Modelaje de alerones con transformaciones de Joukowsky

Matemática Computacional

Armando Apellaniz Solley - 166088
Marco Antonio Chacón Amaro - 165681
Mariana Graciela Martínez Aguilar - 166297

Índice general

Introducción

La Fórmula 1 es la competencia de automovilismo más destacada e importante a nivel internacional, a veces denominada como la "categoría reinad el automovilismo". Se utilizan monoplazas que usan tecnología de punta, mas todos los cambios están regulados por la FIA (Federación Internacional del Automóvil). La velocidad máxima alcanzada hasta el momento es de aproximadamente 370 km/h.

A partir de los 60 se empezó a experimentar con nuevas partes de los automóviles, llamadas alerones para poder alcanzar velocidades mayores. Las partes que nos interesan en este proyecto de un coche de Fórmula 1 son:

Figura 1: Partes de un monoplaza de F1.

Como podemos ver en la Figura 1, el alerón delantero se compone de diversos flaps, los cuales en sí son pequeños alerones juntos. El alerón delantero es fijo mas se puede cambiar durante la carrera si éste se daña. El alerón trasero, también conocido como DRS (por sus siglas en inglés Drag Reduction System) es un sistema que el piloto puede cambiar durante la carrera. El DRS se puede abrir para reducir el arrastre que éste produce y así poder rebasar a algún auto que se encuentre enfrente (ver Figura 2). El DRS puede activarse después de haberse completado dos vueltas de la carrera y bajo ciertas reglas establecidas por la FIA.

Figura 2: DRS cerrado arriba, DRS abierto abajo

Con esto, nos enfrentamos con el problema de cuál será el mejor ángulo para posicionar un alerón y obtener la mayor *downforce* posible. En este proyecto nos vamos a centrar en un alerón común, no nos vamos a enfocar en el sistema DRS o en los flaps delanteros. Vamos a optimizar el ángulo dadas las condiciones para construir este alerón (tamaño, velocidad en la cual se está moviendo el auto y asumiremos la presencia de aire ideal).

Base teórica

Principio de Bernoulli

En 1738, Daniel Bernoulli encontró una relación entre la presión, altura y velocidad de un fluido ideal, este principio describe el comportamiento de este fluido ideal cuando se mueve a lo largo de una línea corriente. Se basa en la idea de que la energía de un fluido consta de tres componentes: energía cinética, energía potencial gravitacional y la energía de flujo (derivada de la presión que ese fluido posee). La ecuación de Bernoulli es la siguiente:

$$\frac{V^2}{2g} + \frac{P}{\rho g} + z = constante$$

Donde

- V es la velocidad del fluido
- g es la aceleración gravitacional
- z es la altura en la dirección de la gravedad
- P es la presión a lo largo de la línea corriente
- ρ es la densidad del fluido

Esta fórmula también se puede ver de la siguiente manera:

$$\frac{V_1^2}{2g} + \frac{P_1}{\rho g} + z = \frac{V_2^2}{2g} + \frac{P_2}{\rho g} + z. \tag{1}$$

Donde los subíndices indican los cambios en velocidad y presión del flujo analizado. Lo que esto nos dice es que para cierto fluido (por ejemplo, aire) con características dadas y densidad constante, si aumentamos su velocidad, su presión baja. De igual manera, si aumentamos su presión, su velocidad bajará.
¿Cómo se relaciona esto con un coche de F1? Desde los 70 (Figura 3), varias escuderías han experimentado con el uso de alerones. Los alerones sirven para crear succión (*downforce*) del coche al piso. Esta fuerza sirve para lograr altas velocidades, mantener al coche estable (especialmente en curvas) y fuerza al coche en la pista. Los alerones funcionan igual que las alas de un avión, pero en sentido contrario (en lugar de crear sustentación, crean *downforce*).

Figura 3: Década de los 70 en F1

La forma del alerón hace que el aire pase más rápido debajo de él que sobre él, lo cual por la ecuación 1, crea una diferencia de presión del aire arriba y abajo del alerón. Dado que la presión debajo del alerón es menor que la presión arriba del alerón, entonces se crea una *downforce*.

Teorema de Kutta-Joukowsky

Para definir la forma del alerón que se tiene que construir se usa el teorema de Kutta-Joukowski, el cual originalmente se basa en sustentación (en inglés, *lift*), no en *downforce*. Este teorema relaciona la sustentación con la circulación.
La circulación se define como la integral línea del vector de velocidad en cada punto con respecto a la superficie en donde se está calculando la circulación. La integral línea es una integral en la cual la función está evaluada en una curva. Esta integral nos ayuda a determinar la circulación alrededor del alerón pues la integral línea en vez de estar evaluada en cualquier curva, está evaluada en una curva cerrada, el círculo (al cual se le va a aplicar la transformación de Joukowski). Dicha integral tiene la forma:

$$\Gamma = \oint_V F\, dl$$

Donde Γ es la circulación, F es el vector de velocidad y dl es el contorno por el cual la circulación va a ser calculada. Si expandimos esta integral tenemos:

$$\Gamma = \oint_0^c F(r(t)) \cdot r'(t)\, dl$$

Donde el intervalo $[0, c]$ indica los extremos de la superficie y $r(t)$ es la parametrización de la superficie. El teorema de Kutta-Joukowski indica lo siguiente:

$$L = -\rho U \Gamma \tag{2}$$

Donde L es la sustentación generada por la superficie estudiada, ρ es la densidad del fluido en el cual se está moviendo esta superficie, Γ es la circulación sobre esta superficie y U es la velocidad del fluido. Este teorema indica que, si hay sustentación, la circulación existe, es negativa y la velocidad del flujo de aire es mayor sobre el ala que debajo de ella. Esto se aplica a aviones. El caso de Fórmula 1 es análogo, la única diferencia es que, en lugar de estudiar un ala, se estudia un alerón y en lugar de generar sustentación, se genera *downforce*. Se estudia un ala vista al revés, se estudia la aerodinámica de un alerón, cuya velocidad de flujo de aire es mayor debajo que sobre él.

Transformación de Joukowsky

Si la velocidad de flujo de aire debe de ser mayor debajo del alerón que sobre él entonces el alerón tiene que ser de una cierta forma para que se cumpla esto. Para poder modelar esta figura se usa un difeomorfismo que se llama transformación de Joukowsky. Un difeomorfismo es una función uno a uno diferenciable e invertible cuya inversa también es diferenciable. A pesar de esto, un difeomorfismo no preserva ángulos ni métrica.
La transformación de Joukowsky se da en el plano complejo. Lo que se hace, para modelar un alerón es tomar un círculo y aplicarle este difeomorfismo. Una característica de este difeomorfismo es que transforma las coordenadas de todos los puntos del plano, por esto podemos estudiar el flujo de aire alrededor de un alerón si estudiamos el flujo de aire alrededor del círculo que se tomó para generar este alerón. Esta transformación tiene la expresión:

$$z' = f(z) = z + \frac{b^2}{z} \tag{3}$$

Donde z son las coordenadas originales del plano complejo, z' son las nuevas coordenadas y b es la intersección del círculo original con el eje real. Dicho círculo inicialmente se ve como en la Figura 4.

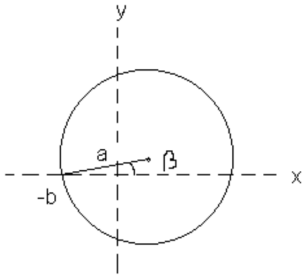

Figura 4: Círculo a transformar

Como podemos ver, cada punto en el plano complejo se puede ver como $z = x + iy$. Si aplicamos esta transformación entonces:

$$f(z) = z + \frac{b^2}{z}$$

$$= x + iy + \frac{b^2}{x + iy}$$

Recordemos que cualquier número complejo también tiene su representación con coordenadas polares: $z = r(\cos\theta + i\sin\theta)$. Por ende:

$$f(z) = r(\cos\theta + i\sin\theta) + \frac{b^2}{r(\cos\theta + i\sin\theta)}$$

$$= r(\cos\theta + i\sin\theta) + \frac{b^2}{r} * \frac{\cos\theta - i\sin\theta}{(\cos\theta + i\sin\theta) * (\cos\theta - i\sin\theta)}$$

$$= r(\cos\theta + i\sin\theta) + \frac{b^2}{r} * \frac{\cos\theta - i\sin\theta}{\cos\theta^2 + \sin\theta^2}$$

$$= r(\cos\theta + i\sin\theta) + \frac{b^2}{r} * (\cos\theta - i\sin\theta)$$

Podemos reorganizar esta expresión para separar las partes imaginarias de las reales:

$$f(z) = r\cos\theta + \frac{b^2}{r}\cos\theta + i(r\sin\theta - \frac{b^2}{r}\sin\theta)$$

Entonces las coordenadas transformadas son:

$$x' = r\cos\theta + \frac{b^2}{r}\cos\theta$$

$$y' = r\sin\theta - \frac{b^2}{r}\sin\theta$$

Si el centro del círculo que vamos a transformar se encuentra en (xc, yc) entonces el alerón generado a partir de la transformación de Joukoswky es una figura no simétrica con una cola filosa, en donde podemos apreciar que esta transformación efectivamente no preserva ángulos.

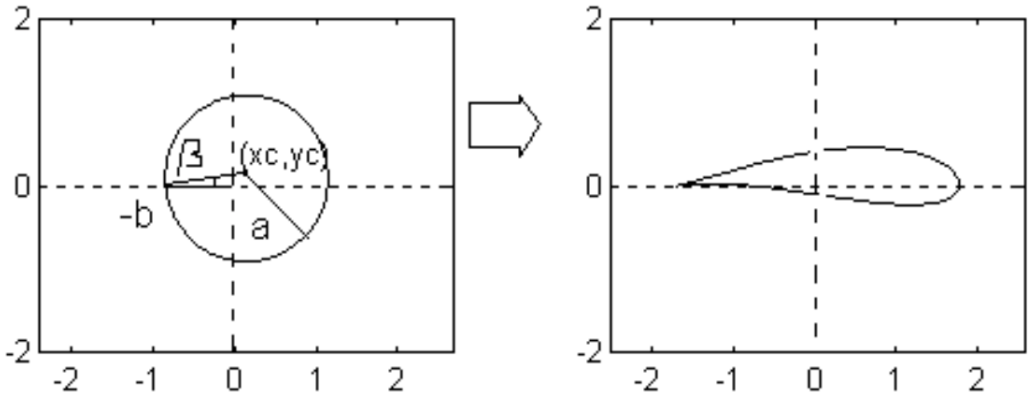

Figura 5: Transformación del círculo con centro en (xc, yc)

Desarrollo

La fuente del código es el archivo **WingOptimization**. El objetivo principal es calcular la fuerza máxima vertical por unidad de distancia (N/m) que puede entregar un círculo transformado en alerón con Joukowsky de dos metros de radio y con un centro en el plano cartesiano de $(0.5, 0.5)$ a un cierto ángulo de rotación y con viento ideal con una velocidad de $55m/s$. Un objetivo secundario es mostrar el resultado del alerón transformado. Hay dos opciones para calcular la fuerza máxima: rotar el ala o rotar el viento y mantener el ala estática. Optamos por rotar el viento y mantener el ala estática por facilidad en el proceso de graficación.

Búsqueda de la sección dorada

Para optimizar el ángulo se tuvo que encontrar el máximo de la fuerza del componente y. Para hacer esto se ocupó el método de Búsqueda de la sección dorada. Dicho algoritmo se describe de la siguiente manera:

1. En un intervalo $[a, b]$ se eligen dos puntos c y d tales que se cumpla:

   $c = ■ * a + (1 - ■) * b$
   $d = (1 - ■) * a + ■ * b$

   Donde ■=0.618.

2. Se evalúan c y d en la función. Si $f(c) > f(d)$ entonces el nuevo intervalo es $[c, b]$ y $a = c$ de lo contrario el nuevo intervalo es $[a, d]$ y $b = d$.

3. Se itera el proceso hasta que $a = b$.

En el caso de este algoritmo, se utiliza una tolerancia de 1^{-10}.

Resolución del problema usando técnicas numéricas

El algoritmo generado para optimizar el ángulo en el cual el alerón debe ser puesto en el coche es:

1. Se Generaró un vector con los ángulos a considerar:

   ```
   x = linspace(0, 180, points);
   ```

   Donde *points* tiene un valor de 60.

2. A través de la función **totalPerpencicularForce**, calculamos la fuerza total hacia arriba a través del Teorema de Kutta-Joukowski (2) en cada ángulo. Cabe destacar que jamás se rota el ala. Como la fuerza que se obtiene es con el ala horizontal y el aire con un ángulo, si queremos

considerar el viento chocando con el ala horizontal y el ala rotada, es necesario descomponer a la fuerza total perpendicular del ala obtenida en su vector vertical de fuerza para obtener la fuerza útil.

```
rad=x(i)*pi/180;
fullForce(i) = totalPerpendicularForce(wSp, x(i), radius);
yForce(i) = cos(rad)*fullForce(i);
```

Cabe mencionar que los cálculos son hechos con ángulos en radianes.

3. Con la fuerza vertical obtenida en cada ángulo interpolamos con el Método de Newton una función a través de la función **newtonAdjustment** que a la vez utiliza la función **newtonCoeficients** (los 60 puntos se eligieron por cuestiones de tiempo y debido a que agregar más puntos generaba una función con cambios de valores del orden de 1^{15}). Dicha parte del código se ve así:

```
n = newtonAdjustment(x, yForce, points);
```

4. Se graficaron tanto los puntos de la fuerza total como la función interpolada de la fuerza vertical.

```
plot(x, fullForce);
plot(x,yForce,'*',y,nEval);
```

Estas gráficas son las siguientes:

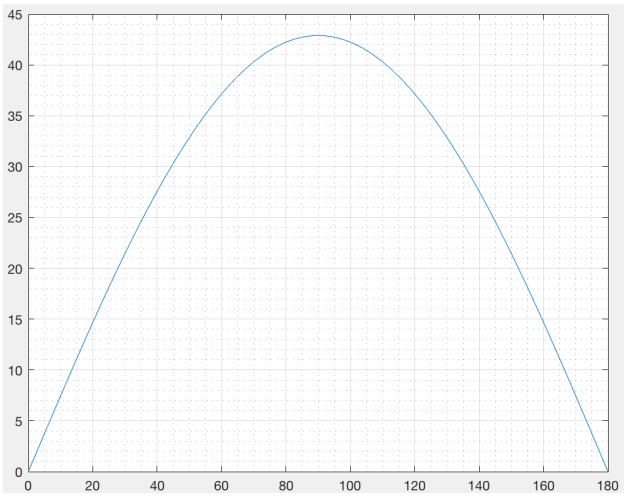

Figura 6: Fuerza total.

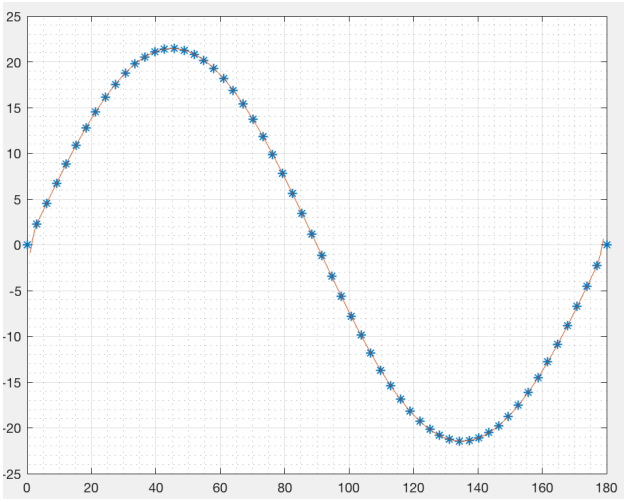

Figura 7: Fuerza total vertical.

Como podemos ver, sí es necesario descomponer la fuerza total con sus componentes x y y para poder optimizar el componente y.

5. Se encontró el valor máximo de la fuerza vertical. Se utilizó la función **gSS** que utiliza el método de Búsqueda de la sección dorada encontrar el máximo.

   maxP=gSS(auxN,5,100)

   Como previamente fue mencionado, el método funciona solamente si la función es unimodal (solo un máximo), lo cual sí cumple esta interpolación.

6. Finalmente, dado el punto con el ángulo que ofrece la fuerza vertical máxima, mostramos con la función **airfoilPlot** el resultado del alerón transformado y además se muestra el flujo potencial de aire.

Conclusiones

Después de dicho análisis se llegó a la conclusión que la fuerza en el componente y se alcanza cuando el ángulo de inclinación es 45. Haciendo los cálculos pertinentes, la fuerza por metro (N/m) del alerón usando la transformación de Joukowsky (3) es 21.4461.
El círculo original optimo con las líneas de flujo (centro en $(0.5, 0.5)$ y radio 2) es el siguiente:

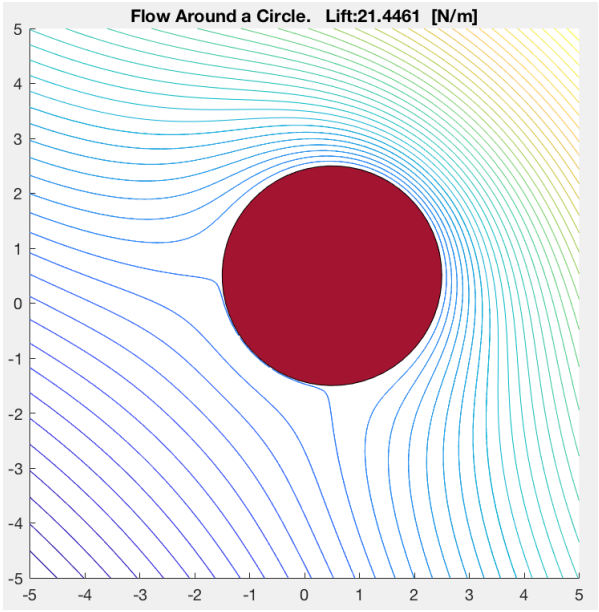

Figura 8: Círculo original.

Y su correspondiente transformación es la siguiente:

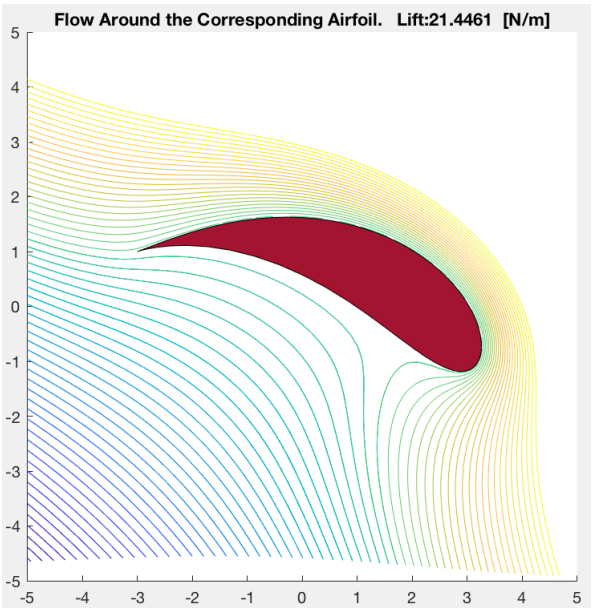

Figura 9: Transformación del círculo usando la transformación de Joukowsky.

Como podemos notar, la figura que obtuvimos es un ala, no un alerón. Como se mencionó al inicio

de este trabajo, el teorema de Kutta-Joukowsky se aplica a aviones y se genera sustentación, no *downforce*. El caso de alerones para monoplazas de Fórmula 1 es análogo, se construye de igual manera, lo que cambia es la forma de montar esta figura en el coche. En lugar de montar la Figura 9 tal cual se aprecia en dicha gráfica, se monta de esta manera:

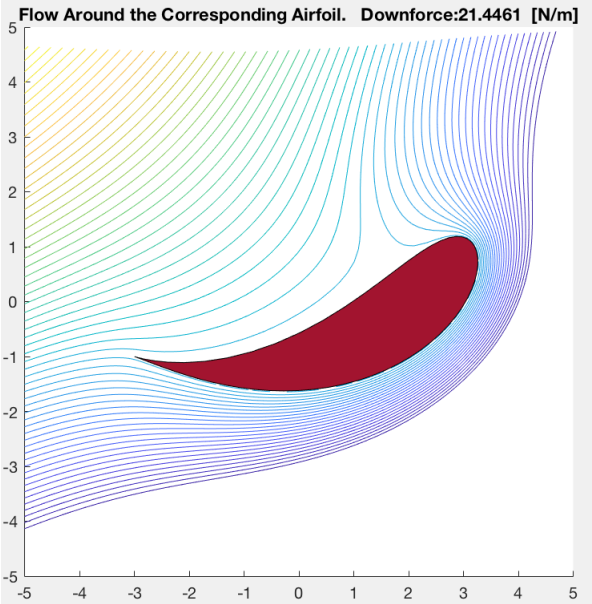

Figura 10: Aleron óptimo.

Referencias

[1] Racing Physics" *NASA*. 9.Mayo.18.
`https://www.nas.nasa.gov/About/Education/Racecar/physics.html`

[2] Johnson, Thomas. Çonformal Mapping in Wing Aerodynamics", (2013): 16.

[3] "Numerical Methods Lecture 6 - Optimization" *Engineering School of Sustainable Infrastructure and Evironment*. 9.Mayo.18.
`https://www.essie.ufl.edu/~kgurl/Classes/Lect3421/NM6_optim_s02.pdf`

[4] Seljak, Gregor. *Race Car Aerodynamics*. 2008.

[5] "Lift forces" 7.Mayo.18.
`http://www1.maths.leeds.ac.uk/~kersale/2620/Notes/chapter_8.pdf`

[6] "Line Integral" *Brilliant*. 11.Mayo.18.
`https://brilliant.org/wiki/line-integral/`

[7] "Lecture 4: Circulation and Vorticity" *Earth System Science*. 11.Mayo.18.
`https://www.ess.uci.edu/~yu/class/ess227/lecture.4.vorticity.all.pdf`

[8] "Joukowski Airfoil Transformation"1.Mayo.18.
`https://la.mathworks.com/matlabcentral/fileexchange/8870-joukowski-airfoil-transformation`

Imágenes usadas en este trabajo por orden de aparición en el mismo (las imágenes no citadas fueron generadas por los mismos autores de este trabajo):

[9] "F1 2018 Testing Day One: Red Bull fastest from Mercedes and Ferrari" *SkySports*. 6.Mayo.18
http://www.skysports.com/f1/news/12433/11268527/f1-2018-testing-day-one-red-bull-fastest-from-mercedes-and-ferrari

[10] "Martini to end Williams sponsorship in 2018" *Planet F1*. 8.Mayo.18
`http://www.planetf1.com/news/martini-to-end-williams-sponsorship/`

[11] "Drag Reduction System" *The Formula 1 Wiki*. 8.Mayo.18
`http://f1.wikia.com/wiki/Drag_Reduction_System`

[12] , "¿Qué impide que un Fórmula Uno salga volando?" *Red Bull*. 8.Mayo.18
`https://www.redbull.com/co-es/formula-uno-carga-resistencia-aerodinamica-alerones-historia`

Apéndice

En esta sección se encuentra el código usado para el análisis de este trabajo.

WingOptimization

```
%First we find the force per unit of distance (N/m) of the perimeter of the
%wing

%Number of points to adjust
points = 60;

%Inicial variables
fullForce = 1:points;
yForce = 1:points;

%Degrees to consider
x = linspace(0, 180, points);

%Speed of the wind(m/s)
wSp=55;

%Center of the circle
xC=0.5;
yC=0.5;

%Radius of the circle
radius =2;

%Obtatinig values for the downforce in specific points
for i=1:points
rad=x(i)*pi/180;
fullForce(i) = totalPerpendicularForce(wSp, x(i), radius);
yForce(i) = cos(rad)*fullForce(i);
end

%Interpoling a Newton Adjustemt bases on the yForce vector
n = newtonAdjustment(x, yForce, points);

%Evaluating points in the resluting Newton funtion
y = linspace(1,179,1000);
nEval = 1:1000;
for h=1:1000
nEval(h)=n(y(h));
end

% %Graphing FullForce vector
```

```
figure(1)
plot(x, fullForce);
grid on
grid minor

% %Comparing yVector vs Newton function
figure(2)
plot(x,yForce,'*',y,nEval);
grid on
grid minor

%Finding the real maximum point of the Newton function.
%Therefore finding the maximum force per unit(N/m)
auxN=@(t)-n(t);
maxP=gSS(auxN,5,100);
maxF=n(maxP);

%Showing the wing
airfoilPlot(wSp,maxP, xC, yC, radius, maxF)
```

totalPerpendicularForce

```
function [L] = totalPerpendicularForce(v_inf,theta, r)
%Return the total force per unit of distance in the perimeter of a circle of radius r;

v = v_inf/v_inf;
theta = theta*pi/180;

% FLUID PARAMETER
rho = 1.225;

% CIRCULATION
beta = (theta);
k = 2*r*v*sin(beta);
Gamma = k/(2*pi); %CIRCULATION

% KUTTA JOUKOWSKI THEOREM
L = v_inf*rho*Gamma;

end
```

newtonCoeficients

```
function [ c ] = newtonCoeficients( x,c,n )
%UNTITLED Summary of this function goes here
```

```
%   Detailed explanation goes here
for j = 2:n
for i=n:-1:j
c(i)=(c(i)-c(i-1))/(x(i)-x(i+1-j));
end
end

end
```

newtonAdjustment

```
function [ n ] = newtonAdjustment( x,c,points)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

c=newtonCoeficients(x,c,points);


n = @(t) c(1);

for i=1:points-1
nT=@(t) c(i+1);
for j=1:i
nT=@(t) nT(t)*(t-x(j));
end
n=@(t) n(t)+nT(t);
end

end
```

gSS

```
function [ min ] = gSS( f,a,b)
%Golden Section Search
%   Function to find minimum values.
golRat = (sqrt(5) + 1) / 2;
tol=1e-10;
c = b - ((b - a) / golRat);
d = a + ((b - a) /golRat) ;


while abs(c - d) > tol
if f(c) < f(d)
b = d;
else
```

```
a = c;
end
c = b - ((b - a) /golRat);
d = a + ((b - a) / golRat);
end
min =(b + a) / 2;
end
```

airfoilPlot

```
function [force] = airfoilPlot(v_inf,theta, s_x, s_y, r, force)
v = v_inf/v_inf;
theta = theta*pi/180;
s = s_x + 1i*s_y;

% TRANSFORMATION PARAMETER
lambda = r-s;
% CIRCULATION
beta = (theta);
k = 2*r*v*sin(beta);

%COMPLEX ASYMPTOTIC SPEED
w = v * exp(1i*theta);

%TOLLERANCE
toll = +5e-2;

% GENERATING MESH
x = meshgrid(-5:.1:5);
y = x';

% COMPLEX PLANE
z = x + 1i*y;

% Inside-circle points are Excluded!
for a = 1:length(x)
for b = 1:length(y)
if abs(z(a,b)-s) <=  r - toll
z(a,b) = NaN;
end
end
end

% AERODYNAMIC POTENTIAL
f = w*(z) + (v*exp(-1i*theta)*r^2)./(z-s) + 1i*k*log(z-s);
```

```
% JOUKOWSKI TRANSFORMATION,
J = z+lambda^2./z;

%GRAPHIC - Circle and Joukowski Airfoil
angle = 0:.1:2*pi;
z_circle = r*(cos(angle)+1i*sin(angle)) + s;
z_airfoil = z_circle+lambda^2./z_circle;

L_str = num2str(force);

%PLOTTING SOLUTION
figure(3)
hold on
contour(real(z),imag(z),imag(f),50)
fill(real(z_circle),imag(z_circle),[0.6350, 0.0780, 0.1840] )
axis equal
axis([-5 5 -5 5])
title(strcat('Flow Around a Circle.   Lift:  ',L_str,'  [N/m]'));


figure(4)
hold on
contour(real(J),-imag(J),-imag(f),-5:.1:5)
fill(real(z_airfoil),-imag(z_airfoil),[0.6350, 0.0780, 0.1840])
axis equal
axis([-5 5 -5 5])
title(strcat('Flow Around the Corresponding Airfoil.   Downforce:  ',L_str,'  [N/m]'));
end
```