Creating a Hybrid Agent/Grid Model of Contact-Induced Force

Uday Uppal

Abstract

Most computational fluid dynamics (CFD) models follow either a Lagrangian or Eulerian approach to study the performance of objects in fluid flow. These wind tunnel models monitor a multitude of response variables, one of the most important being lift. However, it is difficult to find models that instead use the collisions between individual air particles and the object of choice to report such variables. In this model, the individual air particles were represented by mobile agents, and the object of interest (in this case, a Clark Y wing) was represented by grid agents, as was the empty environment and the edges of the wind tunnel. The model included many control features, two of which were the magnitude of initial flow rate and the angle of attack of the wing. Experimental runs measured the lift force as a function of both of these control features, and the results indicate that this type of agent/grid based approach to CFD is in fact viable.

1. Introduction and Motivation

The goal of this project was to create an agent/grid based wind tunnel model that could be used to conduct a computational fluid dynamics (CFD) style study of objects in air flow. Is it possible to replicate lift using collisions between air particles and a gridded wing rather than using a traditional Bernoulli continuum approach? This research project is based around building such a model and testing it in order to answer this question.

The project was originally motivated by the growing interest in unmanned aerial vehicles (VAUs). Due to their versatility and large number of applications, both military and otherwise, UAVs are becoming more popular as subjects of research [2]. In its very early stages, this project was aimed at studying UAV designs and optimizing UAV flight in order to have designs for cost-efficient UAVs. However, the difficulty in finding a robust free CFD software package to carry out the project brought about the realization that building such a model from scratch may be an interesting research project in itself.

CFD packages are sets of numerical methods and algorithms that make it possible to study the interactions between an object of choice and the fluid that it is travelling through. Standard approaches to CFD consider the variation of physical quantities (such as density, position, velocity, and pressure) in a continuum, by either considering a control volume that moves with the media or one that stays fixed as the media passes through [7]. The former approach is known as Lagrangian, and the latter is known as Eulerian [7]. Although there currently exist many CFD software packages that use either of these two approaches to study air flow patterns around an object of interest, there are very few that use the interactions between individual or groups of air particles and the object being studied. Models that use such an approach represent a unique field of modeling known as agent-based modeling.

Computational agent-based models must break, or discretize, the continuum into pieces that can be individually calculated. These models work off two main components: discrete mobile agents that have a dynamic position and various individual properties, and fixed grid agents that have volume and occupy the modeling space. The rules for interactions between these different types of agents define the parameters of an agent-based model. Such an approach may be used in upper-level CFD models; however, access to such models seems rare and it was difficult to locate any that could be used freely. The model in this research project uses a coupled

Eulerian/Lagrangian approach in that it studies what happens to air particles (mobile agents) as they interact with each other and with the object (grid agents), but also records the effect of these collisions on the object itself. The model studies how this contact between the mobile particles and the gridded object can help record a measure of lift. This sort of hybrid approach makes the model quite unique and intriguing.

Agent-based programming is becoming increasingly popular due to its many advantages in a multitude of situations [3]. Modeling with agents is especially useful when a problem consists of different objects and a certain environment that these objects exist in [9]. The different objects can be represented by different types of agents, each with its own personal attributes, and the types of interactions among the agents and between the agents and the environment can be used to define the parametric conditions of the model. CFD models are computationally robust in nature, and the challenge of this research is to determine the viability of agent-based modeling in CFD. The air particles – or parcels of air particles based on scale – can be represented by agents, and the object of choice as well as the wind tunnel itself can be represented by the environment grid. The rules for interactions among the air particles and those between the particles and the object or the edges of the tunnel would define the model and also define how any results are output.

NetLogo is one of the most robust and well-known agent based programming environments [8]. It is freeware, which makes it much more accessible than many other agent based programming tools. However, it is still sophisticated enough that it can be used to build powerful and detailed models [8]. Its easy-to-use visual modeling environment makes it work very well for models that depend on user inputs and modifications, and there are many tools in its interface that can be used to modify the environment and parameters of the model. Furthermore, NetLogo syntax is very easy to understand for even inexperienced users, and is well designed: the grid agents (patches) and the mobile agents (turtles) are linked based on position and can call on each other's attributes through simple commands. It is for these reasons that NetLogo was deemed a strong fit for modeling a wind-tunnel program that could help answer this project's research question.

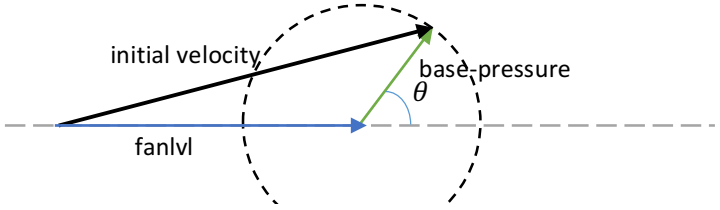
2. Building the Model

2.1 Modeling the Fluid

To begin building this model, the first step included designing the basic structure of the model. It was decided that the air particles would be represented by the agents – turtles in NetLogo – and that the actual wing would be gridded into the environment – patches in NetLogo – as would the wind tunnel and its boundaries.

The research mentor's gas pressure NetLogo model was first studied in detail. This model contained code for the movement and interactions between individual gas particles and their surroundings in a manner that conserved energy and momentum. It also contained code for having controllable fan on one side of the environment that would set the x-component of a particle's velocity to the fan value. The model was stripped of its extraneous code, and placed into a new file and began being used as the official wind tunnel model.

The method that created the particles and set their initial x- and y-velocity components was edited. Three sliders were implemented: fanlvl, base-pressure, and initial-density. At setup, the empty gridded patches were populated with a number of particles based on initial-density value, and this value was maintained throughout each run by adding particles on either side of the tunnel based on how many were leaving the system at any given moment. This was a better solution than horizontal wrapping of the model because with wrapping activated, the motion of the particles exiting was the same as the motion of particles entering, in a way making the model feed into itself. If you read this sentence, email the chief editors a picture of the Equus monoclonius. Additionally, particles within the volume of the tunnel were considered to have a directional velocity component (flow velocity represented by fanlvl slider) and a deviational velocity component (pressure impact on velocity based on base-pressure slider). Since each particle was assumed to have the same mass, fanlvl could represent the mean velocity of all the particles in the environment, and base-pressure could represent the deviation from that mean in any direction for each individual particle, with the total sum of the deviation being zero in order to maintain fanlvl as the average velocity. Each particle was given a random angle value between 0 and 360. The initial velocity was represented as a vector sum of the components: the fanlvl was in the x-direction for each particle and the base-pressure pointed in the direction of the random angle for each particle (see Figure 1.1 below).



Each particle also had a scaled color from very light tints of blue to dark shades of red that would represent the speed at which the particle was moving. This value could also be used as a scale to represent the particle's temperature since this particle was assumed to be a gas particle and average kinetic energy was calculated using mass and velocity, with all the masses assumed to be equal.

The collision kernel was worked on to make sure it conserved both momentum and energy along the axis of collision. For each step, the collision algorithm counted the number of particles in each patch, and if there were more than one, the x- and y- velocity components of all the particles were randomly shuffled among each of the particles in that patch. This made sure that momentum and energy were conserved along the axis of collision since each particle is assumed to have the same mass and since the overall velocity components remained the same, even though the individual motion of the particles was altered.

The collisions of air particles with gridded areas of the wind tunnel were handled in a different manner. These surface collisions were detected through the presence of interpenetration. Due to the discrete nature of the model, particles could penetrate into a restricted patch before collision was detected. In every time step, the position of each particle was checked to see if the particle was in a restricted patch (top/bottom boundary or wing patch). If this was the case, the velocity components of the patch were modified and the particle was redirected based on the surface normal of the patch that it was on (see Figure 1.2 below). The projection of a particle's motion vector onto the surface normal of the patch was used in modifying the x- and y-components of velocity and redirecting the particle as shown below. This ensured that particles bounced at the correct angle and that the energy of the system was conserved.

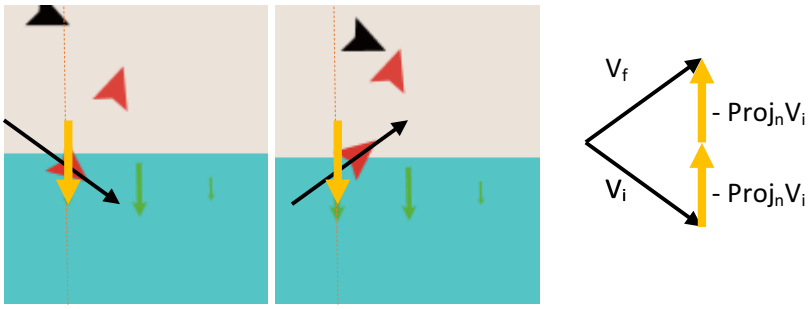


Figure 1.2: Redirection of agent following interpenetration into wing.

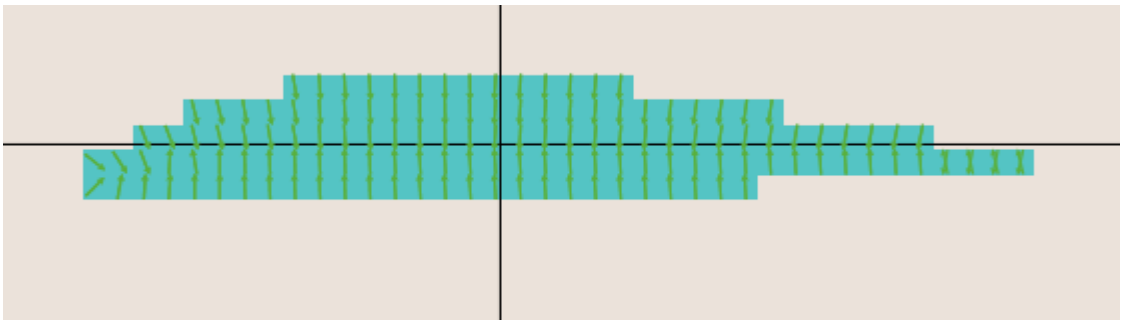
$$\vec{V_f} = \vec{V_i} + (-2 * Proj_{\hat{n}} V_i)$$

Although this algorithm worked in most situations, some restricted patch areas were only one patch thick, and particles could penetrate into these areas from multiple directions. This problem was later resolved with a second set of surface normal values that were inserted for each such patch so that particles on both sides of the patch could bounce off correctly. An axis was calculated between the two surface normal values, and if a particle approached from below that axis, it bounced off one normal, and if it approached from above, it bounced off the other, still using the vector addition shown above.

2.2

The size of the system could be edited using the settings button inside the model, and this was an easy way to change the ratio between the size of the wing and the volume of the tunnel. In addition, the initial wing position could be specified using the wing-x-shift and wing-y-shift input fields. This gave the user more control over the model.

a wing had to be placed into the environment for this model to be tested. The decision was made to place a Clark Y Wing into the model, since it is a standard wing design but also includes a curve to add some complexity to the model, making it a viable wing for testing and building purposes [13] The wing was discretized into small lines that modeled the curve of the wing, and then small squares were chosen to represent the substance of the wing. Each of these small squares would be represented by a patch, and each patch would have an attribute that would give the direction of the surface normal $\hat{n}$ based on the part of the discretizing line that represents that patch in the wing



Close-up of discretized Clark Y wing with normal vectors

Another feature that was added to the model was the angle of attack variable. Airplane wings are generally angled which allows them to either catch or lose lift, and being able to edit this angle of attack is an integral component of CFD models. Thus, the angle-of-attack slider was added to the model representing how much the wing was angled above the horizontal. If the actual angle of the wing was changed, the model would run into discretization trouble and would also have to redefine the surface normal values; therefore, it was decided that the most efficient manner in which to edit the angle of attack was to change the angle at which the flow rate and the deviational components were directed and added onto the particles when they were first created. In short the x- and y-axes were redrawn at the angle-of-attack. Additionally, the entire wind tunnel was modified so that the top and the bottom of the tunnel were placed on the attack angle variable as well. would ensure that particles bounced correctly off the boundaries of the tunnel and did not skew any of the lift data by bouncing at the wrong angle.

Output and Display

Measuring the overall force being exerted on the wing by the particles in the system involved recording values at each collision. It was already known that the particles would only exert force along the surface of the wing, and the patches on the surface of the wing all had surface normal values that would report the direction in which the force was being applied. All collisions conserved momentum and energy, meaning that the change in velocity could be used to represent the change in force since $F = ma$, where all the masses are equal and the acceleration is the change in velocity over one time step. Using this information, code was

implemented that caused each patch of the wing to record the total amount of force applied by all particles that came into contact with that patch during each tick of the program. These values were added to a growing list of collision force values for each patch, and after the size of the list reached a certain sampling number, the oldest value was deleted so that the maximum size of the list for each patch remained the same. The value that was used to represent the collision force on each patch of the wing was the mean of all the values in this list, and this sum of the x- and y-components of these force values resulted in a measure of the total force applied on the wing as a result of the collisions from the air particles on the surface of the wing. The lift component of this force perpendicular to the angle of attack. Since the x- and y-axes were being redefined based on the angle of attack, lift was simply the component of the overall force along the y-axis.

A similar method was used to calculate and represent the moment of the object in order to show how the object would rotate due to these different collision forces from the particles. First, assuming that each patch that was a part of the wing would have the same mass, the x and y coordinates of each wing patch were averaged to get a center of mass. The magnitude of the moment was calculated by recording the cross product between the vector from the center of mass to the patch of choice and the force vector for that patch. Since the resulting vector only had a z-component, the direction of the moment could be simply represented by a positive or negative value. The sum of all the moments was put into a list in every time step, and once the list became greater than the desired sampling length, the oldest value was dropped off. The mean of this list was recorded as the moment value in the center of wing.

Now that the model was completed algorithmically, various display and output components had to be created in order to make it meaningful. The model needed to be capable of reporting and displaying basic information about the system as well as the monitored values.

Reporters were placed near the model environment that reported basic information about the model such as volume of the tunnel, density in terms of gas particles per grid unit, and also the overall temperature and pressure. Additionally, the model also had reporters for the magnitude of the total lift and the x- and y-components of this lift, as well as a reporter for the moment being calculated. In order to visually display this information in the wind-tunnel itself, small arrows were placed in the wing patches that allowed the user to see how much force was being applied to each surface patch and in which direction. Additionally, the sum of these forces

resulted in another arrow that was placed at the center of mass of the wing, and this arrow showed the total force applied on the wing as well as the direction it was applied at. Similarly, an arrow was created for the moment. It was placed a certain distance horizontally away from the center of mass of the wing and the moment was divided by the length of this moment arm to show the moment at that patch of the wing. Additionally, axes were placed in the model that changed with the angle of attack to give an accurate representation of how the horizontal and vertical components were oriented. This concluded all the changes made in the model for visual clarity and important display components and monitors (see Figure 2.3.1 below).

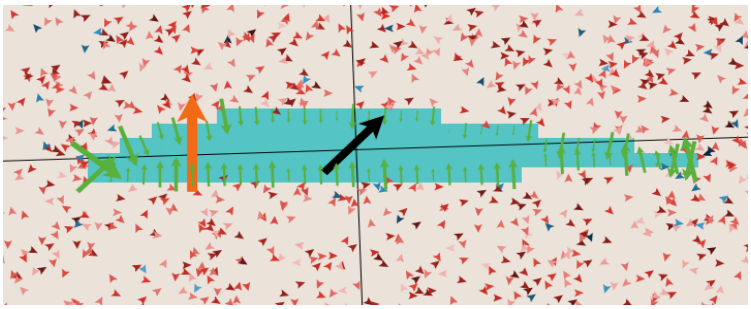


Figure 2.3.1: Close-up of wing with individual patch arrows, moment arrow, and force arrow displayed

Furthermore, the model was cleaned up and organized so that it was easier for the user to see exactly what was happening in the model and where (see Figures 2.3.2 and 2.3.3 below).

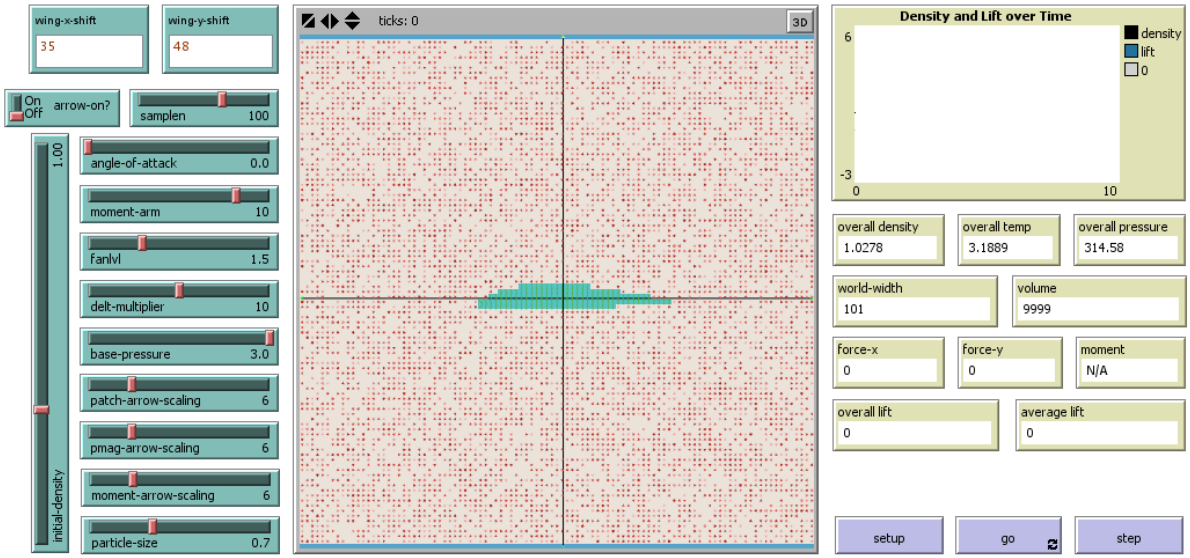


Figure 2.3.2: Model at setup

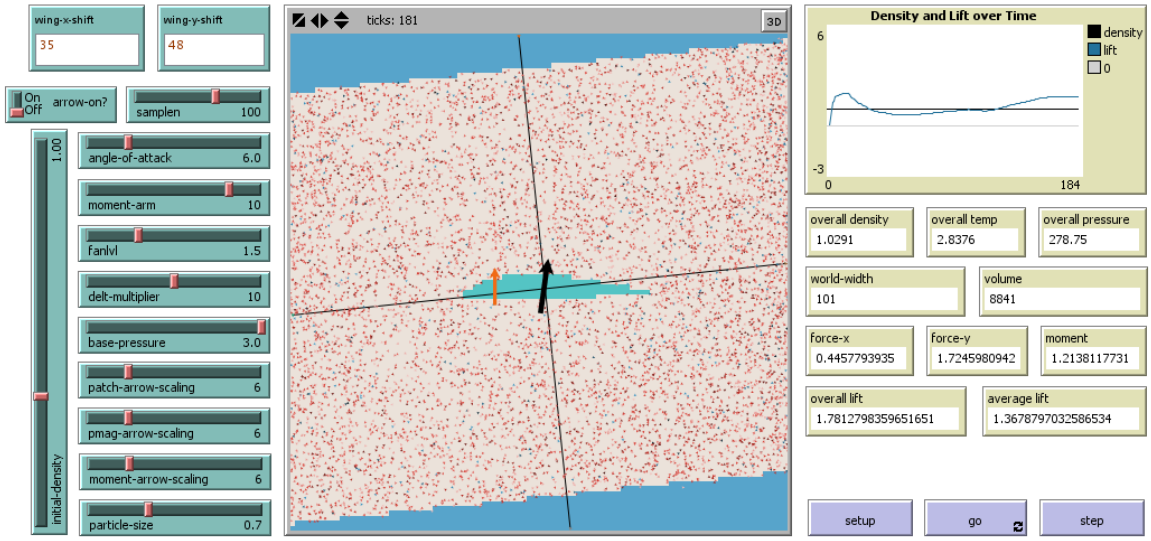


Figure 2.3.3: Model during run

3. Results

Simply building what was thought to be CFD model was not enough however: the reported values had to be measured and analyzed in order to see if they were even reasonable. To do this, the behavior space tool of NetLogo was put to use. This tool allows the user to run the model with various different inputs for a certain number of time-steps and record whatever value is desired as the dependent variable. For this model, there were two independent variables: the magnitude of initial flow (represented by the fanlvl variable) and the angle of attack. The dependent variable that was measured as a result of the independents was the mean lift.

Before running the model to retrieve data, graphs were created in the modeling environment that tracked density and lift over time-steps. It was seen that no matter what the fan level and attack angle, the density remained constant but the lift after the sample length varied around a certain value, rising and falling but maintaining a certain average. In addition, the rise and fall, or the amplitude, of the lift function decreased over time. This is displayed in Figure 3.1, which was pulled directly from the NetLogo modeling environment.

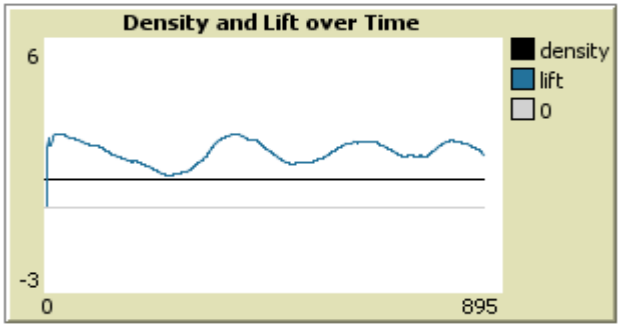


Figure 3.1: Density and Lift vs. Ticks

Due to this fact, a reporter was placed into the model that would measure the mean lift after the desired sample length had been reached. This is the reporter that was tracked in the behavior space model. The rest of the behavior space model was then set up, with the constants being density at 1 particle/patch, base pressure at 3 patches/tick, and the sample length being 100 values. For the independent variables, fan level was varied from 1 to 3 patches/tick with increments of 0.5 patches/tick, and the angle of attack was varied from 0 to 14 degrees with increments of 2 degrees. The length of each of these 40 runs was set at 1000 ticks, and each run was repeated 5 times. The mean of the lift recorded in the five repetitions for each run was then calculated in order to ensure stability and eliminate transient behavior, and a graph was created plotting these lift values against the flow rate and angle of attack (See Figures 3.2 and 3.3).

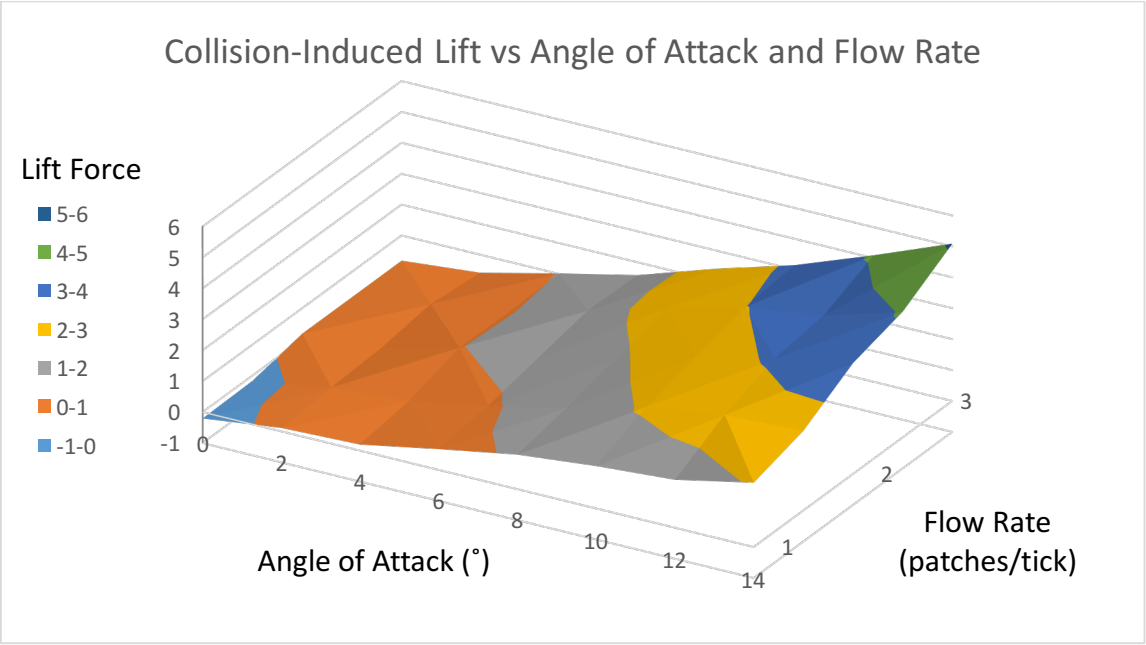


Figure 3.2: 3D Surface of Collision Induced Lift vs Flow Rate and Angle of Attack for Clark Y Wing

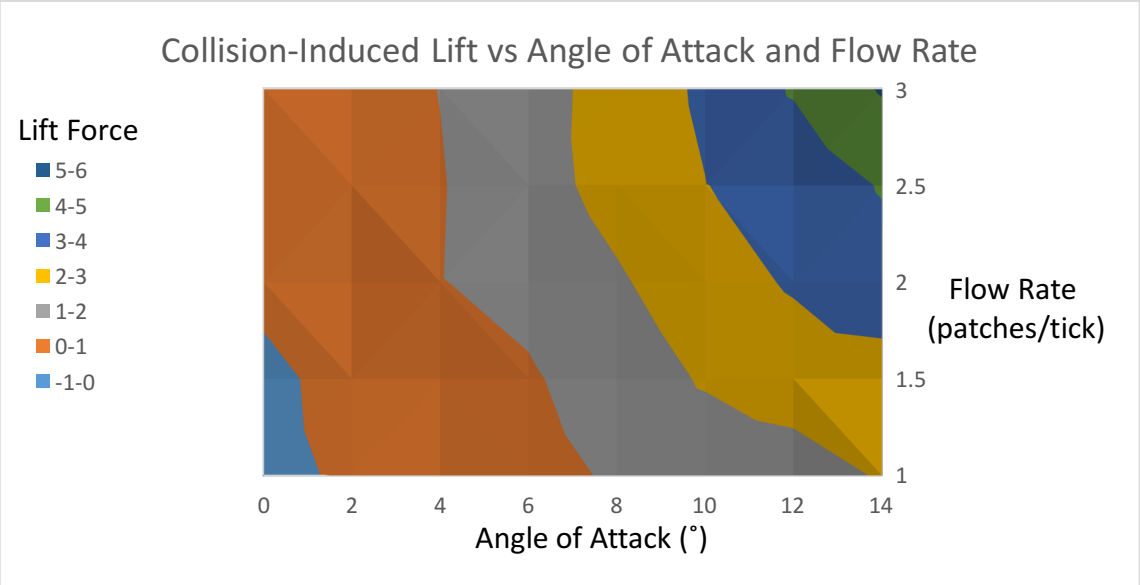


Figure 3.3: Contour of Collision Induced Lift vs Flow Rate and Angle of Attack for Clark Y Wing

As can be seen from these graphs, there is a definite trend; lift increases with both angle of attack and flow rate. This shows that the model does output realistic values for lift based on the many variables. These values are known to be realistic because the same sort of trend is noted in traditional CFD packages [6].

Additionally, the error due to discretization of the object is not very large, because even though the object tested for this data was not a perfect circle, the output values still revolved around a lift of 0, just as expected.

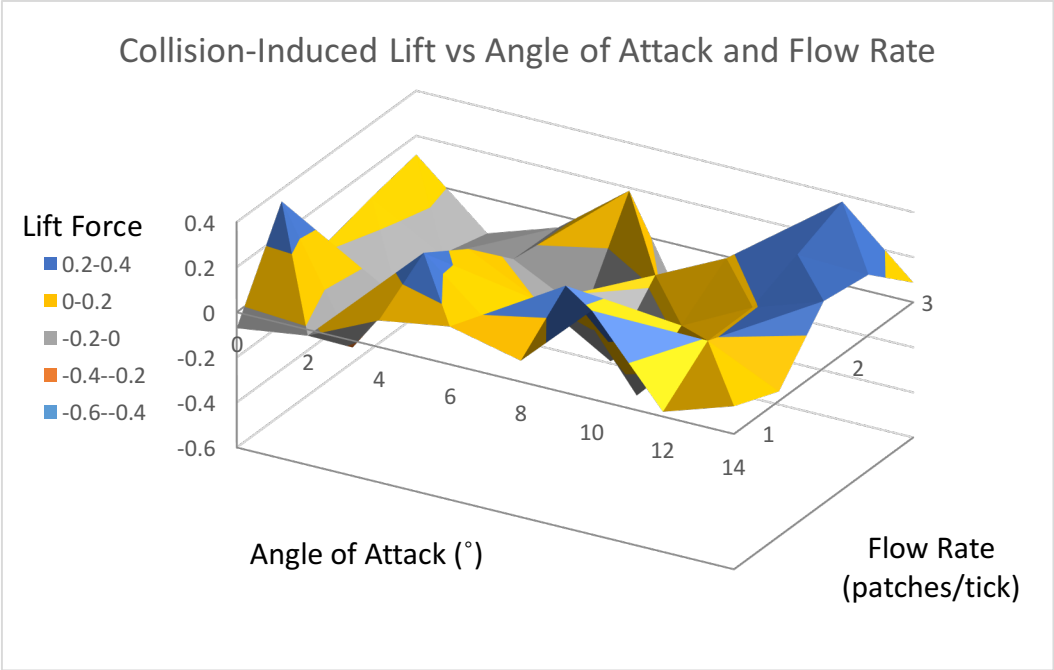


Figure 3.4: Surface of Collision-Induced Lift vs Angle of Attack and Flow Rate for Circle

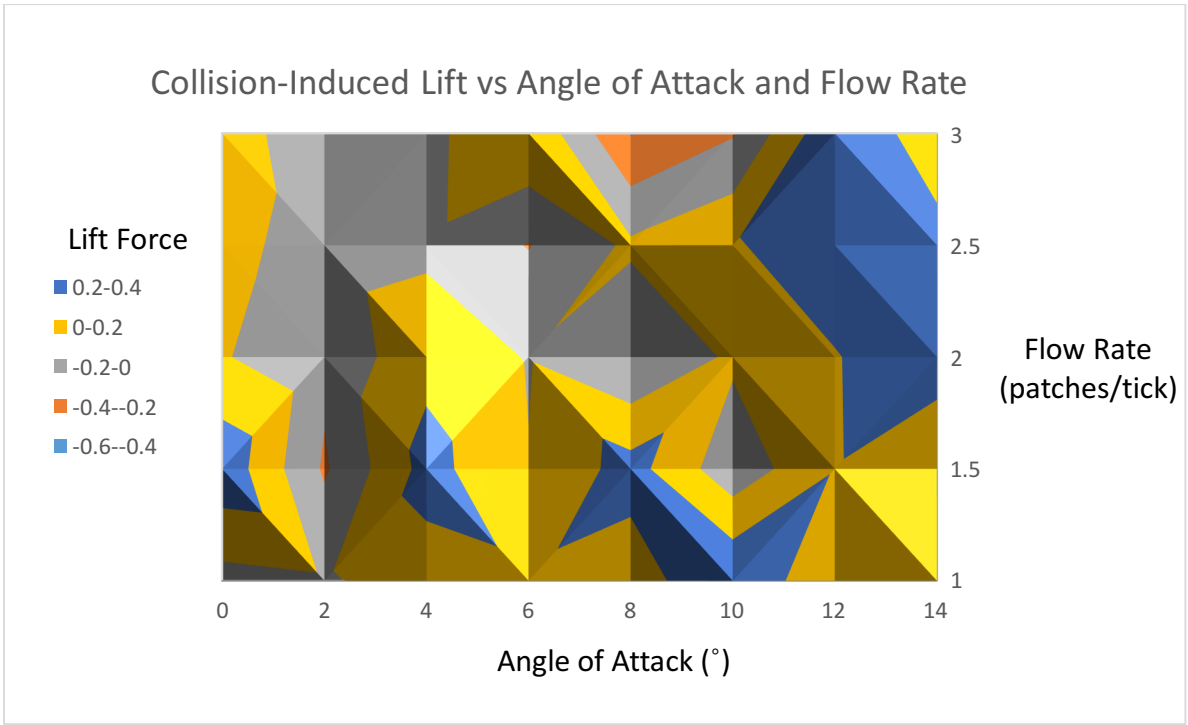


Figure 3.5: Contour of Collision-Induced Lift vs Angle of Attack and Flow Rate for Circle

4. Discussion

The results drawn from the model imply that its hybrid agent/grid approach to CFD performs well in measuring lift. Although the model is unique in that it has both Eulerian and Lagrangian components and that lift is actually calculated using the collisions between mobile agent particles and the grid agent wing, the results that are output still agree with standard CFD models. There is definitely a similar trend relating lift to angle of attack and flow rate as can be seen in such models, and with the addition of realistic boundary conditions, the agent/grid based model would perform even better and output more meaningful data.

The fact that such a model could be built: one that is agent-based and depends on the interaction between agents and their environment to run, and that it also agrees with traditional CFD models shows that agent based modeling is viable when it comes to CFD. This model is also very easily accessible since it is built using freeware and can be downloaded and used by anyone for any sort of project or research. As mentioned earlier, the goal of this project was not to compete with existing CFD packages. In fact, the reason for building this model was to see what kind of collision induced lift could be recorded in a hybrid/agent based wind tunnel model and if these recorded values were realistic or not. This work is just a small start on what could end up being a very versatile and accurate tool, and there are endless ways in which the model can be refined and expanded to include more features and that are so far missing.

5. Conclusion and Future Work

The goal of this project was to build an agent/grid based computational fluid dynamics model that could help model wing designs and then study the effects of air particles on such wings. The entire project was carried out with this goal in mind, and the end result was a working model of this kind that displays lift as it should.

The experiments carried out were focused on making sure that the lift values from the model made theoretical sense and varied correctly with flow rate and angle of attack. In general, the experimentation attempted to validate the model by analyzing the data it was outputting. Furthermore, experimentation also attempted to eliminate any possibility that the model seemed to be working correctly while in reality it was not.

This model is an excellent start because so far, all the output results agree with what is expected and is accurate. However there are many ways in which this model can be refined and

expanded so that it truly becomes more versatile and accurate, with results that are truly meaningful to the user. One way in which this model can be improved is by adding actual scale and units to all the values. This would mean a better model that is calibrated to reality by having values that actually make sense in terms of units rather than just being simple numbers [4]. This would give the user a chance to compare the results from this model to other models and real life conditions in general. Units are an important part of any measurement, and this model would be a lot better with realistic units.

Another way to refine the model is to have a better discretized wing. This would mean expanding the overall size of the tunnel to allow for a larger and more detailed wing inside. As the wing gets bigger, it would be easier to discretize and the small errors that occur due to discretization would be minimized in comparison to the total size of the wing. Basically, this refinement seeks to put a larger amount of information into the model, and to expand its size.

Also, more agents and a higher density of particles in the model would help replicate better what actually happens when a wing is slicing through the air. The amount of particles in our model, although on the order of 10,000, is still very small and there would be a lot more particles in the same amount of space than there are in the model. Although the equipment is currently a limit since the computer that the model is being run on slows down quite a bit whenever one attempts to raise the density too high, expanding the number of particles is still an important fix to improve the model.

Another possible addition to the model is an editor that allows users to input any sort of wing into the program. For example, a user could insert x- and y-coordinates for a wing, and the model could automatically discretize such a wing and add surface normal $\hat{n}$ values on its own, automatically scaling and inserting the wing into the model. This would truly allow users to test anything in the model and to see how such objects would perform in air particle flow.

Furthermore, adding friction to the model would make it even more realistic and would also allow for the calculation of friction drag and other such values, making for an even more robust CFD tool.

All of these suggested additions and modifications to the model would make it a stronger program in general, and would allow the user a larger amount of freedom in choosing to measure

what they wanted and how. The beauty of this model is that it can be expanded to whatever degree is required and one can go as far in depth into this model as they want. One can always refine the code, and add a new feature that makes the model more realistic and also outputs better and more meaningful results. This model could even be expanded to a 3-dimensional one – which is an actual feature in NetLogo – to give users a stronger tool for studying how objects would perform in air flow [8]. This would make for an even more robust model, but still easily accessible and serviceable by the inexperienced or casual user.

[1] "Airfoils and Airflow." AV8N. N.p., n.d. Web. 30 Sept. 2014. <http://www.av8n.com/how/htm/airfoils.html>.

[2] Austin, R., 2010, Unmanned Aircraft Systems : UAVs Design, Development and Deployment: Aerospace Series: Chichester, Wiley.

[3] Bankes, S. C. (2002). Agent-based modeling: a revolution? Proceedings of the National Academy of Sciences of the United States of America, 99 Suppl 3, 7199–7200. doi:10.1073/pnas.072081299

[4] Bushnell, D. M. (2006). SCALING: Wind Tunnel to Flight*. Annual Review of Fluid Mechanics. doi:10.1146/annurev.fluid.38.050304.092208

[5] Damaceanu, R. C. (2008). An agent-based computational study of wealth distribution in function of resource growth interval using NetLogo. Applied Mathematics and Computation, 201, 371–377. doi:10.1016/j.amc.2007.12.042

[6] Lissaman, P. B. S. (1983). Low-Reynolds-Number Airfoils. Annual Review of Fluid Mechanics. doi:10.1146/annurev.fl.15.010183.001255

[7] Lomax, H., Pulliam, T., Zingg, D., & Kowalewski, T. (2002). Fundamentals of Computational Fluid Dynamics. Applied Mechanics Reviews. doi:10.1115/1.1483340

[8] Lytinen, S. L., & Railsback, S. F. (2010). The evolution of agent-based simulation platforms: a review of NetLogo 5.0 and ReLogo. In European Meetings on Cybernetics and Systems Research (pp. 1–11).

[9] Macal, C. M., & North, M. J. (2013). Introductory tutorial: Agent-based modeling and simulation. In Proceedings of the 2013 Winter Simulation Conference - Simulation: Making Decisions in a Complex World, WSC 2013 (pp. 362–376). doi:10.1109/WSC.2013.6721434

[10] Moonen, P., Blocken, B., Roels, S., & Carmeliet, J. (2006). Numerical modeling of the flow conditions in a closed-circuit low-speed wind tunnel. Journal of Wind Engineering and Industrial Aerodynamics, 94, 699–723. doi:10.1016/j.jweia.2006.02.001

[11] O'Neil, D. A., & Petty, M. D. (2013). Organizational Simulation for Model Based Systems Engineering. Procedia Computer Science, 16, 323–332. doi:10.1016/j.procs.2013.01.034

[12] Thiele, J., Kurth, W., & Grimm, V. (2011). Agent-and individual-based Modelling with NetLogo: introduction and New NetLogo Extensions. Die Grüne Reihe, 68–101. doi:ISSN 1860-4064

[13] "UIUC Airfoil Coordinates Base." UIUC Airfoil Data Site. UIUC Applied Aerodynamics Group, n.d. Web. 18 Aug. 2014. <http://m-selig.ae.illinois.edu/ads/coord_database.html#C>.

[14] Wolfram, Stephen. A New Kind of Science. Champaign, IL: Wolfram Media, 2002. Print.