5-13-2021

Validation and Optimization of Ptera Software: An Open-Source Unsteady Simulator for Flapping Wings

Cameron Urban
Washington University in St. Louis

Ramesh K. Agarwal
Washington University in St. Louis

E37 MEMS 400 09: Independent Study Report

Validation and Optimization of Ptera Software: An Open-Source Unsteady Simulator for Flapping Wings

Cameron Urban[*] and Ramesh K. Agarwal[†]
Washington University in St. Louis, St. Louis, MO, 63130, USA

Scientists have long used the unsteady vortex lattice method (UVLM) to simulate flapping-wing flight. However, there are few open-source UVLM solvers designed for research in this field. The newly released Ptera Software is, to the best of the authors' knowledge, the only open-source UVLM solver for flapping wings that is in active development. This report documents the next steps in Ptera Software's progress: the validation of its results and the optimization of its performance. Comparing Ptera Software's output to high-fidelity experimental data of the pressures on a flapping-wing robot shows that the simulated results predict the trends and magnitudes of the net lift over time with reasonable accuracy. Also, by using several computational methods, the researchers optimized the source code such that the solver's latest iteration is over three times faster than previous versions. The present results demonstrate that Ptera Software correctly implements the UVLM and can simulate flapping-wing flight with reasonable accuracy under this method's assumptions. Additionally, it is now fast enough for use as an iterative tool in the design of novel flapping-wing micro aerial vehicles (FWMAVs).

I. Introduction

ANIMALS are better at flying than any human invention. Although animals may not fly as fast as a jetliner or carry as much weight as a cargo plane, they fly with higher maneuverability, adaptability, and grace than anything aerospace engineers have invented before or after the Wright brothers first took flight at Kitty Hawk. In the words of aerodynamicists McMasters and Henderson, "…humans fly commercially or recreationally, but animals fly professionally" [1].

These advantages indicate that natural flight is a rich area for research that could improve human flying machines. Unfortunately, analyzing how animals fly is extremely difficult. After first becoming interested in natural flight, many researchers assume that the simplistic methods used to analyze conventional aircraft will be applicable. They quickly realize that the high performance of flying animals is matched by the complexity of their aerodynamics. Thankfully, today's scientists can begin their research on the foundations laid by other aerodynamicists who have worked tirelessly to unlock the secrets of flapping-wing flight.

One of the tools discovered by these researchers is the unsteady vortex lattice method (UVLM). This tool, described in more mathematical detail in the literature [2] than in this paper, is a well-established algorithm for analyzing unsteady aerodynamics. Past work has also used it to simulate flapping wings. As a final advantage over other methods, the UVLM also strikes a favorable balance between accuracy and the computational time required to run the simulations. For these reasons, many researchers select it as their tool of choice in exploring natural flight.

Unfortunately, they quickly stumble into a roadblock. Although journals have published many papers which use the UVLM to analyze flapping wings, it is difficult to find any commercial or open-source UVLM programs capable of doing so without significant modifications to their source code. The authors of this paper decided to fix this problem.

After a few months of work, they released Ptera Software, one of the few open-source UVLM solvers designed to analyze flapping wings and the only such solver currently being maintained, to the best of the authors' knowledge. Ptera Software is free to use so as to encourage further research into the field of animal aerodynamics and the collective

[*] Undergraduate Student, Dept. of Mechanical Engineering and Materials Science, Student Member AIAA
[†] William Palm Professor of Engineering, Dept. of Mechanical Engineering and Materials Science, Fellow AIAA

use and improvement of its source code. For more information on Ptera Software, or to download the source code, visit the project page on GitHub [3].

This paper documents the next steps in Ptera Software's development: validating its results against experimental results and increasing its computational performance. To validate the solver, the authors first reviewed the literature to find appropriate experimental data. Next, they used Ptera Software to emulate the experimental setup that produced this data. Finally, they ran Ptera Software's UVLM solver on this emulation and compared its simulated findings against those from the original experiment.

Increasing the solver's performance means decreasing the solve time. To accomplish this, the researchers created a simple simulation in Ptera Software and benchmarked how long it took to solve using code timing best practices. Then, they modified the code to run specific computationally expensive calculations only when necessary, switched to a more efficient method of object deep-copying, and implemented just-in-time (JIT) compilation to expensive calculations

II. Validation: Literature Review

A. Analyzing Flapping Wings with the UVLM

The literature has demonstrated that the UVLM can accurately predict the aerodynamic forces and moments exerted on flapping wings by comparing their results to data gained experimentally. For example, researchers Fritz and Long validated plunging and pitching UVLM simulations against experimental data and analytical solutions [4]. Lambert et al. corroborated the results of these experiments by showing that the UVLM accurately modeled the forces on a flapping micro-air vehicle (MAV) as long as the flow around its wings remained attached [5].

The UVLM has also been used to simulate the flapping-wing flight of real animals. For example, Gardiner et al. used the UVLM to study the flight of barnacle geese [6]. However, the researchers did not provide experimental data that validated their simulated results. Some authors have even used a modified form of the UVLM to simulate the flight of insects [7].

Almost every paper in this field uses a home-grown code developed by each researcher's laboratory. There is no widely accepted, easily accessible framework for conducting low-fidelity flapping-wing research. Therefore, once validated, Ptera Software could aggregate each institution's disparate advances and eliminate the barrier to entry of developing a solver for those newly interested in this field.

B. Experimental Data for Validation

To validate the theoretical results produced by Ptera Software, the authors of this work needed experimental data to use as a benchmark. They found this data in "Experimental and Analytical Pressure Characterization of a Rigid Flapping Wing for Ornithopter Development" by Yeo et al., which details the analysis of a robotic ornithopter, an umbrella term for any flapping-wing aircraft [8].

Yeo et al. mounted their ornithopter to a test stand within a wind tunnel, as shown in the reprinted Fig. 1. While flapping in the wind tunnel, the authors gathered aerodynamic data from the robot via an array of nine differential pressure sensors mounted on one of the wing halves. The sensors measured the difference in pressure between the wing's lower and upper surfaces.

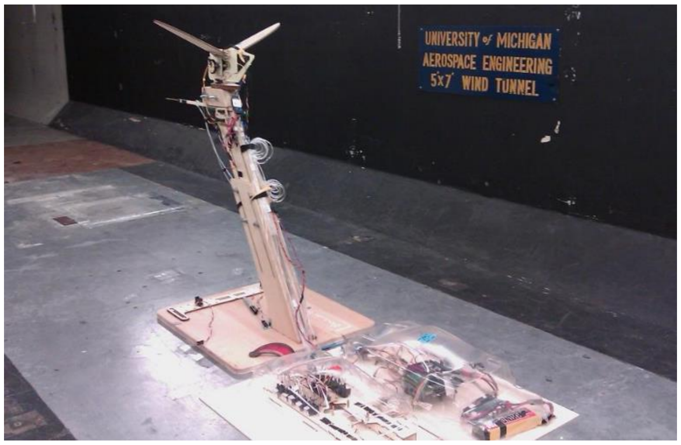


Figure 1. A robotic flapping-wing testbed in a wind tunnel [8].

The authors chose the data from "Experimental and Analytical Pressure Characterization of a Rigid Flapping Wing for Ornithopter Development," here on referred to as Yeo et al., 2011, to validate Ptera Software for three reasons. (1) The data provided by this paper is well suited for comparison with the data from the UVLM. By using pressure sensors instead of a more traditional force sensor, Yeo et al. bypassed the experimental confounding factor of separating aerodynamic and inertial loads. (2) Yeo et al. analyzed forward flapping-wing flight along with hovering flapping-wing flight. The inclusion of forward flapping-wing flight data is critical because the traditional UVLM is not the proper tool for analyzing hovering flapping-wing flight due to the prevalence of separated flow under hovering conditions. Flow separation is a viscous phenomenon where a fluid ceases to follow the curvature of a surface. A real-world example of this is an aircraft stalling at too steep an angle of attack for its airspeed. As the UVLM is an inviscid solver, it cannot account for separation. However, flow separation is significantly less prevalent during forward flapping-wing flight because of the lower effective angle of attack. Therefore, this study will validate the forward flapping-wing flight case. (3) The University of Michigan, where this research took place, is a well-respected institution for the study of aerodynamics, which lends credence to the quality of the data collected.

III. Validation: Methodology

After selecting the experimental data to use for the validation, the authors needed to process it and model the geometry of Yeo et al.'s robotic ornithopter in Ptera Software. Additionally, they had to check the results for convergence with respect to temporal and spatial discretization. Fig. 2 displays the steps of this procedure.

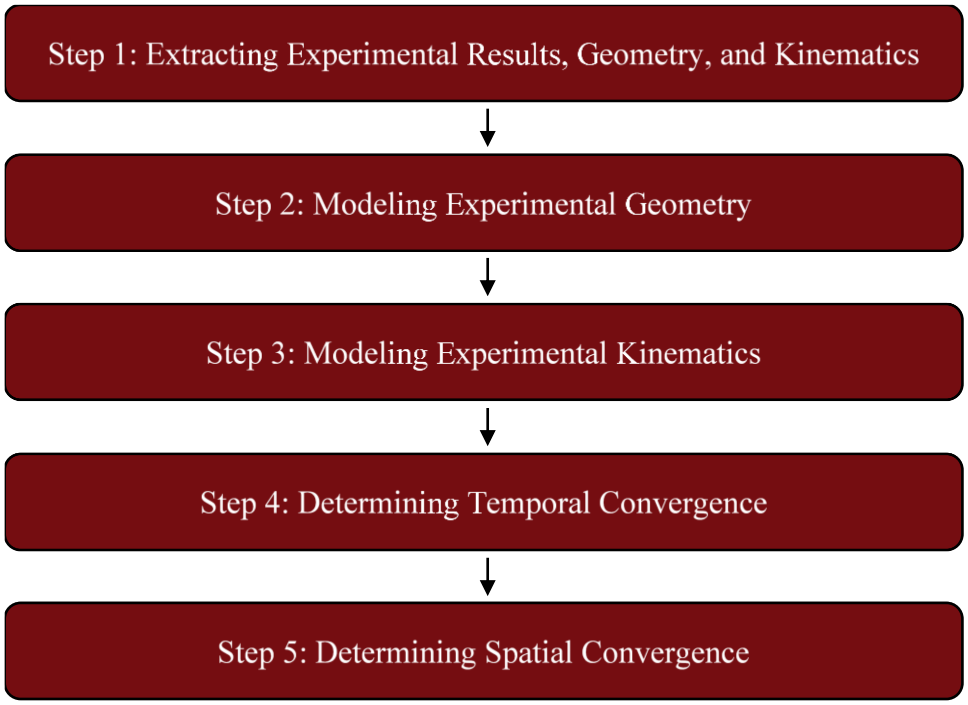

Step 1: Extracting Experimental Results, Geometry, and Kinematics
Step 2: Modeling Experimental Geometry
Step 3: Modeling Experimental Kinematics
Step 4: Determining Temporal Convergence
Step 5: Determining Spatial Convergence

Figure 2. The process of running the validation study.

A. Extracting Experimental Results, Geometry, and Kinematics

The researchers chose to validate against the experiment referred to in Yeo et al., 2011 as case C. The data from this run is shown in Fig. 3, reprinted from Yeo et al., 2011, which contains five subplots. The upper-left subplot is a sketch of the nine pressure sensors' approximate positions on the instrumented wing. The lower-left subplot shows Yeo et al.'s measured versus predicted wing flap angle during the test. Each of the right three subplots displays the cycle-averaged pressure data for three of the nine sensors. The upper-right, middle-right, and lower-right subplots show the data for the "blue," "orange," and "green" sensors, respectively. Readers can use the upper-left subplot to identify each sensor's color and number designations.

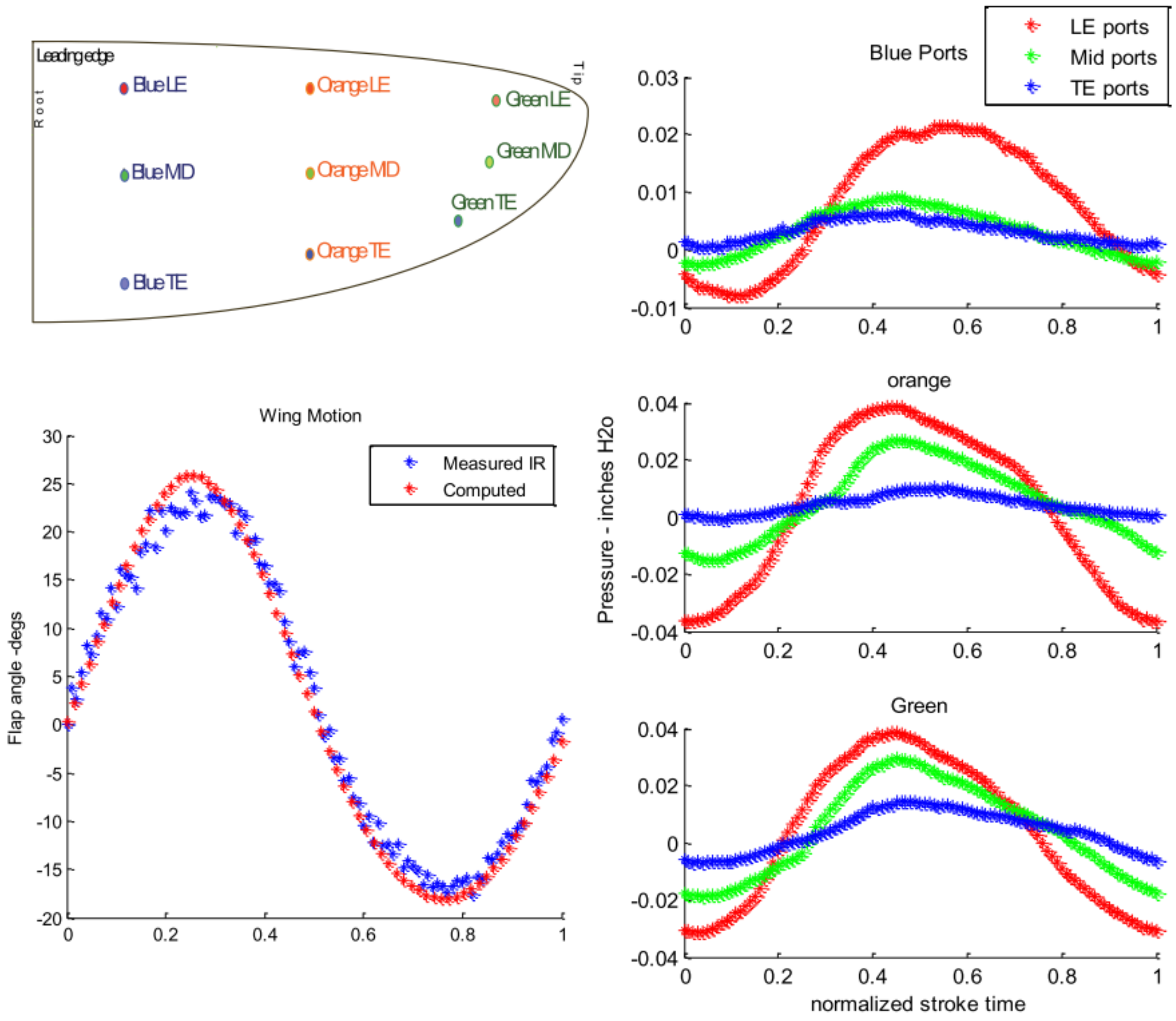


Figure 3. Data from a flapping-wing robot at the flight condition of interest [8]**.**

Yeo et al. did not provide the raw data in tabulated form, so the authors used the popular data analysis tool WebPlotDigitizer to extract approximated raw pressure data from screenshots of Fig. 3's pressure plots that they uploaded to the tool's website [9]. After appropriately scaling the axes, the program automatically detected the positions of the graph's data points and compiled them into a downloadable comma-separated values (CSV) file.

B. Modeling Experimental Geometry

The second step was to model the geometry of Yeo et al.'s robotic wings in Ptera Software. First, the authors saved a plot from Yeo et al., 2011 of the wing's shape and sensor locations, reprinted here as Fig. 4. Then, they uploaded the plot to WebPlotDigitizer, which provided interpolated coordinates of the wing planform's curve, shown in Fig. 4 as a solid black line.

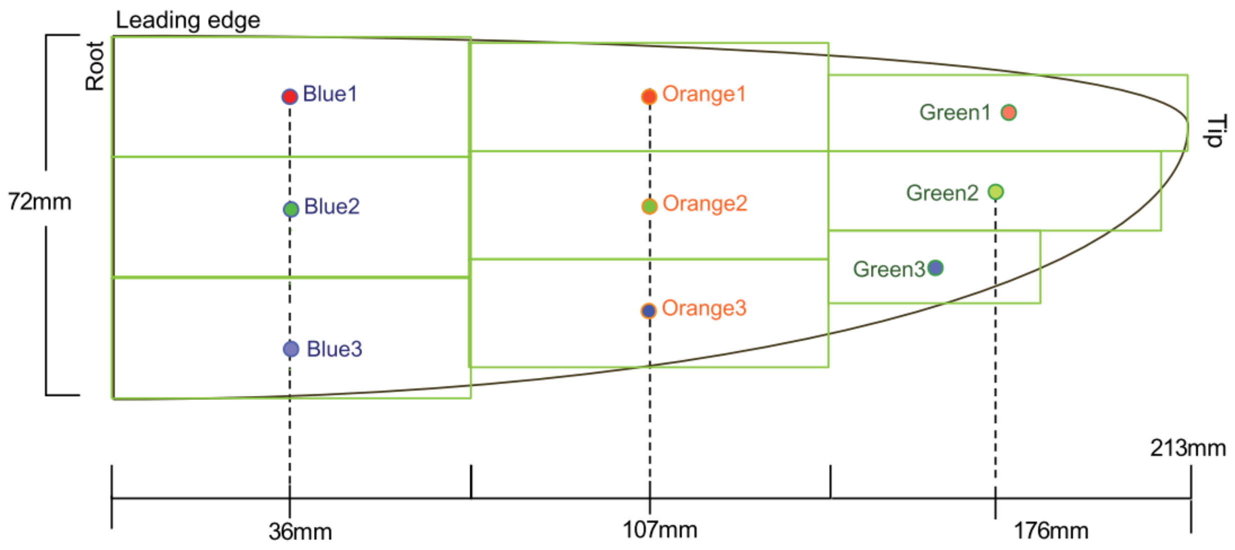


Figure 4. The geometry of the robotic half-wing [8].

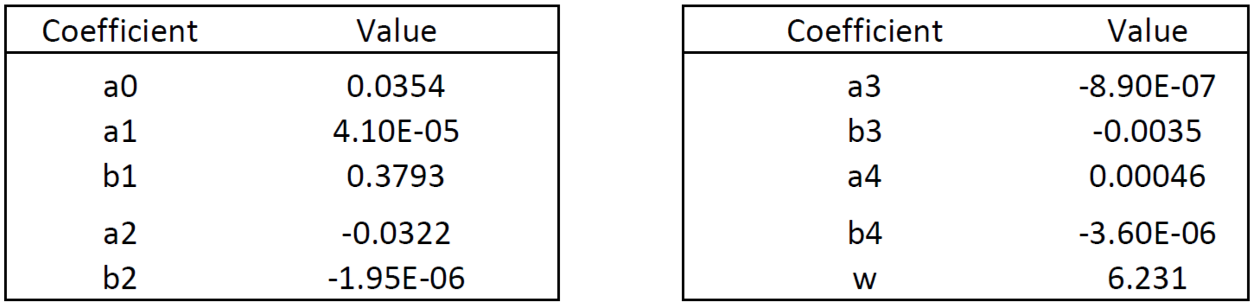
The authors were then able to discretize the wing in the spanwise direction into any number of uniformly-spaced wing cross sections. As shown in the reprinted Fig. 4, the experimental wing planform extends to a smooth tip. They did not design Ptera Software to analyze wings with a tip chord length of zero. So, to avoid the possibility of numerical errors, they sliced off the outermost five millimeters of the wingtip so that the last wing cross section had a non-zero chord length.

From Fig. 4, the authors also extracted the approximate areas upon which each pressure acts. The figure represents these areas as the green rectangles around each of the ports. Using this data, the normal force on each rectangle is simply the pressure at its port multiplied by its area. The lift on each panel is the vertical component of this force at that particular time step, which they calculated by multiplying the normal force by the cosine of the current flapping angle. Finally, the total lift on the robot is the sum of the lift forces on each rectangle multiplied by two (because there are two wing halves, and only one has pressure sensors).

Although necessary, the data extraction step introduces potential errors into the validation because the results are approximate. Future work would allow for collaboration with Yeo et al. to rerun this validation study with the exact experimental data.

C. Modeling Experimental Kinematics

The third step is to model the experimental kinematics. In a separate paper, the authors of Yeo et al., 2011 analyzed the same robot's flapping mechanism to characterize the flapping angle as a function of time [10]. They found that the flapping was experimentally repeatable and well modeled by a fourth-order Fourier series. Fig. 5, which the authors have reprinted here, shows the Fourier coefficients. The lower-left subplot in Fig. 3 displays a comparison between the experimental and modeled kinematics of the flapping.



Coefficient	Value
a0	0.0354
a1	4.10E-05
b1	0.3793
a2	-0.0322
b2	-1.95E-06

Coefficient	Value
a3	-8.90E-07
b3	-0.0035
a4	0.00046
b4	-3.60E-06
w	6.231

Figure 5. The Fourier series coefficients for the flapping angle equation [10].

Using these Fourier coefficients and the emulated test's flapping frequency, the authors wrote a Python function that Ptera Software used to simulate the flapping kinematics.

D. Determining Temporal Convergence

The UVLM is a time-stepping simulation, and the fourth step in setting up the study is picking how much time to simulate. During each time step, the wing sheds a new row of ring vortices into its wake. The panels on the wing experience the effect of the panels in the wake during the next time step. If the simulation were allowed to run for an infinitely long time, there would be an infinitely long sheet of wake panels stretching behind the wing. At this point, the pressure vs. time graphs for each subsequent flapping cycle would be identical. We would say that the system has reached a quasi-steady state (quasi because the pressure distributions are still varying over each flap cycle, but this variation is constant from one flap cycle to another).

However, simulating for an infinitely long time is impractical. Thankfully, as the wake ring vortices get further away from the wing, their effects decrease rapidly. Therefore, there should be some number of flap cycles, after which additional flap cycles produce only a negligible difference in the pressure vs. time functions.

The researchers found this threshold to be three flap cycles. They determined this after running several test simulations for one, two, three, and four flap cycles with different spatial discretizations and observing that there was qualitatively no difference in the results for two and three cycles.

E. Determining Spatial Convergence

The final step in running the experiment in Ptera Software is to determine how to arrange the wing panels. Ptera Software's implementation of the UVLM first turns a wing into a 2D surface and then discretizes it into an array of rectilinear panels. Generally, the more panels used, the more accurate the result. Additionally, the authors weighed several other considerations before choosing the number of panels and determining their shape.

1) Spacing: In many steady vortex solvers, the panels are smaller and more tightly grouped near the wing's leading edge, trailing edge, root, and tip. Researchers use this spacing scheme because the pressure gradient is usually higher at these locations. However, most unsteady solvers use a uniform spacing, at least in the chordwise direction, so that the wake panels are roughly the same area as wing panels that shed them. This choice satisfies the wake transport equation, one of the governing laws of the UVLM [11]. Although not required by the wake transport equation, this study used uniform spacing for the spanwise panels to reduce the code required to run these cases. Future work should revisit this decision.
2) Aspect Ratio: The panel's aspect ratio is its average chordwise dimension divided by its average spanwise dimension. This value should be kept near one to reduce computational error [12].
3) Computation Time: The computation time increases dramatically with the number of panels, specifically with the number of chordwise panels. Therefore, researchers should use the smallest number of panels required to achieve convergence.

As per the preceding principles, the authors decided to space the panels uniformly, keep their aspect ratios as close to one as possible, and design a small study increasing the number of chordwise panels from a bare minimum amount until they saw qualitative convergence. Convergence emerged after reaching five chordwise panels with 18 spanwise panels.

IV. Validation: Results

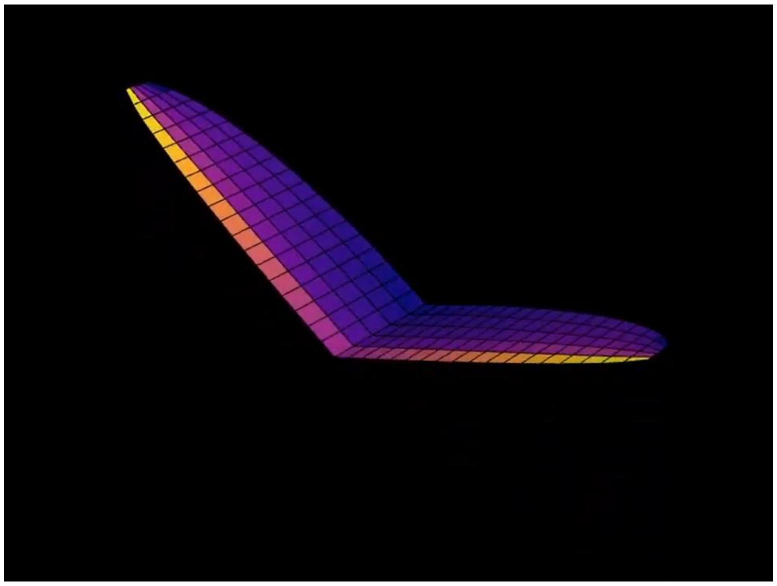


Figure 6. One frame from a close-up animation of the simulated experiment's wings. The wing panel colors represent the magnitudes of the pressures. Hot colors indicate higher pressures on the lower surfaces and vice versa. Click here to see the entire animation.

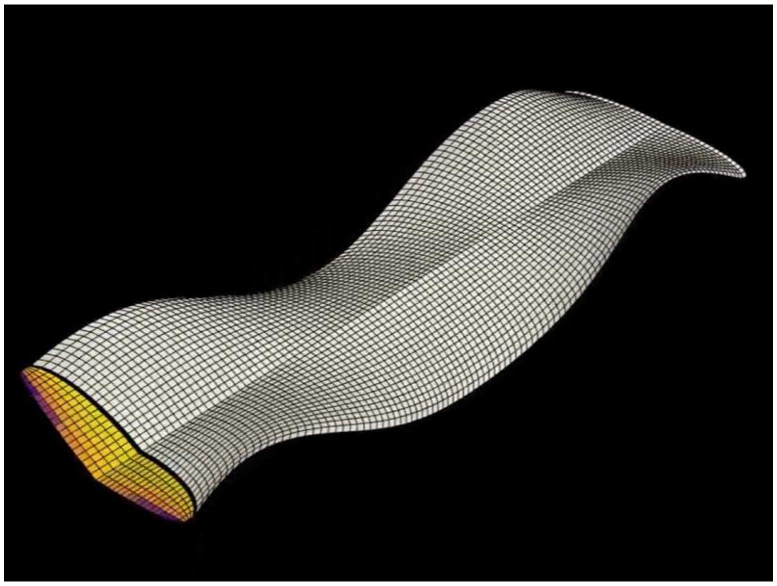


Figure 7. One frame from an animation of the simulated experiment's wings and wake. The wing panel colors represent the magnitudes of the pressures. Hot colors indicate higher pressures on the lower surfaces and vice versa. The white vortex rings represent the wake shed from the trailing edge. Click here to see the entire animation.

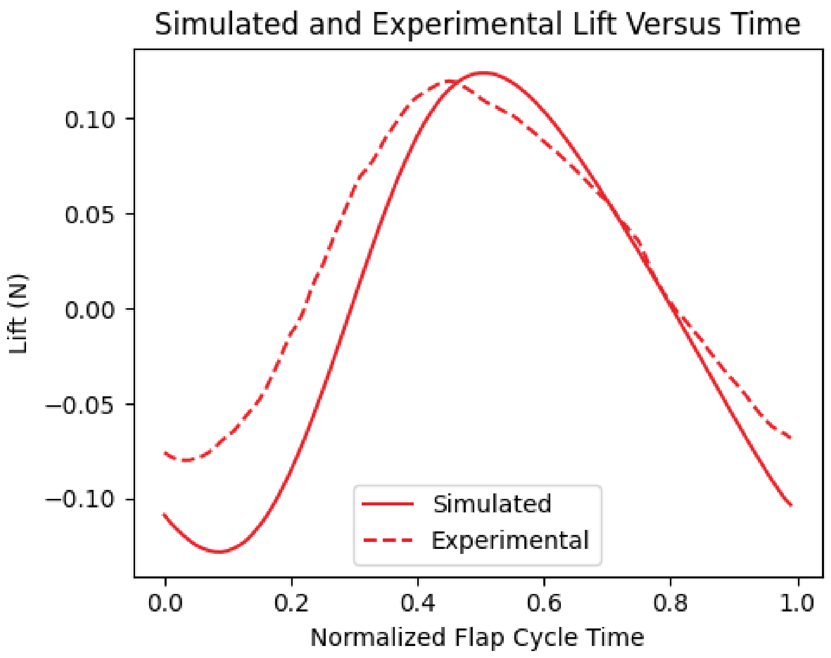


Figure 8. The net simulated and experimental lift across the entire wing over a flap cycle normalized from zero to one. The simulated lift values comprise those calculated during the final flap.

V. Validation: Discussion

A. Simulation Results

The descriptions of figures 6 and 7 contain embedded links to the videos produced by the simulation. These visuals help determine quickly if something obvious is wrong with the solver. For example, the videos show that the wing is the correct shape and is flapping as expected. Additionally, the pressures on two wing halves look symmetric throughout, which the authors expected given that the sides of the robot are identical and experienced identical conditions.

Figure 8 shows that the fully converged simulation models the experimental lift trends and magnitudes relatively well. The simulated lift had a mean absolute error (MAE) of 0.0291 N over the flap cycle with respect to the experimental lift. The root mean square (RMS) of the experimental lift values is 0.0717 N.

This paper uses MAE as an accuracy metric instead of mean absolute percent error (MAPE) because both the experimental and simulated lift trends oscillate around 0 N. A percent error method is ill-equipped to deal with values near zero, so I chose this error-based approach instead.

B. Inaccuracies

The two noticeable inaccuracies in the simulated lift trend are a rightward phase shift and an overestimation of the lift force magnitude in the first and final quarter of the flap cycle. The phase shift is more difficult to explain than the force magnitude error, despite being less significant. Previous papers have discussed how procedures for averaging experimental data over a cycle could affect the observed phase [13]. However, without access to the raw experimental data, it is impossible to investigate this further.

The negative lift magnitude overestimation during the first and last quarter of the flapping cycle is likely caused by flow separation. Flow separation, an aerodynamic phenomenon where the streamlines of a fluid detach from a surface, is created by viscosity. As the UVLM assumes the fluid is inviscid, it cannot predict this behavior. As shown in Fig. 6 and Fig. 7, the wing starts its cycle halfway through the upstroke. Therefore, the separated flow conditions correspond to the wing's upstroke. During the upstroke, the wing experiences a highly negative effective angle of attack, which likely caused the flow on the wings' lower surfaces to separate due to a strong adverse pressure gradient.

Additionally, inconsistencies in Yeo et al., 2011 may have contributed to inaccuracies in the results. In the report, the authors wrote on two separate occasions that the free stream velocity for the case I simulated was 2.9 m/s. However, in a later section, this was changed to 2.8 m/s. The authors of this report chose to use the more frequently cited value of 2.9 m/s. Future work should include contacting the original authors to resolve this inconsistency.

The literature documents work to modify the UVLM to correct for flow separation effects. The most promising attempt modified the UVLM to begin shedding vortex rings off the wing's leading edge after a certain effective angle of attack was reached [14]. Future work should implement this method in Ptera Software and analyze its results.

C. Limitations

Despite this study's promising results, researchers should use Ptera Software with an understanding of when its assumptions lose validity. For example, the UVLM assumes inviscid, irrotational, and incompressible flow. Although no actual flow exactly satisfies these requirements, in practice, they imply that Ptera Software will only produce accurate results for objects operating at relatively high Reynolds numbers, with attached flow, and whose airspeed is much lower than the speed of sound.

Additionally, this software is in its early stages of development. Engineers should not use it to make safety-critical design choices. For situations where high reliability is necessary, the authors suggest that users run preliminary studies in Ptera Software to iteratively approach a workable solution, then use a higher fidelity simulation technique to check the result.

VI. Optimization: Methodology

Before any optimization, the validation simulation took just under 22.5 minutes to run on a laptop with an Intel® Xeon® E3-1505M v5 CPU running at 2.81 GHz base speed and 15.3 GB of usable RAM. This converged configuration used a prescribed wake, five uniformly spaced chordwise panels, 18 uniformly spaced spanwise panels, and simulated three flap cycles. The solver discretized these cycles into 224 time steps, the difference in time between each being roughly 4.1 milliseconds. On the last time step, Ptera Software was analyzing a total of 4,104 ring vortices. For reference, this translates to multiple calculations involving matrices with hundreds of thousands of elements during each of the later time steps.

However, this time includes several importing and pre-processing steps that would not execute during a typical simulation. Therefore, to benchmark the software's performance, the authors created a separate simulation representing a typical use case. This simulation consisted of one symmetric pair of flapping wings with 200 total panels and ran for 133 time steps with a prescribed wake. Each reported solve time is the fastest out of three consecutive simulations. Reporting the fastest of multiple runs is standard practice in software performance benchmarking as it reduces error due to transient computational loads. The benchmarking simulation solved in 463 seconds before any modification.

With the baseline performance found, the authors incrementally implemented optimization schemes and reran the benchmark simulation. See Table 1 for the recorded benchmark times.

Modification	Solve Time (Best of Three)	Change from Baseline
Baseline	463 s	N/A
Fewer Force Calculations	309 s	33.3%
No Streamline Calculations; JIT Cross Products and Norms	248 s	46.4%
Fast Deep-Copies; JIT Centers	227 s	51.0%
JIT Subtractions	220 s	52.5%
Parallelized JIT Algorithms	149 s	67.8%
Vectorized Large Iterative Functions	144 s	68.9%

Table 1. The performance of the benchmark simulation after different modifications.

VII. Optimization: Discussion

As shown in Table 1, the authors decreased the run time of a typical simulation by 68.9%. This speedup is due to several computational techniques such as JIT compilation, parallelization, and vectorization. The authors also produced increases in efficiency by eliminating several unnecessary computation steps.

After a user has ensured that their simulation is iterating through enough flapping cycles to converge, they will only be interested in the forces on the wing during the final cycle. The force calculations are also highly computationally expensive. Therefore, the authors first eliminated the force calculations for all of the time steps before the last cycle, which resulted in a 33.3% drop in solve time for the benchmark simulation.

Next, the authors turned off the calculation of streamlines and implemented JIT compilation for two methods that perform cross-product and norm operations on massive matrices for each time step. JIT compilation is a technique that can increase the speed of functions in interpreted languages, such as Python. JIT compilation is especially beneficial for functions that are called many times by one program, which is the case in all time-stepping simulations. A JIT compiler pre-compiles an interpreted function so that the computer is not reinterpreting it before each execution. For this project, the authors used the popular open-source Numba package to integrate JIT compilation in Ptera Software [15]. By implementing Numba's JIT compilation in more functions and replacing an inefficient method for copying large objects, the authors reduced the benchmark solve time by 52.5%.

Then, the authors turned to the open-source community on Stack Overflow for advice on achieving further performance increases. One community member suggested modifications that used Numba to combine JIT compilation and parallelization, a computational technique that allows a computer's multiple processors to solve a problem concurrently [16]. After implementing this change, the authors also used vectorization, which increases the speed of matrix operations by reducing the frequency of interpreted function calls. Now, the benchmark's solve time had been decreased by 68.9%, meaning that it was more than three times faster.

While the solver is now fast enough to enable a user to iterate on aerodynamic designs quickly, its performance could be improved further. For example, future work will modify Ptera Software to detect symmetric geometry and operating conditions, such as the two sides of the flapping wing in this paper's simulations. Once detected, the software would copy the results of any required computations from one of the sides to the other. Other researchers are also conducting exciting work to optimize the underlying time complexity of UVLM itself [17]. The authors plan on testing these novel methods and, if appropriate, implementing them as new features in Ptera Software.

Conclusions

Based on this study's results, the authors of this study conclude that Ptera Software accurately implements the UVLM. They came to this conclusion due to good agreement between the trends and magnitudes of Yeo et al.'s experimental data, and Ptera Software's simulated results of the total lift force on a flapping-wing robot

The simulated and experimental lift force time histories differed in two noticeable ways:

1) The simulated force plot leads the experimental force plot by a slight phase shift.
2) The simulated results show the wings producing a significantly more negative lift force during the first and last quarter of the flap cycle, which corresponds to the robot's upstroke. The phase shift may be due to errors in how Yeo et al. cyclically averaged their experimental results. To investigate this further would require access to the researchers' raw data. The difference in magnitude is most likely due to flow separation on the wings' bottom surfaces during its upstroke.

The authors were also successful in dramatically increasing Ptera Software's computational efficiency. Future work should attempt to increase Ptera Software's speed further using computational techniques such as symmetry analysis. Additionally, a collaboration with the experimental study's original authors could improve this accuracy report's results. Such a collaboration would provide access to the actual wing geometry and raw data, eliminating any error associated with extracting this data from the paper and eliminating errors from the typos in the original document. Finally, Ptera Software should be modified to account for flow separation using a leading-edge vortex scheme described in the literature [14]. The inclusion of leading-edge vortices could increase Ptera Software's accuracy for all use cases and expand its capabilities to analyze hovering flapping-wing flight.

In the words of statistician George Box, "All models are wrong, but some are useful." In this spirit, researchers should only use Ptera Software within the bounds of the UVLM assumptions. Additionally, more well-established simulation tools or hand calculations must support any safety-critical engineering decisions informed by Ptera Software's results. Despite these limitations, Ptera Software has proved to be a powerful tool for simulating the dauntingly complex aerodynamics of flapping-wing flight. It has the additional benefits of being free to use, open-source, actively maintained, and designed to simulate flapping wings out of the box. For these reasons, Ptera Software is an excellent choice for the research and analysis of volant locomotion and ornithopter development.

References

[1] Shyy, W., Lian, Y., Tang, J., Viieru, D., and Liu, H. *Aerodynamics of Low Reynolds Number Flyers*. Cambridge University Press, Cambridge, 2007.

[2] Katz, J., and Plotkin, A. *Low-Speed Aerodynamics*. Cambridge University Press, Cambridge, 2001.

[3] Urban, C. Ptera Software. https://github.com/camUrban/PteraSoftware.

[4] Fritz, T. E., and Long, L. N. “Object-Oriented Unsteady Vortex Lattice Method for Flapping Flight.” *Journal of Aircraft*, Vol. 41, No. 6, 2004, pp. 1275–1290. https://doi.org/10.2514/1.7357.

[5] Lambert, T. *Modeling of Aerodynamic Forces in Flapping Flight with the Unsteady Vortex Lattice Method*. University of Liege, 2015.

[6] Gardiner, J., Razak, N. A., Dimitriadis, G., Tickle, P., Codd, J. R., and Nudds, R. L. Simulation of Bird Wing Flapping Using the Vortex Lattice Method. 2013.

[7] Nguyen, A. T., Kim, J. K., Han, J. S., and Han, J. H. “Extended Unsteady Vortex-Lattice Method for Insect Flapping Wings.” *Journal of Aircraft*, Vol. 53, No. 6, 2016, pp. 1709–1718. https://doi.org/10.2514/1.C033456.

[8] Yeo, D., Atkins, E. M., and Shyy, W. Experimental and Analytical Pressure Characterization of a Rigid Flapping Wing for Ornithopter Development. 2011.

[9] Rohatgi, A. WebPlotDigitizer. https://automeris.io/WebPlotDigitizer.

[10] Yeo, D., Atkins, E. M., Bernal, L. P., and Shyy, W. Experimental Investigation of the Pressure, Force, and Torque Characteristics of a Rigid Flapping Wing. 2012.

[11] Binder, S., Wildschek, A., and De Breuker, R. Extension of the Continuous Time Unsteady Vortex Lattice Method for Arbitrary Motion, Control Surface Deflection and Induced Drag Calculation. No. June, 2017.

[12] Deperrois, A. *Guidelines for QFLR5 v0.03 XFLR5 Analysis of Foils and Wings Operating at Low Reynolds Numbers*. 2009.

[13] Lambert, T., Razak, N. A., and Dimitriadis, G. “Vortex Lattice Simulations of Attached and Separated Flows around Flapping Wings.” *Aerospace*, Vol. 4, No. 2, 2017. https://doi.org/10.3390/aerospace4020022.

[14] Roccia, B. A., Preidikman, S., Massa, J. C., and Mook, D. T. “Modified Unsteady Vortex-Lattice Method to Study Flapping Wings in Hover Flight.” *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2628–2642. https://doi.org/10.2514/1.J052262.

[15] Lam, S. K., Pitrou, A., and Seibert, S. Numba: A LLVM-Based Python JIT Compiler.

[16] Urban, C., and Richard, J. Can I Speed up This Aerodynamics Calculation with Numba, Vectorization, or Multiprocessing? - Stack Overflow. https://stackoverflow.com/q/66750661/13240504. Accessed May 13, 2021.

[17] Jones, B., Dunning, P., and Maheri, A. “The Double-Tree Method: An O(n) Unsteady Aerodynamic Lifting Surface Method.” *International Journal for Numerical Methods in Fluids*, Vol. 92, No. 10, 2020, pp. 1394–1414. https://doi.org/10.1002/fld.4833.