Evaluating the Impact of Airfoil Design on Formula 1 Lap Times: A Simulation-Driven Approach with XFOIL

Sarvajit Karanth

Introduction

Airfoil design and its study plays a large role in motorsport design and manufacturing, where the profile of an airfoil can directly impact the drag force and downforce acting on the vehicle as well as its maneuverability and maximum velocity. Therefore, the optimization of airfoil design in the overall aerodynamic design of the race vehicle can lead to huge differences in track times as well as stability and driveability of the car. In determining the relationship between airfoil aerodynamics and an actual track time, the simulation of Coefficients of Drag (Cd) and Coefficients of Lift (Cl) is pertinent in the time calculations. Traditionally, wind tunnels have been used to test airfoils, as well as Computational Fluid Dynamics (CFD) software because of their reliability and accuracy. However, both are expensive to use either because of their sheer operational cost or computational requirement. In addition, both solutions can often be time-consuming, especially the usage of wind tunnels. As a result, computationally inexpensive simulation solutions are necessary to optimally and efficiently design airfoils. One such solution is XFOIL, a 2D airfoil analysis program that focuses on both inviscid and viscous flow computations around airfoils at subsonic flow conditions that produces values such as Cl, Cd, and the coefficient of moment Cm for the airfoil at the angle of attack chosen [1]. This research paper will focus on leveraging XFOIL to dynamically model and simulate the impact of different airfoil designs on F1 car performance around different turn radii and straight sections. The objective of this work is to develop an inexpensive, robust, and reliable simulation tool that utilizes various different variables related to the car to generate lap time differences and aid in airfoil design.

Rear Wing/Airfoil Specifics

The main requirement of any motorsport rear wing is to produce a high amount of downforce (the opposite of lift force) and a low amount of drag. Lift is generated by a difference between the pressure on the top-side and bottom-side of an airfoil. For example, on a regular airplane, its wings will have high air pressure on the bottom and low air pressure on the top, pushing the wing up, generating lift [2]. These differences in lift can also change with the angle of attack of the airfoil, which is the angle at which the airfoil is oriented with respect to the relative wind direction. Unlike a normal airfoil on an airplane, the airfoils in a rear wing are placed upside-down so that instead of generating lift force like on an airplane, it produces downforce, pushing the car into the ground. The airfoil on a rear wing would have high pressure on the top and low pressure on the bottom. The overall behavior of lift can be concisely represented by a single number, being the coefficient of lift (Cl). An increase in downforce can increase tire grip which in turn increases maximum cornering speeds and overall maneuverability. However, such aerodynamic features like rear wings also are bound to generate drag, such as friction drag, form drag, and lift-induced drag, the latter being mitigated

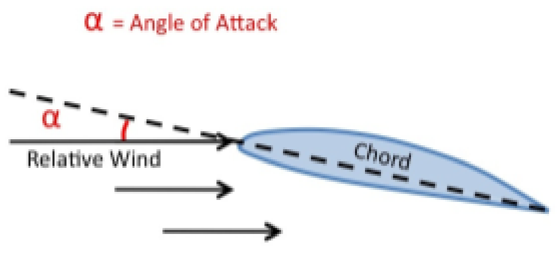


Figure 1 - Angle of attack (alpha) of an airfoil [3]

by rear wing endplates. All of these different drag forces can be combined into a single value, being the coefficient of drag Cd, which describes the overall drag force on the airfoil based on certain parameters. F1 rear wings are governed by the current FIA regulations, which state that they must consist of two sections 10-15mm apart with a single endplate body on each side unified into a single part called the “Rear Wing Bodywork” [4].

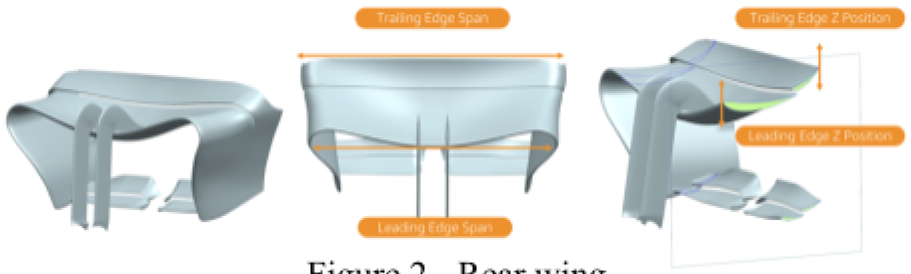


Figure 2 - Rear wing geometry [5]

After running a simulation on an airfoil in XFOIL, it returns Cl, Cd, and Cm values as well as the pressure distribution around the airfoil. A pressure distribution shows the pressure at every point in the airfoil, which is directly related to lift. These pressure distributions for an airfoil can

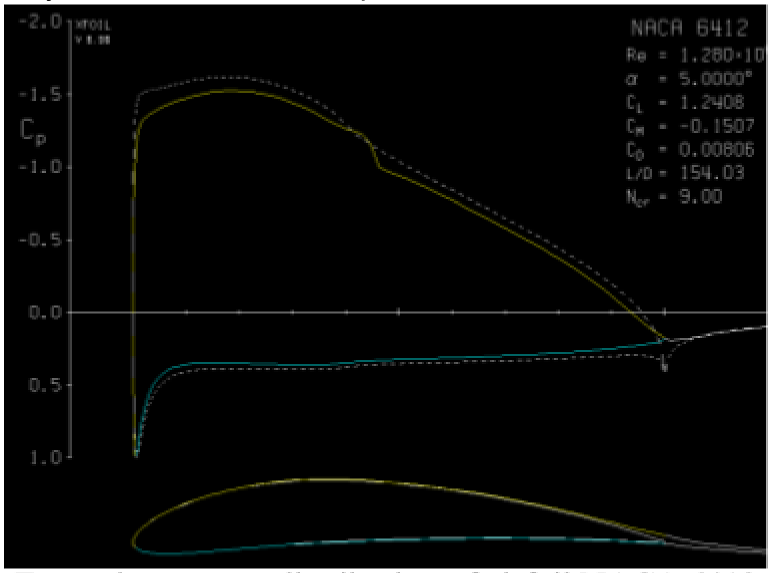


Figure 3 - Example pressure distribution of airfoil NACA 6412 at a Re of 1280000 and alpha of 5 degrees

change with things such as the Reynolds number (Re) and the angle of attack of the airfoil. The Reynolds number is the ratio of inertial to viscous force, which can be used to model different flow conditions in simulation. The Reynolds number changes with the airfoil’s velocity, as seen in its formula: $Re = \frac{\rho VL}{\mu}$, where ρ (rho) is the density of the air, V is the velocity of the airfoil, L is the length of the airfoil, and μ (mu) is the dynamic viscosity of the air, which at 20°C is 1.6 x 10^{-5} kg/ms. The Reynolds number is a dimensionless quantity. An important fact to airfoil simulation is the “no-slip condition" which states that for any fluid, the velocity of the fluid on the surface of the moving object is zero. The boundary layer is a layer of air at the surface of the airfoil where the velocity of the air transitions from zero to its free stream velocity. As seen in figure 4, boundary layers and air flow can either be laminar or turbulent; laminar flow is when each streamline flows separate and in an orderly fashion, while turbulent flow is when the streamlines flow into each other, 'mixing' the air together and diffusing the energy of each streamline into each other.

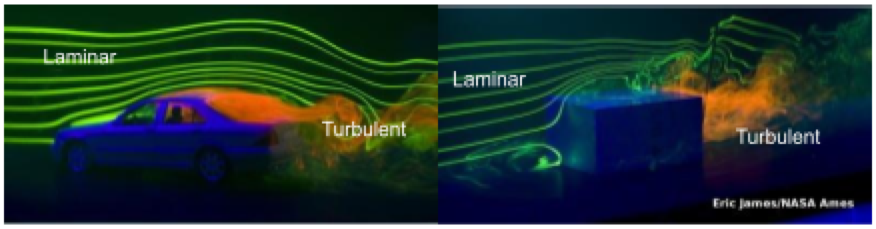


Figure 4 - Laminar and turbulent boundary layers and air flow [6]

Usually, when the Reynolds number is below 2000, the flow will be laminar, when it is between 2000-4000 it is in a transitional state, and when it is above 4000 it will be turbulent [7]. Boundary layers can also experience something called flow separation, where it does not have enough energy to keep following the path of the airfoil so it separates from the airfoil, generating drag and stopping the generation of lift. Having a turbulent boundary layer can help to prevent this by evenly distributing the energy of the layer. The yellow and blue lines in Figure 3 represent the viscous pressure while the white dotted line represents the inviscid pressure. At the rear of the airfoil, the boundary layer is also clearly visible.

Exploration of design space & airfoil selection

The ratio of the Cl and Cd values of an airfoil at different angles of attacks and Reynolds numbers can display the efficiency of a certain airfoil. Figures 5-8 show the Cl/Cd values for several different airfoils at different angles of attack and Reynolds numbers, calculated by XFOIL:

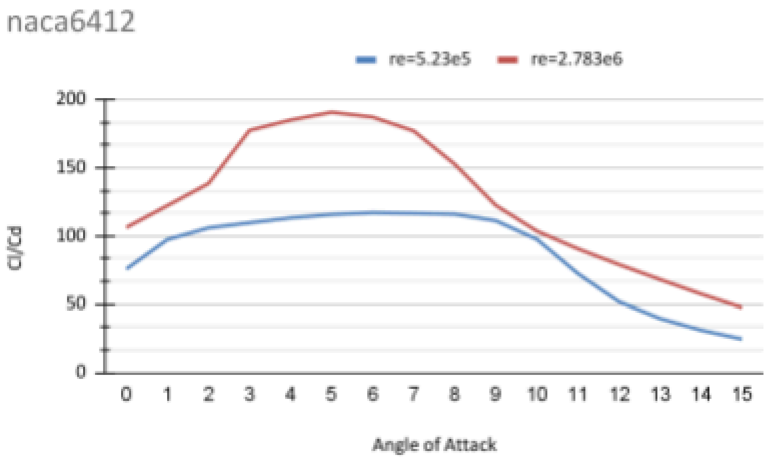


Figure 5

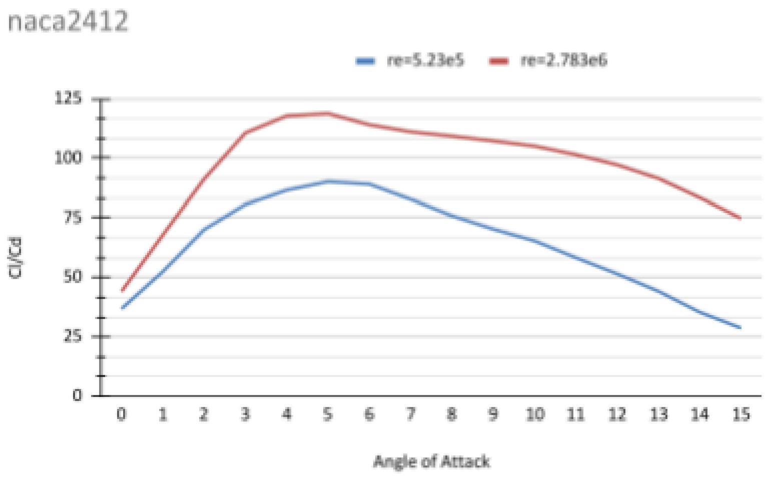


Figure 6

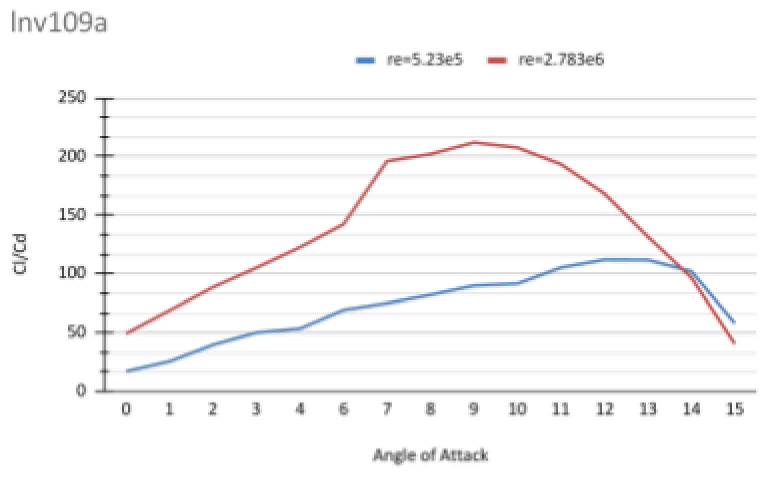


Figure 7

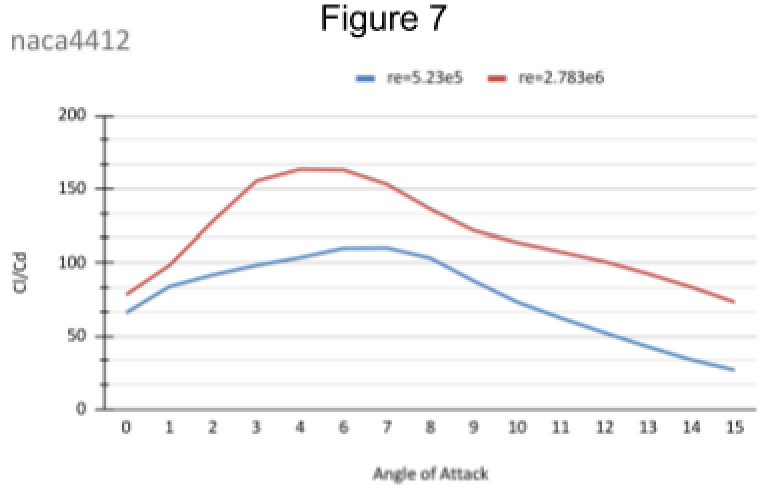


Figure 8

The red graphs represent the Cl/Cd values at high Reynolds numbers and by definition, high velocities, while the blue graphs represent the Cl/Cd values at low velocities. They have been plotted against the angle of attack of simulation. As seen above, for almost every graph there is a large disparity between the Cl/Cd values at low and high velocities, which translates to there being a large difference in handling and driveability for the driver in the straights and turns. Obviously, this is not a good thing for drivers when they want their car to be predictable.

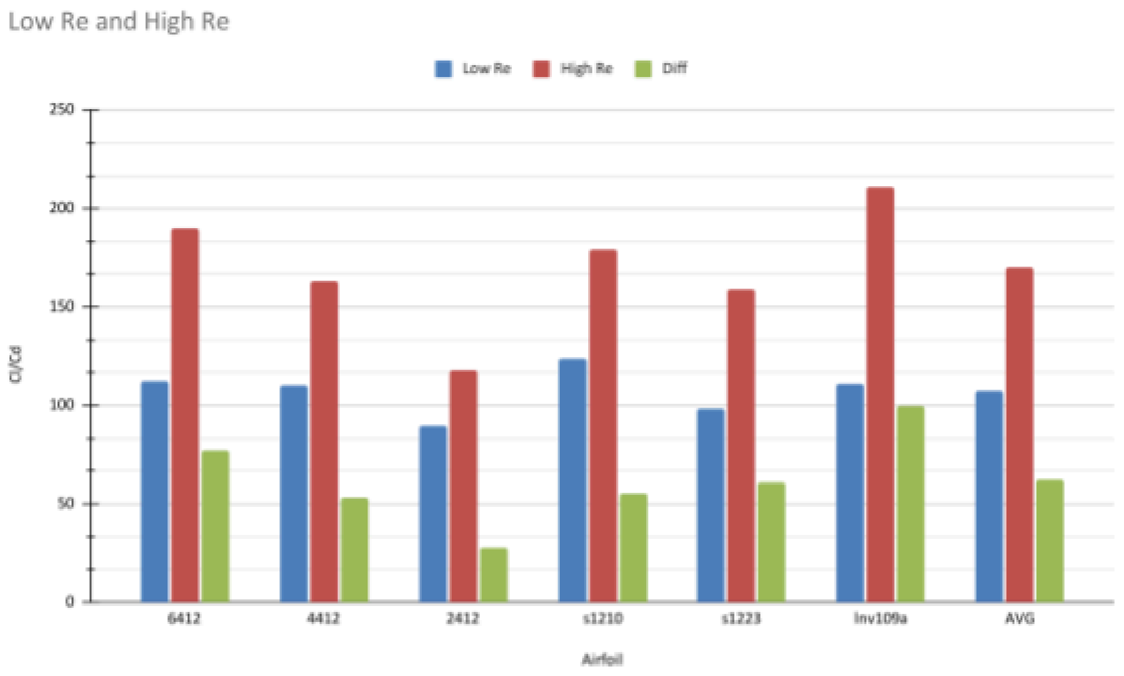


Figure 9

Figure 9 visualizes the differences between the Cl/Cd values at high and low velocities for airfoil above. Another thing to keep in mind is that drag force will not be as important in the turns as it will be on the straights, because the maximum turn velocity is already constrained by things like tire grip and downforce. Lift force (downforce) is highly desirable in the turns because as it increases, so does the friction force, and therefore the maximum velocity at which the driver can take the turn. Also, while turning F1 cars have considerably less velocity than they do in the straights, meaning that they also have lower Reynolds numbers. Therefore, to be optimal for F1 circuits, rear wing airfoils should have a relatively low Cd at high Reynolds numbers and a relatively high Cl at low Reynolds numbers.

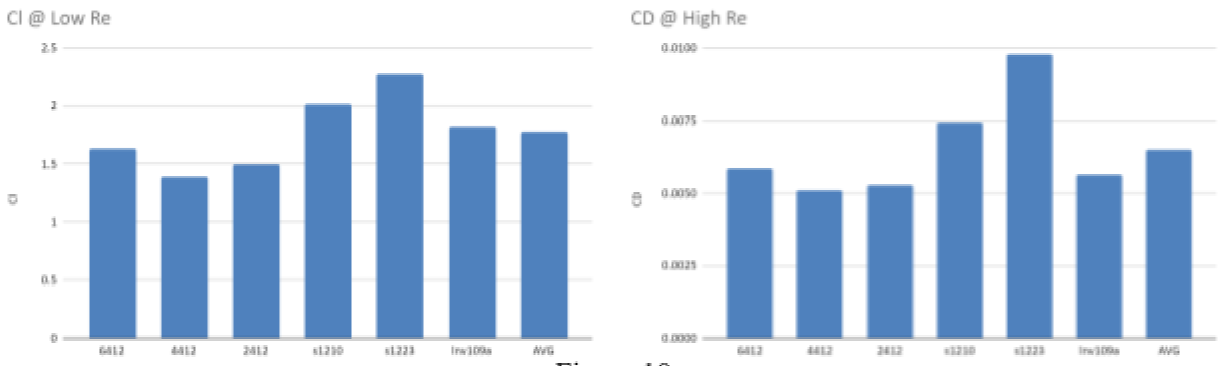


Figure 10

It can be seen in figure 10 that the airfoil LNV109a has the third highest Cl at low Reynolds numbers and the third highest Cd at high Reynolds numbers out of the six airfoils tested. It also had the highest Cl/Cd value at high reynolds numbers and a moderately high Cl/Cd value at low reynolds numbers. Using this approach and data collected, it can be concluded that out of the six airfoils tested above, the airfoil LNV109a is the most optimal for F1 rear wings. Out of all the airfoils, LNV109a has the third highest Cl value at high Reynolds numbers and the third lowest Cd value at low Reynolds numbers, meaning it is the most balanced out of the airfoils selected and therefore could be a good candidate for an F1 rear wing.

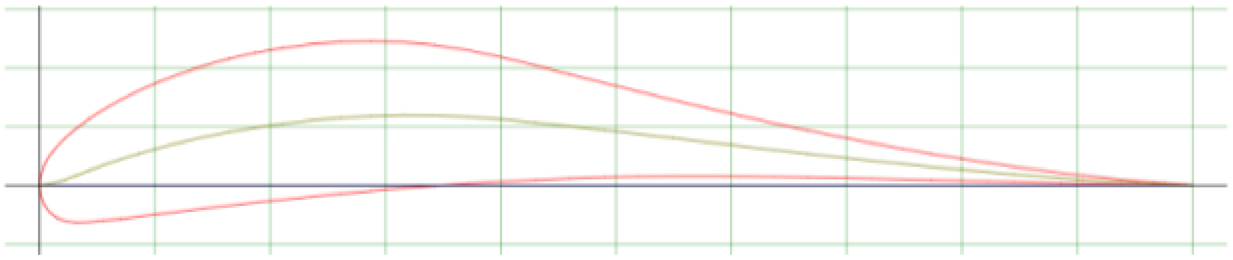


Figure 11 - LNV109a airfoil profile [8]

Mathematics of simulation

The relevant variables for analysis are:

Quantity	Symbol	Units
Force	F	N
Mass	m	Kg
Acceleration	a	m/s^2
Velocity	V, v	m/s
Density	ρ (rho)	kg/m^3
Area	A	m^2
Power (watts)	W, P	$kg \cdot m^2 \cdot s^{-3}$

Newton's Second Law states that the sum of forces on an object F = mass multiplied by acceleration (F = ma). The force of drag (F_D) is equal to: $\rho v^2 CdA$, where ρ is the air density, v is the object's velocity, and A is the cross-sectional area. The force of lift/downforce (F_L) is equal to: $\rho v^2 ClA$. For any object on the ground, it experiences a gravitational force pulling downwards towards the center of the earth. This force can be represented as $F_{gravity}$ =mg, where g is the gravitational acceleration constant for the earth (g = 9.81). According to Newton's third law, for each action there is an equal and opposite reaction. Therefore, there must be a force acting on the object opposite to the force of gravity. In this case, this would be "normal force," perpendicular to the surface which the object is on and pointing upwards. However, downforce is also in the same direction as the force of gravity, so the normal force, for the object on a level surface, would be equal to the sum of the force of gravity and downforce, in the perpendicular to the surface of contact and pointing upwards. The final force acting in the opposite direction of the car’s motion is the force of friction. It is equal to μ x normal force, where μ is the coefficient of friction. This is not the same as the last μ, dynamic viscosity. The coefficient of friction is similar to the coefficients of drag or lift, characterizing the object’s properties in one concise number. To summarize, there are two forces acting in the opposite direction to the car’s motion: the force of friction ($F_f = \mu(mg + F_L)$) and the force of drag F_D.

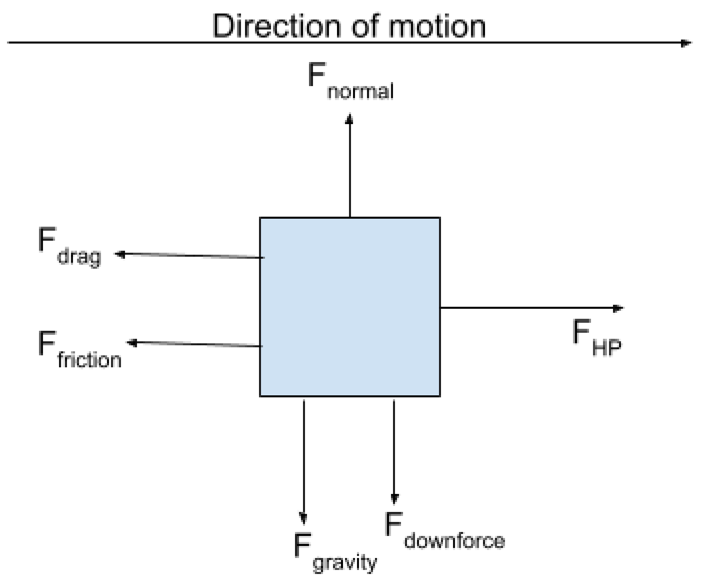


Figure 12 - Free-body diagram of the forces acting on the car

For the forces acting in the direction of the car’s motion, there is only one: the force generated by the car’s engine and electric motor system. Generally, the output of an engine can be represented as its horsepower, which is measured as the brake horsepower (BHP) at the engine’s crankshaft. However, this BHP cannot be used in calculations as it does not consider the energy loss the horsepower experiences as it moves through the transmission to the wheels. Therefore, the output of the car’s engine will be measured as its wheel horsepower (WHP). For the purposes of this paper, an F1’s total WHP will be about 1050 [9]. Converting this value from WHP to watts (multiplying WHP by 745.7), the car’s power at the wheels in watts is 782984.9 W. Power is equal to $\frac{work}{time}$. Rearranging this equation, you get that the force of the car in the forward direction is equal to $\frac{power}{velocity}$, where the velocity is the current velocity of this car. To calculate the motion of the car, the first step is to calculate the sum of the forces, which is:

$$F = ma = \frac{power}{velocity} - \rho v^2 CdA - \mu(mg + \rho v^2 ClA)$$

In this equation, the acceleration of the car is in terms of its velocity at that moment. To make everything in terms of velocity, acceleration can be represented as the first derivative of velocity with respect to time, or $\frac{dv}{dt}$. Substituting that into the equation and separating the terms:

$$\frac{1}{Pv^{-1} - \rho v^2 CdA - \mu mg - \mu\rho v2ClA} dv = dt \frac{1}{m}$$

To solve for velocity as a function of time, both sides must be integrated, where the limits of integration go from initial velocity to final on the left and 0 to a time t on the right :

$$\int_{vo}^{v} \frac{1}{Pv^{-1} - \rho v^{2}CdA - \mu mg - \mu\rho v^{2}ClA} dv = \frac{1}{m}\int_{0}^{t} dt$$

However, this equation cannot be integrated, as it has no closed-form solution. Therefore, to solve for the car's velocity, a different method must be used, utilizing numerical integration. Although the force acting on the car is not constant as the velocity changes, at a very small time interval dt, it can be assumed that the force is constant. Therefore, starting from a velocity 0, the acceleration can be calculated, and then the velocity of the car in that time step, and the distance it travels, using the velocity of the car in the last time-step. The equations for these values, where n is the current time step are as follows:

$$a_{n} = \frac{P(v_{n-1})^{-1} - \rho(v_{n-1})^{2}CdA - \mu mg - \mu\rho(v_{n-1})^{2}ClA}{m}$$

$$v_{n} = v_{n-1} + a_{n} \cdot dt$$

$$x_{n} = x_{n-1} + v_{n} \cdot dt$$

$$t_{n} = t_{n-1} + dt$$

$$dt = \lim_{k \to \infty} \frac{1}{k}$$

$$v_{0} = 0,\ t_{0} = 0, x_{0} = 0$$

As dt decreases, the computations necessary will increase, but so will the accuracy of the results you get. One limitation of these equations, although fully encompassing the forces acting on the car, only apply to when the car is driving straight. Therefore, different equations must be used to calculate the motion of the car through bends and turns.

As the car moves through a turn, it is the friction between the tires and the track keeping the car in a circular path. Assuming uniform circular motion, the velocity of the car moving through the turn can be calculated by (where r is the radius of the turn in meters):

$$\frac{mv^{2}}{r} = \mu(mg + \frac{1}{2}\rho v^{2}ClA)$$

Because we assume that the car is moving in uniform circular motion, that means that its speed is constant through the turn. Therefore, the equation can be rearranged like so to solve for v:

$$v_{turn} = \sqrt{\frac{\mu mgr}{m - \frac{1}{2}\mu\rho ClAr}}$$

Using this velocity and the distance of the turn, the time it takes to go around the turn can easily be calculated like so:

$$t = \frac{d}{v}$$

Implementation of mathematics, XFOIL

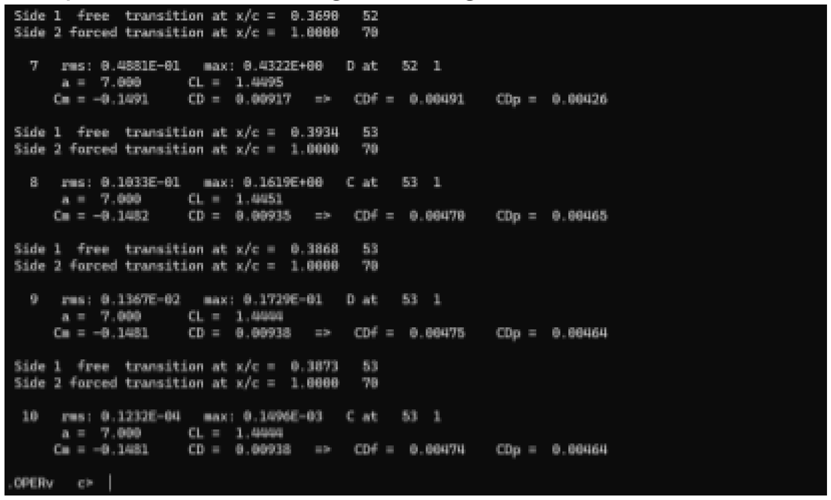
Constant	**Value in Code**
A (characteristic area of airfoil)	0.696 meters
μ (coefficient of friction)	0.7

ρ (air pressure)	1.225 kg/m^3
m (mass of car)	728 kg
hp (wheel horsepower)	1050 HP, 782984.9W
μ (dynamic viscosity of air)	1.5111e-5 kg/ms
Deceleration of car	5gs, 49.05m/s^2

Although the numerical integration method only uses basic algebra, to calculate the lap time of an F1 car around a track by hand with this method would be impractical due to the number of calculations needed to be made. Therefore, a computer solution such as Python can be used to implement these equations.

Two values highly important to the calculations that are not constant are the Cl and Cd values. Because of the time step method, we can generate the Cl and Cd values for the velocity of the past time step using XFOIL, which should be sufficiently accurate as the difference between adjacent velocities is very small. First, the program must call XFOIL and input the Reynolds number (calculated with the velocity), airfoil file path, and angle of attack. This is one possible implementation:

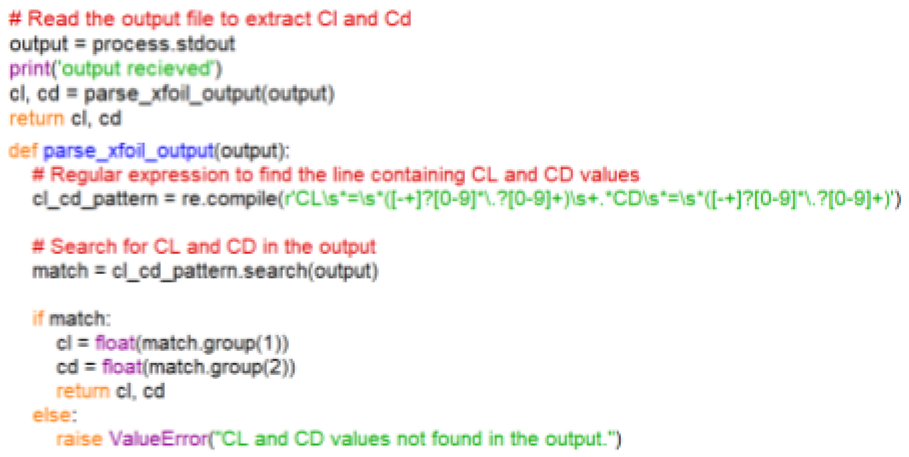
Then, the output of XFOIL must be parsed to extract the Cl and Cd values. Here is an example of the final output of XFOIL after generating Cl and Cd values:



```
Side 1  free  transition at x/c =  0.3690   52
Side 2 forced transition at x/c =  1.0000   70

  7   rms: 0.4881E-01   max: 0.4322E+00   D at   52  1
       a =  7.000      CL =  1.4495
     Cm = -0.1491     CD =  0.00917   =>   CDf =  0.00491    CDp =  0.00426

Side 1  free  transition at x/c =  0.3934   53
Side 2 forced transition at x/c =  1.0000   70

  8   rms: 0.1033E-01   max: 0.1619E+00   C at   53  1
       a =  7.000      CL =  1.4451
     Cm = -0.1482     CD =  0.00935   =>   CDf =  0.00470    CDp =  0.00465

Side 1  free  transition at x/c =  0.3868   53
Side 2 forced transition at x/c =  1.0000   70

  9   rms: 0.1367E-02   max: 0.1729E-01   D at   53  1
       a =  7.000      CL =  1.4444
     Cm = -0.1481     CD =  0.00938   =>   CDf =  0.00475    CDp =  0.00464

Side 1  free  transition at x/c =  0.3873   53
Side 2 forced transition at x/c =  1.0000   70

 10   rms: 0.1232E-04   max: 0.1496E-03   C at   53  1
       a =  7.000      CL =  1.4444
     Cm = -0.1481     CD =  0.00938   =>   CDf =  0.00474    CDp =  0.00464

.OPERv   c>
```

Figure 13

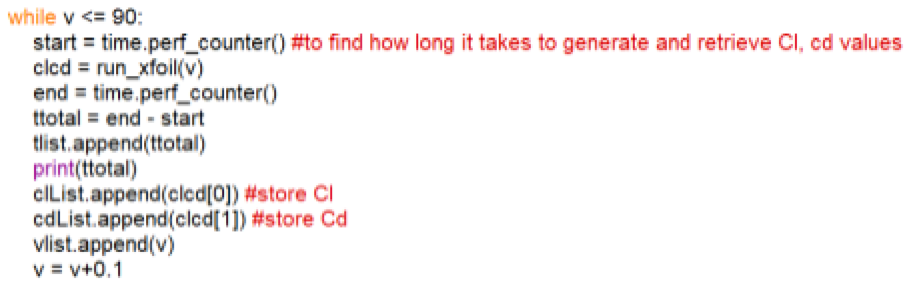
Using this output, we can temporarily save it, parse it, and extract and return the values necessary like so:



```
# Read the output file to extract Cl and Cd
output = process.stdout
print('output recieved')
cl, cd = parse_xfoil_output(output)
return cl, cd
def parse_xfoil_output(output):
   # Regular expression to find the line containing CL and CD values
   cl_cd_pattern = re.compile(r'CL\s*=\s*([-+]?[0-9]*\.?[0-9]+)\s+.*CD\s*=\s*([-+]?[0-9]*\.?[0-9]+)')

   # Search for CL and CD in the output
   match = cl_cd_pattern.search(output)

   if match:
      cl = float(match.group(1))
      cd = float(match.group(2))
      return cl, cd
   else:
      raise ValueError("CL and CD values not found in the output.")
```

Figure 14

Although this method is precise, the average time it takes for the code to generate and retrieve the Cl and Cd values from XFOIL is 0.54 seconds. Although 0.54s itself is not a long time, when there are thousands of times where Cl and Cd values need to be generated, the code would take much more time to run. One solution to this issue is to pre-generate the Cl and Cd values for all the velocities between zero and the car's max velocity, and then use these values while running the calculations. The level of precision of this list can be determined by the increment in between the velocity values - 1m/s, 0.1m/s, etc. Then, when retrieving each Cl and Cd values, the car's current velocity can correspond to a certain Cl and Cd value in the list according to its corresponding index. We can first generate the Cl and Cd value lists in a separate code using the run_xfoil function defined earlier, using a 0.1m/s increment and a max velocity of 90.0 m/s:



```
while v <= 90:
   start = time.perf_counter() #to find how long it takes to generate and retrieve Cl, cd values
   clcd = run_xfoil(v)
   end = time.perf_counter()
   ttotal = end - start
   tlist.append(ttotal)
   print(ttotal)
   clList.append(clcd[0]) #store Cl
   cdList.append(clcd[1]) #store Cd
   vlist.append(v)
   v = v+0.1
```

Figure 15

The contents of cdList and clList can them be copied to the simulation code, and retrieved based on the current velocity like so:

```
vtemp  = round(v, 1)

index = int((vtemp*10)-1)
print(index, vtemp)
cl = clList[index]
cd = cdList[index]
```

Figure 16

Because each Cl and Cd value corresponds to a velocity between 0.0 and 90.0 m/s, by first rounding the current velocity to one decimal point it should correspond directly to a Cl and Cd value in the list based on those value's indexes.

A third method of generating Cl and Cd values is by pre-generating the Cl and Cd values for velocities from 0 m/s to 90 m/s with an interval of 0.1 m/s and determining a piecewise function that accurately models the Cl and Cd values with a polynomial regression:

```
if v <= 10:
    cl = 0.01382*(v**2) + (-0.3065)*v + 3.445

else:
    cl = 1.7587
if v <= 16:
    cd = (3.407e-06*(v**3)) + (-0.0004676*(v**2)) + (0.007893*v) + (-0.002454)
else:
    cd = 0.01821
```

Figure 17

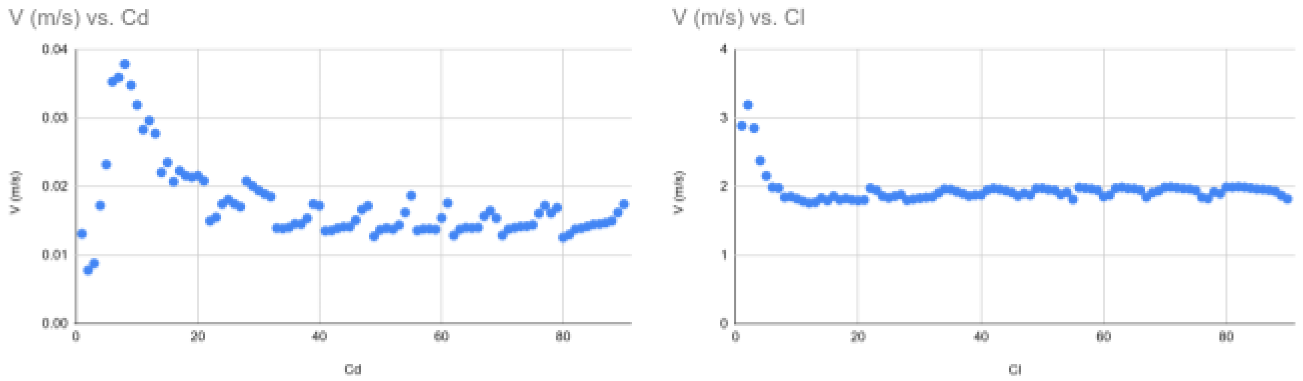
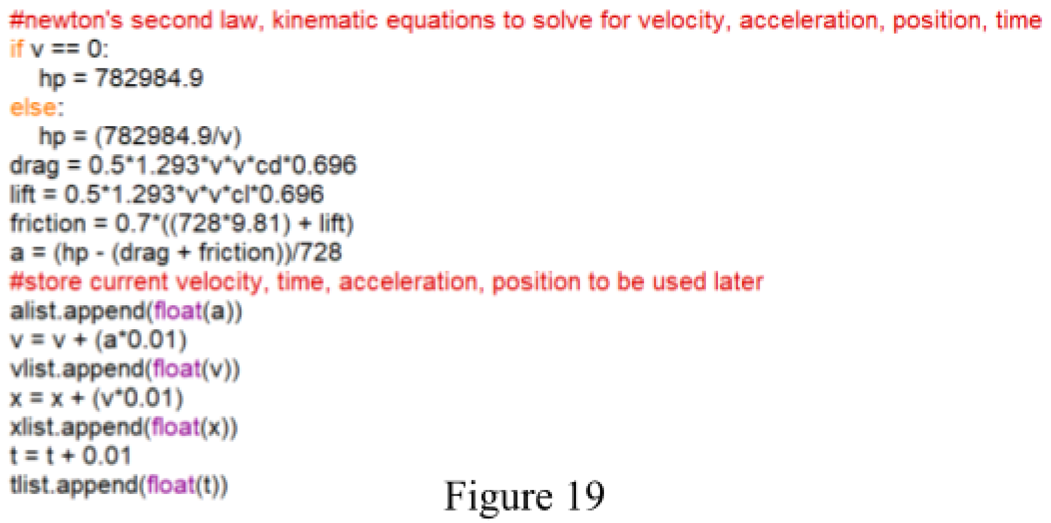


Figure 18 - Graphs of Cl, Cd vs V for LNV109a

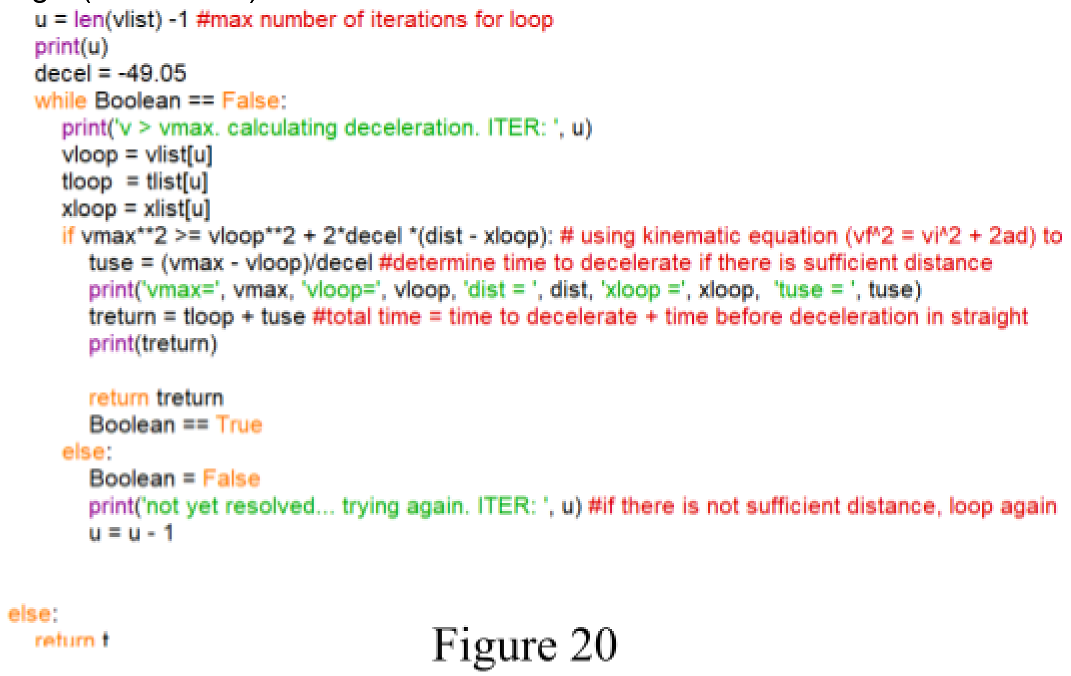
After retrieving Cl and Cd values, the equations for the motion of the car on the straights can be implemented like so, where the time step is 0.01s:



```
#newton's second law, kinematic equations to solve for velocity, acceleration, position, time
if v == 0:
    hp = 782984.9
else:
    hp = (782984.9/v)
drag = 0.5*1.293*v*v*cd*0.696
lift = 0.5*1.293*v*v*cl*0.696
friction = 0.7*((728*9.81) + lift)
a = (hp - (drag + friction))/728
#store current velocity, time, acceleration, position to be used later
alist.append(float(a))
v = v + (a*0.01)
vlist.append(float(v))
x = x + (v*0.01)
xlist.append(float(x))
t = t + 0.01
tlist.append(float(t))
```

Figure 19

This line of code will loop until the x value is equal to the straight distance. One thing that this process does not account for is the car's deceleration before entering a turn. By storing all the acceleration, velocity, distance, and corresponding time values to their respective lists, they can be retrieved later to determine the amount of time needed to decelerate at a max deceleration of 5g's (49.05 m/s^2) like so:

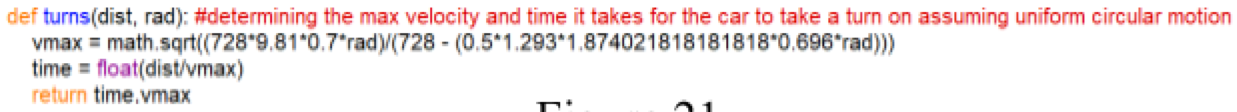


```
    u = len(vlist) -1 #max number of iterations for loop
    print(u)
    decel = -49.05
    while Boolean == False:
        print('v > vmax. calculating deceleration. ITER: ', u)
        vloop = vlist[u]
        tloop  = tlist[u]
        xloop = xlist[u]
        if vmax**2 >= vloop**2 + 2*decel *(dist - xloop): # using kinematic equation (vf^2 = vi^2 + 2ad) to
            tuse = (vmax - vloop)/decel #determine time to decelerate if there is sufficient distance
            print('vmax=', vmax, 'vloop=', vloop, 'dist = ', dist, 'xloop =', xloop,  'tuse = ', tuse)
            treturn = tloop + tuse #total time = time to decelerate + time before deceleration in straight
            print(treturn)

            return treturn
            Boolean == True
        else:
            Boolean = False
            print('not yet resolved... trying again. ITER: ', u) #if there is not sufficient distance, loop again
            u = u - 1


else:
    return t
```

Figure 20

This loop will iterate through each velocity value of the car at each time step starting at the end velocity, searching for a point where it can decelerate to the max turn velocity of the turn ahead in the distance there is before the turn at that time step. It then calculates the time it takes for the car to decelerate to this velocity, adds it to the amount of time the car had been on the straight before the deceleration, and returns that as the total time for that straight.

Finally, before running the code, the max turn velocity for each turn can be calculated like so:



```
def turns(dist, rad): #determining the max velocity and time it takes for the car to take a turn on assuming uniform circular motion
    vmax = math.sqrt((728*9.81*0.7*rad)/(728 - (0.5*1.293*1.874021818181818*0.696*rad)))
    time = float(dist/vmax)
    return time,vmax
```

Figure 21

Results and Limitations

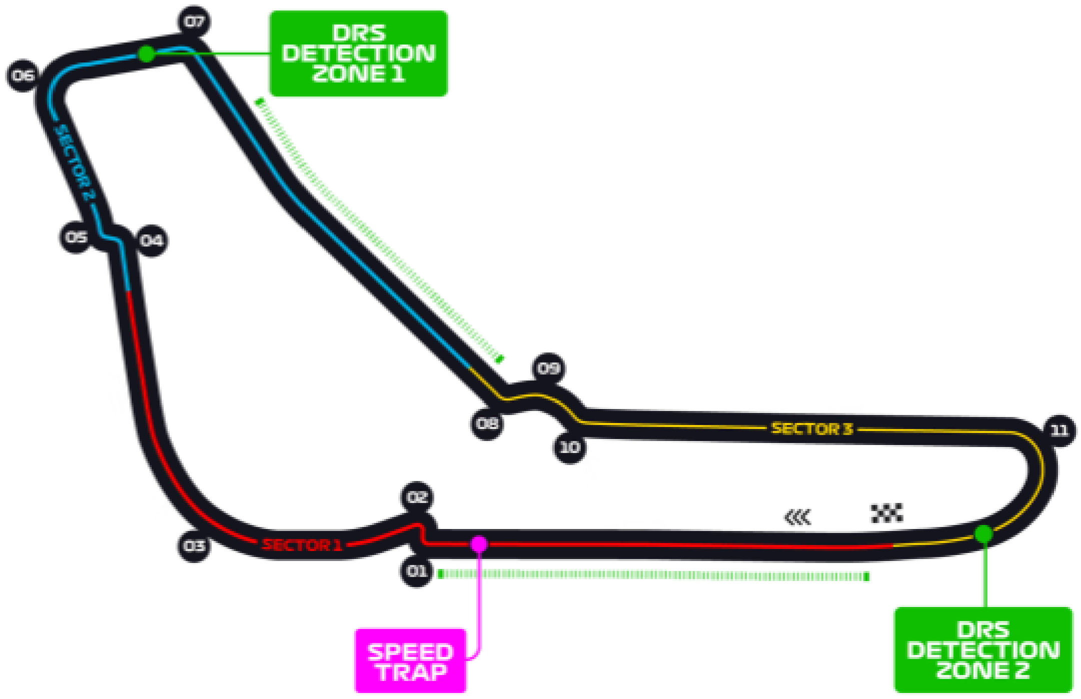


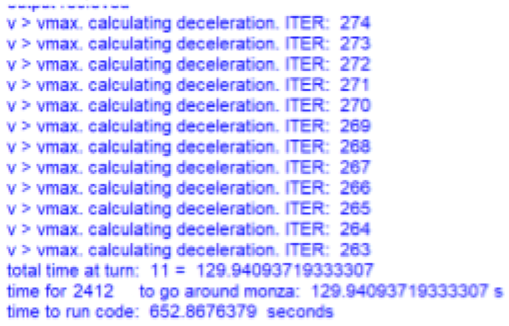
Figure 22 - Monza F1 configuration [10]

aight lengths sed in code) (m)	600	0	500	700	0	400	200	800	0	0	1000	0
rn radii (m)	16	16	327	18	18	50	45	30	40	50	80	300
rn lengths (m)	70	70	320	75	75	90	85	100	100	100	120	120

After running the code (with a time step of 0.01s) for the F1 track Monza, the total time for an F1 car with a rear wing using the LNV109a airfoil would be 131.64 seconds, or 2.194 minutes. This time is off from the current track record on Monza - 1:21:046 - set by Rubens Barichello by over a minute. This inaccuracy is a product of the code's limitations.

For example, in an attempt to reduce the amount of time the code took to calculate the lap time, the time step was set to 0.01 seconds. However, if it was set to 0.001 seconds, or 0.0001 seconds, the code's accuracy would be greater but the time it would take to compute on the same computer would increase. On a computer with an Intel i7-10700F CPU @2.90 GHz, NVIDIA GeForce RTX 2060 GPU, and 32 GB of DDR4 RAM, the program took 2009.73 seconds, or about 33.5 minutes to compute the lap time for the LNV109a car. Another limitation of the code is that it does not account for things like the driver's racing line, turns around corners which are not uniformly circular, DRS, tire temperature, and ERS deployment, which could all affect the total lap-time significantly.

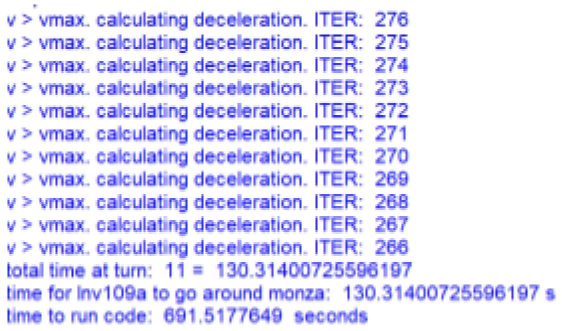
Although these limitations are present in the code, it can still be used to determine the relative lap time differences between different airfoils. Since the code will always have the same limitations every time you run it, they will affect the lap time for any airfoil equally. For example, earlier it was determined that out of the six airfoils researched, LNV109a was the most optimal for F1. Because the naca 2412 airfoil had a lesser Cl/Cd value at both low and high Reynolds numbers we can hypothesize that its lap time would be less than that of LNV109a.



```
v > vmax. calculating deceleration. ITER:  274
v > vmax. calculating deceleration. ITER:  273
v > vmax. calculating deceleration. ITER:  272
v > vmax. calculating deceleration. ITER:  271
v > vmax. calculating deceleration. ITER:  270
v > vmax. calculating deceleration. ITER:  269
v > vmax. calculating deceleration. ITER:  268
v > vmax. calculating deceleration. ITER:  267
v > vmax. calculating deceleration. ITER:  266
v > vmax. calculating deceleration. ITER:  265
v > vmax. calculating deceleration. ITER:  264
v > vmax. calculating deceleration. ITER:  263
total time at turn:  11 =  129.94093719333307
time for 2412     to go around monza:  129.94093719333307 s
time to run code:  652.8676379  seconds
```

Figure 23

After using the active XFOIL computation method to calculate the amount of time it took for the naca 2412 airfoil to go around Monza, the simulation determined that it would take the car 129.94 seconds to go around the track, or about 2.16 minutes.



```
v > vmax. calculating deceleration. ITER:  276
v > vmax. calculating deceleration. ITER:  275
v > vmax. calculating deceleration. ITER:  274
v > vmax. calculating deceleration. ITER:  273
v > vmax. calculating deceleration. ITER:  272
v > vmax. calculating deceleration. ITER:  271
v > vmax. calculating deceleration. ITER:  270
v > vmax. calculating deceleration. ITER:  269
v > vmax. calculating deceleration. ITER:  268
v > vmax. calculating deceleration. ITER:  267
v > vmax. calculating deceleration. ITER:  266
total time at turn:  11 =  130.31400725596197
time for lnv109a to go around monza:  130.31400725596197 s
time to run code:  691.5177649  seconds
```

Figure 24

Simulating the LNV109a airfoil using the same time step and XFOIL method as for the naca 2412 airfoil, we can see that it actually took this airfoil longer to go around Monza than naca 2412, disproving the hypothesis made before[1]. Although this is not the actual time it would take the F1 car with that airfoil to go around Monza, this simulation tool still accurately models the difference in performance between airfoils. Here, there is a 0.14% difference in performance between the two airfoils.

[1] The direct XFOIL result was used in lieu of the regression analysis to allow for direct comparison

Conclusion

This research demonstrates the role of airfoil selection, testing, and optimization in the overall performance differences between different F1 car rear wing configurations. By leveraging XFOIL's rapid, accurate simulation to dynamically calculate coefficients of lift and drag with respect to the car's instantaneous velocity, this study translates differences in lift and drag to differences in performance and lap time. A traditional aerodynamic analysis of efficiency and performance was disproved using accurate simulated time differences that took into account the car's non-constant velocity and deceleration.

Although there were limitations such as tire temperature and racing lines not being accounted for, computational time, this paper's simulation model accurately calculates relative airfoil performance, an important tool in early design stages where quick performance differences are needed. Future work could incorporate a better friction model, DRS, and an improved racing line motion calculation. Ultimately, this work provides a foundation for more accessible aerodynamic analysis in motorport in both the design and enthusiast fields, offering a cost and time effective solution to refine airfoil designs for rear wing and overall track performance.

References

[1] Drela, Mark, and Harold Youngren. “XFOIL executables.” *MIT*, MIT, 11 December 2000, http://web.mit.edu/drela/Public/web/xfoil/. Accessed 22 November 2024.

[2] NASA Glenn Research Center. “Bernoulli and Newton.” *NASA Glenn Research Center*, 13 November 2024, https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/bernoulli-and-newton/#:~:text=When%20a%20gas%20flows%20over,aerodynamic%20force)%20on%20the%20object. Accessed 22 November 2024.

[3] SKYbrary. “Angle of Attack (AOA).” *SKYbrary*, https://skybrary.aero/articles/angle-attack-aoa. Accessed 22 November 2024.

[4] Fédération Internationale de l’Automobile. *2024 FORMULA 1 TECHNICAL REGULATIONS*. 2024 Formula 1 Technical Regulations. no. 7, Paris, FIA, 31 July 2024, https://www.fia.com/sites/default/files/fia_2024_formula_1_technical_regulations_-_issue_7_-_2024-07-31.pdf. Accessed 22 November 2024.

[5] Moreno, Pablo Hermoso. “Optimize F1 aerodynamic geometries via Design of Experiments and machine learning.” *AWS*, 19 May 2022, https://aws.amazon.com/blogs/machine-learning/optimize-f1-aerodynamic-geometries-via-design-of-experiments-and-machine-learning/. Accessed 22 November 2024.

[6] Woodford, Chris, and Eric James. “Aerodynamics: an introduction.” *Explain that Stuff*, 21 November 2022, https://www.explainthatstuff.com/aerodynamics.html. Accessed 23 November 2024.

[7] Cadence CFD. “Calculating Laminar Flow Reynolds Number and Its Limits.” *Cadence*, https://resources.system-analysis.cadence.com/blog/msa2021-calculating-laminar-flow-reynolds-number-and-its-limits. Accessed 23 November 2024.

[8] Airfoil Tools. “LNV109A - Douglas/Liebeck LNV109A high lift airfoil.” *Airfoil Tools*, http://airfoiltools.com/airfoil/details?airfoil=lnv109a-il. Accessed 23 November 2024.

[9] Edelstein, Stephen. “How F1 engines make 1,000 hp.” *Motor Authority*, 31 December 2023, https://www.motorauthority.com/news/1138958_how-f1-engines-make-1-000-hp. Accessed 23 November 2024.

[10] Formula One World Championship Limited. “FORMULA 1 PIRELLI GRAN PREMIO D’ITALIA 2023.” *F1*, https://www.formula1.com/en/racing/2023/italy/circuit. Accessed 23 November 2024.