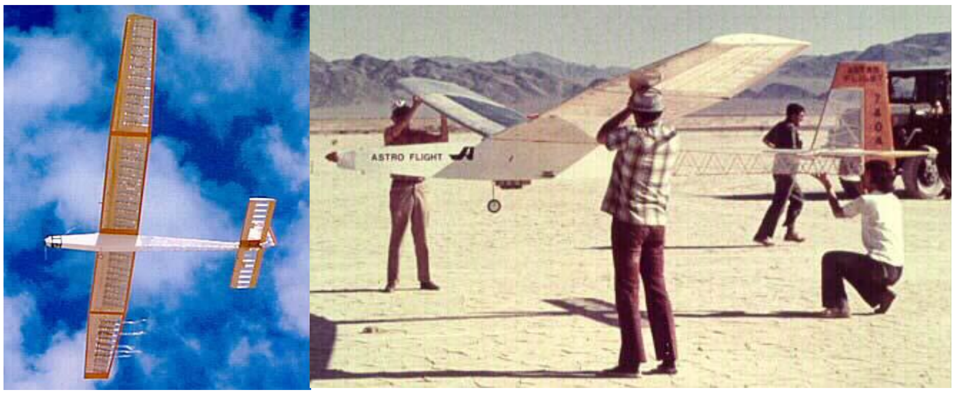


Figure 5 – Sunrise I (1974) and Sunrise II (1975)

At the same time in Germany, Helmut Bruss worked on a solar model airplane, though unsuccessful in achieving level flight. Later on 16 August 1976, his friend Fred Militky first flew his solar-powered *Solaris* airplane, completing three 150-second flights and reaching an altitude of 50 meters [15].

2.1.2 Manned solar-powered aircraft

The feasibility of solar power in model aircraft motivated its development on manned aircraft. David Williams and Fred To launched the *Solar One* in Hampshire UK, 19 December 1978. This aircraft was intended to be human powered and cross the English Channel, however it proved too heavy and was thus converted to solar power [13], [16]. On 29 April 1979, Larry Mauro flew the *Solar Riser*, an electric airplane capable to charge its battery with solar power, but not able to fly longer than ten minutes [13].

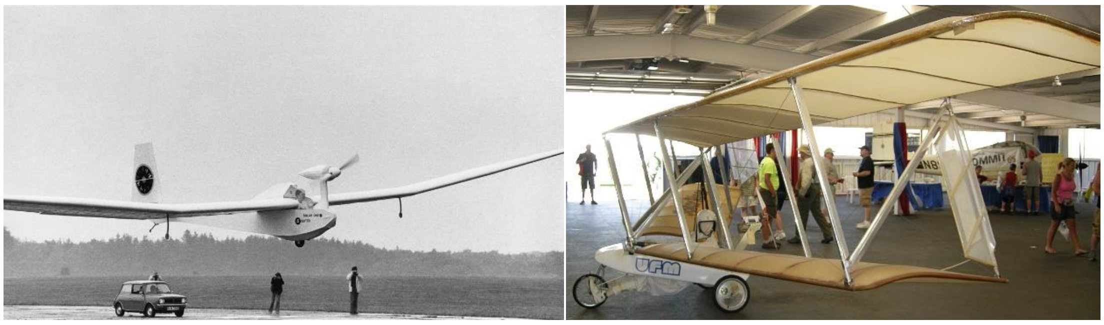


Figure 6 – Solar One (1978) and Solar Riser (1979)

Flying on the use of solar power alone without any batteries was first achieved on 18 May 1980 by the *Gossamer Penguin*, built by Dr. Paul B. MacCready and his company AeroVironment Inc. This was also the world's first piloted, solar-powered flight. Though the *Gossamer Penguin* was only able to reach a few meters in altitude, Dr. MacCready also built a new solar airplane, the *Solar Challenger*, which crossed the English Channel on 7 July 1981, covering 262.3km of distance with only solar energy as its power source and no onboard batteries [13].

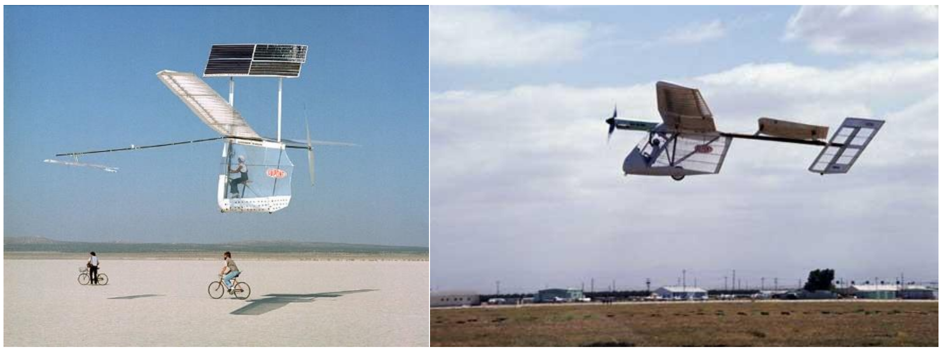


Figure 7 – Gossamer Penguin (1980) and Solar Challenger (1981)

Other important late 20th century manned solar aircraft developments include Günther Rochelt's *Solair I*, a 16m wingspan solar airplane which flew for 5 hours and 41 minutes in August 1983, Eric Raymond's *Sunseeker*, which crossed the continental USA in 21 solar-powered flights and 121 flight hours in August 1990 [17], and Prof. Rudolf Voit-Nitschmann's *Icaré 2* solar-powered motor glider which flew on 7 July 1996 [18].

In 2003, Bertrand Piccard initiated the *Solar Impulse* project in partnership with the Swiss Institute of Technology in Lausanne (EPFL), to develop long-endurance manned solar-powered aircraft that could circumnavigate the globe. The first prototype, *Solar Impulse 1*, first flew in December 2009, and performed a multi-stage flight across the USA in 2013. The second-generation aircraft, *Solar Impulse 2*, was built in 2014. It managed to circumnavigate the globe, starting in Abu Dhabi, 9 March 2015, flying West-to-East and returning on 26 July 2016 [19], [20].

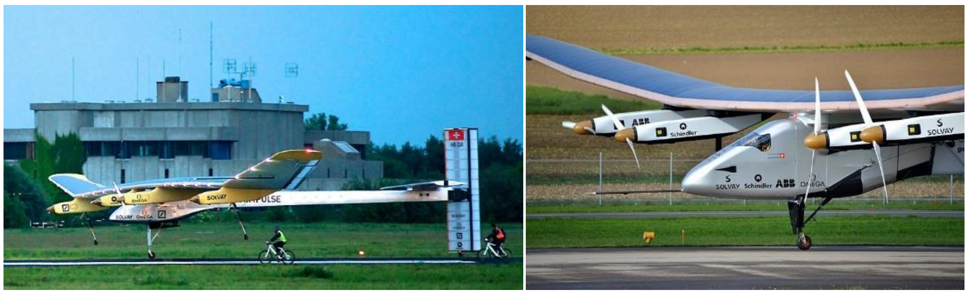


Figure 8 – Solar Impulse 1 (2011) and Solar Impulse 2 (2014)

Another solar-powered manned airplane, the two-seater *SolarStratos*, was built and flown by Raphaël Domjan on 5 May 2017, and is projected to be the first to reach the stratosphere in the very near future [21].

2.1.3 Long Endurance UAVs

Following the success of the *Solar Challenger*, AeroVironment Inc. received US government funding to secretly develop a remotely controlled High-Altitude Solar (HALSOL) LEEUAV, and built a prototype in 1983. However, contemporary solar PV and energy storage technologies were not mature enough to allow a HALSOL flight [22]. 10 years later, the prototype was flown again by NASA and was transferred to NASA's ERAST program in 1994, being renamed *Pathfinder*. In 1995, it surpassed *Solar Challenger*'s altitude record, reaching 15392m (50500ft), and two years later it reached 21802m (71530ft). NASA's *Pathfinder* was modified into the *Pathfinder Plus*, having a larger wingspan and new solar, aerodynamic, propulsion and system technologies. Two successor aircraft were followed in NASA's ERAST program: the *Centurion* and the *Helios*. The latter was intended to reach 30480m (100000ft) and fly non-stop for at least 24 hours. It flew at an altitude of 29261m (96000ft) for 40 minutes, was able to carry a payload up to 329kg. Unfortunately, it crashed into the Pacific Ocean before it could validate its 24-hour endurance goal [23].

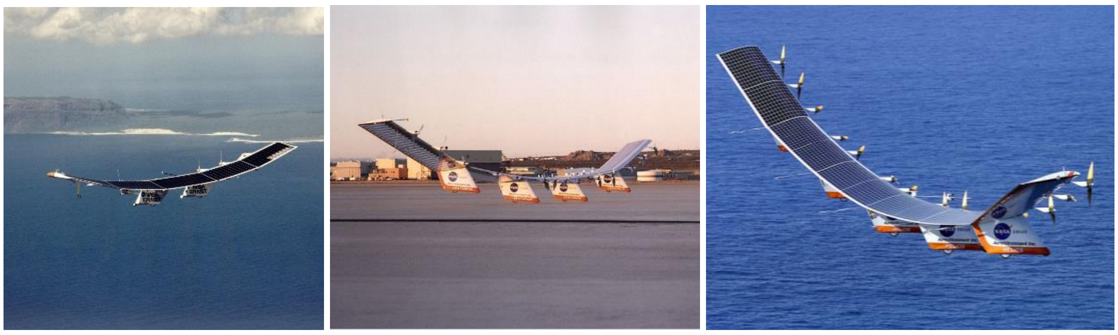


Figure 9 – NASA Pathfinder (1997), Centurion (1998) and Helios (2001)

Meanwhile in Europe, many other projects also started to appear to develop a High-Altitude Long Endurance (HALE) UAV, namely the *Solitair*, at DLR Institute of Flight Systems (1994-1998) [24], the *Heliplat*, developed by several European partners (2000-2003) and *Shampo*, developed by the Politecnico di Torino [25], [26].

On 22 April 2005, Alan Cocconi flew his *Solong* for 24 hours and 11 minutes using only solar power and wind thermals, finally validating the goal of eternal flight for the first time. Two months later, in the Colorado Desert CA, he managed to perform an even more ambitious flight, which lasted for a total of 48 hours and 16 minutes [17].

Also competing in the solar HALE platforms field, British Defence contractor QinetiQ flew a *Zephyr* aircraft on 10 September 2007 for a duration of 54 hours, also using solar power and wind thermals only. The aircraft was able to reach an altitude of 17786 meters, weighed 30kg and had a wingspan of 18 meters [27]. Later in 2008, the *Zephyr 6* performed an 82-hour flight at an altitude of 19000m, but did not set any records because of the absence of World Air Sports Federation (FAI) officials [28]. The *Zephyr* was also selected to carry the Mercator remote sensing system in 2005, which allows to perform forest fire monitoring, urban mapping, coastal monitoring, oil spill detection, etc.

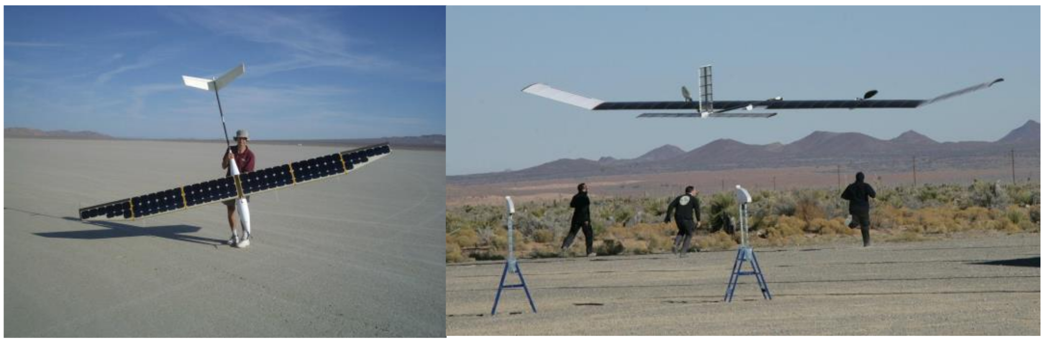


Figure 10 – Solong and Zephyr (2005)

The *Sky-Sailor* is another solar-powered LEEUAV project first designed in 2004 by the Swiss Institute of Technology in Lausanne (EPFL) to demonstrate HALE flight, this time for the scientific exploration of Mars. It was designed to be autonomous, to have its flight path planned beforehand and to be able to navigate using range sensors and vision only (as GPS is not available outside of Earth). The prototype's flight characteristics were successfully tested in June 2008, and it was the first ever to fly continuously for 27 hours without using altitude gain or thermal soaring [29].

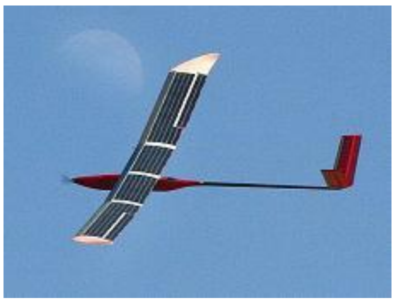


Figure 11 – Sky-Sailor prototype in flight (2008)

An example of recent developments in long endurance UAVs can be found in Titan Aerospace's *Solara 50*. This 50-meter wingspan autonomous UAV was designed to function as an atmospheric satellite operating at an altitude of 20km above any possible storms in the troposphere, and able to carry 31.75kg of telecom, reconnaissance, atmospheric sensors or other payloads, while operating as part of a fleet of other UAVs for as long as five years. Titan Aerospace was bought by Google in 2014, however the project was abandoned in 2016. The only built prototype got to fly for 4 minutes and 16 seconds before crashing after an in-flight structural failure on 1 May 2015 [30]-[32].

Another example of an autonomous solar LEEUAV is that of Facebook's *Aquila*. It was designed to provide communications service (mainly Internet connection) to remote areas in the world, operating between 18.3km and 27.4km of altitude. The only 42m-wingspan built prototype first flew in 2016 for an hour and 36 minutes, however the whole program was cancelled by Facebook in 2018, which favoured their partnership with other companies in the field [33]–[35].

Figure 12 – Solara 50 (concept design) and Aquila (prototype, 2016)

The *Zephyr* program, founded in 2003 by QinetiQ, is still in development today. Recently in the summer of 2018, the new *Zephyr S* reached an astonishing flight of 25 days 23 hours 57 minutes, doubling the record of 14 days set by its predecessor. This aircraft was designed to provide persistent surveillance in response to a natural or human-caused disaster, or to act as a telecommunications relay station [36].

Figure 13 – Zephyr S prototype before launch (2018)

2.2 Mission Planning

In order to better understand the motivation behind the making of this work, a comprehensive review of recent developments in UAV mission planning techniques and current knowledge must be undertaken.

As seen in the previous section, solar-powered aircraft have come a long way since the first solar models, and a lot can still be done to improve in regards to flight performance. In a way, the very development of the technology to harness the sunlight and convert it to energy is a big game changer in the operation of UAVs in particular, because it allows the vehicle to remain in operation for an indefinite amount of time, if energy storage so allows. With this, a particular range of applications have been improved in Unmanned Aerial Systems (UAS) operations, such as, among others [37]:

- continuous security surveillance;

- facilitation of communications and broadcast;
- monitoring of linear network infrastructure (e.g. railway tracks, power lines, pipelines);
- photography and cartographic survey;
- atmospheric research.

A lot more solar LEEUAV-related projects will, of course, be started in the very near future, providing ever more knowledge in the art of UAV operation. But most of solar flights in the previous decades were manually planned and performed using the best conditions possible, which is understandable for new experimental aircraft.

Therefore, new and effective mission planning and coordination techniques have also been developed to improve the operation of a solar-powered UAV, even when conditions are not favourable. The use of these techniques allows the optimization of routes, according to goals like: energy usage, travel time or distance minimization; obstacle/collision avoidance; surveillance/coverage time maximization; or endurance/autonomy maximization. Not only by decreasing the amount of consumed energy of the mission, but also by increasing the amount of energy obtained from solar power, the LEEUAV can be operated efficiently and use its capabilities to the fullest possible.

Based on the work of Klesh *et al.* [38], Dai *et al.* investigated a unit quaternion-based method used to design the optimal UAV trajectory with maximum sun exposure for solar UAVs. Considering a mission delimited by two boundary points with fixed flying time and constant speed, this work optimizes the route of the UAV by adjusting its attitude – pitch (γ), heading (ψ) and roll (ϕ) angles, subject to constraints – in a level and three-dimensional flight models, using a branch and bound (BNB) and nonlinear programming (NLP) approaches [39]. An example of one of the simulation results is displayed in Figure 14.

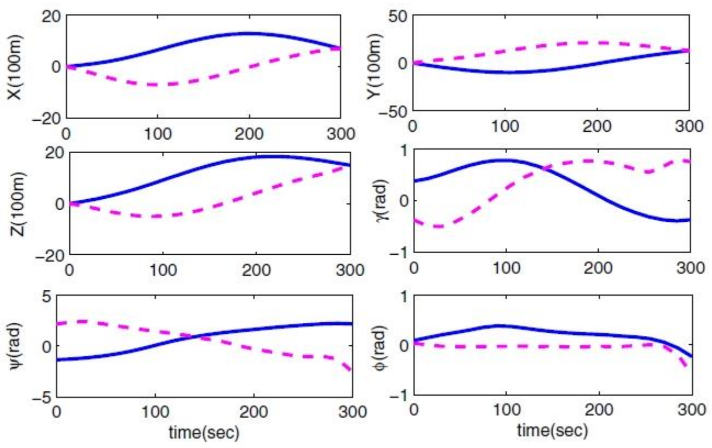


Figure 14 – Dai *et al.* simulation result example - time history of 3D flight state and control variables from BNB (solid line) and NLP (dash line) [39]

In 2016, Huang *et al.* developed an online method to obtain energy-optimal trajectories with the mission of ground target tracking. Although different from missions with no target tracking, this method presents a simple energy integrated model in two-dimensional space to calculate the instantaneous power, collect more energy and track the moving target using the optimization method of receding horizon control (RHC) with particle swarm optimization (PSO) [40]. Several numerical simulations demonstrated the feasibility and flexibility of this method, one of which is shown in Figure 15.

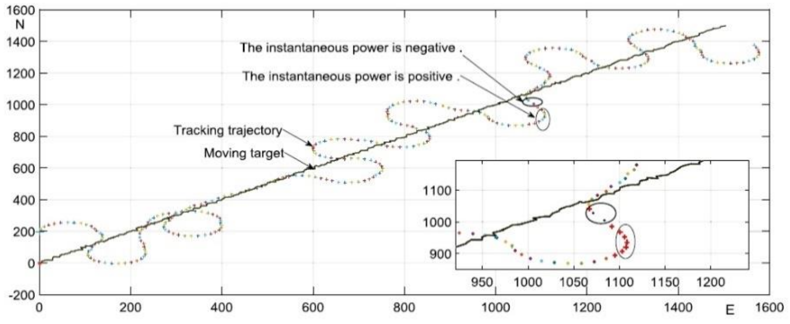


Figure 15 – Example of a 2D path energy-optimization for a mission of ground target tracking [40]

Later in 2019, Huang *et al.* developed a similar 3D path planning method for solar-powered UAVs, albeit intended for fixed target and solar tracking. Taking into consideration the UAV's motion and attitude, mission constraints, energy production and energy consumption, the loitering flight paths are planned on a virtual cylinder surface in 3D space, with the fixed target centre at its origin [41]. One of the test results of this method is shown in Figure 16.

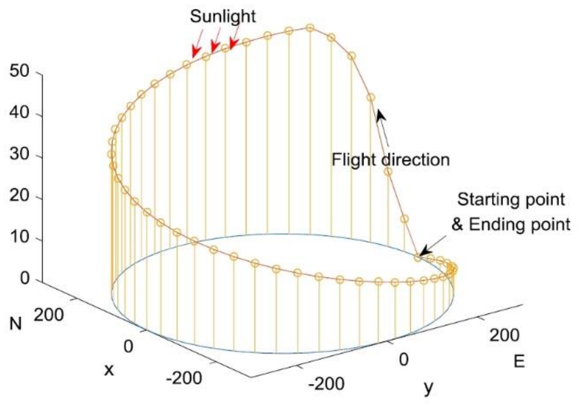


Figure 16 – Test result of fixed target tracking path of UAV in 3D space [41]

The work of Kiam *et al.*, presented in March 2017, demonstrates a multilateral quality mission-planning tool to increase the endurance of solar-powered LEEUAVs. The focus of the tool is set on very-long endurance missions, such as surveillance, using high-altitude pseudo satellites (HAPS fixed-wing UAV platforms) on certain locations of interest (LOI). It uses a highly automated mission management system (MMS) which produces an optimal plan subject to the specific application's requirements and multilateral constraints, i.e. mission, energy and safety constraints. This MMS adopts the hybrid architecture of a symbolic planner based on the hierarchical task-network (HTN), working with a Markov decision process (MDP) based policy generator to reduce the search space for a numerical path planner [42].

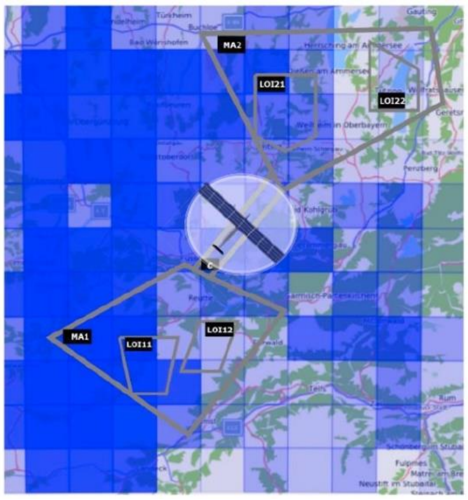


Figure 17 – Cloud coverage map containing mission areas (MA) and locations of interest (LOI) where the UAV will perform surveillance within a time frame, in the MMS application test [42]

A new solar LEEUAV dubbed *AtlantikSolar* by its developer, the Swiss Federal Institute of Technology at Zurich (ETHZ), has been developed since 2015 to fly autonomously in Low-Altitude Long Endurance (LALE) missions, for applications such as industrial and agricultural sensing and mapping, large-scale disaster relief support missions, meteorological surveys in remote areas and continuous border or maritime patrol. It was especially designed with efficient autonomous path planning and operation in mind [43], [44].

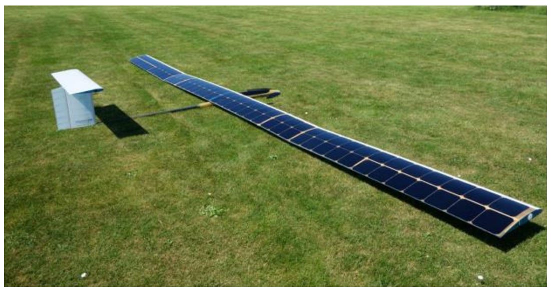


Figure 18 – The *AtlantikSolar* prototype (2015)

The software responsible for this autonomous path planning is the MetPASS – Meteorology-aware Path Planning and Analysis Software for Solar-powered UAVs. It consists of an optimization software that combines an A* algorithm, a dynamic programming point-to-point planner and a local scan path planner (a simple camera model), to yield cost-optimal aircraft paths. The cost function considers both safety and performance variables, such as terrain collision risk, system state (time since launch, battery state of charge, power consumption and generation) through a comprehensive energetic model, and meteorological data (thunderstorms, precipitation, humidity, 2D winds, gusts, sun radiation and clouds) through global weather models. Furthermore, the MetPASS was fully implemented with a GUI that allows an easy-to-use, detailed mission feasibility analysis, pre-flight planning and in-flight re-planning using updated weather data. Several real flight tests (a loitering 81-hour endurance flight in June 2015 and a series of Arctic glacier multi-goal survey flights in July 2017) were performed for the validation of the MetPASS. It was also validated through extensive testing for the planning of a 4000km crossing of the Atlantic Ocean, from Newfoundland to Portugal, the flight of which is yet to take place [45].

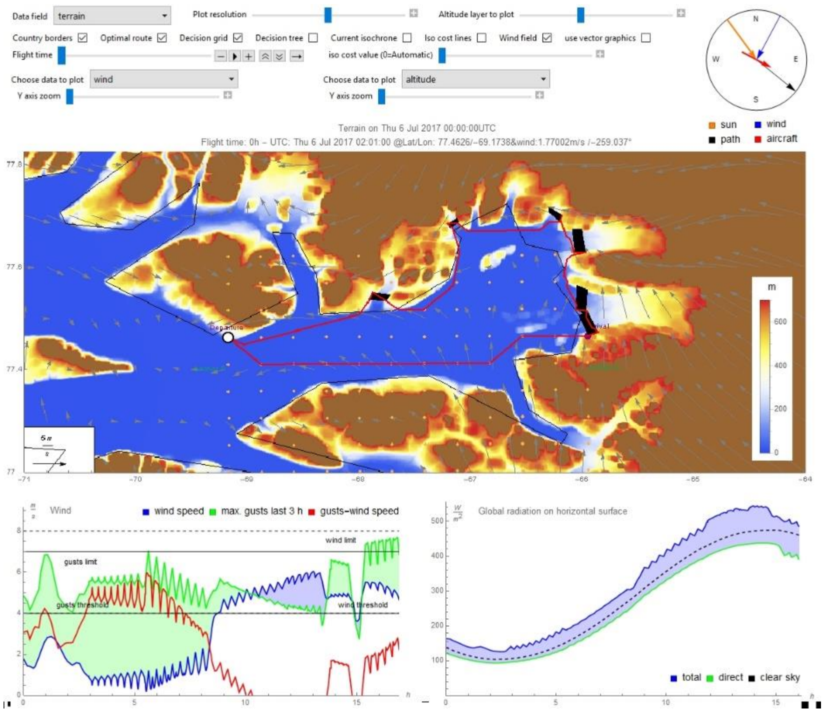

Data field terrain
Plot resolution
Altitude layer to plot
Country borders ✓ Optimal route ✓ Decision grid ✓ Decision tree ☐ Current isochrone ☐ Iso cost lines ☐ Wind field ✓ use vector graphics ☐
Flight time
iso cost value (0=Automatic)
Choose data to plot wind
Y axis zoom
Choose data to plot altitude
Y axis zoom
N
W
E
S
sun
wind
path
aircraft
Terrain on Thu 6 Jul 2017 00:00:00UTC
Flight time: 0h – UTC: Thu 6 Jul 2017 02:01:00 @Lat/Lon: 77.4626/−69.1738&wind:1.77002m/s /−259.037°
m
Wind
wind speed
max. gusts last 3 h
gusts−wind speed
wind limit
gusts limit
gusts threshold
wind threshold
Global radiation on horizontal surface
total
direct
clear sky

Figure 19 – GUI window of the *Atlantiksolar* path planning and analysis software MetPASS [45]

In 2018, Amorosi *et al.* developed an energy-efficient mission planning method intended for 5G network coverage in rural zones (using a swarm of non-solar-powered UAVs), by solving the "RURALPLAN" optimization problem, a variant of the unsplittable multicommodity flow problem defined on a multiperiod graph. Not only this method limits the amount of energy consumed above minimum battery level constraints, it also ensures the coverage of selected zones and determines sites where an UAV should land to recharge, considering the amount of energy provided by the PV panels and batteries installed at those ground sites [46]. Although not intended for the operation of solar-powered UAVs, future work on this method using an onboard solar component would ensure greater endurance performance, proving to be particularly useful at Winter days.

Another work, presented by Wu *et al.* in the same year, shows the development of a solar-powered UAV path planning framework aimed at urban environments. This framework addresses three main aspects: modify the Interfered Fluid Dynamic System (IFDS) to allow the UAV to avoid obstacles and respect dynamic constraints and energy model; resolve the path planning issue using and improving a novel intelligent optimization algorithm called Whale Optimization Algorithm (WOA), to overcome the drawback of local minima; and solve the accurate modelling problem of solar energy in urban environment, taking into account sunlight occlusions and solar power obtained by slant surfaces of the PV cells [47]. One of several path planning tests performed is shown in Figure 20.

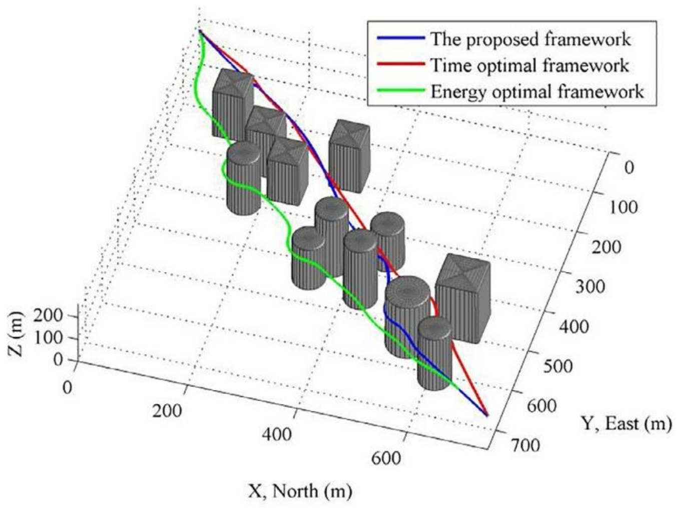


Figure 20 – 3D path planning example of a solar UAV in an urban environment. The "proposed framework" consists of a faster-converging optimization process [47]

Although not using the solar component, in 2019 Schellenberg *et al*. developed an on-board real time trajectory planning program (RTTP) for fixed-wing UAVs operating in extreme environments. This work, based on previous developments from the University of Bristol on long-range, high-altitude volcanic monitoring and ash-sampling, focuses on the optimization of routes using a genetic algorithm running on a Raspberry Pi 3 B+ single-board microcomputer. It includes obstacle, terrain and "no-go" zones avoidance, energy, altitude and climb/descent constraints and length minimization, as part of a cost function. Four successful RTTP-validation flight tests were performed in March-April, near Volcán de Fuego, Guatemala [48].

Although a reality, autonomous planning and piloting technology for fixed-wing UAVs only recently took off, and is more prevalent in the case of rotorcraft UAVs. Also in part due to on-going regulations, intuitive planning and operation by a human pilot is still a must in the operation of fixed-wing UAVs.

Some computer-based ground control station (GCS) software have been developed to provide a human pilot the ability to remotely plan and operate a UAV fitted with an autopilot, or even manually pilot the aircraft (as is the case of RPAS). Examples include the “QBase 3D Mission Control System”, a payware GCS created by Quantum Systems [49], open-source software “QGroundControl”, a GCS for MAVLink protocol created by the Dronecode project and funded by the Linux Foundation [50] and “MissionPlanner”, the free software created by Michael Oborne for the open-source autopilot project ArduPilot, used for the development of the Mission Optimization Interface of this thesis [3].

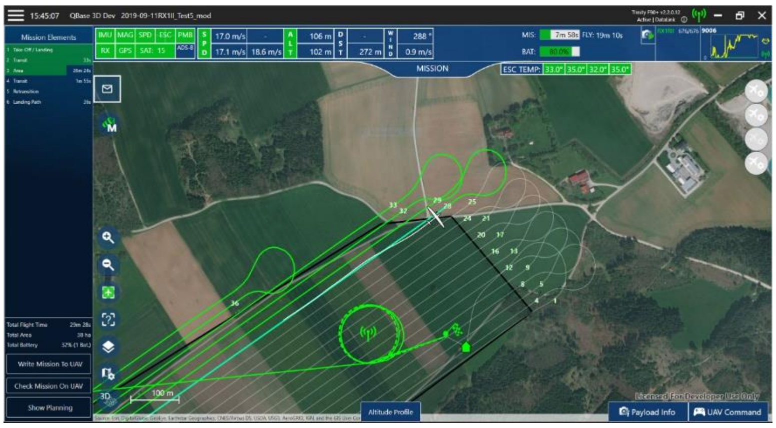


Figure 21 – An example of a QBase 3D route planning

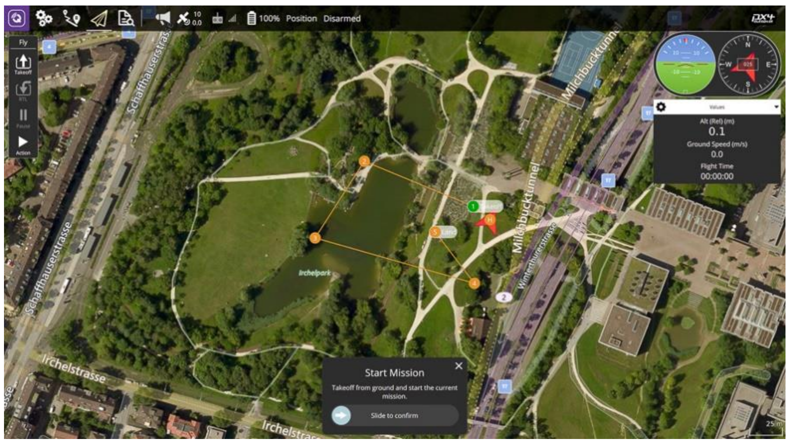
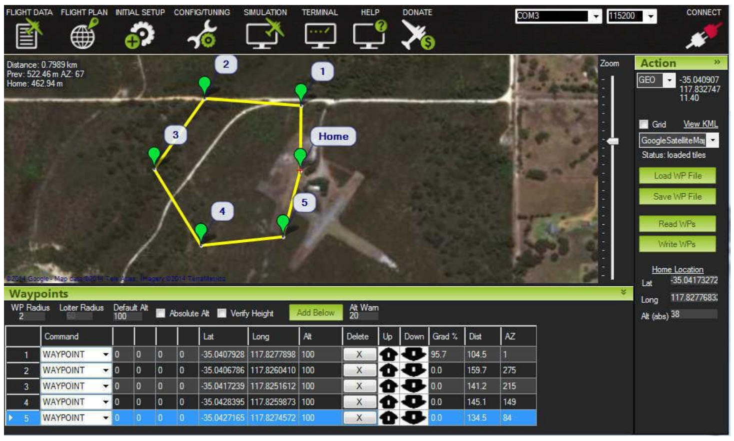


Figure 22 – QGroundControl mission planning example



Waypoints

WP Radius 2 | Loiter Radius | Default Alt 100 | Absolute Alt | Verify Height | Add Below | Alt Warn 20

	Command					Lat	Long	Alt	Delete	Up	Down	Grad %	Dist	AZ
1	WAYPOINT	0	0	0	0	-35.0407928	117.8277898	100	X			95.7	104.5	1
2	WAYPOINT	0	0	0	0	-35.0406786	117.8260410	100	X			0.0	159.7	275
3	WAYPOINT	0	0	0	0	-35.0417239	117.8251612	100	X			0.0	141.2	215
4	WAYPOINT	0	0	0	0	-35.0428395	117.8259873	100	X			0.0	145.1	149
5	WAYPOINT	0	0	0	0	-35.0427165	117.8274572	100	X			0.0	134.5	84

Figure 23 – Example of MissionPlanner route planning

While many mission planning and GCS software tools are available, it is the link between the creation of an optimized path, a human pilot and the autopilot system of the fixed-wing solar UAV that lacks the most development. Therefore, the GUI designed in the present work addresses this issue by providing a better link between the user and a computer-based Mission Planner Program that, using the FFSQP algorithm, optimizes a route based on time, energy or distance minimization. Thereafter, the GCS software “MissionPlanner” was chosen to be used, interfaced with the designed GUI, to send the route to the autopilot of the UAV and finally operate it.

Chapter 3

Mission Planner Program

3.1 Description and Functioning Structure

As part of his thesis, Coelho [11] created a mission planner program (MPP) capable of optimizing a pre-defined route based on the minimization of time, energy or distance (its objective function). Although the program allows for the optimization of routes for gasoline fuel-powered UAVs, it is mainly focused in optimization for solar-powered UAVs.

The whole program itself is built from several models that are basically independent Fortran code subroutines that return the needed input data into the main program. This main program code routine can then be run in two different modes: Analysis (which only calculates several parameters of the input route, such as time, energy usage/flow, distance, etc.) or Optimization (which actually changes the input route to optimize the objective function).

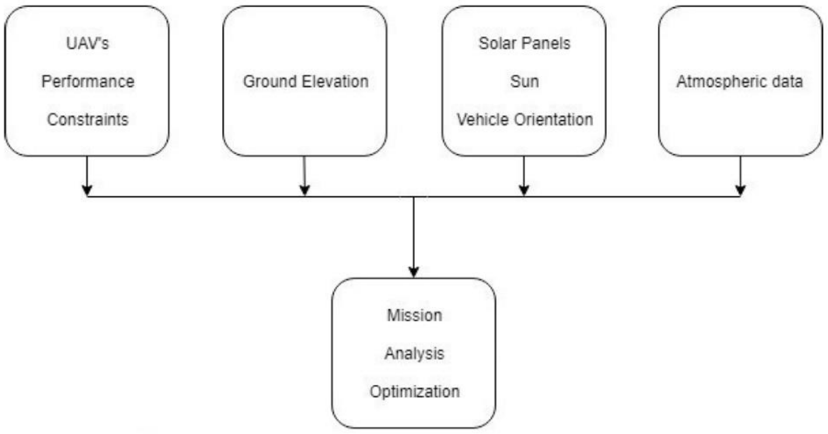


Figure 24 – The basic structure of the Mission Planner Program [11]

The four Fortran code models are, for the purpose of simplification, subdivided into 6 different sections (represented by folders in the “missioncode” directory of the program), where data is edited and stored in text files which are in turn loaded by the program while running:

- Aerodynamics – includes all data needed from geometry, as well as parameters that represent the drag and lift of the aircraft.

- Earth – includes all data needed from terrain elevation and atmospheric weather data. Only weather data may be manually input, while both can be downloaded from APIs (Application Programming Interfaces), which may or may not be free-for-use.
- Masses – includes all data related to the mass of the aircraft.
- Systems – includes all the systems present on the UAV, either active or not, as well as their respective mass.
- Propulsion – includes data related to the propulsion of the UAV. It is further subdivided into 3 subsections (text files): “Battery”, which defines the data related to the battery of the UAV; “Engine”, which defines the data related to the combustion engine(s) or electric motor(s) used; and “Propeller”, where data that represent the propeller(s) model to be used is present.
- Mission – includes data related to the route to be analysed/optimized, as well as parameters that define the optimization of the MPP. It is also further subdivided into 3 subsections (text files): “Waypoints”, which defines the route to be analysed/optimized; “Optimization”, where the parameters active or not for optimization are set, as well as other data that define the optimization process to be made, including the objective function; and “FFSQP”, a subsection that defines a particular algorithm used for the optimization (explained in section 3.1.5).

Several functionality aspects of the Mission Planner Program (MPP) are described in the following sections 3.1.1 through 3.1.5 below. Afterwards, section 3.2 is dedicated to features added to the MPP in the scope of this work. All analysis is made based on the work of Coelho [11].

3.1.1 Mission Analysis

Though the optimization mode can be deactivated by choice, every time the MPP is run a mission analysis must be made, even before an optimization. During the analysis mode the objective function, as well as several variable constraints initially specified by the user, are calculated depending on a list of design variables for each waypoint in the route.

There are 4 design variables for each waypoint – the coordinates chosen by the user, which can either be of the Geodetic (GEO) type (latitude and longitude) or the East-North-Up (ENU) type (x and y positions), altitude and airspeed.

The calculation of all relevant or necessary parameters is mainly obtained by the average of values between two waypoints. In other words, an effective analysis of segments is made. The results obtained are copied over to a text file (“mission_out.txt”) at the end of the analysis. The full list of parameters can be viewed in Annex A, and a better detailed summary of the calculations made in mission analysis can be viewed in Annex B.

3.1.2 Ground Elevation and Atmospheric Data

Essential to the functionality of the Mission Planner Program, data from the surrounding terrain elevation and atmospheric conditions are used. Though manual input of weather allows the run of an optimization, elevation data must be downloaded from an external Application Programming Interface (API), in order to ensure that elevation constraints are not infringed. Real-time atmospheric data can also be downloaded from an external API the same way as elevation.

For both APIs, the input data, latitude and longitude, is sent as a URL address and the output data (elevation or weather) is retrieved via request and saved to a text file. In order to get a full map of coordinates and output values, several requests are automatically retrieved with a Python code implementation, the number of which depends on the resolution of the map (number of points at the border). The resulting data has the following format [11]:

$$Q_{Elevation}(Q_{lat_i}, Q_{lon_i}, Q_{ele_i}) \tag{3.1}$$

$$Q_{Weather}(Q_{lat_i}, Q_{lon_i}, Q_{Temp_i}, Q_{windspeed_i}, Q_{winddirection_i}, Q_{Clouds_i}) \tag{3.2}$$

where i is the point index, Q_{lat_i}, Q_{lon_i} are the geodetic coordinates (deg), Q_{ele_i} the elevation (m), Q_{Temp_i} the temperature (K), $Q_{windspeed_i}$ the wind speed (m/s), $Q_{winddirection_i}$ the wind direction (deg) and Q_{Clouds_i} the cloud index $(\%)$.

Whatever points are used in the analysis/optimization that do not exactly correspond to those obtained above, its elevation and weather values are interpolated from the four nearest points in the map, as observed in Figure 25.

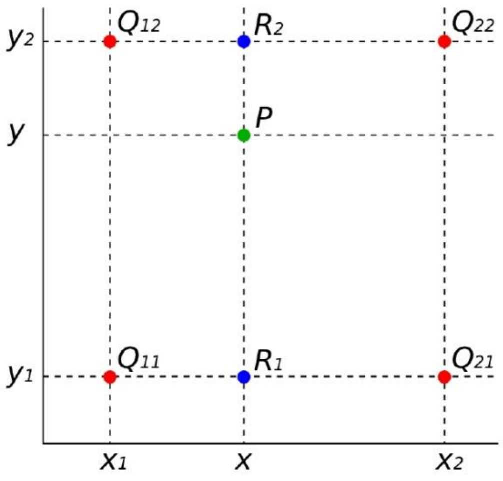


Figure 25 – Example of a point P (centre) obtained by the interpolation of four coordinates ("Q" dots) [11]

3.1.3 Solar Model

In order to take full advantage of the solar component and its impact on the endurance of the LEEUAV, a solar model was developed to better optimize a route, taking into account:

- the orientation of the PV panels in relation to the sun (θ_i);
- the area (S_{PV}) and efficiency (η_{PV}) of the PV panels;
- the irradiance of the sun (J), which itself depends on the hour of the day (H), the day of the year (d_n) and the latitude of the location of the vehicle (φ);
- the clear sky index (K_C), where a value of 1 means a clear sky and a value of 0 means an overcast sky with total obstruction of the solar irradiance.

Values for the clear sky index are obtained directly from the weather text files or from manual input in the pre-defined route. The calculation of the total energy the PV panels can harvest is summarized by the equation:

$$E_{solar|total} = \int_{t}^{t_f} J \; K_C \; S_{PV} \; \eta_{PV} \cos(\theta_i) \, dt \tag{3.3}$$

In the way that the distribution of power is set, solar power gained from the PV panels is directly used to compensate the required power of the electric motor plus all active systems onboard. If the solar power is greater than the required, its excess power is used to charge the battery. On the other hand, if all of the solar power is not enough to compensate the required power of the UAV, battery power must also be used.

Internally to the program and present in output files, a convention in the sign of used or flow of energy and power was set – Positive values indicate energy/power consumption, while negative ones indicate energy/power gain (from the only source of energy, the PV panels).

3.1.4 Propulsion Data

Although the MPP may use representation from combustion piston engines, it was primarily developed to use electric motors. As such, some considerations in the present work must be taken to describe the priority propulsion model that represents the solar LEEUAV with which to fly a mission. As pointed out in [11] this model was presented in reference [51]. The propulsion of the vehicle is mainly represented in two parts: data for its motor(s)/engine(s) and data for its propeller(s).

An electric motor converts electrical power to mechanical torque and rotational speed, while the propeller converts this torque into forward thrust. The power required at the end of this system (absorbed by the propeller) depends on a number of variables, and it must equal the power available at the motor shaft, P_{shaft}. Propeller and motor matching conditions are thus achieved when motor shaft power and propeller absorbed power are equal at the same rotational speed. This comes from the definition that power is the torque (Q_{motor}) multiplied by the angular speed ($2\pi n$).

Starting with the propellers, in order for the program to calculate their performance, values for the diameter, pitch and number of blades are required. In addition, values for the propeller efficiency η_{prop} and the power coefficient C_p must also be known. Both of these depend on the propeller advance ratio (J_{prop}), which is defined by the equation:

$$J_{prop} = \frac{V}{n.\,d} \tag{3.4}$$

where V is the linear velocity, n is the propeller's revolutions per second and d is the diameter. The power required at the shaft of the motor is then calculated by:

$$P_{shaft} = C_p \rho \ n^3 d^5 \tag{3.5}$$

The values needed for the propeller efficiency and power coefficient may be obtained using one out of five different representations that can be chosen in the program:

- Linear interpolation of data points ($C_p(J)$ and $\eta_{prop}(J)$);
- Least squares polynomial approximation of data points ($C_p(J)$ and $\eta_{prop}(J)$);
- Polynomial representation from user coefficients ($C_p(J)$ and $\eta_{prop}(J)$);
- Approximation using default polynomial curves ($C_p(J,d,p)$ and $\eta_{prop}(J,d,p)$);
- Approximation using user polynomial curves ($C_p(J,d,p)$ and $\eta_{prop}(J,d,p)$).

The last two representations' polynomial curves depend not only on the propeller advance ratio (J), but its diameter (d) and pitch (p) as well. The default representation uses polynomial coefficients defined in [51] and is explained in Annex C. The other uses user-input coefficients.

On the other part of the system, the total power consumed by the motor is obtained by:

$$P_{ele} = U \ I \tag{3.6}$$

, already considering all voltage and current losses. These losses are represented by the motor resistance R and the no-load current I_0. Together with its mass, speed constant K_v, maximum current I_{max} and voltage U_{max}, these values are specified by the manufacturer, and are needed by the program. The useful power, otherwise known as effective power (P_{eff}) is calculated as follows:

$$P_{eff} = U_{eff} I_{eff} = (U - RI)(I - I_0) \tag{3.7}$$

At the end of the calculation of the segment required power P_{req}, which depends on the drag and speed of the UAV (explained in Annex B, page 97), the analysis/optimization process will calculate three basic parameters – the motor power setting δ_{set}, the motor speed n and the input voltage U. In this iterative process the input current I and the motor power setting δ_{set} are adjusted to meet the condition of the propeller-motor matching (rotational speed, torque and power must be the same) and the condition where propeller power must be equal to flight required power. Following that, the total electric power for a segment j is calculated, using the electric motor power P_{ele_j} and the systems required power P_{sys_j}:

$$P_{T_j} = P_{ele_j} + P_{sys_j} = (UI + (R_{ESC} I^2 + R_U I^2))_j + P_{sys_j} \tag{3.8}$$

where R_{ESC} and R_U are the electronic speed controller and battery resistance, respectively. The total energy used is obtained from the sum of electric power of all segments times the respective segment elapsed time dt_j:

$$E = \sum_{j=1}^{n} (P_{T_j} dt_j) \tag{3.9}$$

3.1.5 Mission Optimization

The optimization of a mission is the main goal of the Mission Planner Program. Out of all previously discussed models and data, the MPP will try to optimize a route based on its design parameters, constraints and objective function.

FFSQP Algorithm

The only algorithm used for the optimization of all routes in previous and present work was the FFSQP algorithm. FFSQP stands for “Fortran Feasible Sequential Quadratic Programming”. It is basically a set of Fortran subroutines created to solve optimization problems using nonlinear or linear inequality and equality constraints [52]. These constraints are set by the user and must always be respected throughout the iteration process. This process can be summarized in Figure 26.

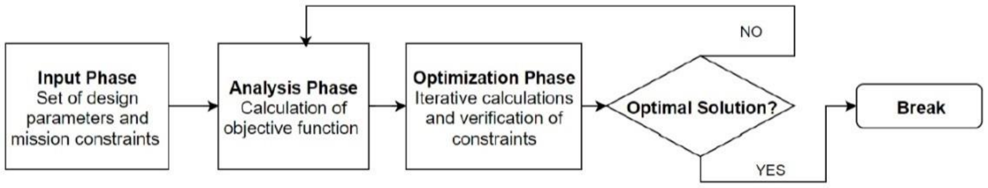

Input Phase
Set of design parameters and mission constraints
Analysis Phase
Calculation of objective function
Optimization Phase
Iterative calculations and verification of constraints
Optimal Solution?
NO
YES
Break

Figure 26 – Flowchart of the iterative process [11]

In a symbolic explanation, the SQP algorithm minimizes the objective function ($f_i(x_n)$), using a set of design parameters x constrained by lower and upper boundaries ($g_{ref}(x_n)$, reference values defined by the user) at every iteration i, within a feasible space X and a number of iterations I ([52]):

$$min\{f_i(x_n)\}, \qquad x \in X \ and \ i \in I \tag{3.10}$$

The m number of inequality constraints are defined as ([52], [53]):

$$g_j(x_n) \leq g_{ref}(x_n), \qquad j = 1,2, \dots, m \tag{3.11}$$

The complete list of constraint functions and design parameters used can be found in Annex D.

Throughout the process, the SQP algorithm estimates gradients for a certain constraint function, using forward finite differences (though the MPP provides the choice to use backward or central finite differences) with a certain increment Δx set by the user. In short, this estimation has the goal of finding a step direction p_i which, when multiplied with the gradients $\nabla f(x_i)$, has to be negative to minimize the objective function:

$$[p_i, \nabla f(x_i)] < 0 \tag{3.12}$$

After that, the algorithm calculates the step length α_i, whereby the new objective function must be validated by:

$$f(x_i + p_i\alpha_i) < f(x_i) \tag{3.13}$$

On this condition, the new set of design variables x_{i+1} are calculated and the next iteration begins. As soon as the Hessian matrix of the objective function reaches 0 or within a tolerance value ε, a final converged solution has been achieved [11], [53]. Figure 27 presents a summarized flowchart of the process ([53]):

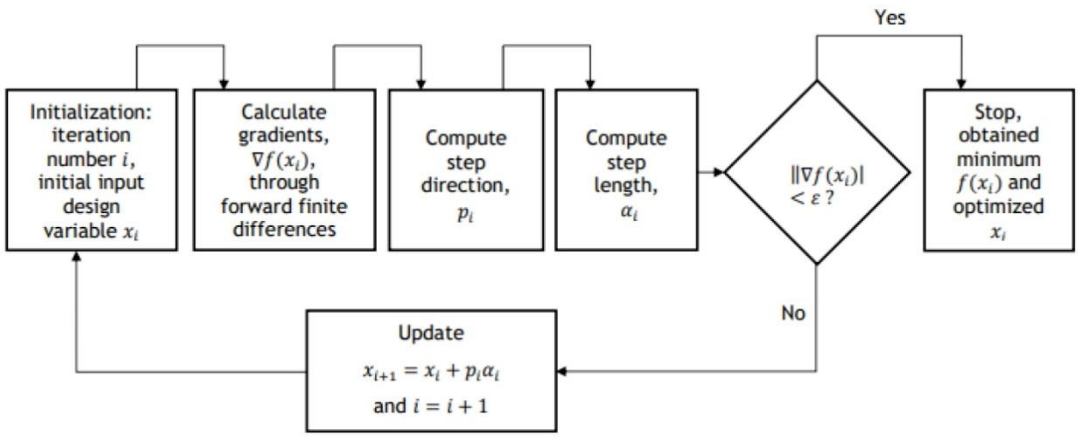

Initialization: iteration number i, initial input design variable x_i
Calculate gradients, $\nabla f(x_i)$, through forward finite differences
Compute step direction, p_i
Compute step length, α_i
$\|\nabla f(x_i)\| < \varepsilon$?
Yes
Stop, obtained minimum $f(x_i)$ and optimized x_i
No
Update $x_{i+1} = x_i + p_i\alpha_i$ and $i = i + 1$

Figure 27 – Flowchart of the iterative process of the FFSQP algorithm [11]

Objective functions

There are three calculated mission performance values serving as objective functions, that the optimization process will try to minimize: mission total consumed energy (E); mission flight time; and mission distance. Only one can be chosen by the user at a time.

Design variables and Constraint functions

The set of route parameters that the mission planner changes to optimize the objective function are the basic waypoint's coordinates – latitude (φ), longitude (λ) and altitude (h) – and the LEEUAV's airspeed (V). Another design parameter is the mission initial hour (t_{init}) which was added in the scope of this work and is further explained in section 3.2. Besides these, there is a group of variables that must satisfy equality and inequality constraints established by the user. The constraint functions act in every iteration of the FFSQP algorithm to limit solutions to only viable ones, so as to resemble reality as much as possible. There are two possible types of constraints that can be established by the user: equality and inequality constraints.

Although not used in the optimization process, the only design variables that can be limited by equality constraints are the engine/motor setting (δ_{set}):

$$\delta_{set} = \delta_{limit} \quad , \delta_{limit} \in]0,1] \tag{3.14}$$

where δ_{limit} is the limit engine/motor setting of the constraint, and the motor current (I):

$$I = I_{ref} \tag{3.15}$$

where I_{ref} is the reference motor current value of the constraint.

The inequality constraints include the engine/motor setting (equation 3.16) and motor current (equation 3.17) constraints:

$$\delta_{set} \leq \delta_{limit} \quad , \delta_{limit} \in]0,1] \tag{3.16}$$

$$I \leq I_{max} \tag{3.17}$$

where I_{max} is the maximum current of the motor; the stall speed condition, to prevent the stall of the LEEUAV:

$$\left(\frac{V_{min}}{V_s}\right)^2 C_L \leq C_{L_{max}} \tag{3.18}$$

where V_s is the stall speed and V_{min} is the minimum speed to guarantee, by default, a safety speed factor ($\frac{V_{min}}{V_s}$) of 1.2. $C_{L_{max}}$ is the maximum lift coefficient allowed by the constraint; the height condition:

$$h \geq h_{ref} \quad , h_{ref} > 0 \tag{3.19}$$

which, at waypoints not including take-off or landing, limits the LEEUAV to fly at or above a reference height value (h_{ref}) set by the user; the minimum required power constraint:

$$P_{req} \geq P_{ref} \quad , P_{ref} > 0 \tag{3.20}$$

which limits the segment required power (P_{req}) of the LEEUAV to be at or above a reference value (P_{ref}) set by the user; and the battery energy left condition:

$$E_{left} \geq E_{ref} \tag{3.21}$$

which limits the battery energy remaining at the end of a segment (E_{left}) at or above the reference value E_{ref}.

3.2 Added features

In the scope of the present work, some additional features were added to enhance the functionality of Coelho's Mission Planner Program.

The first, a stretch of Fortran code added to the "main program" subroutine, instructs the MPP to create an empty "dummy checker" text file into the executable's directory. This allows the present work's Mission Optimization Interface to check that the program has finished all calculations, and that it is cleared to proceed to the retrieval of data from output text files ("mission_convergence.txt" and/or "mission_out.txt") created by the Mission Planner Program. The GUI automatically terminates the MPP and deletes the "dummy checker" once this file is detected.

The second feature involves a small correction to the minimization of energy usage in the optimization method and consequently the calculation of energy flow. Before this modification, the algorithm would only try to maximize the energy obtained from the solar component (calculated by equation 3.3), neglecting the minimization of consumed energy (calculated by equation 3.9), during the optimization of the design parameters. Additional power and energy flow values were added to one of the output files, “mission_out.txt”.

The third feature is ground-breaking – the addition of a “mission initial hour” design parameter for the optimization process. This design parameter allows for a new range of better optimization results for both energy and time minimization, taking into account a greater amount of data from the ever-changing weather. It also provides the user an expectation of the best time of the day to launch the LEEUAV.

In the way this new design parameter works, a new time system had to be designed. This system works much like the Unix Timestamp, which is the UTC count in seconds since 1st January 1970, used in a number of applications and computers [54]. However, the timestamp in this program is set in hours, and it is referenced from 0h 1st January year 0. Therefore, from data specified in the “mission_waypoints.txt” file, the mission initial hour takes the following format, internally to the code:

$$t_{init} = t_{init_h} + t_{init_{day}} \times 24 + t_{init_{year}} \times 365 \times 24 \tag{3.22}$$

where $t_{init_{day}}$ is the day of the year, ranging from 1 to 365/366.

The setting of the mission initial hour acts as a reference to the “mission initial hour” design parameter. From it, a non-zero offset value in hours may be introduced, and from this reference offset, lower and upper bounds for the minimum and maximum time instants, respectively, are introduced. The optimization is then undertaken solely within this interval.

Finally, a correction was added to the stall speed constraint. This constraint was applied only at a segment average speed condition and not at the waypoints. This could result in very low speeds at the waypoints violating the safety speed margin. Now this constraint is enforced at the waypoints as well.

3.3 Directory Structure Breakdown

Understanding the directory structure that makes up the Mission Planner Program is crucial to the development of the Mission Optimization Interface, explained in chapter 4.

In the main folder containing the program, one can find three parts: the “Mission.exe” executable through which the MPP is run; the “mission_directory.txt” text file, which points the paths to the various necessary data text files; and the “Missioncode” folder. The functioning

Fortran code files of the MPP are spread out in 10 folders within “Missioncode”. They are sorted, together with input and output data text files, according to their respective section. The diagram in Figure 28 sums up this structure.

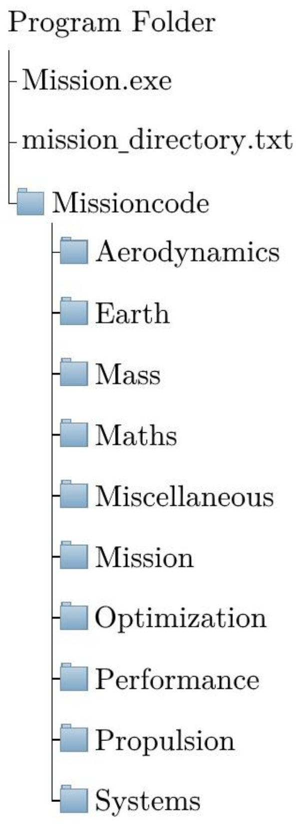


Figure 28 – Breakdown of the directory structure of the Mission Planner Program

Chapter 4

GUI Development

This chapter is dedicated to a thorough explanation of the development and functionality of the Mission Optimization Interface, a graphical user interface intended to be used alongside the Mission Planner Program created by Coelho, described in the previous chapter.

The GUI was developed using a binding of Python v3.7 code with The Qt Company's Qt v5 application framework – better known as PyQt5 – under a GPL license [55]. Although Qt's priority development code is C++ and Java, Python was chosen for its simplicity and ease of learning, characteristic of a high-level programming language.

Besides defining the elements that make up the framework of the interface, the way PyQt5 works is focused around signals and slots. Every time an action of potential interest happens in the interface framework (like clicking a button), a signal that may or may not be connected to a slot is triggered [56]. Slots may take different shapes, varying between a simple change of a label text, to the calling of complex code functions.

Due to its complexity, the project was divided in two parts:

- Creation of the GUI application framework – the window itself – using Qt Designer. Only a few signals and slots where introduced into buttons, widgets and other items within the main window.
- Implementation of the necessary Python code functions that provide the majority of signal/slot functionality to the interface as a whole. Within these are also the code functions that bridge the GUI with the Mission Planner Program, as well as the freeware MissionPlanner.

Worth noting is the fact that the development of the window framework was dependent on the Python code implementation. Changes in widgets, buttons and other items' names had to be accompanied with necessary changes to this code, and vice-versa. Therefore, after first finishing the basic framework, these two parts were continuously developed in parallel with each other towards the end of this thesis.

4.1 Basic Features

The GUI framework is divided into three basic areas of operability, as seen in Figure 29 – the menu bar (outlined in red), a central tab widget (outlined in green), and a bottom section (outlined in blue).

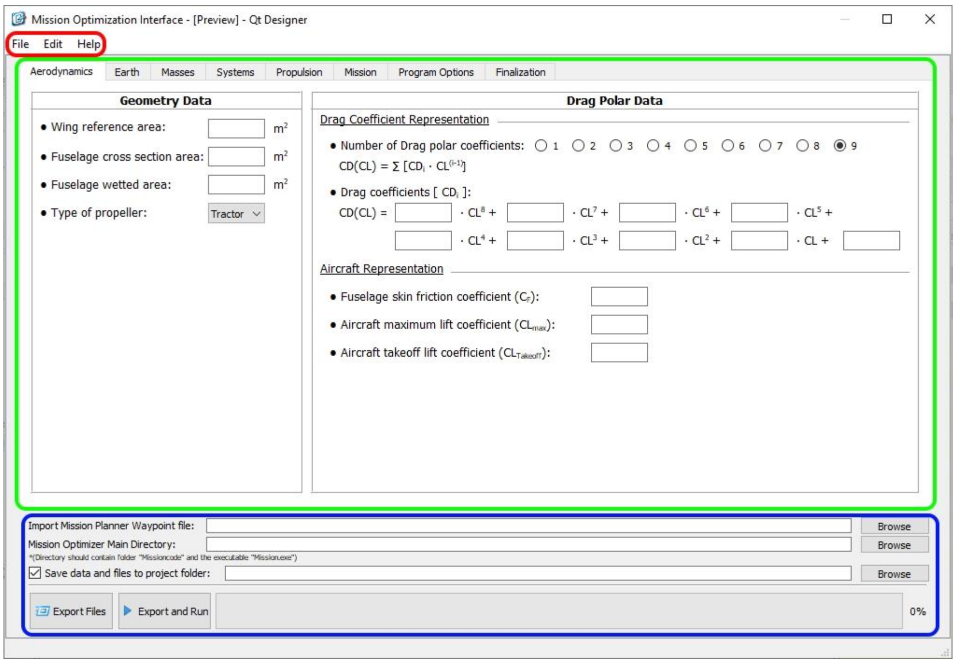

Mission Optimization Interface - [Preview] - Qt Designer
File
Edit
Help
Aerodynamics
Earth
Masses
Systems
Propulsion
Mission
Program Options
Finalization
Geometry Data
• Wing reference area: m²
• Fuselage cross section area: m²
• Fuselage wetted area: m²
• Type of propeller: Tractor
Drag Polar Data
Drag Coefficient Representation
• Number of Drag polar coefficients: 1 2 3 4 5 6 7 8 [✓ 9]
CD(CL) = Σ [CDᵢ · CL⁽ⁱ⁻¹⁾]
• Drag coefficients [CDᵢ]:
CD(CL) = · CL⁸ + · CL⁷ + · CL⁶ + · CL⁵ +
· CL⁴ + · CL³ + · CL² + · CL +
Aircraft Representation
• Fuselage skin friction coefficient (C_F):
• Aircraft maximum lift coefficient (CL_max):
• Aircraft takeoff lift coefficient (CL_Takeoff):
Import Mission Planner Waypoint file:
Browse
Mission Optimizer Main Directory:
Browse
*(Directory should contain folder "Missioncode" and the executable "Mission.exe")
[✓] Save data and files to project folder:
Browse
Export Files
Export and Run
0%

Figure 29 – Areas of the GUI main window: menu bar (top, red), tab widget (centre, green) and bottom section (bottom, blue)

The top menu bar contains various functions, some of which are also present in the central and bottom areas of the interface. Figure 30 shows the appearance of the File menu, the actions of which are described below:

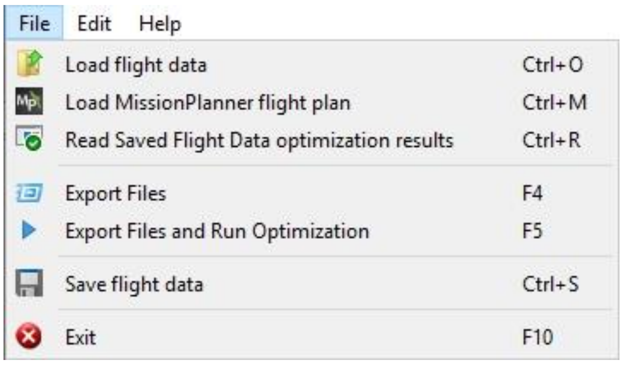

File
Edit
Help
Load flight data Ctrl+O
Load MissionPlanner flight plan Ctrl+M
Read Saved Flight Data optimization results Ctrl+R
Export Files F4
Export Files and Run Optimization F5
Save flight data Ctrl+S
Exit F10

Figure 30 – Interface File menu

- "Load flight data" allows the user to load a .dat file containing edited data of the UAV or mission specifications. With this, the interface replaces data in all fields (except API keys in the Earth tab) with the loaded data.
- "Load MissionPlanner flight plan" directs the interface to read a .waypoints file created by the freeware MissionPlanner. This action is explained in section 4.9.1.

- “Read Saved Flight Data optimization results” loads results from a previous optimization/analysis run of the Mission Planner Program and shows it in the “Finalization” tab of the central widget, which is further explained in section 4.8.3.
- “Export Files” instructs the interface to export the data edited in the GUI into text files, which are placed into a folder specified at the bottom section browsers (explained ahead), as well as the directory text file necessary to the MPP executable.
- “Export Files and Run Optimization” not only exports the files, but it also commands the start of the MPP using the options described in the “Program Options” tab.
- “Save flight data” allows the user to save the edited interface data into a .dat file, in a folder of his/her choice.
- “Exit” simply terminates the GUI.

The Edit menu contains actions related to data editing and other miscellaneous actions:

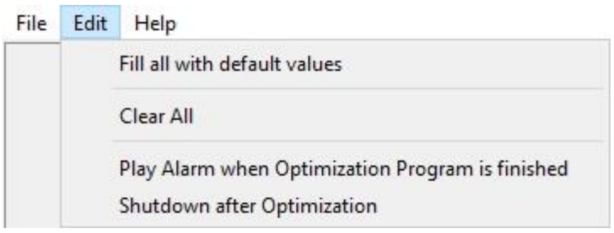
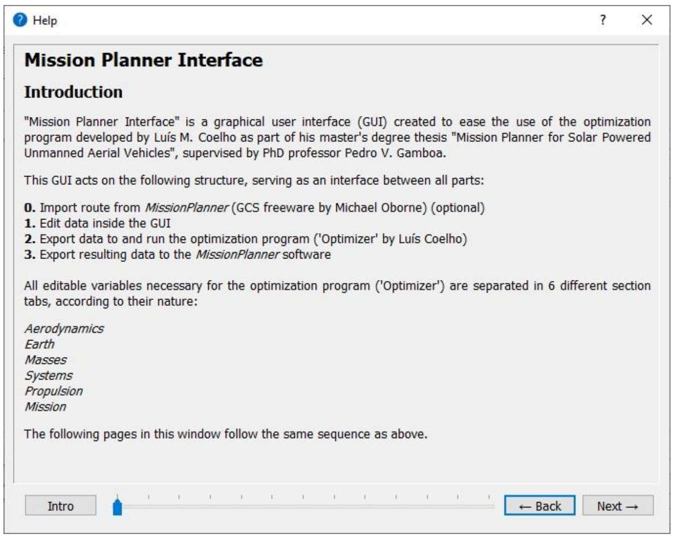

File
Edit
Help
Fill all with default values
Clear All
Play Alarm when Optimization Program is finished
Shutdown after Optimization

Figure 31 – Interface Edit menu

- “Fill all with default values” replaces all editable fields in the interface with default data loaded from a text file (“default_values.txt”), which can be manually edited, in the interface executable’s directory.
- On the other hand, “Clear All” simply clears the interface and deletes any data in all editable fields.
- “Play Alarm when Optimization Program is finished” commands the GUI to trigger a two-beep alarm when the Mission Planner Program is terminated;
- “Shutdown after Optimization” commands the GUI to shutdown the computer when the Mission Planner Program is terminated.

The Help menu lists “About” information of the interface, and “Help”, which is a widget with a separate framework created to aid the user with the operation of the interface. This widget is merely presentational and does not feature any functionality whatsoever. Figure 32 shows the first of 13 pages of the “Help” window.



Help

Mission Planner Interface

Introduction

"Mission Planner Interface" is a graphical user interface (GUI) created to ease the use of the optimization program developed by Luís M. Coelho as part of his master's degree thesis "Mission Planner for Solar Powered Unmanned Aerial Vehicles", supervised by PhD professor Pedro V. Gamboa.

This GUI acts on the following structure, serving as an interface between all parts:

0. Import route from *MissionPlanner* (GCS freeware by Michael Oborne) (optional)
1. Edit data inside the GUI
2. Export data to and run the optimization program ('Optimizer' by Luís Coelho)
3. Export resulting data to the *MissionPlanner* software

All editable variables necessary for the optimization program ('Optimizer') are separated in 6 different section tabs, according to their nature:

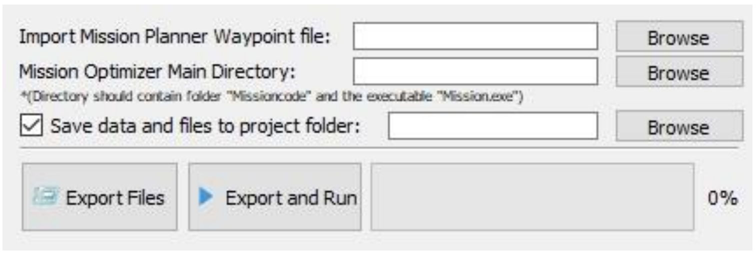
Aerodynamics
Earth
Masses
Systems
Propulsion
Mission

The following pages in this window follow the same sequence as above.

Figure 32 – Introduction page of the “Help” window

At the centre area (outlined in Figure 29), the main tab widget is where data processing takes place. It was divided according to the text files’ subjects where data is stored as input to the Mission Planner Program – 6 tabs – as was described in section 3.1. Two more tabs are also present: “Program Options”, which is an intermediate step between exporting files and starting the MPP; and “Finalization”, where output data is processed and shown to the user (explained in detail in section 4.8.3).

Lastly, as seen in Figure 33, the Bottom area section handles directory management and operation triggers:



Import Mission Planner Waypoint file:
Mission Optimizer Main Directory:
*(Directory should contain folder "Missioncode" and the executable "Mission.exe")
☑ Save data and files to project folder:

Figure 33 – Bottom section of the interface

- The first browser, “Import Mission Planner Waypoint file”, is an action equal to that described in the File menu, “Load MissionPlanner flight plan” and is explained in section 4.9.1;
- The “Mission Optimizer Main Directory” browser indicates the path to the Mission Planner Program, which contains the executable “Mission.exe” and the “Missioncode” folder containing all Fortran subroutines and text files. This is necessary for a large number of functions within the GUI, including one calling the start of the executable;
- The “Save data and files to project folder” browser specifies a path to which all files are exported, and the MPP’s working directory, in other words where it will export files

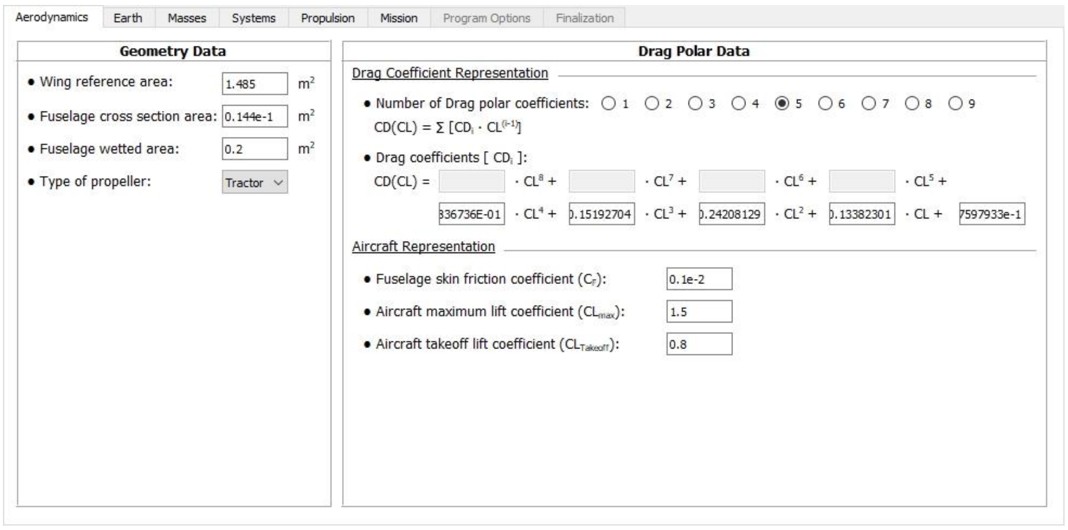
with output data from analysis/optimization processes. This path is written to the "mission_directory.txt" file in the directory above. If it is not specified, the MPP uses files exported to various folders within "Missioncode".

- The last part contains two buttons – "Export files" and "Export and Run" – which when clicked instruct actions already described in the File menu. It also contains a progress bar to the right, which indicates the status of the export operation of the GUI.

Throughout sections 4.2 to 4.7, every subject's data export operation to a text file involves the running of Python functions which are called as soon as the GUI checks the integrity of all of the GUI's edited data and directory specification. The check processes are described particularly in every section, and overall in 4.8.

4.2 Aerodynamics

The Aerodynamics subject, important for the calculation of aerodynamic forces and power requirements for propulsion, involves the exporting of data in two parts: the LEEUAV's geometry, and data representing its drag and lift coefficients.



Geometry Data

- Wing reference area: 1.485 m^2
- Fuselage cross section area: 0.144e-1 m^2
- Fuselage wetted area: 0.2 m^2
- Type of propeller: Tractor

Drag Polar Data

Drag Coefficient Representation

- Number of Drag polar coefficients: 1 2 3 4 [✓ 5] 6 7 8 9
  $CD(CL) = \Sigma\,[CD_i \cdot CL^{(i-1)}]$
- Drag coefficients [CD_i]:
  $CD(CL) =$ · CL^8 + · CL^7 + · CL^6 + · CL^5 +
  336736E-01 · CL^4 +).15192704 · CL^3 +).24208129 · CL^2 +).13382301 · CL + 7597933e-1

Aircraft Representation

- Fuselage skin friction coefficient (C_F): 0.1e-2
- Aircraft maximum lift coefficient (CL_{max}): 1.5
- Aircraft takeoff lift coefficient ($CL_{Takeoff}$): 0.8

Figure 34 – GUI's Aerodynamics section tab

4.2.1 Geometry Data

In the geometry subsection, only four variables are needed:

- Wing reference area (S);
- Fuselage cross section area (S_F);
- Fuselage wetted area (S_{Wet});
- Type of propeller (tractor or pusher).

Once the GUI is commanded to export the data, a “checker” function is called. This function checks if all editable text fields – *lineEdits* – are convertible to valid numbers. The last variable is a combo box and it always displays a valid option (index), regardless of the user’s choice. The flowchart in Figure 35 describes this process.

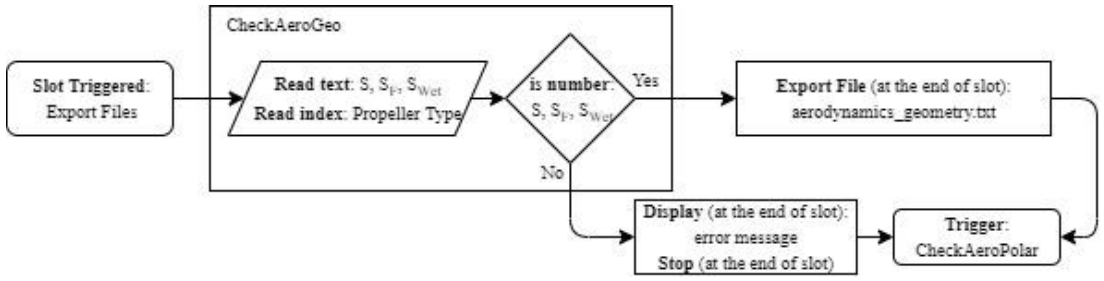

CheckAeroGeo
Slot Triggered: Export Files
Read text: S, S_F, S_Wet
Read index: Propeller Type
is number: S, S_F, S_Wet
Yes
No
Export File (at the end of slot): aerodynamics_geometry.txt
Display (at the end of slot): error message
Stop (at the end of slot)
Trigger: CheckAeroPolar

Figure 35 - Aerodynamics/Geometry export operation flowchart

If any of the fields is an invalid number, the process continues to the next section (Drag Polar data), but displays an error message and stops the GUI operation “Export Files” at its end, preventing any files from being exported. Otherwise, the “aerodynamics_geometry.txt” file is exported at the end of all interface checks.

4.2.2 Drag Polar Data

In this subsection there are 12 editable text fields:

- Up to 9 coefficients, to be used in the drag equation:

$$C_D(C_L) = \sum_{i=1}^{n} C_{D_i} . C_L^{(i-1)} \tag{3.23}$$

- Fuselage skin friction coefficient (C_F);
- Maximum lift coefficient ($C_{L_{max}}$);
- Take-off lift coefficient ($C_{L_{takeoff}}$).

The first field of the Drag Polar data subsection specifies the number of coefficients to be used in equation 3.23, and also enables/disables *lineEdit* boxes accordingly. The rest of the fields are treated much like in the Geometry section.

After the geometry section is complete, while in the “Export Files” GUI operation, a “checkAeroPolar” function is called, to validate any and all text inputs. If unsuccessful, an error message is displayed at the end and no data is exported. Otherwise, a “aerodynamics_polar.txt” file is exported. This process is described by the flowchart in Figure 36.

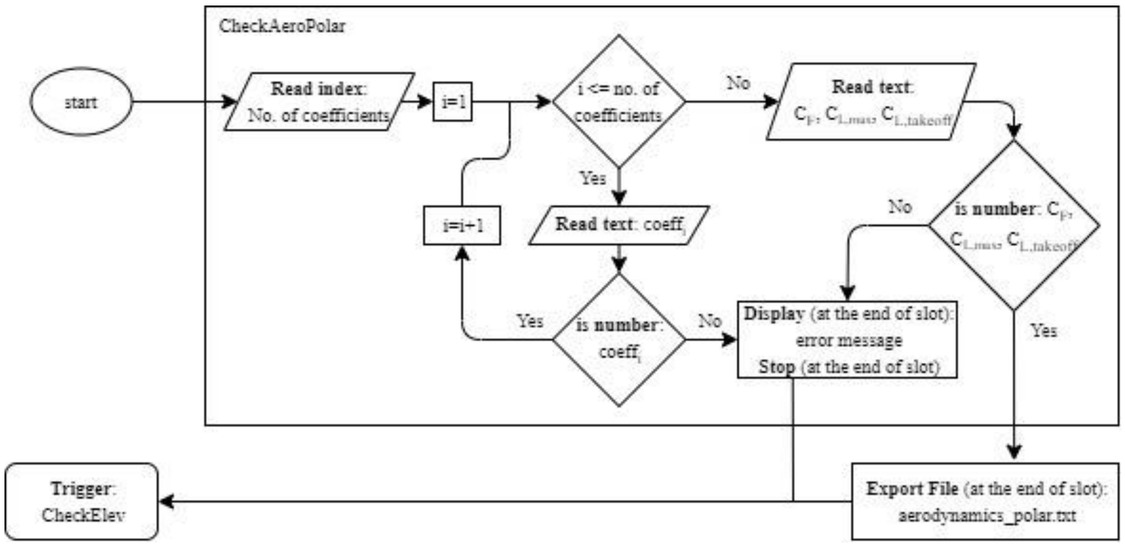
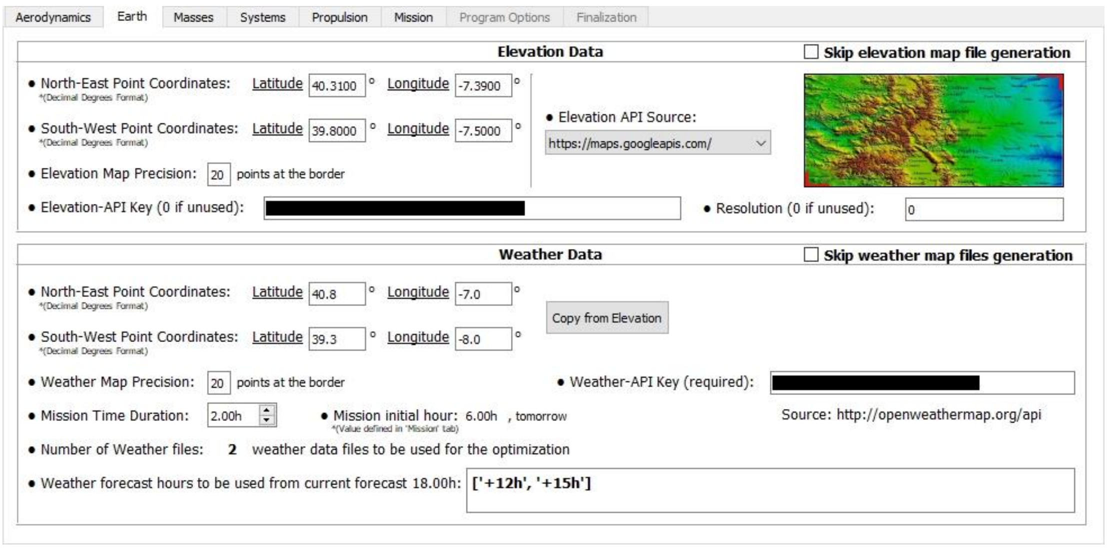

CheckAeroPolar
start
Read index: No. of coefficients
i=1
i <= no. of coefficients
No
Read text: C_F, $C_{L,max}$, $C_{L,takeoff}$
Yes
Read text: $coeff_i$
i=i+1
is number: C_F, $C_{L,max}$, $C_{L,takeoff}$
No
Yes
is number: $coeff_i$
No
Yes
Display (at the end of slot): error message
Stop (at the end of slot)
Trigger: CheckElev
Export File (at the end of slot): aerodynamics_polar.txt

Figure 36 – Aerodynamics/Drag Polar export operation flowchart

4.3 Earth

The Earth section is one of the most complex and important sections developed in the Mission Optimization Interface. Besides providing terrain elevation data in one part, the other provides the weather conditions, which may or may not help the optimization process. In some cases, weather can make this process impossible or even forbid the operation of the LEEUAV altogether.

L. Coelho's Python routines that generate the elevation and weather maps, described in section 3.1.2, were integrated into the interface's "Export Files" slot as callable functions by the Elevation Data and Weather Data file generation functions, respectively. In addition, new features were added to these code routines, which will be explained in their respective sections.


Aerodynamics | Earth | Masses | Systems | Propulsion | Mission | Program Options | Finalization

Elevation Data — ☐ **Skip elevation map file generation**

- North-East Point Coordinates: *(Decimal Degrees Format) Latitude 40.3100 ° Longitude -7.3900 °
- South-West Point Coordinates: *(Decimal Degrees Format) Latitude 39.8000 ° Longitude -7.5000 °
- Elevation Map Precision: 20 points at the border
- Elevation API Source: https://maps.googleapis.com/
- Elevation-API Key (0 if unused): [redacted]
- Resolution (0 if unused): 0

Weather Data — ☐ **Skip weather map files generation**

- North-East Point Coordinates: *(Decimal Degrees Format) Latitude 40.8 ° Longitude -7.0 °
- South-West Point Coordinates: *(Decimal Degrees Format) Latitude 39.3 ° Longitude -8.0 °

Copy from Elevation

- Weather Map Precision: 20 points at the border
- Weather-API Key (required): [redacted]
- Mission Time Duration: 2.00h
- Mission initial hour: 6.00h , tomorrow *(Value defined in 'Mission' tab)

Source: http://openweathermap.org/api

- Number of Weather files: **2** weather data files to be used for the optimization
- Weather forecast hours to be used from current forecast 18.00h: **['+12h', '+15h']**

Figure 37 – Design of the Earth tab in the interface

4.3.1 Elevation Data

Besides the original API website from which L. Coelho's elevation map generation function retrieves data [57], an additional elevation API from Google was added as a possible source of data [58]. Either one or the other may be chosen in a combo box within the interface, while also specifying the resolution and keys required.

In addition to this, the following data is also needed:

- A southwest (SW) and northeast (NE) corner points' coordinates (Lat_{SW}, Lon_{SW} and Lat_{NE}, Lon_{NE}) delimitating the map to be generated;
- The number of points at the border (precision), the square of which gives the total number of points in the map.

The check system of this section (function "checkElev") is similar to that of Aerodynamics, as all values must also be numbers. Additionally, however, the northeast point coordinates must have greater values than the ones from the southwest point, the latitudes and longitudes must be within the intervals $[-90, 90]$ and $[-180, 180]$, respectively, and the precision value must be a positive integer. Unless the user checks the box "Skip elevation map file generation", stopping the function altogether, the GUI must ensure an internet connection is established to one of the API's IP addresses. These are listed in the file "ip_config.txt".

If all above conditions are met, then a map can successfully be generated and exported at the end of the "Export Files" slot operation. This can be summarized by the flowchart in Figure 38.

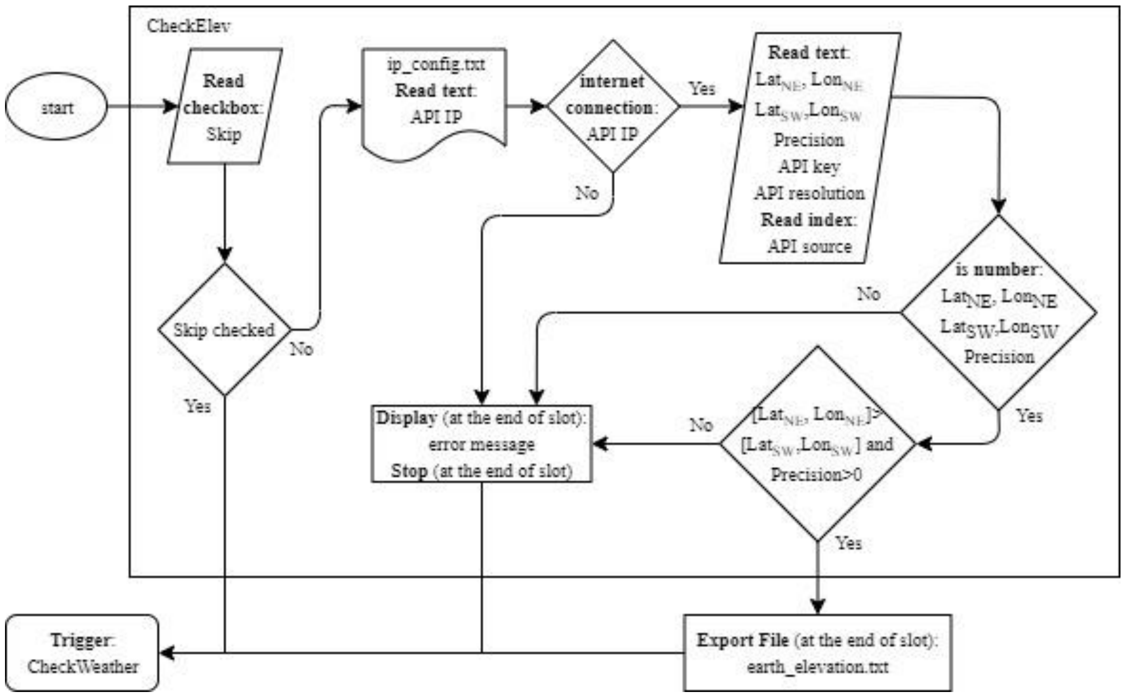

CheckElev
start
Read checkbox: Skip
Skip checked
No
Yes
ip_config.txt Read text: API IP
internet connection: API IP
Yes
No
Read text: Lat_NE, Lon_NE Lat_SW, Lon_SW Precision API key API resolution Read index: API source
is number: Lat_NE, Lon_NE Lat_SW, Lon_SW Precision
No
Yes
[Lat_NE, Lon_NE]> [Lat_SW, Lon_SW] and Precision>0
No
Yes
Display (at the end of slot): error message Stop (at the end of slot)
Trigger: CheckWeather
Export File (at the end of slot): earth_elevation.txt

Figure 38 – Earth/Elevation export operation flowchart

4.3.2 Weather Data

The weather data interface panel is similar to the elevation section. However, a few additional features are displayed, as seen in Figure 37.

While not added to the Mission Planner Program Fortran code itself, a new improvement made to the Python weather map generation function created by L. Coelho allows for the automatic retrieval of several forecasts on the same map. It also automatically adds three more columns in the generated “earth_weather_n.txt” text files, representing information on the date of those forecast conditions – hour, day and year.

The weather data API allows for the download of data up to 5 days from the current forecast hour, which is set to the nearest 3-hour multiple in the 24-hour day cycle. Forecasts are set 3 hours apart from each other, therefore up to 40 weather files can be downloaded at once for the same map. Of course, the LEEUAV currently developed by UBI and IST is not designed to fly more than 8 hours straight, and no more than 5 to 7 weather files have to be used. The number of files needed to fly a route is specified by the part of the interface shown in Figure 39.

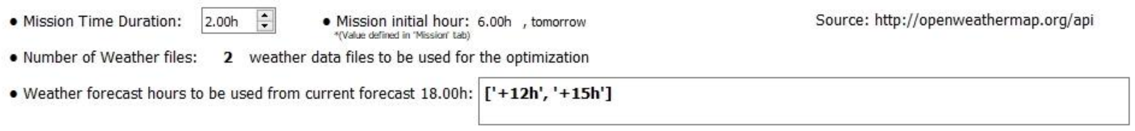

• Mission Time Duration: 2.00h • Mission initial hour: 6.00h , tomorrow
*(Value defined in 'Mission' tab)
Source: http://openweathermap.org/api
• Number of Weather files: **2** weather data files to be used for the optimization
• Weather forecast hours to be used from current forecast 18.00h: **['+12h', '+15h']**

Figure 39 – Closeup of the lower part of the GUI’s Weather Data section

In an internal slot function, two variables are needed to calculate the number of weather files for the same map:

- The “Mission initial hour” (t_{init_h}) and day of the year ($t_{init_{day}}$), which are set in the “Mission” tab;
- The “Mission time duration” ($duration$), the expected duration of the mission adjustable by the user.

The way this function works is that it computes the current forecast ($fore_{now}$), using the computer’s current hour (t_{now_h}) and rounds it to the nearest 3-hour multiple in the 24-hour day cycle, where:

$$t_{now_h} \in [i-1.5; i+1.5[\implies fore_{now} = i, \qquad i = 0,3,6,9,12,15,18,21,24 \tag{3.24}$$

Following that, in a similar way, the mission initial forecast ($fore_{init}$) and mission final forecast ($fore_{final}$) hours are computed:

$$t_{final_h} = t_{init_h} + (t_{init_{day}} - t_{now_{day}}) \times 24 + duration \tag{3.25}$$

$$t_{init_h} \in [i-1.5; i+1.5[\implies fore_{init} = i, \qquad i = 0,3,6,9,... \tag{3.26}$$

$$t_{final_h} \in [i-1.5; i+1.5[\implies fore_{final} = i, \qquad i = 0,3,6,9,... \tag{3.27}$$

where t_{init_h} is the mission initial hour, $t_{init_{day}}$ is the mission initial day (1 to 365/366), $t_{now_{day}}$ is the current day (1 to 365/366) and t_{final_h} is the mission final hour. Finally, the number of files to be used can then be calculated:

$$n_{weather} = \frac{fore_{final} - fore_{init}}{3} + 1 \tag{3.28}$$

Displayed in the box at the bottom right corner of Figure 39 is the list of forecast hours to be used (from $fore_{now}$). This list is the group of all usable forecasts contained in the interval $[fore_{init}, fore_{final}]$.

As in elevation, the rest of the data needed for the generation of weather maps is the following:

- A southwest (SW) and northeast (NE) corner points' coordinates (Lat_{SW}, Lon_{SW} and Lat_{NE}, Lon_{NE}) delimitating the map to be generated;
- The number of points at the border (precision), the square of which gives the total number of points in the map;
- An API key.

The check system for the weather data section (function "checkWeather") is almost identical to that of the elevation data section, however: because the weather API cannot provide past forecasts, the "checkWeather" function must ensure that the date of the start of the mission is not set before the current date; on the other hand, because the weather API is limited to forecasts up to 5 days, the mission initial forecast ($fore_{init}$) and/or mission final forecast ($fore_{final}$) hours must not cross this threshold. Should any of these happen, an ERROR message with description of the violated condition will fill the forecasts' list box.

Regardless of this, the user still has the choice to run an optimization/analysis using past weather data files, if the "Skip weather map files generation" checkbox is checked and the mission date (start-to-finish) is contained within the weather forecasts (files) to be used.

The flowchart in Figure 40 resumes the operation of "checkWeather" after the "Export Files" slot is called.

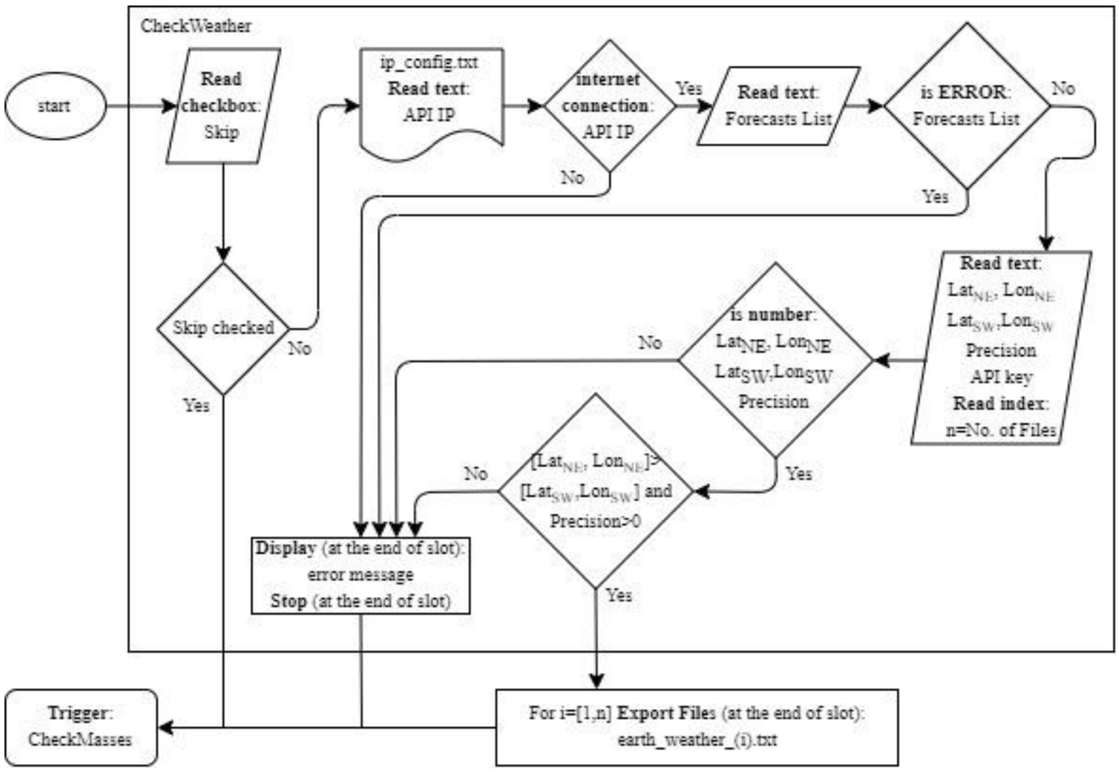
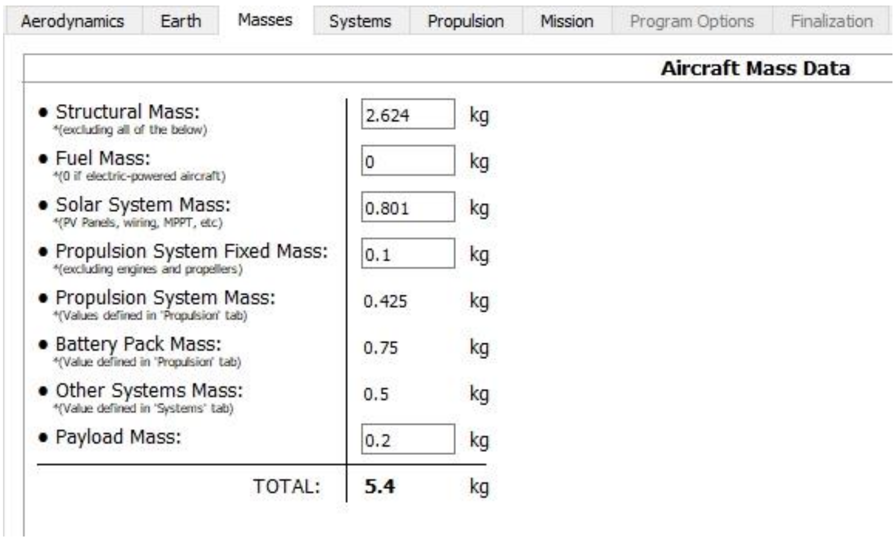

CheckWeather
start
Read checkbox: Skip
ip_config.txt Read text: API IP
internet connection: API IP
Yes
No
Read text: Forecasts List
is ERROR: Forecasts List
No
Yes
Read text: Lat_{NE}, Lon_{NE} Lat_{SW},Lon_{SW} Precision API key Read index: n=No. of Files
is number: Lat_{NE}, Lon_{NE} Lat_{SW},Lon_{SW} Precision
No
Yes
Skip checked
No
Yes
[Lat_{NE}, Lon_{NE}]> [Lat_{SW},Lon_{SW}] and Precision>0
No
Yes
Display (at the end of slot): error message Stop (at the end of slot)
Trigger: CheckMasses
For i=[1,n] Export Files (at the end of slot): earth_weather_(i).txt

Figure 40 - Earth/Weather export operation flowchart

4.4 Masses

As its name suggests, the Masses section is the one responsible for exporting data related to the mass of the aircraft. A simple list of mass values obtained from various parts and systems that make up the aircraft was embedded into the interface:



Aircraft Mass Data

• Structural Mass: *(excluding all of the below)	2.624	kg
• Fuel Mass: *(0 if electric-powered aircraft)	0	kg
• Solar System Mass: *(PV Panels, wiring, MPPT, etc)	0.801	kg
• Propulsion System Fixed Mass: *(excluding engines and propellers)	0.1	kg
• Propulsion System Mass: *(Values defined in 'Propulsion' tab)	0.425	kg
• Battery Pack Mass: *(Value defined in 'Propulsion' tab)	0.75	kg
• Other Systems Mass: *(Value defined in 'Systems' tab)	0.5	kg
• Payload Mass:	0.2	kg
TOTAL:	**5.4**	kg

Figure 41 – Design of the Masses tab in the interface

- “Structural mass of the aircraft” excludes all other parts in this list;
- “Fuel mass” is present in the case of combustion piston aircraft;
- “Solar system mass” is the sum of the mass of all parts in the solar component;

- “Propulsion system fixed mass” excludes rotating parts such as motors and propellers;
- “Propulsion system mass” is the sum of the engine/motor mass and the propeller mass, values that are edited from the “Propulsion” tab;
- “Battery pack mass” is the product of the number of cells in series, number of cells in parallel and single cell mass, values that are edited from the “Propulsion” tab;
- “Other systems mass” is the sum of the mass of all items in the list present in the “Systems” tab;
- “Payload mass” is the useful mass carried by the aircraft.

The sum of the mass of all items in the list is displayed after all fields have valid numbers and is automatically updated whenever any *lineEdit* field is edited. After the “Export Files” slot function is triggered, the check function “checkMasses” is called, checking if all *lineEdit* fields are valid numbers and if there is a “Total” value. The flowchart for this process is very simple.

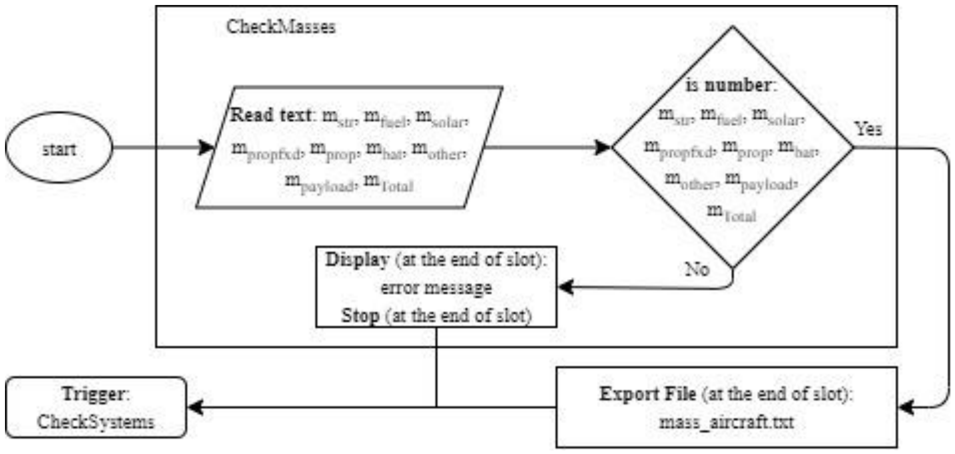
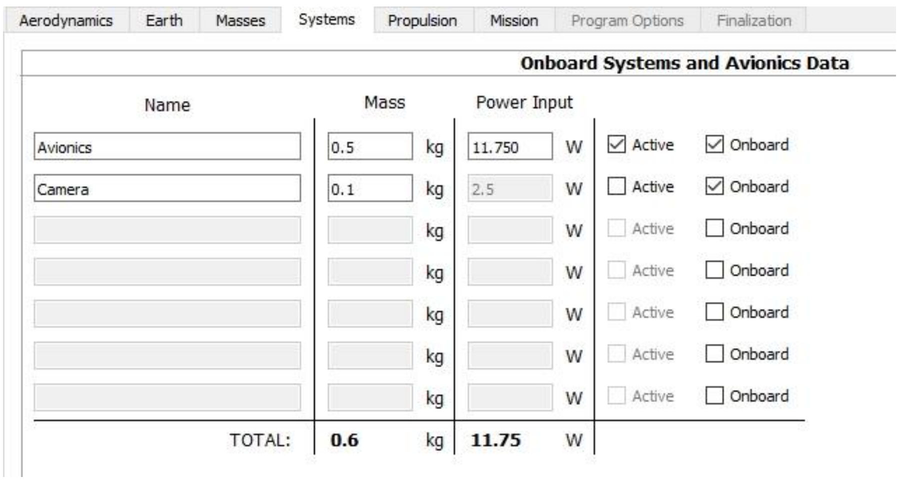

CheckMasses
start
Read text: m_{str}, m_{fuel}, m_{solar}, $m_{propfxd}$, m_{prop}, m_{bat}, m_{other}, $m_{payload}$, m_{Total}
is number: m_{str}, m_{fuel}, m_{solar}, $m_{propfxd}$, m_{prop}, m_{bat}, m_{other}, $m_{payload}$, m_{Total}
Yes
No
Display (at the end of slot): error message
Stop (at the end of slot)
Trigger: CheckSystems
Export File (at the end of slot): mass_aircraft.txt

Figure 42 - Masses export operation flowchart

4.5 Systems

The Systems tab allows the user to edit data regarding mass and power consumption of various systems present in the aircraft. These systems exclude the propulsion and solar components, and the edited values are important for the calculation of the total electric power (P_T) and total energy used (E). As in the Masses tab, a list was embedded into the interface, which allows for the input of up to 7 systems in the aircraft. In the end, all active systems’ power and their masses are summed up and their total is displayed at the bottom of the list.

Each line (system) of the list contains 3 editable fields - the system’s name, mass and input power (consumed power). The system can also be specified to be active or not in a checkbox to the right. If left unchecked, the power input data is ignored in the calculation of total power input. To allow editing of the line (system), the “onboard” checkbox also has to be checked, in the far right. Figure 43 shows an example of a list of systems.



Onboard Systems and Avionics Data

Name	Mass		Power Input			
Avionics	0.5	kg	11.750	W	☑ Active	☑ Onboard
Camera	0.1	kg	2.5	W	☐ Active	☑ Onboard
		kg		W	☐ Active	☐ Onboard
		kg		W	☐ Active	☐ Onboard
		kg		W	☐ Active	☐ Onboard
		kg		W	☐ Active	☐ Onboard
		kg		W	☐ Active	☐ Onboard
TOTAL:	**0.6**	kg	**11.75**	W		

Figure 43 - Design of the Systems tab in the interface

Before any file is exported in the "Export Files" slot, the function "checkSystems" ensures the validity of data in the Systems tab. All active *lineEdit* fields in this tab must not be empty, and the mass and power input boxes must be numbers. Only positive values are allowed in the mass, but negative values are accepted in the power input field, a way for the user to specify a system that provides, instead of consuming, power. This function is summarized in the flowchart of Figure 44.

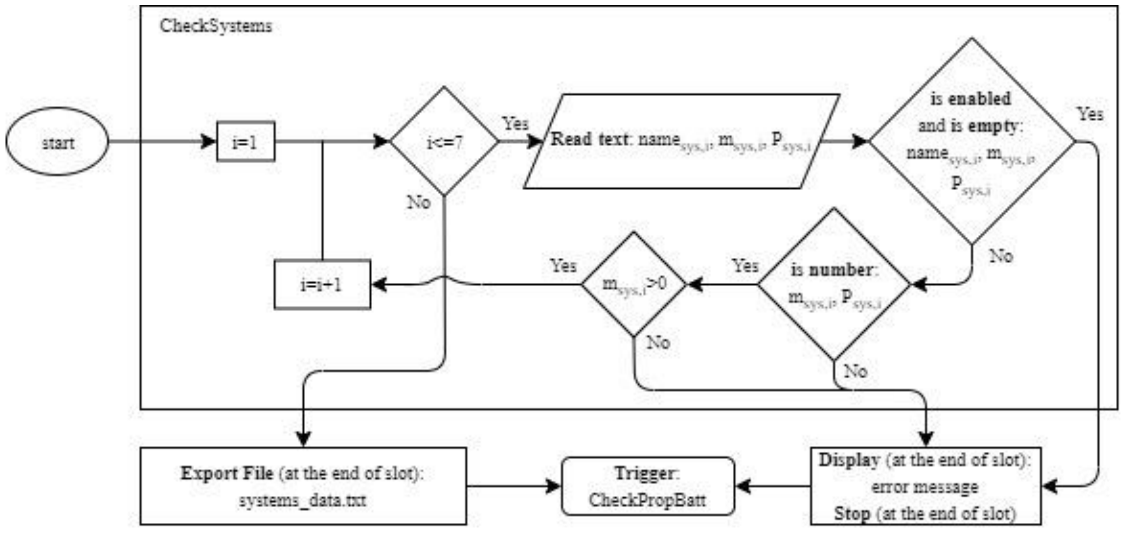
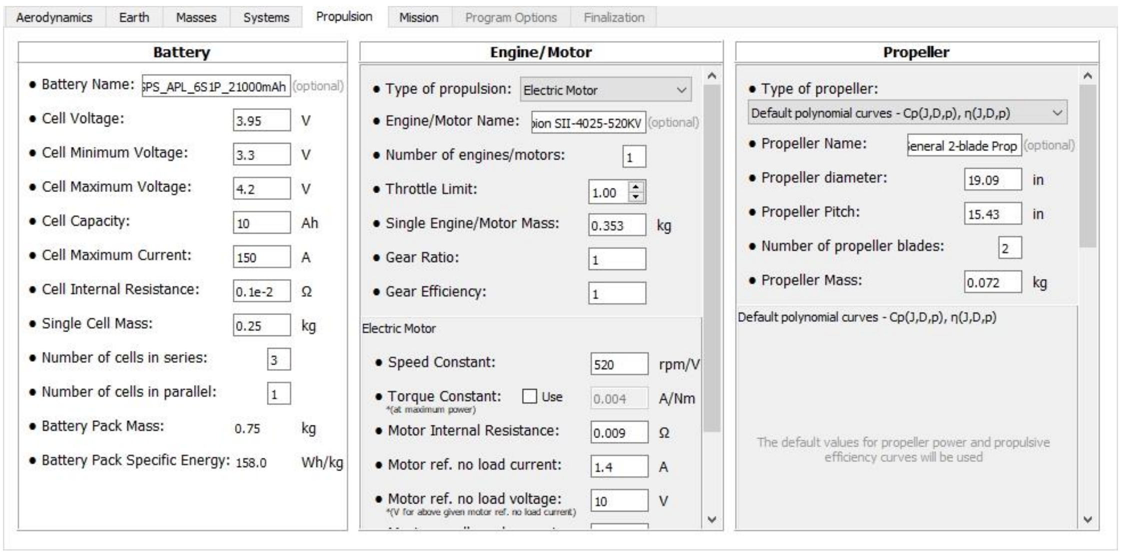

CheckSystems
start
i=1
i<=7
Yes
No
Read text: name$_{sys,i}$, m$_{sys,i}$, P$_{sys,i}$
is enabled and is empty: name$_{sys,i}$, m$_{sys,i}$, P$_{sys,i}$
Yes
No
is number: m$_{sys,i}$, P$_{sys,i}$
Yes
No
m$_{sys,i}$>0
Yes
No
i=i+1
Export File (at the end of slot): systems_data.txt
Trigger: CheckPropBatt
Display (at the end of slot): error message
Stop (at the end of slot)

Figure 44 – Systems export operation flowchart

The calculation of the total mass and power input values is automatically made whenever a *lineEdit* field is edited and the content in all enabled fields are valid numbers. Following a successful calculation of the total mass of these systems, this value is also copied over to the Masses tab.

4.6 Propulsion

The subject of propulsion is very important to the optimization/analysis process of the LEEUAV. Editing of data for export in this section is spread through three main parts, all of them dependent on each other - battery data, motor/engine data, and data representing the propeller model.


Aerodynamics | Earth | Masses | Systems | Propulsion | Mission | Program Options | Finalization

Battery
- Battery Name: ;PS_APL_6S1P_21000mAh (optional)
- Cell Voltage: 3.95 V
- Cell Minimum Voltage: 3.3 V
- Cell Maximum Voltage: 4.2 V
- Cell Capacity: 10 Ah
- Cell Maximum Current: 150 A
- Cell Internal Resistance: 0.1e-2 Ω
- Single Cell Mass: 0.25 kg
- Number of cells in series: 3
- Number of cells in parallel: 1
- Battery Pack Mass: 0.75 kg
- Battery Pack Specific Energy: 158.0 Wh/kg

Engine/Motor
- Type of propulsion: Electric Motor
- Engine/Motor Name: ɔion SII-4025-520KV (optional)
- Number of engines/motors: 1
- Throttle Limit: 1.00
- Single Engine/Motor Mass: 0.353 kg
- Gear Ratio: 1
- Gear Efficiency: 1

Electric Motor
- Speed Constant: 520 rpm/V
- Torque Constant: ☐ Use 0.004 A/Nm
  *(at maximum power)
- Motor Internal Resistance: 0.009 Ω
- Motor ref. no load current: 1.4 A
- Motor ref. no load voltage: 10 V
  *(V for above given motor ref. no load current)

Propeller
- Type of propeller: Default polynomial curves - Cp(J,D,p), η(J,D,p)
- Propeller Name: ieneral 2-blade Prop (optional)
- Propeller diameter: 19.09 in
- Propeller Pitch: 15.43 in
- Number of propeller blades: 2
- Propeller Mass: 0.072 kg

Default polynomial curves - Cp(J,D,p), η(J,D,p)

The default values for propeller power and propulsive efficiency curves will be used

Figure 45 - Design of the Propulsion tab in the interface

4.6.1 Battery

The first part of the propulsion tab concerns representation of the battery used in the LEEUAV. Input of data in this part of the GUI is very straightforward. The following is the list of values required by the MPP:

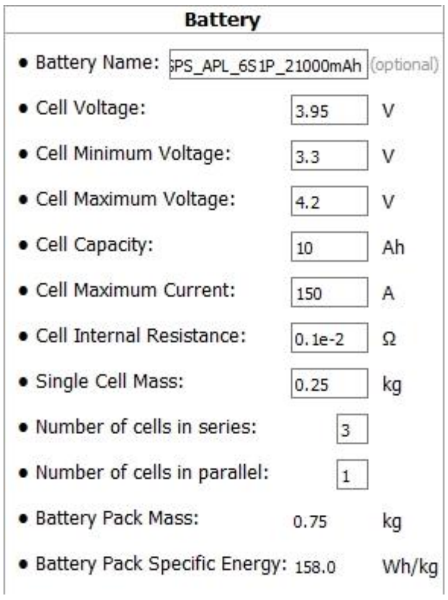

Battery
- Battery Name: ;PS_APL_6S1P_21000mAh (optional)
- Cell Voltage: 3.95 V
- Cell Minimum Voltage: 3.3 V
- Cell Maximum Voltage: 4.2 V
- Cell Capacity: 10 Ah
- Cell Maximum Current: 150 A
- Cell Internal Resistance: 0.1e-2 Ω
- Single Cell Mass: 0.25 kg
- Number of cells in series: 3
- Number of cells in parallel: 1
- Battery Pack Mass: 0.75 kg
- Battery Pack Specific Energy: 158.0 Wh/kg

Figure 46 – Closeup of the battery data section

- Values for nominal (U_{cell}), minimum ($U_{cell_{min}}$) and maximum ($U_{cell_{max}}$) cell voltages;
- Cell capacity (C_{cell});
- Cell maximum current ($I_{cell_{max}}$);
- Nominal cell resistance (R_{cell});
- Single cell mass (m_{cell});
- Number of cells in series ($n_{cells,series}$) and in parallel ($n_{cells,parallel}$).

Values for the battery pack mass (m_{batt}) and specific energy (u_{batt}) are automatically calculated as soon as valid entries into the number of cells in series ($n_{cells,series}$) and parallel ($n_{cells,parallel}$), cell voltage (U_{cell}), cell capacity (C_{cell}) and cell mass (m_{cell}) are introduced, via the following equations:

$$m_{batt} = n_{cells,series} \times n_{cells,parallel} \times m_{cell} \tag{3.29}$$

$$u_{batt} = \frac{U_{cell} \times C_{cell}}{m_{cell}} \tag{3.30}$$

As soon as a value of the battery pack mass is obtained, it is copied over to the Masses tab. During the export operation to a text file, values for the battery resistance (R_U), nominal (U_{batt}), minimum ($U_{batt_{min}}$) and maximum ($U_{batt_{max}}$) voltages, battery capacity (C_{batt}) and battery maximum current ($I_{batt_{max}}$) are also computed internally, by:

$$U_{batt} = n_{cells,series} \times U_{cell} \tag{3.31}$$

$$U_{batt_{min}} = n_{cells,series} \times U_{cell_{min}} \tag{3.32}$$

$$U_{batt_{max}} = n_{cells,series} \times U_{cell_{max}} \tag{3.33}$$

$$C_{batt} = n_{cells,parallel} \times C_{cell} \tag{3.34}$$

$$I_{batt_{max}} = n_{cells,parallel} \times I_{cell_{max}} \tag{3.35}$$

$$R_U = \frac{n_{cells,series} \times R_{cell}}{n_{cells,parallel}} \tag{3.36}$$

The export operation of battery data into the “propulsion_battery.txt” text file first involves the calling of the check function “checkPropBatt” to ensure all values were introduced (input of the battery name is optional). All *lineEdit* fields have built-in validators that only allow the input of positive numbers. The process flowchart can be seen in Figure 47.

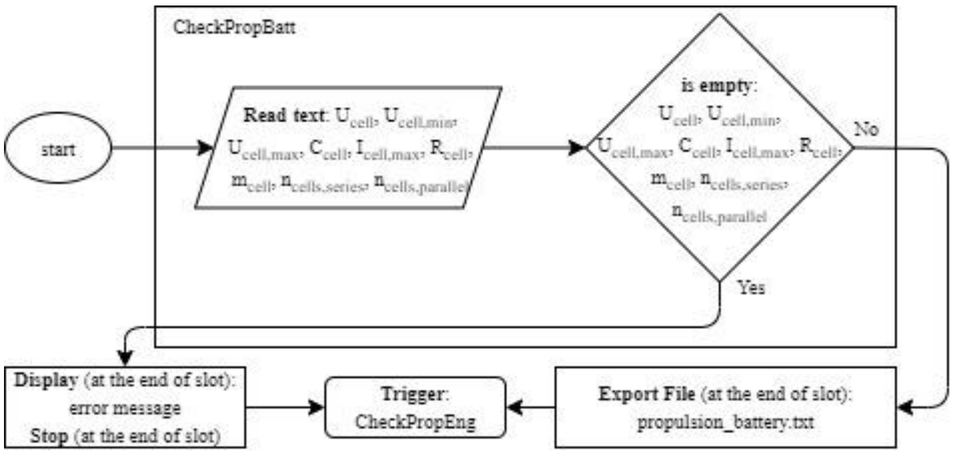
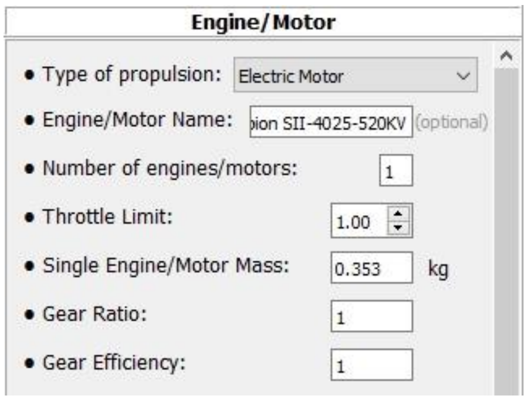

CheckPropBatt
start
Read text: U_{cell}, $U_{cell,min}$, $U_{cell,max}$, C_{cell}, $I_{cell,max}$, R_{cell}, m_{cell}, $n_{cells,series}$, $n_{cells,parallel}$
is empty: U_{cell}, $U_{cell,min}$, $U_{cell,max}$, C_{cell}, $I_{cell,max}$, R_{cell}, m_{cell}, $n_{cells,series}$, $n_{cells,parallel}$
No
Yes
Display (at the end of slot): error message Stop (at the end of slot)
Trigger: CheckPropEng
Export File (at the end of slot): propulsion_battery.txt

Figure 47 – Propulsion/Battery export operation flowchart

4.6.2 Engine/motor

The engine/motor part is located at the centre of the Propulsion tab. In it, one of two types of propulsion systems can be chosen: a combustion piston engine, or an electric motor. Afterwards, six editable general data fields proceed the "Type of propulsion" combo box:



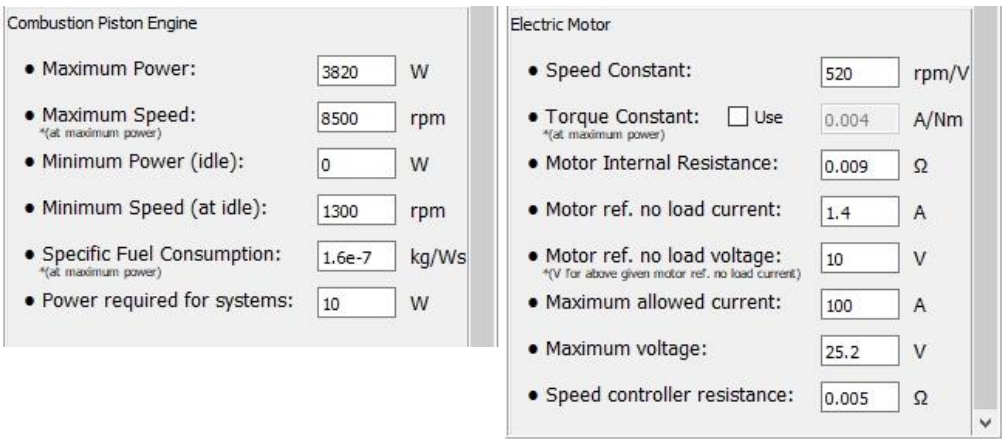
Engine/Motor

- Type of propulsion: Electric Motor
- Engine/Motor Name: ion SII-4025-520KV (optional)
- Number of engines/motors: 1
- Throttle Limit: 1.00
- Single Engine/Motor Mass: 0.353 kg
- Gear Ratio: 1
- Gear Efficiency: 1

Figure 48 – Closeup of the Engine/Motor upper part

- Name of the engine/motor, which is an optional entry;
- Number of engines/motors (n_{eng});
- The throttle limit of the LEEUAV (δ_{limit});
- Mass of a single engine/motor unit (m_{eng});
- Gearbox ratio (r_{gear}), which is set to 1 by default or if a gearbox is not present;
- Gearbox efficiency (η_{gear}), also set to 1 by default or if a gearbox is not present.

Depending on the chosen type of propulsion, different data to be edited appears in a stack widget below the general data. This is shown in Figure 49.



Combustion Piston Engine

- Maximum Power: 3820 W
- Maximum Speed: 8500 rpm
  *(at maximum power)
- Minimum Power (idle): 0 W
- Minimum Speed (at idle): 1300 rpm
- Specific Fuel Consumption: 1.6e-7 kg/Ws
  *(at maximum power)
- Power required for systems: 10 W

Electric Motor

- Speed Constant: 520 rpm/V
- Torque Constant: ☐ Use 0.004 A/Nm
  *(at maximum power)
- Motor Internal Resistance: 0.009 Ω
- Motor ref. no load current: 1.4 A
- Motor ref. no load voltage: 10 V
  *(V for above given motor ref. no load current)
- Maximum allowed current: 100 A
- Maximum voltage: 25.2 V
- Speed controller resistance: 0.005 Ω

Figure 49 – Closeup of the Engine/motor lower part. The left section shows combustion piston engine data, while the right section shows electric motor data

In case the combustion piston engine is chosen as the propulsion type, 6 values must be introduced:

- The maximum power of the engine (P_{max});
- Its maximum speed, at maximum power (N_{max});
- Minimum power at idle throttle (P_0);
- Minimum speed at idle throttle (N_0);
- Specific fuel consumption (sfc);
- Power required for systems ($P_{sys,eng}$).

In case the electric motor is chosen as the propulsion type, 7/8 values must be introduced:

- The motor speed constant (K_v);
- Torque constant (K_t) (if chosen to be used, in the checkbox);
- Motor internal resistance (R_{motor});
- Motor reference no-load current, representing current losses (I_0);
- "Motor reference no load voltage" (U_0), the voltage for the situation of no-load current;
- Maximum input current (I_{max});
- Maximum input voltage (U_{max});
- Electronic speed controller (ESC) resistance (R_{ESC}).

When the propulsion data is to be exported into the file "propulsion_engine.txt", a "checkPropEng" function checks if the relevant *lineEdit* fields are not empty. This is enough to ensure the validity of data, as all fields (except the engine/motor name) have built-in validators that only allow input of positive numbers (integers in the number of engines/motors). In either type of propulsion chosen, the other's set of data in the stack widget is irrelevant to this function – if, in it, a field is found to be empty, its default value will be exported instead. The flowchart in Figure 50 describes the export process.

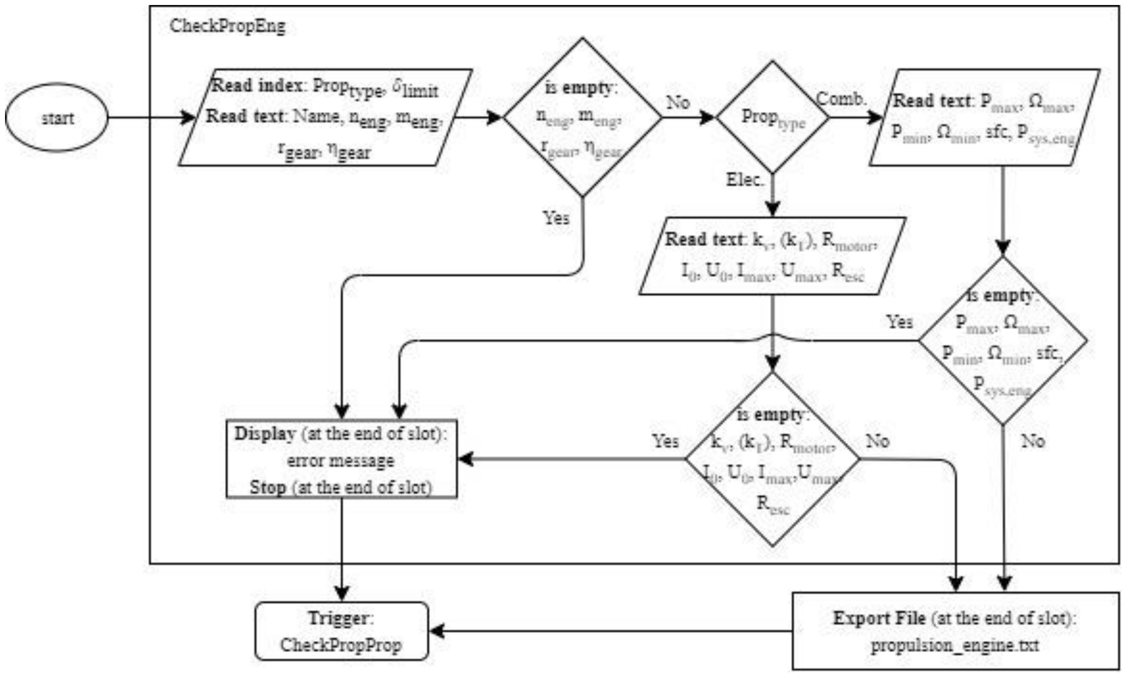
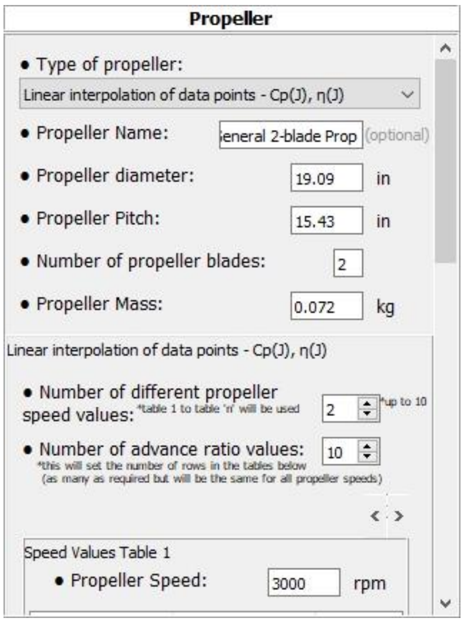

CheckPropEng
start
Read index: $Prop_{type}$, δ_{limit}
Read text: Name, n_{eng}, m_{eng}, r_{gear}, η_{gear}
is empty: n_{eng}, m_{eng}, r_{gear}, η_{gear}
No
Yes
$Prop_{type}$
Comb.
Elec.
Read text: P_{max}, Ω_{max}, P_{min}, Ω_{min}, sfc, $P_{sys,eng}$
Read text: k_v, (k_I), R_{motor}, I_0, U_0, I_{max}, U_{max}, R_{esc}
is empty: P_{max}, Ω_{max}, P_{min}, Ω_{min}, sfc, $P_{sys,eng}$
Yes
No
is empty: k_v, (k_I), R_{motor}, I_0, U_0, I_{max}, U_{max}, R_{esc}
Yes
No
Display (at the end of slot): error message
Stop (at the end of slot)
Trigger: CheckPropProp
Export File (at the end of slot): propulsion_engine.txt

Figure 50 - Propulsion/Engine export operation flowchart

4.6.3 Propeller

The last part of the Propulsion tab is the propeller section, where data of the LEEUAV's propeller must be introduced. Besides the input of the propeller's diameter (d), pitch (p), number of blades (n_{blades}) and mass (m_{prop}), one out of five representations must be chosen and edited for the calculation of the propeller's power coefficient (C_P) and efficiency (η_{prop}), as seen in section 3.1.4.



Propeller

- Type of propeller:

Linear interpolation of data points - Cp(J), η(J)

- Propeller Name: eneral 2-blade Prop (optional)
- Propeller diameter: 19.09 in
- Propeller Pitch: 15.43 in
- Number of propeller blades: 2
- Propeller Mass: 0.072 kg

Linear interpolation of data points - Cp(J), η(J)

- Number of different propeller speed values: *table 1 to table 'n' will be used 2 *up to 10
- Number of advance ratio values: 10

*this will set the number of rows in the tables below (as many as required but will be the same for all propeller speeds)

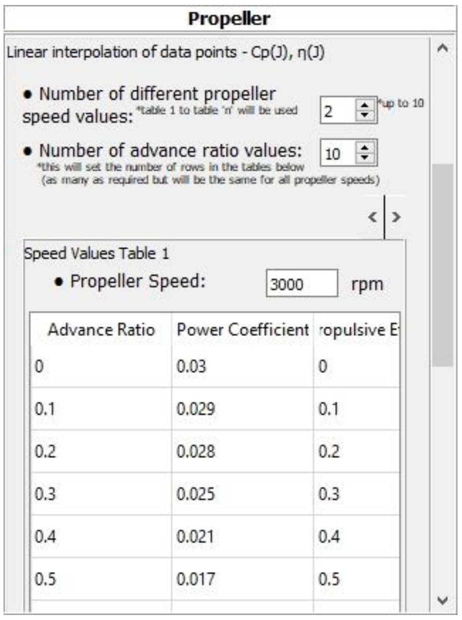
Speed Values Table 1

- Propeller Speed: 3000 rpm

Figure 51 - Closeup of the Propulsion/Propeller section

The representation is chosen in the “Type of propeller” combo box, and according to the type, a set of data is shown in a stack widget at the lower part, below the “Propeller mass” *lineEdit* field. The following is a description of how data is introduced in the stack widget, for each representation:

- Linear interpolation of data points ($C_p(J)$, $\eta_{prop}(J)$) - a certain number of tables with specified propeller speeds (N) must be filled with values of advance ratio (J), power coefficient (C_P) and propulsive efficiency (η_{prop}) in each line. The number of tables, as well as the number of lines in each table (“number of advance ratio values”) are specified in two spin boxes in the upper part of the stack widget;
- Least squares polynomial approximation of data points ($C_p(J)$, $\eta_{prop}(J)$) - similar to the above representation, except only one table for a single propeller speed (N) is filled;
- Polynomial representation from user coefficients ($C_p(J)$, $\eta_{prop}(J)$) - a vector of coefficients for either $C_P(J)$ and $\eta_{prop}(J)$ polynomials must be introduced;
- Approximation using default polynomial curves ($C_p(J,d,p)$, $\eta_{prop}(J,d,p)$) - the default model, described in Annex B;
- Approximation using user polynomial curves ($C_p(J,d,p)$, $\eta_{prop}(J,d,p)$) - coefficient vectors for $J_{max}(d,p)$, $C_{P_0}(d,p)$, $\eta_{max}(d,p)$, $C_P(C_{P_0},J,J_{max})$ and $\eta(\eta_{max},J,J_{max})$ polynomials must be introduced.



Propeller

Linear interpolation of data points - Cp(J), η(J)

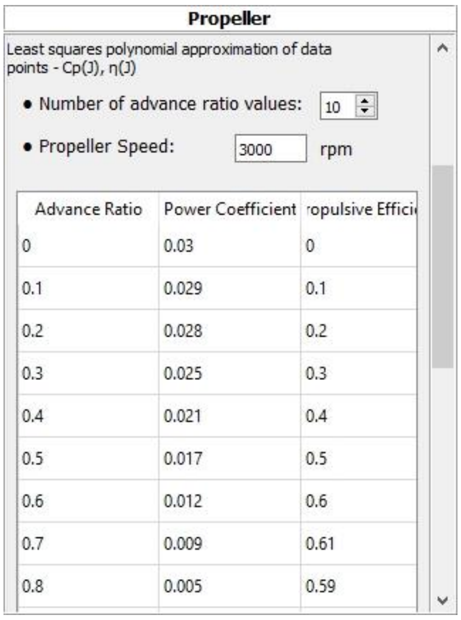
- Number of different propeller speed values: *table 1 to table 'n' will be used — 2 — *up to 10
- Number of advance ratio values: 10
  *this will set the number of rows in the tables below (as many as required but will be the same for all propeller speeds)

Speed Values Table 1

- Propeller Speed: 3000 rpm

Advance Ratio	Power Coefficient	ropulsive E
0	0.03	0
0.1	0.029	0.1
0.2	0.028	0.2
0.3	0.025	0.3
0.4	0.021	0.4
0.5	0.017	0.5

Figure 52 – Linear interpolation of data points, in the Propulsion/Propeller section



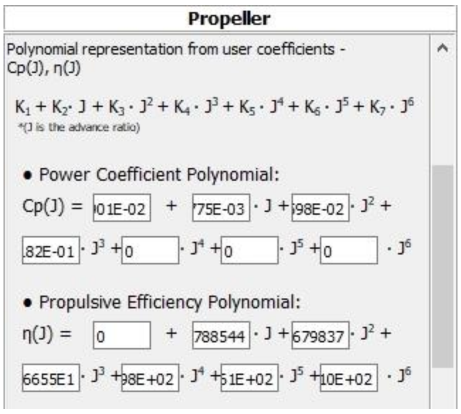
Propeller

Least squares polynomial approximation of data points - Cp(J), η(J)

- Number of advance ratio values: 10
- Propeller Speed: 3000 rpm

Advance Ratio	Power Coefficient	ropulsive Effici
0	0.03	0
0.1	0.029	0.1
0.2	0.028	0.2
0.3	0.025	0.3
0.4	0.021	0.4
0.5	0.017	0.5
0.6	0.012	0.6
0.7	0.009	0.61
0.8	0.005	0.59

Figure 53 – Least squares polynomial approximation of data points, in the Propulsion/Propeller section



Propeller

Polynomial representation from user coefficients - Cp(J), η(J)

$K_1 + K_2 \cdot J + K_3 \cdot J^2 + K_4 \cdot J^3 + K_5 \cdot J^4 + K_6 \cdot J^5 + K_7 \cdot J^6$
*(J is the advance ratio)

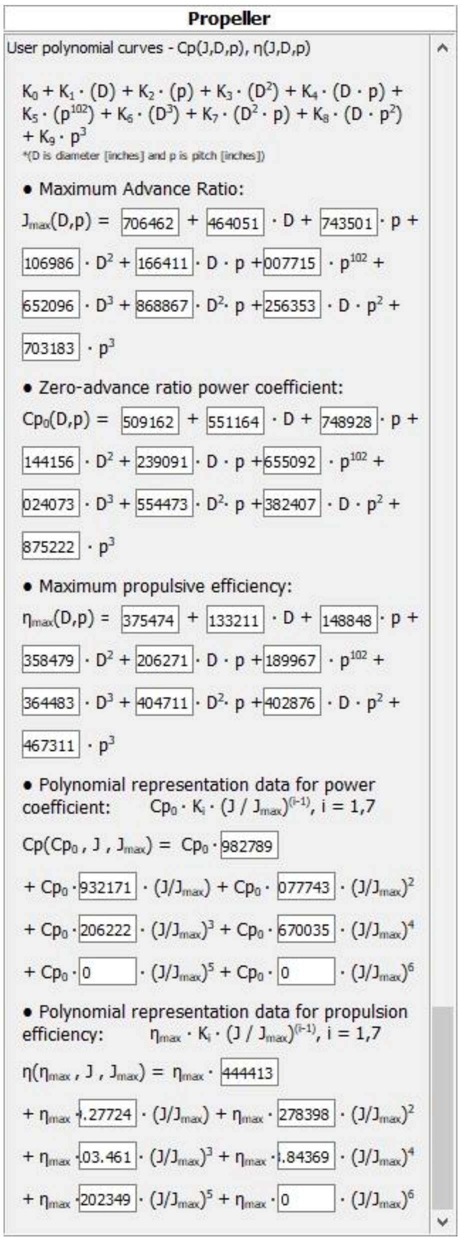
- Power Coefficient Polynomial:

Cp(J) = 01E-02 + 75E-03 · J + 98E-02 · J^2 + .82E-01 · J^3 + 0 · J^4 + 0 · J^5 + 0 · J^6

- Propulsive Efficiency Polynomial:

η(J) = 0 + 788544 · J + 679837 · J^2 + 6655E1 · J^3 + 98E+02 · J^4 + 51E+02 · J^5 + 10E+02 · J^6

Figure 54 – Polynomial representation from user coefficients, in the Propulsion/Propeller section



Propeller

User polynomial curves - Cp(J,D,p), η(J,D,p)

$K_0 + K_1 \cdot (D) + K_2 \cdot (p) + K_3 \cdot (D^2) + K_4 \cdot (D \cdot p) + K_5 \cdot (p^{102}) + K_6 \cdot (D^3) + K_7 \cdot (D^2 \cdot p) + K_8 \cdot (D \cdot p^2) + K_9 \cdot p^3$
*(D is diameter [inches] and p is pitch [inches])

- Maximum Advance Ratio:

J_{max}(D,p) = 706462 + 464051 · D + 743501 · p + 106986 · D^2 + 166411 · D · p + 007715 · p^{102} + 652096 · D^3 + 868867 · D^2 · p + 256353 · D · p^2 + 703183 · p^3

- Zero-advance ratio power coefficient:

Cp_0(D,p) = 509162 + 551164 · D + 748928 · p + 144156 · D^2 + 239091 · D · p + 655092 · p^{102} + 024073 · D^3 + 554473 · D^2 · p + 382407 · D · p^2 + 875222 · p^3

- Maximum propulsive efficiency:

η_{max}(D,p) = 375474 + 133211 · D + 148848 · p + 358479 · D^2 + 206271 · D · p + 189967 · p^{102} + 364483 · D^3 + 404711 · D^2 · p + 402876 · D · p^2 + 467311 · p^3

- Polynomial representation data for power coefficient: $Cp_0 \cdot K_i \cdot (J / J_{max})^{(i-1)}$, i = 1,7

Cp(Cp_0 , J , J_{max}) = Cp_0 · 982789
+ Cp_0 · 932171 · (J/J_{max}) + Cp_0 · 077743 · $(J/J_{max})^2$
+ Cp_0 · 206222 · $(J/J_{max})^3$ + Cp_0 · 670035 · $(J/J_{max})^4$
+ Cp_0 · 0 · $(J/J_{max})^5$ + Cp_0 · 0 · $(J/J_{max})^6$

- Polynomial representation data for propulsion efficiency: $\eta_{max} \cdot K_i \cdot (J / J_{max})^{(i-1)}$, i = 1,7

η(η_{max} , J , J_{max}) = η_{max} · 444413
+ η_{max} · .27724 · (J/J_{max}) + η_{max} · 278398 · $(J/J_{max})^2$
+ η_{max} · 03.461 · $(J/J_{max})^3$ + η_{max} · .84369 · $(J/J_{max})^4$
+ η_{max} · 202349 · $(J/J_{max})^5$ + η_{max} · 0 · $(J/J_{max})^6$

Figure 55 – User polynomial curves representation, in the Propulsion/Propeller section

Upon the valid entry of the mass of the propeller, its value is automatically copied over to the Masses tab.

As happened in all previous sections, a function “checkPropProp” is responsible for ensuring the validity of Propeller data, when the export operation is triggered by the user. Though most of the *lineEdit* fields have built-in validators that only allow input of values (positive real numbers in diameter, pitch, mass and propeller speed; positive integers in the number of blades), this function must check if all fields are not empty. Moreover, if one of the first two propeller representations are chosen, the check function must ensure all items in the active tables are valid positive real numbers.

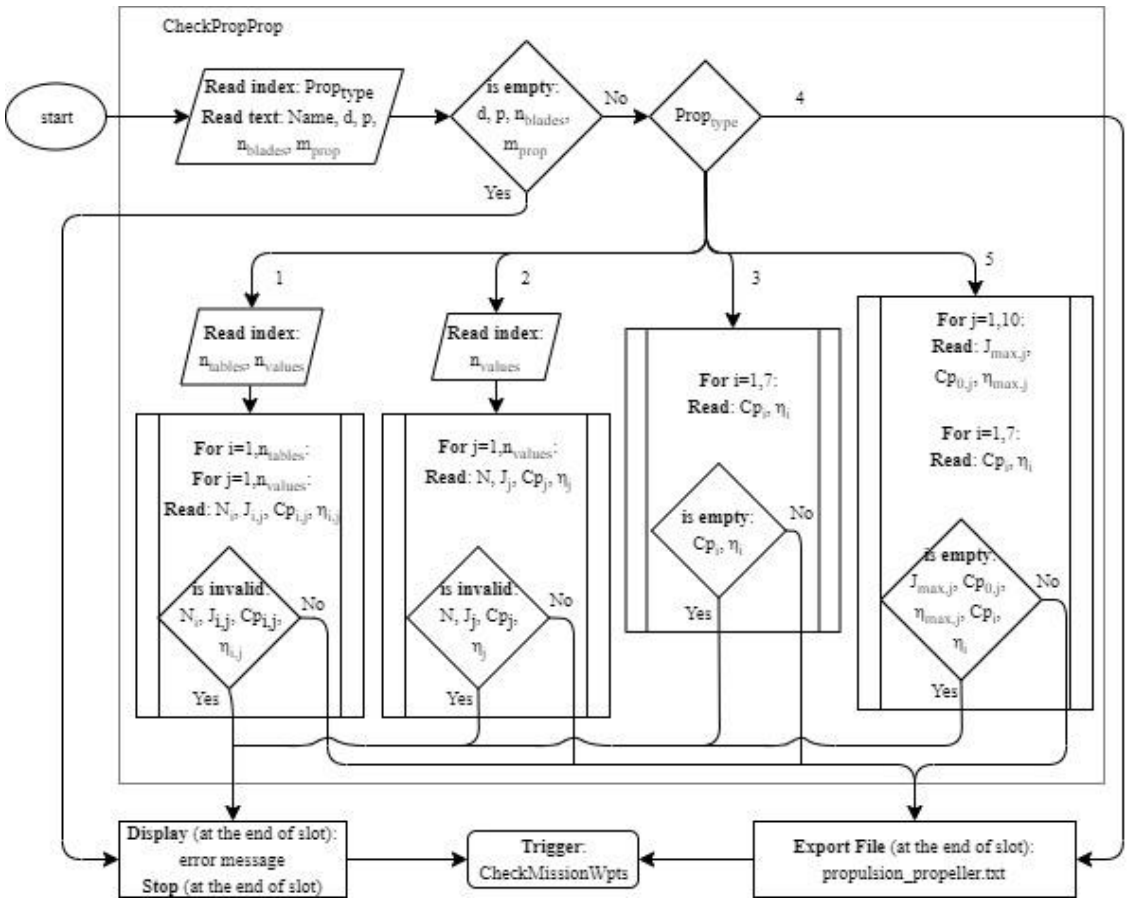
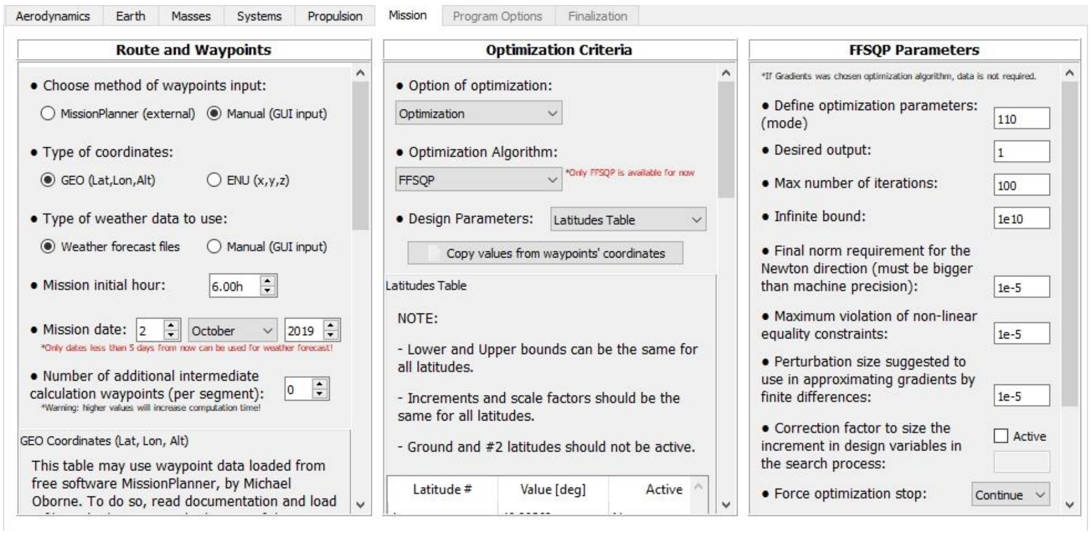

CheckPropProp
start
Read index: $Prop_{type}$
Read text: Name, d, p, n_{blades}, m_{prop}
is empty: d, p, n_{blades}, m_{prop}
No
Yes
$Prop_{type}$
4
1
2
3
5
Read index: n_{tables}, n_{values}
Read index: n_{values}
For i=1,n_{tables}: For j=1,n_{values}: Read: N_i, $J_{i,j}$, $Cp_{i,j}$, $\eta_{i,j}$
is invalid: N_i, $J_{i,j}$, $Cp_{i,j}$, $\eta_{i,j}$
For j=1,n_{values}: Read: N, J_j, Cp_j, η_j
is invalid: N, J_j, Cp_j, η_j
For i=1,7: Read: Cp_i, η_i
is empty: Cp_i, η_i
For j=1,10: Read: $J_{max,j}$, $Cp_{0,j}$, $\eta_{max,j}$
For i=1,7: Read: Cp_i, η_i
is empty: $J_{max,j}$, $Cp_{0,j}$, $\eta_{max,j}$, Cp_i, η_i
Display (at the end of slot): error message
Stop (at the end of slot)
Trigger: CheckMissionWpts
Export File (at the end of slot): propulsion_propeller.txt

Figure 56 - Propulsion/Propeller export operation flowchart

4.7 Mission

The Mission tab involves the editing and exporting of data from the input route to be analysed/optimized, design parameters, equality/inequality constraints, algorithm information and generic optimization options, such as the objective function. Three files are required by the MPP, therefore, three sections were also designed in the GUI: Route and Waypoints, Optimization Criteria and FFSQP Parameters.


Aerodynamics | Earth | Masses | Systems | Propulsion | Mission | Program Options | Finalization

Route and Waypoints

- Choose method of waypoints input: MissionPlanner (external) [✓ Manual (GUI input)]
- Type of coordinates: [✓ GEO (Lat,Lon,Alt)] ENU (x,y,z)
- Type of weather data to use: [✓ Weather forecast files] Manual (GUI input)
- Mission initial hour: 6.00h
- Mission date: 2 October 2019

*Only dates less than 5 days from now can be used for weather forecast!

- Number of additional intermediate calculation waypoints (per segment): 0

*Warning: higher values will increase computation time!

GEO Coordinates (Lat, Lon, Alt)

This table may use waypoint data loaded from free software MissionPlanner, by Michael Oborne. To do so, read documentation and load

Optimization Criteria

- Option of optimization: Optimization
- Optimization Algorithm: FFSQP *Only FFSQP is available for now
- Design Parameters: Latitudes Table

Copy values from waypoints' coordinates

Latitudes Table

NOTE:

- Lower and Upper bounds can be the same for all latitudes.

- Increments and scale factors should be the same for all latitudes.

- Ground and #2 latitudes should not be active.

Latitude #	Value [deg]	Active

FFSQP Parameters

*If Gradients was chosen optimization algorithm, data is not required.

Define optimization parameters: (mode)	110
Desired output:	1
Max number of iterations:	100
Infinite bound:	1e10
Final norm requirement for the Newton direction (must be bigger than machine precision):	1e-5
Maximum violation of non-linear equality constraints:	1e-5
Perturbation size suggested to use in approximating gradients by finite differences:	1e-5
Correction factor to size the increment in design variables in the search process:	Active
Force optimization stop:	Continue

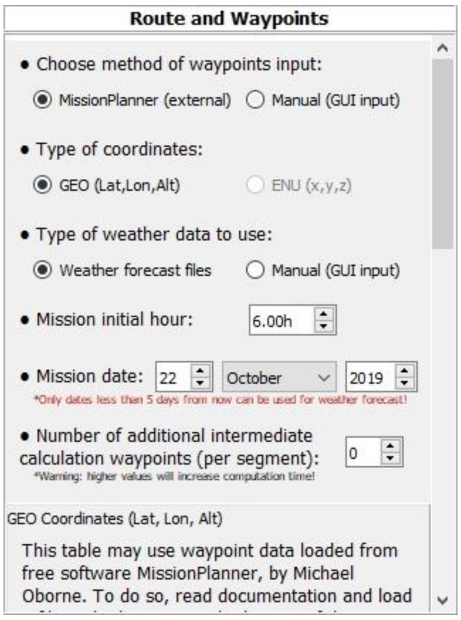
Figure 57 - Design of the Mission tab in the interface

4.7.1 Route and Waypoints

This is a common starting point in the input of data in the interface, and as the name suggests, the route to be analysed/optimized is introduced here. This section is divided in two parts: general data in the upper part, and the waypoints' stack widget at the lower part (which includes the waypoints table and loiter table).

In the upper part, the following data is edited:

- "Method of waypoints input" – specifies the input method of the route, either externally from MissionPlanner (explained in section 4.9.1) or internally (manual);
- "Type of coordinates" – choosing from either Geodetic (GEO) or East-North-Up (ENU), each type of coordinates displays the corresponding page in the waypoints' stack widget at the lower part. ENU coordinates are only available if the "Method of waypoints input" is checked to manual;
- "Type of weather input" – allows the user to instruct the MPP to use downloaded forecast files (generated at the Weather tab) or data from manual input in the waypoints table;
- "Mission initial hour" and "Mission date" – self-evident. These values are used as the reference date (h) in the added Mission Initial Hour design parameter (sections 3.2 and 4.7.2) and in the calculation of forecasts in the Weather tab (section 4.3.2);
- "Number of additional intermediate calculation waypoints (per segment)" – self-evident. Higher values specified will severely increase the MPP's overall run time.


Route and Waypoints

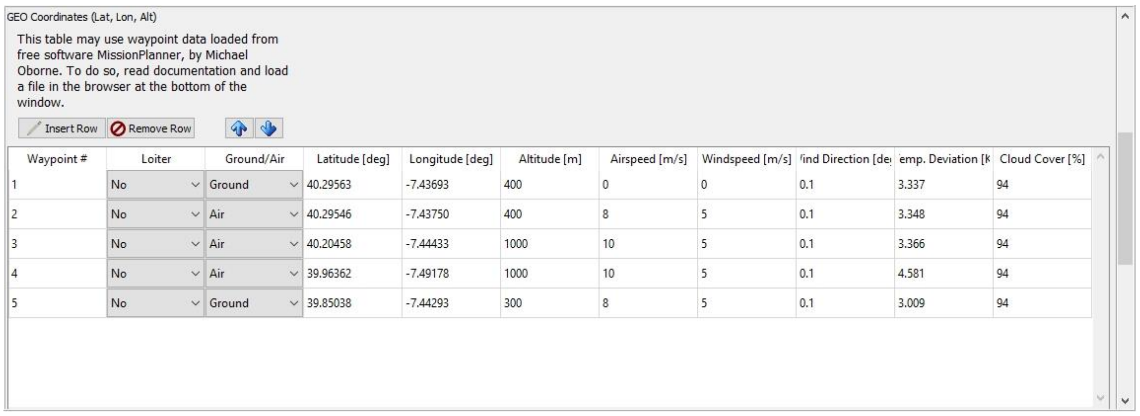
- Choose method of waypoints input: [✓ MissionPlanner (external)] Manual (GUI input)
- Type of coordinates: [✓ GEO (Lat,Lon,Alt)] ENU (x,y,z)
- Type of weather data to use: [✓ Weather forecast files] Manual (GUI input)
- Mission initial hour: 6.00h
- Mission date: 22 October 2019
  *Only dates less than 5 days from now can be used for weather forecast!
- Number of additional intermediate calculation waypoints (per segment): 0
  *Warning: higher values will increase computation time!

GEO Coordinates (Lat, Lon, Alt)

This table may use waypoint data loaded from free software MissionPlanner, by Michael Oborne. To do so, read documentation and load

Figure 58 - Closeup of the Mission/Route and Waypoints upper section

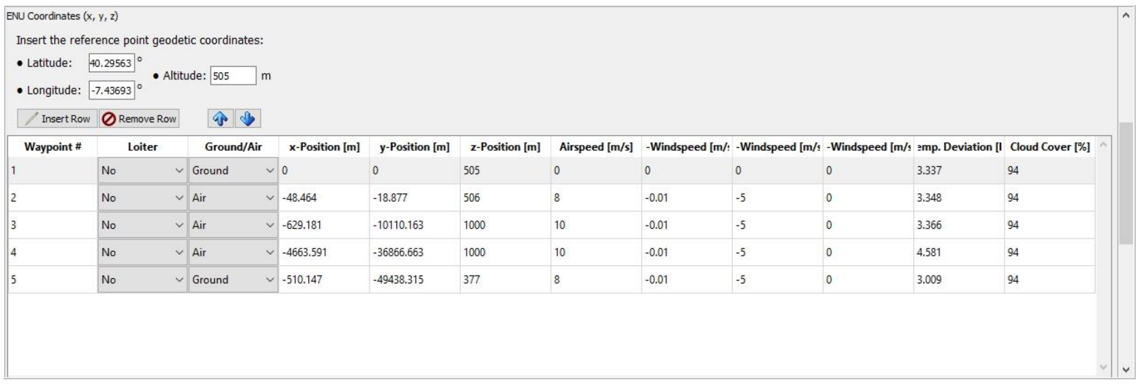
The lower part is where the waypoints' stack widget is located, containing both a waypoints and loiters table. An expanded view of both coordinates types' waypoints tables is presented in Figures 59 and 60.


GEO Coordinates (Lat, Lon, Alt)

This table may use waypoint data loaded from free software MissionPlanner, by Michael Oborne. To do so, read documentation and load a file in the browser at the bottom of the window.

Waypoint #	Loiter	Ground/Air	Latitude [deg]	Longitude [deg]	Altitude [m]	Airspeed [m/s]	Windspeed [m/s]	'ind Direction [de	emp. Deviation [K	Cloud Cover [%]
1	No	Ground	40.29563	-7.43693	400	0	0	0.1	3.337	94
2	No	Air	40.29546	-7.43750	400	8	5	0.1	3.348	94
3	No	Air	40.20458	-7.44433	1000	10	5	0.1	3.366	94
4	No	Air	39.96362	-7.49178	1000	10	5	0.1	4.581	94
5	No	Ground	39.85038	-7.44293	300	8	5	0.1	3.009	94

Figure 59 – Example of a route in the GEO coordinates waypoints table


ENU Coordinates (x, y, z)

Insert the reference point geodetic coordinates:

- Latitude: 40.29563 °
- Longitude: -7.43693 °
- Altitude: 505 m

Waypoint #	Loiter	Ground/Air	x-Position [m]	y-Position [m]	z-Position [m]	Airspeed [m/s]	-Windspeed [m/:	-Windspeed [m/s	-Windspeed [m/s	emp. Deviation [I	Cloud Cover [%]
1	No	Ground	0	0	505	0	0	0	0	3.337	94
2	No	Air	-48.464	-18.877	506	8	-0.01	-5	0	3.348	94
3	No	Air	-629.181	-10110.163	1000	10	-0.01	-5	0	3.366	94
4	No	Air	-4663.591	-36866.663	1000	10	-0.01	-5	0	4.581	94
5	No	Ground	-510.147	-49438.315	377	8	-0.01	-5	0	3.009	94

Figure 60 – Same route as in figure 59, ENU coordinates waypoints table

Each line in the tables represents a waypoint, which can be specified to have or not loiter, and to be or not ground waypoints, in combo boxes at the 2nd and 3rd columns, respectively.

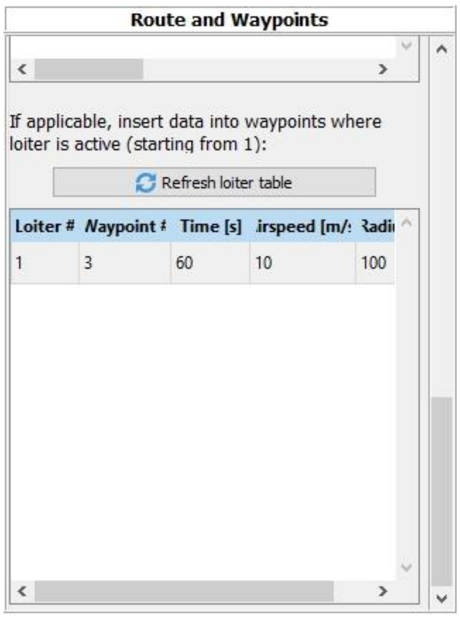
Ground and #2 waypoints are fixed in the optimization process. In other words, they do not have active latitude, longitude, or altitude design parameters, while the airspeed design parameter is only inactive in the first waypoint (see section 4.7.2). First and last waypoints must always be "ground" and not have loiter. Data for the proceeding columns is summarized in Table 1.

If the user chooses to input a route manually, he/she can add, remove or move up and down waypoints using buttons above the table. Moreover, in the case of ENU coordinates, an additional reference point with geodetic coordinates must be provided to the interface. If the route is loaded from a MissionPlanner file, the user is also presented the choice to set the reference point's coordinates with the ones from the "Home" point set in the MissionPlanner's imported route.

Table 1 – Variables for GEO and ENU waypoint tables

Waypoints Table Variables			
GEO	**Description [Units]**	**ENU**	**Description [Units]**
Latitude	$[deg]$	x-position	From reference point $[m]$
Longitude	$[deg]$	y-position	From reference point $[m]$
Altitude	Above MSL (Absolute) $[m]$	z-position	Above MSL $[m]$
Airspeed	True airspeed $[m/s]$	Airspeed	True airspeed $[m/s]$
		x-Windspeed	True airspeed $[m/s]$
Windspeed	True airspeed $[m/s]$	y-Windspeed	True airspeed $[m/s]$
Wind direction	True heading $[deg]$	z-Windspeed	True airspeed $[m/s]$
Temperature Deviation	From ISA $[°C]$	Temperature Deviation	From ISA $[°C]$
Cloud Cover	0 to 100 $[\%]$	Cloud Cover	0 to 100 $[\%]$

In case a loiter is added or removed from the waypoints table, the loiters table must be updated, by clicking the "Refresh loiter table". An example is shown in Figure 61.



Route and Waypoints

If applicable, insert data into waypoints where loiter is active (starting from 1):

Loiter #	Waypoint #	Time [s]	irspeed [m/:	Radi
1	3	60	10	100

Figure 61 – Closeup of the loiters table

Each line in this table represents a single loiter. The first column indicates the loiter's order, while the second indicates the waypoint it is located at. The rest of the columns have the following data to be filled:

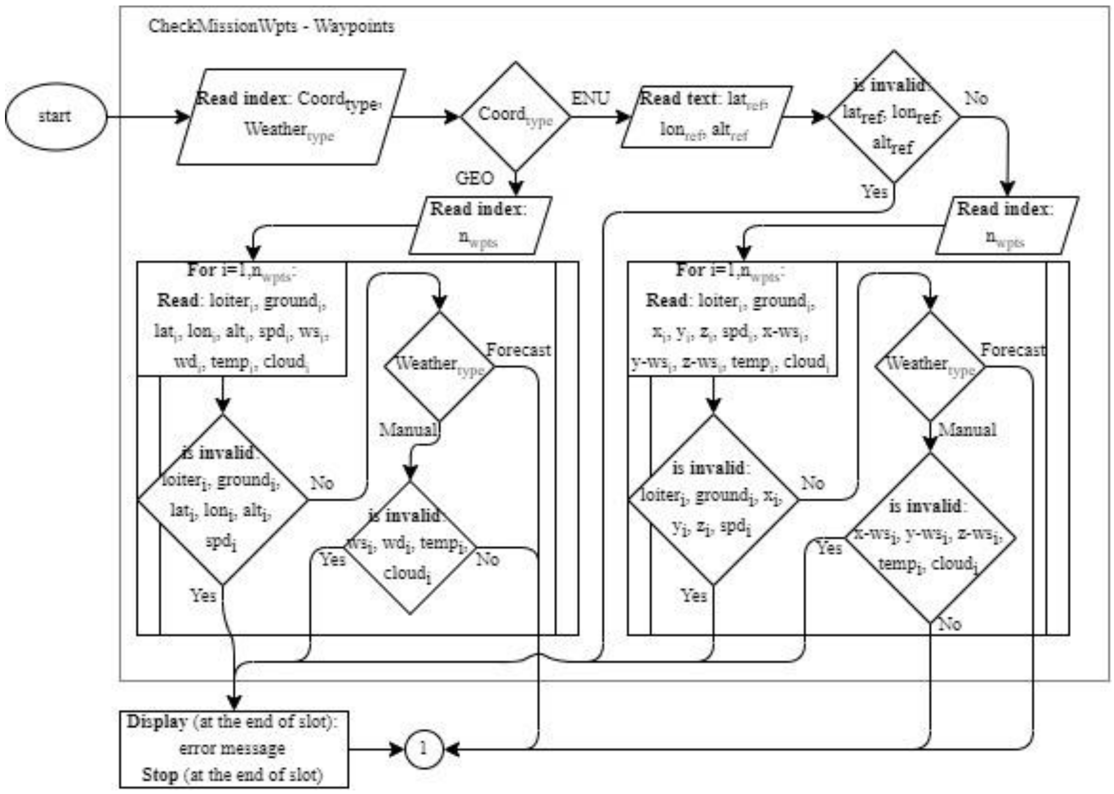
- Time – how long will the LEEUAV circle around. A positive value will make it circle clockwise, while a negative one will make it circle counter clockwise;
- Airspeed, at which the LEEUAV will circle around;

- Radius, of the loiter circle.

In the "Export files" operation slot, the "checkMissionWpts" function is called to check the validity of data in this section. As one of the most complex in the GUI, this function checks if any required field of data is empty or invalid, and these required fields are determined depending on the user's choice of the "Type of coordinates" and "Type of weather data to use".

If the GEO type of coordinates is chosen, "checkMissionWpts" will only check tables from the GEO page of the waypoints stack widget. The same applies to ENU coordinates. Meanwhile, if the user chooses to use weather from forecast files, this function will ignore the last 4 (GEO) or 5 (ENU) columns regarding the manual input of weather.

Furthermore, if the number of active loiters in the waypoints table is different from that of the loiters table, or any waypoint numbers are not coincidental, or any field is empty or invalid, the check function will also prevent the export operation and display an error at the end of the slot. The flowcharts in Figures 62 and 63 describe the check-export process of this section, with the final generated file being "mission_waypoints.txt".



CheckMissionWpts - Waypoints
start
Read index: Coord_type, Weather_type
Coord_type
ENU
GEO
Read text: lat_ref, lon_ref, alt_ref
is invalid: lat_ref, lon_ref, alt_ref
No
Yes
Read index: n_wpts
For i=1,n_wpts:
Read: loiter_i, ground_i, lat_i, lon_i, alt_i, spd_i, ws_i, wd_i, temp_i, cloud_i
is invalid: loiter_i, ground_i, lat_i, lon_i, alt_i, spd_i
Weather_type
Forecast
Manual
is invalid: ws_i, wd_i, temp_i, cloud_i
Read: loiter_i, ground_i, x_i, y_i, z_i, spd_i, x-ws_i, y-ws_i, z-ws_i, temp_i, cloud_i
is invalid: loiter_i, ground_i, x_i, y_i, z_i, spd_i
is invalid: x-ws_i, y-ws_i, z-ws_i, temp_i, cloud_i
Display (at the end of slot): error message
Stop (at the end of slot)
1

Figure 62 – Mission/Route and Waypoints export operation flowchart #1

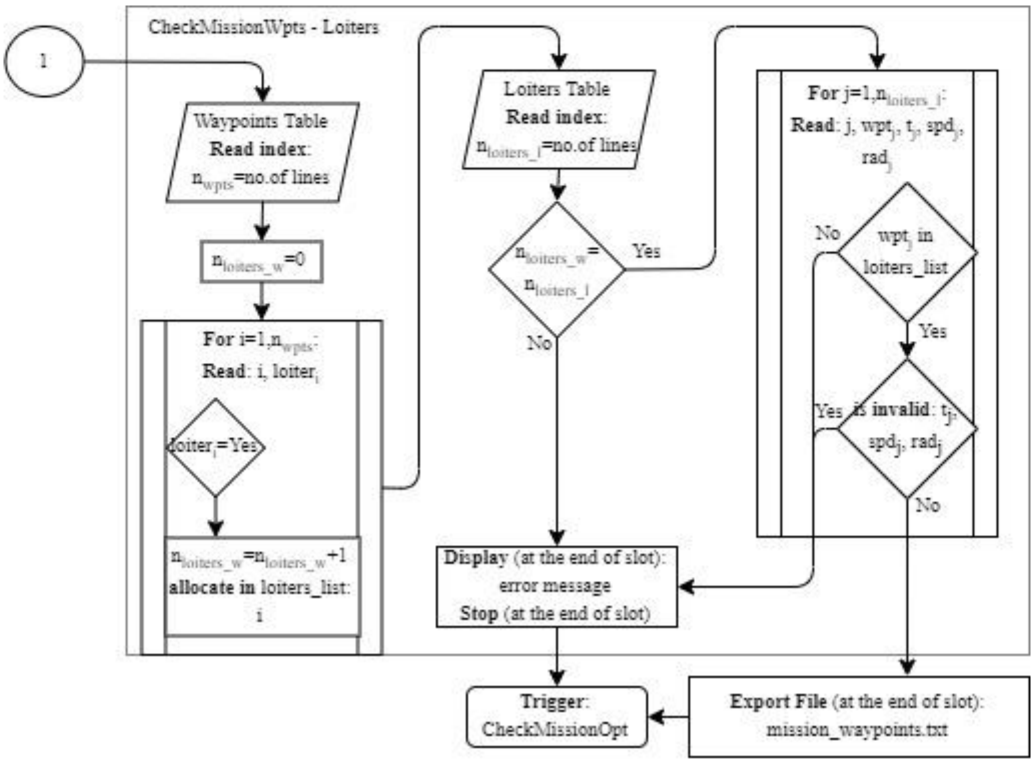

CheckMissionWpts - Loiters
1
Waypoints Table
Read index:
n_{wpts}=no.of lines
$n_{loiters\_w}$=0
For i=1,n_{wpts}:
Read: i, loiter$_i$
loiter$_i$=Yes
$n_{loiters\_w}$=$n_{loiters\_w}$+1
allocate in loiters_list: i
Loiters Table
Read index:
$n_{loiters\_l}$=no.of lines
$n_{loiters\_w}$= $n_{loiters\_l}$
Yes
No
For j=1,$n_{loiters\_l}$:
Read: j, wpt$_j$, t$_j$, spd$_j$, rad$_j$
wpt$_j$ in loiters_list
No
Yes
is invalid: t$_j$, spd$_j$, rad$_j$
Yes
No
Display (at the end of slot): error message
Stop (at the end of slot)
Trigger: CheckMissionOpt
Export File (at the end of slot): mission_waypoints.txt

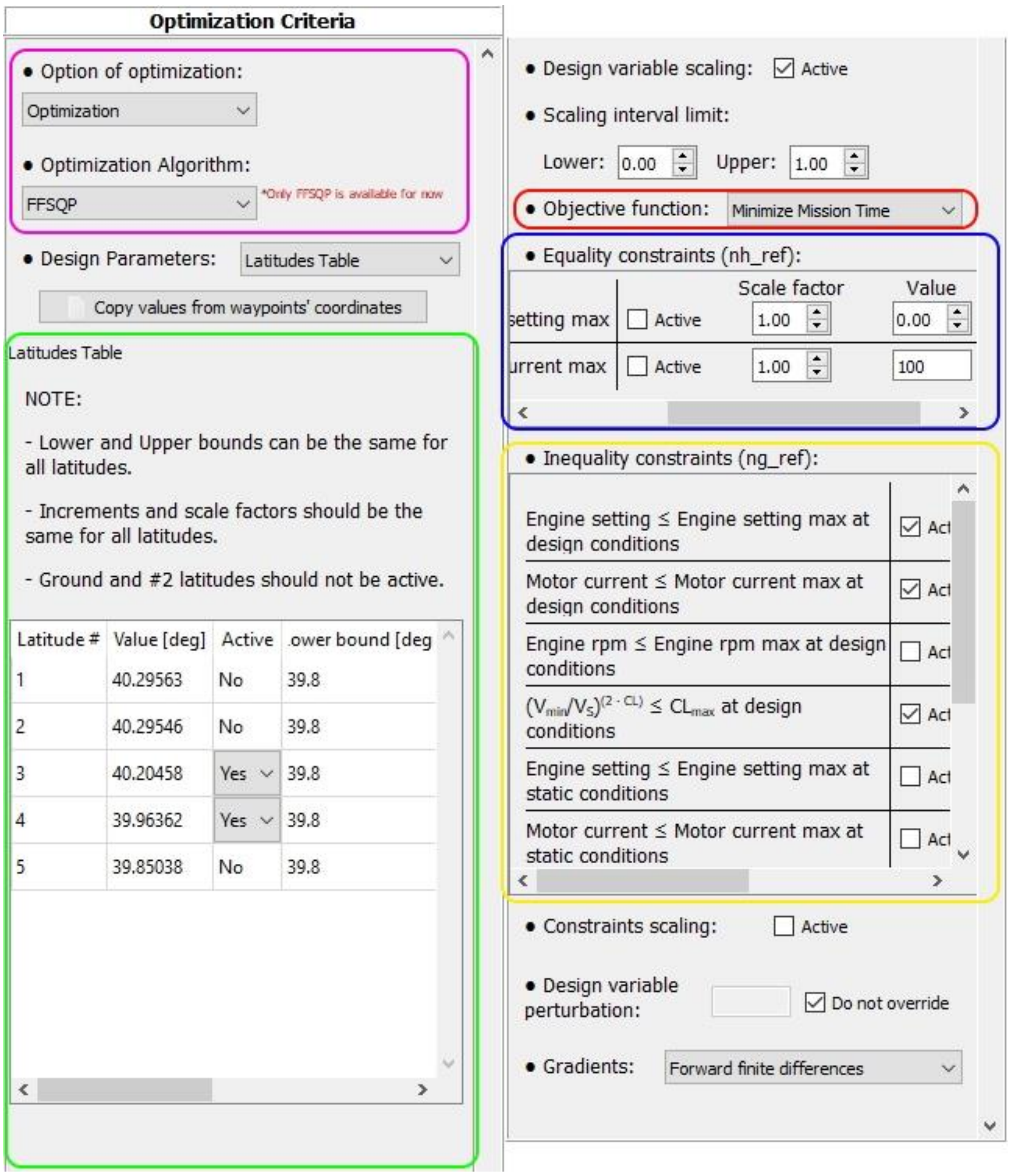
Figure 63 - Mission/Route and Waypoints export operation flowchart #2

4.7.2 Optimization

The optimization section allows the user to edit data relative to the way the Mission Planner Program is going to run, in particular its optimization method. The following is a list of variables needed for the MPP:

- Option of optimization – the two methods that are available – analysis only, or optimization;
- Optimization algorithm – the choice of algorithm to be used in the optimization (FFSQP is set as default);
- Design parameters – 5 types of design parameters are present: latitudes, longitudes, altitudes, airspeeds and the mission initial hour. These parameters are edited in 5 tables, which become visible in a stack widget according to the “Design parameters” combo box displayed option. Special instructions/restrictions are shown in notes above each table, for a correct introduction of values.
- Design variable scaling – specify if the scale factor values in the tables are active or not;
- Scaling interval limit – interval limit for the scaling;
- Objective Function – one of three objective functions can be chosen: Minimize Energy, Minimize Time or Minimize Distance;
- Equality Constraints – the equality constraints limit the optimization of the design parameters to certain values. The user specifies if the constraint is active in a checkbox, a scale factor, and a valid value. Up to two can be currently used;

- Inequality Constraints - the inequality constraints limit the optimization of the design parameters to above or below certain values. The user specifies if the constraint is active in a checkbox, a scale factor, and a valid value. Up to ten can be currently used;
- Constraints scaling – specify if scale factors in the constraints are to be used or not;
- Design variable perturbation – instructs the MPP to override (or not) the approximation of gradients by finite differences with a specified value;
- Gradients – specifies the type of finite differences to use during convergence of values.


Optimization Criteria

- Option of optimization: Optimization
- Optimization Algorithm: FFSQP *Only FFSQP is available for now
- Design Parameters: Latitudes Table

Copy values from waypoints' coordinates

Latitudes Table

NOTE:

- Lower and Upper bounds can be the same for all latitudes.

- Increments and scale factors should be the same for all latitudes.

- Ground and #2 latitudes should not be active.

Latitude #	Value [deg]	Active	ower bound [deg
1	40.29563	No	39.8
2	40.29546	No	39.8
3	40.20458	Yes	39.8
4	39.96362	Yes	39.8
5	39.85038	No	39.8

- Design variable scaling: ☑ Active
- Scaling interval limit:

Lower: 0.00 Upper: 1.00

- Objective function: Minimize Mission Time
- Equality constraints (nh_ref):

		Scale factor	Value
etting max	☐ Active	1.00	0.00
rrent max	☐ Active	1.00	100

- Inequality constraints (ng_ref):

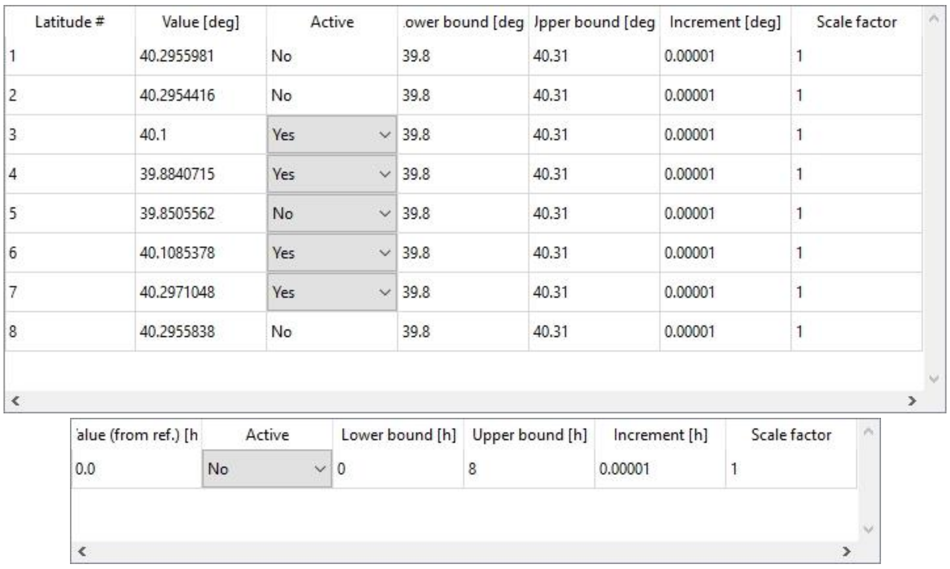
Engine setting ≤ Engine setting max at design conditions	☑ Act
Motor current ≤ Motor current max at design conditions	☑ Act
Engine rpm ≤ Engine rpm max at design conditions	☐ Act
$(V_{min}/V_S)^{(2 \cdot CL)} \leq CL_{max}$ at design conditions	☑ Act
Engine setting ≤ Engine setting max at static conditions	☐ Act
Motor current ≤ Motor current max at static conditions	☐ Act

- Constraints scaling: ☐ Active
- Design variable perturbation: ☑ Do not override
- Gradients: Forward finite differences

Figure 64 – Expanded view of the Mission/Optimization section. Outlined are the optimization options (pink), design parameters stack widget (green), objective function (red), equality constraints (blue) and inequality constraints (yellow)

A line in every design parameter table (except Mission Initial Hour) represents a waypoint of the input route, with its design parameter value in the 2nd column. Before anything can be edited however, the button just above the stack widget, “Copy values from waypoints’ coordinates” must be clicked in order to update all tables with the correct number of lines and values from the Route and Waypoints section. If the waypoints table was filled with ENU coordinates, the copying function automatically converts them to GEO coordinates. The function “checkMissionWpts” is also called, ensuring that data in the Route and Waypoints section was correctly introduced. If not, the copying operation is halted with an error message.

Following the values’ column, the 3rd column indicates whether the design parameter is active or not. By updating the tables, the first, second and last waypoints automatically have their latitude, longitude and altitude design parameters set to inactive, and only the first waypoint’s airspeed design parameter is set to inactive, for the optimization. The rest of data is divided into four columns: Lower bound, the lowest value that the parameter can get during the optimization; Upper bound, the highest value that the parameter can get during the optimization; Increment, to be used while changing values during the optimization; and Scale Factor, used to scale the design parameter value for a balanced optimization (kept as 1 by default). Figure 65 illustrates examples of the design parameter tables used.



Latitude #	Value [deg]	Active	ower bound [deg	Jpper bound [deg	Increment [deg]	Scale factor
1	40.2955981	No	39.8	40.31	0.00001	1
2	40.2954416	No	39.8	40.31	0.00001	1
3	40.1	Yes	39.8	40.31	0.00001	1
4	39.8840715	Yes	39.8	40.31	0.00001	1
5	39.8505562	No	39.8	40.31	0.00001	1
6	40.1085378	Yes	39.8	40.31	0.00001	1
7	40.2971048	Yes	39.8	40.31	0.00001	1
8	40.2955838	No	39.8	40.31	0.00001	1

'alue (from ref.) [h	Active	Lower bound [h]	Upper bound [h]	Increment [h]	Scale factor
0.0	No	0	8	0.00001	1

Figure 65 – Expanded view of Latitude design parameter table (similar to longitude, altitude and airspeed tables) (upper) and mission initial hour design parameter table (lower)

The design parameter value in the mission initial hour assumes a value of 0 by default, meaning it will use the reference defined in section 4.7.1 and adjust it (if active) between the lower and upper bounds. However, the user can change its optimization reference for better results.

Like in all previous sections, when the “Export files” slot is triggered, the data for this section is checked with the function “checkMissionOpt” before it is exported into the file “mission_optimization.txt” for the MPP to use. This function checks that all tables do not have empty or invalid fields, the number of waypoints, as well as its values, are the same as the waypoints table in the Route and Waypoints section (4.7.1), and that the "ground" waypoints are not active design parameters. It also checks, for all active constraints, and if the "Design variable perturbation" is set to override, that their "Value" *lineEdit* fields have valid numbers. This operation is summarized in the flowchart of Figure 66.

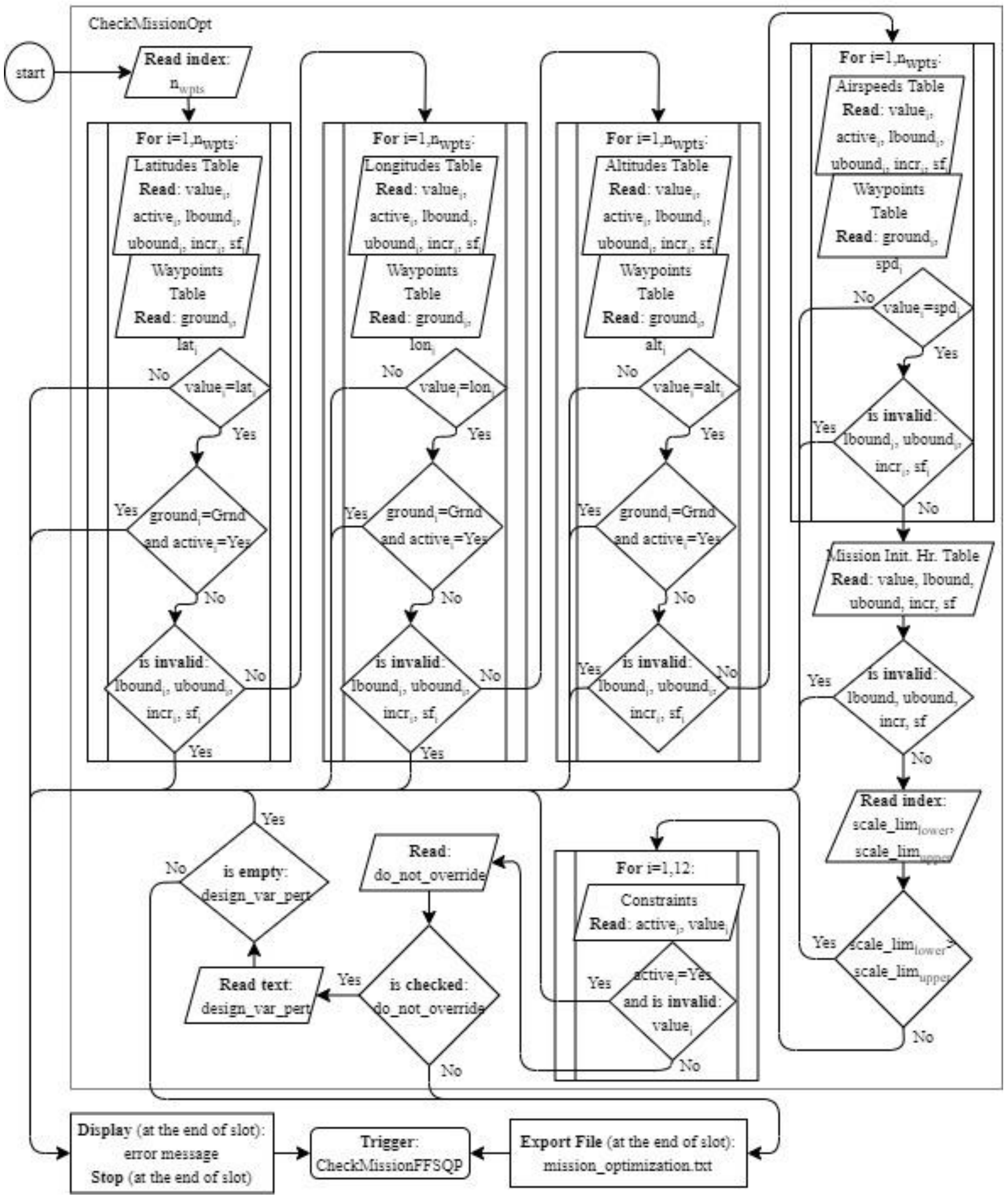
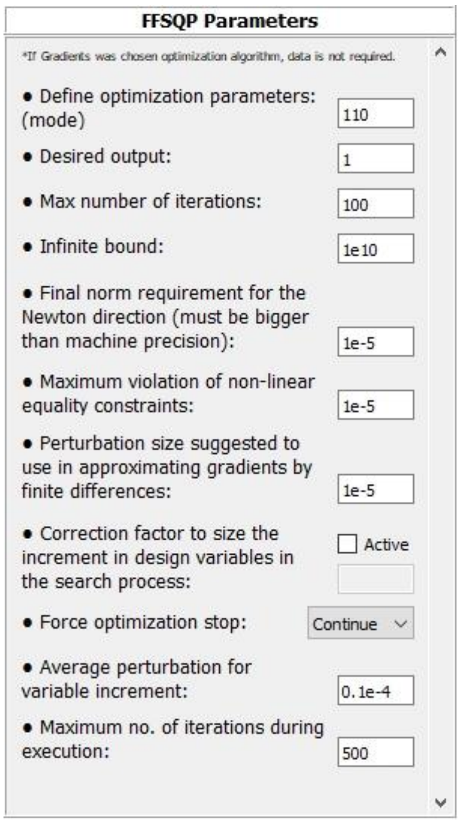

CheckMissionOpt
start
Read index: n_{wpts}
For i=1,n_{wpts}:
Latitudes Table Read: value$_i$, active$_i$, lbound$_i$, ubound$_i$, incr$_i$, sf$_i$
Waypoints Table Read: ground$_i$, lat$_i$
value$_i$=lat$_i$
ground$_i$=Grnd and active$_i$=Yes
is invalid: lbound$_i$, ubound$_i$, incr$_i$, sf$_i$
Longitudes Table Read: value$_i$, active$_i$, lbound$_i$, ubound$_i$, incr$_i$, sf$_i$
Waypoints Table Read: ground$_i$, lon$_i$
value$_i$=lon$_i$
Altitudes Table Read: value$_i$, active$_i$, lbound$_i$, ubound$_i$, incr$_i$, sf$_i$
Waypoints Table Read: ground$_i$, alt$_i$
value$_i$=alt$_i$
Airspeeds Table Read: value$_i$, active$_i$, lbound$_i$, ubound$_i$, incr$_i$, sf$_i$
Waypoints Table Read: ground$_i$, spd$_i$
value$_i$=spd$_i$
Mission Init. Hr. Table Read: value, lbound, ubound, incr, sf
is invalid: lbound, ubound, incr, sf
Read index: scale_lim$_{lower}$, scale_lim$_{upper}$
scale_lim$_{lower}$> scale_lim$_{upper}$
For i=1,12:
Constraints Read: active$_i$, value$_i$
active$_i$=Yes and is invalid: value$_i$
Read: do_not_override
is checked: do_not_override
Read text: design_var_pert
is empty: design_var_pert
Yes
No
Display (at the end of slot): error message Stop (at the end of slot)
Trigger: CheckMissionFFSQP
Export File (at the end of slot): mission_optimization.txt

Figure 66 – Mission/Optimization export operation flowchart

4.7.3 FFSQP

The FFSQP section contains data essential to the functionality of the FFSQP algorithm used in the optimization method.



FFSQP Parameters

*If Gradients was chosen optimization algorithm, data is not required.

- Define optimization parameters: (mode) 110
- Desired output: 1
- Max number of iterations: 100
- Infinite bound: 1e10
- Final norm requirement for the Newton direction (must be bigger than machine precision): 1e-5
- Maximum violation of non-linear equality constraints: 1e-5
- Perturbation size suggested to use in approximating gradients by finite differences: 1e-5
- Correction factor to size the increment in design variables in the search process: ☐ Active
- Force optimization stop: Continue
- Average perturbation for variable increment: 0.1e-4
- Maximum no. of iterations during execution: 500

Figure 67 – Closeup of the Mission/FFSQP section

Data edited is this section includes:

- "Optimization parameters (mode)" – specifies the optimization mode, with an assumed C-B-A number structure, the meaning of which is defined in Table 2, according to [52];

Table 2 – Modes of Optimization values table

Modes of Optimization		
Number	**Value**	**Job Options**
C	1	During line search, the function that rejected the previous step size is checked first; all functions of the same type ("objective" or "constraints") as the latter will then be checked first
	2	All constraints will be checked first at every trial point during the line search
B	0	Monotone decrease of objective function after each iteration
	1	Monotone decrease of objective function after at most 4 iterations
A	0	Ordinary minimax problems
	1	Ordinary minimax problems with each individual function replaced by its absolute value, i.e. an L_{infty} problem

- “Desired output” – print level indicator ($iprint$). According to its value, the following options apply, as described in [52]: 0 - “no normal output except error information (this option is imposed during phase 1)”; 1 - “a final printout at a local solution”; 2 - “a brief printout at the end of each iteration”; 3 - “a detailed information is printed out at the end of each iteration for debugging purposes”; $10 \times N + M$ - “N any positive integer, $M = 2$ or 3. Information corresponding to $iprint = M$ will be displayed at every $(10 \times N)$th iteration at the last iteration”;
- “Maximum number of iterations” ($miter$), allowed to solve the problem;
- “Infinite bound” ($bigbnd$) – assumes a positive infinite value;
- “Final norm requirement for the Newton direction” (eps) – stopping criterion that ensures, at a solution, that the norm of the Newton direction vector is smaller than the specified value;
- “Maximum violation of non-linear equality constraints” ($epseqn$) - the tolerance of the violation of non-linear equality constraints allowed by the user at an optimal solution;
- “Perturbation size suggested to use in approximating gradients by finite differences” ($udelta$) – perturbation size in computing gradients by finite differences. The true perturbation is determined by $sign(x^i) \times \max\{udelta, rteps \times \max(1, |x^i|)\}$ for each component x^i of x ($rteps$ is the square root of the machine precision). Should be set to 0 if the user has no idea how to choose it;
- “Correction factor to size the increment in design variables in the search process” ($ufactor$) – self-explanatory. User may check this correction factor to be used or not in a checkbox (if not used, the correction factor is set to a value of -1, internally).
- “Force optimization stop” – forces a controlled stop of the optimization. This is not used at the start, and “Continue” should be chosen at all times. The manual controlled stop of the MPP from the GUI is a feature not yet developed;
- “Average perturbation for variable increment” ($delta$) - perturbation size when variable increments are used;
- “Maximum no. of iterations during execution” ($countmax$) – the convergence process will stop if a solution is repeated more than the specified times.

Data from this section is exported to the file “mission_ffsqp.txt”, when the “Export files” slot is triggered and the function “checkMissionFFSQP” verifies the validity of all data. This function simply ensures that no active *lineEdit* fields are empty. The “mode”, $miter$, and $countmax$ variables have built-in validators that only allow positive integers, while the rest of variables accept all sorts of values, positive and negative. Still in terms of input of data, this section demands a greater level of attention from the user than the rest of the GUI. The check function before export is summarized in the flowchart of Figure 68.

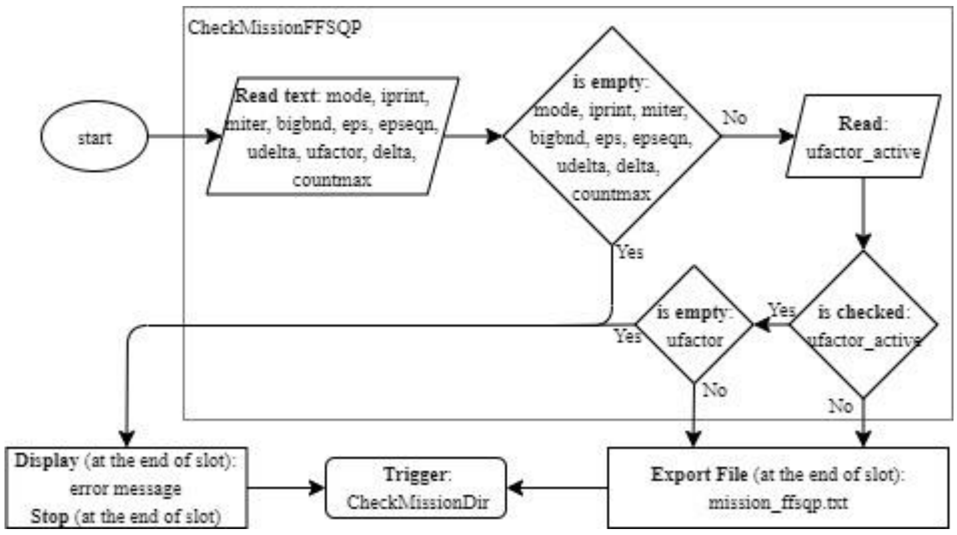

CheckMissionFFSQP
start
Read text: mode, iprint, miter, bigbnd, eps, epseqn, udelta, ufactor, delta, countmax
is empty: mode, iprint, miter, bigbnd, eps, epseqn, udelta, delta, countmax
No
Read: ufactor_active
Yes
is empty: ufactor
Yes
is checked: ufactor_active
Yes
No
No
Display (at the end of slot): error message Stop (at the end of slot)
Trigger: CheckMissionDir
Export File (at the end of slot): mission_ffsqp.txt

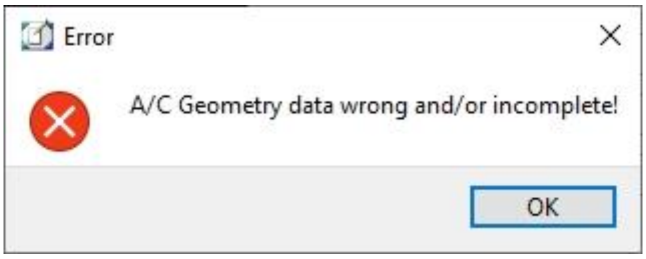
Figure 68 - Mission/FFSQP export operation flowchart

4.8 Functioning Structure

Following input in all 6 tabs of the central tab widget and directory fields of the bottom section, the user is able to export the data, and if so wishes, to run the Mission Planner Program. These operations are both triggered by clicking the “Export and Run” button, whereas the “Export Files” button only triggers the export operation. Hence, the basic functioning structure of the Mission Optimization Interface is built around two main operation slots: “Export Files”, made up by the “check” functions and the “run” functions (that export files); and “Export and Run”, that calls “Export Files” and proceeds to run the MPP.

4.8.1 Exporting Files

There are 13 “check” and 13 “run” functions, which corresponds to the total number of text files exported. Each check function returns a “stop” variable at its end, where a value of 1 corresponds to a successful validation, and a value of 0 corresponds to an unsuccessful validation. If the product of the 13 “stop” check values is 0, the operation triggered by the user stops, and all errors sent by check functions of affected sections appear. Otherwise, the export part of the operation – the “run” functions – may commence.


Error

A/C Geometry data wrong and/or incomplete!

Figure 69 – Example of an error box showing the affected section

At each “run” function’s end, a signal is sent to update the progress bar at the bottom section, reaching 100% when all files will have been exported. The “Export Files” slot is summarized in the flowchart of Figure 70.

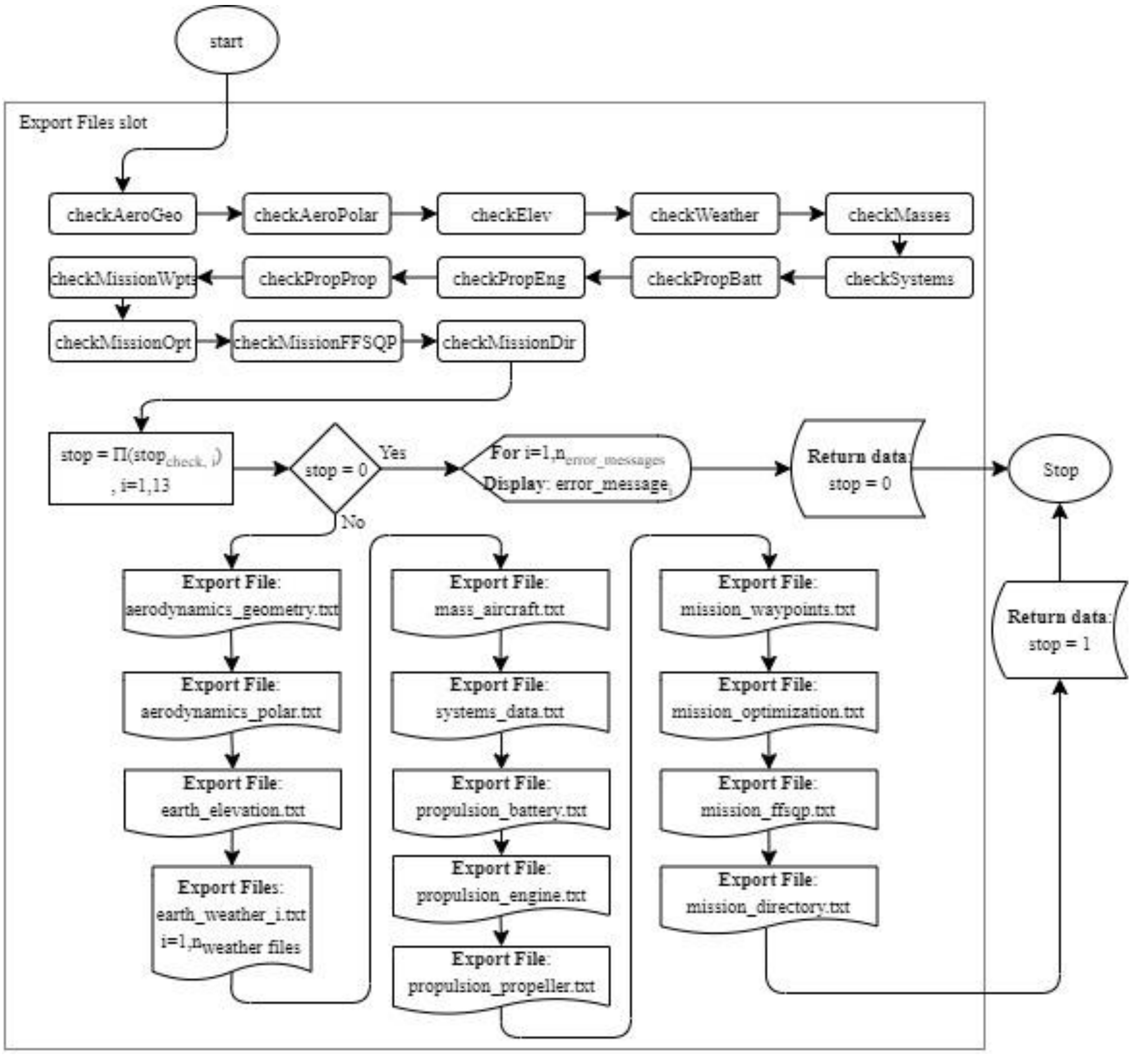

start
Export Files slot
checkAeroGeo
checkAeroPolar
checkElev
checkWeather
checkMasses
checkSystems
checkPropBatt
checkPropEng
checkPropProp
checkMissionWpts
checkMissionOpt
checkMissionFFSQP
checkMissionDir
stop = Π(stop_check, i), i=1,13
stop = 0
Yes
No
For i=1,n_error_messages Display: error_message_i
Return data: stop = 0
Stop
Export File: aerodynamics_geometry.txt
Export File: aerodynamics_polar.txt
Export File: earth_elevation.txt
Export Files: earth_weather_i.txt i=1,n_weather files
Export File: mass_aircraft.txt
Export File: systems_data.txt
Export File: propulsion_battery.txt
Export File: propulsion_engine.txt
Export File: propulsion_propeller.txt
Export File: mission_waypoints.txt
Export File: mission_optimization.txt
Export File: mission_ffsqp.txt
Export File: mission_directory.txt
Return data: stop = 1

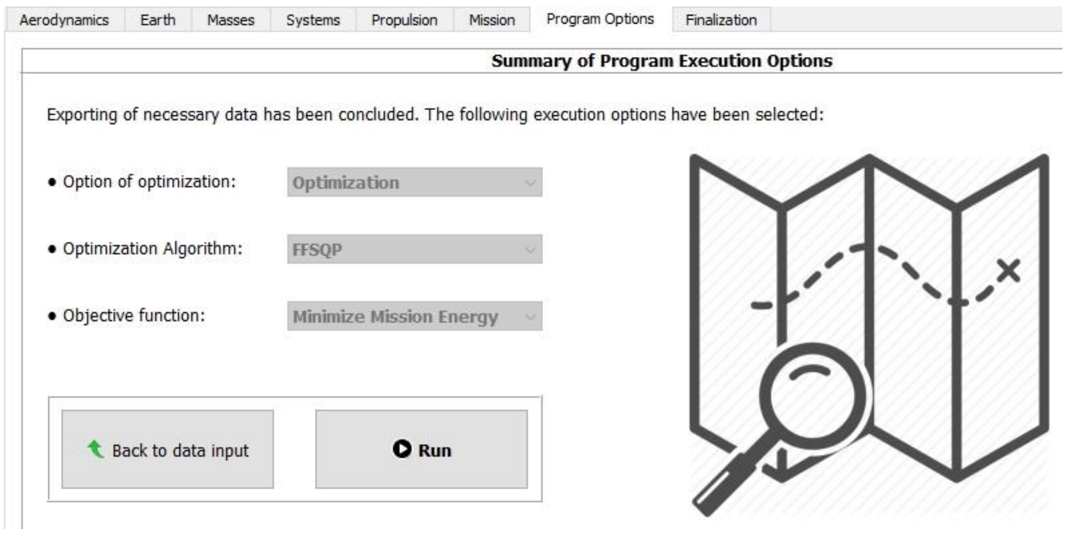
Figure 70 – “Export Files” slot operation flowchart

The function “checkMissionDir” checks that the folders, the directories of which specified at the bottom section – “Mission Optimizer Main Directory” and, if applicable, “Save data and files to project folder” – and needed files, such as “Mission.exe”, exist. If successfully checked, the directory texts are exported to the file “mission_directory.txt”.

4.8.2 Running the MPP

The “Export and Run” slot is triggered by clicking the button of the same name, at the bottom area section of the interface. The “Export Files” slot is then called, and in case the return value “stop” of that function is 1, then the MPP can be started. Otherwise, if it is 0, the GUI stops the operation, because the check system in section 4.8.1 detected errors in data input. This system prevents crashes of the GUI or MPP, as well as bad optimization runs.

At the end of the export process, all 6 previous tabs are locked and a new tab is shown – "Program Options" – where three basic execution options are displayed: option of optimization, optimization algorithm and objective function (set in section 4.7.2). This is the last chance for the user to edit data and cancel the start of the MPP, which can be done by clicking the button "Back to data input". Otherwise, the MPP is started by clicking the "Run" button.


Summary of Program Execution Options

Exporting of necessary data has been concluded. The following execution options have been selected:

- Option of optimization: Optimization
- Optimization Algorithm: FFSQP
- Objective function: Minimize Mission Energy

Figure 71 - GUI's Program Options tab

By clicking the "Run" button, the GUI starts the "Mission.exe" file in the program's main directory, and automatically inputs the option with which to run the program, using virtual keyboard strokes. Because of the limitations of this interface system, it is important to keep the GUI and then the "Mission.exe" window up front in the computer screen, otherwise the key strokes are applied elsewhere and the optimization is not successfully started.

Depending on the amount or type of input data and optimization options, the "optimizer" may take from some minutes to several hours to finish. For the interface to know when that happens, a loop function checks at every second if a "dummy_checker.txt" file is present in the executable's main directory, because that file is created by the MPP when it finishes (see section 3.2).

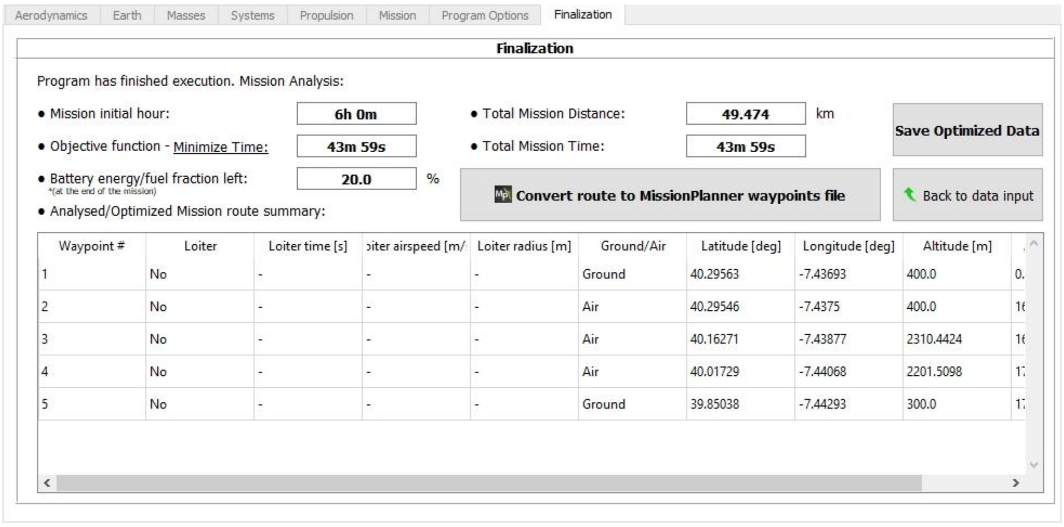
If by any reason the executable's window is unknowingly or manually closed, the GUI stops the loop check function and displays an error message informing about the situation. Upon this, it does not read any output data files.

4.8.3 Finalization

After the MPP finishes and creates the "dummy_checker.txt" file, the executable's window is closed and the checker file is deleted by the GUI. Afterwards, a new tab called "Finalization" is open, and one of three situations may then happen:

1. The Mission Planner Program successfully optimized a route and created the output files “mission_out.txt” and “mission_convergence.txt” with no errors;
2. The Mission Planner Program could not find an optimized solution for the input route, and only the “mission_out.txt” analysis file was created, with no errors;
3. The Mission Planner Program could not find an optimized solution, neither could successfully analyse the input route, in which case only the “mission_out.txt” output file was created, with errors.

In a normal situation (1), a message will be displayed, warning the user to save the results. The output data is retrieved from “mission_out.txt” and “mission_convergence.txt”, and displayed in several fields:



Finalization

Program has finished execution. Mission Analysis:

- Mission initial hour: 6h 0m
- Objective function - Minimize Time: 43m 59s
- Battery energy/fuel fraction left: 20.0 %
  *(at the end of the mission)
- Analysed/Optimized Mission route summary:
- Total Mission Distance: 49.474 km
- Total Mission Time: 43m 59s

Waypoint #	Loiter	Loiter time [s]	oiter airspeed [m/	Loiter radius [m]	Ground/Air	Latitude [deg]	Longitude [deg]	Altitude [m]	.
1	No	-	-	-	Ground	40.29563	-7.43693	400.0	0.
2	No	-	-	-	Air	40.29546	-7.4375	400.0	1(
3	No	-	-	-	Air	40.16271	-7.43877	2310.4424	1(
4	No	-	-	-	Air	40.01729	-7.44068	2201.5098	1:
5	No	-	-	-	Ground	39.85038	-7.44293	300.0	1:

Figure 72 – GUI’s Finalization tab

- Mission initial hour – depending on whether the respective design parameter was active or not, the calculated or fixed initial hour is displayed in this field;
- Objective function – the obtained optimized value for the chosen objective function is displayed here;
- Battery energy/fuel fraction left – indicates the remaining fraction of energy left at the end of the mission;
- Analysed/Optimized Mission route summary – a table that displays the new optimized route, or the input route;
- Total Mission Distance – always shows the calculated distance of the route;
- Total Mission Time – displays the calculated or optimized duration of the mission.

These results can then be saved, via the “Save Optimized Data” button, into a .dat file, which is readable by the action “Read saved flight data optimization results” in the file menu. The new route is also displayed in the table at the lower part, and can be converted to a MissionPlanner’ readable .waypoints file, using the button “Convert route to MissionPlanner waypoints file” (explained in section 4.9.2).

In case situation 2 happens, the GUI will display an error message, informing the user that a solution was not found in the optimization process. Only data from the analysis process is loaded into the fields above, the objective function field will display the calculated used energy instead, and the table at the lower part will display the input route. The results can still be saved into a .dat file.

In case situation 3 happens, the GUI will not display any data at all, and no export/save actions are possible.

The user can then go back to data editing (from the present session, and not from .dat results files) using the button “Back to data input”.

4.9 Interface with MissionPlanner (freeware)

This section is dedicated to explaining the interface process behind the import and export of routes to and from the freeware MissionPlanner by Michael Oborne.

4.9.1 Import route

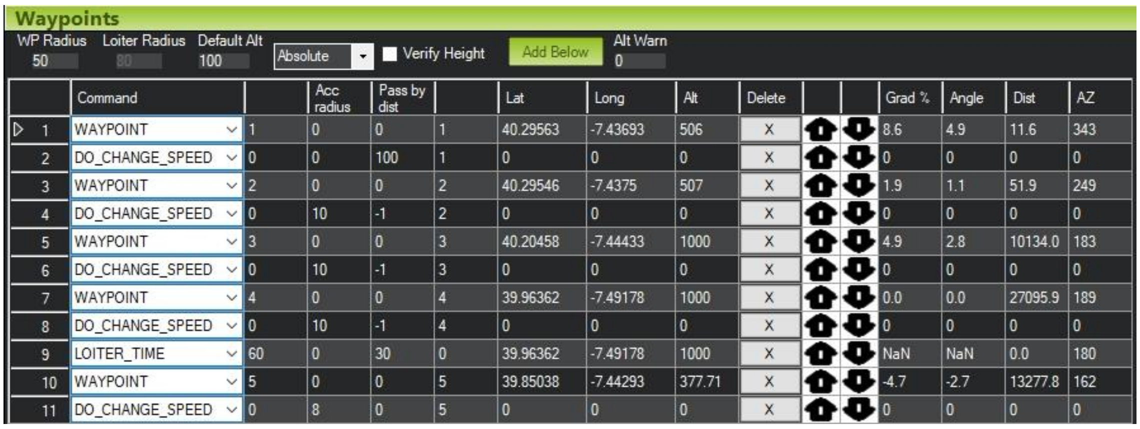
Importing a route from MissionPlanner facilitates the insertion of data into the waypoints’ table inside the GUI’s Mission tab, Route and Waypoints section. It also provides intuitive visual aid to the creation of the waypoints themselves, as the route can be visualized on a map.

In MissionPlanner, a route is defined by a set of commands. These are executed by the UAV (in autopilot mode) by the order they appear in the list. Due to this, specific commands should be set in a certain priority in relation to one another, so that the UAV can successfully fly the route. Additionally, some commands like *TAKEOFF* or *LAND* are necessary for the UAV’s autopilot (ArduPilot).

For the creation of a route to be exported to the Mission Optimization Interface, some of these rules may not be necessary. However, each waypoint (from start to finish) should have the following command sequence:

1. *WAYPOINT* (coordinates);
2. *DO_CHANGE_SPEED* (airspeed);
3. *LOITER_TIME* (loiter, optional).

If a waypoint does not have loiter, the *LOITER_TIME* command can be ignored. Like a normal route creation, a set of *HOME* coordinates must also be specified to be exported to the GUI, and they can be the same as the first waypoint coordinates. Also, the use of absolute (above MSL) altitudes is recommended, as the first two and last waypoints of the final optimized route use altitudes directly from the input route (see section 4.9.2). An example of a route ready for export to the GUI is shown in Figure 73.



Waypoints

WP Radius 50 | Loiter Radius 80 | Default Alt 100 | Absolute | Verify Height | Add Below | Alt Warn 0

	Command		Acc radius	Pass by dist		Lat	Long	Alt	Delete			Grad %	Angle	Dist	AZ
▷ 1	WAYPOINT	1	0	0	1	40.29563	-7.43693	506	X	⇧	⇩	8.6	4.9	11.6	343
2	DO_CHANGE_SPEED	0	0	100	1	0	0	0	X	⇧	⇩	0	0	0	0
3	WAYPOINT	2	0	0	2	40.29546	-7.4375	507	X	⇧	⇩	1.9	1.1	51.9	249
4	DO_CHANGE_SPEED	0	10	-1	2	0	0	0	X	⇧	⇩	0	0	0	0
5	WAYPOINT	3	0	0	3	40.20458	-7.44433	1000	X	⇧	⇩	4.9	2.8	10134.0	183
6	DO_CHANGE_SPEED	0	10	-1	3	0	0	0	X	⇧	⇩	0	0	0	0
7	WAYPOINT	4	0	0	4	39.96362	-7.49178	1000	X	⇧	⇩	0.0	0.0	27095.9	189
8	DO_CHANGE_SPEED	0	10	-1	4	0	0	0	X	⇧	⇩	0	0	0	0
9	LOITER_TIME	60	0	30	0	39.96362	-7.49178	1000	X	⇧	⇩	NaN	NaN	0.0	180
10	WAYPOINT	5	0	0	5	39.85038	-7.44293	377.71	X	⇧	⇩	-4.7	-2.7	13277.8	162
11	DO_CHANGE_SPEED	0	8	0	5	0	0	0	X	⇧	⇩	0	0	0	0

Figure 73 – Example of a 5-waypoint route in MissionPlanner ready to be exported to the GUI

The route can then be exported to a .waypoints file using the button "Save WP File" to a folder of the user's choice, and imported by the Mission Optimization Interface.

The way the import function of the GUI works, is that it amalgamates the three *WAYPOINT*, *DO_CHANGE_SPEED* and *LOITER_TIME* commands' data into single waypoints, and displays the converted route in the GEO waypoints' table in the Route and Waypoints section. The *HOME* coordinates can be copied to the ENU waypoints' table reference point if the user chooses to do so. Finally, an information message is displayed if the import process is successful.

4.9.2 Export route

This is the final step in the mission planning process that began in data editing and route import. In a similar way as route import, the export process to MissionPlanner allows the user to actually fly the optimized/analysed route in a real UAV, using the ArduPilot autopilot.

As was explained in the previous section, a number of commands, particularly *TAKEOFF*, *LAND* and *DO_SET_HOME* are essential to the correct functioning of the autopilot in the UAV. As such, those commands are added in the function that converts the route displayed in the table at the lower part of the "Finalization" tab. It is worth noting that the altitude to be used in MissionPlanner should always be Absolute, without the "Verify Height" checkbox checked.

The *DO_SET_HOME* command is added right at the top of the list that will make up the route, as displayed in the MissionPlanner. It automatically sets the *HOME* coordinates from the first waypoint displayed in the GUI. ArduPilot does not take into account any starting waypoint, as it only starts the commands list with *TAKEOFF*, maintaining the heading set by the user on the ground, until it reaches the height specified in the command. In essence, the *HOME* coordinates become waypoint #1.

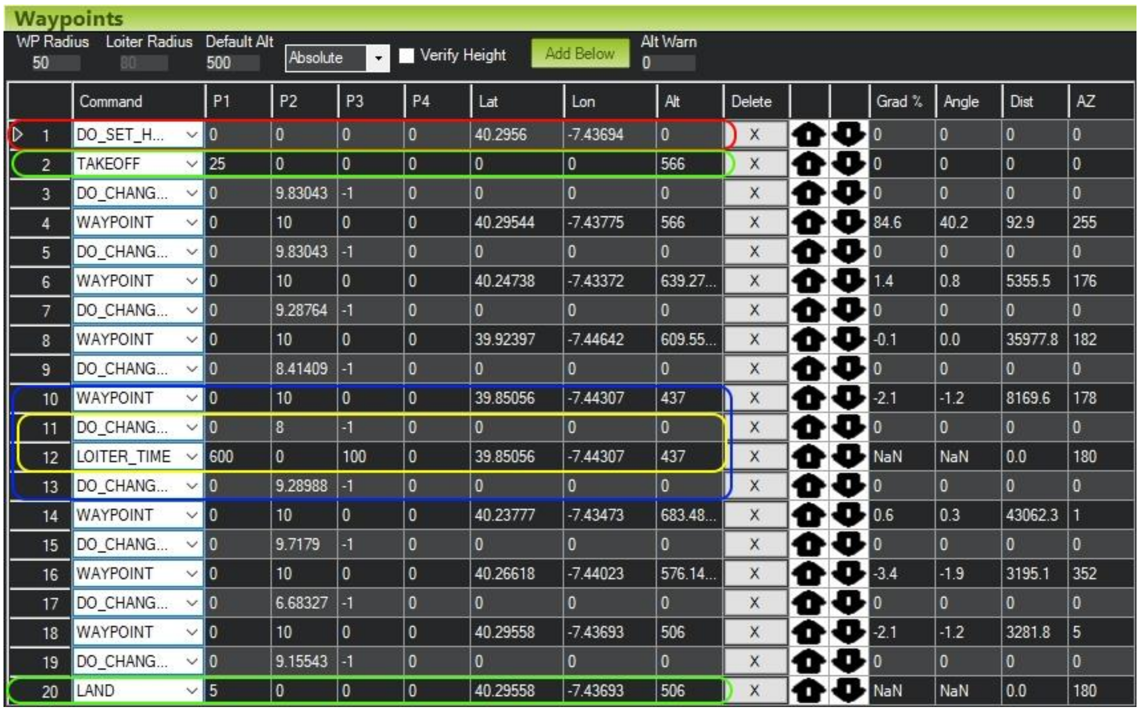
TAKEOFF is the second command added to the list. During its execution, the ArduPilot instructs the UAV to fly, maintaining its initial heading, until the altitude and with a "pitch angle" specified in the command line. By default, the altitude set is the first waypoint's altitude plus

60 metres, and the pitch angle is set to 25 degrees. These can later be changed by the user in MissionPlanner. Following this, the speed is changed to that of waypoint #1 using a *DO_CHANGE_SPEED* command. The 4th command sets the coordinates of waypoint #2, using the altitude set in *TAKEOFF*, and the 5th sets the airspeed of waypoint #2, using a *DO_CHANGE_SPEED* command.

The rest of commands are added in a similar fashion, always with the coordinates first and the change in speed in second.

In case a loiter was added to a waypoint, a *DO_CHANGE_SPEED* command that sets the loiter speed and a *LOITER_TIME* command that sets the time and radius, are added between the *WAYPOINT* and *DO_CHANGE_SPEED* commands of the affected waypoint. This is done so that after the UAV finishes loitering, it will change its speed to that originally set in the waypoint.

The *LAND* command is the last to be added to the list, with the same coordinates and altitude as the last waypoint. By default, its abort altitude is set to 5 metres.


Waypoints

WP Radius 50 | Loiter Radius 80 | Default Alt 500 | Absolute | Verify Height | Add Below | Alt Warn 0

	Command	P1	P2	P3	P4	Lat	Lon	Alt	Delete			Grad %	Angle	Dist	AZ
1	DO_SET_H...	0	0	0	0	40.2956	-7.43694	0	X			0	0	0	0
2	TAKEOFF	25	0	0	0	0	0	566	X			0	0	0	0
3	DO_CHANG...	0	9.83043	-1	0	0	0	0	X			0	0	0	0
4	WAYPOINT	0	10	0	0	40.29544	-7.43775	566	X			84.6	40.2	92.9	255
5	DO_CHANG...	0	9.83043	-1	0	0	0	0	X			0	0	0	0
6	WAYPOINT	0	10	0	0	40.24738	-7.43372	639.27...	X			1.4	0.8	5355.5	176
7	DO_CHANG...	0	9.28764	-1	0	0	0	0	X			0	0	0	0
8	WAYPOINT	0	10	0	0	39.92397	-7.44642	609.55...	X			-0.1	0.0	35977.8	182
9	DO_CHANG...	0	8.41409	-1	0	0	0	0	X			0	0	0	0
10	WAYPOINT	0	10	0	0	39.85056	-7.44307	437	X			-2.1	-1.2	8169.6	178
11	DO_CHANG...	0	8	-1	0	0	0	0	X			0	0	0	0
12	LOITER_TIME	600	0	100	0	39.85056	-7.44307	437	X			NaN	NaN	0.0	180
13	DO_CHANG...	0	9.28988	-1	0	0	0	0	X			0	0	0	0
14	WAYPOINT	0	10	0	0	40.23777	-7.43473	683.48...	X			0.6	0.3	43062.3	1
15	DO_CHANG...	0	9.7179	-1	0	0	0	0	X			0	0	0	0
16	WAYPOINT	0	10	0	0	40.26618	-7.44023	576.14...	X			-3.4	-1.9	3195.1	352
17	DO_CHANG...	0	6.68327	-1	0	0	0	0	X			0	0	0	0
18	WAYPOINT	0	10	0	0	40.29558	-7.43693	506	X			-2.1	-1.2	3281.8	5
19	DO_CHANG...	0	9.15543	-1	0	0	0	0	X			0	0	0	0
20	LAND	5	0	0	0	40.29558	-7.43693	506	X			NaN	NaN	0.0	180

Figure 74 – Example of a converted route in MissionPlanner. Highlighted are the *DO_SET_HOME* command (red), the *TAKEOFF* and *LAND* commands (green), and the example of a waypoint (blue) with loiter commands (yellow)

On a final note, the route exported to MissionPlanner should not directly be used by the UAV. All altitudes should first be checked (as warned by a message in the GUI), and some waypoints may have to be moved for operational reasons. Ultimately, the user has the authority and responsibility for the operational functionality of the route to be flown automatically by the autopilot, or to act as an aid to manual flying by a human pilot. Flexibility is one of the many advantages that MissionPlanner presents to the operation of UAVs.

Chapter 5

GUI Application and Mission Optimization Tests

5.1 Introduction

In order to prove the efficiency and reliability of both the Mission Planner Program and the latest added features to the software, including the developed interface of this thesis, several tests under different conditions, parameters and objective functions were performed, and are presented in this chapter.

Although a large number of tests were performed to the GUI throughout its development, they are not described in the present work as they are considered to be part of the development phase itself. Rather, only testing of the final product sequence – MissionPlanner-GUI-Optimization-GUI-MissionPlanner – is described and discussed in the following sections.

5.1.1 Choice of route

For the matter of simplicity and availability of time, a single route was chosen for the testing procedures, where a large number of conditions can be tested and its results compared much more easily. Using more than one route is unnecessary to prove the functionality of the Mission Planner.

The departing point of the route was chosen at Terlamonte airfield (N40.2956° W7.4369°, elevation 506 m), near the city of Covilhã, Portugal. The LEEUAV then flies to a point 60 metres above Castelo Branco Municipal Airport (N39.8506° W7.4431°, elevation 377 m), where it loiters for 10 minutes, and proceeds to return to Terlamonte, where it finally lands. There was a desire in reducing the number of waypoints as much as possible, in order to reduce the running time of the optimization process of the MPP. A minimum of 8 waypoints were used for the definition of the input route, a number which is high enough to not compromise the optimization process in any way, guaranteeing 2 free waypoints in the Terlamonte-Castelo Branco route, and another 2 in the back course.

In addition, the route was designed to not have a segment climb or descent inclination of more than 8 degrees, to make sure that the LEEUAV can climb or descend within the maximum power setting and minimum power constraints. Moreover, after an initial testing sequence, the route was feasible in order to be flyable by the LEEUAV. This was necessary, as the tests were made in October-November and the weather conditions were not the most favourable.

Tables 3 and 4 show how the input route and some of its design parameters should look like in the Mission Optimization Interface' Mission tab, Route and Waypoints section.

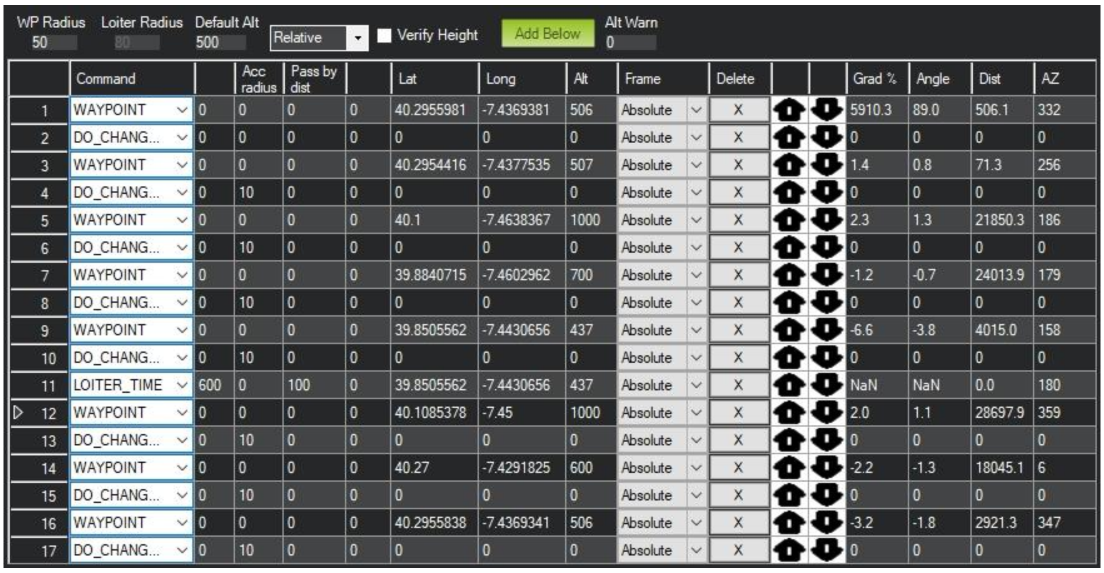
Table 3 – Waypoints table of the input route

Waypoints table						
Waypoint Number	Ground/ Air	Loiter	Latitude [deg]	Longitude [deg]	Altitude [m]	Airspeed [m/s]
1	G	N	40.2955981	-7.4369381	506	0
2	G	N	40.2954416	-7.4377535	507	10
3	A	N	40.1000000	-7.4638367	1000	10
4	A	N	39.8840715	-7.4602962	700	10
5	A	Y	39.8505562	-7.4430656	437	10
6	A	N	40.1085378	-7.4500000	1000	10
7	A	N	40.2700000	-7.4291825	600	10
8	G	N	40.2955838	-7.4369341	506	10

Table 4 – Loiters table of the input route

Loiters table				
Loiter Number	Waypoint Number	Time [s]	Airspeed [m/s]	Radius [m]
1	5	600	10	100

Figures 75 and 77 display the input route as was edited in the freeware MissionPlanner, before exporting it to the Mission Optimization Interface.


WP Radius 50 | Loiter Radius 80 | Default Alt 500 | Relative | Verify Height | Add Below | Alt Warn 0

	Command		Acc radius	Pass by dist		Lat	Long	Alt	Frame	Delete			Grad %	Angle	Dist	AZ
1	WAYPOINT	0	0	0	0	40.2955981	-7.4369381	506	Absolute	X	⬆	⬇	5910.3	89.0	506.1	332
2	DO_CHANG...	0	0	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
3	WAYPOINT	0	0	0	0	40.2954416	-7.4377535	507	Absolute	X	⬆	⬇	1.4	0.8	71.3	256
4	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
5	WAYPOINT	0	0	0	0	40.1	-7.4638367	1000	Absolute	X	⬆	⬇	2.3	1.3	21850.3	186
6	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
7	WAYPOINT	0	0	0	0	39.8840715	-7.4602962	700	Absolute	X	⬆	⬇	-1.2	-0.7	24013.9	179
8	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
9	WAYPOINT	0	0	0	0	39.8505562	-7.4430656	437	Absolute	X	⬆	⬇	-6.6	-3.8	4015.0	158
10	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
11	LOITER_TIME	600	0	100	0	39.8505562	-7.4430656	437	Absolute	X	⬆	⬇	NaN	NaN	0.0	180
▷ 12	WAYPOINT	0	0	0	0	40.1085378	-7.45	1000	Absolute	X	⬆	⬇	2.0	1.1	28697.9	359
13	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
14	WAYPOINT	0	0	0	0	40.27	-7.4291825	600	Absolute	X	⬆	⬇	-2.2	-1.3	18045.1	6
15	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0
16	WAYPOINT	0	0	0	0	40.2955838	-7.4369341	506	Absolute	X	⬆	⬇	-3.2	-1.8	2921.3	347
17	DO_CHANG...	0	10	0	0	0	0	0	Absolute	X	⬆	⬇	0	0	0	0

Figure 75 – List of commands of the input route in the MissionPlanner

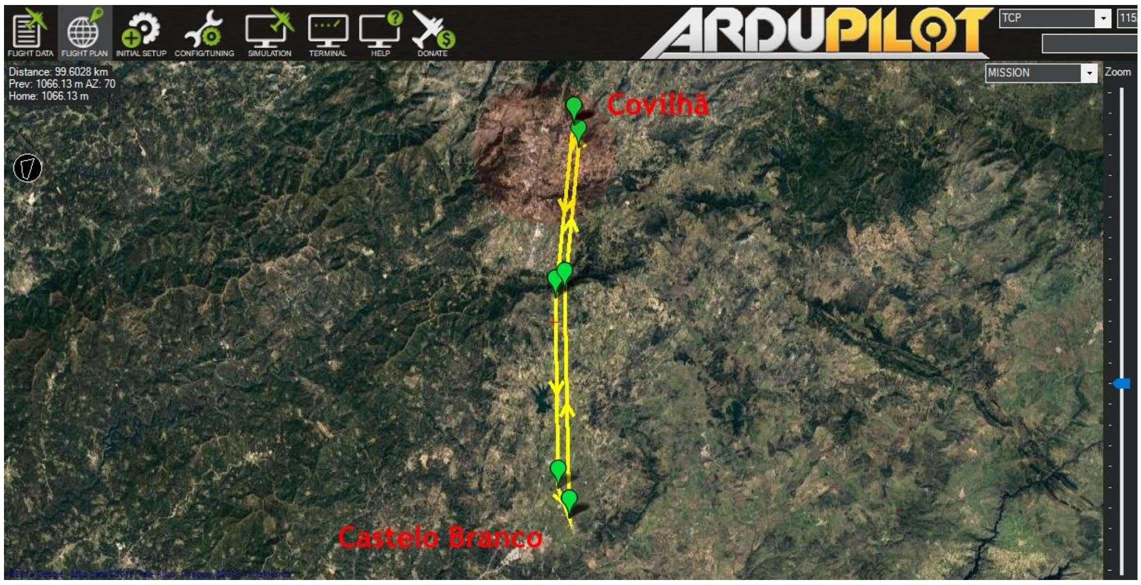
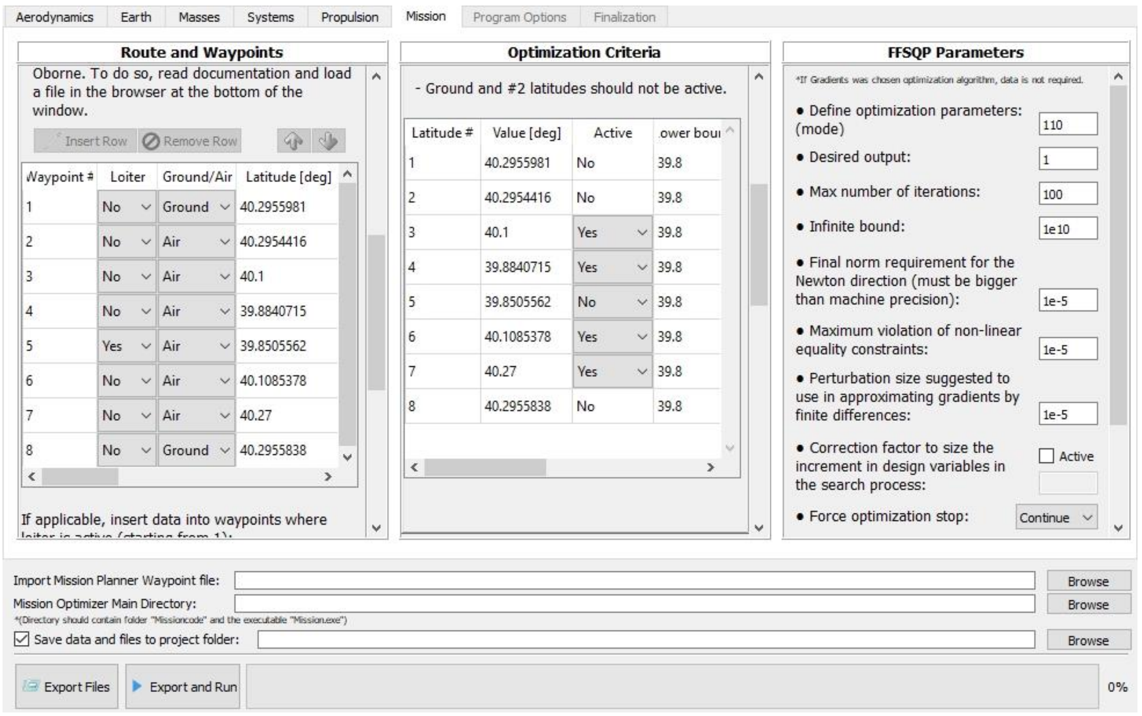


Figure 76 – Top-down map view of the route in MissionPlanner

After exporting the MP route to the GUI, waypoint #5's latitude, longitude and altitude design parameters were deactivated for optimization in the Design Parameters' tables, ensuring that the loiter in waypoint #5 is fixed. The automatic copying process described in section 4.7.2 ensures that waypoints #1, #2 and #8 are also fixed, as shown in Figure 76.



Route and Waypoints

Oborne. To do so, read documentation and load a file in the browser at the bottom of the window.

Waypoint #	Loiter	Ground/Air	Latitude [deg]
1	No	Ground	40.2955981
2	No	Air	40.2954416
3	No	Air	40.1
4	No	Air	39.8840715
5	Yes	Air	39.8505562
6	No	Air	40.1085378
7	No	Air	40.27
8	No	Ground	40.2955838

If applicable, insert data into waypoints where

Optimization Criteria

- Ground and #2 latitudes should not be active.

Latitude #	Value [deg]	Active	ower bou
1	40.2955981	No	39.8
2	40.2954416	No	39.8
3	40.1	Yes	39.8
4	39.8840715	Yes	39.8
5	39.8505562	No	39.8
6	40.1085378	Yes	39.8
7	40.27	Yes	39.8
8	40.2955838	No	39.8

FFSQP Parameters

*If Gradients was chosen optimization algorithm, data is not required.

• Define optimization parameters: (mode)	110
• Desired output:	1
• Max number of iterations:	100
• Infinite bound:	1e10
• Final norm requirement for the Newton direction (must be bigger than machine precision):	1e-5
• Maximum violation of non-linear equality constraints:	1e-5
• Perturbation size suggested to use in approximating gradients by finite differences:	1e-5
• Correction factor to size the increment in design variables in the search process:	☐ Active
• Force optimization stop:	Continue

Import Mission Planner Waypoint file:

Mission Optimizer Main Directory:

*(Directory should contain folder "Missioncode" and the executable "Mission.exe")

☑ Save data and files to project folder:

Figure 77 – View of input route data in the GUI's Mission tab

5.1.2 Elevation and Weather conditions

The tests were run using downloaded elevation and real-time weather data for 5 days, optimizing according to two objective functions – "Minimize Energy" and "Minimize Time" – and two types of LEEUAV propulsion data, detailed in section 5.1.3. Also, on each choice, 5 tests were run using a fixed mission initial hour, plus another with an active (variable) mission initial hour design parameter. At the end, the best fixed-hour solution for each day was further optimized with a variable initial hour parameter. This adds to a total of 140 tests performed.

The elevation map's latitude and longitude coordinates are comprised within the intervals $[39.30°;\ 40.80°]$ and $[-8.00°;\ -7.00°]$, respectively. The software Tecplot enables an in-depth visualization of the elevation map, which is shown in Figures 78 and 79, with the initial route.

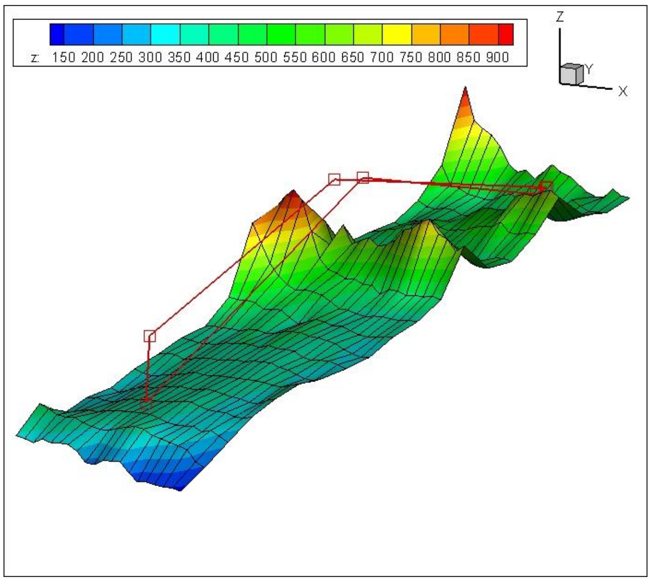

z: 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900
Z
Y
X

Figure 78 – 3D view of the terrain map and input route

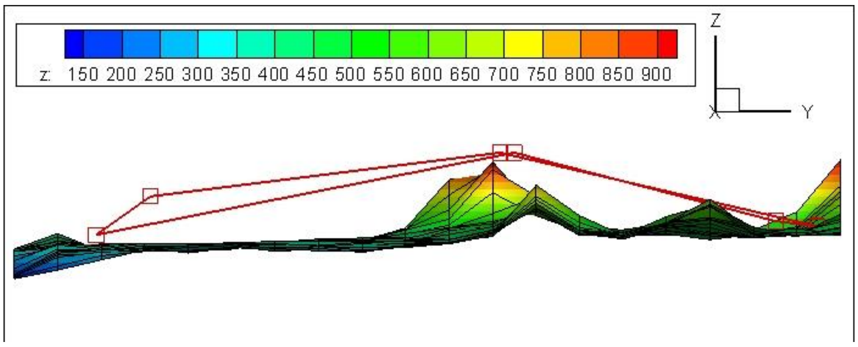

z: 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 900
Z
X
Y

Figure 79 – Side view of the terrain map and input route. Departing point is on the right side

The weather map's latitude and longitude coordinates are comprised within the intervals $[39.30°;\ 40.80°]$ and $[-8.00°;\ -7.00°]$, respectively. The different weather conditions obtained from 5 days (24-27 October and 16 November) are set in the earlier hours, from 3h to 15h. Figures 80 and 81 show the representation of the wind and cloud percentage maps, for 9h and 12h, respectively.

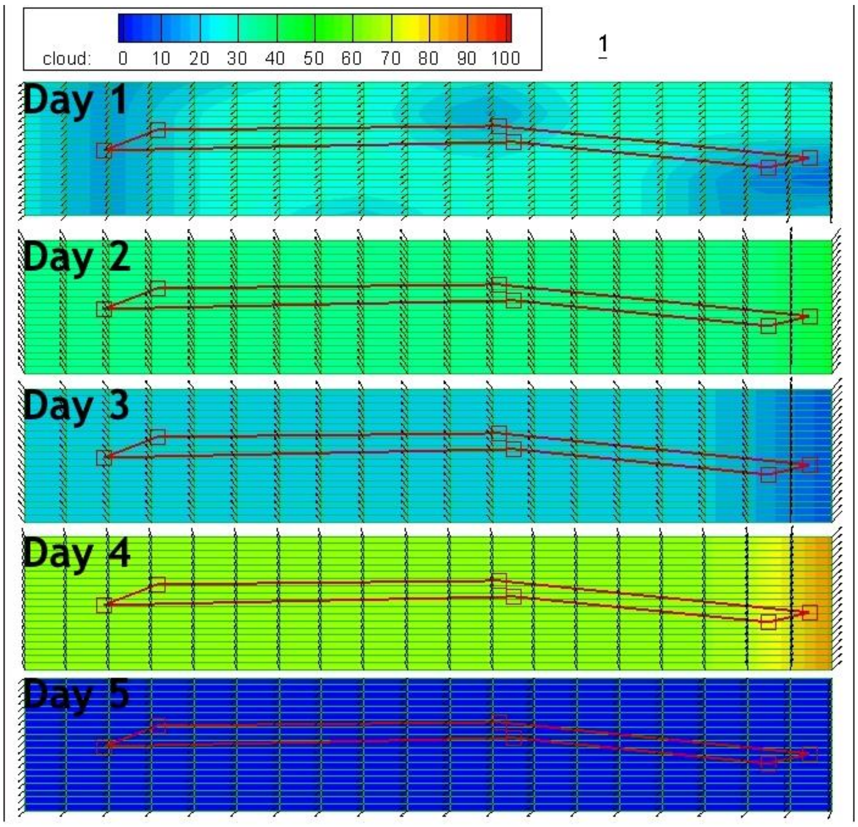


Figure 80 - Tests' daily cloud and wind analysis at 9 o'clock. North is to the right side

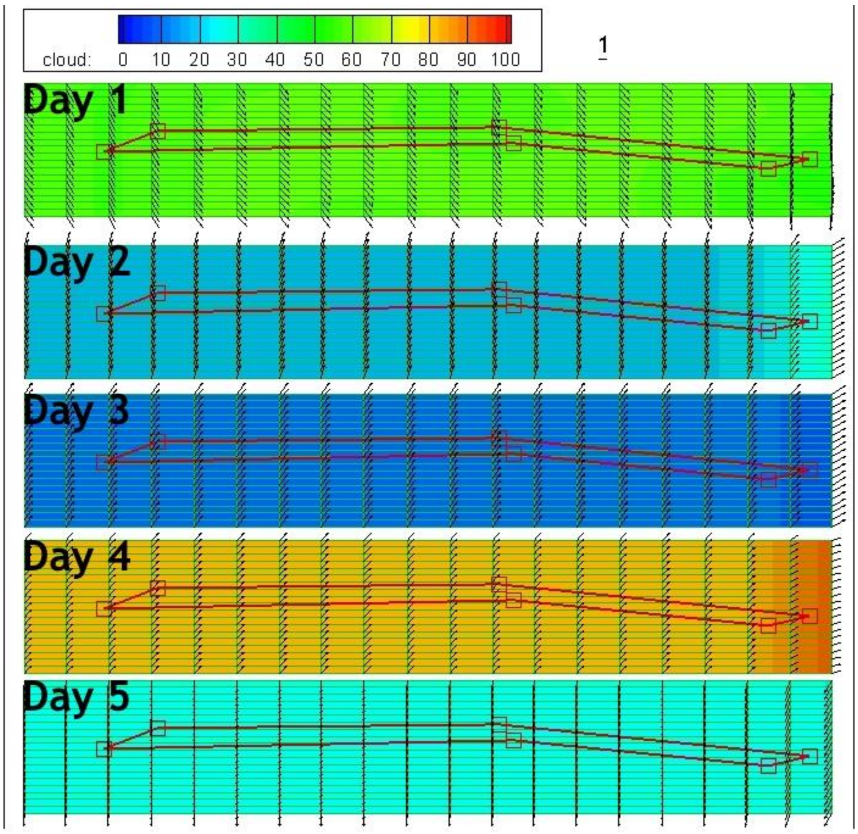


Figure 81 - Tests' daily cloud and wind analysis at 12 o'clock. North is to the right side

An analysis of the previous maps shows that cloud coverage conditions worsen from 9h to 12h in days 1, 4 and 5, while wind changes are mixed in all 5 days. Day 4 appears to have the worst weather to fly a mission, showing the highest cloud coverage, and a change in wind direction from due west to due north-west. In opposite terms, days 3 and 5 appear to have the best cloud coverage to fly, ranging from complete clear sky (0%) to about 25% partially overcast. However, wind changes may result in higher or lesser energy usage, depending on the starting hour of the mission. Stronger winds present in days 2, 3 and 4 may outweigh any advantage from a clearer sky. It is also worth noting that the temperatures for day 5 (16 November) are considerably lower than that of days 1-4 (24-27 October), by a margin of approximately 6 to 8 degrees Celsius.

Despite in-depth analysis of the weather, only a thorough calculation of energy usage and flight time of the route by the MPP can provide the clearest picture of what are the best conditions and times to fly the LEEUAV.

5.1.3 LEEUAV data

Two types of propulsion ("engine") data for the same LEEUAV model were used in the optimization tests, for a greater diversity of result values. This model uses only one motor and a 2-blade propeller. In summary, the differences between the two types are presented in tables 5 and 6, while data related to aerodynamics, mass, systems and battery is the same for both engine types 1 and 2. Unspecified parameters are defined at the Nomenclature section.

Table 5 – LEEUAV Motor Data

LEEUAV Motor Data		
	Motor 1 - Scorpion SII-4025-520kV	**Motor 2 - Hyperion ZS3025-10**
δ_{limit}	1	1
$m_{eng}\ [kg]$	0.353	0.198
r_{gear}	1	1
η_{gear}	1	1
$K_v\ [rpm/V]$	520	775
$R_U\ [\Omega]$	0.009	0.019
$I_0\ [A]$	1.4	1.61
$U_0\ [V]$	10	10
$I_{max}\ [A]$	100	65
$U_{max}\ [V]$	25.2	16.8
$R_{ESC}\ [\Omega]$	0.005	0.006

Table 6 – LEEUAV Propeller Data

LEEUAV Propeller Data		
	Propeller 1 - General 2-blade	**Propeller 2 - General 2-blade**
$d\ [in]$	19.09	16
$p\ [in]$	15.43	8
n_{blades}	2	2
$m_{prop}\ [kg]$	0.072	0.072

Table 7 – LEEUAV Aerodynamics-Geometry Data

Aerodynamics - Geometry			
$S\ [m^2]$	$S_F\ [m^2]$	$S_{Wet}\ [m^2]$	**Type of propeller**
1.485	0.0144	0.2	Tractor

Table 8 - LEEUAV Aerodynamics-Drag Polar Data

Aerodynamics - Drag Polar Coefficients				
C_{D_0}	C_{D_1}	C_{D_2}	C_{D_3}	C_{D_4}
0.057597933	-0.13382301	0.24208129	-0.15192704	0.041836736

Table 9 – LEEUAV Aerodynamics-Aircraft Representation data

Aerodynamics - Aircraft Representation		
C_F	$C_{L_{max}}$	$C_{L_{takeoff}}$
0.001	1.5	0.8

Table 10 – LEEUAV Masses Data

Masses $[kg]$			
Structural	**Fuel**	**Solar System**	**Propulsion System Fixed**
2.624	0	0.801	0.1
Propulsion System	**Battery Pack**	**Other Systems**	**Payload**
0.425 (1) or 0.27 (2)	0.75	0.5	0.2

Table 11 – LEEUAV Systems Data

Systems		
System	**Mass $[kg]$**	**Power $[W]$**
Avionics	0.5	11.750

Table 12 – LEEUAV Battery Data

Battery - SPS_APL_3S1P_10000mAh				
U_{cell} $[V]$	$U_{cell_{min}}$ $[V]$	$U_{cell_{max}}$ $[V]$	C_{cell} $[Ah]$	$I_{cell_{max}}$ $[A]$
3.95	3.3	4.2	10	150
R_{cell} $[\Omega]$	m_{cell} $[kg]$	$n_{cells,series}$	$n_{cells,parallel}$	u_{batt} $[Wh/kg]$
0.001	0.25	3	1	158.0

5.1.4 Mission constraints

Some of the constraint functions described in section 3.1.5 were used during the optimization tests to actively limit the calculation of variables in those functions to only feasible ones. No equality constraint functions were ever used, and the inequality constraints used are listed in Table 13.

Table 13 – Inequality constraint functions used during the optimization tests

Inequality Constraint Functions		
Description	**Scale factor**	**Value**
Engine setting ≤ Engine setting max ($\delta \leq \delta_{limit}$) at design conditions	1	1.00
Motor current ≤ Motor current max ($I \leq I_{max}$) at design conditions	1	100 A
$\left(\frac{V_{min}}{V_s}\right)^2 C_L \leq C_{L_{max}}$ at design conditions	1	1.5
Segment height min ≥ [sel. value] ($h_j \geq h_{ref}$)	1	50 m
Segment power min ≥ [sel. value] ($P_j \geq P_{ref}$)	1	0.1 W
Mission energy left min ≥ [sel. value] ($E_{left} \geq E_{ref}$)	1	20.0 %

In order to better obtain similar results from any subsequent repetition of the tests made in section 5.2, it is also necessary to mention that the values used for the upper and lower bounds of the latitude, longitude, altitude and airspeed design parameters are listed in Table 14.

Table 14 – Lower and Upper bounds used during the optimization tests for the design parameters

Design Parameters' Lower and Upper bounds			
Description	**Lower Bound**	**Upper Bound**	**Units**
Latitude (φ)	39.8	40.31	deg
Longitude (λ)	-7.5	-7.39	deg
Altitude (h)	0	2500	m
Airspeed (V)	0	100	m/s

An increment value of 0.00001 and a scale factor of 1 was used for all design parameters.

When active, values for the lower and upper bounds of the initial hour design parameter were adjusted, according to the reference initial hour used, in order to maintain a $[8h, 12h]$ time interval. The exceptions are in the 8h and 12h reference initial hours, where the time intervals of $[7h, 12h]$ and $[8h, 13h]$ were used, respectively.

5.2 Test Results

In order to facilitate organization, a project folder was created for the GUI to export the files to. This folder contains several subfolders with test data, the names of which follow the code division presented in Table 15.

Table 15 – Codes to organize the tests

Test codes	
E1	Engine-propeller data 1
E2	Engine-propeller data 2
ME	Minimize Energy objective function
MT	Minimize Time objective function
D(i)	Day (number) of weather data
F(i)H	Fixed i-hour optimization
VarH	Variable hour optimization
VarH2	Variable hour 2nd optimization

5.2.1 ME Tests

The first batch of tests executed were the "Minimize Energy" (ME) tests, for both E1 and E2 engine-propeller types. Firstly, five fixed mission initial hour optimizations (4h, 6h, 8h, 10h and 12h) were performed, as well as optimizations with variable mission initial hour parameters within the $[8h, 12h]$ time interval. Following this, a series of 2nd optimizations, with variable mission initial hour parameters of the underlined best solutions obtained from the fixed initial hours, was performed. The results obtained are presented in Tables 16 and 17.

Table 16 – Engine-propeller type 1, Minimize Energy (E1-ME) test results

Test E1 – ME					
Time Variable	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Initial hour: 8h	653492 J	690911 J	649817 J	665903 J	674464 J
Initial hour: 10h	649320 J	657011 J	649416 J	660647 J	679598 J
Initial hour: 12h	669576 J	673894 J	680306 J	NS	746196 J
Variable Initial hour (2nd opt.)*	648735 J (9h:58m)	656140 J (9h:58m)	640907 J (8h:51m)	659853 J (9h:58m)	670074 J (8h:49m)
Variable Initial hour	644454 J (9h:13m)	655562 J (9h:12m)	645275 J (9h:3m)	652641 J (8h:47m)	669888 J (9h:28m)

Data presented as: *energy* (*initial hour*); *Optimization of previous best result; NS=No Solution

Table 17 – Engine-propeller type 2, Minimize Energy (E2-ME) test results

Test E2 – ME					
Time Variable	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Initial hour: 8h	744497 J	757012 J	756580 J	751675 J	769485 J
Initial hour: 10h	739432 J	749986 J	756870 J	754011 J	775785 J
Initial hour: 12h	777085 J	770858 J	785644 J	NS	858212 J
Variable Initial hour (2nd opt.)*	736077 J (9h:52m)	749998 J (9h:59m)	742366 J (8h:25m)	749780 J (8h:14m)	764210 J (8h:57m)
Variable Initial hour	739546 J (9h:28m)	752454 J (9h:31m)	733002 J (8h:56m)	746759 J (8h:47m)	764059 J (9h:12m)

Data presented as: *energy* (*initial hour*); *Optimization of previous best result; NS=No Solution

All Mission Planner Program runs in the fixed 4h and 6h initial hours are not displayed in these tables, because no solution was obtained from the optimization process. This is due to the fact that the UAV runs out of energy before reaching Castelo Branco, as there is no solar light in that time of the day to balance the energy consumption (during the last days of October and early days of November, the sun rises at just past 7 o'clock). The testing experience also showed that fixed 8h initial hour tests are more prone to optimizations with no solutions, because of very low angles of incidence between the sunlight and the PV panels surface.

In short, engine-propeller data 1 is more efficient for the flight conditions than engine-propeller data 2. This may be explained by the fact that motor 1 can run on a higher power setting ($U_{max} \cdot I_{max}$), and propeller 1 has a higher diameter and pitch values than propeller 2. Due to the lower efficiency of the propulsion system of data 2, more altitude and airspeed modifications to the initial route had to be made to turn the initial route flyable, while optimizations using data 1 used less or none at all. In general, there is not much difference between both data types, regarding the best time to launch the UAV, which is mostly set between 9h and 10h.

The lack of any solutions at 12h of day 4 confirms the expectation made in section 5.1.2 that this is the worst time to fly the aircraft. On another note, energy values from day 5 are higher than any other day, despite good cloud coverage, whereas the lowest energy values are found in days 1 and 3. Explanation for this can be found in the changing wind directions in days 1-4 (Figures 80 and 81), which are always favourable if the aircraft takes off at 9h, and returns around 12h. Wind direction barely changes in day 5.

Despite the variable initial hour time window ranging from 8h to 12h, the results from *E2-ME-D1-VarH* and *E2-ME-D2-VarH* are not the lowest energy used values obtained from the optimization tests for each corresponding day. However, the remaining variable-hour optimizations did get the lowest energy usage for each day, proving the reliability of this added design parameter.

An analysis of the series of second optimizations shows that, with the exception of *E1-ME-D3-VarH2*, *E2-ME-D1-VarH2* and *E2-ME-D2-VarH2*, the values obtained from these optimizations are generally higher than the first variable initial hour tests. This can be understood by looking at the obtained initial hours, which are mostly near the design parameter references. This shows that the program gets "stuck" in the local minimum of the existing solution, and does not converge to the lowest minimum of the objective function, evidencing a limitation in the converging process of the optimization.

5.2.2 MT Tests

The second batch of tests executed were the "Minimize Time" (MT) tests, for both E1 and E2 engine-propeller types. Likewise, results from a set of fixed and variable mission initial hour tests, followed by a series of second optimizations of the best solutions from the fixed initial hour tests, are presented in Tables 18 and 19.

Table 18 - Engine-propeller type 1, Minimize Time (E1-MT) test results

Test E1 – MT					
Time Variable	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Initial hour: 8h	2h:16m:2s	2h:16m:5s	2h:15m:33s	2h:32m:8s	2h:26m:54s
Initial hour: 10h	2h:2m:23s	2h:0m:36s	1h:59m:36s	2h:26m:26s	2h:8m:7s
Initial hour: 12h	2h:2m:31s	2h:0m:27s	2h:1m:8s	NS	2h:13m:6s
Variable Initial hour (2nd opt.)*	2h:1m:49s (10h:30m)	1h:59m:8s (10h:51m)	1h:58m:33s (10h:39m)	2h:24m:0s (9h:15m)	2h:6m:55s (10h:29m)
Variable Initial hour	2h:1m:49s (10h:30m)	1h:58m:53s (10h:52m)	1h:58m:33s (10h:39m)	2h:24m:0s (9h:15m)	2h:6m:55s (10h:28m)

Data presented as: *energy* (*initial hour*); *Optimization of previous best result; NS=No Solution

Table 19 - Engine-propeller type 2, Minimize Time (E2-MT) test results

Test E2 – MT					
Time Variable	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**
Initial hour: 8h	2h:24m:58s	2h:24m:14s	2h:37m:8s	2h:48m:1s	2h:37m:54s
Initial hour: 10h	2h:10m:24s	2h:8m:5s	2h:6m:59s	2h:40m:58s	2h:16m:41s
Initial hour: 12h	2h:13m:3s	2h:7m:56s	2h:9m:19s	NS	2h:22m:23s
Variable Initial hour (2nd opt.)*	2h:9m:27s (10h:27m)	2h:6m:18s (10h:46m)	2h:5m:59s (10h:34m)	2h:36m:29s (9h:10m)	2h:15m:32s (10h:23m)
Variable Initial hour	2h:9m:33s (10h:26m)	2h:6m:39s (10h:47m)	2h:6m:16s (10h:33m)	2h:36m:29s (9h:10m)	2h:33m:50s (10h:32m)

Data presented as: *energy* (*initial hour*); *Optimization of previous best result; NS=No Solution

Like in the previous section, optimization tests for fixed initial hours 4h and 6h did not result in any solution, already explained by the lack of sunlight. The very low angles of incidence between the sunlight and the PV panels surface also affected the fixed 8h initial hour tests, which are more prone to optimizations with no solutions.

The availability of excess power allows the UAV to fly faster and higher, resulting in lower flight times. This is confirmed by comparing the results (in both data) of fixed 8h initial hour tests with the rest. The lower solar power available at that time of the day limits the UAV to fly slower and at lower altitudes, resulting in higher flight times. The more efficient propulsion system of engine-propeller data 1 also allows for optimization solutions with lower flight times, in comparison with engine-propeller data 2.

Little difference of flight time is shown between all days, however, days 2 and 3 show the best flight times, and days 4 and 5 show higher flight times, for both data. Results from day 4 are the worst, having the highest flight times and optimizations at 12h with no feasible solutions, again confirming the expectation of this being the worst time to fly.

The best times to launch, obtained from the variable initial hour tests, are generally different from those obtained in “Minimize Energy” by a margin of plus one hour. This might be explained by the necessity of the UAV to have more excess power at the start of the mission, in order to fly faster and higher. This excess power can only be obtained from solar power, which is greater close to noon. The exception is found in the very high cloud coverage close to noon in day 4, which limits the time to launch to an earlier hour, closer to 9h, where there is a greater availability of solar energy.

Unlike the “Minimize Energy” tests, the flight times resulting from the second optimizations are almost always equal or lower than each respective variable initial hour time obtained from the first optimizations. The convergence to better solutions in this optimization process may be due to a greater flexibility in the variation of the design parameters, whereas the design parameters obtained in “Minimize Energy” often result in solutions closer to the limit constraints, therefore not allowing for changes in these parameters, which would otherwise result in an unflyable route. Therefore, the local minima convergence limitation seen previously is more prevalent in the “Minimize Energy” tests than in the “Minimize Time” tests.

5.2.3 Running time comparisons

Complementing the analysis of objective functions’ results, an analysis of the running time of the optimization process for each test was also made. The graph in Figure 82 shows the running time for each “Minimize Energy” test of the MPP, according to the mission initial hour, engine-propeller data type and central processing unit used (an older or newer CPU).

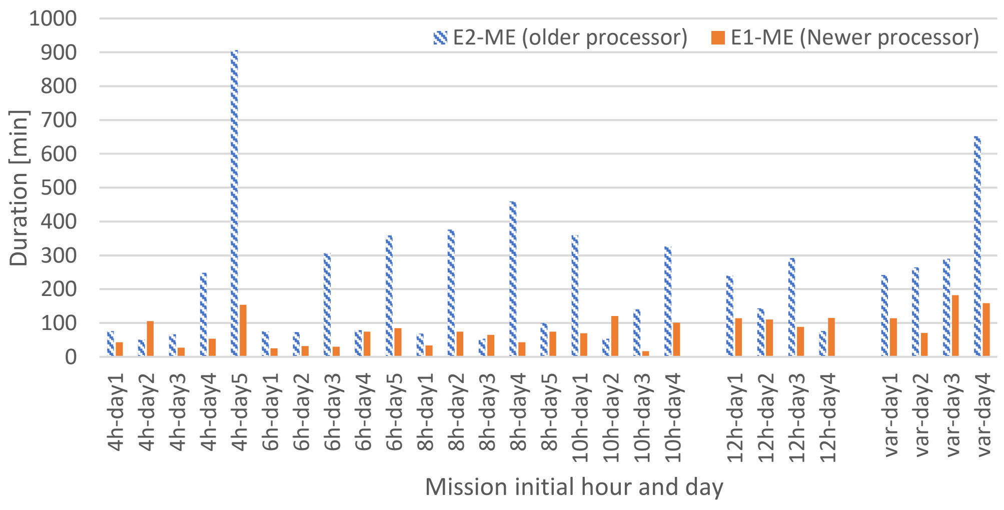


Figure 82 – Running time comparison between different CPUs for the ME optimization tests

The results clearly show, in general, the greater amount of time required by the older CPU (a 2007 Intel Core 2 Duo) to run the tests. During the testing process, the newer CPU (a 2017 Intel Core i7-8550U) allowed to run about 4 optimizations for each one using the older CPU.

A comparison of running times was also made, for engine-propeller data 1, between the types of objective function used ("Minimize Energy" or "Minimize Time"), which is shown in Figure 83.

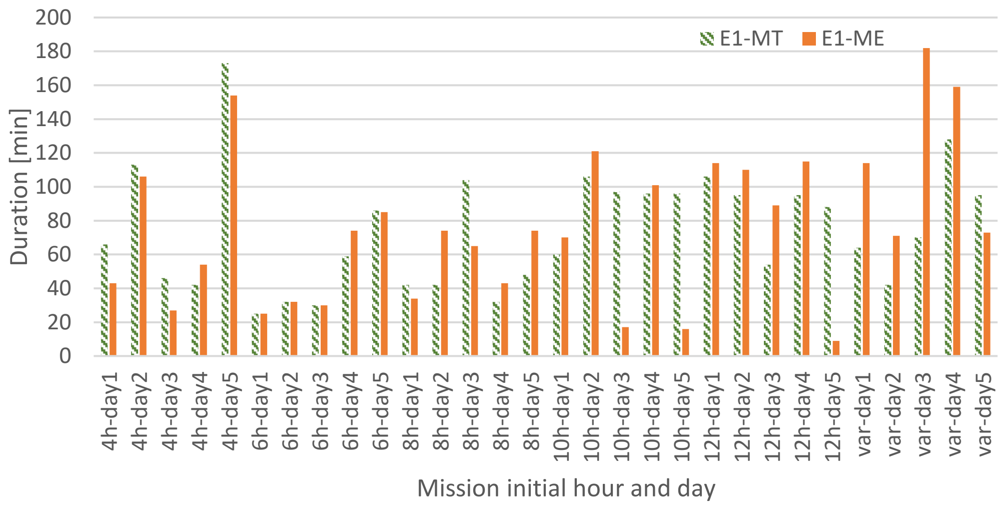


Figure 83 – Running time comparison between ME and MT optimization tests for engine-propeller data 1

The graph shows that almost no difference is present between the type of objective function used. However, for both graphs, the running time also depends on how close the initial route is to a solution - the closer it is, the less time the MPP will take to reach the solution. Also, if the initial route is barely or not flyable at all (considering all constraints), the MPP's running time will also be shorter, because no solution can ever be reached and/or optimized.

Chapter 6

Conclusions

6.1 Achievements

This thesis followed Coelho's work [11] on the creation of a mission planner program that optimizes a route based on the minimization of energy, time or distance, taking into account the initial route's design parameters, constraints, terrain and weather conditions, LEEUAV specifications and available solar power. A graphical user interface (GUI) was developed to complement the Mission Planner, in order to provide the user with an easy way of changing the various necessary data otherwise edited in a series of text files within the mission planner program's folders. This allowed for the application of an extensive testing procedure on the mission planner, which validated both the planning and interface parts, and corrected, improved and enlarged the capabilities of the mission planner software as a whole.

The development of the GUI allowed the importing and exporting of routes from and to a ground control station that is able to connect to a LEEUAV's autopilot, therefore providing a way of visualizing routes on a map before applying the planning process to a real-world flight. This addresses the issue underlined in section 2.2 of the State of the Art, of providing a flexible, easy and fast way for the user to link the mission planning process with the piloting phase of the LEEUAV.

The tests made at the end of this thesis identified the best capabilities, but also evidenced major limitations inherent to the use of the mission planner, which should be tackled in future developments. They relate to the optimization algorithm, weather, route, terrain and solar model. If necessary, these developments should, as done in the past, be accompanied with changes to the GUI.

6.2 Future Work

As a finishing note on this thesis, the development of the mission planner should be undertaken with a continuous improvement philosophy in mind. Therefore, based on the limitations found throughout this work, a series of developments are suggested, such as:

- Improve the optimization algorithm used: add stochastic processes that search the complete design space, and/or other gradient-based methods, allowing a broader analysis of local minima in the objective function, which ensures that the obtained solutions are closer to the global minimum and do not get “stuck” in a local minimum;

- Add more optimization algorithms to the Mission Planner;
- Improve the weather model and select a more complete database that: provides three-dimensional wind gusts and thermals; provides weather conditions according to altitude levels; takes account of precipitation, storms or other extreme events prejudicial to the operation of the LEEUAV;
- Improve the time system used in the mission planner, in order to take account of time zones, daylight-saving time (DST) changes or the more consistent local solar time;
- Add terrain or LEEUAV attitude-induced shadows to the solar model;
- Add no-fly zones and restrictions of heading to the optimization process, which is particularly useful for the approach sequence before landing on a runway;
- Improve the computational time of the mission planner;
- Add visual components to the GUI, such as the visualization of weather conditions and terrain profile with the initial and optimized routes;
- Add editable LEEUAV solar system data to the GUI;
- Perform real flight tests to prove the functionality of the Mission Planner Application Software.

It would also be interesting to use this mission planning software for a LEEUAV that is able to fly through the night, i.e. for endurances of 24 hours or more.

References

[1] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick, “An overview of emerging results in cooperative UAV control,” in *Proceedings of the IEEE Conference on Decision and Control*, 2004 [Online]. Available: https://www.researchgate.net/publication/4142631_An_overview_of_emerging_results_in_cooperative_UAV_control

[2] A. C. Marta and P. V. Gamboa, “Long endurance electric UAV for civilian surveillance missions,” *29th Congr. Int. Counc. Aeronaut. Sci. ICAS 2014*, 2014.

[3] M. Oborne, “Mission Planner. ArduPilot.,” 2010. [Online]. Available: https://github.com/ArduPilot/ardupilot_wiki/tree/master/planner

[4] L. F. V. Cândido, “Projeto de um UAV Solar de grande autonomia,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2014.

[5] J. C. C. Sousa, “Solar System for a Long Endurance Electric UAV,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2015.

[6] A. E. G. Duarte, “Development part of the Structure of a Long Endurance Electric UAV,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2016.

[7] L. M. A. Parada, “Conceptual and preliminary design of a Long Endurance Electric UAV,” MSc Thesis, Instituto Superior Técnico (IST), Lisboa, 2016.

[8] A. S. Rodrigues, “Airframe Assembly , Systems Integration and Flight Testing of a Long Endurance Electric UAV,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2017.

[9] L. Freitas, “Aerodynamic Analysis of a Forward-Backward Facing Step Pair on the Upper Surface of a Low-Speed Airfoil,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2017.

[10] P. A. L. Moutinho, “Real-Time Estimation of Remaining UAV Flight Time Based on the Total Energy Balance,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2017.

[11] L. M. M. Coelho, “Mission Planner for Solar Powered Unmanned Aerial Vehicles,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2019.

[12] S. Houston, “The important skills pilots acquire from flying,” 2019. [Online]. Available: https://www.thebalancecareers.com/important-skills-pilots-acquire-282950. [Accessed: 17-Oct-2019]

[13] R. J. Boucher, “History of Solar Flight,” in *Proc. of AIAA/SAE/ASME 20th Joint Propulsion Conference, AIAA Paper 84-1429*, 1984 [Online]. Available: http://astrobobb.com/solarhistory.pdf

[14] US government, “Timeline of the Achievements in Solar,” *Office of Energy Efficiency and Renewable Energy*. [Online]. Available: https://www.energy.gov/eere/solar/history. [Accessed: 25-Oct-2019]

[15] H. Bruss, “Solar Modellflug Grundlagen, Enwicklung, Praxis.” Verlag für Technik und handwerk, Baden-Baden, 1991.

[16] D. Stinton, *The Design of the Aeroplane*, 2nd ed. Oxford, UK: Blackwell Science, 2001.

[17] A. Noth, “Design of Solar Powered Airplanes for Continuous Flight,” Phd Thesis, Ecole Polytechnique Fédérale de Lausanne, Switzerland, 2008.

[18] J. Ewald, “The Real Flight With Solar Energy, Icaré 2,” 2007. [Online]. Available: https://web.archive.org/web/20070703230453/https://www.ifb.uni-stuttgart.de/icare/Englisch/flugberengl.html. [Accessed: 27-Oct-2019]

[19] B. Piccard, “My Ups and Downs with Solar Impulse,” 2016. [Online]. Available: https://www.linkedin.com/pulse/my-ups-downs-solar-impulse-bertrand-piccard. [Accessed: 27-Oct-2019]

[20] “Solar Impulse Foundation,” *Around the World*, 2019. [Online]. Available: https://aroundtheworld.solarimpulse.com/. [Accessed: 27-Oct-2019]

[21] “SolarStratos, To the Edge of Space,” 2019. [Online]. Available: https://www.solarstratos.com/en/. [Accessed: 29-Oct-2019]

[22] G. Goebel, “The Prehistory Of Endurance UAVs,” *Unmanned Aerial Vehicles*, 2009. [Online]. Available: https://web.archive.org/web/20090211170130/http://www.vectorsite.net/twuav_12.html. [Accessed: 27-Oct-2019]

[23] A. Noth, R. Siegwart, and W. Engel, “Autonomous Solar UAV for Sustainable Flight,” in *Advances in Unmanned Aerial Vehicles, State of the Art and the Road to Autonomy*, K. P. Valavanis, Ed. Springer Verlag, 2007.

[24] B. Keidel, “Auslegung und Simulation von hochfliegenden, dauerhaft stationierbaren Solardrohnen,” Phd Thesis, Technische Universität München, 2000.

[25] T. C. Tozer, D. Grace, J. Thompson, and P. Baynham, “UAVs and HAPs - Potential Convergence for Military Communications,” in *IEE Colloquium on “Military Satellite Communications,”* 2000.

[26] G. Romeo and G. Frulla, “HELIPLAT: high altitude very-long endurance solar powered UAV for telecommunication and Earth observation applications,” *Aeronaut. J.*, vol. 108, no. 1084, pp. 277–293, 2004.

[27] “Solar Powered, Unmanned QinetiQ Zephyr Plane Achieves Another Record,” *Airline World*, 2007. [Online]. Available: https://airlineworld.wordpress.com/2007/09/11/solar-powered-unmanned-qinetiq-zephyr-plane-achieves-another-record/. [Accessed: 27-Oct-2019]

[28] J. Amos, “Solar plane makes record flight,” *BBC News*, 2008. [Online]. Available: http://news.bbc.co.uk/2/hi/science/nature/7577493.stm. [Accessed: 27-Oct-2019]

[29] “Sky-Sailor, Autonomous Solar Airplane for Mars Exploration,” 2008. [Online]. Available: http://www.sky-sailor.ethz.ch/. [Accessed: 27-Oct-2019]

[30] “Solara 50 Atmospheric Satellite,” *Aerospace Technology*, 2013. [Online]. Available: https://www.aerospace-technology.com/projects/solara-50-atmospheric-satellite/. [Accessed: 29-Oct-2019]

[31] “NTSB Identification: DCA15CA117,” 2015 [Online]. Available: https://www.ntsb.gov/_layouts/ntsb.aviation/brief.aspx?ev_id=20150505X85410&key=1

[32] S. Weintraub, “Alphabet cuts former Titan drone program from X division, employees dispersing to other units,” *9to5Google*, 2017. [Online]. Available: https://9to5google.com/2017/01/11/alphabet-titan-cut/. [Accessed: 29-Oct-2019]

[33] G. Warwick, “Facebook’s UAV Flies, Builds On Developments In Solar Power,” *Aviation Week & Space Technology*, 2015. [Online]. Available: https://aviationweek.com/technology/facebook-s-uav-flies-builds-developments-solar-power. [Accessed: 29-Oct-2019]

[34] S. Trimble, “Facebook Unveils 42m-wingspan Aquila UAV,” *Flight Global*, 2015. [Online]. Available: https://www.flightglobal.com/news/articles/facebook-unveils-42m-wingspan-aquila-uav-415331/. [Accessed: 29-Oct-2019]

[35] “Facebook encerra projeto Aquila, o drone solar que ia levar internet às zonas remotas do planeta,” *SAPO Notícias*, 2018. [Online]. Available: https://tek.sapo.pt/noticias/internet/artigos/facebook-encerra-projeto-aquila-o-drone-solar-que-ia-levar-internet-as-zonas-remotas-do-planeta. [Accessed: 29-Oct-2019]

[36] D. Thisdell, “Airbus sets flight endurance record with Zephyr UAV,” *Flight Global*, 2018. [Online]. Available: https://www.flightglobal.com/news/articles/airbus-sets-flight-endurance-record-with-zephyr-uav-451006/. [Accessed: 29-Oct-2019]

[37] “Unmanned Aerial Systems (UAS),” *SKYbrary*, 2019. [Online]. Available: https://www.skybrary.aero/index.php/Unmanned_Aerial_Systems_(UAS). [Accessed: 26-Oct-2019]

[38] A. T. Klesh and P. T. Kabamba, “Energy-optimal path planning for Solar-powered aircraft in level flight,” *AIAA Guid. Navig. Control Conf. Exhib.*, vol. 3, no. August, pp. 2966–2982, 2007.

[39] R. Dai, U. Lee, S. Hosseini, and M. Mesbahi, “Optimal path planning for solar-powered UAVs based on unit quaternions,” in *Proceedings of the 51st IEEE Conference on Decision and Control*, 2012, pp. 3104–3109 [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6425972&tag=1

[40] Y. Huang, H. Wang, and P. Yao, “Energy-optimal path planning for Solar-powered UAV with tracking moving ground target,” *Aerosp. Sci. Technol.*, vol. 53, pp. 241–251, 2016 [Online]. Available: http://dx.doi.org/10.1016/j.ast.2016.03.024

[41] Y. Huang, J. Chen, H. Wang, and G. Su, “A method of 3D path planning for solar-powered UAV with fixed target and solar tracking,” *Aerosp. Sci. Technol.*, vol. 92, pp. 831–838, 2019 [Online]. Available: https://doi.org/10.1016/j.ast.2019.06.027

[42] J. J. Kiam and A. Schulte, “Multilateral quality mission planning for solar-powered long-endurance UAV,” *IEEE Aerosp. Conf. Proc.*, pp. 1–10, 2017 [Online]. Available: https://ieeexplore.ieee.org/document/7943802

[43] P. Oettershagen *et al.*, “Design of small hand-launched solar-powered UAVs: From concept study to a multi-day world endurance record flight,” *J. F. Robot.*, vol. 34, no. 7, pp. 1352–1377, 2017 [Online]. Available: https://www.doc.ic.ac.uk/~sleutene/publications/JFR_81hFlight_paper_final-1.pdf

[44] “AtlantikSolar, A UAV for the first-ever autonomous solar-powered crossing of the Atlantic Ocean,” 2017. [Online]. Available: https://www.atlantiksolar.ethz.ch/. [Accessed: 30-Oct-2019]

[45] P. Oettershagen, J. Förster, L. Wirth, J. Ambühl, and R. Siegwart, “Meteorology-Aware Multi-Goal Path Planning for Large-Scale Inspection Missions with Long-Endurance Solar-Powered Aircraft,” Zurich, 2017 [Online]. Available: https://arxiv.org/pdf/1711.10328.pdf

[46] L. Amorosi, L. Chiaraviglio, F. D’Andreagiovanni, and N. Blefari-Melazzi, “Energy-efficient mission planning of UAVs for 5G coverage in rural zones,” in *IEEE International Conference on Environmental Engineering, EE 2018 - Proceedings*, 2018, pp. 1–9 [Online]. Available: https://ieeexplore.ieee.org/document/8385250

[47] J. Wu, H. Wang, N. Li, P. Yao, Y. Huang, and H. Yang, “Path planning for solar-powered UAV in urban environment,” *Neurocomputing*, vol. 275, pp. 2055–2065, 2018 [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0925231217316880?via%3Dihub

[48] B. Schellenberg, T. Richardson, A. Richards, R. Clarke, and M. Watson, “On-board real-time trajectory planning for fixed wing unmanned aerial vehicles in extreme environments,” *Sensors*, vol. 19, no. 19, 2019 [Online]. Available: https://www.researchgate.net/publication/335981017_On-Board_Real-Time_Trajectory_Planning_for_Fixed_Wing_Unmanned_Aerial_Vehicles_in_Extreme_Environments

[49] “QBase 3D, Intuitive Mission Planning,” *Quantum Systems*, 2019. [Online]. Available: https://www.quantum-systems.com/qbase3d/. [Accessed: 04-Nov-2019]

[50] “QGroundControl, Intuitive and Powerful Ground Control Station for the MAVLink protocol,” *Drone Control*, 2019. [Online]. Available: http://qgroundcontrol.com/. [Accessed: 04-Nov-2019]

[51] P. F. Godinho Lopes Fernandes de Albuquerque, “Mission-Based Multidisciplinary Design Optimization Methodologies for Unmanned Aerial Vehicles with Morphing Technologies,” PhD Thesis, Universidade da Beira Interior (UBI), Portugal, 2017.

[52] J. L. Zhou, A. L. Tits, and C. T. Lawrence, “User’s guide for FFSQP version 3.7: A FORTRAN code for Solving Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality and Linear Constraints,” Electrical Engineering Department and Institute for Systems Research, University of Maryland, College Park, 1998 [Online]. Available: https://www.researchgate.net/publication/2693409_User’s_Guide_for_FFSQP_Version_37_A_FORTRAN_Code_for_Solving_Constrained_Nonlinear_Minimax_Optimization_Problems_Generating_Iterates_Satisfying_All_Inequality_and_Linear_Constraints

[53] N. S. Ribau, “Automatic engine and propeller selection for mission and performance optimization,” MSc Thesis, Universidade da Beira Interior (UBI), Portugal, 2018.

[54] “Unix Time Stamp,” *Dan’s Tools*, 2014. [Online]. Available: https://www.unixtimestamp.com/. [Accessed: 12-Nov-2019]

[55] “Qt for Python,” *The Qt Company*, 2019. [Online]. Available: https://www.qt.io/qt-for-python. [Accessed: 05-May-2019]

[56] “PyQt5 Support for Signals and Slots,” *Riverbank Computing Ltd.*, 2019. [Online]. Available: https://www.riverbankcomputing.com/static/Docs/PyQt5/signals_slots.html. [Accessed: 13-Nov-2019]

[57] M. Granger, “Elevation API,” 2019. [Online]. Available: https://elevation-api.io/. [Accessed: 14-Nov-2019]

[58] “Google Maps Platform,” *Geo-Locations APIs*, 2019. [Online]. Available: maps.googleapis.com. [Accessed: 14-Nov-2019]

Annex A – List of parameters obtained from Mission Planner Program's mission analysis

Mission Analysis Obtained Parameters

Parameters	SI Units
Mean Altitude (h_{ave})	m
Ground distance (dg)	m
Altitude variation (dh)	m
Flight path distance (ds)	m
Angle of trajectory relative to the ground (xy-plane) (γ_g)	deg
Angle of trajectory projection on xy-plane relative to the x-axis (θ_g)	deg
Mean temperature deviation from standard ISA temperature (dT_{std})	K
Air density (ρ)	Kg/m^3
Average cloud coverage ($cloud$)	-
Average solar incidence radiation (P_{sun})	W/m^2
Average airspeed (V_a)	m/s
Average airspeed components (V_{a_x}, V_{a_y} and V_{a_z})	m/s
Average wind speed (V_{W_a})	m/s
Average wind speed components ($V_{W_{a_x}}$, $V_{W_{a_y}}$ and $V_{W_{a_z}}$)	m/s
Average ground speed (V_{g_a})	m/s
Average ground speed components ($V_{g_{a_x}}$, $V_{g_{a_y}}$ and $V_{g_{a_z}}$)	m/s
Average airspeed for aerodynamic forces (V_{ave})	m/s
Average lift and drag (L and D)	N
Average lift and drag coefficients (C_L and C_D)	-
Average lift-to-drag ratio (L/D)	-
Average load factor (n)	-
Air path angle (γ_a)	deg
Bank angle (ϕ_a)	deg
Rate of climb (RC)	m/s
Average acceleration (a)	m/s^2
Average longitudinal inertial force ($m.a$)	N
Average ground friction force (F)	N
Average required thrust (T_{req})	N
Average required power (P_{req})	W

Average electric power (P_{ele})	W
Average energy or fuel flow (P_{flow})	W or N/s
Average segment time (dt)	s
Segment required energy or fuel (E_{used})	J or N
Segment consumed energy or fuel (E)	J or N
Segment energy left in battery or fuel in tank (E_{left})	- or N
Power setting (f)	-
Engine/motor speed (rpm)	rpm
Motor current (I)	A

Annex B – Detailed summary of the MPP’s Mission Analysis mode

The following is a summary of the methodology of the mission analysis mode of the Mission Planner Program developed by Coelho [11], which can be found in his thesis.

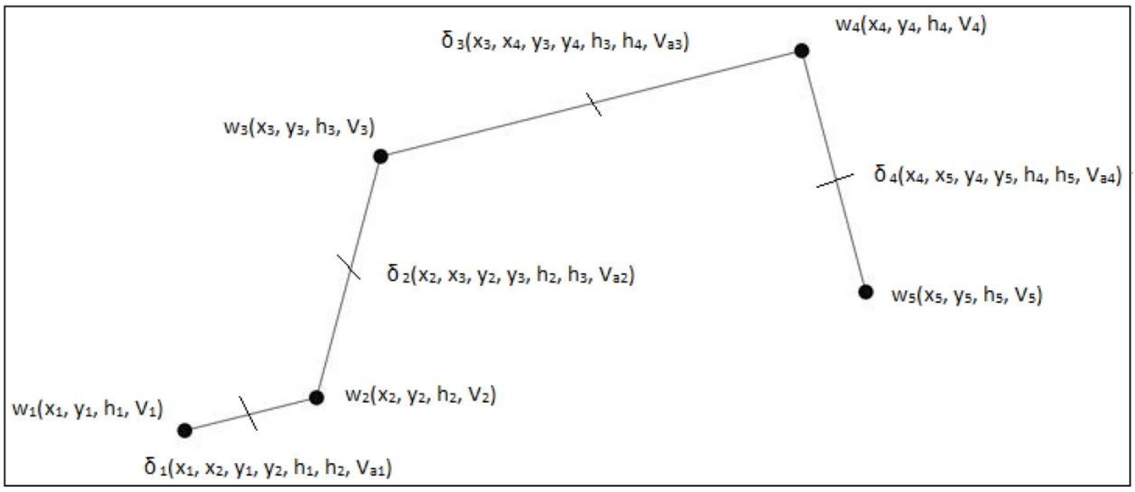
In the analysis mode, the calculation of the objective function and its constraints from the design variables is the main purpose. There are 4 design variables for each waypoint – the coordinates chosen by the user, which can either be of the Geodetic (GEO) type (latitude and longitude) or the East-North-Up (ENU) type (x and y positions), altitude and airspeed. The GEO and ENU sets of design variables are represented by the following equations, respectively:

$$dv^{4n} = (\varphi_1, \varphi_2, \dots, \varphi_n, \lambda_1, \lambda_2, \dots, \lambda_n, h_1, h_2, \dots, h_{n-1}, h_n, V_1, V_2, \dots, V_n) \qquad \varphi_i, \lambda_i, h_i, V_i \in \mathbb{N} \text{ for } 1 \leq \text{i} \leq \text{n} \tag{3.37}$$

$$dv^{4n} = (x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, h_1, h_2, \dots, h_{n-1}, h_n, V_1, V_2, \dots, V_n) \qquad x_i, y_i, h_i, V_i \in \mathbb{N} \text{ for } 1 \leq \text{i} \leq \text{n} \tag{3.38}$$

where i is the waypoint index, n is the total number of waypoints, φ and λ are the latitude and longitude of the Geodetic coordinates (in decimal degrees), h is the altitude of flight measured from the take-off ground elevation, V is the airspeed, and x and y are the East-North-Up coordinates (in meters). By default, the algorithm of analysis uses the ENU coordinate system, therefore, an internal conversion of GEO to ENU coordinates is provided, if Geodetic coordinates are used.

Afterwards, the calculation of the segment flight performance δ_j between two consecutive waypoints w_i and w_{i+1} is done, according to the design variables of those waypoints. This is visually represented in Figure 84.


w₄(x₄, y₄, h₄, V₄)
δ₃(x₃, x₄, y₃, y₄, h₃, h₄, Va₃)
w₃(x₃, y₃, h₃, V₃)
δ₄(x₄, x₅, y₄, y₅, h₄, h₅, Va₄)
δ₂(x₂, x₃, y₂, y₃, h₂, h₃, Va₂)
w₅(x₅, y₅, h₅, V₅)
w₂(x₂, y₂, h₂, V₂)
w₁(x₁, y₁, h₁, V₁)
δ₁(x₁, x₂, y₁, y₂, h₁, h₂, Va₁)
Figure 84 - Theoretical example of a route with 5 waypoints. δ_j represent the segment flight performance calculated between consecutive waypoints [11]

Firstly, the segment total distance is calculated by:

$$ds_j = \sqrt{dg_j^2 + dh_j^2} \tag{3.39}$$

where dg_j is the segment ground distance:

$$dg_j = \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \tag{3.40}$$

and dh_j is the segment height variation:

$$dh_j = h_{j+1} - h_j \tag{3.41}$$

Next, the MPP calculates the average segment airspeed:

$$V_{a_j} = \frac{V_{i+1} + V_i}{2} \tag{3.42}$$

followed by the calculation of average segment windspeed components, dependent on the atmospheric data model:

$$V_{W_{a_{x_j}}} = \frac{V_{W_{x_i}} + V_{W_{x_{i+1}}}}{2} \tag{3.43}$$

$$V_{W_{a_{y_j}}} = \frac{V_{W_{y_i}} + V_{W_{y_{i+1}}}}{2} \tag{3.44}$$

$$V_{W_{a_{z_j}}} = \frac{V_{W_{z_i}} + V_{W_{z_{i+1}}}}{2} \tag{3.45}$$

where the waypoint windspeed components are in turn dependent on the windspeed magnitude V_{W_i} and the wind direction $\theta_{W_i} \in [0, 360[$ (degrees, where 0 represents the North $\rightarrow$ South way):

$$V_{W_{x_i}} = V_{W_i} \cos\,(90 - \theta_{W_i}) \tag{3.46}$$

$$V_{W_{y_i}} = V_{W_i} \sin\,(90 - \theta_{W_i}) \tag{3.47}$$

$$V_{W_{z_i}} = 0 \tag{3.48}$$

The angle of the wind direction is two-dimensional, so $V_{W_{z_i}}$ is considered as zero. The average segment windspeed is therefore given by:

$$V_{W_{a_j}} = \sqrt{{V_{W_{a_{x_j}}}}^2 + {V_{W_{a_{y_j}}}}^2 + {V_{W_{a_{z_j}}}}^2} \tag{3.49}$$

Afterwards, the flight path direction angles – of the trajectory relative to the ground (xy-plane) (γ_g), and of the projection on xy-plane relative to the x-axis (θ_g) – are calculated, between two consecutive waypoints:

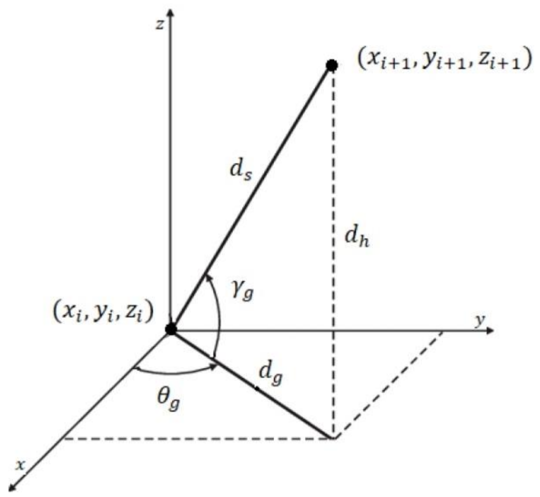


Figure 85 – Flight path direction scheme [11]

$$\theta_{g_j} = \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right) = \arcsin\left(\frac{dg_{y_j}}{dg_{x_j}}\right) \tag{3.50}$$

$$\gamma_{g_j} = \arctan\left(\frac{dh_j}{ds_j}\right) \tag{3.51}$$

With these angles known, the average segment ground speed components are calculated by the following equations:

$$V_{g_{a_{x_j}}} = V_{g_{a_j}} \cos\bigl(\gamma_{g_j}\bigr) \cos\bigl(\theta_{g_j}\bigr) \tag{3.52}$$

$$V_{g_{a_{y_j}}} = V_{g_{a_j}} \cos\left(\gamma_{g_j}\right) \sin\left(\theta_{g_j}\right) \tag{3.53}$$

$$V_{g_{a_{z_j}}} = V_{g_{a_j}} \sin\left(\gamma_{g_j}\right) \tag{3.54}$$

After a deduction process described in [11], the calculation of $V_{g_{a_j}}$ is obtained by solving the equation:

$$a.V_{g_{a_j}}{}^2 + b.V_{g_{a_j}} + c = 0 \tag{3.55}$$

where:

$$a = 1$$
$$b = V_{W_{a_{x_j}}}{}^2 + V_{W_{a_{y_j}}}{}^2 + V_{W_{a_{z_j}}}{}^2 - V_{a_j}{}^2$$
$$c = V_{W_{a_{x_j}}}{}^2 + V_{W_{a_{y_j}}}{}^2 + V_{W_{a_{z_j}}}{}^2 - V_{a_j}{}^2$$

The time in each segment (dt_j) and the total mission time (t_{total}) can then be calculated with:

$$dt_j = \frac{ds_j}{V_{g_{a_j}}} \tag{3.56}$$

$$t_{total} = \sum_{j=1}^{n-1} dt_j \tag{3.57}$$

After knowing the time of the mission, calculations to find the mission energy then proceed, starting with the calculation of the air path angle γ_{a_j}:

$$\gamma_{a_j} = \arcsin\left(\frac{V_{a_{z_j}}}{V_{a_j}}\right) \tag{3.58}$$

where $V_{a_{z_j}}$ can also be represented as the average rate of climb in the respective segment, RC_j.

For the calculation of the aerodynamic forces, the following equations apply:

$$V_{ave_j} = \sqrt{\frac{V_j^2 + V_{j+1}^2}{2}} \tag{3.59}$$

$$L_j = \frac{W_i.\cos\left(\gamma_{a_j}\right)}{\cos\left(\phi_{a_j}\right)} \tag{3.60}$$

$$C_{L_j} = \frac{L_j}{\frac{1}{2}\rho.V_{ave_j}{}^2} \tag{3.61}$$

$$C_{D_j} = f\left(C_{L_j}\right) \tag{3.62}$$

$$D_j = \frac{1}{2}\rho.\, {V_{ave_j}}^2 S.\, C_{D_j} \tag{3.63}$$

where V_{ave_j} is the squared average velocity, L_j is the average segment lift, D_j is the average segment drag, C_{L_j} is the average segment lift coefficient, C_{D_j} is the average segment drag coefficient (a function depending on the lift coefficient), W_i is the aircraft weight, ϕ_{a_j} is the average bank angle, S is the wing area, and ρ is the air density.

Following this, calculation of the average acceleration a_j, average inertial force Q_{IF_j} and average rolling friction force F_j leads to the calculation of the average required thrust T_j, which is shown by Equations 3.64 through 3.67:

$$a_j = \frac{V_{g_{a_{j+1}}}}{dt_j} \tag{3.64}$$

$$Q_{IF_j} = \frac{W_j}{g} a_j \tag{3.65}$$

$$F_j = \mu_j (W_j - L_j) \tag{3.66}$$

$$T_j = D_j + W_j \sin\left(\gamma_{a_j}\right) + Q_{IF_j} + F_j \tag{3.67}$$

where μ_j is the coefficient of ground rolling friction. Finally, the average required power P_{req_j}, average electric power P_{e_j} and consumed energy at each segment dE_j are calculated by:

$$P_{req_j} = T_j V_{ave_j} \tag{3.68}$$

$$P_{e_j} = U_j I_j \tag{3.69}$$

$$dE_j = P_{e_j} dt_j \tag{3.70}$$

where U_j and I_j are the input motor voltage and current, respectively. After calculations in the solar model, the positive required energy in each segment is subtracted by the total energy gained from solar power, resulting in the final energy balance and power flow at each segment as well as at the end of the mission, values which are shown in the “mission_out.txt” and defined in Annex A.

Annex C - Default representation of propeller power coefficients and efficiency values

$$C_p = C_{p,0} \times \left[A_0 + \sum_{i=1}^{4} \left(A_i \left(\frac{J_{prop}}{J_{max}} \right)^i \right) \right]$$

$$\eta_p = \eta_{p,max} \times \left[B_0 + \sum_{i=1}^{6} \left(B_i \left(\frac{J_{prop}}{J_{max}} \right)^i \right) \right]$$

where:

$$J_{max} = C_1 d + C_2 p + C_3 d^2 + C_4 dp + C_5 p^2 + C_6 d^3 + C_7 d^2 p + C_8 dp^2 + C_9 p^3$$

$$C_{p,0} = D_1 d + D_2 p + D_3 d^2 + D_4 dp + D_5 p^2 + D_6 d^3 + D_7 d^2 p + D_8 dp^2 + D_9 p^3$$

$$\eta_{max} = E_1 d + E_2 p + E_3 d^2 + E_4 dp + E_5 p^2 + E_6 d^3 + E_7 d^2 p + E_8 dp^2 + E_9 p^3$$

and the default coefficients are:

$$\bar{A} = \begin{bmatrix} 0.9999747473830 \\ 0.0026886303943 \\ -0.0542821394531 \\ -0.8141198610786 \\ 0.2382888347204 \\ -0.1060271581734 \\ 0.0222789611099 \end{bmatrix} \quad \overline{B} = \begin{bmatrix} 0.0000000000000 \\ 2.8358158896651 \\ -4.6740787983266 \\ 17.2094772778345 \\ -45.734194221401 \\ 55.789219497612 \\ -25.395785093511 \end{bmatrix}$$

$$\bar{C} = \begin{bmatrix} 0.706462000000 \\ -0.046405100000 \\ 0.074350100000 \\ 0.001069860000 \\ -0.001664110000 \\ -0.000007715000 \\ -0.000006521000 \\ 0.000008688670 \\ 0.000002563530 \\ -0.000000703183 \end{bmatrix} \quad \overline{D} = \begin{bmatrix} 0.050916200000 \\ -0.005511640000 \\ 0.007489280000 \\ 0.000144156000 \\ -0.000239091000 \\ 0.000065509200 \\ -0.000002407300 \\ 0.000005544700 \\ -0.000003824100 \\ 0.000000875200 \end{bmatrix} \quad \bar{E} = \begin{bmatrix} 0.375474000000 \\ 0.013321100000 \\ 0.014884800000 \\ -0.000358479000 \\ 0.000020627100 \\ -0.000189967000 \\ 0.000003644830 \\ -0.000004047110 \\ 0.000004028760 \\ -0.000000467311 \end{bmatrix}$$

Annex D – List of Design Parameters and Constraint Functions

Design Parameters	
Description	**Units**
Mission Initial Hour (t_{init})*	h
Latitude (φ)	deg
Longitude (λ)	deg
Altitude (h)	m
Airspeed (V)	m/s

*added in the present work

Equality Constraint Functions	
Description	**Units**
Engine setting = Engine setting max ($\delta = \delta_{limit}$)	%
Motor current = Motor current max ($I = I_{ref}$)	A

Inequality Constraint Functions	
Description	**Units**
Engine setting ≤ Engine setting max ($\delta \leq \delta_{limit}$) at design conditions	%
Motor current ≤ Motor current max ($I \leq I_{max}$) at design conditions	A
Engine rpm ≤ Engine rpm max ($N \leq N_{max}$) at design conditions	rpm
$\left(\frac{V_{min}}{V_s}\right)^2 C_L \leq C_{L_{max}}$ at design conditions	-
Engine setting ≤ Engine setting max ($\delta \leq \delta_{limit}$) at static conditions	%
Motor current ≤ Motor current max ($I \leq I_{max}$) at static conditions	A
Engine rpm ≤ Engine rpm max ($N \leq N_{max}$) at static conditions	rpm
Segment height min ≥ [sel. value] ($h_j \geq h_{ref}$)	m
Segment power min ≥ [sel. value] ($P_j \geq P_{ref}$)	W
Mission energy left min ≥ [sel. value] ($E_{left} \geq E_{ref}$)	%