**Tracjectory optimisation for terminal flight stage of Urban air mobility vehicles**

A. Barr

Department of Aerospace Engineering, University of Bristol, Queen’s Building, University Walk, Bristol. BS8 1TR. UK.

**aBSTRACT**

**Keywords:** Trajectory Optimisation, eVTOL, Urban air mobility, Aeroacoustics, Noise

1. **iNTRODUCTION**

With the continuous development of technologies beneficial to Urban Air Mobility vehicles such as improvements to electrical propulsion and batteries, many concept vehicles of different configurations have either been announced as in development **[1]**, all aiming to provide a new form of green transport to tackle problems in the urban environment. Due to the potential of widespread adoption of these UAM aircraft in urban areas around the globe, particular attention must be paid to the noise footprint of these aircraft. Indeed, the effects of noise pollution on health are well known **[2]** and the practice of implementing not just technological improvements **[3]** to aircraft, with a particular focus on reducing noise from their propulsion, to reduce their footprint but also tactical and operational considerations, for example Heathrow’s dual runway operations **[4]**, which switches the active landing and departing runways on a rota to avoid certain residence living under the approach paths to each runway being consistently exposed to landing aircraft noise. Furthermore, Air Traffic Control operational planners work to reduce noise impact on local populations through the design and enforcement of procedures that reduce noise, such as noise instructions for each airport, published in the Aeronautical Information Publication (AIP) and by aiming for continuous descent approaches where possible **[5]**.

It is important that these same considerations are applied to the emerging field of UAM vehicles to ensure they can be implemented into urban communities without causing adverse health and living conditions. The emerging technology has the potential to provide many advances, for example in personal transport and delivery sectors, however there must be significant caution that this progress does not come at a cost to the health and well-being of citizens in areas where said technology is deployed. Research to this effect has been conducted on reducing aeroacoustic footprint through design optimization of the vehicles themselves, for example through reducing propeller/rotor noise **[6-10]**. However, research into optimising trajectories on the noise footprint of these vehicles is more limited but cover important observations about some specific configurations.

This paper aims to build on top of *Clarke M, et al* **[12]** which used SUAVE, to perform a detailed assessment and comparison of different eVTOL aircraft configurations. The aim of this paper is using this methodology in an optimisation problem with the goal to minimise the noise impacting an area around a UAM landing area.  This is done by utilising SUAVE, which is a library for Python developed by Stanford University as a tool for research and students. As part of its toolkit, there are many useful features, for example optimisation tools and mission design tools as well as the flexibility to allow for a trajectory optimisation problem to be defined. This allows for relatively accurate modelling of the noise output of an aircraft on its approach phase. The goal of this paper is to carry out such optimisation process, to provide some insight to planners of aircraft procedures, eg NATS in the UK, to allow for design of approach paths which are mindful to reduce annoyance and noise to residence surrounding air traffic hubs.

1. **METHODOLOGY**

**2.1 PROJECT WORKFLOW**

The overall work process of this project has been represented through a workflow diagram show in Figure 1. This shows the general steps taken to achieve the research scope and goals of the project. It can be seen from this, that a significant portion of the project was learning and understanding how to properly implement the SUAVE toolbox, and the design of the code.

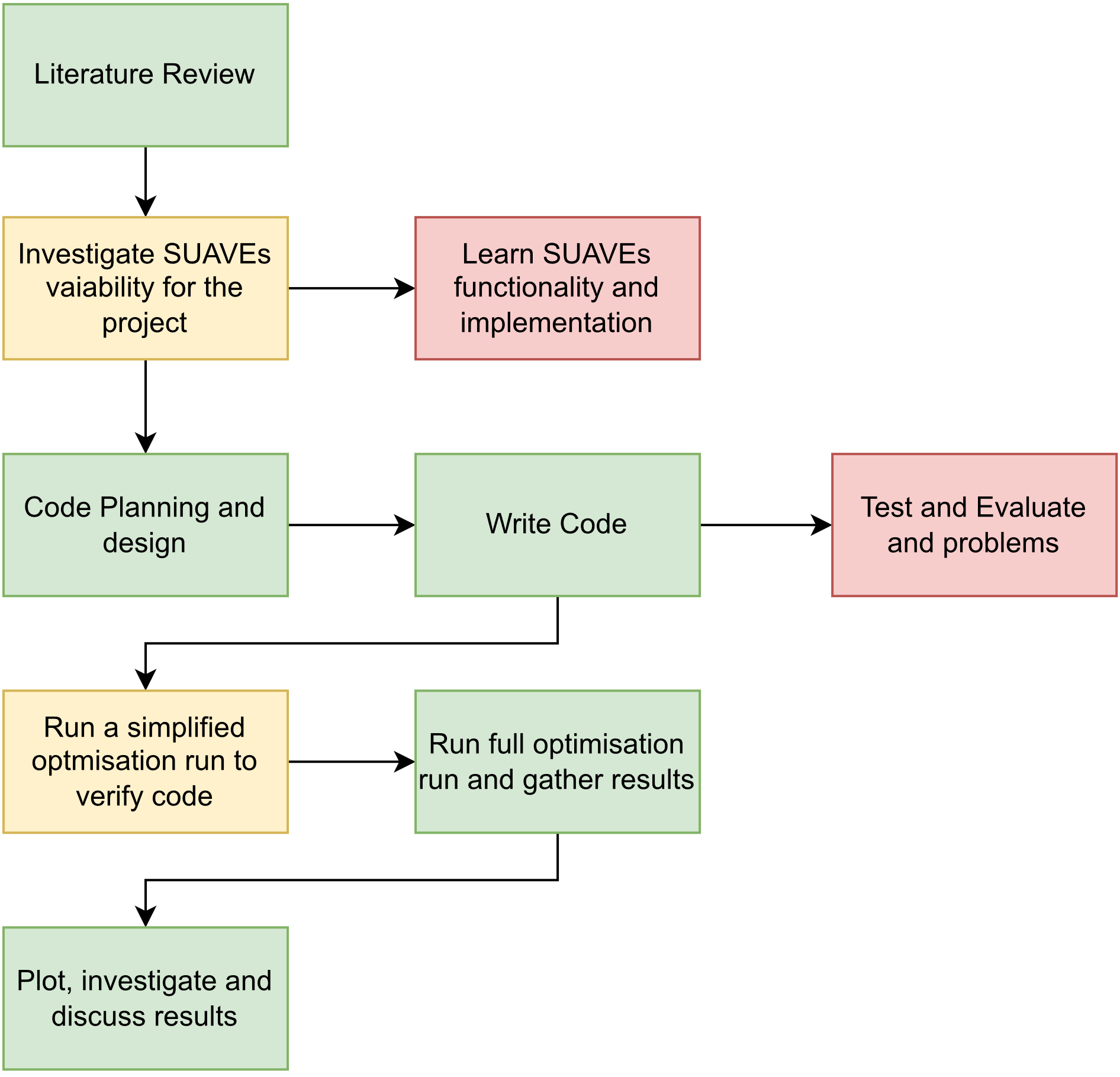


Figure 1 - Work Flow Diagram

A major slowdown in the project, was due to relatively lacking documentation regarding the implementation of SUAVEs methods. There is a doxygen available, which is an automated system for generating documentation for software source trees, however this lacks the in-depth explanations and examples you would expect from a properly documented tool/library. There also exist tutorials on the SUAVE website, however this covers certain use cases quite generally, and is by no means exhaustive. As far as the research performed for this paper, there exists no documentation on the web for how to implement a trajectory optimisation in SUAVE, nor could any examples of this being achieved be found.

**2.2 SUAVE WORKFLOW**

Every problem within SUAVE follows the same generic structure. The vehicle, analysis to be performed, and mission all need to be defined. In the case of an optimisation problem, the optimisation variables, constraints and objectives also need to be defined. SUAVE works by firstly building the vehicle, and then setting up the analysis defined for the vehicles defined, which is then attached onto the specific mission to be analysed. Therefore, each of these aspects had to be carefully considered to ensure that the results gathered were accurate. The rest of the methodology section will specifically highlight how this was achieved, and the specific code used can be seen on the Github repository.

**2.3 APPROACH AND LANDING PHASE TRAJECTORY**

**2.3.1 Defining The Trajectory**

The trajectory for the UAM on approach which would be optimised for this paper was chosen with inspiration from the traditional visual circuit. The reason for this choice, is that currently, due to the emerging nature of UAMs as a vehicle class, there is no specific approach methodology for the piloting of these vehicles and that the tradition visual circuit is in essence what forms the approaches for all aircraft types. It also allows flexibility in defining parts of the trajectory which can be included as variables for optimisation process. The standard visual circuit along with the terminology for each “leg” is shown in Figure 1.

Diagram

Description automatically generated

Figure 2 - Aerodrome Traffic Circuit [13]

This sets up the basis for how the trajectory which would be defined for the purpose of this paper is defined. For our simulation, the effect of wind and its direction is not taken into account and as such the wind direction does not play a part in the definition of the trajectory. In reality, it would be preferable that aircraft make landings into the wind as much as possible, therefore it is assumed that for this optimisation the aircraft is always flying into the wind at its final leg. The upwind, and crosswind legs are not included in the definition as these are departure legs. The trajectory is set to start at the beginning of the downwind leg.

With the basic legs contained within specifically the approach part of this circuit, namely the downwind, base and final legs, this idea was built upon to add more nuanced sections to the trajectory to capture more of the behaviour we could expect from a UAM. This was done by adding a hover leg at the end of the final leg, as well as a transitional leg where the transition from forward flight to hover would be achieved. This final defined approach path is shown in Figure 2.

Diagram

Description automatically generated

Figure 3 - Vertical slice (side view) of approach trajectory

The final trajectory consists of an initial approach (downwind) leg, 2 nautical miles long, which has a fixed altitude and speed. This is then followed by a turn onto the base leg, which is 1 nautical mile long, while maintaining the same altitude and speed. Next, there is another turn onto the final approach, and the altitude starts reducing at a set descent rate, until a set altitude. Once at this altitude, the aircraft is now transitioning to a hover, while maintaining altitude. The last leg is then a vertical landing from that altitude. This trajectory was then defined in SUAVE using the in-built methods and initial values for all altitudes and speeds were chosen based on typical values expected for the class of aircraft.

**2.3.2 Variables to Optimise**

With the trajectory established, the next step was to select appropriate parts of the trajectory to modify as part of the optimisation. These would intuitively include variables which would have a direct effect on the noise output from the vehicle, for example the forward flight speed of the vehicle, but would also include variables which would affect the physical routing of the aircraft, for example the initial altitude of the approach. It was decided to include the turn angles between the downwind, base and final legs as part of this, to allow more variation than just a simple “box” circuit as this would allow our analysis to consider more interesting approach routes, for example straight in. The full list of variables that would be changed is contained in Table 1.

|  |  |  |
| --- | --- | --- |
| Property | Legs Affected | Reasoning |
| Approach speed, from initial approach up until hover | Initial Approach, Base, Final, transition | Speed will have a big effect on noise, due to increased aerodynamic noise, and increased engine noise due to higher engine throttle. |
| Descent rate during the final leg | Final | Similar to speed, the rate of the descent will affect the aerodynamic noise, and also the throttle setting and hence the engine noise |
| Hover Height | Hover | The higher the hover, the more time at a higher throttle setting and therefore this could affect the footprint |
| Turn angle to base | Base | Effects the routing of the aircraft over the ground |
| Turn to final | Final | Effects the routing of the aircraft over the ground |

Table 1- Optimization Variables

**2.3.3 Mission Setup Within SUAVE**

The definition of this trajectory is achieved within SUAVE using a Class file, Missions.py, which sets up the mission flown by the vehicle through different segments and attaching the analyses to be preformed to said mission. This is handled within SUAVE using pre-defined segment “types” using preset options, such as constant speed and constant altitude legs, and then the settings of those legs are set within the code, for example the altitude, air speed, and length of the leg. The class file then returns the mission, with the attached analyses. Unknowns, such as the throttle settings, are solved later when the problem itself is evaluated as part of the analysis.

This missions.py file, contains all the variables that would be changed as part of the optimisation process. How this was achieved is highlighted and explained in Section x of this paper.

**2.4 AIRCRAFT FOR ANALYSIS**

**2.4.1 Background**

“Urban Air Mobility” as a term encompasses a large variety of different designs, which have different handling characteristics, number of engines, etc. The main categories currently emerging in the market are: <**INSERT HERE WITH REFERENCE 14>**. In theory, the methodology defined in this paper could be used to analyse any variation of these designs given that the design could be accurately defined within SUAVE. This could be useful in future to observe if there is a difference in how approaches should be flown for the different design types, in order to minimize noise. However, this falls outside of the scope of this paper and for our purposes a multi-rotor type vehicle is used for the analyses.

**2.4.2 Aircraft Definition Within SUAVE**

As part of the SUAVE workflow, the aircraft needs to be “built”. This then allows for proper aerodynamic and noise analysis. This is done by defining set properties utilizing the SUAVE library. This includes defining the geometry of the vehicle’s fuselage, rotor positions, number of engines, battery capacity etc. For this paper, a slightly modified version of the multirotor vehicle used in **[15]** is used. This provides a simplistic multirotor design to allow for a general analysis of a multirotor type vehicle to be performed. As mentioned previously, this could be easily changed to perform the same trajectory optimisation on another specific UAM if desired, however this is sufficient for the purposes of carrying out a general trajectory optimisation.

Practically, this is implemented through defining a class file, Vehicles.py, which defines all the relevant parameters of the vehicle, along with any configurations of that vehicle. Configurations allow variation of things such as flap angles, so that different setups of the same vehicle can be used for different parts of the mission. In our case, different rotor tilts of 0, 3 and 5 degrees were defined. These would be used for standard forward flight, transition to hover, and hover respectively.

**2.5 NOISE ANALYSIS**

**2.5.1 SUAVE Fidelity-One**

In order to accurately minimize for noise, the different relevant sources of noise needed to be simulated. For this paper, this achieved using SUAVE’s Fidelity-One noise simulation module. This uses a Frequency-Domain approach to solve the Ffowcs-Williams and Hawkings governing equation of acoustics. This consists of: Harmonic noise components, solved using Hanson’s formulation, which is noise due to the rotor blades movement through the air, Broadband noise, which includes contributions from several phenomena, such as the interaction of a blade with a preceding blades wake known as Blade Wake Interaction. SUAVE also captures noise due to Self-Noise, however currently it only models Turbulent Boundary Layer-Trailing Edge noise, which is due to the shedding of turbulence on the surfaces of the blade. Research by Clarke et al. **[15]**, however shows that omitting the other components of Self-Noise, would still lead to relatively accurate prediction of noise, unless at high speed as in these conditions, predictions of some components would be too low. Given that the speeds in this analysis will be relatively low, this should not have a major impact on the results.

**2.5.2 Implementation within SUAVE**

In terms of implementation in the code, the Fidelity-One module is straight forward to implement. Similarly, to how the aircraft and trajectory were setup, a Class file, analyses.py is used to define the analyses to be used. This is done by passing through the vehicle and configurations of said vehicle and defining which analyses withing SUAVE to perform. In order to carry out an accurate simulation of the vehicles flight through this approach phase, the vehicles weights had to be analysed. This was done through the SUAVE weights\_eVTOL method. This then allowed analysis of the aerodynamics through the aerodynamic fidelity zero method. Then, the noise was analysed using the noise fidelity one method. This method works by setting up a grid of ground microphones, around the starting location of the aircraft. The number of these microphones, as well as the resolution of the microphone placement is defined. An example of such a setup is shown in Figure 3, which shows a setup grid of microphones with the routing of the aircraft from a bird’s eye view.

In figure 3 we can see the route represented by a set number of points. These points are “computational points” which is defined as part of the mission setup and defines how many points are used to represent each segment of the mission. A higher number of points leads to a higher resolution with the analysis defined being preformed at more points along each segment. By default, SUAVE sets a control point value of 16, however according to the SUAVE developers, highly accurate results of even 0.1% error can be achieved with just 4 control points **[16]**. Given this, 4 control points were set.

**2.5.3 Computational Issues**

As mentioned above, the number of microphones and resolution of the placement is defined for the noise analysis. During coding, it was found that these settings had a major impact on memory use of the simulation. This meant that increasing the number of microphones even quite a small amount would not be possible under the current setup without running the computer running the simulation our of memory. The computer in question has 32Gb of RAM. As a result of this, an alternative way to run the simulation was designed as to optimise memory use and allow for a more accurate (higher resolution) of microphones.

The solution consisted of running the same simulation several times, however only considering a small segment of the overall grid per run of the simulation. This is depicted graphically in figure x. This was initially developed using a quadrant system, where the overall grid was split into four, and then the simulation was run four times with the results of each grid then being assembled at the end for the overall noise capture by all the microphones. Once this proved successful with testing, the same methodology was used to split the grid into nine to further increase the resolution of the grid of microphones. Further in addition to splitting the grid, the grid was also centred about the end of the trajectory, the landing point, rather than the start point which is how SUAVE sets the problem up by default. This made it easier to ensure that the whole trajectory remained within the grid.

This functioned well, with the trade-off being that the simulation would now take longer to run, due to the increased times the simulation needed to be ran. This would not be an issue for single run cases, however in the case of optimisation, where the simulation would need to be run many times in order to find a minimum, this provided a significant issue for retrieving results within a sensible timeframe. This is discussed further in the results section of this paper.

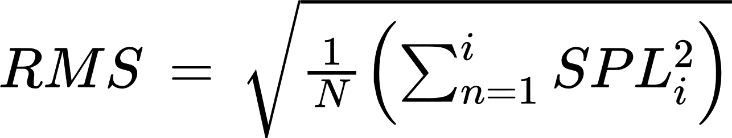
**2.5.4 Quantifying Overall Noise to Optimise**

As part of any optimisation problem, there needs to be an objective function to minimise or maximise. In this case, the general goal of the problem was to reduce overall noise disturbance for the area surrounding a landing site for UAM vehicles. To complete the optimisation problem, there needed to be a way to represent this general goal as one numerical value that the optimiser would attempt to minimise.

Firstly, it was decided to represent the sound output recorded by each microphone in the grid using the A-weighted system. This is a method for representing sound pressure levels in a way that accounts for relative loudness as heard by the human ear. It works by adjusting sound pressure levels to account for the sensitivity of the human ear at certain frequencies. This makes it suitable for this task, as our goal is to reduce the impact on people living around a landing site.

Indeed, the A-weighted system is used by regulators and stakeholders in the industry for measuring and representing the sound impact of aircraft on the surrounding population. For example, NATS, the UK air traffic control provider, utilises the A-weighted system as part of its system for measuring aircraft noise around airports. This metric is used to determine things such as “Leq16h” which is a measure of noise events over a 16-hour period, or SEL (sound exposure level) which is used to determine the likelihood of sleep disturbance as a result of aircraft noise events **[17]**.

SUAVEs Fidelity-One method already outputs the A-weighted SPL for each microphone in the grid, for each control point as part of its simulation. Once we have each microphones A-weighted SPL for each timestep of the simulation, the last step is to represent these results in a way representative of the noise experienced by people living near this landing site. This is achieved by taking the RMS of the recorded A-weighted SPLs of each microphone for each time step of the simulation. RMS is used as this provides a good measure of the overall sound energy emitted by the aircraft over the duration of the flight and provides a good representation of an “average” that the entire population within the set grid experiences. The mathematical formula used in shown in equation 1.



Equation 1- RMS formula

Another method considered was to use the average of the maximum recorded A-weighted SPL for each microphone, however this was decided against as this would not capture information regarding the duration or frequency content of the noise. This is a useful metric however to observe where the loudest points during a set trajectory occur, and therefore was carried out for the “optimised” trajectory.

The implementation of this calculation was handled as part of a post process following the completion of a run of the simulation. The results for each quadrant is stored in a “pickle” file, and then the SPL data for each microphone, at each control point is extracted from these and stored into arrays. This data is then passed into a function which calculates the RMS for that run of the simulation, and then this is returned to the optimiser as the objective variable.

**2.6 OPTIMISATION**

**2.6.1 Optimisation Within SUAVE**

Optimisation in SUAVE is handled using a class called Nexus. This stores all information regarding the optimisation, for example input variables, constraints and objective information as well as the information regarding the analysis to be carried out which is defined as described in earlier chapters of this paper. The Nexus links together all subfunctions of the problem and allows them to interact and communicate using aliases. SUAVE then packages the problem defined and described by the nexus, for it to be handled by a third-party optimiser, for example SciPy (a scientific python library) optimisers. This provided us with a few options for which type of optimiser we could select for our problem.

**2.6.2 Gradient Based Optimisers**

The first time of optimisers considered were gradient optimisers. These function by using the derivatives of a “loss function” to guide the direction of the optimisation in order to find a minima. Due to the need to calculate the gradient for the entire data set in one update, these tend to be slow and computationally expensive. [**18-19]**. Further, gradient optimisers require the objective function to be relatively smooth and continuous in order for the derivatives to be calculated or at least approximated. For our problem, given that the noise output is measured at certain control points during the simulation and is not distributed continuously over a period of time, the objective function is likely not to be very smooth, rendering it unsuitable for these methods.

Within SUAVE using SciPy, the problem can be attempted using a SLSQP (Sequential Least Squares Programming) optimisation. As expected, when this was attempted, the problem failed to reach the global minimum. This was expected given the objective function not being smooth.

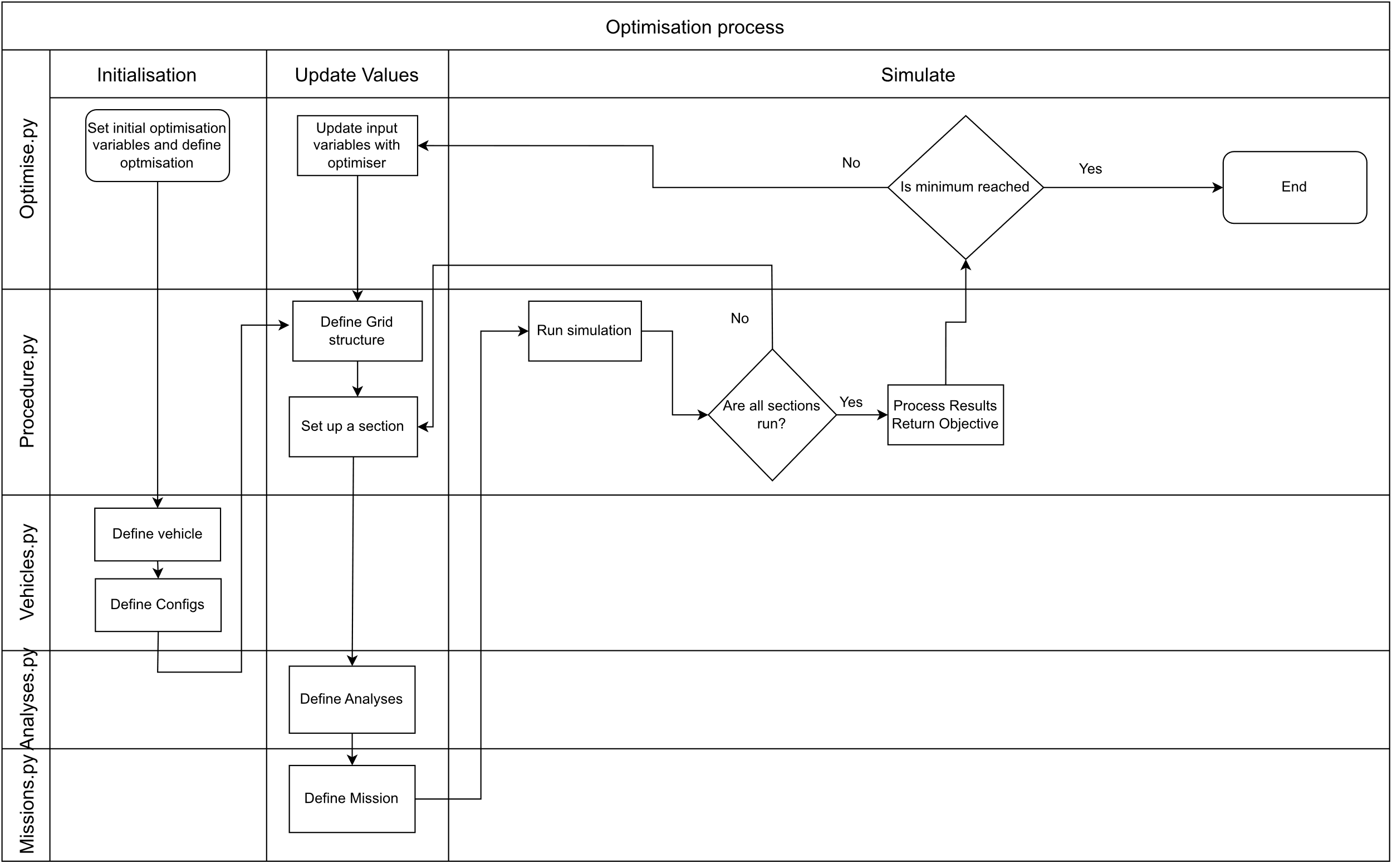
**2.6.3 Particle Swarm Optimisation**

The other type of optimisation investigated was PSO. This is a randomized search technique inspired by social interaction principles. It functions by updating a set of candidate solutions, known as a swarm at each step. For each step, the candidate solutions tend to move closer to each other [**18**]. It is a heuristic solution, and often the solution found is quite close to the global optimum [**20**].

Unlike gradient optimisers, gradient information is not needed which makes it useful for solving a problem such as ours which has a non-smooth objective function. Further, PSO tends to have a fast convergence which is beneficial as noise simulations can take a long time, and as previously mentioned one potential issue for our optimisation is the time to completion being too long. Due to these factors, the decision was made to use SciPys PSO algorithm for the optimiser in this problem.

**2.6.4 Code Implementation**

Figure 4 shows a cross functional flow chart representing the general code structure of the optimisation problem. It can be shown that the only variables changed as part of the optimisation process occur within the analyses and the mission. As previously explained, the only variables changing as part of the optimisation process are to do with the actual trajectory and are found in the missions class. However, the reason that the analyses needs to be updated at each optimiser cycle is because currently within SUAVE, there is no way to only update the mission without modifying the analysis.

 Figure 4 - Cross Functional Flow Chart of code

This introduces some coding inefficiency when performing a trajectory optimisation such as the one performed in this paper, as unnecessary computational steps are carried out. This also led to a messy work around in order to update the mission values in order to preform the optimisation. Firstly, the optimiser inputs had to be unpacked from the Nexus, and then these values had to be assigned to a new mission and analysis which had to be created. This was then evaluated to find the noise results, and the results from this then had to be attached into the optimisation nexus.

**3. Results**

**REFERENCES**

[1] The Vertical Flight Society. (2022, July 11). *eVTOL Aircraft Directory*. Https://Evtol.News/Aircraft.

[2] Basner, M., Babisch, W., Davis, A., Brink, M., Clark, C., Janssen, S., & Stansfeld, S. (2014). Auditory and non-auditory effects of noise on health. *The Lancet*, *383*(9925), 1325–1332. https://doi.org/10.1016/S0140-6736(13)61613-X

[3] Sadeghian, M., & Gorji Bandpy, M. (2020). *Technologies for Aircraft Noise Reduction: A Review*. https://doi.org/10.35248/2168

[4] Heathrow Airport. (2022, July 11). *Heathrow runway alteration programme*. [Https://Www.Heathrow.Com/Company/Local-Community/Noise/Operations/Runway-Alternation](https://www.heathrow.com/Company/Local-Community/Noise/Operations/Runway-Alternation).

[5] *Noise and emissions* (2021) *NATS*. Available at: https://www.nats.aero/environment/noise-and-emissions/ (Accessed: April 9, 2023).

[6] Marinus, B., Roger, M., & van den Braembussche, R. (2010, June 7). Aeroacoustic and Aerodynamic Optimization of Aircraft Propeller Blades. *16th AIAA/CEAS Aeroacoustics Conference*. https://doi.org/10.2514/6.2010-3850

[7] Pagano, A., Federico, L., Barbarino, M., Guida, F., & Aversano, M. (2008, September 10). Multi-Objective Aeroacoustic Optimization of an Aircraft Propeller. *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. https://doi.org/10.2514/6.2008-6059

[8]Higgins, R. J., Barakos, G. N., Shahpar, S., & Tristanto, I. (2021). A computational fluid dynamic acoustic investigation of a tiltwing eVTOL concept aircraft. *Aerospace Science and Technology*, *111*, 106571. <https://doi.org/10.1016/j.ast.2021.106571>

[9] içke, ramiz ö., Baysal, O., Lopes, L. v., Zhou, B. Y., Diskin, B., & Moy, A. (2020, June 15). Toward Adjoint-Based Aeroacoustic Optimization for Propeller and Rotorcraft Applications. *AIAA AVIATION 2020 FORUM*. <https://doi.org/10.2514/6.2020-3140>

[10] Huang, Z., Yao, H., Sjögren, O., Lundbladh, A., & Davidson, L. (2020). Aeroacoustic analysis of aerodynamically optimized joined-blade propeller for future electric aircraft at cruise and take-off. *Aerospace Science and Technology*, *107*, 106336. <https://doi.org/10.1016/j.ast.2020.106336>

[11] Casalino, D., van der Velden, W. C., & Romani, G. (2019, January 7). Community Noise of Urban Air Transportation Vehicles. *AIAA Scitech 2019 Forum*. <https://doi.org/10.2514/6.2019-1834>

[12] Clarke, M. A., & Alonso, J. (2021, August 2). Evaluating the Performance and Acoustic Footprint of Aircraft for Regional and Urban Air Mobility. *AIAA AVIATION 2021 FORUM*. <https://doi.org/10.2514/6.2021-3205>

[13] *Aerodrome Traffic Circuit* *SKYbrary Aviation Safety*. Available at: https://skybrary.aero/articles/aerodrome-traffic-circuit (Accessed: April 7, 2023).

[14] European paper I forgot

[15] Clarke, M.A. *et al.* *Towards a regional and Urban Air Mobility Future: The development of computational approaches for quantifying trade-offs in Electric Aircraft Design*. thesis. [16] *SUAVE FAQ* (no date) *Frequently asked questions*. Available at: https://suave.stanford.edu/documentation/faq.html (Accessed: April 9, 2023).

[17] *Measuring noise* (2020) *NATS*. NATS. Available at: https://www.nats.aero/environment/noise-and-emissions/measuring-noise/ (Accessed: April 7, 2023).

[18] Chong, E.K.P. (2023) *Introduction to optimization*. S.l., New Jersey: JOHN WILEY.

[19] Goyal, C. (2021) *Complete guide to gradient-based optimizers in deep learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-gradient-based-optimizers/ (Accessed: April 7, 2023).

[20] Sanyal, S. (2021) *An introduction to particle swarm optimization (PSO) algorithm*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/10/an-introduction-to-particle-swarm-optimization-algorithm/ (Accessed: April 7, 2023).