Optimisation of the propulsion chain of an inflatable drone

Louis BAETENS*, Pedro GALEB*
François DEFAY†
*Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055 Toulouse, FRANCE
Email: {louis.baetens,pedro.grueiro-galeb}@student.isae-supaero.fr
†Institut Supérieur de l'Aéronautique et de l'Espace (ISAE-SUPAERO), Université de Toulouse, 31055 Toulouse, FRANCE
Email: francois.defay@isae-supaero.fr

***Abstract*—This project aims to develop an optimization method for the propulsion chain of a drone, and more particularly the DIODON, the inflatable drone of the eponymous start-up created at the ISAE-SUPAERO. The DIODON is already functional, and can fly for about 15 minutes, which can be improved. To achieve this goal, the method should find the best components for a given constrained situation. A computing environment will be developed, with the requirements that the code should be readable, scalable, reliable and open-source as much as possible. It will include the optimization part itself, the database on which it relies upon, and a graphical interface.**

I. General context

Although drones are a relatively recent technological application, which is not yet fully regulated, several works and studies on this subject exist, some of which are devoted to the study of optimization of the propulsion chain It is easy to understand the reason, when one takes into account the fact that the propulsion system contributes to more than 60% of the drone weight in a general case [1].

The electric propulsion of drones normally consists of the propeller, the electric motor, the battery, the gearbox, the connectors and cables, and possibly a cooling system. Most of the studies and articles related to the subject have a similar methodology for approaching the problem: they divide the propulsion system into four parts (motors, propellers, ESC and batteries). Indeed, even if all the elements are important, the latter are the most influential on the performance of the drone, by their mass and their efficiency [1].

The optimization of the chain of propulsion is an non-trivial process caused by, among other things, the non-linear characteristics of the propellers [4]. Moreover, the batteries expose an unwanted behaviour, since the delivered voltage change depending on the discharge and the time [3]. These two examples reflect the importance of the details for a software that modelize a drone, so as to obtain results that are close to the reality.

One can find various softwares dedicated to the search of the best combination of motors, propellers, ESC and batteries, each of these working in a different way and giving different results.

II. Existing computing environments

In order to justify the creation of a new optimization software, and to take inspiration from the qualities and flaws of the other existing codes, a state of the art is realized below. It is obviously non-exhaustive, and only the most relevant are presented here.

A. DroneCalc [3]

Optimization software created as part of an End-of-Studies Project at the ISAE-SUPAERO, it relies on a exhaustive and large database. The user inputs the boundary values for the drone mass, the diameter of the propellers, and the necessary flight conditions. In output, it retrieves a ranking of the best propeller-motor pairs. Its execution time, of the order of the hour because of its exhaustiveness and its architecture, is problematic. In addition, it uses different databases, which are not necessarily reliable nor harmonised. Finally, its code is not very readable, and therefore difficult to maintain and improve. These shortcomings leaded to the decision that the development of this code will not be continued for this project.

B. eCalc [5]

One of the most used online softwares. It gives the possibility of optimizing the propulsion chain of a multirotor or a fixed wing drone. The input parameters are numerous, and based on the boundaries that one wishes to impose on each component. Its main problem lies in the fact that its code is not accessible and, despite its speed of execution, it is complicated to verify the given results. Also, one can not access the used database, which means that the reliability is not assured.

C. TetaCalc [6]

This software is actually an Excel sheet. It is rather oriented towards fixed-wing drones. It is based on several modules, including the geometry of the drone, its propulsion, and the aerodynamic block. Its database is open and modifiable, and based on experimental data. This is therefore positive, since it is developed for the purpose of use in a real and practical case. However, it is rather obscure at first sight, and not necessarily adapted to the case of a multirotor like the DIODON. In addition, it seems impossible to optimize for several flight points simultaneously.

III. Requirements

In the light of the above softwares, and their respective defects, it has been decided to start from scratch. Before starting the project, the requirements for it had to be decided.

Concerning the global goals:

- The computing environment needs to be easy to use and to modify if needed
- It has to provide accurate results
- The execution time of the optimization process should be small enough
- All the parts of the projects should be readable and quite easy to understand
- If possible, the project will be developed with open-source tools only, to maximise the number of users and contributors

It leads to the need for a user interface and a database. The user interface should be an improvement of the DroneCalc one, therefore its requirements are:

- The UI will be clear and with sufficient explanation
- The units should be stated clearly
- The navigation between the different parts of the program needs to be easy and intuitive

Concerning the database, the needs are stated below:

- The structure has to be clear for the user
- Possibility has to be given to add new data easily, but consistently. The units should be the same, and it should not be possible to add incomplete components.
- Quick search must be possible
- The existing data must be easily accessible

IV. Developed solution

For the new software, three distinct parts have been identified: the user interface, the database management, and the optimization part itself. The last part is treated by Daniel Barraza, thus it is not mentioned here. In the following, the new software is called PyDroneCalc, as it is developed in Python. Python has been chosen as it is more convenient: it posses lots of libraries, it is open-source and free, thus it matches with the requirements stated above, more than Matlab for example.

A. Graphical User Interface

The main idea of the creation of a Graphical User Interface is to create an easy way, for users who are not familiar with the world of engineering and especially programming, to explore all the functions in the code. In addition, knowing that the program can be modified in the future by adding new functions or correcting some errors, it must have a readable code to make life easier for future developers.

The interface is also intended to integrate the optimization part and the database part. All this is done while keeping in mind the mentioned goals, readability and easy understanding for users who are not necessarily experts in this area.

Firstly, an interface inspired by DroneCalc was created, with the same options and the same sequence of steps to start the optimization. However, this model was not compatible with PyDroneCalc, which despite having a similar name and a similar objective, are two extremely different codes. Therefore a new logic of interfaces and steps until the end of the optimization had to be created and adapted to the functions treated by Daniel Barraza, with all the possibilities created in the database part.

a. Graphical User Interface structure

The current version of the Graphical User Interface consists of 10 classes, 7 different interfaces, 2 sub interfaces for the addition of 2 more operating points and an executable class for the program to be started.

The diagram below shows in simplified form how the relationship between the classes works.

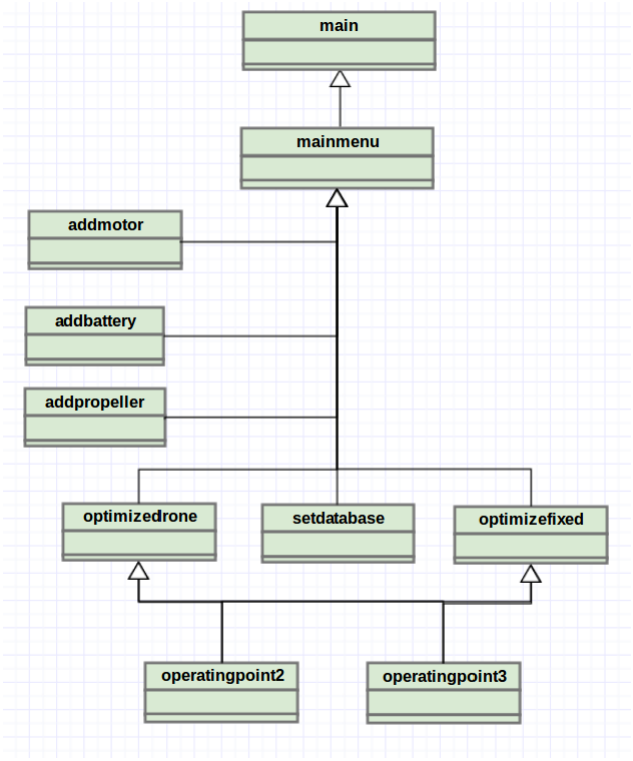


Fig. 1. Simple operation schema of the Graphical User Interface

The mainmenu.py, presents a short project presentation text, followed by a menu with the functions that may be interesting for the user.

The classes addbattery.py, addmotor.py and addpropeller.py are only dependent on mainmenu.py, and work very similarly: the user complete a registration form with the information pertinent to the item (name, company, weight, ...) and automatically the item and all its characteristics are added to the database. The class addpropeller.py also includes a last step of adding a csv formatted file with the curves of the propeller, which will be treated in the database.

The class setdatabase.py (see figure 2) is about adjusting the database for the user's requirements. He can impose one or more items such as the propeller, or create a sub database

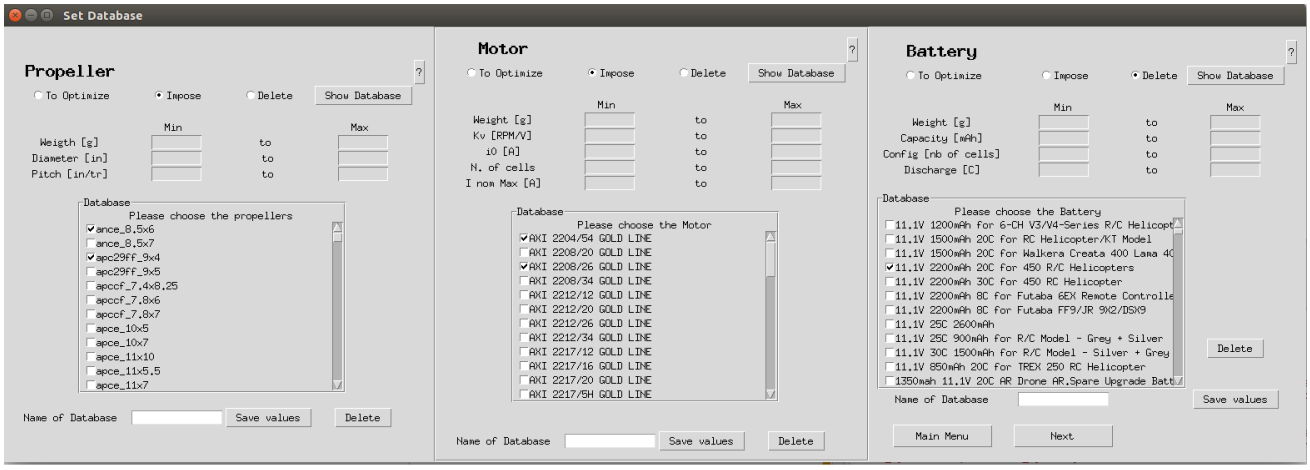


Fig. 2. Window generated by setdatabase.py

based on the properties range (weight, diameter, ...). The same procedure can be done for the battery and motor. This interface also includes the option of deleting items from the databases. When an item is deleted from the database it is automatically deleted from all the sub-databases in which it is present.

The classes optimizefixed.py and optimizedrone.py (see figure 3) are the last step before starting the optimization. The user chooses the most interesting sub-database for his requirements, and gives the operating point information (additional operating points can be added through the subinterfaces operatingpoint2.py and operatingpoint3.py).

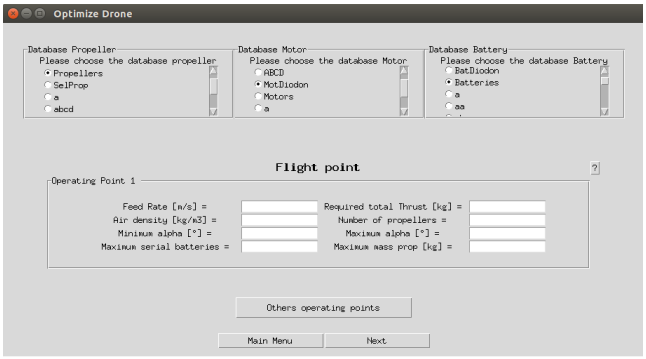


Fig. 3. Window generated by optimizedrone.py

b. Python codes

There are several GUI Programming toolkits for Python, however for this software the choice was Tkinter. Despite the various positive points presented by the others toolkit, Tkinter was chosen due to its versatility, being compatible with Windows and Macintosh systems, the fact that the installation of the basic Python package already includes Tkinter and it also offers a vast online documentation in English with several examples and applications, making life easier for programmers.

To summarize, Tkinter consists of a number of modules. The Tk interface is provided by a binary extension module named _Tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers.

All classes created during the code are initialised creating a Tk root widget by the sentence root = Tk(). A root widget is an ordinary window where the others widgets will be placed. It is important to create only one root widget for each program, that is why every time a change of interface is needed, the previous interface root widget is destroyed.

In the code several widgets were used, such as Button, Frame, Label, Scrollbar among others. The placement methods used in it were always .grid(), with rare exceptions in which-pack() was necessary. The recommendation often found in online documentation of not mixing the .grid() method and the .pack() method to avoid positioning conflicts has always been respected, and is recommended for the future development of this software.

c. Articulation

It is important to remember that in this project the graphic interfaces played an important role related to the assembly and operation of the optimization code and the database codes. More than a simple dialogue between the user and the code, they also make a dialogue between the other code's parts.

That is why it is very important to have clarity in the nomenclature of variables and to maintain consistency throughout the program, in order to avoid problems. As the code presents a considerable number of variables that may or may not be used in different situations, this coherence is fundamental.

d. Easy to use and to modify

The interfaces were created and ordered in order to be more intuitive for users, they all have a small button with a [?] symbol in the upper right corner to inform or answer the possible questions that the user could have. For the same purpose, the order of the interfaces and the way the input parameters are requested are also designed to facilitate the use.

In addition every time the user enters absurd values or leaves some mandatory field empty during the addition of items to

the database, messages are sent to him saying what error was committed and directing what to do to correct the error. Still regarding the consistency at the level of user data entry, all relevant fields inform in which units the data should be sent to have coherent results.

In order to simplify possible changes that the graphic interfaces can receive, the code presents several comments indicating what the lines of code generate in the interface. All the variables and widgets created have a name that briefly explain what the variable represents, this facilitates the creation of functions that deal with the values received and will also be of great help to anyone who will generate changes in the interface. Besides all the buttons present functions created in the final part of the class, the functions also present comments quickly explaining its objective and how it works.

As English is the most common language for occidental people, all interfaces, comments, and error messages or information messages were written in English so that the program reaches the largest number of possible users.

B. Database

As stated previously, the various data present in DroneCalc are in many files, their formatting differ, and it is difficult to add new data. Therefore, the data structure has been redesigned.

It has been decided to implement a database that is exploited with SQL. To stay with the open-source philosophy, MySQL has been chosen for the relational database management system. Finally, the library pymysql is used to make the link between the database requests and Python.

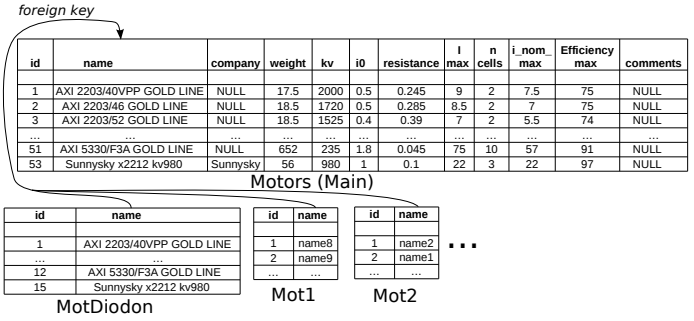
In fact, three databases are created: one for the batteries, one for the motors, and one for the propellers. The idea is that it is possible to add other databases based on their structure, in following projects that aim to improve this version of PyDroneCalc. For example, one could add an ESC database, or one for the electrical wires.

Each database is composed of several Tables.

a. Database structure

The structure is one of the most important part of any database, and has to be thought thoroughly. Concerning the motors and batteries databases (called MOTORS_db and BATTERIES_db), it was quite simple. The main table ('Motors' or 'Batteries') contains the name of the component, and its various properties (see figures 4 and 5 for the details).

Then, the user should be able to select some of the components that he wants to use for his optimization. This selection can be made in two different ways: he can either select directly the names of the components in a list (see the GUI for the example), or impose a range for the properties (weight, capacity, kv, and so on). Once he selected the desired components, the output has to be stored in a permanent manner, such that if he wants to realise another optimization with the same components, he can do it quickly. Therefore, the database structure is such that new tables can be created, with only an id (which is auto-incremented) and the names of the components. Those tables (referred by 'BatDiodon', 'Bat1', 'Bat2', in figure 5) were designed such that the user can create many tables without taking too much space, and all the components properties are stored in the main table.



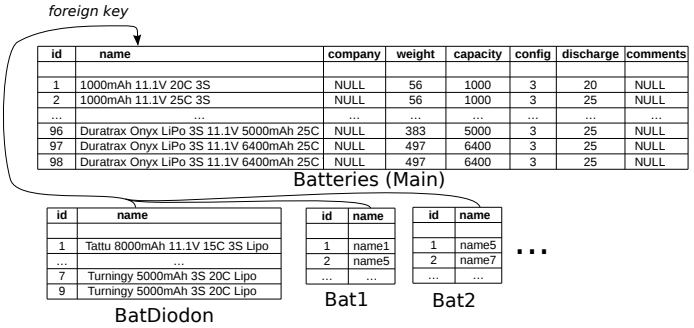
id	name	company	weight	kv	i0	resistance	I max	n cells	i_nom_ max	Efficiency max	comments
1	AXI 2203/40VPP GOLD LINE	NULL	17.5	2000	0.5	0.245	9	2	7.5	75	NULL
2	AXI 2203/46 GOLD LINE	NULL	18.5	1720	0.5	0.285	8.5	2	7	75	NULL
3	AXI 2203/52 GOLD LINE	NULL	18.5	1525	0.4	0.39	7	2	5.5	74	NULL
...	...	...	...	...	...	...	...	...	...	...	...
51	AXI 5330/F3A GOLD LINE	NULL	652	235	1.8	0.045	75	10	57	91	NULL
53	Sunnysky x2212 kv980	Sunnysky	56	980	1	0.1	22	3	22	97	NULL

id	name
1	AXI 2203/40VPP GOLD LINE
...	...
12	AXI 5330/F3A GOLD LINE
15	Sunnysky x2212 kv980

id	name
1	name8
2	name9
...	...

id	name
1	name2
2	name1
...	...

Fig. 4. MOTORS_db database



id	name	company	weight	capacity	config	discharge	comments
1	1000mAh 11.1V 20C 3S	NULL	56	1000	3	20	NULL
2	1000mAh 11.1V 25C 3S	NULL	56	1000	3	25	NULL
...	...	...	...	...	...	...	...
96	Duratrax Onyx LiPo 3S 11.1V 5000mAh 25C	NULL	383	5000	3	25	NULL
97	Duratrax Onyx LiPo 3S 11.1V 6400mAh 25C	NULL	497	6400	3	25	NULL
98	Duratrax Onyx LiPo 3S 11.1V 6400mAh 25C	NULL	497	6400	3	25	NULL

id	name
1	Tattu 8000mAh 11.1V 15C 3S Lipo
...	...
7	Turningy 5000mAh 3S 20C Lipo
9	Turningy 5000mAh 3S 20C Lipo

id	name
1	name1
2	name5
...	...

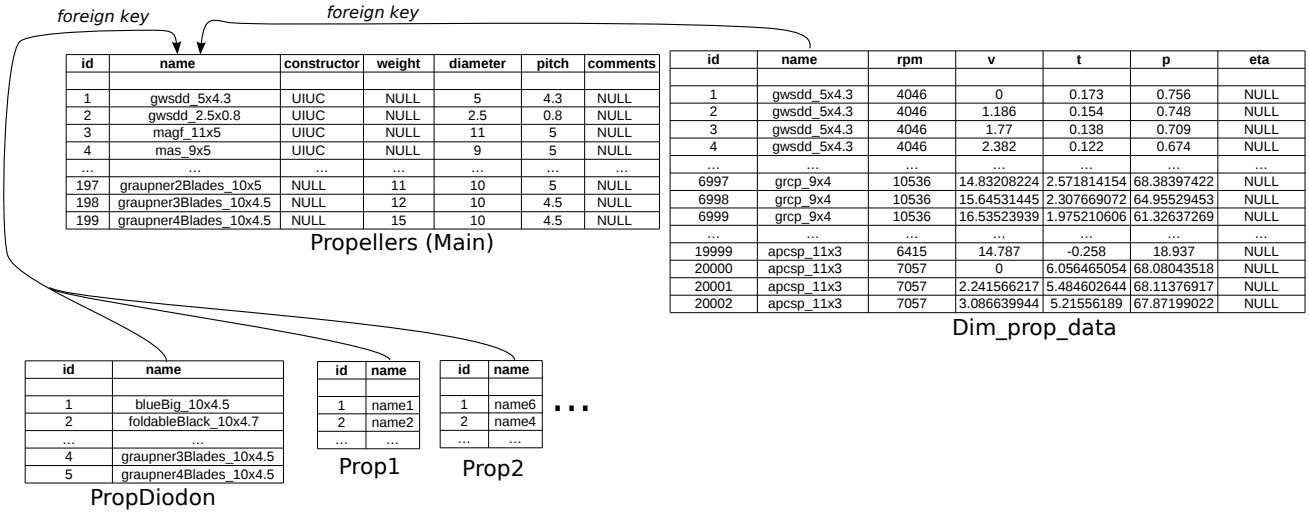
id	name
1	name5
2	name7
...	...

Fig. 5. BATTERIES_db database

Concerning the propellers database PROPELLERS_db, it is a bit more complicated. Indeed, propellers have scalar properties, such as the weight or the diameter, but also some others that should be stored in a vector fashion, such as the tuples [rpm, speed, thrust, power], that correspond to different operating points. Therefore, the most efficient structure that has been found is the one exposed in figure 6. The main table 'Propellers' is like the one for the motors and batteries. Another table, called 'Dim_prop_data', stores the vectors. It is therefore a large table, containing all the operating points of all the propellers. The link between the two tables is made by the name of the propeller. As before, subtables with only the names of the propellers can be created (see figure 6). This organisation allows the requests to be efficient.

b. Unicity and validity constraints

To make the databases more robust, constraints need to be set. First of all, apart for the table 'Dim_prop_data', the name in each table is unique, such that it is impossible to add another component with the same name but different properties. For the 'Dim_prop_data' table, the uniqueness is ensured by the tuple [name, rpm, v], which means that for a propeller, only one couple rpm/speed can exist. Indeed, for this couple, only one thrust and power can be deduced.



id	name	constructor	weight	diameter	pitch	comments
1	gwsdd_5x4.3	UIUC	NULL	5	4.3	NULL
2	gwsdd_2.5x0.8	UIUC	NULL	2.5	0.8	NULL
3	magf_11x5	UIUC	NULL	11	5	NULL
4	mas_9x5	UIUC	NULL	9	5	NULL
...	...	...	...	...	...	...
197	graupner2Blades_10x5	NULL	11	10	5	NULL
198	graupner3Blades_10x4.5	NULL	12	10	4.5	NULL
199	graupner4Blades_10x4.5	NULL	15	10	4.5	NULL

id	name	rpm	v	t	p	eta
1	gwsdd_5x4.3	4046	0	0.173	0.756	NULL
2	gwsdd_5x4.3	4046	1.186	0.154	0.748	NULL
3	gwsdd_5x4.3	4046	1.77	0.138	0.709	NULL
4	gwsdd_5x4.3	4046	2.382	0.122	0.674	NULL
...	...	...	...	...	...	...
6997	grcp_9x4	10536	14.83208224	2.571814154	68.38397422	NULL
6998	grcp_9x4	10536	15.64531445	2.307669072	64.95529453	NULL
6999	grcp_9x4	10536	16.53523939	1.975210606	61.32637269	NULL
...	...	...	...	...	...	...
19999	apcsp_11x3	6415	14.787	-0.258	18.937	NULL
20000	apcsp_11x3	7057	0	6.056465054	68.08043518	NULL
20001	apcsp_11x3	7057	2.241566217	5.484602644	68.11376917	NULL
20002	apcsp_11x3	7057	3.086639944	5.21556189	67.87199022	NULL

id	name
1	blueBig_10x4.5
2	foldableBlack_10x4.7
...	...
4	graupner3Blades_10x4.5
5	graupner4Blades_10x4.5

id	name
1	name1
2	name2
...	...

id	name
1	name6
2	name4
...	...

Fig. 6. PROPELLERS_db database

As the name is unique, it seems natural to make the link between the tables from it. It is done by the so-called 'foreign keys'. They assure that the names in the non-main tables already exist in the main tables, and are therefore valid. Furthermore, this allow to delete all the references to a component just by deleting the corresponding row in the main table, without to do it in all the tables.

In the requirements, it is stated that the user should not add components with incomplete data. Therefore, for each database, some of the columns can not be left empty in the main table. For the BATTERIES_db, in the table Batteries, it concerns the fields name, weight, capacity, config and discharge. For the MOTORS_db, in the table Motors, it concerns the fields name, weight, kv, i0 and n_cells. For the PROPELLERS_db, in the table Propellers, it concerns the fields name, diameter and pitch; and in the table Dim_prop_data the fields name, v, t, p (t and p being in reality T/ρ and P/ρ)

c. Python codes

Then, various code have been developed in Python. The idea here, is that the user does not need to know any of the SQL syntax to interact with the database. He just has to call a Python function, with the required inputs, and the desired outputs or modifications will be produced. However, minimum knowledge in SQL are required if the project has to be continued. The functions are made in a way that the graphical interface can use them easily.

The following codes were produced. Only a short description of them is given here, as many comments are written in the corresponding files.

database_manipulation.py allows the user to connect to MySQL. It can be used to use, restore, save or delete a database.

table_manipulation.py is the largest code. It has many functions in it. Among them, we can cite the creation of the different tables with their description and constraints, the selection of the data by specifying property ranges, the creation of subtables from the names of the components, the addition of a component to a database, and other functions linked to the manipulation of the tables with SQL.

plot_characteristics.py is called to plot the characteristics curves of the propellers. It can take the name of a propeller, fetch the needed data, and plot them.

save_display_data.py is used to display the properties of the components in a pretty fashion. The user specify the name of the database, and it produces a text file with the columns as displayed by a SQL query. The text file is then open, such that the functions of this code can be called directly from the GUI.

C. Expanding the database

For the motors and the batteries, databases can be found on the internet. Their trustworthiness can vary, but even the constructors give the specifications. For the propellers however, it is much more difficult. The diameter and pitch can be found on the constructors website, but the curves can not (or are obviously skewed). A good database is the UIUC one [2], that has been validated by a comparison with a study from the Ohio State University (see [3] for more information). Therefore, its data have been put into the PROPELLERS_db.

DIODON wanted to test some other propellers they could set on their drones. Therefore, they had been put on the test bench. However, this test bench can only provide the static setup, so their data is incomplete. The results are plotted in figures 7 and 8.

The blueBig is a standard propeller, used as a reference. The four others are foldable ones. The Graupner have two, three or four blades. In figure 7, the standard representation of the propellers curves is given. One can observe that their behaviour is coherent with what can be expected. However, it is more useful to compare their 'efficiency'. As the definition of the dynamic efficiency fails for the static case (the efficiency is null), one can redefine the static efficiency as being proportional to the thrust per unit power. Thus, the thrust versus

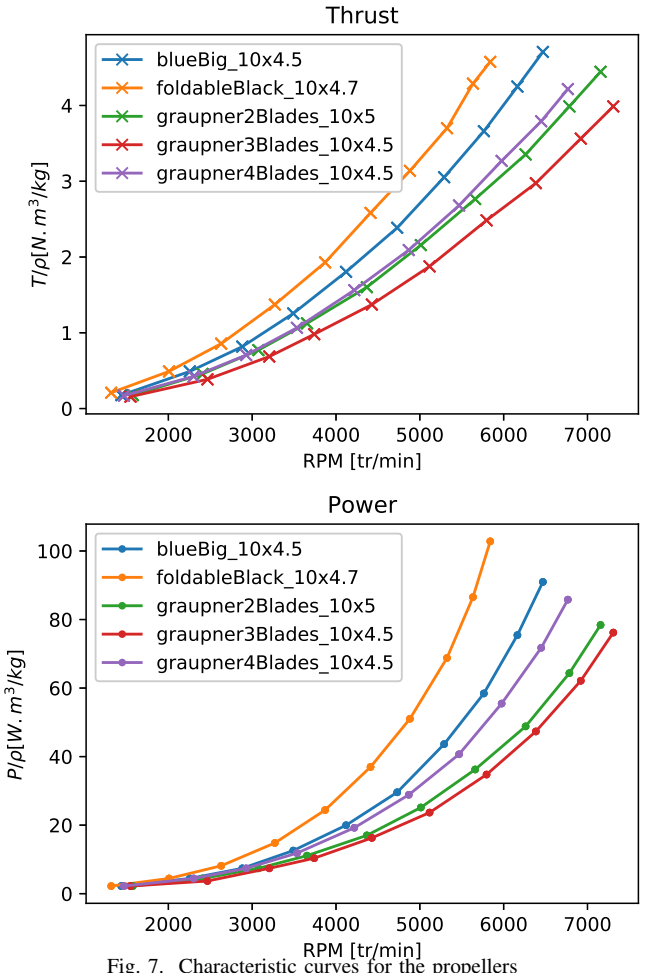


Fig. 7. Characteristic curves for the propellers

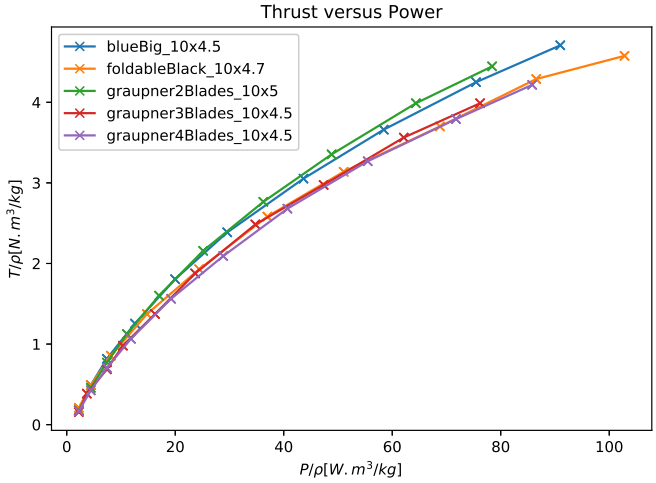


Fig. 8. Comparison between the tested propellers in the static case

the power is plotted in figure 8. One can observe that the most efficient one is the Graupner with two blades, but all of them are pretty close. Therefore, for the static case, it does not make sense for the DIODON to give preference to one of these propellers based on the efficiency defined above.

As the optimization part of this project is not finished yet, the best components for the DIODON can not be deduced at the time of writing.

V. Future development

This project lays the foundations for a promising computing environment. All the tasks could not been tackled in the allocated time, thus the authors see different ways that could improve the software.

More functions could be implemented. In particular, the controllability of the drone even with the loss of a motor can be added as an optimization constraint. The wind conditions and the inclination of the drone regarding the horizontal could also be taken into account.

The database has to be expanded. More propellers could be added, but for that one needs a good test bench, possibly with a wind tunnel. The batteries should be tested, and their discharge curve could be taken into account in the program. The motors performances could also be tested, and their curves used to obtain a more realistic approach than just a linear representation. For this matter, the database structure could be inspired from the one for the propellers.

The database could be put on a server, such that different users would be able to access it and modify it.

The price of the components could be set as a limiting factor, or as a constraint.

As the user interface should be easy to use, but also look neat, one could try to make it less bulky and more appealing. Other fonts could be tried, the colours changed, and the general appearance improved.

Knowing that the code of optimisation now takes into account the role of ESC, an ESC database could be implemented and showed in the interface as it is already done with the batteries, propellers and motors.

All those suggestions and ideas are proposed in the hope that this software will be improved in the future, keeping in mind that some of them could be achieved quickly, and some others not.

Acknowledgment

The authors would like to thank Francois DEFAY for his availability and his advice, Jean-Francois DASSIEU for his help with the test bench, Roman LUCIANI and Antoine TOURNET for their clarifications on the constitution and operation of a drone, and Christophe GARION for his advice on the databases.

References

[1] Ohad Gur and Aviv Rosen. Optimizing electric propulsion systems for unmanned aerial vehicles. *Journal of aircraft*, 46(4):1340–1353, 2009.

[2] Gavin K. Ananda John B. Brandt, Robert W. Deters and Michael S. Selig (University of Illinois Urbana-Champaign). Uiuc propeller data site, http://m-selig.ae.illinois.edu/props/propdb.html. 2015.

[3] Victor Lebrun. Optimisation de la chaîne de propulsion d'un drone avion ou vtol. 2015.

[4] Øyvind Magnussen. Multirotor design optimization: The mechatronic approach. 2015.

[5] Markus Müller. ecalc, www.ecalc.ch.

[6] Guillaume Rouby. Tetacalc, www.aerotrash.over-blog.com.