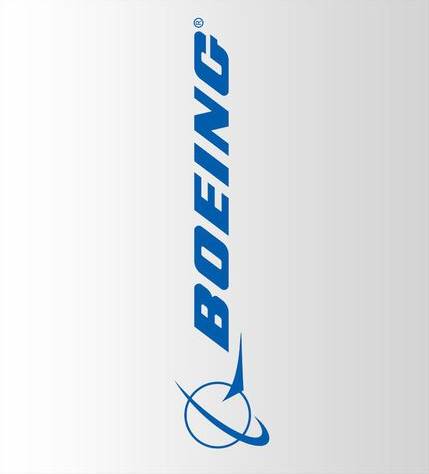
****

**Cairo University**

**Faculty of Engineering**

 **4Th Year Aerospace Department  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**COURSE: AER 402**

**ASSIGNMENT:**

NACA 0012 airfoil **Project**

**Submitted TO:**

**Prof. Mohamed Madbouli**

**Eng: karim mohamed**

**Emg: Hussien Hafez**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **:** Mohamed Hussein Nofal | **Sec:** 1B | **BN:** 43 |

**Date: 10 – 6 - 2021**

Contents

[Problem Definition 3](#_Toc74240243)

[1- Construct a suitable grid over the airfoil (H-grid) 4](#_Toc74240244)

[2- The surface distribution of the metrics of transformation 5](#_Toc74240245)

[3- Boundary and Initial conditions 6](#_Toc74240246)

[Outer boundary 6](#_Toc74240247)

[Airfoil boundary 6](#_Toc74240248)

[Initial condition 6](#_Toc74240249)

[4- Time step for the simulation 7](#_Toc74240250)

[5- Solve the Laplace equation 8](#_Toc74240251)

[6- Velocity and Pressure distributions over the airfoil 9](#_Toc74240252)

[Angle of attack = -5 9](#_Toc74240253)

[Angle of attack = -2.5 10](#_Toc74240254)

[Angle of attack = 0 11](#_Toc74240255)

[Angle of attack = 2.5 12](#_Toc74240256)

[Angle of attack = 5 13](#_Toc74240257)

[7- Lift coefficient and pitching moment coefficient from Code 14](#_Toc74240258)

[8- Code VS experimental data of the airfoil 14](#_Toc74240259)

[Cl 14](#_Toc74240260)

[Cd 14](#_Toc74240261)

[Cm 14](#_Toc74240262)

[9- Code VS ANSYS 15](#_Toc74240263)

[velocities 15](#_Toc74240264)

[pressure 16](#_Toc74240265)

[Cl 18](#_Toc74240266)

[Cd 18](#_Toc74240267)

[Cm 18](#_Toc74240268)

[10- Experimental data VS ANSYS 19](#_Toc74240269)

[Cl 19](#_Toc74240270)

[Cd 19](#_Toc74240271)

[Cm 19](#_Toc74240272)

[Appendix 20](#_Toc74240273)

[ANSYS 20](#_Toc74240274)

[Mesh 20](#_Toc74240275)

[Controls 21](#_Toc74240276)

[Velocity and Pressure coefficient Contours 22](#_Toc74240277)

[Angle of attack = -5 22](#_Toc74240278)

[Angle of attack = -2.5 22](#_Toc74240279)

[Angle of attack = 0 22](#_Toc74240280)

[Angle of attack = 2.5 23](#_Toc74240281)

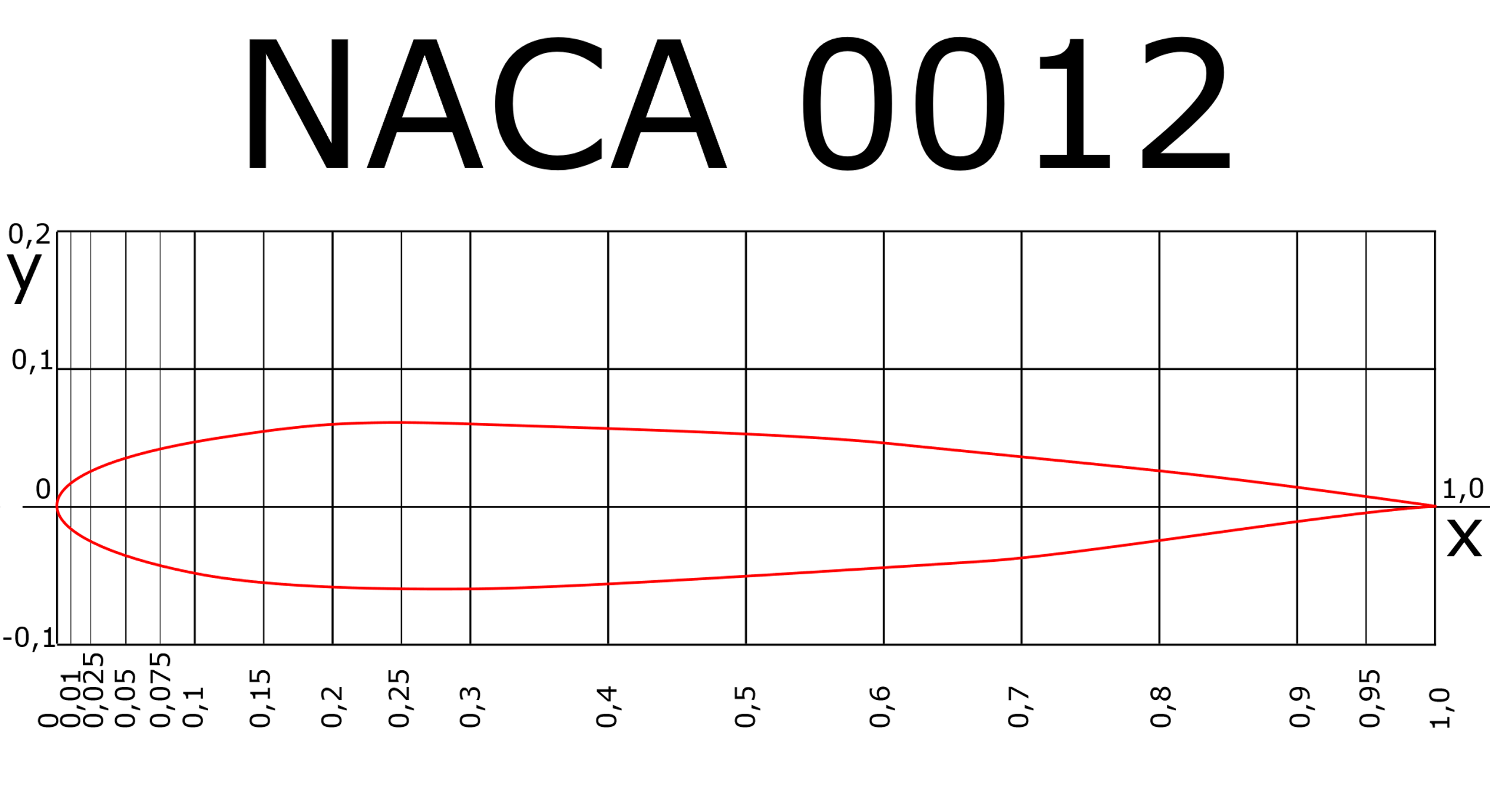
[Angle of attack =5 23](#_Toc74240282)

[Experimental Data 24](#_Toc74240283)

[MATLAB Code 25](#_Toc74240284)

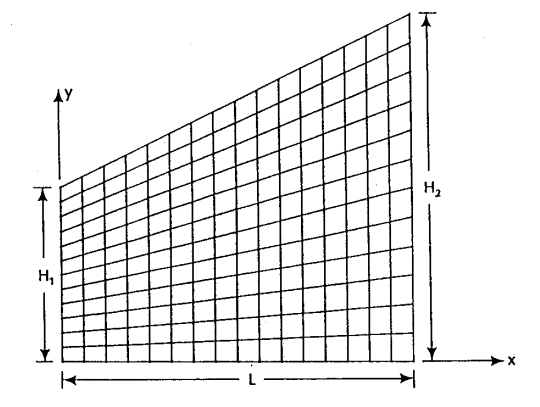
# Problem Definition

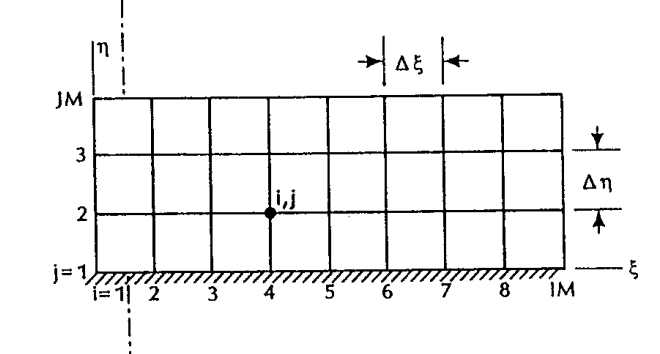
* It is required to use finite difference techniques to solve the Laplace equation over **NACA 0012** airfoil using **MATLAB** and compare the result to the experiment data.
* Solve the same problem with **ANSYS** (inviscid and viscous) and also compare the result to the experiment data and to the difference technique.
* The Range of angles of attack is -5:5 with a step of 2.5 degree.



* NACA 0012 airfoil point (Click [here](https://m-selig.ae.illinois.edu/ads/coord/n0012.dat)).

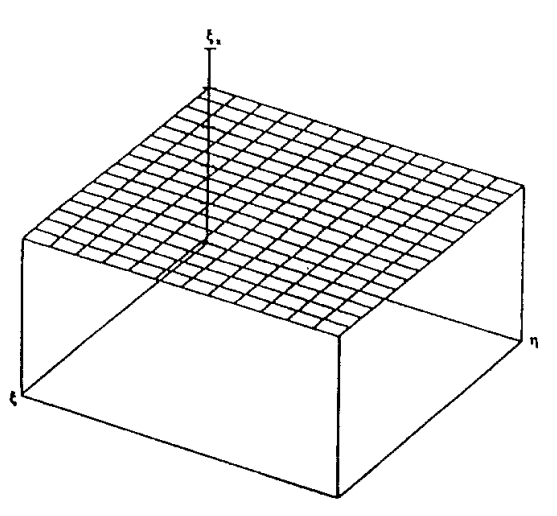
# Construct a suitable grid over the airfoil (H-grid)

* The mesh is chosen to be H-grid for simplification.
* The code divides the domain into four regions {before TE, above airfoil, under airfoil and after LE}.
* Create Mesh





# The surface distribution of the metrics of transformation



* but there is no enough time to plot it.

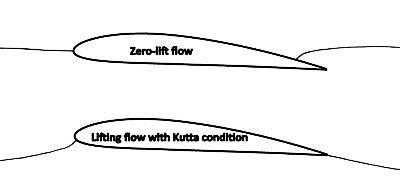
# Boundary and Initial conditions

## Outer boundary

The right points are assumed to have a stream function of zero

The rest of the outer boundary points are defined according to the following equation

## Airfoil boundary



The value of the stream function at the airfoil is not known, so **Kutta condition** is imposed

As the airfoil is considered as a stream line

## Initial condition

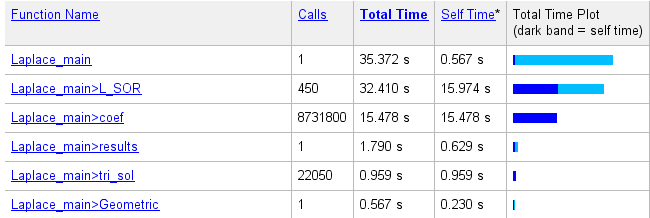
The initial condition of the stream function is calculated by **linear** **interpolation** between inner and outer boundaries.

# Time step for the simulation

As we can see in the figure, the error reduction becomes slower after 450 iterations and the change in the solution becomes very small which is in order of , so we can use only **450** as an optimum value and we will still have pretty much good results.



* The computational time is 35.372 sec .



# Solve the Laplace equation

The Laplace equation is one of the examples of the elliptic equations. The code numerically integrates the Laplace equation at each point of the elliptic grid using both PSOR and LSOR scheme as following

* Laplace Equation
* The PSOR scheme
* The LSOR scheme
* Calculate Pressure coefficient

# Velocity and Pressure distributions over the airfoil

## Angle of attack = -5

|  |  |
| --- | --- |
|  |  |
|  |  |

## Angle of attack = -2.5

|  |  |
| --- | --- |
|  |  |
|  |  |

## Angle of attack = 0

|  |  |
| --- | --- |
|  |  |
|  |  |

## Angle of attack = 2.5

|  |  |
| --- | --- |
|  |  |
|  |  |

## Angle of attack = 5

|  |  |
| --- | --- |
|  |  |
|  |  |

# Lift coefficient and pitching moment coefficient from Code

|  |  |  |  |
| --- | --- | --- | --- |
| AOA | Cl | Cd | Cm |
| -5 | -0.6674 | 0.0410 | 0.0298 |
| -2.5 | -0.3294 | 0.0641 | 0.0150 |
| 0 | 0.0000 | 0.0000 | 0.0000 |
| 2.5 | 0.3337 | 0.0352 | -0.0150 |
| 5 | 0.6674 | 0.0410 | -0.0298 |

**Note that :** Cd should be equal to zero but due to the numerical approximations and numerical errors it doesn’t.

# Code VS experimental data of the airfoil

## Cl

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| **Experimental** | -0.5 | -0.25 | 0 | 0.25 | 0.5 |
| **Code** | -0.6674 | -0.3294 | 0 | 0.3337 | 0.6674 |

## Cd

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| **Experimental** | 0.007 | 0.0065 | 0.006 | 0.0065 | 0.007 |
| **Code** | 0.00410 | 0.00641 | 0 | 0.00352 | 0.00410 |

## Cm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| **Experimental** | 0.01 | 0.005 | 0 | -0.005 | -0.01 |
| **Code** | 0.0298 | 0.0150 | 0.0000 | -0.0150 | -0.0298 |

# Code VS ANSYS

## velocities

|  |  |  |
| --- | --- | --- |
| **AOA** | **CODE** | **ANSYS** |
| -5 |  |  |
| -2.5 |  |  |
| 0 |  |  |
| 2.5 |  |  |
| 5 |  |  |

## pressure

|  |  |  |
| --- | --- | --- |
| **AOA** | **CODE** | **ANSYS** |
| -5 |  |  |
| -2.5 |  |  |
| 0 |  |  |
| 2.5 |  |  |
| 5 |  |  |

## Cl

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| Code | -0.6674 | -0.3294 | 0 | 0.3337 | 0.6674 |
| ANSYS(inviscid) | -0.56162 | -0.28 | 0.00645 | 0.29336 | 0.5715 |
| ANSYS(viscous) | -0.47794 | -0.24873 | 0.00222 | 0.24873 | 0.4812 |

## Cd

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| Code | 0.00410 | 0.00641 | 0 | 0.00352 | 0.00410 |
| ANSYS(inviscid) | 0.00344 | 0.00158 | 0.000126 | 0.001829 | 0.004037 |
| ANSYS(viscous) | 0.016113 | 0.013517 | 0.010753 | 0.01312 | 0.015999 |

## Cm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| Code | 0.0298 | 0.0150 | 0.0000 | -0.0150 | -0.0298 |
| ANSYS(inviscid) | -0.002233 | -0.000647 | 0.00135 | 0.0032 | 0.003788 |
| ANSYS(viscous) | 0.012645 | 0.0046904 | 0.0004195 | -0.00469 | -0.012539 |

# Experimental data VS ANSYS

## Cl

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| Experiment | -0.5 | -0.25 | 0 | 0.25 | 0.5 |
| ANSYS(viscous) | -0.47794 | -0.24873 | 0.00222 | 0.24873 | 0.4812 |
| ANSYS(inviscid) | -0.56162 | -0.28 | 0.00645 | 0.29336 | 0.5715 |

## Cd

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| Experiment | 0.016 | 0.014 | 0.012 | 0.014 | 0.016 |
| ANSYS(viscous) | 0.016113 | 0.013517 | 0.010753 | 0.01312 | 0.015999 |
| ANSYS(inviscid) | 0.00344 | 0.00158 | 0.000126 | 0.001829 | 0.004037 |

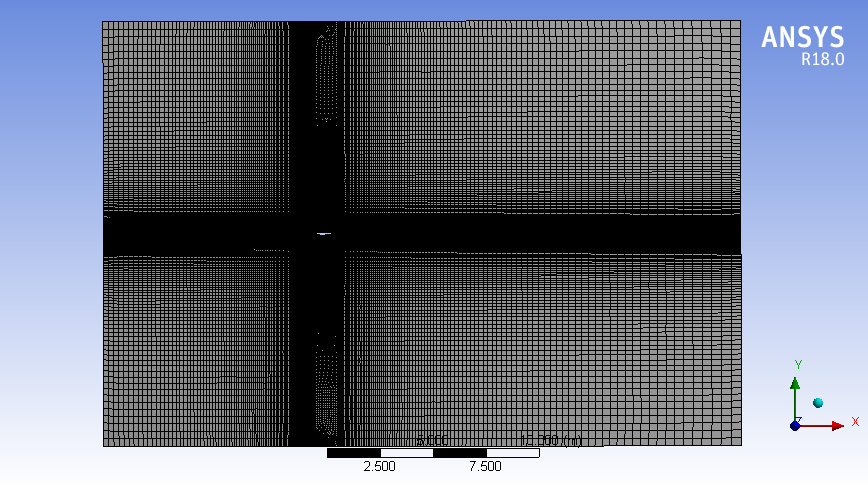
## Cm

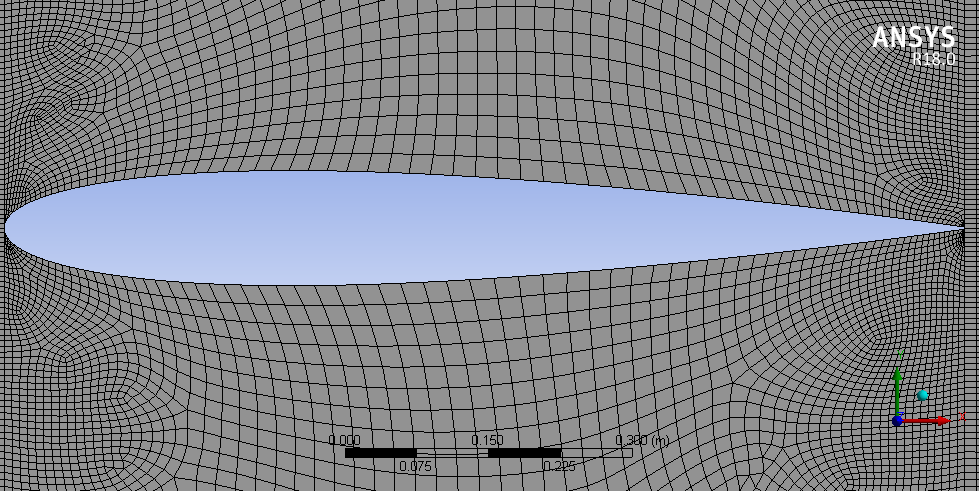
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| AOA | -5 | -2.5 | 0 | 2.5 | 5 |
| Experiment | 0.01 | 0.005 | 0 | -0.005 | -0.01 |
| ANSYS(viscous) | 0.012645 | 0.004690 | 0.00041948 | -0.00469 | -0.012539 |
| ANSYS(inviscid) | -0.002233 | -0.000647 | 0.00135 | 0.0032 | 0.003788 |

# Appendix

## ANSYS

### Mesh





### Controls

|  |  |
| --- | --- |
| **Viscous Flow** | **Inviscid Flow** |
|  |  |

## Velocity and Pressure coefficient Contours

### Angle of attack = -5

|  |  |
| --- | --- |
|  |  |

### Angle of attack = -2.5

|  |  |
| --- | --- |
|  |  |

### Angle of attack = 0

|  |  |
| --- | --- |
|  |  |

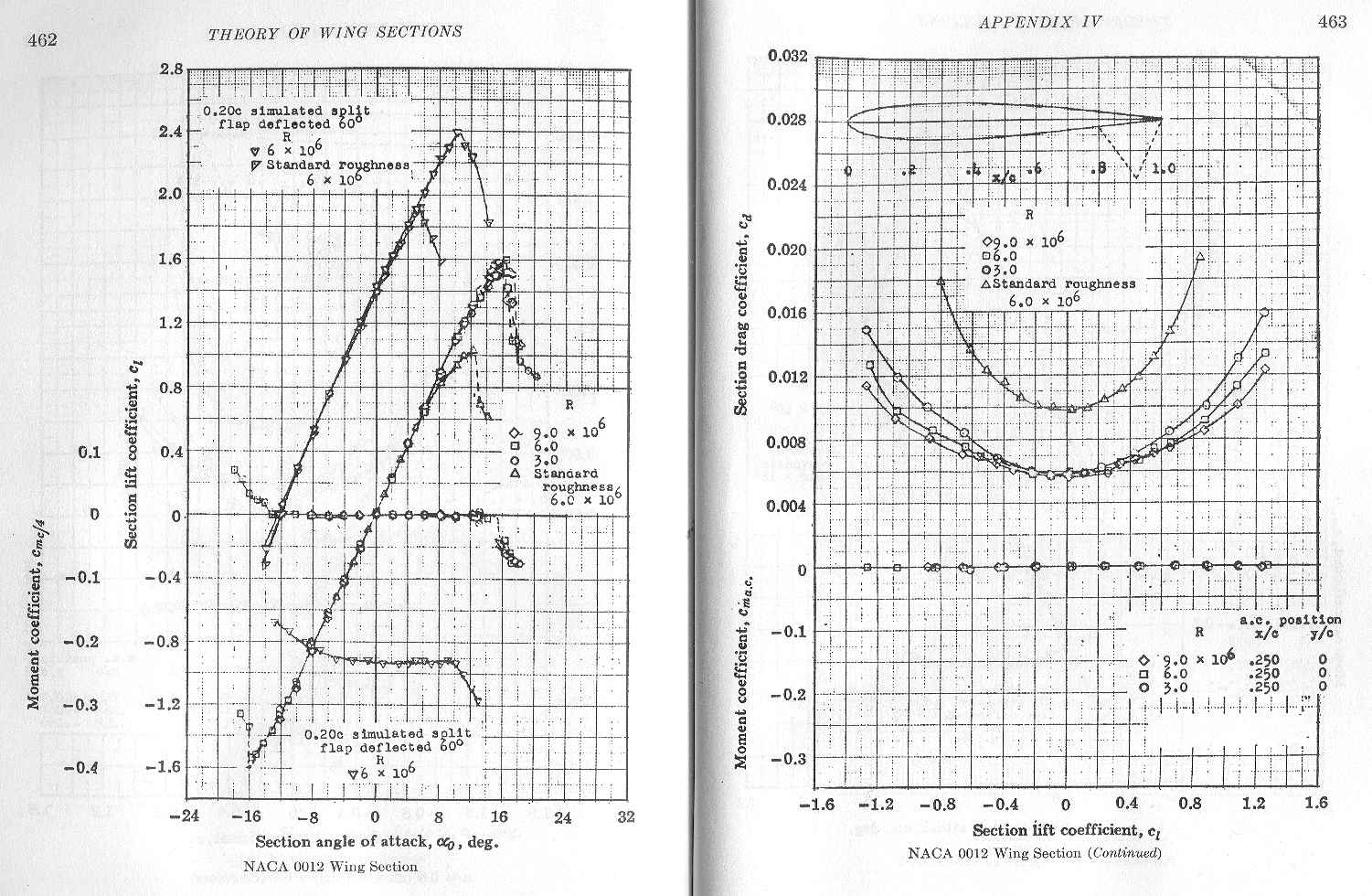
### Angle of attack = 2.5

|  |  |
| --- | --- |
|  |  |

### Angle of attack =5

|  |  |
| --- | --- |
|  |  |

## Experimental Data



## MATLAB Code

|  |
| --- |
| clc;clear all;close all; %profile on  % tic  %% Input Data  Vinf = 10;  alfad = -2.5;  cord = 1;  nitd = 0;il = 31; it = 71; imax = 101; jair = 26; jmax = 51;  omega = 1; per = 1\*10^-6; nmax = 450;  %% Calculated Data  alfa = alfad \* pi / 180;  cosa = cos(alfa); sina = sin(alfa);  uxinf = Vinf \* cosa; uyinf = Vinf \* sina;  iimax = 2 \* imax - 1; jjmax = 2 \* jmax - 1; jjair = 2 \* jair - 1;  lx = 3 \* cord; ly = cord; M = imax;  nx = imax - 1; ny = jmax - 1; d1 = 1 / (it - il); d2 = 1 / ny;  dx = cord / (it - il); dy = ly / ny;  r = dx / dy; t1 = omega / (2 \* (1 + r \* r));  t2 = t1 \* r \* r;  %% Call Geometric Function  [x,y,yal,yau]=Geometric(imax,jmax, jair, il, it, cord);  %% Method of solution PSOR or LSOR  % Method = 0 ; %if you want to solve by PSOR  Method = 1 ; %if you want to solve by LSOR  %% Boundary values & Initialization  ps(1, 1) = 0;  i = 1;  for j = 1 : jmax - 1  ii=2\*i-1; jj=2\*j-1;  ps(i, j + 1) = ps(i, j) + uxinf \* (y(ii, jj + 2) - y(ii, jj));  end  j = 1;  for i = 1 : imax - 1  ii=2\*i-1; jj=2\*j-1;  ps(i + 1, j) = ps(i, j) - uyinf \* (x(ii + 2, jj) - x(ii, jj));  end  i = imax;  for j = 1 : jmax - 1  ii=2\*i-1; jj=2\*j-1;  ps(i, j + 1) = ps(i, j) + uxinf \* (y(ii, jj + 2) - y(ii, jj));  end  j = jmax;  for i = 1 : imax - 1  ii=2\*i-1; jj=2\*j-1;  ps(i + 1, j) = ps(i, j) - uyinf \* (x(ii + 2, jj) - x(ii, jj));  end  for i = 2 : imax - 1  for j = 2 : jmax - 1  ps(i, j) = ps(i, 1) + (ps(i, jmax) - ps(i, 1)) \* (j - 1) / (jmax - 1);  end  end  psp = ps;  for n=1:nmax  %while erps > .001 ; %n = nitd; %n = n + 1;  % Calculation of new values of ps(i,j) = psp(i,j)  if Method == 0 ; psp=P\_SOR(x,y,imax, jmax, jair, il, it, yal, yau, ps, r, omega, d1, d2) ; end  if Method == 1 ; psp=L\_SOR(y ,imax, jmax, jair, il ,it ,yal, yau, r, x, d1, d2,ps,psp) ; end  % Errors calculation  mder = 0;  for i = 2 : nx  for j = 2 : ny  der = abs(psp(i, j) - ps(i, j));  if der > mder ; mder = der; ier = i; jer = j ; end  ps(i, j) = psp(i, j);  end  end  if mder > 0 ; lmder = log10(mder) ; end  % Updating the value of psi on the airfoil  psold = psp(it, jair);  psnew = psp(it + 1, jair);  erps = abs(psnew - psold);  j = jair;  for i = il : it  ps(i, j) = psnew;  psp(i, j) = psnew;  end  a\_n(n)=n ; a\_lmder(n)=lmder;  % Check convergence  %' if ((mder > per) AND (n <= nmax)) THEN GO: iter    end    % % THE KUTTA CONDITION  % % NEED SOME CODE TO RETURN THE INDEX OF THE NEAREST POINT TO CAMBER LINE EXTENSION POINT  % % in dr oroginal code yKutta=jair  % psnew = psp(it + 1, jair);  % erps = abs(psnew - psold);  % j = jair;  % for i = il : it  % ps(i, j) = psnew;  % psp(i, j) = psnew;  % end  % a\_n(n)=n ; a\_lmder(n)=lmder; a\_erps(n)=erps;  % if (rem(n,10)==0&&n~=10)  % line([a\_n(n-10),n],[a\_lmder(n-10),lmder])  % pause(.0001);  % end    figure  plot(a\_n,a\_lmder,'linewidth',2)  grid on;axis tight  xlabel('Iteration number', 'fontsize',14)  ylabel('Log\_1\_0 (Error)', 'fontsize',14)  title('Convergence history using LSOR for the flow past NACA-0012 airfoil ','fontsize',14)  %% Call Result Function  results(x, y, imax, jmax, jair, il, it, cord, yal, yau, ps, d1, d2, Vinf, cosa, sina ,psp)  % toc  % profile off  % profile viewer  %% Helping Functions  function [c11,c12,c22]=coef(ip,jp,x,y, d1, d2)  % global x y d1 d2  % calculate the metric terms and the jacobian.  d1x = (x(ip + 1, jp) - x(ip - 1, jp)) / d1;  d1y = (y(ip + 1, jp) - y(ip - 1, jp)) / d1;  d2x = (x(ip, jp + 1) - x(ip, jp - 1)) / d2;  d2y = (y(ip, jp + 1) - y(ip, jp - 1)) / d2;  jaco = d1x \* d2y - d1y \* d2x;  c11 = (d2x \* d2x + d2y \* d2y) / jaco;  c12 = -(d1x \* d2x + d1y \* d2y) / jaco;  c22 = (d1x \* d1x + d1y \* d1y) / jaco;  end  function psp=P\_SOR(x,y,imax, jmax, jair, il, it, yal, yau, ps, r, omega, d1, d2)    iimax = 2\*imax-1 ; jjmax = 2\*jmax-1;jjair = 2\*jair-1;  for j = 2 : jmax - 1  if j == jair - 1 ; for ii = 1 : iimax; y(ii, jjair) = yal(ii); end; end  if j == jair + 1 ; for ii = 1 : iimax; y(ii, jjair) = yau(ii); end; end  for i = 2 : imax - 1  ii = 2 \* i - 1;  jj = 2 \* j - 1;  ip = ii + 1; jp = jj; [c11ip c12ip c22ip]=coef(ip,jp,x ,y, d1, d2);  ip = ii - 1; jp = jj; [c11im c12im c22im]=coef(ip,jp,x ,y, d1, d2);  ip = ii; jp = jj + 1; [c11jp c12jp c22jp]=coef(ip,jp,x ,y, d1, d2);  ip = ii; jp = jj - 1; [c11jm c12jm c22jm]=coef(ip,jp,x ,y, d1, d2);  sij = c11ip + c11im + r \* r \* (c22jp + c22jm);  sim = c11im - r \* r \* (c12jp - c12jm) / 4;  sip = c11ip + r \* r \* (c12jp - c12jm) / 4;  sjm = r \* r \* c22jm - r \* (c12ip - c12im) / 4;  sjp = r \* r \* c22jp + r \* (c12ip - c12im) / 4;  smm = r \* (c12im + c12jm) / 4;  smp = -r \* (c12im + c12jp) / 4;  spm = -r \* (c12ip + c12jm) / 4;  spp = r \* (c12ip + c12jp) / 4;  psp(i, j) = sim \* ps(i - 1, j) + sip \* ps(i + 1, j) + sjm \* ps(i, j - 1) + sjp \* ps(i, j + 1);  psp(i, j) = psp(i, j) + smm \* ps(i - 1, j - 1) + smp \* ps(i - 1, j + 1);  psp(i, j) = psp(i, j) + spm \* ps(i + 1, j - 1) + spp \* ps(i + 1, j + 1);  psp(i, j) = psp(i, j) / sij;  psp(i, j) = ps(i, j) + omega \* (psp(i, j) - ps(i, j));  end  if j == jair; for i=il:it ;psp(i,j)=ps(i,j);end; end  end  end  function psp=L\_SOR(y ,imax, jmax, jair, il ,it ,yal, yau, r, x, d1, d2 ,ps,psp)    iimax = 2\*imax-1 ; jjmax = 2\*jmax-1;jjair = 2\*jair-1;  for j = 2 : jmax-1  if (j == jair - 1) ; for ii = 1 : iimax; y(ii, jjair) = yal(ii); end ; end  if (j == jair + 1) ; for ii = 1 : iimax; y(ii, jjair) = yau(ii); end ; end  b(1) = 0; d(1) = 1; a(1) = 0; c(1) = ps(1, j);  b(imax) = 0; d(imax) = 1; a(imax) = 0; c(imax) = ps(imax, j);  for i = 2 : imax-1  ii = 2 \* i - 1;  jj = 2 \* j - 1;  ip = ii + 1; jp = jj; [c11ip c12ip c22ip]=coef(ip,jp,x,y,d1,d2);  ip = ii - 1; jp = jj; [c11im c12im c22im]=coef(ip,jp,x,y,d1,d2);  ip = ii; jp = jj + 1; [c11jp c12jp c22jp]=coef(ip,jp,x,y,d1,d2);  ip = ii; jp = jj - 1; [c11jm c12jm c22jm]=coef(ip,jp,x,y,d1,d2);  sij = c11ip + c11im + r \* r \* (c22jp + c22jm);  sim = c11im - r \* r \* (c12jp - c12jm) / 4;  sip = c11ip + r \* r \* (c12jp - c12jm) / 4;  sjm = r \* r \* c22jm - r \* (c12ip - c12im) / 4;  sjp = r \* r \* c22jp + r \* (c12ip - c12im) / 4;  smm = r \* (c12im + c12jm) / 4;  smp = -r \* (c12im + c12jp) / 4;  spm = -r \* (c12ip + c12jm) / 4;  spp = r \* (c12ip + c12jp) / 4;  b(i) = -sim; d(i) = sij; a(i) = -sip;  c(i) = sjm \* psp(i, j - 1) + sjp \* ps(i, j + 1) + smm \* psp(i - 1, j - 1);  c(i) = c(i) + smp \* ps(i - 1, j + 1) + spm \* psp(i + 1, j - 1) + spp \* ps(i + 1, j + 1);  if ((j == jair) && (i >= il) && (i <= it)) ;  b(i) = 0; d(i) = 1; a(i) = 0; c(i) = psp(i, j);  end  end  ps\_p=tri\_sol(a,b,c,d,imax);  for i=1:imax ; psp(i,j)=ps\_p(i); end  end  end  function [x,y,yal,yau]=Geometric(imax,jmax, jair, il, it, cord)  % of H-grid for NACA-0012  % il = i of the leading edge  % it = i of the trailing edge  % cord = chord length  figure  axis equal;axis tight ;  iil = 2 \* il - 1;  iit = 2 \* it - 1;  iimax = 2 \* imax - 1;  jjmax = 2 \* jmax - 1;  jjair = 2 \* jair - 1;    for jj = 1 : jjmax  for ii = 1 : iimax  x(ii, jj) = (cord / (iit - iil)) \* (ii - iil);  end  end    %toc is the thickness to chord ratio for NACA 0012 toc=.12  toc = .12;  for ii = 1 : iil; y(ii, jjair) = 0; end  for ii = iil : iit; xp = x(ii, jjair);  y(ii, jjair) = -5 \* toc \* (.2969 \* sqrt(xp) - .126 \* xp - .3537 \* xp ^ 2 + .2843 \* xp ^ 3 - .1015 \* xp ^ 4);  end  for ii = iit : iimax; y(ii, jjair) = 0; end  for ii = 1 : iimax; yal(ii) = y(ii, jjair); end  for ii = 1 : iimax; y(ii, 1) = -cord; end  for ii = 1 : iimax  for jj = 2 : jjair - 1  y(ii, jj) = y(ii, 1) + (jj - 1) \* (y(ii, jjair) - y(ii, 1)) / (jjair - 1);  end  end  %'toc is the thickness to chord ratio for NACA 0012 toc=.12  toc = .12;  for ii = 1 : iil; y(ii, jjair) = 0; end  for ii = iil : iit; xp = x(ii, jjair);  y(ii, jjair) = 5 \* toc \* (.2969 \* sqrt(xp) - .126 \* xp - .3537 \* xp ^ 2 + .2843 \* xp ^ 3 - .1015 \* xp ^ 4);  end  for ii = iit : iimax; y(ii, jjair) = 0; end  for ii = 1 : iimax; yau(ii) = y(ii, jjair); end  for ii = 1 : iimax; y(ii, jjmax) = cord; end  for ii = 1 : iimax  for jj = jjair + 1 : jjmax - 1  y(ii, jj) = y(ii, jjair) + (jj - jjair) \* (y(ii, jjmax) - y(ii, jjair)) / (jjmax - jjair);  end  end  % Plot the H-Grid  for j = 1 : jair - 1; jj = 2 \* j - 1;  x1 = x(:,jj); y1=y(:,jj);plot (x1,y1,'k'); hold on;  end  jj = jjair ;x1 = x(:,jj); y1=yal(:);plot (x1,y1,'k'); hold on  x1 = x(:,jj); y1=yau(:);plot (x1,y1,'k'); hold on  for j = jair + 1 : jmax; jj = 2 \* j - 1;  x1 = x(:, jj); y1 = y(:, jj);plot (x1,y1,'k'); hold on;  end  y(:, jjair) = yal(:);  for i=1:il; ii=2\*i-1; x1 = x(ii,:);  y1 = y(ii,:);plot (x1,y1,'k'); hold on; end  y(:, jjair) = yal(:);  for i=il+1:it-1; ii=2\*i-1;  for j=1:jair; jj=2\*j-1;  x2(j)= x(ii,jj);y2(j)= y(ii,jj);  end  plot (x2,y2,'k'); hold on;  end  y(:, jjair) = yau(:);  for i=il+1:it-1;ii=2\*i-1;  for j=jair:jmax; jj=2\*j-1;k=j-jair+1;  x2(k)= x(ii,jj);y2(k)= y(ii,jj);  end  plot (x2,y2,'k'); hold on;  end  for i=it:imax;ii=2\*i-1;x1 = x(ii,:);  y1 = y(ii,:);plot (x1,y1,'k'); hold on;  end  xlabel('X-axis', 'fontsize',14)  ylabel('Y-axis', 'fontsize',14)  title('H-Grid for NACA-0012 airfoil ','fontsize',14)  end  function e=tri\_sol(a,b,c,d,M)  for i=2:M  t = b(i) / d(i - 1);  d(i) = d(i) - t \* a(i - 1);  c(i) = c(i) - t \* c(i - 1);  end  e(M) = c(M) / d(M);  for k=2:M  i = M - k + 1;  e(i) = (c(i) - a(i) \* e(i + 1)) / d(i);  end  end  function results(x, y, imax, jmax, jair, il, it, cord, yal, yau, ps, d1, d2, Vinf, cosa, sina ,psp)  iimax = 2\*imax-1 ; jjmax = 2\*jmax-1;jjair = 2\*jair-1;  % calculation of the velocity and the pressure coefficients  uxinf = Vinf \* cosa; uyinf = Vinf \* sina;  i=1 ; for j=1:jmax ; a\_vx(i,j)=uxinf; a\_vy(i,j)=uyinf; end  i=imax ; for j=1:jmax ; a\_vx(i,j)=uxinf; a\_vy(i,j)=uyinf; end  j=1 ; for i=1:imax ; a\_vx(i,j)=uxinf; a\_vy(i,j)=uyinf; end  j=jmax ; for i=1:imax ; a\_vx(i,j)=uxinf; a\_vy(i,j)=uyinf; end    for i=2:imax-1  for j=2:jmax-1  ii=2\*i-1;jj=2\*j-1;  d1x = (x(ii + 1, jj) - x(ii - 1, jj)) / d1;  d1y = (y(ii + 1, jj) - y(ii - 1, jj)) / d1;  d2x = (x(ii, jj + 2) - x(ii, jj)) / d2;  d2y = (y(ii, jj + 2) - y(ii, jj)) / d2;  jaco = d1x \* d2y - d1y \* d2x;  et1x = d2y / jaco; et1y = -d2x / jaco;  et2x = -d1y / jaco; et2y = d1x / jaco;  d1u = (ps(i + 1, j) - ps(i - 1, j)) / 2 / d1;  d2u = (ps(i, j + 1) - ps(i, j - 1)) / 2 / d2;  a\_vx(i,j) = d1u \* et1y + d2u \* et2y;  a\_vy(i,j) = -(d1u \* et1x + d2u \* et2x);  end  end    j = jair;  % for upper surface  for ii = 1 : iimax; y(ii, jjair) = yau(ii); end  for i = il : it  ii = 2 \* i - 1; jj = 2 \* j - 1;  d1x = (x(ii + 1, jj) - x(ii - 1, jj)) / d1;  d1y = (y(ii + 1, jj) - y(ii - 1, jj)) / d1;  d2x = (x(ii, jj + 2) - x(ii, jj)) / d2;  d2y = (y(ii, jj + 2) - y(ii, jj)) / d2;  jaco = d1x \* d2y - d1y \* d2x;  et1x = d2y / jaco; et1y = -d2x / jaco;  et2x = -d1y / jaco; et2y = d1x / jaco;  d1u = (ps(i + 1, j) - ps(i - 1, j)) / 2 / d1;  d2u = (4 \* ps(i, j + 1) - 3 \* ps(i, j) - ps(i, j + 2)) / 2 / d2;  vx = d1u \* et1y + d2u \* et2y;  vy = -(d1u \* et1x + d2u \* et2x);  a\_vx\_up(i)=vx ; a\_vy\_up(i)=vy;  % a\_vx(i,j)=vx; a\_vy(i,j)=vy;  Vel = sqrt(vx \* vx + vy \* vy);  Vru(i) = Vel / Vinf;  Cpu(i) = 1 - Vru(i) \* Vru(i);  xup(i) = x(ii, jj);  yup(i) = y(ii, jj);  end  % for lower surface  j = jair;  for ii = 1 : iimax; y(ii, jjair) = yal(ii); end  for i = il : it  ii = 2 \* i - 1; jj = 2 \* j - 1;  d1x = (x(ii + 1, jj) - x(ii - 1, jj)) / d1;  d1y = (y(ii + 1, jj) - (y(ii - 1, jj))) / d1;  d2x = (x(ii, jj) - x(ii, jj - 2)) / d2;  d2y = (y(ii, jj) - y(ii, jj - 2)) / d2;  jaco = d1x \* d2y - d1y \* d2x;  et1x = d2y / jaco; et1y = -d2x / jaco;  et2x = -d1y / jaco; et2y = d1x / jaco;  d1u = (ps(i + 1, j) - ps(i - 1, j)) / 2 / d1;  d2u = (-4 \* ps(i, j - 1) + 3 \* ps(i, j) + ps(i, j - 2)) / 2 / d2;  vx = d1u \* et1y + d2u \* et2y;  vy = -(d1u \* et1x + d2u \* et2x);  a\_vx\_lo(i)=vx ; a\_vy\_lo(i)=vy;  %a\_vx(i,j)=vx; a\_vy(i,j)=vy;  Vel = sqrt(vx \* vx + vy \* vy);  Vrl(i) = Vel / Vinf;  Cpl(i) = 1 - Vrl(i) \* Vrl(i);  xlo(i) = x(ii, jj);  ylo(i) = y(ii, jj);  end    figure  hold on;grid on  plot(xup,Vru,'b',xlo,Vrl,'r','linewidth',2)  plot(xup,yup,'k',xlo,ylo,'k','linewidth',2)    xlabel('Chord line', 'fontsize',14)  ylabel('Non-dimensional velocity', 'fontsize',14)  title('Non-dimensional velocity over NACA-0012 airfoil surface','fontsize',14)  legend('upper surface','lower surface','Location','best');grid on;      figure(1)  plot(xup,yup,'c',xlo,ylo,'c','linewidth',1)  area(xup,yup,'FaceColor','c')  area(xlo,ylo,'FaceColor','c')  xlim([-0.75 1.75])    figure  hold on;grid on;axis tight  plot(xup,Cpu,'b',xlo,Cpl,'r','linewidth',2)  plot(xup,yup,'k',xlo,ylo,'k','linewidth',2)  xlabel('Chord line', 'fontsize',14)  ylabel('Pressure coefficient', 'fontsize',14)  title('Pressure coefficient over NACA-0012 airfoil surface','fontsize',14)  legend('upper surface','lower surface','Location','best')    figure  i=1; ii=2\*i-1; for j=1:jmax ; jj=2\*j-1; x1(j)=x(ii,jj); y1(j)=y(ii,jj);end  i=imax; ii=2\*i-1; for j=1:jmax ; jj=2\*j-1; x2(j)=x(ii,jj); y2(j)=y(ii,jj);end  j=1; jj=2\*j-1; for i=1:imax ; ii=2\*i-1; x3(i)=x(ii,jj); y3(i)=y(ii,jj);end  j=jmax; jj=2\*j-1; for i=1:imax ; ii=2\*i-1; x4(i)=x(ii,jj); y4(i)=y(ii,jj);end    plot (x1,y1,x2,y2,x3,y3,x4,y4)  hold on;axis equal    j=jair; jj=2\*j-1;  for i=il:it; ii=2\*i-1;k=i-il+1;x5(k)= x(ii,jj);y5(k)=yal(ii); end  plot (x5,y5,'k','linewidth',2); hold on;  for i=il:it; ii=2\*i-1;k=i-il+1;x6(k)= x(ii,jj);y6(k)=yau(ii); end  plot (x6,y6,'k','linewidth',2); hold on;  area(x5,y5,'FaceColor','k')  area(x6,y6,'FaceColor','k')  for ii = 1 : iimax; y(ii, jjair) = yau(ii); end  for i=1:imax  ii=2\*i-1;  for j=jair:jmax  jj=2\*j-1;  k=j-jair+1; x7(i,k)=x(ii,jj); y7(i,k)=y(ii,jj);p7(i,k)=ps(i,j);  end  end    contour(x7,y7,p7,30,'b')  hold on;    for ii = 1 : iimax; y(ii, jjair) = yal(ii); end  for i=1:imax  ii=2\*i-1;  for j=1:jair  jj=2\*j-1;  x8(i,j)=x(ii,jj); y8(i,j)=y(ii,jj);p8(i,j)=ps(i,j);  end  end  % figure  contour(x8,y8,p8,30,'b')  xlabel('X-axis', 'fontsize',14)  ylabel('Y-axis', 'fontsize',14)  title('Stream lines for the flow past NACA-0012 airfoil ','fontsize',14)      figure  i=1; ii=2\*i-1; for j=1:jmax ; jj=2\*j-1; x1(j)=x(ii,jj); y1(j)=y(ii,jj);end  i=imax; ii=2\*i-1; for j=1:jmax ; jj=2\*j-1; x2(j)=x(ii,jj); y2(j)=y(ii,jj);end  j=1; jj=2\*j-1; for i=1:imax ; ii=2\*i-1; x3(i)=x(ii,jj); y3(i)=y(ii,jj);end  j=jmax; jj=2\*j-1; for i=1:imax ; ii=2\*i-1; x4(i)=x(ii,jj); y4(i)=y(ii,jj);end    plot (x1,y1,x2,y2,x3,y3,x4,y4)  hold on  xlabel('X-axis', 'fontsize',14)  ylabel('Y-axis', 'fontsize',14)  title('Velocity vector for the flow past NACA-0012 airfoil ','fontsize',14)    j=jair; jj=2\*j-1;  for i=il:it; ii=2\*i-1;k=i-il+1;x5(k)= x(ii,jj);y5(k)=yal(ii); end  plot (x5,y5); hold on;  for i=il:it; ii=2\*i-1;k=i-il+1;x6(k)= x(ii,jj);y6(k)=yau(ii); end  plot (x6,y6); hold on;    for ii = 1 : iimax; y(ii, jjair) = yau(ii);end  for i=il:it ; a\_vx(i,jair)=a\_vx\_up(i);a\_vy(i,jair)=a\_vy\_up(i); end  for i=1:imax  ii=2\*i-1;  for j=jair:jmax  jj=2\*j-1;  k=j-jair+1; x7(i,k)=x(ii,jj); y7(i,k)=y(ii,jj);a\_vx\_7(i,k)=a\_vx(i,j);a\_vy\_7(i,k)=a\_vy(i,j);  end  end  quiver(x7,y7,a\_vx\_7,a\_vy\_7)  hold on;    for ii = 1 : iimax; y(ii, jjair) = yal(ii); end  for i=il:it ;a\_vx(i,jair)=a\_vx\_lo(i);a\_vy(i,jair)=a\_vy\_lo(i); end  for i=1:imax  ii=2\*i-1;  for j=1:jair  jj=2\*j-1;  x8(i,j)=x(ii,jj); y8(i,j)=y(ii,jj);a\_vx\_8(i,j)=a\_vx(i,j);a\_vy\_8(i,j)=a\_vy(i,j);  end  end  quiver(x8,y8,a\_vx\_8,a\_vy\_8)  hold on;  axis tight    % figure  % hold on  % a\_vu = sqrt(a\_vx\_7.^2+a\_vy\_7.^2);  % contourf(x7,y7,a\_vu,'LineColor','none')  % a\_vl = sqrt(a\_vx\_8.^2+a\_vy\_8.^2);  % contourf(x8,y8,a\_vl,'LineColor','none')  % plot(xup,yup,'k','LineWidth',1.2)  % plot(xlo,ylo,'k','LineWidth',1.2)  % colormap('jet');  % colorbar  % title('Velocity Contor for the flow past NACA-0012 airfoil ','fontsize',14)  % Calculation of the lift and drag coefficients  cx = 0; cy = 0;  j = jair; jj = jjair;  for i = il : it - 1  ii = 2 \* i - 1;  x1 = x(ii, jj); y1 = yau(ii); x2 = x(ii + 2, jj); y2 = yau(ii + 2);  cx = cx + .5 \* (Cpu(i) + Cpu(i + 1)) \* (y2 - y1) / cord;  cy = cy - .5 \* (Cpu(i) + Cpu(i + 1)) \* (x2 - x1) / cord;  % cm\_u\_=cy\*(x1-0.25);  x1 = x(ii, jj); y1 = yal(ii); x2 = x(ii + 2, jj); y2 = yal(ii + 2);  cx = cx + .5 \* (Cpl(i) + Cpl(i + 1)) \* (y2 - y1) / cord;  cy = cy + .5 \* (Cpl(i) + Cpl(i + 1)) \* (x2 - x1) / cord;    cm\_l\_=cy\*(x1-0.25);  M\_Cx(i)=cy;  M\_X(i)=x1    end  cl = cy \* cosa - cx \* sina  cd = cy \* sina + cx \* cosa    Cm = trapz(M\_Cx,M\_X)  end |