Homework 2: Horseshoe Vortex Method

Isaac Gibert, Ian Martorell, Sara Piñeiro, Esteban Ruiz, and Eduard Sulé

220024 - Aerodynamics, UPC ESEIAAT

Dated: January 2016

1. Efecto del alargamiento en el $c_{L,\alpha}$

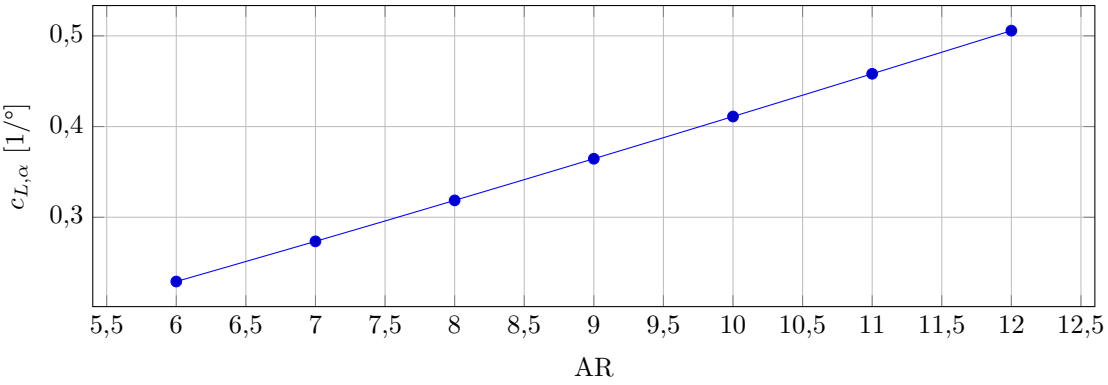

Figura 1: Pendiente de sustentación en función del alargamiento

Vemos como aumenta la pendiente de sustentación de forma proporcional con el alargamiento, aunque sabemos que según la teoría de perfiles delgados ésta debería tender a 2π, ya que un alargamiento mayor se aproxima cada ves más a una placa plana. Sin embargo, para un alargamiento menor obtenemos un ángulo de sustentación máximo mayor. Si escogemos un valor de alargamiento igual a 6 y lo comparamos con la figura del libro Katz-Plotkin podemos ver que no nos asemejamos a los resultados experimentales, aunque vemos también que en la misma figura aparece una línea recta que hace referencia a una tendencia lineal de la pendiente de sustentación que nos hace pensar que al aumentar demasiado el alargamiento, nuestra aproximación teórica (con el código del programa) se vuelve inexacta, ya que debería tender a 2π, según la teoría de perfiles delgados.

2. Efecto de la flecha en el $c_{L,\alpha}$ y el x_{ac}

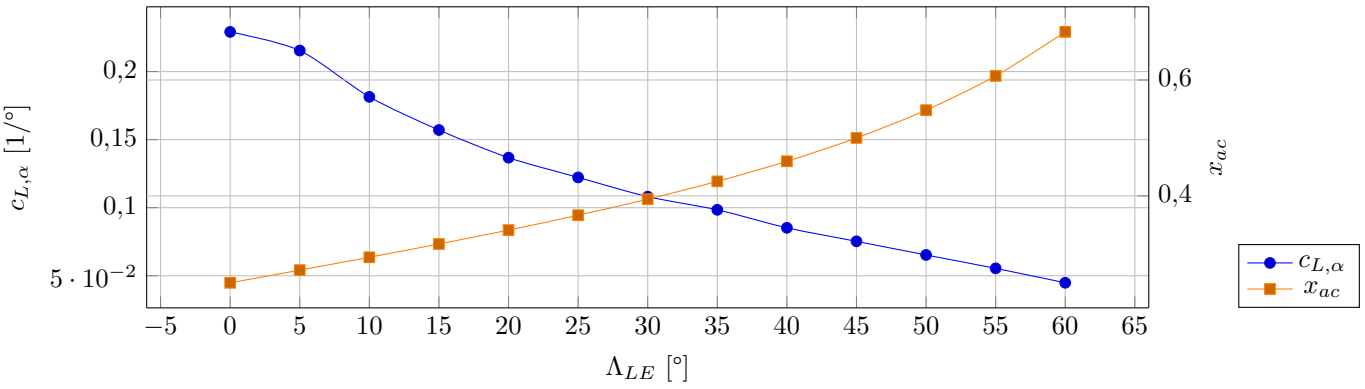

Figura 2: Pendiente de sustentación y centro aerodinámico en función de la flecha

De la siguiente gráfica podemos apreciar como un aumento del ángulo de flecha deriva en una disminución del coeficiente de sustentación total dela ala, provocando una disminución de la eficiencia aerodinámica (CL/CD).

Podemos concluir que, a bajas velocidades, aunque la flecha disminuye su capacidad a nivel aerodinámico, se consigue una mayor estabilidad, reduciendo la carga estructural.Además, para altas velocidades, aumenta el Mach crítico, disminuyendo los efectos de perturbaciones como ondas de choque, que reducen notablemente las prestaciones aerodinámicas del ala.

Finalmente, podemos apreciar que el centro aerodinámico en el ala se retrasa a medida que aumenta la flecha, por simples relaciones geométricas.

3. Efecto del estrechamiento en la distribución de sustentación local

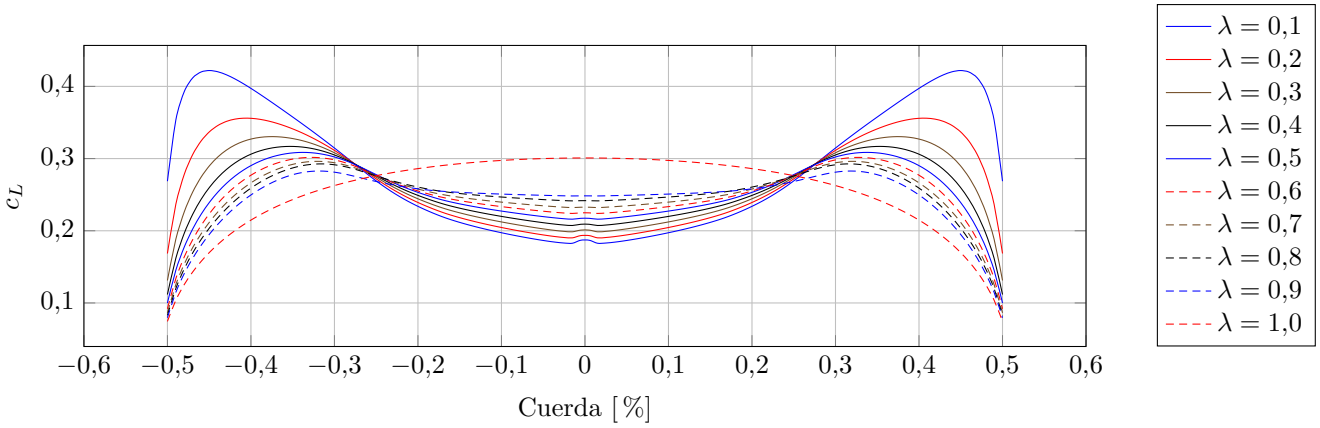

Figura 3: Distribución de la sustentación para diferentes valores de estrechamiento

Según la teoría de perfiles delgados, un ala rectangular produce una distribución de coeficiente de sustentación elíptica, mientras que para un ala cuya planta sea elíptica, la distribución del coeficiente resulta una constante.

Observando esta gráfica podemos comprobar de forma aproximada con un estrechamiento igual a la unidad, la distribución de CL es elíptica, mientras que a medida que lo disminuimos, la carga de sustentación disminuye en las puntas lo que provoca que el CL aumente en dichas zonas, como se puede apreciar en el siguiente gráfico.

4. Efecto del alargamiento y la flecha en el factor de eficiencia de Oswald

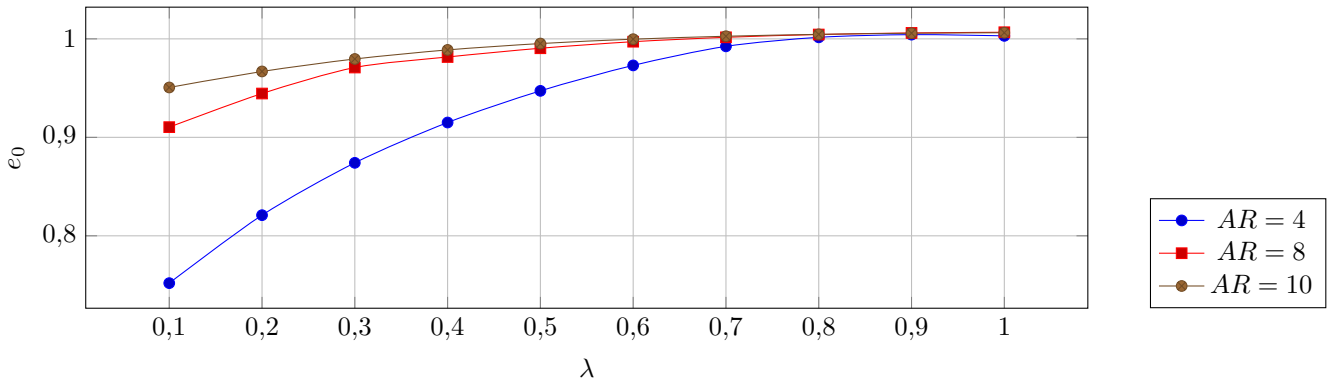

Figura 4: Efecto del alargamiento en el factor de eficiencia de Oswald

El factor de Oswald puede definirse como factor de eficiencia ligado a la forma en planta del ala. Cuanto mayor es este factor, menor es el ángulo inducido. Para un ala elíptica, el factor de Oswald es el máximo (igual a uno), es decir, el ala elíptica es la que posee un menos coeficiente de resistencia inducida. En la gráfica podemos observar que, en primer lugar, para estrechamientos mayores, el factor de Oswald tiende a uno, concordando con lo dicho anteriormente. Además, podemos ver que el aumento del alargamiento provoca que el factor tienda más rápido a uno, és decir, se cumple que para un alargamiento que tienda a infinito, la pendiente de sustentación se acerca a la ideal, la del perfil.

Visualmente, un alargamiento infinito puede relacionarse con un hilo de torbellino, lo que nos lleva a la teoría de perfiles delgados, donde obtenemos una pendiente de sustentación igual a 2pi.

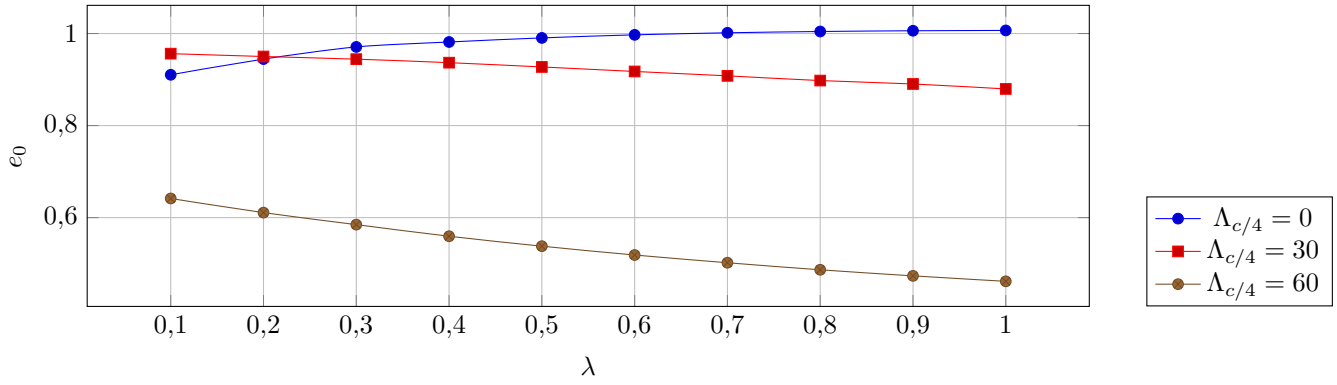

Figura 5: Efecto de la flecha en el factor de eficiencia de Oswald

De la figura podemos apreciar que a medida que se aumenta la flecha, el ala tiene un menor factor de Oswald, eso es debido a que, como se ha comentado anteriormente, la flecha reduce la eficiencia aerodinámica. Por lo tanto, deducimos que el factor de Oswald disminuirá a medida que disminuye la eficiciencia aerodinámica.

5. Distribución básica y adicional de sustentación

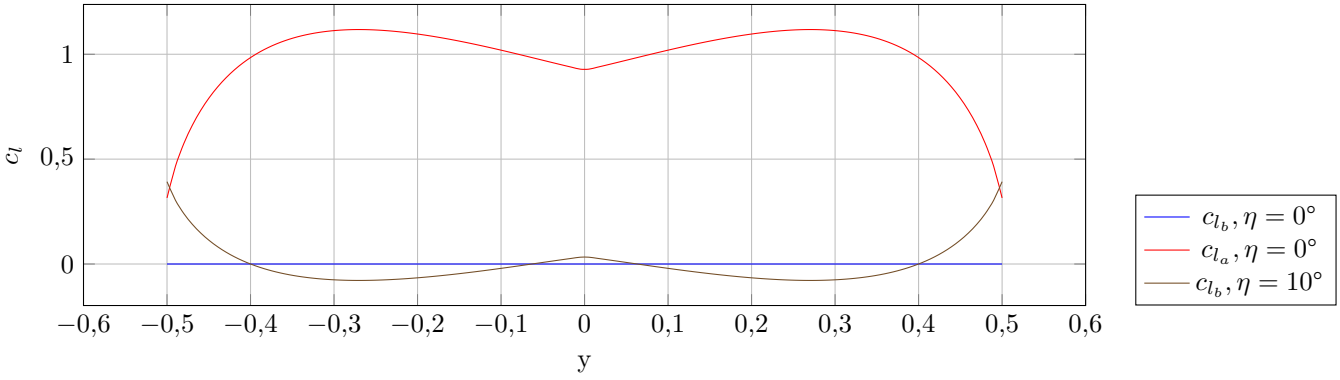

Figura 6: Distribuciones de c_l con y sin flap

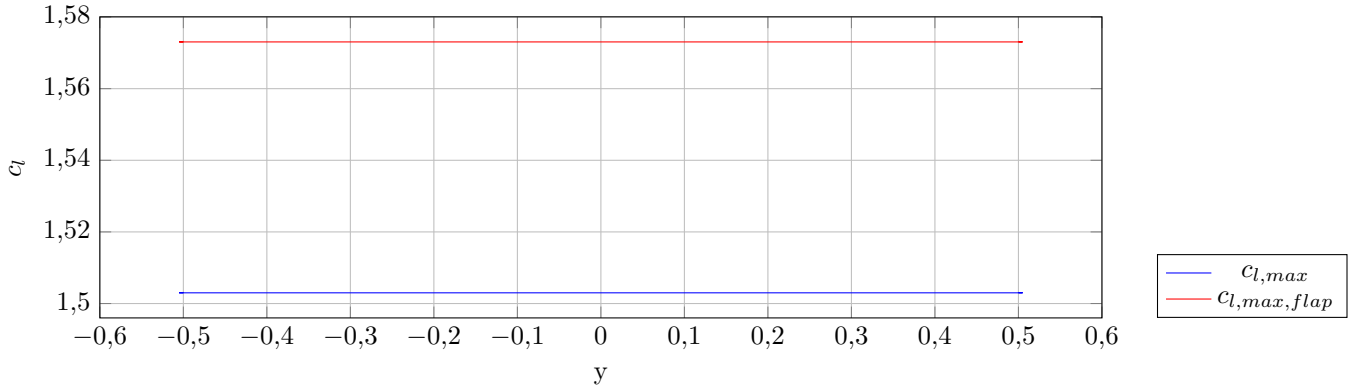

Figura 7: $c_{l,max}$ con y sin flap

El siguiente gráfico pretende mostrar el coeficiente de sustentación básico y adicional, que son dos elementos fundamentales para el estudio de ala finita. En el primer caso hemos calculado hemos calculado estos coeficientes para una configuración

sin flap y con flap para un perfil NACA 2412. El Cl adicional no varía por el hecho de tener flap, ya que hace referencia a la planta de ala, por lo tanto se mantiene constante para la configuración con flap y sin flap.

Podemos apreciar también, que el área contenida por el CL adicional resulta aproximadamente uno, hecho que concuerda con la teoría estudiada en clase, donde la integral de este coeficiente por la distribución de cuerda a lo largo de la envergadura debe ser uno.

Refiriéndonos ahora al coeficiente de sustentación básico, sabemos que este depende de factores como la torsión, el ángulo de sustentación nula, etc... Este coeficiente encierrra un área aproximadamente igual a cero. Fijándonos en el coeficiente de sustentación básico sin flap, éste está sometido a un ángulo de ataque igual a cero, lo que implica una circulación igual a cero, resultando un coeficiente de sustentación básico igual a cero, demostrando así que encierra un área nula. En contraste con el Clbásico sin flap, el Clbásico con deflexión del flap presenta unas perturbaciones en las distribución del Cl local debidas a una variación del ángulo de sustentación nula. A pesar de esto, sigue cumpliéndose que el área que encierra es igual a uno. Por otra parte, nos piden calcular el coeficiente de momento libre y el centro aerodinámico, cuyos valores obtenidos son -0.076 y 0.28237, respectivamente.

Por último, en la siguiente figura mostramos los valores de Cl máximos tanto para el ala sin flap como con flap. Dichos valores fueron escogidos como los mínimos entre los Cl máximos de cada sección. Una vez calculados, podemos hallar los ángulos geométricos máximos, con los que se alcanzarían tales coeficientes de sustentación. Haciendo referencia a los ángulos, los valores que se obtienen son, respectivamente, 12.302 y 18.467º.

Desde el punto de vista estructural, hemos comentado anteriormente que un mayor ángulo de flecha disminuye la eficiencia aerodinámica, aumenta la estabilidad, y disminuye cargas estructurales en el encastre.

6. Apéndice: Código

```
function [ midPoints, controlPoints, bounded_nodes, trailing_nodes, panelAngles, panelAreas, panelChords ] =
    wing_discretization(aspectRatio, taperRatio, quarterChordSweep, angleOfAttack, wingTipTwist, nPanels)
  %wing_discretization: Returns a vector of points along the quarter chord line of the
  %wing, each centered along the y axis of every panel. Also returns the control points
  %at 3c/4 and the angle of every panel after applying angle of attack and twist.
  midPoints = zeros(nPanels, 3);
  controlPoints = zeros(nPanels, 3);
  bounded_nodes = zeros(nPanels+1, 3);
  trailing_nodes = zeros(nPanels+1, 3);
  panelAngles = zeros(nPanels, 1);
  panelAreas = zeros(nPanels, 1);
  panelChords = zeros(nPanels, 1);
  %Compute y, which is distributed linearly
  panelWidth = 1/nPanels;
  lastY = 0.5 - panelWidth/2;
  midPoints(:, 2) = linspace(-lastY, lastY, nPanels);
  controlPoints(:, 2) = linspace(-lastY, lastY, nPanels);
  bounded_nodes(:, 2) = linspace(-0.5, 0.5, nPanels+1);
  trailing_nodes(:, 2) = linspace(-0.5, 0.5, nPanels+1);
  %Compute some needed constants
  surfaceArea = 1/aspectRatio;
  chordRoot = 2*surfaceArea/(1+taperRatio);
  chordTip = taperRatio*chordRoot;
  %Find equation of sweep(y) = x(y) for quarter chord points
  sweepSlope = tand(90-quarterChordSweep);
  sweepOrd = -sweepSlope*0.25*chordRoot;
  %Find equation for twist, which has a zero y-intercept
  twistSlope = wingTipTwist/0.5;
  for i = 1:nPanels
    panelChords(i) = chordRoot + (chordTip - chordRoot) / 0.5 * abs(midPoints(i, 2));
    panelAreas(i) = panelChords(i)*panelWidth;
    panelAngles(i) = twistSlope * abs(midPoints(i, 2)) + angleOfAttack;
    %Calculate x position, if sweep is 90 degrees, the slope will be infinity
    if isinf(sweepSlope)
      midPoints(i, 1) = 0.25*panelChords(i);
      bounded_nodes(i, 1) = 0.25*panelChords(i);
    else
      if midPoints(i, 2) > 0
        midPoints(i, 1) = (midPoints(i, 2) - sweepOrd) / sweepSlope;
        bounded_nodes(i, 1) = (bounded_nodes(i, 2) - sweepOrd) / sweepSlope;
      else
        midPoints(i, 1) = (midPoints(i, 2) + sweepOrd) / -sweepSlope;
        bounded_nodes(i, 1) = (bounded_nodes(i, 2) + sweepOrd) / -sweepSlope;
      end
    end
    trailing_nodes(i, 1) = bounded_nodes(i, 1) + 20;
    %Correction due to panel angle
```

```
    midPoints(i, 1) = midPoints(i, 1) + panelChords(i) * (1 - cosd(panelAngles(i)));
    %Calculate z position
    midPoints(i, 3) = sind(panelAngles(i)) * panelChords(i);
    %Calculate control point position
    controlPoints(i, 1) = midPoints(i, 1) + panelChords(i)/2 + (1 - cosd(panelAngles(i)))/4;
    controlPoints(i, 3) = sind(panelAngles(i)) * panelChords(i)/4;
  end
  %Last bounded and trailing node
  if isinf(sweepSlope)
    bounded_nodes(nPanels+1, 1) = 0.25*panelChords(i);
    trailing_nodes(nPanels+1, 1) = bounded_nodes(nPanels+1, 1) + 20;
  else
    bounded_nodes(nPanels+1, 1) = (bounded_nodes(nPanels+1, 2) - sweepOrd) / sweepSlope;
    trailing_nodes(nPanels+1, 1) = bounded_nodes(nPanels+1, 1) + 20;
  end
end
```

wing_discretization.m

```
function coordinates = rectangular_horseshoe(midPoint, panelAngle, nPanels)
  panelWidth = 1 / nPanels;
  coordinates = zeros(4, 3);
  %Points b, c, a, d
  coordinates(2, :) = [ midPoint(1) midPoint(2)-panelWidth/2 midPoint(3) ];
  coordinates(3, :) = [ midPoint(1) midPoint(2)+panelWidth/2 midPoint(3) ];
  coordinates(1, :) = [ midPoint(1)+20 coordinates(2,2) -sind(panelAngle)*20 ];
  coordinates(4, :) = [ midPoint(1)+20 coordinates(3,2) -sind(panelAngle)*20 ];
end
```

rectangular_horseshoe.m

```
function inducedVelocity =  compute_induced_velocity(xa, xb, xp, circulation)
  %Cross products
  x = (xp(2)-xa(2))*(xp(3)-xb(3)) - (xp(3)-xa(3))*(xp(2)-xb(2));
  y = -(xp(1)-xa(1))*(xp(3)-xb(3)) + (xp(3)-xa(3))*(xp(1)-xb(1));
  z = (xp(1)-xa(1))*(xp(2)-xb(2)) - (xp(2)-xa(2))*(xp(1)-xb(1));
  d = x*x + y*y + z*z;
  r1 = sqrt((xp(1)-xa(1))*(xp(1)-xa(1)) + (xp(2)-xa(2))*(xp(2)-xa(2)) + (xp(3)-xa(3))*(xp(3)-xa(3)));
  r2 = sqrt((xp(1)-xb(1))*(xp(1)-xb(1)) + (xp(2)-xb(2))*(xp(2)-xb(2)) + (xp(3)-xb(3))*(xp(3)-xb(3)));
  %We set the induced velocity to zero if r1, r2 or their cross product is less
  %than a small constant, to avoid dividing by zero
  if d<(10^-6) || r2<(10^-6) || r1<(10^-6)
    inducedVelocity = [ 0; 0; 0 ];
  else
    ror1 = (xb(1)-xa(1))*(xp(1)-xa(1)) + (xb(2)-xa(2))*(xp(2)-xa(2)) + (xb(3)-xa(3))*(xp(3)-xa(3));
    ror2 = (xb(1)-xa(1))*(xp(1)-xb(1)) + (xb(2)-xa(2))*(xp(2)-xb(2)) + (xb(3)-xa(3))*(xp(3)-xb(3));
    com = (circulation/(4*pi*d))*((ror1/r1)-(ror2/r2));
    inducedVelocity = [x*com; y*com; z*com];
  end
end
```

compute_induced_velocity.m

```
function angle = quarter_chord_sweep(leadingEdgeSweep, aspectRatio, taperRatio)
  angle = atand(tand(leadingEdgeSweep)-(1/aspectRatio)*((1-taperRatio)/(1+taperRatio)));
end
```

quarter_chord_sweep.m

```
function [ cL, cLY, cDi, alpha_i ] = HVM(aspectRatio, taperRatio, quarterChordSweep, angleOfAttack,
    wingTipTwist, horseshoeShape, nPanels)
  %HVM: Computes the lift coefficient of a wing using the Horseshoe Vortex Method
  %horseshoeShape: can be 'rectangular' or 'trapezoidal'
  density = 1.25;
  freestreamVelocity = [ 1 0 0 ];
  %Perform wing discretization
  [ midPoints, controlPoints, bounded_nodes, trailing_nodes, panelAngles, panelAreas ] =
    wing_discretization(aspectRatio, taperRatio, quarterChordSweep, angleOfAttack, wingTipTwist, nPanels);
  %Initialize variables
  influenceCoefficients = zeros(nPanels, nPanels);
  RHS = zeros(nPanels, 1);
  for i = 1:nPanels
    midPoint = [ midPoints(i, 1) midPoints(i, 2) midPoints(i, 3) ];
    normalUnitVector = [ sind(panelAngles(i)) 0 cosd(panelAngles(i)) ];
    for j = 1:nPanels
      midPoint = [ midPoints(j, 1) midPoints(j, 2) midPoints(j, 3) ];
```

```
      if strcmp(horseshoeShape, 'rectangular')
        horseshoe = rectangular_horseshoe(midPoint, panelAngles(j), nPanels);
      else
        horseshoe = [ trailing_nodes(j,:); bounded_nodes(j,:); bounded_nodes(j+1,:); trailing_nodes(j+1,:)];
      end
      inducedVelocity = zeros(3,1);
      for k = 1:3
        inducedVelocity = inducedVelocity + compute_induced_velocity(horseshoe(k,:), horseshoe(k+1,:),
    controlPoints(i,:), 1);
      end
      influenceCoefficients(i, j) = dot(inducedVelocity, normalUnitVector);
    end
    RHS(i) = -dot(freestreamVelocity, normalUnitVector);
  end
  circulation = influenceCoefficients \ RHS;
  %Compute lift
  lift = zeros(nPanels, 1);
  for i = 1:nPanels
    lift(i) = density * freestreamVelocity(1) * circulation(i) / nPanels;
  end
  wingLift = sum(lift);
  %Compute lift coefficient
  cL = 2 / freestreamVelocity(1) * aspectRatio * sum(circulation/nPanels);
  %Compute local lift distribution
  cLY = 2*circulation./panelAreas/nPanels/freestreamVelocity(1);
  %Compute moment
  momentLE = zeros(nPanels);
  for i = 1:nPanels
    momentLE(i) = lift(i) * midPoints(i, 1) * cosd(panelAngles(i));
  end
  chordRoot = 2/aspectRatio/(1+taperRatio);
  geometricChord = (2/3)*chordRoot*((1+taperRatio+taperRatio^2)/(1+taperRatio));
  cMLE = ((-2)/(freestreamVelocity(1)/aspectRatio*geometricChord))*sum(momentLE);
  [ alpha_i local_drag cDi ] = compute_cdi(nPanels, midPoints, bounded_nodes, trailing_nodes, panelAngles,
    circulation, 1/aspectRatio);
end
```

HVM.m

```
aspectRatios = 6:12;
taperRatio = 1;
quarterChordSweep = 0;
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
%Initialize output vector
cLAlphas = [];
for aspectRatio = aspectRatios
  %Compute cL for -2 and 2 degrees so we can draw a line
  [ cL1 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, -2, wingTipTwist, horseshoeShape, nPanels);
  [ cL2 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, 2, wingTipTwist, horseshoeShape, nPanels);
  cLAlphas = [ cLAlphas; (cL2-cL1)/4 ];
end

csvwrite('data/hw2_1.csv', [ aspectRatios' cLAlphas ]);
```

hw2_1.m

```
aspectRatio = 6;
taperRatio = 1;
leadingEdgeSweeps = 0:5:60;
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
%Initialize output vectors
cLAlphas = [];
aerodynamicCenters = [];
for leadingEdgeSweep = leadingEdgeSweeps
  %Although unnecessary for a unitary taper ratio, we calculate the quarter chord sweep angle
  quarterChordSweep = quarter_chord_sweep(leadingEdgeSweep, aspectRatio, taperRatio);
  %Compute cL for -2 and 2 degrees so we can draw a line
  [ cL1 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, -2, wingTipTwist, horseshoeShape, nPanels);
  [ cL2 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, 2, wingTipTwist, horseshoeShape, nPanels);
  cLAlphas = [ cLAlphas; (cL2-cL1)/4 ];
  aerodynamicCenters = [ aerodynamicCenters; 0.25 +
    tand(quarterChordSweep)/6*(1+2*taperRatio)/(1+taperRatio)];
end
```

```
csvwrite('data/hw2_2.csv', [ leadingEdgeSweeps' cLAlphas aerodynamicCenters ]);
```

hw2_2.m

```
aspectRatio = 5;
taperRatios = 0.1:0.1:1;
quarterChordSweep = 0;
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
% Initialize output vectors
wingSpan = linspace(-0.5, 0.5, nPanels)';
cLYs = [];
for taperRatio = taperRatios
  % Compute cL for -2 and 2 degrees so we can draw a line
  [ cL1 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, -2, wingTipTwist, horseshoeShape, nPanels);
  [ cL2 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, 2, wingTipTwist, horseshoeShape, nPanels);
  cLAlpha = (cL2-cL1)/4;
  % Find angle of attack for a 0.25 lift coefficient
  alpha = 0.25/cLAlpha;
  % Find local lift distribution for computed angle of attack
  [ cL, cLY ] = HVM(aspectRatio, taperRatio, quarterChordSweep, alpha, wingTipTwist, horseshoeShape,
    nPanels);
  cLYs = [ cLYs cLY ];
end

csvwrite('data/hw2_3.csv', [ [ -1 taperRatios ]; wingSpan cLYs ]);
```

hw2_3.m

```
aspectRatios = [ 4 8 10 ];
taperRatios = 0.1:0.1:1;
quarterChordSweep = 0;
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
% Initialize output vector
oswaldFactors = [];
for aspectRatio = aspectRatios
  oswaldFactorsColumn = [];
  for taperRatio = taperRatios
    [ cL, cLY, cDi ] = HVM(aspectRatio, taperRatio, quarterChordSweep, -2, wingTipTwist, horseshoeShape,
    nPanels);
    oswaldFactorsColumn = [ oswaldFactorsColumn; cL^2/cDi/pi/aspectRatio ];
  end
  oswaldFactors = [ oswaldFactors oswaldFactorsColumn ];
end

csvwrite('data/hw2_4_aspect.csv', [ [ -1 aspectRatios ]; taperRatios' oswaldFactors ]);
```

hw2_4_aspect.m

```
aspectRatio = 8;
taperRatios = 0.1:0.1:1;
quarterChordSweeps = [ 0 30 60 ];
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
% Initialize output vector
oswaldFactors = [];
for quarterChordSweep = quarterChordSweeps
  oswaldFactorsColumn = [];
  for taperRatio = taperRatios
    [ cL, cLY, cDi ] = HVM(aspectRatio, taperRatio, quarterChordSweep, -2, wingTipTwist, horseshoeShape,
    nPanels);
    oswaldFactorsColumn = [ oswaldFactorsColumn; cL^2/cDi/pi/aspectRatio ];
  end
  oswaldFactors = [ oswaldFactors oswaldFactorsColumn ];
end

csvwrite('data/hw2_4_sweep.csv', [ [ -1 quarterChordSweeps ]; taperRatios' oswaldFactors ]);
```

hw2_4_sweep.m

```
aspectRatio = 8;
taperRatio = 0.5;
%0 degree sweep at c/2 equals 10.61 degrees at the leading edge
quarterChordSweep = quarter_chord_sweep(10.61, aspectRatio, taperRatio);
angleOfAttack = 0;
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
%cLAlpha, clMax and alphaL0 taken from airfoil_data.pdf for NACA 2412
cLAlpha = 0.105;
clMax = 1.68;
alphaL0 = -2;
%We transform panels to wing span from -0.5 to 0.5
wingSpan = linspace(-0.5, 0.5, nPanels)';
%Find additional and basic lift distributions
[ cL1, cLY1, cDi, alpha_i ] = HVM(aspectRatio, taperRatio, quarterChordSweep, -2, wingTipTwist,
    horseshoeShape, nPanels);
[ cL2, cLY2 ] = HVM(aspectRatio, taperRatio, quarterChordSweep, 2, wingTipTwist, horseshoeShape, nPanels);
additionalLift = (cLY1-cLY2)/(cL1-cL2);
basicLift = cLY1 - cL1*additionalLift;
totalLift = basicLift + additionalLift*cL1;
csvwrite('data/hw2_5_lift.csv', [ wingSpan basicLift additionalLift totalLift ]);
%Find cLMax
cLYs = (clMax-basicLift)./additionalLift;
[ cLMax, panel ] = min(cLYs);
alphaMax = cLMax/cLAlpha + alpha_i(panel) + alphaL0;
csvwrite('data/hw2_5_max.csv', [ cLMax alphaMax ]);
%Find cLY variation for 10 degrees of flap
flapAngle = 10;
flapPosition = 0.8;
theta = acos(1-flapPosition*2);
deltaAlphaL0 = (1-theta/pi+sin(theta)/pi)*flapAngle;
deltaCl = deltaAlphaL0*cLAlpha;
%Find additional and basic lift distributions
cLY1 = cLY1 + deltaCl;
cL1 = sum(cLY1)/nPanels;
basicLift = cLY1 - cL1*additionalLift;
totalLift = basicLift + additionalLift*cL1;
csvwrite('data/hw2_5_lift_flap.csv', [ wingSpan basicLift additionalLift totalLift ]);
%Find cLMax again, for flap configuration
cLYs = (clMax-basicLift)./additionalLift;
[ cLMax, panel ] = min(cLYs);
alphaMax = cLMax/cLAlpha + alpha_i(panel) + alphaL0 + deltaAlphaL0;
csvwrite('data/hw2_5_max_flap.csv', [ cLMax alphaMax ]);
```

hw2_5_lift.m

```
aspectRatio = 8;
taperRatio = 0.5;
%0 degree sweep at c/2 equals 10.61 degrees at the leading edge
quarterChordSweep = quarter_chord_sweep(10.61, aspectRatio, taperRatio);
angleOfAttack = 2;
wingTipTwist = 0;
horseshoeShape = 'rectangular';
nPanels = 100;
%cm0 taken from airfoil_data.pdf
cm0 = -0.047;
%We transform panels to wing span from -0.5 to 0.5
wingSpan = linspace(-0.5, 0.5, nPanels)';
[ midPoints, controlPoints, bounded_nodes, trailing_nodes, panelAngles, panelAreas, chords ] =
    wing_discretization(aspectRatio, taperRatio, quarterChordSweep, angleOfAttack, wingTipTwist, nPanels);
integral = 0;
for i=1:nPanels
  integral = integral+chords(i)^2;
end
cM0 = cm0*integral
aerodynamicCenter = 0.25 + tand(quarterChordSweep)/6*(1+2*taperRatio)/(1+taperRatio)
```

hw2_5_moment.m

```
hw2_1;
hw2_2;
hw2_3;
hw2_4_aspect;
hw2_4_sweep;
```

shia_lebouf.m