UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

Titulació:

Grau en Enginyeria de Tecnologies Aeroespacials

Alumne:

Alexandre Cortiella Segarra

Títol TFG:

Study of numerical techniques for structural optimization in aeronautics

Director TFG:

Juan Carlos Cante Teran

Codirector TFG:

Àlex Ferrer Ferré

Joaquín Hernández Ortega

Tutor TFG:

Daniel Garcia Almiñana

Convocatòria de lliurament del TFG:

juny 2014

Contingut d'aquest volum: **ANNEXOS**

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

Study of numerical techniques for structural optimization in aeronautics

Escola Tècnica Superior d'Enginyeria Industrial i Aeronàutica de Terrassa

(ETSEIAT)

Grau en Enginyeria de Tecnologies Aeroespacials

Director TFG:

Juan Carlos Cante Teran

Codirector TFG:

Àlex Ferrer Ferré

Joaquín Hernández Ortega

Tutor TFG:

Daniel Garcia Almiñana

June 2014

[This page intentionally left blank]

Contents

List of Figures

List of Tables

APPENDIX A: NACA 2412 airfoil model

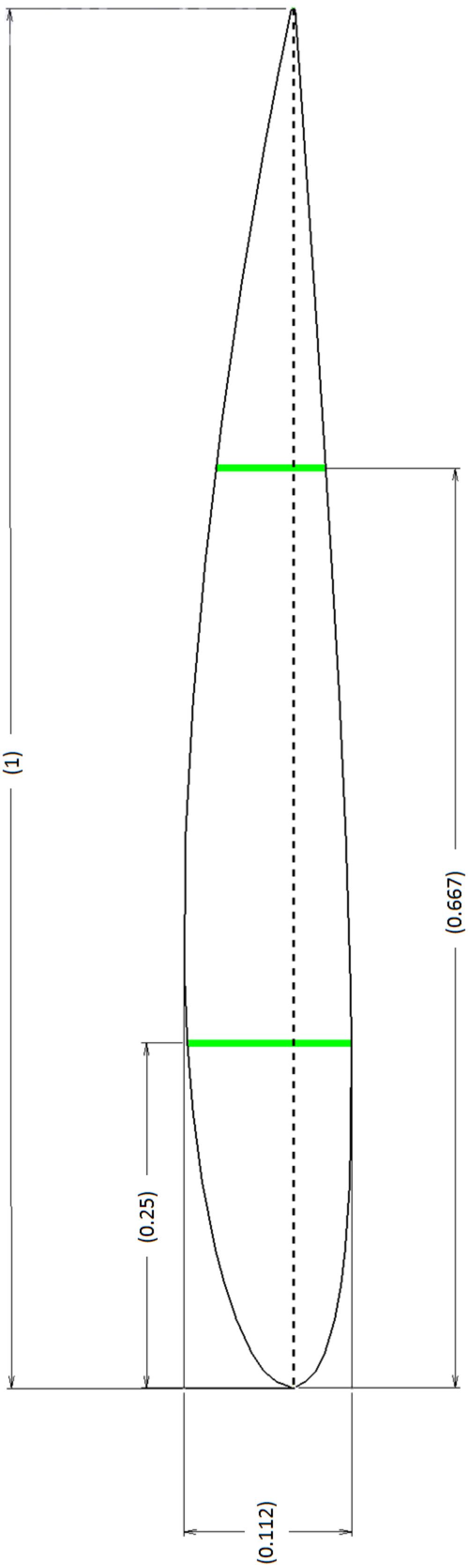


Figure 1: CAD drawing of NACA 2412 Airfoil. Measures are dimensionless.

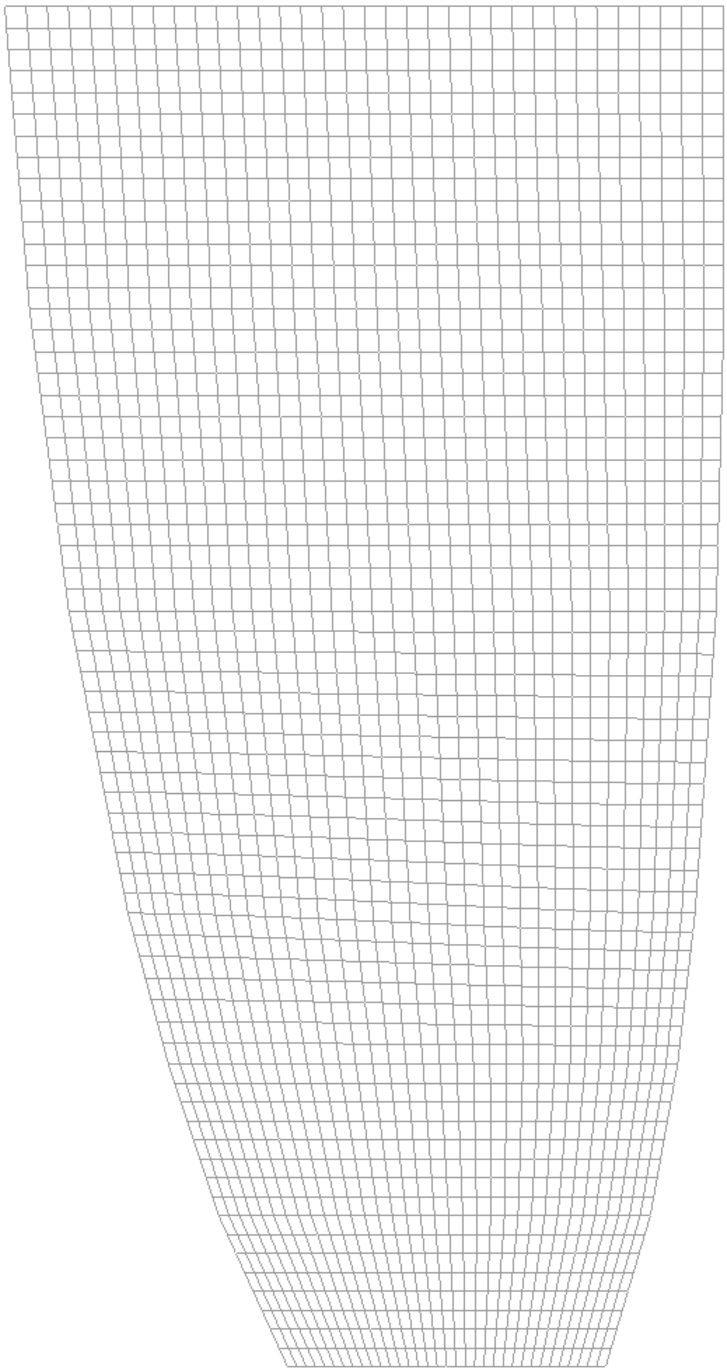


Figure 2: Leading Edge mesh of NACA 2412 airfoil (elements: 2345, nodes: 2244).

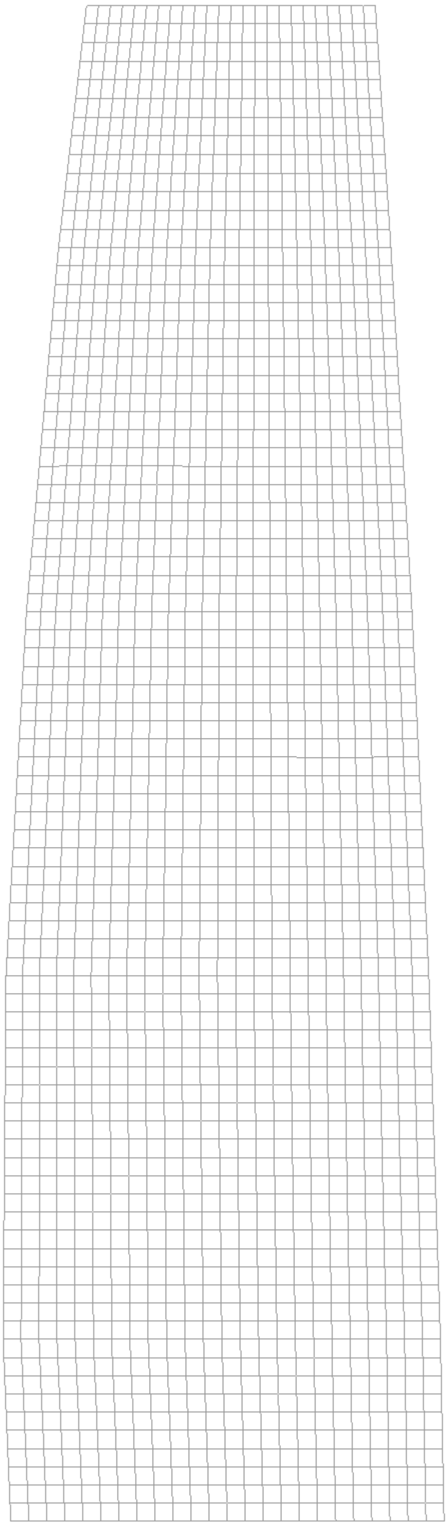


Figure 3: Wing Box mesh of NACA 2412 airfoil (elements: 1992 , nodes: 2100).

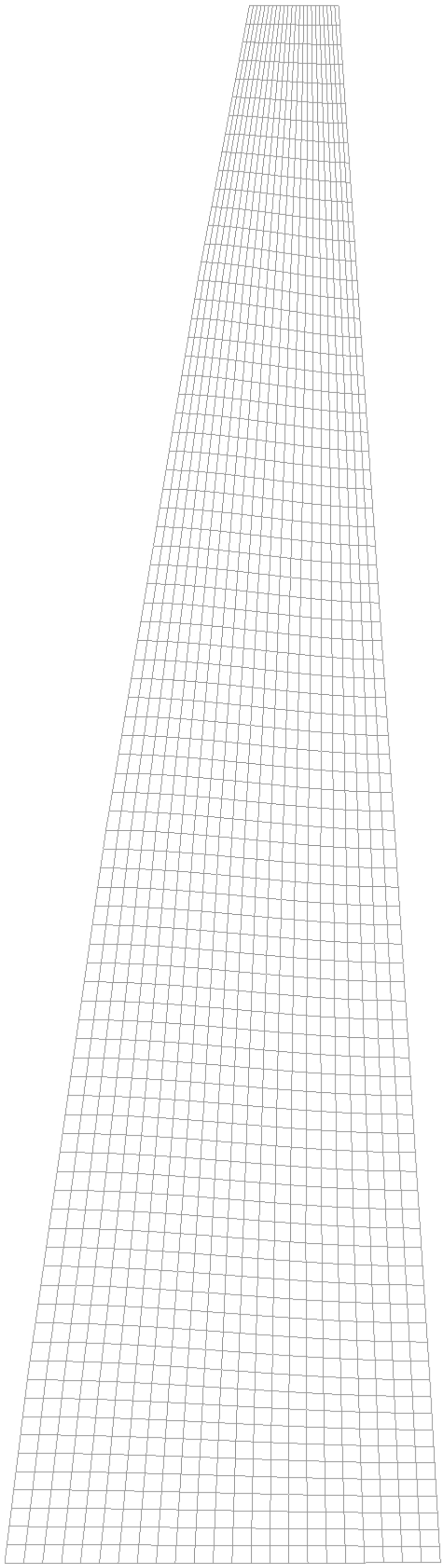


Figure 4: Trailing Edge mesh of NACA 2412 airfoil (elements: 1909, nodes: 2016).

APPENDIX B: XFLR5 Data

Table 1: XFLR5 Direct Analysis Data.

XFLR5 Data for case AoA = 5°	**XFLR5 Data for case AoA = 18°**
Re = 7200000	Re = 7200000
Alpha = 5.00°	Alpha = 18.00°
Mach = 0.200	Mach = 0.200
NCrit = 9.0	NCrit = 9.0
CL = 0.81526	CL = 1.86543
CD = 0.00642	CD = 0.03885
Cm = -0.05221	Cm = -0.00087
Cdp = 0.00224	Cdp = 0.03515
Cpmn = -1.75414	Cpmn = -14.36142
XCP = 0.30784	XCP = 0.22191
Top Transition = 0.10861	Top Transition = 0.00559
Bot Transition = 0.92855	Bot Transition = 1.00000
Number of panels = 199	Number of panels = 199

Table 2: Pressure coefficient distribution at extrados. AoA = 5°.

X	Cpe	X	Cpe	X	Cpe
0,0000	-0,3750	0,2494	-1,0570	0,7868	-0,2520
0,0002	-0,8200	0,2611	-1,0370	0,8005	-0,2330
0,0008	-1,2110	0,2730	-1,0160	0,8140	-0,2140
0,0017	-1,5110	0,2851	-0,9950	0,8273	-0,1950
0,0030	-1,7150	0,2974	-0,9740	0,8404	-0,1750
0,0047	-1,8370	0,3099	-0,9530	0,8533	-0,1550
0,0067	-1,8980	0,3225	-0,9310	0,8660	-0,1340
0,0092	-1,9160	0,3353	-0,9090	0,8783	-0,1130
0,0119	-1,9070	0,3482	-0,8860	0,8904	-0,0920
0,0150	-1,8820	0,3613	-0,8630	0,9022	-0,0700
0,0185	-1,8470	0,3746	-0,8400	0,9136	-0,0470
0,0223	-1,8070	0,3880	-0,8150	0,9246	-0,0230
0,0264	-1,7660	0,4015	-0,7870	0,9352	0,0020
0,0309	-1,7240	0,4152	-0,7590	0,9454	0,0290
0,0357	-1,6840	0,4289	-0,7360	0,9550	0,0570
0,0408	-1,6450	0,4428	-0,7130	0,9641	0,0870
0,0463	-1,6080	0,4568	-0,6910	0,9725	0,1190
0,0520	-1,5730	0,4709	-0,6690	0,9802	0,1560
0,0581	-1,5400	0,4850	-0,6480	0,9871	0,1970
0,0645	-1,5080	0,4993	-0,6280	0,9929	0,2480
0,0712	-1,4780	0,5136	-0,6070	0,9975	0,3110
0,0781	-1,4500	0,5280	-0,5870	1,0000	0,4390
0,0854	-1,4230	0,5424	-0,5680		
0,0930	-1,3980	0,5569	-0,5480		
0,1008	-1,3730	0,5714	-0,5290		
0,1090	-1,3490	0,5860	-0,5100		
0,1174	-1,3260	0,6005	-0,4910		
0,1261	-1,3040	0,6151	-0,4730		
0,1350	-1,2820	0,6297	-0,4540		
0,1442	-1,2610	0,6442	-0,4360		
0,1537	-1,2400	0,6588	-0,4170		
0,1634	-1,2190	0,6733	-0,3990		
0,1733	-1,1990	0,6878	-0,3810		
0,1835	-1,1790	0,7022	-0,3630		
0,1939	-1,1590	0,7165	-0,3440		
0,2046	-1,1380	0,7308	-0,3260		
0,2155	-1,1180	0,7450	-0,3080		
0,2266	-1,0980	0,7591	-0,2890		
0,2379	-1,0780	0,7730	-0,2710		

Table 3: Pressure coefficient distribution at intrados. AoA = 5°.

X	Cpi	X	Cpi	X	Cpi
0,0000	-0,3750	0,2494	0,1520	0,7868	0,1420
0,0002	0,0580	0,2611	0,1490	0,8005	0,1440
0,0008	0,4360	0,2730	0,1470	0,8140	0,1460
0,0017	0,7210	0,2851	0,1450	0,8273	0,1490
0,0030	0,9010	0,2974	0,1430	0,8404	0,1510
0,0047	0,9900	0,3099	0,1410	0,8533	0,1540
0,0067	1,0100	0,3225	0,1390	0,8660	0,1580
0,0092	0,9870	0,3353	0,1380	0,8783	0,1610
0,0119	0,9390	0,3482	0,1360	0,8904	0,1650
0,0150	0,8790	0,3613	0,1350	0,9022	0,1700
0,0185	0,8150	0,3746	0,1330	0,9136	0,1750
0,0223	0,7520	0,3880	0,1310	0,9246	0,1810
0,0264	0,6920	0,4015	0,1270	0,9352	0,1880
0,0309	0,6360	0,4152	0,1240	0,9454	0,1960
0,0357	0,5840	0,4289	0,1220	0,9550	0,2050
0,0408	0,5370	0,4428	0,1210	0,9641	0,2160
0,0463	0,4950	0,4568	0,1210	0,9725	0,2290
0,0520	0,4570	0,4709	0,1200	0,9802	0,2460
0,0581	0,4220	0,4850	0,1200	0,9871	0,2680
0,0645	0,3910	0,4993	0,1200	0,9929	0,2980
0,0712	0,3630	0,5136	0,1200	0,9975	0,3420
0,0781	0,3380	0,5280	0,1210	1,0000	0,4390
0,0854	0,3160	0,5424	0,1210		
0,0930	0,2950	0,5569	0,1220		
0,1008	0,2770	0,5714	0,1230		
0,1090	0,2610	0,5860	0,1240		
0,1174	0,2460	0,6005	0,1240		
0,1261	0,2330	0,6151	0,1250		
0,1350	0,2210	0,6297	0,1260		
0,1442	0,2100	0,6442	0,1280		
0,1537	0,2010	0,6588	0,1290		
0,1634	0,1930	0,6733	0,1300		
0,1733	0,1850	0,6878	0,1310		
0,1835	0,1780	0,7022	0,1320		
0,1939	0,1720	0,7165	0,1340		
0,2046	0,1670	0,7308	0,1350		
0,2155	0,1630	0,7450	0,1370		
0,2266	0,1590	0,7591	0,1390		
0,2379	0,1550	0,7730	0,1400		

Table 4: Pressure coefficient distribution at extrados. AoA = 18°.

X	Cpe	X	Cpe	X	Cpe
0,0000	-20,1250	0,2494	-2,3810	0,7868	-0,3910
0,0002	-21,3170	0,2611	-2,2990	0,8005	-0,3590
0,0008	-21,3770	0,2730	-2,2200	0,8140	-0,3270
0,0017	-20,4810	0,2851	-2,1440	0,8273	-0,2950
0,0030	-19,0170	0,2974	-2,0700	0,8404	-0,2630
0,0047	-17,3320	0,3099	-1,9980	0,8533	-0,2310
0,0067	-15,6530	0,3225	-1,9270	0,8660	-0,1990
0,0092	-14,0990	0,3353	-1,8590	0,8783	-0,1660
0,0119	-12,7140	0,3482	-1,7920	0,8904	-0,1340
0,0150	-11,5040	0,3613	-1,7260	0,9022	-0,1010
0,0185	-10,4570	0,3746	-1,6610	0,9136	-0,0680
0,0223	-9,5520	0,3880	-1,5960	0,9246	-0,0340
0,0264	-8,7690	0,4015	-1,5280	0,9352	0,0010
0,0309	-8,0900	0,4152	-1,4630	0,9454	0,0370
0,0357	-7,4980	0,4289	-1,4040	0,9550	0,0730
0,0408	-6,9800	0,4428	-1,3490	0,9641	0,1120
0,0463	-6,5240	0,4568	-1,2950	0,9725	0,1530
0,0520	-6,1190	0,4709	-1,2440	0,9802	0,1960
0,0581	-5,7590	0,4850	-1,1940	0,9871	0,2450
0,0645	-5,4370	0,4993	-1,1460	0,9929	0,3010
0,0712	-5,1470	0,5136	-1,1000	0,9975	0,3680
0,0781	-4,8840	0,5280	-1,0550	1,0000	0,4980
0,0854	-4,6450	0,5424	-1,0110		
0,0930	-4,4270	0,5569	-0,9680		
0,1008	-4,2260	0,5714	-0,9270		
0,1090	-4,0420	0,5860	-0,8860		
0,1174	-3,8710	0,6005	-0,8460		
0,1261	-3,7120	0,6151	-0,8080		
0,1350	-3,5640	0,6297	-0,7700		
0,1442	-3,4250	0,6442	-0,7320		
0,1537	-3,2950	0,6588	-0,6960		
0,1634	-3,1720	0,6733	-0,6600		
0,1733	-3,0560	0,6878	-0,6250		
0,1835	-2,9450	0,7022	-0,5900		
0,1939	-2,8410	0,7165	-0,5560		
0,2046	-2,7410	0,7308	-0,5220		
0,2155	-2,6450	0,7450	-0,4890		
0,2266	-2,5540	0,7591	-0,4560		
0,2379	-2,4660	0,7730	-0,4230		

Table 5: Pressure coefficient distribution at intrados. AoA = 18°.

X	Cpi	X	Cpi	X	Cpi
0,0000	-20,1250	0,2494	0,7890	0,7868	0,4130
0,0002	-18,0800	0,2611	0,7750	0,8005	0,4070
0,0008	-15,2760	0,2730	0,7610	0,8140	0,4020
0,0017	-12,1620	0,2851	0,7470	0,8273	0,3960
0,0030	-9,2180	0,2974	0,7340	0,8404	0,3910
0,0047	-6,7180	0,3099	0,7210	0,8533	0,3860
0,0067	-4,7340	0,3225	0,7080	0,8660	0,3810
0,0092	-3,2190	0,3353	0,6960	0,8783	0,3760
0,0119	-2,0870	0,3482	0,6840	0,8904	0,3720
0,0150	-1,2490	0,3613	0,6720	0,9022	0,3680
0,0185	-0,6300	0,3746	0,6600	0,9136	0,3650
0,0223	-0,1750	0,3880	0,6480	0,9246	0,3620
0,0264	0,1610	0,4015	0,6360	0,9352	0,3600
0,0309	0,4090	0,4152	0,6240	0,9454	0,3590
0,0357	0,5910	0,4289	0,6130	0,9550	0,3590
0,0408	0,7250	0,4428	0,6020	0,9641	0,3600
0,0463	0,8220	0,4568	0,5920	0,9725	0,3640
0,0520	0,8920	0,4709	0,5820	0,9802	0,3700
0,0581	0,9400	0,4850	0,5720	0,9871	0,3800
0,0645	0,9730	0,4993	0,5620	0,9929	0,3970
0,0712	0,9940	0,5136	0,5530	0,9975	0,4280
0,0781	1,0060	0,5280	0,5440	1,0000	0,4980
0,0854	1,0100	0,5424	0,5360		
0,0930	1,0090	0,5569	0,5270		
0,1008	1,0040	0,5714	0,5190		
0,1090	0,9950	0,5860	0,5110		
0,1174	0,9850	0,6005	0,5030		
0,1261	0,9720	0,6151	0,4950		
0,1350	0,9590	0,6297	0,4870		
0,1442	0,9440	0,6442	0,4800		
0,1537	0,9290	0,6588	0,4730		
0,1634	0,9130	0,6733	0,4660		
0,1733	0,8970	0,6878	0,4590		
0,1835	0,8810	0,7022	0,4520		
0,1939	0,8650	0,7165	0,4450		
0,2046	0,8500	0,7308	0,4380		
0,2155	0,8340	0,7450	0,4320		
0,2266	0,8190	0,7591	0,4260		
0,2379	0,8040	0,7730	0,4190		

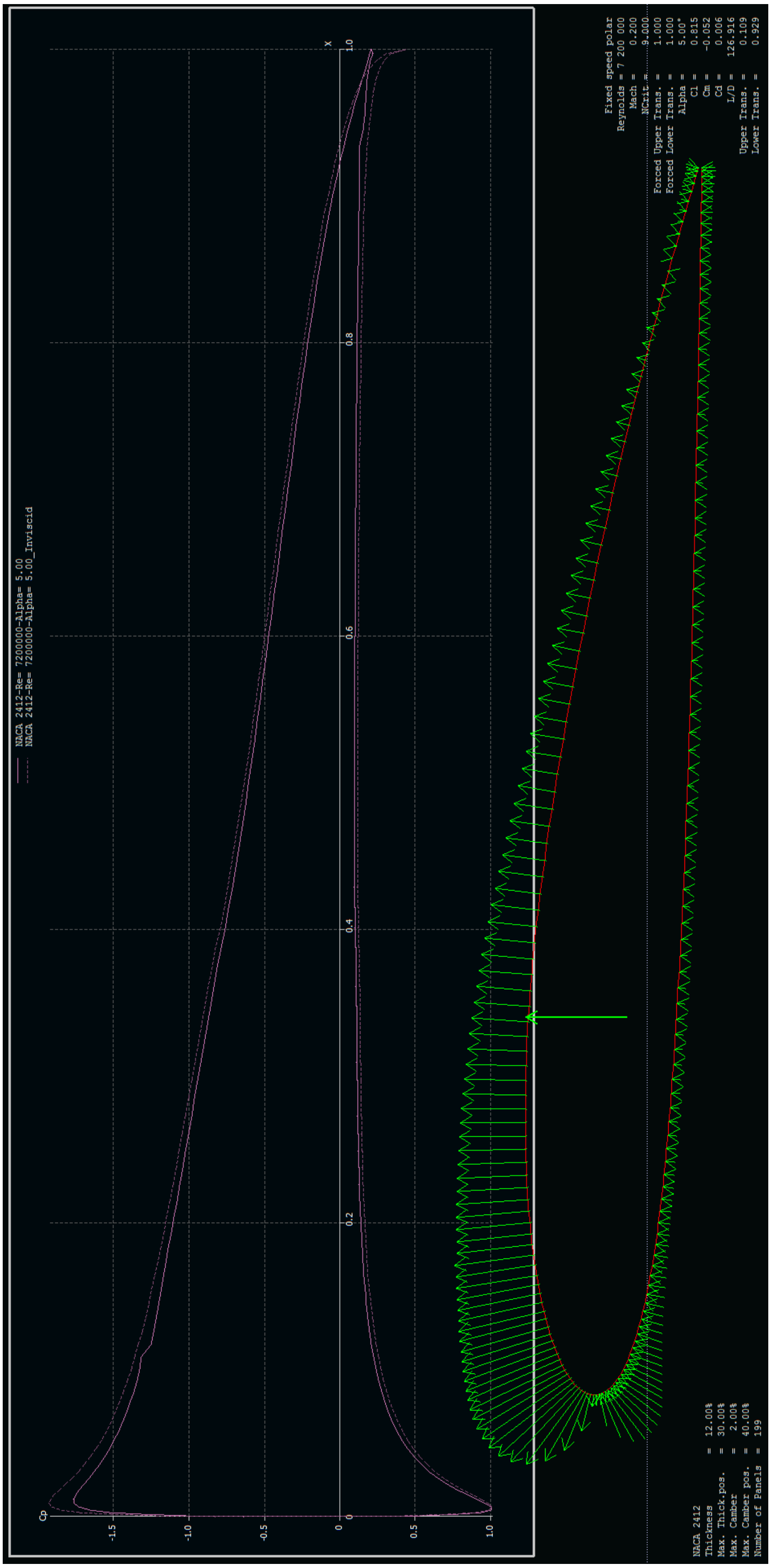


Figure 5: XFLR5 Cp Graph and pressure coefficient distribution on the airfoil at AoA=5°

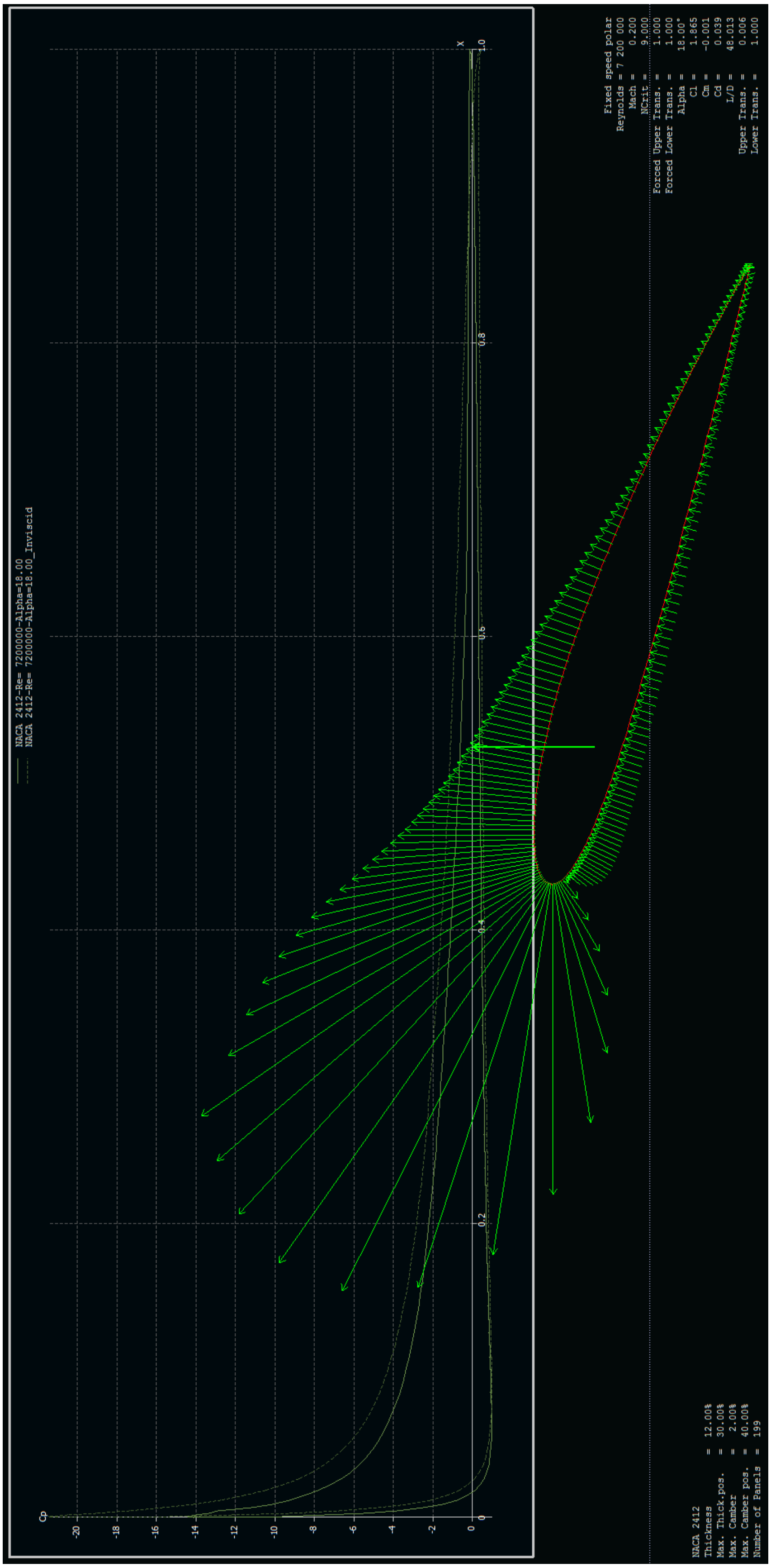


Figure 6: XFLR5 Cp Graph and pressure coefficient distribution on the airfoil at AoA=18°

APPENDIX C: MATLAB Code

0.1 MAIN FILE

This is the file were input parameters are introduced and it calls the core function: STRUCTOPOPT. This file also reads the optimal density vector and plots it.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Author: Alexandre Cortiella Segarra
%% Institution: Universitat Politecnica de Catalunya (UPC)/
    ETSEIAT
%% Project: Treball Final de Grau (Bachelor's Degree Final
    Project)
%% Date: 22/06/2014
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%---------------------------------------------------------
%% INPUT PARAMETERS
%---------------------------------------------------------



%---------- MESH GENERATION ----------

filename ='NACA2412_WingBox_(0.25)_0.01.xlsx';
coordinates = xlsread(filename,1,'A1:B510')';
connectivity = xlsread(filename,2,'A1:D457')';
thickness = 1;

%Draw the mesh with node numbers
draw_mesh_Q4(coordinates, connectivity);
pause
```

```
%Storage in a vector of volumes of each element
[volume] = volume(coordinates, connectivity, thickness);



%----------- MATERIAL PROPERTIES -----------
Young = 1;
nu = 0.3;

%----------- PROBLEM TYPE -----------

%Select between PLANESTRESS or PLANESTRAIN
problemtype = 'PLANESTRESS';

%----------- BOUNDARY CONDITIONS -----------

%The user needs to introduce the boundary conditions by hand.

%DIRICHLET B.C.

%In this case, the constrained displacements are set to 0.

%Dirichlet_BC = (3, nodes where displacement is imposed)
%[columns] = [node; 1 or 2 (x or y); value]
%[rows] = nodes where displacement is imposed

fixedNodes = [1 3 6 11 17 25 36 48 61 500 497 496 498 501 502 503
     505 506 507 508 509 510];

Dirichlet_BC(1,1:2:2*length(fixedNodes))=fixedNodes;
Dirichlet_BC(1,2:2:2*length(fixedNodes))= fixedNodes;
Dirichlet_BC(2,1:2:2*length(fixedNodes))=ones(1,length(fixedNodes
    ));
Dirichlet_BC(2,2:2:2*length(fixedNodes))=2*ones(1,length(
    fixedNodes));
Dirichlet_BC(3,:)=zeros(1,2*length(fixedNodes));

%NEUMANN B.C.

Neumann_BC=[];

%----------- LOADS -----------

%NODAL LOADS

%[columns] = [node; Fx; Fy]';
Nodal_Loads = [289 0 1]';
```

```
%BODY LOADS

%b = [bx; by];

b=[];

%---------- SIMP PROPERTIES ----------

Vf = 0.5;%Volume fraction
p = 3; % SIMP penalization parameter
rho_0 = rand*ones(size(connectivity,2), 1);%Initial density
    vector

%---------- SOLVER PARAMETERS ----------

tolerance = 1e-3;%Input tolerance
maxiter = 100;%Maximum allowed number of iterations

%---------- FILTER PARAMETERS ----------

rmin= 0.01;%Sensitivity filter radius
[H] = H_Filter(coordinates, connectivity, rmin);%Preparation of
    filter radius

%----------------------------------------------------------
%% STRUCTURAL TOPOLOGY OPTIMIZATION
%----------------------------------------------------------

%Start topology optimization code
[rho, StressElement, StrainElement, StressVM, fo_store, k] =
    STRUCTOPOPT(rho_0, p, Vf, coordinates, connectivity, Young,
    nu, b, Neumann_BC, Dirichlet_BC, Nodal_Loads, problemtype,
    tolerance, maxiter, volume, thickness, H);

%Plot optimal density distribution
figure(1);
plot_density(coordinates,connectivity,rho);
```

Listing 1: STRUCTOPOPT_GID_MAIN.m

0.2 STRUCTOPOPT

This is the main function of the program. It calls the three main modules (FEM MODULE, SENSITIVITY ANALYSIS ans OPTIMIZATION MODULE) and then gives the optimal solution.

```
function [rho, StressElement, StrainElement, StressVM, fo_store,
    k] = STRUCTOPOPT_TEST_GID(rho_0, p, Vf, coordinates,
    connectivity, Young, nu, b, Neumann_BC, Dirichlet_BC,
    Nodal_Load, problemtype, tolerance, maxiter, volume,
    thickness, H)

%------------DESCRIPTION------------

%INPUT DATA

% rho_0: initial SIMP design variables/densities
% p: SIMP penalty parameter
% Vf: volume fraction (fraction of V0 to be reduced)
% coordinates: x and y coordinates of each global node
% connectivity: connectivity table. Global node number =
    connectivity(local node number, element)
% Young: Young's modulus of the material
% nu: Poisson's modulus of the material
% b: body forces, force per unit volume
% Neumann_BC: prescribed tractions along the boundary of the
    domain
% Dirichlet_BC: prescribed nodal displacements on the domain
% Nodal_Load: nodal point force vector
% problemtype: selection between 'PLANESTRESS' or 'PLANESTRAIN'
    for 2D linear elasticity
% tolerance: stopping criterion by tolerance
% maxiter: stopping criterion by maximum number of iterations

%OUTPUT DATA

% rho: updated design variables
% StrainElement:elemental strains
% StressElement: elemental stresses
% StressVM: Von Misses stresses

%----------PRELOCATION AND PARAMETERS-----------
nelem = length(rho_0);%element number
rho_store = zeros(nelem,maxiter);%Storage of density vector at
    iteration k
fo_store = zeros(1, maxiter);%Storage of objective function at
```

```
    iteration k
rho_k = rho_0;%initialize density vector
k = 1;%initialize iteration
rho_new = rho_k;

%----------GID POSTPROCESSING-------------------
selem.conec = connectivity';
selem.matno = ones(1,nelem);
selem.etype = 'Quadrilateral';
fname = 'Topology_Optimization';
istep=1;
gid_write_msh(fname,istep,coordinates',selem);
[fid]=gid_write_headerpost(fname,istep);
nameres='Density';
%-----------------------------------------------

%Initialize iterations
    while (norm(rho_new - rho_k)/norm(rho_k) >= tolerance && k <=
         maxiter) || k == 1


        rho_k = rho_new;
        rho_store(:,k)= rho_k;

        %GID Postprocessing
        iteration = k;
        gid_write_density(fid,'Density',iteration,rho_k); %SIMP
            element densities

        %FINITE ELEMENT ANALYSIS

        [u,ue,StrainElement,StressElement,StressVM,F, Ke0] =
            FEM_MODULE(coordinates,connectivity,Young,nu,
            thickness,rho_k,p,b,Neumann_BC,Dirichlet_BC,
            Nodal_Load, problemtype);

        %GID Postprocessing
        %stress = [StressElement; zeros(1,nelem)];
        stress = StressElement';
        gid_write_tensorfield(fid,'Stress',iteration,stress);%
            Stress field

        %SENSITIVITY ANALYSIS

        [DfoDrho,~,fo,~] = SENSITIVITY_MODULE_FILTER(rho_k, p, u,
             ue, Ke0, F, volume, Vf, H);
        fo_store(1,k) = fo; %Storage of the compliance at
```

```
            iteration k

        %OPTIMIZATION ALGORITHM (UPDATE OF DESIGN VARIABLES)

        [rho_new] = OPTIMIZATION_MODULE_OC(rho_k,DfoDrho, volume,
             Vf);

        %New iteration k + 1

        k = k + 1;


    end

    rho = rho_new;
    compliance = fo;

    fprintf('**********  OPTIMAL VALUE FOUND  **********\n\n');
    fprintf('Optimal rho: %6.5f\n',rho);
    fprintf('Minimum value of compliance: %6.5f\n',compliance);
end
```

Listing 2: STRUCTOPOPT.m

0.2.1 FINITE ELEMENT MODULE

```
function [u,ue,StrainElement,StressElement,StressVM,F, Ke0] =
    FEM_MODULE(coordinates,connectivity,Young,nu,thickness,rho,p,
    b,Neumann_BC,Dirichlet_BC, Nodal_Load, problemtype)

%----------DESCRIPTION----------------
%This function performs a finite element analysis in order to
%obtain the displacement field, stresses and strains and other
%parameters needed fot the topology optimization algorithm

%OUTPUT DATA

% u: nodal displacements
% ue:elemental/local displacements
% StrainElement:elemental strains
% StressElement: elemental stresses
% StressVM: Von Misses stresses
% F: external load vector
% volume: volume/area of each element
```

```
%INPUT DATA

% coordinates: x and y coordinates of each global node
% connectivity: connectivity table. Global node number =
    connectivity(local node number, element)
% Young: Young's modulus of the material
% nu: Poisson's modulus of the material
% rho: SIMP design variables/densities
% p: SIMP penalty parameter
% Ke0: elemental stiffness matrix without being penalized by SIMP
% b: body forces, force per unit volume
% Neumann_BC: prescribed tractions along the boundary of the
    domain
% Dirichlet_BC: prescribed nodal displacements on the domain
% Fnodal: nodal point force vector
% problemtype: selection between 'PLANESTRESS' or 'PLANESTRAIN'
    for 2D linear elasticity

%----------------------NOMENCLATURE----------------

%nnod: number of global nodes
nnod=size(coordinates, 2);

%nd: space dimensions (nd=2 in 2D, nd=3 in 3D)
nd=size(coordinates,1);

%nnodel: number of local nodes per element
nnodel=size(connectivity,1);

%nelem: number of elements
nelem=size(connectivity,2);

%ndofel: number of degrees of freedom per element
ndofel=nd*nnodel;

%ndof: number of global degrees of freedom
ndof=nd*nnod;

%----------------------PRELOCATION and PARAMETERS----------------

u = zeros(ndof,1);

%Lame coefficients
lambda=nu*Young/((1+nu)*(1-2*nu));
mu=Young/(2*(1+nu));
%------------------------------------------------------------------
```

```
%Selection of problem-type: PLANESTRAIN or PLANESTRESS

    selectionstress = strcmpi(problemtype,'PLANESTRESS');
    selectionstrain = strcmpi(problemtype,'PLANESTRAIN');

    while selectionstress == 0  && selectionstrain == 0

        problemtype = input('Please select problem type:
            PLANESTRESS or PLANESTRAIN\n\n','s');
        selectionstress = strcmpi(problemtype,'PLANESTRESS');
        selectionstrain = strcmpi(problemtype,'PLANESTRAIN');
    end

    if selectionstress == 1

        Celastic = Young/(1-nu^2)*[1 nu 0; nu 1 0; 0 0 0.5*(1-nu)
            ];%Plane stress

    else

        Celastic=mu*[2,0,0;0,2,0;0,0,1]+lambda
            *[1,1,0;1,1,0;0,0,0];%Plane strain

    end

%------------------------------------------------------------------
%------------------------------------------------------------------

%ASSEMBLY OF THE STIFFNESS MATRIX
[K, Belement, Ke0] = K_assembly(coordinates,connectivity,Celastic
    , thickness, rho, p);

%ASSEMBLY OF THE LOAD VECTOR
[F] = Load_Vector(coordinates, connectivity, Neumann_BC,
    Nodal_Load, b);

%PARTITION OF THE SYSTEM (STIFNESS MATRIX AND LOAD VECTOR)

% |Kff     Kfc| |uf|   |Ff|     |0|
% |           |*|  | = |  |  +  | |
% |Kcf     Kcc| |g |   |Fc|     |R|

GlobalDof = 1:ndof;
%Extraction of constrained nodes and displacements from
%the given matrix Dirichlet_BC
[uconstrained,g] = Constrained_displacements(Dirichlet_BC);
ufree = setxor(GlobalDof,uconstrained);
```

```
%Partition of the stiffness matrix
Kff = K(ufree,ufree);
Kfc = K(ufree, uconstrained);
%Kcf = K(uconstrained, ufree);
%Kcc = K(uconstrained, uconstrained);

%Partition of the load vector
Ff = F(ufree);
%Fc = F(uconstrained);

%System of equations to solve

uf = Kff\(Ff - Kfc*g); %Free displacements
%R = Kcf*uf + Kcc*g - Fc;%Reactions

%Global displacement vector
u(uconstrained) = g;
u(ufree) = uf;

%STRESSES, STRAINS AND LOCAL DISPLACEMENTS
[StrainElement, StressElement, StressVM, ue] =
    Stresses_And_Strains(connectivity, Belement, u, Celastic,
    nelem, ndofel);

end
```

Listing 3: FEM_MODULE.m

Note: The load vector assembly code was not added in this appendix because its generation depends on the way the Dirichlet, Neumann and body load conditions are given. The user must create a routine to generate the global load vector.

0.2.1.1 Stiffness matrix assembly

```
function [K, Belement, Ke0] = K_assembly(coordinates,connectivity
    ,Celastic, thickness, rho, p)

%-----------------------DESCRIPTION-----------------------
%Given coordinates and connectivity matrices and the elastic
%coeficients (Young and Poisson), this function returns the
%globalstiffness matrix.

%-----------------------INPUT PARAMETERS------------------
```

```
%coordinates = [space dimension, global nodes] gives the
    coordinates of global nodes

%connectivity = [local nodes, elements] gives the global node
    number of the
%local node a of element e

%Celastic: Constitutive 2D matrix (PLANESTRAIN or PLANESTRESS)

%---------------------PRELOCATION PARAMETERS----------------
coordn = zeros(nnodel,nd);%rows are the local nodes and columns
    correspond to the space dimension, all for each element
Belement = zeros(3,ndofel,4,nelem);
elementDof=zeros(ndofel,nelem);% local degrees of freedom per
    element elementDof=[nel, nnodel*nd]


%ASSEMBLING ALGORITHM

% compute stiffness matrix
% for plane stress Q4 elements

K_all=zeros(ndofel*ndofel,nelem);
Ke0 = zeros(ndofel, ndofel, nelem);
Ke = zeros(ndofel);

% 2 by 2 quadrature
[GaussWeights,GaussPoints]=GaussQuadratureQ4;

elementDof(1:2:ndofel,:)=2*connectivity - 1;% in case of a
    quadrilater Q4 elementDof(:,[1 3 5 7])=2*lnods'-1
elementDof(2:2:ndofel,:)=2*connectivity;% in case of a
    quadrilater Q4 elementDof(:,[2 4 6 8])=2*lnods'

%Start element loop

for e=1:nelem

    coordn([1 2 3 4],1)= coordinates(1, connectivity([1 2 3 4],e)
        );
    coordn([1 2 3 4],2)= coordinates(2, connectivity([1 2 3 4],e)
        );

    % Start Gauss points loop

    for gp=1:size(GaussWeights,1)%loop for Gauss Points
```

```
    xi=GaussPoints(gp,1);
    eta=GaussPoints(gp,2);

    % shape functions and derivatives
    [~,NaturalDerivatives]=ShapeFunctionQ4(xi,eta);

    % Jacobian matrix, inverse of Jacobian,
    % derivatives w.r.t. x,y
    [detJacobian,DNDxy]= Jacobian(coordn,NaturalDerivatives);


    % B matrix
    B=zeros(3,ndofel);
    B([1,3],[1,3,5,7]) = DNDxy;
    B([3,2],[2,4,6,8]) = DNDxy;
    Belement(:,:,gp,e) = B;

    Ke = Ke + thickness*B'*Celastic*B*GaussWeights(gp)*
        detJacobian;


    end %end of Gauss Point loop

    % Storage in columns of element stiffness matrices with
        density
    % penalization
    K_all(:,e) =  rho(e)^p*Ke(:);

    %Storage of non-penalized element stiffness matrices
    Ke0(:,:,e) = Ke;

    Ke = zeros(ndofel);

end%end of element loop

indx_j = repmat(1:ndofel,ndofel,1);
indx_i = indx_j';
Ki = elementDof(indx_i(:),:);
Kj = elementDof(indx_j(:),:);

K = sparse(Ki,Kj,K_all);
```

Listing 4: K_assembly.m

Q4 Gauss quadrature rule

```
function [weights, points] = GaussQuadratureQ4

points=...
[ -0.577350269189626 -0.577350269189626;
0.577350269189626 -0.577350269189626;
0.577350269189626 0.577350269189626;
-0.577350269189626 0.577350269189626];

weights=[ 1;1;1;1];

end
```

Listing 5: GaussQuadratureQ4.m

Q4 Shape functions

```
function [ShapeFunctions,NaturalDerivatives]=ShapeFunctionQ4(xi,
    eta)

%This function returns the shape functions and its derivatives
%w.r.t the natural variables.

ShapeFunctions = 1/4*[(1-xi)*(1-eta); (1+xi)*(1-eta); (1+xi)*(1+
    eta); (1-xi)*(1+eta)];

NaturalDerivatives = 1/4*[-(1-eta), -(1-xi); 1-eta, -(1+xi); 1+
    eta, 1+xi; -(1+eta), 1-xi];

end
```

Listing 6: ShapeFunctionQ4.m

Jacobian

```
function [detJacobian, CartesianDerivatives]=Jacobian(coordn,
    NaturalDerivatives)
%This function returns the Jacobian, the inverse of the Jacobian
%and the derivatives of the shape functions w.r.t. the x and
%y cartesian coordinates.


%Jacobian=[2,2]
```

```
%invJacobian=[2,2]
%CartesianDerivatives=[4,2]

Jacobian=NaturalDerivatives'*coordn;

detJacobian = Jacobian(1,1)*Jacobian(2,2)-Jacobian(2,1)*Jacobian
    (1,2);% determinant of the Jacobian (detJ = J11*J22 - J21*J12
    )

invJacobian = 1/detJacobian*[Jacobian(2,2), -Jacobian(1,2); -
    Jacobian(2,1), Jacobian(1,1)]; %invJ = 1/detJ * [J22, -J12; -
    J21, J11]

CartesianDerivatives = invJacobian*NaturalDerivatives';
```

Listing 7: Jacobian.m

0.2.1.2 Constrained displacements

```
function [uconstrained, g] = Constrained_displacements(
    Dirichlet_BC)

%This function arranges the constrained nodal displacements  to
%split the displacement vector u and gives the g values.

g = zeros(size(Dirichlet_BC,2),1);
uconstrained = zeros(size(Dirichlet_BC,2),1);

    if ~isempty(Dirichlet_BC)

        %loop which goes over Dirichlet_BC matrix and takes
        %columns that correspond to constrained displacements
        for ndof=1:size(Dirichlet_BC,2)

            %g([2*Dirichlet_BC(1,n) - 1; 2*Dirichlet_BC(1,n)]) =
            % = [Dirichlet_BC(2,n); Dirichlet_BC(3,n)];
            g(ndof) = Dirichlet_BC(3,ndof);

            uconstrained(ndof) = 2*(Dirichlet_BC(1,ndof) - 1) +
                Dirichlet_BC(2,ndof);
        end
    end

end
```

Listing 8: Constrained_displacements.m

0.2.1.3 Strains, stresses and elemental displacements

```
function [Strain, Stress, StressVM, ue] = Stresses_And_Strains(
    connectivity, Belement, u, Celastic, nelem, ndofel)

%This function gives the strain and stress vectors as well as the
%elemental displacements

%Strain(:,element) = [ex; ey; exy]
%Stress(:,element) = [sigmax; sigmay; sigmaxy]
%StressVM(element) = [sigmaVM]

%------------PRELOCATION-------------
ngp = size(Belement,3);%number of Gauss points
Strain_Gauss_Points = zeros(3, ngp, nelem);
Stress_Gauss_Points = zeros(3, ngp, nelem);
Strain = zeros(3,nelem);
Stress = zeros(3,nelem);
StressVM = zeros(1, nelem);
ue = zeros(ndofel, nelem);
elementDof = zeros(ndofel,1);
%-----------------------------------


for e = 1:nelem

    elementDof(1:2:ndofel)=2*connectivity(:,e)-1;% in case of a
        quadrilater Q4 elementDof(:,[1 3 5 7])=2*lnods'-1
    elementDof(2:2:ndofel)=2*connectivity(:,e);% in case of a
        quadrilater Q4 elementDof(:,[2 4 6 8])=2*lnods'

    for gp = 1:ngp

    %Elemental strains at each Gauss Point
    Strain_Gauss_Points(:,e,gp) = Belement(:,:,gp,e)*u(elementDof
        );

    %Elemental stresses at each Gauss Point
    Stress_Gauss_Points(:,e,gp) = Celastic*Strain_Gauss_Points(:,
        e,gp);

    %Elemental averaged strains
    Strain(:,e) = Strain(:,e) + 0.25*Strain_Gauss_Points(:,e,gp);

    %Elemental averaged stresses
    Stress(:,e) = Stress(:,e) + 0.25*Stress_Gauss_Points(:,e,gp);

    %Von Misses Stresses
```

```
    term1 = 0.5*(Stress(1,e) + Stress(2,e));
    term2 = sqrt((0.5*(Stress(1,e) - Stress(2,e)))^2 + Stress(3,e
        )^2);

    Stress1 = term1 + term2;
    Stress2 = term1 - term2;

    StressVM(e) = sqrt(Stress1^2 + Stress2^2 - 2*Stress1*Stress2)
        ;

    end

    %Elemental displacements
    ue(:,e) = u(elementDof);

end

end
```

Listing 9: Stresses_And_Strains.m

0.2.2 SENSITIVITY MODULE

```
function [DfoDrho,DgcDrho,fo,gc] = SENSITIVITY_MODULE_FILTER(rho,
    p, u, ue, Ke0, F, volume, Vf, H)

%------------DESCRIPTION------------
%This module receives information from input parameters and FE
%module to give the objective and constraint functons and
%their derivatives evaluated at the current iteration density
%vector.

%INPUT DATA

% rho: SIMP design variables/densities
% p: SIMP penalty parameter
% u: nodal displacements
% ue:elemental/local displacements
% Ke0: elemental stiffness matrix without being penalized by SIMP
% F: external load vector
% volume: volume/area of each element
% V0: initial volume of the ground structure
% Vf: volume fraction (fraction of V0 to be reduced)

%OUTPUT DATA
```

```
% DfoDrho: sensitivity of the objective function/ compliance
% DgcDrho: sensitivity of the constraint function (volume
    constraint)
% fo: objective function evaluated at k-iteration rho
% gc: constraint function evaluated at k-iteration rho

%-----------PARAMETERS-------------

nelem = length(rho);
V0 = sum(volume);

%-----------PRELOCATION------------
%DfoDrho = (n. elements,1), there is only one constraint
DfoDrho = zeros(nelem,1);
%----------------------------------


%Sensitivity of the compliance/objective function
    for e=1:nelem

        %SENSITIVITY WITH NO DEPENDENT LOAD VECTOR ON RHO
        DfoDrho(e, 1) = -p*rho(e)^(p-1)*ue(:,e)'*Ke0(:,:,e)*ue(:,
            e);
    end

[DfoDrho] = SENSITIVITY_FILTER(DfoDrho, volume, rho, H);

%Sensitivity of the constraint
DgcDrho = volume'./V0;

%Objective function evaluated at k-iteration rho
fo = F'*u;

%constraint function evaluated at k-iteration rho
gc = rho'*volume' - Vf*V0;

end
```

Listing 10: SENSITIVITY_MODULE_FILTER.m

0.2.2.1 Sensitivity filter

```
function [DfoDrho] = SENSITIVITY_FILTER(DfoDrho, volume, rho, H)

%This function gives filters the sensitivity of the objective
%function.

Hk = sum(H,2);
DfoDrho = volume'./rho./Hk.*(H*(rho.*DfoDrho)./volume');%
    Sensitivity filtered

end
```

Listing 11: SENSITIVITY_FILTER.m

0.2.3 OPTIMIZATION MODULE

0.2.3.1 Optimality Criteria

```
function [rho_new] = OPTIMIZATION_MODULE_OC(rho_k,DfoDrho, volume
    , Vf)

%-----------DESCRIPTION-----------
%This module gives the new density vector using the Optimality
%Criteria (OC) method. The update process is performed
%by a bisection method.

%INPUT DATA

% rho_k: history of design variables at (outer) iteration k
% DfoDrho: sensitivity of the objective function/ compliance
% volume: element volume vector
% Vf: volume fraction

%OUTPUT DATA

%rho_new: design variable/density vector at next iteration k+1

l1 = 0; l2 = 100000000; move = 0.2;
V0 = sum(volume);

while ((l2-l1)/(l1+l2) > 1e-4)
    lmid = 0.5*(l2+l1);
    rho_new = max(0.001,max(rho_k-move,min(1,min(rho_k+move,rho_k
        .*sqrt(-DfoDrho./(lmid*volume'))))));
```

```
    if volume*rho_new - Vf*V0 > 0;
        l1 = lmid;
    else
        l2 = lmid;
    end
end
```

Listing 12: OPTIMIZATION_MODULE_OC.m

0.2.3.2 Method of Moving Asymptotes

Note: STRUCTOPOPT code must be modified slightly so that L and lambda values are updated at each iteration.

```
function [rho_new, L, lambda_opt] = OPTIMIZATION_MODULE_MMADUAL(
    rho_store, k, L, fo, volume, DfoDrho, Vf, lambda0)

%------------DESCRIPTION------------
%This module gives the new density vector using the method of
%moving asymptotes. The update process is performed by a dual
%method and the subproblem by a Newton method.

%INPUT DATA

% rho_store: history of design variables at (outer) iteration k
% k: iteration number
% fo: objective function evaluated at k-iteration rho
% DfoDrho: sensitivity of the objective function/ compliance
% lambda0: current Lagrange multiplier

%OUTPUT DATA

%rho_new: updated design variable/density vector
%L: updated lower asymptote
%lambda_new: updated Lagrange multiplier

%------------PRELOCATION and PARAMETERS------------
nelem = size(rho_store,1);
gamma = ones(nelem,1);
Sini=0.5;
change = 1;
nmax=300;
volume = volume';
V0 = sum(volume);
n=0;
```

```
rho_k = rho_store(:,k);

%Upper and lower bounds of the domain
rho_max = ones(nelem,1);
rho_min = 0.001*ones(nelem,1);% Suggested value to avoid
    numerical errors related with 0 denominators (xmin = 10^-3)

%Calculation of asymptotes at iterations = 1,2
    if k <= 3
        L = rho_k - Sini*(rho_max-rho_min);
    end

%Update asymptotes at iterations >= 3

    if k >= 4

        %Storage of old values of design variables
        rho_old1 = rho_store(:, k-1); %rho_(k-1)
        rho_old2 = rho_store(:, k-2); %rho_(k-2)

        signparam = (rho_k - rho_old1).*(rho_old1 - rho_old2);

        gamma(signparam > 0)= 1.2;
        gamma(signparam < 0) = 0.7;

        L = rho_k - gamma.*(rho_old1 - L);

    end

%Calculation of upper and lower bounds alpha and beta at each
    iteration k
s = 0.1;
temp = L + s*(rho_k - L);
alphak = max(temp,rho_min);

%--------APPROXIMATION OF THE OBJECTIVE FUNCTION-------

qo = (rho_k - L).^2.*max(0,-DfoDrho);
pqterm_obj = qo./(rho_k - L);
ro = fo - sum(pqterm_obj);

%-----------OPTIMIZATION BY A DUAL SUBPROBLEM-----------

lambda = lambda0;

while change>=1e-5 & n<=nmax
```

```
    rho_lambda = L + sqrt(qo./(volume.*lambda));
    rho_star = min(max(rho_lambda, alphak), rho_max);

    %SOLVING THE DUAL BY A NEWTON METHOD
    DdualDlamb = -Vf*V0 + sum(volume.*rho_star);
    DdualDlamb2 = sum(-0.5*lambda^(-3/2)*sqrt(qo.*volume));

    lambda_new = lambda - DdualDlamb/DdualDlamb2;

    change=abs(lambda_new-lambda);
    lambda=lambda_new;

    n=n+1;
end

lambda_opt = lambda;

rho_lambda = L + sqrt(qo./(volume.*lambda));

rho_star = min(max(rho_lambda, alphak), rho_max);

rho_new = rho_star;
```

Listing 13: OPTIMIZATION_MODULE_MMADUAL.m

0.3 GID DATA FILES GENERATION

0.3.1 Header

```
function [fid]=gid_write_headerpost(fname,istep)
% write header requered for the post
%
res_file = strcat(fname,'_',num2str(istep),'.flavia.res');
fid = fopen(res_file,'w');
fprintf(fid,'Gid Post Results File 1.0 \n');
fprintf(fid,'### \n');
fprintf(fid,'# ENG_AERO_COMP  V.1.0 \n');
fprintf(fid,'# \n');

fprintf(fid,['GaussPoints "My Gauss" ElemType Quadrilateral\n']);
fprintf(fid,['Number Of Gauss Points: 1\n']);
fprintf(fid,['Natural Coordinates: Internal\n']);
fprintf(fid,['End gausspoints\n']);
```

```
end
```

Listing 14: gid_write_headerpost.m

0.3.2 Scalar field

```
function [ ] = gid_write_sclfield(fid,nameres,time,sclfield)

nelem = size(sclfield,1);
s =['Result' ' "' nameres '" ' '"time"' '%12.5d' ' Scalar ' '
    OnGaussPoints ' ...
     '"' 'My Gauss' '"' '\n'];
fprintf(fid,s,time);
fprintf(fid,['Values \n']);
for i = 1 : nelem
   fprintf(fid,['%6.0i %12.5d \n'],i,sclfield(i));
end
fprintf(fid,['End Values \n']);
fprintf(fid,'# \n');

end
```

Listing 15: gid_write_sclfield.m

0.3.3 Vector field

```
function [ ] = gid_write_vfield(fid,nameres,time,vfield)

npnod = size(vfield,1);
s =['Result' ' "' nameres '" ' '"time"' '%12.5d' ' Vector OnNodes
     \n'];
fprintf(fid,s,time);
fprintf(fid,['Values \n']);
for i = 1 : npnod
   fprintf(fid,['%6.0i %13.5d %13.5d \n'],i,vfield(i,:));
end
fprintf(fid,['End Values \n']);
fprintf(fid,'# \n');

end
```

Listing 16: gid_write_vfield.m

0.3.4 Tensor field

```
function [ ] = gid_write_tensorfield(fid,nameres,time,tfield)

nelem = size(tfield,1);
nstre = size(tfield,2);
str = zeros(6,1);
s =['Result' ' "' nameres '" ' '"time" ' '%12.5d' ' Matrix ' '
    OnGaussPoints ' '"' 'My Gauss' '"' '\n'];
fprintf(fid,s,time);
s =['ComponentNames ' '"Sx", ' '"Sy", ' '"Sxy", ' '"Sz", '  '"Syz
    ", ' '"Sxz" ' '\n'];
fprintf(fid,s);

fprintf(fid,['Values \n']);
for i = 1 : nelem
   str = tfield(i,:);
%    str(2) = tfield(2,i);
%    str(3) = tfield(3,i);
   fprintf(fid,'%6.0f %12.5d %12.5d  %12.5d 0.0  0.0  0.0 \n',i,
       str(:) );%element number|Sx|Sy|Sxy|0|0|0
end
fprintf(fid,['End Values \n']);
fprintf(fid,'# \n');

end
```

Listing 17: gid_write_tensorfield.m

APPENDIX D: Generation of aerodynamic load vector from XFLR5 Data

```
%CALCULATION OF THE BOUNDARY FORCE VECTOR GIVEN THE PRESSURE
%DISTRIBUTION ON A NACA2412 FROM XFLR5

%SPLINE GENERATION FROM XFLR5 DATA

%Excel data file from XFLR5 — Cp distribution
filename='Pressure_distribution_TestRe72M0.2AoA5.xlsx';
xe = xlsread(filename,1,'A1:A100');
cpe = xlsread(filename,1,'B1:B100');
xi = xlsread(filename,2,'A1:A100');
cpi = xlsread(filename,2,'B1:B100');

figure(1);
plot(xe,cpe,'o');
hold on;
CpeSpline = csapi(xe,cpe);
fnplt(CpeSpline);
figure(2);
plot(xi,cpi,'o');
CpiSpline = csapi(xi,cpi);
hold on;
fnplt(CpiSpline);


%VALUES OF CP AT NODES OF THE FINITE ELEMENT MESH

%Excel data file from GID — Boundary Nodes
filename2='WingBoxStructured0.005BoundaryNodes.xlsx';
 xBupper = xlsread(filename2,1,'B1:B84');
 xBlower = xlsread(filename2,2,'B1:B84');
```

```
 yBupper = xlsread(filename2,1,'C1:C84');
 yBlower = xlsread(filename2,2,'C1:C84');
 nodesUpper = xlsread(filename2,1,'A1:A84');
 nodesLower = xlsread(filename2,2,'A1:A84');
%-------------------------------------------------------------------

cpeB=fnval(CpeSpline, xBupper);
cpiB=fnval(CpiSpline, xBlower);

Nodal_Cpressure_upper = [nodesUpper cpeB];
Nodal_Cpressure_lower = [nodesLower cpiB];

%PRESSURE DISTRIBUTION

%Cp = (p-pinf)/(0.5*rho*V^2)

%p = pressure on the airfoil
pinf = 101325; %pinf: freestream pressure [Pa]
rho=1.225; %rho: density of air [kg/m3]
V = 64.82;%V: aerodynamic velocity [m/s]

peB = pinf + 0.5*rho*V^2*cpeB;
piB = pinf + 0.5*rho*V^2*cpiB;

Nodal_pressure_upper = [nodesUpper peB];
Nodal_pressure_lower = [nodesLower piB];

%LINEAR INTERPOLATION OF PRESSURES FOR EACH ELEMENT

coordinates_upper = [nodesUpper xBupper yBupper];
coordinates_lower = [nodesLower xBlower yBlower];

localnode1_upper = xlsread(filename2,1,'A1:A83');
localnode2_upper = xlsread(filename2,1,'A2:A84');
localnode1_lower = xlsread(filename2,2,'A1:A83');
localnode2_lower = xlsread(filename2,2,'A2:A84');
%------------------------------------------------------------------

connectivity_upper = [localnode1_upper localnode2_upper];
connectivity_lower = [localnode1_lower localnode2_lower];

%Element distance

%Extrados
xau = xlsread(filename2,1,'B1:B83');
yau = xlsread(filename2,1,'C1:C83');
xbu = xlsread(filename2,1,'B2:B84');
```

```
ybu = xlsread(filename2,1,'C2:C84');

%Intrados
xal = xlsread(filename2,2,'B1:B83');
yal = xlsread(filename2,2,'C1:C83');
xbl = xlsread(filename2,2,'B2:B84');
ybl = xlsread(filename2,2,'C2:C84');

dupper = sqrt((xau-xbu).^2 + (yau-ybu).^2);
dlower = sqrt((xal-xbl).^2 + (yal-ybl).^2);

%-----------------------------------------------------------------

%Element angle
alpha_upper = atan(abs(ybu-yau)./abs(xbu-xau));
alpha_lower = atan(abs(ybl-yal)./abs(xbl-xal));
nelem_upper = size(connectivity_upper,1);
nelem_lower = size(connectivity_lower,1);

for e=1:nelem_upper
if (ybu(e)-yau(e))*(xbu(e)-xau(e)) > 0
    alpha_upper(e) = - alpha_upper(e);
end
end

for e=1:nelem_lower
 if (ybl(e)-yal(e))*(xbl(e)-xal(e)) > 0
     alpha_lower(e) = -alpha_lower(e);
 end
end

%ELEMENTAL FORCE VECTOR IN GLOBAL COORDINATES

%Distributed load linearly interpolated between two boundary
%nodes

nodesU = size(nodesUpper,1);

%Upper pressure force points downwards (towards the airfoil)
%w.r.t. the global coordinate system, for this reason a
%- sign is added to Nodal_pressure_upper

pressure_element_upper = zeros(nelem_upper,2);

pressure_element_upper(:,1) = -Nodal_pressure_upper(1:nodesU-1,2)
    ;
pressure_element_upper(:,2) = -Nodal_pressure_upper(2:nodesU,2);
```

```
Fboundel_upper = zeros(4,nelem_upper);

for e = 1:nelem_upper
    Fboundel_upper(:,e) = dupper(e)/6*[(2*pressure_element_upper(
        e,1)+pressure_element_upper(e,2))*sin(alpha_upper(e));
        (2*pressure_element_upper(e,1)+pressure_element_upper(e
        ,2))*cos(alpha_upper(e)); (2*pressure_element_upper(e,2)+
        pressure_element_upper(e,1))*sin(alpha_upper(e)); (2*
        pressure_element_upper(e,2)+pressure_element_upper(e,1))*
        cos(alpha_upper(e))];
end


nodesL = size(nodesLower,1);

pressure_element_lower = zeros(nelem_lower,2);

pressure_element_lower(:,1) = Nodal_pressure_lower(1:nodesL-1,2);
pressure_element_lower(:,2) = Nodal_pressure_lower(2:nodesL,2);

Fboundel_lower = zeros(4,nelem_lower);

for e = 1:nelem_lower
    Fboundel_lower(:,e) = dlower(e)/6*[(2*pressure_element_lower(
        e,1)+pressure_element_lower(e,2))*sin(alpha_lower(e));
        (2*pressure_element_lower(e,1)+pressure_element_lower(e
        ,2))*cos(alpha_lower(e)); (2*pressure_element_lower(e,2)+
        pressure_element_lower(e,1))*sin(alpha_lower(e)); (2*
        pressure_element_lower(e,2)+pressure_element_lower(e,1))*
        cos(alpha_lower(e))];
end

%ASSEMBLY OF THE BOUNDARY FORCE VECTOR

nnodes=2100;%Number of total nodes of the wng rib section
dofs = 2*nnodes;

Fboundary = zeros(dofs,1);

%Extrados Nodes
for e=1:nelem_upper
    for a=1:2
        for i=1:2
            r=2*(a-1)+i;
            A=connectivity_upper(e,a);
            p=2*(A-1)+i;
```

```
            Fboundary(p) = Fboundary(p) + Fboundel_upper(r,e);

        end
    end
end

%Intrados Nodes
for e=1:nelem_lower
    for a=1:2
        for i=1:2
            s=2*(a-1)+i;
            B = connectivity_lower(e,a);
            q=2*(B-1)+i;
            Fboundary(q) = Fboundary(q) + Fboundel_lower(s,e);
        end
    end
end

%Write the global load vector in an Excel file
filename = 'BoundaryLoadsStructured0.005.xlsx';
xlswrite(filename,Fboundary)
```

Listing 18: PressureDistribution.m

APPENDIX E: Wing Rib optimized designs

0.4 Wing Box

0.4.1 Wing box test designs

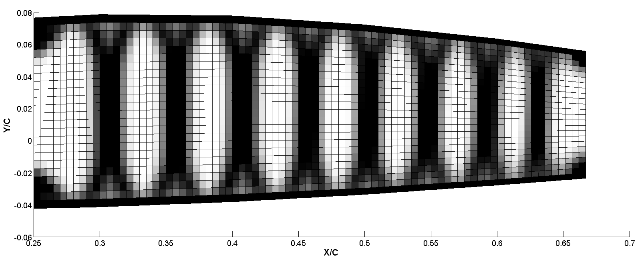


Figure 7: Wing Box optimized design at an AoA=5° with Rmin=0.005.

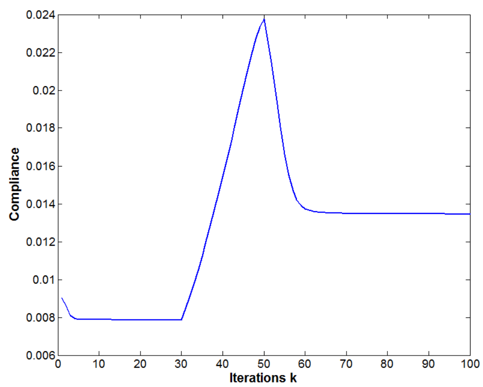


Figure 8: Wing Box convergence at an AoA=5° with Rmin=0.005. Minimum compliance: 0.01344.

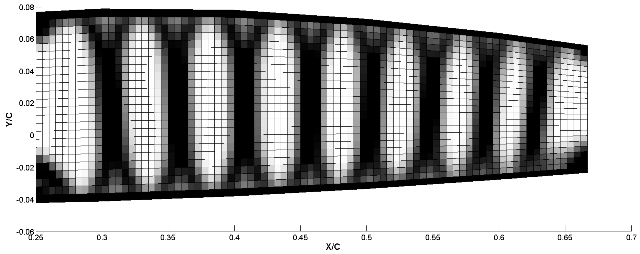


Figure 9: Wing Box final design at an AoA=18° with Rmin=0.005.

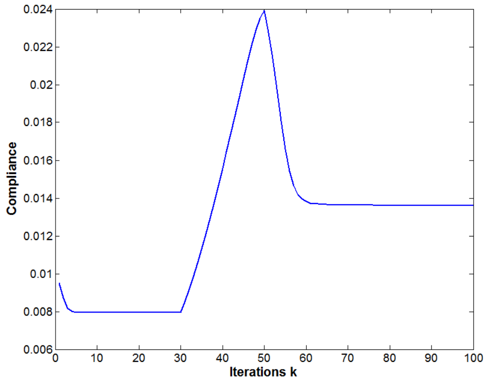


Figure 10: Wing Box convergence at an AoA=18° with Rmin=0.005. Minimum compliance: 0.01360.

0.4.2 Stresses on the final design

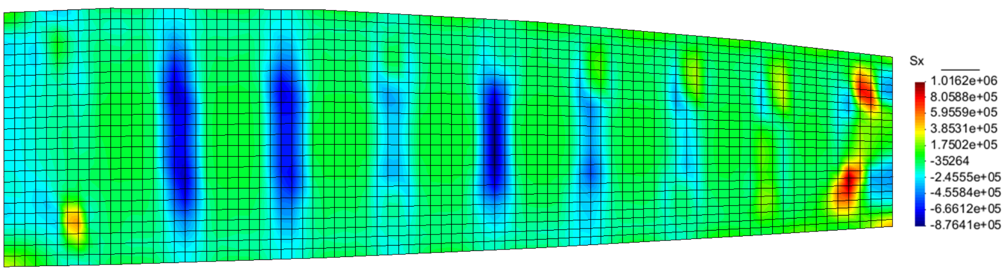


Figure 11: X-Stress distribution on the final wing box design at an AoA=18° with Rmin=0.005.

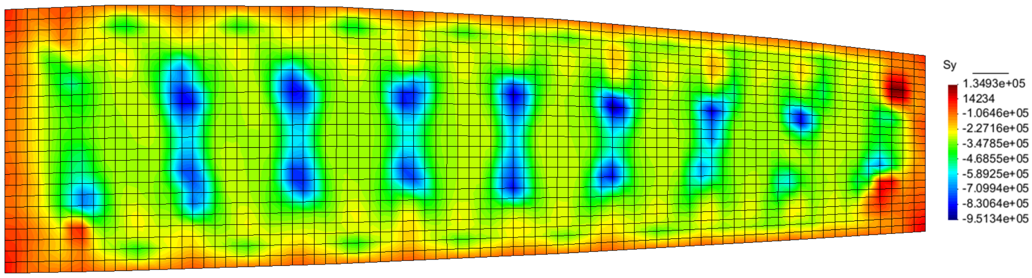


Figure 12: Y-Stress distribution on the final wing box design at an AoA=18° with Rmin=0.005.

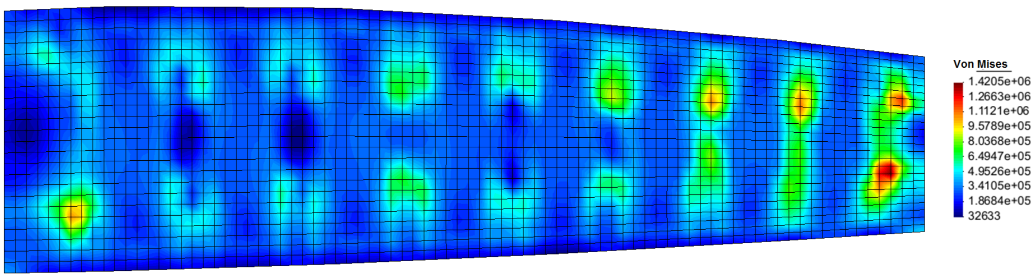


Figure 13: Von Mises stress distribution on the final wing box design at an AoA=18° with Rmin=0.005.

0.5 Trailing Edge

0.5.1 Trailing edge test designs

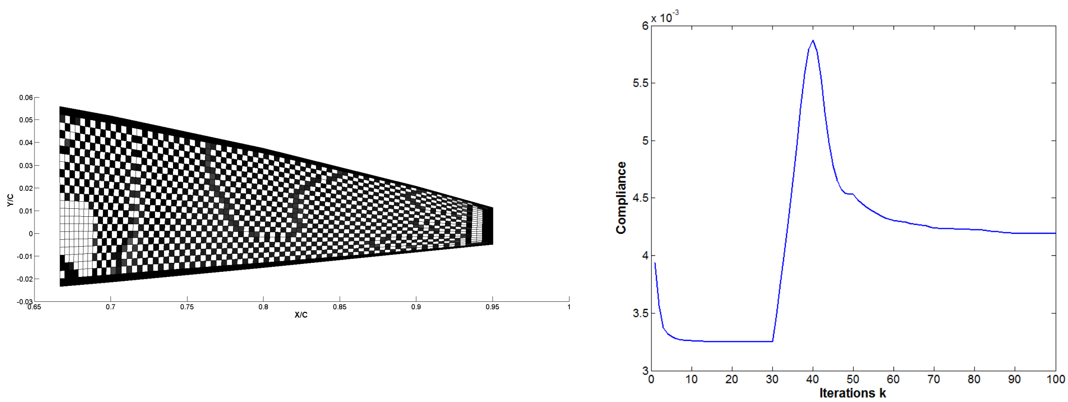


Figure 14: Trailing edge optimized design at an AoA=5° with Rmin=0.001. Minimum compliance: 4.19186e-3.

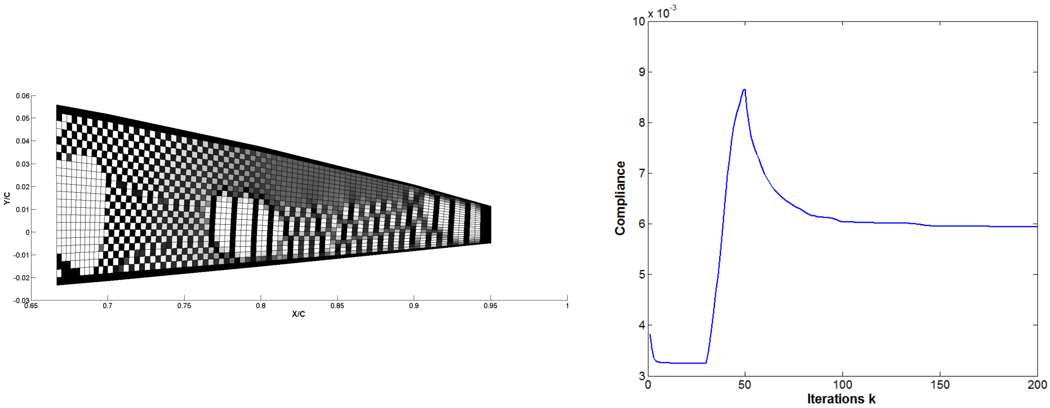


Figure 15: Trailing edge optimized design at an AoA=5° with Rmin=0.003. Minimum compliance: 5.93859e-3.

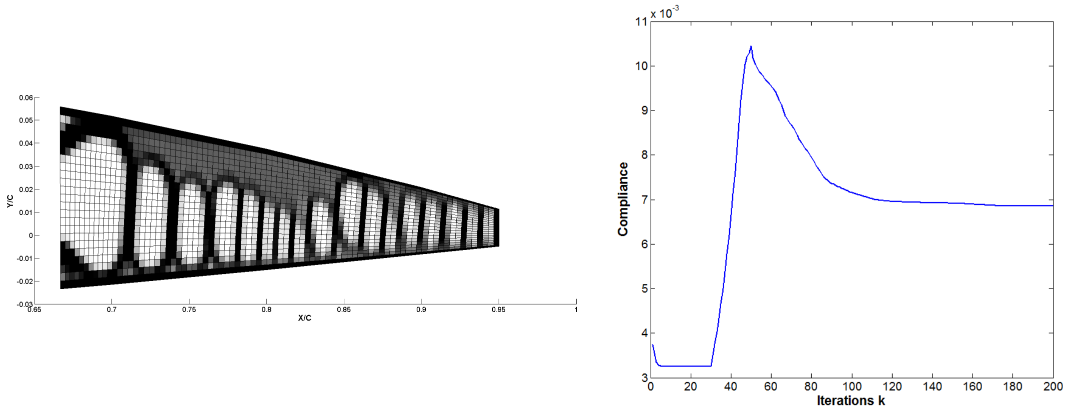


Figure 16: Trailing edge optimized design at an AoA=5° with Rmin=0.004. Minimum compliance: 6.85090e-3.

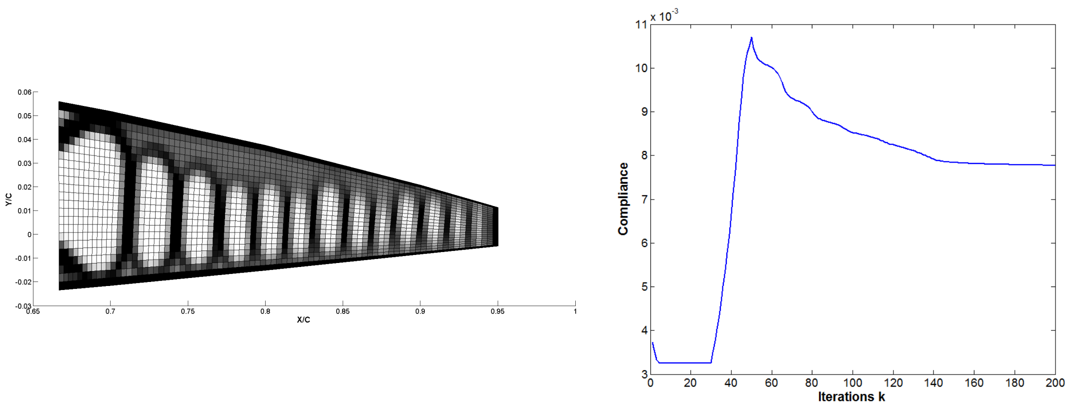


Figure 17: Trailing edge optimized design at an AoA=5° with Rmin=0.005. Minimum compliance: 6.85090e-3.

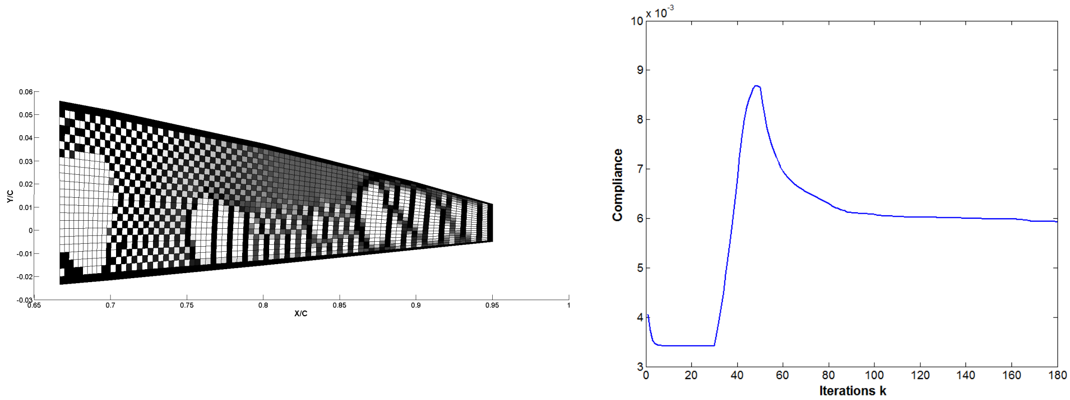


Figure 18: Trailing edge optimized design at an AoA=18° with Rmin=0.003. Minimum compliance: (5.93204e-3).

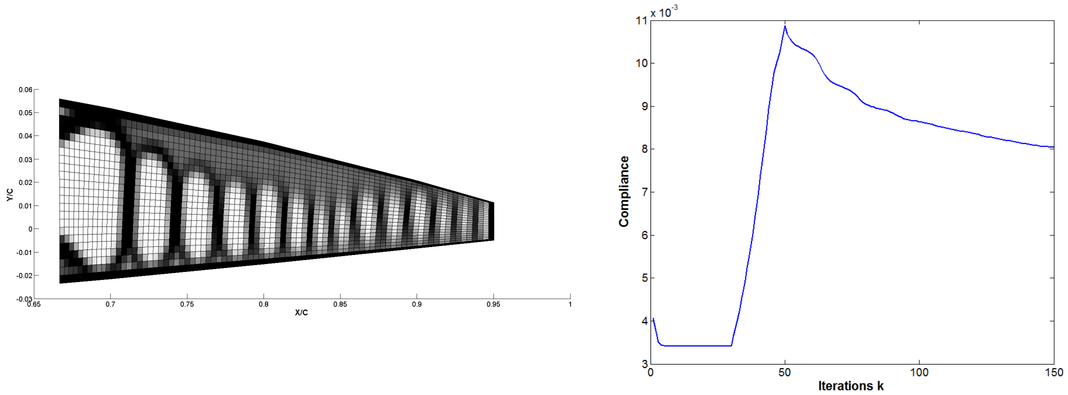


Figure 19: Trailing edge optimized design at an AoA=18° with Rmin=0.005. Minimum compliance: 8.04249e-3.

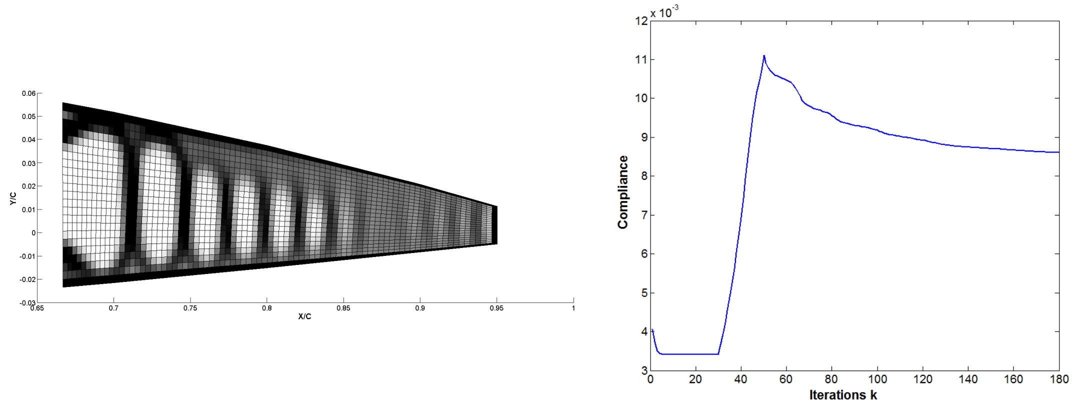


Figure 20: Trailing edge optimized design at an AoA=18° with Rmin=0.006. Minimum compliance: 8.60443e-3.

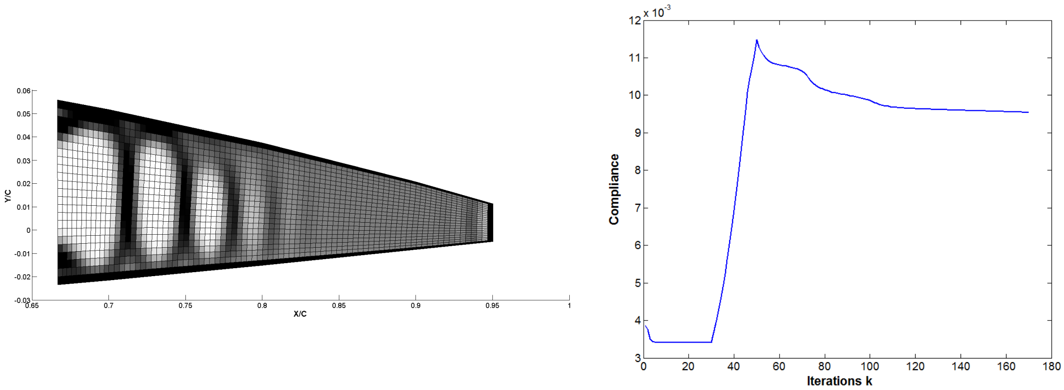


Figure 21: Trailing edge optimized design at an AoA=18° with Rmin=0.008. Minimum compliance: 9.54270e-3.

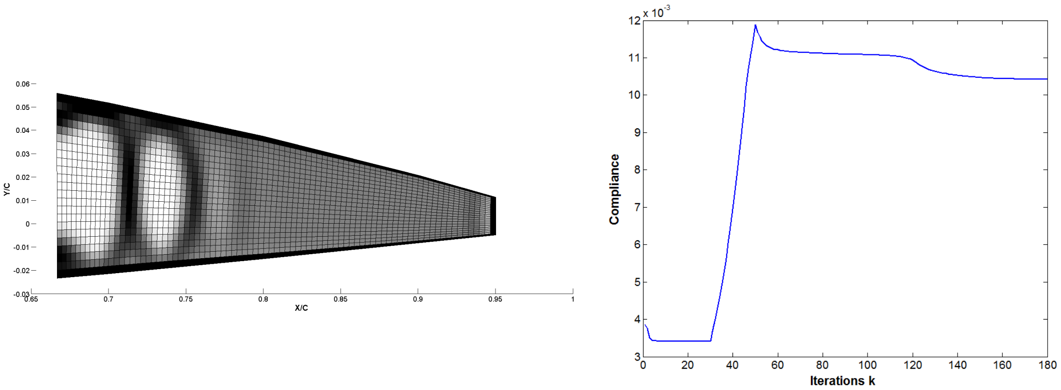


Figure 22: Trailing edge optimized design at an AoA=18° with Rmin=0.01. Minimum compliance: 10.41890e-3.

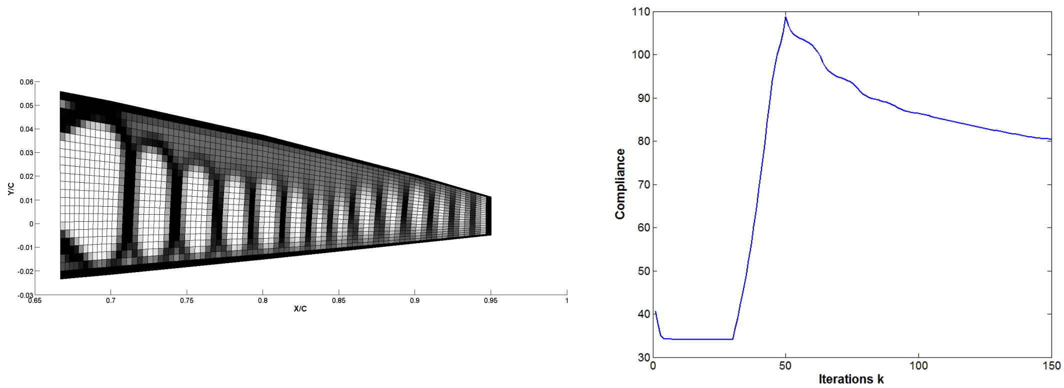


Figure 23: Trailing edge optimized design at an AoA=18° with Rmin=0.005 and an augmented load by a factor of 100. Minimum compliance: 80.43618.

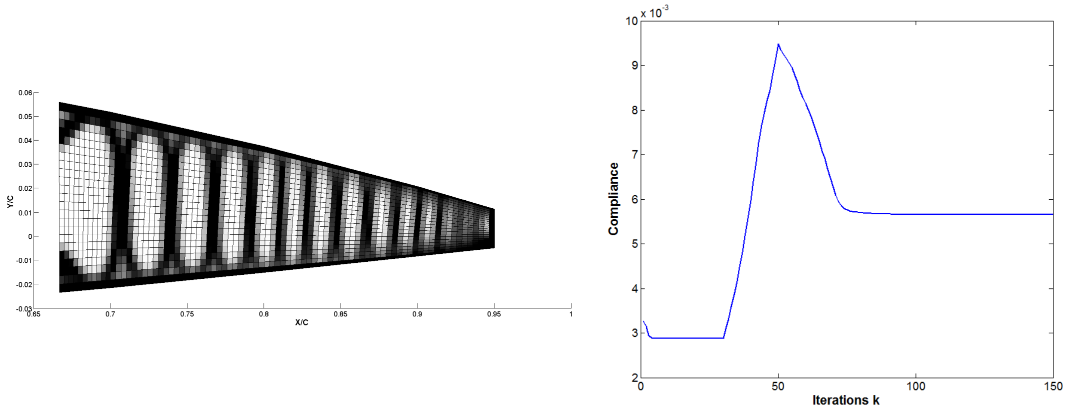


Figure 24: Trailing edge optimized design at an AoA=18° with Rmin=0.005 and the two ends clamped. Minimum compliance: 5.65248e-3.

0.5.2 Stresses on the final design

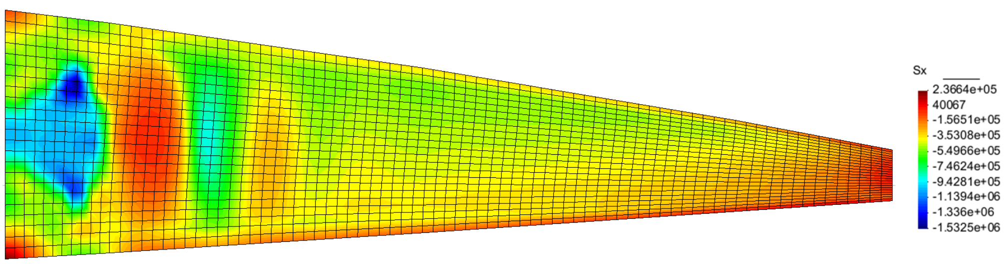


Figure 25: X-Stress distribution on the final leading edge design at an AoA=18° with Rmin=0.005.

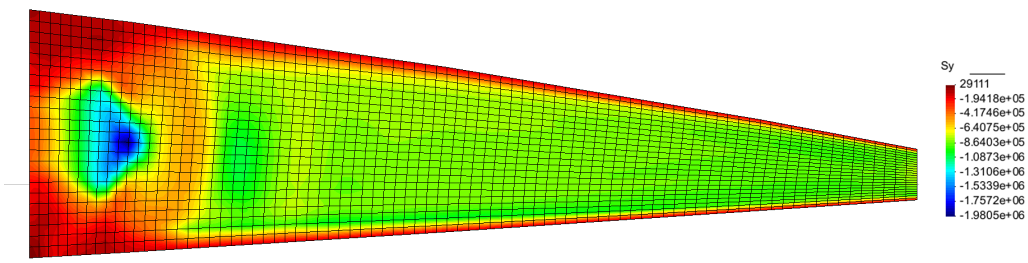


Figure 26: Y-Stress distribution on the final leading edge design at an AoA=18° with Rmin=0.005.

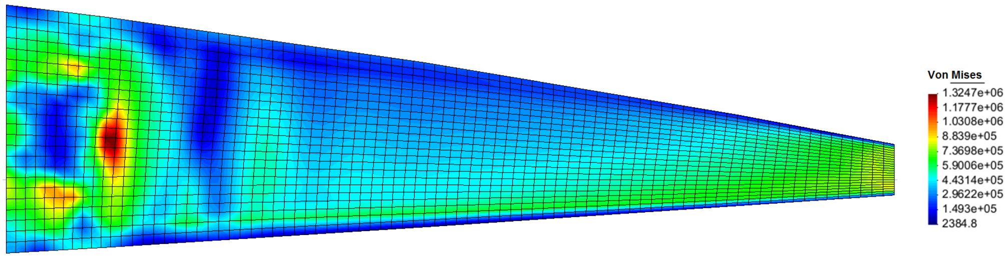


Figure 27: Von Mises stress distribution on the final leading edge design at an AoA=18° with Rmin=0.005.

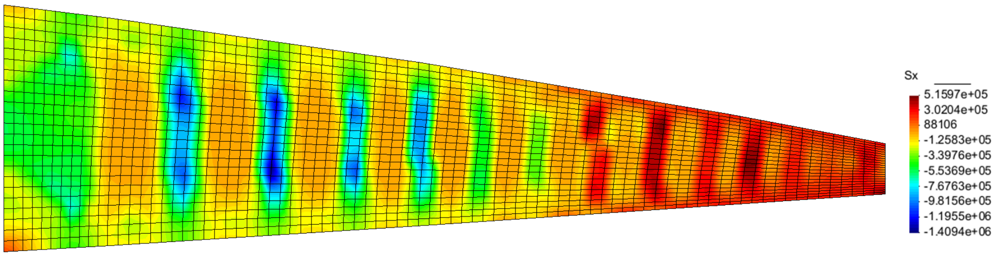


Figure 28: X-Stress distribution on the final leading edge design at an AoA=18° with Rmin=0.005 and clamped ends.

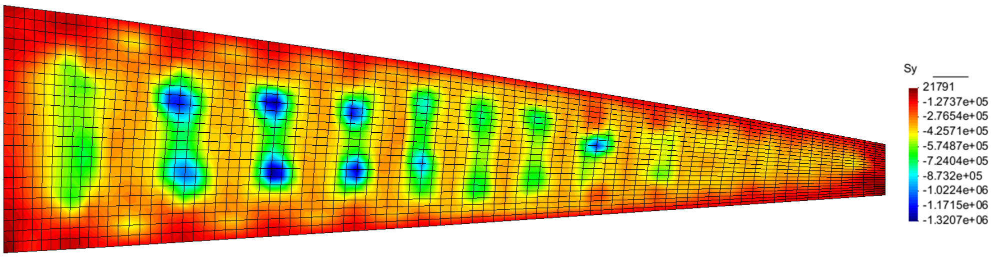


Figure 29: Y-Stress distribution on the final leading edge design at an AoA=18° with Rmin=0.005 and clamped ends.

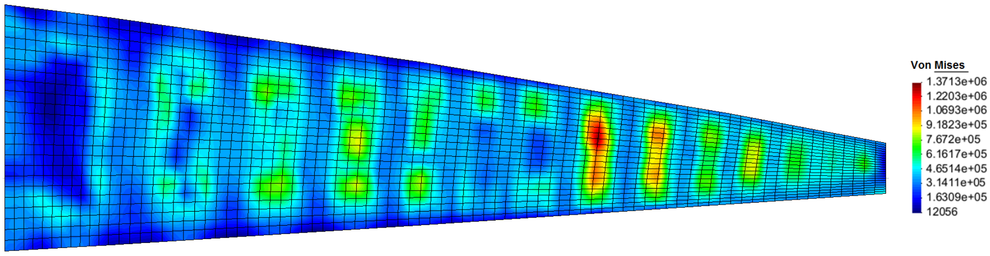


Figure 30: Von Mises stress distribution on the final leading edge design at an AoA=18° with Rmin=0.005 and clamped ends.

0.6 Leading Edge

0.6.1 Leading edge test designs

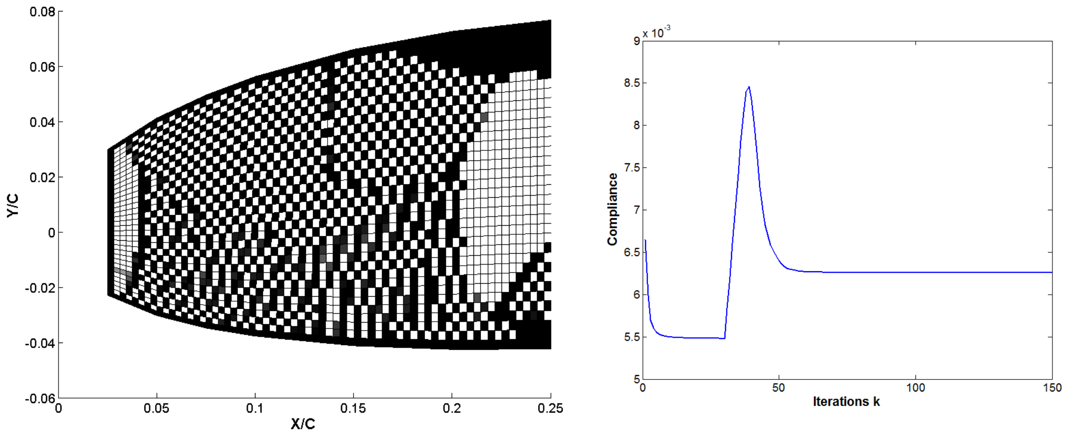


Figure 31: Leading edge optimized design at an AoA=18° with Rmin=0.001. Minimum compliance: 6.25761e-3.

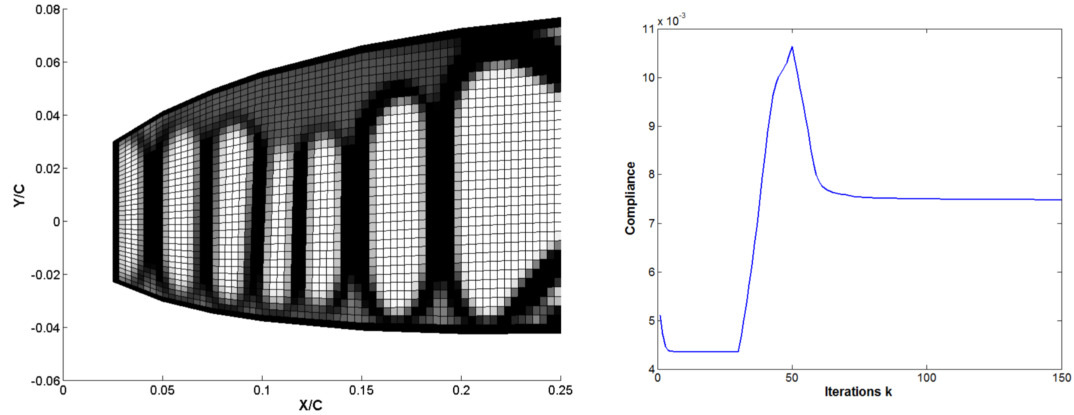


Figure 32: Leading edge optimized design at an AoA=5° with Rmin=0.005. Minimum compliance: 7.47930e-3.

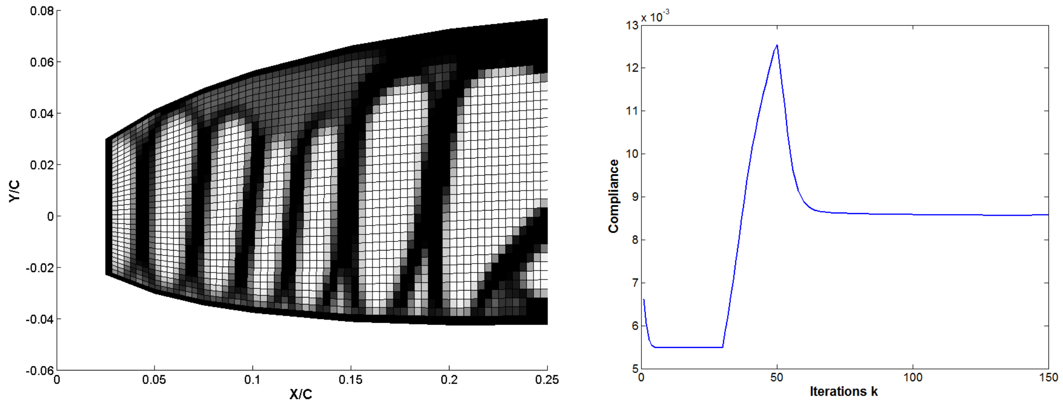


Figure 33: Leading edge optimized design at an AoA=18° with Rmin=0.005. Minimum compliance: 8.57423e-3.

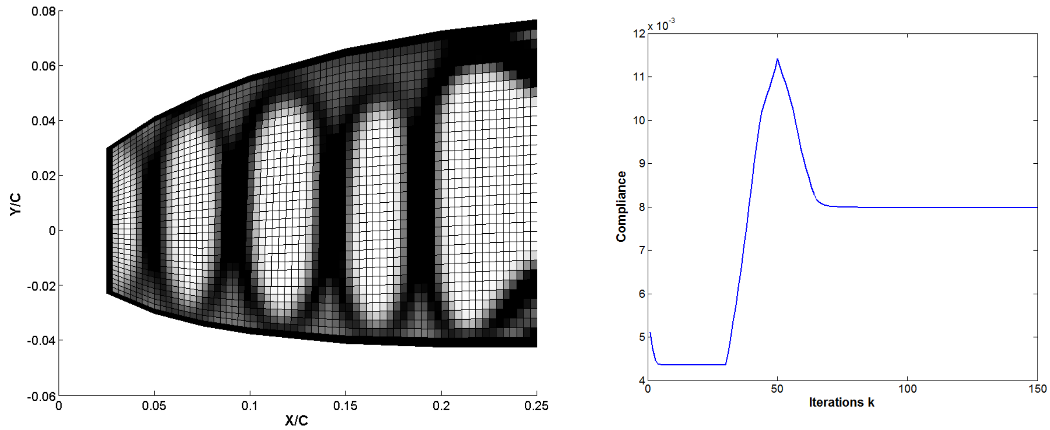


Figure 34: Leading edge optimized design at an AoA=5° with Rmin=0.008. Minimum compliance: 7.98401e-3.

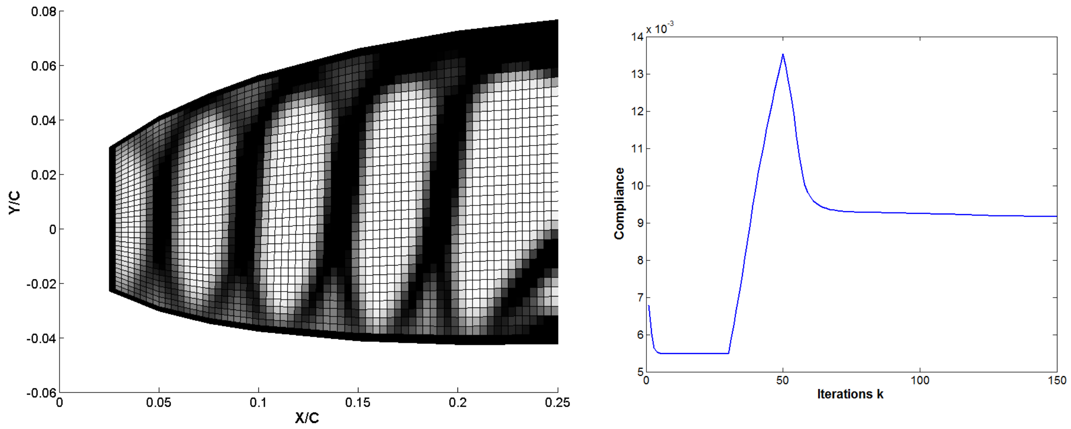


Figure 35: Leading edge optimized design at an AoA=18° with Rmin=0.008. Minimum compliance: 9.17093e-3.

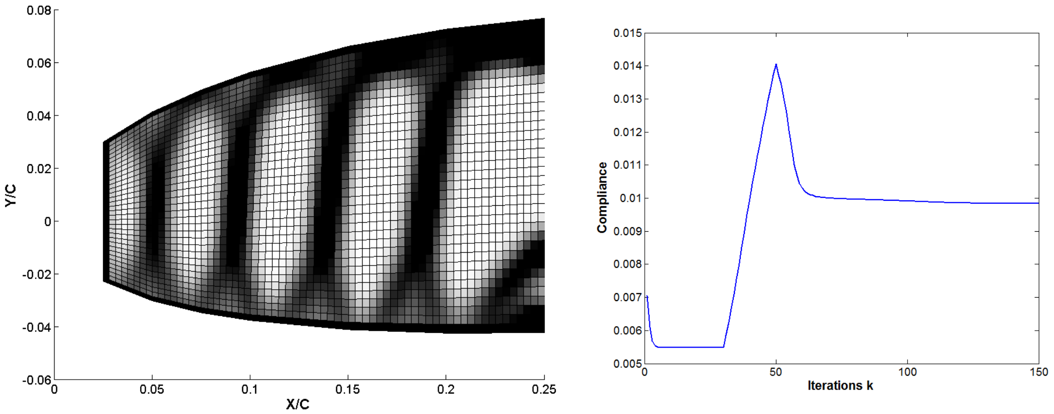


Figure 36: Leading edge optimized design at an AoA=18° with Rmin=0.01. Minimum compliance: 9.83919e-3.

0.6.2 Stresses on the final design

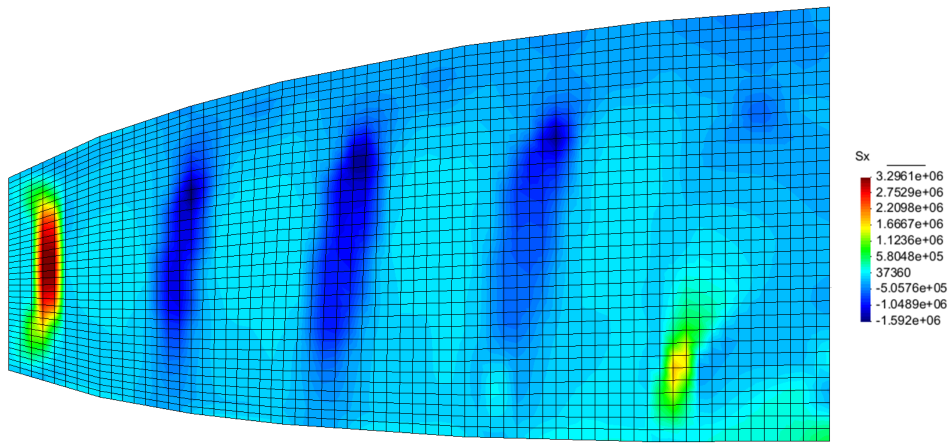


Figure 37: X-Stress distribution on the final leading edge design at an AoA=18° with Rmin=0.008.

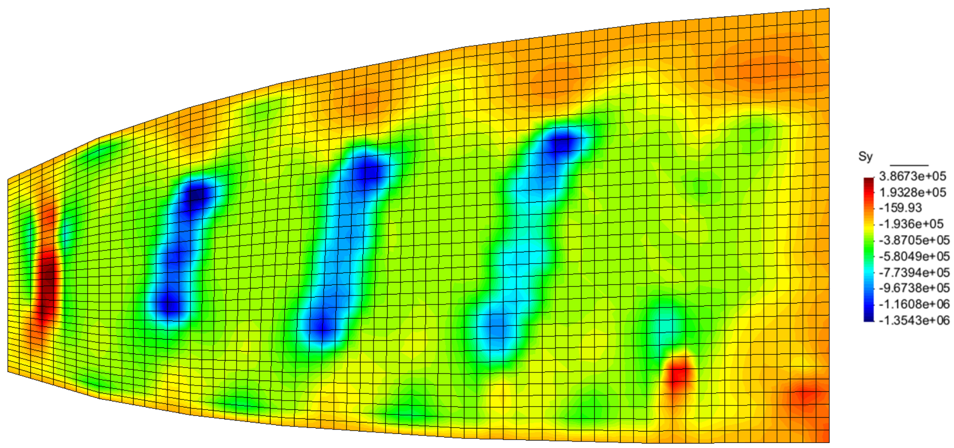


Figure 38: Y-Stress distribution on the final leading edge design at an AoA=18° with Rmin=0.008.

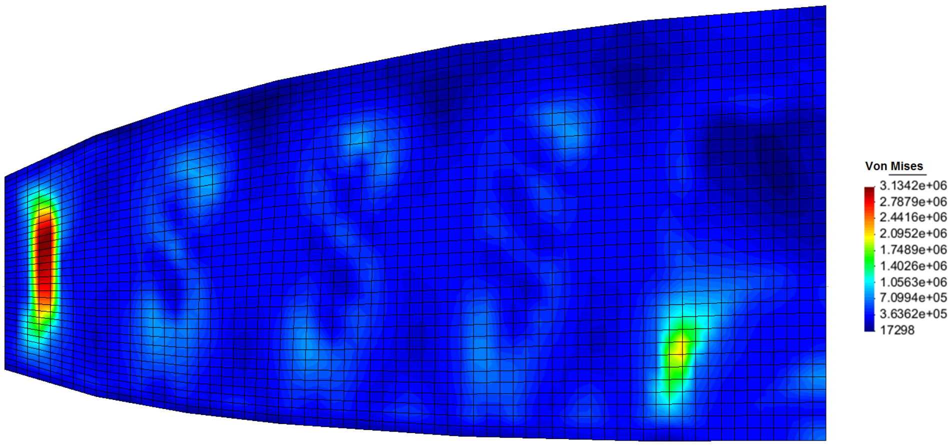


Figure 39: Von Mises stress distribution on the final leading edge design at an AoA=18° with Rmin=0.008.

APPENDIX F: Cessna 172s Skyhawk Technical Sheet

SKYHAWK

MODEL 172S

Specification & Description

Initial ____________________________

Exhibit “A”

December 2012
Beginning with Serial # 172S11284

Cessna
A Textron Company

1. GENERAL DESCRIPTION

All information herein applies to the Skyhawk (Model 172S). The Skyhawk aircraft is an all-metal, single-engine piston, high-wing monoplane with a four-person seating capacity including a crew of one or two. Suitable allowance for luggage is provided.

1.1 Certification

The Model 172S is certified to the requirements of U.S. FAA Federal Aviation Regulation Part 23 through amendment 23-6, including day, night, VFR and IFR.

1.2 Approximate Dimensions

Overall Height .... 8 ft 11in (2.72 m)
Overall Length .... 27 ft 2 in (8.28 m)

Wing

Span (overall) .... 36 ft 1 in (11.00 m)
Area .... 174 sq ft (16.2 sq m)

Cabin

Height (max) .... 48 in (1.22 m)
Width (trim to trim) .... 39.5 in (1.00 m)
Length (firewall to aft baggage bulkhead) .... 142 in (3.61 m)

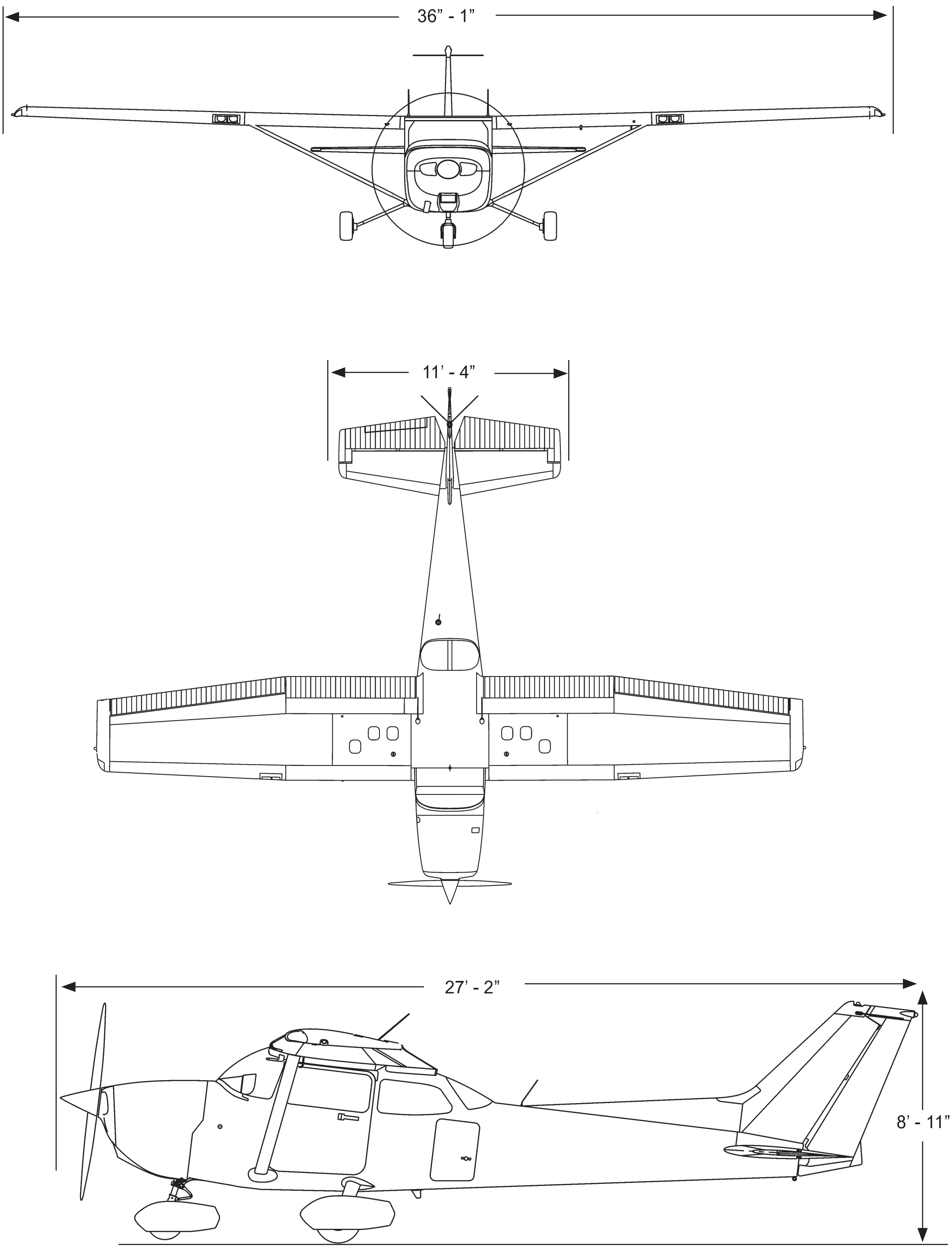
Cabin Door

Height (front) .... 40.5 in (1.03 m)
Height (rear) .... 39 in (0.99 m)
Width (top) .... 32.5 in (0.83 m)
Width (bottom) .... 37 in (0.94 m)

Baggage Door

Height (front) .... 22 in (0.56 m)
Height (rear) .... 21 in (0.53 m)
Width .... 15.3 in (0.39 m)

1. GENERAL DESCRIPTION (Continued)



Note: Optional Wheel Fairings Shown

FIGURE I — SKYHAWK EXTERIOR DIMENSIONS

1. GENERAL DESCRIPTION (Continued)

1.3 Design Weight and Capacities

Ramp Weight
- Normal Category .... 2,558 lbs (1,160 kg)
- Utility Category .... 2,208 lbs (1,002 kg)

Takeoff Weight
- Normal Category .... 2,550 lbs (1,157 kg)
- Utility Category .... 2,200 lbs (998 kg)

Landing Weight
- Normal Category .... 2,550 lbs (1,157 kg)

Standard Empty Weight........ 1,641 lbs (744 kg)

Maximum Useful Load
- Normal Category .... 917 lbs (415 kg)
- Utility Category .... 567 lbs (257 kg)

Baggage Allowance
- Normal Category .... 120 lbs (54 kg)

Fuel Capacity
- Total Capacity .... 56 gal (212 L)
- Total Useable .... 53 gal (200.6 L)
- Total Capacity each Tank .... 28 gal (106 L)
- Total Useable Capacity each Tank .... 26.5 gal (100.3 L)

Oil Capacity
- Sump .... 8 qts (7.6 L)
- Total Capacity .... 9 qts (8.5 L)

NOTES

1. Standard empty weight based upon:
   a) 0.6-mil primer on all details, 0.6-mil primer on all exterior surfaces and 2.0-mil paint on all exterior surfaces.
2. Total oil capacity is with 8 qts. in sump and 1 qt. in oil filter

2. PERFORMANCE

All estimated performance data are based on airplane weights at 2,550 pounds; standard atmospheric conditions; level, hardsurface, dry runways; and no wind. They are calculated values derived from flight tests conducted by Cessna Aircraft Company under carefully documented conditions and will vary with individual airplanes, pilots, and numerous other factors affecting flight performance.

Service Ceiling	14,000 ft
Takeoff Distance S.L. (Ground Roll)	960 ft
Takeoff Distance S.L. (To Clear 50 ft. Obstacle)	1,630 ft
Maximum Climb Rate S.L.	730 fpm
Maximum Speed S.L.	126 kts / 145 mph
Maximum Range and Endurance	638 nm / 6.72 hrs
Cruise Speed (75% pwr at 8,500 ft)	124 kts / 143 mph
Cruise Range and Endurance (75% pwr at 8,500 ft)	518 nm / 4.26 hrs
Landing Distance (Ground Roll)	575 ft
Landing Distance (To Clear 50 ft. Obstacle)	1,335 ft

3. POWERPLANT & ACCESSORIES

- Lycoming IO-360-L2A Engine
- 180 HP @ 2700 RPM
- Certified for 100LL & 100 Fuel
- Fuel Injection System
- Tubular Steel Engine Mount
- Dynafocal Rear Mount
- Engine Driven Vacuum Pump
- Automatic Alternate Engine Air
- Oil Cooler
- Shock Mounted Cowling
- Induction Air Filter
- Full Flow Oil Filter
- Throttle Control
- Vernier Mixture Control
- Dual Ignition System, Shielded Magneto
- Engine Exhaust Muffler
- McCauley Fixed Pitch 2 – Blade Metal Propeller
- Propeller Spinner, Polished
- Electric Starter