Modeling Airflow Over an Airfoil

NEW MEXICO
SUPERCOMPUTING CHALLENGE

April 7, 2006

Team 080

RIO RANCHO MID-HIGH

Project Members:

Alex Baker

Gunther Wong

Teacher:

Debra Loftin

Mentor:

Nick Bennet

Table of Contents

Executive Summary

In an online training manual by John S. Denker we learned that the air just ahead of the wing is moving upward as well as directed at the airfoil.
Of course, at the back of the wing the air is moving downward and passed the wing.
We also learned that along the leading edge of the wing is there is a stagnation line that divides the air.
At high angles of attack the stagnation line is found below the leading edge of the wing and when the air hits it the air will backtrack then flow over the leading edge of the wing.

Problem Definition:

The problem that is being addressed in this program is how to model the difference in pressure of air molecules flowing over an airfoil. With the current technology, this is important because it will help make a more efficient air foil so that it takes less power to make it fly increasing an airplane's efficiency. We learned that the air just ahead of the wing is moving upward as well as directed at the airfoil. Of course, at the back of the wing the air is moving downward and passed the wing. We also learned that along the leading edge of the wing is there is a stagnation line that divides the air. At high angles of attack the stagnation line is found below the leading edge of the wing and when the air hits it the air will backtrack then flow over the leading edge of the wing.

Problem Solution:

In this program we have used Star Logo to create a cross-section of an airfoil. The program is showing us how moving air particles are affected by the collision with the airfoil. Air pressure will be calculated for each based on the velocity of particles and the drag experienced along the surface of the airfoil.

Introduction

We did this project because we wanted to learn how the air pressure near the wing effects the lift of the wing. We have written this program in StarLogo. We wanted to write this program in StarLogo because we wanted to start off programming in an easy language. Right now, our program is calculating velocity and lift and our air molecules are responding to difference in pressure.

Facts of Aerodynamics

One major factor in aerodynamics is lift. Lift depends on the density of the air, the square of the wing's velocity, the air's viscosity and compressibility, the surface area of the wing, the shape of the airfoil, and the inclination of the airfoil to the air flow. In the lift equation (L= Cl*r*V^2/2*A), L is lift, Cl is the lift coefficient, r is density, V is velocity, and A is wing area. A huge factor in aerodynamics is Bernoulli's Law. Bernoulli's law describes the behavior of a fluid under varying conditions of flow and height. Bernoulli's law is the main reason that an airfoil has lift, as it describes how a fluid (in this case: air) would act when it encounters a difference in pressure and how it enables the airfoil to fly.

Wing Dynamics

The stagnation line is the point on the front of a wing in which air flow splits, some going up and some down. The splitting of the airflow depends on the wings dimensions and angle of attack. The trailing edge of the wing is the back edge of the wing. When it reaches the trailing edge of the wing, the air usually speeds up .The leading edge on the wing is the front edge of the wing. When it reaches the leading edge of the wing air usually speeds up. Viscosity is the measure of how an object clings to another. The viscosity of the air makes it cling to the airfoil.

The Program

The program we are using is StarLogo 6to model airflow over an airfoil. This has been the first time we have used a programming language. Are mentor Nick Bennet was very helpful in our creation of our program. We found that StarLogo is an agent based language. Our program is modeling how air particles move over an airfoil and eventually seeing how the wing reacts to the particles.

Currently the program we have written calculates the air particles velocity and the amount of pressure they exert on the airfoil. The user can change the shape of the airfoil by pressing a button. In the program, the velocity of the airfoil will determine the lift of the wing.

```
set :index (:index + 1)
                    set :running-total (:running-total + (:list-product / :item))
                    if (:throw < :running-total) [
                        set :y-cand (item :index :y-delta-list)
                        setxy xcor (ycor + :y-cand)
                        set y-delta :y-cand
                        stop
```

This is part of our code that tells the turtles how to move. In this part, the turtles are throwing a random sided die to decide where to move based on how many turtle are above and below it. The way the turtles move creates the simulated pressure when there are more turtles there is more pressure.

Assumptions

In order to limit the scope of the project, a number of important assumptions have been made.

- Viscocity and compressibility of the air particles are negilitabe
- The altitude remains constant
- The temperature remains constant
- The mass of the gas doesn't have an effect on the wing
- The angle of attack remains the same

StarLogo Airfoil

Airfoil 1

We have three more airfoils but have not been able to put them in.

Conclusion

This project has been going very well and we have learned a lot. The program still has a little work to be done that we will finish with our mentor. We plan on doing this challenge again next year. We think our project might be modeling a fluid through a nozel. We could put some of the knowledge we have learned this year to help us create our model. We hope to jump up to C++ next year but we might end up using Java.

Bibliography

Laminar Airflow. 2002 All rights reserved. © The Aviation History On-Line Museum. <http://www.aviation-history.com/theory/lam-flow.htm>

Beginner's Guide Home Page NASA. Last Updated Wed, Feb 16 08:15:29 AM EST 2005 by Tom Benson. Glenn Learning Technologies Home Page < http://www.grc.nasa.gov/WWW/K-12 >

<http://www.aa.washington.edu/faculty/eberhardt/lift.htm> "A physical description of Flight" by David Anderson

<http://www.aerospaceweb.org/question/airfoils/q0041.shtml> "NACA Airfoil Series"

<http://theory.uwinnipeg.ca/mod_tech/node68.html> "Bernoulli's Principle"

<http://scienceworld.wolfram.com/physics/BernoulliʼsLaw.html > "Bernoulli's Law" by Dana Romero

<http://www.daviddarling.info/encyclopedia/B/Bernoullies_law.html> "The Encyclopedia of Astrobiology and Spaceflight (Bernoulli's Law)"

*<http://www.av8n.com/how/htm/airfoils.html#sec-flow-intro>*John S. Denker 1996. See how it Flies, new spin on the perception, procedures, and principals of flight

<http://user.uni-frankfurt.de/~weltner/Mis6/mis6.html> "Misinterpretations of Bernoulli's Law"

Acknowledgements

We would like to thank Nick Bennett for coming to our school to help us learn how to use StarLogo. We would also like to thank Ms Loftin for being the best sponsor in the whole wide world. Finally, we would like to thank our parents for putting up with us and driving us around.

Appendix

Turtle Commands

```
turtles-own [
    y-delta
]


to initialize
    setxy (random screen-width) (random screen-height)
    if ((pc div 10) = 1) [die]
    setcolor blue
    set shape bubble-shape
end


to move
    let [:y-cand 0]
    let [:count-list []]
```

```
    let [:y-delta-list []]
    dotimes [:y-loop-count 3]
    [
        set y-delta (:y-loop-count - 2)
        if (((pc-at 0 y-delta) div 10) != 1) [
            set :count-list (lput (1 + count-turtles-at 0 y-delta) :count-
list)
            set :y-delta-list (lput y-delta :y-delta-list)
        ]
    ]
    set y-delta 0
    ifelse ((length :count-list) > 0)
    [
        ifelse ((length :count-list) > 1)
        [
            let [:list-product (product :count-list)]
           let [:throw (random (subproduct :count-list))]
            let [:running-total 0]
            let [:index 0]
            dolist [:item :count-list] [
                set :index (:index + 1)
```

```
                    set :running-total (:running-total + (:list-product /
:item))
                    if (:throw < :running-total) [
                        set :y-cand (item :index :y-delta-list)
                        setxy xcor (ycor + :y-cand)
                        set y-delta :y-cand
                        stop
                    ]
                ]
            ]
            [
                set :y-cand (item 1 :y-delta-list)
                setxy xcor (ycor + :y-cand)
                set y-delta :y-cand
            ]
        ]
        [
            let [:index 2]
            loop [
                let [:turtles-up maxnum]
                let [:turtles-down maxnum]
                if (((pc-at 0 :index ) div 10) != 1) [
```

```
                set :turtles-up (count-turtles-at 0 :index)
            ]
            if (((pc-at 0 (0 - :index )) div 10) != 1) [
                set :turtles-up (count-turtles-at 0 (0 - :index))
            ]
            ifelse (:turtles-up < maxnum and :turtles-down <
maxnum) [
                ifelse ((random (:turtles-up + :turtles-down + 2))
<= :turtles-down) [
                    setxy xcor (ycor + :index)
                    set y-delta :index
                    stop
                ]
                [
                    setxy xcor (ycor - :index)
                    set y-delta (0 - :index)
                    stop
                ]
            ]
            [
                if (:turtles-up < maxnum) [
                    setxy xcor (ycor + :index)
```

```
                    set y-delta :index
                    stop
                ]
                if (:turtles-down < maxnum) [
                    setxy xcor (ycor - :index)
                    set y-delta (0 - :index)
                    stop
                ]
            ]
            set :index (:index + 1)
        ]
    ]
end

to product :list
    let [:list-product 1]
    dolist [:item :list] [
        set :list-product (:list-product * :item)
    ]
    output :list-product
end
```

```
to subproduct :list
    let [:sum 0]
      let [:index 0]
    dolist [:item :list] [
            set :index (:index + 1)
            set :sum (:sum + (product (remove :index :list)))
    ]
    output :sum
end


to calculate-velocity :x-wing-velocity :y-wing-velocity
    output sqrt ((:x-wing-velocity * :x-wing-velocity) + ((y-delta - :y-wing-velocity) * (y-delta - :y-wing-velocity)))
end
```

Observer

```
patches-own [new-color neighbor-color-sum x-sum y-sum average-squared-velocity]

globals [turtle-force-scale moving-average-y moving-average-x total-lift lift-angle total-x total-y]


to setup
    no-display
```

```
    clear-turtles
    create-turtles-and-do 10000 [
        initialize
    ]
    set turtle-force-scale 0.025
    set moving-average-y []
    set moving-average-x []
    display
end


to go
    ask-patches [
        ;; NEXT LINE NEEDS TO BE ADJUSTED FOR
DIFFERENT X & Y VELOCITIES OF WING
        ifelse (((pc-at 1 0) div 10) = 1) [
            set new-color red
        ] [
            set new-color black
        ]
    ]
    ask-patches [
        setpc new-color
```

```
    ]
    ask-turtles [
        move
    ]
    nsum4 new-color neighbor-color-sum
    ask-patches-with [(pc = black) and (count-turtles-here > 0) and
(neighbor-color-sum > 0)] [
        let [:my-turtles list-of-turtles-here]
        let [:total-squared-velocity 0]
        dolist [:current-turtle :my-turtles] [
            ;; NEXT LINE NEEDS TO BE ADJUSTED FOR
DIFFERENT X & Y VELOCITIES OF WING
            set :total-squared-velocity (:total-squared-velocity + 1 +
((y-delta-of :current-turtle) * (y-delta-of :current-turtle)))
        ]
        set average-squared-velocity (:total-squared-velocity / length
:my-turtles)
        set x-sum 0
        set y-sum 0
        if (((pc-at 0 1) div 10) = 1) [
            set y-sum (y-sum - 1)
        ]
        if (((pc-at 1 0) div 10) = 1) [
```

```
            set x-sum (x-sum - 1)
        ]
        if (((pc-at 0 -1) div 10) = 1) [
            set y-sum (y-sum + 1)
        ]
        if (((pc-at -1 0) div 10) = 1) [
            set x-sum (x-sum + 1)
        ]
        ifelse x-sum != 0 and y-sum != 0 [
            set x-sum (x-sum * average-squared-velocity * (sqrt 2) /
2)
            set y-sum (y-sum * average-squared-velocity * (sqrt 2) /
2)
        ] [
            set x-sum (x-sum * average-squared-velocity)
            set y-sum (y-sum * average-squared-velocity)
        ]
    ]
    set total-x sum-of-patches-with [(pc = black) and (count-turtles-here >
0) and (neighbor-color-sum > 0)] [x-sum]
    set total-y sum-of-patches-with [(pc = black) and (count-turtles-here >
0) and (neighbor-color-sum > 0)] [y-sum]
    set total-lift (sqrt (total-x ^ 2 + total-y ^ 2))
```

```
    set lift-angle (atan total-x total-y)
    ;; NEED TO ADJUST VELOCITY, BASED ON LIFT/DRAG FORCE
end

to product :list
    let [:list-product 1]
    dolist [:item :list]
    [
        set :list-product (:list-product * :item)
    ]
    output :list-product
end

to subproduct :list
    let [:sum 0]
      let [:index 0]
    dolist [:item :list] [
        set :index (:index + 1)
        set :sum (:sum + (product (remove :index :list)))
    ]
    output :sum
```

end